mathematics

Article

Using an Artificial Neural Network for Improving the Prediction of Project Duration

Itai Lishner * and Avraham Shtub

Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology, Haifa 320003, Israel
* Correspondence: itailishner@campus.technion.ac.il

Abstract: One of the most challenging tasks in project management is estimating the duration of a project. The unknowns that accompany projects, the different risks, the uniqueness of each project, and the differences between organizations' culture and management techniques, hinder the ability to build one project duration prediction tool that can fit all types of projects and organizations. When machine learning (ML) techniques are used for project duration prediction, the challenge is even greater, as each organization has a different dataset structure, different features, and different quality of data. This hinders the ability to create one ML model that fits all types of organizations. This paper presents a new dynamic ML tool for improving the prediction accuracy of project duration. The tool is based on an artificial neural network (ANN) which is automatically adapted and optimized to different types of prediction methods and different datasets. The tool trains the ANN model multiple times with different architectures and uses a genetic algorithm to eventually choose the architecture which gives the most accurate prediction results. The validation process of the prediction accuracy is performed by using real-life project datasets supplied by two different organizations which have different project management approaches, different project types, and different project features. The results show that the proposed tool significantly improved the prediction accuracy for both organizations despite the major differences in the size, type, and structure of their datasets.

Keywords: machine learning; artificial intelligence; prediction; GPU; artificial neural network; project management

MSC: 68T07

Citation: Lishner, I.; Shtub, A. Using an Artificial Neural Network for Improving the Prediction of Project Duration. *Mathematics* **2022**, *10*, 4189. https://doi.org/10.3390/math10224189

Academic Editor: Freddy Gabbay

Received: 17 October 2022
Accepted: 4 November 2022
Published: 9 November 2022

1. Introduction

A project is a "temporary endeavor undertaken to create a unique product, service, or result." [1]. Due to the non-repetitive nature of the projects, the ability to predict their outcome is limited. Several studies have compared the predicted schedule with the actual duration of projects [2–4]. The results show that 70% of the projects are late. While the majority of the projects eventually experience schedule overruns, the surveys show that the top reason for project failure is missing deadlines [2,4,5]; hence, the importance of good prediction for project duration is essential for project success.

Several studies have tried to solve the scheduling overrun problem and improve the prediction of project duration. Nevertheless, most of the solutions are theoretical, as few practical applications in large-scale real-life projects have been reported [6–8]. One of the most common practical tools for project duration estimation and planning is the Gantt chart [9]; other methods developed to allow project duration prediction include the critical path method (CPM) and PERT (program evaluation and review technique) [10]. While applying the CPM is a very popular way for project duration estimation [11], the uncertainty that characterizes projects affects the quality of project planning and, as a consequence, the estimation of the project duration is poor. The Gantt chart lacks the ability to foresee changes in resources or activities; it does not take into account the uncertainty

of the project execution and it is exposed to estimation errors [12–16]. All of the above prevent Gantt from being a good schedule prediction method that can produce acceptable prediction results. Some studies [17,18] suggest relying on information from past projects in order to improve the project's duration prediction. The prediction can be made by creating a regression model that foresees the gap between the original duration estimation and the actual duration of the project. With the right regression model and quality data extracted from past projects, the model can potentially produce better predictions than the traditional methods. Collecting more data and using more project features as input for the prediction model will probably increase the prediction accuracy but will also make the regression calculation more complex to solve. The use of artificial intelligence (AI) techniques allows one to create and solve a regression model with multiple features without the need to handle complex calculations.

