Current Natural Sciences & Engineering
Volume 3, Issue 1, 2026, 964-976

Deep Learning for Marine Animal Recognition: A Comparative Study of CNN Architectures and Triplet Loss Optimization

Mir Ihrar Ali

Trine University, United States

Received date:22/09/2025, Acceptance date: 17/06/2026

DOI: *http://doi.org/10.63015/3ai-2503.3.1*

Corresponding Author: ihrar2022@gmail.com

Abstract

The conservation of marine biodiversity has become. increasingly critical in the face of climate change and environ- mental degradation. Identifying individual animals is essential for understanding species behavior, migration patterns, and population dynamics, yet traditional methods remain time-intensive and reliant on expert observation. This study leverages deep learning to automate individual whale and dolphin identification using data from the Kaggle Happywhale challenge. The performance of several Convolutional Neural Network (CNN) architectures is evaluated, including ResNet, DenseNet, EfficientNet, and In- ceptionV3, and explore the combination of Softmax classification with a semi-hard triplet-loss approach. The results reveal that InceptionV3 achieves superior accuracy, while the hybrid Softmax and triplet-loss method offers limited benefits, hindered by dataset challenges such as the prevalence of hard triplets. These findings emphasize the need for refined loss strategies to handle unbalanced datasets in wildlife identification. This work highlights the potential of machine learning to revolutionize marine conservation efforts by enabling scalable and efficient individual recognition systems.

Keywords: Deep Learning, Kaggle Happywhale challenge, Marine biodiversity, Triplet-loss, Machine learning, Marine Conversation, Convoluational Neural Network Architectures (CNN)

1. Introduction

As climate change and environmental pollution intensify, nature conservation becomes increasingly important. A part of nature conservation is also the study of animal behavior, migration routes, and population density to better understand the problems and obstacles certain species are facing to tackle and prevent these problems to the best of the abilities. To do this, one must be able to differentiate between individuals of a species. In humans, this is easily done by face or fingerprint recognition. But what about animals? To date, researchers manually differentiate them by the shape and markings on their tails, dorsal fins, heads, and other body parts. This is time-consuming and difficult work since it takes the eye of a good researcher and much time to identify, match, or tell individual animals apart. Hence the question arises: When it is possible to use automated identification for humans, is there a similar approach possible for animals? The recent advances in facial recognition were mostly commercially motivated. Photo identification for animals seems like a less lucrative endeavour which moves it out of the research limelight. However, a technique like this could simplify the analysis of wildlife and therefore, nature conservation significantly. This is why it is decided to take on the Kaggle Happywhale challenge [1]. The goal of this challenge is to train a machine learning model to identify whales and dolphin's individuals. The competition model will be used on happywhale.com [2], a research collaboration platform that aims at increasing global understanding and caring for marine animals. This challenge is used to look for relevantly easy approaches that could be implemented by conservation organizations to automatically detect individuals on their datasets as well [3].

II. Kaggle happywhale challenge

In the following chapter, the Kaggle Happywhale challenge is explained. This includes goal setting and a comprehensive analysis of the dataset.

A. Goal

The Happywhale Challenge is a research prediction com- petition open to everybody with a Kaggle account. Its goal is to build a machine-learning model that can reliably recognize individual whales and dolphins. The model should also be able to classify individuals it has never seen before as" new." Such a model would save experts, who - up until now - must analyze the images manually, a tremendous amount of work.

B. Data

The data [1] to be used for this challenge is split in to training and testing data. The training data consists of 51 thousand JPEG images of whales and dolphins. The training images are labelled with the according species and a unique individual identification string. The testing data consist of 28 unlabelled images containing some new whales and dolphins which are not present in the training data. Typical examples from the Happywhale dataset are shown in **Fig 1**.

In total there are about 15 thousand different individuals and 30 different species. This data was manually curated by researchers from all over the world.

