Computers in Biology and Medicine 142 (2022) 105205

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/compbiomed

# CNN architecture optimization using bio-inspired algorithms for breast cancer detection in infrared images

Caroline Barcelos Gonçalves *, Jefferson R. Souza, Henrique Fernandes

*Faculty of Computing, Federal University of Uberlandia, 2121, Joao Naves de Avila Avenue, Uberlandia, 38408-100, MG, Brazil*

ARTICLE INFO

*Keywords:*
Breast cancer
Infrared images
CNN
GA
PSO
Bio-inspired optimization

ABSTRACT

The early detection of breast cancer is a vital factor when it comes to improving cure and recovery rates in patients. Among such early detection factors, one finds thermography, an imaging technique that demonstrates good potential as an early detection method. Convolutional neural networks (CNNs) are widely used in image classification tasks, but finding good hyperparameters and architectures for these is not a simple task. In this study, we use two bio-inspired optimization techniques, genetic algorithm and particle swarm optimization to find good hyperparameters and architectures for the fully connected layers of three state of the art CNNs: VGG-16, ResNet-50 and DenseNet-201. Through use of optimization techniques, we obtained F1-score results above 0.90 for all three networks, an improvement from 0.66 of the F1-score to 0.92 of the F1-score for the VGG-16. Moreover, we were also able to improve the ResNet-50 from 0.83 of the F1-score to 0.90 of the F1-score for the test data, when compared to previously published studies.

## 1. Introduction

According to Ref. [1], cancer is an abnormal cell growth that becomes out of control. Breast cancer is one of the cancer types that starts in the breast and mainly affects women, but it can also appear in men [1]. The American Cancer Society in Ref. [2] estimates that in 2021 more than 280,000 new cases of breast cancer will be diagnosed in the USA. They also reported that the incidence of breast cancer had increased 0.5% over recent years [2]. In Brazil, the number of estimated new cases of breast cancer for 2021 is over 66,000 [3]. After skin cancer, breast cancer is the most common both in Brazil and the USA [2,4]. Due to its prevalence, early diagnosis of breast cancer is essential, especially when it comes to improving the chances of patient survival and recovery [5].

There are many imaging techniques used to support the early detection of breast cancer. Among such, mammography is considered the standard exam. One of its disadvantages is that mammography comes with low doses of X-ray, which are harmful to patients. Although the mammography is the standard imaging technique, it does not demonstrate good sensitivity in breasts with denser tissue, which is common in younger woman [3,6]. Therefore, in such cases, other imaging technique might also be used to support the diagnosis, for example MRI or ultrasound [6]. Infrared thermography is another imaging technique that can be used in the support of early detection of breast cancer, as it has recently shown potential in providing support to the diagnosis of this disease [7–9].

Thermography is an imaging technique that measures the temperature emitted by the human body through infrared radiation [7,8]. This imaging technique is useful for the early detection of breast cancer, as tumor areas are hotter than healthy areas due to higher metabolic activity of the cancerous cells [8,10,11]. Thermography is also a cheap, simple and noninvasive procedure [8].

Convolutional Neural Networks (CNNs) are one of the deep neural networks that have shown great potential in solving computer vision problems (recognition and classification tasks for example) performing, in some cases, even better than human experts [12–14]. This technique is also used in many medical imaging application, including breast cancer detection, and more specifically when using thermography images [7,8,11,15].

Developing a CNN involves optimizing many parameters, as well as choosing its architecture. Choosing the suitable parameters is essential reaching fitting results with CNNs. As such, it is not a simple task, as it demands a high level of expertise. In addition, it is common to experiment with multiple parameters in order to find the most appropriate, which also makes it time consuming [12–14].

Population based algorithms, such as genetic algorithms (GA) and

* Corresponding author.
*E-mail address:* caroline.goncalves@ufu.br (C.B. Gonçalves).

https://doi.org/10.1016/j.compbiomed.2021.105205
Received 1 November 2021; Received in revised form 28 December 2021; Accepted 29 December 2021
Available online 5 January 2022
0010-4825/© 2022 Elsevier Ltd. This article is made available under the Elsevier license (http://www.elsevier.com/open-access/userlicense/1.0/).

particle swarm optimization (PSO), present good results especially in problems that have a broad search space [16]. GA and PSO are two bio-inspired meta-heuristic algorithms that can be used in optimization problems [12,13]. In our study, we use algorithms of this type in the context of optimizing some of the parameters of our CNNs. In our study, we are using transfer learning with the following state of the art CNNs: VGG-16, ResNet-50 and DenseNet-201. Each CNN has its entire fully connected layers replaced and its architecture chosen by the PSO or GA. In addition, some optimization algorithms are also used to choose some of the network hyperparameters.

## 2. Related work

In this section, we are going to review some literature on related studies that use machine learning techniques for breast cancer detection, through the application of thermography images and some bio-inspired algorithms, specifically those of GA and PSO, which are applied when searching for adequate CNNs architectures and hyperparameters.

The study developed by Torres-Galván et al. in Ref. [15] uses images from three different infrared databases: DMR-IR, *Instituto Jaliciense de Cancerología (IJC)* and *Instituto de Seguridad y Servicios Sociales de los Trabajadores del Estado (ISSSTE)* with a total of 311 images, in which 267 was healthy and 44 sick. Transfer learning using ResNet-101 was the chosen technique for the classification. The authors in Ref. [15] applied random data augmentation techniques to the training data, for instance: data rotation on angles between 0° and 359°, reflection on each axis and translation on angles between 0° and 50°. In addition, they augmented the data by a factor of 67. The authors report on two different setups for the experiments: an unbalanced dataset keeping the original data proportion and a balanced dataset each with different hyper-parameters, with both using data augmentation. The authors find a performance of 84.6% sensitivity using the unbalanced dataset. Moreover, with the balanced dataset, they obtained even higher results with 92.3% of sensitivity.

The DMR-IR database proposed by Silva et al. is found in Ref. [17]. These authors created a database of infrared breast images collected at the *Hospital Universitário Antônio Pedro (HUAP)* of the Fluminense Federal University, which has healthy patients, along with those patients with breast cancer. The authors collected thermography images from two different protocols static and dynamic. For each static patient, approximately 5 poses (images) were made available: frontal, lateral left and right on two angles: 45° and 90°. For the dynamic images, 20 images were collected over a period of 5 min from the same position, frontal with their own proposed protocol. Moreover, two extra images were collected from lateral positions at 90°. With both protocols, the authors performed a patient acclimation process before collecting the data. In the paper, the authors state that the database possessed images from 141 patients with 3534 images. However, the database has increased since the publication of the paper. The database also contains mammography images.

Research conducted by Zuluaga et al. in Ref. [7] presents a study with many different state-of-the-art CNNs, using a fine-tunning approach (ResNet-50, SeResNet-50, SeResNet-18, SeResNet-34, VGG-16, Inception-V3, Inception-ResNet-V2 and Xception). Moreover, the authors propose a surrogate model based on a tree parser estimator optimization in order to find better CNNs architecture. Finally, they investigate the impact of data augmentation in the classification. In their study, 57 patients from the dynamic protocol of DMR-IR database were used, in which 19 were healthy and 38 sick. Next, the pre-processing of the images was performed in order to extract the region of interest (ROI). In addition, cropping, resizing and normalization was performed across all images. For the data augmentation, the authors used vertical and horizontal flips, rotation of angles between 0 and 45°, 20% zoom and normalized noise. In the optimization flow, the authors defined the following hyperparameters: number of blocks, convolutional layers and filter, type of optimizer, kernel and pooling layer size, dropout rate, batch normalization, number of dense units and top layer type. The optimized model is chosen based on the F1-score results. The best result obtained by the authors using the state-of-the-art CNNs is 0.91 of the F1-score using the SeResNet-18. The CNN obtained from the surrogate model was the best result reported by the authors with 0.92 of the F1-score.

The study of Farooq and Corcoran [18] elucidates many different healthcare applications of thermography, which include cancer detection (breast cancer, skin cancer and brain cancer) and diabetes detection, for example. These authors also propose a computer aided diagnoses (CAD) system using the dynamic images of the database DMR-IR and the CNN Inception-V3. The authors used images from 40 patient, 18 sick patients and 22 healthy patients, these were then divided into 70% for training, 20% for validation and 10% for test. Their image pre-processing approaches, used by the authors, included a sharpening filter and a histogram equalization applied to training and validation data. The CNN hyperparameters involve 5000 epochs, learning rate of 0.001 and an SGD optimizer. The proposed CAD obtained an accuracy of 80% and an F1-score of 76.89%.

The study by Baffa et at. in Ref. [8] proposed a new CNN for the classification of thermography images from the DMR-IR database. The authors used both static and dynamic images. For the static images, they used 300 images, 174 images from healthy patients and 126 images from sick ones, whereas for the dynamic images, they used 137 patients, 95 healthy and 42 sick. For both setups, data augmentation was applied to the sick class as a way of balancing the database. These authors also tested four different approaches for dealing with dynamic images, which involves, for example, creating a new image with the mean of all the available images and creating a new image with the difference found between the first and the last image. The CNN proposed by the authors has two convolutional layers (size of 5x5 and 32 outputs); followed by two max pooling layers also 5x5, a stride of 3 and a fully connected layer. They also tested color and gray scale images, and find out that the color images performed better. Their best result for static images was 98% of accuracy with color images and for the dynamic images 95% of accuracy with the color images.

