Prediction of Covid-19 cases using constructed features by Grammatical Evolution

Ioannis G. Tsoulos[1]*, Alexandros Tzallas[1], Dimitrios Tsalikakis[2]

[1]Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece
[2]University of Western Macedonia, Department of Engineering Informatics and Telecommunications, Greece

Abstract

A widely used method that constructs features with the incorporation of the so - called Grammatical Evolution is proposed here to predict the Covid-19 cases as well as the mortality rate. The method creates new artificial features from the original ones using a Genetic algorithm and is guided by a BNF grammar. After the artificial features are generated, the original data set is modified based on these features and an artificial neural network is applied to the modified data and the results are reported. From the comparative experiments done, it was clear that feature construction has an advantage over other machine learning methods for predicting pandemic elements.

Keywords: Genetic algorithms; feature construction; Covid-19 pandemic; stochastic methods; Grammatical Evolution

1 Introduction

The world changed in December 2019, when the first reports emerged of a mysterious infection in the Chinese province, Wuhan. Subsequently, the WHO (World Health Organization) declared a new global pandemic on March 11, 2020. The name given to the new coronavirus was SARS-CoV-2 (Severe Acute Respiratory Syndrome) and the name given to the corresponding disease was Covid-19. As Andersen mentioned [1], the virus's origin is natural selection in an animal host before the zoonotic transfer or natural selection in humans following the zoonotic transfer. At the time of writing, the number of diagnosed cases of Covid-19 was about 650.000.000 and the number of people who had died was over 6.500.000. So the fatality rate was about 1%, although this rate may be lower since many people have fallen ill without undergoing any diagnostic test.

*Corresponding author. Email: itsoulos@uoi.gr

Due to the seriousness of the topic, since the beginning of the pandemic until now, a multitude of publications have appeared in the relevant literature on this topic and many of them use computational techniques to predict the course of the disease. For example, Wang [2] used the Patient Information Based Algorithm (PIBA), that uses real - time data collected from patients in Wuhan. With this data, he wrote an algorithm to forecast the death rates. Also, Tomar and Gupta [3], used Long Short-Term Memory (LSTM) to predict the number of COVID-19 cases. Zhang et al[4] used a Poisson model to analyze the Covid-19 cases in Canada,France,Germany, Italy, UK and USA. Pinter et al. [5] used an ANFIS system and a multilayered perceptron- imperialist competitive algorithm (MLP-ICA) to predict the mortality rate for Hungary. Smith et al. [6], used machine learning techniques for a dataset made of blood samples taken from Covid-19 patients from a hospital in the region of Wuhan, in order to estimate the mortality rate. Additionally, papers [7, 8] using machine learning techniques to recognize faces behind masks has emerged in recent years.

The current work utilizes a feature construction method initially presented in [9], to predict the Covid-19 cases and deaths for a series of randomly selected countries. The method is based on Grammatical Evolution [10] and creates artificial features from the original ones without a priory knowledge of the structure of the problem. The method has been used with success in a series of scientific fields such as Spam identification [11], Fetal heart classification [12], epileptic oscillations [9] etc. Also, recently a software that implements the feature construction method using modern programming approaches has been published [14]. In the case of the Covid-19 data, the only feature is the recording time in ascending form and the output is the number of cases or deaths for that recording. The proposed method creates 1-3 artificial features from the recording time and, subsequently an artificial neural network [15, 16] is trained on the modified data. The experimental results are compared against other methods used to train neural networks and the proposed technique appears to significantly outperform the others in terms of accuracy.

The rest of this article is organized as follows: in section 2 the main aspects of Grammatical Evolution and the steps of the proposed technique are outlined in detail, in section 3 the experimental results are outlined and finally in section 4 some conclusions and guidelines for the extension of the proposed technique are listed.

2 Method description

The proposed technique is divided into two phases. During the first phase, artificial new features are created from the original ones using Grammatical Evolution and a Genetic Algorithm. The new features are evaluated using a Radial Basis Function (RBF) network [17] with H processing units. The RBF network is selected as the evaluator because the training procedure of RBF is much faster than those of artificial neural networks. Subsequently, in the second phase, the original dataset is transformed using the best located features and a

Genetic Algorithm is used to train a neural network on the modified dataset.

