Individual photo-identification of seahorses based on artificial intelligence

M2 IAAA - 2022

Mathias Aloui
Aix-Marseille Université
mathias.aloui@etu.univ-amu.fr

Julien Renoult
Centre d'Ecologie Fonctionnelle et Evolutive
julien.renoult@cefe.cnrs.fr

Abstract

The study of Evolutionary Significant Unit (ESU) is important in ecology. The long-snouted seahorse lineage of the Thau lagoon (Hérault, France) is an ESU and is studied by the association Peau-Bleue. In order to better understand this specie and better protect it, a comprehensive census is necessary. However, "manual" identification on photograph is long and hard for scientists. To help them, we are going to use deep learning methods to support their individual re-identification (re-ID). Animal re-ID using deep learning methods is new and is becoming more and more popular. Recent studies have shown that Convolutional Neural Network (CNN) trained with a triplet loss are able to accurately identify individuals from numerous species. In this study, we are going to prove that this method can also work on a difficult specie, because it's aquatic and is expert in camouflage, the long-snouted seahorse.

Contents

1 Introduction

The long-snouted seahorse lineage of the Thau lagoon (Hérault, France) differs genetically and morphologically from the marine population of the same species (*Hippocampus guttulatus*); it is endemic to a few Occitanian lagoons only (less than 150 square km). It is thus considered an Evolutionary Significant Unit (ESU), which deserves being studied and managed as such (Riquet et al., 2019). Involved in the scientific study of this population, the association Peau-Bleue and its partners have engaged in the challenging project of a comprehensive underwater census of Thau lagoon seahorses. The census necessitates to individually identify seahorses from underwater photographs. Individual identification is necessary to estimate various demographic parameters using models of sighting-resighting ("capture-recapture"; population size, dispersal, longevity), but also to better understand the life history (growth, reproductive cycles. . .), the behaviour (inter-individual relationships, pair fidelity, day vs. night behaviour. . .) and the ecology (habitat preferences over seasons) of seahorses. This information would ultimately allow setting an efficient conservation policy for this charismatic species.

The re-identification (re-ID) of animal individuals has been a long-standing challenge for research in ecology. Techniques such as tagging, scarring, banding, and DNA analyses of hair follicles or feces are accurate but expensive, intrusive to the animal and laborious for the field research team (Krebs, 1989; Ferreira et al., 2020). Alternatively, re-ID can be made by visual identification based on natural marks, for example on pictures from camera traps. However, it usually requires months of training and practical experience to be able to accurately re-identify individuals of a given species, and even so, visual re-ID remains subject to human error and bias (Foster and Harmsen, 2012; Meek et al., 2013). The technique is also time consuming and expensive because of the large quantity of data to analyse manually.

Models of computer vision can save time to researchers, opening the way to big data analyses in animal demography. The first computer vision models used to identify animal individuals were developed in the 1990s (Whitehead, 1990). Historically, species-specific algorithms were created to extract features from the pictures, and traditional machine learning models like support vector machine (Hearst et al., 1998) were applied to these features to classify individuals. Nowadays, neural networks and deep learning model like Convolutional Neural Network (CNN) (Krizhevsky et al., 2012) do not require hand-crafted features anymore: the model automatically builds and selects features that are most efficient for achieving the task of individual identification, thereby circumventing the limitations of arbitrarily creating and selecting potentially important features.

Conventional CNN-based re-ID methods are classification tasks with one class per individual and require a large number of pictures per class to train the recognition model. To re-identify new individuals, the model needs to be retrained with pictures of these new individuals. This approach is referred to as closed-set re-ID. In contrast, open-set re-ID allows re-identification of individuals not present in the training data set. The problem of open-set re-ID can be formulated as a one-shot learning task, in which a single image is used as reference for a class. For animal re-ID, this means that when an individual is first sighted, its image is used as reference. Then, when the individual is re-sighted, the model should successfully classify the image with the reference image. Such a task can be solved using a similarity comparison network. In the context of Re-ID, this type of network is trained to distinguish if two images represent the same individual or not.

The goal of my project is to develop an AI-based solution to help scientists in their seahorse census. The first challenge is to extract the relevant part of the pictures. Seahorses like to hide and, in most of the pictures, the seahorse represents only a small percentage of the pixels and the majority of the image represents only environment/background. Our solution is to use Yolo, a simple object detection network, to crop the pictures and have images centered on the head of the seahorses, where scientists assume cues for individual identification are mostly present. The second challenge is to be able to use our solution in open population. Indeed, the population of seahorses is different each year, with new individuals being regularly discovered (either immigrants or individuals previously undetected). An approach named "similarity comparison network" using a triplet loss has been shown to tackle this challenge with high accuracy on several species including: human, chimpanzee, humpback whale, fruit flies and Siberian tiger (Schneider et al., 2022). However, our case study is more challenging. On the one hand, seahorses

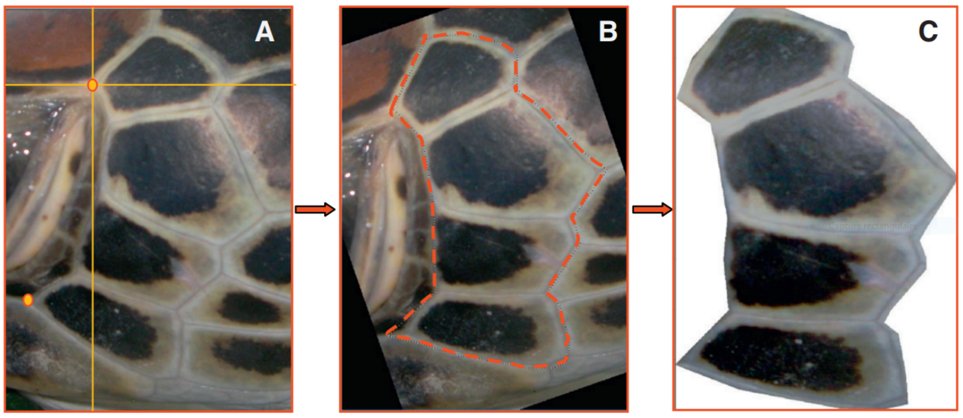


Figure 1: Pre-processing of turtle images before the artificial neural network

are known for their exceptional camouflage capacities. Seahorses can change color and are sometimes covered with algae which help them to hide in their habitat. On the other hand, lagoons are ever changing environments, for example because of algae bloom greening the water during the summer season, and sudden in-fillings by adjacent rivers during floods, which brings in sediment-rich, turbid water. These create harsh environmental conditions that may hamper the use of the dotted pattern present on the head, which is traditionally used by scientists to identify individuals. All these reasons make the long-snouted seahorse a good biological model for testing the performances of state-of-the-art methods in open-set re-ID.

Two issues remain : i) adapt the current models used on terrestrial animals to aquatic animals, and ii) train the models to identify individuals in harsh environmental conditions. For both of these challenges, we can simply use data augmentation. We can recreate the effects of the environment like change in color and luminosity or particles in suspension between the individual and the camera. We can also mimic the difficulty to identify the individuals on certain photos because they are hidden or cover with algae.

In this work, we have trained Yolov5 to crop the pictures with a focus on the head of the seahorses. We also have investigated which data augmentations could simulate the variety of color, luminosity and particles in suspension that can be encountered in the lagoon. Then, we searched for the best resolution for our image. The white dots on the seahorses disappear with low resolutions. We used the tensorflow implementation of EfficientNet to test different models with different input sizes. Finally, during the remaining time, we started to test variation on the architecture to improve the model performances. Our ultimate goal is to have a complete pipeline for seahorse re-ID in open population.

2 State of the art

For decades, animal re-ID was made with custom species-specific algorithms created to extract certain known features from the species of interest. Only recently, deep learning approaches have started to be used for animal re-ID. The first works using computer vision for re-ID were used for human re-ID. Many studies have demonstrated well-performing methods using conventional CNNs and large amount of training data. Studies have shown that human re-ID can be made with pictures from mobile devices (Lisanti et al., 2015), which show potential for animal re-ID with pictures from camera traps.

With the growing success a human re-ID, animal re-ID potential grew. One of the first studies published for animal re-ID with deep learning used an ensemble of neural networks to re-ID green turtle individuals from pre-processed pictures (see figure 1) in a closed population (Carter et al., 2014). This work has been considered a large success and is still used to monitor the green turtle population in the southern Great Barrier Reef.

Later works often considered great ape species for a variety of reasons like conservation related questions. Most of the great apes studied are in closed environment. It means that a conventional network with a fixed number of individuals to identify is usable. Studies have shown the improved accuracy of

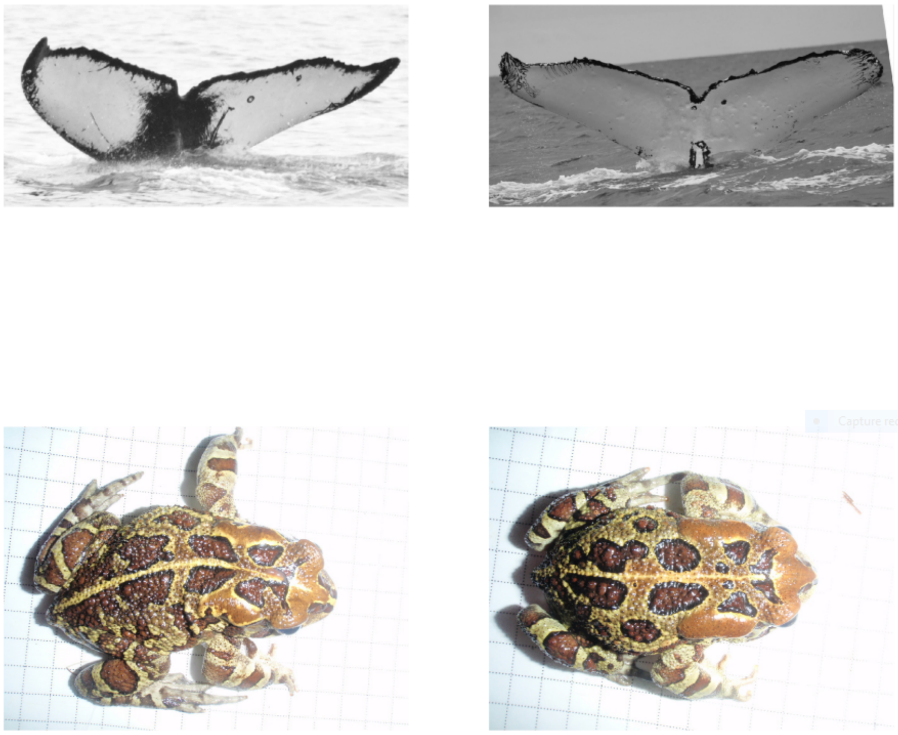


Figure 2: Examples of pictures of whales and toad. The pictures are not underwater.

deep learning models over previous methods on chimpanzee re-ID. One study (Freytag et al., 2016) considered two data sets, C-Zoo and C-Tai, and reported accuracy of 92.0% and 75.7% in comparison to the original accuracy of 84.0% and 68.8% with the Support Vector Machine method (Loos and Ernst, 2013). Another study used an object detection network to crop the images on the faces of gorillas and do re-ID upon those faces (Brust et al., 2017). This shows that object detection can be used to find certain part of an animal on an image. Lastly, researchers have successfully tracked and identified 23 chimpanzees on video using a CNN (Schofield et al., 2019).

