Automatic Feature Extraction with Vectorial Genetic Programming for Alzheimer's Disease Prediction through Handwriting Analysis

Irene Azzali[1, 5], Nicole D. Cilia[2,3], Claudio De Stefano[4], Francesco Fontanella[4*], Mario Giacobini[5], Leonardo Vanneschi[6]

[1] IRCCS Istituto Romagnolo per lo Studio dei Tumori (IRST) "Dino Amadori", Meldola, Italy

[2] Department of Computer Engineering, University of Enna "Kore", Italy

[3] Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands

[4] Department of Electrical and Information Engineering Mathematics, University of Cassino and Southern Lazio, Italy

[5] Data Analysis and Modeling Unit, Department of Veterinary Sciences, University of Torino, Italy

[6] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Portugal

[*] Corresponding author

This is the accepted author manuscript of the following article published by Elsevier:

Azzali, I., Cilia, N. D., De Stefano, C., Fontanella, F., Giacobini, M., & Vanneschi, L. (2024). Automatic feature extraction with Vectorial Genetic Programming for Alzheimer's Disease prediction through handwriting analysis. Swarm and Evolutionary Computation, 87, 1-11. Article 101571. https://doi.org/10.1016/j.swevo.2024.101571

Automatic Feature Extraction with Vectorial Genetic Programming for Alzheimer's Disease Prediction through Handwriting Analysis

Irene Azzali[a,e], Nicole D. Cilia[b,c], Claudio De Stefano[d], Francesco Fontanella[d,1], Mario Giacobini[e], Leonardo Vanneschi[f]

[a]*IRCCS Istituto Romagnolo per lo Studio dei Tumori (IRST) "Dino Amadori", Meldola, Italy, Italy*
[b]*Department of Computer Engineering, University of Enna "Kore", Italy*
[c]*Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands*
[d]*Department of Electrical and Information Engineering Mathematics, University of Cassino and Southern Lazio, Italy*
[e]*Data Analysis and Modeling Unit, Department of Veterinary Sciences, University of Torino, Italy*
[f]*NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Portugal*

Abstract

Alzheimer's Disease (AD) is an incurable neurodegenerative disease that strongly impacts the lives of the people affected. Even if, to date, there is no cure for this disease, its early diagnosis helps to manage the course of the disease better with the treatments currently available. Even more importantly, an early diagnosis will also be necessary for the new treatments available in the future. Recently, machine learning (ML) based tools have demonstrated their effectiveness in recognizing people's handwriting in the early stages of AD. In most cases, they use features defined by using the domain knowledge provided by clinicians. In this paper, we present a novel approach based on vectorial genetic programming (VE_GP) to recognize the handwriting of AD patients. VE_GP is an enhanced version of GP that can manage time series directly. We applied VE_GP to data collected using an experimental protocol, which was defined to collect handwriting data to support the development of ML tools for the early diagnosis of AD based on handwriting analysis. The experimental results confirmed the effectiveness of the proposed approach in terms of classification performance, size, and simplicity.

*Corresponding author
Email addresses: irene.azzali@irst.emr.it (Irene Azzali), nicoledalia.cilia@unikore.it (Nicole D. Cilia), destefano@unicas.it (Claudio De Stefano), fontanella@unicas.it (Francesco Fontanella), mario.giacobini@unito.it (Mario Giacobini), lvanneschi@novaims.unl.pt (Leonardo Vanneschi)

1. Introduction

Alzheimer's disease (AD) is a neurodegenerative disease due to an abnormal growth of amyloid plaques in nerve cells, which causes progressive atrophy of the brain. Currently accounting for 60-80% of dementia cases worldwide, if not adequately treated, it causes a continuous decline of the cognitive functions needed to live a normal life without the intervention of any caregiver. Therefore, an early diagnosis is fundamental for current and future patients (once disease-modifying treatments are available). In both cases, it will significantly improve the effectiveness of the available treatments: in the first case, it can slow the progress of the disease, whereas in the second case, it has been found that removing amyloid plaques is effective only at an early stage of AD.

It is widely agreed that handwriting is one of the daily skills ed by AD from the early stages when other signs are not yet evident. Therefore, in the last years, there has been a growing interest in using Artificial Intelligence (AI) to investigate the peculiarities and irregularities of the handwriting [1, 2, 3] of people affected by AD. This interest is mainly focused on the analysis of online handwriting, where handwriting traits are collected as time series of coordinates on the surface of the acquisition device, sampled at a given frequency. The main advantage of online acquisition is the possibility of acquiring the kinematics (dynamics) of the writing process, which are lost in offline data.

Once data are collected, AI-based systems for handwriting analysis consist of three steps: (i) preprocessing; (ii) feature extraction; and (iii) pattern recognition. Raw data are typically preprocessed by means of standard signal processing algorithms, e.g., filtering, noise reduction, and smoothing. However, their use must be carefully designed since they require tuning several parameters, e.g., cut frequencies for filtering, and the wrong settings of those parameters may affect the overall performance of the system [4]. Once data are preprocessed, informative dynamics features, such as velocity, acceleration, and jerk, are extracted together with many others and used to distinguish AD patients from healthy people. In this context, the studies presented so far have shown that the feature extraction process is a very challenging task since neurodegenerative

processes can affect different areas of the brain, and alterations in writing performance can concern cognitive or memory aspects and aspects related to fine motor control. This is the reason why a large number of features is typically considered to cope with the large variability of the effects associated with the presence of neurodegenerative disorders. The result is that some of these characteristics may be redundant, irrelevant, or even misleading. Feature selection techniques are typically used to alleviate this problem [5]. Finally, the extracted and selected features are used to train a classifier for classification. Typical classification algorithms used are k-nearest neighbors, support vector machine, and random forest, among the others [6]. The above-outlined scenario shows that AI systems for handwriting analysis have two main drawbacks: (i) the performance of these steps may affect each other; for example, if the extracted features are not effective, the feature selection and classification steps can not achieve a satisfactory performance; and (ii) their implementation requires rich domain knowledge, needed, for example, to choose informative features.

In this framework, Vectorial Genetic Programming, VE_GP [7], can be a promising tool to automate feature extraction from online handwriting data, since it allows us to directly exploit the raw data without applying any feature selection technique and without using any apriori knowledge on the problem at hand.

VE_GP is a method that extends Genetic Programming (GP) [8, 9] by using functions that take in input vectors (e.g., time series). VE_GP provides a rich and easily extensible set of primitive operations for directly processing time series data, which do not need to be collapsed in a predefined set of features. Furthermore, the inclusion of aggregating parametric functions in the primitive set allows VE_GP to automatically generate informative features from data without any need for human intervention. Last but not least, VE_GP can generate models that are simple enough to be read and interpreted, in some cases offering some valuable insights on the application [10, 11]. It is worth noting that the capability to provide interpretable models is of paramount importance in the medical field, where trust is at the basis of the employment of machine learning models in practice [12].

Starting from the above considerations, in a previous article, we used VE_GP to classify the handwriting of people affected by AD based on their ability to accomplish

nine well-defined tasks [13]. The proposed VE_GP-based system took in input the time series of the X and Y coordinates and the pressure of the movements performed to accomplish the handwriting tasks acquired using a graphic tablet. The experimental analysis on a dataset containing data from 130 participants showed that VE_GP outperformed a decision tree classifier trained on a set of features previously presented in [14].

This paper presents a further development of the research activity first presented in [14]. In particular, we extended our analysis to the 25 tasks of the experimental protocol presented in [15], on a larger version of the dataset analyzed in [7]. To further assess the effectiveness of our approach, we compared its results with those achieved by a standard feature engineering approach [6] and a deep learning based approach that shares with our system the automatic feature extraction ability [16]. In the first case, we compared our results with those achieved by nine well-known and widely-used classifiers, whereas in the second case, the comparison was made with the results achieved by four deep neural networks. Finally, we also investigated the behavior of VE_GP during evolution, considering both the complexity (in terms of the number of functions), and the interpretability of the models evolved.

In summary, the primary objective of the research discussed in this article is to formulate a novel methodology for discerning alterations in the handwriting of individuals affected by Alzheimer's Disease (AD). This is achieved by leveraging the ability of VE_GP to autonomously extract features from raw time series data. To fulfill this objective, we enhanced the conventional vectorial functions of VE_GP by incorporating a suite of functions tailored to the specific challenges of the problem. The resulting system demonstrates the capability to generate insightful and distinctive models from raw data, that incorporate several vectorial functions.

The paper is organized as follows. After a brief overview of the research activities related to our study (Section 2), Section 3 introduces VE_GP, whereas Section 4 describes the handwriting data analysed; Section 5 details the experimental setup, whereas Section 6 reports the experimental results; finally, Section 7 concludes the paper and proposes ideas for future research.

