Heriot-Watt University

Masters Thesis

Parkinson's Disease Diagnosis using Deep Neuroevolution

Author:
Lucía Parga Basanta

Supervisor:
Michael Lones

A thesis submitted in fulfilment of the requirements for the degree of MSc. Artificial IntelligenceArtificial Intelligence

in the

School of Mathematical and Computer Sciences

May 2020

Declaration of Authorship

I, Lucía Parga Basanta, declare that this thesis titled, 'Parkinson's Disease Diagnosis using Deep Neuroevolution' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Lucía Parga Basanta

Date: 04/2020

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Abstract

Parkinson Disease is the second most common neurodegenerative disease and currently without a cure, although early diagnosis is beneficial. Handwriting tasks can be used to evaluate symptoms in early stages of PD, using convolutional neural networks to analyse the image data. This project uses multi-objective evolutionary algorithms to optimize the deep network's architecture in order provide simpler and more accurate model to support Parkinson diagnosis.

Acknowledgements

I would like to thank my supervisors Michael Lones and Marta Vallejo for their help and advice towards the completion of this work.

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **iv**

Contents **v**

List of Figures **vii**

List of Tables **viii**

Abbreviations **ix**

1 Introduction **1**

2 Literature Review **2**
- 2.1 Parkinson Disease . . . 2
- 2.2 Medical diagnosis and Deep Learning . . . 3
- 2.3 Deep Neural Networks . . . 4
  - 2.3.1 Convolutional Neural Networks . . . 5
- 2.4 Deep Neuroevolution . . . 7
- 2.5 Evolutionary Algorithms . . . 7
- 2.6 Multi-Objective Evolutionary Algorithms . . . 8
  - 2.6.1 NSGA-II . . . 10

3 Methodology **12**
- 3.1 Overview . . . 12
- 3.2 Data preparation . . . 12
- 3.3 The deep learning network . . . 14
- 3.4 MOEA . . . 15
- 3.5 Neuroevolutionary Tools . . . 15
- 3.6 Training, Testing and Evaluation . . . 17

4 Requirements Analysis **19**

4.1 Project Goals 19
4.2 Deliverables 19
4.3 Priorities 20
4.4 Project Risk Assesment 20

5 Discussion of Professional, Legal, Ethical, and Social Issues 23
5.1 Legal Issues 23
5.2 Ethical Issues 23
5.3 Social Issues 23

6 Project Plan 25
6.1 Ghant Chart 25

A Extension of NSGA-II Algorithm 27

Bibliography 28

List of Figures

2.1 A common form of CNN architecture: convolutional layers activation function ReLu, pooling layers, and fully connected layers with ReLu and without. Figure reproduced from O'Shea and Nash [2015] 6
2.2 Pareto front of non-dominated solutions for a two-dimensional objective space, between cost and energy efficiency [Luke, 2013]. 9
2.3 Multi-objective lexicographic with two objectives from Luke [2013]. In this algorithm, the objectives are ordered by the user preference in importance and solutions are compared based on the objective value of those preferences. 10
2.4 An abstract version of the NSGA-II from Luke [2013]. Algorithm 103 refers to Figure A.1 in Appendix A. 11

3.1 (a) Illustration of the spiral pentagon template, (b) a pentagon drawing gathered from a patient, (c) a cube drawing without zero-pressure information, and (d) a cube drawing with zero-pressure information [Alissa et al., 2020]. 13
3.2 Examples of the augmentation process. Original images of a pentagon and a cube drawing (left) and examples of the generation of augmented images (right) 13
3.3 CNN architecture from Alissa et al. [2020], consisting of two convolutional layers with 32 filters followed by two convolution layers with 64 filters and another two convolutional layers with 128 filters, three max-pooling layers of size (2×2), six dropout layers, three dense layers and one flattened layer 15

6.1 Gantt Chart 26

A.1 Non-dominated sorting lexicographic tournament selection with sparsity from Luke [2013] 27

List of Tables

3.1 Discussion of different frameworks available for deep learning 14
3.2 Discussion of different frameworks available for neuroevolution in Python 16

4.1 MoSCoW prioritization for tasks of the project 20
4.2 Details of impact, likelihood and mitigation plan for each risk factor. . . . 22

Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
ConvNet	**C**onvolutional **N**etwork or **C**onvolutional **N**eural **N**etwork
DL	**D**eep **L**earning
EA	**E**volutionary **A**lgorithm
GG	**G**enetic lgorithm
MOEA	**M**ulti–**O**bejctive **E**volutionary **A**lgorithm
NN	**N**eural **N**etwork
PD	**P**arkinson **D**isease
RL	**R**enforcement **L**earning
RNN	**R**ecurrent **N**eural **N**etworks

Dedicated to my family...

Chapter 1

Introduction

Parkinson's Disease (PD) is a progressive neurodegenerative disorder that causes the loss of motor coordination and movement control of muscles [Turner and Desmurget, 2010]. It is the second most common neurodegenerative disorder [T.Pringsheim et al., 2014], affecting around 1% of the population at the age 65 and around 5% at age 85 [Reeve et al., 2014]. Current PD diagnosis requires the presence of bradykinesia (slowness of movements) in addition to as muscle rigidity, tremors or postural instability - symptoms that can easily lead to misdiagnosis as they generally overlap with other medical conditions or diseases. For an accurate diagnosis of PD, continuous clinical evaluation over time is needed, around three years to reach 90% of accuracy [Hughes et al., 2002]. Successful detection at early stages allows for appropriate treatment and information could be given to patients. The rate of diagnosis of PD is only 10-25% [Hughes et al., 1992]. There is a need for quicker and more accurate tests that can provide results to support clinicians diagnosis.

In early stages of PD, handwriting kinematics are affected, due to symptoms such as micrographia and dysgraphia. Drawing tasks have been found to be a successful way to evaluate and detect this symptoms and, as a result, a sensitive way for providing support for PD diagnosis Pereira et al. [2016].

Handwriting kinematics data can be used as an input for image classification using computational models. The previous work of Alissa et al. [2020] successfully created a model that uses CNNs to deep analyze the images develop an tool to supports PD diagnosis. Inspired by this, this research aims to find the simplest architecture that can successfully perform as image classifier for the same dataset. To optimize this architecture, deep neuroevolutionary tools will be used.

Chapter 2

Literature Review

2.1 Parkinson Disease

Parkinson disease (PD) is a progressive neurodegenerative disorder with both motor and non-motor symptoms, that causes neuronal loss. As found in Dexter and Jenner [2013], PD is characterised by the death of dopaminergic neurons in the *substantia nigra pars compacta* (SNpc) and the presence of Lewy bodies in various parts of the brain. This is correlated with the loss of motor coordination and movement control of muscles, affecting the control of voluntary muscular actions [Turner and Desmurget, 2010].

