ADVANCED IMAGE PROCESSING

ASSIGNMENT 4

NAME - JYOTISH RANJAN

M. Tech AI, 2023

Qn 1:

CODE IMPLEMENTATION:

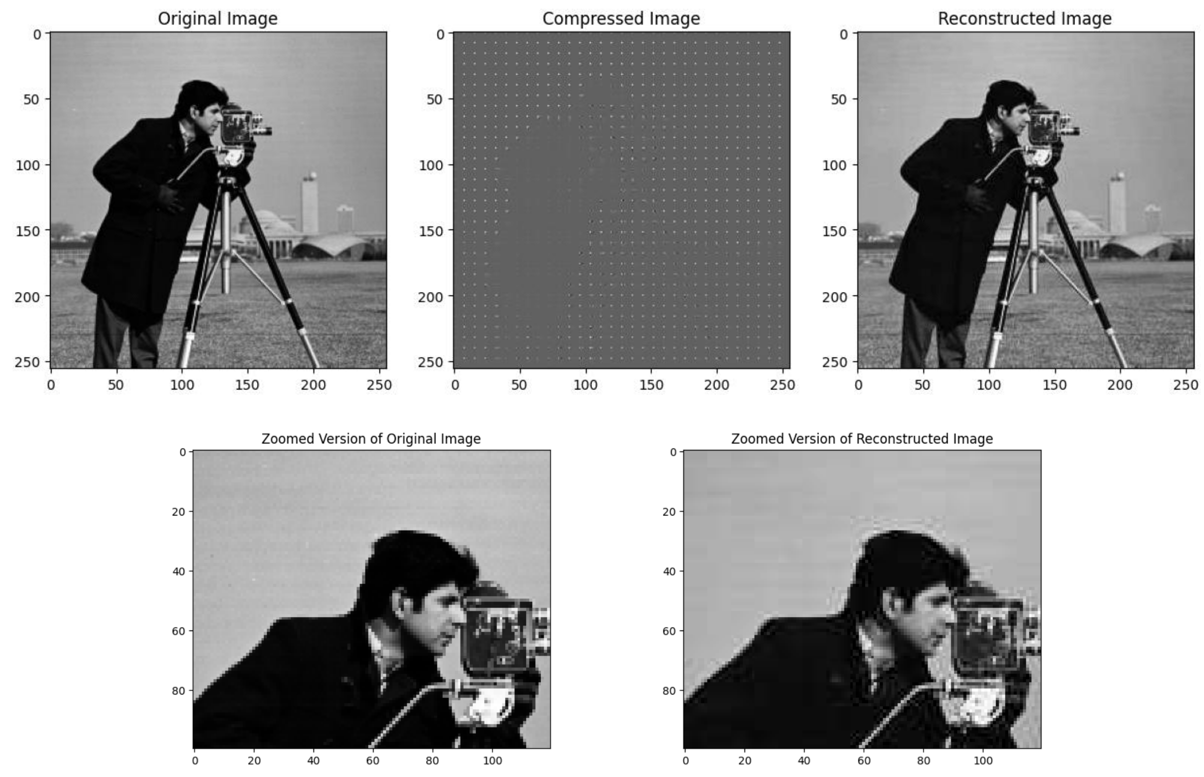
CODE link

1. **Building quantization matrix to quantize each DCT coefficient in a given 8x8 block.**
2. **Transform: Computing an 8x8 discrete cosine transform (DCT) for every non overlapping block in the input grey scale image.**
3. **Quantization: Using the quantization matrix to quantize each DCT coefficient in a given 8 x8 block.**
4. **Using Zigzag scan to read the patch which helps identifying and removing high-frequency components during compression.**
5. **Lossless source coding: Using the given table to encode the quantized index corresponding to each DCT coefficient.**
6. **Computed the mean squared error between the reconstructed image and the original image. And found the compression ratio.**
7. **Using PIL library in Built Function, find the compressed image size which has approximately same MSE as calculated using custom function.**

Results:

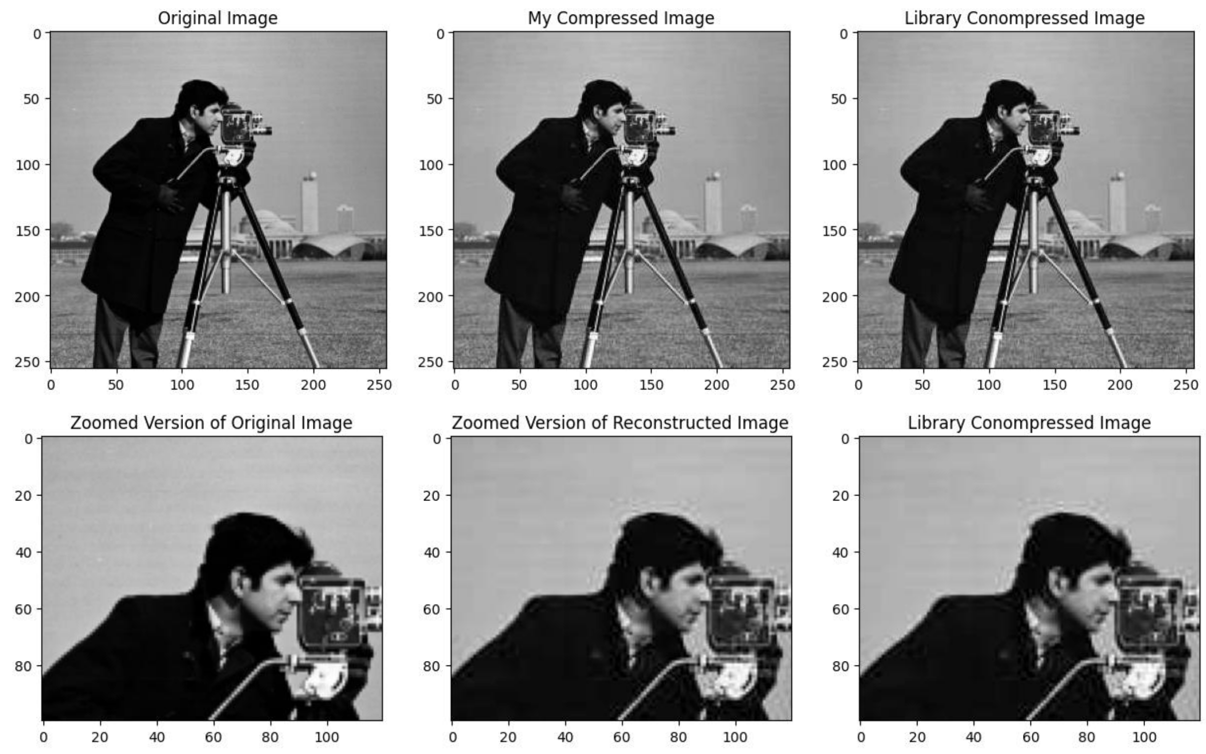
1. **Using Custom Function**

   - **Original Image file size in bits: 524288**
   - **Compressed Image file size in bits Using Custom Function: 108260**
   - **Compressed Ratio: 4.84286**
   - **Mean Squared Error between the Original and Reconstructed Image: 25.96**


