# Computer vision non-invasive individual fish identification based on skin patterns

Juan Aguilera[1], Felipe Villenas[1]
[1]M.Sc. Electronic Engineering, Universidad Técnica Federico Santa María

***Abstract*** **The growth of salmon farming in countries such as Chile has increased interest in scientific and technological development, seeking new ways to improve both production and fish welfare. The objective of this research is to confirm the feasibility of the technologies implemented in the state of the art. In previous articles, the identification of Atlantic salmon (Salmo salar) in an automatic and non-invasive way using visible characteristics in the body, is performed under ideal conditions, so this work replicates the proposed methods using a CNN and HOG as a feature descriptor, and evaluates a dataset consisting of Salmon in underwater conditions. Also, to further improve the results from the HOG approach, we implement a UNet to pre-process the datasets and remove the background information leaving only the fish skin pattern. The average classification accuracy obtained was an 80.4%, 70.9% and 74.2% for the CNN, HOG and UNet+HOG approaches, respectively. Thus, we can conclude that the proposed identification techniques also work for a sub-aquatic setting.**

***Index Terms*****—Convolutional neural networks, Histogram of oriented gradients, Computer vision, Individual identification, Non-invasive identification, Skin pattern.**

## I. Introduction

THE production in aquaculture has been increasing in recent years. Salmon farming is particularly prominent in the southern zone of Chile [1]. The rise in fish farming leads to the automation of many tasks in the cultivation process, such as feeding, fish sampling, and controlling fish welfare and diseases [2]. The automation of these tasks help the fish farmers to monitor and control the fish cultivation process. One of the most critical tasks is the early disease detection and prediction of outbreaks to safeguard the livestock of fish. Thus, the automated identification of individual fish is of the utmost importance [3].

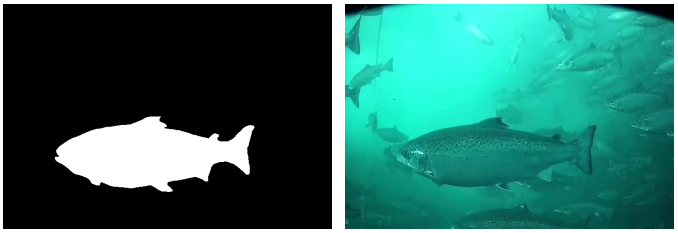
Photographic or video based identification methods are common in the field of agriculture and wildlife to ensure the population well-being and estimating the size of the population. These methods use visible patterns of the animal for species or individual identification [4].

In the aquaculture section, computer vision techniques are often used to identify the fish species. The first fish recognition study, was focused on fish species identification by analysis of their shape in digital images [5]. The fish body shape was computed from images of different fish species and geometrical shape descriptors were used to obtain a 90% accuracy in species classification. More recent techniques [6] have used convolutional neural networks (CNN) where the algorithm was trained with 900,000 images for nine species, and it was able to obtain an accuracy of 94.9%. In [7] they also focus on species classification testing multiple methods for feature extraction, such as, support vector machine (SVM), k-nearest neighbor (KNN), and decision trees. The authors were able to obtain an F-score of 88% in this work. The work in [8] details an extensive review of many different machine learning methods applied to aquaculture for fish identification and classification.

The identification of fish individuals of the same species is more challenging because of the similarity among fish. One of the basic approaches of individual fish identification from the same species is marking and tagging, however, these methods are invasive and can lead to negative effects to the fish. Thus, non-invasive identification is important to avoid said negative effects. There have not been many works for individual fish identification where the process is automated. Some studies were based in manual fish identification where experts performed the recognition for randomly selected individuals [9], [10].

There has also been works where the individual fish identification is automated. In [11] they perform the identification on five individuals of zebrafish using the HSV colorspace obtaining an accuracy of 99% for the five samples. Some more deep works, such as the one in [2], where they use a sample of 68 fish in total and after segmenting each individual fish, they use the *histogram of oriented gradients* (HOG) as a feature extraction method where the region of interest (ROI) was the stripes located in the body of the fish, after this process, a KNN-based classifier is used to obtain the ID of each individual fish. The authors report a 100% accuracy classification using a combination of the ROI mentioned before and a HOG descriptor with 4 pixels cell size. Another work also utilizes computer vision for individual fish identification based on the skin dot patterns that the fish present [12]. In this case they used a sample of 328 farmed Atlantic salmon for the experiment and the ROI was the area between the eye of the fish and its upper fin. Then for the skin dot detection they tested two different methods. The first one uses a CNN to classify each fish in one of two classes; dots or no-dots. The second approach is the same as the previous work of using an HOG for feature extraction. The authors also report that they were able to obtain a 100% classification accuracy with both methods employed to their respective dataset.

On the previous two works mentioned, however, the dataset was composed of images of the fish taken out of the water and on top of a green screen, which in turn results in very ideal lighting conditions for the posterior segmentation and identi-



(a) Binary image. (b) Original image.

