Leopard Seal Recognition and Identification Using Machine Learning

Carson Buntin, Nathan Hontz, Ryan Huntington, Juncheng Li, Bryce Robinson, Carsyn Smeda

1 Introduction

1.1 Overview

The Advanced Leopard Seal Recognition System (ALSRS) is a set of machine learning (ML) models, training weights, and API endpoints designed to process images of leopard seals (*Hydrurga leptonyx*) and provide identifications for the individual specimens in those images. The API was designed with the end goal of integration with Happywhale to speed their leopard seal identification process in the same way that they have implemented identification processes for various whale species.

The ALSRS works by running images through two ML models: the first designed to detect the presence of leopard seals in an image and provide a bounding box for each specimen, and the second designed to analyze the area within that bounding box for identifying markings that can be used to provide a specific ID for that specimen based on pre-existing IDs from Happywhale and our research team.

1.2 Motivation

The current method for identifying leopard seals from images involves a manual process where images are uploaded to Happywhale that experts review and can identify the seals visually. This process can be time consuming and tedious for experts who may have other responsibilities or priorities. In the case of whale identification, Happywhale has developed WhaleID, an image-processing machine learning algorithm to match photos of humpback and sperm whales with scientific collections based on their unique patterns and shapes on their tails [1]. The current method of identifying leopard seals manually can be automated by leveraging machine learning algorithms to identify a seal based on the unique spot patterns on its body, similarly to how Happywhale has automated the process of identifying whales based on images of whale tails. This automated process would increase the speed, and potentially the accuracy, of leopard seal identification.

1.3 Contributions

This paper details the use of two machine learning models: one for detection of leopard seals in an image, and another for identification of individual seals detected in an image. Region-based convolutional neural networks (R-CNNs), Mask R-CNN and Faster R-CNN, were both used for the detection model to allow a user to be able to toggle between the two based on performance. After research, training, and testing, it was found that Mask R-CNN had a 91% accuracy and Faster R-CNN had a 98.7% accuracy in detecting leopard seals in an image. Data-Efficient Image Transformer (DeiT), which is a vision transformer, was used for the detection model and had a 99% top-5 accuracy for identifying leopard seals in an image.

These two models are part of a larger pipeline that allows a user to upload one or more images to Happywhale, calls the detection model, preprocesses the images based on the where the selected detection model detected seals, calls the identification model on the preprocessed

images, and returns the model's predicted top five identities for the seals found in the image as well as the model's confidence for each of the identities.

2 Related Work

2.1 Existing Methods

This project was based on similar architecture from HappyWhale and from the proof-of-concept completed by the client's research. HappyWhale employs similar techniques on detection and identification on whales and their tail fins. The proof-of-concept provided to the team was completed using YOLO (You Only Look Once) and ViT (Vision Transformer).

Current leopard seal identification is completed by experts in the field by using tracking information and unique spot patterns found on the face, neck, and body.

2.2 Machine Learning Approaches

The proof-of-concept was completed on leopard seals using YOLO and ViT, which is currently the only research done into identification of leopard seals using machine learning.

3 System Architecture

3.1 Overview

The end-to-end pipeline is able to take image input in any of three different formats: S3 data, Base64 encoded images, and image URLs. Once images are provided on the frontend, they are sent to the detection model. By default, Faster R-CNN is the detection model that the images are run through, but the user is able to change the model to Mask R-CNN if they wish. The selection for which model the user wants to run the images through is sent to the backend, along with the image data in whichever format was provided. From there, the images are run through the appropriate detection model which returns a list of bounding boxes for each seal detected in each image. These bounding boxes are used to preprocess and crop the input images so that the detected leopard seal is the primary focus of the image. These cropped images are passed into the identification model which returns a list of the top 5 potential identities for each seal image. Finally, this list of identities is displayed on the frontend for the user to view.

3.2 Data Collection and Preprocessing

3.2.1 Data Collection for the Detection Model

Happywhale's collection of marine life images on their website was a primary source of data that was collected for training, testing, and validating the detection model. Images of leopard seals were collected, as well as images of other seal species and sea lions. The selection of images that were able to be used to train the model was limited by copyright permissions on the images, so only images with Creative Commons licensing were taken from Happywhale. Images of scenery, items, and animals that were not seals were also collected from Creative Commons websites and from the COCO dataset. In the detection model's dataset, there were 1000 total images, with 57% of them being images of leopard seals, 38% of them being seals of a different species or sea lions, and 5% of them being items or animals that were not seals at all.

3.2.2 Data Annotation for the Detection Model

The images collected for the detection model's dataset had to be annotated so the model would be able to recognize leopard seals in an image. For training, testing, and validating Faster R-CNN, the images were manually annotated with bounding boxes, indicating where leopard seals were in the images, if at all. The dataset was annotated similarly for training Mask R-CNN, except instead of bounding boxes, the images were annotated with masks. Any images that did not have leopard seals were annotated indicating they were not seals. The annotations for both the bounding boxes and the masks were done through the Roboflow website which converted the bounding box or mask information into numerical data that could be exported to specially formatted JSON or text files that could be fed into the detection models for training. The data was split with 70% of the images in the training set, 20% in the validation set, and 10% in the test set

3.2.3 Data Collection for the Identification Model

For the identification model, identified leopard seals with multiple images and their associated identities were needed for training, testing, and validating. Once again, Happywhale was a large source of the images for the dataset. From Happywhale, 53 identified seals were added to the dataset with anywhere from 3 to 42 images for each identity. Additionally, a dataset provided by the client was also incorporated into the dataset for the identification model. This provided dataset had 913 total images with 13 identities. The data was split with 70% of the images in the training set, 20% in the validation set, and 10% in the test set. When splitting the images among these three sets, it was required that each identity had at least one image in each of the three sets. Initially, there could have been potentially zero images in a set for a given identity, but with this requirement in place, the accuracy of the identification model increased from 68% to 89%.

