applied sciences

Article

A Novel UCP Model Based on Artificial Neural Networks and Orthogonal Arrays

Nevena Rankovic [1,*], Dragica Rankovic [1], Mirjana Ivanovic [2] and Ljubomir Lazic [1]

[1] School of Computing, Union University, 11000 Belgrade, Serbia; drankovic@raf.rs (D.R.); ljlazic@raf.rs (L.L.)
[2] Faculty of Sciences, University of Novi Sad, 21000 Novi Sad, Serbia; mira@dmi.uns.ac.rs
* Correspondence: nrankovic@raf.rs

Abstract: Adequate estimation is a crucial factor for the implementation of software projects within set customer requirements. The use of Case Point Analysis (UCP) is the latest and most accurate method for estimating the effort and cost of realizing software products. This paper will present a new, improved UCP model constructed based on two different artificial neural network (ANN) architectures based on Taguchi Orthogonal Vector Plans. ANNs are an exceptional artificial intelligence tool that have been proven to be reliable and stable in this area of software engineering. The Taguchi method of Orthogonal Vector Plans is an optimization method that reduces the number of iterations required, which significantly shortens estimation time. The goal is to construct models that give a minimum magnitude relative error (MRE) value concerning previous approaches and techniques. A minimum number of iterations (less than six) and a minimum value of MMRE (less than 10%) have been achieved. The obtained results significantly improve the accuracy and reliability of estimating the effort and cost involved in the implementation of software projects.

Keywords: software development estimation; Use Case Point Analysis; orthogonal array-based experiment; artificial neural networks design

Citation: Rankovic, N.; Rankovic, D.; Ivanovic, M.; Lazic, L. A Novel UCP Model Based on Artificial Neural Networks and Orthogonal Arrays. *Appl. Sci.* **2021**, *11*, 8799. https://doi.org/10.3390/app11198799

Academic Editor: Rocco Furferi

Received: 22 August 2021
Accepted: 17 September 2021
Published: 22 September 2021

1. Introduction

The most important activity in the software development process is the assessment of effort, which includes assessing the time and money required for the software project to be successfully completed. Project development time is critical, both for project clients and project implementers. The amount of money required for investment in a project influences whether the project will be started or not, and whether it will end within the set framework. Software companies use a variety of software tools and services to meet customer requirements. Many methods measure the size of software, its complexity, and the time needed to build it. All methods can be divided according to whether they are parametric or nonparametric [1–3]. Within this division, there are several different approaches, three of which are the most commonly used:

1. An approach based on analyzing the number of source code lines and estimating the effort required to implement the project. The most commonly used model of this approach is the Constructive Cost Model (COCOMO2000) [4,5].
2. An approach based on the analysis of functional points to estimate the magnitude of the functionality of the software being developed. Within this approach, two models were initially distinguished: IFPUG (created by the International Function Point Users Group) [6] and Mk II (Mark II) [7]. Subsequently, within the IFPUG model, the following were developed: NESMA NESMA (created by the Netherlands Software Metrics Association) [8], IFPUG 4.1, and COSMIC FP (the COmmon Software Measurement International Consortium function point) [9].
3. An approach based on the analysis of users and use cases for the assessment of effort. Within this approach, the most commonly used models are: COBRA (COnstraint-Based Reconstruction and Analysis) [10] and UCP (Use Case Point Analysis) [11].

1.1. Use Case Point Analysis

The UCP method is most often used for estimating the real size of a software project. This method of estimating the effort required to implement a particular system considers the use cases of the system. It analyzes system users and different scenarios to adequately assess the effort required. It uses twenty-one parameters for assessment, of which thirteen parameters are technical characteristics of the system, and the remaining eight are environmental factors [12–15].

The technical characteristics of the system being evaluated are as follows: distributed system, system response time, efficiency, complexity of internal processes, code reuse, ease of installation, ease of use, transfer to other platforms, system maintenance, competitiveness, parallel processing, security requirements, access to external systems, and end-user training.

The following environmental factors are assessed: compliance with the used development process, experience with applications, experience in object-oriented technologies, ability of the chief analyst, team motivation, stability requirements, adaptation of working hours of team members, and complexity of the programming language.

The system users and use cases are used together to determine the real size of the UCP method. The users of the system are divided into three groups: simple (depending on the interaction with the system, they are assigned a weight factor of (1); medium (depending on internal/external communication, they are assigned a weight factor of (2); and complex (depending on the complexity of interactions, they are assigned a weight factor of (3). There are also three categories of use cases that are defined based on the number of transactions executed (number of users and system messaging): simple (for less than three transactions, with weighting factor of five assigned); medium (from 4 to 7 transactions, with weighting factor of 10 assigned); and complex (more than eight transactions, with a weighting factor of 15 assigned).

The size of the system is defined based on a six-dimensional vector whose elements represent the complexity of the previously mentioned users and the cases of users in the system.

The estimated value is calculated based on the formulas established by G. Karner [16]:

UAW (Unadjusted Actor Weight)—this input value is a functional point that can determine the level of complexity of system users. Users can be simple system operators or other external systems. Each user is ranked according to their level of complexity and can be: *Simple*, *Average*, and *Complex* (1)–(4).

$$SimpleA = \sum(SimpleActor) * SimpleWeight,\ \text{where } SimpleWeight = 1; \tag{1}$$

$$AverageA = \sum(AverageActor) * AverageWeight,\ \text{where } AverageWeight = 2; \tag{2}$$

$$ComplexA = \sum(ComplexActor) * ComplexWeight,\ \text{where } ComplexWeight = 3; \tag{3}$$

$$\text{UAW} = SimpleA + AverageA + ComplexA \tag{4}$$

UUCW (Unadjusted Use Case Weight)—this input value is a functional point that can determine the level of complexity of use cases. Each use case is ranked according to its level of complexity and can be: Simple, Average and Complex (5)–(8).

$$SimpleUUCW = \sum(SimpleUCW) * SimpleWeight,$$

where

$$SimpleWeight = 5, (\text{transactions} \leq 3, \text{analysis classes} < 5) \tag{5}$$

$$AverageUUCW = \sum(AverageUCW) * AverageWeight,$$

where

$$AverageWeight = 10, (4 \leq \text{transactions} \leq 7, 5 \leq \text{analysis classes} \leq 10) \tag{6}$$

$$ComplexUUCW = \sum(ComplexUCW) \times ComplexWeight,$$

where

$$ComplexWeight = 15; \text{(transactions} > 7\text{, analysis classes} \geq 10) \tag{7}$$

$$\text{UUCW} = SimpleUUCW + AverageUUCW + ComplexUUCW \tag{8}$$

UUCP (Unadjusted Use Case Points) is determined by following Equation (9):

$$\text{UUCP} = \text{UUCW} + \text{UAW} \tag{9}$$

TCF (Technical Complexity Factor) is an estimate of the technical complexity of the system and can be described by the following Equations (10) and (11):

$$\text{TCF} = 0.6 + (0.01 \times \text{FactorT}) \tag{10}$$

$$\text{FactorT} = \sum Weight * AssignedValue, \tag{11}$$

where *AssignedValue* is from 0 to 5 and represents a technical factor of the estimated process.

ECF (*Environmental Complexity Factor*) is one of the factors affecting the size of the project expressed in Use Case points. It is calculated according to the following Equations (12) and (13):

$$\text{ECF} = 1.4 + (-0.03 \times \text{FactorE}) \tag{12}$$

$$\text{FactorE} = \sum Weight * AssignedValue, \tag{13}$$

where *AssignedValue* is from 0 to 5 and represents an environmental factor of the estimated process.

AUCP (*Adjusted Use Case Point*) is the final size of the system expressed in Use Case points and is calculated as follows (14):

$$\text{AUCP} = \text{UUCP} \times \text{TCF} \times \text{ECF} \tag{14}$$

Real effort is represented, via the UCP approach, as a six-dimensional vector, where its value is calculated as the norm of the vector as follows (15) and (16):

$$\text{UCP} = (\text{UAW, UUCW, UUCP, TCF, ECF, AUCP}) \tag{15}$$

$$\|\overrightarrow{UCP}\| = UAW + UUCW + UUCP + TCF + ECF + AUCP \tag{16}$$

Real effort is represented, via the UCP approach, as a four-dimensional vector, where its value is calculated as the norm of the vector as follows (17) and (18):

$$\text{UCP} = (\text{UAW, UUCW, TCF, ECF}) \tag{17}$$

$$\|\overrightarrow{UCP}\| = UAW + UUCW + TCF + ECF \tag{18}$$

where UUCP= UAW+ UUCW, and AUCP = UUCP $\times$ TXF $\times$ ECF.

