GAMER SKILL PREDICTION USING GAZE TRACKING

JONATHAN PERRY

UNDERGRADUATE HONOURS THESIS

ONTARIO TECH UNIVERSITY — FACULTY OF SCIENCE

COMPUTER SCIENCE

April 2020

SUPERVISORS:

CHRISTOPHER COLLINS

FAISAL QURESHI

ABSTRACT

Classifying skill level of an individual for different tasks has been accomplished to different levels of accuracy. This paper explores different machine learning techniques in an attempt to classify the skill level of an individual in a highly complex task. Eye tracking of individuals was used to collect eye gaze data during the task of playing a game of Dota 2. A classifier was not successfully produced but there is reason to believe that it is possible.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Evaluation of an individual's proficiency in a complex task can take a significant amount of time and data accumulation before being able to provide a skill determination for that individual. Many popular online multiplayer team games (e.g. Overwatch, League of Legends, and Dota 2) require a set amount of games played to calibrate an individual player's skill level. Based on skill level, the player can then be placed among players of approximately the same skill level to ensure that games are balanced to prevent one team from overrunning the other. However, accurately determining proficient individuals who are on brand new, uncalibrated accounts, is still an ongoing problem [11].

1.2 PROBLEM STATEMENT

Is it possible to use eye gaze data of an individual to determine their skill level in a complex task? If possible, skill verification could be done through eye gaze alone, detection of expert players would take less time, and eye gaze could be used to detect novice players and focus their gaze. A novel approach to attempt and quickly determine a player's Dota 2 proficiency, based on eye gaze data alone, will be examined and expanded upon. The ultimate goal is to create a model that can predict a player's Dota 2 skill level from eye gaze data. Dota 2 was the game chosen to study this problem because of its overwhelming complexity.

1.2.1 The Complexity of Dota 2

As mentioned previously, Dota 2 is an online multiplayer game. It is a game between two teams with five individual players per team. Dota 2 requires raw mechanical skill, extensive game knowledge, and team coordination. A typical game of Dota 2 takes between 20 and 40 minutes to play on average and is an extremely complex task with no two games played being identical. A player has approximately 1000 possible actions that can be made at every tick of gameplay compared to roughly 35 for chess [9]. There are numerous reasons why Dota 2 is

complex, but some simple calculations can be done based on central components of the game to give a vague intuition into the complexity of it.

Each player in each game picks and plays one of 119 playable heroes and no hero can be represented twice in a game i.e. there are 10 unique heroes played in a game. Most heroes in the game have 4 abilities, some have 5, and one hero has 10. Each ability does something different and unique. Depending on which abilities are used, the interaction between abilities is either entirely existent, partially existent, or non-existent. Skilled players will pick their hero based on its abilities and how they complement their teammates' heroes' abilities, negate enemy heroes' abilities, or both.

Assuming just the lower limit of 4 abilities per hero, 119 heroes provide a total of 476 distinct abilities. The number of couplings of different abilities — using 2 distinct abilities with each other — can be calculated using the binomial coefficient formula $\frac{n!}{k!(n-k)!}$ where n is the number of distinct abilities (476) and k is the number of different abilities being used (2). The number of combinations is 113,050. However, the order in which abilities are used in game matters so the permutation of two different abilities can be calculated with the formula $\frac{n!}{(n-k)!}$ which calculates to 226,100 permutations. However, within each game, assuming only 4 abilities per hero, there are only 40 abilities that can be used. With only 40 abilities available, increasing the number of abilities used to a nominal 3, there are 59,280 permutations possible.

In addition to abilities, there are also items that players can purchase with in-game gold earned. There are currently 159 items available every game once enough gold is earned. Similar to abilities, the interaction between these items is entirely existent, partially existent, or non-existent. Not only can items interact with other items, but they can also interact with hero abilities. Hero abilities can also interact with items. The permutations possible between all abilities and all items is astronomical.

Dota 2 is not limited to the interactions between abilities and items, but the numbers in a static context provide insight into the level of complexity. Complexity also comes from interacting with other players in the game. Dota 2 is a fluid game where each player consistently makes decisions based on the current state of game being played to determine what their next move is, their team's next move, and their opponent's next move. These decisions are dependent on many things, but centrally: the heroes on a player's team, what heroes are on the

Figure 1: Dota 2 Highlighted User Interface Elements

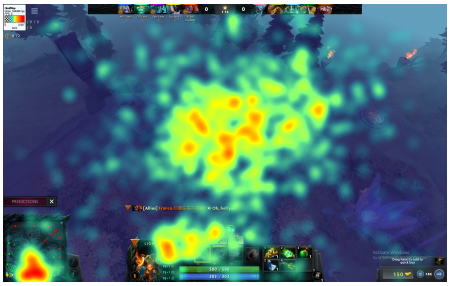
enemy team, and what items each player has. Generally, the first 500 to 1000 hours of gameplay people are just beginning to have a solid grasp of the foundations and touching on the intricacies of the game.

1.2.2 Why it Could Work - A Hypothesis

Figure 1 shows a screenshot of gameplay as well as highlights, in red boxes, elements of the game's user interface. The elements highlighted are always visible and they are always in the same spot – they are static. However, what information is contained within them changes depending on the state of the game as well as decisions made by the player. An abundance of information can be gathered from these static elements about the current state of a game. From gaze alone, an experienced player could give a general overview of the current state of a game from a simple screenshot of an ongoing game. While playing the game, players can use these static elements to determine their next move.

For example, the in-game clock consistently ticks every second and keeps track how much time has elapsed so far. The clock can be used to help a player keep track of time related events in game such as: stacking jungle camps every minute, power runes spawning every two minutes (starting four minutes into the game), bounty runes spawning every five minutes (starting at zero minutes in), the time range which Roshan can spawn again, and protecting or taking outposts every ten minutes. It is likely that the previous sentence listed jargon that you cannot make sense of. This exemplifies the information provided by in-game static user interface elements. A novice

(a) Low Skill - Level 1



(b) Average Skill - Level 4

(c) High Skill - Level 7

Figure 2: Relative Gaze Heatmaps of Different Skill Levels for a Dota 2 Game Played

player playing for the first would not know to use the clock to track these in-game events. However, a more experienced player uses the clock to be in the right place at the right time for these events.

1.2.3 Preliminary Heatmaps

Using Tobii Pro Studio's heatmap creation tool, three different heatmaps were created from an initial pilot study explained in Section 2.1. One from an individual with low skill level, one from an individual with average skill level, and one from an individual with high skill level. These heatmaps can be viewed in Figure 2. A distinct difference can be seen between a low skill player compared to both average and high skill. There are noticeable differences between the average skilled player and the highly skilled player — the level 4 player has more focus on their abilities than the level 7 player. However, as hypothesized in Section 1.2.2, a novice player does not make use of the static user interface elements as frequently as as those with experience.

1.2.4 Secondary Considerations

A secondary consideration of the research composed contemplates if there could be a point where eye data is no longer sufficient at accurately predicting individual skill level (i.e. when it can no longer differentiate between two skill levels). Is there a difference detectable

between a top 0.1% player and a top 0.001% player? The data collected for this study did not contain two individuals of such a high caliber, but one individual ranked within the top 500 players in North America participated. Another secondary consideration is whether individuals exhibit a pattern of gaze data which could be used to uniquely identify them.

1.3 RELATED WORK

Many different studies have been conducted which focus on eye tracking and the gaze data produced. The following studies showcase the ability to classify individuals performing a task from eye gaze data, the differences in eye gaze between skill levels, and the effectiveness of training novices with eye gaze data from proficient individuals.

1.3.1 Finding Waldo

Although the use of eye gaze data was not used by Brown et al. [2], they tracked a user's mouse movements during the activity of finding Waldo — a reasonable proxy for eye gaze in this task. This activity relies on a user's gaze to search through an image to find an individual hidden within it. They were able to predict, between 62% and 83% accuracy, whether an individual would be fast or slow completing the task.

1.3.2 Who is the expert?

Liu et al. [4] were successfully able to predict, with up to 96% accuracy, the expertise level of individuals during a collaborative session creating concept maps. They were able to achieve this as early as as the first minute or rather the first 5% of observation time.

1.3.3 Gaze Tracking within Sports

A study looking at the differences between near-expert and expert baseball umpires was conducted by Millslagle et al. [5]. Their study found that experts found where the ball would be released earlier compared to near-experts. They also determined that experts can track the ball longer in the air compared to near-experts.

Raab and Johnson [8] looked at the difference in search and option generation strategies between non-expert, near-expert, and expert handball players. It was determined that the average number of options created between different levels of expertise was similar, but the quality of options generated and the final option chosen were different.

1.3.4 Training Individuals on Expert Gaze Data

Wilson et al. [12] conducted a study to determine if gaze training enhances laparoscopic skill acquisition and multi-tasking performance. Participants were randomly placed into three different groups for training: trained with footage of the eye gaze as well as movement of an expert, footage showing the movement of an expert, and shown nothing. The gaze-trained group outperformed both groups by completing the the test task faster. They spent more time gazing at the target of the task rather than the tools used to complete the task.