While the accuracy increases and new complex tasks are solved in closed population, similarity comparison networks offer a promising solution for open population. Similarity comparison networks originate from "Siamese networks" first developed in 1994 to detect signature forgery (Bromley et al., 1993). In recent year, similarity comparison networks have become popular within ecology. One study has shown the potential of using contrastive loss for animal re-ID, considering three species: chimpanzees, lemurs, and golden monkeys (Deb et al., 2018). They defined three performance metrics for animal re-ID : verification, if two images are from the same individual, closed-set identification, if an individual is present in a given set, and open-set identification, if an individual is present in a given set or absent from the set). Accuracies of the verification, closed-set, and open-set were respectively for lemurs: 83.1%, 93.8%, 81.3%; golden monkeys: 78.7%, 90.4%, 66.1%; and chimpanzees: 59.9%, 75.8%, and 37.1%. These results demonstrate that open-set Re-ID is doable with terrestrial animals.

For aquatic animals, individuals of humpback whales and western leopard toads have been successfully re-ID with an object detector and a similarity comparison network working in tandem (Kabuga, 2019). If this work shows that aquatic animals can be re-ID using similarity comparison, the pictures were not taken underwater, and thus the special environmental conditions seen on underwater images were not investigated (see figure 2). With this respect, a study conducted on zebrafishes is more appropriate (Haurum et al., 2020); however, this study used images taken in a perfectly controlled (laboratory) environment, thus with clear water and without complex background, and was conducted on a close population. Moreover, the marks used to identify zebrafish individuals are big in comparison to the body of the fish. In our case, the white dots are harder to see and the patterns harder to notice.

If re-ID was almost exclusively used on species with descript phenotypic patterns, some studies tried to use similarity comparison network to identify species with non-descript patterns. One example is the re-ID of brown bears using facial features (Clapham et al., 2020). The facial features of bears are not distinct markings that can be used by humans to identify an individual. The study shows that, using in tandem face localization and similarity comparison network, brown bear can be successfully re-identified. This success gives great hopes when generalizing to species whether or not they have descript or non-descript phenotypic patterns.

Similarity comparison network initially used the contrastive loss during training. Recent studies have shown that the triplet loss outperforms the contrastive loss. For human re-ID, two studies from 2014 and 2015 have demonstrated the higher performances of triplet loss for human face recognition (Schroff et al., 2015; Hoffer and Ailon, 2014). This increase in performance in human re-ID suggests a similar increase in performance in animal re-ID. This assumption is validated by three studies showing high performances for dolphin and seal re-ID (Bouma et al., 2019; Nepovinnykh et al., 2020; Chelak et al., 2021). Finally, a very recent study showed that the triplet loss outperforms the contrastive loss in animal re-ID for 5 species (Schneider et al., 2022). In addition, in the fives species, there is an aquatic species (humpback whales) and a species with non-descript phenotypic pattern (fruit flies).

We want to test the triplet loss on the long-snouted seahorse, because it's an aquatic species with underwater images taken in its natural environment, this species is expert in camouflage, being often hidden or covered with algae, and it has descript phenotypic patterns but they are difficult to perceive (small and potentially confounded with sand particles in suspension in the water), and are asymmetrical.

3 Method et data

3.1 Data from Peau-Bleue

In 2005, the association Peau-Bleue made an alarming observation. While seahorses were in the process of being depleted worldwide, we knew virtually nothing about the seahorses on our coast. That is why the "programme EnQuête d'Hippocampes" was launched in response to the urgent need to study European seahorses, to estimate the fragility of their populations, the vulnerability of their habitats but also to raise public awareness of their preservation. Since then, it has been extended to other species of the Syngnathidae family, notably species of genus *Syngnathus*, some of which are probably even more vulnerable than seahorses.

One project, called "Hippo-THAU", aims to study the seahorse population in the Thau lagoon. Since 2020, this project includes a comprehensive census. Members of the association have started collecting pictures of seahorses in the studied area and scientists have "manually" identified individuals by visual comparison of the photos. Older photos were also used, even though they were not taken for the purpose of individual identification. Counting older photos from 2018 and 2019 and recent photos made specifically from re-ID from 2020 and 2021, the seahorse image database includes thousands of photos of hundreds of individuals. After I selected only the photos where the head of the seahorse is visible, we were left with 14,638 photos representing 342 individuals, all identified with high level of confidence by the same person (Sylvie Louisy).

The next campaigns will bring a large number of new photos and the association wishes to shift from a visual, time-consuming, individual identification to a fully automated, or semi-automated process based on artificial intelligence. Initially, these new photos should have been used for testing the algorithm but, due to delays in acquiring these photos, we split the photos we already have into two datasets: one for training and validating and one for testing the algorithm. The first dataset (train-validation dataset) includes 13,204 photos of 176 individuals, with at least 20 pictures per individual. The second dataset (test dataset) includes 1,434 photos of 166 individuals. Each individual is represented by 1 to 19 photos. This split between train-validation and test datasets was made on the individual level (no individual is present in both datasets), for two reasons. First, we wanted a split around 90/10 for train-validation/test, as typically done in machine learning. Second, the triplet loss required at least 2 pictures for each individual (anchor and positive). But the implementation of the triplet loss we are using searches for the hardest triplet. This means that during training, the more photos we have per individual, the more likely

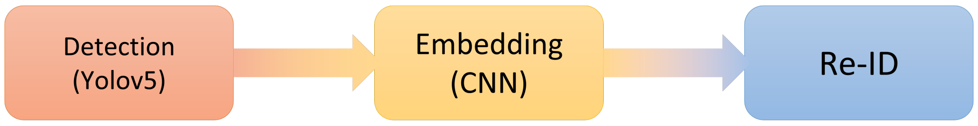


Figure 3: Diagram of the pipeline

it is that the model will learn meaningful features to differentiate individuals.

3.2 Pipeline

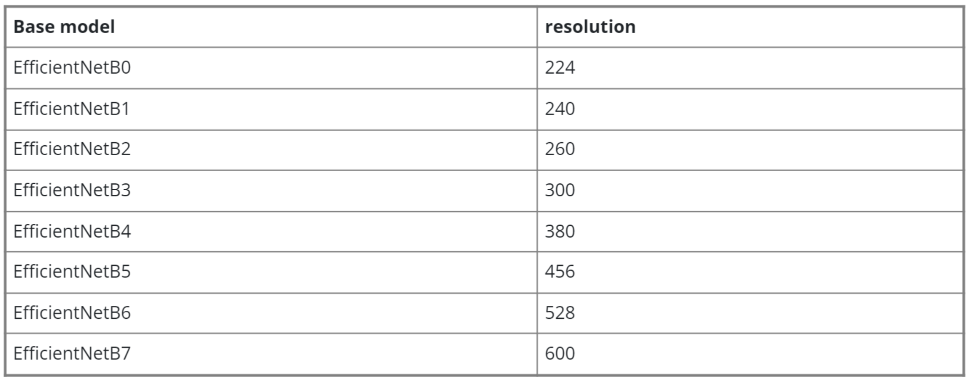
To facilitate the work of the scientists of the Peau-bleue association, we need to create a complete pipeline. The first task is to crop the photos to ensure that most of each picture display the head and the upper part of the body, and a minimum of surrounding environment. Then, the cropped image is sent to the trained model that produces an embedding for this image. This embedding is then compared to the ones in the reference database. The output of our pipeline is a list of candidate individuals and a suggestion of whether or not their identity matches the queried individual. This pipeline corresponds to open-set re-ID because the reference database can include pictures of individuals that were not present during model training. In our case, the test set contains exclusively pictures of individuals *not* present during training; the results presented here are thus conservative regarding the future use of our pipeline by the scientists.

3.2.1 Yolov5

In order to crop the photos around the head and the upper body part of the seahorses, I have trained the Pytorch implementation of Yolov5. To that end, during my internship I have annotated all 14,638 photos with a square centered on the head of the seahorse. Since the task of cropping is independent of the individual identity, the photos of the train-validation dataset were shuffle and split between crop-train, crop-validation and crop-test datasets with respectively 11,724, 1,427 and 1,487 photos. No customized data augmentation was added at this point because Yolov5 has its own augmentation built-in. We used the largest architecture possible with a reasonable training time with our computing power : the large architecture Yolov5l. I used the GitHub implementation of Yolov5 which include tutorials and scripts to train and test the CNN.

3.2.2 Data augmentation

Once the photos have been cropped, for the training step of the re-ID, we need to recreate all the diversity of non-informative variation between same-individual pictures that are caused by underwater environmental conditions. We used the package imgaug, which has been already used in animal re-ID (Miele et al., 2021) but only to manipulate the image without visual modifications. I used it to change the size of the photos and add padding when necessary. I also used the rotation function. Indeed seahorses can sometimes be upside-down or leaning because of the current. I used a function to change the luminosity of the photos, which depends on the flash used and the camera settings, the time of the day and the depth. I used a function to change the coloration of the photos to recreate the changing color of the water. I used two functions to create particles and blur on the photos to mimic turbid water with sand and other small particles in suspension between the seahorse and the camera. With the help of the scientists from the Peau-Bleue association, I identified the possible range of values for each augmentation. During preliminary experiments, too much data augmentation, i.e. a probability to close to 1, induced under-fitting. In order to avoid this issue, each augmentation is applied to an image of a batch with a probability of 50%, and when it is applied, the value of the augmentation (e.g., the rotation angle) is picked randomly from the selected range. There are 5 types of data augmentation : Rotation, the image is rotated by an angle between -180 and 180, Blur, the image is blurred by a kernel of 2 or 3 pixels, Salt, random white pixels on the image, Temperature, change the color of the image (cold = blue, hot = red), Luminosity, increase or decrease of the luminosity in the picture.



