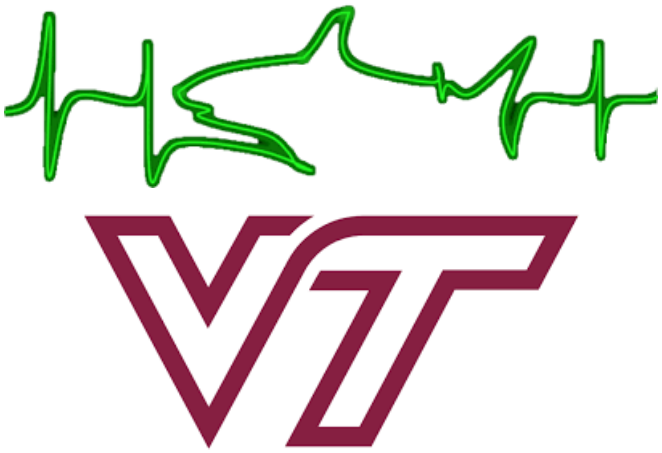
Automated Shark Classification using Vision Transformers

CS 4624
Multimedia, Hypertext, and Information Access



Final Report
May 7, 2023
Virginia Tech, Blacksburg, VA 24061

Instructor: Dr. Edward A. Fox
Client: Dr. Francesco Ferretti
Authors: Alfred Premkumar, Nicholas Cho, Charan Srinivasan, Jack Golden

Contents

List of Figures

List of Tables

1 Abstract

In recent years, collaborative work between previous CS4624 capstone groups and Dr. Francesco Ferretti's team contributed to the development of the SharkPulse project. With the goal of enhancing shark conservation and elevating public awareness through the collection and analysis of global, crowd-sourced shark sightings data, SharkPulse developed a data / machine-learning pipeline to detect and classify sharks from a given image. This report presents the improvement of the machine-learning pipeline previously established in *Shark detection and classification with machine learning* (Jenrette et al.) [1]. The improvements to the pipeline increased classification accuracy as well as species breadth. Mainly, the existing classification architectures are replaced with Transformers (ViTs). The updated shark identifier achieves an accuracy of 96%, the updated genus classifier, an accuracy of 72%, and the updated genus-specific species classifiers, an average accuracy of 74%. This updated classification system is able to classify 27 genera and 51 species. A framework for automating data-collection, model training, and maintenance is also introduced. Potential future work is discussed, including integrating the model into the SharkPulse platform.

2 Introduction

2.1 Background

Shark populations around the world are now more than ever at risk of extinction, declining more than 70% since 1970 [4]. Sharks are a great indicator of a healthy ocean ecosystem, as they maintain and shift populations of lower fish species, allowing the marine food chain to progress. Without sharks, overpopulation of sick and weak species can become invasive, disrupting the ecosystem's natural cycle [2]. Over-fishing, more specifically shark-finning, has been the main issue shark populations are facing. Sharks are often a by-catch (accidentally caught or undesired catch) of large fisheries, and when caught, their fins are removed and harvested to be sold illegally for medicinal and culinary cuisines, particularly in China. After a shark's fins are harvested, their bodies are discarded in the ocean, not being able to swim, eventually drowning [3]. There are many reasons for the decline in shark populations: over-fishing as mentioned above, as well as poor fishery enforcement, lack of taxonomic knowledge, and underdeveloped methods of monitoring shark populations [1].

SharkPulse, created by Dr. Francesco Ferretti and the SeaQL lab team at Virginia Tech, aims to solve the issue of monitoring shark populations by creating a crowd sourced database of shark sightings around the world. The SharkPulse team utilizes its shark detector feature to fill their database with images for users to validate and identify. The cycle between shark detection and image validation is meant to eventually build a highly accurate species and genus classification model. [1].

2.2 Previous Groups

1. Shark Validation Monitor Team: Spring 2021

   The team was responsible for building the foundation for SharkPulse's current validation monitor and its database along with the WordPress webpage for users to login to, and participate in, the validation monitor. The validation monitor was designed to have people all over the world be able to help identify the shark images that were collected in the SharkPulse database. The team also had a focus to "gamify" the validation monitor, with a score keeping leaderboard for users to engage in friendly competition, the end result being shark population awareness and better monitoring [8]. Our task as the Shark Detection team is to improve the current training model to provide the validation monitor valid images of sharks in the database. Without the validation monitor, data collection, and web pages the previous team developed, our team would not be able to help contribute to the SharkPulse initiative.

2. Shark Validator Game Team: Fall 2021

   This team's goal was to implement the validation game while also improving the front end and back end of the previous team's work. Notable work completed by this team includes improving the point system of the validation monitor, introducing a species rarity system, and updating the identification

guide [7]. The focus on the validation monitor game is relevant to our group as we are attempting to improve the SharkDetector and the data it sends to the validation monitor game.

3. SharkPulse Validator Game Team: Spring 2022

   This team was tasked with changing the previous group's web design and back-end development into a more dynamic and responsive experience for SharkPulse users. The team also went on to improve the validation monitor point system. A majority of the back-end implementation utilized JavaScript and PHP networking. The front-end improvements were built on the pre-existing WordPress web design, changing layouts and forms throughout the website [6]. Being the most recent team to work on SharkPulse, this group's work definitely has been a major part in our team's development and work as we are again working on improving the SharkDetector to enhance the data set provided to the validation monitors.

2.3 Existing Work

SharkPulse encompasses a large suite of functionality focused on tracking shark species. For the purposes of this project, we are only concerned with improving the data-mining and machine-learning aspects. The initial piece of software at the time of this project's initiation was composed of a data-pipeline whose data came from two sources: SharkPulse and the internet. The SharkPulse team had yet to integrate the new data coming out of the validation monitors.

Figure 1 shows the flow of data throughout the major components of the machine-learning pipeline. At the beginning of the flow, there is labeled data that exists on the internet. Image scrapers fetch labeled data from various sources such as Instagram and Flickr by querying for species hashtags. This labeled data has uncertainty, so it must be run through a validation monitor in order to avoid collecting inaccurate data. Data from SharkPulse, however, is sent straight to the database. Note that this view of the pipeline does not reflect any implementation details and is intended only to show how labeled shark images (the input for machine-learning techniques) are obtained.

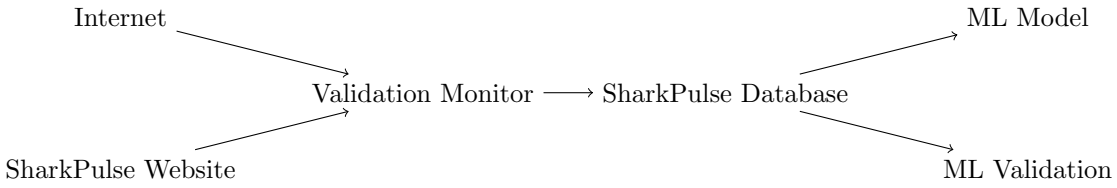


Figure 1: Data Flow Diagram

2.4 Objectives

Our team's objective is to help Dr. Ferretti and his team improve SharkPulse's current shark detection model. We have been given the SharkPulse shark detector code base and its database to try and create a more streamlined pipeline of data to train the model to become more accurate and be able to identify more shark species.

1. Increase Species Classification Accuracy by a significant amount (10%).
2. Species Breadth: There are 48 species currently and we aim to increase this number.

