Comparative Study of Machine Learning Techniques for Object Detection and Classification of Turtles and Penguins

1st Bingyu Yang
Department of Engineering Faculty
UNSW Information Technology - COMP9517
Sydney, AU
z5328465@ad.unsw.edu.au

2nd Xiaojie Shi
Department of Engineering Faculty
UNSW Information Technology - COMP9517
Sydney, AU
z5333341@ad.unsw.edu.au

3rd Zhaoyuan Wang
Department of Engineering Faculty
UNSW Information Technology - COMP9517
Sydney, AU
z5373961@ad.unsw.edu.au

4th Rui Sun
Department of Engineering Faculty
UNSW Information Technology - COMP9517
Sydney, AU
z5391910@ad.unsw.edu.au

***Abstract*—This report presents a comprehensives study on the application of computer vision techniques for object detection and classification tasks. Turtles and penguins are the main objects of study in this scenario. This report focuses on exploring and comparing the effectiveness of different machine learning and deep learning methods, including K-Means, decision tree-based methods, Convolutional Neural Networks (CNNs) and advanced detection toolboxes like MMDetection and RTMDet. A diverse dataset of images containing turtles and penguins in various environments was used for evaluation. With multiple experiments of different methods on the dataset, this report discusses the role and performance of each method, considering accuracy, precision, recall, and computation efficiency. Through this study, valuable insights were obtained for the suitability of these techniques for the computer vision tasks. The findings in this report have potential benefits for wildlife monitoring and other automated detection and classification systems.**

I. Introduction

Computer vision has significantly transformed the realm of automated systems, with numerous applications ranging from autonomous driving to wildlife monitoring. Among the myriad of tasks that computer vision can perform, object detection and classification stand out as key components for understanding visual data. This report focuses on the application of various computer vision techniques to detect and classify two distinct classes of animals: turtles and penguins.

In this report, we explore multiple machine learning techniques, including but not limited to Convolutional Neural Networks (CNNs) and decision tree-based methods for the task at hand. We apply these techniques to a comprehensive dataset comprising images of turtles and penguins in diverse settings. Each method's effectiveness is evaluated based on its accuracy, precision, recall, and computational efficiency.

The main contribution of this report is a comparative analysis of the role of different machine learning techniques in detecting and classifying sea turtles and penguins. In the following sections, we will demonstrate an understanding of the different literatures, delve into the research methodology, as well as present the experimental results and provide a comprehensive discussion of the findings, culminating in a conclusion that is closer to reality.

II. Literature Review

A. Traditional Machine Learning: Histogram of Oriented Gradients (HOG), K-Means & Random Forest

Among different methods utilized for feature extraction, Histograms of Oriented Gradients (HOG) has shown remarkable performance due to its ability to capture local image gradients effectively [1]. It achieves its goal by computing the gradient information of pixel points and then calculating the gradient direction and magnitude for each pixel point.

To achieve accurate classification using HOG features, various machine learning techniques were explored. One adopted in this study is to combine HOG with K-means clustering. K-means is a popular unsupervised clustering algorithm that groups data into clusters based on similarity [2]. By using K-means with HOG, the feature space can be effectively partitioned for the classification purpose.

Another effective classification method that has gained popularity in computer vision is Random Forest [3]. Random Forest is an ensemble learning technique that constructs multiple decision trees and vote based on each individual tree to improve classification accuracy. When dealt with HOG, Random Forest takes advantage of the discriminative power of HOG features and the ensemble approach to achieve robust and reliable classification results.

B. Detection Transformer

DEtection A novel approach to object detection called TRansformer (DETR) [4] was motivated by transformers' performance in set-based tasks. It formulates the detection of objects as a direct set prediction issue, without the requirement for labor-intensive elements like non-maximum suppression and anchor formation. DETR generates parallel predictions by using a transformer encoder-decoder architecture to reason about object relations and global context. Bipartite matching is used to guarantee distinct object predictions, increasing robustness. DETR achieves accuracy and runtime performance equivalent to more established techniques like Faster R-CNN. It sets itself apart from other detecotors by streamlining the detection pipeline, simplifying the model, and avoiding specialised libraries.

