Waste generation prediction under uncertainty in smart cities through deep neuroevolution

Predicción de la Producción de Residuos con incertidumbre en la Ciudad inteligente mediante neuroevolución profunda

Andrés Camero [1*], Jamal Toutouh [2], Javier Ferrer [1], Enrique Alba [1]

[1]Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga. Campus de Teatinos s/n. C.P. 29071. Málaga, España.
[2]MIT Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. 32 Vassar St. C. P. MA 02139. Cambridge, USA.

CITE THIS ARTICLE AS:

A. Camero, J. Toutouh, J. Ferrer and E. Alba. "Waste generation prediction under uncertainty in smart cities through deep neuroevolution", *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 93, pp. 128-138, Oct-Dec 2019. [Online]. Available: `https://www.doi.org/10.17533/udea.redin.20190736`

ARTICLE INFO:

Received: October 31, 2018
Accepted: July 24, 2019
Available online: July 24, 2019

KEYWORDS:

Deep neuroevolution, deep learning, evolutionary algorithms, smart cities, waste collection

Neuroevolución profunda, aprendizaje profundo, algoritmos evolutivos, ciudad inteligente, gestión de residuos

ABSTRACT: The unsustainable development of countries has created a problem due to the unstoppable waste generation. Moreover, waste collection is carried out following a pre-defined route that does not take into account the actual level of the containers collected. Therefore, optimizing the way the waste is collected presents an interesting opportunity. In this study, we tackle the problem of predicting the waste generation ratio in real-world conditions, i.e., under uncertainty. Particularly, we use a deep neuroevolutionary technique to automatically design a recurrent network that captures the filling level of all waste containers in a city at once, and we study the suitability of our proposal when faced to noisy and faulty data. We validate our proposal using a real-world case study, consisting of more than two hundred waste containers located in a city in Spain, and we compare our results to the state-of-the-art. The results show that our approach exceeds all its competitors and that its accuracy in a real-world scenario, i.e., under uncertain data, is good enough for optimizing the waste collection planning.

RESUMEN: El desarrollo insostenible de los países ha creado un problema debido a la imparable generación de residuos. Más aún, la recogida de residuos se realiza siguiendo una ruta predefinida que no tiene en cuenta el nivel real de los contenedores recogidos. Por lo tanto, optimizar la forma en que se recolectan los residuos presenta una oportunidad interesante. En este estudio, abordamos el problema de predecir la tasa de generación de residuos en condiciones reales, es decir, bajo incertidumbre. En particular, utilizamos una técnica neuroevolutiva profunda para diseñar automáticamente una red recurrente que encapsula el nivel de llenado de todos los contenedores de residuos en una ciudad a la vez, y estudiamos la idoneidad de nuestra propuesta cuando nos enfrentamos a datos ruidosos y defectuosos. Validamos nuestra propuesta utilizando un caso real, que consta de más de doscientos contenedores de residuos ubicados en una ciudad de España, y comparamos nuestros resultados con el estado del arte. Los resultados muestran que nuestra propuesta supera a todos sus competidores y que su precisión en un escenario del mundo real, es decir, bajo datos inciertos, es lo suficientemente buena para optimizar la planificación de la recolección de residuos.

1. Introduction

The World's population is moving from rural to urban areas and it is expected that this trend will continue. The number of inhabitants in cities will be about 75% of the World's population by 2050 [1].

* Corresponding author: Andrés Camero
E-mail: andrescamero@uma.es
ISSN 0120-6230
e-ISSN 2422-2844

The fast demographic growth, together with the concentration of the population in cities and the increasing amount of daily waste are factors that push to the limit the ability of waste assimilation by Nature. This fact has forced the authorities to examine the cost-effectiveness and environmental impact of our economic system.

The linear structure of our economy has reached its limits and the natural resources of our planet are drained. Thus, a more sustainable model of economy is needed. For example, the *circular economy* [2, 3], which consists

DOI: 10.17533/udea.redin.20190736

in the transformation of our waste into raw materials, proposing a new paradigm for a more sustainable future.

The unsustainable development of countries has created a problem due to the unstoppable waste generation. In addition, there are hardly any technological means to make an optimal management of the waste collection process. Nowadays, the solid waste collection is carried out without a previous analysis of the demand, i.e. following a manually defined route. This approach has severe limitations, one of the most important is the variability in the amount of waste that needs to be picked up. This is especially critic in the case of selective collection (plastic, paper, glass,...), where the waste volume is smaller than in the organic case. Thus, when dealing with recyclable waste, the planning of the optimal collection routes is even more influential.

An alternative to tackle the planning of the collection routes is to determine which containers should be collected. Note that the recyclable waste collection process represents 70% of the operational cost in waste treatment [4]. Thus a reduction in the number of unnecessary visits to semi-empty containers will save money! We aim to provide an alternative to predict if a container should be collected or not. Particularly, we propose to predict the filling level of the waste containers (all the containers involved in the operation at once) using a Recurrent Neural Network (RNN).

RNNs are top-notch at predicting time series, however as all Deep Learning (DL) techniques the selection of an appropriate network design is a tough task [5]. The use of automatic intelligent tools seems a mandatory requirement when addressing the design of RNNs, since the vast possible RNN architectures that can be generated defines a huge search space. In this sense, metaheuristics [6] emerged as efficient stochastic techniques able to address hard-to-solve optimization problems. Indeed, these algorithms are currently employed in a multitude of real world problems, e.g., in the domain of Smart City [7–13], showing a successful performance. Nevertheless, the use of such a methodology in the domain of DL is still limited [14].

On the other hand, real-world problems present many challenges, from technological issues to political restrictions. In our particular problem, there is an interesting problem that arises when dealing with the prediction of the filling level: *uncertainty*. Following previous works [15], we distinguish two types of uncertainty, *reliability against noise* and *robustness*. The latter measures uncertainty caused by imprecision of the decision variables of the solution, which is not relevant for our problem because solutions can always be implemented precisely. The former measures uncertainty which may come from many different sources such as sensory/human measurement errors as it is the case of the historical data of filling level of the waste containers. Therefore, in this study, we propose to study whether RNN could provide reliable solutions when provided with noisy/faulty data.

