Pattern Recognition Letters 133 (2020) 247–255

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Multiple Instance Learning with Genetic Pooling for medical data analysis ☆

Kamanasish Bhattacharjee[a,*], Millie Pant[a], Yu-Dong Zhang[b], Suresh Chandra Satapathy[c]

[a] Department of Applied Science and Engineering, Indian Institute of Technology Roorkee, India
[b] Department of Informatics, University of Leicester, Leicester LE1 7RH, UK
[c] School of Computer Engineering, Kalinga Institute of Industrial Technology, Bhubaneswar, Odisha, India

ARTICLE INFO

Article history:
Received 10 October 2019
Revised 25 January 2020
Accepted 28 February 2020
Available online 5 March 2020

Keywords:
Multiple Instance Learning (MIL)
Genetic Algorithm (GA)
Pooling
Neural network

ABSTRACT

Multiple Instance Learning is a weakly supervised learning technique which is particularly well suited for medical data analysis as the class labels are often not available at desired granularity. Multiple Instance Learning through Deep Neural Networks is relatively a new paradigm in machine learning. The most important part of Multiple Instance Learning through Deep Neural Networks is designing a trainable pooling function which determines the instance-to bag relationship. In this paper, we propose a Multiple Instance pooling technique based on Genetic Algorithm called Genetic Pooling. In this technique, instance labels inside a bag are optimized by minimizing bag-level losses. The main contribution of the paper is that the bag level pooling layer for generating attention weights for bag instances are replaced by random initialization of attention weights and finding the optimized attention weights through Genetic Algorithm.

1. Introduction

Machine learning and artificial intelligence has brought tremendous advancement in the field of data analysis be it text data or image and video data. The medical domain is where a huge part of machine learning research is focused. In most of the real-world datasets, the data is acquired from real scenarios rather than controlled experiments. Hence, there can be inaccuracy, incompleteness, inconsistency, imbalances in the acquired data. For example, most patients wouldn't perform all examinations in hospital, only necessary and required ones; different doctors approach the diagnosis in different ways and suggest different experiments but finally arrive at the same conclusion. In real world scenario, decisions in medical diagnosis are taken based on a few valuable and informative attributes i.e. features, rather than analysing the entire patient record. Hence, the computational models for extracting significant information from huge amounts of low-quality and incomplete data to produce robust diagnostic results, has become a very important topic in the medical domain.

In case of medical image data, the issue is more relatable. A common problem of implementing machine learning for medical images is the lack of annotated data or weakly annotated data. The problem of weakly annotated data is that an image is typically described by a single label (benign/malignant) or a Region Of Interest (ROI) is roughly given but labels for each pixel or patch is not given. Also, technically, assigning class label for each pixel or patch is time-consuming, computationally expensive or not possible, but clinical decisions of the overall condition of a patient i.e. global labeling for entire images are available more readily. For example, by examining a brain tumor MRI of a patient, a doctor or a radiologist can come to the conclusion whether the tumor is benign or malignant. But this is not the case for a machine. A machine scans the whole image pixel by pixel and then takes a decision. Hence, for machine learning, labeling each pixel or patch is an important process. These problems become more prominent in medical Big Data analysis. Amin et al. [1] have done extensive experiments on eight challenging brain tumor Big datasets for brain tumor segmentation through deep convolutional neural networks.

In such cases of incomplete data or low-quality data or weakly annotated data, the Multiple Instance Learning (MIL) comes into picture. MIL is a recent machine-learning paradigm that is particularly well suited to medical image and video analysis tasks [2]. The global label of an image is not applicable to every pixel or every patch of that image. For example, in a brain tumor MRI, only a certain portion of the image represents the tumor and not the whole image. For a machine, it is important to learn which pixels or patches are representing the tumor and which are not for efficient future predictions.

☆ Editor: Prof. G. Sanniti di Baja.

* Corresponding author.
E-mail address: kbhattacharjee@as.iitr.ac.in (K. Bhattacharjee).

https://doi.org/10.1016/j.patrec.2020.02.025
0167-8655/

In MIL, a machine is trained on bag-level classification where a bag consists of many instances for which the labels are unknown. The objective of MIL is to predict the label of a new bag through training a classifier. And to predict the label of a bag, first one has to learn what effects this bag-level classification i.e. how instances effects in deciding the label of a bag, which instances are contributing towards deciding bag labels. This task is to determine the instance-to bag relationship. And to determine this relationship, one has to design a pooling function which correlates instance labels to bag label.

This paper deals with this problem of designing an MIL pooling function through Genetic Algorithm (GA) [3] which is termed as Genetic Pooling.

The rest of the paper is divided into five sections. Section 2 discusses MIL in details and its implementation through deep neural networks. Section 3 gives a brief description of Genetic Algorithm. Section 4 describes the proposed method. Section 5 discusses the experiments done and corresponding results. Finally, Section 6 concludes the paper.

2. Multiple Instance Learning

The goal of conventional supervised learning is to map each instance in the input dataset X to the label set Y and learn the corresponding function. For example, in binary classification, the model learns to predict the value of target variable $y \in \{0, 1\}$ for a given instance $x \in X$ in input dataset. This instance is nothing but a feature vector or a tuple in the input dataset. In Multiple Instance Learning, the task is to map a bag of instances $\{x_1, x_2, \ldots, x_k\}$ to a label instead of a single instance. Here k could vary from bag to bag for the same problem. The common rule is for each bag, the bag is labeled positive if there is at least one positive instance otherwise it is labeled negative. Furthermore, we assume that individual labels exist for the instances within a bag, i.e. $y_1, y_2, \ldots, y_K$ where $y_k \in \{0, 1\}$ for $k = 1, 2, \ldots, K$, however, there is no access to those labels. Assumptions of MIL can be represented as -

$$Y = \begin{cases} 0, & iff \ \sum_k y_k = 0 \\ 1, & otherwise \end{cases} \tag{1}$$

MIL was first proposed by Dietterich et al. in [4] to solve the problem of drug discovery. For a deeper understanding of MIL, [5,6] can be referred. Fig. 1 depicts a pictorial representation of a general architecture of Deep Neural Networks. Depending on the type of input, the first layer can be Embedding layer (for text data, this layer embeds the instances into a dense vector) or Convolutional layer (for image or video data, this layer extracts various features). MIL through Deep Neural Networks is relatively a new paradigm in machine learning. [7–11] are some of the examples of applications of MIL through Deep Learning.