In both cases, Real Effort is obtained as the norm of the UCP vector and represents the real functional size or number of points of use cases. This method is currently most commonly used to assess effort [17], although it is not standardized within ISO standards such as the previous two.

1.2. Taguchi Orthogonal Arrays

This paper aims to present a new, improved UCP model constructed using artificial neural networks based on Taguchi's orthogonal vector plans. ANNs represent a tremendous artificial intelligence tool and are often used in combination with parametric methods [18,19]. Using various ANN architectures, we can arrive at a fast, accurate, and reliable estimate of the effort and cost required to develop a project. Each ANN architecture is based on robust design methods, i.e., Taguchi methods of orthogonal vector plans. A robust method design implies meeting the prescribed criteria when planning and implementing software. Taguchi's orthogonal vector plans are based on a unique set of Latin Squares [20,21]. The discovery of orthogonal vector plans and their application minimizes crucial parameters for the project's successful development. The impact parameters are not duplicated, which achieves a much faster estimation of the efforts and costs of a particular project. This design method is based on a factorial experiment realized only with all possible experimental combinations of parameter values. The construction of, e.g., the artificial neural ANN-L36prim architecture using Taguchi's orthogonal plan achieves a higher convergence rate, reducing the time and number of iterations required to achieve the minimum MMRE. The number of iterations required for the implementation of the Full Factorial Plan (FFP) within a robustly designed experiment is $N = L^P$ (for example, when three levels with 16 parameters are used according to FFP, it is necessary to execute $N = 3^{11}2^{4}3^{1}$ = 8,503,056.00 experiments). Using the Taguchi orthogonal vector plan with 16 parameters (weight coefficients) on three levels, only 36 experiments are necessary. The Taguchi method of robust design reduces experiments by 99.99% $(0.9999957662\ldots = 1 - (36/8\,503\,056))$. It is expected that the new, improved UCP approach constructed based on different ANN architectures that are in line with the Taguchi Orthogonal Plans will give better results than the previously proposed UCP model.

This article is structured as follows: Section 2 provides an overview of previous studies that applied UCP for effort estimation in software projects. Section 3 explains the new, improved UCP model with the methodology used. Section 4 discusses the obtained results. The concluding remarks are given in the last section.

2. Related Work

The UCP method is the latest and the most widespread method for estimating the effort and costs involved in the realization of software products. The most significant advantage of this method is that the lowest values of relative error in estimation are obtained—between 20% and 35%. The best result achieved by this method is an error value of about 10% [22]. Many researchers [17,23–25] have combined this method with other parametric models and models of artificial intelligence. In a previous study [26], the UCP method was used for the estimation of size and effort for mobile applications. Android mobile applications were considered as a case study, and modified UCP was also proposed. The authors of [14] proposed a framework for UCP-based techniques to promote reusability in the development of software applications. The results showed that the framework met five quality attributes, and that it can be used in the early stages of software development. In [27], a systematic review of studies with the best practices in terms of use case point (UCP) and expert judgment-based effort estimation techniques was given. The authors of [28] presented the results of four different models that include the UCP method and Neuro-Fuzzy logic. It was concluded that the Neuro-fuzzy logic model using revised use case points and modified environmental gives the best fitting accuracy at an early stage compared to other models. In another relevant study [29], the authors compared the benefits of statistical analyses of effort estimation approaches for seven real-world software development projects. In addition, they contrasted a conventional Use Case points method with iUCP, an HCI (Human-centric)-enhanced model. Furthermore, they proposed an enhancement of the original iUCP effort estimation formula.

The critical decisions that defined the new, improved model within the UCP approach were as follows:

- Examination of the influence of two linearly dependent input values (UUCP and AUCP) on the change in the MMRE value;
- Comparative analysis of two different architectures of artificial neural networks and the obtained results;
- Division of the used dataset to a scale of 70:30, i.e., 70 projects from the selected dataset were used for the training process, while 30 were used for the testing process;

- Finding the most efficient methods of encoding and decoding input values, such as the fuzzification method;
- The requirement of a minimum number of performed experiments;
- Testing and validation on other datasets.

3. New, Improved UCP—Our Approach

For the new, improved UCP model, the following architectures and corresponding orthogonal vector plans are used:

1. UCP and ANN-L16

The first proposed architecture is denoted as ANN-L16. It consists of six input values, one hidden layer with two nodes, one output, and a total number of fifteen weighting factors (W_i, $i = \overline{1,\ 15}$), whose initial values are from the interval $[-1, 1]$. The Taguchi Orthogonal Array used in the construction of this proposed architecture contains two levels, L1 and L2 (Figure 1; Table 1) [2,3,30,31].

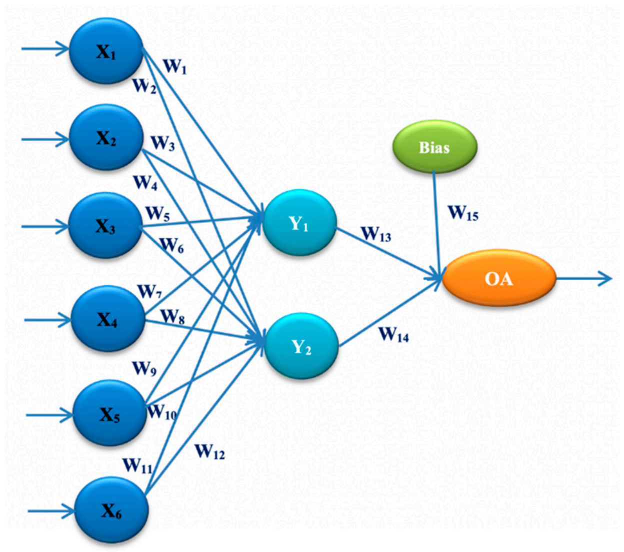

Figure 1. ANN architecture with one hidden layer (ANN-L16).

Table 1. Taguchi Orthogonal Array (L16 = 2^{15}).

ANN-L16	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L1	L1	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L2
ANN3	L1	L1	L1	L2	L2	L2	L2	L1	L1	L1	L1	L2	L2	L2	L2
ANN4	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L1	L1	L1	L1
ANN5	L1	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2
ANN6	L1	L2	L2	L1	L1	L2	L2	L2	L2	L1	L1	L2	L2	L1	L1
ANN7	L1	L2	L2	L2	L2	L1	L1	L1	L1	L2	L2	L2	L2	L1	L1
ANN8	L1	L2	L2	L2	L2	L1	L1	L2	L2	L1	L1	L1	L1	L2	L2
ANN9	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2
ANN10	L2	L1	L2	L1	L2	L1	L2	L2	L1	L2	L1	L2	L1	L2	L1
ANN11	L2	L1	L2	L2	L1	L2	L1	L1	L2	L1	L2	L2	L1	L2	L1
ANN12	L2	L1	L2	L2	L1	L2	L1	L2	L1	L2	L1	L1	L2	L1	L2
ANN13	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2	L1
ANN14	L2	L2	L1	L1	L2	L2	L1	L2	L1	L1	L2	L2	L1	L1	L2
ANN15	L2	L2	L1	L2	L1	L1	L2	L1	L2	L2	L1	L2	L1	L1	L2
ANN16	L2	L2	L1	L2	L1	L1	L2	L2	L1	L1	L2	L1	L2	L2	L1

2. UCP and ANN-L36prim

The second proposed architecture is denoted as ANN-L36prim. It consists of four input values, one hidden layer with three nodes, one output, and a total number of sixteen weighting factors (W_i, $i = \overline{1, 16}$), whose initial values are from the interval [−1, 0, 1]. The Taguchi Orthogonal Array used in the construction of this proposed architecture is combined, where the first eleven parameters and the last sixteenth parameter are with three levels, L1, L2, and L3, while the remaining four parameters are with two levels, L1 and L2 (Figure 2; Table 2) [2,3,30,31].