C. MMDetectiona and RTMDet

MMDetection: Open MMLab Detection Toolbox and Benchmark [5]: MMDetection is a toolbox integrating object detection and instance segmentation continuously developed and promoted by the MMDet team that won the COCO challenge in 2018. It is known for its highly flexible modular design, which helps users to personalise and combine the required components, and supports more than 20 frameworks and the methods from single-state to the multi-stage, as well as a training pipeline with hooking mechanism. Based on the testing of the model performance, it used balanced L1 loss as the regression less and adjusted the normalisation layers. In addition, it resizes the image scale to 1333 x 800 and adds a new hyper-parameter to control the ratio of negative samples to positive samples.

RTMDet: An Empirical Study of Designing Real-Time Object Detectors [6]: RTMDet is a real-time detector developed by the MMDet team in order to outperform the YOLO series by using a large convolution kernel to form the basic components and matching a considerable number of formations in the neck and head, and then by introducing soft labelling to improve the discriminative power of the cost matrix for high-quality matches, and to reduce the noise of the labelling assignments. This means that the convolutions layer is widened and deepened, while the neck adopts a pyramid structure similar to the backbone, with top-down and bottom-up propagation.

D. CNN and Faster-RCNN [7]

Convolutional Neutral Network model utilize the convolutional layer to extract target features, then using a single or multiple fully connected layers to make the final decision. Faster-RCNN is an extension of CNN, it includes two stages during detection. The first one is regional proposal, before extract key features, the input images are proposed, then fall into the CNN network and give us the final decision.

III. Methods

A. Traditional Machine Learning

Our strategy is to first use traditional machine learning methods mentioned in the course to extract features from the training set images. Then, based on these features, we perform the classification between penguins and turtles. Although we do not expect very accurate results, we want to try training and testing using traditional machine learning methods and use the results as a baseline. After that, we will explore new models and methods to continuously improve the accuracy of the results.

One observation we made is that the training set images do not have any labels in their names indicating whether the subject in each image is a penguin or a turtle. This suggests an unsupervised learning problem. We believe we need to use clustering algorithms to group the images in the training set, clustering similar images together without explicitly knowing whether they belong to penguins or turtles. In this case, we decided to use the k-means model. Since we only need to categorize the images into penguins or turtles, we can simply set k to 2 and assign each sample to the nearest cluster. Figure 1 clearly compares the original data and clustered data.

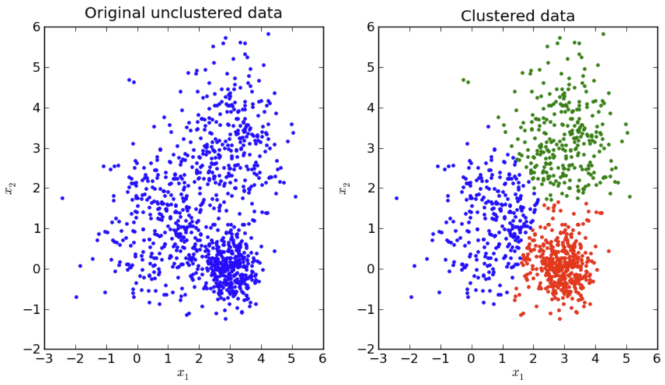


Fig. 1. K-Means

Before applying k-means, we need to extract features from the training set images. Here, we adopt the Histogram of Oriented Gradients (HOG) method. The principle of this method is to compute the gradient information of pixel points and then calculate the gradient direction and magnitude for each pixel point. To achieve better results, we pre-process the input training set images by converting them to gray-scale. However, we encountered a memory error when trying to create the train_features array for the first time. This is because the dimension of the HOG feature vector is very high and takes up a lot of memory. Therefore, we used Principal Component Analysis (PCA) to reduce the dimension of the feature vector. After multiple tests, we found that the following settings provide the best results for HOG:

- winSize: 64x128;
- blockSize: 16x16;

- blockStride: 8x8;
- cellSize: 8x8;
- nbins: 9.

B. Detection Transformer

DEtection TRansformer (DETR) is a deep learning method for object detection and segmentation. Inspired by transformer architectures, DETR treats detection as a set of prediction problems without the need for anchor boxes. It combines a CNN backbone for feature extraction with a transformer encoder-decoder for global context understanding. DETR directly predicts bounding boxes and class labels as a set, enabling end-to-end training and adaptability to varying object numbers. Its anchor box-free approach reduces complexity and improves efficiency.