The study by Cabioglu et at. in Ref. [11] used the pre-trained AlexNet to classify 181 static images from the DMR-IR database. The authors used 147 images from healthy individuals and 34 images from individuals which have cancer. Their experiments were performed using both an unbalanced and a balanced dataset (using data augmentation techniques for the sick class in order to obtain a more balanced database). The authors also tested two different types of data preprocessing. The first considers the thermography images as gray scale images, which makes a channel replication to obtain a 3-channel image. The second involves converting the temperature matrix into RGB images based on the Matlab® Jet Colormap. The authors also tested the overwriting of different parts of the CNN, for example: all the fully connected layers or some of the last convolutional layers in order to verify how the CNN would perform. The best result, obtained by these authors, is 94.3% of accuracy, which was obtained with Jet Colormap preprocessing, balanced dataset and the changing of the entire fully connected layers.

The study developed by Mambou et al. in Ref. [19] uses a CNN (Inception-V3) associated with an support vector machine (SVM) to classify dynamic thermography images from the DMR-IR database. The data used is composed of 64 patients, 32 from each class, each patient with 20 images. The SVM is only used when the CNN output probability is between 0.5 and 0.6. The authors preprocessed the images to extract ROI and used a grayscale for conversion to RGB. Their CNN was trained with a learning rate of 0.000 1, 15 epochs, and 4000 steps. The result reported for classification is receiver operating characteristic (ROC) of 1 and precision recall of 1.

In Rosalidar et al. [20], the authors compare different fine-tuned CNNs (ResNet-101, DenseNet-201, MobileNet-V2, and ShuffleNet-V2) for the classification of thermography images from the database DMR-IR in cancer or healthy. Here, both static and dynamic images were

used. The static images were used in the training; however, the testing was done twice, once with other static images and another with the dynamic images. Moreover, for the training images, data augmentation was applied. The authors reported that DenseNet-201 was the best network, classifying both test sets with 100% accuracy. Furthermore, MobileNet-V2 also obtained very good results in a less demanding training time, with 100% accuracy for the static testing set and 99.6% accuracy over the dynamic tests.

Another study that also uses the DMR-IR and pre-trained CNNs, developed by Kiymet et al. can be found in Ref. [21]. The authors here used 144 patients in which 88 are healthy and 56 sick. Moreover, they used four different pre-trained CNNs, those being: VGG-16, VGG-19, ResNet-50 and Inception-V3. The images were scaled and converted to RGB as their preprocessing step. The best result in their work was obtained using the ResNet-50 with 88.89% of accuracy.

In the study by Chaves et al. [9], 88 DMR-IR static images were used (44 for each class), using five pre-trained CNNs: AlexNet, GoogleNet, ResNet-18, VGG-16, and VGG-19. The authors used all 5 static images from each patient, and applied the following data augmentation technique to the training images: horizontal and vertical flips, moving the images 30 pixels and resizing with a value between 0.9 and 1.1. The authors also tested a different number of epochs. The best result, from these authors, was obtained with VGG-16 and VGG-19, both attaining 77.5% accuracy, the first with 85% sensitivity and the second with 90% sensitivity.

Table 1 presents a summary of the related studies described herein. Next, we will review studies that proposed the use of bio-inspired algorithms, specifically GA and PSO, used to find CNN architectures.

The research conducted by Junior et al. in Ref. [12] proposes a particle swarm optimization (PSO) algorithm, denominated psoCNN as a tool that quickly defines good CNN architectures. This study describes in detail all the PSO operators, i.e. the particle definition, velocity calculation, position update and fitness evaluation. Their algorithm is used to find the entire CNN architecture, with all the convolutional layers, pooling layers and fully connected layers. The particle definition that the authors use allows for flexibility in the number of layers, as well as proposing how to compare different particle lengths. They use three different units to compose the particle: convolutional, pooling and fully connected layers, each one with its own parameters and adding the units together generates the particle, which represents a CNN architecture (with all the CNN restrictions). In order to validate their psoCNN, these authors use nine public databases: MNIST, MNIST - RD, MNIST - RB, MNIST - BI, MNIST - RD + BI, Rectangles, Rectangles-I, Convex, and MNIST - Fashion. The CNN model established by the psoCNN performs better than the state of the art models for six of the nine databases tested. Although these did not outperform all the state of the art model results, they were able to find very good results with smaller and simpler networks, which leads to faster conversion and are comparable to the state of the art models.

Research performed in Sun et al. in Ref. [13] also proposes a bio-inspired technique in order to find good CNN architectures automatically, by using a genetic algorithm called EvoCNN. The paper in Ref. [13] describes many of the genetic algorithm definition as the gene encoding, crossover and mutation operators, selection, fitness evaluation. Each gene is composed of a multitude of unit types. There are three unit types available: the convolutional layers unit (with the convolutional layers and hyperparameters associated to it), the pooling unit (which also encapsulates the polling layers parameters), and the fully connected unit. The definition of the gene delivered in Ref. [13] provides a flexible CNN structure. However, with the flexible GA individual, comes the complexity of the crossover, for which the authors also propose an algorithm. In order to evaluate the EvoCNN proposed, the authors tested it on nine different databases: Fashion, Rectangle, Rectangle Images (RI), Convex Sets (CS), MNIST Basic (MB), MNIST with Background Images (MBI), MNIST with Random Background (MRB), MNIST with Rotated Digits (MRD) MNIST with RD plus Background Images (MRDBI) compared to many different classifiers used in these datasets. The CNNs obtained by the EvoCNN outperform almost all of the classifiers compared in the stated datasets with the advantage of much smaller CNNs.

The study in Fidelis et al. in Ref. [22] uses a genetic algorithm to find comprehensive rules in two databases, a dermatology database and a breast cancer database both from the University of California Irvine (UCI) machine learning repository. In their study, the authors propose an individual encoding which provides a very flexible approach in turning on and off one rule, while also maintaining a simple approach for the crossover and other GA operators. Each individual is composed of many genes and each gene has three parts: weight, operator and value. The authors consider the weight above 0.3 as being present in the rule and below 0.3 as not present. Therefore, only changing the weight might mean turning on or off the rule. The dermatology database has five rules and the authors here generated rules for each of the rules with test fitness (*specificity*sensitivity*) between 1 and 0.78. For the breast cancer dataset, they generated two rules but with a low-test fitness of 0.36 and 0.39. Although their study does not use the GA in the context of CNN optimization, we were inspired by their individual representation and adapted it for our problem, as we will show in the next session.

## 3. Methodology

### 3.1. Database

The database used in this study is the DMR-IR, a public database proposed by Silva et al. [17] that possesses infrared images from two classes of patients healthy and with breast cancer, which we will call ”sick”. The dataset is composed of infrared images from two different

**Table 1**
Summary of related work.

| Reference | Database | Protocol | ML Technique | Best Results |
|---|---|---|---|---|
| [15] | Combined three databases | Static | ResNet-101 | Sensitivity: 84.6% for the unbalanced set and for the balanced set 92.6% |
| [7] | DMR-IR | Dynamic | ResNet-50, SeResNet, VGG-16 Inception-V3, Inception-ResNet-V2 Xception and CNN obtained from the optimization method | 0.92 of the F1-score with the optimized model |
| [18] | DMR-IR | Dynamic | Inception-V3 | F1-score of 76.89% |
| [8] | DMR-IR | Static and Dynamic | Proposed CNN | 98% accuracy with the static 95% accuracy with the dynamic. |
| [11] | DMR-IR | Static | AlexNet | 94.3% accuracy |
| [19] | DMR-IR | Dynamic | Inception-V3 and SVM | ROC of 1 |
| [21] | DMR-IR | Not mentioned | VGG-16, VGG-19, ResNet-50 and Inception-V3 | 88.89% accuracy with ResNet-50 |
| [20] | DMR-IR | Static and Dynamic | ResNet-101, DenseNet-201 MobileNet-V2 and ShuffleNet-V2 | 100% accuracy with DenseNet-201 |
| [9] | DMR-IR | Static | AlexNet, GoogleNet, ResNet-18, VGG-16, and VGG-19 | 77.5% accuracy with both VGG-16, and VGG-19 |

thermography protocols: static and dynamic. In the static protocol, the patient rests some minutes to complete the acclimatization process. Following this, an image is taken from 5 different positions (frontal, lateral left and right with 45° rotation and lateral left and right with 90° rotation), which is a total of 5 images per patient. For the dynamic protocol, 20 images are taken over a period of 5 min from only the frontal position. Additionally, two extra images were taken from the 90° lateral rotation. All the images are of size 640 x 480 px, and on to each pixel, a temperature value is associated.

Initially, we decided to focus on the static images. The subset of the static images is composed of 864 healthy images (176 different patients) and 193 images from sick patients (39 patients). There are still some patients with unknown classes, and as such were not used in the present study.

