VRDL Final Project Report

Topic: NOAA Fisheries Steller Sea Lion Population Count

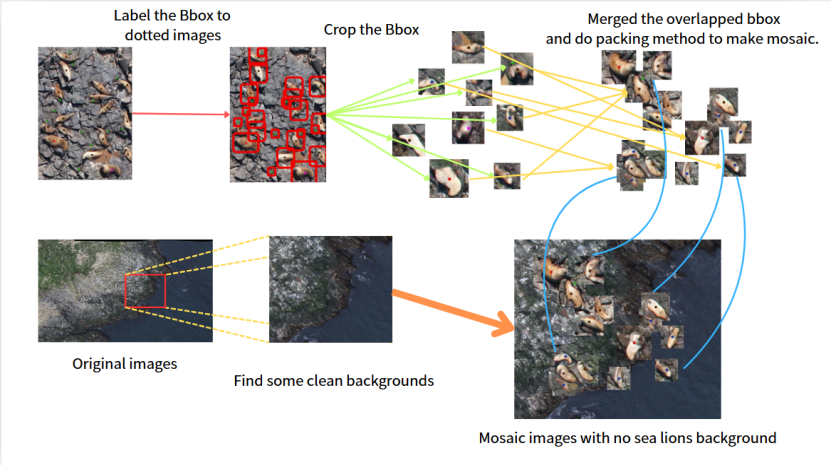
Github link: https://github.com/sharkccy/VRDL_final

Group 16: 110612117 張仲瑜, 110652032 許元瑞
110550128 蔡耀霆, 109612019 林伯偉

I. Introduction:

1. Competiton:
   Checking the effectiveness of sea lion conservation is crucial since the number of them has decreased over 94% during recent years; however, counting the number of them manually is time consuming and laborious. As a result, a modern automatic counting method is required.
2. In this project, we come up with innovative methods aiming to boost the performance compared to existing methods.
3. Problems and Solver
   - High resolution data (over 3000 * 5000) which requires a patch dividend process, the memory requirement is extensive.
   - There are 18000 images in the testing dataset that takes more than 6 hours to predict in most of the methods, time consuming in any movement.
   - We failed to achieve a top satisfying result due to special task properties. It's hard to strike a balance between counting and detection.
   - Since most of the patches contain no sea lions and the distribution of sea lions is skewed, dense data and severe data unbalanced.

- We applied patches constrain, mosaic preprocessing [6] and focal loss [7] aiming to solve these problems.
- Patches Constrain: We set the proportion between the positive block (at least one sea lion) and the negative block
- Focal Loss: use 1/freq as the loss weight to increase the weight of hard samples.
- Mosaic [10]: Mixcut to increase the data density.



II. Related works (Object Detection):

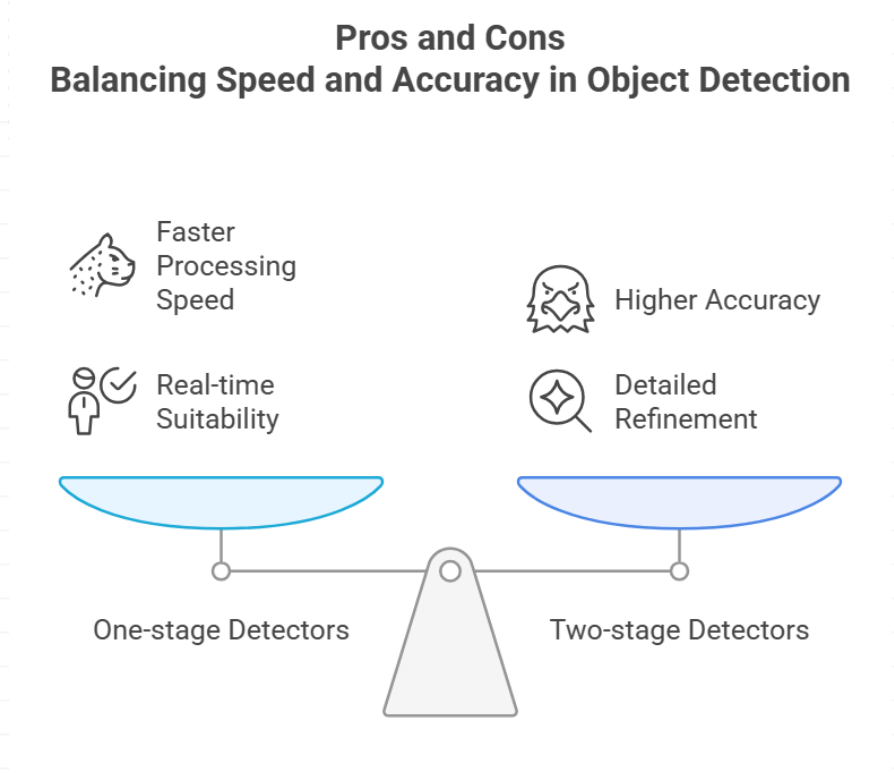
1. One-Staged method:

   Approach that directly predicts object locations and classes in a single network pass, and generates bounding box predictions and classifies them simultaneously across the image, eliminating the need for a separate proposal generation step.

