COMP9517: Computer Vision 2023 Term 2 Group Project Report

Eric Jiang
z5314105

Raman Seysan
z5238839

YuHao(Gary) Guo
z5318712

ZiYi(Zeal) Liang
z5325156

***Abstract*—This paper describes a team project in the field of computer vision aimed at exploring different deep learning approaches to solve the tasks of classification and target detection for penguins and turtles. We compare classical convolutional neural networks (VGG16, ResNet50 and EfficientNetV2) and implement YOLOv8 for target detection. By analyzing the experimental results of the different methods, we discuss the performance and limitations of each method.**

***Index Terms*—Computer Vision, Deep Learning, VGG16, ResNet50, EfficientNetV2, YOLOv8, Feature Extraction, SVM Classification, Data Enhancement, Target Detection, Classification, COCO dataset**

I. Introduction

This report examines advancements in computer vision, specifically in object detection and classification. It traces the evolution of the YOLO model from its inception to the 2023 version, YOLOv8, discussing its improved accuracy and increased computational complexity. The report also reviews models like VGG16, ResNet50, and EfficientNetV2. A project involving image classification of penguins and turtles is highlighted, addressing issues of overfitting due to limited datasets by using data augmentation and transfer learning. Performance comparisons are drawn between the DETR, HOG, VGG16, ResNet50, and EfficientNetV2 models. Furthermore, the report explores the application of YOLOv8 in classification and detection. Detailed experimental results provide insights into these methodologies and their effectiveness in addressing common issues in computer vision tasks.

II. Literature Review

A. VGG 16

Karen Simonyan & Andrew Zisserman [4] developed a convolutional networks model with the investigation of the effect of depth on its accuracy. The depth of their model is 16-19 and win first and second in price ImageNet Challenge 2014.

B. ResNet50

This paper [5] proposes a residual learning framework to simplify the training of deep neural networks. The authors reformulate layers as learning residual functions based on layer inputs, which makes it easier to optimize these residual networks and gain accuracy by increasing depth. They evaluated residual networks with depths up to 152 layers on the ImageNet dataset and achieved an error rate of 3.57%. The authors also analyzed CIFAR-10 with 100 and 1000 layers. Their deep residual network won first place in several tasks in the ILSVRC & COCO 2015 competition.

C. EfficientNetV2

The paper [6], ”EfficientNetV2: Smaller Models and Faster Training,” describes EfficientNetV2, a new kind of convolutional network with faster training speeds and better parameter efficiency than previous models. The authors use a combination of training-aware neural architecture search and scaling to optimize training speed and parameter efficiency. They also propose an improved asymptotic learning method that adaptively adjusts regularization as the image size changes.The EfficientNetV2 model significantly outperforms previous models on the ImageNet and CIFAR/Cars/Flowers datasets. They also achieved 87.3% top-1 accuracy on ImageNet ILSVRC2012 with 5x-11x faster training.

D. YOLOv8

YOLO is an efficient feature detection method first published by R. Joseph et al. in 2015 [7]. Better generalization than competitors at the time - Showed better performance in another domain test (images; trained on ImageNet). Reduced false positives in the background part of the image.

By 2023 YOLO has evolved to v8. As per the official description, YOLOv8 is a SOTA model that builds on the success of previous YOLO versions and introduces new features and improvements to further enhance performance and flexibility. Specific innovations include a new backbone network, a new Ancher-Free detection header and a new loss function.

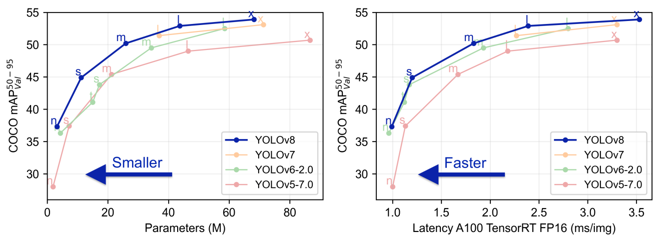


Fig. 1: Yolo-comparison-plots.

From Fig. 29 you can see the official results of mAP, number of parameters and FLOPs tested on the COCO Val 2017 dataset. It can be seen that the accuracy of YOLOv8 is very much improved compared to YOLOv5, but the corresponding number of parameters and FLOPs for N/S/M models have increased quite a lot, and it can also be seen from the above figure that most of the model inference has become slower compared to YOLOV5.

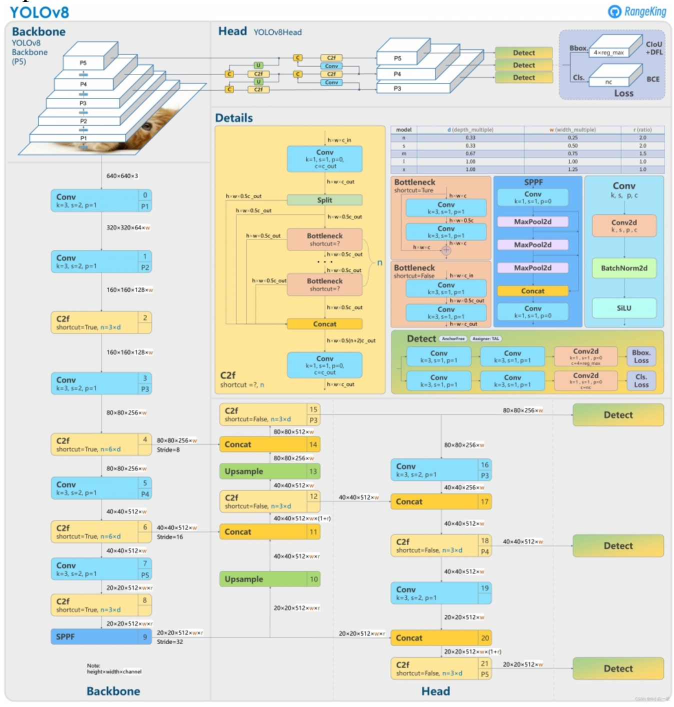


Fig. 2: YOLOv8's structure diagram.

In short YOLOv8 is an efficient algorithm that includes image classification, Anchor-Free object detection, and instance segmentation. the design of the detection part refers to a large number of excellent and up-to-date YOLO improvement algorithms, and implements a new SOTA. This may be the best YOLO model to date.

E. Detection – Feature extraction with SVM classification model

Many past approaches have solved the object detection problem using a single feature descriptor such as HOG along with a classification model. This method is widely used to solve the task until the introduction of deep learning models. Feature descriptors are often used to classify different types of objects, Hog is mainly used to identify objects with different shapes [8] wile LBP is used to classify objects with different textures [9].

F. Detection – DEtection TRansformer (DETR)

Carion et al. [10] present a new object detection method, DEtection TRansformer (DETR), which eliminates the need for many hand-designed components by treating detection as a direct set prediction problem. Utilising a bipartite matching-based global loss and a transformer encoder-decoder architecture, DETR directly outputs predictions, considering both object relations and global image context(Fig. 3). DETR not only matches Faster RCNN's performance on the COCO dataset but also easily generalises to produce superior panoptic segmentation [11].

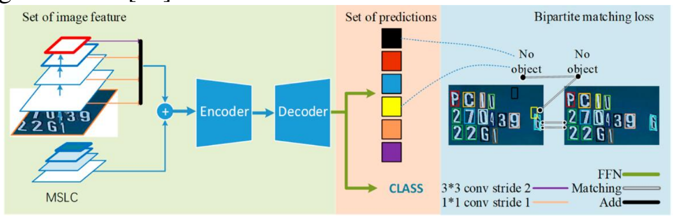


Fig. 3: DEtection TRansformer.