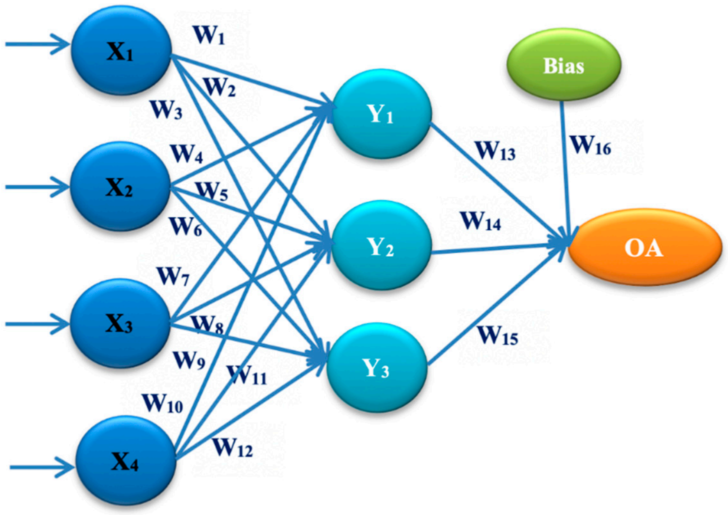

Figure 2. ANN architecture with one hidden layer (ANN-L36prim).

The experiment presented in this paper consists of three parts:

1. Training of two different ANN architectures constructed according to the corresponding Taguchi orthogonal vector plans (ANN-L16 and ANN36prim);
2. Testing of the ANN that gave the best results (the lowest MMRE value) in the first part of the experiment, for two proposed architectures on the same dataset;
3. Validation of the ANN that gave the best results (the lowest MMRE value) in the first part of the experiment, for each selected architecture, but using different datasets.

3.1. Data Sets Used in the UCP Approach

For the first and second part of the experiment, the Use Case Point Benchmark Dataset by Radek Silhavy (UCP Benchmark Dataset) [32] was used. In contrast, in the third part, the combined datasets, composed of projects of different industrial companies, were used. The results in Table 3 indicate a more homogeneous structure of the projects used in all three parts of the experiment, which can be concluded based on the standard deviation results in Table 4.

3.2. The Methodology Used within the Improved UCP Model

The appropriate methodology was selected for the experimental part of the UCP approach based on several trial experiments. The order of the experiment was constructed based on a robust design algorithm and is shown in Figure 3.

Table 2. Taguchi Orthogonal Array (L36prim = $3^{11}2^{4}3^{1}$).

ANN-L36prim	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L1	L1	L1	L1
ANN3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L1	L1	L1	L1
ANN4	L1	L1	L1	L1	L2	L2	L2	L2	L3	L3	L3	L3	L1	L2	L2	L1
ANN5	L1	L1	L1	L1	L3	L3	L3	L3	L2	L2	L2	L2	L1	L2	L2	L1
ANN6	L3	L3	L3	L3	L1	L1	L1	L1	L2	L2	L2	L2	L1	L2	L2	L1
ANN7	L1	L1	L2	L3	L1	L2	L3	L3	L1	L1	L1	L3	L2	L1	L2	L1
ANN8	L2	L2	L3	L1	L2	L3	L1	L1	L2	L3	L3	L1	L2	L1	L2	L1
ANN9	L3	L3	L1	L2	L3	L1	L2	L2	L3	L1	L1	L2	L2	L1	L2	L1
ANN10	L1	L1	L3	L2	L1	L3	L2	L3	L2	L1	L3	L2	L2	L2	L1	L1
ANN11	L2	L2	L1	L3	L2	L1	L3	L1	L3	L2	L1	L3	L2	L2	L1	L1
ANN12	L3	L3	L2	L1	L3	L2	L1	L2	L1	L3	L2	L1	L2	L2	L1	L1
ANN13	L1	L2	L3	L1	L3	L2	L1	L3	L3	L2	L1	L2	L1	L1	L1	L2
ANN14	L2	L3	L1	L2	L1	L3	L2	L1	L1	L3	L2	L3	L1	L1	L1	L2
ANN15	L3	L1	L2	L3	L2	L1	L3	L2	L2	L1	L3	L1	L1	L1	L1	L2
ANN16	L1	L2	L3	L2	L1	L1	L3	L2	L3	L3	L2	L1	L1	L2	L2	L2
ANN17	L2	L3	L1	L3	L2	L2	L1	L3	L1	L1	L3	L2	L1	L2	L2	L2
ANN18	L3	L1	L2	L1	L3	L3	L2	L1	L2	L2	L1	L3	L1	L2	L2	L2
ANN19	L1	L2	L1	L3	L3	L3	L1	L2	L2	L1	L2	L3	L2	L1	L2	L2
ANN20	L2	L3	L2	L1	L1	L1	L2	L3	L3	L2	L3	L1	L2	L1	L2	L2
ANN21	L3	L1	L3	L2	L2	L2	L3	L1	L1	L3	L1	L2	L2	L1	L2	L2
ANN22	L1	L2	L2	L3	L3	L1	L2	L1	L1	L3	L3	L2	L2	L2	L1	L2
ANN23	L2	L3	L3	L1	L1	L2	L3	L2	L2	L1	L1	L3	L2	L2	L1	L2
ANN24	L3	L1	L1	L2	L2	L3	L1	L3	L3	L2	L2	L1	L2	L2	L1	L2
ANN25	L1	L3	L2	L1	L2	L3	L3	L1	L3	L1	L2	L2	L1	L1	L1	L3
ANN26	L2	L1	L3	L2	L3	L1	L1	L2	L1	L2	L3	L3	L1	L1	L1	L3
ANN27	L3	L2	L1	L3	L1	L2	L2	L3	L2	L3	L1	L1	L1	L1	L1	L3
ANN28	L1	L3	L2	L2	L2	L1	L1	L3	L2	L3	L1	L3	L1	L2	L2	L3
ANN29	L2	L1	L3	L3	L3	L2	L2	L1	L3	L1	L2	L1	L1	L2	L2	L3
ANN30	L3	L2	L1	L1	L1	L3	L3	L2	L1	L2	L3	L2	L1	L2	L2	L3
ANN31	L1	L3	L3	L3	L2	L3	L2	L2	L1	L2	L1	L1	L2	L1	L2	L3
ANN32	L2	L1	L1	L1	L3	L1	L3	L3	L3	L3	L2	L2	L2	L1	L2	L3
ANN33	L3	L2	L2	L2	L1	L2	L1	L1	L3	L1	L3	L3	L2	L1	L2	L3
ANN34	L1	L3	L1	L2	L3	L2	L3	L1	L2	L2	L3	L1	L2	L2	L1	L3
ANN35	L2	L1	L2	L3	L1	L3	L1	L2	L3	L3	L1	L2	L2	L2	L1	L3
ANN36	L3	L2	L3	L1	L2	L1	L2	L3	L1	L1	L2	L3	L2	L2	L1	L3

Table 3. Information on used datasets (UCP).

	Dataset	Number of Projects	Experiment
Dataset_1	UCP Benchmark Dataset	50	Training
Dataset_2	UCP Benchmark Dataset	21	Testing
Dataset_3	Combined	18	Validation1
Dataset_4	Combined Industrial projects	17	Validation2

Table 4. Basic statistics about dataset (UCP).

