Contents lists available at ScienceDirect

Applied Soft Computing

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

Evolutionary optimization of convolutional neural networks for cancer miRNA biomarkers classification

Alejandro Lopez-Rincon[a,b,c,*], Alberto Tonda[a,b,c], Mohamed Elati[a,b,c], Olivier Schwander[a,b,c], Benjamin Piwowarski[a,b,c], Patrick Gallinari[a,b,c]

[a] UPMC LIP6; UMR7606 – Laboratoire d'informatique de Paris 6, France
[b] Univ. Lille, UMR 9189 CRIStAL, Centre de Recherche en Informatique Signal et Automatique de Lille, France
[c] UMR 782 GMPA, Université Paris-Saclay, INRA, AgroParisTech, France

ARTICLE INFO

Article history:
Received 22 May 2017
Received in revised form
18 November 2017
Accepted 25 December 2017
Available online 3 January 2018

Keywords:
Cancer classification
miRNA biomarker
Convolutional neural networks
Tensorflow
Evolutionary algorithms

ABSTRACT

Cancer diagnosis is currently undergoing a paradigm shift with the incorporation of molecular biomarkers as part of routine diagnostic panel. This breakthrough discovery directs researches to examine the role of microRNA in cancer, since its deregulation is often associated with almost all human tumors. Such differences frequently recur in tumor-specific microRNA signatures, which are helpful to diagnose tissue of origin and tumor subtypes. Nonetheless, the resulting classification problem is far from trivial, as there are hundreds of microRNA types, and tumors are non-linearly correlated to the presence of several overexpressions. In this paper, we propose to apply an evolutionary optimized convolutional neural network classifier to this complex task. The presented approach is compared against 21 state-of-the-art classifiers, on a real-world dataset featuring 8129 patients, for 29 different classes of tumors, using 1046 different biomarkers. As a result of the comparison, we also present a meta-analysis on the dataset, identifying the classes on which the collective performance of the considered classifiers is less effective, and thus possibly singling out types of tumors for which biomarker tests might be less reliable.

1. Introduction

Cancer diagnosis is currently undergoing a paradigm shift with the incorporation of molecular biomarkers as part of routine diagnostic panel. The molecular alteration ranges from those involving the DNA, RNA, microRNAs (miRNAs) and proteins. This breakthrough discovery directed researches to examine role of miRNA in cancer since miRNAs are involved in the development, cell differentiation, and regulation of cell cycle. Moreover, their deregulation is often associated with almost all human tumors with respect to the normal tissue counterpart. These differences frequently recur in tumor-specific miRNA signatures, which are very helpful to diagnose the tissue of origin and tumor subtypes. Increasing evidence also supports a role for miRNAs as prognostic biomarkers of human cancers. If this new information makes it possible to refine the general characteristics of a given disease, they mainly highlight subtle molecular variations capable of explaining the differences observed between each patient. The integration of these individual variations into clinical practice and therapeutic management is a new strategy called personalized medicine or precision medicine.

Given the importance of this topic, it is not surprising that there are currently several attempts to build international databases of miRNA expressions [1–5]. From an analysis of this vast corpus, it is possible to extract the overexpression of certain biomarkers: such overexpression can be used as a non-invasive diagnostic. For example, in [6] miR-21 is mentioned as a common over-expressed microRNA in cancer, and a proven oncogene; in [7], miR-143, miR-222, and miR-452 are shown to be useful as tumor stratification and noninvasive diagnostic biomarkers for bladder cancer; and in [8], miR-22 is proved to be involved in various cellular processes related to carcinogenesis. There are other advantages coming from the study of miRNA, for example to modulate certain miRNA can be used to suppress tumors: miR-23b and miR-124 have therapeutic potential against cervical cancer and gastric cancer, respectively. Considering advances in miRNA detection through blood [9], it could be possible in the foreseeable future to create simple cancer tests to detect such early biomarkers [10,11].

* Corresponding author at: UPMC LIP6; UMR7606 – Laboratoire d'informatique de Paris 6, France.
E-mail addresses: alejandro.lopezrn@hotmail.com (A. Lopez-Rincon), alberto.tonda@inra.fr (A. Tonda), mohamed.elati@univ-lille.fr (M. Elati), olivier.schwander@lip6.fr (O. Schwander), benjamin.piwowarski@lip6.fr (B. Piwowarski), patrick.gallinari@lip6.fr (P. Gallinari).

https://doi.org/10.1016/j.asoc.2017.12.036
1568-4946/© 2017 Elsevier B.V. All rights reserved.

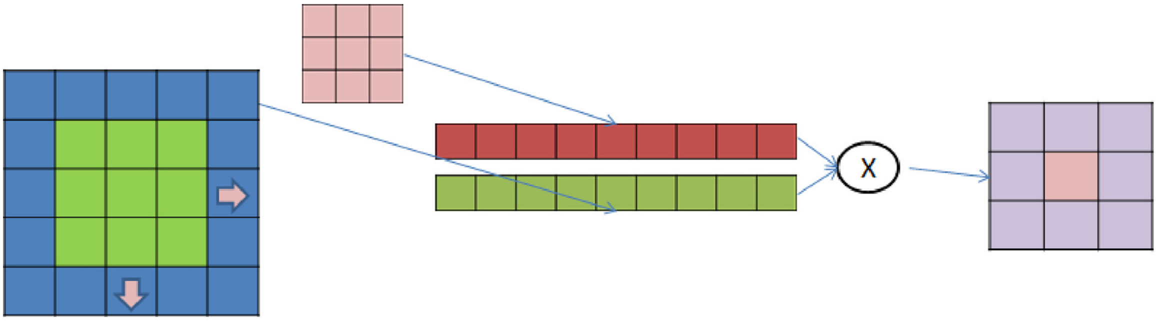


Fig. 1. CNN example. A 5 × 5 input tensor (for example, a 5 × 5 pixel picture in black and white), is passed through a weight tensor of size 3 × 3.

Still, analyzing thousands of case studies featuring hundreds of miRNA types is far from trivial, and even traditional statistical analyses may fall short, as the emergence of certain types of tumors might be non-linearly correlated to the presence of several miRNA overexpressions. When facing such a difficult problem, with a significant amount of data available, machine learning techniques proved again and again to be particularly effective, in applications ranging from image recognition [12,13] to finding the sources of Ischemia and re-entry of the electrical activity of the heart [14–16], to many other real-world classification problems [17–20].

In this paper, a novel machine-learning approach for the classification of cancer types is proposed. Starting from thousands of case studies featuring miRNA biomarkers, extracted from established online databases, a convolutional neural network is trained resorting to an evolutionary optimization algorithm. The presented approach is then tested against 21 state-of-the-art classifiers, comparing classification accuracy on unseen data. As a result of the comparison, we are also able to identify the cancer classes on which a biomarker-based classification seems the most and the least effective, providing insight on the types of cancers that might be harder to identify using the proposed procedure.

The rest of the paper is organized as follows. Section 2 provides the necessary background elements to introduce the scope of the work. The target dataset and the proposed approach are detailed in Section 3. Experimental results, along with a comparison with several state-of-the-art classifiers, are reported in Section 4, while Section 5 concludes the paper.

2. Background

In this section, basic concepts related to convolutional neural networks and evolutionary algorithms are provided, in order to better introduce the scope of the present work.

2.1. Convolutional neural networks

While Artificial Neural Networks (ANNs) have been a popular machine learning technique since the late 80s [21], it's only recently that the community proposed new layouts to make them even more effective [22]. This breakthrough is commonly known as *deep learning* (DL) [23]. Among several DL variations, *Convolutional Neural Networks* (CNNs) currently represent the state-of-the-art for complex classification problems, especially related to image analysis [24]. CNNs' connectivity pattern between neurons is inspired by the organization of the visual cortex in animals. Individual cortical neurons respond to stimuli in a restricted region of space known as the *receptive field*. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation.

