The Review and Addition of Machine Learning Models in Brain Data

Submitted Apirl 2022, in partial fulfillment of
the conditions for the award of the degree **BSc Computer Science w/ AI.**

20180247

Supervised by Max Wilson

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature ________________

Date _____ / _____ / _____

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

Public access to this dissertation is restricted until: DD/MM/YYYY

Contents

Abstract:

Functional near-infrared spectroscopy (fNIRS)[1] is extremely useful system in brain imagery. Its portability compared to other systems and relative accuracy [1] in data gathered allows for wide adoption. However, there is a lack of standardization in the field, which causes issues with result reproducibility and comparison for datasets as well as models. To combat these issues led to the creation of the BenchNIRS[2] framework, on which this project is largely based on.

By exploring methods of labelling and trying to cover ground left by BenchNIRS unsupervised learning was added to the framework. This was done in various configurations as influence by the original exploration of labelling. Eventually resulting in splits of the dataset and run on, after automatically determining the number of clusters, k-means [3], then passed through mode filter proved to be the superior option. The method also surpassed the supervised learning [4] labelling of the tested dataset, 89.4% to 91.2%. Fallowing these results, dimensionality reduction also explored as allow k-means to work with higher dimensional data, with PCA[5] creating a principal component of 84.9%.

Acknowledgements:

This project would not have been possible without the aid and tutoring of both the supervisor, Max Wilson, and the Phd student that is developing the framework, Johann Benerradi. The weekly meetings and the message support for resources and insight for improvement were invaluable and the dissertations quality is hopefully reflective of this.

1- Introduction

1.1 Background

Since the discovery of near-infrared spectroscopy's (fNIRS) in 1992, it has become a key part of many studies involving brain data as a non-invasive neuroimaging tool. One major reason for this is due to the portability and cost for fNIRS devices, which allowed for more studies to use them and enabled the gathering of larger data samples than would be viable by using more traditional systems, such as magnetic resonance imaging (MRI) machines [6], which are expensive and requires a specific location for use.

While these two systems are comparable in the sense of both gathering brain data, their methods vary. MRI uses a magnetic field to vibrate water molecules to produces signals [6] to see the brain's structure, while fNIRS collects information uses infrared light to measure blood oxygenation levels [1] in different parts of the brain. fNIRS does this by the placement of sensors on the cranium which gather the data in real-time. Data is gained by placing the participant through various levels of mental exertion and correlating it to increased change in blood oxygenation and hemodynamic [7] changes that a person experiences in various parts of their cerebral cortex.

This focus on the cerebral cortex is explained by the lack of penetration capability compared to the MRI, due to near-infrared light diffusing quickly in neural tissue. However, an added benefit between the systems is that fNIRS is not as prone to problems in the data due to movement and has better temporal resolution [8] compared to other systems, including the electroencephalogram (EEG)[9]. The EEG has the added disadvantage of having a poor localization of responses [10] comparatively, which affects the accuracy of the data.

Nevertheless, the systems face the same issue of producing an unreasonably large amount of data for a person to be able to make meaningful conclusions. This is an area in which machine learning excels. Feeding the data into a machine learning model produces a useful output on which assertations can be made, such as predicting the level of mental strain the person experienced.

This reoccurring task of comparing the performance of different machine learning models in an efficient manner will be done by using an integrated framework that is currently in the final stages of development, BenchNIRS. This framework was designed for the purpose of reproducibility in the evaluation of models that consist of a wide range of AI techniques, including but not limited to RNN [11] (Recurrent Neural Networks) and ANN [12] (Artificial Neural Network). While many research papers on fNIRS exist, no universally designed frameworks have been developed. BenchNIRS, however, only uses supervised learning methods. To allow for further applications, this project focuses specifically on unsupervised learning methods [13], which may find connections in the data points that were unseen by the given labelling system and in turn can improve accuracy, even if unlikely due to the datasets labelling generally being optimised. The project also explores the labelling methods used and their possible improvement.

1.2 Aims

The aim of this project is to expand and improve BenchNIRS to allow for further use cases through the addition of models, specifically related to unsupervised learning and the improvement of its current labelling methods. This would allow researchers to use the framework to increase the duplicability/reliability of results to adhere to the scientific method. This accompanying research focusing the issue of labelling and its related systems as to improve general accuracy and proved alternative methods for use.

The key objectives of this project are:

1. Investigating of various machine learning models, with a particular focus on models used with respect to brain data exploitation or models that have potential application to be later applied.
2. Exploring labelling for task difficulty allocation in J. Bennerradi, H. Maior, et al [14], as to gain insight to optimum labelling techniques and layouts to be applied to the framework.
3. Using of gathered information on the task difficulty allocation, amend the difficulty allocation system to allow for a more reliable outcome, this outcome to later inform labelling strategy in the framework.
4. Adding unsupervised learning models in the BenchNIRS framework, by utilizing previously researched techniques, to process fNIRS data effectively, accuracy no worse than 5% compared to the given labelling, a novel challenge as supervised learning almost always out preforms unsupervised.
5. Automatic methods for determining the number of clusters for the user without interaction, the novelty comes from how its not a researchable technique to implement.
6. Testing and evaluating the newly added models using the BenchNIRS to gage their effectiveness, requiring all features to work seamlessly with the framework.

The use of the framework allows for increased comparability of methods, as normally they do not share data sets easily, with the metrics of their evaluation also varying. The framework allows for the easy exchange of datasets and a robust methodology that provides a benchmark for accurate and efficient comparison, with expansions to it increasing its versatility of application.

1.3 Motivation

The motivation of the project follows closely from its aims – to contribute to the development of the BenchNIRS framework and allow for the effective and reliable comparison of machine learning models by providing an accurate benchmark that allows research to progress, as previously results could not be easily duplicated. This is because previously there was no framework to allow for the comparison of machine learning models using fNIRS data, in terms of mental workload, making reproducibility and evaluation of results difficult or inaccurate. This resulted in a lack of consensus between various researchers in the field of machine learning related to workload brain data from fNIRS. By removing this hurdle, progress can increase faster. Essentially, improving the speed and ease of research and scientific advancement is my motivation.

1.4 Potential Use Cases

The uses cases for the framework are tied to the study of brain data created by the participation of individuals [30], in this case the level of mental effort a person exerts for tasks. The framework provides a basis for expanding into numerous areas of application, such as the early identification of

Alzheimer's disease in people. One study shows a “significant and positively correlated clinical scores” by using fNIRS in its decision making with $R \geq 0{,}4$ [15], the correlation coefficient, and $P<0.05$, the probability of the hypothesis is extremely high. The paper mentions the promise that fNIRS holds for use in medical applications, but also the crucial lack of a framework. This is only one of the innumerable implementations of a universal fNIRS framework. Essentially, many areas of research that apply fNIRS can use this framework to their advantage. Another example is use of the frameworks use in neuroscience studies of exercise, such as cycling, to measure the effect on brain activation [16]. This was then correlated to various elements, such as the working memory of the participant to see if there is a link, which was not the case with its $P > 0.05$, meaning no distinctive correlation found and the hypothesis is most likely wrong.

With the addition of unsupervised learning to the framework, there are more opportunities for application, with there being two situations in which it can be advantageous to use. The first, as mentioned previously, is finding patterns in the data that can then be used to group data points, in turn likely reducing the dimensionality [17] of the data and increasing the accuracy of the machine learning model [18]. The reduction in dimensionality comes with some key benefits, as it reduces the complexity of the data, aiding both in the storage of the data as well as the time complexity, due to a reduction in the curse of dimensionality [19], for processing. Additionally, the data is more interpretable by the user as, for instance, it is easier to make conclusions on 2D data, due to simpler visualisation, than 3D data.

The second opportunity is due to the lack of requirement for labels. The framework can be applied to datasets that do not have labels clearly defined, which could also become a step that the researcher now does not have to worry about, speeding up the process and allowing for a wider application of various datasets.

Unsupervised learning, however, does not come without its draw backs. It costs more processing time, as labels must be assigned to a dataset through a method rather than having them pre-assigned and ready to use. Another issue is the difficulty in assessing the effectiveness of methods, as there is no given label against which to compare the output. This obstacle was overcome, however, by using unsupervised learning methods on an already labelled dataset, which allowed the comparison and review of the method. Another key issue is that unsupervised learning methods will in many cases have a lower accuracy then supervised learning methods. This is partly due to some element of randomness in their generation[4], such as seeding [20], which can reach situations of local optimum rather than a global optimum, which a carefully constructed supervised learning method may achieve in machine learning.

2- Related Work

Machine learning aims at improving prediction, including improving existing models and creating new ones, such as GANs [21] (Generative Adversarial Networks) that were created in 2014. In recent years, a greater focus has been placed on machine learning in respect to brain data in multiple forms, such as different data gathering methods like MRIs and fNIRS and the development of supportive frameworks.

2.1 – Frameworks

Other frameworks have been developed for the review of machine learning models in relation to brain data, such as the identification of early Alzheimer’s disease in people, as mentioned in Section 1.4 above. With E. Moradi, A. Pepe, et al. [22], taking it a step further by providing a framework for machine learning models to be developed and tested, much like the proposed BenchNIRS. This study resulted in a cross-validation [22] AUC (Area Under Curve) of 0.902, resulting in a P < 0.05, in the prediction of Alzheimer's disease three years before diagnosis, which is an excellent prediction score. This shows that when a standard is created against which models can be developed great strides can be made in the respective fields, even in contrast to the already strong score seen in the Alzheimer's method used in section 1.4.

This positive outcome was achieved using different approaches in combination. The use of biomarkers of Alzheimer's [23] disease as well as age with cognitive measurement, which shows two points. First, a combined approach to the classification of data is used to consider all elements that may affect the result. Second, the measure of a person's cognitive abilities, which can be partially inferred by how difficult they found the task, as done in the current project framework, is an important metric. This applied in many different areas of research, demonstrating both the need to record it and the necessary value in its accuracy.

The limitation of this framework developed by E. Moradi, A. Pepe, et al., is its highly specialized capabilities. It is only designed to be used to identify Alzheimer's disease, meaning that even if the framework functioned perfectly, it is still very limited. The BenchNIRS framework, due to its increased customizability and range of applications, far exceeds the potential of the Alzheimer's identification paper and may support other use cases. Additionally, BenchNIRS is open to download by researchers and, because of its more universal nature, people only need to learn one framework to understand the process and apply it in many areas of research. The source code is easily viewable and can be tailored to the exploration criteria.

A framework for machine learning is not limited to brain data, as seen in the framework developed by L. Ward, A. Agrawal, et al., which predicts the properties of inorganic materials [24]. This framework breaks what would be expected to change in the durability, weight, tensile strength, etc., of materials. This includes the review of 145 different attributes that would change the physical properties of the material, which can be separated into 4 distinct categories: 1-Stoichiometric attributes, 2- Elemental property statistic, 3-Electronic structure attributes and 4-Ionic compound attributes. This is important due to labelling of the data, with each of these categories making up an important part of the expected properties. For example, the Stoichiometric attributes depend solely on the number of elements present in the compound. By providing the structure using the list of attributes and how they further break down into easily identifiable categories, these eases adding new materials and machine learning models.

This approach enabled the project to achieve 90.1% accuracy on the identification of metallic glass when performing a 10-fold cross-validation [27]. The difference between current project and Ward, A. Agrawal, et al., is a clear example of a different area of focus, with it not covering a fNIRS by its nature. Additionally, adding unsupervised learning to the fNIRS method removes the necessity of categorising data, which is fundamental to how the other framework function. This shows not only a difference but a benefit, as categorisation can be both time consuming and possibly lead to missed patterns in the data which would have resulted in a higher accuracy from the machine learning model.

2.2 – Machine Learning on fNIRS Data

Machine-learning models, as previously discussed, can be compared outside of using frameworks simply by giving them the same data sets to work off. This has the advantage of quickly producing a measurable outcome, as there is no need for a larger designed framework. In addition to the use of fNIRS to predict Alzheimer's disease, as discussed in Section 1.4 and 2.1, R. Rojas, X. Huang and K. Ou used [25] fNIRS to predict the amount of pain by using hot and cold temperatures. The Gaussian SVM [26] (Support Vector Machine) produced an output of 94.17% in terms of the machine learning algorithm being able to predict if a person is in pain or not. This is interesting, as in the previous examples a 10 cross-fold validation produced the strongest outcome, while here an SVM produced the strongest outcome. This shows that machine learning is not a "one size fit all" proposition, with fNIRS data possibly having a SVM preference regarding the accuracy of prediction.

Interestingly, while R. Rojas, X. Huang and K. Ou, do not formalise a framework and give researchers the bases to build their programs to be comparable to the models, they discuss 25 criteria separated into 4 types of pain. This again demonstrates the need for effective labelling in machine learning. However, with no real discussion of benchmarking or how comparison would work in the future, the study is a good example of the middle ground and the distinction between framework enabled machine learning and simply machine learning research.

The next example of machine learning on fNIRS data looks at mental exertion during a balloon game. J. Bennerradi, H. Maior, et al., used the high amount of mental exertion for the harder scenarios to create a correlation, using the established concept of a higher mental workload from large amounts of information given to a person [7]. The program attempts to predict how difficult a person would find the task by classification, producing a 2-class classification accuracy of 81.54% using a Generalised SVM between its predicted and real values. It is important to note that SVM again performed best, which supports the previous assumption about its relationship with fNIRS. While this may be sampling bias, there is an explanation for this outcome. Due to the also low number of participants in both examples and the high amount of data and inputs due to working with the brain, a curse of dimensionality problem can arise, with which SVMs are characteristically good at dealing [26].

The lack of a unified benchmark for testing and creation that would allow for the easy expansion and addition of models has led to the J. Bennerradi, H. Maior, et al. study's admission that "little work has evaluated the different machine learning approaches that will work best for the task." This is a problem that would be solvable through the work being placed in a framework that allows further testing, as is being done in the current project. This could produce a value even higher than the admittedly strong outcome reported. The J. Bennerradi, H. Maior, et al. study is the origin of the data explored below as its application in the wider developed BenchNRIS framework.

3- Data - Initial Investigation

This project is predominately research-based, meaning that a lot of information and analysis, including both the review of research papers and active coding, is required to draw meaningful conclusions. Discussed below is how the data was gathered and manipulated by J. Bennerradi, H. Maior, et al. for the initial part of the investigation into the foremost machine learning models and their corresponding labelling to explore techniques of organisation that can be applied.

3.1- Data Collection – Initial Investigation

The data for J. Bennerradi, H. Maior, et al. was gathered from the sensors on the heads of eleven participants, 6 men and 5 women with the mean age of 29 years old, an SD = 6.8 with an age range of 19-42. The data was collected with the informed consent of each participant, using the configuration of the sensor layout seen below.

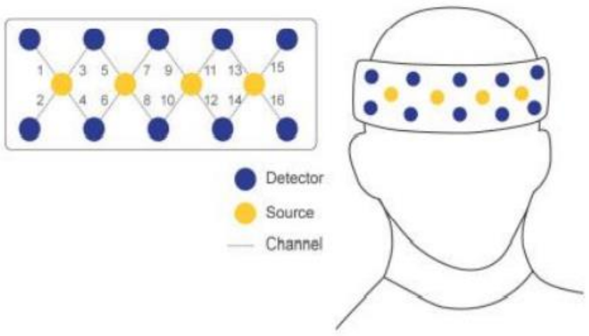


Figure 1: Biopac fNIRS sensor layout [14]

The headband used was from Biopac Systems Inc [14] is a 16-channel transducer, as seen in Figure 1, which allows for continuous NIRS (Near Infrared Spectroscopy). Additionally, the headband integrates 4 infrared emitters which work in the range 700- 900nm and 10 infrared detectors with a data gathering rate of 2 Hz.

