electronics

MDPI

Brief Report

Image Classification with Convolutional Neural Networks Using Gulf of Maine Humpback Whale Catalog

Nuria Gómez Blas [1,†,‡], Luis Fernando de Mingo López [1,*,†,‡], Alberto Arteta Albert [2,†] and Javier Martínez Llamas [1,†,‡]

[1] Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, 28031 Madrid, Spain; nuria.gomez.blas@upm.es (N.G.B.); javier.martinez.llamas@alumnos.upm.es (J.M.L.);
[2] Department of Computer Science, College of Arts and Sciences, Troy University, Troy, AL 36082, USA; aarteta@troy.edu
* Correspondence: fernando.demingo@upm.es; Tel.: +34-91-067-3566
† These authors contributed equally to this work.
‡ Current address: Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Calle Alan Turing s/n, 28031 Madrid, Spain.

Received: 19 March 2020; Accepted: 27 April 2020; Published: 29 April 2020

Abstract: While whale cataloging provides the opportunity to demonstrate the potential of bio preservation as sustainable development, it is essential to have automatic identification models. This paper presents a study and implementation of a convolutional neural network to identify and recognize humpback whale specimens by processing their tails patterns. This work collects datasets of composed images of whale tails, then trains a neural network by analyzing and pre-processing images with TensorFlow and Keras frameworks. This paper focuses on an identification problem, that is, since it is an identification challenge, each whale is a separate class and whales were photographed multiple times and one attempts to identify a whale class in the testing set. Other possible alternatives with lower cost are also introduced and are the subject of discussion in this paper. This paper reports about a network that is not necessarily the best one in terms of accuracy, but this work tries to minimize resources using an image downsampling and a small architecture, interesting for embedded system.

Keywords: convolutional neural networks; pattern recognition; machine learning

1. Introduction

Humpback whales have patterns of black and white pigmentation and scars on the underside of their tails that are unique to each whale, just as fingerprints are to humans. Researchers document the marks or flukes on the right and left lobes of the tail and rate the percentage of dark vs. light skin pigmentation in a range (0–100, 0–100) (100 percent white to 100 percent black).

While whale cataloging provides the opportunity to demonstrate the potential of bio preservation as sustainable development, and at the same time honoring the principles of conservation, it is essential to have automatic identification models.

For scientific purposes, each humpback whale sighted in the North Atlantic is assigned to a catalog number. The unique scarring and shading patterns also provide the inspiration for common names. For Gulf of Maine humpbacks, researchers and naturalists work together each year to name new adult whales and young animals sighted in a second year. New calves are not named because their coloring and scarring often dramatically change during that first year.

Information collected for humpbacks in the sanctuary constitutes the longest and most detailed data set for baleen whales in the world. Photographs in the Gulf of Maine Humpback Whale Catalog,

maintained by the Provincetown Center for Coastal Studies, and the North Atlantic Humpback Whale Catalog, maintained by the College of the Atlantic in Maine, allow scientists and naturalists to identify and monitor individual animals and gather valuable information about population sizes, migration, health, sexual maturity and behavior patterns. Photographing individual whales and their calves each year helps to identify family relationships. Four generations of humpback whales have been documented in certain maternal lines or matrilines.

The computing performance of Artificial Intelligence has increased remarkably in recent years. While it has been available to more and more people at the same time, its technological and social impact will grow exponentially. This fact has included Artificial Intelligence in almost any field of Information Technology, with massive companies such as Google, Facebook, Amazon or Microsoft that offer services, solutions and tools based on Artificial Intelligence such as: Video Games, Virtual Assistants and Financial Services.

One of the main areas that has the most development is the field of image/pattern recognition. This is mainly due to the utility and social precision (compared to the human eye) offered by this field. This type of technology is used in a wide range of systems, from surveillance tools and facial recognition to medical applications such as the early identification of tumors.

Their potential and social involvement is so high that their development must be deeply linked to ethical responsibility. It is notable that Artificial Intelligence and field recognition are not subject to controversy, as long as they receive criticism due to abusive practices or lack of ethics/privacy.

From the first years of Information Technology, the possibility that the machines were able to think has been really attractive and has reached the minds of several writers and artists along with history. They imagined androids that were absolutely indistinguishable from humans and artificial intelligence with capabilities that the human mind is incapable of understanding, among many others. However, to understand the feasibility of these examples, it is necessary to understand what Artificial Intelligence is and how it works.

The term Artificial Intelligence has been raised historically from different points of view: the ability to think or the ability to intelligently act [1]. On the one hand, the first focuses on the approach of a human idea of intelligence, in which machines think and are rational. On the other hand, the second approach is based not so much on the process as on the result, considering the Artificial Intelligence to the ability to act and emulate what would be the result of a strictly rational action. A fundamental part of intelligence lies in learning, a process through which, through information, study and experience, a certain amount of training is achieved. That is why the need arises within the framework of Artificial Intelligence to adequately equip the knowledge systems.

With this objective, automatic learning or automatic learning was born, which, thanks to data processing, seeks to identify common patterns that allow the elaboration of increasingly precise and improved predictions. However, these algorithms have historically required complex statistical knowledge. Following the evolution of machine learning, recent years have seen the birth of a new concept, known as deep learning. Unlike machine learning, deep learning understands the world as a hierarchy of concepts [2], diluting the information in different layers through the use of modules, which transform their representation into a higher and more abstract level. This allows the amplification of the relevant information and eliminates the superfluous one [3].

2. Convolutional Neural Networks

Artificial neural networks or neural networks are mathematical models that try to emulate the natural behavior of biological neural networks. In these models, a network of logical units or neurons interconnected with each other is established. With this connection, they can process the received information and issue a result to the next layer determined by an activation function that takes into account the weight of each input, see Figure 1b. This behavior adds more importance to specific incoming connections.

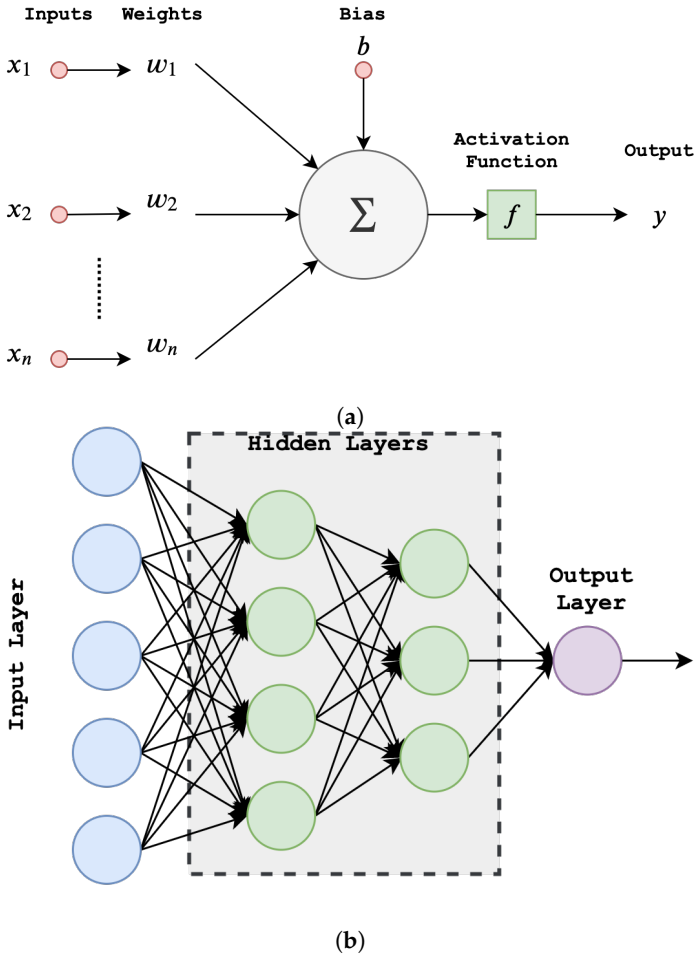


Figure 1. Illustration of an artificial neuron and a simple neural network with 2 hidden layers. In practice, there could be many layers and many neurons per layer. Note that the output of a neuron depends on a non-linear combination of the inputs, provided $f(x)$ is a non-linear function. (**a**) Artificial neuron. (**b**) Multilayer perceptron.

In this model, output obtained in neuron y (Figure 1a) is given by Equation (1), where $\bar{x} = \{x_1, \cdots, x_n\}$ represents the input data, $\bar{w} = \{w_1, \cdots, w_n\}$ is the weight matrix and b is the so-called the bias term.

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{1}$$

During the training phase of a neural network the weight and bias parameters are readjusted in order to adapt the model to a specific task and improve the predictions. The activation function f will be selected according to the problem to solve (a sigmoid or hyperbolic function).

Multilayer neural networks organize and group artificial neurons into levels or layers. These have an input layer and an output layer and might have a variable number of hidden layers between them.