2.1 Grammatical Evolution

Grammatical evolution is an evolutionary approach, where the chromosomes are random numbers standing for production rules of a given BNF (Backus-Naur form) grammar [18]. The Grammatical Evolution was initially used for regression [19, 20] and to solve trigonometric identities [21], but it has been applied in a variety of scientific fields, such as automatic composition of music [22], neural network construction [23, 24], automatic constant creation [25], evolution of video games [26, 27], energy demand estimation [28], combinatorial optimization [29], cryptography [30] etc. The BNF grammar is defined as the set $G = (N, T, S, P)$ with:

- N is the set of the non - terminal symbols, used to produce series of terminal symbols through production rules.
- T is the set of terminal symbols of the grammar. For example, terminal symbols could be the digits in an arithmetic expression of the operators.
- S is the starting non - terminal symbol of the grammar. The production of a valid expression initiates from this symbol.
- P is the set of production rules. Every rule is in the form $A \to a$ or $A \to aB,\ A, B \in N,\ a \in T$.

The original grammar is expanded in Grammatical Evolution, with the addition of a sequence number for each production rule. The chromosomes in Grammatical Evolution are integer numbers representing sequence numbers of production rules. Subsequently, the production procedure starts from the start symbol of the grammar and produces valid programs by replacing non - terminal symbols with the right hand of the selected production rule. The selection of the rule has two steps:

1. Retrieve the next element from the given chromosome and denote it as V.
2. The next production rule R is calculated by

   $$R = V \text{ MOD } K$$

   where K is the total number of production rules for the current non - terminal symbol.

In the current work the extend BNF grammar is illustrated in Figure 1. The non - terminal symbols are enclosed in the symbols $< >$ and the number N denotes the number of original features. For the Covid-19 case the N is considered as 1. An example of producing a valid expression is shown in the Table 1. The chromosome is $x = [9, 8, 6, 4, 16, 10, 17, 23, 8, 14]$ and $N = 3$. The valid expression created finally is $f(x) = x_2 + \cos(x_3)$.

Considering the above mapping procedure, the steps to produce N_f artificial features for a given chromosome g are:

Table 1: Steps to produce a valid expression from the BNF grammar.

String	Chromosome	Operation
<expr>	9,8,6,4,16,10,17,23,8,14	$9 \mod 3 = 0$
(<expr><op><expr>)	8,6,4,16,10,17,23,8,14	$8 \mod 3 = 2$
(<terminal><op><expr>)	6,4,16,10,17,23,8,14	$6 \mod 2 = 0$
(<xlist><op><expr>)	4,16,10,17,23,8,14	$4 \mod 3 = 1$
(x2<op><expr>)	16,10,17,23,8,14	$16 \mod 4 = 0$
(x2+<expr>)	10,17,23,8,14	$10 \mod 3 = 1$
(x2+<func>(<expr>))	17,23,8,14	$17 \mod 4 = 1$
(x2+cos(<expr>))	23,8,14	$23 \mod 2 = 1$
(x2+cos(<terminal>))	8,14	$8 \mod 2 = 0$
(x2+cos(<xlist>))	14	$14 \mod 3 = 2$
(x2+cos(x3))		

1. Divide the into N_f parts. Each part $g_i,\ i = 1,..,N_f$ will construct a separate feature.

2. A feature t_i is constructed for every g_i using the grammar given in Figure 1

3. Construct a mapping function

$$\mathbf{FC}(\overrightarrow{x}, g) = \left(t_1\left(\overrightarrow{x}, g_1\right), t_2\left(\overrightarrow{x}, g_2\right), \ldots, t_{N_f}\left(\overrightarrow{x}, g_{N_f}\right)\right) \qquad (1)$$

with $\overrightarrow{x}$ being the pattern from the original set.

2.2 Feature creation step

In this step a genetic algorithm in conjunction with the mapping procedure of subsection 2.1 is used to produce artificial features. The fitness function of the genetic algorithm is the training error of an RBF neural network with H processing units. The steps are the following:

1. **Initialization step**

   (a) **Set** iter=0, the current number of generations.