The participants then played a training version of the stimulus task, so that they had a concept of how the game is played and for the researchers to gain a baseline of the participant. The participants were then asked to rate the difficulty every 45 seconds as they did the stimulus task by using the ISA [14],[29], (Instantaneous Self-Assessment) Technique based upon a 5- point scale, with 1 being the lowest to 5 being the highest.

The stimulus task was designed to cause a mental demand – aiming a joystick at the correct balloons and shooting them by using a yellow button with the objective of shooting the balloons before they reached the yellow line. The yellow line moved up the screen if all targets were destroyed and down the screen if a target was missed. The intensity of the tasks changed throughout the run both going up and subsequently down again.

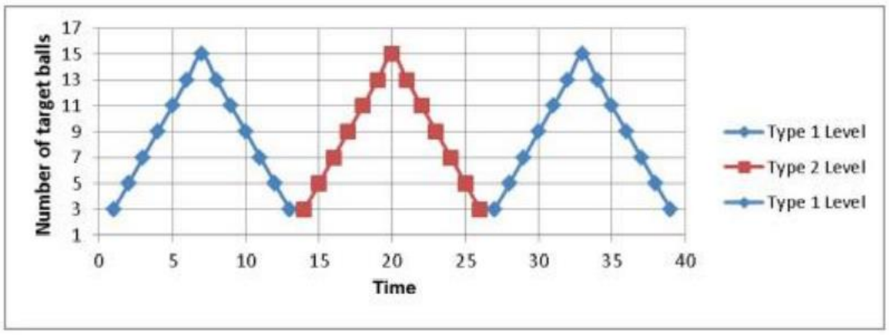


Figure 2: Variation in Task Demand [14]

In Figure 2, Type 1 and Type 2 represent the two different modes of the game. The Type 1 task was to only shoot at red balls and the Type 2 task was to only shoot at balls with odd numbers, no matter the colour. Type 2 involved higher mental strain due to deciding between the number type rather than just the colour. Each test set took place over a period of 40 seconds.

Not all data gathered from the 11 participants was usable from J. Bennerradi, H. Maior, et al., because the shape of the headband did not correctly fit all the test subjects. This led to corrupted and incomplete data for participants 1 and 10, who were removed from the data set [28]. It is a general practice in data science to exclude unusable data and is common with fNIRS studies due to "physiological differences" as shown in other papers [30].

3.2 Data Manipulation– Initial Investigation

The initial step is to use MBLL [31] (Modified Beer-Lambert Law) to transform the gathered data from a raw input into de/oxygenated haemoglobin levels, which is simply a modifed version of Beer-Lambert Law that is specialised for the use case of the brain.

<u>Beer-Lambert Law</u> - $A = \varepsilon b C$ A: Absorbance b: Length of light path

ε: Molar absorptivity C: Concentration

MBLL works by how "the change in light attenuation is proportional to the changes in the concentrations of tissue chromophores, mainly oxy and deoxy haemoglobin." [32]. Essentially, the greater the amount of light found by the sensor, the greater the presence of haemoglobin. As can be seen from this quotation, however, it also reflects changes in concentration of chromophores, which include but are not limited to haemoglobin, such as the presence cytochrome oxidase [31][32] which can get picked up at the same wavelength and must be accounted for.

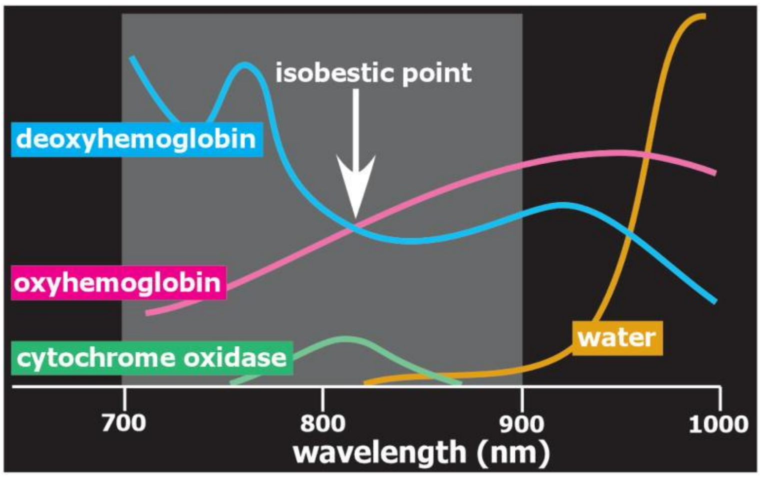


Figure 3: fNIRS Wavelength Reception [31]

fNIRS differentiates deoxyhaemoglobin from oxyhaemoglobin, as the system varies its frequency to detect these changes, with the isosbestic point in Figure 3 being able to measure the total haemoglobin in the system and the grey area showing the range of the sensor in the experiment. This allows for isolation of cytochrome oxidase and its removal from the data.

Additionally, noise can be introduced in the data by other factors. The movement of the participant while playing the game [32], even with fNIRS handling it better than an MRI, crosstalk between sensors and their respective sources [32], and even the heartbeat of the participant can all cause issues with the data gathered. This is due to movement repositioning the sensors during recording, lifting them from the scalp causing spikes

The data is pre-processed through multiple stages and transformations before it should be used in the exploration of which machine learning models work best. To address the issue of noise explained above, filtering algorithms were then applied to the output to get rid of residual noise and physical artefacts. This likely involved the use Kalman [33] and Least Mean Square Error [34] filters but was not specified.

Then the second step J. Bennerradi, H. Maior, et al is to use CBSI [35] (Correlation Based Signal Improvement), to improve the signal quality gathered through fNIRS. CBSI has relevance to fNIRS, as it was designed for this purpose. It improves the signal quality by using the negative correlation between oxygenated and deoxygenated haemoglobin [35]. If the subject moves, then they become more positively correlated. By using CBSI, the right regions of brain activity can be identified by carefully observing the correlation percentage between oxygenated and deoxygenated haemoglobin and removing the movement artefacts.

The data is then normalised due to the multiple participants, to account for day-to-day and individual variations of the sample group, as seen with the equation,

$$X_{\text{normalized}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad \text{x: Dataset}$$

Then the gathered ISA scores of the participants and the output from CBSI and its various channels are correlated to the mean of the ISA scores, then the data is normalized to a range of 0 to 1. This is due to the day-to-day variance in how people perform on the tasks. By creating a tighter range, the correlation between the ISA and the CBSI data is more prominent.

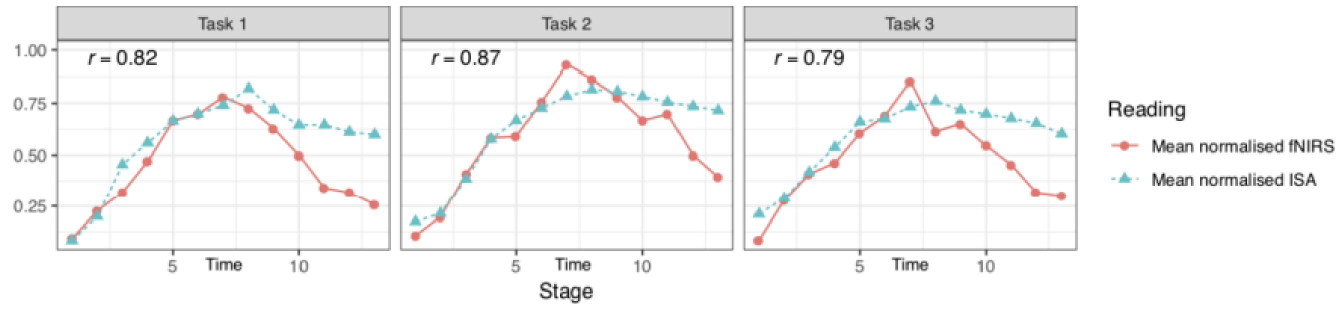
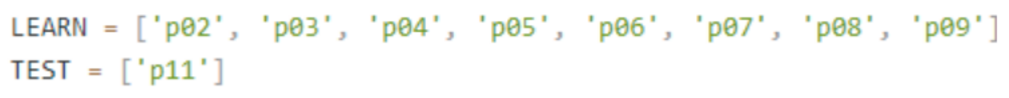


Figure 4: CBSI Normalized Output against Normalized ISA [14]

Figure 4 shows the close and positive correlation of the mean normalized fNIRS and normalised ISA. The average R of 0.83 shows an extremely positive correlation, with any value above 0.7 showing a high correlation [36]. The outcome validates both data sets, as this is what would be expected if the data of both were gathered correctly. This allowed the data to be used without concerns about their authenticity and allowed the study to have more legitimacy in its findings. Both measure mental workload, with one using the participant input and the other using the sensors.

4- Implementation & Analysis – Initial Investigation

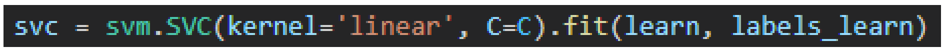
The refined data is then used in conjunction with various machine learning models to be able to predict the mental exertion of the participant. Then splitting the ISA scores into training and testing sets, as seen in Figure 5, with p02-p09 to be used with the learning aspect of the machine learning program and p11 to be used for the testing case.



```
LEARN = ['p02', 'p03', 'p04', 'p05', 'p06', 'p07', 'p08', 'p09']
TEST = ['p11']
```

Figure 5: Testing and Learning Sets

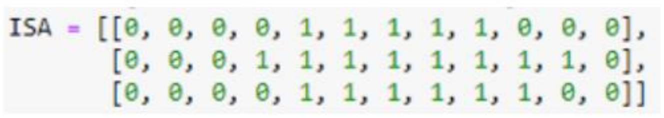
These data sets are then trained with the Generalised SVM,



```
svc = svm.SVC(kernel='linear', C=C).fit(learn, labels_learn)
```

Figure 6: Generalised SVM

By using the average of the participants ISA being used to gage the success of the program.



```
ISA = [[0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0],
       [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0],
       [0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0]]
```

Figure 7: ISA Averages for Task 1-3

This produced an accuracy, as previously mentioned, of 81.54%. The rationale for how the data is organized can be seen in Figure 2. Each row represents a task, in the following order: Task 1, Task 2, Task 1. The presence of task 2 can be easily seen in the relatively greater level of difficulty, with Task 1 having an average difficulty of 0.46 and Task 2 of 0.58. As previously described in Section 3.1, this outcome was to be expected. The SVM was chosen for investigation, because of its higher accuracy and that generalised models work well with fNIRS data, which is backed up by the related works reviewed earlier in this paper.

J. Bennerradi, H. Maior et al., produce the ISA presented in Figure 7 by classifying the different ISA averages into the respective values, in this case into 0 and 1.

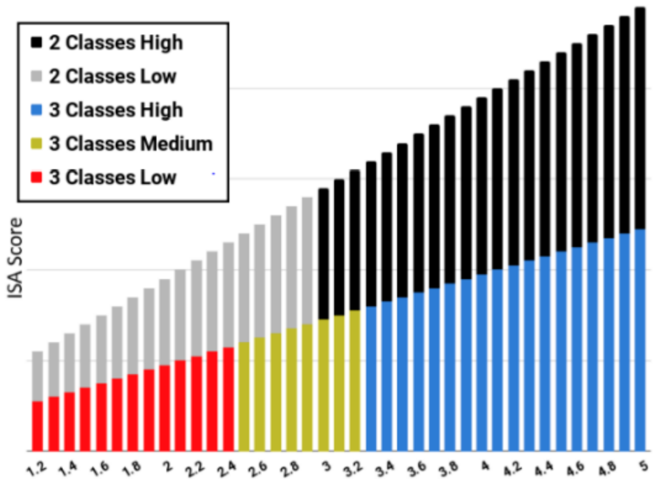
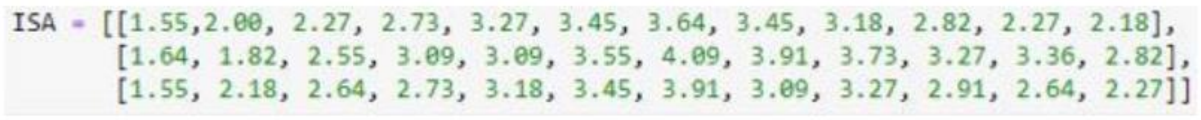


Figure 8: ISA classification split for 2 and 3 classes

Figure 8 shows that the data was organised with purpose. Importantly, how the data is used and structured arguably has the greatest effect in producing a better outcome, which has large implications for the use of machine learning models. This led to an investigation into what is the best version to use for the ISA to achieve the objective of improving the task difficulty allocation, so that the findings can be later applied to the wider framework.

The first approach was to take a step back from the 0 and 1 format and look at the average output of the participants,

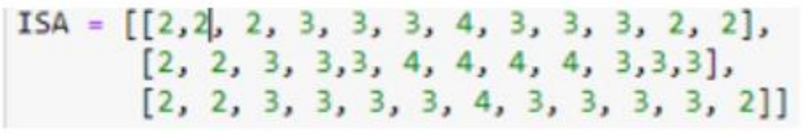


```
ISA = [[1.55,2.00, 2.27, 2.73, 3.27, 3.45, 3.64, 3.45, 3.18, 2.82, 2.27, 2.18],
       [1.64, 1.82, 2.55, 3.09, 3.09, 3.55, 4.09, 3.91, 3.73, 3.27, 3.36, 2.82],
       [1.55, 2.18, 2.64, 2.73, 3.18, 3.45, 3.91, 3.09, 3.27, 2.91, 2.64, 2.27]]
```

Figure 9: ISA Averages Without 1-0 Correction for Tasks 1-3

The results were quickly tested with the SVM, which led to a terrible outcome of 2.78% in the accuracy of prediction. This is to be expected, as the model does not handle continuous [37] data as its classification, as compared with regression [37] which works with continuous datasets. The only reason for any result is the chance that one of the values is a whole number, 2.00.

To rectify this situation, all the averages are rounded to their nearest whole number and the SVM model was run once again,



```
ISA = [[2,2, 2, 3, 3, 3, 4, 3, 3, 3, 2, 2],
       [2, 2, 3, 3,3, 4, 4, 4, 4, 3,3,3],
       [2, 2, 3, 3, 3, 3, 4, 3, 3, 3, 3, 2]]
```

Figure 10: ISA Averages for Tasks 1-3 into Whole Numbers

This gives a lower output than the first example, but with a much better outcome. With rounding to the nearest whole number, SVM produces an accuracy of 70.83%. As the program is attempting to predict 3 classes rather than just 2, and the chance for the program to get it right is reduced. This is due both to a lower random chance and more options for which the program to account. which partially explains the reduced accuracy compared to the two-class output. For this reason, the results should be compared to the output of the 3-class outcome in the report, causing a visible improvement compared to the reported output of 64.81%. Even though the investigation is not complete, the structuring of data is evidently important, as reflected in the successful change – an increase of 6% in the accuracy of the SVM classification prediction. Thus, instead of using the process seen in Figure 8, J. Bennerradi, H. Maior, et al., could achieve a better outcome using the method above.

