PHYS3888 Final Report

Pablo Bonilla (SID: 480344010)* and Jamie Hepburn (SID: 480273835)†
School of Physics, University of Sydney, NSW 2006, Australia
Word Count: 3926
(Dated: June 4, 2020)

Innate physics skills are of paramount importance in developing the interdisciplinary instrumentation to tackle the ever increasing number and complexity of problems present in our current society. In this paper we demonstrate how the merging of physics, statistics and computer science ideas allow us to develop a final product aimed to reduce the short term effects of Digital Eye Strain (DES). We use the principles of Electrooculography (EOG) signals together with hardware and software from the Backyard Brains team to collect eye movement gestures into a computer where we filter the raw data and identify the eye signals in the input stream. We employ supervised machine learning models to train classifiers that accurately differentiate between up, down, left, right and blink eye movements. Finally, we use the human computer interface (HCI) to play a Tetris game, aimed at engaging users to exercise strained eyes in a fun and engaging manner. The results obtained from our best classifier on the all-eye movement live testing (78%) is comparable to that found in the literature, however we were limited to single person testing.

I. INTRODUCTION

The evolution and nature of prominent, well funded scientific research is intimately linked with the demands of society and the interest of the government which continuously increase in complexity ([1, 2]). The ever increasing challenges have prompted the amalgamation of separate scientific domains, unifying the skills from each discipline to form an indispensable toolbox in tackling the modern multifaceted obstacles. In particular, the growth of physics over the last century is firmly tied to the growth of its interdisciplinary component. In 1910 only 8% of the scientific literature were physics publications whilst this number has now increased to $\sim 22\%$, with 12% of these being in interdisciplinary fields ([3, 4]) (see Fig. 1)

The practicality of physics in modern interdisciplinary research stems from its key paradigms ([5, 6]) which include the abstraction of complex multivariate problems to their core fundamental variables and their connections, a deep understanding of ubiquitous phenomena like waves, energy and entropy and the phenomenal computational prowess of the modern physicist. The latter allows for the simulation of physics and conduction of experiments, provides fast numerical analysis tools and facilitates quality research output without the need for heavy granular theoretical knowledge on a topic ([7]). The research presented in this paper epitomises the influence and importance of key physics skills in developing an innovative product in an interdisciplinary setting to answer societal demands.

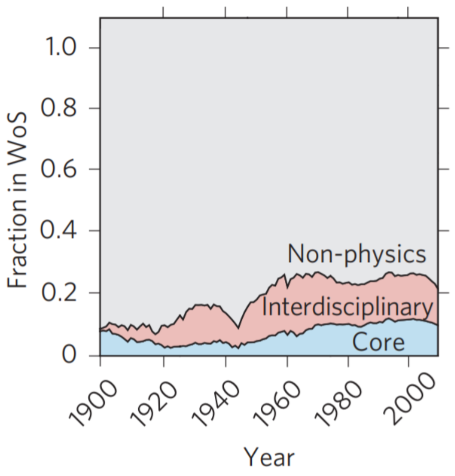

FIG. 1: The growth of Physics and its interdisciplinary component over the past century. The figure shows the fraction of the total publications in the Web of Science (WoS) that are physics-based, including those that only contribute to the field of Physics and those that are related to interdisciplinary fields. Image Source: [3]

Poor eye health is a prominent issue in our technology-driven society, especially amongst individuals who spend a large portion of their day in front of a computer screen ([8]). The problem is formally coined digital eye strain (DES) or computer vision syndrome (CVS) ([9]) and can have detrimental short term consequences such as headaches, dry and itchy eyes, blurred vision and increased tiredness ([8, 9]) as well as long term eye damage, particularly among the younger population, like a reduction in the calibre of retinal arterioles which can lead to symptoms of cardiovascular disease including obesity and metabolic syndrome ([9, 10]). Along with the known adverse health effects, DES has also been shown to severely impact an individual's productivity at work ([11]). It is therefore natural that a large portion of research is devoted to developing strategies to minimise DES ([8, 9]). These studies are only becoming more relevant due to the staggering and continual increase of computer use in modern society. By 2019 99% of adults aged 16-44 were

* jbon0136@uni.sydney.edu.au
† jhep7570@uni.sydney.edu.au

internet users in the UK ([12]) and over 80% of adults in the USA used digital devices for over 2 hours daily, of which 59% reported DES symptoms ([13]). Amelioration techniques for DES include lubricating eye drops, the use of computer glasses, correcting pre-existing refractive eye errors through surgery or otherwise and regular eye care including taking breaks and training the eyes ([8, 9, 14]).

In this paper we exploited the latter non-invasive DES management strategy to create a visual training product which allows users to perform targeted eye motions in a fun and engaging manner. The final product is a modified version of the renowned Tetris game, which is playable with the eyes. With the suggested way of playing detailed in Section IV we believe that the controlled eye movements that the game demands coupled with the enjoyment of playing Tetris will aid in regular ocular training which can help mitigate the temporary effects of DES.

In order to link the eye movement of the user directly to an electronic system we formed a human-computer interface (HCI) by exploiting the physics of electrophysiological wave signals. Eye movement data can be non-invasively collected by measuring electric potential variations between a pair of electrodes placed on either side of the eyes of the user. The collected oscillatory electric signal is termed electrooculogram (EOG) ([15]). The detection and accurate classification of EOG signals has been of paramount importance in health research and, among other things, has led to eye controlled wheelchairs for people with disabilities ([16]), virtual keyboards ([17]) and activity recognition software ([18]). EOGs can be delineated in terms of physics principles. The human eye can be represented as a dipole with the cornea as the positive pole and the retina as the negative end (see Fig. 2) ([19]).

When the user looks left (right), the dipole shifts so that the positive cornea moves closer to the left (right) electrode, making it more positive and thereby inducing a potential difference between the electrodes. EOG signals can also be produced from up, down and blink eye movements ([15, 20]). EOG signals are very faint, of the order of 100 mV, and therefore have to be amplified before being fed into the recording system. The process of EOG signal collection through electrodes, signal amplification and injection into the computer system is done using the Spiker Box provided by the Backyards Brain (BYB) team ([21]). The two electrodes through which we measure the voltage difference are attached to the forehead through a headband and there is a third electrode, called the ground electrode, which can take either of 2 positions (see Fig 3). With electrode set up 1 the SpikerBox can differentiate between left and right eye movements and with set up 2 it can detect blink signals and according to the BYB team it can poorly detect up and down signals. We however found that electrode set up 2 could accurately differentiate between all 5 eye movements, namely up, down, left, right and blink with the help of supervised ML. See Section II for further discussion on data collection.

