Transfer Learning for Fish Detection in Underwater Images

Humishka Zope
Stanford University
Department of Computer Science
zope@stanford.edu

Ishvi Mathai
Stanford University
Department of Computer Science
ishvim@stanford.edu

Abstract

Fish detection in underwater images is crucial for ecological research to enhance the understanding of aquatic habitats. However, this task presents significant challenges due to low illumination of underwater images, complex backgrounds, and the vast diversity of fish species. Existing detection models often underperform when detecting tiny fish, especially in murky water conditions. In this paper, we investigate the impact of Transfer Learning in improving the generalization of fish detection models for small-sized fish and low-quality underwater images. Using the Ozfish dataset, which comprises images of small fish and murky water conditions, we used the YOLOv7 model architecture to study fish detection with Transfer Learning. We evaluated three model variations on Ozfish: a pretrained YOLOv7 model, the same model fine-tuned on Ozfish through Transfer Learning, and a model trained from scratch on Ozfish. Additionally, we examined the impact that varying data quantities can have on the effectiveness of Transfer Learning. Our results indicate that increasing the amount of data used in Transfer Learning on a pretrained model improves model performance on our dataset. However, our results also show the difficulty of improving fish detection model performance on tiny fish, as our best performing model with Transfer Learning demonstrated very similar performance to our pretrained model on Ozfish.

1. Introduction

Fish detection in underwater images has long been an important task for ecological researchers, to better understand underwater habitats. However, this task can be quite difficult due to low illumination of underwater images, complex backgrounds, and the wide diversity of fish species. Currently, existing fish detection models do not perform as well on tiny fish, that are difficult to distinguish from the background due to their size. In addition, fish detection models often struggle to detect fish in lower quality underwater images due to murkiness in the water, which makes it more difficult to detect smaller fish as well. In our project, we focus on investigating the impact of Transfer Learning on building a fish detection model that generalizes better for small size fish and opaque images. Our dataset is Ozfish which contains images of small size fish and pictures taken in murky water.

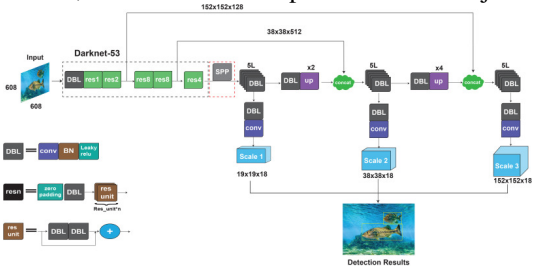
The input of our model is an image taken underwater. We then use a YOLOv7 model architecture (described below) to detect fish. The output is the image with predicted bounding boxes drawn over each detected fish.

Since the goal of our project is to investigate the impact of Transfer Learning on this task, we evaluated Ozfish on three types of models. First, we evaluated a pretrained YOLOv7 fish detection model (which had been pretrained on a large, open source fish dataset from the images on the public domain [14]). Then, we enhanced this pretrained model with Transfer Learning on Ozfish. Finally, we also trained the same model architecture on Ozfish from scratch (without any pretrained weights). We also ran further experiments to explore how the amount of data affects Transfer Learning.

2. Related Work

Al Muksit et al. [1] describe several variations of YOLO architectures for fish detection. The main model is YOLO-Fish which is a robust fish detection model which was published in December 2022 and accepted at Ecological Informatics. YOLO-Fish is one of the most widely used fish detection model used today. The YOLO-Fish model was trained on the DeepFish and Ozfish dataset, and it was found that it performed quite well on DeepFish (about 95% precision) but did not generalize as well to the Ozfish dataset (82% precision). The DeepFish dataset contains pictures of larger fish with an average of 3-4 fish per frame, while the Ozfish dataset contains small fish with an average of 25 fish per frame. The paper describes two different versions of YOLO-Fish: YOLO-Fish-1 and YOLO-Fish-2. The YOLO-Fish-1 architecture is diagramed below, where Scale 1 corresponds to large object detection, Scale 2 corresponds to medium object de-

tection, and Scale 3 corresponds to small object detection:



The model uses multiple upsampling layers to extract features based on the scale, and uses SPP (Spatial Pyramid Pooling) which consists of 4 layers of max pooling that are concatenated together for feature enhancement.

