Fish recognition model for fraud prevention using convolutional neural networks

Rhayane S. Monteiro · Morgana C. O. Ribeiro · Calebi A. S. Viana · Mário W. L. Moreira · Glácio S. Araúo · Joel J. P. C. Rodrigues

Received: 14 June 2022 / Revised: 25 October 2022 / Accepted: 10 November 2022 / Published online: 19 December 2022
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2022

Abstract
Fraud, misidentification, and adulteration of food, whether unintentional or purposeful, are a worldwide and growing concern. Aquaculture and fisheries are recognized as one of the sectors most vulnerable to food fraud. Besides, a series of risks related to health and distrust between consumer and popular market makes this sector develop an effective solution for fraud control. Species identification is an essential aspect to expose commercial fraud. Convolutional neural networks (CNNs) are one of the most powerful tools for image recognition and classification tasks. Thus, the objective of this study is to propose a model of recognition of fish species based on CNNs. After the implementation and comparison of the results of the CNNs, it was found that the Xception architecture achieved better performance with 86% accuracy. It was also possible to build a web application mockup. The proposal is easily applied in other aquaculture areas such as the species recognition of lobsters, shrimp, among other seafood.

Keywords Computer vision · Convolutional neural networks · Fish recognition · Food fraud

1 Introduction
According to the International Featured Standards (IFS) guide, the deliberate and intentional substitution, incorrect labeling, adulteration or falsification of food, raw materials, ingredients, or packaging is defined as product fraud. The food fraud, in turn, is a growing global concern (Böhme et al. 2019). With globalization and complex distribution systems, food fraud has become commonplace, with far-reaching impact and adverse consequences on the well-being of the population. According to the Food Integrity Manual, prepared by Food Integrity, adulteration is the most common, considering the types of food fraud. However, food authentication covers many other topics, such as geographic origin, variety identification, production method, as well as the detection of genetically modified organisms. According to Song et al., food fraud is a problem that has a significant impact on individuals, companies, and society and can, therefore, cause a public health risk, business loss, and reduced consumer confidence in brands (Song et al. 2020).

Fish and other seafood are one of the most commercialized products in the world (Böhme et al. 2019). According to Probst, due to its performance on a global scale, the fishing sector is frequently challenged by the confidence lack in various stages of the production process and the supply chain.
(Probst 2020). The Food and Agriculture Organization of the United Nations (FAO) report states that fisheries and aquaculture are recognized as one of the sectors most vulnerable to food fraud3. Adulteration in fish is committed when the fish is purposely placed on the market, for financial gain, to deceive the customer. There are different types of fraud that can occur at various points along the fish supply chain. The most common type involves intentional labeling and species substitution. In contrast, the ability to verify the authenticity and provenance of seafood, from catch to consumption, has become an essential requirement for sustainable, safe, and compatible commercial practices (Vo et al. 2020). Thus, quick and easy tools are needed to assess the authenticity of these products to address economic fraud throughout the supply chain.

Fish recognition and classification is a challenging task and is a promising field for further research (Rachmatullah and Supriana 2018). In addition, the fish identification is presented as a fundamental aspect to expose commercial fraud. Previously, fish species were identified manually by observation and, therefore, it was necessary to remember or study various characteristics of fish to recognize them. Furthermore, the identification of fish species using the only appearance is difficult for humans, because species from the same family have a similar appearance, in addition to the environment and surrounding conditions, such as habitats, can change the fish color (Masuda et al. 2020). In this context, there is a need for techniques capable of performing a correct, automatic, and rapid identification of fish. When analyzing different species recognition systems to avoid counterfeiting, several decision criteria must be considered. These challenges include response time, accuracy, resolution, and features. According to Hridayami et al., advanced technology, combined with artificial intelligence (AI) and deep learning methods, facilitates the task of recognizing fish species (Hridayami et al. 2019).

The Internet of Things (IoT) has presented itself as a novel network paradigm in a widespread and ubiquitous way, offering distributed and transparent services (Cui et al. 2018). By IoT, many smart devices are connected, such as sensors, smartphones, and other smart devices. These devices can communicate with each other and exchange a large amount of information. However, in real-world scenarios, it is difficult to transmit large amounts of heterogeneous data, perceive complex environments from that data, and then make smart decisions in a timely manner (Zhang and Tao 2020). It is expected that AI technologies to be widely applied to IoT devices to collect, share, and analyze more data and valuable information (Ma et al. 2019). AI, especially deep learning techniques, has become a proven success in several areas, including speech recognition, natural language processing, and computer vision. Advances in deep learning are a path to improve the perception capacity of IoT systems from a large number of heterogeneous data.

Deep learning is defined as a method of learning by representation, to preserve the data entry information, however, transforming it in a way that becomes useful for use. Thus, deep learning techniques refine the representation at various levels, using a deep artificial neural network and composed of multiple layers of neurons. Due to the efficient ability to learn resources, many complex problems can be solved quickly and effectively (Zhou et al. 2019). Deep learning has several applications, such as automatic speech recognition, natural language processing, and image recognition (Rachel and Varalakshmi 2019). Convolutional neural networks (CNNs) are one of the most popular methods of deep learning (Hridayami et al. 2019). According to Sanghvi et al., this method remains the most sought after choice by computer scientists for image recognition, processing, and classification (Sanghvi et al. 2020).

