Goldsmiths
UNIVERSITY OF LONDON

ANALYSIS OF PHYSIOLOGICAL SIGNALS USING MACHINE LEARNING METHODS

Hooman Oroojeni Mohammad Javad

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Department of Computing
Goldsmiths, University of London

Declaration of Authorship

I, Hooman Oroojeni Mohammad Javad hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Date: 28 July 2021

Acknowledgements

I would like to express my utmost gratitude to the following people who have made this thesis possible through their support and valuable input:

Firstly and most importantly, I am indebted to my primary supervisor Dr Mihalis A. Nicolaou, for his most valuable methodical guidance, comments, inspiration and mentorship throughout the years.

Secondly, I am immensely grateful to my second supervisor Dr Mohammad Majid al-Rifaie, for his technical expertise, encouragement, advice and for the many opportunities he gave me.

My friends, Dr Mohammad Ali Javaheri Javid, Asei Akanuma, and Dr Farzad Parvinzamir for the many technical and valuable exchanges of thoughts at different stages of my research work.

The department of computing, especially Dr Lahcen Ouarbya and Dr Aaron Gerow for helping me through all the difficulties I faced during this period. Also, all my former and current peers at Goldsmiths, especially those located in RHB329 lab for their support, the many lengthy conversations and comments, especially Terence Broad, Daniel Berio and Dr Prashant Aparajeya.

My friends Nadine Usta and Dima Fehmi for giving me their time and help in proofreading my thesis.

Above all, I would like to thank all my family, especially my parents and my wife Mira for their limitless support and love, which has enabled me to stay strong and complete my research successfully.

Abstract

Technological advances in data collection enable scientists to suggest novel approaches, such as Machine Learning algorithms, to process and make sense of this information. However, during this process of collection, data loss and damage can occur for reasons such as faulty device sensors or miscommunication. In the context of time-series data such as multi-channel bio-signals, there is a possibility of losing a whole channel. In such cases, existing research suggests imputing the missing parts when the majority of data is available.

One way of understanding and classifying complex signals is by using deep neural networks. The hyper-parameters of such models have been optimised using the process of backpropagation. Over time, improvements have been suggested to enhance this algorithm. However, an essential drawback of the backpropagation can be the sensitivity to noisy data.

This thesis proposes two novel approaches to address the missing data challenge and backpropagation drawbacks: First, suggesting a gradient-free model in order to discover the optimal hyper-parameters of a deep neural network. The complexity of deep networks and high-dimensional optimisation parameters presents challenges to find a suitable network structure and hyper-parameter configuration. This thesis proposes the use of a minimalist swarm optimiser, Dispersive Flies Optimisation (DFO), to enable the selected model to achieve better results in comparison with the traditional backpropagation algorithm in certain conditions such as limited number of training samples. The DFO algorithm offers a robust search process for finding and determining the hyper-parameter configurations. Second, imputing whole missing bio-signals within a multi-channel sample. This approach comprises two experiments, namely the two-signal and five-signal imputation models. The first experiment attempts to implement and evaluate the performance of a model mapping bio-signals from A to B and vice versa. Conceptually, this is an extension to transfer learning using Cycle Generative Adversarial Networks (CycleGANs). The second experiment attempts to suggest a mechanism imputing missing signals in instances where multiple data channels are available for each sample. The capa-

bility to map to a target signal through multiple source domains achieves a more accurate estimate for the target domain.

The results of the experiments performed indicate that in certain circumstances, such as having a limited number of samples, finding the optimal hyper-parameters of a neural network using gradient-free algorithms outperforms traditional gradient-based algorithms, leading to more accurate classification results. In addition, Generative Adversarial Networks could be used to impute the missing data channels in multi-channel bio-signals, and the generated data used for further analysis and classification tasks.

Table of Contents

List of Figures

List of Tables

1. Introduction

If there's one thing that people love more than chocolate,
it's science claiming that chocolate is good for you.
Leo Benedictus

According to the WHO, cardiovascular disease (CVD) is the number one cause of death globally, with approximately 18 million deaths recorded annually [4]. As such, the use of electrocardiogram (ECG) monitoring devices play a vital role in diagnosis and control of CVDs. While technology has greatly aided the evolution of these devices, there remain unexplored research opportunities for Machine Learning to further improve the efficacy of these devices. The accurate detection of false alarms in ECG monitoring systems used in clinical settings, such as Intensive Care Units (ICUs), is one area in which ML offers potential to providing obvious benefits to both patients and the healthcare system. To clarify, false alarms occurring in ECG monitoring devices may result in a range of negative outcomes, such as noise disturbance, disruption of continuity of care, lack of sleep, all of which may impact patients' stress levels, and, more generally, compromise the quality of recuperative care afforded in ICU settings. It is important to note that only an estimated 2 - 9% of the ECG monitoring device alarms are considered critical.

This study presents a novel architecture and optimisation approach that imputes missing bio-signals within a sample using Generative Adversarial Networks (GANs) and suggests an alternative for neural network hyper-parameter optimisation. In this research, we deal mainly with arrhythmias, abnormalities in the heart function which can occur in healthy and unhealthy subjects. The ICU is equipped with ECG monitoring devices capable of detecting dangerous arrhythmias, namely asystole, extreme bradycardia, extreme tachycardia, ventricular tachycardia and ventricular flutter/fibrillation. Arrhythmias are potentially fatal and in line with AAMI guidelines, appropriate responses should be taken within 10 seconds of the event's commencement [5]. Triggering of the alarm when an arrhythmia occurs could improve the chance of saving lives. Misconfigurations, defective wiring, staff manipulation, and patient manipulation or movement may increase the false alarm rate to as much as 86%. Clinically, 6% to 40% of the ICU alarms proved to be lower priority incidents which did not require immediate responses [6]. False alarms stimulate mental discomfort in patients [7] and may desensitise the reactions of clinical staff, causing slower responses to triggered alarms [8]. True alarms which are rated with high priority and require an urgent response make up only 2 - 9% of all ICU alarms [9]; therefore, the detection and elimination of false alarms are important areas for research.

1.1 Motivation

In the case of faulty ECG devices or patient manipulation, there is the possibility of losing parts or the whole signal. The challenge of deciding to remove incomplete samples or impute missing signals while a small number of samples are available encouraged us to explore the possibility of imputing missing signals within a sample through each of the two-signal and the five-signal imputation approaches. The two-signal imputation approach is practical when only two signals are available for each sample; otherwise, the five-signal imputation approach should be used. This benefits from a single generator and discriminator model.

In the two-signal imputation approach, we use the CycleGAN to learn the cross-domain relations between ECG signals while also using paired data in a denoising auto-encoder (DAE) to learn between-view correspondences and denoise the signals generated by CycleGAN in order to improve their quality by learning a shared representation from pairs (x, y). More specifically, the CycleGAN estimates and maps the given lead II using lead V and vice versa by concentrating on domain translation. The DAE extracts features from each view in belonging primary parallel layers. Subsequently, the features are concatenated and fed to a stack of layers to reduce the dimensions and form a shared representation tensor. Lastly, each view is gathered through separate output layers. The reconstruction functions and internal representations of each view provide the basic structure of the multi-view data and are optimised jointly during the training process. On the other hand, the five-signal approach uses the desired target domain label l to map a physiological signal from one domain to the target domain.

Additionally, the hyper-parameter optimisation method improves the classification performance and the generalisation capabilities in pattern recognition and regression tasks. The deep neural network structure, including the number of neurons, layers and hyper-parameter configuration plays a significant role in the training phase. Therefore, several network structures and hyper-parameter configurations are exercised to train a network. These experiments may result in the derivation of a set of models that generally have different performances on pattern recognition tasks. It is relatively challenging to find a suitable network structure and hyper-parameter configuration due to the complexity of deep networks and high-dimensional optimisation parameters.

Moreover, the network classifier used to predict results on a high dimensional and large-scale dataset has limitations, such as weak generalisation ability and instability in the training phase. To answer the above issues and further enhance the performance of deep networks in regression and pattern recognition tasks, the suggested approach mainly aims to construct an efficient model with its hyper-parameters. This approach offers a robust search process to determine the hyper-parameter configuration by using Dispersive Flies Optimisation (DFO).

1.2 Research Questions

Q1: How can the Swarm Intelligence and Evolutionary Algorithms help improve classification or regression tasks?

Q2: Would hybridisation of Swarm Intelligence and the Neural Networks techniques help overcome some of the drawbacks of using these approaches individually?

Q3: How useful are Generative Adversarial Networks (GANs) to impute missing signals? Furthermore, would the imputed signal improve the classification score?

1.3 Scope and Objectives

This study builds upon foundations of machine learning and population-based stochastic algorithms, where a dataset, including bio-signals used and multiple deep-neural networks implemented, is evaluated for the classification task. Since this dataset includes multiple channels of data for each sample and suffers from random missing signals, two approaches have been taken:

- **Approach1:** Select the samples that contain the same leads and do not have missing signals.
- **Approach2:** Attempt to include as many samples as possible and aim to impute the missing signals.

By taking **Approach1**, a deep neural network is implemented and trained using the traditional method (backpropagation) and accordingly evaluated. Subsequently, a novel architecture and optimisation approach is explored to find the optimal hyper-parameters of the model using the Dispersive Flies Optimisation (DFO) algorithm. The results have been gathered and a comparison established to compare the performance of the hybrid model using DFO versus backpropagation algorithm.

By taking **Approach2**, the model is implemented to impute the missing signals while using multi-channel bio-signal samples. This objective is achieved using GANs, which are analysed and explored. Two different setups to impute a signal using two-signal and five-signal models have been implemented and the results reported. Subsequently, the two methods used in **Approach1** to find the optimal hyper-parameters were utilised to gather the Physionet score (and accuracy) as a measurement.

The feature selection and engineering were not set as an objective for this research since the primary focus is on hyper-parameter optimisation and missing data imputation. We utilise the feature set suggested by [10].

1.4 Contributions

This thesis has made a number of contributions to the fields of computer science, machine learning, and swarm intelligence. These contributions are listed below:

C1: A new approach is suggested to impute missing signals within five-signal samples.

C2: A new hybrid system is suggested to optimise neural network hyper-parameters.

C3: An update for the DFO mechanism is suggested, including dynamic disturbance threshold (Δ) and population update equation.

1.5 Structure of Thesis

The structure of this thesis is organised as follows:

Chapter 2: Machine Learning

In this chapter, we review relevant research and literature in the field of Machine Learning. This chapter comprises two sections, namely Machine Learning and Generative Adversarial Networks (GANs). The first section offers a brief history of Machine Learning elements such as neural networks and their associated components. It then discusses deep learning and relevant applications. The second section provides an insight into GAN models and belonging loss functions and their real-world implemented applications.

Chapter 3: Swarm Intelligence for Machine Learning

In this chapter, we review relevant research and literature in Swarm Intelligence algorithms and their implementation in real-world applications as well as literature on Swarm Intelligence for Machine Learning. In addition, it offers a review of the approaches which have led to hybrid swarm-intelligence-machine-learning algorithms.

Chapter 4: Deep Neuroevolution for Bio-Signal Classification

This chapter explains our approach, data selection, feature selection, proposed hybrid algorithm, and results, along with a discussion of the results obtained.

Chapter 5: Signal Imputation with Adversarial Networks

This chapter explains our approach to impute missing signals using samples with two and multiple leads to collect the results using the hybrid algorithm proposed

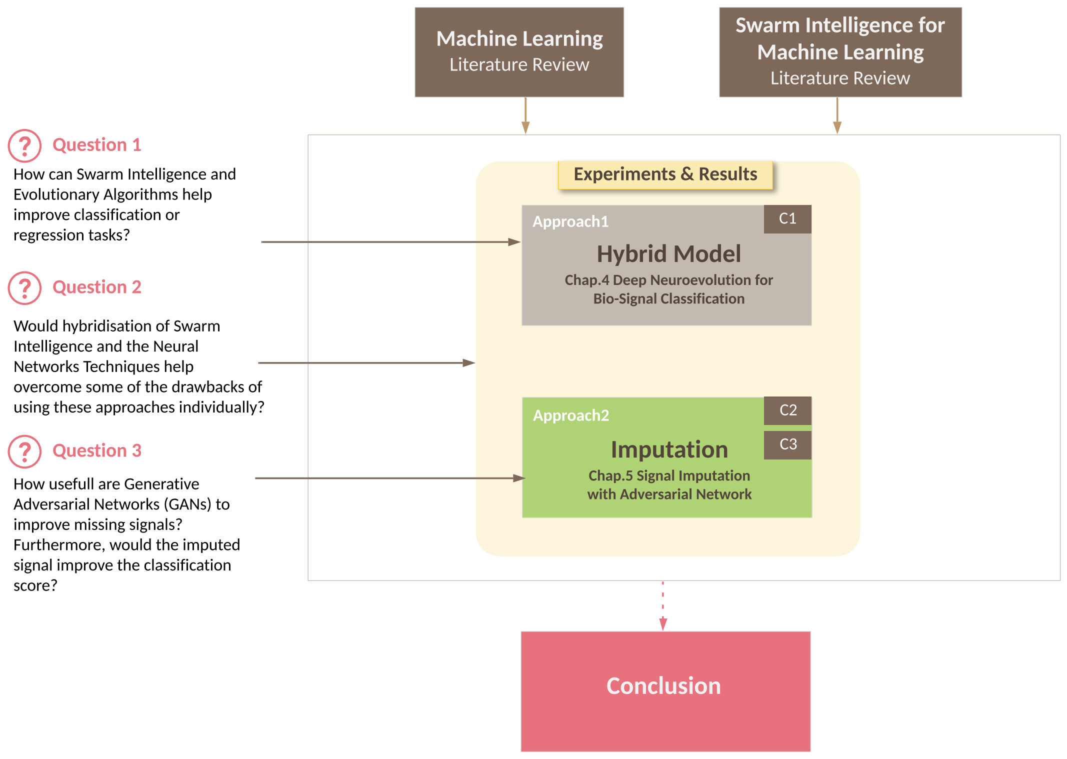
Machine Learning
Literature Review
Swarm Intelligence for Machine Learning
Literature Review
Question 1
How can Swarm Intelligence and Evolutionary Algorithms help improve classification or regression tasks?
Question 2
Would hybridisation of Swarm Intelligence and the Neural Networks Techniques help overcome some of the drawbacks of using these approaches individually?
Question 3
How usefull are Generative Adversarial Networks (GANs) to improve missing signals? Furthermore, would the imputed signal improve the classification score?
Experiments & Results
Approach1
C1
Hybrid Model
Chap.4 Deep Neuroevolution for Bio-Signal Classification
Approach2
C2
C3
Imputation
Chap.5 Signal Imputation with Adversarial Network
Conclusion

Figure 1.1: *The structure of this thesis. C represents contributions.*

in this chapter.

Chapter 6: Conclusion and Future Directions

This chapter summarises and synthesizes the thesis by discussing the results of the experiments, exploring further applications of the developed models, highlighting our contributions, and outlining directions for future research.

2. Machine Learning

2.1 Introduction

This chapter is formed to review the initial components of deep learning [11] and Generative Adversarial Networks (GANs), given that this thesis focuses on utilising this knowledge while performing experiments. The first section includes discussion of the concepts of neural networks and deep neural networks along with their components and elements. It also addresses various applications for each type. The second section addresses the components of GANs, Loss-Variant and Architecture models and their applications.

2.2 Neural Networks

A mathematical model comprising a series of synapses or computational units is called an Artificial Neural Network (ANN). Synapses are artificial neurons interconnected by uni-directional communication channels. A specific numerical weight determines the relative influence of each synapse. Concerning connectivity, a common approach is to form an acyclic-directed graph by combining a layered feed-forward neural network with full connectivity between adjacent layers. This model is referred to as a Recurrent Neural Network (RNN) when it exhibits loops. An RNN will be identical to a feed-forward network if we unroll it over time and each time stamp comprises a layer [12, 13]. Between layers, to ensure equivalence, the weights need to have a specified value. In such circumstances, the challenge is that the size of the obtained deep neural network becomes immense due to its expansion.

This will add extra complexity to the training of the model. The research on RNNs is currently a domain of interest as there have been remarkable improvements such as research undertaken by [14], [15] and others.

Concerning ANNs, there are many studies on medical datasets. Some of the examples of the usage of ANNs are: 1- [16] use of ANNs to diagnose a tumour by applying Multi-Layer Perceptron (MLP) as a practical pattern recognition instrument to differentiate between cancer patients and healthy ones. 2- [17] development of the MLP method to assist in diagnosing patients with heart diseases based on a decision support system. 3- [18] diagnosis of lower back pain and sciatica by training an MLP network to recognise these conditions. The interested reader is referred to [19] for more information.

Types of Parameters

The terms "parameters" and "weights" are often used interchangeably as they are typically the central focus of many ANN researchers. However, while investigating further to identify the components of a neural model, it is essential to define the following terms:

Weight w_{ij} - the value controls the level of conduction of a synaptic signal between two pairs of neurons (i, j).

Connection c_{ij} - It is possible to add or remove the synapses between neurons. This can be achieved by the assignation of a zero or non-zero value to weight in a fully connected network. This link between neurons is called a connection.

Neuron model n_i - Neuron types change frequently, and those in the brain are either inhibitory or excitatory [20], so when ANN weight polarity changes, this is considered a change of neuron type. This type of change is not feasible in the brain, but it is possible to transform a neural network model, with mixed synapses, into a network that solely includes inhibitory or excitatory neurons [21]. The study in [22] presents a variety of neuron types used in the nervous system such as Pyramidal Cells (PCs), Nest Basket Cells (NBCs), Small Basket Cells (SBCs), Large Basket Cells (LBCs), Bi-Tufted Cells (BTCs) and Martinotti Cells (MCs). Each of these types has specific properties; as such, it is possible to obtain interesting results by combining several types. This combination can be implemented in the ANN context, such as in the case of Long-Short Term Memory (LSTM), [23] which is a typical network element used frequently and successfully in handwriting recognition [24].

Network model *s* - Usually, in the design stage, the number of neurons and layers have fixed parameters. However, these properties can also be optimised to minimise error, for instance, by using evolutionary genetic algorithms to construct the network [25].

Learning

The process of discovering model parameters using a data-driven mechanism is called learning. The process of learning, the classification of learning algorithms, and relevant learning trends have been described here. Machine Learning techniques are used to minimise the workload for humans by substituting it with machine computations for specific tasks. With a suitable algorithm, it is possible to trade off the manual effort of designing solutions with the machine's memory and computational powers.

Traditionally, programmers have been principally responsible for developing algorithms, but the length and complexity of the code produced has not increased in linear correlation with the growth of the model needed, which has caused problems due to scarcity of processing time and available resources. On the other hand, a relatively complex formula is used by Machine Learning techniques without the traditional constraints rendered by the size of the problem. This increases the potential for Machine Learning to reduce software development limitations. The domain-specific expert-based solutions are being replaced by a data-driven approach due to fast-changing demands from the IT industry. The domain-specific models need extra design effort. The complexity of the problem is turned into values of neural network parameters, which are determined as a function of a training set, thereby making direct programming redundant. Additionally, the training set is precisely where the effort has shifted. Having rich training data and using proper parameter tuning increases the quality of overall machine learning algorithms since they are highly data-driven. The size of the training set has the most substantial influence on output accuracy, although the quality of data also has a major impact on the results. As a result, the increase in hardware performance, and the quality of training sets and machine-learned solutions, are improving. Speech recognition using large-scale deep neural networks has become very successful [26] and can exceed solutions programmed by human experts; for example, a deep-learning model achieved higher accuracy than solutions based on complex features such as SIFT [27].

Learning Algorithms

This section summarises three well-known learning algorithm approaches, including supervised and unsupervised learning and reinforcement learning.

Supervised

Supervised algorithms use labelled data sets for training. This method aims to generalise the relationship between the data and the labels to the unknown samples. The cost function is specified concerning the desired output. The learning process utilises a gradient descent-based optimisation method to minimise the cost function. One of the most popular methods for this purpose is called the backpropagation algorithm [28]. This algorithm is a generalised delta rule that applies to the model a collection of limitations. For instance, activation function of the nodes has to be uniform and differentiable. Other general supervised algorithms can also be used. However, they may not primarily be designed to be applied to an ANN, for instance, simulated annealing [29].

Unsupervised learning

The labelling of data is a laborious and time-consuming task carried out after data has been collected for a specific purpose to serve as an input to the learning algorithm. This is the reason why most available data are generally not labelled. Clustering and hidden Markov models (HMMs) are included in unsupervised approaches. They have proven to be useful in neural networks, mainly when combined with supervised learning, specifically in deep learning [30, 31, 32]. From a biological plausibility perspective, unsupervised clustering occurs in the mammalian brain's primary visual cortex, making it possible to generate localised receptive fields by using an unsupervised learning algorithm [33]. An equivalence between the expectation maximisation algorithm (EM) and STDP is another type of connection to biological models that applies under certain conditions[34]. The EM algorithm is a statistical method, including separate specialised learning agents, discovering the hidden relationships within complex input data.

Reinforcement

This learning approach is structured on the interactions between an agent and the environment. The agent selects an activity from a probability distribution, which produces a response from the environment that can be measured. The learning aims to maximise the total reward function. An interesting result has been obtained in [35], where applies these principles to ANNs. The input to the model is visible to the screen and is supposed to learn to play several Atari2600 games. In the same way that dopamine is used in the brain as a reward signal, providing hints to synapses in reinforcement learning by saving the temporary weights, their changes can also be seen as biologically plausible [36].

2.3 Backpropagation

The supervised learning multi-layer feed-forward neural network algorithm was proposed by Rumelhart, Hinton and Williams [37]. The oldest and most popular version of this type of algorithm is Backpropagation Neural Network (BPNN)[38]. The main aim of supervised training is frequently updating the network weights in order to minimise discrepancies between the actual outputs of that network and relevant labels. This method calculates the gradient of a loss function with respect to all the weights in the network. In an attempt to minimise the loss function, the weights are then updated by the optimisation method that this gradient feeds into. The basic equation of a backpropagation Algorithm is:

$$W_{t+1} = W_t - \eta \frac{\partial J}{\partial W_t} \tag{2.1}$$

where η is the learning rate and J and W represent loss function and model parameters respectively.

Components of Backpropagation Algorithm (BPA)

BPA uses the gradient descent learning rule to update model parameters. This rule requires that parameters such as initial weights and biases, learning rate value and the activation function are selected carefully. Without this this, slow network convergence, a network error or even convergence failure may occur. These shortcomings have led previous researchers to propose a range of variations in the gradient descent BPNN algorithm in order to improve training efficiency [39].

- **Activation Function:** The activation function, or transfer function, is used to transform the activation level of a unit (neuron) into an output signal. This function is applied to the output nodes of each layer before it is fed to the subsequent layer.

  The activation function is used for two purposes. The first is to change the unit into an active (near +1) one when the correct inputs are provided or an inactive (near 0) one when the incorrect inputs are given. The second prevents the NN from collapsing into a linear function by making the activation non-linear. A few of the basic types of activation functions are the identity function, the step function, and the sigmoidal function. As the input of these functions changes, the output varies continuously but not linearly. On another note, the sigmoid units are more similar to real neurons than linear or threshold units.

- **Learning rate coefficient (η):** The learning rate coefficient determines how to adjust the size of the weights at each iteration - the reason it influences the convergence rate. Selecting a wrong coefficient may lead to a failure in the convergence. For instance, if the learning rate is too fast or too slow, this would damage the network convergence.

- **Momentum Term (α):** Adding momentum to the gradient equation can enhance the convergence rate. This can be accomplished by adding a fraction of the previous weights change to the current one. Rumelhart et al. [28] introduced a commonly-used update rule, including this type of momentum term. The updating equation used by Rumelhart is defined as follows:

$$[\Delta W]^{t+1} = -\eta \frac{\partial J}{\partial W} + \alpha [\Delta W]^{t} \tag{2.2}$$

It is added to smooth out oscillation and increase convergence speed.

Proportional Factor (β):

The standard Backpropagation Algorithm (BP) usually utilises two-term parameters; Learning Rate α and Momentum Factor β, but sometimes a third term called Proportional Factor is also added to increase the convergence speed, and to escape from local minima.

- **Cost Functions:** The Backpropagation algorithm uses the Mean Squared Error (MSE) cost function. There are some drawbacks to MSE that have been observed, such as incorrect saturation and the tendency to be trapped in the local minima, leading to slow convergence and poor performance. Also, the squaring in MSE emphasises reducing the larger errors rather than the smaller ones. To rectify this, research was conducted to find better cost functions and as a result new ones were proposed such as the Bernoulli error measure [40], New Modified cost function [40], Classification-Based (CB) cost functions [41].

Backpropagation (BP) is a well-established neural network training algorithm as it allows itself to learn and improve, thus achieving higher accuracy than other algorithms [38], and it is well known for this reason. [42] worked on comparing genetic and backpropagation algorithms through different problems such as Sin function, Iris plant and Diabetes datasets and deduced that BPLA is faster than Genetic Algorithms (GA) in terms of training speed as well as CPU time required. [43] has proven that BPLA outperforms GA when experimenting on pattern recognition. On the other hand, BP is used less frequently as it requires longer times to train the network to achieve the best possible results. With larger datasets, BP suffers from the local minima issue, yet some researchers like [44] suggested novel ideas that would avoid the local minima problem in complex datasets by providing scope to increase the speed of BPLA for this type of data.

