Identification of Whales and Dolphins from Images with limited data

Name: Nitish Krishna Sadhu
Student No: 220283937
Supervisor: DR. Stefan Poslad
Programme of study: MSc Big Data Science

***Abstract*—This project report presents the study and implementation of neural networks and deep learning in the identification of whales and dolphins from images with limited data. The study addresses the challenge of species classification and identification of the individual_IDs from limited data. Employing image recognition techniques and deep learning methodology involves data preprocessing, feature extraction, neural networks, and model fine tuning. The aim of this project is to achieve high validation accuracy(>80%) in identifying the species and individual_IDs of the animals. The project achieved the high validation accuracy while identifying the species but could not achieve the same when identifying the individual_IDs of the animals. This project showcases the challenges of identifying the labels when the data available is limited.**

I. INTRODUCTION

The process of human identification has long relied upon the uniqueness of fingerprints and the precision of facial recognition. In a similar manner, researchers in the field of marine biology employ analogous techniques to identify and monitor marine creatures, which are whales and dolphins in this context. These marine beings are distinguished and catalogued based on the distinctive markings that grace their tails, heads, dorsal fins, and various other intricate parts of their body. This practice, is known as photo identification (photo ID), stands as an important tool within the realm of marine mammal science.

The significance of photo ID resides in its profound capability to facilitate the tracking of marine organisms, specifically the lives of whales and dolphins in this context. This innovative methodology empowers researchers to not only document the migratory patterns and behavioural nuances of these creatures across temporal spans but also furnishes a robust framework for assessing the dynamics of their populations. This knowledge helps us understand how healthy these species are and figure out the complex patterns that affect their survival in the ever-changing ocean environment.

The aim of this project is to use deep learning and automating the identification of the species and individual_IDs of the animals which could make the identification process way quicker. This will have a significant impact in the field of marine biology as Not only would it increase the pace of research, but it would also open avenues for researchers to engage in methodologies that were previously hindered by financial constraints or technical limitations.

As it stands, the predominant approach in research institutions revolves around the meticulous and process of manual matching, which is prone to misidentification, conducted by the researchers. Countless hours are invested in scrutinizing photographs, endeavouring to discern matches and identify novel individuals amidst the vast ocean of images. While researchers undoubtedly find joy in immersing themselves in the captivating imagery of whales, the method of manual matching inadvertently imposes constraints on the scope and goals of their inquiries.

The challenge of this project comes from the available data of the whales and dolphins being limited as collecting the photographs of whales and dolphins requires the knowledge of habitats of the animals and waiting for the animals to surface for respiration to capture the photos of their body parts.

Fig. 1. Photo ID of species “bottle_nose_dolphin” with an individual_ID “a22f9b1c9bc5”.

II. PROBLEM STATEMENT

In this project, the first part aims at identifying the species of the animal from the images with a high validation accuracy of above 80 percent. The second part of this project aims at identifying the individual_IDs of the animals, for this purpose, the project aims at designing a custom-made model and compare it with the second model which utilizes pre-trained *ResNet18* model and compares the performance of both the models as utilizing pre-trained models via transfer learning is the current norm for the problems where the data available is limited.

III. LITERATURE REVIEW

The data available for the current project is limited. The industrial standard in the context of limited availability of data is to use pre-trained models using transfer learning. This project utilizes the pretrained *ResNet18* model through transfer learning to compare with a custom model which is based on *AlexNet.*

A. Transfer Learning:

When the data available is plenty, it makes sense to design and train a model from scratch. But when the data available is limited, training a model from scratch might not yield reasonable results. This is when transfer learning comes into play.

Currently, the research on transfer learning is generalized with a broad spectrum of areas in deep learning including the context of deep learning for image classification.

The deep transfer learning into 17 categories, they review the approach taken in each category (Zhang et al, 2020).

Deep Learning and Transfer Learning Approaches for Image Classification (Sajja and Kalluri, 2019) includes a few paragraphs regarding transfer learning and some image classification results uses transfer learning.

The Deep transfer learning for image classification: a survey (Plested and Gedeon, 2022) suggests different strategies to use transfer learning based on the type of data.

In the context of this project, in the first part of this project where the species labels are needed to be predicted, the data is comparatively large and the data is similar, in this context, The Deep transfer learning for image classification: a survey(Plested and Gedeon, 2022) recommends using a lower learning rate and momentum and less regularization is required as the dataset is comparatively large. They also suggest that any technique works well in this scenario. So, a custom-made model has been used for this part of the project.

In the second part of this project, where the individual_ID of the animal is needed to be predicted, the number of labels needed to be predicted are higher and the data available per label and the overall data are very low. The images are also very similar to each other i.e., the data is small and similar, in this case, The Deep transfer learning for image classification: a survey(Plested and Gedeon, 2022) recommends using transfer learning as transfer learning excels in these scenarios. They also suggest using low learning rate and momentum as the need not be moved far from their pre trained values.

B. Data Augmentation:

Data augmentation refers to methods for constructing iterative optimization or sampling algorithms via the introduction of unobserved data or latent variables (Van Dyk & Meng, 2001).

The current data augmentation methods are of two types traditional, white box or black box methods. Currently the most popular techniques of data augmentation are traditional methods such as rotation, reflection, scaling, and shearing. These geometric transformations are widely used to increase the effective size of the dataset(Kwasigroch et al, 2018).

The geometric methods of data augmentation like rotation outperform the photometric methods of data augmentation(Taylor and Nitschke, 2017).) Improving deep learning using generic data augmentation (Taylor and Nitschke, 2017) suggests that the data augmentation technique of cropping improves the performance by greater extent.

In the context of this project, where the data is limited, the geometric data augmentation methods recommended by Improving deep learning using generic data augmentation (Taylor and Nitschke, 2017) including the cropping technique have been used along with the photometric techniques like colour jittering.

C. Resnet18 vs AlexNet:

In the current project, the second part involves the identification of the individual_ID of the animal and the aim of the project is to build a custom model based on *AlexNet* and compare it with the model that uses *ResNet18* through transfer learning.

