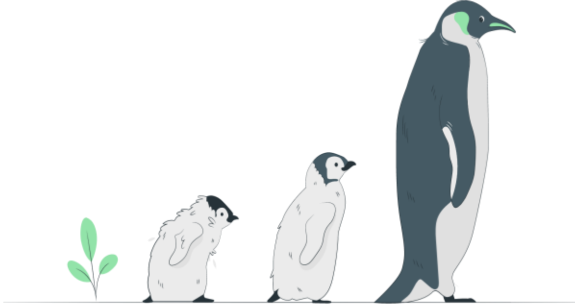
MULTI-CLASS WILDLIFE SPECIES CLASSIFICATION

Team Members: Rajshree Mishra, Samarth Mishra, Amritangshu Mukherjee, Aishwarya Sarkar, Vivian Wang

01	OBJECTIVE
02	DATASET
03	APPROACH
04	RESULTS
05	NEXT STEPS

Objective



Detect and classify over 80 species of wildlife captured by wildlife cameras

1.

Recognize predatory animals which may pose threat to other species or humans

2.

Monitor endangered species and determine long term viability

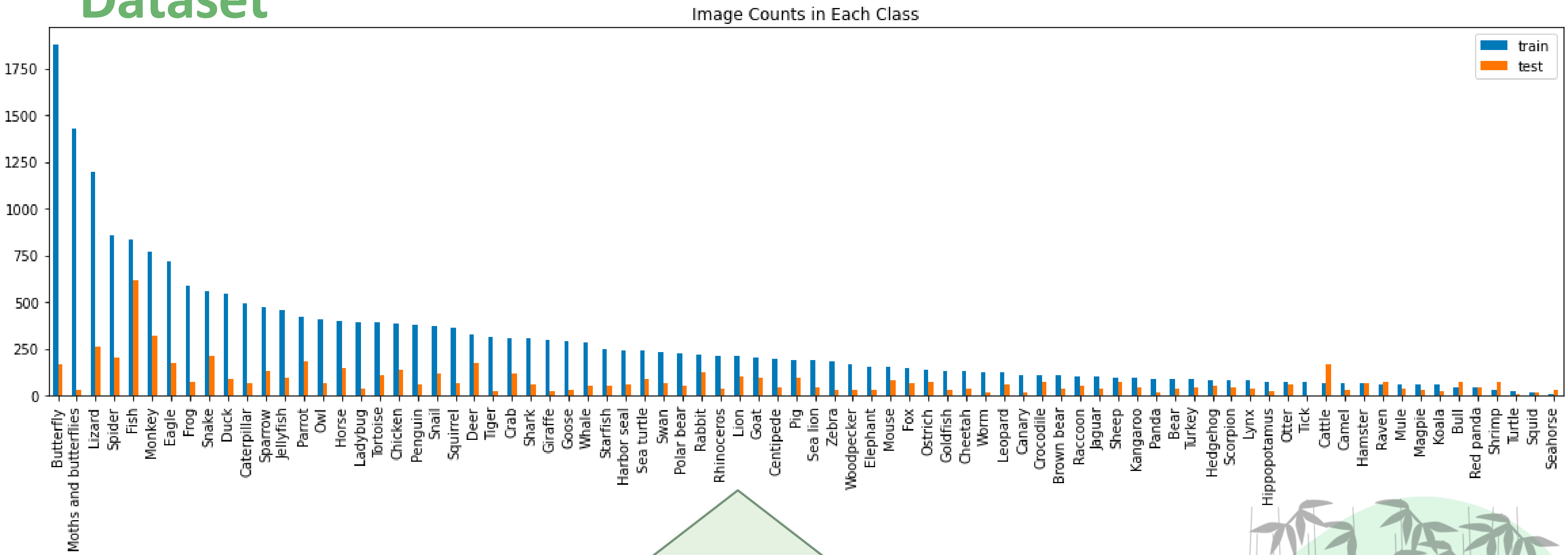
3.

Reduce the number of traffic accidents in various regions

4.

