Automated Assessment on Clinical Drawing Test for Diagnosis and Analysis of Parkinson's Disease using Evolutionary Algorithm

Tian Xia

MSc by research

University of York

Electronic Engineering

August 2019

Abstract

The deadly Parkinson's disease is always a focused area in medical research, even as today there is no cure for such disease. Patient does not have any vivid symptoms until the late stage of the disease when the condition has been threatening patient's life already. Current diagnosing approach on Parkinson's disease at early stage often includes test sets in questionnaire form with verdicts from clinical examiners. Such conventional approach has subjective assessment standard as well as verdicts with examiners' personal judgement, which inevitably may contain human errors. In addition, such assessment method often takes several days for one patient, making it very inefficient. This research project proposed an objective approach by using machine learning to assess one of the tests from the questionnaire – the clinical drawing test, which can classify patients' drawing performance automatically and is very efficient. This approach also allows the algorithm to catch the smallest detail in the drawing whilst minimise human errors from human-orientated assessments. The result proves that, given the same assessment, the algorithm tends to perform better than human verdicts in terms of distinguishing patients in different stages. What's more, the algorithm proposed in this thesis has an overall advantage over some conventional algorithm models, which not only optimised the computational effort, but also can allow clinical experts to understand how the figure data are used by the algorithm and assist them in further research in Parkinson's disease.

List of Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to thank my supervisor Prof Stephen Smith for his great effort on supporting my academic life since the final year of my undergraduate programme, who introduced me to the world of researching in machine learning and medical engineering, as well as providing impeccable support during my short but unforgettable research career. Everything I have achieved so far would be impossible without him.

The author would also want to thank the Department of Neurology, Leeds General Infirmary for their role in investigating neurodegenerative disease and their effort in collecting drawing data from Parkinson's disease patients. This thesis would be impossible without their support.

I would like to show my appreciation to my parent Weilong Xia and Jianhua Wu who provided impeccable support in every aspect throughout my whole University life for these four years. I would also want to dedicate this thesis to them as my final work during my whole aboard studying period.

I also want to thank Dr. Amir Dehsarvi for providing me with impeccable suggestions and encouragement since my undergraduate programme. This research programme would been much more difficult without your assistance.

Special thanks to Dr. Xinwei Gao for giving me valuable advice on programming and his generous effort on my career advice and congratulation on his Doctoral graduation.

Author Declaration

I declare that this thesis is a presentation of original work during 2018/19 MSc by Research period and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

Parts of content in this thesis have been published in the following conference proceedings:

- T. Xia, J. Cosgrove, J. Alty, S. Jamieson and S. Smith, "Application of classification for figure copying test in Parkinson's disease diagnosis by using cartesian genetic programming," in *GECCO '19 Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1855-1863, Prague, Czech Republic, 2019

1. Introduction

People's daily behaviours are controlled by their brain, which is further controlled by the nervous system in our body. However, we often ignore the importance of our nervous system until it is compromised. Any disorder of our nervous system will affect both our health and safety severely. According to the World Health Organization, as of 2006, neurological disorder and their sequelae has affected around one billion people in worldwide [1]. Neurological disorder has a life-changing impact on patients' mental functioning, learning disabilities or intellectual disability. Clinical research has been carried out on this topic to study how neurological disorder can be detected at its early stage, such as questionnaires to measure patients' cognitive impairment, and figure drawing task, which requires patients to complete a specific drawing task in order to assess their cognitive functions. However, the strategy to judge those results is considered subjective as those tasks are often assessed by the examiner's prospective which may include human error by only observing the patients manually.

This project purposes an objective, non-invasive approach to assess Parkinson's disease patients' figure drawing test result by applying machine learning, which is a sub-division of artificial intelligence, to learn the pattern of the figure categories by using supervised learning. Hence, an algorithm which can classify those test results fairly can be generated and improve its performance by training the algorithm as more drawing samples will be generated from such drawing task assessment. In addition, such algorithm can capture small details from the figure, which may not be observable to human's naked eyes, but could be a vital signal of neurological disorder at early stages. Visualisation of the algorithm can also assist clinical researchers to understand how the algorithm uses the drawing data, to further investigate different aspects of the cause and the impact of neurological disorder.

1.1. Clinical Requirements on Diagnosing Neurological Disorder

1.1.1. Overview

Neurological disease can be very deadly. In 2015, neurological disorder caused 9.4 million death globally in 2015, and the death rate reached its peak in 1990, at 36.7%, making it the second-leading cause of mortality [2]. The symptom of neurological disorder varies by the disorder type but no other than mental disorder, memory loss, motor disorder, cognitive impairment, etc., all of which has negative impacts on patient's life quality. Among all of the neurological disorders, the infamous Parkinson's disease (PD) attracts a lot of attention with 6.2 million affected [3] and 117,400 fatalities in 2015 [4]. The average life expectancy is around 7 to 14 years [5]. The impact of PD on several celebrities such as Olympic cyclist Davis Phinney [6] and boxer Muhammad Ali [7] has raised public awareness on this disease.

1.1.2. Parkinson's Disease

The definition of PD is a group of conditions in motor dysfunction [8], which includes primary symptoms such as tremor, stiffness, bradykinesia and postural instability [8]. PD is a long-term neurodegenerative disease as early stage symptoms are very subtle, and it develops gradually over time [8]. Victims of PD are usually over 60 years old [8]. As the condition of PD worsens, non-motor symptoms, such as memory loss and cognitive impairment start to develop [8, 9]. Apart from those symptoms, what makes PD notorious is there is no cure, so current treatment methods tend to focus on improving symptoms and patients' life quality, usually with levodopa combined with carbidopa [8, 10]. However, due to the short life expectancy of PD, the allowed time for symptom improvement may not be enough if the patient is in very late stage of PD. Therefore, a method to diagnosis and monitor PD at early stage is very essential, which could provide more time for further treatments as well as taking precautions.

1.2. Project Motivation

This section gives a description of the clinical requirement on Parkinson's disease and how such disease requires the assistance from artificial intelligence technique.

1.2.1. Parkinson's Disease Effects

In general, PD has three categories of symptoms – motor skills, cognitive dysfunction and memory loss. Motor skills impairment may include slowness of movement, muscular stiffness such as in limbs and trunk and trembling in common part of the body such as hands, legs and face [8]. Cognitive impairment may affect patients' ability in daily tasks such as read or write and recognising items. Memory loss causes patients to forget various items such as people, even close relatives, procedures for simple actions and locations, which may further be developed into dementia, a neurological disease that caused thousands of elderlies to lose their way home. Such memory loss may be irreversible. All of those three aspects can have huge impact on patients and their families' life quality. To minimise the effects of PD, we need to diagnosis and monitor PD from early stages to gain enough time on improving their symptoms and life quality, even though the symptoms of PD at early stage are very subtle.

1.2.2. Current Diagnosing Approach

Currently, there is no definitive test for PD diagnosis, so PD must be diagnosed by certain clinical criteria [11]. In general, there are two common ways to diagnose PD – Pathological examination and questionnaire-based test set.

Research on pathological examination of PD has been carried out for decades with several clinical criteria were proposed. However, most of them are very complex with the involvement of sophisticate medical instruments. In addition, it is believed that pathological examination cannot decisively classify the clinical syndrome [12]. Although it has a complete set of clinical criteria to diagnosis PD at early stage, such diagnosing approach is too complicated to carry out and time-consuming for such disease that develops over time.

Questionnaire-based test set is a non-invasive approach that usually contains several small tests to assess test subject's memory, motor skills and cognitive function. These tests are designed to expose patients' early stage symptoms which are usually very subtle. The downside, however, is the rating of the tests are usually completed manually. Normal procedure requires an examiner to rate the test according to a pre-defined criteria-set. Both the criteria and the manual assessment method are subjective and possibly contains human error or misjudgement. For example, even though the patient can perform a memory test perfectly according to the criteria, patient's time that spent on thinking may contribute to potential threat of PD, while both the criteria and examiner may not be able to assess patient's thinking-time. Such test sets are simple, non-invasive, which can be carried out even in home, but the assessment method is not convincing.

1.2.3. Proposed Diagnosing Approach and Advantages

The aim of the new diagnosing approach that being proposed in this research is to be simple, non-invasive, but also has an objective approach to classify patient's PD stage. This approach adapts a clinical drawing test, which is common in questionnaire-based test set. In general, it requires patient to copy a certain figure by hand on a digitised tablet to test patient's cognitive skills. After a certain amount of time, it requires patient to redraw that figure without any hint and references to test patient's memory functions. Both task tests patients motor skills through hand-drawing.

After the test, patient's drawing is stored on the computer through the tablet in the form of data stream, which record patient's stylus position, tilt information and pen pressure at certain timestamp. This data stream will further be used to produce a data set, which will describe the figure by its features. This data set will be used to train a machine learning algorithm by using features as data input, while patient's PD stage, which is determined by other tests, as output. The training algorithm will adapt supervised learning strategy. With this strategy, algorithm will be provided with inputs and their correspondence output. This will allow the algorithm to find the pattern for all the data set, which enables the algorithm to become an automated classifier to classify patient's drawing objectively.

2. Background Literature

This section expands the content from the introduction section and provides theoretical evidence on the practical aspect of applying artificial intelligence technology on medical area with background knowledge.

2.1. Parkinson's Disease

Historically, PD is widely recognised as a motor disorder without any other side effects [13]. However, it is increasingly considered that PD will affect patients with cognitive impairment, eventually leads to dementia [13]. In this research project, we assess three main symptoms of a PD patient – motor skills, cognitive disorder and memory function, and discuss the downside of current diagnosing method and the counter measurements to compensate this downside.

2.1.1. Motor Skills

The definition of motor skills is the ability to initiate a muscular movement with total control from the initiator, which is further divided into gross and fine motor skills [14]. There are four basic motor symptoms which are inducted by PD: tremor, bradykinesia, rigidity and postural instability [11].

2.1.2. Cognitive Disorders

Cognitive disorders are a category of mental health disorders that affect cognitive abilities such as visual perception, memory, recognition, etc. Physical diseases such as genetics, brain trauma, etc. can cause cognitive disorders. In the meantime, neurodegenerative diseases such as Alzheimer's and Parkinson's can also cause cognitive impairment. Therefore, cognitive disorders alone cannot be a decisive factor in diagnosing Parkinson's disease.

2.1.3. Memory Function

Memory is one of the key functions in the brain that can store and retrieve information from the external environment. Memory loss, or Amnesia, is an impairment in memory functions which may be caused by brain damage or disease [15]. Memory is important to experiences and therefore, information storage can influence the future actions and decisions [16], including skill developments, daily tasks and social relationships [17]. Several factors could lead to memory loss, including physical brain damage and atrophy, a symptom that part of the complete shrinkage of part of the body, which may present with PD [10, 18]. Memory impairment may also be caused by dementia, which in PD, is in very late stage. Therefore, memory loss is not the sole symptom of PD, but PD may cause memory loss at late stage.

Those three symptoms alone cannot be used to identify PD alone; however, the combination of those three symptoms can exclude some common neurodegenerative disease. For example, both Alzheimer’s disease (AD) and PD share the symptoms of movement difficulties and cognitive impairment, but AD is less related with memory loss [19]. PD patients may not have both motor skills deficit and cognitive dysfunction comes together, but AD patients often shows visuospatial and constructional ability deficit with the sign of movement difficulties as those two symptoms are hard to be isolated from AD patients. [20, 21, 22, 23].

2.1.4. Disadvantages of Conventional Diagnosing Method

The mainstream of PD diagnosing can be divided into two tracks – Imaging and Questionnaire assessment. In imaging, one of the most commonly used technology, computed tomography (CT), usually appears normal when scanning PD patients [24], while magnetic resonance imaging (MRI) is more accurate in diagnosis of PD over time [25]. Current research has been investigating on the method of using evolutionary algorithm on classification of fMRI imaging of PD patients [26]. However, imaging requires professional clinical equipment and often companied by potential health risk on the patients by the exposure to radioactive beam and, in some scenarios, not applicable to certain patients with exceptional circumstances that exclude them from using radioactive beam.

Questionnaire-based diagnosis addressed those issues by asking patients to answer and/or perform certain simple questions and tasks. Famous assessments method such as Montreal Cognitive Assessment (MoCA) and Unified Parkinson's disease rating scale (UPDRS) are widely recognised by clinical experts worldwide and are proven that they can distinguish patients and normal cognitive. In MoCA, with 30 marks available, people without cognitive impairment scored an average of 27.4 while mild cognitive impairment has a lower average of 22.1, with even lower average of 16.2 by AD patients. However, concerns raised over the difficulty and the final marking of the tests. In January 2018, then President of the United States Donald J. Trump has taken a MoCA test during regular health check-up and scored full marks of 30/30 [27, 28]. However, the difficulty of the MoCA is questioned with the speculation that Trump has shown some symptoms of early stage of dementia when he claimed that his father, Fred Trump was born in Germany while Fred is actually born in the US [29, 30, 31], and stumbling on word of 'origin' with 'orange' [31].

Another concern on such test is the fairness of its marking scheme. Because the result fully relies on the accumulative marks which is usually done by clinician according to a set of criteria, this marking method is considered subjective, which is under discretion of the marker. The criteria itself is considered incomplete in testing cognitive disorder patients as most of the components in the test have marks ranging only 0 or 1, with maximum range of 0 to 5, which is highly inaccurate in terms of detecting details of patients' movement, cognitive skills and memory function as those are developed gradually. Therefore, based on current diagnosing method, a simple-in-form but with certain difficulty to complete and objective assessment criteria method is needed to compensate current diagnosing approach.

2.2. Artificial Intelligence

The definition of artificial intelligence (AI) is the intelligence manipulated by machines, mostly computers. Key point of AI is the demonstration of properties from human intelligences by machines, such as 'learning' and 'problem solving'

[32]. AI research is divided into subfields based on technical considerations or particular tools. Problem-solving AI may be in the form of automation, using a pre-set program to solve a set of problem automatically. Learning AI requires AI to find a solution for a problem by learning by itself, one of the fundamental concepts of learning AI research is machine learning [33].

2.2.1. Machine Learning

Machine Learning (ML) is a research topic on the algorithms and statistical models which allows computer system to accomplish a task by 'self-learning'. When implementing ML, people only need to define the learning target by providing the algorithm with sample data, known as 'training data', so that the algorithm will attempt to generate a mathematical model that can fit as many of those training data as possible, without explicitly interfere the programming of the algorithm for specific task [34].

In ML, there are two training strategies – supervised and unsupervised learning. Supervised learning requires all training data are labelled for the ML algorithm to find pattern to classify all those data according to the labels. Each sample data is a pair with inputs and expected output value (or 'supervisory signal'). This training strategy allows the ML algorithm to predict the output value from an unseen data pair, therefore, reaching the 'learning' objective of the algorithm. The counterpart of supervised learning in human and animal psychology is concept learning, which is defined as "the search for and listing of attributes that can be used to distinguish exemplars from non-exemplars of various categories" by Bruner, Goodnow & Austin in 1967.

2.2.2. Support Vector Machine

This section introduces the concept of support vector machines and its features as well as the application on the real-life problem. It also discusses the disadvantages of the support vector machine and possible approach to compensate its disadvantages.

2.2.2.1. Overview

Support Vector Machines (SVM), also support-vector networks [35], are mathematical models that can perform supervised learning in machine learning. SVM is mostly used to analyse data for classification and regression analysis. To realize data classification, during the algorithm training, SVM builds the model which can assign new data to one or another category. SVM can also be used for unsupervised learning when the data is not labelled by using the support-vector clustering algorithm [36]. This algorithm applies the statistics of support vectors to categorise unlabelled data.

The formal definition of SVM is a constructed hyperplane or a set of hyperplanes in a multi-dimensional space, which this hyperplane can be used for classification and/or regression [37]. The performance of one SVM model is defined by the maximum distance from the hyperplane to the nearest training data point of any class. The larger margin indicates lower generalisation error of the model, which means less overfitting [38]. Fig.1 shows a sample of a simple SVM model with clearly distinguishable data points.

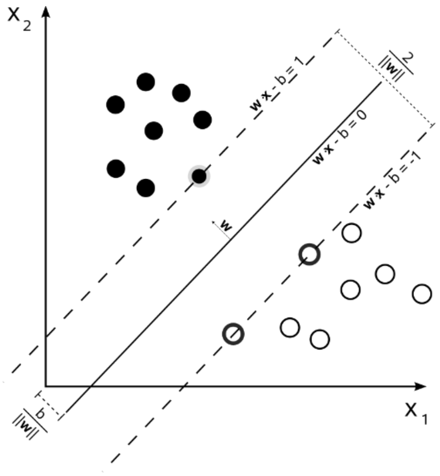

x_2
x_1
$\mathbf{w}\cdot\mathbf{x} - b = 1$
$\mathbf{w}\cdot\mathbf{x} - b = 0$
$\mathbf{w}\cdot\mathbf{x} - b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

Figure 1 - A linear, binary SVM classifier and the optimal hyperplane, taken from [39]

Application of the SVM ranges widely for real-world problems. Hypertext categorisation is one of the key areas of SVM application as the training instances are labelled with their text content. In natural language processing (NLP), a process called semantic role labelling, or shallow semantic parsing, labels the semantic role of words or phrases in one sentence. Such process is entirely based on support vector machines [40, 41]. Classification of images, hand-writing recognition and classification for biological researches are also using SVM [42].

2.3.2.2. Advantages

As a popular and efficient machine learning algorithm, there are large amount of development toolkits that allows people without any knowledge of SVM to implement SVM, and very fast on generating result. Because SVM uses a hyperplane to 'separate' data set from different classes, it will yield great result for data sets with distinguishable features. The model structure is also easy for researchers to optimise current model by several techniques, such as observing the position of the hyperplane and the minimum distance between data point and the plane. This model structure can also be used to investigate the clustering in the selected features, which can be used for PD disease research.

2.3.2.3. Disadvantages

Because the main task of the SVM is to find a hyper-plane in n-dimension that can maximise the minimum distance between the plane and the data point, where n is the number of features from the data, the model visualisation of the final result is very difficult for people without any background knowledge of algorithms and mathematics to understand, while one of our main objectives is to assist clinical experts to study Parkinson's with the help of artificial intelligence technology. As SVM can be easy to use and fast to generate result, it may not be an ideal tool for a non-computer expert to assist their research.

In addition, clustering in the data point is not essential for SVM to find a hyperplane but is one desired characteristic. In this research, clustering is not a guaranteed characteristic from the extracted features. Therefore, SVM may not be suitable for this project. However, SVM will still be used to verify those hypotheses.

2.2.3. Artificial Neural Network

Artificial Neural Network (ANN) is a bio-inspired model that uses computational nodes to 'simulate' biological neuros. In an ANN model, there are three basic layers – input, hidden and output. Input layer simulates any biological part that take information from the outside world, in AI, this corresponds to the features from the data. Hidden layer simulates biological brain, to process the information from the input layer, and finally, output layer simulates the behaviour from biological creatures, in ANN, this corresponds to the result from the model. Fig. 2 illustrated a simplified ANN structure with three basic layers.

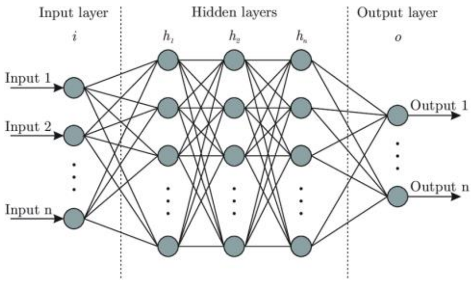


Figure 2 - Illustration of ANN's layer model [43]

A branch of ANN, Deep Learning, is derived as the requirements of AI increases. Instead of taking features as input, deep learning uses multiple layers to process raw input in each layer and extract higher level features from them. This gives the possibility for deep learning to discover rich, hierarchical models [44, 45] that can perform very complex task with high accuracy, such as image recognition, natural language processing [45] etc. However, to reach this performance for an AI model, a very large scale of dataset and high-performance computation (HPC) is required to sufficiently train the algorithm. The author considers current research progress of this project is still in the preliminary stage and is not capable of providing such scale of data set. Therefore, we need to use a light-weight ML algorithm set and using several validation procedures for this research project.

2.2.5. Evolutionary Algorithm

Evolutionary algorithm (EA) is a subset of evolutionary computation [46]. EA is a metaheuristic optimisation algorithm which is based on generic population. EA is inspired by biological evolution. Mechanisms in biological evolution, such as reproduction, mutation, crossover and elimination are all simulated in EA. EA is evolved through numbers of generations of 'chromosome', new generation is generated by evolving the 'parent' chromosomes from the previous generation. The main mechanism to produce offspring is crossover and mutation.

Crossover, or recombination, is a technique to combine parts of parental chromosomes, to form a new chromosome. Such technique attempts to keep the 'good' part from the parent chromosomes that contributes to their 'fitness' and combine them, so that the offspring is expected to be more superior from its previous generation. However, the possibility that the 'bad' part is mixed for the new offspring still exists.

Mutation is also involved during the offspring generation process. The crossover mechanism may have issue where the parent and the offspring have a limited maximum performance, or local maxima, which limits the general evolution performance of the algorithm. Mutation allows a very small portion of the offspring chromosome randomly changes to different value so that it allows the performance of the algorithm is not compromised by the maximum limit. However, mutation cannot guarantee that mutant is better than original chromosome, so mutation can also lead to drop of performance, but this can be compensated by further evolution.

The form of the chromosome is not limited, but often defined by the specific type of EA, those forms include but not limited to binary, mathematical equation and encoding of the visualisation of the algorithm. The process of an evolution of EA is as follows:

1. Generate initial population of chromosomes with a random seed;
2. Select the chromosomes as parent by evaluating their fitness against a pre-set fitness function;
3. Offspring is generated through crossover and mutation from parent chromosomes;
4. New individuals replace the least-fit population;
5. The remaining chromosomes form a new generation;
6. Repeat Step 2-5 until the optimum chromosome is generated or the maximum generation number is reached.

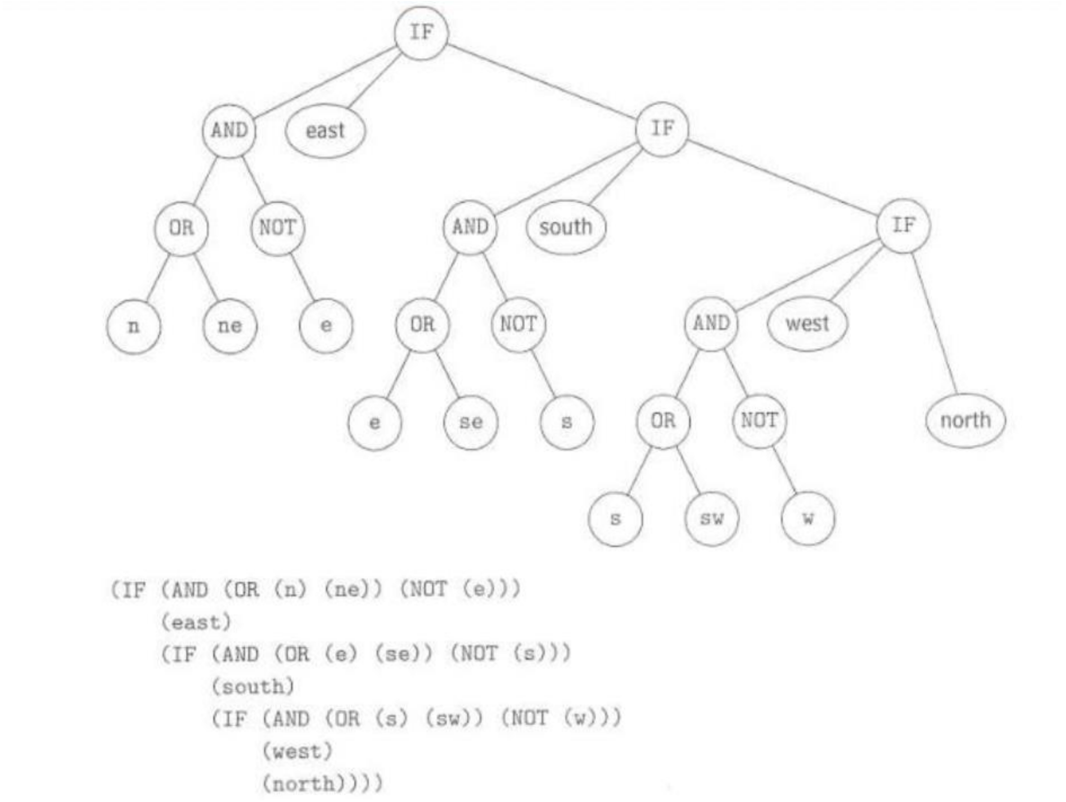
Apart from the general form of the EA, there are many subsets of EA that is designed for various type of problems. One of them, genetic programming, are designated for computer programs to solve computational problem.

2.2.6. Genetic Programming

This section introduces one forms of evolutionary algorithm – genetic programming, and its advantage on data classification problems as well as its disadvantages due to the genetic programming's nature structure.

2.2.6.1. Overview

Genetic programming (GP) is a sub-division of evolutionary algorithm. The name of 'genetic' comes from the representation of the algorithm – each algorithm model is represented as a chromosome. A group of chromosomes form a generation. This algorithm is evolved by crossover and mutation. Crossover indicates the combination of two parent chromosomes to form an offspring chromosome, while mutation means a random change in one sole element in the chromosome. Conventional genetic programming is represented in the form of binary tree, which will expand as the algorithm model evolves. Each tree node indicates either value data or mathematical operation indicator, therefore, each chromosome can be described as a mathematical equation as well. Fig. 3 shows a simple robot logic that represented in binary tree, which can be used as a GP model as well.



```
(IF (AND (OR (n) (ne)) (NOT (e)))
    (east)
    (IF (AND (OR (e) (se)) (NOT (s)))
        (south)
        (IF (AND (OR (s) (sw)) (NOT (w)))
            (west)
            (north))))
```

Figure 3 - Example for a perfect wall-following robot program in LISP [47]

2.2.6.2. Application on Data Classification

Since each element in the binary tree node is either data or mathematical operation indicator, the output node must be a data value. The number of output nodes is not limited as it should accommodate with the fitness function. This property of GP makes it applicable in data classification problems due to its input-output form. Input nodes are data inputs and the output can be used for classification. The standard for classification is called fitness function where it will decide whether the output matches the expected class, hence evaluate the fitness, or the performance of the chromosome.

2.2.6.3. Downside

As mentioned in section 2.2.6.1, the binary tree for a chromosome will expand along with algorithm evolvement. Real-life evolutionary mathematical problem may require thousands of generations' evolution. The fitness of the offspring chromosome may improve as the number of nodes increases, but the computational effort for such chromosome increases dramatically. Also, not all nodes are necessary used, but unused data will still use computer system's memory. In conclusion, the evolution speed drops along with the evolution, which leads to more nodes are created and more time on reaching the final result. Fig.4 shows an offspring that has more complexity than its parental chromosomes'.

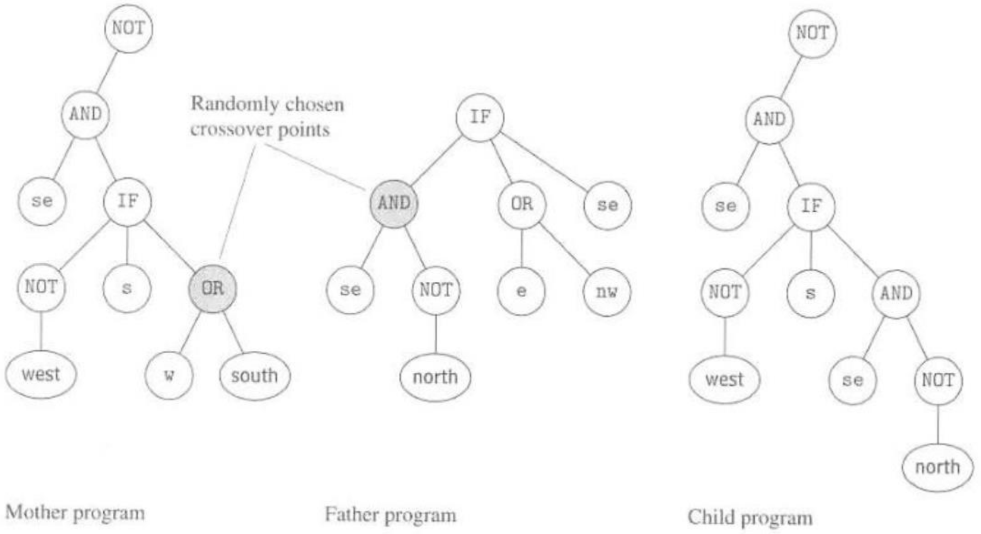


Figure 4 - Example of GP chromosome uses crossover to generate offspring with the model expands [47]

2.2.7. Cartesian Genetic Programming

This section introduces another form of genetic programming – Cartesian Genetic Programming, and how its unique structure addresses GP's issue while maintaining GP's core features.

2.2.7.1. Overview

Cartesian Genetic Programming (CGP) is an alternative form of GP. The name 'Cartesian' is derived from its grid form of nodes, which addresses the redundancy issue from GP, which will be further discussed in section 2.2.7.2. CGP's evolution strategy is different from GP's, CGP evolves only through mutation. The mutation rate is configurable, typically ranging between 0-10%. The arity of each node, or the number of connections allowed for each node, is also configurable. Each chromosome in CGP indicates the arrangements and connections between each node as well as the content in those nodes, which can also be represented as a mathematical model. The unique grid form of CGP compensates the major issue from the GP.

2.2.7.2. Advantages over Conventional Genetic Programming

As the GP expands along with the evolution, CGP uses a fixed number of nodes, which also accommodate that it evolves only through mutation. The chromosome evolves by changing its nodes arrangements, connections and content, as well as the allocation of input and output nodes. The content in the node is mathematical operation indicator only, and the calculated values travel through the nodes. In the end, the computational resources of CGP never increases as no new elements are being added to the model along with evolution. This allows CGP to be trained for almost unlimited amount of generations without memory usage issue as well as computational speed issue, which potentially leads to better training result with much more chromosomes are generated. Fig.5 shows a simple illustration of CGP structure and actual CGP model that comes from a training session.

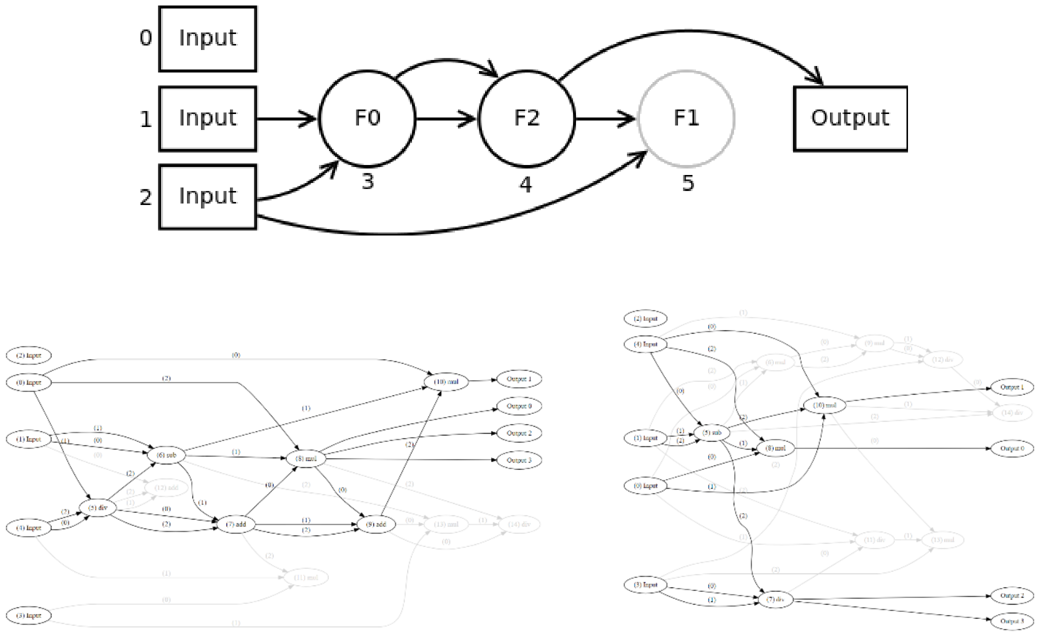


Figure 5 – Sample of CGP structure, taken from, [48] and illustration of CGP evolution, notice the size of the model remains unchanged

2.2.8. Previous Studies on Medical Engineering using Machine Learning

Preliminary research has been done on applying machine learning on medical engineering, especially diagnosis and monitoring of neurodegenerative diseases. In 2018, Gao et al. [49] conducted a study on using evolutionary algorithm to derive a classifier that can measure the severity of bradykinesia in PD, and 'had a potential to differentiate early stage PD from normality' [49]. In their study, a finger-tapping test was conducted on PD patients which requires PD patients to tap their thumb and index finger as quick as possible. A pair of simple measurement devices was installed on the two fingers to measure the altitude between them, as shown in Fig. 6 [49]. The output data from the devices was used to evolve a classifier using EA to measure the PD severity along with existing Parkinson's disease cognitive test. The result shows that such simple test and classifier achieved at least 89.7% accuracy in bradykinesia severity detection [49]. Fig.6 is the device that used from that research project and sample illustration of data measurement.

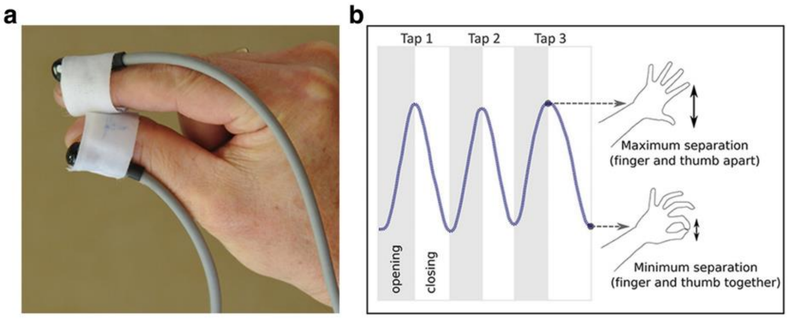


Figure 6 – Illustration of data collection devices on test subject's fingers, taken from [49]

In this thesis, a similar approach is used to adapt a simple test that can assess PD patients' feature abilities and classify their test result by using EA evolved classifier and the raw data from the test. This thesis will also study the correlation between the test and some existing cognitive test sets, to prove the effectiveness of the proposed test on detection of PD severity. The proposed test in this thesis will also assess PD in different aspect, from explicit symptoms to the function in the central nervous system.

3. Methodology

This section discusses the methodology for this research project, including how the data from the patients are collected and how recruited test subjects are evaluated. This section also covers the test that conducted on the test subjects – the clinical drawing test, which is the core of this research project.

3.1. Data Collection

Data collection is performed by using digitised tablet which can track the movement of the stylus and record data with a certain sampling rate. Test subject is required to perform a certain figure drawing task by using the stylus and drawing on the tablet. In addition to minimise environmental impacts, such as the handling of the stylus and tablet's effects on test subject's performance, we put a sheet of paper on the tablet, as well as a modified stylus which can feed ink onto the paper, to give the closest feeling of actual drawing whilst the data can be collected. Fig. 7 shows the equipment used to capture test subject's hand drawing data.

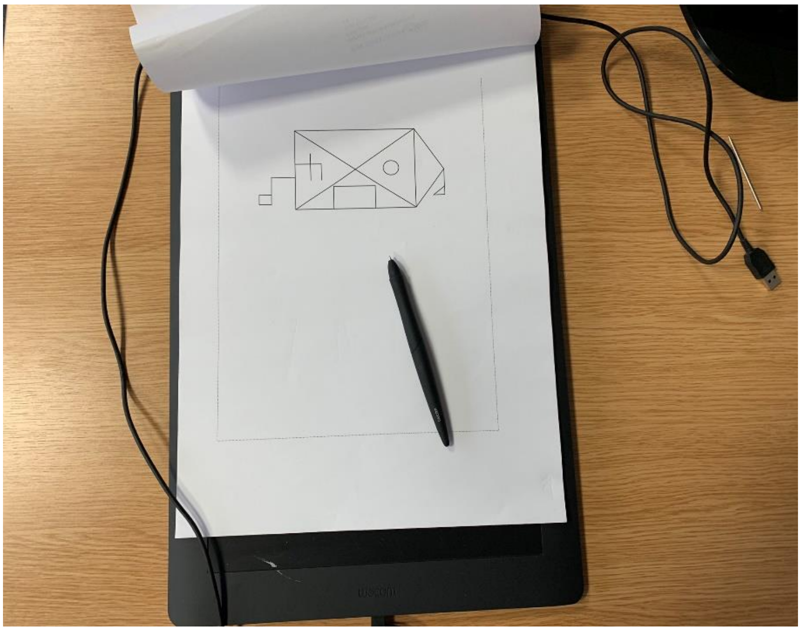


Figure 7 - Device that used to collect drawing data from patients

The tablet can capture following data: stylus position, stylus tilt information, and pressure from the stylus to the tablet. Each drawing will generate approximately 4,000 ~ 5,000 lines of raw data, from which we can extract features which can reflect several neurological disorders that are caused by the Parkinson's disease.

3.2. Test Subjects

Test subject includes 29 people for control group with no signals of Parkinson's disease, and 58 Parkinson's disease's patients with three different disease stages: PD-NC (Normal Cognitive), PD-MCI (Mild Cognitive Impairment) and PD-D (Dementia), in the order of severity. The classification is done by using several cognitive questionnaires, which will be explained in section 3.2.3.

3.2.1. Demographics Information

In protection of test subjects' privacy, all sensitive data are removed and not collected when this research carries out. The only known demographic data are age, sex, dominant hand and disease duration. Hand preference is disregarded as the motor skill impairment from the PD will affect both hands. Gender is also not considered as there's no indication shows that PD has significant development trend in certain gender. However, age and disease duration are important information as the research is aiming for PD early stage's detection and monitoring.

Control group consists people from 50 to 79 years old, with an average of 66.01 years old. PD patients' age ranges from 44 to 85 years old, with an average of 69 years old. Among them, the shortest disease duration is 0.5 years while the longest is up to 20 years, with an average of 6.2 years.

3.2.2. Previous Experiment Result

Before the project is carried out, several cognitive tests have been done on both patient group and control group. Table 1 shows the list of tests that performed on patient and control group.

Table 1 - List of cognitive tests carried out on the test subjects

Test	Description
Benson Figure Copy/Recall	A figure task which consists copy and recall
Pentagon	A figure task which requires subject to draw a pentagon
Unwired Cube	A figure task which requires subject to draw a non-transparent cube structure
Total Trails	A figure task which requires subject to trace a certain figure
Montreal Cognitive Assessment (MoCA)	A test set which includes tests on motor skills, memory and cognitive.
Unified Parkinson's Disease Rating Scale (UPDRS)	A questionnaire-based test set which tests the subject in several aspects
Clinical Dementia Rating (CDR)	A numeric scale to measure dementia symptom's severity

3.2.3. Parkinson's Disease Stages

Patients were classified into three stages – PD-NC (Non-Cognitive Impairment), PD-MCI (Mild Cognitive Impairment), and PD-D (Dementia), in the order of ascending in severity. The classification is done by the patient's MoCA and CDR score, the classification scheme is defined by M. Emre et al [13], and I. Litvan et al [50], and is further summarised by J. Cosgrove [51] in table 2.

Table 2 - Criteria of PD stage classification [51, 52]

Criteria	**Classification**
MoCA > 26	PD-NC
MoCA <= 26 && CDR < 1	PD-MCI
MoCA <= 26 && CDR >= 1	PD-D

In table 2, MoCA is used to distinguish normal cognitive patients and patients with cognitive impairment, while PD-MCI is distinguished from dementia patients by CDR interview.

3.3. Clinical Drawing Test

This section will introduce the clinical drawing test, which is widely used by clinical experts in neurological dysfunction diagnosing, and a certain type of drawing test – the Benson Figure Test.

3.3.1. Overview

The first available source on application of drawing test for healthcare is in 1958 by E. F. Hammer [53]. According to Hammer, projective drawings were widely used on children for various purposes [53]. For example, Draw-a-Person test (DAP) was developed by F. Goodenough in 1926 and later involved by K. Machover in 1948 [54], for evaluating children's intelligence. Hammer discussed the possibility on applying those projective drawing tasks for clinicians 'as a diagnostic aid and as an adjunct to psychotherapy' [53]. Fig. 8 shows the initial idea of drawing test that illustrated a smiling person shape using minimal number of segments, taken from [54].

