Household Energy Consumption Prediction: A Deep Neuroevolution Approach

Alexander Soudaei
Dept. of Computer Science, Oslo Metropolitan University, Oslo, Norway
asoudaei@gmail.com

Jianhua Zhang
Dept. of Computer Science, Oslo Metropolitan University, Oslo, Norway
jianhuaz@oslomet.no

Mohamed Elmi
Dept. of Mechanical, Electronic and Chemical Eng., Oslo Metropolitan University, Oslo, Norway
s341927@oslomet.no

Mikael Tsechoev
Dept. of Computer Science, Oslo Metropolitan University, Oslo, Norway
mikael.tse88@live.no

Zishan Khan
Dept. of Mechanical, Electronic and Chemical Eng., Oslo Metropolitan University, Oslo, Norway
mo-elmi1998@hotmail.com

Ahmed Osman
Dept. of Mechanical, Electronic and Chemical Eng., Oslo Metropolitan University, Oslo, Norway
a.ozman723@gmail.com

ABSTRACT

Accurate energy consumption prediction can provide insights to make better informed decisions on energy purchase and generation. It also can prevent overloading and make it possible to store energy more efficiently. In this work, we propose a new deep learning model to predict the household energy consumption. In the new model, we employ differential evolution (DE) algorithm to automatically determine the optimal architecture of the deep neural network. The energy prediction results are presented and analyzed to show the effectiveness of the deep neuroevolution model constructed.

CCS CONCEPTS

• **: Computing methodologies** → Neural networks; Computing methodologies; Supervised learning by regression; Applied computing; computer-aided design.

KEYWORDS

Deep learning, Neural networks, Neuroevolution, Differential evolution, Evolutionary algorithm, Energy consumption prediction

ACM Reference Format:
Alexander Soudaei, Jianhua Zhang, Mohamed Elmi, Mikael Tsechoev, Zishan Khan, and Ahmed Osman. 2023. Household Energy Consumption Prediction: A Deep Neuroevolution Approach. In *2023 3rd International Conference on Artificial Intelligence, Automation and Algorithms (AI2A '23), July 21–23, 2023, Beijing, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3611450.3611474

1 INTRODUCTION

In this paper, we present a new approach to predict the household energy use, where we employ differential evolution (DE) algorithm to optimize the architecture and parameters of a deep learning (DL) model.

AI2A '23, July 21–23, 2023, Beijing, China

ACM ISBN 979-8-4007-0760-5/23/07.
https://doi.org/10.1145/3611450.3611474

The DE was inspired by the evolutionary aspects of living organisms [1-5]. The biological plausibility makes the DE a competitive candidate for selecting the optimal architecture of a deep neural network, since it provides remarkable approximate solutions that cannot be easily obtained by using other algorithms [6]. Approximate solutions, provide by the evolutionary algorithms or swarm intelligence algorithms [7] in the field of computational intelligence [8], are a good way to mitigate the computational overhead for developing DL models with appropriate architectures.

Reference [9] presented a method using DE to optimize artificial neural network (ANN) for classification of parity-p problem. The results were compared to those of the Levenberg-Marquardt (LM) and backpropagation algorithms. It was shown that the DE-ANN approach had a higher percentage of correct classification in four of ten cases than the LM algorithm. In [10], the authors presented four different approaches to energy prediction in home appliances, namely Multiple linear regression, Gradient Boosting Machines (GBM), Support Vector Machine (SVM) with radial kernel, and Random Forest. It was shown that GBM has achieved the best results in terms of the modeling accuracy metrics of RMSE and R^2. These four algorithms proved to have certain limitations in their modeling abilities.

In [11], Njock *et al.* presented an ANN optimized by the DE algorithm for predicting diameters of jet grouted columns in 2021. Though the results reported in the paper were shown certain improvement, unfortunately the authors only use R^2 as the metric for accuracy. Reference [12] used different ways of presenting and validating the modeling results.

In fact, different types of ANNs can be combined with DE or other evolutionary algorithms (EAs) (see [13-15]) to make predictions. In [16], long short-term memory (LSTM) system was utilized for short-range prediction of electricity load and price. On the other hand, other EAs have also been integrated with an ANN model. In [17], Han and others employed genetic algorithm (GA) to optimize hyperparameters of Convolutional Neural Network (CNN). The algorithm showed reduced CNN training time and encouraging modeling results.

