EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

Research in Computing Science

Vol. 152 No. 5
May 2023

Research in Computing Science

RESEARCH IN COMPUTING SCIENCE, *Año 23, Volumen 152, No. 5, mayo de 2023, es una publicación mensual, editada por el Instituto Politécnico Nacional, a través del Centro de Investigación en Computación. Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, Ciudad de México, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor responsable: Dr. Grigori Sidorov. Reserva de Derechos al Uso Exclusivo del Título No. 04-2019-082310242100-203 ISSN: en trámite, ambos otorgados por el Instituto Nacional del Derecho de Autor. Responsable de la última actualización de este número: el Centro de Investigación en Computación, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Fecha de última modificación 05 de mayo de 2023.*

RESEARCH IN COMPUTING SCIENCE, *Year 23, Volume 152, No. 5, May 2023, is a monthly publication edited by the National Polytechnic Institute through the Center for Computing Research. Av. Juan de Dios Bátiz S/N, Esq. Miguel Othón de Mendizábal, Nueva Industrial Vallejo, C.P. 07738, Mexico City, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor in charge: Dr. Grigori Sidorov. Reservation of Exclusive Use Rights of Title No. 04-2019-082310242100-203. ISSN: pending, both granted by the National Copyright Institute. Responsible for the latest update of this issue: the Computer Research Center, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Last modified on May 5, 2023.*

Computational Intelligence and Applications

Rocio Erandi Barrientos-Martínez,
Marcela Quiroz-Castellanos,
Héctor Gabriel Acosta-Mesa,
Efrén Mezura-Montes (eds.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2023

ISSN: in process

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional “Adolfo López Mateos”, Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX, DBLP and Periodica

Electronic edition

ISSN 1870-4069

Table of Contents

Page

ISSN 1870-4069

HOK-Means: A Hybrid and Parallel Clustering Algorithm Oriented to Big Data

Joaquín Pérez Ortega, Nancy Salgado Antunez[1], Sandra Silvia Roblero Aguilar[1, 2], Yasmín Hernández[1], Nelva Nely Almanza Ortega[2], Vanesa Landero Nájera[3]

[1] Tecnológico Nacional de México, Centro Nacional de Investigación y Desarrollo Tecnológico, Mexico

[2] Tecnológico Nacional de México, Instituto Tecnológico de Tlalnepantla, Mexico

[3] Universidad Politécnica de Apodaca, Computer Systems, Mexico

jpo_cenidet@yahoo.com.mx, {m20ce047, yasmin.hp}@cenidet.tecnm.mx, {sandra.ra, nelva.ao}@tlalnepantla.tecnm.mx, vlandero@upapnl.edu.mx

Abstract. Using the K-Means algorithm to analyze large datasets demands much time and computational resources. An approach to reducing the time is to parallelize the algorithm. However, the processing time is still high to process large datasets like those presented in Big Data. In this sense, a hybrid clustering algorithm with parallel execution is proposed to solve large datasets. The proposed algorithm is inspired by a highly efficient sequential variant of the K-Means algorithm named O-K-Means. Experimental results with synthetic and real large datasets with conventional equipment showed that Hybrid OK-Means reduces the time to 7.54 times compared to the sequential variant. It is noteworthy that as the size of the datasets grows, the speedup results tend to improve, preserving the quality of the solution. Highlighted, the proposal presented in this article shows a significant improvement in speedup, surpassing the works reported by other researchers.

Keywords: Big data, clustering, k-means, parallel programming.

1 Introduction

Technological development has caused an exponential increase in data generation and storage in recent years. Therefore, there is an interest in extracting knowledge from these massive amounts of data since it would allow us to make better decisions [1, 2].

One of the ways to gain insight from large amounts of data is by identifying clustering patterns. To perform clustering of massive amounts of data (Big Data) with standard tools is generally limited by computing resources [3]. In this regard, our contribution is to provide a strategy to deal with the problem of clustering objects according to their attributes in a Big Data environment.

Clustering techniques have been used in various areas, such as data science, data engineering, and business [1]. Clustering consists of partitioning a set of n objects in k non-empty subsets called clusters in such a way that the objects in one cluster have attributes similar to each other and, at the same time, different from the objects in other clusters [2].

In this article, a parallel algorithm, which we call Hybrid O-K-Means (HOK-Means), is proposed, which is inspired by an improvement of the K-Means algorithm called O-K-Means[2]. This variant has shown to be highly efficient in solving large datasets of the Big Data type.

The structure of this paper is organized as follows. Section 2 presents related work. Section 3 describes the algorithms used in this research and shows the improvement proposal. Section 4 describes performance metrics and the design of the experiment. Section 5 reports the results obtained. Conclusions and ideas for future research are given in Section 6.

Algorithm 1: HOK-Means

Master Node
Initialization:
  $P := \{p_1, \ldots, p_t\}$; // The set of available processors is allocated
  $N := \{x_1, \ldots, x_n\}$; // Load Dataset
  $M := \{\mu_1, \ldots, \mu_k\}$; // Initialize random centroids
  $U := 0.72$; //Threshold value for determining O-K-Means convergence;
  Send start order, centroids (M), *and* n_o to slaves;
Slave Node
Classification:
  For $x_i \in N_p$ and $\mu_k \in M${
    Calculate the Euclidean distance from each x_i to each k centroid;
    Assign object x_i to the nearest centroid μ_k;
  Calculate the number of objects that changed groups;
  Send *MR* results matrix;
  Send the number of objects that changed the group v_r
Master Node
Receive end status and information from all slave nodes;
Calculate the percentage of change $\gamma_r = 100(v_r/n)$;
Calculate centroids:
  Calculate the centroid M;
Convergence:
  If ($\gamma \leq U$):
    Stop the algorithm;
  Otherwise:
    Go to step **7**;
End of algorithm

Table 1. Datasets used in experiments.

id	1	2	3	4	5	6	7	8	9	10	11	DSAS	EPCS
n	2	2	2	2	2	1	1	1	0.5	1.2	1	1,140,000	2,049,280
d	2	4	6	5	7	4	7	10	11	5	11	45	7

2 Related Work

According to [3, 4], the clustering problem type K-Means when k>=2 or d>=2 is NP-Hard, so obtaining an optimal solution for a dataset of considerable size is intractable. An approach to reducing the processing time of sequential algorithms is parallelizing them. The standard K-Means algorithm has been parallelized using different platforms and architectures.

For example, in Map Reduce [3, 4], Spark [5], Peer to Peer [6], GPU [7, 8], Multi-core [9], and FGPA [10], among others. Improvements have been made to the K-Means algorithm that increases its efficiency. However, there are few parallelized versions of improvements to the K-Means algorithm. Three of the parallelized enhancements are described below.

In [9], a parallel variant of K-Means++ is proposed, using multi-core on a single computer. This work presents the division of the dataset in such a way that each processor works a subset of objects in parallel. The best speedup achieved was 7.7, with a computer of 12 cores and a parallel efficiency of 0.64.

In [11], the K-Means++ algorithm is implemented on three different architectures for shared memory: multicore CPU, high-performance GPU, and the massively multithreaded Cray XMT platform. The objective of this research was to show a performance relationship of each platform with the number of objects, attributes, and groups.

In [5], an improvement in the initialization phase of the K-Means algorithm named Canopy is proposed. This improvement consists of selecting the set of centroids using the weighted density method to reduce the impact of outliers on the clustering results. The best result is a speedup of 4.0 and a parallel efficiency of 2.0.

The size of the largest dataset is 1.72 GB. The algorithm is parallelized using the Spark platform with eight cores. Although there are already algorithms that parallelize K-Means improvements, the efficiency they report is limited, and it is foreseeable that the solution of large datasets is long time-consuming.

In this sense, the proposal presented in this article shows a significant improvement in speedup, surpassing the works mentioned. Consequently, enabling the solution of larger datasets in less time.

3 New HOK-Means Algorithm

This section briefly describes the background that gave rise to the algorithm and then describes the proposed algorithm in detail. K-Means is an iterative method that consists

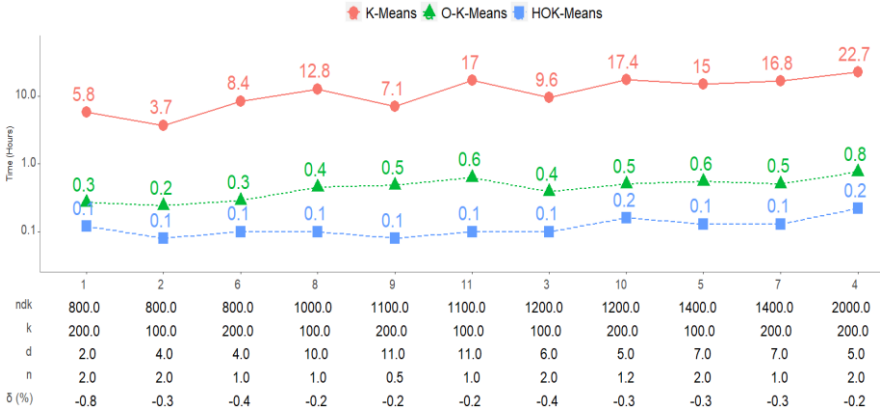


Fig. 1. Solution time with synthetic datasets.

of partitioning a set of N objects into $k \geq 2$ non-empty groups so that the objects of each group have similar attributes and, at the same time, are different from the objects of any other group [2].

There are different improvements to the K-Means algorithm [12]. In [2], a heuristic called O-K-Means is proposed, which accelerates the convergence process, stopping the algorithm when the total number of objects that change the cluster in an iteration is less than a threshold. This value expresses a relationship between the computational effort and the quality of the solution.

Therefore, although the gain in solution quality is minimal, the algorithm invests the same computational effort to perform each iteration. The proposed stopping threshold decreases the number of iterations while preserving most of the quality of the solution of the K-Means algorithm. In [2], the experimental results presented achieved an average execution time reduction of 93.88%, with only a 0.4% loss in clustering quality compared to standard K-Means.

HOK-Means is a parallel version of the O-K-Means sequential algorithm, and for both algorithms, the same outputs will be obtained with the same inputs. HOK-Means is an algorithm that parallelizes the classification and convergence phases of the sequential O-K-Means algorithm.

To describe the HOK-Means algorithm, we will rely on Algorithm 1, which shows the sequence of execution of instructions in the master node and the slave nodes.

Initialization: The initialization of the master node is shown in lines 2 to 7 in Algorithm 1. In this phase, the variables are initialized. A remarkable parameter is the value of the threshold; in this case, a value of 0.72 U is assigned. In Line 7, the master node sends the start order, data of the centroids, and the subset of objects that each slave node will work on.

Classification: Lines 8 to 15 show the instructions carried out in parallel by the slave nodes. Lines 10 and 11 have the instructions for calculating the distance of each object to each of the centroids. Line 12 shows the instruction to assign an object to the group whose centroid is closest to the object. The next line determines the number of objects that changed groups. This information is relevant to determining when the algorithm should stop.

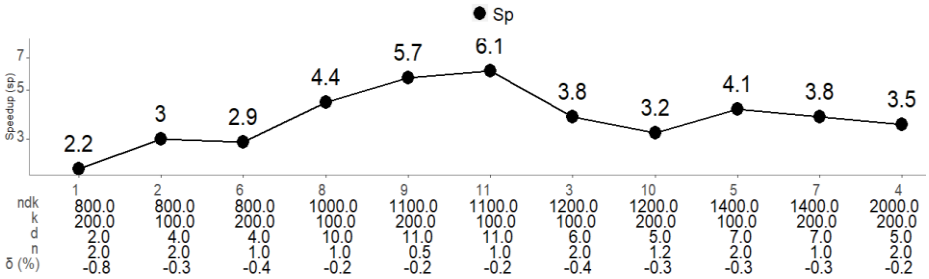


Fig. 2. Speedup obtained when solving synthetic datasets.

In line 14, the slave nodes transmit to the master node a matrix, whose number of rows corresponds to the number of centroids. In the first column, there is the value of the centroid identifier; in the second, the sum of the distance of all the objects that belong to this group; and the third corresponds to the number of objects in the group. Line 15 transmits the number of objects that changed the group.

Calculation de γ: Line 17 of the master node concentrates the data that all the slave nodes have transmitted to it and calculates the percentage of change of objects γ. Note that the value of γ is used in the stopping criterion of the algorithm.

Calculation of the Centroid: In line 20, the calculation of the new centroids is performed.

Convergence: In line 22, it is determined if the object change percentage is less than or equal to the predetermined threshold. If it is affirmative, the algorithm stops. Otherwise, the algorithm continues at step 7.

The HOK-Means programming was coded in Python 3.8 language on a Windows 10 operating system. Some libraries used were: Pandas, Numpy, and parallel Python (PP). The equipment used was a Dell laptop processor Intel® Core™ i7-6700HQ CPU 2.60GHz, 8 processors, 16GB of memory, and 1TB fixed hard drive.

4 Experimental Evaluation

In this section, the metrics to evaluate the parallelization of the algorithm are first described, then the experiments carried out are shown and, finally, the solved data sets are shown.

The speedup (*Sp*) and parallel efficiency (*Ep*) are two metrics used to evaluate the quality of a parallel algorithm. The rate of acceleration or speedup, indicates the relationship between the sequential execution time (*Ts*) and the parallel execution time (*Tp*) Eq. (1).

The ideal speedup is a linear value $Sp = p$, where p is the number of processors [13]. The parallel efficiency indicates the relationship between the speedup and the number of processors used (p) Eq. (2). The ideal parallel efficiency value is 1 [13]:

$$\text{speedup: } Sp = \frac{Ts}{Tp}, \tag{1}$$

$$\text{parallel efficiency: } Ep = \frac{Sp}{p}. \tag{2}$$

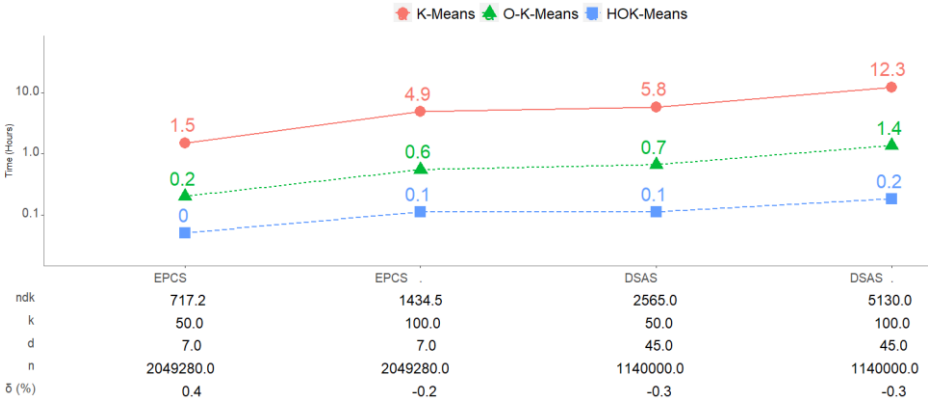


Fig. 3. Solution time with real datasets.

The measure of the quality of the clustering is the objective function value, the sum of the squared error (*SSE*), according to the K-Means algorithm[14]. Eq. (3) shows *SSE*. The clustering quality is better when the value of SSE is lower.

The percentage of quality loss (δ (%)) is the ratio of two quality measures $\delta(\%)=100(1-z/z_o)$, where z is the SSE obtained by solving with standard K-Means and z_o is the SSE obtained by O-K-Means, which is equal to that obtained with HOK-Means.

The *ndk* is an indicator related to the size of the dataset, it is the product of the number of objects in the dataset *n*, the number of attributes *d*, and the number of groups to form *k*:

$$SSE = \sum_{j=1}^{k} \sum_{x \in \mu_j} \left|\left|x - \mu_j\right|\right|^2. \tag{3}$$

4.1 Design of Experiments

To evaluate the performance HOK-Means, 1350 experiments were realized in total. Two real datasets were resolved, with k=50 and 100. 11 synthetic datasets with k=100 and 200 were resolved. All datasets were resolved with the HOK-Means, standard K-Means, and O-K-Means algorithms.

Each configuration of dataset and k was solved 30 times with different initial centroids. Note that the configuration is the same used in [2], but the initial centroids are not the same, so there is a small variation in the quality percentage δ(%).

4.2 Datasets Used in Experiments

In Table 1 describes the set of datasets synthetics and real used. The first row contains the identifier of the dataset; the second shows the number of objects (n) and the third row shows the number of attributes (d).

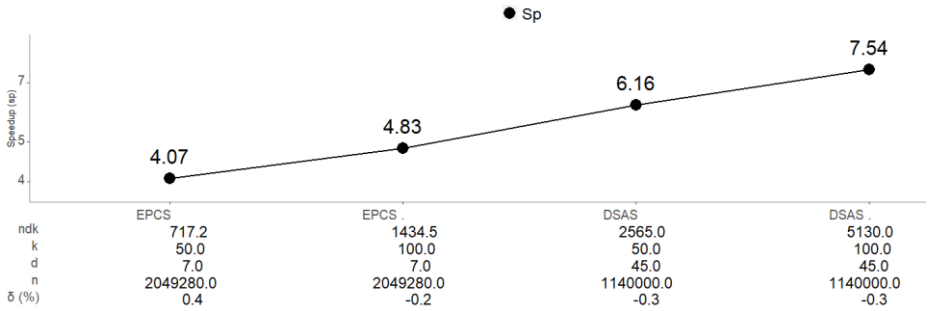


Fig. 4. Speedup obtained when solving real datasets with HOK-Means vs O-K-Means.

In the synthetics datasets (columns 2-12) the value of *n* is represented in millions. In the two last columns the real datasets used are described, which were obtained from the UCI machine learning repository [15].

5 Results

This section shows the results of the solution of the set of datasets using the K-Means, O-K-Means, and HOK-Means algorithms. Fig. 1 and 2 show the results obtained in the solution of synthetic datasets and Fig. 3 and 4 for real datasets.

The *x*-axis represents dataset *id*, which is ordered by *ndk* index from lowest to highest. In the graph in Fig. 1, the processing times are observed, when solving synthetic datasets, obtained with K-Means, O-K-Means, and HOK-Means.

In each case, the processing time with HOK-Means is the least. In the lower part of Fig. 1, it is shown: in the first row, the indicator *ndk* is given in millions; in the second, the percentage of quality loss of HOK-Means with respect to K-Means δ (%); in the third, the number of groups *k*; and in the last one the identifier of the dataset (Table 1).

It is remarkable that the quality of the solution (*SSE*) for the HOK-Means and O-K-Means algorithms is the same, and that the δ (%) was not greater than 0.8% in the experiment with id = 1 with the solution of synthetic datasets. For readers interested in seeing in detail the solution quality of the datasets, refer to [2].

Fig. 2 shows the speedup performance corresponding to each of the solved datasets. In the best case, with dataset 11, the solution time was reduced from 17 hours to less than six minutes, 99.4% of the processing time compared to K-Means, with a quality loss of only 0.21 %, achieving a speedup of 6.1 with respect to O-K-Means. Fig. 3 shows the processing times obtained when solving real datasets, using standard K-Means, O-K-Means and HOK-Means.

The processing time with HOK-Means (orange line) is the shortest. In the lower part of Fig. 3, it is shown: in the first row, the indicator *ndk* is given in millions; in the second, the percentage of quality loss of HOK-Means with respect to K-Means δ (%); in the third, the number of groups k; and in the last one the identifier of the dataset (Table 1). In Fig. 4, the speedup performance is observed, with each real dataset.

The best result was with the DSAS dataset, and k = 100. Processing time with the HOK-Means algorithm was reduced by 98.5% compared to K-Means, with a quality loss of only 0.29%.

A speedup of 7.54 was achieved with respect to O-K-Means. That is, using the same equipment, with HOK-Means the solution time was reduced from 1.36 hours to only 11 minutes, obtaining the same quality of the solution.

6 Conclusions and Future Work

This paper shows that it is feasible to parallelize a highly efficient general-purpose improvement of the K-Means algorithm. To validate the proposal, which we call HOK-Means, a set of experiments composed of real and synthetic datasets was designed. To compare the results of the algorithms, all the datasets were solved using HOK-Means, the standard K-Means, and O-K-Means algorithms. The HOK-Means algorithm is robust even when using larger data sets; computational resources limit it.

Based on the results, it was observed that HOK-Means, in the best of cases, reduce up to 7.54 times the solution time of a real dataset, with the following parameters n =1,140,000; d=45 y k=100. The proposal presented in this article shows a significant improvement in speedup, surpassing the works reported by other researchers.

It is underlined that HOK-Means tends to obtain better processing time as the number of attributes increases. It is noteworthy that due to the data structures implemented and the pagination in the Pandas library of the Python language, it was possible to process large datasets.

HOK-Means is recommended for the solution of large datasets, and in particular with a large number of attributes. To continue this research, we suggest parallelizing other K-Means enhancements.

References

1. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE transactions on emerging topics in computing, vol. 2, no. 3, pp. 267–279 (2014). DOI: 10.1109/TETC.2014.2330519.
2. Pérez–Ortega, J., Almanza-Ortega, N. N., Romero, D.: Balancing effort and benefit of K-means clustering algorithms in big data realms. PLOS ONE, vol. 13, no. 9, pp. 1–19 (2018). DOI: 10.1371/journal.pone.0201874.
3. Ansari, Z., Afzal, A., Sardar, T. H.: Data categorization using hadoop MapReduce-based parallel K-means clustering. Journal of The Institution of Engineers, vol. 100, no. 2, pp. 95–103 (2019)
4. Sardar, T. H., Ansari, Z.: An analysis of distributed document clustering using MapReduce based K-means algorithm. Journal of The Institution of Engineers, vol. 101, no. 6, pp. 641–650 (2020). DOI: 10.1007/s40031-020-00485-2.
5. Wang, Z., Xu, A., Zhang, Z., Wang, C., Liu, A., Hu, X.: The parallelization and optimization of K-means algorithm based on spark. In: 15th International Conference on Computer Science and Education, pp. 457–462 (2020). DOI: 10.11 09/ICCSE49874.2020.9201770.
6. Azimi, R., Sajedi, H., Ghayekhloo, M.: A distributed data clustering algorithm in P2P networks. Applied Soft Computin, vol. 51, pp. 147–167 (2017). DOI: 10.1016/j.asoc.2016.11.045.
7. Al-Ayyoub, M., Yaseen, Q., Shehab, M. A., Jararweh, Y., Albalas, F., Benkhelifa, E.: Exploiting GPUs to accelerate clustering algorithms. In: IEEE/ACS 13th International

Conference of Computer Systems and Applications, pp. 1–6 (2016). DOI: 10.1109/AICCSA.2016.7945796.
8. Lutz, C., Breß, S., Rabl, T., Zeuch, S., Markl, V.: Efficient K-means on GPUs. In: Proceedings of the 14th International Workshop on Data Management on New Hardware, pp. 1–3 (2018). DOI: 10.1145/3211922.3211925.
9. Hadian, A., Shahrivari, S.: High performance parallel K-means clustering for disk-resident datasets on multi-core CPUs. The Journal of Supercomputing, vol. 69, no. 2, pp. 845–863 (2014). DOI: 10.1007/s11227-014-1185-y.
10. Dafir, Z., Lamari, Y., Slaoui, S. C.: A survey on parallel clustering algorithms for big data. Artificial Intelligence Review, vol. 54, no. 4, pp. 2411–2443 (2021). DOI: 10.1007/s10462-020-09918-2.
11. Mackey, P., Lewis, R. R.: Parallel K-means++ for multiple shared-memory architectures. In: 45th International Conference on Parallel Processing, pp. 93–102 (2016). DOI: 10.1109/ICPP.2016.18.
12. Pérez-Ortega, J., Almanza-Ortega, N. N., Vega-Villalobos, A., Pazos-Rangel, R., Zavala-Diaz, J. C., Martínez-Rebollar, A.: The K-means algorithm evolution. Introduction to Data Science and Machine Learning, Chaper 5 (2019)
13. Zavala-Díaz, J. C., Cruz-Chávez, M. A., López-Calderón, J., Hernández-Aguilar, J. A., Luna-Ortíz, M. E.: A multi-branch-and-bound binary parallel algorithm to solve the knapsack problem 0–1 in a multicore cluster. Applied Sciences, vol. 9, no. 24, p. 5368 (2019). DOI: 10.3390/app9245368.
14. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, pp. 281–297 (1967)
15. UCI Machine Learning Repository, archive.ics.uci.edu/ml (2022)

ISSN 1870-4069

Improving Text Representations: A Systematic Literature Review

José Hernández Hernández, Guillermo de Jesús Hoyos Rivera,
Efrén Mezura Montes

Universidad Veracruzana,
Instituto de Investigaciones en Inteligencia Artificial,
Mexico

jclementehdzhdz@gmail.com, {ghoyos,emezura}@uv.mx

Abstract. Natural Language Processing (NLP) is the area charged of designing and developing algorithmic techniques to automatically process text aiming to perform specific tasks related to natural language, its use, and interpretation. These tasks must transform the text into numerical representations so that they can be treated by computational models, such as those of Deep Learning (DL) and Machine Learning (ML). However, we have not found so far a literature review about text representation improvement techniques used in NLP. Most papers are somehow centered on describing Neural Network Language Models (NNLMs). This systematic literature review aims to provide researchers with an overview of the different techniques for improving text representations to identify new areas of opportunity.

