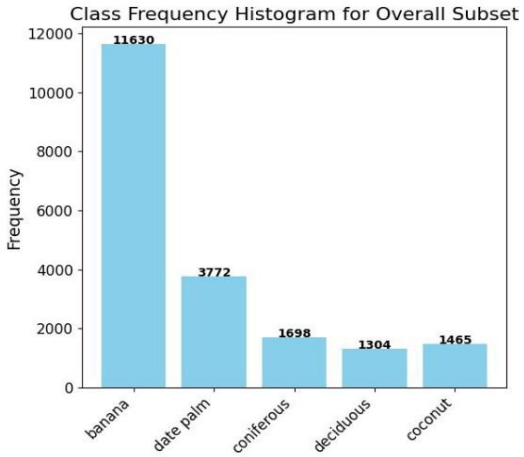
Monthly Report II

Onboard Classification of Tree Species using RGB UAV Imagery

Mahendra Priolkar, Shantanu Prabhudessai, Mansel Martins

Previous Dataset Problems

The original Augmented Multi-Tree Species Aerial Detection (MTAD) Dataset created by Anish Natekar et al., consisted of five classes, namely; Banana, Coconut, Date Palm, Deciduous and Coniferous. This was very imbalanced, and except banana, all classes had around 1.5k instances, while the recommended minimum amount is 10k instances per class.



Dataset

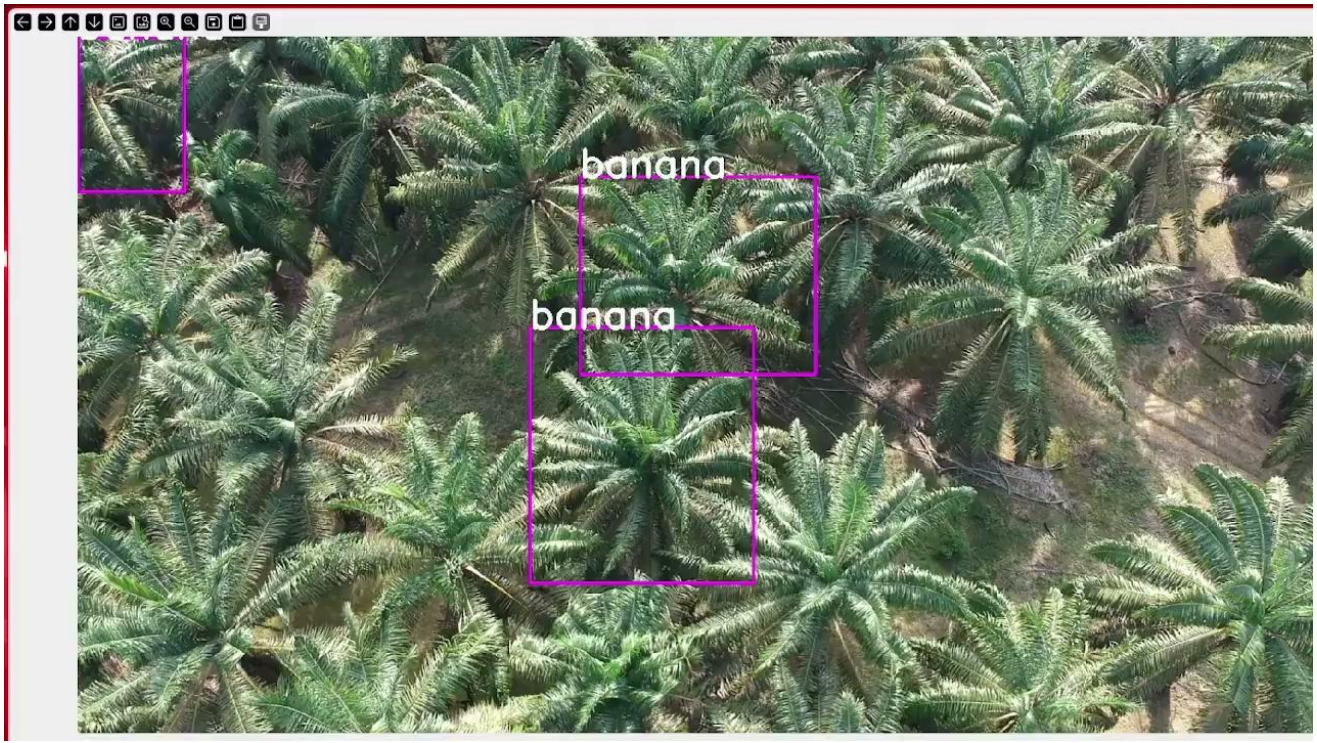
- **Images per class.** ≥ 1500 images per class recommended
- **Instances per class.** ≥ 10000 instances (labeled objects) per class recommended
- **Image variety.** Must be representative of deployed environment. For real-world use cases we recommend images from different times of day, different seasons, different weather, different lighting, different angles, different sources (scraped online, collected locally, different cameras) etc.
- **Label consistency.** All instances of all classes in all images must be labelled. Partial labelling will not work.
- **Label accuracy.** Labels must closely enclose each object. No space should exist between an object and it's bounding box. No objects should be missing a label.
- **Label verification.** View `train_batch*.jpg` on train start to verify your labels appear correct, i.e. see example mosaic.
- **Background images.** Background images are images with no objects that are added to a dataset to reduce False Positives (FP). We recommend about 0-10% background images to help reduce FPs (COCO has 1000 background images for reference, 1% of the total). No labels are required for background images.