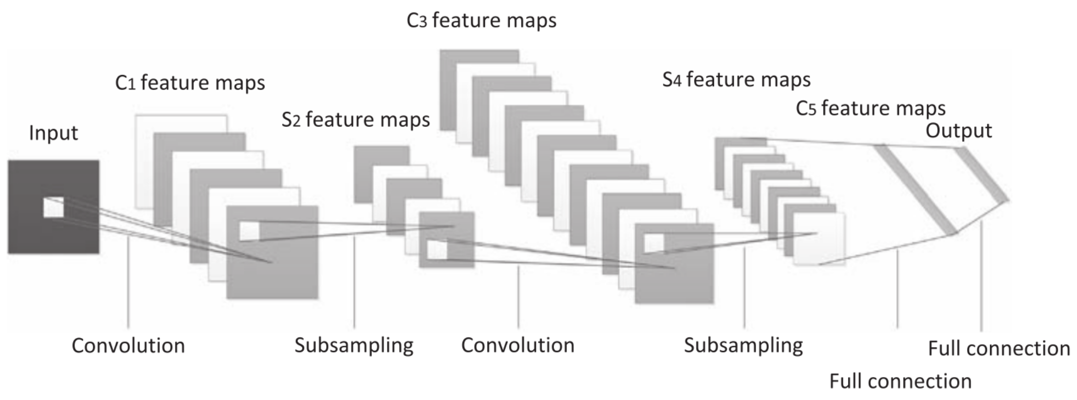

Figure 2.2: *Schematic structure of CNNs.An insight in the Convoloutional Neural Networks [1].*

2.4 Deep learning

In recent years, a significant development in NN research has been Deep Learning (DL), A fast algorithm which can be used in deep belief networks, as suggested by [45]. For each layer, this algorithm uses layer-wise unsupervised learning. Combining supervised with unsupervised learning produced significant results, which has made DL an active field of research. Currently, there are many DL applications; examples include speech recognition [46] and large-scale feature detectors such as the Google experiment [47]. A review of [48] offers a cogent historical analysis of the advances in the field of DL. The automatic development of features for the network through the usage of deep learning ensures there are fewer limitations for backpropagation as the random initialisation of the weights is no longer a dependency for the results. Backpropagation usage in a multi-layered network is not efficient and does not yield good results when there are more than 1-2 hidden layers; this is because local minima are highly likely to occur in this scenario. DL techniques have significantly improved big data analysis [49, 50]. The fields of social media, cyber-security, and medical informatics have yielded enormous data troves [51], and which are freely available to the public. From these datasets, DL can extract high-level features and then create hierarchical representations [1].

Convolutional Networks

Convolutional Networks[52], also known as Convolutional Neural Networks or CNNs, are types of NNs which are optimal for the processing of grid-like data. These data are either one-dimensional, such as time series that take samples at regular intervals or a two-dimensional grid of pixels such as image data. CNNs use convolution, a type of linear operation [53, 1], which takes two functions with real value arguments: $s(t) = (x * w)(t)$ where w is a function of valid probability density or kernel and x refers to an input. s is the output which is referred to as a $featuremap$. In machine learning, operations usually deal with multi-dimensional array parameters (tensors) as input and kernel. For instance, if we use convolutions over a 2-D image, considering input I and hidden layers kernel K, then we can define convolution following the equation and schematic structure of CCNs presented in figure 2.2 [1]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.3}$$

CNNs have had a significant effect on deep learning evolution as they represent a successful application of ideas inspired by studying the brain. They are among the primary deep models that perform well and have significant viability in commercial applications, for instance reading bank cheques [54] and handwriting recognition [55]. Various researchers applied Deep NNs to medical datasets. For instance [56] offers a new automatic approach proposed to detect cerebral microbleeds (CMBs) from magnetic resonance (MR) images by using 3D CNN. For example, [57] proposes a CNN approach to classifying interstitial lung diseases (ILDs) while [58] predicts neo-adjuvant chemotherapy using a CNN approach.

2.5 Generative Models

We can categorise machine learning implementations into Generative and Discriminative models. Discriminative models learn mapping an input to the desired output class; in addition, Generative models also learn the rule to generate the output from the input. A Discriminative model estimates the hidden parameters of the model or the conditional probability distribution of y conditioned on x: $P(y|x)$.The Discriminative model approximates the probability of y and $P(y|x)$ although x is known but does not have knowledge regarding marginal distribution of y and x independent from the other variables($P(y)$ and $P(x)$). This model has the potential to learn a map,$\hat{f}$, to offer an approximation of the distribution $P(y|x)$, which defines a boundary with an optimal split among available classes.

The Generative model estimates the joint probability distribution x and y, which can be represented as $P(y|x)p(x)$. This model has the ability to learn a map, $\hat{f}$, to approximate the distribution $P(y, x)$.

We can use Bayes' theorem to calculate $P(x|y)$ and $P(y|x)$, while $P(y, x)$ is estimated, having the knowledge that the joint probability is symmetrical. After moving around the $P(x)$ and $P(y)$ terms, we derive Bayes' theorem:

$$\begin{aligned} P(x,y) &= P(y,x) \\ P(x|y) &= \frac{P(x,y)}{P(y)} \\ P(y|x) &= \frac{P(y,x)}{P(x)} = \frac{P(x,y)}{P(x)} \\ P(x|y)P(y) &= P(y|x)P(x) \\ P(y|x) &= \frac{P(x|y)P(y)}{P(x)} \end{aligned} \tag{2.4}$$

An advantage of Generative models is their ability to process and learn from complex and big data by utilising a rather modest number of parameters. In addition, in contrast to Discriminative models, they can learn highly relevant features from datasets without processing their labels. Such models have been applied to different concepts, such as speech synthesis, model-based control, and image generation.

The most recent work in Generative models has focused on GANs and likelihood-based methods, including auto-regressive models, Variational Auto-encoders (VAEs), and flow-based models. The following sections describe likelihood-based models and their variations. Later, I will describe the GAN framework in detail.

2.5.1 Autoregressive Models

This model estimates the conditional distribution of y with dependency on the previous time-step or offered values of y. For instance, in audio composition, the sample is estimated with consideration of previous audio samples and spectrograms. The simplest form of an auto-regressive model with dependency on the previous time-step and time-invariant bias term can be demonstrated as follows:

$$Y_t = \alpha + \sum_{i=1}^{p} \beta_i Y_{t-1} + \epsilon_t, \tag{2.5}$$

Models' coefficients and bias are represented as β and α respectively. Y_{t-1} and ϵ_t appear for previous time-step and white noise respectively. The current output is explicitly dependent on the former output. Auto-regressive models use maximum likelihood estimates as a training approach, which adds benefits of stability and simplicity.

An example of auto-regressive models for image synthesis is PixelCNN [59], and for audio synthesis WaveNet [60].

2.5.2 Variational Autoencoders

The aim of Auto-encoder models is to model the joint probability of the latent variable and the observed data: where $P(x, z)$ can be represented as $P(x|z)P(z)$. Using Bayes' rule, we calculate the posterior probability of z given x; $P(z|x)$ in eq. 2.6 since we are in favour of finding optimal z values that produce observed data:

$$\frac{P(x|z)P(z)}{P(x)} \tag{2.6}$$

Variational auto-encoders are more straightforward to train and infer in parallel, in comparison with auto-regressive models, although they are difficult to optimise. An example of this model is Deep feature-consistent Variational auto-encoders to generate images [61]. The images produced with models based on VAEs tend to be blurry however.

2.5.3 Reversible Flows

Flow-based generative models are naturally reversible and a single model P with a parameter Θ estimates $P_\Theta(x|z)$ and $P_\Theta(z|x)$.

Such an attribute of reversible flow models gives the means-precise log-likelihood evaluation and latent variable inference in the absence of approximation. Reversible flow models give impressive results in speech and image synthesis. A example of speech synthesis is the work done by NVIDIA [62] and for image generation, GLOW [63], which presents faces generated by a flow-based model. The process of discovering model parameters using a data-driven mechanism is called learning.

2.6 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are able to learn the distribution of data and generate believable fake samples. Producing additional samples using GANs can help create deeper NNs while avoiding overfitting the model [64]. One of the main advantages of GANs is that, unlike other approaches that use approximation methods to compute intractable functions or inference, such as VAEs, GANs do not require an approximation method.

GANs are increasingly becoming an interesting area for the research community [65, 66, 67, 68]. Many domains, such as semantic segmentation [69, 70, 71, 72, 73], computer vision [74, 75, 76, 77, 78, 79], natural language processing [80, 81, 82, 83], and time series synthesis [84, 85, 86, 87, 88] have taken advantage of GANs. In comparison with other generative models, GANs have the capability to handle sharp-estimated density functions, generating desired samples in an efficient way, with satisfactory compatibility with internal components of the neural network [89]. Although researchers demonstrate successful scenarios in the field of computer vision, GANs are hard to train [90, 91, 92, 93] and evaluate [94, 95, 96, 97, 98, 99, 100].

Different GAN types are introduced in the literature for better performance, and can be categorised into Architecture-variants and Loss-variants. In the following sections I briefly discuss these.

The architecture of a GAN comprises two components: The Discriminator, whose role is to distinguish between real and generated samples, and the Generator, which generates samples to fool the discriminator. The aim of a GAN is to learn the distribution of generator P_g that approximates the distribution of real data P_r. A GAN is optimised through considerations of joint loss function for generator and discriminator:

$$\min_G \max_D \mathbb{E}_{X\sim P_r} \log[D(x)] + \mathbb{E}_{Z\sim P_Z} \log[1 - D(G(Z))]. \tag{2.7}$$

GANs have a set of advantages that distinguish them from traditional deep generative models and enable them to achieve state-of-the art performance to produce synthetic data. They can produce superior output; any generator model can be trained, and latent variables can have any size [89].

2.6.1 Architecture-Variant GANs

Architecture variant GANs are introduced to help in data generation challenges such as text-to-image generation [101], image completion [102], image-to-image transfer [103], and image super-resolution [104]. In the following sections I offer a review of this category of GANs in the context of improving sample quality, training stability, and sample diversity improvement.

Fully-Connected GAN

In Fully-Connected GANs (FCGANs), fully-connected neural networks are used to implement both generator and discriminator [65]. This model does not generalise performance for complex image types, as results from some simple datasets demonstrate [105, 106, 54].

Laplacian Pyramid of Adversarial Networks

This model aims to upscale images, i.e. achieve higher resolutions from images with lower resolutions [107, 108].

Deep Convolutional GAN

This model uses de-convolutional neural networks in generator since they perform well in spatial up-sampling and visualising the features for convolutional neurons [109, 110]. This enables Deep-Convolutional GANs (DCGANs) to produce higher resolution images.

Boundary Equilibrium GAN

This model uses an auto-encoder for discriminator and its loss function uses Wasserstein distance instead of directly matching distribution of data [111]. This approach prevents the discriminator from winning easily against the generator at early stages of training because of the discriminator's loss modification. The generator can reconstruct data more easily for the auto-encoder since the generated data is close to zero at the beginning and the generator has not yet learned the data at the early stages.

Progressive GAN

This architecture takes advantage of using progressive neural networks [112] that are able to retain prior knowledge due to the use of lateral connections to formerly-learnt features. In this implementation, both generator and discriminator grow throughout the training process while all parameters remain trainable. This training strategy makes the process of learning for both models more stable and results in impressive and plausible samples [112, 113, 114].

Self-attention GAN

The main reason for this implementation is the limitation of CNNs in learning multi-class image datasets because they are able to only capture local spatial information and possibly not cover adequate structure by receptive field. This may cause a shift in key components of the image while generating it [115]. Self-attention GANs (SAGANs) implement a self-attention procedure for the generator and discriminator of the GAN. This algorithm shows improved performance on multi-class image generation while tested on ImageNet datasets [116].

BigGAN

This architecture is based on SAGAN with a much bigger batch size and more complex model and, as a result, this produces diverse and high quality images [114].

2.6.2 Loss-Variant GANs

Another approach to increase the performance of GANs is to explore possible improvements in the original loss function (see eq. 2.7). The original loss function offers global convergence and optimisation, although it is likely to encounter instability problems while training [65]. If the discriminator is optimised for any generator, the global optimality is attained and therefore the derivative of the discriminator in eq. 2.7 is zero. In summary we have

$$\begin{aligned}-\frac{P_r(x)}{D(x)}+\frac{P_g(x)}{1-D(x)}&=0\\ D^*(x)&=\frac{P_r(x)}{P_r(x)+P_g(x)}\end{aligned} \tag{2.8}$$

Where $D^*(x)$ is the optimal discriminator, $P_r(x)$ and $P_g(x)$ are real data and generated data distribution respectively. x serves as real and generated data. When considering eq. 2.7 and 2.8, the loss function of the generator while the discriminator is optimised is as follows:

$$\mathcal{L}_G=\mathbb{E}_{x\sim P_r}\log\frac{P_r(x)}{\frac{1}{2}[P_r(x)+P_g(x)]}+\mathbb{E}_{x\sim P_g}\log\frac{P_g(x)}{\frac{1}{2}[P_r(x)+P_g(x)]}-2\cdot\log 2. \tag{2.9}$$

The above equation (2.9) is related to Kukkback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence measurement metrics. Considering these metrics, the loss function for the generator can be reformulated as

$$\begin{aligned}KL(p_1\parallel p_2)&=\mathbb{E}_{x\sim p_1}\log\frac{p_1}{p_2},\\ JS(p_1\parallel p_2)&=\frac{1}{2}KL(p_1\parallel\frac{p_1+p_2}{2})+\frac{1}{2}KL(p_2\parallel\frac{p_1+p_2}{2}),\\ \mathcal{L}_G&=2\cdot JS(p_r\parallel p_g)-2\cdot\log 2.\end{aligned} \tag{2.10}$$

While training discriminator step-by-step, optimisation of generator becomes equal to minimising JS divergence between p_g and p_r. The JS divergence causes unstable training problems since discriminator often wins against generator.

If there is no overlap between p_r and p_g, JS divergence remains unchanged, and if they have an overlap, it means the gradient of JS divergence for training the generator is non-zero [117]. For instance, when discriminator is close to optimal, the vanishing gradient will emerge for generator. It is highly likely that p_r and p_g have poor overlap or do not overlap at all.

To avoid gradient vanishing, the original GANs paper [65] suggests minimising

$-\mathbb{E}_{x\sim p_g}\log[D(x)]$ for training generator, although this approach will lead to mode-dropping problems. The $KL(p_1 \parallel p_2)$ with an optimal $D*$ can be redefined as follows

$$\begin{aligned} KL(p_g \parallel p_r) &= \mathbb{E}_{x\sim p_g}\log\frac{p_g(x)/(p_r(x)+p_g(x))}{p_r(x)/(p_r(x)+p_g(x))}, \\ &= \mathbb{E}_{x\sim p_g}\log\frac{1-D*(x)}{D*(x)}, \\ &= \mathbb{E}_{x\sim p_g}\log[1-D*(x)] - \mathbb{E}_{x\sim p_g}\log[D*(x)]. \end{aligned} \tag{2.11}$$

In eq. 2.7 by switching the order of the two sides, the loss function for generator will be

$$\begin{aligned} -\mathbb{E}_{x\sim p_g}\log[D*(x)] &= KL(p_g \parallel p_r) - \mathbb{E}_{x\sim p_g}\log[1-D*(x)], \\ &= KL(p_g \parallel p_r) - 2\cdot JS(p_r \parallel p_g) + 2\cdot\log 2 + \mathbb{E}_{x\sim p_x}\log[D*(x)], \end{aligned} \tag{2.12}$$

where in the updated loss function in eq. 2.12, the last two terms are constant and the first terms advance the generated distribution towards real distribution and the second terms aim to push in the opposite direction. Considering the circumstances, this will lead to instability while training the generator. Also KL divergence is a distribution measurement which is asymmetrical:

- $When p_g(x) \to 0, p_r(x) \to 1, KL(p_g \parallel p_r) \to 0.$
- $When p_g(x) \to 1, p_r(x) \to 0, KL(p_g \parallel p_r) \to +\infty.$

Using the vanilla-loss function in eq. 2.7 results in vanishing gradient while training generator and also using alternative loss function (eq. 2.12) will achieve a mode-collapse problem. Modifying the model architecture will not solve such obstacles. Instead, an alternative can be to redesign loss functions as loss-variant GANs improve stability of training GANs.

Wasserstein GAN

Wasserstein GAN [118] uses earth-mover (EM) distance [119] as loss measure to solve the above-mentioned problems. The EM distance is specified as follows

$$W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(X,Y)\sim\gamma} \parallel X - Y \parallel, \tag{2.13}$$

where $\Pi(p_r, p_g)$ is a collection of all joint distributions and p_g and p_r are marginals of $\gamma(X, y)$. EM, in comparison with JS and KL, has the capability to provide a meaningful gradient while training the generator and show the distance even in situations where p_g and p_r are not overlapping. WGAN has a smoother gradient while training generator in comparison with original GAN, although it is hard to control the infimum in eq. 2.13. Therefore Wasserstein distance can be estimated as

$$\max_{w\sim W} \mathbb{E}_{x_{p_r}}[f_w(x)] - \mathbb{E}_{z\sim p_z}[f_w(G(z))], \tag{2.14}$$

where f_w represents discriminator, z is the noise as input to the generator. The discriminator aims to maximise eq. 2.14 by making the optimisation distance closer to Wasserstein distance by using its parameters w. After optimising discriminator, generator aims to minimise Wasserstein distance (eq. 2.13). Loss of the generator is

$$-\min_{G} \mathbb{E}_{z\sim p_z}[f_w(G(z))], \tag{2.15}$$

In contrast with the original GAN, Wasserstein GAN (WGAN) fits Wasserstein distance in discriminator instead of binary classification and does not use sigmoid in the last layer.

WGAN-GP

WGANs do not show exceptional generalisation on deeper models since belonging parameters are localised at -0.01 and 0.01 due to parameter clipping, which will reduce the modelling capability of discriminator significantly. Applying gradient penalty on discriminator to regulate $\parallel f \parallel_L \leq K$ and update loss function of discriminator as follows

$$\mathcal{L}_D = \mathbb{E}_{X_g \sim p_g}[D(X_g)] - \mathbb{E}_{X_r \sim p_r}[D(X_r)] + \lambda \mathbb{E}_{\hat{X} \sim p_{\hat{X}}}[(\| \bigtriangledown_{\hat{X}} D(\hat{X}) \|_2 - 1)] \tag{2.16}$$

where X_g and x_r, are the data selected from distribution of generated p_g and real p_r data. The last term in the above equation is the gradient penalty where $p_{\hat{x}}$ are samples uniformly selected along straight lines between pairs of points sampled from p_r and p_g. WGAN-GP performs in a more stable way during training and trained parameters and have better distribution in comparison to WGAN [120].

Least Square GAN

To solve the vanishing gradient of generator, Least Square GAN (LSGAN) suggests using the least square loss for discriminator rather than sigmoid cross entropy which is originally used in original GAN [121] algorithm. The loss function is stated as

$$\begin{aligned} \min_D \mathcal{L}_D &= \frac{1}{2}\mathbb{E}_{X \sim p_r}[(D(X) - b)^2] + \frac{1}{2}\mathbb{E}_{z \sim p_z}[(D(G(X)) - a)^2], \\ \min_G \mathcal{L}_G &= \frac{1}{2}\mathbb{E}_{z \sim p_z}[(D(G(X)) - c)^2] \end{aligned} \tag{2.17}$$

where a and b are the labels for generated and real samples respectively and c is the label that the generator uses to fool the discriminator. This new approach enables the decision boundary of the discriminator to penalise large errors obtained by processing generated samples to help to push them towards the decision boundary. Also the idea of penalising the samples that are far away from the decision boundary can offer better gradient while updating generator and solving the vanishing gradient issue.

f-GAN

The purpose of f-GAN is to train the model by applying f-divergence [122]. The idea is to use probability distributions and calculate the difference $D_f(P \parallel Q)$ between P and Q.

Unrolled GAN

This approach is suggested to solve the mode-collapse problem while training. The idea is to add a gradient term to update the generator based on the way the discriminator will respond [123]. Assuming an iterative procedure to find optimal parameters of D, these parameters can be declared as fixed points.

$$\begin{aligned} \theta_D^0 &= \theta_D, \\ \theta_D^{k+1} &= \theta_D^k + \eta^k \frac{df(\theta_G, \theta_D^k)}{d\theta_D^k}) \\ \theta_D^*(\theta_G) &= \lim_{k\to\infty} \theta_D^k, \end{aligned} \tag{2.18}$$

where θ_D and θ_G serve as parameters of discriminator and generator respectively. The η^k represents the learning rate. The new loss function can be expressed as follows, by unrolling for K a number of steps

$$f_K(\theta_G, \theta_D) = f(\theta_G, \theta_D^K(\theta_G, \theta_D)). \tag{2.19}$$

The above replacement for loss can be used to update parameters of generator and discriminator

$$\begin{aligned} \theta_G &\leftarrow \theta_G - \eta \frac{df_K(\theta_G, \theta_D)}{d\theta_G}, \\ \theta_D &\leftarrow \theta_D + \eta \frac{df(\theta_G, \theta_D)}{d\theta_D} \end{aligned} \tag{2.20}$$

The following term states the gradient to update generator

$$\frac{df_k(\theta_G, \theta_D)}{d\theta_G} = \frac{\partial f(\theta_G, \theta_D^K(\theta_G, \theta_D))}{\theta_G} + \frac{\partial f(\theta_G, \theta_D^K(\theta_G, \theta_D))}{\partial \theta_D^K(\theta_G, \theta_D)} \frac{d\theta_D^K(\theta_G, \theta_D)}{d\theta_G} \tag{2.21}$$

where the first term is the gradient of the vanilla GAN, the second term is the reaction of discriminator while generator changes - meaning while generator tends to collapse to one mode, discriminator increases the loss of generator.

Loss Sensitive GAN

The aim of this type of GAN is to minimise the designated margins between generated and real samples, resulting in generating close-to-reality samples. The Loss-Sensitive GAN (LS-GAN) paper [124] claims that a non-parametric hypothesis, which enables the discriminator to make decisions between generated and real samples, is the reason for mode collapse and vanishing gradient. The ability of the discriminator to classify is limited in LS-GAN. It is learned by loss function $L_\theta(x)$ with parameters θ, assuming that real samples have lower loss that generated ones. The loss function is composed as

$$L_\theta(x) \leq L_\theta(G(x)) - \Delta(x, G(z)), \tag{2.22}$$

where $\Delta(x, G(z))$ is the difference between generated and real samples' margin measuring. The above equation (2.22) implies that the real and generated samples are separated leastwise by a margin of $\Delta(x, G(z))$. The LS-GAN optimisation is

$$\begin{aligned} &\min_D \mathcal{L}_D = \mathbb{E}_{x\sim p_r} L_\theta(x) + \lambda \mathbb{E}_{\substack{z\sim p_z \\ x\sim p_r}} (\Delta(x, G(z)) + L_\theta(x) - L_\theta(G(z)), \\ &\min_G \mathcal{L}_G = \mathbb{E}_{z\sim p_z} L_\theta(G(z)), \end{aligned} \tag{2.23}$$

where θ represents the parameters of discriminator and λ is a positive-balancing parameter. In order to avoid over-fitting the generated and real samples in discriminator, the term $\Delta(x, G(z)$ is added to $\mathcal{L}_D$ in the above equation.

Mode Regularised GAN

This type of GAN suggests penalising missing modes in order to increase the chance of solving the mode-collapse challenge [125]. The main idea is, instead of using noise, utilising an encoder in order to produce a latent variable z for generator $(E(x) : x \to z)$. In this method, reconstructing the encoder can add extra knowledge to generator and therefore help the discriminator to easily distinguish between the real and generated samples. The encoder ensures using correspondence between x and z, which means samples generated by the generator cover various modes in the space of x, therefore avoiding the mode-collapse problem. The loss function for this architecture is stated as

$$\begin{aligned}L_G &= -\mathbb{E}_z[\log[D(G(z))]] + \mathbb{E}_{x\sim p_r}[\lambda_1 d(x, G\circ E(x)) + \lambda_2 \log[D(G(x))]],\\ L_E &= \mathbb{E}_{x\sim p_r}[\lambda_1 d(x, G\circ E(x)) + \lambda_2 \log[D(G(x))]],\end{aligned} \tag{2.24}$$

where d is a geometric measurement that can be selected from various options such as distance of features, for example Euclidean norm.

Geometric GAN

This architecture uses SVM separating hyper-plane which helps to avoid mode collapse and achieve more stable training. In this method between two classes we have high marginals [126].

Relativistic GAN

The aim of Relativistic GAN (RGAN) is to suggest a new cost function by modifying the existing one, for instance using Integral Probability Metric (IPM) GANs [127, 128, 129]. In RGAN, discriminator considers how the real sample is more realistic than generated samples. The new loss function is as follows

$$\begin{aligned}&\min_D \mathbb{E}_{(x_r,x_g)\sim(p_r,p_g)}[\log(sigmoid(C(x_r) - C(x_g)))],\\ &\min_G \mathbb{E}_{(x_r,x_g)\sim(p_r,p_g)}[\log(sigmoid(C(x_g) - C(x_r)))],\end{aligned} \tag{2.25}$$

where $C(x)$ is the layer which is non-transformed. If the loss function belongs to IPMs, RGAN can be generalised to alternative kinds of GANs.The generalisation of this architecture can be stated as

$$\begin{aligned}L_D &= \mathbb{E}(x_r, x_g) \sim (p_r, p_g)[f_1(C(x_r) - C(x_g))] + \mathbb{E}(x_r, x_g) \sim (p_r, p_g)[f_2(C(x_g) - C(x_r))],\\ L_G &= \mathbb{E}(x_r, x_g) \sim (p_r, p_g)[g_1(C(x_r) - C(x_g))] + \mathbb{E}(x_r, x_g) \sim (p_r, p_g)[g_2(C(x_g) - C(x_r))],\end{aligned} \tag{2.26}$$

where $g_1(y) = f_2(y) = y$ and $g_2(y) = f_1(y) = -y$.

Spectral Normalisation GAN

The idea is to normalise weights in order to stabilise the process of training discriminator. This method has the advantage of being easily applicable to already implemented GANs and is computationally light. It has been stressed that discriminator should be from the set of K-Lipshitz continuous functions [120, 118, 124], which means the function does not change abruptly [130, 131, 132]. This gentle behaviour of discriminator stabilises the training process. The goal of this implementation is to apply spectral normalisation of each layer of discriminator to control the Lipschitz constant. The spectral normalisation is stated as

$$\bar{W_{SN}}(W) = \frac{W}{\sigma(W)}, \tag{2.27}$$

where W stands for weights belonging to each layer of discriminator and $\sigma(W)$ represents the weights Euclidean norm matrix.

2.7 Real-world Applications

There are many processes that are using DL in their data analysis. Some examples of those that widely use it are:

- Speech recognition [133].
- Image processing such as handwriting classification [134].
- High-resolution remote-sensing scene classification [135].
- Single image super-resolution [136].
- Multi-category rapid serial visual presentation Brain Computer Interfaces (BCI) [137].
- Domain adaptation for large-scale sentiment classification [138].
- Multi-task learning for NLP with an enhanced inference robustness [139, 140].
- Enhancement of the diagnostic accuracy of micro-calcifications by evaluating the performance of deep learning method on a large dataset for its discrimination [141].

- Solves the challenges of body pat recognition using DL [142].
- Proposes an approach for a high-level latent and shared feature representation from neuro-imaging modalities using DL for Alzheimer's disease and Mild Cognitive Impairment [143].
- An architecture of DL for automated basal cell carcinoma cancer detection [144].

