Modeling, Identification and Control, Vol. 42, No. 1, 2021, pp. 1–16, ISSN 1890–1328

NorFisk: fish image dataset from Norwegian fish farms for species recognition using deep neural networks

A. M. Crescitelli [1] L. C. Gansel [1] H. Zhang [2]

[1] *Department of Biological Sciences, Norwegian University of Science and Technology, Ålesund, Norway. E-mail:* {alberto.m.crescitelli,lars.gansel}@ntnu.no

[2] *Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, Ålesund, Norway. E-mail:* hozh@ntnu.no

Abstract

Long-term autonomous monitoring of wild fish populations surrounding fish farms can contribute to a better understanding of interactions between wild and farmed fish, which can have wide-ranging implications for disease transmission, stress in farmed fish, wild fish behavior and nutritional status, etc. The ability to monitor the presence of wild fish and its variability with time and space will improve our understanding of the dynamics of such interactions and the implications that follow. Automatic fish detection from video streams at farm sites using neural networks may be a suitable tool. However there are not many image datasets publicly available to train these neural networks, and even fewer that include species that are relevant for the aquaculture sector. This paper introduces the first version of our dataset, NorFisk, which can be found publicly available at Crescitelli (2020). It contains 3027 annotated images of saithe and 9487 of salmonids and it is expected to grow in the near future to include more species. Annotated image datasets are typically built manually and it is a highly time-consuming task. This paper also presents an approach to automate part of the process when generating these types of datasets with fish underwater. It combines techniques of image processing with deep neural networks to extract, label, and annotate images from video sources. The latter was used to produce NorFisk dataset by processing video footage taken in several fish farms in Norway.

Keywords: Annotated image dataset, Deep neural networks, Fish detection, Fish species recognition, Marine aquaculture applications.

1 Introduction

Wild fish is attracted to aquaculture sites for a variety of reasons and previous studies have shown that large numbers of wild fish may congregate near fish farms Tuya et al. (2006), Fernandez-Jover et al. (2008). Wild fish can interact with farmed fish in varying ways. For instance farmed fish may perceive wild fish as potential predators. If they try to enter fish cages through the netting, this can stress farmed fish and change their behavior. Wild fish may also feed on lost feed pellets or farmed fish feces, which can impact their foraging behavior and body composition. Diseases and parasites can spread between farmed and wild fish as well. In order to understand such interactions and to identify and manage potentially negative impacts of interactions between wild and farmed fish, it is important to have good knowledge of the presence and behavior of wild fish in the vicinity of fish farms. This includes variations of the distribution and composition of wild

doi:10.4173/mic.2021.1.1

fish assemblages. To date we lack proper monitoring systems for wild fish in the water column around and on the bottom below fish farms. As many authors have been proposing in recent years (Bjelland et al. (2015), Grotli et al. (2015), Utne et al. (2015)), these conditions are drivers for the development of autonomous systems for long-term monitoring to support farmers' decisions and increase fish welfare and farms production.

One of the problems to tackle with respect to autonomous monitoring systems is fish species detection. Reliable systems that continuously recognize fish species surrounding fish farms would enable scientists to evaluate farm-wild fish interactions with more detail. The oceans present a high environmental diversity that makes it hard for scientists to reach a final solution with respect to fish detection. Because of this, different authors have proposed several approaches.

Zhang et al. propose an automatic method for fish counting using image density grading with local regression in water tanks Zhang et al. (2020). Hossain et al. have applied techniques of computer vision with machine learning for fish species classification Hossain et al. (2016). Others have used convolutional neural networks (CNN) for the same task (Salman et al. (2016), Tamou et al. (2018)). They all have used the well known LifeCLEF Fish dataset Joly et al. (2015) as a benchmark. Rauf et al. also use CNNs for fish species classification Rauf et al. (2019), but they apply it on the Fish-Pak dataset Shah et al. (2019). Pelletier et al. Pelletier et al. (2018) implement CNN architectures for classifying fish on board fishing boats with a dataset provided by The Nature Conservancy Nat (1964).

Rathi et al. Rathi et al. (2018) applied CNN on another public benchmark called Fish4Knowledge Boom et al. (2012), as well as Wang et al. (2019), proposing a deep encoding-decoding network for fish classification. Other authors have created their own fish species datasets for image classification from video footage. They have annotated the images manually. Examples of this are found in Villon et al. (2018) and Siddiqui et al. (2018). On the side of object recognition/segmentation for fish species, Cutter et al. Cutter et al. (2015) utilize Haar-cascades Viola and Jones (2001) for detecting rockfish using their own dataset. Salman et al. Salman et al. (2019) apply region-based convolutional neural networks (R-CNN) Girshick et al. (2014) under the benchmark LifeCLEF; Mandal et al. Mandal et al. (2018) implement Faster R-CNN Ren et al. (2017) and train the neural networks with their own dataset. Liu et al. Liu et al. (2018) use YOLOv3 Redmon and Farhadi (2018) with the benchmark Fish4Knowledge to detect and track fish. Raza and Hong Raza and Hong (2020) propose an improved version of YOLOv3 to perform detections on a custom dataset of four fish species.

The task of annotating images is usually done manually (Boom et al. (2012), Joly et al. (2015), Villon et al. (2018)). As Raza Raza and Hong (2020) and other authors state, bounding boxes labeling and annotations must be taken with much care to reach an accurate model, which translates into a very time-consuming process. Most fish classification/recognition solutions have been built using the LifeClef and Fish4Knowledge benchmarks that contain coral reefs species. There is a lack of data, publicly available, regarding fish species that are relevant for aquaculture.

The first contribution of this paper is a public dataset of annotated images with salmonids and saithe species which can be found at Crescitelli (2020). The second contribution is the approach used to generate and annotate this dataset. It consist of a semi-automatic system combining basic techniques of computer vision with the state-of-the-art of neural networks.

In the next section, the proposed approach and the different parts of the system are explained. In section 3 the procedure and experiments setup performed to validate the system are presented. Section 4 presents and discusses the results. Sections 5 and 6 present the paper's conclusions and directions for future work.

2 Proposed Approach

We propose to use techniques of image processing (IP) combined with deep neural networks (DNN), to build a mechanism that enables the creation of large datasets of fish images, with annotations, in a semi-automatic approach. Scientists typically annotate images manually, sometimes using software such as *LabelImg* to speed up the process. This type of software requires the user to draw a bounding box around every object of interest in each image and give it a label. The software essentially takes the coordinates of the bounding boxes and labels and generates the annotation files in the right format. In this work the goal is to go a step further and generate the bounding boxes automatically.

2.1 Top level description

The top-level architecture of the developed system is shown in figure 1. It has three main components: *Image Processing Stage (IPS)*, *Image Extraction Stage (IES)*, and *Neural Network Stage (NNS)*.

The IPS is especially useful for creating a dataset of a certain class (e.g. salmon) from scratch, without any previous data regarding that class. It uses basic

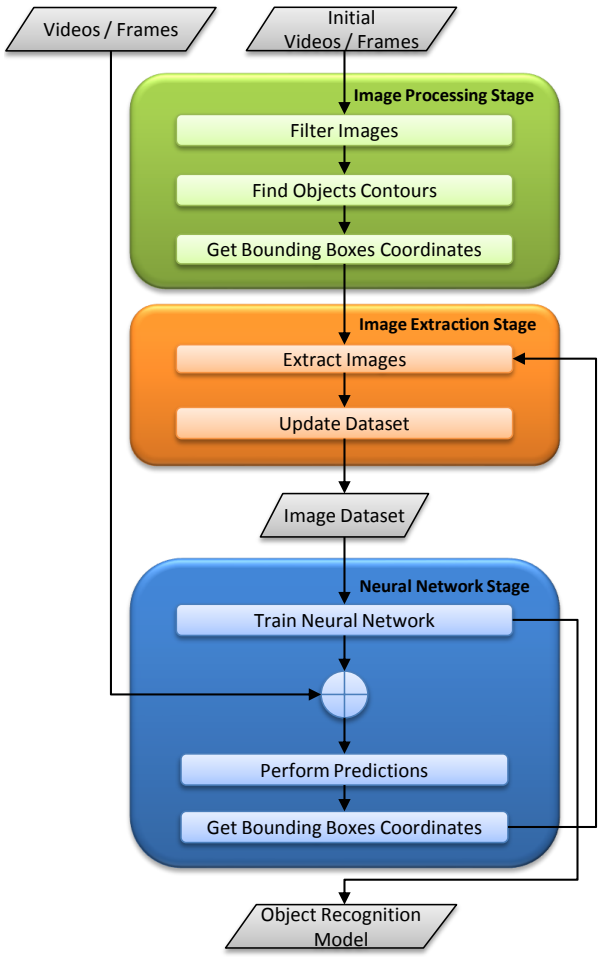

Figure 1: Flow diagram of the top level view

techniques of image processing to enhance and isolate objects of interest in environments where objects are easily separable from the background, such as fish underwater. The output of this stage is a set of bounding boxes coordinates enclosing the isolated objects. The details are explained in section 2.2.