The input layer is formed by neurons that introduce the information into the network, but they do not produce processing, so they only act as a receiver and propagator. The hidden layers are formed by those neurons where both the entrance and the exit connect with other layers of neurons. The output layer is the last level of the network and produces a set of results out of it. The connections of the multilayer neural networks usually move forward, connecting the neurons with their next layer. They are called feed forward networks.

Convolutional neural networks (ConvNet or CNNs) [4,5] are a class of multilayer feed forward neural networks specially designed for the recognition and classification of images. Classification is simply a more general term than pattern recognition. In both cases, you have a set of classes K and a

collection of observations, where each observation is represented by a set of features. The problem is to find a mapping of features to a member of K such that you minimize some measure/estimate of out-of-sample classification error. In pattern recognition, you are simply using a very complex/large feature space. For example, facial recognition will have an input space equal to the number of pixels in the image (this is no different than classifying a loan application as high or low risk based on several measures of creditworthiness).

Computers perceive images in different ways to humans, as long as for these an image consists of a two-dimensional vector with the values relative to the pixels (Figure 2). They have got a channel for greyscale images or three of them for color (RGB).

182	179	79	···	102	101	104
202	202	195	···	95	99	102
172	175	179	···	114	118	122
			···			
64	64	179	···	95	100	126
64	64	64	···	30	81	114
68	72	68	···	135	106	151

Figure 2. Whale tail image vector. Left image is the picture taken and right one is the coded image using a matrix of real values $\in [0, 255]$, where 0 means a white pixel and 255 a black one.

Convolutional networks follow a certain structure, with three main types of layers: Convolutional layer, pooling layer and fully-connected layer, see Figure 3. A series of alternate conversions and subsamples or reductions are made, until finally through a series of completely connected layers (multilayer perception) the desired output is obtained, equivalent to the number of classes.

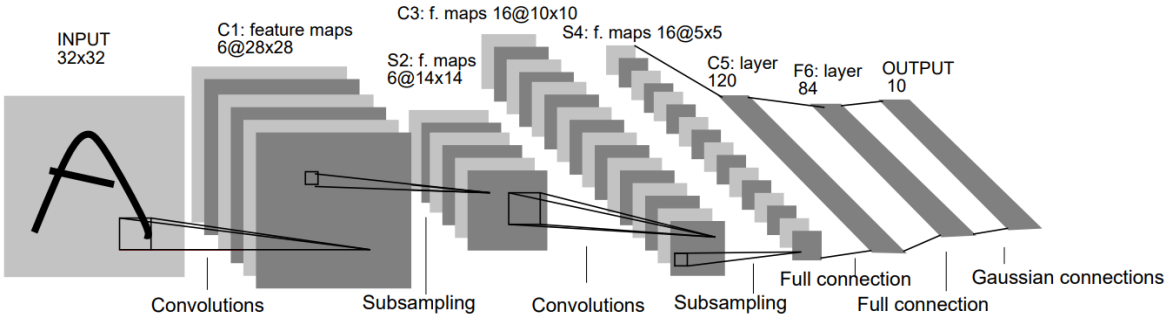
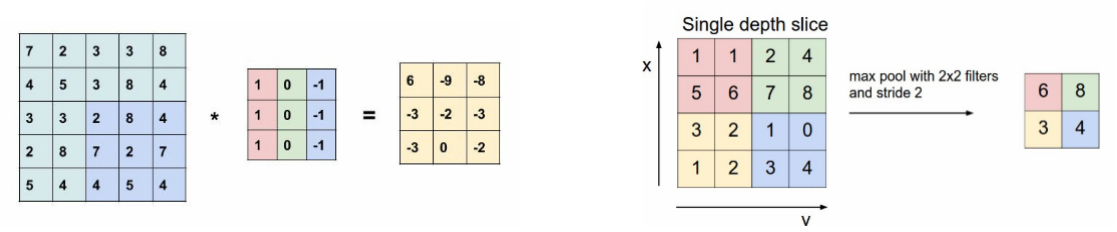


Figure 3. A classic convolutional neural network (CNN) model: LeNet-5 architecture (original image published by LeCun Y. et al. [6]). It consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.

The convolution makes a series of products and sums between the starting matrix and a kernel matrix or filter of size n. On the other hand, the sub-sampling reduces the dimension of the input matrix by dividing it into sub-regions and allowing the generalization of the characteristics (Figure 4).

There are different architectures that are currently considered as the state of the art, such as AlexNet, Inception or VGGNet, highlighting residual neural networks or ResNet [7] among them.



(**a**) (**b**)

Figure 4. Matrix operators in convolutional neural networks. Convolution involves a sum of element by element multiplication, which in turn is the same as a dot product on multidimensional matrices which machine learning researchers call tensors. (**a**) Convolution. (**b**) Max Pooling.

2.1. Evolution of Convolutional Networks: ResNet, AlexNet, VGGNet, Inception

The task of training the whole network from the scratch can be carried out using a large dataset like ImageNet using convolutional neural networks (CNN). The reason behind this is, sharing of parameters between the neurons and sparse connections in convolutional layers. It can be seen in Figure 3. In the convolution operation, the neurons in one layer are only locally connected to the input neurons and the set of parameters are shared across the 2-D feature map.

Most CNNs have huge memory and computation requirements, especially while training. Hence, this becomes an important concern. Similarly, the size of the final trained model becomes important to consider if you are looking to deploy a model to run locally on mobile. As you can guess, it takes a more computationally intensive network to produce more accuracy. Therefore, there is always a trade-off between accuracy and computation.

Apart from these, there are many other factors like ease of training, the ability of a network to generalize well, etc. There are other architectures that are the most popular ones and are presented in the order that they were published and they also had increasingly better accuracy from the earlier ones, see Table 1.

- AlexNet [8]: This architecture was one of the first deep networks to push ImageNet Classification accuracy by a significant stride in comparison to traditional methodologies. It is composed of 5 convolutional layers followed by 3 fully connected layers, as depicted in Figure 1. AlexNet, proposed by Alex Krizhevsky, uses Rectified Linear Unit (ReLu) for the non-linear part, instead of a Tanh or Sigmoid function which was the earlier standard for traditional neural networks. The advantage of the ReLu over sigmoid is that it trains much faster than the latter because the derivative of sigmoid becomes very small in the saturating region and therefore the updates to the weights almost vanish. This is called vanishing gradient problem. In the network, ReLu layer is put after each and every convolutional and fully-connected layer (FC). Another problem that this architecture solved was reducing the over-fitting by using a Dropout layer after every FC layer. Dropout layer has a probability associated with it and is applied at every neuron of the response map separately. It randomly switches off the activation with the probability. The idea behind the dropout is similar to the model ensembles. Due to the dropout layer, different sets of neurons which are switched off, represent a different architecture and all these different architectures are trained in parallel with weight given to each subset and the summation of weights being one. For n neurons attached to DropOut, the number of subset architectures formed is 2^n. It amounts to prediction being averaged over these ensembles of models. This provides a structured model regularization which helps in avoiding the over-fitting. Another view of DropOut being helpful is that since neurons are randomly chosen, they tend to avoid developing co-adaptations among themselves thereby enabling them to develop meaningful features, independent of others.
- VGG16 [9]: This architecture is from VGG group, Oxford. It was an improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. With a given receptive field

(the effective area size of input image on which output depends), multiple stacked smaller size kernel is better than the one with a larger size kernel because multiple non-linear layers increases the depth of the network which enables it to learn more complex features, and that too at a lower cost. For example, three 3×3 filters on top of each other with stride 1 ha, a receptive size of 7, but the number of parameters involved is $3(9C^2)$ in comparison to $49C^2$ parameters of kernels with a size of 7. Here, it is assumed that the number of input and output channel of layers is C. In addition, 3×3 kernels help in retaining finer level properties of the image.

- GoogLeNet/Inception [9]: While VGG achieves a phenomenal accuracy on ImageNet dataset, its deployment on even the most modest sized GPUs is a problem because of huge computational requirements, both in terms of memory and time. It becomes inefficient due to large width of convolutional layers. In a convolutional operation at one location, every output channel is connected to every input channel, and so we call it a dense connection architecture. The GoogLeNet builds on the idea that most of the activations in a deep network are either unnecessary (value of zero) or redundant because of correlations between them. Therefore the most efficient architecture of a deep network will have a sparse connection between the activations. There are techniques to prune out such connections which would result in a sparse weight/connection. Kernels for sparse matrix multiplication are not optimized in BLAS or CuBlas (CUDA for GPU) packages which render them to be even slower than their dense counterparts. Therefore, GoogLeNet devised a module called inception module that approximates a sparse CNN with a normal dense construction. Since only a small number of neurons are effective, the width/number of the convolutional filters of a particular kernel size is kept small. In addition, it uses convolutions of different sizes to capture details at varied scales. Another salient point about the module is that it has a so-called bottleneck layer. It helps in the massive reduction of the computation requirement.
- ResNet [9]: According to the evolution of CNN, increasing the depth should increase the accuracy of the network, as long as over-fitting is taken care of. The problem with increased depth is the signal required to change the weights, which arises from the end of the network by comparing ground-truth and prediction becomes very small at the earlier layers, because of increased depth. It essentially means that earlier layers are almost negligibly learned. This is called vanishing gradient. The second problem with training the deeper networks is performing the optimization on huge parameter space and therefore naively adding the layers leading to higher training error. Residual networks allow training of such deep networks by constructing the network through modules called residual models as shown in the figure. This is called degradation problem. The architecture is similar to the VGGNet consisting mostly of 3×3 filters. From the VGGNet, a shortcut connection as described above is inserted to form a residual network. This can be seen in the figure which shows a small snippet of earlier layer synthesis from VGG-19.

