Improving Camouflage Object Detection

Vinay Awasthi,
SCPD Stanford University
Baltimore, MD
v365747@stanford.edu

Yin-Li Liu
Department of Electrical Engineering
Stanford, CA
liuyinli@stanford.edu

Max Meyberg
Department of Computer Science University
Stanford, CA
mmeyberg@stanford.edu

Abstract

This paper outlines an approach to identifying camouflaged objects of different shapes in complete harmony with their surrounding. The YOLOv8 algorithm operates by extracting features and applying non-maximum suppression to detect overlapping bounding boxes. On the COD10K dataset, YOLOv8 achieved a mean average precision (mAP) of 18.2% in our training dataset. The CAMO dataset, converted to YOLO1.1 format using CVAT.AI, also showed poor training performance with a mean precision (mAP50) of 3.89%, which we believe is due to issues with identifying the center in our bounding boxes for ground truth. We are working on addressing this issue. Using these two datasets, we explored different approaches to improve performance, including edge detection with Fourier transform, wavelet transforms, shape separation, and transfer learning. We achieved over 50% mAP50 by continuing to train the entire YOLOv8 small model with the COD10K and CAMO-COCO datasets, and over 40% mAP50 by performing transfer learning on the YOLOv8 nano model.

1. Introduction

1.1. Introduction

In order to detect camouflaged objects, it is necessary to separate these objects from their surroundings through edge detection. For example, camouflaged objects behind tree stems, under grass, or underwater require separating shape contours to identify specific shapes. Currently, we have narrowed our data augmentation choices to processing images using CVNN [2], with frequency filters such as high-pass filters and wavelet transforms. This approach will make our neural network more complex by transitioning from real numbers to **complex-valued numbers**, as a result creating 4x the total parameters. PyTorch (version 1.7) has added support for complex-valued designs by allowing `torch.complex64`, and TensorFlow also supports many libraries that facilitate designs with complex-valued numbers.

We expect that complex-valued transformations in the form of upstream CNN layers will help suppress the backgrounds of images sufficiently for YOLOv8 to properly detect the structural characteristics of camouflaged objects, even when they match the texture and color of their surroundings.

We are approaching camouflaged object detection as a problem of edge detection and shape separation. The YOLOv8 architecture weakens edge features as information passes through various layers, so we need to augment data to address this issue. Shape identification by detecting contours of camouflaged objects, by working in the CVNN domain (Complex Value Neural Nets), can further assist us in identifying occluded structures.

1.2. Weaknesses of other Research

One of the biggest challenges faced in detecting camouflaged objects is in regards for practical applications for image section techniques, specifically for object detection tasks focused on camouflaged objects. The issue arises in the fact that since true labels for camouflage objects use bounding boxes, this results in an excessive amount of redundant background information which acts as "noise". As a result, researchers have faced challenges in further improving camouflage detection by utilizing YOLO v8.

An additional challenge includes the loss of texture information when reducing image dimensions.

1.3. Related Work

YOLOv8 with augmented CNNs is currently detecting about 70 percent of the objects in COD10K dataset [1]. Anabranch[4] network for camouflaged object detection [4], reported 66 percent correct identification (segmentation only) as these objects are large, and at time cover, more that 70 percent of picture (Stingray submerged under sandy ocean, large coiled python over dried leaves etc..), YOLOv8 in general, will find these dataset challenging, due to its limitation of not carrying edge features deep into its neural net layers.

2. Dataset

2.1. Data Pre-processing

We used CVAT.ai and scripts to convert data from existing format to YOLO 1.1 format carrying class, center coordinates along with width and height values. We ran into an issue of not identifying center of the bounding box carrying camouflaged object using script so our training accuracy is lower than expected. We are working on correcting this issue.

CVAT.AI can process data in many formats, however this process of creating custom dataset that can be processed using YOLOv8 for object detection is a manual process. There are services that can draw bounding boxes on our 1250 CAMO dataset images but we decided not use them as we need to also get familiar with data, various shapes, lighting conditions, various occlusions of camouflage in nature so that we can properly augment our data and deploy correct approach to separate object and its shape.

We tried roboflow which was unable to tag camouflaged dataset using foundational model "grounding DINO". It did however tag non camouflaged objects correctly.

We ended up going with `https://github.com/SYED-M-HUSSAIN/COD/` data conversion scripts which use *imutils* python package to draw bounding box using segmentation mask.

3. Methodology

3.1. Model Selection

Many current state of the art methods (Sparse R-CNN, Anabranch, YOLOv8, Vision Transformers using mask separable attention etc...) find it challenging to detect occluded camouflaged objects due to not being able to detect edges as at times these objects are large and windy taking almost entire image. We want to focus on increasing precision so we need to get to know our dataset well (COD10K `https://dengpingfan.github.io/pages/COD.html`, CAMO `https://sites.google.com/view/ltnghia/research/camo`).

