A System for Secure and Categorized Video-Sharing

Neil Prakasam

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Masters of Science

University of Washington

2023

Reading Committee:

Brent Lagesse

Marc Dupuis

Program Authorized to Offer Degree:

Computing and Software Systems

University of Washington

Abstract

A System for Secure and Categorized Video-Sharing

Nathaniel Prakasam

Chair of the Supervisory Committee:
Brent Lagesse
Computing & Software Systems

Online video sharing is a phenomenon which continues to be increasingly utilized by the entire population. Preserving the privacy of videos shared online is of utmost importance, but there is one use case that hasn't yet been covered by current mainstream video sharing platforms. The goal of this project was to provide the ability to categorize whether multiple videos are of the same event, so that users can share them only amongst others who were also present at the event and have video evidence. The main method of categorization will be through DNA sequencing, where video files will be converted into literal dna in order to be categorized into 4 categories. This includes those that are of the same event, space, activity, or are completely different videos. The research has shown rather lackluster results that could potentially be further optimized to categorize videos between the 4 categories, let alone whether or not they are of the same event. This paper will introduce and implement multiple methods of doing so, as well as set the stage for future exploration.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1 INTRODUCTION

Online video sharing is a phenomenon which continues to be increasingly utilized by the majority of the population. This in turn has led to a greater demand for mobile device manufacturers to install larger digital storage and higher-resolution cameras into their products every year. “More than 54% of internet users post original photos/videos online that they themselves have created” [3]. With the growing awareness of general online privacy, users are re-evaluating how they would proceed to share their videos only with certain people. One of the main ways is to simply send videos to a person or group chat through instant messaging. However, this can be inefficient for sharing to another category of people this study will refer to as “approved users”. These are currently some mainstream ways of specifically “posting” videos on popular platforms so that only approved users can see it:

- YouTube/Vimeo: The user can set it to “unlisted/private” so it will not show up publicly in a YouTube search, and distribute a secret link which will be the only way for other users to view the video [16].
- Facebook: The user can post the video to their own profile and set the privacy to “public, friends-only, only me”, or manually select users whom the video will be visible to on their profile and newsfeed. There is also the possibility of a “private” or “closed” group where the video will only be visible to other users in the group.
- Instagram: The user can set their profile to private so that only approved followers can view your content

One type of “approved user” would consist of only people who were also present at the event where the video was taken, although they may or may not fall into the already existing

groups/group chats as these settings aren't always created just for events. Therefore, there is a need for a service that would be able to host and share videos securely among a group of people who were present at the same place a video was taken. This project aims to create a system for said scenario through the use of bioinformatics computation.

The system utilizes simultaneous observation [30] to accomplish privacy-preserving, semi-automated video-sharing. The observation techniques were employed to find similarities between multiple videos. Once a certain threshold of similarities is reached, the video would be visible to other users whose videos have been categorized as "similar" to each other. Potential stakeholders of this application would be anyone wishing to share videos securely online and any interested individuals in the cybersecurity research community. This system has the availability to be open-source and used as a tool for other applications to distribute videos in a private and secure way.

The main result is a system that attempts to efficiently categorize videos into 4 different classifications but fails in accuracy. This new efficiency is obtained through the use of bioinformatics techniques used for DNA Sequencing. After converting videos to DNA files, a Machine Learning model was trained on multiple datasets of said files and tested for video categorization. After preliminary testing, multiple modes of optimization were applied to both the DNA Mapping algorithms and Penalty Assignments in order to improve categorization accuracy. The previous SSO approach produced high accuracy but was slower and had room for optimization. The resulting categorization effectiveness of this approach leaves more to be desired at a mere 60% accuracy, but yields a significant increase in performance and can be applied to larger datasets.

CHAPTER 2 BACKGROUND

The main focus of this study is to search for an improved method of video similarity identification. The previous iteration of this project used the Kullback-Leibler divergence (KLD) algorithm, along with a neural network to boost accuracy [30]. Such methods require copious amounts of resources for processing. This system proposes the use of bioinformatics techniques in order to attempt more efficient video categorization.

```
 2 TTHHHTHHHCTTHHTTCCTCHHHHHHTHHCCTTHHHHTTGTTTHHHHHHC     51
   ||   |   .||  |||.|          .|||    ||||||      .
 4 TTHTCTHGTTTTCTTTCGT----------TCTT--TTTTGTTTCTTTCCG     41
```

Figure 2.1: Global Alignment

```
 9 TTCTHHTTTTTHCTHTHTHTHTTCTTTTHHHHTHTTTHHTTHCTHTT---      5
             || | | | ..|||.|
           5  CTCTGTTTHHCCTTGT 20
```

Figure 2.2: Local Alignment

Bioinformatics is a field where programming techniques are applied in order to perform operations on and organize large sets of molecular data [11]. The area we apply to this project in particular is the concept of sequence alignment, where DNA strands are compared with one another in order to determine if there is level of similarity required to be classified as a match [12]. There are two types of sequence alignment to consider: global and local. Global alignment computes the strongest alignment between all the characters of the sequences, whereas local does the same for only the subsets of characters in each sequence which have the strongest similarity

as shown in figures 2.1 and 2.2 respectively. Pairwise alignment uses multiple methods in order to find the best global or local alignment(s) between 2 sequences. In essence, our datasets will be converted from video files to DNA sequences and thoroughly analyzed and categorized using these alignment methods.

"The 'Best Local Similarity' is related to other algorithms for sequence comparison and alignment:

$$Score(i,j) = MAX(Score(i-1, j-1) + Sim(aa_i, aa_j),$$
$$Score(i— 1, j) + gap\ penalty,$$
$$Score(i, j— l) + gap\ penalty)$$"[9]

The most widely used global sequence alignment is Needleman-Wunsch [13], known for its ability to find the more robust global alignment. The algorithm goes through all the possible alignments for the characters in both sequences, and provides the analysis of the longest, most consecutive string of matches. Similarly for local alignment, there is the Smith-Waterman algorithm which is slightly varied. The biggest difference between the two is that the overall score in Smith-Waterman can be no less than zero, whereas Needleman-Wunsch can be negative.

The scoring scheme determines the factors of the sequences that will be considered for the computation of the overall score. There are 3 possible outcomes in a comparison of characters from the 2 different DNA sequences: a correct match, a mismatch, or a gap. For every character match between 2 sequences, usually a point is added to the score. For every character mismatch between 2 sequences, a certain number of points is deducted to the score. For every extra character in either sequence that interrupts a matching set, a certain number of points is deducted to the score [1]. In our use case, we modified the scoring scheme in order to account for

the foreign type of data that we would be using. That is, using videos converted to DNA rather than authentic DNA sequences.

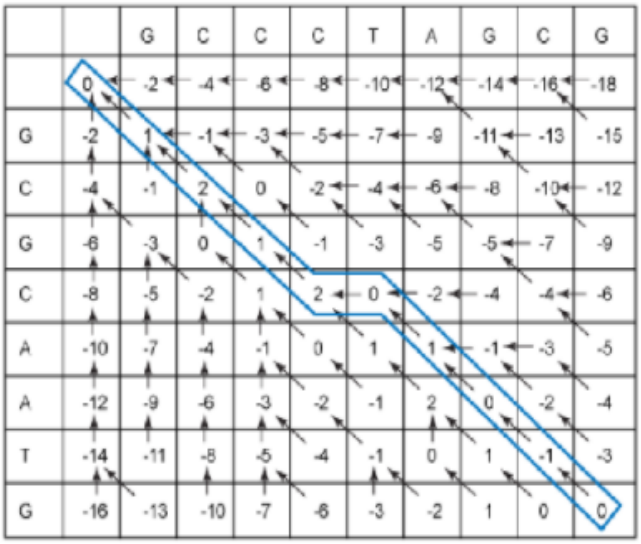

Figure 2.3: Global Alignment Table [18]

Figure 2.3 shows an example of a global alignment table. The table is completed by marking the score of one of the characters from both sequences with their corresponding meeting box. A path can then be charted in order to add up to the highest possible score.

CHAPTER 3 RELATED WORK

There are a few other studies that have been developed for the purpose of video categorization, however none of them have conducted the categorization through the specific application of bioinformatics. The following will summarize the efforts made in other studies with emphasis on performance and accuracy measurements as applicable.

This project is a continuation of another project under Professor Lagesse which involved detecting hidden webcams with similarity of simultaneous observation (SSO) [30]. For the SSO research, it lended insight into various techniques such as KLD, Bhattacharyya distance, and Cramer distance. These outputs are more usable in FHE[6][30]. The project achieved an impressive accuracy of 87%+ and an F1 score of 0.88 [30]. In comparison, this system is able to convert videos to DNA files in around 2.54 seconds per minute of video. The problem with this method is that it could not be applied on large datasets without enormous wait times. Therefore, this is a motivation for increasing processing efficiency to search for videos in much bigger databases. A business application for this would be to verify data retention policies for companies, proving that similar data does not exist in their database after the period noted in their policy.

Students at the University of Washington Bothell conducted the first iteration of this experiment led by Professor Brent Lagesse [6]. The work proposed to run an algorithm over a dataset that was fully homomorphically encrypted, which allows for performance improvements. There were multiple stages of processing for this experiment which would each add to computational time. The results were satisfactory with an F1 score at an average 97%, although the shortcoming of this method was the computational time. The algorithm was able to process a video in 1.5 seconds [6]. However, the use of FHE added an additional 17 seconds of processing

time. Accordingly, the current iteration in this research primarily seeked to improve the computational efficiency while sacrificing minimal precision. Along with the new method of video similarity scoring, this project will also dive deeper into categorization by introducing 2 new categories to threshold.

Nathan Perkins of Boston University created a system of Convolutional Neural Networks in order to generate features for video near-duplicate detection. However the application for the project was different in that the purpose was to detect “common distortions, such as small rotations, cropping, re-encoding and changes to color and brightness” [5]. Its performance came down to >16 seconds of processing to identify features per one minute query video and “naive implementation of this search process, querying took 7-43ms, depending on the number of potential matches.”

D Wang created a study in order to test Multi-View Action Recognition [26]. They used a view classifier among multiple CNN layers in order to detect the likeliness of each video belonging to the same subject. They achieved an overall accuracy of 88% but the performance metrics are unknown.