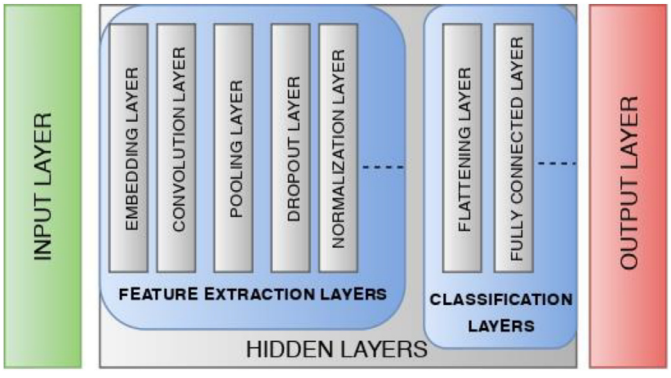

Fig. 1. General architecture of Deep Neural Network.

Ramon *et.al.* utilized neural networks first time in [12] for MIL. They determined instance probabilities and evaluated bag probabilities by a log-sum-exp operator in pooling layer. Zhou et al. [13] introduced a network called BP-MIP, using max operator in pooling layer replacing the log-sum-exp operator. Later, the same researchers [14] improved BP-MIP with two extensions BP-MIP-DD and BP-MIP-PCA that are enhanced with Diverse Density [15] and PCA [16] respectively. Wang et al. proposed the MI-Net [17] which primarily focuses to learn the bag embeddings, unlike the aforementioned works that focus on inferring instance labels.

The MIL pooling used in neural networks can be categorized as non-trainable and trainable. Non-trainable techniques are simpler - max pooling, mean pooling, sum pooling, log-sum-exp pooling [12] etc. Trainable pooling techniques are more complex such as gated attention-based pooling, attention-based pooling [18] adaptive pooling [19], dynamic pooling [20] etc. Various researchers used the various pooling methods together for neural networks i.e. in MI-Net [21], max, mean and log-sum-exp pooling are used for instance-level pooling; in AMI-Net [22], sum pooling is used as the instance-level pooling while attention-based pooling is applied on bag-level; in AMI-Net+ [23], Wang et al. proposed a self-adaptive MIL pooling technique which uses max, mean, sum and log-sum-exp pooling.

3. Genetic algorithm

Genetic Algorithm (GA) was proposed in 1992 by John Holland [3]. It is a technique mimicking the biological evolutionary process to solve complex optimization problems. The main operations of GA are selection, crossover and mutation. Parents are chosen or selected from individuals of a generation and through crossover and mutation, children are produced which can be potential individuals for next generation. The population evolves towards an optimal solution throughout the generations. Flowchart for GA is presented in Fig. 2.

3.1. Operations in Genetic Algorithm (GA)

Selection: The first operation in after initialization and fitness calculation is selection of chromosomes for mating pool which will produce the children for next generation. This can be done through various methods: a. Roulette Wheel Selection, b. Rank Selection, c. Steady State Selection, d. Tournament Selection, e. Elitism Selection

Crossover: After selecting the mating pool, two of the parents are mated to produce children. Again, there are various techniques for crossover: a. Single point crossover, b. Two point crossover, c. k point crossover, d. Uniform crossover etc. It is observed that, mating good parents can produce better children. This is equivalent to exploiting the search space for convergence.

Mutation: Mutation introduces random changes in children produced by crossover. This is equivalent to exploring the search space so that the solution doesn't get stuck in local optima.

4. Proposed method

The proposed MIL pooling method is described in this section.

4.1. Attention based pooling

In [18], Ilse et al. proposed to use a weighted average of instances (low dimensional embeddings) where weights are determined by a neural network. Let $H = \{h_1, h_2, \ldots, h_k\}$ be a bag of K instances. Then, the MIL pooling is –

$$z = \sum_{k=1}^{K} a_k h_k \tag{2}$$

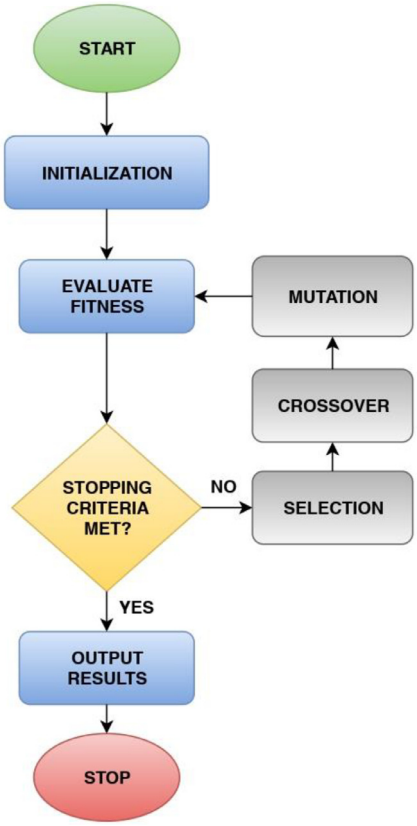

Fig. 2. Genetic Algorithm (GA) Flowchart.

where a_k is weight corresponding to h_k instance in the bag. Now, finding the optimum combination of these weights is where Genetic Algorithm comes into picture.