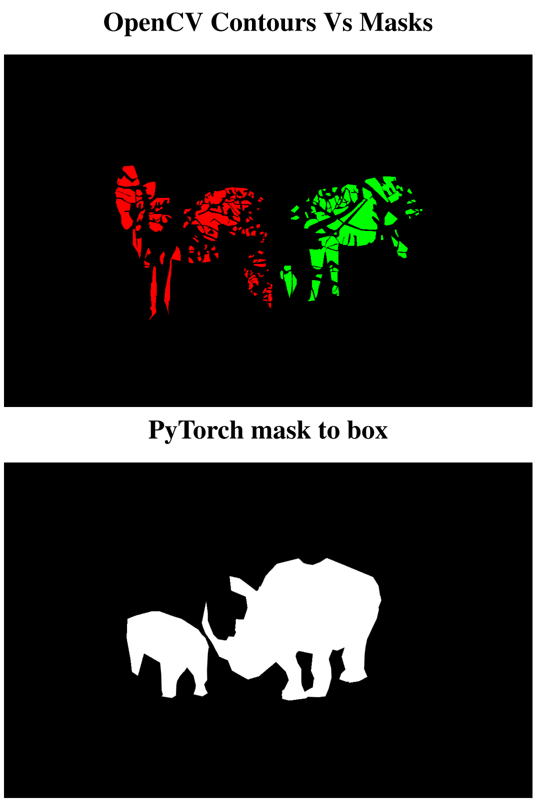


Figure 1. Highlighting failing cases of various bbox scripting based methods.

Full survey of various concealed object detection methods can be found here. `https://github.com/ChunmingHe/awesome-concealed-object-segmentation`

3.2. YOLOv8 Baseline

Pre-trained YOLOv8 model with CAMO dataset, did not perform well, giving only about 3.96% on mAP(50:95). We attribute this to:
1. Smaller model (YOLOv8 nano) not being able to capture occluded, camouflaged edges and shapes and
2. Training with dataset of 152 images.

3.3. YOLOv8 COCO Data Format alignment

We ran YOLOv8x (largest model) as is for 120 epochs with incorrect bounding box centers to get about 18.2 percentage average precision. We decided to use YOLOv8s and train on CAMO-COCO and COD dataset.

We first used OpenCV collect contour method, to draw bounding boxes, which did not draw correct bounding boxes for zebra and giraffe etc.. in COD10K dataset as it just picked contours, so each stripe on zebra became a camouflaged object thus bringing our training accuracy to single digits.

While debugging low success rates of YOLOv8s on

Pytorch correctly drawing bboxes

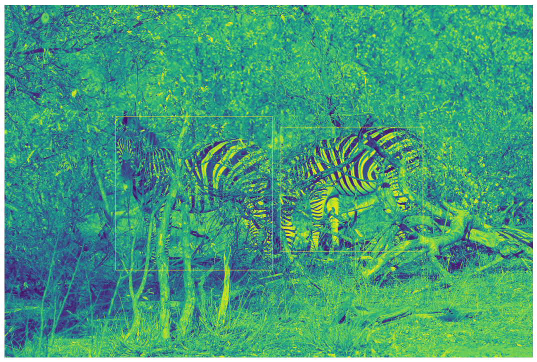


Figure 2. Correct Bounding boxes on Patchy objects.

Models	Size	Param.	Speed (ms)	mAP(50:95)
YOLOv8n	640	3.2 M	0.99	37.3
YOLOv8s	640	11.2 M	1.2	44.9
YOLOv8m	640	25.9 M	1.83	50.2
YOLOv8l	640	43.7 M	2.39	52.9
YOLOv8x	640	68.2 M	3.53	53.9

Table 1. YOLO Models Complexity and Performance

COD10K dataset, we noticed that some label files, carried lots of bounding boxes, and training so many images with many instances was taking up to 7.8 GB of GPU memory. We looked at the training dataset for images of patched objects, which were failing detection (Giraffe, Zebra, Shrimp). We noticed that OpenCV collect contour method, that worked well for CAMO-COCO dataset as objects were simpler (no stripes etc..), was failing to recognize entire objects as one. After switching to PyTorch's $torchvision.ops$ "mask to box" method, we started seeing correct bounding boxes which were then fed to clean run of YOLO pretrained model to learn camouflaged objects giving training precision in 50 percentage range (mAP50) and 25 percentage range for mAP50-95 metric just like CAMO-COCO dataset.