1.3.5 Summary

Eye gaze data is a useful metric which has a variety of different uses. The studies above (barring the Waldo study) used it as a central measurement for their research conducted. The studies show the potential of what could be accomplished if a classifier can correctly predict the skill level of an individual completing a complex task.

2 METHODOLOGICAL APPROACHES AND RESULTS

Creating a classifier to predict a player's Dota 2 skill level from eye gaze data was an extensive process. Two of the major steps involved were the collection of data and the creation of a classifier. Many different classifiers were created and will explored in the following sections. Classifiers were iteratively improved upon to increase performance and ultimately test their generalizability.

2.1 PILOT STUDY DATA COLLECTION

The software used to record the eye gaze data collected was Tobii Pro Studio which used a Tobii X60 Pro Eye Tracker that records at 60 Hz. Eye gaze data from seven different individuals of different skill levels was captured. Although the eye tracking hardware was the same for all participants, individuals could choose to use their own mouse and/or keyboard if desired. Personal mouses were preferred by experienced players whereas individuals with no experience simply used the computer mouse provided. The setup used for the controlled study data collected can be seen in Figure 3 and is near identical to what was used in the pilot study except for the tape used to secure the eye tracker in place. The screen resolution used to play Dota 2 on was 1920×1200 which has an aspect ratio of $16 : 10$. The game took up the entirety of the screen when being played.
Each participant would calibrate the eye tracker using the recording software and then play Dota 2 as their eye gaze was collected for the duration of a game. The length of Dota 2 games is variable so the amount of data collected per participant varied depending how long their game lasted. The eye gaze recording begun at the beginning of the game played and was stopped when a team won the game or the participant ended the session.

In total, 11 games of eye gaze data were collected from 7 participants. The games played ranged in length from approximately 18–67 minutes with an average game length of roughly 39 minutes and a standard deviation of approximately 14 minutes. The game lengths and skill level of the individual who played those games can be seen in Table 1. Each row indicates a different player.

Figure 3: Eye Tracker Setup

Table 1: Game Data Collected

Skill Level	Game Length(s) in Minutes
1	37.1
1	24.1
2	50
3	48.1
4	45.2, 67.4, 46.6
4	23.9
7	32.3, 37.7, 17.7

2.1.1 Setting up the Data

The software used captures many different metrics including each individual eye pupil's estimated size and the distance between the eye and the tracker. For each game recording, a tab separated file is generated with 82 columns containing a variable number of lines of data depending on the game's length. Each line of data represents an individual gaze recording. One second of gaze recording would produce 60 lines of data.

For the purposes of this study, the data collected was reduced to three metrics: *RecordingTimestamp*, *GazePointX*, and *GazePointY*. These three metrics, as referred to by their names from the recording software's manual, were the only data points used throughout the study. *RecordingTimestamp* represents the number of milliseconds that have occurred since the beginning of the recording ($t_0 = 0ms$). *GazePointX*

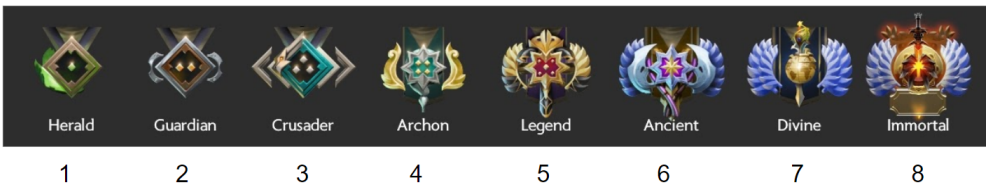


Figure 4: Skill Tiers Defined By Valve

represents the horizontal coordinate of the averaged left and right eye gaze point on the screen. Similarly, *GazePointY* represents the vertical coordinate of the averaged left and right eye gaze point on the screen. After reduction, any data that was deemed null — it did not contain *GazePointX* and/or *GazePointY* — was not included.

Each gaze data point was labeled with a number dependent on the proficiency level of the participant that produced it. This proficiency label was based on the the existing skill levels within the game currently. Valve, the developer of Dota 2, have 8 different tiers of ranks which can be seen in Figure 4. They are assigned to players based on skill level derived from in-game performance. Each tier currently contains 5 divisions, but for the purposes of this study only the tier of skill was considered when labelling individuals. The lowest level of skill is "Herald" (referred to as level 1) and the highest is "Immortal" (referred to as level 8). The number below each tier in Figure 4 represents the label used to distinguish between gaze points collected from individuals.

The pilot study data consists of the eye gaze data collected from the games in Table 1. The level 1 participants had never played before and were given a level 1 skill level and an overview of the game as well as time to learn the basic controls before playing a game. Roughly two minutes was spent giving an overview of the game and about 5 minutes to learn the basics of controlling their hero. All participants played a real match with human players except for the two level 1 players who played with computer controlled players. Players were allowed to choose whether they wanted to play a real game or a game with computers.

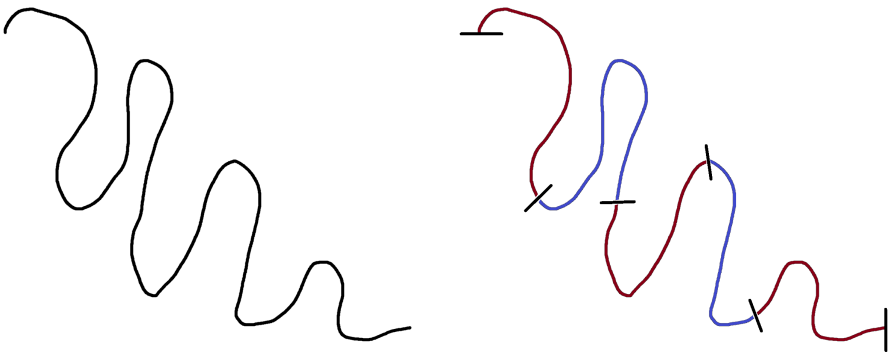
2.2 A SUPPORT VECTOR MACHINE CLASSIFIER

A popular machine learning library provided by scikit-learn [10] was used to instantiate a support vector machine (SVM) — a supervised machine learning model. The model can be trained and tested to classify data. Unless specifically stated, the model was instanti-

ated with the default values which entail: a penalty parameter of $C = 1.0$, $kernel = rbf$, and $gamma = auto$. The training and test data was randomly split using scikit-learn's $train\_test\_split$ method which has a default training size of 75%, a test size of 25%, and used the $ShuffleSplit$ method for randomizing the order of the samples. Many different SVMs diverging from the default values were used to create Dota 2 skill classifiers that predicted based on eye gaze data inputted within a feature vector.

What is a feature vector?

Before continuing, it is important to define what is meant by a feature vector. A feature vector represents the structure of the input data into the model. Each sample of data uses the same format when training or testing an individual model. A feature vector can be affected by factors of how the data was obtained, created, or altered. The contents of a feature vector are reflected in the output results meaning that different feature vectors created from the exact same data can produce different results. Which features are used is a choice by the designer and can be altered. An input takes the general form of: $[feature_0, ...feature_n] \Leftarrow label$ where the features are derived or taken from data points and the label specifies the skill level of the player that the data was taken from. For example, one data input using the feature vector $[GazePointX, GazePointY]$ from data collected from an "Immortal" player could take the form $[843, 128] \Leftarrow 8$.



(a) Gaze Trail **(b)** Gaze Trail Split By Period

Figure 5: Splitting Up Eye Gaze Data Based On Period

One important alteration of the data amongst all the feature vectors used is referred to as a *period*. A period refers to a set amount of time which the eye gaze data is split up into during pre-processing i.e. how many seconds of gaze data will be within one sample. If there

are 60 seconds of data, and a *period* of 5 seconds is used, there will be 12 discrete, non-overlapping samples created of which each sample contains 5 seconds of continuous gaze data. Figure 5a provides a simple illustration of a hypothetical gaze trail of a user moving their eyes from the top left of the screen to the bottom right in a winding path. Figure 5b has split the gaze trail based on an arbitrary period where a colour change indicates the end of one *period* and beginning of the next.

The following sections will expand on the different SVM models instantiated, the different feature vector design choices made, the settings used when processing the data into feature vectors, and the results produced. The sections are presented in the order the research was conducted. Only level 1, 4, and 7 data was collected for the trials mentioned in Sections 2.2.1 and 2.2.2.

Results are presented in confusion matrix (CM) format. Each row in a CM represents a different true skill level and each column shows how that particular skill level was predicted. For example, in Figure 6 the first row represents how level 1 player samples were predicted. Each column in the first row represents how a level 1 player was predicted. For example, 41% of the level 1 samples tested were classified as level 1, 0% of the samples were classified as level 4, and 59% of the samples were classified as level 7. A perfect classifier's CM would have a diagonal solid in colour indicating that each skill level was predicted correctly for every sample tested.

An important caveat to note is that when the data was split into sample periods for SVM trials, data that was marked as being off screen was not retained. If either or both of the *GazePointX* or *GazePointY* were not recorded as being on the screen, they were not included in a sample. This means that two different 5 second samples may have a different number of final gazes included, depending on the number of gazes recorded that were "off screen".