CHAPTER 4 METHODOLOGY

4.1 Bioinformatics Approach

This section provides details of the different aspects of the Bioinformatics Approach, beginning with the selected algorithms for DNA Sequencing, which has been used to develop our solution.

4.1.1 DNA Sequencing:

The ultimate application of bioinformatics into this project is to conduct alignment on videos just as it is done with DNA sequences. However, the videos needed to be converted into DNA strings in order for the alignment tools to be able to process them. The process of determining the sequence of a DNA molecule is simply called DNA Sequencing. For this approach, mp4 files are converted into dna files for processing using algorithm createArray. Essentially, the algorithm of createArray (Figure 4.1) works by decoding the frames of the video into an array of bytes. These bytes are then assigned 1 of 4 values for each threshold increment of .25 from 0 - 1. These values by default are H(anything less than .25), T(.25-.5), C(.5-.75), or T(.75- anything).

```
91  def convert_byte_array_to_DNA_seq(byteArray):
92      #byteArraySum = sum(byteArray)
93      normalizedByteArray = list()
94      byteArrayMin = min(byteArray)
95      byteArrayMax = max(byteArray)
96      print("min" + str(byteArrayMin))
97      print("max" + str(byteArrayMax))
98      dnaByteArray = list()
99      for i in range (0, len(byteArray)):
100         print (str(byteArray[i]))
101         normVal = float(byteArray[i]-byteArrayMin)/float(byteArrayMax - byteArrayMin)
102         dnaChar = ''
103         if normVal < 0.25:
104             dnaChar = 'H'
105         elif normVal < 0.50:
106             dnaChar = 'T'
107         elif normVal < 0.75:
108             dnaChar = 'C'
109         else:
110             dnaChar = 'G'
111         dnaByteArray.append(dnaChar)
112     return dnaByteArray
```

Figure 4.1: CreateArray Code Snippet

Multiple processing frameworks were used in order to have various instances of data to parse. The initial framework used for processing DNA strands was the Water package from the NCBI BLAST Application for Ubuntu [27]. BLAST stands for Basic Local Alignment Search Tool and hence the name, will determine the local similarity for sequences. Another framework was Bio.Align/Biopython [29] which also runs and installs directly from Ubuntu/Command Line, performing a pairwise type alignment. However, one other tool used was EMBOSS Needle [28], which had an online interface which accepted DNA files and delivered the scoring reports with pairwise alignment as well. The use of each and every one of these tools allowed the representation of results derived from diversely implemented algorithms for alignment. This in turn would lower the margin of error because all of the possible routes of sequencing are explored.

4.1.2 Testing Methodology

The video tests were separated into 4 categories:

1. Videos that overlap in time and space
    a. This refers to videos that are of the exact same event, but from different perspectives. This classification of multi-angles is most commonly used in many applications, such as a “system that learns to combine event predictions from multiple video streams” [2]
2. Videos that overlap in Space
    a. This refers to videos that are taken in the same place, with overlapping fields of view, but at different times that do not ever overlap.
3. Videos that don’t overlap at all
    a. This refers to videos that are taken in different places, and the times are not considered at all.
4. Videos of the same activity.
    a. This refers to videos that are different in place, but are of the same activity.

4.2 Score Normalization:

Score Normalization is the process of converting one’s score with a holistic approach, the consideration of multiple factors in order to make a proper comparison to multiple specimens [32]. When determining an approach for normalization, it was crucial to keep the overall length of the strings as one of the factors as it would be the most relevant and measurable common denominator between. Various methods of score normalization were attempted, including the multiplication and division of the scores with the various lengths, as well as the duration of the

videos themselves. Of these methods, the most viable approach was to divide the yielded score by the length of the shortest string in order to maximize the normalized score at the same time while remaining.

The final conversion for normalization became:

Normalized Score = (Score / Length of the Shortest DNA String)

4.3 Gap and Mismatch Score Optimization

The next goal was to find the most ideal gap scores and mismatch scores. The most ideal scores would be those that yielded the highest overall scores for videos that overlap in time and space, which would be the self-done videos in this scenario. The original use case for this method was to compare a shorter clip to an extremely long length of footage to find a match in the correct section. In order to redirect to this use case, I concatenated some of the datasets into a single video in order to test this principle.

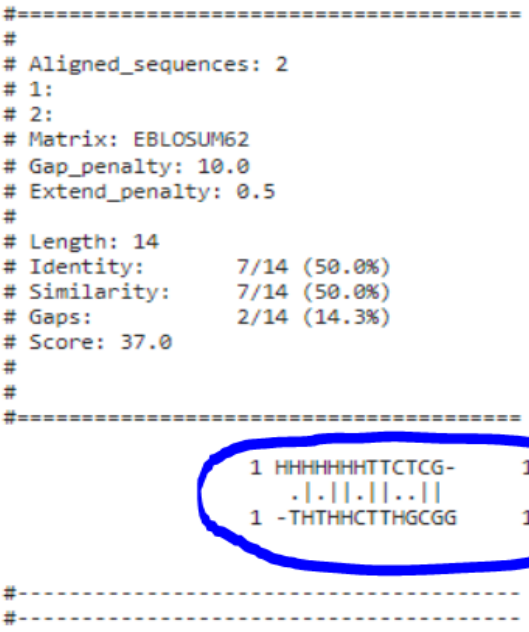

Figure 4.2: Example DNA Alignment

As shown in Figure 4.2, there were char matches scattered across the entire length of the DNA strings. In actuality, the objects in the videos first intersected in the 6th character but it is also worth noting that the backgrounds of the videos were similar as well.

4.4 Method Optimization

In order to improve the results and maximize accuracy rate for the method, there was an additional effort to utilize alternative formulas for a couple of the areas within the process of DNA Alignment.

4.4.1 First Area of Improvement: Mapping

The first stage is the mapping of the (MP4) videos to DNA Sequencing characters. The following is a method from the createArray code which was the area for optimization in this stage.

4.4.1.1 - Squaring Idea:

The first idea was to implement the use of a squaring function onto the float numbers before mapping. The reasoning for this is to amplify the differences between the characters so that their variance would cause more accurate readings. For example: The difference between 2 and 3 is 1, but when squaring them, the difference is 5. To implement this idea, “normVal” defined on line 101 was squared so that some values would be mapped differently were they not squared.

4.4.1.2 - ByteDivision4 Idea:

Another idea would be to perform operations on the raw video before their conversion to bytes. This would prevent the additional noise potentially caused by converting it to a float number first. In the original method, the float value is partitioned into one of four buckets. To modify this for bytes, the buckets were separated from 0 to 255 in bytes. However, most of the test files yielded characters that were almost all of (H), meaning they were in the range of 0 to 64. To combat this, the algorithm was modified to get the entire range of the data, split the difference of min and max into 4, and assign a character for each as shown in Figure 4.3. This still did not solve the problem and continued to yield H’s, meaning that the characters were still not divided enough. It was then clear that it would be beneficial to split the difference by 4 again, essentially using 1/16 of the difference in range to assign each character.

```
dnaByteArray = list()
for i in range (0, len(byteArray)):
    print (str(byteArray[i]))
    normVal = byteArray[i]
    dnaChar = ''
    if normVal < (min(byteArray)+(divi/4)):
        dnaChar = 'H'
    elif normVal < (min(byteArray)+(divi/2)):
        dnaChar = 'T'
    elif normVal < (min(byteArray)+(divi*(3/4))):
        dnaChar = 'C'
    else:
        dnaChar = 'G'
    dnaByteArray.append(dnaChar)
```

Figure 4.3: ByteDivision4 Idea Code

4.4.1.3 - ReduceRepetition Idea:

The next idea was to condense all the consecutive character repetitions into one for the sake of computational efficiency.

Example: [A, C, C, C, D] -> [A, C, D]

While one would think this concept does arguably defy the principle of traditional DNA Alignment in that it theoretically cuts out characters that could have a potential match, it compensates for this fact by simply applying the same measures to both DNA strings.

4.4.1.4 - Sigmoid Idea:

The next idea was to employ the sigmoid function in order to again amplify variation among the DNA values. The logistic function has found many applications throughout mathematics. It was originally proposed by Pierre François Verhulst between 1838 and 1847 [22]. Originally, the function was used to model population growth and decay. There are two important properties of the function that are being leveraged in this example. First, the existence of two horizontal asymptotes that cause the function to approach a stable value at both positive

and negative infinity. Second, the rate of change goes to zero as the function approaches these asymptotes. Because of these two properties, we can cause values to accelerate in some areas and stay relatively stable in others.

The function proposed, however, is a piecewise modification of the original logistic formula that splits our input range into two distinct subranges. Specifically, we split the inclusive input range [0,1] into two subranges, [0, 0.5) and [0.5, 1].

The function is defined as:

$$x < 0.5 : 0.5/(1 + e^{(-20(x-0.2))})$$

$$0.5 <= x : 0.5/(1 + e^{(-20(x-0.8))} + 0.5)$$

This causes the rightmost horizontal asymptote of the first function to approximately align with the leftmost horizontal asymptote of the second one. Effectively, this causes the following transformation: values less than 0.5 are mapped to values higher than the input and values greater than 0.5 are mapped to values less than the input. This causes all values on the input range to attract towards the combined asymptote.

Overall, this creates acceleration points. Theoretically the squared method worked because it pushed the higher values away from everything else. And everything lower than that is clustered together. This was implemented using the SKLearn Sigmoid API call.

4.4.2 Second Area of Improvement, Penalties:

One key difference between the method used and the standard method for dna matching is the measure of fault between missed matches. In the standard approach within biology, all misses are treated the same and yield an equal score. However, this approach seeks to explore and account for the difference between the characters through their numeric byte range. The idea

is to increase the penalty depending on how large the miss is, and that may provide better results. For Example: H= <.25, T=..25-.5 C=.5-.75, G= >.75. If a value (H) is being compared to value (G), it should be penalized more than if it were compared to C or T since they are farther apart in range.

For each iteration of the concept, there will be an increase in penalty for every increase in misses from the categories/buckets. The iterations include Linear, Quadratic, and Cubic. Table 4.1 shows how the penalties were calculated and applied for each iteration.