Original Image
Compressed Image
Reconstructed Image
Zoomed Version of Original Image
Zoomed Version of Reconstructed Image

2. Using in Built Function:

- **Quality Parameter for similar MSE is : 49**
- **Compressed file size with default JPEG compression: 56984 bits**
- **Compression Ratio : 9.20**


Original Image
My Compressed Image
Library Conompressed Image
Zoomed Version of Original Image
Zoomed Version of Reconstructed Image
Library Conompressed Image

Observations:

1. The mean squared error between the original and reconstructed image is 25.96, which indicates that the quality of the reconstructed image is quite low.
2. This is because the quantization matrix used in the compression technique is not very fine-grained, resulting in high quantization error and loss of image details.
3. The compression ratio achieved in this case is 4.84, which means that the compressed image is approximately 4.84 times smaller than the original image. This is because the high-frequency components of the image are removed during compression, resulting in a significant reduction in image data.
4. Overall, the above analysis highlights the trade-off between compression ratio and image quality in lossy image compression techniques, and the importance of selecting an appropriate quantization matrix to balance the two.
5. Using in built functions gives better compression ratio (9.2) for same MSE. This is due to sophisticated techniques used in current state of art JPEG compression techniques.

Qn 2:

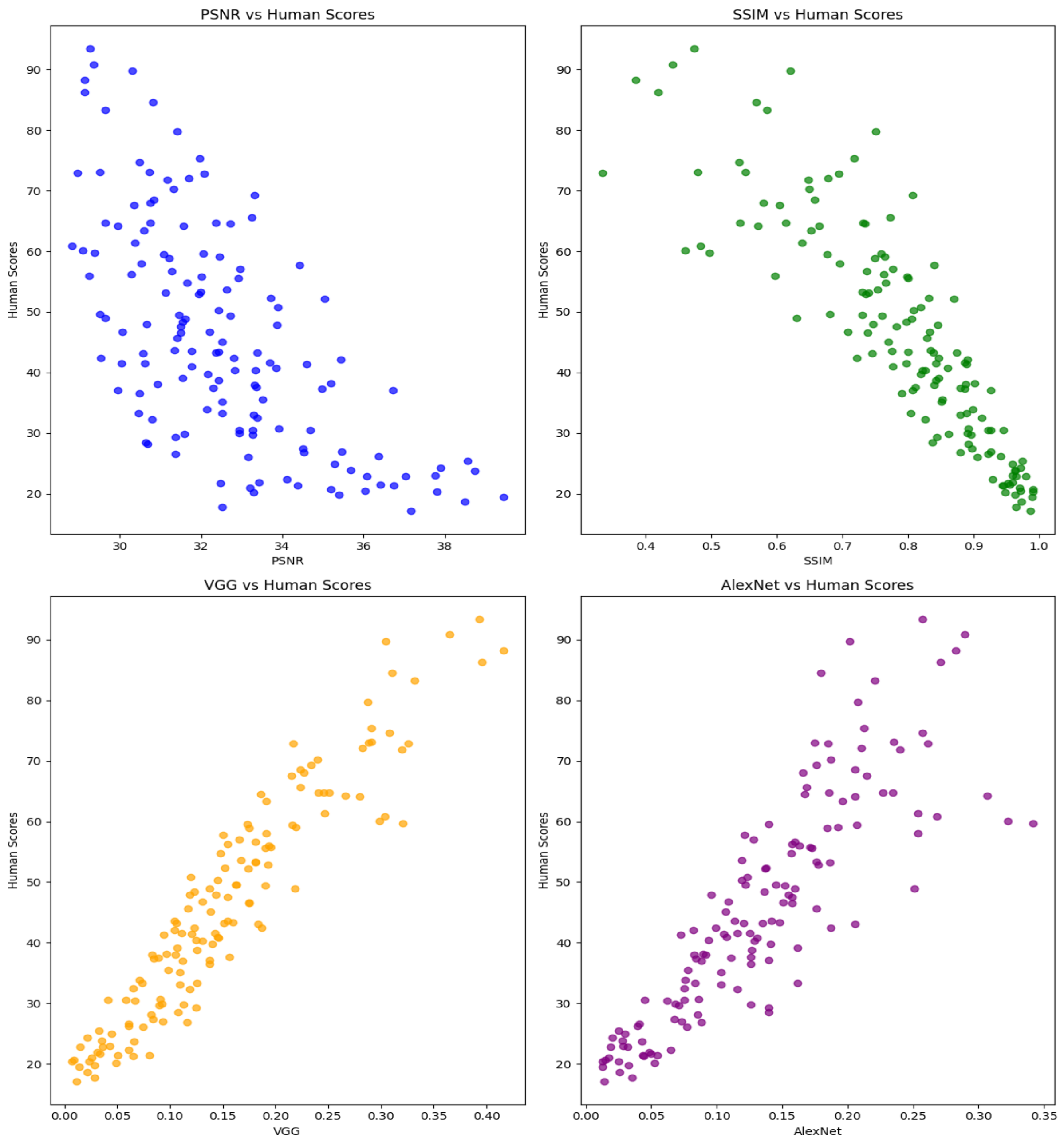
CODE IMPLEMENTATION:

CODE link

1. **Load data from a given file containing human opinion scores and image names.**
2. **Calculate image quality metrics (PSNR, SSIM, VGG, Alex Net, Human) for distorted images compared to their corresponding reference images.**
3. **Save the calculated metrics into files to avoid recomputation.**
4. **Compute Spearman rank-order correlation coefficients between each metric and human opinion scores.**
5. **Print out the correlation coefficients to assess the relationship between different metrics and human perception of image quality.**