Source: https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/

v8s trained on MTAD Dataset

Accuracy that'll make you go:

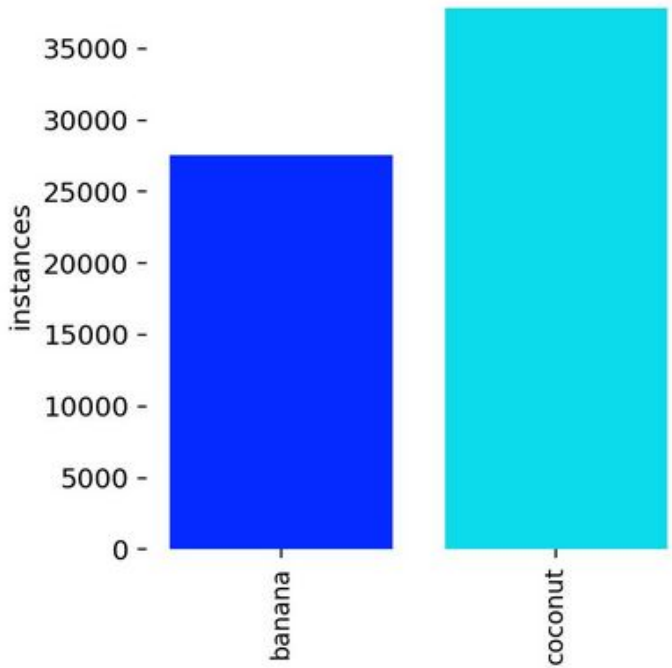
"That's Bananas!"



https://drive.google.com/file/d/11JjfwB1-fRfNIYadkvp1geigeTuYvozI/view?usp=drive_link

The new Bi-Class Dataset

- Features two classes, namely; Banana and Coconut
- 11.6k images of Banana trees, and 14.4k images of Coconut trees; both above the recommended minimum.
- It was created by combining smaller datasets, and augmenting the train data to a large degree.