Miss from Category	Default Penalty	Linear Penalty (x)	Quadratic Penalty (x^2)	Cubic Penalty (x^3)
1	1	1	1	1
2	1	2	4	16
3	1	3	9	27

Table 4.1: Penalty Optimization Table

CHAPTER 5 RESULTS

5.1 Evaluation (Datasets):

The following is a description of the datasets used for testing. They were each selected with the criteria of including multiple videos taken of the same subject, space, and time with different angles. All of these datasets provided a satisfactory amount of results for each category referenced above, and were used to train the ML model that would be used to predict future categories. The experimental setup was the Self-Done videos

5.1.1 Self-Done Videos

The initial dataset was created using just 2 angles. I set up 2 cameras in my room and filmed myself for an hour. The activities captured included entering/exiting the room, walking, eating, sitting down, using the computer, etc. One of the videos was preserved at 60 minutes in length and the other was split into 60 videos of 1 minute lengths. The videos were first tested using the framework WATER. The use of this control dataset is to test the effects of varying gap and mismatch scores in order to maximize the contrast between videos that are the same and not the same.

5.1.2 VIRAT (Video Image Retrieval and Analysis Tool) Video Dataset

Contributed by students at Columbia University [8], this dataset was created with Computer Vision in mind, particularly for testing and experimenting in continuous visual event recognition. The advantage of using this dataset is that it features real-world footage of diverse environments including several parking lots and various locations on college campuses. The data

also included aerial footage, which was not incorporated into testing as it is relevant enough to this project's primary use case of event recognition [8].

5.1.3 YKS EgoCentric Dataset

Another dataset was used from Yonetani at the University of Japan [17] which includes a number of first-person and second-person views of the same event. This dataset was created in order to analyze facial reactions and micro-reactions, and the subjects are more based on only people rather than the interaction of objects and people.

5.1.4 Multiple Cameras Fall Dataset

This will be referred to in this project as Falling Dataset [21]. This video dataset was developed to test fall-detection in different environments compared to the scenarios found in elderly homes. It offered multiple perspectives of the same event, in the same room indoors. However, the activity in each sample set were more or less the same, being any given action such as walking or running, etc, and ending in a fall onto a mattress.

5.1 Self-Done Dataset (Control) Results:

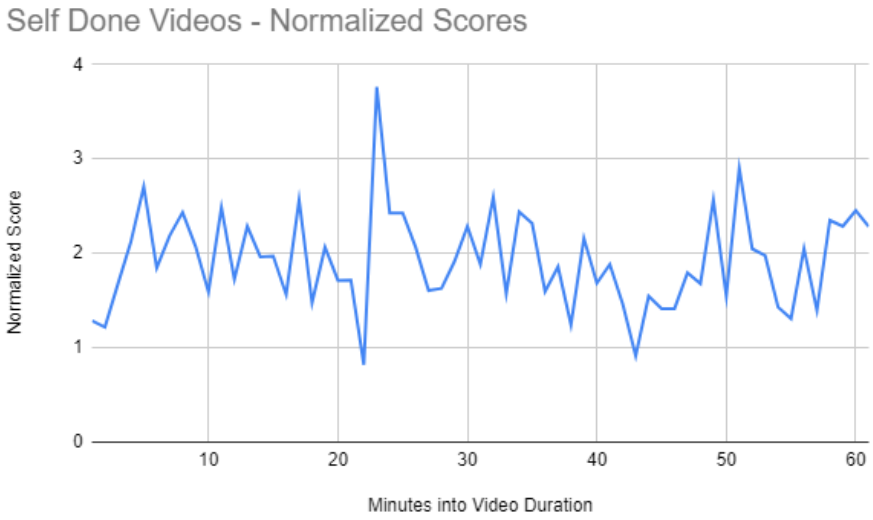

Figure 5.1: Self-Done Dataset Results

For the initial evaluation, we wanted to have a control set score for a category 1 set of videos. This control set consisted of 2 video angles with a duration of 1 hour. This set of videos fell into category 1, sharing both time and space. In order to aggregate the global alignment capabilities, one of the videos was split into 60 separate files (one per minute). Consequently, there were 60 tests performed, which provided a multitude more points of data than comparing just two videos together. The average normalized score for the 2 angles of the self done videos was 1.92 (using Test Results - VidSplit vs IMG5050) shown in Figure 5.1.

5.2 Multi-Datasets Testing Setup:

The remaining video datasets were tested using the 3 tools: Water, Biopython, and Emboss Needle. The following figures show the average normalized score for the video couplings in each category.

5.3 VIRAT Dataset Results:

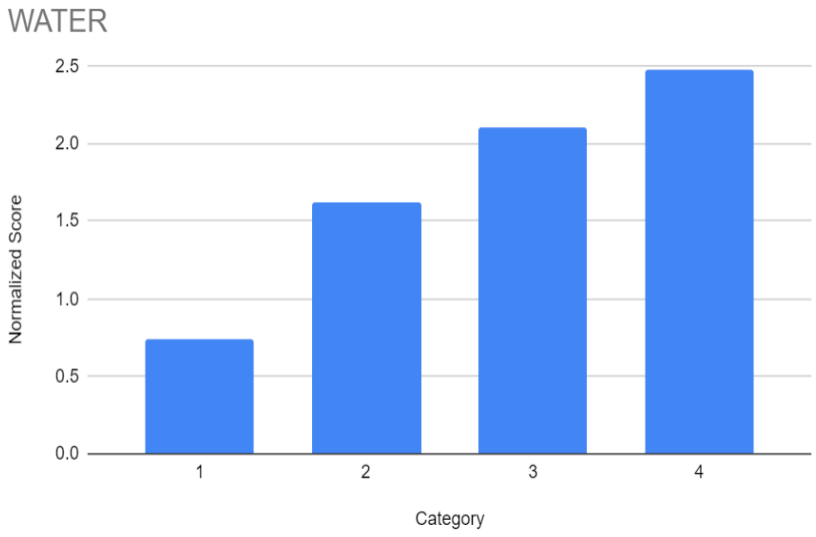

Figure 5.2: VIRAT Results Graph (WATER)

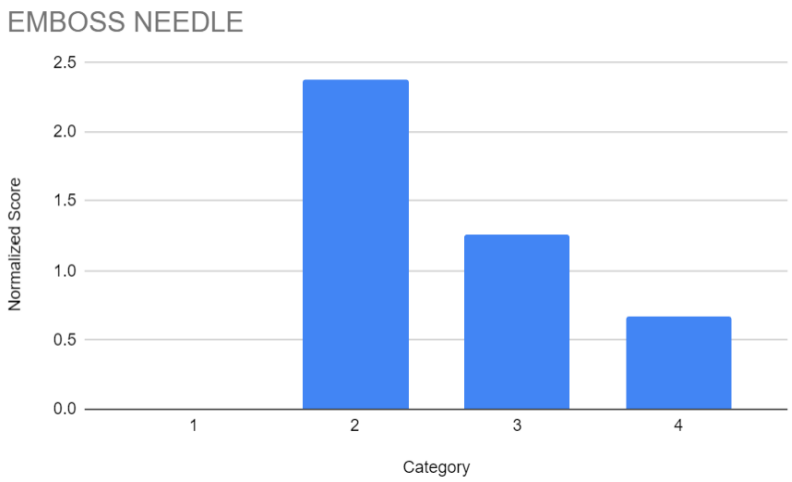

Figure 5.3: VIRAT Results Graph (Needle)

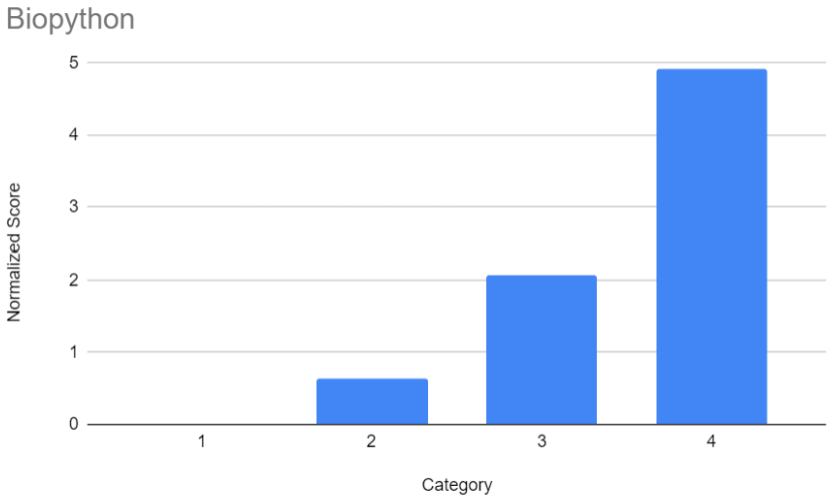

Figure 5.4: VIRAT Results Graph (Biopython)

For the VIRAT dataset (Figure 5.2-5.4), each of the tools returned relatively similar results in which the scores increased rather linearly with each video classification. This is the opposite of the results we are looking for. Since the category 1 data is returning the lowest score in similarity as it is close to 0 on average, it is counter intuitive to the hypothesis of having the highest possible score. These results could be attributed to the lack of data for Category 1 videos, as there were only 2 instances for testing while the rest of the categories delivered 4-5.