   - Eg: YOLO (You only look at once) Family [2]
2. Two-Staged method:

   Approach that first generates region proposals, then refining and classifying these proposals, making it suitable for scenarios requiring high precision.

- Eg: Fast / Mask RCNN family [3][4]

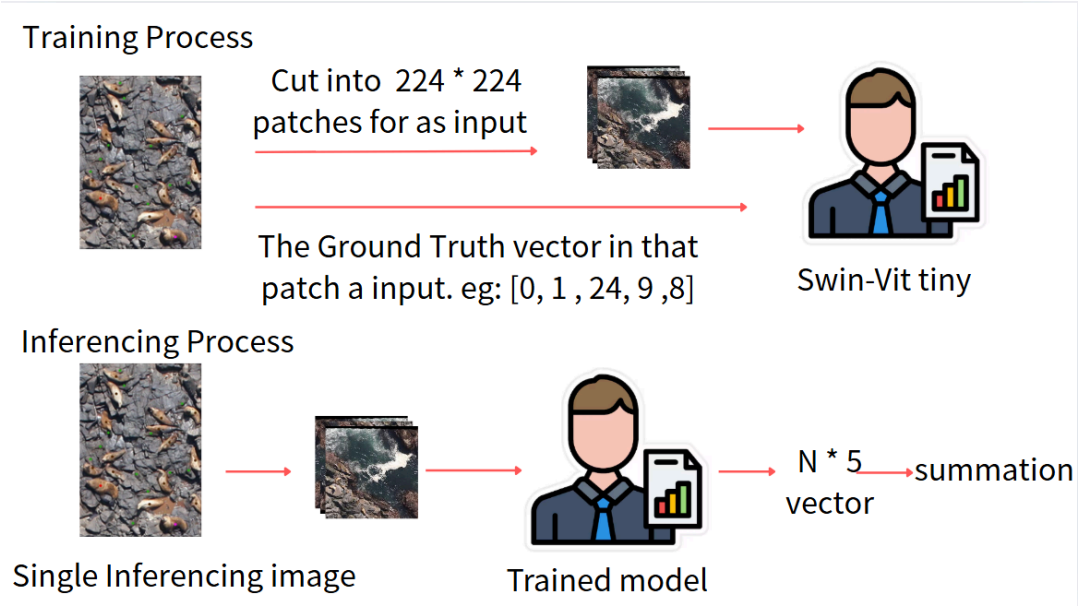


III. Method / Approach:

1. Swin Transformer + MLP Regression:

Inspired by the #1 [4], #4 [5] in the leaderboard, we tried the method that combining the feature extractor + regression to directly learn the relationship between the images and a 1 * 5 vector output representing the counts of 5 types of sea lions.

