1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Paulo Alexandre Castillo Fernandes

SUPERVISED DATA AUGMENTATION

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Fernando Jorge Penousal Martins Machado and Prof. João Nuno Gonçalves Costa Cavaleiro Correia, and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

October 2020

Acknowledgements

This dissertation is the climax of two years of work. Throughout this time there were many moments of joy and frustration and nights without sleep. Nevertheless, this was a journey that I feel would not have been able to make on my own. To all the people that stood with me, a big "thank you".

To my supervisors, Prof. Fernando Jorge Penousal Martins Machado and Prof. João Nuno Gonçalves Costa Cavaleiro Correia, thank you for the knowledge, guidance and support that made it possible to materialize this project. Thank you CISUC-DEI, for making me part of the ECOS group and the CDV lab as well as for providing the working environment and infrastructure to make this work possible. I also want to thank the IPN, the UC and the FCT for granting funds to support this project. And last but not least, I want to thank my friends and family for all the love and support that gave me strength to work harder and better and kept me going even through hardships.

Once again, thank you!

Paulo Alexandre Castillo Fernandes,

Coimbra, October 2020

Faculty of Sciences and Technology

Department of Informatics Engineering

Supervised Data Augmentation

Paulo Alexandre Castillo Fernandes

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Fernando Jorge Penousal Martins Machado and Prof. João Nuno Gonçalves Costa Cavaleiro Correia, and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

October 2020

1 2 9 0

UNIVERSIDADE Ð COIMBRA

This page is intentionally left blank.

Abstract

Machine Learning (ML) has seen tremendous advances in recent years and is currently invading many areas of technology. ML, though, is dependent on stored information in datasets to learn how to perform a certain task. Many times, datasets suffer from imbalances and missing information which makes it more difficult to train ML models. The way to solve this is by performing Data Augmentation (DA). Although there are many ways to perform DA, there are still only a few pieces of research that look into the usage of Generative Adversarial Networks (GANs) to produce samples for this task. A GAN is a model that is able to learn the distribution of a dataset through training and can generate samples according to this distribution. The amount of different samples that a GAN can produce is innumerable, and many of these samples are even distinct from what is found in the training set because of the combination of different features. The main issue is that usually there is no control over how the model generates new samples. The literature indicates that it might be necessary to include an extra form of management in the generation phase of the GAN. This begs the questions "How can this management be done?" and "How can a GAN be made to generate a certain type of data?".

In this dissertation is explored the usage of GANs to perform DA while also trying to answer the previous questions. Thus, a framework is proposed for performing DA in datasets of images, the Evolutionary Latent Space Exploration Generative Adversarial Network (ELSEGAN). More specifically, GANs will be used to generate images for several datasets in order to improve classifiers in tasks of Image Classification (IC). A supervisor module will be added to the GAN to manage the generation and addition of images. The supervision is performed by Evolutionary Computation (EC) that is used to explore the latent space of the GAN and search for sets of images that optimise a certain objective. Different EC algorithms are explored as well as different metrics and criteria for supervision. Finally, a classifier is used to attest the performance of the models that were created by the framework and DA approaches.

The first experiments were centred around the process of supervision and exploration of the latent space using EC. The exploration featured three EC algorithms, namely Random Sampling (RS), Genetic Algorithm (GA) and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites), guided by an image similarity criterion that was tested using two distinct images similarity metrics, Root-Mean-Squared Error (RMSE) and Normalized Cross-Correlation (NCC). Overall, the experiments performed on a set of image datasets show that it is possible to guide the exploration of latent space with EC to find sets of images that show optimisation of a certain criterion.

In a second set of experiments, the ELSEGAN was put to the test by performing DA within the context of a real-world problem using the Human Sperm Head Morphology (HuSHeM) dataset, a bio-medicine multi-class problem with a small number of samples that provides a challenge to the different supervised classification approaches. Additionally, another method of supervision was explored that was guided by the loss of a previously trained classifier. Furthermore, the possibility of a new process of training that features dynamic DA was also tested. The results of classifier performance tests, revealed that the classifiers that were trained with DA showed an overall improvement over those with no DA, increasing the performance by more than 5% in some cases.

In the end, the experimental results attained in the experiments throughout the dissertation show the validity and potential of the ELSEGAN approach.

Keywords

Machine Learning, Image Classification, Data Augmentation, Generative Adversarial Network, Evolutionary Computation, Latent Space Exploration.

This page is intentionally left blank.

Resumo

A Aprendizagem Máquina (AM) tem visto tremendos avanços nos últimos anos e está atualmente a invadir muitas áreas da tecnologia. Porém, a AM está dependete da informação guardada em conjuntos de dados para aprender a realizar determinadas tarefas. Muitas vezes, os conjuntos de dados sofrem de desiquilíbrios e falta de informação, o que faz com que seja mais difícil treinar os modelos de AM. O caminho para resolver isto é através do Aumento de Dados (AD). Apesar de já existirem muitas formas de realizar AD, ainda há poucos estudos que procurem a utilização de Redes Generativas Adversariais (RGA) para produzir amostras para esta tarefa. Uma RGA é um modelo que é capaz de aprender a distribuição de um conjunto de dados através do treino e é capaz de gerar amostras consoante esta distribuição. A quantidade de amostras diferentes que uma RGA é capaz de produzir é inumerável, e muitas destas amostra são ainda distintas das que são encontradas no conjunto de treino devido à combinação de diferentes características. O principal problema é o facto de normalmente não existir controlo sobre a geração de novas instâncias. A literatura indica que poderá ser necessário incluir uma forma extra de gerir esta geração. Isto levanta as questões "Como poderá ser feita esta gestão?" e "Como se poderá fazer com que uma RGA gere um determinado tipo de dados?".

Nesta dissertação é explorada a utilização de RGAs para realizar AD ao mesmo tempo que tenta responder às questões anteriores. Desta forma, é proposta uma estrutura para realizar AD em conjuntos de images, a ELSEGAN (Rede Generativa Adversarial de Exploração Evolucionária do Espaço Latente). Mais especificamente, irão ser usadas RGAs para gerar imagens para vários onjuntos de dados de modo a melhorar classificadores em tarefas de Classificação de Image (CI). Um módulo supervisor irá ser adicionado à RGA para gerir a produção e adição de imagens. A supervisão é feita através de Computação Evolucionária (CE) que é usada para explorar o espaço latente da RGA de modo a procurar conjuntos de images que optimizem um certo objetivo. São explorados diversos algoritmos de CE tal como diferentes métricas e critérios de supervisão. No fim, um classificador é usado para verificar o desempenho dos modelos criados pela estutura e pelas abordagens de AD.

As primeiras experiências foram centradas à volta do processo de supervisão e exploração do espaço latente por CE. A exploração destacou três algoritmos, nomeadamente Geração Aleatória (GA), Algoritmo Genético (AG) e Arquivo Fenotípico Multi-dimensional de Elites (MAP-Elites), guiados por um critério de semelhança de imagem que foi testado com duas métricas de semlhança de images distintas, Raíz do Erro Quadrático Médio (RMSE) e Correlação Cruzada Normalizada (NCC). Globalmente, as experiênicas realizadas numa seleção de conjuntos de images mostram que é possível guiar a exploração do espaço latente através de CE para encontrar conjuntos de imagens que mostrem a optimização de um certo critério.

Num segundo conjunto de experiências, a ELSEGAN foi posta à prova através da realização de AD dentro do contexto de um problema do mundo real utilizando o conjunto HuSHeM (Morfologiaia da Cabeça de Esperma Humano), um problema de multiclasse em biomedicina com um número reduzido de amostra que fornece um desafio para diferentes abordagens de aprendizagem supervisionada. Adicionalmente, outro métode de supervisão foi testado, que se guiava através do erro causado num classificador pré treinado. Para além disso, foi ainda testado um novo processso de treino que utiliza AD dinâmica. Os resultados de desempenho dos classificadores revelaram que classificadores treinados com AD mostram, na generalidade, uma melhoria em relação aos treinados sem AD, aumentando a performance em mais de 5% em certos casos.

No fim, os resultados experimentais obtinos nas experiências ao longo da dissertação mostram a viabilidade e o potencial da abordagem ELSEGAN.

Palavras-Chave

Aprendizagem Máquina, Classificação de Imagem, Aumento de Dados, Rede Generativa Adversarial, Computação Evolucionária, Exploração de Espaço Latente.

This page is intentionally left blank.

Contents

1 Introduction **1**
1.1 Hypothesis . . . 2
1.2 Objectives . . . 2
1.3 Outline . . . 3

2 State of the Art **5**
2.1 Context and Machine Learning Models . . . 5
2.1.1 Datasets . . . 6
2.1.2 Classification . . . 6
2.1.3 Artificial Neural Networks and Deep Learning . . . 7
2.2 Data Augmentation & Generative Adversarial Networks . . . 7
2.3 Latent Space Exploration . . . 8

3 Approach **11**
3.1 Overview . . . 11
3.2 Classifier . . . 12
3.3 Generator . . . 13
3.4 Supervisor . . . 13
3.5 Offline and Online Training . . . 14
3.6 Development Environment . . . 15

4 Latent Space Exploration **17**
4.1 Datasets . . . 17
4.2 Generator . . . 18
4.3 Classifier . . . 19
4.4 Supervisor . . . 22
4.4.1 Setup . . . 22
4.4.2 Results . . . 24

5 Latent Space Exploration for Classifier Improvement **30**
5.1 Dataset: Human Sperm Head Morphology (HuSHeM) . . . 30
5.2 Generative Adversarial Networks . . . 31
5.3 Offline Training Experiments . . . 31
5.3.1 Supervisor . . . 32
5.3.2 Classifier . . . 33
5.3.3 Experimental Results . . . 34
5.4 Online Training Experiments . . . 40
5.4.1 Classifier . . . 40
5.4.2 Experimental Results . . . 41

6 Conclusion **43**

This page is intentionally left blank.

Acronyms

ADGN Applied Deep Generative Networks. 30

ANN Artificial Neural Network. xiv, 7, 10

AUROC Area Under Receiving Operator Characteristic Curve. xvii, 34, 35, 40

CS Computer Science. 1, 5

CT computed tomography. 8

CV Computer Vision. 1

CycleGAN Cycle-Consistent Adversarial Networks. 8

DA Data Augmentation. iii, xvii, 1–3, 5, 7, 8, 11–14, 18, 30–32, 34, 35, 40, 41, 43, 44

DAGAN Data Augmentation Generative Adversarial Network. 8

DCGAN Deep Convolutional Generative Adversarial Network. xvii, 12–14, 17–19, 21, 22, 24, 30, 31, 33

DCNN Deep Convolutional Neural Network. 12, 13, 40, 43

DL Deep Learning. 7

DNN Deep Neural Network. 7

EA Evolutionary Algorithm. 9

EC Evolutionary Computation. iii, 2, 5, 8, 9, 11, 14, 17, 22, 32, 43

ECAI European Conference on Artificial Intelligence. 30

ELSEGAN Evolutionary Latent Space Exploration Generative Adversarial Network. iii, vi, xiv, 11–15, 30, 31, 43, 44

FD Face Detection. 1

GA Genetic Algorithm. iii, xvii, 22–24, 29, 32, 43

GAN Generative Adversarial Network. iii, 2, 3, 5, 8, 9, 11, 12, 14, 17, 18, 34, 43, 44

HuSHeM Human Sperm Head Morphology. iii, vi, xiv, 30, 31, 43

IC Image Classification. iii, 2, 11, 12, 18, 43

IR Image Recognition. 1, 7, 43

KPCA Kernel Principal Component Analysis. 8

MAP-Elites	Multi-dimensional Archive of Phenotypic Elites. iii, xvii, 22–24
ML	Machine Learning. iii, 1, 3, 5, 6, 15, 43
NCC	Normalized Cross-Correlation. iii, xiv, 23–25, 27, 29, 33, 43
OffT	Offline Training. xv, 12, 14, 19, 31, 40–44
OnT	Online Training. xiv, xv, 12, 14, 16, 40–44
RMSE	Root-Mean-Squared Error. iii, xiv, 23–25, 28, 29
RS	Random Sampling. iii, xvii, 22, 23
SL	Supervised Learning. 6
VAE	Variational Autoencoder. 8

This page is intentionally left blank.

List of Figures

2.1 Representation of a dataset for classification in supervised learning. A set of instances $x_{c,n}$ and corresponding targets t_c, where c is the class and n the number of the instance. 10
2.2 Representation of an artificial neuron. Input: $x = \{x_1, x_2, x_3\}$; Weights: $w = \{w_1, w_2, w_3\}$; Bias: b; Activation function: f; Output: y. 10
2.3 Two representations of Artificial Neural Networks (ANNs), one with one layer (left) and with another with three layers (right). 10

3.1 Visual representation of the Evolutionary Latent Space Exploration Generative Adversarial Network (ELSEGAN) framework 12
3.2 Workflow of the process of evaluation of the ELSEGAN 13
3.3 Latent space exploration cycle via evolutionary computation. 16
3.4 Visual representation of the Online Training (OnT) process 16

4.1 Samples of the datasets used in this work. From left to right: MNIST, Fashion-MNIST, Cats vs. Dogs, Facity 18
4.2 On the left: Images generated by MNIST and Fashion-MNIST generators after training. On the right: Evolution of training from MNIST and Fashion-MNIST generators over epoch - d_loss: discriminator loss, g_loss: generator loss, d_acc: discriminator accuracy (0-1). 19
4.3 On the left: Images generated by cats, dogs and Facity generators after training. On the right: Evolution of training from cats, dogs and Facity generators - d_loss: discriminator loss, g_loss: generator loss, d_acc: discriminator accuracy (0-1). 20
4.4 Evolution of classifier performance with MNIST dataset comparing addition of generated images incrementally 21
4.5 Average Maximum values across iterations for the different datasets and similarity metrics: on the left Normalized Cross-Correlation (NCC) and on the right Root-Mean-Squared Error (RMSE). The values are averages of 10 runs. 25
4.6 Heatmap of the Map constructed with Map-Elites for every dataset and similarity metric. Images correspond to 0th, 12000th and 25000th iterations of the algorithm. The color represent fitness, yellow for better fitness and blue for worse, while the red represents the best individual. The x axis corresponds to the average similarity and the y axis to the maximum similarity 26
4.7 The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the NCC metric. 27
4.8 The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the RMSE metric. 28

5.1 Class examples for each of the Human Sperm Head Morphology (HuSHeM) classes 31

5.2 Class examples for real (left column) and generated (right column). On each row from, top to bottom: Normal, Tapered, Pyriform, Amorphous 32
5.3 Evolution of the best individual fitness with the image similarity criteria for each class of the problem: 0-Normal, 1-Tapered, 2-Pyriform, 3-Amorphous . 36
5.4 Evolution of the best individual fitness with the classifier loss criteria for each class of the problem: 0-Normal, 1-Tapered, 2-Pyriform, 3-Amorphous . 36
5.5 Best individual in the first (left) and last (right) generations for each class, by the image similarity criteria. 37
5.6 Best individual in the first (left) and last (right) generations for each class, by the classifier loss criteria. 38
5.7 Visual comparison, via F1-score (left) and AUROC (right) metrics, of classifiers' performance, over 30 runs, between offline trainings with original dataset, with randomly augmented dataset, with augmentation by supervision through image similarity and with augmentation by supervision through classifier loss. 39
5.8 Visual comparison, via F1-score metric, of classifier's performance, over 30 runs, between offline trainings with randomly augmented datasets with different amounts of added images, from 8 additional images per class to 216 additional images per class. 41
5.9 Visual comparison, via F1-score metric, of classifier's performance, over 30 runs, between OnT and Offline Training (OffT) trainings with randomly augmented datasets with different amounts of added images, from 24 additional images per class to 120 additional images per class. 42

This page is intentionally left blank.

List of Tables

4.1 Deep Convolutional Generative Adversarial Network (DCGAN) parameters 18
4.2 Classifier parameters 20
4.3 Random Sampling (RS), Genetic Algorithm (GA) and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) Parameters 23

5.1 Generative Adversarial Network training parameters 31
5.2 Genetic Algorithm Parameters 32
5.3 Classifier parameters for supervision 33
5.4 Classifier parameters for Offline Training 34
5.5 Fitness value of the first and last best individuals according to the Image Similarity and Classifier Loss criteria for each class of the problem. 35
5.6 Classifier performance in testing phase across all metrics measured. Performance comparison of mean, standard deviation (std.) and max values between training with original dataset, with randomly augmented dataset, with augmentation by supervision through image similarity and with augmentation by supervision through classifier loss 39
5.7 Statistical comparison between the different Data Augmentation (DA) approaches for the average results of the F1-score and Area Under Receiving Operator Characteristic Curve (AUROC) metrics. O: Original (no DA; IS: image similarity DA; CL: classifier loss DA; R: random DA) 40
5.8 Classifier parameters for Online Training 40

This page is intentionally left blank.

Chapter 1

Introduction

Contents

1.1 Hypothesis 2
1.2 Objectives 2
1.3 Outline 3

Machine Learning (ML) is a branch of Computer Science (CS) that has seen great developments in recent years due to the major increase in computer capabilities. From social media services to autonomous vehicles, from online fraud detection to virtual personal assistants, although slowly, it is steadily being introduced into our lives even without us noticing it most of the time. As only recently it has started to see noticeable developments, it is still an area that has a lot of space for improvement and optimization.

One of the areas where ML is most important nowadays is Image Recognition (IR). IR is a branch of Computer Vision (CV) that refers to the analysis of images or videos in order to identify detect objects or attributes that give information about its contents such as people, places, buildings, etc. This makes it possible to have, for example, face recognition systems and autonomous vehicles.

IR that relies on ML is dependent on a dataset that is used to train an algorithm to perform a certain task such as classification, regression, detection or other. This means that while the algorithm is really important to achieve the threshold of success of a certain task, the quality of the dataset used to train it is also of the most importance. Having a poor quality dataset will invariably lead to poor results, independently of the time that it has to train. Likewise, the improvement of the quality of the training dataset, which can be a complex and time-consuming process, will, undoubtedly, improve the performance of the algorithm in the specified task.

There are different issues that can affect the quality of a dataset. Representativeness and completeness are two of the major types of problems. Each sample in a dataset has its own set of features, such as hair color or wearing glasses in case of a Face Detection (FD) problem. If these are not properly distributed throughout the dataset, and more specifically in each different class, not covering enough of the solution spaces, the algorithm may be led into wrong assumptions about the data such as assuming a correlation, between classes and its features, that does not correspond to the reality. For instance, if we train an algorithm to detect faces using a dataset where all people with dark hair wear glasses, it may not be able to detect a face with dark hair wearing no glasses. These issues end up affecting negatively the performance of said algorithms. This is where DA comes in. Its aim is to complement datasets by introducing new samples in order to increase the representation of certain parts of the data or even add data that was completely missing from the beginning. It is especially important in cases where there is an aggravated difficulty in gathering external data samples that belong to certain parts of the solution space. Problems like online fraud detection suffer greatly from this since the number of attempted cases of fraud (detected or not) is so much lower than that of non-fraud cases. Therefore a fraud attempt using a new pattern that differs, even that only slightly, enough from previously observed cases is more susceptible to end up escaping under the radar.

1.1 Hypothesis

In this research, the problem of DA will be applied to supervised Image Classification (IC) tasks. It will be addressed through the usage of Generative Adversarial Networks (GANs) for generation of new images that will be used to complement image datasets. GANs have gained popularity over the years, presenting state-of-the-art results in the generation of samples that follow the distribution of the input training dataset[12]. In general, this model of adversarial learning works by having a generator and a discriminator training together and competing against each other in a min-max game. The discriminator is trained with real data as well as fake data created by the generator from latent space vectors. The generator evolves from the feedback given by the discriminator on the generated data.

"Can GANs be used to perform DA on incomplete datasets to effectively improve the performance of image classifiers?" - This is the main research question to which this dissertation will try to provide an answer. The addition of new images should improve the quality of the dataset by tackling the issues of representativeness and completeness.

Along with the generation of samples, there's also a need for selection. One way to perform this process is by doing it randomly, which means we will be generating the samples randomly and automatically add them to the training set. Although, this way, there is no criterion over the samples that we are adding, which means we may end up using flawed or redundant samples, which might end up undermining the performance of the algorithm. Consequently, it is fundamental the generation and selection of samples to be made in a meaningful way such that the chances of the performance of the algorithm being improved are greater. The generator produces an image from a vector of values from its latent space which is usually of high dimensionality. Therefore, that input directly affects the resulting images, which means that one way to control the images that are generated is by selecting the vectors from the latent space with a certain criterion. Thus, by exploring the latent space of GANs it is possible to uncover hidden patterns, which might help in the proposed task. One way to perform this exploration is by using Evolutionary Computation (EC) in order to find sets of images that optimize as much as possible a certain criterion or objective function such as diversity.

1.2 Objectives

This dissertation will have some objectives to attend to:

- Research and analysis of the state of the art that will provide knowledge which will be the stepping stone for the development of this work.
- Selection of the scenarios and datasets which will be used to test the hypothesis;

- Implementation of algorithm(s) that will be used to provide the baseline for the work;
- Implementation of a framework that will allow the generation of new samples;
- Implementation of a framework responsible for the selection of generated samples;
- Analysis of results from the training of algorithms;
- Assess the viability of the solution for application to real world problems.

The novelty in this work, is only associated with the DA process and usage GANs. The development of new ML algorithms and optimization of algorithms used in the process are out of scope, although it may be necessary to experiment with different models and methods for the generation of new samples. The implementation of new methods of evaluation and selection of images created by the generative models is also out of scope.

1.3 Outline

The present document is aligned with the following structure:

- Chapter 2 - State of the Art - Contains the state of the art for the aspects related to this work about, DA, GANs, and latent space exploration;
- Chapter 3 - Approach - Explains the goals of the dissertation and the plan to achieve them;
- Chapter 4 - Latent Space Exploration - Describes the preliminary steps and experiments performed for latent space exploration;
- Chapter 5 - Latent Space Exploration for Classifier Improvement - Describes the experiments performed for the application of Latent Space Exploration to the DA task in a real-world problem;
- Chapter 6 - Conclusions - Presents a discussion of the results obtained and conclusions taken.

This page is intentionally left blank.

Chapter 2

State of the Art

Contents

2.1 Context and Machine Learning Models 5
2.1.1 Datasets 6
2.1.2 Classification 6
2.1.3 Artificial Neural Networks and Deep Learning 7
2.2 Data Augmentation & Generative Adversarial Networks 7
2.3 Latent Space Exploration 8

Machine Learning (ML) relies on two main components: model and dataset. The optimization of both is of the utmost importance in order to reach good performance. Working with an a exceptional model or dataset means nothing if the weakest link bottlenecks its performance. Keeping this in mind, this work will be focused on the betterment of datasets through Data Augmentation (DA). In order to perform this, new instances will be generated through the usage of Generative Adversarial Networks (GANs) and added to the original dataset. GANs are ML models capable of generating an innumerable number of new samples by learning from the distribution of the training dataset. As much as it is not possible no count the number of different samples that GANs can produce, it is also not trivial to know exactly what is being generated with a specific input. As a consequence, an extra step of selection of samples will be introduced. This selection will be performed by exploration of the latent space of the GAN using Evolutionary Computation (EC) towards finding relevant images to be added to the original training set.

The remaining of this chapter will focus on giving insight on the topics addressed by this dissertation.

2.1 Context and Machine Learning Models

ML is a vast field in Computer Science (CS) which mostly refers to systems that are able to "learn" by changing their structure, program or data based on observations in order to improve their future performance [20]. One example is a speech recognition machine which improves after hearing several samples of a person's speech.

There are several types of learning methods. In this work, we will be looking into 3 types of learning, namely supervised learning, unsupervised learning and evolution learning [18]:

- Supervised Learning - A set of observations and correct responses (targets) is provided to train the algorithm, usually written as a set of data $\{(x_i, t_i), i = 1, ..., N\}$, where each x_i is a single input (instance observed) and t_i its corresponding target (expected output). The algorithm tries to generalise to respond correctly.