**Figure 1:** Image from the underwater dataset.

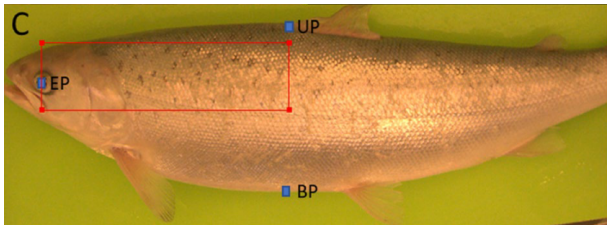



**Figure 2:** Fish with localized points **EP**, **UP** and **BP**; Red rectangle represents the ROI.

fication. These conditions are very different from pictures of fish taken underwater. Therefore, one of the contributions of the work in this project, is to verify the performance of the algorithms of [2], [12] by using a dataset composed of different salmon in underwater conditions, and compare the results with the results of the authors of said articles. Furthermore, to improve the results, we propose the use of deep learning for individual fish identification for both feature extraction and classification.

## II. Materials and Methods

### A. Datasets

To compare the results, we created or own dataset using monitoring videos from salmon farms. This dataset consists of a set of 11 different underwater salmon followed for a number of frames. We selected from 20 to 10 images for every entity, and divided them on train and test data. From the selected images, we obtained the binaries as shown in Fig. 1. Data augmentation on the train data, using mostly small color and rotation changes, allow us to get a large dataset of different individual fish.

### B. Algorithms

#### 1) ROI Selection

For the underwater salmon dataset, the ROI used on all the algorithms to compare is defined in [12] as the area between the eye of the fish and its upper fin, which was extracted using the following method:

The detected and segmented fish was rotated using an estimated ellipse to compensate for rotation of the fish in the image. The approximate area of the upper fin was estimated to be in the region of 3/8 to 5/8 of the length of the detected object (without the tail fin). This area was used to search for the beginning of the upper fin. The segmented fish border of the area was divided into the left and right half. The line was fitted to the left and right pixels and the intersection was taken as the beginning of the upper fin. The localized position was marked as **UP**. The approximate area of fish eye was estimated between 0 and 1/3 of the **UP** position (in horizontal axis). The exact area of the fish eye was detected by the thresholding of the grayscale image of the selected area. All pixels with intensities lower than 20 were marked as eye pixels. The eye position **EP** was calculated as the centroid of the eye pixels. The belly point **BP**, which represents the height of the fish at the horizontal position of **UP**, was detected as the point at the border of the fish belly. This point was used to determine the height of the ROI. Based on the determined points **UP**, **BP** and **EP**, the ROI was selected in the image as the area between **EP** and **UP** in the horizontal direction and between **UP** + (**BP**-**UP**)/20 and **BP**/2 in the vertical direction. See an example of ROI in Fig. 2.

**Table I:** Summary of the CNN model.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Sequential | (None, 200, 100, 3) | 0 |
| Rescaling | (None, 200, 100, 3) | 0 |
| Conv2D | (None, 200, 100, 8) | 224 |
| MaxPooling2D | (None, 100, 50, 8) | 0 |
| Conv2D | (None, 100, 50, 16) | 1168 |
| MaxPooling2D | (None, 50, 25, 16) | 0 |
| Conv2D | (None, 50, 25, 32) | 18464 |
| MaxPooling2D | (None, 25, 12, 32) | 0 |
| Dropout | (None, 25, 12, 32) | 0 |
| Flatten | (None, 9600) | 0 |
| Dense | (None, 1152) | 11060352 |
| Dense | (None, 11) | 12683 |

#### 2) CNN approach

Identification based on the dot's exact position on the fish skin in the underwater dataset, based on [12], was simplified using a single CNN that receives as an input the ROI selection scaled as an image of 200x100 pixels of 3 channels.

**CNN architecture**: The CNN that was used for ROI identification is described as follows:

- The network performs the classification into 11 classes: Salmon1, Salmon2, Salmon3, ..., Salmon11.
- The network architecture contains five trained layers: three convolutional and two fully connected layers.
- The **ReLU** activation function was used for convolutional layers.
- The **Softmax** activation function was used for the classification layer.
- **Max pooling** layers were used between the convolutional layers.

The training was performed with Keras & Tensorflow Deep learning python library. The architecture of the network is shown in Tab. I and some training examples are shown in Fig. 3.

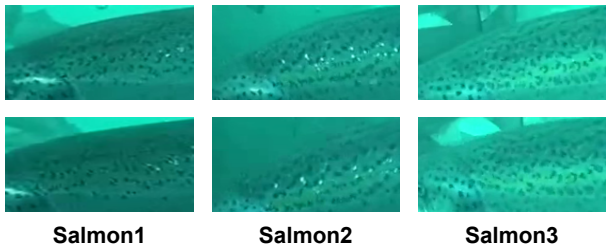



**Figure 3:** Example of data for CNN training.

*3) HOG approach*

Histogram of oriented gradients (HOG) [13] is a feature descriptor mainly used for object detection in computer vision. The HOG descriptor codes the gradients of the image, which mainly represent the edges in the image, and thus, it is also a good texture descriptor invariant to illumination changes.