- Network Overview

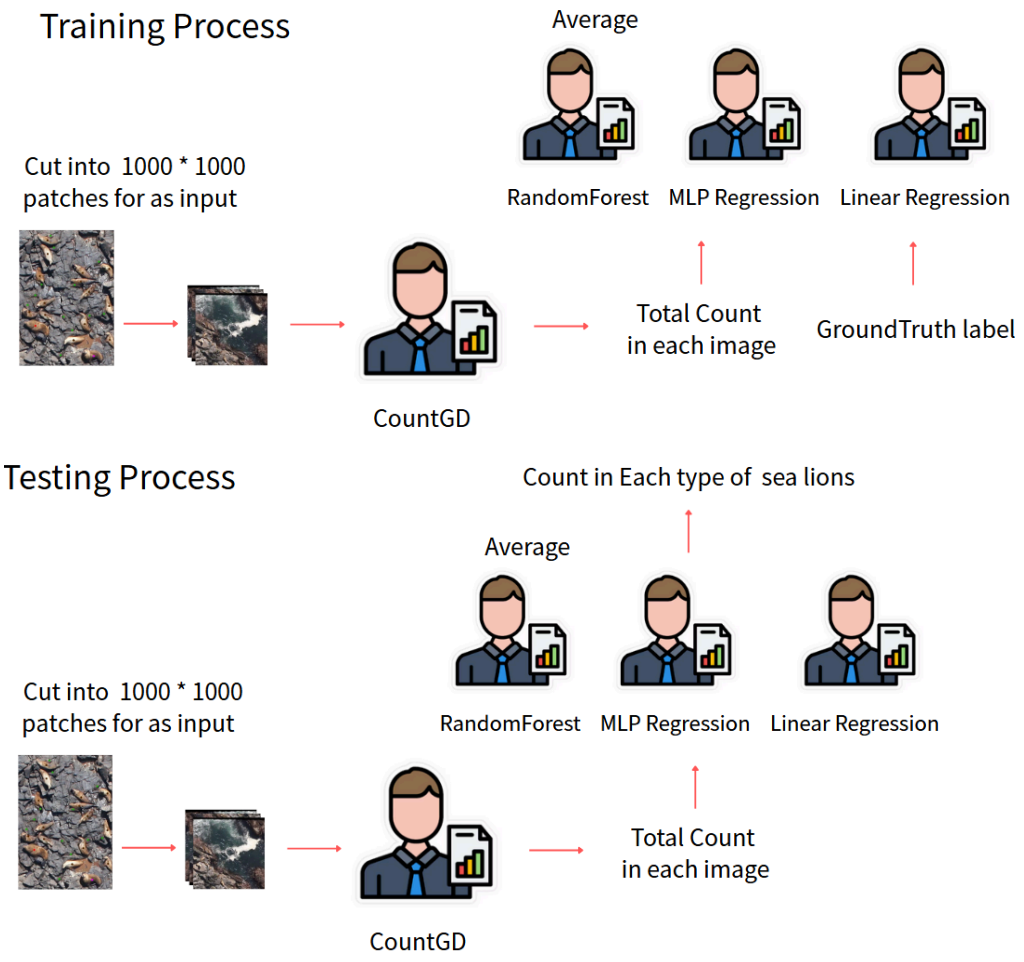


- Details of your approach
    i. Cut the Train and TrainDotted images into patches.
    ii. Use existing codes to detect dotted in those patches to generate patches ground truth label
    iii. Train the Swin Vision transformer[6] model with a regression layer in the end with patches and labels.
    iv. Cut the Test data into patches and predict with a trained model.

2. **Counting model + Voting (MLP Regression + Random Forest + Linear Regression)**

   After some surveys, we found that there's an existing relationship between each type of sea lion with the given number of sea lions As a result, we try this method to learn the relation