- Unsupervised Learning - Only a set of observations is provided. The algorithm identifies similarities between inputs and tries to infer relationships between them in order to assign categories.

- Evolutionary Learning - A biology inspired process rooted in the evolution of organisms and generation of better offspring guided by a function of individual fitness.

2.1.1 Datasets

A dataset is defined as a collection of observations of a problem [11], which, in the case of Supervised Learning (SL), are accompanied by the corresponding targets. Each observation (also called instance, data point, example or sample) is a single input to the ML model, which, depending on the approach, may go through a step of pre-processing before being read by the model. The set of targets contains each output expected for each instance of the problem. It is used in the evaluation the model according to a defined metric by comparing the predictions given by the algorithm with the respective expected value and allows its optimization.

In learning process 3 different datasets are used [11]:

- Train set - Collection of instances of a problem that are actively used in the training step of the algorithm and from which the model performs inferences.

- Validation set – A set of observations which is usually constructed from the training data. It passively used in the training process to estimate the generalization error and evolution of the model.

- Test set – Collection of observations which are used post-training to ascertain the final performance of the algorithm.

2.1.2 Classification

This work is focused on improving the performance of image classifiers. Classification tasks are discrete problems which means that they work with discrete targets, one for each class of the problem. The objective is to categorise inputs into single classes. In general, classification algorithms aim to find decision boundaries that can be used to separate out the different classes [18]. One basic example of this type of problem is that of "cats vs. dogs" where, as the name suggests, a classifier has to distinguish between images of cats and images.

Classification is a type of problem that can be solved via all types of learning aforementioned. This work specifically, revolves around improvement of classifiers trained via supervised learning.

2.1.3 Artificial Neural Networks and Deep Learning

Inspired by the nature of the brain, Artificial Neural Networks (ANNs) are algorithms that aim to produce a certain output from a given input. For instance, in a classifier, $y = f*(x)$ maps an input x to a category y [11]. These networks are called this way, because they are a composition of several different functions. The artificial neuron, which is mathematical model of the biological neuron, is the basic component of an ANN, representing a single function in the network. These ANNs are built by connecting several of these artificial neurons.

Essentially, the artificial neuron applies an activation function on a weighted sum of inputs to which is also added a bias: $y = f(w^T x + b)$ [Figure 2.2].

An ANN is an organisation composition of neurons into one or more layers [Figure 2.3]. When an ANN is composed of multiple layers of neurons it is called a Deep Neural Network (DNN). With the addition of more layers and more neurons within a layer, a DNN can represent functions of increasing complexity [[11]]. This depth that is given to the model is what gives Deep Learning (DL) its name. The behaviour of the layers that are not connected to the inputs is not directly specified by the training data. As such, the learning algorithm must decide how to use these layers to best implement an approximation of $f*$ with the conditioning that the behavior of the other layers is not directly specified by the training data. This is why these are typically called "hidden layers".

These models are usually trained using an algorithm of back-propagation that corrects the weights and bias of the neurons based on the value given by a cost function over the predictions made during training. With this, the network learns to approximate $f*$ by minimising the error.

2.2 Data Augmentation & Generative Adversarial Networks

There are 3 aspects three aspects that affect the quality of a dataset [4]:

- Correctness - correct labelling of the dataset samples;
- Representativeness – inclusion of instances representative of each assigned class;
- Completeness – coverage of the solution space for each class.

While the correctness is usually relatively easy assess, requiring only the time spent on manually verifying and correcting each pair image-label, the aspects of representativeness and completeness are a lot more difficult to verify and correct and may often lead to loss of generalisation. In order to tackle this issue DA is fundamental.

DA can be a really complex and tedious tasks. It must take into account the aspect of cost-benefit. Oftentimes the complexity of the problem makes it so that it is not beneficial to use DA to achieve full completeness and representativeness in a dataset. As such, it is important to optimize instance generation and selection in order to make the most benefit with the least cost.

There are currently many ways to perform DA in Image Recognition (IR) which may include geometric operations such as rotation, mirroring, scaling and distortion, or color operations which distort color channels [32]. Other methods include randomly erasing or adding noise to patches of images [17], usage of occlusion patterns [27] or copy-pasting objects of interest into other backgrounds [6]. Further methods include merging of objects from the same class [16] and generation of augmented data, using a Bayesian approach, based on the distribution learned from the training set [25]. Another method, for DA, which serves as inspiration for this work, was explored at [4] where a EC engine was used to evolve synthetic images. The evolution engine is complemented with a module that supervises the images that are being generated, selecting and filtering images to be added to the dataset according to a certain objective or criteria.

Recent studies have also proposed the usage of GANs to perform DA. GANs were originally proposed as a method of unsupervised learning [12] for generative models in order to produce synthetic samples of data that follow the distribution of the training dataset. These models even have the possibility to produce samples that are not present in the original dataset, by combination of the features from original samples. GANs are trained through a face-off between a generator and a discriminator. The discriminator is responsible for distinguishing original (real) samples from generated (fake) samples. On the other hand, the generator learns by the feedback of the discriminator on the fake samples generated. These samples originate from a random noise vector, a randomly sampled vector from a high dimensional latent space, according to the prior which is a predetermined sampling distribution. In time, the generator learns to map its latent space into the distribution of observations from the original dataset, producing increasingly more realistic samples.

In their work [5], Tanaka et al study the possibility of using GANs to train a classier without using the original dataset, due to privacy reasons, as well as to over-sample the least represented class in a imbalanced dataset. Their results show that the usage of GANs is applicable in both cases. Another work that may be mentioned is that of Antoniou et al. [1] which developed the Data Augmentation Generative Adversarial Network (DAGAN), a framework that enables the generation of an image output that is a transformation of a given input image. This framework shown satisfying results across all tests.

In the same topic, and specifically applied to bio-medicine, there are two works, among others, that can be mentioned. The work of Frid-Adar et al. in liver lesion classification [10] shows improvement in performance of classifiers trained with synthetic images generated by a GANs over classifiers trained traditional forms of DA. Furthermore, there is also a work [23] that explored the usage of Cycle-Consistent Adversarial Networkss (CycleGANs) for generation of non-contrast computed tomography (CT) images from contrast CTs so as to perform data augmentation in CT segmentation tasks. It is shown that U-nets trained with CycleGAN augmentation provide better results even when compared to other forms of DA.

2.3 Latent Space Exploration

In order to generate a sample, a GAN is usually given a random noise vector from the latent space sampled using the training prior. While the exploration of latent space in generative models is a topic still in its beginnings, there's already some work that has been made in this area, and not only with GANs, which is the type of model that we are going to work within this experiment. For instance, latent space exploration was performed in generative models using Kernel Principal Component Analysis (KPCA) [30], showing navigation through image features and novelty detection; but also in Variational Autoencoder (VAE), in, for example, mapping genes into a lower-dimensional space in order uncover underlying gene expression features in cases of tumour or cancer [28].

There are different ways that we can explore generative models latent space. White,

T. shows three different arithmetical ways we can use to navigate in it [29]. The first, interpolation, is used to disclose the path between 2 samples in the space. Usually, a linear interpolation is used, but White suggests a new way, the spherical linear interpolation - slerp - which takes into account its prior distribution used for sampling noise. Analogy, the second approach, can be simplified as "King – Man + Woman = Queen", meaning that it is possible to perform operations between images in order to form new images with generally predictable results. Finally, the manifold traverse, more specifically Manifold Interpolated Neighbor Embedding (MINE), which makes use of nearest neighbours and interpolation to construct a manifold of the space. Also worth noting is that he shown that by combining generative models with labelled data, attribute vectors can be computed using simple arithmetic, like, for example, a smile vector which, by traversing uncovers several states of smiling in an image.

As for the use of EC in order to evolve images, there are various works that could be mentioned. For example, evolving master print templates [22] that, like a master key, could be able to open multiple fingerprints closed locks. In their work, Roy et al. compared four different Evolutionary Algorithm (EA), namely Hill-Climbing, Covariance Matrix Adaptation Evolution Strategy, Differential Evolution and Particle Swarm Optimization to evolve Synthetic MasterPrints according to the metric proposed by them, the Modified Marginal Success Rate. The samples were generated from two datasets, namely Authentec AES3400, with latter algorithm getting the best results, and FVC 2002 DB1-A, for which the second approach had the most success. Moreover, and with a two-stage workflow similar to what was implemented in this work which includes first the unsupervised training of GANs and second the evolution of latent space, there are two works, one that implements Interactive Evolutionary Computation [3] for image generation and the other, in the topic of videogames, that uses GANs and latent space evolution to learn and improve Mario Levels [26] using Covariance Matrix Adaptation Evolution Strategy.

There is also a new approach to generative models which was inspired by GANs, the Generative Latent Optimization [2]. This approach takes away the adversarial discriminator and replaces it with simple reconstruction losses where the focus is to evolve the latent space to match the one learnable noise vector to each one of the images in the training dataset.

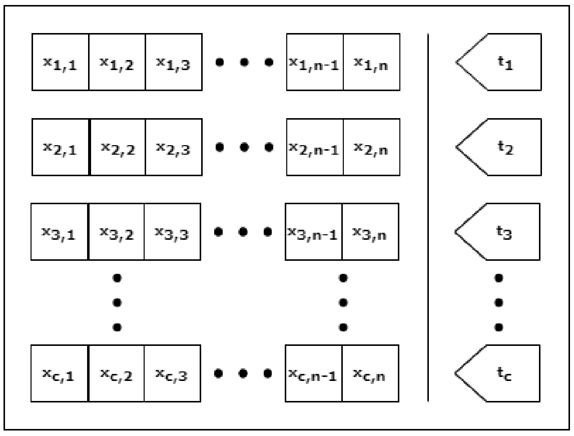

$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ ... $x_{1,n-1}$ $x_{1,n}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ ... $x_{2,n-1}$ $x_{2,n}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ ... $x_{3,n-1}$ $x_{3,n}$
$x_{c,1}$ $x_{c,2}$ $x_{c,3}$ ... $x_{c,n-1}$ $x_{c,n}$
t_1
t_2
t_3
t_c

Figure 2.1: Representation of a dataset for classification in supervised learning. A set of instances $x_{c,n}$ and corresponding targets t_c, where c is the class and n the number of the instance.

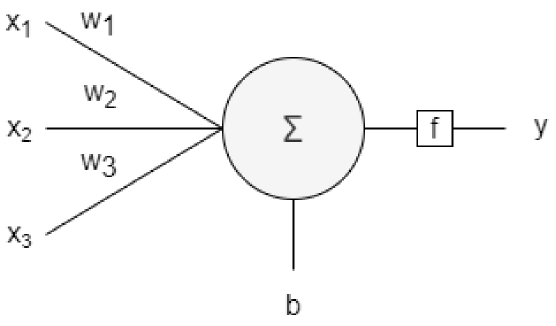


Figure 2.2: Representation of an artificial neuron. Input: $x = \{x_1, x_2, x_3\}$; Weights: $w = \{w_1, w_2, w_3\}$; Bias: b; Activation function: f; Output: y.

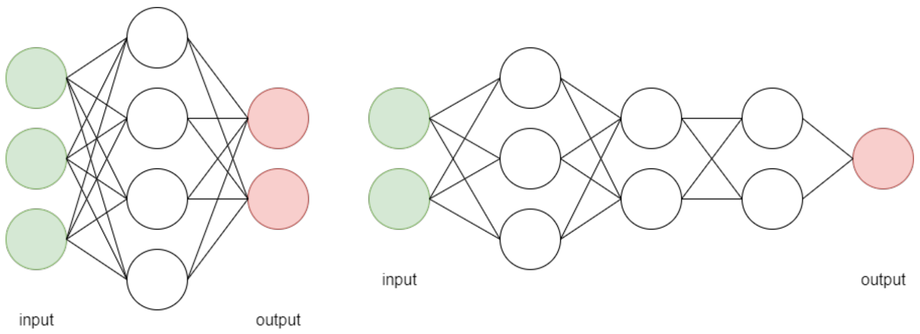


Figure 2.3: Two representations of ANNs, one with one layer (left) and with another with three layers (right).

Chapter 3

Approach

Contents

3.1 Overview **11**
3.2 Classifier **12**
3.3 Generator **13**
3.4 Supervisor **13**
3.5 Offline and Online Training **14**
3.6 Development Environment **15**

This dissertation proposes a new framework, the Evolutionary Latent Space Exploration Generative Adversarial Network (ELSEGAN). This framework combines GANs and EC to perform DA. Our approach intends to use generative models trained via GANs as means to create new samples of data in order to complement datasets so as to tackle the issues of completeness and representativeness.

The ELSEGAN was developed for supervised Image Classification (IC) tasks. By using GANs, it is possible to generate new images from the original dataset, without the need of external sources. Although, it is important to keep in mind that the generated images will follow a distribution that is dependant on the original set. This means that it is not possible to create something that wasn't there from the beginning. For instance, it is not possible to generate a person, from a dataset of dogs, as such the objective in this case would be to combine features from different dogs which might help improve the balance in the dataset.

3.1 Overview

In this work, 3 instances are fundamental for the ELSEGAN framework: a classifier, which is responsible for discriminating images into different classes, a generator, responsible for generating new images from an array of the latent space, and a supervisor, responsible for managing the generation and selection of images [Figure 3.1]. With this in mind, in order to analyse, as much as possible, the effectiveness of the framework, the performance will be measured in 3 steps [Figure 3.2]. First, we need the baseline of the performance, which is the performance of the classifier trained using only the the original dataset, with no DA. Afterwards, using a generator from a GAN trained using the original dataset, new samples will be generated randomly and added to the training set of the classifier, which then will have its performance measured and compared to the baseline. Here, there are many options that can be considered for the definition of the training set in terms of the images used either from the original set, either from the generated images, meaning that we can use any combination of original and generated images. Ultimately, the supervisor will be included. This will allow the control of the generated images in order to optimize the dataset that will be used for training the classifier. The performance of the classifier will also be tested and analysed against the previous ones.

Since this is a work of IC, it is important to note that for the generation of images a type of GAN suitable for this task, the Deep Convolutional Generative Adversarial Network (DCGAN), will be used [21]. This specific type of GAN is composed of Deep Convolutional Neural Networks (DCNNs), which are a type of artificial neural network model that makes use of deep convolutional layers to better explore space correlation in images as to produce more realistic images. This is important in order to approximate as best as possible to the quality of images in the original datasets. Accordingly, for the classification step, a DCNN will also be used.

On another note, a new type of training will also be explored. This new type of training changes the way DA is performed, namely changing from a static to a dynamic DA. This new type of training will be referred as Online Training (OnT), as opposed to the common way of training, which will be referred as Offline Training (OffT).

3.2 Classifier

One of the key instances of this work is the classifier. Its performance through the various steps of experimentation will demonstrate the effectiveness of the framework and guide the progress of the research, since we will be measuring the performance of the dataset through the performance of the classifier. For this, a classification algorithm is needed, one that receives an image as input, since the work is being applied to IC, and returns its predicted classification as an output. Having this in mind, a DCNN will be used. This neural network will be trained through supervised learning [11] with each different set of images. At the end of each training the performance is then measured.

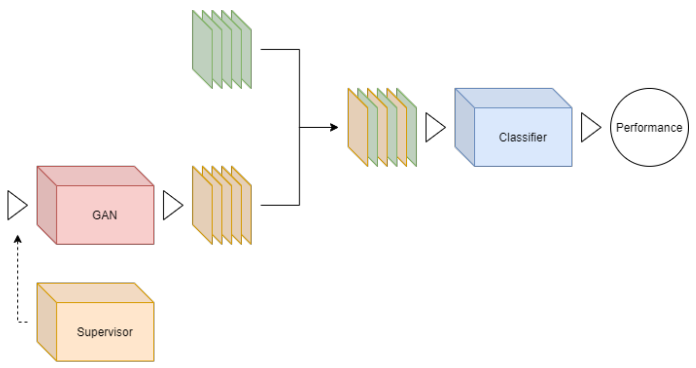


Figure 3.1: Visual representation of the ELSEGAN framework

3.3 Generator

In order to create new images, a generative model is necessary, this is the first step of the DA process. For this task, a DCNN generator will be trained using a DCGAN, which will be put against a discriminator similar to the classifier used to evaluate the performance of the framework. The training of this module is performed using the original dataset as the only source of images. In an unsupervised way, the DCGAN learns to generate images that resemble those of the training set. This means that no information is given to the DCGAN to guide the generation of images, the training progresses purely through the differentiation between real(original) and fake(generated) images [12]. For this reason, in order to keep the balance of each class in the dataset during the addition of generated images to the training set, a number of generators will be trained, equal to the number of classes of the problem, one for each class. This does not go against the premise of combination of features because it should not be desirable to combine features from members of different classes, as it could more easily lead to undesirable results. For instance, in the "Cats vs. Dogs" problem, combining features of cats and dogs would most likely lead to more highly ambiguous images, which would lower the performance of the classification algorithm. The generation of images by the generative model is done from a highly dimensional latent space vector.

3.4 Supervisor

The second and last step of the DA process is supervision. This step is crucial for the optimization of the training dataset. Adding random images to the training set might prove to be unproductive, since, with no criteria, there is now way to know if the images added are relevant to the solution of the problem. The introduction of flawed and redundant images might end up undermining the performance of the classification algorithm [4]. As such, it is important that the addition of images follows a certain criteria. For this reason, the introduction of the supervisor module is fundamental. The generator from the DCGAN produces an image from a vector of an usually high dimensional space called the latent space. This means that by controlling the latent space we can control the output images.

The exploration of the latent space and set of latent vectors has the potential to promote the generation of samples according to pre-determined criteria to solve and adapt to other problems. It could be used during the training of the DCGAN, to have a few latent vectors

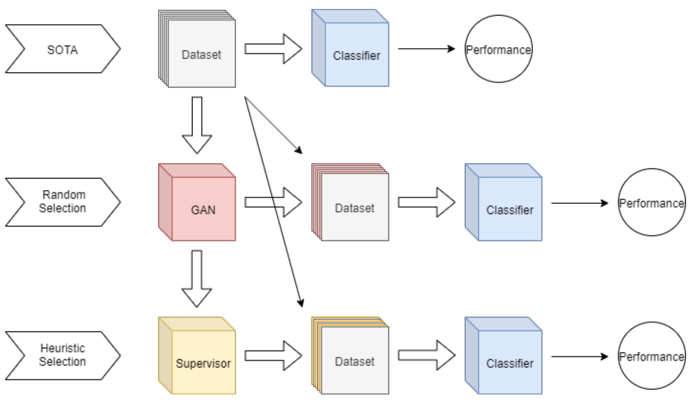


Figure 3.2: Workflow of the process of evaluation of the ELSEGAN

generated that maximize the loss of the model, instead of randomly generating all of them. It could be used to generate samples of a particular type. Some metrics of evaluation of DCGANs must draw samples from the latent space, generate the samples and test them on another model [14]. In the case of this work specifically, exploring the latent space will help in the supervision task by finding sets of images that optimize a certain objective such as diversity.

One way to find sets of images that follow a certain criteria is using EC in order to evolve sets of images that specifically optimize an objective function. The exploration starts with a pool of random individuals, each one representing a number of latent space vectors. For every individual, the corresponding set of images is generated using the previously trained generator for the GAN. These sets of images then go through an EC module where they are evaluated according to a certain fitness function and where transformation operations on the original individuals occur, creating new individuals. The fitness function is the aspect most responsible for the effective evolution of the individuals. The evaluation, will be done in two different ways: The first, a measure of diversity in the sets of images, by using a predefined metric of image similarity, Likewise, the choice of the metric is heavily important since different metrics, which compare images differently, will end up steering the evolution in distinctive ways. The second, a measure of loss from a previously trained classifier, which will search for sets of images that cause the most error. As for the variation operators, it will be taken into account the distribution of values used to train the DCGANs since the exploration of values that fall out of the scope of training would not relevant and could undermine the optimization task. After all EC operations are completed, the result is a new pool of individuals. This cycle is repeated until a certain condition is met. In the end, a set of images should have been found that comes closer to the intended objective and better tackles the issues at hand.

3.5 Offline and Online Training

The ELSEGAN framework will also be applied to two different types of training, the OffT and the OnT.

The OffT corresponds to the most common type of training of Machine Learning models, where a static dataset is prepared and given to the model to train. A dataset is previously prepared, with or without DA, and given to the classifier so it can learn from it. Essentially, the dataset does not change, it is kept the same until the end of the process of training.

On the other hand, with OnT, the dataset is dynamic, changing over the course of the training. Essentially, the training process is separated into several intervals where at the start of each a new dataset is prepared to replace the previous. More specifically, before everything else, the classifier goes through a warm-up phase where it is trained using only the original dataset. Afterwards, for the following intervals, DA is performed on the original dataset by adding a set of synthetic images and then training the classifier with the resulting set. This process is repeated several times, each time replacing the previously added images by new images, until the end of the training [Figure 3.4]. In short, the OnT is the OffT repeated several times for shorter periods and with different datasets. Therefore, there are extra parameters that had to be defined:

- Warm-up - number of epochs, at the beginning of training, to train the classifier without augmentation.
- Number of intervals - number of repetitions of augmented dataset training.

- Epochs per interval - number of epochs in each interval of augmented training.
- Images per interval - number of the images to perform augmentation.

This way, it will be analysed how a dynamic dataset augmentation affects the training of the classifier and whether or not it could be beneficial.

3.6 Development Environment

This section presents that technologies that were used in the development of this work.

The programming technologies used were the following:

- Python - Programming language used to build and test the ELSEGAN framework.
- Keras (Tensorflow) - Library used to build and manage the neural network models (with GPU optimization).
- Scikit-learn - Library used to wrap and manage the training process of the ML models, as well as to test them and calculate metrics.
- OpenCV - Library used to manage image data, namely loading and saving images.
- SciPy - Library used to compute statistical analysis.
- Matplotlib - Library used to plot results.
- Numpy - Library used to manage vector data and perform mathematical computations.
- Pandas - Library used to manage numerical results and statistical data.

The computer programs used were the following:

- Sublime Text - Text editor used to write and manage code
- WinSCP - Tool used for file management in the VPN of the Department of Informatics Engineering.
- PuTTY - Tool used to run the scripts in the VPN of the Department of Informatics Engineering.

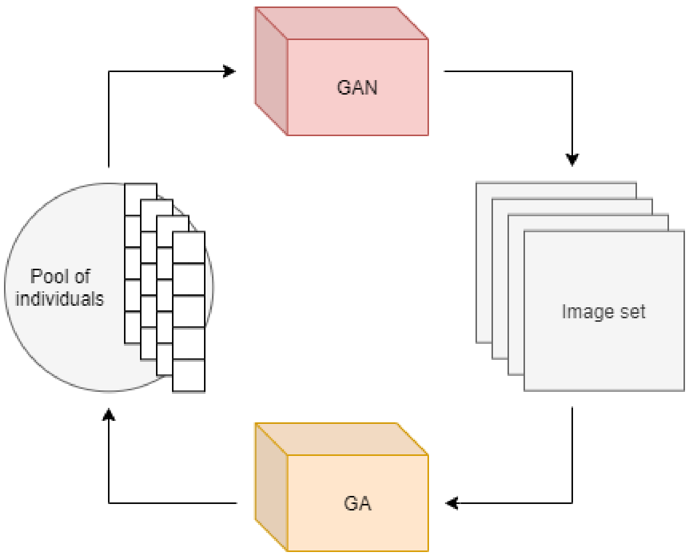


Figure 3.3: Latent space exploration cycle via evolutionary computation.