Table 1. Convolutional neural networks architectures evolution (The top-1 accuracy rate is the ratio of images whose ground truth category is exactly the prediction category with maximum probability, while the top-5 accuracy rate indicates the ratio of images whose ground-truth category is within the top-5 prediction categories sorted by the probabilities, accuracy is obtained using ImageNet dataset).

Architecture	Top-1 Accuracy	Top-5 Accuracy	Year
Alexnet	57.1	80.2	2012
Inception-V1	69.8	89.3	2013
VGG	70.5	91.2	2013
Resnet-50	75.2	93	2015
Inception-V3	78.8	94.4	2016

2.2. Programming Language

Among the most popular programming languages in the field of machine learning, two of them stand out: Python and R, being the latter largely oriented towards statistical analysis. In terms of deep learning, the clear dominator is Python, thanks in part to the large number of libraries and frameworks developed for this language, such as PyTorch, Caffe, Theano, TensorFlow or Keras. That is why Python has been chosen in its version 3.6. In addition, different libraries implemented for this language will be used to facilitate the development process. Some of the most relevant packages are the following:

- NumPy, focused on the scientific computation, provides vectors or arrays as well as powerful mathematical tools on them. Pandas for the analysis of data, mainly the reading process of the different CSV files needed. Pandas allows quick access to the data, as well as a powerful treatment of it.
- Scikit-learn, a machine learning library, with powerful statistical tools that will be used mainly during the pre-processing of the data, prior to the implementation of the neural network.
- Python Imaging Library (PIL) used for reading and converting images. In the development phase, although eliminated in the final version, the Matplotlib library has been used in order to create the needed graphs. More specifically, it has been used only for visualization and to obtain the images, both original and processed, during the elaboration of this document. Therefore, it lacks relevance and usefulness in the final versions.
- Developed by Google in order to meet their needs along the machine learning environment, TensorFlow is a library for numerical calculations that mainly uses data flow diagrams. Published under a license of open code in 2015, it has since become one of the most popular benchmarks in the development of deep learning systems and neural networks.
- Born with flexibility and usability in mind, Keras is a library for high level neural networks that was developed for Python. One of the biggest advantages when it comes to Keras usage is that it seeks to greatly simplify the development-related tasks. It is possible to run it with libraries such as TensorFlow, Theano or CNTK as a backend, understanding each other as an interface rather than as a framework. In 2017, Keras was integrated into the source code of TensorFlow allowing its development with a higher level of abstraction. Deep learning and its development have been greatly facilitated by the use of Graphics Processing Unit (GPUs). The high number of calculations that are carried out and their high complexity, especially during the training of a neural network, make high-performance hardware an actual need. GPUs come into play in this current scenario, as long as its usage in the training of neural networks allows to reduce considerably the time taken, compared to the exclusive use of CPUs. This is due to its high number of cores that allow parallel processing.
- çDIA parallel calculus architecture, next to cuDNN, its library for deep neural networks, allows the use of GPUs in TensorFlow.

The performance of the application can vary considerably depending on the available specifications, being critical in the final results and, especially, in the total execution time.

The system used has an Intel Core i7-8700 6-core processor with a base frequency of 3.2 GHz up to 4.6 GHz and 16 GB of DDR4 RAM. It also has a NVIDIA GeForce GTX 1060 graphics card with 3 GB of VDR DDR5 memory with a total of 1152 CUDA cores. This hardware is able to perform the training of the neural network and the processing of images in a comfortable way, however different approaches to the problem would require more memory allocated in the GPU. Please note that all code, datasets and samples are available at https://github.com/javiermzll/CCN-Whale-Recognition.

3. State-of-the-Art in Kaggle Competition: Humpback Whale Identification Methods

After centuries of intense whaling, recovering whale populations still have a hard time adapting to warming oceans and struggle to compete every day with the industrial fishing industry for food.

To aid whale conservation efforts, scientists use photo surveillance systems to monitor ocean activity. They use the shape of whales' tails and unique markings found in footage to identify what species of whale they are analyzing and meticulously log whale pod dynamics and movements. For the past 40 years, most of this work has been done manually by individual scientists, leaving a huge trove of data untapped and underutilized.

> "In this competition, you're challenged to build an algorithm to identify individual whales in images. You will analyze Happywhale's database and Gulf of Maine Humpback Whale Catalog of over 25,000 images, gathered from research institutions and public contributors. By contributing, you will help to open rich fields of understanding for marine mammal population dynamics around the globe. Note, this competition is similar in nature to this competition with an expanded and updated dataset. We'd like to thank Happywhale and Gulf of Maine Humpback Whale Catalog for providing this data and problem. Happywhale is a platform that uses image process algorithms to let anyone to submit their whale photo and have it automatically identified."

Next, find a brief state-of-the-art of more interesting and catchy methods used by other researchers in terms of accuracy, see Table 2 for a quick overview.

- SIFT-Based [10]. This is one of the most beautiful and, at the same time, unusual. David, now a Kaggle Grandmaster (Rank 12), was 4th on the Private LeaderBoard and shared his solution as a post on Kaggle Discussions forum. He worked with full-resolution images and used traditional key point matching techniques, utilizing SIFT and ROOTSIFT. In order to deal with false positives, David trained a U-Net to segment the whale from the background. Interestingly, he used smart post-processing to give classes with only one training example more chance to be in the TOP-1 prediction. The takeaway is that we should never be blinded by the power of deep learning and underestimate the abilities of traditional methods.
- Pure Classification [11,12]. The team Pure Magic thanks radek (7th place), consisting of Dmytro Mishkin, Anastasiia Mishchuk and Igor Krashenyi, pursued approach that was a combination of metric learning (triplet loss) and classification, as Dmytro described in his post. They tried Center Loss to reduce overfitting when training classification models for a long time, along with temperature scaling before applying softmax. Among the numerous backbone architectures that were used, the best one was SE-ResNeXt-50, which was able to reach 0.955 LeaderBoard. Their solution is way more diverse than that, and I highly suggest you to refer to the original post.
- CosFace, ArcFace [13]. As it was mentioned in the post by Ivan Sosin (his team BratanNet was 9th in this competition), they used CosFace and ArcFace approaches. From the original post: Among others Cosface and Arcface stand out as newly discovered state-of-the-art (SOTA) for face recognition task. The main idea is to bring examples of the same class close to each other in cosine similarity space and to pull apart distinct classes. Training with cosface or arcface generally is classification, so the final loss was CrossEntropy. When using larger backbones like InceptionV3 or SE-ResNeXt-50, they noticed overfitting, so they switched to lighter networks like ResNet-34, BN-Inception and DenseNet-121.The team also used carefully selected augmentations and numerous network modification techniques like CoordConv and GapNet. What was particularly interesting in their approach is the way they dealt with new whales. From the original post: Starting from the beginning we realized that it is essential to do something with new whales in order to incorporate them into the training process. Simple solution was to assign each new whale a probability of each class equal to 1/5004. With the help of weighted sampling technique it gave us some boost. Then we realized that we could use softmax predictions for new whales derived from the trained ensemble. Therefore, we came up with distillation. We choose distillation instead of pseudo labels, because new whale is considered to have different labels from the train labels. Though it might not really be true. To further boost the model capability we added

test images with pseudo labels into the train dataset. Eventually, our single model could hit 0.958 with snapshot ensembling. Unfortunately, ensembling trained this way did not give any score improvement.