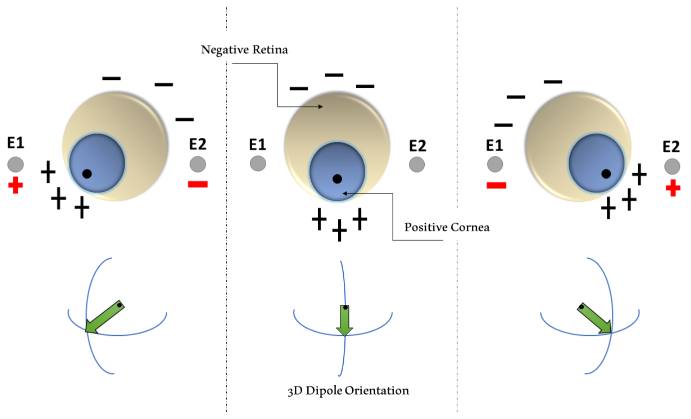

FIG. 2: The eye as a dipole. The back end of the eye, termed the retina has a net negative charge and the front end of the eye ball, termed the cornea has a net positive charge. The charge differential between the eye extremities leads to an electric dipole. The change of this dipole as the eye moves can induce a change in voltage between 2 fixed electrodes attached to the forehead of a human. In the figure the electrodes are called E1 and E2. Image Source: created by the authors.

Once the data collection process is complete we need to accurately differentiate between the different possible eye movement signals which are represented on the stream input as large amplitude deviations from the equilibrium position. We call the code that achieves the differentiation the classifier. To implement the classifier we turn to supervised machine learning (ML), an indispensable skill from our Statistic Major colleagues. ML, and more broadly the field of artificial intelligence (AI), has spectacularly blossomed since the 1950s ([22, 23]) extending interdisciplinarily to fields like predicting the stock market ([24, 25]), forecasting sporting results ([26]), medical diagnosis of cancer ([27]), developing refined drugs in pharmacology ([28]) and in agriculture to boost production by identifying pests ([29]). The rapid multidisciplinary spread and high-impact of ML stems from its simple implementation and universal adaptation to different settings. The fundamental paradigm is to allow the machine to learn trends in a dataset with minimal human intervention. Supervised ML is a subset of ML where the models are trained with known results (labels) based on properties of the input data (attributes) ([30]). The trained model is a division of attribute space with each region predicting a different label. Learning of the model occurs by tuning the weight of different features so as to drive the topology of attribute space into a state of minimum error with respect to prediction on the features of the known input labels. The algorithm used to train the models is of utmost importance and an active research frontier in ML.Three universal, robust learning algorithms that we will take advantage of in this paper

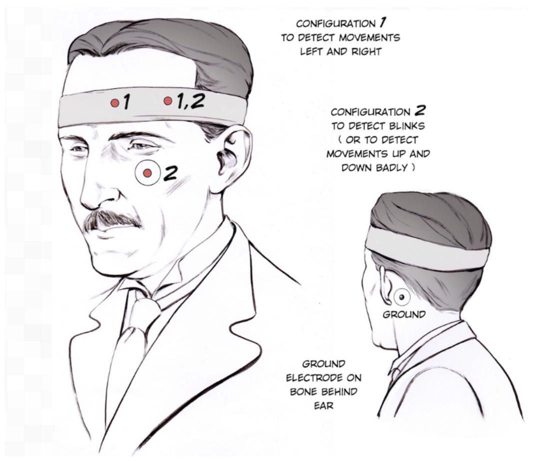

FIG. 3: The two possible electrode set ups. According to BYB's instructions set up 1 accurately differentiates between left and right eye movements while set up 2 can detect blinks and poorly detect up and down eye movements. With the help of machine learning we found that with set up 2 we were able to differentiate accurately between up, down, left, right and blink eye movements. [a]

[a] Image sourced from `https://backyardbrains.com/experiments/eog`

are Support Vector Machine (SVM) ([31]), Random Forest (RF) ([32]) and k-Nearest Neighbours ([33]). Once the model is trained, it can be fed attributes of potentially unknown data and a label prediction is obtained based on the position of the attributes in the trained feature space.

By unifying the physics of EOG signal data collection, amplification and filtering and the statistics of supervised Machine Learning we developed a human-computer interface (HCI) able to accurately predict left, right, up, down and blink eye movements. We effectively blend the physics and statistics disciplinary skills with computer science knowledge to use the label outputs of the HCI to play a Tetris game developed in Python, demonstrating the power of interdisciplinary collaboration in efficiently developing a product aimed to address societal challenges.

In Section II we thoroughly detail the methodology of data collection, filtering, training and testing of the ML models, label prediction, implementing the game and efficiently unifying all components. In Section III we present the results regarding data collection and processing, model evaluation and overall cohesiveness and discuss them in Section IV. Finally we conclude in Section V where we also propose avenues for future endeavours.

II. METHODOLOGY

In this section we elucidate the development of our product from the moment the user moves the eyes to when the Tetris piece moves in the game.

A. Data Collection

As described in Section I, in order to detect eye movement signals we begin by collecting the EOG data using Backyard Brain's (BYB) Spiker Box ([21]). The collection system consists of a headband with two closely spaced metal electrodes which go attached to the forehead of the user, with one electrode on either side of the left eye (it can also work with the right one). The EOG signals produced from eye movements are recorded as potential differences between these 2 electrodes ([15]). The ground electrode is an extra gel-coated patch which goes attached to either the cheek or the bone marrow. This provides a reference point which facilitates the measurement of the differential voltage. A metal lead is attached to each of the 3 electrodes which transfers the signal to the Spiker Box. The transferred EOG signal is of the order of 100 mV and contaminated with instrumental and confounding brain wave noise which is approximately equivalent to a 1^o eye deflection ([19]). When the signal reaches the box, there is an amplifier which boosts the EOG readings and filters out part of the noise ([34] Chapter 70). The amplified signal is fed into an A/D converter which transforms the analog signal into digital form ([19]) with a 10-bit resolution and a 10 kHz sampling rate ([35]) that is then fed into the computer system via a USB connection. The Brain Box is powered with a 9V battery and consumes it in approximately 15 hours of continuous connection. We loaded the live data input into the computer using the skeleton MATLAB code provided by Alessandro and the Backyard Brain team ([36, 37]).