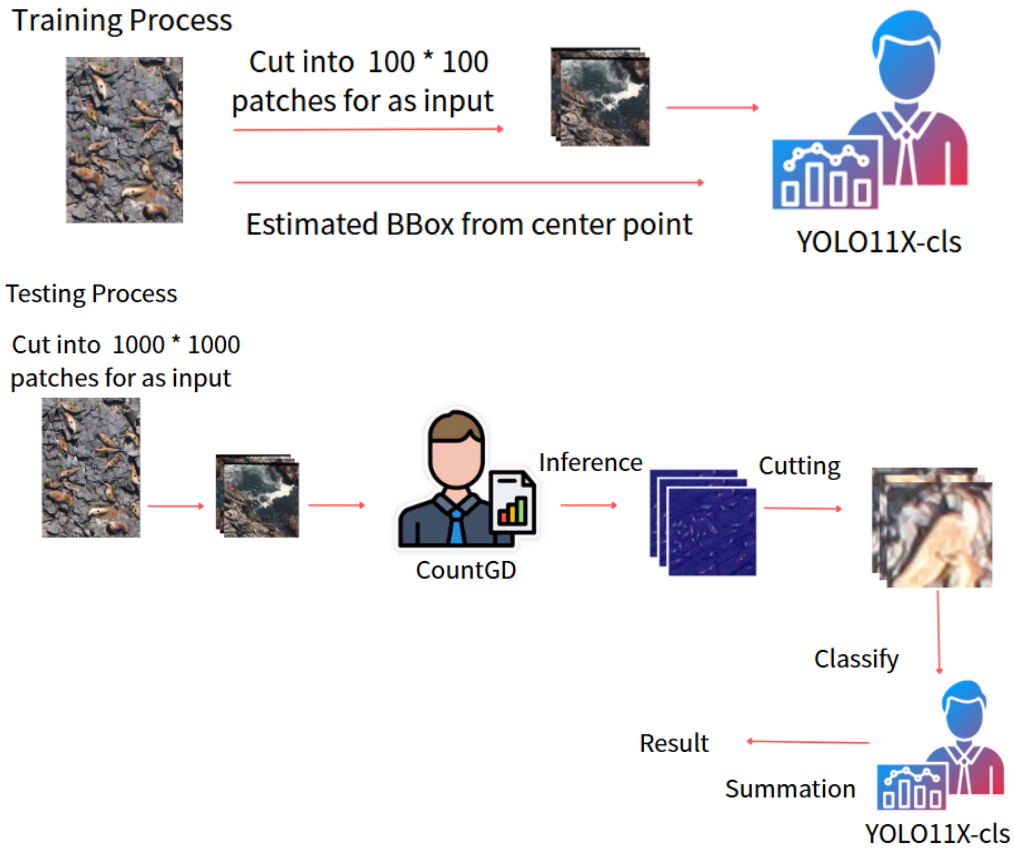
   - Network Overview



- Details of your approach
    i. We use countGD [8], which is a prompt-based model that can predict the heatmap center of objects and the total number of objects.
    ii. With the total sea lion number, we train multiple regression models and average their output to predict the number of each type of sea lion.
    iii. The testing process is similar to training with trained regressors.

3. Counting + Vision Transformer [9] / YOLO

- Network Overview



- Details of your approach
    i. Based on the colored dots in the Traindotted labels, we know each sea lion's class and center point. We then

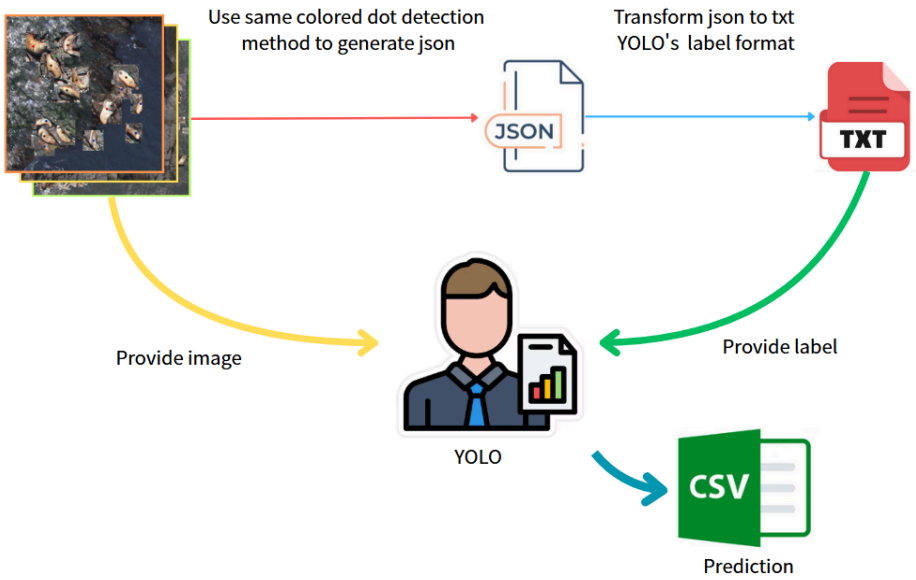
crop the same region from both the train and Traindotted images.

ii. Bounding box sizes are set based on the sea lion's class, and the cropped images are saved into class-specific folders — creating a dataset format compatible with YOLO.

iii. During testing, we use CountGD to scan each test image in 1000×1000 patches. It outputs the total number and center positions of sea lions in each patch. We then crop fixed-size bounding boxes around those centers and feed them into our trained YOLO/Vit model.

4. Pure YOLO detection

- Overview: Network figure



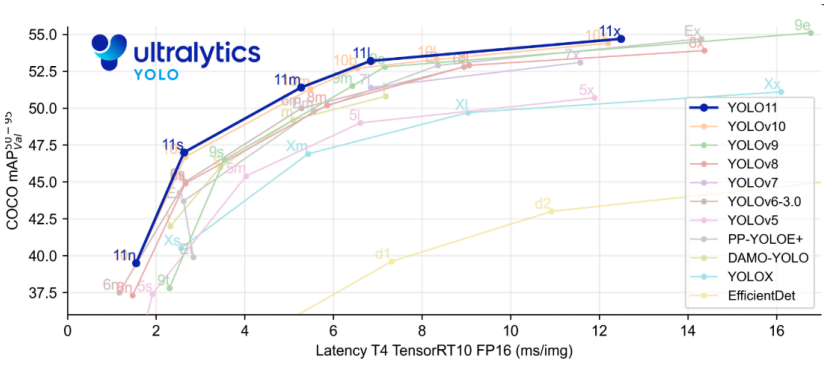
- Details of your approach

i. We use a color dot detection program on the generated mosaic images (Traindotted version) to count the number of sea lions of each class in every image within

each mosaic. This program generates a JSON file with the counts. Then, we use another program to convert the JSON into YOLO-compatible txt label format.

ii. Start training YOLO.

