2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Evolved neural networks for building energy prediction

Roberto Santana
Intelligent Systems Group
University of the Basque Country (UPV/EHU)
San Sebastian, Spain
roberto.santana@ehu.eus

Irati Prol-Godoy
Energy in Buildings Group
University of the Basque Country (UPV/EHU)
Vitoria, Spain
rati.prol@ehu.eus

Ana Picallo-Pérez
Energy in Buildings Group
University of the Basque Country (UPV/EHU)
Vitoria, Spain
ana.picallo@ehu.eus

Iñaki Inza
Intelligent Systems Group
University of the Basque Country (UPV/EHU)
San Sebastian, Spain
inaki.inza@ehu.eus

***Abstract*—Improving buildings' energy efficiency is an essential component in the efforts for reducing the carbon footprint. The design of more accurate machine learning models for forecasting energy use in buildings can help to reach this goal since these models can be integrated as part of the management systems. A variety of machine learning algorithms have been used for different classes of building energy predictions problems. In this paper we investigate two questions related to the use of neural networks for building energy predictions: The benefits of optimized neural network configurations that include the architecture and some hyperparameters, and the impact on the performance of the amount of data available to train the networks. Our results show that combine optimization of architectures and hyperparameters can significantly improve the accuracy of the neural networks in some problems and that the availability of training data should be taken into account when deciding to apply neural networks over other machine learning methods for building energy prediction problems.**

***Index Terms*—energy efficiency, energy use prediction, neural networks, machine learning**

I. Introduction

The emergence of climate change has accentuated the importance of search for efficient uses of energy. Building energy consumption accounts for an important part of the total global energy consumption [1] and obtaining precise models for building energy predictions has become critical for decreasing energy usage. However, creating such accurate models remains a significant challenge due to several factors. Among them are the variety and complexity of the elements that impact building energy consumption. Aspects such as the changing behavior of the buildings' users, the weather conditions, and the structural characteristics of the building should be taken into account at the time of defining or learning the models.

Building energy prediction models can be grouped into three classes [2]: physical energy models (i.e., white box), data-driven models (i.e., black box), and hybrid models (i.e., gray box). Data-driven models have received an increasing attention in the literature [1], [3], [4] since they allow to exploit the advantages of machine learning algorithms and benefit from the developments in home sensors. Among the ML approaches, a number of works have reported successful applications of neural networks for building energy prediction [5], [6].

One of the known limitations of neural networks is the difficulty associated to find an optimal set of hyperparameters and architecture. Usually, the choice of the neural network components is made by the practitioner based on previous experience or evaluating a limited number of neural network configurations. Hyperparameter optimization methods have been extensively applied in machine learning [7]. Recently, neural architecture search (NAS) [8], [9] methods have been proposed as a more sensible way to also search for optimal network architectures. Among the most used approaches for NAS are: reinforcement learning [10], Bayesian optimization [11] and neuroevolutionary algorithms [12]–[14].

In this paper, we propose the application of neuroevolutionary algorithms for searching neural network configurations (i.e., choice of the architecture and the hyperparameters) that improve the models' accuracy for building energy prediction problems. We build on the works of Tsanas and Xifara [15] and Candanedo et al. [16] which introduce two different types of building energy prediction problems. The goal of the first problem is the prediction of heating and cooling loads for different building configurations and the goal of the second is the design of data driven prediction models of energy use of appliances. In [15], the characterization of the buildings configurations is used to predict the target variables by means

The authors would like to thank the Misiones Euskampus 2.0 programme for the financial help received through Euskampus Fundazioa. I. Inza and R. Santana acknowledge partial support by the Research Groups 2022-2024 (IT1504-22) and the Elkartek Program (KK-2020/00049, KK-2022/00106, SIGZE, KK-2021/00065) from the Basque Government, and the PID2019-104966GB-I00 and PID2022-137442NB-I00 research projects from the Spanish Ministry of Science.

of the machine learning algorithms, and in [16], weather data and information collected from sensors in a low-energy house are used to create classical machine learning models for the energy use of appliances.

The rest of the paper is organized as follows: The next section introduces the energy prediction problem we address in the paper and reviews related work on the use of ML algorithms for energy use prediction. In Section III, we introduce the neuroevolutionary approach to learn models for building energy prediction. Section IV presents the experimental framework and reports the results of the experiments, as well as a comparison with other regression algorithms. We conclude the paper in Section V.