The challenge was discovered where each set might not have the ability to have at least one image in the event that a specific identity might have limited representation within our dataset. To address this issue, a strategy was devised to offer more flexibility and to ensure the integrity of our testing and training datasets. The first option for a solution was that if the number of images for an identity fell below a specific threshold, all available images would be added to the training or testing set, depending on the specific requirements of the experiment. Alternatively, if a more balanced distribution was sought, a data augmentation technique was employed to create additional copies of existing images and add a random rotation up to 30 degrees. Then, the normal 70-20-10 split would be applied to the seal identity.

In preprocessing our identification dataset, various augmentation techniques were employed to mitigate the limitations posed by the relatively small number of images available. Additional augmentation strategies encompassed a range of transformations including rotation, color jittering, random perspective adjustments, resizing, and normalization. These techniques collectively aimed to enhance the robustness of the dataset, ensuring optimal performance and generalization capabilities of our identification model.

3.2.4 Data Preprocessing for the Identification Model

After collecting the images required for the identification model's dataset, they needed to be cropped in a similar way to how they would be in the pipeline. The images collected were cropped manually so that the leopard seals would be centered as the primary focus of the image. It was found that some of the images, after being cropped, retained a transparent background where the rest of the image was cropped out. Feeding images with transparent backgrounds to the identification model caused it to run into performance issues. To combat this, a python script was written to remove any transparency in the images before running them through the model. Additionally, another python script was created to resize the images to 600 by 600 pixels, and convert all images to png as some of the images were quite large. This reduced the overall size of the dataset, which promoted faster uploading and testing.

3.3 API Design

The API has three endpoints to run the full pipeline given S3 data, image URLs, or Base64 encoded images. Due to the nature of HTTP requests, the data is given to the endpoints in JSON format and is parsed out into a list of strings in the case of S3 data and image URLs, and a map of strings in the case of Base64 encoded images. These three endpoints also have a path variable that indicates whether Mask R-CNN or Faster R-CNN should be used. These three endpoints take the data in the given format and the path variable and call upon a service that has methods to call the appropriate detection model for each of the three input types. There are also methods for preprocessing the images in each of the three input types which takes the bounding boxes returned from the detection model and crops the input images accordingly. These preprocessing methods return the newly cropped images as a collection of files which can be passed to the method that runs the identification model.

To allow users to upload images on the frontend, there is also an endpoint that runs the full pipeline and takes an array of MultipartFiles and a boolean both as request parameters. The array of MultipartFiles are the uploaded files from the frontend and the boolean is to determine which detection model should be run. The reason why none of the previously mentioned full pipeline endpoints could be used was due to the format the images have to be uploaded in on the frontend. This endpoint calls upon a service that converts the array of MultipartFiles to a map of Base64 encoded images and their file names so that the method for the full pipeline given Base64 images can be called.

In addition to the endpoints that allow the full pipeline to be run, there are also endpoints that allow just the preprocessing step to be run for Base64 encoded images, S3 images, and image URLs. The endpoint for preprocessing Base64 encoded images takes JSON for a list of Base64 encoded bounding boxes and a map of the encoded files and their names. Similarly, the endpoint for preprocessing image URLs takes JSON for a list of bounding boxes and a map of the URLs and their file names. The endpoint for S3 images takes a string of JSON for the S3 keys and the bounding boxes. All three of these endpoints return a collection of the files of the cropped images.

Endpoints to just call the detection model or identification model for all three input types are also available. The endpoint for calling the detection model on Base64 encoded images takes a map

of filenames to image encodings. The endpoint for calling the detection model on S3 images takes a list of the S3 keys. The endpoint for calling the detection model on image URLs takes a list of image URLs. All three detection model endpoints also take a boolean which if false indicates that the pipeline should use Faster R-CNN and if true indicates that the pipeline should use Mask R-CNN. These detection model endpoints all also return a collection of bounding boxes. The endpoint for calling the identification model on Base64 encoded images takes a string of JSON for the filenames and the image encodings. The endpoint for calling the identification model on S3 images takes a list of the S3 keys. The endpoint for calling the identification model on image URLs takes a list of the image URLs. All three of the identification model endpoints return a collection of strings which contains the possible IDs for the given images.