Base model	resolution
EfficientNetB0	224
EfficientNetB1	240
EfficientNetB2	260
EfficientNetB3	300
EfficientNetB4	380
EfficientNetB5	456
EfficientNetB6	528
EfficientNetB7	600

Figure 4: Resolutions for each EfficientNet architecture (from Keras docs)

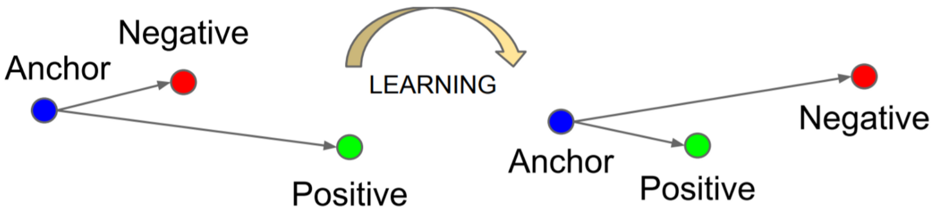


Figure 5: Diagram illustrating one training step with the triplet loss, from "The intuition of Triplet Loss" by Susmith Reddy on the website Medium. After the training step, the distance between the anchor and the positive input decreases and the distance between the anchor and the negative input increases.

3.2.3 Triplet loss

Once the data augmentations set, we need to find the optimal resolution for the photos. The original photos were taken with really high resolutions. When a seahorse is "manually" re-identified, the scientist search for patterns in the white dots on the head of the individual. When we lower the resolution, some of these white dots disappear. The resolution must be high enough to discern the dots, but not too high because higher resolution means bigger model, which requires more time and data to be trained. We tested different TensorFlow implementation of EfficientNet, a CNN with 8 architectures with input resolution varying between 224x224 and 600x600 (see figure 4).

To train these architectures, we use the triplet loss (Schneider et al., 2022) :

$$L(A, P, N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0) \tag{1}$$

Where A is an anchor input, P is a positive input of the same class as A, N is a negative input of a different class from A, α is a margin between positive and negative pairs, and f is an embedding. This loss function reduces the distance between the anchor embeddings and the positive embeddings and increases the distance between the anchor embeddings and the negative embeddings (see figure 5). In the end, the network can generate embeddings for any new image, which can be compared with the embedding of known individuals : a small distance between two embeddings means that the two images are likely from the same individual, a high distance between two embeddings means the two images are likely from two different individuals.

We used one variation of the triplet loss implemented in TensorFlow : the TripletSemiHardLoss. This implementation makes the positive distance (between a pair of embeddings with the same label) smaller than the smallest negative distance (between a pair of embeddings with different labels). This smallest negative distance is among distances greater than the positive distance plus a margin constant in the considered mini-batch. If this negative distance doesn't exist, the largest one is used instead (Schroff et al., 2015). The TensorFlow implementation can use 3 distance metrics : L2, squared-L2 (Euclidean) and angular.

3.3 Re-ID

Once a model is trained with the triplet loss, we use it to create the embeddings for every photos in the database. An embedding describes an image as it is encoded by the neural network in the layer before the terminal triplet loss calculus. It corresponds to a low-dimensional (a few hundreds or thousands) vector.The embedding of any given picture is then compared with those in the database. To do so, we use the Scikit-Learn implementation of the Nearest Neighbors with either L2, Euclidean or cosine metrics. This Nearest Neighbors model is used to get the top k closest embeddings/photos (with the given metric) of any given query (embedding from a new photo). We calculate the Accuracy top-k for each query as :

$$Acc_k = \begin{cases} 1 & \text{if top-k embeddings contain the query identity} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We then define 2 metrics : the Cumulative Matching Characteristics top-k (CMC@k) and the mean Average Precision at k (mAP@k). CMC@k is defined as the mean of the Accuracy top-k. This metric only shows if the correct individual is in the top-k without considering the order. Our mAP@k is a variant of the one used for the Kaggle challenge "Humpback Whale Identification" (https://www.kaggle.com/competitions/humpback-whale-identification/overview/evaluation) define as :

$$mAP@k = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} P(j) * rel(j) \tag{3}$$

with N the number of queries, $P(j)$ the precision at cutoff j (meaning $\frac{TruePositive}{TruePositive+FalsePositive}$ for the top-j) and $rel(j)$ an indicator function equaling one if the embedding at rank j is the same individual and it's the first match. This metric only takes into account the first match (like the CMC@k) but further considers position of the match within the top-k. In our case, we don't need more information because the top-k photos will be given to the scientists to check the results and, if they agree with the match, they will add the photo to the database.

4 Experiences and results

4.1 Detection

Using the recommended hyper-parameters from GitHub tutorials, we trained a highly accurate seahorse detector with crop-train dataset for training, crop-validation for validation and crop-test for testing. Training was set to 600 epochs, but stopped at epoch 390 ("early-stop" procedure), after 100 epoch without improvement. The best performance in validation were reached at epoch 290 (see Table 1). The corresponding confusion matrix (see Figure 6) shows that 99% of the seahorses are detected (recall), 1% of the seahorses are not detected (1-recall) and, since there is only the class Hippocampus, 100% of the wrong Hippocampus detection are background. Checking some batches from the validation, we can corroborate the conclusions made from the confusion matrix : the vast majority of individuals are detected with high confidence, a few are missed falsely detected. But we can also add that the false defections seem to have lower confidence score (example on Appendix D). This is confirmed with the testing (see Table 2) : increasing the confidence threshold increases the precision and decreases the recall. Given the high performance with 0.5 confidence threshold and 0.5 Intersection-Over-Union threshold, we have not search for better parameters.

Precision	Recall	mAP@.5	mAP@.95
0.995	0.99	0.995	0.792

Table 1: Validation performances of Yolov5 after first training

Confidence threshold	IOU threshold	Precision	Recall	mAP@.5	mAP@.95
0.5	0.2	0.992	0.988	0.993	0.796
0.8	0.2	0.995	0.976	0.987	0.794
0.5	0.5	0.991	0.991	0.994	0.797

Table 2: Testing performances of Yolov5 after first training

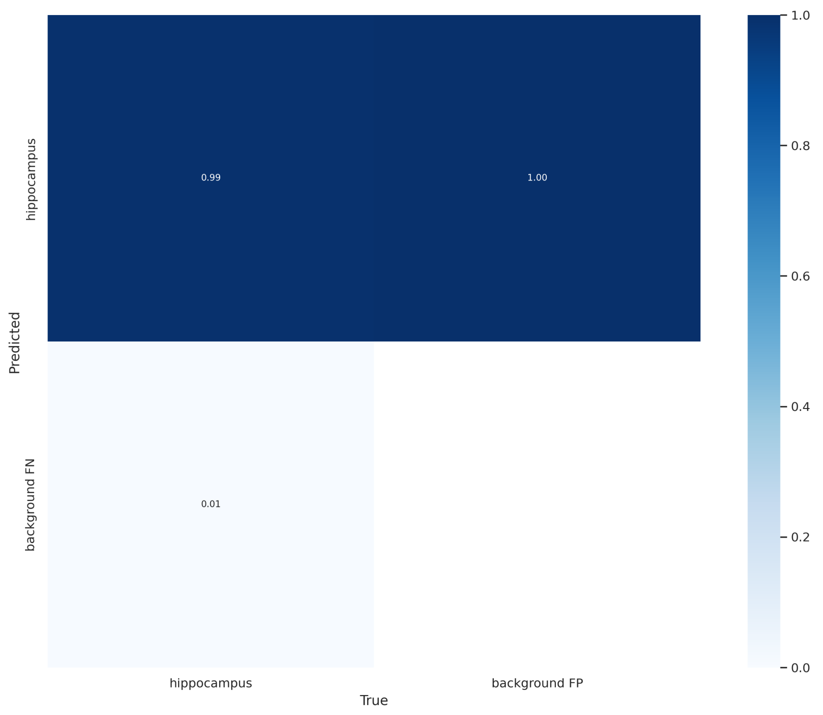


Figure 6: Confusion Matrix of Yolov5 validation

4.2 Re-Id

The first tests made on Google Colab with only 5% of the photos showed that increasing the size of the embedding decreased the performances. Furthermore, the architectures from EfficientNet B4 and above were extremely long to train, with very poor performances. Thus, when searching for the best EfficientNet architecture, we tested only architectures from B0 to B3 and only added to the base model a dense layer of size 2,048, a dropout layer with a varying value and a final dense layer (the embedding) of size 128. The results (see Appendix E) showed that architecture B3 with a low dropout and squared-L2 metric for the triplet loss was the best combination. However, all the combinations had the same problem : a big gap between the training and validation loss (see Figure 7). This situation is a classic over-fitting example where the model increases its performances on the training data without increasing its performances on the validation data. One simple solution is to use dropout layers (Srivastava et al., 2014). But in our case, we already have a dropout layer and increasing it's value decreased the performances. Another solution is to decrease the batch size (Kandel and Castelli, 2020). However, upon investigation, I found out that the problem doesn't rely on the batch size but on the batch constitution. To create a batch when training with a triplet loss, we can choose how many individuals and how many images per individual we want. Initially, we used 3 classes with 10 images per class for each batch. With lots of images for only a few classes, the model learned the differences for each pair of classes, which caused over-fitting. To limit this effect, we tested 10 classes with 3 images per class, and then 15 classes with 2 images per class. With many classes per batch and only a few images per class, the model learns differences between multiple classes and is forced to find them between the few images per class. We can see the improvement in Table 3 with the performances for the model with EfficientNetB3, dropout

0.1, squared-L2 for the triplet metric and Euclidean for evaluation metric. Once the over-fitting problem resolved, we can remove the dropout layer which seems to have a negative effect on the performances. In addition, we test other sizes for the two dense layers (see the results in Table 4).