Convolution is better described by an example: let's consider an input tensor (a generic *n*-dimensional matrix) of size 5 × 5, with a tensor of weights 3 × 3. In CNNs, a sliding window is used to represent the receptive field: using a sliding window of size 3 × 3 in the example would result in the convolution operation presented in Fig. 1. The sliding window and the weights are turned into a vector, and the dot product of each is summed up. After the dot product, we have an activation function (for each neuron), typically a rectified linear unit, commonly referred to as *ReLU* [25], with a function $f(x) = max(x, 0)$. This will result in a single value. We repeat the same procedure to the whole input tensor, resulting in a 3 × 3 output tensor. In practical applications, input tensors can have even higher dimensions. To further reduce the output size, a *max pooling* operation can be performed, where only the highest value in a sliding grid is reported to the subsequent layer. Following the example of the convolution operation, the max pooling of the 3 × 3 output tensor with a 2 × 2 sliding window, will result in a 2 × 2 output tensor. In a complete CNN, the resulting 4 values will be transmitted to a fully connected rectifier layer with a *softmax* function, for classification applications. The *softmax* function reduces a *k*-dimensional vector of arbitrary real values to a *k*-dimensional vector of real values in range (0, 1), that add up to 1: this can be used to associate each class with a probability.

Once the structure of a CNN is defined, its internal weights need to be optimized to fit the target problem. This operation is performed through backpropagation. In backpropagation, the initial system output is compared to the desired output, and the system is adjusted until the difference between the two is minimized, typically resorting to gradient descent optimization, using cross-entropy loss as the function to minimize [26]. Cross-entropy loss for one-hot labels is defined as:

$$L = \sum_{j=1}^{N} \sum_{i=1}^{M} -t_j^{(i)} \log z_j^{(i)}$$

where $t_j = (0, \ldots, 0, \underbrace{1}_{k}, 0, \ldots, 0)$ is the desired output vector and z_j the calculated output vector, if the example *j* belongs to the *m*-th class, and

$$z_j^{(i)} = \frac{e^{f_j}}{\sum_{i=1}^{M} e^{f_i}}, \tag{1}$$

is the softmax function, and *N* the number of samples.

In machine learning, overfitting a classifier to the training data is a common risk. For CNNs, common practices to reduce this issue are termed *regularization*, defined as *any modification we make to the learning algorithm that is intended to reduce the generalization error, but not its training error* [26]: dropout and weight penalty are among the most common techniques in this group. Dropout randomly *drops* units during the training phase, in order to avoid hyper-specialization of a few units [27]. Weight penalty modifies

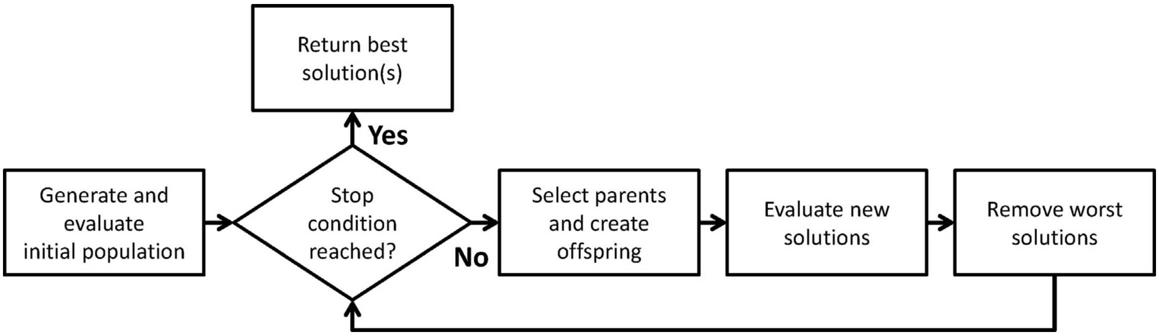


Fig. 2. Flowchart of a generic Evolutionary Algorithm (EA). Parent selection is usually stochastic, with the best candidate solutions having a higher probability to generate offspring. Removal of individuals is usually deterministic: the solutions are sorted by fitness values, and the worst ones are deleted.

the L function to include a β value, preventing weights' values from becoming too large:

$$L = \sum_{j=1}^{N}\sum_{i=1}^{M} - t_j^{(i)} \log z_j^{(i)} + \frac{1}{2}\beta \sum_{K}\sum_{L} W_{k,l}^2, \qquad (2)$$

where $W_{k,l}$ is the weight for connection k in layer l, K is the total number of connections in layer l, and L is the total number of layers.

Even though CNNs have proven to be powerful classification tools, finding the best CNN layout for a specific application is far from trivial: layouts proposed in literature are often created through trial-and-error attempts, or inspired by previous publications. This is understandable, as the search space of all possible layouts is so vast to be practically impossible to explore exhaustively. Traditional ANNs featured similar issues, and a few research lines in literature proposed solutions based on stochastic optimization of layouts, probing the vast search space with evolutionary algorithms [28,29].

2.2. Evolutionary algorithms

Evolutionary algorithms (EAs) are stochastic optimization techniques loosely inspired by the neo-Darwinian paradigm of natural selection [30]. For complex problems, they usually perform better than traditional optimization methods, due to their capability of escaping local optima in the search space.

In an EA, a single candidate solution is termed *individual*; the set of all candidate solutions that exist at a particular step of the algorithm is called *population*. Evolution proceeds through discrete steps called *generations*. In each of them, the population is first expanded and then collapsed, mimicking the processes of breeding and struggling for survival (Fig. 2). The ability of an individual to solve the target problem is measured by the *fitness function*, which influences the likelihood of a solution to propagate its characteristics to the next generation. In some approaches individuals may die of old age, while in other they remain in the population until replaced by fitter ones.

The word *genome* denotes the whole genetic material of the organism, although its actual implementation strongly differs from one approach to another. The *gene* is the functional unit of inheritance, or, operatively, the smallest fragment of the genome that may be modified in the evolution process. Genes are positioned in the genome at specific positions called *loci*. The alternative genes that may occur at a given locus are called *alleles*.

To generate the offspring, EAs implement both sexual and asexual reproduction. The former is named *recombination*; it involves two or more participants, and implies the possibility for the offspring to inherit different characteristics from different parents. When recombination is achieved through an exchange of genetic material between the parents, it often takes the name of *crossover*. Asexual reproduction may be named *replication*, to indicate that a copy of an individual is created, or, more commonly, *mutation*, to stress that the copy is not exact. All operators exploited during reproduction can be cumulatively called *evolutionary operators*, or *genetic operators* because they act at the genotypical level.

The main limitation of EAs lies in their reliance upon an evaluation function, that is called a considerable number times during each run. Should the fitness function be computationally expensive to run, this might severely impair the usability of the algorithm, especially for applications where timeliness is important. Conversely, if finding the best possible solution is the priority, EAs can deliver better results that the state-of-the-art in traditional optimization methods based on gradient descent [31], albeit in a longer time.

Unique features of EAs are their capability of tackling problems where the structure of a solution is extremely complex, ranging from binary trees [32] to directed graphs [33]; and a straightforward capability of parallel evaluations, as all offspring at a given generation can be evaluated at the same time. For this reason, it is not surprising that EAs are applied to difficult tasks such as evolving the layout and the weights of an ANN, with techniques such as NEAT (Neuro-Evolution of Augmenting Topologies) [28], HyperNEAT [29], and Convolutional Neural Fabrics [34]. These algorithms evolve different topologies by optimizing the connections inside the networks.