In this article, we extend the ideas presented by [16]. Particularly, using a hyper-parameter technique based on evolutionary computation, we design and train an RNN that predicts the filling level of the containers of a whole city. Then, we study the behavior of this approach when faced with prediction under uncertainty. To validate our proposal, we analyze a real-world case consisting of more than two hundred waste containers located in a city in Spain, and we compare our results against the ones presented by [17].

As a summary, in this study:

- We define a deep neuroevolutionary technique to automatically design an efficient RNN.
- We use our proposal to design and train an RNN that predicts the filling level of the waste containers of a real city and benchmark our results against the state-of-the-art.
- We study the RNN approach for predicting under uncertainty.

Therefore, the main contribution of this work in contrast to the previous work [16] is the analysis of the behavior of deep neuroevolution and RNN when faced with uncertain data.

The remainder of this paper is organized as follows. The next section briefly reviews the state-of-the-art of smart waste management. Section 3 discusses about the use of DL to predict the waste generation rate. Section 4 presents a deep neuroevolutionary approach to design an artificial neural network-based predictor of the filling level of the waste containers. Section 5 presents the experiments carried out, results, benchmark, and analyses. Finally, Section 6 outlines our conclusions and proposes the future work.

2. Smart waste management

The waste collection is a process with uncountable variants and constraints which have led to a multitude of studies in recent years due to its importance. The works in the literature could be classified, among other ways, according to the waste type that is treated: *residential waste* commonly known as garbage [10, 18], *industrial waste* where customers are more dispersed and the amount of waste is higher [19], *recyclable waste* [20] increasingly important for our society, where the collection frequency is lower than organic waste and *hazardous waste* where the probability of damage is minimized [21].

In the municipal solid waste collection [22], the authorities need global studies to quantify the waste generated in a period of time to be able to manage them. Particularly, the waste generation forecasting for Xiamen city (China) inhabitants was studied by [23]. The main difference with our approach is the granularity of the object under study. They predict the amount of waste produced by the whole city, in contrast, we predict every single container in a city (i.e. a disaggregated prediction of the whole city). This supposes a considerable increase of the complexity of the problem that is solved, because it is necessary to consider multiple aspects such as the location, the customs of the citizens, the population density of the area, etc. In the same research line, the impact of the intervention of local authorities on waste collection has also been studied [24], being this relevant in the medium-long term.

Regarding the location where the collection takes place, there exist multiple variants of the problem. There are *communal collections* where the local authority identifies a place shared by the community [11, 25], in most cases a local waste facility for recycling. In the other side we found the *kerbside collection* [26] where the household waste is collected from individual small containers located near each house. The intermediate case studied here is the analysis of containers that give service to several streets and blocks of flats [27].

In previous works [17, 28] the authors used machine learning techniques to predict the filling level of a container. Particularly, the authors used Linear Regression, Gaussian Processes and Support Vector Machines for regression to predict each container individually. In this work, we present a unique RNN able to generate predictions for the whole set of containers instead of creating and training individual predictors for each container.

3. Deep learning for waste generation prediction

In this study, we focus on waste generation prediction by applying DL based on specific type of artificial neural networks (ANN), RNN. As other ANNs, this type of networks are composed of multiple hidden layers between input and output layers. RNNs incorporate feedforward and feedback connections between layers to capture long-term dependency in an input. Thus, RNNs have successfully applied to address learning applications which involve sequential modeling and prediction as natural language, image, and speech recognition and modeling [29]. In turn, they have been applied in Smart Cities problems that require time dependent prediction [14].

We apply supervised learning, which consists in an iterative process that requires a training data set (N input-output pairs). As this study deals with the prediction of the filling levels, the inputs are the current filling level each container and the outputs are the next (future) filling levels. Thus, for each input, the ANN produces an output (i.e., a tentative future filling rate) which is compared to the expected output by using an error (cost or distance) function. Then, a procedure is applied to reduce this error by updating the network until a given stop criteria is reached [30].

Minimizing such learning error is a tough task. Backpropagation [31] (BP), a first-order gradient descent algorithm, is the most widely used method to address such issue. In order to apply BP on RNN, the network has to be unfold [32], i.e., the network is copied and connected in series a finite number of times (known as look back) to build an unrolled version of the RNN.

Large ANNs (as unfolded RNNs) suffer from overfitting to the training data set, i.e., the error on the training set is driven to a very small value, but when unseen new data is presented to the network the error dramatically increases [33]. In order to address this issue, a technique called dropout, which consists in including a stochastic procedure to the training process, is applied [34].

The accuracy and the generalization capability of the RNN prediction depends on a set of configuration hyper-parameters: number of layers, number of hidden units per layer, activation function, kernel size of a layer, etc. Thus, a promising research line in DL proposes to find specific hyper-parameters configurations for an ANN to improve its numerical accuracy [35, 36]. The results demonstrated that selecting the most suitable hyper-parameters for a given dataset provides more competitive results than using generalized networks.

Since training an RNN is costly (in terms of computational resources) and the number of RNN architectures is infinite (or extremely large if we impose restrictions to the number of hidden layers or neurons), we are enforced to define a smart search strategy to find an optimal RNN.

Among the many potential optimization techniques to find efficient ANN hyper-parameterization, a few authors have already applied metaheuristics [37, 38]. However, these solutions cannot be directly applied to deep neural networks (DNN), i.e. ANNs with one or more hidden layer, due to the high computational complexity of DNNs. Recently, new solutions specifically defined to address hyper-parameter optimization of DNNs by using metaheurisitcs are emerging: the deep neuroevolutionary approaches [5, 14, 39–41], showing competitive results in finding parameters that improve the accuracy and minimize the generalization error.

