COMP9517 Group Project Report

Penguins vs Turtles classification

By no fail Group

Hongyu Chen
z5097965

Ruofan Zhou
z5370291

Aiwen Zhang
z5398780

Shiwudi Dong
z5427640

***Abstract*—This study explores the application of CNN and YOLOv8 models in turtle and penguin image classification. YOLOv8 exhibits high accuracy and fast processing, making it suitable for real-time object detection. CNN demonstrates broad applicability.**

I. Introduction

A. Background and Motivation

Image classification that distinguishes between similar species is critical for wildlife monitoring, species conservation, and other related activities. Many species are on the verge of extinction due to human activities such as poaching, natural environment pollution, and habitat destruction. Enforcing anti-poaching measures and monitoring illegal trade or activities related to these animals is easier when these species can be correctly identified and differentiated in pictures. Because of this, there is a huge need for species detection that can help researchers in making speedy and precise identifications.

B. Purpose of the objects

Our group's goal in this research is to create an image categorization system for turtles and penguins. Using the same datasets, we applied existing popular deep learning object detection models such as Yolov8 and basic CNN. Our team will compare the performance of different models, as well as the analytical results and limitations.

As a result, our team will provide an appropriate model for image classification systems that distinguishes between turtles and penguins quickly and accurately to support research organizations.

II. Dataset

id	integer, id
Image_id	integer, image id
category_id	1 for penguin, 2 for turtle
bbox	list of integers representing the bounding box coordinates in Pascal VOC format [xmin, ymin, xmax, ymax]
area	integer representing area of bounding box.
segmentation	empty list; add segmentation masks if you'd like!
iscrowd	integer 0 or 1; whether the instance is a crowd or individual. Not relevant to this particular use case, but is a necessary key for some models.

There are 572 images where the training dataset is splitted into 500 images and the testing set is separated as the remaining 72 images. We have the train and test folder contains the images, train annotations document and test annotation document.

III. Literature Review.

A. Introduction

Image classification is a fundamental task in the field of computer vision and plays a crucial role in various practical applications. Accurate classification of different objects' images is essential in wildlife conservation, surveillance, and environmental monitoring. In this literature review, we will explore two popular deep learning models, Convolutional Neural Network (CNN) and YOLOv8, for the classification of turtle and penguin images.

B. Literature Review of Sliding window detection

In the realm of image processing, the sliding window technique is a staple methodology primarily utilized in tasks such as object detection, feature extraction, and pattern recognition. The technique involves systematically moving a sub-window or 'patch' across an image to analyze local regions, typically with the aim of identifying specific features or objects.

The operation of convolution in CNNs, fundamental to deep learning-based image processing, is inherently a form of sliding window where filters (or kernels) move over the input image to extract features.[1] This step can help to preprocess training data or images, giving a more accurate model for prediction.

C. Literature Review of CNN

Convolutional Neural Networks have had a revolutionary impact on image classification since their introduction. In 1998, LeCun et al. proposed LeNet, one of the pioneering CNN architectures. Subsequently, the development of CNN models such as AlexNet, VGG, ResNet, and Inception has made significant progress in terms of accuracy and generalization performance. These models have achieved state-of-the-art results on large-scale image classification datasets such as ImageNet.[2]

In the context of turtle and penguin image classification, researchers have explored various CNN architectures. However, the accuracy of CNN models may be affected by factors such as dataset size, quality, and class imbalance.

D. Literature Review of YOLOv8

The YOLO (You Only Look Once) series of models are known for their real-time object detection capabilities. YOLOv8 is the latest version, with many improvements over its predecessors. One key feature is multi-scale feature fusion, enabling it to efficiently capture features of objects of different sizes.[3][4]

In the realm of image classification, researchers have also explored the potential of YOLOv8 for detecting and classifying objects like turtles and penguins. With its real-time performance and multi-scale feature fusion advantages, YOLOv8 has shown promising results in detecting and classifying various objects, including wildlife species. Additionally, YOLOv8's end-to-end optimization makes it attractive for real-time applications.