CNNs are deep learning algorithms that receive an image as input, assign importance or weight to various aspects of it, and can differentiate one from the other. The name of this network type is due to the convolution operation. Convolution consists of extracting useful features from the input data, in this case, from the images. In addition to convolution, a CNN consists of the input layers, the grouping layer (pooling), the fully connected layer, and the output layer. CNNs are commonly used for image classification and recognition due to their high accuracy (Sanghvi et al. 2020). Besides, this method is flexible and can adapt to new data received as the data set grows. Rachel and Varalakshmi claim that the main advantage of CNNs is that they automatically detect resources without any human intervention (Rachel and Varalakshmi 2019). Regarding fish recognition, it is worth noting that fish images are often not clean and fish only cover small areas in an image. Thus, forcing an image classification model, such as CNN, to pay attention to the correct position of images becomes a challenging problem.

In this context, this research proposes a fish recognition model for fraud prevention based on CNNs. The main reason for choosing CNNs is that, according to the state of the art in the area, this type of artificial neural network presents itself as the best method to deal with tasks associated with the images identification and classification. In addition, as it is a deep learning method, CNN eliminates the manual feature detection step, making it easier to use and making it possible to analyze its performance in different scenarios. Then, the dataset built in this study was submitted to CNN architectures for classification. Subsequently, tests and experiments were carried out to evaluate and validate the model with the best performance. Finally, a prototype of a web application is presented using a CNN model.

The main objectives of this paper are listed as follows:
– Create a data set with images of fish in real fishmonger environments.
– Develop CNN architectures using transfer learning and fine-tuning techniques.
– Compare and discuss the performance of CNNs in different scenarios.
– Propose a prototype web application to be developed on mobile and desktop devices with the CNN model with the best performance.

The rest of the paper is organized as follows. Section 2 addresses the relevant background in the topic and Sect. 3 elaborates on the most relevant related work for the study. The materials and methods are shown in Sect. 4 and results analysis and discussion are considered in Sect. 5. Finally, Sect. 6 concludes the paper and presents suggestions for future works.

2 Background

This section summarizes the necessary concepts and background in the topic and purposes of this study.

2.1 Convolutional neural networks

CNNs are a class of deep learning algorithms used to analyze data and one of the best learning algorithms for understanding image content. In addition, they showed exemplary performance in tasks related to image segmentation, classification, detection, and retrieval (Khan et al. 2020). Some of the areas of application for CNNs include object detection, video processing, image classification, and segmentation. Deep CNN’s powerful learning ability is mainly due to the use of various stages of resource extraction that can automatically learn representations of the data. In addition, CNNs need less pre-processing time compared to other classification algorithms since it can learn the characteristics automatically, whereas, in traditional methods, the process is done manually.

CNN consists of neurons that composed of weights, bias, and activation functions. Similar to other neural networks, CNN comprises input, hidden, and output layers, which perform operations that alter data to investigate specific resources. A basic CNN architecture has the input, convolution, and pooling layers, which alternate, fully connected, and output layers, as shown in Fig. 1. CNN receives an image as input. The convolution layer acts as a resource extractor for the input images, whose dimensionality is then reduced by the grouping layers. The convolutional layers encode a number of lower-level features for more discriminating features, which transform one input image into another, highlighting specific patterns. The fully connected layer acts as a classifier, which exploits the high-level resources learned by CNN to classify input images into predefined classes. In short, this layer receives a vector as an input and produces another vector as an output.

The availability of a large amount of data and the improvement of hardware technology has accelerated research on CNNs. Consequently, several new architectures have been proposed. The listed models of CNNs used in this study are covered in the next section. The criteria for choosing these models are based on the most recent state-of-the-art algorithms. Besides, these architectures are from public sources.

2.1.1 VGG16

Simonyan and Zisserman proposed the VGG architecture, which presented a simple and effective design principle for CNN architectures (Simonyan and Zisserman 2014). The VGG16 has 23 layers to simulate the depth relationship with the representational capacity of the network. This CNN showed good results for both image classification and location problems, however, it is commonly recognized due to its simplicity, homogeneous topology and greater depth (Khan et al. 2020).

2.1.2 InceptionV3

The primary idea of InceptionV3 was to reduce the computational cost of deep networks without affecting the generalization. To that end, Szegedy et al. replaced large size filters with small, asymmetric filters and used the $1 \times 1$ convolution as a bottleneck before the large filters (Szegedy et al. 2016). Such characteristics have become the main contributions of this architecture. This CNN is 159 layers deep.

2.1.3 DenseNet121

Elaborated by Huang et al., the DenseNet was proposed to solve the disappearance gradient problem (Huang et al. 2017). For this, this CNN used cross-layer connectivity, which connected each previous layer to the next arrival layer in a feed-forward way. Thus, feature maps for all previous layers were used as inputs for all subsequent layers. As DenseNet concatenates the characteristics of the previous layer instead of adding them, the network can gain the ability to explicitly differentiate between the information that is added to the network and that which is preserved (Khan et al. 2020). This architecture presents 121 layers of depth.