As many bounding boxes found in the previous step might be inaccurate for our purpose, IES will evaluate the list of boxes and decide which boxes to keep and which ones to discard. The images inside the remaining boxes will be prepared for extraction and added to the dataset with their corresponding labels and annotations. The description of the algorithm is given section 2.3.

Once a small dataset of some dozens or a few hundreds of images is created, it will be used to train a neural network (NN) for object recognition, such as YOLOv3 Redmon and Farhadi (2018). This step corresponds to the third block in figure 1 (i.e. NNS). From this point forward an NN is used to find the bounding boxes instead of the IPS. Details are described in 2.4.

2.2 Image processing stage

This is the starting point of the system when we do not have any image of the class we are dealing with. In this initial stage, bounding boxes for the fish present in each frame are found by filtering the objects of interest from the background.

In order to train the NN to recognize fish species, at least a few hundred annotated images are needed. To obtain this first dataset, bounding boxes are created by using simple techniques of image processing to "filter" the objects of interest. A set of parameters must be tuned in order to filter the desired objects. This stage is sensitive to variations in light conditions and therefore, if the conditions change much, the parameters need to be readjusted. In consequence, this stage is not a general solution for any type of objects but it is very suitable for objects easily separable from their background, such as fish underwater. Once the parameters are tuned (which only takes a few minutes), the algorithm can process an entire video filtering the fish. It generates the bounding boxes and annotations, with a predefined label, automatically.

The steps of the algorithm are shown in figure 2 and it works as follows. Firstly, each image is split into HSV (Hue-Saturation-Value) channels and each channel is processed independently. This gives us more degrees of freedom. In each channel the following functions are applied:

- *Blurring*: To smooth the image reducing noise.
- *Global Histogram Equalization*: To balance the global contrast. Under water there is usually a gradient of brightness as a consequence of the light coming from the surface.
- *Contrast Limited Adaptive Histogram Equalization (CLAHE)*: This is an adaptive equalization by regions. It is a more refined equalization than the previous step.
- *Morphological Transformations Closing-Opening*: *Closing* fills the small holes present in objects, while *opening* removes small regions of pixels in the background that act as noise.

Each of the steps described thus far were implemented using the library OpenCV and the details regarding how each operation works are available at ope (2020). The parameters for each operation are tuned differently for each channel in order to maximize flexibility. After the image is "cleaned", the three channels (H-S-V) are merged together again. As shown in figure 2, at this point of the process, the object of interest (e.g. fish) is enhanced. This facilitates detection later in the process.

The following step is to apply a binary threshold in each channel to filter the image. By default, each pixel in each channel can take a value between 0 and 255. Therefore, a lower and upper threshold are defined in order to decide which range of pixel values will survive the filtering. There might be an undesired remainder in the borders of the frame as a consequence of applying the histogram equalization. To delete this effect, the option to trim the borders of the frames is included. At this point in the chain, the result is a binary mask with only the fish. The following step is to perform closed contour detection. In this part every contour present in the image will be detected. The list of contours is then sent to the function "Clean Contours". The Clean Contours function performs two controls. First, it discards all contours that are under a specified size. This is due to the fact that some contours might be too small. Second, it controls that contours are not on the borders of the frame. This is because only full-body fish are considered for extraction. The rest are dropped from the list.

The final task of the IPS is to output the bounding boxes coordinates of the remaining contours. This is easily done by taking two vertices that correspond to the minimum and maximum values of (x,y) for each contour.

2.3 Image extraction stage

When using a dataset for image classification, it is enough to have images that just contain the object of interest (OOI). Object recognition, on the other hand, makes it necessary to provide information regarding the coordinates of the OOI inside the image. This information is contained in an annotation file. There are several formats and standards for annotation files, but essentially, it is a text file that includes the following data:

- Path to the image file.
- Name of the class (e.g. saithe, salmon, etc.).
- Coordinates of the bounding box that encloses the OOI inside the image.

As the goal is to create datasets for multi-purpose applications, the latter case is considered. In consequence, for every image that is extracted from a frame, an annotation file is also created.

In this work, for simplicity, the focus was placed on creating images with only one object per image. Future work will include images with multiple objects. The original frames may have several fish. At the same time, in order to help NNs to generalize better, there

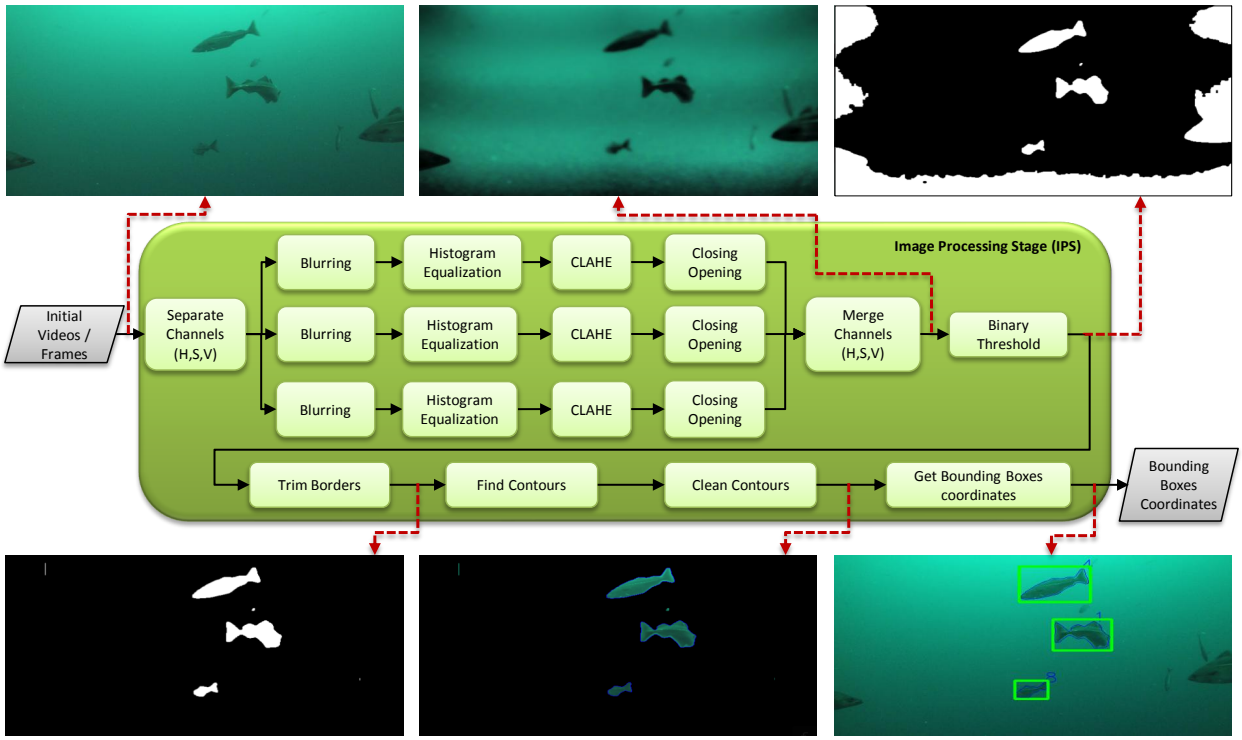

Figure 2: Flow diagram of the image processing stage (IPS)

has to be as much background as possible in each image. Therefore, regions of interest for each fish are cropped, annotated, and saved as different images in our dataset. These are the images that will be used to train NNs. In conclusion, the following two boxes for each fish are handled:

- *Bounding Box:* It is the bounding box found in the previous stage. It encloses the contour of the fish.
- *Region Of Interest (ROI)*: It is a scaled version of the bounding box that includes the object and part of the background.