Comparative Analysis: Comparing the performance of CNN and YOLOv8 in turtle and penguin image classification, we find that each model has its strengths and weaknesses. CNN excels in capturing complex features and has wide applicability in image classification tasks. However, its performance may be influenced by factors such as dataset quality, size, and class imbalance, making real-time image processing challenging. On the other hand, YOLOv8's real-time performance, multi-scale feature fusion, and simplified design make it perform exceptionally well in fast and efficient object detection and classification scenarios, particularly suited for real-time wildlife monitoring.[5]

E. Conclusion

In conclusion, both CNN and YOLOv8 show potential in turtle and penguin image classification. CNN's powerful feature learning ability and broad applicability make it a reliable choice. Meanwhile, YOLOv8's real-time performance, multi-scale feature fusion, and simplified design make it an excellent option for fast and efficient object detection and classification. Choosing the appropriate model should depend on specific requirements, such as desired accuracy levels and real-time processing. As technology advances, further research and improvements in both CNN and YOLOv8 will enhance their capabilities in image classification tasks and strengthen their applications in wildlife conservation and environmental monitoring domains.

IV. METHODS AND RESULTS

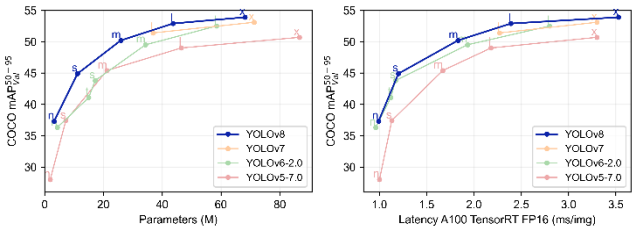
A. YOLOv8

1) Introduction

The most recent and cutting-edge YOLO model, YOLOv8, can be utilized for applications including object identification, image classification, and instance segmentation.

The yolov8 provides a brand-new SOTA model with a YOLACT-based instance segmentation model and object detection networks with P5 640 and P6 1280 resolutions. According to YOLOv5, diverse N/S/M/L/X scale size models can be obtained based on the scaling factor to satisfy the requirements of various scenarios.

The yolov8 also provides a new backbone network, a new Ancher-Free detection head, and a new loss function that can run on several hardware platforms, from CPUs to GPUs, are among the specific advances. The backbone network and neck portion may relate to the YOLOv7 ELAN design thought, which replaces the YOLOv5 C3 structure with a C2f structure with a richer gradient flow and modifies various channel numbers for various scale models. The Head component features two significant upgrades over Yolov5, first, the classification and detection heads are separated by the decoupled head construction, which is the current standard. Second, Anchor-Based is replaced with Anchor-Free. For the loss aspect, YOLOv8 employed the Task-Aligned Assigner positive and negative sample matching approach instead of the prior IOU matching or unilateral ratio distribution method and secondly the yolov8 introduced a concept known as Distribution Focal Loss (DFL).[6]



2) Experiment Implementation

a) Parameters setting of Yolov8

What we use is the yolo in ultralytics:

pip install ultralytics

penguin_turtle.yaml

```
# paths:
path: ../datasets/penguin_vs_turtle
train: images/train
val: images/train
test: images/valid

names:
  0: penguin
  1: turtle

```

We configure the yaml file, give training and test paths, and provide YOLO with classification categories.

The command to train the model:

yolo train data=datasets/penguin_turtle.yaml model=yolov8m.yaml pretrained=yolov8m.pt

epochs=100 optimizer=Adam lr0=0.001 workers=4 device=0

This model used GPU with epochs of 100, Adam optimizer which is better than SGD, and initial learning rate equals to 0.001. Setting workers to 4 and device to 0 for GPU mode training guarantees much better efficiency and accuracy.

Commands for predictive models:

yolo predict model=runs/detect/train/weights/best.pt source=valid/ save=True save_txt=True

This result is the best model for prediction completion, source (predicted picture) variable that setting to valid and we save the final pictures and labels by settingg save and save_txt to true.

b) Model Evaluation Indicators

We apply a number of common measurements in the Yolov8 model. There are four outcomes that could occur: True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN), these measurements will envolved in the confusion matrix of yolov8.

The True Positive(TP): The true category of the sample is positive, and the result predicted by the model is also positive, and the prediction is correct.