We utilized DETR as the method to distinguish between penguins and turtles. By using DETR, we can directly predict bounding boxes and class labels for both penguins and turtles, without relying on anchor boxes. This approach not only simplifies the detection process but also enables efficient training and handling of varying object quantities in the input images. The end-to-end training of DETR allows it to effectively learn to detect both penguins and turtles in a unified manner, leading to accurate and robust results.

We use the annotations from the training dataset and group them by image ID for further processing. During the training process, we iterate through each image in the training dataset and sequentially load the image along with its corresponding annotations. Since the DETR model requires class IDs to be mapped to continuous integers starting from 0, we map class ID 1 to 16 and class ID 2 to 0.

C. RTMDet Detector

Real-time detector (RTMDet) [6] method can be divided into three components: the backbone, neck, and head, as illustrated in Figure 2.

Fig. 2. Illustration of the RTMDet Architecture

The Backbone network, designed to extract hierarchical feature maps from the input image, is built upon a CSP-blocks [8], a pyramid-structured layer, which is then connected to a neck that employs the same capacity formation.

The Neck network of RTMDet is responsible for the fusion and up scaling of these multi-scale feature maps produced by the Backbone. Layer, C3, C4 and C5, are transferred to the neck section and the neck constitutes about the same number of blocks as the backbone.The FPN accomplishes this via lateral connections and top-down pathways, effectively creating a rich, multi-scale feature representation suitable for detecting objects of varying sizes.

The final part of the RTMDet architecture is the detection head, which uses the fused feature maps to predict the class and bounding box coordinates of potential objects in the image. The detection head is designed to perform efficient and accurate detection, contributing to the real-time capabilities of our model.

In this method due to the large convolutional kernel used in RTMDet, there are two balances that need to be taken care of. The first balance is to increase the inference speed by widening the block to compensate for the reduced inference speed due to the introduction of large kernel and point-wise convolution. The second balance is by shifting the basic block of the backbone to the neck allowing the neck and thus better computation of multi-scale features.

D. Fast-RCNN

CNN and R-CNN are both popular methods for computer vision tasks, They utilize two stages to make the final decision.First it divides the object detection process into two stages: region proposal and feature extraction. Then by using the output dense layer, it gives us the final result.

For task1, we use FasterRCNN, which is a popular deep learning method with high accuracy and high testing-speed compared to other methods. For task 2, The CNN model was trained and evaluated on the turtles and penguins dataset. The final result of CNN model demonstrate the effectiveness of CNNs' skill of solving image classification problems.

The raw images were preprocessed to standardlize the input for the CNN model. The preprocessing steps involved resizing all images to a common resolution, also normalizing pixel values, and data augmentation to increase the diversity of the training data.

The MobileNet-Large Backbone we used in the model is designed for efficient processing on to achieve a good trade-off between accuracy and the model size. Also it combines FPN to enhance the performance of object detection network. This image is shown firgure 3.

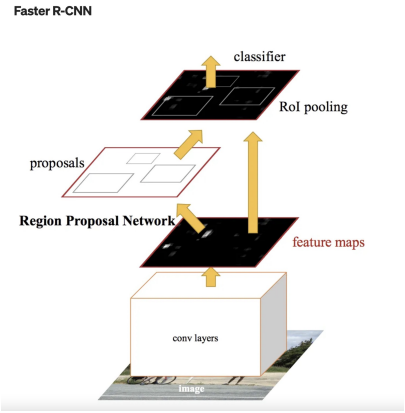


Fig. 3. Model structure of Faster-RCNN

Our CNN model used several convolutional laters with ReLu activation for non-linearity. Also a max-pooling layer with size (2, 2) to reduce the spatial dimensions the feature maps by selecting the maximum value over a(2,2)region. Then

we used a dropout layer to prevent over-fitting and a Flatten layer to flatten the data to one-dimensional vector.By using the last three fully connected layers, we mapped the flattened features to out final classification.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      896

 max_pooling2d (MaxPooling2  (None, 99, 99, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 97, 97, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 48, 48, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 23, 23, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 21, 21, 256)       295168

 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 256)       0
 g2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 512)         1180160

 max_pooling2d_4 (MaxPoolin  (None, 4, 4, 512)         0
 g2D)

 dropout (Dropout)           (None, 4, 4, 512)         0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 256)               2097408

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 3699009 (14.11 MB)
Trainable params: 3699009 (14.11 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. 4. Model structure of CNN

IV. Experimental Results

A. Traditional Machine Learning

Below are the accuracy values obtained using the HOG and k-means method in table I. We noticed that the accuracy of the prediction is just over 50% (38 out of 72 test images were predicted correctly), and the success rate for penguins' predictions is significantly higher than that for turtles.