In the present work, we try to combine DE and ANN for regression-model-based energy prediction. We use four standard

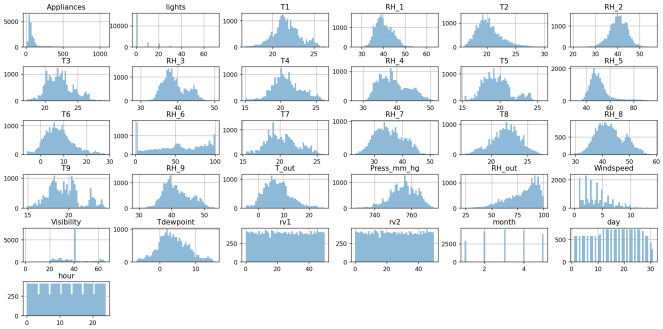

Figure 1: The histograms of all the 31 variables (30 inputs and one output) in the dataset.

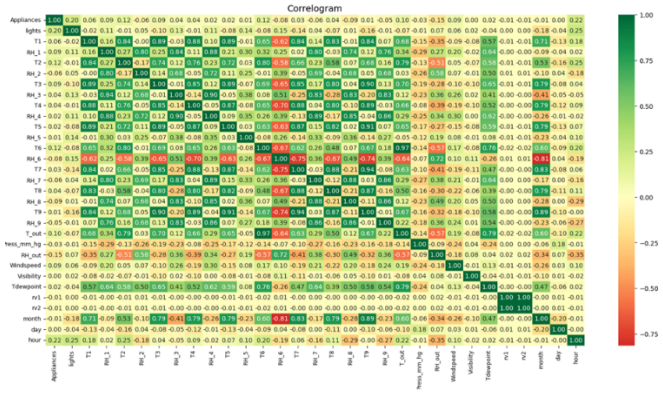

Figure 2: Correlation matrix between the inputs and output variables.

metrics, i.e., RMSE, MAE and MAPE, and R^2, to evaluate and compare, in a more comprehensive manner, the accuracy of our proposed model with several state-of-the-art methods in literature.

2 PROBLEM STATEMENT AND DATASET

We consider a time series model with 28 inputs (humidity and temperatures of 9 rooms in a house with lower energy use, wind speed, pressure, dew point, among others) and a single output (the energy consumption rate of appliances in the house in Watts per hour (Wh)).

The data was measured from Chievres airport station using a ZigBee Wireless Sensor Network (WSN). The available dataset contains 19736 instances, measured every 10 min for a period of about 4.5 months (from Jan. 11 to May 27, 2016). There are no missing values in this dataset. For more details about the dataset, the interested readers are referred to [10].

As the dataset is a time series, we divide the date of each data point into three separate features: *Day*, *Month*, and *Time*. After this conversion we now have a total of 30 input features in our dataset. Another typical problem to deal with are outliers. Fig. 1 shows the distribution of the model I/O variables in the dataset.

From the distributions, we can see that most of the variables are normally distributed while a few of them stand out. The prediction variable, energy consumption rate of appliances, seems to be slightly right-skewed. Lights, month, day and hour also stands out here, as they are discrete variables which only take a number of discrete values.

Detecting and removing outliers is a common practice in machine learning, as this allows us focus on the *regular* values.

We remove these outliers (extreme values) as we prefer a higher accuracy on values that we commonly see to a decent accuracy on both the common and extreme values.

To achieve this, we use the common 3σ method, i.e., the instances whose output (appliance energy consumption rate) is out of the range $mean \pm 3\sigma$ will be removed before further processing. Using this outlier detection and removal method, 540 instances were removed from the original dataset and 19195 instances are left.

Fig. 2 shows the correlation between the different I/O variables. We can observe that the four inputs which have the highest correlation to the model output (appliance energy consumption) are *Lights*, *T2*, and *T6* (all with a positive correlation to the output), and *RH_out* (with a negative correlation to the output).

Before splitting, we shuffle the dataset, i.e., we change the order of data instances to make sure that the model would be created with reliable training and testing sets. Shuffling makes the model training resistant to memorizing the patterns of the dataset. It also helps with faster convergence which is convenient when working with a large number of data instances. Then we split the dataset into a training set and a testing set to evaluate its accuracy on a disjoint testing set. The training set is used for training the model, and the test set is used for testing the accuracy of the model. The validation set may also be created to make sure that the model does not overfit the training data, by stopping the training process if the training error keeps decreasing while the validation error keeps increasing. To train our model, we allocate 60% of the original dataset as a training set, 15% as validation set, and the rest 25% as a testing set.