2.2.1 Gaze Distance Travelled per Period

The earliest classifier created used a simple feature vector consisting of the gaze distance travelled during a 5 second *period*. The gaze length was calculated for a period by using $\sum_{n=1}^{numSamples} d(gaze_n, gaze_{n-1})$ where d is the Euclidean distance between two points. The period size was arbitrarily chosen to be 5 seconds as this did not seem too long nor too short and captured a length of gameplay which a lot of abilities could be used within. One trial's confusion matrix result using

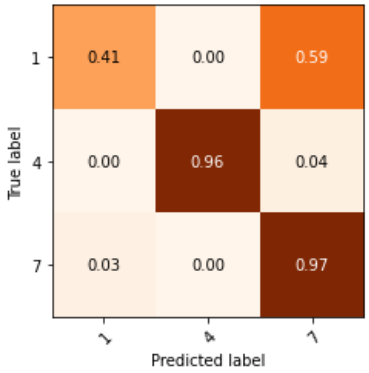


Figure 6: Gaze Distance Travelled in 5 Seconds

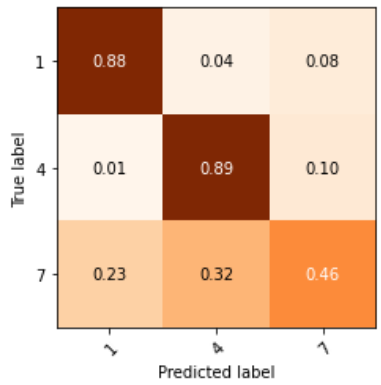


Figure 7: Average Change in Gaze Per 5 Seconds

this feature vector can be seen Figure 6. The result shows an incomplete model that cannot predict level 1 players accurately but is able to predict level 4 and level 7 players accurately.

2.2.2 Average Delta Change per Period

Another early feature vector designed used the delta change between each successive gaze within a 5 second period, taking the form of $[\sum_{1}^{n}(x_n - x_{n-1}, y_n - y_{n-1}]$ where n is the number of gazes in the sample. This captures whether the user looked left, right, up, down or fixated on the previous gaze point. One trial's resulting confusion matrix can be seen in Figure 7. The results are similar to the previous feature vector suggesting an incomplete model that cannot predict a level 7 player accurately but can predict level 1 and level 4 players accurately.

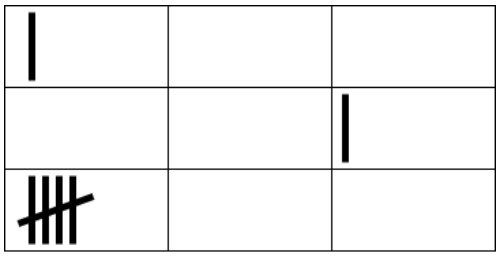


Figure 8: Gaze Bin Representation

2.2.3 Gaze Bins

There were small numbers of trials for the feature vectors mentioned in Sections 2.2.1 and 2.2.2. From these two feature vectors, a more extensive feature vector was conceptualized and then implemented. The main problem with the first two feature vectors used is that the temporal nature of the data was completely lost. The final representation, once processed, was simply a number or two which did not preserve any information about the sample of gazes that produces it.

Gaze bins organizes gazes into bins based on where each gaze took place on the screen. A representation of an example gaze bin can be seen in Figure 8. In Figure 8, the screen was split up into 3 rows and 3 columns of bins. It can be interpreted as: one gaze took place in the top left corner of the screen, one gaze took place in the right centre of the screen, and 5 gazes took place in the bottom left corner of the screen.

Using gaze bins partially maintains the temporal nature of the data. Within a sample, it is possible to know approximately where other gazes took place during that period. The different settings to produce a gaze bin feature vector were: *bin_rows*, *bin_columns*, *period*, *overlap*, and *kernel*. Adjusting these settings changed how the data was processed into distinct *gaze_bin* feature vectors. The feature vector produced, dependent on these settings, was then used to create samples to train and test on.

Explaining the settings

Both *bin_rows* and *bin_columns* were determined at the same time and were based upon the aspect ratio of the screen that was used during data collection. The screen uses an aspect ratio of of $16:10$ so natural values to use would be 10 *bin_rows* and 16 *bin_columns* i.e. a 10×16 *bin_size*. Based on these number, the screen is split into 10 equaled sized rows based on the vertical resolution and 16 equal sized columns based on the horizontal resolution. This creates a grid

of 160 equal sized bins. Other bin setups tried were 5×8 and 20×32 – half and double the aspect ratio. The grid of bin tallies is the input into the SVM classifier to train or test upon. The *gaze_bin* representation in Figure 8 used a *bin_size* of 3×3 and would be inputted as $[1, 0, 0, 0, 0, 1, 5, 0, 0]$.

The definition of a *period* can be found above in Section 2.2. *Overlap* refers to a pre-processing step when splitting up the data into periods. With no overlap, the data samples are completely discrete. With overlap, it is possible to have samples that contain the exact same data that is contained in a different sample. Splitting up 30 seconds of total gaze data into samples based on a 10 second period and no overlap would roughly be represented by $[gb_{t_0-t_{10}}, gb_{t_{10}-t_{20}}, gb_{t_{20}-t_{30}}]$ whereas a 5 second overlap used is $[gb_{t_0-t_{10}}, gb_{t_5-t_{15}}, gb_{t_{10}-t_{20}}, gb_{t_{15}-t_{25}}, gb_{t_{20}-t_{30}}]$ where $gb_{t_x-t_y}$ is a gaze bin sample processed starting at x seconds (inclusive) and ending at y seconds (non-inclusive).

One benefit of overlapping is that the total number of samples produced can essentially be doubled if an overlapping time of half the period length is used. Additionally, arbitrary discretization of the data samples can be partially mitigated because the data contained within one sample can have data contained within another sample. This could potentially capture relationships between a number of gazes that happened to be split into different samples.

Limiting the samples used

The number of samples available is dependent on the *period* length and the *overlap* length. To compare the results of different models trained and tested on the settings mentioned, it was necessary to ensure that each model was trained and tested on the same number of data samples. The lower limit of samples produced is determined by the highest period length used. The period lengths used, in seconds, were: 5, 20, 30, 60, 90, 105, and 120. The period lengths chosen were based upon the initial 5 second period length used previously and extended to examine the effect a longer duration had on results. Either half of the *period* (rounded up) was used for the *overlap* length or no overlapping was used. Thus, 140 training samples and 45 testing samples could be produced with no overlapping. With overlapping, 280 training samples and 90 test samples could be produced.

Running a trial

For each recorded game, the data was split into samples based upon both *period* and *overlapping* length. Next, each sample was pro-

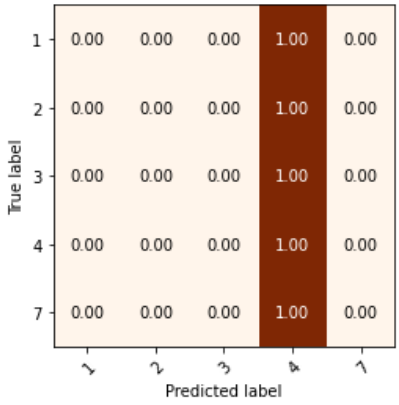


Figure 9: Trials Using an RBF Kernel

cessed into its gaze bin representation dependent upon *bin_rows* and *bin_columns*. After, the sample was placed into a central collection that includes all samples. The central collection was randomly split up into a training set and test set after all samples had been processed into gaze bins. This means that each trial has data used to train and test upon that is different than other trials. The source of the data is the exact same for every trial, but the data selected to be used is different for each trial.

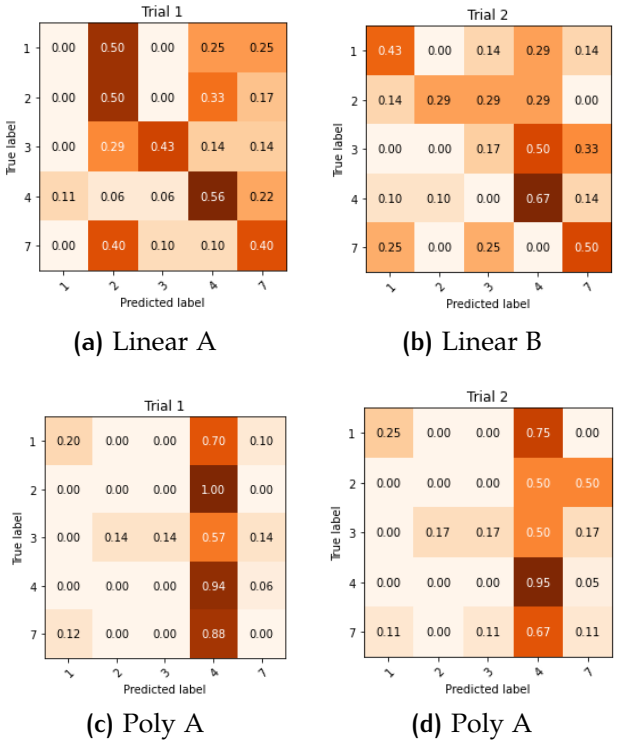
The sets were then reduced in size to only include the first n samples which is dependent on the use of overlapping. An SVM classifier was then instantiated with a specific *kernel*: linear, polynomial, or rbf. The classifier was then trained on the training set. Finally, the classifier was tested on the test set to evaluate its suitability.