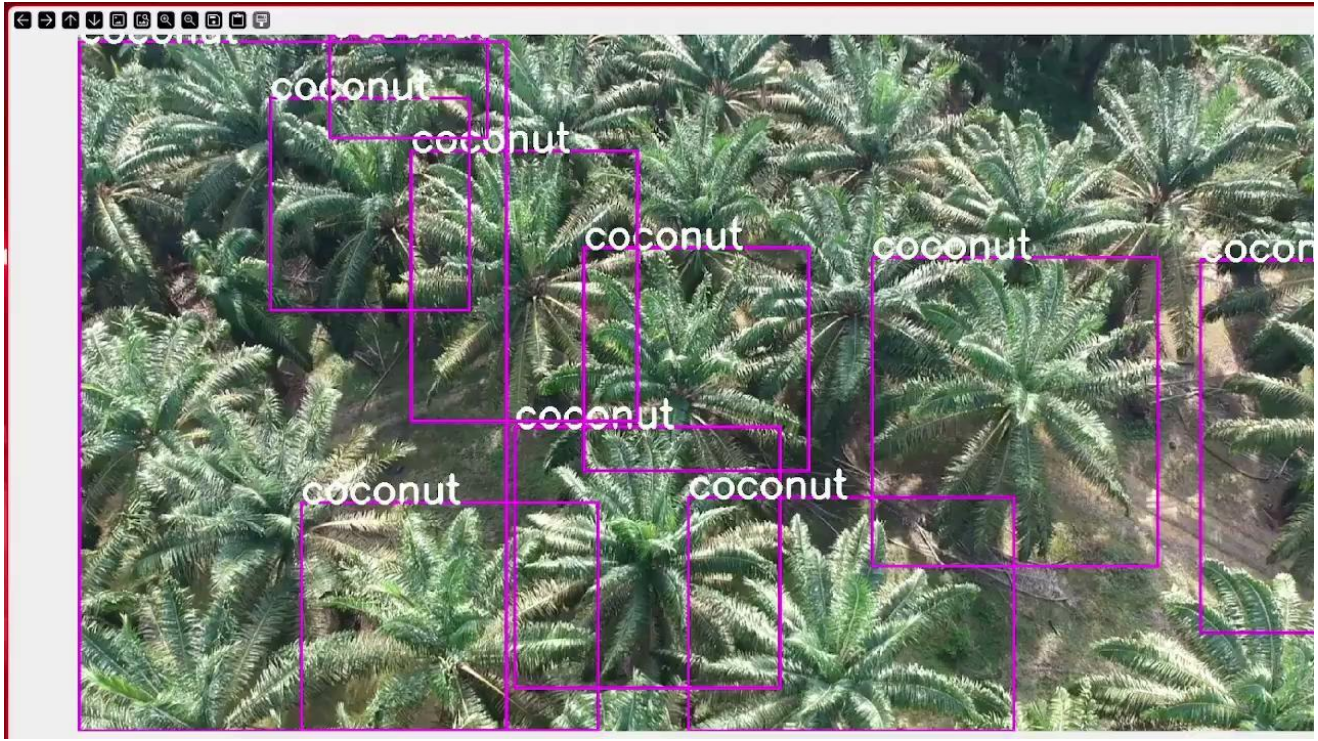
Dataset Images Example

Class Labels

Banana: 0

Coconut: 1

v8s trained on Bi-Class Dataset



https://drive.google.com/file/d/1jjPrQPQ7Lsqbc4alYgl9OLgdW6TpjU0J/view?usp=drive_link

YOLO v10 Improved ECA

Similar to modifying YOLOv8, the Improved ECA module can be added to YOLOv10 following the C2F block and the new C2f-CIB block.

A snippet from the YOLO v10 Improved ECA YAML file ->

```
# YOLOv10.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 13

  #ECA Block
  - [-1, 1, Improved_ECA, [512,]]

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f, [256]] # 16 (P3/8-small)

  #ECA Block
  - [-1, 1, Improved_ECA, [256,]]

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 13], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f, [512]] # 19 (P4/16-medium)

  #ECA Block
  - [-1, 1, Improved_ECA, [512,]]

  - [-1, 1, SCDown, [512, 3, 2]]
  - [[-1, 10], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2fCIB, [1024, True, True]] # 22 (P5/32-large)

  #ECA Block
  - [-1, 1, Improved_ECA, [1024,]]

  - [[16, 19, 22], 1, v10Detect, [nc]] # Detect(P3, P4, P5)
```