PD is the second most common neurodegenerative disorder after Alzheimer's disease [T.Pringsheim et al., 2014]; it affets around 1% of the population at the age 65, which increases to around 5% at age 85, being the age group in highest risk for developing PD [Reeve et al., 2014]. At present, the cause of PD is not clear, but studies show that a combination of genetic susceptibility and environmental factors are involved Lesage and Brice [2009].

Currently, no cure for Parkinson is known, and nothing can prevent the progressive neural degeneration and death. However, successful detection at early stages of PD is important in order to provide patients with appropriate treatment and information. Current diagnosis of PD requires the presence of bradykinesia (slowness of movements), in addition to muscle rigidity, tremors, or postural instability; symptoms that can overlap with other conditions or diseases, leading to misdiagnosis (with a rate of 10-25% [Hughes et al., 1992]). Clinical evaluation over time needs to be performed for accurate diagnosis, taking up to 2.9 years to reach 90% accuracy in detecting PD [Hughes et al., 2002]. The need for quicker and non-invasive trustful tests that provide objective results to support clinicians diagnosis is required.

In early stages of PD, aspects of handwriting kinematics are affected and as neurodegeneration progresses, disorders as micrographia (reductions in writing size) and dysgrafia (decreased ability to write in general) appear. This symptoms can be evaluated by undertaking tests with specific writing tasks with the aim to support the diagnosis of PD. Drawing tasks have been studied and found to be sensitive for detecting early signs of PD [Pereira et al., 2016]. The image data used in this study corresponds to the ones used in the work of Vallejo et al. [2016] and Alissa et al. [2020]. The analysis of these drawings provide significant data from the force, speed, time, tightness and uniformity generated by the patient for a period of time. This data can be used as input for image classification computational models to support diagnosis of PD. In previous work of Alissa et al. [2020], convolutional neural networks (CNN or ConvNet) are used in order to deep analyse the information in the patients' drawings and create a model to successfully support the diagnosis of PD. This research aims to evolve the architecture of the CNN used for image classification on the same dataset in order to obtain higher accuracy on the results while using a model with lower complex architecture (number of layers). As these two are conflicting objectives, a Multi-Objective Evolutionary Algorithm (MOEA) will be used for evolving the CNN (Section 2.6)

2.2 Medical diagnosis and Deep Learning

The recent progress in Deep Learning (DL) has opened up new applications in medical diagnosis. For image classification purposes, CNN are chosen due to their successful results when working image data as inputs. For example, the work of Anthimopoulos et al. [2016] uses a 5 layer CNN to classify interstitial lung diseases patterns, using CT scan datasets to train and evaluate the CNN. Another example of DL in medical diagnosis is the work of T.Pringsheim et al. [2014], who developed a model for Alzheimer's disease diagnosis using a deep supervised adaptable 3D CNN over structural brain MRI scans.

Pereira et al. [2016] were the first in combining DL techniques with the dataset of the drawing tasks images as input of the networks for computer-aided PD diagnosis. A CNN composed of 5 convolution layers, 5 pooling layers and 2 normalization layers was trained to support an input of 264 scanned images of 256 x 256 pixels gathered from 35 individuals for meanders and spiral drawing tasks. A higher accuracy of image classification was obtained for spiral images (89.55%) than meander figures (79.62%). Building on this, Alissa et al. [2020] used a simpler CNN (lower complexity) achieving same or higher performance when applying it over the drawing images. The highest accuracy reached was 93.5%, when trained with images of a pentagon drawing task and augmentation techniques (see Section 3.2). The results on Alissa et al. [2020] inspired this research which aims to find the simplest architecture that can have a good performance as image classifier over the same dataset.

2.3 Deep Neural Networks

In the early days of Artificial intelligence (AI), algorithms were used to solve problems that humans found too complex and challenging, but that computers found straightforward to compute, usually calculating answers based on mathematical rules. However over time, AI shifted focusing on problems that humans typically consider easy, such as recognizing spoken words or faces in images. These are problems humans intuitively and automatically solve in their daily life, but which prove challenging for scientists to simplify and explain. This means developing the required precise algorithms is extremely difficult [Goodfellow et al., 2016].

Previously, Artificial Neural Networks (ANN) were used to solve these problems. However, they presented serious limitations when working with complex, non-linear data [Nielsen, 2015]. For example, when using a very simple ANN model for image classification, the classifier algorithm might be successful when using very basic and simple images, for example, very structured images like numbers framed similarly in all images or faces also framed similarly. However, when working with complex images that show local spatial pixel dependencies throughout, the performance of the networks will be poor. ANNs are not able to take advantage of these local relationships. For instance, when the same number can be located in different places of the image, an ANN is not able to distinguish it. Another limitation of using ANNs is that as all images are represented with a grid-like topology, the images need to be transformed into a vector to be able to be processed by the ANNs.

Deep learning networks tackle these limitations and they are specifically successful at processing grid-like data while also being efficient at extracting features that can show spatial dependencies over the grid data. Deep learning allows the computer to understand the world in terms of a hierarchy of concepts [Goodfellow et al., 2016]. For example, a deep learning network trained with images of different fruits will successfully detect which is the fruit and where in the image appears, by extracting basic features (such as black and white pixels or RGB colour) to more complex features of the image (such as shapes and textures, as it implies more pixels to infer). This is done by the specific architecture of these deep neural networks. The more complex the architecture of the deep network, the more complex the features that the network is able to represent.

Deep learning models are composed of multi-processing layers that are specifically used for data that presents multiple levels of abstraction (for example: colours, shapes and forms in, pitch of speech, sentences, words, which language, background noise in audio, and time series data with all the previous examples in video). This complex, non-linear and unstructured data is given as input to the deep networks, which they transform into a hierarchical structure of features with different and multiple levels of abstraction [LeCun et al., 2015]. For example, in the case deep network model for image processing, the lower layers of the network may identify edges, while higher layers may identify the concepts relevant to a human such as digits, letters or faces. Deep learning methods have dramatically improved the ability of speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics and medical diagnosis such lung diseases [Anthimopoulos et al., 2016] or Alzheimer [T.Pringsheim et al., 2014] as mentioned previously in Section 2.2 [Goodfellow et al., 2016].