Some images in the data being used in this project is similar and some are not. Comparative analysis of Alexnet, Resnet18 and squeezenet with diverse modification and arduous implementation (Ullah et al ,2021) showed that both *ResNet18* and *AlexNet* perform very well when the differences in the data

are minimal. It has also been suggested that *ResNet18* takes the least amount of time to go through the dataset as it has less parameters.

AlexNet also outperforms *ResNet18* in terms of f1 score and precision where the individual data points in a dataset are similar(Ullah et al ,2021)

The study Comparative analysis of Alexnet, Resnet18 and squeezenet with diverse modification and arduous implementation (Ullah et al ,2021) similar to the context of this project in the context of the datapoints in the dataset being similar.

As *AlexNet* performed better that *ResNet18* as shown by Comparative analysis of Alexnet, Resnet18 and squeezenet with diverse modification and arduous implementation(Ullah et al ,2021), *AlexNet* with some modifications for the context of this project has been used to train from scratch and the *ResNet18* has been used via transfer learning and has been trained to fine tune the weights.

IV. METHODOLOGY

A. Data:

The data set contains the images shapes, features and markings (some natural, some acquired) of dorsal fins, backs, heads, and flanks of different whales and dolphins. Some species and some individuals have highly distinct features, others are very much less distinct. Further, individual features may change over time. The data contains images of more than 15,000 unique individual marine mammals from 30 different species collected from 28 different research organizations. Individuals have been manually identified and given an *individual_id* by marine researchers.

i. Training Data:

The training dataset comprises a total of 51,034 labelled images depicting whales and dolphins. These images vary in dimensions. Predominantly, the images are RGB images, having three input channels. However, there are a few outliers that are not RGB and does not have three input channels. These outliers will be dealt with during the pre-processing phase, ensuring the consistency of the data before training the model.

In the training data, there are a total of 30 species labels, the following plot shows the top 7 most occurring species labels.

Species	Count
bottlenose_dolphin	9664
beluga	7443
humpback_whale	7392
blue_whale	4830
false_killer_whale	3326
dusky_dolphin	3139
spinner_dolphin	1700
melon_headed_whale	1689
minke_whale	1608
killer_whale	1493
fin_whale	1324
gray_whale	1123
bottlenose_dolpin	1117

Table 1: The table shows the all the species labels with more than 1000 images associated with them.

In the context of individuals within the dataset, there are a combined count of 15,587 distinct animals, corresponding to a total of 51,033 images. However, the distribution of images among individual animals is not uniform. The following plot illustrates the most prominent seven individual animals, each having the highest quantity of photographs within the dataset.

individual_id	count	species
37c7aba965a5	400	minke_whale
114207cab555	168	minke_whale
a6e325d8e924	155	bottlenose_dolphin
19fbb960f07d	154	minke_whale
c995c043c353	153	bottlenose_dolphin

Table 2: Table showing the top 5 individual_IDs with the greatest number of images and the species of the animal.

ii. Training CSV:

The training data is accompanied by a CSV file, which consists of the names of the images and the labels namely the species of the animal and the individual ID of the animal.

iii. Representing the Photo_ID as three channels:

An image is represented by its Photo_ID and can be seen as a two-dimensional matrix of pixels with defined height and width. Pixels are singular or have three parts (in colour images) of values, forming a three-dimensional matrix or "tensor." The pixel values range from 0.0 to 1.0 in PyTorch, denoting absence (0.0) to maximum intensity (1.0) of colours like RGB (red, green, blue). For example, (1, 0, 0) means full red, (0, 0, 0) signifies black, and (1, 1, 1) denotes white. In other frameworks, values might range from 0 to 255, indicating colour intensity.

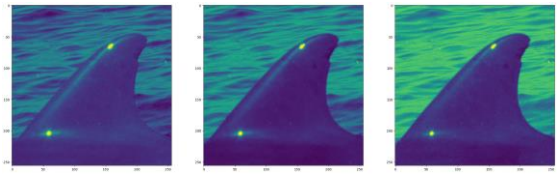

Fig 2. Representing an image as three channels.

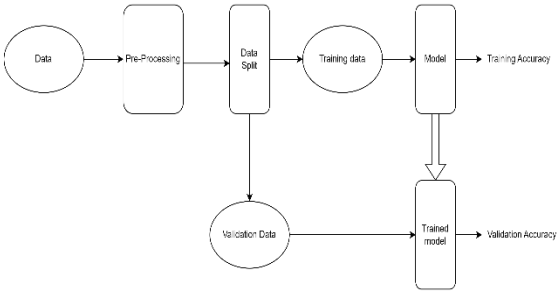

Fig 2: The above schema shows the flow of data.

B. *Preprocessing:*

As part of preprocessing, an extra column was added to the training CSV file containing image file pathways. This aids efficient image retrieval during preprocessing and seamless access during model training.

Regarding image data, most have three RGB colour channels, but some have different counts. To address this, a script was written which goes through the entire dataset, scanning images to exclude those with non-standard channel counts. Since these images were a small portion, excluding them didn't significantly affect dataset composition or quality. The modified CSV had 51,033 images before and 49,096 images after exclusion. The updated CSV was saved in my Google Drive repository.

i. Preparation of the Dataset to train

In the context of the first phase of this project, focused on species prediction for animals, the MyDataset class yields outputs in the form of (label, image). Here, the label denotes the animal's species, and the image is read from the dataset repository. It is essential to emphasize that the image is subject to the prescribed transformations before being presented for further processing.

A customised dataset class called "MyDataset" is created and inherited from the Dataset class in the *torch.utils.data* module as the first step in preparing a dataset for training. A DataFrame and a collection of transformations are both acceptable inputs for the dataset's constructor. The DataFrame specifically relates to the training data, which includes data about images. The changes, meanwhile, consist of a series of actions taken against the visuals.