   (b) **Consider** the set $\text{TR} = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \ldots, (\overrightarrow{x_M}, y_M)\}$, the original training set.

   (c) **Set** N_c the number of chromosomes in the genetic population.

   (d) **Set** N_f the number of constructed features.

   (e) **Initialize** randomly in range $[0, 255]$ every element of each chromosome.

   (f) **Set** N_g as the maximum number of generations.

Figure 1: BNF grammar of the proposed method.

```
S::=<expr>   (0)
<expr> ::=  (<expr> <op> <expr>)  (0)
           | <func> ( <expr> )    (1)
           |<terminal>            (2)
<op> ::=     +      (0)
           | -      (1)
           | *      (2)
           | /      (3)
<func> ::=   sin  (0)
           | cos  (1)
           |exp   (2)
           |log   (3)
<terminal>::=<xlist>                (0)
           |<digitlist>.<digitlist> (1)
<xlist>::=x1    (0)
           | x2 (1)
           .........
           | xN (N)
<digitlist>::=<digit>                  (0)
           | <digit><digit>            (1)
           | <digit><digit><digit>     (2)
<digit>  ::= 0 (0)
           | 1 (1)
           | 2 (2)
           | 3 (3)
           | 4 (4)
           | 5 (5)
           | 6 (6)
           | 7 (7)
           | 8 (8)
           | 9 (9)
```

(g) **Set** $p_s \in [0,1]$ as the selection rate.

(h) **Set** as $p_m \in [0,1]$ the mutation rate.

2. **Termination check step. If** iter$>=N_g$ terminate.

3. **Calculate** the fitness f_i of every chromosome g_i with the following procedure:

   (a) **Create** N_f features using the mapping procedure of subsection 2.1.

   (b) **Construct** the mapped training set

   $$\mathrm{TN} = \left\{ \left(\mathrm{FC}\left(\overrightarrow{x_1}, g_i\right), y_1\right), \left(\mathrm{FC}\left(\overrightarrow{x_2}, g_i\right), y_2\right), \ldots, \left(\mathrm{FC}\left(\overrightarrow{x_M}, g_i\right), y_M\right) \right\} \tag{2}$$

   (c) **Train** an RBF neural network C with H processing units on the new set TN and obtain the following train error

   $$E_i = \sum_{j=1}^{M} \left(C\left(\mathrm{FC}\left(\overrightarrow{x_j}, Z_i\right)\right) - y_j\right)^2 \tag{3}$$

   (d) **Set** $f_i = E_i$

4. **Genetic Operators**

   (a) **Selection procedure:** The chromosomes are sorted according to their fitness. The best $p_s \times N_c$ are copied intact to the next generation. The genetic operations of crossover and mutation are applied to rest of the chromosomes.

   (b) **Crossover procedure:** During this process $(1-p_s) \times N_c$ offsprings will be created. For every couple of produced offsrpings, two parents (z,w) are selected using the well - known procedure of tournament selection. For every pair (z,w) of parents, two offsprings $\tilde{z}$ and $\tilde{w}$ are produced through one point crossover. An example of one point crossover is shown in Figure .

   (c) **Mutation procedure:** For each element of every chromosome a random number $r \in [0,1]$ is produced. Subsequently we change randomly the corresponding element if $r \le p_m$.

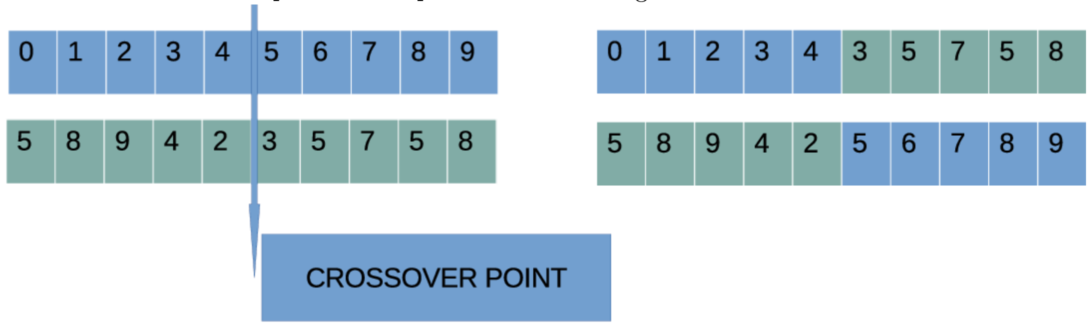
5. **Set** iter=iter+1 and **goto** Step 2.