True Negative (TN): The true class of the sample is a negative example, and the model predicts it as a negative example, and the prediction is correct.

False Positive (FP): The true category of the sample is a negative example, but the model predicts it as a positive example, and the prediction is wrong.

False Negative (FN): The true category of the sample is positive, but the model predicts it as a negative case, and the prediction is wrong.

The precision is the proportion of pairs predicted by the model in all outcomes where the model prediction is Positive. The equation is:

$$\text{Precision} = \frac{TP}{TP+FP}$$

The recall is in all outcomes where the true value is Positive, the proportion of the model predicts correct.

$$\text{Recall} = \frac{TP}{TP+FN}$$

Average precision (AP) refers to the precision of every element of a category of items.

$$\text{AP} = \int_0^1 p(x)dx$$

The mean of the APs for all classes is the mAP.The equation is:

$$\text{mAP} = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

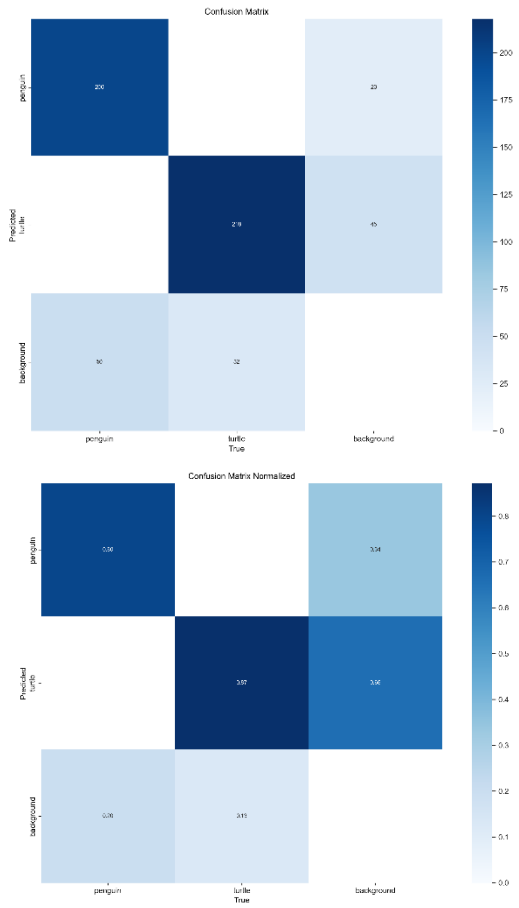
Where AP_k is the AP of class k,and n is number of classes.

The IoU statistic, also known as the Jaccard index, measures how much the anticipated and ground-truth masks overlap. In object detection, IoU can be used to assess the validity of a specific detection.[7]

$$\text{IoU} = \frac{area(gt \cap gd)}{area(gt \cup gd)}.$$

3) Results of this model

Confusion matrix.png:



The confusion matrix provides a summary of the classification problem's prediction results. Summarizing the number of correct and incorrect predictions using count values, and segmenting them by each class, is key to confusion matrices. The confusion matrix indicates which parts of the categorization model are causing problems when making predictions. It not only informs us about the classification model's errors, but also about the types of errors that are occuring. This result breakdown overcomes the restrictions of utilizing simply classification accuracy. The second plot is nomalized confusion matrix where the each column is normalized from 0 to 1.

These charts show that the accuracy of the penguin and turtle values, which were predicted with 0.80 and 0.87 respectively, is very high. It is divided into the background by prediction because the last column is the FP false positive of the background, which indicates that there are 23 penguins and 45 turtles that are not the background. The last line indicates that an object that was not there was mistakenly discovered because the background, which was initially the background, was separated into not the

background by pred. There are 20% and 13% of background detected wrong as the penguin and turtle respectively.

Here is the yolov5 results.png .