The MyDataset class produces outputs in the form of (label, image) in the first phase of this project, which focuses on animal species prediction. Here, the image is taken from the dataset repository and the label identifies the species of the animal. The fact that the image is subject to the recommended alterations before being presented must be emphasised.

ii. Data Transformation:

The dataset available to train the models is limited and training with it might not result in the reasonable results in predicting the species and the identification of the individual_ID of the animal. So, transformations using the PyTorch's *transforms* module have been applied on the dataset to increase the effective size of the dataset during the preprocessing stage.

The following transformations have been applied on the images in the training dataset:

a. transforms.ToPILImage():

This transformation is applied to the input image which is in the tensor form. This transformation converts the image from a tensor form to PIL image.

b. transforms.RandomHorizontalFlip():

This transform is applied on the PIL images. It randomly selects a few images from the dataset and flips them in a horizontal orientation.

c. transforms.RandomVerticalFlip():

This transform is applied on the PIL images from the previous transformation, it randomly selects a few images from the dataset and flips them in a vertical orientation.

d. transforms.RandomRotation(10):

This transformation rotates the images by 10 degrees in clockwise direction or an anticlockwise direction. The direction of rotation is chosen randomly.

e. transforms.ColorJitter(...):

This transformation when applied to a PIL image introduces controlled colour variations. It randomly adjusts the brightness, hue, contrast, and saturation of the image. (brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1) these are the values given to the transformation in the context of this project.

f. transforms.RandomResizedCrop():

This transformation when applied to a PIL image will randomly crop the image and resizes the cropped image to the specified size. In the context of this project, the image is cropped within the scale (0.8, 1.0) and resizes them to the size of 256x256.

g. transforms.ToTensor():

After converting the images from the tensor form to PIL image and applying all the transformations specified above, the PIL image is finally converted back to the tensor form which is suitable for training the CNN model with.

h. transforms.Normalize():

After converting the image back to tensor form, the values in the tensor are normalized by applying the transformation *transforms.Normalize()*. This transformation subtracts the means (0.5, 0.5, 0.5) from the values and divides them by the standard deviations (0.5, 0.5, 0.5). Both the mean and standard deviations have three values as there are three channels (Red, Green, Blue) in the image and the calculations with the mean and standard deviation are applied across all the three channels.

iii. Splitting of the dataset:

For the evaluation of the CNN models, a dataset which contains the data the models have never seen is required. This dataset is known as the validation dataset. To obtain this dataset, the given training data is split into two parts. One part is used for the training of the models and is called as the *train_dataset* and the other part will not be shown to the models until the training using the *train_dataset* is complete, this part is called the *val_dataset* and is used to evaluate the models.

The split of the training dataset is carried out in such a way that the *train_dataset* contains 70 percent of the data whereas the *val_dataset* has the other 30 percent of the data. This split of the training dataset is used in training and evaluating the model in the first part of this project where the model predicts the species of the animal from the images. This 70 percent – 30 percent split has been chosen as it is shown that 70 – 80 percent data needs to be used for training and 30 – 20 percent of the data needs to be used for validation for the best results(Joseph, 2022).

For the second part of the project, a split of 80 percent data for the *train_dataset* and 20 percent for the *val_dataset* has been chosen as the available data per animal is limited and the model requires as much data as possible for training to achieve reasonable results.

The data was split in the above-mentioned ratios by using the *torch.utils.data.random_split* which randomly splits the data into the sizes mentioned.

C. MODELS DESIGN:

i. Part – 1:

The project's initial phase predicts animal species from images. The dataset has 30 labels, with 49,096 images post-preprocessing. The chosen framework is PyTorch. The model architecture includes four convolutional layers, each with batch normalization, ReLU activation, and max pooling. Convolutional layers increase complexity: 1st has 3 inputs and 16 outputs, 2nd has 16 inputs and 32 outputs, 3rd has 32 inputs and 64 outputs, and 4th has 64 inputs and 128 outputs. All use 3x3 kernels, stride 1, and padding 1. Two fully connected layers follow with a ReLU in between. Initial layer has 2048 inputs and 80 outputs, next has 80 inputs and 30 outputs. A kernel size of 3, stride 1, and padding 1 maintains dimensions. After convolution, batch normalization, ReLU, and max pooling, the image tensor flattens from 2D to 1D. This 1D tensor moves through fully connected layers for final prediction.

ii. Part – 2:

The second part of the project involves the prediction of the individual_ID of the animal. There are a total of 15587 unique individual_IDs. But not all the individual_IDs have the same number of images associated with them. The following plot demonstrates the inequality in the data.

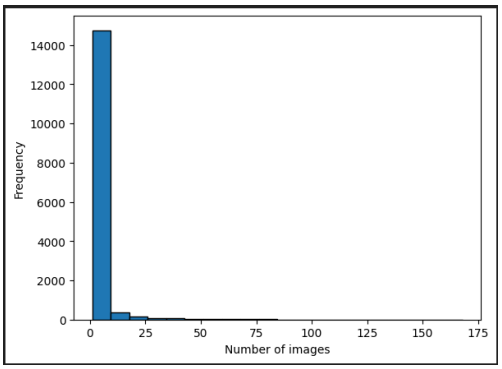

Fig 3: Plot depicting the number of individual_IDs within the range of the given bins.

The plot analysis reveals that many individual IDs are linked to fewer than 25 images, specifically 15,277 IDs. To enhance accuracy, an approach excludes IDs with under 50 images. The model focuses on IDs with over 50 images.

For prediction, two models are used: a custom "dolphinNet" based on AlexNet, and a ResNet18 through transfer learning.

The dolphinNet architecture has two parts: feature extraction and classification. Five convolutions with batch normalization and ReLU form the former, and three fully connected layers with dropout for regularization and interleaved batch normalization and ReLU comprise the latter.

In feature extraction, convolutions increase complexity: 3 to 64, 64 to 128, 128 to 256, 256 to 356, and 356 to 416 output channels. Kernels and padding vary, with the first layer at 11 kernel, 4

stride, and 2 padding, the second at 5 kernel and 2 padding, and others at 3 kernel and 1 padding. Max-pooling layers use 3 kernel and 2 stride.

This architecture transforms data into enriched features for classification.

