signals

MDPI

Article

Language Inference Using Elman Networks with Evolutionary Training

Nikolaos Anastasopoulos [1], Ioannis G. Tsoulos [2,*], Evangelos Dermatas [1] and Evangelos Karvounis [2]

[1] Department of Electrical and Computer Engineering, University of Patras, 26504 Rio, Greece
[2] Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece
* Correspondence: itsoulos@uoi.gr

Abstract: In this paper, a novel Elman-type recurrent neural network (RNN) is presented for the binary classification of arbitrary symbol sequences, and a novel training method, including both evolutionary and local search methods, is evaluated using sequence databases from a wide range of scientific areas. An efficient, publicly available, software tool is implemented in C++, accelerating significantly (more than 40 times) the RNN weights estimation process using both simd and multi-thread technology. The experimental results, in all databases, with the hybrid training method show improvements in a range of 2% to 25% compared with the standard genetic algorithm.

Keywords: recurrent neural networks; genetic algorithms; machine learning

Citation: Anastasopoulos, N.; Tsoulos, I.G.; Dermatas, E.; Karvounis E. Language Inference Using Elman Networks with Evolutionary Training. *Signals* **2022**, *3*, 611–619. https://doi.org/10.3390/signals3030037

Academic Editor: Hua-Liang Wei

Received: 30 April 2022
Accepted: 18 August 2022
Published: 6 September 2022

1. Introduction

Recurrent neural networks (RNN) are well-established mathematical models, and they have been used widely in many scientific and industrial areas such as text processing [1,2], speech recognition [3], sentiment classification [4], handwriting recognition [5], medical signal processing [6], etc. Elman [7] is a multi-layer neural network with recurrent units in the first layer. In its simplest form, the output of the Elman network is derived from the recurrent neuron outputs of the first layer. In the neuron internal states, the information from previous time non-linear transformations is stored and processed, also known as the memory of the cell. These models have been implemented in a wide variety of complex multidimensional classification problems [8,9]. Due to their complex structure, error propagation [10] and hill-climbing techniques [11,12] inherit many deficiencies. A well-known alternative, which avoids trapping the model in local minima, is the use of evolutionary methods such as the standard genetic algorithm (SGA), or some variants of the SGA [13], to estimate the internal weights or/and the structure of the Elman network [14,15], also referred to in the literature as the GA-Elman algorithm. Furthermore, the decoding of the model's output differs significantly from application to application. In natural language classification problems, it is common to use the final output of the network for determining the class of a data instance [16], while in more complex models, the output of the recurrent nodes [17] is part of a greater network. The use of evolutionary algorithms with Elman networks has been successful in several applications [18,19].

In this article, a novel approach to decoding the outputs of the Elman network for binary classification and a hybrid GA–Elman algorithm, which fully automatically estimates the synaptic weights, are presented. The input data of the network are treated as positive or negative symbolic sequences, while the output of the Elman network is decoded to a scalar value representing the probability of a data instance being recognized as a positive symbolic sequence by the classifier. The hybrid GA-Elman training algorithm estimates the weights of the model using both genetic operators and invoking gradient -ased local search [20,21] every few iterations. Even though this approach limits this classifier to binary classification problems, this method is still applicable to a wide range of problems, with

several advantages discussed in Section 5, and extended easily to multi-class problems. The datasets used, either artificially created or obtained from popular databases, are transformed to fit the encoding scheme of our method. All the experiments were conducted tenfold by randomly splitting the instances in test and evaluation data.

The rest of this article has the following structure: in Section 2, the proposed neural classifier and the corresponding training method are described in detail. The implementation details and the specific optimization techniques and tools are presented in Section 3. In Section 4, evaluation experiments from various scientific areas such as biology, chemistry, mathematics, etc., show the capabilities of the proposed algorithm. The accuracy of the proposed classifier and the computational advantages of the proposed classifier are summarized. In the last section, some conclusions from the use of the proposed method and implementation advances are given.

