BioMedInformatics

Article

Deep Learning Architecture Optimization with Metaheuristic Algorithms for Predicting BRCA1/BRCA2 Pathogenicity NGS Analysis

Eric Pellegrino [1,*], Theo Brunet [1], Christel Pissier [1], Clara Camilla [1], Norman Abbou [2], Nathalie Beaufils [1], Isabelle Nanni-Metellus [1], Philippe Métellus [3] and L'Houcine Ouafik [1,4]

1 APHM, CHU Nord, Service d'Onco-Biologie, 13015 Marseille, France; brunet.theo83140@gmail.com (T.B.); christel.pissier@ap-hm.fr (C.P.); clara.camilla@ap-hm.fr (C.C.); nathalie.beaufils@ap-hm.fr (N.B.); isabelle.nanni@ap-hm.fr (I.N.-M.); lhoucine.ouafik@univ-amu.fr (L.O.)
2 APHM, CHU Nord, Service de Biochimie et de Biologie Moleculaire, 13015 Marseille, France; norman.abbou@ap-hm.fr
3 Centre Hospitalier Clairval, Departement de Neurochirurgie, 13009 Marseille, France; philippe.metellus@outlook.fr
4 Aix Marseille Univ, CNRS, INP, Inst Neurophysiopathol, 13005 Marseille, France
* Correspondence: eric.pellegrino@univ-amu.fr

Abstract: Motivation, BRCA1 and BRCA2 are genes with tumor suppressor activity. They are involved in a considerable number of biological processes. To help the biologist in tumor classification, we developed a deep learning algorithm. The question when we want to construct a neural network is how many hidden layers and neurons should we use. If the number of inputs and outputs is defined by the problem, the number of hidden layers and neurons is difficult to define. Hidden layers and neurons that make up each layer of the neural network influence the performance of system predictions. There are different methods for finding the optimal architecture. In this paper, we present the two packages that we have developed, the genetic algorithm (GA) and the particle swarm optimization (PSO) to optimize the parameters of the neural network for predicting BRCA1 and BRCA2 pathogenicity; Results, we will compare the results obtained by the two algorithms. We used datasets collected from our NGS analysis of BRCA1 and BRCA2 genes to train deep learning models. It represents a data collection of 11,875 BRCA1 and BRCA2 variants. Our preliminary results show that the PSO provided the most significant architecture of hidden layers and the number of neurons compared to grid search and GA; Conclusions, the optimal architecture found by the PSO algorithm is composed of 6 hidden layers with 275 hidden nodes with an accuracy of 0.98, precision 0.99, recall 0.98, and a specificity of 0.99.

Keywords: deep learning; genetic algorithm; particle swarm optimization; BRCA1 and BRCA2 genes; new generation sequencing; ngs analysis; neural network optimization; genes prediction; bioinformatics

check for updates

Citation: Pellegrino, E.; Brunet, T.; Pissier, C.; Camilla, C.; Abbou, N.; Beaufils, N.; Nanni, I.; Metellus, P.; Ouafik, L. Deep Learning Architecture Optimization with Metaheuristic Algorithms for Predicting BRCA1/BRCA2 Pathogenicity NGS Analysis. *Biomedinformatics* **2022**, 2, 244–267. https://doi.org/10.3390/biomedinformatics2020016

Academic Editors: Chunhua Su, Celestine Iwendi, Thippa Reddy Gadekallu and Keping Yu

Received: 18 March 2022
Accepted: 9 April 2022
Published: 18 April 2022

1. Introduction

Deep learning is a branch of machine learning. Unlike traditional machine learning algorithms, whose ability to learn is limited regardless of the amount of data acquired, deep learning systems can improve their performance by accessing more data. Once the machines have gained enough experience through deep learning, they can be used for specific tasks such as driving a car, detecting weeds in a field, detecting diseases, and inspecting machines for faults. Deep learning (deep neural networks) methods were originally inspired by the human brain to imitate the vast network of interconnected neurons. Unlike shallow neural networks which are composed of one hidden layer, deep learning is neural networks that use multiple hidden layers. In deep learning, we feed millions of data instances into a network of neurons, teaching them to recognize patterns from raw inputs [1]. From them, the network can learn automatically by adapting and correcting itself to fit observed patterns in the data. The ability to automatically construct data representations is a key advantage of deep neural networks over conventional machine learning [1]. There are many open-source deep learning projects, such as the H2O platform, Theano, Torch, Tensorflow, Caffe, and Keras. For the needs of the project, we used Keras Tensorflow version 2.2 with the R language version 3.6.

The network architecture is made by the choice of how many hidden layers $L = \{L_1, L_2, \ldots, L_n\}$ the network will have and how many hidden units $N = \{N_1, N_2, \ldots, N_n\}$ each hidden layer will have [2]. Parameters L and N are very important and have a major influence on the performance of deep machine learning [3]. Tuning these parameters can be done by hand with the tune grid search method or by choosing an empirical architecture like, for example, minimization of quadratic search [4]. Grid search can save time in setting L and N. With this method, the structure could be tuned until a suitable number of nodes and/or layers is found to reduce or remove overfitting for the problem. Alternately, the model could be overfitting and then pruned by removing nodes until it achieves suitable performance on a validation dataset [5]. However, the grid search method is still time-consuming as the number of combinations is exponential. Moreover, if the list of the tuning parameters is chosen badly or poorly, the network may not be efficient in prediction or learn slowly and perhaps not at all [3].

This paper proposes two parameters selection methods for deep learning models: genetic algorithm (GA) and particle swarm optimization (PSO) to tune the architecture of fully connected deep neural networks (Multi-Layer Perceptron—MLP). We will tune, the number of hidden layers, the number of hidden neurons, dropouts, and activation functions that compose each hidden layer. GA and PSO were written in R language and re-adapted to work with Keras Tensorflow API. To perform our tuning test, we worked on our data collection of BRCA1/BRCA2 genes (BRCA1 benign 2632, BRCA1 pathogenic 2660, BRCA2 benign 3446, BRCA2 pathogenic 3137), which represents from January 2018 until April 2021 a total of 11,875 variants of BRCA1 and BRCA2. These genes are involved in many forms of cancer, mainly breast cancer, where their names come from (BReast CAncer) but also breast cancer ovaries or prostate. However, cancers caused by a BRCA mutation have the advantage of responding well to treatment-inhibiting PARP, which makes it possible to prevent heavier treatments like chemotherapy. It is therefore extremely beneficial to sequence these two genes in the context of personalized medicine or as part of the prevention of cancer risk. The best architecture was implemented routinely to predict the pathogenic character of variants. A genetic algorithm (GA) is an algorithm that attempts to simulate the evolution of populations to find the optimal solution for a problem. It belongs to the method called ‘evolutionary algorithms based on the principles of selection and mutation. GA was introduced by J. Holland [6] and it is based on the natural evolution theory of Darwin. Particle Swarm Optimization (PSO) is one of many optimization algorithms that are inspired by nature. PSO simulates the behavior of a flock of birds or a school of fish. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [7,8]. It is initialized with a pool of random solutions and searches for optima by updating through iterations. In PSO, the potential solutions are called particles and move through the problem space by following the current optimum particles.

2. Materials and Methods

The BRCA1 gene is located on the long arm of chromosome 17 (chromosomal location:17q21.31). It consists of 23 exons, 22 of which are coding, on a locus of 81 kb (1 kb = 1 kilobase = 1000 bases). The transcription of this gene provides the main transcript of 7.2 kb, encoding a protein of 1863 amino acids. The BRCA2 gene is located on the long arm of chromosome 13 (chromosomal location: 13q13.1). This gene contains 27 exons, 26 of which are coding, on a locus of 84 kb genomics. Transcription of this gene provides a 10.98 kb transcript, encoding a protein of 3418 amino acids.

For all clinical samples, after tumoral genomic DNA extraction by the Maxwell RSC DNA FFPE kit or Maxwell RSC Cell DNA (Promega, Charbonnières-les-Bains, France), we performed the full sequencing for the genes BRCA1 and BRCA2 (on coding regions and intron-exon junctions +/− 28 intron base pairs) with the technology of Ion Torrent S5 XL ThermoFisher with a sensitivity of 5% and minimum coverage of 300×. × The kit used is Oncomine BRCA. Then, sequencing data were analyzed through 2 pipelines. The first pipeline was developed by ThermoFisher on the IonTorrent Suite + Ion Reporter. IonTorrent Suite generates FASTQ data and ensures BAM (binary alignment mapping) alignment with the hg19 reference genome by using the TMAP (Torrent Mapping Alignment Program). Ion Reporter makes variant caller and variant annotations. The second pipeline was developed in our laboratory and runs open-source software such as BWA-MEM (Burrows-Wheeler Aligner) for alignment, SAMtools for mpileup, VarScan2 as variant caller, and VEP (Variant Effect Predictor) Ensemble for annotations. All data are stored in our local MySQL (My Structured Query Language) database (Figure 1). Biologists manually validated variants as pathogenic if the following statements were true:

- Variant allele frequency $\geq$ 5%
- Variant reads $\geq$ 300 reads
- Amino acid change is different from synonymous ($\neq$p.(=)). A synonymous variant will probably have a low influence on the gene because the amino acid does not change
- Polyphen (Polymorphism Phenotyping) score is in the range of [0.85, 1] (case of substitution variant)
- Grantham score is in a range of [5; 215] (in the case of substitution variant). [0, 50] = conservative, [51; 100] moderately conservative, [101; 150] moderately radical, over 150 radical
- Manual inspection on different databases and prediction tools: Arup database, VarSeak, Varsome, UMD Predictor, Cancer Genome Interpreter
- We use the tool Integrative Genomics Viewer (IGV) to check if alignment sequences are clear and show no strand bias in the region where the variant is located. It allows us to eliminate false positives.
- We verify the presence of these pathogenic variants in our second-in-house pipeline to validate them

With these requirements, biologists determine whether a variant can be validated or rejected as a function of the patient's pathology.