The algorithm proposed in this work differs from the aforementioned implementations by allowing the user to specify the numbers of hyperparameters that will be taken into account for the optimization, not only the connections between the neurons. For example the algorithm can take into account not only the best hyperparameters for the CNN instance, but the time to run each instance, or the average in k folds. In this sense, our algorithm can take different cost functions besides the bigger accuracy.

3. Dataset and proposed approach

In this section, we summarize the characteristics of the chosen dataset, briefly outline the theory behind the proposed approach, and delineate the details of the implementation used.

3.1. Dataset

The considered dataset, containing miRNA sequencing isoform values, is taken from the Cancer Genome Atlas.[1] In this work, the following 29 types of cancer are considered: Adrenocortical carcinoma [ACC], Bladder Urothelial Carcinoma [BLCA], Breast invasive carcinoma [BRCA], Cervical squamous cell carcinoma and endocervical adenocarcinoma [CESC], Cholangiocarcinoma [CHOL], Esophageal carcinoma [ESCA], FFPE Pilot Phase II [FPPP], Head

[1] http://cancergenome.nih.gov/.

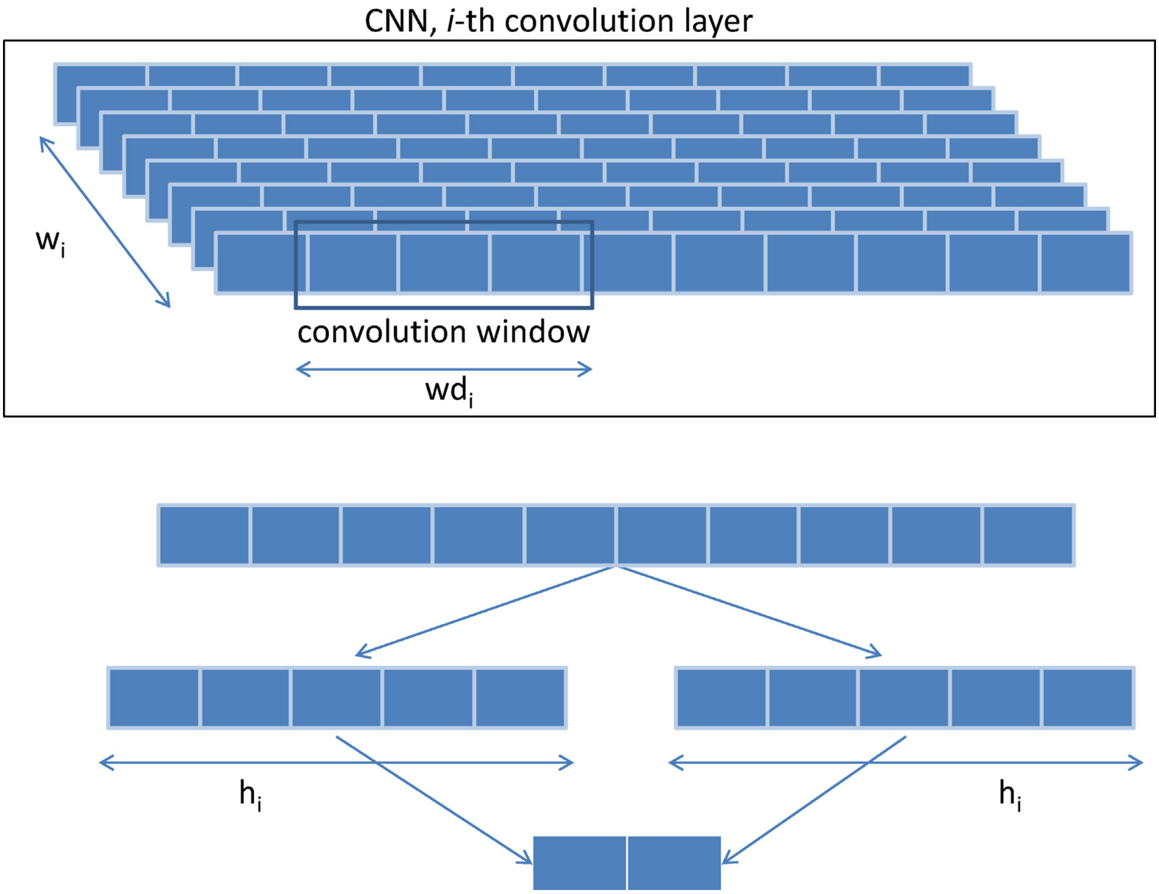


Fig. 3. Visual representation of the hyperparameters w_i,wd_i,h_i in the convolution layer.

and Neck squamous cell carcinoma [HNSC], Kidney Chromophobe [KICH], Kidney renal clear cell carcinoma [KIRC], Kidney renal papillary cell carcinoma [KIRP], Lower Grade Glioma [LGG], Liver hepatocellular carcinoma [LIHC], Lung adenocarcinoma [LUAD], Lung squamous cell carcinoma [LUSC], Lymphoid Neoplasm Diffuse Large B-cell Lymphoma [DLBC], Mesothelioma [MESO], Pancreatic adenocarcinoma [PAAD], Pheochromocytoma and Paraganglioma [PCPG], Prostate adenocarcinoma [PRAD], Sarcoma [SARC], Skin Cutaneous Melanoma [SKCM], Stomach adenocarcinoma [STAD], Testicular Germ Cell Tumors [TGCT], Thymoma [THYM], Thyroid carcinoma [THCA], Uterine Carcinosarcoma [UCS], Uterine Corpus Endometrial Carcinoma [UCEC], Uveal Melanoma [UVM].

These classes are selected among the ones featuring the most case studies in the database, and the resulting dataset contains information for a total of 8129 patients. Using the miRNASeq BCGSC IlluminaHiSeq miRNASeq Level_3, a total of 1046 features for each case study are extracted. As it often happens for real-world datasets, the number of samples per class is highly unbalanced, ranging from 778 for class BRCA, to only 35 for class CHOL. The number of available samples for each class is reported in Table 2.

Data is normalized, and all missing values are replaced with −1, with resulting 1046 rows from each patient's case. From the original 8129 lines, in order to perform a 10-fold cross-validation, the dataset is split into a training set (7316 cases) and a validation set (813 cases), that will be ultimately used to assess the performance of the trained classifier. For each of the 29 classes, case studies are randomly assigned to training or validation during the cross-validation process. It is worth noticing that the proposed approach and each considered algorithm will exploit the training set in a different way: for example, during the training of the CNN, 90% of the training set will be used for training, and 10% to test (validation set) the trained classifier and compute the cross-entropy loss.

3.2. Proposed approach

For the complex task at hand, we propose an EA-optimized CNN classifier. The CNN features 3 layers, with convolution and max pooling. We fixed the number of layers to 3 after several tests as a good balance between performance and time. In our specific case, the input tensor is a single vector with the 1046 features of the dataset. Each convolution layer i can be described by 3 hyperparameters w_i, wd_i, h_i, respectively; output channels, convolution window, and max pooling window, as shown in Fig. 3.

In total we set 3 convolution layers with max pooling, for a total of 9 hyperparameters (w_1, wd_1, h_1, w_2, wd_2, h_2, w_3, wd_3, h_3). In addition to the convolution layers, there's a fully connected rectifier layer with a hyperparameter w_4 for the output channels, with a final softmax layer with dropout. Finally, regularization ($\beta = 0.001$) is added to the cross-entropy loss function, for the weights in each convolution layer. The global model is described in Fig. 4.