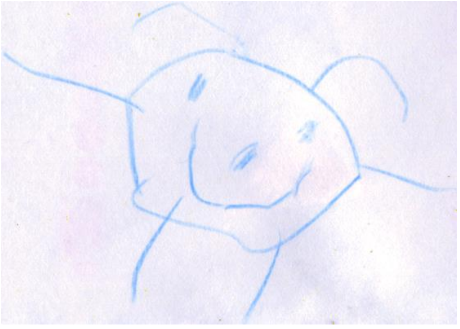


Figure 8 - Smiling person by a 4.5-year-old child [55]

The backbone of clinical drawing test is to provide an assessing method that can be performed naturally by the test subjects, in the meantime, this certain method can easily distinguish test subjects' performance through the drawing quality. In addition, we need a test that can assess some key aspects of Parkinson's disease specifically, which may be consisted by several geometric shapes.

3.3.2. Benson Complex Figure

Figure drawing task is often used for visuospatial assessment for dementia evaluation, but the effectiveness and performance varies on multiple factors, such as target test subjects, figure complexity and structures etc. [56]. Benson figure was developed by F. Benson, it is a simplified version of Rey-Osterrieth Complex Figure (ROCF), which is widely used for visuospatial ability and memory test for neuropsychological evaluation since 1944 [57] and standardised by the National Alzheimer's Coordinating Center [58]. Fig. 9 shows the sample figure of ROCF and Benson figure.

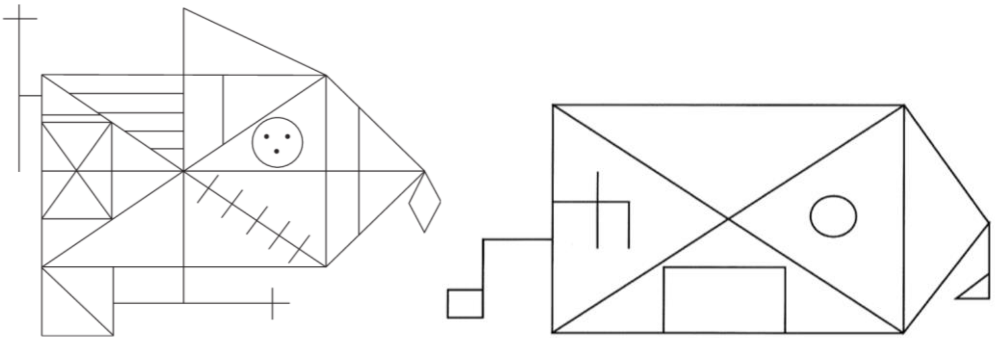


Figure 9 – Rey-Osterrieth Complex Figure (left) [57] and Benson figure (right) [56]

3.3.3. Comparison with other Clinical Approved Drawing Test

This section compares the Benson figure test with other two clinical drawing test that are widely recognised by clinical experts – Clock Drawing Test and Rey-Osterrieth Complex Figure. Through comparison, this section discussed the difference and common point across all three figures and the reason to choose Benson figure test for this research topic.

3.3.3.1. Clock Drawing Test

Clock Drawing Test (CDT) is widely recognised and applied by clinical experts in assessing neurodegenerative disease. During the task, examiner provides a time to the test subject when asking the test subject to draw a clock with that time provided. Because the clock shape is widely known to everybody, a single test is enough to test patient's cognitive skill, motor skill and memory, while Benson test needs two separated tasks to test all three aspects. Fig. 10 shows an example of the classification standard of clock drawing test based on the drawing quality.

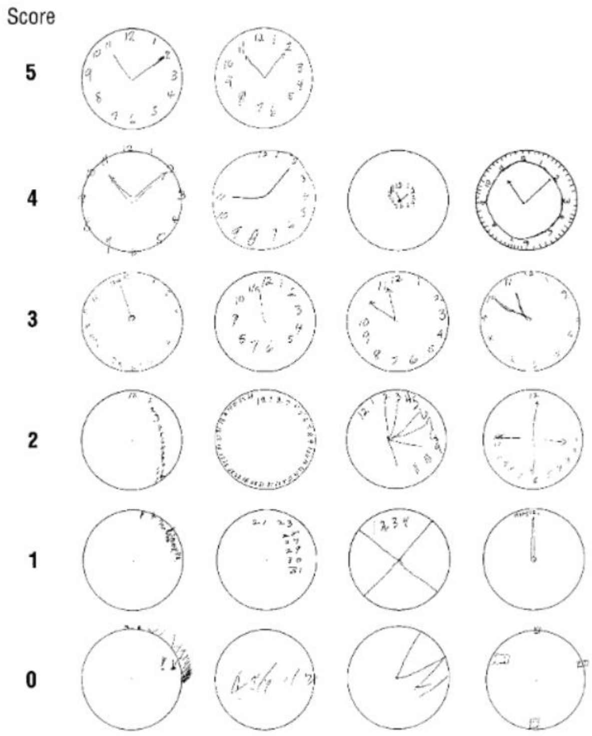


Figure 10 - Clock drawing test result varying in different drawing quality, taken from [59]

This test meets the requirement of assessing neurodegenerative disease, but it comes across various difficulties in combining it with figure digitisation and feature extraction. Firstly, the clock figure is mainly composed by curves and numbers. It's very difficult to set criteria for evaluating the features that extracted from those curves, while the number shape is a combination of different curves.

The second issue on applying CDT with digital data is the importance of component position. Component position is very important in clock drawing test as each clock should have a universal template. With the mixtures of curves, and potential worse scenario where the figure is distorted, which is highly possible as one of the main symptoms of PD is motor impairment. It's very difficult to recognise individual components and calculate its relative position with regard to the outer circle of the clock. Given the limited timeframe of this research, this drawing test is disregarded.

3.3.3.2. Rey-Osterrieth Complex Figure

While CDT's downside is more related on data extraction, Rey-Osterrieth complex figure (ROCF) has a disadvantage due to its complexity. ROCF is mainly composed by line segments and small portion of curves, which is similar from Benson structure. The method to extract features from a digitised Benson is also applicable for ROCF. However, test difficulty and data complexity are main issues of ROCF. Fig. 11 shows the three core layers of the ROCF. Those three layers not only play an important role in constructing the ROCF figure, but also a good reference for both tester and examiner to draw and assess the ROCF figure.

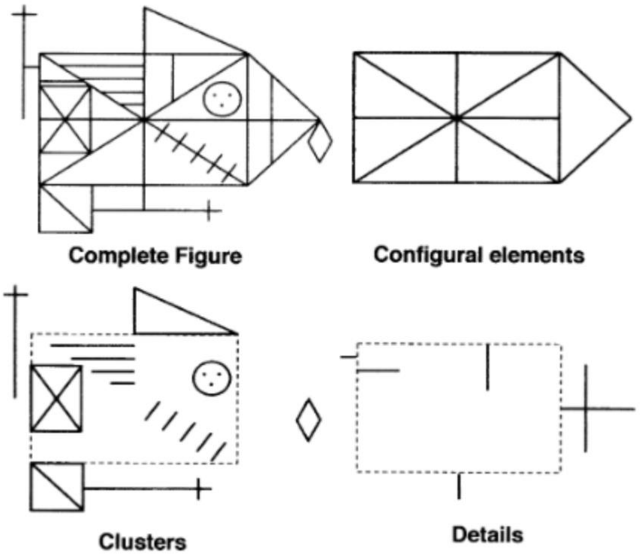


Figure 11 - Illustration and disassemble of Rey-Osterrieth complex figure, taken from [60]

Being similar to Benson test, ROCF also requires two separate copy and recall test to fully assess the patient. One of the main issues is the figure is too complex for both control group and patient group, even normal young people will find difficulties in complete this task set, especially in recall task. Even without applying figure digitising for ROCF, the test itself cannot distinguish patients with different stages efficiently. Fig. 12 shows a side-by-side comparison between drawings from two children with cognitive impairment and normal cognitive abilities [61].

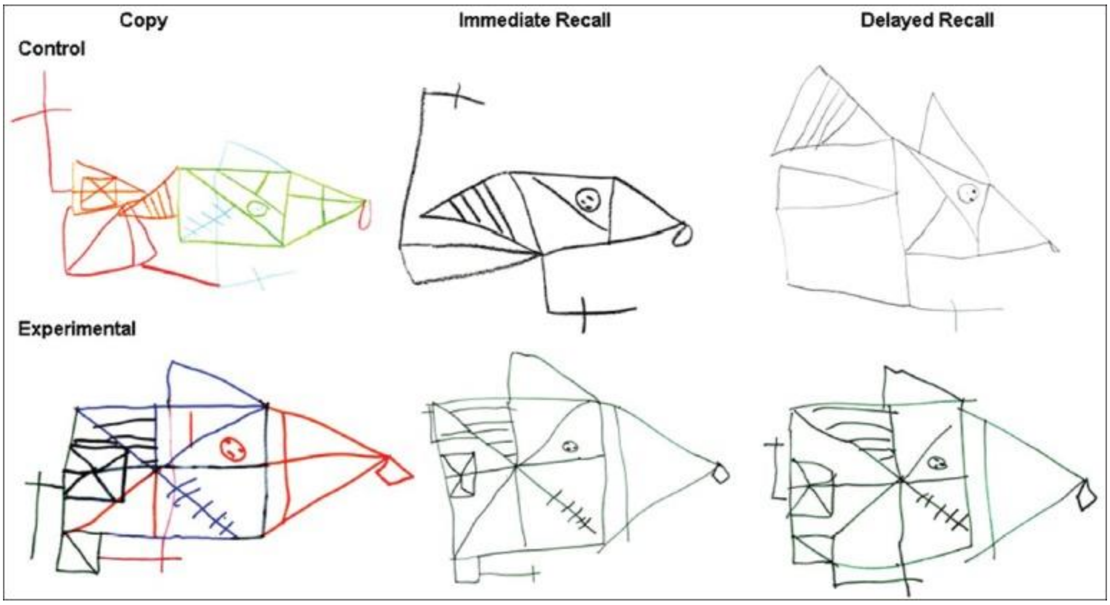


Figure 12 - Copy and Recall result of ROCF, showing figure complexity, taken from [61]

We also estimate that due to its complex structure, the data complexity may cause underfitting, meaning the algorithm will face difficulties in finding patterns to classify all the data, which will lead to failure of algorithm training. The test difficulty also contributes to the potential underfitting as the figure will be similarly distorted between normal people and patients' drawing, therefore the data is not significantly different between control group and patient groups.

In conclusion, Benson's simplified structure compares with ROCF and its line segment structures compares with CDT addresses both downside of current two popular clinical drawing test pattern, and it is ideal for this specific research project.

3.3.4. Correlation with Cognitive Test Set

This section compares Benson figure test with some recognised cognitive test set, the relationship between those and how Benson figure test can substitute or compensate those test sets.

3.3.4.1. Montreal Cognitive Assessment

Montreal Cognitive Assessment (MoCA) was developed as a tool to assess patients with neurodegenerative disease [62]. The main target subject for this tool is for those who present with mild cognitive impairment (MCI) and has a normal performance rating in another cognitive test called Mini-Mental State Examination (MMSE) [62]. MoCA was conducted on the 58 PD patients for this research topic prior to this research. MoCA tests the test subjects from eight disciplines: visuospatial ability, naming, memory, attention, language, abstraction, delayed recall and orientation [63].

For this research topic, we only focus on three disciplines from PD: motor skills, memory and cognitive function. Full MoCA was applied on all test subjects, with few MoCA tasks are separately picked out to investigate its correlation with Benson test.

Table 3 - Tasks from MoCA test set for investigating its correlation with Benson test

MoCA Task	**Description**	**Marking scale**
Cube copying	Patient will copy a cube figure which is given on the form	0~1
Clock drawing	Patient will produce a clock figure with a specific time	0~3
Trail making	Patient will connect numbers of dot in specific order	0~1
Delayed recall	Patient will recall a set of words without any instructions	0~5

Because one of the criteria to define PD stage is MoCA total score, it is meaningless to observe the total test score distribution within the patient group. However, if we separate individual test items from the MoCA test, we can observe that the distribution of the individual test result scores across all PD patients with different stages shows that those tests can detect patients' PD severity very well. Fig. 13-16 shows the experiment result prior to this project from individual MoCA components, which shows vivid correlation between MoCA test and PD condition.

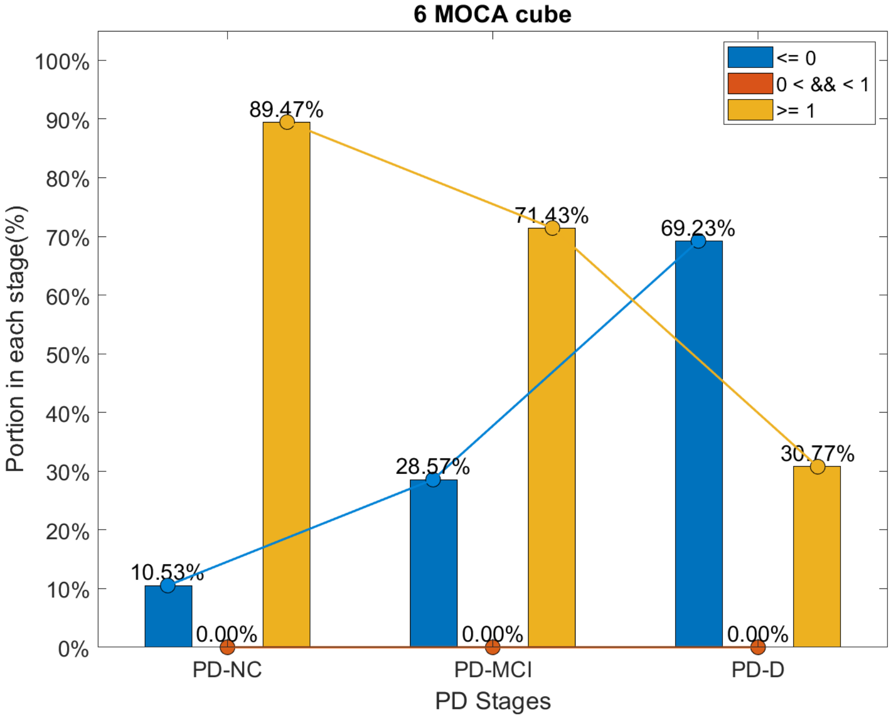


Figure 13 - Distribution of MoCA cube test in all PD categories

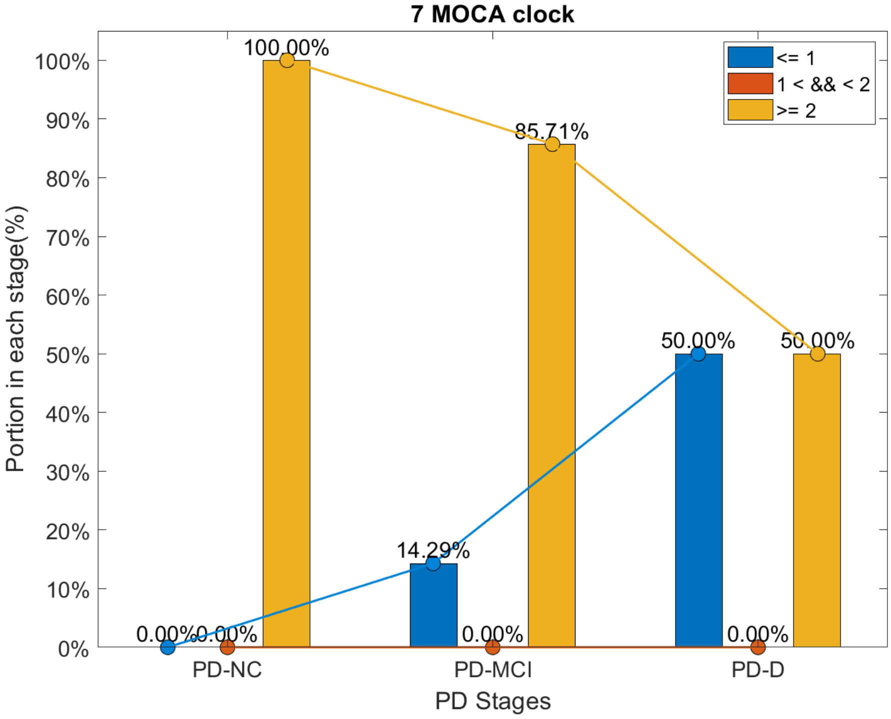


Figure 14 - Distribution of MoCA clock test in all PD categories

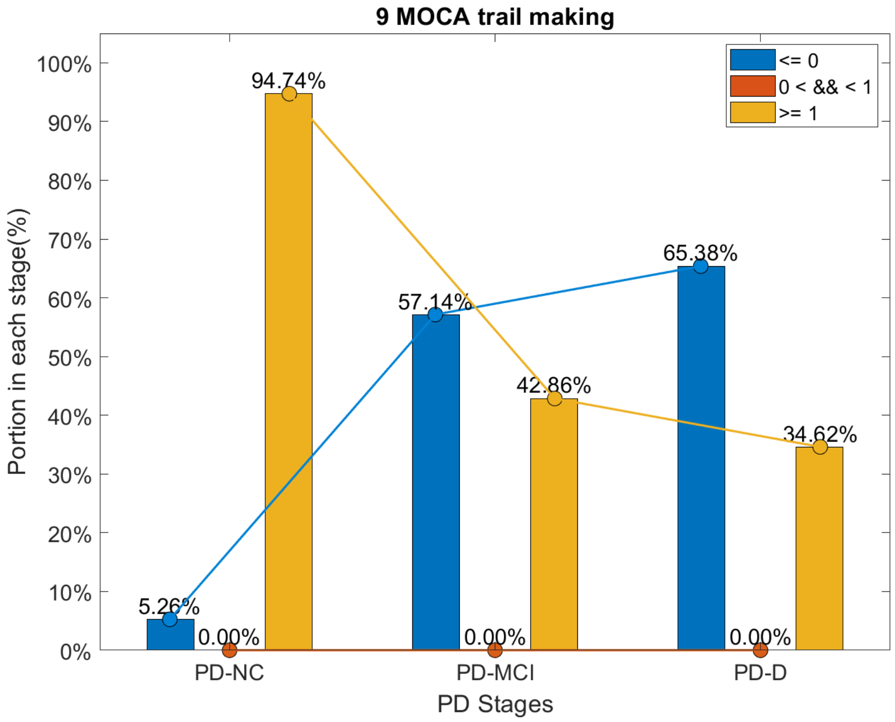


Figure 15 – Distribution of MoCA trail making test in all PD categories

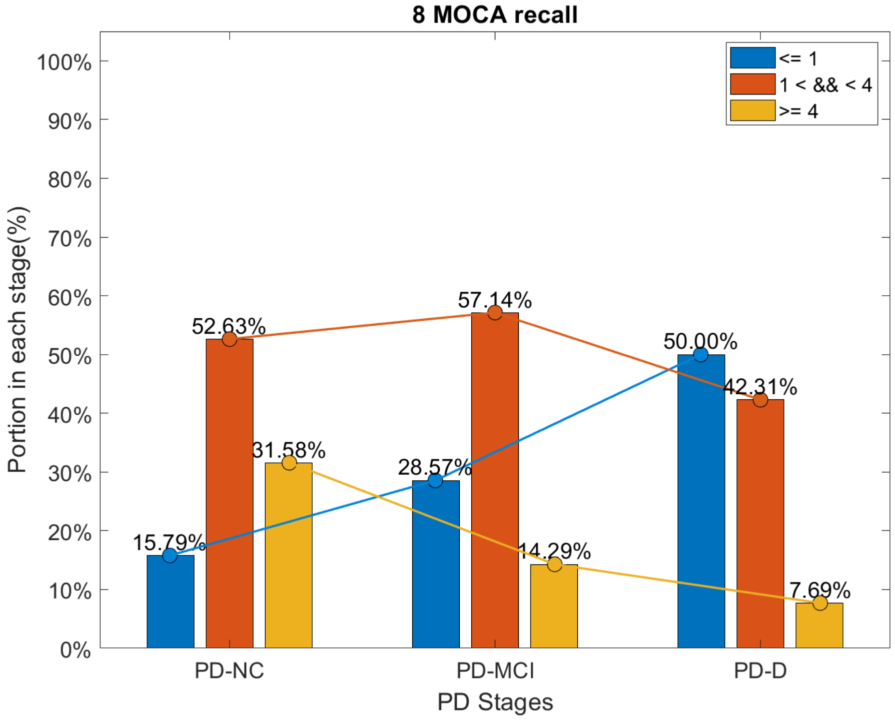


Figure 16 - Distribution of MoCA recall test in all PD categories

Across all four tests, lower mark indicates poorer performance, and vice versa. The portion of the number of test subjects with lower performance increases as the PD stage advances. MoCA cube, clock and trail making tests all assess test subjects' cognitive skills by visuospatial ability and motor skills by asking test subjects to draw a specific shape, while MoCA recall and MoCA clock test their memory function. Because MoCA cube, clock and trail making all requires test subject to perform drawing action, the score from those three tests can be combined, the distribution result shows similar result from the individual tests.

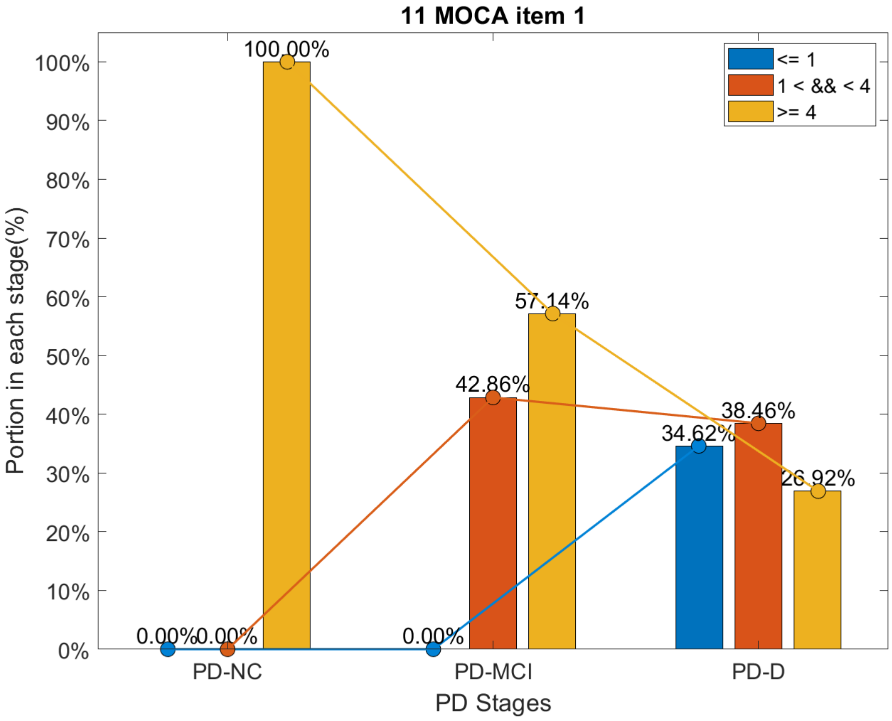


Figure 17 - Distribution of MoCA drawing related test score in all PD categories

With maximum score of 5, Fig. 17 shows that all PD-NC test subjects scored 4 or more, while less than 50% of PD-MCI test subjects achieved score 2 or 3. More than a third of PD-D test subjects achieved 1 or 0 from the combination of three test scores.

3.3.4.2. Unified Parkinson's Disease Rating Scale

Unified Parkinson's Disease Rating Scale (UPDRS or UPD rating scale) is a rating system that is used to monitor patient's PD situation, which is the most commonly used rating scale in PD's clinical study [64]. The name 'Unified' indicates that this scale consists various other rating scale that used to measure PD severity. UPDRS is made up of six sections, from interview on daily life basis, theory evaluation, to compilations of rating scales. In this project, patient's Benson score will be used to compare with their UPDRS score, to find its correlation with UPDRS. Fig. 18 shows the overall distribution of UPDRS score across all three stages.

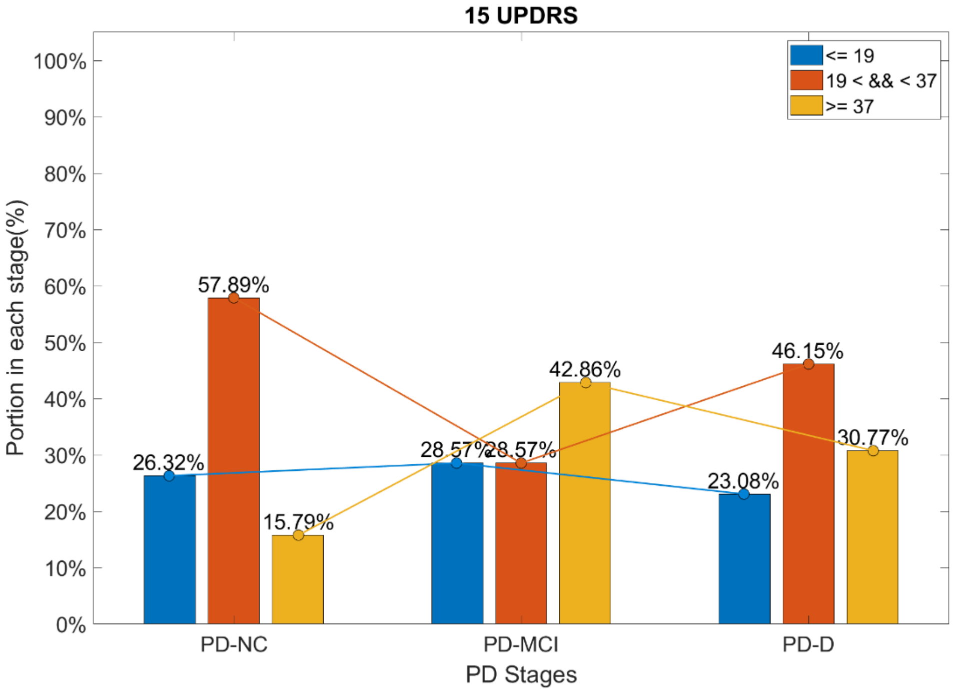


Figure 18 - Distribution of UPDRS scores across three PD stages

However, as UPDRS consists test from various disciplines, it is very hard to find the trend from Benson total score on total UPDRS score as different PD stages has different performance on different discipline. From Fig. 18, portion of higher tier test result in PD-D is even higher than PD-NC, which further shows the necessity of a simple, non-subjective method to diagnosis Parkinson's disease.

4. Algorithm Model Implementation

This section discusses how CGP and SVM are configured for the upcoming classification task.

4.1. Cartesian Genetic Programming

This section describes the practical aspect of CGP, including implementation, configuration, training strategy and fitness function.

4.1.1. Implementation

The implementation of CGP is mostly based from the CGP Library, developed by Andrew J. Turner [48]. It includes all basic functionality that CGP requires. Because the CGP Library is written from C, therefore, the main program for algorithm training uses C as programming language for easier implementation.

4.1.2. Configuration and Parameter Tuning

CGP Library is a highly customisable program API. For this research project, the following parameters will be set prior to each training session:

Number of Input/Output Nodes

This parameter set will define the number of input and output nodes. This is usually determined by the number of features extracted from the raw data, and the number of outputs is determined by the applied fitness function in a classification session.

Nodes

This parameter defines the maximum number of nodes that exists in one CGP model. Each node has the same number of arities, but the node function and their connection nodes may differ. The model may not use every single node, each chromosome may decide the arrangement of active and inactive node. Typical value for number of nodes is ranging from 20 to 100. This usually is defined by the complexity of the problem and may need to accommodate the test machine's specification.

Arity

This parameter defines the number of inputs that each node has. Both the number of node and arity will affect the number of possible arrangements in CGP structure of a chromosome. However, the effect from the number of arities is limited by the node function's designed number of inputs. For example, some node functions only accept up to two inputs, so that three or more arties will not affect the behaviour from this node.

Node Function

This parameter set defines the available function options for CGP nodes. These usually are numerical operations such as adding, subtraction, multiplying and dividing, logic operation such as AND, OR, NOR, XOR etc. Custom node function can also be defined upon the developer's requirement.

Mutation Rate

This parameter defines the possibility that each node in a CGP model can mutate throughout each generation. Because the evolution of CGP relies on mutation completely, mutation rate needs to be set carefully to ensure that the CGP can stably evolve whilst has the ability to jump out from local optima. Typical value for mutation rate should be less than 10%.

Random Number Seed

This parameter is used to randomly generate the first generation of chromosomes of CGP. The random seed is widely used for data cross validation as it defines the evolution behaviour rather than the basic properties of CGP. Therefore, different random number seeds are used to verify the parameters that describes a CGP model.

As most of the parameters are determined by the data model and problem requirement, it is very difficult to list an accurate range of numeric value for those parameters. In Chapter 7 and appendix where the detailed training result is presented, the exact parameter value will be presented for reproducing those results.

4.1.3. Training Strategy

Before the algorithm training, all data will be pre-processed with class labels marked for each data. The whole dataset will be divided into three parts – training, validation and testing data set. Training dataset will be used for the algorithm training, validation and testing dataset will test the chromosome with a 'blind-folded' scenario to the training chromosome.

Because of the small amount of data we have acquired for this study, K-fold cross validation will be applied throughout the training to ensure that each data will be used in training, validation and testing at least once in the CGP training. Different number seed with the same CGP configuration will be used for multiple training as well, as a measurement of data validation.

Multiple fitness functions will be applied for the same dataset and CGP configurations to compare performance of different fitness functions, which will be covered in section 4.1.4.

4.1.4. Fitness Functions

This section introduces several fitness functions that can be applied to the CGP and their main features.

Simple Threshold Classifier

Simple Threshold Classifier (STC) is based on number comparing. STC requires CGP to be configured as single output. Usually all data are classified by different integers if STC is applied. STC requires a threshold array that includes (N-1) integers, which forms an arithmetic sequence in an N class scenario. STC will compare the output of the CGP with the integer array, to get which range the output belongs to. Next, it compares whether the range matches its expected class. Fig. 19 illustrated how STC works with CGP's output result.

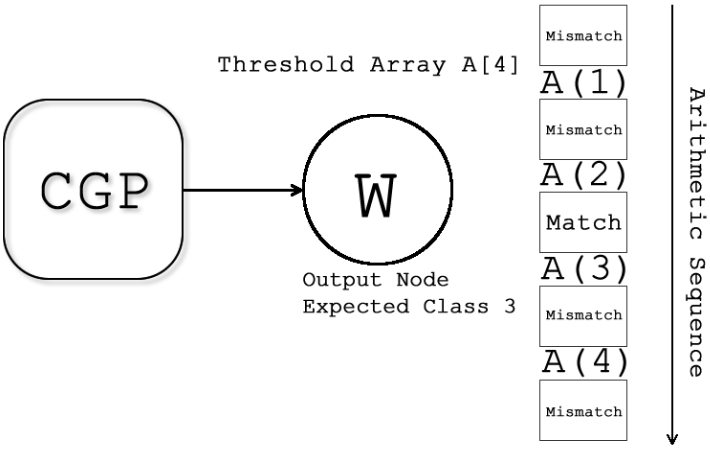


Figure 19 - Illustration of Simple Threshold Classifier [52]

The advantage of STC is it gives more configurable parameters, in this case, the threshold integer array, which improved the configurability of the CGP. This would give more possibility in terms of CGP evolution, which can result in more optimised CGP. However, the threshold array exposes a problem that the performance of the CGP may be rely on the array. Theoretically, the larger the range between each threshold, the higher the classifier's accuracy has. This feature enables STC may cover the overfitting of a CGP model - a CGP model may be overfitting, but the STC has better accuracy as the threshold is relatively lenient. Hence, a better fitness function that the result only relies on the CGP model is required.

Node Weighting Classifier

Node Weighting Classifier (NWC) requires CGP to be configured with multiple outputs, the number of outputs must match with the number of available classes. Given N classes in a dataset, the CGP will have N output nodes with different values, which are calculated by the CGP from the input data. Afterwards, the output node with the maximum value will be used to compare its output node index and expected class. If a class two data has the maximum output node to be its second node, then it is a match. Fig. 20 demonstrate a simple assessment from the NWC.

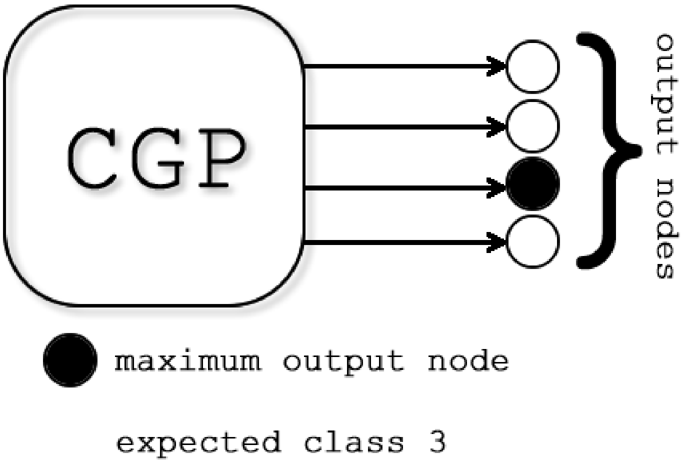


Figure 20 - Illustration of Node Weighting Classifier [52]

4.2. Support Vector Machine

As the main objective of this research project is to study the ability of an EA-evolved classifier in detection of PD severity, as well as how it can assist clinicians in medical research, SVM is used as a benchmark to compare readability of model visualisation with CGP to non-computer researcher. The implementation of SVM in this project is done by using MATLAB's classification learning software package. Parameter tuning for SVM classification task in this project includes different kernel function and using principal component analysis (PCA) to simplify classification task for the model.

4.2.1. Kernel Function

There are six available kernel functions for SVM in this software package: linear, quadratic, cubic, fine gaussian, medium gaussian and coarse gaussian. Each kernel function represents the different structure of the hyperplane. As this project involves a multi-dimensional SVM classification problem, each kernel function will be used to assess their performance.

4.2.2. Principal Component Analysis

In general, Principal Component Analysis, or PCA, is a procedure that used in statistical to simplify data set. It uses a statistical procedure called orthogonal transformation, it will take observations of a set of possibly related variables and attempt to convert them into a set of linearly uncorrelated variables, which is called principal components. By doing this, the dimension of the data set can be reduced, hence reducing the difficulty for a classification task. Fig. 21 shows a sample dimension reduction that done by PCA.

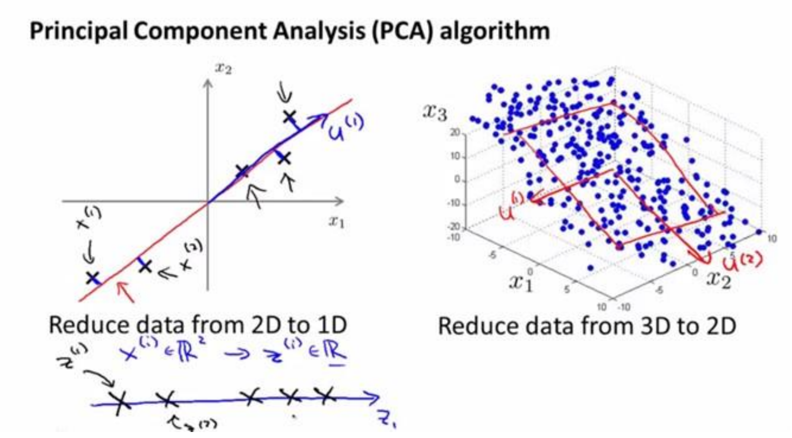

Principal Component Analysis (PCA) algorithm
x_2
x_1
$u^{(1)}$
Reduce data from 2D to 1D
$x^{(i)} \in \mathbb{R}^2 \rightarrow z^{(i)} \in \mathbb{R}$
z_1
x_3
x_1
x_2
$u^{(1)}$
$u^{(2)}$
Reduce data from 3D to 2D

Figure 21 - Demonstration of dimension reduction by PCA, taken from [65]

In SVM, the dimension of the SVM model is normally determined by the number of input data provided to the model. Normal classification task often contains three or more features as input for an SVM training task, which not only increase the difficulty for SVM to find the optimal hyperplane, but also difficult for researchers to observe the model through model visualisation if the dimension is too high. By applying PCA, principal components will be extracted from the features, so that the actual SVM model can handle less dimensional data set and simplifying the result of model visualisation.

5. Data Pre-Processing

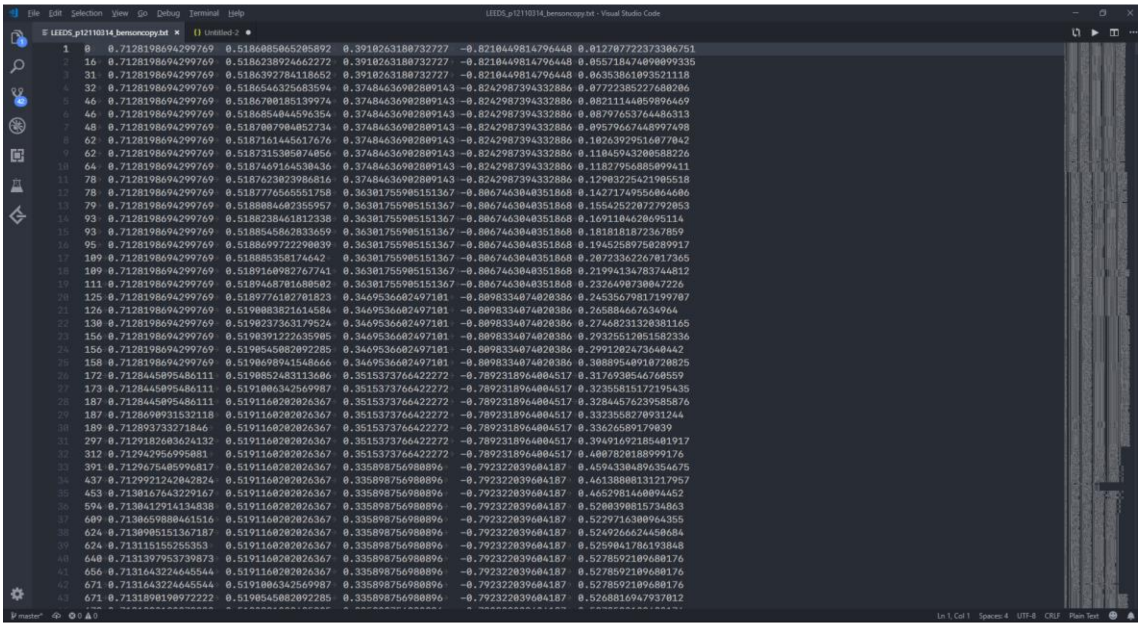
This section introduces how the raw data from the tablet is processed for the algorithm training session, as well as the how to determine the PD stage for each patient and label the data set accordingly.

5.1. Feature Extraction

Raw data of a figure is traced and recorded by the tablet during the drawing session with the patient. Table 4 shows the format from the raw data file, with a snapshot of sample data in Fig. 22.