2.1.4 MobileNet

Elaborated by Howard et al., the MobileNet was proposed for use on mobile devices. For this, this CNN uses in-depth separable convolutions (Howard et al. 2017). This fact significantly reduces the number of parameters when compared to networks with
regular convolutions of the same depth. This results in light deep neural networks. By defining the network in such simple terms, it is possible to easily explore its structure and outline a good network (Khasoggi et al. 2019).

### 2.1.5 Xception

Xception was proposed by Chollet and it can be considered an architecture of extreme beginning, which explores the idea of separable convolution in-depth (Chollet 2017). This architecture modified the original initial block making it wider and replacing the different spatial dimensions with a single dimension, followed by a $1 \times 1$ convolution to regulate computational complexity. Xception is 126 layers deep. Although the transformation strategy adopted by this CNN does not reduce the number of parameters, it makes learning more efficient and results in better performance (Khan et al. 2020).

### 3 Related works

Several approaches using CNNs have been proposed for the recognition of fish species, achieving promising results. In this section, some studies related to the theme of this research are presented. The papers are shown in Table 1 and will be explained below.

Rossi et al. presented the FishAPP, a cloud-based infrastructure for recognizing fish species (Rossi et al. 2016). The approach is through a mobile application developed for Android and iOS, allowing the user to take photos of an entire fish and send them for remote analysis. A cloud system performs image processing and is then subjected to a CNN capable of analyzing the images obtained and performing classification on fish species. The preliminary results obtained from the available data set provided encouraging results. The study imposes, on the user, a series of guidelines when taking the photo of the fish, such as the photo must include the complete fish, it must be photographed from the side and the tail fin must be placed anatomically and relaxed. Such factors restrict the relevance of the application, as it does not suit different and possible real-life scenarios.

The research of Rauf et al. showed a deep learning structure based on the CNN method for identifying fish species (Rauf et al. 2019). The proposed CNN architecture contains 32 deep layers to extract valuable and discriminating features from the image. To test the model performance, the authors created a data set called Fish-Pak, containing 915 images. To validate the proposed CNN, the authors compared their results with other structures of deep learning, involving VGG16, LeNet5, AlexNet, GoogleNet, and ResNet50. Empirical analyses revealed that the proposed method achieves satisfactory performance and exceeds existing methods. In the dataset of this research, six species of fish were considered, segmenting the images in specific regions that characterize each species studied. However, the dataset is also unbalanced. This characteristic can result in an inflexible algorithm, making it ineffective for certain species of fish.

Santos and Gonçalves presented a CNN-based method for the recognition of fish species in the Brazilian Pantanal (Santos and Gonçalves 2019). A CNN, composed of three branches that classify fish by family and order, is proposed to improve the recognition of species with similar characteristics. The results in the data set showed that the proposed method obtained an accuracy of 0.873 and 0.864 using a traditional CNN. The authors recognized 68 species of fish. Despite presenting a variability of images in the dataset, encouraging CNN to adapt to different data inputs, the focus on extracting images in real environments has been shifted. Thus, the model can obtain inferior performance when receiving low-quality images, in environments different from the images presented, among other situations.

Masuda et al. proposed an advanced method of identifying fish species using images of fish and their meristic characteristics. The authors developed a model of multiple entries that presents images and, also, data of characteristics, aiming to become an advanced system of identification of fish species. To show the accuracy of the proposed model, the study compared with models that used only images or just
meristic characters. As a result, it was clarified that the use of two or more types of meristic characters provided greater precision than the use of images only (Masuda et al. 2020). The authors worked with the identification of nine species of fish. However, as exposed in the study, the number of images by species is unbalanced. Then, some species have about double or triple the images of other species, causing overfitting. Therefore, the algorithm erroneously classifies species of fish for presenting a few training data.

The proposal by Majumber et al. focused on recognizing freshwater fish species commonly found in Bangladesh. The authors analyzed seven fish species to build the dataset from images collected at various fish markets in the country. This article used six CNN-based architectures with transfer learning, namely DenseNet201, InceptionResnetV2, InceptionV3, ResNet50, ResNet152V2 and Xception. In the end, the InceptionResnetV2 and Xception approaches achieved the highest accuracy with 98.81% (Majumder et al. 2021). The main limitation of the work is that it does not deal with fish species from the same family, which represents a greater challenge for the model. In addition, the dataset images shown in the work present photos of fish in the same orientation (horizontal) and similar poses, which also facilitates the performance of the model.

From the studies presented, there are still challenges for the task of recognizing fish. They are i) dataset with limited number of images; ii) imbalance in the quantity of images by species, and iii) restriction of images in real scenarios and
under restricted conditions. Therefore, the objective of this study is to propose a model to combat fish fraud using CNNs, thus contributing to overcoming the challenges listed.

4 Materials and methods

In this section, the materials and methods necessary for the implementation of this research proposal are presented.

Figure 2 shows the steps taken to achieve the objective of this research. These phases comprise five main activities, namely, elaboration of the dataset (1), implementation of the CNN architectures (2), training and testing of the architectures (3), evaluation of the results (4), and the web application mockup (5). These steps are explained in detail below.