The hyperparameters of the model to be optimized are thus w_1, wd_1, h_1, w_2, wd_2, h_2, w_3, wd_3, h_3, w_4. This component is what we consider as a CNN instance.

3.3. EA-based optimization

Given the hyperparameters listed in the previous subsection, our objective becomes finding their optimal (or near-optimal) values, for our target classification problem. In order to reach this goal, we resort to EA-based optimization. A candidate solution in this optimization problem will thus be a sequence of integers, as all considered hyperparameters can only assume integer values. The EA we use in this work follows the classical flowchart presented in Fig. 2, with a few significant problem-specific alterations, described in the following.

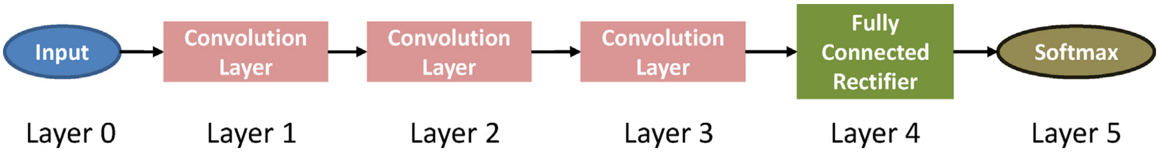


Fig. 4. High-level scheme of the CNN model, to be optimized.

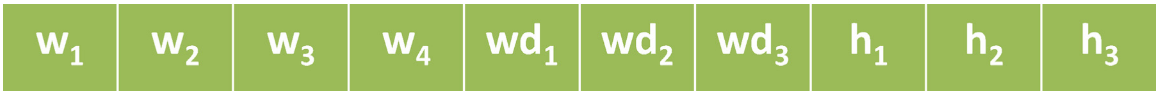


Fig. 5. Genotype of a sample individual in the evolutionary optimization. All considered genes are integers.

First, a maximum (*max*) and a minimum limit (*min*) for the values associated to each gene are set. In our case, *max* is the size of the sliding window, while *min* is 2. These *max* and *min* values are set to avoid errors in the system. While running the system, 2 is the smallest acceptable value for max pooling, as there is no point in using a lower size. On the other hand, the sliding window cannot be greater in size than the input data. Next, a randomly generated set of solutions (individuals) will be created. In our example, the population is 50 individuals ($\mu = 50$), with every individual representing the hyperparameters settings of a CNN instance with its 10 integer values (Fig. 5).

Generated solutions are then evaluated: the fitness function in our problem is the average accuracy of the CNN defined by the aforementioned hyperparameters on a 10-fold validation process. It is worth noticing that this evaluation is computationally expensive, as each individual, representing a CNN, needs to be trained on the 10 dataset generated by cross validation, running a backpropagation algorithm for 1000 iterations in each case.

Subsequently, individuals are sorted, with the first individual having the best accuracy, and the last the worst. This sorting is used as a base for individual reproduction: the first 25 individuals in the ranking will be recombined with an individual selected with uniform probability from the best 50, creating an offspring of size $\lambda = 50$. This mating scheme is inspired by [35], and summarized by Algorithm 1; the recombination operation is depicted in Fig. 6.

Once the two children solutions are created, they are randomly mutated. For each gene in the individual (each integer value in a solution), there is a 10% probability of mutating that parameter to a random integer value, uniformly generated between the *min* and the *max* boundaries for that gene. All the children solutions, representing the offspring for that generation, are then merged with the initial population, to create a pool of size $\mu + \lambda = 100$; the individuals are then sorted by their fitness value, and the worst are removed until the population is again of size $\mu = 50$.

Algorithm 1. Evolutionary Algorithm used in the experience.

```
Create $\mu$ random individuals (the initial population);
Evaluate population;
while stop criterion not reached do
    Sort individuals;
    for ind0 in the best 25 individuals do
        Select random ind1 from population;
        Recombine ind0, ind1;
        Obtain child0, child1;
        Mutate child0, child1;
        Apply heuristic repair to child0, child1;
        Add child0, child1 to offspring;
    Evaluate offspring;
    Merge population and offspring;
    Kill worst individuals;
```

When randomly mutating and recombining solutions, it is possible that the resulting individuals will violate constraints, becoming unfeasible solutions. EAs solve this issue by either discarding the unfeasible individuals without a complete evaluation, or by *repairing* them, modifying the invalid values with a heuristic. In our problem, we choose the latter option, using a linkage tree to represent the constraints on hyperparameters that need to be evaluated together, given the nature of a CNN. In particular, for our case study: w_1, w_2, w_3 and w_4 are independent and can assume values in (2, 256); h_1 and wd_1 are independent and can assume values in (4, 64); while the other hyperparameters are dependent on h_1 and wd_1, see Algorithm 2. These ranges were chosen from several experiments. For example a window size bigger than 256 will take a lot of time to evaluate, thus these ranges are a good commitment between results and performance.

Algorithm 2. Heuristic repair of unfeasible solutions in the EA. This algorithm enforces the proper sizes of the CNN instance, to avoid errors. For example, the convolution window cannot be bigger than the actual data.

```
$s_1 = (\text{input}/h_1)$;
$h_2 = \texttt{random}(2, s_1)$;
$wd_2 = \texttt{random}(2, s_1)$;
$s_2 = (s_1/h_2)$;
while $s_2 < 2$ do
    $h_2 = \texttt{random}(2, s_1)$;
    $wd_2 = \texttt{random}(2, s_1)$;
    $s_2 = (s_1/h_2)$;
if $s_2 == 2$ then
    $h_3 = 2$;
    $wd_3 = 2$;
else
    $h_3 = \texttt{random}(2, s_2)$;
    $wd_3 = \texttt{random}(2, s_2)$;
```

3.4. Implementation

The proposed algorithm is implemented in Python, resorting to the Tensorflow library[2] [36], developed by Google, currently considered the state of the art for deep learning. OAR task manager[3] is used to massively exploit the parallelism potential of EAs. As an individual needs to be evaluated on a 10-fold cross validation process, each fold of the validation can be run on an independent thread, and all individuals can be evaluated at the same time, the evaluation of 50 individuals (the initial population) can be parallelized using 500 threads.

Several programs are devised to monitor and handle the parallel evaluation operations. To adapt to the constraints in the cluster, the EA itself is implemented by two separate entities: *Base* creates the initial population while *Next Generation* incorporates the main body

[2] https://www.tensorflow.org/.

[3] http://oar.imag.fr/.

