Sloop: Pattern Retrieval System for Animal Biometrics

by

Navi Tansaraviput

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Author .....
Department of Electrical Engineering and Computer Science
September 12, 2016

Certified by .....
Srinivas Ravela
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

Sloop: Pattern Retrieval System for Animal Biometrics

by

Navi Tansaraviput

Submitted to the Department of Electrical Engineering and Computer Science
on September 12, 2016, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

The ability to identify individual animals is crucial for non-invasive ecological monitoring and conservation planning. This project proposed two improvements to the recognition process and ranked retrival of Sloop, the first image retrieval engine that couples automated pattern recognition with crowd-sourced relevance feedback for individual animal identification. With a crowd-sourc‘ed relevance feedback simulator, we report a number of studies corroborating the acceration of precision and recall of the retreival results after various rounds of relevance feedback and the effects of the error propagation. Then, we describe Sloop MTurk, the crowdsourced relevance feedback integration of Sloop. In the later part, we propose a new architecture for animal pattern recognition, which could possibly reduce the system necessity for human involvement in feature extraction using transfer learning. Then, we experiment with a variety of binary classifiers in order to identify the algorithm that accomplish good performance on our data. Our results reveal that Convolutional network with linear support vector machine with radial basis kernel function (SVM-RBF) achieves a very robust performance on Otago and Grand data.

Thesis Supervisor: Srinivas Ravela
Title: Principal Research Scientist

Acknowledgments

This thesis would not have been possible without the support and help from many individuals to whom I am sincerely appreciated.

First, I would like to express my deepest gratitude to my advisor, Dr. Srinivas Ravela, for his expert guidance, profound understanding, and encouragement throughout my study and research. Without his counsel and great patience, this work would truly not have been possible.

I would also like to extend my greatest appreciation to Randy Westlund for providing indispensable advises, insights, and invaluable guidence in every aspect of my research, not to mention how much his engineering passion and enthusiasm have inspired me. His willingness to generously spend time discussing and unceasing help were i ndeedimperative to the completion of this thesis.

Additionally, I would like to take this opportunity to thank my family for thier unequivocal supports for which my mere expression of thanks likewise does not suffice.

This research is funded by New Zealand Department of Conservation (DOC).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unbiased information about animal population ecology are crucial to the development of effective conservation strategies for rare and endangered animals. The ability to identify an individual animal by recognizing its photographs allows researchers to monitor the species' diversity, and dispersal in a non-invasive and inexpensive way. Researchers can track the movements and observe the genetic variation of a species by comparing each member's images with the all the existing images collected from different time and locations. Typically, this requires the arduous work of manually comparing over a thousand images of every individual animal and their potential matches.

This project involves designing and implementing two additional features to Sloop, an existing pattern retrieval engine for individual animal identification. The features include:

1. Relevance Feedback Integration

2. Convolutional Neural Network Integration

1.1 Problem Statements

1.1.1 Relevance Feedback

One problem of interest is assigning tasks to workers with the goal of maximizing the quality of completed tasks at a low prices or subject to budget constraints.

1.1.2 Image Recognition

The emergence of convolutional neural network pushes forward the frontiers of all domains of computer vision [8]. Recent studies shows that convolutional neural network architecture clearly dominates the handcrafted features, and traditional orientation-based local descriptors, such as SIFT[9], and SUFT[1], etc. in classification tasks[5, 7].

The second part of the thesis presents a new architecture whose goal is to improve the identification accuracy as well as curtail or eliminate human involvement. We compare the recognition ability of the two algorithms: SIFT and convolutional neural network.

1.2 Challenges

The animal identification task involves two major challenges: *image feature extraction problem*, and *pattern recognition problem*.

Image feature extraction problem involves locating the animal in the image, and extract the features required for the matching. The choice of feature selection varies from species to species. The extracted feature object of an image is then passed into the pattern recognition algorithm to find the matches existed in the system. Current Sloop locates the animal and necessary features of an image by having the user click on the key points of the animal, and then calculates the feature vector using SIFT. It performs the matching by scoring the similarity of the SIFT object.

Traditional approaches in machine learning generally require training samples to be available for all the categories. Moreover, such approaches are designed to handle

only the dataset with finite or limited, preferably small, number of categories. Nevertheless, our application requires ability to recognize high dimensional input, whose categories are not known in advance. In addition, while the number of categories can be very large, the number of examples per category can be very small.

Chapter 2

Related Work

2.1 Sloop

2.1.1 Information Retrieval System

The field of information retrieval emerges from the attempt to provide information access. From the academic perspective, information retrieval (IR) is a principled approach of finding desired materials of an *unstructured* nature from within large collections. The fact that it allows more flexible query operations makes IR a dominant form of information access in practice, compare to database-style searching. Additionally, IR supports ranked retrieval, where it outputs the best answers given a query.

The preceding properties make information retrieval an obvious solution for animal identification task. Tracking population and dispersal of a species from images requires manually identifying all the individuals animals in the images. Traditionally, this involves an arduous work of comparing thousands of images. However, with Sloop, a pattern retrieval engine that can preprocess the images and output an initial possible ranking, the time required to spend on going through all the image pairs one by one could be reduced by an order of magnitude.

2.1.2 Sloop Architecture

Sloop is a distributed image retrieval system. The system is divided into two major components:

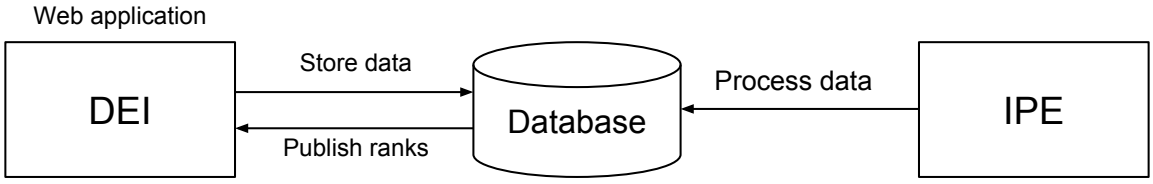


Figure 2-1: Score Distribution

1. Data Exchange and Interaction (DEI) DEI is a web application that provides the user inferface that allows biologists to upload data onto sloop databases.

2. Image Processing Engine (IPE) IPE processes the data stored in the databases and generate descriptors that represent identities of the images.

Sloop identifies each animal on individual base. This provides the similarity ranking of the images, within the same species, in the database relative to the animal in a given image A. Effectiveness of such identification system heavily depends on the choice of features on which the machine learning algorithms are applied.

Current Sloop uses Scale Invariant Feature Transform (SIFT) [9] to perform such images ranking. Given an image and the four fiducial key points annotated by the biologists, Sloop transforms the images into SIFT object and compare it among the SIFTs of the existing images stored in the database, using Euclidean distance. After the ranking is generated, the biologists then look at the top matches and confirm whether the given pairs of images are matches or non-matches.

While SIFT is still interesting for tasks that speed and simplicity are of major concerns, it requires tremendous amount of manual effort and training. Recent studies and competition results have proved that features learned via convolutional neural networks (CNN) outperform previous descriptors, includign SIFT, on classification tasks by a wide margin[5]. Therefore, replacing SIFT with convolutional neural networks seem to be a viable improvement to the system.

2.2 Relevance Feedback

Relvance Feedback is a technique that involves users in the retrival process. Specifically, given a query that returns a set of initial results, the system takes user feedback on the initial results to improve the results returned from the later iterations when given the same or related query [10].

2.2.1 Crowdsourcing

Crowdsourcing market is a platform takes advantage of collective intelligence of the online community. Crowdsourcing has gained popularity as an inexpensive and efficient method to accomplish certain tasks that are usually difficult for machines alone to complete.

In a crowdsourcing market, there are three parties involved:

1. Workers
2. Requesters
3. Crowdsourcing platform

Requesters submit tasks with the amount of *reward* that they are willing to pay *workers* upon the completion of the tasks. Some workers maybe better than others at certain tasks. In other words, some tasks maybe more difficult than others for some workers. The platform provides the environment for the worker and requesters to interact. All parties gain more information about one another and the tasks, and make repeated decision overtime.

2.3 Image Recognition

2.3.1 Convolutional Neural Network

Convolutional Neural Network is a special kind of multi-layer neural networks, whose task is specialized to capture and encode visual patterns directly from raw pixel

images [8].

2.3.2 Face Verification and Siamese Network

The problem of finding matching image pairs from the database given an input image, can be reduced to following *decision problem*:

> Given two images A and B, is the animal in image A the same individual as the animal in image B?

Our problem is somewhat similar to the face verification problem, which involves accepting or rejecting an identity claim based on the image of a human face. There are two major differences between animal identification and face verification. First, the animal pattern has finer-grained details and some can be very subtle that they fuse into the background. Second, the pattern of each individual does not share same overall structure as human face does. This is similar to identifying an identical twin by their blemishes except for, in this case, we have very high-order of multiple births.

Most current face verification methods use hand-crafted features, which are often combined to improve the validation performance. In [3], Chopra presented a new framework, *Siamese Network*, as a solution to this problem. He proposed a training method that is used to learn a similarity metric from the data with the contrastive loss function, comprised of the sum of the *magnitude of the difference between the features vectors* of the incorrect guesses. This loss function encourages matching pairs to be close together in feature space while pushing non-matching pairs apart. However, it may not be the best option in our case because the difference in each feature contributes equal weight to the loss, whereas in our model, each feature may require different weights in the loss function that is not proportional to the weight learned in the convolutional network.

Chapter 3

Datasets

All the experiments in this study are conducted on a database two species of skinks: *Grand* and *Otago* of 3687 images in total created by biologists at New Zealand Department of Conservation.

Figure 3-1: Top: Left view images from Grand dataset.
Bottom: Right view images from the Otago dataset

The Grand database contains 1871 `RGB` images of 206 individuals with variations in sizes, lighting, and background, while the Otago database contains 1816 `RGB` images of 221 individuals, also with such variations. The images are closely cropped to include only the anterior end of the subjects; therefore, the size varies from 1056 $\times$ 564 to 2437 $\times$ 1215 pixels

Table 3.1: Summary of each database

	Number of Images					Images per Individual	
Name	Left View	Right View	Total	Individuals	Singletons	Avg.	Max.
Grand	929	942	1871	206	69	9	31
Otago	903	913	1816	221	58	8	24

3.1 General

3.1.1 Selfscore and Score Distributions

Sloop ranks the likelihood that two capture events contain the same individuals quantatively by a score. The similarity score between two individuals is obtained from comparing sift features of all the images within a capture capture to sift features of the images within the other individual's cohort *normalized by the minimum selfscore among all the images in the comparison.* A capture is a group of images containing the same individual, not necessarily having the same view, that is captured together, whereas a cohort is a group of images that are identified as the same animal, not necessarity having the same view or captured at the same time.

Selfscores

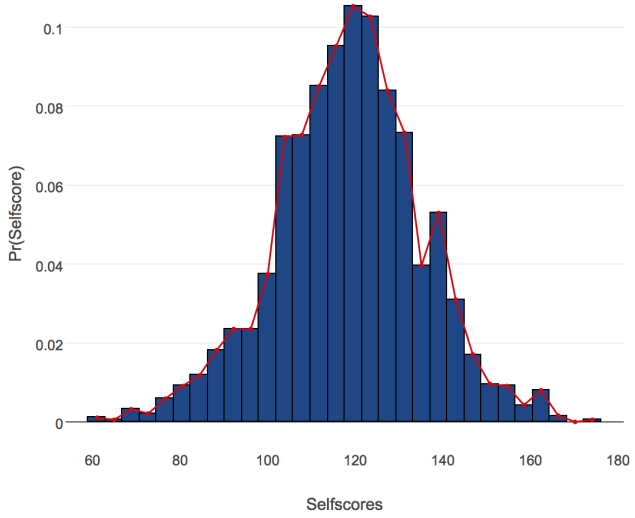
Selfscore of a capture is the *maximum* selfscore of all the images within the capture, where a selfscore of an image is calculated from comparing the sift features of each image with itself.

$$\texttt{selfscore}(C) = \max_{\forall I_i \in C} \texttt{sift\_match}(I_i, I_i)$$

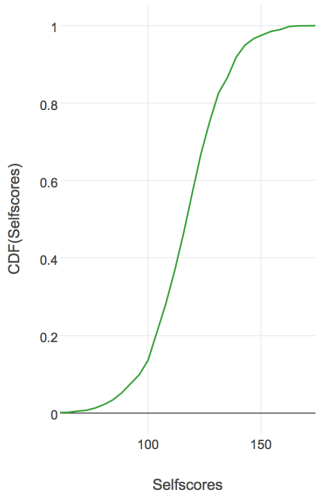
Scores

The score between the two individuals is the maximum score among all the comparisons among the images within the cohorts.

$$\texttt{score}(C_1, C_2) = \max_{\forall I_i \in C_1 \forall I_j \in C_2} \frac{\texttt{sift\_match}(I_i, I_j)}{\min(\texttt{selfscore}(C_1), \texttt{selfscore}(C_2))}$$

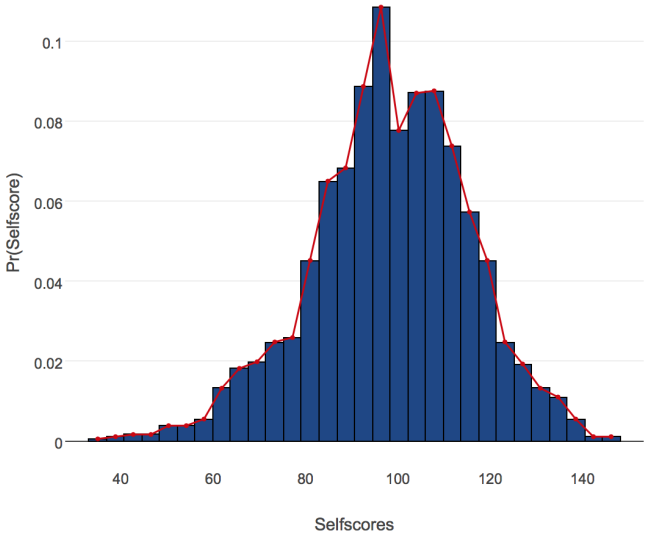


(a) Selfscores

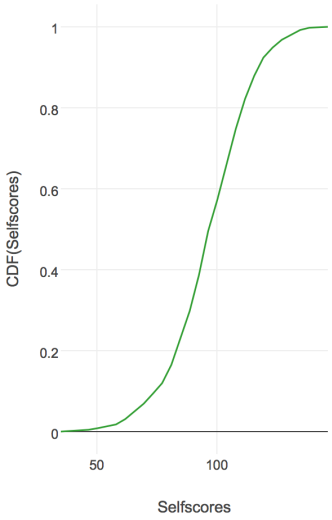


(b) CDF of selfscores

Figure 3-2: Selfscore Distribution



(a) Selfscores



(b) CDF of selfscores

Figure 3-3: Selfscore Distribution

where C is a capture and I is an image in a capture.