TABLE I
Confusion Matrix Using HOG and K-Means

	Penguin	**Turtle**	**Total**
Predicted Penguin	23	21	44
Predicted Turtle	13	15	28
Total	36	36	72

- Accuracy: 52.78%
- Precision for penguins: 52.27%
- Recall for penguins: 63.89%
- Precision for turtles: 53.57%
- Recall for turtles: 41.67%
- F1-score for penguins: 57.53%
- F1-score for turtles: 46.81%

However, this level of accuracy is still relatively low. Therefore, we tried using the random forest method for classification. We set n_estimators to 100 and random_state to 42, and used default values for the other hyperparameters. In this case, we were able to increase the accuracy of the prediction to nearly 60% (43 images were predicted correctly). Below are the relevant results in table II.

TABLE II
Confusion Matrix Using HOG and Random Forest

	Penguin	**Turtle**	**Total**
Predicted Penguin	28	21	49
Predicted Turtle	8	15	23
Total	36	36	72

- Accuracy: 59.72%
- Precision for penguins: 57.14%
- Recall for penguins: 77.78%
- Precision for turtles: 65.22%
- Recall for turtles: 41.67%
- F1-score for penguins: 66.08%
- F1-score for turtles: 50.85%

we present our experimental results from applying two different machine learning methods to our data: K-Means and Random Forest. Both methods were used in conjunction with Histogram of Oriented Gradients (HOG) feature extraction in Figure 5 and 6.

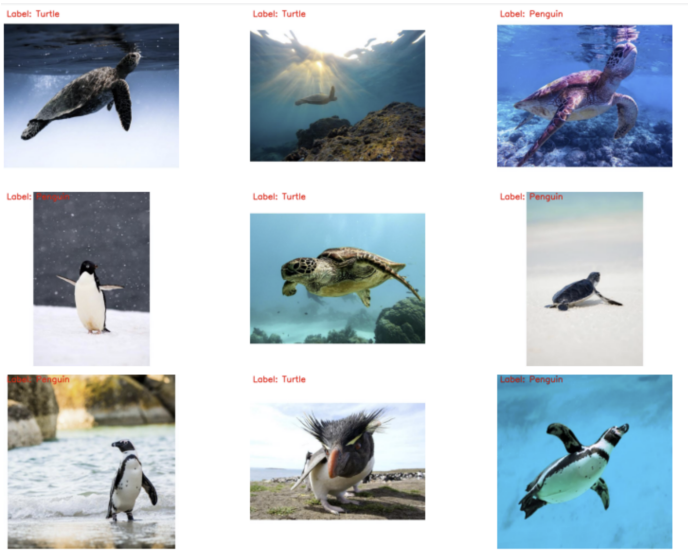


Fig. 5. Experimental Results for HOG & K-Means

B. Detection Transformer

We pass the preprocessed input data to the DETR model and compute the model's loss. We use backpropagation and an optimizer to update the model's parameters to minimize the loss function.

Throughout the training process, we record the average loss for each epoch and plot the loss values on a graph for subsequent analysis and visualization. The DETR model gradually learns feature representations and prediction capabilities for object detection and classification tasks, resulting in improved performance on the training dataset.

Below are the accuracy values obtained using the HOG and k-means method. We noticed that the accuracy of the prediction is just better than random guessing, but apart from the accuracy number, other numbers give us a lot of