B. Data Processing

In order to discriminate between signals in the input stream we must detect strain amplitude deviations from the equilibrium voltage. As such, the first 10 seconds of every SpikerBox use is devoted to raw data collection and the average over this time window is used to establish the baseline voltage.

When the equilibration time is over we continuously filter the input stream with a low-pass Butterworth filter ([38]) of order 5 at 5 Hz. Together with the Chebyshev, Inverse Chebyshev, Bessel and the Cauer filter models, the Butterworth filter is one of the classic analog filter designs and perhaps the most ubiquitous of all ([39]). To better understand the low-pass Butterworth filter we introduce some filter terminology. Suppose we want to get rid of all frequencies greater than f Hz in some time series data and to do so we apply a low-pass filter. The frequencies that the filter leaves unaffected are termed the pass band and those frequencies that the filter eliminates are called the stop band. Ideally we want the pass band (stop

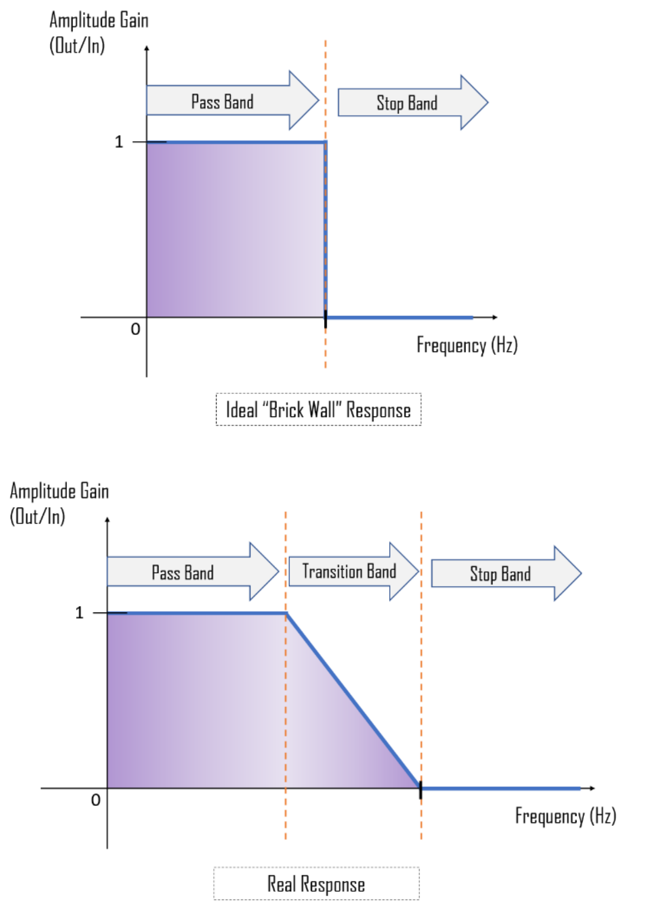

FIG. 4: Images depicting filter theory. (Top) Ideal response, known as the "brick wall" response in the literature, perfectly transmits all frequencies less than the cut-off frequency, f_c, and completely attenuates all frequencies greater than f_c. (Bottom) A real filter response. A transition band is present between the pass band and the stop band. Image source: created by the authors.

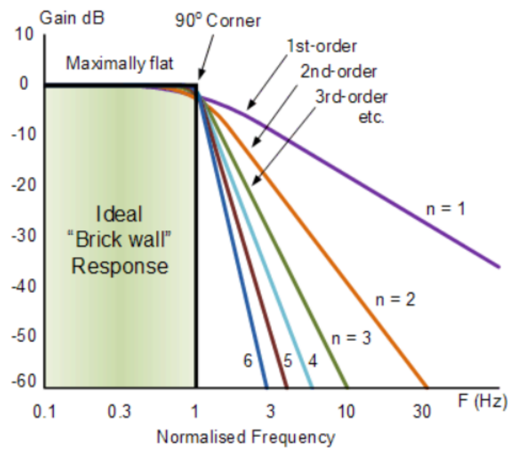

FIG. 5: Butterworth Filter design. It gives a maximally flat response in the pass band at the expense of a slightly wider transition band compared to other state-of-the-art filters. However, it makes up for the wider band by using higher order filters as shown on the image. Note how the units for the Y axis are decibals (dB), a logarithmic measure of power output. This is standard in the filter theory literature. [a]

[a] Image sourced from `https://www.electronics-tutorials.ws/filter/filter_8.html`

band) to be exactly all frequencies less (greater) than f, however no filter is perfect ([39]) and there will always be a non-sharp transition between the pass band and the stop band around f Hz. This transition is known as the roll off or transition band (see Fig. 4).

Two key desirable properties of a filter is to have as little a transition band as possible and to transmit the pass band with a gain close to 1, i.e. the pass band should be perfectly transmitted. The Butterworth filter addresses the latter, allowing the pass band to remain as invariant as mathematically possible when the filter is applied. Graphically this occurs when the filter response is as flat as possible in the pass band. A limitation of this filter is that we sacrifice on the ideal steep frequency cutoff between pass band and stop band, instead generating a wider transition band. However, higher order Butterworth filters attempt to improve on this limitation by making the roll-off steeper (see Fig. 5) ([39]).

For a justification on why 5 Hz was chosen as the cut-off frequency please refer to Section III and Section IV.

