Pythus+

# TensorFlow For Social Good

## Agrifood


by, Jonathan Rivera,

Lorenzo Stewart,

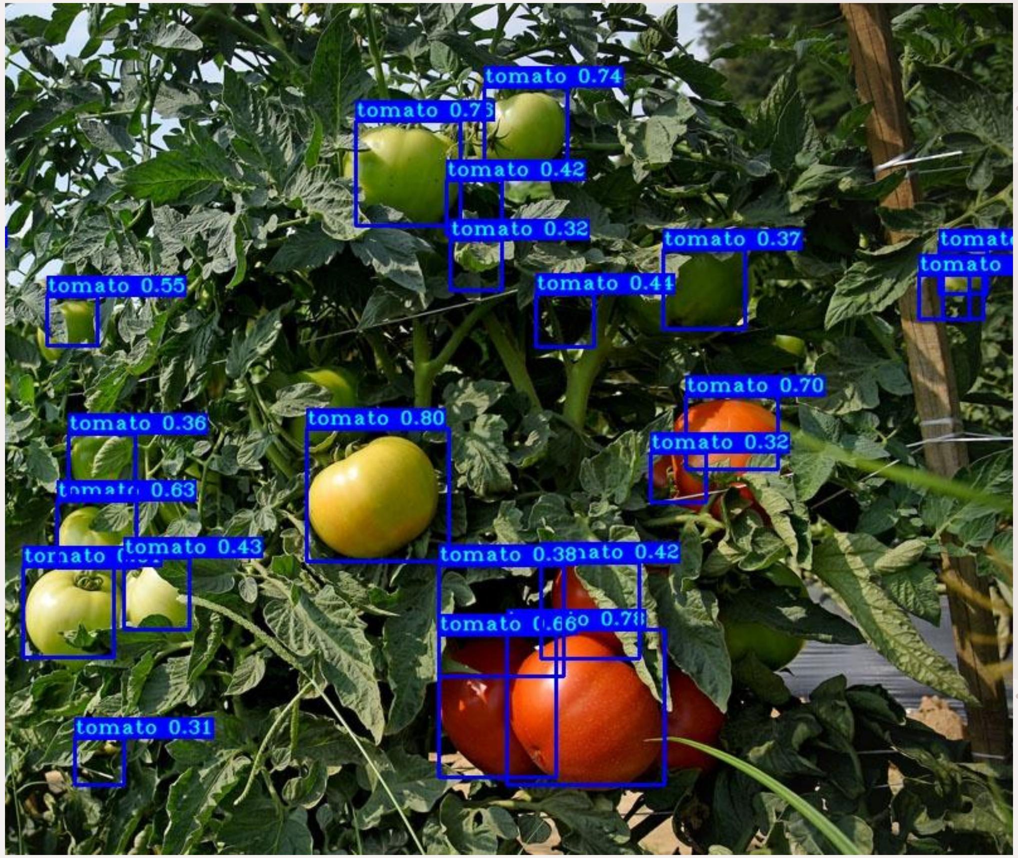
Edison Tran


# Problem

- While brainstorming about problems that we face in the central valley, we decided to pick one based on agriculture.
- The problem specifically that we are trying to solve is food production.
- Worldwide diseases can lay waste to 40% of crops. A massive problem, to an ever expanding increasing population.
- Tomatoes are the 7th most popular crop in California and the state is the leader in vegetable production in the U.S.
- We came up with a solution that could help reduce the loss of crops, benefiting the hungry, the farmers, and everyone else.