2.3 Feature evaluation step

During this step, the best chromosome g_b of the first step is obtained and used to create the modified train set

$$\mathrm{TN} = \left\{ \left(\mathrm{FC}\left(\overrightarrow{x_1}, g_b\right), y_1\right), \left(\mathrm{FC}\left(\overrightarrow{x_2}, g_b\right), y_2\right), \ldots, \left(\mathrm{FC}\left(\overrightarrow{x_M}, g_b\right), y_M\right) \right\}$$

Figure 2: An example of one point crossover. A randomly selected point is chosen and the two subparts of the parents are exchanged.



Afterwards a genetic algorithm is used to train an artificial neural network with H hidden nodes for this dataset. The neural network used here is defined as a function $N(x, w)$, where w is the weight vector to be estimated through the genetic algorithm after the minimization of the following training error:

$$E(N(x,w)) = \sum_{i=1}^{M} \left(N\left(\mathrm{FC}\left(x_i, g_b\right), w\right) - y_i\right)^2 \tag{4}$$

The neural network has a form also used in [31]. If the neural network has one processing level, every output of each hidden node is in the form:

$$o_i(x) = \sigma\left(p_i^T x + \theta_i\right), \tag{5}$$

where p_i is the weight vector and θ_i is considered as the bias for output i. The function $\sigma(x)$ is the well - known sigmoid function given by:

$$\sigma(x) = \frac{1}{1+\exp(-x)} \tag{6}$$

For a neural network with H hidden nodes the final output is given by:

$$N(x) = \sum_{i=1}^{H} v_i o_i(x), \tag{7}$$

where v_i is the weight for hidden node i. Hence, if we use one weight w for hidden nodes and biases the following form could be used for he neural network:

$$N\left(\overrightarrow{x}, \overrightarrow{w}\right) = \sum_{i=1}^{H} w_{(d+2)i-(d+1)}\sigma\left(\sum_{j=1}^{d} x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i}\right) \tag{8}$$

The value d the size of input vector $\overrightarrow{x}$. Also, the total number of parameters of vector $\overrightarrow{w}$ is $(d+2) \times H$. The Genetic Algorithm used here, is a global optimization technique[32, 33], that has been applied with success in many problems such as electromagnetic [34], combinatorial problems [35], design of water distribution networks [36] etc. Also, it has been used to train artificial networks in some works from the relevant literature [37, 38, 39]. The steps of the genetic algorithm used in the second step of the proposed method are the following:

1. **Initialization** step.
   (a) **Set** as N_c the number of chromosomes that will participate.
   (b) **Set** as N_g,the maximum number of allowed generations.
   (c) **Set** as p_m, the mutation rate.
   (d) **Set** as p_s, the selection rate.
   (e) **Set** ϵ a small positive number, i.e $\epsilon = 10^{-8}$.
   (f) **Initialize** randomly the chromosomes $g_i,\ i = 1, ..., N_c$ For the case of neural networks every element of each chromosome is considered as a double precision number. Also, the size of each chromosome is $(d+2) \times H$.
   (g) **Set** iter=0
2. **Check** for termination.
   (a) **Obtain** the best fitness
   $$f^* = \min_{i \in [0 \ldots N_c]} f_i$$
   (b) **Terminate** if iter $\geq N_g$ OR $f^* \leq \epsilon$
3. **Calculate** fitness.
   (a) **For** $i = 1, \ldots, N_c$ **do**
      i. **Create** a neural network using as parameter vector the chromosome g_i.
      ii. **Calculate** the fitness value $f_i = f(g_i)$ using the equation 4.
   (b) **EndFor**
4. **Application** of genetic operators.
   (a) **Selection** operation. During selection, the chromosomes are classified according to their fitness. The first $p_s \times N_c$ are copied without changes to the next generation of the population. The rest will be replaced by chromosomes that will be produced at the crossover.