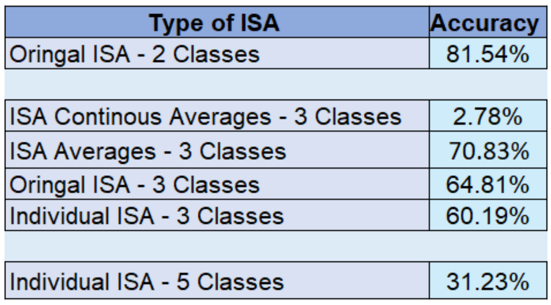
The next step is to adopt a completely different strategy for the organization of data. Instead of averaging all the values and then performing changes by using the strategy suggested in Figure 8, data on all the participants is taken as individuals and fed into the machine learning algorithm. This required a few things that the previous alterations did not. This includes a more radical change to the code present in the machine learning algorithm to allow it to go through all the expected participants, as well as a Python script to run and get all the data out of an Excel file, due to organization as there was a lot more data to handle. Finally, more time is needed, because of how much more data for which the SVM program had to account.

This did not result in an improved output, however, with the SVM producing a 31.23% prediction percentage when using the 1-5 difficulty ratio. This is most likely due to two reasons.

First, averaging out the values creates a higher chance of making a program that would result in the correct conclusion, as it must apply to the whole ISA set, effectively normalising the data. Second, trying to predict for 5 classes rather than 2 will result in a lower prediction percentage, as can be seen with the highest prediction percentage in the paper for 2 classes being 81.54% and the highest for 3 being 64.81%. Again, this is to be expected as the program must predict 3 different values rather than just 2, increasing the chance incorrectly identifying the value.

Then approach adopted is to go through each participant’s output and make it align to three classes. This is achieved by adjusting each ISA score based on a comparison of their highest and lowest points. For instance, some participants only answered in the range of 1-3, as illustrated by the Task 1 responses of Participant 4 [1,2,2,2,3,3,2,2,2,3,1,1,1]), as compared with Participant 6 [2,3,3,4,4,5,5,4,4,3,3,3,2], who answered in the range of 2-5 for the same task. While it is expected that certain people would find the challenge harder than others, this range suggests a different problem of how people would individually view the difficult of the task, even though they may not find it more difficult according to the fNIRS data. For this reason, each participant’s responses were examined individually and aligned with the 1-3 range. This led to a much better output then the normal use of the 1-5 range discussed earlier, with the SVM producing a 60.19% predictive score.

The investigation results can be viewed in table below,



Type of ISA	Accuracy
Oringal ISA - 2 Classes	81.54%
ISA Continous Averages - 3 Classes	2.78%
ISA Averages - 3 Classes	70.83%
Oringal ISA - 3 Classes	64.81%
Individual ISA - 3 Classes	60.19%
Individual ISA - 5 Classes	31.23%

Figure 11: Data Transformation Investigation Results

The investigation was fruitful in providing a basis for improvements, as seen in Figure 11, with the ISA Averages for 3 classes performing better than the scheme laid out in the research paper. Even though the results left further room for improvement, they contributed to a general understanding of why some transformations of data are better than others in particular situations. Based upon these positive outcomes, further exploration should not be discounted. This could involve a more multifaceted approach with increased inputs, such as heart rate, which would have the benefit of increasing the accuracy by allowing other characteristics associated with a heightened mental exertion to be exploited.

This initial investigation highlighted the importance that once again can be placed on labelling. With the investigated ISA structures, the importance of data structuring in how categories are labelled again becomes relevant when discussing the framework. However, the limitations that exist within this system, primarily around how this was done without the use of a framework, led to the general difficulty of adding code to non-modular designed systems. This, in turn, led to complexities when additions were placed within the code base that was not designed for it.

Additionally, while SVMs were found to be the best model, they also have their limitations, especially in not handling larger datasets very well, which as can be seen in figure 10, and is not the case here, and when using datasets that have noise from overlapping target classes. While these were not issues for the present study, they become more evident when working with larger datasets.

5- Framework Investigation

The second phase of the investigation revolved around the framework, using what was learnt in the primary phase. This included the initial step of reviewing the framework and understanding its components.

5.1: Framework Review

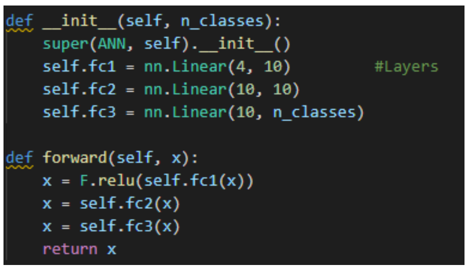
The framework currently being built can be seen as a continuation of the paper by J. Bennerradi, H. Maior, et al., and the work of J. Bennerradi, is the main designer of the framework. The present paper seeks to address the issues faced with the last system and build upon the knowledge gained from it.

An issue that was addressed was the difficulty to add code to the codebase, which was resolved by creating the framework in modular form which allowed additions to be placed within the code without an entire rework. This fact later allowed for the addition of unsupervised learning to work with the other components, provideded they were configured to treat the data in the same way. The use of "deep_learn" and "machine_learn," as well as the methods that specify their functionality of CNN, LSTM [37] and ANN, give the output of unsupervised learning in the form of labelling, a much greater range of application.

This range is seen again in how the data can be further manipulated using the especially useful features of mean, standard deviation, or slope of the linear regression, across the time axis of the given data. An example would be the use of mean averages the dataset to remove outliers. Moreover, the easy addition of features, if the situation calls for it, is handled by a separate function, "_extract_features." Along with the deep_learn automatically tuning the hyperparameters [39] using cross-fold validation, seen in Appendix A, C and not for machine learning in Appendix B, as it is required for deep learning [40]. Implemented due to deep learning producing a stronger output in terms of categorisation accuracy [40]. These benefits of using a framework and the value it can add quickly become clear, as these components do not have to be built from scratch for each new investigation. Another way the framework standardises its output is by placing the data be on the same frequency (10hz), rather than having the datasets vary. This would cause issues with the framework because it is designed for 10hz data and data comparison. For example, the distribution of the data when graphed against each other would never line-up in a situation where you have 25hz data and 10hz data, as the gap between each datapoint compared would be 2.5 times greater than it should be, making the recognition of key differences and effectiveness more difficult.

There are nevertheless limitations to the framework. For example, the organization of the data must conform to the framework. If there is an irregular dataset that cannot be manipulated to work within the framework, it cannot be used. This would occur if the dataset were too small, as cross-fold validation requires multiple training sets [27]. Another limitation is that the framework is simply built for multi-dimensional fNIRS data, as seen in the outputs and data manipulation of the deep_learn function, such as how the data is split to always group the same example set, seen in Appendix A. While this limits its use case, by not overextending its application, it can increase its effectiveness and achieve its goal of high accuracy on brain data.

A new model was added to the framework using a three layered artificial neural network (ANN).



```
def __init__(self, n_classes):
    super(ANN, self).__init__()
    self.fc1 = nn.Linear(4, 10)         #Layers
    self.fc2 = nn.Linear(10, 10)
    self.fc3 = nn.Linear(10, n_classes)

def forward(self, x):
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    x = self.fc3(x)
    return x
```

Figure 12: Added ANN Model

Figure 12 was implemented for multiple reasons. As an SVM was not used because of it is ineffective on large data sizes, as mentioned previously. The smallest data size of 300 examples, with each having 100 time points and each time point having 4 dimensions to it. For perspective, the ISA rating discussed above had 3 examples with 12 time points having 1 dimension, making this dataset roughly 4615% larger, explaining the difference. Additionally, ANNs can understand non-linear complex relationships [14] within the data, which is beneficial for the multi-dimensionality of the dataset. ANNs are also extremely flexibly [14] with the input variables and do not impose any restrictions on the data unlike other techniques, such as SVMs [14]. Along with the fact that ANN's are good at generalising [14], the lack of restrictions makes this the perfect model to implement, which gives more credibility to the results being unbiased.

Another reason for using ANNs was to distribute risk, as was done during the initial investigation. This was achieved by doing the investigation in stages, rather than trying to accomplish the end goal immediately. This addition to the framework gave a foundation from which to build unsupervised learning and gave a product that can already be shown. The resulting in output peaking at 89.4% accuracy. This lower value is not unexpected for a test system that has only 3 layers to train within its neural net and the model was not allowed to train to its full potential as would be possible in the system normally, this was done by editing the deep_learn function to stop after several epochs to reduce the run time for testing purposes. In addition, the low accuracy of this model is not the main concern of this investigation, as the question of labelling and the difference between the unsupervised and supervised labelling is, meaning that only the difference in accuracy is important.

The last reason for implementing a new model in the framework was to understand how the framework fit together in terms of organisation and dependencies, which helped pave the way for the implementation within the framework in the later phase of the dissertation.

5.2: Framework Data

Various datasets came working with the framework. Many were of a larger size, so the program could be better trained. Due to the nature of cross-validation, however, more data means greater time complexity [27], which does not increase linearly O(n) but rather exponentially O(n^2). Consequently, only the smallest dataset within the framework was realistic to run for the investigation, as the study was being carried out on a laptop without the resources to handle the needed processing quantity. This however is not to understate the complexity of this dataset, with 30,000 individual 4-dimensional data points. Even though this is the smallest dataset, it is still large enough for the current inquiry. This dataset required requesting access from authors of the original paper [41], who provided a user specific drop link to pass the necessary information.

As expected, data collection for this dataset closely matched that of the initial investigation, such as the use of Beer-Lambert law. Data collection differed slightly with the use of the Oxymon Mark III [41], which again uses the variation in the wavelength at a frequency of 765 and 865nm to measure the concentration of oxygenated and deoxygenated blood. The reason for using these two frequencies can be seen in Figure 3 (fNIRS Wavelength Reception) above. The set up for the data gathering was four transmitters and four receivers, with each receiver having two sensors spaced at 3.5cm, meaning there are a total of 8 channels in the systems output.

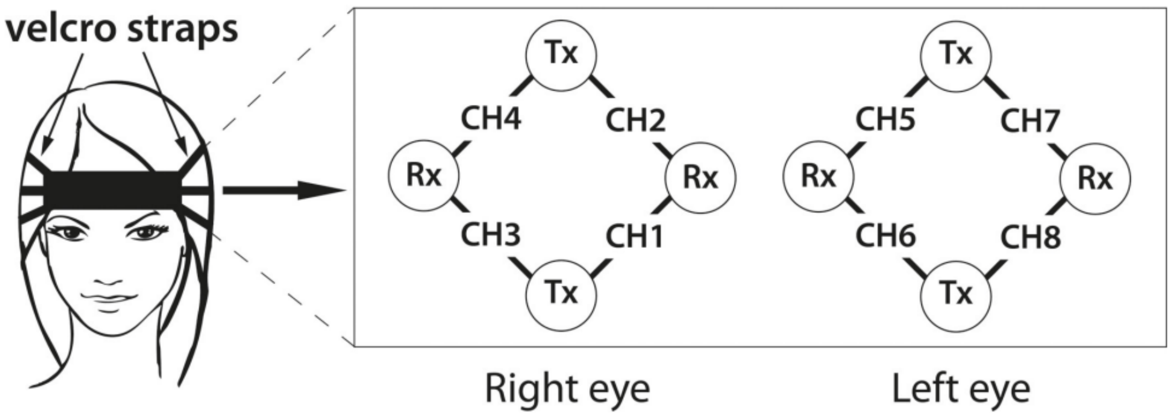


Figure 12: Data Gathering Framework

As Figure 12 shows, C. Herff, D. Heger, et al. used the same type of setup as J. Bennerradi, H. Maior, et al., to collect the dataset in the form of the ISA, as seen with Figure 1. After the equipment was attached to the participant (Figure 12), they were placed through 10 trials on each task. The tasks consisted of presenting a new letter every two seconds, with the letter being shown for 0.5 seconds and the interval of 1.5 seconds being blank space. Each trial consisted of 22 letters with a cross displaying afterwards for 15 seconds and an additional 10 seconds added after 150 second break when they finished with the first half of trials. This gives the participant sometimes to reach a baseline [41] in terms of oxygenated and deoxygenated blood. This is important as otherwise the participant trials would not be isolated from each other, and the data could face cross interference, with these rest periods excluded from the dataset as they were "strongly influenced by the previous hemo-dynamic responses." [41].

This entire process used by C. Herff, D. Heger, et al. can be visualised using this diagram,

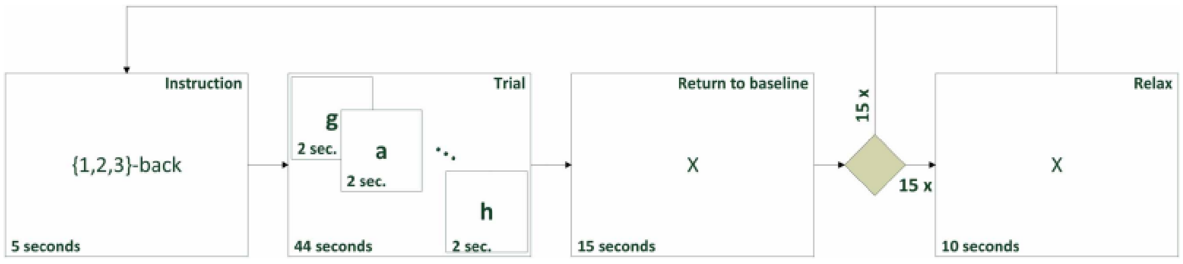
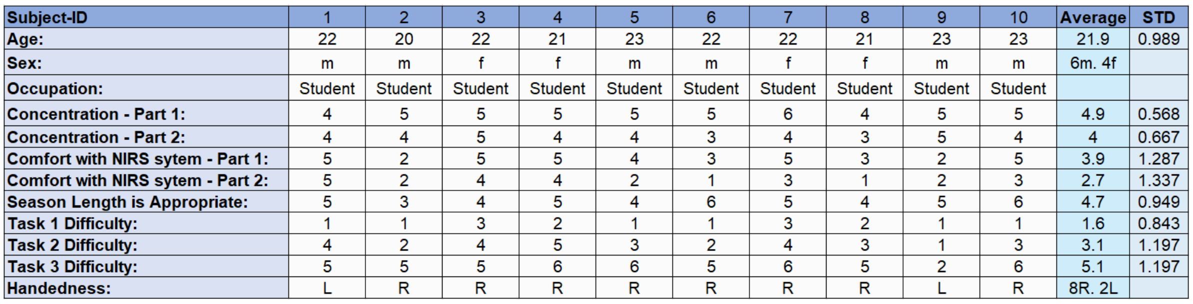


Figure 13: Process of Data Collection

Afterwards the participants were then asked various questions,



Subject-ID	1	2	3	4	5	6	7	8	9	10	Average	STD
Age:	22	20	22	21	23	22	22	21	23	23	21.9	0.989
Sex:	m	m	f	f	m	m	f	f	m	m	6m. 4f	
Occupation:	Student	Student	Student	Student	Student	Student	Student	Student	Student	Student		
Concentration - Part 1:	4	5	5	5	5	5	6	4	5	5	4.9	0.568
Concentration - Part 2:	4	4	5	4	4	3	4	3	5	4	4	0.667
Comfort with NIRS sytem - Part 1:	5	2	5	5	4	3	5	3	2	5	3.9	1.287
Comfort with NIRS sytem - Part 2:	5	2	4	4	2	1	3	1	2	3	2.7	1.337
Season Length is Appropriate:	5	3	4	5	4	6	5	4	5	6	4.7	0.949
Task 1 Difficulty:	1	1	3	2	1	1	3	2	1	1	1.6	0.843
Task 2 Difficulty:	4	2	4	5	3	2	4	3	1	3	3.1	1.197
Task 3 Difficulty:	5	5	5	6	6	5	6	5	2	6	5.1	1.197
Handedness:	L	R	R	R	R	R	R	R	L	R	8R. 2L	

Figure 14: Participant Survey

As there were other considerations when the data was being gathered, such as accounting for all participants having normal vision, a normal distribution of left to right-handed people of 2 to 8 and making sure that the participants had not previously taken part in similar experiments, which would have meant that they had training. The average difficulty in the tasks also give backing to how task 3 is more difficult than task 1 and the difference in difficulty between them being clear, this is important as to make sure.