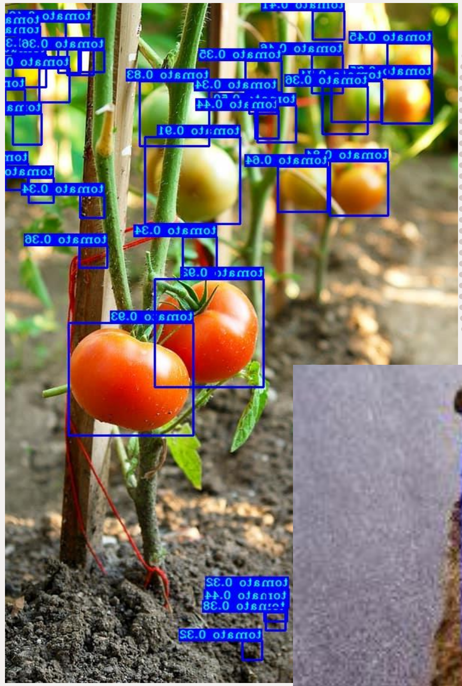
# Solution

- Using machine learning, more specifically object detection. We have created a program that can detect diseases on tomatoes.
- Using YOLO, we can identify diseases fast and accurately, allowing the resulting infections to be located and treated immediately.
- This will increase the efficiency of the “pipeline” of identifying, treating, and producing larger crop yields, creating more food while turning a larger profit.



# Implementation

- Our solution uses the YOLOv4 model, models from “**TensorFlow-2.x-YOLOv3**”, models from the TensorFlow Model Zoo, TensorFlow 2 framework, Darknet framework, virtual environments and image labeling software.
- We gathered and created datasets on common tomato diseases like verticillium wilt, blossom end rot, and early/late blight.



```
# YOLO options
YOLO_TYPE                   = "yolov4" # yolov4 or yolov3
YOLO_FRAMEWORK              = "tf" # "tf" or "trt"
YOLO_V3_WEIGHTS             = "model_data/yolov3.weights"
YOLO_V4_WEIGHTS             = "model_data/yolov4.weights"
YOLO_V3_TINY_WEIGHTS        = "model_data/yolov3-tiny.weights"
YOLO_V4_TINY_WEIGHTS        = "model_data/yolov4-tiny.weights"
YOLO_TRT_QUANTIZE_MODE      = "INT8" # INT8, FP16, FP32
YOLO_CUSTOM_WEIGHTS         = True # "checkpoints/yolov3_custom" # used in evaluat
                            # YOLO_CUSTOM_WEIGHTS also used with TensorRT and cust
YOLO_COCO_CLASSES           = "model_data/coco/coco.names"
YOLO_STRIDES                = [8, 16, 32]
YOLO_IOU_LOSS_THRESH        = 0.5
YOLO_ANCHOR_PER_SCALE       = 3
YOLO_MAX_BBOX_PER_SCALE     = 100
YOLO_INPUT_SIZE             = 416
if YOLO_TYPE                == "yolov4":
    YOLO_ANCHORS            = [[[12,  16], [19,   36], [40,   28]],
                               [[36,  75], [76,   55], [72,  146]],
                               [[142,110], [192, 243], [459, 401]]]
if YOLO_TYPE                == "yolov3":
    YOLO_ANCHORS            = [[[10,  13], [16,   30], [33,   23]],
                               [[30,  61], [62,   45], [59,  119]],
                               [[116, 90], [156, 198], [373, 326]]]
# Train options
TRAIN_YOLO_TINY             = False
TRAIN_SAVE_BEST_ONLY        = True # saves only best model according validation lo
TRAIN_SAVE_CHECKPOINT       = False # saves all best validated checkpoints in trai
TRAIN_CLASSES               = "datasets/blossom_rot/names.txt"
TRAIN_ANNOT_PATH            = "datasets/blossom_rot/train.txt"
TRAIN_LOGDIR                = "log"
TRAIN_CHECKPOINTS_FOLDER    = "checkpoints"
TRAIN_MODEL_NAME            = f"{YOLO_TYPE}_custom"
TRAIN_LOAD_IMAGES_TO_RAM    = True # With True faster training, but need more RAM
TRAIN_BATCH_SIZE            = 4
TRAIN_INPUT_SIZE            = 416
TRAIN_DATA_AUG              = True
TRAIN_TRANSFER              = True
TRAIN_FROM_CHECKPOINT       = True # "checkpoints/yolov3_custom"
TRAIN_LR_INIT               = 1e-4
TRAIN_LR_END                = 1e-6
TRAIN_WARMUP_EPOCHS         = 2
TRAIN_EPOCHS                = 32
```