The parameters of each plot has been explained below:

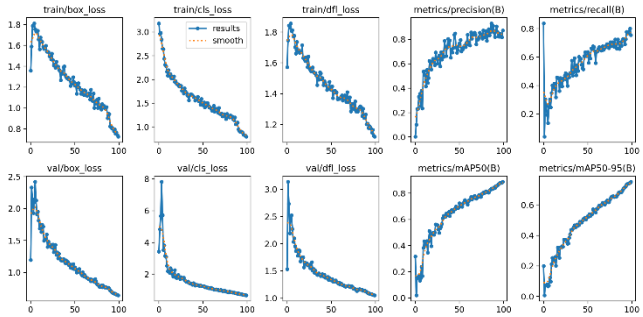
box_loss: YOLO V5 uses GIOU loss as the loss of bounding box, and Box is inferred to be the mean of the GIoU loss function, the smaller the value, the more accurate, this the bounding box loss of training set.

cls_loss: It is classification loss of which the mean of target detection, and the smaller the target, the more accurate the detection.

dfl_loss: DFL (Dynamic Freezing Loss) loss weight. The weight used to adjust the DFL loss to control how much it contributes to the total loss.

mAP50: When the IoU is set to 0.5, determine the average picture quality (mAP) for all photos in each category.

MAP50-95: represents the mean mAP at various IoU thresholds (from 0.5 to 0.95 in 0.05-step increments).



The box, classification, and dfl losses of the train and validation sets that they all steadily reduced throughout the course of iteration 100 epochs indicates that this model was learning and there was no overfitting. Because the precision and recall line does not dramatically vary and is rising, we may conclude that the training was successful. Our map0.5 performance value is close to 0.9, and the map0.5to0.95 value is close to 0.8 and rising steadily, demonstrating the model's effectiveness.

B. *Detection*

By using functions in detection.py, a dynamic size of window is chosen. Then, it will be sliding across an image with given steps and calculate the score of the pixels in the window. The score is consisting of following functions:

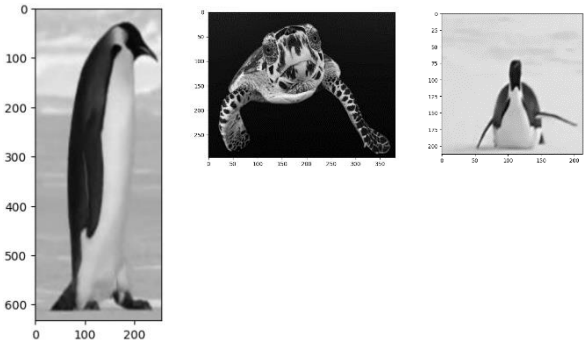
```
def evaluate_window(window):
    return np.mean(window)

def evaluate_edges(window):
    edges = cv2.Canny(window, 50, 150)
    return np.sum(edges > 0)
```

Mean value can give a basic score of pixels in the window, which can avoid getting the background of the image.

Canny is the function to find the edge of possible object. This score can help to figure out the real animal is, aiming to find small object in the image.

Some results:



The results of preprocessing image for training has more accurate features which can make the model more accurate.

C. *CNN*

1) Introduction

Convolutional Neural Network (CNN) is a deep learning architecture specialized in image processing. The functioning of the visual cortex in biology inspired its design. The basic structure of CNN consists of input layer, convolutional layer, pooling layer, fully connected layer and output layer. By stacking these layers together and tuning their parameters, a complete CNN architecture is constructed. There are generally many convolutional layers and pooling layers, and the convolutional layers and pooling layers are alternately connected.

Advantage:

1. Strong feature extraction ability: Through convolution and pooling layers, CNN can effectively capture features and patterns in images
2. Have sparse connectivity: In convolutional layers, each neuron is only connected to the input pixels associated with the local image region, which helps reduce the number of parameters and prevent overfitting.
3. Translational invariance: The filters of the convolutional layers are translational invariant, enabling robust identification of translational transformations in images.

Disadvantage:

1. High computational complexity: The computational complexity of convolutional layers increases with the depth of the network and the size of the filters.
2. Requires a large amount of data: CNNs are prone to overfitting, so a large amount of training data is required to learn the features in the image.
3. Not suitable for non-image data: The structure of CNN is mainly for image data, for other types of data, such as text data, it needs to be improved

As a widely used image classification method, CNN can obtain a more efficient processing method by iterating a more concise and efficient network structure. For example, VGGNet, GoogleNet, ResNet, and DenseNet.