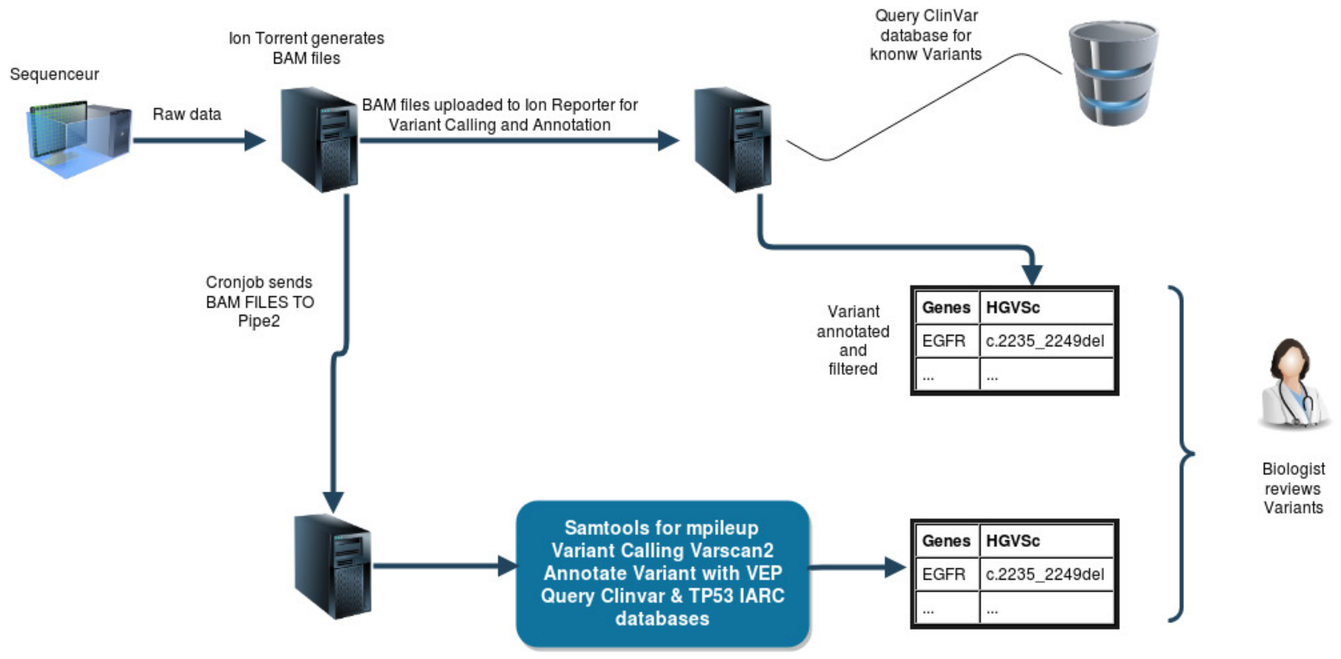

Sequenceur
Raw data
Ion Torrent generates BAM files
BAM files uploaded to Ion Reporter for Variant Calling and Annotation
Query ClinVar database for knonw Variants
Cronjob sends BAM FILES TO Pipe2
Variant annotated and filtered
Genes | HGVSc
EGFR | c.2235_2249del
... | ...
Biologist reviews Variants
Samtools for mpileup
Variant Calling Varscan2
Annotate Variant with VEP
Query Clinvar & TP53 IARC databases
Genes | HGVSc
EGFR | c.2235_2249del
... | ...

Figure 1. General workflow New Generation Sequencing (NGS). Analysis workflow performed at the platform to ensure oncosomatic variant analysis.

2.1. Programming Language

The programming language used to develop the deep learning neural network was R version 3.6 with Keras Tensorflow 2.2. Libraries for genetic algorithms and particle swarm optimization were developed by the platform and they are available on Github and CRAN.

2.2. Code Availability

Codes for this work can be accessed freely for academic use. To install our PSO package in Rstudio:

- install.packages("particle.swarm.optimisation")

To install our GA package in Rstudio:

- install_gitlab("brunet.theo83140/genetics.algorithm")

2.3. Data Selection

BRCA1/BRCA2 dataset contains clinical information of 5292 mutation variants of BRCA1 and 6538 mutation variants of BRCA2. This represents a collection of 11,875 variants composed of Single Nucleoid Variant (SNV), MNV, and INDEL (Figure 2). All variants were labeled by the biologist as benign or pathogenic in the function of the pathology of the patient.

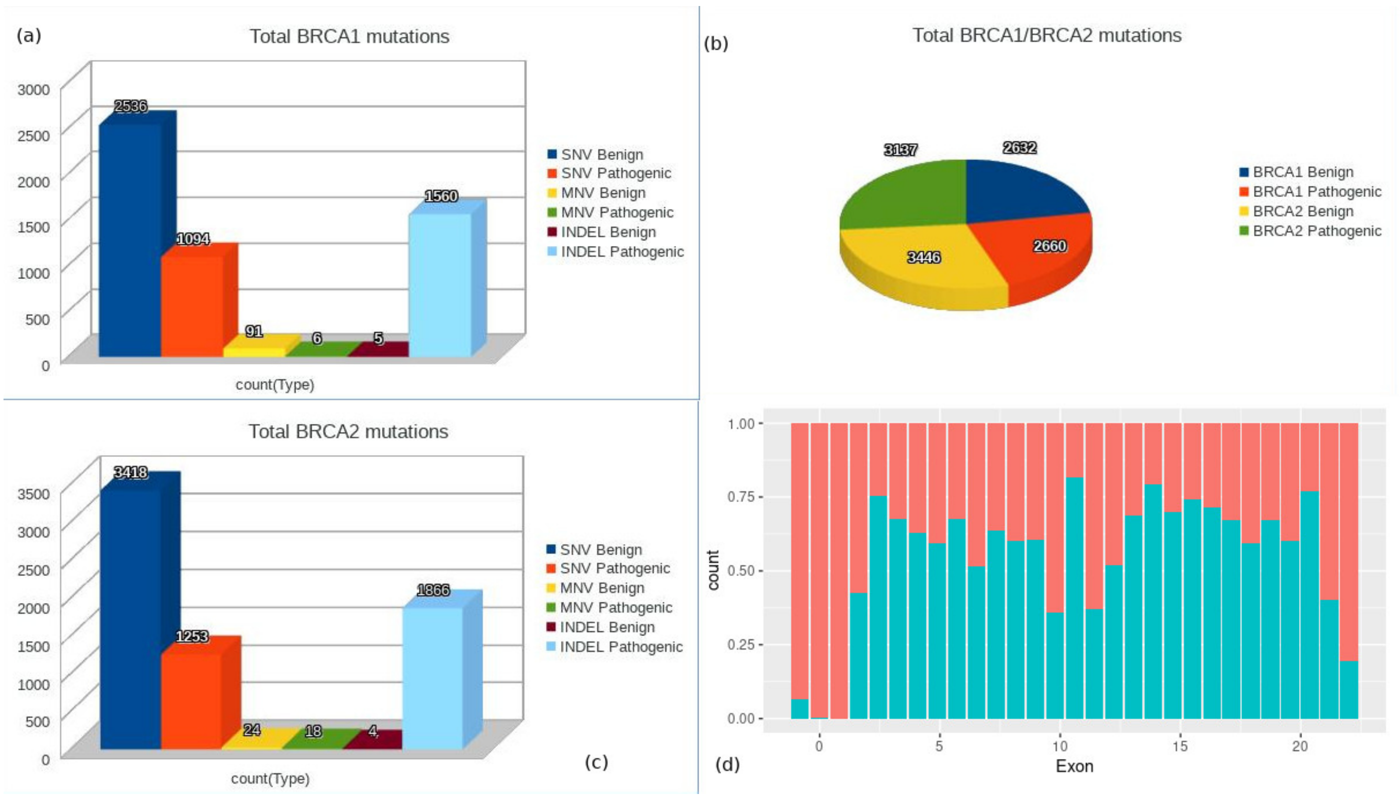


Figure 2. (**a**) Total of BRCA1 mutations benign and pathogenic in our local database. Mutations are grouped by single nucleotide polymorphism (SNV), multiple nucleotide variant (MNV), and insertion deletion (INDEL) in function of their pathogenicity. (**b**) Total mutations BRCA1 and BRCA2 stored in our database. (**c**) Total of BRCA2 mutations benign and pathogenic in our local database. Mutations are grouped by single nucleotide polymorphism (SNV), multiple nucleotide variant (MNV), and insertion deletion (INDEL) in function of their pathogenicity. (**d**) Proportion of pathogenic and benign variants in function of their position in the exon (−1 corresponds to a splice site mutation).

2.4. Problem Formulation

We consider our problem as supervised learning for a classification task. In input, we have real numbers and in output, we have a category. The model we are looking for is a function into a discrete space.

$$f : \mathbb{R}^{15} \rightharpoonup \{Pathogenic, Benign\} \tag{1}$$

2.5. Data Encoding (Feature Construction)

These data come from the laboratory and are produced by the pipeline of the Ion Torrent sequencer. They are present in the form of a .tsv (tabulated separated values) file where each line corresponds to a mutation in the BRCA1 gene or BRCA2. All data were labeled by the biologist as 1 for pathogenic and 0 for benign/uncertain significance (identified by the columns isMut). Our deep learning neural network will take in input 15 features and 2 possible values in output (see Table 1 and Equation (2)). We decided to pick values that the biologist used to interpret. We have decided to exclude data that contain multiple values mixing numbers and unusable characters. Data processing is an important step to improve the performance and the quality of the model. It involves data cleaning, such as removing errors and dealing with missing values or extreme observations. It also requires data transformation by changing in our case the data type by transferring categorical data into numerical data (Figure 3) and by changing the range of the data value by applying normalization.

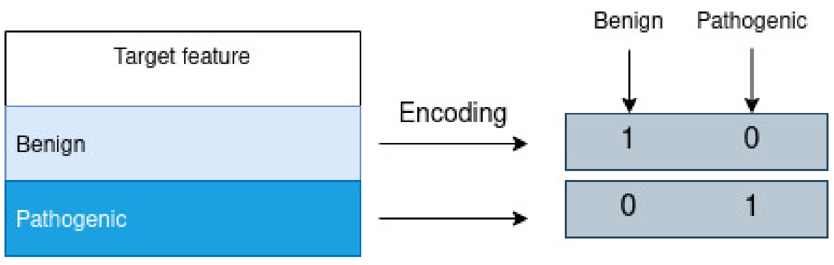


Figure 3. Categorical features are one-hot encoded. This makes them directly appropriate to use with the categorical cross-entropy loss function.

Table 1. Feature encoding for training the deep neural network.

Row Names	Data Description	Data Coding	Data Type
Location	Position of the mutation on the gene (exon, intronic, splicing site)	Intronic = 0, utr_5′ = 1, utr_3′ = 1, splicesite_3′ = 2, splicesite_5′ = 2, exonic = 3	Integer
Genes	BRCA1 or BRCA2	BRCA1 = 0, BRCA2 = 1	Integer
Transcript	Transcript ID	NM_007294.3 = 0, NM_007300.3 = 0, NM_000059.3 = 1	Integer
Locus	Coordinates on the genome	No recoding	Big interger
Type	Type of mutation: single nucleoide variant (SNV), mutiple nucleotide variants (MNV), insertion/deletion (INDEL)	SNV = 0, MNV = 1, INDEL = 2	Integer

Table 1. *Cont*.