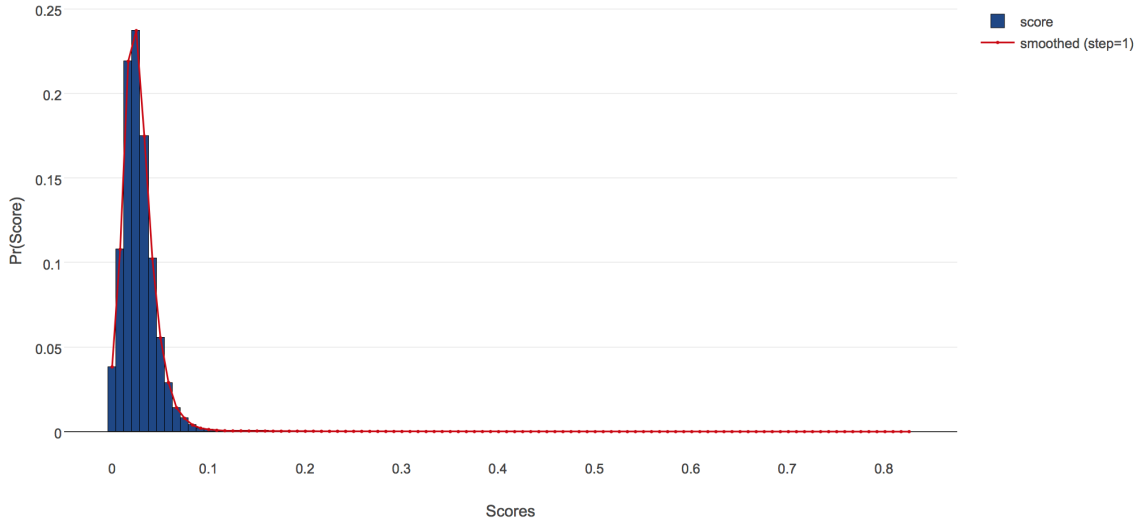


Figure 3-4: Score Distribution

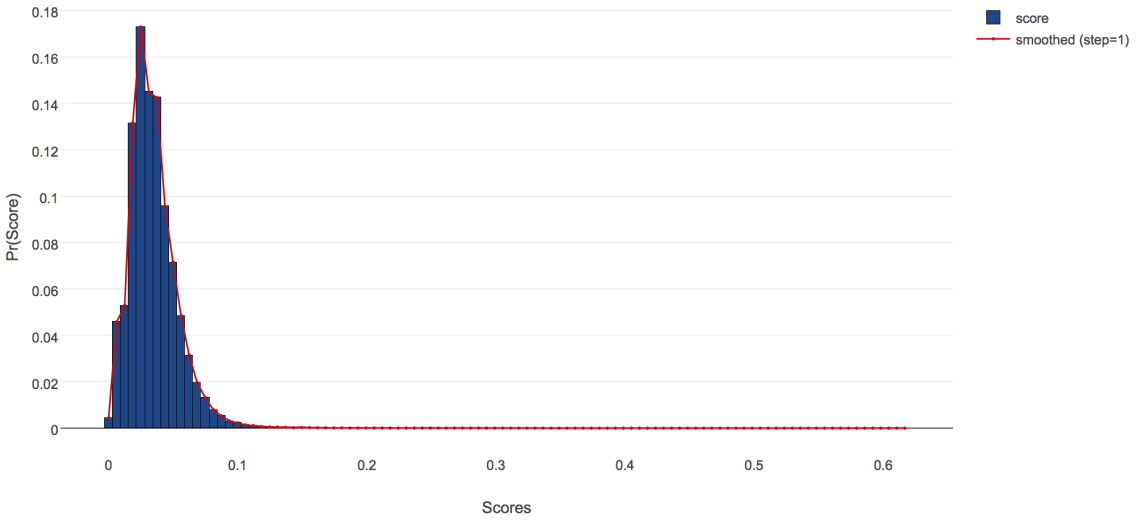


Figure 3-5: Score Distribution

Chapter 4

Relvance Feedback

To study and understand the effect of relevance feedback to the retrival results, we implement a simulator that reproduces the relevance feedback processes of sloop. The implementation of the simulator is described in 4.1 section. We then compare the initial ranked result set outputted from SIFT computation with the results that undergo different rounds of relevance feedback with various configurations in the 4.3 section.

4.1 Methodology

The relevance feedback simulator mimics the behavior of the task submission process by the repeatedly sampling a batch of capture pairs from the pool of all unknown capture pairs. For all the sampled pairs within a batch, the simulator marks them as matching pairs or non-matching pairs with the *correct* answers.

When all the pairs in a specific batch are marked, it constructs an *identity graph* that represents the connections between all the known pairs using the marked answers. The identity graph is undirected. The nodes in the graph represent captures. If there is an edge connecting between node A and B, capture A and B contain the same individual. Graphically mapping the relationship of individuals results in fully-connected subgraphs formed by all the captures containing the same animals.

Upon the retreival of the new information from each batch, the simulator uses

the identity graph to infer the answers of the unknown pairs. This imitates the in-database merging logic of the current version of Sloop[13]. The transitive relations are drawn from both matching pairs and non-matching pairs. For example, given capture A is known to contain the same individual as that in capture B, and capture B is known to contain the same individual as that in capture C, the simulator can infer that capture A contains the same individual as that in capture C, without having known the actual answer of A and C. Once such transitive interence is made, it adds C to the identity graph. Then, it continues to sample a new batch iteratively until the distribution satisfies the convergence condition specified by the user.

The inferred decision plays a very crucial role in relevance feedback. The validity of such inference relies on the validity of the premises. If the answers to the matches are correct then the conclusion must be true. Problems arise when our seemingly *correct* answers are in fact incorrect. In the actual production of Sloop, the inconsistence results from such error is detectable and will be submitted for manual review. However, in the simulation, we would like to also examine the consequences of the incorrect answers. With the auto merging, the error propagates rapidly. Therefore, the simulation allows users to specify the error probability at the initialization.

4.2 Metrices

This section describes the metrices used to evaluate the performance our models. Along with our goal to maximize the precision and recall, we would like to minimize the cost of the crowdsed feedback or to operate optimally within a budget constraint. The simulator also provides a counter that counts the number of iterations a distribution takes to converge. However, this is out of our interest because the time a model takes to converge can also be inferred from the number of task submitted. Given a model, the fewer assignments posted, the faster the model converges.

4.2.1 Precision and Recall

As you can see, in the system where true negatives (true non-matching pairs) largely outnumbers the true positives (true matching-pairs), comparing the *true positive rate (TPR)* or recall to the *false positive rate (FPR)* is not very meaningful. This is because false positive rate, which is $\Pr(\hat{y} = 1|y = 0) = \frac{FP}{FP+TN}$, is going to be approximately 0 as TN is very large. Thus, instead of using ROC (receiver operating characteristic) curve, which is a plot of TPR and FPR, we use *precision-recall curves (PR)* to analyze the performance of our models.

A precision recall curve is a plot of precision and recall as we vary the threshold θ. Precision or positive predictive value (PPV) is defined as following [10]:

$$PPV = \Pr(y = 1|\hat{y} = 1) = \frac{TP}{\hat{P}} = \frac{TP}{TP + FP}$$

We evaluate the ranked retrieval performance of diffrent relevance feedback sampling policies using the *Mean Average Precision (MAP)*, which is roughly the area under the *precision-recall curves*.

4.2.2 Cost

The total amount of money we have to spend to reward the workers is proportional to the *number of tasks we published*. If, at the moment we have just finished marking all the unknown pairs, there are still some leftover tasks on the crowdsourcing, those tasks are not going to be cancelled (unless the user forcefully cancels the tasks himself.) Hence, we can use the number of tasks we published as our cost metric.

4.3 Experiments

In this section, we consider a setting in which time evolves in rounds. In each round, the we, the requester, simulates the task submission process in a crowdsourcing platform assuming that all the workers completes all the task. Our goal as the requester is to maximize the Mean Average Precision value of a preliminary score distribution

Table 4.1: Number of capture pairs for each species

Species	Number of pairs
Grand	507528
Otago	492690

outputted from Sloop with the amount payments we have to make in mind.

We report three experiments corroborating the improvement of the retreival results after various rounds of relevance feedback and different sampling strategies. Experiment 4.3.1 establishes that relevance feedback improves the acceration of precision and recall of the preliminary results at various batch sizes. Experiment 4.3.2 shows the consequence of the errorneous answers at diffferent error probabilities. Experiment 4.3.3 compare the performance of various sampling policies.

All experiments are performed on the Grand and Otago dataset described in Chapter 3. Both datasets are annotated by the biologist, which we take as our true labels. Table

4.3.1 Batch Sizes

This experiment first shows that relevance feedback improves the acceration of precision and recall of the preliminary results at different iterations and batch sizes. To establish these relations, we randomly sample some unknown pairs from our unknown pair pool using *uniform* sampling policy, in which drawing each unknown pair is equally probable. The reason we use uniform random sampling policy is because it is simple and unbiased, which is useful to get a general baseline value.

During the sampling process, we observe the Mean Average Precision (MAP) at each iteraion. A batch of samples is iteratively drawn until we know the answers to *all* the unknown pairs or MAP is equal to 1, whichever happens first.

We then repeat the process for various batch sizes, which is defined to be the total number of unknown pairs marked in between a pair of the following events: initialization, identity inference, and termination. Finally, we compare the number of pairs sampled to reach convergence for each batch size.

4.3.2 Errors

Despite the tasks distribution and the gold standard questions, there is probability of 0.078 of obtaining an incorrect answer, assuming that all the preventive events are independent. However, in practice, errors occur haphazardly rather than systematically. Thus, 0.078 is the worst case, where we assumed the worker always make a guess with equal probability.

This experiment models such error. Given an error probability of ϵ, if a pair of captures is sampled, the simulator marks it with the annotated answer (correct answer) with the probability of $1 - \epsilon$; otherwise, the pair is marked with the opposite label (incorrect answer). Again, we compare the values of MAP at each iteration and the total number of pairs sampled for each error probability.

4.3.3 Sampling Policies

Sampling policy plays a significant role in the retrieval. Given an initial ranking, a pair of captures whose score is within a certain range or higher than certain threshold is more likely to be a match. Such range and threshold is species-specific and relies on Sloop's classification performance.

We experiment with following policies:

Uniform Sample an unknown pair from a uniform distribution where drawing each *pair* is equally probable.

UniformScore Sample an unknown pair from a uniform score distribution where drawing a pair with each *score* is equally probable.

TopMatches Always select an unknown pair with the highest score.

Nonmatches Always select an unknown pair with the lowest score.

Normal Sample an unknown pair from a normal distribution with μ =median and $\sigma = 0.3$.

Percentile Always select an unknown pair at the median.

AllScores Divide the scores into n bins, where n =`batch_size` and then select some unknown pairs from all the bins so that the total number of unknown pairs sums to `batch_size`.

4.4 Results and Discussion

Figure **??** displays the Precision-Recall graph at a given number of sampled unknown pairs. With zero error probability, relevance feedback reduces the number of comparisons required for each specifies by a factor of 317 and 307 for grand and otago respectively. Within four iterations of feedback loop, Mean Average Precision (MAP) reaches 1.0.