Step 1 comprises the elaboration of the dataset. For this, a data set with 338 images was built, divided into 8 species of marine fish, commonly found on the East Coast of Ceará, Brazil. The images were captured between August 2019 and August 2020 using the Canon camera, Smartphones iPhone 7 Plus, and Xiaomi RedMi Note 8 devices. Besides, the photos were collected at fish sales establishments in the city of Aracati, CE, Brazil. Figure 3 shows some examples of images in the data set. These images generally include elements other than fish, such as objects, the human body, among others. These variations encourage the model to select identification characteristics that do not depend on the orientation or focus of the image being classified. All images photographed were used with the intention of building a robust dataset, with variations in poses and fish orientation. After collecting the images, they were divided into folders by fish species. This division was performed manually. Then, the dataset divided into folders was loaded into the development environment. The next step was to perform a pre-processing of the data, to transform each image into a matrix of pixels. As the original images consist of RGB coefficients in the range of 0 to 255, and these values would be too high for the CNN models to process, a resizing of the matrix values of each image was performed, to keep the values in a range of 0–1, scaling with factor 1/255.

After the elaboration of the dataset, the CNN architectures were implemented, which is step 2 of the proposal. For the coding of the models, the TensorFlow4, defined as an open-source platform for machine learning, was used. Also, the Keras5 was also used, which is a high-level API of the TensorFlow, used to create and train deep learning models. Concerning the chosen CNNs, this study opted for the following architectures, namely, VGG16, InceptionV3, DenseNet121, MobileNet, and Xception. These CNNs were chosen because they are the most recent state-of-the-art networks in the area. Besides, these algorithms are available at the API Keras. The architectures were implemented using two techniques, namely, transfer learning and fine-tuning. Transfer learning comprises the use of a pre-trained model applied to a new problem. In summary, a pre-trained neural network is used for a new application. The fine-tuning, on the other hand, reuse the weights of the original network in the new model and, using a new data set, a new training step is carried out to adapt the weights to the new dataset. In this research, the initial layers of pre-trained CNNs served as a feature extractor. Then, the top of the network was modified to suit the classes of the model and new training was carried out.

Step 3 relates to the task of training and testing the models. Figure 4 shows the distribution of images after the elaboration of the dataset. It can be seen that this is partially balanced, presenting an average of 42 images per class. The dataset was divided using the ratio of 70% for training and 30% for testing to train the architectures, respectively. Table 2 shows the division of fish species by family and scientific name. In summary, this study analyzed 4 types of families distributed in 8 types of fish. It is important to highlight that species from the same family have very similar characteristics and the model has the challenge of finding the attributes that differentiate each species.

With the trained and tested CNN models, step 4 was reached, the analysis of results. Further details of the results are shown in Sect. 5. To test the performance of CNNs in each scenario, the confusion matrix was used, which is a table that shows the classification frequencies for each class of the model. As metrics, precision, recall, F score and accuracy were used. These measures were chosen because they are metrics commonly used to analyze the performance of classification models. In addition, it is important to analyze several metrics, as the dataset is partially balanced. Before understanding the mathematical formula for each metric, it is important to understand the following terms:

---

4 Available at https://www.tensorflow.org.
5 Available at https://keras.io/.
Table 2  Species distribution by family

| Family       | Specie (Popular Name) | Scientific Name       |
|--------------|-----------------------|-----------------------|
| Carangidae   | Yellow jack           | Carangoides bartholomaei |
| Lutjanidae   | Lane snapper          | Lutjanus synagris     |
|              | Mutton snapper        | Lutjanus analis       |
|              | Dog snapper           | Lutjanus jocu         |
|              | Yellowtail snapper    | Lutjanus chrysurus    |
| Sciaenidae   | Smooth weakfish       | Cynoscion leariah     |
| Scombridae   | Kingfish              | Scomberomorus cavalla |
|              | Serra Spanish mackerel| Scomberomorus brasiliensis |

- True positive (TP): observations whose actual value is positive and the predicted value is positive, that is, the model was correct.
- False positive (FP): cases where the correct result is negative; however, the result obtained is positive, that is, the model was wrong.
- True negative (TN): observations whose actual value is negative and the predicted value is negative, that is, the model was right.
- False negative (FN): cases in which the correct result is positive; however, the result obtained is negative, that is, the model was wrong.

The precision formula is shown in Equation 1. This measure is used to indicate the relationship between correctly performed positive predictions and all positive predictions (including false ones). Thus, its formula is calculated as the division between the rate of true positives and the sum of true positives and false positives.

\[
Precision = \frac{TP}{TP + FP}. \tag{1}
\]

The recall metric is displayed in Equation 2 and represents the proportion of positive cases that were correctly identified. The result is calculated as the division between the rate of true positives and the sum of true positives and false negatives.

\[
Recall = \frac{TP}{TP + FN}. \tag{2}
\]

F score represents the harmonic mean between precision and recall. The formula for this metric is in Equation 3.

\[
F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{3}
\]

Accuracy is the proportion of hits in the model and its formula is represented in Equation 4. In summary, it represents the sum of the total number of observations that the model got right (true positives and true negatives) and the total number of observations that the model predicted (true positives, false positives, true negatives, and false negatives).

\[
Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4}
\]

Finally, in step 5, the web application mockup was prepared from the software Adobe XD for building prototypes. The mockup presents the layout of a software graphical interface. It is useful to demonstrate the use and the final result of an application. In this study, the web application prototype was built, aiming at future implementation based on the CNN model with the best performance obtained in this research.