(b) **Crossover** operation. In the crossover operation, $p_s \times N_c$ chromosomes are produced. For every couple of produced offsrpings, two parents (z,w) are selected using tournament selection. For every pair (z,w) of parents, two offsprings $\tilde{z}$ and $\tilde{w}$ are produced according to the following equations:

$$\begin{aligned}\tilde{z_i} &= a_i z_i + (1-a_i)\, w_i \\ \tilde{w_i} &= a_i w_i + (1-a_i)\, z_i\end{aligned} \tag{9}$$

where a_i is a random number with the property $a_i \in [-0.5, 1.5]$ [40].

(c) **Mutation** operation. For each element of every chromosome a random number $r \in [0,1]$ is produced and the element is altered if $r \leq p_m$

(d) **Set** iter=iter+1

5. **Goto** step 2.

3 Experimental results

The data was freely available from the URL `https://ourworldindata.org/explorers/coronavirus-data-explorer` and was downloaded for the start of pandemic until September 10, 2022. To enable the machine learning models to better fit the data, the following fair normalization took place in every dataset

1. The number of cases was divided by 10^5.

2. The number of deaths was divided by 10^3.

The following countries were selected for testing: Algeria, Argentina, Australia, Brazil, Bulgaria, Canada, Germany, Greece, Egypt and Japan. For every country, two distinct datasets was obtained from the COVID-19 database, one dataset for the number of cases in every day of the pandemic and one dataset for the number of deaths in every day of the pandemic. Every run was executed 30 times for every dataset and averages were measured. The random number generator used was the drand48() of the C programming language. The parameters for the proposed method are listed in the Table 2. The results for the COVID-19 cases are shown in the Table 3 and the results for the COVID-19 deaths are illustrated in the Table 4. The columns in both tables have the following meaning:

1. The column COUNTRY contains the name of the country.

2. The column ADAM stands for the Adam optimization method [41] used to train a neural network with 10 processing nodes. The ADAM method is implemented in OptimLib and it is available from `https://github.com/kthohr/optim`.

Table 2: Experimental parameters.

PARAMETER	VALUE
N_c	500
N_g	200
H	10
p_s	0.10
p_m	0.05
ϵ	10^{-8}

3. The column MLPPSO represents the results using a neural network trained with the help of Particle Swarm Optimization method [42, 43]. The number of PSO particles was set to N_c and the maximum number of allowed iterations was set to N_g. The value for this parameters are shown in the Table 2. Also, a BFGS method [44] was applied to best particle of the swarm when the PSO finishes, in order to enhance the results.

4. The column MLPGEN stands for the results obtained by a neural network with ten processing units, that is trained using a Genetic algorithm [32, 33]. The parameters for this algorithm are listed in the Table 2. Also, the BFGS was applied to the best chromosome after the termination of the genetic algorithm.

5. The column FC1 stands for the results obtained by the proposed method with one constructed feature ($N_f = 1$).

6. The column FC2 stands for the results obtained by the proposed method with two constructed features ($N_f = 2$).

7. The column FC3 stands for the results obtained by the proposed method with three constructed features ($N_f = 3$).

Also, in both tables, an additional row was added to indicate the average error and is denoted by AVERAGE. This extra column is graphically plotted in Figures 3 and 4 for cases prediction and deaths prediction respectively.

From the execution of the above experiments, it is clear in principle that the efficiency of the methods depends to a great extent on the country in question. In some countries the test error is high and in others quite low. This is probably due to the different course of the number of cases and the mortality rate in each country separately. Also, the proposed method has higher accuracy than the other techniques even if only one artificial feature is created. In fact, in some cases of countries, the test error of the proposed technique is so low that it is almost zero. Using more than one feature appears to drastically reduce the error, although the reduction appears to be more significant between one and two features and less when a third constructed feature is added.