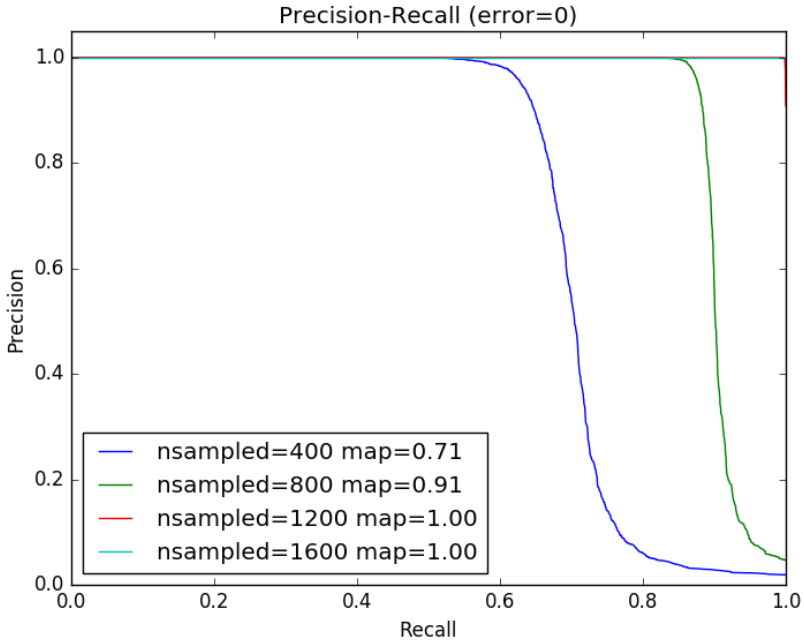
The results corroborates the fact that the relevance feedback dramatically accerates precision and recall given the correct feedback information. Such inclination of precision and recall is expected largely due to the interpolation of the new information.

4.4.1 Batch Sizes

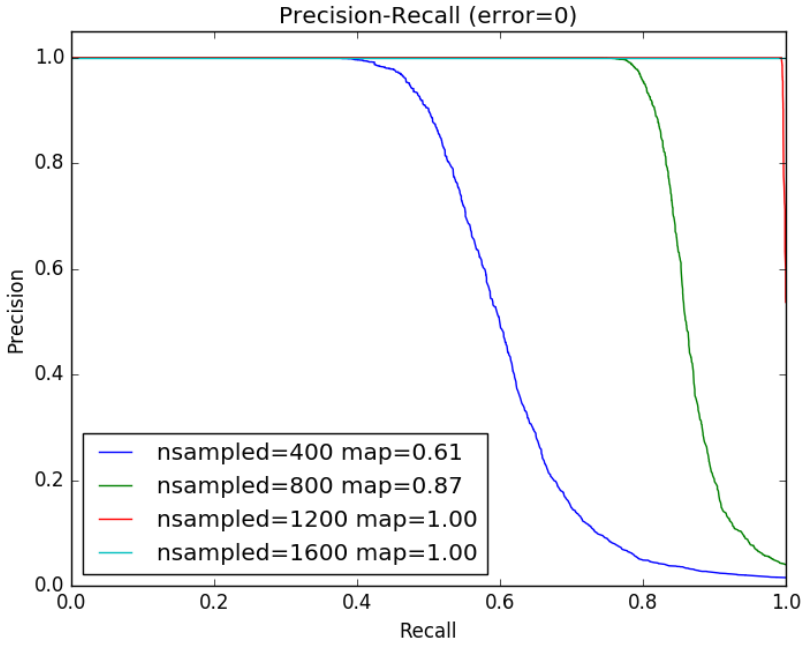
Overall, from the graphs, MAP increases as we feed more data into the system. Figure **??** indicates that we publish more uneccessary tasks as we increase the batch size. The fewer captures there are in a batch, the lower the overall cost. Taking this idea one step further, we would like to infer the matches and merge the individuals as frequently as possible. However, empirically, smaller batch size may upset some workers who would like to continuously work on the tasks. Therefore, we need to find the smallest batch size that is still large enough to engage the workers.

4.4.2 Errors

Despite the present of errors, relevence feedback still yields higher MAP overall with an error threshold of 0.05. With a nonzero error probability, MAP curves downward before it curves up and reach a saturation point. This is because initially when it



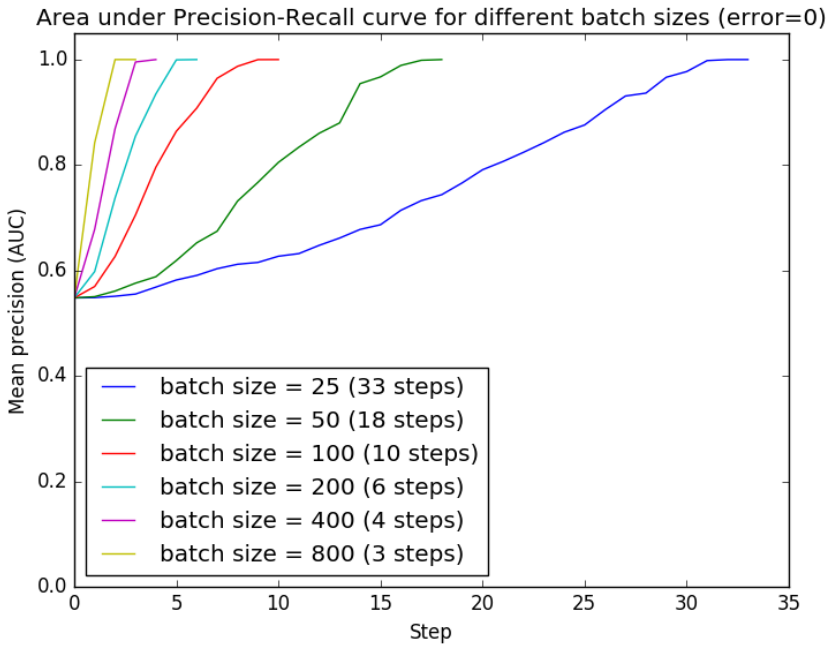
(a) Grand



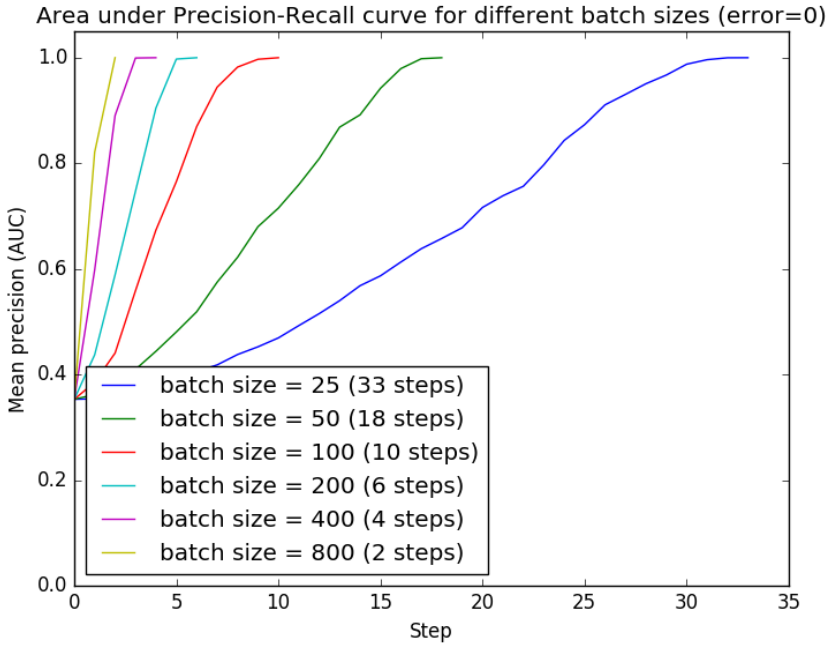
(b) Otago

Figure 4-1: Precision-Recall graph ($Error = 0$, `batch_size`$= 400$)

does not have much data, the system is very sensitive to errors, especially the false negatives, which trigger the irreversible merge operation. Thus, the precision decline



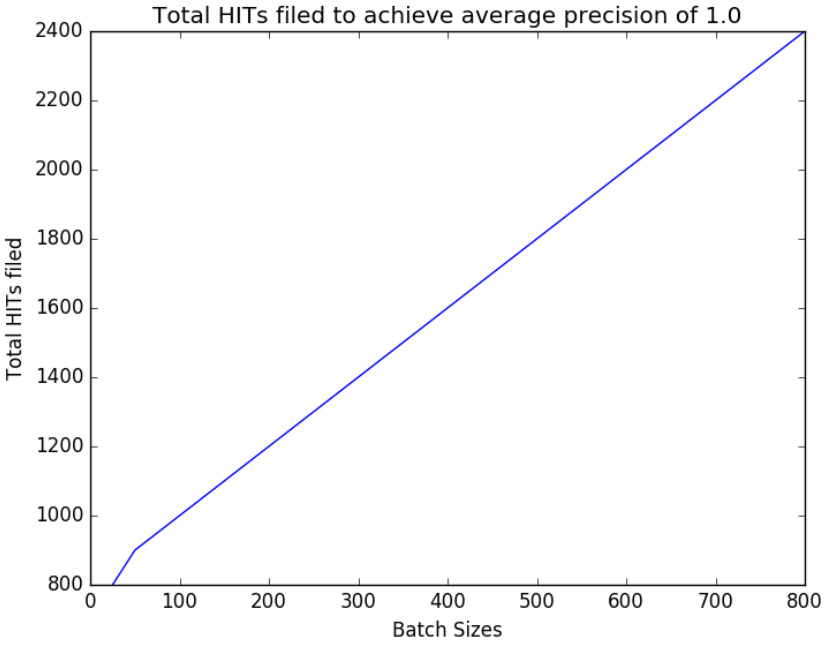
(a) Grand



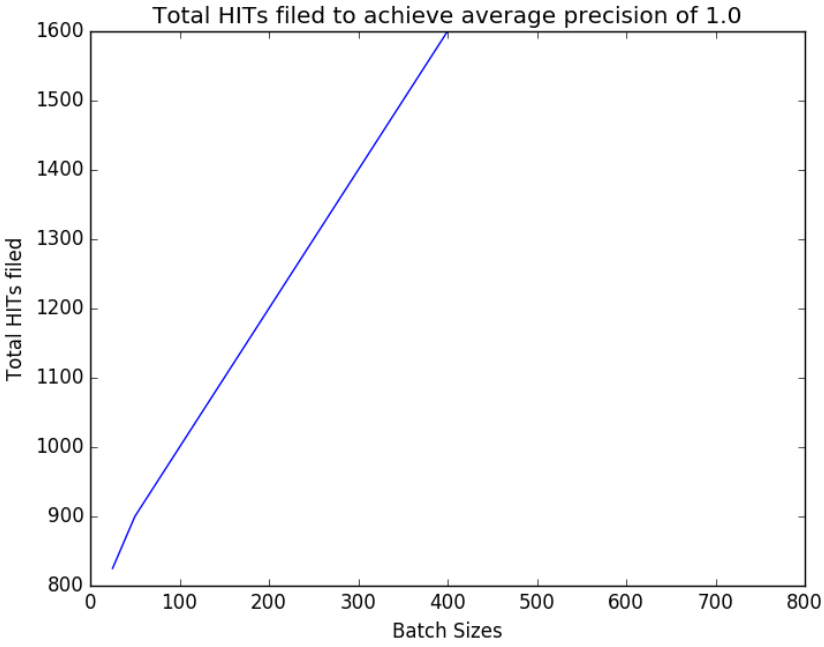
(b) Otago

Figure 4-2: Mean Average Precision (MAP) for different batch sizes ($Error = 0$)

rapidly. As it gains more correct data, it is able to recover from the downward phase. However, the historical merges resulted from the past faulty data are irrevocable, so



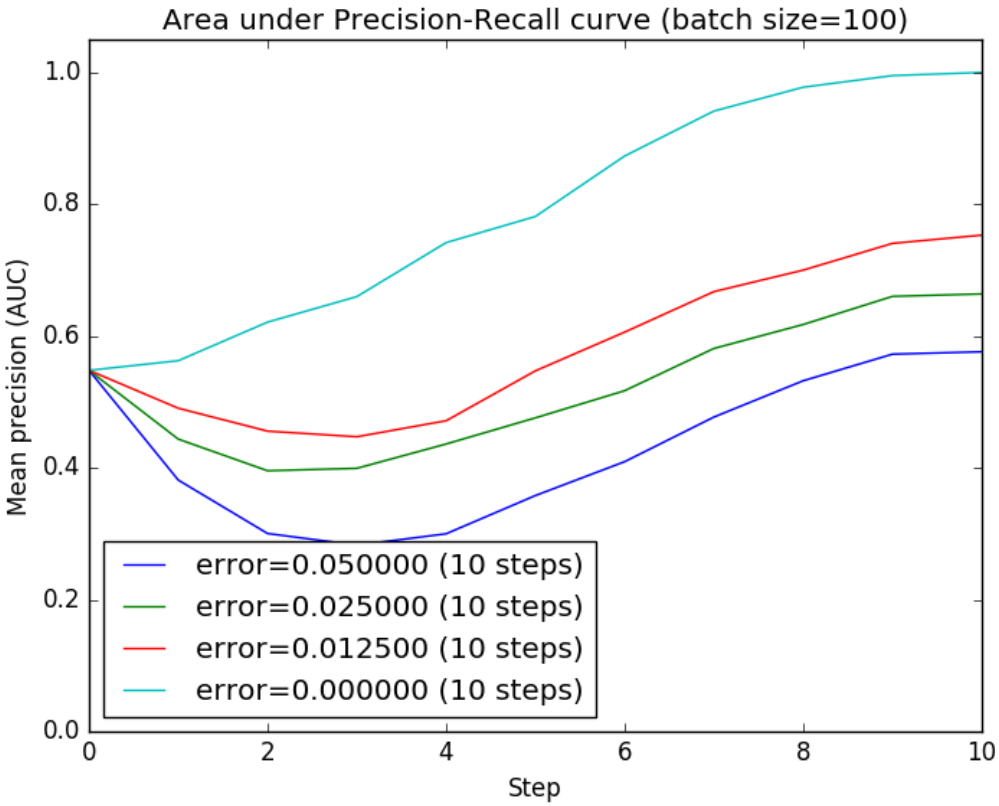
(a) Grand



(b) Otago

Figure 4-3: The total number of unknown pairs marked to achieve $MAP = 1$ ($Error = 0$, `batch_size`$= 100$). Notice that this equals the total number of tasks published to achieve $MAP = 1$.

the precision saturates eventually.