4 R-CNN-based Seal Detection

4.1 Model Choice

For the seal detection model, two models were focused on specifically: Mask R-CNN and Faster R-CNN. An R-CNN, or Region-based Convolutional Neural Network, is a type of deep machine learning model primarily used for object detection in images. A simple overview of how an R-CNN works is that it first performs a region proposal, where the algorithm generates a set of proposed regions likely to contain objects. These proposals are typically generated using a selective search algorithm. Next is feature extraction, where each proposed region is then warped to a fixed size and passed through a convolutional neural network (CNN) to extract features. This CNN is typically pre-trained on a large dataset like ImageNet. The features extracted from each region are then fed into two separate, fully connected layers. One branch predicts the probability of the presence of an object within the region, known as classification, while the other branch refines the bounding box of the detected object. Finally, a post-processing step called non-maximum suppression is applied to eliminate duplicate detections and refine the bounding boxes.

After researching other object detection models, the team decided to focus on R-CNNs due to their higher accuracy and bounding box refinement. However, R-CNNs can be computationally expensive since they process each proposed region independently. Faster R-CNN and Mask R-CNN, which is an extension to Faster R-CNN, address this issue by sharing computation across regions, making the detection process faster and more efficient. Due to YOLO's speed with object detection, a detection model that could be both quick and accurate was necessary to compete. Faster R-CNN models were typically ranked highly among other object detection models. While YOLO was usually quicker due to its "look once" algorithm, most Faster R-CNN models were deemed more accurate, without sacrificing significant speed.

4.2 Architecture Details

Faster Region-Based Convolutional Neural Network (Faster R-CNN) is an advanced deep learning model used for object detections which integrates a Region Proposal Network with a Faster R-CNN detector that enables real-time performance. For this project, the Faster R-CNN model was built on a pre-trained MobileNetV3 Large as backbone. The activation function used

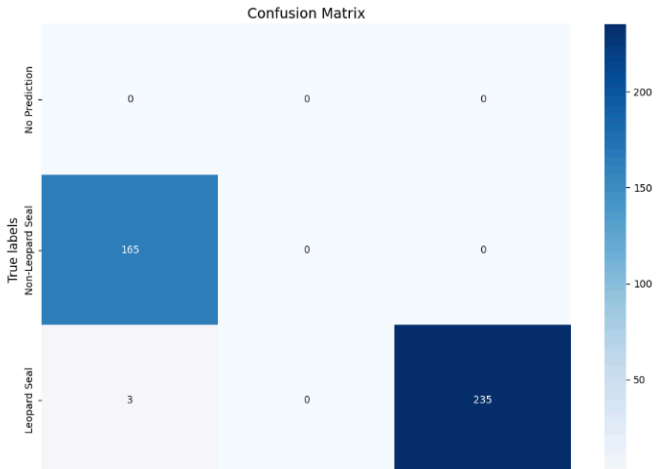
for this Faster R-CNN model is AdamW, a variant of the Adam optimizer, which decouples weight decay from gradient updates, enabling better regularization and convergence than traditional Adam. Furthermore, the learning rate was set to be 0.0001, which is relatively low. By setting a low learning rate, the updates to the pretrained weights are small, maintaining most of the valuable features that the model already learned. In addition, the weight decay was set to be 0.0001 to avoid overfitting and to help regularize the model.

Mask R-CNN, as an extension of Faster R-CNN, has much of the same backbone and architecture. Its main draw is that it adds a branch for predicting segmentation masks as well as bounding boxes. The version of Mask R-CNN used specifically for this project was built on a Feature Pyramid Network and a ResNet 101 backbone. This backbone is used as a feature extractor. The activation functions used within the Mask R-CNN layers include ReLU (Rectified Linear Unit) for the hidden layers and softmax for multi-class classification. Sigmoid activation is typically used for binary classification tasks and for generating the pixel-wise mask predictions. The learning rate was set to 0.001, as smaller learning weights converge faster and prevent exploding weights.

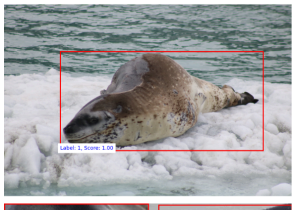
4.3 Performance Evaluation

After retraining many times with different datasets, hyperparameters, and epochs, both models reached a stage where they couldn't be improved any further within the project timeline. Both models were trained with 50 epochs to evaluate their performance on the same playing field.

To evaluate the fine-tuned Faster R-CNN model, results on 300+ testing images containing 165 non-leopard seal objects and 238 leopard seals were visualized. In addition, confusion matrix and classification reports were generated. For leopard seal detections, Faster R-CNN has outstanding performance with precision of 1.00, recall of 0.98, F1 score of 0.99, and accuracy of 0.98. Out of 165 non-leopard seal objects which contain non-seal objects and seals of other species, none of them are falsely classified as leopard seals. Out of 238 leopard-seals, the model is able to detect 235 leopard-seals. Furthermore, the model is capable of detecting multiple seals in one image, drawing precise bounding boxes and distinguishing leopard seals from other species of seals. Since the model only detects one class, leopard seals, and there are two classes, leopard seal and non-leopard seal, in the testing dataset, all objects in the testing dataset that had no prediction were labeled as no-prediction in order to compute confusion matrix and classification report. As a result, the below confusion matrix and classification report is computed with 3 classes.



	precision	recall	f1-score	support
no prediction	0.0000	0.0000	0.0000	0
non-leopard seal	0.0000	0.0000	0.0000	165
leopard seal	1.0000	0.9874	0.9937	238