CBAM: Convolutional Block Attention Module

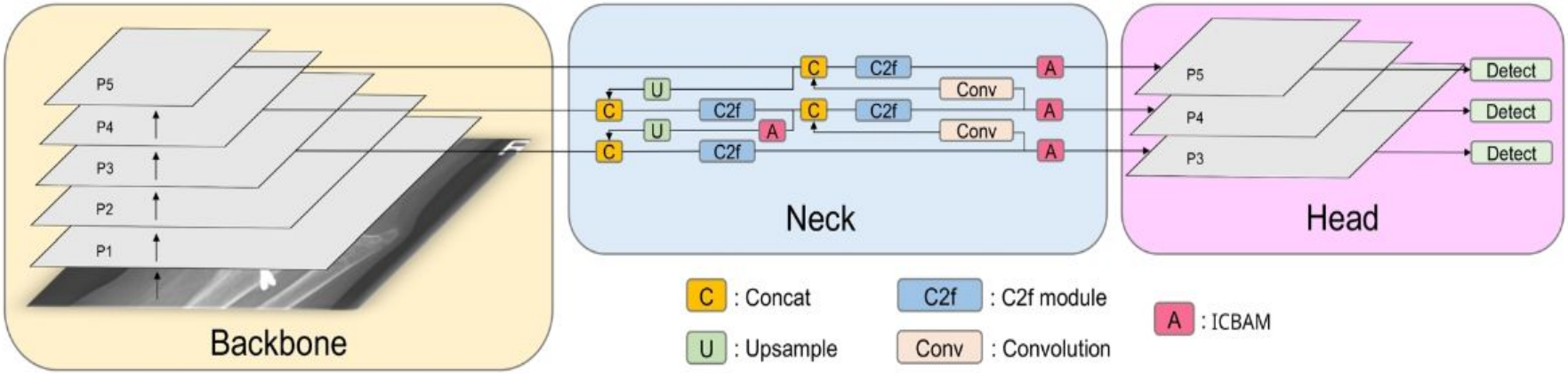
The YOLO CBAM attention process works as follows: First, the input feature map is enhanced by the channel attention block, the output of which is enhanced by the spatial attention block.

- Channel Attention block: squeezes the spatial dimension using average pooling.The Channel Refined Feature F_{CR} can be defined as: $F_{CR} = M_c(F_{input}) \odot F_{input}$,where $F_{input} \epsilon \mathbb{R}^{C \times H \times W}$ is an intermediate feature map, and $\odot$ is the Hadamard (element-wise) product.
- Spatial Attention block: runs average and max pooling across the channel dimension and concatenates them to create a spatial feature descriptor.The Final Refined Feature F_{FR} can be defined as: $F_{FR} = M_s(F_{CR}) \odot F_{CR}$

  For CBAM, output F_{output} is simply F_{FR}.

ICBAM

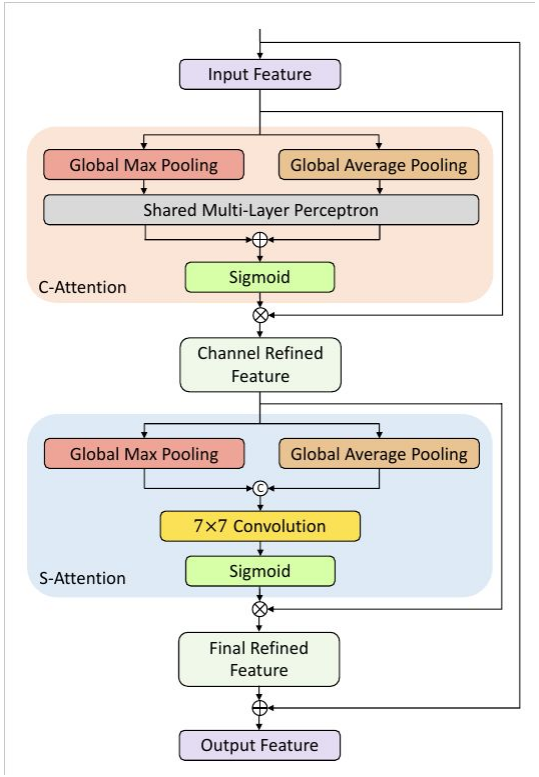
We propose a new attention block, the Improved Convolutional Block Attention Module (ICBAM). The channel attention block from CBAM is replaced by the HyP-ECA block.