Researchers applied GANs in diverse contexts. The following describes a summary of the most recent papers along with their applications in transfer learning:

- Text Generation:
  - Textkd-gan [145].
- Image domain transfer:
  - Small Datasets Image generation [146].
  - TTUR - a two time-scale update rule for training GANs with SGD on arbitrary GAN loss functions [147].
  - Progressive Growing of GANS [112].
  - Simultaneous Deep Transfer Across Domains and Tasks [148].
  - Adversarial Discriminative Domain Adaptation (ADDA) [149].
  - Transferring GANs: Generating images from limited data [150].
  - Unsupervised Pixel-Level Domain Adaptation With Generative Adversarial Networks [151].
  - cGAN [152], proposes a model for the discriminator of cGANs.
- Video Processing:
  - FutureGAN [153].
- HALI [154], A generative model that improves training stability and simplicity.
- Mind2Mind [155], A method to directly train models' external layers against each other and bypass all the intermediate layers.

- Bi-directional Generative Adversarial Networks (BiGANs). [156] suggests a mean to project data back into the latent space.
- Domain-Adversarial Neural Networks (DANNs) [157], proves domain discriminability is another principle that is complimentary to robustness and can improve cross-domain adaptation.
- Conditional Domain Adversarial Networks (CDANs), suggests novel approaches to domain adaptation with multi-modal distributions [158].
- Label-efficient learning of transferable representations across domains and tasks. [159] proposed a method to learn a representation that is transferable across different domains and tasks in a data-efficient manner.

2.7.1 Related Work on Machine Learning for ECGs

In the field of cardiac arrhythmias, plenty of machine learning-based techniques is reported in the literature. These approaches use algorithms such as ANNs [160, 161, 162, 163, 164, 165], Decision Trees [166], Support Vector Machine (SVM) [167, 168, 169, 170], Deep Learning [171, 172, 173], Fuzzy Systems [174], Deep Belief Networks [175], Linear Discriminant classifiers (LDC) [176], and Probabilistic Neural Network (PNN) and Radial Basis Function Neural Network (RBF-NN) [177]. In specific cases, a combination of swarm intelligence algorithms and neural networks could achieve competitive results [178].

2.8 Chapter Summary

This chapter offered a review of current technologies in the field of Deep Learning and Generative Adversarial Networks, their range of types, loss functions and real-world applications. The next chapter covers multiple nature-inspired algorithms and their applications in optimisation challenges. In addition, a detailed analysis of the Dispersive Flies Optimisation (DFO) is provided. We used this algorithm in chapters 4 and 5 for hybrid model experiments. The last section of this chapter gives an overview of the gradient-free methods.

3. Swarm Intelligence for Machine Learning

3.1 Introduction

In the previous chapter we looked at the gradient-based approaches used in Machine Learning algorithms to discover the optimal parameters. However, research shows the gradient-free methods could be promising solutions [179] if a problem can be formulated as an optimisation challenge. As such, this chapter discusses a number of swarm intelligence algorithms, and focuses on Dispersive Flies Optimisation (DFO), one of the methods specifically selected for the experiments in this thesis.

3.2 Population Based Algorithms

Nature has always been a rich source of inspiration for scientists. Observations have been triggering curious minds for centuries, leading to discoveries and breakthroughs in medicine, physics, astronomy and biology among many other fields. More recently, researchers working in the field of computer science and machine learning have likewise drawn inspiration from natural phenomena. From the choreographed movements of birds, behaviours of foraging ants, to the convergence of honey bees while searching for food, these manifestations of nature in action have inspired novel developments in machine learning, offering up the potential to formulate algorithms to solve different optimisation problems.

Genetic Algorithm [180], Particle Swarm Optimisation [181] and Ant Colony

Optimisation [182] techniques belong to the broader category of Swarm Intelligence (SI). SI investigates collective intelligence and intends to model it by considering individuals in a social context and observing interactions among others and with their surrounding environment.

The following sections provide an overview of some of the existing SI techniques. The algorithms introduced briefly in this section are variations of Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), Differential Evolution Algorithm (DE), Genetic Algorithm (GA) and Dispersive Flies Optimisation (DFO). As previously mentioned, the interactions among their population is one of the characteristic features of the algorithms providing possibilities to achieve the final aim of finding the optimal parameters.

3.2.1 Ant Colony Optimisation

The Ant Colony Optimisation (ACO) algorithm includes several steps. It is based on a structure of dynamic memory which holds information about the quality of previously obtained results [183, 184]. Each ant represents a candidate for the solution to the problem. This algorithm includes a forward mode that enables ants to construct their answers based on existing pheromone trails and heuristic information used from the most recent generation. Ants switch to the backward mode and update the shared pheromone table accordingly as soon as their forward mode is complete. For instance, the better the quality of solution, the more the pheromones are kept. ACO has two primary frameworks: evaporation-based [185, 186] and population-based [187]. The approaches that update the pheromones are different for each of these frameworks. The evaporation-based framework applies gradual pheromone trails reduction to eliminate prior poor decisions.

3.2.2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO), developed by Kennedy and Eberhart [181], is a type of population-based optimisation algorithm. PSO is inspired by simulating the choreography of birds flying or fish schooling in coordinated flocks and shoals. Their behaviour shows robust synchronisation in flying, landing, and changes in direction while hovering, although researchers were not able to find leaders in such crowds [188]. The PSO algorithm includes a swarm of many particles. Each particle signifies a point in a multi-dimensional space.

Each particle has a position $\vec{x}$ and personal best $\vec{p}$ attributes. The personal best keeps the best position during optimisation. The neighbourhood best is the best-found position within the whole population or local neighbourhood. Note that the $\vec{x}$ is related to the particles' neighbourhood and experience of the particle.

Clerc-Kennedy PSO (PSO-CK) or constriction PSO is a standard particle-swarm version. The position of each particle is defined by the combination of velocity and its current position. Here is the updating equation for the velocity and position of each particle:

$$v_{id}^{t} = \chi\left(v_{id}^{t-1} + c_1 r_1\left(p_{id} - x_{id}^{t-1}\right) + c_2 r_2\left(g_{id} - x_{id}^{t-1}\right)\right) \tag{3.28}$$

$$x_{id}^{t} = v_{id}^{t} + x_{id}^{t-1} \tag{3.29}$$

where χ, the constriction factor, is set to 0.72984, which is proven to work well in most cases [189]. The v_{id}^{t-1} is the velocity of particle i in dimension d at time step $t-1$. The learning factors or acceleration constants related to personal and neighbourhood are represented as $c_{1,2}$ respectively. Both of these parameters are set as constant. The $r_{1,2}$ are random values used for the purpose of adding stochasticity. These numbers are drawn from a uniform distribution on the unit interval $U(0,1)$. The p_{id} is the personal best position of particle x_i in dimension d. Lastly, g_{id} represented as neighbourhood best.
In terms of PSO application on medical datasets, [190] proposed a method which works based on Chan and Vese's algorithm, which helps to achieve acceptable segmentation performance, regardless of the early choice of the contour. The [191] aimed to determine the possibility of using a radial-basis function neural network based on PSO that is capable of identifying that Parkinsonian tremors are occurring from local field potential signals. [192] presented an approach for human tremor analysis by applying PSO. This study addressed Parkinson's disease and essential tremor.

3.2.3 Differential Evolution Algorithm

Differential Evolution (DE), first introduced by Storn and Price [193] is a kind of evolutionary algorithm (EA). The DE is a simple global numerical optimiser over continuous search spaces. This algorithm aims to search the space to find the optimal parameter. The following definition defines the parameter vectors of the population of this population-based stochastic algorithm:

$$\vec{x}_i^g = \left[x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g\right], i = 1, 2, ..., NP \tag{3.30}$$

where g, D, and NP are the current generation, the problem space dimension, and size of the population respectively. In the first generation, while $g = 0$, the i^{th} vector's j^{th} components are initialised as follows:

$$x_{i,j}^0 = x_{min,d} + r\left(x_{max,d} - x_{min,d}\right) \tag{3.31}$$

where r is a random number picked from a uniform distribution of the unit interval $U(0,1)$, and x_{min}, and x_{max} are the lower and upper bounds of the d^{th} dimension respectively. Evolutionary processes such as mutation, crossover and selection, start after the initialisation of the population.

Mutation

The mutation operation is applied to the target vector x_i^g over every generation g, which will result in the corresponding vector v_i^g (mutant vector). *DE/best/1* variation of mutation approaches is as follows:

$$v_i^g = x_{best}^g + F\left(x_{r_1}^g - x_{r_2}^g\right) \tag{3.32}$$

where r_1 and r_2 are distinct random integers selected from $[1, NP]$; in generation g, x_{best}^g and $F = 0.5$ are the vectors with the best fitness value and a positive control parameter for constricting the different vectors respectively.

Crossover

Crossover is an operation to improve the population diversity by exchanging a few components of v_i^g mutant vector with x_i^g target vector to generate u_i^g trial vector. This process is as follows:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } \ r \leq CR \ \text{ or } j = r_d \\ \\ x_{i,j}^g, & \text{otherwise} \end{cases} \tag{3.33}$$

where r is a uniformly-distributed random number picked from range $U(0,1)$ and r_d is a randomly-generated integer from the range $[1, D]$, which assures that at least one component of the trial vector differs from the target vector. The $CR = 0.5$ is also a control parameter which specifies the level of inheritance from v_i^g.

Selection

The selection process decides if the target or trial vectors, x_i^g or u_i^g, are allowed to pass to the next generation $g+1$. In a scenario with the aim of a solution of minimalisation, the vector with the smaller fitness value is admitted to the next generation:

$$x_i^{g+1} = \begin{cases} u_i^g, & \text{if } \ f(u_i^g) \leq f(x_i^g) \\ \\ x_i^g, & \text{otherwise} \end{cases} \tag{3.34}$$

where $f(x)$ is the fitness function.

3.2.4 Genetic Algorithm

In Genetic Algorithm (GA) [194, 195], the individuals are initialised randomly and an object function is used to evaluate their fitness. Individuals have the probability of being exposed to recombination p_c or mutation p_m in an iterative process. Arithmetic crossover is used as recombination operator. The mutation operator used is Cauchy mutation utilising an annealing scheme. Tournament selection [196] is used to comb out the least fit individual.

The main reason why GA uses the Cauchy mutation operator instead of the

Gaussian mutation operator is that the thick tails of the Cauchy distribution allow it to generate considerable changes and with greater frequency in comparison with the Gaussian distribution. The Cauchy distribution is defined as follows:

$$C\left(x, \alpha, \beta\right) = \frac{1}{\beta\pi\left(1 + \left(\frac{x-\alpha}{\beta}\right)^2\right)} \tag{3.35}$$

where $\alpha \leq 0$, $\beta > 0$, $-\infty < x < \infty$. α and β are parameters that influence the distribution mean and spread. As [195] suggested, all the solution parameters are subject to mutation and the variance is scaled with $0.1 \times$ the range of the specific parameter in question. An annealing scheme was applied to decrease the value β as a function of the elapsed number of generations t while $\alpha = 0$:

$$\beta\left(t\right) = \frac{1}{1+t} \tag{3.36}$$

As for the arithmetic crossover, the offspring is generated as a weighted mean for each gene of the two parents:

$$\text{offspring}_i = r \times \text{parent1}_i + \left(1 - r\right) \times \text{parent2}_i \tag{3.37}$$

where offspring_i is the i'th gene of the offspring, and parent1_i and parent2_i refer to the i'th gene of the two parents, respectively. The weight r is drawn from a uniform distribution in the unit interval $U\left(0, 1\right)$.

In the above-mentioned experiment, the probabilities of crossover and mutation of the individuals is set to $p_c = 0.7$ and $p_m = 0.9$ respectively. The size of the tournament's selection is set to two, and the elitism, with an elite size of one, is deployed to maintain the best-found solution in the population.
Regarding the application of genetic programming in medical data analysis, [197] presented a majority-voting GP classifier for micro-array data classification. [198] proposed a new technique by utilising the feature generated by GP to diagnose breast cancer. [199] developed a detection approach for nodal metastasis from molecular profiles of primary urothelial carcinoma tissues. Samples were run through the GP, which utilises the N-fold cross-validation method to produce classifier instructions of limited complexity.

3.2.5 Dispersive Flies Optimisation

Dispersive flies optimisation (DFO) [200] belongs to the broad family of population-based, swarm intelligence optimisers, which has been applied to various areas, including medical imaging [201], diophantine equations [202], PID speed control of DC motor [203], optimising machine learning algorithms [204, 205], training deep neural networks [206], computer vision and quantifying symmetrical complexities [207, 208], beer organoleptic optimisation [209, 210], and analysis of autopoiesis in computational creativity [211]. DFO is a minimalist, vector-stripped swarm optimiser [212] whose exploration-exploitation balance and zone analyses has also been studied in [213, 214].

Explaining the algorithm

Dispersive Flies Optimisation (DFO) [200] is an algorithm inspired by the swarming behaviour of flies hovering over food sources. The swarming behaviour of flies is determined by several factors and the presence of a threat can disturb their convergence on the marker (or the optimum value). Therefore, having considered the formation of the swarms over the marker, the breaking or weakening of the swarms is also noted in the proposed algorithm.

The swarming behaviour of the individuals in DFO consists of two tightly connected mechanisms. One is the formation of the swarms and the other is its breaking or weakening. The position vector of a fly is defined as:

$$\vec{x}_i^t = \left[x_{i0}^t, x_{i1}^t, ..., x_{iD-1}^t\right], \qquad i = 0, 1, ..., N\text{-}1 \tag{3.38}$$

where i represents the i^{th} individual, t is the current time step, D is the dimensionality of the problem space and N is the population size. For continuous problems, $x_{id} \in \mathbb{R}$ (or a subset of $\mathbb{R}$), and in the discrete cases, $x_{id} \in \mathbb{Z}$ (or a subset of $\mathbb{Z}$).

In the first iteration, $t = 0$, the i^{th} vector's d^{th} component is initialised as:

$$x_{id}^0 = x_{\min,d} + u\left(x_{\max,d} - x_{\min,d}\right) \tag{3.39}$$

where $u \sim \mathrm{U}(0, 1)$ is the uniform distribution between 0 and 1; $x_{\min,d}$ and $x_{\max,d}$

Algorithm 1 Dispersive Flies Optimisation

```
1: procedure DFO ($N, D, \vec{x}_{\min}, \vec{x}_{\max}, f$)*
2:     for $i = 0 \to N-1$ do                                  ▷ Initialisation
3:         for $d = 0 \to D-1$ do
4:             $x_{id}^0 \leftarrow \mathrm{U}(x_{\min,d}, x_{\max,d})$
5:         end for
6:     end for
7:     while ! termination criteria do                         ▷ Main DFO loop
8:         for $i = 0 \to N-1$ do
9:             $\vec{x}_i$.fitness $\leftarrow f(\vec{x}_i)$
10:        end for
11:        $\vec{x}_s = \arg\min\,[f(\vec{x}_i)], \quad i \in \{0,1,2,\ldots,N-1\}$
12:        for $i = 0 \to N-1$ and $i \neq s$ do                ▷ Update each fly
13:            $\vec{x}_{i_n} = \arg\min\,[f(\vec{x}_{(i-1)\%N}), f(\vec{x}_{(i+1)\%N})]$
14:            for $d = 0 \to D-1$ do
15:                if $\mathrm{U}(0,1) < \Delta$ then             ▷ Using Disturbance Threshold
16:                    $x_{id}^{t+1} \leftarrow \mathrm{U}(x_{\min,d}, x_{\max,d})$
17:                else
18:                    $u \leftarrow U(0,1)$
19:                    $x_{id}^{t+1} \leftarrow x_{i_nd}^t + u(x_{sd}^t - x_{id}^t)$    ▷ Update equation
20:                end if
21:            end for
22:        end for
23:    end while
24:    return $\vec{x}_s$
25: end procedure
```

* INPUT: N: swarm size, D: dimensions, $\vec{x}_{\min}$: lower bound, $\vec{x}_{\max}$: upper bound, f: fitness function.

are the lower and upper initialisation bounds of the d^{th} dimension, respectively.

On each iteration, components of the position vectors are independently updated, taking into account:

- current fly's position
- current fly's best neighbouring individual (consider ring topology)
- best fly in the swarm

Therefore, the update equation is

$$x_{id}^{t+1} = x_{i_nd}^t + u(x_{sd}^t - x_{id}^t) \tag{3.40}$$

where $x_{i_nd}^t$ is the position value of $\vec{x}_i^{\,t}$'s best *neighbouring* individual in the d^{th} dimension at time step t, and x_{sd}^t is the value of the *swarm*'s best individual in the d^{th} dimension at time step t.

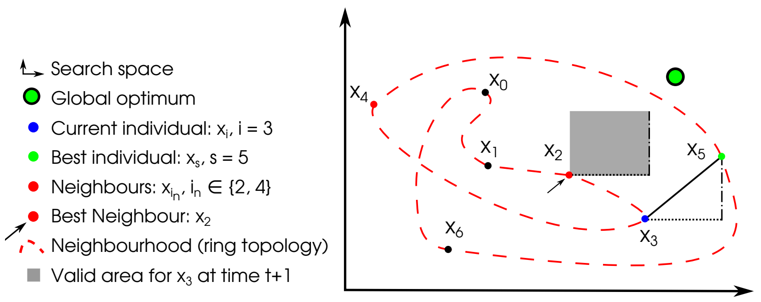
Search space
Global optimum
Current individual: x_i, i = 3
Best individual: x_s, s = 5
Neighbours: x_{i_n}, $i_n \in \{2, 4\}$
Best Neighbour: x_2
Neighbourhood (ring topology)
Valid area for x_3 at time t+1
x_0
x_1
x_2
x_3
x_4
x_5
x_6

Figure 3.3: *Sample update of x_i, where $i = 3$ in a 2D space where the axes are the dimensions (d_1 and d_2) in the 2D search space [2]. Used with permission.*

The algorithm is characterised by two main components: a dynamic rule for updating the population's position (assisted by a social-neighbouring network that informs this update), and communication of the results of the best-found individual to others.

As stated earlier, the swarm is disturbed for various reasons; one of the impacts of such disturbances is the displacement of the individuals, which may in turn lead to discovering better positions. To consider this eventuality, an element of stochasticity is introduced to the update process. Based on this, individual components of the population's position vectors are reset if a random number generated from a uniform distribution on the unit interval $U(0, 1)$ is less than the *disturbance threshold*, Δ. This guarantees a disturbance to the otherwise permanent stagnation over a likely local minima.

Algorithm 1 summarises the DFO algorithm. Note that every time u is called, it generates a new random number between 0 and 1[1]. In this algorithm, each member of the population is assumed to have two neighbours (i.e. ring topology).

3.3 Real-world applications

Researchers applied SI algorithms in diverse contexts. The following describes a summary of the most recent papers along with their application:

- Food Industry
  - Product selection in the beer-brewing process using swarm intelligence [215].

[1]The source code and some relevant resources can be downloaded from the following pages: https://github.com/mohmaj/DFO, http://doc.gold.ac.uk/mohammad/DFO/

- Data Mining
  - The possibility of using a group of incremental classification algorithms for classifying the collected data streams pertaining to the Big Data investigated [216].
  - The applications of SI in the big data analytic and the big data analytic techniques in SI are analysed [217].
- Medical Imaging
  - Investigates particle swarm optimisation (PSO), Darwinian PSO and Fractional Order Darwinian PSO to speed up the algorithm in medical imaging applications concerned with volume reconstruction [218].
  - Discusses the application of Stochastic Diffusion Search in detecting areas of metastasis in bone scans and the identification of potential micro-calcifications on the mammographs [219].
- Drug Design and Pharmaceuticals
  - Reviews the chemical engineering applications of Multi-objective Optimisation (MOO). [220].
  - Various EA techniques used in *denovo* drug design tools are surveyed and analysed in detail, with particular emphasis on the computational aspects. [221].
- Image Processing
  - Proposes a new approach based on features of Genetic Algorithms for gray-scale medical image protection [222].
  - Suggests a multi-level thresholding segmentation method for grouping the pixels of multi-spectral and hyper-spectral images into different homogeneous regions [223].
- Protein Folding
  - Presents a genetic algorithm applied to the protein structure prediction in a hydrophobic-polar model on a cubic lattice. The proposed genetic

algorithm is extended with crowding, clustering, repair, local search and opposition-based mechanisms [224].

  - A simplified three-dimensional protein model was used in order to allow for the fast development of a robust and efficient genetic algorithm-based methodology. [225]

- Molecular Dynamics
  - A quantum-classical algorithm for locating the global minimum on the potential energy surface of a large molecule and simultaneously predicting the associated electronic charge distribution is developed by interfacing classical PSO with a near-optimal unitary evolution scheme for the trial of one-electron density matrix. [226]
  - Presents a newly-developed publicly available genetic algorithm (GA) for global-structure optimisation within atomic-scale modelling [227].

- Weather prediction
  - An approach is proposed that builds an efficient and effective model for heavy rain forecasting 6 hours ahead, based on past weather data [228].
  - Presents a multi-objective optimisation model using Genetic Algorithm (GA) and Artificial Neural Network (ANN) to quantitatively assess technology choices in a building retrofit project [229].

- Structural Optimisation
  - In the empirical methods for reinforcement design of underground excavations, an even distribution of rock bolts is generally recommended. The work represented in [230] proves that this design is not necessarily optimal and shows how state-of-the-art reinforcement design could be improved through topology optimisation techniques.

- Tomography
  - Proposes an innovative Inverse Scattering (IS) technique for the simultaneous processing of multi-frequency (MF) ground-penetrating radar (GPR) measurements [231].

  - Presents a method for reconstruction of EIT images based on FSS and Non-Blind Search (NBS) [232].
  - Uses two different multi-objective particle swarm optimisation approaches to jointly invert synthetic cross-hole tomographic data sets comprising radar and P-wave travel-times. [233].

- Robotics
  - A survey to illustrate various algorithms that have been used to tackle the challenges of imposed swarm-robotics tasks [234].
  - Presents an extensive compilation of original articles on the cross-fertilisation between ER and other research areas [235].

- Computational Fluid Dynamics
  - Proposes a multi-objective optimisation methodology using a stochastic optimisation algorithm, a Genetic Algorithm (GA) with ϵ-constraint method, and a 2D axi-symmetric Computational Fluid Dynamics (CFD)-based Fischer-Tropsch micro-channel reactor model [236].
  - Presents an optimisation method suitable for improving the performance of Archimedes screw axial rotary blood pumps. [237].

- Space applications
  - Presents different combinations of geometrical dimensions of a rectangular space radiator that have been estimated using an inverse method [238].
  - Designs an innovative ground-based automated planning and scheduling system for multiple platforms [239].

- Financial Markets
  - Proposes an integrated moving average rule for the European Union Allowance (EUA) futures market and designs an approach to optimise the weights of rules based on PSO and GAs [240].

  - Provides a meta-survey on state-of-the-art research and reports in the literature of the field [241].

- Reservoir Optimisation in oil fields
  - Applied and evaluated state-of-the-art, adaptive differential algorithms (SHADE and jDE), and non-adaptive evolutionary algorithms (standard DE, PSO) that have been tuned using standard black-box benchmark functions as training instances [242].
  - Presents a multi-objective method with robust optimisation methodology by incorporating three dedicated objective functions [243].

- Energy Systems
  - In this work, a mono- and multi-objective Particle Swarm Optimisation (MOPSO) algorithm is coupled with EnergyPlus building energy simulation software to find a set of non-dominated solutions to enhance building energy performance [244].
  - Formulates an optimal power-flow problem by considering controllable and uncontrollable distributed generators in power networks [245].

- Engineering Design
  - Presents an effective hybrid cuckoo search and genetic algorithm (HCSGA) for solving engineering design optimisation problems involving problem-specific constraints and mixed variables such as integer, discrete and continuous variables [246].
  - Aims to solve structural engineering design optimisation problems with non-linear resource constraints [247].

- Manufacturing Sciences
  - Presents batik production process optimisation using Particle Swarm Optimisation (PSO) methods [248].
  - Develops a novel Hybrid Optimisation Method (HRABC), based on artificial bee colony algorithm and the Taguchi method [249].

- Scheduling
  - Proposes an approach to address the dynamic-scheduling problem reducing energy consumption and make-span for flexible flow-shop scheduling [250].
  - Introduces the objective of minimising energy consumption into a typical production scheduling model, for instance the job-shop scheduling problems, based on a machine-speed scaling framework [251].
- Vehicle Routing
  - Presents a heterogeneous vehicle-routing problem used at a carton collection depot, which can collaboratively pick the cartons up from several carton factories to a collection depot and then from there to serve their corresponding customers by using a heterogeneous fleet [252]
  - Presents a survey of genetic algorithms that are designed for solving multi-depot vehicle-routing problems [253]
- Micro Electro-Mechanical Systems
  - Experimentally demonstrates light focusing through ZnO sample by controlling binary amplitude optimisation using Genetic Algorithm [254].
  - An investigation of non-linear probe behaviour in an atomic force microscope, caused by different excitation frequencies, was carried out as well as an analysis and subsequent regulation using Particle Swarm Optimisation in combination with proportional-derivative control [255].
- Railway applications
  - Provides a comprehensive review regarding the applications of Particle Swarm Optimisation (PSO) in the railway domain [256].
  - Studies the optimisation approach for the speed trajectory of a high-speed train in a single section of track [257].

3.4 Gradient-free Algorithms

Gradient-based algorithms such as backpropagation are typically used to train deep artificial neural networks (DNNs). Evolution strategies (ES) is a competitor of backpropagation-based algorithms, for instance policy gradients and Q-learning [258] on challenging deep reinforcement learning (RL) problems [259]. Since ES performs stochastic gradient descent through a similar operation to a finite-difference approximation of the gradient, it can be regarded as a gradient-based algorithm. Recent researchers have suggested that non-gradient-based evolutionary algorithms can perform on DNNs. For instance, [259] obtained the DNN weights with a gradient-free and population-based Genetic algorithm (GA) which performs admirably on difficult deep-reinforcement learning problems. The problems discussed in this paper were in the context of Atari game-playing and humanoid locomotion. In their proposed implementation, big Deep GA networks including over four million parameters are optimised. This paper suggests that gradient-based algorithms are not the best option in all cases for tuning performance.

