Shark Matcher: Automated, Image-based White Shark Identification

A Thesis
Submitted in partial fulfillment for the degree of
Bachelor of Science in Computer Science
to the Department of Computer Science and Engineering

University of California, Santa Cruz

by
Fabrice Kurmann
June 2024

Abstract

This senior thesis presents Shark Matcher, a tool for shark identification to aid marine biologists in sorting images of sharks by individual. Shark Matcher uses the dorsal fin as a visual biomarker with which to encode images for comparison in vector space. Using retrieval of vector embeddings informed by image metadata including capture location, time, date, shark sex, and shark length, Shark Matcher suggests image to shark matchings and allows for the management of data records by a lightweight user interface.

1 Introduction

Wildlife researchers capture videos and photos to track individuals they encounter in the wild for tracking population statistics and modeling individual behavior [1]. While they are easy to capture and require minimal equipment, distinguishing between individuals from just images requires trained researchers and remains a labor intensive process. Computer vision offers a promising solution: training a model to identify individuals automatically. Re-identification is the task of mapping image or video data to a specific individual and has been applied to both humans and wildlife [2]. Distinguishing features in appearance must be learned while confounding differences in background, image quality, and camera angle are de-emphasized. In the re-identification of great white sharks, the primary biomarker analyzed is the dorsal fin, which often displays unique coloration, shape, and notches along the edge [1].

This senior thesis outlines the key steps implemented for shark re-identification in Shark Matcher, a tool in development at the LINQS Lab at the University of California, Santa Cruz, to automate matching of image based data collected by marine biologists. Shark Matcher encodes fin images using a neural network consisting of a pre-trained foundation model and custom trained projection head for comparison in vector space. By retrieval of vector embeddings informed by image metadata including capture location, time, date, shark sex, and shark length, Shark Matcher suggests image to shark mappings and allows for the management of data records by a lightweight user interface.

I outline progress following 20 weeks of research development assisted and guided by my mentors at the LINQS lab. The following sections will detail the specific tasks targeted and overview the shark media dataset. Following is a detailed report on the underlying image encoder model designed and trained for the specific tasks outlined, along with the retrieval algorithm, which weighs embedding similarity by interpreting metadata in the context of shark biology and behavior. An accompanying analysis of performance testing contextualizes the motivations and architecture decisions. In the appendix, I outline related work, then introduce the Shark Matcher application, enabling interaction with the model to manage shark labeling data, with an underlying database developed for this task.

2 Task Overview

The primary focus of Shark Matcher is to identify the identity of the shark individual present in a fin image. This section continues with an overview of the two key sub-tasks Shark Matcher targets: 1) new image matching; 2) entity resolution and deduplication; both of which apply its general ability to map fin images to shark individuals to the practical needs of wildlife researchers.

2.1 New Image Matching

The first subtask is the matching of newly captured media against shark records already present in the database. Following the refinement of raw media to a fin image, the new fin image to be matched is the query fin image, q. Shark Matcher's embedding model translates q to a vector representation V(q). V(q) is now compared to the existing set of fin image vectors, evaluating distance between the V(q) and Set(V(fin images)). The nearest neighbor fin images, with biased weighting applied based on image metadata, are retrieved and returned as output for query q. The neighbor image retrieval process identifies fin images most similar to the query, an intermediary step in the end goal of shark re-identification. Shark Matcher uses a query q's nearest neighbor images to inform the prediction of most likely fin image to shark matches based on labeling of the aggregation of the identified nearest neighbor images.

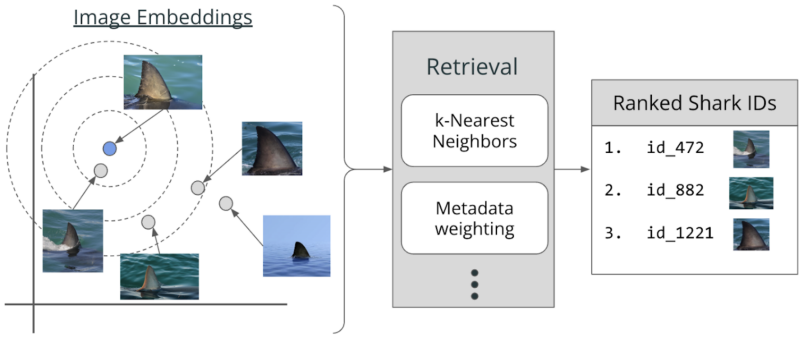

Figure 1: Diagram of the encoder model and retrieval algorithm, the two elements comprising the matching functionality in Shark Matcher.