5 Discussion and result analysis

Next, the results obtained from the CNN test stage are shown. Two scenarios were elaborated to obtain the results shown in this section. The first, in which the test dataset was submitted during the training stages at the top of the network; and the second, in which the test data set was submitted only in the test stage of the model. The results achieved for each scenario are presented below.

Considering scenario 1, Fig. 5 shows the confusion matrices of the CNNs. The confusion matrix is a table that allows you to view the frequency of errors and hits in the model and is useful for analyzing its performance. Considering Fig. 5a, which shows the matrix of the VGG16 network, it is observed that there was a good performance for the species Lane snapper (13), Yellowtail snapper (13), Dog snapper (14), and Yellow jack (17). For the species Mutton snapper (4), Kingfish (5), and Serra Spanish mackerel (5) there was a low prediction. Figure 5b shows the result for InceptionV3, with the species Serra Spanish mackerel (11), Yellow jack (12), Lane snapper (13), Dog snapper (13), and Yellowtail snapper (13) the greatest predictions. Kingfish (7) and Smooth weakfish (7), in turn, had the lowest predictions. The DenseNet121 matrix is shown in Fig. 5c. In this Figure, Lane snapper (13), Yellow jack (14), Dog snapper (16), and Yellowtail snapper (16) were the most correctly predicted species, while...
Kingfish (7) and Mutton snapper (7) again underperformed. Figure 5d shows the result for MobileNet. The species Yellow jack (16), Serra Spanish mackerel (17), and Yellowtail snapper (21) reached a higher frequency of correctly predicted values. In this CNN, in particular, in addition to the species Mutton snapper (6) and Kingfish (7) showing inferior performance, Dog snapper (9) also obtained the same behavior. Finally, Fig. 5e shows the results for Xception. In this matrix, Yellow jack (15), Yellowtail snapper (18), and Dog snapper (20) obtained the highest frequencies of correctly predicted values. In contrast, Mutton snapper (5) and Kingfish (7) had low correct predictions. Briefly, CNNs VGG16, InceptionV3, and DenseNet121 showed better results considering the confusion matrices. Concomitantly, the species with the largest number of images in the dataset performed better, namely, Lane snapper, Dog snapper, Yellowtail snapper, and Yellow jack. On the other hand, the species Kingfish and Mutton snapper showed inferior performance, a fact that may be related to the quality of the images.

For scenario 1, Table 3 shows the metrics precision, recall, and F score. Table 3a presents the results for the precision metric. Accuracy indicates the relationship between positive predictions correctly and all positive predictions (including false ones). In general, the species that obtained the best performance independently of CNN were Dog snapper, Yellowtail snapper, Yellow jack, and Smooth weakfish. Regarding the architectures, the Xception obtained better precision with an average of 0.89. Table 3b shows performance for the recall metric. This parameter lists the positive forecasts correctly made and all forecasts, i.e., true positives and false negatives. In this table, rates fluctuated for all classes and architectures. The CNN with the best recall was MobileNet, with an average of 0.85. The F score metric is shown in Table 3c. This standard, in turn, is the harmonic mean between precision and recall, which indicates the overall quality of the model. Again, rates varied for classes and CNNs, with the Xception architecture performing better, averaging 0.85. In light of the above, the importance of the number of images per class is perceived to assist in the training of the model and, consequently, improve its performance when subjected to tests. Another result obtained was the accuracy of the CNN architectures. Accuracy represents how close the prediction is to the correct result. Thus, the greater the accuracy, the closer to the real value is the result found. Table 3d shows that Xception presented better accuracy, with a rate of 86%, as well as better results for the metrics precision and F score.

The results for scenario 2 will now be presented. The first of these is the confusion matrices, shown in Fig. 6. When considering the VGG16 architecture, the species Lane snapper (13), Serra Spanish mackerel (14), Dog snapper (16), and Yellow jack (18) have the highest frequencies of correct predictions, as shown in Fig. 6a. Kingfish (4) and Mutton snapper (6), however, obtained the lowest frequencies of correct values. Figure 6b shows the confusion matrix of InceptionV3. In this figure, Serra Spanish mackerel (10), Lane snapper (11), Dog snapper (11), Yellowtail snapper (12), and Yellow jack (13) had better results, while Mutton snapper (6) again presented an irregular behavior. The confusion matrix of DenseNet121 is shown in Fig. 5c. For this CNN, Dog snapper (15), Yellowtail snapper (17), and Yellow jack (23) obtained more correct predictions. However, Lane snapper (6), Kingfish (6), and Mutton snapper (6) underperformed. Figure 6d shows the results for the MobileNet architecture. In this result, the classes Lane snapper (13), Yellow jack (14), Yellowtail snapper (15) and Dog snapper (17) have more positive predictions, while Kingfish (8), Mutton snapper (8), and Smooth weakfish (8) resulted in fewer values correct. The matrix for CNN Xception is shown in Fig. 5e, which shows that Lane snapper (12), Dog snapper (13), Yellowtail snapper (13), Yellow jack (16), and Serra Spanish mackerel (16) obtained positive predictions more frequently. However, Mutton snapper (5) and Kingfish (6) showed low performance. Given the above, it appears that the Xception and MobileNet CNNs showed better performances in terms of the confusion matrix. Once more, it is important to note that species with a greater number of images tend to have better results.