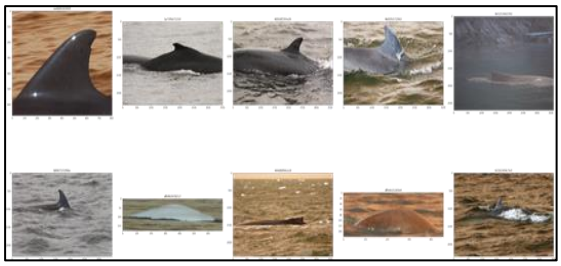

Figure 1: Typical images from the data set

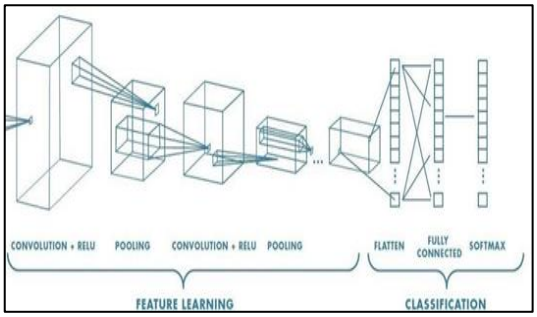

Figure 2: Vanilla image classification [7].

With a median image shape of approximately (3000,1500), most images have high resolution. This makes unique patterns and markings of each individual identifiable by a neural network, even though most images only contain small parts of the marine animals, such as dorsal fins and parts of the backs. Some further analysis shows that 59% of all individuals have only one image of themselves. They make out 18% of all images in the data set [4]. 5% of all individuals have at least 10 images of themselves. These are 47% of all images in the data set. This is a large unbalance in the number of images per individual and will make training the network much harder [5].

III. Model selection

At first glance, one could think that the problems like the Happywhale challenge are generic multi-class classification task. This would mean that the model should be able to classify an input image of an individual correctly as just this individual. For tasks like this, the default approach is the following: Feed the input image to some Convolutional Neural Network (CNN) architecture, which outputs an embedding of this image. This embedding can then be flattened and fed into a Dense Layer, which will output the predicted class. [6]. The conventional image-classification pipeline is illustrated in **Fig 2**.

A. CNN architecture

As a first step of model selection popular CNN architectures are compared to test. It is often a good idea to bet on architec- tures which already solidified their position on the grandstand of the machine learning domain of interest. Therefore, it is important to look at the most implemented models for image classification tasks. Looking at this list, a Residual Neural Network (ResNet) is decided, a Densely Connected Neural Network (DenseNet), an EfficientNet, and an InceptionNet. [8]. All these networks are notorious for being capable of classifying images with very high accuracy while being somewhat computationally efficient (at least compared to other machine-learning approaches).

Additionally, there are countless pre-trained models with easy access points available for these architectures. Especially the models pre-trained on ImageNet [9], one of the world's largest labelled image databases, are extremely powerful for image classification tasks and, therefore, of great interest to us.

It has been decided to try out the following architectures with pre- trained weights from ImageNet: ResNet50, ResNet50V2, DenseNet101, EfficientNet, and InceptionNetV3. We also trained ResNet50 us. More about that can be read in Softmax Model.

B. Softmax activation

The activation function of the last Dense Layer in this pipeline is usually a Softmax function [10], which computes a probability distribution σ over the image membership for each class K. The Softmax function is defined in Equation (1).

$$\sigma(Z_i) = \sum_{j=1}^{K} \frac{e_{zi}}{e_{zj}} \qquad (1)$$

for i = 1, . . . , K and $(z_1, ..., z_K) \in R^K$

K is determined by the constant number of neurons in the final Dense Layer. Unfortunately, we are working with a

dynamically changing number of classes - Every individual whale in the ocean (let alone have images of them) is not known. For now, this eliminates the Softmax activation function as a suitable approach to the task.

C. Triplet loss

Luckily, the domain of metric learning offers us an alternative approach that is more suitable for the problem: Triplet loss - originally presented in the FaceNet [11] paper.

The goal of triplet loss is to learn an embedding space in which similar sample pairs stay close together and dissimilar ones are far apart. It does this by computing a distance between an anchor image a and a positive sample p, as well as between a and a negative sample n. a and p are different images of the same individual, while n is an image of a different individual. A visualization of triplet loss is shown in **Fig 3**.

Figure 3: Triplet loss visualized [11].