Jalal et al. [2] used a combination of Gaussian Mixture Models (GMM) and YOLO to detect and classify fish in two underwater video datasets: LifeCLEF 2015 benchmark from the Fish4Knowledge repository and a dataset collected by The University of Western Australia (UWA). They achieved fish detection F-scores of 95.47% and 91.2% on both datasets respectively. While GMM and optical flow helped extract temporal features, the model relied heavily on YOLO to detect static fish, like we have in our underwater fish images dataset.

Wang et al. [3] describe using YOLO-v3 for underwater object detection and classification on images from side scan sonar which is increasingly being used for underwater search. They trained the model on a set of 7000 underwater SSS images and conducted a series of experiments using the YOLO-v3 network. Evaluating using mAP with thresholds 0.5, 0.55, and 0.6, they found that the model was effectively able to detect objects in underwater data collected in real environment. We will similarly be using a threshold of 0.5 and AP as our metric as we are focused solely on detection.

Kandimalla et al. [4] evaluated YOLO-v3 to detect, count and classify fish from underwater videos from the Ocqueoc River DIDSON dataset. The they trained the YOLO-v3 model with pre-trained weights from ImageNet. We will also be using pre-trained weights to train our model. The initial results of YOLO-v3 was AP of 0.00 to 0.66 for the different classes calculated with an IoU of 0.5. However, after training with augmented data, the AP increased to a range of 0.31 to 0.86 over the different classes. We will also be using augmented data to train our model.

Xu and Matner [5] discuss using Transfer Learning to initialize their weights for training on their datasets, Voith Hydro, Wells Dam, and Igiugig. However they use weights that were trained to a detect a variety of different classes ranging from cats to cars but do not detect fish. In this paper we will be exploring Transfer Learning based on pre-trained weights meant to detect fish.

Saleh et al. [6] describe using Co-scale conv-attentional image Transformer (CoAT) encoders to outperform CNN-based encoders in detecting fish in underwater images. They utilized three different datasets: DeepFish, Seagrass, and Youtube-VOS. The paper found that a transformer-based encoder was better able to detect overlapping fish where CNN-based encoders failed. However, we found that the average precision of the YOLO-based methods is better.

Lin et al. [7] discusses modifications made to the DETR model that allow it to perform better on underwater images. A learnable query recall mechanism is incorporated in the decoder to mitigate the effects of noise like diffraction of light and suspended particles in water that can make images blurry. They also use AdaptFFN to keep track of fine-grained details in small objects. A lightweight adapter module is also added to each encoder and decoder. Depsite this, we found that YOLO-based models perform better than this approach on detecting fish.

Pagire et al. [8] compared Faster R-CNN ResNet50, YOLO-v3 and SSD MobileNetV2 on underwater fish detection and classfication on the Fish4Knowledge dataset. They found that YOLO-v3 had the best recall while SSD MobileNetV2 had the best precision.

Marrable et al. [9] used YOLO-v5 for fish detection on Ozfish and DeepFish. They achieved a precision of 0.898 and recall of 0.699 on Ozfish which is the dataset we are working with. However we will be using a more advanced version of YOLO, YOLO-v7 in our implementation.

We also reviewed Li et al. [10] approach to underwater fish detection using Fast RCNN on ImageCLEF which achieved mAP of 81.4%.

3. Methods

3.1. Project Goal

The goal of our project is to evaluate the impact of Transfer Learning on creating a fish detection model specifically for smaller sized fish.

