If There's a Whale There's a Way
ML Final Project Report

Maren Rieker
212723@mds.hertie-school.org

Dinah Rabe
d.rabe@mpp.hertie-school.org

Reed Garvin
r.garvin@mds.hertie-school.org

Victor Möslein
212963@mds.hertie-school.org

Abstract

Man-made climate change is one of the main challenges for humanity and of concern for many scientific disciplines investigating its multiple, complex impacts on nature and societies. One of the ecosystems impacted are marine ecosystems. Research concerning marine mammals makes an important contribution to our knowledge about changes in the oceans. Marine mammals are indicators of ocean ecosystem health and therefore their protection and conservation is a crucial task. Up until today, most observation and tracking of individual animals and populations is done manually by humans - a time consuming and error-prone method. Motivated by a Kaggle competition funded by the research collaboration "Happywhale", we aim to contribute to the efforts by automating whale and dolphin photo-ID to significantly reduce image identification times using conventional Machine Learning and a Deep Learning algorithm for image classification [1]*. We assess performance in terms of the prediction accuracy, precision, recall and F1 scores, as well as training times. As expected, and under constraints of computational feasibility, we achieved comparably low metrics using conventional classifiers (11.6% precision, 17% accuracy), and substantially better results with a Deep Learning model (60% accuracy on the species, 11% mean average precision on the individuals). The results indicate the need for complex Deep Learning models to correctly distinguish between subtle characteristics of the natural markings of whales and dolphins, which is furthermore confirmed by the winning entries of the Kaggle competition.*[2]

[1] GitHub Link:
https://github.com/Whale-way/happy-whale

[2] Kaggle Results:
https://www.kaggle.com/competitions/happy-whale-and-dolphin/leaderboard

1. Introduction

For several years, the impacts of man-made climate change have been specifically visible through the alteration and destruction of the marine ecosystem.[1, 9] Marine mammals are indicators of ocean ecosystem health. A worrying development is the change in the migratory behaviour of whales, dolphins and sharks as it is a strong indicator for the destruction of the ecosystems they are living in. The protection and conservation of marine mammals is crucial to the balance and therefore the health of ecosystems. To be able to carry out meaningful conservation efforts, the first steps are understanding the status quo of different animal populations and their migration patterns by identification and monitoring of individual animals. The identification by natural markings, via photographs is known as photo-ID. Currently, the majority of research institutions still rely on ressource-intensive and sometimes inaccurate manual matching of photographs by the human eye. This slow and imprecise practice naturally limits the scope and impact of existing research. The automatized Image Classification of whales and dolphins could enable a scale of study previously unaffordable or impossible.

There are first attempts of using Machine Learning in marine biology and environmental protection to address this challenge. Most of the research either uses recordings of dolphin or whale sounds for classification e.g. [7], or photographs of fins [13, 8], given that these are the two most common types (sounds or images) of documenting wild animals. Current methods of these automated classification attempts are mostly Deep Learning (DL) approaches using convolutional neural networks (CNN). Regarding standard Machine Learning (ML) techniques, however, there is not much research to be found. This might be due to the challenging task of distinguishing between unique – but often very subtle – characteristics of the natural markings of whales and dolphins. In this paper, we are developing on the scarce literature on the use of ML for automated photo-ID and test it against the state of the art

use of DL for such tasks.
Our approach focuses mainly on prediction precision and accuracy, and only secondary on prediction and training time, and is limited to images of dorsal fins and lateral body views. We started to break down the task into three main challenges: (1) Removing the animal from the rest of the picture using Image Segmentation, (2) using and testing ML classifiers for species classification and (3) implementing a DL model to extend the use case from the classification of species to individual animals.

Our results show that the subtle differences between 26 whale and dolphin species cannot be detected by classic ML classifiers with great precision. We demonstrate that while it is indeed possible to train the widely used ML models Logistic Regression, Random Forest and XGBoost on a large dataset of images, the limits of computational power are quickly reached before the classifiers are possible of reaching a high precision. Different approaches to hyperparameter tuning that were applied over the course of the project were not executable in a reasonable time and are furthermore incapable of improving precision scores. In fact, we show that the classifiers will never succeed in reaching precision and accuracy scores high enough for them to be applied on real-world tasks of predictions on image data with only minimal differences between classes. Consequently, our analysis finds that advanced Deep Learning models are needed, and that they achieve great accuracy and precision in computer vision tasks and are specifically well-suited to automate mundane image recognition jobs like photo-ID.

2. Related Work

Automated photo-ID with ML & DL: The task of automating the identification of species or individuals through photo-ID is not entirely new to the research world. So far, most of the proposed solutions to this problem have been based on Convolutional Neural Networks (CNN) and related Deep Learning methods. Contributions addressing the effectiveness of conventional ML classifiers to challenges as complex as the one at hand remain scarce. Faaeq et al. (2018) apply Machine Leaning algorithms to animal classification of a data set of 19 different animals [3]. Differently to our dataset, the animals in their data were seen in their entirety on the image, and were of entirely distinct species (e.g. Lion, Elephant, Deer). Logistic Regression produced the best accuracy in their case (0.98). Regarding speed, Random Forest delivered the best results, beating Logistic Regression by a factor of eight. However, the researchers conclude that more powerful image classification models, specifically Deep Learning methods, are needed for classification tasks on more complex image datasets. Dhar and Guha (2021) find XGBoost to perform best in predicting the correct fish species, scoring a significantly higher precision score than Random Forest, k-Nearest-Neighbor (kNN) and Support Vector Machines (SVM) [2]. In their research, two different datasets containing 483 respectively 10 different fish species were used. Again, the images in the dataset depicted the entire fish, which have significant differences in shape, colors and other features, as opposed to only body parts with little differences, as in our dataset. However, these two papers provided us with guidance as to which Machine Learning models are most promising for our project.

Image Segmentation In the past, image segmentation algorithms with different approaches have emerged, but the emergence of Deep Learning models has "caused a paradigm shift" [11] in the field of image segmentation, because they perform so well. Image segmentation with Deep Learning is a widely researched field of computer vision with wide and diverse applications in real life, like autonomous driving, medical imaging, video surveillance, and saliency detection [5]. There is a plethora of widely used algorithms for different fields of application which can be differentiated by being (un-)supervised, their type of learning and the level of segmentation (ibid.). For this project, we used *Tracer*, a convolutional attention-guided tool. *Tracer* performs remarkably well at a relatively low computation time, which made it appropriate for us to use [10].