Datasets	N	Min (PM)	Max (PM)	Mean (PM)	Std. Deviation (PM)
Dataset_1	50	5775.0	7970.0	6506.940	653.0308
Dataset_2	21	6162.6	6525.3	6393.993	118.1858
Dataset_3	18	2692.1	3246.6	2988.392	233.2270
Dataset_4	17	2176.0	3216.0	2589.400	352.0859

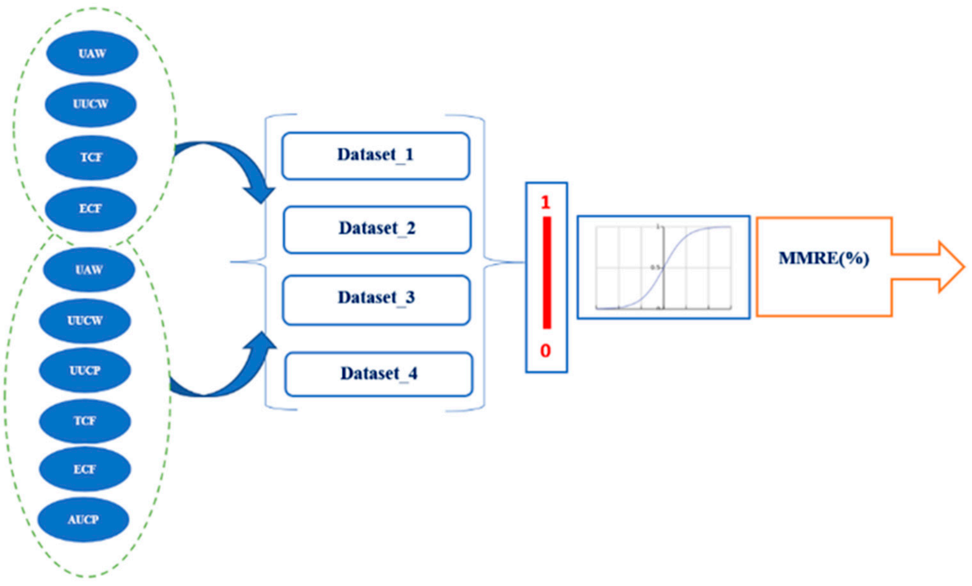

Figure 3. Robust design algorithm for performing the experiment (UCP).

Step 1: Input layer

The input values of the first proposed architecture ANN-L16 are six input values, four of which are independent (UAW, UUCW, TCF, and ECF) and two dependent (UUCP and AUCP). The input values of the second proposed architecture ANN-L36prim are four independent input values: UAW, UUCW, TCF, and ECF.

Step 2:

All input values are transformed according to the following formula: The function $\mu_D(X)$: $R \to [0, 1]$ translates the real values of input signals into coded values from the interval [0, 1] in the following way: $\mu_D(X_i) = (X_i - X_{min})/(X_{max} - X_{min})$ [33,34], where D is the set of data on which the experiment is performed, X_i is the input value, X_{min} is the smallest input value, and X_{max} the greatest input value on the observed dataset.

Step 3:

The sigmoid function, as the activation function of the hidden layer, is used (19):

$$y_i = \frac{1}{1+e^{-x_i}},\ i = \overline{1,n} \tag{19}$$

The construction of the activation function is based on a combination of input values and corresponding weight coefficients W_i for each of the listed ANN architectures.

The hidden and output layer functions, for the ANN-L16 architecture, are as follows (20)–(22):

$$Y_1 = 1/\left(1+e^{-(X_1\cdot W_1+X_2\cdot W_3+X_3\cdot W_5+X_4\cdot W_7+X_5\cdot W_9+X_6\cdot W_{11})}\right) \tag{20}$$

$$Y_2 = 1/\left(1+e^{-(X_1\cdot W_2+X_2\cdot W_4+X_3\cdot W_6+X_4\cdot W_8+X_5\cdot W_{10}+X_6\cdot W_{12})}\right) \tag{21}$$

$$EstEffANN - L16 = 1/\left(1+e^{-(Y_1\cdot W_{13}+Y_2\cdot W_{14}+1\cdot W_{15})}\right) \tag{22}$$

where Y_1, Y_2, and Y_3 are the hidden layer functions and *EstEffortANN-L16* represents the output function.

The hidden and output layer functions for the ANN-L36prim architecture are as follows (23)–(26):

$$Y_1 = \frac{1}{1+e^{-(X_1\cdot W_1+X_2\cdot W_4+X_3\cdot W_7+X_4\cdot W_{10})}} \tag{23}$$

$$Y_2 = \frac{1}{1 + e^{-(X_1 \cdot W_2 + X_2 \cdot W_5 + X_3 \cdot W_8 + X_4 \cdot W_{11})}} \tag{24}$$

$$Y_3 = \frac{1}{1 + e^{-(X_1 \cdot W_3 + X_2 \cdot W_6 + X_3 \cdot W_9 + X_4 \cdot W_{12})}} \tag{25}$$

$$EstEffortANN - L36prim = \frac{1}{1 + e^{-(Y_1 \cdot W_{13} + Y_2 \cdot W_{14} + Y_3 \cdot W_{15} + 1 \cdot W_{16})}} \tag{26}$$

where Y_1, Y_2, and Y_3 are the hidden layer functions and *EstEffortANN-L36prim* represents the output function.

In the first proposed ANN-L16 architecture, an orthogonal vector plan of two levels L1 and L2, and the initial values of the weighting factors W_i that take the values from the interval $[-1, 1]$, are used. The second proposed architecture has an orthogonal vector plan of three levels, L1, L2, and L3, and the initial values of the weighting factors W_i that take the values from the interval $[-1, 0, 1]$. For each subsequent iteration, new weight factor values must be calculated as follows (e.g., for ANN-L16 architecture) [2–4] (27):

$$\begin{gathered} W_1L1 = cost1 + cost2 + \ldots + cost8 \\ W_1L2 = cost9 + cost10 + \ldots + cost16 \\ \ldots \\ W_{15}L1 = cost1 + cost6 + \ldots + cost16 \\ W_{15}L2 = cost2 + cost3 + \ldots + cost15 \\ \text{where } cost(i) = \Sigma\, MRE(ANN(i)) \end{gathered} \tag{27}$$

For each subsequent iteration, the interval $[-1, 1]$ is divided depending on the cost effect function as follows [7,30] (28):

$$\begin{gathered} W_1L1new = W_1L1old \\ W_1L2new = W_1L2old + (W_1L3old - W_1L2old)/2 \\ W_1L3new = W_1L3old \end{gathered} \tag{28}$$

where W_1L1old, W_1L2old, and W_1L3old are values form the previous iteration. The set of input values of each dataset converges depending on the value of the cost effect function.

Step 4:

The defuzzification method is used according to the following Formulas (29) and (30) [35]:

$$Y_i = (X_{min} + \mu_D(X_i)) \cdot (X_{max} - X_{min}) \tag{29}$$

$$OA(ANN\boldsymbol{i}) = Y\boldsymbol{i}, \text{ where } \boldsymbol{i} = 16, \boldsymbol{i} = 36. \tag{30}$$

where OA represents actual effort of the particular project, which is calculated based on ANN-L12 and ANN-L36prim.

Step 5:

For each iteration in our experiment, the output values are obtained according to the following formulas/measures [2,4,30] (31)–(35):

$$Deviation = |ActEffort - EstEffort| \tag{31}$$

$$MAE_i = \frac{1}{n} \sum_{i=1}^{n} |ActEffort - EstEffort| \tag{32}$$

$$MRE = Deviation/ActEffort \tag{33}$$

$$MRE = \frac{1}{n} \cdot \sum_{i=1}^{n} MRE_i \tag{34}$$

$$MMRE = mean\ (MRE) \tag{35}$$

For each of the experimental parts in every iteration, the Gradient Descent is monitored with the condition GA < 0.01, calculated as [4,30,31] (36):

$$GA = MRE_{i1} - MRE_{i2} < 0.01,\ where\ i = 1, \ldots, n\ n\ is\ a\ number\ of\ ANN. \quad (36)$$

Step 6:

This step concerns the influence of the dependent variables UUCP and AUCP on the change in MMRE value.

1. The influence of the input parameter UUCP and its value are calculated as (37):

$$\delta 1 = mean(\mathrm{MMRE}) - mean(\mathrm{MMRE}_1) \quad (37)$$

where MMRE_1 is *mean*(MMRE) when UUCP = 0;
2. The influence of the input parameter AUCP and its value are calculated as (38):

$$\delta 2 = mean(\mathrm{MMRE}) - mean(\mathrm{MMRE}_2)\ \text{when AUCP} = 0; \quad (38)$$

Step 7: Correlation, Prediction

The Pearson's [36], Spearman's [37] and R^2 [38] coefficients are monitored (39).

$$Correl(X,\ Y) = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2 \sum_{i=1}^{N}(y_i - \overline{y})^2}} \quad (39)$$

The second and third parts are executed by the same algorithm as the first part, with different projects and datasets being used. The second part uses the ISBSG dataset, but with projects that were not used in the first part. In the third part, the Desharnais dataset and combined dataset are used.