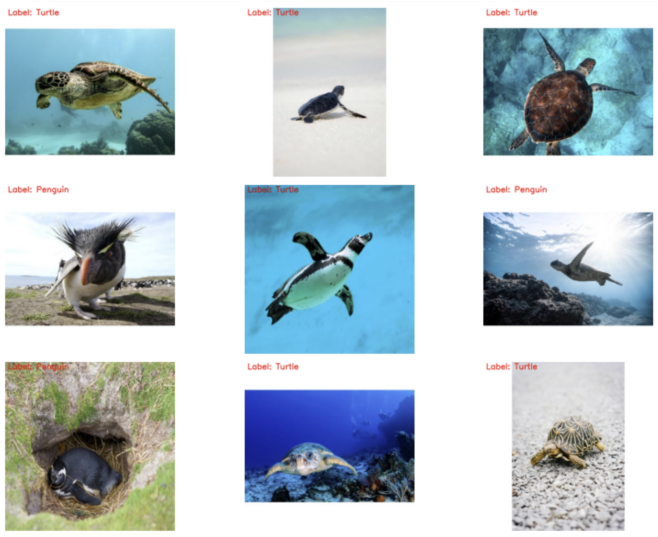


Fig. 6. Experimental Results for HOG & Random Forest

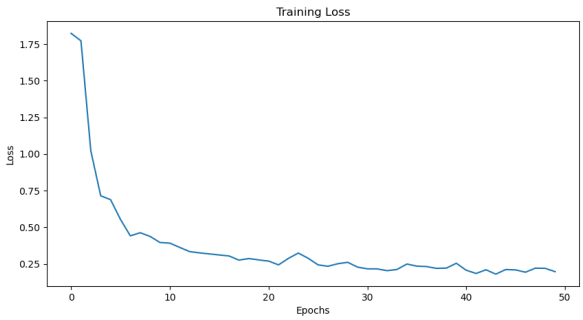


Fig. 7. train loss

information, especially the F1-Scores and Recall in table III and the confusion matrix in figure 8.

TABLE III
CONFUSION MATRIX USING HOG AND K-MEANS

	Penguin	Turtle	Total
Predicted Penguin	34	28	64
Predicted Turtle	2	8	10
Total	36	36	72

- Accuracy: 58.33%
- Precision for penguins: 54.84%
- Recall for penguins: 94.44%
- Precision for turtles: 69.39%
- Recall for turtles: 22.22%
- F1-score for penguins: 69.39%
- F1-score for turtles: 34.78%
- Average IOU: 0.1784
- Standard Deviation of IOU: 0.2016
- Average Distance: 133.0019
- Standard Deviation of Distance: 44.7619

Figure 9 illustrates the performance of our detection model. The model was trained using a variety of machine learning

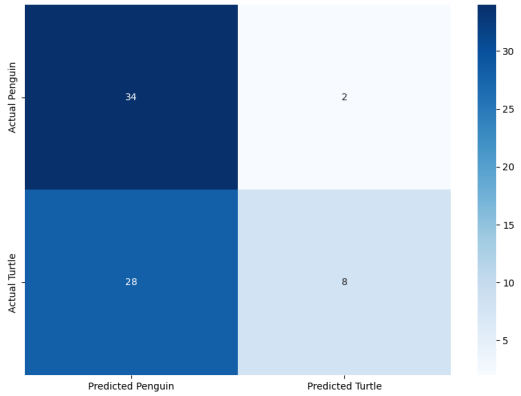


Fig. 8. DETR Confusion Matrix

algorithms and was evaluated on a separate test set. The results indicate that our model successfully identified the presence of turtles and penguins in a diverse range of images, despite variations in image quality, lighting conditions, and the animals' poses.

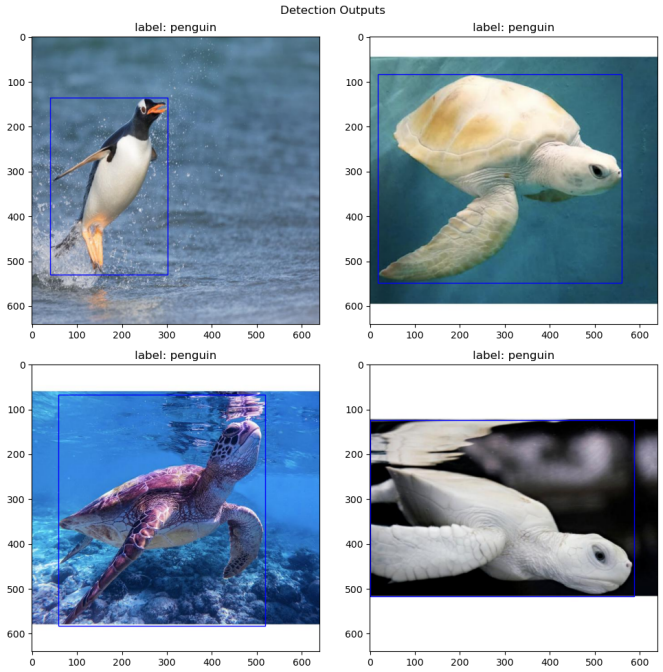


Fig. 9. Detection Outputs

After successfully detecting the target objects, our model classifies each object as either a turtle or a penguin. Figure 10 shows the classification results. Our model shows some accuracy in distinguishing between sea turtles and penguins, with most of the errors occurring in images where the animals are partially occluded or have abnormal poses.