5.4 YKS EGO Motion Dataset Results

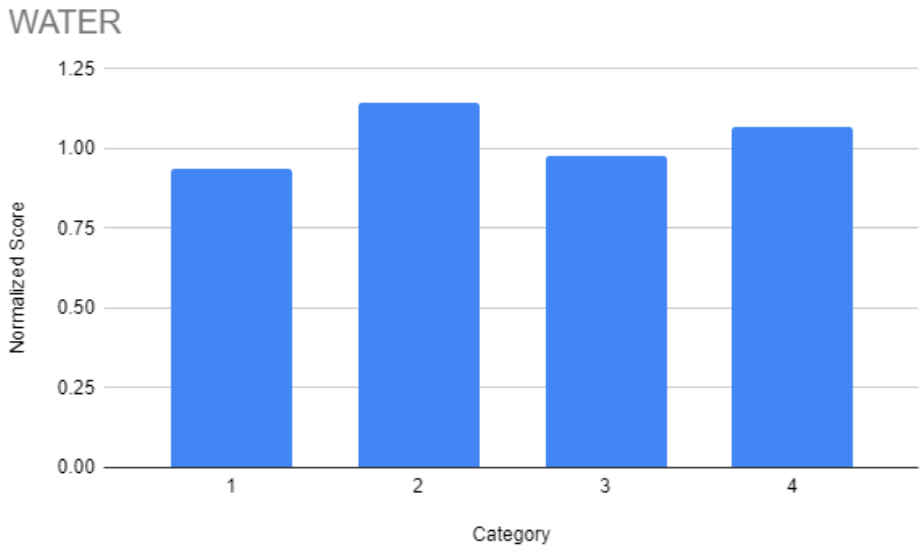

Figure 5.5: YKS Results Graph (Water)

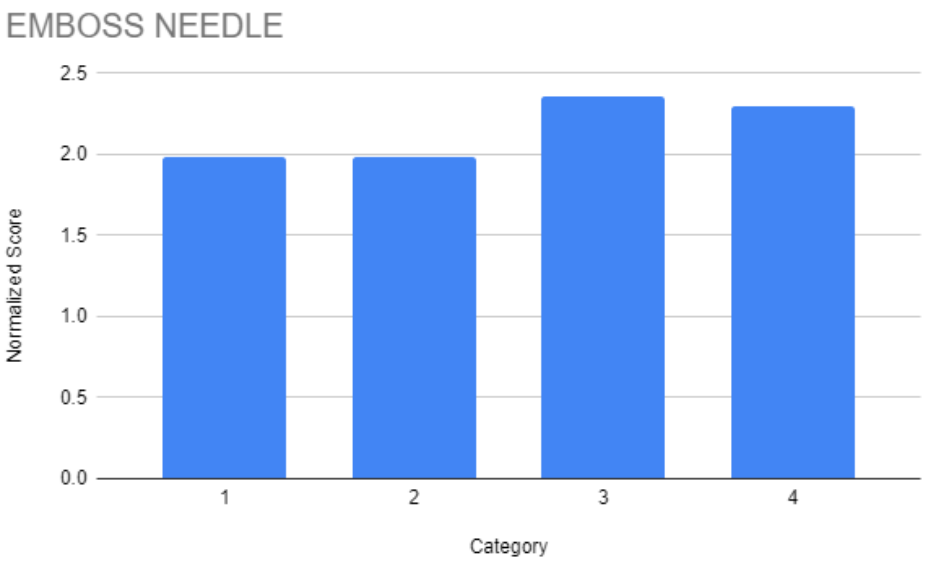

Figure 5.6: YKS Results Graph (Needle)

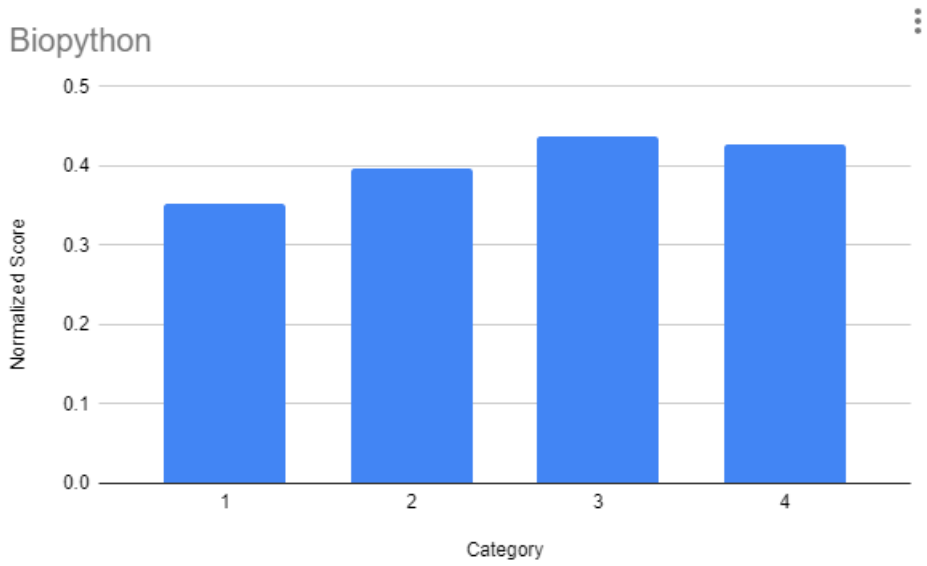

Figure 5.7: YKS Results Graph (Biopython)

The results of the YKS EGO dataset did not have much differentiation between the four categories as shown in Figures 5.5-5.7. For this reason, it would only be included in the training dataset. A possible explanation for this lack of differentiation would be that the videos in this

dataset are all taken in a similar setting, position, and angles, while the subjects are most diversified.

5.5 Falling Dataset Results:

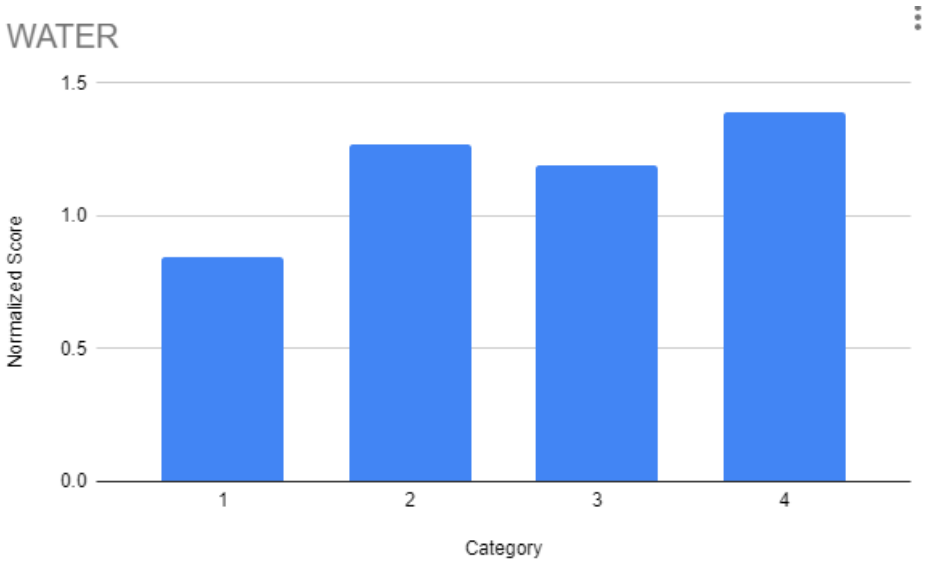

Figure 5.8: Falling Results Graph (WATER)

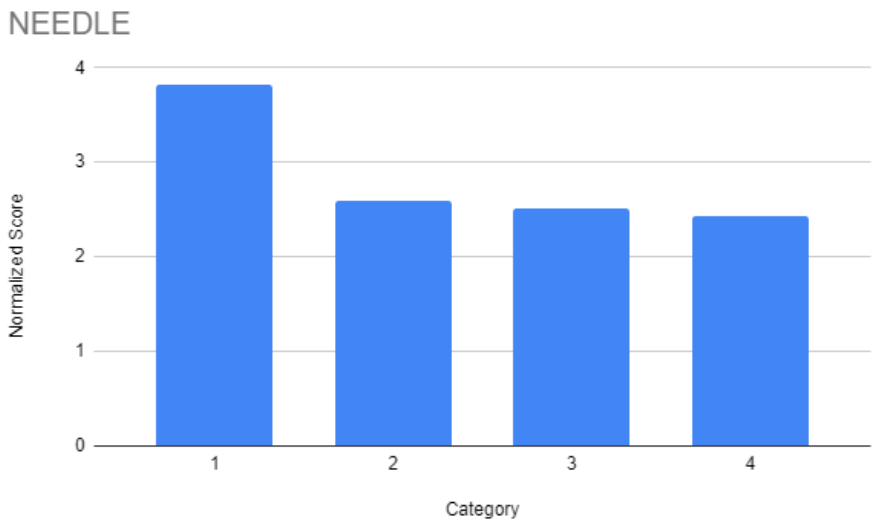

Figure 5.9: Falling Results Graph (Needle)

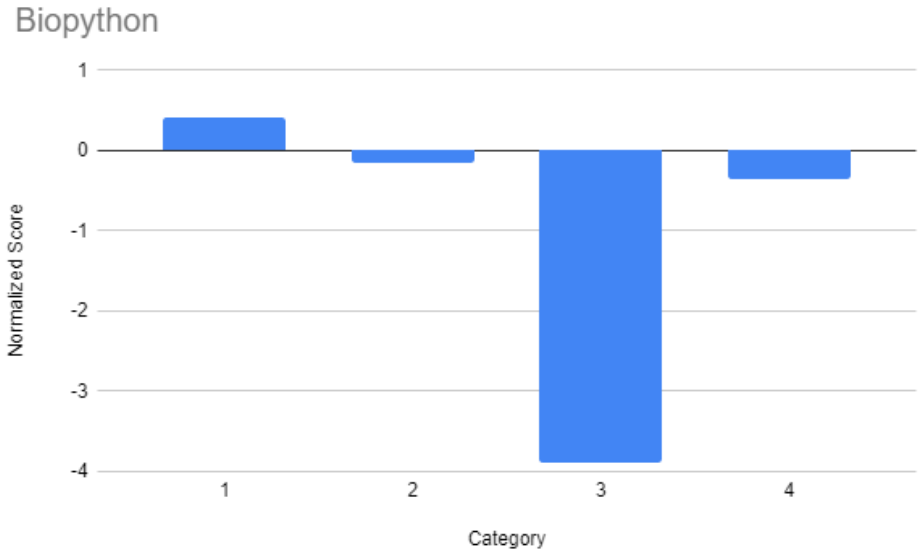

Figure 5.10: Falling Results Graph (Biopython)

The results of the falling dataset showed that Category 1 yielded the highest normalized score for the Biopython and Needle tools (Figure 5.8-5.9), while showing the opposite trend for the WATER tool (Figure 5.7). This is at least a start for aligning normalized scores for levels of similarity. A possible reason for the tools yielding different orders of scores would be that the

local alignment algorithm used for WATER interpreted the DNA for this dataset differently than that of the other tools.