II. Problem description and related work

A. Problem description

There are different ways in which machine learning methods can be used for enhancing building energy prediction. We consider two scenarios. In the first, a set of of building designs is available, and for each design a number of variables that characterize the design (e.g., surface area, wall area, roof area, etc.) are also available together with the heating and cooling loads obtained from simulations of the building specification. The problem is then to predict the cooling and heating loads from the variables that describe the buildings designs.

In the second scenario, the analysis is focused on a single house with multiple rooms, and lights and appliances in each room. A number of variables describe the recording from sensors within the house and other variables capture the weather conditions in the surrounding area. The goal of the problem is then to predict the energy consumption of lights and appliances given these variables.

Both problems can be addressed as the regression of two target variables, and we use the mean squared error (MSE) as a measure of the prediction accuracy. We focus on neural networks since they are able to model non-linear relationships between the variables and have shown good results in previous work. More specifically, we consider the multi-layer perceptron model (MLP) that *simultaneously* predicts the two target variables of the problem. While other types of neural networks could be applied (e.g., recurrent neural networks for the second problem), the choice of MLP is due to its simplicity and to the second objective of this work, that is to investigate the capacity of neuroevolutionary algorithms to improve the results of building energy prediction models.

B. Related work

There have been an increasing number of works that report different machine learning approaches for building energy prediction [1], [3], [6]. In this section, we briefly review some of the works that are related the most to the approach introduced in this paper.

In [15], Tsanas and Xifara introduced the dataset we used for the first problem. They made a geometrical exploration of different building designs, starting from an elementary cube (3,5m × 3,5m × 3,5m) from which 12 building forms composed of 18 elements (elementary cubes) were generated. All the buildings have the same volume but different surface areas and dimensions. Moreover, for each of the 18 elements the material used (walls, floors, roofs, and windows) were the same. Each of the 768 building designs was simulated by evaluating the effect of eight input variables characterizing the designs on the heating load (HL) and cooling load (CL). The authors compared the random forest model to the iteratively reweighted least squares [17] and concluded that the former produced better results. Other machine learning algorithms have been applied to this dataset including multivariate extreme gradient boosting [18], support vector machines and hybrid approaches [19], and ensembles of trees [20]. Artificial neural networks optimized using different metaheuristic methods are proposed for this problem in [21]. However, the optimization is focused in the parameters of a fixed architecture, i.e., the weights and biases of the neural network and not in neural network configurations.

The second dataset used in our paper was introduced by Candanedo et al. in [16]. In this second scenario, the analysis is focused on a single house with multiple rooms and several lights and appliances in each room. Information about different data sources and environmental parameters (indoor and outdoor conditions) was recorded every 10 minutes together with the energy (Wh) data corresponding to the lights and appliances. The problem addressed in [16] was limited to the prediction of the appliances energy consumption. Here, we tackle the question of simultaneously predicting the light and appliances energy consumption. In [16], four regression models were applied: multiple linear regression model, support vector machine with radial basis function kernel (SVM-radial), random forests, and gradient boosting machines. Subsequent works have proposed the application of neural networks that exploit the temporal component [22], [23].

III. Evolution of MLP configurations

The main objective of the optimization problem we address is to find a neural network configuration that maximizes the performance. Therefore, a solution representation is needed for MLPs. The two primary factors that define the MLP architecture are the number of hidden layers and the number of neurons in each layer. In addition, we consider two elements that also influence the behavior of a network: the weight initialization functions and the activation functions used in each layer. The MLP weights are initialized by drawing values from a `normal` or `uniform` distribution, or by applying the `xavier` [24] variation of the `normal` initialization. For the choice of the activation function the following functions have been considered: `Identity`, `ReLU`, `eLU`, `Softplus`, `Softsign`, `Sigmoid`, `Hyperbolic Tangent`. Finally, for each layer, two Boolean variables define whether dropout and batch normalization are applied.

All the factors that define the neural network configuration are encoded as a list. This type of declarative representation which exclusively contains the specification of the neural network has been previously used to address other prediction

problems [25]–[28]. The evaluation of a network configuration implies the creation of a network architecture by decoding the specification in the list. Once the network configuration have been decoded from the list, a train dataset is used to learn parameters of the MLP (weights and biases). These parameters are learned from data using a variant of batch gradient descent, for given batch size and number of epochs. After training have been concluded, the fitness of the neural network is the MSE computed on a validation dataset.