Although the DMR-IR is widely used in the literature, different studies use a different number of images, and do not specify the exact patient numbers used. As we also reported in our previous study [23], we decided to use the same number of patients reported by Ref. [11] (without the data augmentation they reported), even though we do not have the specifics for those patients used in the study. Therefore, we selected 34 sick patients and 147 healthy patients. Similar to that performed in Ref. [23]; in the present study, we are using only the frontal images. Hence, we also have 34 sick images and 147 healthy images.

Due to the fact that we have an unbalanced dataset, we used data augmentation to deal with this unbalance. The data augmentation is performed using rotation at a random angle between −30° and 30°, translation in X-axis and translation in the Y-axis. All the three techniques are randomly applied, thus a single input image is likely to produce different output images. We applied data augmentation to both classes, but with different proportions. For the healthy classes each original image generates one other image with data augmentation applied, whereas for the sick class, each original image generates 7 new images with data augmentation. In total, we have 294 healthy images and 272 sick images.

As also described in Ref. [23], we are using another subset from the database with a balanced number of original images, in which we use 38 patients for each class. In this subset, we are also only considering the frontal images. Therefore, we have 38 images from each class as well.

### 3.2. Image pre processing

The image preprocessing used in this study involves three steps: converting the 2D images of the dataset into an image with 3-channels, parse the temperature values to values between 0 and 1 and resize the image. All the preprocessing was necessary, due to the CNNs that we used. To use the CNNs, we must adapt our images to the input images the CNNs expect. For image resizing, we resized the images to obtain images with 224 x 224 px. For the other two steps, we tested two different approaches. The first combines a min-max conversion with channel replication. The min-max was performed in order to obtain an image with values between 0 and 1. Therefore, for each image we applied the min-max with the min and max of the image itself, which is shown in Equation (1).

$$\overline{X} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where $\overline{X}$ is the obtained value, X is the pixel temperature value, $X_{max}$ and $X_{min}$ are respectively the maximum and minimum values of the image.

After the min-max conversion, the channel replication is performed by taking the single channel and replicating it to obtain a 3-channel image. Therefore, all the 3-channels of the obtained image are equal.

The other approach is what we called the Jet Colormap approach. This approach is inspired on the study in Ref. [11]. Opening the original image in Matlab® and choosing the Jet Colomap generates a 3-channels image with the values in the desired range. Therefore, we processed the images to obtain a similar image to that generated thorough the colormap process. The processing procedure for this approach involves converting the original temperature image into a Matlab® index image and then converting this image to RGB. By use of this approach, we obtain an image with 3-channels, along with the pixel values in a range between 0 and 1.

### 3.3. CNNs

CNN is one of the main techniques used today for pattern recognition in images [7]. They also have been used in breast cancer detection, where it produced good results, as shown in the previous section. A typical CNN is represented in Fig. 1, which is an adaptation of the figure shown in Ref. [24]. Noteworthy here is that the CNNs are mainly divided in two parts, the convolutional part (which is composed of convolutional and pooling layers). The convolutional part, made up of convolutional layers, which are responsible for the feature extraction and the dropout layer, responsible for reducing the dimensions of the image [24]. The second part is the fully connected layer, which is the part that performs the classification through the data extracted by the preliminary part of the network [24].

Transfer learning is a technique that consists of taking a pre-trained model from a database and transferring that knowledge to a new database or domain [21]. Hence, transferring the learned features extracted beforehand. Many pre-trained CNNs are available in the literature so we decided to choose three that have shown promising results for breast thermography: VGG-16, ResNet-50 and DenseNet-201. We are maintaining the convolutional part of the CNNs we use as these are already provided, while only changing the fully connected part, which is the part that performs the classification.

#### 3.3.1. Weight initialization

In [25], Glorot et al. propose a new method for initializing the weights of a deep neural network known as ‘Glorot initialization’ or ‘Xavier initialization’, which is reported to provide faster convergence. The proposed method uses a uniform distribution for the weight initialization and can be defined as:

$$U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \tag{2}$$

where n is the layer size.

In this particular study, this method is used to initialize the weights of the fully connected layers of the CNNs.

### 3.4. Optimization

Due to the manual nature of the trial for finding a good fully connected layer architecture with a good learning rate, along with the time demanded to perform these experiments, we propose the use of optimization techniques for this process. The optimization techniques chosen are two bio-inspired algorithms: GA and PSO. For both algorithms, we considered the same parameters for optimization, which are:

- learning rate;
- number of fully connected layers;
- number of neurons of each fully connected layer;
- after which layers there is dropout;
- dropout rate of each dropout present in the layer;

Moreover, for both bio-inspired algorithms the range of the optimized parameters are the same. The value used to compose the learning rate (LR) is defined as an integer value (lr) along the range [1,6], which in the conversion to the CNN is used as $10^{lr}$. The number of neurons in a specific layer is defined as a value in a range of [3,10], called fc, which is converted as $2^{fc}$ for use in the definition of the fully connected CNN layer. Finally, we have a float value in the range of [0, 0.6] for the dropout rate of a layer. Furthermore, the number of fully connected

layers are implicit arguments, as well as, after which fully connected layer there is a dropout layer, which means they are not explicit as algorithm arguments. Instead, they are consequences of the algorithm individual or particle definition.

These two bio-inspired optimization techniques are used in this study for finding a good architecture for the fully connected layers with a good value for the learning rate. Next, we describe both the GA and PSO approaches.

#### 3.4.1. Genetic algorithm

In Algorithm 1, the GA algorithm we use is presented and in the following, its underlying approach is described in detail.

**Algorithm 1**.
GA algorithm.

The first important definition of the GA is how the individual is going to be defined. Inspired by Ref. [22], a model which reached flexibility for the GA individual with standard crossover operators, we decided to also use a fixed length individual with a flag indicating the turning on or off of that specific chromosome. Fig. 2 shows how we define our individual. Each individual is constituted of 12 chromosomes, and each chromosome is composed of two parts. A flag is incorporated to indicate if that chromosome is enabled or not (the 'Is present?' in this figure), as well as a value that indicates the value of that specific layer or parameter. As we use a fixed individual length, the fist chromosome defines the learning rate. From the second chromosome until the 11th, the fully connected layers as well as the dropouts are defined, where the even chromosomes in this range define the fully connected (FC) layers and the odd chromosomes define the dropout layers. Furthermore, in order for

**Algorithm 1** GA algorithm

```
population ← initializePopulation(populationSize)
fitness ← []
for i < populationSize do
    fitness[i] ← calcIndividualFitness(population[i])
end for
population, fitness ← sortDecreasing(population, fitness)
bestIndividual ← population[0]
bestFitness ← fitness[0]
iteration ← 0
for iteration < generations do
    selectedParents1 ← selectionTour(population, fitness, tourSize)
    selectedParents2 ← selectionTour(population, fitness, tourSize)
    childrenPopulation ← []
    for i < selectedParents1.length do
        child1, child2 ← crossover(selectedParents1[i], selectedParents2[i])
        childrenPopulation+ = child1
        childrenPopulation+ = child2
    end for
    childrenPopulation ← applyMutation(childrenPopulation)
    childrenFitness ← []
    for i < childrenPopulation.length do
        childrenFitness[i] ← calcIndividualFitness(childrenPopulation[i])
    end for
    childrenPopulation+ = bestIndividual
    childrenFitness+ = bestFitness
    childrenPopulation, childrenFitness ←
    sortDecreasing(childrenPopulation, childrenFitness)
    population, fitness ← orderedReinsertion(childrenPopulation, childrenFitness)
    bestIndividual ← population[0]
    bestFitness ← fitness[0]
end for
```

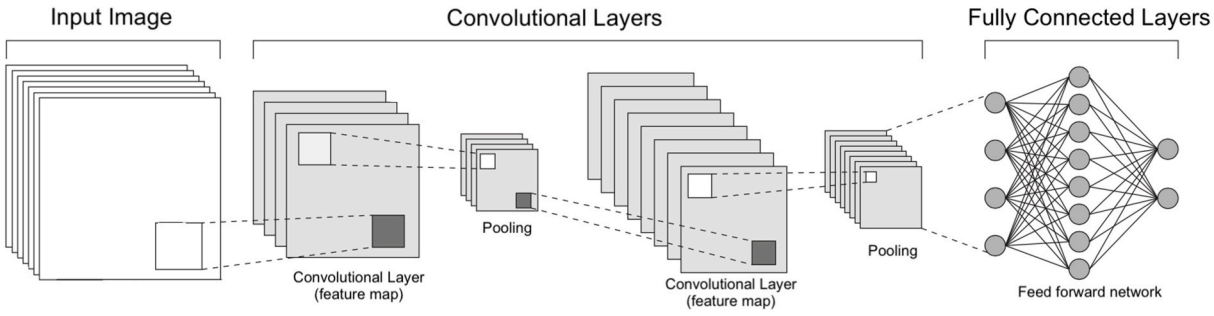


**Fig. 1.** CNN example - image adapted from Baldominos [24].

| GA individual representation | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Type layer | LR | $FC_1$ | $D_1$ | $FC_2$ | $D_2$ | ... | $FC_5$ | $D_5$ | $FC_6$ |
| Is present? | 1 | 0 or 1 | 0 or 1 | 0 or 1 | 0 or 1 | ... | 0 or 1 | 0 or 1 | 1 |
| Value | LR ∈ [1,6] | FC ∈ [3,12] | D ∈ [0, 0.6] | FC ∈ [3,12] | D ∈ [0, 0.6] | ... | FC ∈ [3,12] | D ∈ [0, 0.6] | $N_{DC}$ ∈ [3,12] |

**Fig. 2.** GA individual definition.

the individual to correspond to a valid CNN, in our particular problem, the last chromosome is always an FC layer with a value of 1, as $2^1$ is the CNN output.