2.2 Entity Resolution and Deduplication

The second subtask is entity resolution and deduplication where existing labeling of fin images and sharks is examined and updated. The motivation for this capability stems from the manual matching previously used, where matching was performed within a year and then disjoint sets of individually matched years

were merged with the combined archive of previous years. Duplicate references stemming from matching across different subsets of the total data should easily be identified and addressed using this functionality.

3 Data

Data used during the development of Shark Matcher comes from the Dr. Barbara Block Lab at Stanford University's Hopkins Marine Station, which has employed photo and video tracking of white sharks on the central California coast and built up a database of records consisting of fin images, shark individuals, and image to shark mappings dating from 2006 through 2024. Image data encompasses all data resulting from the photography and filming of sharks including all corresponding metadata. It is worth noting is that an additional source of shark tracking data stems from the use of satellite and acoustic tags. This data is referred to as tag data and, while not currently utilized, may be referenced in future Shark Matcher versions.

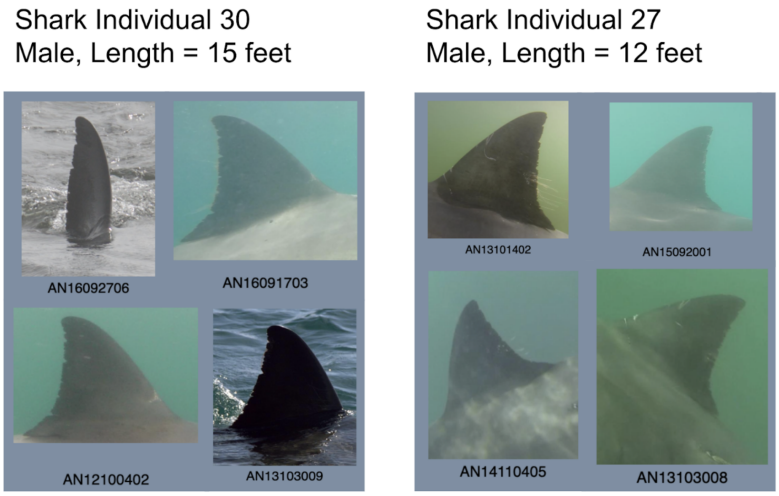

Figure 2: Examples of two shark individuals, 30 and 27 with 4 fin images matched to each.

Shark Matcher matches sharks by comparing fin images, static images stemming from crops of original raw media. In cases where raw media is in video format, an individual frame is first isolated. Video frames and field data images are then cropped to show only the dorsal fin and a narrow surrounding border. Each fin image is named following a format that begins with a location reference - a two or three character abbreviation for capture location - followed by a date reference (yymmdd), followed by a numerical identifier for the image from the particular day (two digit value). In addition to location and temporal

metadata, fin images, when the information is available, labeled with the sex and approximate length of the pictured animal.

For the model development process described in this thesis, we had access to a total of 3140 fin images, of which 2827 were labeled by the Block Lab's marine biologists with a mapping to a shark individual. Based on these mappings there were 1031 distinct shark individuals present in the dataset. A majority of the shark individuals present in data contain only a single or very few referencing fin images. Conversely, a small subset of shark individuals have been encountered and photographed repeatedly both within and across years, with the most commonly encountered shark having 62 associated fin images ranging over an 12 year period. This skew is further investigated during the overview of model optimizations. The dataset is split into training and validation sets by randomly designating one fin image per shark as a validation image for all sharks with greater than 1 associated fin image.

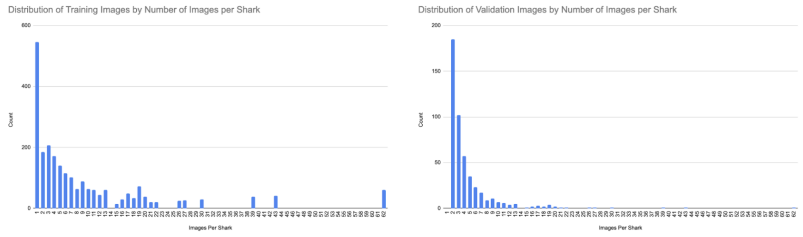

Figure 3: Visualization of the distribution for total quantity fin images (y) sorted by sharks with various associated fin images (x) for training and validation sets, respectively.

To contextualize encoder model evaluation in the following section, we computed hits@1 scores when randomly selecting shark identity in both the training and validation sets to produce a baseline metric. These random selection scores reflect the probability of matching the query shark given the counts and identities of images to choose from in the training and validation datasets, respectively.

