# Whalification

Luca Distefano[a], Anna Fumagalli[b], Giuliano Giampietro[c] and Vincenzo Maria Vitale[d]

[a]University "La Sapienza", Rome, Italy

ARTICLE INFO

*Keywords*:
Biometric Systems
Identification
Verification
Image Processing
Siamese Network
U-Net

ABSTRACT

This paper presents a system for identifying and verifying individual humpback whales from tail fluke images. The approach combines image preprocessing techniques, U-NET segmentation, and a Siamese convolutional neural network for feature extraction. The system performs both verification and identification tasks, using various classifiers for the latter. Results demonstrate the potential of this vision-based non-invasive method for monitoring individual whales, supporting marine biology research and conservation efforts.

## 1. Introduction

Biometric recognition of animals is essential for reasons ranging from scientific research to species conservation. Individual recognition of animals allows them to be tracked over time, which is crucial for understanding migration patterns, social dynamics, survival strategies, estimating population size and monitoring changes over time. Automated biometric recognition drastically reduces the time required for individual identification compared to traditional methods. Furthermore, it is non-invasive and represents a harmless and stress-free alternative to other methods, such as physical tagging or genetic sampling, which can be stressful or harmful to the animals.

Humpback whales (Megaptera novaeangliae) were the first large whales individually identified by photo-identification methods. That was possible thanks to the characteristic and essentially unchanging pigment pattern on the underside of their fins and the unique shape of the trailing edge of the fins Cheeseman (2022).

Our work builds on these expanding methodologies and follows in the wake of a competition launched in 2019, in which people are challenged to *recognize a whale from its tail*, building an algorithm to identify individual whales (Addison Howard (2018)).

## 2. Database

The database used for this project is the competition database, which contains images collected by research institutes and public collaborators. The training data contains thousands of images of humpback whale flukes. Researchers identified the whales and assigned them an ID. Over five thousand whale individuals have an ID. However, there are no IDs in the test data.

In total, we worked with 33323 files, divided into training and test sets. Since we did not participate in the competition, we selected the training folder, which contained each whale ID, for our project. Indeed, we needed a ground truth to evaluate our model's result, however ensuring a clear separation between the training and test sub-folders. To better understand the training folder's structure, we wrote a short analysis code to determine the number of images for each individual. Specifically, we counted 132 individuals represented with more than fifteen images and 4874 with less than that. In addition, there was a *New Whale* class containing all images of unidentified whales with therefore a substantial number of files, which we decided not to consider in the absence of groundtruth. Indeed, ultimately, we chose only individuals with more than 15 images for identification and verification, resulting in a subset of 132 identifiers. This subset represents the sub-database used for training and testing in the various stages of our project, a process we will delve into in the following sections.

## 3. Thresholding

The first step in our image pre-processing was to isolate the whale's tail from the background. This task, as we discovered, was far from straightforward. The predominant blue hues in the images made it challenging to make accurate distinctions. After careful consideration, we decided to use the masks obtained by thresholding as ground truths for a neural network. This network was trained to generate masks on unseen raw images, a process requiring high technical expertise and precision.

As mentioned, in the first instance, we generated masks using computer vision techniques, which this section focuses on. The images were converted to grayscale to apply thresholding. The threshold applied is the inverse binary thresholding, which sets the destination pixel as zero if the corresponding source pixel is greater than the threshold and to the *maxValue* if the source pixel is less than the threshold. The thresholding method employed is *Otsu Thresholding*, a global threshold algorithm which processes the input image, generates histograms representing the distribution of pixels, computes the threshold and finally, replaces image pixels into white in those regions where saturation is greater than the threshold and into the black in the opposite cases.

The thresholded images are then morphologically processed with opening compound operations, where the image is first subject to erosion and then dilation. Morphological opening is valid for removing tiny objects and thin lines from an image while preserving the shape and size of larger

ORCID(s):

objects in the image, in this case, hopefully, of the caudal fin (MathWorks).

We then found the processed image's contours. By contours, we mean a curve joining all the continuous points along the boundary, having the same color or intensity. We wanted indeed to identify the contours to constraint the minimum size of the objects to be considered. Placing this constraint meant that we could discard, after careful tuning, all small objects that were irrelevant to us.