Regulate poaching in restricted areas by keeping a count of the number of animas

Dataset



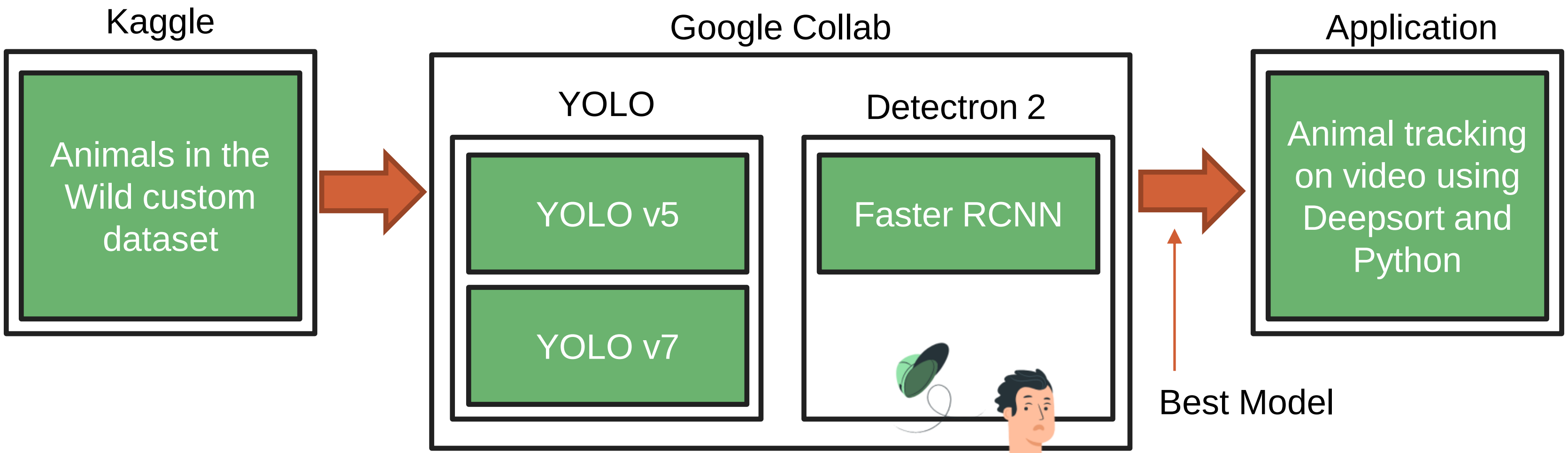
Public dataset on Kaggle

29,000 wildlife images

80 wildlife classes