Fig. 10. Classification Outputs

C. RTMDet Detector

The performance of real-time object detection model, RTMDet, is evaluated using two key metrics: Average Precision (AP) and Average Recall (AR), across varying Intersection over Union (IoU) thresholds and object sizes.

In the AP plot, the AP for small objects, which was unmeasurable, is now represented as zero. This further emphasizes the model's limitation in accurately detecting small objects. Despite these challenges with small objects, the model achieves a notable AP of 0.111 at an IoU of 0.50 for all objects, which decreases to 0.037 when the IoU threshold encompasses a range from 0.50 to 0.95. This highlights the model's struggle to maintain precision at higher IoUs.

The model achieves an AR of 0.074, 0.199, and 0.268 for maxDets of 1, 10, and 100 respectively, across all IoU thresholds from 0.50 to 0.95 for all objects. The increase in AR with more allowed detections suggests that the model can detect a larger number of objects, albeit at the cost of increased false positives.

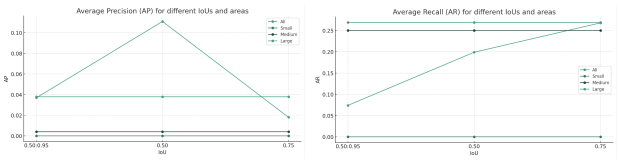


Fig. 11. Average Precision and Average Recall

Figure 3 depicts the learning rate schedule employed during the training phase. The learning rate was initially set to a high value and was gradually decreased over subsequent iterations. The sharp drop observed at the 60th and 120th epochs corresponds to a deliberate learning rate decay, a common practice in training deep learning models to help the model converge. This strategy can prevent the model from overshooting the minimum in the loss landscape. Analyzing the average precision across different epochs also showed interesting trends. Initially, the AP increased rapidly, indicating that the model was quickly learning to identify the target objects. After a certain number of epochs, however, the rate of improvement in AP slowed down, suggesting that the model was starting to converge. The average precision eventually reached a plateau, indicating that further training was not leading to significant improvements in the model's ability to detect objects.

Figure 12 showcases some representative results from our real-time object detection system. Each image includes bounding boxes, drawn by the model around the detected objects, with corresponding labels of 'turtle' or 'penguin'. The labels were assigned based on the classification output of the model.