2.5 Client

The client for this project is Dr. Francesco Feretti, a marine ecologist and researcher. He has numerous research efforts, most notably the coverage of shark populations. His research uses statistical methods to draw important conclusions on the health of shark species around the world. As the founder of SharkPulse,

Dr. Feretti aims to establish a system that can automate the process of tracking sharks.

Jeremy Jenrette, a graduate researcher in the Virginia Tech Department of Fish and Wildlife Conservation, has made significant contributions to the SharkPulse initiative. He worked with Dr. Feretti to create a hierarchical classification model [1] based on the ResNet CNN (Convolutional Neural Network) architecture to identify over 60 different shark species. His work forms the basis for the Shark Detection team to improve upon.

2.6 Deliverables

The client and team agreed on the following deliverables:

Deliverable	Acceptance Criterion
Improved machine learning model	A significant increase in classification accuracy.
Automated data pipeline	Recurrent data collection, maintenance, and model training.

Table 1: Deliverables

2.7 Team

The Shark Detection team is composed of four members and is split into two groups: data preparation and model implementation. Although each member has a defined role, the group aims to collaborate as much as possible in order to ensure a successful outcome and to learn as much as possible along the way. The team communicates weekly with the client in order to stay up to date on all pressing matters.

- Jack Golden: Design and Automation Developer
- Nicholas Cho: Merging PSQL Database Tables
- Alfred Premkumar: Machine Learning Developer
- Charan Srinivasan: Developer, Gathering and Formatting the Testing Results

3 Requirements

3.1 Improved Machine Learning Model

The first of the two requirements is to improve the machine learning model. The model is to be improved in two ways: increasing the species breadth of the classifier and improving the accuracy of the classifier. Currently the classifier is capable of detecting 47 species relating to 26 genera, and our goal is to increase either the species or genera count, or both. The classifier currently is capable of detection if a shark is present at 90% accuracy and is capable of classifying the species at 70% accuracy. Our team will be delivering three new classifier models.

3.2 Automated Data Pipeline

Another requirement for this project is to improve the training data pipeline. Currently, the data table used for training the model isn't updated with any new data. In order to improve the data pipeline, our team is required to make it so that the data that is scraped from Instagram, YouTube, and Flickr is added to the training data. The data also submitted to the SharkPulse website should be added to the data table. Finally, the training data should be updated automatically.

4 Design

4.1 Hierarchical Classifier

The shark classification system is a hierarchical system, made up of four different models. The first model is the shark locator, which locates the shark like image within the picture. This was the only model we did not change, as it was outside of the scope of our project. Images that are determined to have shark like images are then pushed to the next model, the shark identifier. The shark identifier determines if whatever is in the image is a shark or not. Images that are classified as sharks are then moved to the next model, the shark genus classifier. This model determines the genus of a shark. If the number of images of a certain genus is lower than a set threshold, then that genus is placed into an "other genus" category. The final step is the shark species classifier. There are genus-specific species classifier models for each of the genera. The hierarchical model was chosen as per request from the clients. The hierarchical model provides for a simpler way to train and isolate each part of the classification process, while also decreasing the memory usage from the ViT networks. The classifiers used are small-scale CNN models with 5 layers each.

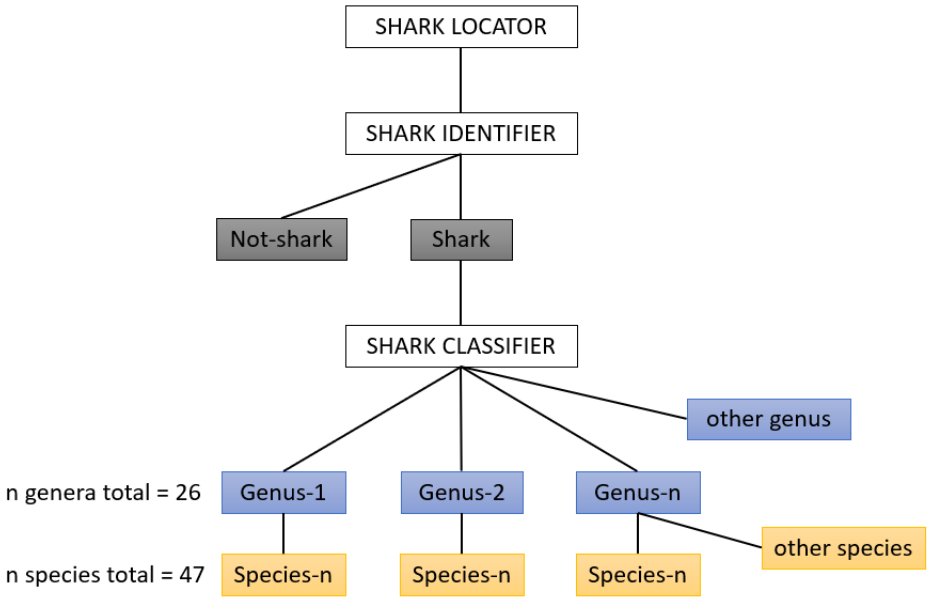


Figure 2: Hierarchical Model (adapted from [1])

4.2 Original Model Architecture

The initial shark classifier implementation (depicted in Figure 3) achieved a 90% accuracy with the Shark Identifier and an overall species classification accuracy of 70%.

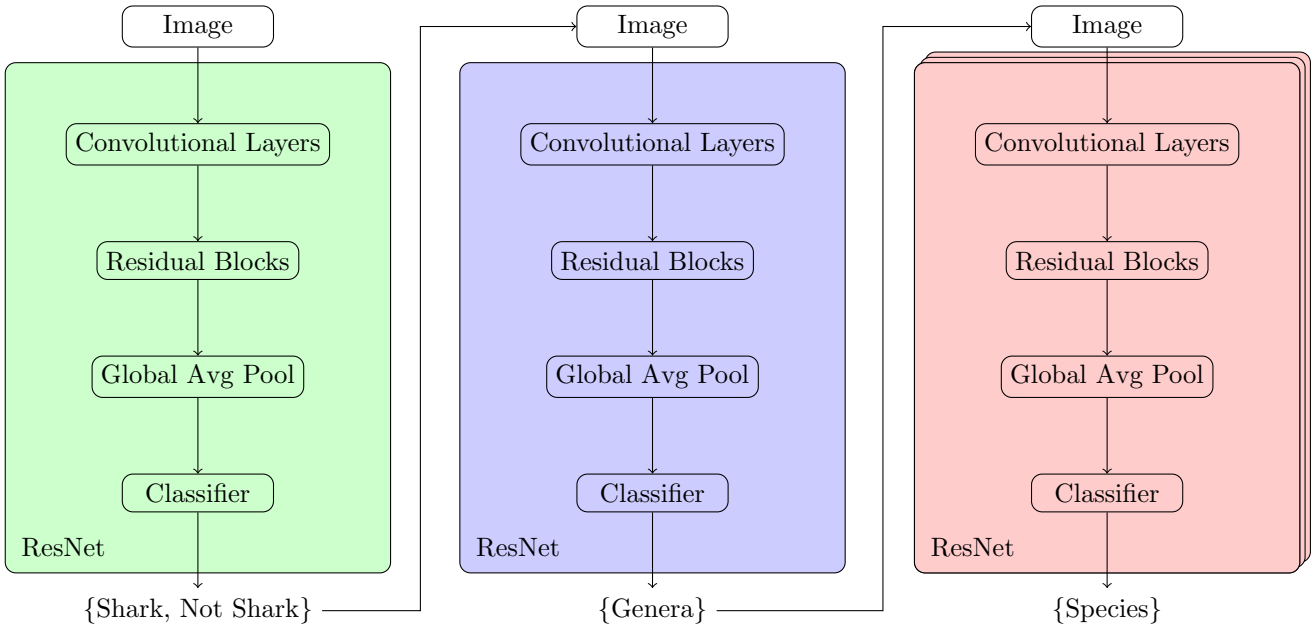


Figure 3: Original Model Architecture (ResNet)

4.3 Revised Design

The goal of the updated design is to boost overall classification accuracy and to automate training of the included models such that this entire classification system can be used in production on SharkPulse's website.