Finally, we assessed which contours remained: the best contour, i.e. the one with the greatest contrast to the background, was the only one considered. The caudal fin, in the end, is always and only one!

## 4. Improving the masks: U-NET

If the masks were already satisfactory with computer vision techniques, a neural network trained with the latter made identifying masks much more robust.

The network chosen for this purpose is a U-NET. The *U* shaped model comprises convolutional layers and two networks: first, an encoder, then the decoder. Its standard structure - which also corresponds to the one used - in shown in Fig. 1. The encoder network is also a contracting one. This network learns a feature map of the input image. Our encoder network consists of four encoder blocks. Each block contains two convolutional layers with a kernel size of $3 \times 3$, followed by a ReLu activation function. That is fed to a max pooling layer with a kernel size of $2 \times 2$. The max pooling layer halves the spatial dimensions. Between the encoder and decoder network, we have the bottleneck layer. It consists of two convolutional layers followed by a ReLu; its output is a final feature map representation. Instead, the decoder is an expansive network that should sample the feature maps to the size of our input image. This network takes the feature map from the bottleneck layer and generates a segmentation mask. It consists of four decoder blocks, each starting with a transpose convolution with a kernel size of $2 \times 2$. The network utilizes two convolutional layers with a kernel size of $3 \times 3$ followed by a Relu activation function. A $1 \times 1$ convolution follows the last decoder block with sigmoid activation, giving the output of a segmentation mask containing pixel-wise classification.

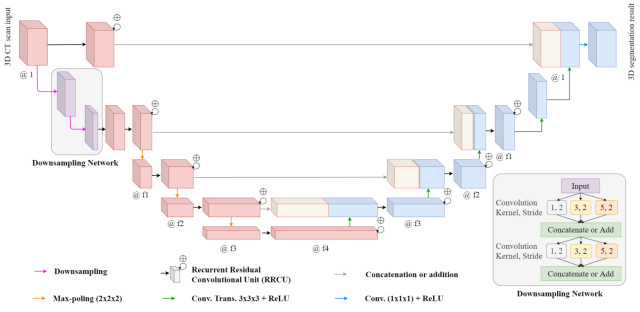



**Figure 1:** U-NET structure

The architecture so structured allows us to capture both global (namely, of the entire image) and local (details) of the images given as input, which can also be of different sizes. Moreover, thanks to the skip connections that link each encoding layer to the relative decoding one, we can keep relevant spatial information, including the ones about borders and contours. These characteristics, combined with its computational efficiency and effectiveness, make this network suitable for the application of image segmentation.

We trained the U-NET on all the images of each of the 132 whales in the subdirectory mentioned in Sec. 2. The ground truths for each image were the masks obtained from the computer vision techniques (Sect. 3). We did not evaluate the U-NET directly with any metric; we did not want it to resemble ground truth as closely as possible but to exceed its accuracy. Indeed, we postponed its performance evaluation until we extracted the feature maps from the masked images with the masks generated with this NN and measured classification performance metrics. Indeed, we noticed a visible improvement compared to not using masks and only those generated with computer vision techniques.

However, we explicitly customised the loss function for image segmentation problems. It combines Binary-Cross-Entropy (BCE) Loss and Dice Loss, weighting them with relative weight coefficients. The BCE Loss measures the differences between model predictions (inputs) and actual values (targets). Indeed, its formula is:

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(p_i) + (1-y_i) log(1-p_i)] \quad (1)$$

where with $y_i$ we refer to the real value (ground-truth), and with $p_i$ we refer to to the prediction, for each $i^{th}$ pixel. That loss measures how well the model can predict each pixel to belong to the target class. On the other hand, the Dice Loss measures the superposition between the prediction of the model and the real values. It is defined as:

$$Dice = 1 - \frac{2 \times \text{intersection} + 1}{\text{predictions sum} + \text{real values sum} + 1}, \quad (2)$$

where the intersection is the number of True Positives. The Dice Loss considers both the False Negative, reaching a good superposition between predictions and real masks.