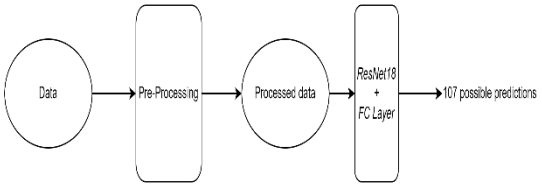

Fig 4: Diagram representing the structure of the second model of part-2

The subsequent component of part 2 involves the integration of transfer learning methodologies. Herein, the renowned pre-trained model, ResNet18, accessible within the *torchvision.models* module, is harnessed. Analogous to the prior phase, only individual IDs associated with over 50 images are considered during training, ensuring a focused and robust training dataset. The ResNet18 model is procured from the designated PyTorch model repository. Extending the downloaded ResNet18 model, a fully connected layer is incorporated. This addition tailors the overall model architecture to be suitable in the context of this project, by enabling the model to predict the 107 individual_IDs.

D. TRAINING:

i. Part- 1:

In the initial part (Part 1) of the project, an iterative approach was taken to determine the optimal batch size. Batch sizes of 16, 32, 64, 128, and 256 were evaluated, with 32 proving to be the best compromise between accuracy and training efficiency.

The training process extended over 15 epochs, chosen after observing a plateau in accuracy beyond this point.

Moving to Part 2, the focus shifts to training two distinct models. Batch size optimization involved testing batch sizes of 16, 32, and 64. Ultimately, a batch size of 32 was selected, offering a favourable balance between model accuracy and training speed for both models.

The dolphinNet architecture model underwent meticulous training over 10 epochs, as accuracy gains plateaued beyond this point. Similarly, the ResNet18-based model, with an added fully connected layer, underwent a truncated training of 3 epochs, given the saturation of accuracy improvements in this short training interval.

E. EVALUATION:

Across both segments of this endeavour, model evaluation was predicated upon two pivotal metrics: training accuracy and validation accuracy. A comprehensive assessment of training accuracy was undertaken following the completion of each training epoch, thereby facilitating the understanding of the models' learning dynamics.

In the first part of this project, a partitioning approach allocated 30 percent of the original dataset for validation purposes. Conversely, for the latter phase, a discerningly selected subset amounting to 20 percent of the initial dataset was designated for validation assessment. This stratified division was informed by meticulous experimentation, revealing that models attain optimal outcomes when trained on 70-80 percent of the data and subsequently validated against the backdrop of the remaining 20-30 percent. Such partitioning balances the amount of data necessary for training and validation of the models.

V. RESULTS

i. Part-1:

In the first part of the project, the model yielded a training accuracy of 85 percent and a validation accuracy of 83 percent.

The progression of loss during training is as follows:

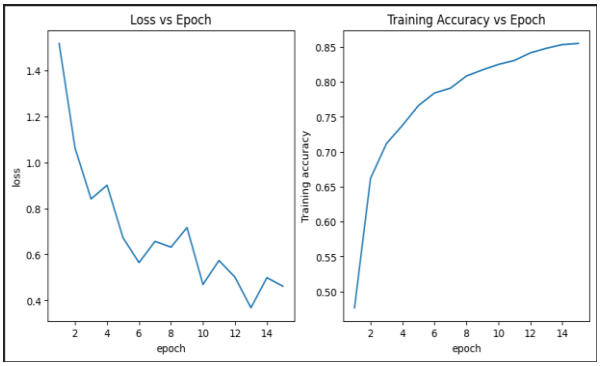

Fig 5: The progression of loss and training accuracy during training

It can be seen in the plot that the loss of the model drastically decreased until the 6th epoch after which it slowed down. After the 6th epoch, the progression of loss is in a jagged manner, but the loss has decreased till the 15th epoch.

The progression of accuracy during the training can be seen in the above plot. The accuracy has increased drastically up until the 6th epoch after which the improvement in accuracy slowed down and at the 14th epoch almost flatlined.

ii. Part-2:

The second part of the project has two models, the first model *dolphinNet* has yielded a training accuracy of 10 percent and a validation accuracy of 11 percent.

The progression of loss during the training is as follows:

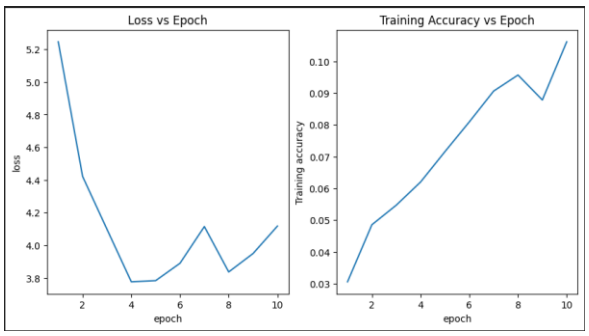

Fig 6: The progression of loss and training accuracy during training of *dolphinNet*

During the training of the model *dolphinNet,* the loss drastically reduced until the 4th epoch after which the loss progressed in a jagged manner with a peak at 7th epoch and it increased from 8th epoch to 10th epoch.

The training accuracy of the model *dolphinNet* has increased until the 8th epoch but had a slump at 9th epoch and again increased at 10th epoch.

The second model in part-2 which uses transfer learning to leverage the pre-trained *ResNet18* model, achieved a training accuracy of 9 percent over three epochs and a validation accuracy of 9.5 percent in the third epoch but achieved its maximum accuracy of 10 percent in the second epoch.

The progression of loss is shown by the following plot.

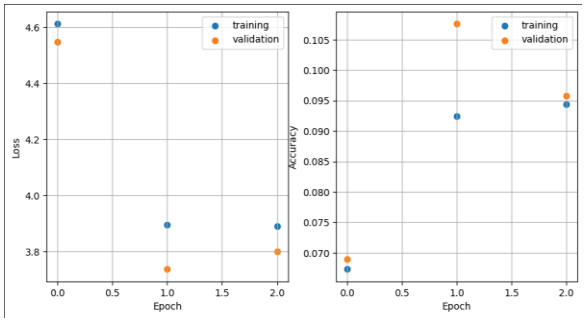

Fig 9: The progression of loss and validation accuracy.