4.2. Genetic Algorithm for Pooling

In the previous pooling methods, extra dense layers are used to generate the attention weights. Hence, the model to be trained becomes bigger in size. Using GA for pooling, we make the model smaller and easier to train. Here, a population of attention weights are generated randomly between [0,1]. in the first generation. The model is trained for each set of attention weights in the population and error is calculated through the corresponding loss functions. Then through generations, these attention weights are optimized to achieve minimum loss.

4.2.1. Selection

In each generation, best half of the population of attention weights are selected as mating pool. Here, elitism selection is used.

4.2.2. Crossover

Set of consecutive 2 individuals are chosen as parents in the mating pool and are mated to produce 2 children. Single point crossover is used. Crossover points are randomly generated for each tuple in an individual. Through crossover operation, same number of children is generated as of mating pool.

4.2.3. Mutation

Mutation points are randomly generated for each tuple of individual in children pool. At the mutation point, the gene is replaced by a random number. After crossover and mutation operations, the worst half of the previous population is replaced by the children pool. The process is repeated for predefined number of generations.

5. Experiments

The experiments are executed on a real-world medical text dataset *Western Medicine (WM)*.

5.1. Datasets

The *Western Medicine (WM)* dataset is collected by Medicinovo Inc. in Beijing. It is a clinical dataset for schizophrenia patients. It has the data of total 3927 patients. There are 88 medical features in total, such as unemployed, married, high levels of prolactin etc. For each patient, there are at most 21 features and at least 5 features. There exist only 224 positive labels out of 3927. The positive rate is only 0.057. Hence, the dataset is extremely imbalanced. Objective of the machine learning model is to predict whether a patient will relapse to schizophrenia within three months. An example and a snapshot of the dataset are presented in Tables 1 and 2 respectively.

Algorithm
Genetic Pooling.

for each bag

- initialize population with P bags
- initialize instance weights a_k for each bag
- set maximum iteration number max_iter
- set iteration number $t = 1$
- set bag number in population $p = 1$
- while t<=max_iter
  - while p<=P
    - $z^p \leftarrow \sum_k a_k^p h_k^p$
    - Run feed-forward pass on neural network
    - calculate loss
    - calculate fitness value fit_p
  - end while
  - choose the fittest half of the population depending on the fitness values
  - perform crossover operation between chosen individuals depending on the Crossover type and Crossover Probability
  - perform mutation operation
  - replace the worst half by new children
- end while

end for
return a_k

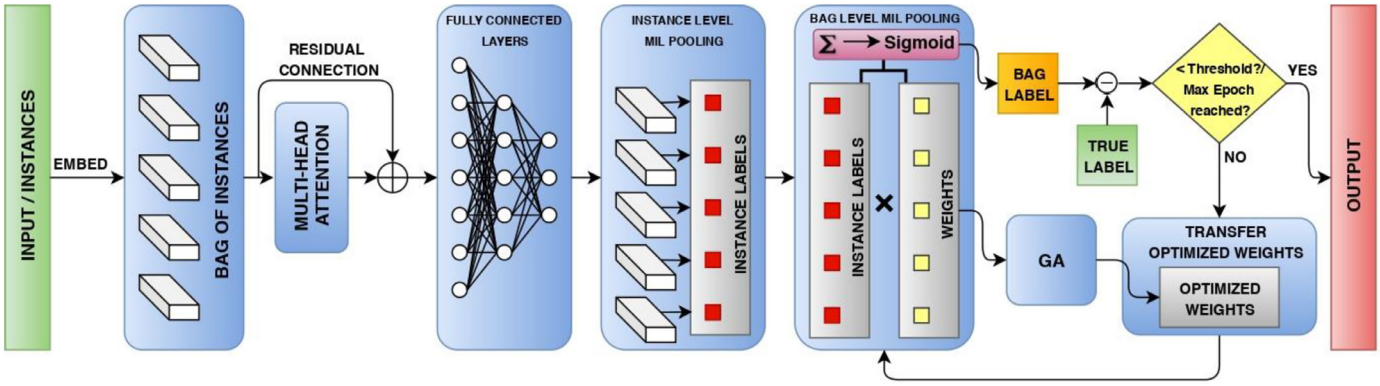

Fig. 3. AMI-Net with Genetic Pooling.

Table 1
Example of WM dataset.

Features	Diagnosis
Unmarried, MECT<=1, Total course<1095 days, LOS<10 days, Onset age<17, Personal income 3000–5000, Lorazepam tablets=0.5mg	Schizophrenia relapse
Married, MECT 1~10, Total course 1095~5840 days, LOS 25 ~49 days, Onset age 1–10, Personal income 1000~3000, Risperidone<=1 mg, Haloperidol injection 5 mg, High levels of corticotrophin, hyperglycemia, High levels of prolactin	No relapse

5.2. Experimental setup

For Genetic Pooling, the architecture of AMI-Net [22] is used. The neural network is pictorially depicted in Fig. 3. It takes a bag of symptoms (i.e., combination of instances) as input and through embedding layer, each instance is mapped to a dense vector as the instance embeddings. In the next layer, multi-head attention [24] is used. In different organs or body parts, various symptoms of a disease are often related to each other. Each one of these organs or body parts can be considered as subspace. Multi-head attention captures this intra-relationship of instances in different embedding subspaces. Also, to improve the model robustness in case of low-quality data, multihead-attention links the standard expressions of symptoms and non-standard ones. Then layer normalization [25] and residual connection is used to mine the instance correlations. Then a set of dense layers is used to estimate the instance representations and instance-level pooling is applied on instance level to obtain the bag representation.