2) Model

Layers	k_num	k_size	Stride
Layer1 convolutional	32	3*3	
Layer2 convolutional	32	5*5	
Layer3 polling			4
Layer4 convolutional	64	5*5	
Layer5 polling			2
Layer6 convolutional	128	3*3	
Layer7 polling			2
Layer8 fully connected		Dropout(0.6)	
Layer9 fully connected		Dropout(0.4)	

The CNN model is shown in the figure above. It used 4 convolutional layers,3pooling layers and 2 full access layers. The four convolutional layers are used respectively, 32 conlvolution kernals of 3*3, 32 conlvolution kernals of 5*5,64conlvolution kernals of 5*5, and 128 conlvolution kernals of 3*3.

Except that the first pooling layer uses a pooling area with a Stride of 4, the other two use a pooling area with a Stride of 2. The two fully connected layers set Dropout to 0.6 and 0.4 respectively.

The fine-tuning data of the pooling layer, convolutional layer, and fully connected layer can be calculated according to related formulas.

If convolution kernels of different sizes are used for feature extraction, convolution kernels of different sizes can represent different receptive fields, so different convolution kernels are used to extract the expression features of different receptive fields. The expression of the convolution layer is as follows

$$C_i = f(x * w_i + b_i)$$

Among them, Ci represents the output result obtained by the first convolution, represents the activation function, and the activation function selects Rectified Linear Units （ ReLU）, x represents the input image value,* represents the convolution operation, wi represents the first convolution kernel, and bi represents the bias of the first convolution kernel. The expression of the ReLU function is as follows ：

$$\mathrm{ReLU}\,(y) = \begin{cases} y, y \geq 0 \\ 0, y < 0 \end{cases}$$

where y refers to （x*wi+bi） in the previous formula.

The same pooling layer also has a corresponding formula：

$$Si = down(\max(ya, b))a, b \in pi$$

Where Si represents the maximum pooling result of the i-th pooling area, down（·） represents the downsampling process （retaining the maximum value of the pooling area ）, ya, b represents the value in the pooling area, and pi represents the i-th pool area. The role of the pooling layer is to reduce the dimensionality behind the convolutional layer, because the dimensionality of the operating features after the convolutional layer will increase and eventually cause a disaster of dimensionality.

$$Full = f(w \times z + b)$$

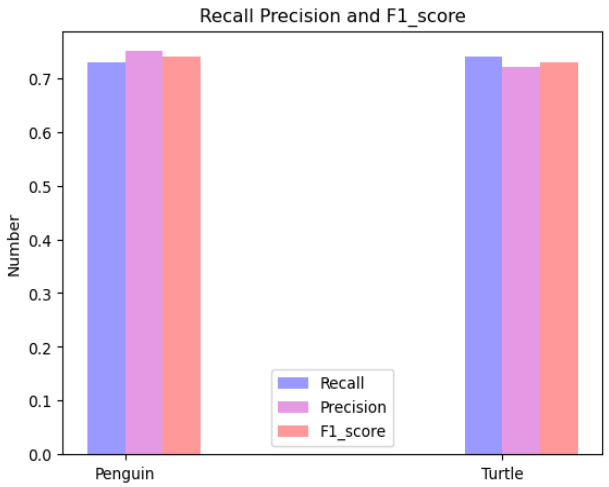
The formula of the fully connected layer is shown in the figure above. Among them, Full indicates the output result of the fully connected layer, f(·) is the ReLU activation function, w represents the weight value of the connection, z is the value of the input fully connected layer, and b is the bias. I used two fully connected layers. Therefore, in order to reduce and prevent overfitting, Dropout is used.