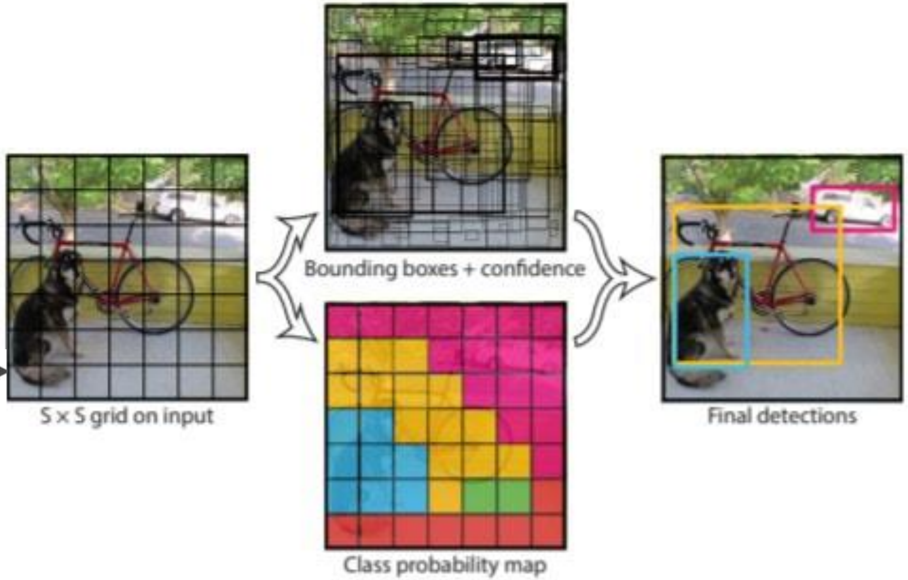
Approach

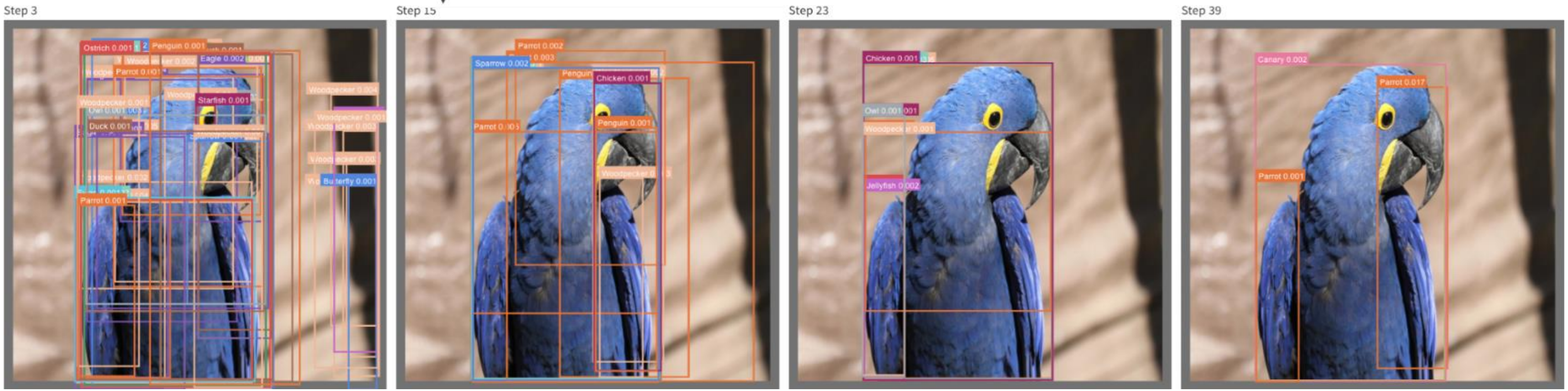
- We trained 3 models using transfer learning on our dataset for object detection



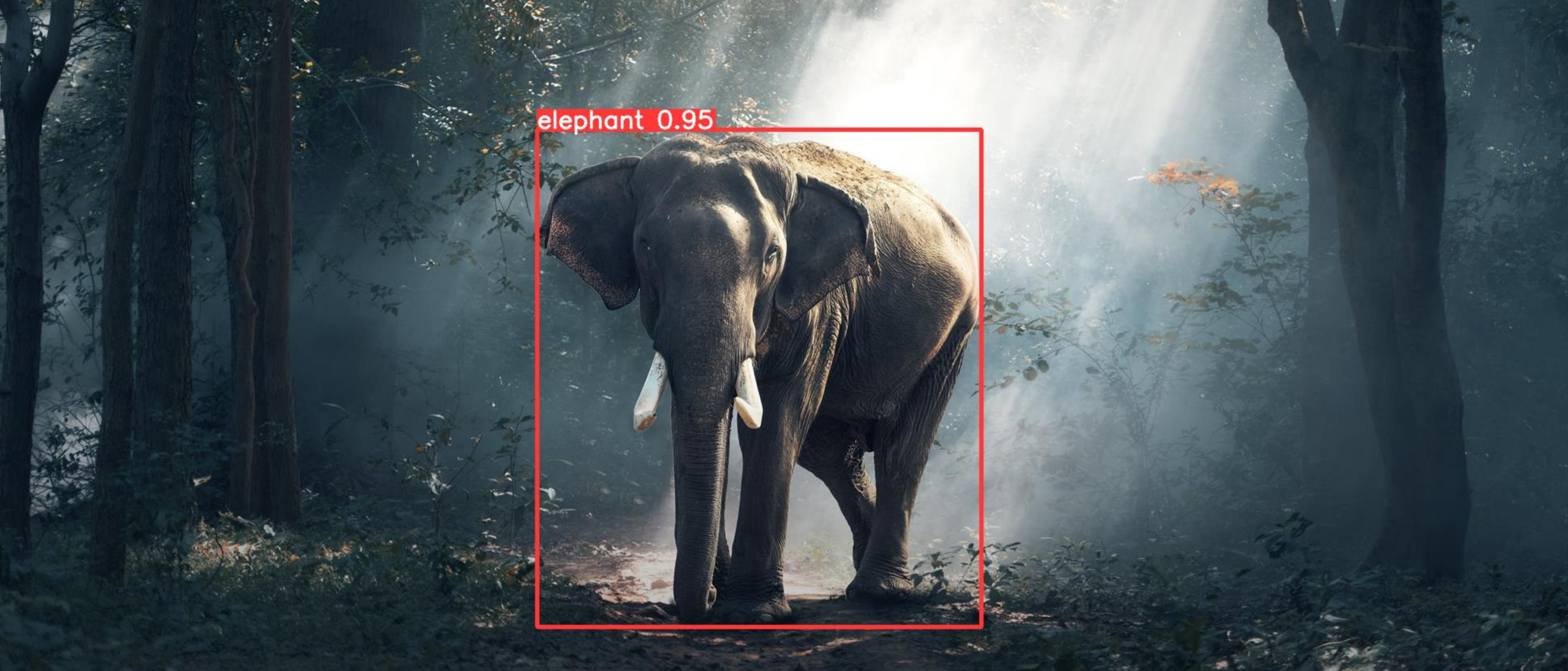
How does YOLO work?