For bag level pooling Genetic Algorithm is used to optimize the attention weights. Finally, sigmoid function is used on bag levels to get the outputs and binary cross-entropy is used to calculate the loss between desired and calculated predictions. As this is a binary classification, binary cross-entropy is used for the final loss calculation. Adam optimizer with learning rate = 0.00001, $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = e^{-8}$ is used to optimize the network. For fair comparison, the experiments are carried out on 5 folds of the dataset. The maximum number of epochs or iterations is set at 50 and 100.

5.3. Evaluation metrics

Area Under Curve (AUC): Area under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. It is one of the most important evaluation metrics for checking any classification model's performance. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model at classification. By analogy, higher the AUC, better the model is at distinguishing between patients with disease and no disease.

Average Precision (AP): AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{3}$$

where P_n and R_n are the precision and recall at the nth threshold.

Table 2
Snapshot of WM dataset.

	Features	Value			
1	Income_0	0	0	1	0
2	Income_1	0	0	0	0
3	Income_2	0	0	1	0
4	Income_3	0	1	0	0
5	Income_4	0	0	0	0
6	Marriage_0	1	1	0	1
7	Marriage_1	0	0	0	0
8	Marriage_2	1	0	0	0
9	Marriage_3	0	1	0	0
10	Job_0	1	0	0	1
11	HDL_0	1	0	0	0
12	HDL_1	0	0	0	0
13	Prolactin_0	0	0	0	0
14	Prolactin_1	1	0	0	0
15	Prolactin_2	0	0	0	0

(*continued on next page*)

Table 2 (*continued*)

	Features	Value			
16	Glucose_0	0	0	0	0
17	Glucose_1	1	0	0	0
18	Glucose_2	0	0	0	0
19	Triglyceride_0	0	0	0	0
20	Triglyceride_1	0	0	0	0
21	Triglyceride_2	0	0	0	0
22	Hemameba_0	0	0	0	0
23	Hemameba_1	0	0	0	0
24	ACTH_0	1	0	0	0
25	ACTH_1	0	0	0	0
26	ACTH_2	0	0	0	0
27	Length_0	0	0	0	0
28	Length_1	1	1	1	1
29	Length_2	0	0	0	0
30	LOS_0	1	0	0	0
31	LOS_1	0	0	1	0
32	LOS_2	0	1	0	1
33	MECT_0	1	0	0	0
34	MECT_1	0	1	1	1
35	MECT_2	0	0	0	0
36	Olan_0	0	0	0	0
37	Olan_1	0	0	0	0
38	Olan_2	0	0	0	0
39	CDT_0	0	0	0	0
40	CDT_1	0	0	0	0
41	CDT_2	0	0	0	1
42	AO_0	1	0	0	1
43	AO_1	0	1	1	0
44	AO_2	0	0	0	0
45	Risperidone_0	0	0	0	0
46	Risperidone_1	0	0	0	0
47	Risperidone_2	0	0	0	0
48	Gap_0	0	0	0	0
49	Gap_1	1	0	0	0
50	Gap_2	0	1	0	0
51	Total Length_0	1	0	0	0
52	Total Length_1	0	0	1	0
53	Total Length_2	0	1	0	1
54	Lorazepam_0	1	0	1	1
55	Lorazepam_1	0	0	0	0
56	Lorazepam_2	0	0	0	0
57	Amisulpride_0	0	0	0	0
58	Amisulpride_1	0	0	0	1
59	Amisulpride_2	0	0	0	0
60	Haloperidol_0	1	0	0	0
61	Haloperidol_1	0	0	0	0
62	Haloperidol_2	0	0	0	0
63	Aripiprazole_0	0	1	0	0
64	Aripiprazole_1	0	0	0	0
65	Aripiprazole_2	0	0	0	0
66	TAG_0	0	0	0	0
67	TAG_1	0	0	0	0
68	TAG_2	0	0	0	0
69	PET_0	0	0	0	0
70	PET_1	1	0	0	0
71	PET_2	0	0	0	0
72	Perphenazine_0	0	0	0	0
73	Perphenazine_1	0	0	0	0
74	Perphenazine_2	0	0	0	0
75	Seroquel_0	0	0	0	0
76	Seroquel_1	0	0	0	0
77	Seroquel_2	0	0	0	0
78	Ziprasidone_0	0	0	0	0
79	Ziprasidone_1	0	0	0	0
80	Ziprasidone_2	0	0	0	0
81	Penfluridol_0	0	0	0	0
82	Penfluridol_1	0	0	0	0
83	Penfluridol_2	0	0	0	0
84	RisperdalConsta_0	0	0	0	0
85	RisperdalConsta_1	0	0	0	0
86	RisperdalConsta_2	0	0	0	0
87	Sulpiride_0	0	0	0	0
88	Sulpiride_1	0	0	0	0
89	Sulpiride_2	0	0	0	0
	Result	0 = No Relapse	0 = No Relapse	1 = Relapse	1 = Relapse