	Hits@1
Train	0.0078
Validation	0.0021

Table 1: Hits@1 for random matching

4 Image Encoder Model

At the center of Shark Matcher is the image encoder model which translates fin images to vector embeddings that aim to reflect shark identity. During the

design of the encoder model, we experimented to find optimizations to handle three distinguishing challenges experienced: 1) a limited quantity of training data; 2) the aforementioned imbalance in training data per shark; 3) overfitting to training data. This section overviews the architecture and training of Shark Matcher's image encoder model and its evolution in the context of addressing these challenges. It begins by highlighting our initial simCLR based model before detailing the later Swin transformer model and optimizations experimented with in data re-balancing, image augmentation, and architecture tuning.

4.1 SimCLR Model

Given the relatively small scale of our dataset in the context of computer vision model training, we prioritized an architecture and learning approach which would efficiently learn the distinguishing features within training images. These considerations motivated us to leverage the simCLR framework [3]
which combines self-supervised contrastive learning with the use of several data augmentations to maximize learning on compact datasets.

Data augmentation is the practice of producing altered copies of original training images and serves the dual purposes of increasing the volume of training data and directing the emphasis to and away from specific image qualities. Informed by manual matching strategies of the Block Lab's marine biologists, we sought to de-emphasize the matching weight of image qualities that bear minimal relation to the identity of the included shark. Identified qualities and corresponding augmentations to address them include image coloration (colorJitter), which may differ greatly based on weather conditions at capture time and the camera used. Additional qualities that merit augmentation for similar reasons include the left to right orientation of the fin in the image (randomHorizontalFlip), the captured angle between the face of the fin and the camera (randomPerspective), and the rotation of the fin in the image (randomAffine).

To produce a training set, two augmentations were produced per fin image. The set of augmentations [colorJitter, randomHorizontalFlip, randomPerspective, randomAffine] were randomly applied with default probabilities. Resulting augmentations and training images were then used to fine tune a ResNet 34 layer convolutional neural network. This model encodes images into 512 dimensional vector embeddings which are compared by their cosine distances. We refer to this simCLR trained ResNet model as version 0.0.1.

Training over 3000 epochs with this architecture, we were able to observe a convergence in loss, especially with more commonly pictured sharks. When filtering evaluation to only sharks with greater than five associated fin images, our hits@10 value peaked at 0.99 and hits@1 at 0.94 for training images. Conversely, training seemed to have little effect on validation hits@k scores, regardless of k value or filtering based on only more prevalent sharks. Best achieved hits@10 scores were 0.06 and 0.09 across all sharks and only those with greater than two associated fin images, respectively. While these results are a significant improvement over random guessing, examining the training convergence curve shows a very flat growth curve for this metric.

	Hits@k	v0.0.1
Train	1	0.75
	10	0.78
	25	0.79
Validation	1	0.02
	10	0.04
	25	0.06

Table 2: Hits@k Results

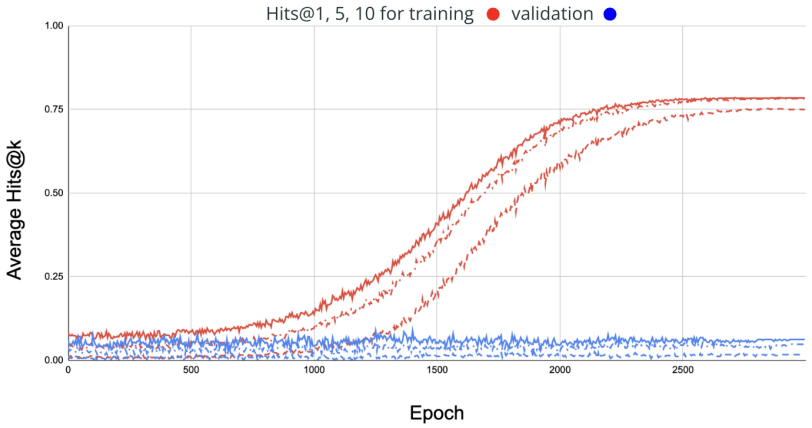

Figure 4: Training convergence over 3000 epochs of training our simCLR based resNet model.

4.2 Swin Model

Evaluating the low scoring validation results for the simCLR model established the need for more generalized learning - difficult given the size limitation of our dataset. The use of general purpose foundation models has emerged as a practice in computer vision to address the considerable magnitude of required training data by leveraging models pre-trained on generalized image datasets. The pre-trained foundation model is then trained for specificity on a more specialized and compact dataset to produce an optimized version for the desired task. In our case, we selected the Swin Transformer model [4], a general purpose backbone model for computer vision trained on 14.2 million images across 22,000 classes in the imageNet library. To customize the Swin model to our shark fin dataset, we train a specialized MLP projection head on top of this foundation using our dataset of fin images. We refer to this new model style as version 0.1.0.