This stage takes the list of bounding boxes coordinates produced by the IPS and performs a number of checks to discard undesired bounding boxes. It is also encharged of generating ROIs while checking that they do not overlap with other OOI. At the end, the IES outputs labeled and annotated images that will be added to the dataset. The flow chart of the functionality is shown in figure 3 and it works as described below.

The first step is to check and discard those bounding boxes that are on the borders of the frame. These are considered to be fish getting in or out of the field of view. With the remaining bounding boxes a ROI for each one needs to be generated. To generate a ROI, first, the bounding box is scaled up with a random size in a predefined range of values. The original aspect ratio of the image is conserved. Then, it is checked that the ROI is not passing the borders of the image. If it does, the ROI is moved inside the image keeping the size (this means that the fish is not necessarily centered inside the ROI). Next, it is checked that ROIs are not overlapped with bounding boxes that belong to other ROIs. If any do, the size of that ROI is reduced to the minimum to avoid the overlap. If the bounding boxes are the ones overlapping (e.g. when one fish is behind another one), they are discarded. Finally, the remaining ROIs are cropped from the original frame, the annotations are created automatically based on the bounding boxes coordinates, and this new data is added to our dataset.

After saving this information for a few dozens or hundreds of fish, there are two choices. One option is to keep expanding our dataset using the IPS and IES with more videos, semi-automatically, by readjusting the IPS parameters when needed. The other option is to automate this process by including an NN, as explained in next section.

2.4 Neural network stage

After creating an initial dataset with a few hundred images, an NN can be trained to recognize the object

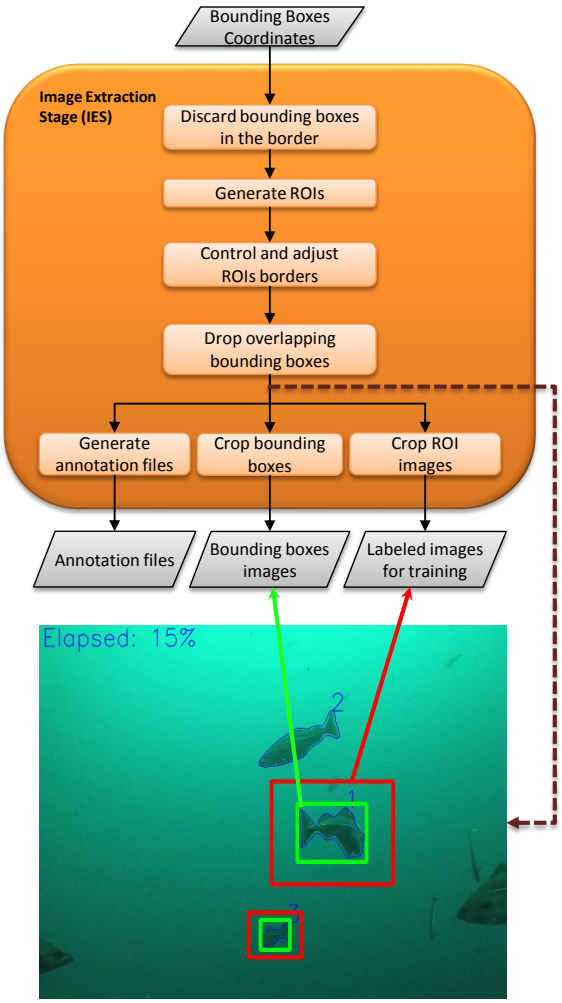

Figure 3: Flow diagram of the image extraction stage (IES)

of interest by itself (e.g. certain species of fish). Therefore, from this point on, the task of finding bounding boxes can be addressed by the NN. By definition CNNs can capture features in images. This makes them more robust against light or contrast variations. In consequence, when using a CNN, there is no need to tune the parameters of the image processing stage anymore. Thus, the whole process, from the video footage as input to the final product (i.e. annotated images in the dataset), can be automatic.

The reader might wonder about the accuracy given the small dataset used for training. This is where the iterative loop of figure 1 comes in. In the first instance of predictions, the performance of the neural network will be poor. It might detect a few fish and with low accuracy. The detections that are above a predefined threshold of accuracy will be added to the dataset. Later on, the NN will be re-trained with these new

images. In consequence, the accuracy and generalization is expected to increase in each iteration. At the same time this should enable the NN to make predictions in more diverse scenarios. The iterative loop of expanding the dataset and re-training the NN to make it more general and accurate can be repeated as long as desired.

Neural Network Selection. Region-based CNNs, such as R-CNN in all its versions, are complex and involve a two-stage pipeline: one stage to generate potential bounding boxes and one stage to run a classifier with post-processing on those proposed boxes. By contrast, YOLOv3 is a single-stage that consists on a regression problem, a single CNN that simultaneously predicts bounding box coordinates and class probabilities straight from the image. This makes YOLOv3 extremely fast compared to other methods. Besides speed, region-based techniques observe images by sliding windows. YOLO's architecture instead, sees the whole picture during training and testing. Thus it also encodes contextual information. This enables YOLO to make a much lower number of background errors compared, for example, to Fast R-CNN.

While YOLO can quickly identify objects, its major downside is that the precision of localization of objects is not as good. The loss function treats errors in small boxes the same as in large boxes. A small error in a small box, for example, has greater effect than a small error in a big box. This is not reflected in the loss function, and therefore YOLO's main source of errors is localization errors. Overall the speed of YOLOv3, together with its accuracy, simplicity, and low number of background errors make it the most suitable object detector for this work.

Bounding Boxes Detection. The input image is divided into a grid of $S \times S$. Each grid is encharged of predicting C conditional class probabilities defined as $P(Class_i|Object)$ and B bounding boxes. YOLOv3 predicts bounding boxes using hand-picked priors called anchor boxes. It only predicts offsets and confidences with respect to the anchor boxes. Using anchor boxes decouples the class prediction from the spatial location. With anchor boxes, YOLOv3 can predict more than a thousand boxes per image.

Each bounding box is defined as follows:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ Objectness &= \sigma(t_o) \end{aligned} \tag{1}$$

where $(b_x,\ b_y)$ represents the center of the box and $(b_w,\ b_h)$ is the width and height. The Objectness is defined as $Objectness = P(object)IOU(b, object)$ and represents the confidence that there is an object in that place. IOU is defined as the intersection area between two bounding boxes divided by the area under the union of the two bounding boxes. It is a measure of how much overlap the two boxes have with respect to the total area they occupy. $IOU = 1$ means the two boxes have equal size and they are fully overlapping. On the right-hand side of the equation 1, $(c_x,\ c_y)$ represents the grid cell's offset from the top left corner of the image. p_w and p_h are the width and height of the bounding box prior (i.e. the anchor). The NN predicts the 5 parameters $[t_x, t_y, t_w, t_h, t_o]$ that form the bounding boxes as offsets of the anchors. $\sigma(.)$ is the sigmoid function. This parametrization makes the NN more stable and easy to learn, as Redmon and Farhadi Redmon and Farhadi (2016) describe.

Architecture. YOLOv3 uses Darknet-53 Redmon (2016) as a backbone architecture which has 53 convolutional layers and has been pre-trained on the ImageNet dataset in order to learn basic and general features. For object detection, YOLOv3 adds 53 more layers on top of Darknet-53, for a total of 106 layers. The full YOLOv3 is trained on the Microsoft COCO dataset Lin et al. (2014) and that is the starting point in this paper. Transfer learning is used to fine-tune the NN with our own fish images.

YOLOv3 is a multi-label detector that uses independent logistic classifiers. This means that every bounding box can predict multiple classes. This architecture predicts boxes at three different scales concatenating feature maps from different layers at different sizes, using a similar concept to feature pyramid networks. This multi-scale feature allows the network to extract finer-grained information from early higher-resolution feature maps. YOLOv3 uses 9 anchors divided in 3 scales. The anchors are found automatically using k-means clustering on the training dataset. Since the model only uses convolutional and pooling layers, it can be resized on the way. By default, the NN changes the input size randomly every 10 batches while training. This forces the network to learn across a variety of input dimensions and allows it to perform predictions at different input resolutions.