Row Names	Data Description	Data Coding	Data Type
Exon	Exon number. 1 for the first exon in gene, 2 for the second one, . . . , n	$exon = \{1, \ldots, n\}$ $exon = 0$ when it is intronic. $Exon \in \mathbb{N}$	Integer
Freq	Variant allele frequency (frequency of the mutation)	No recoding of the values.	Float
MAF	Minor Allele Frequency	MAF value if it exists, -1 when there is no MAF	Float
Coverage	Sequencing coverage (0 if < 300 reads, 1 if > 300 reads)	No recoding	Integer
protdesc	Mutation effect on the protein (amino acid change)	Intronic or splice site (p.? = 0), same acid amino (p.(=) = 1), acid amino change = 2	Integer
Polyphen	Polyphen score (from -1 to 1)	Polyphen score when it exists, -1 when it is not applicable	Float
Grantham	Grantham score of the mutation	Grantham value from 5 to 215. -1 when it is not applicable	$Grantham \in \mathbb{Z}$

Variant.effect	Effect of the mutation on the reading frame (frameshift, missense . . .)	Frameshitf = 3,
		missense = 1,
		nonsense = 2,
		synonymous = 0,
		unknown = -1
			$Variant.effect \in \mathbb{Z}$

aaref	Amino acid
	(before the mutation)
		Arg = 1, His = 2 ,
		Lys = 3 , Asp = 4, Glu = 5,
		Ser = 6 , Thr = 7 , Asn = 8,
		Gln = 9 , Trp = 10 , Sec = 11,
		Gly = 12, Pro = 13, Ala = 14,
		Val = 15, Ile = 16, Leu = 17,
		Met = 18, Phe = 19,
		Tyr = 20, Cys = 21
			Integer
			(more detail on Table 2)

aamut	Amino acid
	(after the mutation)
		Arg = 1, His = 2, Lys = 3,
		Asp = 4,Glu = 5, Ser = 6,
		Thr = 7, Asn = 8, Gln = 9,
		Trp = 10, Sec = 11, Gly = 12,
		Pro = 13, Ala = 14, Val = 15,
		Ile = 16, Leu = 17,
		Met = 18, Phe = 19,
		Tyr = 20, Cys = 21,
		fs = 22, Ter = 22, del = 22
			Interger
			(more detail on Table 2)

isMut	Potential pathogenic variant. Decision of the biologist on the mutation variant	Benign/uncertain significance = 0, pathogenic = 1	Boolean

Data selected to be passed as a parameter in the deep neural network. Some data had to be re-coded in order to be recognized by the deep neural network. The 15 input parameters are: location, genes, transcript, locus, type, exon, frequency (freq), Minor allele frequency (maf), coverage, protdesc, polyphen, grantham, variant.effect, aaref (amino acid reference), aamut (amino acide after the mutation). IsMut is the output parameter for the supervised learning.

The coding of amino acids was grouped according to their chemical properties (see Table 2).

Table 2. Chemical properties of amino acids.

Chemical Properties	**Amino Acids**
Acidic	Aspartic (Asp), Glutamic (Glu)
Aliphatic	Alanine (Ala), Glycine (Glycine), Isoleucine (Ile), Leucine (Leu), Valine (Val)
Amide	Asparagine (Asn), Glutamine (Gln)
Aromatic	Phenylalanine (Phe), Tryptophan (Trp), Tyrosine (Tyr)
Basic	Arginine (Arg), Histidine (His), Lysine (Lys)
Hydroxyl	Serine (Ser), Threonine (Thr)
Imino	Proline (Pro)
Sulfur	Cysteine (Cys), Methionine (Met)

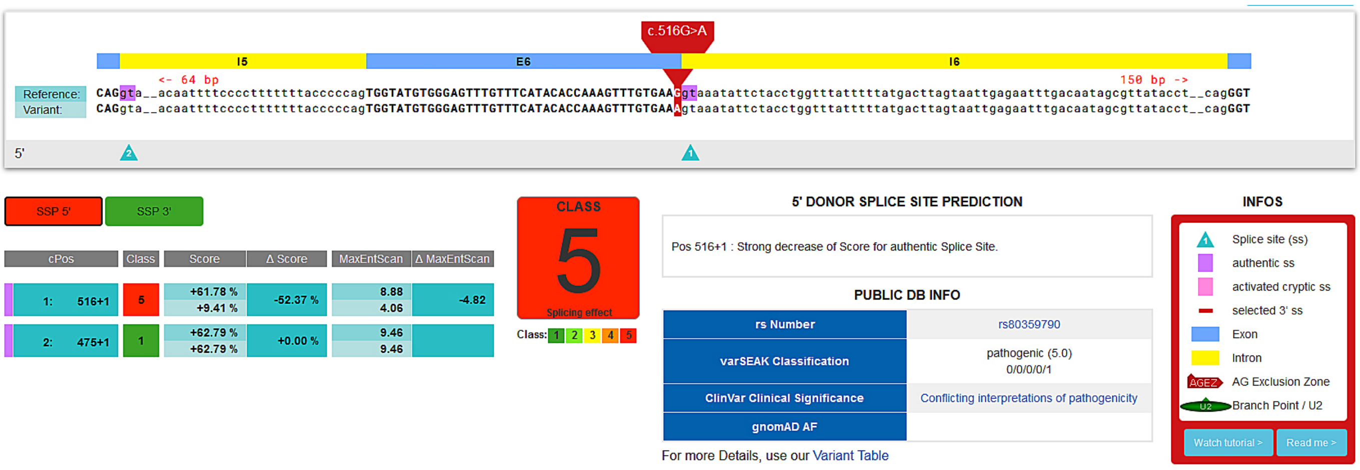
This table groups amino acids according to their chemical properties.

According to Equation (1), we formalized the mutation prediction problem as each candidate mutation site was represented by a feature vector x with the 15 inputs described in Table 1 and the goal is to predict variable y = {potential pathogenic variant} = {0 = benign or 1 = pathogenic}.

$$x_i = \{x_1, \ldots, x_{15}\} \rightarrow y \overset{?}{=} \left\{ \begin{array}{c} 1 = YES(pathogenic) \\ 0 = NO(benign) \end{array} \right\} \tag{2}$$

We have introduced amino acid change information into the inputs of the deep neural network for several reasons. The replacement of an amino acid with a different amino acid in the protein can remain a point of mutation in the DNA sequence. Not all amino acid replacements have an identical effect on the function or the structure of the protein. This will depend on the similarity or dissimilarity of the replacements, but also their position in the sequence or the structure. The similarity between amino acids can be calculated based on the substitution of matrices, the physicochemical distance, or their fundamental properties like the size or the charge [9]. We speak of conservative replacement when the amino acid is changed by another having a similar property. This type of replacement rarely corresponds to a dysfunction in the protein. Oppositely, in the case where the amino acid is changed into another amino acid with different properties, this induces changes in the structure of the protein or in its function which can cause changes in the phenotype and are often pathogenic. For example, a mutation at position 6 of a glutamic acid (negatively charged) that would be changed to a valine (uncharged) is known in human sickle cell anemia. The amino acid modification at the protein level can be tolerated by the cell without deleterious consequences, which explains why many "missense" type sequence variations produce no pathogenic effect and therefore constitute elsewhere a significant part of polymorphisms (SNP type). But depending on the location of the amino acid affected, missense mutations can cause deleterious effects (alteration of protein folding, protein stability, functional domains, sites of interaction with other proteins, etc.), like loss of function or gain of function. In the BRCA2 gene, some silent mutations are encountered as deleterious. This is the case, for example, with the BRCA2 mutation: c.516G>A (p.Lys72Lys/p. (=)) located in exon 6, more precisely, the last base in exon 6 next to the splice donor site (Figure 4).

We also decided to introduce the information on the Grantham score. It assesses the physicochemical distance between the difference-replaced amino acids. Grantham score is a formula for the difference between amino acids that combine properties that correlate best with protein residue substitution frequencies [10]. The score attempts to predict the distance between two amino acids. A low score denotes less evolutionary distance. A high score suggests a more considerable evolutionary distance and consequently, it can be considered deleterious. Indeed, the more distant two amino acids are, the more damaging is their substitution. For example, a substitution of isoleucine (Ile) for leucine (Leu) has a Grantham score of 5, which is predicted to be tolerated. Substitution of cysteine for tryptophan has a score of 215 which involves a very high score and is then predicted to be deleterious. Any variation involving cysteine has a very high Grantham score. Grantham's distance depends on three properties: composition, polarity, and molecular volume [10].



```
Reference: CAGgta__acaattttccccttttttaccccccagTGGTATGTGGGAGTTTGTTTCATACACCAAAGTTTGTGAAGgtaaatattctacctggtttattttatgacttagtaattgagaatttgacaatagcgttatacct__cagGGT
Variant:   CAGgta__acaattttccccttttttaccccccagTGGTATGTGGGAGTTTGTTTCATACACCAAAGTTTGTGAAAgtaaatattctacctggtttattttatgacttagtaattgagaatttgacaatagcgttatacct__cagGGT
```

cPos	Class	Score	Δ Score	MaxEntScan	Δ MaxEntScan
1: 516+1	5	+61.78 % +9.41 %	-52.37 %	8.88 4.06	-4.82
2: 475+1	1	+62.79 % +62.79 %	+0.00 %	9.46 9.46	

CLASS
5
Splicing effect

Class: 1 2 3 4 5

5' DONOR SPLICE SITE PREDICTION

Pos 516+1 : Strong decrease of Score for authentic Splice Site.

PUBLIC DB INFO

rs Number	rs80359790
varSEAK Classification	pathogenic (5.0) 0/0/0/0/1
ClinVar Clinical Significance	Conflicting interpretations of pathogenicity
gnomAD AF	

For more Details, use our Variant Table

INFOS

▲	Splice site (ss)
■	authentic ss
■	activated cryptic ss
▬	selected 3' ss
■	Exon
■	Intron
AGEZ	AG Exclusion Zone
U2	Branch Point / U2

Figure 4. VarSeak result prediction score on the mutation BRCA2:c.516G>A. The mutation is found at the border between exon 6 and Donor Splice Site 5′.

Distance difference D for each pair of amino acids i and j are calculated as:

$$D_{ij} = \left[\alpha\left(c_i - c_j\right)^2 + \beta\left(p_i - p_j\right)^2 + \gamma\left(v_i - v_j\right)^2\right] \tag{3}$$

where $c = composition, p = polarity, and\ v = molecular\ volume$; and are constants of squares of the inverses of the mean distance for each property.

Polyphen score predicts the possible impact of an amino acid substitution on the structure and function of a human protein using straightforward physical and comparative considerations. This score is the probability that a substitution is damaging. The score ranges from 0.0 (tolerated) to 1.0 for deleterious. The possible score range can be interpreted as follows:

- 0.0 to 0.15 $\rightarrow$ predicted to be benign
- 0.15 to 1.0 $\rightarrow$ possibly damaging
- 0.85 to 1.0 $\rightarrow$ more confidently predicted to be damaging

2.6. Data Normalization

When working on a deep neural network, it is important to normalize or standardize the data to make training faster and reduce the chances of getting stuck in local optima. If the distribution of the quantity is normal, then it should be standardized; otherwise, the data should be normalized [5]. In our case, data has varying scales (range of position min and max and frequency min and max), so we will apply a normalization between the minimum and the maximum on each value by using the following formula:

$$x_{norm} = \frac{x - min}{max - min} \tag{4}$$

x_{norm} is the value after normalization, x is the value to normalize, min is the minimum value, max is the maximum value. With this approach, data are scaled to an established range from 0 to 1. In contrast to standardization, we will end up with a smaller standard deviation, which can suppress the effect of outlier values. We divided the main dataset of 11,875 variant calls into two subsets. The deep neural network was trained on 70% of the dataset and 30% remaining for the validation (unseen data). We used the "validation split" function from the Keras library to divide the dataset. This function was designed to ensure that the data are separated in such a way that it always trains on the same portion of the data for each epoch. All shuffling is done within the training sample between epochs. Keras proportionally splits your training set by the value of the variable. The first set is used for training and the second set is for validation after each epoch. After constructing the feature value vector x_i for each BRCA1/BRCA2 candidate mutation position, the problem is to find the best deep learning architecture which optimally separates the true pathogenic mutations from benign mutations. We compared the performance of the different models in the specific context of predicting BRCA1/BRCA2 mutations from NGS data. We used the categorical cross-entropy loss function as it fits for multi-class classification tasks. These are tasks where the model must decide which one belongs to the possible categories. It quantifies the difference between two probability distributions (Figure 5).