The Swin transformer model addresses the tasks of image classification and recognition by including self attention connections over localized windows to reduce computation cost over large image inputs. Localized windows are addi-

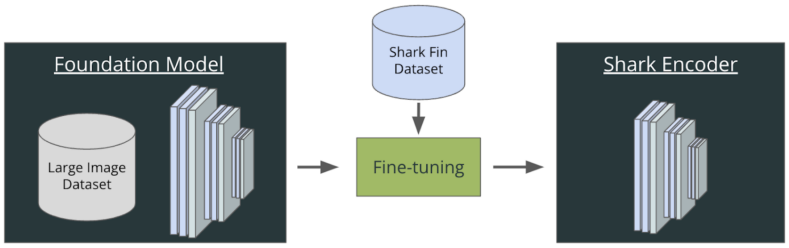

Figure 5: Diagram showing the pre-trained foundation model, trained on external data, and the custom trained projection head combined to become the shark encoder.

tionally shifted to allow for overlaps in the isolated self attention connections [4]. This allows for a high parameter transformer architecture to be applied with feasible computational cost in place of a smaller convolutional neural network as employed in the previous simCLR model. We saw most success using the Swin Base Model with a 37632 dimensional output. We then also experimented with multiple dual layer projection head architectures, settling on a 4096 dimensional input layer and 2048 dimensional output layer, a larger size than with the previous model to preserve more of the information produced by the transformer model.

	Hits@k	**v0.0.1**	**v0.1.0**
Train	1	0.75	0.08
	10	0.78	0.31
	25	0.79	0.45
Validation	1	0.02	0.02
	10	0.04	0.11
	25	0.06	0.19

Table 3: Hits@k Results

Initial experiments switching to a Swin based model and training our projection head over 500 epochs yielded improvements significant improvement in validation. Conversely, training scores saw a decrease, implying that we were not able to customize the behavior of the foundation model enough with our current volume of training data. To then encourage more shark specific learning and provide more training data we began producing 10, instead of two augmented images per original training image.Furthermore, while validation results were improved after switching to our Swin model, overfitting to seen training data was still a clear issue which we aimed to address with the addition of a probability 0.2 dropout. Following these changes, we began to see significant im-

provements in performance, with training hits@k scores more than doubling and validation results also showing proportionally even larger improvements: hits@1 increasing from 0.01 to 0.07, hits@10 reaching 0.22, and hits@25 reaching 0.36. The updated Swin model with dropout and 10 augmentations is referred to as version 0.1.1.

4.3 Data Re-balancing Optimizations

Following the redesign in architecture to address the limited quantity of task specific training data available, we aimed to address the data's imbalance, focusing on minimizing the difference in representation across different shark individuals. With our strategy of 10 augmentations per original image, the model is trained on as many as 682 training instances for the most represented shark individual (having 62 matched fin images) and as few as 11 training instances for all sharks with only one matched fin image.

To address this imbalance, we experimented with a variable number of augmentations based on the number of reference images for the shark.

$$augmentations = \min\left(\lfloor maxCount/sharkImages \rfloor * minAug\right), maxAug \quad (1)$$

Applying the balancing equation to each image based the number of reference images for the most referenced shark (maxCount) divided by the number of images its associated shark has (sharkImages) produces more augmentations for under-represented sharks where the divisor is less. Two tun-able parameters are maintained, minimum augmentations per image (minAug) and maximum augmentations per image (maxAug), which set a floor and ceiling to how many augmentations are produced (augmentations). Experimenting with these parameters to decrease the delta between maximum and minimum training instance count without resulting in a large increase in the total number of augmentations produced led to us selecting minAug and maxAug values of 4 and 20, respectively. This ultimately reduced the spread of total images (original + augmentations) from 11 - 682 to 22 - 310, a notably smaller range.

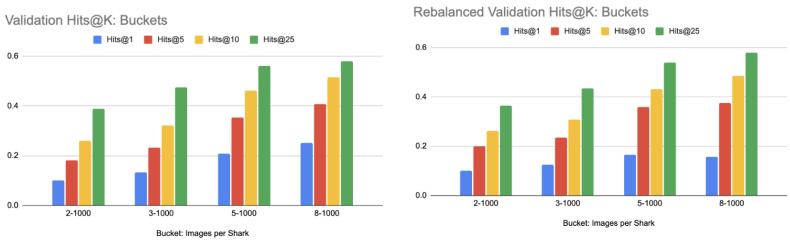

Figure 6: Bucketing results for the encoder model with a constant 10 augmentations per training image compared with variable augmentations to re-balance the dataset.