III. Methods

A. Detection – Feature extraction with SVM classification model

A traditional approach using feature extraction is used to detect objects within the image. The task given involves training a model to detect penguins and turtles, given the unique shape and pattern of these animals feature extraction would appear to be a good candidate for predicting and localising it within an image. A novel method involving two feature descriptors will be tested with the given dataset.

Two feature descriptors will be used for our method, HOG (Histogram of Oriented Gradients) and LBP (Local Binary Pattern). HOG is a feature extraction method used to capture local gradient information from an image. We can use this to describe the shape and structure of an object within an image. Another descriptor used is LBP, it is a method that uses the intensity level of a pixel to determine the surrounding pattern. Using a combination of HOG and LBP we can extract the shape and texture of objects within the image and provide the model a unique characteristic about the object we are trying to detect.

Our chosen classification model is SVM (Support Vector Machine), it is a popular machine learning algorithm that is used for binary classification. This is achieved through training where are used by the model to determine an optimal line that can divide the data into two equal classes. After training we can supply the model with a test data that will result in a prediction with a label of 1 or 0.

1) Method: Images in our given dataset contains exactly one object, either a penguin or turtle. Negative dataset of samples not containing any animals will to be created by extracting the background from the original datasets. Both the positive and negative dataset will be labelled and have their features extracted for model training. Using a sliding window technique which involves iterating through the image in smaller sections, we will used the trained model to predict if there exists an object within the model. Using the result and the coordinates of the window, we can then create a heat map that will assist in isolating regions of interest and remove duplicate predictions by assigning overlapping windows into the same region. This region will then be used to get our final boundary box to localise our object.

B. Detection – DEtection TRansformer (DETR)

With this method, we first attempted to create a Hugging Face "Database" which seemed to make data augmentation and training simpler but facing some challenges that led us back to COCO formatting. The annotations had to be altered however, for this approach. We created an annotation file containing both the image information and annotations in one dictionary.

The dataset's sanity is checked by viewing the bounding boxes on sample images. Inference was tested with a pre-trained version of the model as well. Given an image, it applies a backbone CNN like ResNet to generate features, which, along with learned positional encodings, are processed by a transformer encoder-decoder. DETR uses a fixed set of learned object queries to identify objects in the image. Each query yields a prediction, including a class and a bounding box. A unique feature is the bipartite matching loss function, minimizing cost across all matches between predicted and ground truth boxes. In inference, DETR outputs a fixed number of detections with bounding boxes, class labels, and confidence scores (Fig. 4).

Integrating Spatially Modulated Co-Attention (SMCA) into DETR could enhance its efficiency. Enhancing DETR with Spatially Modulated Co-Attention (SMCA), focusing on initial, bounding box locations, can boost its efficiency. Considering this regression-aware approach in future projects can significantly improve object detection [12].

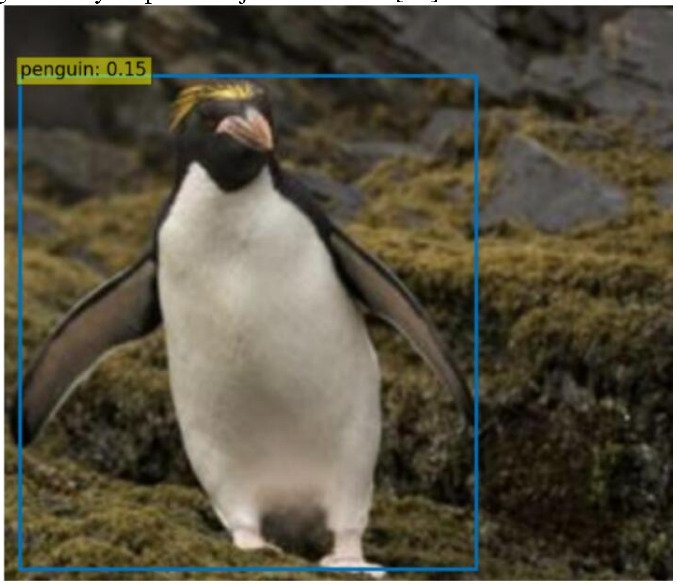


Fig. 4: DEtection TRansformer.

C. Classification

In the preparation of the classification part of the project, we analyze and extract some features of Penguins vs turtles. We found that the amount of dataset is 500 images with 2 labels which indicates that overfitting becomes the main problem in deep learning since a small dataset. Data Augmentation is a useful and effective method for reducing overfitting [1].

In addition, we also took research on the classification of similar datasets to ours. VGG16 [4] (a basic CNN model) and ResNet 50 [5] were used in finding COVID-19 from Chest X-rays [2] with more than 80% accuracy. For solving the main problem of the training model, transfer learning is another frequently used method especially based on small datasets [3].

Another model, EfficientNetV2 [6], was introduced in 2021 and it already gets some good results(generally are top) on some ImageNet and some specific kinds of datasets.

Combining the above information, we used VGG16, ResNet50, and EfficietNetV2 as the base model and apply both data augmentation and transfer learning to train them. The top layer of the base model is removed and the `GlobalAveragePooling2D()`, Dropout, and Dense layers are added for avoiding overfitting. Investigating the datasets, we found there are some images are clear for finding the main part (Fig. 5) which is better for using L2 regularization, and some are not (Fig. 6) which is better for using L1 regularization.

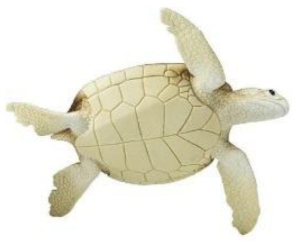


Fig. 5: Clear subject and background.

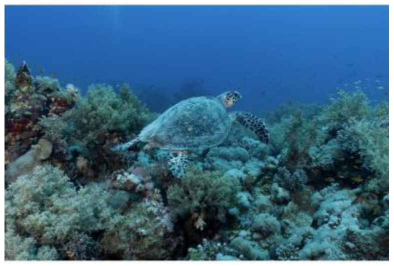


Fig. 6: Unclear subject and background.

So, to further avoid overfitting, elastic regularization (combining with L1 and L2) was used in the Dense layer and learning rate with hyperparameters.

At first of the training of hyperparameters, the range of hyperparameters are L1 & L2: [0.01:0.03] with step 0.005, the learning rate is [1e-2, 1e-3, 1e-4], and the best parameter in this range for EfficientV2B0 is L1 & L2: 0.02 with learning rate: 1e-3, and the history of training is stored in directory 'hype'. In the third train of hyperparameters, the range of hyperparameters (L1 & L2) is bounded with [0.015, 0.02, 0.025] with a maximum of 9 trials which would cover all possible combinations of these 2 regularizations.

D. Classification & Detection By YOLOv8

1) Label Conversion: In order to use the COCO dataset for training the YOLO algorithm, we need to convert the annotation information of the COCO dataset into labels in YOLO format. In this process, we need to convert the location information of the target from COCO format to YOLO format and write the category ID and location information of the target to a text file with the name of the ID of the image.

2) Image Preprocessing: In machine learning, the size and quality of the dataset is critical to the training and performance of the model. The larger the dataset, the more features and patterns the model is able to learn, which improves the accuracy and generalization of the model. At the same time, the quality of the dataset also affects the performance of the model because the model needs to learn the correct features and patterns from the data.

We performed data enhancement on the provided images of penguins and turtles, including adding Gaussian noise and flipping the images. This was done to expand the training set and increase the size and diversity of the dataset, thus improving the accuracy and generalization of the model. The training set is then increased from the original 500 to 1000 images.