Moreover, a consequence of our representation is that if a chromosome from the FC layer has the 'Is present?' equals to 0, the following dropout chromosome is just ignored, as we use the Dropout chromosome only for the immediate predecessor in the FC layer. It is also worthy of note that in the GA, the number of fully connected layers is implicit to the number of chromosomes in the FC layer that have 1 present for the 'Is present?' flag.

In order to verify how well the CNN with the GA definition for the FC layers architecture behaves, it is necessary to train the CNN. Therefore, in order to calculate the fitness of each individual, a CNN is trained. The fitness of the individual is actually the F1-score of the validation set. We noted that some individuals had a non-decreasing number of neurons (where the number of neurons in layer i was smaller than the number of neurons i+1). As the procedure of increasing the number of neurons in the FC part of the CNN from one layer to the next is not common in the literature, we decided to penalize architectures with this behaviour by a factor of 0.7 (empirically chosen). For instance: an individual which maps to an FC layer of 256 -> 256 -> 1024 will be penalized, whereas an FC layer that maps to 1024 -> 1024 -> 256 will not be penalized. Hence, the fitness is defined as:

$$fitness = \begin{cases} 0.7 * \text{F1} - \text{score of validation}, & \textbf{non} - \textbf{decreasing} \\ & \text{number of neurons} \\ & \text{in each layer} \\ \text{F1} - \text{score of validation}, & \textbf{non} - \textbf{increasing} \\ & \text{number of neurons} \\ & \text{in each layer} \end{cases} \quad (3)$$

After choosing te individual representation and the fitness calculation, it is also necessary to define the selection, crossover, mutation and reinsertion operators. Due to the fact that we have a fixed individual length, we decided to use a single crossover point, where two parents generate two children. A position *Cp* is randomly selected and the first child inherits the chromosomes before *Cp* from the first parent and after *Cp* from the second parent. The second child is the opposite (inheriting the first part from the second parent and the second part from the first parent). Fig. 3 shows an example of a crossover.

Another important GA operator that we defined was how the individuals will be selected for the crossover. We elect the tour as the selection method, as it allows for the changing of the selective pressure of the GA by changing the tour size. Therefore, with this approach we can test different tour sizes to validate which one is best suited to our problem. The tour selection algorithm depends on a value, which we call 'tourValue'. We randomly select 'tourValue' individuals from the entire population and the individual with the best fitness is the one selected for the crossover. Hence, for each pair of parents for the crossover, we use the tour selection twice, one for each parent.

The mutation is performed by changing a value from the individual. This value can be from the 'Is present?' part or the actual value part. We randomly select a chromosome to be mutated, and then the part that is going to be mutated. The new value is generated from the specific range of the part of the chromosome. Therefore, if the 'Is present?' is selected to be mutated, it is only changed from 0 to 1 or vice versa. In terms of the values, a new value is randomly generated from the range for that position. Fig. 4 shows an example of a mutation of an individual. In the image, the red values are the values that were changed.

Finally, for reinsertion, we use order reinsertion, keeping the best

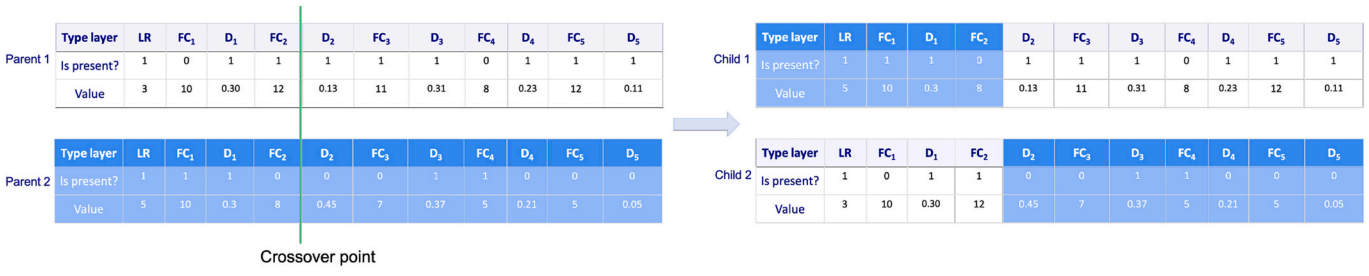


**Fig. 3.** GA crossover.

| Type layer | LR | $FC_1$ | $D_1$ | $FC_2$ | $D_2$ | $FC_3$ | $D_3$ | $FC_4$ | $D_4$ | $FC_5$ | $D_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Is present? | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Value | 3 | 10 | 0.30 | 12 | 0.13 | 11 | 0.31 | 8 | 0.23 | 12 | 0.11 |

| Type layer | LR | $FC_1$ | $D_1$ | $FC_2$ | $D_2$ | $FC_3$ | $D_3$ | $FC_4$ | $D_4$ | $FC_5$ | $D_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Is present? | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| Value | 2 | 10 | 0.30 | 12 | 0.4 | 11 | 0.31 | 8 | 0.23 | 3 | 0.11 |

**Fig. 4.** GA individual mutation.

individual from one generation to another.

### 3.4.2. PSO algorithm

The algorithm for the PSO is described in Algorithm 2, with details of its approach being given in the following.

**Algorithm 2.**
PSO.

In order to use the PSO, many definitions are needed, which include the particle definition, the particle evaluation, how the difference of two particles should be calculated, how to update the particle velocity and finally, the particle position. Although the PSO optimizes the same parameters as the GA, we use a different representation for the particle than that used for the GA individual. Most of the PSO definitions were inspired on the study in Ref. [12].

Our particle is composed of different pieces that we call fragments. Each fragment has two parts: type and value. The type indicates the category of the fragment, which can be LR, FC or dropout. The value stores the actual value of that fragment type, this can be an integer in a range of [1,6] for the learning rate, an integer value in a range of [3,10] for the number of neurons in a specific layer and a float value in a range of [0, 0.6] for the dropout rate of a layer. The value of fragments that has type LR or FC stores the exponent that is used to find the actual numbers, for the LR the base is 10, and for the FC the base is 2. Fig. 5 shows an image with the particle definition. In order to obtain a valid CNN from a particle, the particle has some restrictions. These restrictions are the first fragment of the particle is always an LR fragment and the last fragment is always an FC layer with value 1 (so the output of the CNN is always

**Algorithm 2** PSO

```
swarm ← initSwarm(swarmSize)
for particle in swarm do
    particle.fitness ← calcPositionFitness(particle.position)
end for
gBest ← findGBest(swarm)
iteration ← 0
for i < iteration do
    for particle in swarm do
        diffPbest ← calcParticleDiff(particle.position, particle.pBest.position)
        diffGbest ← calcParticleDiff(particle.position, gBest.position)
        newVelocity ←
        calcParticleVelocity(diffPbest, diffGbest, particle.pBest.position, gBest.position)
        particle.position ← updateParticlePosition(particle.position, newVelocity)
        particle.fitness ← calcPositionFitness(particle.position)
        if particle.pBest.fitness < particle.fitness then
            particle.pBest.fitness ← particle.fitness
            particle.pBest.position ← particle.position
        end if
        if gBest.fitness < particle.fitness then
            gBest.fitness ← particle.fitness
            gBest.position ← particle.position
        end if
    end for
end for
return gBest
```

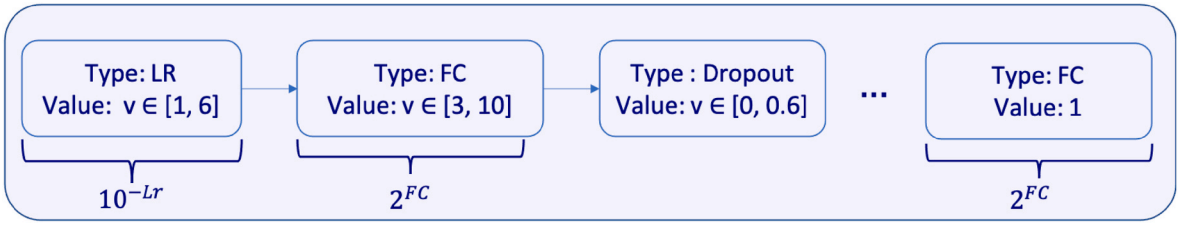


**Fig. 5.** PSO particle definition.

binary, as we have two classes), and the particle cannot have two or more successive dropout fragments. For the sake of simplicity, the last fragment is only added to the fitness evaluation in the conversion to CNN and to return the best particle, as it is not desirable that this fragment be removed from update flow, which the particle undergoes within the algorithm.

Similar to the GA approach, the evaluation of a single particle involves training a CNN in which the architecture of the fully connected layer and the learning rate comes from the particle. Therefore, for each particle, there is the training of the corresponding CNN and the fitness of the particle is the same as described for the GA, i.e., the result of the F1-score of the validation set, along with the penalty, as shown in Equation (3).