Results:

- **Spearmen correlation coefficient between Human and PSNR is : - 0.660**
- **Spearmen correlation coefficient between Human and SSIM is: - 0.92**
- **Spearmen correlation coefficient between Human and VGG is : 0.946**
- **Spearmen correlation coefficient between Human and Alex Net is : 0.901**


PSNR vs Human Scores
SSIM vs Human Scores
VGG vs Human Scores
AlexNet vs Human Scores
Human Scores
PSNR
SSIM
VGG
AlexNet

Observations :

1. **PSNR vs Human Scores: PSNR as weak correlation with human scores because this metrics concentrates on pixel by pixel comparison and are don't capture the nuances of human visual perception. Humans are more sensitive to structural distortions and artefacts than just raw intensity variations.**

2. **SSIM vs Human Scores: SSIM is strongly negative correlated with Human Scores. This metric incorporates aspects of luminance, contrast, and structure similarity, therefore showcasing strong correlation with human scores..**

3. **Perceptual quality metrics derived from deep learning models like VGG and AlexNet are highly correlated with human perception of image quality.**

   Reasons:
   a. **Deep learning models can be trained on diverse datasets, making them adaptable to a wide range of image types, distortions, and applications. In contrast, traditional methods like SSIM and PSNR are often designed with specific assumptions about the image and noise characteristics, limiting their applicability in certain scenarios.**
   b. **Deep learning models can learn complex patterns and features from data, allowing them to better capture the perceptual similarity between images.**

Qn 3:

CODE IMPLEMENTATION:

Steps:

1. **Cloning the YOLOv7 repository from the specified GitHub URL and downloading pre trained weights.**
2. **Updating data.yaml file.**
3. **Training and Testing using provided dataset.**

Observations:

1. **Qualitative Analysis:**
   - The model performs reasonably well across all classes, with an average precision around 0.79 to 0.82 and recall around 0.78 on both training and test sets. This indicates a good balance between correctly identifying positive instances (high recall) and avoiding false positives (moderate precision).
   - There is room for improvement, especially in the higher IoU range (mAP@0.5:0.95 around 0.51), suggesting some challenges in precisely localizing object boundaries.

2. **Individual class:**
   - Fish: The model exhibits high precision (~0.82) but a slightly lower recall (0.77), suggesting it may be conservative in predicting fish instances, potentially missing some true positives. The performance is consistent across training and test sets.
   - Jellyfish: This class stands out with very high recall (0.91 to .94), indicating the model effectively captures most jellyfish instances. The precision is also good (0.84 to 0.86), and the class achieves the highest mAP@0.5 (0.95), demonstrating strong overall performance.
   - Penguin: Both precision and recall are moderate (~0.69-0.76), with mAP values around 0.33-0.35, indicating this class is more challenging for the model. The performance is consistent across datasets.
   - Puffin: This class has the lowest performance among all classes, with precision around 0.69, recall around 0.67, and low mAP values (~0.30). The model struggles to accurately identify and localize puffin instances, likely due to their unique appearance.
   - Shark: The model exhibits good precision (~0.94 to .96) but lower recall (~0.67), suggesting it may be conservative in predicting sharks, potentially missing some true positives. The mAP values (~0.51) indicate moderate overall performance.
   - Starfish: While the precision is perfect (.95) on the training set, indicating no false positives, the recall of 0.73 suggests the model misses some starfish instances. The test set shows lower but still good precision (0.901) and recall (0.66).
   - Stingray: Both precision and recall are consistently high (~0.86) across datasets, with mAP values around 0.79, indicating good overall performance for this class.

3. **Comparison:**
The evolution from YOLO v1 to YOLO v7 has brought significant improvements in architecture design, loss function, and computation complexity, addressing some of the drawbacks of the original YOLO v1 model.

- Architecture design:
  - YOLO v1: Used a single fully convolutional network to predict bounding boxes and class probabilities.
  - YOLO v7: Adopts a more sophisticated architecture with advanced features, enabling better feature extraction and representation.

  These architectural changes improve the model's ability to capture complex spatial and channel-wise relationships, leading to better object detection performance.

- Loss function:
  - YOLO v1: Utilized a single loss function that combined classification, localization, and confidence components.
  - YOLO v7: Employs more advanced loss functions, such as the IoU-based loss (e.g., GIoU, DIoU), which directly optimize the intersection over union (IoU) between predicted and ground truth bounding boxes.

  These improved loss functions provide better gradient signals for bounding box regression, addressing the localization challenges faced by YOLO v1.

- Computation complexity:
  - YOLO v1: Had a relatively high computational complexity due to its fully convolutional nature and the need to evaluate the entire image at multiple scales.
  - YOLO v7: Incorporates techniques like model pruning, quantization, and efficient attention mechanisms to reduce computational complexity without sacrificing performance.

  These optimizations make YOLO v7 more efficient and suitable for deployment on resource constrained devices or real-time applications.

4. **The changes proposed in YOLO v7 address several drawbacks of YOLO v1**:

   a) Improved architecture design:

   The incorporation of advanced architectural enhances the model's ability to capture long-range dependencies and focus on relevant features, leading to better object detection performance.

   b) Advanced loss functions:

   The use of IoU-based losses directly optimizes the model's bounding box regression capability, addressing the localization challenges faced by YOLO v1, which relied on a more simplistic loss function.

   c) Reduced computation complexity:

   The introduction of techniques like model pruning, quantization, and efficient attention mechanisms helps reduce the computational burden of YOLO v7 compared to YOLO v1, making it more suitable for real-time applications or deployment on resource-constrained devices.

   d) Data Augmentation Strategy:

   YOLO v1 relied on traditional, predefined data augmentation techniques like random cropping, flipping, and scaling, which may not be optimally tailored to the specific dataset and task.

   YOLO v7 incorporates the Trainable Bag of Freebies concept, which allows the model to learn and adapt its own augmentation strategies during training. An additional Augmentation Network module generates augmentation parameters based on the input image and the current state of the model. The loss function is modified to encourage the Augmentation Network to produce augmentations that improve the model's performance. This dynamic and adaptive augmentation approach addresses the limitations of fixed, generic augmentation techniques used in YOLO v1, potentially leading to better generalization and robustness.