The updated methodology and data flows are shown in Figure 4.3.

The tiered approach to shark classification still exists in the shark detector, but all classification models will be updated to transformer-based classification models in an attempt to outperform the existing CNN model architecture. The preexisting CNN model has a classic ResNet-like architecture. As explained in *An Image is Worth 16x16 Words: Transformers For Image Recognition at Scale* (Dosovitskiy et al., 2021), Vision Transformer networks have been shown to perform better in terms of accuracy on popular image classification benchmarks when compared to more traditional architectures like ResNet. With this in mind, we intend to see a substantial increase classification accuracy after converted the current ResNet-based classification models to ViT-based classification models.

In addition to this, the training data database table that the existing classifier pulls from for training data will be updated periodically with data from multiple defined sources, including SharkPulse data, curated YouTube data, and the existing Flickr and Instagram sources. Alongside this, the training of the entire classifier will also be automatic, i.e., periodically retraining on new data. The new training data will be pipelined into a new table modeled after the current training table, and will only include validated and unique images from three different tables, sourced from Instagram, Flickr, and SharkPulse. After the new table is updated, the machine learning model will be able to train as well on new data.

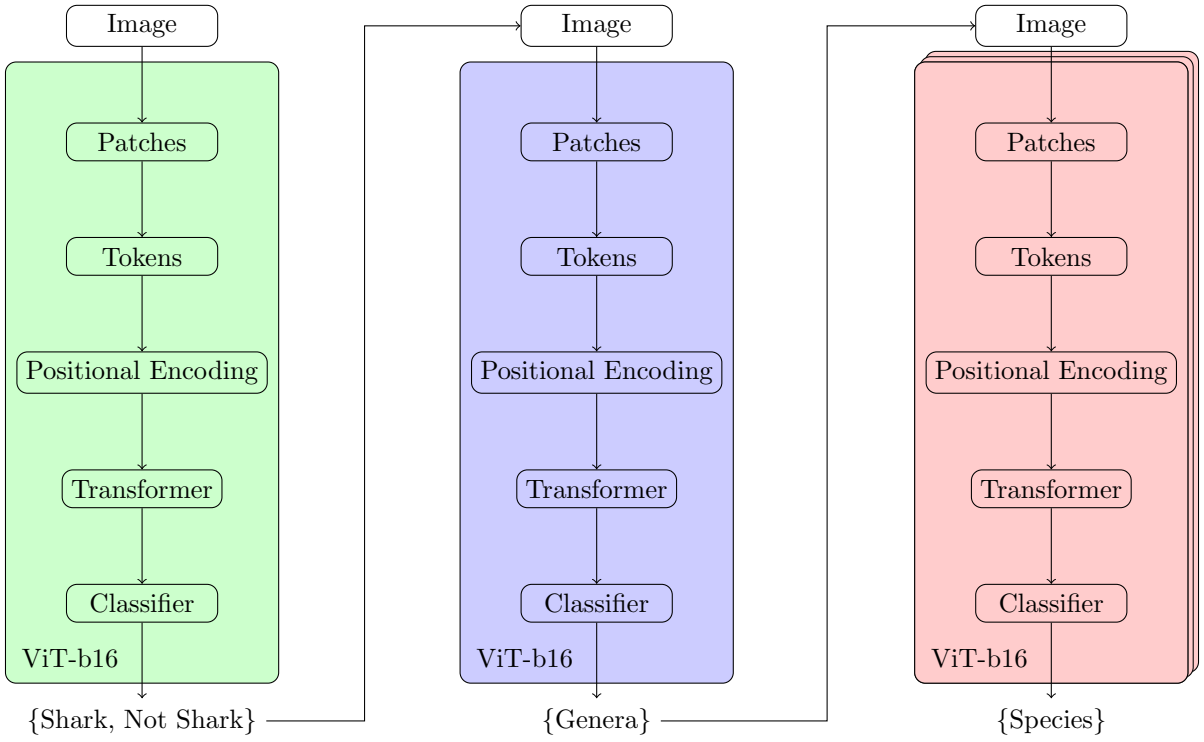


Figure 4: Revised Model Architecture (Vision Transformers)

5 Implementation

Each component in our solution, depicted in Figure 5, is enumerated along with a description in Table 2. The work we did involved replacing the machine learning components of the pipeline with vision transformer based models and congregating labeled images indexed by various tables in the SharkPulse database.

id	Description
1	Modification of the training_data table in the PSQL database. The table merges all sources of training data into a central location.
2	ViT model
3	ViT model
4	ViT model
5	Configurable Python script that runs the hierarchical training model. This script is to be automatically ran via a CRON job on the spr server.
6	cronjob which defines a time interval for which to make repeated calls to both the training $(\text{run}_t raining.py) and scraping (scrape.sh) shell scripts$.
7	Shell script that runs image scrapers. This script is to be automatically run via a CRON job located on the spr server.

Table 2: Component Modifications

5.1 Data Pipeline

One of our tasks was to incorporate the data output of the validation monitors into the data used to train the model. Each validation monitor–flickr, instagram, youtube–has its own table in the SharkPulse database. In order to congregate all of the validated data we created an SQL query which creates a new merged table of all the new validated data and merges it with the pre-existing training data.