Table 4 - Description table for raw data columns

Data	Timestamp	Pen-X	Pen-Y	Pen-Tilt-X	Pen-Tilt-Y	Pen Pressure
Description	Time of the data created	X-coordination of the pen	Y-coordination of the pen	Tilt in X position of the pen	Tilt in Y position of the pen	Pressure applied from the pen to the tablet


File Edit Selection View Go Debug Terminal Help
LEEDS_p12110314_bensoncopy.txt - Visual Studio Code
LEEDS_p12110314_bensoncopy.txt
Untitled-2
master*

```
0   0.7128198694299769 0.5186085065205892 0.3910263180732727 -0.8210449814796448 0.01270772237306751
16  0.7128198694299769 0.5186238924662272 0.3910263180732727 -0.8210449814796448 0.055718474090099335
31  0.7128198694299769 0.5186392784118652 0.3910263180732727 -0.8210449814796448 0.06353861093521118
32  0.7128198694299769 0.5186546325683594 0.37484636902809143 -0.8242987394332886 0.07722385227680206
46  0.7128198694299769 0.5186700185139974 0.37484636902809143 -0.8242987394332886 0.08211144059896469
46  0.7128198694299769 0.5186854044596354 0.37484636902809143 -0.8242987394332886 0.08797653764486313
48  0.7128198694299769 0.5187007904052734 0.37484636902809143 -0.8242987394332886 0.09579667448997498
62  0.7128198694299769 0.5187161445617676 0.37484636902809143 -0.8242987394332886 0.10263929516077042
62  0.7128198694299769 0.5187315305074056 0.37484636902809143 -0.8242987394332886 0.11045943200588226
64  0.7128198694299769 0.5187469164530436 0.37484636902809143 -0.8242987394332886 0.11827956885099411
78  0.7128198694299769 0.5187623023986816 0.37484636902809143 -0.8242987394332886 0.12903225421905518
78  0.7128198694299769 0.5187776565551758 0.36301755905151367 -0.8067463040351868 0.14271749556064606
79  0.7128198694299769 0.5188084602355957 0.36301755905151367 -0.8067463040351868 0.15542522072792053
93  0.7128198694299769 0.5188238461812338 0.36301755905151367 -0.8067463040351868 0.1691104620695114
93  0.7128198694299769 0.5188545862833659 0.36301755905151367 -0.8067463040351868 0.1818181872367859
95  0.7128198694299769 0.5188699722290039 0.36301755905151367 -0.8067463040351868 0.19452589750289917
109 0.7128198694299769 0.518885358174642  0.36301755905151367 -0.8067463040351868 0.20723362267017365
109 0.7128198694299769 0.5189160982767741 0.36301755905151367 -0.8067463040351868 0.21994134783744812
111 0.7128198694299769 0.5189468701680502 0.36301755905151367 -0.8067463040351868 0.2326490730047226
125 0.7128198694299769 0.5189776102701823 0.3469536602497101  -0.8098334074020386 0.24535679817199707
126 0.7128198694299769 0.5190083821614584 0.3469536602497101  -0.8098334074020386 0.265884667634964
130 0.7128198694299769 0.5190237363179524 0.3469536602497101  -0.8098334074020386 0.27468231320381165
156 0.7128198694299769 0.5190391222635905 0.3469536602497101  -0.8098334074020386 0.29325512051582336
156 0.7128198694299769 0.5190545082092285 0.3469536602497101  -0.8098334074020386 0.2991202473640442
158 0.7128198694299769 0.5190698941548666 0.3469536602497101  -0.8098334074020386 0.30889540910720825
172 0.7128445095486111 0.5190852483113606 0.3515373766422272  -0.7892318964004517 0.3176930546760559
173 0.7128445095486111 0.5191006342569987 0.3515373766422272  -0.7892318964004517 0.32355815172195435
187 0.7128445095486111 0.5191160202026367 0.3515373766422272  -0.7892318964004517 0.32844576239585876
187 0.7128690931532118 0.5191160202026367 0.3515373766422272  -0.7892318964004517 0.3323558270931244
189 0.712893733271846  0.5191160202026367 0.3515373766422272  -0.7892318964004517 0.33626589179039
297 0.7129182603624132 0.5191160202026367 0.3515373766422272  -0.7892318964004517 0.39491692185401917
312 0.712942956995081  0.5191160202026367 0.3515373766422272  -0.7892318964004517 0.4007820188999176
391 0.7129675405996817 0.5191160202026367 0.335898756980896   -0.792322039604187  0.45943304896354675
437 0.7129921242042824 0.5191160202026367 0.335898756980896   -0.792322039604187  0.46138808131217957
453 0.7130167643229167 0.5191160202026367 0.335898756980896   -0.792322039604187  0.4652981460094452
594 0.7130412914134838 0.5191160202026367 0.335898756980896   -0.792322039604187  0.5200390815734863
609 0.7130659880461516 0.5191160202026367 0.335898756980896   -0.792322039604187  0.5229716300964355
624 0.7130905151367187 0.5191160202026367 0.335898756980896   -0.792322039604187  0.5249266624450684
624 0.713115155255353  0.5191160202026367 0.335898756980896   -0.792322039604187  0.5259041786193848
640 0.7131397953739873 0.5191160202026367 0.335898756980896   -0.792322039604187  0.5278592109680176
656 0.7131643224645544 0.5191160202026367 0.335898756980896   -0.792322039604187  0.5278592109680176
671 0.7131643224645544 0.5191006342569987 0.335898756980896   -0.792322039604187  0.5278592109680176
671 0.7131890190972222 0.5190545082092285 0.335898756980896   -0.792322039604187  0.5268816947937012
```

Figure 22 - Screenshot of raw data sample

Each figure will generate up to 15,000 lines of data line that are similar to Fig. 22, which means the raw data cannot be used directly for the algorithm. The motivation of feature extraction is to calculate values from those thousands of data line that could represent this figure. In total, 17 features are designed and extracted from those raw data. Those features are categorised into three categories – General, Structural and Dynamical.

5.1.1. General Features

General features include features that exists in every figure, no matter the shape or initial purpose of the shape design. Those features will distinguish distorted images, which will highly occur in PD-D's recall task drawings.

Size

The size of the figure will be calculated by multiplying the height of the image with the width of the image. From the raw data, we can extract four vertex point of the image, which can be used to calculate the height and the width. A clear, high quality drawing should have a reasonable size. Too small or too large in image size often indicates distortion in drawing.

To calculate the size of the figure, we need to obtain the height and the width of the figure. Assume the Pen-X column in the raw data is array ***x***, and Pen-Y column in the raw data is array ***y***, we can obtain:

$$w = |x_{max} - x_{min}| \qquad h = |y_{max} - y_{min}|$$

Equation 1 - Weight and Height calculation

Where ***w*** is the maximum width of the figure, and ***h*** is the maximum height of the figure, thus, the size of the figure is the rectangle than can just surround the whole figure:

$$s = w * h$$

Equation 2 - Size calculation

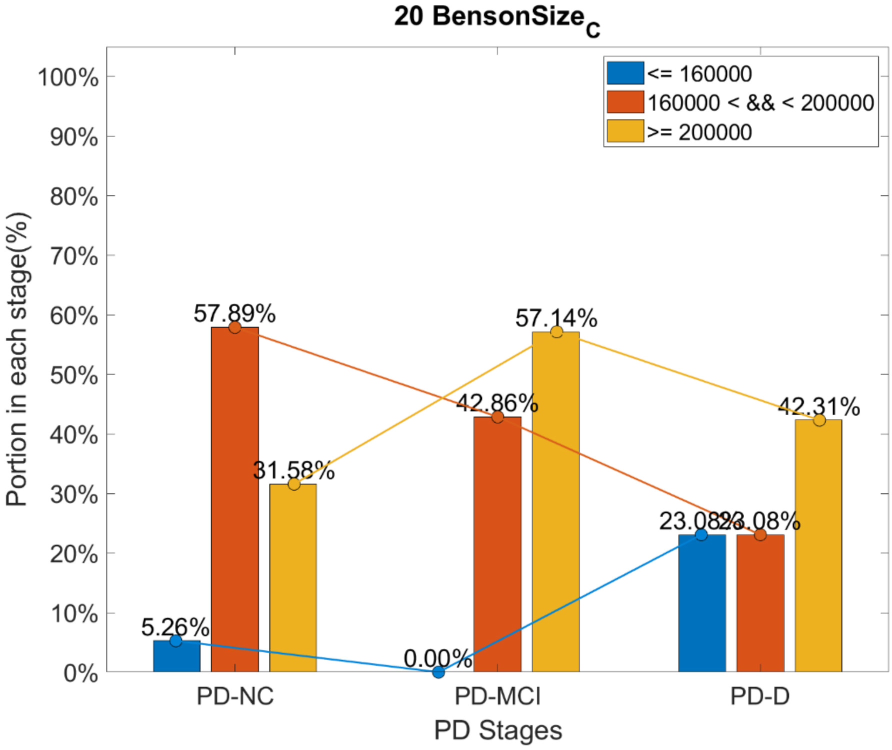


Figure 23 - Distribution of size in different PD group from copy task[1]

Fig. 23 shows the distribution of the figure size in each PD group. Though there is no clear standard for the proper size, too small in size will make the figure readable and possibly suggest difficulty in visuospatial ability from the test subject, while too large may indicate the test subject cannot control their hand movement. Fig. 23 also shows a decline of portion in the number of figures with normal size as the PD stage advances.

Aspect Ratio

The size of the figure is not sufficient to represent the distortion situation of the whole figure as some patients may tend to draw small or large figures, or a heavily distorted figure may have the same size of a normal figure. Aspect ratio is another

---

[1] At the top-right corner of the figure, ***'&&'*** is the placeholder which represents the data point. This applied to all other figures in this thesis.

feature with the figure size to show the distortion situation. As shown from the standard Benson figure in Fig. 24, the sample figure has an aspect ratio of 2.36:1.

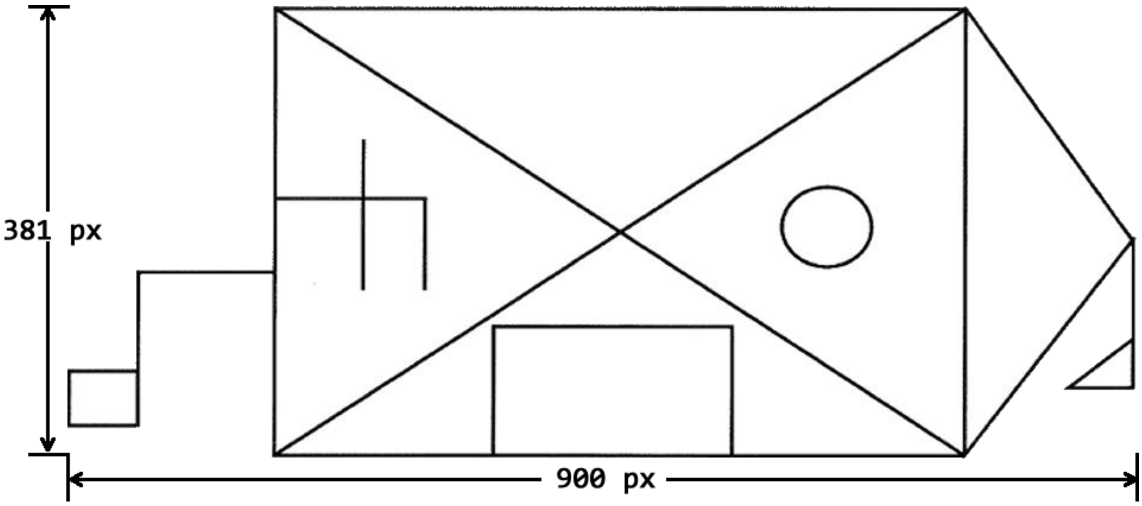


Figure 24 - Measurement of the standard Benson

Variation is acceptable but should not be too far away from this standard. As we have the height H and width W from previous equation, we can calculate the aspect ratio by:

$$ar = \frac{w}{h}$$

Equation 3 - Aspect ratio calculation

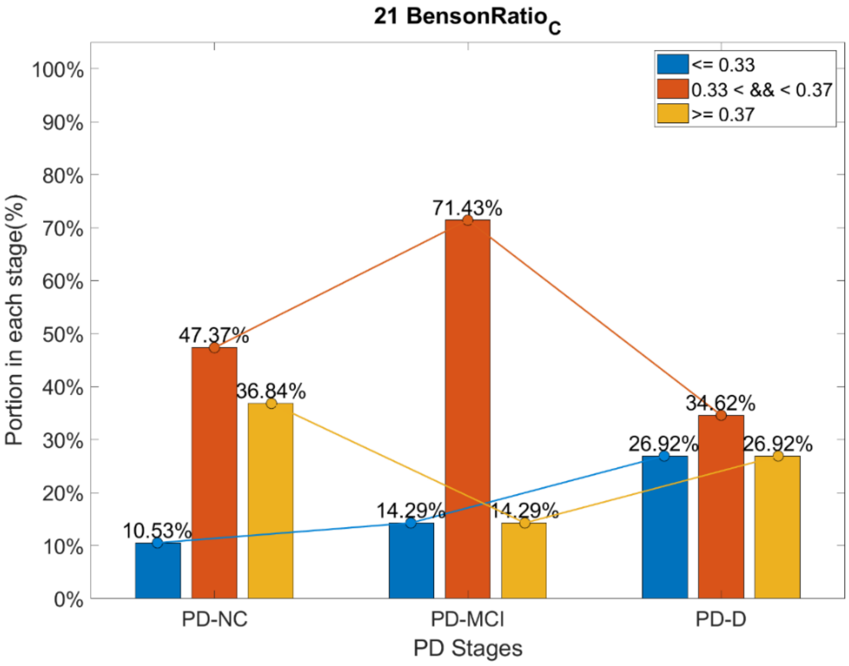


Figure 25 - Aspect Ratio distribution across all PD stages' figure test

Fig. 25 shows the distribution of aspect ratio in three PD stages' figure test result. Due to the system used for data capture in the digitised tablet, figures that generated from the raw data is different from the original drawing in regard to the aspect ratio. Therefore, the expected aspect ratio for figures that regenerated from the data is between 3.3:1 and 3.7:1. Any value that out of this range is considered distorted in terms of aspect ratio.

According to Fig. 25, 47.37% and 71.43% of the figures from PD-NC and PD-MCI test subject performs well on controlling their scale of the figure in the copy test, while the portion in PD-D is only 34.62%. The portion of 'taller' figure increases as the PD stage advances, while PD-NC has the maximum value for the portion of 'wider' figure, this may be due to the fact that the original drawing is a 'wide' figure.

5.1.2. Structural Features

Structural features represent the visual characteristic of the figure, including the portion of lines in different angles, the straightness of line segments and the completeness of the figure. Visually, structural features are easy to distinguish. Fig. 26 shows great visual variance across all three different drawings.

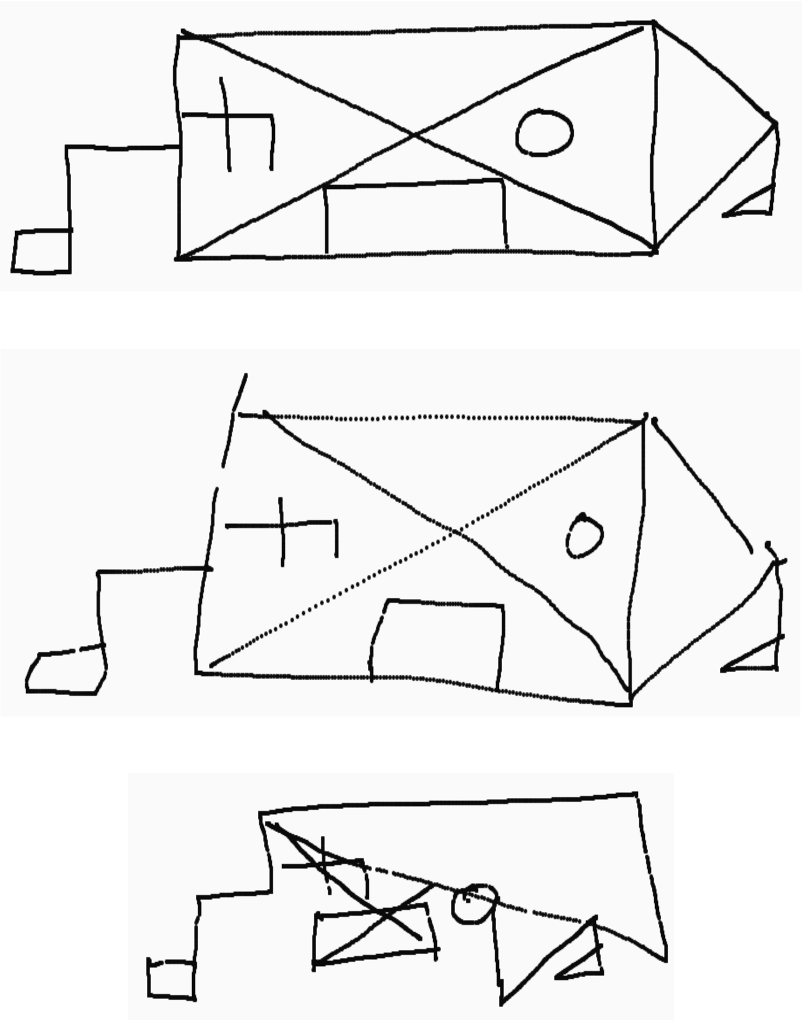


Figure 26 - Comparison of Benson drawing between PD-NC (top), PD-MCI (middle) and PD-D (bottom) from copy task[2]

Total Length

Total length is literally the length of all line segments in one figure. It further compensates the *Size* feature which is only indicative on the position of vertex points. Total length roughly represents the amount of line segments in the given figure area. Given that there are N timestamps in the raw data, Pen-X column is array ***x*** and Pen-Y column is array ***y***, the calculation of total length is done by summing up the distance between points from neighbouring timestamp. Equation 4 shows the mathematical representation of this feature:

---

[2] All figures are proportional to their original size and scale

$$l = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$

Equation 4 - Calculation of total length of the figure

From the value of the pen movement distance, this feature can provide a rough indication of the completeness of the figure.

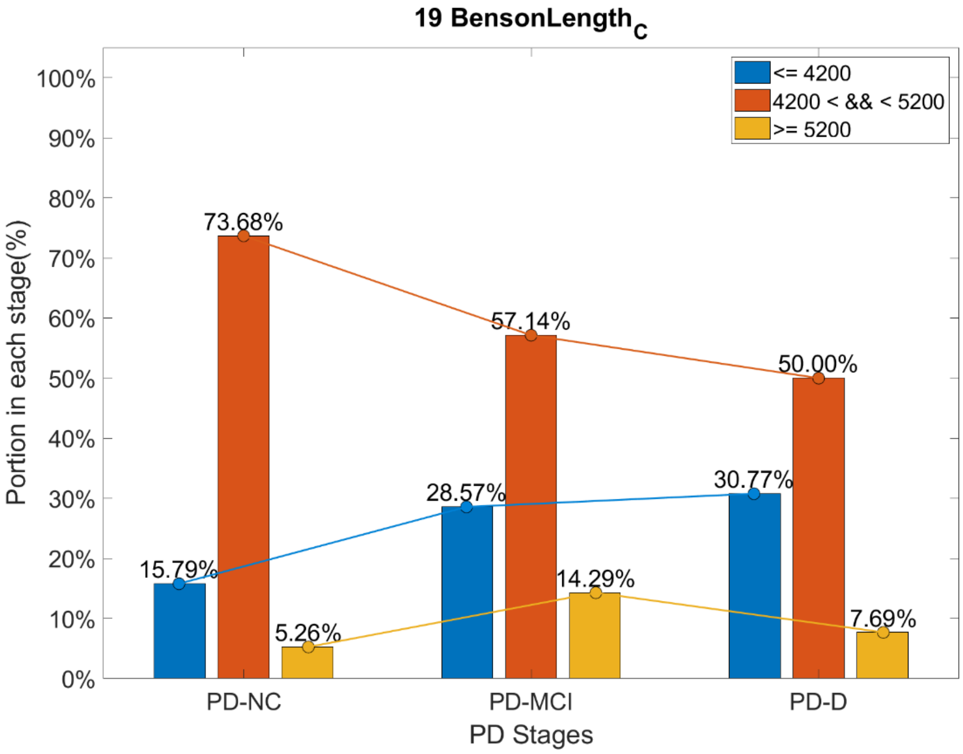


Figure 27 - Figure length value distribution in all three PD stages from copy task

Fig. 27 shows the distribution of the pen travel length of the figure drawn by the test subjects in all PD stages. As the stage advances, the portion of the drawing with less length increases, which may indicate missing or distorting components in the figure. The portion of normal length drawing also decreases while the stage advances, indicating the relevance of this feature and figure data representation.

However, total length does not show the composition of the figure, which is a crucial part in terms of the figure structure. Another set of features are needed to compensate this.

Figure Composition

As Benson figure is consisted by several basic geometrical shapes, data of figure composition will be very representative to show the structure of the figure. Two solutions are proposed to calculate data for composition representation:

First proposal is to separate the figure with line segments, then combine groups of segments according to their length and position, to identify Benson components from the NACC form. The procedure of separate line segments from the figure follows:

1. Separate the raw data according to the pen pressure. When the pen pressure reaches 0, it means the test subject has lifted the pen, thus generate a segment break;
2. Combine all segments according to their length, direction and position into NACC components;
3. Extract features from those components.

However, this method comes with limitation that the calculated number of segments varies a lot as people have different drawing preferences. Some of the test subjects tend to draw different line separately, while others draw lines that are connected together once. Thus, a standard is very difficult to establish to apply this method across all figures as it may consume too much time resources for this research project. In this case the author chooses to compromise the detail and aiming for a complete standard for separating the figure.

Second and the adapted method is to group all drawings into three categories by their gradient: horizontal, vertical and oblique. The reasonable Benson structure should have a fair distribution of these three lines as shown in Fig. 28.

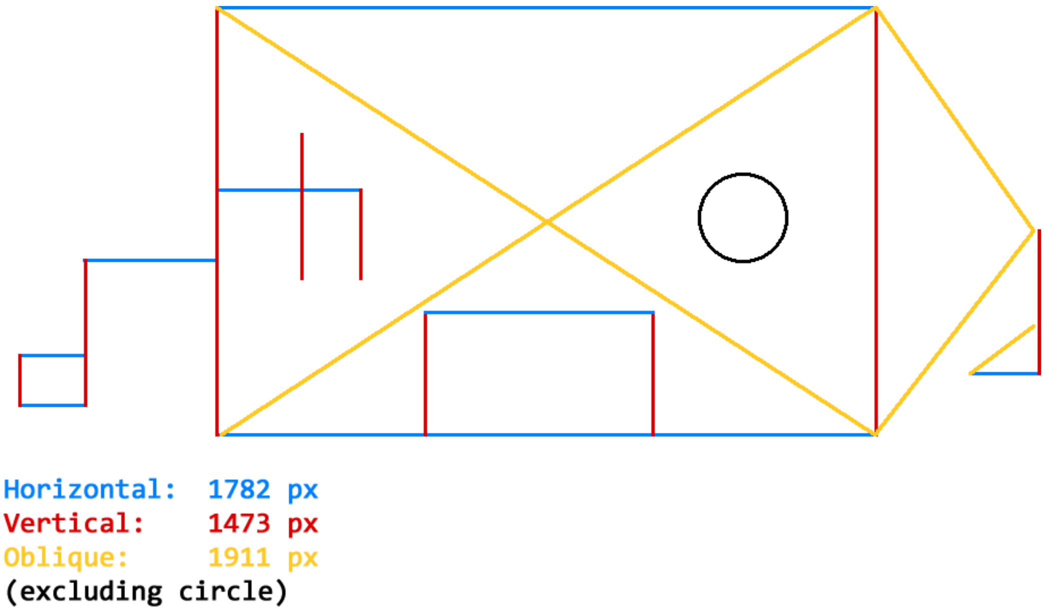


Figure 28 - Decomposition of standard Benson in three-line groups

However, Benson figure is not purely composed by line segments. Curve also exists in the form of circle. Therefore, additional work is needed to decompose curves from the circle to line segments. It is worthy to note that a circle can be divided into 360 equal parts with each part contributes to 1/360 of its circumference. According to the previous definition, lines with angle of 0~10 are classified as horizontal line, and 70~90 are classified as vertical line, the rest are classified as oblique. The diameter of the circle in Fig. 28 is 74px, therefore the circumference is shown in equation 5:

$$C = \pi * 74\text{px} \approx 232.478 \text{ px}$$

Equation 5 - Length in pixel of the circle component in the figure

For a 90-degree range, horizontal, vertical and oblique part of the curve should have the portion of follows:

$$\%_{horizontal} = \frac{10}{90} \approx 11\%$$

$$\%_{vertical} = \frac{20}{90} \approx 22\%$$

$$\%_{oblique} = \frac{60}{90} \approx 66.67\%$$

Equation 6 - Portion of horizontal, vertical and oblique from the circle

Hence, the length of the circle in pixel should be:

$$l_{horizontal_C} = C * \%_{horizontal_c} \approx 26\, px$$

$$l_{vertical_C} = C * \%_{vertical_c} \approx 52\, px$$

$$l_{oblique_C} = C * \%_{oblique_c} \approx 156\, px$$

Equation 7 - Length of horizontal, vertical and oblique in pixels

Instead of using 232.478 px as the circumference of the circle, we sum up $l_{horizontal_C}$, $l_{vertical_C}$ and $l_{oblique_C}$, then get 234 px and thus, 5400 px for the total length of the standard Benson. Hence, we can get the portion of horizontal, vertical and oblique part of a standard Benson as:

$$\%_{horizontal} = \frac{l_{horizontal_C} + 1782}{5400} \approx 33.48\%$$

$$\%_{vertical} = \frac{l_{vertical_C} + 1473}{5400} \approx 28.24\%$$

$$\%_{oblique} = \frac{l_{oblique_C} + 1911}{5400} \approx 38.28\%$$

Equation 8 - Portion of horizontal, vertical and oblique lines in the universal figure

Although this standard is not as precise as previous one, this one is easier to implement to across all figures and extract features from those lines. It is also worth noting that realistic drawing is not as precise as the standard figure. During the drawing, no ruler was given to the test subjects. Therefore, in terms of implementing this method, error allowance is given when calculating the gradient from the raw data to accept normal vibration during drawing as well as general orientation of the whole figure. The technical definition of the line group as follows in table 5:

Table 5 - Definition of horizontal, vertical and oblique lines

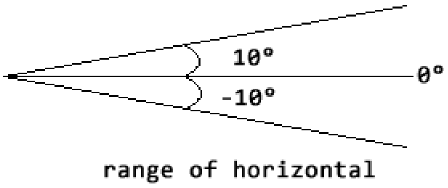
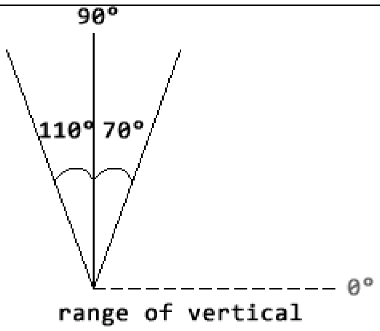
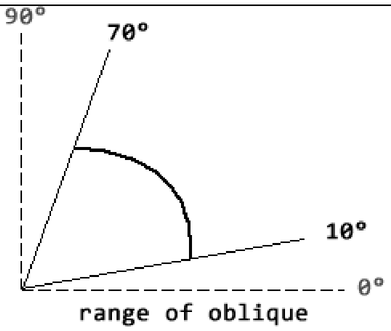
Line Group	**Definition**	**Description**	**Illustration**
Horizontal	Line with the angle between 0~10 degree	Line segments that forms a horizontal direction, 10-degree error is allowed.	
Vertical	Line with the angle between 70~90 degree	Line segments that forms a vertical direction, 20-degree error is allowed	
Oblique	Line with the angle between 10~70 degree	Line segments that forms an oblique direction.	

Fig. 29 - 32 proves that the definition of line grouping in table 5 is applicable in terms of data separation as those figures are generated by those separated data:

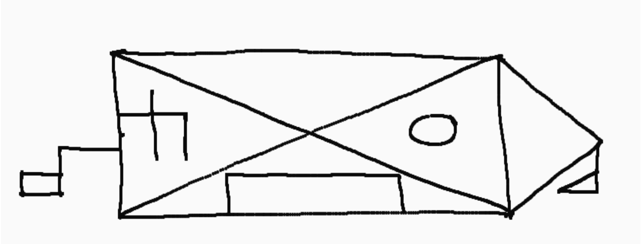


Figure 29 - Real Benson drawing sample by a test subject in control group - Copy

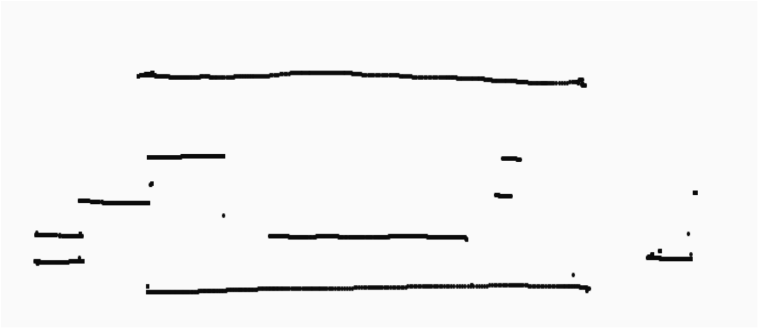


Figure 30 - Data separation of horizontal part of previous figure

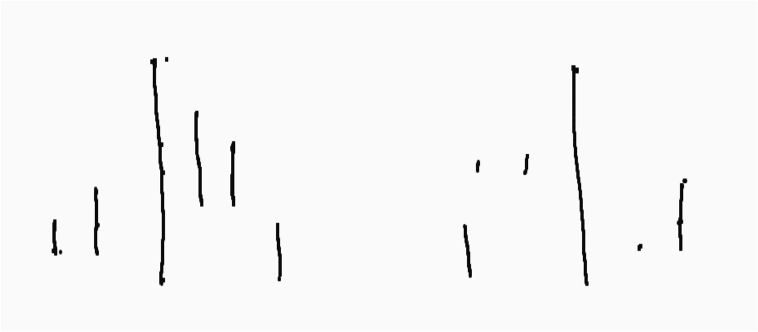


Figure 31 - Data separation of vertical part of previous figure

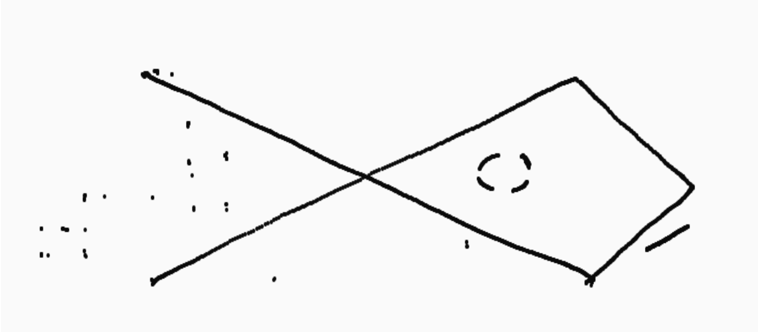


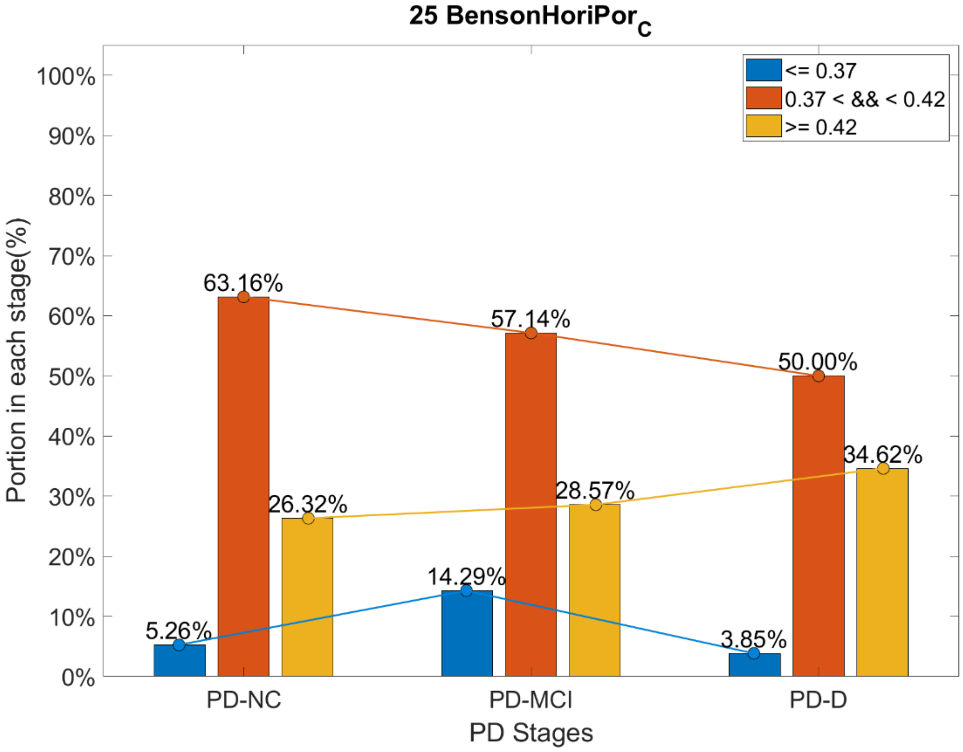
Figure 32 - Data separation of oblique part of previous figure

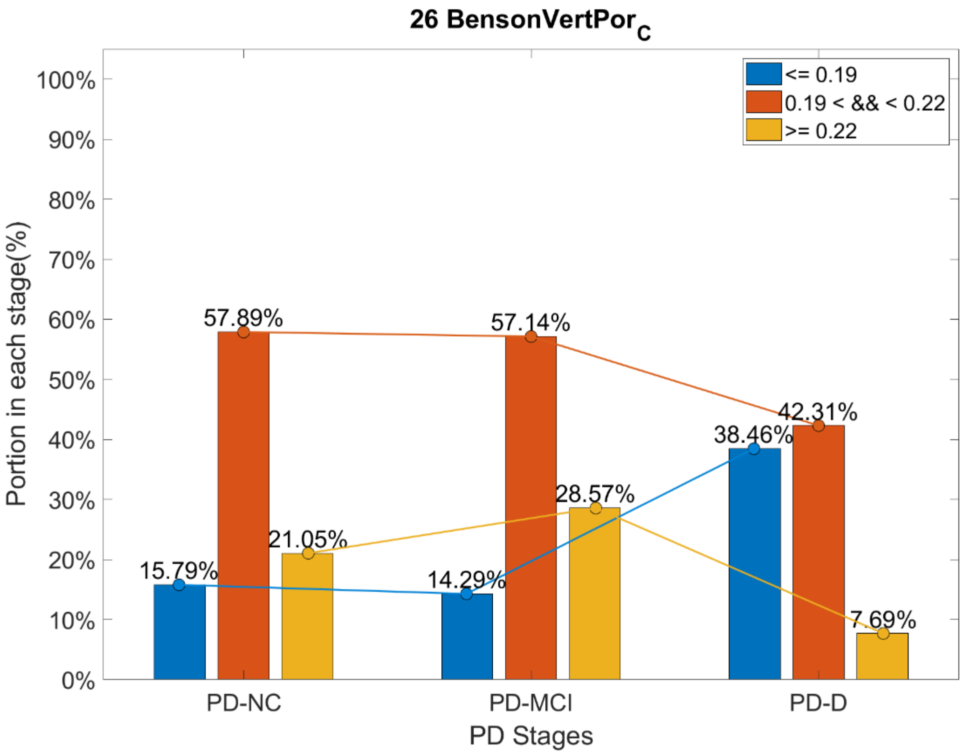
After the separation of the figure, we then calculate the portion of each line group in terms of length. The algorithm to calculate each portion of the line group is given in equation 9:

$$\%_{hvo} = \frac{\sum_{i=1}^{n_{hvo}-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}}{l}$$

Equation 9 - Calculation of portions of each line group

Where ***n*** is the total timestamp in each line group, and ***x*** and ***y*** are there relative coordinate data from the raw data column and ***l*** is the total length of the figure.





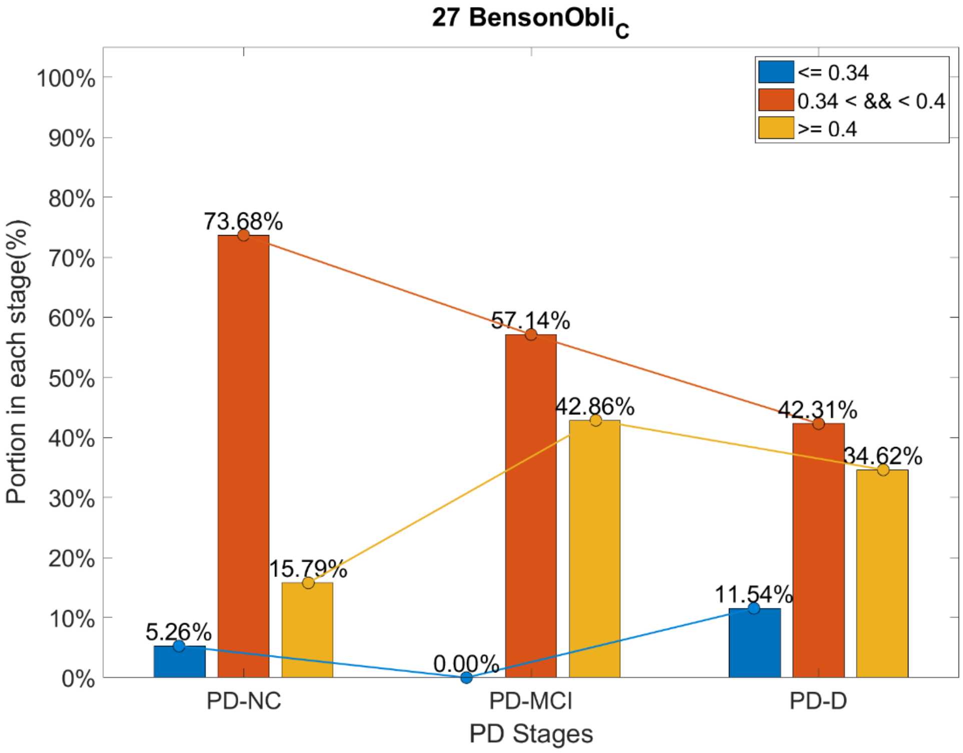


Figure 33 – Distribution of portion from three angles across all PD stages

Line Straightness

Because motor skill is one of the main symptoms in PD, we assess subject's motor skill by assessing the straightness of lines from their Benson figure result as this figure is mainly composed by lines. The straightness of the line is measured by the stability of the gradient between the coordinates of the stylus in different timestamp. Standard deviation is applied to measure the stability because it is often used in statistics to quantify the variation of a set of values [66]. For example, an ideal straight line will have a constant gradient across the drawing period, therefore the standard deviation for this line is 0, representing best drawing stability. Given the x-coordinate column of the raw data is array ***x***, and y-coordinate column of the raw data is array ***y***, the measurement algorithm for Benson's line straightness is:

$$SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n-1}\left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i} - \frac{1}{n}\sum_{i=1}^{n-1}\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right)^2}$$

Equation 10 - Calculation of standard deviation of degrees across lines in the figure

Where ***n*** is the size of the x-y array, which is the number of timestamps that included for the stability calculation. The lower value indicates a better performance in motor skill assessment.

However, one inevitable downside is the potential wrong value of this feature. It is possible that change of direction during a line drawing will affect the calculated value since the combined segment is not considered as a straight line by this algorithm, but in fact, it is combined by two straight lines. Curve also exist in the Benson figure. However, since it is a designated part of the whole figure, we should expect small variation in terms of the overall gradient. But regional gradient, such as data in horizontal and vertical part of the figure, as shown in Fig. 4 and 5, should has less variation than oblique part as shown in Fig. 6. Therefore, both overall and regional part of the figure will be used to calculate the standard deviation of the gradient.