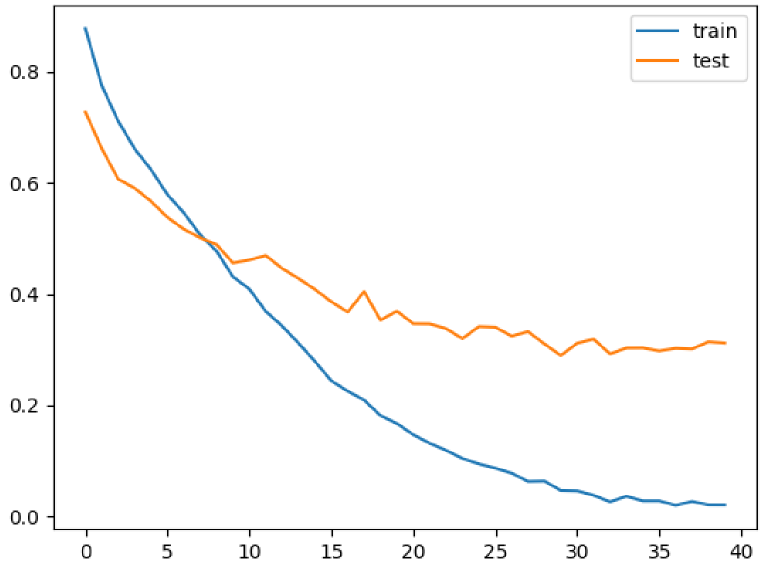


Figure 7: Loss for the training (train) and validation (test) with the combination : EfficientNetB3 / 0.1 / squared-L2

nb of classes	nb of images per class	CMC@1	CMC@5	mAP@1	mAP@5
3	10	0.58926081	0.77405858	0.58926081	0.65983264
10	3	0.65271967	0.81868898	0.65271967	0.71672478
15	2	0.70083682	0.85704324	0.70083682	0.76155277

Table 3: Performance with different combination of number of classes and number of images per class

size first dense	size second dense	CMC@1	CMC@5	mAP@1	mAP@5
2048	64	0.75592748	0.89539749	0.75592748	0.8100186
1024	64	0.73361227	0.87866109	0.73361227	0.78863319
512	64	0.68967922	0.84239888	0.68967922	0.74868666
2048	128	0.76011158	0.89748954	0.76011158	0.81283124
1024	128	0.72663877	0.86052999	0.72663877	0.77842864
512	128	0.67921897	0.84518828	0.67921897	0.74307299

Table 4: Performance with different sizes for the dense layers

5 Conclusions and perspectives

Due to several problems with the server used for GPU computation, there is still a lot of architectures to train and evaluate. For example, we could change the number of dense layers and their size, further improve which images are chosen for each batch and search for better parameters for the data augmentations. We should then find a way to calculate a threshold to be able to tell scientists when an individual is more likely to be new or already in the database. We should then find a metric to give a probability for each individual to be the one in the photo of the query.

Successfully identifying seahorse from underwater photographs is essential for a comprehensive census of the population. Helping the scientists with this identification task will in the end help the conservation effort of this evolutionary significant unit. We have proven that the triplet loss can be used even with difficult species like seahorse and on open-set. The combination of data augmentations, even with untuned parameters, and carefully selected batch has given good results. This work lays the ground for new studies, for instance how to take into account asymmetry in animal re-ID. Indeed long-snouted seahorses have lateral asymmetry in their white dot patterns, which creates different shapes and patterns on each side of the individual (see an example with figure 11 and figure 12 in appendix A). For the asymmetry, we thought of three solutions : use two classes per individual for each side, use multitask learning to make the network differentiate left and right side and use a Neural Structured Network (Gopalan et al., 2021) allowing to set the relationship between the two sides of each individual as a prior.

6 Acknowledgements

I want to thanks Natural Solutions to have funded this internship and the employees who gave me assistance when needed. I want to thanks Oliver Rovellotti (CEO) and Aurélie Jambon (Project Owner) for their support. I want to thanks the Centre d'Ecologie Fonctionnelle et Evolutive for the scientific support. I want to thanks Julien Renoult (CEFE-CNRS) for the supervision in the deep learning work, Patrick and Sylvie Louisy for their expertise with seahorses. This study benefited from the Montpellier Bioinformatics Biodiversity platform supported by the LabEx CeMEB, an ANR "Investissements d'avenir" program (ANR-10-LABX-04-01)."

References

[Bouma et al.2019] Soren Bouma, Matthew D. M. Pawley, Krista Hupman, and Andrew Gilman. 2019. Individual common dolphin identification via metric embedding learning.

[Bromley et al.1993] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Brust et al.2017] Clemens-Alexander Brust, Tilo Burghardt, Milou Groenenberg, Christoph Kading, Hjalmar Kuhl, Marie Manguette, and Joachim Denzler. 2017. Towards automated visual monitoring of individual gorillas in the wild. In *2017 IEEE International Conference on Computer Vision Workshop*, pages 2820–2830, 10.

[Carter et al.2014] Steven Carter, Ian Bell, Jessica Miller, and Peter Gash. 2014. Automated marine turtle photograph identification using artificial neural networks, with application to green turtles. *Journal of Experimental Marine Biology and Ecology*, 452:105–110, 03.

[Chelak et al.2021] Ilia Chelak, Ekaterina Nepovinnykh, Tuomas Eerola, Heikki Kälviäinen, and Igor Belykh. 2021. Eden: Deep feature distribution pooling for saimaa ringed seals pattern matching.

[Clapham et al.2020] Melanie Clapham, Ed Miller, Mary Nguyen, and Chris Darimont. 2020. Automated facial recognition for wildlife that lack unique markings: A deep learning approach for brown bears. *Ecology and Evolution*, 10:12883–12892, 12.

[Deb et al.2018] Debayan Deb, Susan Wiper, Sixue Gong, Yichun Shi, Cori Tymoszek, and Alison Fletcher. 2018. Face recognition: Primates in the wild. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems*, pages 1–10, 10.

[Ferreira et al.2020] André C. Ferreira, Liliana R. Silva, Francesco Renna, Hanja B. Brandl, Julien P. Renoult, Damien R. Farine, Rita Covas, and Claire Doutrelant. 2020. Deep learning-based methods for individual recognition in small birds. *Methods in Ecology and Evolution*, 11(9):1072–1085.

[Foster and Harmsen2012] Rebecca Foster and Bart Harmsen. 2012. A critique of density estimation from camera-trap data. *Journal of Wildlife Management*, 76:224–236, 02.

[Freytag et al.2016] Alexander Freytag, Erik Rodner, Marcel Simon, Alexander Loos, Hjalmar Kühl, and Joachim Denzler. 2016. Chimpanzee faces in the wild: Log-euclidean cnns for predicting identities and attributes of primates. In *German Conference on Pattern Recognition*, volume 9796, pages 51–63, 09.

[Gopalan et al.2021] Arjun Gopalan, Da-Cheng Juan, Cesar Ilharco Magalhaes, Chun-Sung Ferng, Allan Heydon, Chun-Ta Lu, Philip Pham, George Yu, Yicheng Fan, and Yueqi Wang. 2021. Neural structured learning: Training neural networks with structured signals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 1150–1153, New York, NY, USA. Association for Computing Machinery.

[Haurum et al.2020] Joakim Haurum, Anastasija Karpova, Malte Pedersen, Stefan Bengtson, and Thomas Moeslund. 2020. Re-identification of zebrafish using metric learning. In *2020 IEEE Winter Applications of Computer Vision Workshops*, pages 1–11, 03.

[Hearst et al.1998] Marti Hearst, S.T. Dumais, E. Osman, John Platt, and B. Scholkopf. 1998. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13:18 – 28, 08.

[Hoffer and Ailon2014] Elad Hoffer and Nir Ailon. 2014. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, 12.

[Kabuga2019] Emmanuel Kabuga. 2019. *Using Neural Networks to identify Individual Animals from Photographs*. Ph.D. thesis, University of Cape Town. Statistical Sciences.

[Kandel and Castelli2020] Ibrahem Kandel and Mauro Castelli. 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4):312–315.

[Krebs1989] Charles J Krebs. 1989. *Ecological methodology*. Technical report. NY: Harper and Row Publishers Inc.

[Krizhevsky et al.2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Lisanti et al.2015] Giuseppe Lisanti, Iacopo Masi, Andrew D. Bagdanov, and Alberto Del Bimbo. 2015. Person re-identification by iterative re-weighted sparse ranking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1629–1642.

[Loos and Ernst2013] Alexander Loos and Andreas Ernst. 2013. An automated chimpanzee identification system using face detection and recognition. *EURASIP Journal on Image and Video Processing*, 2013:49, 07.

[Meek et al.2013] Paul Meek, Karl Vernes, and Gregory Falzon. 2013. On the reliability of expert identification of small-medium sized mammals from camera trap photos. *Wildlife Biology in Practice*, 9:1–19, 08.

[Miele et al.2021] Vincent Miele, Gaspard Dussert, Bruno Spataro, Simon Chamaillé-Jammes, Dominique Allainé, and Christophe Bonenfant. 2021. Revisiting animal photo-identification using deep metric learning and network analysis. *Methods in Ecology and Evolution*, 12(5):863–873.

[Nepovinnykh et al.2020] Ekaterina Nepovinnykh, Tuomas Eerola, and Heikki Kalviainen. 2020. Siamese network based pelage pattern matching for ringed seal re-identification. In *2020 IEEE Winter Applications of Computer Vision Workshops*, pages 25–34, 03.

[Riquet et al.2019] Florentine Riquet, Cathy Lieutard-Haag, Giulia Serluca, Lucy Woodall, Julien Claude, Patrick Louisy, and Nicolas Bierne. 2019. Effective population size and heterozygosity-fitness correlations in a population of the mediterranean lagoon ecotype of long-snouted seahorse hippocampus guttulatus. *Conservation Genetics*, 20:1281–1288, 12.

[Schneider et al.2022] Stefan Schneider, Graham Taylor, and Stefan Kremer. 2022. Similarity learning networks for animal individual re-identification: an ecological perspective. *Mammalian Biology*, 04.

[Schofield et al.2019] Daniel Schofield, Arsha Nagrani, Andrew Zisserman, Misato Hayashi, Tetsuro Matsuzawa, Dora Biro, and Susana Carvalho. 2019. Chimpanzee face recognition from videos in the wild using deep learning. *Science Advances*, 5, 09.

[Schroff et al.2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[Whitehead1990] Hal Whitehead. 1990. Computer assisted individual identification of sperm whale flukes. *Reports of the International Whaling Commission*, 12:71–77.