   Overall, the evolution from YOLO v1 to YOLO v7 represents a significant improvement in object detection capabilities, addressing key limitations of the original YOLO model while enhancing performance, accuracy, and efficiency.

Results:

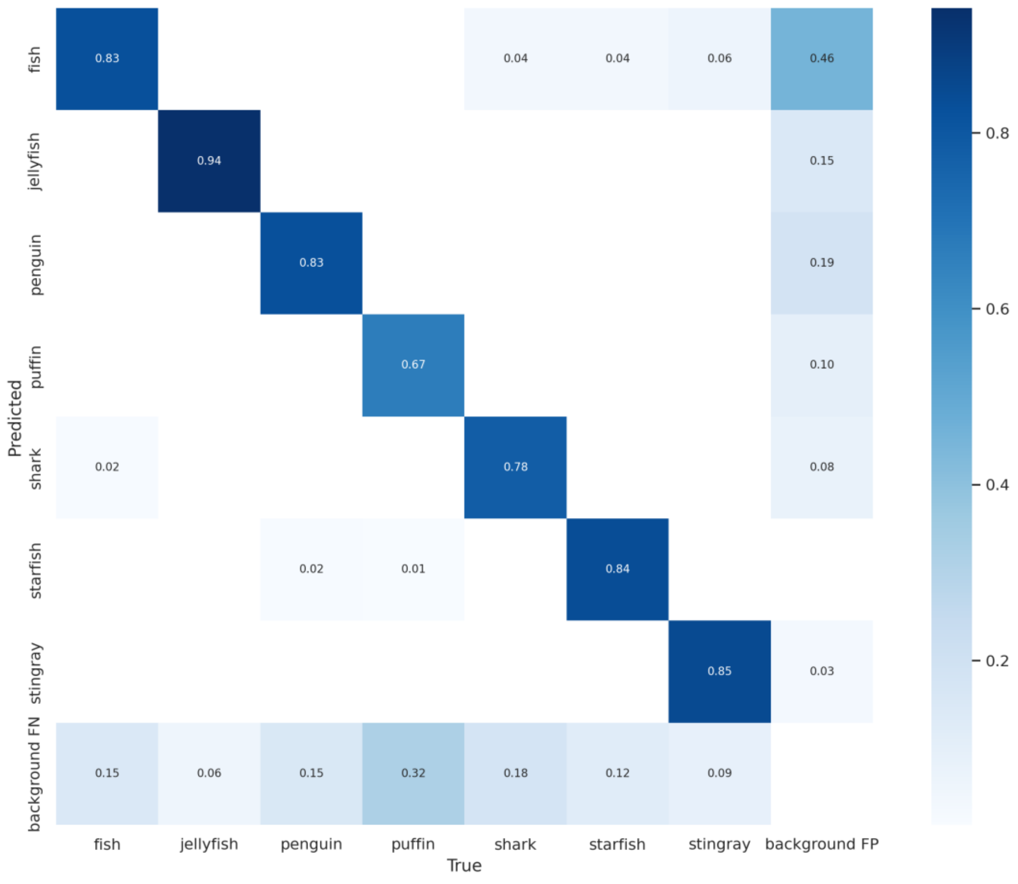
1. Testing Dataset :

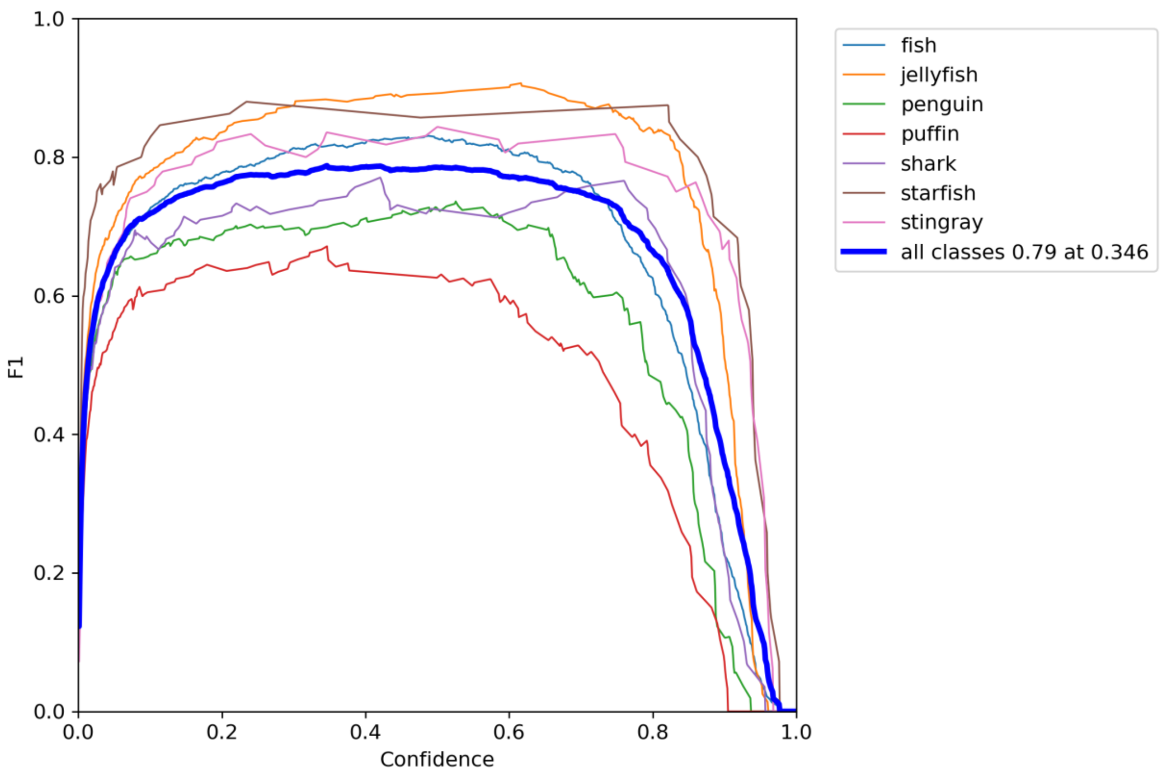
Class	P	R	mAP@.5	mAP@[.5 , .95]
all	0.827	0.777	0.824	0.5
fish	0.819	0.81	0.856	0.51
jellyfish	0.866	0.919	0.953	0.563
penguin	0.692	0.76	0.72	0.334
puffin	0.692	0.676	0.686	0.306
shark	0.904	0.662	0.812	0.513
starfish	0.948	0.815	0.87	0.654
stingray	0.868	0.798	0.868	0.654