2. Method Description

A simple one-layer recurrent neural network (Elman [7]) transforms non-linearly each input symbol to an output symbol through a real-values vector mapping method using time-invariant data and memory-information-derived previous time non-linear transformations of the input sequence (also known as the memory of the cell).

In order to use an Elman-type recurrent neural network for symbol sequences classification, known also as artificial language modeling, several modifications to its original version are required.

The complete network and the binary classification process is separated in three distinguished parts: the data encoding-decoding process, the RNN activation function and the estimation of the output sequence mean log-probability. The binary classification criterion of the mean log-probability of the recurrent neural network outputs is a novel approach, as in the vast majority of implementations [8,9] the output vector at the T_{th} (final) step is interpreted as the output of the classifier.

2.1. Data Encoding–Decoding Process

The proposed method can be used to define binary classifiers of symbol sequences built on Elman recurrent neural networks using sets of positive and negative samples. Each individual symbol is mapped to a unique vector that consists of all zeros except the corresponding symbol index, which is set to one. Therefore, if the alphabet of the input sequences defined by the set $S = \{q_1, q_2, \ldots, q_M\}$, then the binary representation of the input symbols in the M dimensional space, where M is the number of distinct symbols, can be easily derived by $q_i \rightarrow \mathbf{x} = [0, \ldots, 0, 1, 0, \ldots 0]^T$, where the value 1 is met in the i_{th} position of the vector $\mathbf{x}$. Given a sequence of T symbols, the corresponding vectors in the RNN input is equal to $\mathbf{x}(t)$ for $t = 1 \ldots T$.

2.2. Elman Network

Given the input vector of the cell $\mathbf{x(t)}$ at time t and the previous hidden state of the memory cell $\mathbf{h(t-1)}$, represented by a N-dimensional vector, at time $t-1$, the new state of the cell $\mathbf{h(t)}$ is estimated by:

$$\mathbf{h}(t) = \mathbf{f}(\mathbf{W_{hh}} * \mathbf{h(t-1)} + \mathbf{W_{xh}} * \mathbf{x(t)} + \mathbf{b_h}) \quad (1)$$

where $f(.)$ is a non-linear activation function, $\mathbf{W_{hh}}$ are the hidden state synaptic weights, $\mathbf{W_{xh}}$ are the linear transformation weights of the input and $\mathbf{b_h}$ is the input bias factor. At time $t = 0, \mathbf{h}(0) = \mathbf{0}$.

The N-dimensional state $\mathbf{h(t)}$ of the recurrent network in time t is then converted to an M-dimensional vector $\mathbf{p(t)}$:

$$\mathbf{p}(t) = \sigma(\mathbf{W_{hp}} * \mathbf{h}(t) + \mathbf{b}_p), \quad (2)$$

where $\sigma(.)$ represent the softmax function, $\mathbf{W_{hp}}$ is the output synaptic weights and $\mathbf{b_p}$ is the output bias factor.

The softmax function is used to derive the symbols probabilities at the RNN output $\mathbf{p}(t)$, and is formally defined as:

$$p_i = \mathbf{p}_i = \sigma(\mathbf{x_i}) = \frac{e^{\mathbf{x_i}}}{\sum e^{\mathbf{x_i}}}, i = 1, M \tag{3}$$

It is well-known that the softmax [22] function is an efficient transformation function of a real-valued vector into a vector of probabilities due to the monotonicity of the exponential function into the set of positive real numbers and the normalization factor of their sum.

2.3. Estimation of Mean Log-Probability

The mean log-probability of the input sequence at the output of the recurrent cell is given by:

$$LogProb(\mathbf{x}) = \frac{1}{T} \sum_{t=0}^{T-1} log_2(q(t)) \tag{4}$$

In Equation (4), $q(t)$ represents the element of the output $\mathbf{p}(t)$ at index $q(t) = argmax(\mathbf{p}(t))$.

The input sequence is classified as a positive example if the mean log-probability is greater than a threshold. The optimal threshold is estimated during the training process.