3. Proposed Method

The research process consists of the following steps: First, the images are segmented to remove the unnecessary background with *Tracer*. Then a baseline for species prediction is established with an untuned ML algorithm. In the following, two other advanced ML models are implemented and tuned in terms of speed and classification precision and accuracy. As the final part of our project, a Deep Learning algorithm capable of predicting the species as well as individual animals is implemented.

As the goal of the Kaggle competition is the prediction of individual animals out of a pool of 18,000 individuals from 26 species, we decided to divide the project into two stages after preprocessing the images. In Stage 1, only the species are predicted (therefore 26 classes), to be able to apply ML algorithms. For Stage 2, in line with common practice, we implement a Deep Learning model able to predict on the level of individuals (18,000 classes). Based on this setup, the research process consists of the following steps:

STAGE 1: Conventional ML approach
(1) Apply PCA to create manageable data set
(2) Establishing the baseline with untuned ML algorithm

(3) Establishing two advanced ML algorithms and tune them for precision, accuracy and speed

STAGE 2: Deep Learning approach
(1) Model and different Outcomes
(2) Deep Learning Pipeline

3.1. Data Preprocessing

All data pre-processing was performed separately from the modelling, as this is a one time operation. In the following, the different steps of the pre-processing pipeline are described.

Image Segmentation Separating the animals from the background was an elementar step of the picture preprocessing. In our case, the background of the pictures was not relevant for our research interest and therefore did not have to be included in the images we trained our models with. Contrary, the background would only have been disturbing, inflated the data size and disrupted the ML/DL pipelines by making more elaborate calculations necessary. For the segmentation of the images, we use *Tracer* [10]. *Tracer* is a Deep Learning model that works with already cropped images and a specified pixel size. Because our pictures have the size 512x512, we employed the TRACER-efficient-5 model, as specified in the ReadMe.

The model works on the basis of pre-trained weights from efficientnet B7.[3] From there, *Tracer* derives which elements in the training image are part of the object and which are part of the background and are going to be coloured white. In order to make this decision, the program sets thresholds in the variation in the pixels. *Tracer* repeatedly attempts (Epochs) to find the edge of the object. After the first attempts at finding the edge, this attempt is removed and compared to the final attempt at mass edged attention module, creating the segmented image. This comparison attempts for any loss that may have occurred to be accounted for, therefore producing a more accurate segmented image.

Data Cleaning After the segmentation, some final data wrangling is necessary. This includes (1) the cleaning of the provided .CSV-file that includes the individual-IDs (labels for Stage 2), the species (labels for Stage 1) and the filename, as we discovered misspellings in the species column. Additionally, (2) images have to be turned into machine-interpretable data. For this, the package NumPy is used, which can extract all numerical information stored in the color scheme (RGB in our case) by just applying the array method to a picture opened and stored in a variable. For later purposes, a function to resize the picture to different pixel sizes is implemented. The Deep Learning model uses a 224x224 resolution. For the ML models, the images were reduced further to a resolution of 48x48. The reason for that is that the Principle Component Analysis as well as the ML models without a PCA were unable to be run on a larger resolution. Having more computational power (especially CPU power and memory) would have been necessary to keep the resolution at 224x224 pixels, as was originally planned. The last step is to save the transformed datasets, so that the other classification modules can access the numerical representation of the images without re-running the whole preprocessing every time.

[3] https://keras.io/api/applications/efficientnet/

3.2. Stage 1 Conventional ML approach

Apply PCA to create manageable dataset As the images we are working with have an already reduced resolution of 48×48 pixels and are colored using the RGB scale, the number of features is $48 \times 48 \times 3 = 6,912$ per image. We implemented a Principal Component Analysis (PCA) [4] in addition to the reduced image size to further reduce dimensionality and increase training speed. As shown in Figure 1, the segmented images contain a large amount of white pixels, which can be dropped without loosing any information. Furthermore, noise and unnecessary details, which do not facilitate the classification, may be dropped.

Originally, it was planned to tune, train and test the ML models first on the images before PCA is applied and then a second time after PCA has been applied on the data. This would have allowed to compare the scores after applying PCA to the scores on the entire dataset and thus validate this assumption. However, running any model was impossible for us with the aforementioned $6,912$ features per image, as the available hardware could not handle such high dimensionality. For this reason, the application of PCA was moved to the beginning, and then the reduced dataset after PCA was used for the following steps (the Classification script still allows to use the entire dataset and then apply PCA at the end and retrain the models in order to obtain the comparison).

Establishing the baseline with untuned ML algorithm To establish a baseline, we are implementing a Softmax Regression using the Scikit Learn Multiclass Logistic Regression module. Softmax Regression, also known as Multiclass - or Multinomial Logistic Regression and Maximum-Entropy Classifier, is based on the two-class Logistic Regression and generalized to work on multiple mutually exclusive classes [12]. Researchers often rely on Softmax as

a conventional supervised ML method for the good classification performance (see Related Work section). As Softmax delivers similarly accurate results as kNN and SVM and needs significantly less time when working with large datasets, it is the appropriate method for us. This baseline classifier is trained and evaluated in terms of the precision, accuracy, F1 and recall scores. The findings and test configurations are specified in section 4.

Establishing and Tuning two advanced ML algorithms Implementing and training a Random Forest (RF) model and XGBoost with early stopping model is done in a next step, to be able to compare the three selected ML models. RF classifications are popular for achieving similarly good prediction results as kNN, SVM and Softmax, while being significantly faster than those, which is why we are choosing it as a second ML model [3]. Additionally to their good training speed, RF has a number of other advantages that make it useful for our classification task. Most importantly, RF is not sensitive to over-fitting and can handle unbalanced datasets, which enables trying out different hyperparameter configurations [6]. The XGBoost model was selected due to its high popularity for classification tasks in the context of Kaggle competitions and to compare a bagging and a boosting approach to the problem at hand. While being generally slower than Random Forest and Logistic Regression, XGBoost is suitable for the application at hand since the low training speed is not the only crucial criterion and XGBoost is known for achieving better classification than Random Forest [2]. Being also a tree-based method, using XGBoost allowed us to compare two similar ML classifiers on the same hyperparameters.