Additionally, prediction at 25%, 30%, and 50% is the percentage of the total number of ANNs that meet the GA criterion (40) [39–41].

$$\begin{gathered} PRED(x)\frac{1}{n}\cdot\sum_{i=1}^{n}\begin{cases} 1,\ if\ MRE \le x \\ 0,\ otherwise \end{cases} \\ \mathrm{PRED(k)} = \mathrm{count(MRE)} < 25\% \\ \mathrm{PRED(k)} = \mathrm{count(MRE)} < 30\% \\ \mathrm{PRED(k)} = \mathrm{count(MRE)} < 50\%,\ \text{where k} = 25,\ \text{k} = 30,\ \text{and k} = 50. \end{gathered} \quad (40)$$

The second and third parts are executed by the same algorithm as the first part, with different projects and datasets being used. The second part uses the also UCP Benchmark (Mendeley) dataset, but with projects that were not used in the first part. In the third part, the combined industrial datasets were used.

4. Discussion

With the UCP model, it is possible to measure the size of the system as with the model of functional points. A model that uses system user characteristics and use cases is a newer method of software evaluation. It is one of the most commonly used models due to the exceptional evaluation results that its application can achieve. The disadvantage of this model is that it does not consider the data structure in the system because such data are not contained in the cases of use. Table 5 shows the results obtained by training the first proposed ANN-L16 architecture on the used dataset. The number of iterations concerning the set GA criterion was monitored. The GA criterion was met after four iterations. Based on all MRE values in each executed iteration, the "Winner" network, i.e., the ANN network with the lowest MRE value, was determined. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN6) is 6.7%, and the value for MMRE is 7.1%. In addition to examining the MMRE value, the convergence rate on all training data for the two ANN architectures was examined. It can be concluded that the ANN-L36prim architecture quickly converges to the minimum knowledge of MMRE compared to the ANN-L16 architecture (Figure 4).

Table 5. ANN-L16 training results.

No. of Iter.	**1.**		**2.**		**3.**		**4.**	
ANN-L16	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.084	0.084	0.080	0.004	0.076	0.004	0.072	0.004
ANN2	0.196	0.196	0.112	0.084	0.082	0.030	0.073	0.009
ANN3	0.188	0.188	0.113	0.076	0.081	0.031	0.072	0.009
ANN4	0.085	0.085	0.077	0.008	0.074	0.003	0.072	0.003
ANN5	0.161	0.161	0.105	0.056	0.080	0.025	0.073	0.008
ANN6	0.069	0.069	0.068	0.002	0.067	0.000	0.067	0.000
ANN7	0.078	0.078	0.073	0.006	0.071	0.002	0.070	0.001
ANN8	0.151	0.151	0.105	0.046	0.081	0.024	0.073	0.008
ANN9	0.191	0.191	0.120	0.071	0.076	0.044	0.072	0.004
ANN10	0.073	0.073	0.080	0.007	0.074	0.006	0.073	0.001
ANN11	0.078	0.078	0.080	0.001	0.075	0.005	0.072	0.003
ANN12	0.130	0.130	0.084	0.047	0.074	0.010	0.070	0.003
ANN13	0.113	0.113	0.083	0.030	0.074	0.009	0.071	0.002
ANN14	0.094	0.094	0.080	0.014	0.072	0.008	0.071	0.002
ANN15	0.094	0.094	0.080	0.015	0.073	0.007	0.071	0.002
ANN16	0.102	0.102	0.082	0.021	0.074	0.008	0.071	0.002
GA		16		10		5		0
Winner	6.9%		6.8%		6.7%		6.7%	
MMRE	11.8%		8.9%		7.5%		7.1%	

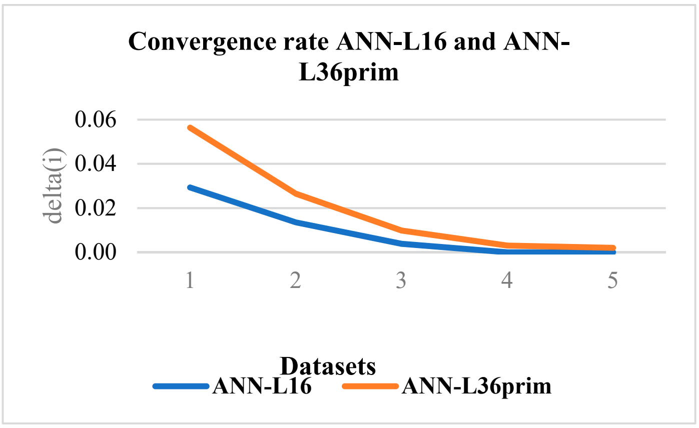

Figure 4. Convergence rate of ANN-L16 vs. ANN-L36prim (UCP).

A graphical representation of the GA values, during four iterations, is shown in Figure 5.

A graphical representation of the MRE value for the Winner network, relative to the MMRE value on the training dataset, during the four iterations, is given in Figure 6.

Table 6 shows the results obtained by training the second proposed ANN-36prim architecture on the used dataset. The number of iterations concerning the set GA criterion was monitored. The GA criterion was met after six iterations. Based on all MRE values in each executed iteration, the "Winner" network was determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN10) is 6.9%, and the value for MMRE is 7.0%.

A graphical representation of GA values, during six iterations, is shown in Figure 7.

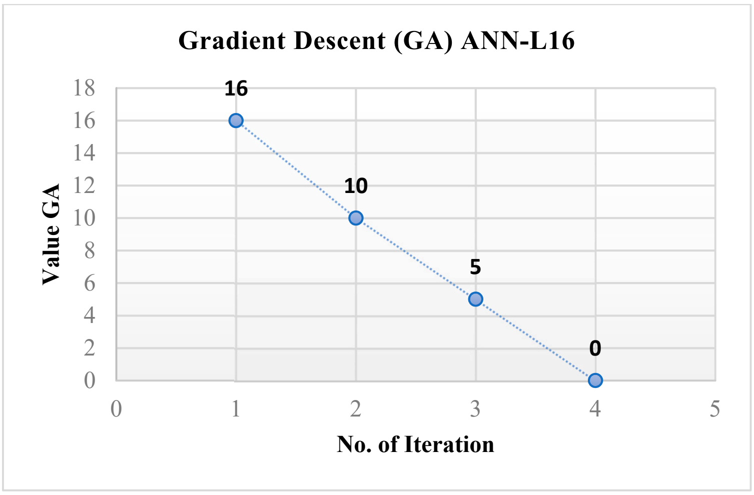

Figure 5. GA for ANN-L16-training part (UCP).

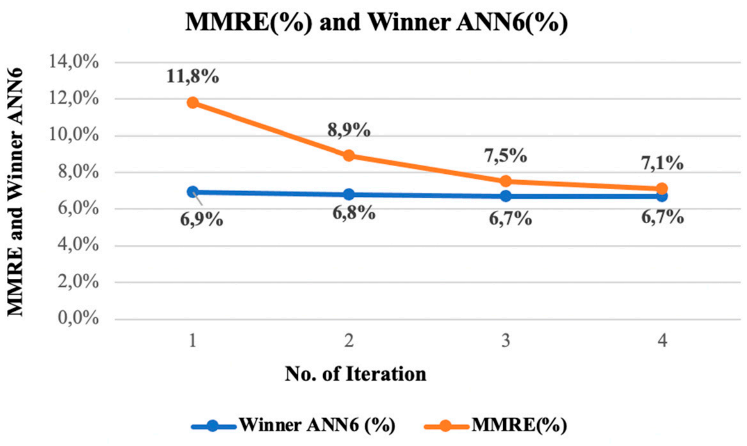

Figure 6. “Winner” MRE vs. MMRE on the training dataset (ANN-L16).

Table 6. ANN-L36prim training results.