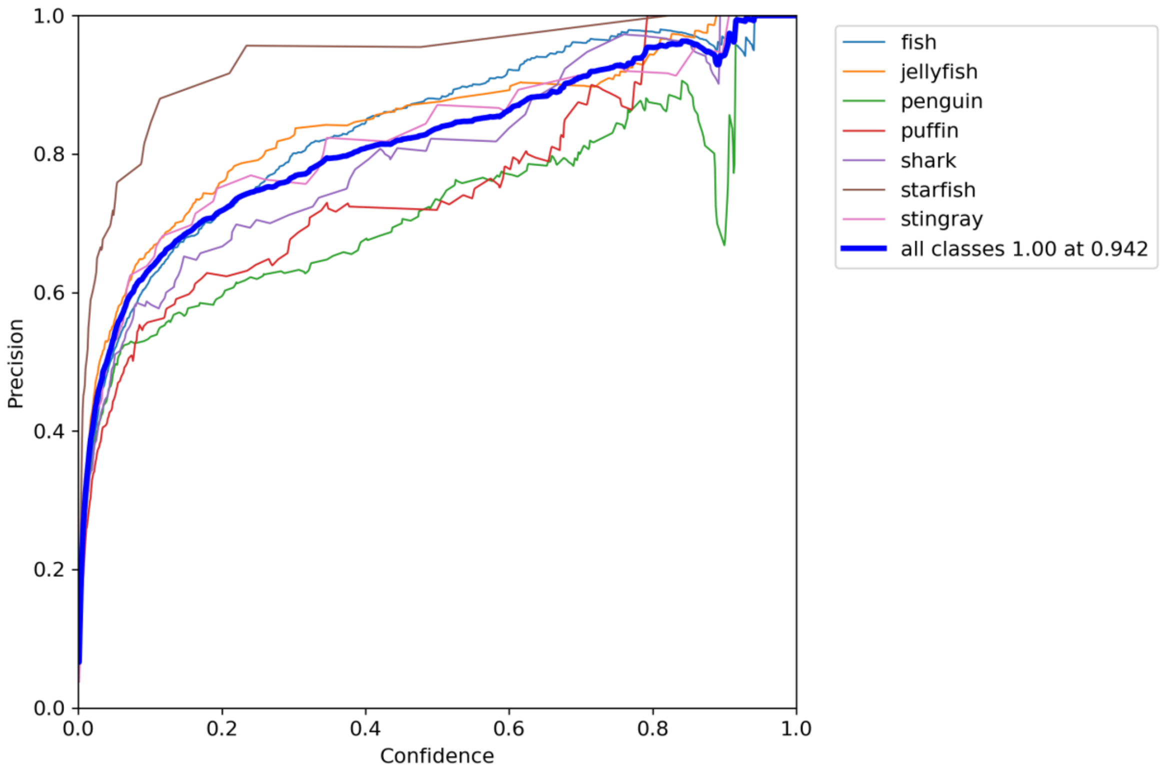
2. Training Dataset :

Class	P	R	mAP@.5	mAP@[.5 : .95]
all	0.794	0.786	0.816	0.51
fish	0.822	0.81	0.865	0.513
jellyfish	0.842	0.929	0.933	0.544
penguin	0.648	0.76	0.727	0.347
puffin	0.73	0.622	0.645	0.298
shark	0.741	0.737	0.805	0.547
starfish	0.956	0.798	0.87	0.687
stingray	0.823	0.848	0.864	0.626

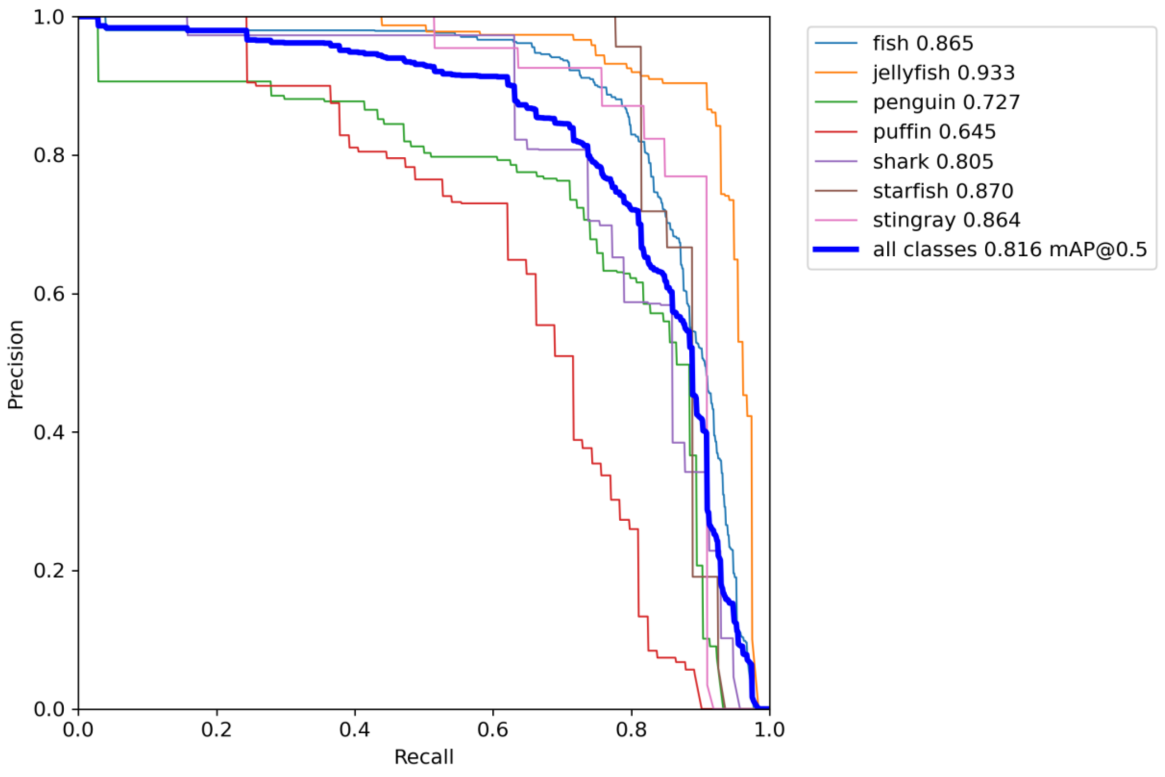
3. Training Plots :