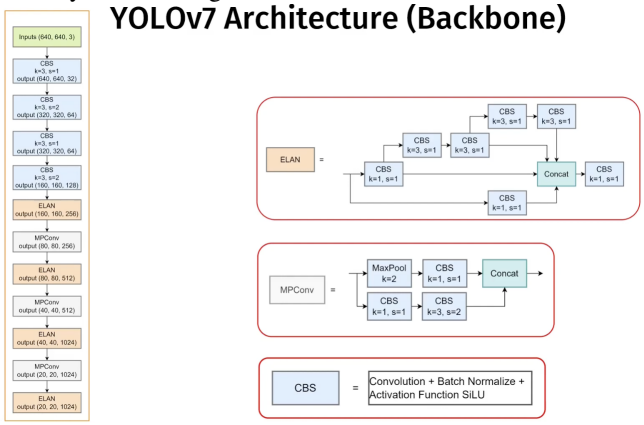
3.2. Model description

We used YOLOv7 [12] as our main model for this project. We used an existing codebase [11] that had most of the setup for training YOLOv7, which we used and modified to load, process, and train on our own dataset. YOLOv7 is a state-of-the-art object detector that has generally surpassed all other object detectors in both speed and accuracy. It is a single-stage object detector, that predicts bounding boxes and class probabilities for each input image.

3.2.1 Backbone Network

For feature extraction, YOLOv7 passes input images through a backbone. The input images are first passed through a series of CBS layers (which is a sequence of a convolutional layer, batch normalization, and a SiLU activation function). After this, the output of those

layers is then passed through alternating ELAN and MP-Conv layers. The diagram of the backbone is shown below:



ELAN (Efficient Layer Aggregation Network) is a structure to create feature maps by combining the outputs of different layers, while controlling the shortest gradient path so that accuracy doesn't deteriorate when modeling scaling is applied. Model scaling refers to learning the number of channels and the number of layers in each computational block. YOLOv7 utilizes model scaling in it's backbone, neck, and head.

3.2.2 Neck

The purpose of the Neck is to process the features extracted from the backbone. The Neck uses a CSPSPP module (a Cross Stage Partial Network with a Spatial Pyramid Pooling (SPP) block) and PAN (Path Aggregation Network), as well as ELAN-H blocks. The output is three different levels of processed input features. The complete diagram of the neck is shown below.

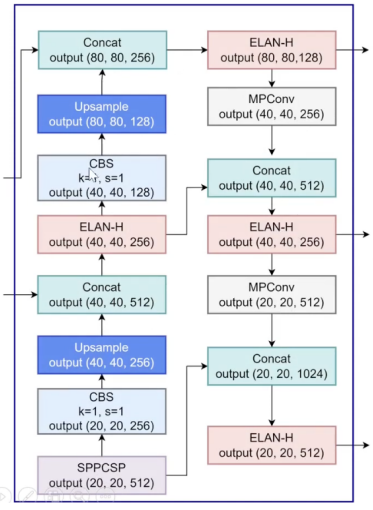


Figure 1. Diagram of YOLOv7 Neck

The CSPSPP module uses SPP (Spatial Pyramid Pooling), a pooling layer that is inserted between CBS blocks that allows for the model to handle a variable-sized input. SPP consists of multiple max pooling layers with different size bins (5, 9, and 13), and the output of those layers is concatenated together to a fixed size output. Note that the CSPSPP module sends the input features down two paths: one is fed through the SPP layer, and the other is fed through a simple 1 by 1 convolutional layer and concatenated at the end. This is done to prevent duplicate gradient information, reduce complexity, and preserve details.

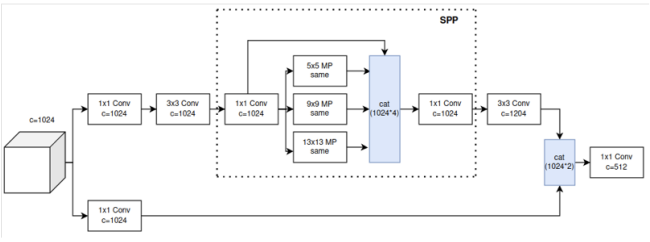


Figure 2. Diagram of CSPSPP module in YOLOv7 Neck

PAN improves the model's ability to localize information by optimizing the path of bottom-up information flow, to ensure that the model can use detailed features at multiple scales. Higher-level details are taken from deeper levels of the backbone and concatenated with lower-level details from later on in the network. This is done through upsampling (shown in Figure 1), where the resolution of higher-level features is increased, and then concatenated with the feature maps from earlier in the network.