2.4. Training Method

Typically, the Elman networks require an extremely long training time when the network weights are estimated from positive Ω^+ and negative Ω^- examples using the traditional error-backpropagation through time class of algorithms. Moreover, in the gradient vanishing problem, the convergence to local-minima of the error function is inevitable in these approaches. Therefore, a genetic algorithm is adopted due to the stochastic nature of this evolutionary search approach, the ability to perform the computations in parallel and the excellent behavior in the presence of large number of local optima of the optimized function.

The weights of the Elman network are derived by a modified genetic algorithm with local search for enhanced convergence. Local search is a procedure in which, at every T_l iterations of the genetic algorithm, N_t chromosomes are randomly selected and optimized through a non-evolutionary method. In the experiments conducted in this paper, the ADAM method was selected as local search optimizer.

The weights of the Elman network can be encoded in different depths of accuracy, i.e. double, single precision floating or fixed point numbers. In typical recurrent structures, the accuracy of an objective function is more sensitive with respect to the recurrent part of the structure, in our case the W_{hh} weights.

The proposed training method is presented in the pseudo-code of Algorithm 1. The fitness of each chromosome is estimated from the log-probabilities of the RNN output for all positive and negative available examples. The overall fitness criterion decreases the log-probability for the negative examples, also increasing the log-probability for the positive examples:

$$F(\Omega^+ \cup \Omega^-) = \sum_{\mathbf{x} \in \Omega^+} LogProb(\mathbf{x}) - \sum_{\mathbf{x} \in \Omega^-} LogProb(\mathbf{x}) \tag{5}$$

The goal of the training method is to maximize the fitness metric of Equation (5).

Algorithm 1 Training Elman-Classifiers

1. (a) Set $i = 0$ (iteration counter) and *MaxIters* (maximum number of iterations);
   (b) Set the p, p_m, p_s the population size, mutation rate and selection rate of the algorithm;
   (c) Set T_l, N_l the frequency and number of chromosomes for local optimization;
   (d) Randomly initialize each chromosome in $[0, 1]^n$.
2. (a) ***For each*** *individual in population calculate its fitness*
   (b) $i = i + 1$
3. (a) Select the best p_s individuals of the population;
   (b) Apply one-point crossover to the selected individuals to derive the new population;
   (c) Randomly mutate chromosomes in respect to p_m.
4. ***If*** i mod $T_l == 0$: apply local optimization to N_l random chromosomes
5. ***If*** $i > MaxIters$ **:**
   (a) Calculate the optimal threshold where the minimum classification error occurs;
   (b) *Return the best chromosome and the optimal threshold*.

   else : goto 2.

The optimal threshold is estimated by scanning the log-probability space and detecting the value where the classification error of the trained Elman RNN classifier is minimum for the training set of both positive and negative examples Ω^+, Ω^-.

Randomly selected chromosomes are optimized for a small number of iterations through the selected local optimization method with respect to the fitness metric.

For the local optimization method, the ADAM [23] optimizer was selected as a lightweight but fast converging algorithm. The improvements in the convergence of the model in all experiments are presented in Table 1.

Table 1. Experimental results on selected datasets.

DATASET	GA-Elman	GA-Elman + ADAM
Regex	78.42%	87.75%
Primes	62.80%	68.50%
Flu	69.00%	75.25%
LogP	64.13%	71.63%
Sentiment	53.83%	54.01%
Mnist	85.63%	87.38%
Dnacvs	61.78%	66.50%
Prop	66.36%	80.53%

3. Implementation

The software developed for the algorithm evaluation, the experiments and the corresponding data are freely distributed in itsoulos.teiep.gr/GeneticRNN.tar.gz (accessed on 29 August 2022).

The proposed classifier and the training method have several benefits in regards to their complexity and their performance on any modern computing system. Modern CPUs mainly offer two types of parallelism: SIMD instructions and concurrent execution of parallel sections. These capabilities are covered by the OpenMP standard that is implemented in the most popular C/C++ compilers [24,25]. The use of these capabilities are quantified in Table 2.