Loss. The loss function to be minimized by YOLOv3 has three components. That is, the loss that accounts for the error in the localization of the bounding boxes, the loss that accounts for the objectness, and the loss in classification. The total loss is given by:

$$Loss = L_{coord} + L_{conf} + L_{class} \tag{2}$$

where:

$$L_{coord} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(t_{x_i} - \hat{t}_{x_i}\right)^2 + \left(t_{y_i} - \hat{t}_{y_i}\right)^2 + \left(t_{w_i} - \hat{t}_{w_i}\right)^2 + \left(t_{h_i} - \hat{t}_{h_i}\right)^2 \right] \tag{3}$$

$$L_{conf} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \text{BCE}\left(t_{o_i}, \hat{t}_{o_i}\right) + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \text{BCE}\left(t_{o_i}, \hat{t}_{o_i}\right) \tag{4}$$

$$L_{class} = \sum_{i=0}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} \text{BCE}\left(p_i(c), \hat{p}_i(c)\right) \tag{5}$$

BCE stands for Binary Cross Entropy and it is defined as:

$$BCE\left(x, \hat{x}\right) = -x \log \hat{x} - (x - \hat{x}) \log\left(1 - \hat{x}\right) \tag{6}$$

The symbols with a hat represent the predicted values and the ones without hats are the ground truth values. $\mathbb{1}_i^{obj}$ is equal to 1 if an object appears in cell i and zero otherwise. $\mathbb{1}_{ij}^{obj}$ will be one when j is equal to the jth bounding box predictor in cell i that is responsible for that prediction. This means the predictor with the highest IOU in that grid cell. $\mathbb{1}_{ij}^{noobj}$ simply takes the opposite value (i.e. 0 or 1) of $\mathbb{1}_{ij}^{obj}$. From 5 it can be seen that the loss function only penalizes classification errors if an object is present in that grid cell. It also penalizes bounding box coordinates errors if that predictor is the one responsible for the ground truth box.

The model weights localization errors equally with classification errors. Also, some grid cells may not contain objects. This takes the confidence scores of those cells to zero, increasing the gradient of those cells that do contain objects. This can lead to model instability causing the training to diverge. λ_{coord} and λ_{noobj} solve this issue by controlling how much to weigh the loss from bounding box coordinates and the confidence. The sum-squared error also weights errors in large boxes and errors in small boxes in the same way. Clearly this is not ideal, since an error in a small box has much greater effect on IOU than in a bigger box. For more details regarding YOLO, refer to Redmon et al. (2016)Redmon and Farhadi (2016)Redmon and Farhadi (2018).

Metrics. The metric used to measure the accuracy of our models is Average Precision (AP). Calculating it requires understanding three other metrics: Intersection Over Union (already explained), Precision, and Recall.

Precision is the proportion of the predictions (i.e. detected boxes) that are correctly identified. Recall is the proportion of the ground truth objects that are correctly detected. A prediction is considered to be correct when its IOU against the ground truth box is above a specific threshold. In this work an IOU threshold of 0.5 was used.

Mathematically:

$$Precision = \frac{TP}{TP + FP}; \quad Recall = \frac{TP}{TP + FN} \tag{7}$$

- True Positives (TP) is the number of the detected bounding boxes with a IOU higher than the threshold.
- False Positives (FP) is the number of detected bounding boxes with a IOU below the threshold (meaning the prediction does not overlap a ground truth bounding box).
- False Negatives (FN) is the number of ground truth boxes that are not detected.

The AP is the average of the precision for every recall value. In other words it represents the area under the precision-recall curve. To build the precision curve as a function of the recall, first the predictions in the whole testing set have to be performed. Then, a table with the confidence of each bounding box sorted by precision in descending order is built. Next, for every predicted bounding box, the values of TP and FP are assigned. These values are used to create an accumulative TP and FP to compute the precision equation for each value. For each precision value the recall is calculated using the accumulative TP. TP+FN is equal to the number of total ground truth boxes in the testing dataset. Once we have the precision and recall values for each predicted bounding box in an accumulative order, the precision-recall curve is built. Finally, the AP is obtained by computing the area under the curve.

In YOLOv3, performance drops significantly as the IOU threshold increases. This indicates that YOLOv3 struggles to get the boxes perfectly aligned with the objects.

Predictions in multiple resolutions. In the training stage, YOLOv3 randomly changes the images input size to make the model more adaptable to different sizes. The prediction stage, instead, runs only at one specific input size, typically small if real-time processing is needed. Due to the process of resizing, small objects may escape detection due to the loss of fine-grained details. On the other hand, if high resolution

images are used, the NN may not recognize big objects if the dataset does not have many. We have found that performing predictions at different resolutions in the same image and combining the outputs gives a better result. Figure 4 shows an example of this, demonstrating that the combined result is better than any of the single-resolution detection results.

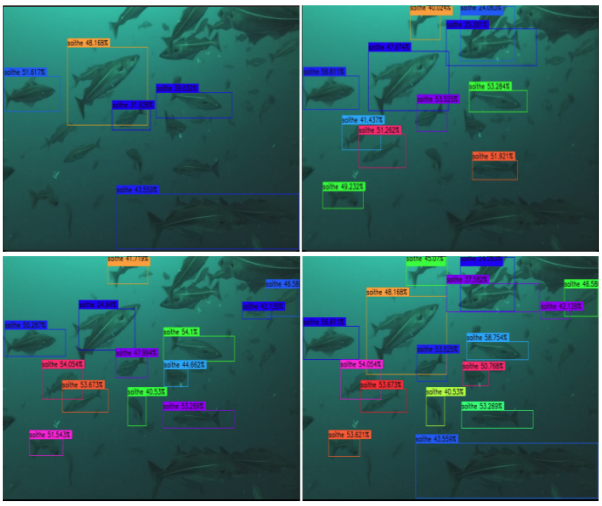

Figure 4: Performing detection in multiple input resolutions. The results are given for input sizes of 416 (top-left corner), 608 (top-right corner), 1024 (bottom-left corner) and the combination of 224, 416, 608, 864, and 1504 simultaneously, applying non-maximum suppression to delete repeated boxes (bottom-right corner).

When the same object is detected in different resolutions, the one with the highest score is kept and non-maximum suppression is applied to delete the other ones. The price for this is, of course, a decrease in speed. Nevertheless, as the goal here is to use this tool to extract images for creating a dataset, running slower than real time does not represent a problem. In this work, the input sizes 224, 416, 608, 864, and 1504 are simultaneously used to perform predictions.

Final model. Doing the work of cropping images, labeling them and creating annotations manually can be extremely time consuming. By doing the job automatically, not only a new dataset is created and expanded fast, but also one ends with a trained model that can already be used for detection in monitoring applications.

3 Experiments

In this section the set of experiments performed to test the system is introduced first. Next, the process of collecting the data is presented. At the end the hardware, software, and NN's configurations are described.

3.1 Experiments Scheme

As the system involves an iterative process of re-training the NN model at the same time the dataset grows, a key for this process is to find the actual best way of re-training. At the same time the authors aim to verify that the system can create large datasets with little human intervention. The system was tested for two species of fish, that is: saithe and salmonids (grouping farmed salmon and trout). The test sequence performed is summarized in the flow diagram of figure 5. This was done for both fish species separately.

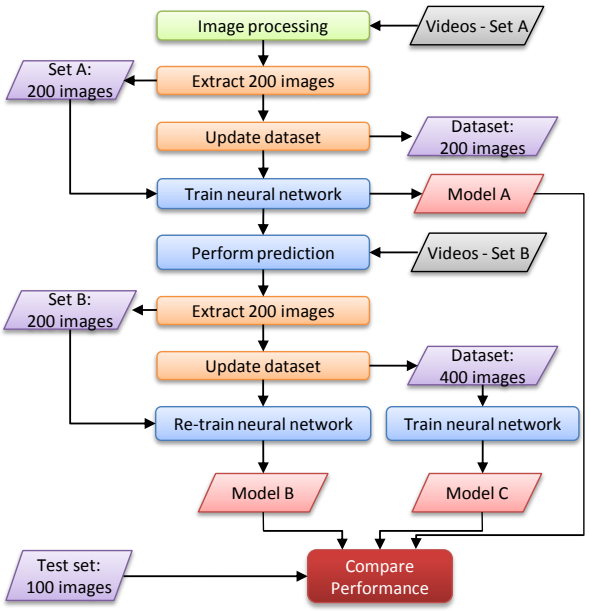

Figure 5: Flow diagram of the experiment's setup.