In this study, we focus on applying a deep neuroevolution approach to address the generation of container filling predictions. Our optimization method deals with the next main RNN parameters: the look back (i.e., how many times the net is unfold during the training), the number of hidden layers, and the number of neurons for each hidden layer.

4. Deep neuroevolutionary architecture optimization

In this section, we present the details of our proposal. First, we formally state the architecture optimization problem, and then we outline our deep neuroevolutionary approach to solve the problem.

4.1 Architecture optimization

Optimizing an ANN consists in finding an *appropriate* network structure (architecture) and a set of weights to solve a given problem [30]. Particularly, we can analyze the *suitability* of an ANN by measuring its generalization capability, i.e. the ability to predict/classify new (unseen) data.

In our case, we are interested in optimizing the architecture of an RNN. Therefore, we decided to train an RNN using BP (i.e. we are finding an appropriate set of weights given a network structure) and measure the *mean absolute error* (MAE) of the predicted values against the observed ones.

Equation 1 states the problem of finding an optimal architecture as a minimization problem, where *N* corresponds to the number of samples in the testing data set (X,Y), z_i stands for the predicted value of the *i*-th sample, and y_i corresponds to the ground truth of the *i*-th sample. Note that the RNN is fed with already predicted data ^{x}, and that the architecture is constraint by *B*, *H*, and *L*.

$$\begin{aligned} \text{minimize} \quad & \text{Fitness} = \frac{1}{N}\sum_{i}^{N} MAE(z_i, y_i) \qquad (1) \\ \text{subject to} \quad & B \leq \text{max\_look\_back} \\ & H \leq \text{max\_hidden\_layers} \\ & L \leq \text{max\_neurons\_per\_layer} \\ & \hat{x}_i = \begin{cases} x_0 & \text{if } i = 0 \\ z_{i-1} & \text{if } i > 0 \end{cases} \end{aligned}$$

4.2 Deep neuroevolution

To solve the problem stated in Equation 1 we designed a deep neuroevolutionary algorithm based on the $(1+1)$ Evolutionary Strategy (ES) [6] and on the Adam weights optimizer [42]. Our proposal is presented in Algorithm 1.

Algorithm 1 Self Adapting (1+1)ES-based RNN architecture optimizer

```
1: solution ← Initialize()
2: Evaluate(solution, evaluation_epochs)
3: evaluations ← 1
4: while evaluations ≤ max_evaluations do
5:     mutated ← Mutate(solution, mut_element_p, mut_length_p, max_step)
6:     Evaluate(mutated, evaluation_epochs)
7:     if Fitness(mutated) ≤ Fitness(solution) then
8:         solution ← mutated
9:     end if
10:    evaluations ← evaluations + 1
11:    SelfAdapting()
12: end while
13: solution ← Evaluate(solution, final_epochs)
14: return solution
```

A **solution** represents an RNN architecture and it is encoded as an integer vector of variable length, solution=$< s_0, s_1, ..., s_H >$. The first element, $s_0 \in [1, \text{max\_look\_back}]$, corresponds to the *look back*, while the following elements ($s_j, j \in [1, H]$), correspond to the number of Long Short-Term Memory (LSTM) cells of the *j*-th hidden layer, subject to $s_j \in [1, \text{max\_neurons\_per\_layer}]$ and $H \in [1, \text{max\_hidden\_layers}]$. Note that the number of hidden layers is defined by the length of the vector. The number of neurons of the output layer is defined accordingly to the inputed time series, i.e. we add a *dense* layer (fully connected) with a number of neurons equal to the number of dimensions of the output.

First, the **Initialize** function creates a new random solution. Then, the **Evaluate** function computes the **Fitness** of the solution. Specifically, the solution is decoded (into an RNN), then the net is trained using the Adam optimizer [42] for *evaluation_epochs* epochs using the *training data set* and finally the fitness value is computed using the *testing data set*.

Then, while the number of evaluations is less or equal than *max_evaluations*, the evolutionary process takes place. Starting from a solution, the **Mutate** function generates a new **mutated** solution, which is later evaluated. The Mutate function consists in a two step process applied to the inputed solution. In the first step, with a probability equal to *mut_element_p* the *j*-th element of the solution is perturbed by adding a uniformly drawn value in the range [-*max_step*,*max_step*]. In the second step, with a probability equal to *mut_length_p* the length of the solution is modified by copying or removing (with equal probability) an element of the solution. Before returning the new solution, a *validation* process is performed to assure that the mutated solution is valid (i.e. its values complies with the restrictions).

Next, the fitness of the original solution and the mutated one are compared. If the fitness of the mutated is less or equal than the original solution, the mutated replaces the original solution.

As the last part of the evolutionary process, a **SelfAdapting** step is performed to improve the performance of the evolutionary process [43]. Particularly, if the fitness of the mutated solution improves the original one, then the *mut_element_p* and *mut_length_p* values are multiplied by 1.5, in other case these probabilities are divided by 4 [43]. In other words, if we are not improving, we narrow the local search space. On the contrary, while the solutions are improving (in terms of the fitness), we widen the local search space.

Finally, the evolved solution is evaluated (using *final_epochs* to feed the number of epochs of the training process) and returned.

5. Experimental study

We implemented our proposal in Python 3, using the DL optimization library **dlopt** [44], and the DL frameworks **keras** [45] and **tensorflow** [46]. Then, we (*i*) selected a data set to test our proposal, (*ii*) optimized an RNN to tackle the referred problem, (*iii*) compared our predictions against the state-of-the-art of urban waste containers filling level prediction, and (*iv*) studied the suitability of the solutions found to predict under uncertainty.

5.1 Data set: filling level of containers

The data set analyzed in this article is the one used in [17, 28], a real case study of an Andalusian city (Spain), where we highlight the benefits of our approach, being effective and realistic at the same time. Our case study considers 217 paper containers from the metropolitan area of a city. The choice of an instance of recycling waste (paper) is more attractive than a organic waste collection to show the quality of our approach because most paper containers do not need to be collected everyday like the organic waste, so they have a high variability in collection frequency.