Table 2. Comparison between implementations.

DATASET	Non-Optimzed	2 Threads + SIMD	8 Threads + SIMD
Regex	205 min.	42 min. (×4.88)	19.4 min. (×10.56)
Primes	162 min.	31 min. (×5.22)	14.4 min. (×11.25)
Prop	111 min.	21.2 min. (×5.23)	10.5 min. (×10.57)

3.1. Hardware Acceleration

The recurrent neural network was implemented explicitly using SIMD instructions through the OpenMP API [26] as it yielded better results than the automatic vectorization capabilities in the case of the GCC compiler that was used. The steps 2 and 4 of the training algorithm, described in Section 2.4, were implemented by using the multi-process capabilities of the OpenMP standard. It is important to mention that, while multi-threading programming can also be used for the Elman classifier, it is best to minimize the number of threads created, as the accumulated overhead may reduce the acceleration rate significantly. Moreover, the fitness estimation procedure is completed in almost constant time in our method, so the static scheduling capabilities [27] of the OpenMP API further minimize overheads by spawning threads for chunks of estimations rather than estimating the fitness of each single chromosome in a single thread.

3.2. Software Techniques

In order to obtain the gradient of the objective function (Equation (5)) with respect to the corresponding data or a batch of data for the ADAM optimizer, the differentiation was implemented with the automatic differentiation (AD) method of the Adept library [28]. This approach enables exact estimation of the gradient for any continuous and smooth function, as in the case of our objective function or any alternative non-linear function adopted by the user. Furthermore, this is performed by template programming in C++, which enables the compiler to generate code at compile time, which enables even more optimization. The results of the previously mentioned techniques are quantified in Table 2, whereas the partially optimized code (-O2 default flag in GCC) and optimized code with vector processing and multi-threading for 2 and 8 thread run times are presented.

4. Experiments

4.1. Datasets

A series of datasets from different fields were used as test cases for the evaluation of the proposed classifier and the corresponding training method. In Table 3, the number of positive/negative sequences, the number of distinct symbols in the dataset, the average length of each instance and the standard deviation are summarized.

Regex Data. Two artificial regular expressions RE_1, RE_2, where $RE_1 \cap RE_2 = \emptyset$ and

- $RE_1 = (a|b|c|d|e|f)\{6,7\}(a|b|c)\{2,3\}(a|b|c|d|e|f)\{6,7\}(a|b|c)\{1,3\}$
- $RE_2 = (a|b|c|d|e|f)\{6,7\}(d|e|f)\{2,3\}(a|b|c|d|e|f)\{6,7\}(d|e|f)\{1,3\}$

where $(a|b|c|d|e|f)\{6,7\}$ is a sequence of six to seven characters from a–f [29] and were used to measure both classification accuracy and encoding efficiency in the well control environment of simple artificial languages with a small number of terminal symbols.

A total number of 300 positive and 300 negative instances were randomly created from the above regular expressions as positive and negative data. The intersection of the positive and negative data sequences is an empty set.

Biological datasets. Two additional biological datasets with only four terminal symbols (A,G,C and T) were sampled from the well-known genome sequences NCBI and [30], called Flu and Dnacvs, as shown in the experimental result Table 1.

The Flu database consists of 200 sequences of DNA from the flu virus strains and prion strains randomly selected from the NCBI dataset [31], aiming to successfully distinguish if an arbitrary length genome sequence represents a virus (positive sequences) or a prion (negative sequences).

The Dnacvs dataset consists of genomes that represent either coding (positive sequences) or non-coding genomic DNA sequences (negative sequences) [30]. A total number of 1000 positive sequences and 1000 negative sequences were randomly sampled, and the goal of the classifier was to distinguish, given an arbitrary DNA sequence, if this sequence can be translated to a protein or not.

Chemical dataset. The LogP dataset was used to predict if a chemical substance has positive [32] or negative LogP [33] value using only the molecule in SMILES [34] representation. If the LogP of a molecule instance is positive, the instance is considered a positive example, else the molecule is considered a negative example. The database consists of 3000 positive and 3000 negative instances, while there are 32 distinct terminal symbols in the dataset.