Specifically, data augmentation can help the model learn more features and patterns, thus improving the robustness and generalization ability of the model. For example, adding Gaussian noise can make the model more robust as it learns the ability to recognize targets in noisy environments. Flipping the image allows the model to better adapt to different viewpoints and orientations, which improves the model's accuracy and generalization ability.

3) Training details: We used the YOLOv8n model for training on the animal dataset. During training, we used the following parameters:

- 'imgsz=640': specifies the size of the input image as 640x640 pixels.
- 'epochs=100': Specifies that the number of training rounds is 100. A smaller number of training rounds may lead to model underfitting, i.e., the model does not adequately learn the features of the dataset. Whereas a larger number of training rounds may lead to model overfitting, i.e., the model remembers the training data too much and fails to generalize to new data. Therefore, choosing a number of 100 training rounds may be a balance between adequately training the model while avoiding overfitting
- 'batch=8': Specifies that the size of each batch is 8. It determines how often the model is updated during each training session. A smaller batch size may result in a faster training process, but it may also result in an unstable training process because only a small amount of data is used for each update. Larger batch sizes may result in a more stable training process, but may also require more memory and may fall into local optima. The choice of 8 as the batch size may have been a decision made with both training speed and stability in mind.

During the training process, we used the Adam optimizer and the cross-entropy loss function. During training, we performed data enhancement operations such as random flipping, random scaling and random cropping on the input images to increase the diversity and size of the dataset. We also normalized and normalized the input images to better fit the input requirements of the model.

During the training process, we used a validation set to evaluate the performance of the model. We used the mAP (mean Average Precision) metric to evaluate the accuracy and generalization ability of the model. After the training was completed, we tested the model using the test set and calculated the accuracy and generalization ability of the model.

By adjusting the training parameters and data enhancement operations, we finally obtained a model with good accuracy and generalization ability that can be used for target detection and recognition of penguin and turtle images.

IV. Experimental Results

A. Detection – Feature extraction with SVM classification model

From the results we can see that although the SVM model has a respectable 80% accuracy (Fig. 8) in predicting our objects within the image, we can see that our result using this method lacked accuracy indicated by the low IOU (intersection over union) value (Fig. 9). There are two possible reasons for this result, either our predicted boundary box is too large, or the predicted boundary is not correct and there is low intersection between the predicted box and the true result indicated by the annotations. To further isolate the issue another metric is used to evaluate the method, a distance value is calculated (Fig. 9) that indicates how far the centre of the predicted box is from the centre of the true box. The high value indicates that the seconds reason is a major factor in the poor IOU value.

B. Detection – DEtection TRansformer (DETR)

By carefully tuning DETR's hyperparameters and optimizing our dataset format, we achieved enhancements in the prediction results, as depicted in the graph below.

By fine-tuning the learning rate we gradually decreased the learning rate over epochs, preventing drastic changes in model parameters and enhancing its learning efficacy. Ideally a learning rate scheduler should have been implemented that hopefully in future endeavours we will attempt. Simultaneously, we performed data augmentation and normalized our dataset, which enhanced its generalization capabilities. The dataset was formatted to ensure the presence of diverse object instances, improving the model's ability to handle a variety of scenarios.

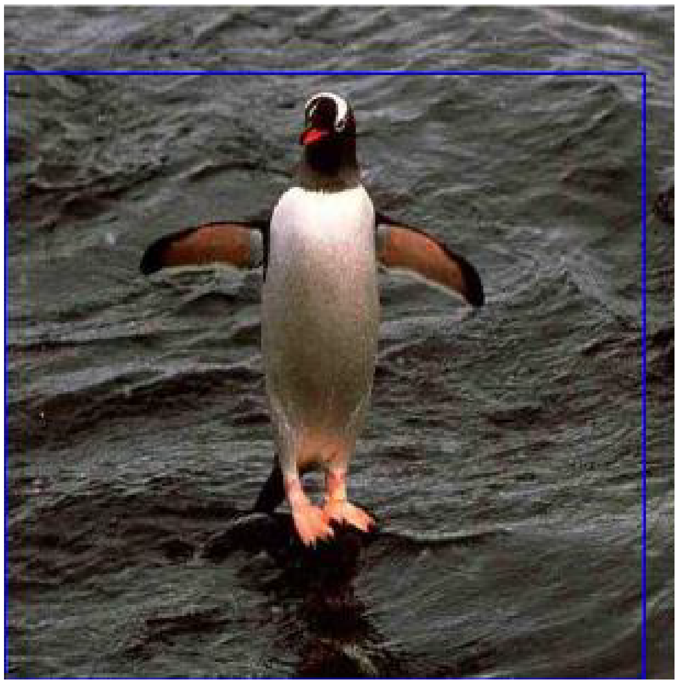
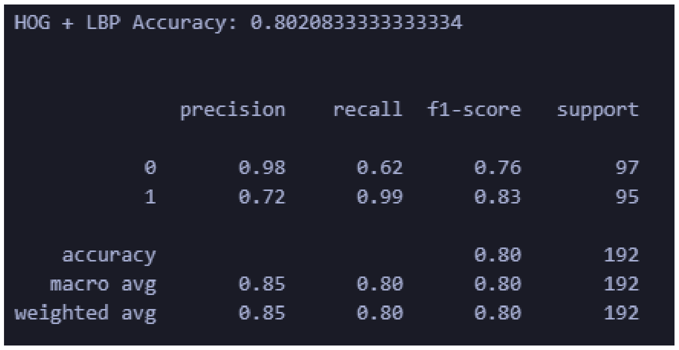


Fig. 7: SVM: Sample Result image.



```
HOG + LBP Accuracy: 0.8020833333333334


              precision    recall  f1-score   support

           0       0.98      0.62      0.76        97
           1       0.72      0.99      0.83        95

    accuracy                           0.80       192
   macro avg       0.85      0.80      0.80       192
weighted avg       0.85      0.80      0.80       192
```

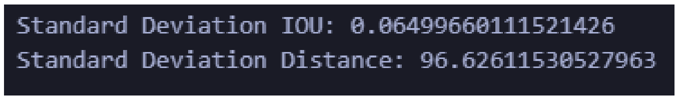
Fig. 8: SMV model evaluation.

C. *Comparison among VGG16, ResNet50 and EfficientV2B0*

1) Model Design: The hyperparameters (EfficientNetV2B0) indicate that 0.02 is the optimal regularization rate of L1 and L2 in the range of [0.015:0.025] and I would use this parameter to train all these three models (Fig. 11).

During the training process of VGG16, the optimizer of it changed from Adam to SGD since Adam is not working for VGG16 and there is no effect on the model of VGG16

2) Model architecture: The content of these images below shows the internal structure of the model



```
Standard Deviation IOU: 0.06499660111521426
Standard Deviation Distance: 96.62611530527963
```

Fig. 9: IOU and Distance values.