The learning goal is then to minimize the distance between a and p and maximize the distance between a and n at the same time [12]. Triplet Loss for some triplet (a, p, n) in the embedding space is defined as: The triplet-loss objective is given in Equation (2).

$$L_{triplet} = \max(0, d(a, p) - d(a, n) + \epsilon) \quad (2)$$

where d denotes the L2 norm (calculated from the Euclidean distance between the embeddings) and the margin parameter ϵ denotes the minimum offset between distances of d(a, n) and d(a, p). This prevents the network from gaming the function and circumventing its actual intention. If there was no ϵ, the network could just map the complete data set onto the same point in the embedding. This would cause all distances and the loss to be 0. The margin ensures that, in this case, the loss would still be positive. [13]

1) Triplet Mining: Of course, triplet loss is far from being a perfect method of choice as well. One of its main limitations is that during one comparison, only one negative example n is compared with the anchor a. The disregard of all other n could lead to an embedding space where some dissimilar pairs are still near each other - just because they were not compared against each other. [14] This is why, in order to train a triplet loss model successfully, one has to pay close attention to the composition of each triplet. To learn the model it should pick an n that is not obviously a different individual than a. To do this, a challenging n that is closer to the a than the positive sample p is chosen.

$$d(a, n) < d(a, p) \quad (3)$$

This is called hard triplet mining. In theory, it should guarantee optimal learning success. However, it is prone to get stuck in local minima and the samples are rather hard to select from the data set. An approach that tries to solve this is semi-hard triplet mining. Here the triplets are chosen in a way, that n is not closer to a than p, but the function still has a positive loss. The hard-triplet condition is expressed in Equation (3).

$$d(a, p) < d(a, n) < d(a, p) + \epsilon \quad (4)$$

The distance between a and n is still in the range of margin ϵ. This way, the model can still learn but is more unlikely to end up in local minima.

It is needed to specify the triplet sampling process even further by choosing between offline and online triplet mining. With offline mining, the triplets are generated at the beginning of each epoch. The embeddings are computed on the training

set, and then only (semi-) hard triplets are selected. This technique is rather inefficient

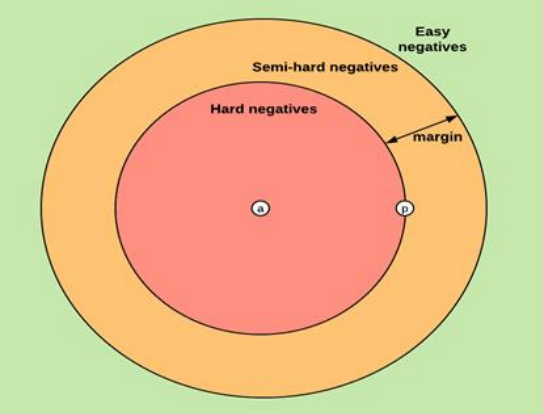

Figure 4: Different kind of Negatives visualised [15].

because it needs to do a full pass on the training set for each epoch and update the triplets. Doing online mining, samples are produced for each batch of inputs. Within a batch of B examples, this can find a maximum of B3 triplets. Many of these triplets are not of shape (a, p, n) and are therefore not valid. Still, it does not require updating all samples offline, and more samples are created for a single batch. This makes the approach way more efficient [15].

2) Shortcomings:

There remain problems with the triplet loss function and contrastive learning approaches in general. It is hard to guarantee that embeddings of the same individual will be pulled together in Euclidean space (expansion prob- lem). In addition, there is the so-called sampling issue [13]: Online triplet mining is hard to implement for very unbalanced data sets like the Happywhale data set. There are modern alternatives to triplet loss such as Cos- and ArcFace, which are based on angular distance and margin. They (partially) solve the above problems, but this nevertheless wanted to try out vanilla triplet loss. Since the paper should also be of some educational value and explain the concepts well to other students, triplet loss is also more suitable because it is less advanced and needs fewer prerequisites to understand. Because of the mentioned advantages in Triplet mining, this also decided to implement online semi-hard triplet mining. There is still one disadvantage of triplet loss which I did not talk about up until now: Evaluation of training progress is computationally expensive. As it is required to evaluate the useful- ness of the different CNN architectures, it needs to compute an evaluation metric (i.e. accuracy) on the validation set after each training epoch. To do this, the embedding of every image from the validation set needs to be compared to all embeddings of the training set. This means there have to be many point- wise distance calculations in a high dimensional Euclidean space calculated. To be specific, it needs to do $37598 * 4177$ computations in a 256-dimensional embedding space for each epoch. Furthermore, the convergence time for triplet loss models is said to be rather high. Doing this for more multiple hyperparameter settings for each architecture is not feasible with the computational resources. Is there a cheaper workaround? The semi-hard-triplet condition is expressed in Equation (4).