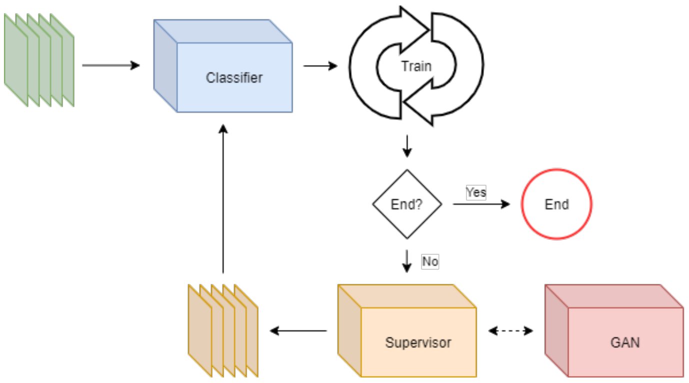


Figure 3.4: Visual representation of the OnT process

Chapter 4

Latent Space Exploration

Contents

4.1 Datasets **17**

4.2 Generator **18**

4.3 Classifier **19**

4.4 Supervisor **22**

4.4.1 Setup 22

4.4.2 Results 24

This chapter details the work that was done in terms of exploration of GANs' latent space. This work was the foundation for a paper submitted and accepted in the EvoStar Evolutionary Machine Learning 2020 [8]. The paper is a research on the exploration of DCGANs' latent space using EC. The experiments will be further explained, but first, the implementation of the DCGAN and the classifier modules will be addressed as well as the results of testing. The workflow took the following order: (i) Creation and testing of the DCGAN module; (ii) Implementation and testing of the classifier module; (iii) Latent Space Exploration.

4.1 Datasets

Throughout this work several datasets were experimented on [Figure 4.1], namely:

- **MNIST** - dataset of closely distributed 0 to 9 handwritten digits separated into sets 60000 digits for training and 10000 for testing with dimensions $28 \times 28 \times 1$ [15]
- **Fashion-MNIST** - dataset of closely distributed of 10 classes of clothing pieces separated into sets 60000 images for training and 10000 for testing with dimensions $28 \times 28 \times 1$ [31]
- **Cats vs. Dogs** - dataset of 25000 evenly distributed images of cats and dogs for training as well as 12500 images for testing with dimensions $64 \times 64 \times 3$;
- **Facity** - dataset of 4204 non-labeled images of faces with dimensions $128 \times 128 \times 3$.

The different datasets were used in order to understand the limits of the framework and how much complexity it was capable to withstand, MNIST being the dataset with the least

Parameter	Setting
latent dimension	100
noise distribution	N(0,1)
optimizer	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
batch size	32
loss function	Binary Cross-Entropy

Table 4.1: DCGAN parameters

complex images and Cats vs. Dogs the one with the most complex. Even though it is not the one with the biggest dimensions, the background of the images from the Cats vs Dogs dataset adds a lot of information that needs to be filtered which makes the task of the DCGAN a lot tougher.

4.2 Generator

In order to perform DA, new images are needed. One way to gather new images is by generating them internally. This work intends to explore the potential of GANs in the generation of new images in order to perform DA. The GANs are trained via unsupervised learning [12] by having a discriminator learn to distinguish the real samples, which come from the input datasets, from the fake samples, that are generated by a generator, and by having the generator learn to produce images that successfully trick the discriminator into classifying them as real samples. With this, the generator is trained into being able to produce samples that closely resemble the training data.

Since the problem at hand is of IC, as stated before section 3.1, DCGANs were used in order to obtain better results. Although, it is important to notice that none of the models were designed for this work. All of them are off-the-shelf models that required only to be trained. For the training of the DCGANs most parameters were fixed [Table 4.1], the only parameters that were shifted were the number of epochs and the shape of the input, which depended on the input dataset. Moreover, since the complexity of the datasets were different, 2 different generators models were used, depending on the dataset: a smaller one for MNIST and Fashion-MNIST and a more powerful one for Cats vs. Dogs and Facity.

Experiments were performed and it became clear the inherent complexity to each dataset. Both MNIST and Fashion-MNIST datasets proved to be relatively simple. With 200 and 800 epochs, respectively, the generator was capable of generating images which very

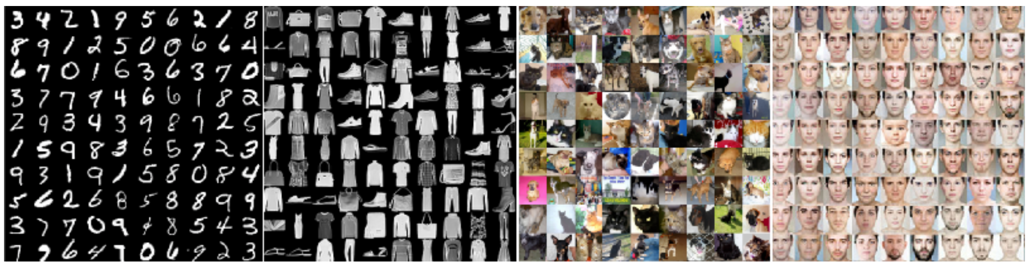


Figure 4.1: Samples of the datasets used in this work. From left to right: MNIST, Fashion-MNIST, Cats vs. Dogs, Facity

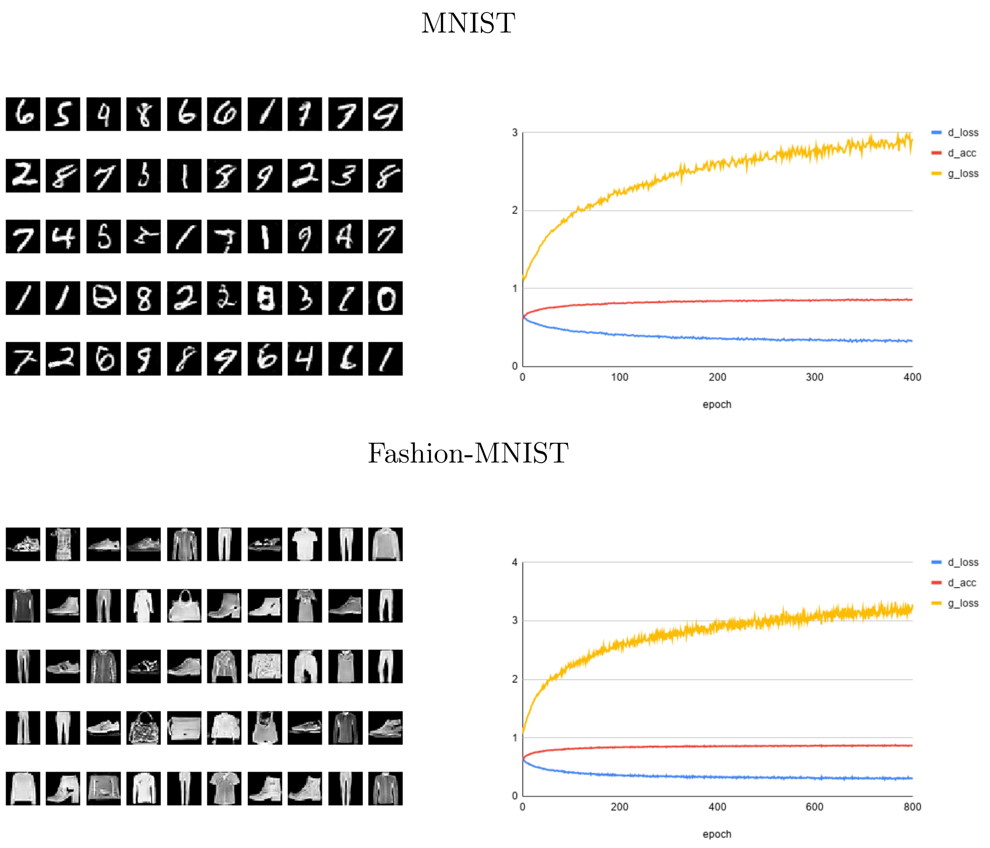


Figure 4.2: On the left: Images generated by MNIST and Fashion-MNIST generators after training. On the right: Evolution of training from MNIST and Fashion-MNIST generators over epoch - d_loss: discriminator loss, g_loss: generator loss, d_acc: discriminator accuracy (0-1).

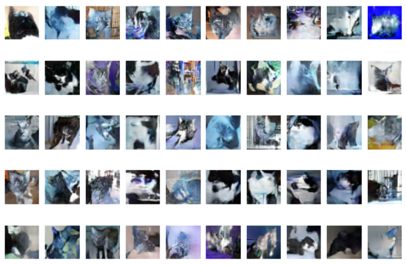
closely resemble the original images, even without separating classes. On the other hand, the generators for the Cats vs. Dogs and Facity were not able to achieve similar results. In the case of the Facity dataset, with 1000 epochs, the generator was able to produce images that, although flawed, resemble faces, but in the case of Cats vs. Dogs, even with 2000 epochs and separation of classes, the images generated harldy resemble animals, although in some it is possible to perceive features that slightly resemble those from cats or dogs. It shows that for these last two situations, some extra work is needed.

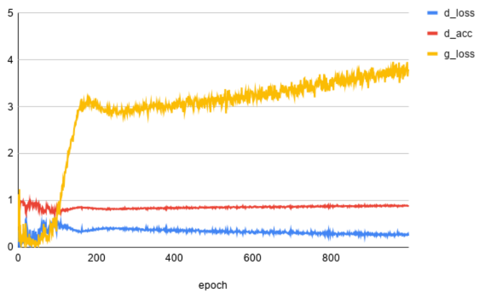
4.3 Classifier

The following module to be developed was the classifier module. It is this module that will allow the verification of the results from the research. In order to maintain cohesion in the work, the model of the classifier used was the the same as the discriminator of the DCGAN. Moreover, the parameters were also kept the same Table 4.2, only changing the epochs and input shape according to the problem to be tested.

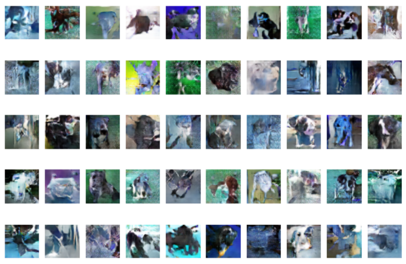
The testing on this module was performed using the MNIST dataset in OffT. This test was also meant to understand how the increase in the number of added images would affect the final performance of the classifier.

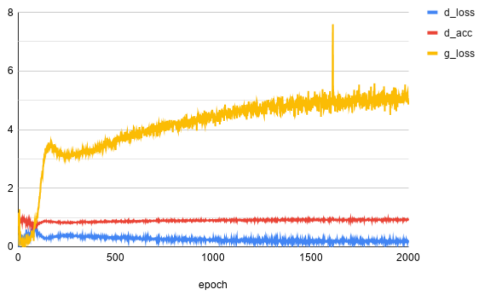
Cats



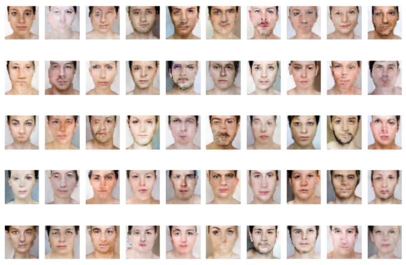


Dogs





Facity



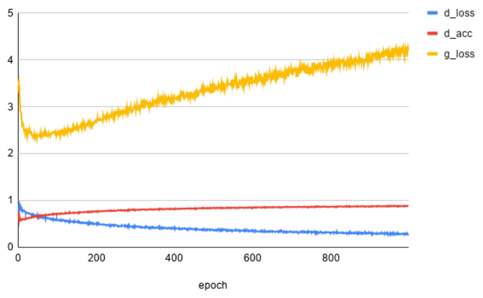


Figure 4.3: On the left: Images generated by cats, dogs and Facity generators after training. On the right: Evolution of training from cats, dogs and Facity generators - d_loss: discriminator loss, g_loss: generator loss, d_acc: discriminator accuracy (0-1).

Parameter	Setting
optimizer	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
batch size	32
loss function	Binary Cross-Entropy

Table 4.2: Classifier parameters

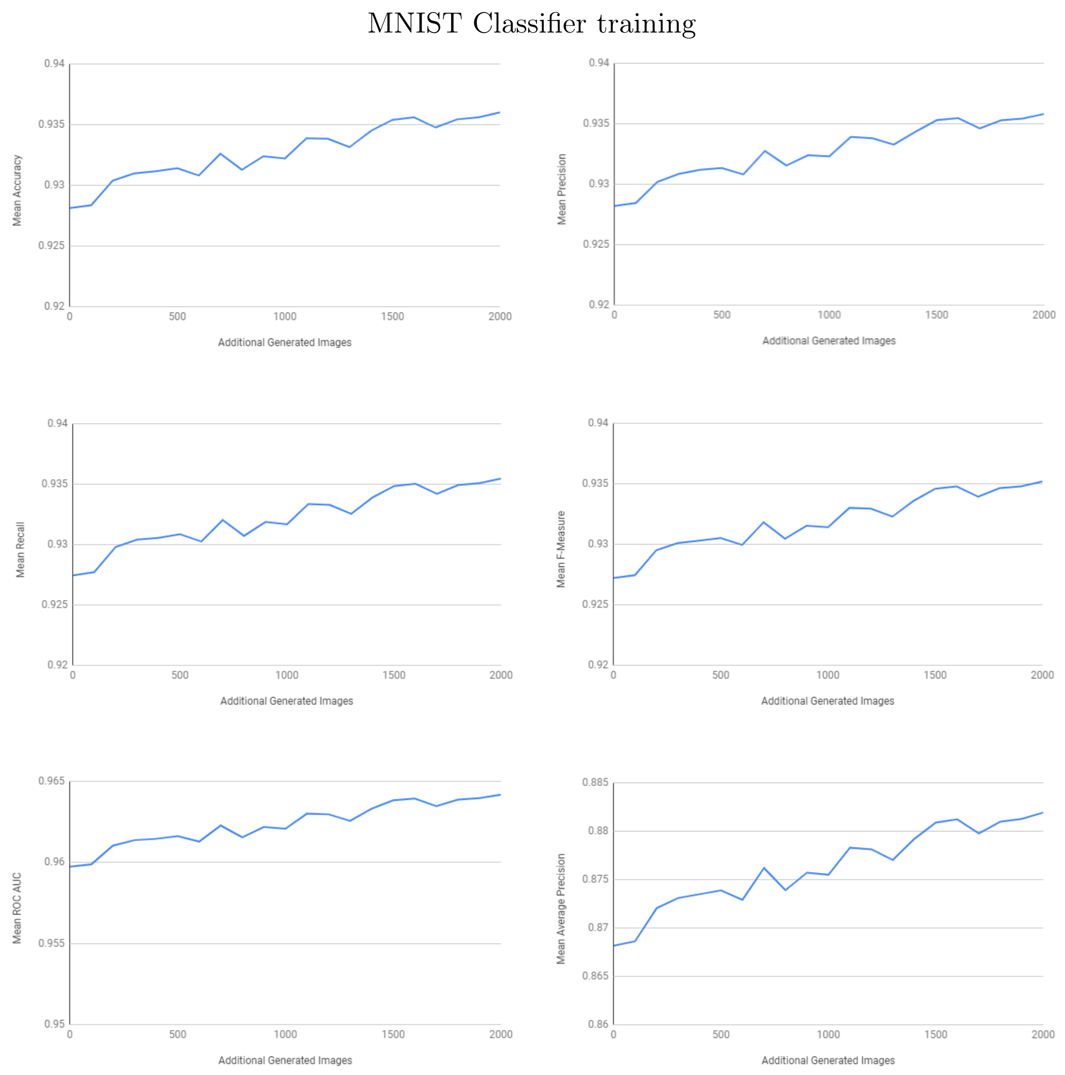


Figure 4.4: Evolution of classifier performance with MNIST dataset comparing addition of generated images incrementally

Beforehand, 10 DCGANs were trained, one for each digit, in order to separately generate images for each class. Since MNIST proved to be an easy problem to tackle, it was decided that the training would only make use of the first 1000 digits of the set (from the original 60000). To train each DCGAN, from the smaller set of 1000 digits, only the ones corresponding to its class were used, which roughly translates to sets of 100 images for each model, although not evenly balanced. Furthermore, the parameters of training were maintained from the previous DCGANs [Table 4.1] with 200 epochs.

The training of the classifiers was made by adding generated images in an incremental way. The first training, the baseline, didn't include any generated images. The following trainings had increments of 100 random images, 10 for each class, from the previous one, up to a total of 2000 new images. Each featured cross-validation using 30 fold Stratified K-fold in order to grant better statistical validation. Each training had a limit of only 20 epochs and the final performance was measured with the full MNIST test set. Even though the results [Figure 4.4] display that adding more images tends to increase performance, it is not possible to say with certainty that this behaviour will prevail without more extensive tests.

4.4 Supervisor

The first experiments with this module focus on the usage of EC to build sets of images according to a certain objective, namely the maximisation of diversity. The experiments included comparison between 3 different EC algorithms as well as 2 distinct metrics of image similarity. The setup was sought the analysis of how different algorithms and distinct metrics affect, for different situations, the end result in terms of diversity, which is the main goal. Moreover, by analysing the observable characteristics of the sets of images obtained, it would also be interesting to see if the diversity measured by an algorithm, was consistent to what humans perceive as a diverse. In this section, the experimental setup is first described and then the results and their discussion are presented.

4.4.1 Setup

The datasets used were the MNIST, Fashion-MNIST and Facity, each one using the DC-GAN described in section 4.2, without any separation between classes. The exploration was performed in sets of 50 images, which translates to individuals having a genotype of size 5000 since the latent dimension of the generators is of size 100. Furthermore, the random initialization of individuals follows the same Gaussian distribution as the prior used to generate fake images in the training of the DCGANs.

As for the exploration of latent space the 3 algorithms used were Random Sampling (RS), Genetic Algorithm (GA) and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites). These three approaches were thought to evolve individuals, but they vary in the way they promote change in each iteration. In the RS approach, a completely new random set of individuals is created at the beginning of each generation and evaluated by an evaluation function, as shown in 1. The GA is an EC approach where we start with a randomly generated population of individuals, but a new population is created trough variation operators (mutation and crossover) which are applied to the individuals of the old population evaluated by a certain fitness function (refer to algorithm 2).

Algorithm 1 Random Sampling

```
1: procedure RANDOMSAMPLING(iterations, popsize, objective, fitnessfunc)
2:     for $i \leftarrow 1$ to $iterations$ do
3:         population = RANDOMPOPULATION(popsize)
4:         population = FITNESSFUNC(population)
5:         best = BEST(population, objective)
       return BEST(population)
```

Algorithm 2 Genetic Algorithm

```
1: procedure GENETICALGORITHM(generations, popsize, problemargs)
2:     population = RANDOMPOPULATION(popsize)
3:     for $i \leftarrow 1$ to generations do
4:         population = VARIATIONOPERATORS(popsize, problemargs)
5:         population = problemargs.FITNESSFUNC(population)
6:         population = problemargs.SORT(objective)
       return BEST(population)
```

The last algorithm, the MAP-Elites, works a little bit differently. It is an illumination algorithm, and it was made for exploring the search space of solutions as much as possible

Parameter	Setting
Population size	50 in RS/GA, initial≤50 in MAP-Elites
Number of generations	500 in RS/GA, 25K in MAP-Elites
Genotype length	50× size of latent dimension
Elite size	1
Selection method	tournament
Tournament size	3
Crossover operator	uniform crossover
Crossover rate	0.7 in GA/MAP-Elites
Mutation operator	random reset
Mutation distribution	N(0,1)
Mutation rate per gene	0.02 in GA/MAP-Elites
feature dimensions	avg similarity : bins=[0:0.01:1], max similarity bins=[0:0.01:1] in MAP-Elites

Table 4.3: RS, GA and MAP-Elites Parameters

[19]. A map of plausible combinations of feature dimensions for the individual's phenotype, which is previously defined, is maintained throughout the training. The algorithm starts by generating a random number of individuals and placing them on the feature map. Afterwards, MAP-Elites, at each iteration, a single new individual is created from applying variation operators to individuals already placed in the map. Each new individual is then evaluated and placed in the map according to its features, granted that each cell of the map, only keeps the best individual, according to the fitness function.

Algorithm 3 MAP-Elites

```
1: procedure mapElites(iterations, initpopsize, map, problemargs)
2:     for $i \leftarrow 1$ to initpopsize do
3:         newind = randomIndividual(problemargs)
4:         placeInMap(newind, map)
5:     for $i \leftarrow 1$ to iterations do
6:         newind = variationOperators(map, problemargs)
7:         placeInMap(newind, map)
       return best(map)
```

So that the results would be more accurate for comparison, the conditions of the algorithms were kept as close as possible, for example, performing a number of iterations on MAP-Elites that would produce the same number of individuals as in the GA [Table 4.3].

About the variation operators, two were used for the experiments: crossover and mutation. In the crossover, the two individuals are chosen from the set of available individuals using tournament selection in the GA and random choice in MAP-Elites. In both algorithms, the crossover method used was the uniform crossover [7]. In the case of the mutation operator, it was used the random reset mutation method, where each gene has a probability of being mutated. The mutation resets the selected genes with completely new values that are taken from the same Gaussian distribution used in the initialization of the individuals.

To measure similarity, the 2 metrics used were Root-Mean-Squared Error (RMSE) and Normalized Cross-Correlation (NCC)[13]. The RMSE metric is calculated as

$$\text{rmse} = 1 - \sqrt{\frac{\sum((A-B)\odot(A-B))}{\text{size}}} \tag{4.1}$$

while NCC is calculated as

$$NCC(A,B) = \frac{\sum((A-B)\odot(A-B))}{\sqrt{\sum(A\odot A)\times\sum(B\odot B)}} \tag{4.2}$$

Where A and B are images, size is a function that measures the size of the images and the $\odot$ corresponds to the Hadamard product between two images. These two metrics were selected because of their fast calculation time and to observe the impact of both since they work in different ways. With RMSE there is a strict and direct pixel by pixel comparison whereas with NCC we are looking for certain contrast in the pixel intensities. These are options in a vast number of different similarity metrics [13]. For the evaluation of individuals, which are sets of images in the form of latent space vectors, the genotype is transformed into images using the corresponding DCGAN [Figure 3.3]. Afterwards, the sets are evaluated through a measure of average similarity. Average similarity consists in taking the measure of similarity between each single image and every other image in the set, using a similarity metric, and then perform the average between all measures. Since the purpose is to maximize diversity, this problem was modelled as a minimisation problem. In terms of the feature dimensions for the MAP-Elites algorithm, both the average similarity and max similarity are used map individuals.

4.4.2 Results

In this section, we analyse the results of the experiments in terms of optimisation of the fitness function and the visual outputs from each algorithm. In order to analyse the results for comparison, we provide graphics with aggregated the values of evaluations. We group the values for Random Sampling and Map Elites in iterations, that corresponds to 50 evaluations. This way, it is possible to compare with the information of the GA. Essentially, one iteration is equivalent to a generation on the GA, which in turn is equivalent to 50 evaluations on Random Sampling and Map Elites algorithms. We also separate the analysis of image diversity in two groups based on the similarity metric: RMSE and NCC.

Observing the results of the different algorithms throughout iterations [Figure 4.5], It is clear that the GA is able to optimise the fitness function. The same does not happen with MAP-Elites, there is little improvement compared to the random sampling, even with different datasets, the difference between them actually gets narrower the more complex is the type of images.

When analysing the MAP-Elites algorithm, the mapping of the individuals is also a matter of concern. As such, in order to understand how much of the space was being explored by the algorithm, the heat-maps of the algorithm across iterations were analysed. This helps understand how the distribution of individuals changes throughout the iterations.

Looking at the results [Figure 4.6], it is possible to see that the space exploration is not really high and that it generally seems to take increasingly more time to expand, therefore, more time being needed find better solutions. It is noticeable the existence of a cluster of solutions in this approach and that the expansion is easier for less fit (dark blue) zones, which corroborates the evolution graphs. It is clear how much MAP-Elites struggles to find better solutions when the mapping is not favourable to the problem. With the facity dataset and the NCC metric, the random initialisation values were clustered into a very small area, which made it really difficult for the algorithm to expand. Even the variation operations caused changes so small that they still fall in the same small area.