While the list representation is easy to interpret and suitable for the application of genetic operators, is has as a drawback that every time that an MLP configuration is evaluated it has to be trained in order to learn its parameters.

The neuroevolutionary algorithm works by applying different types of mutation operators to the selected solutions. The selection method applied is truncation selection and the mutation operators implemented as part of the algorithm are the following:

- layer_change: Randomly reinitializes the description of a layer chosen at random, e.g., its weight initialization and activation functions; and the number of neurons.
- add_layer: Introduces a new (randomly initialized) layer in a random position of the network.
- del_layer: Deletes a randomly chosen layer.
- activ_change: Changes the activation function of a random layer to another randomly chosen function.
- weight_change: Similarly to activ_change, it changes the weight initialization function of a layer.

The neuroevolutionary approach are shown in Algorithm 1.

Algorithm 1: Neuroevolutionary algorithm for MLPs.

Set $t \Leftarrow 0$. Create a population D_0 by generating N random MLP descriptions;
while *halting condition is not met* **do**
  Evaluate D_t using the fitness function;
  From D_t, select a population D_t^S of $Q \leq N$ solutions according to a selection method;
  Apply mutation with probability p_m to D_t^S and create the offspring set O_t. Choice of the mutation operator is made uniformly at random;
  Create D_{t+1} by using the selection method over $\{D_t, O_t\}$;
  $t \Leftarrow t + 1$;
end while

Algorithm 1 has been implemented using the `deatf` library [1], an extension to Tensorflow2 of the Evoflow library [2] [26], originally conceived to evolve neural networks implemented in tensorflow [29], and based on the `DEAP` library [30].

IV. Experiments

The goal of the experiments is to evaluate the performance of the neuroevolutionary approach for obtaining more accurate models. We compare the optimized architectures with a set of neural networks with a random choice of the architectures. Notice that, in general, there is no information that can guide the choice of the architecture for these problems. We also investigate the behavior of the algorithm along generations and present a comparison with other machine learning approaches.

A. Experimental framework

For each of the two problems, we define a similar procedure for splitting the data into train, validation and test datasets. First, we define the percent of the total examples that will be included in the test dataset. We use percent values in $\{20, 40, 60, 80\}$. After separating the test dataset, the remaining data is split in equal parts for the train and validation datasets. The aim of considering different percent values was to assess the sensitivity of the neuroevolutionary algorithm to the amount of training data. For the first problem, the train, validation and test datasets have been randomly selected. For the second problem, and taking into consideration the temporal component of this data, the test dataset will always correspond to the last registers of the time series.

The batch size and number of epochs used for the Heating and cooling problem are 10 and 30, respectively; while for the Energy consumption problem these values are 50 and 10. These settings have been determined taking into account the different size of the data for these two problems: 768 and 19735 examples, respectively. For both problems, the maximum number of hidden layers was set to 8 and the maximum number of neurons in each layer was also set to 8. The gradient descent method used to learn the parameters of the neural network is Adam [31].

The population size of the algorithm is 50 and the number of generations is 40. The mutation probability was 0.8 and the best solution in each population is kept for the next population. For each problem, 20 runs are executed and the statistics are computed using the average of these runs.

B. Evolution of the algorithm

We start our analysis by examining the evolution of the algorithm in terms of the best fitness value achieved in each generation. Notice that these fitness values are calculated using the validation set. The quality of the final solutions will be evaluated using the test dataset that is not used during the evolution.

Figure 1 shows the best MSE values achieved in each generation, computed as the average of 20 runs, for the Heating and cooling load prediction problem and different sizes of the train and test datasets. Figure 2 shows similar information but in this case the fitness values have been normalized by dividing them by the best fitness value in the initial population. This normalization allows us to compare the improvements along the evolution for different percent values.