Image dataset. In this dataset, two hand-written numbers (0, 8) from the MNIST dataset were selected to represent the two classes of the classifier: positive and negative examples. These numbers were selected because they are similar enough to pose a difficult problem for the proposed method. Each instance is a sequence of 784 (28×28 pixels) binary numbers representing the hand-written numbers. A total of 1000 positive and 1000 negative examples were used.

The Sentiment dataset. The dataset [35] was used to build an Elman classifier capable of distinguishing if a sentence describes a movie in a positive or negative mood. The database consists of 1480 positive sequences and 1302 negative sequences. There are 28 terminal symbols in this database representing the English letters and some punctuation symbols.

Mathematical datasets. The Prop dataset represents sequences of decimal numbers for positive data and random sequences of decimal numbers as negative data. The positive sequences of digits were generated by taking a positive integer number b and then by successively multiplying this number by $1, \ldots, 10$ and concatenating the products. The numbers b were in the range $[10, 100]$ for the positive sequences.

The Primes dataset consists of 500 random primes starting from 101 and 500 random not-prime integers starting from 100, both in binary form. While the prime identification problem has been up to now unsolvable by a finite rules grammar, it is well-known that there are several noisy patterns in the binary representation of primes. The experiments carried out with the Primes dataset explore the capabilities of the RNN training method to detect primes from their binary representation.

Table 3. Datasets summary.

DATASET	Positive	Negative	Symbols	Av. Length	STDEV Length
Regex	300	300	6	17	1.5
Primes	500	500	2	12	1.2
Flu	200	200	4	147	13.6
LogP	3000	3000	32	12	8.6
Sentiment	1480	1302	28	86	23.5
Mnist	1000	1000	2	784	0.0
Dnacvs	1000	1000	4	55	0.0
Prop	90	90	10	22	6.3

4.2. Experimental Results

In Table 1, the experimental results derived by the proposed training method using the evaluation datasets are presented. Each row contains the dataset name, the RNN classification accuracy when the training process is completed with the aid of the local search optimization procedures, or using only the genetic learning part of the algorithm. Each experiment was conducted ten times with a different seed for the random generator each time, and the average rates are presented, minimizing the influence of the initial random selection and the random processes involved in the genetic learning algorithm. The values presented in Table 4 were used for each experiment. The accuracy gain from the

use of local optimization is 2–25% depending on the dataset. The size of the memory of the neuron N in all experiments was set to 10, which means that the number of independent variables in the proposed classifier is extremely small. For $N = 10$, the total number of synaptic weights, depending on the number of independent symbols in each experiment, varies from 150 to 250 double precision numbers.

For the MNIST dastaset especially, there have been several methods tested [36,37] using Elman networks. The method proposed in [36], for example, proposes a more complicated inference method and barely outperforms our method for similar tasks in number classification. Our method outperforms the method proposed in [37], but the scope of our dataset is limited (similarly to [36]) in comparison to the test conducted in [37].

Table 4. Algorithm parameters .

Parameter	Value
Memory size ($size(h)$)	10
Chromosomes	500
Generations	300
Elitism	1 chromosome
Selection (p_s)	0.9
Mutation (p_m)	0.05
Local Chromosomes (N_t)	4
Local Frequency (T_l)	20

5. Conclusions

A novel Recurrent Neural Network classifier trained by combined evolutionary and gradient-based techniques is described in this article and implemented. The information combining with the log-prob decoding from the iterative outputs of the classical Elman network shows promising potential in a great variety of binary classification problems. Furthermore, the computational overhead from the log-prob decoding is minimal, as this method does not introduce complex structures. It is also shown that the proposed classifier and the corresponding training method can be applied in different scientific areas including chemical, image, sentiment, mathematical and biological problems. Moreover, the use of local search methods in the training process, combined with an evolutionary learning method, outperforms in all cases the accuracy of the typical GA-Elman algorithm. It is notable that despite the small computational complexity of the model, the classification rates are robust, even in the case of severe scientific problems.