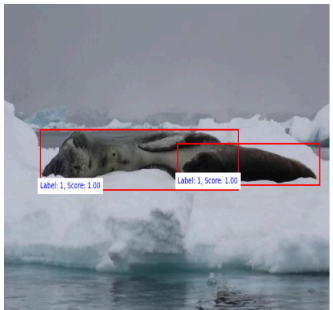


Figure 1: Faster R-CNN Results

The final version of Mask R-CNN had a mAP of 0.91. mAP stands for Mean Average Precision, and is a commonly used metric to evaluate the performance of object detection models. Higher mAP scores indicate better performance, with a perfect score of 1.0 indicating that the model achieves perfect precision and recall for all classes. So, a high score of 0.91 means Mask R-CNN performed very well at predicting seals vs non-seals. However, it was much slower when being tested on images compared to Faster R-CNN, most likely due to Mask R-CNN also having to create a mask for each image. It also took much more time to gather data and train Mask R-CNN, since each image in the training set had to be labeled and masked manually, while Faster R-CNN only needed bounding boxes which were much easier to create. This led to the decision to keep Faster R-CNN as the default detection model in the full pipeline, with the option to use Mask R-CNN if Faster returned poor results.

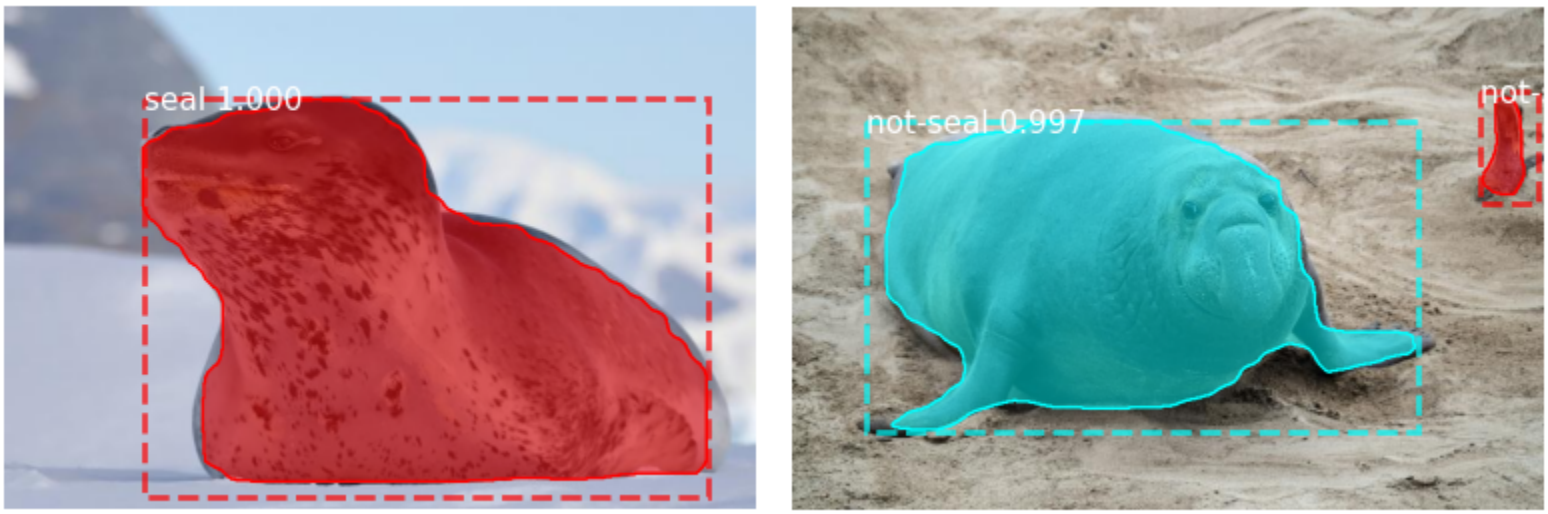


Figure 2: Mask R-CNN Results

5 DeiT-based Seal Recognition

5.1 Model Choice

When left to determine the model for the identification task, proof of concept was given that Vision Transformers (ViT) could be utilized for accurate identification of leopard seals. Further

research proved that ViT could achieve state-of-the-art or near state-of-the-art accuracy on facial recognition benchmarks for humans. Extending that process to leopard seals proved to be worthwhile. However, a limitation on ViT is that the models do not work with a relatively small database, which prompted further investigation.

The team opted to utilize the Data-efficient image Transformer (DeiT) model for the identification task of leopard seals due to its proficiency in handling scenarios with limited data availability. While ViT has demonstrated excellence in tasks such as facial recognition, and has shown promise in identifying leopard seals, the DeiT architecture is particularly well-suited for situations where data is sparse. As the identified leopard seal dataset may not encompass an extensive array of seal images, DeiT's ability to efficiently learn from smaller datasets becomes invaluable. By leveraging techniques like distillation, DeiT can distill knowledge from a large pre-trained model into a smaller one, enabling effective learning even with constrained data resources. Therefore, in our pursuit of accurate and reliable leopard seal identification, the choice of DeiT stands as a strategic decision to maximize performance under data scarcity constraints.

5.2 Architecture Details

The DeiT model comprises several critical elements that contribute to its effectiveness in handling image recognition tasks, particularly in situations with limited data availability.

