Object Classification and Detection with Mobilenet and YOLO

Ayush Rai
School of CSE and Engg, UNSW
z5426583

George Luo
School of CSE and Engg, UNSW
z5317638

Tiffany Wing On Leung
School of CSE and Engg, UNSW
z5391022

Luke Wang
School of CSE and Engg, UNSW
z5440342

Leman Kirme
School of CSE and Engg, UNSW
z5410109

***Abstract*—In this report, we document some approaches taken by our team for object detection and classification on real-world images of Turtles and Penguins.**

I. Introduction

Object recognition and classification in real-world photos or movies are two significant and challenging computer vision tasks with range of applications like surveillance, traffic monitoring, robotics, diagnostic imaging, and biology. Full automation is required in many applications because the amount and complexity of the data make it impossible for humans to undertake precise, thorough, effective, and repeatable recognition and analysis of the pertinent image information. This group project aims to create and assess approaches for the identification and categorization of animals in wildlife photos. We will concentrate on penguins and turtles in particular for this project. The difficulty lies in creating techniques that can accurately and quickly analyze the pictures.

II. Literature Review

A. Object Detection using YOLO:

YOLO (You Only Look Once) is a pioneering object detection algorithm introduced by Redmon et al. [4] in 2016. It adopts a single-stage approach, treating object detection as a regression problem, and predicts bounding boxes and class probabilities directly from the raw image, enabling real-time performance. YOLO's strength lies in its efficiency and accuracy, processing the entire image in one pass through a deep CNN for impressive real-time capabilities. Over time, variants like YOLOv2 and YOLOv3 have refined and extended the algorithm, addressing limitations and improving performance, making it widely adopted in applications such as autonomous vehicles, surveillance systems, and robotics.

B. Object Classification using VGG-16:

VGG-16 [3], introduced by the Visual Geometry Group (VGG) at the University of Oxford in 2014, is a pioneering deep convolutional neural network for image classification. It consists of 16 weight layers and is characterized by using 3x3 convolutional filters throughout the network, resulting in a deeper architecture with fewer parameters. VGG-16 achieved exceptional performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, demonstrating its ability to recognize objects in 1000 categories with high accuracy. It has since become a foundational model for object classification and transfer learning, serving as the base for subsequent CNN models and contributing to the development of deeper networks.

C. Object Classification using Mobilenet:

MobileNet's primary goal is to provide state-of-the-art accuracy in vision tasks while minimizing computational resources for deployment on devices with limited processing power and memory. It achieves this by using depth-wise separable convolutions, splitting spatial and cross-channel processing to reduce parameters and computations. MobileNetV1 uses width multipliers for a trade-off between accuracy and efficiency. MobileNetV2 improves on V1 with an Inverted Residual block and a resolution multiplier. Its impact in computer vision is profound, enabling real-time performance on resource-constrained devices and finding applications in image classification, object detection, and more. [5] [6]

D. Attention Layers- CBAM:

CBAM proposes attention mechanisms to enhance CNNs' representation power. It sequentially applies channel and spatial attention modules to emphasize meaningful features and suppress irrelevant ones. CBAM's adaptability and lightweight design allow easy integration into existing CNN architectures. It improves accuracy significantly on ImageNet-1K and demonstrates wide applicability in object detection on MS COCO and VOC 2007 datasets. Visualization confirms its effectiveness in accurate attention and noise reduction, resulting in performance boos [7].

The impact of CBAM is further demonstrated through visualization using the grad-CAM technique [8]. The enhanced networks with CBAM were found to focus more accurately on target objects, leading to the conjecture that the performance boost arises from accurate attention and noise reduction of irrelevant clutters.

III. Methods

In this section, we present the methodology in the project which involves image pre-processing, object detection, and object classification. For object classification, we implement MobileNetV2 and explore different attention mechanisms for fine-tuning. For object detection, we employ YOLOv8 and investigate various loss functions and YOLO sizes.

A. Pre-processing:

The pre-processing of image data consists of 3 parts, namely image normalisation, image denoising and image augmentation. [9] The first part resizes all images with shape (224,224,3) as the models require consistent input image sizes.

Image Denoising:After conducting data exploration, we observed that a significant number of images in our dataset contain noisy backgrounds. To address this issue, we devised a denoising autoencoder employing Convolutional Neural Networks (CNNs) to remove unwanted noise while retaining crucial image features. [10] The denoising autoencoder is trained on noisy input data, learning to reconstruct the original, clean data as the output. The architecture of the autoencoder is depicted in Figure 1.