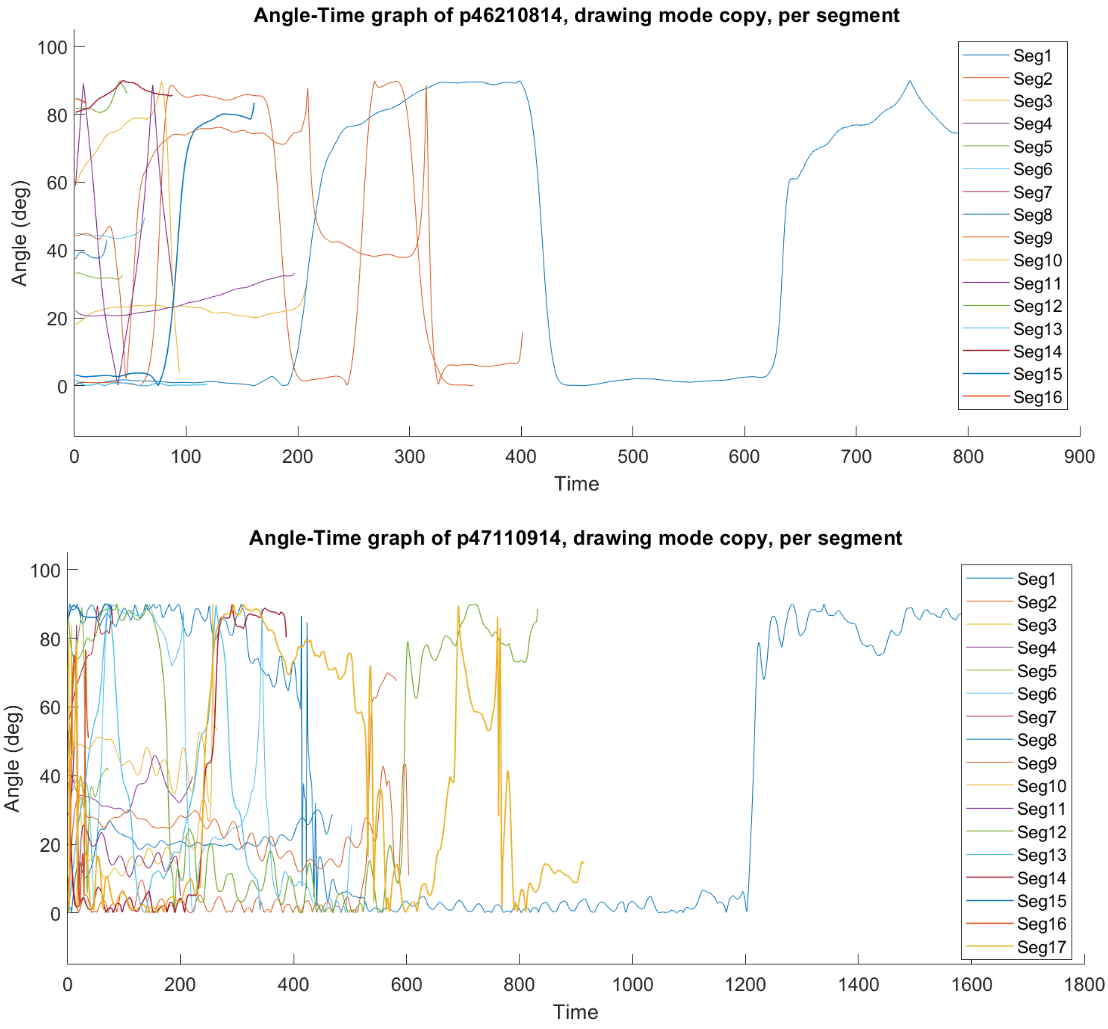


Figure 34 - Comparison of pen direction change over time between PD-NC (top) and PD-D (bottom) patient drawing data

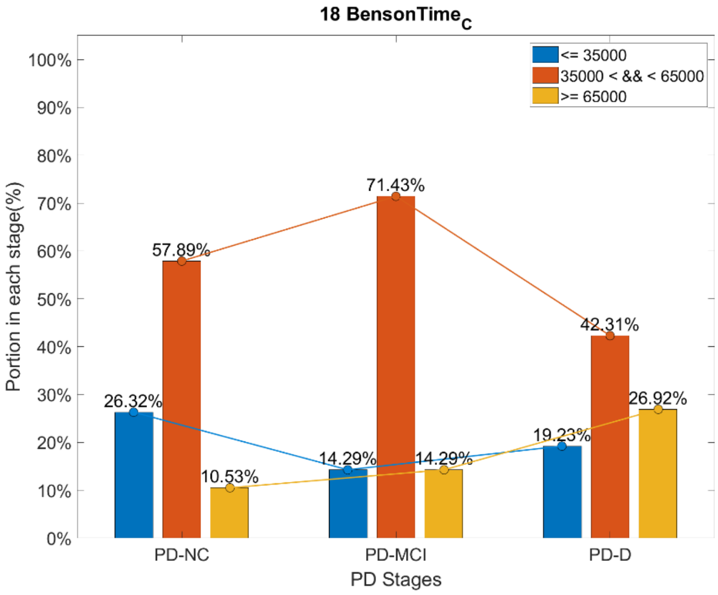
Fig. 34 compares a PD-NC (top graph) and PD-D (bottom graph) patient's drawing test by their pen direction change against time. From Fig. 34 we can see that PD-NC's drawing graph has a relative smooth transition during drawing, while there is a significant instability in PD-D's drawing graph.

5.1.3. Dynamical Features

Dynamical features represent the performance of the test subject in the movement aspects, for example, the time that spent on drawing and thinking, the stability of their drawing speed, the hesitation performance during the drawing, etc.

Total Time

The measurement of total drawing time is determined by the number of timestamps, regardless of the pen pressure. The conversion between timestamp gap and real time in unknown, but the number of timestamps is directly proportional to the actual time. Fig. 35 shows the distribution of drawing time among all patients in different categories.



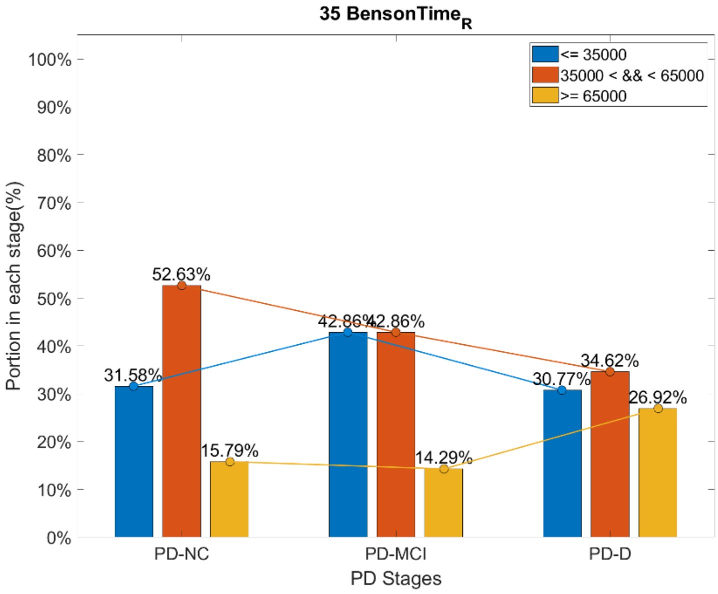


Figure 35 – Distribution of total time spent on drawing across all samples

Velocity Stability

Velocity is a vital part of motor skill. However, the reflection from velocity on the motor skill is not measured by the speed rate. A fast drawing may indicate a confident drawer, but also may show the test subject failed to control the pen steadily. On the contrary, slow drawing could show long hesitation time, or simply because the test subject is careful on what he was doing. Therefore, similar to what we did on the line straightness, we use standard deviation to measure the stability of the velocity. With normal motor skills, we expect the test subject will minimise their change in drawing speed, to provide a stable generation of pen trace. Because the sampling rate of the digitised tablet is consistent when all the figures are captured, timestamp can be used as a unified time unit. Therefore, velocity calculation can be represented by the distance. The equation used for calculating the stability of the velocity is based on standard deviation. First, we need to calculate the distance between certain points:

$$distance_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$

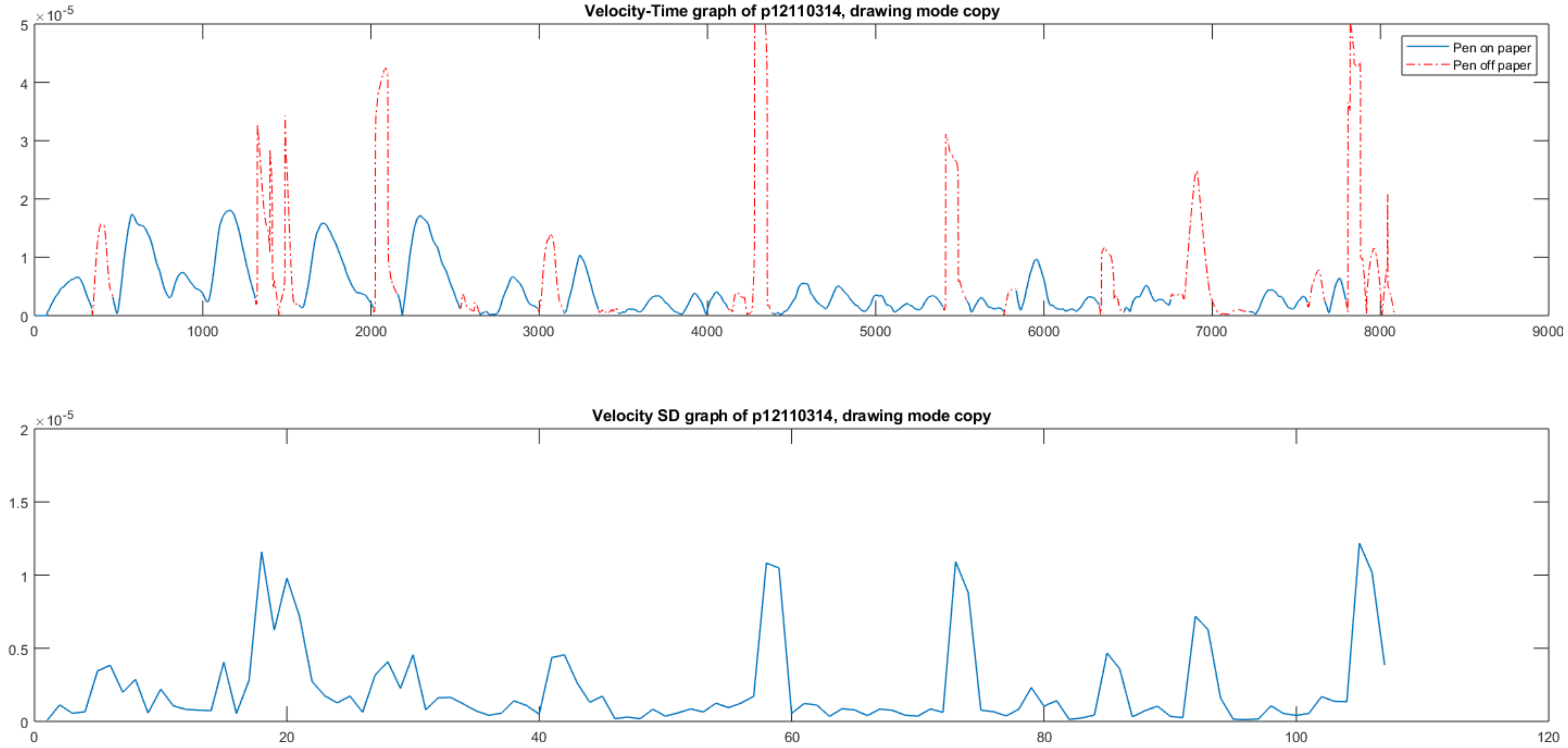
Equation 11 – Calculation of distance between points

where ***x*** is the horizontal coordinate of the pen at timestamp ***i*** and **y** is its vertical counterpart. The stability in terms of drawing velocity can be represented as:

$$SD_{vel} = \sqrt{\frac{1}{n}\sum_{i=1}^{n-1}(distance_i - \frac{1}{n}\sum_{i=1}^{n-1} distance_i)^2}$$

Equation 12 – Calculation of velocity stability

where ***n*** is the number of timestamps used by this test subject. Figure 36 and 37 shows the comparison of velocity stability between a PD-NC and PD-D patient's drawing result.


Velocity-Time graph of p12110314, drawing mode copy

Velocity SD graph of p12110314, drawing mode copy

Figure 36 - Velocity and its SD graph of a PD-NC patient's drawing from a copy task

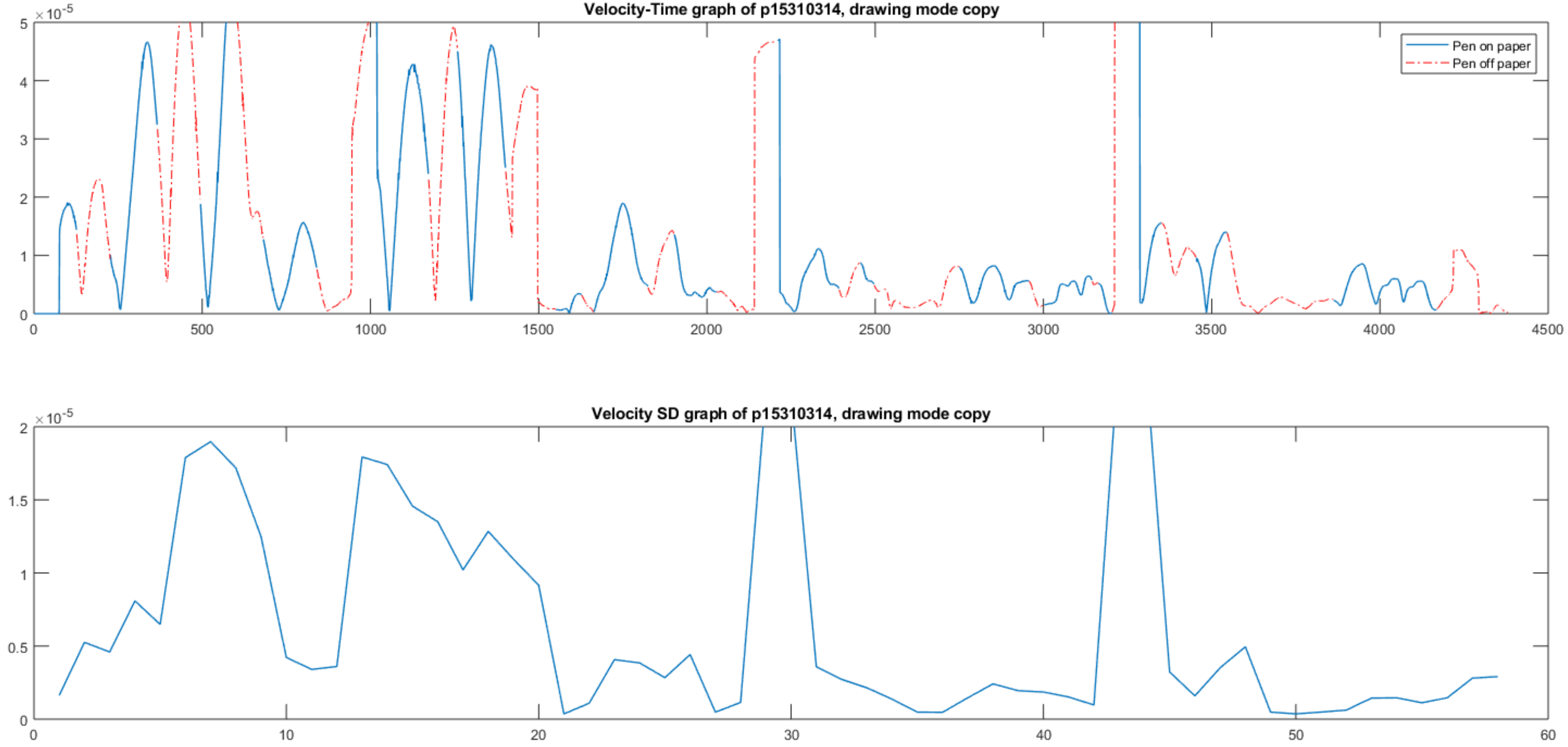
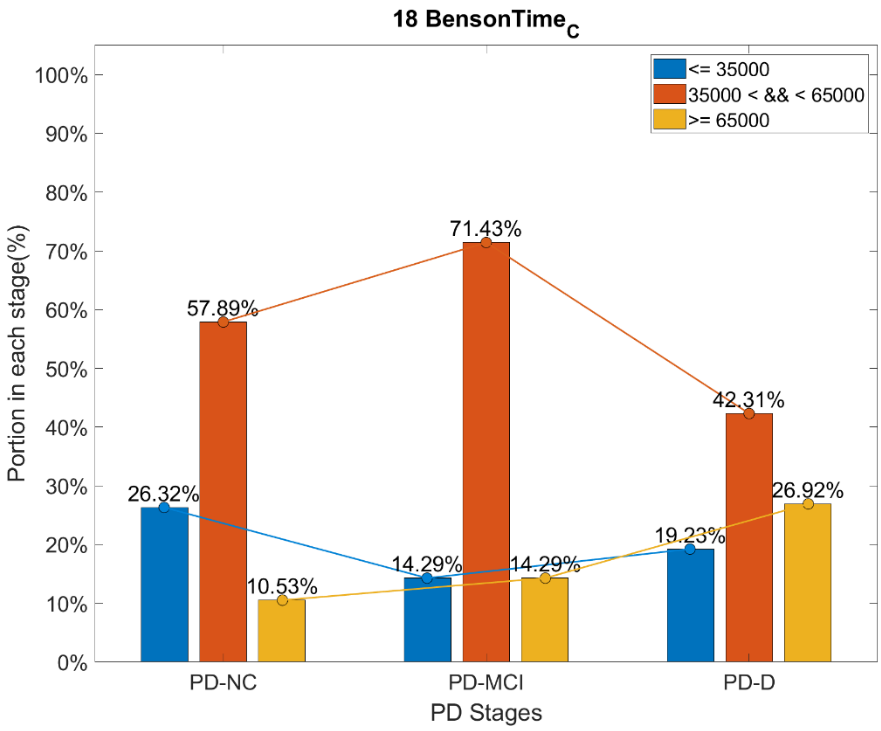


Figure 37 - Velocity and its SD graph of a PD-D patient's drawing from a copy task

By comparing Fig. 36 and 37, we can see a difference in terms of velocity stability during drawing between a PD-NC and PD-D patients. Fig. 37 shows as the stage worsens, the velocity stability tends to descent, Fig. 37 shows greater instability compared with Fig. 36.

Time on Drawing

Apart from the total time that spent for the entire drawing process, another potential feature is the time that the test subject spent while the pen has direct contact with the tablet. Therefore, it will only count the number of timestamps when the pen pressure is not zero. Additionally, it can reflect the portion of time that test subject was spending on thinking next step, which contributes to hesitation assessments. However, hesitation can also occur during the drawing, so addition features are needed for hesitation assessment.



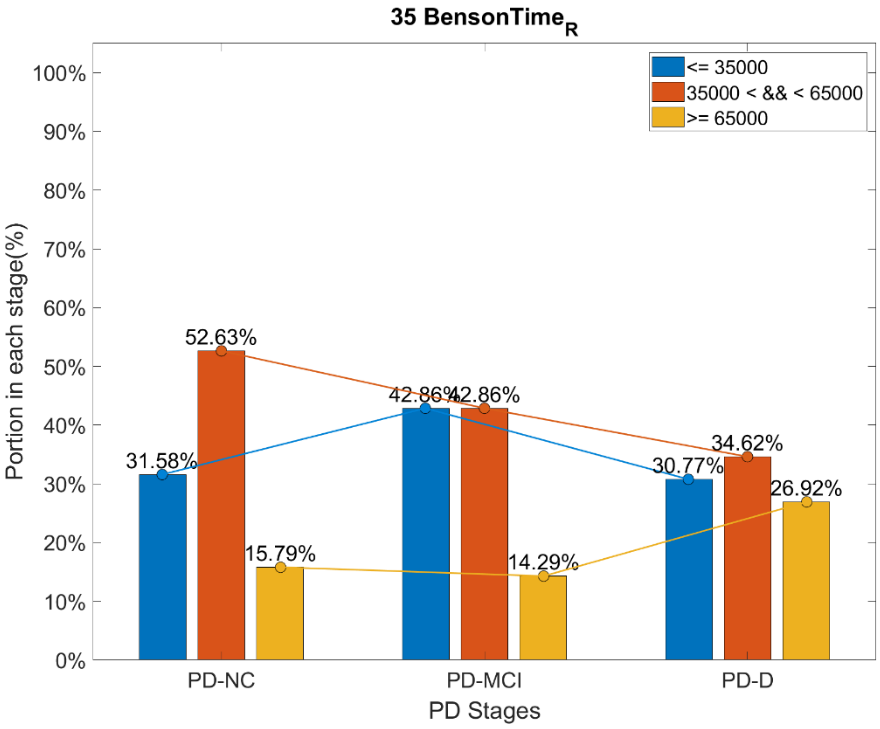


Figure 38 - Distribution of total time spent when pen is on the tablet across all samples, copy and recall

Hesitation Analysis

The movement impairment that inducted by the PD is defined as bradykinesia, meaning slowness in initiating voluntary movements [67], which in combination with one of three physical signs: rigid muscle, vibration when resting and balance disorder. Such impairment in movement can be detected in Benson figure test through hesitation.

The definition of hesitation in this project is the stylus is in a same position for a period of time while the test subject is not moving the stylus away from the tablet. This action from test subject indicates a pause of movement for a period of time. There is an allowance of time limit for unintended movement pause for a short period of time, in which case this will not contribute towards hesitation. Fig. 39-42 shows samples of hesitation marks in some drawing samples, larger mark indicates longer pause time.

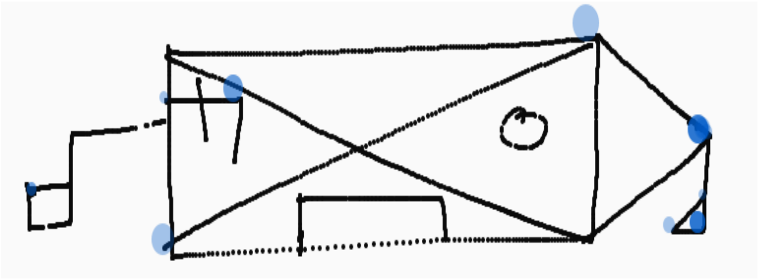


Figure 39 - Copy figure from control group test subject with hesitation marked

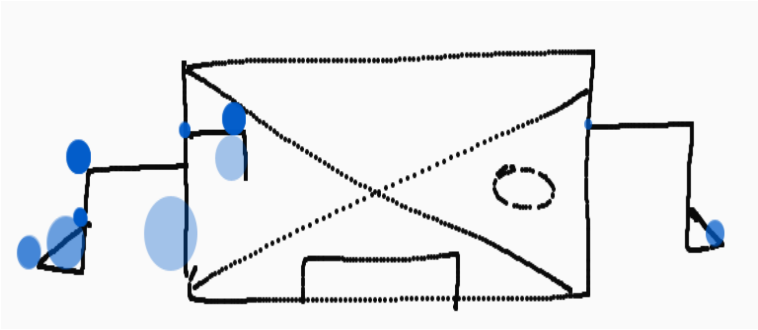


Figure 40 - Recall figure from control group test subject with hesitation marked

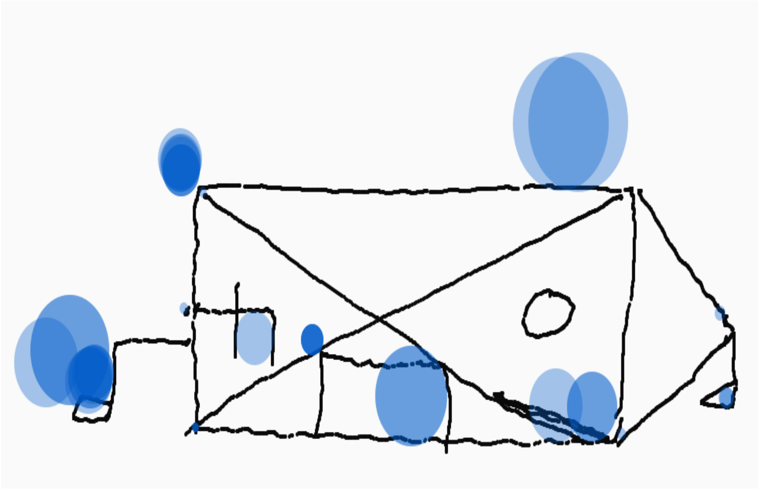


Figure 41 - Copy figure from patient group test subject with hesitation marked

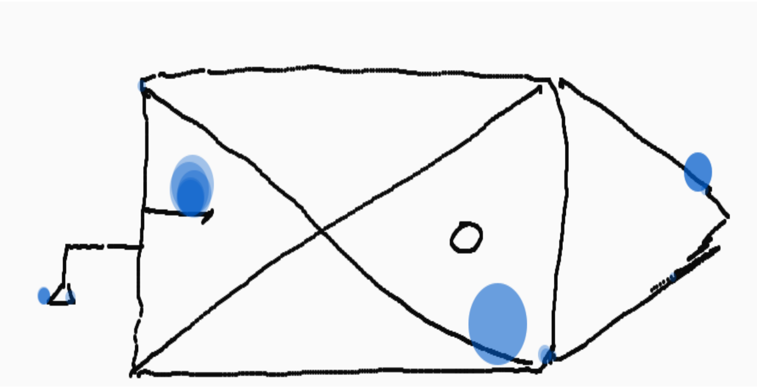


Figure 42 - Recall figure from patient group test subject with hesitation marked

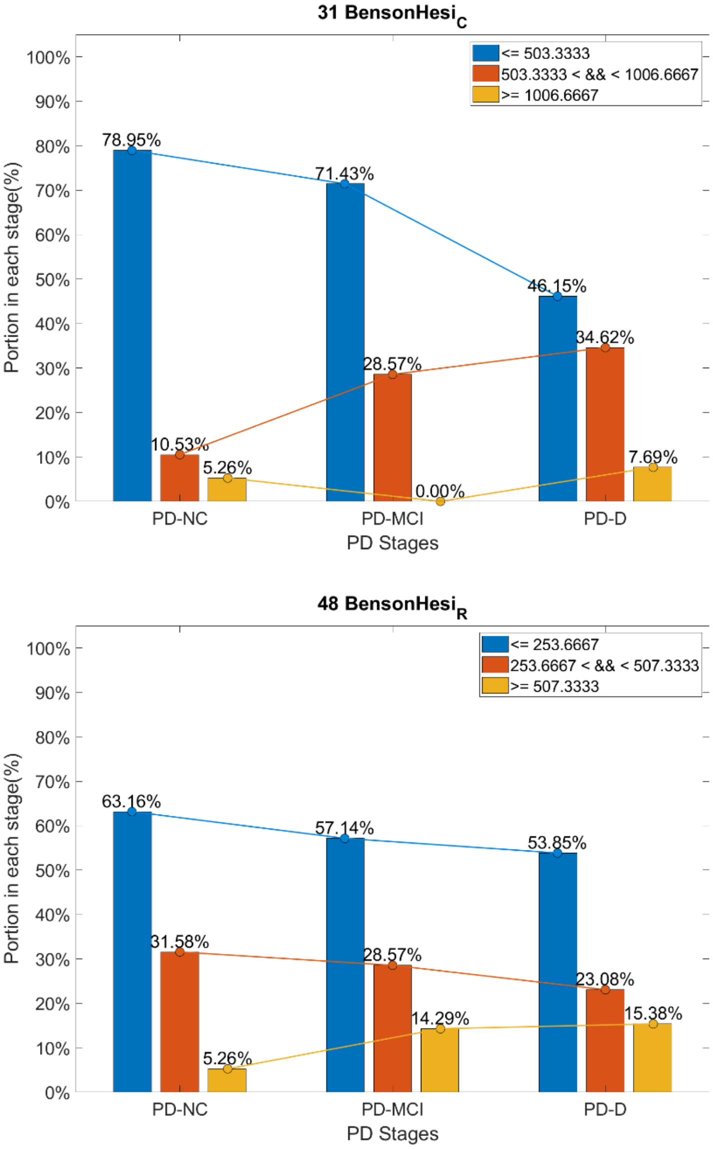


Figure 43 - Hesitation value distribution across all three PD stages in both copy and recall task

Fig. 43 shows a clear indication of how hesitation can distinguish patient in different stages. In the hesitation assessment, less value indicates better cognitive and motor skill performance. As the PD stage advances, the portion of drawing with less hesitation significant drops. In copy task, this value drops from 71.43% to 46.15% from PD-MCI to PD-D while the portion of figures with higher hesitation increases as the stage advances. This feature can only be identified by using data extraction along with evolutionary algorithm. Both traditional visual computation and conventional manual observation cannot compare the figure with the consideration of hesitation performance.

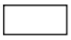
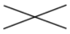
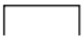
5.2. Classification

This section discussed how those figures collected are initially classified for supervised learning. In general, there are two methods to classify those figures – Benson figure score and test subject's condition.

5.2.1. Benson Figure Score

Along with the standard Benson figure, a standardised marking sheet is also provided on the NACC form. This sheet rates the Benson according to its composition, which is 8 components. Each component is rated against its placement and accuracy. One extra point is available for the universal component placement for the figure. A total of 17 marks is available. Therefore, there are two ways to produce a classification scheme. Table 6 shows the complete marking sheet from the NACC form.

Table 6 - NACC Benson marking scheme [58]

Component	Description	Available Marks	
	Four-sided, 90° angles, width > height, any gaps or overlaps < 8mm	Accuracy 1	Placement 1
	Reasonably straight lines; any gaps or overlaps < 8mm	Accuracy 1	Placement 1
	Connects at middle third, no overlap with diagonals	Accuracy 1	Placement 1
	Reasonably round, doesn't touch sides	Accuracy 1	Placement 1
	Vertical lines > 1, 2 distance to diagonals, width > height, 90° angles	Accuracy 1	Placement 1
	Connects below #3, top of square above bottom	Accuracy 1	Placement 1
	Vertex corresponds to middle third; any gaps or overlaps < 8mm	Accuracy 1	Placement 1
	Gap b, w #7 < 5mm, angle at end of stem = 90°	Accuracy 1	Placement 1
	Overall component placement and accuracy	1	

First approach is to classify those figures into 17 classes with each score indicates one class, the drawing quality improves along with the increase of class number. However, this will encounter an issue where each class will have too few samples for the algorithm to understand the features for each class. In total we performed this test on 58 patients, meaning the number of data is up to 58 for each copy and recall task, which is used to train the algorithm separately. That indicates an average of 3 samples per class, which is too few for data classification problem.

Alternatively, we can regroup those figures into few groups but larger number of samples in each group. We can either average the maximum marks to regroup, for example, 4 marks range provides a new group, or we can develop new scheme based on the unique feature from different groups. Either way can address the lack of sample issue from previous scheme. However, this classification method will encounter marginal marks problem, which means two similar drawing quality figures are put into two different neighbour groups. Figures in the same group may varies a lot in terms of drawing quality as well. Because the original marking scheme is based on each component, an 8-score drawing may indicate 4 perfectly drawn component with the remaining missing, or 8 poorly drawn but recognisable components. In this case the latter should have better rating, but both falls in the same group.

5.2.2. Parkinson's Disease Stage

This classification scheme will classify all drawing by the stages of patients. According to table 2, we can then classify all 58 patients into three PD stages. Among them, 19 are classified as PD-NC, 7 are classified as PD-MCI, and 26 are classified as PD-D. The remaining 6 patients have not completed CDR and MoCA yet, therefore, their data will not be included for algorithm training. Among the 52 valid entries, we expect 52 data for copy test, and 52 entries for recall test.

However, data corruption affected part of the sample. In copy test dataset, 7 data were lost, making the loss rate 13.5%, while recall test dataset has lost 2 entries, with a loss rate of 3.8%. Due to the time limitation of this project, there's no intended plan to regenerate those lost data and complete the test on those who has not yet. Table 7 shows the final count of valid data entries for this project.

Table 7 – Number of data in different classes for different drawing task

PD Stage	**Copy**	**Recall**
PD-NC	18	19
PD-MCI	7	7
PD-D	23	24
Total	**48**	**50**

6. Experimental Strategy

This section discusses the experimental strategy that applied during the experiments on CGP and SVM, including the approach to prevent overfitting and validate those trained models, as well as general procedures to train and select the fittest model for each training session.

6.1. Model Validation

This section discusses the strategies that applied to address the overfitting issue that may exist during the algorithm training session.

6.1.1. Overfitting

Overfitting is an issue where a mathematical model is only applicable to a particular set of data and may fail to fit additional data or future set of data [68]. This terminology is widely applied in statistics and can be used in machine learning for performance analysis. In machine learning, the difference in complexity between the hypothesis and the function may decide if the model is overfitting or not. If the hypothesis is too complex for the function, then the machine learning model tends to be overfitting and vice versa [69]. In this research project, overfitting can be addressed in two ways, one is specific for CGP and another one is applicable to all algorithms.

6.1.2. CGP Self-Validation

For each chromosome in each generation of CGP, each chromosome, or the mathematical model, has to perform three operations on three different data set: training, validation and testing. These three data sets are derived from the overall data set that is used as the prediction reference for the supervised learning of CGP.

Training

Training is the core part of the CGP evolution. During training, CGP will feed all data from the training data set and attempt to produce the output that matches the expectation as good as possible. However, this is the stage where the overfitting will occur. At this stage, the only data set that CGP is aware of is the training data set, which normally is 60% part of the original data set. As CGP adjusting its behaviour to match the expectation of the training data set, the model may fail to be applicable to any other data set and future predictions. Therefore, more operation will follow to attempt to suppress potential overfitting.

Validation

The chromosome with the best fitness rating during training will be selected for the validation process. The purpose of validation process is to assess the chromosome by using a small portion of data set, to check if it is overfitting or not. This small portion of data set is referred as validation data set, it typically takes 20% of the original data set. The validation data set will be used to execute the selected model. The fitness score along with the score from the testing data set will decide whether this chromosome is overfitting.

Testing

The remaining 20% of the original data set will be used for testing. This data set, along with the validation data set, simulates the scenario of blind-folded data set to the model, which is inevitable in real-life application where new data will be generated in daily basis. The fitness score from the testing will be compared with validation fitness score to decide whether the model is overfitting to training and validation data set.

After all three processes are completed, the best chromosome from each generation will have three fitness scores as a reference to choose the best generation from the whole evolution process. The process for selecting the optimal model as follows:

1. Select the first generation.
2. Register next generation’s training, validation and testing score.

3. Any of two conditions below will mark the generation as a better generation and will go through validation process:
    a. The training score is higher than previous high, or
    b. The training score remains the same but any of validation or testing score are higher than previous respective high
4. Validation process is done by calculating the difference between the training score and validation or testing score. Either one of the scores is lower more than 3 samples percentage of training score will mark as overfitting and will not use this generation.
5. Go to next generation and jump to step 3.

The process above will continue until the amount of generation reaches the pre-configured maximum generation number. Most of the results produced from this procedure are the best generation among the whole evolution process, in rare cases, less optimal generation is selected, in which case we need to manually pick the best generation. It is also a good practice to run the best generation to investigate the actual output.

6.1.3. K-Fold Cross Validation

In addition to CGP validation, k-fold cross validation is applied as well in the scenario that the sample size is very small. In this case, all three data sets are partitioned into two parts – keep and swap. This validation approach uses a convey-belt like mechanism that swap the data between data sets. Each 'fold' indicates an algorithm training session with shifted data set. Data shift operation will be executed after each fold, so each fold will use data set with same overall content but different arrangement. In terms of the whole training session, every single data in the data set will be used for training, validation and testing for at least once.

The number of data shift is determined that the training dataset is thoroughly updated once at the half of the number of ‘k’, so that for a whole algorithm training session, the training data set is updated twice. The calculation for the number of data shift is shown in equation 13:

$$N_{datashift} = \frac{N_{training} * 2}{k}$$

Equation 13 - Calculation of number of data shift per fold

in which $N_{training}$ is the number of samples in the training data set. The illustration of the data shift mechanism is shown in Fig. 44:

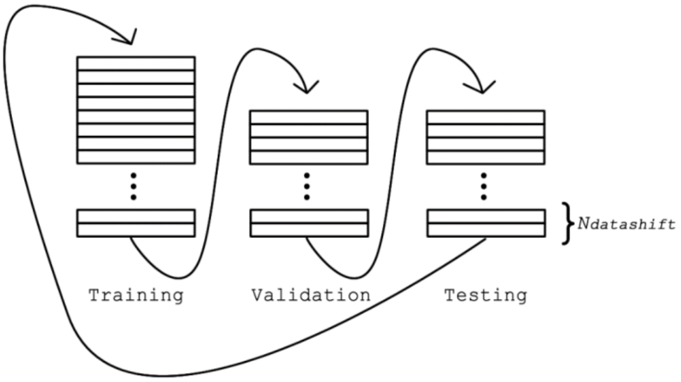


Figure 44 - Illustration of K-fold validation for CGP [52]

However, such convoy belt mechanism is only applicable for pair-wise classification as it ignores the amount of data for each class in each data set. If not handled properly, this mechanism will cause imbalance of number of data in each class across all data set, a more delicate data transfer is needed for multi-class classification task.

Each data set will be separated according to the data class. Assume a n-class problem, each data set will be separated into n parts. Each part will be used to conduct the similar data transfer between training, validation and testing, the number of data transfer is the same as in equation 13. After data transfer, multiple parts will merge again to form the new training, validation and testing data set, as a new fold.

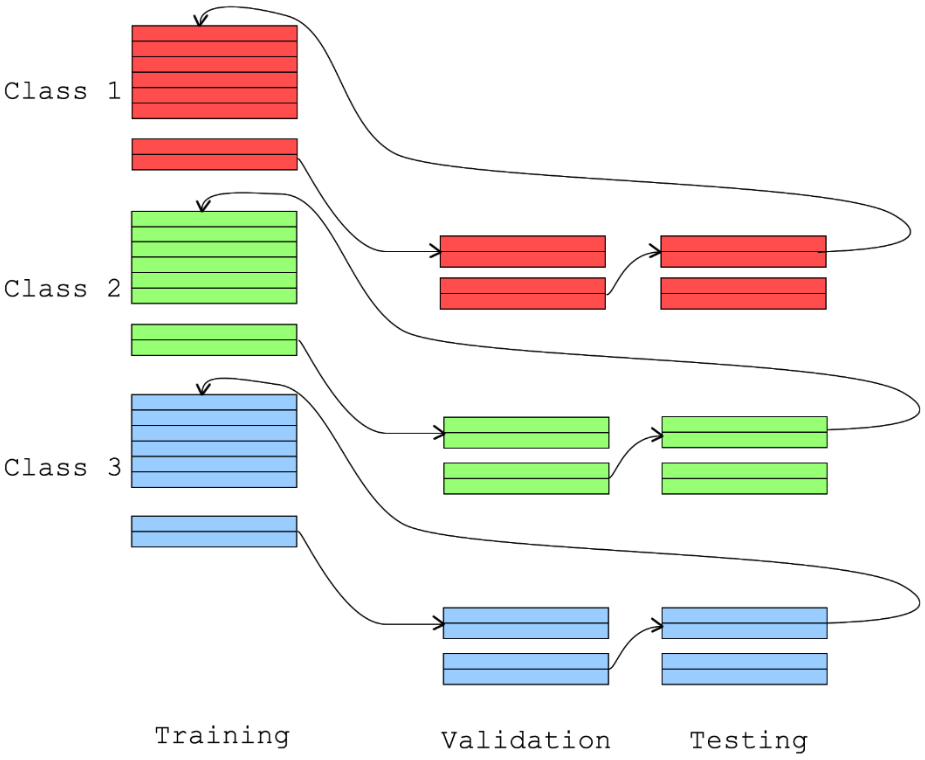


Figure 45 – Illustration of K-Fold to ensure the balance of class number

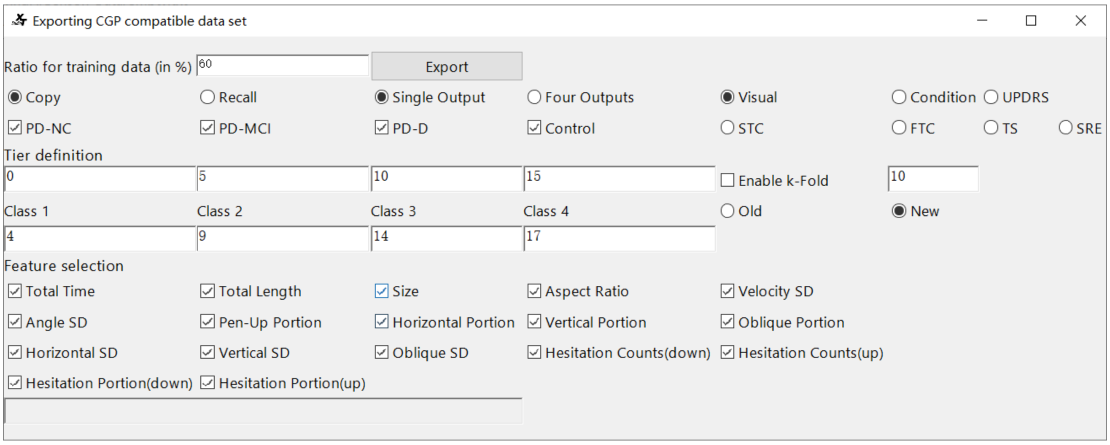
6.2. Cartesian Genetic Programming

Training session with CGP will be divided into four parts: CGP set up, training, testing, and verification.

CGP set up can be further divided into two parts: data parsing and parameter setting. Data parsing requires the data set satisfies the format requirement for CGP. A program written in Java is developed for automatic data parsing:

1. Circulate the raw data to provide the number of data;
2. Generate CGP data set header with features count, output numbers and number of data;
3. Select first Benson figure raw data;
4. Calculate its 17 features and convert all value data type to string;
5. Concatenate 17 string together, split by comma ‘,’;
6. Select next Benson figure and repeat step 4 until all figures are processed.