Keywords: Text representations, natural language processing, language models, word embeddings.

1 Introduction

NLP comprises the set of algorithmic and mathematical methods that are executed by computers to "understand" human language, also known as natural language [4]. NLP is commonly used to process either, voice or written text, and attempts to solve specific tasks, such as language generation, machine translation, question answering, text classification, sentiment analysis, and part of speech tagging, among others.

One of the main problems of NLP is how to represent text on computers so that they can numerically manipulate it. Thanks to DL, in recent years, researchers looking for new NNLMs which return a set or a list of numerical values, have proposed representations known as embeddings [19]. An embedding is a n-dimensional vector representation containing continuous values representing a word, or a set of words, that are part of a document or a sentence.

Such values contain information about the context where the text is found. This information can be taken from lexical resources, or from making a specific task with a learning approach, such as predicting words or sentences, also known as fine-tuning. Most of the time, this information is said to be distributed along the representation embeddings [20].

This outlook corresponds to the so-called distributional hypothesis [13]. It is important to mention that NNLMs are useful for a set of tasks, and they have shown interesting results, but indeed, they require of high computational power, and the training time is long [16]. This is more important if fine-tuning of a pre-trained model is desired.

Therefore, training could be impractical and could fall into over-fitting. For the above-mentioned, it is necessary to improve embeddings, NNLMs, and pre-trained models. This paper aims to review and summarize novel text representations techniques that emerged after the development of NNLMs, such as Word2Vec (Word to Vector representation), GloVe (Global Vectors for Word Representations), and BERT (Bidirectional Encoder Representations from Transformers).

This review shows different perspectives of the state-of-the-art improving representation techniques and, based on them, try to identify opportunity areas. The rest of the paper is organized as follows: Section 2, presents the related works and background of this systematic literature review. Section 3, briefly describes the used method for this review. Section 4, details, in a narrative way, the results found with a synthesis of the different improving text representations techniques. Finally, Section 5, summarizes the conclusions and future work.

2 Background

This review is a meta-syntheses [22], i. e., a search for new theories, concepts, and key subjects that could provide a view for new approaches, with qualitative information about the analyzed research works. As a part of the background, this section briefly describes some of the main surveys and reviews that operated as motivation to do this systematic literature review.

In [20] a survey is conducted, which includes concepts related to pre-trained models and their embeddings. There, the development of the distributional representations is separated in two generations: the first one includes the Word2Vec and GloVe models, while the second includes recurrent and attention models, such as ELMo (Embeddings from Language Models), BERT, GPT (Generative Pre-trained Transformer), and CoVe (Contextualized Word Vectors).

The survey states that pre-trainig is an advantage that could support language representations, convergence speed of models, and also helps to avoid over-fitting. An empirical survey is presented in [24]. Its goal is to describe and test unsupervised models to represent Twitter text. It includes TF-IDF (Term-Frecuency Inverse-Document-Frecuency), Linear Discriminant Analysis (LDA), Word2Vec, GloVe, BERT, XLNet (Extra Long Network), and ELMo, among others.

The authors evaluate the generated representations by using clustering and found that, for example, BERT, which is improved and has many learning parameters, is not necessarily the best one. Other simpler methods such as TF-IDF could be used instead. Another survey, presented in [**?**], describes different strategies to represent text from the symbolic point of view, to the appearance of the distributed representations learning, such as Word2Vec. This survey could serve as an introduction to the text representation techniques in the DL era.

In [1], a survey on word embeddings is presented. The authors describe distributed representations based on vector space models, statistical language modeling, prediction, and count-based models. Models such as TF-IDF, Word2Vec, GloVe, and statistical methods such as Latent Semantic Analysis (LSA) are included. Finally, the idea of improving the results of NLP tasks by tuning models is presented.

Based on [14], Neural Networks (NNs) that generate embeddings or text representations are treated as language models. In this case, the authors describe models such as Word2Vec and classic recurrent models. A very important finding of this survey is that attention models such as BERT are considered better than other text representations.

Finally, in [2], a survey of NNLMs is introduced, where fifty different models, which include shallow, recurrent, convolutional, and attention models, and their variants, are described. The authors of this survey highlight the computational complexity of NNLM, and they propose to generate new strategies by adding common sense and human intuition to improve text representations.

From this related work review, it can be clearly seen that research is mostly interested in describing NNLMs, and highlighting some aspects of the improvements made. This is why this paper focuses on describing other improvement approaches that can be useful to add to future NNLMs implementations. To the best of the authors' knowledge, there is no systematic literature review about the concepts which are presented in this work.

3 Research Method

The research method implemented in this systematic literature review is that expressed in [22], following the steps explained in the following subsections.

Scoping. Aims at responding the next questions: (1) Which are the main NNLM, embeddings, and text representations? (2) Which are the techniques used to improve the representations?

Searching. A sequence of search terms was conducted to identify relevant work, including: (1) text representations, (2) symbolic text representations, (3) numerical text representations, and (4) improvement of text representations, all of them for NLP. Considering this work as a first step to analyze the state-of-the-art in this topic, the search terms were handled through the Google Scholar engine, and the included main databases were IEEExplore, Springer, ACM, Elsevier, and arXiv, with no restriction to finding works in other sources.

Screening. In this stage, a manual screening of the papers was performed, with special emphasis on the abstract and title of the papers.

Eligibility. An in-depth reading was performed to determine the papers eligibility for inclusion. The following information was extracted: (1) main topics, (2) original text representation or NNLM, (3) improving technique, (4), data type used for experiments (word, sentence, paragraph or document), and (5) publication year or last submission year.

Study quality. Finally, the following is the checklist (based on [15]) that was applied to the papers quality assessment:

Table 1. Paper scores of study quality questions.

Paper	Q1	Q2	Q3	Q4	Q5	Q6	Total score	Year
[6]	1	0	0	1	1	0	3	2014
[3]	1	0	1	1	1	1	5	2014
[25]	1	0	1	1	1	1	5	2014
[5]	1	1	1	1	1	0	5	2015
[26]	1	1	0	1	1	1	5	2016
[18]	1	1	0	1	1	1	5	2017
[17]	1	1	0	1	1	0	4	2018
[9]	1	1	0	1	1	1	5	2018
[21]	1	1	1	1	0	0	4	2019
[10]	1	1	0	0	0	1	3	2019
[11]	1	1	1	1	1	1	6	2019
[23]	1	1	0	1	1	0	4	2020
[12]	1	1	1	1	0	1	5	2020
[7]	1	1	1	1	0	0	4	2021

– Q1: Are the aims clearly stated?
– Q2: Is there a comparison with other methods?
– Q3: Are the used data clearly explained?
– Q4: Is it clear what is the technique used to improve the text representation?
– Q5: Are negative findings present?
– Q6: Is it clear what are the future trends in such an improving technique?

The defined checklist has six questions that can be answered with *yes* (1) or *no* (0). Such a checklist is motivated by the research questions and the findings that can generate new areas of opportunity. Table 1, shows the details of the score obtained, and the corresponding publication year or last submission for each paper. It can be clearly seen that the oldest papers were published in 2014, which coincides with the emergence of NNLMs. Selected papers describe improving implementations, which differ from fine-tuning.

4 Narrative of the Results

In this section, the results of the search performed are presented. Also, implicit answers to the research questions are given by comparing and briefly summarizing the information found. In this way, 14 primary studies are considered and described in this section. The information about these papers is focused on improving text representations, NNLMs output vectors, or word embeddings.

We excluded papers that describe text representations without improving them, but it is essential to mention that NNLMs are a clear advance, and give different views of how to process text, using contextual and non-contextual models such as Word2Vec, GloVe, and BERT. From now on, NNLMs embeddings were used to tackle a considerable number of NLP tasks.

Moreover, in various cases those embeddings were improved using different techniques, such as term weighting, retrofitting, and adding sememes into Word Representation Learning (WRL), or simply adding knowledge information into representation vectors.

These tasks are usually known as fine-tuning, but in this review, we take the concept of fine-tuning related to the re-training of an NNLM to solve a specific task, as well as BERT does. The main difference between fine-tuning and improving representations, is that the latter is part of a post-training, i. e., considering a vector representation from an NNLM, and how an extra method can be included to enrich the contained information.

Another view of the improving methods takes place when external information and knowledge, as lexical resources, are involved in model training. In the following paragraphs, improving techniques are briefly described.

Term weighting. Using the original representation vector from the TF-IDF model, and their own data, the authors of [11, 12] enriched the information by applying an algorithm that modified the TF-IDF, and combined original vectors with a weighting algorithm, respectively.

In [11], an algorithm transforms the original TF-IDF embedding into a matrix of weights. Original TF-IDF designates a weight for each term, and in contrast, it is proposed an algorithm that assigns different weights to a single term, considering the classes in which it appears. On the other hand, the authors in [12] used an optimization algorithm to add compactness and expressiveness to vectors at the sentence level.

Taking as a basis Word2Vec, the algorithm adds information about the frequency of terms and it includes a classifier that determines the weights of each sentence. In both cases, the resulting representation of text was employed for a classification task.

Retrofitting. This technique is a way to update resulting vectors from NNLMs by adding semantic and lexical information. The retrofitting approach is firstly described in [5], where authors use lexical resources such as WordNet, FrameNet, and Paraphrase Database (PPDB) to enrich GloVe and Word2Vec embeddings via label propagation. Such lexical resources can be taken as symbolic or knowledge representations of words and sentences.

In [9], an explicit retrofitting approach is described, which incorporates three elements: (1) word embeddings from an NNLM, (2) lexical resources, (3) a learning model, where the first and second element fit the model. The resulting vector contains the mixed information from former both elements. Such a learning model can be a NN that has the optimization task to minimize the similarity distance while maintaining the original embeddings aspects.Considering the first retrofitting paper, in [21] is described a retrofitting process over vectors from ELMo.

Whereas in [26], with medical terms as the language model, applied retrofitting with the purpose of improving the semantic similarity. Finally, in [7], using knowledge graphs, such as those graphs from the Trans (Translation-Based Model) family, and the retrofitting technique on a re-implemented BERT, the authors made biomedical information extraction.

Sememes. Sememes are included in the WRL using an extension of Word2Vec [18]. Here the authors define a sememe as the minimum semantic unit of word meanings. The technique is based on the extension of Word2Vec and the aggregation of the attention mechanism typical in Transformers.

Instead of words as the required context in the original Word2Vec, authors use the HowNet sememes and senses, then the Word2Vec model is trained from scratch using the original formalization.

Statistical methods. These methods are applied to produce embeddings and an improvement over original representation vectors. In [6], the Canonical Correlation Analysis is used at adding information to LSA word representations, by employing semantic and syntactic relations from other languages, such as French, English and Spanish.

A Clustering-based improving technique is implemented in [3], where the algorithm is applied over words that are represented with semantic spaces such as Hyperspace Analogue to Language (HAL), and Correlated Occurrence Analogue to Lexical Semantic (COALS), among other methods which had not been tested in the representation of words, such as LSA.

Knowledge incorporation. With respect to this technique, in [17] the authors incorporate knowledge information at the training of a Word2Vec model. The resulting vectors were evaluated in two steps: (1) predicting the relatedness of sentence pairs, and (2) sentiment classification, also known as sentiment analysis. Another paper about this technique is presented in [25]. In this case, a technique based on Word2Vec and the aggregation of relations information from WordNet and PPDB is jointly trained.

The first training stage encompasses only the representation of words that appear in a data set of text, i. e., tweets, and the latter stage, includes only the relation information to train a model based on Word2Vec. Both stages are mixed to produce better representations. It is relevant to note that the explicit retrofitting [9] and sememes aggregation [18], can be incorporated in this subsection.

Iteresting improvements. This last subsection is focused on describing two techniques, different from the above, used to improve representations. In [10], a combinatorial Genetic Algorithm is used to select vector representations from two different NNLMs, Word2Vec and GloVe, and the chosen one is assigned to only one word.

Finally, in [23], with the objective to replace the position variable (commonly used in BERT), the authors use a generalization of word embeddings through continuous functions. As can be seen, most of the improving techniques use Word2Vec or GloVe as the main NNLMs. On the other hand, WordNet is the knowledge lexical resource that is used to enrich word representations.

Some of the reviewed papers are not clear about what is their improving technique future trend, while the rest propose to continue scaling and improving their technique. These improving efforts could be applied to transformer-based models such as BERT, or even be part of few-shot or zero-shot learning to enrich the input to the models used. Knowledge or symbolic information could be added as a part of a specified task fine-tuning.

5 Conclusions and Future Work

This systematic literature review briefly summarized with a narrative and meta-syntheses way, improving techniques over NNLMs, embeddings, or text representations. Different techniques were found, such as term weighting, retrofitting, knowledge incorporation such as sememes, and two interesting approaches that could provide improvements to NNLMs.

The analyzed research works had the expected level of relevancy, and they follow the main motivation of this review: known improving techniques in text representations. As part of future work, and emphasizing that this study is the first one of its kind, we need to search in-depth new concepts such as WRL, sememes, zero-shot and few-shot learning, because it is possible to find other improving techniques of representations in general, as they can be useful for text representations.

Acknowledgments. The first author acknowledges CONACyT's support to pursue graduate studies.

References

1. Almeida, F., Xexeo, G.: Word Embeddings: A Survey (2019). DOI: 10.48550/ARXIV.1901.09069.
2. Babic, K., Martinčić-Ipšić S., Meštrović , A.: Survey of Neural Text Representation Models. ´ Information, vol. 11, no. 11, pp. 511 (2020). DOI: 10.3390/info11110511.
3. Brychcín, T., Konopík, M.: Semantic Spaces for Improving Language Modeling. Computer Speech and Language, vol. 28, no. 1, pp. 192–209 (2014). DOI: 10.1016/j.csl.2013.05.001.
4. Eisenstein, J.: Introduction Natural Language Processing. The Massachusetts Institute of Technology (2019)
5. Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., Smith, N. A.: Retrofitting Word Vectors to Semantic Lexicons. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1606–1615 (2015). DOI: 10.3115/v1/n15-1184.
6. Faruqui, M., Dyer, C.: Improving Vector Space Word Representations using Multilingual Correlation. In: 14th Conference of the European Chapter of the Association for Computational Linguistics, pp. 462–471 (2014). DOI: 10.3115/v1/e14-1049.
7. Fei, H., Ren, Y., Zhang, Y., Ji, D., Liang, X.: Enriching Contextualized language Model from Knowledge Graph for Biomedical Information Extraction. Briefings in Bioinformatics, vol. 22, no. 3, pp. 1–14 (2021). DOI: 10.1093/bib/bbaa110.
8. Ferrone, L., Zanzotto, F. M.: Symbolic, Distributed, and Distributional Representations for Natural Language Processing in the Era of Deep Learning: A Survey. Frontiers in Robotics and Artificial Inteligence, vol. 6 (2020). DOI: 10.3389/frobt.2019.00153.
9. Glavaš, G., Vulić, I.: Explicit Retrofitting of Distributional Word Vectors. In: 56th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 34–45 (2018). DOI: 10.18653/v1/p18-1004.
10. Gunasegaran, T., Cheah, Y. N.: Evolutionary Combinatorial Optimization for Word Embedding in Sentiment Classification. Malaysian Journal of Computer Science, pp. 34–45 (2019). DOI: 10.22452/mjcs.sp2019no3.3.

11. Guo, B., Zhang, C., Liu, J., Ma, X.: Improving Text Classification with Weighted Word Embeddings Via a Multi-channel TextCNN model. Neurocomputing, vol. 363, pp. 366–374 (2019). DOI: 10.1016/j.neucom.2019.07.052.
12. Gupta, S., Kanchinadam, T., Conathan, D., Fung, G.: Task-Optimized Word Embeddings for Text Classification Representations. Frontiers in Applied Mathematics and Statistics, vol. 5, pp. 1–10 (2020). DOI: 10.3389/fams.2019.00067.
13. Harris, Z. S.: Distributional Structure. WORD, vol. 10, no. 2–3, pp. 146–162 (1954). DOI: 10.1080/00437956.1954.11659520.
14. Jing, K., Xu, J.: A Survey on Neural Network Language Models (2019). DOI: 10.48550/ARXIV. 1906.03591.
15. Kitchenham, B., Charters, S.: Guidelines for Performing Systematic Literature Reviews in Software Engineering (2007)
16. Lasse F. W. A., Kanding, B., Selvan, R.: Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models (2020). DOI: 10.48550/ARXIV.2007.03051.
17. Li, Y., Wei, B., Liu, Y., Yao, L., Chen, H., Yu, J., Zhu, W.: Incorporating Knowledge into Neural Network for Text Representation. Expert Systems with Applications, vol. 96, pp. 103–114 (2018). DOI: 10.1016/j.eswa.2017.11.037.
18. Niu, Y., Xie, R., Liu, Z., Sun, M.: Improved Word Representation Learning with Sememes. In: 55th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 2049–2058 (2017). DOI: 10.18653/v1/P17-1187.
19. Pilehvar, M. T., Collados, J. C.: Embeddings in Natural Language Processing, Springer International Publishing (2021). DOI: 10.1007/978-3-031-02177-0.
20. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained Models for Natural Language Processing: A Survey. Science China Technological Sciences, vol. 63, no. 10, pp. 1872–1897 (2020). DOI: 10.1007/s11431-020-1647-3.
21. Shi, W., Chen, M., Zhou, P., Chang, K. W.: Retrofitting Contextualized Word Embeddings with Paraphrases. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 1198–1203 (2019). DOI: 10.18653/v1/D19-1113.
22. Siddaway, A. P., Wood, A. M., Hedges, L. V.: How to Do a Systematic Review: A Best Practice Guide for Conducting and Reporting Narrative Reviews, Meta-analyses, and Meta-syntheses. Annual Review of Psychology, vol. 70, pp. 747–770 (2019). DOI: 10.1146/annurev-psych-010418-102803.
23. Wang, B., Zhao, D., Lioma, C., Li, Q., Zhang, P., Simonsen, J. G.: Encoding Word Order in Complex Embeddings (2019). DOI: 10.48550/arXiv.1912.12333.
24. Wang, L., Gao, C., Wei, J., Ma, W., Liu, R., Vosoughi, S.: An Empirical Survey of Unsupervised Text Representation Methods on Twitter Data (2020). DOI: 10.48550/ARXIV.2012. 03468.
25. Yu, M., Dredze, M.: Improving Lexical Embeddings with Semantic Knowledge. In: 52nd Annual Meeting of the Association for Computational Linguistics, vol. 2, pp. 545–550 (2014). DOI: 10.3115/v1/p14-2089.
26. Yu, Z., Cohen, T., Bernstam, E. V., Johnson, T. R., Wallace, B. C.: Retrofitting Word Vectors of MeSH Terms to Improve Semantic Similarity Measures. In: 7th International Workshop on Health Text Mining and Information Analysis, pp. 43–51 (2016). DOI: 10.18653/v1/w16-6106.

ISSN 1870-4069

Induction of Convolutional Decision Trees with Differential Evolution for Image Segmentation

Jesús Arnulfo Barradas Palmeros, Efrén Mezura Montes,
Héctor Gabriel Acosta Mesa, Aldo Márquez Grajales,
Rafael Rivera López

Universidad de Veracruz,
Instituto de Investigaciones en Inteligencia Artificial,
Departamento de Sistemas y Computación,
Mexico

{jesus.arnulfo.bap, li.aldoma}@gmail.com, {emezura,
heacosta}@uv.mx, rrivera@itver.edu.mx

Abstract. Convolutional Neural Networks are the dominant approach for solving the image segmentation problem. However, they demand significant amounts of manually labeled data for training and suffer from lacking explainability. As an alternative, Convolutional Decision Trees take advantage of the interpretability and simplicity of decision tree models. Nevertheless, choosing between equivalent trees is a challenging task, given the trade-off between the model's precision and complexity. In this work, we propose using Differential Evolution as a global search metaheuristic for the induction of Convolutional Decision Trees applied to the image segmentation problem. Various tests were conducted on the Weizmann Horse dataset, where the elevated computational cost of determining the individuals' fitness value limited the search. Nonetheless, short and explainable models were induced with promising results for some parts of the dataset. In this way, Differential Evolution appears as an attractive tool for Convolutional Decision Trees induction, expecting future improvements.

Keywords: Convolution, decision trees, differential evolution, image segmentation.

1 Introduction

The image segmentation problem consists of assigning a semantic label to the pixels in an image. Differentiating an object from the image's background is essential in most image analysis systems. Thus, various image segmentation methods have been proposed in the literature [7]. However, in recent years, the high performance of Convolutional Neural Networks (CNN) as a Deep Learning technique has been proclaimed the dominant paradigm in computer vision [12].

Therefore, various approaches using CNN have been studied for the image segmentation problem, as shown in [3]. Despite the high-performance results reached by CNN in different tasks, they suffer from requiring lots of labeled data for training.

Additionally, CNN faces the difficulty of high training time, making some applications unsuitable for their use or demanding a specific hardware infrastructure [8]. Consequently, some challenges remain for CNN, such as their explainability, the efficient use of memory, and the speed to process a new instance on a real-time application [12].

Decision Trees (DT) are a classification model characterized by their simplicity and interpretability where internal nodes represent the test conditions and leaf nodes are the class labels. Nonetheless, the DT induction classic process uses a greedy recursive partitioning heuristic that suffers from adaptability in some applications. Alternatively, various approaches to using metaheuristics for decision tree induction have been proposed in the literature [15].