The process starts by analyzing a set of videos with the IPS. In this stage the first set of 200 images (i.e. *Set A*) was extracted. Next, this set was used to train the NN, producing thus the first model (i.e *Model A*). Prior to the first training, transfer learning was used to initialize the NN with weights previously trained on the MS COCO dataset. This enables the NN to save the time of learning basic features and makes it converge faster by only fine-tuning the model with our particular dataset.

With the first NN model, predictions are performed in order to find bounding boxes on a second set of

videos. Two hundred new images were extracted into what the authors called *Set B*. The previous NN's weights from Model A were used as starting point to re-train the NN with the images from Set B. The result was called *Model B*. It is expected that the NN learns features from the new Set B so it is able to generalize better.

It was also interesting to compare the performance of a "short" re-training only with every new set of images that were extracted with training one time from scratch with the full dataset (e.g. in this case Set A + Set B, with 400 images). The output model of training with the full dataset is called *Model C*.

For each model, 80% of the dataset was taken randomly for training and 20% for validation. As the split is random and the dataset small, a certain random variation of the results could be expected. Therefore, to analyze the variance in the result, the process was repeated and the authors trained 10 independent cases for each model (each with a random train-validation set split).

At each step, the 10 cases were evaluated for each model and the one with the highest testing AP was selected to be the starting point of the re-training in the following step. For instance, in Model A the best case is selected and those weights are used as a starting point to re-train the NN for 10 new random cases on set B, generating Model B.

In order to measure the performance of each model, a *testing set* with 100 images, picked from videos taken in different places and times, was created. The focus on the testing set was to maximize the variability in the frames (e.g. light, quality, turbidity in water, depth, etc.). This is designed to simulate a general case and allows a realistic measurement of how well the models generalize.

In figure 6 a sample of each video set is shown. At the end, the best resulting model for each of the species was chosen to process the entire video material from this work, thus creating the first version of the NorFisk dataset.

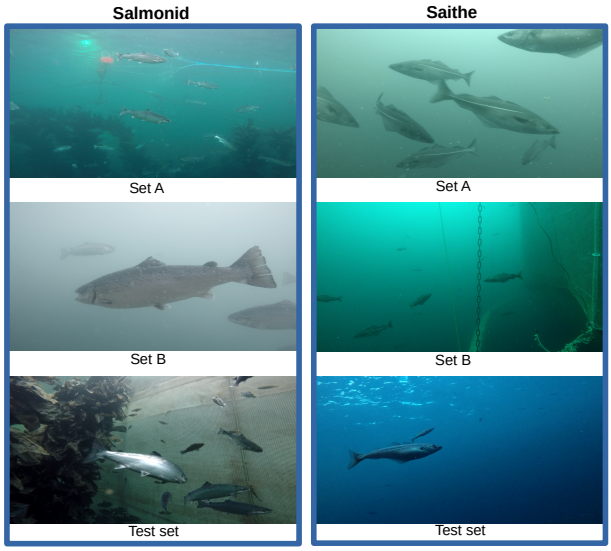

Figure 6: Sample images from the different video sets for each of the species: Salmonids (left column) and Saithe (right column)

Figure 7: Fish farm where video footage for the experiments was collected. View of a salmon cage on the surface (top), wild fish outside the net pens (bottom-left), salmon inside the net pens (bottom-right)

3.2 Data collection

The video footage has been taken in a fish farm at different times on different days between 2017 to 2020. Videos were taken in different weather conditions and at different depths as well. The farm is located near lesund in Norway (see figure 7)

This work started with a collection of 346 videos summing up 58 hours and 56 minutes. The first step was to delete the ones without useful information (e.g. without fish around). This process left a total of 49 hours and 25 minutes of footage. The films were in different resolutions and different orientations, so they were also normalized. A resolution of 1920x1080 pixels was set for all of them. When the video footage was processed, only 1 frame per second was taken in order to have the fish in different positions and avoid repeated images. The recordings contain salmon and trout inside the net pens (which was grouped as salmonids) and most of the wild fish outside the nets are saithe. Videos were taken using GoPro Hero 4, 5, and 8 cameras.

From the 49 hours of video, three hours depict wild fish and the rest only contains farmed fish. Therefore the dataset that is published together with this work

is not balanced and it contains more salmonids than saithe. However, during 2020, more than 150 hours of video footage on wild fish around net pens have been taken and this material will be processed as a future work using the system developed here. The dataset will be updated to include plenty of image data on wild fish.

3.3 System implementation and configuration

An implementation of the neural network's architecture YOLOv3 Redmon and Farhadi (2018) was coded in Python (the original version is written in C) and integrated with the rest of the system. The NN stops the training when it reaches the pre-established maximum number of epochs or when it converges. Convergence is defined as the point where the NN has not improved its loss for five consecutive epochs. When the training stops, the weights that produced the lowest loss are saved. In addition, an adaptive learning rate was used. Every time the loss has not improved for two epochs, the learning rate is divided by 10. Table 1 summarizes the hyperparameters used.

Table 1: Neural Network's hyperparameters.

Training	
IOU threshold	0.5
Learning rate initial value	1e-4 (1e-5 for Model B)
Maximum epochs numb.	100 (12 for Model B)
Min. image input size	288x288
Max. image input size	448x448
Prediction	
IOU threhsold	0.5
NMS threshold	0.4
Class threshold	0.3
Image input sizes	224, 416, 608, 864, 1504

Given the fact that YOLO's weakest point is localization errors, an IOU threshold of 0.5 was established. The initial learning rate was 10^{-4} for models A and C but the starting point was 10^{-5} in Model B. The reason for this is that in Model A the NN has already learned some features. Therefore, in Model B it is only needed to fine-tune this previous model. The maximum number of epochs for Models A and C was 100, although the models converged in an earlier stage in every case. On the other hand, the maximum number of epochs for Model B is 12. This is to avoid a scenario where the NN "memorizes" the new set of images and "forgets" the previous one (this is explained in the next section).

When image extraction is performed, the non-maximum suppression (NMS) threshold is set to 0.4. That means if two bounding boxes have an IOU greater than 0.4 they are considered to belong to the same object. The class threshold is low to enable the NN to capture more objects in an early stage when the model's accuracy is not high (e.g. Model A). This can be increased at the same time the NN gets more accurate in order to avoid false positives. A summary of the hardware and the main libraries used are listed in Table 2. OpenCV is widely used in the IPS while the NN is implemented using Keras embedded in TensorFlow libraries.

Table 2: Hardware and software used for this work.

Hardware	
CPU	Intel Xeon W-2245 (3.9GHz x 16 cores)
GPU	Nvidia Quadro RTX 6000 (24GB, 4608 cores)
RAM	64GB
Software	
OS	Ubuntu 18.04.4
CUDA	10.1
CuDNN	7.6.5
Python	3.7.6
Tensorflow	2.2.0
OpenCV	4.2.0.34

4 Results and discussions

The line of experiments from figure 5 was followed and the three models A, B, and C were created successfully. To analyze the performance, in this section, the process, the loss evolution, and the average precision on validation and testing sets for each model are discussed.

4.1 General overview of the process

The IPS was used to produce set A of 200 images for each species. The input in each case was a set of videos in similar light conditions. First, the filters and parameters of the IPS had to be tuned manually (which takes about 10 minutes) in order to filter fish from the background. Next, the system started processing frames and extracting images automatically and precisely. The processing time of each frame is on the order of milliseconds and this is clearly faster than labeling and annotating images manually.