In order to study the reliability of our approach under uncertainty we propose a synthetic benchmark of instances derived from the original data set. We selected a percentage *p* of random days where the filling data of all containers have errors, which may come from a) sensors errors or b) the loss of the data. From these two source of errors, we generate two types of instances. To represent the former source of errors (a) we generate random values between 0 and 100 to fix errors in data, we call it *random*. For the latter one (b) we use zeros to represent the loss of data, so we call it *zeros*. Combining the percentage of days with errors ($p = 5, 10, 20$) and the type of errors (zeros or random) we generate 6 synthetic instances.

5.2 RNN optimization

We executed 30 independent times our deep neuroevolutionary algorithm considering the combinatorial search space defined in Table 1, using the data set described above, the parameters defined in Table 2, and a fixed *dropout* equal to 0.5. We use an 80% of the data to train the networks and the remainder data to test their performance (i.e., computing the fitness).

Table 1 RNN optimization search space

Parameter	Value
min_look_back	2
max_look_back	30
min_neurons_per_layer	10
max_neurons_per_layer	300
min_hidden_layers	1
max_hidden_layers	8

The initial setup of the algorithm is taken from the related literature [14]. Considering that our proposal performs a self-adapting step, we did not perform a tuning of the parameters of the algorithm.

Table 3 summarizes the results obtained. The MAE, the mean squared error (MSE), the total number of LSTM cells, the look back, and the number of recurrent

Table 2 ES parameters configuration

Parameter	Value
mut_element_p	0.2
evaluation_epochs	10
mut_length_p	0.2
final_epochs	1000
max_step	15
max_evaluations	100

layers correspond to the statistics computed over the final solutions (30 RNN trained). We will refer to the solution returned by the algorithm as *solution*. The time corresponds to the statistics computed over the total time, i.e. the sum of the computation time of all the architectures evaluated, including the solution. The time is presented in minutes.

The results show that the algorithm is *robust* in regard to the MAE (and the MSE), however there is a noticeable variation in the architectures and in the time needed to compute a solution. We analyze the solutions and all the architectures evaluated during the optimization to get insights into the relation between the architecture and the error. Figure 1a presents the architectures (number of LSTM cells and layers) of the solutions along with their respective MAE and Figure 1b shows the same for all architectures evaluated. A small MAE (a darker dot) is desirable. It is important to remark that the MAE presented in both figures is *not comparable*, because in both cases the number of training epochs is different, therefore the results are expected to differ (at least in their magnitude).

It is quite interesting that the solutions are very diverse (see Figure 1a), and that most of them use less than 500 LSTM cells. This is more interesting if we consider that the maximum allowed number of LSTM cells given the problem restrictions (see Table 1) is equal to 2400 and that many architectures evaluated have more than 500 LSMT cells (see Figure 1b).

To continue with our analysis, we ranked all the architectures evaluated (excluding the solutions) into deciles and selected the top one (i.e. the best architectures evaluated). Then we plot the density distribution of the number of recurrent layers (see Figure 2a) and of the total number of LSTM cells (see Figure 2b). We also plot the density distribution of the solutions in both figures. The results show that both densities are relatively similar, therefore we intuit that there is an *archetype* that better suits to the problem. However, further analysis is required to validate this intuition.

5.3 Prediction benchmark

In order to continue with the evaluation of our proposal, we benchmark the predictions made by the RNN against the results published in [17, 28]. In order to compare the approaches we compute the "mean absolute error in the filling predictions of the next month" (MM) using the solutions given by our algorithm, i.e. we predict a whole month using an RNN and summed up the predictions per container, then we compute the mean absolute difference between the predicted values and the ground truth. Table 4 summarizes the results of the MM computed using the solutions. Note that the MM results are better than the MAE (see Table 3).

We selected the median solution (in regard to the MM) and compared the results against the ones presented in [17, 28]. Table 5 presents the benchmark in terms of the prediction error. In that previous work, the authors proposed three time series algorithms used for forecasting the fill level for all containers. Particularly, they used techniques based on Linear Regression (LR), Gaussian Processes (GP), and Support Vector Machines for Regression called SMReg.

The results indicate that our proposal exceeds its competitors. Moreover, we performed a non-parametric Friedman's Two-Way Analysis of Variance Ranks Test that revealed RNN as the best algorithm, followed by the algorithm based on GP, the LR, and the SMReg as last algorithm in the comparison. Regarding the statistical significant differences, the values have been adjusted by the Bonferroni correction for multiple comparisons. There are significant differences between each pair of algorithms except for the particular comparison between LR and SMReg. Thus, the RNN is significantly the most competitive method according to the MM.

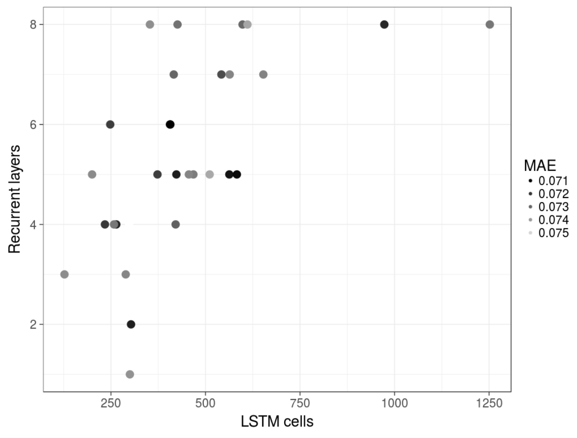
Finally, to relate the results presented in this subsection (see Table 4) to the ones presented in the previous subsection (see Table 3) we plotted the relation between the MAE and the MM (please refer to Figure 3). The figure also includes the architecture of the solutions (number of LSTM cells and number of recurrent layers). Something that caught our attention is that there is not an apparent linear relation between both metrics presented in the plot, however the summarized results presented for both metrics (see Tables 3 and 4) are robust in regard to the referred error measurement.