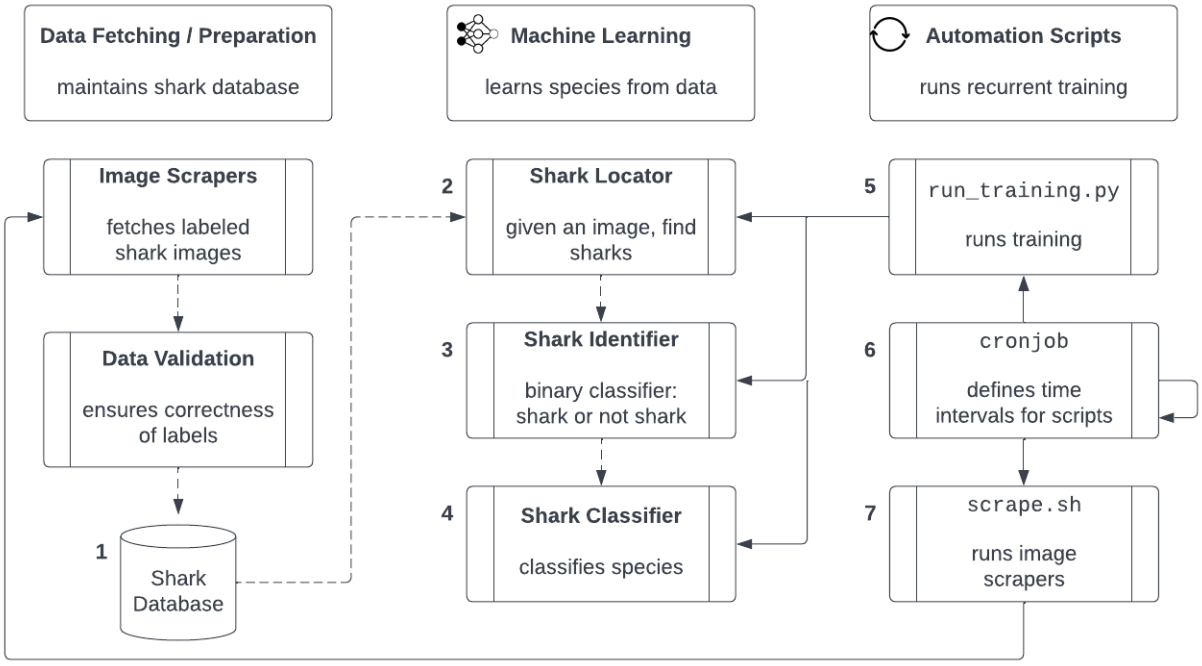


Figure 5: High Level Class Diagram

This is an automatic process in the sense that the new, merged table updates when any of its constituent validation monitor databases gets updated.

Database Schemas			
Instagram	File Name	Genus Name	Species Name
	IGfile1.jpg	Carcharodon	Carcharodon Carcharias
	IGfile2.jpg	Sphyrna	Sphyrna Mokarran
	...	...	...
Flickr	File Name	Genus Name	Species Name
	FLfile1.jpg	Galeocerdo	Galeocerdo Cuvier
	FLfile2.jpg	Ginglymostoma	Ginglymostoma Cirratum
	...	...	...
SharkPulse	File Name	Genus Name	Species Name
	SPfile1.jpg	Carcharhinus	Carcharodon Carcharias
	SPfile2.jpg	Sphyrna	Ginglymostoma Cirratum
	...	...	...

Table 3: Validated Data Table Schemas

Image Name	Genus	Species
IGfile1.jpg	Carcharodon	Carcharodon Carcharias
IGfile2.jpg	Sphyrna	Sphyrna Mokarran
...	...	...
FLfile1.jpg	Galeocerdo	Galeocerdo Cuvier
FLfile2.jpg	Ginglymostoma	Ginglymostoma Cirratum
...	...	...
SPfile1.jpg	Carcharhinus	Carcharodon Carcharias
SPfile2.jpg	Sphyrna	Sphyrna Mokarran
...	...	...

Table 4: Training Table Schema

5.2 PSQL Query Used

```
CREATE TABLE training_backup_new AS TABLE training WITH DATA;

CREATE TABLE merge_v3 AS TABLE training WITH NO DATA;

INSERT INTO merge_v3 (img_name, species)
    SELECT img_name, species_name FROM sharkpulse WHERE validated = 'true';

INSERT INTO merge_v3 (img_name, species)
    SELECT img_name, species_name_1 FROM data_mining WHERE validated;

INSERT INTO merge_v3 (img_name, species)
    SELECT img_name, species_name FROM instagram WHERE validated;

UPDATE merge_v3 SET genus = split_part(species, ' ', 1);

INSERT INTO training_backup_new (img_name, genus, species)
    SELECT DISTINCT(img_name), genus, species from merge_v3;
```

5.3 Hierarchical Vision Transformer Models

Taking into account our revised design, our primary goal for the models put in place was to improve the testing classification accuracy of the binary shark identifier, the genus classifier, and each of the genus-specific species classifiers. To that end, we remade the 3 R scripts that handle binary shark identification, genus classfication, and genus-specific species classification, respectively, as Python scripts that accomplish training similarly, but with a Keras implementation of the Vision Transformer. Specifically, we utilize the base ViT model with a patch size of 16, pretrained on the ImageNet-21k dataset (14 million images, 21,843 classes) at resolution 224x224, and fine-tuned on the ImageNet 2012 dataset (1 million images, 1,000 classes) at resolution 384x384. We then add a flatten layer, a batch normalization layer, a dense layer of size 32 with GELU activation, another batch normalization layer, and finally, a dense output layer with sigmoid activation – and a size of whatever the current model is trying to predict. This output size can vary based on the number of classes, genera, or species we predict.

When loading the training and validation data, we resize every image to 224 by 224 pixels and apply data augmentation, to add more data points of interest. In terms of hyperparameters, we set a base learning rate of 1×10^{-4} with which we use the rectified Adam optimizer. This learning rate can then be automatically reduced during training if a plateau in the validation accuracy is detected. We set the maximum amount of epochs to 30, but implement early stopping based on a plateau in validation accuracy, as the data seems to lend itself to fitting with relatively few epochs. For each of our datasets, we partition with training to testing ratio of 80:20, and then partition the new training partition into training and validation with a ratio of 80:20.

Images are loaded by matching file names from the database's training table to images found in local storage. All images are already sorted by class into shark or not-shark. A subset of these images have a validated genus class, and a smaller subset of those images have a validated species class. We follow the methodology used by the original paper setting data genus and species thresholds. For the genus classifier,

a genus with less than 433 images will have its images sent to a class to be predicted as "other_genus". Out of only the genera that the genus classifier can predict on, the species classifier then only loads images from species that have an image count of 161 or higher. Any species with a lower image count per genus-specific species classifier gets sent to a class to be predicted, containing the other species (e.g., "Alopias_species"). The exception to this is when a genus contains only two species and the same genus passes the genus classifier's image threshold. In this case we predict on both species, regardless of the number of associated validated images.

6 Testing and Results

6.1 Data Pipeline

In order to test that the data we were merging into the database wasn't malformed, we inspected various outputs of the resultant database from the query we used. Namely, we made sure that all data points had a unique image name and an associated genus/species. This was a crucial step to take to ensure that the data would be in a usable form for the machine learning model to use. After the merging process was completed, the model had 73,119 images for training; previously the training table used included 66,794 images.

6.2 Hierarchical Vision Transformer Models

We gather results on testing the shark identifier model, genus classifier model, and species classifier models on the new dataset made available through the work done after merging newly-validated data.

6.3 Binary Classification Results

The first of these is the binary shark classification model (or SI, for shark identification). We are able to give 117060 images in total, 66800 of which belong to the shark class, and 50260 of which belong to the not_shark class. We observed training taking 3 epochs before reaching a plateau in validation accuracy. The shark identifier model was then able to score an overall testing accuracy of 96%. Compared to the original paper's shark identifier accuracy score of 91%, we see a 5.5% increase in accuracy. Vision Transformer models tend to perform better at scale, so with the large number of images, this increase in accuracy falls in line with our expectations, even prior to an increase in training data.