Convolutional Decision Trees (CDT) were proposed in [8] for image segmentation and feature learning problems, performing well and using a fraction of the time needed for training a CNN without a particular hardware configuration. This approach was tested on the Weizmann Horse dataset [2], obtaining an F1-score of 80.4% with a tree depth of 18. However, results showed that trees with short depths have less adequate performance.

Three main metaheuristic-guided DT induction strategies are described in [11]. The first consists of a recursive partition strategy where the metaheuristic finds a near-optimal partition. The second strategy uses the metaheuristic as a global search technique that looks for the complete model of a near-optimal DT.

An essential challenge in this approach is maintaining diversity in the population. Besides, the computational cost of the fitness value calculation increases considerably with high-dimension datasets. Finally, the third strategy uses a previously induced DT and continually optimizes it according to the metaheuristic.

A population-based metaheuristic used in literature for the induction of near-optimal DT is the Differential Evolution (DE) algorithm [15]. DE is one of the most popular metaheuristic search strategies and has been applied successfully for solving several optimization problems. Furthermore, DE is prominent in the algorithm simplicity where few parameters control the search process [6].

Two ways of using DE in DT induction are shown in [10, 15]. Perceptron Decision Trees incorporate a linear combination test condition on each internal node. DE is used to evolve the values of the tree's structure [10]. In [15], the DE-ADT$_{SPV}$ method uses DE as a global search strategy to find near-optimal parallel-axis DT coding the trees as real-value vectors.

Both works achieved decent results in the classification accuracy obtained by their resulting models. The use of DE in two different strategies for the induction of oblique decision trees is studied in [16]. The first is OC1-DE, where a recursive partition strategy is used with DE to find near-optimal partitions for each tree node. The second method is DE-ODT, which uses a global search strategy to find a near-optimal oblique decision tree.

The solutions representation corresponds to a real-value vector that codes the internal values of a tree. The length of the vector depends on the number of attributes given and the predefined depth of the tree. Both methods described showed their effectiveness in decision tree induction.

Based on the literature, it is seen that DE has been used in various DT induction processes. Nevertheless, to the best of our knowledge, DE has not been applied for the particular CDT induction case. Therefore, this paper uses DE as a global search strategy to induce CDT. Consequently, DT and DE characteristics were employed to construct an explainable model for the image segmentation problem.

The remaining structure of this document is divided into four sections. Section 2 includes the DE algorithm description. Implementation details are defined in section 3, whereas section 4 explains the experimentation and the results obtained during tests. Finally, section 5 contains the conclusions and suggested future work.

2 Differential Evolution (DE)

Differential Evolution (DE) is a population-based evolutionary algorithm for optimization of complex problems [13]. DE mainly works with real-valued vectors representing potential solutions to the problem. However, DE is also applied in the discrete and combinatory domain [14].

The basic strategy of DE is called DE/rand/1/bin [16, 1, 5]. The general DE procedure generates a trial vector for every individual x_i or target vector in the population. The first step consist of generating a noise vector using Equation 1 where r_0, r_1 and r_2 are individuals randomly selected from the population and F is a user-defined scale factor:

$$v_i = r_0 + F(r_1 - r_2). \tag{1}$$

Once v_i is computed, the trial vector is generated stochastically. Equation 2 expresses this process. If a random number ($rand_j$) is lower than a Crossing Rate (CR) defined by the user or the position (j) corresponds to one previously determined by chance, the component takes the value from v_i. Otherwise, it takes the value from x_i:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (\text{rand}_j \leq CR) \; or \; (j = J_{rand}); j = 1, ..., |x_i|, \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{2}$$

Finally, the next step is determining which vector will be part of the next generation population. A binary tournament between trial and target takes place where the one with the better fitness value is chosen [16]. This selection method works as an elitism mechanism by always keeping the best individual in the population.

Another DE variant is called DE/best/1/bin, where the main difference is in the computation of v_i. Instead of choosing a random individual as r_0, the individual in the population with the highest fitness value is chosen [6].

3 Proposal Implementation

This work proposes an analysis of the DE effectiveness for CDT induction. The procedure implemented is based on the DE-ODT algorithm for oblique decision tree induction proposed in [16], where the values of each node are represented in a vector that is evolved.

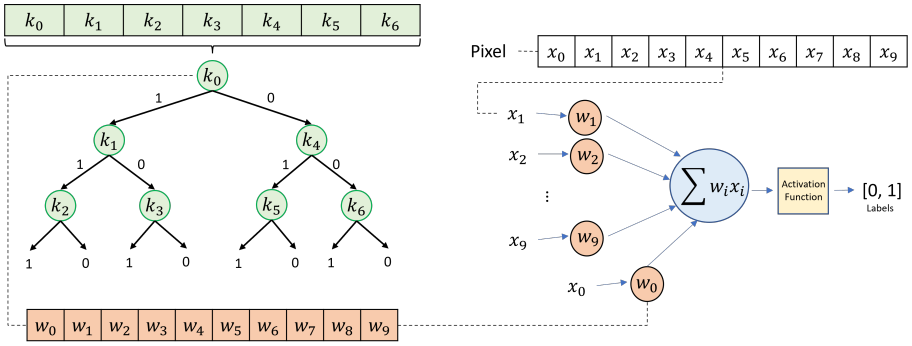


Fig. 1. Codification of a Convolutional Decision Tree internal convolution kernels and how a pixel-associated instance is processed.

In this proposal, the values coded in the vector represent internal convolution kernels for the tree. To determine which tree branch to take while classifying a dataset instance, we propose using a perceptron-like structure similar to [10]. The product between the instance values and the weights of a convolution kernel passes through an activation function returning a 0 or 1 label.

Based on that label, the tree node where the instance needs to go is decided. This procedure is repeated until a leaf node is reached, then a label is assigned to the instance. Figure 1 illustrates the proposal. DE/rand/1/bin and DE/best/1/bin are the two DE variants used in this work. Both versions need user-defined parameters such as scale factor F, crossing rate CR, population size, and how many generations will run the algorithm. Moreover, two additional parameters of the tree structure are required for this application: kernel size and tree depth.

3.1 Images Preprocessing

The images used for the tree induction procedure are preprocessed to obtain a vector associated with each pixel. Then, according to [10], pixels are coded as a vector of values given by the neighbor pixels. The vector length depends on the kernel size defined by the user. Additionally, a value of 1 is included in every instance to operate with the bias value of the proposed perceptron-like structure.

3.2 Coding Potential Solutions

A population's individual represents a solution consisting of a real-valued vector with the values of convolution kernels associated with the tree's internal nodes. The amount of weights needed by each kernel depends on kernel size ($s \times s$). Therefore, the weight amount is computed as s^2+1 where s is the kernel side length, and the one corresponds to the bias value.

Another aspect to be considered when a solution is coded is the number of kernels needed in the tree. This value depends entirely on tree depth and is determined by 2^d-1, where d is the depth defined by the user. Figure 2 shows an example of this codification.

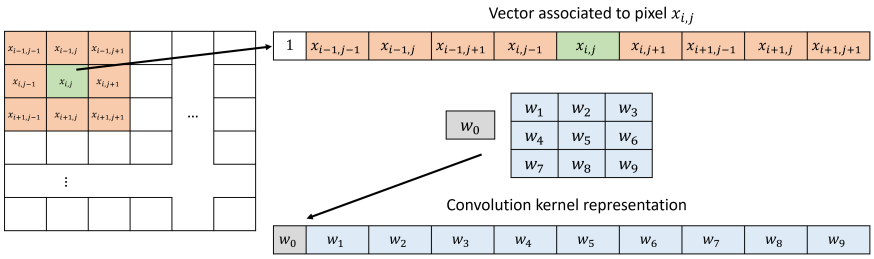


Fig. 2. Coding of pixel-associated instances and convolution kernels.

3.3 Fitness Value

The fitness value of each individual is determined by the F1-score metric based on the precision and recall metrics. To calculate this value, we need to compute the labels assigned to the training instances by the tree coded in each individual of the population and compare them with the actual labels of the instances. The resulting fitness value is between 0 and 1, and we try to maximize it using DE.

The initial population is generated with fixed length vectors of randomly chosen values from a uniform distribution with limits -255 and 255. After that, individuals are evaluated to compute their fitness value, and the DE procedure starts. Additionally, the Mean Point Distance metric [**?**] is used in every generation to measure diversity in the population.

3.4 Repair Operator

A repair operator is used to restrain the search space and keep the values of the tree between -255 and 255. While computing the v_i vector, if a value exceeds one of the limits imposed, a new value is calculated as two times the exceeded limit (-255 or 255) minus the value that infringed the restriction.

4 Experiments and Results

A single user-defined image was used to create short training and test sets as a controlled algorithm initial test. Then, a first algorithm parameters calibration was done, obtaining favorable results in pattern detection. These results identified that higher values of population size and the number of generations resulted in more suitable individuals at the end of the search. Nonetheless, both have a direct impact on the procedure's computational cost. After the initial tests, we executed 13 extended tests using the Weizmann Horse Dataset [2], which consists of 328 manually segmented horses images, allowing us to compare our results with the ones described in [8].

An image resizing procedure was applied to reduce the number of pixel-associated instances processed in the tree induction process. Moreover, multiple tests were conducted varying some algorithm parameters. In addition, population size and generation number values were adjusted to make each test last less than 24 hours. Finally, CR and F parameters were maintained at 0.9 in all executions.

Table 1. Tests of the proposed method for Convolutional Decision Trees induction.

Test	DE-var	Training	Popsize	Generations	Depth	F1-score	Accuracy	Time(hrs)
1	Rand	2/3	40	59	3	0.4296	0.6277	22.31
2	Rand	2/3	40	59	3	0.4421	0.6770	21.21
3	Rand	2/3	40	59	3	0.4671	0.5925	21.73
4	Best	2/3	40	60	3	0.4480	0.6546	18.65
5	Best	2/3	40	60	3	0.4465	0.6473	16.75
6	Best	2/3	40	60	3	0.4730	0.6877	18.32
7	Best	1/3	46	100	3	0.4513	0.6549	21.92
8	Best	1/10	80	200	3	0.4882	0.6798	23.17
9	Best	2/10	60	120	3	0.4819	0.6846	19.76
10	Best	2/10	50	100	5	0.4531	0.6025	16.72
11	Best	1/10	60	110	7	0.4696	0.6827	21.07
12	Best	1/20	50	130	13	0.4367	0.6504	16.32
13	Best	1/25	40	100	17	0.4255	0.5881	22.22

In the first six tests, the fraction of training data was maintained with similar population size, generation number, and a tree depth value of three. Also, the two DE versions previously mentioned in the document were employed. The main obstacle found was the time required for each test. Hence, the selected values of population size and the number of generations were limited.

After that, we tried different configurations of training set size to decrease the time consumed by the induction process, allowing us to increase parameters like the tree depth. Without reducing the training set, the resources and time demanded would have impeded testing deeper tree models. Table 1 shows our proposal's results under the above-mentioned considerations.

This table shows that DE/best/1/bin got better results and was faster than DE/rand/1/bin when tested in similar conditions. As a consequence, this DE variant was used during the remaining tests. Training set reduction did not significantly decrease the method's performance showing the model's capacity for generalization even though the induction process takes place with small amounts of data.

Test 8 resulting model got the highest F1-score while using only 10% of the data for training. More complex models did not imply better results in our tests, but the induction used even more reduced fractions of the dataset. Figure 3(a) presents the tree induced in test 9, whereas Figure 3(b) shows a comparison of the actual mask and the predicted mask of 12 test images from the same test.

We noticed that the model struggles with background and foreground textures, while horses' contour is detected in most images. In none of the test cases we achieved similar results to the 80.4% of F1-score obtained in [8] where their method for CDT induction takes 12 hours for training. Nevertheless, our results are comparable for short-depth trees.

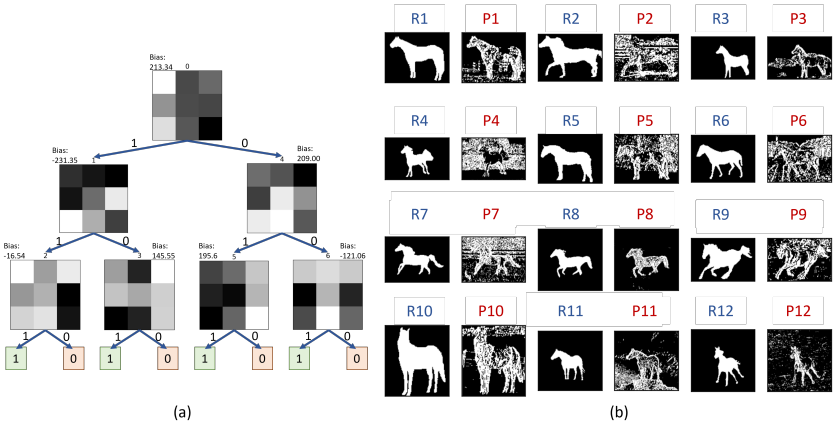


Fig. 3. (a) Tree induced in test 9. (b) A sample of the segmentation results obtained by the tree showed in (a). The letter R is used to identify the real segmented masks, and the letter P is for the predicted ones.

5 Conclusions and Future Work

In this work, a method for CDT induction with DE is proposed and compared with the original method proposed in [8]. The main difference is using a population-based metaheuristic to induce several trees instead of only one with a recursive partition strategy. Applying DE for the CDT induction faces the computational time problem when evaluating the capacity to classify the training instances by the population's individuals.

This classification ability is the fitness function of DE. Moreover, this task increases resource demand as the number of training instances is augmented. When an image dataset is used, the amount of pixel-associated instances is quite considerable, in the order of millions, making the labor of the model induction more complex.

In conclusion, we suffer from the search limits imposed by the computational cost required by the induction process in our proposal. This situation forced us to use less adequate parameters for the DE algorithm and reduce the training data fraction. However, DE was still capable of inducing short and explainable models. Given this, one thing to highlight is the method's capability to train with little data without additional processes to augment the training set.

For the model's explainability, one could successively apply convolutional operations to an image following the tree structure and analyze the results for each branch and leaf node. Nevertheless, deeper tree models reduce explainability, given the elevated model's number of branches and kernels.

In order to make the proposed procedure proficient, it is necessary to overcome the challenge of the computational cost of evaluating an individual. Without accomplishing this, the capabilities of using DE as a global search tool would still be limited for this type of problem. Future work could include trying a self-adapted DE scheme and exploring different parameter values for kernel size and tree depth.

Additionally, some improvements could be implementing techniques like windowing [9] to reduce the number of training instances and methods like pruning to enhance the resulting trees. For future reference, it is necessary to compare the proposal performance with diverse approaches for image segmentation, such as U-Net and other Convolutional Neural Networks methods [3].

Acknowledgments. The first author is funded by a Consejo Nacional de Ciencia y Tecnología (CONACYT) of Mexico.

References

1. Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A.: Differential Evolution: A Review of More Than Two Decades of Research. Engineering Applications of Artificial Intelligence, vol. 90 (2020). DOI: 10.1016/j.engappai.2020.103479.
2. Borenstein, E., Sharon, E., Ullman, S.: Combining Top-down and Bottom-up Segmentation. In: Conference on Computer Vision and Pattern Recognition Workshop, pp. 46–46 (2004). DOI: 10.1109/CVPR.2004.314.
3. Cao, F., Bao, Q.: A Survey on Image Semantic Segmentation Methods with Convolutional Neural nNetwork. In: International Conference on Communications, Information System and Computer Engineering, pp. 458–462 (2020). DOI: 10.1109/CISCE50729.2020.00103.
4. Contreras-Varela, L.: Un estudio sobre diversidad en optimizacion evolutiva con restricciones, Universidad Veracruzana, Xalapa (2018)
5. Dabbagh, R. D. A., Neri, F., Idris, N., Baba, M. S.: Algorithmic Design Issues in Adaptive Differential Evolution Schemes: Review and Taxonomy. Swarm and Evolutionary Computation, vol. 43, pp. 284–311 (2018). DOI: 10.1016/j.swevo.2018.03.008.
6. Faiz-Ahmad, M. Mat-Isa, N. A., Hong-Lim W., Meng-Ang, K.: Differential Evolution: A Recent Review Based on State-of-the-art Works. Alexandria Engineering Journal, vol. 61, no. 5, pp. 3831–3872 (2022). DOI: 10.1016/j.aej.2021.09.013.
7. Hadjiiski, L., Samala, R., Chan, H. P.: Chapter 88 - Image Processing Analytics: Enhancements and segmentation. In: Molecular imaging, pp. 1727–1745 (2021). DOI: 10.1016/B978-0-12-816386-3.00057-0.
8. Laptev, D., Buhmann, J. M.: Convolutional Decision Trees for Feature Learning and Segmentation. In: Pattern Recognition: 36th German Conference, pp. 95–106 (2014). DOI: 10.1007/978-3-319-11752-2 8.
9. Limon, X., Guerra-Hernández, A., Cruz-Ramírez, N., Acosta-Mesa, H. G., Grimaldo, F.: A Windowing Strategy for Distributed Data Mining Optimized Through GPUs. Pattern Recognition Letters, vol. 93, pp. 23–30 (2017). DOI: 10.1016/j.patrec.2016.11.006.
10. Lopes, R. A., Freitas, A., Silva, R. P., Guimaraes, F. G.: Differential Evolution and Perceptron ˜ Decision Trees for Classification Tasks. In: Proceddings of 13th International Conference on Intelligent Data Engineering and Automated Leraning, pp. 550–557 (2012). DOI: 10.1007/ 978-3-642-32639-4 67.
11. Lopez, R. R., Reich, J. C., Montes, E. M., Chavez, M. A. C.: Induction of Decision Trees as Classification Models Through Metaheuristics. Swarm and Evolutionary Computation, vol. 69 (2022). DOI: 10.1016/j.swevo.2021.101006.
12. Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., Terzopoulos, D.: Image Segmentation Using Deep Learning: A Survey. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, pp. 3523–3542 (2022). DOI: 10.1109/TPAMI.2021.3059968.

13. Opara, K. R., Arabas, J.: Differential Evolution: A Survey of Theoretical Analyses. Swarm and Evolutionary Computation, vol. 44, pp. 546–558 (2019). DOI: 10.1016/j.swevo.2018.06.010.
14. Price, K. V.: Differential evolution. In: Zelinka, I., Snásel, V., Abraham, A. (eds) Handbook ˘ of Optimization. Intelligent Systems Reference Library, vol. 38, pp. 187–214 (2013). DOI: 10.1007/978-3-642-30504-7 8.
15. Rivera-Lopez, R., Canul-Reich, J.: Construction of Near-Optimal Axis-parallel Decision Trees Using a Differential-evolution-based Approach. IEEE Access, vol. 6, pp. 5548–5563 (2018). DOI: 10.1109/ACCESS.2017.2788700..
16. Rivera-Lopez, R., Canul-Reich, J.: Differential Evolution Algorithm in the Construction of Interpretable Classification Models. Artificial intelligence-emerging trends and applications Chapter 3, pp. 49–73 (2018). DOI: 10.5772/intechopen.75694.

ISSN 1870-4069

Proposal of a CNN-Based Approach for Traffic Signal Detection

Madaín Pérez Patricio[1], Carlos Alexis Ramírez Mendoza[1], Germán Rios Toledo[1], Juan Antonio de Jesús Osuna Coutiño[2]

[1] Tecnológico Nacional de México, Campus Tuxtla Gutiérrez, Mexico

[2] Instituto Nacional de Astrofísica, Optica y Electrónica, Mexico

{M14270620, madain.pp, german.rt}@tuxtla.tecnm.mx, osuna@inaoep.mx

Abstract. Traffic Signal Detection (TSD) is an important module in autonomous vehicles and Driver Assistance Systems (DAS). Although there are several approaches to TSD, in most cases, these are based on only the 2D localization, i.e., these systems do not provide their information to other drivers or future travel routes. On the other hand, some works only focus on a specific signal (traffic light) causing a bias into the signal set. To address these problems, an alternative is to use deep networks, image metadata, and Information Technology (IT). Motivated by the latter, we propose a methodology for traffic sign detection and geolocation using a CNN-based approach. This strategy combines the abstraction power of deep learning with IT and metadata information. For that, our methodology has three steps. First, traffic sign detection provides the location and classification of the road signs. Second, we use the image metadata to obtain the geolocation. Third, the information technology step presents the geospatial and classification information into an Application Programming Interface (API). Also, we evaluate this methodology in public images and a proposed dataset with metadata information. The quantitative experiments were conformed from the signal detection in two urbanized environments (open imagesV6 and proposed dataset). For that, we analyzed two labels of road signs (Traffic light, Traffic sign). Also, our road sign detection had an average recall of 0.89, i.e., considering the ground-truth, we recognized 89%.

Keywords: Convolutional neural network, image processing, trafic signal.

1 Introduction

The population in cities is increasing continuously, and they require more efficient services. The concept of smart cities try to deal with these requirements incorporating technologies as internet of things, artificial intelligence, cloud computing, among others. These technologies are combined to providing citizens the best place to live.

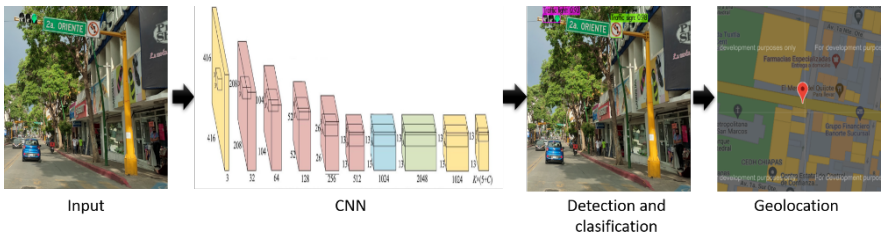


Fig. 1. Methodology.

Transport, health, and living are the most common research subjects in the literature related to smart cities [24]. For the transport area and autonomous vehicles, the main idea is the efficient traffic management to reduce time lost, fuel consumption, and risks of injury.

This problem has been treated from different perspectives. For example, vehicles have been equipped with Advanced Driver-Assistance Systems (ADAS) that includes powerful computer vision systems used to detect pedestrian, other vehicles and traffic signs. On the other hand, smart traffic control systems establish rules of control traffic to help vehicles as ambulances to reduce the lost time [11].

One of the most important objects of the urban infrastructure is the traffic signs. They are designated to regulate the traffic and to fulfill requirements of safety and comfort to drivers. So, an efficient method is required for automatic detection and positioning of the traffic signs. This subject has been extensively studied in the last few years. Autonomous driving and intelligent transportation systems also require the precise identification of traffic signs.

Automatic traffic signs detection and positioning is a challenging task because there are several problems such as: variable light conditions, non-standard form and size signals, and weather changes. An approach to deal with these problems proposes the use of smart traffic signs. They can send wireless messages to vehicles placed in the neighborhood of the sign.

This approach requires efficient protocols of communication between smart traffic signs and vehicles. Another approach uses computer vision based systems [27]. The algorithms and technologies used in computer vision based systems are efficient.

Three stages are generally involved in traffic sign detection: detection, tracking and recognition [16]. Nevertheless, the accuracy achieved in the traffic sign detection is 95.71%. For several applications, this error can be acceptable, but only one traffic sign non-detected can produce injury in the drivers [26].

In this paper, we propose a method suited for embedded processing. The system has two parts: a computer vision based system to traffic sign detection and a cloud based system to traffic sign positioning. The traffic sign detected is recorded in a database and can be updated over the time.