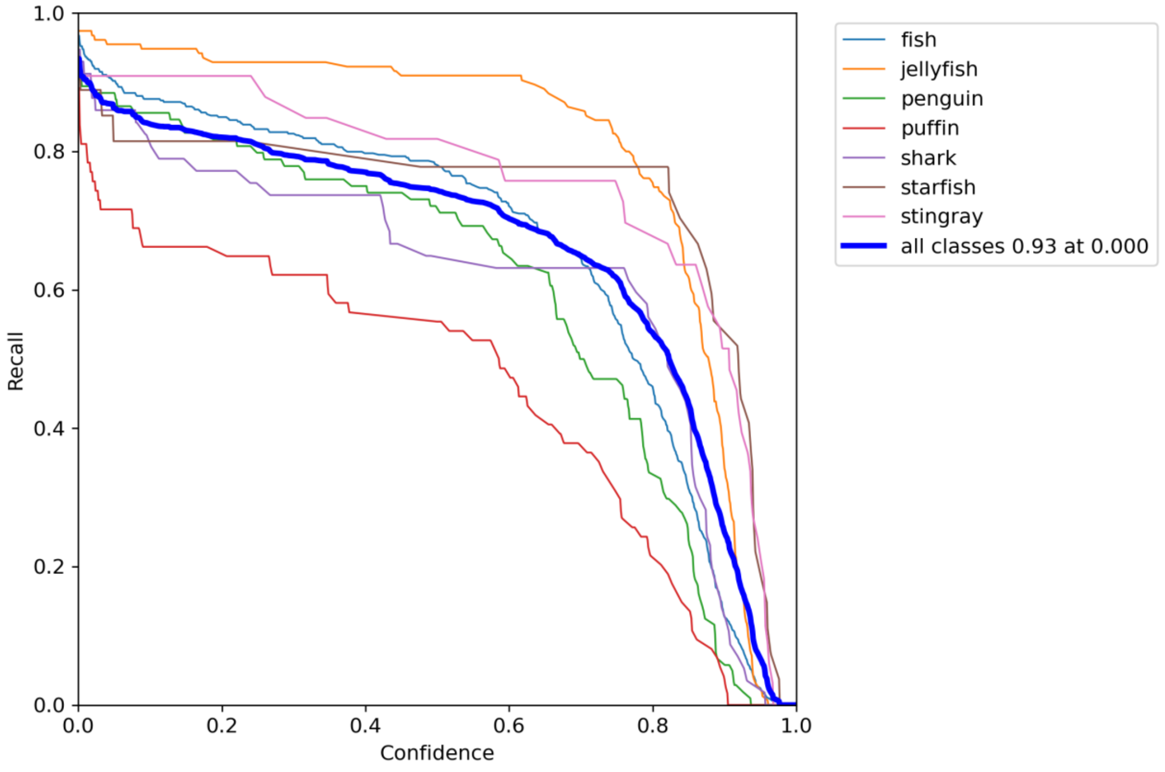




1.0
0.8
0.6
0.4
0.2
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Confidence
fish
jellyfish
penguin
puffin
shark
starfish
stingray
all classes 1.00 at 0.942