It could be that this algorithm isn't fit for this type of problem, but the lack of success is most probably due to the selection of feature descriptor and parametrisation. The mapping functions are, perhaps, too simplistic and are a bottleneck that impedes the ideal exploration of the search space and expansion of the mapped area. This is something that may be further investigated.

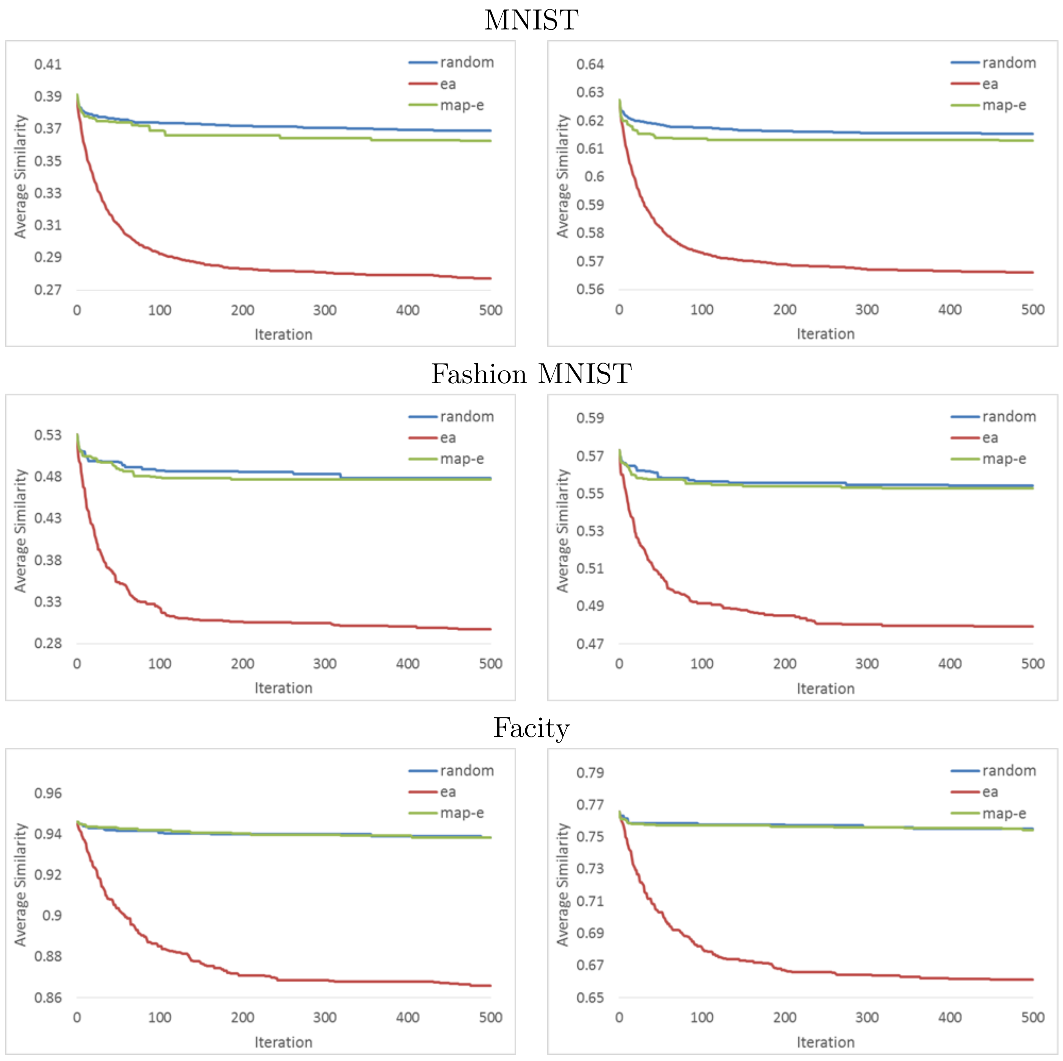


Figure 4.5: Average Maximum values across iterations for the different datasets and similarity metrics: on the left NCC and on the right RMSE. The values are averages of 10 runs.

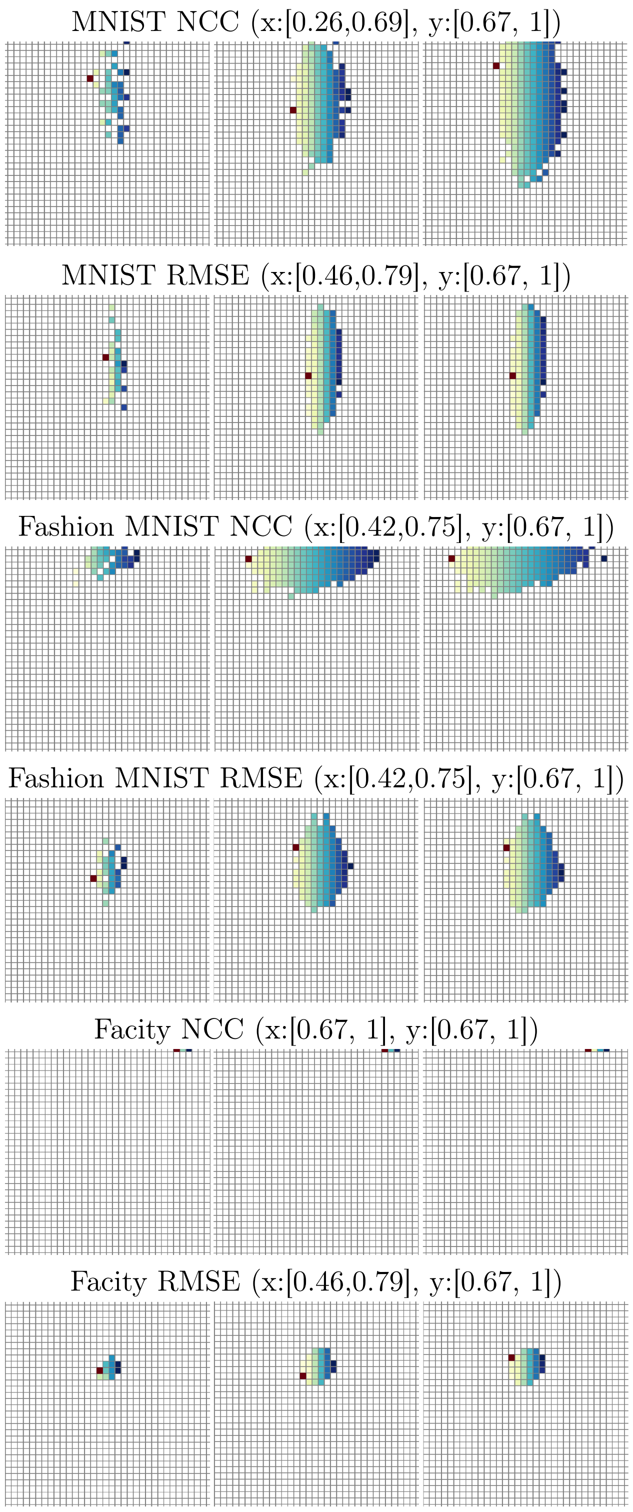


Figure 4.6: Heatmap of the Map constructed with Map-Elites for every dataset and similarity metric. Images correspond to 0th, 12000th and 25000th iterations of the algorithm. The color represent fitness, yellow for better fitness and blue for worse, while the red represents the best individual. The x axis corresponds to the average similarity and the y axis to the maximum similarity

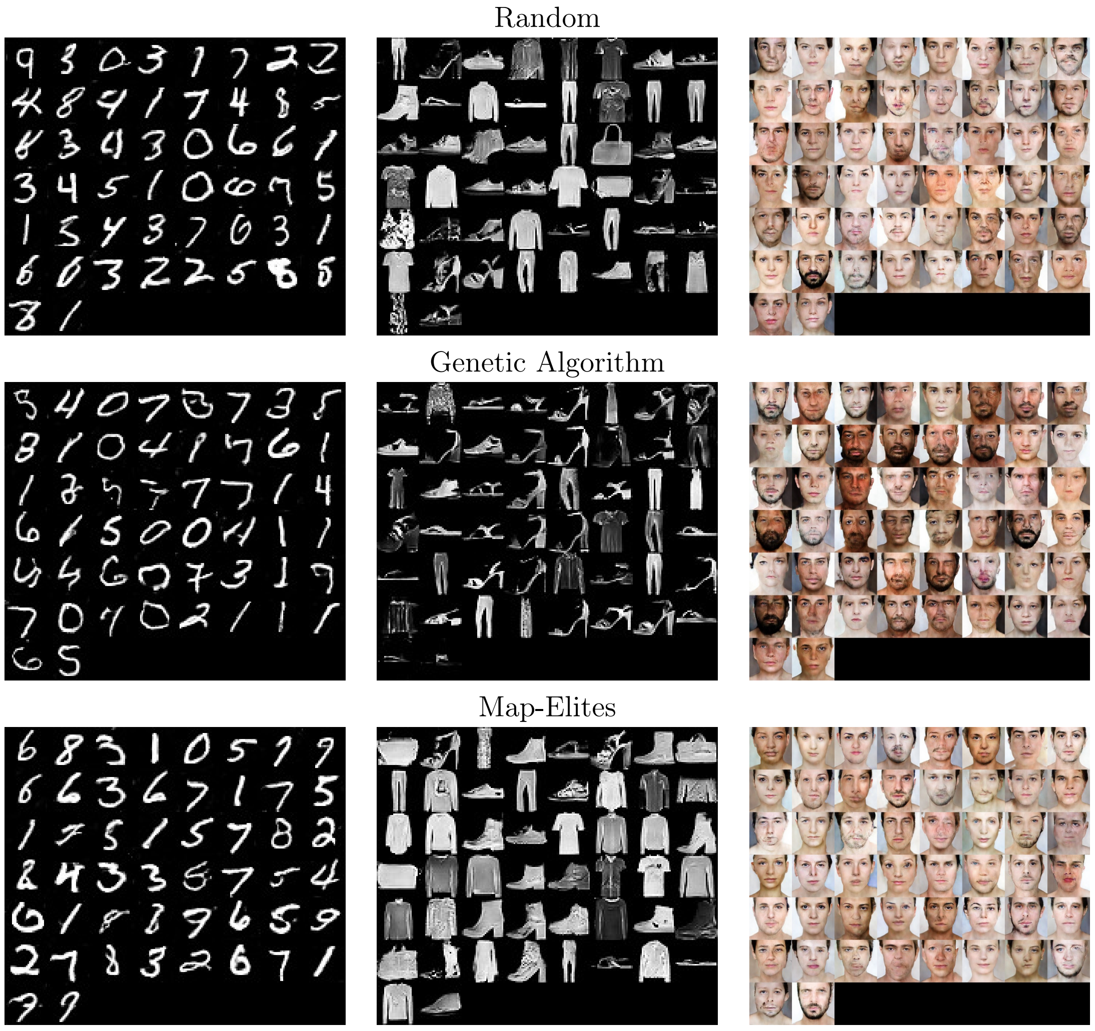


Figure 4.7: The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the NCC metric.

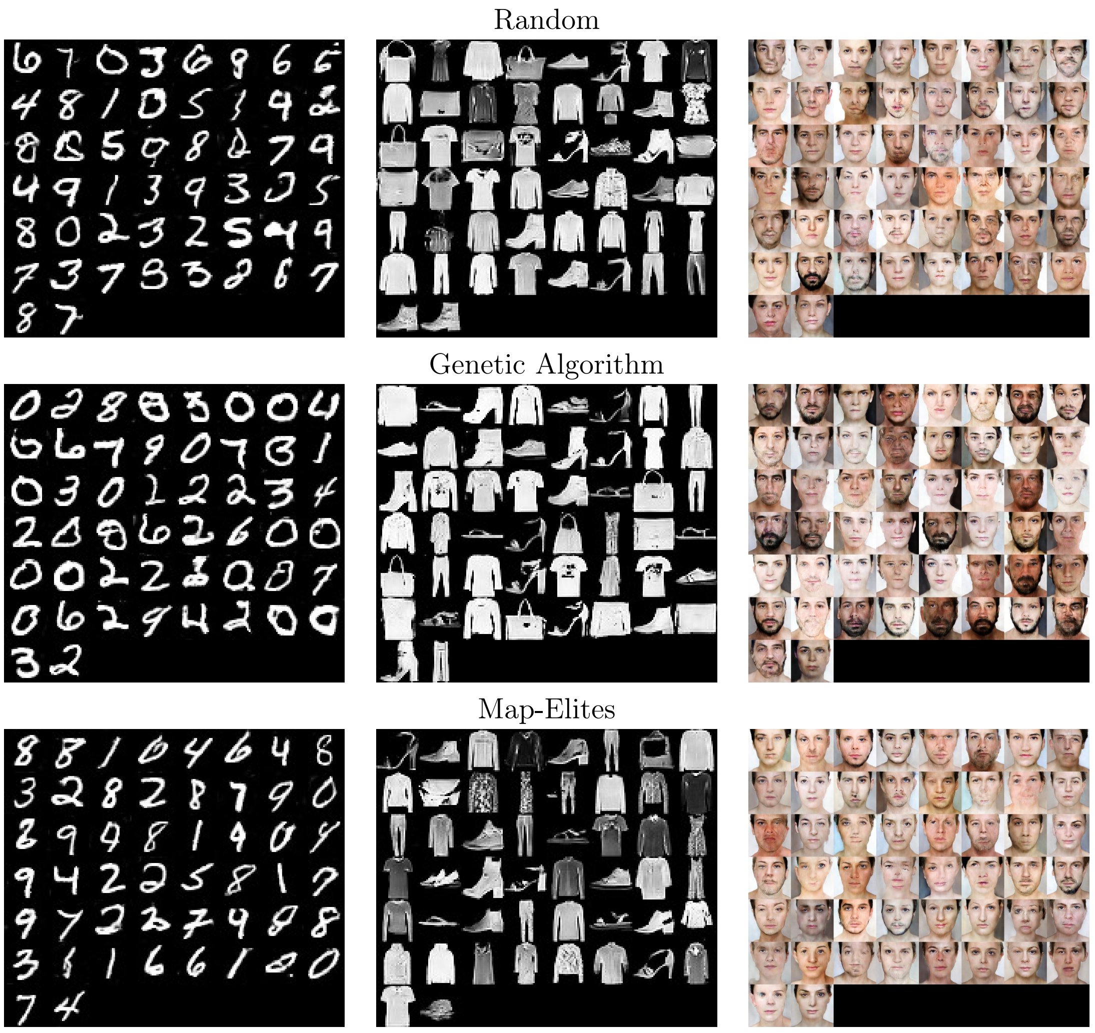


Figure 4.8: The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the RMSE metric.

In terms of visuals, the outputs [Figure 4.7 and Figure 4.8] showcase the best set of images that maximise the pre-determined criterion. For this last analysis, the focus is on the NCC and RMSE metrics. The differences between different approaches and metrics are easily noticeable. In general, each approach, for each metric, ended up having a different set of images in the end.

Between the NCC and RMSE results, one aspect is noticeable, the results showcase the particularity of each similarity metric. In the RMSE, it is noticeable that it tends to pick a set of variables that generated images which minimise the overlap of elements and dissimilar background (facity). The NCC, on the other hand, tends to promote contrast between the different images. This is a clear example of the success of the approach of searching and achieving the pre-determined objective. Among the different approaches, the GA was able to generate the most visually diverse set of images.

Chapter 5

Latent Space Exploration for Classifier Improvement

Contents

5.1 Dataset: Human Sperm Head Morphology (HuSHeM) 30
5.2 Generative Adversarial Networks 31
5.3 Offline Training Experiments 31
5.3.1 Supervisor 32
5.3.2 Classifier 33
5.3.3 Experimental Results 34
5.4 Online Training Experiments 40
5.4.1 Classifier 40
5.4.2 Experimental Results 41

This chapter details the experiments performed in the application of the ELSEGAN to DA for a real world problem. To this end, the preliminary work of this chapter was submitted and accepted in a paper for the Applied Deep Generative Networks (ADGN) workshop at European Conference on Artificial Intelligence (ECAI) 2020 [9]. In this paper we performed the first experiments on the Human Sperm Head Morphology (HuSHeM) dataset. The study exhibited promising results of training using an ELSEGAN DA approach over training with just the original dataset. In this chapter, the experiments go more in-depth than the ones performed in the paper. The workflow took the following order: (i) Completion of the ELSEGAN framework; (ii) Preparation of the HuSHeM dataset; (iii) Training of the DCGAN models; (iv) Evolving sets of images through supervision; (v) Training and testing of classifiers and analysis of results.

5.1 Dataset: Human Sperm Head Morphology (HuSHeM)

The dataset used for the experiments was the HuSHeM [24]. In the bio-medicine context, sperm morphology analysis is a critical factor in the diagnosis process of male infertility. The dataset is sub-divided into 4 classes of sperm head images [Figure 5.1]: Normal (54 instances), Tapered (cigar-shaped, 53 instances), Pyriform (pear-shaped, 57 instances) and Amorphous (various deformities, 52 instances) for a total of 216 images. Due to its small size, it is ideal to test our framework. The dataset was not previously sub-dived, so the decision was made to have 40 images of each class in the training set and keep the remaining

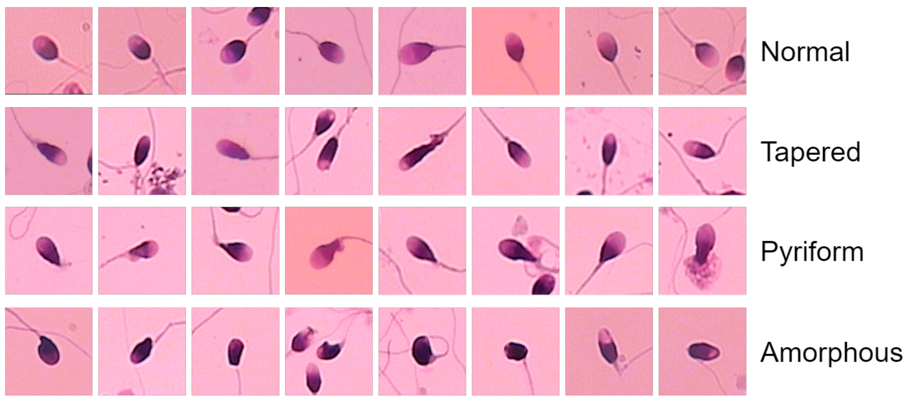


Figure 5.1: Class examples for each of the HuSHeM classes

Table 5.1: Generative Adversarial Network training parameters

Parameter	Setting
latent dimension	100
noise distribution	N(0,1)
optimiser	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
epochs	20000
batch size	32
loss function	Binary Cross-Entropy

in the test set. The images in the following experiments have the dimensions of 132x132 and RGB colour model.

5.2 Generative Adversarial Networks

As stated in section 3.1, DCGANs were used to train generators for the ELSEGAN, in a similar fashion to what was done in section 4.2. The training was performed in unique classes, meaning that 4 distinct models were trained, which are the models that will be used to generate images for each and every test that will be performed next. Each training had as input the 40 corresponding instances of the training set for its class. In the same way as in the latent space exploration experiments [section 4.2], the models used for both the generator and the discriminator are off-the-shelf designs since they are not the object of this study. The training parameters were set, as shown in Table 5.1.

In Figure 5.2 it is possible to see that the trained generators are able to produce good quality images that exhibit features resembling the original instances from its class.

5.3 Offline Training Experiments

The OffT is the most common type of training of Machine Learn models, where a static dataset is prepared and given to the model to train. With DA, a set of images previously is prepared and merged with the original dataset forming a new static dataset that maintained until the end of the process of training.

In this section, first the setup of the supervision process for finding sets of images will be

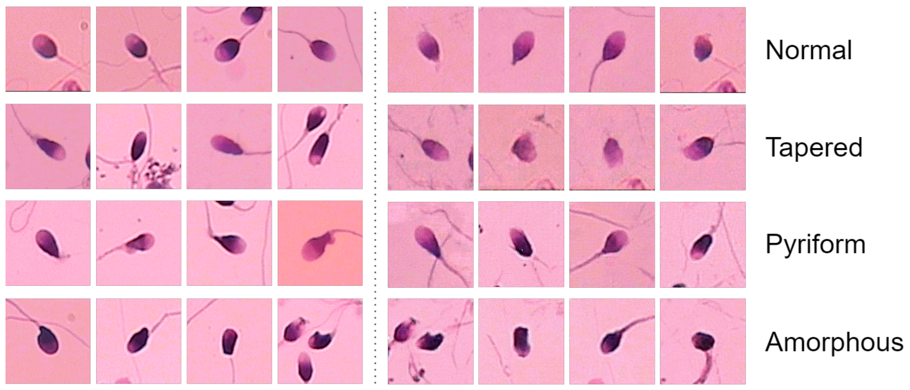


Figure 5.2: Class examples for real (left column) and generated (right column). On each row from, top to bottom: Normal, Tapered, Pyriform, Amorphous

Table 5.2: Genetic Algorithm Parameters

Parameter	Setting
Population size	60
Number of generations	40000
Genotype length	number of images $\times$ GAN's latent dimension
Elite size	1
Selection method	tournament
Tournament size	3
Crossover operator	2-point
Crossover rate	0.7
Mutation operator	random reset
Mutation distribution	N(0,1)
Mutation rate per gene	0.05

explained as well as the setup for the classification process. For supervision, two different criteria will be used, resulting in two distinct sets of images with which to perform DA. Afterwards, the results, both of supervision for each type of DA and classification, will be analysed.

5.3.1 Supervisor

The supervisor module is a key element of this work. It will allow the optimisation of the DA process so that better performance results might be reached.

The supervision is performed through latent space exploration using EC. From the results obtained in section 4.4, the decision was made to proceed with the experiments utilising the GA to motor the optimisation process, in a similar fashion. Each individual in the population of the DA represents a set of images which are coded into its genetic code in the form of latent vectors. At the end of evolution, for each criteria, the best individual will be kept to explore DA

The parameters of the genetic algorithm were defined as in Table 5.2.

The fitness function is the guide of evolution and consequently of supervision. Two criteria (fitness functions) for supervision will be tested, namely image similarity and classifier loss, to find two distinct sets for DA, which will be compared between one another. The main difference between these criteria is that the first should push for intra-class diversity while the latter should push for inter-class similarity.

Table 5.3: Classifier parameters for supervision

Parameter	Setting
optimiser	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
epochs	250
batch size	32
loss function	Binary Cross-Entropy
validation	20%

Supervision criteria: Image Similarity

This criteria of image similarity is identical to the one used in section 4.4, but slightly modified. The fitness of each individual is calculated by averaging the similarities between each image in the set and the centroid image of the set that includes the images from that individual together with the images from the original dataset. It was decided that for these experiments, the similarity between pairs of images was to be calculated using the NCC metric. The calculations are as follows:

$$T = I^\frown O \tag{5.1}$$

$$C = \frac{\sum_{t \in T} t}{length(T)} \tag{5.2}$$

$$F = \frac{\sum_{i \in I} NCC(i, C)}{length(I)} \tag{5.3}$$

T is the set resulting of concatenation of the images from the individual (I) with the images from the original dataset(O). C is the centroid of the set T. F is the fitness of the individual calculated through averaging the similarities measured using the NCC, calculated as in Equation 4.2.

Since this fitness function measures similarity instead of diversity, the objective of the algorithm is, as before, set to minimisation.

Supervision criteria: Classifier Loss

With the idea that sets of images that cause a greater loss on a classifier will be more useful when added to the training of the classifier, this criterion uses a classifier trained on the original dataset to find sets of images that maximise its loss, the error. This value of loss will correspond to the fitness of the individual.

The model used for the classifier was the same off-the-shelf model used for the discriminator of the DCGANs [section 5.2], with the exception of being multi-class.