3.2.3 Head

The Head receives three scales of input features from the Neck, which it uses to predict small, medium, and large size objects. YOLOv7 uses predefined anchor boxes as the starting point for it's prediction of bounding boxes, and three anchor boxes are allocated for each scale. During training, for each anchor box, the model learns to predict offsets of the box to better fit the objects it's detecting.

3.2.4 Loss Functions

Since YOLOv7 predicts both bounding box coordinates and class probabilities, it uses several loss functions.

CIoU (Complete Intersection over Union) Loss is used to measure how much predicted bounding boxes match the true bounding boxes. It takes into account overlap between boxes, distance between center points, and aspect ratio:

$$\text{CIoU Loss} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^g)}{c^2} + \alpha v \qquad (1)$$

Where:

- IoU is the Intersection over Union between the predicted box $\mathbf{b}$ and the ground truth box $\mathbf{b}^g$:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{2}$$

- $\rho(\mathbf{b}, \mathbf{b}^g)$ is the Euclidean distance between the centers of the predicted box and the ground truth box:

$$\rho(\mathbf{b}, \mathbf{b}^g) = \sqrt{(x - x^g)^2 + (y - y^g)^2} \tag{3}$$

  Where (x, y) and (x^g, y^g) are the center coordinates of the predicted and ground truth boxes, respectively.

- c is the diagonal length of the smallest enclosing box covering both the predicted and ground truth boxes:

$$c = \sqrt{(w + w^g)^2 + (h + h^g)^2} \tag{4}$$

  Where w and h are the width and height of the predicted box, and w^g and h^g are the width and height of the ground truth box.

- v measures the consistency of the aspect ratio:

$$v = \frac{4}{\pi^2} \left(\arctan \frac{w^g}{h^g} - \arctan \frac{w}{h} \right)^2 \tag{5}$$

- α is a positive trade-off parameter:

$$\alpha = \frac{v}{(1 - \text{IoU}) + v} \tag{6}$$

Objectness Loss refers to the model's ability to distinguish between the background and objects. It is the sum of Binary Cross Entropy Loss for each bounding box:

$$L_{\text{obj}} = -\left[t_i \log(p_i) + (1 - t_i) \log(1 - p_i) \right] \tag{7}$$

Where:

- p_i is the predicted objectness score for the i-th bounding box.

- t_i is the ground truth label for objectness (1 if an object is present, 0 otherwise).

The overall objectness loss considering all bounding boxes is:

$$\text{Objectness Loss} = \sum_i \left[-t_i \log(p_i) - (1 - t_i) \log(1 - p_i) \right] \tag{8}$$

Lastly, Class Prediction Loss uses Binary Cross Entropy Loss to ensure the correct class is assigned to each bounding box.

The class prediction loss for a single bounding box and a single class is defined as:

$$L_{\text{cls},ij} = -\left[t_{ij} \log(p_{ij}) + (1 - t_{ij}) \log(1 - p_{ij}) \right] \tag{9}$$

Where:

- p_{ij} is the predicted probability for the i-th bounding box belonging to the j-th class.

- t_{ij} is the ground truth label for the i-th bounding box and j-th class (1 if the object belongs to the class, 0 otherwise).

The overall class prediction loss considering all bounding boxes and all classes is:

$$\begin{aligned} \text{Class Prediction Loss} = \sum_i \sum_j \big[& -t_{ij} \log(p_{ij}) \\ & -(1 - t_{ij}) \log(1 - p_{ij}) \big] \end{aligned} \tag{10}$$

4. Dataset

The dataset that we are evaluating on and using for Transfer Learning is Ozfish, which contains 43k bounding box annotations across 1800 frames. The dataset was collected using video footage by the Australian Research Data Commons Data Discoveries program. The Ozfish dataset contains on average 25 fish per frame, and contains a variety of sizes of fish with many small fish, making it a good dataset to utilize for this project. Also, the majority of pictures were taken with low lighting and in more opaque settings, making it a more challenging dataset for fish detection. We used a train-validate-test split of 70-20-10 percent. For data augmentation, we ran experiments with two different datasets: Ozfish without data augmentation (680 images total) and Ozfish with data augmentation (4570 images total). The data augmentations included rotation between -7° and +7°, sheer of ±15°, horizontal and vertical blur up to 1.5 px, and noise up to 3 percent of pixels. The resolution of our images was 416 by 416 pixels.