# Pythus+

## 1) Code

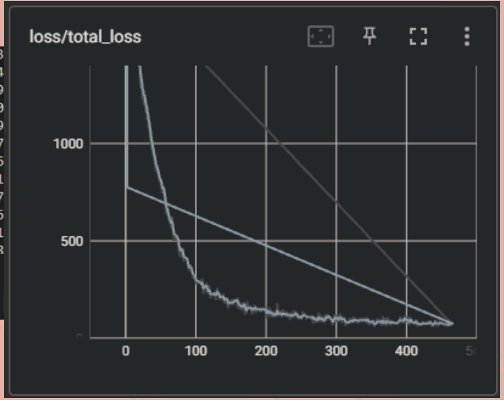
```
# YOLO options
YOLO_TYPE                   = "yolov4" # yolov4 or yolov3
YOLO_FRAMEWORK              = "tf" # "tf" or "trt"
YOLO_V3_WEIGHTS             = "model_data/yolov3.weights"
YOLO_V4_WEIGHTS             = "model_data/yolov4.weights"
YOLO_V3_TINY_WEIGHTS        = "model_data/yolov3-tiny.weights"
YOLO_V4_TINY_WEIGHTS        = "model_data/yolov4-tiny.weights"
YOLO_TRT_QUANTIZE_MODE      = "INT8" # INT8, FP16, FP32
YOLO_CUSTOM_WEIGHTS         = True # "checkpoints/yolov3_custom" # used in evaluat
                            # YOLO_CUSTOM_WEIGHTS also used with TensorRT and cust
YOLO_COCO_CLASSES           = "model_data/coco/coco.names"
YOLO_STRIDES                = [8, 16, 32]
YOLO_IOU_LOSS_THRESH        = 0.5
YOLO_ANCHOR_PER_SCALE       = 3
YOLO_MAX_BBOX_PER_SCALE     = 100
YOLO_INPUT_SIZE             = 416
if YOLO_TYPE                == "yolov4":
    YOLO_ANCHORS            = [[[12,  16], [19,   36], [40,   28]],
                               [[36,  75], [76,   55], [72,  146]],
                               [[142,110], [192, 243], [459, 401]]]
if YOLO_TYPE                == "yolov3":
    YOLO_ANCHORS            = [[[10,  13], [16,   30], [33,   23]],
                               [[30,  61], [62,   45], [59,  119]],
                               [[116, 90], [156, 198], [373, 326]]]
# Train options
TRAIN_YOLO_TINY             = False
TRAIN_SAVE_BEST_ONLY        = True # saves only best model according validation lo
TRAIN_SAVE_CHECKPOINT       = False # saves all best validated checkpoints in trai
TRAIN_CLASSES               = "datasets/blossom_rot/names.txt"
TRAIN_ANNOT_PATH            = "datasets/blossom_rot/train.txt"
TRAIN_LOGDIR                = "log"
TRAIN_CHECKPOINTS_FOLDER    = "checkpoints"
TRAIN_MODEL_NAME            = f"{YOLO_TYPE}_custom"
TRAIN_LOAD_IMAGES_TO_RAM    = True # With True faster training, but need more RAM
TRAIN_BATCH_SIZE            = 4
TRAIN_INPUT_SIZE            = 416
TRAIN_DATA_AUG              = True
TRAIN_TRANSFER              = True
TRAIN_FROM_CHECKPOINT       = True # "checkpoints/yolov3_custom"
TRAIN_LR_INIT               = 1e-4
TRAIN_LR_END                = 1e-6
TRAIN_WARMUP_EPOCHS         = 2
TRAIN_EPOCHS                = 32
```