The procedure for calculating an HOG vector is described as follows:

1) Calculate the gradient of the image. This is the magnitude and direction in both the horizontal and vertical direction, for example, with the Sobel operator.
2) Then, the image is divided in cells of size $8 \times 8$ and an histogram is calculated for each patch across the image.
3) The bins of the histogram are divided according to the angle of the direction of the gradient, from $0^o$ to $180^o$. Thus, the histogram contains 9 bins corresponding to angles $0^o$, $20^o$, ..., $160^o$. Then, the value from the magnitude gradient is stored depending on its angle value from the direction gradient.
4) Now, a region of $16 \times 16$ is selected. This region contains 4 histograms due to its $8 \times 8$ smaller blocks. This 4 histograms are concatenated to form a $36 \times 1$ vector and now is normalized by its magnitude, or equivalently, by diving by its norm-2.
5) Finally, all of the $36 \times 1$ histogram vectors from the $16 \times 16$ cells are concatenated across the whole image to obtain the final histogram of gradient vector.

The identification approach using the HOG descriptor is the following:

– Calculate the feature vector for the pattern from the ROI of the known fish, as shown in Fig. 4.
– Scanning of ROI for the unknown fish.
– Calculation of the best similarity of the HOG vectors between the reference and unknown fish.
– Classification of the unknown fish into one of the known IDs.

The distance between the two HOG vectors from the known and unknown fish is calculated as the norm-2 of the subtraction the two vectors. Then, the similarity is determined by the minimal distance between the known HOG vector and all the HOG vectors generated by scanning over the unknown fish ROI. The authors also tested different settings of the parameters of the HOG feature calculation, such as, the cell size, resolution of the normalization image and the offsets Offx and Offy (see Fig. 5).

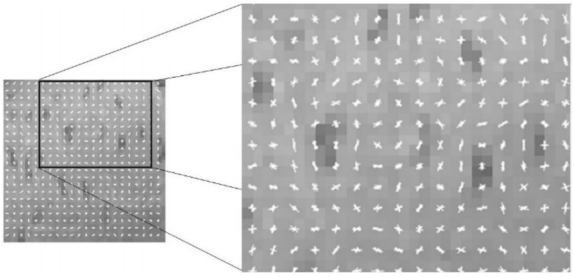


**Figure 4:** Visualization of HOG descriptor for the dots pattern. The orientation of the edges is represented in the large picture coded by the HOG descriptor.

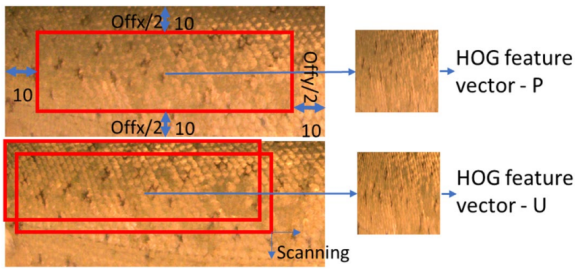



**Figure 5:** Similarity measure of two fish skin patterns using HOG feature descriptor. At the top (P) is the image of the identified fish and its HOG feature vector. Then, at the bottom (U) is the image of the unknown fish. The subpart of the pattern is selected from the image repeatedly by scanning over the image and then the HOG feature vector is calculated to establish the best similarity between the P and U subset.

*4) UNet + HOG approach*

To improve the HOG approach, we add a pre-processing step using a UNet architecture for the fish skin pattern segmentation in the ROI images. The UNet results are then used in the same way as described in the HOG approach, where the classification is made according to the histograms distance.

**UNet architecture**: The UNet [14] is a fully convolutional neural network developed for image semantic segmenteation. The UNet that was used for the dot segmentation is described as follows:

· The network performs the classification into 2 classes: Dot and Background.
· The network architecture contains 24 trained convolutional layers.
· The **ReLU** activation function was used for convolutional layers.
· The **sigmoid** activation function was used for the classification layer.
· **Max pooling** layers were used between the first half of convolutional layers as a down sampling.
· **Up sampling** layers were used between the second half of convolutional layers.

The training was performed with Keras & Tensorflow Deep learning python library. The architecture of the network is shown in Fig. 6.