The results

Many different trials were conducted based upon different combinations of the settings mentioned. At least four trials were conducted for each combination to ensure that any results achieved were consistently possible. Based upon the settings listed so far, there are:
$n\_bin\_sizes \times n\_periods \times overlapping\_used \times num\_kernels =$
$3 \times 7 \times 2 \times 3 = 126$ trial combinations. Four trials per combination gives 504 trials conducted.

All of the results for each combination will not be examined in detail as it would be lengthy and unnecessary. Instead, an overview of the results will be presented covering the general trend of the outcomes achieved by the differing classifiers.

Figure 9 shows the CM for every single combination produced using an *rbf* kernel, regardless of any of the other settings. The CM did not change and in every trial every sample was classified as having



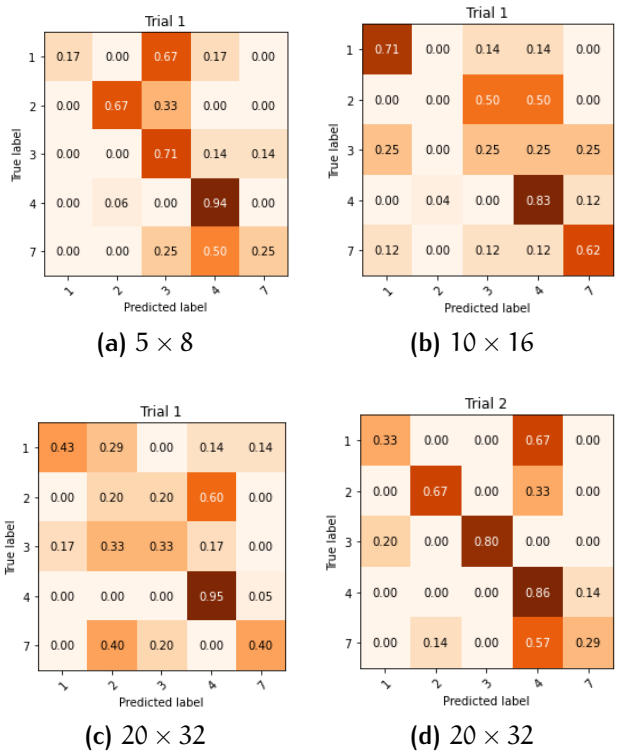
(a) Linear A **(b)** Linear B

(c) Poly A **(d)** Poly A

Figure 10: Kernel Choice: Linear vs. Polynomial
$\texttt{bin\_size} : 10 \times 16,\ \texttt{period} : 5\text{s},\ \texttt{overlap} : 0\text{s}$

been produced by a level 4 player. It was not determined why this occurred because time was not allocated to deciphering the reason.

Figure 10 shows the different nature of results achieved based upon the *kernel* chosen when initially instantiating an SVM. The other settings used in both trials were the exact same and were:
10×16 bin_size, 5 second period, 0 overlapping,
140 train_samples, and 45 test_samples. Both CMs are not accurate, but the trials using a polynomial kernel consistently predicted level 4 skill level more often than other skill levels for each row of discrete level proficiency. The reason for this is likely that the amount of data available between skill levels was not equal and the highest total amount of data per skill level was collected from level 4 players. This distinction was much more pronounced within lower period lengths.



(a) 5×8 **(b)** 10×16

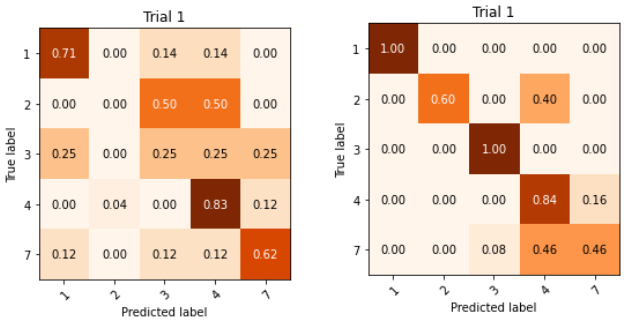
(c) 20×32 **(d)** 20×32

Figure 11: Different Bin Sizes
kernel : linear, period : 30s, overlap : 0s

The *bin_size* ultimately played a non-significant role in the classifier's accuracy. The trials seen in Figure 11 showcase trials with a differing *bin_size*s. The classifiers shown used the exact same settings that were used in Figure 10 except a period of 30 seconds was used instead. The results not shown with different settings followed suit — little to no impact could be seen based upon the *bin_size* used.

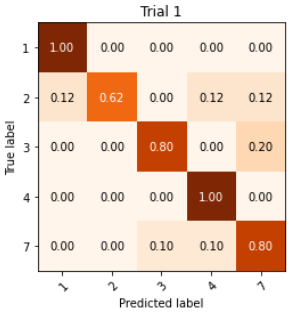
Figure 11 also showcases the inconsistency in results achieved amongst independent trials using the exact same settings. Both (c) and (d) represent the exact same feature vector but two different trials. Each of the CMs are quite different except for they both predicted level 4 players quite accurately. Other skill levels were not accurate and the wrong prediction that led to inaccuracy was different between both trials. However, it can be seen that the trials are more accurate overall than trials shown in Figure 10.

One setting that had noticeable effect on trial results was the *period* length. This is evident in the trials shown in Figure 12 differing only in the period length used. The overall accuracy of the CM produced in trials increased when the period length increased. Overall accuracy also increased when *overlap* was used.

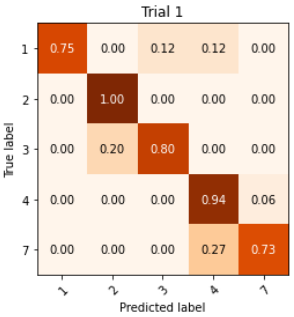


(a) 30 Seconds

(b) 60 Seconds

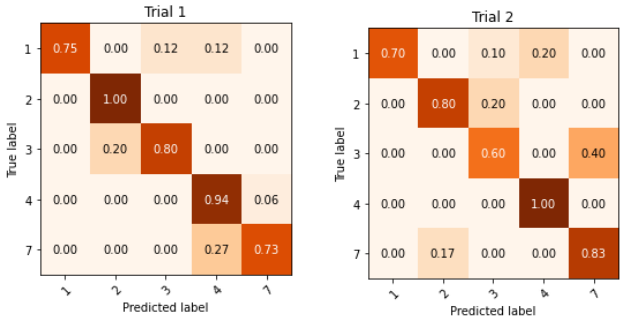


(c) 90 Seconds



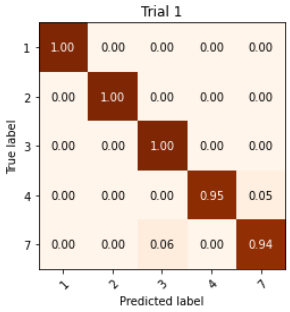
(d) 120 Seconds

Figure 12: Different Period Lengths
bin_size : 10×16, kernel : linear, overlap : 0s

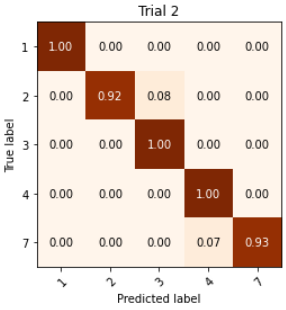


(a) No Overlapping

(b) No Overlapping



(c) Overlapping Used



(d) Overlapping Used

Figure 13: No Overlapping vs. Overlapping Used
bin_size : 10×16, kernel : linear, period : 120s

Figure 13 shows two trials with a 120 second period without the use of overlapping and two trials with overlapping used. It is clear that the use of overlapping increases the overall accuracy of the model. The reason for this is likely the increase in the data available to train upon. When overlapping is used, 280 training samples and 90 tests samples were used in contrast to 140 training samples and 45 testing sample when it was not used.

Top results and testing generalizability

The results achieved by the classifiers shown in Figure 13 were amongst some of the highest overall accuracies achieved from different setting combinations. After trials were run on all different setting combinations, top results were narrowed down. Narrowing down results took into account recall, precision, and F1 score in addition to overall accuracy. To obtain these statistical measures, confusion matrices were converted into a *pandas-ml* [6] confusion matrix which automatically calculates and provides these statistical values with ease to the programmer. A tolerance to have each of the statistical measures above 70% was set to find some of the best setting combinations.