## 2) Collect Dataset

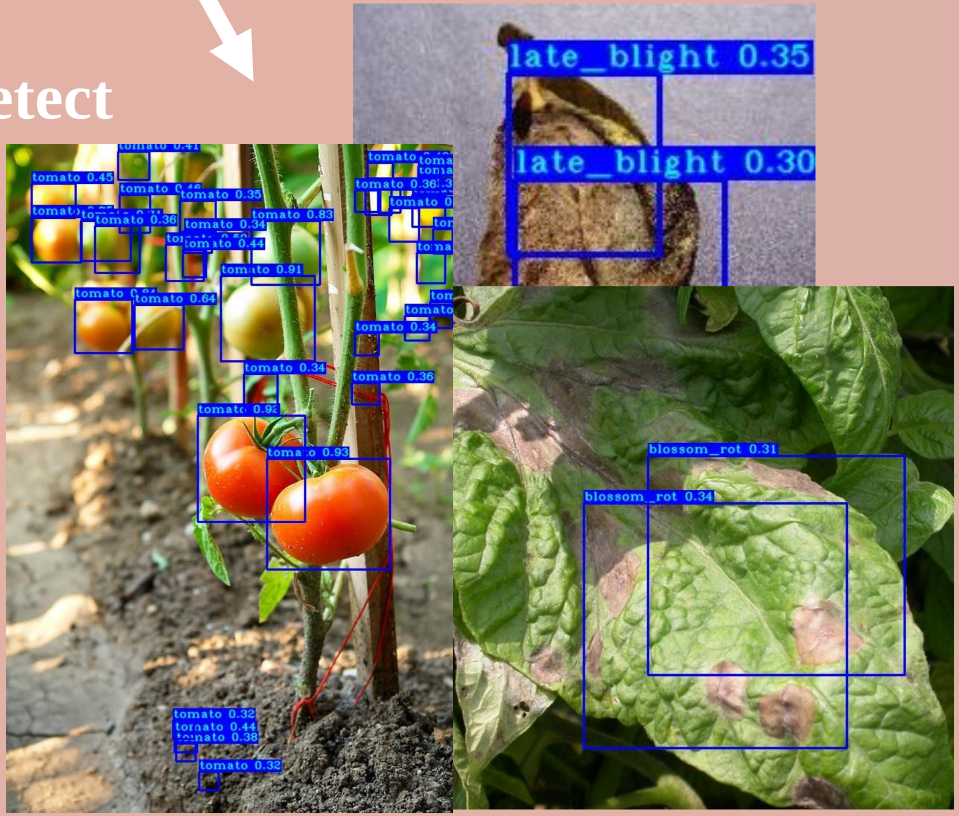
## 3) Label

## 4) Train/Improve

```
epoch:10 step:  111/151, lr:0.000069, giou_loss:   7.25, conf_loss:   8.86, prob_loss:   0.11, total_loss:  16.23
epoch:10 step:  112/151, lr:0.000069, giou_loss:   5.17, conf_loss:   7.99, prob_loss:   0.08, total_loss:  13.24
epoch:10 step:  113/151, lr:0.000069, giou_loss:   4.88, conf_loss:   8.08, prob_loss:   0.03, total_loss:  12.99
epoch:10 step:  114/151, lr:0.000069, giou_loss:   5.26, conf_loss:   5.60, prob_loss:   0.04, total_loss:  10.90
epoch:10 step:  115/151, lr:0.000069, giou_loss:   1.38, conf_loss:   4.90, prob_loss:   0.01, total_loss:   6.29
epoch:10 step:  116/151, lr:0.000069, giou_loss:   4.91, conf_loss:   5.71, prob_loss:   0.05, total_loss:  10.67
epoch:10 step:  117/151, lr:0.000069, giou_loss:   4.21, conf_loss:   5.24, prob_loss:   0.02, total_loss:   9.46
epoch:10 step:  118/151, lr:0.000068, giou_loss:   6.66, conf_loss:   7.00, prob_loss:   0.05, total_loss:  13.71
epoch:10 step:  119/151, lr:0.000068, giou_loss:  12.87, conf_loss:  11.98, prob_loss:   0.22, total_loss:  25.07
epoch:10 step:  120/151, lr:0.000068, giou_loss:   8.56, conf_loss:  10.63, prob_loss:   0.07, total_loss:  19.26
epoch:10 step:  121/151, lr:0.000068, giou_loss:   3.55, conf_loss:   5.74, prob_loss:   0.02, total_loss:   9.31
epoch:10 step:  122/151, lr:0.000068, giou_loss:   4.03, conf_loss:   4.14, prob_loss:   0.01, total_loss:   8.18
epoch:10 step:  123/151, lr:0.000068, giou_loss:   3.30, conf_loss:   5.00, 0.183% = tomato AP
epoch:10 step:  124/151, lr:0.000068, giou_loss:  13.37, conf_loss:   9.40, mAP = 0.183%, 9.01
```