- Siamese Networks [14]. One of the first architecture was a siamese network with numerous branch architectures and custom loss, which consisted of a number of convolutional and dense layers. The branch architectures that were used included ResNet-18, ResNet-34, Resnet-50 and SE-ResNeXt-50. A progressive learning was used, with the resolution strategy $229 \times 229 \rightarrow 384 \times 384 \rightarrow 512 \times 512$. That is, first the network is trained on 229×229 images with little regularization and larger learning rate. After convergence, the net resets the learning rate and increased regularization, consequently training the network again on images of higher resolution. The models were optimized using Adam optimizer with an initial learning rate of 1^{-4}, reducing 5 times on plateau. The best-performing single model with ResNet-50 scored 0.929.
- Metric Learning [15]. Another approach that was used was metric learning with Margin Loss. Numerous ImageNet-pretrained backbone architectures were used, which included: ResNet-50, ResNet-101, ResNet-152, DenseNet-121 and DenseNet-169. The networks were trained progressively mostly using $448 \times 448 \rightarrow 672 \times 672$ strategy. Adam optimizer is used, decreasing the learning rate 10 times after 100 epochs. We also used a batch size of 96 for the whole training. The most interesting part is a 2% boost right away. It is a metric learning method that was developed by Sanakoyeu, Tschernezki, et al. [16]. What it does is every n epochs it splits the training data as well as the embedding layer into k clusters. After setting up the bijection between the training chunks and the learners, the model trains them separately while accumulating the gradients for the branch network. As a result of the huge class imbalance, heavy augmentations were used, which included random flips, rotate, zoom, blur, lighting, contrast and saturation change. During inference, dot products between the query feature vector and the train gallery feature vectors were calculated and a class with the highest dot product value was selected as the TOP-1 prediction. It is noteworthy to mention that the best-performing single model with a DenseNet-169 backbone scored 0.931.
- Classification on Features [17]. It trains the classification model using the features extracted from all previous models and concatenated together (after applying PCA). The head for the classification consisted of two dense layers with dropout in between. The model trained very quickly because precomputed features were used. This approach allowed to get a 0.924 score and brought even more diversity in the overall ensemble.
- New Whale Classification. One of the most complicated parts of this competition was to correctly classify the new whales (as about 30% of all images belonged to the new_whale class). The popular strategy to deal with this was to use a simple threshold. That is, if the maximum probability that the given image X belongs to some known whale class is smaller than the threshold, it was classified as the new_whale. For each best-performing model and ensemble, its TOP-4 predictions are taken, sorted in descending order. Then, for every other model, probabilities for the selected 4 classes are used. The goal was to predict whether the whale is new or not based on these features. A combination of LogRegression, Support Vector Machines (SVM), several k-NN models and LightGBM is used. The combination of all scores is 0.9655.

This paper reports about a network that is not necessarily the best competitor in Kaggle challenge, but this work tries to minimize resources using an image downsampling and a small architecture, interesting for embedded systems, see accuracy results in Table 2.

Table 2. State-of-the-art of Kaggle competition models vs. proposed model. Note that Kaggle submissions are evaluated according to the Mean Average Precision @ 5 (MAP@5). Proposed method uses a straight forward convolutional neural network, main advantages are the low resources needed and fast training process to be embedded in a small computing device. The method is not the best one but it has a good accuracy.

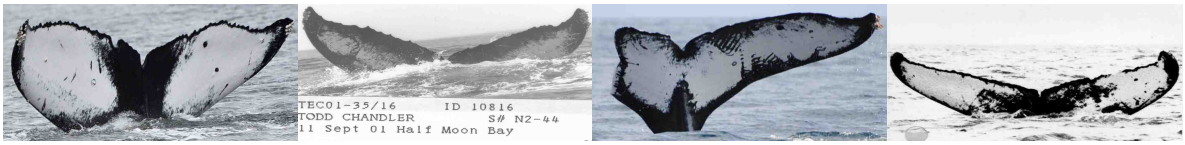
Method	Score/Accuracy
SIFT-Based	0.967
New Whale Classification	0.965
Pure Classification	0.955
CosFace, ArcFace	0.958
Metric Learning	0.931
Siamese Networks	0.929
Classification on Features	0.924
Proposed method	0.785

4. Design

Over the last five years, convolutional neural nets have offered more accurate solutions to many problems in computer vision, and these solutions have surpassed a threshold of acceptability for many applications. Convolutional neural networks have supplanted other approaches to solving problems in these areas, and enabled many new applications. While the design of convolutional neural nets is still something of an art form, in our work we have try to minimize the storage amount for CNN model parameters and processor load using an image downsampling and a small architecture, interesting for embedded systems, such as: cameras mounted on the ship, single board computers, etc. in order to classify images online.

4.1. Dataset

The key aspect to face the construction of a neural network is the study and analysis of the dataset on which it is intended to work. The dataset is constituted by a set of images of humpback whales, more specifically their tails, see Figure 5.



(**a**) (**b**) (**c**) (**d**)

Figure 5. Humpback whale tail images in the dataset. (**a**) 1a36b244. (**b**) 1c5d333f. (**c**) 1efa630a. (**d**) 2c77045a.

As the images show, the tails present different characteristics. The most notable difference a priori is the variation of color between them, which can be presented both in greyscale or in color. The contrast of resolutions, or the presence of elements external to the whale itself, such as annotations, is also a relevant aspect.

There are approximately 25,000 images divided into two sets, one training 9851 images while the other one evaluates 15,600 images. This distribution hinders the subsequent training of the network since the size of the training set is notably smaller to its homologous. Alongside the images, the training set provides a CSV file that collects the labels of each image, check Table 3. Our neural network will deal with the following problem: Identify humpback whale specimens and if there is no record of it, catalogue it as a new whale with the label `new_whale`, counting in total with 4251 different classes or individuals. This paper focuses on an identification problem, that is, since it is an identification challenge, each whale is a separate class and whales were photographed multiple times and one attempts to identify a whale class in the testing set.

Table 3. Header of the training file.

	Image	Id
0	00022e1a.jpg	w_d15442c
1	000466c4.jpg	w_1287fbc
2	00087b01.jpg	w_da2efe0
3	001296d5.jpg	w_19e5482
4	0014cfdf.jpg	w_f22f3e3

Data training samples is the focus of every CNN algorithm whether the training process can achieve effective convergence or whether it will produce overfitting. In this study, the discussion on setting the number of training samples is quite meaningful.

The origin of the 1000-image magic number comes from the original ImageNet classification challenge, where the dataset had 1000 categories, each with a bit less than 1000 images for each class. This was good enough to train the early generations of image classifiers like AlexNet, and so proves that around 1000 images is enough. It seems to get trickier to train a model from scratch in some cases until you get into the low hundreds. The biggest exception is when a transfer learning on an already-trained model is used. It is using a network that has already seen a lot of images and learned to distinguish between the classes, therefore it could usually teach it new classes in the same domain with as few as ten or twenty examples.

What does in the same domain mean? It is a lot easier to teach a network that has been trained on photos of real world objects to recognize other objects, but taking that same network and asking it to categorize completely different types of images like x-rays, faces or satellite photos is likely to be less successful, and at least require a lot more training images.

Another key point is the representative modifier. That is there because the quality of the images is important, not just the quantity. What is crucial is that the training images are as close as possible to the inputs that the model will see when it is deployed. ImageNet consists of photos taken from the web, so they are usually well-framed and without much distortion. A smaller amount of training images that were taken in the same environment that it will produce better end results than a larger number of less representative images.

Augmentations are important too. The training data can be augmented by randomly cropping, rotating, brightening, or warping the original images. TensorFlow controls this with command line flags like flip-left-to-right and random-scale. This has the effect of effectively increasing the size of your training images, and is standard for most ImageNet-style training pipelines. It can be very useful for helping out transfer learning on smaller sets of images as well though. Distorted copies are not worth quite as much as new original images when it comes to overall accuracy, but if dataset only has a few images it is a great way to boost the results and will reduce the overall number of images needed.

The real situation is to try, so if a dataset has fewer images than the rule suggests, do not let it stop you, but this rule of thumb will be a good starting point for planning an approach at least.

4.2. Image Processing

It is necessary to process each image in advance in order to facilitate the extraction of the present features present in it. If all the data is homogenized, this effect is achieved.

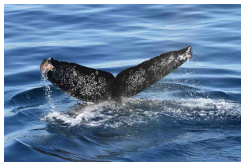
Firstly, the image is converted to a grey scale (Figure 6), ranging from three different channel colors to a single one. There is a double reason behind this conversion—the existence of original images in the black and white dataset and the absence of color characteristics and information. This fact provokes that only the pattern of the tail stands out as useful information.

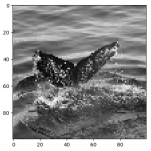
(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 6. RGB image to grey image conversion. (**a**) Original image. (**b**) Red component. (**c**) Green component. (**d**) Blue component. (**e**) Grey image.

The neural network requires the dimension of the input vectors to be fixed, and that is why each image must be rescaled beforehand, resulting, in this case, in 100×100 size matrices with a single channel. The proposal of such downsampling method for inferring the resolutions of images and downsampling images of higher resolution as a preprocessing step in whale recognition. Some authors have demonstrated that image downsampling increases the identification performance of recognition and re-identification [18].