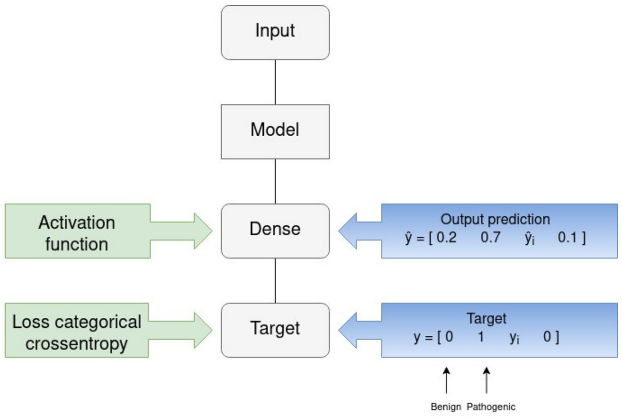


Figure 5. The target indicates which one out of two classes pathogenic,benign classes is correct. The categorical cross-entropy is suited to classification tasks, since one example can be considered to belong to a specific category with the probability one, and to the other categories with probability zero.

2.7. Grid Search Method

The first approach was grid search (hypertuning), which consists of defining a list of parameters for the neural network and testing each of the possible combinations of these parameters to measure the precision and cost of each neural network and therefore, find the most exact configuration of parameters. To achieve this, we have chosen to use a library called tfruns which uses Tensorflow to perform the grid search. Several tests were carried out by trying to vary several parameters: the activation function, dropout, number of neurons, number of layers, and training function. To choose the number of hidden layers and nodes which compose each of them, we can refer to some formulas found in the literature. The number of hidden layers could be selected to be between the number of inputs and outputs [3,11]. The number of hidden layers can be based on the following formula [12]:

$$H = (I + O) \times \frac{2}{3} \tag{5}$$

where H is the number of neurons in the hidden layers, I is the number of input features, O, and is the number of neurons in the output layer. Some suggest that the number of hidden layers should never be over the number of input features [13]. The number of hidden neurons should be less than twice the number of neurons in the input layer [14]. There are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers, such as the following. The number of hidden neurons should be between the size of the input layer and the size of the output layer. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer. The number of hidden neurons should be less than twice the size of the input layer [15]. According to these rules, we will tune our grid search as reported in Table 3.

Table 3. Parameters used for the grid search.

Parameters	**Range**	**Number of Possibilities**
Number of neurons per hidden layers	[1; 50]	50
Number of hidden layers	[1: 5]	5
Activation functions	[relu, elu, tanh, sigmoid]	4
Training functions	Sgd, adam, adamax	3
Dropout	[0.1; 0.9]	9
Total of combination		27,000

Training a deep neural network remains always a challenging problem to make it generalize well to new data. A model composed with a weak capacity cannot learn (underfitting) whereas a model with too much capacity will fit correctly to the training dataset but fails to fit the validation dataset (overfitting). There are several ways to adapt a model to prevent overfitting by playing with some parameters. For example, dropout layers can remain a convenient way to prevent overfitting. Indeed, a dropout layer will randomly drop some of the connections between layers. Using dropout improves the computational efficiency of networks with large amounts of parameters [16]. Another way to prevent overfitting is to stop the training process earlier. Instead of training for a specified number of epochs, we stop as soon as validation loss rises. When we use early stopping, the performance of the model must be monitored during training. We split the training dataset and use a subset as a validation dataset (30% of the dataset) used to monitor the performance of the model during the training. This validation set is not used to train the model and we used the loss on the validation dataset as the metric to monitor accuracy as we are in the case of classification problems. The performance of the model is evaluated on the validation set at the end of each epoch, which computes an additional computational cost during training. The training is stopped as soon as the performance on the validation dataset decreases as compared to the performance on the validation dataset at the prior training epoch (in our case, an increase in the loss value). For our study, we fixed the following parameters for early stopping:

- Monitor: validation loss (val_loss)
- Minimum change in the monitored quantity to qualify as an improvement (min_delta) = 0.0005
- Number of epochs with no improvement after which training will be stopped (patience) = 10
- Training will stop when the quantity monitored has stopped decreasing (mode) = 'min'

When configuring a deep neural network, it is also beneficial to tune the learning rate hyperparameter. It controls how much we are adjusting the weights of the deep neural network for the loss gradient. It is also a challenge to find good value, as too small of a value may take a long time to train the network, whereas too large of a value may result in an unstable training process. Consequently, it will take a long time to converge. The following formula reveals the relationship:

$$\text{new weight} = \text{existing weight} - \text{learning rate} \times \textit{gradient} \tag{6}$$

if we note Θ_1 the weight, α the learning rate and $\frac{\partial}{\partial\Theta_1}J(\Theta_1)$ the gradient, it gives us:

$$\Theta_1 = \Theta_1 - \alpha\frac{\partial}{\partial\Theta_1}J(\Theta_1) \tag{7}$$

when $\alpha \rightarrow 0$ gradient descent is slow and when $\alpha \rightarrow +\infty$ gradient descent fail to converge. Keras offers a function callback_reduce_lr_on_plateau, it reduces learning rate when a metric has stopped improving. Based on our previous tests, we choose the following values: callback_reduce_lr_on_plateau(monitor = 'val_loss', factor = 0.75, patience = 5, mode = 'min')).

We monitored the loss, the factor by which the learning rate will be reduced to new lr $=$ lr $\times$ factor was fixed to 0.75. The patience is the number of epochs with no improvement, after which the learning rate will be reduced and fixed to 5. The mode was fixed to min; the learning rate will be reduced when the quantity monitored has stopped decreasing.

2.8. Genetic Algorithm (GA)

To reduce the calculation time and try to find an architecture with better accuracy, we used genetic algorithms to tune our deep neural networks. Tuned parameters are the number of hidden layers (L), the number of neurons (N_i), activation functions (σ_i), dropout (δ_i), and the training functions (Γ_i). To our knowledge, no one has used the genetic algorithm for setting these parameters with Keras for predicting BRCA1/BRCA2 pathogenicity. GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics-inspired operators of crossover, mutation, and inversion [17]. It is an algorithm that tries to simulate the evolution of populations to find the optimal solution to a problem. To utilize it, it is necessary to define the parameters to vary, which will be called genes. They form an object called a chromosome which will be the essential part of the genetic algorithm. The genetic algorithm will subsequently produce a random population of chromosomes, where each chromosome defines a neural network. Once the chromosomes are formed, we will calculate their fitness scores. Using this score, the algorithm will be able to make a selection of the best individuals in the population. At that point, these individuals will be used to form a new chromosome by crossing-over (the mix of two parent chromosomes) and by mutation (change of one of the genes of the child chromosome). By performing these operations, we obtain a new generation that should be better than the previous generation, and by repeating this cycle ended up obtaining a homogeneous population that must correspond to the best solution found by the genetic algorithm (Figure 6).

The genetic algorithm package for Keras was developed by using the R6 method, which allows doing object programming. The package is available in the git repository or directly in R with devtools.

Since we are in a classification task, we will use classification accuracy as our fitness function (objective function). To guarantee the existence of at least one hidden layer, we apply the constraint:

$$\text{Hidden layer} = \sum_{i=1}^{N} H_i \geq 1 \tag{8}$$

where H_i is a binary variable, which equals the value 1 if the i hidden layer is used otherwise zero. The second constraint is if the models do not use a hidden layer, consequently their neurons are not used. $neuron_{ij}$ is a binary variable that equals the value 1 if the i neuron of the i hidden layer is used; otherwise, it equals zero. We note n_i the number of neurons in the ith hidden layer.

$$\sum_{j=1}^{n_i} neuron_{ij} \leq \alpha \times \text{Hidden Layer}_i \tag{9}$$

where α is a positive scalar and $j = 1, \ldots, n_i$.

The strategy is to use a generic algorithm to crossover the candidates over generations until we can determine the best combination for our model.

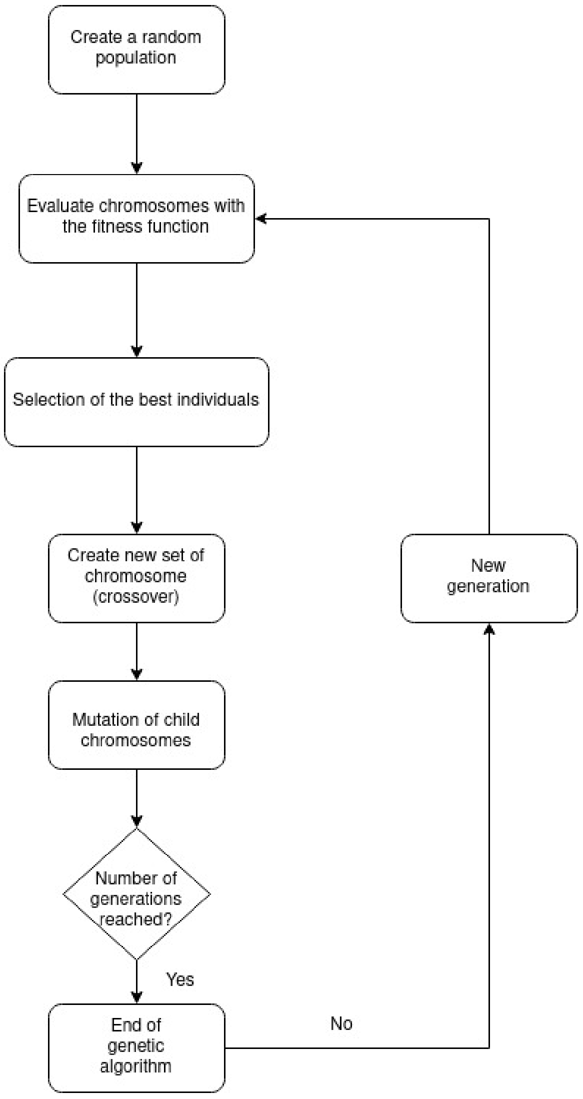

Create a random population
Evaluate chromosomes with the fitness function
Selection of the best individuals
Create new set of chromosome (crossover)
Mutation of child chromosomes
Number of generations reached?
Yes
End of genetic algorithm
No
New generation