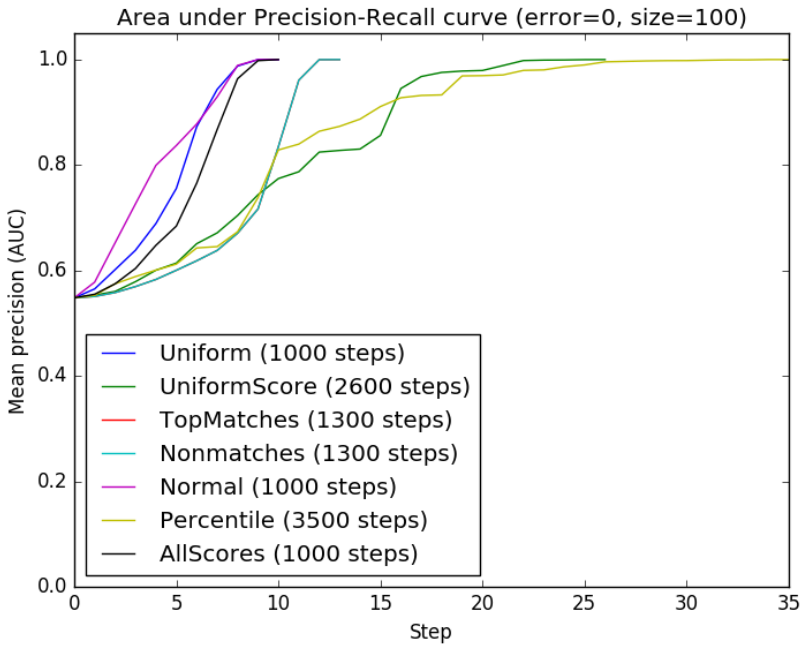


(a) Grand

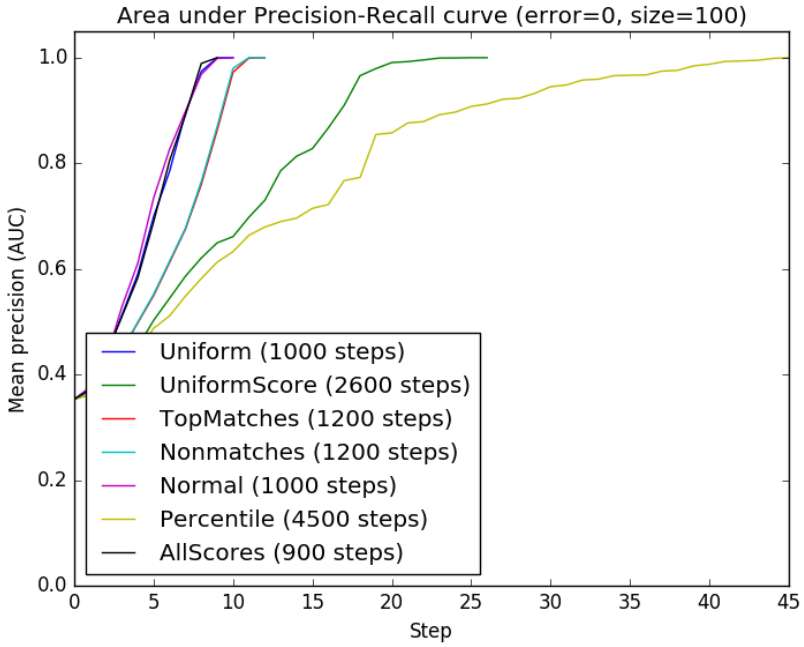
Figure 4-4: Mean Average Precision for different probabilities of error

4.4.3 Sampling Policies

Uniform sampling (Uniform), normal sampling (Normal), and sampling from all the scorces (AllScores) seem to outperform other sampling policies in term of the total cost. The samplers that sample single score values at a time, such as sampling from median (Percentile), performs poorly compare to others. However, this depends largely on the species of the animal that we are interested in. As you can see the performance of the each sampling policy differs slightly between grand and otago.



(a) Grand



(b) Otago

Figure 4-5: Mean Average Precision (MAP) for various sampling policies ($Error = 0$, `batch_size`$= 100$)

Chapter 5

Crowdsourced Relevance Feedback

5.1 Sloop MTurk

As described in 4, multiple rounds of relevance feedback accelerates precision and recall of the recognition. This chapter presents the architecture of *Sloop MTurk*, the crowdsourced relevance feedback integration of Sloop.

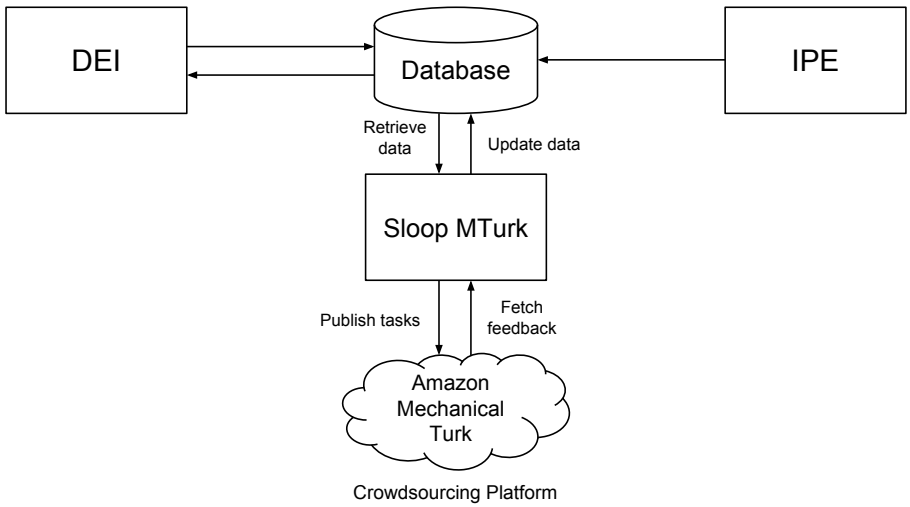


Figure 5-1: Sloop Architecture with SloopMTurk integration

Sloop MTurk communicates with the crowdsourcing platform in order to provide human feedback on the original rankings. Our focus is on

5.1.1 Architecture

Following are the four major functions used within the relevance feedback workflow.

1. Retrieve
2. Publish
3. Fetch
4. Update

Retrieve

In the first stage of the relevance feedback workflow, Sloop MTurk retrieves a number of unknown, and known image pairs from the database in the ratio described in 5.1.3. The metadata about the retrieved pairs, such as the answers to the known pairs, individual identification numbers, and source address, is stored in a lightweight sqlite database while the images are stored separately.

Sloop MTurk never publishes a duplicated *unknown* pair onto MTurk. It actively checks with the local database before any retrival if an unknown pair is ever published before. User can set the sampling policy of Sloop MTurk in the config file. The default value is set to use the normal sampling on the median whose performance has been shown in chapter 4.

Sloop MTurk only downloads the images necessary for the tasks to minimize the bandwidth usage. Upon the retreival, Sloop MTurk caches the image data into its local storage on the file system running on NGINX. Caching image data on the filesystem eliminates the network latency and cuts down the data tranfer overhead because fetching large responses, like the images, over the network is both slow and expensive. The advantage of storing images on the file system, over storing them in the database, is that we can also refer to the images hosted on the filesystem in our actual deployment, instead of having to set up another machine, or to use other external web hosting.

Publish

Sloop MTurk publishes the tasks onto MTurk immediately after all the information is fetched. The tasks published are available on Amazon Mechanical Turk under the title: 'Image Matching – Animals' posted by user 'sloop'.

Fetch

User can fetch the results that are completed via Sloop MTurk fetch. This can also be set up as a cron job. Sloop MTurk searches for all the reviewable tasks, and automatically accepts and retreives the tasks whose answers to all of the known pairs are correct; otherwise, it rejects the tasks. If a task is accepted, an amount of payment will be incurred to the user's account to reward the worker.

Sloop MTurk logs all the answers to the unknown pairs in the accepted assignments into the local sqlite database. No result are going to be deleted until the user runs garbage collection command.

Update

Update command pushes the answers logged in the local database back to the original database so that the data is ready for another round of relevance feedback. The remote database server then infers the captures' cohort from the data pushed and the existing data, and then performs the merging logic accordingly[13].

The update command is seperated from fetch for the debugging purpose. In practice, as we would like to update the upstream database as frequent as possible as we have seen in the Experiments section in 4.

5.1.2 Platform Selection

Workers and requesters interact via a crowdsourcing platform. All of our tasks are published on Amazon Mechanical Turk (MTurk), which is one of the most popular crowdsourcing platforms. Tasks published on MTurk are called Human Intelligence

Tasks (HITs). The term 'HIT' and 'tasks' are going to be used interchangeably throughout this report.

5.1.3 Human Intelligence Task (HIT)

Requesters are able to post jobs known as Human Intelligence Tasks (HITs). Workers (called Providers in Mechanical Turk's Terms of Service, or, more colloquially, Turkers) can then browse among existing HITs and complete them in exchange for the amount of money the requesters have promised.

Task Design

A wide variety of tasks can be crowdsourced. There are two possible designs we have considered:

- Rankings and weights Each worker may be asked to provide a full rankings or a relative weights for a given pool of candidates. Although knowing the correct rankings is valuable to ranked retrieval results, which is Sloop preliminary output, ultimately, we would like to determine a sharp cutoff between matching images and nonmatching images so that each individual can be exclusively identified. Ranks alone do not provide such cutoff. Even if we knew all the correct rankings, we would not be able to decide whether two animals are the same individual animals. Thus, rankings and weights is not a viable option in this case.

- Multiple-choice questions A question can include multiple options, for instance, which of the images in the choices contains an animal that matches, the given individual animal. However, this approach only tells us which of the pictures among all the choices best matches a given image, which is an inefficient approach of ranking images. Alternatively, we can ask workers to provide all the matches among all the options. In this case, there can be none or more than one answer. However, using this method, we only gain the information of

one individual from n comparisons, where n is the number of choices for each question.

We can model our task as a binary questions of whether a pair of images, of the same view, contains the same individual animal (Is a match?). Since, eventually, all the image pairs will be labelled as either a match or a nonmatch, we will be able to constract a mappping that allows us to exclusively identify all the individual animals in the pictures.

Validation

For each task, since the quality of the submitted work is not directly observeable, we add other three *gold standard* questions, for which the correct answers are known a priori. Among the three questions, one is a known pair of images containing the same individual, another is a known pair of images containing different individuals, and the other is a random known pair. These *gold standard* questions work as qualifying tests for eligible workers. Additionally, they also accelerate the workers' learning process and skill level.

To further reassure the correctness of the answers submitted by the workers, we publish the same assignment to three workers, and estimate the correct answer by consensus. We only allow workers with task acceptance rate higher than 0.8 to prevent spam. If the majority agrees on some answer, it maybe safe to assume that it is correct. Obviously, given a task, the more number of workers we assign, the less chance there is of getting an incorrect answer. However, the budget and the time it takes to complete all the assignments is going to increase linearly as we increase the number of the workers, while the correctness only increases logarithmically. Therefore, we decided to publish exactly three assignments, which is the minimal number to reach a concensus, for each task.

5.1.4 Cost Model

Sloop MTurk uses the evaluation metrices mentioned in chapter 4 to measure the performance of the system.

Correctness

To analyze the correctness, we are still using the Mean Average Precision to evaluate the correctness of the results. MAP has been shown to have especially good discrimination and stability on evaluating information retreival systems[10].

Expense

Each worker, upon completion of a task, receives the posted price for that task. The more tasks published, the higher the total payment we have to make.

In the actual production of Sloop, the inconsistence results from such error is detectable and will be submitted for manual review. The conflicts are going to be resolve by the biologists or relevant expert, whose time and effort is a relatively more valuable than that of the online workers. In this case, the cost incurred by the manual review is going to be higher than the cost of the payment we need to reward the workers. Quantitatively, for each manual review, the cost is going to be the number of all the captures involved in the conflict multiplied by a constant.

Chapter 6

Relevance Feedback with Adaptive Sampling

One of the most important questions we would like to answer is: given an initial distribution of scores, where should we be sampling from? In this chapter, we take advantage from the fact that we gradually gain more information about the score distribution over time to dynamically determine where we should be sampling from.

First, we perform different statistical analyses on the data. Then we propose a new sampling scheme based on our analyses. Finally, we set up an experiment to compare the performance of our proposed scheme to the other standard sampling policies we have implemented.