Familiarising oneself with the SI algorithm to optimise the ANNs includes some necessary steps regardless of which SI Algorithm is utilised. Since the hidden layer and output layer weights require optimisation, they need to be traced as the vital entity contingent on the SI approach. The problem area holds the synthesis of all possible weight values for all layers. The search space with n-dimensions where n is the collection of weights that need to be updated and optimised. The SI approach is implemented, and the target function is dependent on the projection accuracy of ANN. The weights are mapped onto the required procedure objective, like a particle position in PSO. While calculating the fitness in SI, the weights are allocated to the ANN, and its prophecy and accuracy are obtained. If the fitness is the finest to date, then it will be recorded as the finest set of weights and thus the finest result to date according to the SI Algorithm. The stages of an ANN optimisation using SI algorithms are stated in Algorithm 2 and summarised in Figure 3.4. In terms of the last work in the context SI-ML, [260] proposed a technique that implements a swarm intelligence technique, Stochastic Diffusion Search (SDS), to find the optimal feature subset. The work of [205] investigated

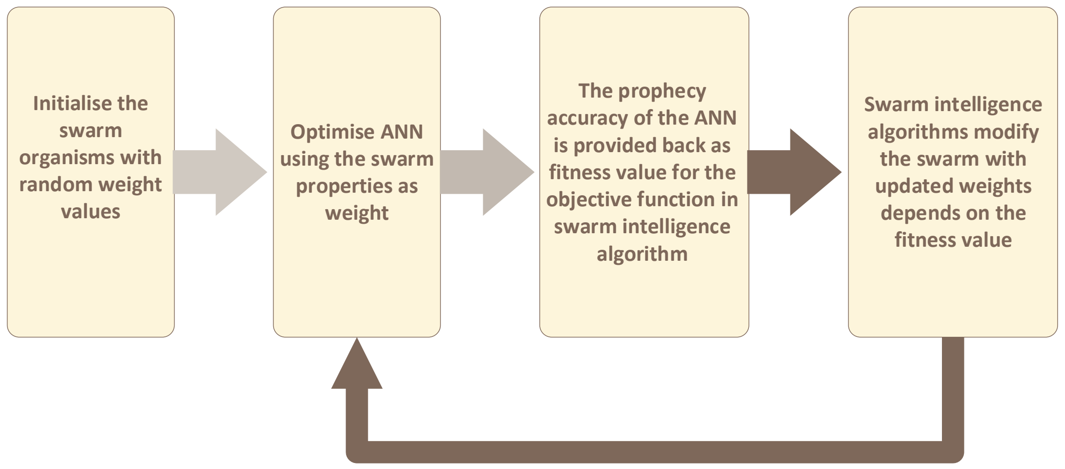
Initialise the swarm organisms with random weight values
Optimise ANN using the swarm properties as weight
The prophecy accuracy of the ANN is provided back as fitness value for the objective function in swarm intelligence algorithm
Swarm intelligence algorithms modify the swarm with updated weights depends on the fitness value

Figure 3.4: *Hybrid Artificial Neural Network-Swarm Intelligence Model* (From [264])

the use of DFO to optimise the RBF kernel's parameters to improve classifier performance without changing the distribution of the dataset by applying data-level solutions such as oversampling or undersampling the dataset. [261] proposes a model which uses a swarm intelligence algorithm (Stochastic Diffusion Search or SDS) to perform the undersampling of the majority classes in imbalanced datasets. [262] proposes a swarm intelligence-based undersampling approach that reduces the sizes of the majority class in a reliable, yet cheap computational way, using the agents and partial evaluation of the majority instance, in which the individuals of the swarm move through the solution space in search of the solution closest to the model. [263] proposes a new approach to addressing imbalanced data by using a combination of both data-level and algorithmic-level solutions.

Algorithm 2 ANN Swarm Intelligence Procedure

```
 1: Define the ANN architecture - number of input, hidden and output neurons
 2: Identify the fitness function which returns the error as difference of actual and
 3:    predicted output for the ANN.
 4: Initiate a swarm of ‘x’ organisms with random weights of ‘n’ dimension where
    n is
 5:    the total number of weights.
 6: while required prediction accuracy is not obtained do
 7:     Find the fitness of each organism as defined in Step 2.
 8:     Update the best solution so far next iteration.
 9:     Update algorithm parameters for next iteration.
10: end while
```

Deep Neuroevolution

Evolution Strategies (ES) are procedures of heuristic search inspired by natural evolution. ES is a category of algorithms for black-box optimisation [265, 266]. In each iteration, the algorithm calculates and evaluates the objective function value by using a perturbed population of parameter vectors. The parameter vectors with the highest scores are utilised to prepare the population for the next iteration. The algorithm repeats this process until it reaches the specified target fitness or is fully optimised. Population representation, parameter mutation and update differ depending on algorithm class [267]. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is the most widely known in the ES class [268]. It utilises full-covariance multivariate Gaussian distribution to represent the population. CMA-ES achieves remarkable results in resolving optimisation problems in low to medium dimensions.

ES is flexible enough to be expanded to execute agents in parallel. In each iteration, agents only communicate at the end of iterations by returning a single scalar, which is their calculated fitness value and receive a parameter update. In contrast with the policy gradient methods that agents require to exchange entire gradients, ES requires exceedingly low bandwidth [267].

ES does not require value function approximations, whereas Reinforcement Learning (RL) requires multiple-function value updates upon a given policy improvement. Following any significant change in policy, multiple iterations are necessary for value function to compensate.

In [267], the authors used a version of ES which belongs to a class of Natural Evolution Strategies (NES) [269, 270, 271, 272, 273, 274, 275], which is closely related to the work of [276]. Let Θ denote parameters and F as the objective function. NES algorithms represent the population with a distribution over parameters $p_\psi(\Theta)$. NES aims to maximize the average objective value $\mathbb{E}_{\Theta\sim p_\psi}F(\Theta)$ over the population with stochastic gradient ascent and uses the following estimator to take gradient steps on ψ:

$$\nabla_\psi \mathbb{E}_{\Theta\sim p_\psi}F(\Theta) = \mathbb{E}_{\Theta\sim p_\psi}\{F(\Theta)\nabla_\psi log p_\psi(\Theta)\} \tag{3.41}$$

Considering the fact that p_ψ is factored Gaussian, the resulting gradient estimator is also known as parameter-exploring policy gradients [276], zero-order gradient estimation [277] or [278]. Algorithm 3 explains the final implementation of an ES for training a model [267].

Algorithm 3 Evolution Strategies

```
    Input:
1: for t=0,1,2,... do
2:     Sample $\epsilon_1, ...\epsilon_n \sim \mathcal{N}(0, I)$
3:     Compute returns $F_i = F(\Theta_t + \sigma\epsilon_i)$ for i=1,...,n
4:     Set $\Theta_{t+1} \leftarrow \Theta_t + \alpha\frac{1}{n\sigma}\sum_{n=1}^{n} F_i\epsilon_i$.
5: end for
```

The following researchers suggest the application of hybrid methods on medical datasets: The GA-PSO hybrid method was proposed for use with gene selection [279], Genetic algorithm and multilayer perceptron (GA-MLP), an enhanced GA technique which works based on the theory of “most highly fit parents are most likely to produce healthiest offspring” [280]. A new machine learning approach is proposed by utilising the SVM-PSO method and cuckoo search [281, 282] to build a rotation forest (RF) ensemble using 20 learners. Two clinical datasets, namely lymphography and backache are used as benchmarks. A new technique was used to find liver cancer by using the hybrid method of PSO-SVM [283, 284] used the hybrid approach of PSO-SVM for the classification of tumours; [285] developed a classification mode of SVM mitochondrial toxicity using the GA-CG-SVM scheme; [286] formalised a robust gene selection method based on a hybrid among GA and SVM to extract fully their respective merits for recognition of main feature genes and for a complex biological phenotype. Some research has been done to combine DL and EA [287] such that the optimal weights have been discovered using EA and at the last step the model is trained using backpropagation. This study shows an improvement in the results obtained by performing the training in two stages.

In summary, neuro-evolution [288, 289, 290] couples artificial neural networks and evolutionary algorithms. In some cases, such as in evolutionary robotics [291, 292, 293] and artificial life [294, 295], the gradient information is difficult to obtain or not available due to the complexity of the problem and the imbalance between the number of samples and the number of parameters to be optimised. Recent work shows NE can scale to derive the parameters of deep neural networks effectively [267, 259].

Recent work also shows NE can scale to derive the parameters of deep neural networks effectively [267, 259], although it historically served in much smaller networks that were limited to tens or hundreds of parameters. This was due to limited computational resources while dealing with high dimensional data [296]. The difficulty is that DNNs create tension between weight disturbances applied and speed of evolution. In the event that only a few weights are updated in each iteration or generation, it would take a considerable number of generations to tune all the weights. On the other hand, if many weights are updated, it might be too drastic for the search to proceed systematically. Such concerns are addressed in indirect encoding research [297], in which a compact genotype is expanded at evaluation time into a larger neural-network phenotype. Researchers, such as [296] explore perturbation in the space of an NN's outputs rather than only in the space of its parameters. This leads to two approaches to generate safer neural network mutations, namely Safe Mutations through Rescaling (SM-R) and Safe Mutation through Gradients (SM-G). In SM-R, a line search can rescale the magnitude of a raw weight perturbation until it is deemed safe, which does not require the NN to be differentiable (At the expense of several NN forward passes). In SM-G, When the NN is differentiable, the sensitivity of the NN to consistent input patterns can be calculated (at the expense of a backward pass). The assumption underlying these approaches is that domain evaluation is expensive relative to NN evaluation, for instance forward or backward NN propagation. Interestingly, both approaches relate to effective mechanisms from deep learning, such as adaptive-learning rate methods [298] or trust regions [299], although here there is clear motivation and setting, for instance, SM-R and SM-G generate pure variation independent of reward, unlike such deep learning methods.

3.5 Chapter Summary

This chapter addressed the concept of Swarm Intelligence and discussed a number of different algorithms including PSO, DE, GA, DFO. Furthermore, the idea of hybrid models has been discussed and elaborated upon. Multiple real-world examples and implementation is provided for each algorithm. The next chapter explains our proposed deep neuro-evoloution algorithm and compares the results obtained using gradient-based and gradient-free approaches.

4. Deep Neuroevolution for Bio-Signal Classification

4.1 Introduction

This chapter covers the first part of my research, exploring the possibility of using Swarm Intelligence to find the optimal parameters of deep neural networks, known as Deep Neuroevolution, and compare its performance with gradient-based algorithms. The structure of the suggested hybrid algorithm, the selected neural network models, dataset, and metrics used to measure the performance are discussed. The gradient-free approach is implemented using the Dispersive Fly Optimisation (DFO) algorithm. This hybrid model suggests further improvements on the DFO's update equation, a dynamic model to control and update the disturbance threshold (Δ), and an improvement on the gathered results in contrast with the gradient-based algorithms.

4.2 Motivation

The accurate detection of false alarms in medical Intensive Care Units (ICUs) is of unquestionable benefit to both patients and the healthcare system. To clarify, a false alarm in the ICU may result in a range of negative outcomes, such as noise disturbance, disruption of continuity of care, lack of sleep, all of which may impact patients' stress levels, and, more generally, compromise the quality of recuperative care. It is essential to note that only an estimated 2 - 9% of the alarms in the ICU are considered essential.

In this research, we deal mainly with arrhythmias, abnormalities in the heart function which can occur in healthy and unhealthy subjects. The ICU is equipped with monitoring devices capable of detecting dangerous arrhythmias, namely asystole, extreme bradycardia, extreme tachycardia, ventricular tachycardia and ventricular flutter/fibrillation. Arrhythmias are potentially fatal and in line with AAMI guidelines, appropriate responses should be taken within 10 seconds of the event's commencement [5]. Triggering of the alarm when an arrhythmia occurs could improve the chance of saving lives. Misconfigurations, defective wiring, staff manipulation, and patient manipulation or movement may increase the false alarm rate to as much as 86%. Clinically, 6% to 40% of the ICU alarms proved to be lower priority incidents which did not require immediate responses [6]. False alarms stimulate mental discomfort in patients [7] and may desensitise the reactions of clinical staff, causing slower responses to triggered alarms [8]. True alarms which are rated with high priority and require an urgent response make up only 2 - 9% of all ICU alarms [9]; therefore, the detection and elimination of false alarms are important areas for research.

Gradient-based learning algorithms such as backpropagation are used to train Deep artificial Neural Networks (DNNs). As an alternative, this study aims to evaluate the application of Dispersive Flies Optimisation (DFO) in order to find the optimal weights of a given neural network. We evaluated the proposed gradient-free method on a subset of the Physionet Challenge 2015 dataset [3]. The goal of this challenge is to reduce the occurrences of false alarms with accurate detection of the above-mentioned life-threatening arrhythmias. This goal is achieved by using multi-modal input data such as respiration (RESP), arterial blood pressure (ABP) and/or Photo-plethysmographs (PPGs).

4.3 Dataset Description

The Physionet Challenge 2015 [3] presented an opportunity for the participants to present different approaches towards improving the classification accuracy of true/false alarms. In this challenge, scoring was based on maximising True Positives

(TP) and True Negatives (TN), while minimising False Negatives (FN) and False Positives (FP). The scoring approach utilises an “err-on-the-safe side” approach, where the suppression of a true alarm (false negative) is penalised much more than events such as raising a false alarm. In other words, the fitness function used for this task is defined and introduced by the organisers of the Physionet Challenge 2015 [3]. The conceptual motivation underlying this metric is to attempt to maximise TP and TN while minimising FP and FN. The scoring weights FN more heavily than the FP. This is described in the following equation,

$$Score = \frac{TP + TN}{TP + TN + FP + 5FN}. \tag{4.42}$$

The Physionet 2015 challenge [3] offered a training dataset containing 750 publicly-available recordings as well as 500 private records for the purpose of scoring. This dataset consists of life-threatening arrhythmia alarm records that were collected from four hospitals in the United States and Europe. The recordings were sourced from devices designed by three major manufacturing companies of intensive care monitor devices. Each recording is 5 minutes or 5 minutes 30 seconds long at 250Hz and contains only one alarm. They are labelled ’true’ or ’false’ by a team of expert annotators. The commencement of the event is not later than 10 seconds before the end of the recordings.

In this challenge, participants could submit their code, which would be evaluated according to two type of events: event 1 (Real-time) and event 2 (retrospective). The aim of event 1 is to reduce the number of false alarms while no information is available after sounding of the alarm. In contrast, the goal of event 2 is to reduce the number of false alarms while an additional 30 seconds of data is available after sounding of the alarm. All recordings have a sample rate of 250Hz and contain two ECG leads and one or more pulsative waveforms (RESP, ABP or PPG). The ECGs may contain noise and pulsatile channels may contain movement artefacts and sensor disconnections. In this study, we focus on event 1 and trim all the data from the end to obtain a consistent length of five minutes; we choose a subset of 572 records. These contain the ECG leads II and V and PLETH signals. This decision is made to ensure that we train the neural network models on identical leads and pulsatile waveform. In this subset, there are 233 True alarms and 339 False alarms.

Disease Name	True Alarm	False Alarm
Asystole	17	77
Bradycardia	35	37
Tachycardia	90	4
Ventricular Flutter/Fibrillation	6	40
Ventricular Tachycardia	54	212

Table 4.1: *Subset of Physionet dataset (572 out of 750 recordings) which contains the ECG leads II and V and PLETH signal. We used this subset to ensure that we train the neural network models on identical leads and pulsatile waveform.*

In each n-fold, this dataset is divided into training, testing and validation, using 70% (400), 20% (114), and 10% (58) respectively. Table 4.1 describes the dataset.

4.4 Feature Selection

Since this research explores deep neuro-evolution rather than feature engineering, we decided to utilise the feature set that [10] suggested, and focus on the hybrid DFO-DeepNNs. The following explains the performed procedure to extract features from the physionet dataset [3]:

The short-time auto-correlation (STA) function is a straightforward approach to assess self-similarity. Let $x(n)$ be a time-discrete signal and $\omega_i(\nu) = x(n_i + \nu)$ be an analysis window with index i centred around n_i. The following equation gives a common definition of the STA for each lag η for a window of constant length L. Note that for simplicity, the index is omitted in the following derivation:

$$S_{STA}(\eta) = \frac{1}{L} \sum_{v=-\frac{L}{2}}^{\frac{L}{2}-\eta} \omega(\nu)\omega(\nu + \eta) \tag{4.43}$$

The η_{opt} is estimated as the interval between two heart beats and the analysis window L is set to a window that contains only two heartbeats ($L \approx 2\eta_{opt}$). In the case where $L \gg 2\eta_{opt}$, an averaging over multiple beats will occur; whereas $L \ll 2\eta_{opt}$, no estimation is possible. This can be fixed by introducing the lag-adaptive short time autocorrelation (LASTA).

$$S_{LASTA}(\eta) = \frac{1}{\eta}\sum_{v=0}^{\eta}\omega(\nu)\omega(\nu-\eta) \tag{4.44}$$

LASTA ensures that the exact number of samples necessary for each candidate ,lag η, is considered [300]. The following is a modified version of the average magnitude difference function (AMDF) used to assess self-similarity:

$$S_{AMDF}(\eta) = (\frac{1}{\eta}\sum_{v=0}^{\eta}|\omega(\nu)-\omega(\nu-\eta)|)^{-1} \tag{4.45}$$

This process also uses the lag-adaptive window and is inverted so that it assumes larger values for lags that indicate more self-similarity [300]. The maximum amplitude pairs (MAP) function considers the amplitude of the signal and accordingly MAP is considered as an indirect peak-detection. The MAP is considered as the third metric.

$$S_{MAP}(\eta) = \max_{v\in\{0,\ldots,n\}}(\omega(\nu)+\omega(\nu-\eta)) \tag{4.46}$$

The maximum of all sums of sample pairs that are separated precisely η time steps from each other is calculated. It was observed that the presented similarity estimators exhibit a complementary noise characteristic and the results can be improved by fusing the estimators based on a Bayesian approach [301], which reduces to:

$$\tilde{S}_{fused}(\eta) = S_{LASTA}(\eta)\cdot S_{AMDF}(\eta)\cdot S_{MAP}(\eta). \tag{4.47}$$

Furthermore, self-similarity is modality-independent and this concept can be extended towards multiple channels and modalities:

$$S_{fused}(\eta) = \tilde{S}_{fused,ECG}(\eta)\cdot\tilde{S}_{fused,PPG}(\eta)\cdot\ldots \tag{4.48}$$

Thus, for every window position i, the optimal interval can be obtained via:

$$\eta_{i,opt} = \underset{\eta}{argmax}[S_{i,fused}(\eta)] \tag{4.49}$$

which is the ratio of the peak height to the area under the curve. It indicates the amount of self-similarity this window exhibits (e.g. if the estimated interval is trustworthy).

In order to prepare the data for NN training, features are calculated in three categories namely ECG, BP, and all other channels (PLETH or ABP, and ALL). ECG: fusing only the available ECG signals (channels 1 and 2); BP: fusing only on the available pressure-based cardiac signals, all channels named PLETH or ABP; and ALL: fusing all cardiac-related signals such as all channels that are not named RESP. 27 features extracted from interval estimation of the above-mentioned categories are as follows:

$(1-3)$ $min(\eta_{i,opt}) : Minimum\ optimal\ interval$

$(4-6)$ $max(\eta_{i,opt}) : Maximum\ optimal\ interval$

$(7-9)$ $mean(\eta_{i,opt}) : Mean\ optimal\ interval$

$(10-12)$ $\sum_i \eta_{i,opt} : Sum\ optimal\ interval$

$(13-15)$ $mad(\eta_{i,opt}) : Median\ absolute\ deviation\ of\ optimal\ interval$

$(16-18)$ $std(\eta_{i,opt}) : Standard\ deviation\ optimal\ interval$

$(19-21)$ $std/mean(\eta_{i,opt}) : Standard\ deviation\ /\ Mean\ optimal\ interval$

$(22-24)$ $mean(Q_i) : Mean\ peak\ height\ to\ area\ under\ curve\ ratio$

$(25-27)$ $median(Q_i) : Median\ peak\ height\ to\ area\ under\ curve\ ratio$

In addition to these 27 features derived from beat-to-beat interval estimations, six features were determined by applying regular auto-correlation in a fixed window in order to estimate the signals average rhythmicity in different interval ranges.

(28) High-frequency ECG: Relative maximum of the auto-correlation function of all ECG signals. Evaluated for a lag of 0 - 2000 ms in a 16-second window before the alarm. Set to zero if the corresponding delay is shorter than 200 ms, to exclude artefacts.

(29) High-frequency BP: Similar to feature 28 using all available pressure-based signals.

(30) Low-frequency ECG: similar to 28, yet set to zero if all corresponding lag is shorter than 900 ms. The aim is to focus on slow rhythms.

(31) Low-frequency BP: Similar to 30, all available pressure-based signals used instead.

(32) Average rhythmicity: Absolute maximum of the average of auto-correlations of all cardiac-related signals. Evaluated for a lag of 0 - 1500 ms in a 5-second

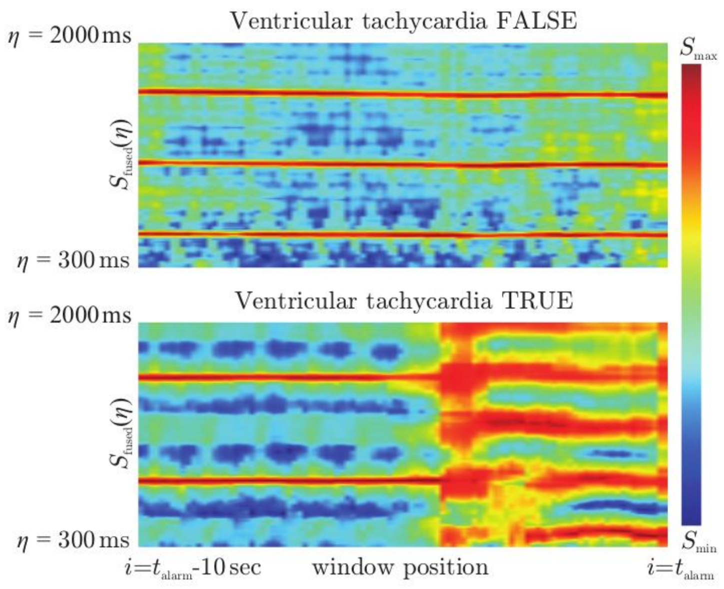

Figure 4.5: *Time courses for a false and a true ventricular tachycardia alarm as two-dimensional correlogram. While the y-axis constitutes η, the colour represents* $S_{i,fused}(\eta)$ [3].

window before the alarm. If the corresponding delay is shorter than 80 ms, values set to zero to exclude artefacts.

(33) Peak rhythmicity: Similar to feature 32, yet calculates the absolute maximum of maximums of all available auto-correlations.

Figure 4.1 draws a two-dimensional correlogram of two samples that diagnosed ventricular tachycardia for comparison. These samples labelled as the True or False alarm. The True alarm exhibits a relatively slow rhythm compared to the false alarm at first but forms an oscillating rhythm approximately four seconds before the alarm.

4.5 DFO Experiments and Results

As explained in section 3.2.5, the swarming behaviour of the individuals in DFO [200] consists of two tightly-connected mechanisms; one is the formation of the swarms, and the other is its breaking or weakening. In our study, we call the NN weights as position. Algorithm 4 describes the adapted DFO for NN. The NN weights' vector of the population is defined as follows:

$$\vec{w}_i^t = \left[w_{i1}^t, w_{i2}^t, ..., w_{iD}^t\right], \qquad i = 0, 1, ..., N\text{-}1 \tag{4.50}$$

where i represents the i^{th} individual, t is the current time step, and D is the dimensionality of the problem space. In our study, D is the number of weights in the given NN model. N is the number of individuals (population size) where in this study N is set to 500. This value has been achieved empirically based on the processing power, hardware limitations, and performance of the computer. For the continuous problems, $w_{id} \in \mathbb{R}$, and in the discrete cases, $w_{id} \in \mathbb{Z}$ (or a subset of $\mathbb{Z}$). In our study w_{id} is a subset of $\mathbb{R}$. In the first iteration, $t = 0$, the i^{th} vector's d^{th} component is initialised as:

$$w^0_{id} = \mathcal{N}(\mathit{0},\ \mathit{1}) \tag{4.51}$$

where $\mathcal{N}$ denotes the Gaussian distribution. Therefore, the population is randomly initialised with a set of weights for each in the search space.

In each iteration of the original DFO equation, the components of the NN weights vectors are independently updated, taking into account the component's value, the corresponding value of the best neighbouring individual with the best Physionet score (consider ring topology), and the value of the best individual in the whole swarm. Therefore the update equation is:

$$w^{t+1}_{id} \quad = \quad w^t_{i_{nd}} + u(w^t_{sd} - w^t_{id}) \tag{4.52}$$

where $w^t_{i_{nd}}$ is the weight (position) value of $\overrightarrow{w}^t_i$'s best $n$eighbouring individual in the d^{th} dimension at time step t, w^t_{sd} is the value of the $s$warm's best individual in the d^{th} dimension at time step t, and $u = U(0, 1)$ is a random number generated from the uniform distribution between 0 and 1. The update equation is illustrated in Fig. 4.6 for when $\overrightarrow{w_3}$ is to be updated. In our study, we investigated several extensions to improve the performance of the DFO for deep-network optimisation. We update equation 4.52 by taking into account the corresponding value of the best neighbouring individual and the value of the best in the whole swarm. Therefore, the adapted update equation is as follows:

$$w^{t+1}_{id} \quad = \quad w^t_{i_{nd}} + u(w^t_{sd} - w^t_{i_{nd}}) \tag{4.53}$$

Figure 4.6: *Sample update of w_i, where $i = 3$ in a 2D space.*

Three main components characterise the algorithm: a dynamic rule for updating the population's position (assisted by a social-neighbouring network that informs this update), and communication of the results of the best found individual to others; a dynamic mechanism to regulate *the disturbance threshold*, Δ, in order to control the behaviour of the population (explore or exploit) in the search space (Figure 4.7). The exploration is achieved by increasing the Δ (towards 1) and exploitation is reached by decreasing the Δ (towards 0).