5. Experiments, Results, Discussion

5.1. Experiment Overview

Since the goal of our project is to investigate the impact of Transfer Learning on fish detection, we first evaluated Ozfish on three types of models. First, we evaluated a pretrained YOLOv7 fish detection model (which had been pretrained on a large fish dataset). Then, we enhanced this pretrained model with Transfer Learning on Ozfish. Finally, we also trained the same model architecture on Ozfish from scratch (without any pretrained weights). When using Transfer Learning and training from scratch, we used

Figure 3. Examples of Ozfish images.

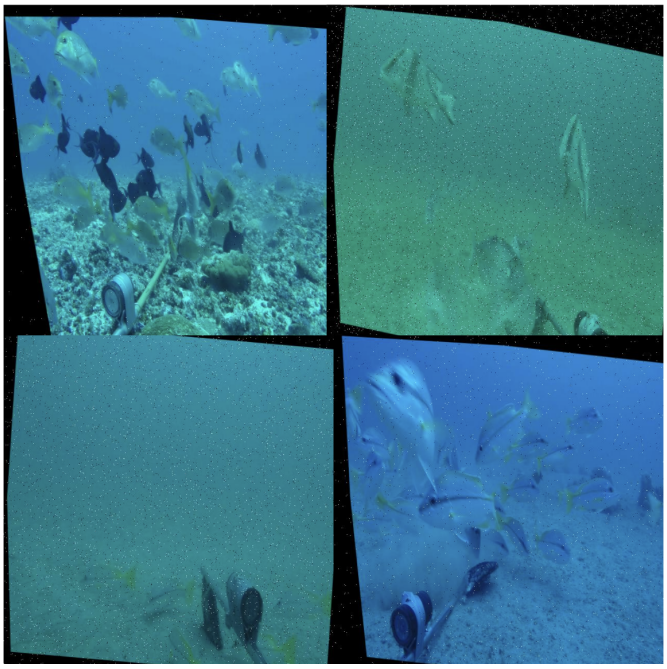


Figure 4. Examples of augmented images.

the two different datasets: one with augmentation, and one without augmentation. After this, we ran further experiments to explore how the amount of data used in Transfer Learning affects model performance.

5.2. Hyperparameters

Our learning rate was 0.01, and the optimizer was SGD with momentum, with a momentum coefficient of 0.937 and weight decay coefficient of 0.0005. Our mini-batch size was 64. We chose these parameters because these were the original parameters that the pretrained model was trained on, and we found that these parameters also worked best on our for scratch and finetuned models as well. To check this, we did a coarse and fine grid search of hyperparameters. To mitigate against overfitting, we plotted learning curves of train and validation error for each model we trained, to ensure that the model was not overfitting. This helped us determine to best number of epochs to train the model for.

5.3. Metrics

To evaluate model performance, we used two metrics: Mean Precision and Mean Recall. To calculate these, we used CIoU (as defined above) between the predicted and true bounding boxes.

We used an CIoU threshold of 0.5, meaning that if CIoU ≥ 0.5, it gets counted as a true positive. Mean Precision and Mean Recall were calculated as follows:

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

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

5.4. Results

c

Model	Mean Precision	Mean Recall
Pretrained	0.7628	0.9064
W/ Transfer Learning (no augmentation)	0.7365	0.7323
W/ Transfer Learning (w/ augmentation)	0.7581	0.9140
From scratch (no augmentation)	0.7408	0.7312
From scratch (w/ augmentation)	0.7366	0.7314

Table 1. Performance on Ozfish of Pretrained, Transfer Learning, and From Scratch YOLOv7

While all the models listed in Table 1 have relatively close Mean Precision values (all in a range of 3% or less) the Mean Recall values differ significantly. Recall corresponds to the model's ability to minimize false negatives (ie. the ability to correctly detect all the instances of fish in an image). Precision corresponds to accuracy of detected images (ie. the model is correctly detecting fish and not other objects). Thus, we see that all the models in Table [1] were relatively similar in the accuracy of their detections, but some models were much better than others in capturing all the fish present in the image. Since the Ozfish dataset contains an average of 25 fish per image (which is much higher than most other fish datasets), Mean Recall is an important metric for us to consider when evaluating performance.