5.4 Prediction under uncertainty

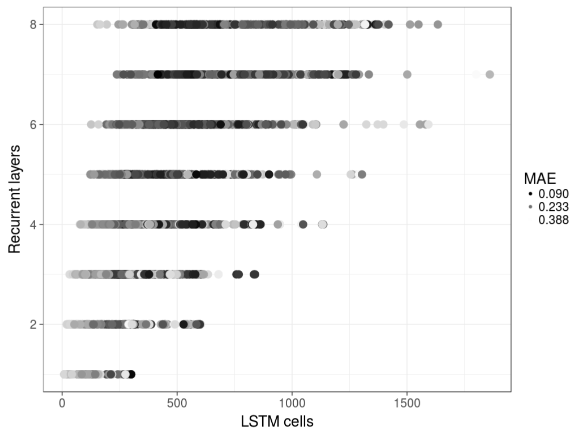
Following up with our experimentation, we studied the reliability of the solutions found. Particularly, we re-trained (Section 5.2) the solutions found (30 RNNs) using the synthetic dataset described previously.

Table 3 ES-based RNN optimization results

	MAE	MSE	LSTM cells	Look back	Rec. layers	Time [m]
Mean	0.073	0.014	450.667	5.933	5.433	96.866
Median	0.073	0.014	419.500	5.000	5.000	70.049
Min	0.071	0.013	127.000	2.000	1.000	33.216
Max	0.076	0.015	1252.000	16.000	8.000	405.339
Sd	0.001	0.000	227.661	3.648	1.906	75.488

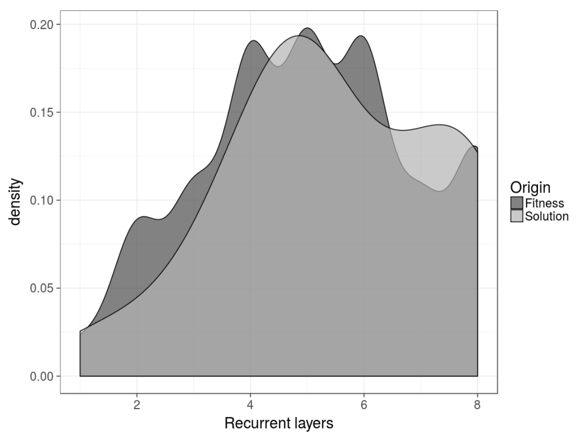


(a) Solution

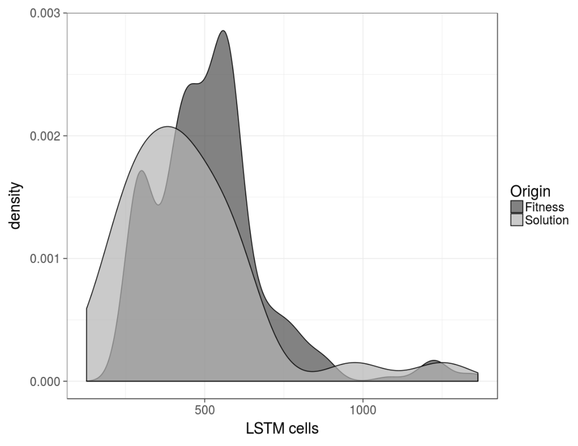


(b) Fitness

Figure 1 Architectures evaluated during the optimization process



(a) Deepness



(b) Gross size

Figure 2 The *best* solutions evaluated (fitness) compared to the final solutions

Table 4 MM statistics computed for the RNN solutions

	RNN
Mean	0.030
Median	0.028
Min	0.027
Max	0.043
Sd	0.004

Table 5 Prediction error of the compared methods

Method	Error
RNN	**0.028**
Gaussian Processes	0.038
Linear Regression	0.074
SMReg	0.095

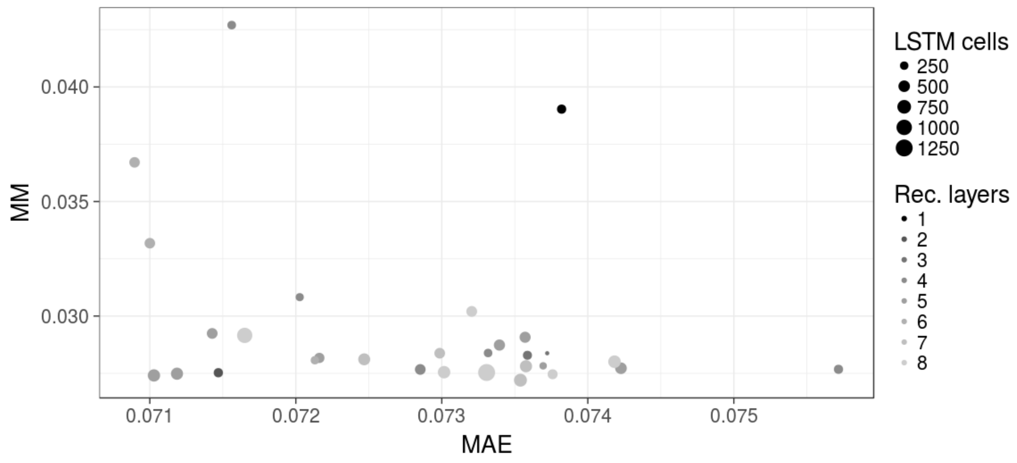


Figure 3 Relation between the MAE and the MM

Tables 6, 7 and 8 summarize the results of the reliability benchmark. At a glance, we notice that the overall results worsen as the uncertainty increases, as it is expected. Basically, the *quality* of the data has a direct impact in the accuracy of the predictions.