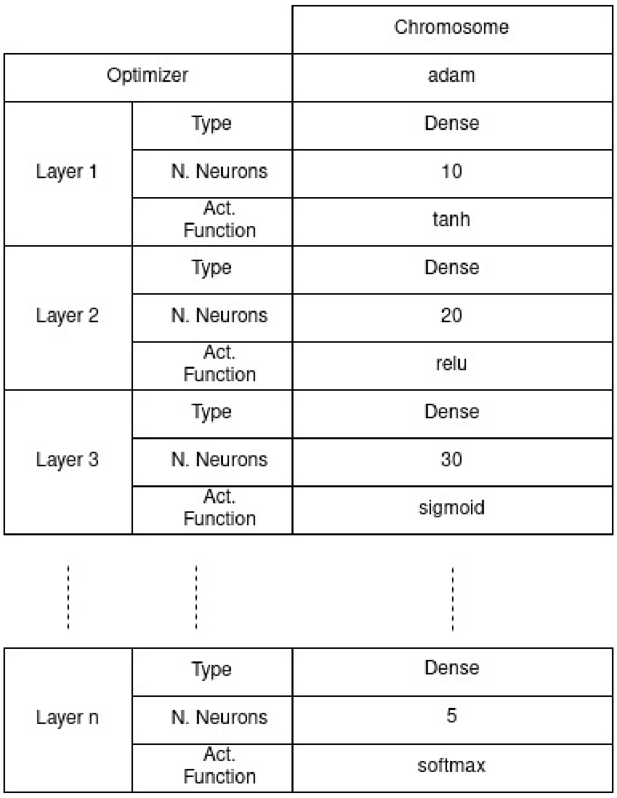
Figure 6. Workflow of GA: 1. Initial Population: first step, we generate randomly the first parents (population). We define the population size and generate each one with a random configuration. Randomly choose each parameter on the chromosome. 2. Evaluate individuals: run evaluation on each model in the current generation. Scores each member of the population based on the goal to reach. This score is called the fitness function. 3. Selection of the best members of the population, applying the evaluator, and sorting individuals by rank. 4. Mutates some members randomly to attempt to find even better candidates. Mutation helps in getting a more diverse opportunity. The obtained population will be used in the next generation. Repeat steps 2 to 4 again for each generation. 5. Stop decision: we define a stop condition; we define it after 20 generations.

Chromosome Structure

For the chromosome structure, we decided that our chromosomes would carry information on hidden layers, neurons, and the model. The chromosome stores:

- Optimizer functions: [adam, adamax, sgd]
- Hidden layer layout: [Number of neurons on a range of [1, 300], for each layer the activation function: [relu, elu, tanh, sigmoid] and dropout on a range of [0.1 to 0.9]

Parameters used for the genetic algorithm can be found in Table 4. They were obtained after several tests of the genetic algorithm on small populations. The ranges of values are presented in Tables 5 and 6. An example of a chromosome is shown in Figure 7.



		Chromosome
Optimizer		adam
Layer 1	Type	Dense
	N. Neurons	10
	Act. Function	tanh
Layer 2	Type	Dense
	N. Neurons	20
	Act. Function	relu
Layer 3	Type	Dense
	N. Neurons	30
	Act. Function	sigmoid
Layer n	Type	Dense
	N. Neurons	5
	Act. Function	softmax

Figure 7. Example of one Chromosome holding information's about model and n hidden layers.

Table 4. Genetic algorithm parameters.

Parameters	First Run (GA1)	Second Run (GA2)
Elitism	0.3	0.3
Random selection	0.1	0.1
Population size	30	50
Mutation rate	0.25	0.25
Number of generations	20	40

Genetic algorithm parameters. **Elitism** remains the proportion of the best individuals selected for the next generation. **Random selection** corresponds to the proportion of random individuals selected for the next generation. This ensures maintaining a significant genetic diversity in each generation and avoids converging on a local minimum. **The Population size** must be significant enough to include the maximum number of solutions. **The mutation rate** defines the frequency at which genes will mutate. **The number of generations** will define the limits of the algorithm's research framework.

Table 5. Neural network parameter intervals (run 1).

Parameters	Number of Hidden Nodes per Hidden Layers	Activation Functions	Number of Hidden Layers	Optimizer
Values	1 to 300	Relu, elu, sigmoid, tanh	1 to 20	Adam, adamax, sgd

Neural network parameter intervals for run 1 of the PSO and the genetic algorithm.

Table 6. Neural network parameter intervals (run 2).

Parameters	Number of Hidden Nodes per Hidden Layers	Activation Functions	Number of Hidden Layers	Optimizer
Values	1 to 300	Relu, elu, sigmoid, tanh	1 to 6	Adam, adamax, sgd

Neural network parameter intervals for run 2 of the PSO and the genetic algorithm.

2.9. Particle Swarm Optimization (PSO)

Particle swarm optimization is a model developed by Craig Reynolds at the end of the 1980s, allowing us to simulate the movement of a group of birds. Another source of inspiration, claimed by the authors, James Kennedy and Russel Eberhart, is social psychology [7]. It is a biologically inspired method for solving optimization problems.

Like artificial neural networks, genetic algorithms, or ant colony algorithms, particle swarm optimization (PSO) is a bio-inspired algorithm. It is based on the principles of self-organization that allow a group of living organisms to act together in complex ways, based on simple "rules". This optimization method is based on the collaboration of individuals among themselves. Thus, thanks to very simple rules of displacement (in the space of solutions), the particles can gradually converge towards a global minimum Figure 8. However, this metaheuristic seems to work better for spaces in continuous variables. At the start of the algorithm, each particle is therefore positioned (randomly, or not) in the search space for the problem. Each iteration makes the particles move according to 3 components:

1. Its current speed V_i^t
2. His best solution $pbest_i$
3. The best solution obtained in its neighborhood $gbest$

The algorithm will simulate the movement of a swarm to find the best value. For the particles to move, the algorithm uses the following parameters [7]:

$$v_i^{t+1} = \omega V_i^t + c_1 \times r \times (pbest_i - X_i^t) + c_2 \times r \times (gbest - X_i^t) \tag{10}$$

Each step t, the position of particle i, X_i^t is updated based on the particle's velocity v_i^t, the new position of the particle would be:

$$X_i^{t+1} = X_i^t + v_i^{t+1} \tag{11}$$

ω is the inertia coefficient of the particle. It helps the particles move by inertia toward better positions. r is a random number between 0 and 1 that follows a uniform distribution. $pbest_i$ is the best value of the particle and $gbest$ is the best value of the swarm. c_1 is the personal acceleration coefficient and c_2 is the global acceleration coefficient; they represent the weights of approaching the $pbest_i$ and the $gbest$ of a particle. By iterating a significant number of iterations of these two formulas Equations (10) and (11), the particles will converge towards a value that should be the best solution achieved by the swarm. Workflow for particle swarm optimization is described in Figure 9.

The particles do not move randomly, they have a goal to achieve. This is determined by a function to be optimized (fitness function or "objective function") which is provided by the user, and which depends on the application concerned. The goal of our fitness function is to maximize the accuracy of the deep neural network on the validation set while applying a penalty on the loss to try to achieve the smallest loss and the highest accuracy. To have the minimum amount of fluctuation, we took the values over the last 5 epochs to smooth the values. PSO has been coded to maximize fitness value. We defined our fitness function as follows:

We note: validation accuracy = ψ_i and validation loss = φ_i

$$fitness = \sum_{epoch}^{epoch-4} \frac{\psi_i}{5} - \sum_{epoch}^{epoch-4} \frac{\varphi_i}{5} \tag{12}$$

In a classification task, the categorical cross-entropy is a loss function and it is defined as the following sum:

$$Loss = -\sum_{i=1}^{\text{output size}} y_i \times log(\widehat{y}_i) \tag{13}$$

where $\widehat{y}_i$ is the i-th scalar value in the model output, y_i is the corresponding target value, and the output size is the number of scalar values in the model output. The loss is closest to 0 when the predicted value equals the target (predicted—target). We apply the same constraints applied to GA defined by Equations (8) and (9). As we did with the genetic algorithm, we varied the following parameters numbers of hidden layers, the number of hidden nodes, activation function, and optimizers (see Tables 5 and 6). For the input parameters for the PSO, we started with some particles equal to 30 and a number of iterations of 20. We consequently increased them for the second PSO run (See Table 7). Parameters were selected based on tests and the article by Nuria Gomez et al. (2010) [18]. themselves.

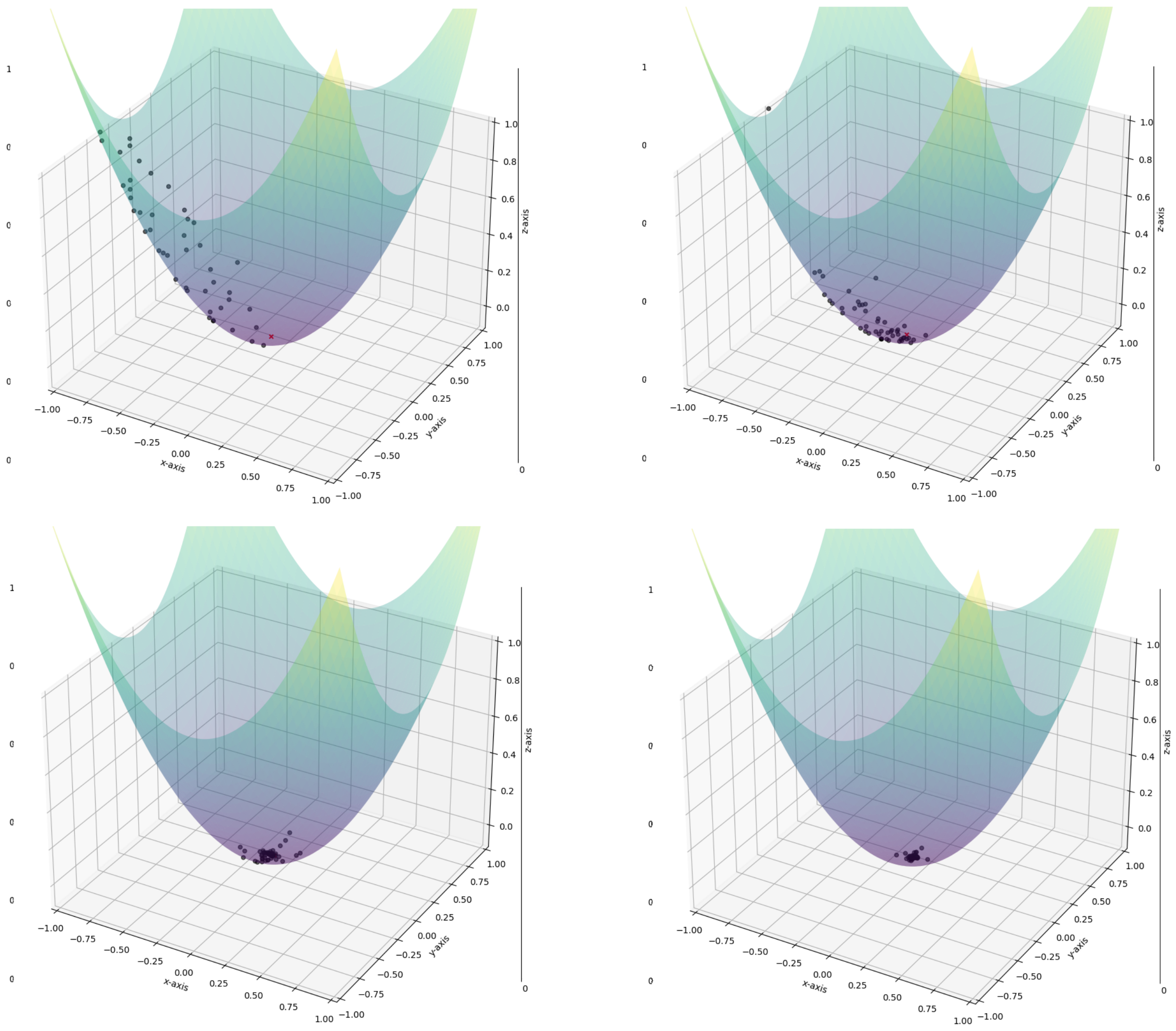


Figure 8. A demonstration of Particle Swarm Optimization in a 3 dimensional space. The particles are searching for the minimum of the surface. From the left to the right, show early, mid, and late stages of the convergence.