The model has been trained only for three epochs as the model flatlines after 3 epochs. The progression of the training loss and validation loss has been shown in the above plot. Both the losses start of very high but comes down the next epoch and pretty much flat lines at the third epoch. Coming to the progression of training accuracy and validation accuracy. Both the accuracies increase from the 1st epoch to the 2nd epoch but decreases from the 2nd epoch to the 3rd epoch.

VI. DISCUSSION

This project has been greatly affected by the availability of the data. For the first part of the project the where the species of the animal needs to be predicted, the data available was sufficient to achieve reasonable results as evident by the results (training accuracy; 85 percent, validation accuracy: 83 percent).

The second part of the project where the individual_ID of the animal needs to be predicted has been greatly affected by the availability of the data as only 107 individuals_IDs has more than 50 images associated with them. After excluding the individual_IDs that has less than 50 images, the model showed a slight improvement from 1 percent accuracy to 10 percent accuracy.

The following plot shows the validation accuracies of the top 3 species with the greatest number of images associated with them and the bottom 3 species with least number of images associated with them.

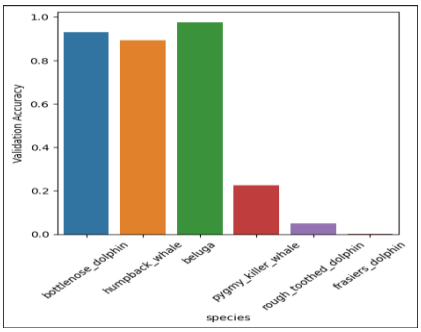

Fig 7: The Validation accuracy of the top 3 and the bottom 3 species labels by number of images.

The following plot shows the number of images associated with the top 3 and the bottom 3 species by number of images associated with them.

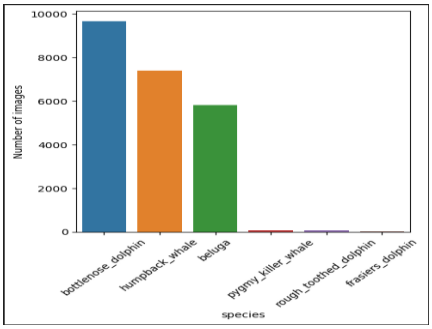

Fig 8: The Number of images associated with the top 3 and the bottom 3 species by number of images associated.

The following plot shows in clear the number of images associated with the bottom 3 species even more clearly.

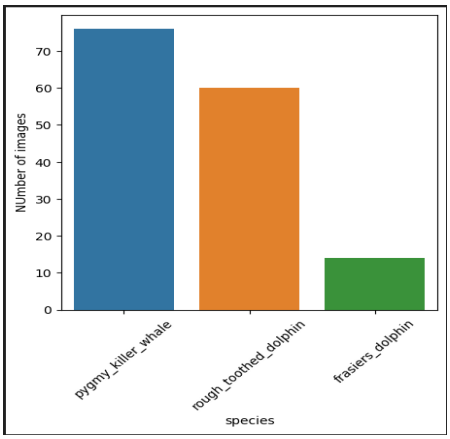

Fig 9: The number of images associated with the bottom 3 species based on the number of images associated.

Despite the model's high overall accuracy (training: 85%, validation: 83%), certain species labels with fewer images exhibit low validation accuracy. Addressing this involves increasing images per label. Data augmentation implementation greatly improves accuracy, enhancing validation from 0.1% to 10%. The impact is evident in top species labels with more images (e.g., bottlenose dolphin: 0.9305, humpback whale: 0.8933, beluga: 0.9747).

In Part 2, limited data significantly affects results (validation accuracy: 10%). Only 107 out of 15220 individual_IDs possess 50+ images. Training with all records yielded 0.1% accuracy; focusing on IDs with >50 images improved accuracy (training and validation: 10%), but further enhancement is needed for reliable predictions.

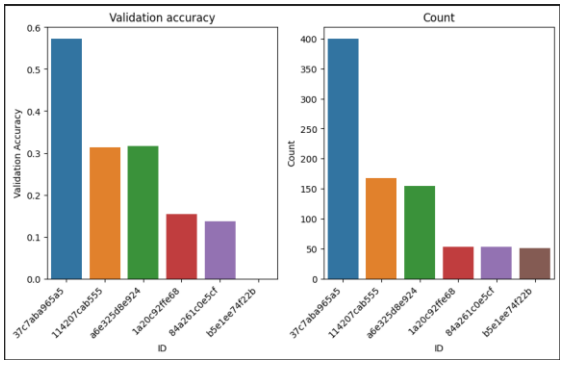

Fig 11: The Validation accuracy and the number of images of the top 3 and the bottom 3 species labels by number of images.

From the above plot it can be clearly seen how the availability of data affects the validation accuracy of the individual_IDs. For the IDs *37c7aba965a5, 114207cab555, a6e325d8e924*, the model achieved high accuracy 57.25, 31.37, 31.61 respectively as they are the top three IDs with the most images associated with them. Similarly, the IDs 1a20c92ffe68, 84a261c0e5cf, b5e1ee74f22b experience the lowest validation accuracies (0.1548, 0.1373, 0.000) as they have the least number of images associated with them among the data used for training the models.

It is the same case with the second model that uses *ResNet18* through transfer learning which is evident by the following plot.

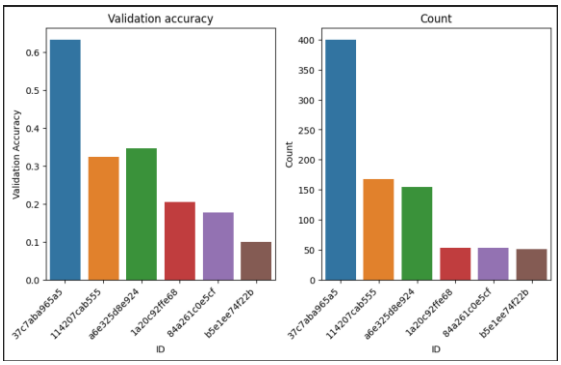

Fig 10: The Validation accuracy and the number of images of the top 3 and the bottom 3 individual_IDs labels by number of images.