	ANN-L36prim					
GA	36	35	23	14	3	0
Winner	7.3%	7.2%	7.1%	7.0%	7.0%	6.9%
MMRE	12.1%	9.4%	8.4%	7.5%	7.2%	7.0%

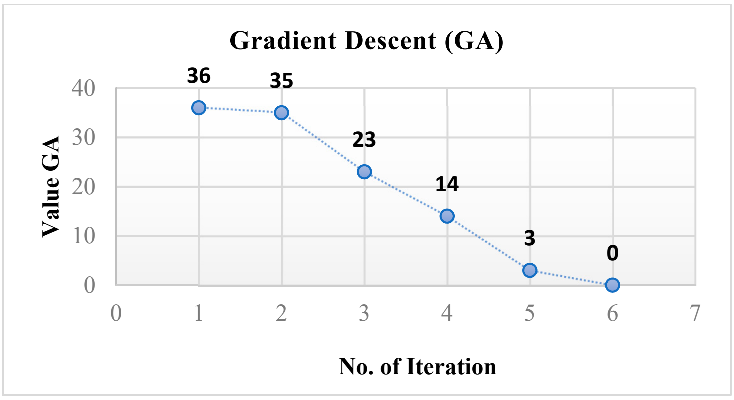

Figure 7. GA for ANN-L36prim-training part (UCP).

A graphical representation of the MRE value for the Winner network, relative to the MMRE value on the training dataset, during six iterations, is given in Figure 8.

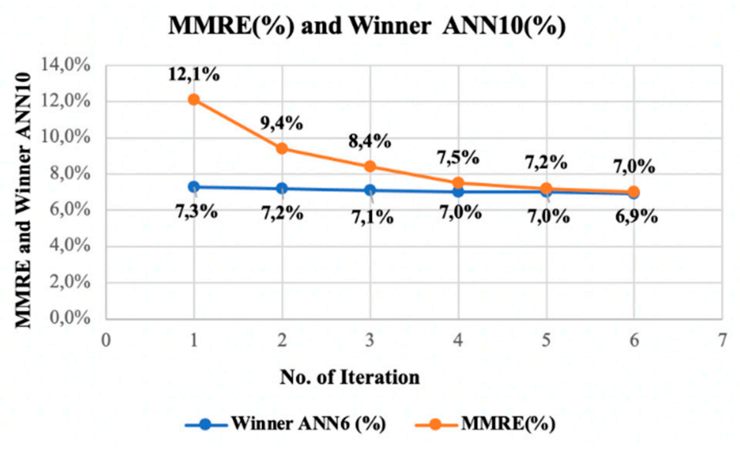

Figure 8. "Winner" MRE vs. MMRE on the training dataset (ANN-L36prim).

The obtained results for the two proposed architectures, ANN-L16 and ANN-L36, in all three parts of the experiment showed that the different nature of the data set does not affect the complexity of the architecture used. That is, it does not depend on the value of the input values. In the first proposed architecture, ANN-L16, all six input values were used (where four are linearly independent and two linearly dependent), and the MMRE value in all three parts of the experiment is 7.5% of Table 7. Using the second architecture, ANN-L36prim, with four independent input values, the same MMRE value, of 7.5%, was obtained in all three parts of the experiment (Table 7). The error differences in individual parts of the experiment are not more than 0.5%, indicating the proposed model's reliability.

Table 7. MMRE value in all three parts of the experiment (UCP).

Datasets	ANN-L16	ANN-L36prim	Part of Experiment
	MMRE (%)	MMRE (%)	
Dataset_1	6.7	7.0	Training
	7.1	7.1	Testing
Dataset_2	8.0	7.5	Validation1
Dataset_3	8.3	8.4	Validation2
AVERAGE(MMRE)	7.5	7.5	

The huge values of the correlation coefficients (Pearson's and Spearman's rho) further show the consistency of the actual and estimated values obtained using the proposed models. In the ANN-L36prim architecture, the Pearson value is 0.983, which indicates an exceptional interrelationship between the observed values (Table 8).

Table 8. Correlation coefficients (UCP).

Correlation	ANN-L16	ANN-L36prim
Pearson's	0.875	0.983
Spearman's rho	0.784	0.962

Prediction represents the number of projects that have an error less than the set criterion. Prediction can further confirm the validity and reliability of the models used. For all three proposed criteria (PRED (25), PRED (30), and PRED (50)) and in all three parts of the experiment, using both proposed architectures, the value is 100% (Table 9).

Table 9. Prediction values (UCP).

	Training	
PRED (%)	ANN-L16 (%)	ANN-L36prim (%)
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0
	Testing	
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0
	Validation1	
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0
	Validation2	
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0

By examining the influence of dependent and independent variables on the change in the MMRE value, it was shown that it is sufficient to use a four-dimensional vector instead of a six-dimensional vector. The error with dependent input values on the four datasets used is between −0.3 and 0.5, which is less than 1%. The most significant influence is the input value of AUCP (Dataset_3), and the change in the value of MMRE is, in this case, higher by 0.5%. The slightest influence has the input value of UUCW (Dataset_4), and the change in the value of MMRE is, in this case, lower by 0.5%, which would mean that the error can be reduced/increased if the observed values are further analyzed. It can be concluded that the architecture with six input sizes can be replaced with the architecture with four input sizes. That is, in the observed approach, the ANN-L16 architecture can be replaced with the ANN-L36prim architecture (Table 10).

Table 10. Influence of the input values on the change in MMRE (UCP).

Dataset	**MMRE**	**UAW**	**UUCW**	**UUCP**	**TCF**	**ECF**	**AUCP**
Dataset_1	6.7%	7.1%	6.7%	7.0%	6.7%	6.7%	7.1%
Dataset_2	7.0%	7.1%	7.0%	7.2%	6.9%	7.1%	7.2%
Dataset_3	8.0%	7.9%	8.1%	7.9%	8.1%	8.1%	7.5%
Dataset_4	8.3%	7.9%	8.4%	7.9%	8.3%	8.2%	8.0%

From Table 11, it can be concluded that the dependent variable UUCP has less impact than the dependent variable AUCP. The most significant influence of AUCP (Dataset_3) on the change in the MMRE value is 0.5%. The slightest influence of AUCP (Dataset_1) on the change in the MMRE value is −0.3%.

Table 11. Influence of dependent variables (UUCP and AUCP) on the change in the MMRE value.

Dataset	**UUCP**	**g − UUCP = MMRE − UUCP**	**AUCP**	**g − AUCP = MMRE − AUCP**
Dataset_1	6.9%	−0.1%	6.8%	−0.3%
Dataset_2	7.1%	−0.1%	7.1%	−0.1%
Dataset_3	8.0%	0.1%	8.0%	0.5%
Dataset_4	8.2%	0.2%	8.2%	0.2%
	max	0.2%	max	0.5%
	min	−0.1%	min	−0.3%

A graphical representation of the dependent input values of UUCP and AUCP with the values of their errors is given in Figure 9.

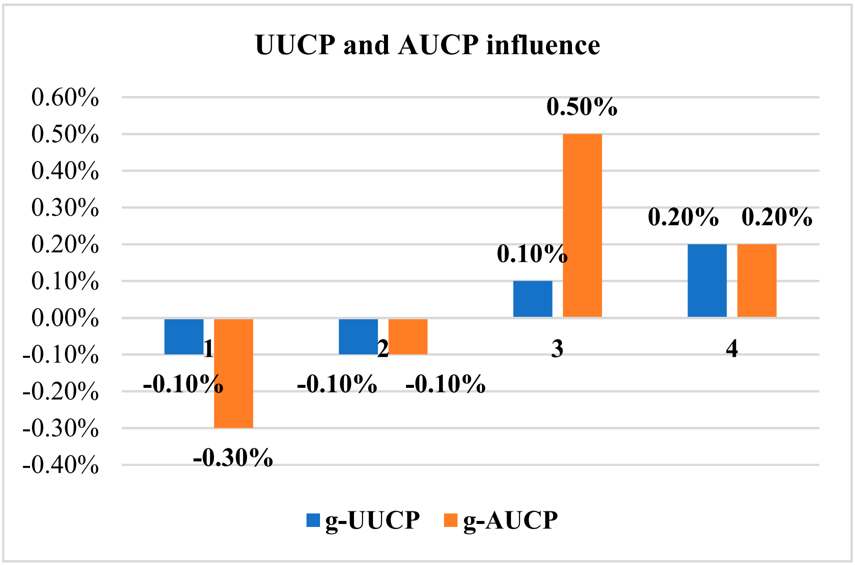

Figure 9. Influence of dependent variables (UUCP and AUCP) on the change in the MMRE value.