Additionally, they were asked about how they found the difficulty of the tasks and their corresponding concentration, this was to disregard participants which did not strain themselves would produce problematic data. The participants were also asked how comfortable they found the head gear for fNIRS, as to see if the comfort of the participants effected the results together with the standard deviation (STD), this turned out to have minimal affect and didn't lead to disregarded data.

All these steps, from the waiting periods to the participant considerations above and in Figure 14 meant that the data produced in the C. Herff, D. Heger, et al. study was very robust, which was the reason it was chosen for the framework in the first place and why it was additionally chosen for this project.

C. Herff, D. Heger, et al. data manipulated the data to produce a usable dataset, much like in the J. Bennerradi, H. Maior, et al., study. This involved the removal of artifacts such as hearts beats, respiration, and slow wave (Mayer Waves) [42], because of their effects on the sensors, as mentioned previously. Interestingly, C. Herff, D. Heger, et al. used other techniques in the cleaning of data, and the paper goes into more detail regarding the overall process. The process started with the use of Principle Components Analysis (PCA) to divide the "NIRS dataset into linearly uncorrelated components," separated by their contribution to the variation of the data. The assumption here is that artifacts will have the largest impact on the varication of the data, which means if the primary component, the one of greatest effect on variation, is taken out of the equation, the issue of artifacts can be vastly reduced. What this means is that the matrix is transformed from,

$Y^TY = U\Omega U^T$ U: Orthogonal matrix of spatial eigenvectors

Ω: Diagonal matrix of the associated eigenvalues

Y: Dataset

This allows for the removal of the first r eigenvectors from the data, creating,

$Y_{corrected} = Y(I - UA_rU^T)$ A_r: A matrix with a positive value given to data to be removed due to their being the first r eigenvectors

Resulting in the creation of the $Y_{corrected}$ dataset.

C. Herff, D. Heger, et al. next used the process of Kalman filtering, mentioned previously, which allows use of the still noisy data to estimate the state of the underlying signal with extremely high accuracy. This can be seen as a two-step system. The first step is to use previous state, as it uses preceding information, to make a prediction of the future state and its uncertainty,

$x_k = Ax_{k-1} + \gamma_k$ A: State transitional matrix

γ_k: System noise

x: State

The transitional method, based on the Yule-Walker method [43], uses past information to make predictions about the future state. This prediction is then compared to what the data experienced, producing a function of the true state,

$Z_k = Hx_k + \delta_k$ H: Observation model, that "maps the true state-space onto the measurement space"

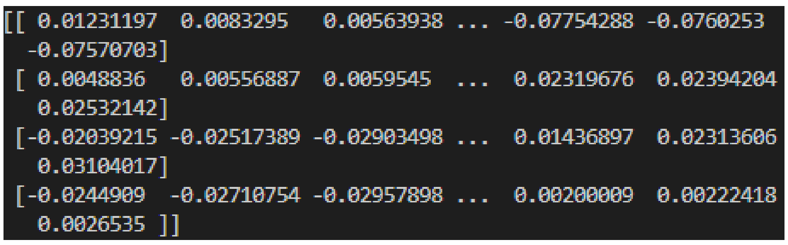
δ_k: Measurement noise

Then the combination of the x_k and Z_k and their error covariances are used to produce an updated estimate for the true state of the data, as determined by the Kalman gain. This is fed back to the prediction step and is repeated until the discovery of the best estimator of the future state, k+1. This helps to further isolate unwanted noise, as the method keeps with the constantly present variables, with the noise not varying, which aids in the cleaning of the data.

C. Herff, D. Heger, et al. discussed other methods for use in the cleaning of the data, such as spline and wavelet [43]. From the observation of the transformations on the dataset, however, these had a limited impact on the results after the application of the first two methods and can be viewed outside the scope of this investigation.

Data from each of the 8 sensors data were then grouped together with their sensor pair, creating a 4 channel and, therefore, 4-dimensional system. This step uses the grouping of the data to normalise its findings and can be seen as a step in reducing the dimensionality of the data. The data was then grouped with the rest of the participants to create an average value at each of the time points through the various examples, to standardize the output to reduce the data's uncertainty. As shown in the initial phase of the investigation, not using an average can cause issues in the generalisation and quality of training data for the model.

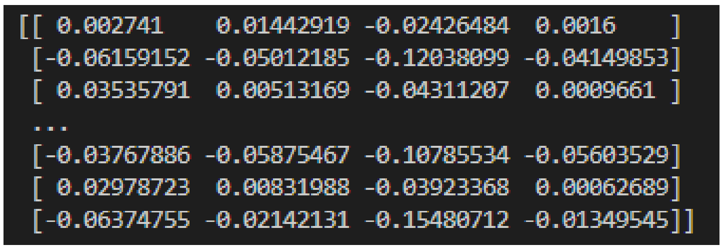
The data, however, was not yet usable within the framework. After importation, the data has to be restructured, as the data format was 300 examples, each with their corresponding task label, with 4 dimensions and 100 time points, meaning there were 4 arrays of 100 time points for each example,



```
[[ 0.01231197  0.0083295   0.00563938 ... -0.07754288 -0.0760253
  -0.07570703]
 [ 0.0048836   0.00556887  0.0059545  ...  0.02319676  0.02394204
   0.02532142]
 [-0.02039215 -0.02517389 -0.02903498 ...  0.01436897  0.02313606
   0.03104017]
 [-0.0244909  -0.02710754 -0.02957898 ...  0.00200009  0.00222418
   0.0026535 ]]
```

Figure 15: Original Data Layout

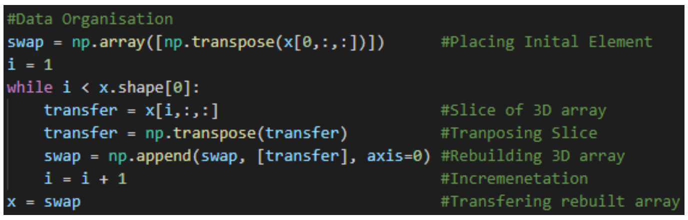
If this data were used in this format, it would be as if there were 4 time points of 100 dimensions, not 100 time points with 4 dimensions,



```
[[ 0.002741    0.01442919 -0.02426484  0.0016    ]
 [-0.06159152 -0.05012185 -0.12038099 -0.04149853]
 [ 0.03535791  0.00513169 -0.04311207  0.0009661 ]
 ...
 [-0.03767886 -0.05875467 -0.10785534 -0.05603529]
 [ 0.02978723  0.00831988 -0.03923368  0.00062689]
 [-0.06374755 -0.02142131 -0.15480712 -0.01349545]]
```

Figure 16: Corrected data formatting

This correction is achieved through the deconstruction of a 3D array into its 2D elements, transposing its values, and then rebuilding the 3D array to reflect how the values should be organised.



```
#Data Organisation
swap = np.array([np.transpose(x[0,:,:])])      #Placing Inital Element
i = 1
while i < x.shape[0]:
    transfer = x[i,:,:]                        #Slice of 3D array
    transfer = np.transpose(transfer)          #Tranposing Slice
    swap = np.append(swap, [transfer], axis=0) #Rebuilding 3D array
    i = i + 1                                  #Incremenetation
x = swap                                       #Transfering rebuilt array
```

Figure 17: Code of 3D Transposition

This is necessary as no function exists in the transposition of arrays when they are greater than 2D, with the swapping of data values and organising the data correctly. Which required creating a new function, Figure 17, that had no researchable design pattern to be implemented as a part of this dissertation. The rest of the framework gets around this challenge through the way it splits the data during its cross-fold validation. While this method is very functional in combination with cross validation, it would slow down the program comparatively, as rather than sorting all the data at once, it would perform this for each consecutive data section. This would increase the time complexity of the overall system and does not allow for use of the transformed dataset for later aspects of the investigation, such as comparing overall labelling accuracy and clustering if done for the unsupervised learning section.

6- Framework Implementation

The unsupervised learning section of the framework is the last phase of this investigation. PyTorch was used to fit the rest of the framework's code, rather than alternatives such as TensorFlow. This phase used the previous research into related implementations of frameworks and unsupervised learning, coupled with the knowledge gained from the initial investigation about the importance of data structures and labelling for the corresponding accuracy of predictions. All played a major role.

6.1: Choice of Model

After deliberation, the choice of what unsupervised learning method to add, came down to two possible options – Agglomerative Clustering and Dendrogram Methods [44]. The Agglomerative Clustering method of k-means has benefits such as commonality, meaning that adding it to the framework would prove more beneficial than other less common methods by encouraging broader uptake and ensuring that the platform will gain more use cases. In addition, Agglomerative Clustering scales to large data sizes effectively. As discussed previously, the size of the data this is an extremely important consideration, as even the smallest datasets in this field are large. It is also adaptive to many data configurations.

Agglomerative Clustering nevertheless has drawbacks, including having to tune the number of clusters (k) [44] increases, the number of seeding issues associated with k-means also increases. Moreover, Agglomerative Clustering also has issues with higher dimensional space, which is problematic as many of the datasets associated with fNIRS have multi-dimensional data. This can require corrections with methods such as PCA and can be aided by using various distance functions, such as L1 [45]. Additionally, k-means has a high chance not to find the optimum solution due to the seed points chosen. To combat these shortcomings other Agglomerative Clustering methods were considered, such as unsupervised k-nearest neighbour. It became evident, however, that due to its lack of widespread adoption and using a brute force approach, which led to a terrible 0-time complexity, meant that this alternative had to be quickly dropped. The run speed of the project became a priority as the system in the investigation is based on has limited capability.

The investigation into the Dendrogram Method found that it has advantages, with a focus its use for Hierarchical clustering [44], particularly Agglomerative. It is also adaptive to similarity and distance attributes, which is an attribute that both k-means and Hierarchical clustering share. Its sum up of data and good utility for smaller sets leads to how it holds more information than other methods, such as k-means, as it shows the hierarchy in the grouping of data.

But Dendogram Method [44] also has its clear disadvantages. It is computationally demanding because it must form multiple layers of hierarchy, making it a slow program. This also affects the memory requirements of a n*n distance matrix, which compared to k-means is extremely high, causing it to fail on larger sets. Moreover, like k-means, Dendogram Method is sensitive to outliers and has a high probability of not finding the optimum solution due to the original seed points.

Based upon the foregoing analysis, k-means was chosen as the unsupervised learning method, particularly because of its capacity to handle large datasets, unlike Hierarchical Clustering approaches, and its lower computational time. This flexibility is necessary when making a program that is meant to cater for a variety of datasets.

Afterwards various k-means clustering functions were explored for their functionality. This led to two leading options – sklearn kmeans and pytorch kmeans. Each has positives and negatives. For example, sklearns affords a higher amount of input parameters and flexibility to accommodate various datashapes, unlike pytorch kmeans. On the other hand, pytorch kmeans has the option of using GPU acceleration, making it a stronger contender. This led to pytorch kmeans to be selected for the framework, reflecting again the prioritisation of speed. A drawback of this choice was it is harder to implement due to more considerations for the datashape, which meant that additional code was required in its formatting, as seen with the transposing of the data, as well as supporting code for the GPU acceleration, seen in Appendix G.

6.2: Implementation to Framework

K-means is then added to the framework. Due to the organisation of the pre-existing code and maintenance of the overall codebase, k-means was placed in largely in learn.py. This is due to the presence of the other callable functions, deep_learn and machine_learn, that are used in the model creation aspect of the framework, making it a good fit. This helps maintenance and more generally usability because it continues to promote the modularity of the framework and it is placed to function with associated components. This involved the use of some pre-made functions that were then edited, due to how they ran on C code. They were extensively tested to be optimum and in turn increased speed.

What k-means tries to achieve within the framework is the grouping data points into cluster groups, with each cluster group [44] separated from other cluster groups as much as possible, while minimising the distances within the cluster. This separation is defined by the distance between the groups, but there are various distance formulas that can be used, such as L1 and L2 [45]. The default that was chosen, however, and the one that was used throughout the investigation was L2 (Euclidean Distance), because of its commonality and the fact that L2 is still effective at 4 dimensions [45],

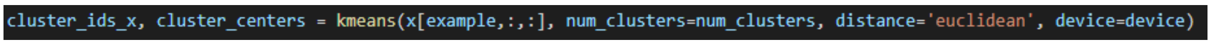
$$2 - \mathrm{Dimensional: d} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

$$\mathrm{n} - \mathrm{Dimensional: d} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 \ldots (n2 - n1)^2}$$

Then due to how the data is 4 dimensional,

$$4 - \mathrm{Dimensional: d} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2 + (a2 - a1)^2}$$

This distance function is used to judge the quality of the various seeding values that affect the cluster groups of the program. Essentially, seeding is the selection of the initial centroids [44] of the clusters, starting at random. Once centroids are found, the mean of these separated groups is used to calculate the repositioning of the centroid in the centre of the cluster group. This is then evaluated by how separated the clusters are and how grouped the points are within the clusters. The process is then repeated with each iteration being compared against the previous iteration to inform the next centroid placement until the evaluation of the output is satisfactory, as defined by how newly formed centroids do not move or the maximum number of iterations is reached. Because the previous iteration informs the placement of the next centroid, if the initial placement of the seed is in a bad position, then the program is likely to fall into a local optimum rather than a global optimum, which will affect the accuracy. The function for the heart of this process is,



```
cluster_ids_x, cluster_centers = kmeans(x[example,:,:], num_clusters=num_clusters, distance='euclidean', device=device)
```

Figure 18: k-means Function

Figure 18 produces the clusters and their corresponding data points. The average number of iterations is then calculated when no constraints were placed on the k-means function for the maximum number of iterations for Figure 18, which resulted in an average of 7 iterations per run. This was deemed acceptable as each iteration of the smaller sets measured in the milliseconds through the measurement of the machine time before and after the function averaging out at 0.043s.

The main issues emerging from the implementation of k-means is the organisation of the data. This was because different components of the overall implementation took various datatypes, such as lists, numpy arrays, tensors, etc. Each of these had to be accounted for and correctly transformed, which can cause issues in organisation. This was especially the case when the data was in a 3D format, where the outer field would transform but not the inter fields, which requires the breaking down the 3D data structure into a 2D form and then transforming when it was to be implemented. This added to the bugs that would normally exist in the program. Nevertheless, the approach to implementing k-means was was done in clear stages, each with a specific task, such as the transposition of 3D data, which allowed bugs to be identified and fixed quickly.