Original source of above figure: https://arxiv.org/abs/2402.09329

Res-CBAM

ResBlock+CBAM (Res-CBAM for short) is a module that has a shortcut that adds the original input feature map back to enhanced output of CBAM followed by an activation function.



https://arxiv.org/abs/2402.09329 [Accessed: 21-09-2024]

Results

Comparison of various YOLO models trained on Bi-Class dataset.

Model	mAP50	mAP50-95	Inference Time(s) On Raspberry Pi	F1 Score
YOLOv8n	0.904	0.549	1.204	0.856
YOLOv8n HyP-ECA	0.914	0.538	**1.191**	0.859
YOLOv8n Improved CBAM	0.893	0.532	1.223	0.844
YOLOv8n Res-CBAM	0.898	0.535	1.444	0.857
YOLOv8s	0.908	0.556	3.302	0.855
YOLOv8s HyP-ECA	0.910	0.542	3.307	0.857
YOLOv8s Improved CBAM	0.913	0.561	3.345	0.859
YOLOv10n	0.906	0.551	1.352	0.831
YOLOv10n HyP-ECA	0.906	0.543	1.872	0.851
YOLOv10n Improved CBAM	0.900	0.527	1.904	0.852
YOLOv10n Res-CBAM	0.903	0.543	2.108	0.846
YOLOv10s	**0.917**	**0.577**	3.170	0.860
YOLOv10s HyP-ECA	0.911	0.559	5.308	**0.861**
YOLOv10s Improved CBAM	0.912	0.564	5.335	0.856