Accuracy: It is the fraction of predictions model correctly predicted. It is calculated as

$$Accuracy = \frac{True\ Positive + True\ Negative}{All\ Predictions} \tag{4}$$

Balanced Accuracy: Accuracy is not a good performance measurement metrics in case of imbalanced data. Balanced Accuracy overcomes this disadvantage. It normalizes the true positives by number of positive samples and true negatives by number of negative samples, and then divides their sum by two. It is defined as the average of recall obtained on each class. For binary classification, it is calculated as

$$Balanced\ Accuracy = \frac{Recall + Specificity}{2} \tag{5}$$

Negative Predictive Value (NPV): As the dataset is extremely imbalanced with positive rate 0f only 0.57, hence Negative Predictive Value (NPV) is used instead of Precision. NPV indicates out of all the negative instances or instances with 0 labels in case of binary classification predicted by the classifier are actually negative i.e. in this case patients with 'No Relapse'. It is calculated as –

$$NPV = \frac{True\ Negative}{True\ Negative + False\ Negative} \tag{6}$$

Specificity or True Negative Rate (TNR): As the dataset is extremely imbalanced with positive rate 0f only 0.57, hence Specificity of True Negative Rate (TNR) is used instead of Recall. Specificity indicates out of all the negative instances or instances with 0 labels in case of binary classification, how many the classifier has correctly identified as negative i.e. in this case patients with 'No Relapse'. It is calculated as –

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive} \tag{7}$$

Zero One Loss: It indicates the fraction of misclassifications.

Hamming Loss: It indicates the fraction of labels that are incorrectly predicted. This is same as zero-one loss in case of multiclass classification but different in case of multi-label classification.

5.4. Hardware and software specifications

Experiments have been conducted on Spyder 4.0.1 Integrated Development Environment (IDE) with Python 3.7.3 through Anaconda distribution on an Intel Xeon 2.5 GHz system with 16 GB RAM, Nvidia Quadro 2000 GPU and 64-bit Windows 10 Operating System.

5.5. Results and discussion

The results are compared with various bag-level pooling methods on the AMI-Net - gated attention-based pooling, max pooling, mean pooling, sum pooling and log-sum-exp pooling. The results are listed in Tables 3 and Table 4 for number of epochs 50 and 100 respectively. The best results are highlighted in each column. The training curves for various bag-level poolings are presented in Figs. 4–15 which show the decaying value of loss for a specified number of epochs or iterations.

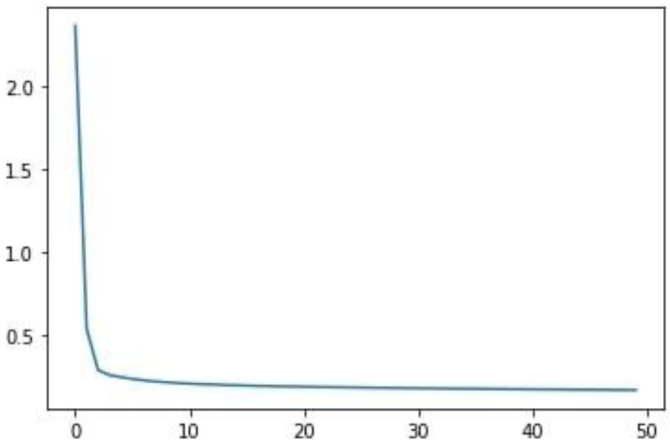

Fig. 4. Training curve for Max pooling (Epoch = 50).

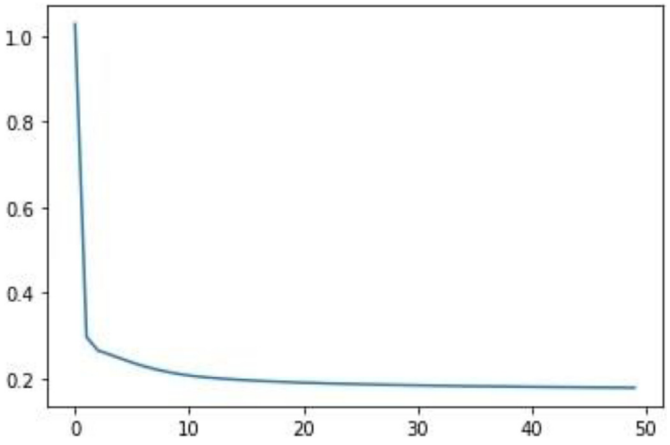

Fig. 5. Training curve for Mean pooling (Epoch = 50).

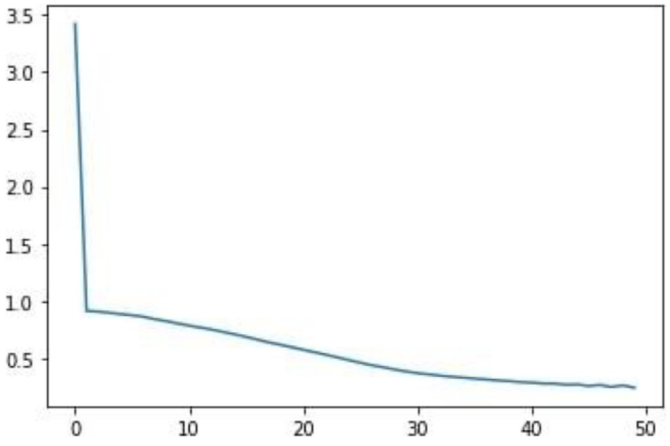

Fig. 6. Training curve for Sum pooling (Epoch = 50).

Table 3
Experimental results on AMI-Net for WM dataset (Epoch = 50).