	Accuracy			
Category	Precision	Recall	F1-score	Support
Not Shark	0.9847	0.9260	0.9544	10052.0
Shark	0.9467	0.9891	0.9675	13361.0
Macro avg	0.9657	0.9576	0.9609	23413.0
Weighted avg	0.9630	0.9620	0.9619	23413.0

Table 5: SI Accuracy Table

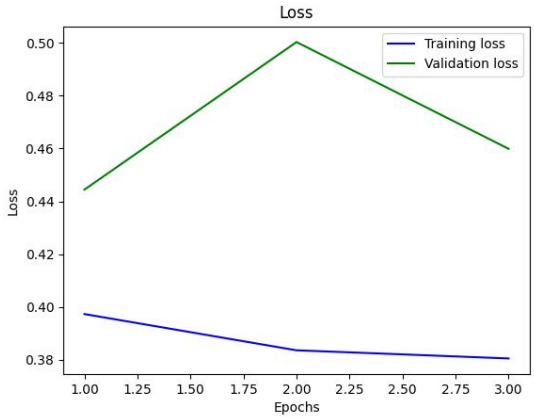


Figure 6: SI Loss

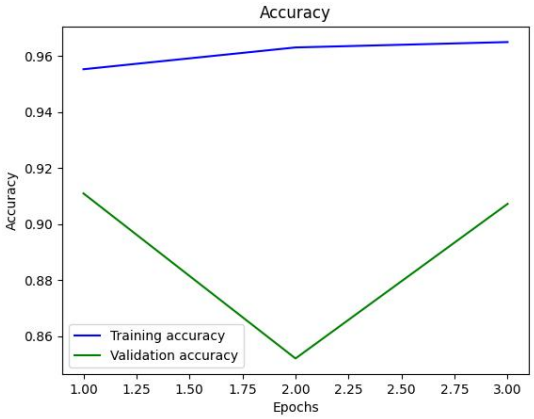


Figure 7: SI Accuracy

6.4 Genus Classification Results

The second of these is the genus classifier model. This is the first half of the shark classifier, which we are able to give 41724 images in total, compared to 26029 before the data merge. We observed training taking 16 epochs before reaching a plateau in validation accuracy. The model was then able to score an overall testing accuracy of 72%. Using the same threshold observed in the original work, we see a total of 27 genera able to be classified, one more genus (Hydrolagus) than the original work.

	Accuracy					Accuracy			
Genus	Precision	Recall	F1-score	Support	Genus	Precision	Recall	F1-score	Support
Alopias	0.7756	0.8008	0.7880	246.0	Galeus	0.9630	0.9043	0.9327	115.0
Brachaelurus	0.8269	0.8958	0.8600	96.0	Ginglymostoma	0.8117	0.7904	0.8009	229.0
Carcharhinus	0.7391	0.7307	0.7349	1322.0	Haploblepharus	0.3906	0.8345	0.5321	139.0
Carcharias	0.8791	0.8919	0.8854	481.0	Heterodontus	0.8700	0.5367	0.6639	449.0
Carcharodon	0.7831	0.8293	0.8055	457.0	Hexanchus	0.7686	0.8889	0.8244	198.0
Cephaloscyllium	0.7427	0.7341	0.7384	173.0	Hydrolagus	0.8352	0.8636	0.8492	88.0
Cetorhinus	0.7438	0.6977	0.7200	129.0	Isurus	0.7782	0.6738	0.7222	328.0
Echinorhinus	0.9667	0.8365	0.8969	104.0	Mustelus	0.3584	0.4559	0.4013	136.0
Galeocerdo	0.8311	0.5652	0.6728	322.0	Negaprion	0.6144	0.5165	0.5612	182.0
Galeorhinus	0.6142	0.7610	0.6798	159.0	Orectolobus	0.8843	0.9432	0.9128	405.0
Prionace	0.5825	0.8384	0.6874	198.0	Scyliorhinus	0.6897	0.5181	0.5917	193.0
Rhincodon	0.9315	0.9456	0.9385	331.0	Sphyrna	0.7061	0.4871	0.5765	503.0
Squalus	0.6424	0.4641	0.5389	209.0	Triaenodon	0.8062	0.9059	0.8532	372.0
Triakis	0.2540	0.0696	0.1092	230.0	Other_genus	0.4726	0.7729	0.5865	546.0
Macro avg	0.7236	0.7197	0.7094						
Weighted avg	0.7340	0.7210	0.7159						