A particularly challenging part of implementing k-means was designing a system to automatically find clusters (k) with which the k-means should run. This is normally a manual input and, thus, is a difficult aspect to figure out. There are multiple reasons to make this automatic. As it helps speed up the testing phase of the framework, since the user would not have to try various values manually. In addition, the rest of the framework is designed to give an output based on no manual experimentation, as it self-tunes the hyper-parameters.

The exploration into how to make finding clusters automatic proved difficult to implement. This is because all the suggested methods were based around using functions to produce a graph, based upon which a human observer would then make assertions. The implementation considered what key features the human observer would look for in the graph and then produce those same results in code.

The first step is by implementing the Elbow method, which was chosen due to its common use and its relatively effectiveness in finding the optimum k value [44],[46]. The Elbow method attempts to determine the optimum k by finding the point where the distance between the centre of the cluster and its data points has not decreased as much compared to the previous step, the user is essentially trying to correctly access the right dip in the graph.

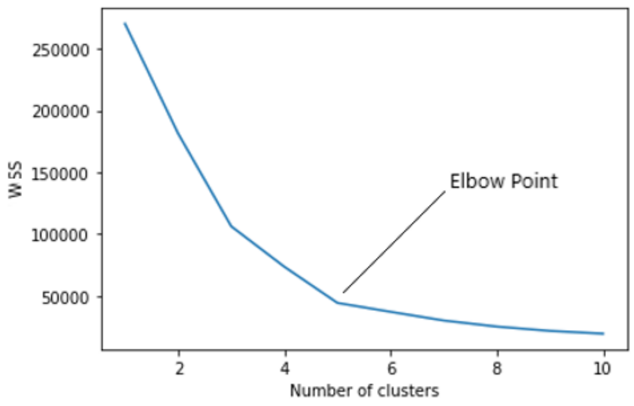


Figure 19: Elbow Method Diagram

As can be seen from Figure 19, the Elbow method tries to find the point of diminishing returns on the use of higher number of clusters, seen with the WSS Score [46]. Here the optimum was identified as k = 5. This is then implemented in the code by comparing the previous drop in WSS to the current one, based upon the percentage change to make the program work with various datasets, where the magnitude may vary. If the percentage change is found not to be large enough comparatively, then the code finds that it has reached the Elbow Point and records the k value.

The advantages of Elbow method are that it is relatively easy to implement, as seen in Appendix D. It works more quickly on non-complex datasets that are smaller in size than other more complex methods of discovering the optimum k, such as using R to calculate the optimum k, which [46] requires more processing time and a larger dataset to function correctly.

A drawback of the Elbow method, compared to other methods to find the optimum k value, is that it only uses distance. This can cause issues at higher levels of dimensionality, as it becomes harder for the program to compare. As points become uniformly distant [46] from each other, it becomes more difficult to separate closer points from those that are further away. This is especially pronounced with Euclidean distance, which can be understood as our view of how the world is structured. This makes it easier to understand for the explanatory phase of the investigation but, like other perceptions of space, breaks down at higher dimensions. It is important to note, however, that this effect is not as pronounced for, L1, Manhattan distance,

$$2-\text{Dimensional: d} = |\text{x1-y1}| + |\text{x2-y2}| \qquad \text{n: Number of dimensions}$$

$$\text{n – Dimensional: d} = d(x,y) = \sum_{i=1}^{n} |\text{x(i)} - \text{y(i)}|$$

This is because of how it works by absolute distance in the equation above, rather than a scaler, meaning that as more dimensions are added, the variations between said distances are greater. To accommodate for this fact, the framework is designed to allow the user to change the type of distance calculation when calling the k-means function. Its application not used for the Elbow method, however, as due to the absolute value it would be more difficult for the graph to point diminishing returns. This leaves Euclidean distance as the default with the option for the user to change if necessary. In additionally, it is not always easy to find the point of decreasing returns as it appears in Figure 19, without any clear point at which the optimum cluster number can be decided.

These issues can be greatly addressed through implementation of Silhouette Score [44],[47],[48], which does not only calculate the distance but also considers variance, skewness, and high-low differences. Silhouette Score works by measuring how close the point is to its nearest neighbour of points and then applies this to all the clusters, which gives an indication of how well grouped the different cluster groups are. This produces a Silhouette Score, which is then recorded, and the process is repeated with various k values, with its stopping condition indicated by a decrease in returns. The decrease in returns is found by the coefficient value, as produced by the programs output, which can then be graphed,

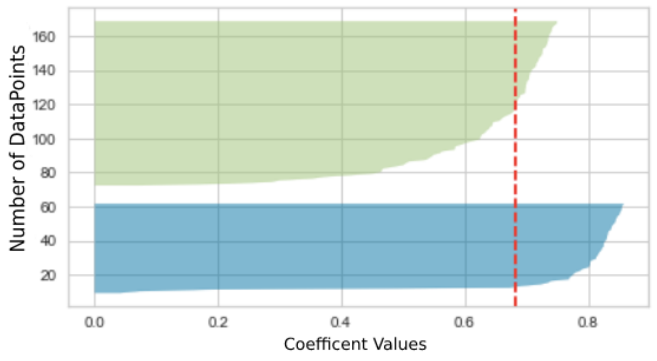


Figure 20: Suboptimum Choice - 2 Clusters [48]

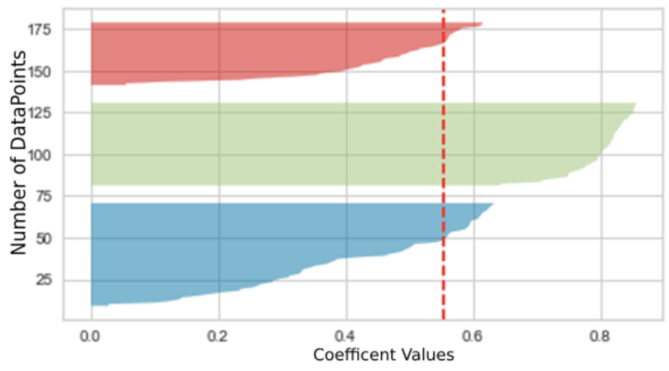


Figure 21: Optimum Choice - 3 Clusters [48]

The optimum choice is found based upon few characteristics, as previously mentioned. This translates to a few aspects to look out for in the graph. The red line in Figures 20 and 21 represents one of these. As the user does not want the graphs to fluctuate below the average score, due to wanting all clusters to be considered well formed, both figures pass this test. The next test is whether the graphs are uniformly spread, such when one cluster to consists of a vast majority of the points. This can be seen in Figure 21, with the green cluster group, making this cluster number suboptimum. To implement this into code, a slightly different approach is taken. The average silhouette score (x), is decided by a combination width of the cluster groups and previous metrics, is used and the number of clusters (y),

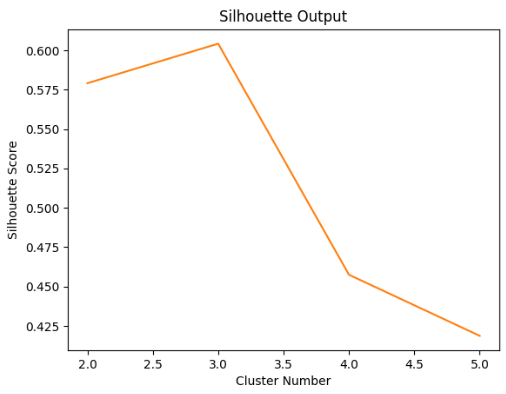


Figure 22: Silhouette Score Output

This graph shows how the optimum number of clusters is 3, which is the highest point on the graph. This was to be expected based upon Figures 20 and 21 and gives validity to this more customised method that was necessary to implement the Silhouette concept into the framework. With its coded implementation for this process seen in Appendix E.

To give a more creditable k value, the data is then split up into smaller randomly divided sections by shuffling the dataset, with an average k being the output to apply to the whole dataset as shown in Appendix F. Essentially, the same principle as cross-fold validation is being applied here. This means that the framework ran the methods for finding k multiple times on various configurations to give an unbiased answer for what the best k value should be.

After the value of k is found, its used on the overall unshuffled dataset using k-means. This required some slight manipulation to the dataset to make it work, as k-means only accepts 2D tensors. This led to the examples and time points being combined, which later had to be account for. Another aspect that had to be accounted for is that the dataset has labels to compare against to see if the system is functional comparatively, but these labels are given to each of the examples rather than to each data point. This is to be expected as each example encompasses one task type.

To increase the reliability of results, rather than only grouping the time points into clusters and then those cluster coordinates having labels placed by k-means, which remains an option for the user as it may fit better in certain situations when compactness [3] is an issue, it is decided that labels would be applied to all the individual data points and then a primary label of each example group (cluster) would be gained by averaging their labels. This idea was derived from its similar implementation in the original investigation in section 4, by splitting up people into individual groups and averaging the data. The best labelling method is found during the initial phase of the investigation and is useful in a few respects. One is to reduce the dimensionality, as k-means is run over the dataset and then the average value of every 100 time points is used to assign a label to a cluster. This mean that if k is found by silhouette or elbow methods to be 4, but when the clusters are found and averaged there is no 4th cluster, then the program can be seen as deciding that the 4th cluster is unnecessary. Another aspect that makes this method better is that it helps in the removal of outliers from consideration when applying a cluster, a specific label, as it takes the average label. This means that even if 30% of data points are outliers and don't represent the true nature of the data, due to aspects such as left-over data noise, the correct number of clusters will still be found, giving more accuracy to the overall system.

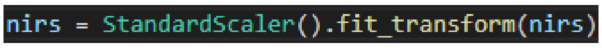
There remains the point of what system of averaging should be used. Originally, the use of mean was on each of the cluster groups to produce the average label, but this immediately raised an issue. For instance, where there are 3 clusters (0,1,2) and the mean of the value is always taken and then rounded to the nearest whole number, then it will be pick 1 more often than it should, even if the average point is nowhere near 1 as the cluster labels are assigned arbitrarily. In a cluster with a 60% label 2 and 40% label 0, the system will just default to giving the whole cluster a label of 1. This is left in as an option for the user because of possible use cases, such as wanting to only label datapoints that are most closely related to the label group, with label one being used as a category for those that do not pass.

To combat these problems, a mode system is then introduced, as seen with Appendix F, which takes the label that was most repeated in cluster and assigns it to the cluster. This gets rid of the issue of how the middle label will be picked disproportionately and helps with giving more credibility to the results as it goes with the majority.

Another issue, in this case dimensionality, which is also addressed, however, through the application of the previously mentioned PCA method in the framework. Normally, with PCA a standardization process is already administered to the data [5],

$$z = \frac{value - mean}{\text{standard deviation}}$$

The equation can also be formatted into code,



```
nirs = StandardScaler().fit_transform(nirs)
```

Figure 23: Data Standardization

Figure 23 is used to prevent a few early variables from dominating the training of the model, which is slightly less necessary here, as all the variables in fNIRS are working on the same metric, but it is nevertheless good practice. After this a covariance matrix is applied to the data,

$$[\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_n] = \begin{bmatrix} X_1(t_1) & X_2(t_1) & \cdots & X_n(t_1) \\ X_1(t_2) & X_2(t_2) & \cdots & X_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ X_1(t_m) & X_2(t_m) & \cdots & X_n(t_m) \end{bmatrix},$$

Figure 24: Covariance Matrix for n Dimensionality [5]

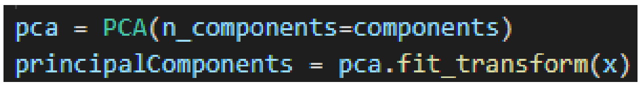
Application of the covariance matrix identifies redundant information, due to variables having overlap, present in the dataset that can then be used to reduce its overall dimensionality. Variables can be highly correlated to each other, and the matrix shows the covariance with each possible paired variable. If the value is positive then they increase and decrease with each other, 1 is perfect correlation and anything above 0.8 is considered strong [5]. If negative, the reverse is true, with -1 being perfect inverse correlation and anything below -0.8 considered strong [5].

After this the data are used to constructing new variables, principal components. This works by constructing linear combinations – as principal components represent the direction of the data – from the data in such a way that each of the principal components are the least correlated from each other as possible. These linear processes and points are done using eigenvalues and eigenvectors to explain the direction and variance of the data, with each principal component having its associated eigenvalue and eigenvector pair. Importantly, the first component accounts for the largest possible variance, explaining the most information, and then the using the left-over uncorrelated information to account for the variances as much as possible, with the data placed in the second component and so on.

The number of principal components is dictated by the number of dimensions to the data. Then, by comparing the various principal components by their eigenvalues – as the eigenvalues show how strong the system is in a corresponding eigenvector direction – it can then be decided whether to keep all the eigenvectors or some can be disregarded to reduce dimensionality. This cut-off value can be seen to vary for the desired outcome in terms of dimensionality reduction. A cut-off value of 0.8 is adopted to represent that the eigenvectors have less information than can be, but this value may vary depending on the dimensionality reduction sought. The remaining principal components are then used to form a feature vector [5], which is then used to recast the data from its original orientation, forming the final set,

$$\text{FinalDataSet} = \text{FeatureVector}^T * \text{OringinalDataSet}^T$$

This process was also codified and from the Figure 19 to the FinalDataSet can be seen with,



```
pca = PCA(n_components=components)
principalComponents = pca.fit_transform(x)
```

Figure 25: PCA Main Function

Figure 25 gave the final dataset and information on the principal components through the variable PCA. It required the conversion of the 3D data into a 2D format and then recombined, much like how was done in the original data organisation. PCA was additionally placed in the model to either be called upon as a part of a k-means run or for the reduction of dimensionality for its own sake, furthering the goal of making the code continually modular.

7- Framework Analysis

The methods and functions shown and explained above were then ran and analysed for their effectiveness. This process started with k-means.

The process of finding the correct k value proved successful when using both the Elbow method and silhouette score. Success was determined based on a comparison of the number of clusters found against the number of clusters as given by the labels, this being k = 3. This confirmed that the framework additions accomplished one of its key objectives as, without any input, the system determined what value to give k.

As expected, silhouette score proved to more reliable, with it effectively having a 100% reliability rate in finding the correct value for k. This proved to not be the case, however, for the elbow method, which alternated between finding a k value of 3 or 4 depending on the run, because of the inherit randomness in the original seeding. Interestingly, the 4th cluster could be caused by a perceived in between state, before the task reaches its full mental workload, leading to the program discovering a separate distinction.

Nonetheless, by using the previously described mode method in the averaging of found labels in the system, and then using the number of clusters present in that average as the k value, the elbow method proved almost as effective, with an 85% chance in finding the correct k value of 3, with a k value of 4 representing the other 15%. This outcome provides further confirmation that silhouette score is the superior method, even if the elbow method performed satisfactorily. The reason why it compared against the given labelling system is that the given labelling system is already optimised and therefore acts as a good comparison point.

The methods, however, validate each other outside of using a labelling method. With an expected overlap of 85% for 3 clusters, the system can validate the number of clusters without needing a given label set. Although, if there is a difference, the silhouette score is prioritised due to the higher accuracy [48].

This k value, as gained by silhouette, when placed inside the k-means labelling process, produced labels for the time points. Then by taking relevant knowledge gained from the initial phase of the project, seen in section 4, this labelling process was placed through several tests to gauge its effectiveness.