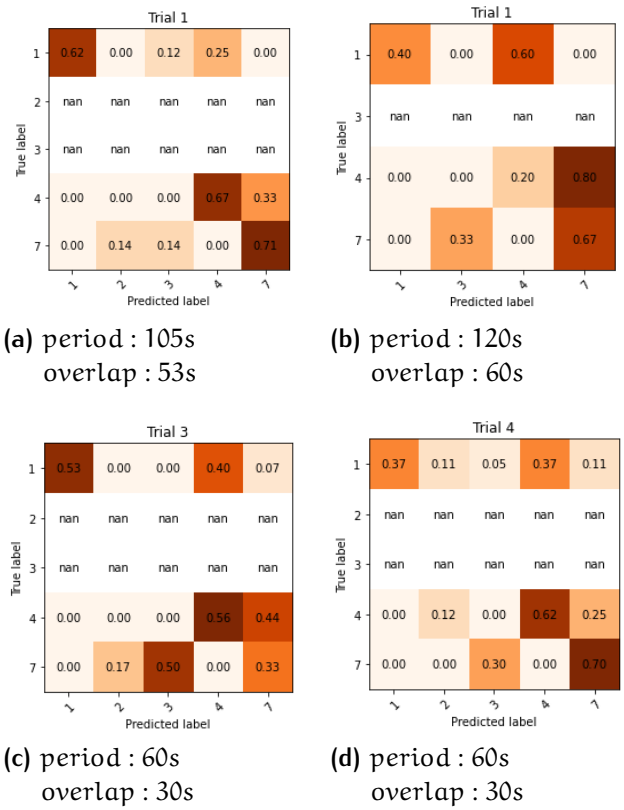
There were 29 unique combinations that were above the tolerance set which can be seen in Table 2. The lowest period length was 60 seconds with overlapping. From there, each increase in period length was represented. The linear kernel was represented more than the polynomial kernel with 19 occurrences compared to 10. All bin sizes were represented. Noticeably, many more results depended on overlapping being used than no overlapping.

Up until this point, the classifiers created and tried were being tested on participants from which data they had seen before. To clarify, the testing set did not contain data samples found within the training set, but it did contain data from individuals represented in the training data. For a classifier to be successful, it must be generalizable to data it has never seen before or, in the case of this study, individuals it has never seen before.

The top feature variable combinations derived in Table 2 were used as the basis to create classifiers to test on totally unseen individuals. Data from a level 1 individual, a level 4 individual, and one level 7 game was left out of the training set. Then, the classifier was trained as normal. Finally, it was tested on data from the games data that was entirely void from the training set.

Table 2: Top Setting Combinations

bin_size	period	overlap	kernel
10x16	60s	30s	linear
10x16	90s	45s	linear
10x16	90s	0s	linear
10x16	105s	53s	linear
10x16	120s	60s	linear
10x16	120s	0s	linear
20x32	60s	30s	linear
20x32	90s	45s	linear
20x32	90s	0s	linear
20x32	105s	53s	linear
20x32	105s	0s	linear
20x32	120s	60s	linear
20x32	120s	0s	linear
5x8	60s	30s	linear
5x8	90s	45s	linear
5x8	105s	53s	linear
5x8	105s	0s	linear
5x8	120s	60s	linear
5x8	120s	0s	linear
10x16	90s	45s	poly
10x16	105s	53s	poly
10x16	120s	60s	poly
20x32	105s	53s	poly
20x32	120s	60s	poly
5x8	60s	30s	poly
5x8	90s	45s	poly
5x8	105s	53s	poly
5x8	120s	60s	poly
5x8	120s	0s	poly



(a) period : 105s
overlap : 53s

(b) period : 120s
overlap : 60s

(c) period : 60s
overlap : 30s

(d) period : 60s
overlap : 30s

Figure 14: Unseen Data Trial Results
bin_size : 10×16, kernel : linear

Four trial results can be seen in Figure 14. These results were selected randomly from all the trials conducted using a *linear kernel* but are indicative of the overall outcome produced. All the selected results also used a *bin_size* of 10×16. Any rows with "NaN" indicate a skill level not present in the test set but it was amongst the predicted skill levels by the classifier. The trend seen in testing unseen individuals was inconsistency and poor overall accuracy. This is evident in (c) and (d) which happened to use the exact same settings. No classifier produced using the Gaze Bin feature vector was able to generalize to completely unseen data. The success seen in Figure 13 is likely attributed to the classifiers becoming overfitted to the training data — it learned the data too well.

2.2.4 Looking at Different Gameplay Stages

One final attempt to create a classifier was used in conjunction with an SVM and builds on top of the Gaze Bin feature vector. It was brainstormed that creating separate classifiers for different stages of gameplay could be more plausible than attempting to create one that oversaw all stages of gameplay. If correct, the classifiers could be combined into one overall classifier that would likely be able to predict a

player's proficiency in Dota 2.

This is plausible because a game of Dota 2 requires different gameplay as well as different knowledge to be drawn upon for different stages of the game. Amongst experienced players, there a different stages within a game that are dependent on how much time has occurred and what is happening within the game. Three acknowledged stages of gameplay include: the early-game, mid-game, and late-game.

A game of Dota 2 does not necessarily reach each stage because the game could end before the stage is reached, but every game has an early game as it begins right when the game starts. There is no concrete definition of when one stage ends and another begins but for the purposes of this study time cutoffs were prescribed. The definitions are as follow: early-game is 0–10 minutes, mid-game is 10–25 minutes, and late-game is 25+ minutes. These times are based upon personal experience and knowledge from playing and watching Dota 2 for eight years along with being a player with level 7 skill proficiency.

Training and testing classifiers was almost identical to what was done at the end of Section 2.2.3. The only difference was that the training and testing data sets for each specific stage classifier were limited to data that occurred during the classifier's specified time range. The data available to train and test on was quite limited.

Trials were also first conducted without omitting any individuals from the training data set. Trials were then conducted by leaving out specific individuals from the training set and testing the classifier specifically on samples from individuals left out. The results for both were near-identical to those presented in the previous Section 2.2.3. This method for creating a specific time classifier was not adequate to classifying unseen individuals.

2.2.5 SVM Conclusions

The top results from trials using an SVM classifier were feature vectors with a longer period setting and used of overlapping. The top feature vector used a linear kernel, a period length of 120 seconds, and used overlapping. Nonetheless, this feature vector was not generalizable and unable to successfully classify a player's Dota 2 skill level.

2.3 A LONG SHORT TERM MEMORY CLASSIFIER

Using a long short term memory (LSTM) network was the next direction decided upon to create a working Dota 2 skill classifier. An LSTM was chosen primarily because the nature of the data is temporal and sequential. A whole game of Dota 2 played can be thought of as producing one sequence of gazes. That sample sequence is further broken down into multiple discrete sequences to create samples to train upon, as was also done in Section 2.2.

Each sequence of data takes place across a set period of time. What is looked at presently can and does determine what will be looked at in the future. A player may gaze at their minimap in the bottom left hand corner and see an enemy approaching to their right, giving them information about the current state of the game — the locations of opponents. From this gaze, they may choose to look to the left side of their screen to run away from the enemy, the right side of their screen to try and engage the enemy, or choose to look at something entirely different. Either way, their gaze choice is almost certainly dependent on their previous gaze. Using an LSTM network would help ensure that relationships between gazes and their sequence could be learned as well as maintained to help accurately predict a skill level based on a sequence of gazes.

PyTorch [7] was used to create an LSTM classifier. One of the main benefits of using PyTorch is that it is simple to perform training and testing with the use of batches on a graphics processing unit (GPU). Using batches significantly speeds up training and testing of a classifier by training and testings on multiple samples at a time rather than one at a time. This is an important point to note because conducting a trial took significantly longer using an LSTM classifier compared to an SVM classifier.

Table 3: LSTM Classifier Architecture

	Layer Type	Parameters Used
1	Linear	in_features, out_features
2	LSTM	input_size, hidden_size
3	Linear	in_features, out_features

2.3.1 LSTM Layers and Feature Vector Considerations

The number of layers used in creating an LSTM for each trial was minimal. The basics of the architecture can be seen in Table 3 as well as the parameters that were used when instantiating each layer as re-

ferred to in PyTorch's [7] *torch.nn* documentation.

Explaining the settings

The major difference between the feature vector used in Section 2.2.3 and the one used with the LSTM classifier is the actual input used for training and testing the classifier. The input for the LSTM is a sequence of samples where each individual sample takes the form: [normalized GazePointX, normalized GazePointY, t] where t is the normalized time the gaze occurred, if included as determined by a boolean value *keep_time*. The number of samples in a sequence is dependent on the *period* length used.

The settings consisted of: *period, overlapping, keep_time, epochs,* and *subsampling_size*. The *period* and *overlapping* are identical to what was seen previously in Section 2.2.3. How many iterations the model trained before being evaluated was the number of *epochs* used.

Model parameters that were changed included *use_dense_layer* and *hidden_dimension_size*. *Hidden_dimension_size* was dependent on the number of individual samples within a sequence. Although they are not necessarily considered a part of the settings which produce the feature vector, these two model parameters will be included within the setting descriptions from now on for ease of clarification.

To prevent classifiers becoming too large with too many learnable parameters subsampling was used. Subsampling took place after data was split up into discretized period samples. Each sample period was reduced in size by keeping one in every four individual samples. For example, one second of gameplay produces approximately 60 individual samples because the eye tracker collects data at 60 Hz. If a period of one second is used, with five seconds of total gameplay, 5 sample periods (sequences) will be created with each sequence containing 60 individual samples of the aforementioned individual sample form. The time difference between samples would be: $4/60 - 0/60 = 0.066\dot{6}$ seconds or approximately 66 milliseconds. In contrast, the average visual reaction time is 284 milliseconds [1]. The amount of data lost is acceptable because it is lower than a realistic visual reaction time which means that the difference in time between gazes is still less than an individual's ability to react to what they are seeing during gameplay on screen.