Table 6: Genus Accuracies

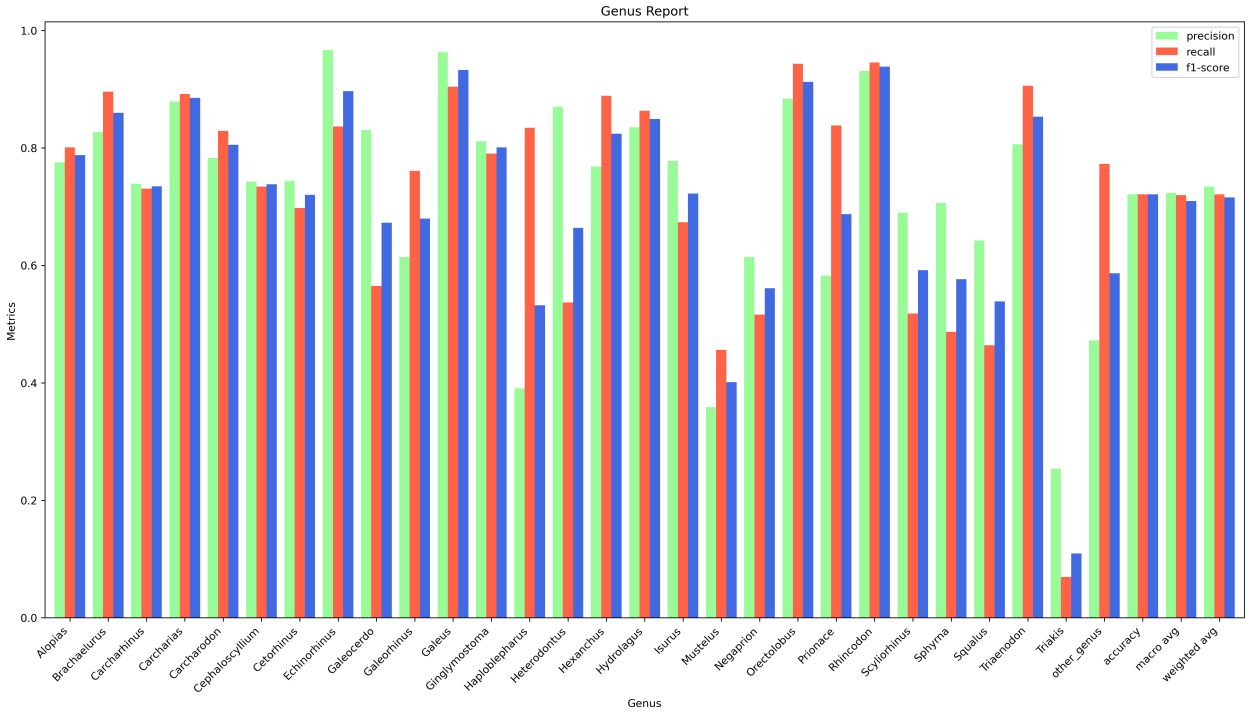


Figure 8: Genus Accuracies

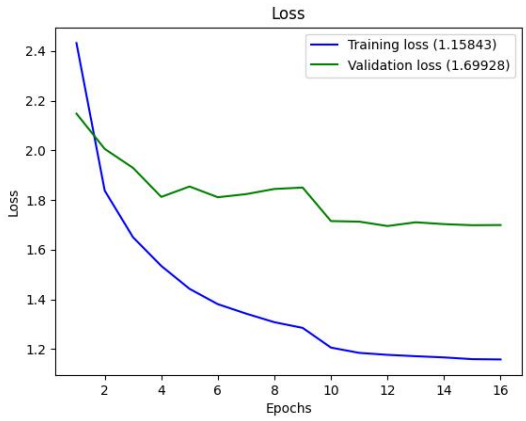


Figure 9: Genus Loss

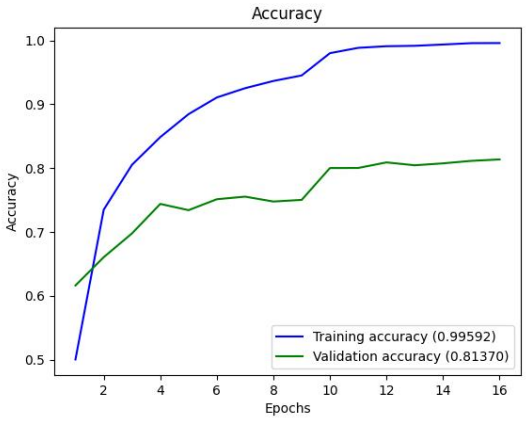


Figure 10: Genus Accuracy

6.5 Species Classification Results

The last part of the hierarchy and the second half of the shark classifier, includes the genus-specific species classifiers, which we are able to give 32109 images in total, compared to 12650 before the data merge. Using the same threshold observed in the original work, we also see a total of 51 species able to be classified, 3 more species than the original work (i.e., adding Carcharhinus galapagensis, Hydrolagus colliei, and Sphyrna zygaena), thanks to the newly-validated images.

Training each of the 27 genus-specific species classification models, counting the single-species models, all of which have a testing classification accuracy of 100%, we get an average classification accuracy of 82%. Discounting the single-species accuracy numbers, we get an average classification accuracy of 74%.

Classifier	Accuracy(%)	Classifier	Accuracy (%)
Alopias	80	Hexanchus	56
Brachaelurus	95	Hydrolagus	94
Carcharhinus	68	Isurus	64
Carcharias	100	Mustelus	62
Carcharodon	100	Negaprion	29
Cephaloscyllium	70	Orectolobus	97
Cetorhinus	100	Prionace	100
Echinorhinus	89	Rhincodon	100
Galeocerdo	100	Scyliorhinus	52
Galeorhinus	100	Sphyrna	40
Galeus	100	Squalus	68
Ginglymostoma	88	Triaenodon	100
Haploblepharus	89	Triakis	99
Heterodontus	94		
Average (no single-species)	74	Average	83

Table 7: Classification Accuracy of Genus-specific Species Classifier

Species	**Img Count(F1)**	**Species**	**Img Count(F1)**	**Species**	**Img Count(F1)**
Alopias	1228(0.79)	**Galeocerdo**	1619(0.67)	**Orectolobus**	2021(0.91)
A. vulpinus	353(0.84)	G. cuvier	1619(1)	O. halei	542(0.96)
Alopias spp.	174(0.73)	**Galeorhinus**	791(0.68)	O. maculatus	1019(0.97)
Brachaelurus	479(0.86)	G. galeus	791(1)	O. ornatus	281(0.96)
B. colcloughi	162(0.92)	**Galeus**	575(0.93)	Orectolobus species	97(0.87)
B. waddi	299(0.96)	G. melastomus	376(1)	**Prionace**	990(0.69)
Carcharhinus	6605(0.73)	**Ginglymostoma**	1142(0.80)	P. glauca	990(1)
C. amblyrhynchos	550(0.68)	G. cirratum	821(0.93)	**Rhincodon**	1651(0.94)
C. galapagensis	625(0.35)	G. unami	124(0.52)	R. typus	1602(1)
C. leucas	402(0.66)	**Haploblepharus**	691(0.53)	**Scyliorhinus**	964(0.59)
C. limbatus	895(0.66)	H. edwardsii	215(0.86)	S. canicula	378(0.64)
C. melanopterus	829(0.88)	H. fuscus	271(0.96)	**Sphyrna**	2514(0.58)
C. obscurus	259(0.74)	Haploblepharus spp.	194(0.80)	S. lewini	866(0.43)
C. perezi	245(0.6)	**Heterodontus**	2244(0.66)	S. mokarran	165(0.30)
C. plumbeus	212(0.58)	H. francisci	337(0.83)	S. tiburo	377(0.52)
Carcharhinus spp.	684(0.66)	H. galeatus	343(0.93)	S. zygaena	451(0.09)
Carcharias	2405(0.89)	H. japonicus	306(0.99)	Sphyrna spp.	17(0.67)
C. taurus	2405(1)	H. portusjacksoni	1172(0.95)	**Squalus**	1044(0.54)
Carcharodon	2290(0.81)	Heterodontius spp.	22(0.71)	S. acanthias	182(0.66)
C. carcharias	2290(1)	**Hexanchus**	990(0.82)	Squalus spp.	130(0.70)
Cephaloscyllium	861(0.74)	H. griseus	792(0.71)	**Triaenodon**	1859(0.85)
C. isabellum	323(0.77)	Hexanchus spp.	8(0.05)	T. obesus	1786(1)
C. laticeps	264(0.63)	**Hydrolagus**	436(0.85)	**Triakis**	1146(0.11)
Cephaloscyllium spp.	76(0.58)	H. colliei	282(0.95)	T. megalopterus	213(1)
Cetorhinus	642(0.72)	Hydrolagus spp.	109(0.90)	T. scyllium	161(0.95)
C. maximus	642(1)	**Isurus**	1636(0.72)	T. semifasciata	673(0.99)
Echinorhinus	516(0.90)	I. oxyrinchus	1360(0.77)	Triakis species	12(0.5)
E. brucus	6(0.27)	I. paucus	62(0.12)		
E. cookei	452(0.94)				

Table 8: Genus/Species F1 scores

6.6 Classification Discussion

For both the genus classifier and the genus-specific specific classifiers, a direct comparison between these results and those of the original work is not entirely possible, as the original paper reports results on newly-seen web-scraped data with the genus-classifier and the species classifiers as a dual-model system acting as an all-in-one shark classifier. The same can be done with the new ViT-based models, but was outside of the scope of this project. Even still, a relatively higher accuracy for both models was expected. Nevertheless, this

does not discount the ViT-based genus and species classifiers. As stated in the original Vision Transformer work, Vision Transformers typically outperform ResNet-based architectures at scale, when tested on much higher amounts of data when compared to ResNet counterparts. As such, we would expect these ViT-based shark classifiers to outperform the ResNet-based shark classifiers as tens of thousands of images become validated and available to the models to train, especially as the SharkPulse system is further automated.

7 User and Developer Manual

7.1 Introduction

The ViT-shark-detection code-base contains code for training and evaluating Vision Transformer (ViT) models for shark classification. The models perform binary shark identification (SI), genus classification, and genus-specific species classification. Note that the binary identification and genus models have one model each, while the species model has one model for each **data-rich** genus; that is, a genus whose number of images exceeds a minimum threshold `GENUS_THRESH` and has at least one species whose number of images exceeds a minimum threshold `SPECIES_THRESH`.

This section contains usage information for model-training as well as general software organization and methodology. It is split into two subsections: one for before the refactor, and one after the refactor. The refactor aimed at centralizing ALL of shark-detector model training into a set of Python classes using OOP principles so that the code is maintainable and easy to run. It segregates training into a data-preparation stage, model-training stage, and a report generation stage. However, it fails in training models due to a bug pertaining to data-fetching.