Deep learning research is a fast growing topic, and so there is a large variety of networks that have been created for specific problems. For example, CNN are specific for processing data with grid-like topologies, such as images as well as speech and audio. Meanwhile, Recurrent Neural Networks (RNNs) are specifically designed to work better with sequential data such as text, speech and audio or video [Nielsen, 2015].

This research focuses on CNNs for image classification, and thus a deeper description of these is given in the Section 2.3.1.

2.3.1 Convolutional Neural Networks

In deep learning, a CNN is a class of deep neural network that is mostly used to analyze visual imagery. The term "convolutional" indicates that the CNNs employ a mathematical operation described as convolution. This is an operation on two functions that produces a third function expressing how the shape of one is modified by the other. CNNs use this convolutional operation in at least one of their layers [Goodfellow et al., 2016].

The architecture of a typical CNN is structured in a series of components. After the input layer, the first component is composed of two types of layers: convolutional and pooling layers. The last components consist of a flatten layer (to transform the grid-data into a vector) and fully connected layers to be used for classification and have the same function as a standard ANN [Goodfellow et al., 2016], finalising with the output layer. An example of this architecture is shown in Figure 2.1.

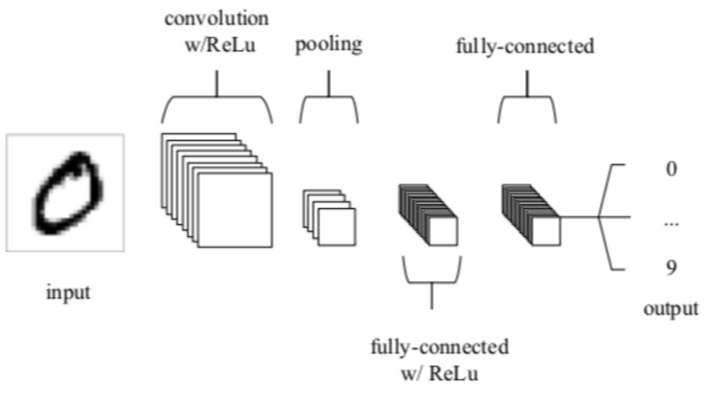


Figure 2.1: A common form of CNN architecture: convolutional layers activation function ReLu, pooling layers, and fully connected layers with ReLu and without. Figure reproduced from O'Shea and Nash [2015]

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. As described by LeCun et al. [2015], units in convolutional layers are organized in feature maps. Each unit from the convolutional layer is connected to local patches in the feature maps of the previous layer. The output of this local weighted sum is then passed through a non-linear activation function, such as a Rectified Linear Unit (ReLu). All units in a feature map share the same weights, while different feature maps in a layer use different weights. Since motifs that appear in one part of the image can appear elsewhere in the image, the same weight is used in different locations of the image to aid in detecting these repeating patterns. The convolutional layers can then detect local features from the previous layers, and this layers are known as the filter parts of the CNN LeCun et al. [2015].

The pooling layers focus on merging similar features with same meaning into one. With the use of max pooling or average pooling algorithms, the position of features forming a motif are extracted and represented in a reduced dimension. This, in turn, decreases the computational power required to process the data, as well as giving the network a higher resistance to noise by removing features that are not important (achieved specifically by the max pooling algorithm). Generally, CNNs contain two or three layers of convolution, non-linearity and pooling stacked, followed by more convolutional and fully-connected layers [LeCun et al., 2015].

Due to the reduction of parameters involved and the re-usability of weights (which is reflected in the filtering capability), limited pre-processing of data is required compared with other classification algorithms. In other words, CNNs successfully can be trained to understand the detailed information of the images without the need of transforming the input data.

Research into CNN architectures is growing rapidly, with new and improved architectures released every few weeks to months, making difficult to standardize the best CNN architectures for a given problem. Another active topic of research is how to evolve their topologies, in other words, the use of evolutionary algorithms to successfully optimize the architecture of the CNN. These are widely used to evolve topologies of ANNs, however research for deep networks is just starting to grow. A deeper view towards these algorithms is described in Sections 2.4 and 2.5.

2.4 Deep Neuroevolution

Deep neuroevolution is combining Evolutionary Algorithms (EA) for evolving deep neural networks. The use EA for evolving ANN has already a big research background but for deep networks, this topic is just starting to grow. There are three ways of applying neuroevolution: (a) training the network by evolving the weights and hyperparameters (training the network) (b) evolving the architecture and then train the deep network with an optimization algorithm as backpropagation and (c) evolve both topology and weights.

Recent studies from Stanley [2017] explain how neuroevolution can successfully be used for evolving deep networks. The work of Risi and Stanley [2019] explains how genetic algorithms can be used to encode and evolve deep reinforcement learning models and apply it successfully for a challenging car racing task. Poulsen et al. [2017] combines deep learning and neuroevolution to create a bot for a simple first person shooter game; the deep learning tool focuses on the visual recognition part and translates the raw pixels to features representations that the evolving networks will use as input to infer actions. In Vidnerová and Neruda [2017]' work, an EA is proposed for evolving the network of a selected deep network. These are only a few of the examples in literature combining EA to evolve deep networks.

In this research, a continuation of the previous work from Alissa et al. [2020], were the use of CNN in image classification to successfully support Parkinson's disease diagnosis. With deep neuoroevolution, the aim of of this project is to find a simpler architecture for a CNN with a good performance for the same dataset of the previous study.

2.5 Evolutionary Algorithms

What all EA have in common is their resemblance to Darwin's theory of evolution: evolve and obtain better individuals in each generation. These individuals are selected by their ability to perform better under their conditions (example: a fish that can swim better and more efficiently than others will be able to escape predators and hence, in this specific conditions, will be a more suitable individual). This better performance is given by a fitness value, which gives information of how close an individual solution is to achieving the set aims.

The algorithm starts by generating a random population of individuals that contain the parameters we want to evolve (called genes). Each individual of the population is evaluated and a fitness value will determine which individuals (with their weights) have performed better and will be selected as parents for the next generation. Generally, the offspring will be constructed constructed by slightly altering the genes of the selected parents, commonly using crossover (exchanging genes between the parents to form new individuals) or mutation (in one individual, altering its own genes genes for create different individuals) operators. Crossover and mutation techniques are ways to explore the search space generating solutions that are in a way closer to their parents [Luke, 2013].

2.6 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEAs) are a type of EA used to solve problems with multiple conflicting objectives. In order to understand the concept of conflicting objectives, consider the example of this research. The aim of this project is to obtain the simplest architecture of CNN that gives a good performance for image classification. Accuracy and complexity are the two conflicting objectives in this case. Thus, there is no single optimal solution and instead, there will be multiple optimal solutions that represent different trade-offs between the objectives. MOEA can be used to find these solutions and it represents them as a set of non-dominated solutions (a non-dominated solution is dominated by no others, it is optimal within a particular objective).