One interesting point to notice here is that if one desires to install a fixed camera underwater to monitor any fish activity in the field of view, the IPS can be used. Five or six different filters that change automatically for different times of the day (light conditions) might have to be programmed. This can efficiently detect and extract anything that is moving around, of course without labels, and it can be used to acquire images of new species for further labeling and training.

After the IPS created the first minimal dataset of 200 images for salmonids and saithe, the NN was trained to generate model A for each of the species. Validation and testing average precision (AP) for each case was measured. The case with the best testing AP in Model A was selected to continue with the next image extraction. For saithe this was case 9 with a testing AP of 0.73 (see table 3) and case 10 for salmonids with a value of 0.78 (see table 4).

Model A was used to extract set B (200 more images). Set B was used to retrain the selected case of Model A, producing 10 new cases for Model B. In the beginning we let the NN converge alone, setting a limit of 100 epochs. However the testing AP was not improving. An investigation indicated the reason was the NN was learning the new images but forgetting the old ones (set A). This was exactly the opposite of what it was initially aimed. To partially solve this problem the maximum number of epochs was constrained to 12. In this way, the NN starts adjusting the weights towards the new dataset without reaching the point of forgetting the old ones. A better approach that the authors propose, for a future work, is to modify the loss function to include information from the past weights in the training stage. This problem is known as catastrophic forgetting and Kirkpatrick et al. (2017) proposes a method to address it in classification problems.

The authors are proposing to make an iterative process of extracting images and retraining to gain precision and generalization capability. In order to evaluate and compare how this works, a new set of 10 cases (Model C) was trained, from zero, using the whole dataset up to this point (400 images adding Set A and Set B). The performance is discussed in 4.3 and 4.4

4.2 Wrong predictions

The advantage of replacing the IPS for the NN in an early stage is that no filters or parameters tuning is needed. The NN is also independent of light conditions because it extracts features of the fish and does not act on the pixels colors themselves. However, in early stages, the NN is not too accurate since it is trained with few images. Therefore it detects some false positives and also misses some fish. In the case of false positives, the image has to simply be deleted from the dataset. Missing some fish does not cause an issue because the goal is to create a dataset as autonomously as possible. The more the NN is re-trained with new images, the more accuracy and more generalization power it gains and the fewer mistakes it is expected to make.

In this work, for simplicity, the focus is on having one object per image. This has a downside when fish are in dense groups. When this occurs, sometimes no image is extracted because the bounding boxes are overlapped and our system automatically discards those situations. Other times a fish is detected, but not its neighbor. In this case, the ROI of the detected fish will contain part or all of the non-detected fish. This type of images have to be discarded from the dataset as well. Nevertheless, the time consumed to visually inspect and delete a few images is not comparable with the time of annotating all images manually. In future work images with multiple objects will be included.

4.3 Loss evolution

The total loss in each epoch for each case in each model for both species is shown in figure 8. In all cases of Model A and C the training converged before reaching the 100 epochs. Model B does not always converge alone since it was forced to terminate in epoch 12 to avoid forgetting the previous data. Model A converges in a variety of epochs (between 12 and 29 epochs). The loss evolution can also be quite different depending on the data. In the case of saithe the evolution seems to be more consistent for different cases while in salmonids the difference can be up to 2 points between case 4 and case 8, for example.

Model B, on the other hand, which is created by retraining the best case of Model A on the dataset B, presents a much smaller variance. The difference between the different cases can be neglected, and the same will be seen with the testing AP. With respect to the values, in the case of salmonids the loss values for Model B are between 3 and 4 while in Model A they reach values below 2. The reason is that for Model B, the weights of the best testing AP was used as a starting point. This does not necessarily mean the lowest loss value. Since the loss follows the validation set, the result can be quite different when the model is evaluated on the testing set, and that was the case for salmonids. Our best testing accuracy for model A was on case 10, which has a loss of 3.58 and this is what sets the starting point for Model B. For saithe, the best case in Model A was case 9 with a loss value of 2.36. This is why the loss evolution in Model B presents these differences from Model A in the different species. However, the goal is to obtain the most general model and, while the loss exposes the trends, variations, and evolution on the training set, our final metric is the testing AP.

When comparing the results between Models A and B, the results thus far suggest that it is important to make several cases in Model A to have the best possible foundation for Model B since the variance is high for small datasets. However, from that point on, the retraining can be done just one time for each model because the differences can be neglected. This is very

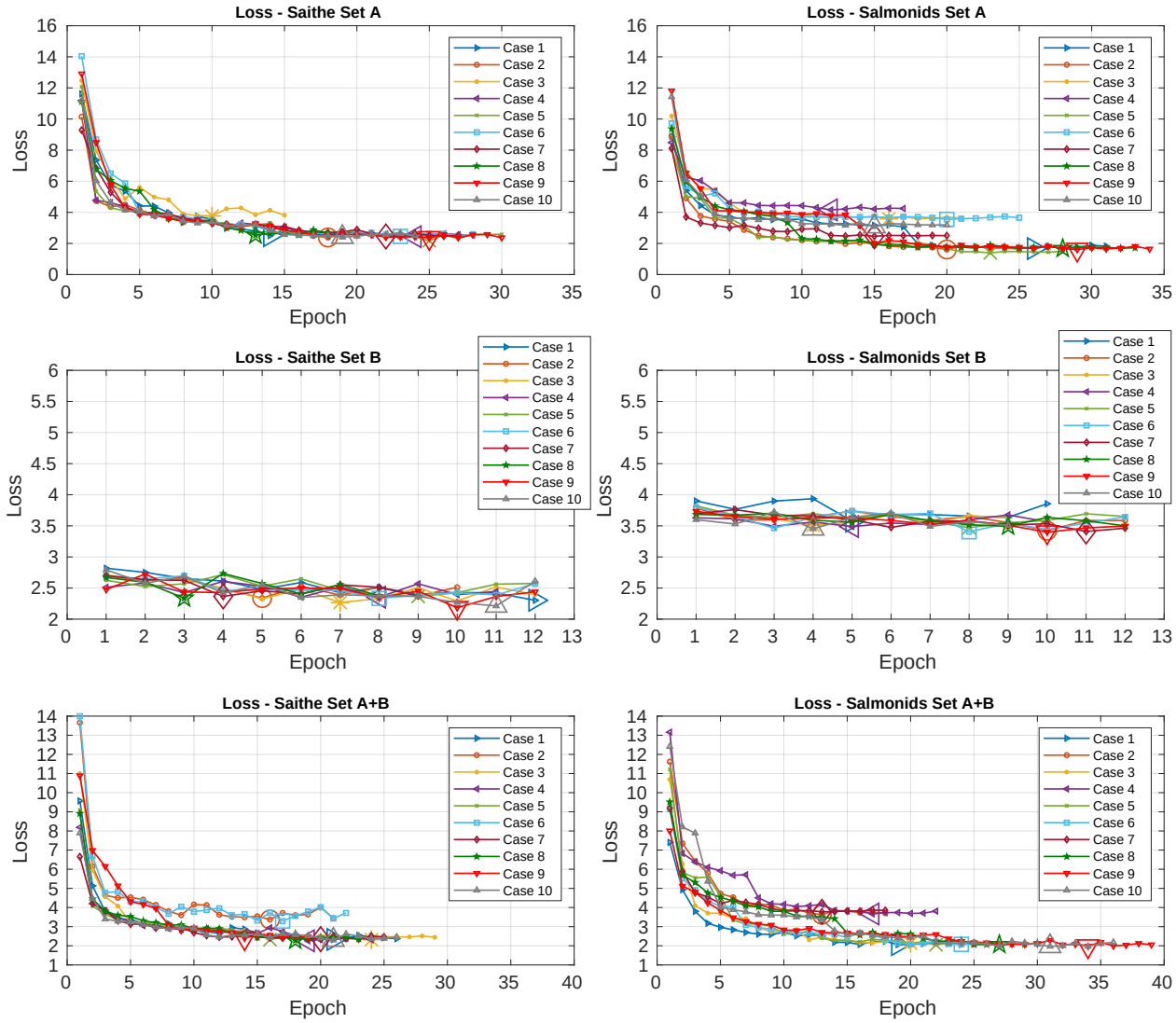

Figure 8: Loss evolution during training of all the models for both saithe (left column) and salmonid (right column). Big markers indicate the position of minimum loss for each case.

important because it means that when a new video or set of videos is processed and extracts a few hundred images, the retraining of the NN can be short enough without sacrificing accuracy.