Each pixel has a value ranging between 0 and 255. This amplitude of range, due to the operation of the convolutional networks, allows the incorrect identification of the characteristics for each vector. To correct this disadvantage, it is convenient to normalize the image previously, in a process known as zero mean and unit variance normalization (Figure 7).





$$\begin{matrix} 182 & 179 & 179 & \cdots & 102 & 101 & 104 \\ 202 & 202 & 195 & \cdots & 95 & 99 & 102 \\ 172 & 175 & 179 & \cdots & 114 & 118 & 122 \\ & & & \cdots & & & \\ 64 & 64 & 179 & \cdots & 95 & 100 & 126 \\ 64 & 64 & 64 & \cdots & 30 & 81 & 114 \\ 68 & 72 & 68 & \cdots & 135 & 106 & 151 \end{matrix}$$

(**a**) (**b**) (**c**)

Figure 7. Scale image process. From an HD image to a resized and gray image represented as a matrix (such representation is needed to train the model). (**a**) HD image. (**b**) Gray and scaled image. (**c**) Matrix representation.

The first step is to center the image on the value 0 by subtracting its mean from each value of the matrix.

$$M = \begin{Bmatrix} 67.8595 & 64.8595 & \cdots & -13.140503 & -10.140503 \\ 87.8595 & 87.8595 & \cdots & -15.140503 & -12.140503 \\ & & \cdots & & \\ -50.140503 & -50.140503 & \cdots & -33.140503 & -0.14050293 \\ -46.140503 & -42.140503 & \cdots & -8.140503 & 36.859497 \end{Bmatrix} \quad (2)$$

Subsequently, the range is compressed dividing each value between matrix's standard deviation.

$$M = \begin{Bmatrix} 1.4328924 & 1.3695457 & \cdots & -0.2774693 & -0.21412258 \\ 1.855204 & 1.855204 & \cdots & -0.31970045 & -0.25635374 \\ & & \cdots & & \\ -1.0587456 & -1.0587456 & \cdots & -0.6997808 & -0.0029668 \\ -0.97428334 & -0.88982105 & \cdots & -0.17189142 & 0.7783096 \end{Bmatrix} \quad (3)$$

4.3. Data Augmentation and Imbalance Class

Neural networks are primarily used for classification tasks where the network learns by looking at data points belonging to different classes. Imagine you have a classification problem where you have to identify whether a picture shown to the network has a dog in it or a cat. Now assume that your training set has a total of 10,000 images, 9998 images are of dogs and there is only one image which has

a cat and the remaining image has none of them. This is largely what class imbalance looks like [19], when you have unequal distribution of labeled data in different classes, see Figure 8. A common practice within the classification of images through neural networks is the increase of data present in the dataset (data augmentation). This is especially useful where the proportion of classes is not balanced and there are numerous categories with only one sample.

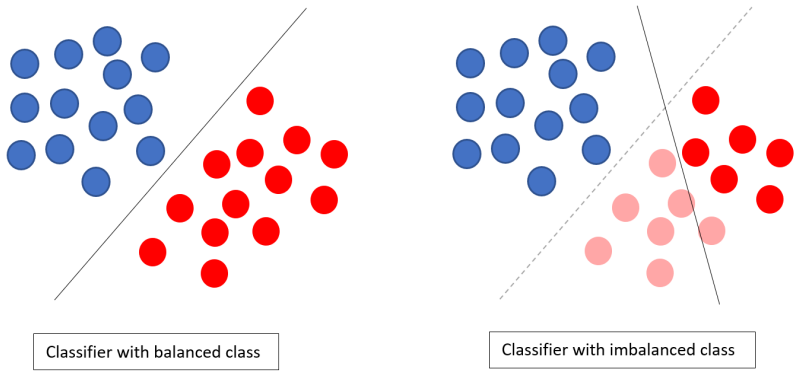


Figure 8. General representation of the problem of classifier trained with imbalance class.

In order to increase the number of data available for the training, a series of processes are performed on an original image, thus generating a series of derived images. Among the possible modifications are the rotation (Figure 9), translation, noise reduction, etc.

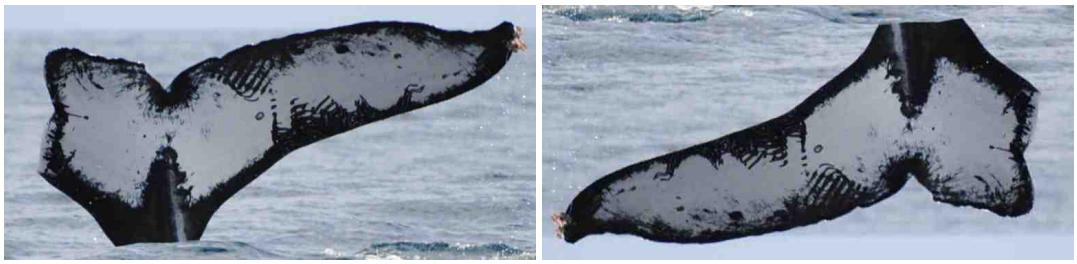


Figure 9. Image rotation.

The effectiveness of this technique lies in the way neural networks understand the images and their characteristics. If an image is slightly modified, it is perceived by the network as a completely different image belonging to the same class.

This decreases the chances of the network to focus on irrelevant orientations or positions while maintaining the relevance of the desired characteristics such as the pattern of the tail in this paper.

An increase of the training set has been made following the following criteria: If a class has less than 10 images, the difference with its original sample number is generated. Therefore, if a specimen had 4 samples, an additional one would be generated from the images associated with a whale, see Figure 10.

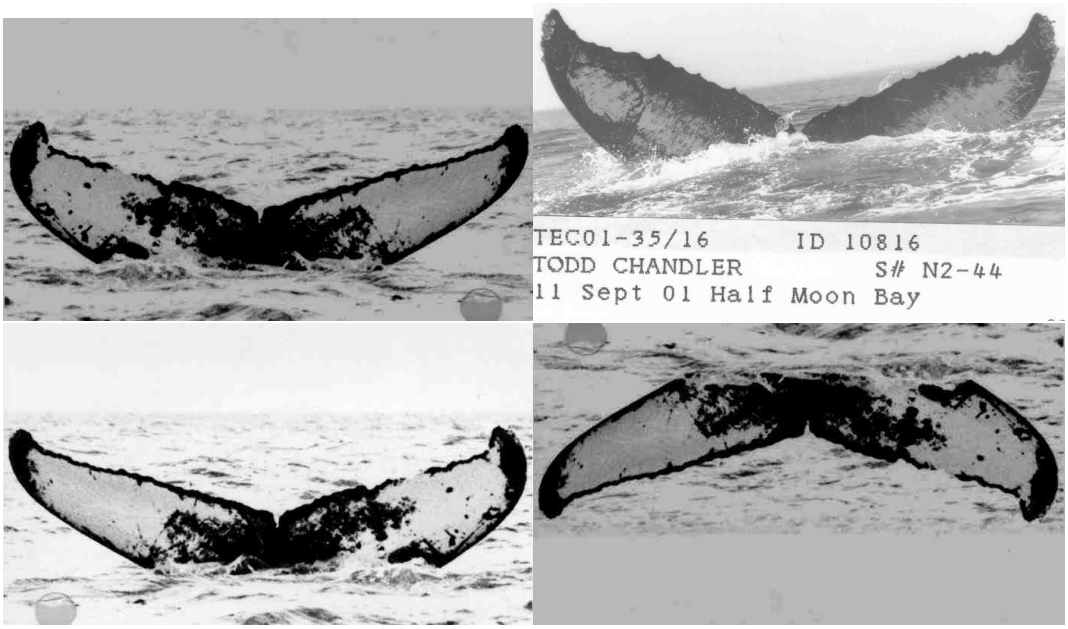


Figure 10. Augment over specimen `w_964c1b3`.

4.4. Neural Network Architecture

The implemented network presents the structure shown in Table 4. A convolutional layer is established, followed by batch normalization [20] and a max pool reduction. Following this fact, another convolutional layer is defined, followed by a reduction by mean or average pooling. This firstly allows extracting the most important characteristics of the image. Secondly, at the next level of depth, smoothing the extraction ensures the lack of loss of relevant information. In both layers, a Rectified Linear Unit (ReLU) activation function is used, defined by:

$$f(x) = x^+ = \max(0, x) \tag{4}$$

where x is the input of the neuron, returning the value x if it is positive or 0 if it is negative. This allows to considerably accelerate the training to be easily computable in comparison to other activation functions such as softmax.

Finally, a conventional multilayer neural network is connected. This network is formed by 450 neurons with dropout (Figure 11), followed by a dense output layer which is completely connected to softmax as an activation function, as well as 4251 neurons that are equivalent to the total number of classes to classify. The first layer receives as input a vector of one dimension, which is achieved compressing the output of two dimensions obtained from the convolutional layers.