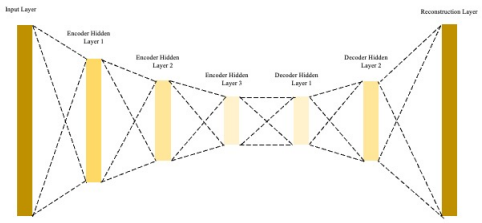

Fig. 1: Architecture of the autoencoder

In the encoder section, we employ three Conv2D layers with 32, 64, and 128 filters, respectively. Each layer utilizes a 3x3 kernel, ReLU activation function, and 'same' padding to perform spatial dimension reduction and channel expansion, resulting in a compressed representation.

In the decoder section, we incorporate two Conv2DTranspose layers with 64 and 32 filters, employing a 3x3 kernel, ReLU activation, and 'same' padding. These layers upscale the compressed representation, restoring it to the original image dimensions. The final Conv2DTranspose layer, equipped with 3 filters and a Sigmoid activation function, is responsible for the reconstruction process.

Figure 2 provides a summarized overview of the autoencoder's architecture.

Layer (type)	Output Shape	Param #
input_7 (InputLayer)	[(None, 224, 224, 3)]	0
conv2d_18 (Conv2D)	(None, 224, 224, 32)	896
conv2d_19 (Conv2D)	(None, 224, 224, 64)	18496
conv2d_20 (Conv2D)	(None, 224, 224, 128)	73856
conv2d_transpose_18 (Conv2D Transpose)	(None, 224, 224, 64)	73792
conv2d_transpose_19 (Conv2D Transpose)	(None, 224, 224, 32)	18464
conv2d_transpose_20 (Conv2D Transpose)	(None, 224, 224, 3)	867

Total params: 186,371
Trainable params: 186,371
Non-trainable params: 0

Fig. 2: Summary of the autoencoder

The autoencoder learns to capture meaningful features and patterns after 10 epochs to effectively remove noise and reconstruct cleaner versions of the input images. Figure 3 are some examples comparing original and denoised images:

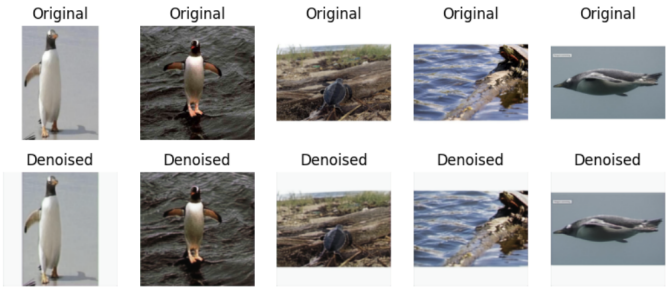

Fig. 3: Examples comparing original and denoised images

Image Augmentation:After image denoising, random image augmentation is implemented to diversify the dataset and improve model generalization. A 2.5K augmented images dataset was generated where it does not give satisfactory training result. A 5K augmented images dataset was generated and used for object detection and classification. Techniques including Rotation, Translation, Scaling, Flipping, Color Temperature Adjustment and Hue multiplication are implemented. Specifically, the Color Temperature Adjustment range is confined to 4000 to 10000 degrees Kelvin (K) to align closely with outdoor color temperature variations. Additionally, bounding boxes (bbox) are recalculated accordingly to ensure consistency. A few examples of the augmented images are depicted in Figure 4.

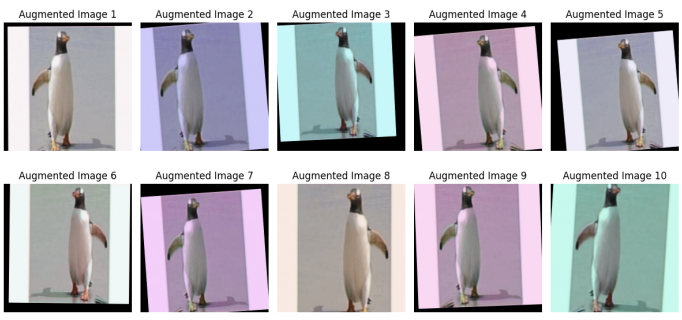

Fig. 4: Augmented images

Figure 5 are some examples shows the augmented images with the corresponding bbox :

Fig. 5: Augmented images with the corresponding bbox

Additionally, we incorporated background padding by extending the area outside the bounding box to white before feeding it to the training model. The primary goal was to mitigate the distraction caused by the cluttered background in the original images. Despite this endeavour, the experimental results showed no discernible impact on the overall outcome. Thus, the background padding dataset was omitted from our approach.