Another definition of the PSO is how it calculates the difference between two particles. The particle difference is calculated fragment by fragment. The fragment difference is defined as: 0 when the two fragments have the same type; the fragment of the first particle itself if the two types are different (or if the second particle does not have a corresponding fragment for that position, for the cases where the second particle is smaller than the first); $-1$ when the second particle has a fragment, for which the first does not have a corresponding partner (the case where the number of fragments of the second particle is greater than the first). Fig. 6 shows two different examples of how we calculate particle difference.

It is also necessary to define how to calculate the particle velocity. The velocity of each fragment is defined by either the difference between $pBest - P$ or $gBest - P$. The difference is selected based on a random value $r$ and $Cg$ (one of the PSO arguments). When $r > Cg$, the fragment is selected from $pBest - P$ whereas, when $r <= Cg$, the selected difference is from $gBest - P$. Fig. 7 shows an example of this calculated difference. As noted in Ref. [12], for their PSO, there is one special case for the velocity, which also occurs in our approach. This case happens when both differences ($pBest - P$ and $gBest - P$) have all the fragments equal to 0. We are also using the same procedure as that reported by Ref. [12], in which instead of using the difference, we use $gBest$ and $pBest$ itself, keeping the random selection for each fragment based on $Cg$.

Furthermore, the particle position update should also be defined. We update the particle position in a way that for a fragment that has the velocity equal to 0, the particle fragment is maintained; in the case of a fragment in which the velocity is either the FC layer or the Dropout, the velocity fragment is kept; finally for a fragment with a $-1$, the particle fragment is removed. An example of how the position is updated is shown in Fig. 8, in which the $P1_t$ is the particle for the moment $t$ and the $P1_{t+1}$ is the particle after the update for the moment $t + 1$, updated with the velocity shown in Fig. 8.

## 4. Experiments and results

In this section, we are going to provide details on the experiments performed and present the obtained results. For the experiments, we used Python, PyTorch and Matlab® for some of the preprocessing. In order to take advantage of graphics processing units (GPUs), we opt for using the Google Colab® tool to run our experiments. Furthermore, as the fitness calculation of both the GA and PSO are computationally expensive and time demanding, parallelism is used to reduce run time. We only use the parallelism for the fitness calculation, as each fitness is completely independent from the other individuals or particle.

The two bio-inspired optimization techniques are used to optimize the architecture of the CNN based on the validation subset (which is used for the fitness calculation). When the PSO or the GA finish evolving, we then take the architecture represented in the best individual or the best particle and run four new CNNs with that architecture. The difference between these four new runs is the number of epochs: we run 30 and 50 epochs both with and without early stopping enabled. The early stopping is used to avoid over fitting. Therefore, after 10 epochs without reduction in the loss of the validation set, we stop the training. Finally, all four CNNs are tested with the test data and we then use the F1-score of the test data to decide which are our best networks. Furthermore, all the experiments used the Adam optimizer.

As previously mentioned, we use two subsets of the original static images database, both only with frontal images: one with 294 healthy images and 272 sick images (which we will call group-1) and the other with 38 images from each class (called group-2). Both subsets were divided into 70% of the data for training, 15% for validation and 15% for test.

Both groups (group-1 and group-2) were tested in the optimization using GA and PSO. Therefore, we have four ample setups for the experiments: group-1 with GA, group-1 with PSO, group-2 with GA and group-2 with PSO. Furthermore, each of these setups is executed for all the three CNNs used: ResNet-50, VGG-16 and DenseNet-201. Moreover, as both GA and PSO are meta-heuristic algorithms, we run each experiment 5 times and the best run is presented in this section.

The baseline results that we use to compare with the optimization results is what we called manual experiments. In these experiments, six pre-defined architectures for the fully connected layer were tested with two different learning rates, which were described in our previous study [23]. In summary, the six setups are:

- (1) 1 FC;
- (2) 2 FC with 256 neurons as the output of the first layer and input of the second (256 output neurons of the first layer, 2 output neurons of the second layer);
- (3) 2 FC with 512 neurons (512, 2);
- (4) 2 FC with 1024 neurons (1024, 2);
- (5) 3 FC with 4096 and 1024 neurons (4096, 1024, 2);
- (6) 3 FC with 4096 neurons (4096, 4096, 2).

The best results for each of the CNNs, that use manual setups, are described in Ref. [23], but for the sake of simplicity and comparison, this is also specified in Table 2 and Table 3. We only added the best result for the VGG-16 in the manual setup for group-2, which was not present before.

### 4.1. GA results

GA experiments used an 80% crossover rate. This value is used to verify whether the pair of individuals selected will go through the crossover process. Therefore, a random number between 0 and 1 is generated. If it is greater than 0.8 the selected pair of individuals are maintained as is, otherwise, a single point crossover is applied and two