Bag-level MIL Pooling methods	AUC	AP	Accuracy	Balanced Accuracy	NPV	Specificity / TNR	Zero one loss	Hamming loss
Max Pooling	0.7213278	0.1821588	0.9405848	0.4989986	0.9423619	0.9979972	0.0594152	0.0594152
Mean Pooling	0.7515591	0.2026722	0.9421571	0.5092896	0.9435605	0.9983227	0.0578429	0.0578429
Sum Pooling	0.5589434	0.0905177	0.8905848	0.5316447	0.9464094	0.9383928	0.1094152	0.1094152
Log-sum-exp pooling	0.7316502	0.1845512	0.9415282	0.5042304	0.9429735	0.9983327	0.0584718	0.0584718
Gated Attention pooling	0.7381759	0.1677373	0.9424716	0.5	0.9424716	1	0.0575284	0.0575284
Genetic pooling	0.8234809	0.2432881	0.9424716	0.5	0.9424716	1	0.0575284	0.0575284

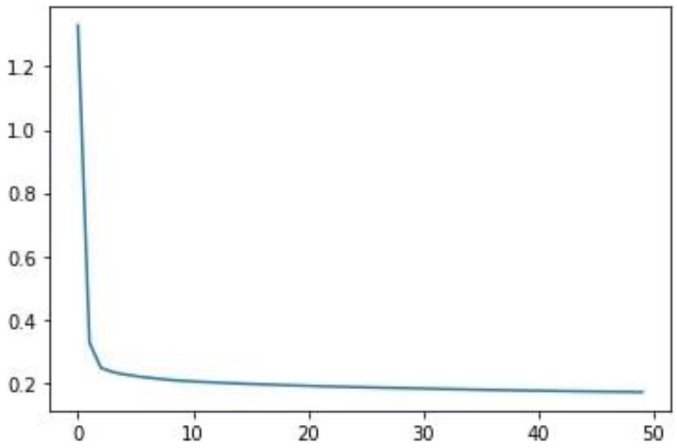

Fig. 7. Training curve for Log-sum-exp pooling (Epoch = 50).

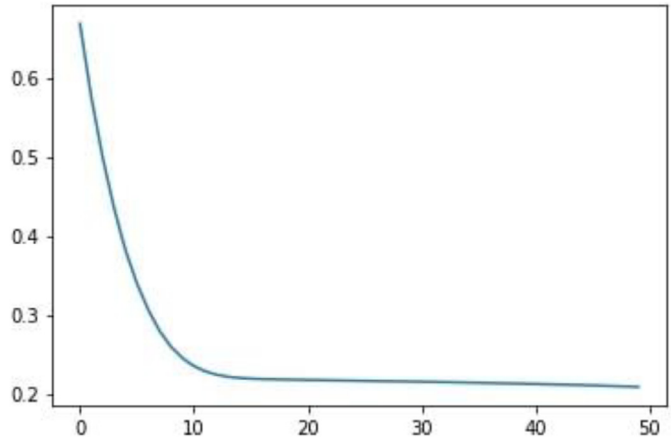

Fig. 8. Training curve for Gated Attention pooling (Epoch = 50).

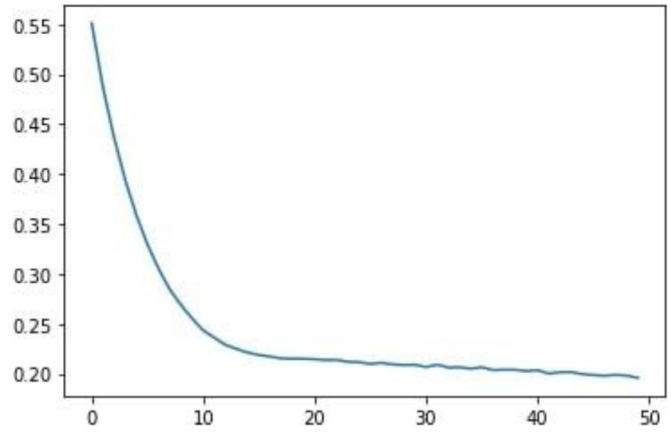

Fig. 9. Training curve for Genetic pooling (Epoch = 50).

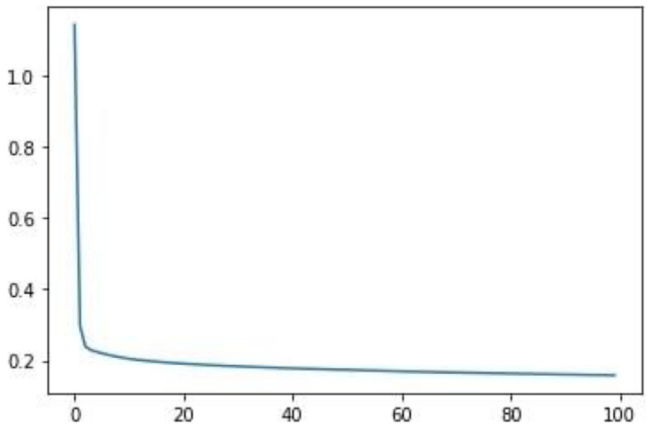

Fig. 10. Training curve for Max pooling (Epoch = 100).

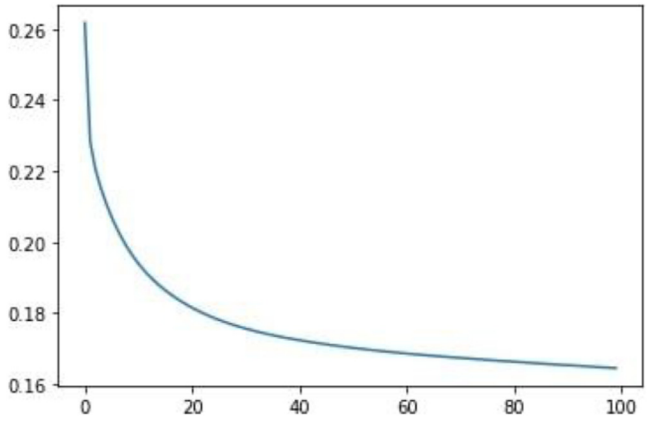

Fig. 11. Training curve for Mean pooling (Epoch = 100).