3 DEEP NEUROEVOLUTION APPROACH

The differential evolution (DE) algorithm is a metaheuristic, stochastic population-based direct search method [18]. One major advantage of DE algorithm is that it has fast convergence to the global minimum. DE uses the GA-like evolutionary operators (e.g., crossover and mutation) to alter the behavior of the swarm. In [4], Bilal *et al.* presented the evolution of the DE algorithm since 1995. In [18], mutation is referred to as the generation of a new position in the search space from three vector donors. The mutation is performed by:

$$v = v_1 + F\,(v_2 - v_3) \tag{1}$$

where the amplification parameter $F \in [0, 2]$ is a crucial parameter for controlling the balance between exploration and exploitation. Increasing F would increases the exploration in the swarm, causing the particles to spend more time finding the solution, while decreasing it would increase the exploitation in the swarm. This can prevent the premature convergence of the particles in the DE algorithm [5].

After mutation, the crossover operator is applied to create candidate vector $\boldsymbol{u}$ with its i-th component:

$$u_i = \begin{cases} v_i, \text{if} r < CR \text{or} i = I \\ x_i, \text{otherwise} \end{cases} \quad (2)$$

where r and I are random numbers in the interval [0, 1] and [0, n], respectively, CR stands for the crossover rate (or probability) larger than the random number r [0,1], and x is the position of a particle.

If CR is set to a low percentage, then it would cause a low amount of breeding between particles; otherwise it would cause high amount of breeding between them. The variable I makes sure that at least one component in the candidate vector u comes from the mutation vector v. Using both mutation and crossover operations we reach the goal of finding candidate vector u. The last step is selection where the candidate vector is decided if it will be included in the upcoming generations. This is done by comparing to the current particle position x_i using the greedy criterion [3]. This criterion makes sure that a new parameter vector is selected and accepted into the new generation only if it reduces the objective function value.

Both crossover and mutation must be carried out to generate a new position of the particle. This process eliminates a higher degree of random position generation, which has been used by many other optimization algorithms.

DE algorithm performs well with combinations of various swarm sizes and iterations and may find the global minimum in most applications. This versatility makes it a widely utilized swarm intelligence (SI) optimization algorithm. Application areas of DE ranges from solution of simple mathematical problems to solution of nuclear engineering optimization problem [2].

In this work, we combine DE, stochastic gradient descent (SGD) and Adadelta optimization algorithms to optimize the architecture and weight parameters of a multi-layer feedforward ANN model. The DE is used to optimize the number of nodes in the four hidden layers of the ANN.

3.1 Encoding Scheme of the DE Algorithm

We represent ANN architecture as a position vector, where the i-th component represents the number of nodes in the hidden layer #i in the network. The units in the input and output layers are not included, as they are predetermined and kept constant. For example, the architecture of a 3-layer network with 51 and 33 nodes in the 1st and 2nd hidden layer, respectively, would be represented as a position (row) vector [51 33]. The positions are then normalized to the unit interval [0, 1] in order to speed up the swarm convergence of the DE algorithm.

3.2 Deep Neuroevolution Approach

As described above, we represent the multi-layer feedforward architecture of ANN as position vector. We first train the network with all possible architectures using Stochastic Gradient Descent (SGD) algorithm due to its fast learning convergence. Then we put the network with specific architecture through a second training phase using Adadelta optimizer with an adaptive learning rate, which has proven to be slow but lead to quality convergence. This two-phase training process is repeated twice for every network architecture, and we use the best result from the two trials as the final measure of its quality, as random initializations of the ANN make the same architecture produce different results in different trials. Due to the consideration of computing resource we restrained the number of runs to two. After architecture evaluation, the old position vector is updated by the DE.

3.3 Setting of Parameters

Table 1 shows the parameters we used in the subsequent energy use prediction experiments. It is noted that the initial value of learning rate would be reduced by 25% if there is no improvement for the last 30 epochs of training.