5.6 Gap and Mismatch Score Tests:

One area to explore was to see the effects of manipulating the parameters available in the algorithms. These included gap scores and mismatch scores for the WATER tool. To test the effect of the gap score, I ran multiple tests for videos in category 1 and 3, incrementing the gap score by .1 with every test. The starting gap score was negative 5. Increasing the gap scores improved the overall score linearly for videos that were the same as shown in Figure 5.11. However, changing the mismatch score had little to no effect (Figure 5.12). For every increase in the gap score by 1, the normalized score increased by 54. This coincided with Taylor's results in which "most secondary structures [were] broken by gaps" [10]. As explained before, gap scores are usually employed to compensate for any "gaps" in string alignment, and are thus negative in order to detract from the overall score. However, the results of this test indicated that since only the datasets of video category 1 improved with the increased gap score and would thus improve the overall accuracy of categorization as it is directly related to the overall score. Using the video with the shortest length drastically increased the scores simply because the length is the denominator in the equation.

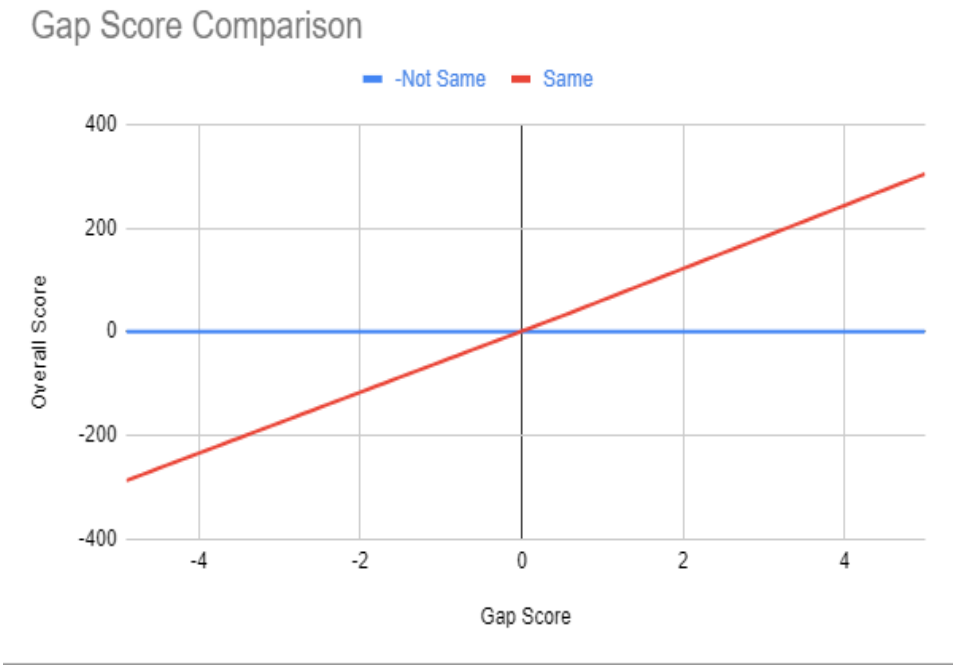

Figure 5.11: Gap Score Comparison Graph

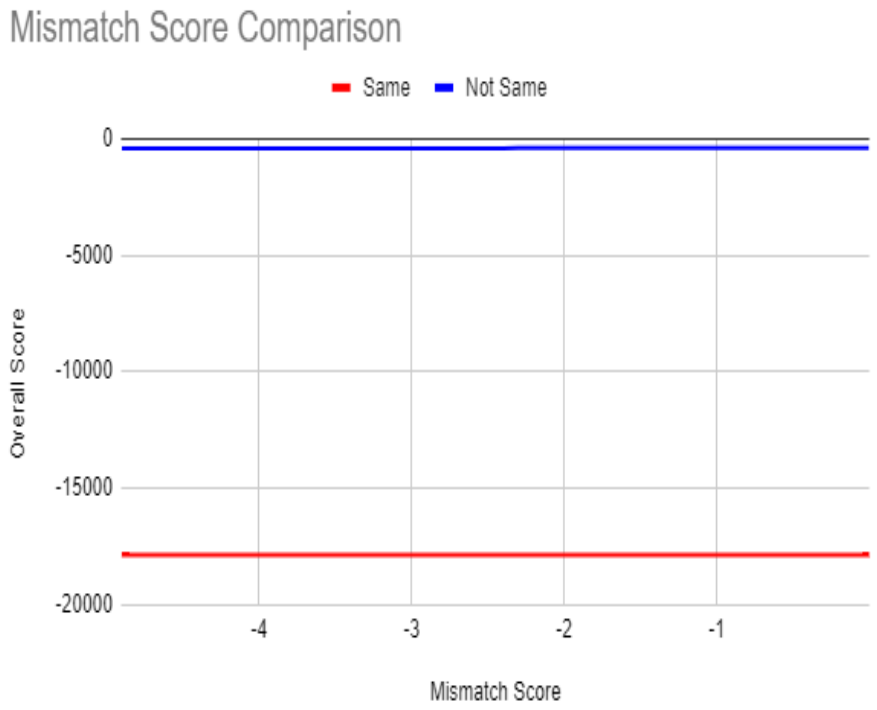

Figure 5.12: Mismatch Score Comparison Graph

5.7 Machine Learning Model Explanation:

In order to map the normalized scores to a category, a machine learning model was utilized and trained on the dataset results yielded by the NEEDLE tool. The model was used to create a prediction for all four categories. The ML model was a relatively simple model with 1 layer and neuron. The model used 'MEAN SQUARED ERROR' for the loss and 'STOCHASTIC GRADIENT DESCENT' for the optimizer. The reason that a simple neural network was used is because the network was being used on very 2 dimensional data such as the classification data results themselves and would only require 1 neuron. This would also boost performance time simultaneously. However, if the network was being used on the raw video files themselves it would make sense to use a more complex system for classification.

As shown in Figure 5.13, the ML model predicted a downward line to represent the transition between each category. Specifically, the equation was: $F(x) = 1.979 - .248x$. The algorithm was then used to predict the categories of subsets of videos based on their given normalized score from each tool.

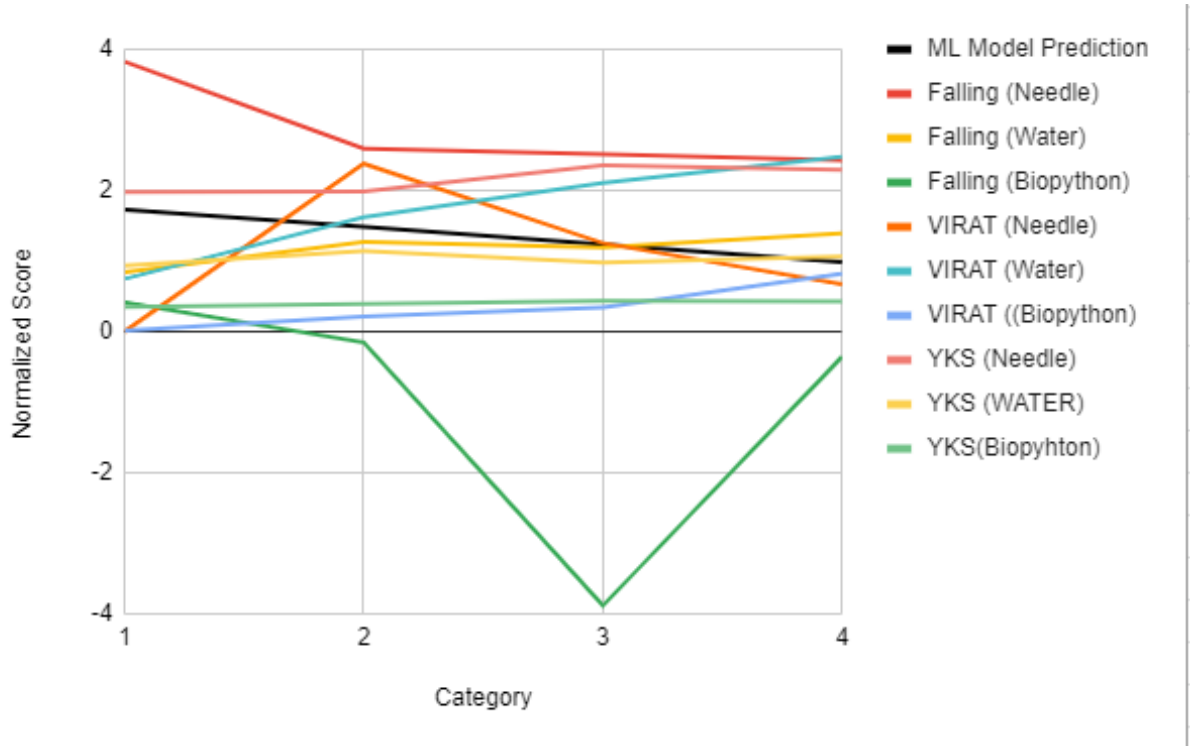

Figure 5.13: ML and Dataset Comparison Graph

	Dataset	ML Model Prediction	Falling	Falling	Falling	VIRAT	VIRAT	VIRAT	YKS/EGO	YKS/EGO	YKS/EGO
	Tool Used		Needle	Water	Biopython	Needle	Water	Biopython	Needle	Water	Biopython
Category	1	1.7314001	3.826923077	0.8461538462	0.4134615385	0.005139500734	0.7465753425	0.01486890995	1.981046695	0.9371165221	0.3524305556
Category	2	1.4832233	2.596153846	1.269230769	-0.1538461538	2.380631942	1.624858664	0.2115396446	1.985529557	1.144174959	0.3958333333
Category	3	1.2350464	2.5125	1.191666667	-3.883333333	1.253261564	2.105537104	0.3440950216	2.361359571	0.9759389671	0.4375
Category	4	0.9868695	2.428571429	1.392857143	-0.3571428571	0.6704193323	2.480613831	0.8189763592	2.296738072	1.066063348	0.4270833333

Table 5.1: Initial Results Table

5.7.1 Initial Results Evaluation:

At this point, the dataset and tool that yielded the best results was the Falling Dataset using Needle. By “best”, we are aiming for category 1 having the greatest distance from the values of the other categories. The ideal outcome would be to have a separate discernable threshold for each category. As seen from Table 5.1 and Figure 5.13, there is no significant variance in the scores per category overall over almost all datasets. The scores do not fall

uniformly and accurately enough into one of the 4 buckets. From this point on, I was advised to experiment with methods of optimization to tune the results into what we are looking for.