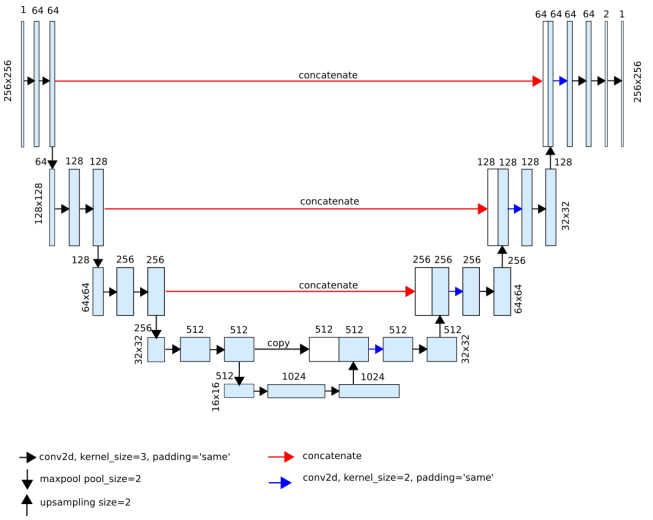



**Figure 6:** UNet model used for this approach, with input and output shape (256,256,1).

## III. Performance Metrics

Since this is a problem of classification and we are not going to focus on the segmentation part of the process, the metrics to test the performance of each algorithm are going to be the standard ones; *precision*, *recall* and *F-score*. Each of them is briefly explained here.

– *Precision:* is the ratio between the number of the algorithm's successful identifications, or true positives (TP), and the total number of positive identifications stated by the model, both present and not present in the groundtruth, where the later is denominated false positives (FP).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

– *Recall:* corresponds to the ratio of TP and the total number of positive identifications stated by the model and the groundtruth. The results stated as negative by the algorithm, but positive in the groundtruth are called false negatives (FN).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

– *F-score:* is a mix of the previous two metrics and thus represents the balance between precision and recall of the model.

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

In the context of our problem a TP would be a correctly identified fish, which means recognizing the individual fish in a different situation. On the other hand, a FP would be classifying the fish as the same individual but in reality is a different fish. Lastly, a FN is mistakenly classifying the fish as a different one, when it was the same individual.

These metrics were chosen since they are the standard metrics for a classification problem and are also the ones that are employed in the referenced works. Thus, allowing us to directly compare the results with the ones from the authors.

**Figure 7:** Example of ROI selection for one of the fishes in the dataset.

## IV. Results

In this section we report the results obtained from implementing the algorithms described in the previous section. All of the algorithms were implemented in the Python language using dependencies from mainly the *OpenCV*, *TensorFlow* and *scikit-learn* libraries.

### A. ROI Selection

The same region of interest described in the previous section was selected from the dataset images. First, a function was used to obtain the minimum rectangle area for the binary mask, which also returned the angle of the rectangle that was used later to get the rotation matrix for the image. The resulting bounding box for the fish is shown in red in Fig. 7.

For the eye position EP, the first 15-20% of the facing direction was used as sub region of interest. In this region, the image is converted to grayscale and then an inverted binary thresholding is applied to obtain the pixels corresponding to the eye, which is darker than the neighbouring region. After that, the centroid is calculated in the threshold image to get the EP coordinates. For the upper fin position UP, the binary mask was used to construct two straight lines with different slopes around the upper fin region in where the intersection of both lines was estimated as the UP coordinates. And lastly, the belly point BP was estimated at 65% of the distance from UP to the bottom border in a straight line.

Fig. 7 shows an example of the ROI selection for a sample image. The EP, UP and BP points are marked in green, whereas the blue rectangle corresponds to the final ROI which is then extracted as a new image for the identification algorithms.

### B. CNN approach

For this approach, we use the model described in section II-B2. For the training process, we use the following parameters:

– Optimizer: *Adam*
– Batch size: 32
– Epoch: 500
– Drop out: 0.6

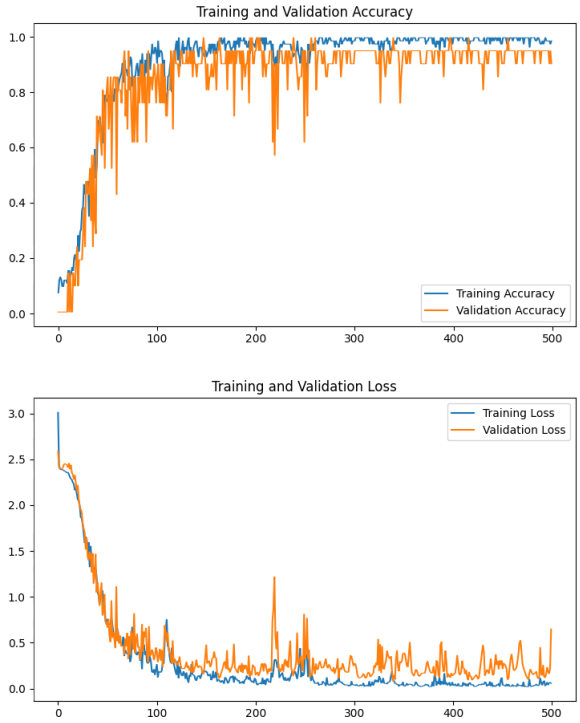



**Figure 8:** Accuracy and loss on train and validation process over 500 epochs.

The dataset for the training process was extracted using the algorithm mentioned in II-B1, where we obtain 107 images for test and validation that are in addition passed through a data augmentation process that consists of small changes in brightness, rotation, zoom, and noise of the images. This augmentation is performed using additional layers on the input of the model using *Keras Sequential* methods. We also created a smaller dataset, of 47 examples, only for test, without data augmentation.