2. Related Work

Thanks to their search-ability, Evolutionary Algorithms [17] have often been employed in health applications, particularly Genetic Algorithms (GAs) and GP. GP has been used in a wide range of applications. For example, [18] proposed a constrained-syntax GP-based algorithm for discovering classification rules in medical data. The authors tested their approach on five datasets and achieved better results than decision trees. In [19], the authors presented a novel approach based on Geometric Semantic GP (GSGP) to solve a problem related to the physio-chemical properties of proteins, involving the prediction of these properties in tertiary structure. GSGP uses specific genetic operators that induce a unimodal error surface for any supervised learning problem, independently of the complexity and size of the underlying data set. The authors demonstrated that the proposed approach was more effective than a set of state-of-the-art Machine Learning algorithms, such as Artificial Neural Networks and Support Vector Machines. GP has also been used as a tool to support medical decisions for treating rare diseases. In particular, in [20], the authors developed and tested six predictive data models using GP. The models were integrated into a web-based application to be used by therapists to support them in their patient's care and treatment activities. More recently, Parziale et al used a Cartesian GP to automatically identify people affected by Parkinson's disease (PD) by analyzing their handwriting and drawings [21]. The proposed approach allowed them to support the diagnosis of PD through explicit classification models, which provided results that physicians could interpret.

GAs have been widely used in medical applications, in most of the medical specialties, among which medical imaging, rehabilitation medicine, and health care management [22]. Regarding neurodegenerative diseases, in [23], the authors used a multi-objective GA to find the relevant volumes of the brain related to Alzheimer's disease. In [24], a GA was used to search for the optimal set of neuropsychological tests for building a predictive model of Alzheimer's disease. More recently, in [2], a GA-based system was presented to improve the performance of a system that had previously been proposed for revealing cognitive impairments through handwriting analysis.

An important focus of our work is the use of GP for feature extraction. The use

of GP for this purpose is not new, and it finds its most active application area in image processing. For instance, in [25], a feature extraction method using multi-objective GP (MOGP) was presented. This method was applied to the well-known edge detection problem in image processing, outperforming the studied baseline algorithms. The authors claimed that the success of the proposed method was mainly due to its effective feature extraction. Two years later, in [26], the use of GP for constructing vision systems was explored. A two-stage approach was proposed, with a separate evolution of the feature extraction and classification stages. A GP-based strategy for feature construction for 2D images, called Genetic Program Feature Learner (GPFL), was proposed in [27]. GPFL executes multiple GP runs. Each run generates a model that focuses on a particular high-level feature of the training images. Then, it combines the models generated by each run into a function that reconstructs the observed images. The presented results indicated that when considering smaller training sets, GPFL achieves comparable/slightly better classification accuracy than state-of-the-art methods such as LeNet5. Also, GPFL was shown to drastically outperform LeNet5 when considering noisy images as test sets. Recently, in [28], the success of GP in image classification was discussed. In particular, the authors presented a GP approach that can conduct feature extraction, feature construction, and classification automatically and simultaneously. It can extract and construct informative image features, select a suitable classification algorithm instead of relying on a predefined classifier, and perform classification for binary and multi-class image classification tasks.

GP was also used for feature extraction in the medical field. For instance, in [29] the authors presented an electroencephalogram (EEG) classification system for detecting mental states. The system integrated a GP-based enhanced feature extraction algorithm (called Augmented Feature Extraction with Genetic Programming, or +FEGP), to search for non-linear transformations able to build new features. +FEGP can be paired with any classification method, outperforming previously existing results in the studied application. M. Zhang and his colleagues are among the most productive contributors in the use of GP for feature extraction, and their work is particularly significant for us because, in several cases, they applied their methods to the medical area. For instance, in [30] they developed a classification system that combines feature con-

struction and ensemble learning using GP. The proposed method was employed for skin cancer detection, using two different skin image datasets. The experimental results indicated that the proposed algorithm significantly outperforms two existing GP approaches, two state-of-the-art convolutional neural network methods, and ten commonly used machine learning algorithms. Many of the contributions of this group of researchers in the area of feature extraction with GP have recently been gathered in the book [31]. Also, recently these authors have proposed a novel diagnosis approach integrating an automatic feature extraction and construction performed by GP (the method was called AFECGP) [32]. AFECGP was shown to automatically generate informative and discriminative features from original vibration signals for identifying different fault types of rotating machinery. In [33], different approaches to constructing multiple features using GP were investigated. The results indicated that multiple-feature construction achieves significantly better performance than single-feature construction. In multiple-feature construction, using multi-tree GP representation was shown to be more effective than using the single-tree GP, thanks to the ability to consider the interaction of the newly constructed features during the construction process. Also, the authors showed that class-dependent constructed features achieve better performance than class-independent ones and contribute to generating interpretable models.

Another area in which the ability of GP to construct new features was recently exploited was land cover classification. In [34], the M3GP variant of GP (specialized for multi-class classification) was used on several sets of satellite images over different countries to create hyperfeatures from satellite bands to improve the classification of land cover types. Hyperfeatures were evolved to the reference datasets and a significant performance improvement was observed compared to a set of state-of-the-art algorithms. Detection of malware is another area in which GP was successfully used for feature extraction. In [35], a system was developed in which the One Side Class Perceptron algorithm was coupled with GP, and GP acted as a feature extraction technique. The proposed method was demonstrated successful. In particular, it was shown that the features produced by GP are better than the best ones obtained by other methods, allowing for an increase in the detection rate.

If we extend our analysis to the wider field of Artificial Intelligence (AI), several

expert opinions seem to indicate the appropriateness of investing in AI research for the early diagnosis of AD [36], and a survey of AI studies for the early diagnosis of AD can be found in [37].

The brief literature review outlined above shows that most GP-based approaches for feature extraction work with images. On the other hand, those working with other types of data typically take input data different from time series, like executable files [35] or gene expression datasets [33]. The only approach working with time series is the AFECGP method for fault detection in rotating machines [32]. However, AFECGP works with fixed-length times series, whereas, as explained in the following section, VE_GP can work with time series whose length is not fixed a priori. Therefore, to the best of our knowledge, this is the first study in which a GP-based approach is used on raw data consisting of variable length time series.

3. Vectorial Genetic Programming

Vectorial genetic programming (VE_GP) is a variant of GP that allows us to deal with time series as predictors or targets. VE_GP enables the use of vectors as terminals, providing a suitable representation for time series, and supplies a set of primitive functions to operate on vectors. In the continuation of this section, we describe these parametrizable primitives, first presenting the functions of arity 1 and 2 previously defined and then the problem-specific functions implemented. Next, we discuss other peculiar characteristics of VE_GP that distinguish it from traditional GP, such as the population initialization and the parameter mutation, which allows us to change the parametrization of the primitive functions. Finally, we discuss how we employed VE_GP for classification in this work. Further details about the primitive functions implemented in VE_GP to manage vectors can be found in [7].

3.1. Functions of arity 1

VE_GP's primitive functions, when applied to vectors, typically return a scalar value that, informally speaking, "represents" the elements of the vector, or "aggregates" them into one single value, able to capture some characteristics of the vector. For this

reason, in some cases, they will be referred to as aggregating functions. VE_GP provides several aggregating functions, along with their parameters. Parameters define the part of the vector (or "window") where the function will be applied. The window can move along the whole vector. Parameters p and q are randomly set at the beginning of a VE_GP run and then evolve during population evolution thanking to specific genetic operators. Let $[v_1, v_2, ..., v_p, ..., v_q, ..., v_n]$ be a vector and let p and q be two indexes, with $p < q$; then $f_{p,q}(v) = f([v_p, \ldots, v_q])$. In case the window extends to not existing elements of the vector, these elements are not included in the calculation. To provide a numerical example let us consider $v = [1, 2.5, 4.3, 0.7, 1.6]$ and $(p, q) = (2, 3)$. The function $\texttt{V\_sum}_{2,3}$ applied to v returns 6.8.

Here we used the following functions of arity 1:

- `V_maxpq, V_minpq`: returns the maximum/minimum of the sub-vector defined by the range $[p, q]$;
- `V_sumpq`: returns the sum of the elements of the sub-vector defined by the range $[p, q]$;
- `V_meanpq`: returns the mean of the elements of the sub-vector defined by the range $[p, q]$;

3.2. Functions of arity 2

Regarding functions of arity 2, they are simply the extension to vectors of the arity 2 functions between scalars often used by traditional GP. In particular, when the inputs of a function are two vectors of length greater than 1, the shorter is completed with the null element of the function up to the length of the longer before applying the function itself. Differently, when a scalar and a vector of length greater than 1 are the inputs, the scalar is initially replicated up to the length of the other vector input.

To provide a numerical example, let us consider two vectors: $v = [1, 2.5, 4.3, 0.7, 1.6]$ and $w = [0.8, 3.6, 1.9]$. The function `VSUMW` applied to v and w returns $\texttt{VSUMW}(v, w) = [1.8, 6.1, 6.2, 0.7, 1.6]$.

Here we used the following functions of arity 2:

- `VSUMW`: computes the element-wise sum between two vectors;

– `V_W`: computes the element-wise difference between two vectors;

– `VprW`: computes the element-wise product between two vectors;

– `VdivW`: computes the protected element-wise division between two vectors; if the divisor is close to 0, then 1 is returned.