We picked YOLOv8s model as base so that we can run many experiments and still provide a solution of performing single shot object detection in field for rescue missions. We desire to not take upt to 40 seconds (R-CNNs) to detect an image with high precision as firefighters and first responders may use our model, to respond in rescue missions. We need to sacrifice accuracy for faster speed.

3.4. Loss Function

We have chosen to implement our loss function as collection of 3 losses just as it is done in ViT paper for detecting camouflaged objects[5]: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10207675`.

1. Classification Loss:
At present, we are classifying objects as camouflaged or not, by using just single class representation. As we further clean up datasets for YOLOv8 consumption; we will add fine-grained classification identifiers representing various object classes.

2. Objectness Loss:
This defines whether our neural net was able to separate out object from the background or not. This loss tells how far off our detection was from the ground truth bounding box. This single loss characterizes the quality of our edge detection and shape separation. This value represents quality of our texture pre-processing, using various CNN layers upfront performing high-pass filters or wavelet processing to separate-out edges and shape, of camouflaged object from its background.

3. Regression Loss:
This loss represents difference in predicted class vs true class. For single class identification this is identical to classification loss above. We intend to use this loss for multi-class identification. This loss will define whether we are correctly identifying whether we have camouflaged bird in picture or an insect or fish.

$$L = L_{classification} + L_{object} + L_{regression}$$

3.5. Training Budget

YOLO - A week of training results in 88% accuracy on non camouflaged objects.`https://arxiv.org/pdf/1506.02640`
ViT - 30 Days of training
Ours - 2 hrs/dataset (< 250 epochs on A100). This limit allows us to verify our model on 2 datasets, one with small objects (CAMO-COCO) and other with large more complex objects with occlusion, blurred edges and multiple instances with varying lighting conditions and reflections..

3.6. Metrics

We intend to add CNN layers to YOLO model and may transform some layers to transformer like architecture supplying some context about camouflage.

YOLOv8 reports F1 scores and mean average precision mAP50 and mAP(50:95), we intend to keep this metric for our classification.

$$F1_{score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

$$mAP = \frac{1}{c}\sum_{K=i}^{N} P(K).\Delta R(K)$$

In above equations, TP is True positive, FP is false positive, FN is false negative, c represents object categories, K represents Intersection over Union (IoU) threshold. P(K) is precision and R(K) is recall.

These metrics are standard in object detection and classification literature.

3.7. Transfer Learning

In addition to training the entire YOLO network, we employed transfer learning by freezing the weights in certain layers of the YOLO backbone model and updating the remaining layers using the CAMO dataset. This method offers an alternative approach for fine-tuning the YOLO model to detect occluded objects without the need for high computing power. We trained the model on a CPU for 50 epochs, which took 17 hours, using the COD10K CAMO dataset and achieved a mAP50 score of 0.411.

We froze 123 out of 184 layers (15 out of 22 in the modules view) in the YOLOv8 nano model, reducing the computational cost from 3,157,200 parameters to 1,420,880 parameters for gradient calculation. This made the training process feasible on CPU machines. The performance of the transfer learning model, as well as the fully-trained model, will be discussed in the results section.

4. Ensemble Learning using 3 models

In order to analyze texture information, we wanted to move to complex domain using Fourier Transforms. We considered scattering transform, which is defined as complex-valued convolutional neural network with wavelets acting as filters, allowing some signals to travel far in the network, while limiting others to shorter distances, limiting their influence. We considered this to be a good plan as there is lot of literature that suggest that texture processing can be done using FFTs. Wavelets CNNs can be seen as... Source :`https://www.di.ens.fr/data/publications/papers/1304.6763v1.pdf`

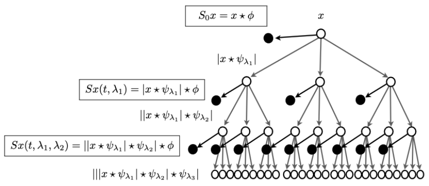

$S_0x = x \star \phi$
x
$|x \star \psi_{\lambda_1}|$
$Sx(t, \lambda_1) = |x \star \psi_{\lambda_1}| \star \phi$
$||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|$
$Sx(t, \lambda_1, \lambda_2) = ||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}| \star \phi$
$|||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}| \star \psi_{\lambda_3}|$

Figure 3. Wavelets traversing CNNs, source: https://www.di.ens.fr/data/publications/papers/1304.6763v1.pdf

Pytorch has support for complex valued tensors and YOLOv5 has a port supporting complex valued CNNs.