Parameter setting is also done by using an external program written in Java to avoid unnecessary compiling, to improve overall experiment sufficiency. Traditional parameter setting requires developer to adjust the parameter written in the CGP main program and recompile to apply the new parameter. This program can set the parameter within a comprehensive UI, then export a text file for the CGP main program. The CGP main program can import the text file to the program and assign all parameters to the CGP function. Such method can train the CGP with different configuration without any unnecessary code alternation and recompile. Fig. 46 & 47 show the sample interface of this program.



Ratio for training data (in %) 60

● Copy ○ Recall ● Single Output ○ Four Outputs ● Visual ○ Condition ○ UPDRS

☑ PD-NC ☑ PD-MCI ☑ PD-D ☑ Control ○ STC ○ FTC ○ TS ○ SRE

Tier definition

0 5 10 15 ☐ Enable k-Fold 10

Class 1 Class 2 Class 3 Class 4 ○ Old ● New

4 9 14 17

Feature selection

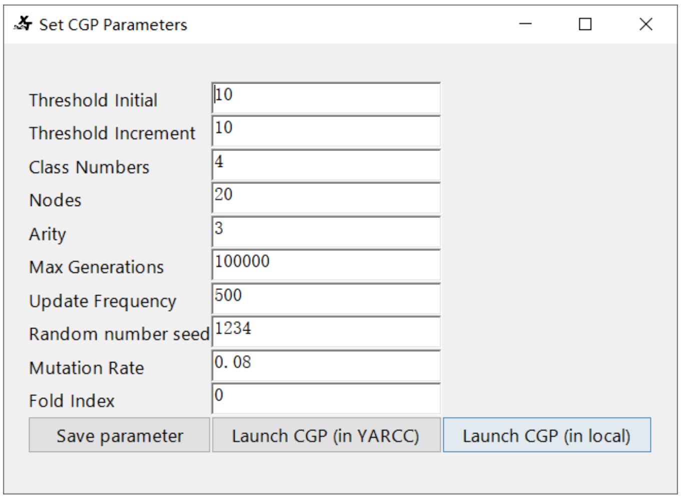
☑ Total Time ☑ Total Length ☑ Size ☑ Aspect Ratio ☑ Velocity SD

☑ Angle SD ☑ Pen-Up Portion ☑ Horizontal Portion ☑ Vertical Portion ☑ Oblique Portion

☑ Horizontal SD ☑ Vertical SD ☑ Oblique SD ☑ Hesitation Counts(down) ☑ Hesitation Counts(up)

☑ Hesitation Portion(down) ☑ Hesitation Portion(up)

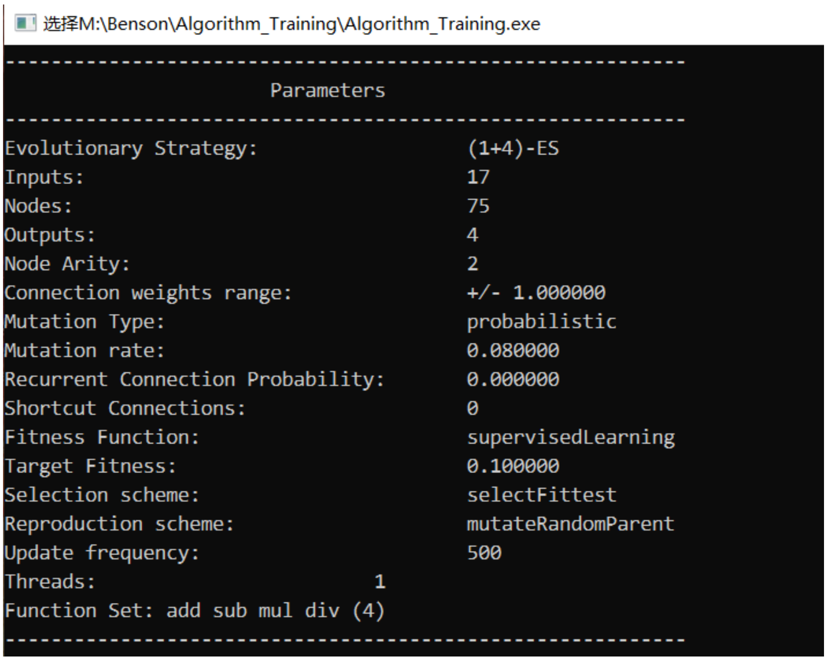
Figure 46 - Screenshot of CGP data exporter



Threshold Initial 10
Threshold Increment 10
Class Numbers 4
Nodes 20
Arity 3
Max Generations 100000
Update Frequency 500
Random number seed 1234
Mutation Rate 0. 08
Fold Index 0

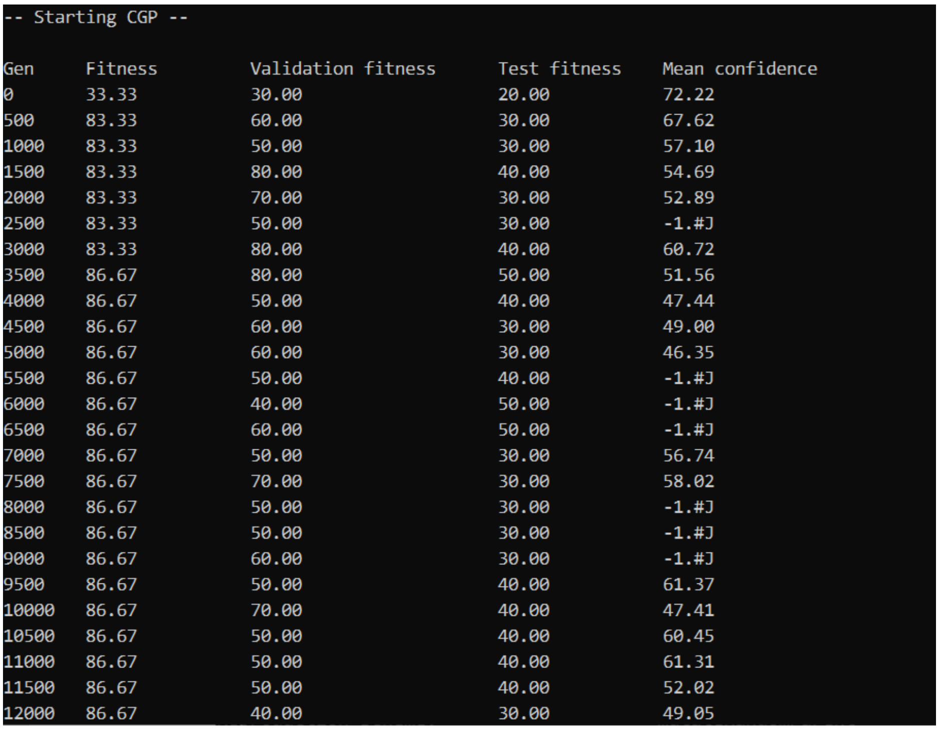
Figure 47 - Screenshot of CGP parameter tuner

In the training process, chromosome will be generated on generation basis. In this project, we used (1+4) Evolution Strategy (ES), in which each generation will produce 5 chromosomes – one parent chromosome and four children chromosomes, which are the mutant of the parent chromosome. Each chromosome is a description of the structure of a CGP, including node connection, node function, input/output node etc. Fitness of each chromosome will be calculated, the fittest chromosome is selected as the parent chromosome of next generation, and four new mutants of this chromosome will be generated, together forms a new generation, such process will repeat until the maximum generation number is reached. Fig. 48 shows a sample CGP task executes, with parameters listed at the beginning of the executable.



```
-----------------------------------------------------------
                       Parameters
-----------------------------------------------------------
Evolutionary Strategy:                  (1+4)-ES
Inputs:                                 17
Nodes:                                  75
Outputs:                                4
Node Arity:                             2
Connection weights range:               +/- 1.000000
Mutation Type:                          probabilistic
Mutation rate:                          0.080000
Recurrent Connection Probability:       0.000000
Shortcut Connections:                   0
Fitness Function:                       supervisedLearning
Target Fitness:                         0.100000
Selection scheme:                       selectFittest
Reproduction scheme:                    mutateRandomParent
Update frequency:                       500
Threads:                        1
Function Set: add sub mul div (4)
-----------------------------------------------------------
```

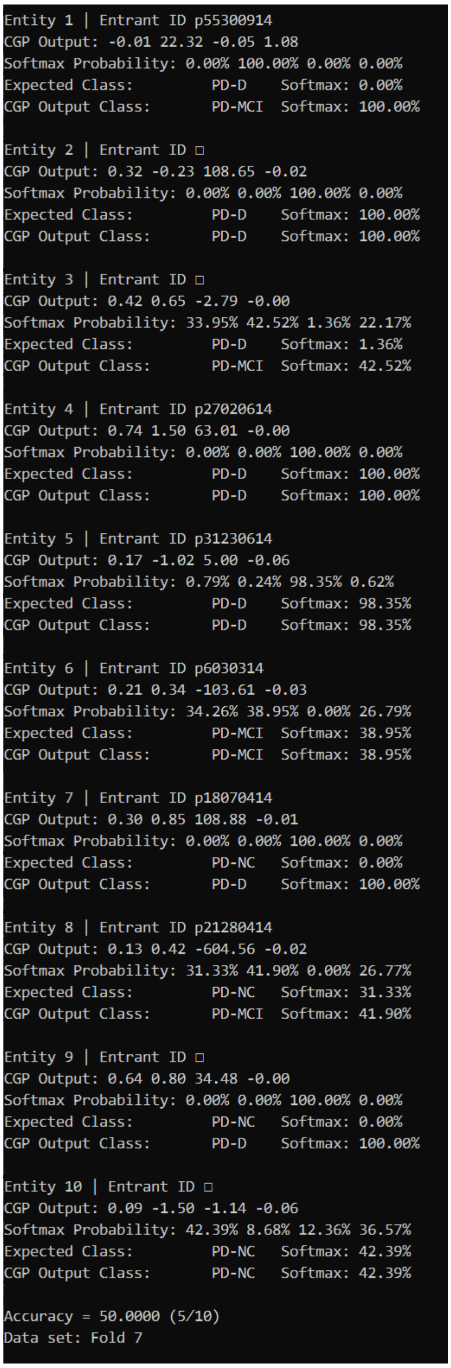
Figure 48 - CGP Training interface, showing the CGP parameters



```
-- Starting CGP --

Gen     Fitness         Validation fitness      Test fitness    Mean confidence
0       33.33           30.00                   20.00           72.22
500     83.33           60.00                   30.00           67.62
1000    83.33           50.00                   30.00           57.10
1500    83.33           80.00                   40.00           54.69
2000    83.33           70.00                   30.00           52.89
2500    83.33           50.00                   30.00           -1.#J
3000    83.33           80.00                   40.00           60.72
3500    86.67           80.00                   50.00           51.56
4000    86.67           50.00                   40.00           47.44
4500    86.67           60.00                   30.00           49.00
5000    86.67           60.00                   30.00           46.35
5500    86.67           50.00                   40.00           -1.#J
6000    86.67           40.00                   50.00           -1.#J
6500    86.67           60.00                   50.00           -1.#J
7000    86.67           50.00                   30.00           56.74
7500    86.67           70.00                   30.00           58.02
8000    86.67           50.00                   30.00           -1.#J
8500    86.67           50.00                   30.00           -1.#J
9000    86.67           60.00                   30.00           -1.#J
9500    86.67           50.00                   40.00           61.37
10000   86.67           70.00                   40.00           47.41
10500   86.67           50.00                   40.00           60.45
11000   86.67           50.00                   40.00           61.31
11500   86.67           50.00                   40.00           52.02
12000   86.67           40.00                   30.00           49.05
```

Figure 49 - CGP Training progress interface

After the training process is completed, the fittest generation will be selected to inspect its output result. In the training process, the CGP program will only print the fitness value rather than its output result, so it is essential to observe the chromosome's output result with test data set. If the best chromosome is not capable of classify most of the data correctly, different parameter will be used to generate another CGP model and assess its classification ability.



```
Entity 1 | Entrant ID p55300914
CGP Output: -0.01 22.32 -0.05 1.08
Softmax Probability: 0.00% 100.00% 0.00% 0.00%
Expected Class:         PD-D    Softmax: 0.00%
CGP Output Class:       PD-MCI  Softmax: 100.00%

Entity 2 | Entrant ID □
CGP Output: 0.32 -0.23 108.65 -0.02
Softmax Probability: 0.00% 0.00% 100.00% 0.00%
Expected Class:         PD-D    Softmax: 100.00%
CGP Output Class:       PD-D    Softmax: 100.00%

Entity 3 | Entrant ID □
CGP Output: 0.42 0.65 -2.79 -0.00
Softmax Probability: 33.95% 42.52% 1.36% 22.17%
Expected Class:         PD-D    Softmax: 1.36%
CGP Output Class:       PD-MCI  Softmax: 42.52%

Entity 4 | Entrant ID p27020614
CGP Output: 0.74 1.50 63.01 -0.00
Softmax Probability: 0.00% 0.00% 100.00% 0.00%
Expected Class:         PD-D    Softmax: 100.00%
CGP Output Class:       PD-D    Softmax: 100.00%

Entity 5 | Entrant ID p31230614
CGP Output: 0.17 -1.02 5.00 -0.06
Softmax Probability: 0.79% 0.24% 98.35% 0.62%
Expected Class:         PD-D    Softmax: 98.35%
CGP Output Class:       PD-D    Softmax: 98.35%

Entity 6 | Entrant ID p6030314
CGP Output: 0.21 0.34 -103.61 -0.03
Softmax Probability: 34.26% 38.95% 0.00% 26.79%
Expected Class:         PD-MCI  Softmax: 38.95%
CGP Output Class:       PD-MCI  Softmax: 38.95%

Entity 7 | Entrant ID p18070414
CGP Output: 0.30 0.85 108.88 -0.01
Softmax Probability: 0.00% 0.00% 100.00% 0.00%
Expected Class:         PD-NC   Softmax: 0.00%
CGP Output Class:       PD-D    Softmax: 100.00%

Entity 8 | Entrant ID p21280414
CGP Output: 0.13 0.42 -604.56 -0.02
Softmax Probability: 31.33% 41.90% 0.00% 26.77%
Expected Class:         PD-NC   Softmax: 31.33%
CGP Output Class:       PD-MCI  Softmax: 41.90%

Entity 9 | Entrant ID □
CGP Output: 0.64 0.80 34.48 -0.00
Softmax Probability: 0.00% 0.00% 100.00% 0.00%
Expected Class:         PD-NC   Softmax: 0.00%
CGP Output Class:       PD-D    Softmax: 100.00%

Entity 10 | Entrant ID □
CGP Output: 0.09 -1.50 -1.14 -0.06
Softmax Probability: 42.39% 8.68% 12.36% 36.57%
Expected Class:         PD-NC   Softmax: 42.39%
CGP Output Class:       PD-NC   Softmax: 42.39%

Accuracy = 50.0000 (5/10)
Data set: Fold 7
```

Figure 50 - Example of detailed output from CGP for individual figures in test data set

Due to the difficulty and limited time on recruiting and testing all PD patients, the size of the data set is relatively small for a machine learning task. Therefore, k-fold cross validation is applied for verification. Each fold is formed by swapping a fixed number of data across all three data sets. For this project, verification is split into two part – same fold with multiple random number seed, same random number seed with different fold. Each random number seed will train 11 data sets

– one original and ten folds. Same procedure will be repeated for 10 times with 10 different number seeds. At the end of the verification process, the mean and the standard deviation of the fitness will be calculated to give an overlook of the performance of this CGP model.

The procedure of parameter tuning of CGP and evaluating the performance of a CGP model as follows:

1. Supply CGP API with compatible data set which contains extracted features
2. Set up CGP parameter
3. Train the algorithm
4. Select the fittest generation, if the overall fitness is lower than 70% or the CGP cannot produce generation with fitness over 70% consistently, alter the CGP model by setting different parameters and repeat step 3
5. Use next fold of data to train the algorithm, repeat step 5 until the fold runs out
6. Set another random number seed, repeat step 5 until ten random number seeds are all used
7. Calculate the overall fitness by using the mean of fitness from all folds and random number seeds.

To lower the difficulty for the algorithm, each PD stage will be paired for initial training, therefore, the classification task for CGP will be simplified from three-class classification task to two-class task. Three PD stages will provide three possible pairs to train the CGP. Then the CGP will be trained using overall data to observe its performance.

Because Benson copy and recall task focuses on different principle of PD symptoms, they are treated as separate tasks. Therefore, data from copy and recall will be trained separately to not confuse the algorithm as the main objective does not include distinguishing results from different tasks. Although the workload is doubled as a consequence of separating data set.

As the Node Weighting Classifier will give multiple outputs to determine the maximum output node index as the output class, softmax function will be used to calculate the confidence of CGP's output result. Softmax function will scale all vector elements to (0, 1) range so that each individual numeric value can be used as probabilities. In this project, this function is used to determine the confidence of the CGP on certain classification result. The softmax function can be expressed as in equation 14:

$$\sigma(\boldsymbol{Z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Equation 14 - Softmax function

in which Z_i is the element of the input vector, N is the number of outputs for i = 1, … , N. For example, table 8 shows the example of CGP output confidence.

Table 8 - Comparison in softmax output of two matching data entries with CGP classification result[3].

	Output 1	Output 2	Output 3	Output 4
CGP Output	7.95	0.16	**9.72**	7.95
e^x	2835.575	1.174	16647.245	2835.575
$\sum_{x=1}^{4} e^x$	22319.569			
Softmax Output	12.70%	0.07%	**74.56%**	12.70%
Expected Class	PD-D (Maximum output value at Output 3)			
CGP Result	PD-D (Maximum output value at Output 3)			

---

[3] Output 4 is intended for HC entry while this training session has no HC entry, causing the CGP using other output's result for this output entity.

	Output 1	Output 2	Output 3	Output 4
CGP Output	**2.28**	0.14	2.10	2.28
e^x	9.777	1.150	8.166	9.777
$\sum_{x=1}^{4} e^x$	28.87			
Softmax Output	**33.89%**	3.99%	28.22%	33.89%
Expected Class	PD-NC (Maximum output value at Output 1)			
CGP Result	PD-NC (Maximum output value at Output 1)			

Both tables show the output details from two data sets with matching classification result. The first sample shows a 74.56% output confidence while the second one has only 33.89%. In real life applications, this output result shows the accuracy of the algorithm model along with its confidence on a particular classification result.

Confidence calculation can also be applied with STC fitness function with a simpler approach. The essence of confidence calculation is to evaluate the free space between the output value and the pre-defined range margin. In STC, this can be simplified as the distance between the output value and any one of the threshold margins. The confidence level from STC is related with the position of the actual output within its threshold interval. The closer the distance between the output value and the threshold interval margin indicates lower confidence. Maximum confidence is expected when the data point is in the middle of the interval. Therefore, the confidence level is proportional to the value delta between the output and the middle point. Assume the threshold margin are $T_a \; and \; T_b$, the middle point can be represented as in equation 15:

$$T_m = \frac{T_a + T_b}{2}$$

Equation 15 - Threshold middle point calculation

Assume the CGP output of sample i is O_i, the value difference between the CGP output and the middle point is:

$$\Delta_i = |O_i - T_m|$$

Equation 16 - Distance between CGP output and threshold middle point

Therefore, the confidence level can be represented as:

$$\sigma_i = 1 - \frac{\Delta_i}{|T_m - T_x|}$$

Equation 17 - STC confidence level calculation

In this equation T_x can be either T_a or T_b as the lower part of the equation is the value difference between the middle point and any one of the threshold margins.

6.3. Support Vector Machine

For SVM training task, only two data set will be used – PD-NC/MCI/D copy and recall. Pair-wise classification is used for feature verification and is applied on CGP only. SVM training session will apply k-fold cross validation only, as the self-validation is included in the MATLAB classifier learner package. The validation mechanism in the software package follows:

1. Specify the number of k
2. Scramble and split the whole data set into k equal parts.
3. Combine random small parts of data set into one data set. The number of small parts is (k-1)
4. Train the SVM with (k-1) data set
5. Test the SVM with the remaining 1 data set to simulate unknown data prediction scenario.
6. Iterate step 2~5 for k times.

After the training session, accuracy is presented with the best model which uses the overall data set. Model can be further investigated by using scatter plot, confusion matrix and ROC curve.

Result representation is consisted by five parts: SVM kernel function, accuracy, true/false positive rate and area under curve, which will all be explained in this section.

SVM kernel function

As explained in section 4.2.1, for this project, six SVM kernel functions will be used to compare their performance under the same configuration.

Accuracy

This will represent the classification accuracy of each trained model against the overall data set.

True/False Positive Rate (T/FP Rate)

This feature is used to observe the detailed accuracy in each class, rather than overall accuracy.

True/False positive rate is used to present the accuracy distribution for a dual-class problem but can also be adjusted for multi-class classification. Assume a dual-class problem, there will be two classes, labelled as positive and negative. The number of true positive (TP) indicates the correct number of predictions for a positive class data. False positive (FP) indicates the incorrect number of predictions for a negative class data. Table 9 shows a confusion matrix that used to demonstrate the concept of true/false positive/negatives by using a 'cried-wolf' scenario, taken from Google's machine learning tutorial [70].

Table 9 - Confusion matrix to demonstrate T/F P/N [70]

True Positive (TP): ● Reality: A wolf threatened ● Shepherd said: “Wolf” ● Outcome: Shepherd is a hero	**False Positive (FP):** ● Reality: No wolf threatened ● Shepherd said: "Wolf." ● Outcome: Villagers are angry at shepherd for waking them up.
False Negative (FN): ● Reality: A wolf threatened. ● Shepherd said: "No wolf." ● Outcome: The wolf ate all the sheep.	**True Negative (TN):** ● Reality: No wolf threatened. ● Shepherd said: "No wolf." ● Outcome: Everyone is fine.

With the concept of T/F P/N explained, we can then use them to calculate the T/FP rate to indicate the prediction accuracy in equation 18:

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

$$\mathrm{FPR} = \frac{FP}{FP + TN}$$

Equation 18 - Calculation of True Positive Rate (TPR) and False Positive Rate (FPR) [71]

These two values can be used to plot Receiver Operating Characteristic Curve (ROC Curve), which is a graph that demonstrates the model classification performance at all configurations [71], which is used to select the best model. The best model should have TPR as high as possible with FPR as low as possible. The performance across all model configuration can be measured by the area under the ROC curve [71], or Area Under Curve (AUC).

Area Under Curve (AUC)

This value gives and overall representation of the ROC curve that generated from each training session. Fig. 51 & 52 demonstrate how T/FPR forms the ROC curve and how the AUC can be used to represent the overall performance across all models in one training session.

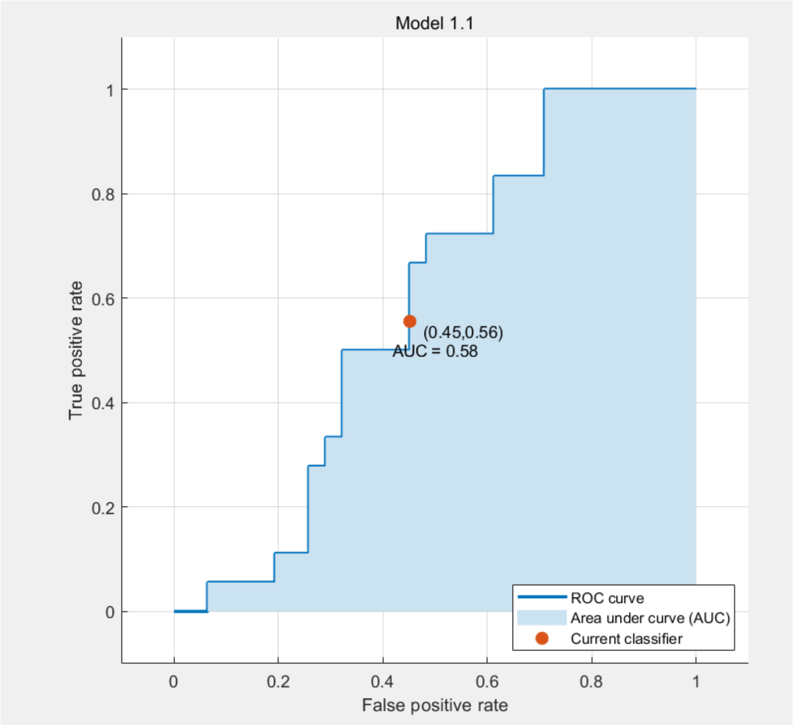


Figure 51 - An SVM model with an AUC of 0.58, classification accuracy 53.1%

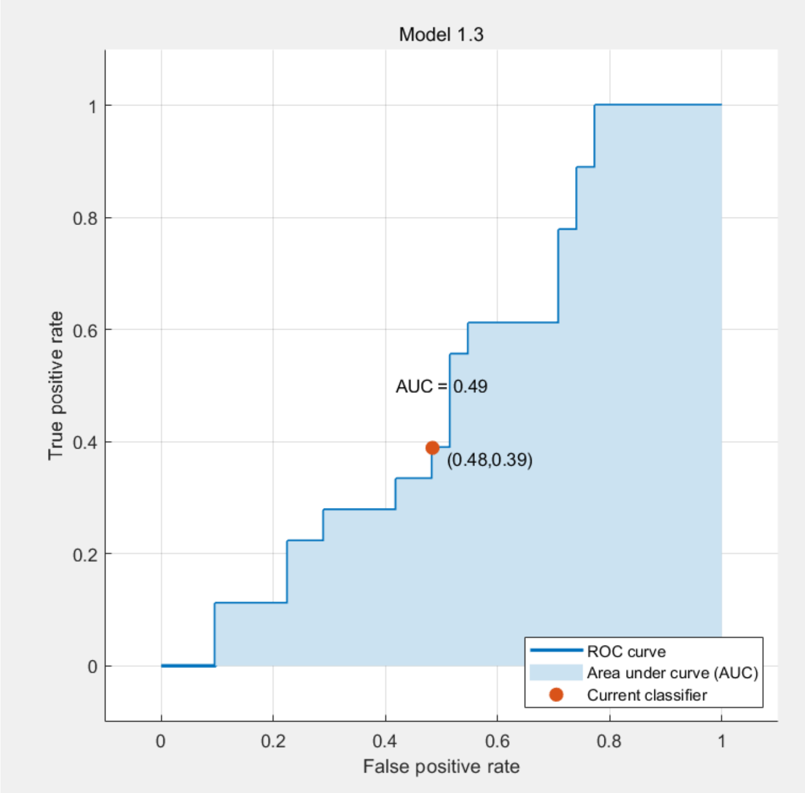


Figure 52 - An SVM model with and AUC of 0.49, classification accuracy 38.8%

7. Experiment Results

This section presents the test result from various tests, including dual classification by CGP, multi-class classification by CGP and SVM. This section introduces how extracted features are verified through simple CGP execution with multiple validation strategies and using multi-class classification to generate the overall classification result. In the end, the classification result and implementation method will be compared for SVM and CGP and conclude which model suites best for such research project. The parameter used to generate those results and full result sheets can be found in Appendix A and B.

7.1. Cartesian Genetic Programming – Dual Classification Result

The objective of training the CGP with only two classes at a time is to authenticate the features extracted from the raw data, to ensure that those features are capable of representing the all necessary aspects of the original figure. By doing dual classification, we can minimise the workload of feature authentication as the classification problem for the algorithm is significantly simplified. However, due to the decrease of sample numbers with less classes, additional validation must be applied to ensure the consistency of the classification result. Table 10-15 are the detailed classification result with verification using dual-classification.

7.1.1. PD-NC/MCI Copy/Recall classification result

Table 10 - PD-NC/MCI Copy dataset classification result, K = 10

Pair: PD-NC/PD-MCI			Drawing Task: Copy			
Dataset	Training (14 entries)		Validation (5 entries)		Test (6 entries)	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Iteration 1	99.351%	0.0205	83.637%	0.0771	84.846%	0.0857
Iteration 2	99.351%	0.0205	89.091%	0.0996	80.301%	0.0642
Iteration 3	100.00%	0	83.636%	0.0771	89.393%	0.1071
Iteration 4	100.00%	0	81.818%	0.0575	83.331%	0.0711
Iteration 5	100.00%	0	83.636%	0.0771	86.361%	0.0643
Iteration 6	100.00%	0	85.455%	0.0891	83.332%	0.1005
Iteration 7	99.351%	0.0205	85.455%	0.0891	83.332%	0.1005
Iteration 8	100.00%	0	87.273%	0.0962	83.329%	0.1005
Iteration 9	99.351%	0.0205	83.636%	0.0771	83.332%	0.1005
Iteration 10	100.00%	0	85.455%	0.0891	81.817%	0.1113
Mean	99.740%	0.0082	84.909%	0.0829	83.937%	0.0906

Table 11 - PD-NC/MCI Recall dataset classification result, K = 10

Pair: PD-NC/PD-MCI			Drawing Task: Recall			
Dataset	Training (15 entries)		Validation (5 entries)		Test (6 entries)	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Iteration 1	100%	0	87.273%	0.0962	81.816%	0.0857
Iteration 2	100%	0	85.455%	0.0891	86.362%	0.0958
Iteration 3	100%	0	87.273%	0.0962	84.847%	0.1113
Iteration 4	100%	0	90.909%	0.0996	83.331%	0.0711
Iteration 5	100%	0	90.909%	0.0996	86.362%	0.0958
Iteration 6	100%	0	85.455%	0.0891	84.847%	0.1113
Iteration 7	100%	0	85.455%	0.0891	86.362%	0.0958
Iteration 8	100%	0	89.091%	0.0996	81.817%	0.1113
Iteration 9	100%	0	83.636%	0.0771	84.847%	0.1113
Iteration 10	100%	0	89.091%	0.0996	83.331%	0.0711
Mean	100%	0	87.455%	0.0935	84.392%	0.0961

7.1.2. PD-NC/D Copy/Recall classification result

Table 12- PD-NC/D Copy dataset classification result, K = 10

Pair: PD-NC/PD-D			Drawing Task: Copy			
Dataset	Training (24 entries)		Validation (9 entries)		Test (9 entries)	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Iteration 1	93.939%	0.0544	77.78%	0.0947	72.73%	0.1097
Iteration 2	95.454%	0.0278	74.75%	0.0958	69.699%	0.0958
Iteration 3	95.074%	0.0429	77.78%	0.0821	69.7%	0.0958
Iteration 4	93.184%	0.0647	70.71%	0.0857	68.689%	0.1143
Iteration 5	94.696%	0.0667	69.699%	0.1169	64.649%	0.1325
Iteration 6	93.94%	0.0761	72.73%	0.0989	66.669%	0.1254
Iteration 7	94.696%	0.0438	75.76%	0.1143	70.71%	0.0857
Iteration 8	93.94%	0.0448	69.695%	0.1429	65.66%	0.0998
Iteration 9	93.56%	0.0625	74.75%	0.0958	67.68%	0.0881
Iteration 10	95.834%	0.0355	73.74%	0.0979	63.638%	0.1069
Mean	94.432%	0.0519	73.739%	0.1025	67.982%	0.1054

Table 13- PD-NC/D Recall dataset classification result, K = 10

Pair: PD-NC/PD-D			Drawing Task: Recall			
Dataset	Training (25 entries)		Validation (9 entries)		Test (9 entries)	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Iteration 1	96.364%	0.0317	73.74%	0.1088	67.679%	0.1000
Iteration 2	95.636%	0.0577	70.71%	0.1088	65.66%	0.1000
Iteration 3	96.727%	0.0333	74.75%	0.0833	67.68%	0.0880
Iteration 4	93.818%	0.0313	69.7%	0.1169	66.669%	0.1254
Iteration 5	96%	0.0566	67.679%	0.1204	64.65%	0.0639
Iteration 6	92.364%	0.0648	68.689%	0.1040	65.658%	0.1378
Iteration 7	94.909%	0.0420	73.74%	0.1089	63.643%	0.0833
Iteration 8	95.273%	0.0445	67.679%	0.1107	66.667%	0.0947
Iteration 9	97.818%	0.0262	73.74%	0.1278	72.73%	0.0728
Iteration 10	97.455%	0.0192	73.74%	0.0857	63.639%	0.1429
Mean	95.636%	0.0408	71.417%	0.1075	66.468%	0.1009

7.1.3. PD-MCI/D Copy/Recall classification result

Table 14 - PD-MCI/D Copy dataset classification result, K = 10

Pair: PD-MCI/PD-D			Drawing Task: Copy			
Dataset	Training (18 entries)		Validation (6 entries)		Test (7 entries)	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Iteration 1	99.495%	0.0160	92.423%	0.0830	84.414%	0.0954
Iteration 2	96.968%	0.0495	90.907%	0.0830	81.817%	0.1071
Iteration 3	97.978%	0.0267	84.846%	0.0857	85.712%	0.0861
Iteration 4	99.495%	0.0160	92.423%	0.0830	87.011%	0.0954
Iteration 5	99.495%	0.0160	89.392%	0.0802	85.712%	0.0861
Iteration 6	98.485%	0.0343	92.423%	0.0830	79.22%	0.0936
Iteration 7	100%	0	92.423%	0.0830	84.415%	0.1132
Iteration 8	99.495%	0.0160	86.362%	0.0958	88.31%	0.1023
Iteration 9	99.495%	0.0160	89.392%	0.0802	83.115%	0.1190
Iteration 10	99.495%	0.0160	90.907%	0.0830	89.607%	0.0636
Mean	99.040%	0.0206	90.150%	0.0840	84.933%	0.0962

Table 15 - PD-MCI/D Recall dataset classification result, K = 10

Pair: PD-MCI/PD-D			Drawing Task: Recall			
Dataset	Training (18 entries)		Validation (6 entries)		Test (7 entries)	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Iteration 1	99.495%	0.0160	86.361%	0.0643	81.816%	0.1232
Iteration 2	98.989%	0.0214	83.331%	0.0711	81.816%	0.0123
Iteration 3	98.485%	0.0343	81.816%	0.0857	77.921%	0.0711
Iteration 4	97.474%	0.0495	86.362%	0.0958	76.622%	0.0687
Iteration 5	99.495%	0.0160	83.331%	0.0711	81.816%	0.0881
Iteration 6	100%	0	86.361%	0.0643	83.115%	0.1022
Iteration 7	98.989%	0.0214	87.876%	0.0742	76.622%	0.1259
Iteration 8	98.485%	0.0343	81.815%	0.0479	79.219%	0.0711
Iteration 9	99.495%	0.0160	84.845%	0.0479	83.115%	0.1337
Iteration 10	99.496%	0.0160	86.361%	0.0643	74.025%	0.0821
Mean	99.040%	0.0225	84.846%	0.0687	79.609%	0.0989

7.2. Cartesian Genetic Programming – Overall Classification Result

This section presents the overall classification result from the CGP by applying all feature data that extracted from the raw data. This section will present the result in the way which will list four attributes – mean, best, worst and standard deviation of four aspects from a model – training, validation & testing accuracy as well as its confidence rating. Due to the equation of softmax function and the actual output value from the CGP, some of the confidence rating is uncalculatable by the computer as it exceeds the 64-bit binary limit. In this case, the presented data will be marked and note the amount of data that are emitted due to this inevitable error. Table 16-25 presents the detailed classification result from CGP overall classification training.

7.2.1. Overall Copy Classification Result

Table 16 – Training score analysis of multi-class classification, K = 10, Copy

	Training			
	Mean	Best	Worst	S.D.
Iteration 1	85.895%	100%	65.52%	0.1013
Iteration 2	84.955%	96.55%	75.86%	0.0611
Iteration 3	84.326%	96.55%	75.86%	0.0741
Iteration 4	84.639%	96.55%	72.41%	0.0711
Iteration 5	88.088%	100%	72.41%	0.1044
Iteration 6	89.341%	100%	75.86%	0.0798
Iteration 7	89.029%	100%	65.52%	0.0951
Iteration 8	89.342%	100%	65.52%	0.0991
Iteration 9	90.282%	100%	72.41%	0.0962
Iteration 10	89.03%	93.1%	82.76%	0.0355

Table 17 – Validation score analysis of multi-class classification, K = 10, Copy

	Validation			
	Mean	Best	Worst	S.D.
Iteration 1	64.545%	80%	50%	0.0988
Iteration 2	59.091%	80%	40%	0.1083
Iteration 3	60%	70%	40%	0.0953
Iteration 4	63.636%	90%	50%	0.1149
Iteration 5	57.273%	70%	40%	0.0962
Iteration 6	63.636%	80%	40%	0.1367
Iteration 7	58.182%	80%	40%	0.1267
Iteration 8	62.727%	90%	40%	0.1543
Iteration 9	65.455%	70%	40%	0.0891
Iteration 10	61.818%	80%	40%	0.0935

Table 18 – Testing score analysis of multi-class classification, K = 10, Copy

	Testing			
	Mean	Best	Worst	S.D.
Iteration 1	64.545%	90%	40%	0.1157
Iteration 2	63.636%	80%	50%	0.0881
Iteration 3	60.909%	70%	50%	0.0793
Iteration 4	63.636%	80%	40%	0.1149
Iteration 5	62.727%	80%	30%	0.1420
Iteration 6	61.818%	70%	50%	0.0833
Iteration 7	60.909%	80%	30%	0.1379
Iteration 8	63.636%	80%	50%	0.0979
Iteration 9	64.545%	90%	50%	0.1076
Iteration 10	71.818%	80%	60%	0.0575

Table 19 – Confidence score analysis of multi-class classification, K = 10, Copy

	Confidence			
	Mean	Best	Worst	S.D.
Iteration 1	60.184%	99.67%	35.08%	0.1879
Iteration 2 (1)[4]	49.131%	76.20%	27.38%	0.1483
Iteration 3 (1)	52.245%	89.90%	29.29%	0.1889
Iteration 4 (2)	67.871%	92.86%	35.27%	0.2108
Iteration 5	49.170%	84.70%	28.21%	0.1833
Iteration 6	56.017%	82.72%	28.81%	0.1564
Iteration 7	55.355%	85.22%	31.43%	0.1720
Iteration 8	46.217%	62.87%	27.80%	0.1056
Iteration 9 (1)	58.889%	90.42%	30.30%	0.2116
Iteration 10	44.061%	82.31%	25.93%	0.1731

Table 20 - Overview of classification result from all attributes, Copy

	All Attributes		
	Mean	**Best**	**Worst**
Training	87.493%	100%	65.52%
Validation	61.636%	90%	40%
Testing	63.818%	90%	30%
Overall Average	70.982%	93.333%	45.173%
Confidence	53.914%	99.67%	25.93%

---

[4] Number in bracket indicates the number of data which emitted due to computer's calculation limit

7.2.2. Overall Recall Classification Result

Table 21 - Training score analysis of multi-class classification, K = 10, Recall

	Training			
	Mean	Best	Worst	S.D.
Iteration 1	94.334%	100%	90%	0.0431
Iteration 2	90.333%	100%	73.33%	0.0749
Iteration 3	91.333%	100%	70%	0.1158
Iteration 4	93%	100%	76.67%	0.0649
Iteration 5	89.333%	100%	76.67%	0.0888
Iteration 6	89.001%	100%	80%	0.0760
Iteration 7	95%	100%	86.67%	0.0050
Iteration 8	89.667%	100%	70%	0.1031
Iteration 9	90.666%	100%	73.33%	0.0848
Iteration 10	90.332%	100%	80%	0.0619

Table 22 - Validation score analysis of multi-class classification, K = 10, Recall

	Validation			
	Mean	Best	Worst	S.D.
Iteration 1	68%	90%	50%	0.0982
Iteration 2	63%	90%	20%	0.2115
Iteration 3	59%	90%	40%	0.1328
Iteration 4	60%	90%	20%	0.2054
Iteration 5	64%	90%	40%	0.1635
Iteration 6	59%	80%	20%	0.1700
Iteration 7	69%	100%	50%	0.1601
Iteration 8	69%	90%	40%	0.1328
Iteration 9	65%	80%	40%	0.1214
Iteration 10	67%	80%	60%	0.0751

Table 23 - Testing score analysis of multi-class classification, K = 10, Recall

	Testing			
	Mean	Best	Worst	S.D.
Iteration 1	74%	90%	50%	0.1191
Iteration 2	60%	80%	30%	0.1662
Iteration 3	64%	80%	50%	0.0934
Iteration 4	67%	90%	50%	0.1221
Iteration 5	65%	90%	50%	0.1214
Iteration 6	66%	80%	40%	0.1293
Iteration 7	71%	100%	40%	0.1483
Iteration 8	65%	70%	50%	0.0820
Iteration 9	62%	80%	50%	0.0905
Iteration 10	68%	90%	60%	0.0982

Table 24 - Confidence score analysis of multi-class classification, K = 10, Recall

	Confidence			
	Mean	Best	Worst	S.D.
Iteration 1	58.406%	91.23%	35.14%	0.1607
Iteration 2	62.161%	75.57%	47.95%	0.1025
Iteration 3 (2)	58.479%	75.25%	40.56%	0.2543
Iteration 4	68.844%	94.27%	45.23%	0.1440
Iteration 5 (2)	48.419%	72.29%	29.93%	0.2459
Iteration 6 (1)	54.882%	72.33%	40.65%	0.1933
Iteration 7 (1)	56.458%	82.26%	41.58%	0.2247
Iteration 8 (2)	51.193%	70.16%	27.95%	0.2352
Iteration 9 (2)	60.958%	84.54%	38.22%	0.2761
Iteration 10 (1)	56.306%	87.29%	30.36%	0.2724

Table 25 - Overview of classification result from all attributes, Recall

	All Attributes		
	Mean	**Best**	**Worst**
Training	91.299%	100%	70%
Validation	64.3%	100%	20%
Testing	66.2%	100%	30%
Overall Average	73.933%	100%	40%
Confidence	57.610%	94.27%	27.95%

In conclusion, the overall training session produced an overall satisfactory result, with maximum accuracy of 90%~100% across training, validation and testing data set. The average overall classification accuracy is 70.982% for copy and 73.933% for recall. Minimum validation and testing accuracies are only up to 40%, which means some of the model did not pass the validation, as an indication of overfitting, which is an expected situation as the small size of the given data set. However, most of the model passed the validation with satisfactory accuracies.