The training parameters for the classifier used in evaluation were Table 5.3

5.3.2 Classifier

The classifier module allows the assertion of experiment results. The model used for the classifier was the same off-the-shelf model used for the discriminator of the DCGANs

Table 5.4: Classifier parameters for Offline Training

Parameter	Setting
optimiser	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
epochs	150
batch size	32
loss function	Binary Cross-Entropy
cross-validation	Stratified
folds	5

[section 5.2], with the exception of being multi-class. The parameters used for the training of the classifier are as in Table 5.4. The training ensures that the classifier reaches a point of plateau, where there should be no further gain in performance. This helps to verify the quality of our solution when new samples are added to the training set. On another note, the training also includes cross-validation for every test for statistical purposes.

5.3.3 Experimental Results

To test the ELSEGAN, a comparison was made between the performance of the classifiers before and after performing DA. Furthermore, three forms of DA using GANs were also compared: (i) randomly; (ii) supervised by image similarity; (iii) supervised by classifier loss. The evaluation of each classifier is conducted in the test dataset, where several metrics were measured, namely Accuracy, Precision, Recall, F1-score, Area Under Receiving Operator Characteristic Curve (AUROC) and Average Precision.

First the sets of images which required supervision were prepared. The decision was made to double the size of the training set, meaning 50% of original images and 50% generated images. Therefore, for each criteria, image similarity and classifier loss, a set of 40 images was evolved, for each class, in a total of 8 sets, that would be used to train the classifiers.

Looking at the evolution results for the image similarity criterion [Table 5.5] [Figure 5.3], it is seen that the absolute variation in fitness between the first and last generations is not high, the greatest difference was in the Tapered class and only about 0.51%. Although, since all values, for every class, stand in a range of just 1% between 99% and 100%, it means that that the image similarity is very high according to the metric used. Likewise, even a small difference such as this one seems to promote good evolution, for all classes. Although, after taking a look at Figure 5.5 it is observable that the algorithm was not very successful in creating a set of dissimilar images, which is mostly perceptible in the Pyriform and Tapered classes where we see repetitions of the same images. This is most probably due to the usage of a centroid to represent the class. Looking specifically at the Pyriform class, roughly half of the set is the same image. An explanation to this would be that this image is really dissimilar to the centroid. As such, the repetition of this images will minimize the similarity.

As for the results of the evolution according to the classifier loss criterion [Table 5.5] [Figure 5.4], the first thing that stands is the disparity in fitness between images of different classes. The Amorphous class proves to be the class that causes the greatest loss with a vast difference to the Tapered and Pyriform classes. It is also seen that, in all cases, there is a peak of evolution in the beginning with small increases throughout the remaining of the process. On a further note, looking at Figure 5.5 there is not much diversity between the images in the set, and, similarly to what was experienced with the image similarity

Table 5.5: Fitness value of the first and last best individuals according to the Image Similarity and Classifier Loss criteria for each class of the problem.

	Generation	
	First	Last
Similarity		
Normal	0.9975	0.9950
Tapered	0.9951	0.9900
Pyriform	0.9956	0.9934
Amorphous	0.9937	0.9890
Loss		
Normal	5.842	6.598
Tapered	1.673	2.066
Pyriform	1.476	2.294
Amorphous	8.124	9.595

criterion, there are a repetitions of images, which can be argued that are the images that are more easily mistaken by other classes. The classifier loss metric finds, in fact, images that show inter-class similarities.

The following step, was the training of the classifiers. First we started with the training with the original dataset and the proceeded with the training with the augmented datasets. The duration of the training, 150 epochs, ensures that no benefits would be gained with further training. Each training was repeated 30 times for statistical analysis.

Looking at the results of the testing phase [Table 5.6] [Figure 5.7] it is observable that, in general, all the augmentation approaches provided an improvement in performance when compared with the original approach. However, the supervised approaches did not perform better than the random approach. The augmentation through image similarity supervision performed, on average, worse than the random augmentation across all metrics. On the other hand, the augmentation guided by classifier loss presents average values which are always higher than the Random augmentation, and with less deviation across runs. The results in Table 5.7, confirm that the DA does improve, with statistical significance, the performance of the classifiers. The Kruskall-Wallis test confirms that there are, in fact, statistical differences between the approaches. The paired Mann-Whitney tests, show that the Original training is statistically inferior to the other approaches. On another note, when testing the classifier loss DA against the random DA, mixed results are obtained for different metrics. Some show statistical difference between the two approaches, as is the case of the F1-score, while others like the AUROC do not.

After the previous results, more tests with random DA were performed in order to explore the addition of different amounts of images to the training set, both by decreasing and increasing. The experiments go from 8 additional images per class (total of 1 batch) to 216 images per class (total of 27 batches) in intervals of 16 images per class (total of 2 batches). Each experience was repeated 30 times.

Figure 5.8 shows the effect of the addition of the different amounts of images according to the F1-score metric, with a similar behaviour in the other metrics. The average performance of the classifier grows as the number of images is increased, until reaching a peak

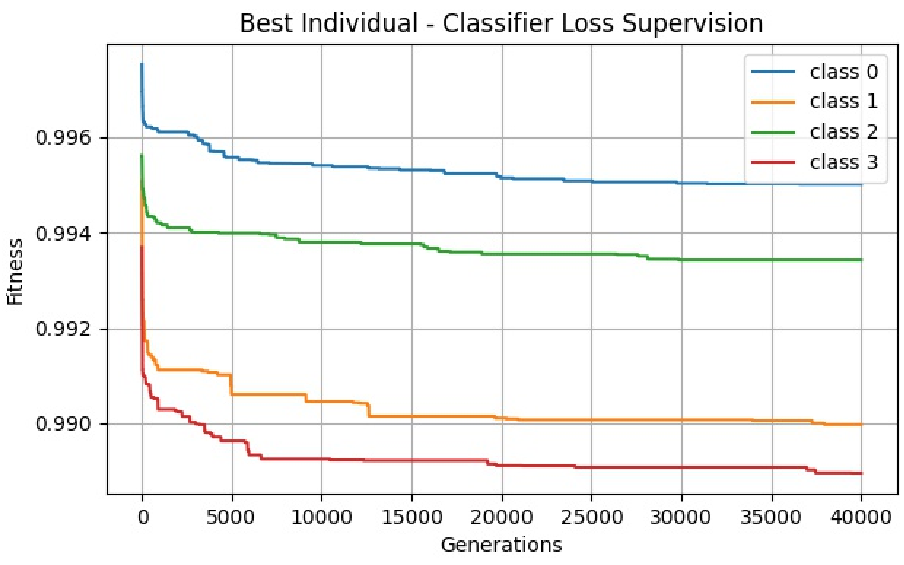


Figure 5.3: Evolution of the best individual fitness with the image similarity criteria for each class of the problem: 0-Normal, 1-Tapered, 2-Pyriform, 3-Amorphous

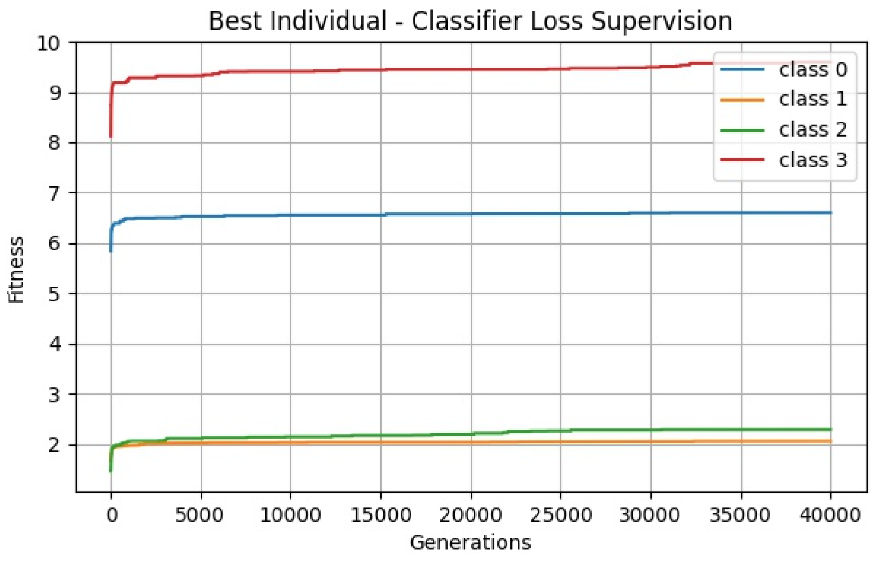


Figure 5.4: Evolution of the best individual fitness with the classifier loss criteria for each class of the problem: 0-Normal, 1-Tapered, 2-Pyriform, 3-Amorphous

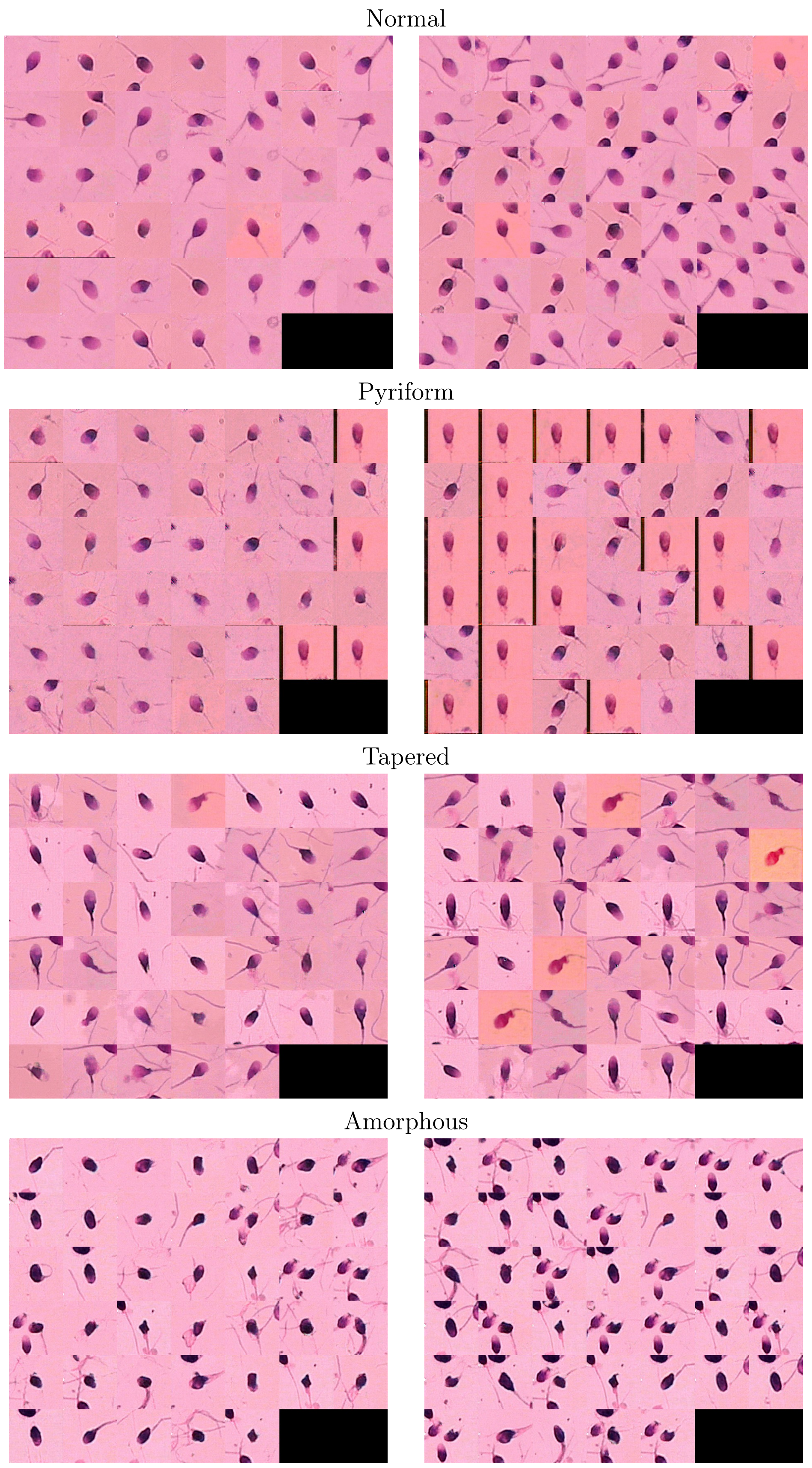


Figure 5.5: Best individual in the first (left) and last (right) generations for each class, by the image similarity criteria.

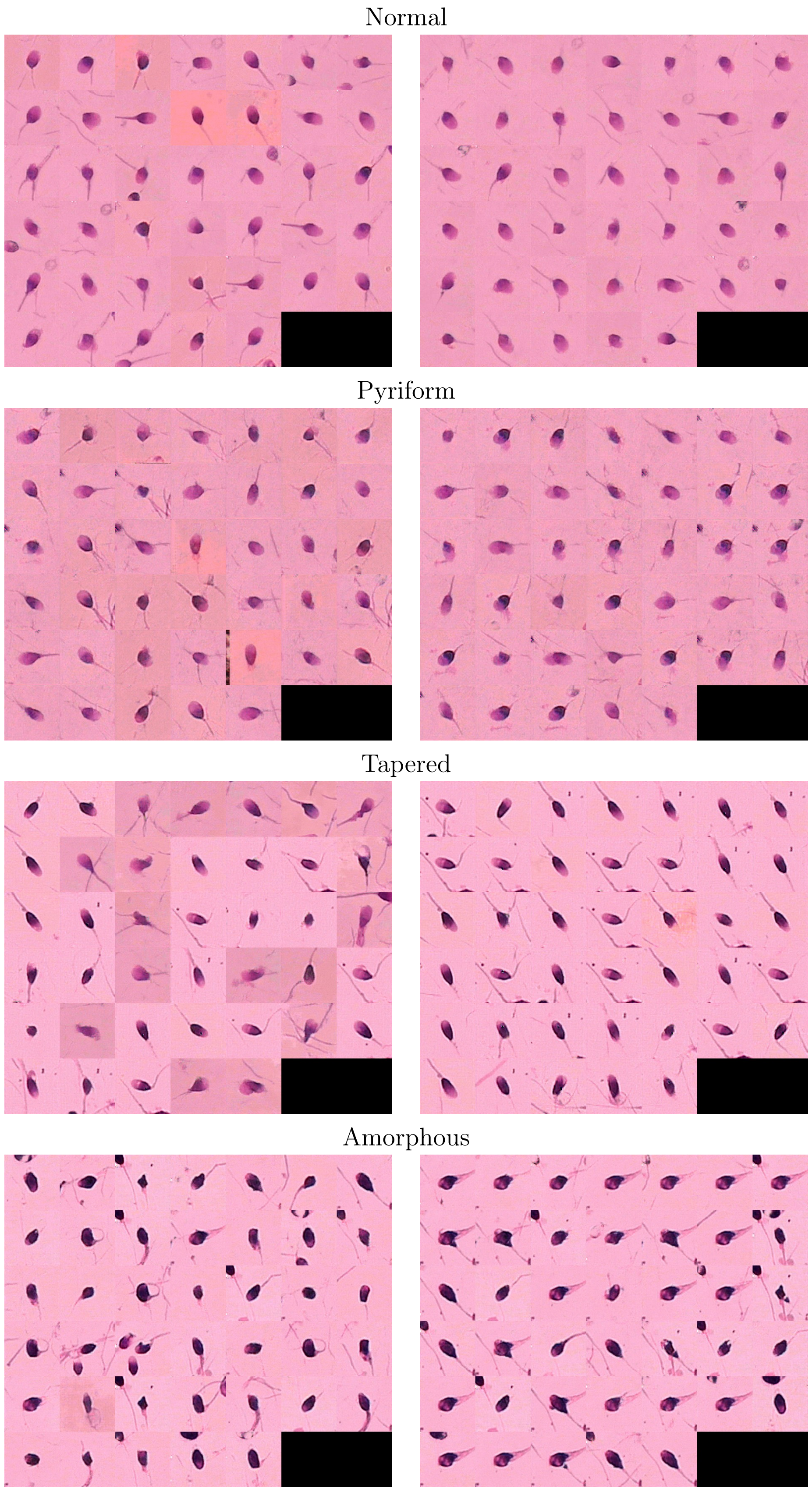


Figure 5.6: Best individual in the first (left) and last (right) generations for each class, by the classifier loss criteria.

Table 5.6: Classifier performance in testing phase across all metrics measured. Performance comparison of mean, standard deviation (std.) and max values between training with original dataset, with randomly augmented dataset, with augmentation by supervision through image similarity and with augmentation by supervision through classifier loss

	Metrics					
Data	Acc	Prec	Rec	F1	AUROC	Avg-Prec
Mean						
Original	0.632	0.678	0.645	0.645	0.761	0.546
Random	0.680	0.696	0.691	0.680	0.792	0.585
Similarity	0.656	0.687	0.666	0.666	0.775	0.562
Loss	**0.682**	**0.720**	**0.696**	**0.692**	**0.795**	**0.593**
Std.						
Original	0.015	0.015	0.016	0.017	0.011	0.017
Random	0.023	0.021	0.021	0.021	0.014	0.021
Similarity	0.016	0.016	0.016	0.017	0.011	0.017
Loss	0.015	0.011	0.015	0.014	0.010	0.015
Max						
Original	0.661	0.713	0.675	0.680	0.781	0.581
Random	**0.725**	**0.738**	**0.730**	**0.726**	**0.818**	**0.634**
Similarity	0.696	0.723	0.705	0.707	0.801	0.603
Loss	0.707	0.737	0.722	0.715	0.812	0.617

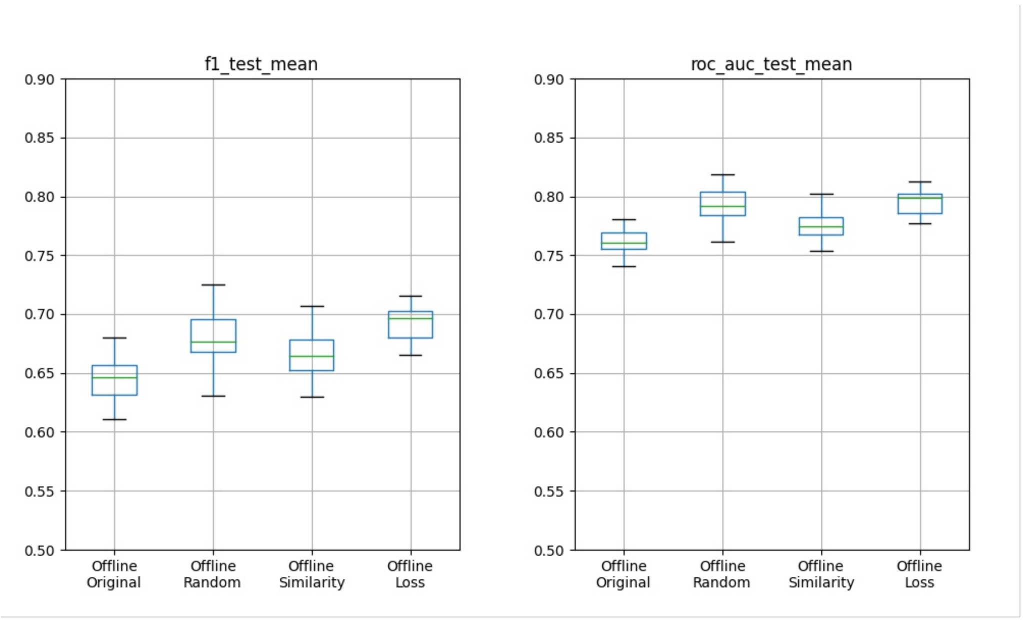


Figure 5.7: Visual comparison, via F1-score (left) and AUROC (right) metrics, of classifiers' performance, over 30 runs, between offline trainings with original dataset, with randomly augmented dataset, with augmentation by supervision through image similarity and with augmentation by supervision through classifier loss.

Table 5.7: Statistical comparison between the different DA approaches for the average results of the F1-score and AUROC metrics. O: Original (no DA; IS: image similarity DA; CL: classifier loss DA; R: random DA)

		Metrics			
		F1-Score Mean		AUROC Mean	
Classifiers	Test	Statistic	P-value	Statistic	P-value
All	Kruskall-Wallis	61.631	2.635e-13	71.276	2.275e-15
O vs IS	Mann-Whitney	178.0	2.985e-05	156.0	7.149e-06
O vs CL	Mann-Whitney	10.0	4.076e-11	6.0	2.747e-11
R vs CL	Mann-Whitney	269.0	0.004	409.0	0.275

Table 5.8: Classifier parameters for Online Training

Parameter	Setting
optimiser	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
batch size	32
loss function	Binary Cross-Entropy
cross-validation	Stratified
folds	5
warm-up	15
number of intervals	15
epochs per interval	10

at around 120 additional images per class, after which it starts declining. 120 additional images per class corresponds to a ratio of 3:1, 3 generated images to 1 original image, in the training set. In the future, it would be interesting to explore if a supervised augmentation approach would reach similar results.

5.4 Online Training Experiments

The OnT, distinguishes itself from the OnT in the sense that the dataset used to train is dynamic instead of static. In this case specifically, we separate the training into several intervals where at the start of each we augment the original dataset with a new set of images, replacing the previously added set of generated images.

After the results observed in the OffT, where the supervision approaches did not surpass in performance the random approach, supervision was not applied to the OnT. Therefore, the OnT experiments will only feature random DA.

5.4.1 Classifier

In the same way as was done OffT, the classifier will measure the performance of the approach, using the same off-the-shelf DCNN model as well. The parameters used for the training of the classifier are also similar to the those of the OnT in order to better compare the 2 approaches and are defined as in Table 5.8. The training time ensures that

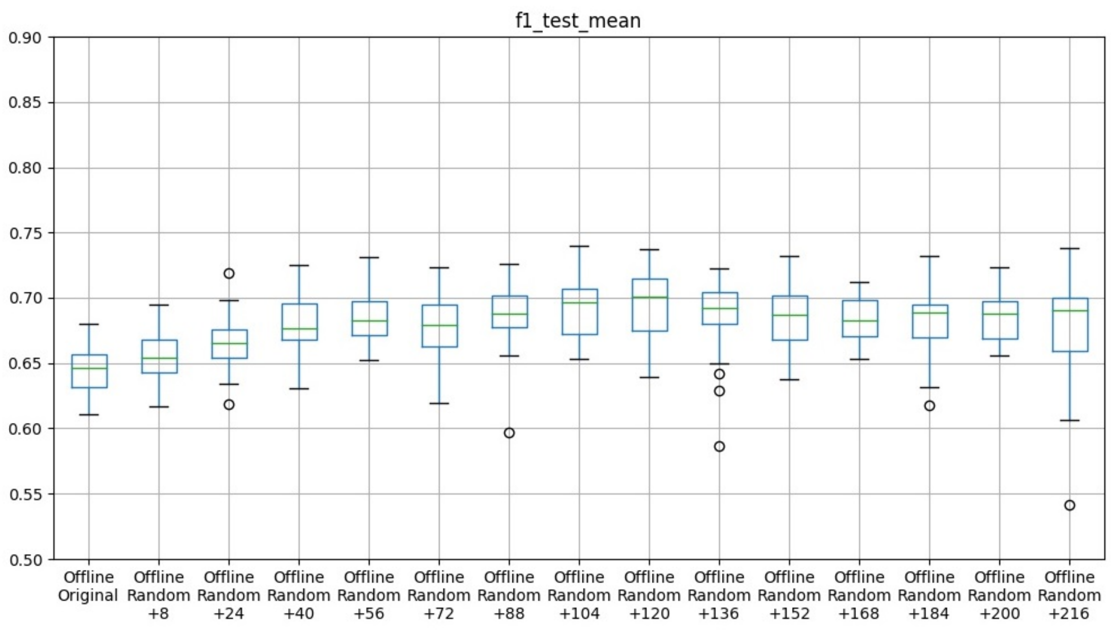


Figure 5.8: Visual comparison, via F1-score metric, of classifier's performance, over 30 runs, between offline trainings with randomly augmented datasets with different amounts of added images, from 8 additional images per class to 216 additional images per class.

the classifiers reach a point of plateau.