Appendix A : CEFE brochure

MUSE

MONTPELLIER UNIVERSITY OF EXCELLENCE

THÉMATIQUE

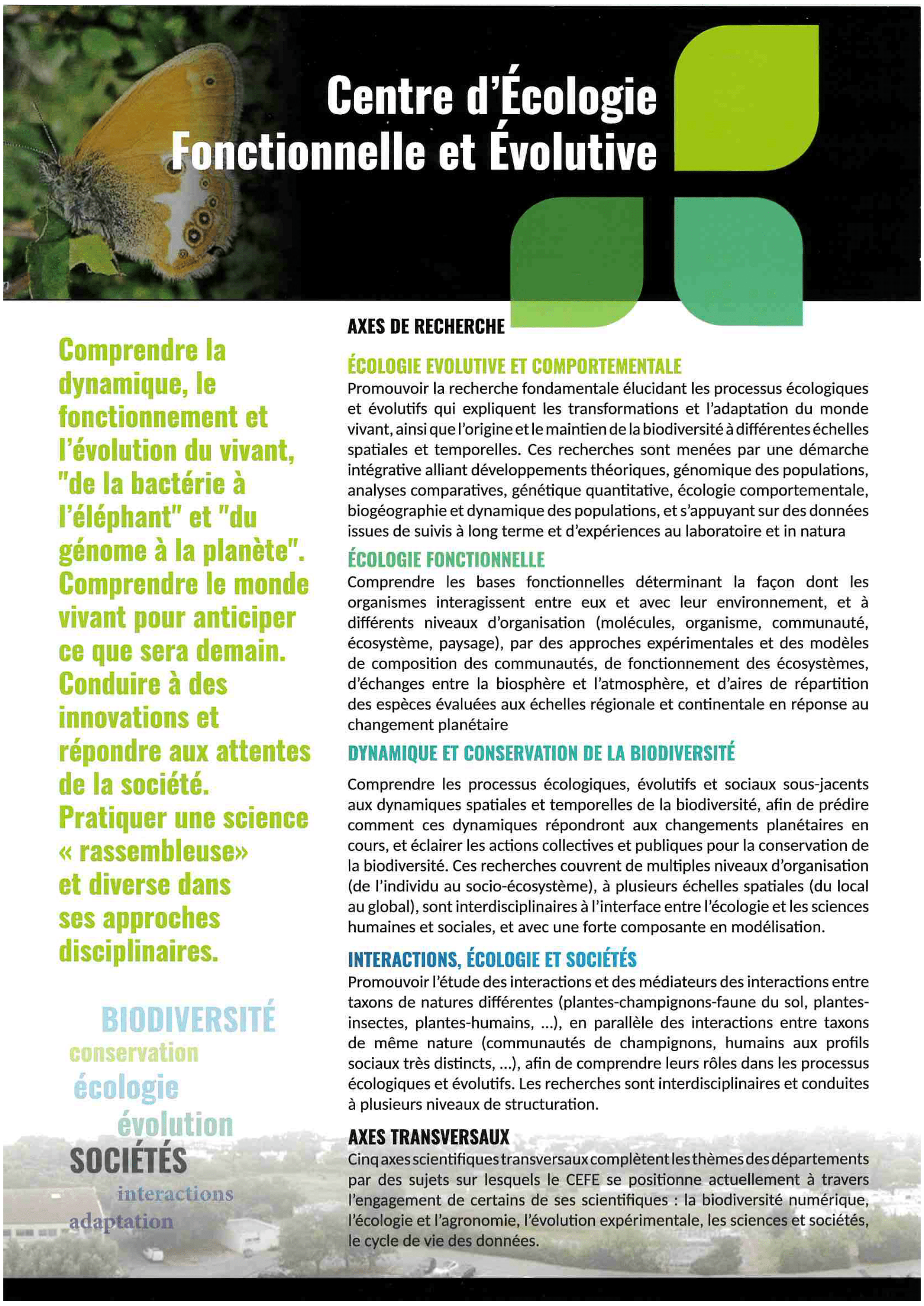
Environnement & Écologie

FICHE UNITÉ DE RECHERCHE

UMR CEFE

Centre d'Écologie Fonctionnelle et Évolutive

Figure 8: CEFE brochure : page 1



Centre d'Écologie Fonctionnelle et Évolutive

Comprendre la dynamique, le fonctionnement et l'évolution du vivant, "de la bactérie à l'éléphant" et "du génome à la planète". Comprendre le monde vivant pour anticiper ce que sera demain. Conduire à des innovations et répondre aux attentes de la société. Pratiquer une science « rassembleuse» et diverse dans ses approches disciplinaires.

AXES DE RECHERCHE

ÉCOLOGIE EVOLUTIVE ET COMPORTEMENTALE

Promouvoir la recherche fondamentale élucidant les processus écologiques et évolutifs qui expliquent les transformations et l'adaptation du monde vivant, ainsi que l'origine et le maintien de la biodiversité à différentes échelles spatiales et temporelles. Ces recherches sont menées par une démarche intégrative alliant développements théoriques, génomique des populations, analyses comparatives, génétique quantitative, écologie comportementale, biogéographie et dynamique des populations, et s'appuyant sur des données issues de suivis à long terme et d'expériences au laboratoire et in natura

ÉCOLOGIE FONCTIONNELLE

Comprendre les bases fonctionnelles déterminant la façon dont les organismes interagissent entre eux et avec leur environnement, et à différents niveaux d'organisation (molécules, organisme, communauté, écosystème, paysage), par des approches expérimentales et des modèles de composition des communautés, de fonctionnement des écosystèmes, d'échanges entre la biosphère et l'atmosphère, et d'aires de répartition des espèces évaluées aux échelles régionale et continentale en réponse au changement planétaire

DYNAMIQUE ET CONSERVATION DE LA BIODIVERSITÉ

Comprendre les processus écologiques, évolutifs et sociaux sous-jacents aux dynamiques spatiales et temporelles de la biodiversité, afin de prédire comment ces dynamiques répondront aux changements planétaires en cours, et éclairer les actions collectives et publiques pour la conservation de la biodiversité. Ces recherches couvrent de multiples niveaux d'organisation (de l'individu au socio-écosystème), à plusieurs échelles spatiales (du local au global), sont interdisciplinaires à l'interface entre l'écologie et les sciences humaines et sociales, et avec une forte composante en modélisation.

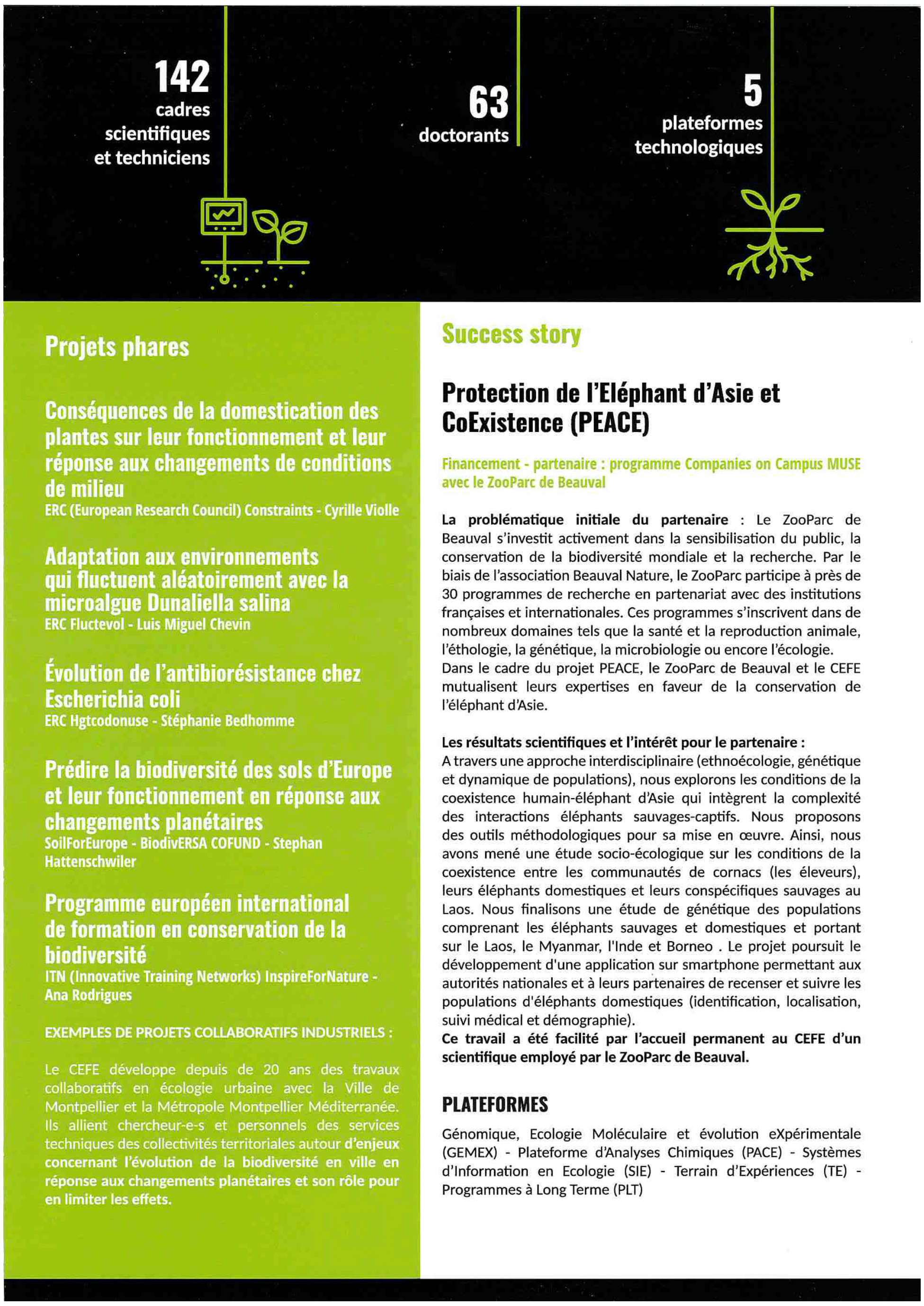
INTERACTIONS, ÉCOLOGIE ET SOCIÉTÉS

Promouvoir l'étude des interactions et des médiateurs des interactions entre taxons de natures différentes (plantes-champignons-faune du sol, plantes-insectes, plantes-humains, ...), en parallèle des interactions entre taxons de même nature (communautés de champignons, humains aux profils sociaux très distincts, ...), afin de comprendre leurs rôles dans les processus écologiques et évolutifs. Les recherches sont interdisciplinaires et conduites à plusieurs niveaux de structuration.

AXES TRANSVERSAUX

Cinq axes scientifiques transversaux complètent les thèmes des départements par des sujets sur lesquels le CEFE se positionne actuellement à travers l'engagement de certains de ses scientifiques : la biodiversité numérique, l'écologie et l'agronomie, l'évolution expérimentale, les sciences et sociétés, le cycle de vie des données.