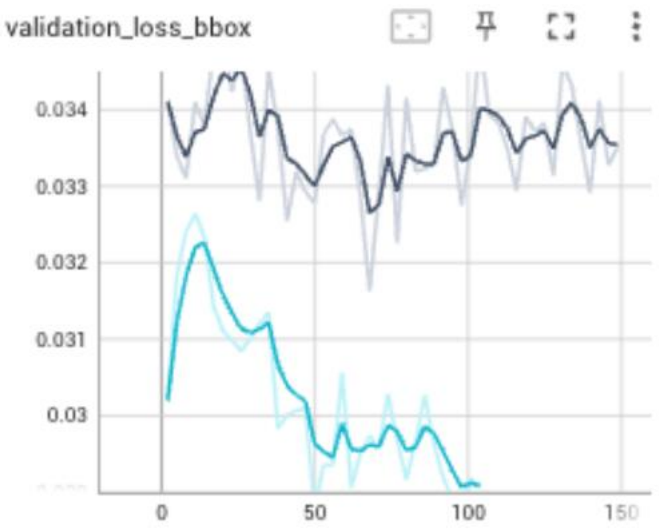
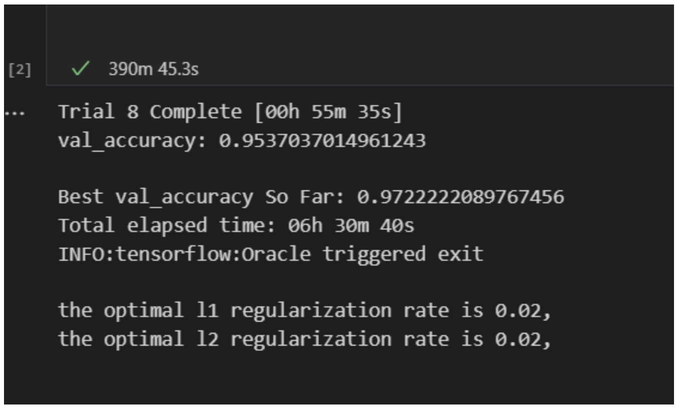
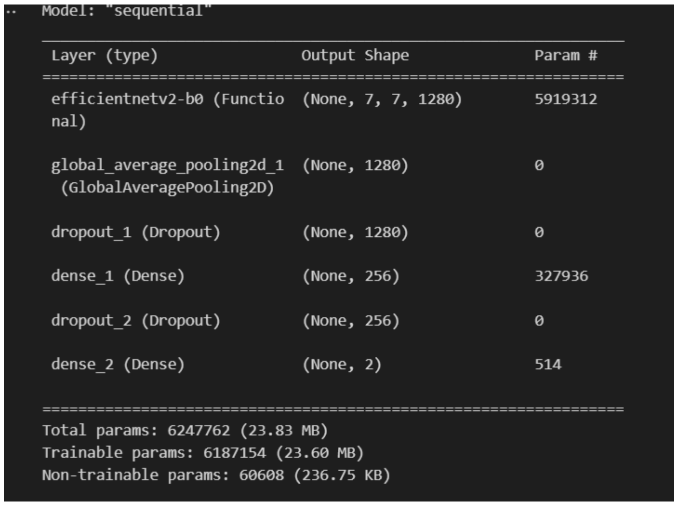


Fig. 10: DETR: prediction results.



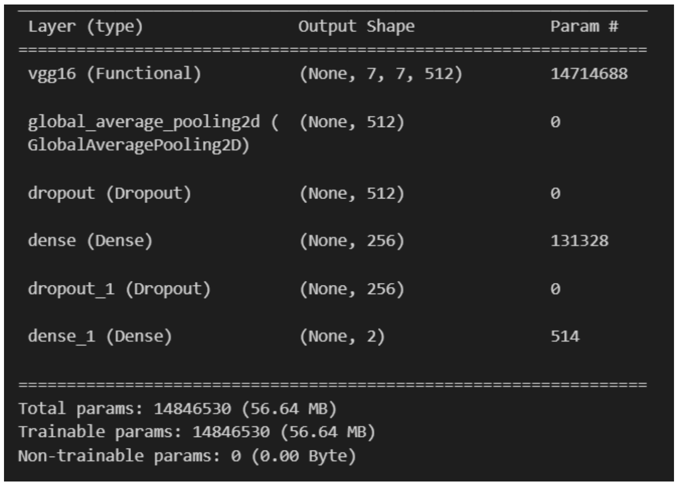
```
[2]  ✓ 390m 45.3s

...  Trial 8 Complete [00h 55m 35s]
     val_accuracy: 0.9537037014961243

     Best val_accuracy So Far: 0.9722222089767456
     Total elapsed time: 06h 30m 40s
     INFO:tensorflow:Oracle triggered exit

     the optimal l1 regularization rate is 0.02,
     the optimal l2 regularization rate is 0.02,
```

Fig. 11: Program Outputs.



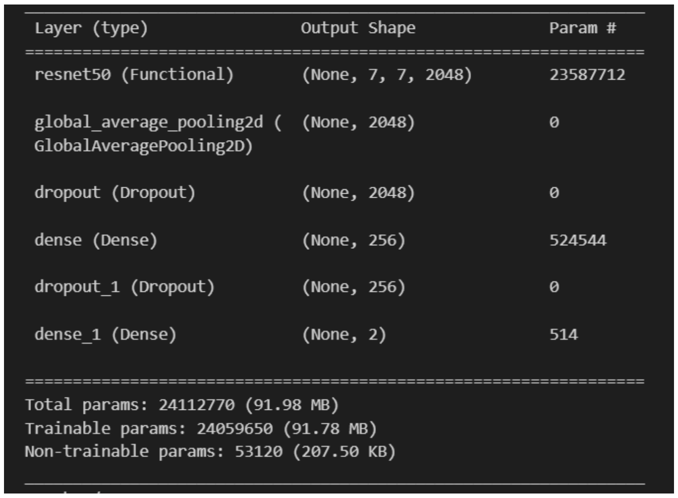
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetv2-b0 (Functio  (None, 7, 7, 1280)        5919312
 nal)

 global_average_pooling2d_1  (None, 1280)              0
  (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 1280)              0

 dense_1 (Dense)             (None, 256)               327936

 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 2)                 514

=================================================================
Total params: 6247762 (23.83 MB)
Trainable params: 6187154 (23.60 MB)
Non-trainable params: 60608 (236.75 KB)
```

Fig. 12: EfficientNetV2B0.



```
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688

global_average_pooling2d (   (None, 512)               0
GlobalAveragePooling2D)

dropout (Dropout)            (None, 512)               0

dense (Dense)                (None, 256)               131328

dropout_1 (Dropout)          (None, 256)               0

dense_1 (Dense)              (None, 2)                 514

=================================================================
Total params: 14846530 (56.64 MB)
Trainable params: 14846530 (56.64 MB)
Non-trainable params: 0 (0.00 Byte)
```

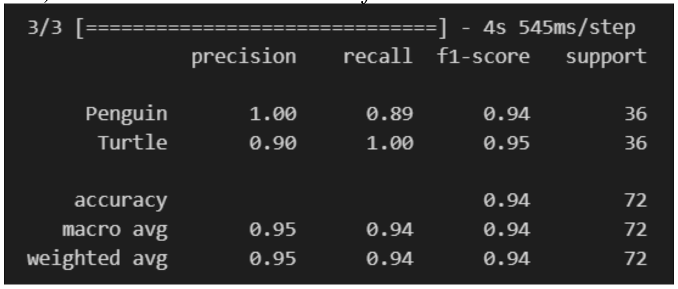
Fig. 13: Vgg16.