4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Modeling Results

From Table 2, we can see that the first two hidden layers have fewer hidden units than the last two hidden layers in the best architecture found. Table 3 shows how the NN model with the best architecture performs on the training, validation and test set. It is shown that the model performs better on the training and validation sets than the test set.

By observing the last column of Table 3, in terms of RMSE, MAE and R^2, the proposed neuroevolution approach outperform the best-performing GBM model in [10] (66.65, 35.22 and 0.57, respectively, therein).

4.2 Statistical Analysis of Modeling Errors

To get a deeper understanding of the testing errors of the proposed deep ANN model, we present the statistics of testing errors in Table 4. We can observe that the model usually predicts too high energy consumption but has a positive mean of errors, the maximum and minimum testing errors are 332 and -249, respectively. The results indicate that the worst prediction error occurs when the model prediction is lower than the actual value.

From Fig. 3, we observe that the majority of prediction errors are negative (with too high predictions) and there are also many big, positive errors.

Table 5 shows the statistics of errors due to too high and too low predictions, where we consider absolute values of the testing errors (i.e., without considering their signs). We can see that 75% of the negative errors (due to too high predictions of the model) fall within the interval [0.006, 26.72], while 75% of the positive errors (due to too low predictions) are in the interval [0.01, 29.00]. To summarize, the testing error analysis above shows that the model more frequently predicts higher than the actual value, however this is outweighed by the magnitudes of the (positive) errors due to too lower predictions. When the actual energy use is in the relatively higher (e.g., in the range between 150 and 400), the model has difficulty in making accurate predictions.

5 CONCLUSION

In this work, we present a DE-algorithm-based approach to determining the optimal architecture of deep neural network for predicting the household energy consumption. In summary, the results showed that the proposed method reduced MSE by 98.32

Table 1: The parameters setting in experiments.

Parameter	Meaning	Value
n_{part}	# of particles in the swarm	20
n_{dim}	Dimen. of search space	4
m	Max # of generations	30
CR	Crossover rate	0.5
F	Mutation rate	0.8
$\text{min}_{\text{nodes}}$	Min. # of nodes in a hidden layer	30
$\text{max}_{\text{nodes}}$	Max. # of nodes in a hidden layer	80
n_{inputs}	Input dimensionality	30
n_{outputs}	Output dimensionality	1
n_{hidden}	# of hidden layers	2
$size_{\text{batch}}$	Size of mini-batch per parameter update	200
$\text{max}_{\text{epochs}}$	Max. # of training epochs	1000
patience	# of training epochs without further improvement before stopping training	50
lr	Initial learning rate	0.001

Table 2: The DE optimization results.

The best validation MSE before optimization	2369.34
Best architecture (NN with 4 hidden layers)	30-**36-33-53-46**-1
The validation MSE of the best architecture	2271.02
No. of iterations required to find the best architecture	17
Reduction in MSE	98.32

Table 3: Training, validation and test error metrics of the DE-ANN model constructed.

	Training	Validation	Testing
RMSE	38.49	47.65	52.79
MAE	21.61	25.88	27.63
MAPE (%)	26.80	32.34	32.66
R^2	0.68	0.48	0.38

Table 4: The statistics of the model testing errors.

Max	333.22
75^{th} percentile	10.09
Mean	3.23
25^{th} percentile	-14.87
Min	-278.82
s.d.	52.69
Percentage of negative errors (%)	55
Percentage of positive errors (%)	45

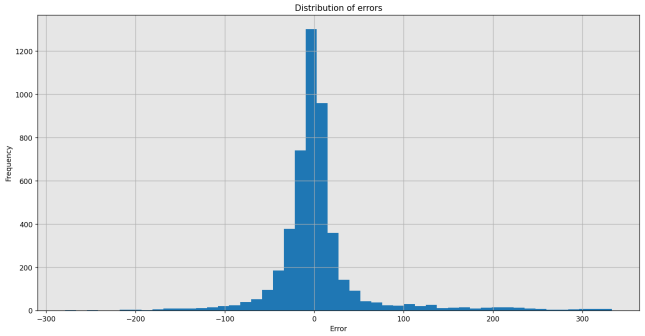

Figure 3: The distribution of model testing errors.

using the best NN architecture found by DE algorithm after 17 generations. Testing error analysis shows that 55% of model predictions are higher than the actual value, 45% of them are lower than the actual value, and the mean error is 3.23 (since the magnitude of the positive errors outweigh that of the negative errors).