This information would be transmitted to drivers to reduce risks. More efficient services are required as the population in cities is increasing. With the use of traffic signs in the adequate positioning and number, the traffic can be reduced in the cities while the trash collection requires efficient management of trucks, workers, and truck rides.

Fig. 2. Scheme of metadata extraction.

The city planners must have a detailed and complete description of the urban infrastructure to take well decisions. The information that city planners can take into count must include the kind and precise position of available urban infrastructure.

Several efforts have been realized to standardize traffic signs around the world. Nevertheless, each country and each city can define their particular traffic sign. Therefore, there is no universal method for traffic sign identification. Fig. 2 shows an example of geolocation using metadata extraction.

The proposed method uses a smartphone which includes a camera and a sensor of positioning. A dataset has been created and used to train a neural network. Images acquired by the smartphone are processed in a personal computer which yields a map where the traffic signs are identified and positioned. This method is suited for embedded systems.

2 Related Work

Variable light conditions, non-standard forms of the signs, changes of size of the observed signs because of distance from the sensor, partial occlusions, and weather changes make the traffic sign detection a challenging task. A huge quantity of algorithms has been proposed to avoid these problems. To test algorithms proposed by the community, several datasets have been created [9, 12, 21, 22]. As traffic signs are different by each country and each city, a particular dataset would be required for each city that uses this technology.

Traffic signs recognition can be performed using different methods. Methods as proposed in [26], are based on color or shape recognition. These methods take advantage of fact that the color or shape of traffic signs are highly visible and contrast the surrounding neighborhood [25]. This kind of proposal has a time of processing reduced but present weakness in presence of light changes, rotation, and viewing angle.

[19] try to deal with occluded and attached traffic signs by estimating the shape of the arc described by the contour of the traffic sign. In [4], the color information and object properties are used to identify regions of interest that reduces the processing times of a support vector machine based classification algorithm. Other methods are learning based, like the proposed by [15, 17]. They are based on the use of convolutional neural networks, where they use a huge number of images to train it.

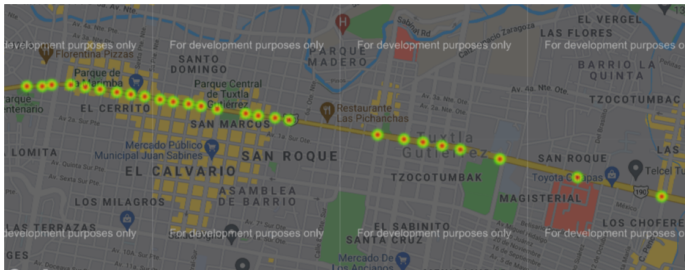


Fig. 3. Geolocation results.

For example, in [7], 40,000 images has been used to train the convolutional network. These methods are robust to light changes and changes of perspective but requires a high computational power. The use of models like the You Only Look Once (YOLO) yields results with a high throughput and real-time processing achieving 95.15% of precision [1]. To achieve traffic sign identification as long as possible, two cameras has been used in [6].

A wide-angle camera select traffic sign candidates while a narrow-angle camera is used to get a high-resolution image of the traffic sign candidate. A hardware implementation using Field Programmable Gate Arrays devices can yield results in real-time [20]. A preprocessing step has been applied to the image and traffic signs using a neural network. The use of simplified Gabor wavelets and convolutional neural networks can also achieve real-time performance [3, 18].

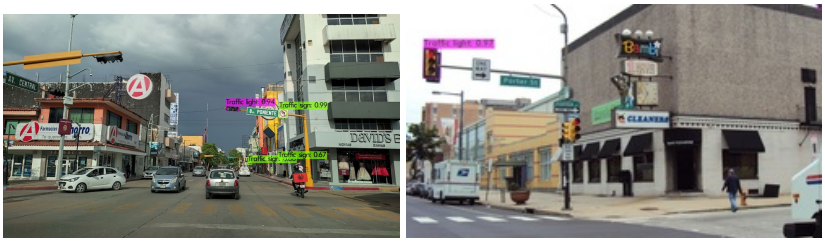
Reported results yields that the average processing time is 5.4 ms. Incorporating a Global Positioning System (GPS) device in the setup can yield interesting information for other kind of applications. In [13], road attributes has been inferred using traffic sign identification and GPS information. On the other hand, geo-tagged Google Street View images and a Gopro camera [27] has been used to traffic sign recognition a positioning.

GPS, inertial sensor, camera, and laser sensor are used in a van [10]. These results show that precision of detection is highly correlated to weather conditions. To reduce issues associated to traffic signs, wireless traffic signs has been developed [23]. They transmit information to road users that can be reproduced with auditive signals even if the traffic sign is not observed by the driver. Although there are several approaches to TSD, in most cases, these are based on only the 2D localization, i.e., these systems do not provide their information to other drivers or future travel routes.

On the other hand, some works only focus on a specific signal (traffic light) causing a bias into the signal set. In our case, unlike the related work we propose a methodology for traffic sign detection and geolocation using a CNN-based approach. This strategy combines the abstraction power of deep learning with IT and metadata information.

3 Proposed Method

This section presents the proposed methodology for traffic signal detection and geolocation. Our strategy combines the abstraction power of deep learning with Information Technology (IT) and metadata extraction.



(a) (b)

Fig. 4. The images show traffic signal detection using our methodology in the proposed dataset (a), and the open images V6 dataset (b).

For that, our methodology has three steps.First, traffic signal detection provides the location and classification of the road signs. Second, we use the image metadata to obtain the geolocation. Finally, the information technology step presents the geospatial and classification information into an Application Programming Interface (API). The schematic representation of the proposal is shown in Fig. 1.

3.1 Traffic Signal Detection

In this work, we use the Yolov4 architecture[2] for traffic signal detection. Although some works only focus on specific signal detection (traffic light). This approach causes a bias into detection, i.e., a variety of signals are omitted by the system. For that, our network learns two labels on urbanized environments (traffic light υ^1 and signal υ^2).

Training set. In the training set of traffic signal detection, we use urban images where the camera looks head-on to buildings. This set is formed of two datasets that provide different outdoor scenes (open images V6 [8], and a proposed dataset which can be accessed using the next link Dataset). In the open images V6 repository, we use 3000 images labeled with traffic lights and traffic signs.

On the other hand, our dataset has 2000 images labeled of the main avenues in Tuxtla city; between 13 street northwestern and Pencil street in Tuxtla, Chiapas, Mexico. For that, we use a Xiaomi Redmi Note 10 with a resolution of 4000×3000 pixels. We apply data augmentation in the training step. For that, we transformed image set via mirroring.

On the other hand, we obtained the image number for training and test using the Pareto principle or 80/20 rule [**?**], (80% training, and 20% test). Finally, we use a Google colaboratory environment with 26 Gb of Ram, 150Gb storage and GPU Tesla P100-PCIE 16Gb.

CNN. The input of the CNN is an RGB image Φ. In this case, we train the YOLOv4 network to learn two labels on urbanized environments (traffic light υ^1 and signal υ^2). Also, our network uses a bounding box ϑ_i to delimit the elements in the image Φ. For that, we use two colors in the bounding box ϑ_i. A purple bounding box ϑ_1 delimits a traffic light υ^1 On the other hand, a green bounding box ϑ_2 delimits a traffic signal υ^2, Fig. 4 shows examples of our detection. The training time of the CNN took roughly 10 hours.

Table 1. Routes of the proposed dataset.

Route	Start	Finish
1	Ave. panamericana Km. 1080	Blvd. Andres Sierra Rojas
2	Blvd. Andres Sierra Rojas	Rd. Juan Crispin
3	north fifth St	Libramiento norte
4	Libramiento norte	Blvd. Presa chicoasen

Table 2. Signal detection evaluation.

Images number	Precision	Recall	F-score
1000	0.76	0.69	0.72
4000	0.87	0.86	0.86
6000	0.86	0.88	0.87
7000	0.88	0.89	0.88

3.2 Metadata Extraction

Metadata is defined as the information provided about one or more aspects of the data. In the case of the image metadata, this has a variety of information about the description of the picture, such as its origin, size, camera, GPS, aperture, shutter speed, among others.

In our methodology, we use the Global Positioning System (GPS) data of the images with signal detection, i.e., we use the metadata to assign geolocation of the signals (traffic light υ^1 and signal υ^2).

3.3 Information Technology

A geolocation system is an information technology solution that determines the location of an object in a physical or virtual environment. In our case, we use the Maps JavaScript [14] to show the geolocation in an API system. For that, we send to the API the GPS data of the images with signal detection (traffic light υ^1 and signal υ^2). This approach allows us to save and display the coordinates of traffic signals detected using the methodology.

Also, we can provide this information to other drivers or future travel routes. Fig. 3 shows an example of the traffic signal set detected in the JavaScript map. These traffic signals are located in the central avenue, between 13 street northwestern and Pencil street in Tuxtla city, Chiapas, Mexico.

4 Discussion and Results

In this section, we present the experiments of signal detection. For that, the problem was addressed as a classification problem (*Traffic light*, *Traffic sign*). On the other hand, we evaluate two datasets that provided different outdoor scenes (open images V6 [8] and proposed dataset). In our proposed dataset, we selected four routes of the main avenues in Tuxtla city, Chiapas, Mexico. Table 1 shows the different routes.

Table 3. Signal detection evaluation in the proposed dataset.

Clase	Precision	Recall	F-score
Traffic light	0.94	0.97	0.95
Traffic sign	0.82	0.81	0.81

The quantitative evaluation compares our detection with the ground truth of the signals detected. We used three measures ($recall$, $precision$ and $F-score$) **Eq. 1-3** based on the number of true positives (Tp), true negatives (Tn), false positives (Fp), and false negatives (Fn). The true positives Tp count the number of signals whose label was predicted correctly w.r.t. the ground truth.

To count the number of true negatives Tn, we proceed as follows: suppose that we are interested in the traffic light, then all images with another classification than traffic light according to the ground truth, should have received any other predicted classification except traffic light; if that is the case, each of these detections is counted as true negatives.

The false positives Fp correspond to all those detections whose label is incorrect. Finally, false negatives Fn correspond to those image sections that should have received a specific label, but the prediction did not assign it correspondingly:

$$\text{recall} = \frac{Tp}{Tp + Fn}, \tag{1}$$

$$\text{precision} = \frac{Tp}{Tp + Fp}, \tag{2}$$

$$F - \text{score} = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2\frac{\text{recall} * \text{precision}}{\text{recall} + \text{precision}}. \tag{3}$$

In our first experiment, we analyzed the performance of our network in signals and traffic light detection. For that, we evaluate this experiment in our dataset. Table 3 shows the result of our approach for signal detection. The evaluation with traffic light showed a better performance since this signal has a shape standardized.

In our second experiment, we implement different training in our signal detection network. In the proposed dataset, we use 1000, 2000, 4000 images. In the open images V6 repository [8], we use 3000 images of the second to fourth training. On the other hand, we only use the proposed dataset in the first training. Also, in each case, we carry out 10,000 epochs. Table 2 shows the result of our approach for signal detection. The evaluation with 7,000 images showed the highest detection.

5 Conclusions

In this work, we have introduced a new methodology for traffic signal detection and geolocation using a CNN-based approach. Our strategy was to combine the abstraction power of deep learning with IT and metadata information.

For that, our methodology has three steps. First, traffic signal detection provides the location and classification of the road signs. Second, we use the image metadata to obtain the geolocation.

Third, the information technology step presents the geospatial and classification information into an API. The quantitative experiments were conformed from the signal detection in two urbanized environments (open images V6 [8] and proposed dataset). For that, we analyzed two labels of road signs (*Traffic light*, *Traffic sign*). Also, our road sign detection had an average $recall$ of 0.89, i.e., considering the ground-truth, we recognized 89%.

On the other hand, the road sign detection had an average $precision$ of 0.88, i.e., considering the classification, we classify 88.0% correctly. We should note that the proposed methodology involves a combination of techniques based on CNN with IT and metadata information. This methodology explores the abstraction power of deep learning and the information that provides geolocation technologies. In our opinion, our approach brings the best of the two worlds to address the difficult problem of traffic signal geolocation.

On the other hand, the results obtained in this research have demonstrated the feasibility of traffic sign detection and geolocation. Based on these results, we propose as future work to implement the proposed methodology with an informatic system to provide vehicle routes. In our opinion, this can increase the accuracy of route prediction time.

References

1. Arief, R. W., Nurtanio, I., Samman, F. A.: Traffic Signs Detection and Recognition System using the YOLOv4 Algorithm. In: International Conference on Artificial Intelligence and Mechatronics Systems, pp. 1–6 (2021). DOI: 10.1109/AIMS52415.2021.9466006.
2. Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M.: YOLOv4: Optimal speed and accuracy of object detection (2020). DOI: 10.48550/arXiv.2004.10934.
3. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning Imbalanced Datasets with Label-distribution-aware Margin Loss. In: 33rd Conference on Neural Information Processing Systems NeurIPS, pp. 1–12 (2019). DOI: 10.48550/arXiv.1906.07413.
4. Coţovanu , D., Zet, C., Foşalău, C., Skoczylas, M.: Detection of Traffic Signs Based on Support Vector Machine Classification using Hog Features. In: International Conference and Exposition on Electrical and Power Engineering, pp. 518–522 (2018). DOI: 10.1109/ICEPE.2018.8559784.
5. Craft, R. C., Leake, C.: The Pareto Principle in Organizational Decision Making. Management Decision, pp. 729–733 (2002). DOI: 10.1108/00251740210437699.
6. Gu, Y., Yendo, T., Tehrani, M. P., Fujii, T., Tanimoto, M.: A New Vision System for Traffic Sign Recognition. In: IEEE Intelligent Vehicles Symposium, pp. 7–12 (2010). DOI: 10.1109/IVS.2010.5548005.
7. Slam, M. T.: Traffic sign detection and recognition based on convolutional neural networks. In: International Conference on Advances in Computing, Communication and Control, pp. 1–6 (2019). DOI: 10.1109/ICAC347590.2019.9036784.
8. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The Open Images Dataset v4: Unified image classification, object detection, and visual relationship detection at scale. International Journal of Computer Vision, (2020). DOI: 10.1007/s11263-020-01316-z.

9. Lau, M. M., Lim, K. H., Gopalai, A. A.: Malaysia Traffic Sign Recognition with Convolutional Neural Network. In: IEEE International Conference on Digital Signal Processing, pp. 1006–1010 (2015). DOI: 10.1109/ICDSP.2015.7252029.
10. Lee, J. S., Yun, D. G.: The road traffic sign recognition and automatic positioning for road facility management. International Journal of Highway Engineering, vol. 15, no. 1, pp. 155–161 (2013). DOI: 10.7855/IJHE.2013.15.1.155.
11. Lee, W. H., Chiu, C. Y.: Design and Implementation of a Smart Traffic Signal Control System for Smart City Applications. Sensors, vol. 20, no. 2, pp. 508 (2020). DOI: 10.3390/s20020508.
12. Mathias, M., Timofte, R., Benenson, R., Van Gool, L.: Traffic Sign Recognition—how Far are We From the Solution? In: International Join Conference on Neural Networks, pp. 1–8 (2013). DOI: 10.1109/IJCNN.2013.6707049.
13. Mèneroux, Y., Le-Guilcher, A., Saint-Pierre, G., Hamed, M. G., Mustière, S., Orfila, O.: Traffic Signal Detection from In-vehicle GPS Speed Profiles using Functional Data Analysis and Machine Learning. International Journal of Data Science and Analytics, vol. 10, no. 1, pp. 101–119 (2020). DOI: 10.1007/s41060-019-00197-x.
14. Googel Platform: Maps javascript api www.maps-backend.googleapis.com. (2021)
15. Radu, M. D., Costea, I. M., Stan, V. A.: Automatic Traffic Sign Recognition Artificial Inteligence-deep Learning Algorithm. In: 12th International Conference on Electronics, Computers and Artificial Intelligence, pp. 1–4 (2020). DOI: 10.1109/ECAI50035.2020.9223186.
16. Ruta, A., Li, Y., Liu, X.: Real-time Traffic Sign Recognition from Video by Class-specific Discriminative Features. Pattern Recognition, vol. 43, no. 1, pp. 416–430 (2010). DOI: 10.1016/j.patcog.2009.05.018.
17. Sermanet, P., LeCun, Y.: Traffic Sign Recognition with Multi-scale Convolutional Networks. In: International Joint Conference on Neural Networks, pp. 2809–2813 (2011). DOI: 10.1109/ IJCNN.2011.6033589.
18. Shao, S., McAleer, S., Yan, R., Baldi, P.: Highly Accurate Machine Fault Diagnosis using Deep Transfer Learning. IEEE Transactions on Industrial Informatics, vol. 15, no. 4, pp. 2446–2455 (2018). DOI: 10.1109/TII.2018.2864759.
19. Soendoro, D., Supriana, I.: Traffic Sign Recognition with Color-based Method, Shape-arc Estimation and SVM. In: International Conference on Electrical Engineering and Informatics, pp. 1–6 (2011). DOI: 10.1109/ICEEI.2011.6021584.
20. Souani, C., Faiedh, H., Besbes, K.: Efficient Algorithm for Automatic Road Sign Recognition and its Hardware Implementation. Real-time Image Processing, vol. 9, no. 1, pp. 79–93 (2014). DOI: 10.1007/s11554-013-0348-z.
21. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German Traffic Sign Recognition Benchmark: a Multi-class Classification Competition. In: International Join Conference on Neural Networks, pp. 1453–1460 (2011). DOI: 10.1109/IJCNN.2011.6033395.
22. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. Neural Networks, vol. 32, pp. 323–332 (2012). DOI: 10.1016/j.neunet.2012.02.016.
23. Toh, C. K., Cano, J. C., Fernandez-Laguia, C., Manzoni, P., Calafate, C. T.: Wireless digital traffic signs of the future. The Institution of Engineering and Technology Networks, vol. 8, no. 1, pp. 74–78 (2019). DOI: 10.1049/iet-net.2018.5127.
24. Toh, C. K., Sanguesa, J. A., Cano, J. C., Martinez, F. J.: Advances in Smart Roads for Future Smart Cities. In: Proceedings of The Royal Society A, vol. 476 (2020). DOI: 10.1098/rspa.2019.0439.
25. Wali, S. B., Abdullah, M. A., Hannan, M. A., Hussain, A., Samad, S. A., Ker, P. J., Mansor, M. B.: Vision-Based Traffic Sign Detection and Recognition Systems: Current Trends and Challenges. Sensors, vol. 19, no. 9, pp. 2093 (2019). DOI: 10.3390/s19092093.

26. Wali, S. B., Hannan, M. A., Hussain, A., Samad, S. A.: An Automatic Traffic Sign Detection and Recognition System Based on Colour Segmentation, shape matching, and SVM. Mathematical Problems in Engineering (2015). DOI: 10.1155/2015/250461.
27. Wu, Z.: Computer Vision-Based Traffic Sign Detection and Extraction: A Hybrid Approach using GIS and Machine Learning. Electronic Theses and Dissertations (2019)

ISSN 1870-4069

Selection of a Fixed-Length Set of Biologically-Constrained Association Rules for Bacterial Vaginosis Diagnosis

María Concepción Salvador-González[1], Juana Canul-Reich[1], Rafael Rivera-López[2], Efrén Mezura-Montes[3], Erick de la Cruz-Hernandez[4]

[1] Universidad Juárez Autónoma de Tabasco,
División Académica de Ciencias y Tecnologías de la Información,
Mexico

[2] Instituto Tecnológico de Veracruz,
División en Ciencias de la Salud,
Mexico

[3] Universidad Veracruzana,
Instituto de Investigaciones en Inteligencia Artificial,
Mexico

[4] Universidad Juárez Autónoma de Tabasco,
División Académica Multidisciplinaria de Comalcalco,
Mexico

mcsalvadorg@gmail.com, {juana.canul, erick.delacruz}@ujat.mx,
rrivera@itver.edu.mx, emezura@uv.mx

Abstract. This paper describes a Differential-Evolution-based approach for selecting a reduced subset of association rules previously generated by the Apriori algorithm. The selected rules are those with biological significance for the diagnosis of Bacterial Vaginosis. We use integer-based vectors as population individuals of the evolutionary algorithm and a combination of various rule metrics to define the fitness function. The experimental results indicate that the DE/best/1/bin variant performs better than the DE/rand/1/bin variant and that the approach reaches the expected results.

Keywords: Differential evolution, association rules, bacterial vaginosis.

1 Introduction

Bacterial Vaginosis is the most common of the vaginal diseases in women of reproductive age. It is associated with several severe health conditions such as preterm delivery, post-abortion infection, pelvic inflammatory disease, and sexually transmitted diseases [10]. As in other fields of knowledge, machine learning techniques have been used to detect this condition [2].

On the other hand, Association Rule Mining is an important topic in data mining used to identify the relationships strongly associated among itemsets in a dataset [15]. In

Table 1. Antecedent itemset values used in the experimental study.

Dataset feature	Values	Description
Cristpatus	1	crispatusA
Gasseri	2	crispatusB
	1	gasseriA
Iners	2	gasseriB
	1	inersA
Jensenii	2	inersB
	1	jenseniA
Megasphaera	2	jenseniB
	1	megasphaeraP
Atopobium	2	megasphaeraN
	1	atopobiumP
Gardnerella	2	atopobiumN
	1	gardnerellaP
	2	gardnerellaN

ID of rule 1	ID of rule 2	ID of rule 3	ID of rule 4	...	ID of rule N
34	125	200	6		29

Fig. 1. Encoding scheme to select N association rules.

the related specialized literature, we found that several computational techniques, such as Simulated Annealing [6], Genetic Programming [9], Differential Evolution [14], and Genetic Algorithms [8], have been applied to generate and optimize Association Rules for a wide range of real applications. In particular, the Differential Evolution algorithm has proven its effectiveness in optimizing machine learning models.

To the best of our knowledge, no study has been found in the existing literature that applies Association Rule Mining and Differential Evolution to select biologically meaningful rules for the diagnosis of bacterial vaginosis infection. This work addresses the adaptation of the Differential Evolution algorithm to determine association rules using biological constraints in cases of Bacterial Vaginosis Positive (BV+).

2 Materials and Methods

For this study, a dataset with 17 features with medical information of 201 sexually active women aged 18 to 50 who underwent their annual gynecological inspection routine at the Laboratory of Research in Metabolic and Infectious Diseases, Universidad Juarez Autonoma de Tabasco is used [12]. According to our interest, we considered the records with a positive result for bacterial vaginosis only.

After this selection, 51 records remained, with the variables representing the Crispatus, Gasseri, Inners, and Jensenii lactobacillus, and the Megasphaera, Atopobium, and Gardnerella bacteria. An association rule has the form $X \rightarrow Y$, where X is the rule's antecedent, and Y is its consequent [1].

The metrics most commonly used for the validation of the obtained rules are the following [7]:

Table 2. Parameters values.

Parameter	Value	Parameter	Value
F (Scale factor)	0.9	CR (Crossover rate)	0.5
NP (Population size)	20	MAX_GEN (Number of generations)	30

– **Support:** The frequency count of a rule.

– **Confidence:** The probability that the elements in the consequent are in the antecedent.

– **Coverage:** The frequency with which the rule antecedent appears.

– **Lift:** It compares the expected frequency of a rule with the expected frequency at random.