D. Combining Softmax and Triplet Loss

It is already found out that the Softmax activation function is not useful for the final model but maybe still utilize it for some architecture comparison. Computing the accuracy of a Softmax-based model is a breeze compared to models based on contrastive loss.

This can imagine that an architecture that performs well over a finite-class classification task using Softmax is also likely to perform well for a classification problem with a dynamical number of classes like Happywhale. So why not try this out? The approach used here will now look like the this:

I. *Research Questions*

Knowing all this, it can motivate the following hypotheses that it needs to investigate further:

1) Is a 'lazy' approach of just using prebuild & pre trained architectures and preconfigured loss functions sufficient to produce significant results in individual clas- sification?

2) Is Softmax classification training success a useful indica- tor for triplet loss training success? To be specific: Does the best performing CNN architecture under Softmax also show the best performance under triplet loss?

3) Can we reduce triplet loss training time with network weights pre-trained on Softmax classification?

II. Data pipeline

A. Preprocessing

This applies several preprocessing steps to get the data into an optimal shape for the model. There is also created One-hot labels.

1) Normalization: Normalization ensures that all images have the same format.

One of the first steps is downsizing the images. Although the mostly high-resolution pictures allow us to see patterns and unique marks of the whales in great detail, this has to downsize the images enough that the memory can handle it. It is also useful to have every image in the same quadratic shape. It has been selected the shape (224, 224) since it is small enough not to run into memory constraints but still large enough for the model to recognize some details. The pretrained ImageNet-models also use this input size. The automatic interpolation of Tensorflow ensures that the images still look somewhat natural and not too distorted.

The pixel values have been also converted from integers to floats to make sure that Tensorflow can work with them. The float values were fixed between -1 and 1.

2) Foreground Extraction: Since the images often contain much ocean in the background and the model should definitely not take this as feature, it would be a good idea to crop the background such that only the whale on white background is in the image. An example of failed foreground extraction is shown in **Fig 5**.

In this mainly used preexisting algorithm from the Kaggle Humpback Whale challenge [16] for this. Unfortunately, large parts of whales were cropped in many images.

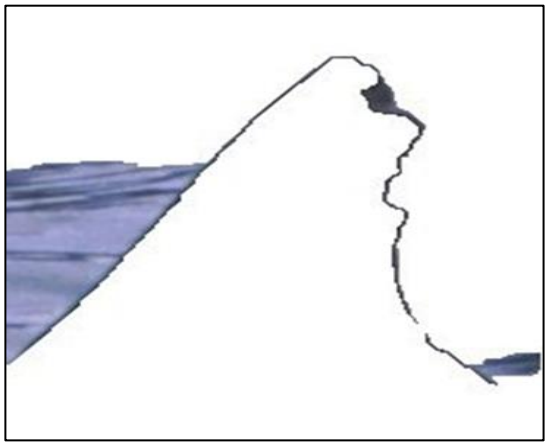

Figure 5: Failed foreground extraction

This is why We decided not to include this preprocessing step in the pipeline.

3) Data Augmentation: Data Augmentation has the goal of increasing the size of the data set.

This is especially useful for obtaining multiple images of whale individuals for which only one or two images exist. However, it was decided against data augmentation because it significantly impacted training speed - even though better results might be obtained with it.

It was decided to incorporate classic augmentation steps, such as random image flipping, contrast, and brightness, to introduce more randomness to the data. This

might lead to a more regularized model performance. Being nit-picky, these steps also count under ”normalization,” but it needed to lay down the reasons for not doing data augmentation in a separate paragraph.

B. Dataset creation

Since online mining samples the triplets within one batch, to ensure on batch creation that there are sufficiently many different pairs of anchors and positives per individual within a batch. As analyzed earlier, there are many individuals with only one or two images in the Happywhale data set.

Suddenly, this turned out to be a quite interesting constraint satisfaction problem (CSP).

First, what to do with the individuals with only one image, which makes up 59% of all individuals and 18% of the dataset was considered. One approach could be to use image augmentation and then pair the individuals with an altered version of their images. However, since the model needed way too much time already, it has been decided to use these individuals not for training but for another process later.