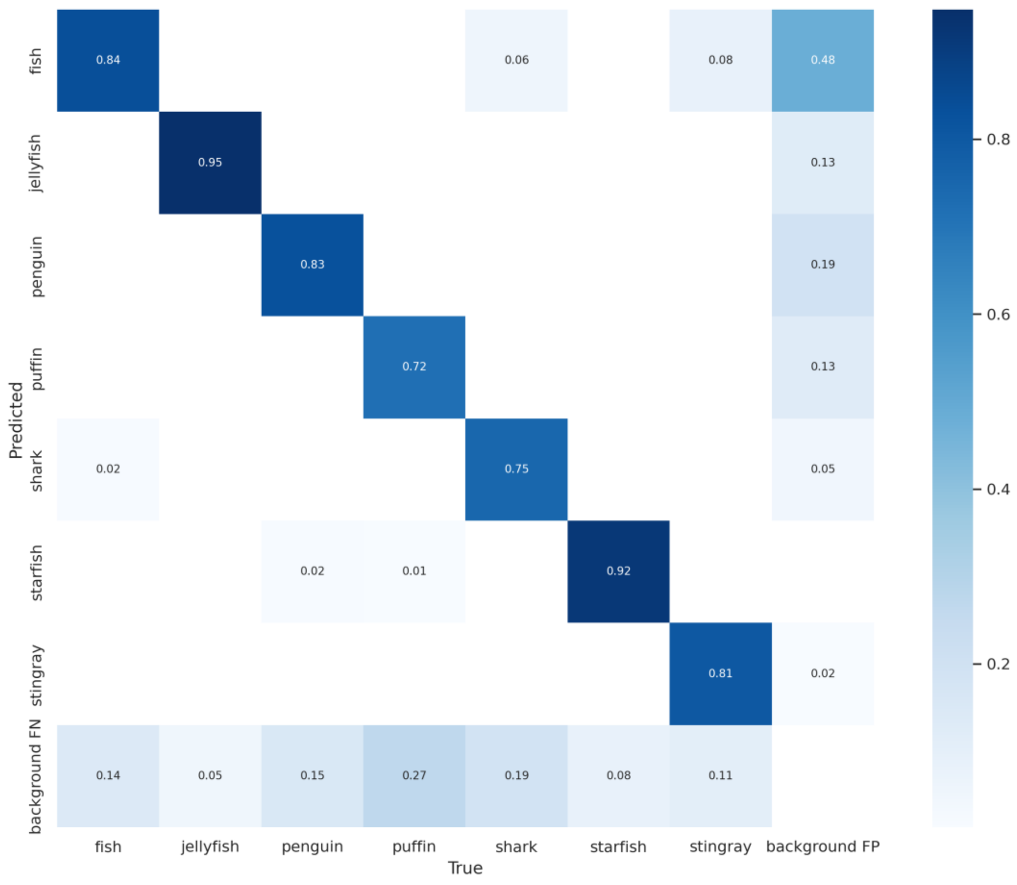


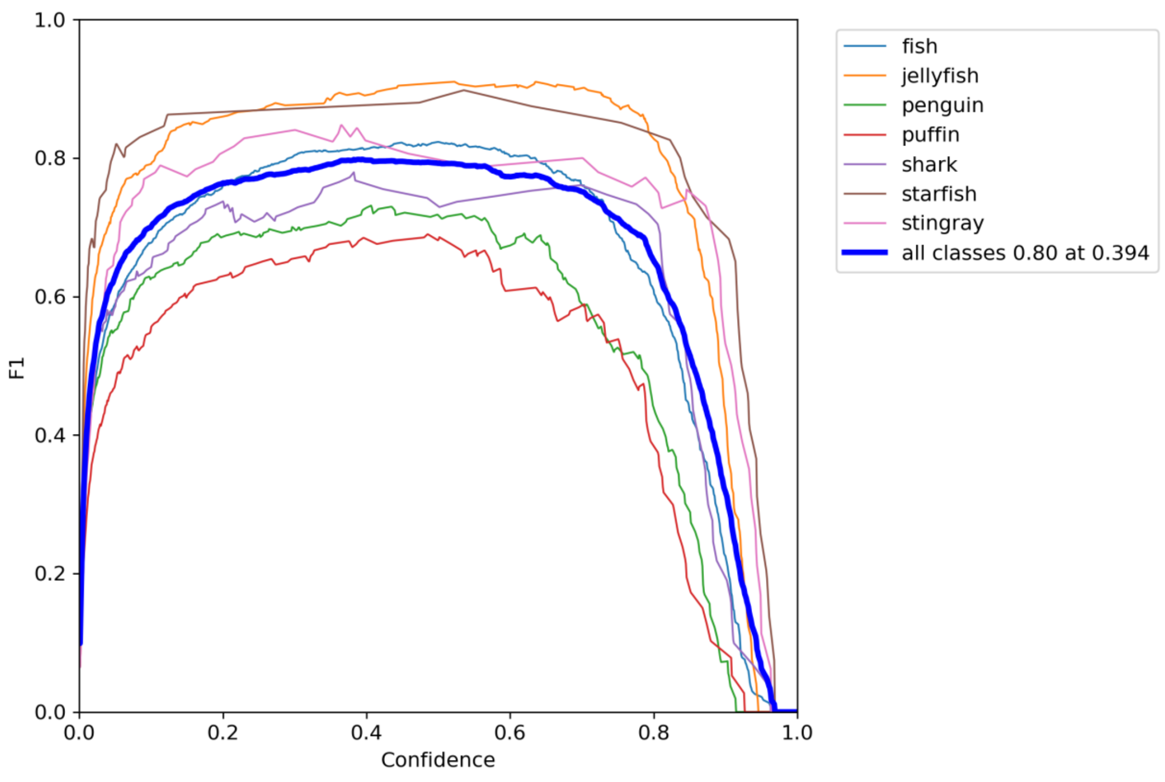
1.0
0.8
0.6
0.4
0.2
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall
fish 0.865
jellyfish 0.933
penguin 0.727
puffin 0.645
shark 0.805
starfish 0.870
stingray 0.864
all classes 0.816 mAP@0.5

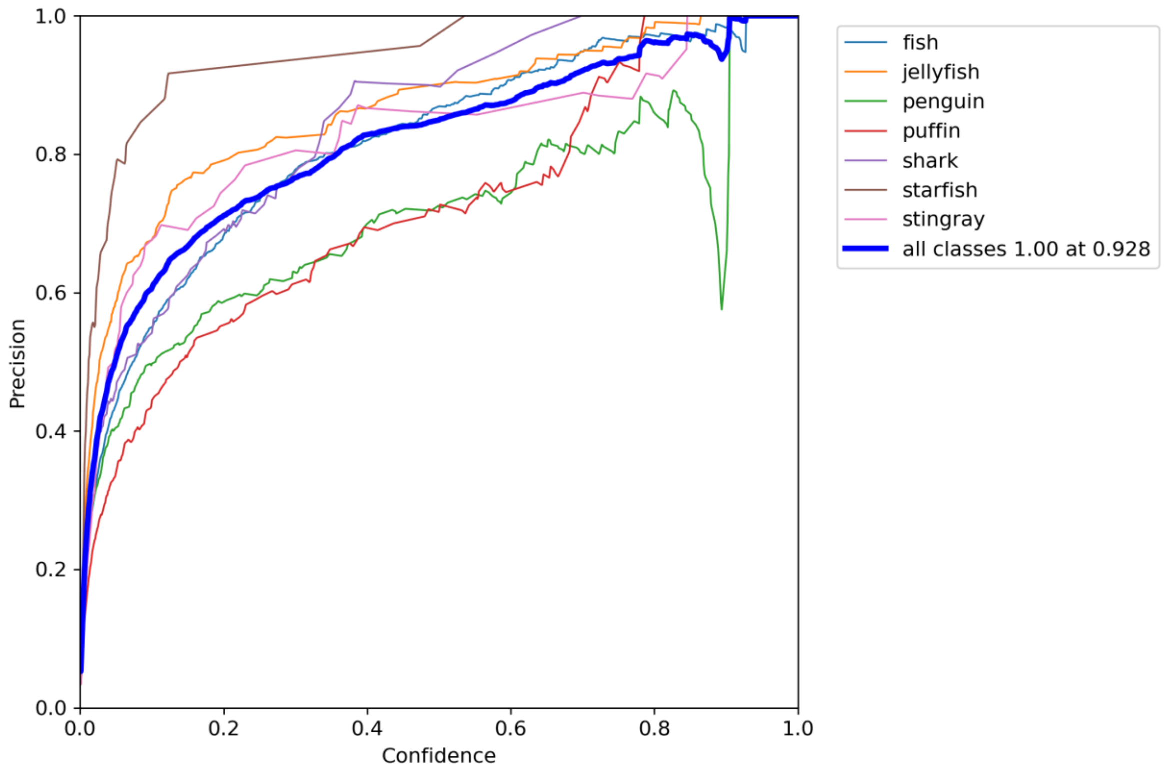


1.0
0.8
0.6
0.4
0.2
0.0
Recall
0.0
0.2
0.4
0.6
0.8
1.0
Confidence
fish
jellyfish
penguin
puffin
shark
starfish
stingray
all classes 0.93 at 0.000