We investigated use of direct cosine transforms for camouflage detection as outlined in `https://openaccess.thecvf.com/content/CVPR2022/papers/Zhong_Detecting_Camouflaged_Object_in_Frequency_Domain_CVPR_2022_paper.pdf`

These approaches used modified loss function, taking discrete cosine transform (DCT) based norms, instead of euclidean norms, to detect, whether given image tile, was similar or different, to other tiles, thus detecting camouflaged object. These modifications also required changes to the way dot products were implemented, changes to softmax function and necessity to train entire network again, as pre-trained weights from YOLOv8s model, wouldn't have been directly applicable. We did not have 7 days of training budget so we decided to implement best of both worlds, by taking inspiration from space image data processing, which discarded imaginary part, after taking FFTs, and just used real part of the complex number, to continue image refinements.

These approaches used modified loss functions, incorporating discrete cosine transform (DCT) based norms instead of Euclidean norms, to determine whether a given image tile was similar to or different from other tiles, thereby detecting camouflaged objects. These modifications also need to change the implementation from dot products to softmax function, which required retraining the entire network, as the pre-trained weights from the YOLOv8s model would not have been directly applicable.

Given our limited training budget of seven days, we decided to implement the best of both worlds by taking inspiration from space image data processing. Specifically, we discarded the imaginary part after taking FFTs and continued image refinements using only the real part of the complex numbers.

We created 3 YOLOv8s models so that we can perform ensemble learning by combining direct, edge enhanced and shape enhanced (texture processed) detection while keeping processing requirements low so that real time detection could still take place. One model ran in **base mode**, using dataset as is. We also passed background images without any labels so that model can generalize better in aquatic scenes to detect submerged turtles.

One model, specializing in **edge detection**, used high pass filter and other specializing in shape detection, by separating out high and low spatial frequencies, in Fourier domain, as it is done in processing data from **space telescopes** `https://arxiv.org/pdf/1504.00647`[3]. We were intending to use these filters as convolutions, in complex domain, and run entire YOLO architecture, in complex domain, however we noticed, that by just processing real part of numbers, was enough, to increase mAP50 and mAP(50:95) scores, by 10 percentage points, in shape related wavelet transforms using tuned hyper-parameters. We used YOLOv8s and YOLOv8n models which are lim-

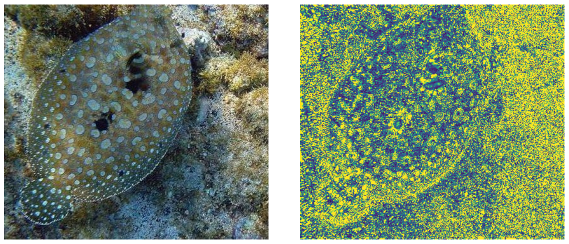


Table 2. High Pass filter transform, extracting edges, Original image is on left, extracted edges are on the right.

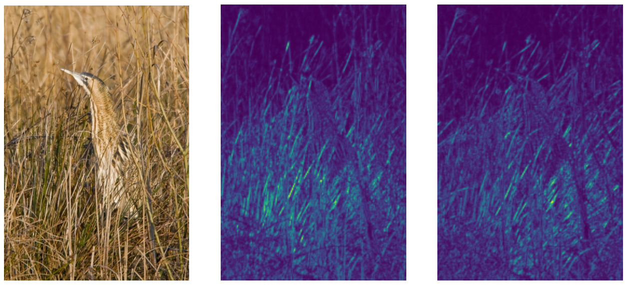


Table 3. Scatter2D wavelet differentiable transforms. Original heron image (L), Scatter2D transform with scatter scattering coefficient=1 (M), Scatter2D transform with scatter scattering coefficient=2 (R)

ited in their power to express complex features.

Third model tried 2 approaches. We first tried Kymatio wavelet scattering net `https://www.kymat.io/`, which is differentiable so these layers can be added to general pipeline of YOLO. This generated 2 sets of Scattering2D coefficients to detect object's shape better (see Table 2. above). We realized that these wavelet transforms were very expensive to run on CPU, taking 20 to 30 seconds/image on a 56 cores machine. We also tried to run it on GPU, but $CuPy$ package needed for processing on GPU destabilized the system, twice, triggering complete data-loss, as CUDA versions did not align between 11.8 and 12.2 of PyTorch, $CuPy$. We then abandoned this approach for wavelet Scattering2D. We want camouflaged object detection to happen in real time to allow for firefighters etc to have this setup in head up display, so that they can find people quickly in case of smoke/under-water or in other visually difficult situations.

We then embarked on extracting shape information using foreground/background processing, using Ftbg algorithm `https://ui.adsabs.harvard.edu/abs/2017ascl.soft11003W/abstract`. We converted the package to handle images in .jpg and .png format as it only handled images in FITS format as most of the data from space telescopes comes in, in this format or processed in this format.