To improve the performance of our selected models we optimize the hyperparameters in an iterative step under the constraints of the computational power available to us. In a first attempt, Scikit-Learn's GridSearchCV method was applied. As this took prohibitively long to run, the more efficient RandomizedSearchCV method was used next, which randomly picks out the parameters also used by GridSearchCV and applies cross validation on every randomly selected parameter set, making it possible to decrease the number of fits. However, this attempt also resulted in runtimes above 20 hours for both Random Forest and XGBoost.

Therefore, a Learning Curve approach was chosen to overcome those computational barriers. This method also uses GridSearchCV, however with a pre-specified maximal depth of the trees and the only parameter to be tuned being the number of trees (i.e., number of estimators). The learning curve depicts the precision score vs. number of trees, as well as the training time vs. number of trees for both a training set and a testing set used in the cross validation. Therefore, the best number of trees can easily be identified, which delivers the best precision while not risking over-fitting and having a good training time.

3.3. Stage 2 Deep Learning Approach

Model and Different Outcomes The Deep Learning model we have employed is a combination of the features of a Transformer and convolutions. This EfficientNet uses a combined Convolutional Neural Network (CNN) and a more efficient version of ResNet trained on Image Net. [4] We have chosen this model for two reasons: the performance recommendation of other Kaggle users in the competition and how lightweight this model is. This ensures high performance, great accuracy and a model that can efficiently, in terms of computational capacity, be carried out on the Hertie Server.

The Deep Learning outcomes can be broken into two parts - one being our submission for the Kaggle competition, which looks at individuals classification, and secondly, species-based prediction for comparing our results with the non Deep Learning portion of this project. To achieve this, we have altered the code to allow for one output to match the requirements of the Kaggle competition and the other gives us the top 5 most likely species of a given image, including "New Species" which implies that it is most likely a species not given in the training dataset.

Deep Learning Pipeline First our model shrinks the images to an array of 32x32x3 to create a more manageable array of the over 50,000 images. To create the array with our images, we used Keras' image data preprocessing package. This uses a Keras version of PIL image package that opens images and automatically transforms them from an image to a tensor array ready for use in our model. Then, using a label encoder, we create the Y values from both the individual IDs and species. For this step we used two models, one based on EfficentNetB7 and a secondly a CNN based model on our predictions from the first model. Afterwards we find the top 5 possibilities plus the option for a "new individual" and output them into a csv file.

4. Experiments

Data Our project is based on a Kaggle competition.[5] They provide a pre-split dataset, containing 27,956 images in the test set and 51,033 images in the training set of whales and dolphins. Additionally, a .CSV-file is provided that contains filename (of the corresponding image), individual-ID

[4] https://keras.io/api/applications/efficientnet/

[5] The competition can be found here: https://www.kaggle.com/c/happy-whale-and-dolphin

and the species. Images focus on dorsal fins and lateral body views. The raw images have different pixel resolutions.
In the course of the project we decided to use a pre-cropped dataset provided by a fellow Kaggle-competitor [6] to focus on the implementation of the classifiers.

Figure 1. Example of original cropped image and after segmentation. Source: Own illustration

As the corresponding labels to the test-set images were not published after the end of the Kaggle competition we could only use the 51,033 images in the training set as the basis for our ML and DL models. Before the modelling, we split our dataset into a training set (30.000 images), a validation set (10.826 images) and a test set (10.207 images). The splitting was stratified to achieve a similar distribution of classes in all sets. Important to note is that the dataset is quite unbalanced, both with regards to species as well as individuals. There are 8 (out of 26) species which, pre-splitting, were only represented less than 200 times compared to the top three classes that are represented each over 5000 times. The challenges this poses will be discussed in the following paragraphs and chapters.

Software We implemented the whole project with Python using Jupyter Notebook as an development environment. Furthermore, we used GitHub and GitHub Desktop for version control and file sharing of smaller files. Lastly, we relied on AirDrop to share larger files like the image data folders and the numerical input files resulting from the data pre-processing.

Hardware The code development took place on the laptops of the team members, testing it with the a small sample/dummy dataset containing 100 random pre-processed images to make sure that the code runs properly before performing the actual pre-processing and modelling with the full dataset on the Hertie Server. Due to the temporarily unavailability of the Hertie Server and the CPU-heavy tasks performed, the entire training, tuning and testing of Stage 1 had to be performed on a 2017 Macbook Pro with Intel Core i5 processor and 8GB RAM in 2 CPU cores. Stage 2 was run on the Hertie School Server which has 4 NVIDEA A100 GPU with a total of 40GB of processing power.

Evaluation method For the **evaluation of our ML classifiers**, we use the following metrics: the confusion matrix and the accuracy, precision, recall and F1 score. To have an initial check for over-fitting we evaluate the classifiers both on the training and a separate evaluation set. An alternative would have been to use a cross validation score, but due to the large amount of available data we decided to split the training set again. The confusion matrix is used to give a graphical representation of how well the classifiers are predicting the true label. We decided to use the confusion matrix to get an idea how well (or difficult) the prediction of different species is and how well the different models are performing - having the unbalanced dataset in mind.
Classification accuracy is the measure of accurate classification results. The precision and recall scores are the scores which determine the share of true positives in all instances classified as positive (precision) and in all positive instances in reality (recall). For our purposes, it is most desirable to have a good ratio of true positive outcomes among positive classified images. In line with this description of the importance of a low number of false positives over the low number of false negatives, a higher precision score is more important than a higher recall score regarding the quality of our outcome. We, additionally, use the F1-score, which is a sub-contrary mean of recall and precision, to combine them. Overall, the accuracy and precision are the two most important metrics for us, as these are the ones also used in the Kaggle competition, our Deep Learning classification, and in most image classification literature. Additionally, the training times are analyzed.

For the **evaluation of our Deep Learning classifier**, we used categorical crossentropy, which is a loss function that is used in mutually-exclusive multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.
For the evaluation of the results of the individuals precision, the mean average precision score at 5 was calculated by Kaggle. This allows to predict five IDs per image, whereby a score of one is reached for the image if one of those five is correct. For the purpose of species prediction, the accuracy score and precision scores were used as the primary metric, to make the results of the ML models comparable.