Figure 9: CEFE brochure : page 2



142 cadres scientifiques et techniciens

63 doctorants

5 plateformes technologiques

Projets phares

Conséquences de la domestication des plantes sur leur fonctionnement et leur réponse aux changements de conditions de milieu
ERC (European Research Council) Constraints - Cyrille Violle

Adaptation aux environnements qui fluctuent aléatoirement avec la microalgue Dunaliella salina
ERC Fluctevol - Luis Miguel Chevin

Évolution de l'antibiorésistance chez Escherichia coli
ERC Hgtcodonuse - Stéphanie Bedhomme

Prédire la biodiversité des sols d'Europe et leur fonctionnement en réponse aux changements planétaires
SoilForEurope - BiodivERSA COFUND - Stephan Hattenschwiler

Programme européen international de formation en conservation de la biodiversité
ITN (Innovative Training Networks) InspireForNature - Ana Rodrigues

EXEMPLES DE PROJETS COLLABORATIFS INDUSTRIELS :

Le CEFE développe depuis de 20 ans des travaux collaboratifs en écologie urbaine avec la Ville de Montpellier et la Métropole Montpellier Méditerranée. Ils allient chercheur-e-s et personnels des services techniques des collectivités territoriales autour **d'enjeux concernant l'évolution de la biodiversité en ville en réponse aux changements planétaires et son rôle pour en limiter les effets.**

Success story

Protection de l'Eléphant d'Asie et CoExistence (PEACE)

Financement - partenaire : programme Companies on Campus MUSE avec le ZooParc de Beauval

La problématique initiale du partenaire : Le ZooParc de Beauval s'investit activement dans la sensibilisation du public, la conservation de la biodiversité mondiale et la recherche. Par le biais de l'association Beauval Nature, le ZooParc participe à près de 30 programmes de recherche en partenariat avec des institutions françaises et internationales. Ces programmes s'inscrivent dans de nombreux domaines tels que la santé et la reproduction animale, l'éthologie, la génétique, la microbiologie ou encore l'écologie. Dans le cadre du projet PEACE, le ZooParc de Beauval et le CEFE mutualisent leurs expertises en faveur de la conservation de l'éléphant d'Asie.

Les résultats scientifiques et l'intérêt pour le partenaire :
A travers une approche interdisciplinaire (ethnoécologie, génétique et dynamique de populations), nous explorons les conditions de la coexistence humain-éléphant d'Asie qui intègrent la complexité des interactions éléphants sauvages-captifs. Nous proposons des outils méthodologiques pour sa mise en œuvre. Ainsi, nous avons mené une étude socio-écologique sur les conditions de la coexistence entre les communautés de cornacs (les éleveurs), leurs éléphants domestiques et leurs conspécifiques sauvages au Laos. Nous finalisons une étude de génétique des populations comprenant les éléphants sauvages et domestiques et portant sur le Laos, le Myanmar, l'Inde et Borneo . Le projet poursuit le développement d'une application sur smartphone permettant aux autorités nationales et à leurs partenaires de recenser et suivre les populations d'éléphants domestiques (identification, localisation, suivi médical et démographie).
Ce travail a été facilité par l'accueil permanent au CEFE d'un scientifique employé par le ZooParc de Beauval.

PLATEFORMES

Génomique, Ecologie Moléculaire et évolution eXpérimentale (GEMEX) - Plateforme d'Analyses Chimiques (PACE) - Systèmes d'Information en Ecologie (SIE) - Terrain d'Expériences (TE) - Programmes à Long Terme (PLT)

Figure 10: CEFE brochure : page 3

Appendix B : Individual F028 - example for asymmetry

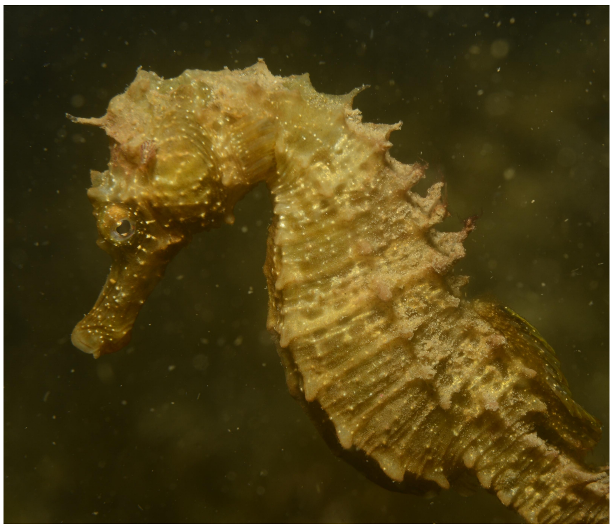


Figure 11: Picture of the left side of the individual F028

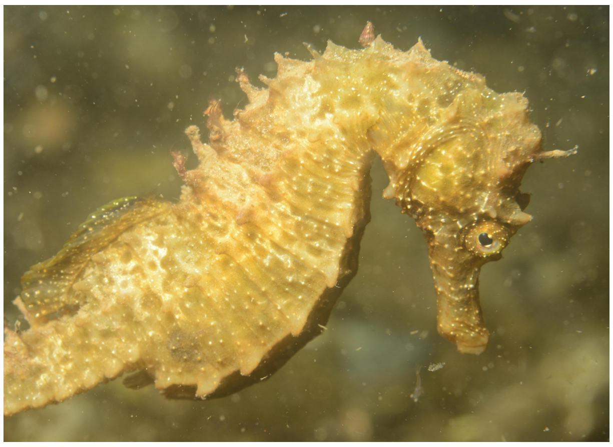


Figure 12: Picture of the right side of the individual F028

Appendix C : Examples of Yolov5 performances in validation set

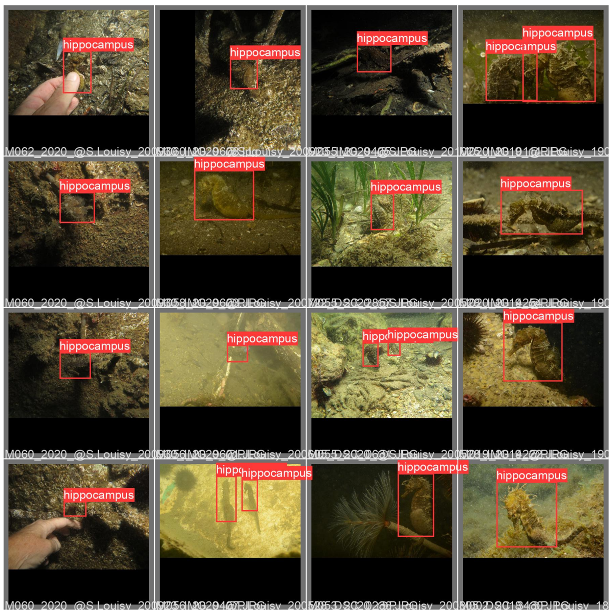


Figure 13: Annotations of a validation batch for Yolov5 training

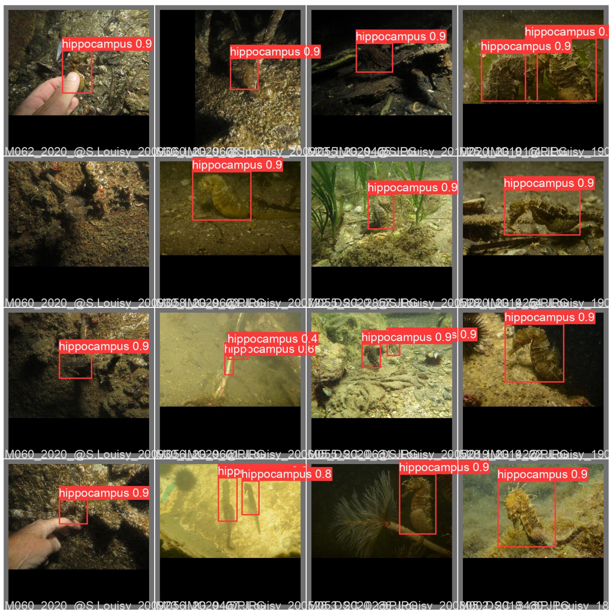
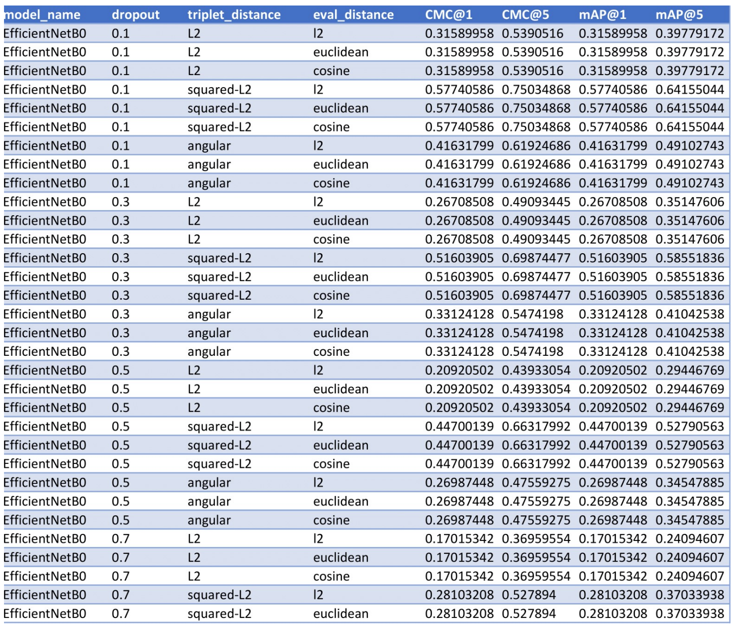