It can be seen in Figure 1 that in comparison to the initial random configurations there is an improvement in the fitness for all percent values. The MSE values at the end of the evolution are lower when the percent of solutions in the test

[1] Available from https://github.com/IvanHCenalmor/deatf
[2] Available from https://github.com/unaigarciarena/EvoFlow

dataset is smaller. This seems to indicate that the algorithm takes advantage of having a higher number of examples in the training and validation sets. However, when considering the normalized results shown in Figure 2, it is clear that the relative fitness improvement is similar for all percent values. There is a five-fold decrease in the error in comparison with the random architectures. Also, it is apparent from figures 1 and 2 that the main improvement to the fitness function occurs in the initial generations and a relatively small number of evaluations is required in order to find well-performing configurations.

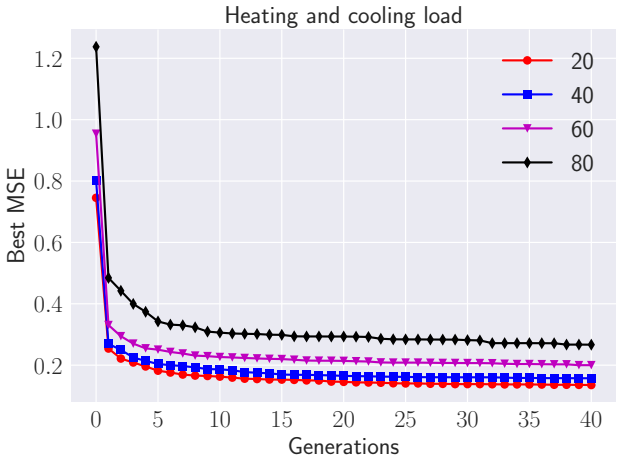

Fig. 1. Best MSE value in each generation for the Heating and cooling load prediction problem and different proportions of examples in the test dataset.

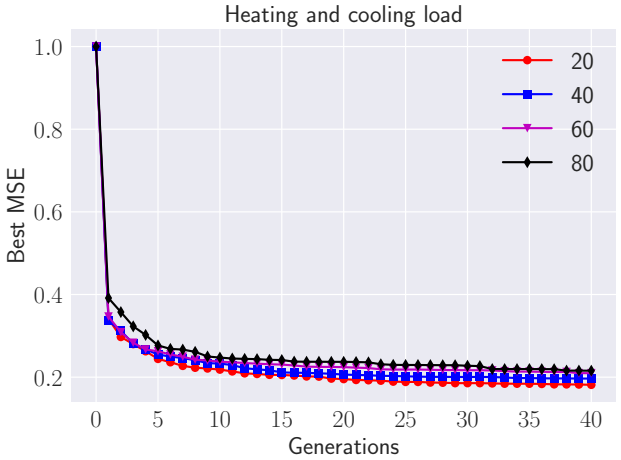

Fig. 2. Best (normalized) MSE value in each generation for the Heating and cooling load prediction problem and different proportions of examples in the test dataset.

We conduct a similar analysis for the Energy consumption problem. The results are shown in figures 3 and 4. For this problem, there are more significant differences in the quality of the final solutions according to the amount of examples used for training and validation. Using a high number of training solutions is more critical in this scenario. This might be due to the fact that there is a temporal component in this problem and the patterns that relate the features with the target variables are more complex and difficult to capture by the models. However, an inspection of Figure 4 reveals an interesting phenomenon: The performance of the neuroevolutionary algorithm is better when the percent of test data is 60%, and not, as in the previous scenario, when this percent is 20. It is also clear that for this problem more generations are required in order to find optimized configurations. The improvements to the MSE achieved by the algorithm are between 13 and 23 percent.

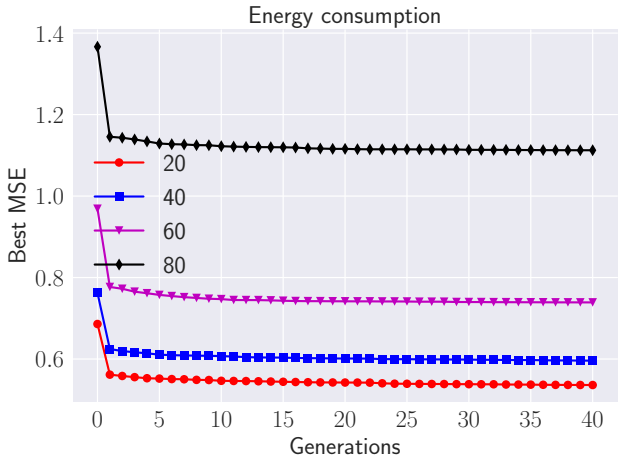

Fig. 3. Best MSE value in each generation for the Energy consumption problem and different proportion of examples in the test dataset.