Experimental details The **ML models** were configured as follows. The PCA was configured to retain 95% of the variance, which lead to 502 features per image being retained. The baseline Logistic Regression was im-

[6] https://www.kaggle.com/datasets/phalanx/whale2-cropped-dataset

plemented using the Stochastic Average Gradient descent solver, which is especially suited for large datasets, and 500 maximal iterations. It was trained in 82.40 seconds. As mentioned above, the tuning of hyperparameters for both Random Forest and XGBoost were done using a Learning Curve. Multiple test runs of fitting the models with different depths and number of trees were first performed manually so that the training time for the cross-validation grid search could be estimated. It was further concluded that these two parameters are the only ones to be tuned as they control the variance-bias trade-off and because it was computationally infeasible to tune more hyperparameters. It was concluded that the maximum depth of the trees needs to be fixed to 10, while the number of estimators should be examined on a range from 10 to 500. Furthermore, the scoring is done by using the weighted precision, which takes into account class-imbalance.

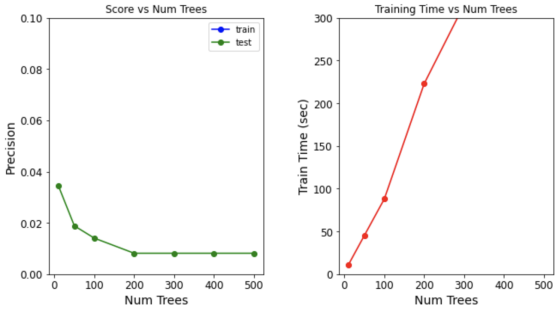

Figure 2. Learning Curve of Random Forest classifier. Source: Own illustration

A highly uncommon result was obtained, namely that the precision decreased with increasing number of trees. Figure 2 illustrates on the left side that the best precision is obtained with just 10 trees. The line for the training set is not visible, but follows a similar behavior. The precision for the training set with 10 trees is at 0.69, indicating over-fitting. However, the precision on both the training and the test set drops dramatically already for 50 trees while the training time increases significantly when increasing the number of trees. This unusual behavior can only be explained because of the complexity of the dataset and the incapacity of Random Forest to handle the subtle differences between the species.

As the time to run the same hyperparameter grid on the XGBoost Classifier was too long (above 10 hours), and a comparison between the method should be estalished, it was concluded that the resulting set of best hyperparameters will be used for both Random Forest and XGBoost. Thus, the number of trees and the maximum depth are both set to 10. We further set the "class_weight" parameter of the Random Forest Classifier to "balanced_subsample" to account for the imbalance of our dataset. Furthermore, the maximum number of features to consider when looking for the best split is set to "sqrt". The XGBoost classifier is implemented with three rounds of early stopping in order to increase training speed. The labels are encoded into numbers, as XGB cannot handle strings as dependent variables. Furthermore, to account for the class imbalance, "scale_pos_weight" is set to 500, as the factor of over-representation of the largest class compared to the smallest is around 500. We are using the evaluation metric Receiver Operating Characteristic Area under the Curve ("auc") as the evaluation metric for the validation data and the softmax function for multiclass-classification ("multi:softprob") as learning objective. The tuned models were tested against the untuned models on the training and validation sets. As our tuned hyperparameters are aiming to improve precision, this scores significantly increases from the baselines to the tuned RF and XGB models, with the other scores remaining roughly constant. This set of hyperparameters was thus used for the evaluation on the test set.

In the first attempt of the **Deep Learning model** we attempted to use six layers. The following were not used as the Hertie Server could not run them: a global average pooling layer, a drop out layer, and an extra 512 output layer with reLU activation. The layers we used in our first model include: one layer with an 1024 output and a reLU activation, a layer to flatten the model, and one layer to create percentages using softmax activation. We then used the gc collect package to clear any memory before running the model. Followed by another CNN model to apply to our data by using Conv2D, a CNN for image processing twice, next a flatten layer and a dense layer with 512 neurons. This is then ended with a soft max activation layer for percentages. To ensure peak learning of the model, we used two Keras packages that stop the learning when it is no longer improving, called reduceLROnPlateau and EarlyStopping. This helps to protect the model from over-fitting and ensures that it is as accurate and precise as possible. The overall time when running 500 epochs (the most epochs we ran) was, on average, 8 hours when using individual IDs as classifiers, with each epoch taking an average of 40 seconds.

Results For the **ML models** the following results on our evaluation scores were obtained after predicting the labels in the test set. The tuned XGBoost Classifier yielded the best scores in every category except for the training time, with a top precision score of 11.8% and accuracy of 17%. The training time of 26 minutes 48 seconds is significantly higher than that of the Softmax (LG) and Random Forest models. Notably, the precision of the baseline yielded almost similarly positive results to the tuned XGBoost model,

and is being trained much faster. The tuned Random Forest model achieves the best training time with only 11.94 seconds, and achieves a precision score of 10.2% and performs thus not much worse in that regard than LG and XGBoost.

	Accuracy Score	Precision Score	Recall Score	F1 Score	Training Time
LG: Baseline	0.11	0.116	0.11	0.112	82.4 sec
RF: Tuned	0.026	0.102	0.026	0.034	11.94 sec
XGBoost: Tuned	0.17	0.118	0.17	0.124	26 min 48 sec

Figure 3. Machine Learning Results

The accuracy of the untuned XGB was 0.168, the precision 0.036, the recall 0.04 and the F1-score 0.025. The accuracy of the untuned RF was 0.11, the precision 0.044, the recall 0.21 and the F1-score 0.073. The confusion matrices of the three models furthermore yield interesting insights (see Appendix B). XGBoost consistently predicts those classes that occur more often in the dataset, while Random Forest and Softmax handle the imbalance in the data set much better. There are relatively large values in off-diagonal cells in the first few columns of the confusion matrix of the Softmax and XGBoost models, meaning that the classifiers predicted bottlenose dolphins, belugas and humpback whale many times when the label was in fact different. This is not surprising as those are the three biggest classes, and the classifier could get a lot of its predictions right just by guessing one of those classes.

The **Deep Learning models** yielded the following scores:

Deep Learning	Precision Score	Loss Score	Time
B0: Baseline	0.00005	~1.5	~3 Hours
B1: Tuned	0.11	~1	~8 Hours
Species Eval	(Accuracy Score) 0.63	1.3	3 min

Figure 4. Deep Learning Results

The Baseline model had a mean average precision score of 0.00005 after 200 epochs, but by switching from EficentNetB0 to B1 and increasing the number of epochs to 500, we were able to increase the score to 11%. Running the adjusted model, however, more than doubled the running time from around 3 to around 8 hours. The loss of the baseline model is around 1.5, and for the tuned model around 1. Our model in predicting species was already quite high in accuracy at 60% after only two epochs, showing its performance when presented with 26 vs over 11,000 classes. Its loss score was 1.3 and it ran in only three minutes.