5.4.2 Experimental Results

In the same way we did several tests of different sizes of DA for the random DA in OffT, we also explored several amounts of DA in the OnT, namely from 24 images per class (total of 3 batches) up to 120 images per class (total of 15 batches) with intervals of 16 images per class (total of 2 batches), to observe how the classifier would perform in each of these situations, but also to compare directly with the OffT. All the trainings were performed with a warm-up of 15 epochs followed by 15 intervals of 10 epochs, and repeated 30 times for statistical analysis.

As it is possible to observe in Figure 5.9, the behaviour with increasing the number of images in the dataset is not the same in the two types of training. While the OffT had its best performance at around 120 additional images per class, the OnT peaked at 40 additional images per class. Another fact that is also recognizable is that the OnT also did not surpass the OffT in any of the tests.

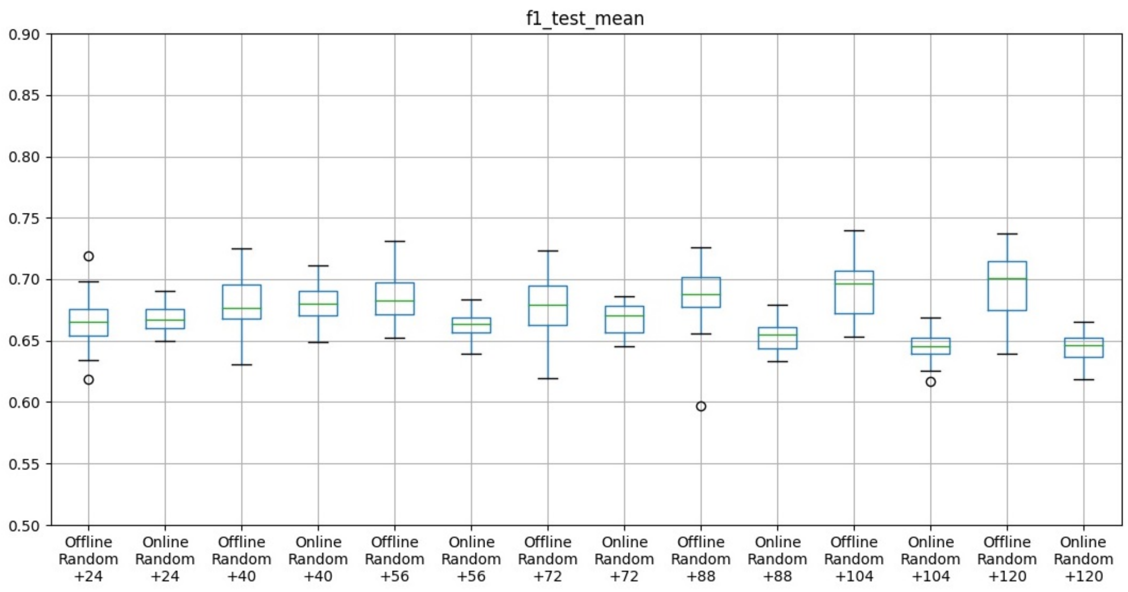


Figure 5.9: Visual comparison, via F1-score metric, of classifier's performance, over 30 runs, between OnT and OffT trainings with randomly augmented datasets with different amounts of added images, from 24 additional images per class to 120 additional images per class.

Chapter 6

Conclusion

In ML the quality of a dataset with which an algorithm is trained directly impacts its performance. As such, besides having a good algorithm, it is fundamental to gather a good dataset. A model is only as good as its training dataset allows it to be. The problem arises when datasets are faulty in terms of representativeness and completeness. In order to solve this, DA is crucial. There are several ways to perform DA in IR problems, which include creation of new images through transformation of existing ones or generation of new images through merging of samples, using learning the distribution of the original dataset or applying an evolutionary computation engine. The usage of GANs, which are models that learn the distribution of the training dataset and generate synthetic images accordingly, to perform DA is something that has not seen much research yet.

This dissertation explored the hypothesis that GANs may be an effective way to perform DA on incomplete datasets. The overall revolves towards the improvement of image classifiers through the usage of GANs to generate samples to be added to the classifier's training set. As such, this work suggests a 3 step framework for DA, applied specifically to the task of IC, the ELSEGAN. This framework is composed, of a GAN to generate new images, a supervisor that uses EC to manage the generation and selection of images, and a classifier, a DCNN to measure the performance. With this framework it also became possible to experiment no a new type of training, the OnT which, opposed to the common static OffT, featured dynamic DA with a partitioned training process.

The first experiments, which were focused in the generation of images and exploration of the latent space, that we applied to Cats vs Dogs and Facity datasets. The results of the GAN raised some concerns about the dependency on the quality of the generator in these datasets. Despite that, the evolutionary approaches showed promising results in the supervision process for building sets of diversified images through latent space exploration.

When stepping into the real-world application scenario of the HuSHeM dataset, we were able to train GANs that generated good quality images for its corresponding class which took out the concerns left from the previous experiments. The supervision process was focused on using a GA to perform optimisation and explore the latent space of the GANs. Two criteria for optimisation were tested. The first criterion makes use of an image similarity metric, namely the NCC, to find a set of images as distinct as possible between each other and the images from the original set. The second criterion requires an already trained classifier to find a set of images that maximise its loss. The results of supervision for both criteria showed repetition of several images, which means that each of them still needs work to present optimal results.

Afterwards, the performance of trained classifiers was analysed and compared between classifiers trained without augmentation, with random augmentation and with supervised augmentation according to both the image similarity criterion and classifier loss criterion. Experiments featured cross-validation as well as several repetitions to take statistical measures in the calculated performance metrics. Overall, all the DA approaches to training performed better than the original, although the supervised ones did not improve on the random DA, which might due to the supervision process results. The image similarity approach was able to improve the performance by around 2.5%, on average, while the random and classifier loss was able to improve about 5%. Beyond that, additional tests were performed with the random augmentation with an increased number of added instances. The performance increased in some instances up to around plus 6%. This leaves the question of whether or not the classifier loss approach would follow the same results.

Beyond this, a new type of training was introduced and experimented on, the OnT. This type of training consists of partitioning the process of training of the classifier into several intervals and performing a dynamic data augmentation by feeding the model with a new set of synthetic images at each interval. Due to the results obtained in the OffT, this was only tested with the random augmentation. Results show equivalent performance when comparing the two, but only for a small number of added images. As the number of images increases, both the relative and absolute performance diminishes.

To conclude, the tests performed, mostly those of the OffT, show that GANs may be a viable solution for the problem of DA, since, in fact, the addition of synthetic images generated by the ELSEGAN was able to effectively improve the performance of the classifiers.

A lot of work can still be made in the future on this topic. The results show that the hypothesis of using GANs to perform DA might be a viable solution. Future work may feature exploration of other datasets as well as new and/or more refined supervision techniques. In the Offline, it would be interesting to perform an exploration with supervised augmentation similar to the one performed with random augmentation. An idea for OnT is to include the classifier being trained in the classifier loss supervision to find sets of images to add to its own training dataset. Finally, an important step to validate this work will be to compare the ELSEGAN approach with other data augmentation approaches.

References

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks, 2018.

[2] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks, 2017.

[3] Philip Bontrager, Wending Lin, Julian Togelius, and Sebastian Risi. Deep interactive evolution. In Antonios Liapis, Juan Jesús Romero Cardalda, and Anikó Ekárt, editors, *Computational Intelligence in Music, Sound, Art and Design*, pages 267–282, Cham, 2018. Springer International Publishing.

[4] João Correia. *Evolutionary Computation for Classifier Assessment and Improvement.* PhD thesis, University of Coimbra, 2018.

[5] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data augmentation using gans, 2019.

[6] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection, 2017.

[7] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[8] Paulo Fernades, João Correia, and Penousal Machado. Evolutionary latent space exploration of generative adversarial networks. In *EvoApps 20' Proceedings of the 23rd European Conference on the Applications of Evolutionary and bio-inspired Computation - Evolutionary Machine Learning*, page to appear, 2020.

[9] Paulo Fernandes, João Correia, and Machado Penousal. Towards Latent Space Exploration for Classifier Improvement. In Jer Hayes, Cathal Gurrin, Azzurra Pini, and Mark Keane, editors, *Proceedings of the Workshop on Applied Deep Generative Networks co-located with 24th European Conference on Artificial Intelligence (ECAI 2020)*, Santiago de Compostela, Spain, 2020. CEUR-Workshop Proceedings.

[10] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification, 2018.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[13] A Ardeshir Goshtasby. Similarity and dissimilarity measures. In *Image registration*, pages 7–66. Springer, 2012.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[15] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[16] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[17] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D. Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation, 2019.

[18] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition.* Chapman & Hall/CRC, 2nd edition, 2014.

[19] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015.

[20] Nils J Nilsson. Introduction to machine learning: An early draft of a proposed textbook, 1996.

[21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[22] A. Roy, N. Memon, J. Togelius, and A. Ross. Evolutionary methods for generating synthetic masterprint templates: Dictionary attack in fingerprint recognition. In *2018 International Conference on Biometrics (ICB)*, pages 39–46, Feb 2018.

[23] Veit Sandfort, Ke Yan, Perry J. Pickhardt, and Ronald M. Summers. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks, 2019.

[24] Fariba Shaker. Human sperm head morphology dataset (hushem), 2018.

[25] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models, 2017.

[26] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network, 2018.

[27] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection, 2017.

[28] Gregory P. Way and Casey S. Greene. *Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders*, pages 80–91. 2018.

[29] Tom White. Sampling generative networks, 2016.

[30] D Winant, Joachim Schreurs, and J Suykens. Latent space exploration using generative kernel pca. In *Proc. of the 28th Belgian Dutch Conference on Machine Learning (Benelearn2019).* BNAIC/Benelearn, 2019.

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[32] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection, 2019.

Appendices

This chapter contains copies of the publications attained during the dissertation.

Evolutionary Latent Space Exploration of Generative Adversarial Networks

Paulo Fernandes, João Correia, and Penousal Machado

CISUC, Department of Informatics Engineering
University of Coimbra
`pcastillo@student.dei.uc.pt, jncor@dei.uc.pt, machado@dei.uc.pt`

Abstract. Generative Adversarial Networkss (GANs) have gained popularity over the years, presenting state-of-the-art results in the generation of samples that follow the distribution of the input training dataset. While research is being done to make GANs more reliable and able to generate better samples, the exploration of its latent space is not given as much attention. The latent space is unique for each model and is, ultimately, what determines the output from the generator. Usually, a random sample vector is taken from the latent space without regard to which output it produces through the generator. In this paper, we move towards an approach for the generation of latent vectors and traversing the latent space with pre-determined criteria, using different approaches. We focus on the generation of sets of diverse examples by searching in the latent space using Genetic Algorithms and Map Elites. A set of experiments are performed and analysed, comparing the implemented approaches with the traditional approach.

Keywords: Generative Adversarial Network, Evolutionary Computation, Latent Space Exploration.

1 Introduction

GANs have been presenting state-of-the-art results in the generation of samples that follow the distribution of the input training dataset [1]. In general, this model of adversarial learning works by having a generator and a discriminator training together and competing against each other in a min-max game. The discriminator is trained with real data as well as fake data created by the generator from latent space. The generator evolves from the feedback given by the discriminator on the generated data Figure 1. Although able to produce results with high quality, GANs are often really hard to train, requiring a lot of fine-tuning through trial and error.

While a lot of research is being made in order to make GANs more reliable and able to generate better samples, the exploration of its latent space does not have as much attention. The latent space is unique for each model and is, ultimately, what determines the output from the generator since, in simple terms, we have *g(z)=x'*, being *x'* the output of a latent vector *z* through the

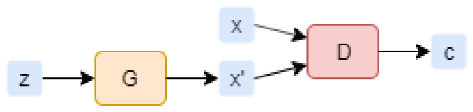


Fig. 1. Base model of a Generative Adversarial Network. G - generator; D - discriminator; z - latent vector; x - real samples; x' - generated samples

generator *g*. Typically we use the trained model to draw random samples from the latent space without any particular criteria.

In this work, we move towards the search and exploration of solutions from the latent space according to pre-determined criteria. The overall idea is to enable the search and generation of sets of latent vectors that accomplish a certain objective. We will start from exploring approaches that enable us to draw samples and traverse the latent space moved by a certain objective function. We will use Genetic Algorithms (GAs) and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) to assist on such task. Thus, the contributions of this paper are the following: (i) modelling of the latent space exploration as a search problem, enabling the use Genetic Algorithms and Map Elites (ii) a generalised approach to generate sets of images from a GAN according to with different objectives, e.g. generate diverse sets of images; (iii) a comparison analysis of the implemented latent space search algorithms with the conventional approach. Without lacking generalisation, we apply the ideas of this paper to the image domain, using Deep Convolutional Generative Adversarial Networkss (DCGANs) [2].

The remaining of the paper goes as follows: in the next Section, we cover approaches relate to this work (Section 2). In Section 3, we describe our approaches to this search problem. We describe the experimental setup in Section 4 and analyse and discuss the results in Section 5. In Section 6 we draw overall conclusions.

2 Related Work

GANs are generative models that are trained through a face-off between a generator and a discriminator, mostly used to train a generator that can produce realistic images. In order to generate an image, the generator is usually given a random noise vector, a high dimensional vector that, in training, is randomly sampled from a distribution, for example, a gaussian distribution, called the prior. The high dimensional space from which images are created is called latent space. Some work has already been made in this area, and not only with GANs, which is the type of model that we are going to work within this work. For instance, latent space exploration was performed in generative models using Kernel Principal Component Analysis (KPCA) [3], showing navigation through image features and novelty detection; but also in Variational Autoencoders (VAE), in, for example, mapping genes into a lower-dimensional space in order uncover underlying gene expression features in cases of tumour or cancer [4].

There are different ways that we can explore generative models latent space. White, T. shows three different arithmetical ways we can use to navigate in it [5]. The first, interpolation, is used to disclose the path between 2 samples in the space. Usually, a linear interpolation is used, but White suggests a new way, the spherical linear interpolation - slerp - which takes into account its prior distribution used for sampling noise. Analogy, the second approach, can be simplified as "King – Man + Woman = Queen", meaning that it is possible to perform operations between images to form new images with generally predictable results. Finally, Manifold Interpolated Neighbor Embedding (MINE), is an exploration model which makes use of nearest neighbours and interpolation to construct a manifold of the space. Also worth noting is that the authors showed that by combining generative models with labelled data, attribute vectors can be computed using simple arithmetic, like, for example, a smile vector which, by traversing uncovers several states of smiling in an image.

As for the use of Evolutionary Computation (EC) in order to evolve images, some works can be mentioned. For example, evolving master print templates [6] that, like a master key, could be able to open multiple fingerprints closed locks. In their work, Roy et al. compared four different Evolutionary Algorithms (EA), namely Hill-Climbing, Covariance Matrix Adaptation Evolution Strategy, Differential Evolution and Particle Swarm Optimization to evolve Synthetic MasterPrints according to the metric proposed by them, the Modified Marginal Success Rate. The samples were generated from two datasets, namely Authentec AES3400, with latter algorithm getting the best results, and FVC 2002 DB1-A, for which the second approach had the most success. Moreover, and with a two-stage workflow similar to what we implemented in this paper which includes first the unsupervised training of GANs and second the evolution of latent space, there are two works. One that implements Interactive Evolutionary Computation [7] for image generation and the other, in the topic of video-games that uses GANs and latent space evolution to learn and improve Mario Levels [8] using Covariance Matrix Adaptation Evolution Strategy.

There is also a new approach to generative models which was inspired by GANs, the Generative Latent Optimization [9]. This approach takes away the adversarial discriminator and replaces it with simple reconstruction losses where the focus is to evolve the latent space to match the one learnable noise vector to each one of the images in the training dataset.

3 The Approach

Our goal in this paper is to use approaches that can explore the latent space and create a set of latent vectors towards an objective. In particular, we set the objective to: find a set of images that can maximize a diversity measure. We pursuit the objective by exploring the latent space of GAN models using EC. Therefore, this experiment was separated into two main parts:

– Training of the GANs - develop a generative model that can produce images which follow the distribution of a certain training input

– Exploration of the latent space via EC - navigate the latent space of the generative model in order to find a possible solution for our problem.

The exploration of the latent space and set of latent vectors has the potential to promote the generation of samples according to pre-determined criteria to solve and adapt to other problems. It could be used during the training of the GAN, to have a few latent vectors generated that maximize the loss of the model, instead of randomly generating all of them. It could be used to generate samples of a particular type. Some metrics of evaluation of GANs must draw samples from the latent space, generate the samples and test them on another model [10]– "what if we guide that generation?" We can use this approach to spot problems on the training of models.

Also, this problem was applied to 3 distinct sets of images in order to comprehend how well our approach would work for different situations:

– handwritten digits – MNIST
– clothing – Fashion – MNIST
– faces – Facity

3.1 Model definition

We are using a type of GANs suitable for generating images, the DCGAN [2]. They make use of deep convolutional layers to better explore space correlation in images, producing more realistic images. For every GAN, the latent space, which is the input for all generators, is an array of floats of shape 1×100. The generators used in both MNIST datasets are built with the following model, generating output with shape $28 \times 28 \times 1$:

```
Dense((7, 7, 128), activation="relu")
UpSampling2D()
Conv2D(128, kernel_size=3, padding="same"))
BatchNormalization(momentum=0.8))
Activation("relu"))
UpSampling2D())
Conv2D(64, kernel_size=3, padding="same"))
BatchNormalization(momentum=0.8))
Activation("relu"))
Conv2D(1, kernel_size=3, padding="same"))
Activation("tanh"))
```

The generator used with the facity dataset, has an output shape of $128 \times 128 \times 3$ and is built with the following model:

```
Dense((8, 8, 128), activation="relu")
UpSampling2D()
Conv2D(256, kernel_size=3, padding="same")
BatchNormalization(momentum=0.8)
Activation("relu")
```

```
UpSampling2D()
Conv2D(128, kernel_size=3, padding="same")
BatchNormalization(momentum=0.8)
Activation("relu")
UpSampling2D()
Conv2D(64, kernel_size=3, padding="same")
BatchNormalization(momentum=0.8)
Activation("relu")
Conv2D(3, kernel_size=3, padding="same")
Activation("tanh")
```

The discriminators used for each GAN, all follow the same model, having an input shape defined by the output of the generators.

```
Conv2D(32, kernel_size=3, strides=2, padding="same")
LeakyReLU(alpha=0.2)
Dropout(0.25)
Conv2D(64, kernel_size=3, strides=2, padding="same")
ZeroPadding2D(padding=((0, 1), (0, 1)))
BatchNormalization(momentum=0.8)
LeakyReLU(alpha=0.2)
Dropout(0.25)
Conv2D(128, kernel_size=3, strides=2, padding="same")
BatchNormalization(momentum=0.8))
LeakyReLU(alpha=0.2))
Dropout(0.25))
Conv2D(256, kernel_size=3, strides=1, padding="same")
BatchNormalization(momentum=0.8)
LeakyReLU(alpha=0.2)
Dropout(0.25)
Flatten()
Dense(1, activation='sigmoid')
```

After building both the model of the discriminator and of the generator, they are combined into a single, combined, model which receives the same input as the generator and has the output of the discriminator.

3.2 GAN Training

The adversarial neural networks are trained by having the discriminator learn to distinguish the real samples, which come from the input datasets, from the fake samples, that are generated by the generator, and by having the generator learn to produce images that successfully trick the discriminator into classifying them as real samples [1].

The input datasets used Figure 2, were the following:

– MNIST – dataset of handwritten digit with 70000 (60000 for training and 10000 for testing) 28x28x1 (black and white) images, divided into 10 classes[11].

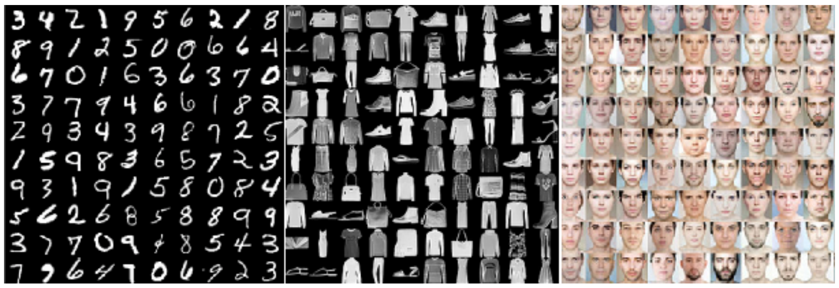


Fig. 2. Samples from the datasets used for training. From left to right: MNIST, Fashion-MNIST, Facity

- Fashion MNIST – dataset of pieces of clothing with 70000 (60000 for training and 10000 for testing) 28x28x1 (black and white) images, divided into 10 classes [12].
- Facity – dataset of faces with 4024 128x128x3 (rgb) images, not divided by training or classes.

3.3 Latent Space Exploration

In order to traverse the latent space, we require to generate sets of individuals. To pursuit our objectives, we used the following approaches:

- Random Sampling (RS)
- GA
- MAP-Elites

The three approaches were thought to evolve set of individuals, but they vary in the way they promote change along with iterations.

In the RS approaches a completely new random set of individuals is created at the beginning of each generation and evaluated by an evaluation function, as shown in 1.

Algorithm 1 Random Sampling

```
1: procedure RANDOMSAMPLING(iterations, popsize, objective, fitnessfunc)
2:     for $i \leftarrow 1$ to iterations do
3:         population = RANDOMPOPULATION(popsize)
4:         population = FITNESSFUNC(population)
5:         best = BEST(population, objective)
       return BEST(population)
```

The GA is an EC approach where we start with a randomly generated population of individuals, but a new population is created trough variation operators

Algorithm 2 Genetic Algorithm

```
1: procedure geneticAlgorithm(generations, popsize, problemargs)
2:     population = randomPopulation(popsize)
3:     for $i \leftarrow 1$ to generations do
4:         population = variationOperators(popsize, problemargs)
5:         population = problemargs.fitnessfunc(population)
6:         population = problemargs.sort(objective)
       return best(population)
```

(mutation and crossover) which are applied to the individuals of the old population evaluated by a certain fitness function (refer to algorithm 2).

The last algorithm, the MAP-Elites, works a little bit differently. It is an illumination algorithm, and it was made for exploring the search space of solutions as much as possible [13]. A map of plausible combinations of feature dimensions for the individual's phenotype, which is previously defined, is maintained throughout the training. The algorithm starts by generating a random number of individuals and placing them on the feature map. Afterwards, MAP-Elites runs by iterations. At each iteration, a single new individual is created from applying variation operators to individuals already placed in the map. Each new individual is then evaluated and placed in the map according to its features, though in each cell of the map, only the best is kept according to the fitness function.

Algorithm 3 MAP-Elites

```
1: procedure mapElites(iterations, initpopsize, map, problemargs)
2:     for $i \leftarrow 1$ to initpopsize do
3:         newind = randomIndividual(problemargs)
4:         placeInMap(newind, map)
5:     for $i \leftarrow 1$ to iterations do
6:         newind = variationOperators(map, problemargs)
7:         placeInMap(newind, map)
       return best(map)
```

Individuals, Initialization For this problem, we the decided to work with sets of 50 images, meaning that, since we are working with a latent dimension of size 100, each individual has a genotype of size 5000 that is represented in an 1×5000 array. In order to maintain consistency, The initialization of individuals follows the same Gaussian distribution as the prior used to generate fake images in the training of the generative models.

Evaluation, Metrics and Variation Operators All algorithms use the same method to evaluate the fitness of the individuals, an average similarity function.

Through the use of the specified GAN generator, the genotype of the individuals is transformed into a set of images. Afterwards the images are compared to one another using a similarity metric, averaging between all values at the end. Since we want the most diversity possible, we have modelled the problem for minimization. In terms of the feature dimensions for MAP-Elites, both the average similarity and max similarity are used map individuals.