3.3. Problem-specific functions

Exploiting the advantage of VE_GP to easily include new functions, in order to simulate some of the features extracted by the standard classification approach, we included innovative functions in VE_GP. In particular, with the help of domain experts, we have defined the following functions:

– `V_normpq`: computes the 2-norm of the sub-vector defined by the range $[p, q]$; This function extracts spatial information since it calculates the distance from the border (left or bottom, depending on the feature involved) of the handwriting trait defined by $[p, q]$. In the case of the pressure (Z coordinate), it computes the amount of pressure exerted in that trait;

– `V_distpq`: computes the one-dimension Euclidean distance between the element of position p and the element of position q of the input vector, thus $|p - q|$; This function extracts spatial information since it computes the horizontal/vertical (depending of the feature involved, horizontal for the feature capturing the position of the trait on the horizontal-axis, vertical for the feature capturing the position of the trait on the vertical-axis) extension of the handwriting trait by $[p, q]$;

– `V_length`: computes the length of the vector. This function extracts temporal information since it allows us to gain knowledge about the duration of the task;

– `V_duration0`: computes the total duration of the task in-air traits (the pen tip is lifted from the sheet, see Section 4).

The above-defined functions show that VE_GP can be straightforwardly adapted to a specific application domain (they are simple and easy to implement). Even more

importantly, these problem-specific functions provide useful information, making the models using them easy to interpret. Furthermore, it is important to remark that all the functions detailed above (also those of arity one and two described in the previous subsections) take in input vectors (time series in our case) whose length must not be apriori defined.

Finally, it is worth highlighting that, in general, problem-specific functions can be defined by asking the domain experts what information (of interest for that specific problem) to extract directly from the vectors containing the problem's raw data.

3.4. Initialization

VE_GP proposes a peculiar initialization method with the objective of using in a correct way the aggregating functions included in the primitive set. A portion of n_1 individuals are forced to apply aggregating functions only to vectorial variables. This feature ensures a pool of individuals that properly use aggregating functions. The remaining n_2 individuals are generated with one of the classical initialization techniques [9]. Both n_1 and n_2 are parameters of VE_GP that must be set by the user.

3.5. Parameter mutation

The genetic operator of parameter mutation (PM) is developed in VE_GP to let the evolution find the most informative time windows. PM simply looks in an individual for parametric aggregating functions, randomly selects one of them, and randomly changes one of its parameters. The parameter is mutated without violating the rule that $p < q$.

3.6. VE_GP for classification and fitness evaluation

VE_GP was originally introduced to deal with time series forecasting. However, it can be easily extended to classification. Classification, in fact, can be performed using the individuals as discriminating functions at threshold 0, as for instance in [38]. In this way, evaluations ≥ 0 correspond to class 0, while the others correspond to class 1. Given that we need a scalar output to apply this threshold function, in this work, if an individual returns a vector, a mean is applied to it in order to obtain a scalar.

4. Dataset

Our handwriting data were collected according to the acquisition protocol described in [15]. The protocol includes 25 tasks, belonging to the following categories (see Table 1): (i) graphic (tasks 2-5, 21, 24); (ii) Copy (tasks 6-13, 15-17, 18); (iii) Memory and dictation (tasks 1, 14, 18, 20, 23). The dataset contains data from 174 participants: 89 AD patients and 85 healthy people.

The participants in the study were recruited with the support of the geriatric ward, Alzheimer unit, of the "Federico II" hospital in Naples (Italy). The cognitive abilities of the participants were evaluated using standard clinical tests, namely, Mini-Mental State Examination (MMSE), Frontal Assessment Battery (FAB), and Montreal Cognitive Assessment (MoCA). These tests use questionnaires to assess cognitive skills covering many areas, ranging from orientation in time and place to registration recall.

The handwriting movements of the participants while performing the 25 tasks were collected by using a Wacom's Bamboo tablet equipped with a pen that allowed participants to write in normal ink on A4 white paper sheets. The tablet sampled three values (X, Y, and Z) for each point at a constant sampling rate equal to $200Hz$. The first two coordinates (X and Y) are the pen tip position in the two-dimensional space representing the surface where the writing is produced, whereas the third one (Z) is the pressure exerted by the pen tip at that point. Pressure values are positive when the pen tip touches the sheet and are null when the pen tip is lifted from the sheet. In the following, we will refer to the first points as "on-paper" and to the second as "in-air". Note that the tablet recorded in-air movements of the pen tip up to a maximum distance of 3 *cm* from the sheet.

5. Experimental Setup

For each task separately, we evolved a VE_GP classifier through the following process.

1. We randomly partitioned the available data into 5 folds of identical size, each containing the same proportion of AD and healthy samples as the original dataset (five-fold stratified sampling). One random fold was selected as the test set,

Table 1: The tasks involved in the analysis.

Task#	Description
1	Signature drawing
2	Join two points with a horizontal line, continuously for four times
3	Join two points with a vertical line, continuously for four times
4	Retrace a circle (6 cm in diameter) continuously for four times
5	Retrace a circle (3 cm in diameter) continuously for four times
6	Copy the letters 'l', 'm' and 'p'
7	Copy the letters on the adjacent rows
8	Write in cursive a sequence of four lowercase letters 'l', in a single smooth movement
9	Write in cursive a sequence of four lowercase cursive bigrams 'le', in a single smooth movement
10	Copy the word "foglio"
11	Copy the word "foglio" above a line
12	Copy the word "mamma"
13	Copy the word "mamma" above a line
14	Memorize the words "telefono", "cane", and "negozio" and rewrite them
15	Copy in reverse the word "bottiglia"
16	Copy in reverse the word "casa"
17	Copy six words (regular, non-regular, non-words) in the appropriate boxes
18	Write the name of the object shown in a picture (a chair)
19	Copy the fields of a postal order
20	Write a simple sentence under dictation
21	Retrace a complex form
22	Copy a telephone number
23	Write a telephone number under dictation
24	Draw a clock with all hours and put hands at 11:05 (Clock Drawing Test)
25	Copy a paragraph

3 out of the remaining folds were randomly selected as the training set, while the 4^{th} was used as the validation set.

2. We run a VE_GP algorithm based on training set data with parameters set according to Table 2. The accuracy of the evolving individuals was used as the fitness function during this training. Accuracy is defined as the percentage of correctly classified samples; therefore, each problem was turned into a maximization one.

3. We evaluated the accuracy of each individual of the final population on the validation set. The individual with the best accuracy was selected as the classifier (best individual), and its performance was measured as the accuracy on the test set.

Table 2: Parameters setting of the algorithm of VE_GP.

Parameter	Value
maximum numb. of generations	100
population size	200
genetic operators	subtree crossover, subtree mutation, parameter mutation
genetic operators probabilities	0.5, 0.1, 0.4
initialization	n_1 = 30%, n_2 = 70% Ramped Half-and-Half
functions set	`VSUMW, V_W, VprW, VdivW, V_maxpq, V_minpq, V_meanpq, V_sumpq, V_normpq, V_distpq, V_length, V_duration0`
terminals	input variables + randomly generated numbers (as constants)
fitness	accuracy
parents selection	Lexicographic Parsimony Pressure
elitism	keep best
maximum depth of the individuals	17

4. We repeated the previous steps for 30 independent runs (stochastic cross-validation).

All results reported in the following section have been averaged over the 30 runs performed.

On average, the training step took approximately 1940 seconds (32 minutes and 24 seconds) for each run. A run of Task 4 required the longest time, i.e., 9430 seconds (157 minutes and 20 seconds) for each run, whereas a run of Task 13 required the shortest time, i.e., 576 seconds (9 minutes and 36 seconds) during evolution.

6. Experimental Results

To implement VE_GP we used GPLab, a Genetic Programming toolbox for Matlab [39]. We performed the experiments detailed below on an Intel Xeon CPU E7-8880 v4 @2.20GHz equipped with 256 GB of RAM. As a software framework, we used Matlab R2021a (Update 1) and Red Hat Enterprise Linux (release 8.9).

To assess the effectiveness of our system and investigate its behavior, we carried out several experiments. In the first set of experiments, we tested the generalization ability

of VE_GP and its performance in terms of specificity and sensitivity. In the second set of experiments, we investigated the behavior of VE_GP during evolution. In the third set, we analyzed the complexity of the solutions found, whereas, in the fourth, we compared our results with those presented in [6] and those achieved by using a deep learning approach [16]. Finally, we analyzed the features extracted by VE_GP. The experiments are detailed in the following subsections.

6.1. Testing the generalization ability of VE_GP

To test the generalization ability of VE_GP we plotted a histogram where, for each task, a couple of bars show the accuracy achieved on the learning set (made of the training and validation sets) and on the test set for that task. This histogram is shown in Figure 1. The aim of this histogram is twofold. On the one hand, it provides an overview of the performance achieved by VE_GP on the 25 tasks of the protocol. On the other hand, it allows the detection of possible overfitting visually.