Once the data was filtered, the next step in the process was to detect signals as amplitude deviations from the baseline. The program detects a signal as a variation by more than 7% above the baseline or less than 6% below the baseline. This is justified by the fact that, after filtering and keeping the gaze straight, random noise about the equilibrium position never exceeded 4% above or 3% below the baseline and every detected signal had either a peak with value $>$ 7% of the noise or a trough with value $<$ 6% of the noise. We first observed these findings empirically and results can be found in Section III but it was superbly validated when each of the over 1000 eye movements performed in the project (both for training the models and in a live setting) was successfully detected as a signal. Once such a deviation is detected, we wait for 1.7 sec so that the full signal comes in, trim around it using basic array slicing in MATLAB, remove the baseline (i.e. subtract the average noise so that the equilibrium level is at 0) and save it. We collected over 100 signals for each of left, right and blink eye movements with set up 1 and each of left, right, blink, up and down looks with set up 2, however for the final products only the set-up 2 eye movements were used.

C. Model Training and Feature Selection

Once the signals were identified, the next task was to create a program which could accurately differentiate between the eye movement signals with minimal processing time using a supervised machine learning model, termed the classifier. When training a classifier program, it is paramount to incorporate features or observations in the given datasets that provide highly useful information for differentiating the possible signals [40].

We tested 2 different sets of differentiating features to use for our classifier. The first involved calculating the features using an existing R package called tsfeatures, which included a series of statistical summaries for time series data such as entropy and lumpiness ([41]). The second method consisted of developing our own features based on an analysis of each eye-movement signal (left, right, up, down and blink) to discover its differentiating characteristics. Due to the differences in amplitude and duration of the different signals we constructed 8 features that mostly relied on some variation of amplitude and time difference calculations. The eight features that were created are displayed in Table I. See also Fig. 6 for an example of a signal and our custom features labeled on it.

In order to quantify the accuracy and discriminatory performance of all known features, two statistical methods known as Gini importance and forward stepwise variable selection were applied. Gini importance (also known as Mean Decrease in Impurity), works by counting the number of times that a feature is used to split a decision or node into a separate tree and weights this number by the total number of samples split ([42]). Forward stepwise variable selection technique uses 5-fold cross validation calculations to iterate over all features, and with each step to successively include the feature that has the strongest ability to differentiate between signals ([43]). The results of these statistical methods were used to rank the features on their importance in differentiating eye signals, which allowed us to decide between using the existing features or the ones we created ourselves.

Once the features were created, we implemented 3 supervised ML models: Support Vector Machine (SVM), K nearest neighbours (KNN) and random forest (RF). Each model was developed to work on the eye signal data collected from the MATLAB pipeline for each of 3 datasets:

1. Left, Right and Blink eye movements
2. Up, Down and Blink eye movements
3. Left, Right, Blink, Up and Down eye movements

The accuracy of each model was determined by 5-fold cross validation, which separates the training data into 5 equal sets and runs 5 separate supervised training experiments with the testing data being a different set each time ([44]), see Section III for the results.

D. Game Development

A python game was developed using a template available within the python open source structure pygame ([45]). The game was an adaptation of the highly popular arcade game Tetris, in which falling blocks are manipulated with the goal of taking up as little space as possible ([46]). The original file was adjusted to create three possible game modes for playing Tetris, which corresponded to the three separate machine learning models for each game play condition. Within each game mode, the original keyboard inputs of the game were changed to the predicted outputs of the machine learning models so that a user could play the game effectively with only eye movements. See Table II for the different game modes established.

E. Live Development

To put all the elements together the MATLAB pipeline collects the live stream as described, packages the individual waves and saves them as files. The trained model is stored in an R file which detects the creation of the new wave file, opens it, calculates the attributes on the wave and feeds them to the trained model which predicts the label of the signal. The predicted label is then appended at the end of an "instructions.txt" file. The Tetris game, which is running from the terminal, detects the presence of a new instruction in the file "instructions.txt" and executes the appropriate command in the game.

III. RESULTS

We present the results for the MATLAB pipeline, the model evaluation and the product as a whole in this Section.

A. Data collection and processing evaluation

Refer to Fig. 7 for the morphology of each individual eye movement. Each of the movements presented was done with electrode set-up 2. On the same plots we indicate the region where random noise about the baseline is always found and the 2 threshold lines. Each eye movement signal performed for training or during live operation has always been found to have either a peak higher than the upper threshold boundary or a trough lower than the lower threshold line, or both.

See also Fig. 8 where we present the performance of the low-pass Butterworth filter for varying frequencies when applied to the raw data of a Right look.

The temporal resolution of the MATLAB pipeline is 1.72 seconds. Once the program detects the start of a signal as a deviation outside of the threshold it waits 1.7 seconds for the full signal to come in and then trims

Feature	Data Type
Minimum amplitude	Numeric Float
Maximum amplitude	Numeric Float
Amplitude variance	Numeric Float
Time difference between peak and trough	Numeric Float
Amplitude difference between peak and trough	Numeric Float
Time difference between 2 peaks	Numeric Float
Amplitude difference between 2 peaks	Numeric Float
Indicator of 1st peak being larger than 2nd peak	Numeric boolean

TABLE I: The 8 features used by the supervised ML models to accurately differentiate between up, down, left, right and blink eye movements. The same 8 features were used for all 3 games modes.