In a MOEA, solutions have more than one measure of fitness, known as their objective values. These values can be represented in an objective space where each axis represents one objective. The optimal solutions lie on some curve (in the case of bi-objective MOEA) through the objective space, called the Pareto optimal front (Figure 2.2) and contains all the best solutions to the problem [Luke, 2013].

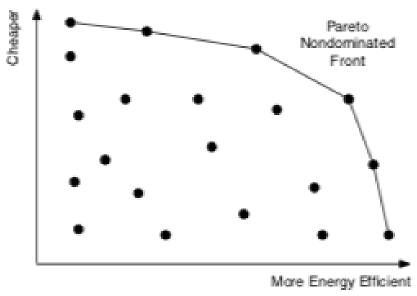


FIGURE 2.2: Pareto front of non-dominated solutions for a two-dimensional objective space, between cost and energy efficiency [Luke, 2013].

In the objective space there will be two different regions separated by the Pareto front of non-dominated solutions: the unachievable solutions which will lay beyond the Pareto front and the dominated solutions[1] which will lay in the enclosed region under the Pareto front. The multi-objective algorithm starts by creating the initial population, as usual, by creating randomly solutions. Each is then evaluated against all the objectives (Figure 2.3). During selection, the non-dominated solutions in the population are identified and copied into the child population, as well as typically some of the better dominated solutions to promote diversity in the child population, same procedure as in single objective EA (see 2.5). The other dominated solutions are discarded, and the population is filled up by applying crossover and mutation. The process is then repeated and gradually the population's Pareto front moves towards the optimal Pareto front.

One of the central issues in the design of effective MOEAs is how to maintain a good spread of non-dominated solutions, so as to ensure the entire Pareto optimal front gets explored. Ideally the population is scattered along the whole Pareto optimal front, and this can be achieved using several different methods. One such strategy is NSGA-II, that uses sparsity-ranking to achieve this [Luke, 2013]. This is further explained in Section 2.6.1.

[1]a solution is said to dominate when it is better at least at one objective and no worse in all others

```
Best ← individual picked at random from population with replacement
O ← {O_1, ..., O_n} objectives to assess with        ▷ In lexicographic order, most to least preferred.
t ← tournament size, t ≥ 1

for i from 2 to t do
    Next ← individual picked at random from population with replacement
    for j from 1 to n do
        if ObjectiveValue(O_j, Next) > ObjectiveValue(O_j, Best) then        ▷ Clearly superior
            Best ← Next
            break from inner for
        else if ObjectiveValue(O_j, Next) < ObjectiveValue(O_j, Best) then   ▷ Clearly inferior
            break from inner for
return Best
```

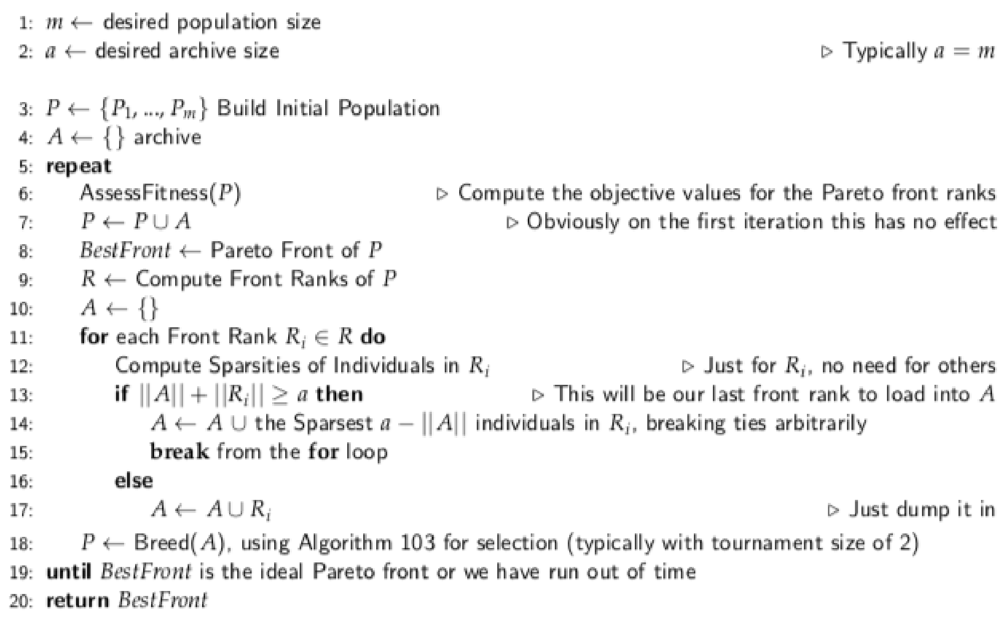
FIGURE 2.3: Multi-objective lexicographic with two objectives from Luke [2013]. In this algorithm, the objectives are ordered by the user preference in importance and solutions are compared based on the objective value of those preferences.

In summary, MOEAs aim to find all non-dominated solutions to a problem, which form a Pareto optimal set, by exploring trade-offs within a design space and compare solutions based on dominance. After the Pareto front is obtained, the user is now responsible for choosing among these solutions.

2.6.1 NSGA-II

There are many types of MOEAs available, but the best known and most widely used is NSGA-II, which stands for Non-dominated Sorting Genetic Algorithm version 2, first published by Deb et al. [2002]. Figure 2.4 shows an algortihm description of the NSGA-II.