Nonetheless, we believe it is important to continue working on the refactored version for the sake of the maintainability of this code-base.

Both the refactored code and the old code are available in the SharkPulse node on the Virginia Tech Computer Science network.

7.2 Usage (pre-refactor)

To run model training, you must first use one of the image loader scripts (see Figure 11).

The image loaders populate a directory of shark images segregated into folders by species or genera, depending on the image loader.

After an image loader has been run, use the train scripts to run model training.

7.3 Project Directory (pre-refactor)

```
csteam@sharkpulse.cs.vt.edu:~/SDv3/
├── train_SI_vit.py:  Trains Shark Identifier Model
├── train_gen_vit.py:  Trains Shark Genus Classifier Model
├── train_SP_vit.py:  Trains Shark Species Classifier Models
├── img_loader.py:  Loads images from dataset for training genus model
├── img_loaderSI.py:  Loads images from dataset for training SI model
└── species/
    └── img_loader.py:  Loads images from dataset for training species model
```

Figure 11: File Structure (pre-refactor)

The custom environment being used is a Linux based environment, which holds all the files required for training the model. This environment can be reached by making an ssh connection to the SharkPulse virtual machine. The files for training are stored in the folder SDv3. In order to run any of the files, the developer must be in a special conda environment, which can be activated via "conda activate vit".

7.4 Installation

To install the required packages, run:

```
pip install -r requirements.txt
```

This project used Python 3.10.10 for development.

7.5 Running the Code

The main entry point for training is `run_training.py`. It accepts command-line arguments to specify which models to train, the levels of the hierarchy to train, an optional output path to redirect output files, and an optional list of email addresses.

```
python run_training.py -m {vit,res,swin} -l {si,gen,spec} [-o OUTPATH] [-e EMAILS]
```

- `-m`, `--model`: Which model to train. Choose from 'vit' (ViT), 'res' (ResNeT), and 'swin' (Swin Transformer).
- `-l`, `--levels`: List of the levels of the hierarchy to train. Choose from 'si' (Binary Classifier), 'gen' (Genus Classifier), and 'spec' (Species Classifier). You can train multiple levels by providing a space-separated list, e.g., `-l si gen`.
- `-o`, `--outpath`: (Optional) Specify the output path for the trained models and other outputs. Defaults to `./out`.
- `-e`, `--emails`: (Optional) Provide a list of email addresses separated by space.

7.6 Example

To train the ViT model for Binary Classification and Genus Classification and save the output in the ./trained_models directory:

```
python run_training.py -m vit -l si gen -o ./trained_models
```

This will train the specified models using the default parameters. You can modify the parameters (e.g., image size, batch size, number of epochs) directly in the script if needed.

After training, the trained models and their corresponding logs will be saved in the specified output directory (i.e., `./trained_models` in this example).

7.7 Project Directory

```
csteam@sharkpulse.cs.vt.edu:~/SDv3/refactor/
├── data_utils.py:  provides utility functions for processing and manipulating data
├── Dockerfile:  used to build a Docker image for the project
├── frontend/
│   ├── textttapp.py:  a textttgradio-based webapp that uses textttFastAPI to serve model
│   │   calls
│   ├── Dockerfile:  used to build a Docker image for the frontend
│   ├── README.md:  contains information on how to run and use the frontend
│   └── requirements.txt:  lists the required Python packages for the frontend
├── out/:  a folder that contains output files generated by the project
├── report_utils.py:  provides utility functions for generating run output
├── requirements.txt:  lists the required Python packages for the project
├── resnet_model.py:  defines a ResNet model architecture for the project
├── resnet_training.py:  contains functions for training the ResNet model
├── run_training.py:  main entrypoint for running training
├── swin_model.py:  defines a Swin Transformer model architecture for the project
├── swin_training.py:  contains functions for training the Swin Transformer model
├── swintransformer.py:  provides a TensorFlow implementation of the Swin Transformer
├── test_util.py:  provides utility functions for testing the project
├── vit_model.py:  defines the Vision Transformer (ViT) model architecture for the
│   project
└── vit_training.py:  contains functions for training the ViT model
```

Figure 12: File Structure of Refactored Code

7.8 Build the Docker Image

The refactored code also comes with a Docker image to containerize your machine learning runs. To build the Docker image, run the following command in your terminal:

```
docker build -t shark-detection .
```

7.9 Run the Docker Container

To run the Docker container, run the following command in your terminal:

```
docker run -v ~/sd_images:/app/sd_images shark-detection
```

Replace `/sd_images` with the path to your `sd_images` directory.

7.10 Automated Training with CRON

Automated monthly training can be implemented by adding the following string to your server's `crontab`:

```
0 0 1 * * /usr/bin/python /path/to/script/run_training.py <args>
```

Note that it would be better to implement a script `check_data_influx.sh` that checks if the training data has grown sufficiently before executing the training script.

8 Lessons Learned

8.1 Timeline

See timeline table.

Date	Milestone
02/17/2023	Meet with Dr. Ferretti and SharkPulse team
03/03/2023	Fully understand code-base and plan design
03/17/2023	Merge new data for training model
03/31/2023	Improve previous model to ViT model
04/14/2023	Run the new model with new data
04/28/2023	Finalize model and data pipeline (CRON job)

Table 9: Timeline

8.2 Problems

The shark detection group ran into a few roadblocks along the way.

1. There were images in the SharkPulse database that hadn't been validated yet, were duplicated in other tables, or did not contain sharks at all.
2. After merging the new images, we were notified by the model team that the new data was not usable by the model as the img_loader script they used was expecting missing attributes (genus and species name).
3. The training would crash and run of out of memory unexpectedly. This was mostly fixed by tuning the model hyperparameters in such a way to account for the computational resources we were given, such as using a smaller-size ViT model or lowering the batch size.

8.3 Solutions

Solutions

1. The database team were tasked with parsing through the provided PSQL database to create a pipeline of new training data for the model to use. As mentioned in Section 8.2.1, the database contained a wide variety of images with attributes that needed to be organized and routed towards the training table called "training". To solve this issue, we created a PSQL query that collected records with unique image names and marked them as validated from the "instagram", "data_mining" (images scraped from Flickr), and "sharkpulse" (images uploaded to the SharkPulse website) tables. The records matching the query were then stored into a backup table we created called "training_backup", containing the records found in "training" as long as the img_name attribute was not also found in the table already. This solution had the correct intention, but ended up contributing to the next problem, discussed in Section 8.2.2.

2. After the merging process discussed in Section 8.2.1, the backup training table was attempted to be used for training by the model team. However, there was a glaring issue that the new images were not able to be loaded into the image folder that trained the model. After further investigation, it was discovered that the provided img_loader.py script required the img_name, genus, and species name in order to load the new images. We were able to resolve this issue by creating a new query. First we pull image names and species names from the "instagram", "data_mining", and "sharkpulse" tables where the validated attribute is true and store those into a new table called "training_backup_new". Then we applied an update to the genus column of the new table using the species names we previously pulled before. We are also attempting to make this process automatic through a CRON job.