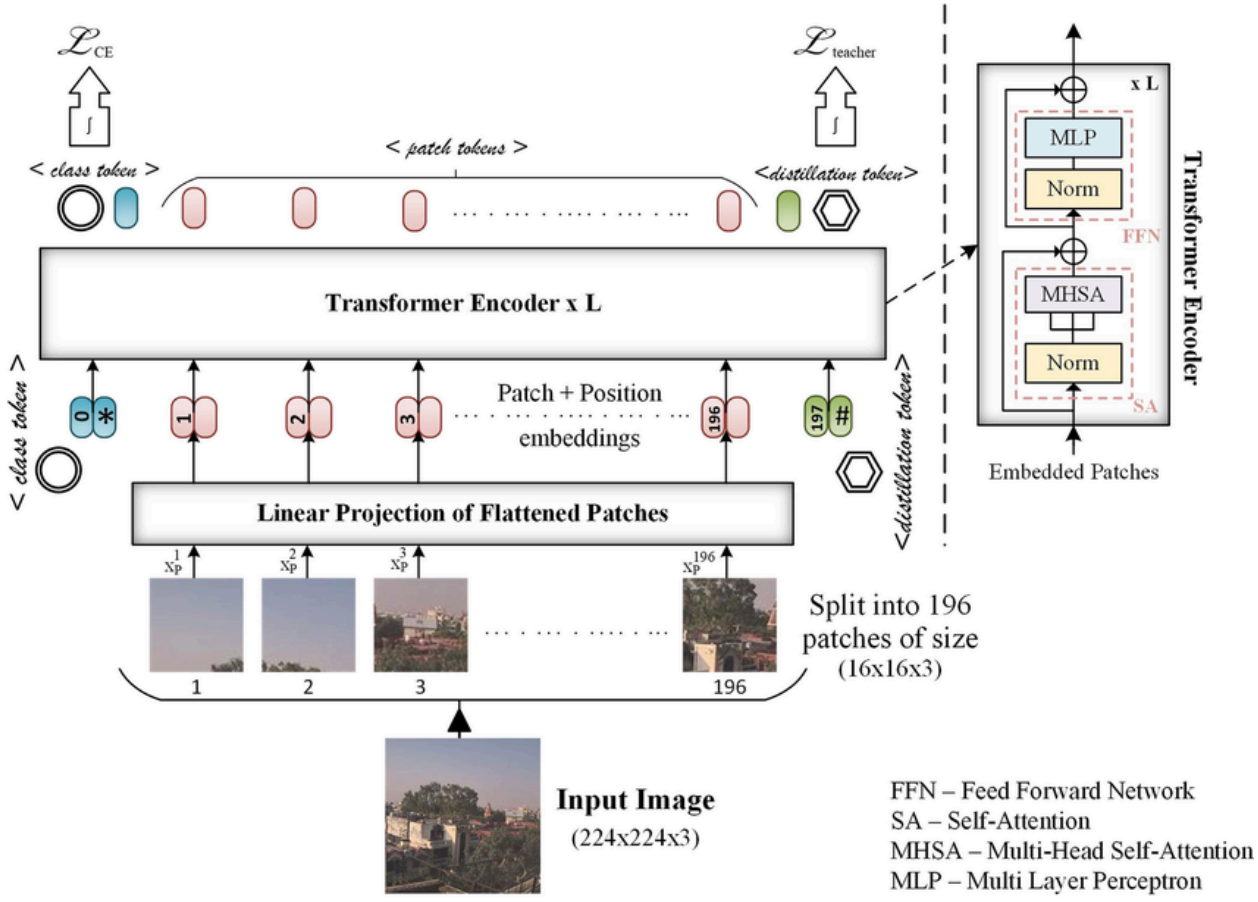


Figure 3: Visual Representation of DeiT model architecture

5.2.1 Attention Mechanisms
The attention mechanisms in DeiT play a vital role in capturing information in input images. DeiT leverages this capability to attend different parts of an image, and can learn meaningful representations for accurate identification. DeiT employs a self-attention mechanism, allowing each token to attend all other tokens within an input sequence, capturing long-range dependencies efficiently.

5.2.2 Patch Size
A defining characteristic of both ViT and DeiT is the patch-based processing approach, where an image is divided into smaller parts. Input images are divided into smaller patches, each of which is then treated as a token in the transformer architecture. The choice of patch size determines granularity of information captured in the model, where larger patch sizes will have larger context, but overlook finer details. On the other hand, smaller patch sizes will have a higher resolution, but may increase the computational complexity.

5.2.3 Embedding Dimensions
Embedding dimensions refers to the size of the token embeddings in the DeiT model. These embeddings are initial representations of input patches before being processed by the transformer layers. The dimensionality of these embeddings influence the capacity of the model, to capture and represent information from the input image. Typically, larger embedding dimensions capture more feature representations, but will also require more computational resources.

By understanding the architectural parameters of the DeiT model, it is possible to tailor it to suit the specific requirements of the leopard seal identification task, optimizing both performance and efficiency.

5.3 Explainability
Understanding the inner workings of the DeiT model and how it focuses on different parts of input images during seal recognition is crucial for interpreting and gaining insights into its decision making process. This section presents attention visualization to clarify the model's focus on different aspects of seal images.