3) The Results

```
Accuracy:  0.7361111111111112
Precision :  0.7297297297297297
F1 Score:  0.7397260273972601
Recall:  0.75
```

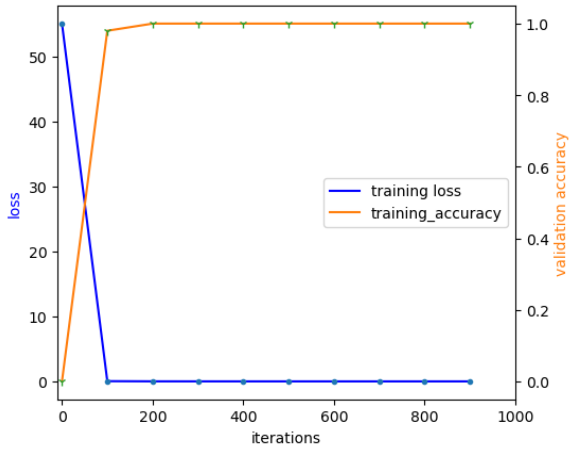
The picture shows the results I got from one of the runs. We can see that Accuracy is 0.7361, Precision is 0.7297, F1_score is 0.7397.

Recall is 0.75. According to these results, we can conclude that the CNN method used in this essay has a very efficient image classification ability for this dataset. The running time is 5m36s, and the processing time is not very long. Part of the reason is that the data set is not very large. On the other hand, the environment configuration of the code used in this article is tensorflow 1.0-CPU. Due to version limitations, more efficient languages cannot be used, and the processing speed of cpu is also slower than that of gpu. I might get better results if I use tensorflow2.0 - GPU.

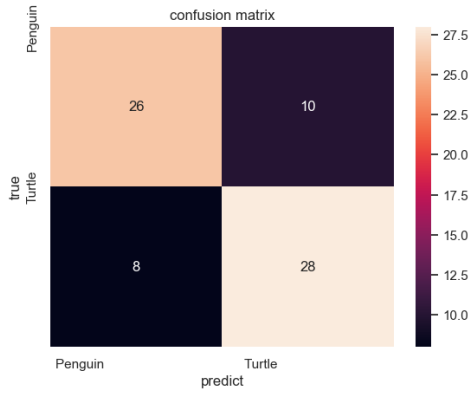


According to the data in the figure, we can see that the method has similar processing effects on two different types of images (Penguin, Turtle). The Recall precision and F1_score data of the two images are highly similar, and both have very satisfactory results. Of course, this may also be because there are only 72 test data sets. At the same time, the animals in the image are clearly distinguished from the

background, which is very suitable for CNN to classify images.



Here is a plot of loss and accuracy for the training set. It can be seen that the overall error is close to 0 fitting, and the accuracy is close to 100%, which shows that the learning efficiency of CNN is very good.



It can be seen from the figure that 26 of the 36 penguins were correct and 8 were misjudged. Of the 36 turtles,28were correct and 6 were misjudged. The obtained results are consistent with the previously obtained accuracy, recall, precision, f1_score result.

V. Discussion

Our project delves into the exploration of diverse object detection models, each harnessing unique methods and structures to tackle the challenge of detecting objects within images. Among the models under investigation are YOLOv8 and CNN, each offering distinct characteristics that manifest in a wide range of outcomes.

The CNN, as a relatively basic algorithm, boasts a remarkable feature learning ability, high accuracy, and transferability. Its potential in image recognition tasks has been widely recognized. However, during our experimentation, we encountered some limitations in its performance. The accuracy of our CNN model, measured during the computation process, hovered around 78%, falling short of our expectations. Moreover, we occasionally encountered abrupt errors during the execution, raising concerns about the code implementation within our group or the potential mismatch between the CNN model and the specific problem at hand.

On the other hand, YOLOv8 presented itself as a promising alternative with a plethora of advantages. Its real-time performance sets it apart, designed specifically for efficient real-time object detection, which proves valuable in applications requiring rapid response times. The end-to-end optimization approach streamlined the model's design and usage, simplifying implementation and deployment. Furthermore, YOLOv8 excelled in detecting small objects, gaining an edge in scenarios with densely packed targets. Unlike two-stage detection models, YOLOv8 processes the entire image in a single pass, enabling faster and more efficient detection. Our experimentation with YOLOv8 yielded an impressive accuracy of 92.2%.

Conversely, other models like Faster R-CNN fell short in competitive accuracy, coupled with significantly slower inference times. While YOLOv8 showcased competitive accuracy, it necessitated larger model and input sizes to achieve such results. Nonetheless, when considering remote sensing with SAR data, the superior performance of YOLOv8 stands out, making it the recommended architecture for accurate and fast object detection with easy customization options.