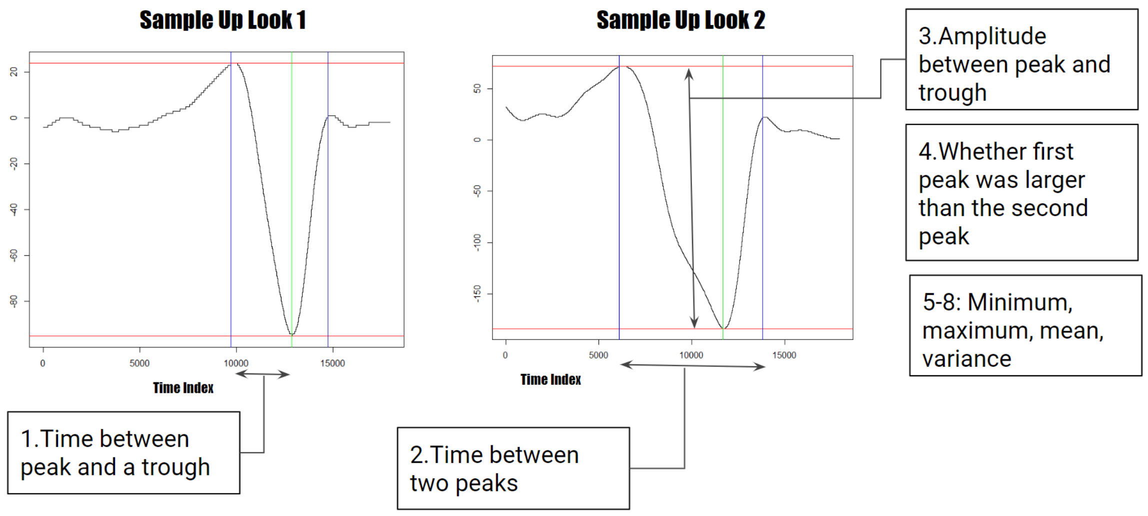

FIG. 6: Custom features utilised on two sample up movement signals

around it, saving it as a .wav file. The extra 0.2 seconds is a delay introduced at the end of each signal collection process to ensure that there is no mixing between different signals.

The total results collected using the MATLAB pipeline for the training of the final ML models were exactly 100 signals for each of up, down, left, blink and right eye movements with electrode set up 2.

B. ML model evaluation

We analysed the performance of the different features that were utilised using the statistical methods outlined in section II. The results of the step wise variable selection produced the accuracies shown in Table III for the respective classifiers using different features.

Along with analysing the performance of the features, we also calculated the average run time for the two separate feature classes. We found that the tsfeatures package had an average run time of 2.6 seconds per wave while the created features had an average run time of 0.2 seconds per wave. Since we observed higher classification accuracies and an overall reduction in classification time of 90% using our created features, we decided to use the eight features we created in the final product. An example graph for the step wise variable selection process portraying the relative importance of each custom feature is shown in Fig. 9

The performance of each Machine Learning model was then tested using 5-fold cross validation. As a consequence of the poor classification accuracies the KNN model was excluded from the future tests, see Fig.10.

C. Evaluation of product as a whole

Once the final product was developed, we conducted live gameplay testing using signals fed straight from the brainbox. The tests were conducted on the two remaining models to determine the final model for use in production.

Fig.11 shows the accuracies recorded for each of the three game modes. Due to a slight edge in performance during live testing of the all five eye movement condition, the Random Forest model was decided as the final model to use in the end product.

The final test involved out of sample live data, where a completely new user was brought in to test the product (Pablo's brother). One test was performed on the left,

Game Mode	Command no.	EOG Signal	Tetris game action
1 (LRB)	1	Move eyes left	Piece moves left
	2	Move eyes right	Piece moves right
	3	Blink	90^o clockwise rotation
2 (UDB)	1	Move eyes up	Piece moves right
	2	Move eyes down	Piece moves left
	3	Blink	90^o clockwise rotation
3 (ALL)	1	Move eyes left	Object moves left
	2	Move eyes right	Object moves right
	3	Move eyes up	90^o clockwise rotation
	4	Move eyes down	90^o anticlockwise rotation
	5	Blink	Fast drop object to floor

TABLE II: All game modes possible in the Tetris game developed. We show the eye movement performed and the corresponding action of the Tetris pieces in the game.

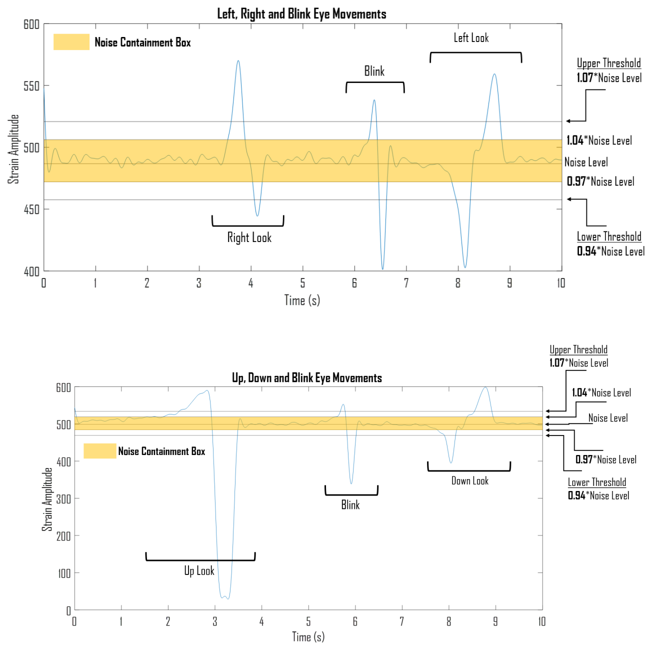

FIG. 7: (Top) Sample Collected data containing a left, right and blink eye movement. The yellow region represents the box within which random noise about the baseline has always been observed to be contained in. The upper and lower threshold lines indicate the amplitude deviation required for a disturbance to be called a signal. If either the trough or peak of a signal exceeds the corresponding thresholds then we wait 1.7 seconds, trim around it and package it for model training or live classification in R. (Bottom) Sample collected data containing a down, blink and up eye movement.

right and blink game mode. The recorded accuracy was 92%

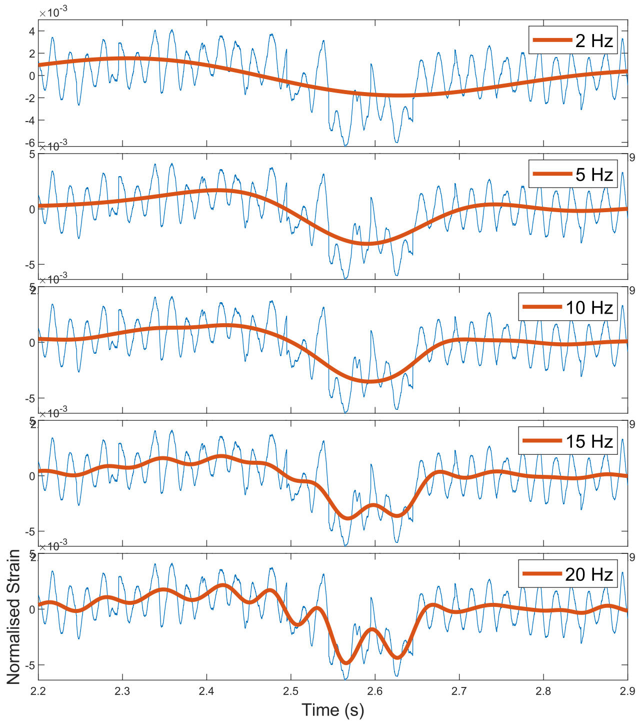

FIG. 8: Butterworth low-pass filter performance as a function of the cut-off frequency on normalised raw data (blue curve). The signal presented in the plots is a zoomed in Right look.