Explaining the layers

The first layer is a linear layer, sometimes referred to as a dense layer in other popular machine learning libraries. Its individual layer parameters are *in_features* and *out_features*. The *in_features* parameter is the number of of variables contained within an individual sample from the input sequence — 2 if time is not included and 3 if time is included. The *out_features* was arbitrarily chosen to be 32 and is a number that was planned on being adjusted to see how it affects the results. This layer was initially thought to be a pseudo-embedding layer for the initial input. The use of this layer was also a setting consideration as it could be left out. Past early trials, it was discussed and resolved that this was an incorrect use of a linear layer so this layer was left out entirely for later trials.

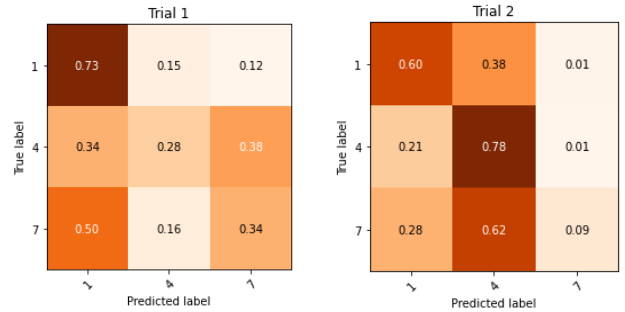
The second layer is the LSTM layer. If the first linear layer was included, the *input_size* to this player would the *out_features* of the linear layer. However, if the first linear layer was not used, the *input_size* would be the same as what would be used for the first linear layer had it been used. The *hidden_size* was set to be the same as the number of individual samples within an input sequence.

The second linear layer *in_features* was equal to the *hidden_size* in the LSTM layer. The *out_features* was set to the number of distinct skill groups present in the training data set being used in each specific trial.

2.3.2 Running a Trial for an LSTM Classifier

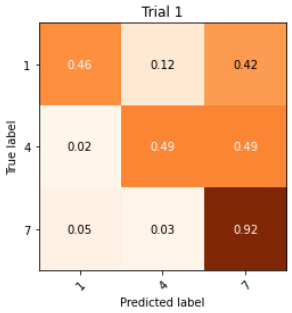
Trials were conducted similarly to what was seen in Section 2.2.3. However, it was decided that the train and data sets would be gameplay data from unique individuals. Eye gaze data in the training set would be data from individuals not present in the testing set with an exception during early trials that the level 7 data is from the same player but the games it was derived from were not the same. The majority of trials using an LSTM classifier ensured no intersection of players found in the training set and testing set.

Each classifier trial was run four times to ensure consistent results. Every classifier used an SGD optimizer with $\texttt{learning\_rate} = 0.001$ and $\texttt{momentum} = 0.9$. The loss function (criterion) used was *nn.CrossEntropyLoss* which combines *nn.LogSoftMax* and *nn.NLLLoss* into one criterion [7]. After the classifier was created, it performed learning iterations over the training data set for the determined num-

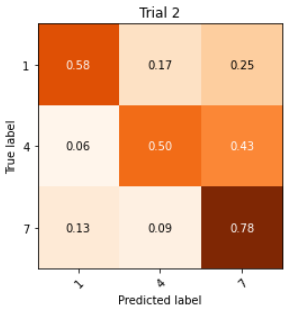


(a) 30 Seconds
hidden_dim : 300

(b) 30 Seconds
hidden_dim : 300



(c) 5 Seconds
hidden_dim : 75



(d) 5 Seconds
hidden_dim : 300

Figure 15: Different Period Lengths
$\text{epochs} : 50,\ \text{overlap} : \frac{\text{period}}{2},$
$\text{lin\_out} : 32,\ \text{keep\_time} : \text{False}$

ber of *epochs*. After, it was tested on the test data set to determine it's effectiveness in classifying a gaze sequence effectively.

2.3.3 Early Trials and Results

Early trials only included data from players with a skill level 1, level 4, or level 7. Although undetermined, it was thought that the difference between a level 3 and level 4 player may be more difficult to determine than the difference between a level 1 player and level 4 player or a level 4 player and a level 7 player. Thus, level 2 and level 3 data was not included to possibly aid in making classification easier between players skills levels further away from each other.

Trials performed early saw a noticeable difference in the effect of one setting used compared to the trends presented in Section 2.2.3. With $\text{epochs} = 50,\ \text{overlap} = \frac{\text{period}}{2}$ $\text{lin\_out} = 32$, and $\text{keep\_time} = \text{False}$ as the other settings, Figure 15 shows two classifiers using different period lengths i.e. a longer sequence of gazes as input.

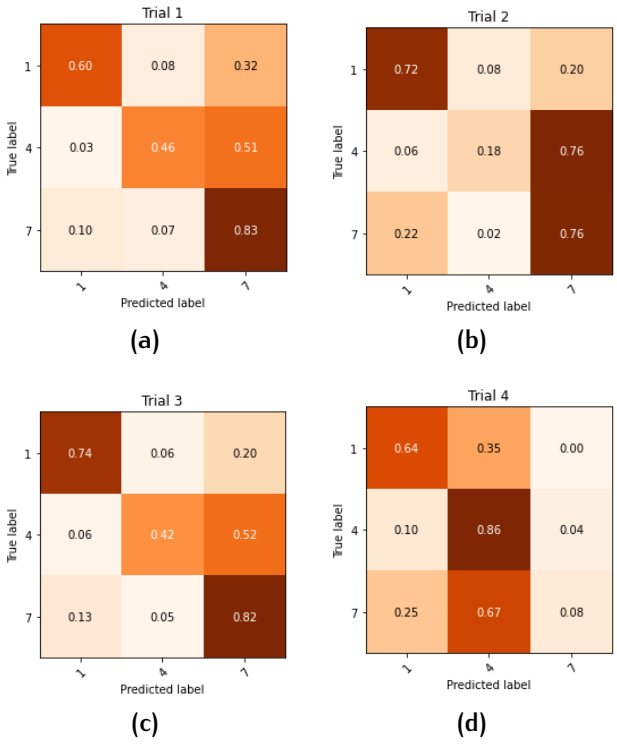


Figure 16: No Overlapping
epochs : 50, period : 5s, overlap : 0s, lin_out : 32,
keep_time : False, hidden_dim : 75

The trend noticed was that as *period* length increased, the overall accuracy of the classifier reduced. The trials using a period of 30 seconds had an overall accuracy ranging from 37.4%–48.4% whereas a period of 5 seconds ranged from 62.4%–69.1%. A smaller period produced higher accuracy and reduced inconsistency between trials.

The use of overlapping still increased the overall accuracy of models. Figure 16 shows trials using a 5 second period length with the same feature variables as in the previous paragraph except no overlapping was used. The overall accuracy ranged from 48.5%–69.1%. Although the highest accuracy achieved was on par compared to the trials using overlapping shown in the previous paragraph, the lack of overlapping appeared to produce an inconsistency within the trials.

An increase in the amount of *epochs* showed an increase in the overall accuracy achieved. This is an expected result as in increase in epochs allows the classifier to potentially learn more because it iterates over the training data more times. With the use of 100 *epochs*, the overall accuracy was between 52.8%–76.7%. Trials can be seen in Figure 17.

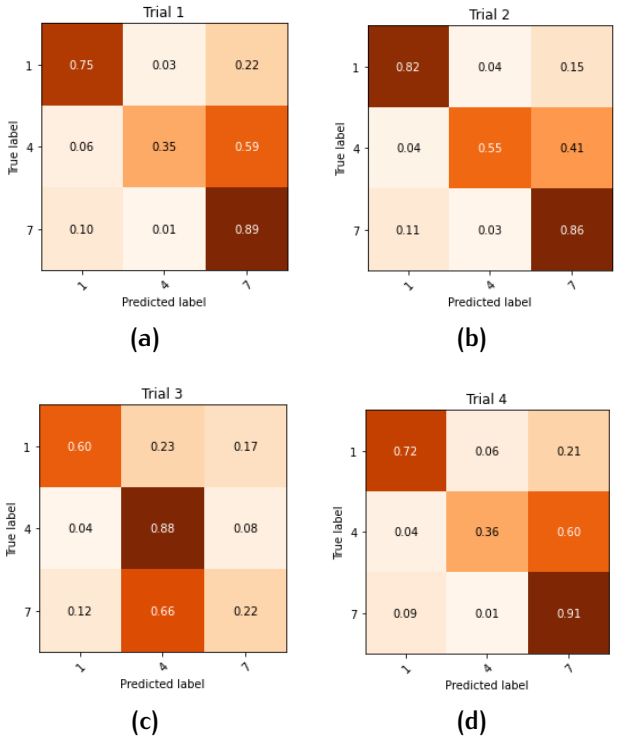
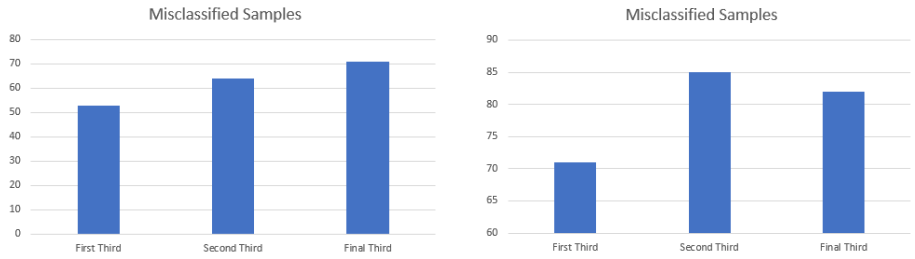


Figure 17: Training with 100 Epochs
epochs : 100, period : 5s, overlap : 3s lin_out : 32,
keep_time : False, hidden_dim : 75

2.3.4 Looking at Different Gameplay Stages Again

Because the highest accuracy of classifiers developed was not generalizable, the idea mentioned in Section 2.2.4 was revisited. However, rather than creating separate classifiers for each different stage of the game, it was faster and more straightforward to check what part of the game a misclassified sample took place. The number of times a gaze was incorrectly predicted was tracked for each specific part of a game.