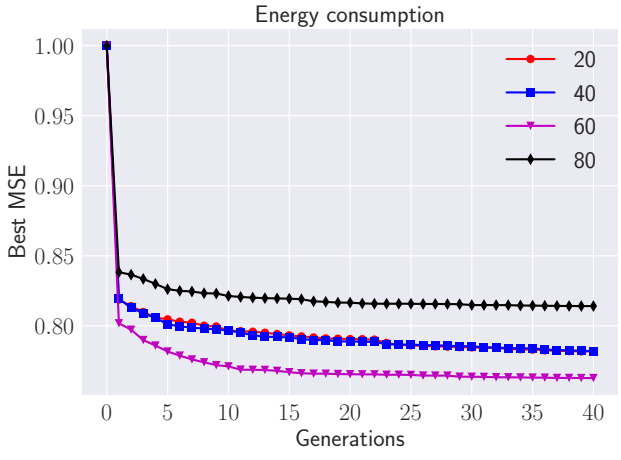

Fig. 4. Best (normalized) MSE value in each generation for the Energy consumption problem and different proportion of examples in the test dataset.

C. Quality of the final solutions

To compute the quality of the neural networks evolved by the algorithm, we first retrain the neural networks for all the configurations in the final population using the train and validation data. Then, these networks are used to predict the

test dataset and the MSE is computed from these predictions. Figure 5 shows the average MSE computed from the 20 runs for each combination of problem and percent of test data. It can be seen in the figure, that the predictions improve as the size of the test dataset increases. This is a somewhat unexpected behavior since we would expect that the accuracy of the model will improved with the data. Therefore, we carefully analyzed the predictions produced by the models on the test data. The effect seems to be due to the fact that, in the test dataset, there are more examples in those regions where the model produces accurate predictions, and therefore the MSE is better. However, there is a limit for this pattern, as the difference between the percent values of 80 and 60 is much smaller than the difference between 40 and 20.

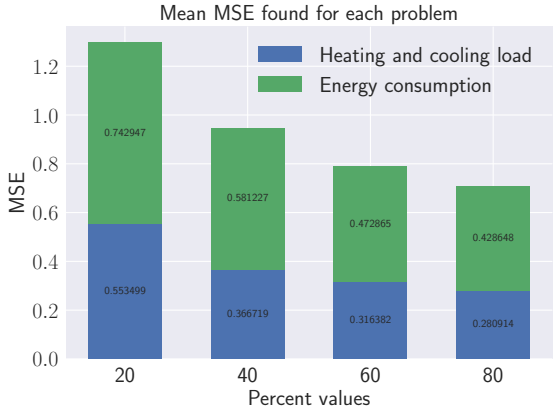

Fig. 5. Mean MSE for the solutions in the final population.

We also computed the best absolute neural networks obtained out of the 20 runs. To determine which is the best solution in the final population, we use the predictions made on the train+validation datasets. Then, we identified the MSE value obtained by this network on the test data. This means that the test dataset is not used to decide which of the networks in the final population is the best. The MSE values corresponding to the best solutions found by the neuroevolutionary algorithm are shown in Figure 6 for the combination of problems and percent value. The same trend observed in Figure 5 can be appreciated in this figure. It can be seen that the algorithm is able to obtain very accurate neural networks.

D. Comparison with other methods

Finally, we compare the results achieved by the evolved networks to five regression methods previously used for these problems [15], [16], [18], [20]. To implement the regressors, we used the scikit-learn library [32] with the values defined by default in the library. We only selected regressors whose implementation allows for simultaneously predicting the two target variables. The MSE values obtained by the algorithms for each problem and percent values are shown in Table I, where the best values of each column are underlined.