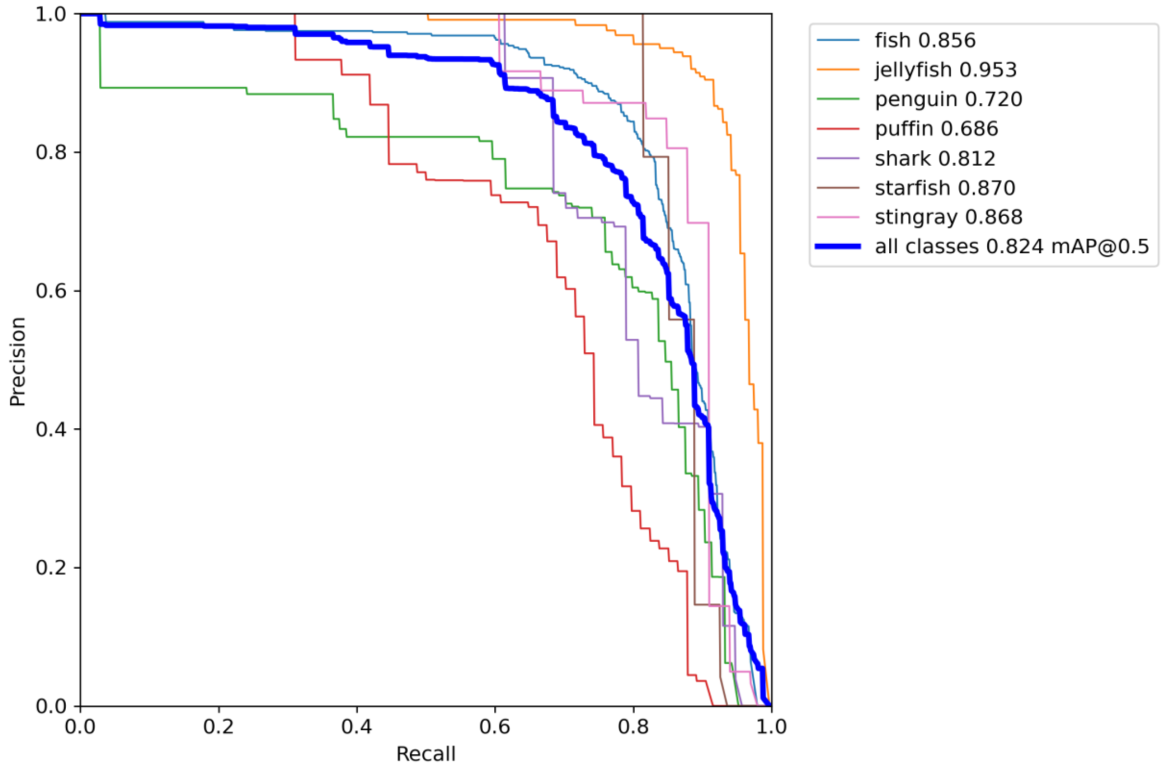
4. Testing Plots :



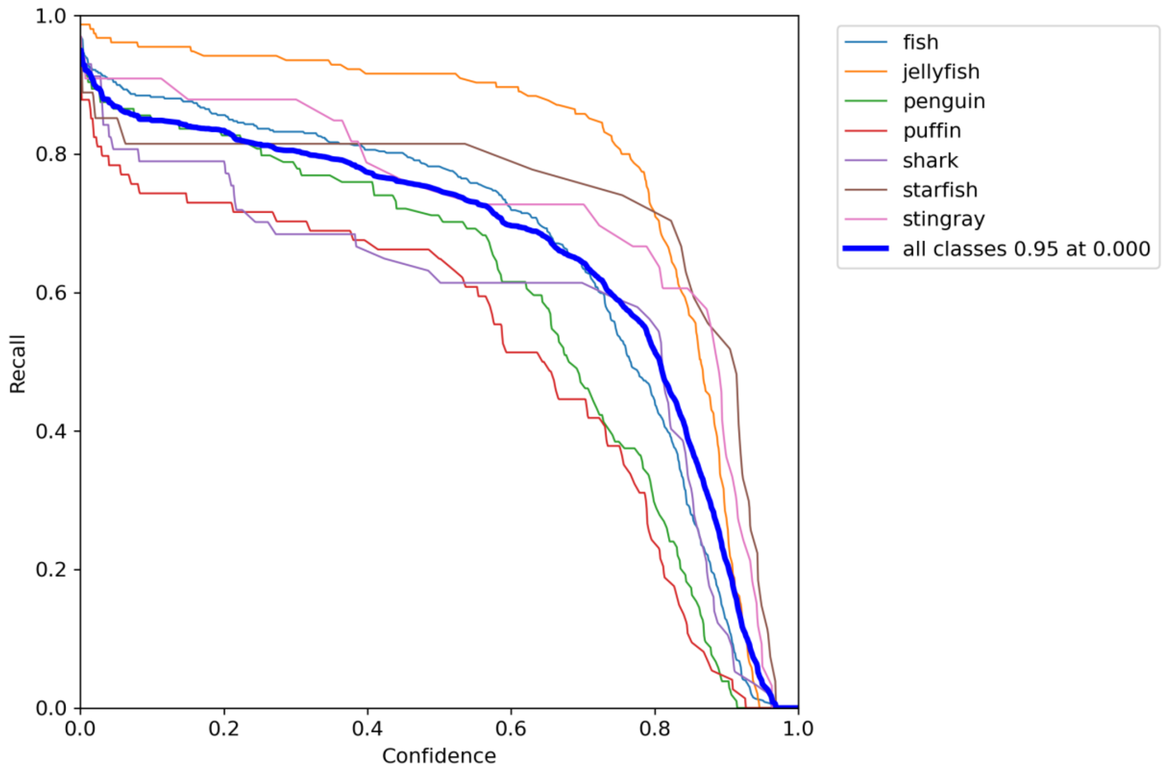




Precision
Confidence
fish
jellyfish
penguin
puffin
shark
starfish
stingray
all classes 1.00 at 0.928



Precision
Recall
fish 0.856
jellyfish 0.953
penguin 0.720
puffin 0.686
shark 0.812
starfish 0.870
stingray 0.868
all classes 0.824 mAP@0.5



1.0
0.8
0.6
0.4
0.2
0.0
Recall
0.0
0.2
0.4
0.6
0.8
1.0
Confidence
fish
jellyfish
penguin
puffin
shark
starfish
stingray
all classes 0.95 at 0.000