Table 3: Total test error for the prediction of COVID-19 cases.

COUNTRY	ADAM	MLPPSO	MLPGEN	FC1	FC2	FC3
Algeria	0.31	0.08	0.286	0.0025	0.0006	0.0002
Argentina	178.60	21.03	69.20	3.21	0.81	0.91
Australia	144.46	20.33	30.96	1.52	0.37	0.34
Brazil	198.26	81.94	75.79	11.93	8.97	6.50
Bulgaria	4.01	1.29	2.67	0.037	0.0098	0.01
Canada	27.68	6.12	20.65	0.33	0.20	0.15
Germany	274.34	135.15	92.50	25.05	25.11	14.36
Greece	25.07	12.08	9.25	3.62	1.60	2.75
Egypt	0.24	0.13	0.44	0.005	0.029	0.003
Japan	271.11	95.56	75.76	8.92	2.15	1.82
AVERAGE	**112.41**	**37.37**	**37.75**	**5.46**	**3.92**	**2.68**

Table 4: Predicting deaths per country.

COUNTRY	ADAM	PSOGEN	MLPGEN	FC1	FC2	FC3
Algeria	0.40	0.15	0.66	0.009	0.002	0.002
Argentina	18.50	19.70	25.81	2.03	1.35	1.70
Australia	1.69	0.44	1.00	0.05	0.03	0.03
Brazil	416.54	282.46	230.52	52.42	16.75	16.57
Bulgaria	3.80	2.76	16.50	0.15	0.10	0.08
Canada	9.12	4.78	18.12	0.27	0.15	0.15
Germany	116.56	37.06	40.49	3.03	2.07	4.17
Greece	3.29	2.97	2.86	0.74	0.08	0.07
Egypt	2.57	0.79	8.88	0.07	0.03	0.02
Japan	43.10	22.07	13.74	0.32	0.12	0.13
AVERAGE	**61.56**	**37.32**	**35.86**	**5.91**	**2.07**	**2.29**

Figure 3: Graphical representation of average error for predicting COVID-19 cases.

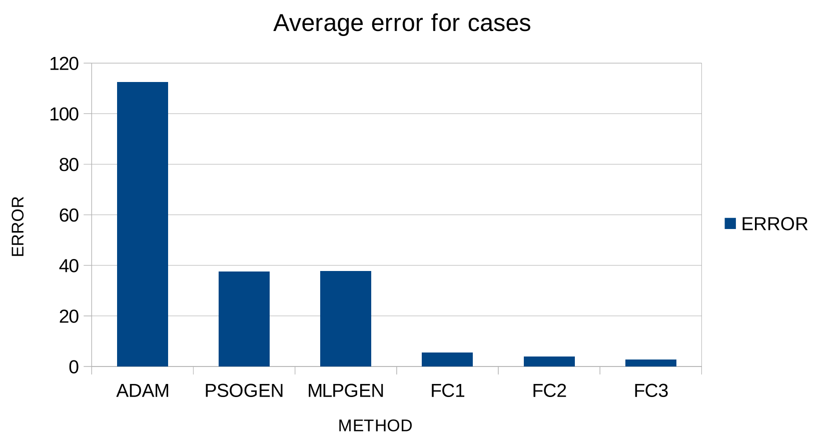
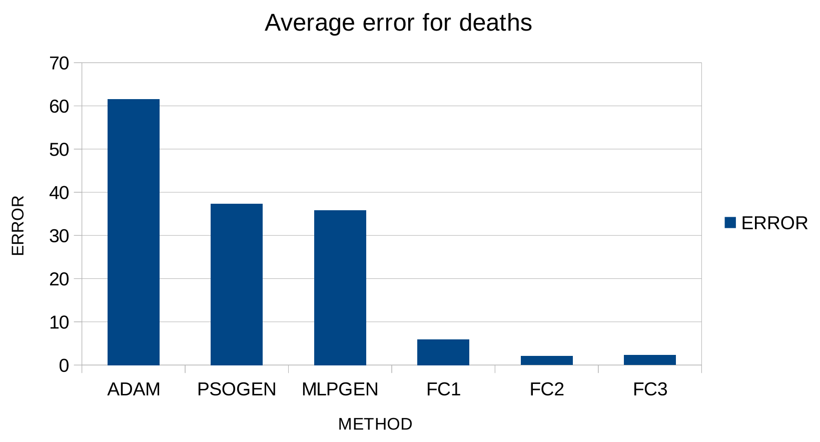


Figure 4: Graphical representation of average error for predicting COVID-19 deaths.