Attention visualizations offer valuable insight into how the DeiT model attends to different parts of an input image when making predictions. These provide information on which regions the model focuses on and how strongly it attends to them by visualizing attention weights across different spatial positions. These visualizations aid in understanding the model's reasoning and decision-making.

Figure 4 showcases attention heatmaps that illustrate the different sections the model attends to across a variety of seal images. The predicted identity and actual identity is also provided to understand which parts of an image will positively affect the models ability to predict seal identities. This gives valuable insight into which parts of the image receive the most attention from the model.

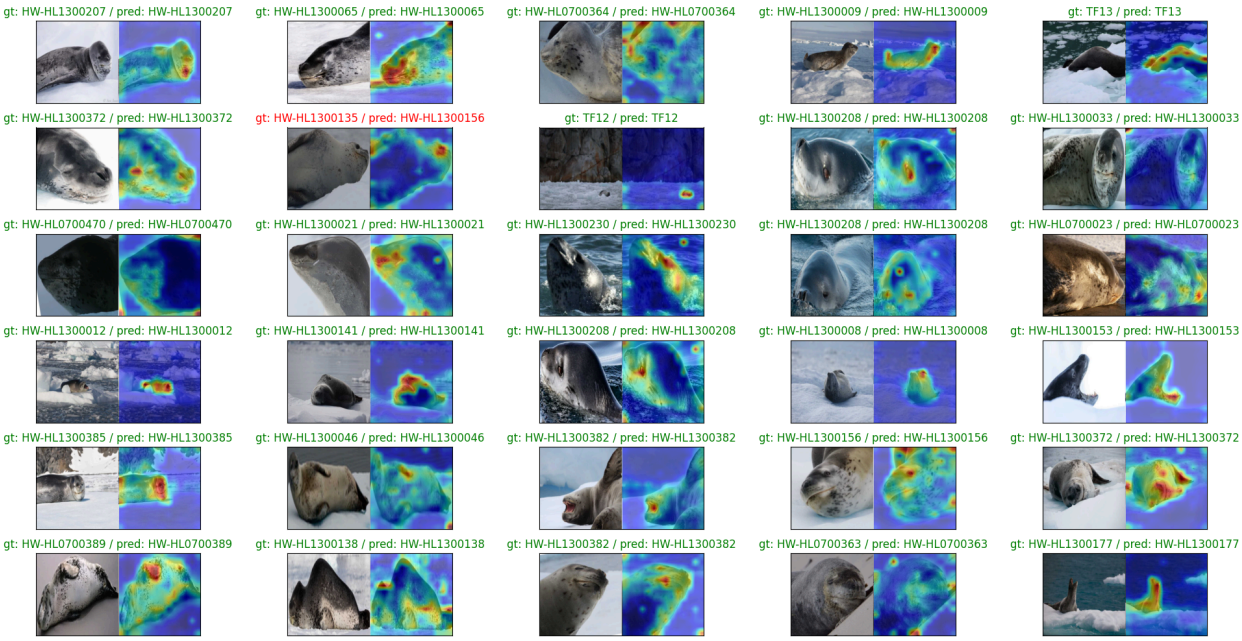


Figure 4: Attention Heat Maps generated from DeiT model

5.4 Performance Evaluation

Evaluating the performance of the DeiT model in identifying leopard seals includes both qualitative and quantitative analysis through visual examination and numerical metrics, respectively. This section presents the performance metrics of the DeiT model and highlights successful and challenging scenarios.

Assessment of the DeiT model through standard metrics such as accuracy, top 5 accuracy, balanced accuracy, and F-1 score all provide insights into the models ability to correctly identify leopard seals. These metrics were analyzed across a variety of models with different datasets. Table 1 highlights the different dataset options, along with the control ViT. Each of the metrics were calculated using the test dataset.

Dataset Used/Model	Top-1 Accuracy	Top-5 Accuracy	Balanced Accuracy	F-1 Score
Default/ViT	.92	.96	.91	.90
Default/DeiT	.92	.98	.91	.91
Favor train/DeiT	.92	**.99**	.91	.91
Favor test/DeiT	.89	.94	.88	.88
Additional copies of images/DeiT	**.93**	**.99**	**.92**	**.92**

Table 1: Evaluation of metrics on different model/dataset options

Table 1 highlights key interests from the model. The data augmentation technique of copying and randomly rotating proved to have the highest scores among all of the categories. This could have been caused by the model training on images and then being tested on images similar to those training images. The favor test dataset proved to be the most inaccurate, also due to the fact that the model was tested for leopard seals it had not been exposed to.

Sample images are also provided to show successful and challenging identification scenarios. The heatmaps in Figure 5 show that the DeiT model is highly accurate in predicting seal identities, regardless of position, angle, pose, lighting condition, or background clutter. Figure 5 shows the model accurately recognizing parts of the seal and paying attention to patterns along its torso. The spot patterns on a leopard seal is the most accurate way to identify identities, and reinforcing the model to attend areas with the spot patterns provide higher results. It is also interesting to note that the DeiT model pays particular attention to the mouths of a leopard seal in any image, as seen in Figure 6. This has not affected accuracy of the model in current tests.