## 5) Detect



# Implementation

Low Epochs = Low MaP

```
epoch: 4 step:  139/151, lr:0.000096, giou_loss:   6.35, conf_loss:  18.76, prob_loss:   0.23, total_loss:  25.33
epoch: 4 step:  140/151, lr:0.000096, giou_loss:  11.87, conf_loss:  19.80, prob_loss:   0.71, total_loss:  32.38
epoch: 4 step:  141/151, lr:0.000096, giou_loss:   4.17, conf_loss:  18.14, prob_loss:   0.25, total_loss:  22.56
epoch: 4 step:  142/151, lr:0.000096, giou_loss:  13.64, conf_loss:  28.24, prob_loss:   0.86, total_loss:  42.73
epoch: 4 step:  143/151, lr:0.000096, giou_loss:   3.76, conf_loss:  19.65, prob_loss:   0.16, total_loss:  23.58
epoch: 4 step:  144/151, lr:0.000096, giou_loss:  13.58, conf_loss:  26.72, prob_loss:   0.68, total_loss:  40.98
epoch: 4 step:  145/151, lr:0.000096, giou_loss:   9.16, conf_loss:  20.39, prob_loss:   0.34, total_loss:  29.89
epoch: 4 step:  146/151, lr:0.000096, giou_loss:   6.30, conf_loss:  16.64, prob_loss:   0.19, total_loss:  23.13
epoch: 4 step:  147/151, lr:0.000096, giou_loss:   9.03, conf_loss:  19.93, prob_loss:   0.39, total_loss:  29.34
epoch: 4 step:  148/151, lr:0.000096, giou_loss:  11.06, conf_loss:  22.51, prob_loss:   0.55, total_loss:  34.13
epoch: 4 step:  149/151, lr:0.000096, giou_loss:  16.10, conf_loss:  28.36, prob_loss:   0.82, total_loss:  45.27
epoch: 4 step:  150/151, lr:0.000096, giou_loss:   5.67, conf_loss:  18.40, prob_loss:   0.17, total_loss:  24.24
epoch: 4 step:    0/151, lr:0.000096, giou_loss:   3.30, conf_loss:  16.85, prob_loss:   0.07, total_loss:  20.23
epoch: 4 step:    1/151, lr:0.000096, giou_loss:   8.71, conf_loss:  18.44, prob_loss:   0.37, total_loss:  27.52
```