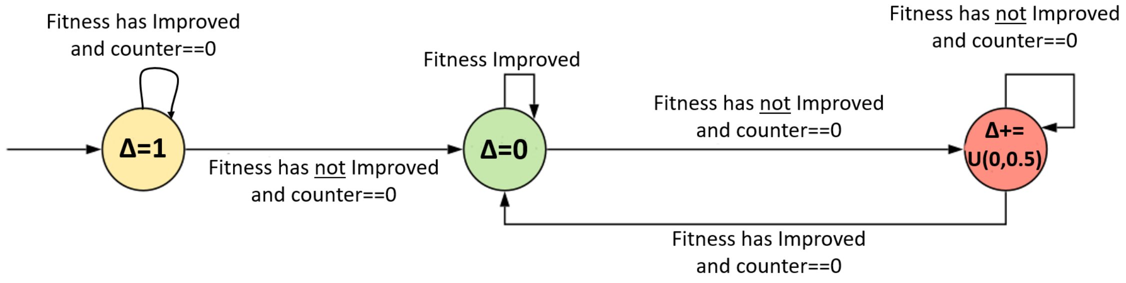
Fitness has Improved and counter==0
Δ=1
Fitness has not Improved and counter==0
Fitness Improved
Δ=0
Fitness has not Improved and counter==0
Fitness has not Improved and counter==0
Δ+= U(0,0.5)
Fitness has Improved and counter==0

Figure 4.7: *To dynamically adjust the disturbance threshold (Δ), a counter is used to monitor improvements. Initially the counter is set to 50. This value has been achieved experimentally. In each iteration, when the fitness improves, the counter is set to 50; otherwise, it is decreased by one.*

As stated earlier, the swarms are disturbed for various reasons. One of the impacts of these disturbances is the displacement of the individuals, which may lead to achieving a better Physionet score through the discovery of better weights for the NN. To consider this eventuality, an element of stochasticity is introduced to the update process. Based on this, the individual components of the population's weights' vectors are reset if a random number, u is less than the Δ. This process guarantees a disturbance to the otherwise permanent stagnation over the likely local maxima. In the original DFO equation, the disturbance is done by updating the parameter with a random number in the acceptable range of minimum and maximum value. In our study, we changed this parameter's update to correlate with the current Δ and the best neighbour, that is w_{id}^{t+1} is sampled from a Gaussian with mean set to $w_{i_{nd}}$ and variance to Δ^2. Figure . 4.11 demonstrates Δ behaviour in 3000 iterations.

Algorithm 4 summarises the adapted DFO algorithm.

In this algorithm, each member of the population is assumed to have two neighbours (i.e. ring topology).

Algorithm 4 Adapted DFO for Training

Input: population size N, model structure L, network weights $\overrightarrow{w_i}$, length of weights vector D, loss function $f()$.

```
1:  $\Delta = 1$
2:  while not converged do
3:      $\overrightarrow{w}_s = \arg\max\,[f(L(\overrightarrow{w}_i))], \quad i \in \{0, \ldots, N-1\}$
4:      for $i = 0 \rightarrow N\text{-}1$ and $i \neq s$ do
5:          $\overrightarrow{w}_{i_n} = \arg\max\,[f(L(\overrightarrow{w}_{i-1})), f(L(\overrightarrow{w}_{i+1}))]$
6:          for $d = 0 \rightarrow$ D-1 do
7:              if $U(0,1) < \Delta$ then
8:                  $w_{id}^{t+1} \leftarrow \mathcal{N}(w_{i_{nd}}^t, \Delta^2))$
9:              else
10:                 $w_{id}^{t+1} \leftarrow w_{i_{nd}}^t + u(w_{sd}^t - w_{i_{nd}}^t)$
11:             end if
12:         end for
13:         Dynamically update $\Delta$ (see Section 4.5 and figure 4.7)
14:     end for
15: end while
```

Output: Best agent's weight vector, $\overrightarrow{w_s}$.

In summary, the DFO is a minimalist numerical optimiser over continuous search spaces. Despite the algorithm's simplicity, it is shown that the DFO outperforms the standard versions of the well-known Particle Swarm Optimisation (PSO), Genetic Algorithm (GA) as well as Differential Evolution (DE) algorithms on an extended set of benchmarks over three performance measures of error, efficiency and reliability [200]. It is shown that DFO is more efficient at 84.62% and more reliable in 90% of the 28 standard optimisation benchmarks used; furthermore, when there exists a statistically significant difference, DFO converge to better solutions in 71.05% of the problem set.

4.5.1 Model Configuration

In NN, forward-propagation includes a set of matrix operations, where parameters include bias and weights that connect NN layers to each other. Our focus is to find the optimal weights and biases of the whole network which provide the most accurate classification for the given input. Depending on the type of neurons and input shape in an NN, the shape of the output weights of that neuron varies. This study uses two models: Model 1 uses four dense layers, and Model 2 uses one convolution

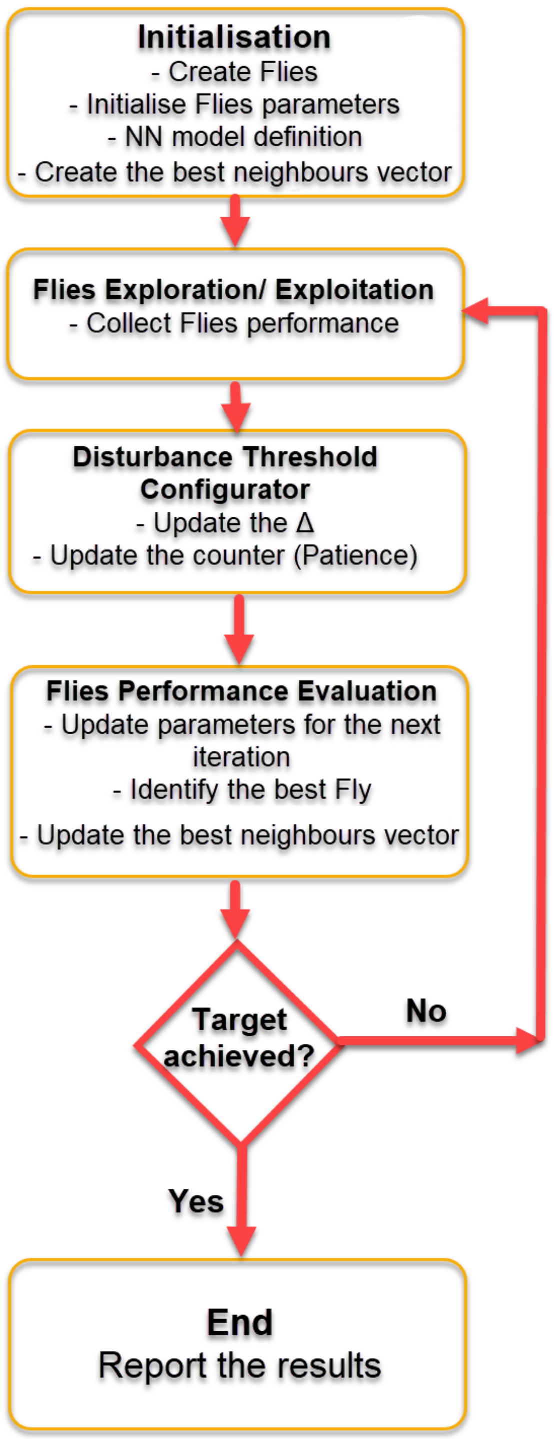
Initialisation
- Create Flies
- Initialise Flies parameters
- NN model definition
- Create the best neighbours vector
Flies Exploration/ Exploitation
- Collect Flies performance
Disturbance Threshold Configurator
- Update the Δ
- Update the counter (Patience)
Flies Performance Evaluation
- Update parameters for the next iteration
- Identify the best Fly
- Update the best neighbours vector
Target achieved?
No
Yes
End
Report the results

Figure 4.8: *An overview of the hybrid algorithm. Initially the the number of Flies (agents), NN model, best neighbour vector, and NN parameter vector is created. In the next step, the performance of each Fly (agent) is gathered and evaluated. Based on the counter value, the disturbance threshold, (Δ) if there is an improvement in the fitness function, the counter will be reset to 50 otherwise it will be reduced by one. In the next step, based on a comparison between the Δ and a randomly-generated value for each parameter, each Fly's parameter will be updated and neighbour vector and the best Fly in the population will be updated. The process of collecting Flies' performance will be repeated until either the maximum number of iterations is reached or the target fitness value is reached.*

and four dense layers. Due to the complexity of search space and the challenge of having very low number of samples (572), two shallow models with a small number of neurons is selected. This approach helped to formulate a less complex target function with fewer number of parameters to be optimised.

In model 1 of our study, the first layer is dense, with 64 neurons. In this layer, the input shape is (-1, 33, 1) and the output weights are (1, 64) where 1 is the third axis of the layer's input shape, and 64 is the number of neurons in this layer: [(-1, 33, 1) $\times$ (1, 64)] + (64) = (-1, 33, 64).
Model 2 includes a convolution as the first layer. This model has a similar input shape (-1, 33,1). This layer has 32 neurons, and the shape of the connecting weights to the next layer is (-1, 2, 1, 32), where 2 is the convolution window size, 1 is the third axis of the input shape, and 32 is the number of neurons in the convolution layer.

Layer Name	**No of Neurons**	**Weights Shape**	**Total Weights**	**Bias**
Dense 1 (Input)	64	(1, 64)	64	64
Dense 2	64	(64, 64)	4096	64
Dense 3	32	(2112, 32)	67584	32
Dense 4 (Output)	2	(32, 2)	64	2

Table 4.2: *Dense Model Structure*

Layer Name	**No of Neurons**	**Weights Shape**	**Total Weights**	**Bias**
Convolution 1 (Input)	32	(2, 1, 32)	64	32
Dense 2 (Input)	64	(32, 64)	2048	64
Dense 3	64	(64, 64)	4096	64
Dense 4	32	(1024, 32)	32768	32
Dense 5 (Output)	2	(32, 2)	64	2

Table 4.3: *Convolution-Dense Model Structure*

4.5.2 DFO Configuration

In our study, DFO is used to find the optimal weights in both NN models. The numbers of parameters in model 1 and model 2 are 71970 and 39234 respectively. The number of parameters is the sum of the number of weights and biases from all the layers (see Tables 4.2 and 4.3).

In our study, each member of the population (Fly or agent) has a set of parameters representing the weights (including biases) of the NN model. These parameters are initialised according to Eq. 4.51. Once all the parameters of each agent are initialised and loaded onto the NN model, the fitness (score) of each agent calculates the mean Physionet score using 5-fold cross-validation. After each iteration, each agent's best neighbour, and the best agent in the swarm are identified. The best agent holds the highest Physionet score amongst the population. An overview of this hybrid model is provided in figure 4.8

Before updating each component, a value u is sampled from a uniform distribution $U(0,1)$. If u is less than Δ, the component is updated with the agent's best neighbour as focus (μ), therefore, $w_{id}^{t+1} \leftarrow \mathcal{N}(w_{i_nd}^{t}, \Delta^2)$. Otherwise, the agent's component is updated with the focus of the best agent in the swarm $w_{id}^{t+1} \leftarrow w_{i_nd}^{t} + u(w_{sd}^{t} - w_{i_nd}^{t})$.

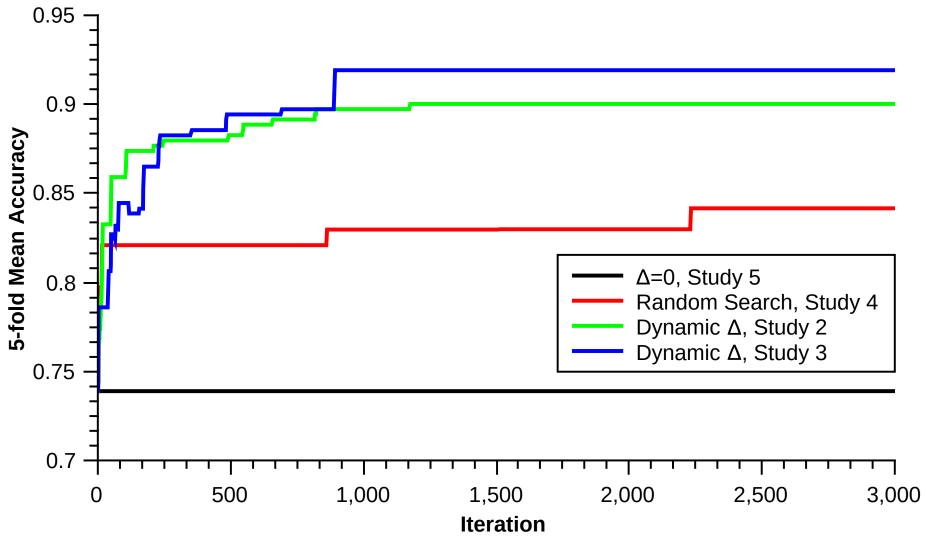

Figure 4.9: *Model 1: 5-fold cross validation mean accuracy over 3000 iterations. Mean accuracy trend for standard DFO, updated DFO, dynamic and constant disturbance threshold (Δ).*

We implemented a mechanism to control the value of Δ. In the first phase or the parameter optimisation, to manage the algorithm towards exploration, Δ to set to 1; this process continues until there is no improvement in 50 iterations[2] (see figure 4.11). Afterwards, Δ is set to zero, allowing the agents to converge to

[2]Note that 50 is an experimentally suggested value and further studies are required to find a theoretically optimal value.

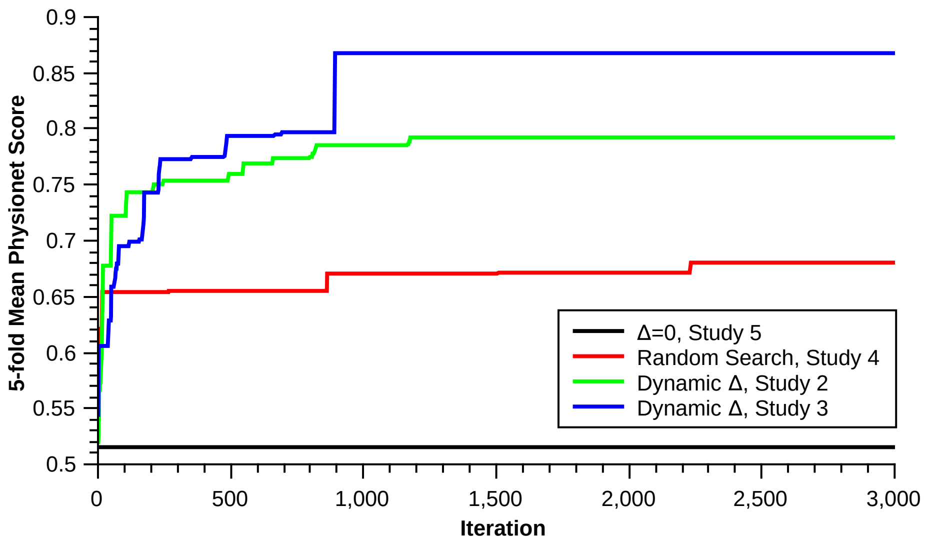

Figure 4.10: *Model 1: 5-fold cross validation mean Physionet score over 3000 iterations. Mean Physionet score trend for standard DFO, updated DFO, with dynamic and constant disturbance threshold* (Δ)*.*

the best location they can find. Once again, if no improvement is noticed in the Physionet score in 50 iterations, Δ is increased by a random number between 0 and the experimental value of 0.5 ($\Delta \leftarrow \Delta + U(0, 0.5)$). The algorithm is then set to run. If there is an improvement followed by a 50 iteration state of idleness, Δ is then set to 0 again to exploit the recent finding. Alternatively (i.e. if there is no improvement) after the permitted idle time-frame, Δ is incremented further. In a situation when $\Delta > 1$, Δ is set to $U(0, 1)$, and the process continues as explained above until the termination points, which is $3,000$ iterations. Figure 4.9 and 4.10 demonstrate trend of improvement in accuracy and Physionet score over iterations. In this experiment, we compared 5-fold cross-validation of the first and fifth rank in the Physionet challenge 2015 with our designed NN models (see Table 4.4). The optimal weights of our models are calculated through both backpropagation and gradient-free algorithms. Model 1 consists of four dense layers and Model 2 consists of a convolution layer and four dense layers (see Table 4.2 & 4.3).

As a benchmark, we calculated the accuracy and Physionet scores of [302] and [10]. They achieved the accuracy and the Physionet scores of (87.24%, 85.50%) and (87.78%, 80.09%) respectively. We investigated various implementations of DFO as well as random search. The DFO's modified version for Model 1 & 2 achieved

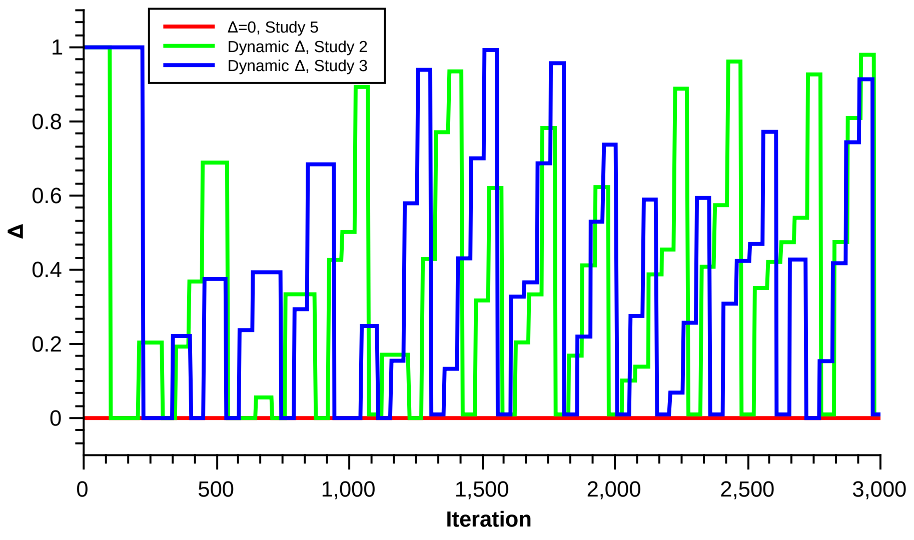

Figure 4.11: *Model 1: 5-fold cross validation disturbance threshold (Δ) trend. Visualising Δ trend, considering dynamic and constant disturbance threshold (Δ) over 3000 iterations.*

the highest score among the other results (see Table 4.4). Their accuracy and Physionet scores are ((91.91%, 86.77%) and (91.88%, 86.81%) respectively. The behaviour of DFO, while having a constant Δ value of 0, and Random Search is also investigated. Their accuracy and the Physionet scores are (73.89%, 51.53%) and (84.16, 68.03) respectively. The models optimised via neuroevolution with DFO outperforming both (i) the networks trained by backpropagation, as well as (ii) the winning entries of the Physionet challenge. The reason for poor performance of backpropagation in contrast with the hybrid model can be justified by the fact that that backpropagation is performing function minimisation, in which we can apply the chain rule of derivatives at each layer because they are function compositions. First order optimisation is looking at the gradient and moving in a downhill direction (minimum error) by some fixed-step size, by which it is possible that we end up at the local minima or saddle point instead of global minima in our parameter search space [303, 304].

Author	Method	Mean Accuracy	Physionet Score
By [10]	SVM, BCTs, DACs	87.24% (+/- 2)	85.50% (+/- 3)
By [302]	Fuzzy Logic	87.78% (+/- 4)	80.09% (+/- 8)
Our Study 1	Dense Network, Backpropagation	87.70% (+/- 3)	75.35% (+/- 7)
Our Study 2	Dense NN, Standard DFO, Dynamic Δ, using Eq. 4.52	90.02% (+/- 3)	79.23% (+/- 5)
Our Study 3	**Dense NN, Adapted DFO, Dynamic Δ, using Eq. 4.53**	**91.91% (+/- 4)**	**86.77% (+/- 4)**
Our Study 4	Dense NN, Random Search	84.16% (+/- 3)	68.03% (+/- 5)
Our Study 5	Dense NN, Standard DFO, $\Delta = 0$ (i.e. no disturbance)	73.89% (+/- 2)	51.53% (+/- 4)
Our Study 6	Conv-Dense NN	88.21% (+/- 3)	76.34% (+/- 6)
Our Study 7	**Conv-Dense NN, Adapted DFO, Dynamic Δ**	**91.88% (+/- 2)**	**86.81% (+/- 4)**
Our Study 8	Conv-Dense NN, Random Search	75.63% (+/- 8)	52.37% (+/- 9)

Table 4.4: *Accuracy and Physionet score over 5-fold cross validation for first and fifth rank in Physionet challenge 2015, NN optimised with backpropagation and adapted DFO algorithm with constant and dynamic disturbance threshold (Δ)*

4.6 Chapter Summary

In summary, we have presented a method for training neural networks based on neuroevolution, by utilising the DFO algorithm in a gradient-free, population-based scheme. We evaluated our approach to the problem of detecting false alarms in ICUs using the Physionet dataset. The results obtained show that the proposed method outperforms (i) backpropagation-trained networks with the same or similar architecture, as well as (ii) the winning entries of the Physionet challenge. We also addressed the possibility of imputing missing signals while having two or five signals for each sample. Later, the imputed signals using the five-signal method were used for the purpose of classification to discover the model's hyper-parameters using the hybrid model discussed in this chapter.

5.Signal Imputation with Adversarial Networks

5.1 Introduction

In the previous chapter, the possibility of using gradient-free in place of gradient-based algorithms was explored in conditions of a limited number of samples. This was due to a smaller number of samples (572 out of 750) with the same type of bio-signal data. In this chapter, we aim to use Generative Adversarial Networks (GANs) to impute those missing bio-signals. This enables us to include more samples (738 out of 750) in the task of reducing false Arrhythmia alarm events in the ICU.

5.2 Motivation

In this research we conducted two separate studies to evaluate the quality of an imputed missing signal from another signal and also to impute five signals in instances of samples from multiple channels. Later, we used the imputed samples to train the model described in chapter 3.5 to evaluate the effect of imputed samples on accuracy and the Physionet score.

As mentioned in the previous chapter, each signal has a length of 82,500 data points with a frequency of 250hz, equating to a recording with a length of 5 minutes 30 seconds. The recording includes two leads of ECG(II and V) and maximum two pulsatile waveforms (arterial blood pressure waveform [’ABP’], respiration ['RESP'], photo-plethysmogram ['PLETH']). In this chapter, we consider using a window size of 250 (equivalent to one second) to slice the signals to ensure the inclusion of at least one complete pulsation of the heart (heartbeat) at each window. Therefore, each signal is separated into 300 samples with 250 length. See table 5.5 for detailed information about total number of each channels within the dataset used.

Table 5.5: *Subset of Physionet dataset (648 out of 750 recordings) that contain the ECG leads II and V and Photo-plethysmogram, Arterial blood pressure, and Respiration signals. We used this subset to ensure that we train the NN models on an identical set of signals.*

Lead II	**Lead V**	**Arterial blood pressure**	**Respiration**	**Photoplethysmogram**	**Total samples**
648	648	289	243	572	648

5.3 Methodology - Two-Signal Imputation

The idea for our first approach is inspired from an imputation method that uses Generative Adversarial Networks (GANs) for missing view problems [305]. In this method we combine CycleGAN [103] and a multi-modal Denoising Auto-Encoder(DAE), and use it to generate ECG leads II and V from each other. A DAE is utilised and trained to operate on a generated signal by CycleGAN, and reconstruct a repaired signal [306]. We use the CycleGAN to learn cross-domain relations between ECG signals and from paired data in a DAE to learn between-view correspondences. Additionally, we denoise the signals generated by CycleGAN to improve their quality by learning a shared representation from pairs (x, y). The CycleGAN consists of G_{XY}, G_{YX}, D_{XY}, and D_{YX}, representing generators and discriminators of the model respectively.

The mapping between domain space II and V is addressed as $G_{IIV} : II \to V$ and $G_{VII} : V \to II$.

$DAE : II \times V \to II \times V$.

In this approach, The CycleGAN estimates and maps given II using V and vice versa by concentrating on domain translation. Considering the complex structure of this method, there are four loss functions to consider. In order to produce an equation for overall loss function minimisation, the adversarial loss, multi-modal

DAE loss, cycle consistency loss, and overall loss will be explained and then applied.

5.3.1 Multi-modal DAE Loss

The DAE extracts features from each view in their primary parallel layers. Later, the features are concatenated and fed to a stack of layers to reduce the dimensions and form a shared representation tensor. Lastly, each view is gathered through separate output layers. The reconstruction function and inner representation of each view which creates the basic structure of the multi-view data is optimised jointly during the training process. Considering the original pair from the dataset (II, V) and the CycleGAN mappings $G_{IIV} : II \rightarrow V$ and $G_{VII} : V \rightarrow II$, the inputs to the DAE will be two pairs of (II, G_{IIV}) and (V, G_{VII}), which are the reconstructed leads II and V. The objective function is as follows:

$$\begin{aligned}\mathcal{L}_{DEA}(DAE, G_{IIV}, G_{VII}) = \\ &\mathbb{E}_{(II,V)\sim P_(data)((II,V))}[\| DEA(II, G_{IIV}(II)) - (II, V) \|_2^2] + \\ &\mathbb{E}_{(II,V)\sim P_(data)((II,V))}[\| DEA(G_{VII}(V), V) - (II, V) \|_2^2]\end{aligned} \tag{5.54}$$

5.3.2 Adversarial Loss

Assuming projection of lead II and V are shown as $P_{II}(II, V) = II$ and $P_V(II, V) = V$; The II part or V part are taken from the pair (II, V), and the adversarial loss of the combined functions, $P_{II} \circ DAE(G_{IIV}(II), II) : II \rightarrow V$ and $P_{II} \circ DAE(G_{VII}(V), V) : V \rightarrow II$, is as follows:

$$\begin{aligned}\mathcal{L}^{V}_{DEAGAN}(DAE, G_{IIV}, D_V) = \\ &\mathbb{E}_{(V)\sim P_{data}(V)}[\log(D_V(V))] + \\ &\mathbb{E}_{(II)\sim P_{data}(II)}[\log(1 - D_V(P_V \circ DAE(II, G_{IIV}(II)))],\end{aligned} \tag{5.55}$$

and

$$\mathcal{L}^{II}_{DEAGAN}(DAE, G_{VII}, D_{II}) = \\ \mathbb{E}_{(II)\sim P_{data}(II)}[\log(D_{II}(II))]+ \\ \mathbb{E}_{(V)\sim P_{data}(V)}[\log(1 - D_{II}(P_{II} \circ DAE(G_{VII}(V)))]. \tag{5.56}$$

As it can be understood from the above equations, the adversarial loss takes effect on the GAN as well as the DAE. The loss function in eq.5.55 measures variations between the output of the complex function $P_V \circ DAE(II, G_{IIV}(II))$ and the observed lead V and the loss represented in eq.5.56 computes the difference between the output of the complex function $P_{II} \circ DAE(G_{VII}(V))$ and the observed lead II. The discriminators D_{II} and D_V are responsible for discriminating between real and generated signals using the above-mentioned complex functions. To optimise the proposed model, at the final stage of training, following the standard GAN mechanism, which solves the minmax challenge, we target optimisation of $min_{DAE,G_{IIV}} max_{D_V} \mathcal{L}^{V}_{DEAGAN}$ and $min_{DAE,G_{VII}} max_{D_{II}} \mathcal{L}^{II}_{DEAGAN}$ respectively for DAE, G_{II}, D_V and DAE, G_V, D_{II} networks.