The games that this was tested on were roughly 33 minutes, 33 minutes, and 38 minutes in length. The games were divided into thirds to determine what part of the game a classification was incorrect in: the first third of the game, second third of the game, or final third of the game. Figure 18 shows the tallied results for two different classifiers instantiated and evaluated. It was decided that creating separate game stage classifiers would be highly unlikely to produce a more accurate classifier because there was not a significant difference between the different times of when samples were misclassified.



(a) Trial 1 **(b)** Trial 2

Figure 18: Count of Misclassified Samples by Gameplay Stage

2.3.5 Changing to Binary Classification

The highest classification achieved was with a low period of 5 seconds, regardless of other feature vector variables used. The results of these preliminary trials are not generalizable. This caused a pivot towards attempting to create a classifier that could predict if a player was skilled or novice. The player skill levels already in place were used to determine if a player was a novice — a skill level of 1 was labelled novice and any other skill level was labelled as experienced.

It was also decided shortly after switching to binary classification that the first linear layer would not be used anymore. The trial results following, unless otherwise noted, only used one level 1 individual and one level 4 individual to train as well as a different level 1 and different level 4 individual to perform classifier evaluation.

The final setting from the original listing, *keep_time*, showed minimal effect. The overall accuracy for no time included in the input ranged from 77.6%–78.9% and the overall accuracy for time included ranged from 78.2%–79.5%. The overall trend of including the normalized time within each individual sample was a slight increase in overall accuracy.

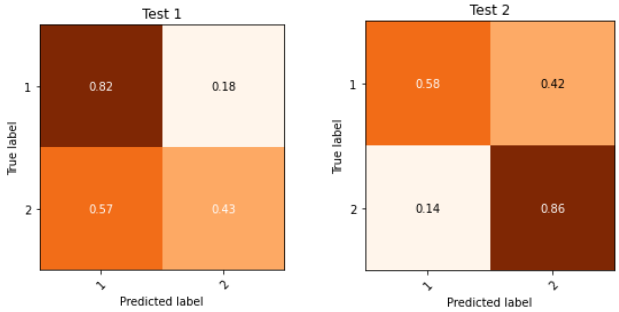
Unfortunately, due to a programming error, no more trials using a *period* of 5 seconds and an *overlapping* of 3 seconds were run. As a consequence, only data from trials using a period of 5 seconds with no overlapping will be referenced for the remainder of the paper. It is likely that better results would have been achieved with the use of overlapping, but it cannot be said for certain.

2.3.6 Labelling Gazes based upon Static UI Elements

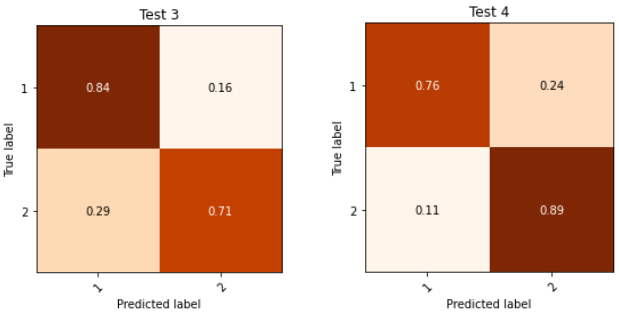
The next iteration in feature vector design changed the individual input format. Each individual input took the form $[gaze\_type, t]$

where t is the normalized time the gaze type occurred. There were many different gaze types defined which were based upon the highlighted static UI elements seen previously in Figure 1. The intuition behind this design was that the classifier would easily be able to learn a pattern or order of gazes that may be able to identify an experienced player. It was hypothesized that more experienced players have routine eye gaze habits that make use of UI elements. This approach would reduce the possible number of inputs drastically because there is a much smaller number of labels being used than possible pairings of normalized eye gaze coordinates.

In total, 9 gaze type labels were defined: *[radiant_heroes, clock, dire_heroes, minimap, gold_status, hero_items, hero_abilities, gameplay, and unclassified]*. More labels could have been used, but it would have required additional information beyond eye gaze data. For example, to determine if what was being looked at was gameplay or the in-game item-shop, it would be required to know when the player opened the shop by either hitting a specific keybind or manually clicking on the shop to open it. When the shop is open, it covers a portion of gameplay on the right hand side of the screen. It was not feasible to make such distinctions with limited time remaining.

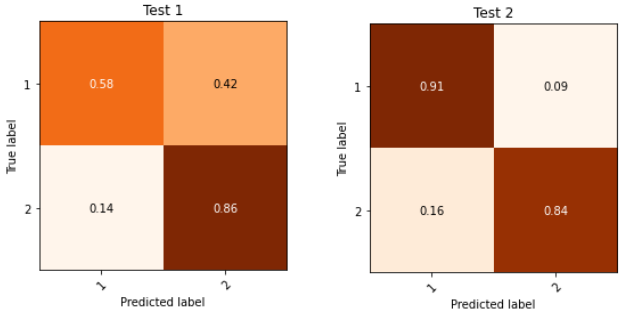


(a) 60.5% Accuracy **(b)** 73.1% Accuracy

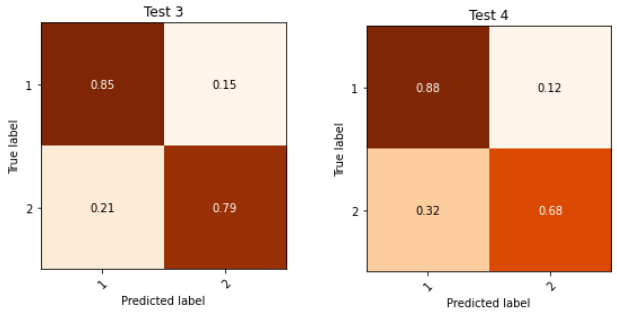


(c) 76.7% Accuracy **(d)** 83.4% Accuracy

Figure 19: Using Gaze Types
epochs : 100, period : 5s, overlap : 0s,
keep_time : True, hidden_dim : 75



(a) 73.5% Accuracy (b) 87.2% Accuracy



(c) 81.9% Accuracy (d) 77.0% Accuracy

Figure 20: Using Gaze Types
epochs : 300, overlap : 0s,
keep_time : True, hidden_dim : 75

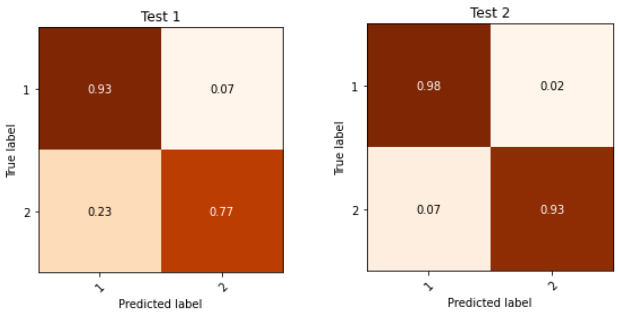
If a gaze occurred within a pre-defined area of the screen, it was labelled as the area it took place in. If it did not occur in a specific area, but was still determined to be on-screen, it was labelled as gameplay. If it was off-screen, it was labelled as unclassified.

A set of trials with 100 *epochs* can be seen in Figure 19. The overall accuracy ranged from 60.5%–83.4%. Although inconsistent, a higher overall accuracy was achieved. The results of increasing the *epochs* to 300 can be seen in Figure 20. It had an overall accuracy range of 73.5%–87.2% which was more accurate and more consistent.

2.3.7 Classification using Multiple Samples

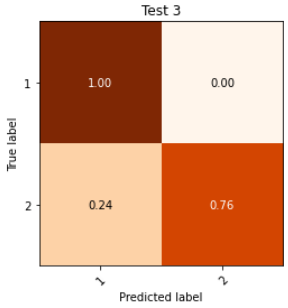
The final iteration on the classifier was a new method of evaluation. Instead of testing on one sequence, multiple sequences from the same player were tested on. The classifications were tallied and the majority prediction was the final classification for the sequence of samples.