To better understand the impact of re-balancing, we sorted our hits@k metric by the number of fin images available per shark. In the most general bucket, all validation data is included, as validation images are only produced for sharks with two or more matched fin images. We then created buckets containing only those images matched to sharks with three, five, and eight matched fin images, respectively. As expected due to a larger number of training images, results increased as the buckets filtered more underrepresented sharks. Evaluating a model trained on the re-balanced dataset (version 0.1.2), we observed a small increase in fairness between the performance of lightly and heavily represented sharks with marginal increases in validation accuracy for the most general 2-1000 image per shark bucket at the lowest k values of one and five. However, the more noticeable impact was a drop in overall performance in all other buckets at all other k values, which led us to continue with uniform augmentations for subsequent testing.

	Hits@k	**2-1000**	**8-1000**
Constant Augmentations	1	0.09	0.25
	10	0.26	0.52
	25	0.39	0.58
Rebalanced Augmentations	1	0.10	0.16
	10	0.26	0.48
	25	0.36	0.58

Table 4: Bucketed Hits@K Results

4.4 Augmentation and Architecture Optimizations

To explore additional avenues for improvement, we began to assess our model's learning by exploring different image similarity predictions it produced. Assessing performance in this way allowed us to better understand the specific image traits that seemed to be influencing predictions, either to the benefit or deficit of accurate matching. Such visual inspection affirmed the need for additional experimentation with augmentation to divert matching decisions away from misleading image similarities. In particular, we conducted experiments manipulating the augmentation set while also adjusting the probability of augmentations. An augmentation that had been brought to our attention was randomErasing, which, by removing all information from a portion of the image, encourages a focus on more granular details, which we hoped would translate to better learning of distinctive features on the fin edge. This experimentation revealed an increase in performance with the addition of each augmentation and best results when including all augmentation methods with increased probabilities ranging from 0.5 to 0.75.

An additional variable we experimented with optimizing based on our investigations was dropout rate of our projection head together with training epoch

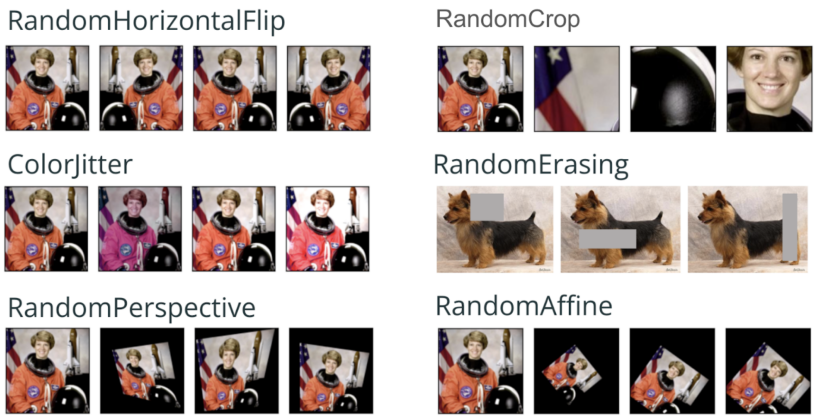

Figure 7: Visualization of the augmentations experimented with and selected for use in training the shark encoder.

count. Having seen a significant decrease in overfitting by adding a dropout with 0.2 probability, we experimented with rates of 0.5 and 0.3, both of which resulted in decreased performance while increasing the number of training epochs for loss convergence from 500 to 1000. Further experimentation to understand the architecture changes that may be needed to benefit from increased dropout are pending.

	Hits@k	**v0.0.1**	**v0.1.0**	**v0.1.1**	**v0.1.2**
Train	1	0.75	0.08	0.58	0.61
	10	0.78	0.31	0.84	0.87
	25	0.79	0.45	0.91	0.95
Validation	1	0.02	0.02	0.07	0.07
	10	0.04	0.11	0.22	0.23
	25	0.06	0.19	0.36	0.34

Table 5: Hits@k Results

A final variable we studied was the dimensionality of the foundation model and projection head MLP. The Swim transformer is available in multiple size configurations and switching from the Swin base model with 88 million parameters to the tiny model with 28 million did lead to the training time decrease we were targeting, but at the expense of significant performance in the trained model. Similarly, we found that to account for the output dimension (37632) of the base foundation model, be benefited from a larger MLP. We ultimately achieved best success to date training with 10 augmentations per image across the selected set of augmentations, and including a dropout rate of 0.2 using the Swin base model as our foundation with 4096 * 2048, our largest experimental

size, parameter projection head atop. This configuration matches that of version 0.1.1, the strongest performing model we have trained at the time of this writing.

