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.

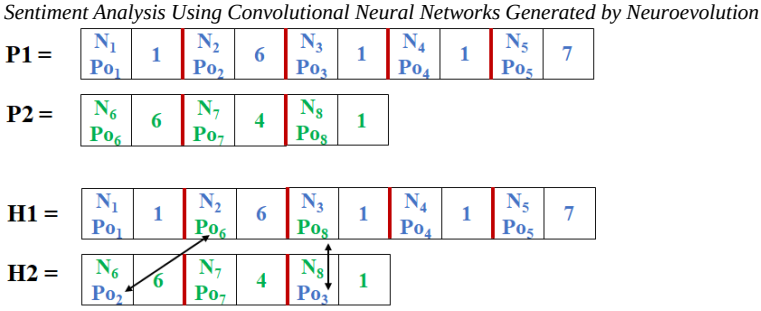

Fig. 1. Element crossover in the CNP blocks; each CNP block has its respective elements: a non-linear activation function $\mathbf{N_i}$, the pooling filter and its operation $\mathbf{Po_i}$, and the similarity number; they are divided by a red color line.

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.

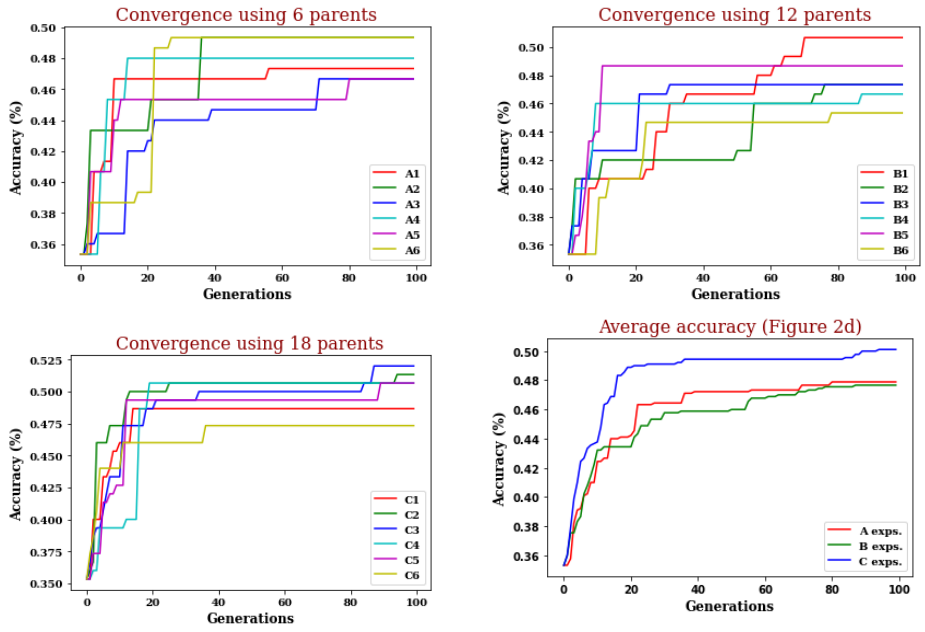

Fig. 2. Accuracy convergence with different number of selected parents.

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.