We also observed that losing data (i.e. replacing measurements with a zero) is not as important as having random errors. In other words, it is preferable to have a missing data (non-functioning sensor) than having an imprecise measurement (or faulty sensor). This particular insight presents a new challenge (or problem) to real waste management companies, because it is clear that a non-functioning sensor is easy to found, however a faulty one might be hard to detect.

Table 6 Reliability benchmark (5%)

	Random 5%		Zeros 5%	
	MAE	**MSE**	**MAE**	**MSE**
Mean	0.084	0.015	0.075	0.014
Median	0.081	0.014	0.074	0.014
Min	0.075	0.014	0.067	0.011
Max	0.113	0.021	0.097	0.019
Sd	0.008	0.002	0.006	0.001

Table 7 Reliability benchmark (10%)

	Random 10%		Zeros 10%	
	MAE	**MSE**	**MAE**	**MSE**
Mean	0.099	0.018	0.076	0.015
Median	0.092	0.016	0.075	0.014
Min	0.087	0.015	0.070	0.013
Max	0.215	0.063	0.098	0.022
Sd	0.023	0.009	0.007	0.002

In order to compare the predictions under uncertainty against the predictions made by our competitors (Table 5), we selected a solution per combination (random/zeros and percentage) whose MAE is equal to the median and we computed the MM. Table 9 presents the described results. As expected, the results show that adding uncertainty to the data has a negative impact on the MM. Moreover, a missing datum (zeros) has less impact than a random noisy datum. On the other hand, if the uncertain data represent less than the 5%, the RNN still beats all its competitors (Table 5). Note that the results shown in Table 5 do not consider uncertain data.

Table 8 Reliability benchmark (20%)

	Random 20%		Zeros 20%	
	MAE	**MSE**	**MAE**	**MSE**
Mean	0.118	0.023	0.080	0.016
Median	0.114	0.021	0.076	0.015
Min	0.093	0.016	0.069	0.014
Max	0.186	0.047	0.116	0.029
Sd	0.018	0.007	0.011	0.003

In order to gain insights into the relation between the performance and the architecture, specially in regard to the variation of the uncertainty, we computed the Pearson correlation between the MAE and the architecture definition of each solution. Particularly, we computed the correlation between the total number of LSTM cells, the look back, and the number of recurrent layers, and the MAE, MAE with a 5% missing (zeros) or faulty data (random). Table 10 presents the correlations computed. The results show that there is a small correlation between the variables. Therefore, further analysis is needed to conclude that there is a relation between the performance and the architecture in this case (adding uncertainty to the dataset). Please refer to Table 11 in Appendix for a detailed version of the results.

Table 9 Reliability benchmark (up to 20% uncertainty)

Uncertainty	Missing or noisy data		
	5%	**10%**	**20%**
Random	3.72	6.43	7.44
Zeros	2.88	2.98	4.24

Table 10 Architecture and MAE correlation

Corr	**MAE**	**Z_5**	**R_5**
LSTM cells	0.138	0.118	-0.036
Look back	-0.218	-0.130	-0.093
Rec. layers	0.237	0.205	0.092

Appendix

Table 11 presents the detailed results of the experimentation. Particularly, LSTM stands for the total number of LSTM cells, LB is the look back, and RL is the number of recurrent stacked layers.

Table 11 Detailed results of the experimentation

No	LSTM	LB	RL	MAE	MSE	Time [m]	$Zeros_5$		$Random_5$		$Zeros_{10}$		$Random_{10}$		$Zeros_{20}$		$Random_{20}$	
							MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
1	511	3	5	.074	.014	203.178	.07	.013	.079	.014	.07	.013	.09	.015	.071	.014	.121	.022
2	598	3	8	.073	.014	328.807	.097	.019	.101	.02	.094	.019	.102	.02	.1	.02	.11	.022
3	200	3	5	.074	.014	145.541	.078	.016	.077	.014	.089	.019	.088	.015	.077	.016	.111	.02
4	468	5	5	.073	.013	320.292	.069	.012	.079	.014	.073	.014	.087	.015	.071	.014	.114	.02
5	234	5	4	.071	.013	241.278	.07	.013	.076	.014	.081	.016	.088	.015	.091	.02	.116	.02
6	127	4	3	.071	.013	158.371	.078	.014	.088	.016	.07	.013	.087	.015	.072	.014	.111	.019
7	289	4	3	.073	.014	149.882	.075	.013	.082	.014	.077	.014	.102	.018	.078	.015	.12	.022
8	611	4	8	.074	.014	370.989	.07	.013	.08	.014	.079	.016	.104	.019	.076	.015	.108	.019
9	653	2	7	.074	.014	246.725	.082	.015	.081	.015	.085	.017	.215	.063	.102	.023	.11	.021
10	426	14	8	.071	.014	464.491	.074	.013	.082	.014	.073	.013	.095	.016	.074	.014	.121	.022
11	406	16	6	.071	.014	453.58	.079	.015	.086	.017	.081	.015	.109	.021	.085	.017	.186	.047
12	407	5	6	.069	.012	427.573	.072	.013	.097	.016	.073	.014	.089	.015	.084	.017	.098	.017
13	564	2	7	.074	.014	258.821	.067	.011	.083	.014	.071	.013	.097	.017	.079	.016	.128	.025
14	353	3	8	.074	.014	262.942	.074	.014	.083	.015	.071	.014	.088	.015	.074	.014	.116	.021
15	416	9	7	.073	.014	368.101	.076	.014	.081	.014	.073	.014	.095	.017	.071	.014	.11	.019
16	303	5	2	.07	.012	242.58	.072	.013	.084	.015	.076	.014	.119	.028	.076	.014	.135	.03
17	423	10	5	.072	.013	480.457	.072	.013	.079	.014	.071	.013	.092	.016	.081	.017	.093	.016
18	1252	7	8	.073	.014	794.752	.077	.015	.079	.015	.078	.015	.089	.015	.073	.015	.111	.022
19	298	10	4	.075	.014	237.645	.069	.013	.08	.014	.075	.014	.091	.016	.073	.014	.107	.019
20	264	6	4	.071	.014	304.879	.075	.014	.079	.014	.098	.022	.091	.016	.116	.029	.109	.019
21	300	2	1	.072	.013	203.085	.073	.013	.08	.014	.073	.014	.094	.016	.076	.015	.163	.045
22	258	5	4	.073	.014	182.12	.083	.015	.093	.017	.087	.016	.105	.02	.092	.018	.123	.024
23	563	5	5	.069	.012	632.708	.074	.014	.095	.02	.074	.013	.092	.016	.076	.014	.112	.02
24	456	4	5	.072	.013	441.383	.074	.014	.075	.014	.072	.014	.091	.016	.074	.015	.113	.022
25	542	4	7	.072	.014	284.908	.075	.014	.081	.015	.073	.013	.092	.016	.073	.015	.115	.021
26	373	5	5	.072	.012	303.125	.073	.014	.082	.014	.076	.015	.092	.016	.075	.015	.103	.018
27	973	8	8	.072	.014	4158.109	.073	.013	.078	.014	.072	.014	.101	.019	.075	.015	.109	.019
28	421	6	4	.072	.014	196.231	.073	.013	.079	.014	.071	.014	.091	.016	.073	.015	.117	.021
29	248	14	6	.072	.014	360.562	.069	.012	.079	.014	.085	.017	.088	.015	.069	.014	.116	.02
30	583	5	5	.07	.012	467.417	.086	.016	.113	.021	.084	.015	.094	.017	.084	.016	.125	.025
Mean	451	6	5	.072	.013	456.351	.075	.014	.084	.015	.078	.015	.099	.018	.08	.016	.118	.023
Median	419	5	5	.072	.014	304.002	.074	.014	.081	.014	.075	.014	.092	.016	.076	.015	.114	.021
Max	1252	16	8	.075	.014	4158.109	.097	.019	.113	.021	.098	.022	.215	.063	.116	.029	.186	.047
Min	127	2	1	.069	.012	145.541	.067	.011	.075	.014	.07	.013	.087	.015	.069	.014	.093	.016
Sd	228	4	2	.001	.001	714.285	.006	.001	.008	.002	.007	.002	.023	.009	.011	.003	.018	.007