```
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712

global_average_pooling2d (   (None, 2048)              0
GlobalAveragePooling2D)

dropout (Dropout)            (None, 2048)              0

dense (Dense)                (None, 256)               524544

dropout_1 (Dropout)          (None, 256)               0

dense_1 (Dense)              (None, 2)                 514

=================================================================
Total params: 24112770 (91.98 MB)
Trainable params: 24059650 (91.78 MB)
Non-trainable params: 53120 (207.50 KB)
```

Fig. 14: ResNet50.

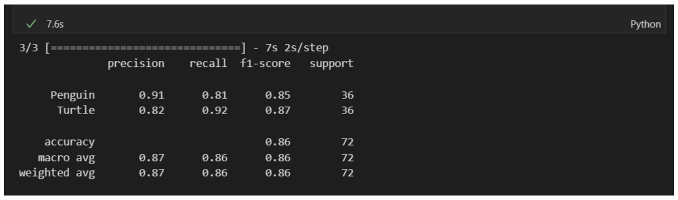
3) Model Performance:



```
3/3 [==============================] - 4s 545ms/step
              precision    recall  f1-score   support

     Penguin       1.00      0.89      0.94        36
      Turtle       0.90      1.00      0.95        36

    accuracy                           0.94        72
   macro avg       0.95      0.94      0.94        72
weighted avg       0.95      0.94      0.94        72
```

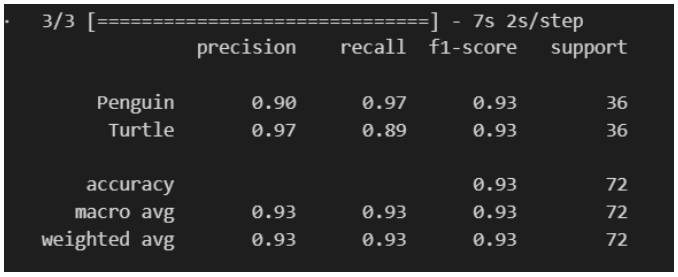
Fig. 15: EfficientNetV2B0 Performance.

EfficientNetV2B0: the hyperparameters tuner run above is an accuracy of 3 executions which has a validation rate of 0.9722



```
✓ 7.6s                                                          Python
3/3 [==============================] - 7s 2s/step
              precision    recall  f1-score   support

     Penguin       0.91      0.81      0.85        36
      Turtle       0.82      0.92      0.87        36

    accuracy                           0.86        72
   macro avg       0.87      0.86      0.86        72
weighted avg       0.87      0.86      0.86        72
```

Fig. 16: VGG16: Performance.



```
3/3 [==============================] - 7s 2s/step
              precision    recall  f1-score   support

     Penguin       0.90      0.97      0.93        36
      Turtle       0.97      0.89      0.93        36