– **Confidence-boost:** The relationship between the confidence of rules that have the same consequent but different elements in the antecedent.

As part of association rule mining, the dataset is processed for the Apriori algorithm, one of the most widely used algorithms for pattern discovery using frequent itemsets to generate association rules [5]. A disadvantage of the Apriori algorithm is the combinatorial exploitation of the rules produced, so applying techniques to obtain a reduced set of high-quality rules is essential. Differential Evolution (DE) is an efficient evolutionary algorithm for solving optimization problems in continuous spaces [13].

DE encodes candidate solutions through real-valued vectors and applies a difference vector to disrupt a population of these solutions. First, a population of candidate solutions is randomly created, then applying the DE evolutionary process that builds a new population using mutation, crossover, and selection operators at each iteration.

Instead of implementing traditional crossover and mutation operators, DE applies a linear combination of several candidate solutions selected randomly to produce a new solution. Finally, DE returns the best candidate solution in the current population when the stop condition is fulfilled. An advantage of DE is that it uses a few control parameters: a crossover rate Cr, a mutation scale factor F, and a population size NP.

Since the information in the dataset is not numerical, the DE algorithm must be adapted to generate optimized results. We encode the values with integer-valued vectors, implying that the algorithm's operators must be modified to create only feasible solutions. Another critical element is the definition of the objective function, which must correctly guide the evolutionary process. In the present work, metrics used with association rules should be considered, as well as those defining the biological significance levels for the problem under study.

3 Experimental Study

The experimental study includes three stages. First, the meaning of the values that each attribute can take on are defined as indicated in Table 1. In total, there are 51 records in the dataset.

Table 3. Results of 30 independent runs for each DE variant.

Test	rand/1/bin	best/1/bin	Test	rand/1/bin	best/1/bin
1	36.8267	37.6078	16	37.1960	36.8431
2	37.9019	37.6666	17	37.7450	38.0588
3	37.1372	37.4919	18	37.4509	38.1372
4	36.9411	37.5294	19	37.1372	36.8235
5	37.2549	37.5686	20	36.9215	36.5098
6	36.5882	37.6666	21	37.3333	36.7058
7	36.7254	37.0980	22	37.0588	36.6862
8	37.5686	37.1568	23	38.0588	36.6862
9	36.6470	**38.3333**	24	**38.8039**	36.9019
10	37.5490	37.1764	25	36.8627	37.0000
11	36.5098	37.8431	26	37.7647	37.2941
12	36.6666	37.7450	27	38.1372	37.1372
13	36.8039	37.5098	28	37.4117	36.6470
14	37.3921	37.3921	29	37.0588	37.0000
15	36.8039	37.3921	30	38.2549	36.9019

Next, the Apriori algorithm[5] is applied and 332 association rules are generated, all for cases of BV+. Each rule ends up with one or more features in the antecedent part, and the value of BV+ is set as the consequent since these are the cases of interest in this work. Finally, the DE algorithm is used to find a reduced set of association rules, based on their biological significance.

3.1 Implementation of The Differential Evolution (DE) Algorithm

Three elements are first defined to implement the DE algorithm: the individuals' encoding scheme, the fitness function, and the variation operators.

1. **Encoding scheme:** An individual of the population is a subset of N association rules each identified with an ID number. Fig. 1 shows an example of this codification. In this work, the value of N is set to 6 since in [4] authors obtained five rules with biological significance which were determined by a human expert, so $N = 6$ rules ensures the algorithm will find this minimal set of rules.
2. **Fitness function:** Each i-th individual in the population is evaluated to define the fitness value. In this work, the fitness function $f(x_i)$ is the sum of the M metrics of the association rules encoded on the individual as follows:

$$f(x_i) = \sum_{j=1}^{N} \sum_{k=1}^{M} m_{j,k}, \tag{1}$$

where N is the number of desired association rules, M is the number of metrics involved to define the solution quality, and $m_{j,k}$ is the k-th metric computed for the j-th rule.

[5] In this work, the arules R package is used to create the association rules (cran.r-project.org/web/packages/arules/index.html).

Table 4. Statistical values of the experimental study.

Statistical measure	**rand/1/bin**	**best/1/bin**
Best value	38.8039	38.3333
Mean	37.2849	37.2836
Median	37.1666	37.2352
Standard deviation	0.5575	0.4773
Worst value	36.5098	36.5098
Best test number	24	9
Median test number	16	26

Seven metrics are used in the fitness function: support, confidence, coverage, lift, confidence boost, frequency of positive bacteria in the rules, and the occurrences of high values of lactobacillus iners. The first five metrics are previously described in Section 2.

The other two metrics are used to determine the presence of some bacteria, and lactobacillus [4]. These metrics are included to define the biological significance of the association rules in this sense the higher results of the addition of the metrics have higher significance.

3. **Variation operators:** Differential mutation and crossover operators are defined to create feasible offsprings.
   – **Mutation:** Three randomly chosen individuals of the current population (x^{r_1}, x^{r_2} and x^{r_3}), being different from each other and also different from the target vector, are linearly combined to yield a *mutated vector* v^i, using a user-specified scale factor F to control the differential variation, as follows:

$$v^i = \lfloor x^{r_1} + \mathrm{F}\big(x^{r_2} - x^{r_3}\big) \rceil. \tag{2}$$

   Eq. 2 is related with the DE/rand/1 variant defined in [11]. Other commonly used variant is known as DE/best/1, where the best individual in the population $\mathbf{x}^{best}$ is combined with two random chosen individuals of the current population, as follows:

$$v^i = \lfloor x^{best} + \mathrm{F}\big(x^{r_1} - x^{r_2}\big) \rceil. \tag{3}$$

   – **Crossover:** The mutated vector is recombined with the target vector to build the trial vector u^i. For each $j \in \big\{1, \ldots, |x^i|\big\}$, either x^i_j or v^i_j is selected based on a comparison between a uniformly distributed random number $r \in [0, 1]$ and the crossover rate CR. The recombination operator also uses a randomly chosen index $l \in \{1, \ldots, |x^i|\}$ to ensure that u^i gets at least one value from v^i, as follows:

$$u^i_j = \begin{cases} v^i_j & \text{if } r \leq \text{CR or } j = l, \\ x^i_j & \text{otherwise.} \end{cases} \tag{4}$$

In the Eqs. 2 and 3, $\lfloor w \rceil$ symbol denotes that the w value is rounded to the nearest integer since the encoding scheme defined for this work indicates that the parameter values are only integers. If a parameter value of a mutated vector is outside its range, it is replaced with a random value between 1 and 332.

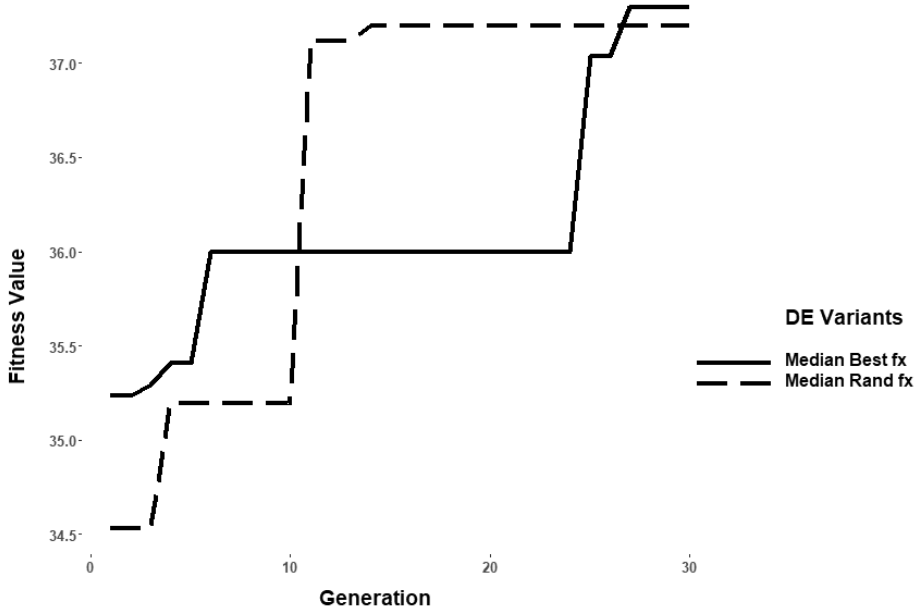


Fig. 2. Convergence plot for the median values of the two DE variants.

3.2 Algorithm Parameters

It's well known that the performance of the DE algorithm is affected by the values of its parameters: F, CR, and NP [3]. The parameter values used in this work are based on those commonly used in the existing literature [11]. Since this experimental study is a work in progress, no parameter tuning process has been carried out.

4 Results

Table 3 shows the results of 30 independent runs with the two DE variants included in this study (rand/1/bin and best/1/bin). The best fitness value for the rand/1/bin version is 38.8039 on test number 24 and for the best/1/bin version is 38.3333 on test 9. The best fitness values are highlighted in bold, and the best median value of each variant is underlined.

The statistics comparison for each variant is shown in Table 4, and Fig. 2 depicts the convergence plot of the run reaching the median value of the two variants. When comparing the results of the two variants using the Wilcoxon statistical test, the calculated p-value is 0.4065, indicating that the two variants have the same behavior.

Table 5 shows the rules encoded by the best individuals of each variant. According to the statistical results, the best value is obtained with the rand/1/bin variant. However, the results obtained in the independent runs and the behavior of the convergence graph show that the best/1/bin variant had better performance in selecting the association rules.

Table 5. Reduced set of association rules selected by the two DE variants.

ID	Association rule
Variant: best/1/bin	
306	{atopobiumP, crispatusA, gardnerellaP, gasseriA, jenseniA} → {VB+}
139	{atopobiumP, gardnerellaP, inersA, megasphaeraP} → {VB+}
224	{atopobiumP, crispatusA, gardnerellaP, megasphaeraP} → {VB+}
328	{atopobiumP, crispatusA, gardnerellaP, gasseriA, inersA, jenseniA} → {VB+}
268	{atopobiumP, crispatusA, gardnerellaP, inersA, megasphaeraP} → {VB+}
210	{atopobiumP, inersA, jenseniA, megasphaeraP} → {VB+}
Variant: rand/1/bin	
212	{atopobiumP, gasseriA, inersA, megasphaeraP} → {VB+}
209	{atopobiumP, gardnerellaP, gasseriA, inersA, jenseniA} → {VB+}
103	{atopobiumP, gardnerellaP, inersA} → {VB+}
124	{atopobiumP, gardnerellaP, jenseniA} → {VB+}
245	{atopobiumP, gardnerellaP, inersA, jenseniA, megasphaeraP} → {VB+}
296	{atopobiumP, crispatusA, gardnerellaP, inersA, jenseniA} → {VB+}

Likewise, all resulting rules comply with the biological significance requirement of having at least two bacteria present [12]. Biological significance adds weight to rules that carry bacteria and, at the same time, show high levels of lactobacillus iners.

5 Conclusions and Future Work

The experimental results shown have been validated by an expert biologist, who observed that multiple combinations of present bacteria (indicated with the letter P) and absent lactobacillus (indicated with the letter A) could lead to the disease appearance in the resulting rules.

Thus, the algorithm's behavior using the coding scheme and the fitness function lead to rules with biological significance. Furthermore, our results show that using DE to select association rules created with Apriori is a promising approach to identifying a high-quality and compact rule set for BV diagnosis.

In future work, it is crucial to continue with the validation of the rules by a human expert to corroborate their feasibility. Another point is to add penalties in the fitness function for antecedent itemsets unlikely to occur when there exists a positive consequent.

Additionally, new encoding schemes will be studied so that the number of selected rules is not previously defined. In this sense, it is also proposed to test with other DE variants and try different techniques for the algorithm-parameter-tuning to improve the algorithm performance.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: Proceedings of the International Conference on Management of Data, pp. 207–216 (1993). DOI: 10.1145/170035.170072.
2. Baker, Y. S., Agrawal, R., Foster, J. A., Beck, D., Dozier, G.: Applying Machine Learning Techniques in Detecting Bacterial Vaginosis. In: The International Conference on Machine Learning and Computing, vol. 1, pp. 241–246 (2014). DOI: 10.1109/ICMLC.2014.7009123.
3. Das, S., Suganthan, P. N.: Differential evolution: A Survey of The State-of-the-art. In: IEEE Transactions on Evolutionary Computation, vol. 15, pp. 4–31 (2011). DOI: 10.1109/TEVC.2010.2059031.
4. De la Cruz, F., Canul-Reich, J.: Reglas de asociacion para estudiar patrones bacterianos ´ involucrados en el desarrollo de vaginosis bacteriana. Komputer Sapiens, vol. 14, no. 2 (2022)
5. Dongre, J., Prajapati, G. L., Tokekar, S. V.: The Role of Apriori Algorithm for Finding the Association Rules in Data Mining. In: International Conference on Information and Computer Technologies, pp. 657–660 (2014). DOI: 10.1109/ICICICT.2014.6781357.
6. Guo, H., Li, Y., Liu, X., Li, Y., Sun, H.: An Enhanced Self-adaptive Differential Evolution Based on Simulated Annealing for Rule Extraction and its Application in Recognizing Oil Rservoir. Applied Intelligence, vol. 44, no. 2, pp. 414–436 (2016). DOI: 10.1007/s10489-015-0702-x.
7. Hahsler, M.: A Probabilistic Comparison of Commonly Used Interest Measures for Association Rules. Southern Methodist University (2015)
8. Leske, M., Bottacini, F., Afli, H., Andrade, B. G. N.: BiGAMi: Bio-objective Genetic Algorithm Fitness Function for Feature Selection on Microbiome Datasets. Methods and Protocols, vol. 5, no. 3 (2022). DOI: 10.3390/mps5030042.
9. Luna-Romera, J. M., Reyes, O., del Jesús-Díaz, M. J., Soto, S. V.: Reglas de asociacionnen datos multi-instancia mediante programación genética gramatical. In: Congreso de la Asociacion Española de Inteligencia Artificial: Avances en Inteligencia Artificial, pp. 815–820 (2018)
10. Pérez-Gómez, J. F., Canul-Reich, J., Hernández-Torruco, J., Hernández-Ocaña, B.: Predictor Selection for Bacterial Vaginosis Diagnosis using Decision Tree and Relief Algorithms. Applied Sciences, vol. 10, no. 9, pp. 3291 (2020). DOI: 10.3390/app10093291.
11. Price, K., Storn, R. M., Lampinen, J. A.: Differential evolution: A Practical Approach to Global Optimization (2006). DOI: 10.1007/3-540-31306-0.
12. Sanchez-Garcia, E. K., Contreras-Paredes, A., Martinez-Abundis, E., Garcia-Chan, D., Lizano, M., de la Cruz-Hernandez, E.: Molecular Epidemiology of Bacterial Vaginosis and Its Association with Genital Micro-organisms in Asymptomatic Women. Journal of Medical Microbiology, vol. 68, no. 9, pp. 1373–1382 (2019). DOI: 10.1099/jmm.0.001044
13. Storn, R., Price, K.: Differential Evolution – a Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces. Global Optimization, vol. 11, no. 4, pp. 341–359 (1997). DOI: 10.1023/A:1008202821328.
14. Wang, C., Liu, Y., Zhang, Q., Guo, H., Liang, X., Chen, Y., Xu, M., Wei, Y.: Association Rule Mining Based Parameter Adaptive Strategy for Differential Evolution Algorithms. Expert Systems with Applications, vol. 123, pp. 54–69 (2019). DOI: 10.1016/j.eswa.2019.01.035.
15. Zhang, C., Zhang, S.: Association Rule Mining: Models and Algorithms (2002). DOI: 10.1007/ 3-540-46027-6.

ISSN 1870-4069

Vehicle Make and Model Recognition with Generation of New Classes Using Clustering Techniques

Diana Itzel Vázquez-Santiago, Héctor Gabriel Acosta-Mesa, Efrén Mezura-Montes

Universidad Veracruzana,
Artificial Intelligence Research Institute,
Mexico

zs21000454@estudiantes.uv.mx, {heacosta, emezura}@uv.mx

Abstract. One of the main problems faced by supervised learning classification algorithms is scalability. No matter how good their classification accuracy is, they are not able to classify objects for which they were not trained. In this paper we propose a solution to this problem specifically aimed at vehicle make and model recognition. We used a Convolutional Neural Network (CNN) for classification and feature extraction, addressing the scalability problem by using two clustering techniques: K-means and MOCK. For the generation of new classes, we used the feature vectors extracted by the CNN of the images that do not belong to any of the classes with which the model was trained. The results showed that with the learning generated by a CNN it is possible to generate feature vectors with similarities for objects of the same class even if the network was not trained to classify them, which made it possible to generate new classes using unsupervised learning such as clustering.

Keywords: Scalability, CNN, clustering, MOCK.

1 Introduction

Automatic vehicle makes and model recognition aims to offer innovative services to improve the efficiency and safety of transportation networks. Some of these services are intelligent traffic analysis and management, electronic toll collection, emergency vehicle notifications, automatic enforcement of traffic rules, etc. The main problem, in the specific case of this application domain, is that most of the applied approaches for vehicle make and model recognition require large amounts of data to correctly train a model.

It is estimated that there are currently more than 3,300 vehicle makes in the world, which have added and removed models from the market, modifying the design in each generation and producing different versions of the same vehicle which has made impossible to have a database containing all existing vehicles.

Algorithm 1: PESA-II Pseudocode

```
1  Initialize a random (internal) population IP
2  Evaluate each member of IP
3  Initialize the external population EP to the empty set
4  repeat
5  Incorporate non-dominated vectors from IP into EP
6  Delete the current contents of IP
7  repeat
8  With probability Pc, select two parents from EP, where Pc= Crossover probability
9  Produce a single child via crossover
10 Mutate the child created in the previous step
11 With probability 1 – Pc, select one parent
12 Mutate the selected parent to produce a child
13 until the population IP is filled;
14 until termination criteria is met;
15 Return the members of EP as the result
```

This limitation leaves us with a scalability problem that results in vehicles that cannot be classified correctly because the algorithms were not trained to recognize them.

In this work, we propose a feasibly solution to the scalability problem of classification algorithms that use supervised learning by using clustering algorithms to generate new classes using the feature vectors extracted by our classification algorithm. In the state of the art, the scalability problem has been addressed by authors such as Nazemi et al. [1] from an anomaly detection approach.

Their base system is capable of classify 50 specific vehicle models, to which they added an anomaly detection to identify vehicles that do not belong to any of the 50 classes, to subsequently classify them based on their dimensions within 2 new classes: "Unknown heavy" and "Unknown light".

Other authors such as Kezebou et al. [4] proposed a Few-Shots Learning approach requiring between 1 and 20 images for the generation of new classes.

2 Methodology

For this work, the VMMRdb database [2] was used since it is one of the most cited in the specialized literature. With the intention of simplifying the problem for analysis, only five classes were used: Dodge Grand Caravan 2005, Ford Explorer 2002, Ford Mustang 2000, Nissan Altima 2005 and Toyota Camry 2007 to train a CNN whose architecture was proposed in the Microsoft technical documentation library [3] with which training and testing times of 2m24s were achieved with accuracies between 90%-95% in 5 epochs.

Since the main objective of this project is to have an algorithm capable of classifying vehicles even if they do not belong to any predefined class, an algorithm was designed

Table 1. Comparison between data volumes per class.

Class	Images (Rear view)	Training (Before augmentation)	Training (After augmentation)	Test (No augmentation)
Dodge Grand Caravan 2005	164	144	1,000	20
Ford Explorer 2002	234	214	1,000	20
Ford Mustang 2000	216	196	1,000	20
Nissan Altima 2005	294	274	1,000	20
Toyota Camry 2007	170	150	1,000	20

to generate new classes from the clustering of images that do not belong to the classes with which the CNN was trained.

The feature vectors of the images (extracted with the CNN) were used to perform clustering using two algorithms for comparison purposes. The first clustering algorithm is K-means.

The original version of the algorithm was implemented in Python and in the interests of have an additional comparison, a second version was developed using the sklearn library, which implements the K-means clustering algorithm. See section 2.3 for more details.

The second clustering algorithm named MOCK employs a multi-objective evolutionary approach named PESA-II. This algorithm attempts to minimize two objectives that are in conflicting with each other (intra-cluster variation and the number of clusters).

The concept of Pareto dominance is used to find a set of different non-dominated clustering solutions that achieve a good trade-off between the two objectives. PESA-II's pseudocode is shown in Algorithm 1. See section 2.4 for more details.

2.1 Image Preprocessing

The images went through a few processes to fit the model. The first was to segment the images of each class according to the views they showed (front, rear, and side). Due to the scope of the project, we work only with the rear views of the five classes mentioned in Section 2.

Table 1 shows the volume of images (rear views) available per class. Twenty images from each class were chosen for testing and the rest for training, however, as can be seen in Table 1 the volume of images was low and not balanced to adequately train the CNN.

To solve these problems, data augmentation and balancing processes were performed on the training set. After these processes, the final number of images for training per class was 1,000 and the twenty images that were originally selected for testing were kept without data enhancement.

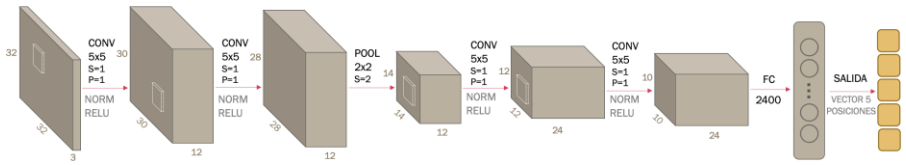


Fig. 1. Architecture of the convolutional neural network used and proposed in the Microsoft technical documentation library [3].

The next process was to reduce the dimensions of the images (training and testing) since the CNN architecture required input dimensions of 32x32x3. With the processes mentioned above, the classification model achieved accuracies between 80%-85%.

To improve recognition accuracy the images were cropped to preserve only the region of interest (ROI), which improve the accuracy by up to 10%.

2.2 Convolutional Neural Network (CNN)

This project was implemented in Python mainly because of the access to the PyTorch library which provides support for the development of applications related to machine learning, computer vision, natural language processing, etc. More specifically, it has a base class for all Convolutional Neural Networks modules, which facilitates its implementation and execution.

Initially in this project, it was proposed to work with well-known CNN architectures such as AlexNet or VGG, however, in the first stages, tests were carried out and execution times were time consuming (30min-50min) achieving a maximum accuracy of 60%.

Because of this, we chose to use an architecture found in the Microsoft technical documentation library [3], which can be seen in Fig 1. Originally, the network was designed to work with the CIFAR10 database, so the input dimensions were 32x32x3.

Even with the possible loss of information, it was decided to keep this architecture and resize the images of the VMMRdb database to fit, since even without the data balance, the cropping of the ROI and with the resizing, accuracies of 70% were achieved. The only modification to the architecture was to change the output of the fully connected layer to five (number of classes).

In the training stage, 20-image subsets of the training data were generated and reorganized at each epoch to reduce overfitting. For the update of the network weights during training, the Adam optimizer was used with a learning rate of 0.001 and a weight decrease of 0.0001.

Finally, for the performance evaluation, two indicators were used in each epoch, the first indicator was to evaluate the classification accuracy of the network with the whole test set (20 images as shown in Table 1) and the second indicator was with the Cross Entropy loss function which is mathematically expressed in (1). Where i is the class, c is the number of classes, y_i is the actual class and $\hat{y}_i$ is the predicted class:

$$-\sum_{i=1}^{c} y_i \log \hat{y}_i, \tag{1}$$

Table 2. Comparison between Convolutional Neural Network executions. The execution with the most accurate result is highlighted in bold.