6. Conclusions and future work

Deep neuroevolution has emerged as a promising field of study and is growing rapidly. Particularly, the use of Evolutionary Algorithms to tackle the hyper-parametrization optimization problem is showing unprecedented results, not only in terms of the performance of the designed networks, but also in terms of the reduction of the computational resources needed (e.g., the configurations are evaluated using a heuristic, therefore not all configurations are actually trained [47, 48]).

In this study, we present a deep neuroevolutionary algorithm to optimize the architecture of an RNN (given a problem). We test our proposal using the filling level of 217 waste containers located in Andalusia, Spain, recorded over a whole year and benchmark our results against the state-of-the-art of filling level prediction. Our experimental results show that an "appropriate" selection of the architecture improves the performance (in terms of the error) of an RNN and that our prediction results exceeds all its competitors.

In regards to the quality of the predictions under uncertainty, the result show that the quality gets worse as the percentage of missing or faulty data increases. Nevertheless, the median RNN (not the best) is able to outperform all its competitors (using correct data) even when the RNN uses an instance which has 10% of missing data. In addition, by analyzing in detail the RNN results' under uncertainty, we conclude that it is preferable to have missing data than imprecise data coming from a faulty sensor. This fact should be considered when we receive an outlier from a sensor.
As future work, we propose to explore train-free approaches for evaluating a network configuration. Specifically, we propose to study the use of the MAE random sampling [47, 48] to compare RNN architectures, aiming to reduce the computational power and the time needed to find an appropriate architecture.

7. Acknowledgements

This research was partially funded by Ministerio de Economía, Industria y Competitividad, Gobierno de España, and European Regional Development Fund grant numbers TIN2016-81766-REDT (`http://cirti.es`), and TIN2017-88213-R (`http://6city.lcc.uma.es`). European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 799078. Universidad de Málaga, Campus Internacional de Excelencia Andalucía TECH.

References

[1] T. Bakici, E. Almirall, and J. Wareham, "A smart city initiative: the case of barcelona," *Journal of the Knowledge Economy*, vol. 4, no. 2, pp. 135–148, 2013.

[2] P. Ghisellini, C. Cialani, and S. Ulgiati, "A review on circular economy: The expected transition to a balanced interplay of environmental and economic systems," *Journal of Cleaner Production*, vol. 114, pp. 11–32, 2016.

[3] A. Tukker, "Product services for a resource-efficient and circular economy - a review," *Journal of Cleaner Production*, vol. 97, pp. 76–91, 2015.

[4] J. Teixeira, A. P. Antunes, and J. P. de Sousa, "Recyclable waste collection planning—a case study," *European Journal of Operational Research*, vol. 158, no. 3, pp. 543–554, nov 2004.

[5] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 97 – 116, 2017.

[6] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford university press, 1995.

[7] J. Ferrer, J. García, E. Alba, and F. Chicano, "Intelligent testing of traffic light programs: Validation in smart mobility scenarios," *Mathematical Problems in Engineering*, vol. 2016, pp. 1–19, 2016. [Online]. Available: http://www.hindawi.com/journals/mpe/2016/3871046/

[8] J. García, J. Ferrer, and E. Alba, "Optimising traffic lights with metaheuristics: Reduction of car emissions and consumption," in *International Joint Conference on Neural Networks*, 2014, pp. 48–54.

[9] R. Massobrio, J. Toutouh, S. Nesmachnow, and E. Alba, "Infrastructure deployment in vehicular communication networks using a parallel multiobjective evolutionary algorithm," *International Journal of Intelligent Systems*, vol. 32, no. 8, pp. 801–829, 2017.