Looking at the histogram, we can observe that, as expected, VE_GP performed differently on the 25 tasks. On tasks 18 and 23 (writing the name of an object and copying a telephone number, respectively) VE_GP achieved an accuracy greater than 0.80. It is worth noting that these tasks require writing short sequences: five letters for task 18 (the Italian word ”sedia” for the chair in the figure) and seven figures for the telephone number. The other tasks on which VE_GP achieved a good performance (accuracy approximately equal or greater than 0.75) are 7 (copying the letters ’n’, ’l’, ’o’ and ’g’), 9 (writing four times the bigram ’le’), 10 (copy the word ”foglio”), 11 (copy the word ”mamma”), and 12 (copy the word ”mamma” above a line). These results confirm the ability of VE_GP to achieve a good performance on short sequences. Regarding the tasks on which VE_GP achieved the worst performance (accuracy approximately equal or less than 0.65), they are 4 (retrace a 6 cm circle for four times), 14 (memorize and rewrite the words ”telefono”, ”cane”, and ”negozio”), and 21 (retrace a complex form). These tasks require writing longer sequences, and the related results confirm that VE_GP performs better with short sequences. To statistically validate our findings, we compared the length of the worst-performing tasks (4, 14, and 21) with the remaining ones considered in our analysis. To this aim, we computed the average length for

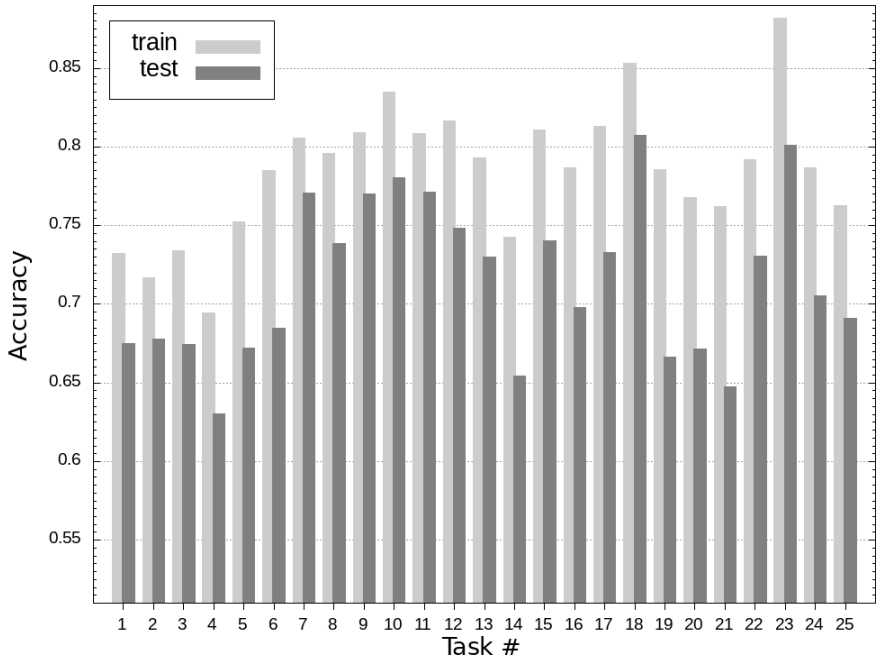


Figure 1: Learning and test accuracy achieved on each task.

each task by averaging the number of time series points of all participants. To statistically validate the comparison, we used the Wilcoxon rank-sum test ($\alpha = 0.05$), a non-parametric statistical test. Table 3 shows the comparison between each pair of tasks: the cell $[i, j]$ of the table contains the symbols '+', '-', and '$\approx$', if the average length of the task i is respectively longer, equivalent, or shorter than the task j according to the Wilcoxon test. From the table, we can observe that the worst-performing tasks are longer than the remaining ones and the length differences are statistically significant.

To investigate whether VE_GP was affected by overfitting, we considered the difference between the learning and test accuracy achieved on a given task (referred to as "learning-test difference" in the following). Then, we analyzed the histogram from two perspectives. First, we investigated the test accuracy achieved on the tasks with the highest and lowest learning-test differences. From the second perspective, we analyzed this difference for the best and worst tasks in terms of test accuracy. The histogram shows that learning-test differences range from values equal to or greater than 0.10

Table 3: Length comparison.

Task #		7	9	10	11	12	18	23
	length	1940±320	2050±380	1390±300	1290±230	1390±220	1020±250	2270±330
4	5830±660	+	+	+	+	+	+	+
14	3070±480	+	+	+	+	+	+	+
21	8400±1150	+	+	+	+	+	+	+

(tasks 6, 19, and 21) to less than 0.04 (tasks 2, 7, 9, and 11). As for the highest learning-test differences, test accuracy values vary in the range $[\approx 0.65, \approx 0.68]$, which do not correspond to the worst results, whereas the lowest learning-test differences vary in a much wider range: $[\approx 0.68, \approx 0.77]$. On the other hand, from the histogram, we can also observe that the learning-test differences for the best-performing tasks (18 and 23, see above) are 0.05 and 0.08. In contrast, those for the worst ones (4, 14, and 21, see above) range from ≈ 0.06 to ≈ 0.11. Overall, these results suggest that: (i) the low performance of VE_GP is not due to overfitting (e.g., task 4); (ii) also the best-performing tasks may have a significant learning test difference. Then, we can conclude that overfitting did not compromise the performance of VE_GP.

To further assess the effectiveness of VE_GP in discriminating the handwriting of people affected by AD, we also evaluated its performance in terms of sensitivity and specificity. In a diagnostic test, the first measures how well the test identifies positive cases (people affected by AD in our case), whereas the second measures how well the test classifies negative cases (healthy people in our case). We computed those measures as follows:

$$Sensitivity = \frac{TP}{TP+FN}; \qquad Specificity = \frac{TN}{TN+FP};$$

Where TP is the number of true positives, FN is the number of false negatives, FP is the number of false positives, and TN is the number of true negatives.

Those results are shown in Table 4. From the table, we can see that, on average (last row), VE_GP achieved a sensitivity (0.82) higher than specificity (0.71), with similar standard deviations, higher than that of accuracy. This last result depends on the fact that accuracy is a weighted sum of sensitivity and specificity with respect to the fraction of positive cases. From the table, we can also observe that VE_GP achieved

Table 4: Accuracy, sensitivity, and specificity results.

Task #	Accuracy		Sensitivity		Specificity	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
1	0.67	0.03	0.76	0.10	0.73	0.09
2	0.68	0.04	0.72	0.17	0.81	0.09
3	0.67	0.03	0.86	0.11	0.68	0.14
4	0.63	0.04	0.78	0.11	0.66	0.12
5	0.67	0.03	0.79	0.10	0.68	0.10
6	0.68	0.03	0.67	0.15	0.84	0.09
7	0.77	0.01	0.78	0.09	0.86	0.08
8	0.74	0.03	0.74	0.14	0.83	0.08
9	0.77	0.04	0.83	0.08	0.85	0.08
10	0.78	0.03	0.96	0.03	0.58	0.13
11	0.77	0.04	0.60	0.11	0.91	0.05
12	0.75	0.03	0.93	0.05	0.58	0.14
13	0.73	0.03	0.62	0.23	0.91	0.07
14	0.65	0.04	0.83	0.13	0.61	0.13
15	0.74	0.05	0.84	0.11	0.77	0.11
16	0.70	0.04	0.83	0.06	0.65	0.11
17	0.73	0.03	0.82	0.08	0.73	0.09
18	0.81	0.01	0.93	0.03	0.67	0.10
19	0.67	0.05	0.83	0.13	0.57	0.15
20	0.67	0.05	0.85	0.11	0.66	0.16
21	0.65	0.03	0.91	0.08	0.41	0.15
22	0.73	0.07	0.90	0.05	0.67	0.17
23	0.80	0.03	0.87	0.06	0.78	0.07
24	0.71	0.04	0.91	0.07	0.62	0.14
25	0.69	0.05	0.84	0.10	0.67	0.14
Average	0.71	0.04	0.82	0.10	0.71	0.11

a good sensitivity and specificity performance ($\geq$ 0.90) on tasks 6 and 2, respectively. Looking at the best-performing tasks in terms of accuracy ($>$ 0.75), namely 7 and 9 (0.77), 10 (0.78), 18 (0.81), and 23 (0.80), we can note that also in this case, on average, we achieved a sensitivity higher than specificity. From the table, we can see that in task 10 we achieved a sensitivity equal to 0.96. Overall, the above results are good for a diagnostic test since, typically, the costs the costs of not identifying sick people are much higher than those of misclassifying healthy people as sick.