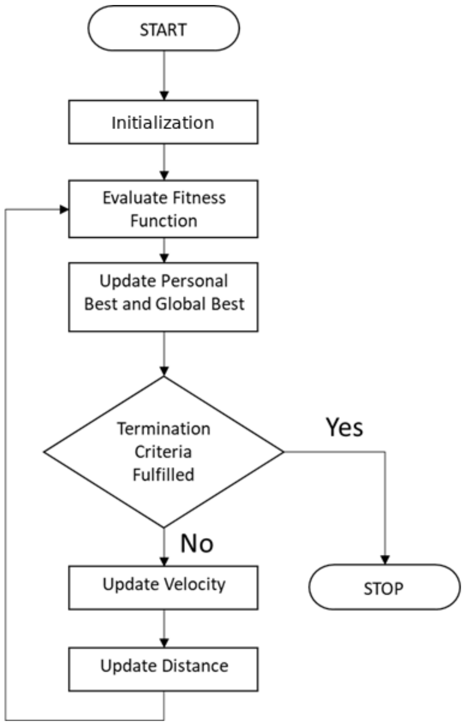

START
Initialization
Evaluate Fitness Function
Update Personal Best and Global Best
Termination Criteria Fulfilled
Yes
No
Update Velocity
STOP
Update Distance

Figure 9. Particle swarm optimization algorithm workflow.

Table 7. Particle swarm optimization parameters PSO (run 1) and PSO (run 2).

Parameters	PSO (Run 1)	PSO (Run 2)
Inertia	0.5	0.5
c1	[0:4]	[0:4]
c1	[0:4]	[0:4]
Number of particles	30	50
Number of iterations	20	50

Particle swarm optimization parameters were applied on the first run (PSO run 1) and on the second run (PSO run 2). Inertia stayed the same for both runs. The personal acceleration coefficient and the global acceleration coefficient were fixed to a range from 0 to 4 for both runs. The number of particles and number of iterations were increased for the second run.

3. Results

The experiments we conducted we evaluated the predictive models on validation sets and we compared the results of validation to obtain the best deep neural architecture for identifying pathogenicity in BRCA1/BRCA2 genes. Finally, we tested the best model based on a test dataset on other genes to obtain general predictions on oncosomatic mutation variants. To assess the performance of each model, accuracy, recall (sensitivity), precision and sensibility were calculated to measure the performance of classification models obtained with the particle swarm optimization and the genetic algorithm. These parameters are defined as follows:

- $Accuracy = \dfrac{TP + TN}{TP + TN + FP + FN}$
- $Precision = \dfrac{TP}{TP + FP}$
- $Recall = \dfrac{TP}{TP + FN}$

True positives (TP) and true negatives (TN) are defined as the number of mutations that are classified correctly as pathogenic and benign. False positives (FP) and false negatives (FN) are defined as the number of mutations that are misclassified. FP is a result where the model incorrectly predicts the positive class. An FN is a result where the model incorrectly predicts the negative class. Precision is defined as the number of positive samples the model predicts correctly (true positives) divided by the true positives plus the false positives. It attempts to determine the proportion of positive identifications that were correctly identified by the model. Recall attempts to determine the proportion of actual positives identified correctly. It is defined as true positives divided by true positives plus false positives. Model performance was evaluated using the receiver operating characteristic area under the curve. The receiver operating curve (ROC) is a graph where sensitivity is plotted as a function of $1 - specificity$. The area under the ROC is denoted AUC. The true positive rate (TPR) also known as sensitivity (or recall) is defined as the percentage of pathogenic mutations that are correctly identified. The specificity or true negative rate (TNR) is defined as the percentage of mutations that are correctly identified as benign.

- $Specificity = TNR = \dfrac{TN}{TN + FP}$

3.1. Grid Search Results

Grid search was used to find an optimal deep neural network architecture. The parameters used are presented on Table 3. To test all parameters, we had to evaluate the 27,000 combinations. The result obtained with this method is composed of 5 hidden layers with 19 hidden neurons with the elu as activation function and adam function as optimizer. The accuracy of this deep neural network reached 97.4% with an error rate of 1.8% on the validation set (test set). AUC is 98.2% (Figure 10). This neural network is capable of correctly differentiating between pathogenic and benign variants but keeps an error rate too high.

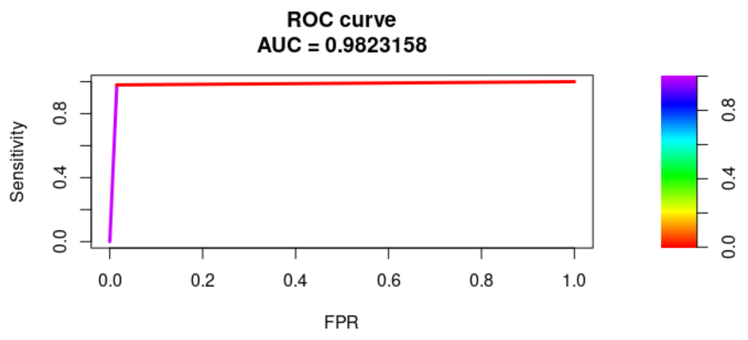


Figure 10. ROC curve for the deep neural network composed of 5 hidden layers with 19 hidden nodes each obtained with the grid search method. AUC is 0.98.

3.2. GA Results

Initially, the genetic algorithm was implemented to provide the optimal number of hidden layers as well as the number of hidden nodes. The neuron network obtained consists of 6 hidden layers of 297 hidden nodes, each with the relu activation function (Tables 8 and 9). The dropout value is equal to 0.3 on each hidden layer. With this configuration, we obtained a precision of 98.51% during the training phase and an AUC of 98.49% (Figure 11). The error rate on the validation set is 1.35% with a precision of 98.97% (GA run 1—Table 10). The results obtained with this neural network are superior to those obtained by the grid search technique. This deep neural network is capable of distinguishing between pathogenic and benign mutations. When it predicts that a variant is pathogenic, it was correct 98.97% of the time. With the help of these primary results, we re-implemented the genetic algorithm to vary the activation functions and the number of neurons in each layer independently of the others (Table 6). The results are visible in Table 10. The result represents a deep neural network with a maximum precision of 98.83% (GA run 2—Table 10) on the training set, with an error rate of 1.13% on the test set (validation set) and an AUC of 98.69% (Figure 11).

There is an improvement compared to the previous results and allows to have a better prediction of the data. Specificity and sensitivity were improved with this configuration. With a sensitivity of 98.79%, it correctly identified 98.97% of pathogenic tumors in BRCA1/BRCA2 mutation variants. Specificity of 99.01%, the model is correct 99.01% of the time when a variant is not pathogenic (benign).

We additionally note that in this run all the individuals of the population involve the same genes (except those having undergone a mutation during the last generation). This means that the genetic algorithm worked and gained a homogeneous population, the number of generations is therefore sufficient.

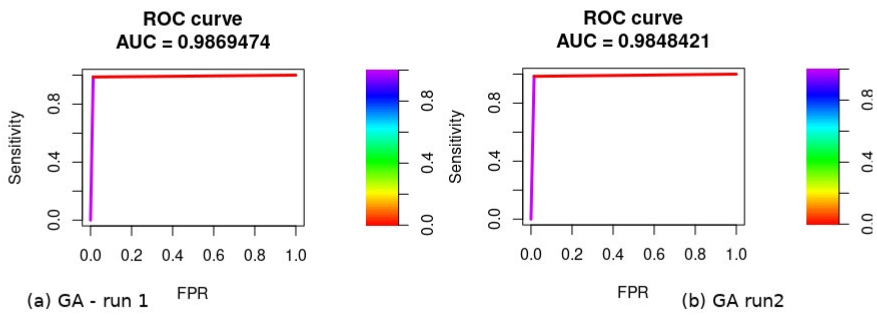


Figure 11. ROC curve for the validation set. Model performance was evaluated using the receiver operating characteristic (ROC) curve. The area under the curve is denoted AUC. The AUC obtained was 0.9869 for the first run with the genetic algorithm and 0.9848 for the second run. A large area under the curve was observed for both models. These models could then determine whether a variant was benign or pathogenic with a minimal error rate.

Table 8. Architectures returned by GA and PSO (run 1).

Models	**Number of hidden nodes per hidden layers**	**Number of hidden layers**	**Activation function**	**Optimizer**
GA (run 1)	297	6	relu	adam
PSO (run 1)	275	6	relu	adam

Architectures returned by GA and PSO on the first test run.

Table 9. Architectures returned by GA and PSO (run 2).

Models	**H1**	**F1**	**H2**	**F2**	**H3**	**F3**	**H4**	**F4**	**H4**	**F5**	**H6**	**F6**	**Optimizer**
GA run 2	212	Elu	134	Relu	246	Sigmoid	285	Relu	61	Relu	196	Relu	Adam
PSO run 2	45	Relu	156	Relu	132	Relu	284	Elu	105	Tanh	169	Tanh	Adam

Architectures returned by GA and PSO on the second test run. We varied the number of neurons hidden on the two runs. H_i denotes the number of the hidden layer. F_i denotes the activation function on the i-th hidden layer.

Table 10. Metaheuristic performances on run 1 and run 2 .

Models	**Precision**	**Sensitivity**	**Specificity**	**Accuracy**	**AUC**
GA (run 1)	0.9897	0.9822	0.9899	0.9861	0.9848
GA (run 2)	0.9896	0.9879	0.9901	0.9892	0.9869
PSO (run 1)	0.9965	0.9863	0.9966	0.9852	0.9858
PSO (run 2)	0.9898	0.9821	0.9883	0.9915	0.9865

The performance comparison of various deep neural network models architecture found by GA and PSO on validation set.

3.3. PSO Results

With the first PSO run (run 1), we obtained slightly different results from those obtained with GA (run 1). We obtained an architecture comprising 6 layers of hidden layers consisting of 275 neurons (Table 8 and Figure 12). The activation function is relu on each layer with a dropout of 0.3. Accuracy during the training phase is 98.61% and 98.52% on the validation set (Table 10) with an AUC of 0.9858 (Figure 13). Manipulating these values, we performed a second run (PSO run 2—Table 6) from which we varied the number of neurons as well as the activation function within the 6 hidden layers. We improved the accuracy of the model with a value of 99.15%. The architecture is described on (PSO run 2—Table 9). However, we lose an average of 0.64% in quality on the precision, specificity, and sensitivity, but the model stays efficient in the pathogenicity prediction. The AUC obtained for the second run (PSO run 2) is 0.9865 (Figure 13).