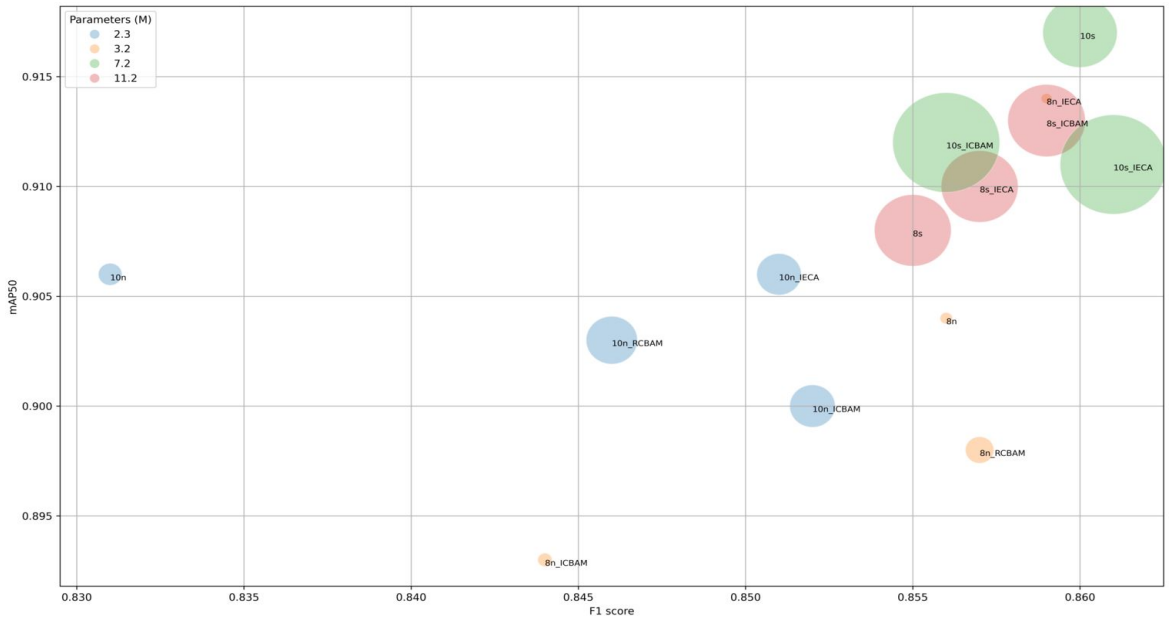
Results

Comparison of various quantized YOLO models trained on Bi-Class dataset.

Model	mAP50	mAP50-95	Inference Time(s) On Raspberry Pi	F1 Score
YOLOv8n NCNN	0.879	0.528	**0.509**	0.833
YOLOv8n HyP-ECA NCNN	**0.883**	0.508	0.535	0.828
YOLOv8n Improved CBAM OpenVINO	0.875	0.515	0.613	0.829
YOLOv8n Res-CBAM OpenVINO	0.876	0.515	0.664	**0.842**
YOLOv8s NCNN	0.879	**0.531**	1.147	0.840
YOLOv8s HyP-ECA NCNN	0.881	0.521	1.138	0.835
YOLOv8s Improved CBAM OpenVINO	0.882	**0.531**	1.226	0.835
YOLOv10n OpenVINO	0.868	0.522	0.588	0.818
YOLOv10n HyP-ECA OpenVINO	0.872	0.516	0.671	0.824
YOLOv10n Improved CBAM OpenVINO	0.872	0.504	0.670	0.821
YOLOv10n Res-CBAM OpenVINO	0.864	0.512	0.717	0.821
YOLOv10s OpenVINO	0.864	0.525	1.178	0.817
YOLOv10s HyP-ECA OpenVINO	0.877	**0.531**	1.472	0.827
YOLOv10s Improved CBAM OpenVINO	0.866	0.526	1.445	0.824

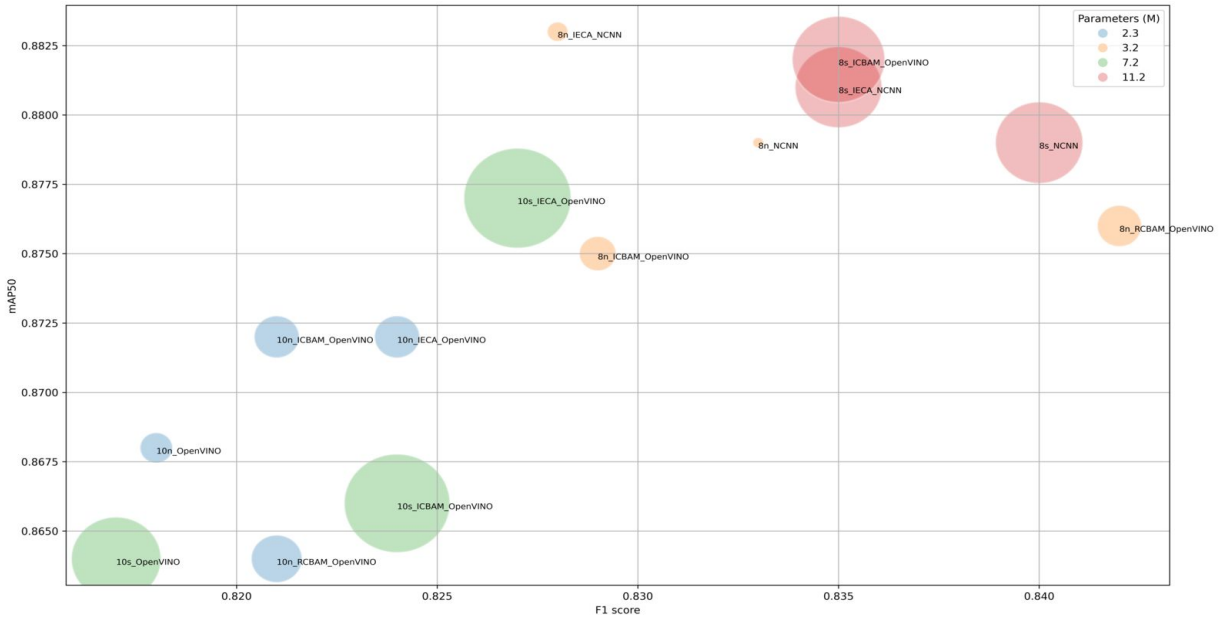
Performance Metric Plots (Standard Models)