The dropout allows ignoring a percentage of input units or neurons during training, in this case 80%. This ignorance disembogues in a neural network smaller than the original and with therefore fewer parameters, which decreases the dependencies between neurons and thus avoids overtraining.

The softmax activation function is used in classification problems with multiple options, as in this case, and it returns the probabilities of each class. The aforementioned function is given by:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{4251} e^{x_j}} \tag{5}$$

It compresses the values between 0 and 1 and makes the sum of all the resulting values equal to 1, being x an input vector of size equal to the number of units or neurons of the layer (4251).

Table 4. Architecture of the implemented network.

Layer (Type)	Output Shape	Param
conv0 (Conv2D)	(none, 94, 94, 32)	1600
bn (BatchNormalization)	(None, 94, 94, 32)	128
activation (Activation)	(None, 94, 94, 32)	0
max_pool (maxPooling2D)	(None, 47, 47, 32)	0
conv1 (Conv2D)	(None, 45, 45, 64)	18496
activation_1 (Activation)	(None, 45, 45, 64)	0
avg_pool (AveragePooling2D)	(None, 15, 15, 64)	0
flatten (Flatten)	(None, 14400)	0
ReLU (Dense)	(None, 450)	6480450
dropout (Dropout)	(None, 450)	0
softmax (Dense)	(None, 4251)	1917201
Total params: 8,417,875		
Trainable params: 8,417,811		
Non-trainable params: 64		

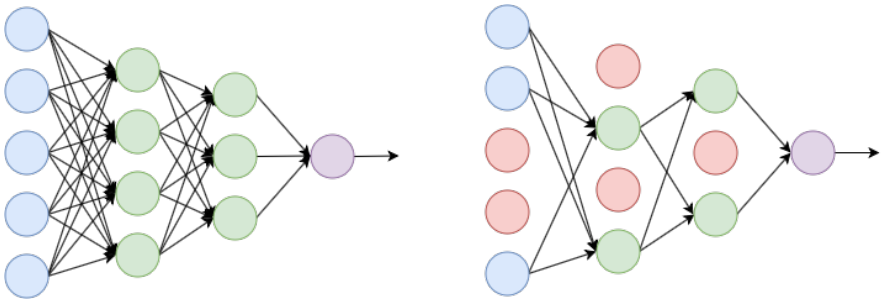


Figure 11. Dropout schema used in convolutional neural networks (Input layer in blue, hidden layers in green, output layer in purple and dropout neurons in red).

4.5. Training

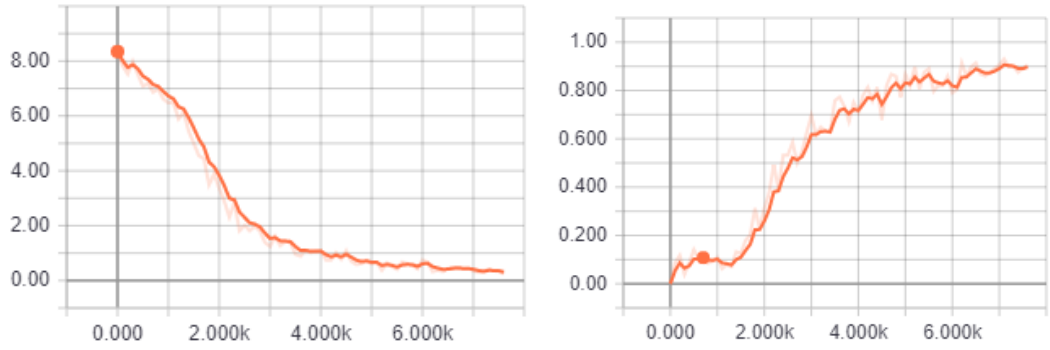
TensorBoard tool (belonging TensorFlow) is used in this section to monitor and visualize our neural network during the training phase, performing a simulation while obtaining the precision and loss graphs.

A key aspect of neural networks is the loss function that reveals about how far away the predictions of a model are ($\hat{y}$) from the real labels y. It is a positive value that improves the performance of the model while decreasing.

The loss function used on the implemented model is Categorical Cross-Entropy (CCE) [21,22], which is especially useful in problems where several excluding classes exist. CCE is preferred for training deep networks with softmax outputs, since: It puts more emphasis on correct classification and can distinguish better between "almost correctly classified" and "totally wrongly classified" than 0-1-loss, It corresponds to maximum likelihood for networks with softmax output and It has an information-theoretical interpretation based on probability and therefore makes sense to use with probabilities (which is the nature of our predictions). The function is given by:

$$H(y,\hat{y}) = \sum_x y_x \log \frac{1}{\hat{y}_x} = -\sum_x y_x \log \hat{y}_x \tag{6}$$

where x is a discrete variable and $\hat{y}$ is the prediction for the real distribution y. The accuracy of the model during the training increases while the value of the loss function decreases, see Figure 12.



(**a**) (**b**)

Figure 12. Loss/Accuracy during the >7000 iterations in the training process. (**a**) Loss during training. (**b**) Accuracy during the training.

The simulation of the training takes as reference the exit of the first dense layer before proceeding to the classification. This does not allow an exact representation of what would be the final output with the total classes, but easily exemplifies the behavior of the neural network and the data during the training phase.

Apart from the performance aspect, it is also important to know what the model has learned. This is necessary so as to ensure that the model has not learned something discriminatory or biased. One way of approaching this problem is from a data visualization perspective. By visualizing how the model groups the data, we can get an idea of what the model thinks are similar and dissimilar data points. This is beneficial in order to understand why a model makes certain predictions and what kind of data is fed into the algorithm.

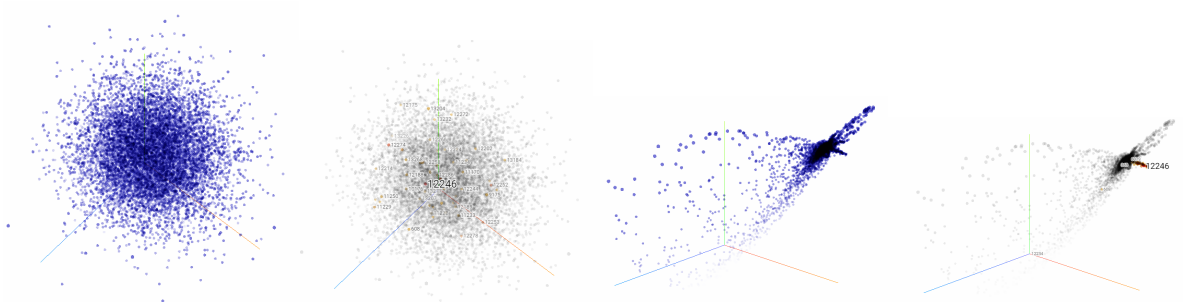
An embedding is essentially a low-dimensional space into which a high dimensional vector can be translated. During translation, an embedding preserves the semantic relationship of the inputs by placing similar inputs close together in the embedding space. Multidimensional space helps to group semantically related items together while keeping the dissimilar items apart. This can prove to be highly useful in a machine learning task. Consider the following visualizations of real embeddings, see Figures 13 and 14.

The Embedding Projector is open-sourced and integrated into the TensorFlow platform or can be used as a standalone tool at projector.tensorflow.org. It offers four well-known methods for reducing the dimensionality of the data. Each method can be used to create either a two- or three-dimensional view for exploration.

- Principal Component Analysis (PCA) is a technique which extracts a new set of variables called Principal Components from the existing data. These Principal Components are a linear combination of original features and try to capture as much information from the original dataset. The Embedding Projector computes the top 10 principal components for the data, from which we can choose two or three to view.
- t-SNE or T-distributed stochastic neighbor embedding visualizes high-dimensional data by giving each data point a location in a two or three-dimensional map. This technique finds clusters in data thereby making sure that an embedding preserves the meaning in the data.
- A custom projection is a linear projection onto the horizontal and vertical axes which have been specified using the data labels. The custom projections mainly help to decipher the meaningful directions in data sets.
- UMAP stands for Uniform Manifold Approximation and Projection for Dimension Reduction. t-SNE is an excellent technique to visualize high dimensional datasets but has certain drawbacks like high computation time and loss of large scale information. UMAP, on the other hand, tries to overcome these limitations as it can handle pretty large datasets easily while also preserving the local and global structure of data.