From our results, we see that the model trained from scratch with augmentation performed the worst in terms of both Mean Precision and Mean Recall. While the

from scratch without augmentation performed slightly better, both From Scratch models did not perform relatively well. The fact that both of the From Scratch models did not perform as well confirms the notion that Ozfish is a difficult to data to train from scratch with, as it is difficult for the model to learn about the shapes and features of fish with smaller sized fish and more opaque images.

From our results, we see that the Transfer Learning with augmentation model performed very similar to the pretrained model, with less than a 0.5% difference in Mean Precision and less than a 0.8% difference in Mean Recall. This reveals to us that the Transfer Learning task on the augmented dataset did not have a noteworthy impact on the overall performance of the pretrained model on Ozfish. However, further qualitative analysis shows that model with Transfer Learning was able to better handle certain cases of images found in Ozfish. Figure 3 and Figure 4 show some images that were failure cases on the pretrained model, and their corresponding performance on the Transfer Learning with augmentation model. We see that with Transfer Learning, the model does appear to learn more about identifying fish from backgrounds, especially in cases where fish blend in with the background, but still struggles with pictures with many fish and has other similar failure cases.

Interestingly, our results show a significant difference between the Mean Recall for Transfer Learning without augmentation (0.7323) and Transfer Learning with augmentation (0.9140). Thus it seems like the method of Transfer Learning without augmentation actually significantly reduced the performance of the pretrained model, instead of enhancing it. We hypothesized that this was potentially due to the size of datasets used for the Transfer Learning. While the dataset utilized for Transfer Learning without augmentation contained only 680 images from Ozfish, the dataset used for Transfer Learning with augmentation contained 4570 images. Thus, in order to investigate the impact that the amount of data had on the quality of Transfer Learning, we decided to run further experiments where we utilized Transfer Learning on the same pretrained model, using only 25%, 50%, 75%, and 88% of the augmented Ozfish training data. Our results are reported in Table 2.

c

Model	Mean Precision	Mean Recall
From scratch w/ 100% training data	0.7366	0.7314
Pretrained, no Transfer Learning	0.7628	0.9064
Transfer Learning w/ 25% training data	0.6848	0.6902
Transfer Learning w/ 50% training data	0.7193	0.7203
Transfer Learning w/ 75% training data	0.7472	0.7332
Transfer Learning w/ 88% training data	0.7521	0.8379
Transfer Learning w/ 100% training data	0.7581	0.9140

Table 2. Model Performance on Ozfish, with different amounts of data used in Transfer Learning

From Table 2 we see that the Mean Precision and Mean Recall increase as we train with greater percentages of augmented Ozfish data. This follows our previous theory about the amount of data affecting the quality of Transfer Learning. We observed that Transfer Learning using 25% and 50% of the augmented training data performed worse than the from scratch model, which suggests that 25% and 50% of the augmented training data was not enough for the pre-