- YOLO system divides the input image into an S × S grid
- Each grid cell predicts B bounding boxes. Each bounding box consists of 5 predictions- x, y, w, h, and confidence
- Each grid cell also predicts C conditional class probabilities, Pr(Class|Object)

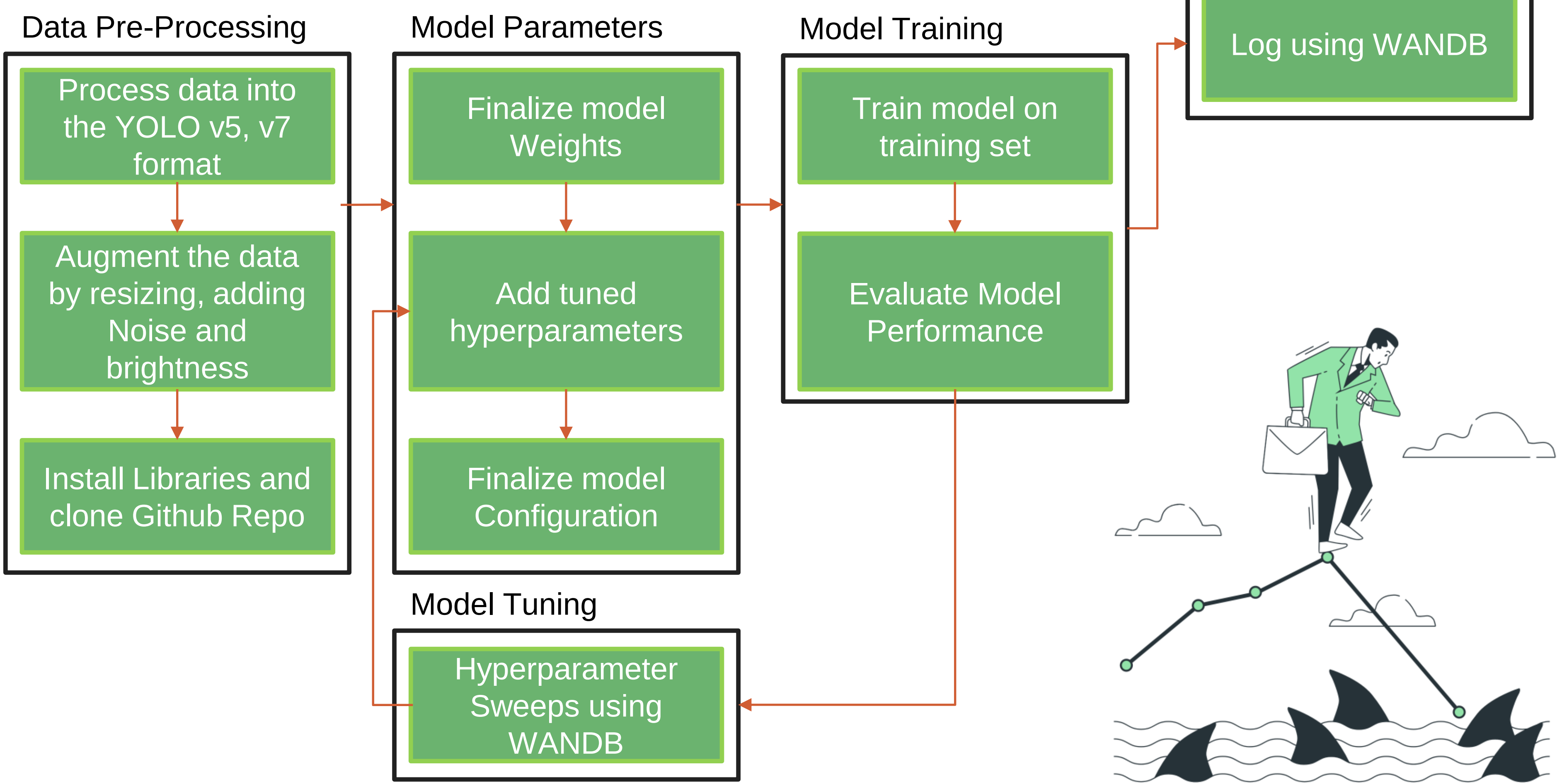




YOLO v5,v7 IMPLEMENTATION AND RESULTS



Model Training Methodology