F1 Score vs mAP50 vs Inference time(sec, bubble size) vs Parameter count



Performance Metric Plots (Quantized Models)

F1 Score vs mAP50 vs Inference time(sec, bubble size) vs Parameter count



Result Analysis

Standard Models

8n Improved ECA strikes the right balance between mAP50 and F1 score, along with decent inference time(1.2s). Nevertheless, other models such as **8s Improved CBAM**, **10s**, and **8s Improved ECA** show promising results, despite having higher inference time(3.2s).

Quantized Models

Either **8s Improved CBAM OpenVINO** or **8s NCNN** can be considered, with both trading off one of the performance metrics (mAP, F1) in exchange for the other. However, if speed needs to be prioritised, **8n NCNN** is the superior choice, with the best inference time(0.5s), but with very low accuracy.

ICBAM vs Improved ECA

Although the Improved CBAM architecture performed worse than HyP-ECA for the n sized models, it performed better for the two s sized models (8s,10s) that were trained.

Model	mAP50	mAP50-95	Inference Time(s) On Raspberry Pi	F1 Score
YOLOv8s Improved ECA	0.910	0.542	3.307	0.857
YOLOv8s Improved CBAM	0.913	0.561	3.345	0.859
YOLOv8s Improved ECA NCNN	0.881	0.521	1.138	0.835
YOLOv8s Improved CBAM OpenVINO	0.882	**0.531**	1.226	0.835
YOLOv10s Improved ECA	0.911	0.559	5.308	0.861
YOLOv10s Improved CBAM	0.912	0.564	5.335	0.856

ResCBAM Analysis

While ResCBAM performs almost equivalently to Improved ECA and ICBAM, due to time constraints, we could only train n sized models for ResCBAM. Perhaps, like ICBAM, ResCBAM could show measurable performance increases in the s sized model category.

Hardware

Running inference on the UAV itself using just a Raspberry Pi 4 imposes a large limit to the size of the model that can be used, leading to a lower accuracy.

One option would be to use better edge devices, such as the Raspberry Pi 5, or a NVIDIA Jetson Nano.

Future Work

ResCBAM

In the future, YOLOv10s and YOLOv8s ResCBAM could be trained, with the hope that it, similar to ICBAM, might perform better than Hyp-ECA in the s sized model category. This could indicate that models of the CBAM family perform better as the model size increases, but this is all speculation at this point, and would need detail studies conducted into the matter.

Dataset

Training the models on higher resolution images would likely improve accuracy to a large extent. One approach that could be taken would be to upscale the existing images using upscalers such as ERSGAN. Another downside with both datasets is that the images were captured when the sun was low in the sky, casting long shadows, which are highly undesirable. A possible solution would be to conduct surveys for capturing images between 10am and 3pm, when the sun is nearly vertical, casting the smallest shadows, preferably on an overcast day.

Future Work

Hyperparameter Tuning

Various kernel sizes for the attention block in ICBAM could be tried, to determine the best.

THANK YOU

Acknowledgement

We would like to thank Dr. Clint Pazhayidam George and Dr. Shitala Prasad for their guidance throughout this internship project duration,
And our GEC Faculty coordinators, Dr. Rachel Dhanaraj and Prof. Shruti Pednekar.