We produced predictions in all 3 models separately

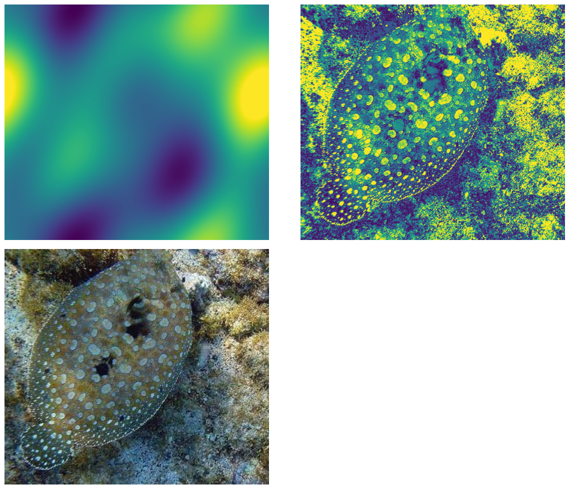


Table 4. Fourier Transforms isolating foreground (structure), by separating high spatial signals from low ones. First row shows background and foreground, 2nd row is showing original image

and then looked at failing cases in base models, to check whether FFT/wavelet processing could have helped in further detection, from other two models. We found out that in some cases, shape detection produced camouflaged object detection, where other two methods have failed.

We also found that edge detection triggered camouflaged object detection with low IOU (i.e. bounding box was away from the object).

We also noticed that shape based detection, increase our mAP50 to 0.56 and mAP(50:95) to 0.30 from earlier mAP50 of 0.5 and mAP(50:95) of 0.25.

We intend to combine all the models to create one final prediction as we noticed that each model has its own strengths in isolating camouflaged objects where other two might have failed.

5. A comparison of YOLO Architectures

YOLOv1 - Single CNN to perform real time object detection, by dividing image into GRID, making multiple predictions about existence of an object and then picking one with highest probability after removing overlapping boxes.

YOLOv2 - Darknet-19 backbone, used anchor boxes idea from Faster R-CNN and batch norm in all its convolution layers, increasing prediction accuracy.

YOLOv3 - Darknet 53 backbone, utilizing residual connections, up-sampling and logistic classifiers achieving 3x speed with comparable accuracy to RetinaNet.

YOLOv4 - This YOLO deployed spatial attention.

YOLOR - Added 2 pipelines to provide explicit knowledge and implicit knowledge to Discriminator thus adding capability to handle multiple tasks.

YOLOX - Deployed Anchor Free, CIOU loss giving

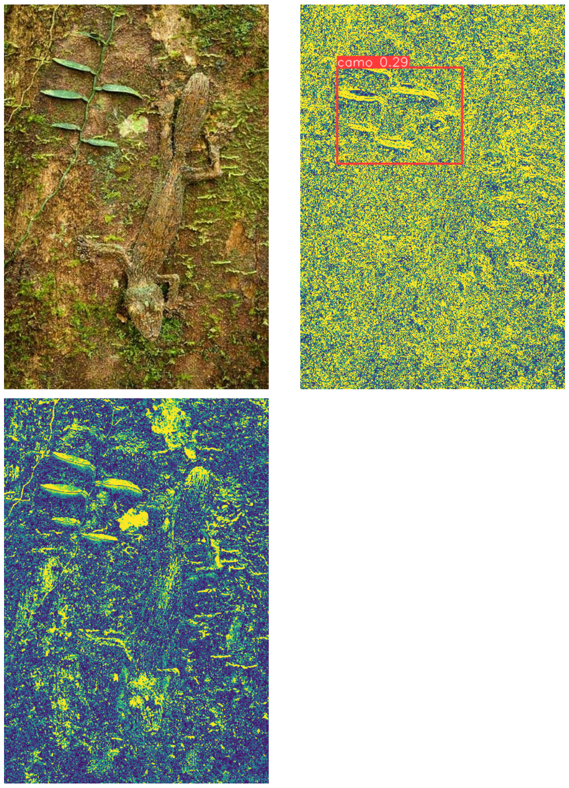


Table 5. Edge enhanced CAMO object detection, (2nd row) shape based detection is failing.