1) Smart Batches Algorithm: To solve this CSP, it came up with an algorithm that can easily be generalized for other triplet loss applications.

First and foremost, it is decided to only accept an even batch size, which relaxes the CSP quite a lot.

It is chosen 64 as a batch size simply because it yielded the best performance - in terms of training time - on the setup (Nvidia GTX 1080).

Let’s consider N images of different animals, with at least two images per animal. It also has an even batch size b. It needs to divide by the number of batches, which is ⌈N/b⌉ = M. The last batch will not have size b but rather M %b = L. Now, the N images over these batches to be distributed in such a way that there is never a single image of an individual in a batch and that one batch never contains images of only one individual.

The way this problem is solved by creating two separate pools. In the one pool, all the images of animals are put in with an even number of images, and in the other, all the animals with an uneven number of images. Then it was iterated through every animal in the uneven pool.

The first three images of every animal has been and keep them in the uneven pool. The even amount of rest can be thrown into the even pool.

For example, if a whale has 17 images, it will keep the first 3 pictures in the even pool and put the next 17 − 3 = 14 pictures into the even pool.

Now, only triplets have left in the uneven pool.

For a healthy amount of randomness, it has been shuffled them around. For reproducibility, need to set a seed beforehand.

There is a small special case when N - and subsequently L-is uneven. The even pool still contains an even number of images. So, the only source for the unevenness of N can be in the uneven pool. This would mean that an uneven number of triplets. In this case just took the first triplet and putted it into the last batch.

This was enforced that:

1) Every batch has an even amount of space left.

2) There is an even number of images in the even pool

3) There is an even number of triplet-pairs in the (initially) uneven pool

This has been solved the CSP with this algorithm:

1) Combine the triplet pairs into pairs of 6 each.

2) Distribute them over the batches.

3) Form positive triplet-pairs of 2 in the even pool and shuffle them (with seed).

4) The batches with the triplet-pairs of 2 were filled up.

Because it has at least two or three images of every animal in a batch, as there is never a single image of an individual in a batch. Because it was shuffled the first pairs of two, one batch most likely never contains images of only one individual. If this is not the case, it can be simply chosen another seed. To now keep this order, it will make sure to not shuffle not to shuffle at a later step in the pipeline.

III. Softmax model

For training the Softmax model on species classification, it has been decided to use all available training images. For Softmax standards, 51000 images are not that many. This is why it is decided on a train/validation/test split of 0.8/0.1/0.1.

To train, validate, and evaluate or model predicting all 30 species, it is considered to have at least one image of every species present in every dataset. To achieve this, the Smart Batches Algorithm, is used simply because it is likely to create such a split. It named all functions with different seeds and quickly found the first valid candidate, ”5”, which It is then used for all Softmax models.

A. Configurations and Hyperparameters

All architectures loaded with the includetop setting as false and set pooling to max. It then simply put a dense layer with 30 neurons for the species and a Softmax activation function on top. This chose Adam as an optimizer with a learning rate of 0.001 and set Categorical Cross Entropy loss.

Nothing special - standard image classification settings.

B. Evaluation

1) ImageNet vs Random Weights: This first wanted to test whether loading the models with weights, which were pre-trained on the ImageNet dataset, would lead to faster convergence was trained. To do this, ResNet50 for 10 epochs each. Once with the pre-trained ImageNet weights and once with randomly initialized weights. The ImageNet weights beat the other in every metric. The comparison between ImageNet and random initialization is presented in **Fig 6**.

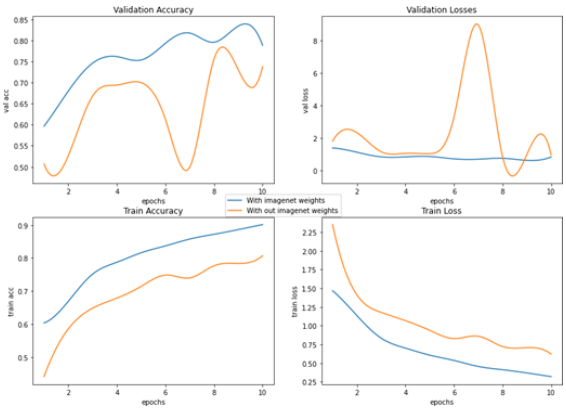

Figure 6: ImageNet vs random performance

Because of this, it has been decided stick to models with ImageNet weights for further analysis.