Comment on quantitative results For the **ML models** it was expected to achieve a low performance. In that regard, the scores of our models achieve better results than expected, especially as the precision score is above 10% for each classifier. An issue that persists across all our models is the high degree of over-fitting to the training data. The implemented ML models were not capable of handling the complexity of the data set, especially since more intense grid search cross-validation could not be performed given the limitations in computational power. It is however questionable whether more hyperparameter tuning would even have been helpful and achieved significantly better results (see Learning Curve in Figure 2). To point this out visually, a graph showing the feature importance of the Random Forest model was created (see Figure 5). The plot is created based on the dataset before PCA was applied and thus depicts the 48x48 pixel resolution of the images.

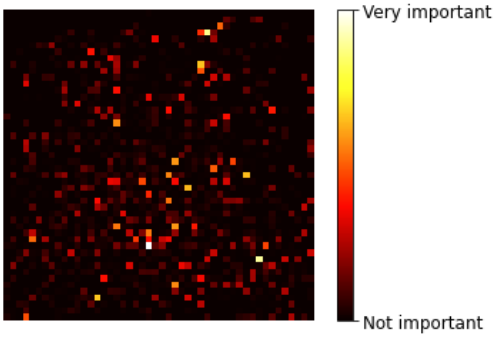

Figure 5. Feature importance of Random Forest classifier before PCA. Source: Own illustration

The plot shows which features are contributing significantly to the accuracy of the model, and which ones are just noise. As becomes obvious, the classification task is extremely difficult and no feature is especially useful. The lightest pixels have a feature importance of 0.014. It was expected that some form of fin of a dolphin or whale would be visible. However, only a small number of pixels in the lower part of the images seem to be important and helpful for the classification task, which explains the low scores. It is notable how good the scores of the baseline Softmax regression are, outperforming Random Forest in every metric except speed. This is largely in line with the literature and also comes as a result of the complexity of our dataset. The confusion matrixes furthermore show that RF handled the prediction of smaller classes much better than Softmax. Additionally, the large deviation in training times is surprising, as XGBoost with early stopping was not expected to take as significantly longer to train. The fast training time of Random Forest was expected given its known characteristics from the literature and the low number of trees.

The **Deep Learning models** were expected to achieve higher precision than 0.00005, however after changing our parameters after our Baseline results by increasing the number of epochs and switching to a different version of EfficentNet we were able to increase both our accuracy and precision score, while decreasing our loss. As a result, we were able to deduce that increasing the number of epochs would increase precision by comparing our results to those of other Kagglers.

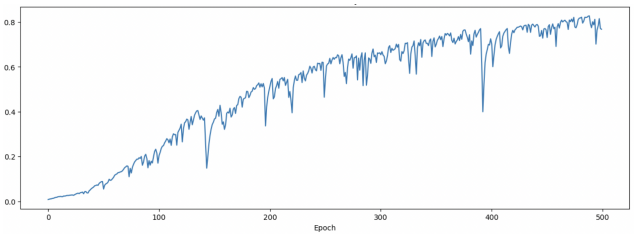

Figure 6. B1 Mean average precision versus number of Epochs

Figure 6 shows the mean average precision of the Deep Learning Model with increasing number of epochs. It becomes obvious from the graph that increasing the number of epochs beyond our further 500 epochs would not have resulted in a significant increase in prediction performance.

5. Analysis

Next to our quantitative assessment we looked more deeply at errors and their origin in the dataset. For this task we created error tables that enable us to identify the image IDs for pictures that were predicted wrongly. In addition, the confusion matrix is used to get an idea of the error distribution. This analysis is for sake of clarity limited to the tuned classifiers. Considering the multiple steps of our project, the first possible source for errors is the pre-cropped dataset which includes distorted and blurred images that could be considered outliers. Considering the imbalance of our dataset, there is a large amount of errors stemming from underrepresented species. There are multiple classes which none of the ML classifiers was able to correctly predict at all. This could be due to the fact that there are some species which do not have very unique or distinguishable fins (or no fins at all, like beluga whales). Therefore separately training and setting the thresholds for each species could be a way forward to improve prediction success.

6. Conclusions

This project confirms current research practice that conventional ML algorithms are not solving complex classification tasks in a very precise and fast manner, and that Deep Learning approaches are needed for tasks such as automating photo-ID. We achieved and accuracy scores of 17% with the conventional ML models on species identification, but accuracy of over 60% with a only limitedly trained Deep Learning model. Especially when aiming for the original goal of the Kaggle Competition to identify individual animals, a extensively trained DL model is needed to build on the 11% accuracy achieved.

Limitations Our research was bound by resource constraints, especially in terms of time and computational power. Keeping that in mind, in the following we want to discuss some possible improvements. The project could have been improved with more extensive hyperparameter tuning, and by including further hyperparameters. We do not expect the results for the conventional ML models to increase in a way that they are comparable to the DL model, but certainly some improvements in prediction performance. As discussed, DL models dominate the literature, therefore we would suggest to focus the improvements on the DL model. For the implemented Deep Learning model, it would have been beneficial to have more GPU power and training time to train it more extensively. The leading participants of the Kaggle competition submitted multiple hundred attempts and re-trained their models according to the results they achieved.

7. Acknowledgements

As explained above, we decided in the course of the project to rely on a pre-cropped dataset. We are therefore grateful to the Kaggle User Phalanx for providing it. In addition, we thank Kaggle User Adnan Pen for inspiring us to use *Tracer* for the Image Segmentation part.

8. Contributions

In general, we were very happy concerning collaboration, division of tasks and the learning aspect for all of us. Since our project included multiple phases, we decided to split into 2 teams of 2 members to parallelize tasks. Victor and Dinah took over the conventional ML approaches. They discussed conceptualisation of methodology and code together and then both contributed to the implementation of the code and the writing of the reports. Maren and Reed focused on the DL approach. They took over image segmentation and the Deep Learning models, one for individual ID prediction and the other one for species. Reed as the more experienced coder took care of the conceptual side and building the models. The writing of the report was done by all four team members, with both sub-teams leading the writing of their respective sections. The literature review and the writing of other fundamental chapters was also done by all members.

References

[1] K. S. Collins, S. M. Edie, G. Hunt, K. Roy, and D. Jablonski. Extinction risk in extant marine species integrating palaeontological and biodistributional data. *Proceedings: Biological Sciences*, 285(1887):1–8, 2018.