We wanted to have the best properties of the two combined in one: the BCE is capable of evaluating the differences pixel by pixel between prediction and target but could not capture well the whole structure, while the Dice is sensitive to the overall superposition, but might not penalize sufficiently each false positive and false negative. We report the final loss of the model in the Fig. 2.

```
Final best loss: 0.20278334993906696
```

**Figure 2:** U-NET loss

A comparison between the original image, the computer vision-obtained mask and the U-NET one are shown in the Fig. 3.

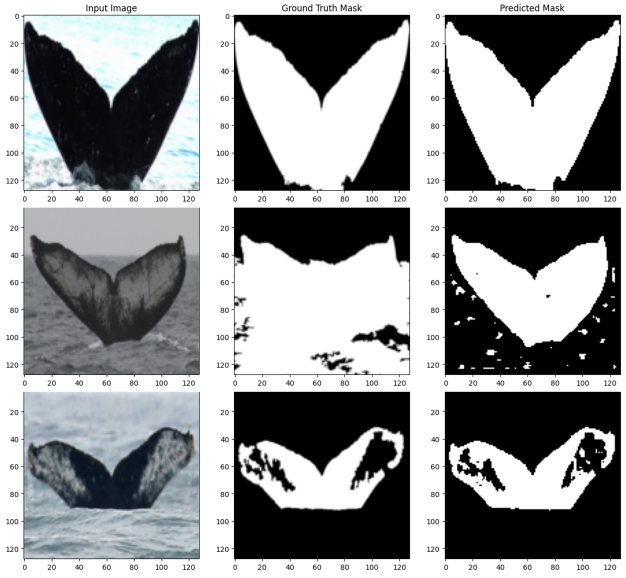



**Figure 3:** Comparison between the original image, the computer vision mask (ground truth for the U-NET, and the U-NET mask.

**Table 1**

| Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|
| 0.79 | 0.80 | 0.79 | 0.79 |

After configuring the U-NET, we generate masks for each image in our subset. These masks are then applied to the original images, effectively isolating the regions of interest. The resulting cropped images serve as input for the Siamese Network in the subsequent stage. We will further elaborate into the classification performance in the following sections.

## 5. Features Extraction

The first attempt was to employ a ResNet-50, a pre-trained Convolutional Neural Network excellent for image classification. We wanted to employ ResNet-50 for image classification, dividing the (sub-)database randomly into training, validation, and testing (with a percentage of 80-10-10). From the layer prior to the output layer, we could take the relevant feature vectors and instead, through classification, assess how much they were relevant and how well they allowed for good image classification, i.e. how well the neural network was able, with those features, to tell from the image which whale it was. To assess its performance, we used standard accuracy, precision and recall metrics, the results of which, normalized to one, are in Table 1. The training trend, on the other hand, is shown in the graphs in the Fig. 4.

The second attempt was to employ EfficientNets, which promise, as is easily understood from the name, to be more efficient than other pre-trained networks in image classification tasks.

EfficientNet has several implementations, specifically 8, called B0 to B7. The EfficientNet from B0 to B7 differs

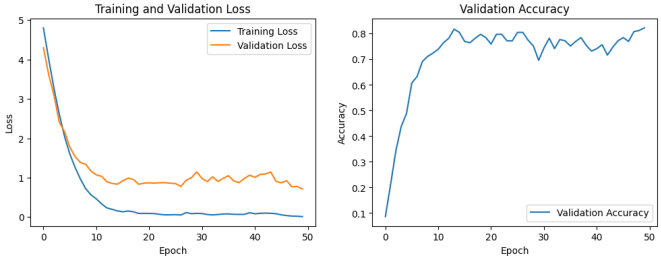



**Figure 4:** Training and Validation Loss, and Validation Accuracy in the ResNet-50

**Table 2**
Quantitative metrics comparison between different model complexity

| | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| **B0** | 0.78 | 0.80 | 0.78 | 0.79 |
| **B1** | 0.78 | 0.79 | 0.78 | 0.78 |

in three main factors: image resolution, network depth, and width. Those factors affect the model's ability to capture complex information and handle higher resolutions. B0 is the base model with lower image resolution, while from B1 to B7, the image resolution, network depth, and width progressively increase. As those parameters grew, the computational time and complexity also grew; we only tried two implementations to fit within reasonable computational time limits: B0 and B1. Their metrics are in Table 2; the training trends instead are in Fig. 5