In the case of Model C, the loss evolution has similar behavior as Model A, with high variance. This might suggest that using 400 images was not very different from using 200. This will of course depend on the variability of the images as well.

4.4 Average Precision

The AP for validation and testing sets for each case of each model are shown in Tables 3 and 4 for saithe and salmonids respectively. The average for each model was also calculated. Lastly, to have an idea of the overfit, the Root Mean Square (RMS) error between validation and testing APs was obtained.

There are several points to analyze and conclude from these results. First of all, as a general overview, it can be seen that validation AP is always higher than testing AP for all models. When one looks at the RMS error, the differences between validation and testing AP are around 10% and 20% for the different species. This is not a surprise, since NNs tend to perform better on the datasets they are trained with. In the early stages this effect might be abrupt given the small number of images. That is why it becomes important to

Table 3: Average precision obtained for different cases and models trained for Saithe. The range of values goes from 0 (for a precision of 0%) to 1 (for a precision of 100%)

Case Number	Model A (Set A)		Model B (Set B)		Model C (Set A + Set B)	
	Validation mAP	Testing mAP	Validation mAP	Testing mAP	Validation mAP	Testing mAP
1	0.7000	0.5673	0.9750	0.7496	0.8461	0.6300
2	0.8200	0.6500	0.9423	0.7500	0.7500	0.7100
3	0.6750	0.4050	0.9189	0.7500	0.7221	0.5300
4	0.5500	0.3000	0.9416	0.7000	0.5269	0.2731
5	0.8328	0.6137	0.9429	0.7000	0.8498	0.7588
6	0.8750	0.5776	0.9455	0.7400	0.7743	0.7052
7	0.9537	0.6843	0.9750	0.7500	0.8000	0.4500
8	0.8750	0.4640	0.9410	0.6600	0.7250	0.6600
9	0.9500	0.7300	0.9455	0.6700	0.9583	0.7900
10	0.8027	0.6351	0.9481	0.6800	0.7640	0.6900
Average	0.8034	0.5627	0.9475	0.7149	0.7716	0.61971
RMS Error	0.2522		0.2352		0.1797	

Table 4: Average precision obtained for different cases and models trained for Salmonids. The range of values goes from 0 (for a precision of 0%) to 1 (for a precision of 100%)

Case Number	Model A (Set A)		Model B (Set B)		Model C (Set A + Set B)	
	Validation mAP	Testing mAP	Validation mAP	Testing mAP	Validation mAP	Testing mAP
1	0.6355	0.6497	0.5514	0.5000	0.8011	0.6120
2	0.6735	0.6777	0.8897	0.8000	0.7732	0.5779
3	0.8359	0.7046	0.9040	0.7992	0.7299	0.5232
4	0.7080	0.7297	0.8963	0.7750	0.7975	0.6091
5	0.7809	0.7209	0.8268	0.7242	0.8232	0.5794
6	0.5449	0.5135	0.8955	0.7992	0.7077	0.5386
7	0.5691	0.3783	0.8269	0.76953	0.5284	0.3906
8	0.6300	0.4324	0.8833	0.7500	0.7000	0.4811
9	0.5261	0.6367	0.8659	0.7750	0.6672	0.5033
10	0.8141	0.7837	0.8769	0.7977	0.5300	0.4051
Average	0.6718	0.62272	0.84238	0.748983	0.70582	0.52203
RMS Error	0.1054		0.0964		0.1869	

have a separate test set in order to have a realistic measurement of the accuracy.

When comparing the RMS error for each type of fish, in Model C for example, it is similar for both species. In the other two models, salmonids have a lower difference between validation and testing AP (i.e. smaller RMS error). This is probably because the salmonid datasets have greater variety in the environment and backgrounds, which makes the training dataset more general. In the saithe datasets, images tend to be more similar. In consequence, the difference between training and validation sets are smaller, and this causes bigger validation AP and bigger differences with respect to the testing AP (i.e. bigger RMS error).

With respect to the testing AP, Model B gives the best average in both species. It shows an increase of about 15% with respect to Model A. This suggests that Model B is actually able to generalize better. Model C, on the other hand (trained with the full dataset) presents a lower value (in average) than model B in both types of fish.

Another interesting fact about Model B is the very low variance across the different cases. Every case shows a very similar result of testing AP. This, again, supports the idea that there is no need to make multiple cases to "select the best one." It is good enough to make just one case when retraining. This enables us to quickly retrain for every new batch of data included in the dataset, speeding up the process without compromising accuracy. In the case of Model C, the same

pattern as in Model A is observed. This might be due to the fact that in the beginning the NN has not been initialized for the type of fish in question and in consequence there is more randomness in the convergence process.

With respect to the AP values for each case, in the tables, the highest testing AP for each model is colored. In the case of salmonids, Model B presents the highest AP in case 3 with a value of 0.79. In the case of saithe, Model B was beaten by Model C with the highest value of 0.79 in case 9. However, this case is also an outlier compared to the rest of the model, as it is almost 20% from the average. In Model B, the highest value for saithe was 0.75 but it is barely 4% off the average.

Overall, in conclusion, there was a clear improvement from Model A to B in both species. The testing AP increases and therefore the NN acquires more generalization capability. This supports the idea that there is no need to train the whole dataset in every extraction (investing a huge amount of time). Instead, a short training with the last few hundreds of images in each extraction can be made in order to incorporate the new information to the NN.

4.5 Full dataset

Last but not least, the best case of Model B was chosen for each type of fish to generate a public dataset. The winners were case 7 for saithe and case 3 salmonids. These models were used to process the entire video material. 3027 images for saithe were obtained and 9487 for salmonids. Each image contains only one fish of the corresponding class. For each image the bounding box image, the labeled image, and the annotation file (cf. figure 3) were obtained. The authors called this dataset *NorFisk Dataset v1.0* and is publicly available at Crescitelli (2020). This dataset can now be used to train neural networks and develop models to efficiently monitor saithe and also wild salmon outside the net pens for example. It can be used not only in fish farm but also in other locations to get a better understanding on migration patterns, stock estimation, etc. At the same, a model can be trained with the full dataset in one class called "fish" and use it to detect other species to be labeled later on and expand the dataset.

5 Conclusion

A new dataset of annotated images corresponding to saithe and salmonids and which can be found at Crescitelli (2020) was introduced.

A semi-automatic approach to create annotated image datasets for fish species in aquaculture environments with little human intervention using video footage as input was also proposed. This method was used to generate the dataset presented in this work. The system uses object recognition and consists of an iterative process of extracting images, growing the dataset and retraining to gain generalization power. The system was put to a series of preliminary tests to evaluate the performance and the results are satisfactory. The accuracy and generalization of the NN increases from one iteration to the next one.

6 Future Work

Since this is the beginning of the work to create a new approach, there are still many things to try and improve in the future. The first feature to add is to include multiple objects (i.e. fish) per image in the dataset. The detection algorithm can be improved to increase the accuracy and decrease the errors in localization. The logic of maximizing the ROIs for every object can also be improved. The loss function should be modified to address the topic of catastrophic forgetting. In this way, The NN can be let to converge in each of the iteration steps without constraining the number of epochs.

More exhaustive experiments should be made to characterize the retraining stages deeply. This could provide answers to questions such as: After how many additional images should one retrain? What is a good trade-off between accuracy and time spent on retraining? An evaluation for many iterations should also be made. Such an evaluation would answer questions such as: When does the NN stop improving the precision (based on iterations or number of images)?. Finally, we leave as future work the update of the published dataset with large amounts images of wild fish surrounding the net pens. For this more than 150 hours of video footage taken during 2020 will be processed.

References

The Nature Conservancy. *Nature*, 1964. 202(4930):337–337. URL https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring/data, doi:10.1038/202337d0.

The OpenCV Reference Manual. 2020. URL https://docs.opencv.org/4.4.0/.