4 Conclusions

An automated artificial feature construction method was presented in this work to predict Covid-19 cases and deaths. The method is based on Grammatical Evolution through two phases. In the first phase, 1-3 features were constructed using Genetic Algorithm and Grammatical Evolution, and in the second phase, the new features were evaluated by an artificial neural network trained using a Genetic Algorithm. This procedure was applied to 10 randomly selected countries from all continents and appeared to be significantly superior to other techniques. Future improvements of the method may include:

1. Incorporation of more advanced stopping rules for the genetic algorithms of the two phases.
2. Usage of another machine learning models instead of the RBF network to evaluate the constructed features.
3. Usage of parallel techniques to speed up the feature creation process.
4. Use of the technique on data that will also contain demographic characteristics of each country, in order to establish whether there is a correlation of the rate of cases or mortality with any particular characteristic of some countries.

References

[1] K.G. Andersen, A. Rambaut, W.I. Lipkin, E.C. Holmes, R.F. Garry, The proximal origin of sars-cov-2. Nature medicine **26**, pp. 450-452, 2020.

[2] L.Wang, J.Li, S.Guo, N.Xie, L.Yao, Y.Cao, S.W. Day, S.C. Howard, J.C. Gra, T.Gu, et al. Real-time estimation and prediction of mortality caused by covid-19 with patient information based algorithm. Science of the Total Environment, page 138394, 2020.

[3] A.Tomar, N.Gupta, Prediction for the spread of covid- 19 in india and eectiveness of preventive measures. Science of The Total Environment, page 138762, 2020.

[4] X.Zhang, R.Ma, L.Wang, Predicting turning point, duration and attack rate of covid-19 outbreaks in major western countries. Chaos, Solitons and Fractals, page 109829, 2020.

[5] G.Pinter, I.Felde, A.Mosavi, P. Ghamisi, R.Gloaguen, Covid-19 pandemic prediction for hungary; a hybrid machine learning approach. Mathematics, 8(6), 2020.

[6] M.Smith, F. Alvarez, Identifying mortality factors from machine learning using shapley values a case of covid19. Expert Systems with Applications, 176:114832, 2021.

[7] M.Loey, G.Manogaran, M.H.N. Taha, N.E.M. Khalifa, A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic. Measurement, 167:108288, 2021.

[8] I.Q.Mundial, M.S. Ul Hassan, M.I.Tiwana, W.S. Qureshi, E.Alanazi, Towards facial recognition problem in covid-19 pandemic. In 2020 4rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM), pp. 210-214, 2020.

[9] D.Gavrilis, I.G. Tsoulos, E. Dermatas, Selecting and constructing features using grammatical evolution, Pattern Recognition Letters **29**, pp. 1358-1365, 2008.

[10] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Transactions on Evolutionary Computation **5**, pp. 349-358, 2001.

[11] D.Gavrilis, I.G. Tsoulos, E.Dermatas, Neural recognition and genetic features selection for robust detection of e-mail spam. In Hellenic Conference on Articial Intelligence, pp. 498-501. Springer, 2006.

[12] G.Georgoulas, D.Gavrilis, I.G. Tsoulos, C.Stylios, J.Bernardes, P.P. Groumpos, Novel approach for fetal heart rate classication introducing grammatical evolution. Biomedical Signal Processing and Control **2**, pp.69-79, 2007.

[13] O.Smart, I.G. Tsoulos, D.Gavrilis, G.Georgoulas, Grammatical evolution for features of epileptic oscillations in clinical intracranial electroencephalograms. Expert systems with applications **38**, pp.9991-9999, 2011.