6.2. Investigating the behaviour of VE_GP

In the second set of experiments, we investigated the behavior of VE_GP during evolution. To this aim, for each task, we plotted the average and best fitness values as

a function of the generation number, averaged over the thirty runs performed. Twelve out of the twenty-five plots drawn are shown in Figure 2 (the other ones showed similar trends). From the figure, we can observe that for some plots, there was a slow and steady convergence of the average fitness towards a sort of ”target value” (e.g., tasks 1, 6, 22, and 25). Looking at these plots, we can see that the best fitness also follows this trend. This behavior confirms the exploration ability of VE_GP, avoiding the best individual's premature convergence toward a local optimum. It is also worth noting that this ability was maintained when the average fitness increased rapidly from the first generations (e.g., tasks 6, 9, 11, and 13). Indeed, also in these cases, the best fitness keeps the same slow trend seen above. Even more importantly, from all plots, we can see a wide distance between the average and best fitness until the last generations. This wide distance confirms that, on average, the individuals in the population are quite different from the best one. Therefore, we can state that VE_GP is not subjected to any genetic drift phenomena. Overall, we can conclude that VE_GP has a good exploration ability in the solution space made of the tree-based models used to distinguish the handwriting of people affected by AD.

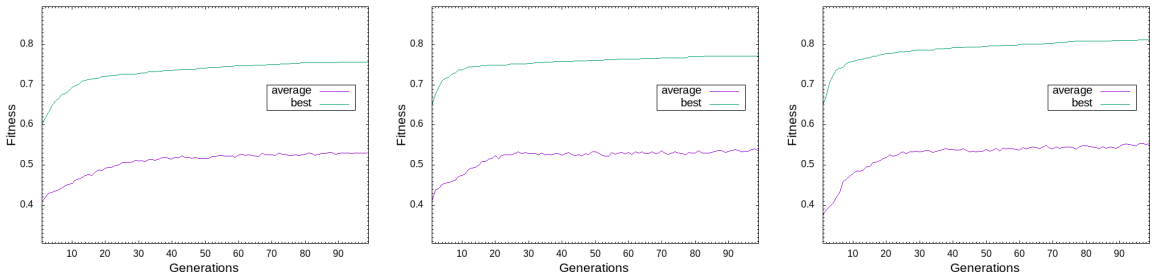
6.3. Comparison findings

To test the effectiveness of our approach, we compared its accuracy performance with a standard feature engineering approach [6] and a deep learning-based approach, which shares with our system the automatic feature extraction ability [16]. In [6], the authors extracted features suggested by clinicians (e.g., velocity, acceleration, and pressure), whereas in [16], the authors used a deep learning-based approach to classify images generated from online handwriting data. It is worth noticing that both studies used our own dataset, described in Section 4.

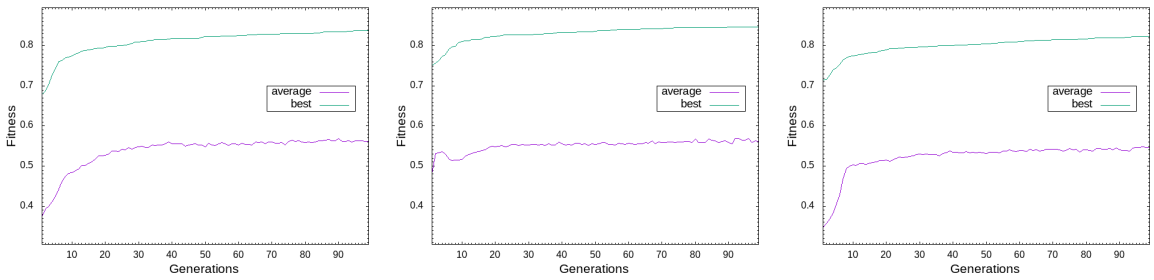
6.3.1. Standard feature engineering approach

From the data described in Section 4, the authors of [6] extracted 18 features, including total time, velocity, acceleration, jerk, and pressure [1]. For a more detailed explanation of these features, see Section 3 of [6].

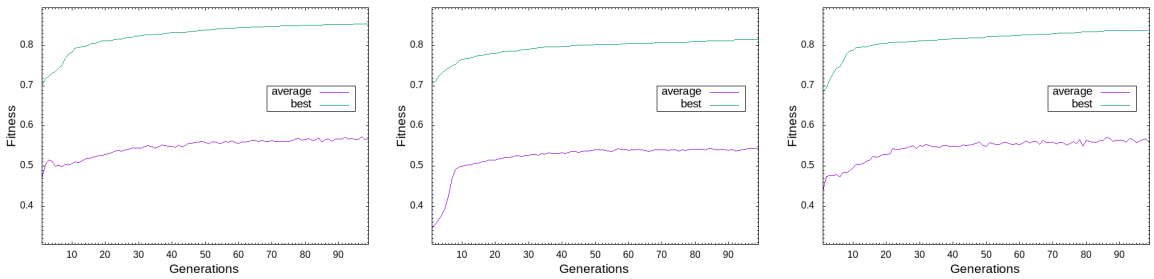
[1] The dataset is publicly available on the following page: `https://archive.ics.uci.edu/dataset/732/darwin`



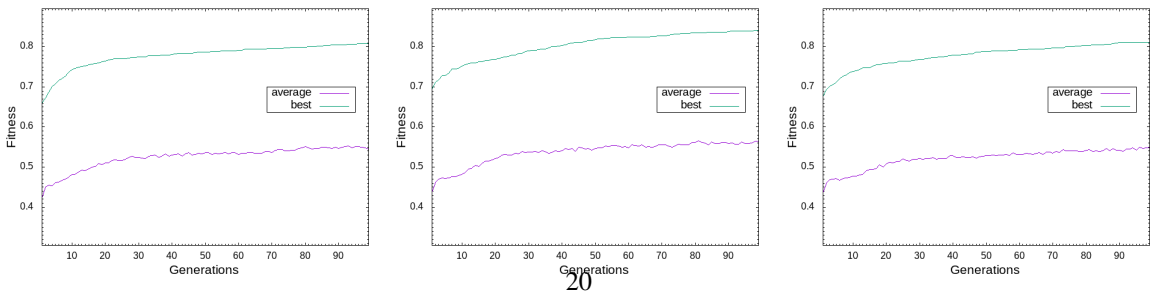
(a) Task 1 (b) Task 3 (c) Task 6



(d) Task 9 (e) Task 10 (f) Task 11



(g) Task 12 (h) Task 13 (i) Task 17



(j) Task 20 (k) Task 22 (l) Task 25

Figure 2: Evolution of the average and best fitness.

Once the feature had been extracted, the related data was used to train seven classifiers: Decision Tree (DT), Gaussian Naive Bayes (GNB), Linear Discriminant Analysis (LDA), Logistic Regression (LR), Learning Vector Quantization (LVQ), Multilayer Perceptron (MLP), and Support Vector Machine (SVM). In the training phase, for each classifier, the authors optimized the related hyperparameters. To statistically validate the results, we used the Wilcoxon rank-sum test ($\alpha = 0.05$). Comparison results are shown in Table 6. Whenever VE_GP's performance was found to be statistically superior, worse, or equivalent to any other classifier (according to the Wilcoxon rank-sum test), the symbols '+', '-', and '$\approx$' were used, respectively. To provide a "task-based" overview of the comparison results, the last row also shows how many times VE_GP performed better/equivalently/worse (w/t/l) than the compared approaches.

From Table 6, we can observe that in terms of win/tie/loss (see last row), VE_GP outperformed all classifiers used for the comparison. As for the task-based point of view, from the table we can see that VE_GP:

- outperformed the seven compared approaches on five tasks (from 10 to 13, and 18). It is interesting to note that although VE_GP did not achieve its better performance (except for task 18) on those tasks, it achieved a much higher accuracy than all the compared methods. This result confirms the ability of VE_GP to find well-performing models (accuracy ≥ 0.70) on data where the state-of-the-art algorithms have performed poorly.
- achieved superior or equivalent performance on nine tasks (1, from 7 to 9, 15, 22, 23, 24);
- achieved equivalent performance on the seven compared approaches on four tasks (3, 17, 20, and 25);
- achieved equivalent or worse performance on five tasks (2, from 4 to 6, and 16);
- was never outperformed by all the compared approaches.

6.3.2. Deep learning based approach

To further investigate the effectiveness of our approach, we compared its results with those achieved by three Convolutional Neural Networks (CNNs). We trained

those CNNs on synthetic RGB images built from the same online handwriting data used for our approach. In practice, from those data, we built a synthetic image for each task performed by a given participant by interpolating consecutive points of the time series representing the movements performed by that participant to accomplish that task. Furthermore, to exploit the dynamics information in the time series, each image color channel contained information about the handwriting movements, i.e., velocity, jerk, and pressure. We built two types of images. The images of the first type were built by interpolating only the on-paper points (those produced when the pen tip touched the sheet). In contrast, the images of the second category were built by interpolating both on-paper and in-air points (the pen tip was lifted from the sheet). Further details about the image construction process can be found in [16, 40].