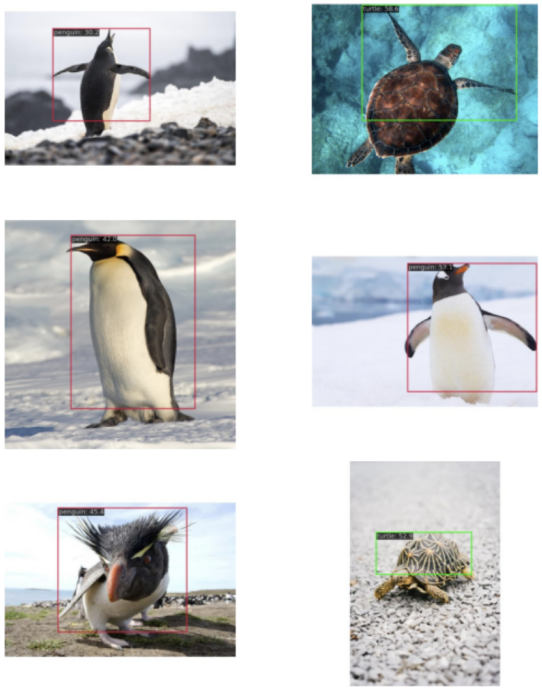


Fig. 12. Penguins and Turtles

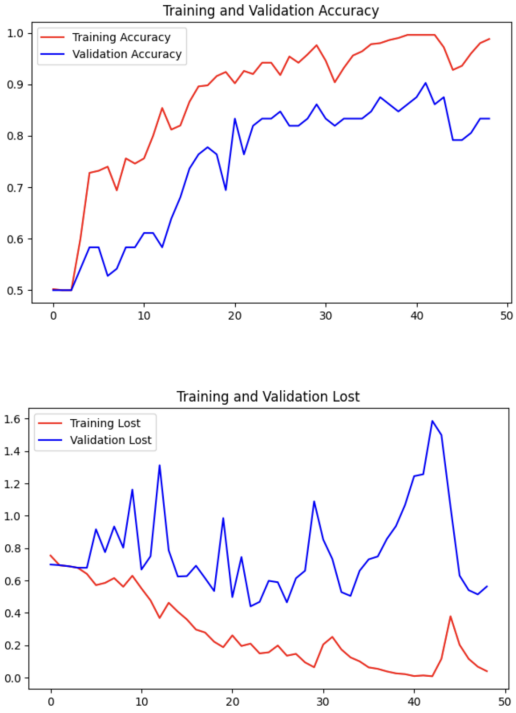
D. Faster-RCNN

As can be seen in Figure 13, the model has classified a variety of images. The images of turtles and penguins, despite their varying poses, environments, and lighting conditions, have been correctly recognized by our model.

After adjusting the parameters and the layers of the model, We obtained the accuracy of nearly 90.28%.

The CNN model employed for the classification of turtles and penguins has demonstrated remarkable performance with an accuracy of 90%. This achievement highlights the

Fig. 13. Turtles and Penguins



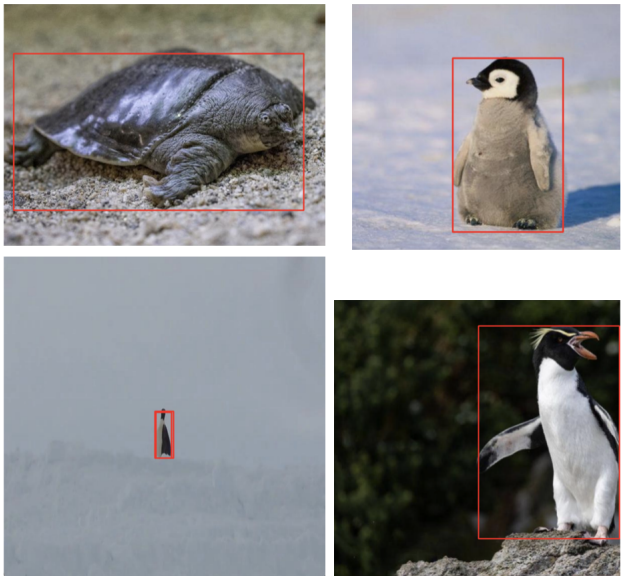


Fig. 14. Detection Outputs of Faster-RCNN

Fig. 15. Classification Outputs of CNN

effectiveness of deep learning techniques(CNN) in accurately discerning between penguins and turtles. Except for some extreme cases. The model's success holds significant potential for wildlife conservation and research, aiding experts in studying and preserving these animals' habitats and populations.

- Accuracy: 90.28%
- Precision for penguins: 91.8%
- Precision for turtles: 89.76%

V. Discussion

A. Traditional Machine Learning

Upon analysis, we think the binary clustering of k-means may be too simplistic, leading to the grouping of some similar samples together. On the other hand, random forest is composed of multiple decision trees, which can utilize the voting results of multiple trees to make more accurate classification predictions. However, even with random forest, the overall accuracy is still not satisfactory. Especially, we noticed that although the overall accuracy improved after using the random forest model, the accuracy of the traditional method for predicting turtles did not improve at all. Just by observing the images with the naked eye, we can also notice that many turtle images have a high resemblance to penguins in terms of appearance. Therefore, we decided to try some deep learning methods to further explore our problem.

B. Detection Transformer

we think there are some areas where the DETR model's in this project performance could be improved. Exploring

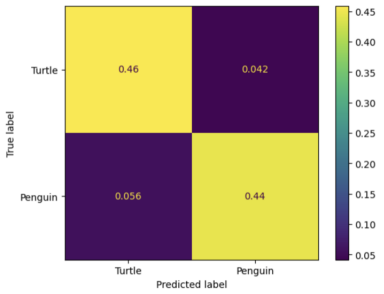


Fig. 16. Confusion Matrix

different CNN backbones and Transformer architectures to see if there is a better combination for this specific task. Augmenting the training data with various transformations (e.g., rotation, flipping, scaling) to increase data diversity and help the model generalize better to different variations. Adjusting the class weights during training to give more importance to the underrepresented class (turtles). Adjusting the model specifically for turtle detection.