IV. DISCUSSION

For a detailed discussion of the results, see Individual Discussion assessment.

V. CONCLUSION

In summary the results illustrated the effectiveness of interdisciplinary study schemes within university settings. The knowledge from the combined data science

Model	ts features (%)	Custom features (%)
KNN	75.38	90.1
RF	84.75	95.5
SVM	84.21	93.3

TABLE III: Average accuracies obtained for each supervised ML model used and for each of the 2 sets of features calculated.

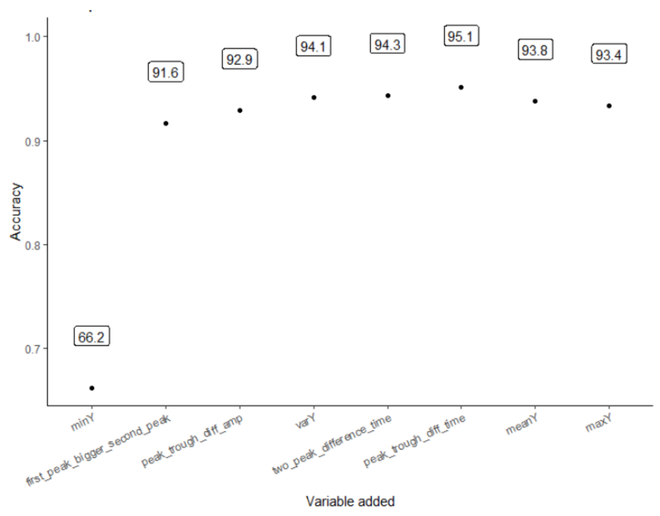

FIG. 9: Five fold cross validation accuracies with repeats after performing stepwise variable selection for the up, down and blink classifier. The image shows the most important features from left to right and the agreggate accuracy when the next most important feature is used to train the models.

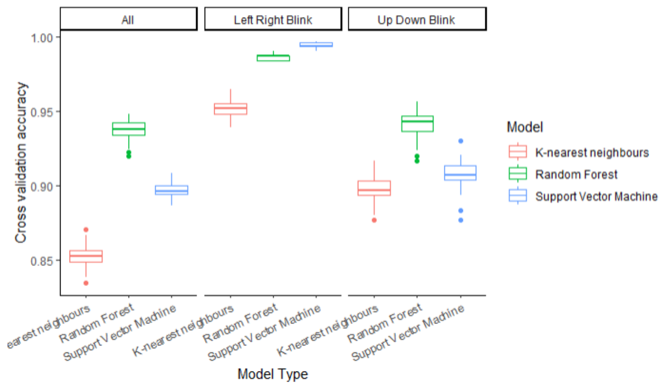

FIG. 10: Accuracy results for repeated 5-fold cross validation for each ML model in each eye movement condition.

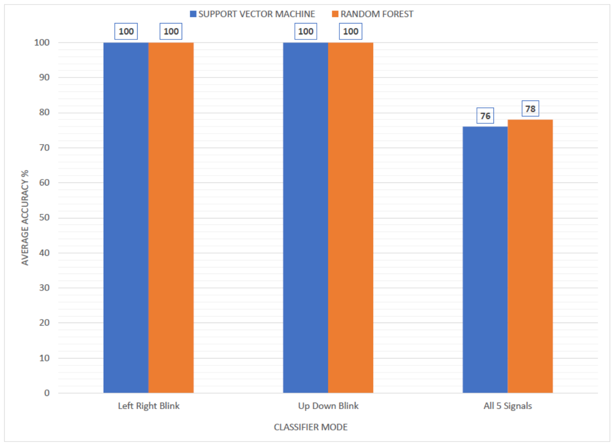

FIG. 11: Accuracy results live condition testing of RF and SVM models in each eye movement condition

and physics disciplines allowed for the development of a streamlined process where eye movement data is collected, filtered, streamed into a ML classifier, and imported into a game setting. Game-play includes short response time accurate inputs, which fits the needs of the original game design, thus meeting the suggested criteria for an engaging gaming experience ([47]). Given the high accuracy of the results, the methodologies and analysis techniques applied in this paper can be modified to many important research topics in the future.

In terms of eye training, this method and the resulting product can be used as a template to create many more simple and engaging eye movement games. One study has already shown that engaging in eye training regimes along with regular breaks and neck movements can reduce the symptoms of eye related issues like Computer Vision Syndrome ([48]). We envisage that including games rather than simple eye movement task may assist to increase participation and engagement in the training regimes, particularly for younger participants. Additionally, the product can aid in the assistance of more advanced eye training regimes in elite sports such as baseball ([49]). Aside from eye training, we also believe that the formation of accurate and robust eye signal classifier models will help improve research methodologies for those affected by motor control disabilities. By adapting the methods of data collection and processing, improvements can be made for keyboards that can be used solely with eye movements [50], which will allow faster communication due to low processing times. Furthermore our product demonstrates competitive accuracies and classification times of EOG signals with a relatively affordable product for the general public, epitomising the universality of our results and product development.

ACKNOWLEDGEMENT

We would like to thank Stacey, Ben, Chloe and Suri for being awesome collegues during the project, training and evaluating the machine learning models and providing us with Figures 9 and 10. We would also like to thank Jean and Alessandro for their continual support and responsiveness throughout the semester.