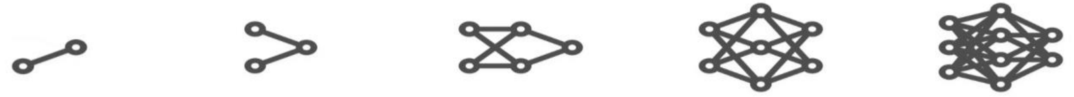


YOLO v5 Setup for training on custom dataset using transfer learning

- !python train.py
- --data animal_detect1.yaml
- --weights yolov5s.pt
- --batch-size 16
- --epochs 50
- --cfg yolov5s.yaml
- --hyp hyp.scratch-low.yaml

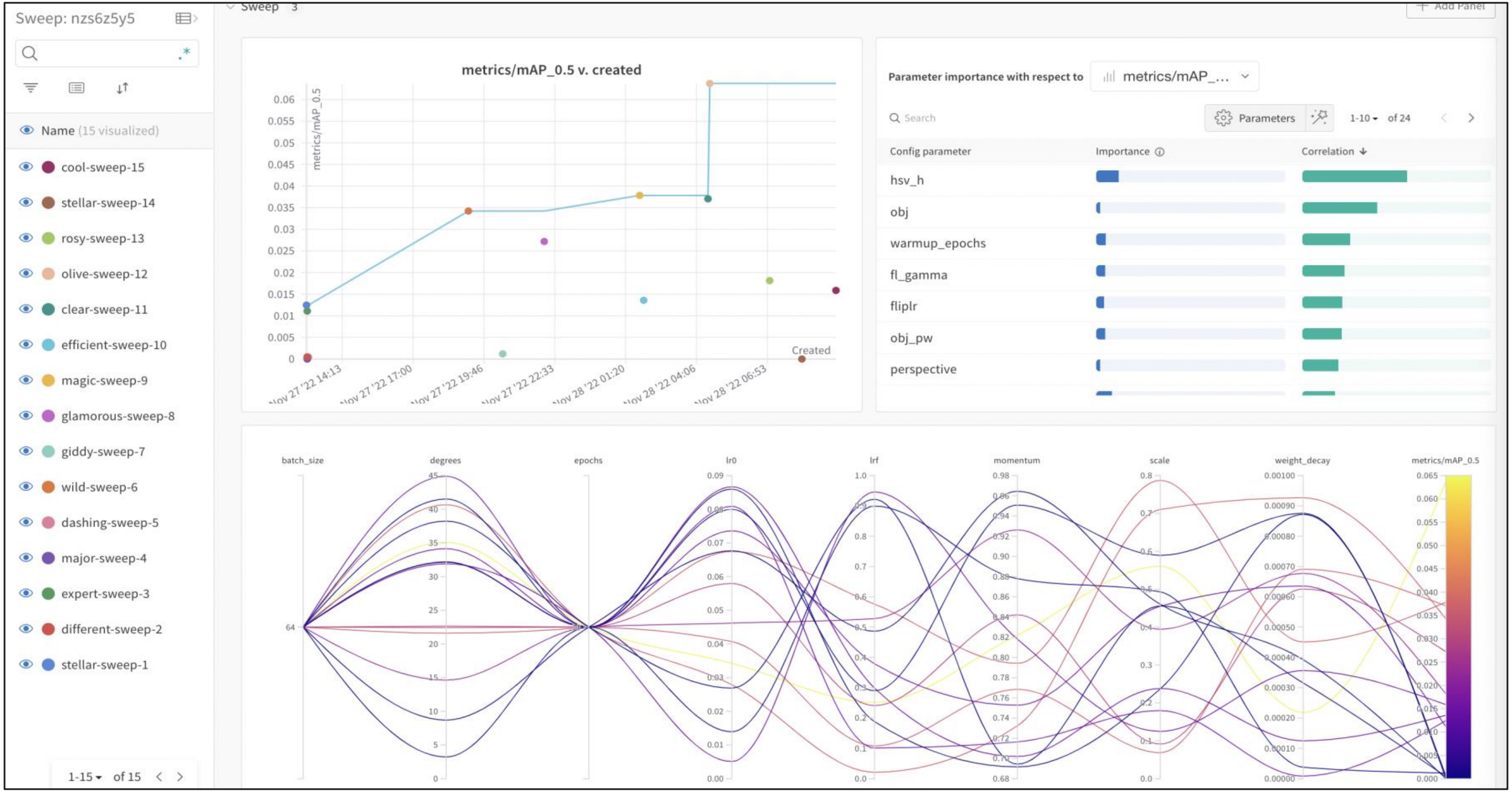
```yaml
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