We used those images to re-train three widely used and publicly available CNNs, namely InceptionV3 (IV3), ResNet50 (R50), and VGG19 (V19), which were trained on Imagenet, a database containing millions of annotated images. The CNNs were re-trained according to a two-step procedure. In the first step, we adopted a transfer learning approach by re-training only the final fully connected layers, whereas the remaining convolutional layers were frozen. In the second step, we implemented a fine-tuning approach by re-training all the layers.

The obtained results are shown in Table 7. From the table, we can observe that VE_GP outperformed the three CNNs, on both types of images. In the best case (R50 on on-paper data) VE_GP achieved better results on 23 out of 25 tasks and tied on the remaining two tasks. On the other hand, in the worst case (V19 on in-air & on-paper) VE_GP achieved better results on 15 out of 25 tasks, two ties, and eight worse results. From the table, we can also observe that VE_GP was the best-performing method on 14 tasks (1, 3, 7, 9, 10-13, 15, 17, 18, 22-24).

These results suggest that the size of the training set limited CNN performance, and the fine-tuning did not provide a significant improvement. On the other hand, it is worth noting that the small size of the training data did not limit VE_GP's ability to find good solutions.

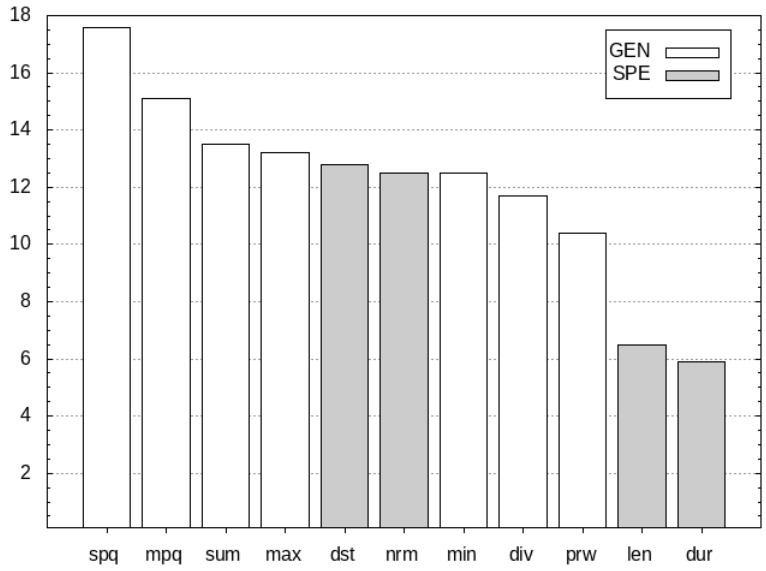


Figure 3: Number of occurrences of the functions in the best individuals for the 25 tasks averaged over the thirty runs. White bars represent the general-purpose functions (GEN), whereas the grey bars represent the problem-specific functions (SPE). The acronyms are the following: `V_sumpq` (spq), `V_meanpq` (mpq), `V_sumw` (sum), `V_maxpq` (max), `V_distpq` (dst), `V_normpq` (nrm), `V_minpq` (min), `V_divw` (div), `V_prw` (prw), `V_length` (len), `V_duration0` (dur).

6.4. Models Evolved by VE_GP and extracted Features

As the last experiment, we analyzed the features automatically extracted by VE_GP, using the set of functions used (see Table 2). We first investigated the use of those functions. To this aim, we counted the number of occurrences in the best individuals for each function and averaged it over the thirty runs carried out. We plotted the histogram shown in Figure 3 with those values. We used white and grey bars to distinguish the general-purpose functions (GEN) from the problem-specific ones (SPE). To avoid flattening the remaining bars, we did not plot the bar of the highest value, i.e., 76.3 achieved by the `V_W` function. As detailed in Subsection 3.2, `V_W` is a function of arity 2, which computes the element-wise difference between the two input vectors. As concerns the other three general purpose element-wise functions of arity 2, namely `VsumW` (summation), `VprW` (product), and `VdivW` (protected division), their number of occurrences are 13.5, 10.4, and 11.7, respectively. Those values can be considered comparable and confirm that the element-wise difference function is more important

for the automatic feature extraction process implemented by VE_GP. From the plot, we can also observe that the number of occurrences of `V_distpq` and `V_normpq` is comparable with that of the GEN functions. This result confirms the importance of those functions, used on most tasks, whereas the other two functions (`V_length` and `V_duration0`) were used in a limited number of tasks. It is also interesting to note that the first two functions extract information from the part of the input vector (time series) represented by the subvector defined by the range $[p, q]$. In practice, that result confirms that, on most tasks, it is important to extract the features from specific parts of the task.

We also investigated the importance of the three coordinates used, namely the (X,Y) pair representing the position of the pen tip on the surface of the paper sheet, and Z, which is the pressure of the pen tip on the sheet. To this aim, we computed the number of occurrences of those coordinates on the thirty runs carried out for each task. We achieved the following average values and standard deviations:

X: 21.1 (3.8)

Y: 19.9 (3.5)

Z: 18.0 (3.8)

This result confirms that the three coordinates were equally used by the automatic feature extraction process implemented by VE_GP.

As mentioned in the Introduction, one of the most interesting characteristics of VE_GP is the ability to generate simple, readable models. This ability is crucial to any machine-learning algorithm, especially in medicine. To investigate VE_GP's ability to generate simple and readable models, we analyzed the models evolved by the best run for each task. The analysis of the learned classifiers provides hints about the most discriminating time series ranges and features. In practice, this allows us to characterize the fine movements of AD patients with respect to those of healthy people. For example, some models suggest that for the related tasks the most important feature to consider is pressure, whereas others highlight the importance of the vertical position, which is related to the spatial organization ability.

For the sake of conciseness, in the following we report the analysis of some of them. These models are shown in Table 5. For the sake of readability, the operators '+', '-

', and '/' represent the element-wise functions `VSUMW`, `V_W`, and `VdivW` respectively. Furthermore, as detailed in Subsection 3.6, if an individual returns a vector, a mean is applied to it in order to obtain a scalar (evaluations ≥ 0 correspond to class 0, while the others correspond to class 1).

Overall, from the table we can see that some models are very simple and give precise indications of which task parts or features have been paid attention to by the VE_GP. For example, from the table we can see that for tasks 2 and 21 (Join two points with a horizontal line and retrace a complex form, respectively), the most important quantity to consider is pressure; these models also suggest that pressure differences (task 21) or the duration of the in-air traits (task 2) are important. These models suggest that, in this case, both on-paper (pressure) and in-air traits (their duration) are discriminant for distinguishing the handwriting of AD patients. The hint provided by these models is that AD patients lose fine motor control abilities both for on-paper and in-air traits. From the table we can also observe that the models evolved for the copy tasks 9 and 17 (writing four times the bigram "le" and copying six words, respectively) use all three coordinates (task 9) or the vertical position of some traits (identified by the related ranges). In particular, the model of task 9 uses temporal (function `V_length`) and spatial information (function `v_sumpq` on the X coordinate) whereas the model of task 17 uses only spatial information (function `v_normpq` on the X and Y coordinates). The hint provided by these models is that copy tasks highlight the spatial-temporal coordination problems of AD patients.

7. Conclusions and Future Work

In this article, we have proposed using vectorial genetic programming (VE_GP) to solve the classification problem of recognizing the handwriting of people affected by Alzheimer's (AD). More specifically, we have used data extracted from the handwriting of AD patients and a control group while performing 25 tasks. Thanks to its ability to deal with time series directly into the evolved model, VE_GP has revealed a very appropriate method for this type of problem, paving the way for its practical use to support the early diagnosis of AD.

Table 5: best models according to the test accuracy.

Task #	Best model
2	$Z/0.7595 - \texttt{V\_duration0}(Z) - \texttt{V\_min}_{2219,8389}(\texttt{V\_duration0}(Z) + Z) + Z - \texttt{V\_length}(Z)$
3	$\texttt{V\_max}_{1952,6786}\Big(\texttt{V\_sum}_{6183,6309}\Big(\texttt{V\_sum}_{5037,6816}(Y)\Big) - X\Big)$
6	$Y - \texttt{V\_norm}_{1439,2375}(Y) - Z$
9	$\texttt{V\_sum}_{1208,8223}\Big(\texttt{V\_sum}_{1388,3126}\Big(Z + 1 - \texttt{V\_length}(Z)\Big) - \texttt{V\_length}\Big(\texttt{V\_length}(X) - X - Y\Big) - 0.88526 + Y - \texttt{V\_sum}_{4936,7055}(X - 0.11474)\Big)$
10	$\texttt{V\_sum}_{1141,1165}\Big[Z - \texttt{V\_sum}_{1976,2986}(Y) - \texttt{V\_length}\Big(\texttt{V\_sum}_{2124,3061}\Big(\texttt{V\_sum}_{264,4844}(\texttt{V\_sum}_{1976,5528}(X))\Big)\Big)\Big]$
15	$\frac{X}{Z} - \texttt{V\_norm}_{2744,11842}(Z)$
16	$\texttt{V\_max}_{1076,1414}(X - Y - \texttt{V\_length}(Z - Y))$
17	$\texttt{V\_norm}_{1979,6693}(Y) + \texttt{V\_norm}_{8258,10818}\Big(\texttt{V\_norm}_{1979,7259}(Y) + 2X\Big) + \texttt{V\_norm}_{1979,9757}(Y) + \texttt{V\_norm}_{1979,13190}\Big(\texttt{V\_norm}_{1979,7259}(Y)\Big) - \texttt{V\_sum}_{5887,12482}(Y)$
21	$\texttt{V\_dist}_{1226,14806}(Z) - Z$

The experimental results confirmed that VE_GP: (i) did not produce overfitted models; (ii) achieved good performance in accuracy, specificity, and sensitivity; (iii) did not suffer from any genetic drift phenomena. As for the comparison findings, VE_GP outperformed the seven standard feature engineering approaches and the six deep learning approaches in terms of win/tie/loss over the 25 handwriting tasks analyzed. Finally, VE_GP has generated small and simple predictive models that can be understood by domain experts and provide them with useful insights.