The process of gauging effectiveness started with looking at how well the two label groups, those created and those given. The first focus of investigation was how the individual time-points faired in terms of labelling, meaning that no average was used. This was done as a continuation from the initial phase of the investigation into how the breakup of data points from their averages affected the program's effectiveness. This required the creation of specific functions to calculate the average distance in 4-dimensional space.

The average distance calculated was 5.46e2 from the centre of the cluster to the data points, which is a very strong outcome in terms of the cluster's compactness. This cannot be used, however, as a label is assigned to each example group. For this reason, a mode filter was applied on each example group due to the reasons discussed above. The result was a distance of 3.56e3 from clustering the data points. One of the reasons for this decline in effectiveness is the grouping of datapoints. This was due to a high impurity in terms of what cluster labels exist in the example set compared to what the overall label was found to be. Some reached percentages of 50% impurity and the average was 23% impurity, as various time points of the system more closely resembled other tasks in their level of mental exertion. What this means is that all the labels that then existed in datasets were tied to a cluster centre that might be nowhere nearby.

Other organisational structures were also explored to correct this issue. As during the initial phase of the project, the values were averaged, in this case by running k-means over the mean point in the example groups, with k-means producing 3.213e3 from the centre. This is an improvement from the mode over the datapoints method and is in line with what was expected. The initial phase of the investigation found that the use of averages when placing data through machine learning functions is an effective tool to increase effectiveness, because of the training fitting the average data point better, as represented in this case by the compactness of the clusters.

The reason for this effectiveness change in terms of functionality is that when the points are placed in the k-means program they are already grouped with their example set. Consequently, when k-means is run, it is designed to fit the average point more than the best grouping of the individual points. The individual method, however, did show that when left to its own devices k-means was able to find patterns in the data between the various tasks, as it labelled the individual points not necessarily uniformly in each example set.

This led to the using the mean value from example groups for k-means labelling to determine how the unsupervised method was to decide the labelling of data points. Importantly however, it did not perform as well as the given labelling.

By using the mean value of all the time points in each cluster, the given labelling provided an average distance of 2.437e3 from the centre. This means that the given labelling produced a closer grouping in terms of the clusters by an average of 24.26%. The reason for this is most likely the found in the weaknesses of k-means described earlier in section 6.1. It is sensitive to outliers, seeding issues and intent on spacing the cluster centres. The reason for using distance from the centre as a metric is to see how effective the grouping of label methods is in terms of compactness, as seen in the overview below,

Labelling Type	Distance From Center
kmeans - Individual	5.46E+02
kmeans - Mode Filter	3.56E+03
kmeans - Clustered Time Points	3.21E+03
Given Labeling	2.44E+03

Figure 26: Time Point Distance Overview

The spread of the clusters was another metric that was sought after. As k-means specifically spaces the cluster group centres [3] from each other, however, rather than just grouping by centration by the given method, this caused the spacing to be on average 10.4x larger in the assigned label cluster group. This shows that, while one metric may prove to better in one case, there are other elements to consider.

Interestingly, when the mode filter method was compared to the clustered time points in the deep_learn function, the mode filter method outperformed the clustered method by 6%, making it the leading choice for the labelling method of the investigation. This shows that the compactness a cluster, while important, does not show all aspects.

The given labeling and created are tested to see definitively which method proved superior, with both outputs placed in the deep_learn function, with the previously created ANN model.

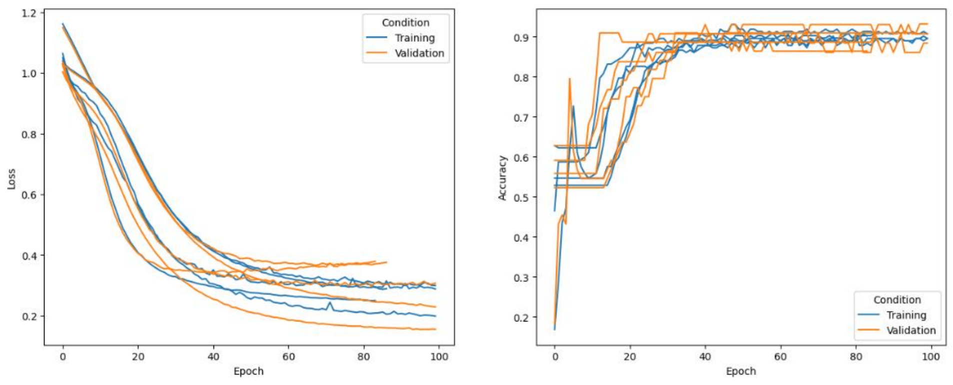


Figure 27: Output for Given Labels

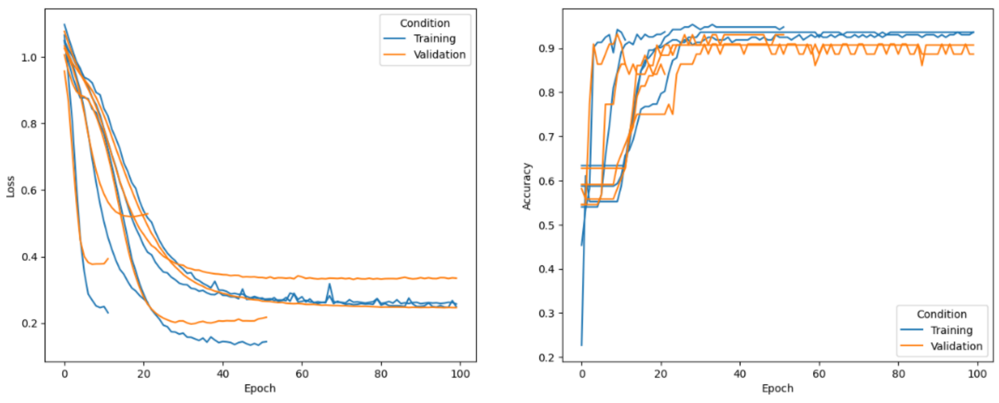


Figure 28: Accuracy for Created Labels

As seen in Figure 27 and 28, there is a clear and noticeable difference in accuracy in favour of the created labels. Importantly, these graphs show multiple training runs of the code, as cross-fold validation is performed on the labels and dataset to give a more accurate view of how well the models preformed, with some of the runs cut off due to their lack of performance in comparison to the leading model. Figure 27 averages out at 89.4% accuracy on this run, with Figure 28 at 91.2%. This is an exceptional outcome, as not only did unsupervised learning perform to a very high accuracy, but it also even outperformed the assigned labelling. This was not expected of the unsupervised learning and shows the strength of the created method.

What can be seen from both the loss and accuracy aspects of Figure 27 and 28 is how much faster the created labels improved in comparison in both the training and validation sets. As the created labels were the stronger candidate throughout the training process, as seen from the steepness of the graphs in Figure 28. Multiple runs of this process to confirm the results fell into the boundary of error, and each time the created labels outperformed. Intriguingly, the loss value is harder to compare, with the created labelling coming out ahead at 0.3 to 0.32, but with given labelling having the lowest fold value at 0.24. This makes it questionable which has the better generalisation [49], with the results leaning in favour of the created labelling method.

K-means proved to be successful and an interesting addition to the framework, with it even outperforming the given labelling. The difference shows its effectiveness and possible widespread application in the mentioned use cases, with it here finding patterns for which the original labellers, C. Herff, D. Heger, et al. , did not account. Importantly, other metrics should be compared by use of a confusion matrix to check the integrity of the created dataset through a combination of all the runs,

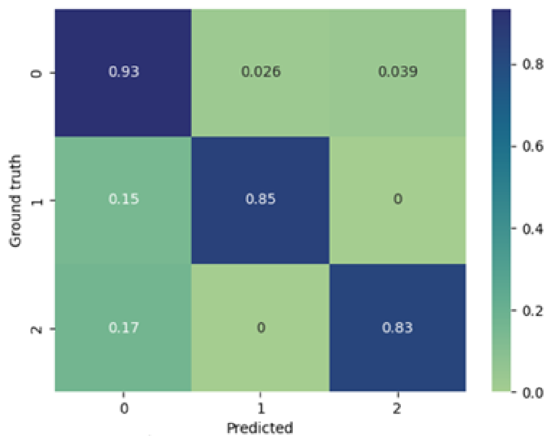


Figure 29 : Confusion Matrix Given Labelling

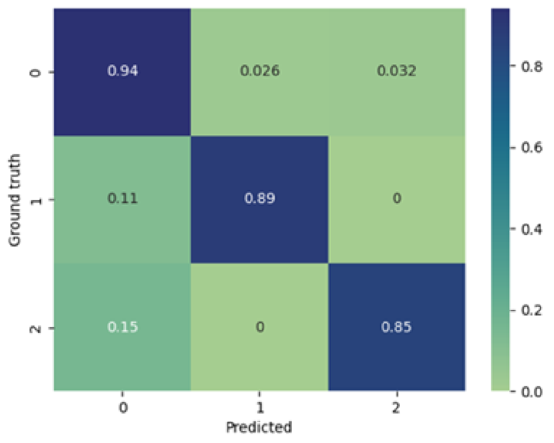


Figure 30 : Confusion Matrix Created Labelling

As expected from the lower accuracy of the given labelling model, the confusion matrix, Figure 29, has an overall lower accuracy, as seen in its ground truth against predicted tabling. Interestingly, Figure 29 and Figure 30 shows signs of slightly overfitting. Labels 1 and 2 never accidently predict each other, with the only source of error in both matrixes being not predicting 0 correctly. This required looking at other issues, such as how the labelling method created 5% more label 0s than the even spread of the original dataset, which would create a bias towards labels 0s performance. The core issue here is an unbalanced dataset [50]. However, this also led to a higher accuracy and Figure 29 shows the same issue.

Interestingly, however, the increase in labelling 0 did led to the issue. If the process of all the tasks is considered, they have a low starting point of origin in terms of mental workload and then work up to their maximums. This means that overall, the system mimics task 0 for more time due to it being the easiest task, rather than the hardest task, 2. When the total data of each example group is considered, it is much harder to tell between task 0 and task 1 then task 1 and task 2, as the difference is less notable. A solution is to disregard the starting data and look at the middle of the example set. Comparing only the peaks of each task rather than its whole run would allow for a more comparable difference; however, this simply raises issues with the dataset rather than method or model.

The difference between the two models should be compared with other metrics, particularly in an unbalanced dataset, to tell the whole story,

TP: True Positive FP: False Positive FN: False Negative

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Recall, out of all the positive cases how many are predicted correctly, higher is better

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Precision, for all the positive predictions, how many are truly positive, higher is better

$$F - measure = \frac{2(\mathrm{Recall} * \mathrm{Precision})}{Recall + Precision}$$

F – measure is a combination of both recall and precision as a weighted harmonic mean [47], which is important when testing and trying to increase one score without decreasing the other, higher is better

The recall of the created labelling on average is 0.893 and the recall on the assigned is 0.870. The precision of the created labelling on average is 0.906 and the precision of the assigned is 0.891. This allows for F-measure to be calculated by using the average values, 0.899 for the created labelling and 0.880 for the given, but normally is not used for non-binary datasets [50]. The reason could the added complexity in its calculation and whether averaging the F-measure of each class or averaging the precision and recall lead to different values for the F – measure. What was found by all the metrics was a slight but noticeable difference in favour of the created label dataset, making it by all metrics the superior option.

The other unsupervised learning method that was introduced into the dataset was PCA as an effort to reduce the dimensionality of the data and improve data visualisation [5]. Initially, the dataset was tested to be reduced from 4 dimensions to 3, with the output being graphed and the time points being created and automatically inserted,

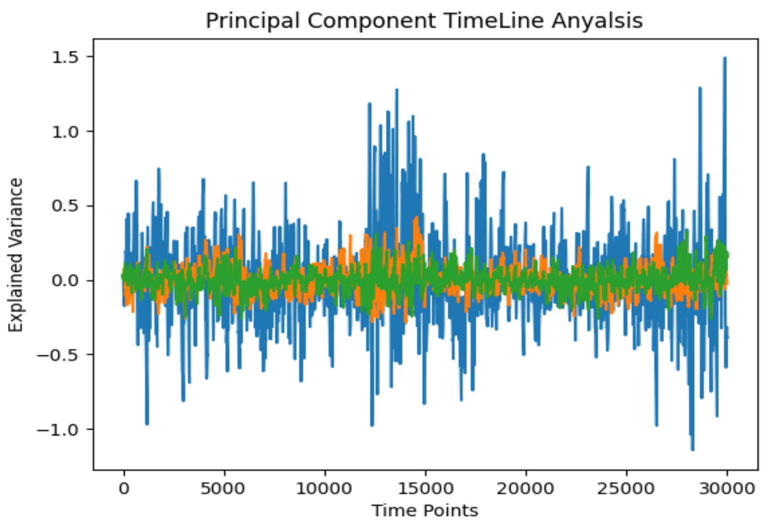


Figure 31: 3-Dimensional Explained Variance

The output showed that the principal component could explain 84.9% of the variance on its own, as seen in blue, with a further 7.2% explained by the second principal, as seen in orange, and 6.1% explained by the third, as seen in green. This means that the model lost 1.8% of explained variance. Interestingly, 84.9% in the principal component is above the 80% threshold and therefore can be considered accurate enough to use on its own, allowing for the effective reduction in dimensionality, if necessary, to a single component. The main reason for this ability in dimensionality reduction is most likely since when one region in the frontal lobe experiences increased activity, the entire frontal lobe is more likely also to similarly show increased activity [7]. This allows for PCA to find similarities between the data and group them effectively.

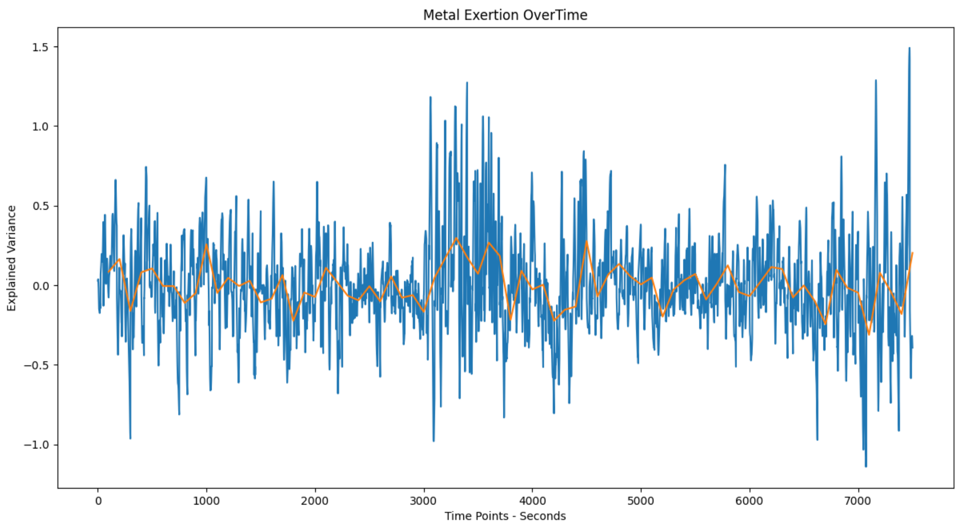


Figure 32: Variance in Mental Exertion

The blue again represents the principal component, graphed here rather than the time points, in seconds. This is not the total time to take the test but is how the data is viewed because of the standardization of frequency by the framework. What can be seen from the orange trend line in Figure 32 is the change in mental exertion. On average, mental workload is increasing much more in the middle of the experiment. This can be explained by the waiting periods not fully allowing the subject to recover in-between tests, with the sudden decrease being the period of prolonged break previously explained in Section 5.2. The decrease can also be seen to be occurring at the end of the tests.