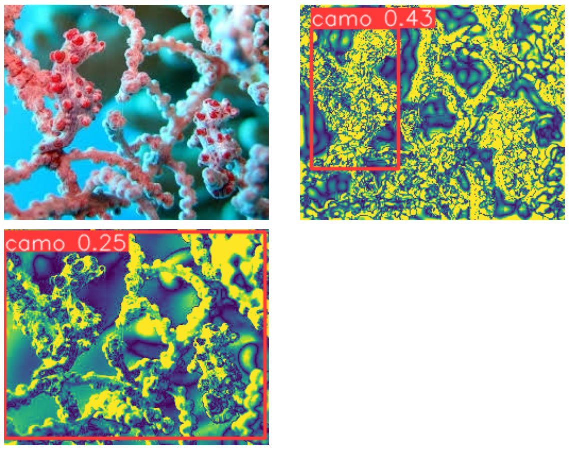


Table 6. Ensemble Model detecting camouflaged object, Model that did not use fourier transforms failed to detect camouflage object, other two models (Edge enhanced, Shape enhanced) were able to detect. Edge enhanced did best

moderate increase in precision. CIOU loss not only penalizes incorrect bounding box co-ordinates but also considers the aspect ratio and center distance of the box.

YOLOv6 - Deployed DIOU loss. Distance IOU loss incorporates normalized distance between the predicted box

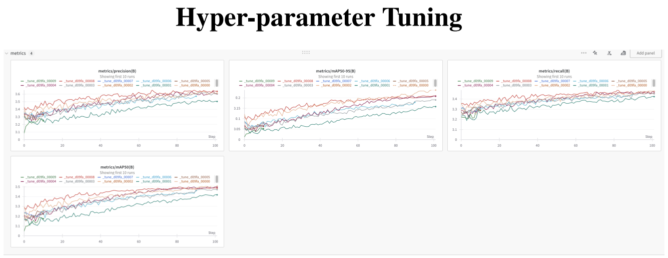


Figure 4. Ray-tune and Weights and Bias based hyper parameter tuning for 100 epochs

and the target box which converges much faster than generalized IOU (GIOU) loss.

YOLOv7 - Added Leaky ReLU activation and CSPDartnet-Z Backbone. CSPNet partitions the feature map of base layer into two parts and then merges them through a cross stage hierarchy. Use of split and merge allows for more gradient to flow through the network, giving 1% increase in mAP.

YOLOv8 - This adds Exponential Linear Unit (ELU) for activation, multi-scale object detection and new backbone architecture called CSPDarkNet + C2F module which combines high level features with contextual information. ELU is added to address vanishing gradient problem using CIOU and DFL (Binary cross entropy for classification loss) loss (over Generalized Intersection over Union loss `https://giou.stanford.edu/`) addressing how close shapes are to each other (i.e if no intersection, so intersection over union (IOU) is not helpful), central point difference between predicted box and ground truth and aspect ratio difference in predicted box to ground truth box. `https://encord.com/blog/yolo-object-detection-guide/`, `https://arxiv.org/html/2304.00501v6`.

6. Hyper Parameter Tuning

We used Ray-Tune and Weights and Biases to fine tune our model. Ray-tune performed a grid search, on 30 or so available YOLO hyper parameters, resulting in most optimal set after running for 100 epochs. These runs are very costly as they run for typically 6-8 hours on NVIDIA Dual A100 GPUs.

We then used these best hyper-parameters to further train/tune our shape and base models.

7. Results/Evaluation

We trained YOLOv8s (pretrained) using CAMO dataset for 289 epochs and then used this trained model on COD10K dataset for 500 epochs.

We have been experimenting with various features on model performance to fine-tune the best approach.

Dataset	Images Val	Box(P	R	mAP50	mAP(50-95)
CAMO	250	0.58	0.468	0.473	0.211
COD10K	2026	0.655	0.481	0.52	0.255

Table 7. CAMO-COCO and COD10K Dataset Validation-Set Results

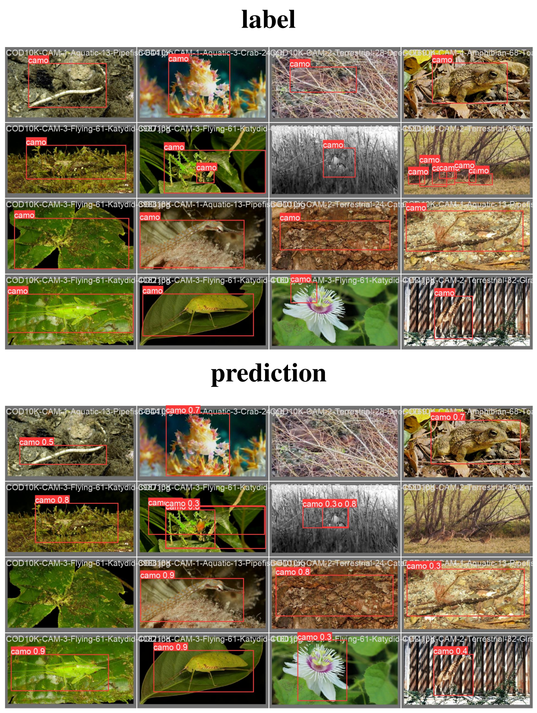


Figure 5. base line model prediction example after training for 500 epochs

We achieved 50 % accuracy (mAP50) on both of these datasets.