VII. LIMITATIONS

This project has been severely limited by the unavailability of data. The first part of the project, even though achieved a high validation accuracy of 83 percent, was affected by the lack of enough data. The species labels with a greater number of images associated with them showed very high accuracy and the species labels with low number of images showed very low accuracy. This can lead to a problem when there is a need to predict the species labels where the image might belong to one of the species classes with lower number of images associated with them in the training data. Collecting more data can resolve this problem. The effect of lack of enough data in the second part of the project is even more drastic as there are more labels to predict.

The data of this project is stored in google drive and the code is run in google colab. Reading image data from the google drive takes a long time and the compute resources are not enough to train and validate more complex models.

It has been tried to mitigate the imbalances in data through SMOTE (Synthetic Minority Over-sampling Technique), reading the data from google drive was taking a very long time and the lack of compute resources and RAM kept crashing the script. Storing and training the model on a local device have been tried. Even though it improved the

reading of the data by the model, lack of a powerful GPU and a complete absence of TPU eliminated this option. With more computing power and much power data augmentation techniques and slightly more data, this project can be expanded to achieve reliable results.

REFERENCES:

David A van Dyk & Xiao-Li Meng (2001) The Art of Data Augmentation, Journal of Computational and Graphical Statistics, 10:1, 1-50,

Joseph, V.R. (2022) Optimal ratio for data splitting, arXiv.org. Available at: https://doi.org/10.48550/arXiv.2202.03326

Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," 2018 International Interdisciplinary PhD Workshop (IIPhDW), Świnoujście, Poland, 2018, pp. 117-122, doi: 10.1109/IIPHDW.2018.8388338.

Plested, J. and Gedeon, T. (2022) Deep Transfer Learning for Image Classification: A Survey.

Sajja, T.K. and Kalluri, H.K. (2019) Deep learning and transfer learning approaches for Image Classification.

Taylor, L. and Nitschke, G. (2017) Improving deep learning using generic data augmentation, arXiv.org. Available at: https://arxiv.org/abs/1708.06020 (Accessed: 24 August 2023).

Ullah, A. et al. (2021) - arabian journal for science and engineering.

Zhang, J. et al. (2020) Recent advances in transfer learning for cross-dataset visual recognition: A problem-oriented perspective: ACM computing surveys: Vol 52, no 1, ACM Computing Surveys.

APPENDIX

Fig A1: The Architecture of the ImageClassifier Model.