The use of an AI tool requires a dataset in order to train the AI model; the quality of the data and the amount of the data are important factors when training an AI model. The type and the amount of the data will affect the architecture of the AI model. Organizations have many differences when it comes to managing projects. Each organization uses a different methodology, tools, and techniques; the type of project is different, the experience of the organization is different, and the people involved are different. The location, the technology, and the industry make it almost impossible to build one prediction tool that fits all. When using an AI model, there are even more challenges, as the AI model is directly related to the dataset being used. Different organizations have different ways of collecting data; the amounts of historical data are different and the features that are recorded are different as well. There are very few up-to-date project datasets that are public and validated; this makes the ability to train, test, and validate a new tool more difficult. In order to build a prediction tool that can fit different organizations and produce good enough results, it is required to create a dynamic method that can automatically adjust to the type of organization using it. In this paper, we present a dynamic project duration prediction method that can be used by many types of organizations despite their many differences in culture, industry, and project records. As most AI project prediction methods today remain in the theoretical stage and still have not been validated with real projects, this paper also brings practical validation and implementation. The novelty of this paper and its contribution to the body of knowledge is that it presents a new proven practical method that improves project duration prediction, is validated with real projects, and includes auto-adjustment to different features and formats that different organizations have. The results show a significant improvement in prediction accuracy compared to traditional prediction methods. The paper presents one of the biggest project datasets from a single organization and uses it to validate the presented solution. In addition, the paper also includes two hardware implantation options and the limitation of each method.

The paper is organized as follows:

- Literature review on ANN architecture optimization, ANN training optimization, and the use of ML for project prediction outcomes;
- Detailed description of the dynamic project duration prediction tool;
- Dataset description;
- Results;
- Discussion;
- Conclusions.

2. Literature Review

2.1. Optimization of the Architecture of an Artificial Neural Network

An artificial neural network (ANN) [19–21] is a class of machine learning algorithms that is used to model nonlinear relations in datasets. The ANN architecture is constructed from multiple layers which include hidden layers and activation functions. The basic building block of an ANN is the artificial neuron. An artificial neuron is a mathematical function in which inputs are separately weighted and the sum is passed through a transfer

function to an output connection. The input of an artificial neuron is a vector of numeric values; the artificial neuron receives the vector's values with an independent parameter called weight. The neuron calculates its internal state by summing the weighted product of the input vector and a numerical parameter called bias; this sum is passed through a nonlinear function, which scales all the possible values of the internal state into the desired interval of output values. Artificial neurons are organized into layers to comprise a network. An ANN has an input layer that connects to the input variables; the number of artificial neurons in the input layer is equal to the number of features in the data. The ANN can also contain one or more interior (hidden) layers, which can have many or few artificial neurons, and an output layer that produces the output variables; the number of these variables defines the number of artificial neurons in this layer [22]. Figure 1 shows the general structure of an ANN.

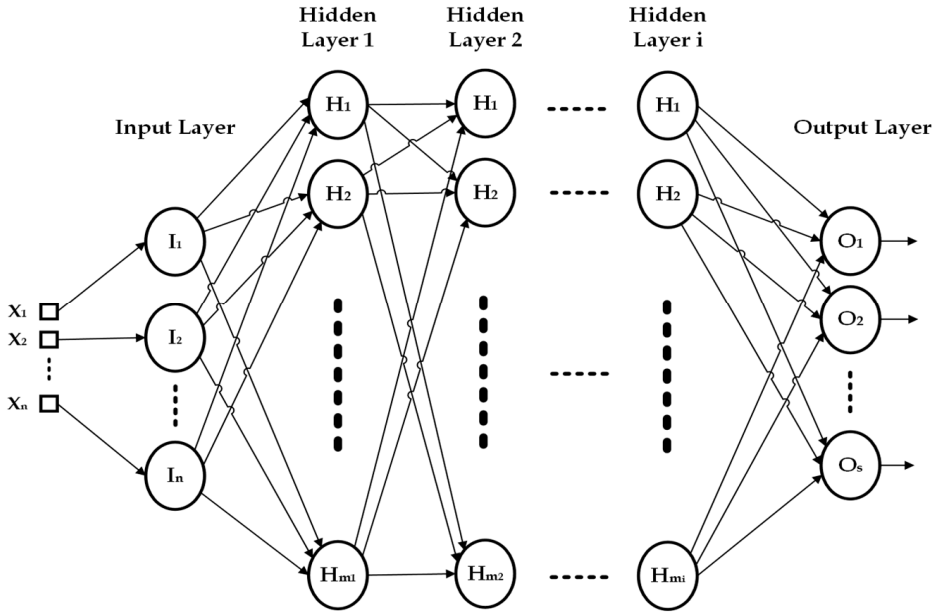

Figure 1. ANN structure.