5.7.2 Neural Network Precision Results

	VIRAT (Water)				VIRAT (Emboss)				VIRAT (Biopython)			
Category	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Precision	0	0.3333	1	0	0	NA	NA	0.7143	NA	0	0	0.2857
Recall	0	0.6667	0.1111	0.7692	0	0	0	0.8333	0	0	0	0.667
Accuracy	0.47	0.7368	0.5789	0.5263	0.4615	0.7692	0.7692	0.7692	0.9375	0.75	0.5625	0.25
F1 Score	0	0.444	0.2	0	0	0	0	0.7692	0	0	0	0.4

Table 5.2: VIRAT ML Prediction Results

	Falling (Water)				Falling (Emboss)				Falling (Biopython)			
Category	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Precision	0	1	0.2	0.125	0.2667	0	0	0	0	0	0	0.25
Recall	0	0.25	0.25	0.25	1	0	0	0	0	0	0	1
Accuracy	0.625	0.8125	0.5625	0.375	0.3125	0.6875	0.75	0.75	0.75	0.75	0.75	0.25
F1 Score	0	0.4	0.2222	0.1667	0.4211	0	0	0	0	0	0	0.4

Table 5.3: Falling ML Prediction Results

	YKS (Water)				YKS (Emboss)				YKS (Biopython)			
Category	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Precision	0	0	0	0.1429	0.25	0	0	0	0	0	0	1
Recall	0	0	0	0.5	0.5	0	0	0	0	0	0	0.2
Accuracy	0.6	0.8	0.5	0.3	0.2	0.7	0.8	0.7	0.6	0.8	0.8	0.2
F1 Score	0	0	0	0.2222	0.3333	0	0	0	0	0	0	0.3333

Table 5.4: YKS Ego ML Prediction Results

Category	1	2	3	4
Neural Prediction	Y>1.731	1.483< Y< 1.7314	1.23< Y<1.483	Y<1.23

Table 5.5: ML Categorization Threshold

5.7.3 Neural Network Precision Evaluation

The ML model was tested on all datasets after using thresholds for each category. Since this would be for categorization, only the outer categories (1 and 4) would include out of bound regions. Table 5.5 shows how the normalized scores would be distributed among the 4 categories through strict thresholds.

The metrics calculated included precision, accuracy, recall, and F1 score. The ML model equation was tested on those metrics for each of the categories within each dataset. The majority of the metrics results were rather disappointing, however the more accurate readings weren't centered around any particular dataset or tool used. The majority of precision, recall, and F1 scores sat at 0%. Rather, some of the categories were found to be more accurate than others. The following will be a breakdown of the prediction results by category.

5.7.3.1 Category 1:

There was an impressive accuracy on the VIRAT/Biopython results, though not very meaningful with 0 true positives. The Falling and YKS datasets using Emboss yielded lower measurements greater than 0, with the Falling value for recall being 100%.

5.7.3.2 Category 2:

The outliers for the category 2 predictions included VIRAT and Falling datasets using the WATER tool. The Falling/Water results yielded an impressive precision of 100%, accuracy at 81.25%, and F1 score at .4. The VIRAT/Water results also had improved metrics in comparison to the rest of the dataset, but not nearly as substantial.

5.7.3.4 Category 3:

The most notable data metrics were the VIRAT/Water results, with a 100% precision and 57.89% accuracy. The Falling/Water results were better than the 0s received in other configurations, but nothing impressive with results ranging from 20–50%.

5.7.3.5 Category 4:

This category saw the most non-zero results out of all 4. However, the majority of the metrics fell in the range of 10-30%.

5.7.4 Highest Performing Metrics:

This section will discuss where the results with each of the greatest metrics were found in the format of *Dataset/Tool/Category. Precision:* The highest precision of 100% was found in VIRAT/Water/3, Falling/Water/2, YKS/Biopython/4. *Recall:* The highest recall was found in Falling/Emboss/1 and Falling/Biopython/4 at 100%. *Accuracy:* The highest accuracy was found in both the Falling and YKS datasets at category 2, ranging from 70-81%. *F1 Score:* The highest F1 score was found in VIRAT/Emboss/4 at 76%.

5.7.5 ML Metrics Discussion:

The results of the metrics indicate that there is some accuracy potential for the ML predictions, at least for categories 2-4. Unfortunately, the goal was to obtain the higher metrics from category 1 as it is classified as the highest in similarity and should therefore yield the highest normalized scores. Since this is the case, the focus will now shift into the optimization of

results in order to obtain higher scores from category 1, which would in turn improve all of the metrics described here.

5.8 Method Optimization Results

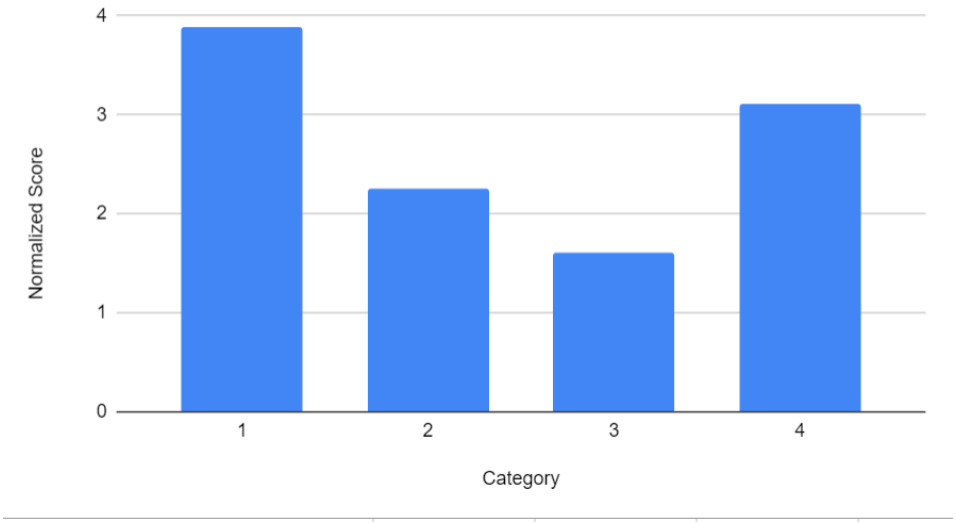

Figure 5.14: Squared Mapping Results Graph

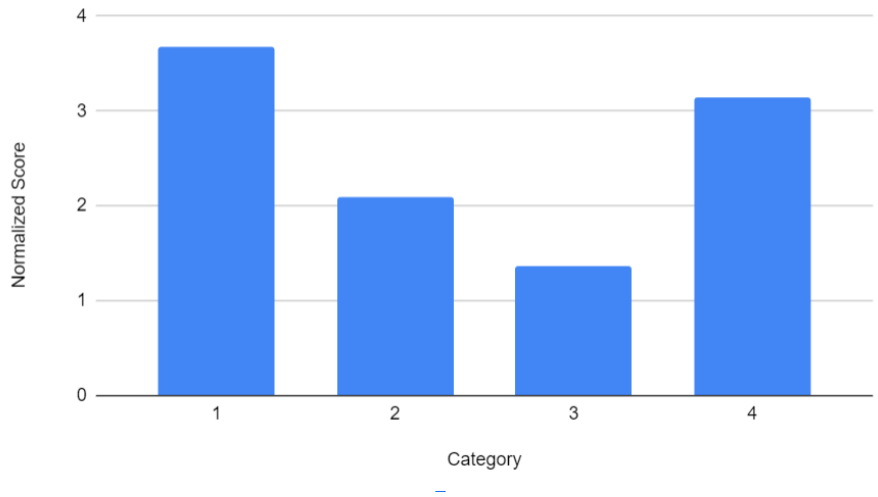

Figure 5.15: ByteDivision Results Graph

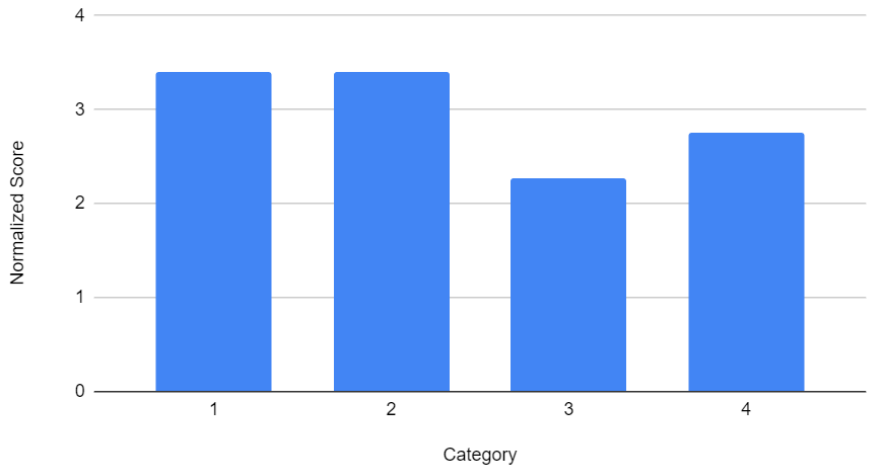

Figure 5.16: ReduceRepetition Results Graph

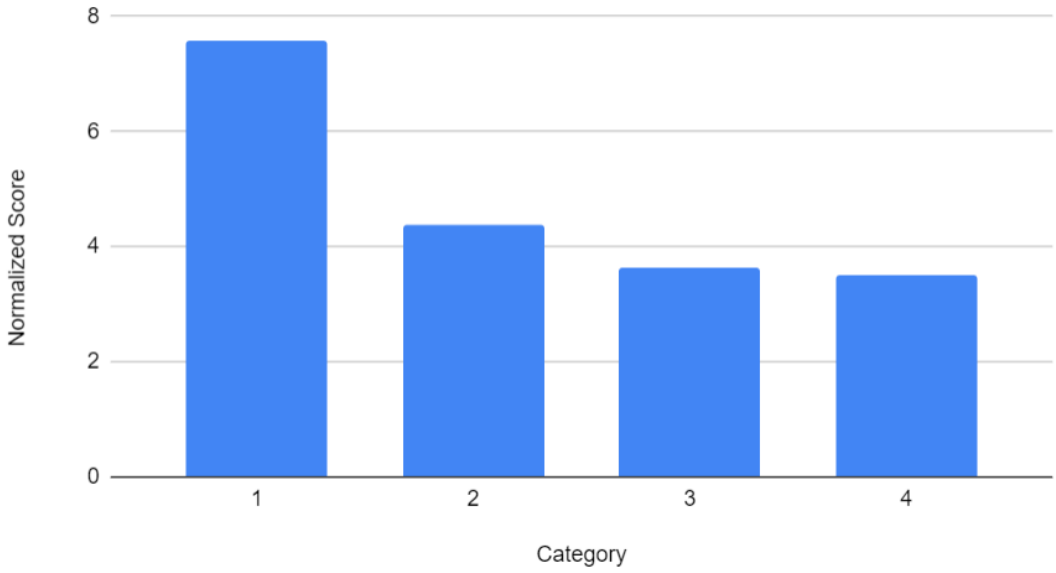

Figure 5.17: Sigmoid Mapping Results Graph

5.8.1 Mapping Optimization Evaluation:

Of the four methods introduced (shown in Figures 5.14-17), using the Sigmoid equation for mapping yielded the best outcome, meaning its results produced the greatest difference of Category 1 from the rest of the categories, allowing for a better chance of thresholding more