5.3.3 Cycle Consistency Loss

The aim is to enable the adversarial network to map the input ECG lead back to itself by passing it through the two generators G_{IIV} and G_{VII}. This is done by minimising the cycle consistency loss and GAN loss simultaneously. The cycle consistency loss function guarantees that the mapping function can map an input to the aimed output. To increase the probability of the desired outcome of the mapping function, we use the cycle consistency loss function as follows [103]:

$$\mathcal{L}_{CYC}(G_{IIV}, G_{VII}) = \\ \mathbb{E}_{(II)\sim P_{data}(II)}[\| G_{VII} \circ G_{IIV}(II) - II \|_1]+ \\ \mathbb{E}_{(V)\sim P_{data}(V)}[\| G_{IIV} \circ G_{VII}(V) - V \|_1] \tag{5.57}$$

The cycle consistency loss function has been used in DualGAN [307], Disco-

GAN [308], and CycleGAN [103].

5.3.4 Overall Loss

The overall loss function utilises all the above-mentioned losses and is formulated as follows:

$$\begin{aligned}\mathcal{L}(DAE, G_{IIV}, G_{VII}, D_{II}, D_V) = \\ &\lambda_{DAE}\mathcal{L}_{DAE}(DAE, G_{IIV}, G_{VII}) + \\ &\lambda_{CYC}\mathcal{L}_{CYC}(G_{IIV}, G_{VII}) + \\ &\mathcal{L}^{II}_{DAEGAN}(DAE, G_{VII}, D_{II}) + \\ &\mathcal{L}^{V}_{DAEGAN}(DAE, G_{IIV}, D_V)\end{aligned} \tag{5.58}$$

The hyper-parameters λ_{CYC} and λ_{DAE} are then used to balance different terms in the objective function. In the above equation, $\mathcal{L}_{CYC}$ and $\mathcal{L}_{DAE}$ do not use the correspondence in pairs, because the $\mathcal{L}_{CYC}$ loss will measure the full-cycle projection of signal II or V to itself and the $\mathcal{L}_{DAE}$ loss uses the samples that have been randomly generated.

We solve the minmax challenge by finding the optimal parameters for the model's generators G_{II} and G_V, discriminators D_{II} and D_V, and the denoising auto-encoder DAE as follows:

$$\min_{DAE, G_{IIV}, G_{VII}} \max_{D_{II}, D_V} \mathcal{L}(DAE, G_{IIV}, G_{VII}, D_{II}, D_V) \tag{5.59}$$

5.4 Experiments and Results - Two-Signal Imputation

In this experiment, we use a pair of signals (leads II and V) and aim to generate one lead from another. To apply the idea, we implemented a model by extending the image domain transfer idea (VIGAN). This includes two sets of discriminators D_{IIV}, D_{VII} and generators G_{IIV}, G_{VII} and a denoising autoencoder DAE. In the following sections we discuss model setup and evaluate the quality of the generated leads.

5.4.1 Model Setup

This model consists of two generators which include a stack of four 1D Conv and two 1D De-CONV layers and two dense layers on the top. The generator G_{IIV}, inputs lead II and maps lead V and the generator G_{VII}, inputs lead V and maps it to lead II; both have been configured with $ADAM$ optimiser and the learning rate set to $2e-4$. The discriminators D_{IIV} and D_{VII} include a stack of three layers of 1D-CONV and a dense layer at the top. In these models, the learning rate has been set to $2e-4$.

The denoising auto-encoder DAE includes eleven dense layers and accepts two leads II and V, one of which is real while the other is generated as inputs and outputs two denoised signals. The optimiser is set as $ADAM$ with $2e-3$ as the learning rate.

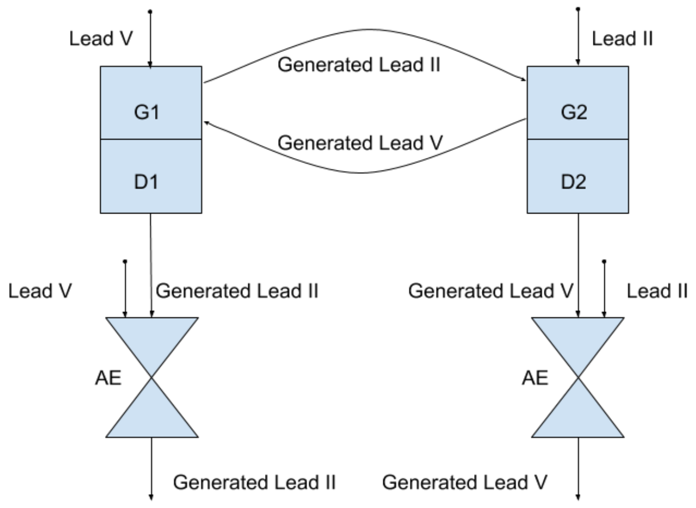

Figure 5.12: *The structure of two-lead imputation model.*

5.4.2 Model Performance

After setting up the model, first we trained DAE using the real II and V leads where the denoising auto-encoder learns both the signal domains and is able to map the input signals to corresponding domains. Later, we trained the G_{IIV} and D_{IIV} and G_{IIV} and D_{IIV} alternately where the discriminators learn to distinguish between real and generated signals (maximisation), and the generators learn to map the input signals from one domain to another. Table 5.6 demonstrates the results achieved using two generators with discriminators and also adding a denoising auto-encoder at the top. Two sample outputs considering two pairs of different input leads II and V, $CycleGAN$, and $CycleGAN$ with DAE are shown in fig. 5.13 and 5.14.

The following plots represents the full cycle of generating each lead and the output of DAE:

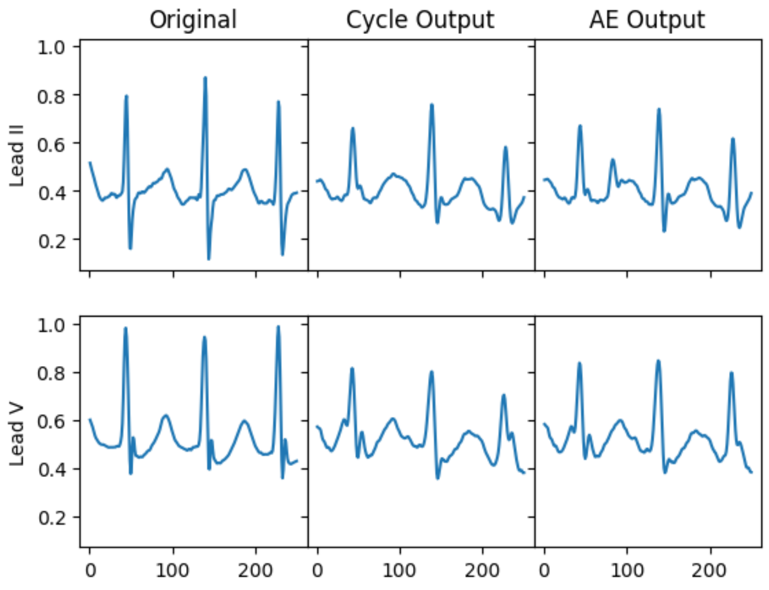

Figure 5.13: *Sample 1, a clear heartbeat used to impute lead II and V from each other. The plots demonstrate the original signals, output of the cycle and DAE.*

We have evaluated the performance of the model by calculating the MSE of the actual and generated TEST signals.

Lead	**Cycle MSE**	**DAE MSE**
II	0.0009	0.0007
V	0.0022	0.0008

Table 5.6: *The MSE results of generated Lead II and V using cycleGan and DAE on the top of cycleGan.*

5.5 Methodology - Five-Signal Imputation

This section explains our suggested model consisting of an auto-encoder and only one generator G and discriminator D; it uses the desired target domain label l to map a physiological signal from one domain to the target domain. Considering the available dataset, we built a model to map between leads II and V, blood

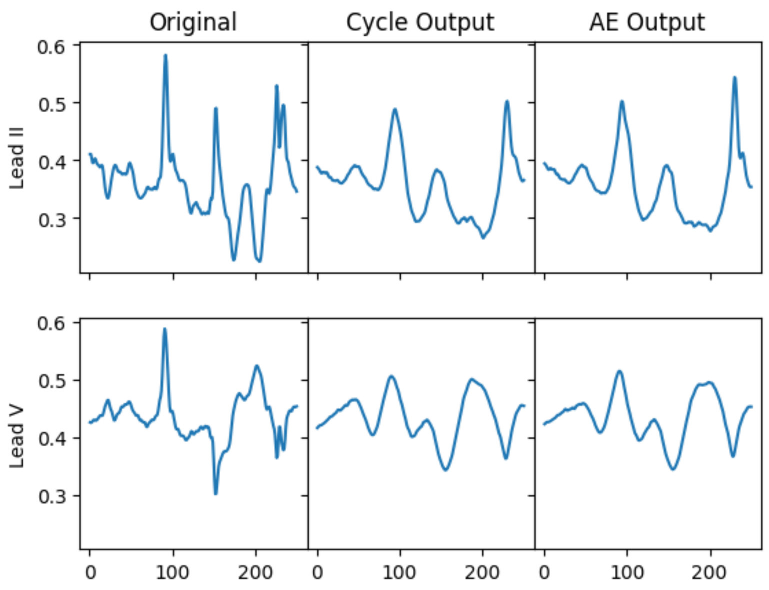

Figure 5.14: *Sample 2, a sample of abnormal heartbeat used to impute lead II and V from each other. The plots demonstrate the original signals, output of the cycle and DAE.*

pressure(BP), photo-plethysmogram($PLETH$), and respiration($RESP$) signals. The generator and discriminator are demonstrated as follows:

$$\begin{aligned} &G(X,l) \rightarrow Y, \quad where\ X\ and\ Y\ \in \{II, V, BP, RESP, PLETH\}, \\ &D:\ X\ \rightarrow\ \{D_{source}(X),\ D_{label}(X)\}, \end{aligned} \tag{5.60}$$

where D_{source} is the probability distribution over the source signal and $D_{label}(X)$ represents the probability distribution of the target domain label.

5.5.1 Adversarial Loss

Adversarial loss is calculated to distinguish the generated signal from the real signal considering projection of the input signals X and target label l as $P_Y(X,l) = Y$ where $XandY \in \{II, V, BP, RESP, PLETH\}$ and l represents the target signal name. Each target signal can be projected through four other signals. For instance, lead II can be generated through $P_{II}(V, l = II) = II, P_{II}(BP, l = II) = II, P_{II}(RESP, l = II) = II, P_{II}(PLETH, l = II) = II$. We use an auto-encoder to reconstruct the target signal using the four generated signals with the same target label.

The generator G, generates a signal while conditioned on the target label l and input signal X and the discriminator D aims to identify real and fake signals. The discriminator tries to maximise the objective and the generator tries to minimise it. The adversarial loss is formulated as follows:

$$\begin{aligned}\mathcal{L}_{adv} =& \mathbb{E}_X[D_{source}(X)] - \\ & \mathbb{E}_{X,l}[D_{source}(G(X,l))] - \\ & \lambda_{gp}\mathbb{E}_{\hat{X}}[(\| \nabla_{\hat{X}} D_{source}(\hat{X}) \|_2 - 1)^2],\end{aligned} \tag{5.61}$$

where λ_{gp} has been set to 10 and $\hat{X}$ represents a straight line between each pair of real and generated signals that has been uniformly sampled.

5.5.2 Domain Classification Loss

This loss is used to measure the optimisation process of generator G and discriminator D while training them to map input signals and target labels to the target signals. The D aims to minimise belonging objective to classify a given real signal to belonging domain label. The G aims to generate signals to be categorised as a member the target domain. The above-mentioned objectives are formulated as follows:

$$\begin{aligned}\mathcal{L}^{fake}_{class} &= \mathbb{E}_{X,l}[-\log D_{class}(l'|X)] \\ \mathcal{L}^{real}_{class} &= \mathbb{E}_{X,l'}[-\log D_{class}(l|G(X,l))]\end{aligned} \tag{5.62}$$

5.5.3 Reconstruction Loss

This loss function is used to confirm the model is able to map the input signal to itself by passing it through the G with relevant target labels l and l'. Using the adversarial and classification loss functions does not guarantee that we are able to map the generated signal to the original signal, although they enable the G to learn to map the input to the target signal. To tackle this problem, reconstruction has been calculated. This will measure the difference between the input signal X

and the regenerated signal $G(G(X, l), l')$:

$$\mathcal{L}_{rec} = \mathbb{E}_{X,l,l'}[\| X - G(G(X, l), l') \|_1] \tag{5.63}$$

5.5.4 Auto-encoder Loss

Given that there are five different signals, and each can be used as an input signal fed to the G, we can therefore generate one signal through four other signals. The mapping considering multi-lead projection can be demonstrated as follows:

$$\begin{aligned} P_{II} \circ A(G(V, II), G(BP, II), G(RESP, II), G(PLETH, II)) :& \\ \{V, BP, RESP, PLETH\} \to II,& \end{aligned} \tag{5.64}$$

$$\begin{aligned} P_{V} \circ A(G(II, V), G(BP, V), G(RESP, V), G(PLETH, V)) :& \\ \{II, BP, RESP, PLETH\} \to V,& \end{aligned} \tag{5.65}$$

$$\begin{aligned} P_{BP} \circ A(G(II, BP), G(V, BP), G(RESP, BP), G(PLETH, BP)) :& \\ \{II, V, RESP, PLETH\} \to BP,& \end{aligned} \tag{5.66}$$

$$\begin{aligned} P_{RESP} \circ A(G(II, RESP), G(V, RESP), G(BP, RESP), G(PLETH, RESP)) :& \\ \{II, V, BP, PLETH\} \to RESP,& \end{aligned} \tag{5.67}$$

$$\begin{aligned} P_{PLETH} \circ & A(G(II, PLETH), \\ & G(V, PLETH), G(BP, PLETH), G(RESP, PLETH)) : \\ & \{II, V, BP, RESP\} \to PLETH \end{aligned} \tag{5.68}$$

Considering different mapping from multiple sources for the same destination signal, we use an auto-encoder A to extract features from each view in separate initial parallel layers. The calculated features are combined into a tensor and fed into a stack of layers to form a shared representation tensor. The output layer produces a signal with dimensions identical to the original signal X. The reconstruction function and inner representation of the view that establishes the basic structure of the multi-sourced-view data are optimised jointly during the training process. Considering the target label l and original signal from the data set $X, Y \in \{II, V, BP, RESP, PLETH\}$ and the model's mappings $G(X, l) \to Y$, the inputs to the A will be five sets mentioned in eq. 5.64, 5.65, 5.66, 5.67, and 5.68, which are the reconstructed leads II and V, BP, $RESP$, and $PLETH$.

The objective function is formulated as follows:

$$\begin{aligned}
&\mathcal{L}_A(A, G_{II,l}, G_{V,l}, G_{BP,l}, G_{RESP,l}, G_{PLETH,l}) = \\
&\mathbb{E}_{(X,l=II)\sim P_(data)((X,l=II))}[\| A(G_{V,l}(V, l), \\
&G_{BP,l}(BP, l)), G_{RESP,l}(RESP, l)), G_{PLETH,l}(PLETH, l)) - II \|_2^2]+ \\
&\mathbb{E}_{(X,l=V)\sim P_(data)((X,l=V))}[\| A(G_{II,l}(II, l), \\
&G_{BP,l}(BP, l)), G_{RESP,l}(RESP, l)), G_{PLETH,l}(PLETH, l)) - V \|_2^2]+ \\
&\mathbb{E}_{(X,l=BP)\sim P_(data)((X,l=BP))}[\| A(G_{II,l}(II, l), \\
&G_{V,l}(V, l)), G_{RESP,l}(RESP, l)), G_{PLETH,l}(PLETH, l)) - BP \|_2^2]+ \\
&\mathbb{E}_{(X,l=RESP)\sim P_(data)((X,l=RESP))}[\| A(G_{II,l}(II, l), \\
&G_{V,l}(V, l)), G_{BP,l}(BP, l)), G_{PLETH,l}(PLETH, l)) - RESP \|_2^2]+ \\
&\mathbb{E}_{(X,l=PLETH)\sim P_(data)((X,l=PLETH))}[\| A(G_{II,l}(II, l), \\
&G_{V,l}(V, l)), G_{BP,l}(BP, l)), G_{RESP,l}(RESP, l)) - PLETH \|_2^2]
\end{aligned} \tag{5.69}$$

5.5.5 Overall Loss

The Overall objective function considering the generator G, discriminator D, and auto-encoder A is formulated as follows:

$$\begin{aligned}\mathcal{L}(A, G_{X,l}, D) =& \lambda_A \mathcal{L}_A(A, G_{II,l}, G_{V,l}, G_{BP,l}, G_{RESP,l}, G_{PLETH,l})+ \\ & \lambda_{rec}\mathcal{L}_{rec}(X, l, l')+ \\ & \lambda_{fake}\mathcal{L}_{class}^{fake}(D, X, l, l')+ \\ & \lambda_{real}\mathcal{L}_{class}^{real}(G, X, l, l')+ \\ & \lambda_{adv}\mathcal{L}_{adv}(G, D, l, \hat{X})\end{aligned} \tag{5.70}$$

5.6 Experiments and Results - Five-Signal Imputation

In this experiment, we implemented a model that shares the generator and discriminator between multiple signal domains. This experiment is an adaptation extension to the StarGAN model [309], which is proposed for style transfer between domains. In our model, we added an auto-encoder on the top of the proposed StarGAN, which accepts a collection of the specific signals along with a target label, in order to produce a target signal. The difference between this approach and the previous approch (two-signal imputation) is the use of a shared model structure instead of the implementation of separate models for each signal generation.

This approach simplifies the process of training, including calculating loss and updating weights. To specify the expected (target) signal while training or predicting using the shared generator, we concatenate the input signal with the expected target label as input to the generator. The discriminator also produces two outputs, namely a binary judgement to distinguish between real and fake signals and a classifier of real signals to its corresponding domain.

Signal Name	**II**	**V**	**Blood Pressure**	**Respiration**	**Photoplethysmogram**
Label	00001	00010	00100	01000	10000

Table 5.7: *List of target labels mapped in binary vector.*

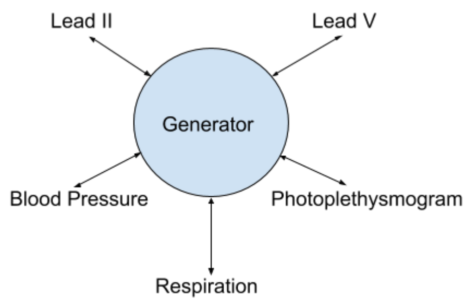

Figure 5.15: *The multi-lead generator structure*

Since we can generate a signal through the other four signals, we will have four generated signals with the same label. We added an auto-encoder on top of the model to combine the generated signal with identical labels.

5.6.1 Model Setup

This approach includes a generator G, discriminator D, and an auto-encoder D. The G includes a stack of six layers of 2D CONV and three 2D De-CONV layers which accept an input signal X along with a target label l. This model has one output, which is the generated target signal. The D consists of a stack of five layers of 1D CONV, which receives a signal as an input and outputs the target label l' and a decision on generated or real signal Y. Both G and D benefit from the $ADAM$ optimiser with learning rate of $2e-4$. The auto-encoder A includes a stack of ten dense layers and enables us to combine the generated signals for the same domain - having the same target label l - with different source signals, in order to produce the target signal.

5.6.2 Model Performance

After setting up the model, we trained the G and D alternatively, where the discriminator learns to distinguish between real and generated signals considering the target label l, and output the target label l' , and the generator learns to map

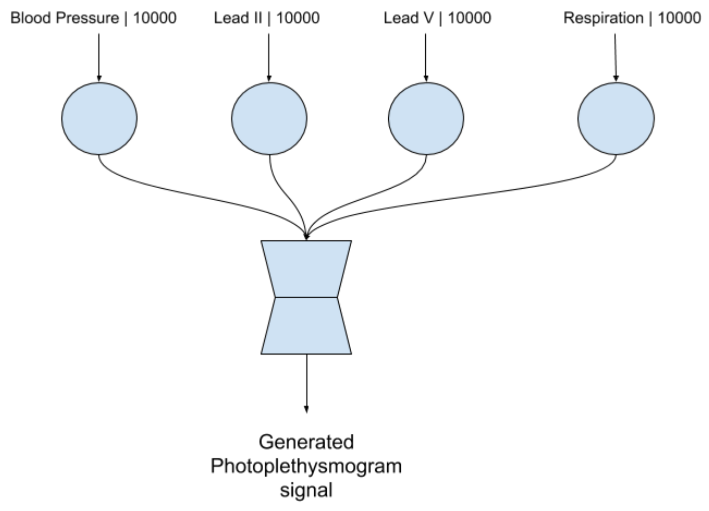

Figure 5.16: *photo-plethysmogram generated using the rest of available signals*

the input signals along with the target label l from one domain X to another Y. Table 5.7 lists the target labels and their corresponding binary vectors fed into the generator along with the source signal X. After training for 100 epochs, the generated signals are evaluated before and after passing through the A. The results are shown in table 5.8. Analysing the results, the model shows very poor performance in terms of generating missing respiration signals due to a lack of relationship between the heartbeat and respiration rate. Within the dataset, there are plenty of samples with similar bio-signals, but with different respiration rates. In the following section we evaluate the performance of the same model discussed in chapter 4.5 using the new partially-generated samples. We have removed the respiration signal from all samples since their quality is very low.

	Source of Generated Signal					
	Lead II	**Lead V**	**Respiration**	**Blood P.**	**Pleth.**	**AE Output**
Lead II	-	0.0009	4695356	0.0795	0.0148	0.0011
Lead V	0.0007	-	4225240	0.0708	0.0170	0.0052
Respiration	7482491	6665219	-	4181521	5871420	8765679
Blood P.	0.0310	0.0207	4610502	-	0.0540	0.03333
Pleth.	0.0034	0.0065	4883125	0.0752	-	0.0039

Table 5.8: *The MSE of generated signals with different sources.*

5.6.3 Classification Performance

We generated samples using the method discussed in section 5.6. Detailed information regarding the type of disease and alarm type is offered in table 5.9. The total number of samples is 738, which comprises 444 and 294 samples with the False and True alarm label respectively. Later we followed the same steps discussed in sections 4.4 and 4.5 to select features and calculate the Physionet score. Both approaches using backpropagation and DFO have been used to optimise the model separately and the results obtained are demonstrated in table 5.10, and a comparison results table with our past experiments and benchmarks are addressed in table 5.11.

Disease	**Alarm Type**	**No. of Samples**
Asystole	False	96
Asystole	True	22
Bradycardia	False	42
Bradycardia	True	46
tachycardia	False	9
tachycardia	True	131
Ventricular flutter fib	False	51
Ventricular flutter fib	True	6
Ventricular tachycardia	False	246
Ventricular tachycardia	True	89
Total False Alarm Samples		444
Total True Alarm Samples		294

Table 5.9: *Detailed information regarding the number samples used to evaluate the classification performance.*

Method	**TP**	**FP**	**TN**	**FN**	**Mean Accuracy**	**Physionet Score**
Exp 2, Backpropagation, (StarGAN with AE)	262	32	417	27	91.98	80.02
Exp 2, Adapted DFO using Eq. 4.53 (StarGAN with AE)	**265**	**29**	**429**	**15**	**94.01**	**86.96**
Exp 3, Backpropagation (StarGAN only)	260	34	413	31	91.19	78.07
Exp 3, Adapted DFO using Eq. 4.53(StarGAN only)	264	30	425	19	93.36	84.64

Table 5.10: *The Accuracy and Physionet score obtained using the generated data over 5-fold cross validation. The results are gathered using StarGAN only and StarGAN with Auto Encoder. Comparison between the results shows that using an auto-encoder on the top of StarGAN improves the accuracy and the Physionet score.*

5.7 Conclusion

In this chapter we explored the possibility of imputing missing signals within a sample through a two-signal and a multi-signal imputation approach. The two-signal imputation approach is useful where we have only two signals for each sample.

Author	**Method**	**Mean Accuracy**	**Physionet Score**
By [10]	SVM, BCTs, DACs	87.24% (+/- 2)	85.50% (+/- 3)
By [302]	Fuzzy Logic	87.78% (+/- 4)	80.09% (+/- 8)
Our Study 1	Dense Network, Backpropagation	87.70% (+/- 3)	75.35% (+/- 7)
Our Study 2	Dense NN, Standard DFO, Dynamic Δ, using Eq. 4.52	90.02% (+/- 3)	79.23% (+/- 5)
Our Study 3	Dense NN, Adapted DFO, Dynamic Δ, using Eq. 4.53	91.91% (+/- 4)	86.77% (+/- 4)
Our Study 4	Dense NN, Random Search	84.16% (+/- 3)	68.03% (+/- 5)
Our Study 5	Dense NN, Standard DFO, $\Delta = 0$ (i.e. no disturbance)	73.89% (+/- 2)	51.53% (+/- 4)
Our Study 6	Conv-Dense NN	88.21% (+/- 3)	76.34% (+/- 6)
Our Study 7	Conv-Dense NN, Adapted DFO, Dynamic Δ	91.88% (+/- 2)	86.81% (+/- 4)
Our Study 8	Conv-Dense NN, Random Search	75.63% (+/- 8)	52.37% (+/- 9)
Our Study 9	Exp 2, Backpropagation	91.98% (+/- 3)	80.02% (+/- 3)
Our Study 10	**Exp 2, Adapted DFO using Eq. 4.53**	**94.01% (+/- 3)**	**86.96% (+/- 2)**

Table 5.11: *Accuracy and Physionet score over five-fold cross validation for first and fifth rank in Physionet challenge 2015, NN optimised with backpropagation and adapted DFO algorithm with constant and dynamic disturbance threshold (Δ) along with results using generated samples.*

Otherwise multiple models should be implemented, which will include multiple generators and discriminators. This will make the training process complicated and cumbersome. In this experiment we used ECG leads II and V signals to train two sets of two generators G_{IIV} and G_{VII} and discriminators D_{IIV} and D_{VII} along with a denoising auto-encoder AED. Using this approach, we could impute the lead V from lead II with $MSE = 0.0007$ and lead II from lead V with $MSE = 0.0008$ (see table 5.6). By imputing the missing signals, we were able to improve the accuracy over models trained without this data augmentation strategy. The results are reported in table 5.11.