[14] I.G. Tsoulos, QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution, Algorithms **15**, 295, 2022.

[15] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[16] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems 2, pp. 303-314, 1989.

[17] J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.

[18] J. W. Backus. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, 1959, pp.125-132.

[19] C. Ryan, J. Collins, M. O'Neill, Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1998. Lecture Notes in Computer Science, vol 1391. Springer, Berlin, Heidelberg, 1998.

[20] M. O'Neill, M., C. Ryan, Evolving Multi-line Compilable C Programs. In: Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1999. Lecture Notes in Computer Science, vol 1598. Springer, Berlin, Heidelberg, 1999.

[21] C. Ryan, M. O'Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.

[22] A.O. Puente, R. S. Alfonso, M. A. Moreno, Automatic composition of music by means of grammatical evolution, In: APL '02: Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications July 2002 Pages 148–155.

[23] Lídio Mauro Limade Campo, R. Célio Limã Oliveira,Mauro Roisenberg, Optimization of neural networks through grammatical evolution and a genetic algorithm, Expert Systems with Applications **56**, pp. 368-384, 2016.

[24] K. Soltanian, A. Ebnenasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, Evolutionary Computation **30**, pp 291–327, 2022.

[25] I. Dempsey, M.O' Neill, A. Brabazon, Constant creation in grammatical evolution, International Journal of Innovative Computing and Applications **1** , pp 23–38, 2007.

[26] E. Galván-López, J.M. Swafford, M. O'Neill, A. Brabazon, Evolving a Ms. PacMan Controller Using Grammatical Evolution. In: , et al. Applications of Evolutionary Computation. EvoApplications 2010. Lecture Notes in Computer Science, vol 6024. Springer, Berlin, Heidelberg, 2010.

[27] N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, 2012 IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 304-31.

[28] D. Martínez-Rodríguez, J. M. Colmenar, J. I. Hidalgo, R.J. Villanueva Micó, S. Salcedo-Sanz, Particle swarm grammatical evolution for energy demand estimation, Energy Science and Engineering **8**, pp. 1068-1079, 2020.

[29] N. R. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems, IEEE Transactions on Evolutionary Computation **17**, pp. 840-861, 2013.

[30] C. Ryan, M. Kshirsagar, G. Vaidya, G. et al. Design of a cryptographically secure pseudo random number generator with grammatical evolution. Sci Rep **12**, 8602, 2022.

[31] I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.

[32] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.

[33] Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.

[34] R.L. Haupt, An introduction to genetic algorithms for electromagnetics, Antennas and Propagation Magazine 37, pp. 7-15, 1995.

[35] J.J. Grefenstette, R. Gopal, B. J. Rosmaita, D. Van Gucht, Genetic Algorithms for the Traveling Salesman Problem, In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 160 - 168, Lawrence Erlbaum Associates, 1985.

[36] D. A. Savic, G. A. Walters, Genetic Algorithms for Least-Cost Design of Water Distribution Networks, Journal of Water Resources Planning and Management 123, pp. 67-77, 1997.

[37] F. H. F. Leung, H. K. Lam, S. H. Ling and P. K. S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks **14**, pp. 79-88, 2003.

[38] A. Sedki, D. Ouazar, E. El Mazoudi, Evolving neural network using real coded genetic algorithm for daily rainfall–runoff forecasting, Expert Systems with Applications **36**, pp. 4523-4527, 2009.

[39] A. Majdi, M. Beiki, Evolving neural network using a genetic algorithm for predicting the deformation modulus of rock masses, International Journal of Rock Mechanics and Mining Sciences **47**, pp. 246-253, 2010.

[40] P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.

[41] D. P. Kingma, J. L. Ba, ADAM: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–15, 2015.

[42] J. Kennedy, R.C. Eberhart, The particle swarm: social adaptation in information processing systems, in: D. Corne, M. Dorigo and F. Glover (eds.), New ideas in Optimization, McGraw-Hill, Cambridge, UK, pp. 11-32, 1999.

[43] F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial, Chemometrics and Intelligent Laboratory Systems **149**, pp. 153-165, 2015.

[44] M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.