This raises an interesting issue with the created labelling. If the data is peaking and not in direct correspondence to the tests, the intensity of the tasks may not match to their counterparts. This may explain why the created labelling has a higher accuracy than the assigned, as it gave labels that account for this difference, while the assigned labelling did not. A way to fix this would be to further segment the tests to reduce what is effectively cross chatter[7] from mental workload from heightened mental exhaustion.

Further testing to see how PCA would affect the accuracy would likely show a negative effect in the predictive capacity of both. If the number of principal components is lower than the number of dimensions, the program would have less information and therefore less of a chance to produce an accurate output. This was shown in testing, producing similar metrics to those above when the components were reduced to 3, as expected, because they encompassed 98.2% of the information.

It would positively impact the k-means labelling system in comparison to given labelling, as the given labelling would even less correspond to the newly created data set. To note four dimensions is functional for both the deep_learn program as well as k-means, as the value at which k-means starts to suffer more prominently under dimensionality issues is greater [3].

8- Conclusion

The main goal of the project was to understand methods of labelling and in turn use that understanding to put unsupervised learning into the framework. This was accomplished. Importantly, unsupervised, and supervised learning had, contrary to the predicted result, a difference in accuracy in favour of unsupervised learning. The goal and predicted outcome of simply producing a viable unsupervised learning that did not differ from its supervised counterpart too negatively was not only achieved but exceeded. The fact that the difference is only minor, however, even with the dataset issues, show that the created supervised learning method is still very applicable. This is especially the case when the datasets and models are tailored to benefit it, taking the suggestions set out in Section 7.

For use cases section 1.3 can be looked at, however, its important to note that this doesn’t replace the system for supervised learning. As the labels are directly connected to the tasks that are done. The end goal is to design a system that can predict the tasks the user is doing, rather then what task the users output most look like. Additionally, it also requires more time to this extra process which cannot be ignored. What the results show however, is that a pattern exists outside of the task intensity, and it is a feature that needs to be account for.

9- Reflection

9.1 Project Management

Project management to be challenging, principally because of the extended period that the project encompasses. To manage time, a waterfall methodology [51] was adopted for larger tasks, as it provides a clear methodology for moving forward, as represented in the Gantt Chart,

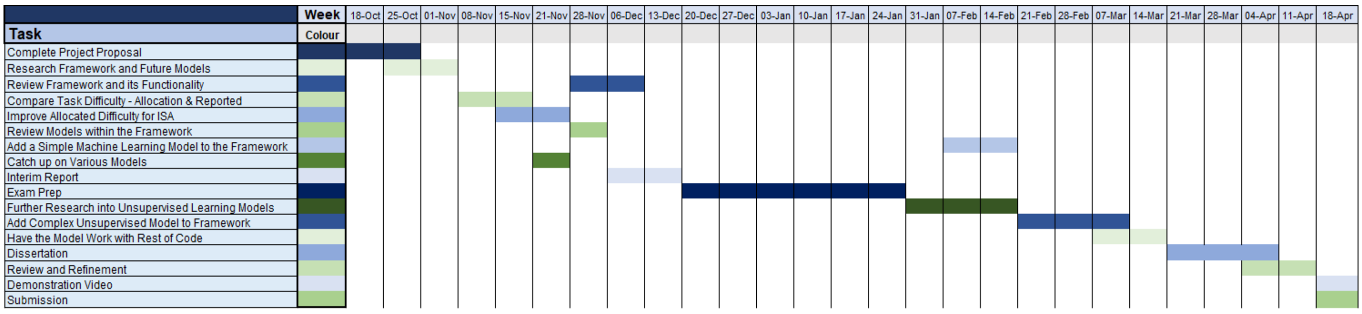


Figure 33: Project Management Plan

The time was split into weekly bricks for what should be done at what point, with the Gantt chart giving more time to tasks toward the end of the project than they truly needed, in order to account for overrun of earlier tasks. For example, one week was allocated for the demonstration video, which can be considered a smaller task. This allowed for flexibly within a rigid method. This flexibility was used on multiple occasions when certain aspects, such as the data organisation, took more time than expected. Extending the time required for having a working model within the framework by one week was manageable, because of the flexibility previously built into the project management plan. This adaptation reduced the risks of not accomplishing the project or not finishing it with satisfactory quality. As there are many unknowns when working on a project in terms of how long each task will take, giving an effective margin for error can allow the project still to be completed on time.

Importantly, the Gant chart above evolved from the start of the investigation, which is found in Appendix H. Some elements were assigned more time or moved around, as more it became clear that certain aspects of the project would take to longer complete or were better left for later in the investigation. Adding a simple machine learning model eventually resulted in the Gantt chart seen in Figure 33. A lesson learned from experience with the original work plan was to give more leeway to accommodate setbacks and mistakes. While a rigid structure provides a clear path forward, it is also not dynamic enough to deal with real world stresses, ranging from coursework to illness.

Other real-world stresses can also be seen with the credit split of the chosen modules throughout the year. As a 70:50 credit split was chosen to give more time in the second term to account for a possible increased level of work arising from work needing more time to complete the project, as well as additional features to be added to the framework, such as was done with PCA. This also allowed for a clearer focus on the bulk of the work for the project, including the writing of the dissertation itself during the second term, relatively unobstructed by other coursework. Another real-world stress arose from the system that was the basis for the dissertation, with the risk that the computational power required would eventually become too great to realistically allow for testing and producing results. This led to multiple outcomes, including choosing the smallest dataset to reduce training time. Another consequence was the decision to make speed a key focus in the investigation to attempt to stop the ballooning of computational times. Finally, in case the two previous attempts to reduce time complexity did not work, access to the university cluster was obtained early on, so as to hedge bets.

Complexity was also reduced by subdividing large tasks into smaller aspects that they encompass, which were then done in a sprint format on a weekly basis. This coincided with the weekly meetings with the supervisor. The overall structure, therefore, was a seemingly combination of both the Waterfall and Scrum methodologies [51],[52].

Before each meeting, the tasks that were accomplished were laid out, as well as the meeting agenda. This allowed both the supervisor and the overseeing PhD student to understand what the meeting would cover and allowed them to research features relating to questions and ensure meaningful and measurable outcomes. For example, sharing a meeting agenda in advance that highlighted the need for a technical writing resource, which was provided during the subsequent meeting and then reviewed and discussed, something that would not be possible if the issue was raised during the meeting due to the limited time frame. Another benefit of this organisation choice was that it kept the supervisor well informed of the state of the project and allowed the supervisor to indicate if more work needed to be done to accomplish the project in time for submission. This helped drive the project with a more continues workflow, spreading out the pace and generally increasing its quality, due to the lack of rushed work.

To further reduce risk, the project was also done in clear stages, the initial investigation, the creation of the ANN model and then moving on to unsupervised learning. This progression is seen with the addition of a simple k-means program, then tuning the number of clusters automatically with the elbow method, then employing silhouette, and finally adding the PCA unsupervised learning method to the program. What can be seen from this simplified order of work is how the project started off from a simple point and continued to build from that base. This meant that even if the more challenging aspects of the project, such as PCA, could not be accomplished, there was still a lot of work to fall back on, leading to an effective reduction of risk. Due to effective management, however, this did not happen.

The project management strategy also proved effective for accomplishing work faster, with a focus on clearer smaller term goals, rather than trying to reach too early for the highest complexity. In multiple ways, this was an effective strategy and eventually resulted in a successful project that clearly demonstrated the potential of unsupervised learning.

What was learnt from this method of project management is that it is extremely important to have clear goals that can result in turned around and completed weekly. This allows for projects to move at a steady pace and with better time management than setting even monthly goals, which can feel far off and be seen as something that can be put off.

A healthy balance between rigid and dynamic methodologies seems to have performed the best, as a rigid format allows for a clear path forward and dynamic features allows the project to cope with internal and external pressures, as achieved with this project. Another lesson learnt was to hedge bets when taking risks through mitigation actions. When unsure about what will eventually be required to build a product, it may be prudent to plan for the worst outcome, as was done in this case by thinking ahead and gaining access to the university cluster. Another lesson that was known but was truly learnt during this project is to start simple and build complexity from there. This helps in both the organisation of time and goals but also in the product, with the general goal in the reduction of risk by always having a deliverable from the early stages.

9.2 Accomplishments

The project overall was ambitious, as it took a complex framework and a topic that was not previously explored, sought to understand both, and then add more features and complexity to the framework in hopes of creating features to aid the related fNIRS field. This goal was achieved and exceeded.

In addition to the agreement of adding a model to the framework and investigating labelling, the project's aims were exceeded by the addition of PCA, which allows for easy data visualisation and dimensionality reduction and provides the user the minimum number of dimensions to make a viable dataset. Moreover, the additions to the model also implement not one, but two automatic methods for tuning k-means cluster number.

The project's novelty can also be seen in the fact that the work is based upon an unreleased framework, which does not have an existing comparator, and adds to its functionality. The work involved was not something that, in many cases, could be looked up and co-opted but rather had to be made from scratch. An example of this is the systems that find the cluster number, which can be also used for future unsupervised learning additions. What was novel about these systems is that they give an output that is meant to be interpreted by a user, but by taking what the user is looking for and breaking that down into criteria, then encoded it and allowing the cluster number to be found automatically without user interaction.

Further, the novel unsupervised learning method provided here, using mode filtering, allowed the project to exceed the normally stronger supervised learning. The approach easily passed the 8% worse threshold, clearly reaching a measurable outcome. This allows its application in both avoiding labelling and finding patterns in the data that was not considered by the researchers as seen in section 7.

Another novelty that came with the project was a clear function for the transposition of 3D in the correct configuration, something that library functions were not able to accomplish and making it a new addition. Additionally, the overall project was not chosen, but created. The goals and objectives were not among those listed or suggested, but rather representing a new concept to develop and again shows its uniqueness.

All this helped bring together a product that is viable for use and added functionality to a framework that can aid the field of fNIRS research with its models, methods and standardization.

9.3 Limitations

The created functions were not explored with other datasets, which may have brought up issues. This would especially be the case for the use of the elbow and silhouette method, as they are untested outside of the dataset. This means that they could break in some hypothetical situations, such as when the elbow is never clear for the elbow method. Time constraints on the project disallowed further testing and the limitations of the system, as previously described, required use of the smallest functional dataset. New datasets are easily accommodated, however, as the code is designed to fit various data shapes, as it takes information from the dataset. This is one of the clear successes in the development of the framework, including the added methods. It is a usable product and not only applicable to the tested dataset, which is also a clear achievement.

Conversely, a limitation is the complexity of the framework, including the additions. While the outcomes of the project and framework in general can be used professionally, it has a higher barrier of entry than would be preferable. The user must have at least a basic understanding of code, preferably Python, in order to change the settings and add datasets in the framework rather than having an easy-to-use UI. Another limitation is that, to fully investigate their model or dataset within the framework, a vast amount of computing power must be used, which would require access to a cluster, in order to run the program multiple times over multiple models with a dataset that is larger than the one used in this investigation.

Additionally, the accuracy of the labelling was only done with a single model, ANN. This model was selected correctly for its lack of bias and naturally high generalisation, with this fact backed up by the results of the tests ran in Section 7, which showed a low difference between the training and validation sets. It nevertheless still lacks the thoroughness that would be found if it was tested with a range of models which may have brought additional issues to light, such as what impact use of a LSTM model would have, due to the use of recurrent memory [38],[11] affecting the training and its effect on the supervised and unsupervised labelling in the framework.

The code in the framework, while modular, also could be more accommodating to adding additional unsupervised learning models. While new datasets are easily accommodated, this is an issue because the added functionality in the framework would need to be reworked for any new additions. This highlights some of the limitations of the project for when used and naturally expanded.

The dataset used is also a limitation, as when trained, naturally gives bias to tasks closer to the ground state. There are solutions to this, as discussed in Section 7, as a function could be created to deal with this issue. This would not apply to every situation or dataset, however, and would limit the universal of the framework for fNIRS data. Additionally, it would be important to see how the supervised and unsupervised methods perform when the dataset is less biased. Therefore, unlike Figure 32 where the change in variance did not always correspond closely to the tests but rather to the built-up exertion of taking the tests, it would be important to see how unsupervised learning would perform comparatively to supervised learning in this situation.

Another limitation is that certain aspects, such as the distance function used for k-means, while easily changeable, are not a setting for the user and the default is simply applied. This done due to the limitations of time, as it would inevitability create bugs, but also because it would add further complexity to the frameworks, as more settings and parameters would need to be changed.

9.4 Future Additions

Some of the issues and limitations outlined in Section 9.3 can be fixed in future additions to the framework.

It may be possible to use more forms of data to make more accurate predictions, such as including the heart rate of the participant, which increases in correlation to increased cognitive thought. Other data gathering could include recording pupil dilation and skin temperature [7], both of which increasing with greater brain actively. This would allow for more accurate predictions of the level of mental exertion, by relying on more than one form of data gathering with the use of fNIRS. fNIRS is not a perfect system, with data noise and sensors experiencing cross talk. These weaknesses can be mitigated by further information to disprove apparent peaks in mental exertion.

Another addition would be the use of latent class analysis for further unsupervised learning. Latent analysis addresses a lot of the criticism that hierarchical clustering faced in this investigation. The choice is no longer arbitrary, as the classification groups has a statistical backing, and this in turn provides a formal criterion. It would be interesting to see this applied for fNIRS data, as it is particularly useful for identifying shared characteristics between the data. And creates a formal criterion for the classification of certain groups makes it an interesting choice.

Additionally, there are other methods for deciding what k value to use that can be added, even if they are less often applied. The Gap statistic method, for example, focuses on maximizing the gap between clusters, and computing the cluster number with R [44]. This is the most complex, as it not only finds the number of clusters but the optimum method (i.e., Elbow, Silhouette or Gap) to cluster the values. This would provide even more of an automatic benefit to the user.

The last future addition would be to include a catalogue system. This would simply require the addition of the methods names and the methods being placed in the same directory to be searched and used. This would work very similarly to how the supervised learning methods are called up by simply typing their name when calling a learn function, such as either machine_ learn or deep_ learn. This allows for the quick addition of new models and the future expansion of the framework by researchers, with a UI overlay to aid the less technically inclined.