The training process results are shown in Fig. 8. With the saved weights of the training process, we then obtain the confusion matrix for the train and test datasets as shown in Fig. 9 and 10 respectively. The performance metrics described in section III are shown in Tab. II, where *support* is the number of images that belong to the class being tested, with a total of 46 examples tested. From those results, we observe that the majority of the classes in the red have f1-score above 80% and the global accuracy of the model was of 80%.

Training the model with the 106 images with the parameters aforementioned had a runtime of almost 3 minutes, and evaluation of the 47 test examples had a runtime of less than 10 seconds.

All of those metrics and the train of the model were obtained using a computer with the following resources:

- GPU: NVIDIA GeForce RTX 2060 with Max-Q Design (6GB)
- CPU: Intel(R) Core(TM) i7.10750H
- RAM: 32 GB 2933 MHz

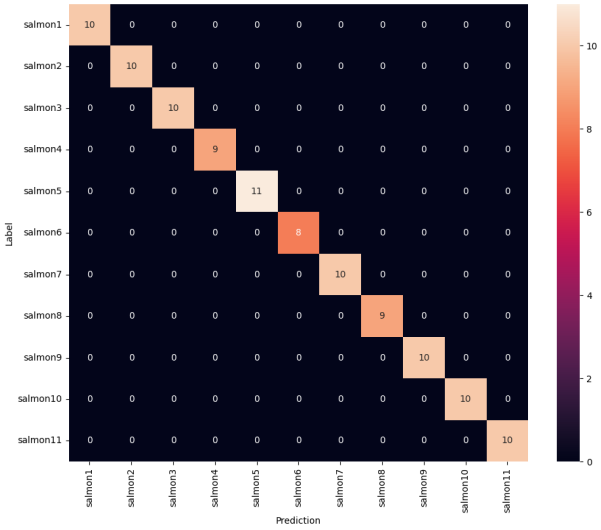



**Figure 9:** Confusion matrix on the train dataset.

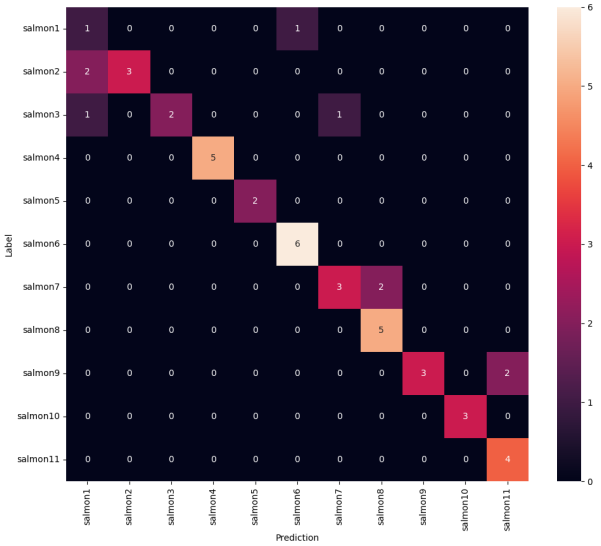



**Figure 10:** Confusion matrix on the test dataset.

**Table II:** Performance metrics for test dataset using the CNN approach.

| Class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Salmon1 | 0.250 | 0.500 | 0.333 | 2 |
| Salmon2 | 1.000 | 0.600 | 0.750 | 5 |
| Salmon3 | 1.000 | 0.500 | 0.667 | 4 |
| Salmon4 | 1.000 | 1.000 | 1.000 | 5 |
| Salmon5 | 1.000 | 1.000 | 1.000 | 2 |
| Salmon6 | 0.857 | 1.000 | 0.923 | 6 |
| Salmon7 | 0.750 | 0.600 | 0.667 | 5 |
| Salmon8 | 0.714 | 1.000 | 0.833 | 5 |
| Salmon9 | 1.000 | 0.600 | 0.750 | 5 |
| Salmon10 | 1.000 | 1.000 | 1.000 | 3 |
| Salmon11 | 0.667 | 1.000 | 0.800 | 4 |
| **accuracy** | **-** | **-** | 0.804 | 46 |

**Table III:** Performance metrics for test dataset using the HOG approach.

| Class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Salmon1 | 0.778 | 0.700 | 0.737 | 10 |
| Salmon2 | 0.875 | 0.538 | 0.667 | 13 |
| Salmon3 | 0.600 | 0.500 | 0.545 | 12 |
| Salmon4 | 0.714 | 0.833 | 0.769 | 12 |
| Salmon5 | 1.000 | 0.727 | 0.842 | 11 |
| Salmon6 | 0.600 | 0.750 | 0.667 | 12 |
| Salmon7 | 1.000 | 0.538 | 0.700 | 13 |
| Salmon8 | 0.786 | 0.917 | 0.846 | 12 |
| Salmon9 | 0.750 | 0.692 | 0.720 | 13 |
| Salmon10 | 0.458 | 1.000 | 0.629 | 11 |
| Salmon11 | 0.800 | 0.667 | 0.727 | 12 |
| **accuracy** | - | - | 0.710 | 131 |

### C. HOG approach

For the HOG approach we used the same datasets combined, and selected two images per fish to obtain the template histograms, the same as described in [12]. The rest of the images were used as a testing set to evaluate the performance of the algorithm.