By comparing the results of the parametric method COCOMO2000 with improved COCOMO2000 and ANN, it can be concluded that the model error is reduced by 193.1/43.3 = 4.5 times. In the second proposed approach, comparing the parametric method COCOMO2000 and the improved COSMIC FP and ANN, the model error reduction is 193.1/28.8 = 6.7 times. Compared with the COCOMO2000 parametric method with UCP and ANN, the model error reduction is 193.1/7.5 = 25.7 times (Table 12; Figure 10). In the first proposed approach, the lowest model error value is 43.3% for the ANN-L36 architecture. In the second proposed approach, the lowest error value is achieved with ANN-L36prim, with a value of 28.8%. In the third proposed approach, both proposed architectures, ANN-L16 and ANN-L36prim, give the lowest model error value of 7.5% (Table 12; Figure 10). It can be concluded that the third proposed UCP approach achieves the lowest MMRE value. In addition, the ANN-L16 architecture in this approach converges rapidly and reaches the "stop criterion" after the fourth iteration, which is also the lowest number of repeated iterations that apply to all architectures used in all three proposed approaches. The influence of dependent variables on the change of MMRE values in the ANN-L16 architecture is less than 0.5%. It can be concluded that the improved UCP model using the ANN-L16 architecture is the best-proposed estimate of effort and cost for software project development.

Table 12. MMRE values for the approaches used.

MMRE (%)		**COCOMO2000 and ANN**			**COSMIC FP and ANN**		**UCP and ANN**	
COCOMO2000	ANN-L9	ANN-L18	ANN-L27	ANN-L36	ANN-L12	ANN-L36prim	ANN-L16	ANN-L36prim
193.1%	72.0%	59.7%	45.3%	43.3%	29.7%	28.8%	7.5%	7.5%

By selecting the best ANN architectures, which achieved the lowest MMRE value for each of the three proposed improved models, it can be concluded that: COSMIC FP and ANN are 43.3/28.8 = 1.5 times better than COCOMO2000 and ANN; UCP and ANN are 48.8/7.5 = 5.8 times better than COCOMO2000 and ANN; and UCP and ANN are 28.8/7.5 = 3.8 times better than COSMIC FP and ANN (Table 13).

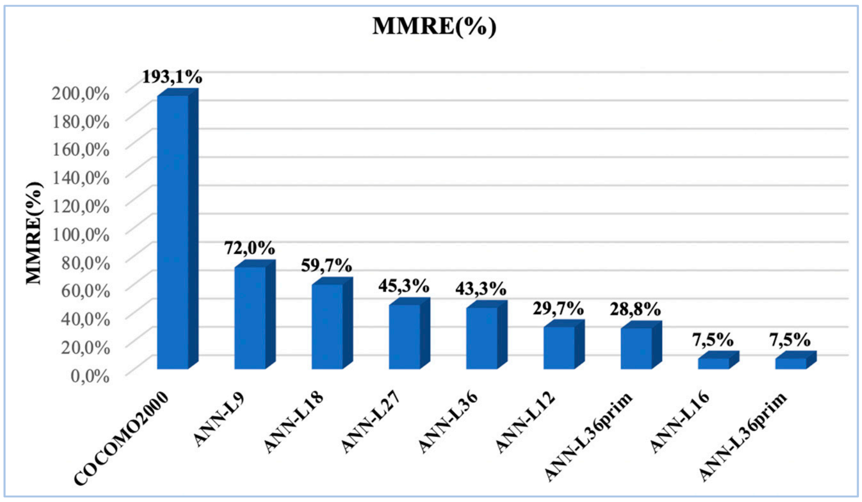

Figure 10. MMRE value for the approaches used.

Table 13. COCOMO2000 and ANN vs. COSMIC FP and ANN vs. UCP and ANN.

	COCOMO2000 and ANN	**COSMIC FP and ANN**	**UCP and ANN**	
MMRE (%)	ANN-L36 43.3%	ANN-L36prim 28.8%	ANN-L16 7.5%	ANN-L36prim 7.5%

The results shown for this approach (UCP) in the previous tables and figures were processed in the R programming language and checked in the Python programming language within the RStudio environment. The data required for statistical analysis were processed using the IBM SPSS Statistical 25 software tool.

5. Conclusions

The proposed UCP model uses two different ANN architectures and four different datasets, a sigmoid activation function, a fuzzification method, and a Taguchi method to estimate the effort and cost of software development. By monitoring the MMRE value and the convergence rate of each of these architectures, this model gives much better results compared to the previous two models. Based on the three performed parts of the experiment, it was concluded that the ANN-L16 architecture converges after the fourth iteration and gives an MMRE value of only 7.5%, which is 35.8% better than the first COCOMO2000 model. The error value of the UCP model is 21.3% lower than the other proposed COSMIC FP model. Following the prediction through all parts of the experiment, both ANN architectures of this model have a value of 100%, which means that the model is exact, accurate, and reliable. In addition to the MMRE value, the influence of the dependent variables UUCP and AUCP was monitored to check the influence on the change in the MMRE value. The resulting error is less than 0.5%, so it can be concluded that the ANN-L36prim architecture and vice versa can replace the ANN-L16 architecture. Compared to the results obtained in previous studies, the best so far being 10% [22], our proposed approach gave a better result. The main advantages of this model are as follows: the number of iterations being in the interval from 4 to 6, which means reduced effort estimation time thanks to the exceptional convergence rate of both architectures; the simplicity of the two proposed ANN architectures; the high coverage of different real effort values; and the lowest MMRE value being 7.5%. A possible drawback is the finding of new methods that could further reduce the MMRE value. There are no specific limitations in applying this approach. This model can be used alone or in combination with the previous two for assessment depending on the company's historical data for which the software is implemented. Although not as standardized as the previous two, it is increasingly used by software companies, software engineers, and teams to assess the effort required to implement software projects effectively.

Possible applications of the proposed model are as follows: signal processing; image and speech recognition; the recognition and processing of natural languages and different types of knowledge; the recognition of printed texts; and others. This model can also be successfully used in the medical sciences to construct various software solutions to diagnose many diseases. In addition, it is widely used in meteorology to forecast weather conditions. It can be relevant in nuclear science, robotics, automatic control, telecommunications, finance, and banking services.

Numerous new applications of the proposed model of artificial intelligence are expected in the future. Future research will focus on constructing models to solve problems related to cybercrime.

Author Contributions: Conceptualization, N.R., D.R., M.I. and L.L.; Data curation, N.R., D.R., M.I. and L.L.; Formal analysis, N.R., D.R., M.I. and L.L.; Funding acquisition, N.R., D.R., M.I. and L.L.; Investigation, N.R., D.R., M.I. and L.L.; Methodology, N.R., D.R., M.I. and L.L.; Project administration, N.R., D.R., M.I. and L.L.; Resources, N.R., D.R., M.I. and L.L.; Software, N.R., D.R., M.I. and L.L.; Supervision, N.R., D.R., M.I. and L.L.; Validation, N.R., D.R., M.I. and L.L.; Visualization, N.R., D.R., M.I. and L.L.; Writing—original draft, N.R., D.R., M.I. and L.L.; Writing—review & editing, N.R., D.R., M.I. and L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study is based on the research and data that will be presented in detail in the doctoral dissertation of Nevena Rankovic.

Acknowledgments: Mirjana Ivanovic acknowledge financial support of the Ministry of Education, Science and Technological Development of the Republic of Serbia (Grant No. 451-03-9/2021-14/ 200125).