B. Detection:

How does YOLO work:

The YOLO Detection System (see Fig. 6.). [4] Processing images with YOLO is simple and straightforward.

1) resizes the input image to 448 × 448
2) runs a single convolutional network on the image
3) thresholds the resulting detections by the model's confidence.

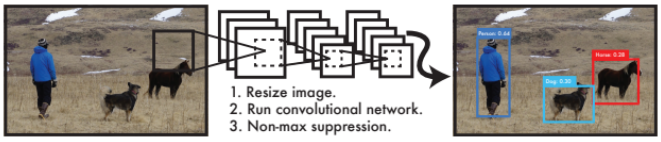

Fig. 6: The YOLO Detection System

The system models detection as a regression problem. [4] It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B × 5 + C) tensor. See Figure 7.

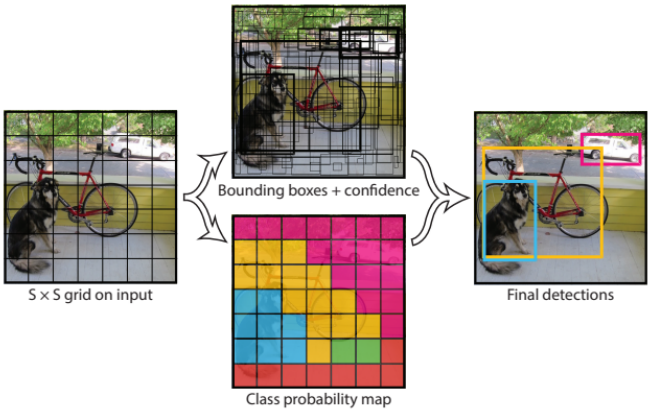

Fig. 7: The Model

The architecture (see Fig. 8) works [4] as follows:

- Resizes the input image into 448x448 before going through the convolutional network.
- A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
- The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

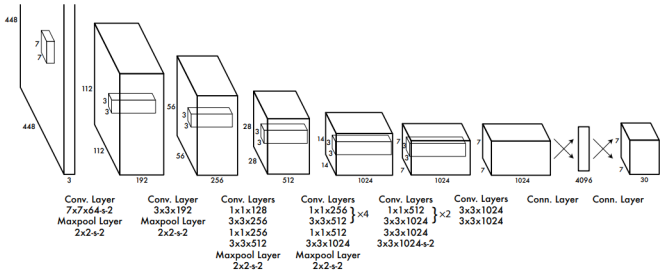

Fig. 8: The Architecture

YOLO Custom Dice Loss Function Model:

YOLOv8 uses the CIoU loss function [4].

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v$$

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2$$

$$\alpha = \frac{v}{(1 - IoU) + v}$$

The CIoU loss introduces two new concepts compared to IoU loss. The first concept is the concept of central point distance, which is the distance between the actual bounding box center point and the predicted bounding box center point. The second concept is the aspect ratio, we compare the aspect ratio of the true bounding box and the aspect ratio of the predicted bounding box. CIoU loss simultaneously considers three metrics: overlapping area, the distance between center points, and the aspect ratio. With these 3 measures, we can measure the quality of the predicted bounding box. It is an extension to DIoU loss. where b and bgt denote the central points of B and Bgt, ρ is the Euclidean distance, c is the diagonal length of the smallest enclosing box covering the two boxes, α is a positive trade-off parameter, and v measures the consistency of aspect ratio. The α is a positive trade off parameter where overlaps are given higher priority over non-overlap cases and v gives information about consistency of aspect ratio.

In our custom model, we change the loss function with the Dice Loss function. [13]

$$DL(y, y') = 1 - \frac{2 \cdot y \cdot y' + 1}{y + y' + 1}$$

Dice Loss = 1 - (2 * Intersection + Smooth) / (Sum of Predicted + Sum of Target + Smooth). The intersection represents the sum of element-wise multiplication between the predicted output pred and the target tensor target. Smooth is a small constant added to avoid division by zero and improve numerical stability. The denominator consists of the sum of all elements in the predicted output and the sum of all elements in the target tensor, both added to the smooth constant.