Future work will focus on investigating several aspects. First, we will explore the possibility of selecting only some of the 25 tasks used and then combining the responses provided by the models trained on the selected tasks. Second, we will assess the performance of VE_GP models trained on several tasks at the same time. Third, we will extend the function set with problem-specific functions considering handwriting dynamics, e.g., velocity. Fourth, we will use regularization techniques such as soft

Table 6: Comparison results with feature engineering-based approaches.

Task #	VE_GP	DT	GNB	LDA	LR	LVQ	MLP	SVM
1	0.67	0.58 +	0.63 ≈	0.61 +	0.63 +	0.62 +	0.62 +	0.62 +
2	0.68	0.67 ≈	0.70 ≈	0.71 ≈	0.72 -	0.67 ≈	0.69 ≈	0.73 -
3	0.67	0.70 ≈	0.69 ≈	0.70 ≈	0.70 ≈	0.67 ≈	0.66 ≈	0.69 ≈
4	0.63	0.64 ≈	0.70 -	0.65 ≈	0.68 -	0.67 -	0.63 ≈	0.70 -
5	0.67	0.66 ≈	0.73 -	0.68 ≈	0.69 ≈	0.66 ≈	0.66 ≈	0.66 ≈
6	0.68	0.71 ≈	0.71 ≈	0.68 ≈	0.73 -	0.69 ≈	0.69 ≈	0.70 ≈
7	0.77	0.69 +	0.72 +	0.75 ≈	0.75 ≈	0.71 +	0.73 +	0.76 ≈
8	0.74	0.67 +	0.72 ≈	0.73 ≈	0.75 ≈	0.74 ≈	0.70 ≈	0.76 ≈
9	0.77	0.68 +	0.73 +	0.72 +	0.72 ≈	0.69 +	0.70 +	0.74 ≈
10	0.78	0.60 +	0.67 +	0.66 +	0.69 +	0.65 +	0.64 +	0.68 +
11	0.77	0.62 +	0.62 +	0.62 +	0.65 +	0.61 +	0.63 +	0.63 +
12	0.75	0.63 +	0.59 +	0.65 +	0.63 +	0.62 +	0.60 +	0.61 +
13	0.73	0.63 +	0.66 +	0.67 +	0.68 +	0.65 +	0.66 +	0.68 +
14	0.65	0.67 ≈	0.59 +	0.62 +	0.64 ≈	0.68 ≈	0.72 -	0.61 ≈
15	0.74	0.74 ≈	0.71 ≈	0.68 +	0.74 ≈	0.71 ≈	0.69 ≈	0.71 ≈
16	0.70	0.75 -	0.73 ≈	0.69 ≈	0.74 ≈	0.73 ≈	0.73 ≈	0.75 -
17	0.73	0.71 ≈	0.70 ≈	0.71 ≈	0.72 ≈	0.73 ≈	0.72 ≈	0.71 ≈
18	0.81	0.61 +	0.57 +	0.69 +	0.69 +	0.68 +	0.72 +	0.70 +
19	0.67	0.69 ≈	0.55 +	0.72 -	0.71 -	0.72 -	0.69 ≈	0.72 -
20	0.67	0.66 ≈	0.65 ≈	0.68 ≈	0.66 ≈	0.68 ≈	0.66 ≈	0.67 ≈
21	0.65	0.66 ≈	0.67 ≈	0.72 -	0.73 -	0.70 ≈	0.72 -	0.71 -
22	0.73	0.69 ≈	0.67 +	0.70 ≈	0.67 ≈	0.70 ≈	0.67 ≈	0.69 ≈
23	0.80	0.82 ≈	0.69 +	0.70 +	0.74 +	0.73 +	0.71 +	0.74 +
24	0.71	0.67 ≈	0.67 +	0.68 ≈	0.71 ≈	0.73 ≈	0.69 ≈	0.69 ≈
25	0.69	0.69 ≈	0.68 ≈	0.71 ≈	0.72 ≈	0.68 ≈	0.69 ≈	0.70 ≈
win/tie/loss	–	9/15/1	12/11/2	10/13/2	7/13/5	9/14/2	9/14/2	7/13/5

target and functional complexity reduction [41] to improve the generalization ability of the trees generated by VE_GP. Finally, we will test the effectiveness of VE_GP in dealing with different types of time series data, e.g., those from wearable sensors. This data type is currently largely available and needs effective machine learning techniques to be processed.

Acknowledgments. This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152/2020 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS.

This work was partially supported by EU in NextGenerationEU plan through MUR Decree n. 1051 23.06.2022 "PNRR Missione 4 Componente 2 Investimento 1.5" -

Table 7: Comparison results with deep learning approaches.

Task #	VE_GP	On-paper			In-air & on-paper		
		IV3	R50	V19	IV3	R50	V19
1	0.67	0.52 +	0.59 +	0.56 +	0.55 +	0.55 +	0.55 +
2	0.68	0.61 +	0.59 +	0.68 +	0.66 ≈	0.64 +	0.70 -
3	0.67	0.55 +	0.61 +	0.58 +	0.62 +	0.63 +	0.62 +
4	0.63	0.73 -	0.70 -	0.69 -	0.71 -	0.66 -	0.72 -
5	0.67	0.68 ≈	0.64 +	0.67 ≈	0.67 ≈	0.65 ≈	0.71 -
6	0.68	0.63 +	0.57 +	0.62 +	0.70 -	0.66 ≈	0.73 -
7	0.77	0.65 +	0.58 +	0.59 +	0.65 +	0.64 +	0.72 +
8	0.74	0.65 +	0.67 +	0.68 +	0.67 +	0.72 ≈	0.68 +
9	0.77	0.62 +	0.62 +	0.69 +	0.71 +	0.62 +	0.67 +
10	0.78	0.65 +	0.60 +	0.68 +	0.62 +	0.72 +	0.65 +
11	0.77	0.60 +	0.65 +	0.60 +	0.67 +	0.70 +	0.61 +
12	0.75	0.61 +	0.63 +	0.60 +	0.61 +	0.55 +	0.57 +
13	0.73	0.63 +	0.59 +	0.63 +	0.60 +	0.56 +	0.65 +
14	0.65	0.66 ≈	0.58 +	0.64 ≈	0.64 ≈	0.66 ≈	0.65 ≈
15	0.74	0.62 +	0.58 +	0.61 +	0.64 +	0.59 +	0.67 +
16	0.70	0.59 +	0.60 +	0.64 +	0.67 +	0.64 +	0.72 -
17	0.73	0.58 +	0.53 +	0.67 +	0.65 +	0.63 +	0.70 +
18	0.81	0.62 +	0.56 +	0.59 +	0.57 +	0.53 +	0.62 +
19	0.67	0.68 ≈	0.61 +	0.70 -	0.68 ≈	0.61 +	0.69 -
20	0.67	0.55 +	0.57 +	0.70 -	0.70 -	0.60 +	0.68 ≈
21	0.65	0.68 -	0.66 -	0.65 ≈	0.67 -	0.59 +	0.67 -
22	0.73	0.49 +	0.53 +	0.57 +	0.62 +	0.61 +	0.73 +
23	0.80	0.56 +	0.51 +	0.63 +	0.70 +	0.62 +	0.66 +
24	0.71	0.61 +	0.68 +	0.67 +	0.65 +	0.64 +	0.66 +
25	0.69	0.62 +	0.60 +	0.67 +	0.64 +	0.62 +	0.72 -
win/tie/loss	–	20/3/2	23/0/2	19/3/3	17/4/4	20/4/1	15/2/8

CUP H33C22000420001.

References

[1] J. Garre-Olmo, M. Faundez-Zanuy, K. L. de Ipiña, L. Calvo-Perxas, O. Turro-Garriga, Kinematic and pressure features of handwriting and drawing: Preliminary results between patients with mild cognitive impairment, Alzheimer disease and healthy controls, Curr Alzheimer Res 14 (2017) 1–9.