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

References

1. Fadhil, A.A.; Alsarraj, R.G.H.; Altaie, A.M. Software Cost Estimation Based on Dolphin Algorithm. *IEEE Access* **2020**, *8*, 75279–75287. [CrossRef]
2. Stoica, A.; Blosiu, J. Neural Learning using orthogonal arrays. *Adv. Intell. Syst.* **1997**, *41*, 418.
3. Khaw, J.F.; Lim, B.; Lim, L.E. Optimal design of neural networks using the Taguchi method. *Neurocomputing* **1995**, *7*, 225–245. [CrossRef]
4. Rankovic, N.; Rankovic, D.; Ivanovic, M.; Lazic, L. A New Approach to Software Effort Estimation Using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays. *IEEE Access* **2021**, *9*, 26926–26936. [CrossRef]
5. Langsari, K.; Sarno, R. Optimizing effort and time parameters of COCOMO II estimation using fuzzy multi-objective PSO. In Proceedings of the 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Yogyakarta, Indonesia, 19–21 September 2017; pp. 1–6. [CrossRef]
6. Quesada-López, C.; Madrigal-Sánchez, D.; Jenkins, M. An Empirical Analysis of IFPUG FPA and COSMIC FP Measurement Methods. In *International Conference on Information Technology & Systems, Bogota, Colombia, 5–7 February 2020*; Springer: Cham, Switzerland, 2020; pp. 265–274.
7. Symons, C. Function point analysis: Difficulties and improvements. *IEEE Trans. Softw. Eng.* **1988**, *14*, 2–11. [CrossRef]
8. Lavazza, L.; Liu, G. An Empirical Evaluation of the Accuracy of NESMA Function Points Estimates. *ICSEA* **2019**, *2019*, 36.
9. Ochodek, M.; Kopczyńska, S.; Staron, M. Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names. *Inf. Softw. Technol.* **2020**, *123*, 106310. [CrossRef]
10. Kläs, M.; Trendowicz, A.; Wickenkamp, A.; Münch, J.; Kikuchi, N.; Ishigai, Y. Chapter 4 the Use of Simulation Techniques for Hybrid Software Cost Estimation and Risk Analysis. *Adv. Organomet. Chem.* **2008**, *74*, 115–174. [CrossRef]
11. Kirmani, M.M.; Wahid, A. Revised use case point (re-ucp) model for software effort estimation. *Int. J. Adv. Comput. Sci. Appl.* **2015**, *6*, 65–71.
12. Azzeh, M.; Nassif, A.B.; Banitaan, S. Comparative analysis of soft computing techniques for predicting software effort based use case points. *IET Softw.* **2018**, *12*, 19–29. [CrossRef]

13. Grover, M.; Bhatia, P.K.; Mittal, H. Estimating Software Test Effort Based on Revised UCP Model Using Fuzzy Technique. In *International Conference on Information and Communication Technology for Intelligent Systems, Ahmedabad, India, 25–26 March 2017*; Springer: Cham, Switzerland, 2017; pp. 490–498.
14. Ani, Z.C.; Basri, S.; Sarlan, A. A reusability assessment of UCP-based effort estimation framework using object-oriented approach. *J. Telecommun. Electron. Comput. Eng. JTEC* **2017**, *9*, 111–114.
15. Sharma, P.; Singh, J. Systematic Literature Review on Software Effort Estimation Using Machine Learning Approaches. In Proceedings of the 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS), Jammu, India, 11–12 December 2017; pp. 43–47.
16. Gustav, K. *Resource Estimation for Objectory Projects*; Objective Systems SF AB: Kista, Sweden, 1993; pp. 1–9.
17. Nassif, A.B.; Capretz, L.F.; Ho, D. Enhancing Use Case Points Estimation Method using Soft Computing Techniques. *J. Glob. Res. Comput. Sci.* **2010**, *1*, 12–21.
18. Couellan, N. Probabilistic robustness estimates for feed-forward neural networks. *Neural Netw.* **2021**, *142*, 138–147. [CrossRef] [PubMed]
19. Mukherjee, S.; Malu, R.K. Optimization of project effort estimate using neural network. In Proceedings of the 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, India, 8–10 May 2014; pp. 406–410.
20. Dar, A.A.; Anuradha, N. Use of orthogonal arrays and design of experiment via Taguchi L9 method in probability of default. *Accounting* **2018**, *4*, 113–122. [CrossRef]
21. Alshibli, M.; El Sayed, A.; Kongar, E.; Sobh, T.; Gupta, S.M. A Robust Robotic Disassembly Sequence Design Using Orthogonal Arrays and Task Allocation. *Robotics* **2019**, *8*, 20. [CrossRef]
22. Carroll, E.R. Estimating software based on use case points. In Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications—OOPSLA '05, San Diego, CA, USA, 16–20 October 2005; pp. 257–265.
23. Nassif, A.B. Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models. Ph.D. Thesis, Western University, London, ON, Canada, 2012.
24. Azzeh, M. Fuzzy Model Tree for Early Effort Estimation Machine Learning and Applications. In Proceedings of the 12th International Conference on Machine Learning and Applications, Miami, FL, USA, 4–7 December 2013; pp. 117–121.
25. Urbanek, T.; Prokopova, Z.; Silhavy, R.; Sehnalek, S. Using Analytical Programming and UCP Method for Effort Estimation. In *Modern Trends and Techniques in Computer Science*; Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2014; Volume 285, pp. 571–581.
26. Kaur, A.; Kaur, K. Effort Estimation for Mobile Applications Using Use Case Point (UCP). In *Smart Innovations in Communication and Computational Sciences*; Springer: Singapore, 2019; pp. 163–172.
27. Mahmood, Y.; Kama, N.; Azmi, A. A systematic review of studies on use case points and expert-based estimation of software development effort. *J. Softw. Evol. Process.* **2020**, *32*, e2245. [CrossRef]
28. Gebretsadik, K.K.; Sewunetie, W.T. Designing Machine Learning Method for Software Project Effort Prediction. *Comput. Sci. Eng.* **2019**, *9*, 6–11.
29. Alves, R.; Valente, P.; Nunes, N.J. Improving software effort estimation with human-centric models: A comparison of UCP and iUCP accuracy. In Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, London, UK, 24–27 June 2013; pp. 287–296.
30. Rankovic, D.; Rankovic, N.; Ivanovic, M.; Lazic, L. Convergence rate of Artificial Neural Networks for estimation in software development projects. *Inf. Softw. Technol.* **2021**, *138*, 106627. [CrossRef]
31. Rankovic, N.; Rankovic, D.; Ivanovic, M.; Lazic, L. Improved Effort and Cost Estimation Model Using Artificial Neural Networks and Taguchi Method with Different Activation Functions. *Entropy* **2021**, *23*, 854. [CrossRef] [PubMed]
32. Available online: https://data.mendeley.com/datasets/2rfkjhx3cn/1 (accessed on 4 February 2020).
33. Lam, H. A review on stability analysis of continuous-time fuzzy-model-based control systems: From membership-function-independent to membership-function-dependent analysis. *Eng. Appl. Artif. Intell.* **2018**, *67*, 390–408. [CrossRef]
34. Ghosh, L.; Saha, S.; Konar, A. Decoding emotional changes of android-gamers using a fused Type-2 fuzzy deep neural network. *Comput. Hum. Behav.* **2021**, *116*, 106640. [CrossRef]
35. Ritu, O.P. Software Quality Prediction Method Using Fuzzy Logic. *Turk. J. Comput. Math. Educ.* **2021**, *12*, 807–817.
36. Boldin, M.V. On the Power of Pearson's Test under Local Alternatives in Autoregression with Outliers. *Math. Methods Stat.* **2019**, *28*, 57–65. [CrossRef]
37. Blum, D.; Holling, H. Spearman's law of diminishing returns. A meta-analysis. *Intelligence* **2017**, *65*, 60–66. [CrossRef]
38. Wang, H.; Wang, Z. Deterministic and probabilistic life-cycle cost analysis of pavement overlays with different pre-overlay conditions. *Road Mater. Pavement Des.* **2019**, *20*, 58–73. [CrossRef]
39. Qiao, L. Deep learning based software defect prediction. *Neurocomputing* **2020**, *385*, 100–110. [CrossRef]
40. Shah, M.A. Ensembling Artificial Bee Colony with Analogy-Based Estimation to Improve Software Development Effort Prediction. *IEEE Access* **2020**, *8*, 58402–58415. [CrossRef]
41. Manali, P.; Maity, R.; Ratnam, J.V.; Nonaka, M.; Behera, S.K. Long-lead prediction of ENSO modoki index using machine learning algorithms. *Sci. Rep.* **2020**, *10*, 365.