7.3. Support Vector Machine

The result from SVM training session are generally lower than CGP, with the maximum accuracy of 57.1% for overall copy data set, with PCA set as 3, and maximum of 54.0% for recall data set, with PCA set as 5. Table 26 & 27 show the detailed SVM training result with different kernel functions as comparisons.

Table 26 – SVM classification result on overall copy data set, PCA 3, K = 5

SVM Kernel Function	**Accuracy**	**TP Rate**	**FP Rate**	**AUC**
Linear	57.1%	0.72	0.48	0.62
Quadratic	40.8%	0.5	0.48	0.54
Cubic	51.0%	0.61	0.42	0.6
Fine Gaussian	55.1%	0.56	0.35	0.54
Medium Gaussian	46.9%	0.06	0.13	0.57
Coarse Gaussian	49.0%	0	0	0.59

Table 27 – SVM classification result on overall recall data set, PCA 5, K = 5

SVM Kernel Function	Accuracy	TP Rate	FP Rate	AUC
Linear	46.0%	0.47	0.39	0.55
Quadratic	42.0%	0.47	0.29	0.63
Cubic	48.0%	0.63	0.26	0.74
Fine Gaussian	52.0%	0.16	0.06	0.63
Medium Gaussian	54.0%	0.58	0.29	0.65
Coarse Gaussian	48.0%	0	0	0.54

7.4. Overall

As expected, SVM classification yields poorer result than CGP in the determination of PD stage with raw drawing data from Benson drawing test. The analysis of this result comparison are mainly two points: Poor clustering in features and lack of samples. Fig. 53 shows the data clustering situation of the extracted data set.

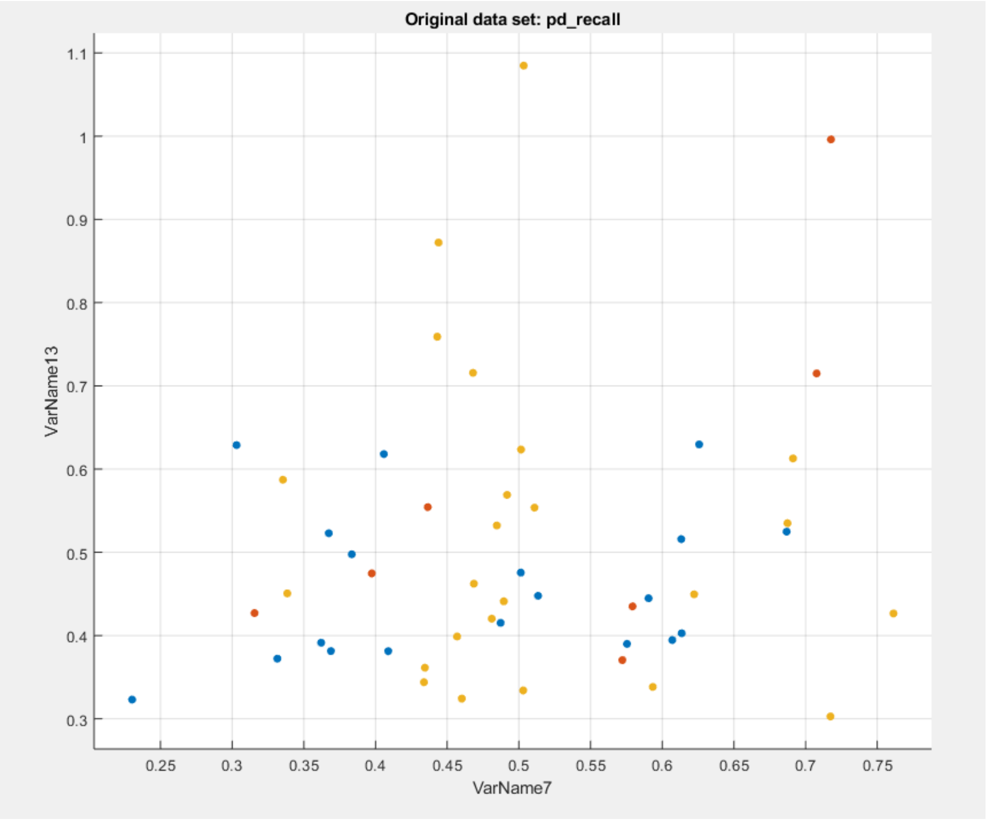


Figure 53 – Data clustering situation of PD Recall data set

As shown in Fig. 53, features that extracted from the raw data has very poor clustering for the SVM to find an optimal solution. It is also clear that scatter plot with SVM will be too difficult for non-computer researcher to investigate how the algorithm model uses those features.

Also, from Fig. 54, we can observe that the SVM managed to ‘cheat’ by ignoring all PD-MCI entries. Because PD-MCI only has 7 entries, ignoring all of them may yields better accuracy while lowering the difficulty of the classification problem. For this model, attempt to classify PD-MCI entries may compromise accuracy from other classes, which may yield lower overall accuracy.

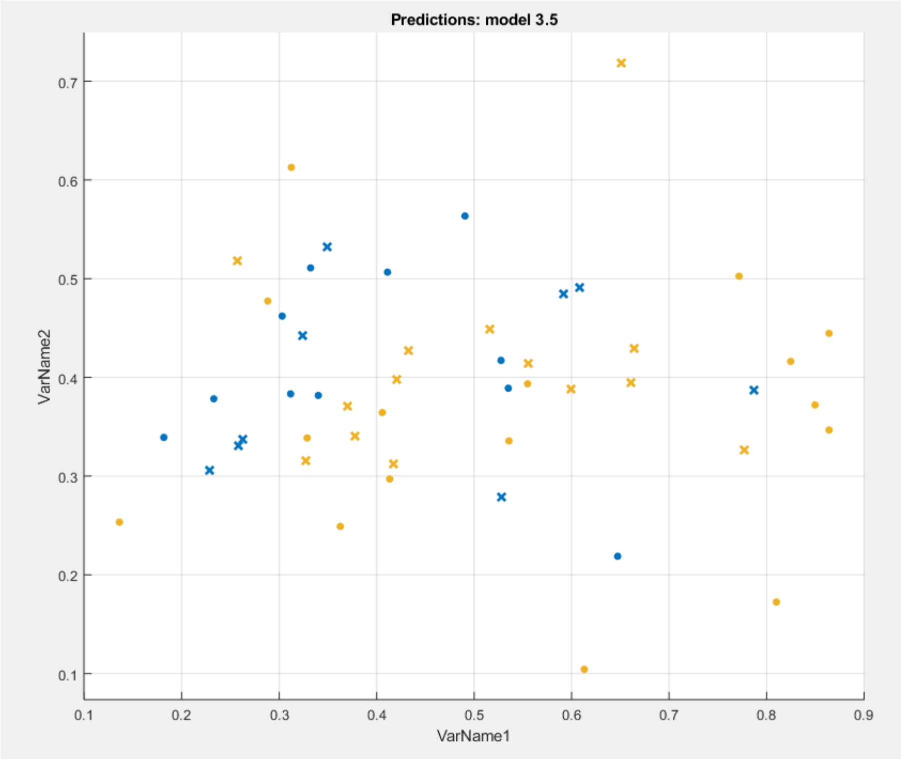


Figure 54 – An SVM model showing ‘cheating’ by emitting all PD-MCI entries

Table 28 – Comparison between SVM and CGP classification result

	SVM - Copy	CGP - Copy	SVM - Recall	CGP - Recall
Mean	49.983%	70.982%	48.333%	73.933%
Best	57.1%	93.333%	54.0%	100%
Worst	40.8%	45.173%	48.0%	40%

By comparing the overall accuracy with CGP and SVM in table 28, we can also see that CGP surpassed SVM in every training session in terms of classification accuracy. With maximum SVM accuracy of 57.1% for copy, while CGP's maximum overall accuracy of 93.33% for copy. The only accuracy that SVM is better than CGP is the worst model accuracy in recall data set classification task, which SVM has 48% while CGP only has 40%.

8. Further Works

This section explores further ideas that inspired by the challenged and unsolved problems from this research project.

8.1. Deep Learning and Cartesian Genetic Programming

One possible way to compensate the disadvantage on image recognition by CGP is to combine the deep learning technique on computer vision (CV) with CGP. The general idea is to extract image features using CV, combine with movement features from the patients, then feed all the features together into CGP. This will involve the use of raw image file, rather than relying on raw data alone. Possible approach is to analyse the image file pixel-by-pixel, using integer value to represent the structure of the image.

Similar to all image pre-processing for deep learning, figure will be regenerated from the raw data, the image file will be scaled into same smaller size for better computational efficiency and unifying data set, for example, 64x64 resolution will provide 4096 pixels of information from an image file. In terms of image structure, Benson figure only contains line on and off, the pixel can be represented as 0 or 1 to indicate whether the pen track has covered certain pixel or not. Fig. 55-58 shows the initial idea to transform image to binary form data.

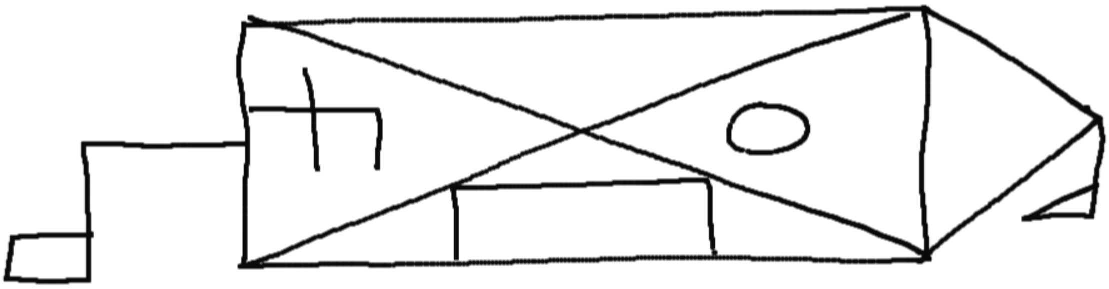


Figure 55 - An original figure regenerated from the raw data for demonstration

A 64x64 image can be further modularised into 64x8x8 smaller parts. Each part is an 8x8 image, with each pixel line provides 8 data points from pixel information. Each pixel line can be represented in binary form and therefore can be converted to decimal, so that each pixel line provides a single integer number as one feature. Each smaller block will then provide eight values, so the overall image contains 64x8 = 512 features.

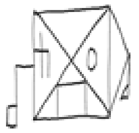


Figure 56 - 64x64 compressed version of Fig.55

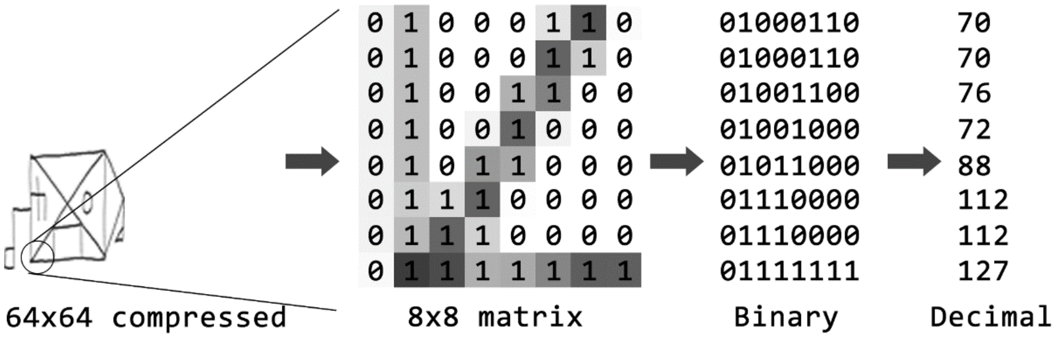


Figure 57 - One of the 8x8 sub-figure presented in numerical form

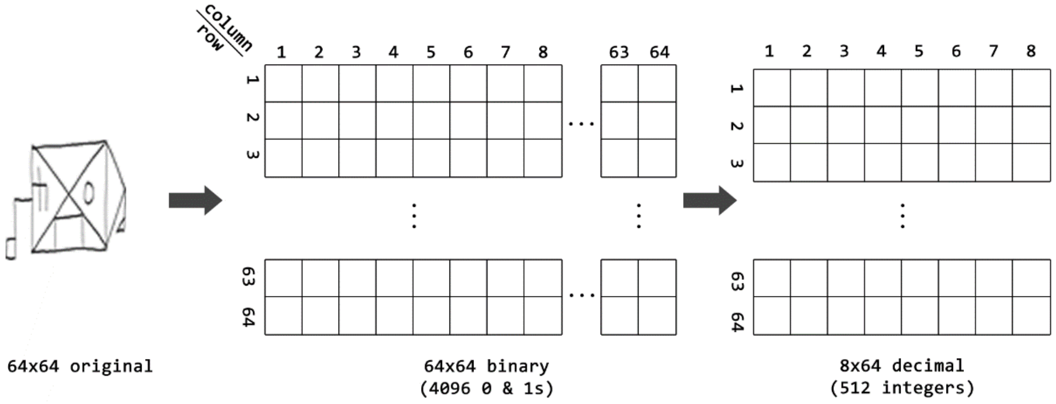


Figure 58 - Overall conversion process of an image file to numerical form

Those 512 features are extracted directly from the image file, combing with the 17 features that are extracted from the raw data, we can either train a deep learning model and a CGP model separately, or to combine those features to train a single CGP model.

8.2. More Accessible Option for Test Subject for Self-Assessing

Current method of data acquisition requires a digital tablet and a pairing stylus in order to capture detailed data of the pen, such as pen position, pen tilt and detailed pen pressure information. However, in this research, we did not find any relevance on how pen tilt will represent patient's movement difficulties, as well as the control of the pen pressure. Pen pressure is only used to identify whether the pen is on the tablet or not, detailed pressure value is not used.

A possible substitution of bulky tablet is smartphone. A mobile app can be developed to ask patient to perform certain copy and recall task by drawing on the phone. Current popular mobile operation systems, iOS and Android, are all support position tracking and touch detection according to their SDK documentation, so it's possible to extract all features needed for this research by using smartphone only. Data can either be uploaded to a central server for machine learning purposes, or use on-device machine learning development kit, such as iOS' Core ML. This will allow potential PD patients to conduct the drawing test on their own with easier way of conducting the test and possibly gather more data by gaining popularity among users, so that the CGP model can be trained with more data samples, the model itself can be more accurate as well.

Concerns arise with the interaction between people and the device. First, the digital tablet provides a larger area in physical space and drawing feeling, with a sheet of paper covered on the tablet, so that the data acquisition can be done while minimising the effect on the drawing performance that inducted by the drawing feeling. Most of the smartphones requires user to draw with finger, with limited stylus support. Apple's iPad Pro with Apple Pencil is considered to have the closest feeling on drawing a real paper, but such device compromises the accessibly for the patients due to its price range. Also, digital tablet provides feature that can track the stylus within certain distance from the tablet when the stylus is not on the tablet, so that off-paper action can also be tracked and used for further analyses, while iOS and Android SDK all indicate that it's not possible for current mobile operating system. In conclusion, by developing a mobile app, we can gather more data and allows user to conduct the test on themselves easier, but the data accuracy is slightly compromised due to the drawing experience and SDK limitation.

Sampling rate is another issue for consumer products. There is no fixed standard for screen touch sensor's sampling rate while current data extraction calculates the time of the drawing on the basis of the number of the data lines. Most of consumer products have a touch sampling rate of 60 Hz while few devices also support 120 Hz touch sampling rate (e.g. Apple iPhone XR [72]). A device with greater sampling rate will generate more data line as they collect data quicker in the same given time, while current extraction method will consider this drawing spent more time than on a device with less sampling rate. Although it is useful to collect user device's non-sensible data, such as the device model, with that to find the sampling rate for this device and calculate the drawing time accordingly, this will increase unnecessary workload for extra data collection of the devices. For research purposes, specialised tablet is still a preferred approach.

9. Conclusion

This thesis has presented a preliminary research on how machine learning can be used to diagnose and monitor Parkinson's disease, and how computer technology can assist researchers from different discipline. Overall, a satisfactory result is achieved with most of the hypothesis proven from the classification result produced by the SVM and the CGP. There are three parts I would like to conclude my thesis – project management, medical practicability and development on machine learning.

9.1. Project Management

Overall, this project undergoes smoothly without any external interference on the planned schedule. As a rewind, the project history as follows:

October 2018:	Project kicks off, preliminary reading, selected test set
November:	Simple program developed to pre-process all the data set, determines the features to be extracted from the raw data
January 2019:	Determined the fitness function for CGP, test strategy designed for CGP
March:	Validation approach determined, initial results from pair-wise classification
April:	Thesis structure determined
May:	Initial pair-wise results presented for GECCO conference
Mid-July:	Satisfactory results from multi-class classification by CGP, assessing performance between CGP and SVM
End-July:	Thesis final check, content, grammar, format, etc.
Early-August:	Thesis Submission

Most of works were focused on developing programs to automate CGP training process, as well as reading resources on possible solutions to optimise CGP for this specific project. As I wrote this thesis along with the project, I can catch every detail possible for this project, treating the thesis as a work log. Meanwhile, presenting paper for GECCO conference gives me opportunity to conclude my work at certain stage, gathering feedback from peer review, as well as attending conference that full of ideas on the research and application of machine learning, which affects my thesis content heavily.

9.2. Medical Practicability

As said before, this thesis only presents a preliminary research idea, it will be very difficult to push the current work as a usable package to the medical industry. However, current progress has shown the possibility to use such technique to tackle modern clinical problems. Over decades, researchers from worldwide found it difficult to find proper diagnostics method for Parkinson's disease. The idea behind this thesis not only is objective, fair, but also very simple and highly efficient. On patient side, patient can conduct such test on their own, and the application of machine learning also allow them to assess their result without an external examiner. On researchers' side, our work on feature extractions can help them understand different aspects that affect Parkinson's disease patients from different stages.

9.3. Machine Learning Development

Classification problem is always a popular topic in machine learning research. This research shows the potential of CGP that can handle abstract data which may have minimal correlations with each other. Whether in research area or commercial area, deep learning plays an important role, which makes people often ignores the power of genetic programming. It is the fact that deep learning can handle heavy AI tasks and more powerful than genetic programming, but there is a requirement to apply light-weight simple machine learning technique, such as this research project. It is very exciting to notice how many possibilities in this area to explore which are often ignored as people tend to consider them 'not a big deal', but has a large potential on practicability.

9.4. Research Programme

Overall, I am satisfied with the final research progress and result, considering the given time limit and small data size. Every aspect in this project, from background reading, project implementation, to paper and thesis writing, has opened my view on the application of ML in medical engineering using light-weighted algorithm package, which is similar to my undergraduate programme's final year project that finished one year ago. My previous project performed very simple classification on a simpler raw data from the 3-D cube shape. It is surprised to see that despite the similar nature between these two projects, my Master by research project has escalated a lot from my work which is only one year ago. From feature extraction, model validation, to software implementing and thesis writing. Despite I consider myself was struggling to make this project not as a copy of my previous one, the outcome of this research project showed that this is definitely a whole new challenge and the work involved is no similar to the work I have been done.

I also consider this one-year research programme an invaluable experience for me. From the aspect of personal development, I have to plan everything precisely and ensure that every estimation of workload is as accurate as possible. The transfer from taught programme to research programme also gives me an opportunity to get used to the life pace with minimum hand-holding, which seems unnecessary, but I am convinced that this would be very helpful when I enter the industries.

From professional prospective, research programme allows me more time to practice my programming skill, as I always aiming for a job as a software engineer, while the programming practice involved was very limited as an undergraduate in the Electronic Engineering department. This skill also helped me a lot in this project as most of the part are repetitive and boring parameter tuning and data export. I cannot image how this project can carried out in only one year without my own software package to extract features and automatically select the best model for me.

I have been fortunate enough to be given the opportunity to attend the ACM's Genetic and Evolutionary Computation Conference (GECCO) and have one paper published. Not only it gives me confidence on thesis writing, but also have a chance to understand what the trend is in the top tier of the evolutionary algorithm researching. I consider what I have experienced and learnt in this one single year, is way more than what I have got in the past 5 years combined.

Appendix A. Complete Pair-Wise Training Result

Pair 1. PD-NC/PD-MCI

Drawing mode: copy

CGP Parameters:

Parameter	Value
Node	75
Arity	3
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	3271,1886,3554,1880,3331,2217,2646,4642,1931,1452

Iteration 1			**Seed**	**3271**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	49500	100	80	100
fold 1	500	92.86	80	83.33
fold 2	84000	100	80	66.67
fold 3	139000	100	80	83.33
fold 4	151000	100	80	83.33
fold 5	65500	100	80	83.33
fold 6	133500	100	80	100
fold 7	104000	100	100	83.33
fold 8	113500	100	100	83.33
fold 9	149500	100	80	83.33
fold 10	163500	100	80	83.33
Average		99.35091	83.63636	84.84636
Standard Deviation		0.020526	0.077139	0.085708
Iteration 2			**Seed**	**1886**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	69500	100	100	83.33
fold 1	3000	92.86	80	83.33
fold 2	62000	100	100	83.33
fold 3	155000	100	80	66.67
fold 4	144000	100	100	83.33
fold 5	198000	100	80	83.33

fold 6	199000	100	80	83.33
fold 7	187000	100	80	66.67
fold 8	163000	100	100	83.33
fold 9	37500	100	100	83.33
fold 10	197000	100	80	83.33
Average		99.35091	89.09091	80.30091
Standard Deviation		0.020526	0.099586	0.064257
Iteration 3			**Seed**	**3554**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	190000	100	100	100
fold 1	33000	100	80	100
fold 2	53500	100	80	100
fold 3	172500	100	80	83.33
fold 4	95500	100	80	66.67
fold 5	105000	100	80	83.33
fold 6	185000	100	80	100
fold 7	108500	100	80	100
fold 8	180500	100	100	83.33
fold 9	188500	100	80	83.33
fold 10	162500	100	80	83.33
Average		100	83.63636	89.39273
Standard Deviation		0	0.077139	0.10714
Iteration 4			**Seed**	**1880**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	77500	100	80	83.33
fold 1	199500	100	80	83.33
fold 2	60000	100	80	83.33
fold 3	159500	100	80	83.33
fold 4	145000	100	80	66.67
fold 5	82000	100	80	83.33
fold 6	183000	100	80	100
fold 7	191000	100	80	83.33
fold 8	191500	100	80	83.33
fold 9	191500	100	80	83.33
fold 10	138000	100	100	83.33
Average		100	81.81818	83.33091
Standard Deviation		0	0.057496	0.07106
Iteration 5			**Seed**	**3331**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	190000	100	100	83.33
fold 1	36000	100	80	100

fold 2	82500	100	80	100
fold 3	91000	100	80	83.33
fold 4	198000	100	80	83.33
fold 5	166000	100	80	83.33
fold 6	199500	100	80	83.33
fold 7	159500	100	80	83.33
fold 8	79000	100	100	83.33
fold 9	123500	100	80	83.33
fold 10	184000	100	80	83.33
Average		100	83.63636	86.36091
Standard Deviation		0	0.077139	0.064295
Iteration 6			**Seed**	**2217**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	80500	100	80	100
fold 1	110500	100	80	66.67
fold 2	182500	100	80	66.67
fold 3	182000	100	80	83.33
fold 4	82000	100	80	100
fold 5	159000	100	80	83.33
fold 6	73500	100	100	83.33
fold 7	64000	100	80	83.33
fold 8	149000	100	100	83.33
fold 9	35000	100	100	83.33
fold 10	48500	100	80	83.33
Average		100	85.45455	83.33182
Standard Deviation		0	0.089072	0.100494
Iteration 7			**Seed**	**2646**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	182500	100	100	100
fold 1	133000	100	80	83.33
fold 2	164500	100	80	66.67
fold 3	2500	92.86	80	83.33
fold 4	6000	100	80	83.33
fold 5	200000	100	80	83.33
fold 6	174000	100	100	83.33
fold 7	91500	100	80	83.33
fold 8	143000	100	100	66.67
fold 9	126500	100	80	83.33
fold 10	174500	100	80	100
Average		99.35091	85.45455	83.33182
Standard Deviation		0.020526	0.089072	0.100494

Iteration 8			**Seed**	**4642**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	140500	100	100	83.33
fold 1	147000	100	80	66.67
fold 2	200000	100	100	83.33
fold 3	114500	100	80	66.67
fold 4	171500	100	80	100
fold 5	72000	100	80	83.33
fold 6	196000	100	80	83.33
fold 7	91500	100	100	83.3
fold 8	77500	100	100	83.33
fold 9	195000	100	80	83.33
fold 10	92500	100	80	100
Average		100	87.27273	83.32909
Standard Deviation		0	0.096209	0.100494
Iteration 9			**Seed**	**1931**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	78000	100	80	83.33
fold 1	19000	100	100	83.33
fold 2	131500	100	80	83.33
fold 3	157500	100	80	66.67
fold 4	4000	92.86	80	100
fold 5	188000	100	80	83.33
fold 6	176500	100	80	100
fold 7	56500	100	100	83.33
fold 8	148000	100	80	83.33
fold 9	199000	100	80	83.33
fold 10	151000	100	80	66.67
Average		99.35091	83.63636	83.33182
Standard Deviation		0.020526	0.077139	0.100494
Iteration 10			**Seed**	**1452**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	193000	100	100	100
fold 1	183000	100	80	66.67
fold 2	62500	100	80	83.33
fold 3	195000	100	80	83.33
fold 4	182500	100	80	83.33
fold 5	187000	100	80	66.67
fold 6	47000	100	100	83.33
fold 7	188500	100	80	66.67
fold 8	166500	100	80	100
fold 9	160500	100	80	83.33

fold 10	68000	100	100	83.33
Average		100	85.45455	81.81727
Standard Deviation		0	0.089072	0.111326

Drawing mode: recall

CGP Parameters:

Parameter	**Value**
Node	75
Arity	3
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	1396,989,3147,1940,4625,1093,1692,3992,2150,3755

Iteration 1			**Seed**	**1396**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	182500	100	80	83.33
fold 1	192000	100	80	66.67
fold 2	187000	100	80	83.33
fold 3	114000	100	100	83.33
fold 4	8000	100	100	100
fold 5	42000	100	100	83.33
fold 6	102500	100	100	83.33
fold 7	180500	100	80	83.33
fold 8	20500	100	80	83.33
fold 9	136500	100	80	66.67
fold 10	158000	100	80	83.33
Average		100	87.27273	81.81636
Standard Deviation		0	0.096209	0.085695
Iteration 2			**Seed**	**989**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	173000	100	80	100
fold 1	199000	100	80	66.67
fold 2	129000	100	80	100
fold 3	116500	100	100	83.33
fold 4	11500	100	100	83.33
fold 5	194500	100	80	83.33
fold 6	29500	100	80	83.33
fold 7	182000	100	100	83.33
fold 8	125000	100	80	83.33
fold 9	7000	100	80	83.33
fold 10	126000	100	80	100
Average		100	85.45455	86.36182

Standard Deviation		0	0.089072	0.095827
Iteration 3			**Seed**	**3147**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	186000	100	80	83.33
fold 1	193500	100	80	66.67
fold 2	165000	100	80	83.33
fold 3	189000	100	100	83.33
fold 4	137000	100	100	100
fold 5	169500	100	100	83.33
fold 6	146000	100	80	100
fold 7	114000	100	100	83.33
fold 8	188500	100	80	66.67
fold 9	124000	100	80	100
fold 10	144500	100	80	83.33
Average		100	87.27273	84.84727
Standard Deviation		0	0.096209	0.111333
Iteration 4			**Seed**	**1940**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	116000	100	100	83.33
fold 1	186000	100	80	66.67
fold 2	149000	100	100	83.33
fold 3	54000	100	80	100
fold 4	162500	100	100	83.33
fold 5	146000	100	80	83.33
fold 6	197500	100	100	83.33
fold 7	46500	100	100	83.33
fold 8	179500	100	80	83.33
fold 9	157000	100	80	83.33
fold 10	124500	100	100	83.33
Average		100	90.90909	83.33091
Standard Deviation		0	0.099586	0.07106
Iteration 5			**Seed**	**4625**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	110500	100	100	83.33
fold 1	2000	100	80	83.33
fold 2	189500	100	80	83.33
fold 3	187000	100	100	83.33
fold 4	198500	100	100	100
fold 5	167500	100	80	100
fold 6	139500	100	100	83.33
fold 7	33000	100	100	100
fold 8	154000	100	80	83.33

fold 9	169500	100	80	83.33
fold 10	177000	100	100	66.67
Average		100	90.90909	86.36182
Standard Deviation		0	0.099586	0.095827
Iteration 6			**Seed**	**1093**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	36000	100	100	83.33
fold 1	151000	100	80	83.33
fold 2	101500	100	80	83.33
fold 3	161000	100	100	83.33
fold 4	187000	100	80	100
fold 5	23500	100	100	100
fold 6	159000	100	80	83.33
fold 7	160000	100	80	100
fold 8	195500	100	80	66.67
fold 9	181500	100	80	66.67
fold 10	179000	100	80	83.33
Average		100	85.45455	84.84727
Standard Deviation		0	0.089072	0.111333
Iteration 7			**Seed**	**1692**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	156500	100	100	83.33
fold 1	128500	100	80	83.33
fold 2	33000	100	100	83.33
fold 3	53500	100	80	100
fold 4	52000	100	100	100
fold 5	102500	100	80	100
fold 6	160000	100	80	83.33
fold 7	177000	100	80	83.33
fold 8	58000	100	80	83.33
fold 9	172000	100	80	66.67
fold 10	187000	100	80	83.33
Average		100	85.45455	86.36182
Standard Deviation		0	0.089072	0.095827
Iteration 8			**Seed**	**3992**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	146000	100	100	83.33
fold 1	200000	100	80	66.67
fold 2	191500	100	100	83.33
fold 3	42500	100	100	83.33
fold 4	1765600	100	100	100
fold 5	87500	100	100	83.33

fold 6	165500	100	80	83.33
fold 7	196000	100	80	100
fold 8	141000	100	80	66.67
fold 9	196000	100	80	66.67
fold 10	177000	100	80	83.33
Average		100	89.09091	81.81727
Standard Deviation		0	0.099586	0.111326
Iteration 9			**Seed**	**2150**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	152500	100	80	100
fold 1	171500	100	80	66.67
fold 2	141500	100	80	83.33
fold 3	45500	100	100	83.33
fold 4	136500	100	100	100
fold 5	87500	100	80	83.33
fold 6	179500	100	80	83.33
fold 7	176500	100	80	100
fold 8	107500	100	80	83.33
fold 9	178500	100	80	66.67
fold 10	77000	100	80	83.33
Average		100	83.63636	84.84727
Standard Deviation		0	0.077139	0.111333
Iteration 10			**Seed**	**3755**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	39000	100	80	100
fold 1	112500	100	80	83.33
fold 2	193000	100	100	83.33
fold 3	104500	100	80	83.33
fold 4	193000	100	100	83.33
fold 5	144500	100	100	83.33
fold 6	171000	100	100	83.33
fold 7	196500	100	100	83.33
fold 8	24000	100	80	83.33
fold 9	194500	100	80	66.67
fold 10	178500	100	80	83.33
Average		100	89.09091	83.33091
Standard Deviation		0	0.099586	0.07106

Pair 2. PD-NC/PD-D

Drawing mode: copy

CGP Parameters:

Parameter	**Value**
Node	65
Arity	2
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	3727,928,2626,2076,4637,1079,4064,633,3961,4926

Iteration 1			**Seed**	**3727**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	165000	100	66.67	77.78
fold 1	52000	87.5	77.78	88.89
fold 2	174500	95.83	88.89	66.67
fold 3	200000	100	66.67	77.78
fold 4	158500	100	88.89	66.67
fold 5	97500	95.83	88.89	66.67
fold 6	10500	87.5	66.67	55.56
fold 7	200000	91.67	77.78	55.56
fold 8	54500	87.5	66.67	77.78
fold 9	110000	100	88.89	88.89
fold 10	27000	87.5	77.78	77.78
Average		93.93909	77.78	72.73
Standard Deviation		0.054363	0.094746	0.109714
Iteration 2			**Seed**	**928**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	133000	95.83	55.56	77.78
fold 1	84000	91.67	66.67	44.44
fold 2	163000	100	88.89	66.67
fold 3	56000	100	66.67	77.78
fold 4	200000	95.83	77.78	77.78
fold 5	39500	95.83	66.67	77.78
fold 6	200000	95.83	77.78	66.67
fold 7	196000	95.83	77.78	66.67

fold 8	200000	91.67	77.78	66.67
fold 9	150500	95.83	77.78	77.78
fold 10	157000	91.67	88.89	66.67
Average		95.45364	74.75	69.69909
Standard Deviation		0.02782	0.095817	0.095841
Iteration 3			**Seed**	**2626**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	85500	100	66.67	77.78
fold 1	21000	83.33	77.78	66.67
fold 2	42500	95.83	66.67	55.56
fold 3	22000	95.83	77.78	55.56
fold 4	88500	95.83	77.78	77.78
fold 5	71000	95.83	66.67	66.67
fold 6	121500	95.83	88.89	66.67
fold 7	198500	100	77.78	66.67
fold 8	199000	95.83	88.89	66.67
fold 9	160500	91.67	88.89	88.89
fold 10	120000	95.83	77.78	77.78
Average		95.07364	77.78	69.7
Standard Deviation		0.042857	0.082053	0.095817
Iteration 4			**Seed**	**2076**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	40500	91.67	55.56	77.78
fold 1	3000	79.19	77.78	77.78
fold 2	8500	91.67	66.67	77.78
fold 3	97500	100	66.67	44.44
fold 4	161500	100	55.56	77.78
fold 5	43500	100	66.67	66.67
fold 6	31000	91.67	77.78	77.78
fold 7	199000	95.83	77.78	66.67
fold 8	149000	95.83	77.78	55.56
fold 9	7000	83.33	77.78	77.78
fold 10	116000	95.83	77.78	55.56
Average		93.18364	70.71	68.68909
Standard Deviation		0.06468	0.085701	0.114288
Iteration 5			**Seed**	**4637**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	21000	87.5	77.78	55.56
fold 1	116000	95.83	66.67	77.78
fold 2	199500	100	66.67	66.67
fold 3	35500	100	44.44	55.56

fold 4	161500	100	55.56	55.56
fold 5	142500	95.83	77.78	55.56
fold 6	109500	95.83	88.89	77.78
fold 7	182000	100	77.78	77.78
fold 8	500	79.17	66.67	44.44
fold 9	27000	87.5	66.67	88.89
fold 10	126000	100	77.78	55.56
Average		94.69636	69.69909	64.64909
Standard Deviation		0.066684	0.116936	0.132474
Iteration 6			**Seed**	**1079**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	84500	91.67	66.67	66.67
fold 1	145000	100	66.67	77.78
fold 2	89500	100	66.67	77.78
fold 3	158500	100	77.78	66.67
fold 4	193500	95.83	66.67	77.78
fold 5	145500	100	66.67	66.67
fold 6	65000	95.83	77.78	55.56
fold 7	1500	79.17	77.78	66.67
fold 8	3000	79.17	55.56	33.33
fold 9	147000	100	88.89	77.78
fold 10	192500	91.67	88.89	66.67
Average		93.94	72.73	66.66909
Standard Deviation		0.07612	0.098959	0.125362
Iteration 7			**Seed**	**4064**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	34500	91.67	88.89	88.89
fold 1	174000	87.5	77.78	66.67
fold 2	184500	100	66.67	77.78
fold 3	200000	95.83	66.67	66.67
fold 4	147000	95.83	55.56	55.56
fold 5	168500	100	77.78	66.67
fold 6	4500	91.67	88.89	66.67
fold 7	71000	95.83	88.89	66.67
fold 8	200000	95.83	88.89	66.67
fold 9	103500	100	66.67	77.78
fold 10	3000	87.5	66.67	77.78
Average		94.69636	75.76	70.71
Standard Deviation		0.043841	0.114268	0.085701
Iteration 8			**Seed**	**633**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**

original	178000	95.83	77.78	77.78
fold 1	158000	100	66.67	66.67
fold 2	29000	91.67	77.78	55.56
fold 3	11500	95.83	77.78	66.67
fold 4	3000	91.67	66.67	55.56
fold 5	69000	100	66.67	55.56
fold 6	58000	91.67	66.67	55.56
fold 7	36500	91.67	88.89	77.78
fold 8	15500	87.5	44.44	55.56
fold 9	17500	87.5	44.4	77.78
fold 10	153500	100	88.89	77.78
Average		93.94	69.69455	65.66
Standard Deviation		0.04481	0.142932	0.099985
Iteration 9			**Seed**	**3961**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	140500	87.5	77.78	77.78
fold 1	5000	87.5	66.67	77.78
fold 2	194500	95.83	88.89	77.78
fold 3	166500	100	55.56	66.67
fold 4	160000	95.83	66.67	66.67
fold 5	20000	95.83	66.67	55.56
fold 6	90500	95.83	77.78	66.67
fold 7	156000	91.67	77.78	66.67
fold 8	11000	79.17	77.78	55.56
fold 9	55000	100	88.89	77.78
fold 10	76500	100	77.78	55.56
Average		93.56	74.75	67.68
Standard Deviation		0.062459	0.095817	0.08805
Iteration 10			**Seed**	**4926**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	114500	95.83	88.89	77.78
fold 1	65500	100	77.78	66.67
fold 2	152500	100	77.78	55.56
fold 3	13500	91.67	55.56	44.44
fold 4	3500	91.67	66.67	66.67
fold 5	157500	100	77.78	66.67
fold 6	167000	95.83	77.78	66.67
fold 7	171000	91.67	77.78	66.67
fold 8	58000	95.83	77.78	77.78
fold 9	48000	91.67	55.56	44.44
fold 10	136500	100	77.78	66.67
Average		95.83364	73.74	63.63818

Standard Deviation	0.035519	0.097923	0.106921

Drawing mode: recall

CGP Parameters:

Parameter	Value
Node	50
Arity	2
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	4772,2647,3023,2657,4320,4121,2314,3270,4572,4040