[2] P. Dhar and S. Guha. Fish image classification by xgboost based on gist and glcm features. *International Journal Information Technology and Computer Science*, 4:17–23, 2021.

[3] A. Faaeq, H. Gürüler, and M. Peker. Image classification using manifold learning based non-linear dimensionality reduction. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. Ieee, 2018.

[4] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2019.

[5] S. Ghosh, N. Das, I. Das, and U. Maulik. Understanding deep learning techniques for image segmentation. *ACM Comput. Surv.*, 52(4), aug 2019.

[6] N. Horning et al. Random forests: An algorithm for image classification and generation of continuous fields data sets. In *Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, Osaka, Japan*, volume 911, 2010.

[7] H. C. Huang, J. Joseph, M. J. Huang, and T. Margolina. Automated detection and identification of blue and fin whale foraging calls by combining pattern recognition and machine learning techniques. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–7. IEEE, 2016.

[8] B. Hughes and T. Burghardt. Automated visual fin identification of individual great white sharks. *International Journal of Computer Vision*, 122(3):542–557, 2017.

[9] J. B. C. Jackson. The future of the oceans past. *Philosophical Transactions: Biological Sciences*, 365(1558):3765–3778, 2010.

[10] M. S. Lee, W. Shin, and S. W. Han. Tracer: Extreme attention guided salient object tracing network. *arXiv preprint arXiv:2112.07380*, 2021.

[11] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[12] S. Raschka. What is softmax regression and how is it related to logistic regression. `https://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html`, 2016. [Online; accessed 09-April-2022].

[13] V. Renò, G. Gala, P. Dibari, R. Carlucci, C. Fanizza, G. Castellano, G. Vessio, G. Dimauro, and R. Maglietta. Innovative classification of dolphins using deep neural networks and grabcut. 11 2020.

Appendix A: Occurrence of species in the dataset

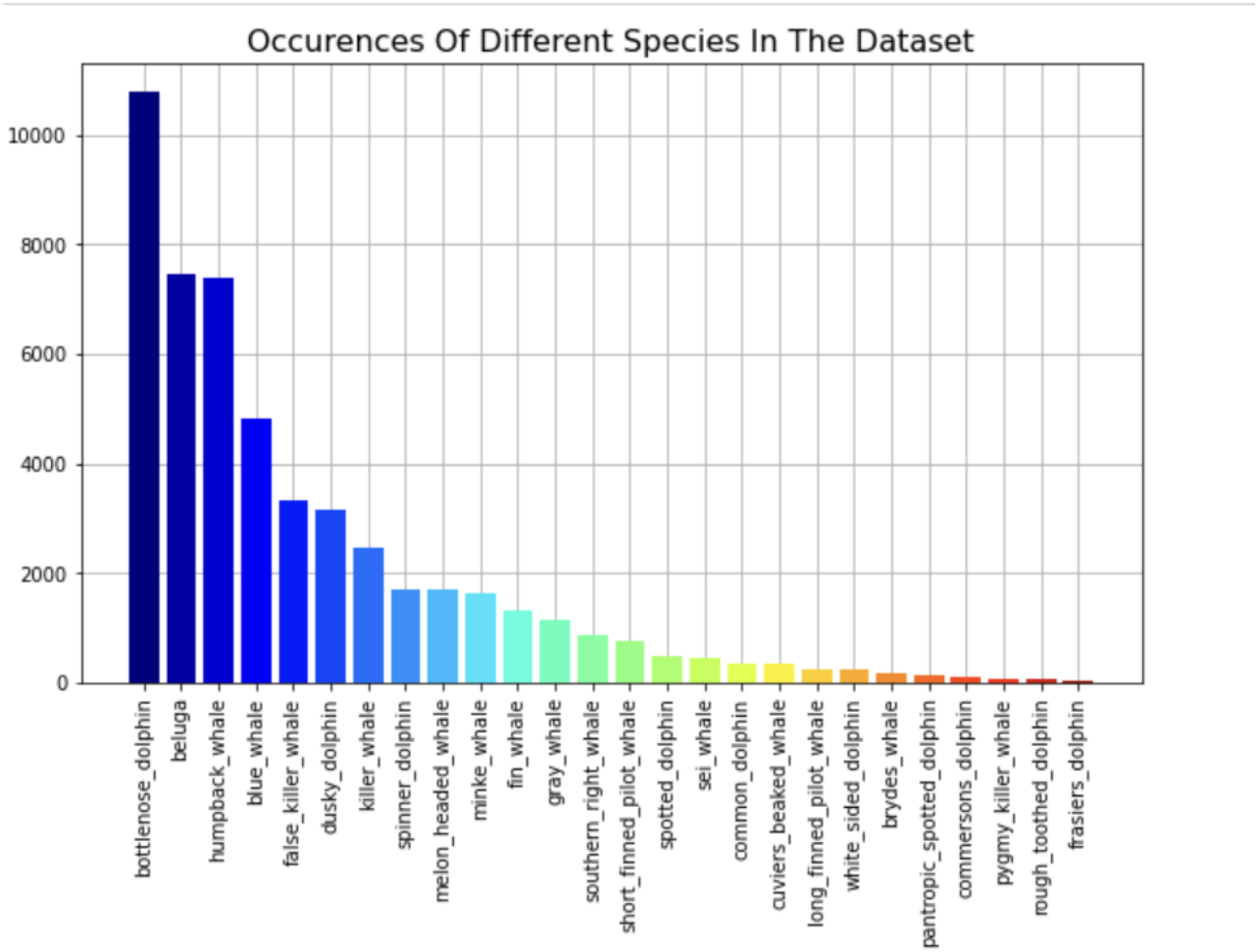

Figure 7. Occurrences of different species in the data set. Source: Own illustration