6.1 Statistical Analysis

6.1.1 Probability of Score given matches/non-matches

We can estimate the probability of score given a match by

$$\Pr(score = s \mid match) = \frac{\texttt{\# matches\_whose\_score=s}}{\texttt{\# matches}}$$

In the same way,

$$\Pr(score = s \mid nonmatch) = \frac{\texttt{\# nonmatches\_whose\_score=s}}{\texttt{\# nonmatches}}$$

From the plot above, we can deduct following conclusions:

- Most of the cohorts has score between 0 and 0.07 overall so most matches and nonmatches have score less than 0.06.
- We expected $\Pr(score \mid match)$ to increase as the function of score, but it turns out that it actually decreases. This results from the fact that there are *very few* matches with scores higher than 0.33, which is the 99th percentile of the matches' score.
- The number of matches with $score \in [0.08, 0.35)$ is greater than that of non-matches with the score in the same range. That is

$$\Pr(score \mid match) > \Pr(score \mid nonmatch) \text{ where } score \in [0.08, 0.35)$$

6.1.2 Probability of a matching score at a given cohort size

Initially, we expected that as the cohort size increases, $\Pr(score \mid \#cohort)$ should peak closer to 1 since `selfscore` should decrease and $\max(\texttt{sift\_match}(I_i, I_j))$ should increase as the number of captures in a cohort increases. However, the distribution of $\Pr(score \mid \#cohort)$ for all numbers of cohorts seem to be similar to one another.

6.1.3 Earth Mover's Distance

We use Earth Mover's Distance (EMD)[12] as our metric to calculate the minimal cost required to transform $Pr(score|nonmatch)$ to $Pr(score|match)$.

Initially, we also expected that the relationship between EMD and cohort size should be gaussian. However, there seem to be no relationship between $Pr(score|nonmatch)$ and $Pr(score|match)$.

6.1.4 Entropy of $\Pr[score|match]$ at different cohort sizes

Entropy is the average amount of information we get from each distribution. H(X—Y) = amount of randomness in the random variable X given the event Y.

6.1.5 Probability of a Matches/Non-matches

$$\Pr(match \mid score) = \frac{\Pr(score \mid match)\Pr(match)}{\Pr(score)}$$

$$\Pr(nonmatch \mid score) = \frac{\Pr(score \mid nonmatch)\Pr(nonmatch)}{\Pr(score)}$$

6.2 Adaptive Sampling Policies

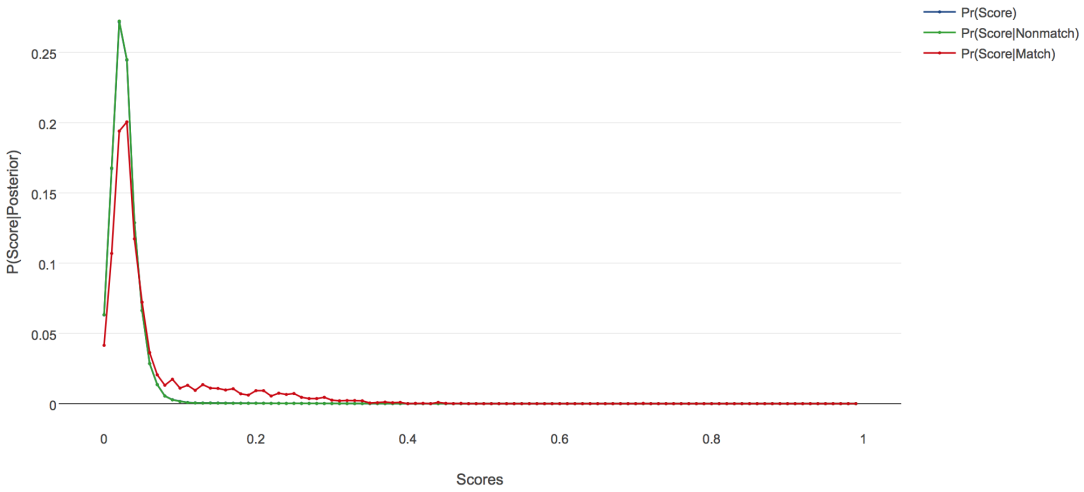
Starting with a completely unknown pairs, we would like to first get the idea of how the overall distribution looks like. As we have seen in chapter 4, the unbiased sampling policies, such as Uniform, and AllScores, that broadly sample the data from the distribution seem to form a set of good candidates.

Upon the retrieval of the general distribution, we would like to adaptively utilize the known data to determine the optimal sampling site. We experiment with sampling from the peak of the difference between the CDF of the score given matches and non-matches using various sampling policies.

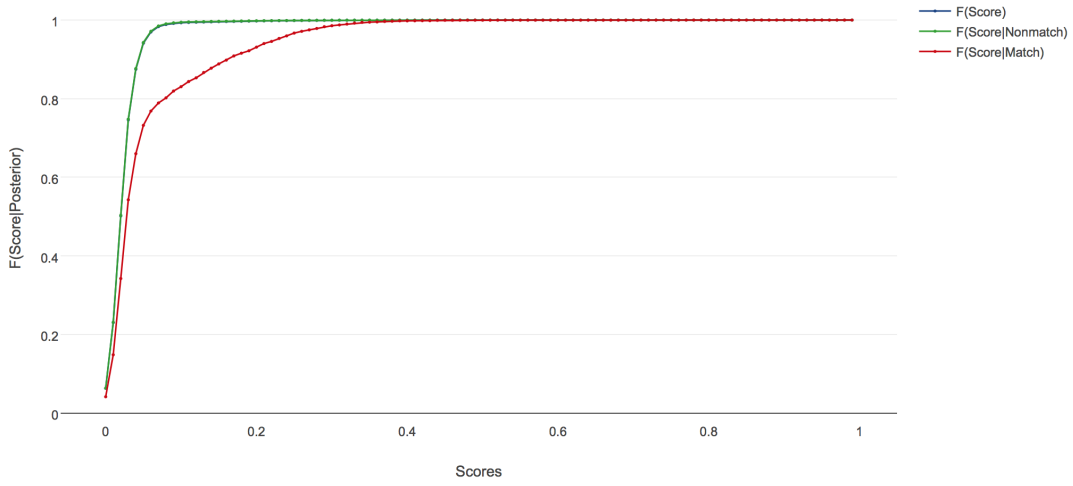
After we get the general picture of the distribution, we repeatedly reconstruct the plot of the difference between the CDF of the score given matches and non-matches, then uses normal sampling policy to overlay a gaussian distribution, with μ =`peak_of_the_CDF_difference` and $\sigma = 0.3$, on the data. We continue each cycle by the sampling probability distribution reconstruction, and sampling from the generated distribution. Additionally, we also use Specific sampling policy to sample specificly at the `peak_of_the_CDF_difference` to compare the performance with the Normal sampling policy.

6.3 Results

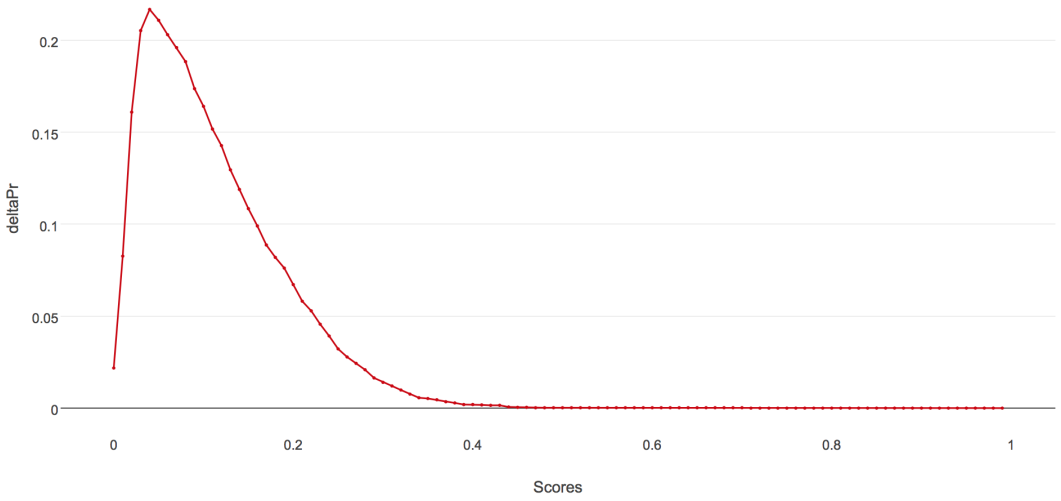
The adaptive sampling policy seem to perform averagely well. The performance does not differ from that of the other static sampling policies we have seen in chapter 4.



(a) Probability of Score given matches/non-matches

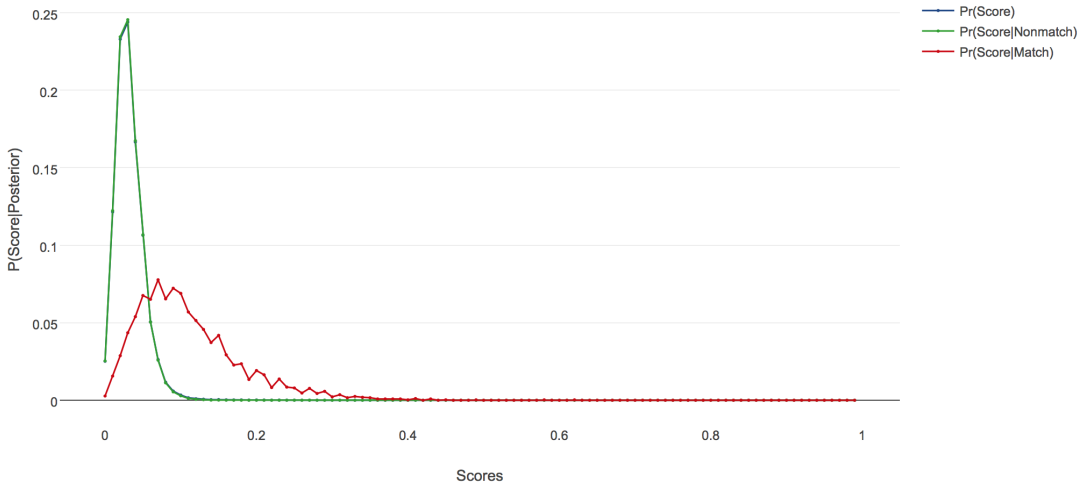


(b) CDF of Score given matches/non-matches

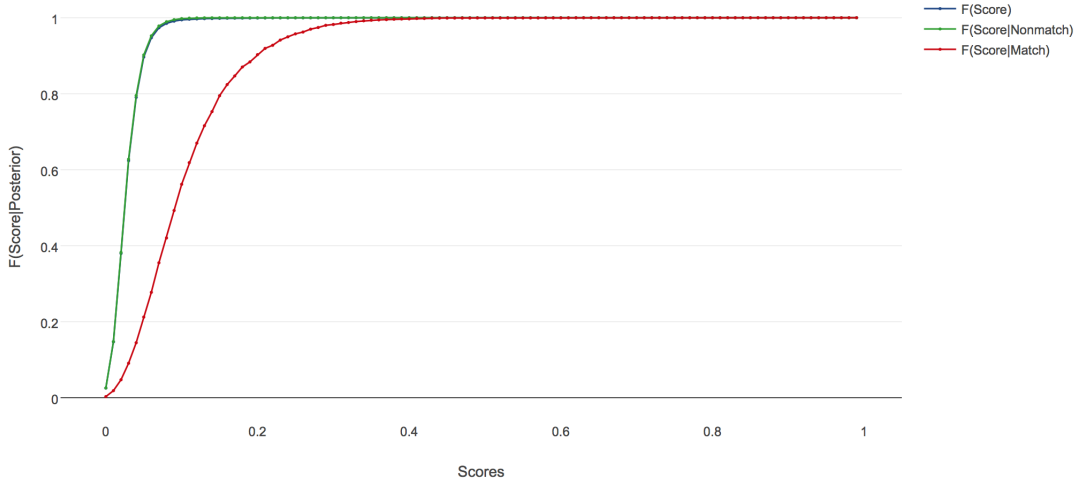


(c) Difference between the CDF of Score given matches/non-matches

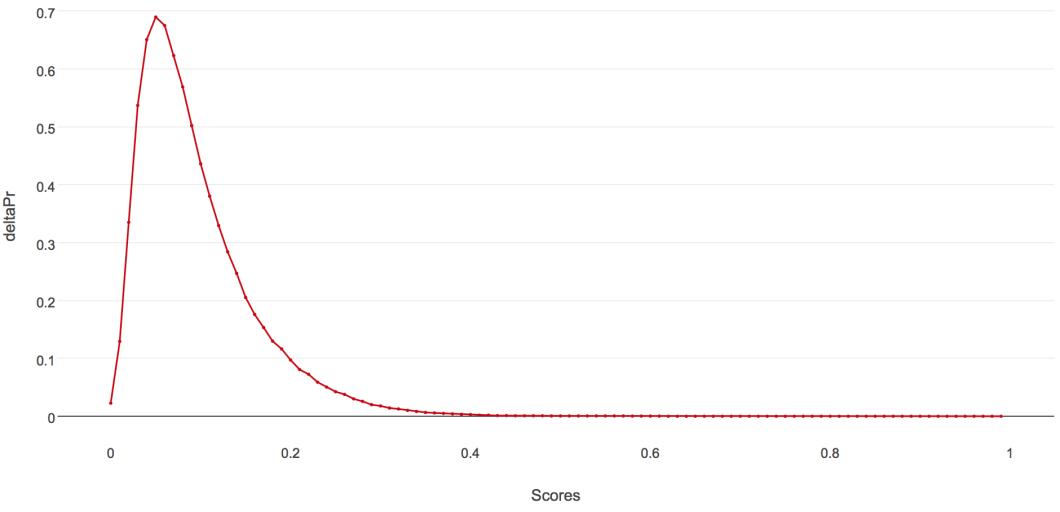
Figure 6-1: Functions of Score given matches/non-matches