    accuracy                           0.93        72
   macro avg       0.93      0.93      0.93        72
weighted avg       0.93      0.93      0.93        72
```

Fig. 17: ResNet50: Performance.

ResNet50: As shown above

4) Confusion matrix: The content of these images below shows the Confusion matrix of the model

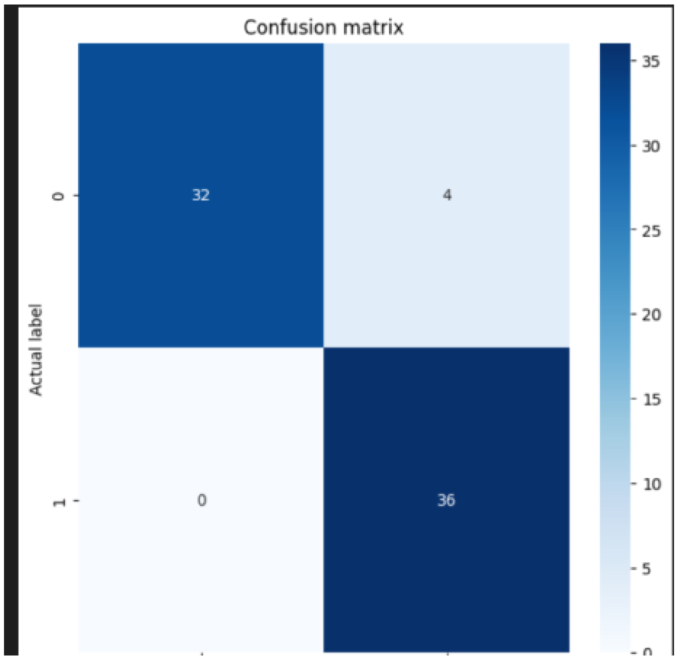


Fig. 18: EfficientNetV2B0: Confusion matrix.

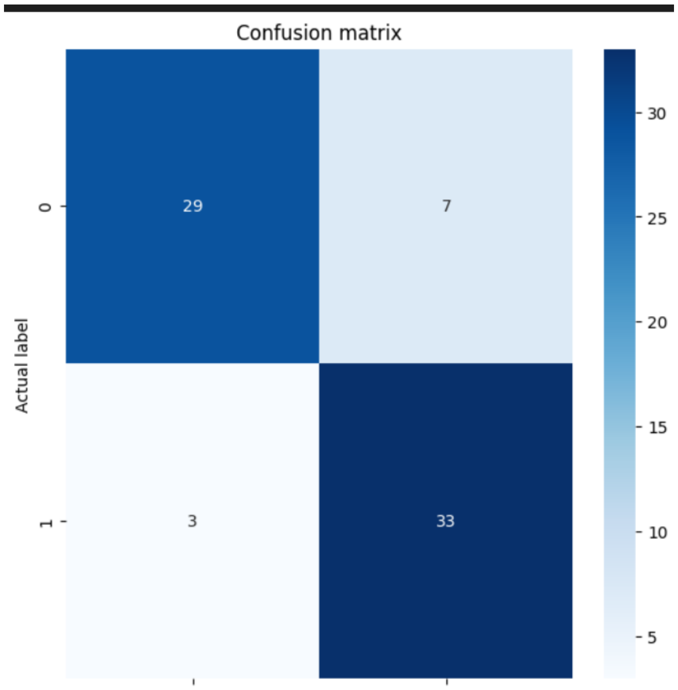


Fig. 19: VGG16: Confusion matrix.

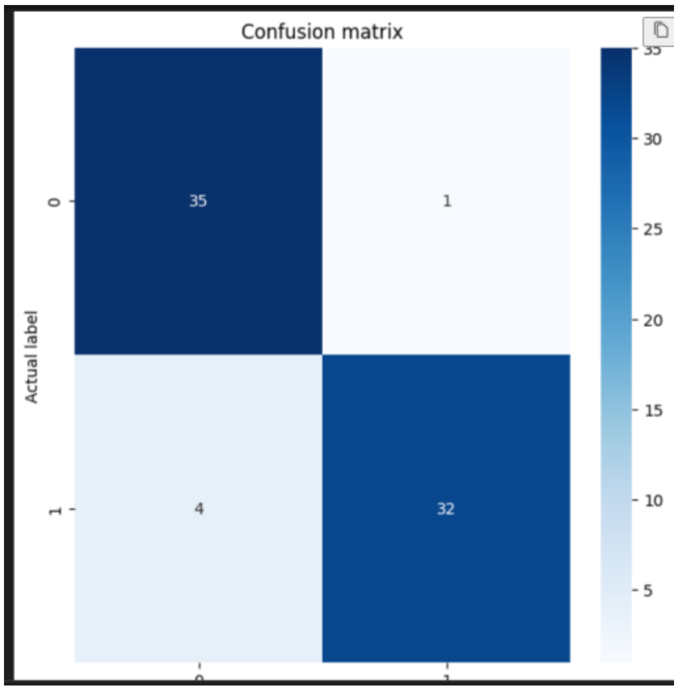


Fig. 20: ResNet50: Confusion matrix.

5) Learning curve: The content of these images below shows the Learning curve of the model

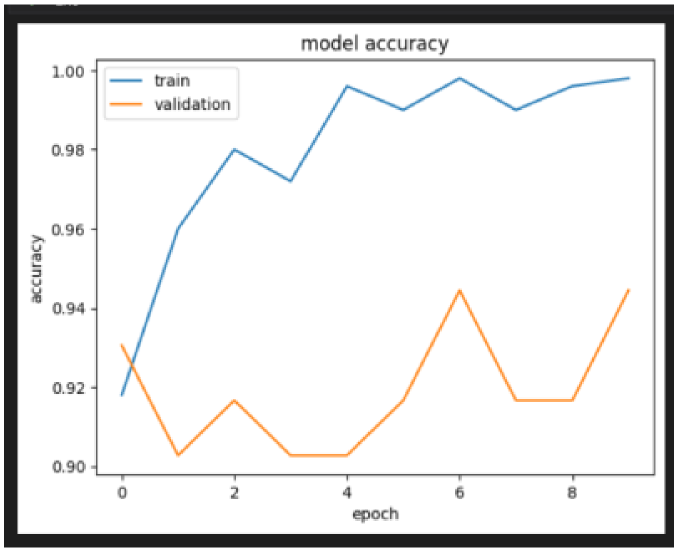


Fig. 21: EfficientNetV2B0: Accuracy

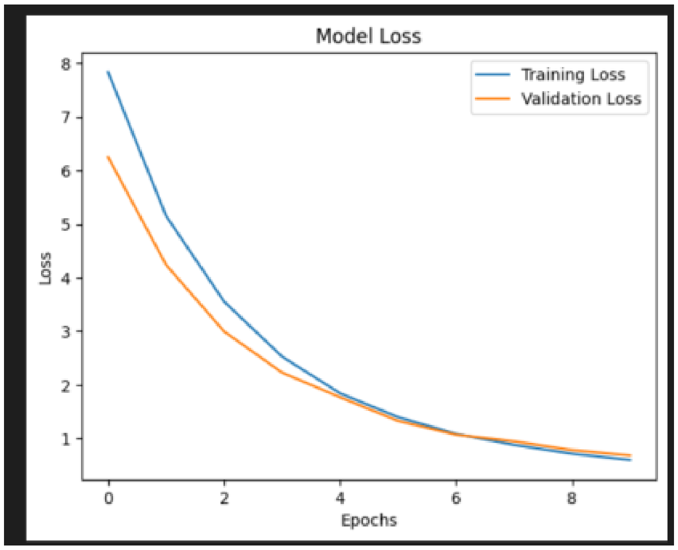


Fig. 22: EfficientNetV2B0: Loss

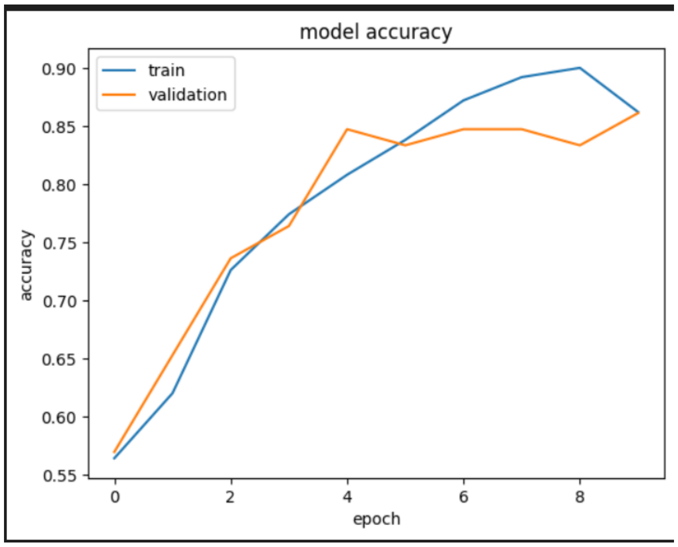


Fig. 23: VGG16: Accuracy

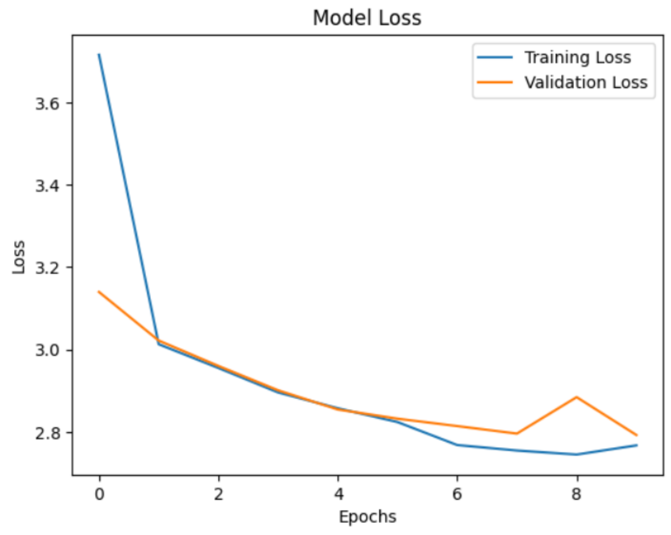


Fig. 24: VGG16: Loss

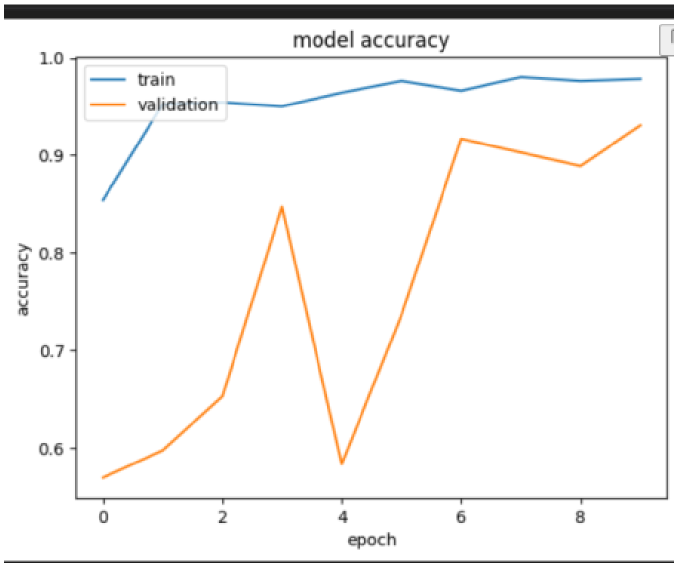


Fig. 25: ResNet50: Accuracy

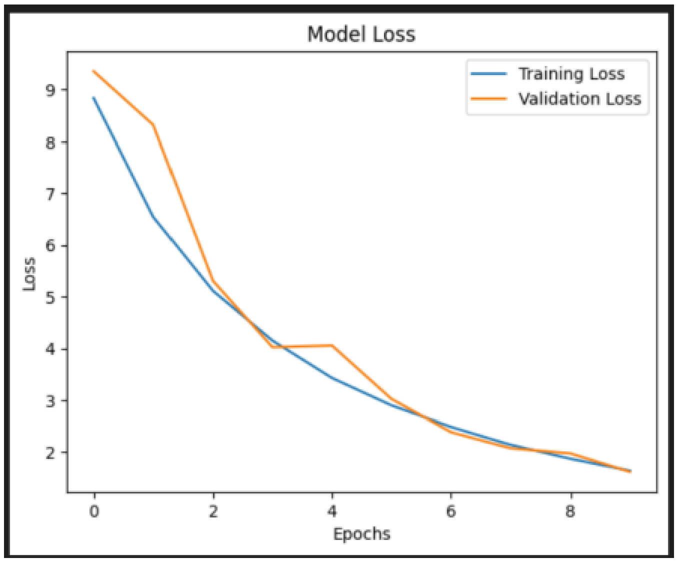


Fig. 26: ResNet50: Loss

D. *YOLOv8*

1) Model Performance: The following image shows the prediction results after YOLOv8 training

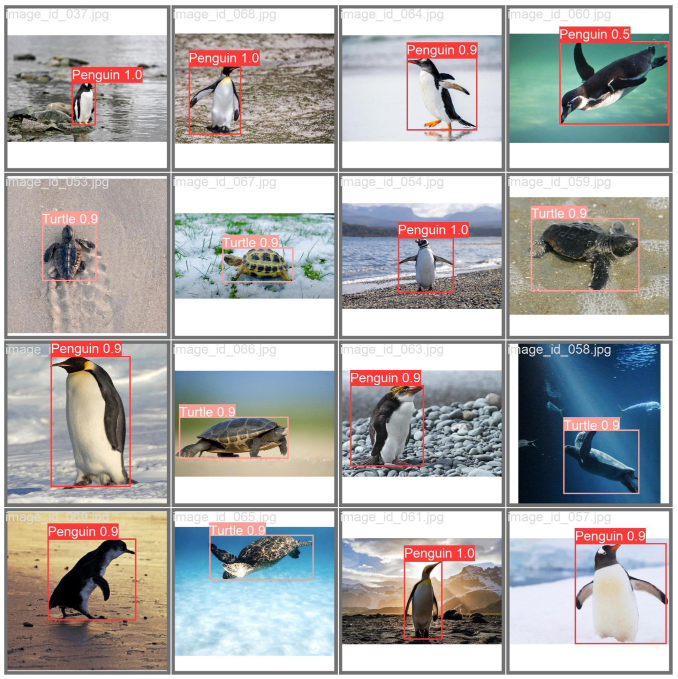


Fig. 27: YOLOv8: Predicted results 1.

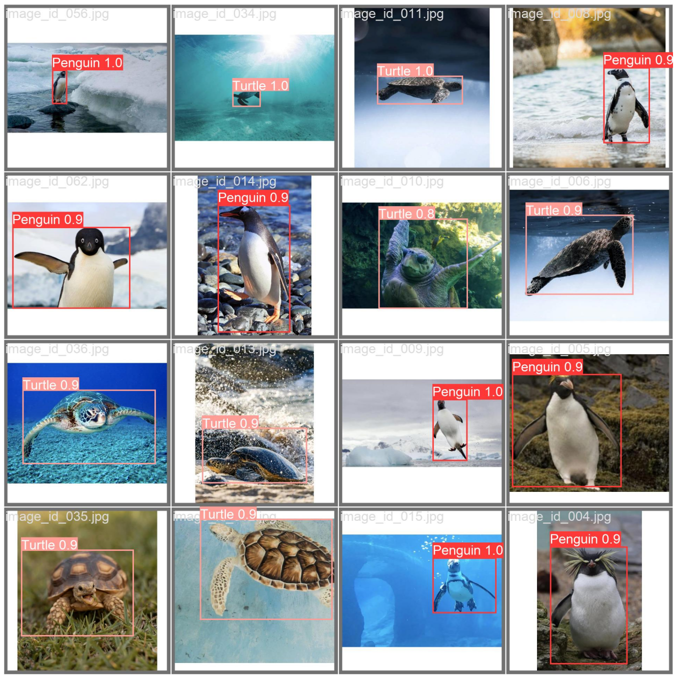


Fig. 28: YOLOv8: Predicted results 2.

2) Confusion matrix: Fig. 29 shows the Confusion matrix for the penguins and turtles and the background for the classification task.

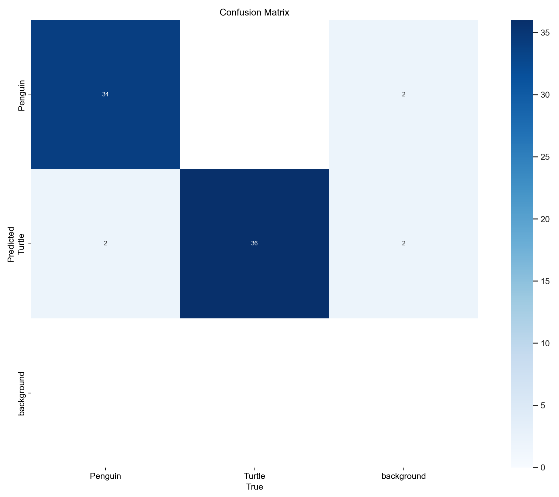


Fig. 29: YOLOv8: Confusion matrix.

3) Learning curve: The following images show the results of the performance analysis of YOLOv8 after training

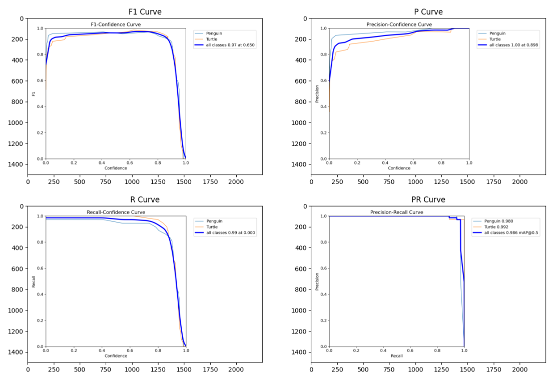


Fig. 30: YOLOv8: Curve.

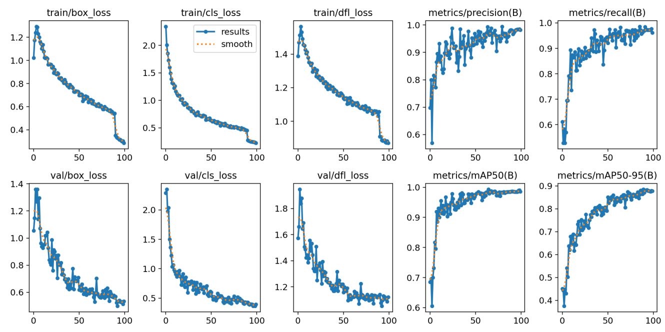


Fig. 31: YOLOv8: Learning Results.

V. Discussion

A. Detection – Feature extraction with SVM classification model

Given the poor results of the traditional method of feature extraction and binary classification, it appears that there are many factors that contribute to this. One major factor could be the lack of good negative samples used for the binary classification. Since the background is extracted from the original dataset it will lack a variety in characteristics as some images contain simple backgrounds with minimal features. This can create bias in the model which will result in false positives and false negatives. Another factor could be the complexity of the objects themselves as we are detecting both penguins and turtles. We are required to create a model that predicts the location of both animals. However, given the differences between the two animals in shape and texture it is hard for a single model to predict both animals at the same time.