Table 4 presents the metrics precision, recall, and F score for scenario 2. Considering the precision metric, the Dog snapper species and Yellowtail snapper obtained higher rates, according to Table 4a. The CNN DenseNet121 performed better among classes, averaging 0.87. When observing the recall metric in Table 4b, it appears that all species had varying rates for all CNNs. Once more, DenseNet121 achieved better efficiency with an average rate of 0.84. The metric F score, shown in Table 4c, shows that the classes Yellow jack and Smooth weakfish obtained the best results. DenseNet121 also had the best result, averaging 0.83. In short, the DenseNet121 architecture performed better for the metrics in Table 4. The final accuracy of the CNN models is shown in Table 4d. It is observed that for scenario 2, the DenseNet121 architecture presented better accuracy, with a rate of 85%, also demonstrating its effectiveness for the metrics previously analyzed.

In summary, it can be seen that, in scenario 1, the CNN Xception performed better with 86% accuracy, and in scenario 2, the CNN DenseNet121 achieved 85% performance in terms of accuracy. The difference, in terms of accuracy for both scenarios, is minimal, although for each scenario there was a different CNN.

After obtaining the CNN model with the best result from the tests performed, the mockup for the web application was developed, which is detailed as follows. On the home screen, shown in Fig. 7, the user is introduced to the Fish Classifier system. It was decided to establish a pattern of screens,
Fig. 5 Confusion matrices for scenario 1

(a) VGG16

(b) InceptionV3

(c) DenseNet121

(d) MobileNet

(e) Xception
### Table 3: Metrics for scenario 1

|                | Lane snapper | Kingfish | Mutton snapper | Dog snapper | Yellowtail snapper | Yellow jack | Smooth weakfish | Serra Spanish mackerel |
|----------------|--------------|----------|----------------|-------------|--------------------|-------------|-----------------|------------------------|
| (a) Precision Metric. |              |          |                |             |                    |             |                 |                        |
| VGG16          | 0.52         | 0.50     | 0.80           | 0.93        | 0.81               | 0.81        | 1.00            | 0.56                   |
| InceptionV3    | 0.81         | 1.00     | 0.91           | 0.93        | 0.54               | 1.00        | 1.00            | 0.61                   |
| DenseNet121    | 0.52         | 1.00     | 0.70           | 0.94        | 1.00               | 1.00        | 1.00            | 0.57                   |
| MobileNet      | 0.92         | 1.00     | 0.75           | 0.75        | 0.84               | 0.94        | 0.71            | 0.85                   |
| Xception       | **1.00**     | **0.88** | **1.00**       | **0.91**    | **0.72**           | **0.83**    | **1.00**        | **0.85**               |
| (b) Recall Metric.  |              |          |                |             |                    |             |                 |                        |
| VGG16          | 1.00         | 0.83     | 0.33           | 0.64        | 0.68               | 0.77        | 1.00            | 0.71                   |
| InceptionV3    | 0.81         | 0.64     | 0.77           | 0.87        | 0.93               | 0.60        | 0.78            | 1.00                   |
| DenseNet121    | 1.00         | 0.64     | 0.64           | 1.00        | 0.70               | 0.82        | 0.60            | 1.00                   |
| MobileNet      | **0.86**     | **0.70** | **0.67**       | **0.90**    | **0.84**           | **0.84**    | **1.00**        | **1.00**               |
| Xception       | 0.73         | 0.64     | 0.62           | 0.91        | 0.95               | 0.94        | 1.00            | 0.92                   |
| (c) F-Score Metric. |              |          |                |             |                    |             |                 |                        |
| VGG16          | 0.68         | 0.62     | 0.47           | 0.76        | 0.74               | 0.79        | 1.00            | 0.63                   |
| InceptionV3    | 0.81         | 0.78     | 0.83           | 0.90        | 0.68               | 0.75        | 0.88            | 0.76                   |
| DenseNet121    | 0.68         | 0.78     | 0.67           | 0.97        | 0.82               | 0.90        | 0.75            | 0.73                   |
| MobileNet      | 0.89         | 0.82     | 0.71           | 0.82        | 0.84               | 0.89        | 0.83            | 0.92                   |
| Xception       | **0.84**     | **0.74** | **0.77**       | **0.91**    | **0.82**           | **0.88**    | **1.00**        | **0.88**               |
| (d) Accuracy Metric.  |              |          |                |             |                    |             |                 |                        |
| VGG16          | 0.72         |          |                |             |                    |             |                 |                        |
| InceptionV3    | 0.78         |          |                |             |                    |             |                 |                        |
| DenseNet121    | 0.79         |          |                |             |                    |             |                 |                        |
| MobileNet      | 0.85         |          |                |             |                    |             |                 |                        |
| Xception       | **0.86**     |          |                |             |                    |             |                 |                        |
Fig. 6 Confusion matrices for scenario 2