At each generation, the NSGA-II algorithm initially assigns a rank to every population member based on their non-dominance (non-dominant solutions will have a higher rank, then this solutions are set aside and the next non-dominated solutions will be ranked as rank 2, for example, and so on). Also, at the beginning of each generation, an empty child population is created. For each front (in each generation), firstly up to half the child population is filled with the highest rank members (these solutions will be copied to the child population). Each solution in the next rank is given a sparsity value, based on the distance to its nearest neighbours and the solutions are sorted and ordered by decreasing sparsity in a list. The solutions from the top of this list will be copied across into the child population (until half the child population is occupied). The other half of the child population is filled using crossover and mutation from those already copied across [Luke, 2013].



```
$m \leftarrow$ desired population size
$a \leftarrow$ desired archive size                                      ▷ Typically $a = m$

$P \leftarrow \{P_1, ..., P_m\}$ Build Initial Population
$A \leftarrow \{\}$ archive
repeat
    AssessFitness($P$)                    ▷ Compute the objective values for the Pareto front ranks
    $P \leftarrow P \cup A$                         ▷ Obviously on the first iteration this has no effect
    $BestFront \leftarrow$ Pareto Front of $P$
    $R \leftarrow$ Compute Front Ranks of $P$
    $A \leftarrow \{\}$
    for each Front Rank $R_i \in R$ do
        Compute Sparsities of Individuals in $R_i$             ▷ Just for $R_i$, no need for others
        if $||A|| + ||R_i|| \geq a$ then                ▷ This will be our last front rank to load into $A$
            $A \leftarrow A \cup$ the Sparsest $a - ||A||$ individuals in $R_i$, breaking ties arbitrarily
            break from the for loop
        else
            $A \leftarrow A \cup R_i$                                          ▷ Just dump it in
    $P \leftarrow$ Breed($A$), using Algorithm 103 for selection (typically with tournament size of 2)
until $BestFront$ is the ideal Pareto front or we have run out of time
return $BestFront$
```

FIGURE 2.4: An abstract version of the NSGA-II from Luke [2013]. Algorithm 103 refers to Figure A.1 in Appendix A.

The overall objective of this algorithm is to obtain a set the best individuals that are not only close to the true Pareto front, but that they are also well spread out along it.

Chapter 3

Methodology

3.1 Overview

The aim of this project is to optimize the architecture of CNNs using MOEA for diagnosis of Parkinson's disease. In this section, the methodology used for this project is presented. Firstly an overview of the data preparation needed for CNNs is presented. Secondly, an outline of how the CNNs are developed, together with the neuroevolutionary frameworks available that include MOEA for the optimization of deep networks. Finally, an overview of the training, testing and validation steps of the model.

3.2 Data preparation

The dataset comprises information acquired from 87 subjects (58 PD patients and 29 healthy controls of comparable ages). All subjects were asked to copy the drawings of a wire cube and Archimedean spiral pentagon on top of template images, using an inking stylus and a digitising, pressure-sensitive Wacom tablet. Example drawings are shown in Figure 3.1. In each sample, time information of the x and y coordinates of each pen location, the angles in which the pen is used with respect to the x-y plane, and the relative pressure exerted against the tablet were stored as a multivariate time series dataset. The x and y coordinates and pressure values are represented in the range of $[0, 1]$, the pen angles in the range $[-1, 1]$, and timestamp entries are monotonic integer values starting from zero. When zero-pressure values are collected, it can be interpreted that the pen at this location was not in contact with the tablet.

Following the guidelines of the previous work from Alissa et al. [2020], the following preprocessing of the images is needed for the evaluation of the model: firstly, the x

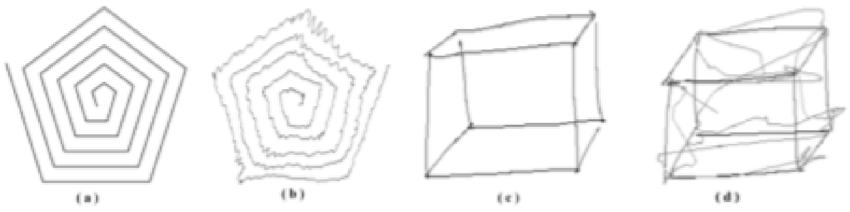


FIGURE 3.1: (a) Illustration of the spiral pentagon template, (b) a pentagon drawing gathered from a patient, (c) a cube drawing without zero-pressure information, and (d) a cube drawing with zero-pressure information [Alissa et al., 2020].

and y coordinate data is extracted, zero-pressure values and angles discarded, and the information is translated into a 2-dimensional black and white image (Figure 3.1.c.); secondly, zero-pressure information is added (coordinates where the pen passed without touching the tablet) as grey strokes to the black and white image (Figure 3.1.d.); then the black and white representation is extended to a grey-scale image by scaling the pressure values from $[0, 1]$ to $[0, 254]$.

Next, information is added for differentiating between areas where the pen did not pass and areas where the pen passed, but without touching the table. Then, the images are trimming and resizing to 32 by 32 pixels (other sizes are proposed in the previous paper but to minimize the computational process, only the smallest size will be used).

Afterwards, due to the imbalanced nature of the sample (more patients than controls) additional images are created from the originals, using augmentation techniques as specified in the previous work Alissa et al. [2020]. Augmentation techniques are also used when working with small datasets and DL networks, to improve the generality by including more information in the training data, which results in a stronger model. Examples of the new images created are shown in Figure 3.2.

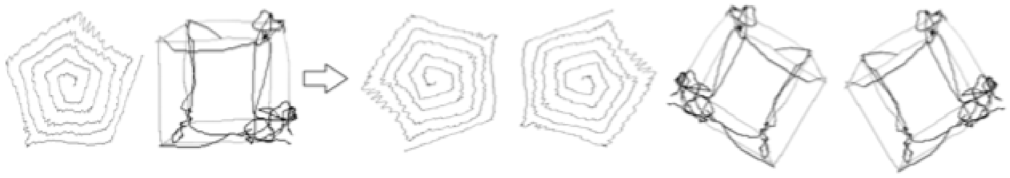


FIGURE 3.2: Examples of the augmentation process. Original images of a pentagon and a cube drawing (left) and examples of the generation of augmented images (right)

As a first stage of the project the pentagon dataset will be used for the training and testing of the networks, and hence, only this images will be needed to be pre-processed.

3.3 The deep learning network

Although other languages are available for working under deep-learning frameworks and neuroevolutionary networks, Python 3 has been chosen for the experimental work in this project. The most popular deep learning frameworks include PyTorch, Keras and TensorFlow, and the frameworks available for neuroevolution use Java or Python. Keras is the framework chosen as it allows for easy and fast prototyping with deep learning in Python while also offering a high-level API with a simple and consistent interface, in the Table **??**, an overview of the pros and cons of the most popular frameworks is shown. Personal choice of the author and consistency with the previous paper's decision on the framework ([Alissa et al., 2020]) which our research is building on has also been considered on the choice of using Keras. Hence, the choice for the neuroevolutionary frameworks will be based on a Python environment. A discussion of these frameworks can be seen in the next section 3.4.