ANNs have several hyper-parameters that control the architecture of the network; two of them are the number of layers and the number of artificial neurons in each hidden layer. Determining the number of neurons in the hidden layers is a very important part of the overall ANN architecture. Both the number of hidden layers and the number of artificial neurons in each of the layers must be carefully considered, as they have a great influence on the final output. Using too few artificial neurons in the hidden layers will result in underfitting [23]. Underfitting occurs when there are too few artificial neurons in the hidden layers to adequately detect the signals in a complicated dataset. Using too many artificial neurons in the hidden layers can result in overfitting [23]. Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. Hence, the number of artificial neurons in the hidden layers is directly related to the amount of data and features in the dataset. Another side effect of too many neurons in the hidden layers is the complexity of the ANN which increases the time it takes to train the network. There is no single way to build and optimize an ANN architecture to a specific or general use case. According to Lippmann [24], an ANN with two hidden layers is enough for creating classification regions of any desired shape, although there is no indication in his paper of what the implementation of such a network should look like, how many artificial neurons to use in each layer, or how to train the weights. Some theoretical findings have shown that an ANN with one hidden layer can approximate any function required [22,25,26]. On the other hand, it seems that there is no upper limit on the

number of hidden layers [23,27], other than computing, memory, and time requirements. Nevertheless, there seems to be no advantage of using more hidden layers than training cases, since suboptimal local minima do not appear with so many hidden layers [28]. Some studies have offered "rules of thumb" for choosing an architecture. "One rule of thumb is that it [the hidden layer] should never be more than twice as large as the input layer." [29]; "you will never require more than twice the number of hidden units as you have inputs" in an ANN with one hidden layer [30]. "Typically, we specify as many hidden nodes as dimensions [principal components] needed to capture 70–90% of the variance of the input dataset." [31]. While these rules of thumb may be true for some cases, they are ignoring the complexity of the function, the number of training cases, the amount of noise in the targets, and edge cases which are not suitable for these types of rules. For this reason, it is wise to see these rules as a starting point for one to consider when building an ANN.

While it seems that there are many opinions on how to build an optimized ANN there is one consensus that no study has refuted yet—ultimately, designing an appropriate ANN architecture for a specific dataset will come down to trial and error. This "primitive" method is the most commonly used method of designing and optimizing ANN architectures [32]. To improve the efficiency of the trial-and-error procedure, a genetic algorithm (GA) can be applied in order to effectively find the ANN architecture [33]. GA is a method for optimization based on the Darwinian theory that the survival of the generation is by natural selection [34]. The GA is significantly more efficient than the traditional trial-and-error method for designing ANN structures. Many studies have applied GA to optimize ANN parameters [34–40]. While there are several optimization methods for parameter tuning [41–45], each one of them has its own pros and cons. However, the GA method is considered to be accurate and fast [42–45]. Moreover, GA is suitable for parallel computing where multiple genetic algorithms are executed, and the best solution is selected from each algorithm. Then, these best solutions will be evaluated with each other, and the best among them is selected for training the model.

2.2. Optimization of Training an Artificial Neural Network

ANNs are trained using stochastic gradient descent optimization algorithms. This type of algorithm uses the training dataset to make a prediction with the current state of the model, then compares the prediction to the expected value and uses the difference as an estimate of the error gradient. This error gradient is then used to update the model weights; this process is repeated until the stop condition is fulfilled. The amount of change applied to the weights during training is referred to as the learning rate. The learning rate is a hyper-parameter in the range between 0 and 1. It controls the rate at which the model is adapted to the problem. Small learning rates require more training epochs, due to the smaller changes made to the weights in each cycle, whereas larger learning rates cause bigger changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck. The learning rate is considered one of the most important hyper-parameters for model training [25]. The error gradient is a statistical value; the more training examples are used, the more accurate the estimate, which will cause better adjustment of the ANN weights that will in turn improve the performance of the prediction [25]. The number of training examples used in the estimation of the error gradient is called a "batch" [25]. A batch size of N means that N samples from the training dataset will be used to estimate the error gradient. A training epoch means that the learning algorithm has made a single pass through the training datasets, which were separated into randomly selected batch groups. In the case of a small batch value, the error gradient will be less accurate and will be relevant mostly to the specific training examples used. Using a small number of examples may cause a noisy estimate that will have noisy updates to the model weights. Nevertheless, these noisy updates can result in faster learning and sometimes a more robust model. By using randomly selected batches for each epoch, generalization is achieved. In practice, smaller batch sizes are usually used

as they are noisy, offering a regularizing effect and a lower generalization error. In addition, smaller batch sizes are fitted to the use of GPUs as it is more likely to fit one full batch of training data in its memory. The commonly used batch size is usually 32 [25,46].