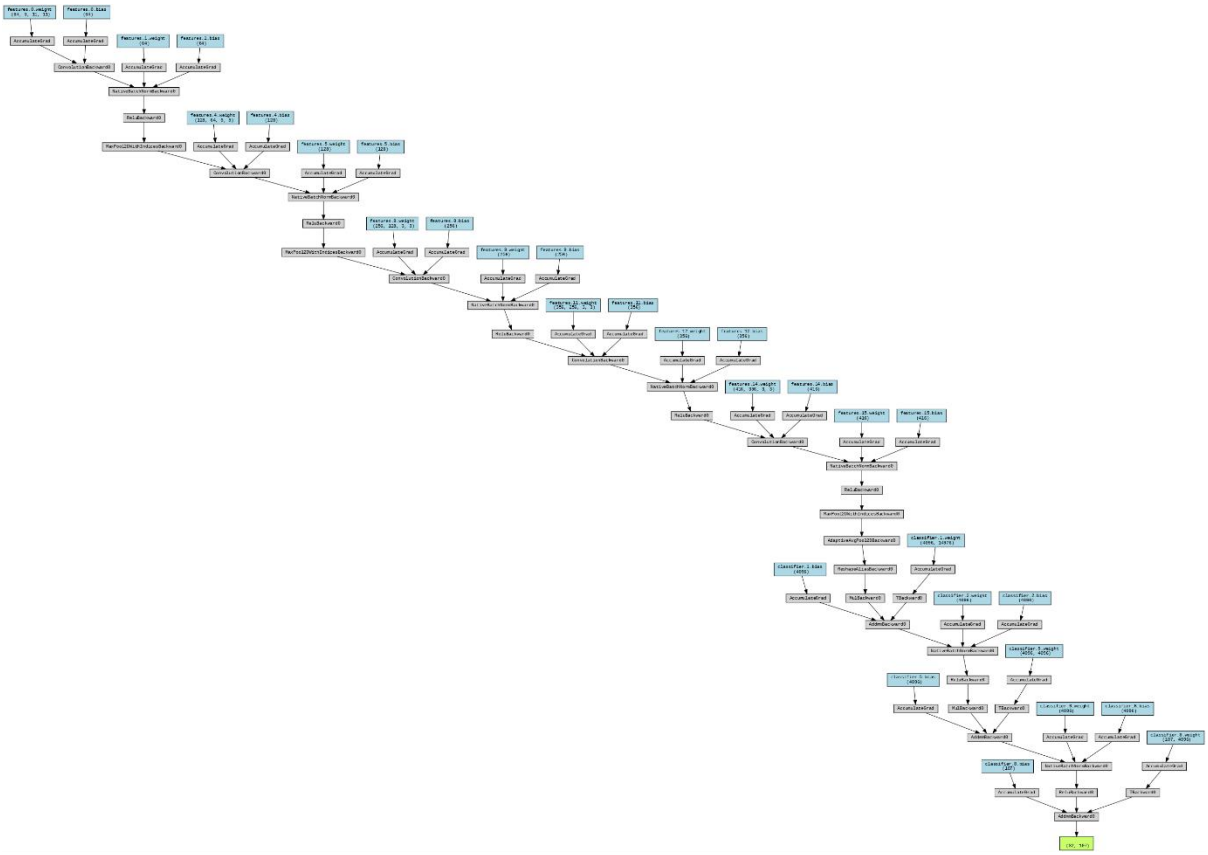

Fig A 2: The Architecture of the dolphinNet model

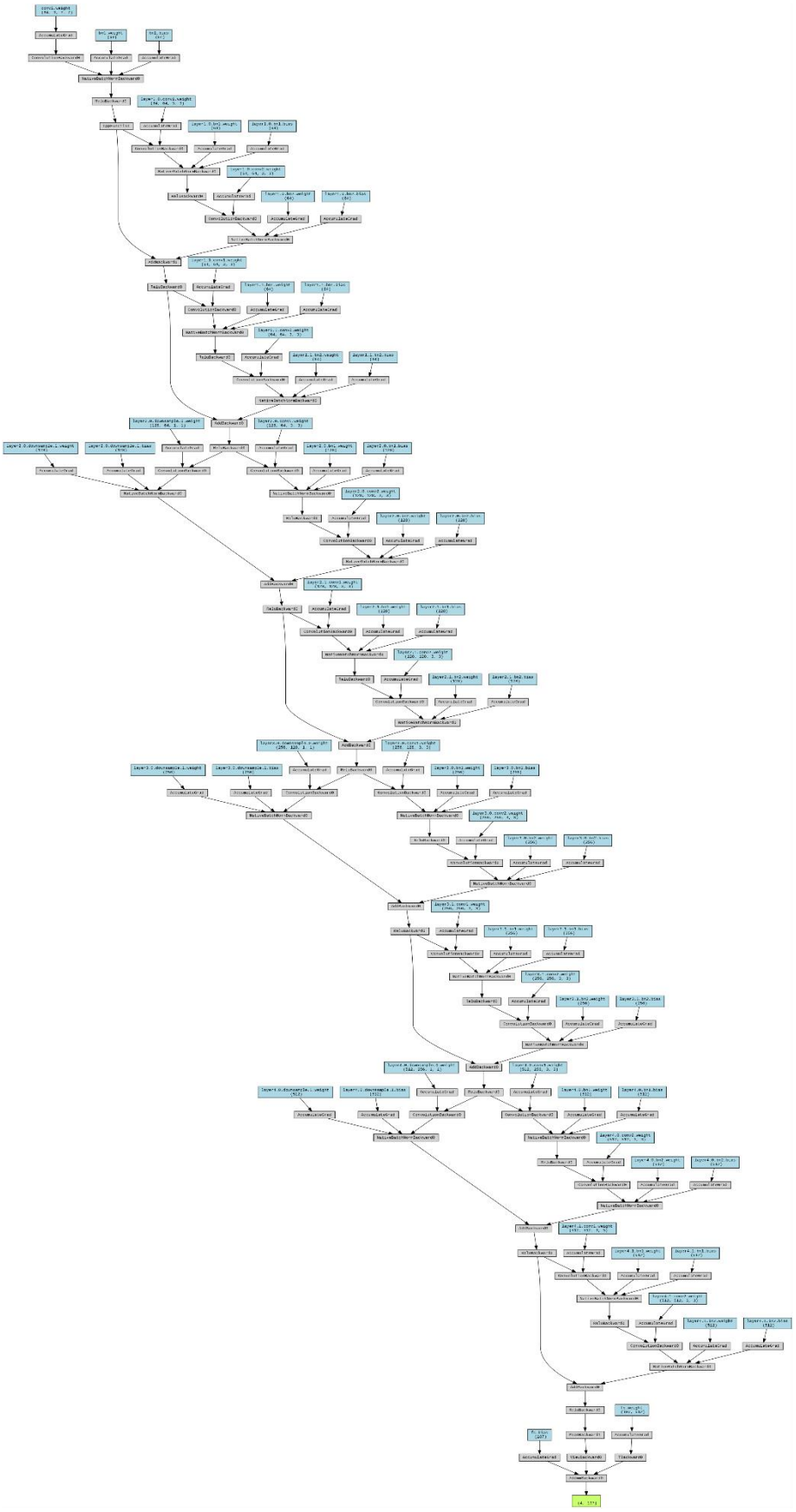

Fig A 3: The Architecture of the model where ResNet18 was used through transfer learning.

MSc Project - Reflective Essay

Project Title:	Identification of Whales and Dolphins from Images with limited data
Student Name:	Nitish Krishna Sadhu
Student Number:	220283937
Supervisor Name:	Dr. Stefan Poslad
Programme of Study:	MSc Big Data Science

Introduction:

The project involves the classification from images using deep learning. This project consists of two parts, part one involves the identification of the species label of the animal from images. The second part of the project involves the identification of individual_ID of the animal which is unique to the animal from the images of their tails and fins. This project aims at aiding marine researchers in identifying the marine animals in effective and time saving manner.

Approach:

The first part of the project where the identification of the species of the animal is required, a CNN model called *ImageClassasifer* has been designed and used which yielded reasonable results(Training accuracy: 85%, validation accuracy: 83%). The second part of the project is where the individual_ID of the animal needs to be identified; two models have been used. The first model named *dolphinNet* is based on AlexNet, the model has been altered to suit the context of this project. The second model uses ResNet18 through transfer learning. The results of both the models have been compared.

Strengths and Weaknesses:

A. Strengths:

- The *ImageClassifer* model shows reasonable and reliable results in identifying the species of the animals with a validation accuracy of 83 percent.
- The *ImageClassifer* model shows validation accuracies over 90 percent in identifying the species with a greater number of images associated with them.
- The *ImageClassifer* model also shows reasonable results for the species labels with lesser number of images associated with them.
    i. The *pygmy_killer_whale* species label has only 76 images associated with it but the *ImageClassifer* model showed a validation accuracy of 22 percent.
- The model *dolphinNet,* even when trained with a smaller number of images per individual_ID showed reasonable accuracies for the individual_ID labels with a greater number of images associated with them.
    i. The individual_ID *37c7aba965a5* even though has only 400 images associated with it, the model *dolphinNet* showed a validation accuracy of 57 percent.
    ii. The individual_ID *114207cab555* even though has only 400 images associated with it, the model *dolphinNet* showed a validation accuracy of 32 percent.