Framework	**Pros**	**Cons**
Keras	High level API Simple architecture Good for prototyping Modular Specifically created for ANN Easy to debug	Slow Basic documentation
TensorFlow	High and low level API. Good visualization Good documentation Easy Model Building	Slow Bad debugging capabilities More specific
PyTorch	Good for debugging Modular Pre-trained models Specific framework for MOEA. Decision making feature.	Low Level API Very basic documentation Complex architecture

Table 3.1: Discussion of different frameworks available for deep learning

Based on the architecture of the previous work [Alissa et al., 2020] (Figure **?**), max-pooling layers will be used. Depending on the time available, experiments including average pooling layers could also be performed. All the activation functions used will be Rectified Linear Unit (ReLU), except for the last dense layer, where a sigmoid activation function was selected to map the binary output. The aim is to not modify the activation functions when evolving the architecture.

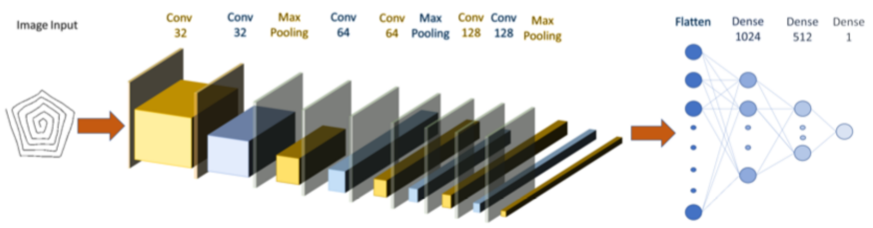


FIGURE 3.3: CNN architecture from Alissa et al. [2020], consisting of two convolutional layers with 32 filters followed by two convolution layers with 64 filters and another two convolutional layers with 128 filters, three max-pooling layers of size (2×2), six dropout layers, three dense layers and one flattened layer

3.4 MOEA

NSGA-II (Non-dominated Sorting Genetic Algorithm version 2) [Deb et al., 2002], PAES (Pareto Archived Evolution Strategy) Knowles and Corne [1999] and SPEA (Strength Pareto Evolutionary Algorithm) Luke [2013] are widely extended Evolutionary Algorithms, but not much information is available on how to implement with deep network topologies, the research that has been conducted mainly focuses on ANN. Although there are some examples in literature that use SPEA with the same database Vallejo et al. [2016], evolutionary objectives were very different compared with the objective of this project. NSGA-II is the most extended and widely used, and for this reasons and availability of examples for coding and implementations (at least for simple ANNs) it will be used in the project for the evolution of the CNNs.

3.5 Neuroevolutionary Tools

Before deciding how to encode the CNN in the MOEA, a wide look at the documentation and the tools available need to be done first.

Ideally, the same language should be used to evaluate and evolve the topology of the CNNs, since it will ease the computational effort. Since Keras (Python environment) has been chosen as the framework for developing the deep networks, in this section an overview of the available Python frameworks is given. Within neuroevolutionary frameworks, very little information is available about the implementation of deep networks in the frameworks available. Table 3.2 shows the pros and cons of the available frameworks.

Framework	**Pros**	**Cons**
Platypus	Specific framework for MOEA.	Under active development, may contain bugs. Very basic documentation.
Pymoo	Good documentation and support. Specific framework for MOEA. Decision making feature.	Under active development, code may conain bugs.
jMetalPy	Based on a well known framework for deep neuroevolution for Java. Very complete documentation and variety of features like: parallel computing, real-time visualization, variety of mutation and cross-over operators. Contains multiple examples for each MOEA algorithm.	Ongoing online modifications, may contain bugs.
DEAP	Well known framework for EA. Simple implementation.	Very basic documentation.
NEAT	Specific for NeuroEvolution of Augmenting Topologies. Good documentation. UberAI labs supports some deep neuroevolution algorithms with NEAT.	No implementation of MOEA available.

Table 3.2: Discussion of different frameworks available for neuroevolution in Python

Although Platypus framework [Hadka, 2015] claims to be specific for MOEA, the documentation is very basic and acknowledges the ongoing development and possibility of a code with bugs. On the other hand, Pymoo [Blank and Deb, 2020] includes good documentation and support while also being a specific framework for MOEA, which also includes a decision making feature. jMetalPy offers a more complete documentation and variety of tools and features for the implementation of EA, like a variety of operators for selection, cross-over and mutation; the more features available, the more potential combinatory options for evolving CNNs, and thus obtain the best architecture. As Pymoo and Platypus, it is currently under active development so the code could include some bugs. DEAP [Fortin et al., 2012] is a very well known framework for EA with a simple and easy implementation but also with very basic documentation. In the case of NEAT [Stanley, 2015], although it is a framework specific for neuroevolution of augmenting topologies that comes with good documentation, no implementation in their page of MOEA is available and only a few papers [Abramovich and Moshaiov, 2016] have tried and successfully to implement MOEA with NEAT.

The best framework for this research would be one that would not be heavy computationally and that allows for an easy implementation with good documentation for possible debugging. None of the above choices have specific implementations for deep networks, so an estimation of how much time and processing the evolving of CNN cannot be done. Having discussed all frameworks, jMetalPy is the most complete framework and most likely to be robust enough for the desired implementation.

The option of hard-coding MOEA is still not discarded, for if the framework and deep learning do not pair well together.

3.6 Training, Testing and Evaluation

The biggest challenge in using EAs on deep CNNs is the evaluation of each individual. CNN training is computationally very heavy and even when using a powerful GPU it can take hours or even days to train fully. Thus, even though we can select the size of the population of individual CNN to be evolved, we need to take into account that this will require a large computational resource and very long evolving time, as a large number of individuals have to be evaluated.

The CNN model will be trained in supervised mode using backpropagation to update the weights and biases for image classification purposes. Different datasets for training and testing procedures will be employed, with 90% and 10% for training and validating respectively of the samples extracted from the main datasets. The samples for each procedure will be randomly selected and a 10-fold cross validation will be conducted on the evaluation of the accuracy of the framework, as is suggested by Alissa et al. [2020].

For initialization of algoritm, hyperparameters as initial learning rate, decay function and momentum will be based on the previous work too [Alissa et al., 2020]. Other techniques, such as using mini-batch learning or early stopping mechanism as a regularisation techniques, will be performed as well. A batch size of 16 is passed with the objective of speeding up the learning. To avoid overfitting, a twofold stopping condition is defined: a maximum number of epochs equal to 150 and stopping after 25 epochs without improvement in the validation set.

After training, the model is tested two-fold. Initially it is tested as a classifier to differentiate between healthy subjects and patients with PD using a test set of previously unseen images and obtaining a value determining the accuracy of the model. Secondly, the complexity of the model is evaluated by counting the number and variety of layers of each individual.