accurately. Regarding ByteDivision and the remaining optimization method results, this finally caused some variation in the DNA files to include the other characters significantly more. The results were simply the transposed version of that of the squared idea.

5.9 Penalty Optimization Results

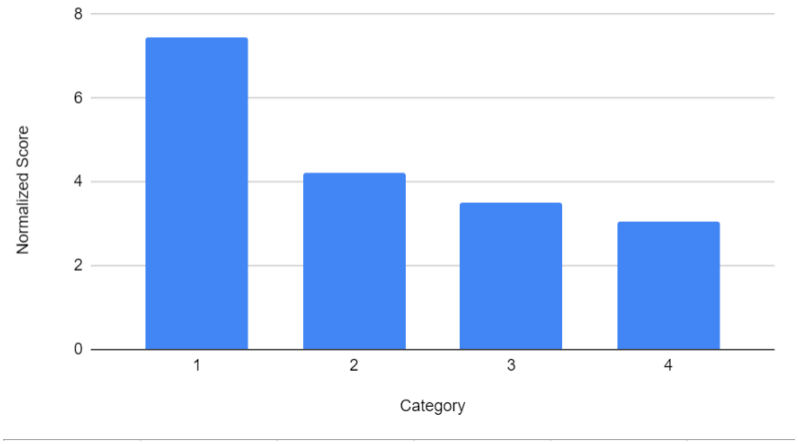

Figure 5.18: Linear Optimization Results Graph

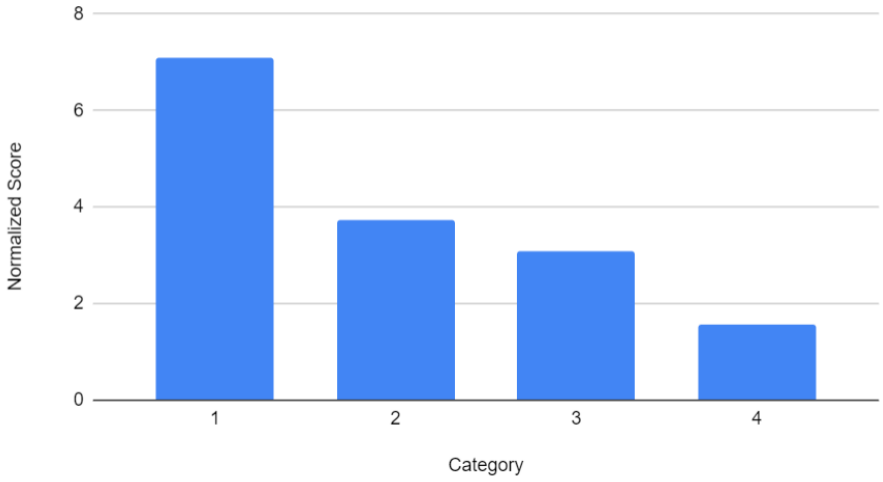

Figure 5.19: Quadratic Optimization Results Graph

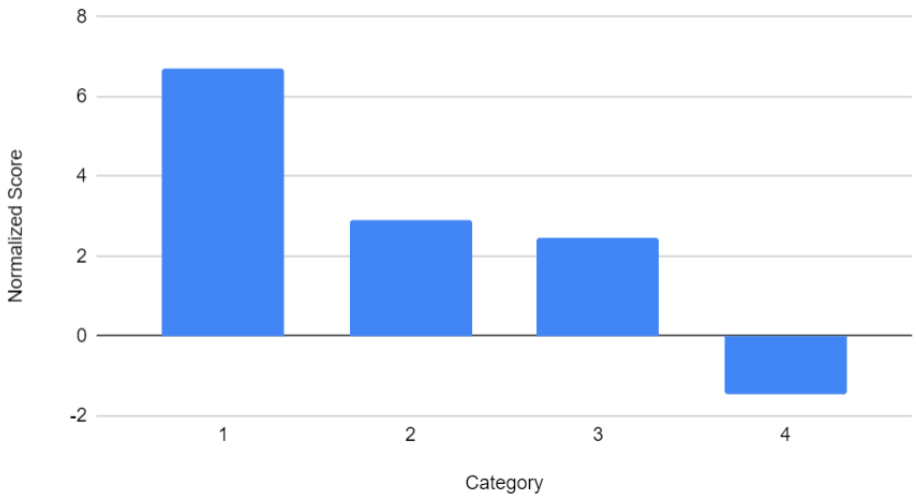

Figure 5.20: Cubic Optimization Results Graph

5.9.4 Penalty Optimization Results Evaluation:

In the results for Linear Optimization (Figure 5.18), there was a small decrease in the normalized scores but not significant enough to change the categorization. Overall, increasing the penalties up to cubic proportions (5.20) improved upon the Sigmoid results. It showed the most variance between 1 and the other categories, especially number 4. This shows that there is a possibility to threshold the categories separately.

5.10 Combined Optimization Precision

	Falling (Emboss)			
Category	**1**	**2**	**3**	**4**
Precision	0.4	Na	0	0.5
Recall	1	0	0	0.6667
Accuracy	0.6	0.73333	0.6667	0.8
F1 Score	0.5714	0	0	0.5714

Table 5.6: Optimization Precision

5.10.1 Optimization Precision Evaluation:

Using the previous metrics on the configuration of Falling/Emboss/1, there was improvement in almost all 4 values in the final optimized metrics shown in Table 5.6, barring the recall. The previous values consisted of: precision 26.67%, recall 100%, accuracy 31.25%, and F1 score of .4211. Not only was this an improvement over the specific combination of dataset/tool but also an improvement of all results.

5.11 Processing Performance

All testing and conversions were done on a single linux machine with the specs: (i7-4770k proc, 16GB of RAM, 500GB hybrid SSHD). In the first step of converting the videos to DNA, CreateArray, the average time for computation was 2.78 seconds per minute of video. The ssh process which executed the WATER and Biopython tools on the datasets would then take around 3.51 seconds for comparing 2 dna files regardless of DNA file size. This is because strings are much easier to process. For the online web interface Emboss Needle, it would depend more on server usage and internet speed so this tool would not be as consistently measured. On a larger scale and video, the average time for a 10gb video was 5 minutes and 45 seconds.

5.12 Noise Introduction

Noise Introduced	Average Similarity Detected
10%	92.425
15%	86.325
20%	80.1125
25%	79.35
30%	73.725

Table 5.7: Noise Introduction Values

A new approach was to introduce noise levels via randomness to find a difference threshold. The reason this is justified is because the algorithm will return a perfect similarity score (measured in percentage produced by comparison tool) if it's comparing the same video against itself. By introducing small amounts of uniformly distributed randomness, we can detect a general randomness threshold where the algorithm provides low scores for one video compared against a copy of itself with random noise. The reason I am choosing this approach is that it gives us a clear example of success (when the videos are the same) and helps us find a measurable cutoff of reliability.

For this experiment, 8 videos were tested against their own copies with incrementing levels of random values injected to their DNA strings (Table 5.7). The incrementation was by 5% and started at the 10% randomized level. The cutoff point I decided was when the similarity score reached below 75%, showing a no longer accurate reading. The randomization level was at 30% when this threshold was reached. The meaning of this experiment is that even a 10-30% difference from a video being compared to itself will show a false negative. This would also

prove the same for the inverse (False Positives) as we could take two completely different videos, and increment randomness into them until they test positive.

5.13 - Bucket Increase

For this experiment, the goal is to combine noise introduction with increasing the bins to prove even a slight difference will cause a false categorization. For example, if there are only 4 characters for a 5 letter word, I can only write 4^5 possible 5 letter words (with repeating characters). Likewise 10 characters can produce 10^5 5 letter words. The more characters you have to work with, the more possible permutations can be expressed in that alphabet. Therefore, the odds that you will find 2 identical permutations decreases with the more possible characters you have. This is the explanation for why the similarity percentage decreases as you increase the buckets.

Video A	Video B	Length	4 Buckets	10 Buckets	25 Buckets	50 Buckets	100 Buckets
Video 1 + 10% random	Video 1	1000	92.1	90.8	90.3	89.2	88.5
Video 1 + 25% random	Video 1	1000	80.4	77.2	75.5	74.4	73.2
Video 1 + 50% random	Video 1	1000	60.5	54.6	51.5	50.3	48.5
Video 1 + 100% random	Video 1	1000	26.05	12.58	6.02	3.5	2.15

Table 5.8: Bucket Increase

However, the categories from the buckets have their own problems. The lower the bucket count, the larger the range for each bucket. Therefore, lower bucket counts hide the variability and difference of the data.

CHAPTER 6 DISCUSSION

The goal of this research project was to explore methods of video similarity identification, and specifically through means of DNA sequencing. The application of accurate and robust video categorization would allow for a system that processes video in a privacy-preserving setting for a use case that has not yet seen mainstream representation.