Bjelland, H. V., Fore, M., Lader, P., Kristiansen, D., Holmen, I. M., Fredheim, A., Grotli, E. I., Fathi, D. E., Oppedal, F., Utne, I. B., and Schjolberg, I. Exposed Aquaculture in Norway. In *OCEANS 2015*

- *MTS/IEEE Washington*. IEEE, pages 1–10, 2015. doi:10.23919/OCEANS.2015.7404486.

Boom, B. J., Huang, P. X., Beyan, C., Spampinato, C., Palazzo, S., He, J., Beauxis-Aussalet, E., Lin, S. I., Chou, H. M., Nadarajan, G., Chen-Burger, Y. H., van Ossenbruggen, J., Giordano, D., Hardman, L., Lin, F. P., and Fisher, R. B. Long-term underwater camera surveillance for monitoring and analysis of fish populations. In *Workshop on Visual observation and Analysis of Animal and Insect Behavior (VAIB), in conjunction with ICPR 2012*. Tsukuba Science City, Japan, pages 2–5, 2012. URL http://homepages.inf.ed.ac.uk/rbf/VAIB12PAPERS/boom.pdf.

Crescitelli, A. M. NorFisk Dataset. 2020. doi:10.18710/H5G3K5.

Cutter, G., Stierhoff, K., and Zeng, J. Automated Detection of Rockfish in Unconstrained Underwater Videos Using Haar Cascades and a New Image Dataset: Labeled Fishes in the Wild. In *2015 IEEE Winter Applications and Computer Vision Workshops*. IEEE, pages 57–62, 2015. doi:10.1109/WACVW.2015.11.

Fernandez-Jover, D., Sanchez-Jerez, P., Bayle-Sempere, J. T., Valle, C., and Dempster, T. Seasonal patterns and diets of wild fish assemblages associated with Mediterranean coastal fish farms. *ICES Journal of Marine Science*, 2008. 65(7):1153–1160. doi:10.1093/icesjms/fsn091.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pages 580–587, 2014. doi:10.1109/CVPR.2014.81.

Grotli, E., Vagia, M., Fjerdingen, S., Bjerkeng, M., Transeth, A., Svendsen, E., and Rundtop, P. Autonomous Job Analysis: a method for design of autonomous marine operations. In *OCEANS 2015 - MTS/IEEE Washington*. IEEE, pages 1–7, 2015. doi:10.23919/OCEANS.2015.7401888.

Hossain, E., Alam, S. M. S., Ali, A. A., and Amin, M. A. Fish activity tracking and species identification in underwater video. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, pages 62–66, 2016. doi:10.1109/ICIEV.2016.7760189.

Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.-P., Planqué, R., Rauber, A., Palazzo, S., Fisher, B., and Müller, H. LifeCLEF 2015: Multimedia Life Species Identification Challenges. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9283, pages 462–483. 2015. doi:10.1007/978-3-319-24027-5_46.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017. 114(13):3521–3526. doi:10.1073/pnas.1611835114.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. 8693 LNCS(PART 5):740–755. URL http://arxiv.org/abs/1405.0312.

Liu, S., Li, X., Gao, M., Cai, Y., Nian, R., Li, P., Yan, T., and Lendasse, A. Embedded Online Fish Detection and Tracking System via YOLOv3 and Parallel Correlation Filter. In *OCEANS 2018 MTS/IEEE Charleston*. IEEE, pages 1–6, 2018. doi:10.1109/OCEANS.2018.8604658.

Mandal, R., Connolly, R. M., Schlacher, T. A., and Stantic, B. Assessing fish abundance from underwater video using deep neural networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, volume 2018-July. IEEE, pages 1–6, 2018. doi:10.1109/IJCNN.2018.8489482.

Pelletier, S., Montacir, A., Zakari, H., and Akhloufi, M. Deep Learning for Marine Resources Classification in Non-Structured Scenarios: Training vs. Transfer Learning. In *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, volume 2018-May. IEEE, pages 1–4, 2018. doi:10.1109/CCECE.2018.8447682.

Rathi, D., Jain, S., and Indu, D. S. Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning. *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*, 2018. pages 1–6. doi:10.1109/ICAPR.2017.8593044.

Rauf, H. T., Lali, M. I. U., Zahoor, S., Shah, S. Z. H., Rehman, A. U., and Bukhari, S.

A. C. Visual features based automated identification of fish species using deep convolutional neural networks. *Computers and Electronics in Agriculture*, 2019. 167(September):105075. doi:10.1016/j.compag.2019.105075.

Raza, K. and Hong, S. Fast and Accurate Fish Detection Design with Improved YOLO-v3 Model and Transfer Learning. *International Journal of Advanced Computer Science and Applications*, 2020. 11(2):7–16. doi:10.14569/IJACSA.2020.0110202.

Redmon, J. Darknet: Open Source Neural Networks in C. 2016. URL http://pjreddie.com/darknet/.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2016-Decem. IEEE, pages 779–788, 2016. doi:10.1109/CVPR.2016.91.

Redmon, J. and Farhadi, A. YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2017-Janua:6517–6525. doi:10.1109/CVPR.2017.690.

Redmon, J. and Farhadi, A. YOLOv3: An Incremental Improvement. *Tech report*, 2018. URL http://arxiv.org/abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 39(6):1137–1149. doi:10.1109/TPAMI.2016.2577031.

Salman, A., Jalal, A., Shafait, F., Mian, A., Shortis, M., Seager, J., and Harvey, E. Fish species classification in unconstrained underwater environments based on deep learning. *Limnology and Oceanography: Methods*, 2016. 14(9):570–585. doi:10.1002/lom3.10113.

Salman, A., Siddiqui, S. A., Shafait, F., Mian, A., Shortis, M. R., Khurshid, K., Ulges, A., and Schwanecke, U. Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system. *ICES Journal of Marine Science*, 2019. (February). doi:10.1093/icesjms/fsz025.

Shah, S. Z. H., Rauf, H. T., IkramUllah, M., Khalid, M. S., Farooq, M., Fatima, M., and Bukhari, S. A. C. Fish-Pak: Fish species dataset from Pakistan for visual features based classification. *Data in Brief*, 2019. 27. doi:10.1016/j.dib.2019.104565.

Siddiqui, S. A., Salman, A., Malik, M. I., Shafait, F., Mian, A., Shortis, M. R., and Harvey, E. S. Automatic fish species classification in underwater videos: Exploiting pre-trained deep neural network models to compensate for limited labelled data. *ICES Journal of Marine Science*, 2018. 75(1):374–389. doi:10.1093/icesjms/fsx109.

Tamou, A. B., Benzinou, A., Nasreddine, K., and Ballihi, L. Transfer Learning with deep Convolutional Neural Network for Underwater Live Fish Recognition. In *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*. IEEE, pages 204–209, 2018. doi:10.1109/IPAS.2018.8708871.

Tuya, F., Sanchez-Jerez, P., Dempster, T., Boyra, A., and Haroun, R. J. Changes in demersal wild fish aggregations beneath a sea-cage fish farm after the cessation of farming. *Journal of Fish Biology*, 2006. 69(3):682–697. doi:10.1111/j.1095-8649.2006.01139.x.

Utne, I., Schjølberg, I., and Holmen, I. Reducing risk in aquaculture by implementing autonomous systems and integrated operations. In *Safety and Reliability of Complex Engineered Systems*, pages 3661–3669. CRC Press, 2015. doi:10.1201/b19094-481.

Villon, S., Mouillot, D., Chaumont, M., Darling, E. S., Subsol, G., Claverie, T., and Villéger, S. A Deep learning method for accurate and fast identification of coral reef fishes in underwater images. *Ecological Informatics*, 2018. 48(September):238–244. doi:10.1016/j.ecoinf.2018.09.007.

Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1. IEEE Comput. Soc, pages I–511–I–518, 2001. doi:10.1109/CVPR.2001.990517.

Wang, X., Ouyang, J., Li, D., and Zhang, G. Underwater Object Recognition Based on Deep Encoding-Decoding Network. *Journal of Ocean University of China*, 2019. 18(2):376–382. doi:10.1007/s11802-019-3858-x.

Zhang, L., Li, W., Liu, C., Zhou, X., and Duan, Q. Automatic fish counting method using image density grading and local regression. *Computers and Electronics in Agriculture*, 2020. 179(October):105844. doi:10.1016/j.compag.2020.105844.