2) Comparing CNN architectures: Further, the performance is tested of several CNNs. As mentioned in CNN architecture, too many parameters not required so it could still train them locally. All architectures for 10-15 epochs were tested.

These were the results:

Table I: ACC10 denotes the validation accuracy after 10 training epochs. s =seconds.

Architectures	***ACC_{10}***	***s/epoch***	***ACC_{10}*/time**
ResNet50	81,9	532	0.15
ResNet50V2	85,4%	411	0.21
DenseNet100	89,7%	506	0.18
InceptionNetV3	90%	271	0.33

This has been also tried to use several versions of Efficient Net, but sadly, the kernel always died quickly.

As it can see, InceptionNetV3 had - compared to the others

An outstanding performance considering training time. This is why the model is selected, which further will trained and of which recycled the weights later for the Siamese Model Training.

It has been then decided to train for 70 epochs in total and evaluate the best model checkpoint with the best validation accuracy on the test dataset. This has been achieved a test accuracy of 94.45% and a k3-accuracy of 98.79% on the test data, which was quite impressive.

It is analyzed the accuracy for every species and found that, as one might expect, the model was good at recognizing species with many images while struggling with the uncommon ones.

For example, it completely failed to detect the least common species (" Frasier Dolphins" with only 14 images). Strangely, the best performing species were the" Commersons Dolphins", with only 90 images in the database. It assumes this has to do with the unique, black-and-white look of their dorsal fins and the very high image quality of the images, which you can see here.

IV. Triplet loss model

A. Specific Dataset

Since the Kaggle Challenge already provides a testing mechanism, a validation dataset was chosen. It was already disregarded more than half of individuals, so to throw away more data was not required. Hence, it was decided to only consider individuals with > 3 images. It was then used a method very similar to the Smart Batches Algorithm:

1) Iterate over all the individuals with > 3 images.

2) Take the first two images and put them to the side to make sure it don't disregard individuals.

3) Shuffle the remaining images (with a seed).

4) According to the split ratio S (We chose 0.1), take the first $\lceil S * \text{len(remaining images)} \rceil$ individuals as your validation dataset.

5) Use all the other images as the training dataset.

Then the Smart Batches Algorithm called and do the remaining standard preprocessing steps. The seed 0 is being used.

B. Configurations and Hyperparameters

To test the hypotheses, these models are trained:

a. InceptionV3 Model with the weights pre-trained on the Softmax species.

b. InceptionV3 with imagenet weights.

c. Resnet50V2 with imagenet weights.

d. For simplicity, it is referred to the models from now on as" Species-Weights"," ImageNet-Weights" and" Control- Model".

e. As for the embedding block, which is put on top of the CNN, it is chosen this architecture: 3 Dense-Layers with 512, 256, and 256 output neurons, respectively, and ReLu - activation functions. Then choosen to use an l2-normalizing layer as the final output layer to enforce more evenly distributed distances to then later be able to better identify new individuals.

f. An automated online semi hard triplet mining with a default margin of 1 was done.

C. Evaluation

1. Procedure: Since the Kaggle Challenge offers a nice automatic way of testing models, this had to only come up with a validation procedure for the inter-model comparison. It was decided on this 2-stage validation procedure:

Firstly, since the model should learn to recognize individuals, it has seen before, It was decided to measure the accuracy of the images of the known individuals being correctly matched, as well as the k5-individual accuracy and the accuracy of classifying the right species. To achieve it has been calculated the embeddings of the train and validation datasets. Then, computed the pairwise distance matrix with a Euclidean metric of the two embeddings and sorted out the labels with the 5 closest values. With these labels, now calculated the previously mentioned accuracies. In addition, used UMAP to visualize a dimensional representation of the embeddings. Secondly, the model should also recognize when it has not seen an individual before. For that, used the individuals with only one picture and calculated their embeddings. Then calculated the pairwise distance matrix of those embeddings with the train embeddings and sorted out the distance to the nearest neighbour. Then compared this distance to the average distance of individuals, which the model is already familiar with. For that, created density plots of the distributions.