(a) VGG16

(b) InceptionV3

(c) DenseNet121

(d) MobileNet

(e) Xception
### Table 4  Metrics for scenario 2

|                  | Lane snapper | Kingfish | Mutton snapper | Dog snapper | Yellowtail snapper | Yellow jack | Smooth weakfish | Serra Spanish mackerel |
|------------------|--------------|----------|----------------|-------------|--------------------|-------------|-----------------|------------------------|
| (a) Precision Metric |              |          |                |             |                    |             |                 |                        |
| VGG16            | 0.81         | 1.00     | 0.60           | 0.89        | 0.50               | 1.00        | 0.78            | 0.88                   |
| InceptionV3      | 0.73         | 0.50     | 0.46           | 0.92        | 0.80               | 0.93        | 0.90            | 0.71                   |
| **DenseNet121**  | **1.00**     | **0.67** | **0.86**       | **0.83**    | **0.71**           | **0.92**    | **1.00**        | **1.00**               |
| MobileNet        | 0.81         | 0.62     | 0.73           | 1.00        | 0.79               | 0.82        | 1.00            | 0.88                   |
| Xception         | 1.00         | 1.00     | 0.33           | 1.00        | 0.76               | 1.00        | 1.00            | 0.70                   |
| (b) Recall Metric |              |          |                |             |                    |             |                 |                        |
| VGG16            | 0.76         | 0.57     | 0.75           | 0.70        | 0.82               | 0.90        | 1.00            | 0.88                   |
| InceptionV3      | 0.85         | 0.80     | 0.86           | 0.65        | 0.67               | 0.76        | 0.82            | 0.62                   |
| **DenseNet121**  | **0.55**     | **0.86** | **0.75**       | **0.88**    | **0.94**           | **0.88**    | **1.00**        | **0.86**               |
| MobileNet        | 0.93         | 1.00     | 0.80           | 0.81        | 0.94               | 0.82        | 1.00            | 0.47                   |
| Xception         | 0.80         | 0.46     | 1.00           | 0.72        | 0.68               | 1.00        | 1.00            | 1.00                   |
| (c) F score Metric |              |          |                |             |                    |             |                 |                        |
| VGG16            | 0.79         | 0.73     | 0.67           | 0.78        | 0.62               | 0.95        | 0.88            | 0.88                   |
| InceptionV3      | 0.79         | 0.62     | 0.60           | 0.76        | 0.73               | 0.84        | 0.86            | 0.67                   |
| **DenseNet121**  | **0.71**     | **0.75** | **0.80**       | **0.86**    | **0.81**           | **0.90**    | **1.00**        | **0.92**               |
| MobileNet        | 0.87         | 0.76     | 0.76           | 0.89        | 0.86               | 0.82        | 1.00            | 0.61                   |
| Xception         | 0.89         | 0.63     | 0.50           | 0.84        | 0.72               | 1.00        | 1.00            | 0.82                   |
| (d) Accuracy Metric |              |          |                |             |                    |             |                 |                        |
| VGG16            | 0.79         |          |                |             |                    |             |                 |                        |
| InceptionV3      | 0.73         |          |                |             |                    |             |                 |                        |
| **DenseNet121**  | **0.85**     |          |                |             |                    |             |                 |                        |
| MobileNet        | 0.82         |          |                |             |                    |             |                 |                        |
| Xception         | 0.80         |          |                |             |                    |             |                 |                        |
Fig. 7  Home screen of the web application

Fig. 8  Classification screen of the web application

Fig. 9  Result screen of the web application
starting from the initial screen. Relevant detail is that the system logo takes you to the home page, anywhere in the application. The “About” button, in turn, redirects you to the screen with information from the Fish Classifier. Also on the home screen, there is a brief text about the application and the species available for classification. The user can perform the classification in the button “Start Here!”. The next screen is the classification screen, shown in Fig. 8. On this screen, the user can upload the fish image that he wants to classify. Then, the user must click on the “Determine the Class!” button. Then, the system redirects to the result page, shown in Fig. 9. The user obtains the result of the classifier, indicating the species to which the image belongs. If he wants to know more about the species, just click on the option “Learn more about this species”. Finally, on the About screen, shown in Fig. 10, a text about the application is presented with general information, such as description, objective, target audience, species available for classification, among others. The user can return to the system home screen by clicking on the Fish Classifier logo. When building the mockup, the objective was to show a prototype of the system to be implemented with the model that presented the best performance, in this case, Xception with an accuracy of 86%. When developing the web application, it is expected that it has a simple and, at the same time, intuitive interface for the user, according to the prototype shown in this section.

6 Conclusion and future works

This paper presented a fish recognition model for preventing fish fraud. From the above, it was possible to build a dataset with images in real environments. Furthermore, after the implementation and comparison of the results of the CNNs, it was found that the Xception architecture achieved better performance with 86% accuracy. It was also possible to build a web application mockup. The proposal is easily applied in other aquaculture areas such as the species recognition of lobsters, shrimp, among other seafood. This can be done by collecting images of the species to be analyzed and then submitting them to CNN so that it can be trained and the data learned by the model. It is important that a part of the data is destined for testing the model to assess its classification power. Given the above, it is concluded that the model can be applied to prevent fraud in the fish supply chain.