The model we selected:



Performance

Detection (COCO) | Segmentation (COCO) | Classification (ImageNet) | Pose (COCO) | OBB (DOTAv1)

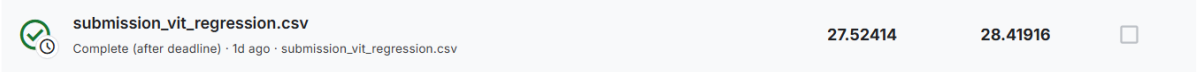
See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

We choose YOLO 11L, which offers strong performance on the COCO dataset (with mAP50=53.4) and maintains a practical inference time.

Results and Ablation study:

1. Swin Transformer + MLP Regression:

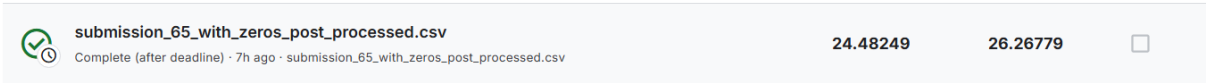


- With Mosaic: Testing: RMSE 27.52
- Without Mosaic: Testing: RMSE 28.70

- Implication:
  - The result is only slightly better than all zero prediction, which means we fail to reproduce the #1 solution.
  - We found that the size differs a lot in training data and testing data, making the val RMSE less reliable.
  - Mosaic preprocessing may the reduce too much information between the environment and the sea lions, making the model

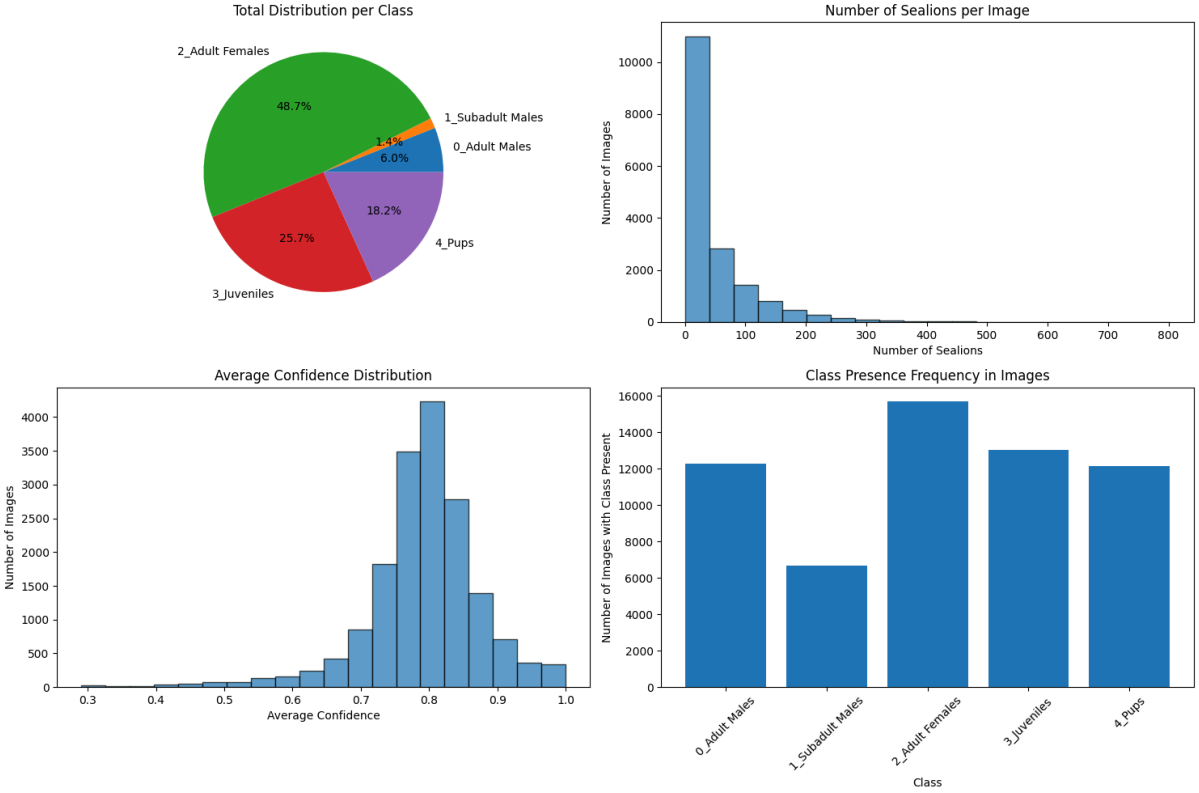
2. **Counting model + Average Regression (MLP Regression + Random Forest + Linear Regression)**
   a. RMSE: 20.40