The analysis of Table I indicates that for the Heating and cooling load prediction problem the best results are achieved

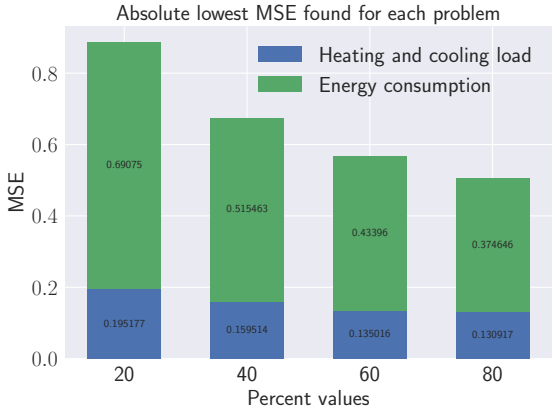

Fig. 6. Best (absolute) MSE of the neural networks found by the neuroevolutionary algorithm.

	Load prediction				E. consumption prediction			
Percent	20	40	60	80	20	40	60	80
ENN	0.20	0.16	0.14	0.13	0.69	0.52	0.43	0.37
LR	0.19	0.13	0.16	0.20	0.69	0.53	0.53	0.53
RFR	0.05	0.06	0.08	0.09	1.59	0.90	0.62	0.61
KNNR	0.11	0.09	0.10	0.10	0.75	0.56	0.46	0.41
DTR	0.06	0.06	0.08	0.09	1.91	1.44	0.89	0.77
ETR	0.06	0.06	0.08	0.09	1.12	0.65	0.52	0.52

TABLE I
COMPARISON OF THE BEST EVOLVED NEURAL NETWORKS (ENN) TO OTHER FIVE REGRESSION APPROACHES COMMONLY APPLIED IN THE LITERATURE. LR: LINEAR REGRESSION, RF: RANDOM FOREST REGRESSOR, KNNR: KNN REGRESSOR, DTR: DECISION TREE REGRESSOR, ETR: EXTRATREES REGRESSOR.

using a linear regression approach. For this problem, for which a limited amount of data is available, the neural networks are not competitive with the other machine learning approaches. The situation is reversed for the Energy consumption prediction problem. In this case, the evolved neural networks produce the best results and significantly outperform the RFR. It should be emphasized that the complexity of the neural networks was limited to 8 hidden layers with a maximum of 8 neurons each. This means that there is room for improvement in the evolution of the neural networks if more representation capacity were needed for dealing with the second problem.

V. CONCLUSIONS

There is an urgent need for accurate predictive models that could contribute to energy conservation. In the context of building energy predictions, learning such models is very difficult due to the multiple and heterogeneous factors that influence the energy consumption. Therefore, efforts are required to investigate how to improve the accuracy of these models in order to support better decision-making. In this paper we have looked into the ability of neuroevolutionary algorithms to enhance the neural network accuracy. We have tackled two building energy prediction problems with different characteristics. For these problems, we have also analyzed the role played by the amount of training data in the performance of the neuroevolutionary algorithms.

Our results show that important gains can be achieved in the accuracy of the models. However, these gains are not of the same magnitude for all the problems. Moreover, as seen in our experiments, for some problems, improving the models may require a higher number of generations. While neuroevolutionary and other neural architecture search algorithms can be costly in terms of computational time, once the best performing models have been learned they can be used multiple times. Their application is therefore justified if the building energy models are going to be used on a regularly basis and if the gains in accuracy are translated into remarkable energy savings. Future work could consider the application of neuro-evolutionary approaches with partially labeled data [28], extracting valuable problem information from evaluated solutions [33], and the combination of neuroevolution with surrogate of the problem fitness functions [34].

REFERENCES

[1] Y. Sun, F. Haghighat, and B. C. Fung, "A review of the-state-of-the-art in data-driven approaches for building energy prediction," *Energy and Buildings*, vol. 221, p. 110022, 2020.

[2] Y. Chen, M. Guo, Z. Chen, Z. Chen, and Y. Ji, "Physical energy and data-driven models in building energy prediction: A review," *Energy Reports*, vol. 8, pp. 2656–2671, 2022.

[3] M. Khalil, A. S. McGough, Z. Pourmirza, M. Pazhoohesh, and S. Walker, "Machine learning, deep learning and statistical analysis for forecasting building energy consumption-a systematic review," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105287, 2022.

[4] H.-x. Zhao and F. Magoulès, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586–3592, 2012.

[5] C. Lu, S. Li, and Z. Lu, "Building energy prediction using artificial neural networks: A literature survey," *Energy and Buildings*, vol. 262, p. 111718, 2022.