Execution	Cross Entropy Loss	Accuracy	Accuracy per class				
			Dodge Grand Caravan 2005	Ford Explorer 2002	Ford Mustang 2000	Nissan Altima 2005	Toyota Camry 2007
1	0.294	**95%**	95%	100%	95%	90%	95%
2	0.290	91%	100%	85%	90%	90%	90%
3	0.295	92%	100%	95%	90%	95%	80%
4	0.313	93%	95%	95%	85%	95%	95%
5	0.300	91%	95%	100%	95%	90%	75%
6	0.311	92%	90%	100%	85%	100%	85%
7	0.247	90%	100%	95%	90%	90%	75%
8	0.313	93%	100%	90%	90%	85%	100%
9	0.303	91%	90%	85%	95%	95%	90%
10	0.321	94%	100%	85%	95%	95%	95%
Average		92.2%	97%	93%	91%	93%	88%

2.3 K-means Clustering

As mentioned above, the aim of this project is to have an algorithm capable of classifying vehicles even if they do not belong to any predefined class. The first approach implemented for the generation of new classes was K-means clustering. Two implementations of this algorithm were developed.

The first was implementing the classical version of the algorithm. The second was using the version developed by sklearn library, initializing the centroids of the clusters with the Lloyd algorithm.

We decided to test the k-means algorithm in its two versions with nine images of three unknown classes to the CNN (three images of each class), to confirm if the clustering algorithms were able to group the images by model. The clustering process was performed using the feature vectors extracted by the CNN, which, as can be seen in Fig 1. had 2,400 features.

2.4 MOCK Clustering with Multi-Objective Evolutionary Approach (PESA-II)

The second approach implemented for the generation of new classes was a multi-objective clustering algorithm with automatic determination of the number of clusters (MOCK) optimized with a multi-objective evolutionary algorithm (MOEA), called PESA-II proposed by Corne et al. [5].

The encoding of individuals uses a representation where each individual g is made up of N genes, $g1,...,gN$, where N is the number of data to be clustered and the value j

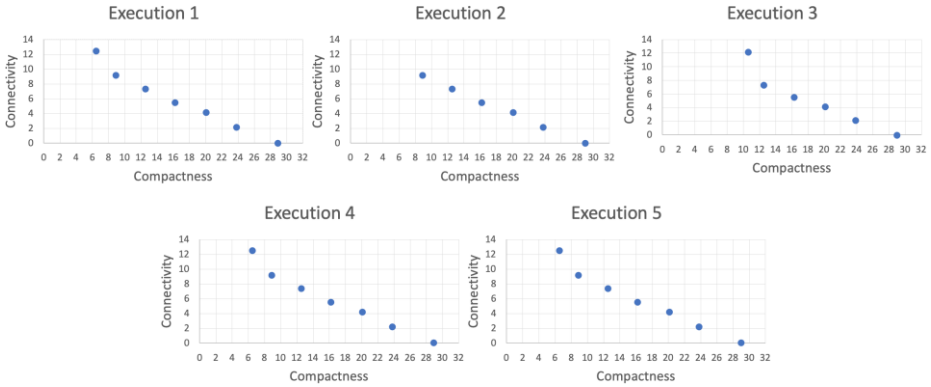


Fig. 2. Pareto fronts obtained in 5 executions of the MOCK algorithm.

assigned to the *i*-th gene represents a union between the *j* and *i* data. A minimum spanning tree (MST) was generated with the Prim algorithm to initialize the population.

The MST represented the first individual and for the subsequent generation of the population the (*i* - 1) longest connections were eliminated. In the genotype, this was reflected by assigning to itself the gene representing the connection. The decoding of this representation requires the identification of all the subgraphs since the data belonging to each subgraph is assigned to a cluster.

The variation operators used are the uniform crossover and the nearest neighbor mutation operator, which limits the search space since it can only generate connections between the nearest neighbors of the gene being mutated. In this approach, there are two objective functions to be minimized:

1. **Cluster Compactness** or global deviation, which is calculated by summing the distances between each datum and its corresponding centroid in a given cluster and is mathematically represented as:

$$\text{Dev}(C) = \sum_{C_k \epsilon C} \sum_{i \epsilon C_k} \delta(i, \mu_k), \tag{2}$$

where *C* is the set of clusters M_k is the centroid of cluster C_k, *i* is each element of the data set and δ(...) is the Euclidean distance.

2. **Cluster Connectivity** which evaluates if the nearest neighbors of an element have been placed in the same cluster as the current element and is mathematically represented as:

$$\text{Conn}(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_i(j)} \right), \text{ donde } x_{r,s} = \begin{cases} \frac{1}{j} \; si \; \nexists C_k : r, s \in C_k, \\ 0 \text{ de lo contrario,} \end{cases} \tag{3}$$

where *C* is the set of clusters, *N* is the amount of data in the dataset, *L* is the amount of nearest neighbors (user-defined parameter), $nn_i(j)$ is the *j*th nearest neighbor and $x_{r,s}$ is the penalty function.

There will only be penalties if any *j*th nearest neighbor is not in the same cluster as the *i*th data. PESA- II's pseudocode is shown in Algorithm 1 where the use of two populations of solutions can be highlighted: IP which has a fixed size and is responsible

for exploring new solutions and EP which has a limited, but not fixed size and has the job of exploiting good solutions since it consists of "niches" distributed over the objective space (Pareto Front).

In each generation, the generated solutions (IP) are evaluated and those that are in the objective space are selected to become part of EP, preferring those solutions that occupy less crowded spaces (the "niches" are avoided) to try to completely cover the Pareto Front.

3 Results

To address the problem of scalability of classification algorithms that use supervised learning, it is essential for our proposal to have a classifier with good recognition accuracy to differentiate between images that belong to the predefined classes from those that do not.

To test the classification accuracy of our model the CNN implemented and detailed in Section 2.2 was trained and tested 10 times with the 5 classes of the VMMRdb database [2] mentioned in Section 2 which went under the data augmentation and balancing processes mentioned in Section 2.1. Table 2 shows the accuracy results obtained in the 10 training-testing executions.

Due to the scope of the project, it is left as future work the implementation of a novelty detection technique to automatically detect images that do not belong to the predefined classes to which the clustering will be applied to generate new classes.

Given the above, in order to show that the K-means and MOCK algorithms were able to cluster the images of the unknown classes for the CNN and thus generate new classes, nine images of three unknown models (three of each class) were entered into the CNN: Ford Ranger 2019, Toyota Prius 2019 and Volkswagen Beetle 2013 which, as expected, generated a misclassification as it was not trained for those classes, however, what was important in this case was to obtain the feature vectors generated by the CNN to enter them into the clustering algorithms.

For the execution of MOCK the parameters were calibrated as follows: Number of generations = 100, Maximum External Population Size = 15, Internal Population Size = 8, L nearest neighbors = 3, Crossover Probability = 0.5. Fig. 2 shows the Pareto Fronts obtained in five executions of the algorithm.

It is important to remember that the clustering process by definition is subjective. Also, it should be considered that each time the CNN is trained, it will learn in a unique way and will be reflected in the weights that are set at the end of the training, therefore, the features extracted after each training will depend on those learned weights.

Taking these two points into consideration, five tests of the clustering algorithms were performed using five sets of feature vectors obtained from five executions of the CNN with different weights. Only with one of the sets the desired clustering was achieved by both clustering approaches as shown in Fig. 3.

Initially the hypothesis was that MOCK would outperform K-means, however the results showed similar performances where the only advantage shown by MOCK was the automatic determination of the number of clusters. To get a better understanding of the results, five distance matrices of the five sets of feature vectors extracted by the

Fig. 3. Classification expected and obtained with the K-means and MOCK algorithms.

CNN were performed using the Euclidean distance since the same metric was used in the clustering algorithms.

This test showed that only in one of the matrices the feature vectors belonging to the same classes were spatially close and it was with that set that the clustering algorithms achieved the desired clustering.

One point to highlight with the K-means approach is that the addition of the Lloyd algorithm implemented by sklearn library for the initialization of the centroids gave it a great advantage over the classical version that did not achieve the expected classification in any of the executions.

Finally, it was noted that in most cases the clustering were related to the predominant shades in the images, which indicates that it may be preferable to work with grayscale images to prevent bias.

4 Conclusions and Future Work

In this work a feasibly solution to the scalability problem was presented, confirming that it is possible to generate new classes using clustering algorithms to group images based on the feature vectors extracted by the classification model even if the classes are unknown. However, it was observed that the feature vectors belonging to the same classes were not in close regions in terms of Euclidean distance.

The research carried out after this project reflected that this distance metric is recommended only to compare points in two or three dimensions. In larger dimensional spaces, all points tend to be far apart, then other measures will be explored such as the cosine distance. Another point to consider is that the feature vectors will depend on the training of the network, specifically the learned weights, which can influence negatively.

To achieve a better control, we propose the use of neuroevolution of CNNs with an objective function that measures the consistency of the feature maps. By doing this, we could get feature vectors located spatially close if they belong to the same class, and far away if they belong to different classes. In the literature review, this has been achieved using techniques such as Contrastive Loss [7].

Regarding the clustering algorithms, according to authors such as Martínez-Peñaloza et al. [6], better results were achieved using the MOEA NSGA-II compared to the original version of MOCK, which uses PESA-II.

Since in this work there were no explicit differences between K-means and MOCK, it is proposed to use the optimized version with NSGA-II to try to achieve better results. Finally, it is proposed as future work to increase the number of classes and perform a novelty detection to automatically recognize vehicles that do not belong to the labels with which the CNN is trained and perform clustering on them.

Acknowledgments. The authors would like to thank the Consejo Nacional de Ciencia y Tecnología (CONACYT), an institution of the Government of Mexico, for the financial support provided through the "Beca Nacional" as part of the Programa de Becas para Estudios de Posgrado.

References

1. Nazemi, A., Azimifar, Z., Shafiee, M., Wong, A.: Real-time vehicle make and model recognition using unsupervised feature learning. IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 7, pp. 3080–3090 (2019). DOI: 10.1109/TITS.20 19.2924830.
2. Tafazzoli, F., Frigui, H., Nishiyama, K.: A large and diverse dataset for improved vehicle make and model. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1–8 (2017). DOI: 10.1109/CVPRW.2017.121.
3. Kezebou, L., Oludare, V., Panetta, K., Agaian, S.: Few-shots learning for fine-grained vehicle model recognition. In: IEEE International Symposium on Technologies for Homeland Security, pp. 1–9 (2021). DOI: 10.1109/HST53381.2021.9619823.
4. Corne, D. W., Jerram, N. R., Knowles, J. D., Oates, M. J.: PESA-II: region-based selection in evolutionary multiobjective optimization. In: Proceedings of the 3rd annual conference on genetic and evolutionary computation, pp 283–290 (2001). DOI: 10.5555/29552 39.2955289.
5. Martínez-Peñaloza, M. G., Mezura-Montes, E., Cruz-Ramírez, N., Acosta-Mesa, H. G., Ríos-Figueroa, H. V.: Improved multi-objective clustering with automatic determination of the number of clusters. Neural Computing and Applications, vol. 28, no. 8, pp. 2255–2275 (2017). DOI: 10.1007/s00521-016-2191-1.
6. Wang, F., Liu, H.: Understanding the behaviour of contrastive loss. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2495–2504 (2021). DOI: 10.1109/CVPR46437.2021.00252.

ISSN 1870-4069

Western Blot Pattern Classification Using Convolutional Neural Networks for Breast Cancer Diagnosis

José Luis Llaguno-Roque[1], Rocio Erandi Barrientos-Martínez[2], Héctor Gabriel Acosta-Mesa[2], Tania Romo-González[1]

[1] Universidad Veracruzana,
Instituto de Investigaciones Biológicas,
Mexico

[2] Universidad Veracruzana,
Instituto de Investigaciones en Inteligencia Artificial,
Mexico

{lllaguno, rbarrientos, heacosta, tromogonzalez}@uv.mx

Abstract. In Mexico, breast cancer is the leading cause of women's death. This work aims to discriminate between healthy and breast cancer patients based on the band patterns obtained by western blotting using deep learning techniques. This work proposes Convolutional Neural Networks (CNN) to classify breast cancer. CNN reaches 68.24% of the classification rate in three classes (healthy, benign breast pathology, breast cancer) and 81.43% in two class labels (healthy, breast cancer).

Keywords: Breast cancer, convolutional neural networks, western blot, Fourier transform.

1 Introduction

Breast cancer has become a global health problem since it represents the first place in incidences and the fifth place in cancer mortality worldwide [1]. In Mexico, breast cancer is the leading cause of death in women between the ages of 30 and 54, surpassing cervical cancer since 2006, becoming a public health problem and a severe challenge for the health system [2].

Several methods complementing each other as a whole are proposed for its diagnosis. These methods include a clinical breast examination, ultrasound, mammography, and biopsy. However, these methods are ineffective in the early cancer detection, since they aim is to identify the disease. Moreover they are invasive, subjective, expensive, and in sometimes painful [3-4].

In contrast to the traditional methods for breast cancer diagnosis, some other techniques detect tumor particles before the disease develops. In other words, these

Fig. 1. Example of an image containing 15 to 17 nitrocellulose membrane strips obtained from the Western Blot method for specific protein antigens (T47D) in each patient's serum sample.

methods identify the autoantibodies dedicated to recognizing tumor proteins present up to 4 years before the disease detection [5].

For example, Desmetz et al. [6] discriminate accurately between healthy patients and those with early-stage breast cancer, especially carcinoma in situ, by evaluating the autoantibody response to a set of tumor-associated antigens. The result obtained from this work could help in the early detection of breast cancer, especially in women under 50.

Similarly, Romo-González et al. [7] describe a method that corroborates the presence of autoantibodies against tumor cells of the T47D cell line (ductal carcinoma of the breast), allowing distinguishing women with and without breast pathology. In this work, the bands' analysis expressed in the one-dimensional Western Blot images in which the autoantibodies react of the T47D tumor line antigens.

Although the results are promising, the image analysis is very complex, subjective, and time-consuming, taking up to a month to create the binary database. It is because image analysis requires the expert to align the bands of each patient's strips with the Quantity One software from Bio-Rad Laboratories (Fig. 1). As a result, the final bands' identification and their position depend on the expert eye.

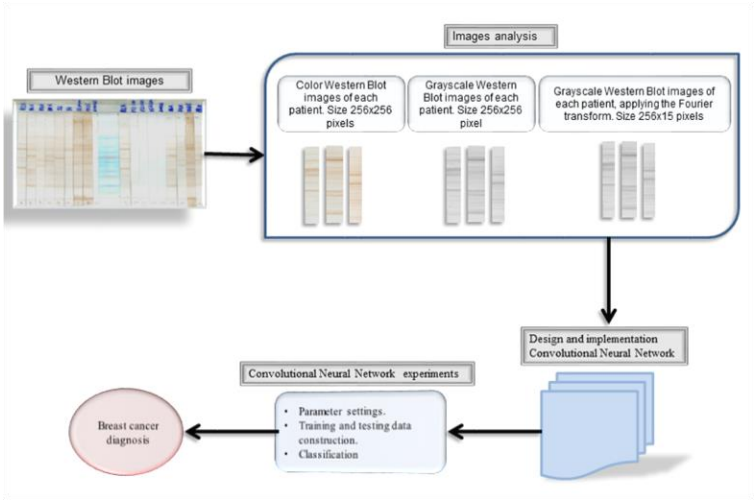


Fig. 2. Proposed Methodology.

A semi-automated protein band analysis system was designed to avoid subjectivity and delays due to image analysis, classifying band patterns by time series [8]. The time series data corresponds to the band's pixel shade variation.

Since time series were different lengths, they were adjusted to the same length with a geometric scaling transformation. Afterward, the K-nearest neighbor algorithm with Euclidean, Mahalanobis, and Correlation similarity distances was used for classifying time series. This method reaches a classification rate of 65.40% with three classes (healthy, benign breast pathology, breast cancer) and 86.06% with two class labels (healthy, breast cancer).

Although the classification rate achieved was high and similar to the expert, the method is considered semi-automatic since an area is subjectively chosen in each strip for the band analysis, resulting in a variation of the time series length.

For this reason, in the present work, we proposed to discriminate between healthy patients, patients with benign breast pathology, and patients with cancer using the bands of Western blot images of antigen-reactive antibodies (tumor line T47D - ductal carcinoma) and convolutional neural networks.

Our primary objective is to reach a classification rate of 84% at least, avoiding subjectivity and analyzing images directly instead of extracting time series from selected areas [8].

2 Methodology

Figure 2 shows the proposed methodology. The employed database contains 149 nitrocellulose membrane strips images with band expression obtained from the Western Blot of autoantibody binding to specific protein antigens (T47D), of which 50 correspond to patients with breast cancer, 50 with benign breast pathology, and 49 to healthy patients.

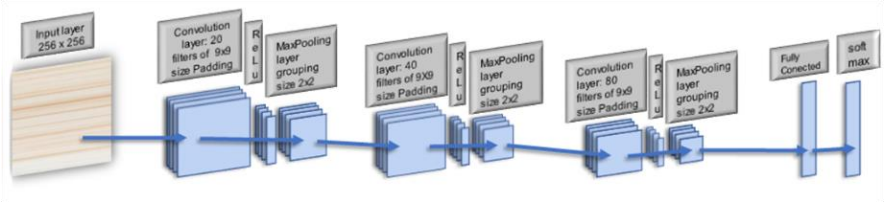


Fig. 3. Convolutional Neural Network Architecture.

These images were provided by the Biology and Integral Health Area of Instituto de Investigaciones Biologicas of the Universidad Veracruzana, following the ethical norms and with the corresponding informed consent of the participants. Furthermore, the protocol was reviewed and approved by the Research Ethics Committee of the Hospital General de México "Dr. Eduardo Liceaga" (DI/12/111/03/064).

Finally, it is essential to mention that this study conforms to the Code of Ethics of the World Medical Association (Declaration of Helsinki), printed in the British Medical Journal (July 18, 1964). The images have been used in 3 ways: 1) Color, with dimensions of 256x256 pixels, 2) Grayscale, with dimensions of 256x256 pixels, 3) Grayscale applying the Fourier transform, with dimensions of 256x15 pixels.

To achieve the classification of western blot bands in healthy patients and cancer patients, a Convolutional Neural Network was trained, which was designed by iteratively and manually adjusting the number and type of hidden layers as well as the parameters of each one of them. The architecture had the following features (Fig. 3):

- Input layer: 256 x 256.
- Convolution layer: 20 filters of 9x9 size Padding.
- Normalization Layer / ReLu Layer.
- maxPooling layer - grouping size 2x2.
- Convolution layer: 40 filters of 9X9 size Padding..
- Normalization Layer / ReLu Layer.
- maxPooling layer - grouping size 2x2.
- Convolution layer: 80 filters of 9x9 size Padding.
- Normalization Layer / ReLu Layer.
- Fully connected layer.
- Classification layer with softmax method for values normalization.

Once the network architecture is defined, the database is divided into the training set (70% of the images) and the test set (30% rest) for the classification task.

Table 1. Experiment 1: Classification with Convolutional Neural Networks with three classes.

Test	Epochs	Kernel	Classification rate	Std dev (-+)	P<0.05 Significant differences
Sánchez-Silva, Acosta-Mesa, & Romo-González, 2018	N/A	N/A	65.40%	N/A	N/A
Color images	20	9	68.24%	62.67% - 73.77%	0.26[1]
Grayscale images.	20	9	66.44%	63.78% - 69.11%	0.03754[1]
Grayscale images applying Fourier Transform	70	3	61.55%	54.76% - 68.34%	0.5964

3 Experiments and Results

Our proposal was programmed and executed in Matlab software. The results described in this section were designed according to two experiments explained below.

Experiment 1. All images were considered with the three classes of the database: healthy patients, patients with benign pathology, and patients with cancer. In the case of the convolution kernels, the kernel sizes used were 3, 7, and 9 coefficients.

As far as the type of data is concerned, three variants were employed:Color images, 70 epochs were tested:

- Color images and 20 epochs were tested.
- Grayscale images and 20 epochs were tested.
- Grayscale images applying Fourier transform, 70 epochs were tested.

Experiment 2. 99 images were used, representing two classes in the database corresponding to healthy and cancer patients. In the case of convolution kernels, the kernel sizes used were 3, 7, and 9 coefficients. As far as the type of data is concerned, three variants were used:

- Color images and 20 epochs were tested.
- Grayscale images and 20 epochs were tested.
- Grayscale images applying Fourier transform, 70 epochs were tested.

[1] Significance values were evaluated using non-parametric techniques.

Table 2. Experiment 2: Classification with Convolutional Neural Networks with two classes.

Test	Epochs	Kernel	Classification rate	Std. dev (-+)	P<0.05 Significant differences
Sánchez-Silva, Acosta-Mesa, & Romo-González, 2018	N/A	N/A	86.06%	N/A	N/A
Color images	20	3	81.99%	77.50% - 86.96%	0.2223[2]
Grayscale images.	20	7	82.33%	74.95% - 89.71%	0.5097[2]
Grayscale images applying Fourier Transform	70	3	86.00%	81.90% - 90.09%	0.351

When the test data satisfied the assumption of normality, Analysis of Variance (ANOVA) was used to evaluate significant differences between more than two groups.

Otherwise, the non-parametric Kruskal-Wallis test was used. Furthermore, the t-student test was used to assess the significant differences between two groups.

On the other hand, the Mann-Whitney test was employed in case the data did not satisfy the normality assumption. If $p<0.05$, the data had significant differences. Both experiments were run ten times, calculating the average, standard deviation, and evaluation of significant differences in the classification percentage obtained in each run. The results are shown in Table 1 and Table 2.

Figure 4 and 5 show the better confusion matrix for both experiments, these allows visualization of the performance of the proposal model. Each row of the matrix represents the instance in the actual class and each column represents the instance in a predicted class.

With this matrix is possible calculate the false negatives, false positives, true negative and true positives values. This allows more detailed analysis than simply observing the proportion of correct classifications (accuracy).

4 Discussion

As show in the tables, we have the comparative results obtained by testing the convolutional neural network with proposed parameters, color spaces in the image, and applying the Fourier transform in the grayscale images.

The best classification rate for the three classes (healthy, benign breast pathology, and breast cancer) was 68.24% in color images (62.67% - 73.77%) and a $p=0.26$ significant difference.

The best classification rate for two classes (healthy, breast cancer) was 86.00% in grayscale images applying Fourier transform (81.90% - 90.09%) with a $p=0.351$ significant difference. For the experiments conducted, we could conclude that exhaustive processing that uses a lot of time and resources is unnecessary, since from

[2] Significance values were evaluated using non-parametric techniques.

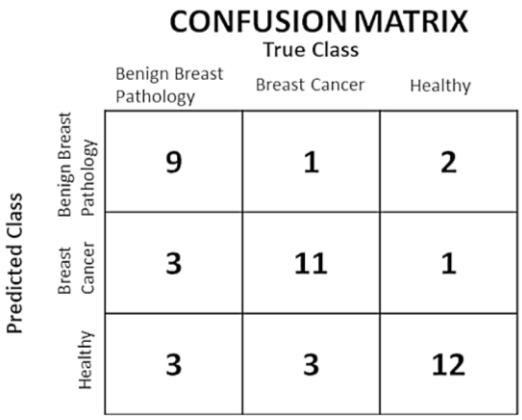


Fig. 4. Confusion Matrix of the Convolutional Neural Networks with three classes.