We trained on CAMO-COCO dataset for 289 epochs and then trained it on COD10K for 500 epochs. This got our validation set accuracy mAP50 to 50 %and mAP(50-95) to 25 %. We are getting almost identical results on both datasets even if YOLO model does not work well with large objects, as bounding box covers entire image.

Ensemble Learning by Feature Fusion
We are combining weights from various models before making prediction by giving equal say, to each set of weights. YOLO allows for this integration by exposing APIs to feature fusion from various models. `https://github.com/ultralytics/yolov5/issues/7905`.

Transfer Learning
For the transfer learning model, we trained on the COD10K dataset for 50 epochs using a pretrained YOLO nano model, achieving a mean precision of 41% (mAP50) on the validation set. Although this is slightly lower than the model trained on all layers, it significantly improves the performance of the original pretrained model. Considering computational cost, model size, training data size, and training time, the results demonstrate that performing transfer learning on a pretrained YOLO model is a viable approach under the constraints of limited devices, data, and time.

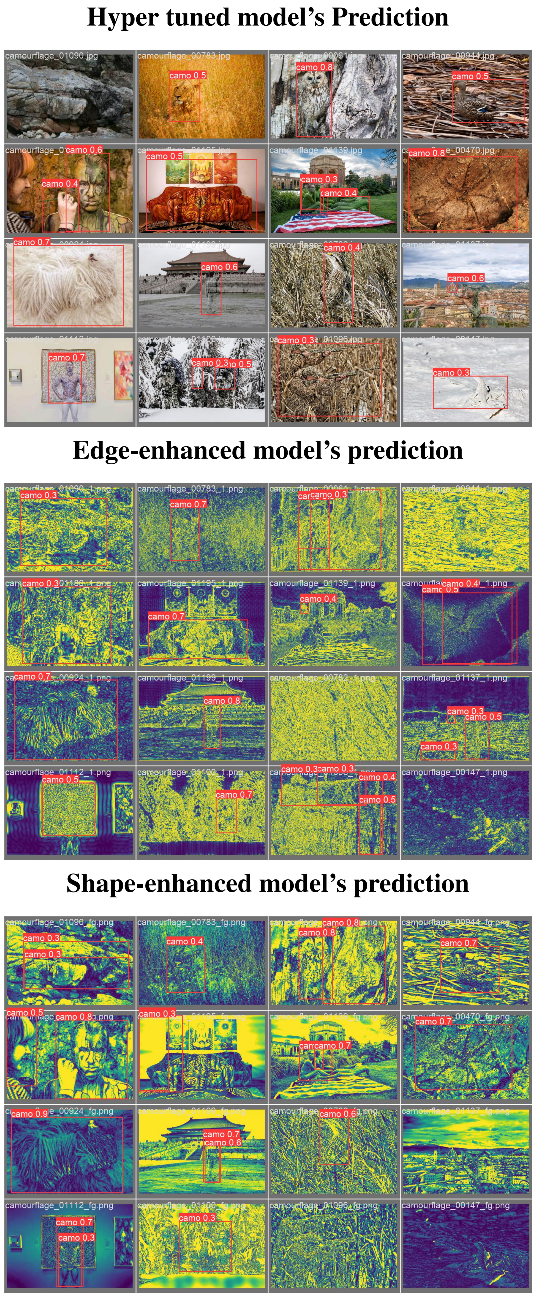

Hyper tuned model's Prediction
Edge-enhanced model's prediction
Shape-enhanced model's prediction

Figure 6. Ensemble Model for Feature fusion based prediction

Model	mAP50	mAP(50:95)	#p(M)
Pretrained YOLOv8s	0.0396	0.076	11.2
Finetuned YOLOv8s	0.520	0.655	11.2
Transfer learning	0.411	0.180	3.2
Edge-enhanced	0.441	0.232	11.2
Shape-enhanced-Hyper	0.556	0.309	11.2