CIoU vs Dice Pros and Cons:

CIoU loss is an extension of IoU (Intersection over Union) loss, which is often used for bounding box regression in object detection tasks. It extends IoU by considering the overlap between the predicted bounding box and the ground truth, as well as the aspect ratio and the central point distance between two boxes. By doing so, it not only penalizes misaligned boxes but also those with incorrect aspect ratios.

Pros:

- It incorporates more information (aspect ratio, central point distance) into the loss function, potentially leading to better localization and scale predictions.
- CIoU loss can be more robust to different object shapes and sizes, as it considers the aspect ratio.

Cons:

- The computation for CIoU loss can be slightly more complex than some other loss functions because of the additional factors it considers.

Dice loss, also known as the Sørensen–Dice coefficient or Dice similarity coefficient (DSC), is a statistical validation metric used to quantify the similarity between two sets. In the context of bounding box regression, the Dice loss would be 1 minus the Dice coefficient, meaning the loss is lower when the predicted bounding box and ground truth overlap more. The dice coefficient is the same as the F1 score, giving the harmonic mean of precision and recall. It balances the trade-off between precision and recall.

Pros:

- Dice loss is particularly useful for dealing with uneven class distribution, as it maintains a good balance between precision and recall.
- It prioritizes precision and recall
- It provides a differentiable and smooth measure of segmentation accuracy
- It is straightforward to compute and does not require any complicated operations, which might lead to faster computation.

Cons:

- Unlike CIoU loss, Dice loss does not account for aspect ratios or distances between the central points of boxes. Therefore, it may be less effective for tasks where these factors are significant.
- It is less intuitive than
- The Dice loss can be sensitive to the location of the bounding boxes. If the boxes do not overlap, the Dice loss will become maximal, even if the boxes are very close.

After trade-off the Pros and Cons, we decided to apply Dice Loss for our custom model to give it a try.

YOLO Data Format Procedure:

The source data provided for the Penguins vs Turtles task was annotated in Pascal VOC format, where the bounding box coordinates are given in the following format: [xmin, ymin, xmax, ymax]. However, YOLOv8 requires bounding box coordinates to be normalized between 0 and 1 and represented as [xcenter, ycenter, boxwidth, boxheight]. Consequently, we had to convert the original annotations to the required format.

For instance, an initial annotation might look like this: ”category_id”: 1, ”bbox”: [119, 25, 205, 606].

In order to adapt this for YOLOv8, it needed to be converted into this format:

1 0.37277 0.54018 0.32589 0.95536

This conversion was vital for ensuring compatibility with YOLOv8 and facilitating effective model training.

YOLO Experiment Procedure:

In our pursuit of optimal model performance, we conducted extensive training experiments across several models, each evaluated carefully.

The models we trained included:

- yolov8n (Nano-Model)
- yolov8s (Small-Model)
- yolov8m (Medium-Model)
- customYolov8n (Custom-Model)

Each model was chosen with the objective of covering a broad spectrum of model sizes, from nano to medium, ensuring robustness in our comparative analysis. Moreover, we tested customized models with a different loss function (Dice loss) to evaluate the impact of the loss function on the model's performance.

To further this analysis, the models were trained on two different datasets:

- Original (500 Images)
- 5k-Augmented (5500 Images)

We used a smaller dataset of original images to understand the baseline performance of the models, while an augmented dataset was used to test how effectively the models could learn and generalize from a larger, more diverse dataset.

For validation, we used the given valid dataset:

- Valid (72 Images)

The validation dataset was kept separate and distinct to assess the model's ability to make accurate predictions on unseen data.

Regarding the hyperparameter choices, we made the following decisions:

- All models were trained for 50 epochs. We believe this provides a balance between computational efficiency and the opportunity for the models to learn the complexities of the data.
- We standardized the batch size to 16 across all models. This consistency ensures an even comparison and prevents the batch size from being a confounding variable.
- We chose an image size of 224 for training. The original image size is 640, but we opted for a smaller size after performing image denoising and augmentation. This was a trade-off made to balance model performance and training time, acknowledging that smaller images can speed up training with potential marginal performance sacrifice.

This comprehensive experimental setup was designed to provide a thorough understanding of the model performance under different conditions and configurations, ultimately leading to a more informed decision about the best model to use.

C. Classification:

MobileNetv2:We initially trialled VGG16 as a base model CNN for this task [3] however, on further testing we ran into an overfitting problem which we could not resolve despite applying different strategies such as data augmentation, dropout and weight regularisation. Ultimately, we deemed that the VGG16 architecture was too complex for this simple task and we opted for something more lightweight, arriving at MobileNetv2 which both performed with higher accuracy at a fraction of the training time.