The advantages of YOLOv8 extend beyond its exceptional accuracy and speed. Its multi-scale feature fusion capability enables the capturing of features from objects of varying sizes, enhancing its adaptability to different scenarios. These advantages make YOLOv8 a highly suitable solution for image classification problems, particularly in the context of remote sensing with SAR data.

Throughout our exploration of these models, we have gained valuable insights into the nuances and strengths of each approach. The journey of experimenting, fine-tuning, and comparing different architectures has been both challenging and rewarding. The knowledge acquired during this project will undoubtedly fuel future advancements in object detection and image classification, contributing to a deeper understanding and utilization of these powerful tools in a multitude of applications.

In conclusion, the YOLOv8 model has proven itself as a standout choice for accurate and fast object detection, especially when dealing with SAR data in remote sensing applications. Its real-time performance, end-to-end optimization, and ability to excel in detecting small objects make it a versatile and efficient solution. As we move forward, we remain committed to refining our models, exploring new possibilities, and applying these advancements to real-world challenges. The journey of innovation continues, and we eagerly anticipate the transformative impact of these object detection models in diverse domains.

VI. Conclusion

In this experiment, we conducted in-depth research on two different object detection models, YoloV8 and CNN, and obtained their respective results. These experiments provided valuable insights, allowing us to gain a deeper understanding of the performance and applicability of these two models.

Firstly, we found that CNN, serving as the foundation of Yolo, contributes to its powerful feature learning capability. CNN excels in image recognition tasks, exhibiting high accuracy and transferability, which has led to remarkable achievements in various image-related problems. However, during our experimentation, we observed that the accuracy of the CNN model was relatively low, around 78%. Moreover, we encountered occasional unexpected errors during the execution, possibly due to coding issues within our group or the CNN model not being entirely suitable for addressing the specific problem we researched.

In contrast, YoloV8 showcased exciting advantages. Firstly, it demonstrated outstanding real-time performance, designed to efficiently achieve real-time object detection, making it suitable for applications requiring rapid responses. Secondly, YoloV8 adopts an end-to-end optimization training approach, simplifying the model's design and usage, making implementation and deployment more straightforward. Additionally, YoloV8 exhibited excellent performance in detecting small objects, granting it a clear advantage in scenarios with dense target presence. Another exciting feature is that, unlike traditional two-stage detection models like R-CNN, YoloV8 can process the entire image in a single pass, achieving faster and more efficient detection. In our experiments, YoloV8's accuracy was impressively high, reaching 92.2%.

On the other hand, other models such as Faster R-CNN did not demonstrate competitive accuracy and had significantly slower inference times. While YoloV8 also showed competitive accuracy, achieving such results required larger model and input sizes.

In conclusion, for image classification tasks, especially in contexts involving image recognition and object detection, we can draw the conclusion that YoloV8 is more suitable than CNN. YoloV8's strengths lie in its high accuracy, fast processing speed, and excellent performance in detecting small objects. These advantages make YoloV8 an ideal choice for solving image classification problems, providing robust support for our future research and applications. As we continue to delve into further research and technological advancements, we look forward to the widespread application of object detection models in various domains, presenting us with more opportunities and challenges.

Therefore, YoloV8's superior performance makes it the recommended architecture for accurate and fast object detection with easy customization options.

VII. References

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[2] Sharma H, Jain J S, Bansal P, et al. Feature extraction and classification of chest x-ray images using cnn to detect pneumonia[C]//2020 10th international conference on cloud computing, data science & engineering (Confluence). IEEE, 2020: 227-231.

[3] Jocher, Glenn. "Yolov5 documentation." (2020).

[4] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).

[5] Solawetz, J. "What is YOLOv8? The Ultimate Guide." (2023).

[6] Buhl, N. (2023, February 8). YOLO models for Object Detection Explained [YOLOv8 Updated]. https://medium.com/cord-tech/yolov8-for-object-detection-explained-practical-example-23920f77f66a

[7] Lin T-Y et al (2014) Microsoft COCO: common objects in context. In Computer Vision – ECCV 2014, Springer International Publishing, Cham, pp 740–755