Table 8. Performance on different models, CAMO and COD10K datasets

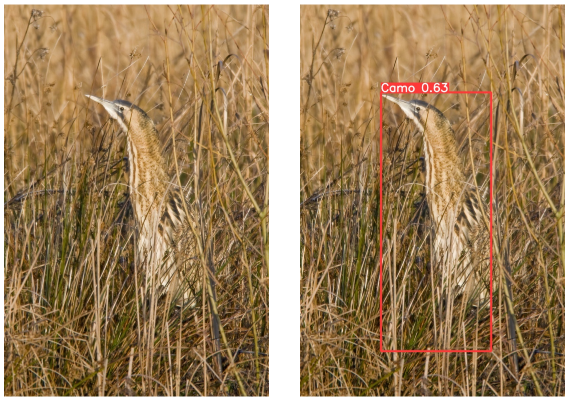


Table 9. Test image of hidden heron and prediction

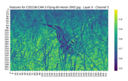
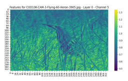
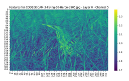
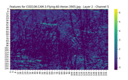
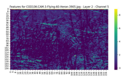
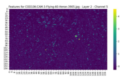
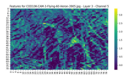
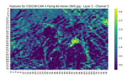
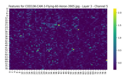
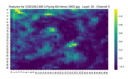
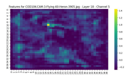
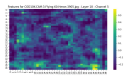
Layer	YOLOv8n	Fine-tuned	Transfer
0			
2			
3			
18			

Table 10. Feature Map Visualization

7.1. Feature Visualization

To gain a deeper understanding of how object detection works for camouflaged objects, we implemented feature map visualization on the convolution layers in YOLO models.

The backbone network of the YOLOv8 nano and small models consists of 22 convolution layers. We sampled feature maps from the output of several layers to compare the differences between the models.

8. Failing cases

YOLO does not do well when **multiple objects are clustered together**. This issue can only be fixed

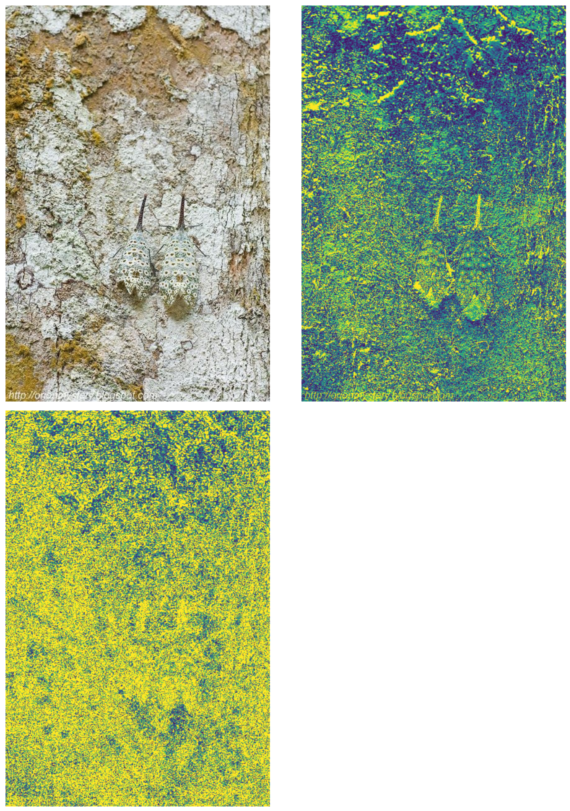


Table 11. YOLO fails to detect multiple objects.

by applying multiple dense layers at the start of neural net to extract relevant features as reported by SINet`https://github.com/DengPingFan/SINet?tab=readme-ov-file`.

9. Conclusion

We think that we have an easy to implement solution for head-up display that can detect occluded, camouflaged objects by performing various cost effective transformations, that can scale from environment to environment for various rescue missions. We noticed that wavelet transforms, used in cleaning out space telescope images of milky-way, can directly be applied to identifying, camouflaged objects, without incurring a lot of computational overhead involved in Scatter2D wavelet transforms.

10. Future Direction

We want to incorporate wavelet transforms to YOLO framework and directly perform object detection by combining long-chain wavelet transforms and short chain convolutions. FFT transforms can provide textural information and information about how similar various image patches are to each other and normal convolutional network can continue to use same GIOU based loss function to detect camouflaged objects.

References

[1] T. Han. Improving the detection and position of camouflaged objects in yolov8. *MDPI Electronics*, 12(20):234–778, 2023.

[2] X. L. Joshua Bassey. A survey of complex valued neural networks. *ARXIV https://arxiv.org/pdf/2101.12249*, 2101(12249):234–778, 2021.

[3] W. Ke. Large-scale filaments associated with milky way spiral arms. *https://ui.adsabs.harvard.edu/abs/2015MNRAS.450.4043W/abstract*, 1(0), 2015.

[4] T.-N. Le. Anabranch network for camouflaged object segmentation. *Computer Vision and Image Understanding*, 184(7):45–56, 2019.

[5] M. A. K. Rohan Putatunda. Vision transformer-based real-time camouflaged object detection system at edge. *2023 IEEE International Conference on Smart Computing*, 1(00029):234–778, 2023.