```yaml
# Parameters
nc: 80  # number of classes
depth_multiple: 0.67  # model depth multiple
width_multiple: 0.75  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]],  # 9
  ]

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)
```



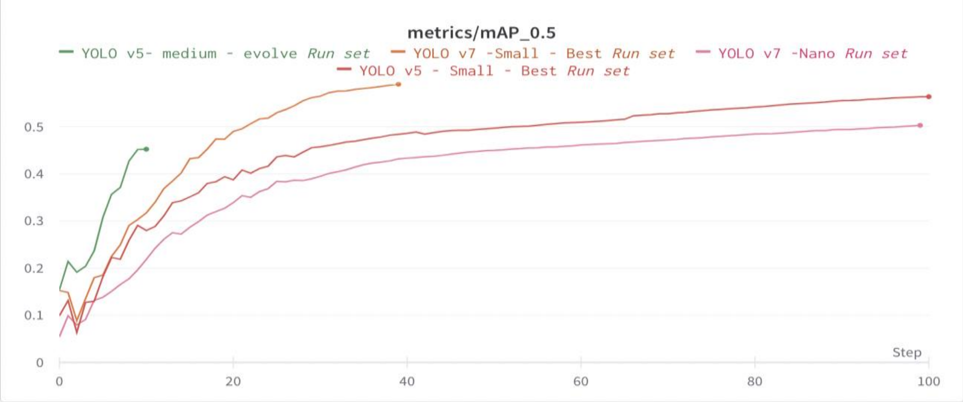
Nano	Small	Medium	Large	XLarge
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
4 MB$_{FP16}$	14 MB$_{FP16}$	41 MB$_{FP16}$	89 MB$_{FP16}$	166 MB$_{FP16}$
6.3 ms$_{V100}$	6.4 ms$_{V100}$	8.2 ms$_{V100}$	10.1 ms$_{V100}$	12.1 ms$_{V100}$
28.4 mAP$_{COCO}$	37.2 mAP$_{COCO}$	45.2 mAP$_{COCO}$	48.8 mAP$_{COCO}$	50.7 mAP$_{COCO}$

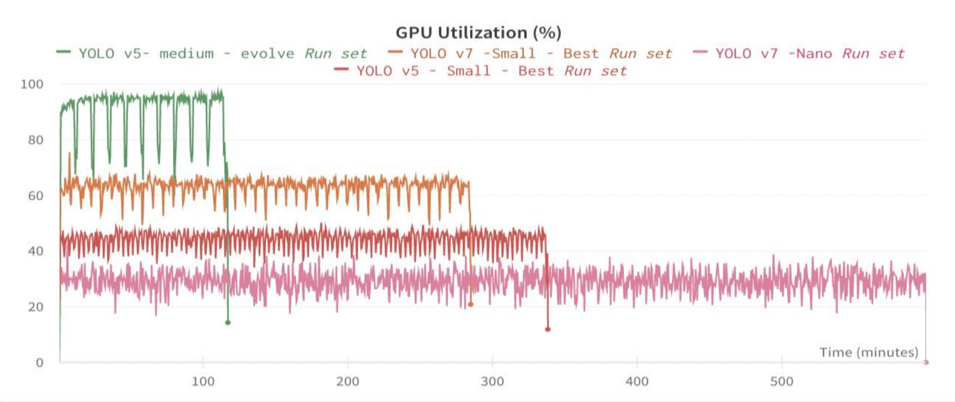
Yolo Hyperparameter Tuning using Sweeps

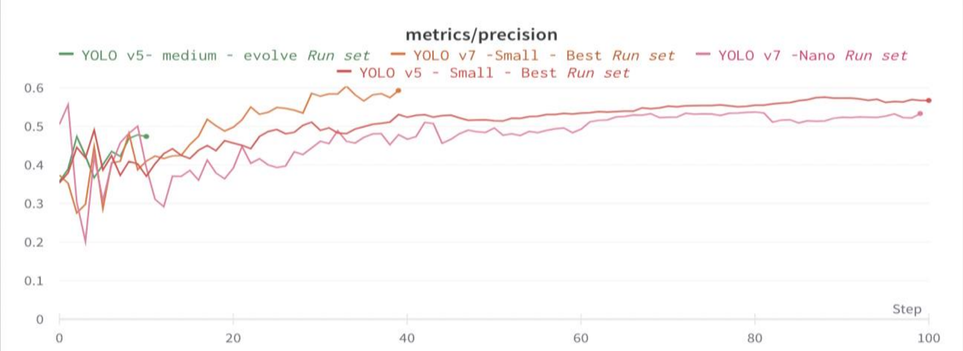


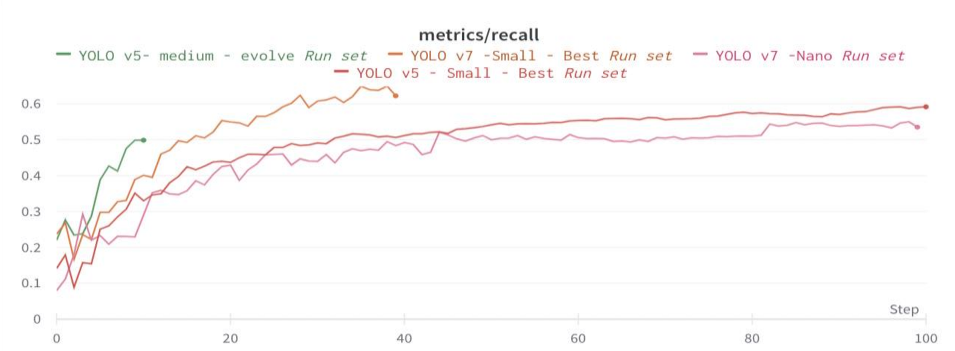
YOLO Run Metrics

https://wandb.ai/amritangshu/YOLOR/reports/Multi-Class-wildlife-detection-using-YOLOv5-and-YOLOv7--VmlldzozMDE1NDcy







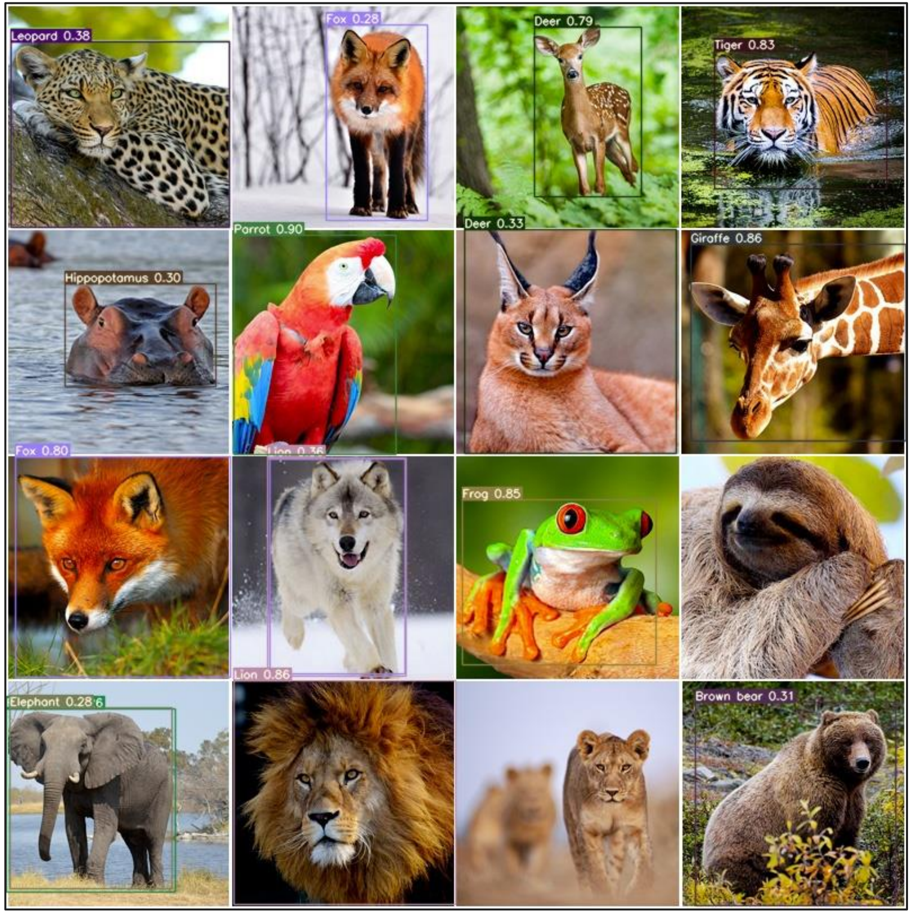