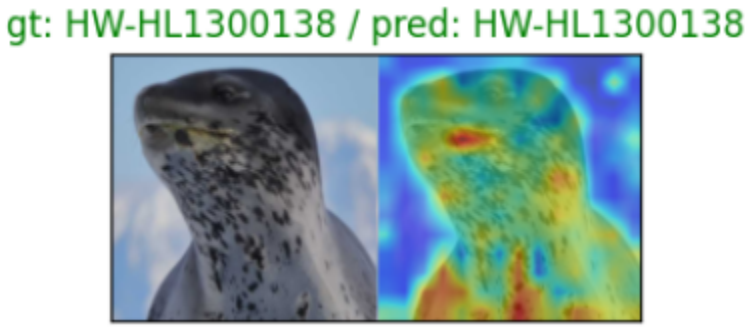


Figure 5: Accurate prediction heatmap focused of spot pattern

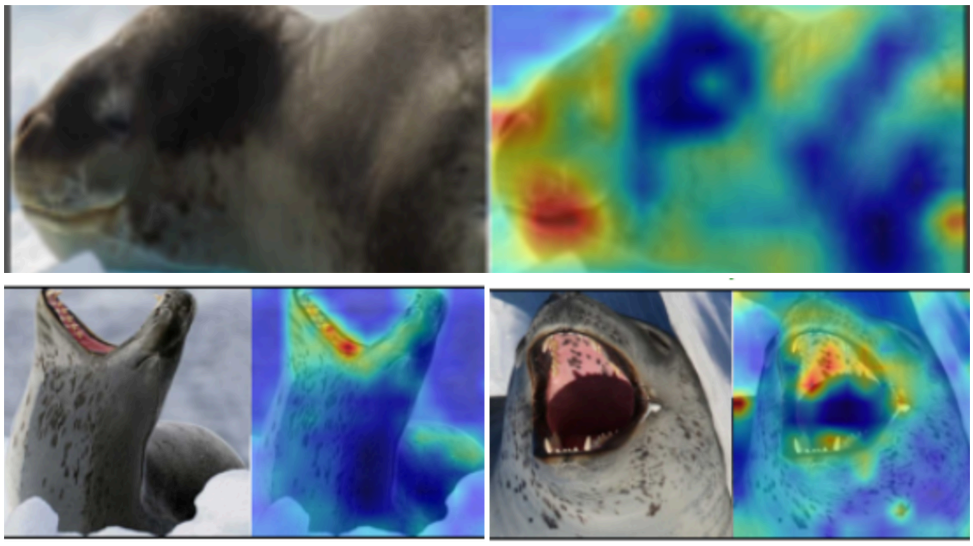


Figure 6: DeiT model focusing on Leopard Seal Mouths

Meanwhile, challenges discovered include the DeiT model focusing on eyes, flippers, or tails of leopard seals, each of which are not as accurate prediction factors for leopard seals. Figure 7 shows the implication of this issue. On the other hand, the DeiT model encounters difficulty in attending to the outer bounds on an image incorrectly. This leads the DeiT model to make a correct prediction on incorrect data. Figure 8 highlights this issue as the DeiT model attends to the top left corner of the input image, where the image has a unique crop or augmentation.

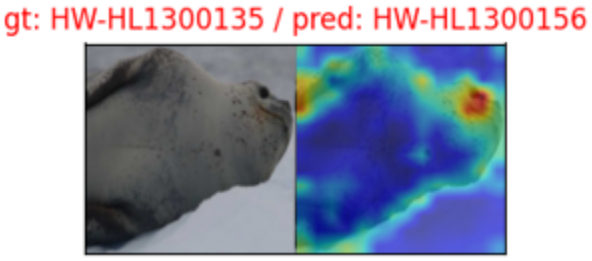


Figure 7: Incorrect identification of leopard seal due to model focus on eye

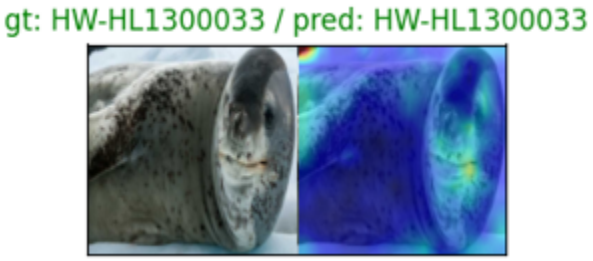


Figure 8: DeiT model is attending to edge of picture

Future iterations of this model would include reinforcing the model to attend to spot patterns found on a leopard seal, and punishing the model for attending elsewhere.

6 Integration and Testing

6.1 System Integration

The R-CNN and DeiT models were initially trained and tested using Jupyter Notebook before relevant functionalities were exported to .py files and trained weights were exported to the appropriate file format for each model. The main .py file for each model runs a single-threaded program that creates an instance of the given model using the appropriate weights file, runs each of the provided images through the model, then returns the results and terminates. This approach was taken due to current uncertainty about the interface of the Happywhale WhaleID python server, which the leopard seal models are intended to emulate.

6.2 Testing Methodology

Testing methodology included utilizing JUnit within the API. The testing framework was designed to validate the functionality of the models and functions within the API by calling them with input images and verifying the correctness of the output results.