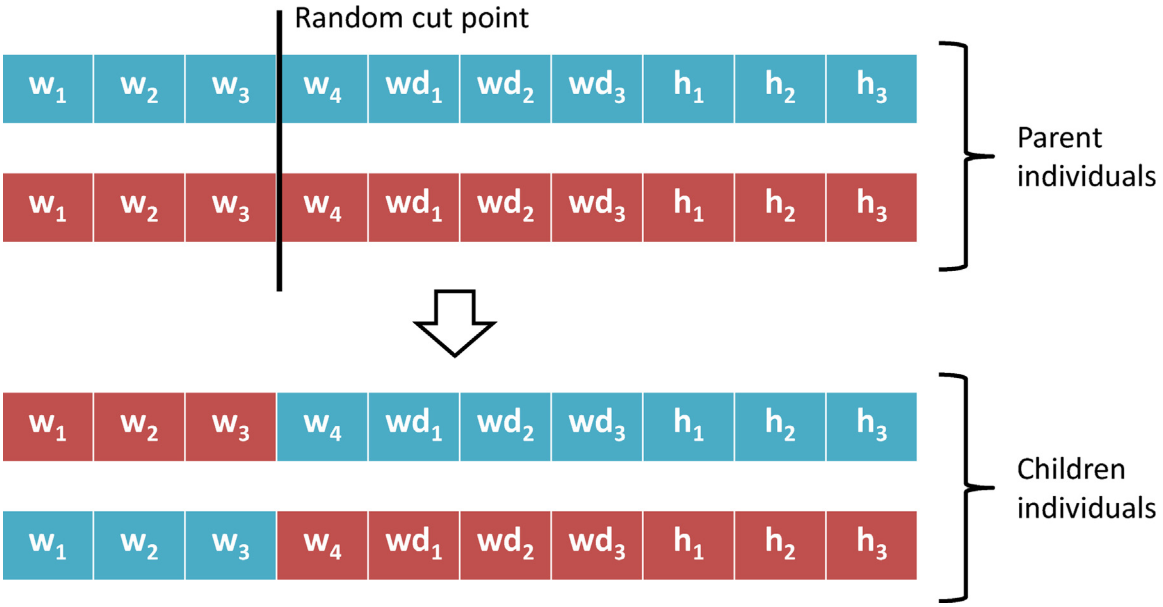


Fig. 6. Recombination procedure for individuals in the proposed approach. The cut point is randomly chosen between all available positions, and the procedure generates two children individuals (to be further mutated in the following steps).

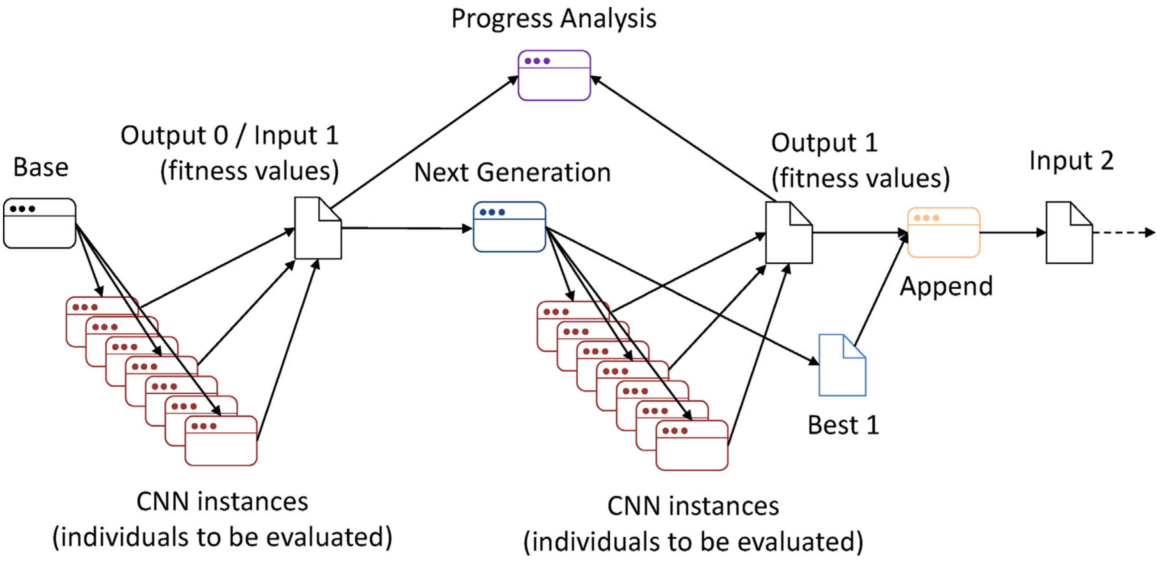


Fig. 7. Representation of the framework designed to parallelize the proposed approach and distribute evaluation over a cluster.

of the EA algorithm, sorting, mating and mutating the individuals; both dispatch the evaluations to the nodes in the cluster.

Once the evaluations from the *Base* program are finished, the output is collected by *Next Generation*, that will continue the process. *Next Generation*, in turn, will produce an output file. A *Progress Analysis* process monitors the individual evaluations currently running on the cluster, showing the user the threads that already terminated, and possible threads in deadlock, that will need to be restarted. The described framework is summarized in Fig. 7.

If all, or enough threads have finished, (decision by the user), the *Append* unit will create the input for the next Generation. The procedure is semi-supervised, because a thread crash in any of the instances should not affect global performance. Such a precaution is necessary, as systems for massively distributed computing cannot be assumed to be completely reliable. The best value in each generation is chosen by the average of the validation set accuracy for folds 0–9.

4. Results and discussion

After 300 evaluations, requiring 14 days of computation on an institutional cluster with 744 cores, the proposed approach is able to find a set of Hyperparameters for the CNN that brings average validation accuracy to 96.6%. Fig. 8 reports accuracy on the training set of the best CNN during the backpropagation process; accuracy on the validation set is reported in Fig. 9. The loss functions over training and validation set are shown in Figs. 10 and 11, respectively. The best hyperparameters found by the EA are: $w_1 = 36$, $wd_1 = 206$, $h_1 = 175$, $w_2 = 251$, $wd_2 = 41$, $h_2 = 4$, $w_3 = 2$, $wd_3 = 133$, $h_3 = 7$, $w_4 = 2$. Classification accuracy by class is provided in Table 2.

The proposed methodology is then tested against 21 reference, state-of-the-art classifiers, whose implementations are taken from the `scikit-learn`[4] Python package [37]. The list of classifiers used

[4] http://scikit-learn.org/stable/.

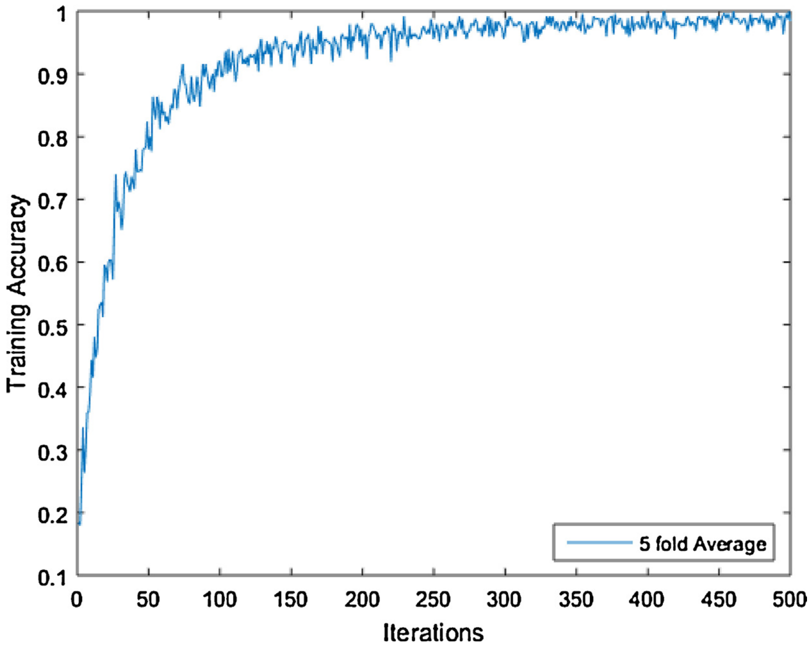


Fig. 8. Training accuracy for the final CNN, average for folds 0–4.