MobileNetv2 is a popular lightweight architecture introduced in 2018 by Sandler et al. [6]. This model utilises depth wise separable convolutions and inverted residual blocks to create a model that is accurate despite a low computational and memory cost which we felt was perfect for our task.

The essence of MobileNets lie in the depthwise separable convolutions it uses instead of traditional convolutional layers. A depthwise separable convolution is split into 2 parts:

- Depthwise convolutions
- Pointwise convolutions

Depthwise convolutions apply a kernel on each channel of the input separately with a ReLU, resulting in M feature maps, a feature map for each channel. These feature maps are then pushed through a pointwise convolution where a set of 1x1xM kernels are applied to combine the outputs of the depthwise convolutions. This also utilises a ReLU. Depthwise separable convolutions reduce the number of parameters and computational cost significantly compared to traditional convolutions while still maintaining necessary information.

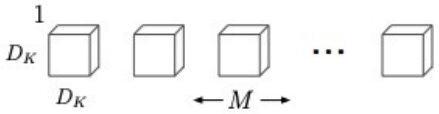

Fig. 9: Depthwise convolution image [5]

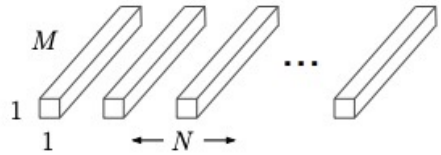

Fig. 10: Pointwise convolution image [5]

MobileNetv2 also introduces inverted residual blocks. The inverted residual block begins with a 1x1 pointwise convolution with a linear activation to reduce the number of channels and create a bottleneck. This helps reduce the computational cost. The depthwise separable convolutions are then applied and the feature maps a finally pushed through an expansion convolution which restores the number of channels to the same as the input. This also ensures that the shape of the output matches the required input shape of the next layer.

In our model, we remove the fully connected layers of the base model and instead add 2 dense layers, ending with a sigmoid activation to give an output between 0 and 1. If the output is closer to 0, the model predicts a penguin and if the output is closer to 1, it predicts a turtle.

Overall, MobileNetv2 provides an efficient trade-off by greatly reducing the computational power required whilst still maintaining most of the expressive power of other CNNs. In our application, this works well for us because the reduction of complexity in the feature maps actually helps us minimize overfitting as there is not a lot of variation in the dataset, resulting in higher accuracy than more complicated CNNs.

Self attention mechanism:-

For our first attention module, we trialled self-attention in combination with MobileNetv2. Self-attention is a mechanism that is more frequently used in transformer based networks however, we see in [Pan et al., 2021] that it also produces desirable results in traditional CNNs.

As there were no modules available to implement self attention to a traditional CNN, we decided to use Luong attention provided by Keras to create a pseudo self attention layer in our network. Luong attention is used is seq2seq models to show information about relationships between words by dot multiplying key, query and value vectors together and pushing the output through a softmax. In our model, we trialled using the feature maps outputted by MobileNetv2 as both the key and query values in the Luong mechanism. This effectively

dot multiplies feature maps together to provide information about relationships between similar features to construct a more global view of the image to help with classification.

CBAM:

The remaining attention based models are based on the paper by Woo et al. [7]. The proposed CBAM model combines channel and spatial attention together to provide more information to allow for better classification. For this task, we decided to break down the mechanism and try channel and spatial attention layers separately, before combining them into a similar model as the paper to show which mechanism is more effective in this task.

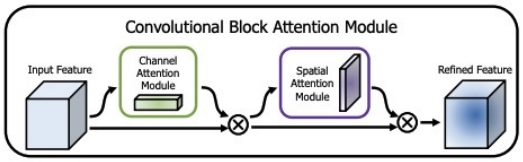

Fig. 11: CBAM model [5]

Channel attention:

Our channel attention model takes the feature maps of MobileNetv2 and applies 2 pooling operations to each feature map. This will create 2 1 dimensional vectors with the pool value for each feature map. These 2 vectors are then added together and pushed through a softmax activation to provide weight values for each feature map. These weight values are multiplied with the feature maps to provide higher weights to more important features and lower weights to less important features.