Implementing this required altering the pre-processing step of the test data set creation. After splitting up test data into sequences, sequences would be grouped together depending on the *test_length*

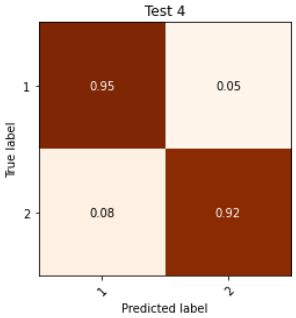


(a) 84.4% Accuracy

(b) 95.3% Accuracy



(c) 86.7% Accuracy



(d) 93.0% Accuracy

Figure 21: Evaluating with Multiple Samples
epochs : 300, overlap : 0s, keep_time : True,
hidden_dim : 75, test_length : 7

used. To ensure that there was not a tie in classification predictions, 7 was chosen as the number of sequences to test upon. For a 5 second period, this meant that 7 samples, each containing 5 seconds of gaze data, would be classified — a total of 35 seconds of gameplay. During evaluation, after the seventh sample was classified, the skill level predicted most was the final prediction for that test sample.

Figure 21 shows the results of this final alteration to a classifier. The accuracy was the highest achieved yet. However, the data used up until this point was still extremely limited. There were only two games of data within the training set and two games in the testing set. The results also show that when accuracy dropped, the experienced player was being classified incorrectly as a novice, not the novice player being predicted as experienced.

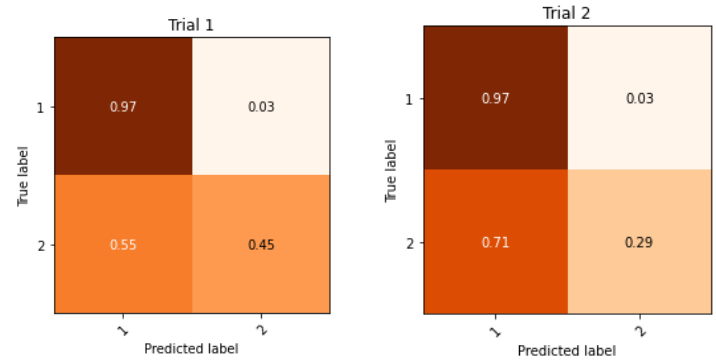
2.3.8 Study Data Collection

More data was accumulated during an official study amid the late stages of the thesis. The aim of the study was to collect data from two distinct individuals for each of the 8 original skill levels defined in Figure 4. There were 10 participants total, of which 1 the eye tracking

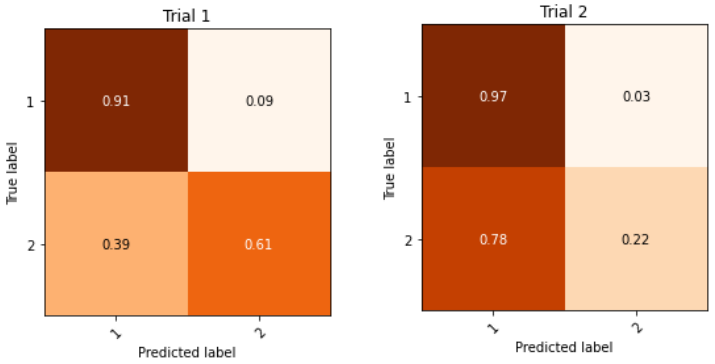
setup was not able to calibrate properly. The 9 remaining players all played a full game of Dota 2. The skill levels were: 1, 1, 1, 1, 3, 5, 7, 7, and 8.

Results from Study Data

The final classifier in the previous section was tested on study participants' data. It was trained and classified in the same manner, but either the experienced or novice player's data in the test set was replaced with a study participant's data. The success seen in the previous section was not generalizable to the new data introduced. The overall accuracies produced were lower and trials produced inconsistent results.



(a) Level 5 - 71.8% **(b)** Level 5 - 67.2% Accuracy

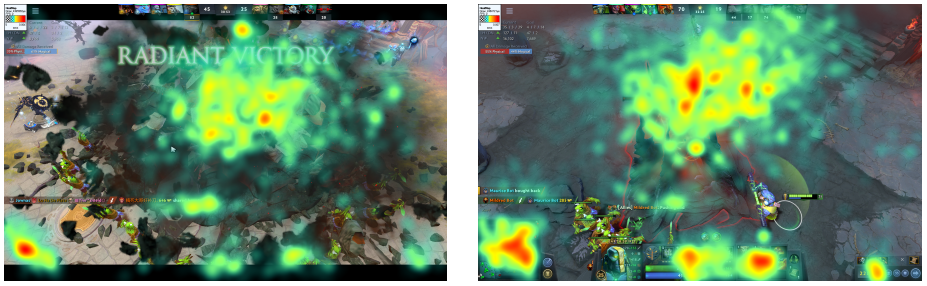


(c) Level 7 - 76.6% Accuracy **(d)** Level 7 - 59.4% Accuracy

Figure 22: Evaluating with Multiple Samples
epochs : 300, overlap : 0s, keep_time : True, hidden_dim : 75, test_length : 7

Figure 22 shows trial results from the newly collected data. The novice data was kept identical as the previous results shown thus far. The test data for the experienced player was switched to the study participant's data. The accuracy is above 50% for the trials, which is better than guessing, but this due to the classifiers ability to correctly

predict novices within the pilot study's data collection. The classifier is not capable of consistently predicting experienced players correctly meaning it is not generalizable.



(a) Level 7 Player **(b)** Level 8 Player

Figure 23: Level 7 vs. Level 8 Heatmap Comparison

An interesting result of the study data was the heatmap produced by the level 8 participant — a top 500 player within the North American region. The heatmap can be seen in Figure 23b. There is a stark difference between the level 7 player's heatmap produced and the level 8 player's heatmap produced. In contrast to the level 7 player's heatmap, there is much more gaze focus on the bottom portion of the screen where abilities, gold status, and items are displayed on the game's UI.

3 CONCLUSION

Two prominent machine learning libraries were employed in an attempt to create a classifier capable of distinguishing the different level of skill between Dota 2 players based on eye gaze alone. A support vector machine classifier was used with many different feature vectors. Yet, no classifier created was generalizable.

Next, a long short term memory network was instantiated with different feature vectors. Classifying players was switched to either novice or experienced. However, successful feature vector designs found using the limited data were not generalizable with any study data collected.

3.1 FUTURE WORK

Figure 23 clearly shows a distinction in gaze between an extremely high skilled player and a high skilled player. Although this study was unable to produce a classifier capable of distinguishing between players, it is likely possible that one could be produced with a different approach.

Nonetheless, one of the severe limitations of this study was the amount of eye gaze data available. A focus on collecting more data would likely be the starting point for future work. One approach to achieve this could take advantage of using webcams to create a tool which people could voluntarily use to record their eye gaze from Dota 2 gameplay. PyGaze [3] provides an example of how this is achievable.

REFERENCES

[1] Human Benchmark. *Reaction Time Statistics*. 2020. URL: https://www.humanbenchmark.com/tests/reactiontime/statistics (visited on 04/17/2020).

[2] E. T. Brown et al. "Finding Waldo: Learning about Users from their Interactions." In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1663–1672.

[3] Edwin Dalmaijer. *Webcam Eye Tracker*. 2015-06-02. URL: https://www.pygaze.org/2015/06/webcam-eye-tracker/ (visited on 09/2019).

[4] Y. Liu et al. "Who is the expert? Analyzing gaze data to predict expertise level in collaborative applications." In: *2009 IEEE International Conference on Multimedia and Expo*. 2009, pp. 898–901. DOI: 10.1109/ICME.2009.5202640.

[5] Duane G. Millslagle, Bridget B. Hines, and Melissa S. Smith. "Quiet Eye Gaze Behavior of Expert, and Near-Expert, Baseball Plate Umpires." In: *Perceptual and Motor Skills* 116.1 (2013). PMID: 23829135, pp. 69–77. DOI: 10.2466/24.22.27.PMS.116.1.69-77. eprint: https://doi.org/10.2466/24.22.27.PMS.116.1.69-77. URL: https://doi.org/10.2466/24.22.27.PMS.116.1.69-77.

[6] pandas-ml. *pandas-ml*. 2019. URL: https://pandas-ml.readthedocs.io/en/latest/conf_mat.html (visited on 12/2019).

[7] PyTorch. *FROM RESEARCH TO PRODUCTION - An open source machine learning framework that accelerates the path from research prototyping to production deployment.* 2020. URL: https://pytorch.org/ (visited on 02/2020).

[8] M. Raab and J.G. Johnson. "Expertise-based differences in search and option-generation strategies." In: *Journal of Experimental Psychology: Applied* 13.3 (2007), pp. 158–170. URL: https://doi.org/10.1037/1076-898X.13.3.158.

[9] Jesus Rodriguez. *The Science Behind OpenAI Five that just Produced One of the Greatest Breakthrough in the History of AI*. July 2018. URL: https://towardsdatascience.com/the-science-behind-openai-five-that-just-produced-one-of-the-greatest-breakthrough-in-the-history-b045bcdc2b69.

[10] scikit-learn. *scikit-learn - Machine Learning in Python*. 2019. URL: https://scikit-learn.org/stable/ (visited on 10/27/2019).

[11] Valve. *Dota 2 Update - February 11th, 2020*. 2020. URL: http://www.dota2.com/news/updates/58435/ (visited on 03/29/2020).

[12] M.R. Wilson et al. "Gaze training enhances laparoscopic technical skill acquisition and multi-tasking performance: a randomized, controlled study." In: *Surg Endosc* 25 (2011), pp. 3731–3739. URL: https://doi.org/10.1007/s00464-011-1802-2.