2. Analyzing General Training Behaviour: When looking at these results in Figure 7 this were initially surprised. Why does every model shows its best performance before training? Therefore some data analysis done:

Before training, all the weights of the embedding block are randomly initiated. Thus, a high Validation Accuracies arises probably from the very uneven distribution of individual counts. First, to look at the dataset used for the Siamese model training and validation, the dataset containing all the individuals with at least 2 images. The top 300 individuals of this data make out 5% of the individuals and 40% of the images. So, if chose random images from the data, the chances of getting the same individual are quite high. Even more interesting is to which species these most 300 common individuals belong to:

This can be clearly seen, that these top 300 individuals belong mainly to the most frequent classes. The validation accuracies obtained during training are shown in **Fig 7**.

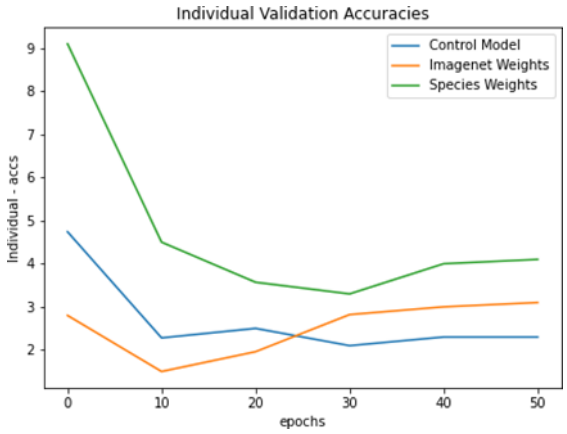

Figure 7: Individual Validation Accuracies

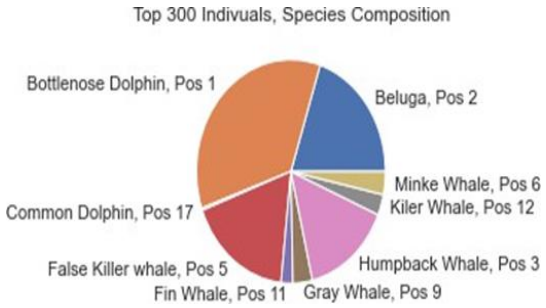

Figure 8: Species Composition of top 300 Individuals. Pos refers to the degree of frequency to the most frequent classes.

In addition, what is very interesting is that the species accuracy, the likelihood of the next neighbor in the embedding space being of the same space, is for all models around around 80% after epoch 30. If combined these two facts, it can refer that the most

frequent classes are quite densely packed together in the embedding space. Therefore, the images of the 300 most common individuals, which make out a huge part of the training data, are also most likely in those densely packed embedding spheres. It is assumed that, therefore, a lot of ”Hard-Triplet-Pairs” within these spheres of the most frequent species. And since training on ”Semi-Hard-Triplets,” the theory now is that the initial accuracy of the model decreases because it arises from correctly matching the most frequent individuals, which the model is then not trained to do. In addition, the 2d UMAP plots of the embeddings support this hypothesis.

1) Species Weights vs ImageNet Weights: All three validation accuracies (Individual, k5-Individual, Species) and also the 2d-UMAP-plots before and while training shows that the Species Weights Model performs significantly compared to the Imagenet-Model.

2) Species Weights vs ImageNet Weights: All three validation accuracies (Individual, k5-Individual, Species) and also the 2d-UMAP-plots before and while training shows that the Species Weights Model performs significantly compared to the Imagenet-Model.

3) Imagenet Weights vs Control Model: Although the In- ceptionV3 Model outperforms the Resnet50V2 model in terms Individual and k5 accuracy the effect is by far not as strong in the Softmax case. Especially if I take into account that it took the InceptionV3 model almost double as long as the Control Model to have a convergent training loss.

D. Kaggle Test Dataset Evaluation

Since the best performing model is, quite ironically, the Species-Model before training even one epoch with the triplet loss, chosen it for the test evaluation. To now test the model, it had to create a submission.csv according to this guideline. Most importantly is that it is not only have to identify individuals which has seen before, but also new individuals. For that, used the density plot of the average distance of new individuals to their next neighbors compared to known individuals. Looking at the distributions decided to expect there to be a new individual at a distance of 0.35. To generate the submission.csv now first had to compute the embeddings of all the data (the Kaggle training data) and the embeddings of the Kaggle test data. Then, computed the pairwise distance matrix and sorted out the 5 closest neighbors and their corresponding labels. For every test image I know, iterated through the list of the 5 closest neighbors. If the distance to the neighbor was bigger than 0.35, inserted ”newindividual” into the list at that position. Like this We achieved a score of 0.077.