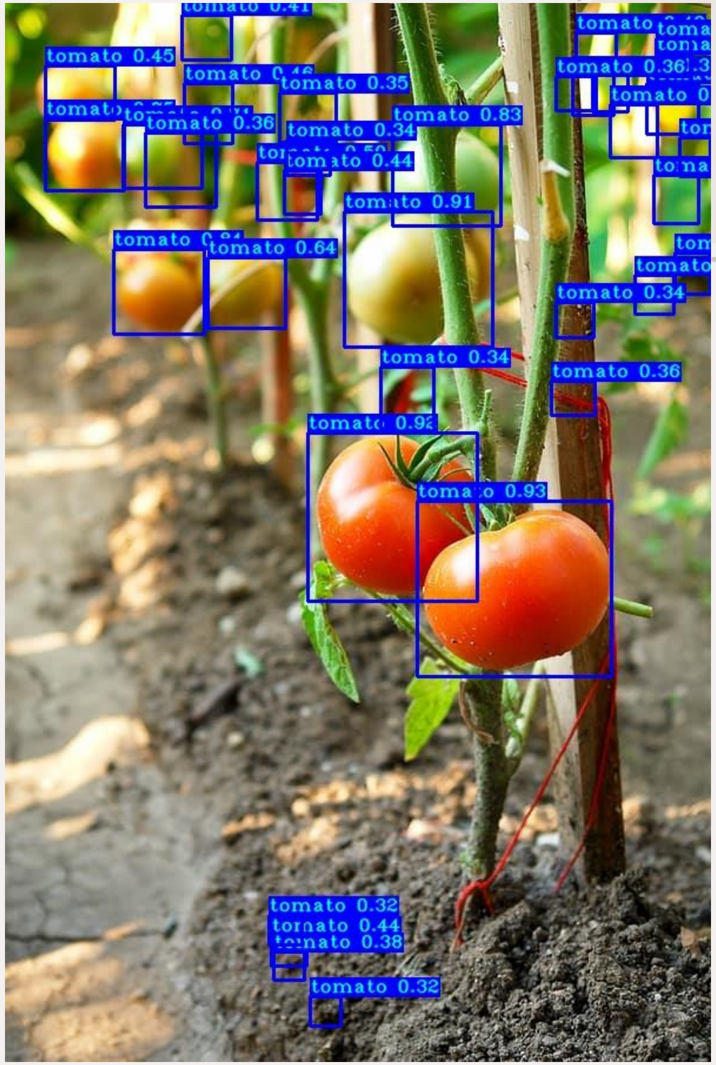
Train with more Epochs (but takes more time)

```
epoch:10 step:  111/151, lr:0.000069, giou_loss:   7.25, conf_loss:   8.86, prob_loss:   0.11, total_loss:  16.23
epoch:10 step:  112/151, lr:0.000069, giou_loss:   5.17, conf_loss:   7.99, prob_loss:   0.08, total_loss:  13.24
epoch:10 step:  113/151, lr:0.000069, giou_loss:   4.88, conf_loss:   8.08, prob_loss:   0.03, total_loss:  12.99
epoch:10 step:  114/151, lr:0.000069, giou_loss:   5.26, conf_loss:   5.60, prob_loss:   0.04, total_loss:  10.90
epoch:10 step:  115/151, lr:0.000069, giou_loss:   1.38, conf_loss:   4.90, prob_loss:   0.01, total_loss:   6.29
epoch:10 step:  116/151, lr:0.000069, giou_loss:   4.91, conf_loss:   5.71, prob_loss:   0.05, total_loss:  10.67
epoch:10 step:  117/151, lr:0.000069, giou_loss:   4.21, conf_loss:   5.24, prob_loss:   0.02, total_loss:   9.46
epoch:10 step:  118/151, lr:0.000068, giou_loss:   6.66, conf_loss:   7.00, prob_loss:   0.05, total_loss:  13.71
epoch:10 step:  119/151, lr:0.000068, giou_loss:  12.87, conf_loss:  11.98, prob_loss:   0.22, total_loss:  25.07
epoch:10 step:  120/151, lr:0.000068, giou_loss:   8.56, conf_loss:  10.63, prob_loss:   0.07, total_loss:  19.26
epoch:10 step:  121/151, lr:0.000068, giou_loss:   3.55, conf_loss:   5.74, prob_loss:   0.02, total_loss:   9.31
epoch:10 step:  122/151, lr:0.000068, giou_loss:   4.03, conf_loss:   4.14, prob_loss:   0.01, total_loss:   8.18
epoch:10 step:  123/151, lr:0.000068, giou_loss:   3.30, conf_loss:   5.00, prob_loss:   0.03, total_loss:   8.33
epoch:10 step:  124/151, lr:0.000068, giou_loss:  13.37, conf_loss:   9.40, prob_loss:   0.16, total_loss:  22.94
```

```
0.183% = tomato AP
mAP = 0.183%, 9.01 FPS
```