5 Retrieval

Following the generation of embeddings for all query images and the dataset of images to compare against, embeddings are compared and best match results are selected in the retrieval process. For the duration of the embedding model testing outlined above, we employed a nearest neighbor based retrieval algorithm, directly scoring embeddings produced by the trained model by the linear distance between them. This method disregards a significant amount of metadata available to assist in matching. All media contains metadata that includes the capture time and location and a majority of more recent media also includes a shark length and sex prediction. Applying a weighting algorithm to favor matching results whose metadata values equal those of a query image can provide a significant increase in Shark Matcher's utility. With an ability to devalue the matching potential of sharks where metadata conflicts, false positive matching suggestions can be minimized in the returned results and a greater number of plausible matches can be returned within a fixed k value. This is directly relevant as Shark Matcher label suggestions are still approved, rejected, and committed manually by marine researchers.

Weighting matching predictions on discrete, high confidence metrics such as sex can be compared almost to filtering results, where the metadata value has a strong effect on validity of results. Other metrics such as location require a more nuanced weighting method: if a shark is observed at a location such as Año Nuevo, there is a small probability that it can be observed at Tomales Point the following day, but a fair probability that it could be seen at this location within the span of one month. Mirroring these behavioral probabilities requires tuning of the retrieval algorithms informed by shark behavioral patterns, an ongoing effort between us and the Block Lab's shark researchers.

6 Conclusion and Discussion

During its development we have improved Shark Matcher in its task of understanding and isolating the distinguishing qualities of fin images to aid in mapping fin images to shark entities. Developing the image encoder model so far has yielded multiple key takeaways relating to the selection and tuning of a model architecture and the preparation and augmentation of data. Clear improvements were achieved when leveraging a large scale, general purpose foundation model, which dramatically improved in performance of shark specific matching after a custom projection head was trained on an adequate size of augmented and original data, in our case 2827 labeled images each with 10 augmentations. Applying dropout of rate 0.2 greatly reduced overfitting without compromising

learning efficiency and achieved accuracy. Supplementing embedding representations with retrieval that accounts for biologically relevant constraints further improves matching utility and is a hybrid approach we are interested in investigating further.

6.1 Future Work

To realize the full workflow optimizations possible by leveraging an automated shark matching model, both the embedding generation and retrieval algorithms, along with the accompanying interface must be optimized for the specific tasks targeted by its users. We will continue experiments targeting model optimizations, focusing next on training with more regulated positive and negative labeling to maximize effective reinforcement of both similarity and difference traits in each training batch. Further we will rely on increasing evaluation of results with the context of our metadata informed retrieval algorithm. Continuing model experiments alongside the metadata informed retrieval algorithm will allow for the most accurate and efficient tuning of our model as this is ultimately the context in which it will be applied.

To maximize its utility to the workflow of marine researchers at the Block lab, Shark Matcher must grow outwards from its central encoder model to address inefficiencies both in the process of isolating quality fin images from raw media and in the storing and management of created labels. To this end, a database for storing matching information paired with an interface with write and commit workflows will enable Shark Matcher to become a fully functional shark matching and database management tool.

7 Acknowledgements

This senior thesis report documents my experience and the work of Charles Dickens, Connor Pryor, and Eriq Augustine, and myself led by Professor Lise Getoor. The opportunity to participate in a research project and complete a senior thesis was a long term goal of mine and I'm grateful for having been given the opportunity to participate in this project.

The Shark Matcher project is a collaborative effort between Stanford's Block Lab at Hopkins Marine Station and the LINQS Lab at UCSC. The Block Lab's data and expert feedback on matching predictions has made the project possible.

References

[1] S. Andrzejaczek, T. K. Chapple, S. J. Jorgensen, S. D. Anderson, M. Castleton, P. E. Kanive, T. D. White, and B. A. Block, "Multi-decadal high-resolution data reveal the cryptic vertical movement patterns of a large marine predator along the californian coast," *Frontiers in Marine Science*, vol. 9, no. 835576, 2022.

[2] J. Parham, J. Crall, D. Rubenstein, J. Holmberg, T. Berger-Wolf, , and C. Stewart, "An animal detection pipeline for identification," *International Journal of Modern Physics B*, 2018.