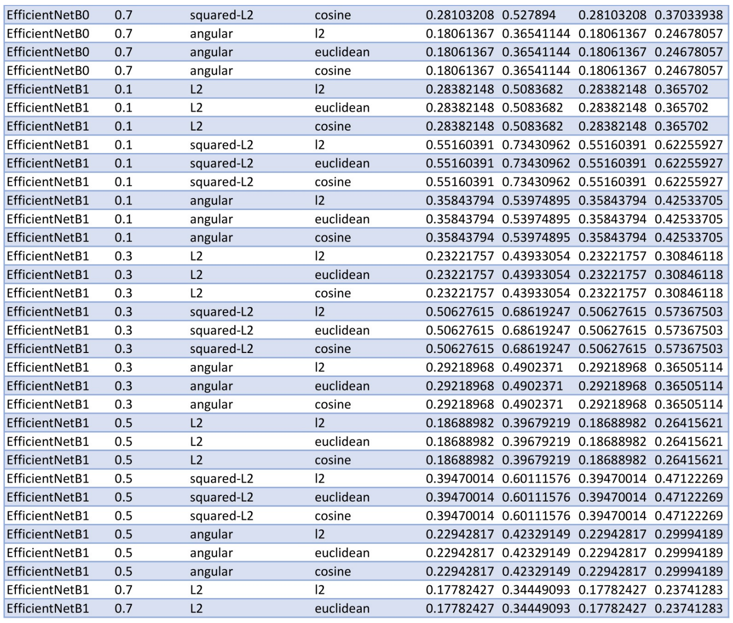
Figure 14: Predictions of the same validation batch for Yolov5 training

Appendix D : Training results for different EfficientNet architectures



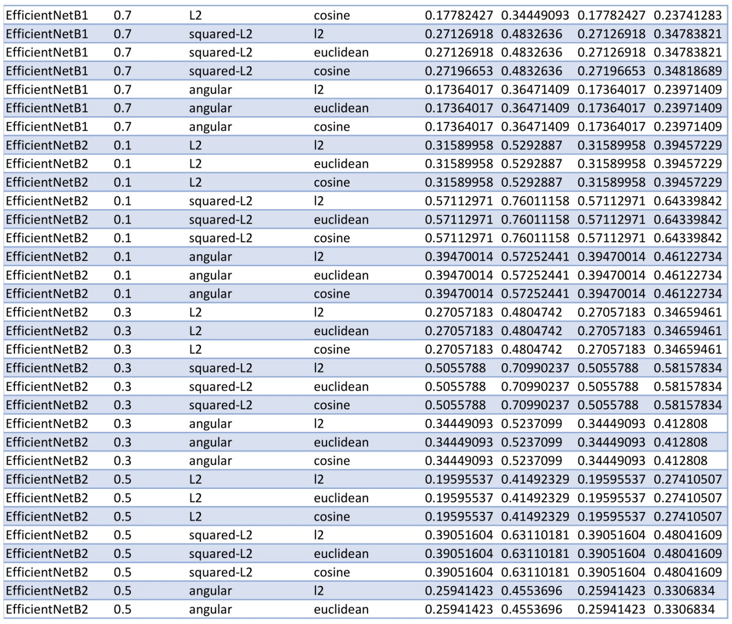
model_name	dropout	triplet_distance	eval_distance	CMC@1	CMC@5	mAP@1	mAP@5
EfficientNetB0	0.1	L2	l2	0.31589958	0.5390516	0.31589958	0.39779172
EfficientNetB0	0.1	L2	euclidean	0.31589958	0.5390516	0.31589958	0.39779172
EfficientNetB0	0.1	L2	cosine	0.31589958	0.5390516	0.31589958	0.39779172
EfficientNetB0	0.1	squared-L2	l2	0.57740586	0.75034868	0.57740586	0.64155044
EfficientNetB0	0.1	squared-L2	euclidean	0.57740586	0.75034868	0.57740586	0.64155044
EfficientNetB0	0.1	squared-L2	cosine	0.57740586	0.75034868	0.57740586	0.64155044
EfficientNetB0	0.1	angular	l2	0.41631799	0.61924686	0.41631799	0.49102743
EfficientNetB0	0.1	angular	euclidean	0.41631799	0.61924686	0.41631799	0.49102743
EfficientNetB0	0.1	angular	cosine	0.41631799	0.61924686	0.41631799	0.49102743
EfficientNetB0	0.3	L2	l2	0.26708508	0.49093445	0.26708508	0.35147606
EfficientNetB0	0.3	L2	euclidean	0.26708508	0.49093445	0.26708508	0.35147606
EfficientNetB0	0.3	L2	cosine	0.26708508	0.49093445	0.26708508	0.35147606
EfficientNetB0	0.3	squared-L2	l2	0.51603905	0.69874477	0.51603905	0.58551836
EfficientNetB0	0.3	squared-L2	euclidean	0.51603905	0.69874477	0.51603905	0.58551836
EfficientNetB0	0.3	squared-L2	cosine	0.51603905	0.69874477	0.51603905	0.58551836
EfficientNetB0	0.3	angular	l2	0.33124128	0.5474198	0.33124128	0.41042538
EfficientNetB0	0.3	angular	euclidean	0.33124128	0.5474198	0.33124128	0.41042538
EfficientNetB0	0.3	angular	cosine	0.33124128	0.5474198	0.33124128	0.41042538
EfficientNetB0	0.5	L2	l2	0.20920502	0.43933054	0.20920502	0.29446769
EfficientNetB0	0.5	L2	euclidean	0.20920502	0.43933054	0.20920502	0.29446769
EfficientNetB0	0.5	L2	cosine	0.20920502	0.43933054	0.20920502	0.29446769
EfficientNetB0	0.5	squared-L2	l2	0.44700139	0.66317992	0.44700139	0.52790563
EfficientNetB0	0.5	squared-L2	euclidean	0.44700139	0.66317992	0.44700139	0.52790563
EfficientNetB0	0.5	squared-L2	cosine	0.44700139	0.66317992	0.44700139	0.52790563
EfficientNetB0	0.5	angular	l2	0.26987448	0.47559275	0.26987448	0.34547885
EfficientNetB0	0.5	angular	euclidean	0.26987448	0.47559275	0.26987448	0.34547885
EfficientNetB0	0.5	angular	cosine	0.26987448	0.47559275	0.26987448	0.34547885
EfficientNetB0	0.7	L2	l2	0.17015342	0.36959554	0.17015342	0.24094607
EfficientNetB0	0.7	L2	euclidean	0.17015342	0.36959554	0.17015342	0.24094607
EfficientNetB0	0.7	L2	cosine	0.17015342	0.36959554	0.17015342	0.24094607
EfficientNetB0	0.7	squared-L2	l2	0.28103208	0.527894	0.28103208	0.37033938
EfficientNetB0	0.7	squared-L2	euclidean	0.28103208	0.527894	0.28103208	0.37033938

Figure 15: Training results for different EfficientNet architecture : part 1



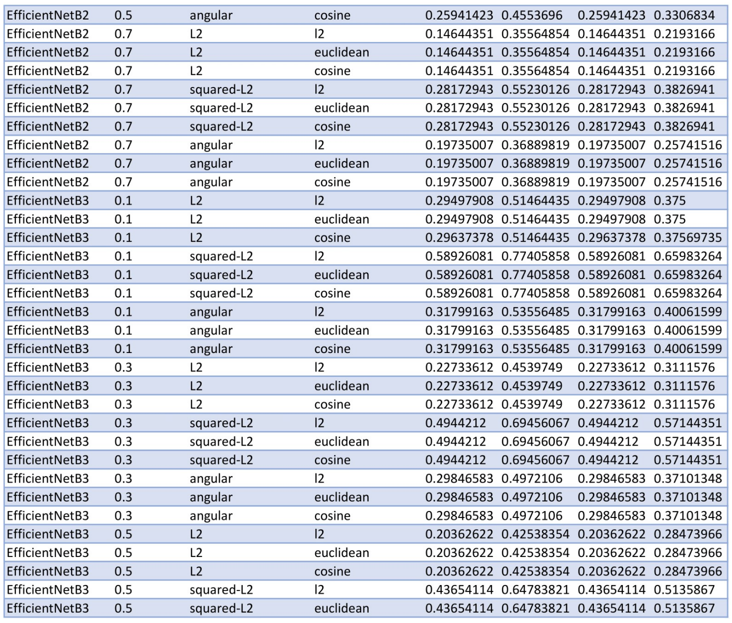
EfficientNetB0	0.7	squared-L2	cosine	0.28103208	0.527894	0.28103208	0.37033938
EfficientNetB0	0.7	angular	l2	0.18061367	0.36541144	0.18061367	0.24678057
EfficientNetB0	0.7	angular	euclidean	0.18061367	0.36541144	0.18061367	0.24678057
EfficientNetB0	0.7	angular	cosine	0.18061367	0.36541144	0.18061367	0.24678057
EfficientNetB1	0.1	L2	l2	0.28382148	0.5083682	0.28382148	0.365702
EfficientNetB1	0.1	L2	euclidean	0.28382148	0.5083682	0.28382148	0.365702
EfficientNetB1	0.1	L2	cosine	0.28382148	0.5083682	0.28382148	0.365702
EfficientNetB1	0.1	squared-L2	l2	0.55160391	0.73430962	0.55160391	0.62255927
EfficientNetB1	0.1	squared-L2	euclidean	0.55160391	0.73430962	0.55160391	0.62255927
EfficientNetB1	0.1	squared-L2	cosine	0.55160391	0.73430962	0.55160391	0.62255927
EfficientNetB1	0.1	angular	l2	0.35843794	0.53974895	0.35843794	0.42533705
EfficientNetB1	0.1	angular	euclidean	0.35843794	0.53974895	0.35843794	0.42533705
EfficientNetB1	0.1	angular	cosine	0.35843794	0.53974895	0.35843794	0.42533705
EfficientNetB1	0.3	L2	l2	0.23221757	0.43933054	0.23221757	0.30846118
EfficientNetB1	0.3	L2	euclidean	0.23221757	0.43933054	0.23221757	0.30846118
EfficientNetB1	0.3	L2	cosine	0.23221757	0.43933054	0.23221757	0.30846118
EfficientNetB1	0.3	squared-L2	l2	0.50627615	0.68619247	0.50627615	0.57367503
EfficientNetB1	0.3	squared-L2	euclidean	0.50627615	0.68619247	0.50627615	0.57367503
EfficientNetB1	0.3	squared-L2	cosine	0.50627615	0.68619247	0.50627615	0.57367503
EfficientNetB1	0.3	angular	l2	0.29218968	0.4902371	0.29218968	0.36505114
EfficientNetB1	0.3	angular	euclidean	0.29218968	0.4902371	0.29218968	0.36505114
EfficientNetB1	0.3	angular	cosine	0.29218968	0.4902371	0.29218968	0.36505114
EfficientNetB1	0.5	L2	l2	0.18688982	0.39679219	0.18688982	0.26415621
EfficientNetB1	0.5	L2	euclidean	0.18688982	0.39679219	0.18688982	0.26415621
EfficientNetB1	0.5	L2	cosine	0.18688982	0.39679219	0.18688982	0.26415621
EfficientNetB1	0.5	squared-L2	l2	0.39470014	0.60111576	0.39470014	0.47122269
EfficientNetB1	0.5	squared-L2	euclidean	0.39470014	0.60111576	0.39470014	0.47122269
EfficientNetB1	0.5	squared-L2	cosine	0.39470014	0.60111576	0.39470014	0.47122269
EfficientNetB1	0.5	angular	l2	0.22942817	0.42329149	0.22942817	0.29994189
EfficientNetB1	0.5	angular	euclidean	0.22942817	0.42329149	0.22942817	0.29994189
EfficientNetB1	0.5	angular	cosine	0.22942817	0.42329149	0.22942817	0.29994189
EfficientNetB1	0.7	L2	l2	0.17782427	0.34449093	0.17782427	0.23741283
EfficientNetB1	0.7	L2	euclidean	0.17782427	0.34449093	0.17782427	0.23741283