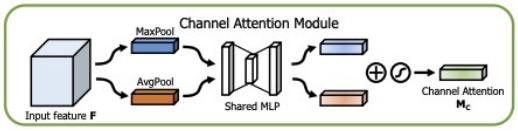

Fig. 12: Channel attention mechanism image [5]

Spatial attention:

This works similarly to channel attention in that it produces a feature map with information gained from both max pooling and average pooling however, here the pooling is applied across the channel dimension. The output of the feature maps is pushed through another convolutional layer to reduce the number of channels back to its original value and similarly to channel attention, it is pushed through a sigmoid which produces a map of weights that is multiplied on the original feature maps. This allow the model to provide higher weights to more important spatial areas on the image.

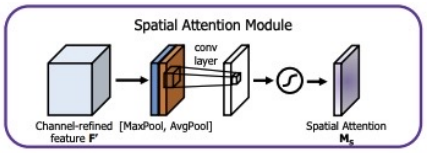

Fig. 13: Spatial attention mechanism image [5]

Final Model:

Our final model consisted of MobileNetv2 as a base model with an added CBAM module before the fully connected layers. In the CBAM layer, both the channel and spatial attention weights are multiplied with the original feature maps to capture both inter-channel and inter-spatial dependencies to allow for richer and more expressive feature maps that focus on important information to aid in classification. Our model utilised the pre-trained weights from imagenet to reduce computational cost and improve generalisation. We used a binary cross entropy loss function as we felt this was the most appropriate for a binary task and to further reduce overfitting, we added a dropout rate of 30% and an early stoppage mechanism. We ran the model for 15 epochs. We chose a higher number as the early stoppage mechanism would stop the training at the optimal stage regardless of how many epochs are set. Note, we ran a smaller amount of epochs in the demo video to speed up the process and allow for easier recording. We trained our model on the augmented dataset and tested it on the validation set provided which consisted of 72 images.

We used the same set of evaluation metrics on all our classification models. These included accuracy, precision, recall and F1 score. For our final model, we also included a visualisation function where an input image is pushed through the model and outputted with a caption of what it predicts.

IV. Experimental Results

A. Detection:

IoU and Center Distance Performance:

As our project specification required, we conducted a performance evaluation of the different models using two key metrics: Intersection over Union (IoU) and the distance between the predicted bounding box center and the true bounding box center. These measurements were calculated for each validation image and we reported both the mean and standard deviation across all validation images for each model.

The results are divided into two categories based on the type of training data: 'Original Data' and '5k-Augmented Data'.

TABLE I: Models trained with Original Data performance results

Model	Mean IoU	Std IoU	Mean Center	Std Center
yolov8n_orig	0.90527	0.15443	12.07699	25.68534
yolov8s_orig	0.90078	0.16623	12.87885	28.90730
yolov8m_orig	0.89265	0.17038	12.88726	26.73082
custom_orig	0.73844	0.19549	29.60602	34.10672

TABLE II: Models trained with 5k-Augmented Data performance results

Model	Mean IoU	Std IoU	Mean Center	Std Center
yolov8n_5k_aug	0.88433	0.08298	10.82376	10.40678
yolov8s_5k_aug	0.86914	0.11846	13.55353	17.84783
yolov8m_5k_aug	0.87482	0.11699	13.44298	19.81542
custom_5k_aug	0.80930	0.17894	20.54384	27.61810

Based on these results, we can list out the comparison for model object detection performance with different evaluation aspects:

- Mean IoU: the model (yolov8n_orig) trained on the YOLOv8 Nano model with the Original dataset has the best performance.
- Std IoU: the model (yolov8n_5k_aug) trained on the YOLOv8 Nano model with the 5k Augmented dataset has the best performance.
- Mean Center Distance: the model (yolov8n_5k_aug) trained on the YOLOv8 Nano model with the 5k Augmented dataset has the best performance.
- Std Center Distance: the model (yolov8n_5k_aug) trained on the YOLOv8 Nano model with the 5k Augmented dataset has the best performance.

After these evaluation comparisons, we can ascertain that the model (yolov8n_5k_aug) trained on the YOLOv8 Nano size pre-trained model with the 5k augmented dataset exhibited the highest overall object detection performance among the tested models.

YOLO mAP Comparison of each Epoch:

The mean average precision comparison results are shown in Fig. 14. and Fig. 15.

After these mAP comparisons, we can see the mAP in those models trained with the 5k Augmented dataset is growing faster than those models trained with the Original dataset.