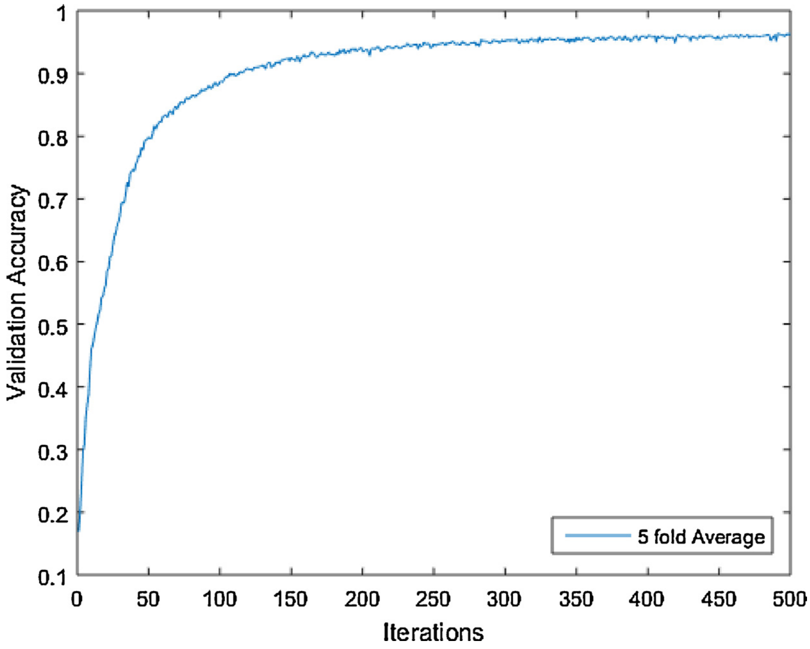


Fig. 9. Validation accuracy for the final CNN, average for folds 0–4.

for the comparison and their respective performance on the validation set are reported in Table 1. As for the proposed approach, a 10-fold cross validation procedure is used for all the reference classifiers.

While it is commonly acknowledged by machine learning experts that mean and standard deviation in performance of a classification algorithm over a cross-validation cannot be interpreted with their classical meaning in statistics [26], nevertheless they can be used as a reference to separate the behavior of two algorithms. In this case, the distribution of results obtained by the proposed approach is separable by the distribution of its closest competitor (`OneVsOneClassifier`) with $p < 0.01$ using a Kolmogorov–Smirnov test. Among the considered techniques, the proposed methodology is thus the best classifier for the target problem.

Additionally, this comparison makes it possible to perform a meta-analysis of the classifiers' accuracy on the different cancer types in the dataset, reported in Table 2. Overall, the set of considered classifiers is able to obtain satisfying classification rates on most classes, but some are noticeably easier to tackle. For example, LIHC and PCPG, where 58 and 50 classifiers respectively are able to reach a 100% classification rate. For cancer types such as CHOL, DLBC, FPPP, and MESO, on the other hand, no classifier can reach 100%; this result, however, can be explained by the relatively low number of samples for each mentioned class in the dataset. ESCA and HNSC, however, prove to be just as hard, but feature a way more generous number of samples, 200 and 488 respectively. This

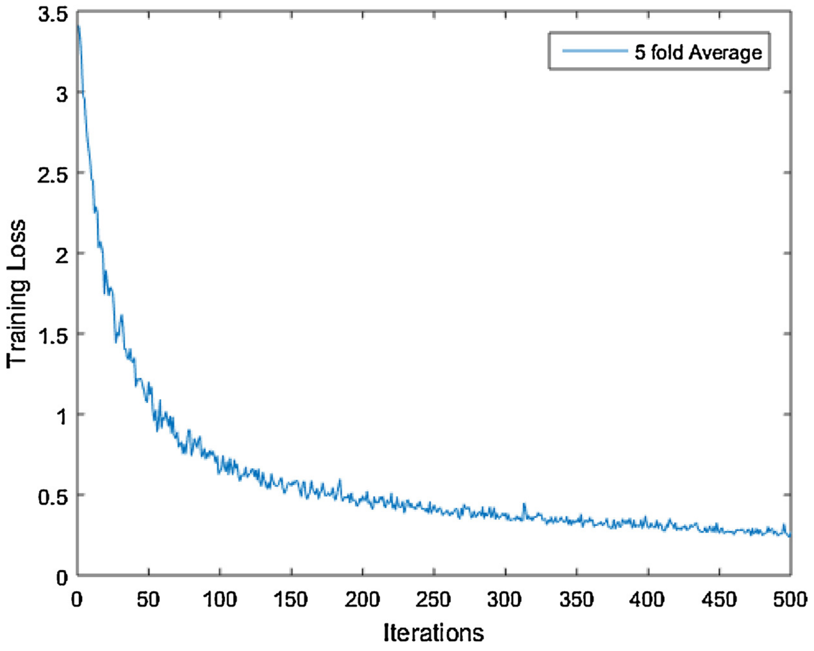


Fig. 10. Training loss for the final CNN, average for folds 0–4.

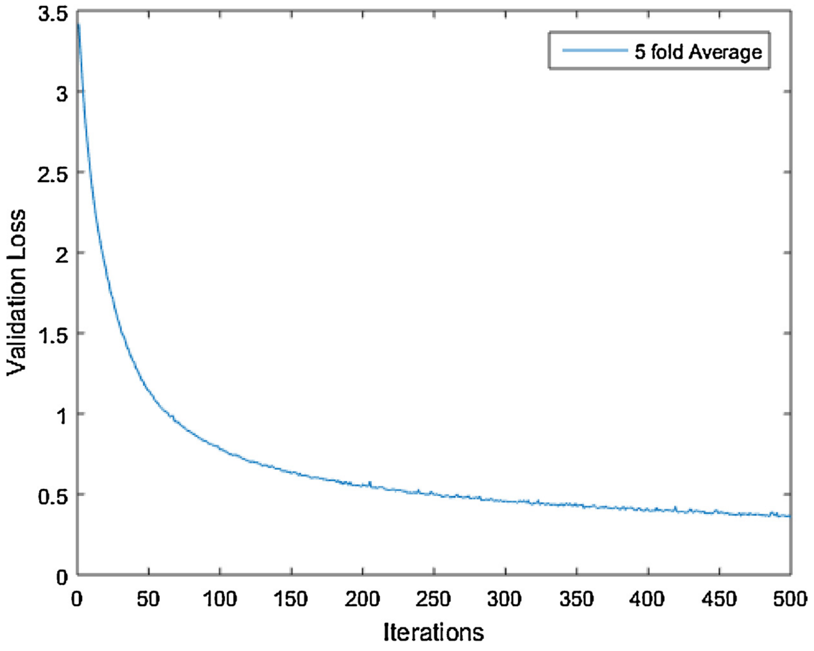


Fig. 11. Validation loss for the final CNN, average for folds 0–4.

is an indication that correctly classifying such illnesses based on RNA biomarkers, only, might not be a viable solution.

5. Conclusions

In this paper, we presented a novel approach to classify cancer types from miRNA biomarkers isoform values. The proposed methodology exploits an EA-optimized CNN, optimized and trained exploiting the inherent parallelism of evolutionary computation. The approach is then tested on a real-world dataset, containing 1046 biomarkers for 8129 patients, affected by 29 different types of cancer, and compared against 21 different state-of-the-art classifiers.