The study limitations are related to the construction of the data set because collecting images of certain species is very time-consuming. Another challenge was to understand the performance of each architecture and its implementation, from the choice stage to the analysis of the results.

Suggestions for further works are based on increasing the accuracy rate of the model and on the coding of the web application from the mockup of the screens. Thus, it is hoped that the model can contribute to the prevention of fish fraud, supporting society, industry, and government.

Acknowledgements This work is partially supported by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020; and by Brazilian National Council for Scientific and Technological Development (CNPq) via Grant No. 313036/2020-9.

Funding This research received no external funding.

Data availability The authors used own data and own coding.

Declarations

Conflict of interest The authors declare no conflict of interest.
References

Böhme K, Calo-Mata P, Barros-Velázquez J, Ortea I (2019) Recent applications of omics-based technologies to main topics in food authentication. TrAC, Trends Anal Chem 110:221–232. https://doi.org/10.1016/j.trac.2018.11.005

Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Honolulu, HI, USA, pp 1251–1258. https://doi.org/10.1109/CVPR.2017.195

Cui L, Yang S, Chen F et al (2018) A survey on application of machine learning for internet of things. Int J Mach Learn Cybern 9(8):1399–1417. https://doi.org/10.1007/s13042-018-0834-5

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861

Hridayami P, Putra IK, Wibawa KS (2019) Fish species recognition using VGG16 deep convolutional neural network. J Comput Sci Eng 13(3):124–130. https://doi.org/10.5626/JCSE.2019.13.3.124

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Honolulu, HI, USA, pp 4700–4708. https://doi.org/10.1109/CVPR.2017.243

Khan A, Sohail A, Zahoor A, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review pp 1–62, https://doi.org/10.1007/s10462-020-09825-6

Khasoggi B, Ermatita E, Sahmin S (2019) Efficient mobilenet architecture as image recognition on mobile and embedded devices. Indones J Electr Eng Comput Sci 16:389, https://doi.org/10.11591/ijeecs.v16.i1.pp389-394

Ma Y, Ping K, Wu C et al (2019) Artificial intelligence powered internet of things and smart public service. Libr Hi Tech 38:165–179. https://doi.org/10.1108/LHT-12-2017-0274

Majumder A, Rajibongshi A, Rahman MM, Biswas A (2021) Local freshwater fish recognition using different cnn architectures with transfer learning. Int J Adv Sci Eng Inf Technol 11(3):1078–1083

Masuda H, Jukei T, Hasegawa T (2020) Fish species identification using a CNN-based multimodal learning method. In: 2nd International Conference on Image, Video and Signal Processing, ACM, Jakarta, Indonesia, pp 15–19. https://doi.org/10.1145/3388188.3389164

Probst WN (2020) How emerging data technologies can increase trust and transparency in fisheries. ICES J Mar Sci 77(4):1286–1294. https://doi.org/10.1093/icesjms/fsz036

Rachel JJ, Varalakshmi P (2019) Classification of breeding fish using deep learning from the captured video. In: 11th International Conference on Advanced Computing, IEEE, Chennai, India, pp 48–55, https://doi.org/10.1109/ICoAC48765.2019.246815

Rachmatullah MN, Supriana I (2018) Low resolution image fish classification using convolutional neural network. In: 2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA), IEEE, pp 78–83

Rauf HT, Lali MI, Zahoor S et al (2019) Visual features based automated identification of fish species using deep convolutional neural networks. Comput Electron Agric 167:105075. https://doi.org/10.1016/j.compag.2019.105075

Rossi F, Benso A, Di Carlo S, et al. (2016) FishAPP: A mobile App to detect fish falsification through image processing and machine learning techniques. In: IEEE International Conference on Automation, Quality and Testing, Robotics, IEEE, Cluj-Napoca, Romania, pp 1–6, https://doi.org/10.1109/AQTR.2016.7501348

Singhvi K, Aralkar A, Sanghvi S, Saha I (2020) Fauna image classification using convolutional neural network. Int J Fut Gen Commun Netw 13(1):08–16

Santos AA, Gonçalves WN (2019) Improving pantanal fish species recognition through taxonomic ranks in convolutional neural networks. Eco Inform 53:100977. https://doi.org/10.1016/j.ecoinf.2019.100977

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Song W, Jiang N, Wang H, Vincent J (2020) Use of smartphone videos and pattern recognition for food authentication. Sens Actuators B Chem 304:127247. https://doi.org/10.1016/j.snb.2019.127247

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Las Vegas, NV, USA, pp 2818–2826, https://doi.org/10.1109/CVPR.2016.308

Vo SA, Scanlan J, Turner P, Ollington R (2020) Convolutional neural networks for individual identification in the southern rock lobster supply chain. Food Control 118:107419. https://doi.org/10.1016/j.foodcont.2020.107419

Zhang J, Tao D (2020) Empowering things with intelligence: a survey of the progress, challenges, and opportunities in artificial intelligence of things. IEEE Internet Things J (in press). https://doi.org/10.1109/JIOT.2020.3039359

Zhou L, Zhang C, Liu F, Qiu Z, He Y (2019) Application of deep learning in food: a review. Compr Rev Food Sci Food Saf 18(6):1793–1811. https://doi.org/10.1111/1541-4337.12492

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.