[3] C. Ting, K. Simon, N. Mohammad, and H. Geoffrey, "A simple framework for contrastive learning of visual representations," *International conference on machine learning*, pp. 1597–1607, 2020.

[4] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Z. Stephen, and L. B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10 012–10 022, 2021.

[5] J. W. Thompson, T. R. Speakman, T. L. McDonald, V. H. Zero, L. H. Schwacke, J. S. Morey, and B. M. Quigley, "finfindr: Automated recognition and identification of marine mammal dorsal fins using residual convolutional neural networks," *Marine Mammal Science*, vol. 10, no. 1111, 2020.

[6] C. Bergler, A. Gebhard, J. R. Towers, L. Butyrev, G. J. Sutton, T. J. H. Shaw, A. Maier, and E. Nöth, "Fin-print a fully-automated multi-stage deep-learning-based framework for the individual recognition of killer whales," *Nature Scientific Reports*, vol. 11, no. 23480, 2021.

8 Appendix

8.1 Related Work

Computer vision tools to identify and match individuals have been implemented for many different species and incorporated into software that can be accessed via an online account or by running the tool locally. WildMe is the largest, most general purpose software offering matching tools and a database for researchers across different focuses. The nonprofit run service has built a matching engine by merging species specific matching tools to cover matching for 53 species at the time of this writing [2]. The algorithms employed are based on seven models, each specializing in different identifying features including fur patterns for species such as zebras or fin tracing for large marine species. The marine matching for shark fins used by WildMe is an edge tracing tool: finFindR [5]. FinFindR is a matching tool for image to individual matching intended for bottlenose dolphins based on the curve of the trailing edge of the pictured dorsal fin. The tool utilizes a static, pre-trained model to locate and trace the trailing edge of a dorsal fin within an image and then capture the qualities of the curve in a vector embedding. This model is trained on 17046 labeled images of bottlenose dolphin fins. Vectorized image embeddings are then compared using euclidean distance to identify the most similar fin edges and by extension the most likely shark match. The image to vector focus and nearest neighbor based classification is a set of traits we also made central to our initial and evolved model due to its intuitive behavior and therefore approachable option for tuning and optimization. We deviated from a curve based vectorization as done by FinFindR in an effort to both remove pre-processing of images and allow the training a more free and generalized model, able to reason on all areas and traits of the images it is provided with. FIN-PRINT, a killer whale matching tool developed by Bergler et al [6]. Designed to handle large bulk uploads of unfiltered image data, FIN-PRINT leverages a strictly defined 4 step pipeline to pre-process image data and match contained individuals. As with FinFindR, FIN-PRINT narrows it's matching decision to a defined subset of the image's data, in this case using the entire area covered by the dorsal fin and the nearby saddle patch, a distinctly shaped white section of skin located in proximity to the dorsal fin of killer whales. Unlike FinFindR, FIN-PRINT's pre-trained model is then locally tuned based on its end user's dataset. This process significantly increases computational cost, time, and complexity for marine researchers using the tool with a potential but untested advantage to final results.

8.2 User Interface

A key element of Shark Matcher is to provide an interface to allow wildlife researchers to leverage the underlying computer vision and retrieval architecture to more efficiently and consistently manage their datasets. Given the ongoing development of the Shark Matcher project beyond the submission of this thesis, the user interface design includes areas to add functionality to be implemented.

Based on discussion with the Block Lab's marine researchers, the software should be operating system agnostic and free of installation procedures or intensive local computation. To address this requirement we selected a browser based approach with a web application currently run on a LINQS host server. In addition to the application the database and accompanying files are also hosted by the server, easing both implementation and user experience by avoiding data organization and sharing as a dependency. Shark Matcher connects a Python backend with a Python server middleware to handle API requests, and a basic single page application frontend.

To reflect the two sub-tasks of focus discussed, the interface is broken into tabs allowing for: 1) new image matching (Find Similar Images tab); 2) entity resolution and deduplication (Show Known Shark tab).

Within the Find Similar Images tab, a user is able to enter a query image and numerous constraint filters as discussed in the section on retrieval. The nearest neighbor images for the query matching any filter conditions are returned with their respective labels and quantified distance from the query image, a gauge of matching confidence. Within the Show Known Shark tab, a user is able to enter a query shark and view all images that have been labeled to match this shark in order to better understand the current state of the database and its stored matching information.