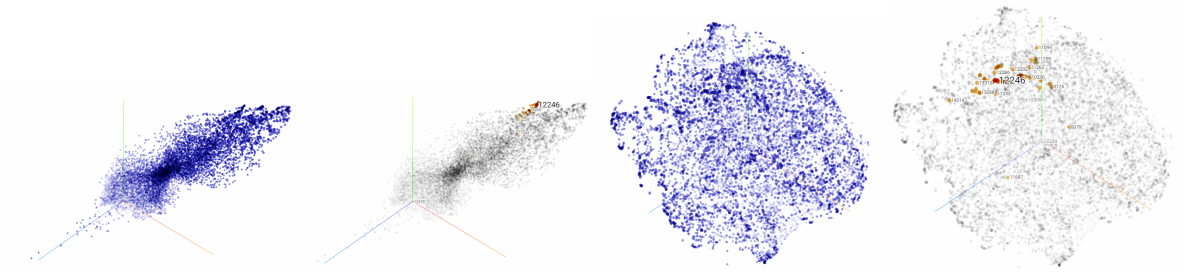
Four situations are differentiated during the execution in the case of study. The beginning, where the network is not trained (Figure 13). The first iterations (Figure 14) where most samples are considered belonging to the new whale class due to their common characteristics among all the images and to their greater proportion in comparison to the other classes. The phase where features from other classes are starting to be noticed (Figure 14a). The final phase, where the network completes its training and differentiates with greater precision (Figure 14c). A group with similar characteristics has been selected to follow its grouping using a t-SNE projection.

The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have an extremely small probability of being picked. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. Note that while the original algorithm uses the Euclidean distance between objects as the base of its similarity metric, this should be changed as appropriate (see [23]).



(**a**) (**b**) (**c**) (**d**)

Figure 13. Different states in the simulation (Figures 13 and 14 are obtained in the training phase, and Figure 13b,d are obtained in testing phase) using a T-distributed stochastic neighbor embedding (t-SNE) projector. (**a**) Init state. (**b**) Init state. (**c**) First iterations. (**d**) First iterations.



(**a**) (**b**) (**c**) (**d**)

Figure 14. Different states in simulation (Figure 14a,c in training, and Figure 14b,d in testing) using a t-SNE projector. (**a**) Last iterations. (**b**) Last iterations. (**c**) Final stage. (**d**) Final state.

4.6. Optimization Algorithm

The learning process is summarized in one idea: the minimization of the loss function by updating the parameters, w, of the network, turning it into an optimization problem. Adam or Adaptive Moment Estimation [24] is an optimization algorithm that allows modifying the learning rate as the training is performed. Specifically, it is a variant of the stochastic gradient descent.

The neural network has a series of hyper parameters among them the learning rate α. This parameter allows the gradient descent algorithms to determine the following point, so:

$$w_j = w_j - \alpha \frac{\partial f(w_j)}{\partial w_j} \tag{7}$$

where w_j is one of the parameters, f the loss function and α the learning rate.

If its value is too small, the convergence, and therefore the learning, will take too long, in the same way as if the rate is sufficiently large the next point will exceed the minimum, making it impossible to reach it. It is therefore true that an adequate value of the learning rate is the key for the correct performance during the training.

The descent of the gradient, see Figures 15 and 16, is an iterative process where the weights are updated in each iteration, which is the reason why, in order to obtain the best result, it is necessary to process the dataset more than once. As long as processing a complete dataset in an iteration is difficult by memory restrictions, it is divided into parts called batches of a certain size, so if the dataset contains 1000 images and a batch size of 100, there will be 10 divisions in total.

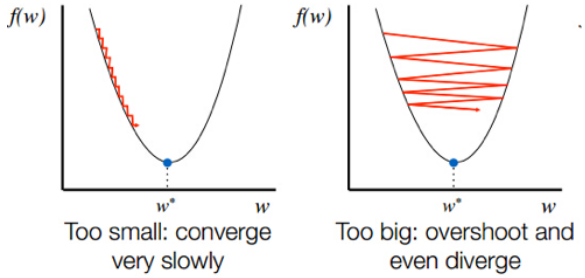


Figure 15. Learning rate α values. A small value produces slow convergence while a high value could overshoot and even diverge.

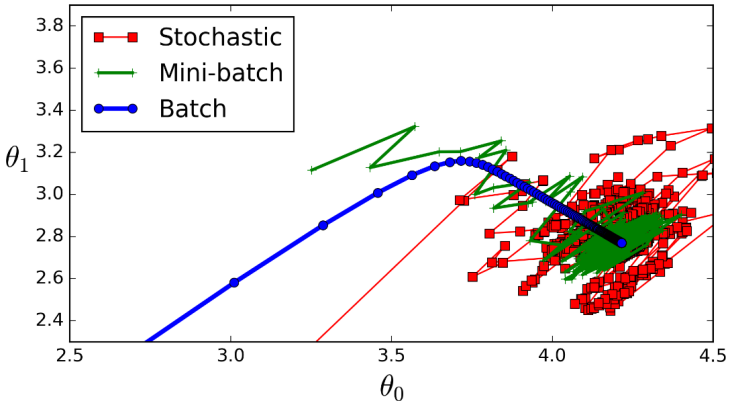


Figure 16. Learning trend using different gradient descent methods (batch, mini-batch and stochastic).

Thus, the learning process is carried out through iterations over the different batches. Depending on the size, the frequency with which the weights of the network are updated varies, so the smaller the size, the higher the frequency of update. Smaller sizes are generally used for the size of the dataset, mini-batch. If the size is equal it is called batch, and if it is equal to 1 it is known as stochastic.

A batch size of 128 over a total of 9850 images of the training set has been defined, which produces a total of 77 iterations for the achievement of the totality of the data. In addition, 100 epochs or tours on the complete dataset have been established.

The dataset subjected to an increase during the intermediate stages of development has a total of 124,065 images, a massive number compared to the original 9850 images. This fact produces a total of 970 iterations per epoch, having been limited to 50.

4.7. Label Coding

Due to the mathematical operations they perform, most of machine learning algorithms do not allow working on categorical data (labels), requiring numerical values. The encoding technique known as One-Hot has been used for the implementation, where a vector is generated for each category where only one of the values can be 1, being the rest 0.

The coding process involves two operations:

$$V = \begin{bmatrix} Blue & Red & Green & White & Black \end{bmatrix} \tag{8}$$

First off, the categories are converted into a whole value $V = [0, 3, 4, 1, 2]$. Being subsequently codified as One-Hot vectors.

$$M = \begin{Bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{Bmatrix} \tag{9}$$

Kaggle submissions are evaluated according to the Mean Average Precision @ 5 (MAP@5).

$$MAP@5 = \frac{1}{U} \sum u = 1^{U} \sum k = 1^{min(n,5)} P(k) \times rel(k) \tag{10}$$

where U is the number of images, $P(k)$ is the precision at cutoff k, n is the number predictions per image and $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant (correct) label, zero otherwise.

Once a correct label has been scored for an observation, that label is no longer considered relevant for that observation, and additional predictions of that label are skipped in the calculation. For example, if the correct label is A for an observation, the following predictions all score an average precision of 1.0.

5. Results and Evaluation

One of the main drawbacks during training is the overfitting or under fitting of the data. If the training data does not contain enough information, the system will not be able to correctly recognize images. On the contrary, if a model is over trained it will increase the precision on the training set but it will be incapable of generalizing when new data is received.

That is why the training data is usually split into two; a training set and an evaluation set. The test set must be representative of the total dataset and must be large enough to be effective, usually between 70% and 80% of the original size. There may also be a third set of tests to check the final version of a model, providing new data and ensuring an unbiased evaluation.

This allows to obtain closer results to reality since the data of the validation set remains hidden from the network during training and thus avoids overfitting.

In the implemented application, the validation set has been established at 10% of the total, while the training set was 90%. It has only been divided during the phase of development, by increasing the data set as described, to adjust the network's hyper parameters and study their performance, using the entire original dataset in advanced versions of the model and not the increased dataset. This is due to two factors: the considerable increase in both the time of image pre-processing and in the training time of the neural network as well as the improvement in the percentage of final success of only 1%. However, if the goal is to achieve the maximum precision, the data increase in the final version is to be incorporated, in addition to a more aggressive processing on the data.

The decisions that justify the need to establish a process of image enhancement are several. First, the asymmetrical distribution of the data set, existing categories with a single example, which would result in the presence of values exclusively in one of the sets when dividing them during the evaluation results in counterproductive. The second and most important reason is the existence of a set of test data, with unknown values, provided by Kaggle [25], allowing us to generate a CSV file with the results, upload them to the competition and compare the performance of the model. Therefore, this set of tests will be used as a validation set in the final versions of the application.

The whole process of designing, implementing and training a neural network is summarized in the prediction stage, where it is expected that the network responds with a certain precision to a series of entries. The model implemented reaches a precision of 0.4407 that can be considered promising, due to the characteristics of the selected dataset (4251 asymmetric classes) and to the hardware limitations that could exist in terms of memory. If we compare the public classification of the Kaggle competition, the developed model would be placed in the 54th position out of a total of 528 participants, with the highest probability being 0.78563 and the second 0.64924, see Table 2.

To extract the results and the resulting classes it is necessary to reverse the labels coding process. In order to achieve this goal, the vector of predictions of length 4251 for each input is obtained, where a probability is established for each class. Then, the five highest probabilities are selected and their positions in the array are reverted to the original label.