In conclusion, the results over the realistic dataset showed that the proposed DL method can significantly reduce the prediction error of the deep neural network by selecting the best network architecture with DE algorithm.

Table 5: The statistics of the absolute errors.

	Absolute value	
	Negative errors	**Positive errors**
s.d.	29.00	58.13
Max	278.02	333.22
75^{th} percentile	26.72	29.00
Mean	22.23	34.20
25^{th} percentile	5.56	5.38
Min	0.006	0.01

REFERENCES

[1] V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis, A review of major application areas of differential evolution. in *Advances in Differential Evolution*, Studies in Computational Intelligence, vol. **143**, Berlin, Heidelberg: Springer, pp. 197-238, 2008.

[2] W. F. Sacco, N. Henderson, A. C. Rios-Coelho, M. M. Ali, and C. M. N. A. Pereira, Differential evolution algorithms applied to nuclear reactor core design. *Annals of Nuclear Energy,* vol. **36** (8), pp. 1093-1099, 2009.

[3] W. Huang, T. Xu, K. Li, and J. He, Multiobjective differential evolution enhanced with principle component analysis for constrained optimization. *Swarm and Evolutionary Computation,* vol. **50**(100571), pp. 1-14, 2019.

[4] Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, Vol. **90**, pp. 1-24, 2020.

[5] M. F. Ahmad, N. A. Isa, W. H. Lim, and K. M. Ang, Differential evolution: A recent review based on state-of-the-art works. *Alexandria Eng. J.*, Vol. **61** (5), pp. 3831-3872, 2022.

[6] R. Wang, J. Zhang, Y. Zhang, and X. Wang, Assessment of human operator functional state using a novel differential evolution optimization based adaptive fuzzy model. *Biomedical Signal Processing and Control*, vol. **7**(5), pp. 490-498, 2012.

[7] P. A. Whigham, and G. Dick, Implicitly controlling bloat in genetic programming. *IEEE Trans. on Evolutionary Computation,* vol. **14**(1), pp. 173-190, 2010.

[8] A. P. Engelbrecht, *Computational Intelligence: An Introduction* (2nd edition), John Wiley & Sons, 2007.

[9] A. Slowik, and M. Bialko, Training of artificial neural networks using differential evolution algorithms. in *Proc. of 2008 Conf. on Human System Interaction (HSI2008),* pp. 60-65, 2008.

[10] L. M. Candanedo, V. Feldheim, and D. Deramaix, Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, Vol. **140**, pp. 81-97, 2017.

[11] P. G. A. Njock, S.-L. Shen, A. Zhou, and G. Modoni, Artificial neural network optimized by differential evolution for predicting diameters of jet grouted column. *J. of Rock Mechanics and Geotechnical Eng.*, Vol. **13**, pp. 1500 – 1512, 2021.

[12] E. Hancer, A new multi-objective differential evolution approach for simultaneous clustering and feature selection. *Engineering Applications of Artificial Intelligence*, Vol. **87**, 103307, pp. 1-9, 2020.

[13] M. M. Fouad, A. I. El-Desouky, R. Al-Hajj, and E.-S. M. El-Kenawy, Dynamic group-based cooperative optimization algorithm. *IEEE Access*, vol. **8**, pp. 148378-148403, 2020.

[14] W. H. Bangyal, K. Nisar, Ag. A. B. Ag. Ibrahim, M. R. Haque, J. J. P. C. Rodrigues, and D. B. Rawat, Comparative analysis of low discrepancy sequence-based initialization approaches using population-based algorithms for solving the global optimization problems. *Applied Science*, vol. **11**, 7591, 2021.

[15] J. Zhang, and S. Yang, A novel PSO algorithm based on an incremental-PID-controlled search strategy. *Soft Comput*, vol. **20**, pp. 991-1005, 2016.

[16] G. Memarzadeh, and F. Keynia, Short-term electricity load and price forecasting by a new optimal LSTM-NN based prediction algorithm. Electric Power Systems Research, Vol. 192, pp. 1-14, 2020.

[17] J.-H. Han, D.-J. Choi, S.-U. Park, and S.-K. Hong, Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network. *J. of Electrical Engineering & Technology*, Vol. **15**, pp. 721-726, 2020.

[18] R. Storn, and K. V. Price, Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, Vol. **11**, pp. 341-359, 1997.