C. RTMDet Detector

RTMDet approach is based on the real-time detector of the mmdetection framework, and a noteworthy issue that arose during the evaluation process was the performance of the method when dealing with certain types of objects in the image (especially small area objects). The experimental results showed relatively low detection precision and recall for these specific categories, and further research is necessary to elucidate the underlying reasons for this discrepancy.

A logical reason for the lower detection performance for small objects may be related to the inherent trade-off between detection speed and accuracy for single-stage detectors. These real-time detectors typically prioritise detection speed for real-time applications, which may inadvertently affect detection accuracy, particularly for small objects.

Another potential contributing factor could be tied to the backbone architecture used in the detection process. As aforementioned, the architecture employs CSP-blocks [3] as its backbone. Although this setup is designed to extract multi-scale feature pyramids efficiently, it might not be able to extract detailed features of small objects sufficiently, leading to a diminished detection capability.

Moreover, the neck architecture, which employs bottom-up and top-down feature propagation strategies, might also contribute to the subpar performance in detecting small area objects. Although the neck is designed to enhance the pyramid feature map, it might not be able to adequately amplify the weak features of small objects during the bottom-up and top-down processes.

According to the confusion matrix, the miss-classification rate of turtles is slightly higher then penguin, this is partly because penguins have a more recognizable colour pattern, makes it easier to be recognised. According to the confusion matrix of all these methods, the misclassification rate for turtles is slightly higher than that for penguins. One of the reasons is that the distinctive color patterns of penguins make them easier to recognize, leading to more accurate identification. The color similarity between this particular turtle and the black-and-white appearance of penguins in the training data. Most of the turtles in the training dataset have a white color In summary, the overall output of the model is highly accurate. This approach proves effective even when detecting animals that are partially occluded or located in cluttered backgrounds.

D. Faster-RCNN

we monitored the performance of model during training, the accuracy continuously growing higher; at the same time, the validation and training lost is decreasing with time, but after the 27th epoch, Lost curve started to grow rapidly, that's the time when we encounter the over-fitting problem. To solve this, We added an early stopped parameter, so as to achieve a better performance.The general output is very precise, because it uses Region Proposal Networks (RPN) to propose candidate regions of interest, and then a Region-based CNN (RCNN) for classifying.Even when animals that are partially occluded or in cluttered backgrounds.

VI. Conclusion

In this report, we have conducted an in-depth comparison of traditional learning methods, detection transformation approaches, real-time detector strategies, and the Fast R-CNN method in the context of object detection. Our findings indicate that the Fast R-CNN method tends to yield the highest accuracy among these techniques. However, it is crucial to note that this superior performance is achieved without taking into account the computational time and efficiency. As we advance further into the age of real-time applications, the efficiency of a model, its speed in processing and detecting, and its classification speed are becoming increasingly important. These factors are often as critical as accuracy, particularly in real-world scenarios where time and computational resources are limited. Therefore, while Fast R-CNN exhibits excellent accuracy, future studies and applications may place a higher emphasis on the efficiency of the model, striving for a balance between high accuracy and swift, efficient processing. This shift in focus will drive the evolution of object detection methods and broaden the horizon for new, more efficient algorithms and techniques.

REFERENCES

[1] Dalal, N., & Triggs, B. (2005) "Histograms of Oriented Gradients for Human Detection," In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego.

[2] MacQueen, J. B. (1967) "Some methods for classification and analysis of multivariate observations," In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, pp. 281-297.

[3] Breiman, Leo. (2001) "Random forests," Machine Learning, Vol. 45, No. 1, pp. 5-32, 2001

[4] Carion, N., Massa, F., Synnaeve, G. Usunier, N., Kirillov, A., & Zagoruyko, S. (2020) "End-to-end object detection with transformers," European conference on computer vision, pp. 213-229.

[5] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., ... & Lin, D. (2019). "MMDetection: Open mmlab detection toolbox and benchmark," arXiv preprint arXiv:1906.07155.

[6] Lyu, C., Zhang, W., Huang, H., Zhou, Y., Wang, Y., Liu, Y., ... & Chen, K. (2022). "Rtmdet: An empirical study of designing real-time object detectors," arXiv preprint arXiv:2212.07784.

[7] Saad AL-AZAWI, Tareq Abed Mohammed. (2018) "Understanding of a convolutional neural network" IEEE.

[8] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh- Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. (2019) "Cspnet: A new backbone that can enhance learning capability of cnn," CVPR.