A few improvements will be proposed to improve the performance of the HOG, LBP + SVM model. Firstly, a better negative dataset should be sampled that contains larger images and better quality than the current extracted set. A good negative dataset will assist in a better classification and reduce the number of false positives and negatives. Another improvement would to the use of two models instead of one to classify penguins and turtles separately. This will eliminate the confusion caused by the combination of two sets of uniquely characteristics and avoid the model classifying both types of animals at the same time.

B. Classification

Model architecture: the total params in ENV2 is around 6 million, and the amount of params of ResNet50 is 4 times more than that. Due to the difference in the number of params, the training time of ENV2 is also much quicker than ResNet50. The score of ENV2 is slightly better than ResNet50. The train learning curves of ENV2 and ResNet50 change stably, but the validation learning curves of ResNet50 indicate that the result is quite varied and the loss rate of ResNet50 is a little bit higher than the ones of ENV2. The performance of vGG16 is the most normal in these three models with the worst efficiency, the highest validation loss and the the lowest accuracy. Considering the validation accuracy of the first epoch, the accuracy of ENV2 is around 0.93 but the accuracy is around 0.55 for both VGG16 and ResNet50 which indicates that the parameters in ENV2 are quite suitable for classifying this task.

C. Classification and Detection error in YOLOv8

In Fig. 32, the model incorrectly predicts the reflection in the water as a penguin.

The reasons for this may be:

1) High target similarity: penguins and turtles may have similar appearance features, such as color, shape, etc., in certain viewpoints or scenes. This similarity may make it difficult for the algorithm to accurately distinguish between them, thus detecting two targets at the same location and giving two possible labels.
2) Model Uncertainty: While the YOLO algorithm tries to learn to distinguish between different classes of targets during the training process, sometimes the model may feel uncertain in the testing phase when faced with some complex or uncommon situations. This uncertainty can lead to multiple possible predictions.

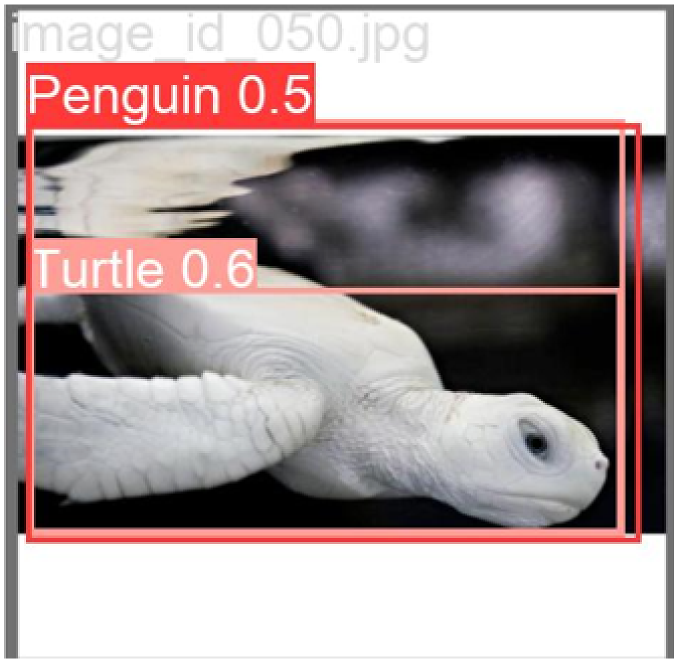


Fig. 32: YOLOv8: Error 1.

In Fig. 33, although the model successfully predicts turtles, it gives me two bounding boxes of different sizes. There are a few possible reasons for this:

1) The YOLO algorithm is a single-stage target detection algorithm that detects multiple targets simultaneously in an image and gives a bounding box and category probability for each target. To detect targets at different scales, YOLO uses feature maps at multiple scales to predict the location and category of the target.
2) In an image of a sea turtle, there may be multiple scales of feature maps to detect the turtle, and each scale of feature maps may result in a different size bounding box to represent the turtle's location. These bounding boxes of different scales may overlap to some extent, resulting in multiple boxes at the same location.