[10] S. Nesmachnow, D. Rossit, and J. Toutouth, "Comparison of multiobjective evolutionary algorithms for prioritized urban waste collection in montevideo, uruguay," *Electronic Notes in Discrete Mathematics*, 2018.

[11] J. Toutouh, D. Rossit, and S. Nesmachnow, "Computational intelligence for locating garbage accumulation points in urban scenarios," in *International Conference on Learning and Intelligent Optimization, LION 12*, 2018, pp. 1–15.

[12] D. G. Rossit, S. Nesmachnow, and J. Toutouh, "Municipal solid waste management in smart cities: facility location of community bins," in *Congreso Iberoamericano de Ciudades Inteligentes (ICSC-CITIES 2018)*, 2018, pp. 1–14.

[13] A. Camero, J. Arellano, and E. Alba, "Road map partitioning for routing by using a micro steady state evolutionary algorithm," *Engineering Applications of Artificial Intelligence*, vol. 71, pp. 155–165, 2018.

[14] A. Camero, J. Toutouh, D. H. Stolfi, and E. Alba, "Evolutionary deep learning for car park occupancy prediction in smart cities," in *Learning and Intelligent OptimizatioN Conference LION*, 2018.

[15] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 303–317, 2005.

[16] A. Camero, J. Toutouh, J. Ferrer, and E. Alba, "Waste generation prediction in smart cities through deep neuroevolution," in *Smart Cities. ICSC-CITIES 2018*, vol. 978, 2019, pp. 192–204.

[17] J. Ferrer and E. Alba, "(bin-ct): Urban waste collection based in predicting the container fill level," jul 2018. [Online]. Available: http://arxiv.org/abs/1807.01603

[18] B. J. Garvin, M. Cohen, and M. B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 61–102, 2011.

[19] S. Sahoo, S. Kim, B. I. Kim, B. Kraas, and A. Popov Jr., "Routing optimization for waste management," *Interfaces*, vol. 35, no. 1, pp. 24–36, 2005.

[20] L. Q. Dat, D. T. Truc, S. Y. Chou, and V. F. Yu, "Optimizing reverse

logistic costs for recycling end-of-life electrical and electronic products," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6380–6387, 2012.

[21] A. Z. Alagöz and G. Kocasoy, "Improvement and modification of the routing system for the health-care waste collection and transportation in istanbul," *Waste Management*, vol. 28, no. 8, pp. 1461–1471, 2008.

[22] J. Beliën, L. De Boeck, and J. Van Ackere, "Municipal solid waste collection and management problems: A literature review," *Transportation Science*, vol. 48, no. 1, pp. 78–102, feb 2014.

[23] L. Xu, P. Gao, S. Cui, and C. Liu, "A hybrid procedure for msw generation forecasting at multiple time scales in xiamen city, china," *Waste management*, vol. 33, no. 6, pp. 1324–31, jun 2013.

[24] C. Cole, M. Quddus, A. Wheatley, M. Osmani, and K. Kay, "The impact of local authorities' interventions on household waste collection: a case study approach using time series modelling," *Waste management*, vol. 34, no. 2, pp. 266–72, feb 2014.

[25] D. V. Tung and A. Pinnoi, "Vehicle routing-scheduling for waste collection in hanoi," *European Journal of Operational Research*, vol. 125, no. 3, pp. 449–468, 2000.

[26] J. Sniezek and L. Bodin, "Using mixed integer programming for solving the capacitated arc routing problem with vehicle/site dependencies with an application to the routing of residential sanitation collection vehicles," *Annals of Operations Research*, vol. 144, no. 1, pp. 33–58, apr 2006.

[27] L. Bodin, A. Mingozzi, R. Baldacci, and M. Ball, "The rollon–rolloff vehicle routing problem," *Transportation Science*, vol. 34, no. 3, pp. 271–288, 2000.

[28] J. Ferrer and E. Alba, "Bin-ct: sistema inteligente para la gestión de la recogida de residuos urbanos," in *International Greencities Congress*, 2018, pp. 117–128.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[30] S. Haykin, *Neural networks and learning machines*. Pearson, 2009, vol. 3.

[31] D. Rumelhart, G. E. Hinton, and R. j. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep. No. ICS-8506, 1985.

[32] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach*. GMD, 2002, vol. 5.

[33] R. Reed, R. Marks, and S. Oh, "Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 529–538, 1995.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International Conference on Machine Learning*, 2013, pp. 115–123.

[36] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[37] E. Alba and R. Martí, *Metaheuristic Procedures for Training Neural Networks*. Springer Science & Business Media, 2006.

[38] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[39] R. Miikkulainen and *et al.*, "Evolving deep neural networks," *arXiv preprint arXiv:1703.00548*, 2017. [Online]. Available: http://arxiv.org/abs/1703.00548

[40] G. Morse and K. O. Stanley, "Simple evolutionary optimization can rival stochastic gradient descent in neural networks," in *Proc. of the Genetic and Evolutionary Computation Conf. 2016*, 2016, pp. 477–484.

[41] X. Su, X. Yan, and C. L. Tsai, "Linear regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] C. Doerr, "Non-static parameter choices in evolutionary computation," in *Genetic and Evolutionary Computation Conference Companion*, 2017.

[44] A. Camero, J. Toutouh, and E. Alba, "(dlopt): Deep learning optimization library," *arXiv preprint arXiv:1807.03523*, july 2018.

[45] F. Chollet and *et al*, "Keras," https://keras.io, 2015.

[46] M. Abadi and *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th (USENIX) Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[47] A. Camero, J. Toutouh, and E. Alba, "Comparing deep recurrent networks based on the mae random sampling, a first approach," in *Conference of the Spanish Association for Artificial Intelligence (CAEPIA) 2018*, 2018, pp. 1–10.

[48] A. Camero, J. Toutouh, and E. Alba, "Low-cost recurrent neural network expected performance evaluation," *arXiv preprint arXiv:1805.07159*, may 2018.