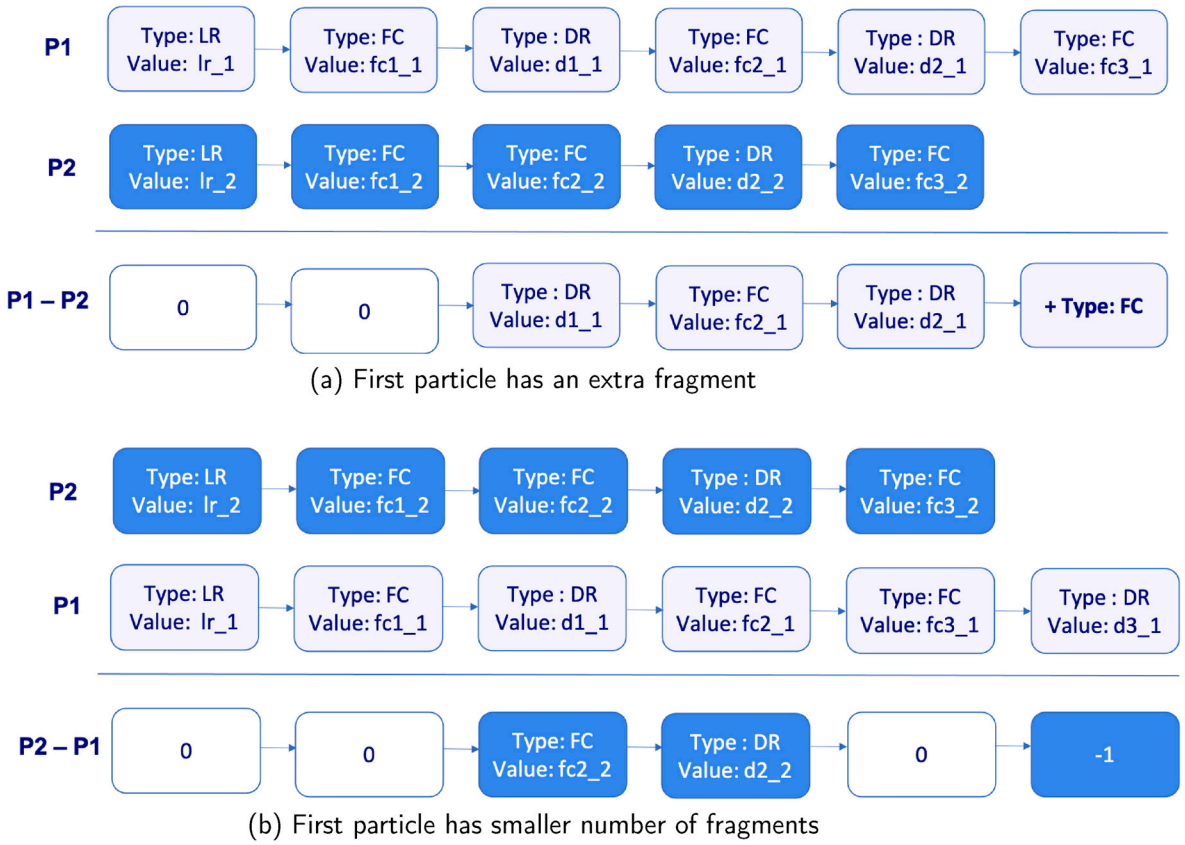


**Fig. 6.** Calculating the difference between two particles.

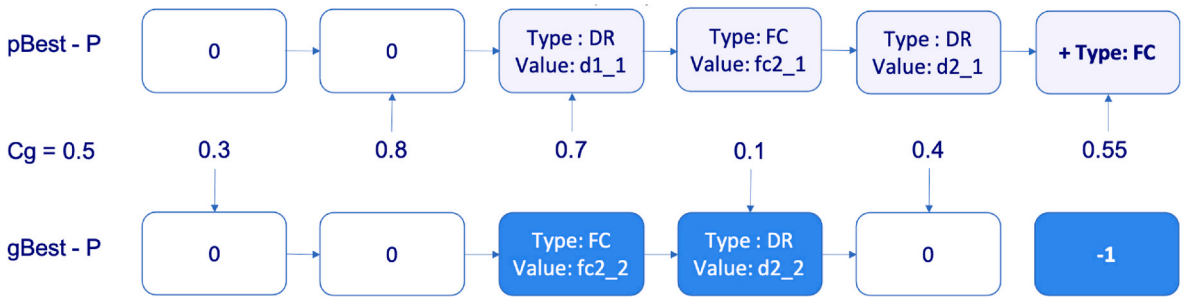


**Fig. 7.** PSO velocity calculation.

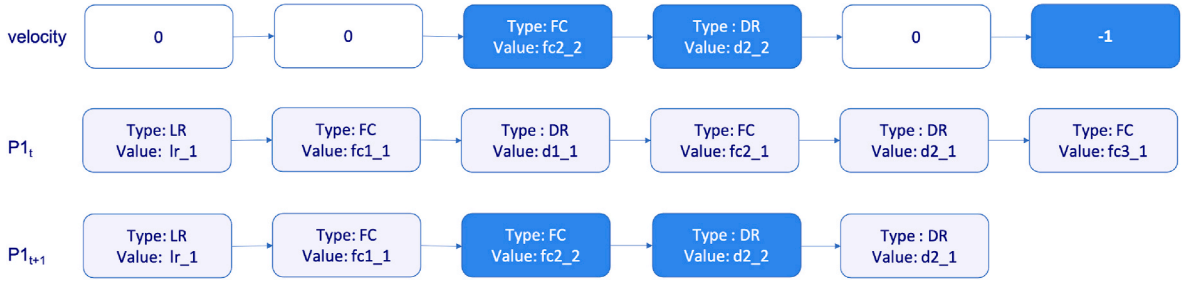


**Fig. 8.** PSO update position.

children are generated. Other parameters that were not changed during the experiments reported in this section, were the tour size, all the experiments used a tour of 2, and the number of iterations, where we used 10 for all experiments. Other parameters such as population size and mutation rate were varied empirically, testing the best ones. For the population size (PS), tested values were 10 and 20; for the mutation rate (MR), we tested 20%, 30%, 40% and 60%. Therefore, we have 8 possible scenarios here to be tested (each PS for each MR).

Table 4 and Table 5 show the best results obtained with our GA for both groups of data previously described. In the tables, 'Best GA Fitness'

**Table 2**
Baseline results manual experiments - group-1.

| CNN | LR | Architecture | Acc(%) | SE(%) | SP(%) | F1-score |
|---|---|---|---|---|---|---|
| DenseNet-201 | 0.000 1 | 2 FC layers with 1024 neurons (1024, 2) | 88.23 | 93.75 | 80.43 | 0.8824 |
| ResNet-50 | 0.000 1 | 2 FC layers with 1024 neurons (1024, 2) | 81.91 | 91.67 | 71.74 | 0.8381 |
| VGG-16 | 0.001 | 2 FC layers with 512 neurons (512, 2) | 77.66 | 72.92 | 82.61 | 0.7692 |

**Table 3**
Baseline results manual experiments - group-2.

| CNN | LR | Architecture | Acc(%) | SE(%) | SP(%) | F1-score |
|---|---|---|---|---|---|---|
| DenseNet-201 | 0.001 | 3 FC layers with 4096 neurons (4096, 4096, 2) | 91.67 | 100 | 83.33 | 0.9230 |
| ResNet-50 | 0.001 | 2 FC layers with 512 neurons (512, 2) | 83.33 | 83.33 | 83.33 | 0.8333 |
| VGG-16 | 0.001 | 2 FC layers with 1024 neurons (1024, 2) | 75.0 | 50.0 | 1 | 0.6666 |

refers to the F1-score of the validation subset of best individual of the final population, whereas the ACC, SE, SP and F1-score refers to accuracy, sensitivity, specificity and F1-score, respectively, of the test subset, which were obtained after the entire GA flow had finished. The NE is the number of epochs from the four possibilities tested that gave that result. The obtained learning rate and fully connected layer architecture obtained for group-1 are:

- DenseNet-201: LR $10^{-4}$; FC layer (4096, 256, dropout 0.340 1, 2);
- ResNet-50: LR $10^{-5}$; FC layer (1024, 2);
- VGG-16: LR $10^{-3}$; FC layer (1024, 8, 2);

Moreover, the obtained learning rate and fully connected layer architecture obtained for group-2 are:

- DenseNet-201: LR $10^{-4}$; FC layer (2);
- ResNet-50: LR $10^{-5}$; FC layer (128, dropout 0.426 7, 32, 2);
- VGG-16: LR $10^{-2}$; FC layer (32, dropout 0.306 5, 8, dropout 0.315 3, 2);

GA evolution was evaluated with 30 epochs without early stopping enabled and 30 epochs with early stopping. For group-1, the GA performed better when the evolution occurred without the early stopping, which are the results shown in Table 4. On the other hand, for group-2 when the GA evolution process had the early stopping enabled, the CNNs performed better in the test data as shown in Table 5. Another change is the fitness penalization factor in which for group-1 we use 0.6 and for group-2 we use 0.7 (Equation (3)), these were empirically chosen.

By analyzing the results in Table 4, we note that the VGG-16 is the CNN that benefited the most from the GA optimization for group-1. DenseNet-201 obtained equivalent results with the manual setup [23] and the setup with the GA, both with 0.88 of the F1-score. ResNet-50 presented slightly better results with the GA obtaining 0.846 1 of the F1-score compared to the manual setup 0.838 1 of the F1-score. The GA optimization for the VGG-16 improved the result from 0.769 2 of the F1-score (with the manual setup) to 0.857 1 of the F1-score. We note also that all the results were obtained with 40% of mutation rate. Moreover, both DenseNet-201 and VGG-16 required more individuals in order to obtain their best results.

In Table 5, which refers to data group-2, we note that the DenseNet-201 showed the same value for the F1-score using the GA as the manual experiments. Nevertheless, for VGG-16 and ResNet-50, results improved significantly using the GA optimization. The VGG-16 achieved without the GA optimization 0.66 of the F1-score and with the optimization, it obtained 0.92 of the F1-score, while, ResNet-50 improved from 0.83 of the F1-score to 0.90.

Moreover, in group-2, the best result obtained with the DenseNet-201 with only 10 individuals was with 60% of mutation rate. Whereas, the best result over all setups was with 20 individuals and 30% of MR. We note that the smaller population required more mutation to generate greater diversity, which lead to better results. Nevertheless, with more individuals, greater MR did not perform as well. A reason for that could be the need of more iterations for convergence. For ResNet-50, the best result was obtained with 20 individuals and 60% of MR. Therefore, the GA with more variability for the ResNet-50 provided better results. Different from that which happened with DenseNet-201 and ResNet-50, VGG-16, with only 10 individuals, performed better with a smaller MR. The VGG-16 might need more iterations in situations where there is more diversity to converge and provide good results. Even though, the result obtained with only 10 individuals and an MR of 30% is still significant.

As each evaluation of an individual involves training a CNN, in order to obtain the fitness, it is computational expensive and time demanding to do so, which made it impracticable to run these evaluations with a larger population size and more than 20 iterations.

**Table 4**
Best results GA - group-1.

| CNN | PS | MR | Best GA Fitness | NE | ACC(%) | SE(%) | SP(%) | F1-score |
|---|---|---|---|---|---|---|---|---|
| DenseNet-201 | 20 | 40 | 0.7407 | 30 with ES | 87.23 | 93.75 | 80.43 | 0.8823 |
| ResNet-50 | 10 | 40 | 0.6341 | 30 | 82.97 | 91.66 | 73.91 | 0.8461 |
| VGG-16 | 20 | 40 | 0.7407 | 30 | 84.04 | 93.75 | 73.91 | 0.8571 |

**Table 5**
Best results GA - group-2.

| CNN | PS | MR | Best GA Fitness | NE | ACC(%) | SE(%) | SP(%) | F1-score |
|---|---|---|---|---|---|---|---|---|
| DenseNet-201 | 20 | 30 | 0.8571 | 30 | 91.66 | 100 | 83.33 | 0.9230 |
| ResNet-50 | 20 | 60 | 0.9230 | 50 | 91.66 | 83.33 | 100 | 0.9090 |
| VGG-16 | 10 | 30 | 0.9090 | 30 | 91.66 | 100 | 83.33 | 0.9230 |

### 4.2. PSO results

The PSO arguments analyzed during the experiments were the number of particles, for which we used 10 and 20, the number of iterations, where we also used 10 and 20; and the *Cg*, in which we tested 0.5, 0.6 and 0.4. Therefore, we tested giving the best particle more or less relevance. As mentioned in the GA section, the particle evaluation also involves training a CNN. Hence, it was impracticable running the PSO with larger swarms and with more iterations. Therefore, the experiment with 20 iterations was only tested with 10 particles and *Cg* of 0.5, but all possible combinations of number of particle and *Cg* were tested for 10 iterations.