The testing process involved several key steps:

1. Input Image selection: A diverse set of input images were selected to represent various leopard seal identities and scenarios to ensure comprehensive coverage of the model's capabilities.
2. API Integration with JUnit: The testing framework leverage JUnit testing in order to automate the execution of test cases.
3. Model Invocation: The various models were executed through the JUnit with the input images as parameters.
4. Result Verification: After executing the models, results were compared to expected results for correctness and accuracy. Errors were also tested if expected.

Leveraging the API with JUnit for testing ensured rigorous validation of the detection and identification models in a controlled and automated manner. This bolstered development and quality assurance of the API and models.

7 Discussion

7.1 Key Findings

One of the main takeaways regarding the training and use of machine learning models is that having a carefully organized dataset is crucial for training DeiT or any CNN like Mask R-CNN or Faster R-CNN. A good dataset forms the basis for training these models effectively as it helps them learn better, reduce biases, and improve their ability to understand different types of images. So, paying attention to how data is collected and organized is essential for getting the best performance out of these models in computer vision tasks.

Another crucial insight obtained was the proper selection of a machine learning model based on resource constraints. The primary reason DeiT is chosen as the identification model is its capability to achieve adequate training levels with fewer images. This highlights the importance of choosing models that can perform well under resource limitations, maximizing efficiency and effectiveness in training.

7.2 Challenges and Limitations

One significant limitation encountered during the project was the scarcity of seal images available for training the models. This limitation proved to be a considerable hurdle as it restricted the quality at which we could train the two models effectively. With fewer seal images in the dataset, the models had fewer examples to learn from, potentially hindering their ability to accurately identify or classify seals in new images. To address this issue, we utilized data augmentation to repurpose images within our dataset, thereby expanding its size and diversity. By applying techniques such as rotation, flipping, and adjusting brightness and contrast to existing seal images, we effectively created new variations of the data.

Another challenge we encountered was the uncertainty surrounding the integration of the models into the API. Our lack of knowledge regarding how Happywhale implemented existing models within their system made it difficult to determine the most suitable course of action for integrating our models. This uncertainty posed a significant obstacle, as it impacted our ability to effectively integrate our models into the API infrastructure. Due to this uncertainty, we opted to

run the models locally to showcase the functionality of our API. This decision opened up the opportunity to demonstrate the capabilities of the models independently of Happywhale's system architecture. By doing so, flexibility was provided for Happywhale to make any necessary adjustments to better align with their existing infrastructure.

8 Conclusion and Future Work

8.1 Recap

Motivated by the laborious manual process of seal identification, the ALSRS leverages machine learning to automate this task, akin to Happywhale's WhaleID for whales. The system encompasses two primary machine learning models, employing Region-based Convolutional Neural Networks (R-CNNs) for detection and Data-Efficient Image Transformers (DeiT) for identification, and achieving notable accuracies.

The system's architecture accommodates various image input formats and comprises three primary endpoints for full pipeline execution or preprocessing. Data collection and preprocessing phases involve extensive efforts to curate datasets and annotations, crucial for training and evaluating the models effectively.

Two primary detection models, Faster R-CNN and Mask R-CNN, offer robust object detection capabilities, while the DeiT model excels in identifying leopard seals with limited data availability. Evaluation metrics showcase the models' performances, revealing nuances in accuracy based on dataset characteristics and augmentation techniques.

8.2 Future Directions

The top priority for future development is the adaptation of the Faster R-CNN, Mask R-CNN, and DeiT python scripts into python servers following the interface used by the existing PyWhaleID server. By incorporating the ALSRS directly into Happywhale's infrastructure as a server, researchers will benefit from a unified platform for both whale and leopard seal identification, streamlining their conservation efforts and enhancing their database's comprehensiveness.

If integration to Happywhale is not possible, future development of the ALSRS API could focus on functionality to account for an ever-growing dataset. This could include methods for creating new seal IDs when identification confidence thresholds are sufficiently low, the addition of a native database to store images and IDs, and the addition of methods for displaying this data in various ways including returning all images for a given seal ID, returning images with similar confidence levels on multiple IDs, or methods allowing manual modification of the existing or auto-generated seal IDs.

Future model development in either environment could include the addition of a feature extractor or similar feature to DeiT, allowing for the reinforcement of spot pattern focus during the automatic identification process as with manual identification. Testing of the current model shows that DeiT often focuses on a seal’s eyes and nose or on the environment around them, which can introduce error when returning IDs and could be improved by this addition.

9 References

[1] Happywhale. (n.d.). https://happywhale.com/whaleid

[2] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghan, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N. (2021) *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* ICLR 2021 https://arxiv.org/pdf/2010.11929

[3] Zhong, Y., & Deng, W. (2021, April 13). *Face Transformer for Recognition*. arxiv.org. https://arxiv.org/pdf/2103.14803v2

[4] Sun, Z., & Tzimiropoulos, G. (n.d.). *Part-Based Face Recognition with Vision Transformers*. https://bmvc2022.mpi-inf.mpg.de/0611.pdf

[5] Mohan, A., & Abraham, L. (2024). *An ensemble deep learning approach for air quality estimation in Delhi, India. Earth Science Informatics*. 1-26. 10.1007/s12145-023-01210-5.