To measure similarity, 2 distinct metrics were used, namely Root-Mean-Squared Error (RMSE) and Normalized Cross-Correlation (NCC) [14]. The RMSE metric is calculated as

$$\text{RMSE} = 1 - \sqrt{\frac{\sum((A - B) \odot (A - B))}{\text{size}}} \tag{1}$$

while NCC is calculated as

$$\text{NCC} = \frac{\sum(A - B) \odot (A - B)}{\sqrt{(\sum A \odot A) \times (\sum B \odot B)}} \tag{2}$$

Where A and B are images, size is a function that measures the size of the images and $\odot$ is the Hadamard product. We selected these metrics for their fast calculation time and to observe the impact of both since they work in different ways. With RMSE we have a strict and direct pixel by pixel comparison whereas with NCC we are looking for certain contrast in the pixel intensities. These are options out of a number of different similarity metrics [14], but covering all of them is out of the scope of .

Two variation operators were used for the experiments: crossover and mutation. In the crossover, the two individuals are chosen from the set of available individuals using tournament selection in the GA and random choice in MAP-Elites. In both algorithms, the algorithm used is the uniform crossover [15].

In the case of the mutation operator, it was used a random reset mutation, where each gene has a probability of being mutated. The mutation resets the selected genes with completely new values that are taken from the same Gaussian distribution used in the initialization of the individuals.

4 Experimental Setup

The experimental setup was thought to analyse how different algorithms and distinct metrics affect, for different situations, the end result in terms of diversity, which is the main goal. Moreover, by analysing the observable characteristics of the sets of images obtained, we also wanted to see if the diversity measured by an algorithm, is consistent to what we, as humans, perceive as a diverse.

To perform the experiments three generative models were trained, 1 for each dataset to be used Figure 3. Most of the parameters are the same Table 1, the only thing that changes is the number of epochs of training: (i) MNIST - 200; (ii) Fashion-MNIST - 800; (iii) Facity - 1000.

So that the results would be comparable, we keep the conditions of the algorithms as close as possible Table 2, for instance, performing a number of

Table 1. GAN Parameters

Parameter	Setting
optimizer	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
batch size	32
loss function	Binary Cross-Entropy
noise distribution	N(0,1)

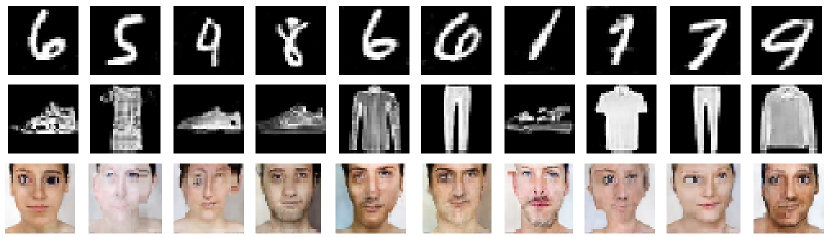


Fig. 3. Random sample of images generated from the 3 generative models

iterations on MAP-Elites that would produce the same number of evaluations as in the GA, maintaing the same number of generated images and attempts to reach the diverse set of images.

5 Experimental Results

In this section, we analyse the results from the experiments in terms of optimisation of the fitness function and the visual outputs from each algorithm. In order to analyse the results for comparison, we provide graphics with aggregated

Table 2. GA and MAP-Elites Parameters

Parameter	Setting
Population size	50 in RS/GA, initial≤50 in MAP-Elites
Number of generations	500 in GA, 25K in RS/MAP-Elites
Genotype length	50× size of latent space
Elite size	1
Tournament size	3
Crossover operator	uniform crossover
Crossover rate	0.7 in GA/MAP-Elites
Mutation operator	gene replacement
Mutation rate per gene	0.02 in GA/MAP-Elites
feature dimensions	avg similarity : bins=[0:0.01:1], max similarity bins=[0:0.01:1] in MAP-Elites

the values of evaluations. We group the values for Random Sampling and Map Elites in iterations, that corresponds to 50 evaluations. This way, it is possible to compare with the information of the GA. Basically, one iteration is equivalent to a generation on the GA, which in turn is equivalent to 50 evaluations on Random Sampling and Map Elites algorithms. We also separate the analysis of image diversity in two groups based on the similarity metric: RMSE and NCC.

In figure 4, we can observe the results of the different algorithms across the iterations. It is clear that the GA is able to optimise the fitness function. The same does not happen with MAP-Elites though, we get little improvement compared to the random sampling, and that does not change when working with different datasets, the difference between the actually gets narrower the more complex is the type of images.

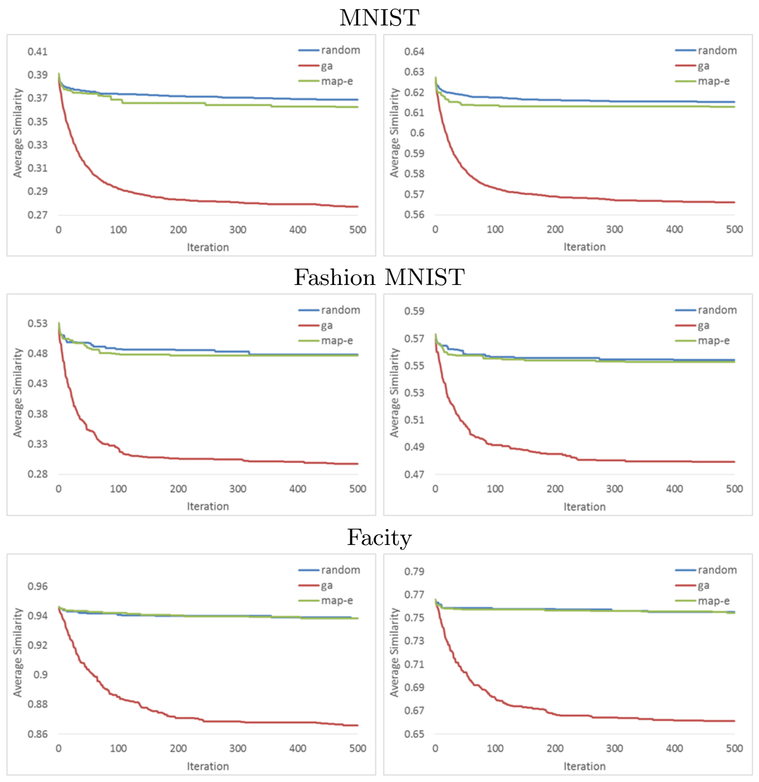


Fig. 4. Average Maximum values across iterations for the different datasets and similarity metrics: on the left NCC and on the right RMSE. The values are averages of 10 runs.

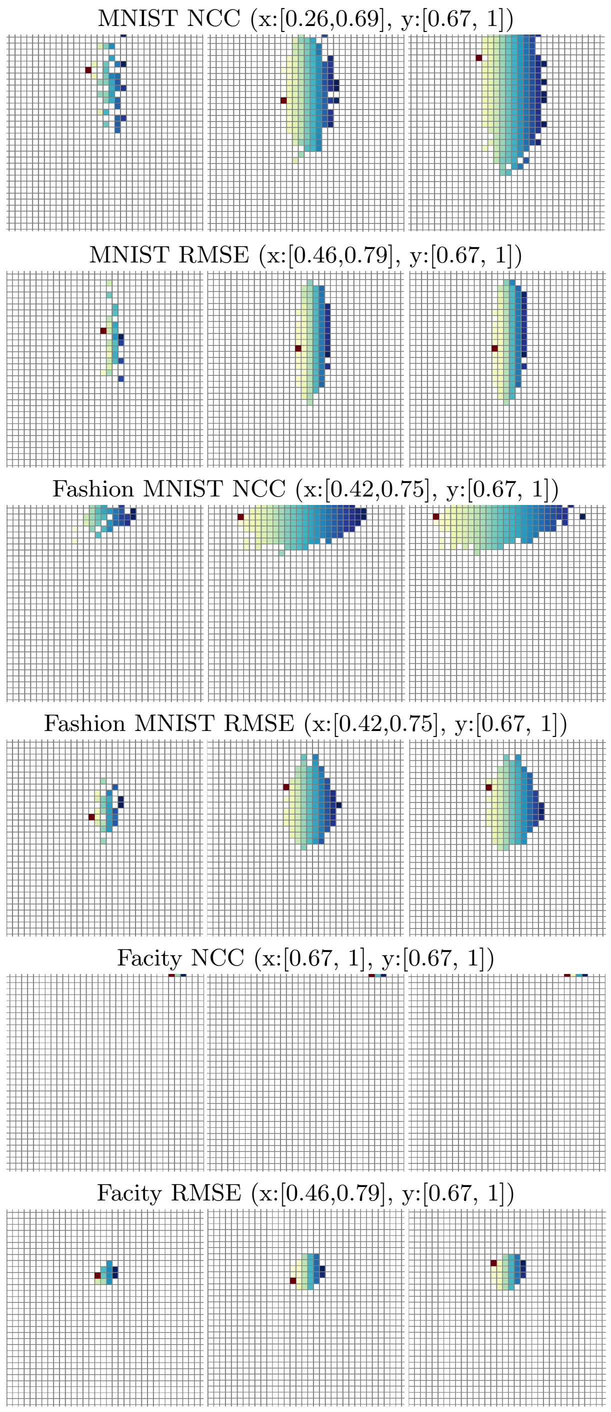


Fig. 5. Heatmap of the Map constructed with MAP-Elites for each dataset and similarity metric. Images correspond to 0th, 12000th and 25000th iterations of the algorithm. The color represent fitness, yellow for better fitness and blue for worse, while the red represents the best individual. The x axis corresponds to the average similarity and the y axis to the maximum similarity.

When analysing the Map Elites algorithm, we are also concerned with the mapping of the individuals. As such, we analysed the heatmaps of the algorithm across iterations for the datasets and metrics, as presented in figure 5, to understand how much of the space was being explored by the algorithm, which helps us to understand the distribution of samples along with the iterations.

We can observe that space exploration is limited and that it generally seems to take increasingly more time to expand and, therefore, more time to find better solutions. It is noticeable the existence of a cluster of solutions in this approach and that the expansion is easier for less fit (dark blue) zones, which corroborates the evolution graphs. Here we clearly see how much MAP-Elites struggles to find better solutions when the mapping is not favourable to the problem. With the facity dataset and the NCC metric, the random initialisation values were clustered into a very small area, which made it really difficult for the algorithm to expand. Even the variation operations caused changes so small that they still fall in the same area.

The results suggest that this algorithm is not suitable for this type of problem. However, it could be related to the selection of feature descriptor and algorithm's parametrisation. The mapping functions tend to be simplistic and become a bottleneck that does not allow the ideal exploration of the search space and expansion of the mapped area, something that we will further investigate.

In terms of visuals the outputs from Figure 6 and Figure 7 showcase the best set of images that maximize the pre-determined criteria. For this last analysis, we only focused on the NCC metric and RMSE metric, respectively. It is noticeable the differences between the different approaches and metrics. In general, every approach, for each metric, ended up having a different set of images at the end.

Between the NCC and RMSE results, one aspect is noticeable, and it showcases the particularity of each similarity metric. In the RMSE, we observe that it tended to pick a set of variables that generated images which minimize the overlap of elements and with a dissimilar background (facity). The NCC, on the other hand, tended to promote contrast between the different images. This is a clear example of the success of the approach of searching and achieving the pre-determined objective. Among the different approaches, we can observe that the GA was able to generate the most visually diverse set of images.

6 Conclusions

This work presents multiple methods that enable the user to explore the latent space with a pre-determined objective. We have implemented Genetic Algorithms and Map Elites and compared with the traditional approach of random sampling from the latent space. Some of the results obtained with were unexpected but suggested that we are able to generate diverse sets of latent variables that translate into samples that correspond to certain criteria. The conducted experiments, in the image domain, with different datasets, point out that it is possible to apply our approach to GANs of different types of datasets, ranging from grayscale to colour images. The overall results worked as a proof of con-

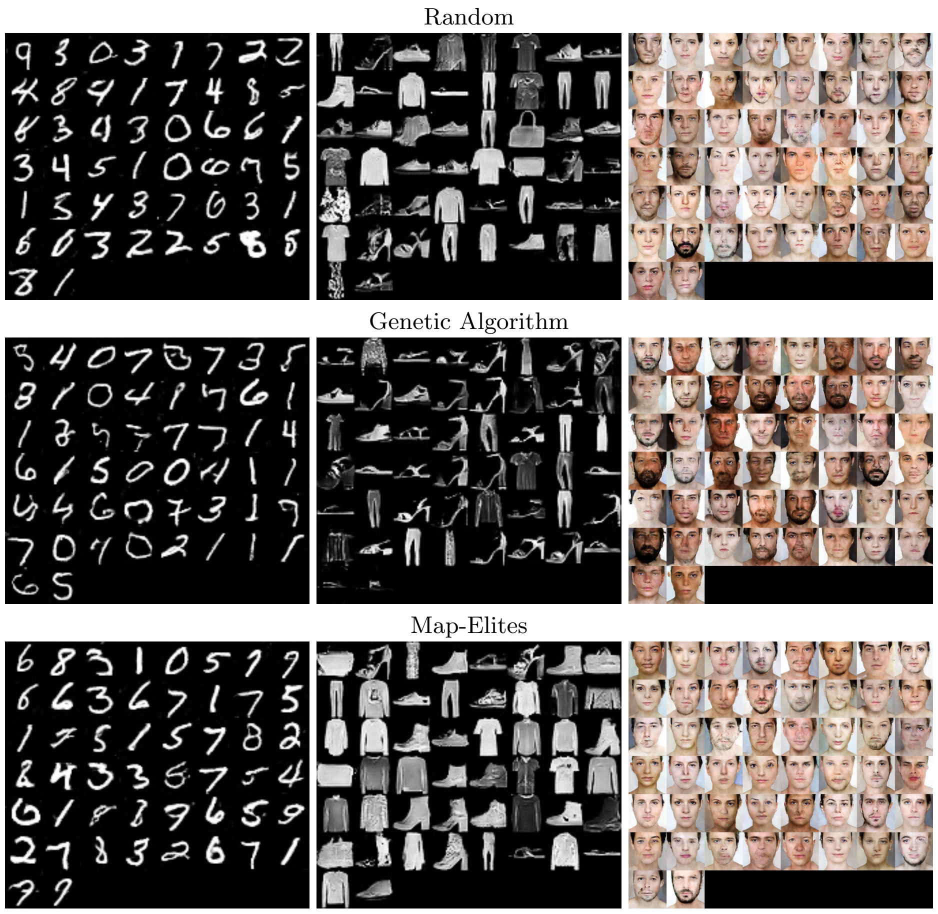


Fig. 6. The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the NCC metric.

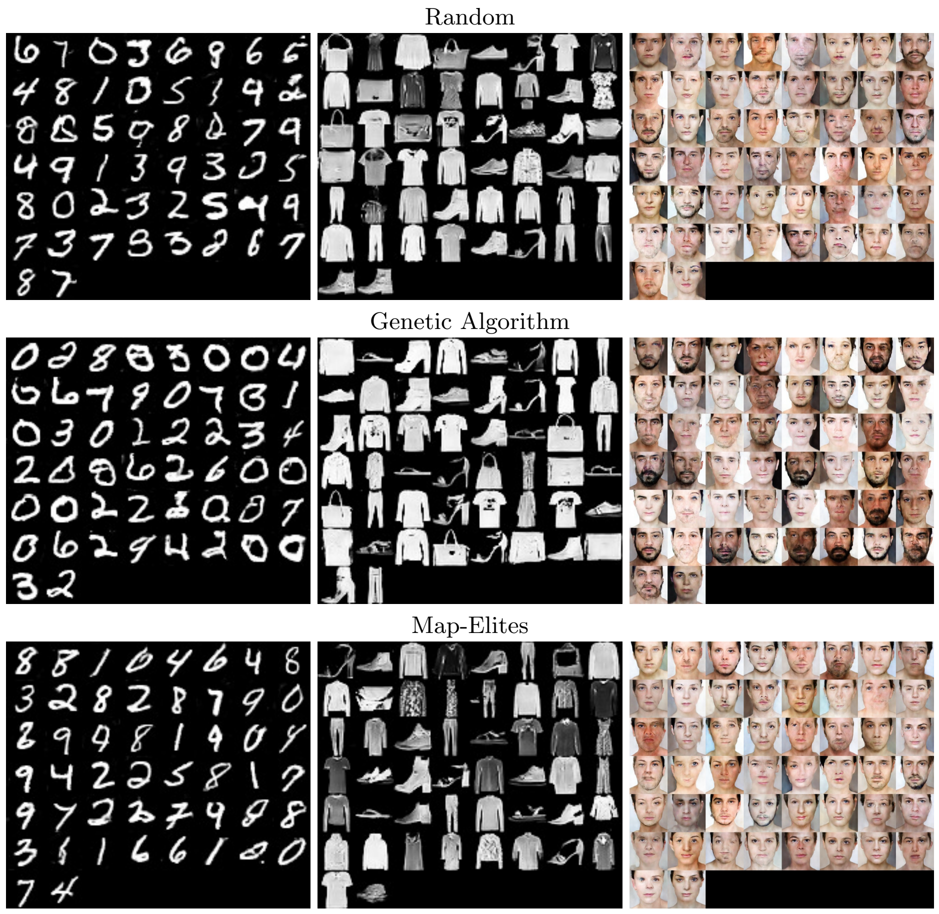


Fig. 7. The best individual of one of the evolutionary seeds for each dataset per type of approach in the last iteration using the RMSE metric.

cept that is possible to guide the generation of latent variables towards certain criteria.

Future work could include other strategies and objectives for navigation, such as the usage of different metrics; fine-tuning of parameters, application of different criteria for illuminating the latent space in map-elites, which may include working with more than 2, usage of multiple seeds per set as well as adding a pre-processing layer that could help focus on the regions of interest in the images (for example the removal of the background). Moreover, the generation of groups of images uncovers problems of generalization. Therefore research could be done in order to explore reinforcement of training datasets.

7 Acknowledgments

This work is partially supported by national funds through the Foundation for Science and Technology (FCT), Portugal, within the scope of the project UID/CEC/00326/2019 and it is based upon work from COST Action CA15140: Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO).

References

1. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS. pp. 2672–2680 (2014)
2. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015)
3. Winant, D., Schreurs, J., Suykens, J.: Latent space exploration using generative kernel pca. In: Proc. of the 28th Belgian Dutch Conference on Machine Learning (Benelearn2019). BNAIC/Benelearn (2019)
4. Way, G.P., Greene, C.S.: Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders, pp. 80–91 (2018)
5. White, T.: Sampling generative networks. CoRR abs/1609.04468 (2016), `http://arxiv.org/abs/1609.04468`
6. Roy, A., Memon, N., Togelius, J., Ross, A.: Evolutionary methods for generating synthetic masterprint templates: Dictionary attack in fingerprint recognition. In: 2018 International Conference on Biometrics (ICB). pp. 39–46 (Feb 2018)
7. Bontrager, P., Lin, W., Togelius, J., Risi, S.: Deep interactive evolution. In: Liapis, A., Romero Cardalda, J.J., Ekárt, A. (eds.) Computational Intelligence in Music, Sound, Art and Design. pp. 267–282. Springer International Publishing, Cham (2018)
8. Volz, V., Schrum, J., Liu, J., Lucas, S.M., Smith, A., Risi, S.: Evolving mario levels in the latent space of a deep convolutional generative adversarial network (2018)
9. Bojanowski, P., Joulin, A., Lopez-Paz, D., Szlam, A.: Optimizing the latent space of generative networks (2017)
10. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. pp. 6626–6637 (2017)

11. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010)
12. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
13. Mouret, J.B., Clune, J.: Illuminating search spaces by mapping elites (2015)
14. Goshtasby, A.A.: Similarity and dissimilarity measures. In: Image registration, pp. 7–66. Springer (2012)
15. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series, Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

Towards Latent Space Exploration for Classifier Improvement

Paulo Fernandes and **João Correia** and **Penousal Machado**[1]

Abstract.
We propose a framework that combines Generative Adversarial Networks and Evolutionary computation to perform Data Augmentation on small datasets in order to improve the performance of image classifiers trained via supervised learning. In this work, we attest the viability and potential of this framework for real-world problems. The framework is composed of a generator module that uses Generative Adversarial Networks to generate samples from a dataset. It employs an Evolutionary Computation approach to evolve sets of images from the latent space. The fitness function is based on the dissimilarity of the subsets generated by the Generative Adversarial Network. A Supervisor module handles the generated samples and chooses which set should be added to the training dataset. To test the framework, we explore the Human Sperm Head Morphology dataset, a bio-medicine multi-class problem with a small number of samples that provide a challenge to the different supervised classification approaches. We deploy the framework to create an augmented dataset to train a classifier, and after the training, we compute the performance on the test set. We compare with classifiers trained using the base datasets without having the generated samples. Overall, with the preliminary tests, we can improve the performance of the classifiers by up to 4% and on average by 1%, showing the viability and potential of our approach.

1 INTRODUCTION

With the evolution of technology and computer capabilities, Machine Learning has seen significant improvements in recent years. Like so, it became much easier to build and apply larger neural networks, such as Deep Neural Networks, to solve real-world problems. It also became possible to build Deep Generative Models which produce synthetic data by learning from already existing data.

While the availability of data has also been accompanying the evolution of technology, there are still many problems that lack enough data to allow Machine Learning algorithms to be viable solutions for them. The performance of Machine Learning algorithms depends not only on the capability of the model used but also on the quality of the dataset used in the training of the model. This means that training a model with a bad dataset will, most likely, lead to poor performance results. As such, improving the quality of the datasets through Data Augmentation may be a way to improve the performance of the algorithm. With this in mind, is it possible to use generative models to enhance the quality of existing datasets, consequently improving the quality of machine learning algorithms?

One of the ways to train generative models is by using Generative Adversarial Networks. These frameworks are more often used to produce very realistic images that follow the distribution of the training dataset [5]. In general, these work by putting a generator and a discriminator against each other in a min-max game. The discriminator is trained to distinguish images from the original dataset from images created by the generator, while the generator learns from the feedback given by the discriminator on the generated data. In this work, we plan to use Generative Adversarial Networks to generate sets of synthetic images in order to understand if the addition of these instances into the training set of a classification model is able to improve its performance.

As a further matter, it is also essential to address the generation of new samples. Even though a capable model is important in the generation of better images, there is also another variable that impacts the instances generated, which is the latent space. It is unique to each generative model and hides underlying patterns in itself. Usually, to generate an image, a vector from the generative model's latent space is chosen at random for input, which means that there is no knowledge about the output. Since the images generated depend on the input given to the generator, the exploration of the latent space may reveal ways to control the output through the selection of input vectors by certain criteria. This way, we can also assure the quality of the images that will be added to the dataset and their relevance to help with the problem at hand. For instance, we might want to ensure that we are not adding redundant samples to the training set. Like so, performing random Data Augmentation should have a higher chance of undermining the performance of the algorithm, which means that ideally, we should prefer an approach of a supervised generation of instances. Bearing this in mind, we chose to perform this supervision by exploring the latent space using Evolutionary Computation. Using a Genetic Algorithm, we evolve sets of latent vectors which optimize a specific criterion such as the diversity of images in the set. This framework for latent space exploration was explored in [4]. Here we explore the usage of such a framework to generate new samples for the training dataset that may improve the performance of classifiers. As proof of concept for real-world problems, we instantiate the approach in the Human Sperm Head Morphology dataset (HuSHeM) [8], a multiclass problem categorized as small data, that provides a challenge by the lack of samples that exists for the problem.

The remaining of the paper goes as follows: in the next Section, we explain our approach to the problem and the framework used to solve it (Section 3). In Section 4, we describe the experimental setup and analyse and discuss the results. In Section 5 we draw overall conclusions.