(a) Probability of Score given matches/non-matches

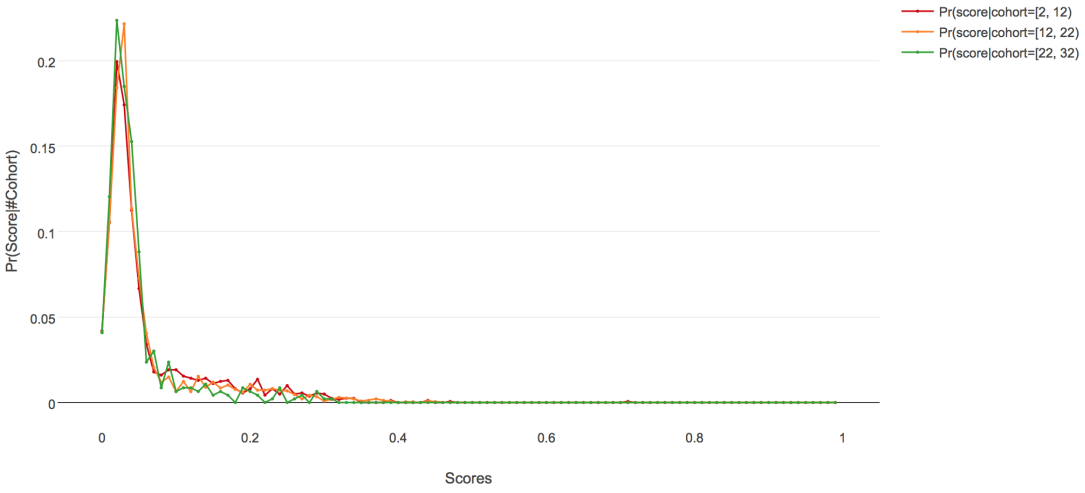


(b) CDF of Score given matches/non-matches

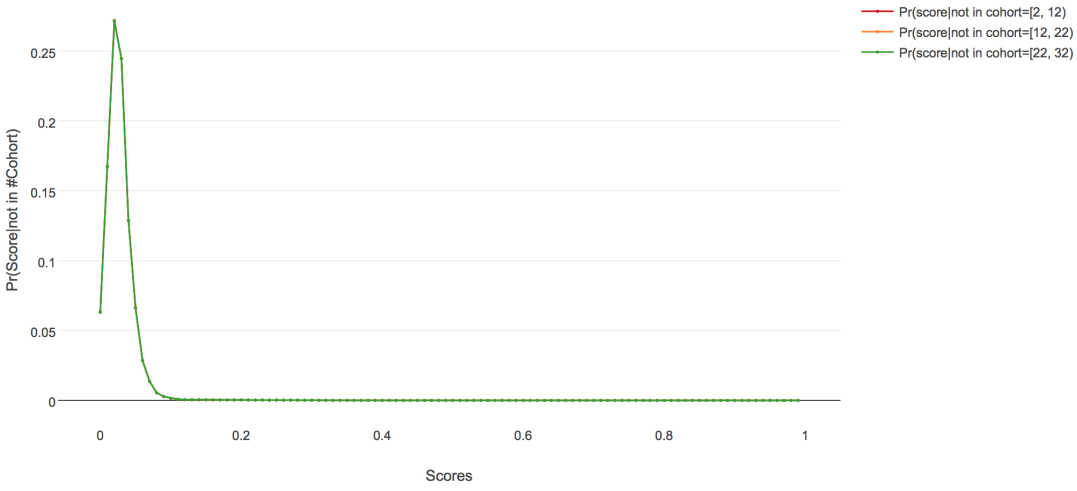


(c) Difference between the CDF of Score given matches/non-matches

Figure 6-2: Functions of Score given matches/non-matches

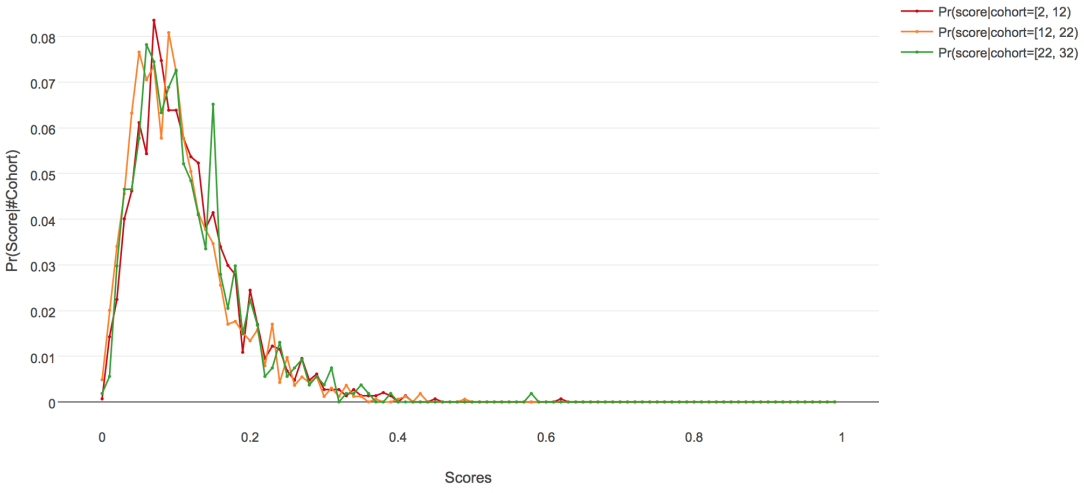


(a) Probability of a matching score at a given cohort size

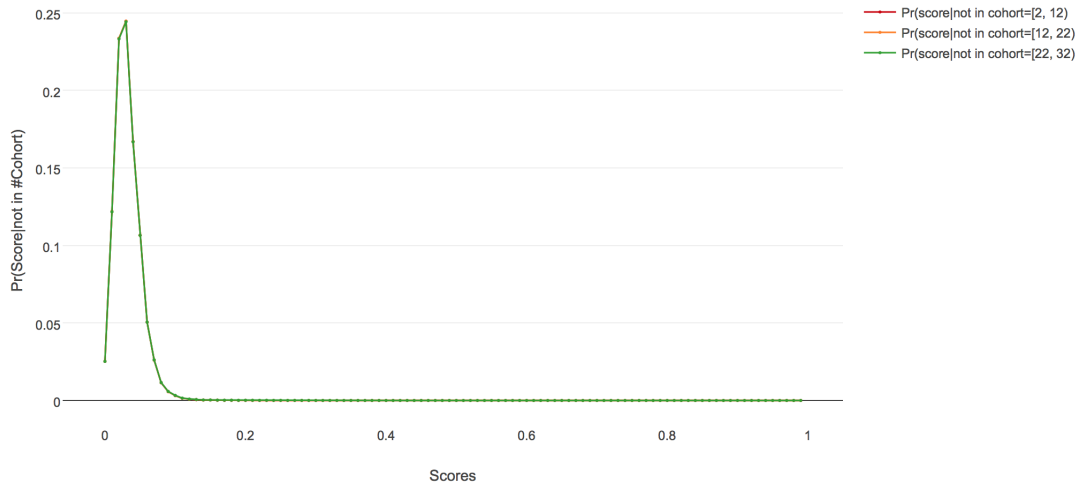


(b) Probability of a score *not* in a given cohort size

Figure 6-3: Probability of a score given a range of cohort size



(a) Probability of a matching score at a given cohort size



(b) Probability of a score *not* in a given cohort size

Figure 6-4: Probability of a score given a range of cohort size

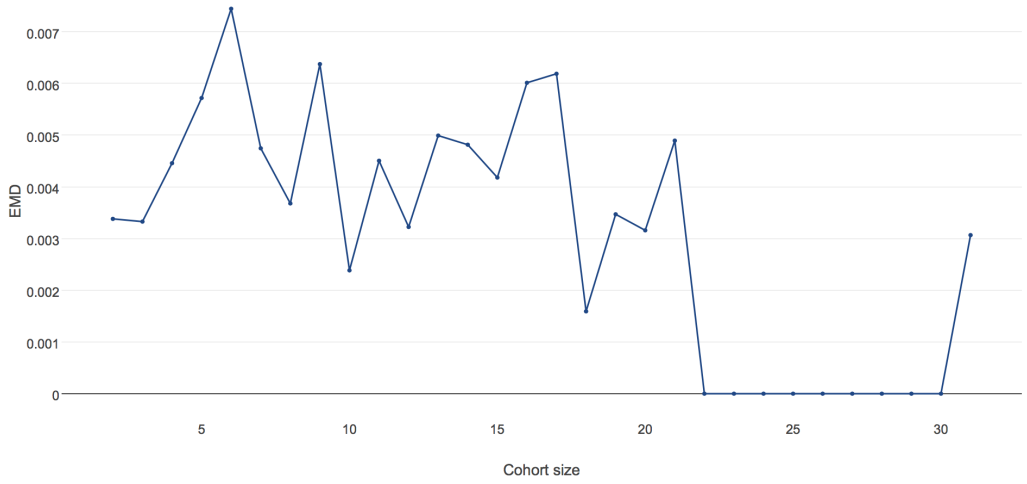


Figure 6-5: Earth Mover's Distance between the probability distributions

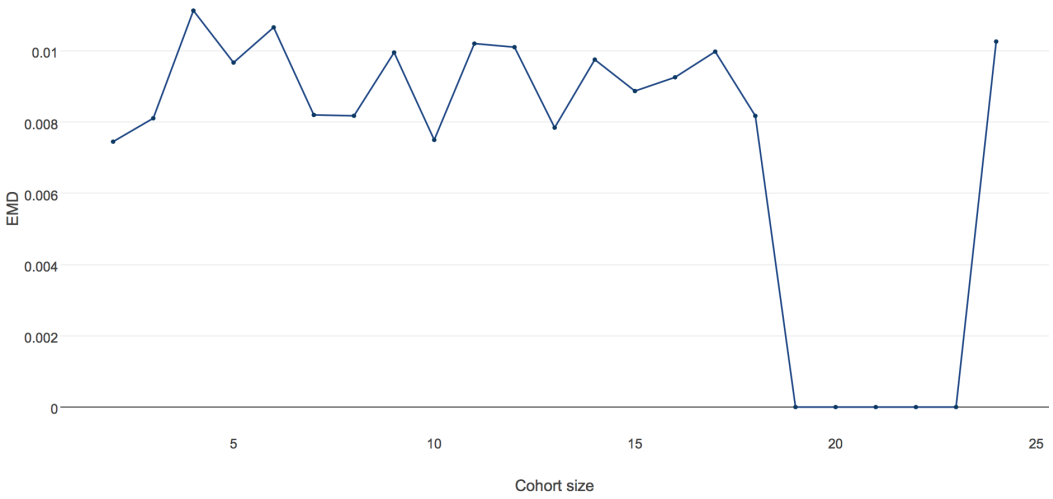
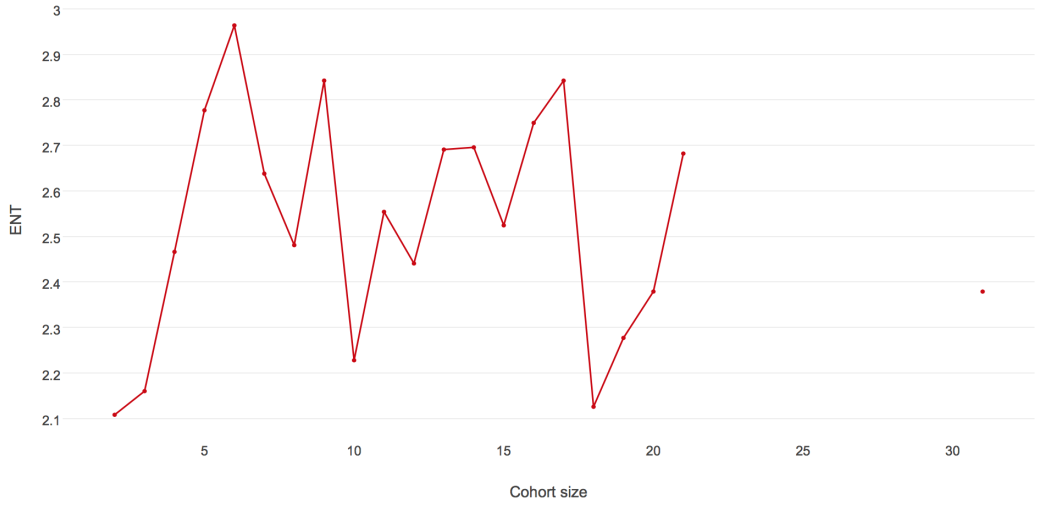
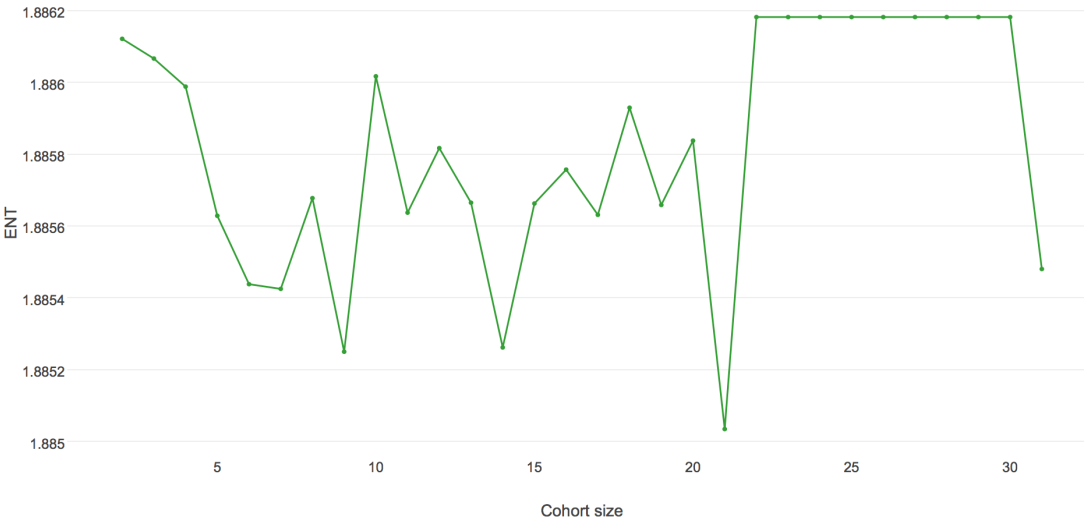