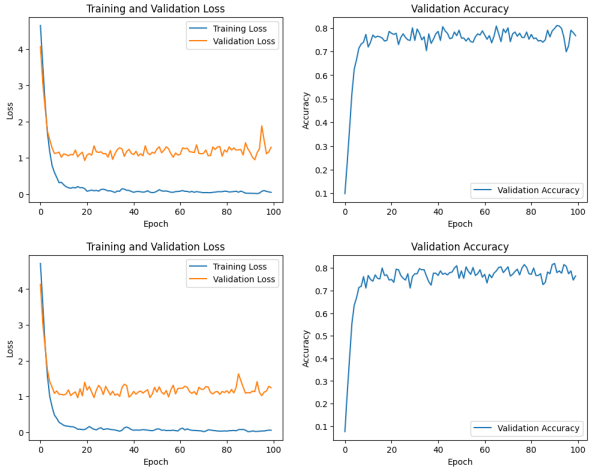



**Figure 5:** Training and Validation Loss and Validation Accuracy for: B0 (images on top), and B1 (images on bottom)

Although both ResNet-50 and EfficientNet had promising classification ML metrics, neither network had positive results when evaluated in biometric contexts with FAR, FRR or ROC metrics, for example. Of the latter metric, we report one of the disappointing plots obtained in Fig. 6.

The networks were very good at classifying the images but did not identify similarities or differences between the images. More technically, they were not good at extracting

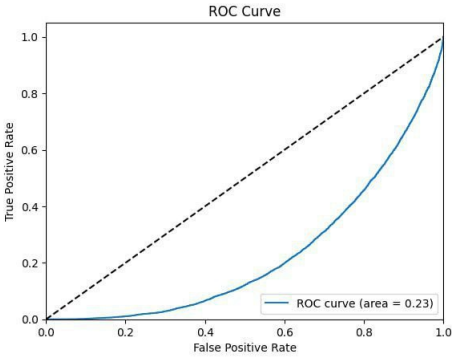



**Figure 6:** Disappointing standard ROC obtained with EfficientNet and ResNet-50

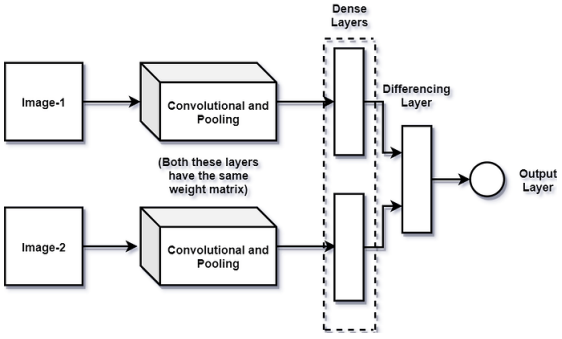



**Figure 7:** Siamese Convolutional Neural Network

features that minimised intra-class distance and maximised inter-class distance. That is the principle behind similarity learning, and just like that, we concluded that it would make sense to employ Siamese Neural Networks employing a contrastive loss.

### 5.1. Siamese Convolutional Neural Networks for feature extraction

Consider having two images we want to compare to see if they are similar or dissimilar pairs (i.e. belonging or not to the same class). The first Convolutional Sub-Net takes the first image (A) as input and returns its vector representation. The second one, which shares the same parameters, architecture, and weights as the first, takes the second image (B) as input and returns its relative features vector. The encodings are then compared to determine whether they are similar and, in turn, the input images. The standard architecture is summarised in Fig. 7. As mentioned, the Siamese network does not classify input images, but is more on differentiating between them. So, the typical loss similarity learning is not generally for classification tasks. The latter is the Contrastive Loss, which has the formula:

$$\mathcal{L}_{cont}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathbb{1}[y_i = y_j]||f_\theta(\mathbf{x}_i - f_\theta(\mathbf{x}_j)||_2^2 + \mathbb{1}[y_i \neq y_j]\max(0, \in -||f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)||_2)^2 \quad (3)$$

and, as can be seen, it depends on the Euclidean Distance of the vectors and maximises the distances between the images belonging to different classes (over a certain threshold $\in$) while minimising the distance of the images of the same class.