Figure 16: Training results for different EfficientNet architecture : part 2



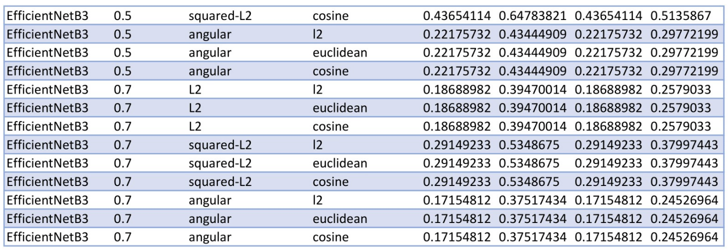
EfficientNetB1	0.7	L2	cosine	0.17782427	0.34449093	0.17782427	0.23741283
EfficientNetB1	0.7	squared-L2	l2	0.27126918	0.4832636	0.27126918	0.34783821
EfficientNetB1	0.7	squared-L2	euclidean	0.27126918	0.4832636	0.27126918	0.34783821
EfficientNetB1	0.7	squared-L2	cosine	0.27196653	0.4832636	0.27196653	0.34818689
EfficientNetB1	0.7	angular	l2	0.17364017	0.36471409	0.17364017	0.23971409
EfficientNetB1	0.7	angular	euclidean	0.17364017	0.36471409	0.17364017	0.23971409
EfficientNetB1	0.7	angular	cosine	0.17364017	0.36471409	0.17364017	0.23971409
EfficientNetB2	0.1	L2	l2	0.31589958	0.5292887	0.31589958	0.39457229
EfficientNetB2	0.1	L2	euclidean	0.31589958	0.5292887	0.31589958	0.39457229
EfficientNetB2	0.1	L2	cosine	0.31589958	0.5292887	0.31589958	0.39457229
EfficientNetB2	0.1	squared-L2	l2	0.57112971	0.76011158	0.57112971	0.64339842
EfficientNetB2	0.1	squared-L2	euclidean	0.57112971	0.76011158	0.57112971	0.64339842
EfficientNetB2	0.1	squared-L2	cosine	0.57112971	0.76011158	0.57112971	0.64339842
EfficientNetB2	0.1	angular	l2	0.39470014	0.57252441	0.39470014	0.46122734
EfficientNetB2	0.1	angular	euclidean	0.39470014	0.57252441	0.39470014	0.46122734
EfficientNetB2	0.1	angular	cosine	0.39470014	0.57252441	0.39470014	0.46122734
EfficientNetB2	0.3	L2	l2	0.27057183	0.4804742	0.27057183	0.34659461
EfficientNetB2	0.3	L2	euclidean	0.27057183	0.4804742	0.27057183	0.34659461
EfficientNetB2	0.3	L2	cosine	0.27057183	0.4804742	0.27057183	0.34659461
EfficientNetB2	0.3	squared-L2	l2	0.5055788	0.70990237	0.5055788	0.58157834
EfficientNetB2	0.3	squared-L2	euclidean	0.5055788	0.70990237	0.5055788	0.58157834
EfficientNetB2	0.3	squared-L2	cosine	0.5055788	0.70990237	0.5055788	0.58157834
EfficientNetB2	0.3	angular	l2	0.34449093	0.5237099	0.34449093	0.412808
EfficientNetB2	0.3	angular	euclidean	0.34449093	0.5237099	0.34449093	0.412808
EfficientNetB2	0.3	angular	cosine	0.34449093	0.5237099	0.34449093	0.412808
EfficientNetB2	0.5	L2	l2	0.19595537	0.41492329	0.19595537	0.27410507
EfficientNetB2	0.5	L2	euclidean	0.19595537	0.41492329	0.19595537	0.27410507
EfficientNetB2	0.5	L2	cosine	0.19595537	0.41492329	0.19595537	0.27410507
EfficientNetB2	0.5	squared-L2	l2	0.39051604	0.63110181	0.39051604	0.48041609
EfficientNetB2	0.5	squared-L2	euclidean	0.39051604	0.63110181	0.39051604	0.48041609
EfficientNetB2	0.5	squared-L2	cosine	0.39051604	0.63110181	0.39051604	0.48041609
EfficientNetB2	0.5	angular	l2	0.25941423	0.4553696	0.25941423	0.3306834
EfficientNetB2	0.5	angular	euclidean	0.25941423	0.4553696	0.25941423	0.3306834

Figure 17: Training results for different EfficientNet architecture : part 3



EfficientNetB2	0.5	angular	cosine	0.25941423	0.4553696	0.25941423	0.3306834
EfficientNetB2	0.7	L2	l2	0.14644351	0.35564854	0.14644351	0.2193166
EfficientNetB2	0.7	L2	euclidean	0.14644351	0.35564854	0.14644351	0.2193166
EfficientNetB2	0.7	L2	cosine	0.14644351	0.35564854	0.14644351	0.2193166
EfficientNetB2	0.7	squared-L2	l2	0.28172943	0.55230126	0.28172943	0.3826941
EfficientNetB2	0.7	squared-L2	euclidean	0.28172943	0.55230126	0.28172943	0.3826941
EfficientNetB2	0.7	squared-L2	cosine	0.28172943	0.55230126	0.28172943	0.3826941
EfficientNetB2	0.7	angular	l2	0.19735007	0.36889819	0.19735007	0.25741516
EfficientNetB2	0.7	angular	euclidean	0.19735007	0.36889819	0.19735007	0.25741516
EfficientNetB2	0.7	angular	cosine	0.19735007	0.36889819	0.19735007	0.25741516
EfficientNetB3	0.1	L2	l2	0.29497908	0.51464435	0.29497908	0.375
EfficientNetB3	0.1	L2	euclidean	0.29497908	0.51464435	0.29497908	0.375
EfficientNetB3	0.1	L2	cosine	0.29637378	0.51464435	0.29637378	0.37569735
EfficientNetB3	0.1	squared-L2	l2	0.58926081	0.77405858	0.58926081	0.65983264
EfficientNetB3	0.1	squared-L2	euclidean	0.58926081	0.77405858	0.58926081	0.65983264
EfficientNetB3	0.1	squared-L2	cosine	0.58926081	0.77405858	0.58926081	0.65983264
EfficientNetB3	0.1	angular	l2	0.31799163	0.53556485	0.31799163	0.40061599
EfficientNetB3	0.1	angular	euclidean	0.31799163	0.53556485	0.31799163	0.40061599
EfficientNetB3	0.1	angular	cosine	0.31799163	0.53556485	0.31799163	0.40061599
EfficientNetB3	0.3	L2	l2	0.22733612	0.4539749	0.22733612	0.3111576
EfficientNetB3	0.3	L2	euclidean	0.22733612	0.4539749	0.22733612	0.3111576
EfficientNetB3	0.3	L2	cosine	0.22733612	0.4539749	0.22733612	0.3111576
EfficientNetB3	0.3	squared-L2	l2	0.4944212	0.69456067	0.4944212	0.57144351
EfficientNetB3	0.3	squared-L2	euclidean	0.4944212	0.69456067	0.4944212	0.57144351
EfficientNetB3	0.3	squared-L2	cosine	0.4944212	0.69456067	0.4944212	0.57144351
EfficientNetB3	0.3	angular	l2	0.29846583	0.4972106	0.29846583	0.37101348
EfficientNetB3	0.3	angular	euclidean	0.29846583	0.4972106	0.29846583	0.37101348
EfficientNetB3	0.3	angular	cosine	0.29846583	0.4972106	0.29846583	0.37101348
EfficientNetB3	0.5	L2	l2	0.20362622	0.42538354	0.20362622	0.28473966
EfficientNetB3	0.5	L2	euclidean	0.20362622	0.42538354	0.20362622	0.28473966
EfficientNetB3	0.5	L2	cosine	0.20362622	0.42538354	0.20362622	0.28473966
EfficientNetB3	0.5	squared-L2	l2	0.43654114	0.64783821	0.43654114	0.5135867
EfficientNetB3	0.5	squared-L2	euclidean	0.43654114	0.64783821	0.43654114	0.5135867

Figure 18: Training results for different EfficientNet architecture : part 4



EfficientNetB3	0.5	squared-L2	cosine	0.43654114	0.64783821	0.43654114	0.5135867
EfficientNetB3	0.5	angular	l2	0.22175732	0.43444909	0.22175732	0.29772199
EfficientNetB3	0.5	angular	euclidean	0.22175732	0.43444909	0.22175732	0.29772199
EfficientNetB3	0.5	angular	cosine	0.22175732	0.43444909	0.22175732	0.29772199
EfficientNetB3	0.7	L2	l2	0.18688982	0.39470014	0.18688982	0.2579033
EfficientNetB3	0.7	L2	euclidean	0.18688982	0.39470014	0.18688982	0.2579033
EfficientNetB3	0.7	L2	cosine	0.18688982	0.39470014	0.18688982	0.2579033
EfficientNetB3	0.7	squared-L2	l2	0.29149233	0.5348675	0.29149233	0.37997443
EfficientNetB3	0.7	squared-L2	euclidean	0.29149233	0.5348675	0.29149233	0.37997443
EfficientNetB3	0.7	squared-L2	cosine	0.29149233	0.5348675	0.29149233	0.37997443
EfficientNetB3	0.7	angular	l2	0.17154812	0.37517434	0.17154812	0.24526964
EfficientNetB3	0.7	angular	euclidean	0.17154812	0.37517434	0.17154812	0.24526964
EfficientNetB3	0.7	angular	cosine	0.17154812	0.37517434	0.17154812	0.24526964

Figure 19: Training results for different EfficientNet architecture : part 5