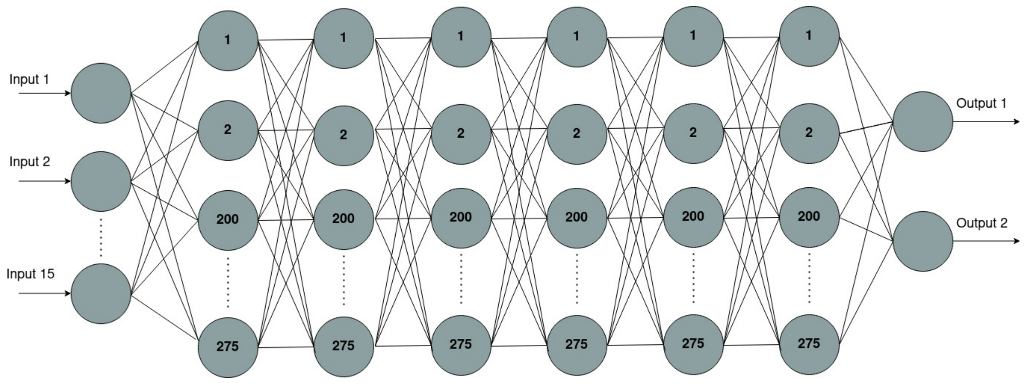


Figure 12. Architecture returned by particle swarm optimization algorithm for predicting pathogenicity of BRCA1 and BRCA2 genes. It is composed of 15 input features, 6 hidden layers of 275 hidden nodes each. Activation function relu on each hidden layers. Sigmoid function is used for the output layer. Adam is the optimizer.

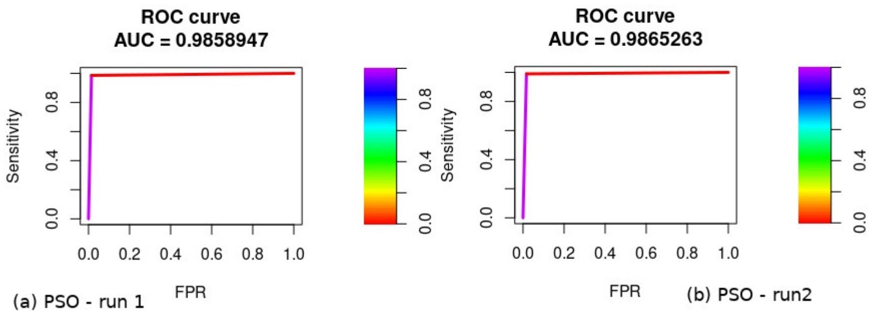


Figure 13. Model performance was evaluated using the receiver operating characteristic (ROC) curve for PSO run 1 and PSO run 2. AUC is 0.9858 for PSO run 1 and 0.9865 for PSO run 2.

3.4. Feature Importance

Feature importance values indicate which input variable had the biggest impact on each prediction that was generated by the classifier model. The goal of feature importance is to determine which input features are important for the prediction. It gives a score for each feature of the data. A higher score means that the specific feature will have a larger effect on the model that is being used to predict a certain variable. The feature importance of each model is described in Figure 14.

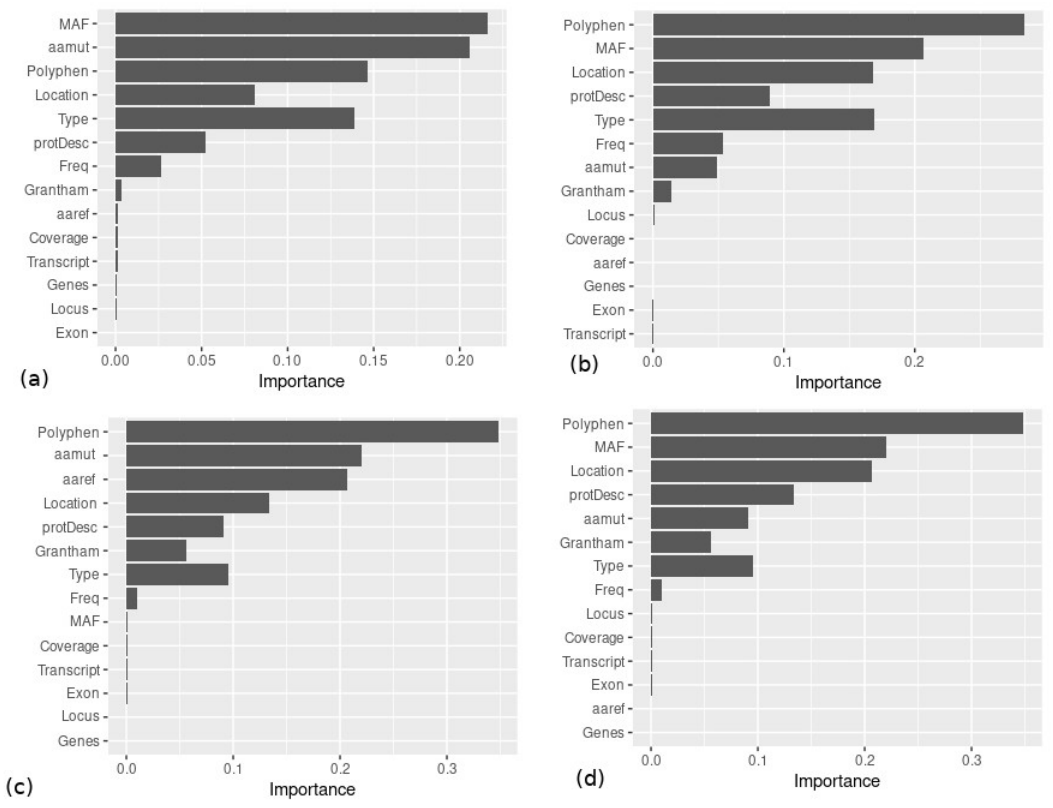


Figure 14. Feature importance for the 4 models found with the metaheuristic algorithm. (**a**) Feature importance for GA run 1 composed of 6 hidden layers with 297 hidden nodes. (**b**) Feature importance GA run 2 composed of 6 hidden layers [212:134:246:285:61:196] hidden nodes. (**c**) Feature importance PSO run 1 composed of 6 hidden layers with 275 hidden nodes. (**d**) Feature importance PSO run 2 composed of 6 hidden layers and [45:156:132:284:105:169] hidden nodes. Deep neural network incorporates an automatic feature selection process and our model PSO run 1 considers the variables Polyphen, aaref and aamut as being the most important and does not take into account the gene variable in input.

3.5. Prediction on Repair Genes

To test the performance of the trained classifier, we applied it to some new genes, more precisely on repair genes present in our homologous recombination repair custom panel. Indeed, according to the feature importance (Figure 14c), we can observe that Genes and Transcript are variables few important in the prediction, whereas Polyphen, aaref, and aamut are important variables for the pathogenicity prediction. We compared the results of the classifier with the biologist's decision. A sample of this test run is available in Table 11 (see Supplementary Materials for the full list).

On 11 samples, we tested our classifier obtained with PSO (run 1) on a list of 290 mutations on the genes: ARID1A, ATM, PALB2, PICK3CA, BRIP1, BRAF, NBN, TP53, RAD51C, RAD51D, CDK12. The classifier had a success rate of 96.55% on mutations never seen by the deep neural network. He was wrong 10 times on 290 mutations. The occurrence ARIDA1A: c.3999_4001del was considered pathogenic with a probability of 87% on his decision. This mutation is known to be benign as it is an equivalent deletion and was present 6 times in the test list. He correctly classified probable polymorphism mutations with a high probability in his decision. However, he missed 4 pathogenic mutation variants: TP53:c.645T>G (p.Ser215Arg), CHEK2: c.1064T>C (p.Leu355Pro), ATM:c.476T>A (p.Ile159Lys) and a PIK3CA:c.3140A>G (p.His1047Arg). A mutation in the splice site BRIP1:c.206-2T>A was missed the first time by the biologist but it was correctly identified as pathogenic by the classifier with high probability (94%).

Table 11. Example of prediction of the deep neural network obtained with PSO (run 1) on repair genes.

Genes	Coding	Protdesc	isMut	PSO1	Pb Benign	Pb Pathogenic	Comment
ARID1A	c.1351-22A>C	p.?	0	0	1	0	Benign
ARID1A	c.2420-18G>C	p.?	0	0	1	0	Benign
ARID1A	c.3999_4001del	p.Gln1334del	0	1	0.04	0.96	Benign
ATM	c.72+36TAA>T	p.?	0	0	1	0	Benign
ATM	c.902-18T>A	p.?	0	0	0.73	0.27	Benign
ATM	c.1607+47GAC>G	p.?	0	0	1	0	Benign
ATM	c.1809_1810dup	p.Pro604fs	1	1	0	1	Pathogenic
ATM	c.1810C>T	p.Pro604Ser	0	0	1	0	Benign
ATM	c.2119T>C	p.Ser707Pro	0	0	1	0	Benign
ATM	c.3078-77C>T	p.?	0	0	1	0	Benign
TP53	c.645T>G	p.Ser215Arg	1	0	0.72	0.28	Pathogenic
ATM	c.79G>A	p.Val27Ile	0	0	1	0	Benign
PALB2	c.3114-51T>A	p.?	0	0	1	0	Benign
PALB2	c.1684+42TGAA>A	p.?	0	0	1	0	Benign
BRIP1	c.3411T>C	p.(=)	0	0	1	0	Benign
BRIP1	c.206-2TA>A	p.?	0	1	0.06	0.94	Pathogenic
NBN	c.1124+18C>T	p.?	0	0	1	0	Benign
NBN	c.553G>C	p.Glu185Gln	0	0	1	0	Benign
NBN	c.381T>C	p.(=)	0	0	1	0	Benign
ARID1A	c.854del	p.Gly285fs	1	1	0	1	Pathogenic

Example of prediction of the deep neural network obtained with PSO (run 1) on a sample of genes outside of training. A list of the 290 mutations is available in the supplementary data section. Lines highlighted in green show the concordance between the biologist's decision and the neural network prediction. By contrast, the color red denotes the discrepancy.

4. Discussion

We aimed to use metaheuristic algorithms to find the optimal fully connected deep learning neural network architecture for predicting BRCA1/BRCA2 pathogenicity. We developed an accurate and efficient model based on the best architecture returned by the PSO algorithm (see Figure 12). Our model can predict BRCA1/BRCA2 pathogenicity with a precision of 99.65% and AUC of 98.6%. Moreover, our tests have revealed the capacity of our model to predict pathogenicity on other genes like the repair genes. Deep learning (DL) incorporates an automatic feature selection process [19] and our model considers the variables aaref and aamut as being the most important and does not take into account the gene variable in input (Figure 14). DL approaches applied to different cancer-related prediction tasks performed better than with other ML approaches that require a feature selection step [20]. Nevertheless, there are few deep neural network techniques established to improve the BRCA1/BRCA2 prediction problem. Training a DL network is an optimization problem [21].