We trained the Siamese Network on the dataset consisting of all classes with at least fifteen images per class, which were trimmed to exactly fifteen images per class to have a balanced dataset. We then split the so-structured dataset into training, validation, and test (in percentage 70-20-10).

**Table 3**

| Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|
| 75.76% | 74.29% | 78.79% | 76.47% |

#### *S-CNN Results*

After the parameter tuning, the best Siamese Neural Network obtained has the architecture shown in Fig. 8[1] and batch size 32.

```
# Create the Siamese model
def create_siamese_model(input_shape):
    input = layers.Input(shape=input_shape)
    x = layers.Conv2D(64, (10, 10), activation='relu')(input)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(128, (8, 8), activation='relu')(x)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(128, (4, 4), activation='relu')(x)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(256, (4, 4), activation='relu')(x)
    x = layers.Flatten()(x)
    x = layers.Dense(4096, activation='sigmoid')(x)
    model = models.Model(input, x)
    return model
```

**Figure 8:** Final structure of one of the two identical CNN in the Siamese Network

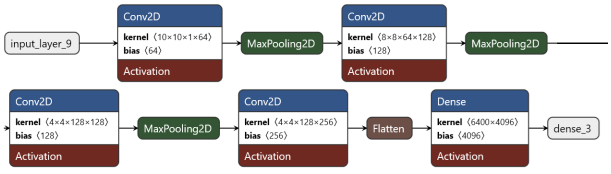



**Figure 9:** Graphical visualizer of the Siamese netowork

The metrics for this model are comparable to the ones obtained with the EfficientNet or the ResNet-50 and are the ones in Tab. 3.

## 6. Evaluation

In this section, we will analyse the results obtained in Verification (Sub-sect. 6.1) and Identification (Sub-sect. 6.2), in terms of those metrics typically applied to biometric systems.

[1] The architecture replicates identically for both the Siamese Convolutional (sub-)Networks.

### 6.1. Verification

Verification in biometric systems is the process of confirming whether a claimed identity matches the true identity of an individual. In our system, verification involves comparing two given whale images and determining whether they belong to the same specimen.

In Figure 10, the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate against the false positive rate at various threshold settings, yielded an area Under the Curve (AUC) of 0.8106. This value denotes that our system performs well in distinguishing between genuine and impostor attempts, demonstrating its effectiveness in whale identification tasks.

For the verification system, we had to choose a similarity threshold above which images were considered to belong to the same whale. We did not just choose it empirically, instead, we used the Equal Error Rate.

In general terms, the EER is the point at which the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) are equal. This equilibrium point is crucial as it measures the tradeoff between safety (FAR minimisation) and usability (FRR minimisation).

As mentioned, we used the EER to find the acceptance threshold. Our system has proven to be extremely robust and secure in discriminating if two images belong to the same id or not, with a threshold of 0.999. At this threshold value, our system demonstrated a balanced False Acceptance Rate (FAR) and False Rejection Rate (FRR) of 0.1332.

In addition to the Euclidean distance used to compare Siamese network output, we also employed cosine similarity -defined as in Eq. (4)- as a metric to evaluate feature similarity. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space, providing a value in range $[-1, +1]$ where 1 indicates perfect similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4}$$

Where $A_i$ and $B_i$ are the $ith$ components of $A$ and $B$ respectively. This metric is particularly useful in high-dimensional spaces as it focuses on the orientation rather than the magnitude of the vectors, making it less sensitive to absolute differences in feature values.

Our system's performance demonstrates its potential for real-world applications in whale research and conservation. Its ability to balance false acceptances and rejections provides an initial foundation for accurate whale identification. As we continue to refine our approach, this system promises to be a valuable tool in expanding our understanding of whale populations and supporting conservation efforts.

### 6.2. Identification

In contrast to verification, identification in biometric systems involves comparing a given sample against a database of known individuals to determine the identity of the unknown sample. In our whale identification system, this means matching a whale image against our database to determine which specific whale it represents. For the identification task, we implemented multiple classifiers that take as input the feature vectors extracted from our Siamese CNN model. We evaluated three main approaches: Support Vector Machines (SVM), Random Forests, and K-Nearest Neighbors (KNN).