Appendix B: Confusion Matrices of ML models

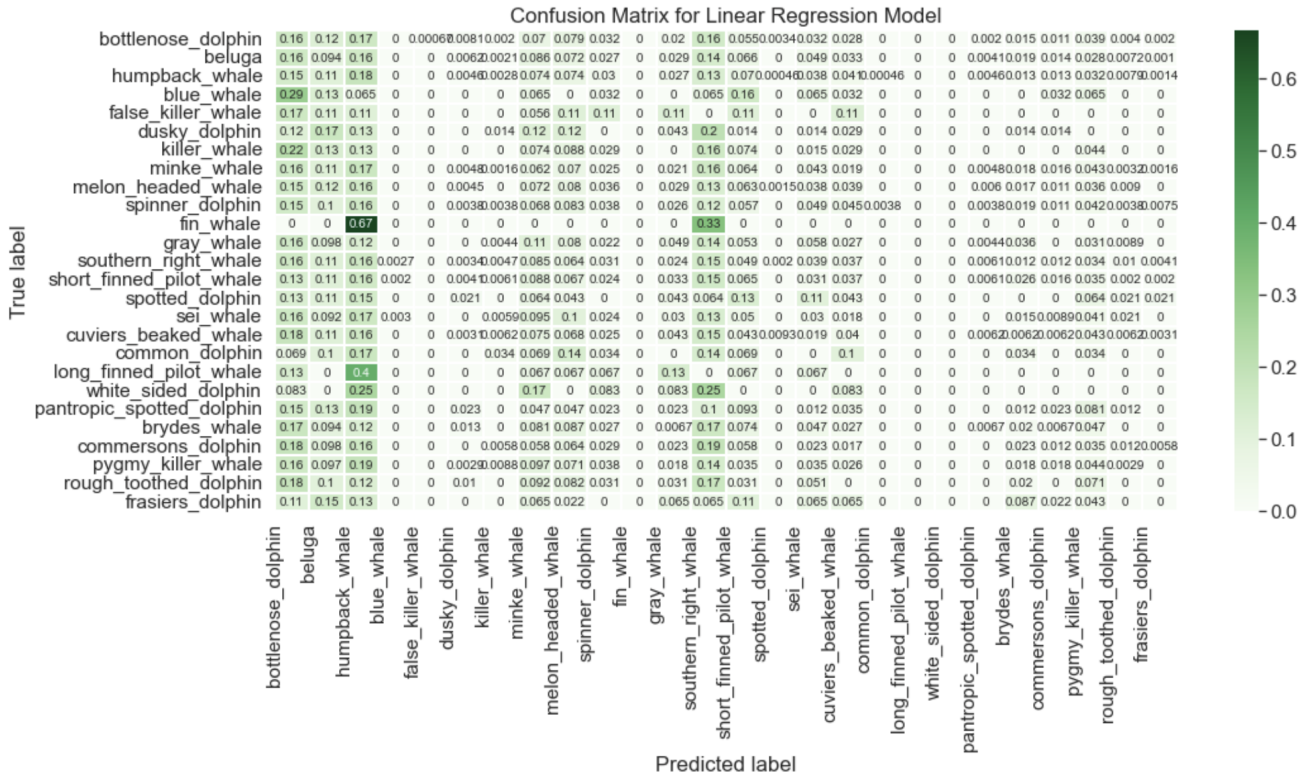

Figure 8. Confusion Matrix Baseline Softmax Classifier

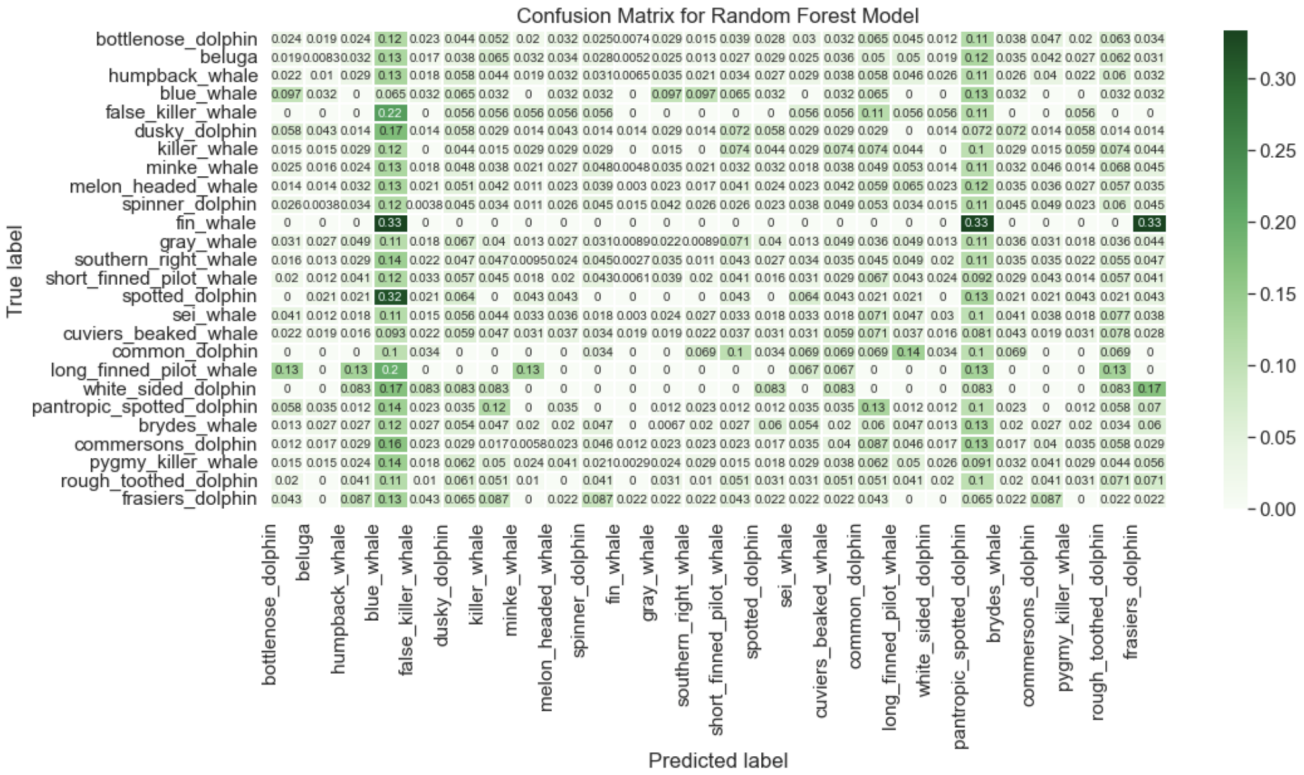

Figure 9. Confusion Matrix Tuned Random Forest Classifier

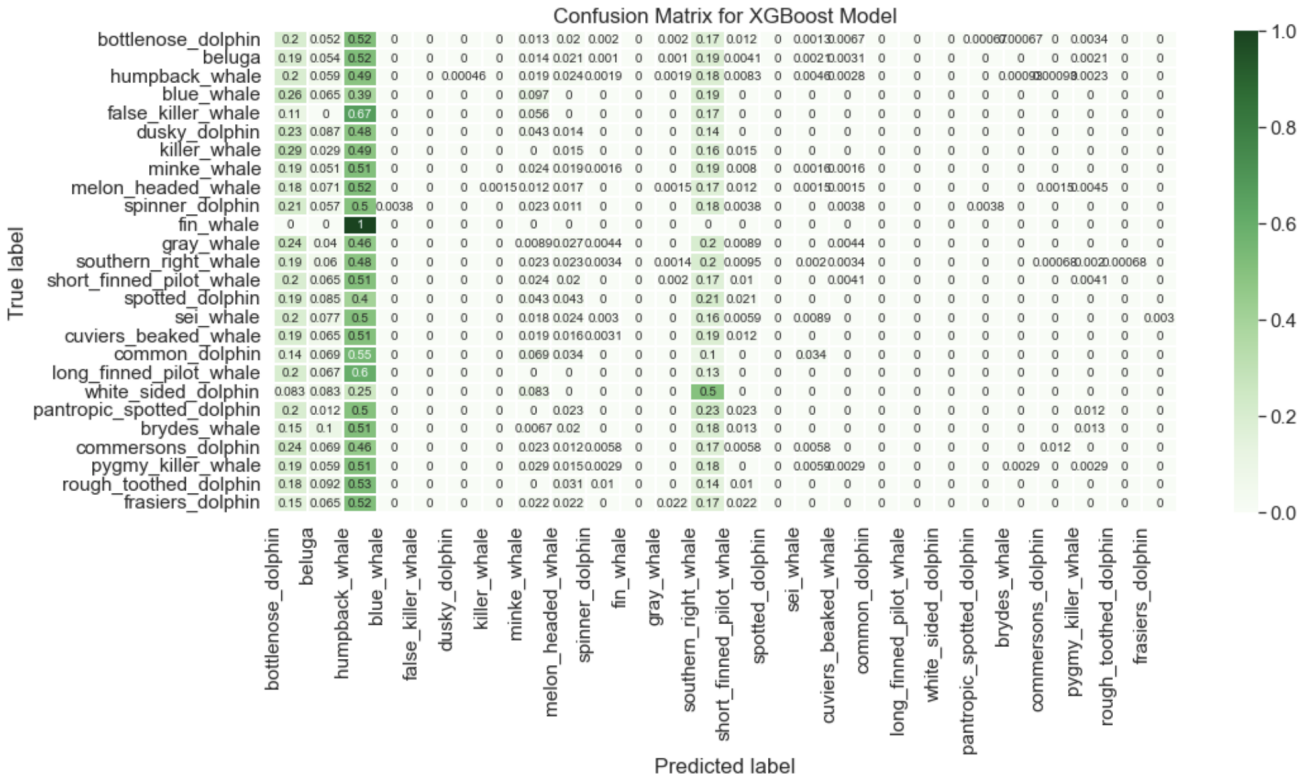

Figure 10. Confusion Matrix Tuned XGBoost Classifier