The best PSO results obtained for each group are shown in Table 6 and Table 7. The generated networks for group-1 are:

- DenseNet-201: LR $10^{-4}$; FC layer (64, dropout 0.295 0, 32, 2);
- ResNet-50: LR $10^{-4}$; FC layer (32, dropout 0.337 8, 16, dropout 0.255 0, 2);
- VGG-16: LR $10^{-4}$; FC layer (128, 128, 16, dropout 0.552 1, 2);

For group-2 we obtained:

- DenseNet-201: LR $10^{-4}$; FC layer (8, dropout 0.444 3, 2);
- ResNet-50: LR $10^{-5}$; FC layer (1024, dropout 0.133 8, 32, 2);
- VGG-16: LR $10^{-3}$; FC layer (16, dropout 0.207 5, 8, dropout 0.149 9, 8, 2);

Similar to the GA, we tested the PSO flow with 30 epochs with and without early stopping. Analogous to the GA results presented for the two tested groups, group-1 demonstrates the results without early stopping, whereas group-2 demonstrates the results from the PSO flow with 30 epochs and the early stopping enabled. Furthermore, these also have the fitness penalization as described for the GA, with the 0.6 for group-1 and 0.7 for group-2.

By analyzing the results in Table 6 for group-1, we note that the DenseNet-201 result is inferior to the manual results although slightly, obtaining 0.87 of the F1-score, whereas the manual setup obtained 0.88 of the F1-score. ResNet-50 obtained a result a little bit better with the PSO obtaining 0.84 of the F1-score compared to the manual results, which obtained 0.83. The results with the VGG-16 were improved using the PSO flow, from 0.666 6 to 0.837 9 of the F1-score for the test data. The behavior of the results is similar to that obtained with the GA, in which the VGG-16 is the CNN that benefited the most from the optimization and the DenseNet-201 in which the optimization was not significant.

In Table 7, with the results from group-2, we note that the PSO performed better, either with greater number of iterations or the number of particles, for two of the CNNs (VGG-16 and ResNet-50) performing better, with 20 particles, and DenseNet-201 performed better with only 10 particles, but with more iterations. The best result for the DenseNet-201 in group-2 performed equivalent to both the manual and GA results with 0.92 of the F1-score. The experiments with the DenseNet-201 and *Cg* 0.6 presented results below the other two *Cg* tested for both of the swarm size (SS). The best ResNet-50 result using the PSO was superior to the manual results. Using the optimization, the network achieved 0.857 1 of the F1-score in the test set against 0.833 3 of the F1-score using the manual setup. This PSO result is below that of the GA, which achieved 0.90 of the F1-score in the test data. Moreover, in opposition to the DenseNet-201 behaviour with the *Cg* of 0.6, for the ResNet-50, using the same value, it achieved the best results with both of the SS tested. Similar to the results of the ResNet-50, the PSO optimization with the VGG-16 outperformed the manual setup (0.833 3 against 0.666 6 of the F1-score), but, still below the GA result, which obtained 0.92.

### 4.3. Discussion

The results with GA and PSO optimization improved the majority of the results obtained previously with the manual setup [23]. For the CNN that presented the worst result in the manual setup, i.e., VGG-16, both of the optimization techniques were able to find architectures and learning rates that significantly improved the results. DenseNet-201 is the only CNN for which the optimization did not find better results. Here, the results obtained were equivalent to the manual, except for the PSO in group-1, which is slightly inferior. This occurred mainly due to there being more room for improvement in the CNNs from which we obtained poor results in the manual setup, contrary to the DenseNet-201, which already had good results. Furthermore, the results obtained with data from group-2 were improved more significantly when using the optimization techniques over the results that used data from group-1. One notes that, using the optimization techniques, specifically the GA, all the networks showed results above 0.90 of the F1-score using group-2, whereas for group-1 the two optimization techniques only produced a slight improvement to the results.

As previously mentioned, the study developed by Junior et al. in Ref. [12] showed that the use of their proposed PSO was successful in finding relevant CNN architecture, which had less parameters and consequently faster conversion in many of the tested databases. These authors used databases that are commonly used to evaluate CNNs (MNIST, MNIST-RD, MNIST-RB, MNIST-BI, MNIST-RD + BI, Rectangles, Rectangles-I, Convex, and MNIST-Fashion). In these databases, even with smaller CNNs, their algorithm found CNNs which provided better results than the state of the art models tested in those databases, in six out of the nine databases tested. Even though Junior et al. used the PSO to find good CNN architecture over the entire network (convolutional layers, dropout layers and fully connected layers), and we only used PSO

**Table 6**
Best results PSO - group-1.

| CNN | Iterations | SS | Cg | Best PSO Fitness | NE | ACC(%) | SE(%) | SP(%) | F1-score |
|---|---|---|---|---|---|---|---|---|---|
| DenseNet-201 | 10 | 10 | 0.4 | 0.7209 | 30 | 86.17 | 93.75 | 78.26 | 0.8737 |
| ResNet-50 | 10 | 10 | 0.5 | 0.7021 | 50 | 81.91 | 95.83 | 67.39 | 0.8440 |
| VGG-16 | 10 | 10 | 0.5 | 0.7323 | 30 with ES | 81.91 | 89.58 | 73.91 | 0.8349 |

**Table 7**
Best results PSO - group-2.

| CNN | Iterations | SS | Cg | Best PSO Fitness | NE | ACC(%) | SE(%) | SP(%) | F1-score |
|---|---|---|---|---|---|---|---|---|---|
| DenseNet-201 | 20 | 10 | 0.5 | 0.7272 | 30 with ES | 91.66 | 100 | 83.33 | 0.9230 |
| ResNet-50 | 10 | 20 | 0.6 | 0.7058 | 30 with ES | 83.33 | 100 | 66.66 | 0.8571 |
| VGG-16 | 10 | 20 | 0.5 | 0.8571 | 50 | 83.33 | 83.33 | 83.33 | 0.8333 |

for the fully connected layer part of the network, our results also support their results in a completely different dataset, one with infrared images. Using the PSO, we were able to improve the majority of the results, which were originally obtained by a manual approach.

A GA algorithm used to find good CNN architectures was presented in Ref. [13]. In similar fashion to the PSO proposed in Ref. [12], the proposed GA algorithm is used to find the entire CNN architecture, whereas we are using it to optimize the fully connected layers. Furthermore, different to our study, the authors proposed a variable length GA individual, whereas we opt for using a fixed length individual, which allows for flexibility, as suggested by Ref. [22]. Nevertheless [13], and our study show that the GA is a suitable technique to help in the specific problem of finding good CNN architectures and hyperparameters associated with it. The study in Ref. [13] showed that the proposed GA outperformed all the 22 compared algorithms from the state of the art in nine different tested databases. In this study, using the proposed GA, we were able to find results that outperformed the majority of the results, when using only the manual tests.

An interesting observation drawn from our results is that the GA performed better for the majority of the experiments compared to the PSO. Although our particle definition is flexible, as it allows for particle from different sizes, the particle operations do not change the values of the fragment of the particle (either a learning rate fragment, or a dropout fragment, or the fully connected layer). It only changes the number of layers and the order of those layers. The crossover of the GA also does not change the values, but the mutation does. For the PSO optimization, our problem might require a more smoother velocity update. Therefore, a future study would be to test either a different velocity calculation or another particle definition.

In [12,13], the finding of smaller CNNs is a very important goal due to the fact that smaller networks converge faster, and consequently demand less time. As we are using the convolutional layers of pre-trained CNNs from the state of the art, the depth of the network is not our main focus in the optimization. Although, as we also understand that smaller fully connected layers are more suitable, also because it demands less time, we use this depth as the tiebreaker in the fitness evaluation. Therefore, when two particles or individuals have the same fitness value, we opt for the one with the smaller number of layers.

Although Zuluaga et al. in Ref. [7] used dynamic infrared images and we are using the static infrared ones (from the same database), both studies showed that, using optimization techniques to find CNN architectures for the specific field of breast cancer detection using infrared images, showed improvements when compared to finding architectures from scratch or even with only transfer learning approaches. In their work, an algorithm based on a tree parzen estimator (Bayesian optimization) is proposed to help find a suitable CNN architecture and the entire CNN is chosen with this technique, and as presented before, we use GA and PSO to find the architecture and hyperparameters of the fully connected layers. Even with all the differences, i.e., the protocol of infrared images used, the number of images and the optimization technique, both studies present an F1-score of 0.92 in their best results.

As shown in Table 1, the work of Cabiglu et al. [11] presented an accuracy of 94.3%, while even with improvements to optimization, the accuracy of our models is around 91.6%. Noteworthy here is that both our study and [11] used the static images. Although in group-1 there are the same number of patients as in Ref. [11], after the data augmentation we ended up with a different number of patients (we obtained 294 healthy images and 272 sick ones whereas in Ref. [11] the authors used a total 147 healthy patients and 135 sick ones). This is due to our opting for the use of data augmentation in both classes, healthy and sick, and Cabiglu et al. augmented only the sick class. We believe that to augment only one class makes the classification easier, as all the rotated images, for example, will be the sick ones, without the need for any other information concerning the cancer in the breast (hotter areas for instance). Even though, we are working on more experiments with the exact same number of patients for the sake of comparison, in order to understand if our results are going to improve, as we expect, using the same approach as [11] (although, not using the exact same patients). Another difference of note is the CNN used, the authors used AlexNet, and we chose VGG-16, ResNet-50 and DenseNet-201.

Due to the high time demand and expensive computational cost for performing of our experiments, a future could be based on the use of a surrogate model in the fitness evaluation of the GA and PSO. This would allow us to evaluate the individuals/particle without the need of training CNNs for each one of them in all the iterations. By using a surrogate model, we might be able to test the optimization techniques with more individuals or particle and with a longer number of iterations, hence, the possibility of discovering even better results.