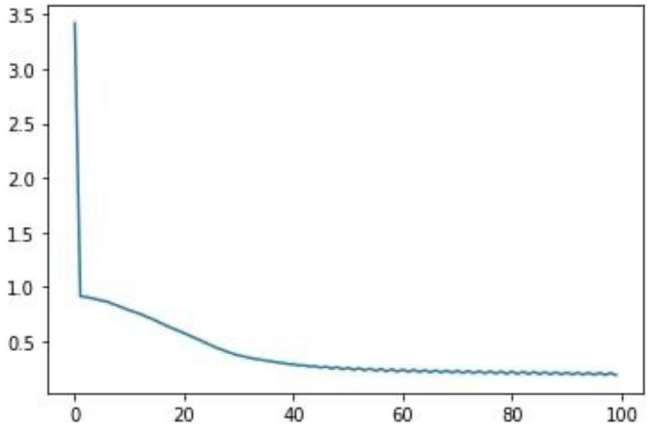

Fig. 12. Training curve for Sum pooling (Epoch = 100).

Table 4
Experimental results on AMI-Net for WM dataset (Epoch = 100).

Bag-level MIL Pooling methods	AUC	AP	Accuracy	Balanced Accuracy	NPV	Specificity / TNR	Zero one loss	Hamming loss
Max Pooling	0.7283375	0.1879182	0.9377546	0.5026506	0.9427535	0.9943237	0.0622454	0.0622454
Mean Pooling	0.7521901	0.2006572	0.9418427	0.5091218	0.9435413	0.9979871	0.0581573	0.0581573
Sum Pooling	0.5755536	0.1003711	0.9201445	0.5401024	0.9472548	0.9697003	0.0798555	0.0798555
Log-sum-exp pooling	0.7359605	0.1830831	0.9405848	0.5107879	0.9437524	0.9963193	0.0594152	0.0594152
Gated Attention pooling	0.7544704	0.1958218	0.9424716	0.5023977	0.9427499	0.9996672	0.0575284	0.0575284
Genetic pooling	0.8862781	0.3782898	0.9424716	0.5023977	0.9427499	0.9996672	0.0575284	0.0575284

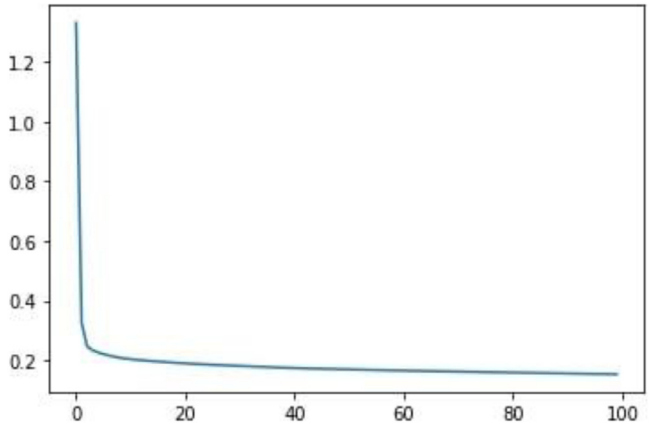

Fig. 13. Training curve for Log-sum-exp pooling (Epoch = 100).

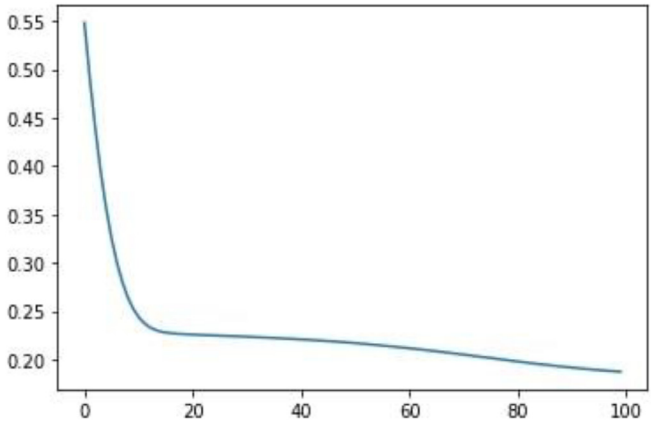

Fig. 14. Training curve for Gated Attention pooling (Epoch = 100).

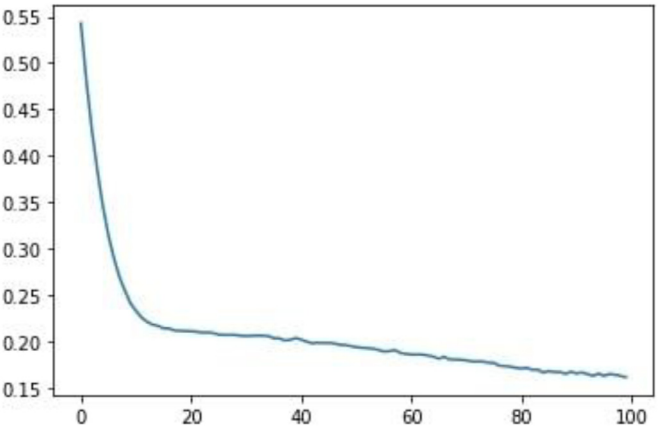

Fig. 15. Training curve for Genetic pooling (Epoch = 100).

From the results recorded in Tables 3 and 4, it can be seen that the proposed Genetic Pooling method shows the best AUC and AP; shows the same performance as Gated Attention Pooling in terms of Accuracy, Specificity and losses. But, in terms of Balanced Accuracy and NPV, sum pooling outperforms both of them. As stated earlier in Section 5.3, AUC is the most important evaluation metrics for checking any classification model's performance and the proposed method outperforms all the other methods in this criteria. For experiments with 50 epochs, almost all the methods achieve similar final training loss values. But, for experiments with 100 epochs, Genetic Pooling is able to achieve the lowest training loss value. Hence, it is an indication that increasing the number of epochs will enhance its performance. This can be tested by simply increasing the number of epochs.