Appendix C: Kaggle submission scores

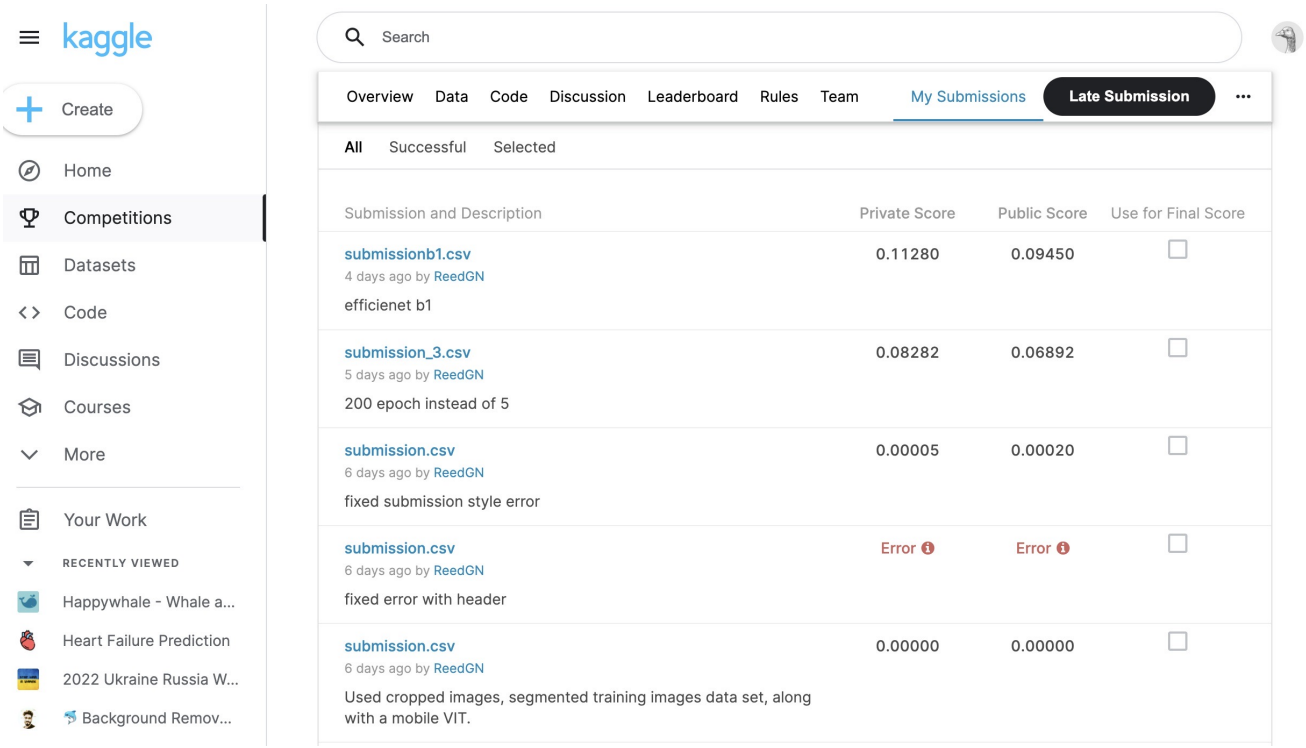

Figure 11. Overview of our Kaggle submissions

Note: the private score measures the accuracy of our model given its performance on the 25% of the test data we could access, while the public score shows the model performance on the full test data set.