This paper tried to minimize resources using an image downsampling and a small architecture. To perform calculations efficiently and save resources, the algorithm process must be simplified, or the size of the data file must be strictly controlled. Sensitivity analysis of convolutional neural networks is related to the concept of adversarial and rubbish examples. Adversarial examples are slightly modified images which show a significant change in the model output compared to the original image. The idea of rubbish examples studies random noise images that lead to arbitrary classification decisions although their appearance can not be related to the particular object category or any natural image at all. See Figure 17 corresponding to the sensitivity analysis of proposed model.

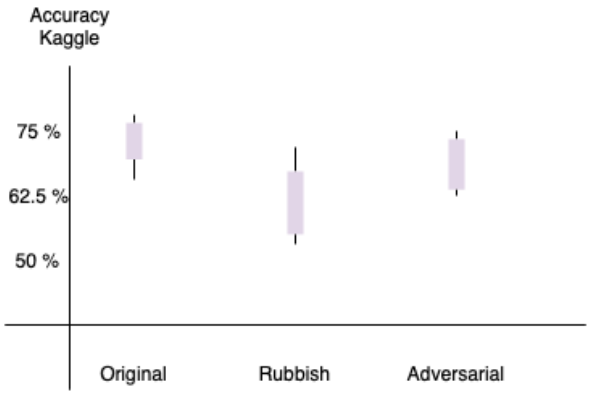


Figure 17. Sensitivity analysis showing candle chart of original method versus rubbish and adversarial modifications.

As for the input data preprocessing, image conversion, feature extraction or band combination are all effective ways to increase the value of the input data and reduce noise. Collect labeled data is expensive. For images, proposed method applies data augmentation with simple techniques like rotation, random cropping, shifting, shear and flipping to create more samples from existing data (other color distortion includes hue, saturation and exposure shifts could be also applied). It is possible to add training data with non-labeled classes. For samples with high confidence prediction, they have been added to the training dataset with the corresponding label predictions. Another interesting approach consists of building a hierarchy of classifiers for image classification with a large number of classes Subcategory classification is another rapidly growing subfield of image classification. Using object part information is beneficial for fine-grained classification. Generally, the accuracy can be improved by localizing important parts of objects and representing their appearances discriminatively. Part annotation information is used to learn a compact pose normalization space. All the above-mentioned methods make use of part annotation information for supervised training. However, these annotations are not easy to collect and these systems have difficulty in scaling up and to handle many types of fine-grained classes. To avoid this problem, it is possible to find localized parts or regions in an unsupervised manner.

6. Conclusions and Further Work

The original approach to this paper has been the study and deepening in the fields of artificial intelligence, deep learning and convolutional neural networks. That is why, despite the fact that the accuracy achieved does not imply a value close to the state of the art, the different methods and techniques used during the development of systems that allow recognizing images have to be explored. Likewise, the influence and importance of previous work on the neural networks themselves has been analyzed, such as the processing of images and the comprehension of data sets.

During the development of this work, several of the original design decisions were altered. The usage of a single library for neural networks, TensorFlow, was raised, but in later versions Keras was also due to two factors: its previous inclusion in the TensorFlow library and the possibility of achieving a greater extent to offer a higher abstraction approach at a programming level compared to TensorFlow.

It has been possible to verify how the importance of the performance of a neural network does not reside exclusively in the architecture that it may have, but that the data set available substantially alters the final result, being a balanced dataset and a large number of samples a critical aspect. The main complications found are divided between the two most relevant goals: the pre-processing of images and the model. The selected dataset has greatly weighed the final accuracy of the application, requiring a much more aggressive data increase than has currently been done. Nonetheless, despite not providing the best results, it offers a clear vision about the importance of the available data, as well as the functioning of the neural networks in terms of the perception of images and how they are treated. In terms of the problems that have occurred in the implementation of the model, there are large hardware demands, especially in terms of RAM memory, that is not available in the equipment used, which has made it impossible to implement a more efficient architecture.

The case study of this paper, the identification of whales by their tail pattern where there are numerous classes belonging the same species, shares similarities with facial recognition applications. Within this field, one of the techniques with the greatest impact and performance are the Siamese networks and the use of triplet loss [26].

However, as promising, the triplet loss has not been implemented due to the memory restrictions, the hardware used and the required large amount of memory and data sets (not available). Nevertheless, the use of a conventional CNN with the available data sets can be considered a valid, interesting and promising method for humpback tail recognition.

Author Contributions: Conceptualization, J.M.L. and N.G.B.; software, L.F.d.M.L. and A.A.A.; investigation, J.M.L., A.A.A. and L.F.d.M.L.; supervision, N.G.B.; all authors analyzed the data and wrote the paper. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This research has been partially supported by project Seguridad de Vehículos AUTOmóviles para un TRansporte Inteligente, Eficiente y Seguro. SEGVAUTO-TRIES-S2013/MIT-2713. (http://insia-upm.es/portfolio-items/proyecto-segvauto-tries-cm/?lang=en).

Conflicts of Interest: The authors declare no conflict of interest. This research has no funding sponsors so they had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson Education Limited: Kuala Lumpur, Malaysia, 2016.
2. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, UK, 2016; Volume 1.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef] [PubMed]
4. Qin, Z.; Yu, F.; Liu, C.; Chen, X. How convolutional neural network see the world—A survey of convolutional neural network visualization methods. *arXiv* **2018**, arXiv:1804.11191.
5. Sewak, M.; Karim, M.R.; Pujari, P. *Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python*; Packt Publishing: Birmingham, UK, 2018.

6. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
7. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
8. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Hasan, M.; Esesn, B.C.V.; Awwal, A.A.S.; Asari, V.K. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *arXiv* **2018**, arXiv:1803.01164.
9. Theckedath, D.; Sedamkar, R.R. Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks. *SN Comput. Sci.* **2020**, *1*, 79. [CrossRef]
10. Yan, K.; Wang, Y.; Liang, D.; Huang, T.; Tian, Y. CNN vs. SIFT for Image Retrieval: Alternative or Complementary? In Proceedings of the 24th ACM International Conference on Multimedia; Association for Computing Machinery, Amsterdam, The Netherlands, 15–19 October 2016; pp. 407–411. [CrossRef]
11. Pouyanfar, S.; Sadiq, S.; Yan, Y.; Tian, H.; Tao, Y.; Reyes, M.P.; Shyu, M.L.; Chen, S.C.; Iyengar, S.S. A Survey on Deep Learning: Algorithms, Techniques, and Applications. *ACM Comput. Surv.* **2018**, *51*. [CrossRef]
12. Komkov, S.; Petiushko, A. AdvHat: Real-world adversarial attack on ArcFace Face ID system. *arXiv* **2019**, arXiv:1908.08705.
13. Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Li, Z.; Gong, D.; Zhou, J.; Liu, W. CosFace: Large Margin Cosine Loss for Deep Face Recognition. *arXiv* **2018**, arXiv:1801.09414.
14. Mokhayeri, F.; Granger, E. Video Face Recognition Using Siamese Networks with Block-Sparsity Matching. *IEEE Trans. Biom. Behav. Identity Sci.* **2020**, *2*, 133–144. [CrossRef]
15. Cheng, G.; Yang, C.; Yao, X.; Guo, L.; Han, J. When Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via Learning Discriminative CNNs. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 2811–2821. [CrossRef]
16. Sanakoyeu, A.; Tschernezki, V.; Buchler, U.; Ommer, B. Divide and Conquer the Embedding Space for Metric Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
17. Feng, F.; Wang, S.; Wang, C.; Zhang, J. Learning Deep Hierarchical Spatial-Spectral Features for Hyperspectral Image Classification Based on Residual 3D-2D CNN. *Sensors* **2019**, *19*, 5276. [CrossRef] [PubMed]
18. Obara, K.; Yoshimura, H.; Nishiyama, M.; Iwai, Y. Low-resolution person recognition using image downsampling. In Proceedings of the 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), Nagoya, Japan, 8–12 May 2017; pp. 478–481. [CrossRef]
19. Buda, M.; Maki, A.; Mazurowski, M.A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Netw.* **2018**, *106*, 249–259. [CrossRef] [PubMed]
20. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
21. Theano. Categorical Cross Entropy. Available online: deeplearning.net/software/theano (accessed on 28 April 2020).
22. Zhang, Z.; Sabuncu, M.R. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. *arXiv* **2018**, arXiv:1805.07836.
23. Wattenberg, M.; Viégas, F.; Johnson, I. How to Use t-SNE Effectively. *Distill* **2016**. [CrossRef]
24. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
25. Kaggle. Humpback Whale Identification Challenge. Available online: kaggle.com/c/humpback-whale-identification (accessed on 28 April 2020).
26. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA, USA, 7–12 June 2015; pp. 815–823.