6. Conclusion

Main objective of the paper is to formulate an MIL pooling function for Multiple Instance Learning through deep neural network based on Genetic Algorithm. The proposed method is termed Genetic Pooling. The method is tested on Western Medicine (WM) dataset and compared with other MIL pooling techniques. Its performance in terms of AUC and AP establishes its superiority over other methods. This is a new application of metaheuristics and evolutionary algorithms. To the best of our knowledge, we are the first to propose Genetic Algorithm in designing MIL pooling function for deep neural network which is significantly used for medical data analysis. This surely is a new aspect to be explored more in the future researches. Other metaheuristics than GA, other variants of GA, hybrid algorithms, parallelizing the GA etc. can be used for designing the pooling function. Also, fine tuning the parameters of GA was not in the scope of this paper which can be further explored to achieve even better classification results. The experiments have been done on a specific architecture – AMI-Net and the Western Medicine (WM) dataset is used is this paper. More network architectures as well as datasets can be employed for research in this topic.

Declaration of Competing Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

References

[1] J. Amin, M. Sharif, M. Yasmin, S.L. Fernandes, Big data analysis for brain tumor detection: deep convolutional neural networks, Futur. Gener. Comput. Syst. 87 (2018) 290–297.
[2] G. Quellec, G. Cazuguel, B. Cochener, M. Lamard, Multiple-Instance learning for medical image and video analysis, IEEE Rev. Biomed. Eng. 10 (2017) 213–234.
[3] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–73.
[4] T.G. Dietterich, R.H. Lathrop, T. Lozano-Pérez, Solving the multiple instance problem with axis-parallel rectangles, Artif. Intell. 89 (1997) 31–71.
[5] M.-A. Carbonneau, V. Cheplygina, E. Granger, G. Gagnon, Multiple instance learning: a survey of problem characteristics and applications, Pattern Recognit. 77 (2018) 329–353.
[6] J. Amores, Multiple instance classification: review, taxonomy and comparative study, Artif. Intell. 201 (2013) 81–105.
[7] W. Zhu, Q. Lou, Y.S. Vang, X. Xie, in: Deep Multi-Instance Networks With Sparse Label Assignment For Whole Mammogram Classification, Springer Verlag, 2017, pp. 603–611. Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics).
[8] X. Liu, S. Bi, X. Ma, J. Wang, Multi-Instance convolutional neural network for multi-shot person re-identification, Neurocomputing 337 (2019) 303–314.
[9] M. Liu, J. Zhang, E. Adeli, D. Shen, Landmark-based deep multi-instance learning for brain disease diagnosis, Med. Image Anal. 43 (2018) 157–168.
[10] Z. Yan, Y. Zhan, Z. Peng, S. Liao, Y. Shinagawa, S. Zhang, D.N. Metaxas, X.S. Zhou, Multi-Instance Deep Learning, Discover discriminative local anatomies for bodypart recognition, IEEE Trans. Med. Imaging 35 (2016) 1332–1343.
[11] J. Wu, Y. Yu, C. Huang, K. Yu, Deep multiple instance learning for image classification and auto-annotation, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., IEEE Computer Society, 2015, pp. 3460–3469.
[12] J. Ramon, L. De Raedt, Multi instance neural networks, ICML-2000 work, Attrib. Relational Learn (2000) 53–60.
[13] Z. Zhou, M.L. Zhang, Neural networks for multi-instance learning, in: Int. Conf. Intell. Inf. Technol., Beijing, China, 2002, pp. 455–459.
[14] M.L. Zhang, Z.H. Zhou, Improve multi-instance neural networks through feature selection, Neural Process. Lett. 19 (2004) 1–10.
[15] O. Maron, T. Lozano-perez, A framework for multiple-instance learning, Adv. Neural Inf. Process. Syst (1998) 570–576.
[16] S. Wold, K. Esbensen, P. Geladi, Principle component analysis, Chemom. Intell. Lab. Syst. 2 (1987) 37–52.
[17] X. Wang, Y. Yan, P. Tang, X. Bai, W. Liu, Revisiting multiple instance neural networks, Pattern Recognit. 74 (2018) 15–24.
[18] M. Ilse, J.M. Tomczak, M. Welling, Attention-based deep multiple, Instance Learn. (2018). arXiv: 1802.04712.
[19] D. Liu, Y. Zhou, X. Sun, Z. Zha, W. Zeng, Adaptive pooling in multi-instance learning for web video annotation, in: IEEE Int. Conf. Comput. Vis. Work., IEEE, 2017, 2017, pp. 318–327.

[20] Y. Yan, X. Wang, J. Fang, W. Liu, J. Huang, J. Zhu, I. Takeuchi, Deep multi-instance learning with dynamic pooling, in: Asian Conf. Mach. Learn., 2018, pp. 1–16.
[21] X. Wang, Y. Yan, P. Tang, X. Bai, W. Liu, Revisiting multiple instance neural networks, Pattern Recognit. 74 (2018) 15–24.
[22] Z. Wang, J. Poon, S. Sun, S. Poon, Attention-based multi-instance neural network for medical diagnosis from incomplete and low quality data, in: Proc. Int. Jt. Conf. Neural Networks, Institute of Electrical and Electronics Engineers Inc., 2019.
[23] Z. Wang, J. Poon, S. Poon, AMI-Net+: a novel multi-instance neural network for medical diagnosis from incomplete and imbalanced data, (2019). arXiv:1907.01734.
[24] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. (2017) 5998–6008.
[25] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer Normalization, 2016. arXiv: 1607.06450.