Name	Runtime	GPU Type	epochs
YOLO v5- medium-evolve	1.95 hr	Tesla T4	10
YOLO v7 - Final - Small	4.75 hr	A100-SXM4-40GB	40
YOLO v7 - Run 2 - Nano	9.99 hr	A100-SXM4-40GB	100
YOLO v5 - Final - Small	5.64 hr	A100-SXM4-40GB	100

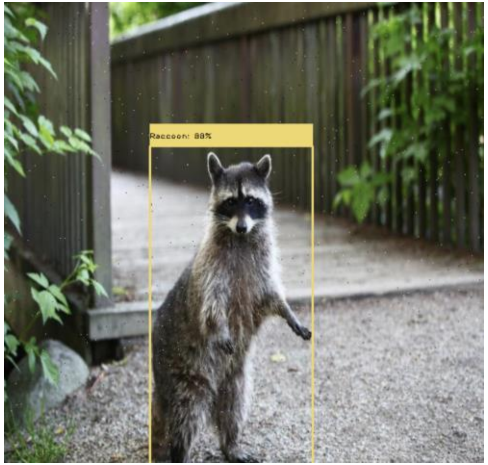
YOLO V5 vs V7 Prediction

YOLO v5

YOLO v7



Testing YOLO models with Changes in Noise and Colors



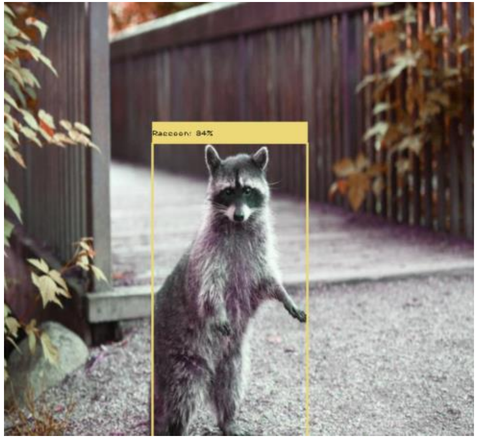
Black and white grainy (animal not detected)

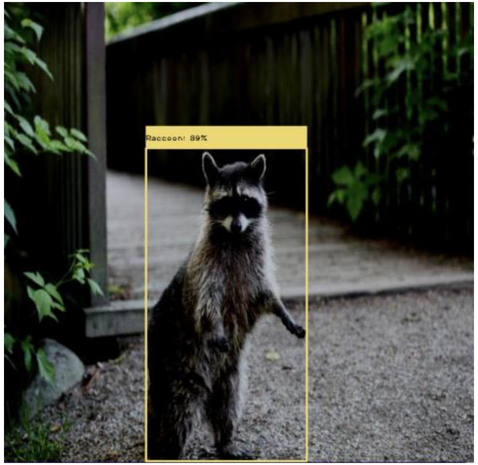
Vintage color palette (84%)

Original Image
Confidence: 88%

Decreased brightness and grainy (88%)

Increased contrast (89%)





Alternative Approach - Detectron2's Faster RCNN

- Developed by Facebook, collection of models
- Uses 9 anchor boxes of different scales and aspect ratios to obtain ROIs (region of interest)
- Reduces number of ROIs with binary classifier and Softmax layer
- 25000 Images labeled with bounding boxes
- Uploaded images to Roboflow, which is linked to in our Colab notebook
- Used pre-trained weights on COCO dataset to initialize model
- 1000 iterations

COCO pretrained model predictions

Instance segmentation model

panoptic segmentation model

Detectron2 Predictions

Name	Runtime	GPU Type	epochs
YOLO v5-medium-evolve	1.95 hr	Tesla T4	10
YOLO v7 - Final - Small	4.75 hr	A100-SXM4-40GB	40
YOLO v7 - Run 2 - Nano	9.99 hr	A100-SXM4-40GB	100
YOLO v5 - Final - Small	5.64 hr	A100-SXM4-40GB	100
Detectron2 Faster RCNN	20 min	Tesla T4	1000 Iterations

The Predictions here are not accurate for our custom dataset due to some data formatting issues in COCO Json format and GPU limitations

Application/ Next Steps

Inference on Video using our best model from Yolo V7 and using the python library for Deepsort
https://www.youtube.com/watch?v=Wr_XIrYdzy4

Semantic Segmentation with YOLO V5 default model

Thank You

Questions?

Image generated using Dalle-2, Yolo v5 inference