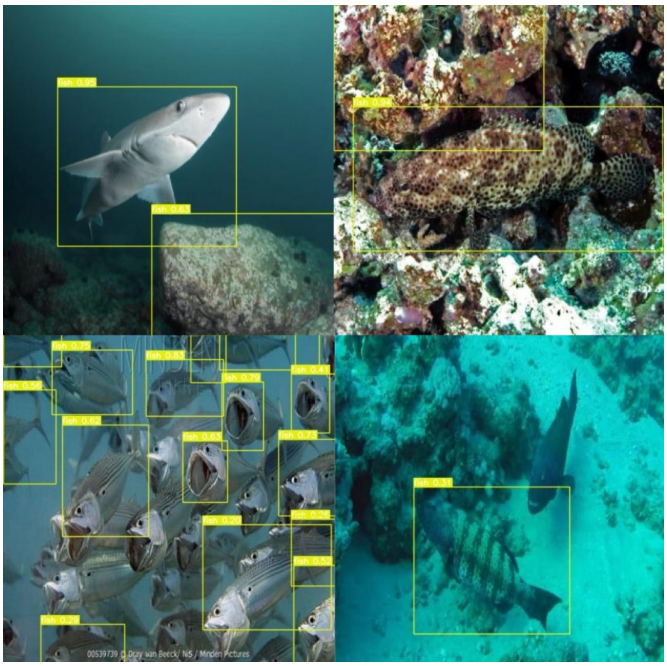


Figure 5. Failure cases of Pretrained model

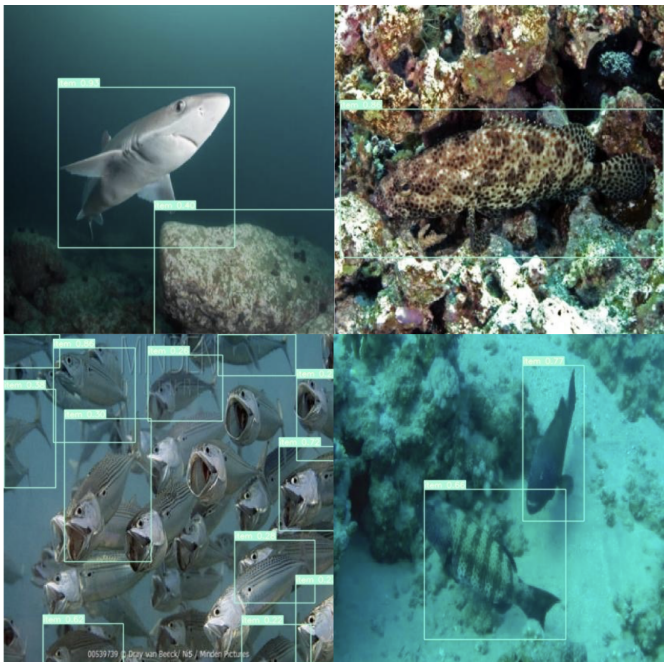


Figure 6. Same cases, with Transfer Learning model with 100% augmented dataset

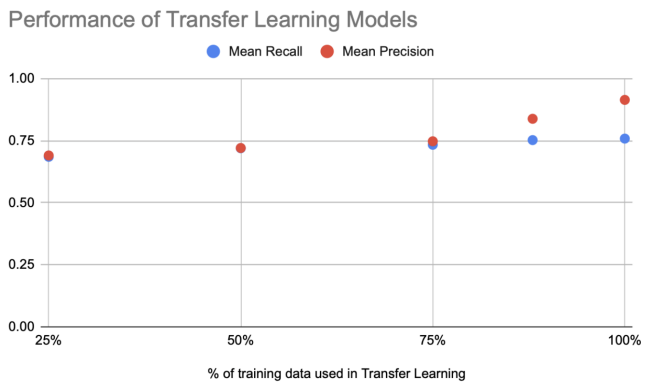


Figure 7. Plot of Mean Recall and Mean Precision versus % of data used in Transfer Learning

trained model to learn to detect fish in the Ozfish dataset. We then see that Transfer Learning using 75% of the Ozfish augmented training data yields better results than the model that was trained solely the Ozfish training data, and as we increase the amount of data used even further to 88 and 100 percent, the performance of the model increases substantially. This shows that given sufficient training data, the model can supplement its learning to perform well on Ozfish as well. Therefore it shows the effectiveness of Transfer Learning in creating better fish detection models. However, we note that in our experiments, we were not able to substantially improve upon our pretrained model's performance on Ozfish, highlighting the difficulty in training a fish detection model for small fish and murky water.