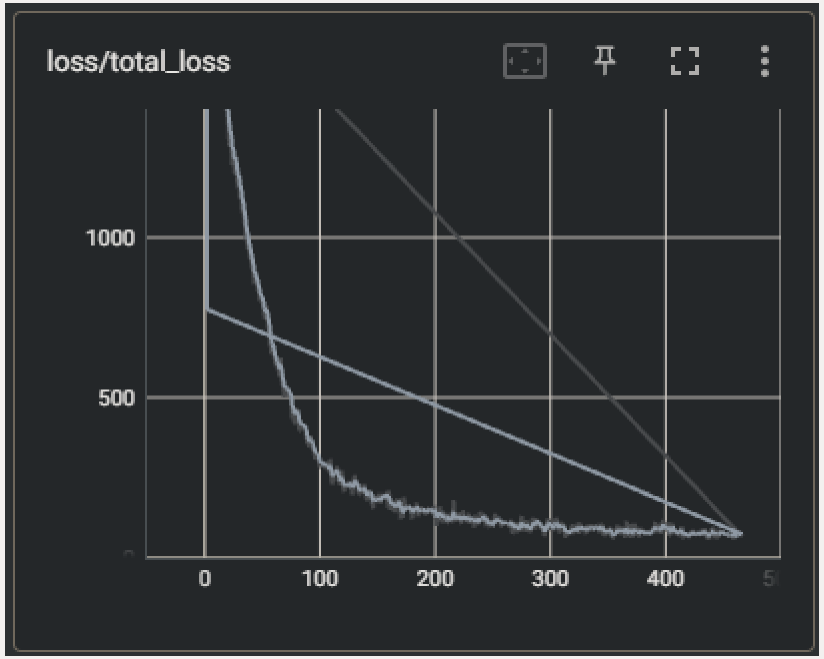
more Accurate Testing Results

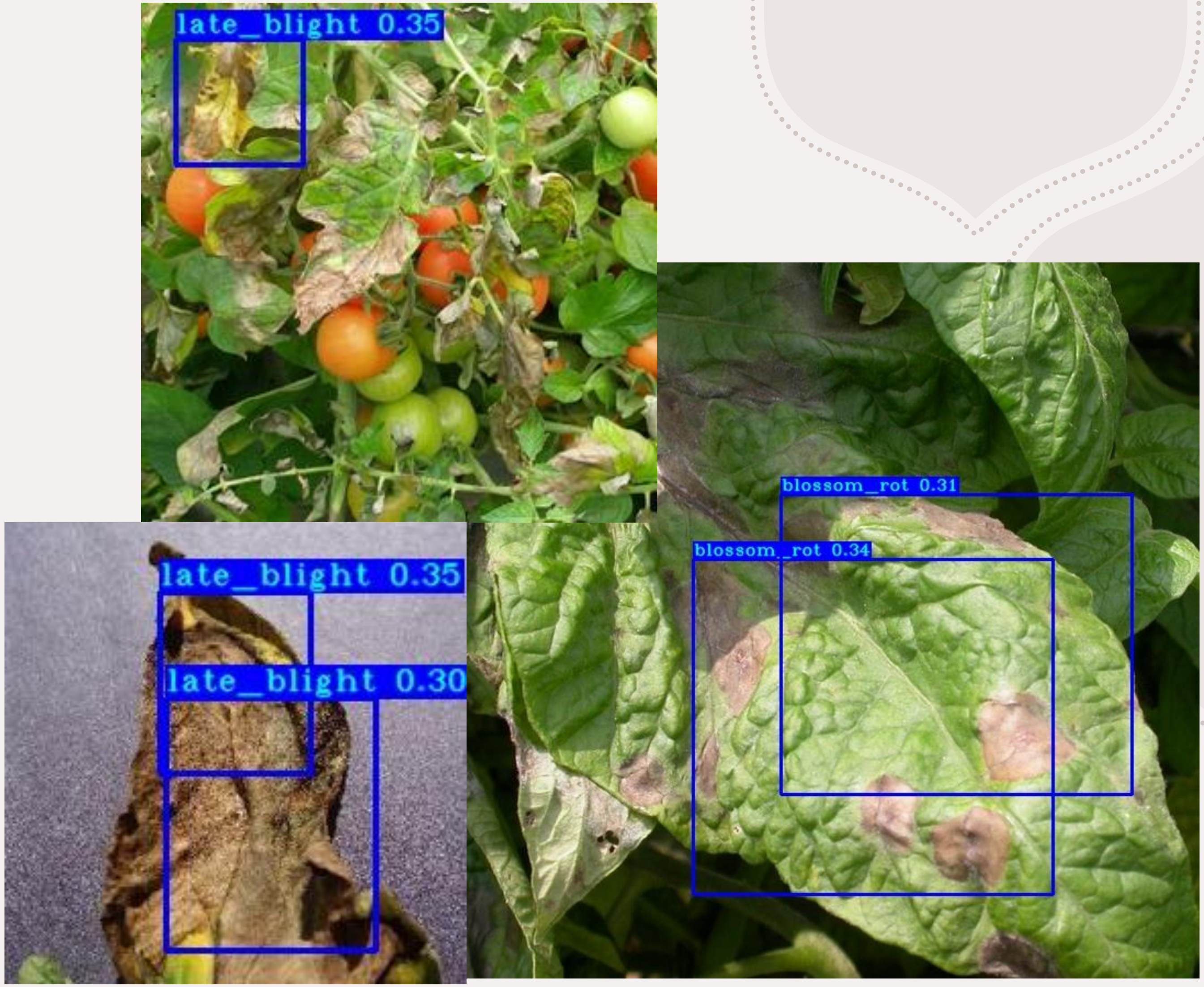
Leads to Accurate Image/object Detection



# Results

- Using YOLO v4 and TensorFlow we were able to train and successfully detect early & late blight diseases as well as tomatoes themselves.
- Many issues we ran into, mostly where with documentation, and compatibility between dependencies and chunks of code.




# Improvements

Our detections didn't provide the highest level of accuracies, this is to be expected since some of the labeling of images contain shadows, random artifacts and in general YOLOV4, and other lightweight models are not accurate due to their speed and purpose of implementation. However, we can repurpose our code and convert to TFRecords, TFJS and whatever necessary to be able to run on on hardware like Jetson Nano’s, Raspberry Pi’s, Android/IOS devices and even Drones.

# Sources


- Huang, Mei-Ling; Chang, Ya-Han (2020), “Dataset of Tomato Leaves”, Mendeley Data, V1, doi: 10.17632/ngdgg79rzb.1
- https://github.com/pythonlessons/TensorFlow-2.x-YOLOv3.git
- GitHub - nicknochnack/TFODCourse
- models/tf2_detection_zoo.md at master · tensorflow/models · GitHub
- API Documentation | TensorFlow v2.11.0
- https://www.kaggle.com/datasets/andrewmvd/tomato-detection?resource=download
- When to Plant Tomatoes in California? (Region By Region) - GFL Outdoors