Figure 6-6: Earth Mover's Distance between the probability distributions

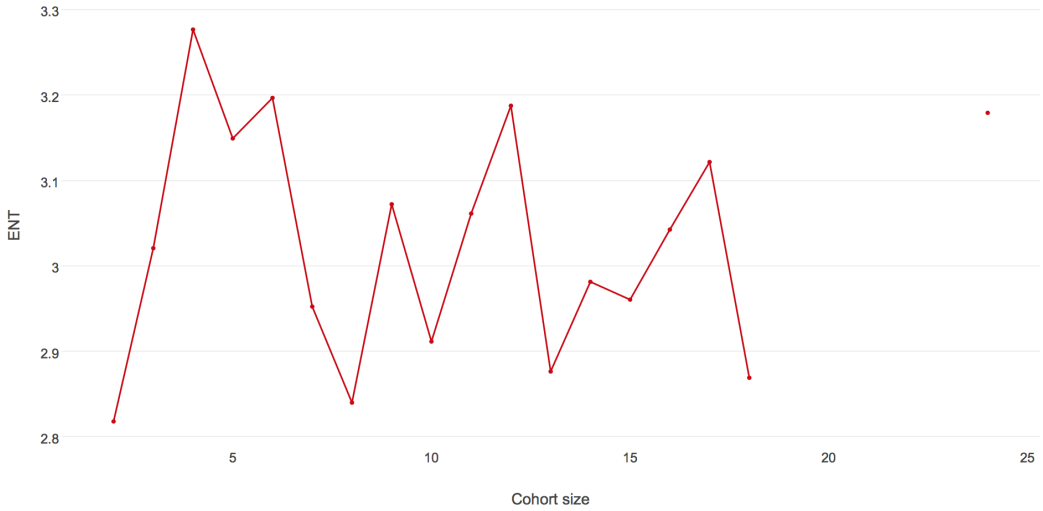


(a) Entropy of Pr [*score*|*match*] at different cohort sizes

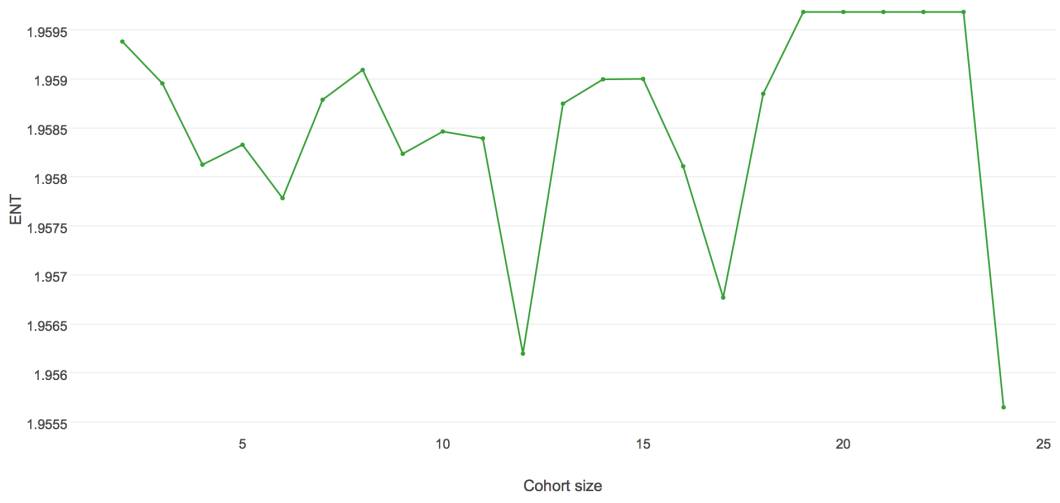


(b) Entropy of Pr [*score*|*nonmatch*] at different cohort sizes

Figure 6-7: Relationship between Pr [*score*|*match*] and Pr [*score*|*nonmatch*]



(a) Entropy of Pr [*score*|*match*] at different cohort sizes



(b) Entropy of Pr [*score*|*nonmatch*] at different cohort sizes

Figure 6-8: Relationship between Pr [*score*|*match*] and Pr [*score*|*nonmatch*]

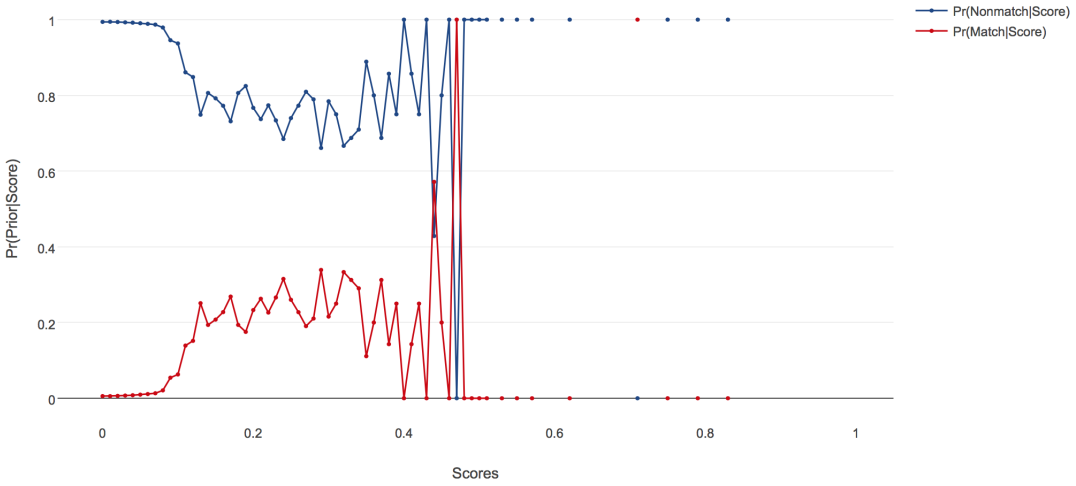


Figure 6-9: Probability of matches/non-matches given score

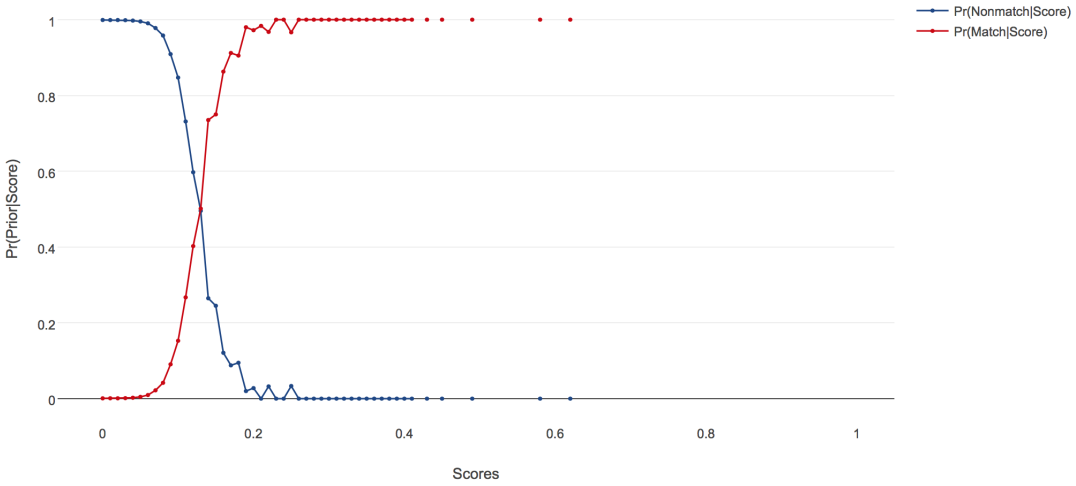


Figure 6-10: Probability of matches/non-matches given score

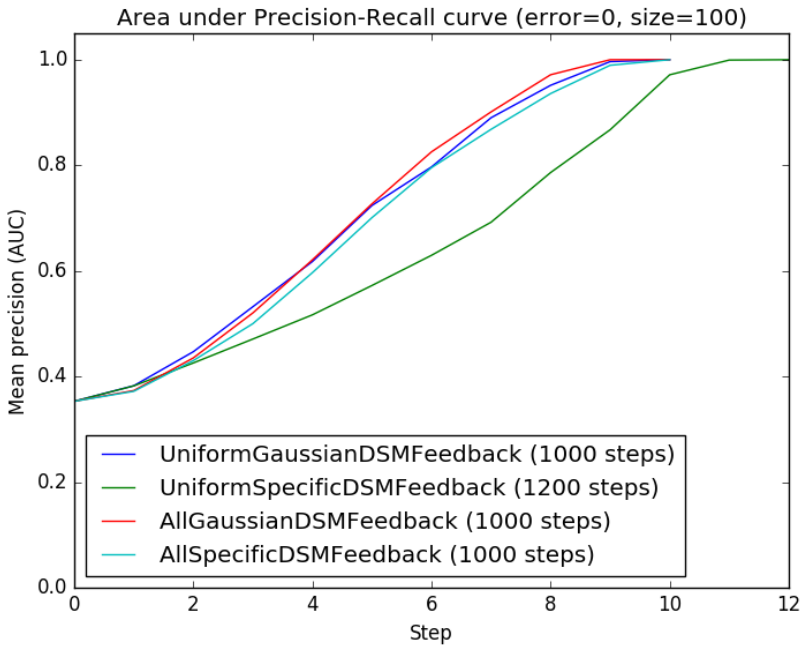


Figure 6-11: Mean Average Precision for the new sampling policies

Chapter 7

Convolutional Neural Network

We first describe the our new architecture in the System Overview (section 7.1). Then in the Experiments (section 7.2), we compare the performance of the current identification method within the existing pattern retrieval engine with that of the new architecture.

7.1 System Overview

In this section, we begin by presenting the architecture that outputs a yes-or-no answer to the question whether, given two images A and B, the animal in image A the same individual as the animal in image B.

The proposed architecture comprises two major modules shown in Figure 7-1.

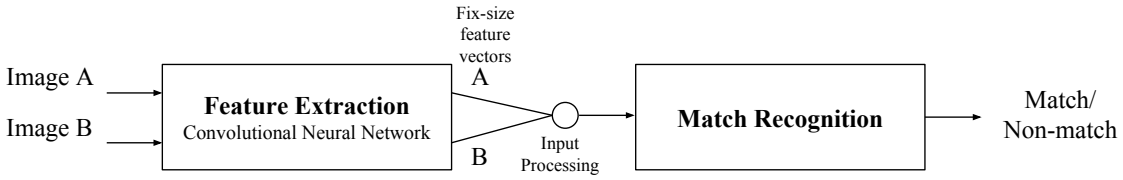


Figure 7-1: System Overview

1. **Feature Extraction**. We use a pre-trained CNN with the last fully-connected layer, the output layer, removed to extract fix-size feature vectors from the input images before feeding them into the *Match Recognition* module.

2. **Match Recognition.**

7.1.1 Feature Extraction

For the pattern retrieving engine such as Sloop, the system continually accumulates more data over time. The availability of abundant data enables the learning based methods to thrive over the engineered features because they can discover and optimize features for the specific task at hand.

Instead of hand-picking the features to learn the similarity matrices from the data, a convolutional neural network (CNN) will be used as a feature extractor similar to the solution proposed in [3] for face verification.

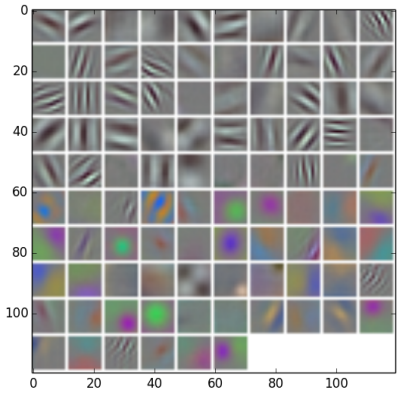
We use a pre-trained CNN, AlexNet by Krizhevsky et al. [7] trained on ImageNet [4], which contains 1.2 million images with 1000 categories. To convert the CNN into a feature extractor, we strip out the last fully-connected layer, which, in this case, outputs 1000 class scores for ImageNet classification task. According to AlexNet architecture, this would output a sparse 4096 dimentional vector for every images. The technique is called transfer learning [2].