STUDENT CONTRIBUTIONS

Chloe

As a data student with computer science background, my main role is to develop and maintain our game product, help generate initial training set using time series features and later on help select model using Random Forest Ensemble Algorithms. For game development, I used open source game structure 'Pygame', implemented a 'Tetris' game logic and UI. I worked with Pablo to rebuild the events receiver interface which fixed the limitation of original game structure that only allows input from keyboard and cursor from users. The final product can consecutively read instructions generated by classification model in R and correctly respond movements with negligible latency. I worked with Stacey to make an event identification algorithm and used that to generate training dataset, while Stacey investigated physics properties of wave signals, I focused on fixing data structure problems and code logic bugs. I worked with other data teammates on model selection, and I was assigned to estimate features' importance using 'Gini index' based on Random Forest Ensemble Algorithms, evidencing that we can delete all features for optimal running time.

Pablo

I am the member assigned to the SpikerBox. Early on I wrote a MATLAB pipeline to differentiate between up, down and blink eye movements. However it performed poorly and could not be extended to include left and right looks. When we merged teams my main focus became bulk collection of clean data for model training. I first filtered the data using an FFT filter but after research in the literature decided to implement the state-of-the-art Butterworth filter in MATLAB. I worked with Jamie to develop a program that could automatically detect signals, truncate around them and save the clean, filtered and normalised wavelets onto files that were ready to be opened in R for model training and/or label classification. I collected over 1000 eye signals for model training. Once this was done I established the R-MATLAB connection by training a knn model in R using Iris data. I worked with Stacey to advise on wave feature selection based on physics principles, with Chloe to efficiently connect the Tetris game to the overall product structure, with Ben to perform the required live tests for model evaluation and with Suri to draft the project pitch presentation and final presentation slides formatting.

Jamie

As the second member of the Physics team, I came into the project with limited programming experience, and due to COVID-19 I was unable to access the SpikerBox. This meant that the role I performed naturally became mainly organizational, but I made sure to help out with data processing, analysis and troubleshooting wherever possible. In order to ensure the project flowed smoothly, I took charge of the meeting notes and endeavoured to accurately record the progress and difficulties occurring throughout the interdisciplinary collaboration which helped to inform our next steps. Using the Gannt chart created by Ben, I continually checked our progress with our initial goals and help to make sure we were working in parallel as much as possible and understood our tasks. In terms of assessments, I drafted each of our presentations in line with the marking rubric and used my research of the field throughout the project to inform how to effectively pitch our product whilst also considering its limitations. On the technical side I assisted with developing the code to allow signals to be read efficiently in the live condition and created scripts to calculate MATLAB counterparts for each feature used in the R classifier.

Stacey

Initially, while our group was trying to figure out how to handle the streaming condition effectively, I worked with Chloe to make an event identification algorithm. We used this to identify events in filtered and unfiltered wave files, and created a training data set of filtered and unfiltered wave files. However, Pablo ended up creating an effective event classification algorithm in Matlab, advised us that the filtering process had minimal lag, and created a large filtered training data set so this work became redundant. With Ben we performed forward variable selection using the tsfeatures package to determine which tsfeatures would be best for the classifier. The classification accuracy we were achieving with tsfeatures was not satisfactory so Ben and I took it to the group to motivate discussion. After discussing the nature of the waves with the whole group, we brainstormed some key features that stood out to us about the wave signals. Then, from that discussion I calculated and performed in-sample testing of these new features which were the final features used in the model. As a result, the variable selection work process with Ben also became redundant. As the weeks progressed I would document my contribution in the meeting notes, and for presentations I wrote my script and created my presentation slides.

Suri

Before having the filtered data from Pablo, I tried to figure out which classifier model is the most robust by training and evaluate different models using the cross-validation method. The initial result gave that Random Forest model seems to be the best option with about 97% accuracy. Then, I attempted to put all the R code into Python using the py2 interface to Pandas, but it was unsuccessful due to the existing bugs in the rpy2 interface when being installed on Windows. For the final presentation, I wrote my script and created my presentation slides. I also took charge of formatting and submitting the final slides. In terms of the reproducible report, I am currently responsible for the aim, background and discussion of limitations and potential improvements in both disciplines. I am also writing the data collection process with the help of Physics students (Pablo and Jamie). At the end of the report, I will take charge of formatting the report as well.

Ben

As a data student, I was involved in the process of ensuring data collection quality, feature selection, evaluating the performance of different models and compiling the final results of the product. My initial role was to determine whether the initial data set already collected by Pablo would be sufficient for model creation, and as it was not, I requested further wave signals to be collected for training. Using the newly collected data, I worked with Stacey to determine which features would best boost the training performance of the model, without adding any lag time, using forward feature selection. In addition, I also worked with Stacey to implement some newly created features (outside of the tsfeatures package) that would lead to higher training performance scores of the models. With this list of features, I then trained and tested the performance of different models (KNN, SVM and Randomforest) to determine the accuracy scores and run times in order to select the best one for the demonstration. I also compiled the results from when we tested the classifiers live in game play with Pablo and his brother, in order to create my script and slides for the presentation.

[1] A. Rörsch, Humanities **3** (2014).

[2] R. Kemp, S. Parto, and R. Gibson, International Journal of Sustainable Development **8** (2005).

[3] R. Sinatra, P. Deville, M. Szell, D. Wang, and A.-L. Barabasi, Nature Physics **11**, 791 (2015).

[4] R. K. Pan, S. Sinha, K. Kaski, and J. Saramäki, Scientific Reports **2**, 10.1038/srep00551 (2012).

[5] L. Bao and K. Koenig, Disciplinary and Interdisciplinary Science Education Research **1**, 2 (2019).

[6] E. Etkina, A. Van Heuvelen, S. White-Brahmia, D. T. Brookes, M. Gentile, S. Murthy, D. Rosengrant, and A. Warren, Phys. Rev. ST Phys. Educ. Res. **2**, 020103 (2006).

[7] S. Gruner, Minds and Machines **23** (2012).

[8] A. Sheppard and J. Wolffsohn, BMJ Open Ophthalmology **3**, e000146 (2018).

[9] M. Rosenfield, Optometry in practice **17**, 1 (2016).

[10] B. Gopinath, L. Baur, J. Wang, L. Hardy, E. Teber, A. Kifley, T.-Y. Wong, and P. Mitchell, Arteriosclerosis, thrombosis, and vascular biology **31**, 1233 (2011).