Iteration 1			**Seed**	**4772**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	180500	100	88.89	77.78
fold 1	35500	92	66.67	66.67
fold 2	159000	100	88.89	77.78
fold 3	14500	96	66.67	66.67
fold 4	183500	96	66.67	77.78
fold 5	11500	92	55.56	66.67
fold 6	13500	96	66.67	44.44
fold 7	47500	96	77.78	55.56
fold 8	199500	100	88.89	77.78
fold 9	67500	92	66.67	66.67
fold 10	155500	100	77.78	66.67
Average		96.36364	73.74	67.67909
Standard Deviation		0.031701	0.10878	0.100006
Iteration 2			**Seed**	**2647**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	192000	100	55.56	55.56
fold 1	1000	80	77.78	66.67
fold 2	121000	100	88.89	66.67
fold 3	133000	96	66.67	66.67
fold 4	136000	96	77.78	66.67
fold 5	196000	96	77.78	66.67
fold 6	199000	92	55.56	55.56
fold 7	175500	100	66.67	88.89
fold 8	78500	100	77.78	55.56
fold 9	82000	92	55.56	55.56

fold 10	151500	100	77.78	77.78
Average		95.63636	70.71	65.66
Standard Deviation		0.057725	0.10878	0.099985
Iteration 3			**Seed**	**3023**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	42500	96	77.78	77.78
fold 1	43500	92	77.78	66.67
fold 2	100000	92	88.89	66.67
fold 3	189500	100	66.67	55.56
fold 4	25000	92	55.56	55.56
fold 5	195500	100	77.78	77.78
fold 6	167500	96	77.78	55.56
fold 7	164000	100	77.78	66.67
fold 8	187500	100	77.78	66.67
fold 9	95000	100	77.78	77.78
fold 10	118000	96	66.67	77.78
Average		96.72727	74.75	67.68
Standard Deviation		0.033328	0.083287	0.08805
Iteration 4			**Seed**	**2657**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	164000	100	66.67	88.89
fold 1	200000	96	77.78	55.56
fold 2	200000	96	88.89	66.67
fold 3	170500	96	55.56	66.67
fold 4	148500	96	55.56	55.56
fold 5	20000	92	88.89	66.67
fold 6	190000	92	55.56	44.44
fold 7	185000	92	77.78	77.78
fold 8	27000	92	66.67	77.78
fold 9	197500	92	66.67	55.56
fold 10	16500	88	66.67	77.78
Average		93.81818	69.7	66.66909
Standard Deviation		0.031281	0.116916	0.125354
Iteration 5			**Seed**	**4320**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	3000	80	77.78	55.56
fold 1	79500	100	77.78	66.67
fold 2	12000	96	55.56	55.56
fold 3	108000	96	66.67	77.78
fold 4	26000	92	44.44	66.67
fold 5	195000	96	77.78	66.67

fold 6	82000	100	66.67	66.67
fold 7	139500	100	66.67	66.67
fold 8	155500	100	55.56	55.56
fold 9	19500	96	66.67	66.67
fold 10	93500	100	88.89	66.67
Average		96	67.67909	64.65
Standard Deviation		0.056569	0.120373	0.063878
Iteration 6			**Seed**	**4121**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	57500	96	77.78	77.78
fold 1	75000	96	77.78	66.67
fold 2	155000	100	77.78	55.56
fold 3	90500	96	66.67	44.44
fold 4	1000	84	44.44	66.67
fold 5	6500	84	55.56	77.78
fold 6	9500	92	66.67	44.44
fold 7	1500	80	77.78	55.56
fold 8	138500	96	66.67	66.67
fold 9	78500	100	66.67	77.78
fold 10	197500	92	77.78	88.89
Average		92.36364	68.68909	65.65818
Standard Deviation		0.064846	0.104007	0.137774
Iteration 7			**Seed**	**2314**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	124000	96	77.78	66.67
fold 1	34000	88	77.78	55.56
fold 2	138000	96	88.89	77.78
fold 3	182500	100	55.56	55.56
fold 4	163000	100	55.56	66.67
fold 5	182000	92	88.89	66.7
fold 6	7500	88	66.67	55.56
fold 7	150000	92	77.78	55.56
fold 8	165500	100	66.67	66.67
fold 9	91000	96	77.78	55.56
fold 10	177000	96	77.78	77.78
Average		94.90909	73.74	63.64273
Standard Deviation		0.042094	0.10878	0.083297
Iteration 8			**Seed**	**3270**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	123500	92	77.78	77.78
fold 1	184000	96	66.67	66.67

fold 2	155500	88	77.78	66.67
fold 3	174500	100	66.67	55.56
fold 4	110000	100	44.44	55.56
fold 5	46000	92	77.78	66.67
fold 6	86500	96	55.56	55.56
fold 7	22500	88	66.67	77.78
fold 8	100000	100	77.78	77.78
fold 9	46000	100	77.78	55.56
fold 10	199000	96	55.56	77.78
Average		95.27273	67.67909	66.67
Standard Deviation		0.044536	0.110659	0.094746
Iteration 9			**Seed**	**4572**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	106000	100	55.56	66.67
fold 1	113500	100	77.78	77.78
fold 2	50500	96	77.78	77.78
fold 3	48000	96	66.67	66.67
fold 4	175500	96	55.56	88.89
fold 5	126500	92	88.89	77.78
fold 6	61500	96	77.78	77.78
fold 7	169500	100	55.56	66.67
fold 8	94500	100	77.78	66.67
fold 9	178000	100	88.89	66.67
fold 10	139500	100	88.89	66.67
Average		97.81818	73.74	72.73
Standard Deviation		0.026222	0.127756	0.072832
Iteration 10			**Seed**	**4040**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	171500	100	77.78	66.67
fold 1	190000	96	66.67	55.56
fold 2	95000	96	88.89	66.67
fold 3	12000	100	77.78	55.56
fold 4	188500	96	77.78	66.67
fold 5	80000	100	77.78	55.56
fold 6	23500	96	66.67	55.56
fold 7	163000	96	66.67	77.78
fold 8	78000	96	77.78	77.78
fold 9	138500	96	55.56	33.33
fold 10	150500	100	77.78	88.89
Average		97.45455	73.74	63.63909
Standard Deviation		0.019242	0.085701	0.142855

Pair 3: PD-MCI/PD-D

Drawing mode: copy

CGP Parameters:

Parameter	**Value**
Node	70
Arity	2
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	1728,4579,4957,3633,19,1261,3448,1423,4056,2141

Iteration 1			**Seed**	**1728**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	155500	100	100	85.71
fold 1	143500	100	100	85.71
fold 2	198500	100	83.33	71.43
fold 3	11500	94.44	83.33	71.43
fold 4	158000	100	100	85.71
fold 5	199000	100	100	85.71
fold 6	64000	100	83.33	100
fold 7	188500	100	83.33	85.71
fold 8	151000	100	100	100
fold 9	176500	100	83.33	85.71
fold 10	74500	100	100	71.43
Average		99.49455	92.42273	84.41364
Standard Deviation		0.015984	0.083005	0.095426
Iteration 2			**Seed**	**4579**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	47000	94.44	83.33	85.71
fold 1	46000	100	100	85.71
fold 2	1000	94.44	83.33	71.43
fold 3	500	83.33	83.33	71.43
fold 4	10500	100	83.33	100
fold 5	93500	100	83.33	100
fold 6	163000	100	100	71.43
fold 7	65500	100	100	85.71

fold 8	2000	100	100	71.43
fold 9	192000	100	100	71.43
fold 10	17000	94.44	83.33	85.71
Average		96.96818	90.90727	81.81727
Standard Deviation		0.049499	0.083005	0.107082
Iteration 3			**Seed**	**4957**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	13500	94.44	83.33	85.71
fold 1	164500	100	100	85.71
fold 2	9000	94.44	83.33	85.71
fold 3	60500	100	83.33	71.43
fold 4	194500	100	83.33	100
fold 5	157500	100	83.33	100
fold 6	5500	94.44	83.33	85.71
fold 7	171500	100	83.33	85.71
fold 8	171000	100	66.67	71.43
fold 9	140500	100	100	85.71
fold 10	17500	94.44	83.33	85.71
Average		97.97818	84.84636	85.71182
Standard Deviation		0.026746	0.085708	0.086142
Iteration 4			**Seed**	**3633**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	127000	100	100	85.71
fold 1	74500	94.44	83.33	100
fold 2	160500	100	83.33	85.71
fold 3	161500	100	100	71.43
fold 4	200000	100	83.33	100
fold 5	149500	100	83.33	100
fold 6	129500	100	100	85.71
fold 7	198000	100	83.33	85.71
fold 8	188000	100	100	85.71
fold 9	160500	100	100	85.71
fold 10	70500	100	100	71.43
Average		99.49455	92.42273	87.01091
Standard Deviation		0.015984	0.083005	0.095434
Iteration 5			**Seed**	**19**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	80000	100	100	85.71
fold 1	12500	100	100	85.71
fold 2	120500	100	83.33	85.71
fold 3	194500	100	83.33	71.43

fold 4	196500	100	83.33	85.71
fold 5	177000	100	83.33	100
fold 6	184500	100	83.33	85.71
fold 7	122500	100	83.33	85.71
fold 8	1500	94.44	83.33	100
fold 9	125000	100	100	85.71
fold 10	125000	100	100	71.43
Average		99.49455	89.39182	85.71182
Standard Deviation		0.015984	0.08019	0.086142
Iteration 6			**Seed**	**1261**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	24500	94.44	83.33	71.43
fold 1	71500	100	100	85.71
fold 2	55500	100	83.33	85.71
fold 3	135500	100	100	71.43
fold 4	34500	100	83.33	100
fold 5	16500	100	100	71.43
fold 6	2000	88.89	83.33	71.43
fold 7	193500	100	83.33	71.43
fold 8	111000	100	100	85.71
fold 9	48000	100	100	71.43
fold 10	39000	100	100	85.71
Average		98.48455	92.42273	79.22
Standard Deviation		0.034256	0.083005	0.093633
Iteration 7			**Seed**	**3448**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	198000	100	100	100
fold 1	45500	100	100	100
fold 2	140000	100	83.33	71.43
fold 3	113000	100	83.33	85.71
fold 4	193000	100	83.33	100
fold 5	131000	100	83.33	71.43
fold 6	42000	100	100	85.71
fold 7	70000	100	83.33	85.71
fold 8	128000	100	100	71.43
fold 9	46000	100	100	71.43
fold 10	131000	100	100	85.71
Average		100	92.42273	84.41455
Standard Deviation		0	0.083005	0.11321
Iteration 8			**Seed**	**1423**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**

original	9500	94.44	83.33	85.71
fold 1	26000	100	100	71.43
fold 2	33500	100	66.67	85.71
fold 3	169500	100	100	100
fold 4	200000	100	83.33	100
fold 5	29000	100	83.33	100
fold 6	175000	100	83.33	85.71
fold 7	84500	100	83.33	85.71
fold 8	107000	100	100	100
fold 9	177500	100	83.33	85.71
fold 10	41500	100	83.33	71.43
Average		99.49455	86.36182	88.31
Standard Deviation		0.015984	0.095827	0.102261
Iteration 9			**Seed**	**4056**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	21000	100	100	85.71
fold 1	143500	100	100	100
fold 2	147500	100	83.33	85.71
fold 3	163500	100	83.33	85.71
fold 4	187000	100	83.33	85.71
fold 5	127000	100	83.33	100
fold 6	199000	100	100	85.71
fold 7	32000	100	100	71.43
fold 8	151000	100	83.33	71.43
fold 9	65000	100	83.33	57.14
fold 10	3000	94.44	83.33	85.71
Average		99.49455	89.39182	83.11455
Standard Deviation		0.015984	0.08019	0.119026
Iteration 10			**Seed**	**2141**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	129000	100	100	85.71
fold 1	39500	94.44	83.33	100
fold 2	143500	100	83.33	85.71
fold 3	24500	100	83.33	85.71
fold 4	185500	100	83.33	100
fold 5	180000	100	83.33	100
fold 6	66500	100	100	85.71
fold 7	88500	100	83.33	85.71
fold 8	88000	100	100	85.71
fold 9	156000	100	100	85.71
fold 10	166500	100	100	85.71
Average		99.49455	90.90727	89.60727

Standard Deviation	0.015984	0.083005	0.063642

Drawing mode: recall

CGP Parameters:

Parameter	**Value**
Node	75
Arity	5
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	4931,4384,4911,639,3321,844,3755,2227,4291,3277

Iteration 1			**Seed**	**4931**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	73000	100	83.33	71.43
fold 1	199500	100	100	71.43
fold 2	159500	100	83.33	100
fold 3	197000	100	83.33	85.71
fold 4	149000	100	83.33	85.71
fold 5	77000	100	100	85.71
fold 6	39000	94.44	83.33	100
fold 7	64000	100	83.33	85.71
fold 8	140000	100	83.33	85.71
fold 9	32000	100	83.33	57.14
fold 10	173000	100	83.33	71.43
Average		99.49455	86.36091	81.81636
Standard Deviation		0.015984	0.064295	0.123201
Iteration 2			**Seed**	**4384**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	198000	100	66.67	71.43
fold 1	128000	100	83.33	100
fold 2	56000	94.44	83.33	85.71
fold 3	119000	100	83.33	85.71
fold 4	134000	100	83.33	71.43
fold 5	95000	100	83.33	85.71
fold 6	78000	100	100	100
fold 7	163500	100	83.33	85.71
fold 8	19500	94.44	83.33	57.14
fold 9	7500	100	83.33	71.43

fold 10	12500	100	83.33	85.71
Average		98.98909	83.33091	81.81636
Standard Deviation		0.021445	0.07106	0.123201
Iteration 3			**Seed**	**4911**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	36000	100	100	71.43
fold 1	2000	88.89	83.33	85.71
fold 2	136000	100	83.33	85.71
fold 3	42500	94.44	83.33	85.71
fold 4	193000	100	83.33	71.43
fold 5	127500	100	83.33	85.71
fold 6	100000	100	83.33	71.43
fold 7	188500	100	66.67	71.43
fold 8	156500	100	66.67	85.71
fold 9	172500	100	83.33	71.43
fold 10	194500	100	83.33	71.43
Average		98.48455	81.81636	77.92091
Standard Deviation		0.034256	0.085695	0.071104
Iteration 4			**Seed**	**639**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	100500	100	83.33	71.43
fold 1	154000	100	100	71.43
fold 2	122500	100	100	85.71
fold 3	1000	83.33	83.33	85.71
fold 4	134500	100	83.33	85.71
fold 5	13500	94.44	83.33	71.43
fold 6	148500	100	100	71.43
fold 7	1000	94.44	83.33	71.43
fold 8	99000	100	83.33	71.43
fold 9	107000	100	66.67	85.71
fold 10	97000	100	83.33	71.43
Average		97.47364	86.36182	76.62273
Standard Deviation		0.049498	0.095827	0.068693
Iteration 5			**Seed**	**3321**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	97500	100	83.33	71.43
fold 1	65500	100	83.33	100
fold 2	191500	100	83.33	85.71
fold 3	12000	94.44	83.33	85.71
fold 4	189500	100	66.67	85.71
fold 5	126500	100	83.33	85.71

fold 6	171500	100	83.33	85.71
fold 7	133500	100	100	71.43
fold 8	118000	100	83.33	71.43
fold 9	197500	100	83.33	71.43
fold 10	49000	100	83.33	85.71
Average		99.49455	83.33091	81.81636
Standard Deviation		0.015984	0.07106	0.088066
Iteration 6			**Seed**	**844**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	197500	100	83.33	71.43
fold 1	98000	100	100	100
fold 2	158500	100	100	100
fold 3	193500	100	83.33	85.71
fold 4	21500	100	83.33	85.71
fold 5	185500	100	83.33	85.71
fold 6	98000	100	83.33	85.71
fold 7	8000	100	83.33	71.43
fold 8	90500	100	83.33	85.71
fold 9	157000	100	83.33	71.43
fold 10	52500	100	83.33	71.43
Average		100	86.36091	83.11545
Standard Deviation		0	0.064295	0.102249
Iteration 7			**Seed**	**3755**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	164000	100	83.33	85.71
fold 1	32500	94.44	83.33	100
fold 2	198000	100	100	85.71
fold 3	175000	100	83.33	71.43
fold 4	60500	100	100	71.43
fold 5	140500	100	83.33	85.71
fold 6	77500	100	83.33	85.71
fold 7	159000	100	83.33	57.14
fold 8	3000	94.44	83.33	57.14
fold 9	120500	100	100	71.43
fold 10	173500	100	83.33	71.43
Average		98.98909	87.87636	76.62182
Standard Deviation		0.021445	0.074242	0.125908
Iteration 8			**Seed**	**2227**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	195500	100	83.33	71.43
fold 1	116000	100	83.33	85.71

fold 2	12000	94.44	83.33	85.71
fold 3	132500	100	83.33	85.71
fold 4	84000	100	83.33	85.71
fold 5	1500	88.89	66.67	71.43
fold 6	104000	100	83.33	85.71
fold 7	187500	100	83.33	71.43
fold 8	200000	100	83.33	85.71
fold 9	87500	100	83.33	71.43
fold 10	158000	100	83.33	71.43
Average		98.48455	81.81545	79.21909
Standard Deviation		0.034256	0.047894	0.071104
Iteration 9			**Seed**	**4291**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	93500	100	83.33	71.43
fold 1	79000	94.44	83.33	100
fold 2	161000	100	83.33	100
fold 3	195500	100	83.33	85.71
fold 4	41000	100	83.33	85.71
fold 5	200000	100	100	85.71
fold 6	530000	100	83.33	100
fold 7	106500	100	83.33	71.43
fold 8	71500	100	83.33	85.71
fold 9	179500	100	83.33	57.14
fold 10	172500	100	83.33	71.43
Average		99.49455	84.84545	83.11545
Standard Deviation		0.015984	0.047923	0.133708
Iteration 10			**Seed**	**3277**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**
original	164000	100	83.33	71.43
fold 1	51000	100	83.33	85.71
fold 2	3000	94.44	83.33	71.43
fold 3	128000	100	83.33	71.43
fold 4	197000	100	83.33	71.43
fold 5	132000	100	100	71.43
fold 6	134000	100	100	71.43
fold 7	96500	100	83.33	85.71
fold 8	200000	100	83.33	85.71
fold 9	29000	100	83.33	71.43
fold 10	46000	100	83.33	57.14
Average		99.49455	86.36091	74.02545
Standard Deviation		0.015984	0.064295	0.082123

Appendix B. Complete Multi-Class Classification Result

Drawing mode: copy

CGP Parameters:

Parameter	**Value**
Node	80
Arity	2
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	1639,2552,2889,2998,3398,3614,3773,4201,4226,4928

Iteration 1				**Seed**	**1639**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**[5]
0	115500	89.66	60	70	45
1	5000	72.41	70	70	58.37
2	50000	89.66	70	70	59.75
3	15000	89.66	50	60	45.36
4	80000	89.66	60	70	35.08
5	500	82.76	60	60	45.95
6	500	75.86	60	60	47.16
7	135500	100	80	90	73.19
8	500	65.52	50	40	99.67
9	158000	96.55	80	60	66.58
10	22000	93.1	70	60	85.91
Iteration 2				**Seed**	**2552**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	6000	75.86	50	50	27.38
1	108500	75.86	60	60	45.84
2	174500	89.66	40	70	59.38
3	16500	82.76	70	50	76.2
4	7000	86.21	60	60	34.35
5	1000	79.31	50	60	40.08
6	13500	82.76	80	70	32.63

---

[5] null indicates value overflow, which exceeds the 64-bit limit of computer.

7	113500	96.55	50	70	null
8	1500	89.66	60	60	63.4
9	12000	89.66	70	70	54.37
10	1000	86.21	60	80	57.68
Iteration 3				**Seed**	**2889**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	4000	75.86	70	60	29.29
1	5000	79.31	50	60	60.17
2	5500	86.21	60	70	89.9
3	1000	75.86	40	50	32.24
4	1000	75.86	50	50	36.66
5	1500	86.21	60	60	64.2
6	8000	89.66	60	70	null
7	56000	96.55	70	70	65.25
8	162000	96.55	70	50	44.59
9	1000	86.21	70	60	66.47
10	6500	79.31	60	70	33.68
Iteration 4				**Seed**	**2998**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	28000	82.76	60	60	71.22
1	153500	93.1	70	70	null
2	4500	82.76	70	60	88.04
3	3500	82.76	50	70	81.86
4	8500	86.21	70	60	35.27
5	1000	79.31	50	50	47.11
6	99000	96.55	90	80	null
7	77500	93.1	70	70	45.02
8	4000	75.86	60	40	92.86
9	1000	72.41	50	60	92.77
10	1000	86.21	60	80	56.69
Iteration 5				**Seed**	**3398**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	1500	72.41	60	30	30.4
1	18000	89.66	60	70	65.97
2	500	75.86	60	50	43.1
3	4000	79.31	50	50	28.21
4	2000	86.21	60	60	28.65
5	152000	10	60	80	45.35
6	100500	100	70	60	84.7
7	171000	100	60	70	65.27
8	1500	75.86	70	70	29.89

9	5000	89.66	40	70	55.22
10	130000	100	40	80	64.11
Iteration 6				**Seed**	**3614**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	2500	79.31	70	50	28.81
1	22000	86.21	70	60	82.72
2	40500	86.21	70	70	57.68
3	1500	75.86	50	60	38.72
4	190500	96.55	80	70	44.14
5	48000	93.1	40	50	68.88
6	190000	100	80	70	62.12
7	196000	96.55	60	50	65.06
8	5000	79.31	50	60	52.67
9	142000	93.1	50	70	73.42
10	27500	96.55	80	70	41.97
Iteration 7				**Seed**	**3773**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	123000	82.76	60	30	47.22
1	84000	82.76	60	60	46.52
2	189000	100	40	50	85.22
3	190500	96.55	40	80	51.8
4	3500	86.21	80	70	80.04
5	3000	89.66	60	70	31.43
6	6500	86.21	70	60	36.49
7	166500	96.55	60	60	59.17
8	123500	96.55	60	60	77.24
10	70000	96.55	70	80	50.94
Iteration 8				**Seed**	**4201**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	500	65.52	40	50	42.84
1	3500	75.86	70	70	36.91
2	156500	93.1	40	50	62.87
3	138000	89.66	70	70	36.03
4	191500	100	70	70	57.98
5	2500	86.21	60	60	41
6	47000	89.66	80	70	44.02
7	31000	93.1	90	70	60.75
8	193000	96.55	50	50	51.38
9	98500	93.1	70	80	27.8
10	192500	100	50	60	46.81

Iteration 9				**Seed**	**4226**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	7000	86.21	60	70	null
1	115000	79.31	60	70	30.3
2	2500	75.86	60	60	50.42
3	69500	93.1	40	50	50.1
4	15000	89.66	70	70	37.06
4	153500	89.66	70	70	39.46
5	48000	96.55	70	60	64.9
6	44000	96.55	70	90	40.49
7	183000	100	70	60	90.42
8	1000	72.41	70	70	83.76
9	190500	100	70	60	70.66
10	182000	100	70	50	90.21
Iteration 10				**Seed**	**4928**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	199000	82.76	60	70	28.5
1	28500	86.21	60	70	70.55
2	53000	89.66	60	70	82.31
3	186500	89.66	70	70	32.08
4	82500	93.1	60	80	25.93
5	76500	93.1	60	80	25.94
6	174500	89.66	80	70	44.64
7	15500	82.76	60	60	48.32
8	92500	89.66	70	80	39.18
9	181500	93.1	40	70	38.82
10	13000	89.66	60	70	48.4

Drawing mode: recall

CGP Parameters:

Parameter	**Value**
Node	75
Arity	2
Mutation Rate	8%
Max Generation	200,000
Node Functions	add,sub,mul,div
Fitness Function	NWC
Random Seed	505,689,2136,2293,2793,3891,4134,4427,4674,4702

Iteration 1				**Seed**	**505**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	98500	90	70	50	77.51
1	193000	93.33	70	80	50.95
2	143500	90	70	70	51.25
3	100500	96.67	60	70	42.06
4	11000	90	50	70	64.68
5	192500	100	60	80	67.61
6	43500	100	70	70	54.18
7	15000	90	70	90	49.45
8	80000	96.67	90	70	91.23
9	32500	96.67	70	90	35.14
10	198000	100	70	60	59.94
Iteration 2				**Seed**	**689**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	185500	90	70	30	47.95
1	73500	96.67	80	70	51.77
2	500	73.33	20	50	64.48
3	68500	93.33	70	80	62.74
4	5000	83.33	60	80	55.15
5	44000	93.33	40	60	73.26
6	85500	96.67	50	40	75.57
7	17000	90	80	60	69.7
8	181500	96.67	80	70	51.29
9	17000	90	80	60	69.7
10	112500	100	90	80	49.26

Iteration 3				**Seed**	**2136**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	500	73.33	50	50	65.97
1	8000	80	60	60	56.08
2	500	70	50	60	63.72
3	12500	93.33	60	50	40.56
4	174500	100	40	60	64.16
5	178500	100	50	70	58.56
6	47500	100	60	70	75.25
7	169000	100	70	80	43.53
8	44000	96.67	90	70	null
9	161500	100	60	70	null
10	6000	96.67	50	70	54.11
Iteration 4				**Seed**	**2293**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	160500	93.33	40	70	58.66
1	2000	76.67	50	50	71.83
2	29500	90	50	70	58.88
3	148000	90	70	70	67.17
4	15000	96.67	70	60	64.63
5	108000	100	20	50	47.9
6	195500	100	70	70	94.27
7	19500	93.33	80	70	65.88
8	42500	93.33	80	80	81.79
9	21000	96.67	70	80	77.43
10	38500	96.67	90	90	45.23
Iteration 5				**Seed**	**2793**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	6500	76.67	70	70	32.22
1	12500	80	80	80	null
2	55000	93.33	50	70	72.29
3	19500	83.33	80	60	65.42
4	55000	93.33	40	50	29.93
5	99500	100	40	60	44.79
6	38500	100	70	50	32.68
7	2500	90	60	60	38.32
8	6000	80	60	60	67.56
9	63500	96.67	90	90	52.56
10	24000	100	70	70	null
Iteration 6				**Seed**	**3891**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**

0	33000	86.67	20	40	66.68
1	73500	83.33	70	70	43.17
2	52000	86.67	70	60	null
3	133500	86.67	40	70	40.65
4	11000	83.33	50	80	72.33
5	111000	100	60	60	56.2
6	13000	100	70	70	56.01
7	1000	80	60	50	55.37
8	3500	86.67	80	80	54.68
9	107500	96.67	70	80	48.85
10	63000	100	60	60	64.36
Iteration 7				**Seed**	**4134**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	139500	93.33	50	40	82.26
1	21500	86.67	60	80	41.58
2	48500	90	60	60	47.34
3	24000	93.33	70	80	42.35
4	88000	96.67	60	100	72.91
5	10500	90	50	70	49.85
6	83000	100	90	70	71.39
7	102500	100	100	70	56.27
8	75000	100	80	70	44.17
9	173000	100	70	70	null
10	37000	100	60	60	69.61
Iteration 8				**Seed**	**4427**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	500	70	40	50	70.16
1	65000	80	70	70	27.95
2	42500	86.67	80	70	39.24
3	1000	80	60	70	51.4
4	27000	90	70	70	null
5	95000	100	70	60	57.68
6	107500	96.67	60	50	42.95
7	79000	100	90	70	61.52
8	20500	93.33	70	70	58.64
9	68000	100	80	70	null
10	55500	100	60	70	45.83
Iteration 9				**Seed**	**4674**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	2000	73.33	60	50	69.36
1	189000	90	70	50	38.22

2	123000	93.33	60	60	null
3	32500	83.33	70	70	null
4	180500	86.67	40	70	49.81
5	73500	100	50	60	48.51
6	3500	90	70	60	60.39
7	61000	100	70	60	84.54
8	22500	90	80	60	62.62
9	157500	100	80	80	74.21
10	80000	100	70	70	39.6
Iteration 10				**Seed**	**4702**
Dataset	**Best Gen.**	**Training**	**Validation**	**Test**	**Confidence**
0	26500	86.67	70	60	75.83
1	1500	80	80	70	32.62
2	3500	83.33	60	60	41.04
3	5000	93.33	70	70	72.7
4	105000	100	70	80	87.29
5	6500	93.33	60	60	52.44
6	12500	93.33	60	60	54.37
7	1000	90	60	60	31.4
8	138000	93.33	70	70	30.36
9	21000	90	70	90	85.01
10	125000	100	80	70	null

Appendix C. Implementation of Softmax Function

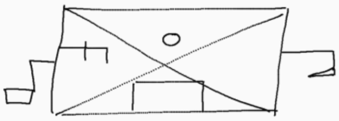
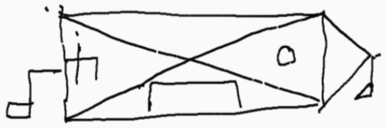
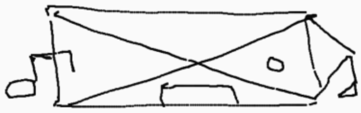
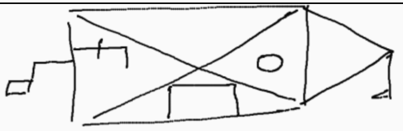
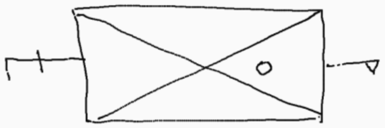
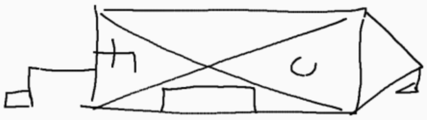
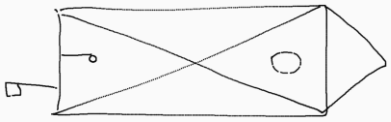
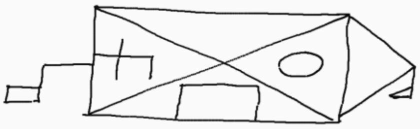
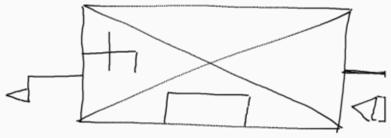
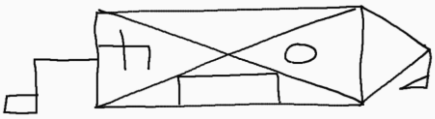
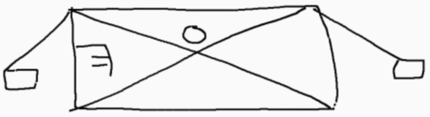
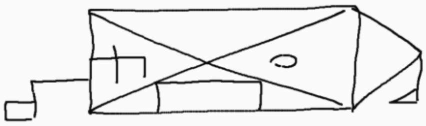
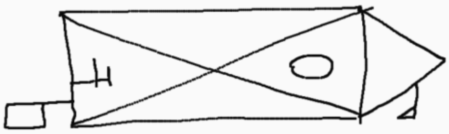
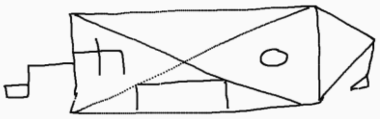
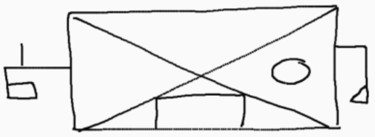
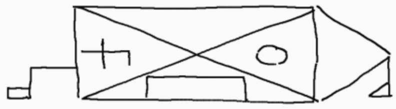
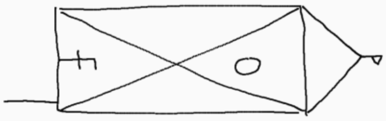
```
/* Implmentation of softmax function */
/* arg 1: double array which contains CGP outputs */
/* arg 2: number of CGP outputs */
double *softmax(double arr[], int arrLength)
{
    double logSum = 0;

    double *logArr = malloc(arrLength * sizeof(double));
    double *logAns = malloc(arrLength * sizeof(double));
    int i = 0;

    for (i = 0; i < arrLength; i++)
    {
        logArr[i] = exp(arr[i]);

        logSum += exp(arr[i]);
    }
    for (i = 0; i < arrLength; i++)
    {

        logAns[i] = logArr[i] / logSum;

    }

    free(logArr);
    return logAns;
}
```

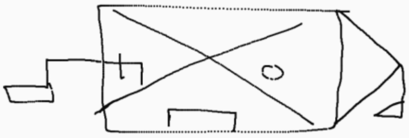
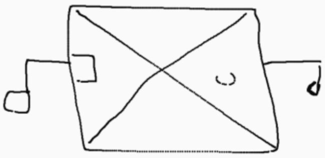
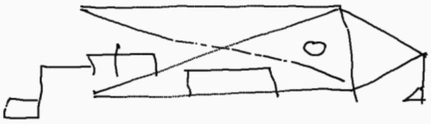
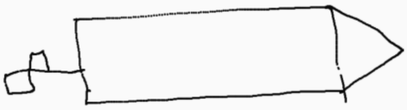
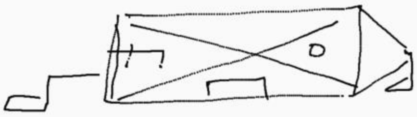
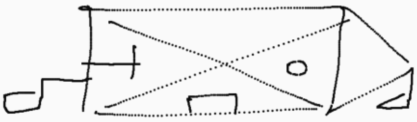
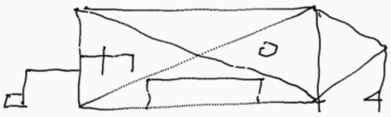
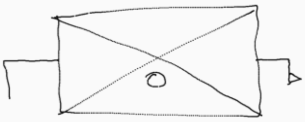
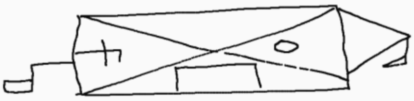
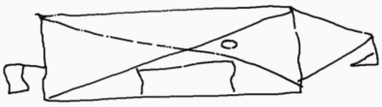
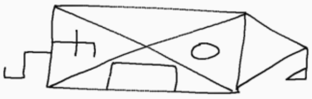
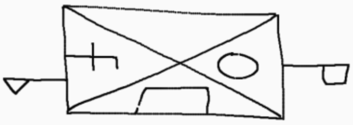
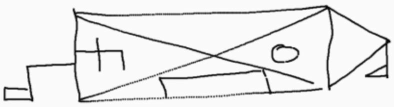
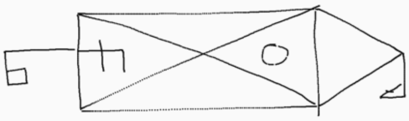
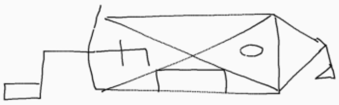
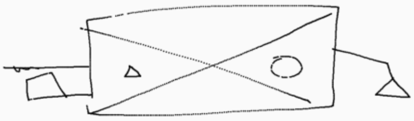
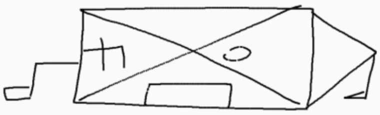
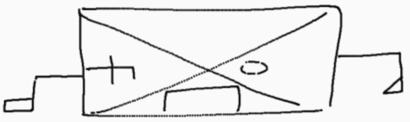
Appendix D. Implementation of NWC Fitness Function

```
double fourOutputFitnessFunction(struct parameters *params, struct
chromosome *chromo, struct dataSet *data)
{
    /* Routine check */
    if (getNumChromosomeInputs(chromo) !=
getNumDataSetInputs(data))
    {
        printf("Error: the number of chromosome inputs must match
the number of inputs specified in the dataSet.\n");
        printf("Terminating.\n");
        exit(0);
    }

    if (getNumChromosomeOutputs(chromo) !=
getNumDataSetOutputs(data))
    {
        printf("Error: the number of chromosome outputs must match
the number of outputs specified in the dataSet.\n");
        printf("Terminating.\n");
        exit(0);
    }

    int i;

    /* Counter to keep a record of error matches */
    double threshError = 0;

    for (i = 0; i < getNumDataSetSamples(data); i++)
    {
        /* Get the chromosome output */
        executeChromosome(chromo, getDataSetSampleInputs(data, i));
        double *chromoOutput = malloc(4 * sizeof(double));
        int j = 0;
        for (j = 0; j < 4; j++)
        {
            chromoOutput[j] = getChromosomeOutput(chromo, j);
        }

        /* Get acutal output */
        double *expectedOutput = getDataSetSampleOutputs(data, i);

        /* Check if the maximum output is in the index of the 
actual maximum output's */
        if (maxIndex(chromoOutput) != maxIndex(expectedOutput))
            threshError++;
    }

    return threshError / (getNumDataSetSamples(data));
}
```

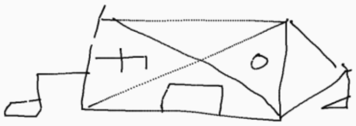
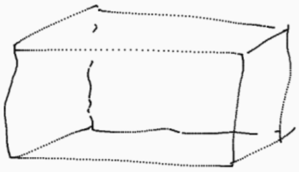
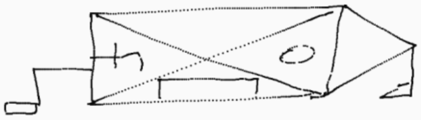
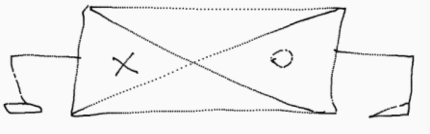
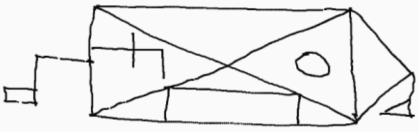
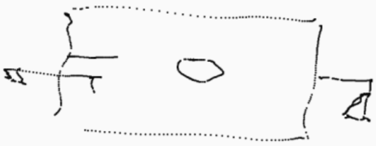
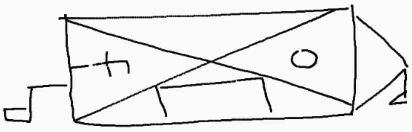
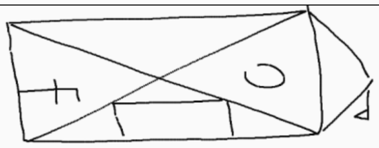
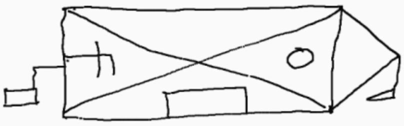
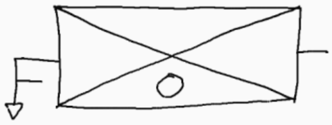
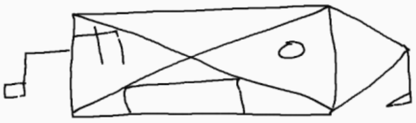
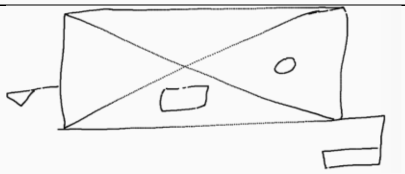
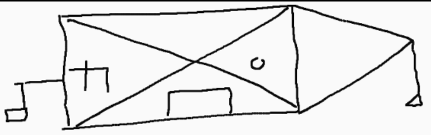
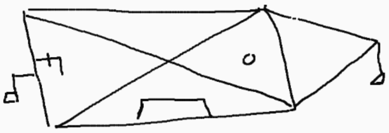
Appendix E: Benson Figure Images