We map all the available images as well as the input image to points in a low dimensional space (4096-D compare to 227×227-D) using the aforementioned CNN architecture.

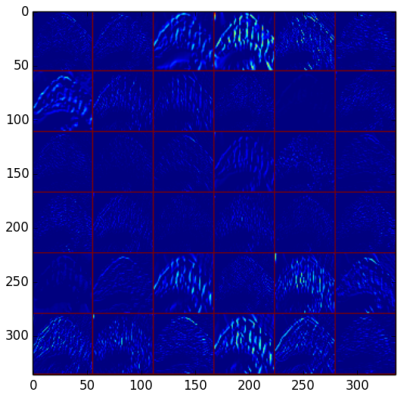
Pretained Convolutional Neural Network

The main advantage of using a pre-trained CNN to our application is that it is robust to geometric distortion. This is a desirable property since the position of the individuals in our dataset are not aligned. This enables us to accurately recognize the animal regardless of its position in the image.

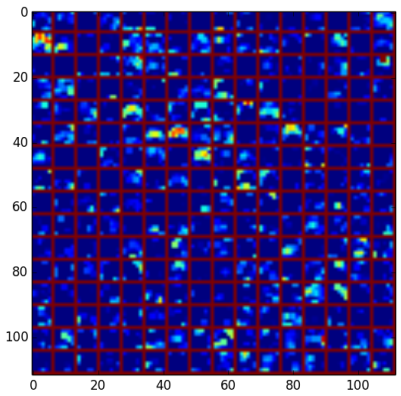
The translation invariant property emerges as a result of the contiguity in our pooling regions and the fact that we only pool features generated from the same hidden units [11]. Even though the property is desirable to reduce the translation noise, this disallows learning some features that are described by their relative positions. For example, individual A with a star-shaped spot right above its eye may be confused with another individual B who also has star-shaped spot underneath its eye.



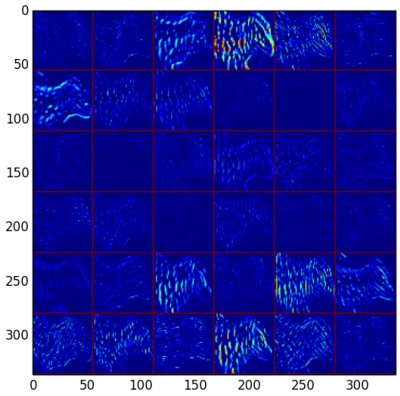
(a) The first layer filters, `conv1`



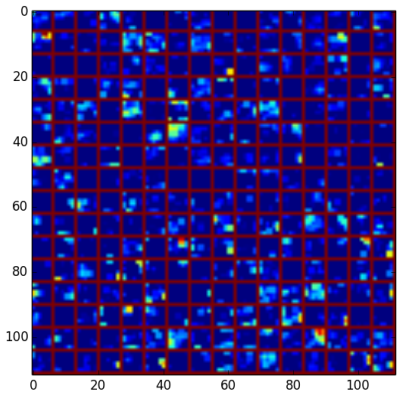
(b) The first layer output, `conv1`



(c) The fifth layer after pooling, `pool5`



(d) The first layer output, `conv1`



(e) The fifth layer after pooling, `pool5`

Figure 7-2: Visualizing Convolutional Network layers

The solution to such problem is to experiment with different pattern of pooling regions, and relax the parameter sharing scheme. Since we are using an existing pretrained architecture, it is hard to tailor to fit our data. The improvement can be included in the future work **??**.

Caffe

The CNN is implemented in python using Caffe[6], a deep learning framework developed by the Berkeley Vision and Learning Center (BVLC).

7.1.2 Match Recognition

Once we have the 4096-D vectors for all images generated from our CNN feature extractor, we transfer them into a second target match recognizer and train it on a target dataset and task. The match recognizer is a linear classifier that, given a pair of image vectors, decides whether the two images contains the same individual.

Input Processing

Upon receiving the input from the feature extractor, we process the input using one of the two method before passing it into the recognizer which outputs a binary label determining whether (A, B) is a match.

1. Concatenation

   Given two input image vectors $\vec{A}$ and $\vec{B}$ (flattened), we concatenate them horizontally so that the output $\vec{E}$ becomes

   $$\vec{E} = [\vec{A}\,\vec{B}].$$

   To make the order deterministic, we always start with the vector with smaller sum; otherwise, if equal sum, we compare each pair of corresponding entries in both vectors until we find an entry with a smaller element.

   We can visualize this as feeding a tuple of image vectors $(\vec{A}, \vec{B})$ to a linear classifier. This is equivalent to giving the classifier all the information we have

and expect it to figure out the optimal parameters. However, in this case there is no separator indicating that two image vectors are actually separated.

2. Computing the absolute difference

   We would like to find a similarity metric that represents how much two images differ from each other. In order to achieve that we come up with following metric:

$$\vec{E} = |\vec{A} - \vec{B}|$$

   Each entries $e_i \in \vec{E}$ is small if $\vec{A}$ and $\vec{B}$ belong to the same category, and large if different. Not only this encapsulates the desired numerical behavior, but it also eliminates the absence of separator problem.

Recognizer

Images can be uploaded to Sloop one by one or in a batch. One major difference is that the batch processing keeps the system weights constant while computing the error associated with each sample in the input, whereas the on-line version constantly updates its weights. For the real-time system like Sloop, we would prefer the on-line version of the classification algorithm given the same classification performance.

We have implemented following algorithms as our match recognizer:

- Linear support vector machine (SVM) with L2 regularization (squared Euclidean norm)
- SVM with radial basis function kernel (SVM-RBF), L2 regularization
- Perceptron (P)
- Passive-Aggressive with hinge loss (PA-I)
- Passive-Aggressive with squared hinge loss (PA-II)

We compare the performance of each classifier in the Experiments section (7.2). The classifier with the best performance will be selected for our final design.

7.2 Experiments

In this section, we first introduce the datasets used in the experiments, then present the detailed evaluation of the variants of our architecture and the comparison between the presented solutions.

7.2.1 Dataset

The training and testing data is available in the existing Sloop system. We base our experiments on both species of skinks: *Grand* and *Otago* mentioned in **??**.

For our experiments, each image from both databases is reduced to the size of 227×227 pixels so that it can be directly fed into the CNN architecture to speed up the feature extraction. Regardless of our preprocess, the CNN will resize the images into 227×227 to fit its input dimension. However, resizing is not necessary because practically our system should maintain its identification capability without regard to the variations in sizes, lighting, or background.

Partitioning

We divide the data from both databases into four datasets by animal species: Gr^L-I, Gr^L-II, Ot^L-I, Ot^L-II where Gr is Grand, Ot is Otago, L is left view. Notice that the right view is not be used for the experiments because the results should be similar to that of left view by symmetry. In addition, due to our memory limit during the processing, each dataset contains only 300 images of individuals whose image per individuals are greater than 11.

For the purpose of generating a test set that resembles the empirical input and another test set with images that are not seen before during training, we build our datasets using two different techniques.

- Gr^L-I, Ot^L-I represents the empirical input where the data in the test set can also be seen in the training set.

- Gr^L-II, Ot^L-II represents the test set with images that are not seen before during training. The set of individuals is split into three disjoint sets for training,

Table 7.1: Details of the train, validation, and test set for the four datasets

Name (NTotal)		NPairs	% Match	% Non-match
Gr^L-I (10878)	Train	6526	0.07	0.93
	Val	2175	0.07	0.93
	Test	2177	0.07	0.93
Ot^L-I (10878)	Train	6526	0.06	0.94
	Val	2175	0.06	0.94
	Test	2177	0.06	0.94
Gr^L-II	Train	2926	0.13	0.87
	Val	351	0.38	0.62
	Test	1035	0.25	0.75
Ot^L-II	Train	3240	0.10	0.90
	Val	561	0.31	0.69
	Test	561	0.26	0.73

validating, and testing. Each images in a set can only be paired with the images within the same set.

For each dataset, we partition the set of the *generated pairs* into three sets: train set (60%), validation set (20%), and test set (20%)respectively. We prevent overfitting problem by monitoring the models performance on a validation set, acting as a representative of future test examples. If the models performance ceases to improve sufficiently on the validation set then we test the model on teh actual test set.

Chapter 8

Analysis and Interpretation

8.1 Convolutional Neural Network

8.1.1 Recognition

To evaluate the initial results, we calculate precision, recall, and F1 values of the predicted results compare to the gold standard annotated by the biologists.

Input Processing

According to the proposed architecture in section 7.1, we have experimented with both method of input processing, namely, concatenation and computing the difference. The results are shown in table 8.1 and **??**.

All the classifiers perform poorly when feature vectors concatenation is used as the input processing method. This happens because of the *curse of dimensionality*, where there are too many unnecessary input features. Doubling the dimension of the input image vector overcomplicates the classifier parameters causing the downfall in the prediction accuracy. On the other hand, the classifiers tend to perform pretty well when using absolute feature difference as the input.

Considering the classification results in **??**, all the classifiers seem to perform equally well on the test data considering the F-beta score. SVM with radial bas kernel function yields the most stable performance with very high precision for both species. The offline algorithms slightly outrun all the online ones on average.

Table 8.1: Precision (P), recall (R), and F-score (F) of the classification results for the input processing with feature vectors concatenation

Dataset		PCT			PA-I			PA-II		
		P	R	F	P	R	F	P	R	F
Gr^L-I	Train	0.01	0.03	0.01	0.04	0.01	0.01	0.00	0.00	0.00
	Val	0.40	0.13	0.08	0.23	0.06	0.08	0.46	0.08	0.06
	Test	0.01	0.05	0.02	0.03	0.01	0.01	0.00	0.02	0.01

Table 8.2: Precision (P), recall (R), and F-score (F) of the linear classifier with absolute feature difference on train, validation, and test set of the four datasets

Dataset		SVM			SVM-RBF			PCT			PA-I			PA-II		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Gr^L-I	Train	0.91	0.21	0.34	0.97	0.18	0.30	0.40	0.26	0.32	0.57	0.31	0.29	0.57	0.29	0.30
	Val	0.95	0.23	0.37	0.99	0.19	0.32	0.58	0.37	0.45	0.65	0.48	0.38	0.67	0.44	0.39
	Test	0.82	0.20	0.33	1.0	0.21	0.35	0.42	0.24	0.30	0.54	0.33	0.28	0.52	0.29	0.28
Ot^L-I	Train	0.64	0.24	0.32	0.88	0.21	0.34	0.45	0.32	0.29	0.37	0.16	0.23	0.36	0.27	0.30
	Val	0.81	0.42	0.51	0.90	0.24	0.37	0.55	0.54	0.42	0.89	0.30	0.39	0.65	0.55	0.59
	Test	0.66	0.26	0.33	0.90	0.20	0.32	0.46	0.39	0.33	0.36	0.17	0.23	0.42	0.32	0.36
Gr^L-II	Train	0.92	0.35	0.50	0.99	0.21	0.34	0.99	0.27	0.43	0.89	0.48	0.63	0.83	0.57	0.67
	Val	0.79	0.20	0.32	1.0	0.20	0.33	0.93	0.19	0.32	0.76	0.21	0.33	0.66	0.24	0.36
	Test	0.85	0.22	0.35	1.0	0.18	0.30	0.92	0.19	0.31	0.86	0.20	0.32	0.69	0.22	0.34
Ot^L-II	Train	0.57	0.29	0.38	1.0	0.22	0.38	0.53	0.70	0.60	0.59	0.78	0.67	1.0	0.25	0.40
	Val	0.57	0.68	0.62	1.0	0.24	0.39	0.53	0.37	0.43	0.56	0.30	0.39	0.97	0.19	0.32
	Test	0.57	0.28	0.38	1.0	0.19	0.32	0.43	0.36	0.39	0.50	0.32	0.39	1.0	0.22	0.37

In the system such as Sloop, we would like to have a high standard on the individuals marked as a match becuase mistakes are extremely detrimental. It can easily propagate over the whole database so we would like to select a classifier with high F-beta score, preferably with high precision. In general, SVM-RBF is a very safe choice. However, the classifier with highest F-beta score are listed as following:

- Gr^L-I SVM-RBF
- Ot^L-I PA-II/SVM/SVM-RBF (Even though PA-II yields highest F-score, its precision is very low.)
- Gr^L-II SVM
- Ot^L-II PCT/PA-I/SVM

Species-wise, classification results for Otago skinks seem to have higher recall, but lower precision, than that of Grands despite the similar proportion between the matching and non-matching pairs in training, validation, and test sets. This implies that the feature vectors of Otago are pretty similar to one another in the classifiers' point of view. The similarity results from the visual patterns of the species, which are less detailed compared to Grand.

Bibliography

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[2] Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664. Morgan Kaufmann, 1995.

[3] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 539–546, Washington, DC, USA, 2005. IEEE Computer Society.

[4] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[5] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. *CoRR*, abs/1405.5769, 2014.

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[11] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, and Caroline Suen. Feature extraction using convolution. Accessed: 2016-09-08.

[12] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000.

[13] Randy Westlund. Sloop 4.7 user manual. Accessed: 2016-09-12.