[11] K. Daum, K. Clore, S. Simms, W. Vesely, D. Wilczek, B. Spittle, and G. Good, Optometry (St. Louis, Mo.) **75**, 33 (2004).

[12] C. Prescott, Internet users, uk - office for national statistics (2019), `https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/bulletins/internetusers/2019`.

[13] Digital eyestrain, the vision council. (2020), `https://www.thevisioncouncil.org/content/digital-eye-strain`.

[14] C. Coles-Brennan, A. Sulley, and G. Young, Clinical and Experimental Optometry **102**, 18 (2019), https://onlinelibrary.wiley.com/doi/pdf/10.1111/cxo.12798.

[15] J. Heo, H. Yoon, and K. Park, Sensors **17**, 1485 (2017).

[16] W. S. Wijesoma, Kang Say Wee, Ong Choon Wee, A. P. Balasuriya, Koh Tong San, and Kow Kay Soon, in *2005 IEEE International Conference on Robotics and Biomimetics - ROBIO* (2005) pp. 490–494.

[17] A. B. Usakli and S. Gurkan, IEEE Transactions on Instrumentation and Measurement **59**, 2099 (2010).

[18] A. Bulling, J. Ward, and G. Gellersen, Hans Troster, Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**, 741 (2011).

[19] J. Webster, *22. Webster, J. G. (ed.), Medical instrumentation: application and design, Fourth edition, John Wiley & Sons, Hoboken, NJ, 2010* (2010).

[20] A. Banerjee, S. Datta, M. Pal, A. Konar, D. Tibarewala, and R. Janarthanan, Procedia Technology **10**, 67–75 (2013).

[21] Backyard brain's spiker box (2020), `https://backyardbrains.com/products/spikerBox`.

[22] D. Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence* (1993) pp. 1–386.

[23] J. Schmidhuber, Neural Networks **61**, 85–117 (2015).

[24] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, Expert Systems with Applications **42**, 259–268 (2015).

[25] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, Expert Systems with Applications **42** (2014).

[26] R. Bunker and F. Thabtah, Applied Computing and Informatics **15** (2017).

[27] K. Kourou, T. Exarchos, K. Exarchos, M. Karamouzis, and D. Fotiadis, Computational and Structural Biotechnology Journal **13** (2014).

[28] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, Drug discovery today **23** (2018).

[29] Y. H. Kim, S. J. Yoo, Y. H. Gu, J. H. Lim, D. Han, and S. W. Baik, IERI Procedia **6**, 52 (2014), 2013 International Conference on Future Software Engineering and Multimedia Engineering (ICFM 2013).

[30] S. Kotsiantis, Informatica (Ljubljana) **31** (2007).

[31] C. Cortes, V. Vapnik, and L. Saitta, International Journal on Machine Learning , 1 (1995).

[32] T. Yiu, Understanding random forest (2019).

[33] T. Cover and P. Hart, IEEE Transactions on Information Theory **13**, 21 (1967).

[34] J. Bronzino, The biomedical engineering handbook, volume 2 (2000).

[35] B. B. Team, Email from BYB to Alessandro, Please find it at the end of Week 4's Lab Notes (2020).

[36] A. Tuniz, Backyard brains code, `https://github.com/PHYS3888/BackyardBrains` (2020).

[37] S. Mircic, Backyard brains read stream code, readsr.m, `https://github.com/BackyardBrains/SpikerShield/blob/master/Muscle/Documentation/Matlab/readSR.m` (2020).

[38] S. Butterworth, Experimental Wireless and the Wireless Engineer **7**, 536 (1930).

[39] S. Winder, in *Analog and Digital Filter Design (Second Edition)*, EDN Series for Design Engineers, edited by S. Winder (Newnes, Woburn, 2002) second edition ed., pp. 41 – 82.

[40] V. Roman, Supervised learning: Basics of classification and main algorithms (2019).

[41] R. Hyndman, Time series feature extraction [r package

tsfeatures version 1.0.1] (2019).
[42] C. Lee, Feature importance measures for tree models - part i (2019).
[43] S. Glen, Forward selection: Definition (2017).
[44] J. Brownlee, A gentle introduction to k-fold cross-validation (2019).
[45] Pygame about - wiki, `https://www.pygame.org/wiki/about`.
[46] D. Gur, davidgur/tetris.
[47] M. W. G. Dye, C. S. Green, and D. Bavelier, Increasing speed of processing with action video games (2009).
[48] S. Lertwisuttipaiboon, T. Pumpaibool, K. J. Neeser, and N. Kasetsuwan, Risk Management and Healthcare Policy **Volume 10**, 71–80 (2017).
[49] D. M. Laby, J. L. Davidson, L. J. Rosenbaum, C. Strasser, M. F. Mellman, A. L. Rosenbaum, and D. G. Kirschen, American Journal of Ophthalmology **122**, 476–485 (1996).
[50] A. N. Belkacem, N. Yoshimura, D. SHIN, H. Kambar, and Y. Koike, (2014).
[51] J. W. Cooley and J. W. Tukey, Math. Comput. **19**, 297 (1965).
[52] Analog continuous-time signals and systems, in *Signal Processing and Integrated Circuits* (John Wiley & Sons, Ltd, 2012) Chap. 2, pp. 9–37, `https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119942306.ch2`.

Appendix A: FFT Filter

Before we applied the Butterworth filter we had a simple FFT Filter.

When the equilibration time was over we continuously applied a low-pass FFT filter at 5 Hz. We begin by characterising the frequencies of the raw data, achieved by applying a Fast Fourier Transform ([51]). The result is known as the spectrum or power spectral density (PSD) of the original data and quantifies the signal's power at each frequency detected. We set to 0 the power at each frequency greater than 1 Hz in the PSD and apply an inverse FFT to the result. This returns the input stream to the time domain but with frequencies greater than 5 Hz filtered out.

However, the response of the FFT filter does not in general improve on the limitations of the Butterworth filter and actually has no conceptual attractive feature over the latter as it too has a relatively wide transition band but its pass band is dominated by the Gibbs phenomena at the boundary with the transition band which makes it curvy and not flat as desired ([52]).