Even though the database has many other static images available, as each patient has roughly five images taken from different angles, as discussed in Ref. [23], for the majority of the images we do not have the information of which breast the cancer is located. This becomes a problem in the lateral images, as the network might see an image from a healthy breast from a patient which belongs to the sick class or even a breast that does in fact have cancer, but in the lateral image, the cancer is not visible. In order to use the lateral images without negatively affecting the classification, we are planning on an alternative that uses all the images without losing the information concerning to which patient they belong. An example is an image preprocessing approach that produces a 5D image, where each dimension is an image from one angle, therefore, we will be able to tie all the images together and show all of them to the network at once. Furthermore, having more images the computational complexity becomes even more important as time will increase, so this might be viable after implementing the surrogate and reducing the fitness calculation time. In this approach, we will also need to make slight adjustments the CNNs, in order to accommodate an input image not only with 3 but with 5 channels.

We are also planning to test the proposed methods on two other subsets of the database DMR-IR: a subset with the dynamic images in order to compare the two types of thermography protocols and a subset of an ROI extraction in order to verify whether using only a ROI containing only the area of interest, i.e., the breast of the patient, outperforms the use of the entire images.

## 5. Conclusions

Thermography, or just infrared imaging, is a technique that is showing good potential in supporting the early detection of breast cancer. In addition, a CAD system based on CNNs is also promising due to the networks ability to perform image classification. In order to find good results using CNNs, it is essential to find suitable hyperparameters and architecture, which is not an easy task, as it is expensive, demands time and may require a human expert.

In this study, we present two bio-inspired algorithms, GA and PSO that were used to optimize some hyperparameters and the architecture of fully connected layers from three state of the art CNNs (VGG-16, ResNet-50 and DenseNet-201), in the context of breast cancer detection with static infrared images. We described the individual representation, fitness calculation, selection, crossover and mutation operator for the proposed GA. Moreover, we also detailed the particle definition, velocity calculation, particle update and fitness of the proposed PSO.

We found that the GA outperformed the PSO in our experiments for the database used, but that both proposed optimizations outperformed the majority of the manual experiments [23]. Through the GA optimization, we were able to improve the VGG-16 result from 0.66 to 0.92 of the F1-score in the test subset with 38 static images from each class (called group-2). We were also able to improve the ResNet-50 results from 0.83 to 0.90 of the F1-score, also for the test set in group-2. Furthermore, the proposed optimization method can also be applied to other networks and in other problem domains.

Results presented in this study are encouraging and confirm the potential of thermography to assist in the task of detecting breast cancer

in its early stages. The next steps of our study include the use of different images from DMR-IR, i.e., consider only the ROI that includes only the affected regions (breasts) and to use images from the dynamic protocol to also consider the transient aspect of the physical phenomenon involved. Finally, we will test a surrogate model for our fitness calculation, for both GA and PSO, to allow the evaluation of individuals/particle without the need of training CNNs for each of the iterations and eventually finding better networks.

## CRediT authorship contribution statement

**Caroline Barcelos Gonçalves:** Conceptualization, Methodology, Software, Write of first draft. **Jefferson R. Souza:** Revision, Supervision. **Henrique Fernandes:** Revision, Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* - Brazil (CAPES) - Finance Code 001, by the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* - Brazil (CNPq) - Finance Code 407 140/2 021-2 and by the *Fundação de Amparo à Pesquisa de Minas Gerais* - Brazil (FAPEMIG).

## Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.compbiomed.2021.105205.

## References

[1] I. American Cancer Society, What is breast cancer?. https://www.cancer.org/cancer/breast-cancer/about/what-is-breast-cancer.html, 2021. (Accessed 12 September 2021).
[2] I. American Cancer Society, How common is breast cancer?. https://www.cancer.org/cancer/breast-cancer/about/how-common-is-breast-cancer.html, 2021. (Accessed 12 September 2021).
[3] I. Instituto Nacional de Câncer, Câncer de mama, 2021. https://www.inca.gov.br/tipos-de-cancer/cancer-de-mama. (Accessed 12 September 2021).
[4] I. Instituto Nacional de Câncer, Controle do Câncer de Mama, 2021. https://www.inca.gov.br/mama. (Accessed 12 September 2021).
[5] I. American Cancer Society, American Cancer Society Recommendations for the Early Detection of Breast Cancer, 2021. https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/american-cancer-society-recommendations-for-the-early-detection-of-breast-cancer.html. (Accessed 12 September 2021).
[6] I. American Cancer Society, Breast Density and Your Mammogram Report, 2021. https://www.cancer.org/cancer/breast-cancer/screening-tests-and-early-detection/mammograms/breast-density-and-your-mammogram-report.html. (Accessed 12 September 2021).
[7] J. Zuluaga-Gomez, Z. A. Masry, K. Benaggoune, S. Meraghni, N. Zerhouni, A CNN-Based Methodology for Breast Cancer Diagnosis Using Thermal Images, arXiv preprint arXiv:1910.13757 .
[8] M.d.F.O. Baffa, L.G. Lattari, Convolutional neural networks for static and dynamic breast infrared imaging classification, in: 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE, 2018, pp. 174–181.
[9] E. Chaves, C.B. Gonçalves, M.K. Albertini, S. Lee, G. Jeon, H.C. Fernandes, Evaluation of transfer learning of pre-trained CNNs applied to breast cancer detection on infrared images, Appl. Opt. 59 (17) (2020) E23–E28.
[10] A.A.A. Figueiredo, J.G. do Nascimento, F.C. Malheiros, L.H. da Silva Ignacio, H. C. Fernandes, G. Guimaraes, Breast tumor localization using skin surface temperatures from a 2D anatomic model without knowledge of the thermophysical properties, URL, Comput. Methods Progr. Biomed. 172 (2019) 65–77, https://doi.org/10.1016/j.cmpb.2019.02.004. ISSN 0169-2607, https://www.sciencedirect.com/science/article/pii/S0169260719300094.
[11] Ç. Cabıoğlu, H. Oğul, Computer-aided breast cancer diagnosis from thermal images using transfer learning, in: International Work-Conference on Bioinformatics and Biomedical Engineering, Springer, 2020, pp. 716–726.
[12] F.E.F. Junior, G.G. Yen, Particle swarm optimization of deep neural networks architectures for image classification, Swarm. Evol. Comput. 49 (2019) 62–74.
[13] Y. Sun, B. Xue, M. Zhang, G. G. Yen, Evolving Deep Convolutional Neural Networks for Image Classification, IEEE Transactions on Evolutionary Computation .
[14] A. Reiling, W. Mitchell, S. Westberg, E. Balster, T. Taha, CNN optimization with a genetic algorithm, in: 2019 IEEE National Aerospace and Electronics Conference (NAECON), IEEE, 2019, pp. 340–344.
[15] J.C. Torres-Galván, E. Guevara, E.S. Kolosovas-Machuca, A. Oceguera-Villanueva, J.L. Flores, F.J. González, Deep convolutional neural networks for classifying breast cancer using infrared thermography, Quant. InfraRed.Thermogr.J (2021) 1–12.
[16] F. Johnson, A. Valderrama, C. Valle, B. Crawford, R. Soto, R. Ñanculef, Automating configuration of convolutional neural network hyperparameters using genetic algorithm, IEEE Access 8 (2020) 156139–156152.
[17] L. Silva, D. Saade, G. Sequeiros, A. Silva, A. Paiva, R. Bravo, A. Conci, A new database for breast research with infrared image, J. Med. Imaging.Health Inf. 4 (1) (2014) 92–100.
[18] M.A. Farooq, P. Corcoran, Infrared imaging for human thermography and breast tumor classification using thermal images, in: 2020 31st Irish Signals and Systems Conference (ISSC), IEEE, 2020, pp. 1–6.
[19] S.J. Mambou, P. Maresova, O. Krejcar, A. Selamat, K. Kuca, Breast cancer detection using infrared thermal imaging and a deep learning model, Sensors 18 (9) (2018) 2799.
[20] R. Roslidar, K. Saddami, F. Arnia, M. Syukri, K. Munadi, A study of fine-tuning CNN models based on thermal imaging for breast cancer classification, in: 2019 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), IEEE, 2019, pp. 77–81.
[21] S. Kiymet, M.Y. Aslankaya, M. Taskiran, B. Bolat, Breast cancer detection from thermography based on deep neural networks, in: 2019 Innovations in Intelligent Systems and Applications Conference (ASYU), IEEE, 2019, pp. 1–5.
[22] M.V. Fidelis, H.S. Lopes, A.A. Freitas, Discovering comprehensible classification rules with a genetic algorithm, in: Proceedings of the 2000 Congress on Evolutionary Computation. Cec00 (Cat. No. 00th8512), vol. 1, IEEE, 2000, pp. 805–810.
[23] C.B. Gonçalves, J.R. Souza, H. Fernandes, Classification of static infrared images using pre-trained CNN for breast cancer detection, in: 2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS), IEEE, 2021, pp. 101–106.
[24] A. Baldominos, Y. Saez, P. Isasi, Evolutionary convolutional neural networks: an application to handwriting recognition, Neurocomputing 283 (2018) 38–52.
[25] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.