3. **Counting + Vision Transformer [9] / YOLO**
   a. Counting + Vit: RMSE 24.48



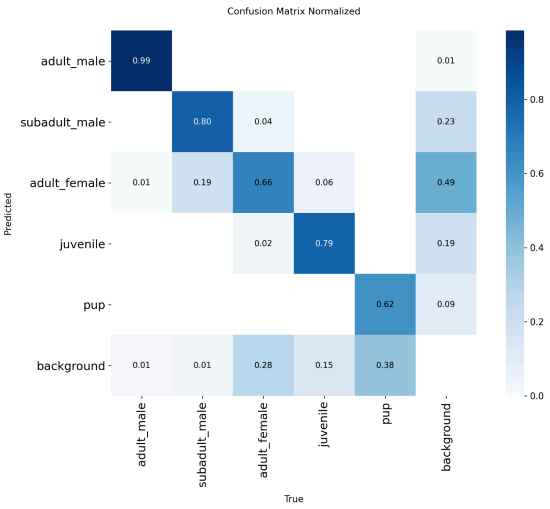
   b. Counting + YOLO: RMSE 20.42

4. **Pure YOLO 11L Detection**
   a. Dataset(s) and metric(s) for evaluation
      i. Total data sample numbers on all mosaic images per class



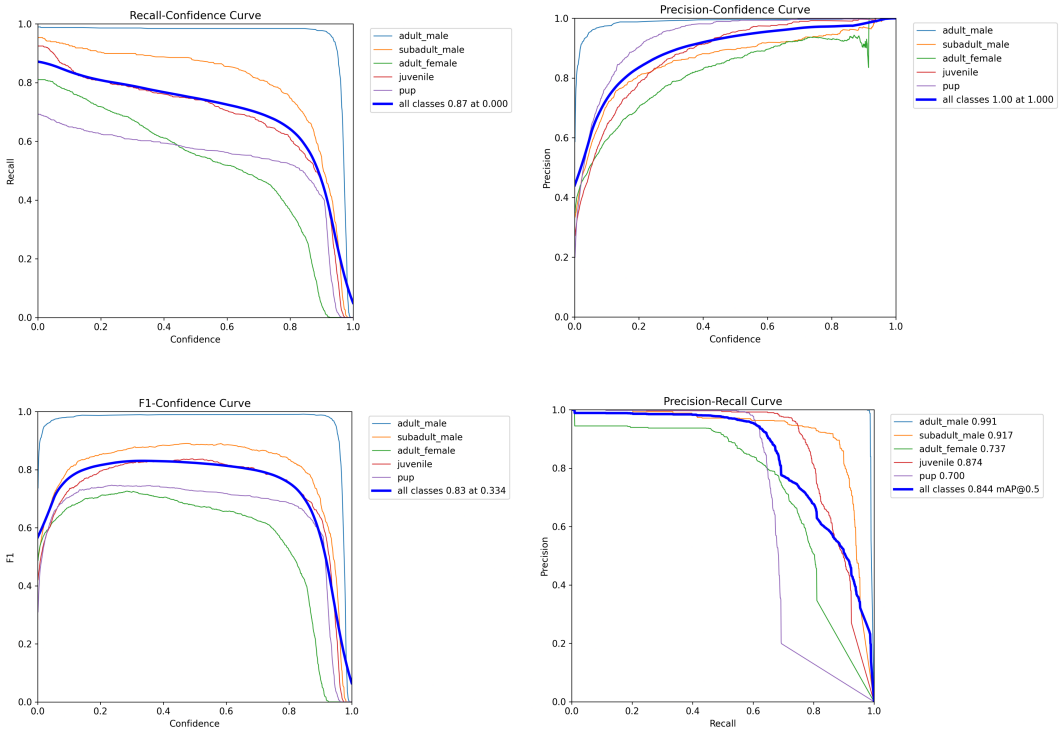
We do data balance on our mosaic images. Then, we get the sample numbers by colored dot mosaic images to get every class' numbers.

ii. Metrics:



Our model's final performance on the validation set demonstrates effective learning of inter-class differences, with minimal misclassification between

categories. However, it struggles significantly with distinguishing between background and sea lions, leading to a high number of false positives (FP) and false negatives (FN), despite correctly detected sea lions being well-classified.



- The precision and recall curves indicate overall decent performance. Adult males, despite having fewer samples, are classified best, maintaining high precision (low misclassification from other classes) and high recall (low misclassification into other classes) across various confidence thresholds. The F1-confidence curve shows stable classification ability in non-extreme confidence intervals, peaking at an F1 score of 0.83 at a confidence of 0.334. The precision-recall curve highlights the balance between precision and recall, with pups showing the most imbalance—decent precision but very poor recall, suggesting the

model is overly strict for this class. Adult females also require attention, exhibiting suboptimal precision and recall despite being the most abundant class, indicating the model may have the least clear understanding of this category.