The limitation of this method is scalability in complex models. This limitation is present in the vast majority of evolutionary methods as each candidate solution must be evaluated on the whole dataset. So, complex models, where the evaluation process is computationally expensive, will need extensive training time.

Future Plans

This method can be extended in order to fit several types of recurrent neural networks. We would like to implement this method in LSTMs, GRUs and the corresponding variants of these architectures. This method can be used without changing anything in the majority of neural-network architectures.

Author Contributions: N.A., I.G.T., E.D. and E.K. conceived of the idea and methodology. N.A. and I.G.T. conducted the experiments, employing several datasets and provided the comparative experiments. E.D. and E.K. performed the statistical analysis and all other authors prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The experiments of this research work were performed at the high-performance computing system established at the Knowledge and Intelligent Computing Laboratory, Dept of Informatics and Telecommunications, University of Ioannina, acquired with the project "Educational Laboratory equipment of TEI of Epirus" with MIS 5007094 funded by the Operational Programme "Epirus" 2014–2020, by ERDF and national finds.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mikolov, T.; Karafiat, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the INTERSPEECH-2010, Chiba, Japan, 26–30 September 2010; pp. 1045–1048.
2. Farkas, J. Towards classifying full-text using recurrent neural networks. In Proceedings of the 1995 Canadian Conference on Electrical and Computer Engineering, Montreal, QC, Canada, 5–8 September 1995; pp. 511–514.
3. Graves, A.; Mohamed, A.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
4. Tang, D.; Qin, B.; Liu, T. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015.
5. Graves, A.; Schmidhuber, J. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In Proceedings of the NIPS'08: 21st International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–11 December 2008; pp. 545–552.
6. Guler, N.F.; Ubeyli, E.D.; Guler, I. Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Syst. Appl.* **2005**, *29*, 506–514. [CrossRef]
7. Liou, C.-Y.; Huang, J.-C.; Yang, W.-C. Modeling word perception using the Elman network. *Neurocomputing* **2008**, *71*, 3150–3157. [CrossRef]
8. Liu, P.; Qiu, X.; Huang, X. Recurrent neural network for text classification with multi-task learning. *arXiv* **2016**, arXiv:1605.05101.
9. Arras, L.; Montavon, G.; Müller, K.R.; Samek, W. Explaining recurrent neural network predictions in sentiment analysis. *arXiv* **2017**, arXiv:1706.07206.
10. Gruslys, A.; Munos, R.; Danihelka, I.; Lanctot, M.; Graves, A. Memory-Efficient Backpropagation Through Time. In Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, 5–10 December 2016.
11. Chopra, S.; Auli, M.; Rush, A.M. Abstractive sentence summarization with attentive recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 14–17 June 2016.
12. Jagannatha, A.N.; Yu, H. Bidirectional RNN for medical event detection in electronic health records. In *Proceedings of the Conference, Association for Computational Linguistics, North American Chapter, Meeting*; NIH Public Access: Bethesda, MD, USA, 2016; Volume 2016.
13. Chandra, R. Competition and Collaboration in Cooperative Coevolution of Elman Recurrent Neural Networks for Time-Series Prediction. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 3123–3136. [CrossRef]
14. Jia, W.; Zhao, D.; Zheng, Y.; Hou, S. A novel optimized GA-Elman neural network algorithm. *Neural Comput. Appl.* **2019**, *31*, 449–459. [CrossRef]
15. Ding, S.; Zhang, Y.; Chen, J.; Jia, W. Research on using genetic algorithms to optimize Elman neural networks. *Neural Comput. Appl.* **2013**, *23*, 293–297. [CrossRef]
16. Yin, W.; Kann, K.; Yu, M.; Schütze, H. Comparative Study of CNN and RNN for Natural Language Processingm, guidance for DNN selection. *arXiv* **2017**, arXiv:1702.01923.
17. Kuen, J.; Wang, Z.; Wang, G. Recurrent Attentional Networks for Saliency Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
18. Liu, H.; Geng, X. Application of GA-DA-Elman Neural Network Algorithm to Urban Air Quality Evaluation. In *IOP Conference Series: Materials Science and Engineering*; No. 5; IOP Publishing: Bristol, UK, 2020; Volume 768.
19. Wang, X.; Li, C. Prediction Model of MBR Membrane Flux for Elman Neural Network Based on PSO-GA Hybrid Algorithm. In Proceedings of the 2018 Eighth International Conference on Instrumentation & Measurement, Computer Communication and Control (IMCCC), Harbin, China, 19–21 July 2018.
20. Dengiz, B.; Altiparmak, F.; Smith, A.E. Local search genetic algorithm for optimal design of reliable networks. *IEEE Trans. Evol. Comput.* **1997**, *1*, 179–188. [CrossRef]
21. El-Mihoub, T.A.; Hopgood, A.A.; Nolle, L.; Battersby, A. Hybrid Genetic Algorithms: A Review. *Eng. Lett.* **2006**, *13*, 124–137.
22. Shibata, K.; Ito, K. Gauss-Sigmoid neural network. In Proceedings of the IJCNN'99, International Joint Conference on Neural Networks, Proceedings (Cat. No. 99CH36339), Washington, DC, USA, 10–16 July 1999; Volume 2.
23. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
24. Novillo, D. OpenMP and automatic parallelization in GCC. In Proceedings of the GCC Developers Summit, Ottawa, ON, Canada, 28–30 June 2006; p. 47.