The value of accuracy will be given by the kappa statistic. Kappa measures the agreement between two parameters, which, in this case, are the predicted class and the original class. Based on Cohen [1960], kappa values > 0.75 are considered excellent, 0.4 to 0.75 is fair to good, and < 0.4 is a poor agreement. A kappa value could be negative, but it is unlikely in practice. This metric is chosen to maintain the continuity of the evaluation of the images with the validation samples from previous work Alissa et al. [2020].

These two-fold testing will update the values of the objective solution for the individual, and eventually update the objective space. By continuing with the steps of the MOEA algorithm, the Pareto Front will be updated and when termination is reached, and a result with the best individuals will be obtained.

Chapter 4

Requirements Analysis

4.1 Project Goals

The aim of the project is to optimize the architecture of CNN applied to Parkinson's Disease Diagnosis. MOEA is the algorithm used for the optimization of the CNN, finding a trade-off between a low complexity of the CNN and a high accuracy when tested as image classification. For the training and testing of the CNN, A dataset of images of drawing tasks will be used for the training and to successfully identify patients with PD.

The project will be be considered as successfully completed if the topology of CNN is successfully optimized by the MOEA under the two objectives described above. Due to the high computationally nature of this project there has to be a level flexibility in the decisions related to the training of the CNN and their evolvability. For instance, it may transpire that the evolvability of the CNN cannot be achieved with the available resources.

4.2 Deliverables

A dissertation paper, describing the project, planning, experimental part and evaluation, together with a link for the code created and details of how the framework was used and modified for deep networks. A discussion of the results and further work will also be included. The ultimate deadline of the project is the 24th of July completed with the submission of a dissertation report.

4.3 Priorities

The tasks planned for this project have different priorities and levels of requirement. In order to quantify this point, the MoSCoW prioritization technique [Khan et al., 2015] has been applied to the project tasks (Table 4.1). The MoSCoW method defines four levels of requirement: ‘must’ (the minimmum/core requirements), ‘should’ (expected to do), ‘could’ (is able to do, but low priority) and ‘won’t’ (out of scope).

Requirements	**MoSCoW**	**Priority**
Select adequate image samples train the CNN	Must	High
Design an adequate test and validation methodology for the study	Must	High
Design an adecuate MOEA framwork that can work with deep networks	Must	High
Gather information about other possible neuroevolutionary algorithms if MOEA is not possible to implement	Should	High
Gather material to make work publishable	Should	Low
Generalize the approach to any drawing image from the original dataset	Could	Low
Design an adequate test and validation methodology for the study	Must	High
Find extra computational power resources for personal computer	Must	Medium

Table 4.1: MoSCoW prioritization for tasks of the project

4.4 Project Risk Assesment

The table below (Table 4.2) shows the Risk, impact, likelihood and how to mitigate the circumstances that may delay or affect the person working on the project.

Risk	**Impact**	**Likelihood**	**Mitigation**
Author's illness	Medium	Medium	Communicate supervisor, work at home and apply for mitigating circumstances (if applicable)
Supervisor's illness	Medium	Medium	Communicate via Skype/email
Loss of	High	Low	Ask supervisors for access to the data
Insufficient processing power	High	Medium	Reduce cost: use less complex CNN, minimize number of CNN used in experiments (at each generation in MOEA), use mini-batch to evolve CNN. Modify objectives accordingly of the experiments. Check with university for additional resources
Inability to fulfil all requirements in the project time-frame	High	Medium	Modify requirements in order to generate results for the project (ex: change choice of evolutionary algorithm)
MOEA frameworks non-compatible with deep networks	High	High	Considerate hard-coding the MOEA algorithm. Search for algorithm alternatives in papers and modify the requirements accordingly. Update supervisor
Data loss	Medium	Medium	Regular backup of codes and results
Tools unavailable at personal computer	Medium	Medium	Use University available resources

Risk	**Impact**	**Likelihood**	**Mitigation**
Disruption to life due to COVID-19 situation	Medium	High	Keep supervisor updated about the current personal situation, and if applicable ask for mitigating circumstances.
Dates of Exams	Medium	Medium	Make sure project is almost finished and written before dates close to exams. Update the Project Plan if needed in case of changes in exam dates

Table 4.2: Details of impact, likelihood and mitigation plan for each risk factor.

Chapter 5

Discussion of Professional, Legal, Ethical, and Social Issues

5.1 Legal Issues

No legal issues corresponding with the dataset used is arisen since data is anonymous and has the inform consent of the patients.

The code and frameworks used for this research is of Open access.

5.2 Ethical Issues

Ethical issues can be arisen from this project since the topic is related with medical diagnosis, specifically with Parkinson's Disease diagnosis. Wrong interpretations of the data or a misleading interpretation on the functionality of the algorithm can potentially cause negatively results in the diagnosis. It needs to be said that these research does not aim to substitute any objective clinical decision and must be only seen as a tool that offers support for the clinician in Parkinson's Diagnosis.

5.3 Social Issues

As computer-guided Medical diagnosis is a growing topic nowadays, special focus on providing sufficient and basic information on how this tools works or on what medical diagnosis research focuses is necessary in other to avoid rejection of the society due to misinformation and uncertainty of the research. Ignorance can only lead to hoaxes and distrust and it is in the hands of the scientists to avoid this.

Chapter 6

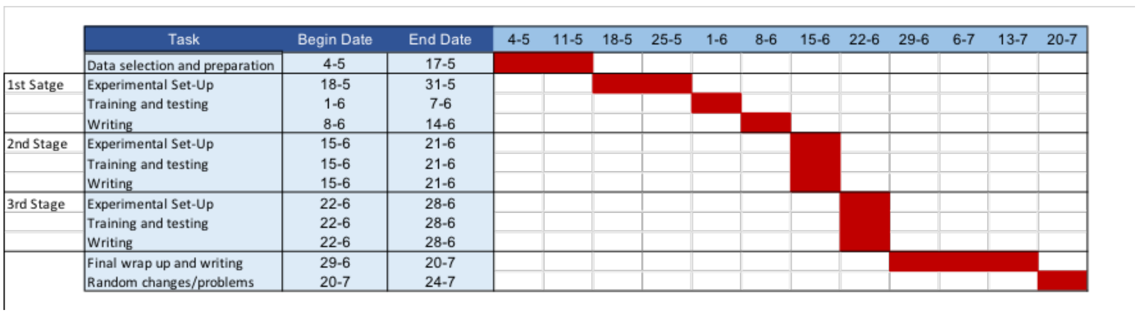
Project Plan

6.1 Ghant Chart

The development of the project will be undertaken from the 4th of May till the 24th of July. The Gantt chart 6.1 below outlines the different stages of the project that include the experimental section and the write, during a period of twelve weeks. The stages are explained as follows:

- Data selection and preparation: this section will focus on the understanding of the data and pre-processing of the images following the guidelines explained in the methodology.
- Experimental Set-Up: in the first stage, it will focus creation of the MOEA algorithm with jMetalPy together with the CNN with Keras. In the next stages, the focus will be in changing the parameters related with the MOEA in order to obtain different results (it can be related to the selection operator, crossover, mutation, algorithm termination, etc.).
- Training and testing: this part focus on the training and testing of the CNN. Depending on the time consuming of the optimization and training, some parameter modifications can be modified. Examples of these hyperparameters can be: modify learning rate, batch size, decay function.
- Writing: focus on the writing of the experiments and conclusions every time a new stage of complexity in the experimental section is reached.
- Final wrap up and writing: this section will focus on the final write up of the research thesis and gathering all the information needed from the different experiments, conclusions and data analysis.