B. Weaknesses:

- The entire project has been greatly affected by the lack of enough data and imbalances in data.
- The data for this project has been stored in google drive and the project was developed in google colab, accessing the data from google drive is very slow and greatly increased the time required to train the models.
- The model *ImageClassifer*, even though exhibited a very high validation accuracy of 83 percent overall, it has been greatly affected by the imbalances in data. As a result, the model falls behind in identifying the species labels with lesser number of images associated with them. In the context of using the model in real world context, the model may struggle to identify rare species as the images data of such species is limited.
- The model *dolphinNet* is greatly affected by the lack of data and the greater number of labels to predict. Even after excluding the individual_ID labels with less than 50 images associated with them, the model exhibited a very low validation accuracy of 10 percent.
- The model where *ResNet18* is used through transfer learning, has been greatly affected by the lack of data. The training accuracy of the model stopped improving after the 3rd epoch which can be attributed to the lack of enough data.

Limitations and Challenges:

The size of data for this project is a total of 65GB. It was a challenge to download such a large data set to my local machine and upload it to Google Drive consuming a lot of time. As the free account of Google drive only provides 15GB of storage, additional storage has been purchased to store the data of the project. Initially, it was decided to store the data on local and build and train the models on the local machine, but this idea was dropped as my local machine does not have a powerful enough GPU to train the model in reasonable time.

The models were trained on google colab, initially the free version of google colab has been used but the time taken to train the model on the free GPUs was more than 2 hours, which is when I decided to purchase the pro version of google colab which provides TPUs to train. To train the model on the google colab TPUs required the knowledge of torch.xla module.

Coming to the data, being stored in Google Drive, the time required for the code to read the data from the google drive is high because of which the iterative approach of preprocessing the images using a *for* loop kept crashing after running for 30 minutes. This problem was rectified by splitting the train.csv dataframe into two parts and preprocessing separately, even after implementing this strategy of preprocessing, the code crashed a few times but was ultimately successful.

The step of training the models has taken the most time in this project. A custom dataset class has been defined to train the models with the data. Even though the dataset has been defined the transformations have been applied in a lazy evaluation manner i.e., all the transformations defined are applied at the time of training. The image is first read from google drive then the transformations are applied, and the transformed image goes to the model to train, this is the case for all the images. The bottleneck here is reading the images from the Google drive which is very slow and resulted in very high training times.

There are a total of three models to train and with the above-mentioned problems with reading the data, each epoch during training has taken somewhere around half and hour to train, and

the training process kept crashing sometimes because the model was unable to read the data from the Google Drive and sometimes the all the TPUs are busy, and the model cannot train any further. This problem has been mitigated by splitting the training process into smaller parts. The models have been trained for a few epochs and saved and reloaded to train further when required.

Coming to the performance of the models, the model *ImageClassifier* exhibited a validation accuracy of 83 percent, I tried to improve further but was unable to do that as making the model any deeper by adding more convolution layers resulted in the model crashing during training because of the lack of RAM. I tried a different approach by keeping the number of convolutional layers the same and but making them wide by adding mode perceptrons which also kept crashing during training due to the lack of enough RAM. The models have been trained on a limited and very imbalanced dataset because of which the models perform better for few labels and very bad for other labels. The problem of the limited data was mitigated using data augmentation techniques, but it was still not sufficient to improve the accuracy to reliable levels. I tried to use SMOTE(Synthetic Minority Over-sampling Technique) to reduce the imbalances in data which required separating the labels and inputs in to separate lists and the script kept crashing once again because reading the data from Google Drive is very slow.

Future Work:

The aim of this project is to classify images with models trained with limited data. This could be achieved by further expanding the models by increasing the number of convolution layers to learn more features, this is possible by increasing the available compute resources and RAM. The reliability and the accuracy of predictions of the models can also be improved by increasing the available data, even though the project aims at classifying the images with limited data, the data currently available is not enough to make accurate predictions. For the second part of this project where the individual_IDs needs to be identified, more than 15400 labels have been excluded as they have less than 50 images associated with them and greatly affected the performance of the models. This problem can be solved by increasing the available images to at least 100 and using data augmentation[1] to further increase the size of the dataset.

The scenario where the data is insufficient, and the data is very rare or there are issues with privacy transfer learning greatly improves performance[2]. Even though transfer learning has been used in this project, it did not improve the performance greatly as the data available is not sufficient. So, increasing the amount of data and implementing even more data augmentation techniques might improve the performance.

The biggest problem of the project is time taken to train the models, as reading the data from the drive is very time consuming. This can be improved by having a dedicated storage for the data which is physically connected to the compute resources or by using a high-speed connection so that the model can read the data fast. This speed of reading the data can also be improved by increasing the available RAM and mounting the entire data on to the RAM[3].

Personal Development:

Even though the project could not produce reliable results in the second part, my journey with the project has been greatly beneficial for my personal development. My technical skills have greatly improved through research and implementation of the project. I have used pytorch framework for deep learning in this project, even though this module has been taught in our

classroom, using it for this project has improved my knowledge and expertise with the module, for improving the training time, I have also gained experience in the skills that have not been taught in classroom like using torch.xla module to use TPUs for training. Apart from the technical skills, I have also gained experience in some non-technical skills like dealing with the limited availability of resources by planning and completing the project with the limited availability of the resources. Time management is one more skill which I gained, as I had to plan the project so that I can meet the deadline. Splitting the project into different parts and setting deadlines for each part has greatly improved my sense of organization and time management.

References:

1. David A van Dyk & Xiao-Li Meng (2001) The Art of Data Augmentation, Journal of Computational and Graphical Statistics, 10:1, 1-50,

2. Plested, J. and Gedeon, T. (2022) *Deep Transfer Learning for Image Classification: A Survey*
3. Markus Püschel, Peter A. Milder, and James C. Hoe. 2009. Permuting streaming data using RAMs. J. ACM 56, 2, Article 10 (April 2009), 34 pages. https://doi.org/10.1145/1502793.1502799