Metaheuristic algorithms were able to define an optimal architecture for predicting the pathogenic character in the BRCA1 and BRCA2 genes. The architecture proposed by the PSO algorithm is equally effective at accurately predicting genes by taking into account the change of a reference amino acid and the mutated one. We have developed a machine learning random forest algorithm to predict variant mutations in colorectal, melanoma, glioma, and lung cancer [22] but the model revealed its limits in the prediction of more complex genes. This is why we are oriented towards the DL. As we stated in the introduction, finding an optimal architecture is up to the challenge. The grid search method remains suitable for small neuron networks, but it very quickly shows its limits. The number of combinations becomes exponential with the number of hyperparameters to be tested. For this reason, we are oriented toward metaheuristic algorithms to define an optimal architecture of the neural network to predict the pathogenicity of the BRCA1 and BRCA2 genes. By comparing the models obtained, GA showed an improvement in results with a precision of 98.96% (Table 10). This approach has the advantage of being relatively simple to set up because it uses very intuitive parameters that allow us to obtain good results.

With the genetic algorithm, we noticed that the probability to apply mutation varies, but if we use exactly 10% of probability to apply mutation on each Chromosome parameter. 10% of probability works well in a population of 20 individuals. If we deplete a more considerable population, we have to reduce the probability. Under a 100 individuals population, consider choosing between 5% and 2% of probability. The latter approach was PSO was used twice, just like the algorithm genetics, using the results of the first run to modify the parameters of the second run. The advantage of the genetic algorithm and the particle swarm optimization remains their ability to achieve the optimal solution from a limited population. Thanks to this, we can reduce the computing time required to provide the most efficient neural network. PSO gave us the best neural network architecture with a configuration of 6 hidden layers, each composed of 275 hidden nodes (Figure 12). Accuracy on the validation set 98.52% and AUC of 0.98. In addition, the algorithm takes into account in its prediction the change in amino acids, which allows us to generalize the predictions on other genes like repair genes. With GA and PSO, it is possible to vary more parameters (like the number of neurons in each layer rather than making uniform layers). Finding neural networks with better capabilities obtained by optimization can rely on more information to predict the outcome as described in Figure 14. These networks are effective at avoiding errors on ambiguous data. In general, the results show that some optimization scenarios are better suited to one method versus the other. Particle swarm optimization performs better in some cases, while genetic algorithms perform better in others, which implies that the two methods traverse the problem hyperspace differently [23]. The results of GA and PSO implementations in the deep neural network optimization problem show that the PSO algorithm is superior in finding the optimal solution in terms of accuracy and iteration. In addition, the PSO algorithm is also superior to the simplicity of the techniques used. The main advantages of PSO are that it is a simple concept, easy implementation, robustness to control parameters, and computational efficiency when compared with mathematical algorithms and other heuristic optimization techniques. Deep learning has achieved high performance for numerous types of cancers. For breast cancer detection, Khan and al [24] demonstrated accuracy up to 98.42%. A wide variety of deep neural network approaches have been developed to improve patient diagnosis in oncology. They use a combination of genomics, transcriptomics, or histopathology data. The purpose of diagnosing, prognosis, and treatment selection. Deep learning is not to replace humans, but to provide decision support tools that assist cancer researchers to study the disease and health professionals in the clinical management of people with cancer [25]. The two GA and PSO packages that we have developed in R language for Keras can be re-adapted according to the problem to be resolved. A new training data phase has to be performed to obtain a deep neural network architecture according to your problem to be resolved.

5. Conclusions

Our model can predict BRCA1/BRCA2 pathogenicity with a Precision of 99.65% and AUC of 98.65%. Moreover, our tests have revealed the capacity of our model to predict pathogenicity on other genes like the repair genes. DL approaches applied to different cancer-related prediction tasks performed better than those with other ML approaches that require a feature selection step. The architecture proposed by the PSO algorithm is equally effective at accurately predicting genes by taking into account the change of a reference amino acid and the mutated one. As we stated in the introduction, finding an optimal architecture is up to the challenge. The grid search method remains suitable for small neuron networks, but it very quickly shows its limits. GA showed an improvement in results with a precision of 98.96%. This approach has the advantage of being relatively simple to set up. The advantage of the genetic algorithm and the particle swarm optimization remains their ability to achieve the optimal solution from a limited population. In addition, the algorithm takes into account in its prediction the change in amino acids, which allows us to generalize the predictions on other genes like repair genes. With GA and PSO, it is possible to vary more parameters (like the number of neurons in each layer rather than making uniform layers). These networks are effective at avoiding errors on ambiguous data. In general, the results show that some optimization scenarios are better suited to one method versus the other. In addition, the PSO algorithm is also superior to the simplicity of the techniques used. The two GA and PSO packages that we have developed in the R language for Keras can be re-adapted according to the problem to be resolved. They can easily be implemented in other molecular biology laboratories and become a true aid in routine NGS analysis interpretation. After implementation in another laboratory, performance will still need to be evaluated to see if the model generalizes to the new conditions, which may involve, for example, different protocols and samples of different characteristics. The article focused on the optimization of a fully connected deep neural network with metaheuristic algorithms. There are various types of deep learning. For example, the recurrent neural network (RNN), the convolutional neural network (CNN), long short term memory (LSTM), deep belief networks (DBN)... It would be interesting to optimize these other types of neural networks with PSO and GA algorithms for other applications in the medical field.

Supplementary Materials: The following are available online at https://www.mdpi.com/article/10.3390/biomedinformatics2020016/s1.

Author Contributions: E.P. wrote the manuscript with the help of L.O. and E.P. designed and conceived the idea. T.B. coded and implemented the packages, genetic algorithms, and particle swarm optimization for deep learning for predicting variants in the R language. E.P. and T.B. were involved in data acquisition and cleansing. I.N.-M., L.O., C.C. and C.P. performed the analysis and interpretation of all NGS variant calls. A critical review of the manuscript for important intellectual content: L.O., N.A., C.C., I.N.-M., N.B., C.P. and P.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Training database is available on demand.

Acknowledgments: We would like to thank the team of technicians (Veronique Brice, Marie-Christine Courtier, Annick Durand, Cindy Fernandez-Muniz, Christine Schiano, Christelle Taillandier, Corinne Terlier, Emmanuelle Toussaint) for their involvement and the work they do daily in the Onco-Biology laboratory. Without them, this project would not have been accomplished.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hope, T.; Resheff, Y.S.; Lieder, I. Learning Tensorflow: A Guide to Building Deep Learning Systems; O'Reilly Media, Inc.: Newton, MA, USA, 2017.
2. Carvalho, M.; Ludermir, T.B. Particle swarm optimization of neural network architectures andweights. In Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS 2007), Kaiserslautern, Germany, 17–19 September 2007.
3. Qolomany, B.; Maabreh, M.; Al-Fuqaha, A.; Gupta, A.; Benhaddou, D. Parameters optimization of deep learning models using particle swarm optimization. In Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017.
4. Idrissi, M.A.J.; Ramchoun, H.; Ghanou, Y.; Ettaouil, M. Genetic algorithm for neural network architecture optimization. In Proceedings of the 2016 3rd International Conference on Logistics Operations Management, Fez, Morocco, 23–25 May 2016.
5. Brownlee, J. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*; Machine Learning Mastery: Melbourne, Australia, 2018.
6. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
7. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995.
8. Adyatama, A. RPubs-Introduction to Particle Swarm Optimization. Available online: https://rpubs.com/argaadya/intro-pso (accessed on 12 October 2021).
9. Dagan, T.; Talmor, Y.; Graur, D. Ratios of Radical to Conservative Amino Acid Replacement are Affected by Mutational and Compositional Factors and May Not Be Indicative of Positive Darwinian Selection. *Mol. Biol. Evol.* **2002**, *19*, 1022–1025. [CrossRef] [PubMed]
10. Grantham, R. Amino acid difference formula to help explain protein evolution. *Science* **1974**, *185*, 862–864. [CrossRef] [PubMed]

11. Blum, A. *Neural Networks in C++ an Object-Oriented Framework for Building Connectionist Systems*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1992.
12. Boger, Z.; Guterman, H. Knowledge extraction from artificial neural network models. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; Volume 4.
13. Swingler, K. *Applying Neural Networks: A Practical Guide*; Morgan Kaufmann: Burlington, MA, USA, 1996.
14. Linoff, G.S.; Berry, M.J. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
15. Heaton, J. *Introduction to Neural Networks with Java*, 2nd ed.; Heaton Research, Inc.: St. Louis, MO, USA, 2008.
16. Beysolow, T., II. *Introduction to Deep Learning Using R: A Step-by-Step Guide to Learning and Implementing Deep Learning Models Using R*; Apress: New York, NY, USA, 2017.
17. Mitchell, M. *An Introduction to Genetic Algorithms*; MIT Press: Cambridge, MA, USA, 1998.
18. Gómez, N.; Mingo, L.F.; Bobadilla, J.; Serradilla, F.; Manzano, J.A.C. Particle Swarm Optimization models applied to Neural Networks using the R language. *WSEAS Trans. Syst.* **2010**, *9*, 192–202.
19. Albaradei, S.; Thafar, M.; Alsaedi, A.; Van Neste, C.; Gojobori, T.; Essack, M.; Gao, X. Machine learning and deep learning methods that use omics data for metastasis prediction. *Comput. Struct. Biotechnol. J.* **2021**, *19*, 5008–5018. [CrossRef] [PubMed]
20. Fakoor, R.; Ladhak, F.; Nazi, A.; Huber, M. Using deep learning to enhance cancer diagnosis and classification. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
21. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
22. Pellegrino, E.; Jacques, C.; Beaufils, N.; Nanni, I.; Carlioz, A.; Metellus, P.; Ouafik, L.H. Machine learning random forest for predicting oncosomatic variant NGS analysis. *Sci. Rep.* **2021**, *11*, 1–14. [CrossRef] [PubMed]
23. Boeringer, D.W.; Werner, D.H. A comparison of particle swarm optimization and genetic algorithms for a phased array synthesis problem. In Proceedings of the IEEE Antennas and Propagation Society International Symposium. Digest. Held in conjunction with: USNC/CNC/URSI North American Radio Sci. Meeting (Cat. No.03CH37450), Columbus, OH, USA, 22–27 June 2003.
24. Khan, S.; Islam, N.; Jan, Z.; Din, I.U.; Rodrigues, J.J.P.C. A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognit. Lett.* **2019**, *125*, 1–6. [CrossRef]
25. Walsh, S.; de Jong, E.E.C.; van Timmeren, J.E.; Ibrahim, A.; Compter, I.; Peerlings, J.; Sanduleanu, S.; Refaee, T.; Keek, S.; Larue, R.T.H.M.; et al. Decision support systems in oncology. *JCO Clin. Cancer Inform.* **2019**, *3*, 1–9. [CrossRef]