The ROI selection to obtain the known HOG feature vector (see Fig. 5) was calculated in the same way as described before, in where the dimensions that gave us the best result was a rectangular region of 180×80, and an offset of 16 pixels from the left and top.

The rest of the parameters for the HOG calculation that led to the best results were: 10 orientations, a cellsize of 2×2, 8×8 pixels per cell, and a normalized size of 64×64 for a grayscale image of the ROI. In the same way, for each salmon, the two images selected for the template HOG feature vectors were chosen according to the ones that returned the highest F-score for every class individually. Finally, when comparing one salmon image with the two HOG templates the one that had the minimum distance was chosen as candidate, and subsequentially the class that returned the lowest norm-2 value was used as the predicted class.

The metrics obtained in this approach are described in Tab. III, and the corresponding confusion matrix is shown in Fig. 11. It can be observed that the lowest F-score is 54.5% for Salmon4 and the highest 84.6% for Salmon8. Also, the global accuracy was 71% which is well within the range reported in [12], where the lowest accuracy reported with the HOG descriptor was 36.6%.

The entire process of calculating the HOG templates for the 11 classes (22 images), and the classification of the 131 test images had a runtime of less than 5 seconds in a AMD Ryzen(TM) 5 6-Core CPU.

### D. UNet + HOG approach

For the UNet approach, we use the model described in section II-B4. For the training process, we use the following parameters:

– Optimizer: *Adam*

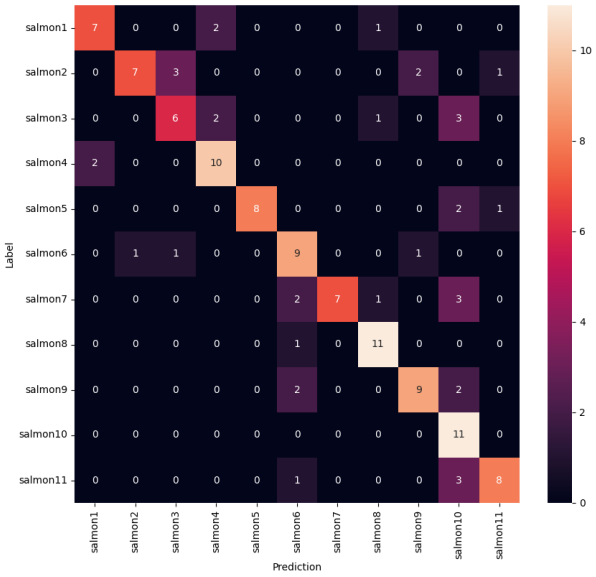



**Figure 11:** Confusion matrix using the HOG approach.


**Figure 12:** Example of data fot the UNet training.

– Steps per epoch: 300
– Epoch: 100
– Drop out: 0.5

The dataset for the UNet approach was extracted using the algorithm mentioned in II-B1 where we obtain 258 images, of which 33 were used for training, and the 225 remaining for testing. The dataset images were reshaped to (256,256,1) due to the input shape of the UNet architecture. For the groundtruth images, we used an adaptive thresholding obtaining a binary mask that was then fine-tuned by hand. An example of the reshaped ROI, the binary mask and the output of the UNet once trained is shown in Fig. 12.

For the HOG classification stage, some of its parameters were changed to adjust to the new input image size. In this case, the parameters that led to the best results were: 10 orientations, a cellsize of 8×8, 16×16 pixels per cell, and a normalized size of 128×128. Also, a subset of the binary images (see Fig. 12) was used as the template images to match the UNet output. In the same way as before, 2 images per class were chosen according to the ones that returned the highest F1-Score.