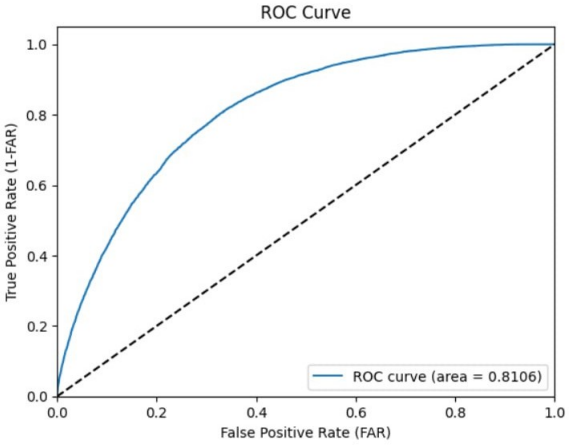


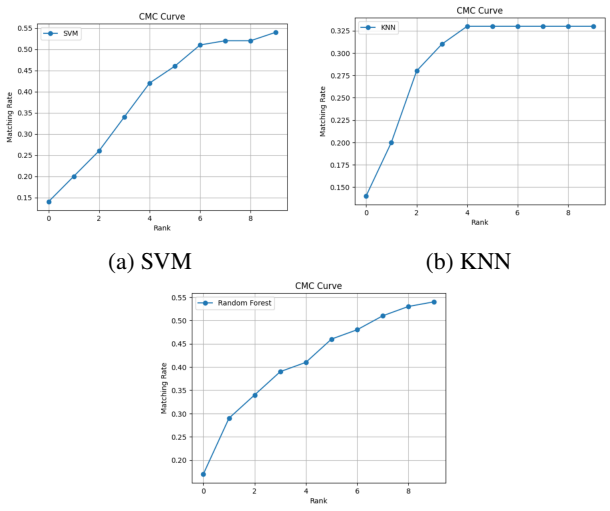

**Figure 10:** Best siamese model ROC curve

The results for top-1 accuracy and top-10 accuracy are shown in Table 4. While in Figure 11 the different Cumulative Matching Characteristics (CMC) curves of those classifiers are shown.

#### Top-$k$ accuracy comparison

Top-$k$ accuracy in identification measures the proportion of times the correct label is among the top $k$ predictions made by the model. It provides an understanding of how often the correct identity is included within the top-$k$ guesses. While the SVM showed the best performance in top-10 accuracy, and KNN provided a balance between top-1 and top-10 accuracies, we chose to prioritize the Random Forest classifier for our final implementation. This decision was based on the significantly higher top-1 accuracy (0.20 compared to 0.05 for SVM and 0.10 for KNN), which is crucial for security and precise identification purposes, rather than using top-10 metrics. The Random Forest classifier's ability to correctly identify the exact match 20% of the time makes it more suitable for applications where precise individual identification is prioritised. The top-10 accuracy of 0.60 for the Random Forest classifier also indicates that the correct whale is often among the top candidates.

It's important to note that our system assumes that the whale trying to being identified exists in the dataset as a labelled whale. This assumption simplifies the identification process but prevents the system's ability to recognize new, previously unencountered individuals, since it would be impossible to recognize.




(a) SVM (b) KNN

(c) Random Forest

**Figure 11:** Comparison of ROC curves between Support Vector Machine (a), K-Nearest Neighbour(b) and Random Forest (c).

**Table 4**

| Model | Top-1 Accuracy | Top-10 Accuracy |
|---|---|---|
| SVM | 5% | 65% |
| KNN | 10% | 35% |
| Random Forest | 20% | 60% |

## 7. Conclusion

Our whale identification and verification system demonstrates promising results in the challenging field of marine vision based biometrics. By employing a Siamese Convolutional Neural Network for feature extraction, followed by specialized classifiers for identification, we've created a quite robust framework for individual whale recognition.
The verification system achieved an AUC of 0.8106 in the ROC curve, indicating good performance in recognizing same-whale images. With balanced FAR and FRR rates of 0.1332 at the EER threshold, our system provides a quite solid foundation for whale verification. In the identification task, our Random Forest classifier achieved a top-1 accuracy of 0.20 and a top-10 accuracy of 0.60. While there is room for improvement, these results are encouraging given the complexity of the task and the variability in whale appearances such as partially submerged fin, from where all the features are being extracted.
The choice of the Random Forest classifier over SVM, despite a slightly lower top-10 accuracy, prioritizes precision in individual identification, which is crucial for security applications.