Stage: PD-NC

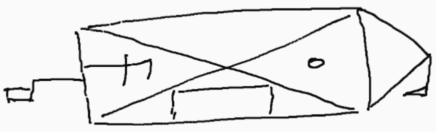
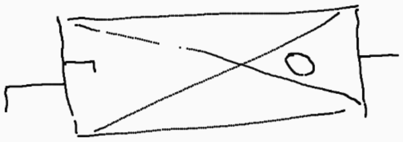
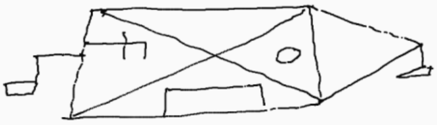
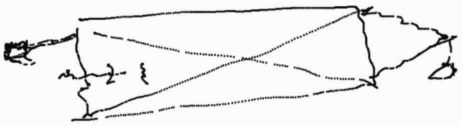
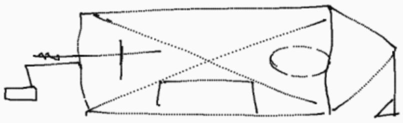
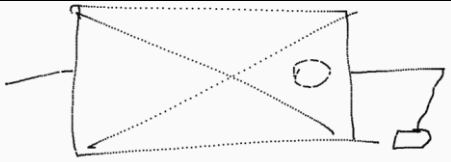
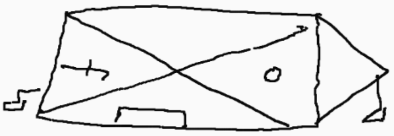
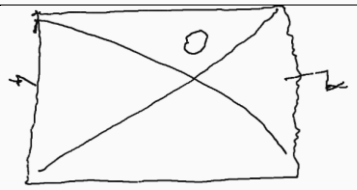
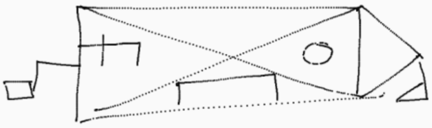
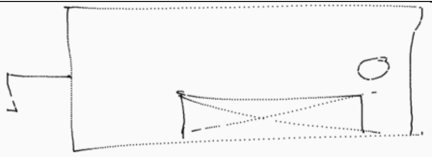
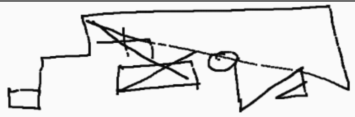
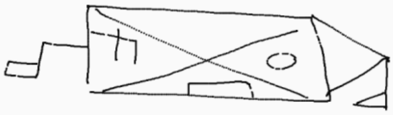
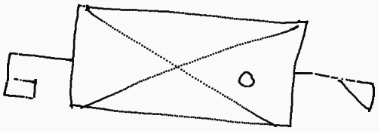
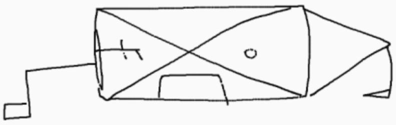
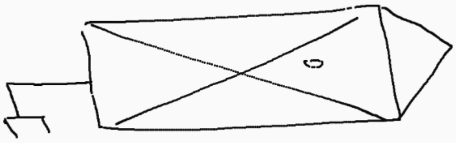
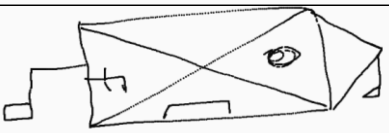
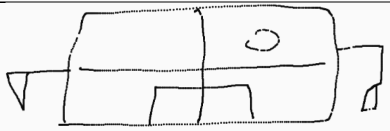
ID	Copy	Recall
1	Data corrupted	
5		
7		
8		
9		
12		
17		
18		
21		

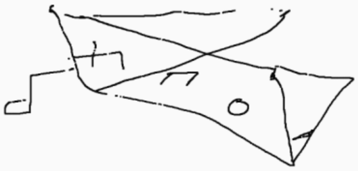
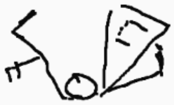
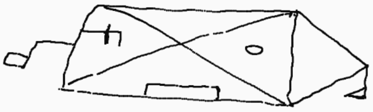
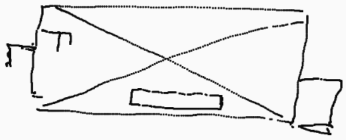
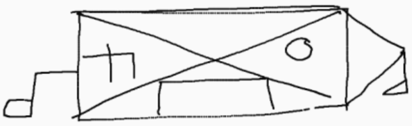
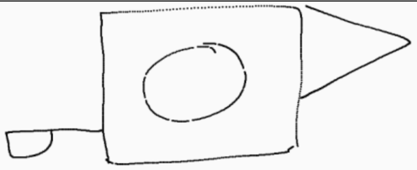
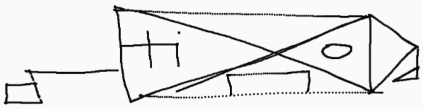
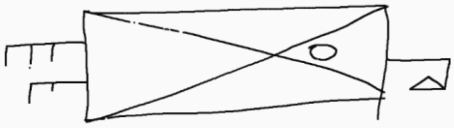
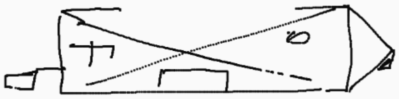
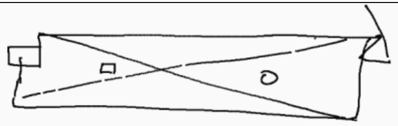
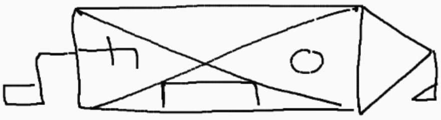
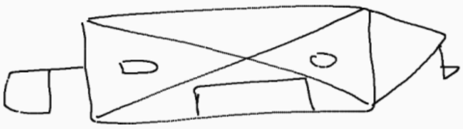
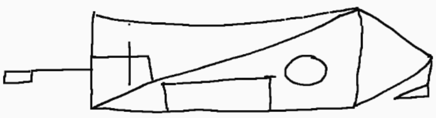
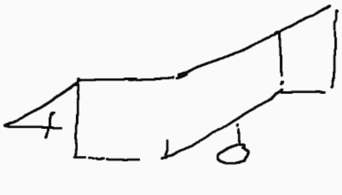
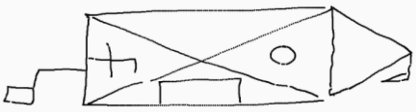
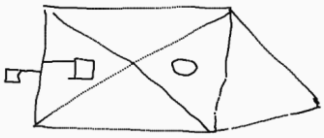
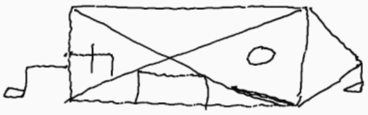
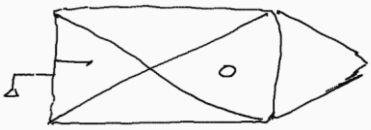
23		
24		
25		
30		
34		
35		
41		
46		
54		

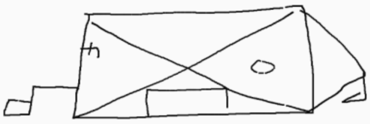
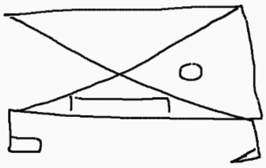
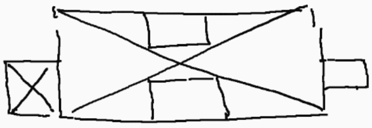
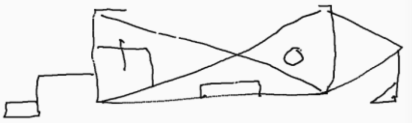
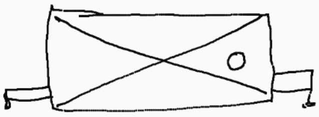
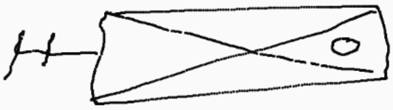
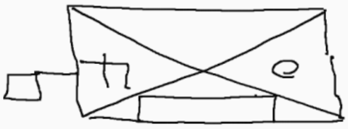
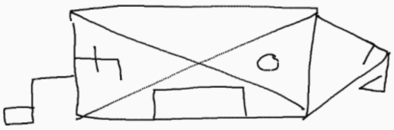
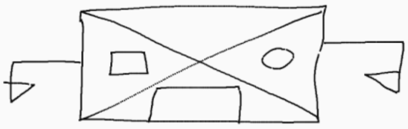
Stage: PD-MCI

ID	Copy	Recall
2		
6		
14		
26		
45		
56		
58		

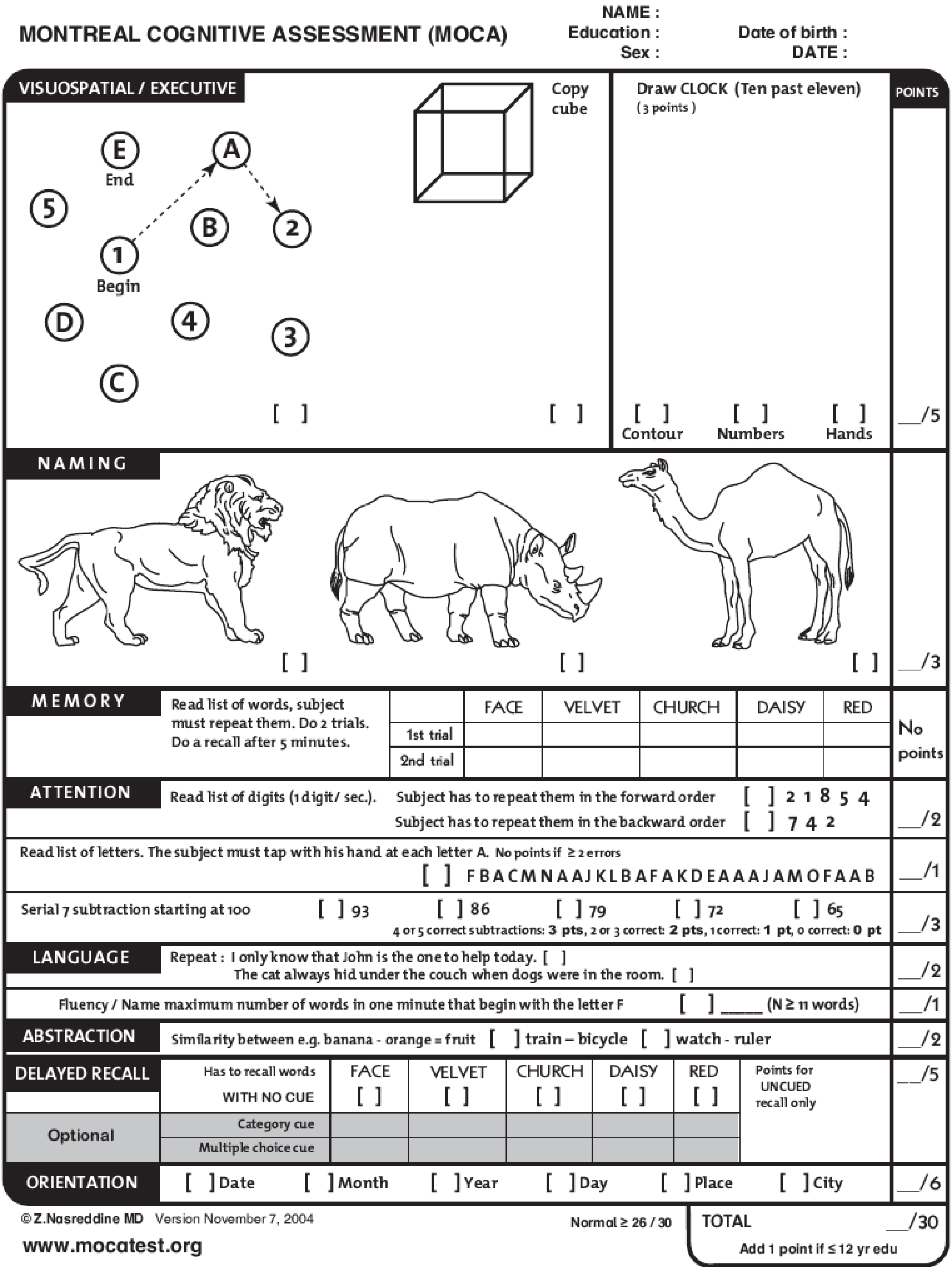
Stage: PD-D

ID	Copy	Recall
3		
10		
11		
13		
15		
16		Voided by test subject
19		
22		
27		

28		
29		
31		
36	Data Corrupted	Data Corrupted
37	Data Corrupted	Data Corrupted
38		
39		
40		
42		
43		
47		

49		
50		
51		
53		
55		
57		

Appendix F: MoCA Exam Sheet, taken from [73]


E
End
A
5
B
2
1
Begin
D
4
3
C

MONTREAL COGNITIVE ASSESSMENT (MOCA)

NAME :
Education :
Sex :
Date of birth :
DATE :

VISUOSPATIAL / EXECUTIVE

[ ]

Copy cube [ ]

Draw CLOCK (Ten past eleven) (3 points)

[ ] Contour [ ] Numbers [ ] Hands

POINTS __/5

NAMING

[ ] [ ] [ ] __/3

MEMORY Read list of words, subject must repeat them. Do 2 trials. Do a recall after 5 minutes.

	FACE	VELVET	CHURCH	DAISY	RED
1st trial					
2nd trial					

No points

ATTENTION Read list of digits (1 digit/ sec.).
Subject has to repeat them in the forward order [ ] 2 1 8 5 4
Subject has to repeat them in the backward order [ ] 7 4 2
__/2

Read list of letters. The subject must tap with his hand at each letter A. No points if ≥ 2 errors
[ ] F B A C M N A A J K L B A F A K D E A A A J A M O F A A B
__/1

Serial 7 subtraction starting at 100 [ ] 93 [ ] 86 [ ] 79 [ ] 72 [ ] 65
4 or 5 correct subtractions: **3 pts**, 2 or 3 correct: **2 pts**, 1 correct: **1 pt**, 0 correct: **0 pt**
__/3

LANGUAGE Repeat : I only know that John is the one to help today. [ ]
The cat always hid under the couch when dogs were in the room. [ ]
__/2

Fluency / Name maximum number of words in one minute that begin with the letter F [ ] _____ (N ≥ 11 words)
__/1

ABSTRACTION Similarity between e.g. banana - orange = fruit [ ] train – bicycle [ ] watch - ruler
__/2

DELAYED RECALL

		FACE	VELVET	CHURCH	DAISY	RED	
DELAYED RECALL	Has to recall words WITH NO CUE	[ ]	[ ]	[ ]	[ ]	[ ]	Points for UNCUED recall only
Optional	Category cue						
	Multiple choice cue						

__/5

ORIENTATION [ ] Date [ ] Month [ ] Year [ ] Day [ ] Place [ ] City
__/6

© Z.Nasreddine MD Version November 7, 2004
www.mocatest.org

Normal ≥ 26 / 30

TOTAL __/30
Add 1 point if ≤ 12 yr edu

Appendix G: Instruction to Set-Up Parameter Extractor and CGP Training Session

1. (With GIT) Execute

```
git remote add upstream https://github.com/OMGCA/Benson.git
```

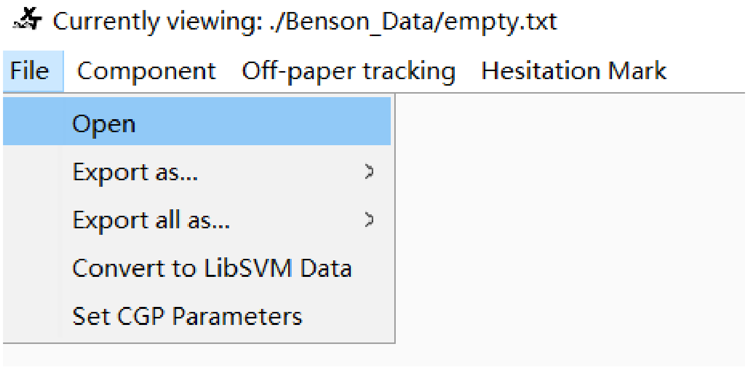
```
git pull upstream master
```

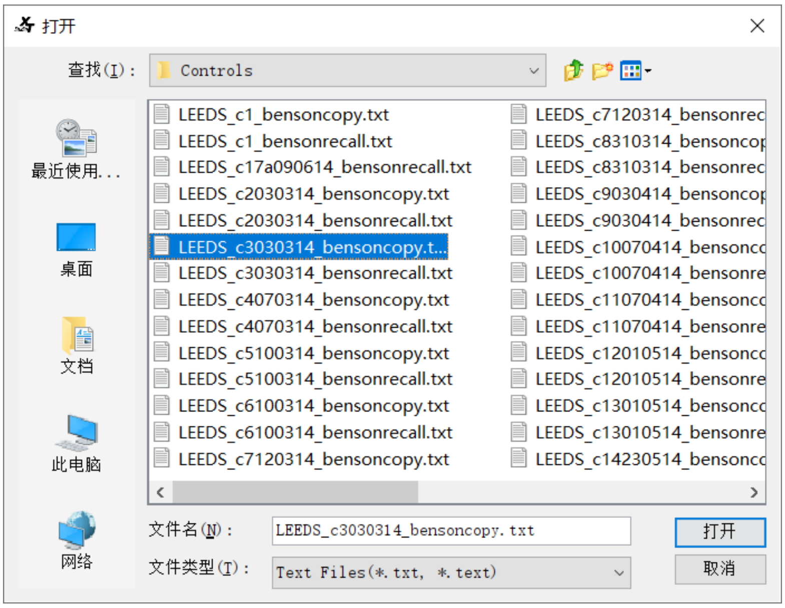
Or (Without GIT)

Download the package at https://github.com/OMGCA/Benson and extract content to an empty folder

2. Import the project into eclipse, compile and run (or export external JAR file)

3. Select File->Open to view a Benson figure from raw data



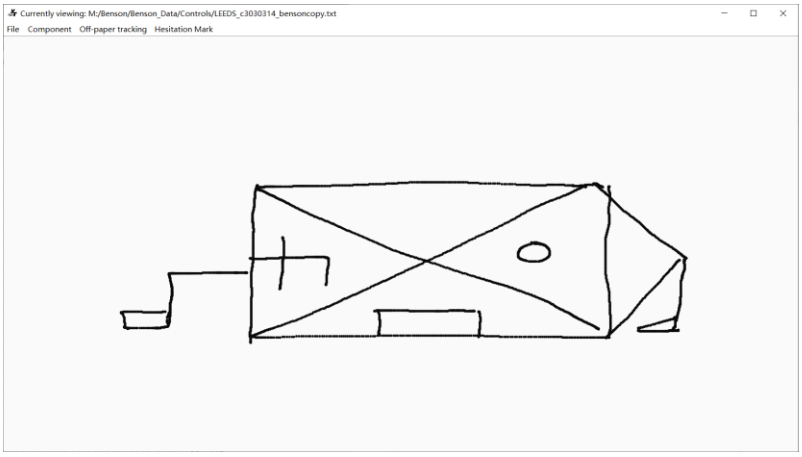


查找(I)：Controls

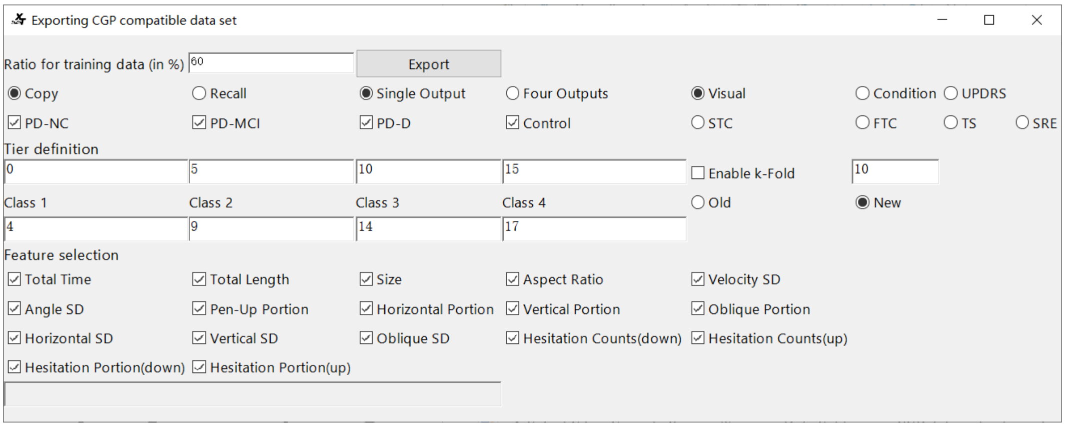
LEEDS_c1_bensoncopy.txt
LEEDS_c1_bensonrecall.txt
LEEDS_c17a090614_bensonrecall.txt
LEEDS_c2030314_bensoncopy.txt
LEEDS_c2030314_bensonrecall.txt
LEEDS_c3030314_bensoncopy.t...
LEEDS_c3030314_bensonrecall.txt
LEEDS_c4070314_bensoncopy.txt
LEEDS_c4070314_bensonrecall.txt
LEEDS_c5100314_bensoncopy.txt
LEEDS_c5100314_bensonrecall.txt
LEEDS_c6100314_bensoncopy.txt
LEEDS_c6100314_bensonrecall.txt
LEEDS_c7120314_bensoncopy.txt
LEEDS_c7120314_bensonrec
LEEDS_c8310314_bensoncop
LEEDS_c8310314_bensonrec
LEEDS_c9030414_bensoncop
LEEDS_c9030414_bensonrec
LEEDS_c10070414_bensonco
LEEDS_c10070414_bensonre
LEEDS_c11070414_bensonco
LEEDS_c11070414_bensonre
LEEDS_c12010514_bensonco
LEEDS_c12010514_bensonre
LEEDS_c13010514_bensonco
LEEDS_c13010514_bensonre
LEEDS_c14230514_bensonco

文件名(N)：LEEDS_c3030314_bensoncopy.txt

文件类型(T)：Text Files(*.txt, *.text)


Currently viewing: M:/Benson/Benson_Data/Controls/LEEDS_c3030314_bensoncopy.txt

4. Select File->Export all as...->Training Data Set to open CGP data output window, you can select features, configure output type, k-fold configurations etc.



Ratio for training data (in %) 60

● Copy ○ Recall ● Single Output ○ Four Outputs ● Visual ○ Condition ○ UPDRS

☑ PD-NC ☑ PD-MCI ☑ PD-D ☑ Control ○ STC ○ FTC ○ TS ○ SRE

Tier definition

0 5 10 15 ☐ Enable k-Fold 10

Class 1 Class 2 Class 3 Class 4 ○ Old ● New

4 9 14 17

Feature selection

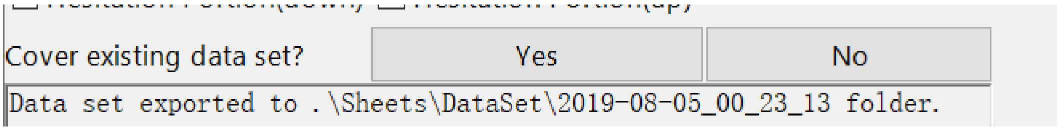
☑ Total Time ☑ Total Length ☑ Size ☑ Aspect Ratio ☑ Velocity SD

☑ Angle SD ☑ Pen-Up Portion ☑ Horizontal Portion ☑ Vertical Portion ☑ Oblique Portion

☑ Horizontal SD ☑ Vertical SD ☑ Oblique SD ☑ Hesitation Counts(down) ☑ Hesitation Counts(up)

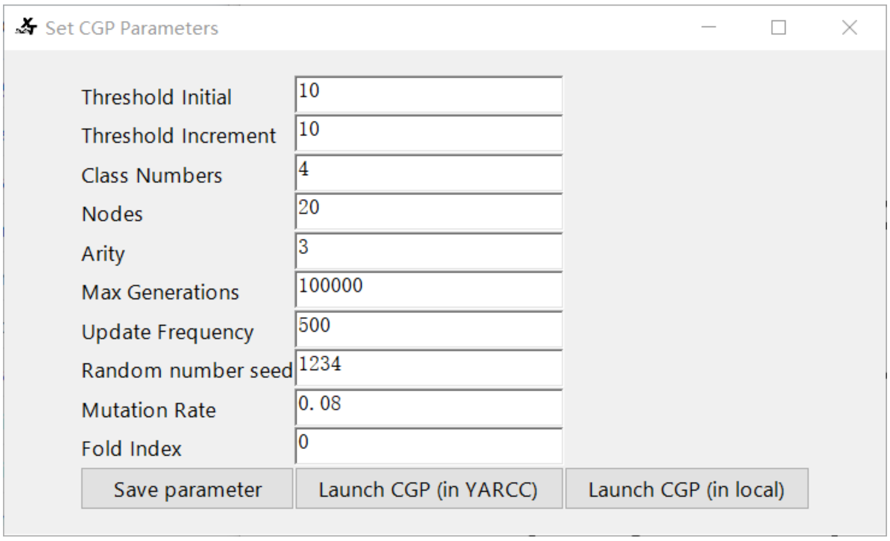
☑ Hesitation Portion(down) ☑ Hesitation Portion(up)

5. Select ‘Export’ to export data set. When finished, you can choose whether to copy the exported data set to the root folder of the CGP executable root folder.



```
Data set exported to .\Sheets\DataSet\2019-08-05_00_23_13 folder.
```

6. At the main menu, select File->Set CGP Parameters to tune the parameters for CGP



Threshold Initial	10
Threshold Increment	10
Class Numbers	4
Nodes	20
Arity	3
Max Generations	100000
Update Frequency	500
Random number seed	1234
Mutation Rate	0. 08
Fold Index	0

7. Click ‘Save parameter’ before launching the CGP by clicking ‘Launch CGP (in local)’. (YARCC requires PuTTY, Internet connection to University of York network and a valid University of York IT Account).

8. CGP training interface should prompt after step 7.

Glossary

AI	Artificial Intelligence
AD	Alzheimer's Disease
ANN	Artificial Neural Network
CDR	Clinical Dementia Rating
CDT	Clock Drawing Test
CGP	Cartesian Genetic Programming
DL	Deep Learning
EA	Evolutionary Algorithm
GP	Genetic Programming
HC	Healthy Control
ML	Machine Learning
MMSE	Mini-Mental State Examination
MOCA	Montreal Cognitive Assessment
NACC	National Alzheimer's Coordinating Center
NWC	Node Weighting Classifier
PCA	Principle Component Analysis
PD	Parkinson's Disease
PD-NC	Parkinson's Disease – Non-Cognitive
PD-MCI	Parkinson's Disease – Mild Cognitive Impairment
PD-D	Parkinson's Disease - Dementia
ROCF	Rey-Osterrieth Complex Figure
SD	Standard Deviation
STC	Simple Threshold Classifier
SVM	Support Vector Machine
UPDRS	Unified Parkinson's Disease Rating Scale

References

[1] World Health Organization, “WHO Neurological Disorders: Public Health Challenges,” [Online]. Available: https://www.who.int/mental_health/neurology/neurodiso/en/. [Accessed 15 4 2019].

[2] T. R. Collins, “Neurologic Diseases Found to Be the Largest Cause of Disability Worldwide,” *Neurology Today,* pp. 32-35, 16 11 2017.

[3] GBD 2015 Disease Injury Incidence Prevalence Collaborators , “Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015,” *Lancet,* pp. 1545-1602, 8 10 2016.

[4] GBD 2015 Disease Injury Incidence Prevalence Collaborators, “Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015,” *Lancet,* pp. 1459-1544, 8 10 2016.

[5] S. Sveinbjornsdottir, “The clinical symptoms of Parkinson's disease,” *Journal of Neurochemistry,* pp. 318-324, 11 7 2016.

[6] The New York Times, “For the Phinney Family, a Dream and a Challenge,” 6 11 2014. [Online]. Available: https://www.nytimes.com/2008/03/26/sports/othersports/26cycling.html. [Accessed 15 4 2019].

[7] R. L. Brey, “Muhammad Ali's Message: Keep Moving Forward,” Neurology Now, 27 9 2011. [Online]. Available: https://web.archive.org/web/20110927022505/http://www.aan.com/elibrary/neurologynow/?event=home.showArticle&id=ovid.com%3A%2Fbib%2Fovftdb%2F01222928-200602020-00003. [Accessed 15 4 2019].

[8] “Parkinson's Disease Information Page,” National Instituite of Neurological Disorders and Stroke, 27 3 2019. [Online]. Available: https://www.ninds.nih.gov/Disorders/All-Disorders/Parkinsons-Disease-Information-Page. [Accessed 29 4 2019].

[9] L. V. Kalia and A. E. Lang, “Parkinson's disease,” *The Lancet,* vol. 386, no. 9996, pp. 896-912, 2015.

[10] A. Samii, J. G. Nutt and B. R. Ransom, “Parkinson's disease,” *The Lancet,* vol. 363, no. 9423, pp. 1783-1793, 2004.

[11] J. Jankovic, “Parkinson’s disease: clinical features and diagnosis,” *Neurosurg Psychiatry,* pp. 368-376, 2008.

[12] D. W. Dickson, H. Braak, J. E. Duda, C. Dyuckaerts, T. Gasser, G. M. Halliday, J. Hardy, J. B. Leverenz, K. D. Tredici, Z. K. Wszolek and I. Litvan, “Neuropathological assessment of Parkinson’s disease: refining the diagnositc criteria,” *The Lancet Neurology,* vol. 8, no. 12, pp. 1150-1157, 2009.

[13] M. Emre, D. Aarsland, R. Brown, D. J. Burn, C. Duyckaerts, Y. Mizuno, G. A. Broe, J. Cummings, D. W. Dickson, S. Gauthier, J. Goldman, C. Goetz, A. Korezyn, A. Lees, R. Levy, I. Litvan, I. McKeith, W. Olanow, W. Poewe, N. Quinn, C. Sampaio, E. Tolosa and B. Dubois, “Clinical Diagnostic Criteria for Dementia Associated with Parkinson's Disease,” *Movement Disorder,* vol. 22, no. 12, pp. 1689-1707, 2007.

[14] L. M. Stallings, Motor Skills: Development and Learning, Boston: WCB/McGraw-Hill, 1973.

[15] M. Gazzaniga, R. Ivry and G. Mangun, Cognitive Neuroscience: The biology of the mind, New York: W.W. Norton & Company, 2009.

[16] L. Sherwood, Huamn Physiology: From Cells to Systems, Cengage Learning, 2015.

[17] M. Eysenck, Attention and Arousal : Cognition and Performance, Berlin: Springer Berlin Heidelberg, 2012.

[18] K. Nuytemans, J. Theuns, M. Curts and C. V. Broeckhoven, “Genetic etiology of Parkinson disease associated with mutations in the SNCA, PARK2, PINK1, PARK7, and LRRK2 genes: a mutation update,” *Human Mutation,* vol. 31, no. 7, pp. 763-780, 2010.

[19] H. Förstl and A. Kurz, “Clinical features of Alzheimer's disease,” *European Archives of Psychiatry and Clinical Neuroscience,* vol. 249, no. 6, pp. 288-290, 1999.

[20] S. Salimi, M. Irish, D. Foxe, J. R. Hodges, O. Piquet and J. R. Burrell, “Can visuospatial measures improve the diagnosis of Alzheimer's disease?,” *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring,* pp. 66-74.

[21] T. Iachini, A. Iavarone, V. P. Senese, F. Ruotolo and G. Ruggiero, “Visuospatial memory in healthy elderly, AD and MCI: a review,” *Current Aging Science,* vol. 2, no. 1, pp. 43-59, 2009.

[22] M. D. Lezak, D. B. Howieson, E. D. Bigler and D. Tranel, Neuropsychological Assessment, New York: Oxford University Press, 2012.

[23] T. Xia, “Automated Analysis of Clinical Drawing Tests using Evolutionary Algorithms,” 2018.

[24] D. J. Brooks, “Imaging Approaches to Parkinson Disease,” *The Journal of Nuclear Medicine,* pp. 596-609, 4 2010.

[25] S. T. Schwarz, M. Afzal, P. S. Morgan, N. Bajaj, P. A. Gowland and D. P. Auer, “The ‘Swallow Tail’ Appearance of the Healthy Nigrosome – A New Accurate Test of Parkinson's Disease: A Case-Control and Retrospective Cross-Sectional MRI Study at 3T,” *PLoS ONE,* 7 4 2014.

[26] A. Dehsarvi and S. Smith, “Classification of Resting-State fMRI using Evolutionary Algorithms: Towards an Olfactory Dysfunction Biomarker for Parkinson’s Disease,” in *GECCO '18 Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Kyoto, Japan, 2018.

[27] The New York Times, “Trump Passed a Cognitive Exam. What Does That Really Mean?,” 19 1 2018. [Online]. Available: https://www.nytimes.com/2018/01/19/health/trump-cognitive-screening-dementia.html. [Accessed 13 5 2015].

[28] The Atlantic, “The President Can Draw a Clock,” 16 1 2018. [Online]. Available: https://www.theatlantic.com/health/archive/2018/01/the-president-can-draw-a-clock/550610/. [Accessed 13 5 2019].

[29] The Guardian, “Donald Trump wrongly claims his father was born in Germany – again,” 3 4 2019. [Online]. Available: https://www.theguardian.com/us-news/2019/apr/03/trump-claims-father-born-germany-false-fred-trump. [Accessed 13 5 2019].

[30] Yahoo! News, “Trump's 'pattern of cognitive decline' alarms psychiatrists,” 6 4 2019. [Online]. Available: https://news.yahoo.com/trumps-pattern-of-cognitive-decline-alarms-psychiatrists-110000099.html?guccounter=1&guce_referrer=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3J

nLw&guce_referrer_sig=AQAAAHMJ0n6mwIJ7i5_MAJi_MY32u_J9pyLKMuOKkGKMilpmYcC0ycPCBLvK6hZgMWKfBtpCymPjtrgW. [Accessed 13 5 2019].

[31] Esquire, “President Trump Just Repeatedly Demanded to Know 'the Oranges of the Investigation',” 2 4 2019. [Online]. Available: https://www.esquire.com/news-politics/a27021746/trump-oranges-of-the-investigation-origin-father-germany/. [Accessed 13 5 2019].

[32] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach (3rd ed.), New Jersey: Prentice Hall, 2009.

[33] A. M. Turing, “Computing Machinery and Intelligence,” *Mind,* vol. LIX, no. 236, pp. 433-460, 1950.

[34] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[35] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[36] A. Ben-Hur, D. Horn, H. Siegelmann and V. N. Vapnik, “Support vector clustering,” *Journal of Machine Learning Research,* vol. 2, pp. 125-137, 2001.

[37] “1.4. Support Vector Machines - scikit-learn 0.20.2 documentation,” 8 11 2017. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html. [Accessed 21 5 2019].

[38] T. Hastie, R. Tibshirani and J. Friedman, in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009, p. 134.

[39] F. Karimi, S. Sultana, A. S. Babakan and S. Suthaharan, “An enhanced support vector machine model for urban expansion prediction,” *Computers, Environment and Urban Systems,* vol. 75, pp. 61-75, 2019.

[40] D. Jurafsky and J. H. Martin, “Chapter 18: Semantic Role Labeling,” 23 9 2018. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/18.pdf. [Accessed 21 5 2019].

[41] S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin and D. Jurafsky, “Shallow semantic parsing using support vector machines,” in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL*, Boston, 2004.

[42] D. Decoste and B. Schölkopf, “Training Invariant Support Vector Machines,” *Machine Learning,* vol. 46, no. 1-3, pp. 161-190, 2002.

[43] F. Bre, J. M. Gimenez and V. D. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks,” *Energy and Buildings,* p. 158, 2017.

[44] Y. Bengio, “Learning Deep Architectures for AI,” *Foundations and Trends® in Machine Learning,* vol. 2, no. 1, pp. 1-127, 2009.

[45] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, “Generative Adversarial Networks,” in *NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, Montreal, Canada, 2014.

[46] P. A. Vikhar, “Evolutionary Algorithms: A Critical Review and its Future Prospects,” in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication*, 2016.

[47] H. Lo, “Genetic Programming,” 26 6 2014. [Online]. Available: https://www.slideserve.com/alida/teaching-assistant. [Accessed 21 5 2019].

[48] A. J. Turner, “CGP-Library Documentation,” 2016. [Online]. Available: http://www.cgplibrary.co.uk/files2/About-txt.html. [Accessed 1 5 2019].

[49] C. Gao, S. Smith, M. Lones, S. Jamieson, J. Alty, J. Cosgrove, P. Zhang, J. Liu, Y. Chen, J. Du, S. Cui, H. Zhou and S. Chen, “Objective assessment of bradykinesia in Parkinson’s disease using evolutionary algorithms: clinical validation,” *Translational Neurodegeneration,* vol. 7, 2018.

[50] I. Litvan, J. G. Goldman, A. I. Troster, B. A. Schmand, D. Weintraub, R. C. Petersen, B. Mollenhauer, C. H. Adler, K. Marder, C. H. Williams-Gray, D. Aarsland, J. Kulisevsky, M. C. Rodriguez-Oroz, D. J. Burn, R. A. Barker and M. Emre, “Diagnostic Criteria for Mild Cognitive Impairment in Parkinson's Disease: Movement Disorder Society Task Force Guidelines,” *Movement Disorders,* vol. 27, no. 3, pp. 349-356, 2012.

[51] J. Cosgrove, “Investigating reach and grasp in Parkinson's disease cognitive impairment,” 2016.

[52] T. Xia, J. Cosgrove, J. Alty, S. Jamieson and S. Smith, “Application of classification for figure copying test in Parkinson's disease diagnosis by using cartesian genetic programming,” in *GECCO '19 Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Prague, Czech Republic, 2019.

[53] E. F. Hammer, The clinical application of projective drawings, Oxford, England: Charles C Thomas, 1958.

[54] E. M. Mpangane, “Draw a Person Test,” University of Limpopo, 2015.

[55] “Draw-a-Person test,” 9 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Draw-a-Person_test. [Accessed 5 6 2019].

[56] K. L. Possin, V. R. Laluz, O. Z. Alcantar, B. L. Miller and J. H. Kramer, “Distinct neuroanatomical substrates and cognitive mechanisms of figure copy performance in Alzheimer's disease and behavioral variant frontotemporal dementia,” *Neuropsychologia,* vol. 49, no. 1, pp. 43-48, 2011.

[57] M.-S. Shin, S.-Y. Park, S.-R. Park, S.-H. Seol and J.-S. Kwon, “Clinical and empirical applications of the Rey-Osterrieth Complex Figure Test,” *Nature Protocols,* vol. 1, pp. 892-899, 2006.

[58] “NACC Uniform Data Set (UDS) - FTLD Module Instructions for Neuropsychological Questionnaires (Forms C2F - C6F) and Tests Reported on Form C1F,” National Alzheimer's Coordinating Center, 2012.

[59] K. I. Shulman, “Clock-drawing: is it the ideal cognitive screening test?,” *International Journal of Geriatric Psychiatry,* vol. 15, no. 6, pp. 548-561, 2000.

[60] K. Watanabe, T. Ogino, K. Nakano, J. Hattori, Y. Kado, S. Sanada and Y. Ohtsuka, “The Rey-Osterrieth Complex Figure as a measure of executive function in childhood,” *Brain and Development,* vol. 27, no. 8, pp. 564-569, 2005.

[61] G. Mcanulty, F. H. Duffy, S. Kosta, N. Weisenfeld, S. Warfield, S. C. Butler, J. H. Bernstein, D. Zurakowski and H. Als, “School Age Effects of the Newborn Individualized Developmental Care and Assessment Program for Medically Low-Risk Preterm Infants: Preliminary FindingsSchool Age Effects of the Newborn Individualized Developmental Care and Assessment Program for Medically,” *Journal of clinical neonatology,* vol. 1, pp. 184-194, 2012.

[62] S. Z. Nasreddine, N. A. Phillips, V. Bédirian, S. Charbonneau, V. Whitehead, I. Collin, J. L. Cummings and H. Chertkow, “The Montreal Cognitive Assessment, MoCA: A Brief Screening Tool For Mild Cognitive Impairment,” *Journal of the American Geriatrics Society,* vol. 53, no. 4, pp. 695-699, 2005.

[63] Z. Nasreddine, “Montreal Cognitive Assessment (MoCA),” 1 2017. [Online]. Available: https://www.mocatest.org/wp-content/uploads/2017/01/MoCA-New-Test-8.1-2017-04.pdf. [Accessed 18 6 2019].

[64] C. Ramaker, J. Marinus, A. M. Stiggelbout and B. J. van Hilten, “Systematic evaluation of rating scales for impairment and disability in Parkinson's disease,” *Movement Disorder,* pp. 867-876, 2002.

[65] “Qualtative Analysis,” ANSTAT Consulting, [Online]. Available: https://amstatisticalconsulting.com/services-2/machine-learning-2/. [Accessed 29 7 2019].

[66] J. M. Bland and D. G. Altman, “Statistics notes: Measurement error,” *BMJ,* vol. 312, p. 1654, 1996.

[67] H. Ling, L. A. Massey, A. J. Lees, P. Brown and B. L. Day, “Hypokinesia without decrement distinguishes progressive supranuclear palsy from Parkinson's disease,” *Brain,* vol. 135, no. 4, pp. 1141-1153, 2012.

[68] Oxford University Press, “Definition of overfitting in English,” [Online]. Available: https://en.oxforddictionaries.com/definition/overfitting. [Accessed 14 5 2019].

[69] E. Alpaydin, Introduction to Machine Learning, London: the MIT Press, 2010.

[70] Google, “Classification: True vs. False and Positive vs. Negative,” 5 3 2019. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative. [Accessed 4 8 2019].

[71] Google, “Classification: ROC Curve and AUC,” 5 3 2019. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. [Accessed 4 8 2019].

[72] C. U. I. Technology, “Apple iPhone XR Specifications,” 20 9 2018. [Online]. Available: https://cuit.columbia.edu/sites/default/files/content/XR_Specs.pdf. [Accessed 26 6 2019].

[73] “Montreal Cognitive Assessment (MoCA),” [Online]. Available: https://www.parkinsons.va.gov/resources/MOCA-Test-English.pdf. [Accessed 30 7 2019].