[6] R. Olu-Ajayi, H. Alaka, I. Sulaimon, F. Sunmola, and S. Ajayi, "Building energy consumption prediction for residential buildings using deep learning and other machine learning techniques," *Journal of Building Engineering*, vol. 45, p. 103406, 2022.

[7] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. Springer Nature, 2019.

[8] T. Elsken, J.-H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, pp. 1–21, 2019.

[9] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, "Neural architecture search: Insights from 1000 papers," *CoRR*, vol. abs/2301.08727, 2023. [Online]. Available: http://arxiv.org/abs/2301.08727

[10] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578

[11] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with Bayesian optimisation and optimal transport," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[12] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[13] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[14] E. Galván and P. Mooney, "Neuroevolution in deep neural networks: Current trends and future challenges," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 476–493, 2021.

[15] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, 2012.

[16] 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.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, 2006, vol. 4, no. 4.

[18] T. H. Saragih, R. Ramadhani, M. I. Mazdadi, and M. Haekal, "Energy efficiency in buildings using multivariate extreme gradient boosting," in *2022 Seventh International Conference on Informatics and Computing (ICIC)*. IEEE, 2022, pp. 1–5.

[19] L. T. Le, H. Nguyen, J. Zhou, J. Dou, and H. Moayedi, "Estimating the heating load of buildings for smart city planning using a novel artificial intelligence technique PSO-XGBoost," *Applied Sciences*, vol. 9, no. 13, p. 2714, 2019.

[20] L. Schmid, A. Gerharz, A. Groll, and M. Pauly, "Tree-based ensembles for multi-output regression: Comparing multivariate approaches with separate univariate ones," *Computational Statistics & Data Analysis*, vol. 179, p. 107628, 2023.

[21] G. Zhou, H. Moayedi, M. Bahiraei, and Z. Lyu, "Employing artificial bee colony and particle swarm techniques for optimizing a neural network in prediction of heating and cooling loads of residential buildings," *Journal of Cleaner Production*, vol. 254, p. 120082, 2020.

[22] E. Batbaatar, H. W. Park, D. Li, M. Li, and K. H. Ryu, "Deepenergy: Prediction of appliances energy with long-short term memory recurrent neural network," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2018, pp. 224–234.

[23] M. Sajjad, Z. A. Khan, A. Ullah, T. Hussain, W. Ullah, M. Y. Lee, and S. W. Baik, "A novel CNN-GRU-based hybrid approach for short-term residential load forecasting," *Ieee Access*, vol. 8, pp. 143 759–143 768, 2020.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[25] U. Garciarena, R. Santana, and A. Mendiburu, "Evolved GANs for generating Pareto set approximations," in *Proceedings of the 2018 on Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 434–441. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3205455.3205550

[26] ——, "EvoFlow: A Python library for evolving deep neural network architectures in tensorflow," in *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI-2020)*. IEEE, 2018, pp. 2288–2295. [Online]. Available: https://ieeexplore.ieee.org/document/9308214

[27] U. Garciarena, A. Mendiburu, and R. Santana, "Analysis of the transferability and robustness of GANs evolved for Pareto set approximations," *Neural Networks*, vol. 132, pp. 281–296, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608020303269

[28] R. Santana, I. Hidalgo-Cenalmor, U. Garciarena, A. Mendiburu, and J. A. Lozano, "Neuroevolutionary algorithms driven by neuron coverage metrics for semi-supervised classification," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2023, pp. 679–682.

[29] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," in *Osdi*, vol. 16, no. 2016. Savannah, GA, USA, 2016, pp. 265–283.

[30] F.-A. Fortin, D. Rainville, M.-A. G. Gardner, M. Parizeau, C. Gagné *et al.*, "DEAP: Evolutionary algorithms made easy," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2171–2175, 2012.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] U. Garciarena, N. Lourenço, P. Machado, R. Santana, and A. Mendiburu, "On the exploitation of neuroevolutionary information," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021, pp. 279–280.

[34] R. Santana, A. Mendiburu, and J. A. Lozano, "Critical issues in model-based surrogate functions in estimation of distribution algorithms," in *Proceedings of the 4th Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO-2013)*, ser. Lectures Notes in Computer Science. Chennai, India: Springer, 2013, pp. 1–13.