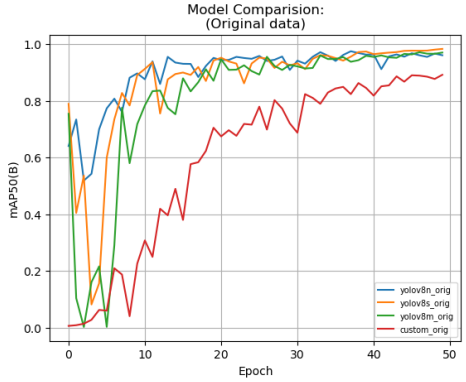

Fig. 14: Model mAP comparison with Original Data

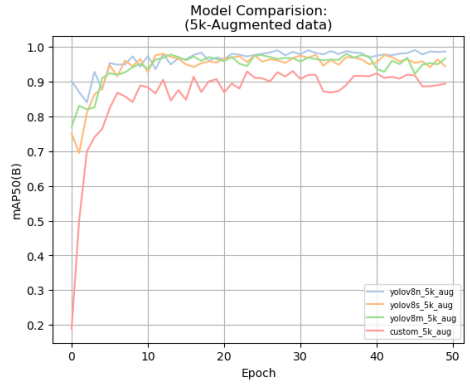

Fig. 15: Model mAP comparison with 5k Augmented Data

YOLO Object Detection Visualisation:

As shown in the figure (see Fig. 16) , we visual the best model (yolov8n_5k_aug) predicted bounding box on validation images.

The ground truth bounding boxes are in green rectangles and the model-predicted bounding boxes are in red rectangles.

Fig. 16: yolov8n_5k_aug visualisation

B. Classification:

To evaluate, we looked at accuracy, precision, recall, F1 score, validation loss and the confusion matrix. We considered validation loss which was not specified in the assignment specifications as the base model was already very high performing only predicting 2 incorrectly and the validation dataset provided was quite small which did not give much differentiation between results with other metrics.

Not included in our notebook, we also trialled VGG16 as a base model however, despite it achieving extremely high training accuracy, it tended to overfit and consistently achieved 85%-90% validation accuracy. Here, we can observe that the switch to MobileNetv2 was a great choice as it already performs very highly with just the base model, only guessing 2 incorrectly. out evaluation metrics for MobileNetv2 and all the subsequent attention mechanisms added are displayed in the table below:

TABLE III: Evaluation of classification models

Model	Accuracy	Precision	Recall	F1 score	Validation loss
MobileNetv2	0.972	0.947	1	0.972	0.269
Added self-attention	0.972	0.972	0.972	0.972	0.167
Added channel attention	0.972	0.972	0.972	0.972	0.137
Added saptial attention	0.972	0.972	0.972	0.972	0.187
CBAM	0.986	0.973	1	0.986	0.262

We can observe that excluding the CBAM model, all the attention models achieved the same accuracy as the base

model however, they all consistently achieved lower validation accuracy. We predict with these findings that given a larger validation set, the attention models would all outperform the base model in all the other metrics.

The CBAM model did however, have the best performance where it only predicted 1 image incorrectly. Most likely an edge case in the validation dataset of an image with very similar features as the opposing class. We did observe however that the validation loss for the CBAM model despite being lower than the base model, was higher than the other attention models. This is because of our implementation of early stopping where we found that CBAM training converge quicker than the other models.

Following training, we included a function that will take an image, predict it's class and caption is appropriately to help visualise the results. An example can be seen in the figure below:

Fig. 17: Captioned image

In addition, we also included a GRAD CAM [8] heatmap function to help visualise how the CBAM model applies more weight to certain areas of the image. The heatmap is produced by computing the gradient between the final convolutional layer and the output. It is then overlaid onto the original image. Despite the high accuracy of the model, we can observe on the heat maps that there were some inaccuracies on the attention mechanisms for example in image 1 in Fig. 18.

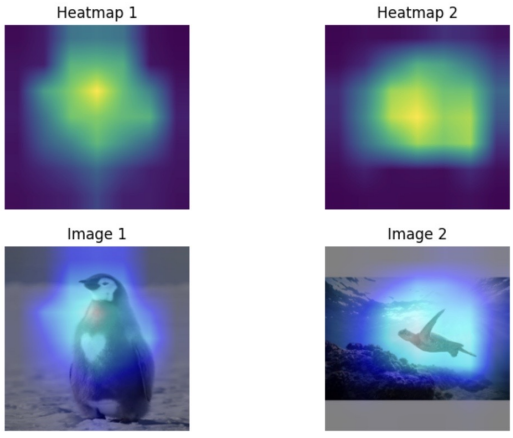

Fig. 18: GRAD CAM [8] heatmap visualisation

V. Discussion

The results of our experiments demonstrated promising performance in both object detection and classification tasks.