In our second experiment, we explored the possibility of imputing multiple missing signals of a sample using a model consisting of a single generator G, discriminator D and auto-encoder A. This model is able to impute lead II, V, $Bloodpressure$, $photo-plethysmogram$, and $Respiration$ signal with $MSE = 0.0011$, 0.0052, 0.03333, 0.0039, and 8765679. All the signals except respiration have been imputed with high similarity. The model was not able to find any relationship between the respiration rate and other bio-signals. Therefore, we removed the respiration signal in the feature selection classification performance process. Using the imputation method suggested in experiment 2, we were able to consider extra samples in our classification model (738).

To classify imputed samples, we have already explored multiple models and trained them using backpropagation and DFO approaches. Their calculated accuracy and Physionet scores are described in table 5.10. Using the DFO to find the optimal weights of the model, we recorded the highest $accuracy = 94.01$ and $Physionet = 86.96$. A complete comparison including our previous experiments and benchmarks are presented in table 5.11. By analysing the results gathered in table 5.10, we observe a significant reduction (44%) in the number of False Negatives while using the DFO algorithm in place of the backpropagation method.

5.8 Chapter Summary

In summary, we have presented a method for training neural networks based on neuroevolution, by utilising the DFO algorithm in a gradient-free, population-based scheme. We evaluated our approach to the problem of detecting false alarms in ICUs by using the Physionet dataset. The results obtained show that the proposed method outperforms (i) backpropagation-trained networks with the same or similar architecture, as well as (ii) the winning entries of the Physionet challenge. Additionally, we addressed the possibility of imputing missing signals while having two or five signals for each sample and later, the imputed signals using the five-signal method used for classification purpose and two approaches used for discovering the model's hyper-parameters. The results demonstrate a significant improvement while using an adapted DFO algorithm to train the model.

6. Conclusions and Further Directions

This thesis suggests a novel approach to impute missing bio-signals and discover optimal hyper-parameters for a deep neural network. Two approaches have been taken: **Approach1**, using only available samples and **Approach2** including samples with missing signal channels and imputing them using a GAN model. Both approaches culminated in the implementation of a deep neural network for classification and subsequent training using backpropagation and gradient-free algorithms. The results have been collected and evaluated using defined measures. This chapter revisits the contributions and research questions that have been explored and the experiments conducted to these ends. The last section addresses possible approaches to be taken forward from this point.

6.1 Thesis Summary and Contributions

In summary, we have presented a method for training neural networks based on neuroevolution, by utilising the DFO algorithm in a gradient-free, population-based scheme. We evaluated our approach to the problem of detecting false alarms in ECG monitoring systems by using the Physionet dataset. The results obtained show that the proposed method outperforms (i) backpropagation-trained networks with the same or similar architecture, as well as (ii) the winning entries of the Physionet challenge. We compared five-fold cross-validation of the first and fifth rank in the Physionet challenge 2015 with our designed NN models (see Table 4.4). The optimal weights of our models are calculated through both backpropagation and gradient-free algorithms. Model 1 consists of four dense layers and Model 2 consists of a convolution layer and four dense layers (see Table 4.2 & 4.3). As a benchmark, we calculated the accuracy and Physionet scores of [302] and [10]. They achieved accuracy and Physionet scores of (87.24%, 85.50%) and (87.78%, 80.09%) respectively. We investigated various implementations of DFO against Random Search as benchmark. The DFO's modified version for Model 1 & 2 achieved the highest score among the other results (see Table 4.4). Their accuracy and Physionet scores are (91.91%, 86.77%) and (91.88%, 86.81%) respectively. The behaviour of DFO, while having a constant Δ value of 0, and Random Search has also been investigated. Their accuracy and the Physionet scores are (73.89%, 51.53%) and (84.16, 68.03) respectively. The models optimised via neuroevolution with DFO outperform both (i) the networks trained by backpropagation, and (ii) the winning entries of the Physionet challenge.

In the second part of this study, we explored the possibility of imputing missing signals within a sample through two-signal and five-signal imputation approach. The Two-signal imputation approach is useful when we have only two signals for each sample. Otherwise, the multiple model should be implemented, which includes multiple generators and discriminators. However, this can make the training process complicated and cumbersome. In this experiment, we used ECG leads II and V signals to train two sets of two generators G_{IIV} and G_{VII} and discriminators D_{IIV} and D_{VII} along with a denoising auto-encoder AED. Using this approach, we could impute the lead V from lead II with $MSE = 0.0007$ and lead II from lead V with $MSE = 0.0008$ (see table 5.6). In our second experiment, we explored the possibility of imputing multiple missing signals of a sample using a model consisting of a single generator G, discriminator D and auto-encoder A. This model is able to impute lead II, V, $Blood\ pressure$, $Photo-plethysmogram$, and $Respiration$ signals with $MSE = 0.0011, 0.0052, 0.03333, 0.0039$, and 8765679. All the signals, with the exception of respiration were imputed with high similarity. The model was not able to find any relationship between the respiration rate and other bio-signals. For this reason, we removed the respiration signal in the feature selection classification performance process. Using the imputation method suggested in experiment 2, we were able to consider extra samples in our classification model (738). To classify imputed samples, we have already explored multiple models and trained them using backpropagation and DFO approaches. Their calculated accuracy and Physionet score are described in table 5.10. Using the DFO to find the optimal weights of the model, we recorded the highest $accuracy = 94.01$ and $Physionet = 86.96$. A complete comparison including our previous experiments and benchmarks are presented in table 5.11. By analysing the results gathered in table 5.10, we observe a significant reduction (44%) in the number of False Negatives by using DFO instead of a backpropagation method.

6.2 Future Directions

Moving forwards, there are several approaches with potential for further investigation:

- expand the gradient-free experiments to evaluate the possibility of having an algorithm that is able to suggest an optimal model including number and type of layers for a given dataset, while measuring the performance of separate populations for each network and optimising them separately.
- investigate the possibility of hybrid backpropagation and gradient-free optimisation methods which will include alternate training between them. Becoming trapped in a local minima is a major weakness of gradient-base algorithms. However, combining a stochastic approach with the above mentioned algorithm to tackle the challenges could possibly improve the overall performance. Although there has been some research suggesting a two stage model [287], it would be interesting to observe the behaviour of a model based on alternation between DFO and gradient-based models.
- recent years have seen significant improvements in embedded computing with offering higher processing power, nano form factor, and having the capability of running machine learning algorithms (TinyML) [310, 311]. Various low-footprint micro-controller devices such as Arduino [311], Raspberry Pi [312] and Nvidia Jetson Nano [313] are available for prototyping and testing concepts. Their applications in forecasting and anomaly detection, image recognition, and audio recognition are very well explored. However, there

is a research gap in using the generative adversarial models on the microcontroller with limited resources. Investigating the possibility of using the multi-lead generative model in sound generation, for instance, multi-channel audio-signal imputation used in hearing-aid devices, can be considered another expansion of this research.

Appendices

A List of Publications

Conference Papers and Book Chapters

1. H. M. J. Oroojeni, M. M. Al-Rifaie, and M. A. Nicolaou, "Deep neuroevolution: training deep neural networks for false alarm detection in intensive care units," in 2018 26th European Signal Processing Conference (EUSIPCO). IEEE, 2018, pp. 1157–1161.

   https://ieeexplore.ieee.org/document/8552944

2. H. M. J. Oroojeni, J. Oldfield, and M. A. Nicolaou, "Detecting early Parkinson's disease from keystroke dynamics using the tensor-train decomposition," in 2019 27th European Signal Processing Conference (EUSIPCO). IEEE, 2019, pp. 1–5.

   https://ieeexplore.ieee.org/document/8902562

3. M. M. al-Rifaie, H. M. J. Oroojeni, and M. Nicolaou, "Dispersive flies optimisation: Modifications and application," inSwarm Intelligence Algorithms.CRC Press, 2020, pp. 145–161.

   https://doi.org/10.1201/9780429422607

Bibliography

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017. 8, 24, 25

[2] M. M. al-Rifaie, "Dispersive flies optimisation: A tutorial," *Swarm Intelligence Algorithms*, pp. 135–147, 2020. 8, 49

[3] G. D. Clifford, I. Silva, B. Moody, Q. Li, D. Kella, A. Shahin, T. Kooistra, D. Perry, and R. G. Mark, "The physionet/computing in cardiology challenge 2015: reducing false arrhythmia alarms in the icu," in *Computing in Cardiology Conference (CinC), 2015.* IEEE, 2015, pp. 273–276. 8, 62, 63, 64, 67

[4] W. team. (2020) World health organisation cardiovascular diseases. [Online]. Available: www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1 11

[5] A. for the Advancement of Medical Instrumentation *et al.*, "Cardiac monitors, heart rate meters, and alarms," *American National Standard (ANSI/AAMI EC13: 2002) Arlington, VA*, pp. 1–87, 2002. 12, 62

[6] S. T. Lawless, "Crying wolf: false alarms in a pediatric intensive care unit." *Critical care medicine*, vol. 22, no. 6, pp. 981–985, 1994. 12, 62

[7] S. Parthasarathy and M. J. Tobin, "Sleep in the intensive care unit," *Intensive care medicine*, vol. 30, no. 2, pp. 197–206, 2004. 12, 62

[8] M.-C. Chambrin, "Alarms in the intensive care unit: how can the number of false alarms be reduced?" *Critical Care*, vol. 5, no. 4, p. 184, 2001. 12, 62

[9] C. L. Tsien and J. C. Fackler, "Poor prognosis for existing monitors in the intensive care unit," *Critical care medicine*, vol. 25, no. 4, pp. 614–619, 1997. 12, 62

[10] C. H. Antink and S. Leonhardt, "Reducing false arrhythmia alarms using robust interval estimation and machine learning," in *Computing in Cardiology Conference (CinC), 2015.* IEEE, 2015, pp. 285–288. 14, 64, 74, 76, 92, 95

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 17

[12] M. L. Minski and S. A. Papert, "Perceptrons: an introduction to computational geometry," *MA: MIT Press, Cambridge*, 1969. 17

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985. 17

[14] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, “Multi-column deep neural network for traffic sign classification,” *Neural networks*, vol. 32, pp. 333–338, 2012. 18

[15] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on.* IEEE, 2013, pp. 6645–6649. 18

[16] P. Seidel, A. Seidel, and O. Herbarth, “Multilayer perceptron tumour diagnosis based on chromatography analysis of urinary nucleosides,” *Neural Networks*, vol. 20, no. 5, pp. 646–651, 2007. 18

[17] H. Yan, Y. Jiang, J. Zheng, C. Peng, and Q. Li, “A multilayer perceptron-based medical decision support system for heart disease diagnosis,” *Expert Systems with Applications*, vol. 30, no. 2, pp. 272–281, 2006. 18

[18] D. G. Bounds, P. J. Lloyd, B. Mathew, and G. Waddell, “A multilayer perceptron network for the diagnosis of low back pain,” in *Proc. IEEE Int. Conf. on Neural Networks*, vol. 2, 1988, pp. 481–489. 18

[19] F. Piccialli, V. Di Somma, F. Giampaolo, S. Cuomo, and G. Fortino, “A survey on deep learning in medicine: Why, how and when?” *Information Fusion*, vol. 66, pp. 111–137, 2021. 18

[20] P. Strata and R. Harvey, “Dale’s principle,” *Brain research bulletin*, vol. 50, no. 5-6, pp. 349–350, 1999. 18

[21] C. Parisien, C. H. Anderson, and C. Eliasmith, “Solving the problem of negative synaptic weights in cortical models,” *Neural computation*, vol. 20, no. 6, pp. 1473–1494, 2008. 18

[22] S. L. Hill, Y. Wang, I. Riachi, F. Schürmann, and H. Markram, “Statistical connectivity provides a sufficient foundation for specific functional connectivity in neocortical neural microcircuits,” *Proceedings of the National Academy of Sciences*, vol. 109, no. 42, pp. E2885–E2894, 2012. 18

[23] J. Schmidhuber and S. Hochreiter, “Long short-term memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997. 19

[24] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2009. 19

[25] C. Ferreira, “Designing neural networks using gene expression programming,” in *Applied soft computing technologies: The challenge of complexity.* Springer, 2006, pp. 517–535. 19

[26] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012. 20

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 20

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986. 20, 22

[29] L. Chen and K. Aihara, "Chaotic simulated annealing by a neural network model with transient chaos," *Neural networks*, vol. 8, no. 6, pp. 915–930, 1995. 20

[30] H. Ghimatgar, K. Kazemi, M. S. Helfroush, K. Pillay, A. Dereymaker, K. Jansen, M. De Vos, and A. Aarabi, "Neonatal eeg sleep stage classification based on deep learning and hmm," *Journal of Neural Engineering*, vol. 17, no. 3, p. 036031, 2020. 20

[31] A. Gong, C. Chen, and M. Peng, "Human interaction recognition based on deep learning and hmm," *IEEE Access*, vol. 7, pp. 161 123–161 130, 2019. 20

[32] S.-A. Grönroos, S. Virpioja, P. Smit, and M. Kurimo, "Morfessor flatcat: An hmm-based method for unsupervised and semi-supervised learning of morphology," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 1177–1185. 20

[33] B. A. Olshausen and D. J. Field, "Natural image statistics and efficient coding," *Network: computation in neural systems*, vol. 7, no. 2, pp. 333–339, 1996. 21

[34] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," *PLoS computational biology*, vol. 9, no. 4, p. e1003037, 2013. 21

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013. 21

[36] H. Z. Shouval, S. S.-H. Wang, and G. M. Wittenberg, "Spike timing dependent plasticity: a consequence of more fundamental learning rules," *Frontiers in Computational Neuroscience*, vol. 4, p. 19, 2010. 21

[37] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural networks*, vol. 1, no. 4, pp. 295–307, 1988. 21

[38] R. Chaudhary, H. Patel, and M. Scholar, "A survey on backpropagation algorithm for neural networks," *International Journal for Technological Research in Engineering*, vol. 2, 2015. 21, 23

[39] S. Solanki and H. Jethva, "Modified back propagation algorithm of feed forward networks," *International Journal of Innovative Technology and Exploring Engineering*, vol. 2, no. 6, pp. 131–134, 2013. 22

[40] M.-Y. Chow, P. Goode, A. Menozzi, J. Teeter, and J. Thrower, "Bernoulli error measure approach to train feedforward artificial neural networks for classification problems," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 1. IEEE, 1994, pp. 44–49. 23

[41] M. Rimer and T. Martinez, "Classification-based objective functions," *Machine Learning*, vol. 63, no. 2, pp. 183–205, 2006. 23

[42] Z.-G. Che, T.-A. Chiang, Z.-H. Che *et al.*, "Feed-forward neural networks training: a comparison between genetic algorithm and back-propagation learning algorithm," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 10, pp. 5839–5850, 2011. 23

[43] C. U. Joy, "Comparing the performance of backpropagation algorithm and genetic algorithms in pattern recognition problems," *International Journal of Computer Information Systems*, vol. 2, no. 5, pp. 7–52, 2011. 23

[44] B. Choi, J.-H. Lee, and D.-H. Kim, "Solving local minima problem with large number of hidden nodes on two-layered feed-forward artificial neural networks," *Neurocomputing*, vol. 71, no. 16-18, pp. 3640–3643, 2008. 23

[45] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006. 24

[46] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2012. 24

[47] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598. 24

[48] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015. 24

[49] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014. 24

[50] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 1, 2015. 24

[51] N. Zeng, Z. Wang, H. Zhang, W. Liu, and F. E. Alsaadi, "Deep belief networks for quantitative analysis of a gold immunochromatographic strip," *Cognitive Computation*, vol. 8, no. 4, pp. 684–692, 2016. 24

[52] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989. 24

[53] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1. 25

[54] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 25, 29

[55] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *ICDAR*, vol. 3, 2003, pp. 958–962. 25

[56] Q. Dou, H. Chen, L. Yu, L. Zhao, J. Qin, D. Wang, V. C. Mok, L. Shi, and P.-A. Heng, "Automatic detection of cerebral microbleeds from mr images via 3d convolutional neural networks," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1182–1195, 2016. 25

[57] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, "Lung pattern classification for interstitial lung diseases using a deep convolutional neural network," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1207–1216, 2016. 25

[58] P.-P. Ypsilantis, M. Siddique, H.-M. Sohn, A. Davies, G. Cook, V. Goh, and G. Montana, "Predicting response to neoadjuvant chemotherapy with pet imaging using convolutional neural networks," *PloS one*, vol. 10, no. 9, p. e0137036, 2015. 25

[59] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," in *NIPS*, 2016. 27

[60] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016. 27

[61] X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV).* IEEE, 2017, pp. 1133–1141. 27

[62] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 215–10 224. 27

[63] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," *CoRR*, vol. abs/1811.00002, 2018. [Online]. Available: http://arxiv.org/abs/1811.00002 28

[64] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, p. 100004, 2021. 28

[65] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. 28, 29, 30, 31

[66] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016, pp. 469–477. 28

[67] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242. 28

[68] M. O. Turkoglu, W. Thong, L. Spreeuwers, and B. Kicanaoglu, "A layer-based sequential framework for scene generation with gans," *arXiv preprint arXiv:1902.00671*, 2019. 28

[69] W. Zhu, X. Xiang, T. D. Tran, and X. Xie, "Adversarial deep structural networks for mammographic mass segmentation," *arXiv preprint arXiv:1612.05970*, 2016. 28

[70] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *arXiv preprint arXiv:1611.08408*, 2016. 28

[71] H. Dong, S. Yu, C. Wu, and Y. Guo, "Semantic image synthesis via adversarial learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5706–5714. 28

[72] Z. Qiu, Y. Pan, T. Yao, and T. Mei, "Deep semantic hashing with generative adversarial networks," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 225–234. 28

[73] N. Souly, C. Spampinato, and M. Shah, "Semi supervised semantic segmentation using generative adversarial network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5688–5696. 28

[74] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," *arXiv preprint arXiv:1505.03906*, 2015. 28

[75] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool, "Pose guided person image generation," in *Advances in Neural Information Processing Systems*, 2017, pp. 406–416. 28

[76] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances In Neural Information Processing Systems*, 2016, pp. 613–621. 28

[77] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6721–6729. 28

[78] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2642–2651. 28

[79] C. Lassner, G. Pons-Moll, and P. V. Gehler, "A generative model of people in clothing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 853–862. 28

[80] W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: better text generation via filling in the_," *arXiv preprint arXiv:1801.07736*, 2018. 28

[81] Z. Yang, J. Hu, R. Salakhutdinov, and W. W. Cohen, "Semi-supervised qa with generative domain-adaptive nets," *arXiv preprint arXiv:1702.02206*, 2017. 28

[82] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov, "Good semi-supervised learning that requires a bad gan," in *Advances in neural information processing systems*, 2017, pp. 6510–6520. 28

[83] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," *arXiv preprint arXiv:1611.08207*, 2016. 28

[84] C. Donahue, J. McAuley, and M. Puckette, "Synthesizing audio with generative adversarial networks," *arXiv preprint arXiv:1802.04208*, vol. 1, 2018. 28

[85] K. G. Hartmann, R. T. Schirrmeister, and T. Ball, "Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals," *arXiv preprint arXiv:1806.01875*, 2018. 28

[86] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017. 28

[87] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks.* Springer, 2019, pp. 703–716. 28

[88] E. Brophy, Z. Wang, and T. E. Ward, "Quick and easy time series generation with established image-based gans," *arXiv preprint arXiv:1902.05624*, 2019. 28

[89] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks: A survey and taxonomy," *arXiv preprint arXiv:1906.01529*, 2019. 28, 29

[90] J. Kossaifi, L. Tran, Y. Panagakis, and M. Pantic, "Gagan: Geometry-aware generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 878–887. 28

[91] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 28

[92] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017. 28

[93] Y. Li, A. Schwing, K.-C. Wang, and R. Zemel, "Dualing gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5606–5616. 28

[94] A. Borji, “Pros and cons of gan evaluation measures,” *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, 2019. 28

[95] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger, “An empirical study on evaluation metrics of generative adversarial networks,” *arXiv preprint arXiv:1806.07755*, 2018. 28

[96] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Advances in neural information processing systems*, 2017, pp. 5767–5777. 28

[97] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637. 28

[98] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012. 28

[99] Z. Wang, G. Healy, A. F. Smeaton, and T. E. Ward, “Use of neural signals to evaluate the quality of generative adversarial network performance in facial image generation,” *arXiv preprint arXiv:1811.04172*, 2018. 28

[100] Z. Wang, Q. She, A. F. Smeaton, T. E. Ward, and G. Healy, “Neuroscore: A brain-inspired evaluation metric for generative adversarial networks,” *arXiv preprint arXiv:1905.04243*, 2019. 28

[101] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” *arXiv preprint arXiv:1605.05396*, 2016. 29

[102] S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Globally and locally consistent image completion,” *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 107, 2017. 29

[103] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232. 29, 78, 80, 81

[104] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690. 29

[105] J. M. Susskind, A. K. Anderson, and G. E. Hinton, “The toronto face database,” *Department of Computer Science, University of Toronto, Toronto, ON, Canada, Tech. Rep*, vol. 3, 2010. 29

[106] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009. 29

[107] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494. 29

[108] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on communications*, vol. 31, no. 4, pp. 532–540, 1983. 29

[109] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015. 29

[110] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833. 29

[111] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016. 29

[112] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017. 30, 39

[113] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410. 30

[114] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018. 30

[115] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 30

[116] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018. 30

[117] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017. 31

[118] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017. 33, 38

[119] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000. 33

[120] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777. 34, 38

[121] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802. 34

[122] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in neural information processing systems*, 2016, pp. 271–279. 34

[123] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016. 35

[124] G.-J. Qi, "Loss-sensitive generative adversarial networks on lipschitz densities," *arXiv preprint arXiv:1701.06264*, 2017. 36, 38

[125] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016. 36

[126] J. H. Lim and J. C. Ye, "Geometric gan," *arXiv preprint arXiv:1705.02894*, 2017. 37

[127] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018. 37

[128] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet, "On integral probability metrics,\phi-divergences and binary classification," *arXiv preprint arXiv:0901.2698*, 2009. 37

[129] A. Müller, "Integral probability metrics and their generating classes of functions," *Advances in Applied Probability*, vol. 29, no. 2, pp. 429–443, 1997. 37

[130] T. Donchev and E. Farkhi, "Stability and euler approximation of one-sided lipschitz differential inclusions," *SIAM journal on control and optimization*, vol. 36, no. 2, pp. 780–796, 1998. 38

[131] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives," *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966. 38

[132] A. Goldstein, "Optimization of lipschitz continuous functions," *Mathematical Programming*, vol. 13, no. 1, pp. 14–22, 1977. 38

[133] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104. 38

[134] I.-J. Kim and X. Xie, "Handwritten hangul recognition using deep convolutional neural networks," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 1, pp. 1–13, 2015. 38

[135] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, pp. 14 680–14 707, 2015. 38

[136] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016. 38

[137] R. Manor and A. B. Geva, “Convolutional neural network for multi-category rapid serial visual presentation bci,” *Frontiers in computational neuroscience*, vol. 9, p. 146, 2015. 38

[138] X. Glorot, A. Bordes, and Y. Bengio, “Domain adaptation for large-scale sentiment classification: A deep learning approach,” in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520. 38

[139] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167. 38

[140] N. D. Lane and P. Georgiev, “Can deep learning revolutionize mobile sensing?” in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 117–122. 38

[141] J. Wang, X. Yang, H. Cai, W. Tan, C. Jin, and L. Li, “Discrimination of breast cancer with microcalcifications on mammography by deep learning,” *Scientific reports*, vol. 6, p. 27327, 2016. 38

[142] Z. Yan, Y. Zhan, Z. Peng, S. Liao, Y. Shinagawa, S. Zhang, D. N. Metaxas, and X. S. Zhou, “Multi-instance deep learning: Discover discriminative local anatomies for bodypart recognition,” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1332–1343, 2016. 39

[143] H.-I. Suk, S.-W. Lee, D. Shen, A. D. N. Initiative *et al.*, “Hierarchical feature representation and multimodal fusion with deep learning for ad/mci diagnosis,” *NeuroImage*, vol. 101, pp. 569–582, 2014. 39

[144] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, “A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2013, pp. 403–410. 39

[145] M. A. Haidar and M. Rezagholizadeh, “Textkd-gan: Text generation using knowledge distillation and generative adversarial networks,” in *Canadian Conference on Artificial Intelligence*. Springer, 2019, pp. 107–118. 39

[146] A. Noguchi and T. Harada, “Image generation from small datasets via batch statistics adaptation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2750–2758. 39

[147] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in neural information processing systems*, 2017, pp. 6626–6637. 39

[148] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076. 39

[149] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7167–7176. 39

[150] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring gans: generating images from limited data," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 218–234. 39

[151] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3722–3731. 39

[152] T. Miyato and M. Koyama, "cgans with projection discriminator," *arXiv preprint arXiv:1802.05637*, 2018. 39

[153] S. Aigner and M. Körner, "Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans," *arXiv preprint arXiv:1810.01325*, 2018. 39

[154] M. I. Belghazi, S. Rajeswar, O. Mastropietro, N. Rostamzadeh, J. Mitrovic, and A. Courville, "Hierarchical adversarially learned inference," *arXiv preprint arXiv:1802.01071*, 2018. 39

[155] Y. Frégier and J.-B. Gouray, "Mind2mind: transfer learning for gans," *arXiv preprint arXiv:1906.11613*, 2019. 39

[156] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016. 40

[157] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *arXiv preprint arXiv:1412.4446*, 2014. 40

[158] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Domain adaptation with randomized multilinear adversarial networks," *ArXiv*, vol. abs/1705.10667, 2017. 40

[159] Z. Luo, Y. Zou, J. Hoffman, and L. F. Fei-Fei, "Label efficient learning of transferable representations acrosss domains and tasks," in *Advances in Neural Information Processing Systems*, 2017, pp. 165–177. 40

[160] R. J. Martis, U. R. Acharya, C. M. Lim, K. Mandana, A. K. Ray, and C. Chakraborty, "Application of higher order cumulant features for cardiac health diagnosis using ecg signals," *International journal of neural systems*, vol. 23, no. 04, p. 1350014, 2013. 40

[161] İ. Güler and E. D. Übeylı, "Ecg beat classifier designed by combined neural network model," *Pattern recognition*, vol. 38, no. 2, pp. 199–208, 2005. 40

[162] C. Kamath, "Ecg beat classification using features extracted from teager energy functions in time and frequency domains," *IET signal processing*, vol. 5, no. 6, pp. 575–581, 2011. 40