Once we train the UNet model with the parameters described earlier, we obtain an average *intersection over union* (IoU) of 73% over the 225 test results compared to their respective binary masks. We then obtain the confusion matrix for test dataset, as shown in Fig. 13, where we can observe that the majority of the data is concentrated in the main diagonal

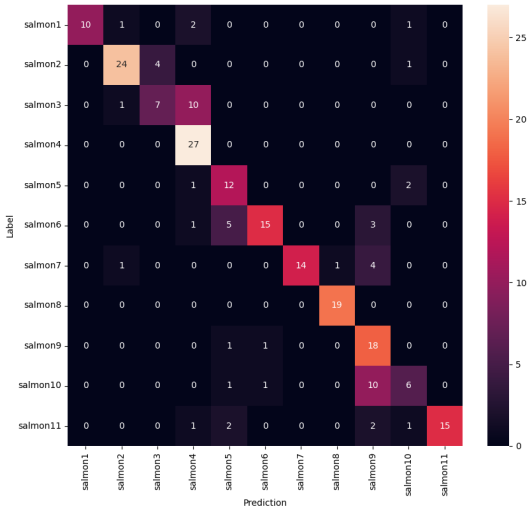



**Figure 13:** Confusion matrix using the UNet+HOG approach.

**Table IV:** Performance metrics for test dataset using the HOG+UNet approach.

| Class | precision | recall | f1-score | support |
|---|---|---|---|---|
| Salmon1 | 1.000 | 0.714 | 0.833 | 14 |
| Salmon2 | 0.889 | 0.828 | 0.857 | 29 |
| Salmon3 | 0.636 | 0.389 | 0.483 | 18 |
| Salmon4 | 0.643 | 1.000 | 0.783 | 27 |
| Salmon5 | 0.571 | 0.800 | 0.667 | 15 |
| Salmon6 | 0.882 | 0.625 | 0.732 | 24 |
| Salmon7 | 1.000 | 0.700 | 0.824 | 20 |
| Salmon8 | 0.950 | 1.000 | 0.974 | 19 |
| Salmon9 | 0.486 | 0.900 | 0.632 | 20 |
| Salmon10 | 0.545 | 0.333 | 0.414 | 18 |
| Salmon11 | 1.000 | 0.714 | 0.833 | 21 |
| **accuracy** | **-** | **-** | 0.742 | 225 |

and it is in general much less spread compared to the original HOG approach. The performance metrics described in section III are shown in Tab. IV. From those results, we observe an increase of 4 percentage points in the global accuracy compared to the solo HOG approach.

Training the model with the 33 images with the parameters aforementioned had a runtime of almost 120 minutes, and evaluation of the 225 test examples had a runtime of less than 15 seconds.

All of those metrics and the train of the model were obtained using a computer with the following resources:

- GPU: NVIDIA GeForce RTX 2060 with Max-Q Design (6GB)
- CPU: Intel(R) Core(TM) i7.10750H
- RAM: 32 GB 2933 MHz

The summary of the results, and the comparison to the results obtained in [12], is shown in Tab. V. The left column results show a range of percentage due to the authors using different subsets of their dataset for training and testing. In general, the results reported in this work are favorably comparable due to the relatively high accuracy in both methods.

**Table V:** Comparison of the results obtained in the referenced work and in this work for the two identification techniques.

| | **Accuracy** % [12] | **Accuracy** % (This work) |
|---|---|---|
| CNN | 70 - 100 | 80.4 |
| HOG | 36.6 - 100 | 70.9 |
| HOG+UNet | - | 74.2 |

## V. Discussion

It can be seen from the results that the CNN method obtains a higher accuracy in comparison to the HOG approach. This is true for both the results obtained in this project, and the results reported in the referenced work. This can also be seen in the confusion matrices, where the CNN confusion matrix is mostly contained in the principal diagonal, which represents the true positives, and has very few elements outside of it. Whereas, the HOG confusion matrix is much more spread due to the algorithm getting more confused with the classification of the salmon. The inclusion of the UNet, however, managed to improved the results after doing the classification with the HOG algorithm. Overall the combination of these two methods result in a higher accuracy than the HOG only approach, but it is still lower compared to the CNN method. This can also be seen in the confusion matrix which is much more close to the CNN one and not as spread compared to the HOG one.

One of the key differences in this work is the nature of the dataset. The dataset of the authors consisted of much higher resolution images of salmon out of the water on top of a green background, and some of them underwater but also with a green background. Due to the nature of this controlled dataset, where the fish are also most of the time in the same position, the authors were able to obtain a 100% accuracy in some cases. In our case, the dataset used was from fish in real sub-aquatic conditions and using the same identification algorithms we achieved a relatively comparable accuracy. A second difference is the number of individual salmon used in both datasets. The work reported by the authors had 328 different fish, while ours only had 11.

A disadvantage of developed CNN method is scalability. Since we used 11 classes to train the CNN, it has a final dense layer with 11 outputs, which represents the probability of each class. If we want to increase the number of salmon to identify, or we want to test the model in an environment where we do not know the total number of salmon, the already trained model would not be able to identify more fish. Thus, we would have to modify the number of outputs of the final layer which in turn would imply to re-train the model, which may not be desirable due to the large time that it can take. To some extent, this would not be an issue in the HOG approach since a new template histogram for a new individual can be calculated in runtime, and the general execution time is much lower in comparison.

## VI. Conclusions

From the results in this work, we demonstrate that, the methods used in the state of the art can be adapted for the detection of salmon under non-lab conditions. We demonstrated that, although not implementing the same CNN approach as the state of the art, we were capable of identifying different individual salmon using a relative small network.