- Random changes/problems: Allow one week before the deadline for any problem arousen out of schedule or plan.



	Task	Begin Date	End Date	4-5	11-5	18-5	25-5	1-6	8-6	15-6	22-6	29-6	6-7	13-7	20-7
	Data selection and preparation	4-5	17-5	🟥	🟥										
1st Satge	Experimental Set-Up	18-5	31-5			🟥	🟥								
	Training and testing	1-6	7-6					🟥							
	Writing	8-6	14-6						🟥						
2nd Stage	Experimental Set-Up	15-6	21-6							🟥					
	Training and testing	15-6	21-6							🟥					
	Writing	15-6	21-6							🟥					
3rd Stage	Experimental Set-Up	22-6	28-6								🟥				
	Training and testing	22-6	28-6								🟥				
	Writing	22-6	28-6								🟥				
	Final wrap up and writing	29-6	20-7									🟥	🟥	🟥	
	Random changes/problems	20-7	24-7												🟥

Figure 6.1: Gantt Chart

Appendix A

Extension of NSGA-II Algorithm

```
1:  $P \leftarrow$ population with Pareto Front Ranks assigned
2:  $Best \leftarrow$ individual picked at random from $P$ with replacement
3:  $t \leftarrow$ tournament size, $t \geq 1$

4:  for $i$ from 2 to $t$ do
5:      $Next \leftarrow$ individual picked at random from $P$ with replacement
6:      if ParetoFrontRank($Next$) $<$ ParetoFrontRank($Best$) then      ▷ Lower ranks are better
7:          $Best \leftarrow Next$
8:      else if ParetoFrontRank($Next$) $=$ ParetoFrontRank($Best$) then
9:          if Sparsity($Next$) $>$ Sparsity($Best$) then
10:             $Best \leftarrow Next$                                  ▷ Higher sparsities are better
11: return $Best$
```

FIGURE A.1: Non-dominated sorting lexicographic tournament selection with sparsity from Luke [2013]

Bibliography

Abramovich, O. and Moshaiov, A. (2016). Multi-objective topology and weight evolution of neuro-controllers. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 670–677.

Alissa, M., Lones, M. A., Cosgrove, J., Alty, J. E., Jamieson, S., Smith, S. L., and Vallejo, M. (2020). Parkinson's disease diagnosis using convolutional neural networks and figure-copying tasks. Manuscript submitted for publication.

Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., and Mougiakakou, S. (2016). Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE transactions on medical imaging*, 35(5):1207–1216.

Blank, J. and Deb, K. (2020). pymoo: Multi-objective optimization in python.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Dexter, D. T. and Jenner, P. (2013). Parkinson disease: from pathology to molecular disease mechanisms. *Free Radical Biology and Medicine*, 62:132—-144.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hadka, D. (2015). Platypus. https://platypus.readthedocs.io/en/latest/index.html.

Hughes, A. J., Daniel, S. E., Ben-Shlomo, Y., and Lees, A. J. (2002). The accuracy of diagnosis of parkinsonian syndromes in a specialist movement disorder service. *Brain*, 125(4):861–870.

Hughes, A. J., Daniel, S. E., Kilford, L., and Lees, A. J. (1992). Accuracy of clinical diagnosis of idiopathic parkinson's disease: a clinico-pathological study of 100 cases. *Journal of Neurology, Neurosurgery & Psychiatry*, 55(3):181–184.

Knowles, J. and Corne, D. (1999). The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. volume 1.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 7553:436–444.

Lesage, S. and Brice, A. (2009). Parkinson's disease: from monogenic forms to genetic susceptibility factors. hum. mol. genet. 18, r48-r59. *Human molecular genetics*, 18:R48–59.

Luke, S. (2013). *Essentials of Metaheuristics*. Lulu, second edition. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *ArXiv e-prints*.

Pereira, C. R., Pereira, D. R., Papa, J. P., Rosa, G. H., and Yang, X.-S. (2016). Convolutional neural networks applied for parkinson's disease identification. In *Machine learning for health informatics*, pages 377–390. Springer.

Poulsen, A. P., Thorhauge, M., Funch, M. H., and Risi, S. (2017). Dlne: A hybridization of deep learning and neuroevolution for visual control. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 256–263.

Reeve, A., Simcox, E., and Turnbull, D. (2014). Ageing and parkinson's disease: Why is advancing age the biggest risk factor? *Ageing research reviews*, 14.

Risi, S. and Stanley, K. O. (2019). Deep neuroevolution of recurrent and discrete world models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 456–462.

Stanley, K. O. (2015). Neat: Neuroevolution of augmenting topologies. `https://neat-python.readthedocs.io/en/latest/index.html`.

Stanley, K. O. (2017). Neuroevolution: A different kind of deep learning.

T.Pringsheim, N.Jette, A.Frolkis, and T.D.Steeves (2014). The prevalence of parkinson's disease: A systematic review and meta-analysis. *Movement disorders*, 29:1583—-1590.

Turner, R. S. and Desmurget, M. (2010). Basal ganglia contributions to motor control: a vigorous tutor. *Current opinion in neurobiology*, 20(6):704—716.

Vallejo, M., Cosgrove, J., Alty, J., Jamieson, S., Smith, S., Corne, D., and Lones, M. (2016). A multi-objective approach to predicting motor and cognitive deficit in parkinson's disease patients:. pages 1369–1376.

Vallejo, M., Jamieson, S., Cosgrove, J., Smith, S. L., Lones, M. A., Alty, J. E., and Corne, D. W. (2016). Exploring diagnostic models of parkinson's disease with multi-objective regression. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8.

Vidnerová, P. and Neruda, R. (2017). Evolution strategies for deep neural network models design.