[163] O. T. Inan, L. Giovangrandi, and G. T. Kovacs, “Robust neural-network-based classification of premature ventricular contractions using wavelet transform and timing interval features,” *IEEE transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2507–2515, 2006. 40

[164] S. Banerjee and M. Mitra, “Ecg beat classification based on discrete wavelet transformation and nearest neighbour classifier,” *Journal of medical engineering & technology*, vol. 37, no. 4, pp. 264–272, 2013. 40

[165] F. A. Elhaj, N. Salim, A. R. Harris, T. T. Swee, and T. Ahmed, “Arrhythmia recognition and classification using combined linear and nonlinear features of ecg signals,” *Computer methods and programs in biomedicine*, vol. 127, pp. 52–63, 2016. 40

[166] J. Park and K. Kang, “Pchd: Personalized classification of heartbeat types using a decision tree,” *Computers in biology and medicine*, vol. 54, pp. 79–88, 2014. 40

[167] R. J. Martis, U. R. Acharya, and L. C. Min, “Ecg beat classification using pca, lda, ica and discrete wavelet transform,” *Biomedical Signal Processing and Control*, vol. 8, no. 5, pp. 437–448, 2013. 40

[168] M. T. Nguyen, A. Shahzad, B. Van Nguyen, and K. Kim, “Diagnosis of shockable rhythms for automated external defibrillators using a reliable support vector machine classifier,” *Biomedical Signal Processing and Control*, vol. 44, pp. 258–269, 2018. 40

[169] S. Raj and K. C. Ray, “Automated recognition of cardiac arrhythmias using sparse decomposition over composite dictionary,” *Computer methods and programs in biomedicine*, vol. 165, pp. 175–186, 2018. 40

[170] W. Yang, Y. Si, D. Wang, and B. Guo, “Automatic recognition of arrhythmia based on principal component analysis network and linear support vector machine,” *Computers in biology and medicine*, vol. 101, pp. 22–32, 2018. 40

[171] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, “Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats,” *Computers in biology and medicine*, vol. 102, pp. 278–287, 2018. 40

[172] G. Sannino and G. De Pietro, “A deep learning approach for ecg-based heartbeat classification for arrhythmia detection,” *Future Generation Computer Systems*, vol. 86, pp. 446–455, 2018. 40

[173] A. Isin and S. Ozdalili, “Cardiac arrhythmia detection using deep learning,” *Procedia computer science*, vol. 120, pp. 268–275, 2017. 40

[174] S. Raj, K. C. Ray, and O. Shankar, “Cardiac arrhythmia beat classification using dost and pso tuned svm,” *Computer methods and programs in biomedicine*, vol. 136, pp. 163–177, 2016. 40

[175] S. M. Mathews, C. Kambhamettu, and K. E. Barner, "A novel application of deep learning for single-lead ecg classification," *Computers in biology and medicine*, vol. 99, pp. 53–62, 2018. 40

[176] M. Llamedo and J. P. Martínez, "An automatic patient-adapted ecg heartbeat classifier allowing expert assistance," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, pp. 2312–2320, 2012. 40

[177] Z.-E. H. Slimane and A. Naït-Ali, "Qrs complex detection using empirical mode decomposition," *Digital Signal Processing*, vol. 20, no. 4, pp. 1221–1228, 2010. 40

[178] S. Shadmand and B. Mashoufi, "A new personalized ecg signal classification algorithm using block-based neural network and particle swarm optimization," *Biomedical Signal Processing and Control*, vol. 25, pp. 12–23, 2016. 40

[179] A. Beznosikov, A. Sadiev, and A. Gasnikov, "Gradient-free methods for saddle-point problem," *arXiv preprint arXiv:2005.05913*, 2020. 41

[180] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989. 41

[181] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. IV. Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948. 41, 42

[182] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006. 42

[183] M. Derigo and T. Stutzle, *Ant Colony Optimization*. The MIT Press, 2004. 42

[184] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996. 42

[185] T. Stutzle and H. Hoos, "Max-min ant system and local search for the traveling salesman problem," in *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 1997, pp. 309–314. 42

[186] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997. 42

[187] M. Guntsch and M. Middendorf, "Applying population based aco to dynamic optimization problems," in *International Workshop on Ant Algorithms*. Springer, 2002, pp. 111–122. 42

[188] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks." *American Association for the Advancement of Science, Washington, DC(USA).*, 1990. 43

[189] D. Bratton and J. Kennedy, “Defining a standard for particle swarm optimization,” in *Proc of the Swarm Intelligence Symposium.* Honolulu, Hawaii, USA: IEEE, 2007, pp. 120–127. 43

[190] D. Mandal, A. Chatterjee, and M. Maitra, “Robust medical image segmentation using particle swarm optimization aided level set based global fitting energy active contour approach,” *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 199–214, 2014. 43

[191] D. Wu, K. Warwick, Z. Ma, M. N. Gasson, J. G. Burgess, S. Pan, and T. Z. Aziz, “Prediction of parkinson’s disease tremor onset using a radial basis function neural network based on particle swarm optimization,” *International journal of neural systems*, vol. 20, no. 02, pp. 109–116, 2010. 43

[192] R. C. Eberhart and X. Hu, “Human tremor analysis using particle swarm optimization,” in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999, pp. 1927–1930. 43

[193] R. Storn and K. Price, “Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces,” 1995, tR-95-012, [online]. Available: http://www.icsi.berkeley.edu/ storn/litera.html. 44

[194] R. Thomsen, “Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids,” *Biosystems*, vol. 72, no. 1-2, pp. 57–73, 2003. 45

[195] J. Vesterstrom and R. Thomsen, “A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems,” in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, 2004, pp. 1980–1987. 45, 46

[196] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation.* IOP Publishing Ltd., 1997. 45

[197] T. K. Paul and H. Iba, “Prediction of cancer class with majority voting genetic programming classifier using gene expression data,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 6, no. 2, pp. 353–367, 2009. 46

[198] H. Guo and A. K. Nandi, “Breast cancer diagnosis using genetic programming generated feature,” *Pattern Recognition*, vol. 39, no. 5, pp. 980–987, 2006. 46

[199] A. P. Mitra, A. A. Almal, B. George, D. W. Fry, P. F. Lenehan, V. Pagliarulo, R. J. Cote, R. H. Datar, and W. P. Worzel, “The use of genetic programming in the analysis of quantitative gene expression profiles for identification of nodal status in bladder cancer,” *BMC cancer*, vol. 6, no. 1, p. 159, 2006. 46

[200] M. M. al-Rifaie, “Dispersive flies optimisation,” in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 2. IEEE, 2014, pp. pages 529–538. 47, 67, 70

[201] M. M. al-Rifaie and A. Aber, "Dispersive flies optimisation and medical imaging," in *Recent Advances in Computational Optimization*. Springer, 2016, pp. 183–203. 47

[202] B. Lazov and T. Vetsov, "Sum of three cubes via optimisation," *arXiv preprint arXiv:2005.09710*, 2020. 47

[203] B. B. Acharya, S. Dhakal, A. Bhattarai, and N. Bhattarai, "PID speed control of dc motor using meta-heuristic algorithms," *Int J Pow Elec & Dri Syst ISSN*, vol. 2088, no. 8694, pp. 86–94. 47

[204] H. Alhakbani, "Handling class imbalance using swarm intelligence techniques, hybrid data and algorithmic level solutions," Ph.D. dissertation, Goldsmiths, University of London, London, United Kingdom, 2018. 47

[205] H. A. Alhakbani and M. M. al Rifaie, "Optimising svm to classify imbalanced data using dispersive flies optimisation," in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2017, pp. 399–402. 47, 55

[206] H. Oroojeni, M. M. al-Rifaie, and M. A. Nicolaou, "Deep neuroevolution: Training deep neural networks for false alarm detection in intensive care units," in *European Association for Signal Processing (EUSIPCO) 2018*. IEEE, 2018, pp. 1157–1161. 47

[207] M. M. al-Rifaie, A. Ursyn, R. Zimmer, and M. A. J. Javid, "On symmetry, aesthetics and quantifying symmetrical complexity," in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2017, pp. 17–32. 47

[208] P. Aparajeya, F. F. Leymarie, and M. M. al-Rifaie, "Swarm-based identification of animation key points from 2d-medialness maps," in *Computational Intelligence in Music, Sound, Art and Design*, A. Ekárt, A. Liapis, and M. L. Castro Pena, Eds. Cham: Springer International Publishing, 2019, pp. 69–83. 47

[209] M. M. al-Rifaie and M. Cavazza, "Beer organoleptic optimisation: Utilising swarm intelligence and evolutionary computation methods," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 255–256. 47

[210] M. M. al-Rifaie and M. Cavazza, "Beer Organoleptic Optimisation: Utilising Swarm Intelligence and Evolutionary Computation Methods," *arXiv e-prints*, p. arXiv:2004.03438, Apr. 2020. 47

[211] M. M. al-Rifaie, F. F. Leymarie, W. Latham, and M. Bishop, "Swarmic autopoiesis and computational creativity," *Connection Science*, pp. 1–19, 2017. [Online]. Available: http://dx.doi.org/10.1080/09540091.2016.1274960 47

[212] M. M. al-Rifaie, “Perceived simplicity and complexity in nature,” in *AISB 2017: Computational Architectures for Animal Cognition*, University of Bath, Bath, U.K., 2017, pp. 299–305. 47

[213] M. M. al-Rifaie, “Investigating knowledge-based exploration-exploitation balance in a minimalist swarm optimiser,” in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 2273–2280. 47

[214] M. M. al-Rifaie, “Exploration and exploitation zones in a minimalist swarm optimiser,” *Entropy*, vol. 23, no. 8, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/8/977 47

[215] M. M. al Rifaie and M. Cavazza, “Beer organoleptic optimisation: Utilising swarm intelligence and evolutionary computation methods,” *arXiv preprint arXiv:2004.03438*, 2020. 49

[216] S. Fong, R. Wong, and A. V. Vasilakos, “Accelerated pso swarm search feature selection for data stream mining big data,” *IEEE transactions on services computing*, vol. 9, no. 1, pp. 33–45, 2016. 50

[217] S. Cheng, Q. Zhang, and Q. Qin, “Big data analytics with swarm intelligence,” *Industrial Management & Data Systems*, vol. 116, no. 4, pp. 646–666, 2016. 50

[218] M. H. Ryalat, D. Emmens, M. Hulse, D. Bell, Z. Al-Rahamneh, S. Laycock, and M. Fisher, “Evaluation of particle swarm optimisation for medical image segmentation,” in *International Conference on Systems Science*. Springer, 2016, pp. 61–72. 50

[219] M. M. al Rifaie, A. Aber, R. Sayers, E. Choke, and M. Bown, “Deploying swarm intelligence in medical imaging,” in *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 14–21. 50

[220] G. P. Rangaiah, *Multi-objective optimization: techniques and applications in chemical engineering*. World Scientific, 2016, vol. 5. 50

[221] R. V. Devi, S. S. Sathya, and M. S. Coumar, “Evolutionary algorithms for de novo drug design–a survey,” *Applied Soft Computing*, vol. 27, pp. 543–552, 2015. 50

[222] N. K. Pareek and V. Patidar, “Medical image protection using genetic algorithm operations,” *Soft Computing*, vol. 20, no. 2, pp. 763–772, 2016. 50

[223] P. Ghamisi, M. S. Couceiro, F. M. Martins, and J. A. Benediktsson, “Multilevel image segmentation based on fractional-order darwinian particle swarm optimization,” *IEEE Transactions on Geoscience and Remote sensing*, vol. 52, no. 5, pp. 2382–2394, 2014. 50

[224] B. Bošković and J. Brest, “Genetic algorithm with advanced mechanisms applied to the protein structure prediction in a hydrophobic-polar model and cubic lattice,” *Applied Soft Computing*, vol. 45, pp. 61–70, 2016. 51

[225] F. L. Custódio, H. J. Barbosa, and L. E. Dardenne, "A multiple minima genetic algorithm for protein structure prediction," *Applied Soft Computing*, vol. 15, pp. 88–99, 2014. 51

[226] R. Shukla, D. Ray, K. Sarkar, M. Kumar Dixit, and S. Prasad Bhattacharyya, "Flying onto global minima on potential energy surfaces: A swarm intelligence guided route to molecular electronic structure," *International Journal of Quantum Chemistry*, vol. 117, no. 5, 2017. 51

[227] L. B. Vilhelmsen and B. Hammer, "A genetic algorithm for first principles global structure optimization of supported nano structures," *The Journal of chemical physics*, vol. 141, no. 4, p. 044711, 2014. 51

[228] J. Lee, S. Hong, and J.-H. Lee, "An efficient prediction for heavy rain from big weather data using genetic algorithm," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication.* ACM, 2014, p. 25. 51

[229] E. Asadi, M. G. da Silva, C. H. Antunes, L. Dias, and L. Glicksman, "Multi-objective optimization for building retrofit: A model using genetic algorithm and artificial neural network and an application," *Energy and Buildings*, vol. 81, pp. 444–456, 2014. 51

[230] T. Nguyen, K. Ghabraie, and T. Tran-Cong, "Applying bi-directional evolutionary structural optimisation method for tunnel reinforcement design considering nonlinear material behaviour," *Computers and Geotechnics*, vol. 55, pp. 57–66, 2014. 51

[231] M. Salucci, L. Poli, N. Anselmi, and A. Massa, "Multifrequency particle swarm optimization for enhanced multiresolution gpr microwave imaging," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 3, pp. 1305–1317, 2017. 51

[232] V. A. Barbosa, R. R. Ribeiro, A. R. Feitosa, V. L. Silva, A. D. Rocha, R. C. Freitas, R. E. Souza, and W. P. Santos, "Reconstruction of electrical impedance tomography using fish school search, non-blind search, and genetic algorithm," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 8, no. 2, pp. 17–33, 2017. 52

[233] H. Paasche and J. Tronicke, "Nonlinear joint inversion of tomographic data using swarm intelligence," *Geophysics*, vol. 79, no. 4, pp. R133–R149, 2014. 52

[234] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016. 52

[235] P. A. Vargas, E. A. Di Paolo, I. Harvey, and P. Husbands, *The horizons of evolutionary robotics.* MIT Press, 2014. 52

[236] J. Na, K. S. Kshetrimayum, U. Lee, and C. Han, "Multi-objective optimization of microchannel reactor for fischer-tropsch synthesis using computational fluid dynamics and genetic algorithm," *Chemical Engineering Journal*, vol. 313, pp. 1521–1534, 2017. 52

[237] H. Yu, G. Janiga, and D. Thévenin, "Computational fluid dynamics-based design optimization method for archimedes screw blood pumps," *Artificial organs*, vol. 40, no. 4, pp. 341–352, 2016. 52

[238] R. Das, "Parameter estimation of a space radiator using differential evolution algorithm," in *Contemporary Computing (IC3), 2016 Ninth International Conference on*. IEEE, 2016, pp. 1–6. 52

[239] C. Iacopino, P. Palmer, N. Policella, A. Donati, and A. Brewer, "How ants can manage your satellites," *Acta Futura*, vol. 9, pp. 57–70, 2014. 52

[240] X. Liu, H. An, L. Wang, and X. Jia, "An integrated approach to optimize moving average rules in the eua futures market based on particle swarm optimization and genetic algorithms," *Applied Energy*, vol. 185, pp. 1778–1787, 2017. 52

[241] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 321–344, 2013. 53

[242] C. Aranha, R. Tanabe, R. Chassagne, and A. Fukunaga, "Optimization of oil reservoir models using tuned evolutionary algorithms and adaptive differential evolution," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 877–884. 53

[243] E. Yasari, M. R. Pishvaie, F. Khorasheh, K. Salahshoor, and R. Kharrat, "Application of multi-criterion robust optimization in water-flooding of oil reservoir," *Journal of Petroleum Science and Engineering*, vol. 109, pp. 1–11, 2013. 53

[244] N. Delgarm, B. Sajadi, F. Kowsary, and S. Delgarm, "Multi-objective optimization of the building energy performance: A simulation-based approach by means of particle swarm optimization (pso)," *Applied Energy*, vol. 170, pp. 293–303, 2016. 53

[245] Q. Kang, M. Zhou, J. An, and Q. Wu, "Swarm intelligence approaches to optimal power flow problem with distributed generator failures in power networks," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 343–353, 2013. 53

[246] G. Kanagaraj, S. Ponnambalam, N. Jawahar, and J. M. Nilakantan, "An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization," *Engineering Optimization*, vol. 46, no. 10, pp. 1331–1351, 2014. 53

[247] H. Garg, "Solving structural engineering design optimization problems using an artificial bee colony algorithm," *J Ind Manag Optim*, vol. 10, no. 3, pp. 777–794, 2014. 53

[248] I. Soesanti and R. Syahputra, "Batik production process optimization using particle swarm optimization method," *Journal of Theoretical and Applied Information Technology*, vol. 86, no. 2, p. 272, 2016. 53

[249] A. R. Yildiz, "A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing," *Applied Soft Computing*, vol. 13, no. 5, pp. 2906–2912, 2013. 53

[250] D. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Computers in Industry*, vol. 81, pp. 82–95, 2016. 54

[251] R. Zhang and R. Chiong, "Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," *Journal of Cleaner Production*, vol. 112, pp. 3361–3375, 2016. 54

[252] B. Yao, B. Yu, P. Hu, J. Gao, and M. Zhang, "An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot," *Annals of Operations Research*, vol. 242, no. 2, pp. 303–320, 2016. 54

[253] S. Karakatič and V. Podgorelec, "A survey of genetic algorithms for solving multi depot vehicle routing problem," *Applied Soft Computing*, vol. 27, pp. 519–532, 2015. 54

[254] Z. Liu, B. Zhang, Q. Feng, Z. Chen, C. Lin, and Y. Ding, "Focusing light through strongly scattering media by a controlling binary amplitude optimization using genetic algorithm," in *Fifth International Conference on Optical and Photonics Engineering*. International Society for Optics and Photonics, 2017, pp. 1 044 927–1 044 927. 54

[255] C.-T. Hsieh, H.-T. Yau, C.-C. Wang, and Y.-S. Hsieh, "Particle swarm optimization used with proportional–derivative control to analyze nonlinear behavior in the atomic force microscope," *Advances in Mechanical Engineering*, vol. 8, no. 9, p. 1687814016667271, 2016. 54

[256] Q. Wu, C. Cole, and T. McSweeney, "Applications of particle swarm optimization in the railway domain," *International Journal of Rail Transportation*, vol. 4, no. 3, pp. 167–190, 2016. 54

[257] W. ShangGuan, X.-H. Yan, B.-G. Cai, and J. Wang, "Multiobjective optimization for train speed trajectory in ctcs high-speed railway with hybrid evolutionary algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2215–2225, 2015. 54

[258] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937. 55

[259] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017. 55, 59

[260] H. Alhakbani and M. M. al Rifaie, "Feature selection using stochastic diffusion search," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 385–392. 55

[261] H. A. Alhakbani and M. M. Al-Rifaie, "Exploring feature-level duplications on imbalanced data using stochastic diffusion search," in *Multi-Agent Systems and Agreement Technologies*. Springer, 2016, pp. 305–313. 56

[262] H. A. Alhakbani and M. M. al Rifaie, "A swarm intelligence approach in undersampling majority class," in *International Conference on Swarm Intelligence*. Springer, 2016, pp. 225–232. 56

[263] M. M. Al-Rifaie and H. A. Alhakbani, "Handling class imbalance in direct marketing dataset using a hybrid data and algorithmic level solutions," in *SAI Computing Conference (SAI), 2016*. IEEE, 2016, pp. 446–451. 56

[264] N. Kayarvizhy, S. Kanmani, and R. Uthariaraj, "Ann models optimized using swarm intelligence algorithms," *WSEAS Transactions on Computers*, vol. 13, no. 45, pp. 501–519, 2014. 56

[265] I. Rechenberg, "Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution," Ph.D. dissertation, Technical University of Berlin, Department of Process Engineering, 1971. 57

[266] H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie*. Birkhäuser, 1977. 57

[267] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017. 57, 58, 59

[268] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001. 57

[269] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural evolution strategies," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 3381–3387. 57

[270] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014. 57

[271] S. Yi, D. Wierstra, T. Schaul, and J. Schmidhuber, "Stochastic search using the natural gradient," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1161–1168. 57

[272] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber, "Efficient natural evolution strategies," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 539–546. 57

[273] T. Glasmachers, T. Schaul, and J. Schmidhuber, “A natural evolution strategy for multi-objective optimization,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2010, pp. 627–636. 57

[274] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber, “Exponential natural evolution strategies,” in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 393–400. 57

[275] T. Schaul, T. Glasmachers, and J. Schmidhuber, “High dimensions and heavy tails for natural evolution strategies,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 845–852. 57

[276] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, “Parameter-exploring policy gradients,” *Neural Networks*, vol. 23, no. 4, pp. 551–559, 2010. 57, 58

[277] Y. Nesterov and V. Spokoiny, “Random gradient-free minimization of convex functions,” Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), Tech. Rep., 2011. 58

[278] J. C. Spall, “Simultaneous perturbation stochastic approximation,” *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, pp. 176–207, 2003. 58

[279] S. Li, X. Wu, and M. Tan, “Gene selection using hybrid particle swarm optimization and genetic algorithm,” *Soft Computing*, vol. 12, no. 11, pp. 1039–1048, 2008. 58

[280] F. Ahmad, N. A. M. Isa, Z. Hussain, and M. K. Osman, “Intelligent medical disease diagnosis using improved hybrid genetic algorithm-multilayer perceptron network,” *Journal of medical systems*, vol. 37, no. 2, p. 9934, 2013. 58

[281] X. Liu and H. Fu, “Pso-based support vector machine with cuckoo search technique for clinical disease diagnoses,” *The Scientific World Journal*, vol. 2014, 2014. 58

[282] I. Mandal, “Svm-pso based feature selection for improving medical diagnosis reliability using machine learning ensembles 1,” 2012. 58

[283] H. Jiang, F. Tang, and X. Zhang, “Liver cancer identification based on pso-svm model,” in *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*. IEEE, 2010, pp. 2519–2523. 58

[284] Q. Shen, W.-M. Shi, W. Kong, and B.-X. Ye, “A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification,” *Talanta*, vol. 71, no. 4, pp. 1679–1683, 2007. 58

[285] H. Zhang, Q.-Y. Chen, M.-L. Xiang, C.-Y. Ma, Q. Huang, and S.-Y. Yang, “In silico prediction of mitochondrial toxicity by using ga-cg-svm approach,” *Toxicology in Vitro*, vol. 23, no. 1, pp. 134–140, 2009. 58

[286] L. Li, W. Jiang, X. Li, K. L. Moser, Z. Guo, L. Du, Q. Wang, E. J. Topol, Q. Wang, and S. Rao, "A robust hybrid between genetic algorithm and support vector machine for extracting an optimal feature gene subset," *Genomics*, vol. 85, no. 1, pp. 16–23, 2005. 58

[287] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 89–103, 2018. 58, 96

[288] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999. 58

[289] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008. 58

[290] J. Lehman and R. Miikkulainen, "Neuroevolution," *Scholarpedia*, vol. 8, no. 6, p. 30977, 2013. 58

[291] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, p. 503, 2015. 58

[292] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006. 58

[293] S. Nolfi and D. Floreano, "Evolutionary robotics," 2000. 58

[294] C. G. Langton *et al.*, *Artificial life.* Addison-Wesley Publishing Company Redwood City, CA, 1989. 58

[295] M. A. Bedau, J. S. McCaskill, N. H. Packard, S. Rasmussen, C. Adami, D. G. Green, T. Ikegami, K. Kaneko, and T. S. Ray, "Open problems in artificial life," *Artificial life*, vol. 6, no. 4, pp. 363–376, 2000. 58

[296] J. Lehman, J. Chen, J. Clune, and K. O. Stanley, "Safe mutations for deep and recurrent neural networks through output gradients," *arXiv preprint arXiv:1712.06563*, 2017. 59

[297] K. O. Stanley and R. Miikkulainen, "A taxonomy for artificial embryogeny," *Artificial Life*, vol. 9, no. 2, pp. 93–130, 2003. 59

[298] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 59

[299] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897. 59

[300] C. H. Antink, H. Gao, C. Brüser, and S. Leonhardt, "Beat-to-beat heart rate estimation fusing multimodal video and sensor data," *Biomedical optics express*, vol. 6, no. 8, pp. 2895–2907, 2015. 65

[301] C. Brüser, S. Winter, and S. Leonhardt, "Robust inter-beat interval estimation in cardiac vibration signals," *Physiological measurement*, vol. 34, no. 2, p. 123, 2013. 65

[302] F. Plesinger, P. Klimes, J. Halamek, and P. Jurak, "False alarms in intensive care unit monitors: detection of life-threatening arrhythmias using elementary algebra, descriptive statistics and fuzzy logic," in *Computing in Cardiology Conference (CinC), 2015.* IEEE, 2015, pp. 281–284. 74, 76, 92, 95

[303] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, 1992. 75

[304] Y. H. Zweiri, J. F. Whidborne, and L. D. Seneviratne, "A three-term backpropagation algorithm," *Neurocomputing*, vol. 50, pp. 305–318, 2003. 75

[305] C. Shang, A. Palmer, J. Sun, K.-S. Chen, J. Lu, and J. Bi, "Vigan: Missing view imputation with generative adversarial networks," in *2017 IEEE International Conference on Big Data (Big Data).* IEEE, 2017, pp. 766–775. 78

[306] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010. 78

[307] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2849–2857. 80

[308] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org, 2017, pp. 1857–1865. 81

[309] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797. 88

[310] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev *et al.*, "Tensorflow lite micro: Embedded machine learning on tinyml systems," *arXiv preprint arXiv:2010.08678*, 2020. 96

[311] P. Warden and D. Situnayake, *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers.* O'Reilly Media, 2019. 96

[312] V. Gonzalez-Huitron, J. A. León-Borges, A. Rodriguez-Mata, L. E. Amabilis-Sosa, B. Ramírez-Pereda, and H. Rodriguez, "Disease detection in tomato leaves via cnn with lightweight architectures implemented in raspberry pi 4," *Computers and Electronics in Agriculture*, vol. 181, p. 105951, 2021. 96

[313] S. Cass, "Nvidia makes it easy to embed ai: The jetson nano packs a lot of machine-learning power into diy projects-[hands on]," *IEEE Spectrum*, vol. 57, no. 7, pp. 14–16, 2020. 96