Our work contributes to the field of marine biology and conservation by providing a tool that can aid in the non-invasive monitoring of whales. The system's ability to verify and identify individual whales can support various research initiatives in which a fast identification/verification of whale specimens is critical.

Future work could focus on improving the feature extraction process, expanding the training dataset, and refining the classification algorithms to enhance both verification and identification accuracies. For example, one could think of using a Neural Network in the classification during the identification phase, instead of employing ML approaches, potentially improving the performance of the system and certainly, reducing its execution time once the training phase is over. Furthermore, an Open-set instead of Closed-set approach could be employed, opening the doors for non-*enrolled* individuals to be identified as new.

Additionally, integrating this system with other data sources, such as geographic information or temporal data, could provide even more comprehensive insights into whale behaviour and population dynamics, to further refine feature extraction.

## *Appendix*: Demo application

This project lead us to create a simple yet effective app to show our system performances. We've created a simple graphical interface to demonstrate our whale identification and verification system. The demo allows users to:

- Choose between verification (comparing two whale images) and identification (matching a single image against the database).
- Drag and drop image files directly into the application.
- Use ready sample images to test the system quickly.

Follow the Readme file for further instructions on using the app and install dependencies to run locally. Next, in Fig. 12, 13, and 14, screenshots of our working app.

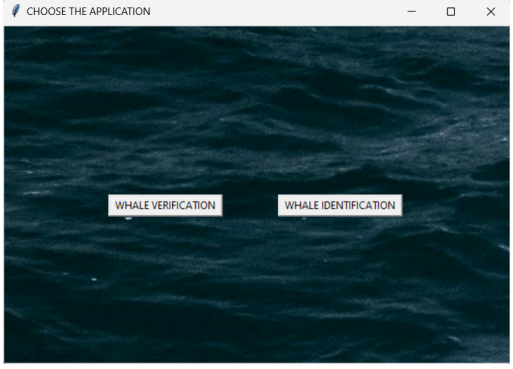



**Figure 12:** Main Interface

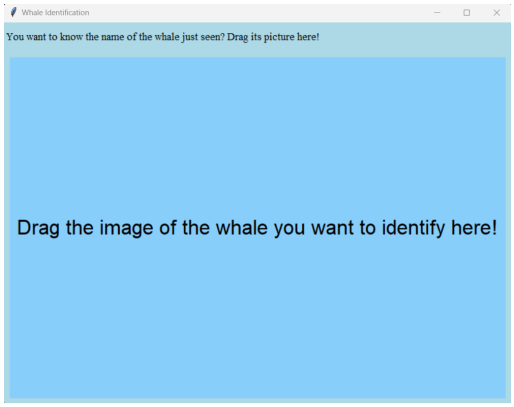



**Figure 13:** Identification Interface

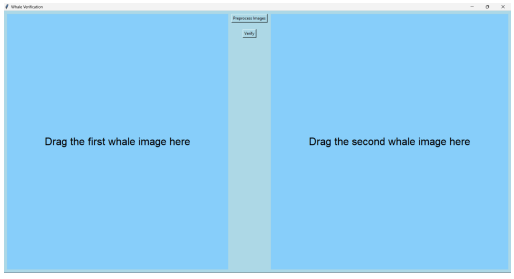



**Figure 14:** Caption

# References

Addison Howard, inversion, K.S.T.C., 2018. Humpback whale identification. URL: https://kaggle.com/competitions/humpback-whale-identification.

Cheeseman, T., S.K.P.J.e.a., 2022. Advanced image recognition: a fully automated, high-accuracy photo-identification matching system for humpback whales. Mamm Biol 102, 915–929. doi:https://doi.org/10.1007/s42991-021-00180-9.

MathWorks, . Types of morphological operations. URL: https://it.mathworks.com/help/images/morphological-dilation-and-erosion.html.