10- Bibliography

[1]- M. Ferrari, V. Quaresima, "A brief review on the history of human functional near-infrared spectroscopy (fNIRS) development and fields of application", ScienceDirect, NeurolImage, pp. 921–935, 2012

[2] - J. Benerradi, "BenchNIRS", gitlab, https://gitlab.com/HanBnrd/benchnirs

[3] - S. Na, L. Xumin, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm", IEEE, Third International Symposium, 2010

[4] - Q. Liu, Y. Wu, "Supervised Learning", Research Gate, 2012

[5] - I. Jollifee, J. Cadima, "Principal component analysis: a review and recent developments", The Royal Society Publishing, Volume 274 - Issue 2065, 2016

[6] - D. Weishaupt, V. Koechli, B. Marincek, "How Does MRI Work?: An Introduction to the Physcis and Function of Magnetic Imaging", Springer-Verlag Berlin HeidelBerg, pp. 1-5, 2008

[7] - M. Causse, Z. Chua, "Mental workload and neural efficiency quantified in the prefrontal cortex using fNIRS", Nature, Sci Rep 7 - 5222, 2017

[8] - H. Maior, M. Pike, S. Sharples, M. Wilson, "Examining the reliability of using fNIRS in realistic hci settings for spatial and verbal tasks", In Proceedings of CHI, Vol. 15. pp. 3807–3816, 2015

[9] - J. Kumar, P. Bhuvaneswari, "Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study", Science Direct, Vol. 38. pp. 2525-2536, 2012

[10] - B. Burle, L. Spieser, et al., "Spatial and temporal resolutions of EEG: Is it really black and white? A scalp current density view",International Journal of Psychophysiology, vol. 97 pp.210-220, 2015

[11] - A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network", ScienceDirect, Physica 404, 2019

[12] - Y. Park, S. Lek, "Developments in Environmental Modelling", ScienceDirect, Chapter 7, 2016

[13] - H. Dike, Y. Zhou, et al., "Unsupervised Learning Based On Artificial Neural Network: A Review", IEEE, International Conference on Cyborg and Bionic Systems, pp. 322-327, 2018

[14] - J. Bennerradi, H. Maior, A. Marinescu, J. Clos, M. Wilson, "Exploring Machine Learning Approaches for Classifying Mental Workload using fNIRS Data from HCI Tasks", Proceedings of Halfway to the Future Symposium, pp.1-11,2019

[15] - R. Li, G. Rui, et al., "Early Detection of Alzheimer's Disease Using Non-invasive Near-Infrared Spectroscopy", Frontiers in aging neuroscience, vol. 10 pp. 366, 2018

[16] - P. Dans, S. Foglia, A. Nelson, "Data Processing in Functional Near-Infrared Spectroscopy (fNIRS) Motor Control Research", Brain sciences, vol 11, 2021

[17] - J. Pena, J. Lozano, et al., "Dimensionality reduction in unsupervised learning of conditional Gaussian networks", Transactions on Pattern Analysis and Machine Intelligence, vol. 23, pp. 590-603, 2001

[18] - S. Lingwal, "Info_PCA: A Hybrid Technique to Improve Accuracy by Dimensionality Reduction", Research Gate, 2021

[19] - N. Venkat, "The Curse of Dimensionality: Inside Out", Research Gate, 2018

[20] - J. Ortiz-Bejar, E. Tellez, et al., "Performance Analysis of K-Means Seeding Algorithms", IEEE, International Autumn Meeting on Power, Electronics and Computing, pp. 1-6

[21] - A. Borji, "Pros and cons of GAN evaluation measures", Science Direct, Computer Vision and Image Understanding pp. 41-65, 2019

[22] - E. Moradi, A. Pepe, C. Gaser, H. Huttunen, J. Tohka, J. Tohka, "Machine learning framework for early MRI-based Alzheimer's conversion prediction in MCI subjects", ScienceDirect, NeuroImage pp. 398-412, 2015

[23] - T. Wang, S. Xiao, Y. Liu, Z. Lin, N. Su, , X. Li, G. Li, M. Zhang, Y. Fang, "The efficacy of plasma biomarkers in early diagnosis of Alzheimer's disease", International journal of geriatric psychiatry, Chapter 7, pp.713-719, 2014

[24] - L. Ward, A. Agrawal, A. Choudhary, C. Wolverton, "A general-purpose machine learning framework for predicting properties of inorganic materials", npj - Computational Materials, 2016

[25] - R. Rojas, X. Huang, K. Ou, "A Machine Learning Approach for the Identification of a Biomarker of Human Pain using fNIRS", Nature, Sci Rep 9, 2019

[26] - H. Xue, Q. Yang, S. Chen, "The Top Ten Algorithms in Data Mining", CRC Press, SVM: Support Vector Machines, 2009

[27] - D. Berrar, "Cross-Validation", Research Gate, 2018

[28] - H. Ayaz and B. Onaral, "Analytical software and stimulus-presentation platform to utilize, visualize and analyze near-infrared spectroscopy measures", Drexel University, 2005

[29] - SD Brennan, "An experimental report on rating scale descriptor sets for the instantaneous self-assessment (ISA) recorder", Portsmouth: DRA Maritime Command and Control Division. DRA Technical Memorandum (CAD5) 92017, 199

[30] - P. Piniti, L. Tachtsidis, ect al., "The present and future use of functional near-infrared spectroscopy (fNIRS) for cognitive neuroscience", Annals of the New York Academy of Sciences, vol.1464, 2020

[31] - L. Kocis, P. Herman, A. Eke, "The modified Beer-Lambert law revisited", Phys Med Biology, 2006

[32] - A. Villringer, B. Chance, "Non-invasive optical spectroscopy and imaging of human brain function", Trends in neurosciences 20, 10, pp.435–442, 1997

[33]- G. Welch, "Kalman Filter", Springer Link, 2020

[34] - L. Xu, "Least mean square error reconstruction principle for self-organizing neural-nets", ScienceDirect, Volume 6, Issue 5, pp. 627-648, 1993

[35] - X. Cui, S. Bray, A. Reiss, "Functional near infrared spectroscopy(NIRS) signal improvement based on negative correlation between oxygenated and deoxygenated hemoglobin dynamics", Neuroimage 49, 4 (2010), 3039–3046, 2010

[36] - R. Taylor, "Interpretation of the Correlation Coefficient: A Basic Review", Journal of Diagnostic Medical Sonography, Vol 6, Issue 1, 1990

[37] - R. Brerton, G. Lloyd, "Support Vector Machines for classification and regression", Royal Society of Chemistry, Issue 2, 2010

[38] - Y. Yu, X. Si, et al., "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures", MIT Press Direct, Volume 31, Issue 77, 2019

[39] - L. Yang, A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice", Science Direct, Volume 415, pp. 295-316, 2020

[40] - N. Chauhan, K. Singh, "A Review on Conventional Machine Learning vs Deep Learning", IEEE, International Conference on Computing, Power and Communication Technologies (GUCON), pp.347-352, 2018

[41] - C. Herff, D. Heger, et al., "Mental workload during n-back task—quantified in the prefrontal cortex using fNIRS", Frontiers in Human Neuroscience, Volume 7, 2014

[42] - M. Ghail, G. Ghali, "Mechanisms Contributing to the Generation of Mayer Waves", Frontiers in neuroscience, vol. 14, 2020

[43] - R. Cooper, J. Selb, et al., "A Systematic Comparison of Motion Artifact Correction Techniques for Functional Near-Infrared Spectroscopy", Frontiers in neuroscience, vol. 6, 2012

[44] - L. Rokach, O. Maimon., "Clustering Methods", Springer Link, Data Mining and Knowledge Discovery Handbook, pp. 321-352, 2005

[45] - M. Faisal, E. Zamzami, E. Sutarman, "Comparative Analysis of Inter-Centroid K-Means Performance using Euclidean Distance, Canberra Distance and Manhattan Distance", IOP Publishing, Volume 1566, 4th International Conference

[46] - F. Liu, Y. Deng, "Determine the Number of Unknown Targets in Open World Based on Elbow Method", IEEE, Transactions on Fuzzy Systems, vol. 29, pp. 986-995, 2021

[47] - K. Matsuhima, "The Silhouette Method", Springer, 2020

[48] - A. Kumar, "Elbow Method vs Silhouette Score – Which is Better?", Data Analytics, 2021

[49] - M. Vidyasagar, "Learning and Generalisation: With Applications to Neural Networks", Springer, 2019

[50] - B. Kitchenham, "A procedure for analyzing unbalanced datasets," IEEE, Transactions on Software Engineering, vol. 24, pp. 278-301, 1998

[51] - R. Sherman, "Waterfall Methodology", Science Direct, 2015

[52] - B. Carvalho, C. Mellom "Scrum agile product development method -literature review, analysis and classification", Research Gate, 2011

11- Appendix

Appendix A: machine_learn & deep_learn – Data Organisation

```
out_split = out_kf.split(nirs, labels, groups)
    for k, out_idx in enumerate(out_split):
        print(f'\tFOLD #{k+1}')
        nirs_train, nirs_test = nirs[out_idx[0]], nirs[out_idx[1]]
        labels_train, labels_test = labels[out_idx[0]], labels[out_idx[1]]

        if groups is None:
            groups_train = None
            if train_size == 1.:
                nirs_train, labels_train = shuffle(nirs_train, labels_train)
            else:
                split = train_test_split(
                    nirs_train, labels_train, shuffle=True,
                    train_size=train_size, stratify=labels_train)
                nirs_train, labels_train = split[0], split[2]
        else:
            groups_train = groups[out_idx[0]]
            if train_size == 1.:
                nirs_train, labels_train, groups_train = shuffle(
                    nirs_train, labels_train, groups_train)
            else:
                split = train_test_split(
                    nirs_train, labels_train, groups_train, shuffle=True,
                    train_size=train_size, stratify=labels_train)
                nirs_train, labels_train, groups_train = (
                    split[0], split[2], split[4])

        all_y_true += labels_test.tolist()
```

Appendix B: Machine Learn – SVC – Results of Run Saved

```
            # Get best SVC
            in_average_accuracies = np.mean(in_accuracies, axis=1)
            index_max = np.argmax(in_average_accuracies)
            best_reg = REG_LIST[index_max]
            all_hps.append(best_reg)

            # Fit best SVC on the whole train set
            svc = LinearSVC(C=best_reg, max_iter=MAX_ITER)
            svc.fit(nirs_train, labels_train)
            y_pred = svc.predict(nirs_test).tolist()
```

Appendix C: deep_learn – Retraining to Test HyperParameters with Cross-Fold Validation

```
# Get best hyperparameters
        in_average_accuracies = np.mean(in_accuracies, axis=1)
        index_max = np.argmax(in_average_accuracies)
        best_hps = hp_list[index_max]
        all_hps.append(best_hps)

        # Retrain with best hyperparameters
        clf, results = _train_dl(model_class, n_classes, best_hps[0],
                                 best_hps[1], nirs_train, labels_train,
                                 early_stop=True)
```

Appendix D: Elbow Method

```
            #Finding optimal k - WSS
            if(wss_setting):
                for count in range(x.shape[1]):
                    current_id = cluster_ids_x[count]        #i is the cluster value,
                    wss += distance.euclidean(cluster_centers[current_id], x[example,count,:])

                    #can make a graph for the dis later using wss
                sse.append(wss)
                if len(sse)>=3:
                    wss_decrease_past = sse[len(sse)-3]/ sse[len(sse)-2]
                    wss_decrease_current = sse[len(sse)-2]/sse[len(sse)-1]

                    if(wss_decrease_past>(wss_decrease_current*cut_off)):
                        k_best_wss = len(sse)
                        wss_setting = False
```

Appendix E: Silhouette Score Method

```
            #Finding optimal k - Silhouette value
            if(silhouette_setting):
                sil.append(silhouette_score(x[example,:,:],
                       cluster_ids_x,metric = 'euclidean'))
                if(len(sil)>=3):
                    if(sil[len(sil)-2] > sil[len(sil)-1]):
                        sil.pop(0)
                        silhouette_setting = False           #Stop condition
                        k_best_silhouette = sil.index(max(sil))+3
```

Appendix F: Process For what k Value to use

-Which Method-

```
#Process for what k's to use and what not too
        if(len(sse) >= 1 and len(sil)>=1):
            wss_setting = True
            silhouette_setting = True
            if(k_best_wss == k_best_silhouette):
                print("Same k - Results back each other")
                k_best.append(k_best_silhouette)
            else:
                print("K is not Same - Default to Silhoutte score")
                k_best.append(k_best_silhouette)
        else:
            if(len(sse) >= 1):
                wss_setting = True
                k_best.append(k_best_wss)
            else:
                silhouette_setting = True
                k_best.append(k_best_silhouette)
```

-How to Average-

```
    k_value = round(sum(k_best)/len(k_best))    #finds best average
```

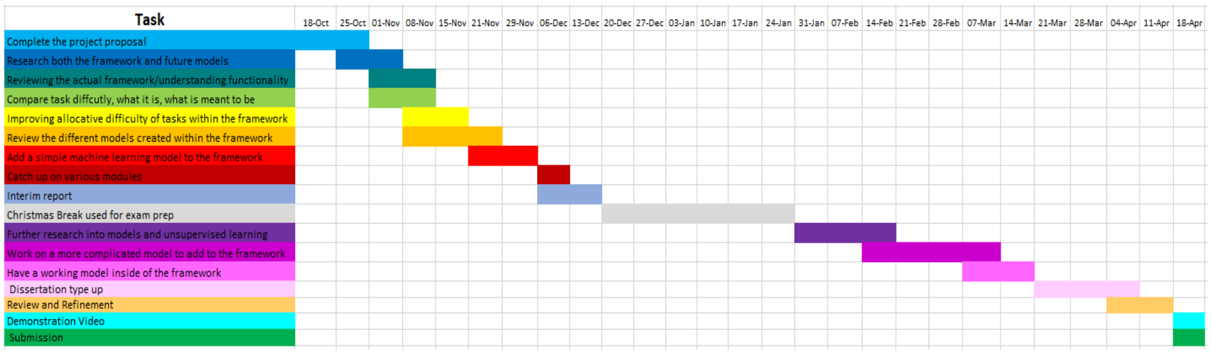
-Mode or Mean Methods-

```
#Mean label calcualtion
    label_mean = []
    for y in range(x.shape[0]):
        label_value = 0
        for i in range(x.shape[1]):         #takes average lable of every 100 time points
           label_value += cluster_ids[i+(y*x.shape[1])].item()
        label_mean.append(round(label_value/x.shape[1]))

    #Mode label calculation
    label_mode = []
    cluster_ids = cluster_ids.numpy()
    for y in range(x.shape[0]):
        label_mode.append(mode(cluster_ids[(y*x.shape[1]):((y*x.shape[1])+x.shape[1])]))

    #Mode or Mean - Also Makes it the Right Data Type
    if Mode:
        for y in range(x.shape[0]):
            y_train[y] = label_mode[y]
    else:
        for y in range(x.shape[0]):
            y_train[y] = label_mean[y]
```

Appendix G: GPU Acceleration Code

```
# set device
    if torch.cuda.is_available():
        device = torch.device('cuda:0')
    else:
        device = torch.device('cpu')
```

Appendix H: Original Gant Chart


Task
18-Oct
25-Oct
01-Nov
08-Nov
15-Nov
21-Nov
29-Nov
06-Dec
13-Dec
20-Dec
27-Dec
03-Jan
10-Jan
17-Jan
24-Jan
31-Jan
07-Feb
14-Feb
21-Feb
28-Feb
07-Mar
14-Mar
21-Mar
28-Mar
04-Apr
11-Apr
18-Apr
Complete the project proposal
Research both the framework and future models
Reviewing the actual framework/understanding functionality
Compare task diffcutly, what it is, what is meant to be
Improving allocative difficulty of tasks within the framework
Review the different models created within the framework
Add a simple machine learning model to the framework
Catch up on various modules
Interim report
Christmas Break used for exam prep
Further research into models and unsupervised learning
Work on a more complicated model to add to the framework
Have a working model inside of the framework
Dissertation type up
Review and Refinement
Demonstration Video
Submission