[1] CISUC, Department of Informatics Engineering, University of Coimbra,Portugal, email: pcastillo,jncor,machado@dei.uc.pt

2 RELATED WORK

Generative Adversarial Networks are generative models that are trained through a face-off between a generator and a discriminator, mostly used to train a generator that can produce realistic images. The generator is given a noise vector to produce new images, usually, a high dimensional vector that is randomly sampled from a distribution, for example, a gaussian distribution, called the prior. The high dimensional space is called latent space. Some work has already been made to explore the latent space of generative models, and not only with Generative Adversarial Networks, which is the framework that we will use in this work. For instance, latent space exploration was performed in Kernel Principal Component Analysis [11] models, showing navigation through image features and novelty detection; but also in Variational Auto-Encoders, in, for example, mapping genes into a lower-dimensional space to uncover underlying gene expression features in cases of tumour or cancer [10].

Evolutionary Computation has also been used in some works in order to evolve images. For instance, evolving master print templates [7] that, like a master key, are able to match multiple fingerprints. In their work, Roy et al. compared four different Evolutionary Algorithms, namely Hill-Climbing, Covariance Matrix Adaptation Evolution Strategy, Differential Evolution and Particle Swarm Optimization to evolve Synthetic MasterPrints according to the metric proposed by them, the Modified Marginal Success Rate. The samples were generated from two datasets, namely Authentec AES3400 and FVC 2002 DB1-A. Beyond these, and with a two-stage workflow similar to what we implemented in this paper which includes first the unsupervised training of Generative Adversarial Networks and second the evolution of latent space, there are two works that can be mentioned. One that implements Interactive Evolutionary Computation [2] for image generation and another that uses Generative Adversarial Networks and latent space evolution to learn and improve Mario Levels [9] using Covariance Matrix Adaptation Evolution Strategy. Finally, a recent approach to generative models which was inspired by Generative Adversarial Networks, the Generative Latent Optimization [1]. This method replaces the adversarial discriminator with simple reconstruction losses where the focus is to evolve the latent space to match the one learnable noise vector to each one of the images in the training dataset.

3 FRAMEWORK

In this paper, we propose a framework that combines Generative Adversarial Networks and Evolutionary computation to perform Data Augmentation on small datasets in order to improve the performance of image classifiers trained via supervised learning.

For the framework, there are 3 fundamental pieces: (i) a classifier responsible for the classification task, discriminating images into classes; (ii) a generator, responsible for generating new images, from an array of the latent space; (iii) a supervisor, responsible for managing the generation images through the exploration of latent space (as in [4]).

3.1 Classifier

The performance of classifier will measure the performance of this framework. Therefore, we will be looking into comparing the performance of the classifier after the baseline training - using only the original dataset - against the performance of the classifier after the supervised augmented training - with selective addition of synthetic images to the training set. This way, we will be able to assess the quality of our approach and guide the progress of research.

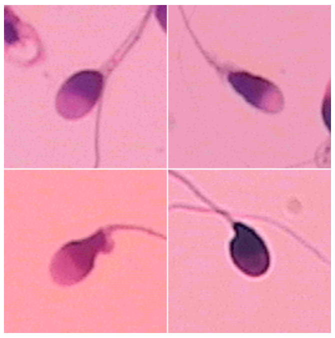


Figure 1. Class example images from original dataset. From left to right, top to bottom: Normal, Tapered, Pyriform and Amorphous.

3.2 Generator

The generator will be obtained through a Deep Convolutional Generative Adversarial Network. These make use of Deep Convolutional layers that better explore space correlation in images [6], which helps to generate better quality images. The training is unsupervised, which means that no information is given to the model to guide the generation of images, the training progresses purely through the differentiation between real(original) and fake(generated) images [5]. Therefore, a generator will be trained for each class of the problem to specifically control the generated images for each one. Each generator will learn to produce images following the distribution for a single class. During training, the generated images are created from random vectors, following a gaussian distribution.

3.3 Supervisor

Lastly, the Supervisor, which is crucial to the optimization of the training dataset. By adding random images with no criteria, there is no way to ensure if these are relevant to the solution of the problem. The introduction of flawed and redundant images might end up undermining the performance of the classification algorithm [3]. One way to control the output of the generators is by controlling its latent space. Selecting generated images through certain criteria will allow for the optimization of the classifier. More specifically, we will be looking into finding sets of images that are as diverse as possible so as to minimize redundancy. The exploration of the latent spaces will be performed through Evolutionary Computation, more specifically using a Genetic Algorithm. Each individual in the algorithm will represent a set of images where its genetic code corresponds to the latent vectors of said set of images. The initial population is created through random sampling from the same Gaussian distribution of the generator training. At each iteration, new populations are created by using Tournament Selection, Uniform Crossover and Random Reset Mutation (which also applies the previous Gaussian distribution to obtain the value of the new genes). The evaluation of individuals and fitness function correspond to the averaging of the similarities between each image in the set and the centroid image of the set that includes the images from that individual together with the images from the original dataset. The similarities between images and centroid are calculated using Normalized Cross-Correlation. Since we are searching for diverse datasets, the objective is to minimize the target function. In the end, we should find a set of images that comes closest to the intended objective and better tackles the issues at hand.

4 EXPERIMENTATION

In order to evaluate our approach, we performed several tests. The conditions on which these tests were carried out are presented in this section.

4.1 Dataset

I order to test our hypothesis we will be using the Human Sperm Head Morphology dataset (HuSHeM) [8]. In the bio-medicine context, Sperm morphology analysis is a key factor in the diagnosis process of male infertility. The dataset is divided into 4 classes of sperm heads images [Figure 1]: Normal (54 instances), Tapered (53 instances), Pyriform (57 instances) and Amorphous (52 instances) for a total of 216 images. A small dataset like this one is an opportunity to explore Data Augmentation approaches. The dataset has no sub-division, as such it was decided that we would use 40 instances of each class for training and cross-validation, leaving the remaining images for testing.

Each image has a original dimensions 131x131x3, but in the experiments we will be working with in dimensions 132x132x1.

4.2 Classifier

The classifier module allows the assertion of the experiment results. The model used was an off-the-shelf model that only required training since the optimization of the model was one of our objectives for this work. The parameters used for the training of the classifier are present in Table 1. The number of epochs chosen ensures that the classifier reaches a point of plateau for the original dataset, where there is no gain in performance. This helps to verify the quality of our solution. On another note, the training also included cross-validation for every test, which means both tests with the original dataset and augmented datasets.

Table 1. Classifier parameters

Parameter	Setting
optimizer	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
epochs	250
batch size	32
loss function	Binary Cross-Entropy
cross-validation	Stratified
folds	5

4.3 Generator

The generators allow for the creation of new samples to perform data augmentation. A generator is obtained through unsupervised training of a Deep Convolutional Generative Adversarial Neural Network. The model used for the discriminator, as the classifier, is an off-the-shelf model, to which we only added 2 extra layers of convolution. As for the discriminator, the model used was the same as the classifier. The training is performed in each individual class, which means that in this case, in particular, we are going to need 4 different generators to produce samples for each class [Figure 2]. Each generator was trained using the 40 class corresponding samples in the training set. The training parameters were set, as shown in Table 2.

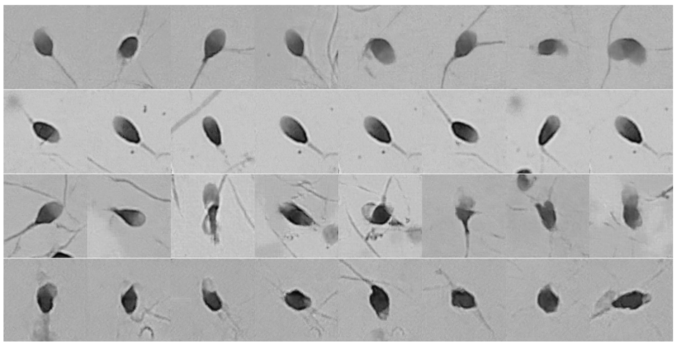


Figure 2. Class example of random images produced by the generator. On each row from, top to bottom: Normal, Tapered, Pyriform, Amorphous

Table 2. Deep Convolutional Generative Neural Network parameters

Parameter	Setting
latent dimension	100
optimizer	Adam
beta1	0.5
beta2	0.999
learn rate	0.0002
epochs	10000
batch size	32
loss function	Binary Cross-Entropy
noise distribution	N(0,1)

4.4 Supervisor

The core step of this work is the supervision of the generation of samples. This is what is going to allow the optimization of the process of performing Data Augmentation and ensure the best results possible. For this, we decided to use Evolutionary Computation, namely a Genetic Algorithm, to explore the latent space of the generators with the intent of finding sets of algorithms that optimize a certain criterion. In this case, specifically, we are looking into maximizing the diversity of the dataset. The supervision is performed for a single generator, or single class, which means that in this problem, we are going to use 4 supervisors to evolve 4 different sets of images that will be added to the original set. The parameters used in the genetic algorithm were as defined in Table 3.

Table 3. Genetic Algorithm Parameters

Parameter	Setting
Population size	20
Number of generations	500
Genotype length	number of images $\times$ latent dimension
Elite size	1
Selection method	tournament
Tournament size	3
Crossover operator	uniform crossover
Crossover rate	0.7
Mutation operator	random reset mutation
Mutation distribution	N(0,1)
Mutation rate per gene	0.02

Each individual represents a set of images that are coded into its genetic code in the form of latent vectors The fitness of each individual is calculated by averaging the similarities between each image in the set and the centroid image of the set that includes the images from that individual together with the images from the original dataset. The similarity is calculated using the Normalized Cross-Correlation metric. The calculations are as follows.

$$T = I^\frown O \tag{1}$$

$$C = \frac{\sum_{t \in T} t}{length(T)} \tag{2}$$

$$F = \frac{\sum_{i \in I} NCC(i, C)}{length(I)} \tag{3}$$

T is the set resulting of concatenation of the images from the individual (I) with the images from the original dataset(O). C is the centroid of the set T. F is the fitness of the individual calculated through averaging the similarities measured using the Normalized Cross-Correlation(NCC). The Calculation of the similarity metric is as follows:

$$NCC(A, B) = \frac{\sum((A - B) \odot (A - B))}{\sqrt{\sum(A \odot A) \times \sum(B \odot B)}} \tag{4}$$

The $\odot$ corresponds to the Hadamard product between two images.

On a last note, since the fitness function measures similarity instead of diversity, the objective of the algorithm is set to minimization. In the end, we should end up with a set of images that are more diverse, than if we just picked a randomly.

4.5 Experimental Results

In order to test our framework, we performed a comparison between the performances of the classifiers before and after performing Data Augmentation. The evaluation of each classifier in the cross-validation was performed in the test dataset, where several metrics were measured, namely Accuracy, Precision, Recall, F1 score, Area Under Receiving Operator Characteristic Curve (AUROC) and Average Precision. Each test was performed 5 times with different seeds. In the following results we will be presenting the mean across these 5 repetitions. Note that the initialization of the weights is the same between Model-X and Seed-X (e.g. Model-0 and Seed-0), Seed-0 differs on the seed used to generate the augmented dataset and, of course, the existance of augmented instances.

The first tests were performed with the original dataset. By looking at Figure 3 we can observe that at 250 epoch the training has already reached a plateau in terms of accuracy, which means that it will most probably not get any benefits from further training since it will tend to overfit. We were also able to find the performance that our solution should be able to overcome [Table 4.5].

The next step was building the sets of images of images to be added to the original dataset and train with the augmented dataset. For this experiment it was decided to test a dataset composed of 50% original images and 50% synthetic images. As such, for each class we generated and evolved sets of 40 images. The selection of these sets was repeated for every repetition of the test with the classifier which was performed 5 times with different seed, similarly to what was done with the baseline test.

First, by analysing the line of evolution in Figure 4 we can see that the values of fitness of the best individuals do not have a great variation between the first and the last generations. All values, for every class, sit on an interval of 0.01 between, 0.99 and 1. This means that the similarity between the images in this dataset, for this metric, is really high and that it promote a good evolution. However, if we take look at Figure 5 which puts side-by-side the best individual of the first generation (top) against the best individual of the last generation (bottom) in the evolution of a set of the class "Amorphous", we can

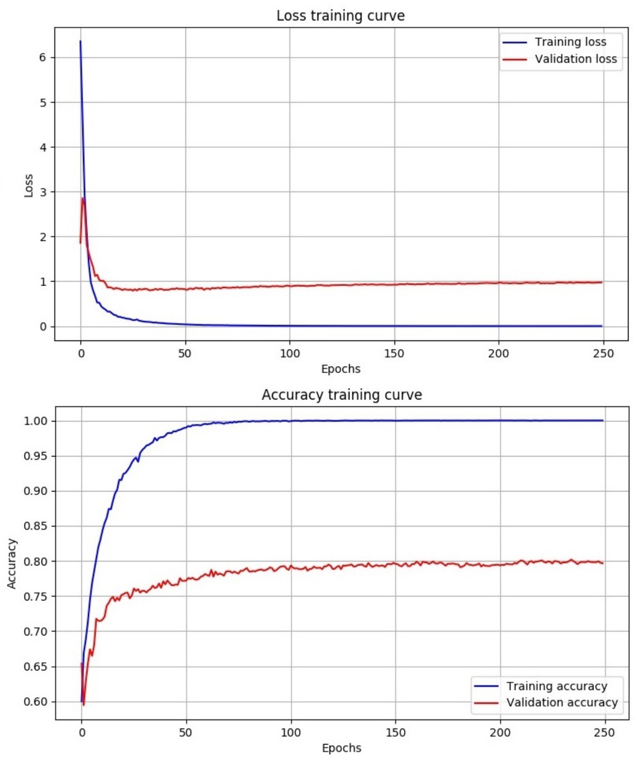


Figure 3. Average of the learning curve of the trainings with the original dataset across the all repetitions

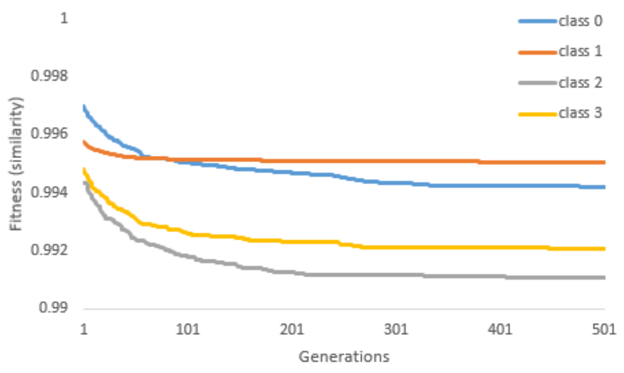


Figure 4. Average of the fitness of the best individual during the evolution process across all repetitions

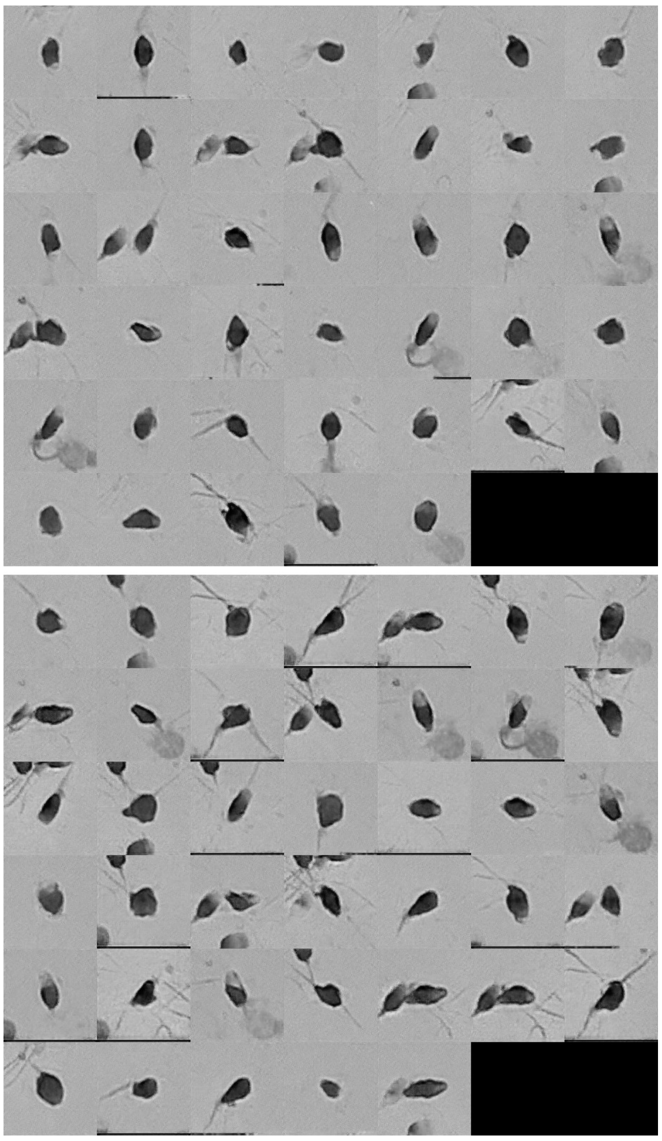


Figure 5. Best individuals at the end of generations 0 and 500, top and bottom respectively, from the process of evolution of individuals from the class "Amorphous"

argue that the latter has in fact, from a subjective perspective, more visual diversity than the first.

The last step is the training of the classifiers with the augmented datasets. By analysing the training curve in Figure 6 we can see that at epoch 250 the training also reaches a plateau accuracy-wise, meaning that further training would not help better the performance.

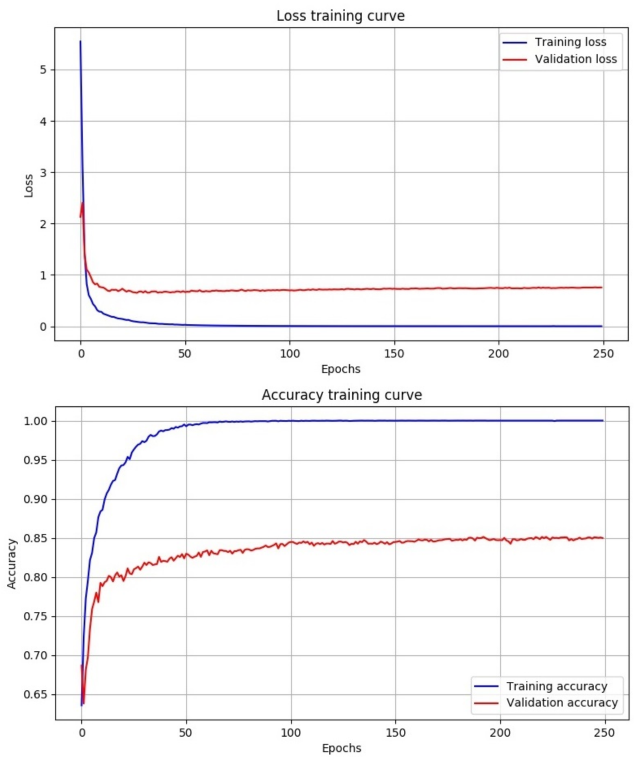


Figure 6. Average of the learning curve of the training with the augmented dataset across the all repetitions

Looking at the test results [Table 4.5], we can see that the performance of the classifiers trained with the augmented dataset was, on average, better for every metric. Although it would be necessary to perform more tests to verify the benefit of our solution, this shows that our approach might indeed be a way to improve datasets and consequently the performance of Classifiers. Each *Seed-X* represents a classifier that was trained by a subset from the Evolutionary Computation process using a different random generator seed. We can observe that in 4/5 seeds we are able to improve beyond the average performance of the original baseline classifier. We have one Seed that improves up to 4% over the baseline average. One of the seeds hindered the performance of the classifier but on average we have improvements over all the metrics when compared with the trained models with different initialization weights.

5 CONCLUSION

We explored an approach that uses Generative Adversarial Networks for Data Augmentation to improve the performance of a supervised classifier applied in a real-world problem. The underlying idea is to explore the latent space of the generative model using Evolutionary Computation to generate sets of instances to be used in the training dataset of a supervised classifier. Since arbitrarily adding instances to the dataset could hinder the performance of the classifier that is

Table 4. The classifier test results for each metric. Each model trained is denoted as *Model-X* and each model trained with the augmented dataset from the Evolutionary process is denoted as *Seed-X*. Augmented is the average of the 5 seeds and Original is the average of 5 models trained.

	Metrics					
Data	Acc	Prec	Rec	F1	AUROC	Avg-Prec
Original	0.578	0.623	0.580	0.582	0.719	0.487
Model-0	0.593	0.610	0.598	0.592	0.731	0.508
Model-1	0.575	0.648	0.580	0.588	0.719	0.497
Model-2	0.571	0.599	0.569	0.563	0.712	0.469
Model-3	0.571	0.621	0.575	0.581	0.715	0.472
Model-4	0.579	0.641	0.580	0.587	0.718	0.486
Augmented	**0.586**	**0.633**	**0.590**	**0.594**	**0.725**	**0.496**
Seed-0	0.589	0.645	0.587	0.594	0.724	0.498
Seed-1	0.589	0.632	0.596	0.596	0.729	0.495
Seed-2	0.539	0.586	0.534	0.537	0.689	0.446
Seed-3	0.596	0.642	0.604	0.609	0.734	0.508
Seed-4	0.618	0.663	0.629	0.633	0.750	0.536

being trained, we rely on a Supervisor module that selects the best set based on different criteria.

We instantiate this framework in a real-world application problem the Human Sperm Head Morphology dataset has a proof of concept. Due to the small number of instances we can categorize it as small data dataset, which presents an opportunity to explore Data Augmentation approaches. We created a baseline classifier with the provided data for comparison with the classifiers created by our framework. We used a Genetic Algorithm to evolve sets of latent space vectors that generated sets of images. We used the normalized cross-correlation similarity metric to calculate the dissimilarity among the sets and used the average value to assign fitness to each one. Overall we were able to guide evolution and generated dissimilarity subsets. The best subset from the last population of the evolutionary algorithm was used to augment the training dataset of the classifier. The classifier was then trained with the synthetic and with a base subset of instances. We used cross-validation to compute performance metrics. Overall the results show that we can increase the performance of the classifier. For example, we were able to raise accuracy by 0.8% and the f1-score by 1.2%. Although more tests are needed to verify this conclusion and even to improve the quality of the solution, it is a first step and a proof of concept of the potential of such an approach.

Future work may include testing with different proportions between original images and generated images in augmented datasets, and even testing on training sets composed of generated images only. Also, we may even test different datasets, use different similarity metrics or even improve the supervision algorithm with the inclusion of other techniques. Finally, we may also look into comparing this approach to other data augmentation approaches.

REFERENCES

[1] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks, 2017.

[2] Philip Bontrager, Wending Lin, Julian Togelius, and Sebastian Risi, 'Deep interactive evolution', in *Computational Intelligence in Music, Sound, Art and Design*, eds., Antonios Liapis, Juan Jesús Romero Cardalda, and Anikó Ekárt, pp. 267–282, Cham, (2018). Springer International Publishing.

[3] João Correia, *Evolutionary Computation for Classifier Assessment and Improvement*, Ph.D. dissertation, University of Coimbra, 2018.

[4] Paulo Fernades, João Correia, and Penousal Machado, 'Evolutionary latent space exploration of generative adversarial networks', in *EvoApps 20' Proceedings of the 23rd European Conference on the Applications of Evolutionary and bio-inspired Computation - Evolutionary Machine Learning*, p. to appear, (2020).

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio, 'Generative adversarial nets', in *NIPS*, pp. 2672–2680, (2014).

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[7] A. Roy, N. Memon, J. Togelius, and A. Ross, 'Evolutionary methods for generating synthetic masterprint templates: Dictionary attack in fingerprint recognition', in *2018 International Conference on Biometrics (ICB)*, pp. 39–46, (Feb 2018).

[8] Fariba Shaker. Human sperm head morphology dataset (hushem), 2018.

[9] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network, 2018.

[10] Gregory P. Way and Casey S. Greene, *Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders*, 80–91, 2018.

[11] D Winant, Joachim Schreurs, and J Suykens, 'Latent space exploration using generative kernel pca', in *Proc. of the 28th Belgian Dutch Conference on Machine Learning (Benelearn2019)*. BNAIC/Benelearn, (2019).