8.4 Future Work

There remains more work to be done in the SharkPulse data-mining / machine-learning pipeline. Techniques to fetch a larger volume of shark data would likely greatly increase model accuracy. A similar paper, titled *A Swin Transformer-based model for mosquito species identification* (Zhao, et al.) [5], achieved 99% classification accuracy on mosquito identification. The main difference is their data–they had 1000 high-definition, professionally taken images of every single species in their study. Further, the model in the paper only identified 17 mosquito species. The findings of the mosquito study indicate that machine learning techniques are capable of extremely accurate species identification.

Assuming a minimum threshold of 1,000 pictures per species (a conservative estimate) for our model to achieve 99% accurate classification, it would require 521,000 images in order for an accurate classifier spanning the entire class of Chondrichthyes. At the moment the classifier is trained on 72,000 images. Clearly, the shark model's species breadth is dependent on the influx of new data from the pipeline. This, combined with how ViTs tend to perform much better than traditional image classification architectures when provided with much larger amounts of data, leads to the conclusion that our classifiers will scale well to data over time. Once shark-recognition is achieved, SharkPulse will be able to realize its purpose which is to automatically track shark populations using machine learning.

9 Acknowledgements

As a team, we, the Shark Detection group, would like to express our appreciation to all those who contributed to the successful completion of our project.

Special thanks to Dr. Feretti, Jeremy Jenrette, and all of the previous groups for their exceptional work on SharkPulse and previous detection models, which provided a strong foundation for our project. Their expertise and guidance were instrumental to our success.

We also extend our thanks to Dr. Fox for his support and assistance throughout the project.

Finally, we express our gratitude to all the individuals who contributed to the SharkPulse database by verifying labeled data. Their contributions enabled us to develop a robust and effective shark detection model.

10 References

References

[1] Jenrette, Jeremy; Liu, Z.Y.C.; Chimote, Pranav; Hastie, T.; Fox, E. and Ferretti, Francesco. (2022). Shark detection and classification with machine learning. Ecological Informatics. 69. 101673. 10.1016/j.ecoinf.2022.101673.

[2] Oceana. The Importance of Sharks. Oceana Europe. (2022, September 23). Retrieved March 2, 2023, from https://europe.oceana.org/importance-sharks-0/ (accessed March 20, 2023)

[3] World Wildlife Fund. (2023). Shark. WWF. Retrieved March 2, 2023, from https://www.worldwildlife.org/species/shark (accessed April 7, 2023)

[4] Hopkins, R. and Miller, K. L. (2021, March 9). Alarming decline in shark populations points to need for stricter conservation measures. The Pew Charitable Trusts. Retrieved March 2, 2023, from https://www.pewtrusts.org/en/research-and-analysis/articles/2021/03/09/alarming-decline-in-shark-populations-points-to-need-for-stricter-conservation-measures (accessed May 7, 2023)

[5] Zhao, Dz., Wang, Xk., Zhao, T. et al. A Swin Transformer-based model for mosquito species identification. Sci Rep 12, 18664 (2022). https://doi.org/10.1038/s41598-022-21017-6

[6] Kalbouneh, O., Marshburn, M., & Carroll, D. (2022). SharkValidatorGame. Retrieved from http://hdl.handle.net/10919/109976 (accessed April 27, 2023). Virginia Tech CS4624 team term project submission.

[7] Kothari, A., Patel, F., Raya, R., Shroff, T., & Tiwari, A. (2021). SharkPulse Validator Game. Retrieved from http://hdl.handle.net/10919/107016 (accessed April 27, 2023). Virginia Tech CS4624 team term project submission.

[8] Change, G., Debay, H., Gordon, S., Jenrette, J., Mulgrew, M., sharkPulse Validation Monitor. (2021). Available from http://hdl.handle.net/10919/103254 (accessed April 27, 2023). Virginia Tech CS4624 team term project submission.

A Methodology

This section is adapted from an assignment completed in the early stages of design.

1. The system needs to support two types of users: SharkPulse Users and SharkPulse Team. The goal of SharkPulse Users is to identify the species of shark in their image with high accuracy using the shark detector. On the other hand, the goal of SharkPulse Team is to have an improved shark detection model on SharkPulse.

    - SharkPulse Users: Be able to upload images to SharkPulse's shark detector to identify the species of shark in their image with high accuracy.
    - SharkPulse Team: Have an improved shark detection model on SharkPulse, have automated training available and customizable. Acceptance criteria:
      - Accuracy greater than 71%
      - Species breadth increased

2. To achieve the goals of the system, each goal is broken down into units of tasks and subtasks, which together make up the goal. The breakdown of goals into tasks and subtasks are as follows:

    - Data preparation: Scrape Internet data for shark images and metadata
      - Scrape new data
      - Join existing and scraped data
      - Preprocess data as necessary
    - Usable Shark Detector model: Detect sharks in images with more than 71% accuracy
      - Train detector model
      - Train binary classification model
      - Train granular classification model
      - Test system of 3 models
    - Develop automated training: Have training of Shark Detector Model automated
      - Create script that can train each model in sequence
      - Host script as part of CRON job running in SharkPulse database

3. Table 10 provides a list of the services required for the implementation of the solution. Each service has a unique ID and is associated with specific input data, libraries, and environments required for its implementation. For instance, the Scraper service (ID: 1A) requires internet access and the SharkPulse environment, whereas the Join Data service (ID: 1B) requires Pics data and the numpy library. The table also provides information on the output file produced by each service, and the other tasks producing that input file.

Service Name	**Service ID**	**Input**	**Libraries; Environments**
Scraper	1A	Internet	SharkPulse
Join Data	1B	Pics	numpy
Preprocess Data	1C	Joined data	numpy + OpenCV
Train Detector Model	2A	Processed Data	PyTorch
Train Binary Classification	2B	Shark Pics	PyTorch
Train Granular Classification	2C	Pics and Labels	PyTorch
Test System of 3 Models	2D	3 Models	PyTorch
Script that Runs Training	3A	3 Models	SharkPulse DB and Dev env.
Script cron-job	3B	Training Script	Bash

Table 10: Service Table

4. The system can be built as a set of workflows, which can be represented in a chart like the one shown below. To ensure that each goal is covered, a list of workflows was created. The service IDs in Figure 13 correspond to Table 10 entries.

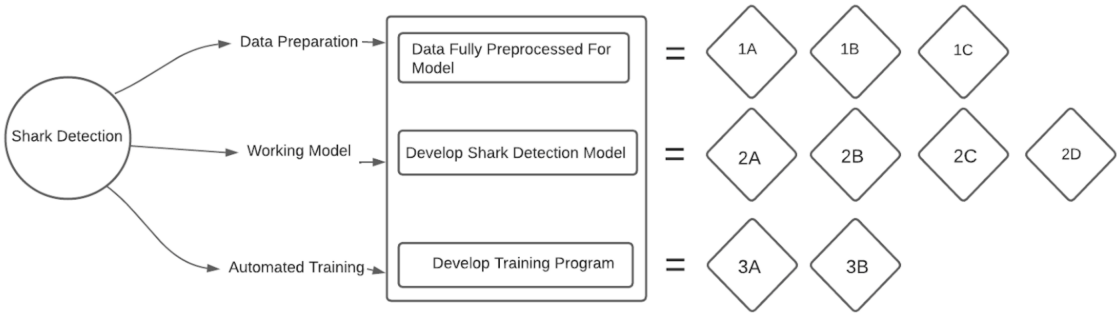


Figure 13: Workflow Chart