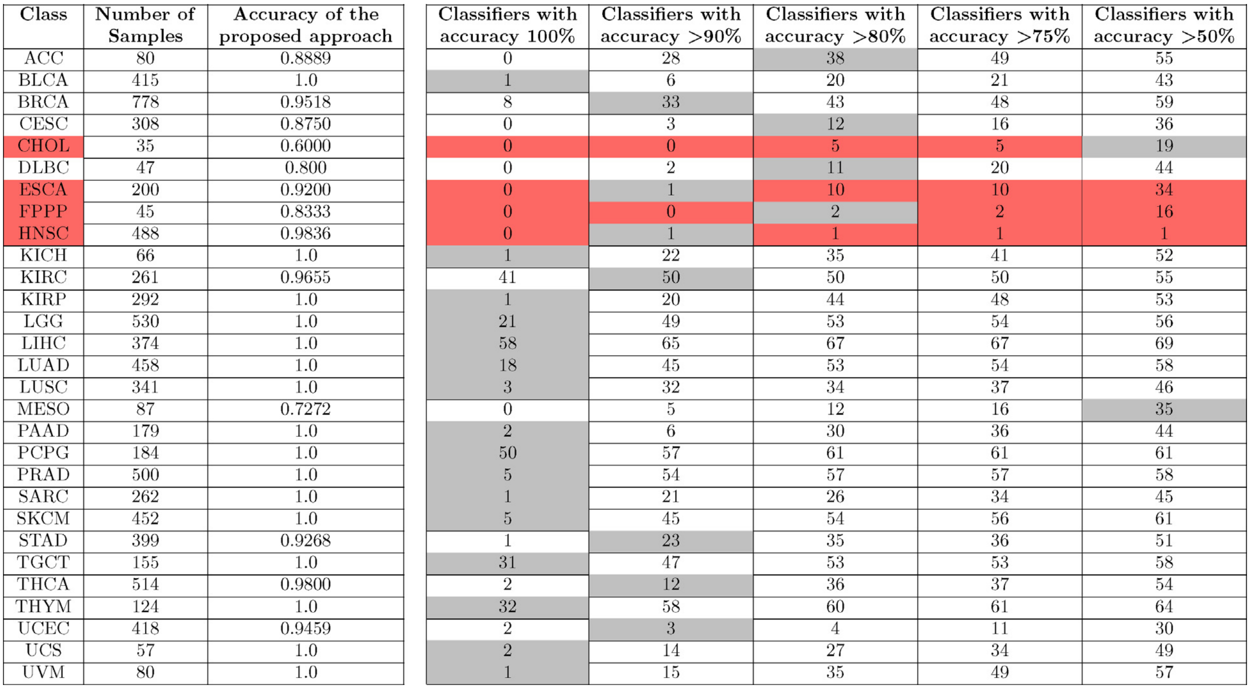
The proposed approach is proven able to outperform the classifiers used as a reference. Results from the comparison are then used for a meta-analysis of the different classes in the dataset, showing which classes are harder to detect, and providing some indications on the cancer types for which miRNA biomarkers might be a valuable resource.

While the results presented in this work are still preliminary, they are nevertheless a step towards non-intrusive tests for early detection of cancer in patients. Future works will focus on a more in-depth analysis of the features of each class, and enhancing the classification process to include data from healthy patients.

Table 1
Accuracy of the classifiers listed in the naive assessment, grouped by family, using a **10-fold cross validation**. As the task is extremely difficult, the vast majority of classifiers with default hyperparameters perform poorly. Methods marked with * create an ensemble of classifiers, each one targeting a specific class, or discriminating between two specific classes in the problem. `OneVsOne`, `OneVsRest`, and `OutputCode` classifiers have been tested with ensembles composed of all the previously listed classifiers. The best results are reported, for `OneVsOne(LogisticRegressionCV)`, `OneVsRest(AdaBoostClassifier)`, and `OutputCode(GradientBoostingClassifier)`, respectively.

Name	Test score	Stdv
`ensemble`		
`AdaBoostClassifier` [38]	0.2889	0.0130
`BaggingClassifier` [39]	0.8851	0.0084
`GradientBoostingClassifier` [40]	0.9253	0.0059
`RandomForestClassifier` [41]	0.8727	0.0140
`linear_model`		
`LogisticRegression` [42]	0.6766	0.0199
`LogisticRegressionCV`	0.9405	0.0106
`PassiveAggressiveClassifier` [43]	0.7072	0.0561
`RidgeClassifierCV` [44]	0.7706	0.0239
`SGDClassifier` [45]	0.6800	0.0361
`naive_bayes` [46]		
`BernoulliNB`	0.7749	0.0097
`GaussianNB`	0.7375	0.0215
`MultinomialNB`	0.2005	0.0094
`neighbors`		
`KNeighborsClassifier` [47]	0.8521	0.7825
`NearestCentroid` [48]	0.5179	0.4885
`RadiusNeighborsClassifier`	0.1426	0.1499
`svm`		
`SVC` [49]	0.1358	0.0087
`tree`		
`DecisionTreeClassifier` [50]	0.8148	0.0169
`ExtraTreeClassifier` [51]	0.5509	0.0184
`multiclass`		
`OneVsOneClassifier*` [52]	0.9594	0.0079
`OneVsRestClassifier*` [52]	0.9385	0.0108
`OutputCodeClassifier*` [53]	0.9311	0.0100
`Proposed approach`		
`EA-optimized CNN`	0.9660	0.0086

Table 2
Performance of the proposed approach, along with a meta-analysis of classifier performances for each type of cancer. Considering the various classifier ensembles used in the experiments, the total number of classifiers tested in our work is 76. Cells highlighted in grey show the accuracy range the proposed approach is placed into, for each class. Lines highlighted in red reveal cancer types that prove hard to correctly classify for all considered algorithms. Class HNSC proved to be particularly challenging, with the proposed approach being the only classifier able to break the 50% accuracy threshold.



Class	Number of Samples	Accuracy of the proposed approach	Classifiers with accuracy 100%	Classifiers with accuracy >90%	Classifiers with accuracy >80%	Classifiers with accuracy >75%	Classifiers with accuracy >50%
ACC	80	0.8889	0	28	38	49	55
BLCA	415	1.0	1	6	20	21	43
BRCA	778	0.9518	8	33	43	48	59
CESC	308	0.8750	0	3	12	16	36
CHOL	35	0.6000	0	0	5	5	19
DLBC	47	0.800	0	2	11	20	44
ESCA	200	0.9200	0	1	10	10	34
FPPP	45	0.8333	0	0	2	2	16
HNSC	488	0.9836	0	1	1	1	1
KICH	66	1.0	1	22	35	41	52
KIRC	261	0.9655	41	50	50	50	55
KIRP	292	1.0	1	20	44	48	53
LGG	530	1.0	21	49	53	54	56
LIHC	374	1.0	58	65	67	67	69
LUAD	458	1.0	18	45	53	54	58
LUSC	341	1.0	3	32	34	37	46
MESO	87	0.7272	0	5	12	16	35
PAAD	179	1.0	2	6	30	36	44
PCPG	184	1.0	50	57	61	61	61
PRAD	500	1.0	5	54	57	57	58
SARC	262	1.0	1	21	26	34	45
SKCM	452	1.0	5	45	54	56	61
STAD	399	0.9268	1	23	35	36	51
TGCT	155	1.0	31	47	53	53	58
THCA	514	0.9800	2	12	36	37	54
THYM	124	1.0	32	58	60	61	64
UCEC	418	0.9459	2	3	4	11	30
UCS	57	1.0	2	14	27	34	49
UVM	80	1.0	1	15	35	49	57

Acknowledgements

This work was partially supported by the INSERM-ITMO cancer project 'LIONS' No. BIO2015-04. The results published in our work are in whole or part based upon data generated by the TCGA Research Network: http://cancergenome.nih.gov/.

References

[1] M.M. Akhtar, L. Micolucci, M.S. Islam, F. Olivieri, A.D. Procopio, Bioinformatic tools for microRNA dissection, Nucleic Acids Res. 44 (1) (2016) 24–44.

[2] A. Bhattacharya, Y. Cui, SomamiR 2.0: a database of cancer somatic mutations altering microRNA-ceRNA interactions, Nucleic Acids Res. 44 (D1) (2015) D1005–D1010.