Due to those results, we see the possibility to scale the project. The most challenging problems, is to find an architecture that is capable of learning and adapting from thousands of individual examples over time and growing of population without the need of changing or retraining it.

For this purpose, we decided that the HOG approach is a better fit since it does not have a need for retraining every time a we to add a new class or salmon to the identification algorithm. The addition of the UNet to pre-process the data also showed some promise since we obtained a higher accuracy with the combination of both methods.

To further improve these results, there are a couple of things that can be made. First we can train the UNet model with more data to obtain a more accurate fish skin segmentation. Also, some degree of error is introduced by doing a semi-automated creation of the groundtruth for out model compared to if we were to label the fish skin dots by hand.

In terms of the algorithm performance, a more sophisticated search method can be implemented to accelerate this step in the HOG classification stage. One way to achieve this could be through the use of a *k-d tree* to order the template histograms. This becomes beneficial when many more classes (for example, 1000 instead of 11) have to be compared to classify an individual fish. Another improvement that can be made is that the algorithm automatically generates a new class based on some distance criteria. For example, if the compared distance against all classes is higher than an arbitrary value, then a new class is generated using the instance as one of the template histograms.

Lastly, the ROI selection in the data can also be improved to generate better results. The algorithm implemented for the ROI extraction only takes into consideration 2D rotations of the images. However, due to the nature of the salmon swimming in underwater conditions, they will not always be parallel to the camera. This means that both *affine* and *projective* transformations must be taken into consideration since they will have an impact in the histogram calculation and subsequent comparison with the reference histograms.

## VII. Data and Code availability

The data and code used for the **CNN** and **HOG** approaches is available in the following public GitHub repository, whereas the improvements made using the **UNet** approach are available in this second repository.

## References


[1] N. I. Carrera, "Breve historia de la acuicultura y salmonicultura en el sur de chile (1856-2000)," *Territorios y Regionalismos*, vol. 3, no. 3, pp. 36–49, 2020.

[2] D. Bekkozhayeva, M. Saberioon, and P. Cisar, "Automatic individual non-invasive photo-identification of fish (sumatra barb puntigrus tetrazona) using visible patterns on a body," *Aquaculture International*, vol. 29, no. 4, pp. 1481–1493, 2021.

[3] I. Yusup, M. Iqbal, and I. Jaya, "Real-time reef fishes identification using deep learning," in *IOP Conference Series: Earth and Environmental Science*, vol. 429, no. 1. IOP Publishing, 2020, p. 012046.

[4] W. Li, Z. Ji, L. Wang, C. Sun, and X. Yang, "Automatic individual identification of holstein dairy cows using tailhead images," *Computers and electronics in agriculture*, vol. 142, pp. 622–631, 2017.

[5] N. J. C. Strachan, P. Nesvadba, and A. R. Allen, "Fish species recognition by shape analysis of images," *Pattern Recognition*, vol. 23, no. 5, pp. 539–544, 1990.

[6] S. Villon, D. Mouillot, M. Chaumont, E. S. Darling, G. Subsol, T. Claverie, and S. Villéger, "A deep learning method for accurate and fast identification of coral reef fishes in underwater images," *Ecological informatics*, vol. 48, pp. 238–244, 2018.

[7] U. Freitas, M. Pache, W. Gonçalves, E. Matsubara, J. Sabino, D. Sant'Ana, and H. Pistori, "Analysis of color feature extraction techniques for fish species identification," in *Anais do XVI Workshop de Visão Computacional*. SBC, 2020, pp. 140–145.

[8] S. Zhao, S. Zhang, J. Liu, H. Wang, J. Zhu, D. Li, and R. Zhao, "Application of machine learning in intelligent fish aquaculture: A review," *Aquaculture*, vol. 540, p. 736724, 2021.

[9] J. Navarro, A. Perezgrueso, C. Barría, and M. Coll, "Photo-identification as a tool to study small-spotted catshark scyliorhinus canicula," *Journal of Fish Biology*, vol. 92, no. 5, pp. 1657–1662, 2018.

[10] L. H. Stien, J. Nilsson, S. Bui, J. E. Fosseidengen, T. S. Kristiansen, Ø. Øverli, and O. Folkedal, "Consistent melanophore spot patterns allow long-term individual recognition of atlantic salmon salmo salar," *Journal of Fish Biology*, vol. 91, no. 6, pp. 1699–1712, 2017.

[11] Q. Al-Jubouri, R. Al-Azawi, M. Al-Taee, and I. Young, "Efficient individual identification of zebrafish using hue/saturation/value color model," *The Egyptian Journal of Aquatic Research*, vol. 44, no. 4, pp. 271–277, 2018.

[12] P. Cisar, D. Bekkozhayeva, O. Movchan, M. Saberioon, and R. Schraml, "Computer vision based individual fish identification using skin dot pattern," *Scientific Reports*, vol. 11, no. 1, pp. 1–12, 2021.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.