In the object detection task, the yolov8n model trained on the 5k augmented dataset achieved the highest mean Intersection over Union (IoU) value, indicating superior accuracy in localizing the objects. This model's success can be attributed to the diverse augmented dataset, which allowed the model to learn from a wide range of variations in the images.

In the object classification task, MobileNetv2 served as an efficient and lightweight base model, achieving high accuracy in predicting the classes of penguins and turtles with very quick training speed. The addition of the CBAM attention mechanism further enhanced the model's classification capabilities, resulting in the best performance with only one misclassification. Despite their effectiveness in our model, the attention mechanisms proved more effective on a base of a more complex CNN like VGG16 as it gave more accurate feature maps compared to MobileNetv2. However, for this task specifically, a more complex base model led to overfitting issues due to lack of size and variability in the dataset.

Additionally, for both tasks but especially classification, as the validation set provided was quite small at 72 images, it was difficult to differentiate clearly between different models. This was because the MobileNetv2 base model already achieved high accuracy with only 2 incorrect predictions.

VI. Conclusion

In conclusion, our work in the project yielded encouraging results for both object detection and classification tasks. The yolov8n model trained on the augmented dataset performed excellently in object detection, while the MobileNetv2 model with the CBAM attention mechanism excelled in object classification.

Although the YOLO model was high performing in general, we found that the custom Dice loss function we implemented was not as effective as the original model. This is most likely because the Dice loss function doesn't consider as many aspects as the original CIoU loss however, we predict it would be more effective for segmentation tasks. The attention mechanisms proved to be effective in improving classification performance, however, given a more diverse dataset, we believe we could improve this performance even further building it on the backbone of a more complex CNN.

For future work, we recommend the following:

- Increase the size of the validation dataset to obtain a more comprehensive evaluation of the models' generalization capabilities.
- Trial a YOLO model with a Dice loss function for a segmentation task.
- Trial attaching the CBAM mechanism onto a more complex CNN that give richer feature maps to provide more

accurate attention information and train on a larger and more diverse dataset to avoid overfitting.

- Investigate other attention mechanisms and architectures to further improve classification accuracy and feature representation.
- Explore joint detection and classification models to leverage the strengths of both tasks and potentially improve overall performance.

Overall, the project provided valuable insights into object detection and classification tasks, paving the way for potential applications in various real-world scenarios. With further exploration and refinement, the models can be made more robust and applicable

REFERENCES

[1] Z.-H. Zhou and J. Feng, "Deep Forest: Towards an alternative to deep neural networks," Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017. doi:10.24963/ijcai.2017/497

[2] H. Sahbi, "Deep total variation support Vector Networks," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019. doi:10.1109/iccvw.2019.00365

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014. doi: 10.48550/arXiv.1409.1556

[4] R. Joseph, D. Santosh, G. Ross, and F. Ali, You Only Look Once: Unified, Real-Time Object Detection, Jun. 2015. doi:10.48550/arXiv.1506.02640

[5] A. G. Howard, M. Zhu, B. Chen, and D. Kalenichenko, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Apr. 2017. doi: 10.48550/arXiv.1704.04861

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018 pp. 4510-4520. doi: 10.1109/CVPR.2018.00474

[7] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," Computer Vision – ECCV 2018, pp. 3–19, 2018. doi:10.1007/978-3-030-01234-21

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74.

[9] S.-T. Bow, Pattern Recognition and Image Preprocessing. New York: M. Dekker, 2002.

[10] X. Bian, "Denoising autoencoder on colored images using tensorflow," Medium, https://medium.com/analytics-vidhya/denoising-autoencoder-on-colored-images-using-tensorflow-17bf63e19dad

[11] X. Pan et al., "On the integration of self-attention and convolution," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. doi:10.1109/cvpr52688.2022.00089

[12] P. Chandra, "IOU loss functions for faster amp; more accurate object detection," LearnOpenCV, https://learnopencv.com/iou-loss-functions-object-detection/

[13] D. Espressius, "3 common loss functions for image segmentation," DEV Community, https://dev.to/aadidev/3-common-loss-functions-for-image-segmentation-545o