25. Antao, S.F.; Bataev, A.; Jacob, A.C.; Bercea, G.T.; Eichenberger, A.E.; Rokos, G.; Martineau, M.; Jin, T.; Ozen, G.; Sura, Z.; et al. Offloading support for OpenMP in Clang and LLVM. In Proceedings of the 2016 Third Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC), Salt Lake City, UT, USA, 14 November 2016.
26. Chandra, R.; Dagum, L.; Kohr, D.; Maydan, D.; McDonald, J.; Menon, R. *Parallel Programming in OpenMP*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001.
27. Bull, J.M. Measuring synchronization and scheduling overheads in OpenMP. In Proceedings of the First European Workshop on OpenMP, Lund, Sweden, 30 September–1 October 1999; Volume 8.
28. Hogan, R.J. Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Trans. Math. Softw.* **2014**, *40*, 16. [CrossRef]
29. Aho, A.V.; van Leeuwen, J. (Eds.) Algorithms for finding patterns in strings. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*; The MIT Press: Cambridge, MA, USA, 1990; pp. 255–300.
30. Yin, C.; Yau, S.S.T. Prediction of protein coding regions by the 3-base periodicity analysis of a DNA sequence. *J. Theor. Biol.* **2007**, *247*, 687–694. [CrossRef] [PubMed]
31. NCBI Resource Coordinators. Database resources of the national center for biotechnology information. *Nucleic Acids Res.* **2017**, *45*, D12. [CrossRef] [PubMed]
32. Takacsne, N.K. Computerized logP prediction using fragment methods. *Acta Pharm. Hung.* **1998**, *68*, 39–48. [PubMed]
33. Zhong, T.; Hao, Y.L.; Yao, X.; Zhang, S.; Duan, X.C.; Yin, Y.F.; Xu, M.Q.; Guo, Y.; Li, Z.T.; Zheng, X.C.; et al. Effect of XlogP and Hansen solubility parameters on small molecule modified paclitaxel anticancer drug conjugates self-assembled into nanoparticles. *Bioconj. Chem.* **2018**, *29*, 437–444. [CrossRef] [PubMed]
34. Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36. [CrossRef]
35. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013.
36. Bausch, J. Recurrent quantum neural networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1368–1379.
37. Manchev, N.; Spratling, M.W. Target Propagation in Recurrent Neural Networks. *J. Mach. Learn. Res.* **2020**, *21*, 1–33.