[3] J. Verigos, A. Magklara, Revealing the complexity of breast cancer by next generation sequencing, Cancers 7 (4) (2015) 2183–2200.

[4] H. Gómez-Rueda, E. Martínez-Ledesma, A. Martínez-Torteya, R. Palacios-Corona, V. Trevino, Integration and comparison of different genomic data for outcome prediction in cancer, BioData Mining 8 (1) (2015) 1.

[5] I. Koturbash, W.H. Tolleson, L. Guo, D. Yu, S. Chen, H. Hong, W. Mattes, B. Ning, microRNAs as pharmacogenomic biomarkers for drug efficacy and drug safety assessment, Biomark. Med. 9 (11) (2015) 1153–1176.

[6] J. Ribas, X. Ni, M. Castanares, M.M. Liu, D. Esopi, S. Yegnasubramanian, R. Rodriguez, J.T. Mendell, S.E. Lupold, A novel source for miR-21 expression through the alternative polyadenylation of VMP1 gene transcripts, Nucleic Acids Res. 40 (14) (2012) 6821–6833.

[7] P. Puerta-Gil, R. García-Baquero, A.Y. Jia, S. Oca na, M. Alvarez-Múgica, J.L. Alvarez-Ossorio, C. Cordon-Cardo, F. Cava, M. Sánchez-Carbayo, miR-143, miR-222, and miR-452 are useful as tumor stratification and noninvasive diagnostic biomarkers for bladder cancer, Am. J. Pathol. 180 (5) (2012) 1808–1815.

[8] W.-N. Wan, Y.-Q. Zhang, X.-M. Wang, Y.-J. Liu, Y.-X. Zhang, Y.-H. Que, W.-J. Zhao, P. Li, Down-regulated miR-22 as predictive biomarkers for prognosis of epithelial ovarian cancer, Diagn. Pathol. 9 (1) (2014) 178.

[9] G. Shao, S. Ji, A. Wu, C. Liu, M. Wang, P. Zhang, Q. Jiao, Y. Kang, DNAzyme-based probe for circulating microRNA detection in peripheral blood, Drug Des. Dev. Ther. 9 (2015) 6109–6117.

[10] V.Y. Shin, E.K. Ng, V.W. Chan, A. Kwong, K.-M. Chu, A three-miRNA signature as promising non-invasive diagnostic marker for gastric cancer, Mol. Cancer 14 (1) (2015) 202.

[11] X. Zhang, Y. Zhang, X. Liu, A. Fang, P. Li, Z. Li, T. Liu, Y. Yang, L. Du, C. Wang, MicroRNA-203 is a prognostic indicator in bladder cancer and enhances chemosensitivity to cisplatin via apoptosis by targeting Bcl-w and survivin, PLOS ONE 10 (11) (2015) e0143441.

[12] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps, arXiv preprint arXiv:1312.6034.

[13] Y. LeCun, C. Cortes, C.J. Burges, The MNIST Database of Handwritten digits, 1998 http://yann.lecun.com/exdb/mnist/.

[14] A. Lopez-Rincon, M. Bendahmane, B. Ainseba, On 3D numerical inverse problems for the bidomain model in electrocardiology, Comput. Math. Appl. 69 (4) (2015) 255–274.

[15] A. Lopez, M. Cienfuegos, B. Ainseba, M. Bendahmane, PSO with tikhonov regularization for the inverse problem in electrocardiography, in: International Conference on Artificial Evolution (Evolution Artificielle), Springer, 2013, pp. 256–270.

[16] A. Lopez Rincon, M. Bendahmane, B. Ainseba, Two-step genetic algorithm to solve the inverse problem in electrocardiography for cardiac sources, Comput. Methods Biomech. Biomed. Eng.: Imaging Vis. 2 (3) (2014) 129–137.

[17] I.C. Gould, A.M. Shepherd, K.R. Laurens, M.J. Cairns, V.J. Carr, M.J. Green, Multivariate neuroanatomical classification of cognitive subtypes in schizophrenia: a support vector machine learning approach, NeuroImage: Clin. 6 (2014) 229–236.

[18] E. Astrand, P. Enel, G. Ibos, P.F. Dominey, P. Baraduc, S.B. Hamed, Comparison of classifiers for decoding sensory and cognitive information from prefrontal neuronal populations, PLOS ONE 9 (1) (2014) e86314.

[19] Y. Wang, J.O. Goh, S.M. Resnick, C. Davatzikos, Imaging-based biomarkers of cognitive performance in older adults constructed via high-dimensional pattern regression applied to MRI and PET, PLOS ONE 8 (12) (2013) e85460.

[20] Y. Höller, J. Bergmann, A. Thomschewski, M. Kronbichler, P. Höller, J.S. Crone, E.V. Schmid, K. Butz, R. Nardone, E. Trinka, Comparison of EEG-features and classification methods for motor imagery in patients with disorders of consciousness, PLOS ONE 8 (11) (2013) e80479.

[21] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Cognit. Model. 5 (3) (1988) 213–220.

[22] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[23] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[24] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[25] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, Proceedings of the 27th International Conference on Machine Learning (ICML-10) (2010) 807–814.

[26] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016 http://www.deeplearningbook.org.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958 http://jmlr.org/papers/v15/srivastava14a.html.

[28] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2002) 99–127.

[29] J. Gauci, K.O. Stanley, A case study on the critical role of geometric regularity in machine learning, AAAI (2008) 628–633.

[30] K.A. De Jong, Evolutionary Computation: A Unified Approach, MIT Press, 2006.

[31] R. Fletcher, Practical Methods of Optimization, John Wiley & Sons, 2013.

[32] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, Vol. 1, MIT Press, 1992.

[33] M.F. Brameier, W. Banzhaf, Linear Genetic Programming, Springer Science & Business Media, 2007.

[34] S. Saxena, J. Verbeek, Convolutional neural fabrics, Advances in Neural Information Processing Systems (2016) 4053–4061.

[35] J. Heaton, Introduction to Neural Networks With Java, Heaton Research, Inc, 2008.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org, 2015 http://tensorflow.org/.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[38] J. Zhu, H. Zou, S. Rosset, T. Hastie, Multi-class adaboost, Stat. Interface 2 (3) (2009) 349–360.

[39] L. Breiman, Pasting small votes for classification in large databases and on-line, Mach. Learn. 36 (1–2) (1999) 85–103.

[40] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. (2001) 1189–1232.

[41] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[42] D.R. Cox, The regression analysis of binary sequences, J. R. Stat. Soc. Ser. B (Methodol.) (1958) 215–242.

[43] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive-aggressive algorithms, J. Mach. Learn. Res. 7 (March) (2006) 551–585.

[44] A.N. Tikhonov, On the stability of inverse problems, Dokl. Akad. Nauk SSSR 39 (1943) 195–198.

[45] M.A. Hearst, S.T. Dumais, E. Osman, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.

[46] H. Schütze, Introduction to information retrieval, Proceedings of the International Communication of Association for Computing Machinery Conference (2008).

[47] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. 46 (3) (1992) 175–185.

[48] R. Tibshirani, T. Hastie, B. Narasimhan, G. Chu, Diagnosis of multiple cancer types by shrunken centroids of gene expression, Proc. Natl. Acad. Sci. U. S. A. 99 (10) (2002) 6567–6572.

[49] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, Proceedings of the Fifth Annual Workshop on Computational Learning Theory, ACM (1992) 144–152.

[50] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.

[51] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.

[52] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2007.

[53] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. 2 (1995) 263–286.