[2] N. D. Cilia, C. De Stefano, F. Fontanella, A. Scotto Di Freca, Using genetic algorithms for the prediction of cognitive impairments, in: P. A. Castillo, J. L.

Jiménez Laredo, F. Fernández de Vega (Eds.), Applications of Evolutionary Computation, Springer International Publishing, Cham, 2020, pp. 479–493.

[3] M. Diaz, M. A. Ferrer, D. Impedovo, G. Pirlo, G. Vessio, Dynamically enhanced static handwriting representation for Parkinson's disease detection, Pattern Recognition Letters 128 (204-210) (2019).

[4] D. Impedovo, G. Pirlo, Dynamic handwriting analysis for the assessment of neurodegenerative diseases: a pattern recognition perspective, IEEE Reviews in Biomedical Engineering (2018) 1–13.

[5] T. Dokeroglu, A. Deniz, H. E. Kiziloz, A comprehensive survey on recent metaheuristics for feature selection, Neurocomputing 494 (2022) 269–296.

[6] N. D. Cilia, G. De Gregorio, C. De Stefano, F. Fontanella, A. Marcelli, A. Parziale, Diagnosing Alzheimer's disease from on-line handwriting: A novel dataset and performance benchmarking, Engineering Applications of Artificial Intelligence 111 (2022) 104822.

[7] I. Azzali, L. Vanneschi, S. Silva, I. Bakurov, M. Giacobini, A vectorial approach to genetic programming, in: Genetic Programming, EuroGP 2019, Lecture Notes in Computer Science, 2019, p. 213–227.

[8] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.

[9] R. Poli, W. B. Langdon, N. F. McPhee, A field guide to genetic programming, Lulu Press, 2008.

[10] I. Azzali, L. Vanneschi, A. Mosca, L. Bertolotti, M. Giacobini, Towards the use of genetic programming in the ecological modelling of mosquito population dynamics, in: Genet Program Evolvable Mach, 21, 2020, p. 629–64.

[11] I. Azzali, L. Vanneschi, I. Bakurov, S. Silva, M. Ivaldi, M. Giacobini, Towards the use of vector based GP to predict physiological time series, Applied Soft Computing (2020) 89.

[12] O. Asan, A. E. Bayrak, A. Choudhury, Artificial intelligence and human trust in healthcare: Focus on clinicians., in: Journal of medical Internet research, 2020, p. 22(6).

[13] I. Azzali, N. D. Cilia, C. De Stefano, F. Fontanella, M. Giacobini, L. Vanneschi, Vectorial GP for Alzheimer's disease prediction through handwriting analysis, in: J. L. Jiménez Laredo, J. I. Hidalgo, K. O. Babaagba (Eds.), Applications of Evolutionary Computation, Springer International Publishing, Cham, 2022, pp. 517–530.

[14] N. Cilia, C. De Stefano, F. Fontanella, M. Molinara, A. Scotto di Freca, Handwriting analysis to support Alzheimer's disease diagnosis: A preliminary study, Lecture Notes in Computer Science - ICIAP Proceedings(2019) (2019).

[15] N. Cilia, C. De Stefano, F. Fontanella, A. Scotto Di Freca, An experimental protocol to support cognitive impairment diagnosis by using handwriting analysis, in: Procedia Computer Science, Proceeding of The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH), Elsevier, 2019, pp. 1–9.

[16] N. D. Cilia, T. D'Alessandro, C. De Stefano, F. Fontanella, M. Molinara, From online handwriting to synthetic images for Alzheimer's disease detection using a deep transfer learning approach, IEEE Journal of Biomedical and Health Informatics 25 (12) (2021) 4243–4254.

[17] A. Petrowski, S. Ben-Hamida, Evolutionary algorithms, in: Wiley-ISTE., 2020, pp. 1–32.

[18] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, E. L. Michalkiewicz, A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets, Artificial Intelligence in Medicine 30 (1) (2004) 27 – 48.

[19] M. Castelli, L. Vanneschi, L. Manzoni, A. Popovič, Semantic genetic program-

ming for fast and accurate data knowledge discovery, Swarm and Evolutionary Computation 26 (2016) 1–7.

[20] I. Bakurov, M. Castelli, L. Vanneschi, M. J. Freitas, Supporting medical decisions for treating rare diseases through genetic programming, in: P. Kaufmann, P. A. Castillo (Eds.), Applications of Evolutionary Computation, Springer International Publishing, Cham, 2019, pp. 187–203.

[21] A. Parziale, R. Senatore, A. Della Cioppa, A. Marcelli, Cartesian genetic programming for diagnosis of Parkinson disease through handwriting analysis: Performance vs. interpretability issues, Artificial Intelligence in Medicine 111 (2021) 101984.

[22] A. Ghaheri, S. Shoar, M. Naderan, S. S. Hoseini, The applications of genetic algorithms in medicine, Oman medical journal 30 (6) (2015) 406–416.

[23] O. Valenzuela, X. Jiang, A. Carrillo, I. Rojas, Multi-objective genetic algorithms to find most relevant volumes of the brain related to Alzheimer's disease and mild cognitive impairment, International Journal of Neural Systems 28 (09) (2018).

[24] P. Johnson, L. Vandewater, W. J. Wilson, P. Maruff, G. Savage, P. Graham, L. S. Macaulay, K. A. Ellis, C. Szoeke, R. N. Martins, C. Rowe, C. L. Masters, D. Ames, P. Zhang, Genetic algorithm with logistic regression for prediction of progression to Alzheimer's disease, BMC Bioinformatics 15 (S11) (2014).

[25] Y. Zhang, P. I. Rockett, Feature Extraction Using Multi-Objective Genetic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, Ch. 4, pp. 75–99.

[26] O. Oechsle, A. F. Clark, Feature extraction and classification by genetic programming, in: A. Gasteratos, M. Vincze, J. K. Tsotsos (Eds.), Computer Vision Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 131–140.

[27] S. Ruberto, V. Terragni, J. H. Moore, Image feature learning with genetic programming, in: T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Em-

merich, H. Trautmann (Eds.), Parallel Problem Solving from Nature – PPSN XVI, Springer International Publishing, Cham, 2020, pp. 63–78.

[28] Q. Fan, Y. Bi, B. Xue, M. Zhang, Genetic programming for feature extraction and construction in image classification, Applied Soft Computing 118 (2022) 108509.

[29] E. Z-Flores, L. Trujillo, P. Legrand, F. Faïta-Aïnseba, Eeg feature extraction using genetic programming for the classification of mental states, Algorithms 13 (9) (2020).

[30] Q. U. Ain, H. Al-Sahaf, B. Xue, M. Zhang, A genetic programming approach to feature construction for ensemble learning in skin cancer detection, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, GECCO '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1186–1194.

[31] M. Z. Ying Bi, Bing Xue, Genetic Programming for Image Classification, An Automated Approach to Feature Learning, Springer, Adaptation, Learning, and Optimization book series, 2021.

[32] B. Peng, S. Wan, Y. Bi, B. Xue, M. Zhang, Automatic feature extraction and construction using genetic programming for rotating machinery fault diagnosis, IEEE Transactions on Cybernetics 51 (10) (2021) 4909–4923.

[33] B. Tran, B. Xue, M. Zhang, Genetic programming for multiple-feature construction on high-dimensional classification, Pattern Recognition 93 (2019) 404–417.

[34] J. E. Batista, A. I. R. Cabral, M. J. P. Vasconcelos, L. Vanneschi, S. Silva, Improving land cover classification using genetic programming for feature construction, Remote Sensing 13 (9) (2021).

[35] C. Vatamanu, D. Gavrilut, R. Benchea, H. Luchian, Feature extraction using genetic programming with applications in malware detection, in: 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2015, pp. 224–231.

[36] L. H. Rasmussen J, Why we need early diagnosis, in: Degener Neurol Neuromuscul Dis. 9, Elsevier, 2019, pp. 123–130.

[37] M. Tanveer, B. Richhariya, R. U. Khan, A. H. Rashid, P. Khanna, M. Prasad, C. T. Lin, Machine learning techniques for the diagnosis of Alzheimer's disease: A review, ACM Trans. Multimedia Comput. Commun. Appl. 16 (1s) (apr 2020).

[38] H. Jabeen, A. Baig, Review of classification using genetic programming, in: International Journal of Engineering Science and Technology, 2010, pp. 94–103.

[39] S. Silva, J. Almeida, Gplab-a genetic programming toolbox for matlab, in: Proceedings of the Nordic MATLAB Conference, 2008, pp. 1–6.

[40] N. D. Cilia, T. D'Alessandro, C. D. Stefano, F. Fontanella, Deep transfer learning algorithms applied to synthetic drawing images as a tool for supporting Alzheimer's disease prediction, Mach. Vis. Appl. 33 (3) (2022) 49.

[41] L. Vanneschi, M. Castelli, Soft target and functional complexity reduction: A hybrid regularization method for genetic programming, Expert Systems with Applications 177 (2021) 114929.