6. Conclusion & Future Work

Through experimenting with augmented data and training using different amounts of data we have seen that the amount of data used in Transfer Learning significantly betters the impact of Transfer Learning for detecting fish in underwater images. Fish detection in murky water and for small fish continues to be a difficult task for object detection models, as is demonstrated in our results.

In the future we hope to continue bettering our fish detection model by training on more data with small fish in murky water. This would include finding other datasets that contain small fish or reaching out to ecological researchers to create our own. We also hope to explore other object detection techniques like SSD, Faster-RCNN, and more advanced versions of YOLO. These object detection models employ a variety of techniques that could serve better or worse in fish detection in underwater images.

7. Contributions

Humishka Zope trained the pretrained model and the finetuned models with 25 percent, 88 percent, and 100 percent of data. Ishvi Mathai trained the for scratch models and the 50 percent and 75 percent of data. We both wrote various sections of the paper. There were no other collaborators. We made use of the public code: https://github.com/pathikg/Underwater-fish-detection/blob/main/YoloV7%20training%20notebook/Training_YOLOv7_on_Fish_Dataset.ipynb.

References

[1] Muksit, A. A., Hasan, F., Emon, M. F. H. B., Haque, M. R., Anwary, A. R., Shatabda, S. (2022). YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment. Ecological Informatics, 72, 101847. https://doi.org/10.1016/j.ecoinf.2022.101847

[2] Jalal, A., Salman, A., Mian, A., Shortis, M., Shafait, F. (2020). Fish detection and species classification in underwater environments using deep learning with temporal information. Ecological Informatics, 57, 101088. https://doi.org/10.1016/j.ecoinf.2020.101088

[3] Y. Wang, J. Liu, S. Yu, K. Wang, Z. Han and Y. Tang, "Underwater Object Detection based on YOLO-v3 network," 2021 IEEE International Conference on Unmanned Systems (ICUS), Beijing, China, 2021, pp. 571-575.

[4] Kandimalla, V., Richard, M., Smith, F., Quirion, J., Torgo, L., & Whidden, C. (2022). Automated detection, classification and counting of fish in fish passages with deep learning. Frontiers in Marine Science, 8. https://doi.org/10.3389/fmars.2021.823173

[5] Xu, W., & Matzner, S. (2018, November 5). Underwater Fish Detection using Deep Learning for Water Power Applications. arXiv.org. https://arxiv.org/abs/1811.01494

[6] Saleh, A., Sheaves, M., Jerry, D.R., & Azghadi, M.R. (2022). Transformer-based Self-Supervised Fish Segmentation in Underwater Videos. ArXiv, abs/2206.05390.

[7] Lin, X., Huang, X., & Wang, L. (2024). Underwater object detection method based on learnable query recall mechanism and lightweight adapter. PloS one, 19(2), e0298739. https://doi.org/10.1371/journal.pone.0298739

[8]Pagire, V., Phadke, A. C., & Hemant, J. (2024). A deep learning approach for underwater fish detection. Journal of Integrated Science and Technology, 12(3). https://doi.org/10.62110/sciencein.jist.2024.v12.765

[9] Marrable, D., Barker, K., Tippaya, S., Wyatt, M., Bainbridge, S., Stowar, M., & Larke, J. (2022). Accelerating species recognition and labelling of fish from underwater video with Machine-Assisted Deep Learning. Frontiers in Marine Science, 9. https://doi.org/10.3389/fmars.2022.944582

[10] Fast accurate fish detection and recognition of underwater images with Fast R-CNN. (2015, October 1). IEEE Conference Publication — IEEE Xplore. https://ieeexplore.ieee.org/document/7404464

[11] Ghugare, P. P. (2022). Underwater Fish Detection. GitHub. https://github.com/pathikg/Underwater-fish-detection

[12] Wang, C. Y., Bochkovskiy, A., Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 7464-7475).

[13] Australian Institute Of Marine Science, 2020. Ozfish dataset - machine learning dataset for baited remote underwater video stations.

[14] Solawetz, J. (2023, February). Fish Dataset. Roboflow Universe. Roboflow. Retrieved June 6, 2024, from https://universe.roboflow.com/roboflow-gw7yv/fish-yzfml