V. Conclusion

A. Hypothesis 1

Is a ”lazy” approach of just using prebuild & pre-trained architectures and preconfigured loss functions sufficient enough to produce significant results in individual classification?

With a final Kaggle Evaluation Score of 0.077, the model did really not achieve sufficient result to be use full in anyway in marine nature conservation. Therefore, it is assumed that a similar approach is most likely also not sufficient enough for other domains of nature conservation.

B. Hypothesis 2

Is Softmax classification training success a useful indicator for triplet loss training success? To be specific: Does the best performing CNN architecture under Softmax also show the best performance under triplet loss?

With the InceptionV3 model with imagenet weights just barely outperforming the Resnet50V2 model with imagenet weights in regard to validation accuracy, it does not

have enough data to either support or deny this hypothesis.

C. Hypothesis 3

Can you reduce triplet loss training time with network weights pre-trained on Softmax classification?

Because of the strong performance of the InceptionV3 Model pre trained on classifying the whale and dolphin species compared to the other models, and have strong reason to follow this hypothesis.

Acknowledgment

The author would like to thank the Kaggle Happywhale challenge community for providing the dataset and resources used in this study.

Conflict of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

References

[1] Kaggle, "Happywhale - whale and dolphin identification," 2022, visited on 2022-04-01. https://www.kaggle.com/c/happy-whale-and-dolphin

[2] n.d., 2022, visited on 2022-04-01. https:// happywhale.com

[3] S. Malhotra, "Self-organizing interaction spaces: A framework for engineering pervasive applications in mobile and distributed environments," 2025. https://arxiv.org/abs/2502. 01137

[4] A. Awasthi, "Leveraging gans for active appearance models optimized model fitting," 2025. https://arxiv.org/abs/2501. 11218

[5] M. Cherukuri, "Comparing image segmentation algorithms," in 2024 IEEE 4th International Conference on Data Science and Computer Application (ICDSCA), 2024, pp. 266–269.

[6] L. Schmid, R. Horn, C. Lange, M. Pink, and K. Kobrock, "05 introduction to cnns,"http://www.studip.uni-osnabrueck.de, University of Osnabrueck, 2021, visited on 2022-04-01.

[7] P. Raghav, "Neural network with many convolutional layers," 2018, visited on 2022-04-01. https://miro.medium.com/max/1200/1*XbuW8WuRrAY5pC4t-9DZAQ.jpeg

[8] Meta Platforms, "Image classification," 2022, visited on 2022-04-01. https://paperswithcode.com/task/ image-classification#papers-list

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition. IEEE, 2009, pp. 248–255.

[10] L. Schmid, R. Horn, C. Lange, M. Pink, and K. Kobrock, "02 training nns," http://www.studip.uni-osnabrueck.de, University of Osnabrueck, 2021, visited on 2022-04-01.

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, jun 2015, visited on 2022-04-01.https:

//doi.org/10.1109%2Fcvpr.2015.7298682

[12] L. Weng, "Contrastive representation learning," lilianweng.github.io, 2021, visited on 2022-04-01. https://lilianweng.github.io/posts/2021-05-31-contrastive#triplet-loss

[13] C. Kha Vu. (2021) Deep metric learning: A (long) survey. Visited on 2022-04-01. https://hav4ik.github.io/articles/deep-metric-learning-survey

[14] K. Sohn, "Improved deep metric learning with multi-class n-pair loss

objective," in Advances in Neural Information Processing Systems,

D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016, visited on 2022-04-01. https://proceedings.neurips.cc/paper/2016/file/ 6b180037abbebea991d8b1232f8a8ca9-Paper.pdf

[15] O. Moindrot. (2018) Triplet loss and online triplet mining in tensorflow. Visited on 2022-04-01. https://omoindrot.github.io/ triplet-loss#triplet-mining

[16] A. D. Kapse, "Foreground extraction-opencv," 2020, visited on 2022-04-01. https://www.kaggle.com/code/akhileshdkapse/ foreground-extraction-opencv/notebook