3) In addition, since the YOLO algorithm is based on an Anchor Box for target detection, different anchor box sizes and shapes may also result in different sized bounding boxes being predicted, further increasing the likelihood of multiple boxes appearing.

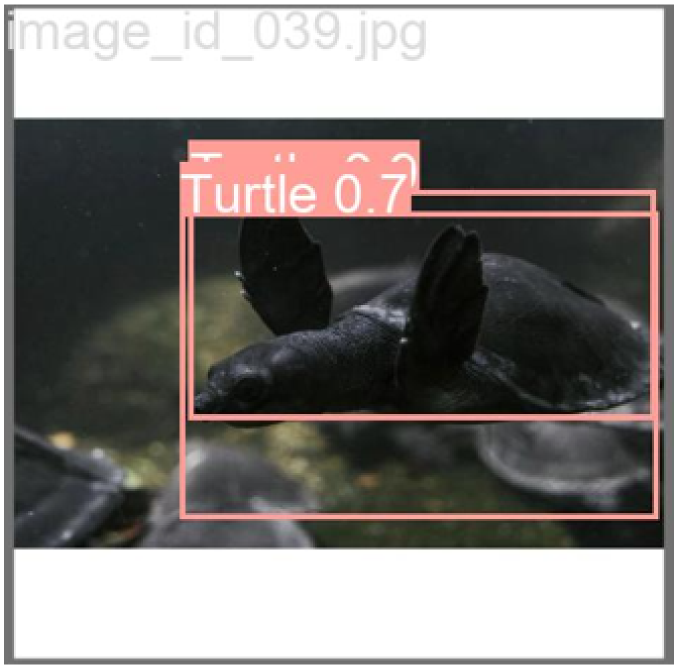


Fig. 33: YOLOv8: Error 2.

VI. Conclusion

In conclusion our team have tried various approaches in tackling the task of detecting and classifying penguins and turtles. Due to the complex nature of the task, it is concluded that the best approaches involved breaking the task into smaller steps that can be solved quickly and effectively. Although some methods such as the traditional detection method was considered state of the art, it is later shown that many newer approaches have supposed the method with better performance and computing time. Through this challenge we can understand the difficulty in the task and see the effectiveness of current state of the art technologies such as deep learning.

References

[1] Shorten, C. and Khoshgoftaar, T.M., 2019. A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), p.60. https://doi.org/10.1186/s40537-019-0197-0
[2] Hall, Lawrence; Goldgof, Dmitry; Paul, Rahul; Goldgof, Gregory M. (2020): Finding COVID-19 from Chest X-rays using Deep Learning on a Small Dataset. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.12083964.v3
[3] Yosinski, J., Clune, J., Bengio, Y. and Lipson, H., 2014. How transferable are features in deep neural networks?. In Advances in neural information processing systems (pp. 3320-3328). https://arxiv.org/abs/1411.1792
[4] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://arxiv.org/abs/1409.1556
[5] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). https://dx.doi.org/10.1109/cvpr.2016.90
[6] Tan, M., Pang, R. and Le, Q.V., 2021. EfficientNetV2: Smaller Models and Faster Training. arXiv preprint arXiv:2104.00298. https://arxiv.org/abs/2104.00298
[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788). https://ieeexplore.ieee.org/document/7780459

[8] Amraee, S., Chinipardaz, M., & Charoosaei, M. (2022). Analytical Study of two feature extraction methods in comparison with deep learning methods for classification of small metal objects. Visual Computing for Industry, Biomedicine, and Art, 5(1). https://doi.org/10.1186/s42492-022-00111-6

[9] Islam, M. A., Yousuf, Md. S. I., & Billah, M. M. (2019, April 28). Automatic plant detection using hog and LBP features with SVM. Automatic Plant Detection Using HOG and LBP Features With SVM. https://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/1384

[10] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, May 28). End-to-end object detection with Transformers. arXiv.org. https://arxiv.org/abs/2005.12872

[11] Detr. DETR. (n.d.). https://huggingface.co/docs/transformers/main/model_doc/detr

[12] Gao, P., Zheng, M., Wang, X., Dai, J., & Li, H. (2021, January 19). Fast convergence of DETR with spatially modulated co-attention. arXiv.org. https://arxiv.org/abs/2101.07448