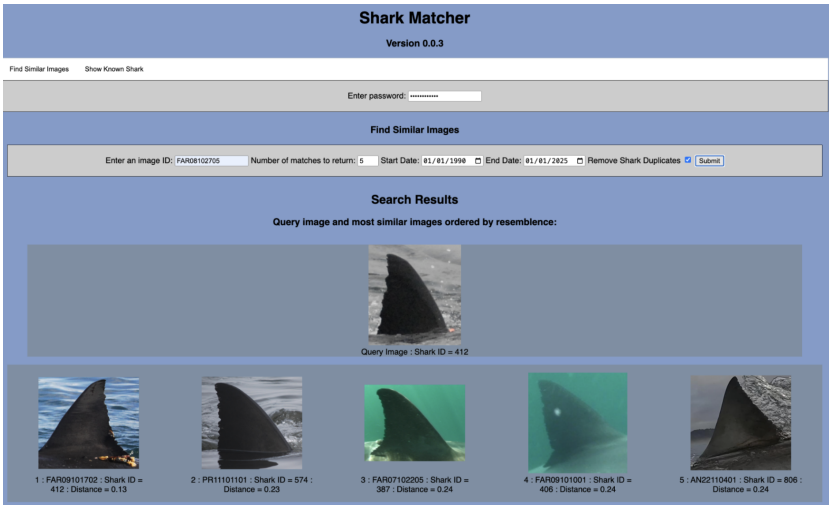

Figure 8: Example of the Find Similar Images tab showing the five closest matching fin images for a query given start data and end date constraints.

8.3 Database

To support both sub-tasks of focus, a well structured database is needed. The motivation for schema design of the Shark Matcher database is to provide a con-

sistent interface to read and modify data of different types including raw media, fin images, shark individuals, and records of mappings across these types. Additionally, traceability, especially for mappings such as shark to image or shark to shark matches is required to support write operations not causing cascading, unwanted side effects. This section provides an overview of the database architecture and schema paired with reasoning behind the design contextualized to support Shark Matcher functionality.

Shark Matcher employs a SQLite database which stores references to raw media and fin images alongside all data storing shark individuals, tags, image to shark mappings, and Shark Matcher users. Following is an overview of the database architecture in the form of a schema overview.

8.3.1 Media

A media table contains a record for every media instance stored, including videos, saved frames of videos, and cropped images taken from frames. Media attributes are a unique ID, a unique hash, an optional name, an optional source, optional associated metadata including capture date and location and pictured shark sex and length, and a media type.

To maintain a lightweight database, only references to media are stored. Media's hash attribute stores a hash of the media record's contents - duplicates are forbidden. In addition to duplicate detection, hashing enables fair distribution of referenced media files into a file system with directories sorted by the leading character(s) of a record's hash.

In the case where a media file is produced by refining another media file, such as a frame from a video, the source attribute lists the parent media file. Following this relationship, raw media files do not have a source attribute while all media files that are not raw files can be traced in one or multiple steps to a raw media file.

8.3.2 fin images

The fin images table stores a record for every fin image, the cropped images encoded by the model and used for matching. fin image attributes are a unique ID and a source where all fin images reference a media item as their source.

8.3.3 Sharks

The sharks table stores a record for every shark individual with attributes of a unique ID in the form of an integer and optionally the shark's sex and length.

8.3.4 AlternateNames

The alternateNames table stores records for any name mapped to a shark entity. The table stores records with an unique numerical ID and reference to the shark table record the alternate name maps to. Additionally, the alternate name itself and the type of the alternate name is stored. Alternate names

types may include nicknames for the shark or names originating from the Block lab's naming convention where a shark is named to match the name of the chronologically oldest fin image it which it appears in. It is worth noting that though this naming convention is used by the Block Lab, Shark Matcher uses the ID attribute in the sharks table as the primary key for shark entities.

8.3.5 Tags

The tags table stores records for each deployment of a tag. A deployment of a tag is distinct from a tag because tags may be deployed, recovered, and deployed onto a different shark repeatedly such as if tag 001 is placed on shark three and then one year later recovered and placed on shark four. For this example two separate records are stored. Records include a unique numerical identifier, a mapping to the shark the tag is deployed to and name and type fields to store the identity of the tag where type references the tag types detailed in the discussion of tag data.

8.3.6 SharkToImageMatches

The sharkToImageMatches table stores records for each matching of a shark entity to a fin image. Records contain a unique numerical identifier along with references to the shark entity and fin image entity being matched.

8.3.7 Users

The users table stores records to hold information on user accounts for the Shark Matcher tool. This is disjoint from the direct task of shark matching. Users stores attributes for a unique numerical identifier, the name, email, a unique username, a role, password hash, and salt for every record.