Notes:

Mosaic introduction:

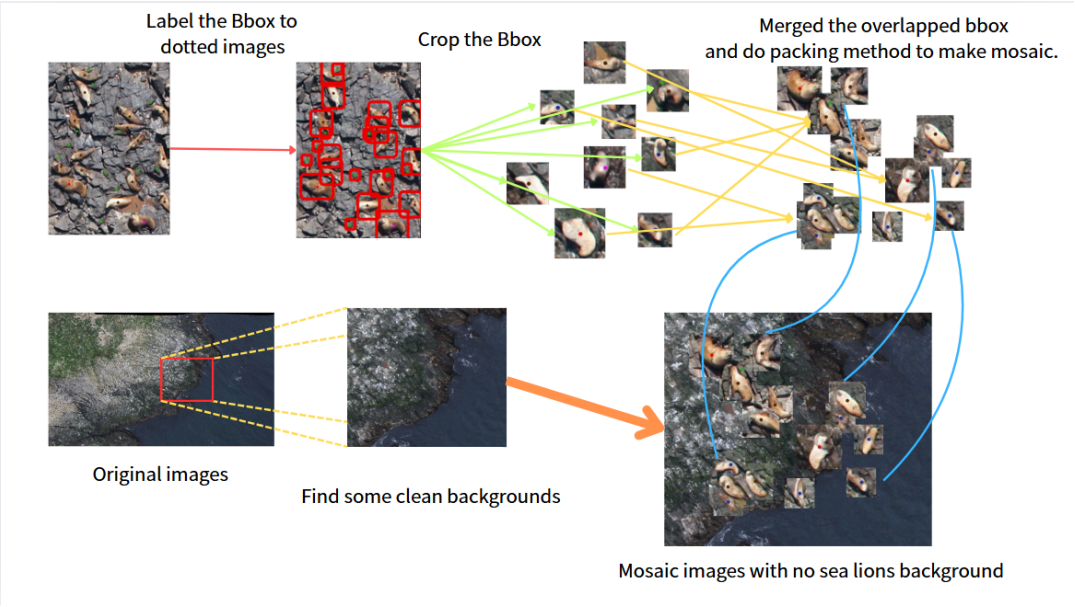
Why We Used This Method

This approach is inspired by the mosaic method proposed in the GitHub link referenced in the Related Work section. It successfully tackles challenges such as small and crowded objects, as well as slow inference speeds caused by the high background ratio in original images.

Details of your approach

Procedure:

1. We start by extracting JSON files from the Traindotted images.



These JSON files contain bounding box (bbox) information for each image, serving as ground truth for training. We establish minimum and maximum thresholds for the number of bboxes per sea lion class. However, the dataset is imbalanced, with adult

males and subadult males comprising only a small portion of the samples. To address this, we triple the bboxes for these two classes.

Before we do data balance:

5392	4345	37537	20118	16285

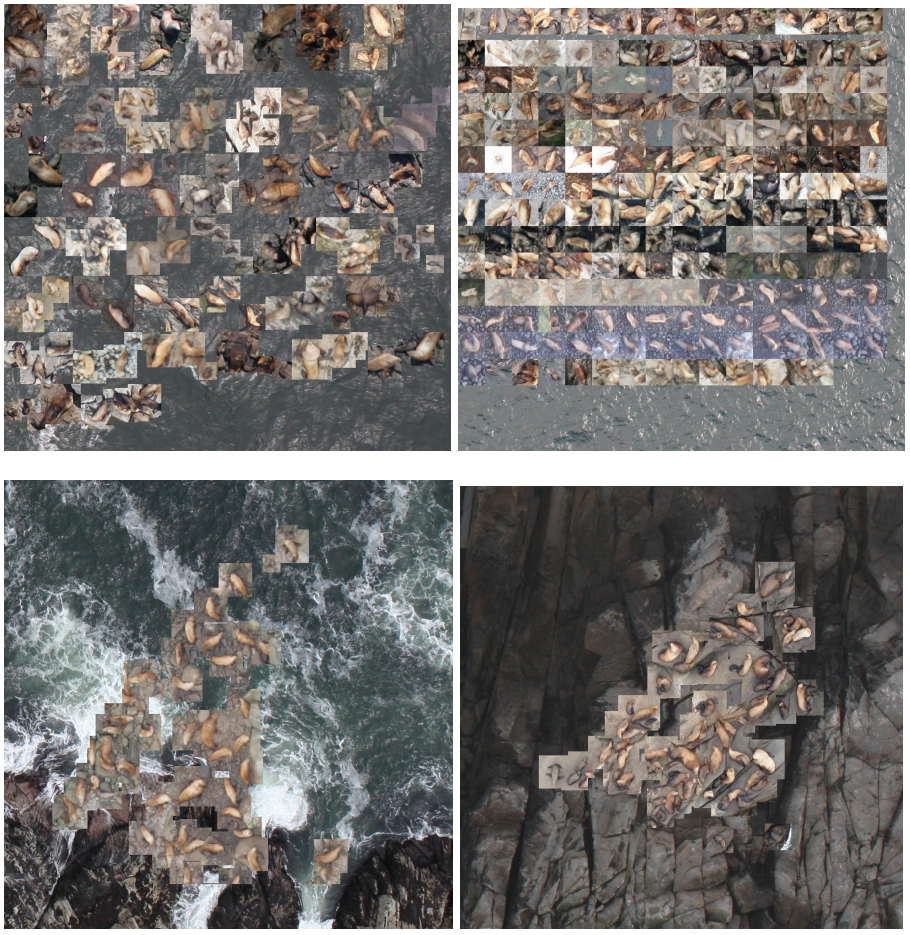
After we do data balance:

5307	4193	10509	6589	5646

2. With the bbox information ready, we proceed to mosaic generation. First, we use the JSON file to identify overlapping bboxes in the original image. If overlaps are detected, we merge them into a larger bbox that encompasses the union of the overlapping regions, repeating this process until no overlaps remain in the image.
3. Next, if any extraordinarily large merged bboxes are created, we crop them into multiple parts. The crop size is determined based on the desired mosaic image size (typically slightly smaller than the mosaic size).
4. At this stage, we have all the "puzzle pieces" (merged bboxes) for forming mosaic images, but we lack backgrounds to place them on. We generate backgrounds using the JSON file by selecting regions as large as the mosaic images that contain zero sea lions.
5. Finally, we can start to pack these merged bboxes to mosaic images. Here is our strategy, We first sort all merged bboxes by area, from largest to smallest. If a merged bbox is too large, we assign it its own mosaic image. For a very large merged bbox that exceeds our threshold, we allocate a mosaic image to it first, then use smaller merged bboxes to fill the gaps as much as possible. To estimate if the current mosaic image can fit the next merged

bbox, we use area statistics: if it exceeds 80% of the mosaic image's area, we stop generating that mosaic image and start the next one.

Some mosaic images:



Conclusion:

In this task, although we attempted to reproduce several existing methods—including various classification and regression approaches—we were unable to achieve satisfying results. We also tried to replicate the techniques used by top-ranking teams, but the performance remained below expectations. We suspect this may be due to a significant domain gap between the training and test datasets. Upon reviewing discussions on the Kaggle forum, we found that some participants noted the importance of sea lion body size as a key feature for classification. In the test set, the size variation across categories was

substantial—for example, juveniles in one image could be larger than adult females in another. Moreover, we later realized that many teams applied post-processing techniques to adjust their predictions and minimize RMSE, with parameters fine-tuned through repeated submissions. In hindsight, we believe we should have incorporated this consideration earlier by introducing scale-aware training strategies to make our model more robust to intra-class size variations of the same sea lion category.

Team member contribution:

Tasks	**contributors(%)**
Literature survey	110612117 (25%), 110652032 (25%), 110550128 (25%), 109612019 (25%)
Approach design	110612117 (25%), 110652032 (25%), 110550128 (25%), 109612019 (25%)
Approach implementation	110612117 (25%), 110652032 (25%), 110550128 (25%), 109612019 (25%)
Report writing	110612117 (25%), 110652032 (25%), 110550128 (25%), 109612019 (25%)
Slide making and oral presentation	110612117 (25%), 110652032 (25%), 110550128 (25%), 109612019 (25%)

Reference:

1. **kaggle #1 solution:** Use keras to count Sea Lions
2. **YOLO (You Only Look Once: Unified, Real-Time Object Detection)**
   Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of

the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788.

3. **Fast R-CNN (Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks)**
   Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1440-1448.
4. **Mask R-CNN**
   He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2961-2969.
5. kaggle #4 solution
   https://www.kaggle.com/competitions/noaa-fisheries-steller-sea-lion-population-count/discussion/35442
6. **Swin Transformer (Hierarchical Vision Transformer using Shifted Windows)**
   Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10012–10022. https://doi.org/10.1109/ICCV48922.2021.00988
7. **Focal Loss (Focal Loss for Dense Object Detection)**
   Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2980-2988.
8. **CountGD (Counting Grid Detection):**
   https://github.com/niki-amini-naieni/CountGD
9. **Vision Transformer (An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale)**
   Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai,

X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations (ICLR).*

10. **Mosaic:**

    The official implementation of UFPMP-Det

11. **Random Forest**

    Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324