6.1 Motivation

6.1.1 Measures of Success

The improved accuracy metrics had an additional increase in success particularly for categories 1 and 3: Videos that share time and space vs videos of the same activity. Unfortunately, the precision/recall/F1 score remained at 0 or N/A for the rest of the categories. It would have been ideal for there to have been an increase in overall accuracy for category 3.

6.1.2 Comparison with Prior Work

The overlying knowledge gained from this work is that there is potential for an increased processing time when compared to previous work. The previous F1 score of the SSO project went over 90% within 20 seconds [30]. Our current method was able to process a video of similar length (60 seconds) within 2-4 seconds, this showing an improvement on processing time. However the F1 score leaves much more to be desired, reaching .76 at most.

6.2 Limitations and Future Work

6.2.1 Hardware and Software Limitations

All of the datasets needed to be downloaded and hosted on a single machine in order to be processed into DNA files using the CreateArray Algorithm. These large datasets spanned

multiple terabytes and could not be hosted locally, so the linux machines at the University of Washington Bothell were relied upon for DNA processing. Towards the end of this study, the machines were not as available to complete testing beyond the Falling/Emboss dataset.

6.2.2 Principal Component Analysis

The fact of converting entire video files to a single string of dna is that there is a lossy conversion when reducing a dataset by multiple dimensions. In a video, there are numerous identifying features and improving this research would involve centralizing the data only to those features. Some of these features include more picture-oriented characteristics such as overall contrast, sharpness, filters, etc. Other more apparent features include video duration, resolution, frame rate, color type, aspect ratio, and more. Future research would aim to enumerate such features from the videos, and only use those to convert to the lower dimension rather than to use bytes as in this study. Another factor to consider is that if powerful ML algorithms are not able to pick the correct CAPTCHA from 9 images for example, there is a high probability that image frames in video would not be correctly categorized by a manual DNA sequencing method. One question to consider in this scenario is where is the cutoff for the type of activity? Activity can be ambiguous. If you were able to deduce that the category is NOT 3, you would probably be able to rule out category 4.

6.2.2 Bucket Problem

Another issue is that the DNA Sequencing algorithms cannot be run on anything other than 4 buckets (characters) according to the problem. If you had one set of identical byte arrays and you swapped in random bytes into various existing bytes of one of the arrays, there would be a threshold where the algorithm no longer sees them as identical. The inverse would be true as well. If you had completely different byte arrays and you incrementally converged one of them into the other by randomly copying bytes into the same index, you would eventually have the

same video. By randomizing 100% of the data of a copy of the original video, it would yield a random video, it is therefore the same as starting with a random video and converging it back to the original video. This proves that both of these cases are inverses of each other.

In this case, there should therefore be a common threshold defined as:

Randomness Threshold = T

Convergence Threshold = (100 - T)

If we can show that the algorithm fails to identify an identical video (which is the most charitable case) over a value *T*, then it is not reliable for the general case where videos would be different even if they were of ostensibly the same event. It would have to be shown that the videos cannot have over *T* between them. This is because the problem exists within identical videos as well. The burden of proof lies with the person who asserts the validity of the results because they have to show that the video has not been contaminated with a certain threshold of obfuscating data/noise. A further research project would be to determine what counts as obfuscation in this scenario. This approach proves that this overlying experiment is not investigating the right hypothesis at this time, and that the previous suggestion would be a more effective experiment to further assess the viability of this approach.

The algorithm doesn't prefer specific characters, but rather the longest matching strings. Therefore, any character that bifurcates a formally matching substring will decrease the score equivalently to any other character. It is thus not important which character is selected for the randomization. It only matters that it is not the same character that it was previously. Due to this observation, a uniform random distribution is sufficient for the purpose of finding a threshold.

CHAPTER 7 CONCLUSION

This section will summarize all of the key findings yielded from this body of work. It started with self-done videos that were created as a control dataset to reach an initial estimate of a normalized score, the resulting average score was 1.92. This dataset was then used to conduct multiple tests in order to find the most ideal gap and mismatch scores. It was found that there was a linear increase of 54.05 for every incremental increase in gap score on videos under category 1, and no effect when increasing the mismatch score. Going forward, the gap score used would be 1 and the mismatch score would be left at default. A portion of the datasets were then processed and used to train a machine learning algorithm to be able to categorize 2 videos into one of the 4 categories based on the normalized score. After the ML Model was trained, it was used on all datasets, averaging 10-14 tests per set, per sequencing tool. There were a number of cases where the ML predictions would mark an entire dataset as false and/or produce no true positives, leading to a greater accuracy score along with precision, F1, and recall scores at a value of little to 0. Since the results of the initial tests were far from ideal, there were multiple methods of optimization applied on the results in order to separate category 1 from others. After implementing the highest performing methods of optimization (cubic and sigmoid functions), there was an increase in all accuracy metrics from the same dataset and tool used prior. In conclusion, the compression of video to DNA characters limited to 4 buckets proves too much loss to be used as an effective video categorization mechanism.

BIBLIOGRAPHY

[1] W. R. Taylor and R. E. J. Munro, “Multiple sequence threading: conditional gap placement,” *Folding and Design*, vol. 2, 1997.

[2] M. H. Coen and K. W. Wilson, “Learning Spatial Event Models from Multiple-Camera Perspectives,” *Managing Interactions in Smart Environments*, pp. 215–226, 2000. 2

[3] M. Duggan.. Photo and video sharing grow online. Pew research internet project. (2013)

[4] Batson,. "Spectral sparsification." PhD diss., 2014.

[5] L. Nathan Perkins,. *Convolutional neural networks as feature generators for near-duplicate video detection*. Boston, MA: Boston University, 2015.

[6] B. Lagesse, G. Nguyen, U. Goswami and K. Wu, "You Had to Be There: Private Video Sharing for Mobile Phones using Fully Homomorphic Encryption," 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2021, pp. 730-735, doi: 10.1109/PerComWorkshops51409.2021.9431029.

[7] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier and L. E. Barnes, "iCrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing," in *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010-2022, 1 Aug. 2016, doi: 10.1109/TMC.2015.2483505.

[8] S. Oh *et al*., "A large-scale benchmark dataset for event recognition in surveillance video," *CVPR 2011*, 2011, pp. 3153-3160, doi: 10.1109/CVPR.2011.5995586.

[9] G. Kordopatis-Zilos, et al. "ViSiL: Fine-grained spatio-temporal video similarity learning." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

[10] S. F. Altschul,., et al. "Basic local alignment search tool." *Journal of molecular biology* 215.3 (1990): 403-410.

[11] N M.Luscombe, , D Greenbaum, and M. Gerstein. "What is bioinformatics? An introduction and overview." *Yearbook of medical informatics* 10.01 (2001): 83-100.

[12] B. Langmead, et al. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome." *Genome biology* 10.3 (2009): 1-10.

[13] Y Jararweh., M Al-Ayyoub, ,M Fakirah, *et al.* Improving the performance of the needleman-wunsch algorithm using parallelization and vectorization techniques. *Multimed Tools Appl* 78, 3961–3977 (2019). https://doi.org/10.1007/s11042-017-5092-0

[14] A Arampatzis, J. Kamps. "A signal-to-noise approach to score normalization." *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009.

[15] M. Vingron. "Near-optimal sequence alignment." *Current Opinion in structural biology* 6.3 (1996): 346-352.

[16] Laura Stein "Policy and participation on social media: The cases of YouTube, Facebook, and Wikipedia." *Communication, Culture & Critique* 6.3 (2013): 353-371.

[17] R. Yonetani, K. M. Kitani, and Y. Sato, “Recognizing micro-actions and reactions from paired egocentric videos,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18]T Chakrabarti and D Sinha. "DNA Sequence Alignment by Parallel Dynamic Programming." (2011).

[19]T F Smith., and M S. Waterman. "Identification of common molecular subsequences." *Journal of molecular biology* 147.1 (1981): 195-197.

[20] S McGinnis. And TL Madden., 2004. BLAST: at the core of a powerful and diverse set of sequence analysis tools. *Nucleic acids research*, *32*(suppl_2), pp.W20-W25.

[21] " E. Auvinet, C. Rougier, J.Meunier, A. St-Arnaud, J. Rousseau, "Multiple cameras fall dataset", Technical report 1350, DIRO - Université de Montréal, July 2010."

[22] D Kucharavy,. and De Guio, R., 2011. Application of S-shaped curves. *Procedia Engineering*, *9*, pp.559-572.

[23] D Danopoulos,., Kachris, C., Soudris, D. (2019). Approximate Similarity Search with FAISS Framework Using FPGAs on the Cloud. In: Pnevmatikatos, D., Pelcat, M., Jung, M. (eds) Embedded Computer Systems: Architectures, Modeling, and Simulation. SAMOS 2019. Lecture Notes in Computer Science(), vol 11733. Springer, Cham.

[24] Abukari, Arnold Mashud, Edem Kwedzo Bankas, and Mohammed Muniru Iddrisu. "A secured video conferencing system architecture using a hybrid of two homomorphic encryption schemes: A case of zoom." *International Journal of Engineering Research & Technology (IJERT)* 9.8 (2020): 237-240.

[25] N Geetha,., and K. Mahesh. "An Efficient Enhanced Full Homomorphic Encryption for Securing Video in Cloud Environment." *Wireless Personal Communications* (2022): 1-19.

[26] D Wang,. *Action Recognition in Multi-view Videos*. Diss. 2018.

[27] M. Johnson, Zaretskaya, I., Raytselis, Y., Merezhuk, Y., McGinnis, S., & Madden, T. L. (2008). NCBI BLAST: a better web interface. *Nucleic acids research*, *36*(suppl_2), W5-W9.

[28] Needle, E. M. B. O. S. S. "Available online: https://www. ebi. ac. uk." *Tools/psa/emboss_needle/(accessed on 2 September 2021)*.

[29] P Cock,. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., ... & De Hoon, M. J. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422-1423.

[30] K Wu, and B. Lagesse. "Detecting hidden webcams with delay-tolerant similarity of simultaneous observation." *Pervasive and Mobile Computing* 65 (2020): 101154.

[31] S. Patro, . G. O. P. A. L., and Kishore Kumar Sahu. "Normalization: A preprocessing stage." *arXiv preprint arXiv:1503.06462* (2015).