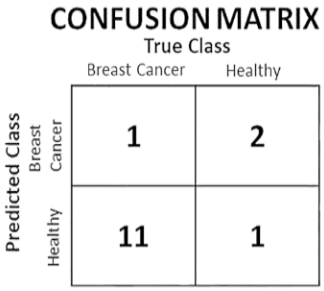


Fig. 5. Confusion Matrix of the Convolutional Neural Networks with two classes.

epoch 20 with color and grayscale images, or in epoch 70 with images applying the Fourier transform, the classification rate remains constant.

Thus, we can conclude image processing performed with convolutional neural networks reduces time and subjectivity compared to those analyses a proteomics specialist would perform with these images.

Our proposal directly classifies the bands of Western blot images of antigen-reactive autoantibodies (tumor line T47D - ductal carcinoma) without a preprocessing stage (delimiting an area to obtain time series).

These results guide us to continue experimenting on how convolutional neural networks allow us to get a better classification rate. On the other hand, artificial intelligence is applied as a support tool to diagnose breast cancer before it manifests itself, leading to better prevention, diagnosis, and treatment of breast cancer.

References

1. Sung, H., Ferlay, J., Siegel, R.L., Laversanne, M., Soerjomataram, I., Jemal, A., Bray, F.: Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries CA: A Cancer Journal for Clinicians, vol. 71, no. 3, pp. 209–249 (2021). DOI: 10.3322/caac.21660.

2. Hernández-Nájera, O., Cahuana-Hurtado, L., Ávila-Burgos, L.: Costos de atención del cáncer de mama en el Instituto de Seguridad y Servicios Sociales de los Trabajadores del Estado, México. Salud Pública de México, vol. 63, no. 4, pp. 538–546 (2021). DOI: 10.21149/12332.
3. Brandan, M.E., Villaseñor, Y.: Detección del cáncer de mama: Estado de la mamografía en México. Cancerología, vol. 1, no 3, pp. 147–162 (2006)
4. Chapman, C., Murray, A., Chakrabarti, J., Thorpe, A., Woolston, C., Sahin, U., Barnes, A., Robertson, J.: Autoantibodies in Breast Cancer: Their Use as an Aid to Early Diagnosis. Annals of Oncology, vol. 18, no. 5, pp. 868–873 (2007). DOI: 10.1093/annonc/mdm007.
5. Desmetz, C., Bascoul-Mollevi, C., Rochaix, P., Lamy, P.J., Kramar, A., Rouanet, P., Maudelonde, T., Mangé, A., Solassol, J.: Identification of a New Panel of Serum Autoantibodies Associated with the Presence of in Situ Carcinoma of the Breast in Younger Women. Clinical Cancer Research, vol. 15, no. 14, pp. 4733–4741 (2009). DOI: 10.1158/1078-0432.CCR-08-3307.
6. Romo-González, T., Esquivel-Velázquez, M., Ostoa-Saloma, P., Lara, C., Zentella, A., León-Díaz, R., Lamoyi, E., Larralde, C.: The Network of Antigen-antibody Reactions in Adult Women with Breast Cancer or Benign Breast Pathology or Without Breast Pathology. Plos One, vol. 10, no. 3, pp. e0119014 (2015). DOI: 10.1371/journal.pone.0119014.
7. Sánchez-Silva, D.M., Acosta-Mesa, H.G., Romo-González, T.: Semi-Automatic Analysis for Unidimensional Immunoblot Images to Discriminate Breast Cancer Cases Using Time Series Sata Mining. International Journal of Pattern Recognition and Artificial Intelligence, vol. 32, no. 01, pp. 1860004 (2018). DOI: 10.1142/S0218001418600042.

ISSN 1870-4069

Spiking Neural Networks Codification Using Bio-Inspired Computation

Carlos Alberto López-Herrera, Héctor Gabriel Acosta-Mesa, Efrén Mezura-Montes

Universidad de Veracruz,
Instituto de Investigación en Inteligencia Artificial,
Mexico

carlosalberto.lopezherrera91@gmail.com,
{heacosta,emezura}@uv.mx

Abstract. In Spiking Neural Networks, the codification of analog signals constitutes a primordial pre-processing step. Hereof, Ben's spiker algorithm, as a temporal coding schemes, is one of the most recently used methods. Nevertheless, having optimal parameters is of great importance. In this paper, the performances of two evolutionary algorithms and one swarm intelligence algorithm are contrasted in said optimization task. Moreover, a comparison against a Grid Search implementation is also presented. Our findings showed that Differential Evolution outperformed its counterparts. Furthermore, it is also proved that the same transformation capabilities, as the Grid Search, are being reached.

Keywords: Ben's spiker algorithm, differential evolution, particle, swarm optimization, genetic algorithm, grid search.

1 Introduction

Spiking Neural Networks (SNNs), the third generation of Artificial Neural Networks (ANN), were introduced as a more biologically realistic approximation [1] regarding how information is spread, compared to past generations. In the brain, the interaction between neurons is done by transmitting action potentials (or spike trains) to other nearby neurons [2].

Since all real-world signals are characterized as analog and temporal, it becomes indispensable to implement a technique capable of transforming them into spike trains and preserve as much information as possible in order to harness the usage of SNNs.

These encoding methods are often divided into two approaches: Rate and Temporal coding schemes [3]. The Rate coding strategy focuses on how information is encoded (count, density or population rate) [3, 4].

On the other hand, Temporal coding methods encode signals based on the timing of significant events [5, 6]. Furthermore, it has been noted that rate coding suffers from wide periods of latency between spikes, which may not be suitable for some SNNs applications [4]. For that reason, temporal encoding has been used in more recent works [4].

Algorithm 1 BSA encoding

input: *S* signal, FIR filter, threshold
$L \leftarrow \text{length(S)}$, $F \leftarrow \text{length(FIR)}$, $\text{Out} \leftarrow \text{zeros}(L)$, $\text{Shift} \leftarrow \min(S)$
$S \leftarrow S - \text{Shift}$
for $t = 1 : (L - F)$ **do**
    $E1 \leftarrow 0$, $E2 \leftarrow 0$
    for $k = 1 : F$ **do**
        $E1 \leftarrow E1 + \text{abs}(S(t + k) - \text{FIR}(k))$, $E2 \leftarrow E2 + \text{abs}(S(t + k - 1))$
    if $E1 \leq (E2 * \text{threshold})$ **then**
        $\text{Out}(t) = 1$
        for $k = 1 : F$ **do**
            $S(t + k + 1) \leftarrow S(t + k + 1) - \text{FIR}(k)$
output: *Out*, *Shift*

Algorithm 2 BSA decoding

input: *Spikes*, *FIR* filter, *Shift*
$L \leftarrow \text{conv}(\text{Spikes}, \text{FIR}) + \text{Shift}$
output: *Out*

For a more in-depth analysis for these schemes, [3] provides a comprehensive review of the subject. Moreover, there are many temporal coding algorithms that have been proposed: Step-Forward (SF), Threshold-Based Representation (TBR), Moving Window (MW) and Ben's Spiker Algorithm (BSA), to name a few [3, 7]. Primarily, the latter has been used to encode data streams (e.g., Electroencephalography) [3, 7].

First introduced in [8], BSA is an extension of Hough Spiker Algorithm (HSA). The core idea behind this technique is that an analog signal can be constructed using the convolution of a spike train and a FIR filter [7, 9]. Hence, BSA uses a suitable filter to produce a spike train based on the comparison of two errors.

The first one involves the sum of differences between the signal and the filter. The second one represents the aggregated value of the signal; a spike is produced whenever the first error is smaller than the weighted (by a threshold) second error [5] (Algorithm 1).

Consequently, the BSA decoding is achieved by the convolution of the encoded spike train signal and the FIR filter (Algorithm 2). Thus, it is evident that the composition of the FIR filter and the threshold value are of great importance. The configuration of this filter relies on two main parameters: Filter size and Cutoff frequency.

In this preliminary proof of concept, two main goals are pursued: To compare the performance of two well known evolutionary algorithms (EA) and one swarm intelligence algorithm (SI) for the optimization of the BSA parameters (Filter size, cutoff frequency and threshold) and to contrast the best performing EA or SI against a Grid Search (GS).

Moreover, the reason to choose GS as a comparative method is not only because it is a deterministic technique, but also because it was used in [7] as a optimization technique. In order to measure the BSA efficiency, three metrics criteria will be used:

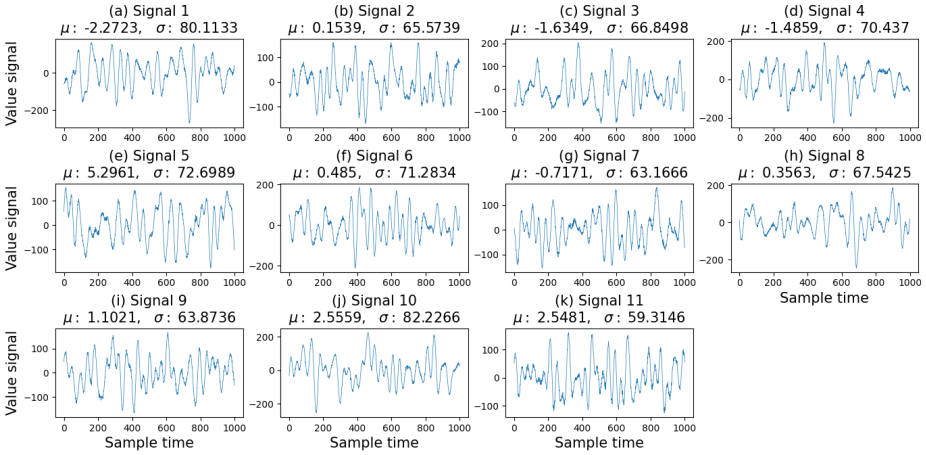


Fig. 1. All signals created with a length of 1001 elements and sampled at 1000 Hz.

Table 1. Range for each variable as established in [7].

Variable	Range	Increment
Filter size	17 - 81	4
Cutoff frequency	20 - 80	2
Threshold	0.8 - 1.1	0.01

– **Signal to Noise Ratio (SNR):** Measures the relation involving the original signal power and the noise signal power. Noise is considered as the difference between the original signal (s) and the decoded signal (r). Higher SNR values mean better results. It is defined as:

$$\mathrm{SNR} = 10 \cdot \log_{10} \left(\frac{\sum_{t}^{N} s_t^2}{\sum_{t}^{N} (s_t - r_t)^2} \right). \tag{1}$$

– **Absolute Firing Rate (AFR):** Indicates the saturation of the spike train (sp). Lower AFR values mean a less saturated signal. It is defined as:

$$\mathrm{AFR} = \frac{\sum_{t}^{N} |sp_t|}{N}. \tag{2}$$

– **Symmetric Mean Absolute Percentage Error (sMAPE):** A percentage error that measures accuracy between the original signal and the reconstructed one [10], [11]. Unlike SNR, this metric considers both, the original signal (s) and the reconstructed signal (r) as independent from each other. Lower sMAPE values mean better results. It is defined as:

$$\mathrm{sMAPE} = \frac{1}{N} \sum_{t=1}^{N} \frac{|r_t - s_t|}{|r_t| + |s_t|} \cdot 100\%. \tag{3}$$

Table 2. Parameter values used for each EA and SI algorithm. These values were selected by a trial-and-error process.

(a) GA

Version	Canonical (Real representation)
Population	50
Crossover	SBX(η+10 - 90%
Mutation	Uniform - 60%
Parent selection	Probabilistic binary tournament - 60%
Elitism	1
Boundary management	Ran[13]
Generations	200

(b) DE

Version	DE/rand/1/bin
Representation	Real
NP	50
Cr	0.8
F	0.5
Boundary management	Ran[13]
Generations	200

(c) PSO

Version	Global-best PSO
Representation	Real
Cumulus size	50
W	0.65
C_1	1.2
C_2	1.4
Boundary management	Ran&RaB[14]
Generations	200

Table 3. Statistical results of 30 independent executions. Values in boldface indicate the best value. **H=1** means that a significant difference was found.

Statistic	GA	DE/rand/1/bin	Global-best PSO	Friedman test	
				p-value	**H**
Best	10.7257	**10.7826**	10.7825		
Mean	10.6966	**10.7726**	10.7294		
Median	10.7024	**10.7825**	10.7250	2.46E-13	1
Worst	10.6349	**10.7310**	10.5168		
Std Dev	0.0206	**0.0151**	0.0497		

The rest of this paper is structured as follows: In Section 2, the methodology for two experiments to be conducted is explained. Section 3 describes the experiments layout, as well as the corresponding results. In Section 4, a general discussion is made of the achieved results and the evidence observed. Section 5 consists of some conclusions attained as well as ideas for future work.

2 Methodology

Using the implementation of [7], eleven signals were created (Fig. 1). These are produced by a composition of sine signals ranging from 2 to 30 Hz with random power and random phase lags. Also, white noise was added with a strength of 3. All signals have a length of 1001 elements, sampled at 1000 Hz.

The first experiment consists in comparing performances for the optimization of the BSA parameters. Two commonly used EA are considered. Namely, Genetic Algorithm (GA) and Differential Evolution (DE). Also, a SI algorithm called Particle Swarm Optimization (PSO) is tested as well. Since SNR is highly recommended [6, 7, 12], this metric was used as the objective function for the three compared approaches. The first signal (Fig. 1a) was utilized.

In the second experiment, the optimization method[1] proposed in [7], where a GS is employed to find the optimal set of parameters, was applied to each remaining signal (Fig. 1b - 1k). After that, the best performing EA or SI from the previous experiment was used to the same task on the same signals.

This test aims at proving the transformation capabilities of an EA or SI against a deterministic and proved method. The GS ranges of each variable are shown in Table 1. Moreover, the parameters for each bio-inspired algorithm are presented in Table 2.

[1] github.com/KEDRI-AUT/snn-encoder-tools

Table 4. Details of the median run, by each algorithm. a) Metrics comparing the results of the algorithms. Values in boldface indicate a better result. b) Set of parameters found by each algorithm.

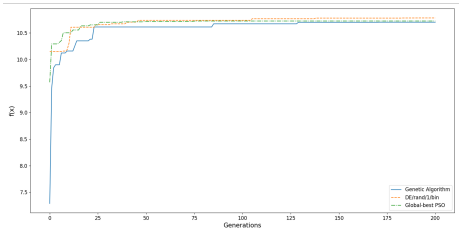
(a) Metrics achieved

	GA	DE/rand/1/bin	Global-best PSO
SNR	10.7024	**10.7825**	10.7250
sMAPE	13.6305	**13.2633**	13.5431
AFR	0.3337	0.3357	**0.3337**

(b) Parameters found

	GA	DE/rand/1/bin	Global-best PSO
Filter size	71	69	71
Cutoff frequency	49.7601	38.8649	49.1676
Threshold	0.9563	0.9572	0.9564

(a) Convergence graph.



(b) Visual comparison

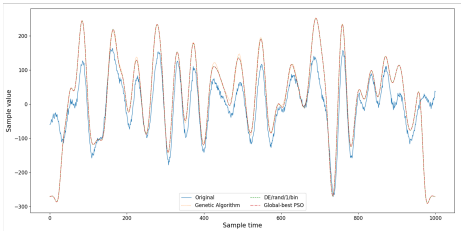


Fig. 2. Graphical results obtained. a) Convergence graph of the median execution by each algorithm. b) Visual comparison between the original signal and the reconstructed signals using the parameters found at the best execution.

3 Experiments and Results

3.1 Experiment 1

The focus of this experiment is to compare the performances of two common EA and one SI algorithm for the optimization of the BSA parameters on a given signal. To achieve this, 30 independent executions were performed per compared algorithm using the first signal (Fig. 1a). The same ranges of the variables (Table 1) were acknowledged in each algorithm implementation.

In Table 3, the statistical analysis of SNR values (objective function) obtained in all executions and the Friedman test results (95%-confidence) are presented. Furthermore, in Table 4 the details of the algorithms median run are shown. Table 4a refers to the metrics, whereas Table 4b presents the parameters found. Finally, the convergence graphs of the three algorithms are presented in Figure 2a. Also, Fig. 2b includes the contrast between the original signal and the reconstructed signal by the three compared algorithms.

From these results, it is noticeable that DE/rand/1/bin outperformed both, GA and Global-best PSO. This is also validated by the Friedman test (Table 3). Furthermore, all parameters found are quite similar. On this regard, DE/rand/1/bin managed to get a better result using a lower filter size (Table 4).

Additionally, all algorithms showed a similar convergence dynamic (Fig. 2a): the exploration seems to decrease rapidly. Similarly, Fig. 2b shows that the reconstructed signals do not exhibit mayor differences among the algorithms implementations despite the variations observed by metrics.

Table 5. Detail of Wilcoxon rank-sum results based on SNR metric. **H=0** means no significant difference was found.

Statistic	GS	DE/rand/1/bin	Friedman test	
			p-value	**H**
Mean	9.1896	9.4786	0.2730	0
Std Dev	0.9367	0.9811		

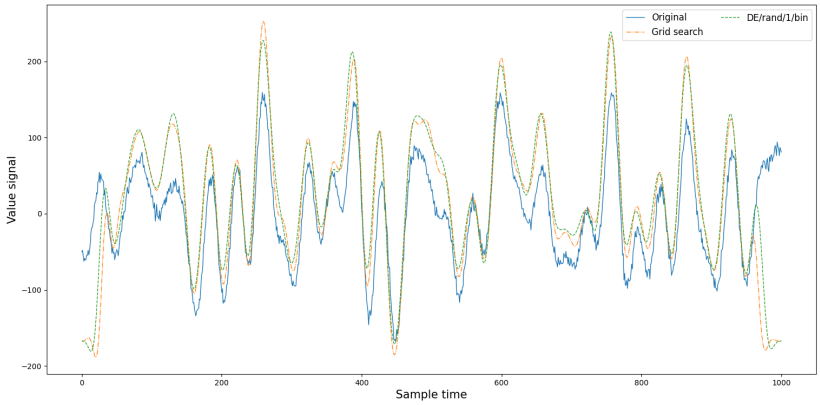


Fig. 3. Visual comparison between signal reconstruction from GS and DE for signal 2.

3.2 Experiment 2

The main goal of this experiment is to test the best performing bio-inspired algorithm out of the previous experiment against the implementation in [7] on ten different signals (Fig. 1b - 1k). Considering the fact that DE showed the best performance, it was selected for this experiment.

The parameters from the previous implementation were kept (Table 2b). In order to contrast the overall performance of GS and DE, Table 5 shows the statistical values obtained from SNR metrics of all signals, as well as the results of the Wilcoxon rank-sum test (95%-confidence).

Finally, a representative visual comparison of all tested methods on signal 2 (Fig. 1b) is presented in Fig. 3. From Table 5, it can be establish that no significant differences were found between GS and DE. This is also supported by the visual comparison highlighted in Fig. 3, where no mayor differences are visible.

4 Discussion

Firstly, experiment 1 shows that DE/rand/1/bin exhibited a better performance over the other algorithms tested (Table 3). Moreover, DE was able to lower the filter size variable the most. This is of great importance since BSA involves the deconvolution/convolution of a signal by the FIR filter. Hence, the smaller the filter size, the lower the cost and computational time.

Nevertheless, the quantitative improvements observed where not greatly reflected during visual comparison (Fig. 2b). Also, it seems that all bio-inspired algorithms converge fairly quick (Fig. 2a). Therefore, more efforts could be made in order to improve the explorations.

Finally, experiment 2 presents a direct contrast between GS and DE implementations applied to ten different signals. From there, the observations of GS and DE showed no significant differences overall (Table 5). This is also reaffirmed by the visual comparison performed between the reconstructed signal of GS and DE against the original signal. This evidence points to the ability of DE to match the performance of a GS with fixed ranges.

5 Conclusions and Future Work

SNNs models represent a paradigm shift from its predecessors; the key differentiation lies in how information is conveyed. Since SNNs use spike trains, a crucial question to be asked is: How can we translate analog signals to an impulse-based representation? In this paper, BSA was chosen as a method to transform analog signals to spike trains.

We tested two evolutionary algorithms and one swarm intelligence algorithm to optimize parameters for the already mentioned technique. It was found that DE/rand/1/bin performed better than its counterparts. Yet, results did not significantly improve the transformation capabilities.

On the other hand, the second experiment compared results between DE and the implementation in [7]. In such reference, a GS was used to find optimal parameters. Nevertheless, this method restricts the search space in order to lower the computational cost and time.

This also implies a certain prior knowledge in order to narrow the variables. In contrast, the DE implementation did not required any increment restriction. Having said that, our findings showed that DE could be considered as a optimizer of BSA parameters. Finally, future work directions could include different paths:

1. Perform experiments with more specialized, bio-inspired algorithms.
2. Implement a parameter tuning method in the algorithms calibration.
3. Evaluate the use of surrogate models for signal transformation.
4. Asses the possibility of a bio-inspired algorithm optimization using the classification performance of a generic SNN as objective function.

Acknowledgments. The first author acknowledges support from Consejo Nacional de Ciencia y Tecnología (CONACyT), Mexico through scholarship No. 1075919 to pursue graduate studies at University of Veracruz.

References

1. Maass, W.: Networks of spiking neurons: The Third Generation of Neural Network Models. Neural Networks, vol. 10, no. 9, pp. 1659–1671 (1997). DOI: 10.1016/s0893-6080(97)00011-7.

2. Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., Maida, A.: Deep Learning in Spiking Neural Networks. Neural Networks, vol. 111, pp. 47–63 (2019). DOI: 10.1016/j.neunet.2018.12.002.
3. Auge, D., Hille, J., Mueller, E., Knoll, A.: A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks. Neural Processing Letters, vol. 53, no. 6, pp. 1–18 (2021). DOI: 10.1007/s11063-021-10562-2.
4. Kasabov, N.: Evolving Spiking Neural Networks for Spatio-and Spectro-temporal Pattern Recognition. International Conference Intelligent Systems (2012). DOI: 10.1109/is.2012.6335110.
5. Dupeyroux, J., Stroobants, S., de-Croon, G.: A Toolbox for Neuromorphic Sensing in Robotics (2021) DOI: 10.1109/EBCCSP56922.2022.9845664.
6. Tan, C., Šarlija, M., Kasabov, N.: Spiking Neural Networks: Background, Recent Development and the neuCube Architecture. Neural Processing Letters, vol. 52, no. 2, pp. 1675–1701 (2020). DOI: 10.1007/s11063-020-10322-8.
7. Petro, B., Kasabov, N., Kiss, R. M.: Selection and Optimization of Temporal Spike Encoding Methods for Spiking Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 2, pp. 358–370 (2020). DOI: 10.1109/tnnls.2019.2906158.
8. Schrauwen, B., Van Campenhout, I.: BSA, a Fast and Accurate Spike Train Encoding Scheme. In: Proceedings of the International Joint Conference on Neural Networks, vol. 4, pp. 2825–2830 (2003). DOI: 10.1109/ijcnn.2003.1224019.
9. Moinnereau, M. A., Brienne, T., Brodeur, S., Rouat, J., Whittingstall, K., Plourde, E.: Classification of Auditory Stimuli from EEG Signals with a Regulated Recurrent Neural Network Reservoir (2018). DOI: 10.48550/ARXIV.1804.10322.
10. Armstrong, J. S., Forecasting, L. R.: From Crystal Ball to Computer. vol. 348 (1985)
11. Hyndman, R. J., Athanasopoulos, G.: Forecasting: Principles and Practice, o texts: Melbourne, 2nd edition (2018)
12. Sengupta, N., Kasabov, N.: Spike-time Encoding as a Data Compression Technique for Pattern Recognition of Temporal Data. Information Sciences, vol. 406–407, pp. 133–145 (2017). DOI: 10.1016/j.ins.2017.04.017.
13. Lampinen, J.: A Constraint Handling Approach for the Differential Evolution Algorithm. In: Proceedings of the Congress on Evolutionary Computation, vol 2, pp. 1468–1473 (2002). DOI: 10.1109/cec.2002.1004459.
14. Clerc, M.: Confinements and Biases in Particle Swarm Optimization. Technical Report hal-00122799 (2006)

ISSN 1870-4069

Sentiment Analysis Using Convolutional Neural Networks Generated by Neuroevolution

José Clemente Hernández-Hernández, Marcela Quiroz-Castellanos, Guillermo de Jesús Hoyos-Rivera , Efrén Mezura-Montes

Universidad de Veracruz,
Instituto de Investigaciones en Inteligencia Artificial,
México

{maquiroz,ghoyos,emezura}@uv.mx,
jclementehdzhdz@gmail.com

Abstract. Sentiment analysis is a sub-field of Natural Language Processing which is focused on determine what is the sentiment expressed in an opinion. In this paper we propose a new neuroevolution algorithm, called Deep NeuroEvolution of Weights and Topologies (DeepNEWT), which is based on a genetic algorithm and is used to evolve convolutional neural networks, to classify text in different polarity sentiments. The proposed algorithm, instead of using backpropagation on several epochs as training mechanism, as other proposals do, implements a plain mutation process adding random values to the current weights and bias. Moreover, the algorithm searches, through mutation and crossover operators, the best topology structure of the networks during a number of generations. This was executed using text data transformed with Word2Vec. The obtained results when varying the number of parents chosen for crossover and different mutation rates are encouraging.

Keywords: Neuroevolution, sentiment snalysis, evolutionary computing, neural networks.

1 Introduction

Sentiment Analysis (SA) is the field of study of the Natural Language Processing (NLP) which explores text data to detect the expressed sentiment [7]. There are several research works in SA, which focus on detecting the sentiment on text using Machine Learning (ML) [13], and, in recent years, due to the incorporation of Deep Learning (DL), Convolutional Neural Networks (CNN) have been used to improve the performance over the traditional ML classifiers [16].

Usually, the creation of a CNN architecture is handcrafted, but tuning them is not an easy task. For this reason, in this paper we propose to use Neuroevolution (NE), a technique that replaces the architecture engineering, doing this process automatically trough Evolutionary Computing (EC) algorithms [11]. The proposed approach is a Genetic Algorithm (GA), which combines interesting features of previous works in the field of Computer Vision (CV) [14].

Table 1. Similarity numbers and their corresponding non-linear activation function and pooling operations.

Similarity number	Non-linear function	Pooling operation
1	Sigm	Max
2	Sigm	Avg
3	Tanh	Max
4	Tanh	Avg
5	ReLU	Max
6	ReLU	Avg
7	PReLU	Max
8	PReLU	Avg

To search for CNNs, and the well known NeuroEvolution of Augmenting Topologies (NEAT) algorithm [12] to evolve fully-connected neural networks (FCNN). Our proposal involves crossover operators to share architecture elements between networks, and a mutation operator in a two-phase way, where new variations of the architectures can be inserted, and weights and bias are trained without using backpropagation.

The proposed experiments include variations in the parameters set to analyze the performance of the algorithm. Opinions used to test the algorithm are transformed using a Word2Vec [8] model. The rest of this paper is organized as follows: in Section 2 previous works about SA and NE are presented, while in Section 3 the proposed algorithm is described in detail. In Sections 4 and 5, the experimental design and the obtained results of the proposed algorithm are shown, respectively. Finally, in Section 6 conclusions and future work are drawn.

2 Related Work

CNNs have been used to automatically extract features from images, and to do different tasks as segmentation or classification [6]. In SA this kind of neural networks are used to classify transformed text, as in [5], where an experimental study was carried out to label movie reviews in different polarity sentiments, demonstrating that CNNs can be used to tackle the SA task.

The CNN created in the previous work, was used to classify tweets written in Spanish [9], where the text was transformed using Word2Vec model. CNNs were handcrafted created. Concerning NE and SA, FCNNs were also used to do SA in text. Using text written in Polish, an automatic system was created for pre-processing data and classify text using NEAT [10].

In [4], a new representation of text was created, and then, NEAT was used to create small versions of fully-connected neural networks to classify tweets written in Mexican Spanish. In these two researches, only FCNNs were evolved and used traditional techniques instead of DL approaches.

On the other hand, NE was also implemented to generate CNNs, which were used for SA. In [3], using a NEAT-based GA, CNNs were generated to classify movie reviews and in [1], based on a Differential Evolution algorithm, CNNs were evolved to classify text in Arabic.

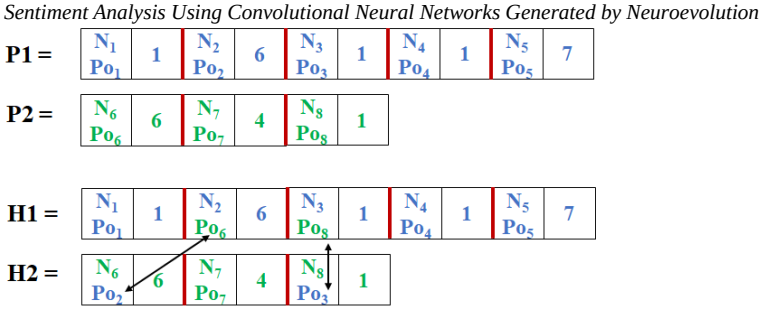


Fig. 1. Element crossover in the CNP blocks; each CNP block has its respective elements: a non-linear activation function $\mathbf{N_i}$, the pooling filter and its operation $\mathbf{Po_i}$, and the similarity number; they are divided by a red color line.

An interesting feature of the use of NE algorithms to evolve CNNs, is that for each generation or iteration, a step is executed to run a number of epochs a backpropagation algorithm. Other algorithms that were used, this time in the field of CV, are described in [2, 14, 15].

3 Deep NeuroEvolution of Weights and Topologies

Deep NeuroEvolution of Weights and Topologies (DeepNEWT), is a GA algorithm created to generate the architecture and weights of a CNN to classify text without using the backpropagation algorithm. Since DeepNEWT is a GA, some elements are introduced in its main process: (1) it uses a direct codification of potential solutions, (2) it allows sharing and mutating elements within solutions, (3) it admits the change of activation functions and pooling operations, (4) it searches layer similarities between solutions, (5) it trains connection weights and bias through a simple addition of random values without using backpropagation, and (6) it allows to use the CNN created in [5].

DeepNEWT uses a block-chained direct encoding, where convolutional, non-linear activation function, pooling, and fully-connected layers are reserved. A potential solution has three different blocks: (1) a convolutional, non-linear activation function, and pooling, called CNP, (2) a convolutional layer extracted from [5], called last-CNP layer, and (3) a fully-connected layer, called FC. DeepNEWT individuals are created randomly subject to certain parameters.

Each CNP block has $z \times z'$ convolutional filters and each filter has a length of $v \times w$, z is the number of input channels and z' is the number of feature maps or output channels. The CNP block also has a non-linear activation function, a pooling operation with its respective filter with dimensions $s \times t$ and a similarity number. A potential solution of a CNN can have n CNP blocks located one after another. A CNP block generates an output of $n'' \times m'' \times z'$ given an input of $n \times m \times z$.

Based on the historical markings in [12] and the crossover operator in [14], a novel element is introduced in this algorithm: the ease of sharing elements between solutions through similarities of the CNP blocks. A similarity number is given by the union of a non-linear activation function and a pooling operation. The similarity numbers in each combination of functions are shown in Table 1.

Table 2. Mutation processes with their corresponding sequential number (**S.**), and their moment; symbol +/- means that an element will reduce or increment its size or length.

	CNP		**last-CNP**		
Mutation	**Moment**	**S.** (1^{st}, 2^{nd})	**Moment**	**S.** (1^{st}, 2^{nd})	**Type**
No. of conv. Filters	-	-	Both	1	+/- 1
No. of output feature maps	Both	2	Both	2	+/- random
Activation function	Both	3, 6	Both	3, 6	Random
Conv. Filters length	Both	4, 7	Both	4, 7	+/- 1
W, b values	Both	5, 8	Both	5, 8	Random sum
No. of CNP blocks	2nd	3	-	-	+/- 1
Pooling operation	2nd	4	2nd	4	Random
Pooling filter length.	2nd	5	-	-	+/- 1

The so-called last-CNP layer, located after n CNP blocks, has different convolutional filters lengths of size $v_i \times w$, where $w = m''$, which is a value corresponding to an output generated by a CNP block or the input to the CNN. The last-CNP layer, given a convolutional filter f_i, generates an output of dimensions $n' \times 1 \times z'$. Later, a non-linear activation function is computed, and then a pooling operation is also executed with its respective pooling filter.

A pooling filter p_i in the last-CNP has dimensions $n' \times 1$, and the pooling operation is computed z' times over the convolutional operation output. The pooling operation generates an output of $1 \times 1 \times z'$ size. The final output of a last-CNP layer is a concatenated list of values of the pooling filters p_i.

The last element of the CNNs is a FC layer, located after the last-CNP. This layer has $|p_i|z'$ number of input neurons and, for this research work, 3 output neurons corresponding to the polarity sentiments: negative, neutral and positive. Before the algorithm executes crossover and mutation operators, a deterministic tournament is carried out in the current population P. It selects l individuals without replacement, in which, only the best individual becomes a part of a set P_s for crossover.

Such a process is done T times. After computing the DeepNEWT crossover operator, a set P_c is created with the recombined individuals. Three crossover processes are computed: (1) with CNP blocks, (2) in the last-CNP layer, and (3) in the FC layer. Crossover is applied to two parents, $P1$ and $P2$, to generate two offspring, $H1$ and $H2$. The similarity numbers of two parent solutions $P1$ and $P2$ are considered in the CNP blocks crossover process.

To compute it, first the same similarity numbers of the CNP blocks in both individuals must be identified. If there is more than one occurrence of the same similarity number in an individual, a CNP block is selected randomly. When the similarity numbers are detected, only the pooling filter and operation is changed with the other parent where the similarity number matches. The procedure of the CNP blocks crossover process is shown in Figure 1.

With respect to the last-CNP crossover operator, only the pooling operator, without the filter, is shared and, in the FC layer crossover, the non-linear activation function is transferred. A novel mechanism is introduced in the mutation operator, where two sequentially executed moments are involved.

Algorithm 1: Deep NeuroEvolution of Weights and Topologies

Data: Continue and discrete parameter limits
Result: Best CNN
Initialize N CNN in the population P;
Initialize ϕ_a and ϕ_b;
Compute CNN fitness;
while max generations not reached **do**
  $P_s \leftarrow$ select T elements from P by tournament;
  $P_c \leftarrow$ crossover elements from P_s ;
  $P_a \leftarrow$ mutate individuals from P_c with probability ϕ_a and M_a set;
  $P_b \leftarrow$ mutate individuals from P_a with probability ϕ_b and M_b set ;
  $P^{t+1} \leftarrow$ best individuals from $P^t \cup P_c \cup P_a \cup P_b$ are the population of the new generation;
end

Both moments have a set, M_a and M_b, respectively, of mutations that may be run with probability ϕ_a, for the first moment, and, with a probability ϕ_b, for the second moment, subject to $0 < \phi_b < \phi_a < 1$. The individuals from P_c set can be subject to modifications (1) at both moments, (2) only at the first moment, or (3) none of them. If an individual is modified at the first moment, the solution is now part of the P_a set, and such solutions can be modified in the second moment.

On the other hand, if a solution from P_a is modified at the second moment, the solution becomes part of the P_b set, with the rest of solutions modified at the second moment. A mutation modification has a sequential number, and it indicates when the modification will be executed. A moment is a set of mutations that modify solutions in a sequential way. A modification m_i from a set M, as mentioned above, has a probability ϕ of happening, depending on the execution moment.

If an individual is modified at m_i, the resulting individual from this modification can be modified again at the next mutation m_{i+1}, and so on. When an individual is modified at a mutation moment, the set P_a or P_b gets an individual after such moment. The modifications by mutation in the CNP blocks and last-CNP layer, including the weights W and bias b updating, are shown in Table 2. With respect to the FC layer, only the weights W, and bias b, can be modified by the mutation mechanisms.

It is important to mention that all modifications which include random variations or values, are carried out using an either, discrete or continue, random uniform distribution. Finally, the generation of a new population, P^{t+1}, is done by the union that involves the current population, P^t, the individuals after crossover, P_c, and the modified individuals in the first and second moments, P_a and P_b. After this union $P^t \cup P_c \cup P_a \cup P_b$, only the best individuals are selected to be part of the new population, P^{t+1}. The complete process of DeepNEWT is described in Algorithm 1.

4 Experimental Settings

Tweets used in this research work were manually labelled in three different polarity sentiments: positive, negative and neutral and all classes are balanced. The total of tweets is 150 and they belong to the Mexican political context. All tweets were transformed using a Word2Vec model trained from scratch. The model was trained with the Gensim 3.8 Python library[1].

[1] radimrehurek.com/gensim/

Table 3. Experiment parameters and their respective final accuracy results.

Exp.	T	ϕ_a	ϕ_b	**Accuracy**	**Exp.**	T	ϕ_a	ϕ_b	**Accuracy**	**Exp.**	T	ϕ_a	ϕ_b	**Accuracy**
A1	6	0.2	0.1	0.4733	B1	12	0.2	0.1	0.5067	C1	18	0.2	0.1	0.4867
A2	6	0.4	0.1	0.4933	B2	12	0.4	0.1	0.4733	C2	18	0.4	0.1	0.5133
A3	6	0.4	0.2	0.4667	B3	12	0.4	0.2	0.4733	C3	18	0.4	0.2	0.52
A4	6	0.6	0.1	0.48	B4	12	0.6	0.1	0.4667	C4	18	0.6	0.1	0.5067
A5	6	0.6	0.2	0.4667	B5	12	0.6	0.2	0.4867	C5	18	0.6	0.2	0.5067
A6	6	0.6	0.4	0.4933	B6	12	0.6	0.4	0.4533	C6	18	0.6	0.4	0.4733

Each word is given as input at the Word2Vec model, being the output a vector with continuous values. This vector has a dimensionality of 60 elements and all word vector were trained with C-BOW. Vector words of tweets are allocated in the center of a matrix according to the row axis. The matrix is padded with zeroes if necessary. A 60×60 matrix is the generated continuous representation of a tweet.

Different experiments were conducted to test the DeepNEWT performance for SA in tweets by using different number of selected parents for crossover T and different mutation probability values, ϕ_a and ϕ_b, for both moments, respectively. Selected values of the parameters are set to visualize the performance difference of the algorithm. A single run per configuration was carried out.

This is because of the computational cost required by each run (about 12 hours using 10,000 tweets). As default parameters, 100 generations and 20 individuals were set to run the experiments. The solutions in DeepNEWT have parameter limits that are necessary to consider. The number of CNP blocks is between $[0, 5]$, the number of output channels in CNP blocks and last-CNP layers is between $[10, 50]$ and the size of the convolutional and pooling filters is between $[3, 6]$. Fitness function of the algorithm is the accuracy of the data set transformed with Word2Vec.

5 Results and Discussion

Table 3 includes the obtained results with their associated parameters. Figure 2 has the convergence plot for each experiment (A, B and C), adding an average convergence plot. With respect to the achieved accuracy, in the experiments with 18 selected parents (C experiments), in 4 out of 6 experiments reached more than 50% accuracy; the highest obtained accuracy is 52% with $\phi_a = 40\%$ and $\phi_b = 20\%$.

All the experiments with 6 selected parents (A experiments), do not exceed 50% accuracy, but the average convergence plot (Figure 6) indicates that the experiments with 6 selected parents are better than those experiments with 12 selected parents (B experiments). The experiments with 12 selected parents have a slower convergence with respect to the shown by those experiments with 6 and 18 selected parents.

On the other hand, in the 18 selected parents experiments, the algorithm produces, with respect to the average convergence, better accuracy than all other experiments. In three cases (experiments C2, C3 and C5) the convergence was better so as to reach an accuracy higher than 50% after 80 generations.

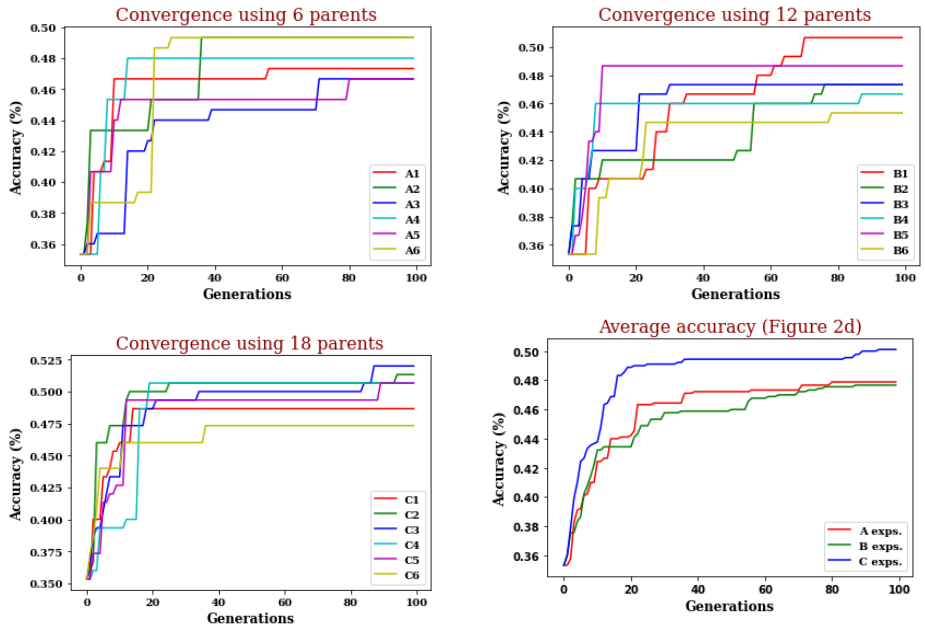


Fig. 2. Accuracy convergence with different number of selected parents.

As a conclusion of the experiments, DeepNEWT benefits the most when the number of selected parents for crossover increases, particularly with mutation values located at the middle of the ranges tested (i.e., 0.4 and 0.2 for each mutation considered).

6 Conclusions and Future Work

In this research work we proposed a NE algorithm based on a GA, that includes a mutation mechanism to update the weights and bias of CNNs without using backpropagation. The algorithm achieved an accuracy of 50%, using a database transformed with Word2Vec, for a number of generations, combining different numbers of selected parents and mutation probability values. The proposed algorithm particularly improved its performance with larger sets of parents selected for crossover and mutation values around 0.4 and 0.2.

The future work includes: (1) running more experiments with different number of selected parents and also with more probability values, (2) performing experiments without considering the $0 < \phi_b < \phi_a < 1$ condition, (3) considering the CNN parameter number besides accuracy, to also search for the simplest structure, (4) running experiments using another pre-processing technique such as BERT, and (5) implementing other kinds of mutation over the weights and bias.

Acknowledgments. The first author thanks the National Council of Science and Technology (CONACyT), for supporting him through a scholarship to carry out his MSc studies at Universidad Veracruzana.

References

1. Dahou, A., Elaziz, M. A., Zhou, J., Xiong, S.: Arabic Sentiment Classification Using Convolutional Neural Network and Differential Evolution Algorithm. Computational Intelligence and Neuroscience, vol. 2019, pp. 1–16 (2019). DOI: 10.1155/2019/2537689.
2. Desell, T.: Accelerating the Evolution of Convolutional Neural Networks with Node-level Mutations and Epigenetic Weight Initialization. In: Genetic and Evolutionary Computation Conference Companion, pp. 157–158 (2018). DOI: 10.1145/3205651.3205792.
3. Dufourq, E., Bassett, B. A.: EDEN: Evolutionary Deep Networks for Efficient Machine Learning. In: Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference, vol. 2018, pp. 110–115 (2017). DOI: 10.1109/RoboMech.2017.8261132.
4. Hernández, J. C. H., Montes, E. M., Hoyos-Rivera, G. J., Rodríguez-López, O.: Neuroevolution for Sentiment Analysis in Tweets Written in Mexican Spanish. In: Lecture Notes in Computer Science, pp. 101–110 (2021). DOI: 10.1007/978-3-030-77004-4 10.
5. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751 (2014). DOI: 10.3115/v1/d14-1181.
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature, vol. 521, no. 7553, pp. 436–444 (2015). DOI: 10.1038/nature14539.
7. Liu, B.: Sentiment Analysis and Opinion Mining (2012). DOI: 10.2200/S00416ED1V01Y201 204HLT016.
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations Ofwords and Phrases and Their Compositionality. Advances in Neural Information Processing Systems, pp. 1–9 (2013)
9. Paredes-Valverde, M. A., Colomo-Palacios, R., Salas-Zárate, M. D. P., Valencia-García, R.: Sentiment Analysis in Spanish for Improvement of Products and Services: A deep learning approach. Scientific Programming, vol. 2017 (2017). DOI: 10.1155/2017/1329281.
10. Sobkowicz, A.: Automatic Sentiment Analysis in Polish Language. Machine Intelligence and Big Data in Industry, pp. 3–10 (2016).
11. Stanley, K. O., Clune, J., Lehman, J., Miikkulainen, R.: Designing Neural Networks Through Neuroevolution. Nature Machine Intelligence, vol. 1, no. 1, pp. 24–35 (2019). DOI: 10.1038/s42256-018-0006-z.
12. Stanley, K. O., Miikkulainen, R.: Evolving Neural Networks Through Augmenting Topologies. Evolutionary Computation, vol. 10, no. 2, pp. 99–127 (2002). DOI: 10.1162/10636560232016 9811.
13. Sun, S., Luo, C., Chen, J.: A Review of Natural Language Processing Techniques for Opinion Mining Systems. Information Fusion, vol. 36, pp. 10–25 (2017). DOI: 10.1016/j.inffus.2016.1 0.004.
14. Sun, Y., Xue, B., Zhang, M., Yen, G. G.: Evolving Deep Convolutional Neural Networks for Image Classification. IEEE Transactions on Evolutionary Computation, vol. 24, no. 2, pp. 394–407 (2020). DOI: 10.1109/TEVC.2019.2916183.
15. Xie, L., Yuille, A.: Genetic CNN. In: IEEE International Conference on Computer Vision, pp. 1388–1397 (2017) doi: 10.1109/ICCV.2017.154.
16. Yadav, A., Vishwakarma, D. K.: Sentiment Analysis Using Deep Learning Architectures: A review. Artificial Intelligence Review, vol. 53, no. 6, pp. 4335–4385 (2020) doi: 10.1007/s10462-019-09794-5.

http://rcs.cic.ipn.mx

Centro de Investigación
en Computación