2.3. Machine Learning for Prediction Project Outcomes

The use of ML for prediction is very common [47–51], and in the last few years, many studies have also focused on the use of ML and ANN to predict project outcomes [52]. ANN and linear regression were used to estimate IT project efforts [20]; however, the methodology was too complicated to be generalized as a generic tool. Hsu et al. [53] compared two types of ANN for predicting project success and found that the differences between the formats of the organizations' data records were preventing these prediction methods from getting proper training. Machine learning was used to predict the performance of construction projects [54], but their outcome focus was more on project success parameters rather than actual estimation values. ANN performance was found superior compared to other multiple regression models in project effort estimation [55], and an algorithm for software project prediction [56] exists but does not fit other types of projects and is validated only with the quite old ISBSG public dataset. The common theme of all of these studies is that none of them presented a validated project duration prediction model that applied to real projects; none of the existing studies presents a practical AI duration prediction tool that is flexible enough to fit different types of projects and organizations and includes a prediction model which can be dynamically adjusted and optimized for any given organization's dataset.

3. Prediction Tool Description

The need of an organization to have a prediction of the project's duration is essential for many reasons [57,58], such as project portfolio planning, product roadmap planning, resource planning, budget planning, cash flow, customer satisfaction, and more. There are many differences between organizations' projects: the culture is different, the project management process, the tools and techniques, and much more. In some organizations, these differences exist also internally, between different sections of the same organization. This required a project duration prediction tool to be agnostic and flexible enough to fit many types of projects, products, and project management methods; it should support different types of inputs, with different types of units; and it should handle the biases and errors caused by these differences. While the input dataset of each organization can be different, the required output is common to all types of projects; the output is a single value, with units of time, which describe the predicted duration of the targeted project.

In this paper, we present a dynamic project duration prediction (DPDP) method. The DPDP methodology employs a supervised machine learning technique to build a predictive model, based on a dataset of features that encapsulate relevant characteristics of projects, to map true inputs to true output. As there are not many constraints regarding the inputs, it is required that the dataset include the actual duration of recorded projects used as an output for the model training. This prediction method adjusts and optimizes itself according to the supplied dataset. The DPDP creates a unique model per organization or a given dataset. The DPDP uses historical data from past projects and learns from them what future projects will look like according to the project features. The method is agnostic to many biases between organizations and project types. The biases are usually caused by different ways of measuring project features, such as risks, duration estimation, parameter definition, different project management procedures, different types of projects, etc. The DPDP method is relevant for all types of projects and products; it supports construction projects, as well as product development projects and more. The prediction model is based on ANN; the ANN was chosen as the preferred ML method as it can address almost any regression problem and supply decent results as long as it is constructed with the proper architecture and optimization. Nevertheless, the generalization comes with a price of not always being optimal in terms of computing resources. Therefore, this method is also

designed to work with multi-tread computing and GPU optimization to compensate for the efficiency of non-optimal resources and to reduce the model training and calculation time. Using ANN for producing quality outputs with different types of data requires re-adjustment, re-optimization, and adaptation for each type of input dataset. Usually, these adjustments are made by qualified personnel who use mostly trial and error and heuristic methods to optimize and adapt the tool for a new dataset. The DPDP includes a semi-automatic method to adjust and optimize the ANN to the different datasets, so no special qualification is needed to adjust the tool for the organization.

3.1. DPDP Tool Architecture

The DPDP method is constructed from two sequential phases: data processing, ANN model optimization, and model compilation. The data processing phase includes loading the data, cleaning it, and converting it into a format that the ANN can accept. In the ANN model optimization phase, the model is trained and tested multiple times until it is optimized to produce the most accurate prediction.

3.1.1. Data Processing Phase

The data are loaded automatically from a spreadsheet. They are required to have a certain structure: each line marks a different project; each column marks a project feature; the output feature column (the actual duration of the project) is the first column. After the data are loaded, there is a processing procedure that eliminates invalid data (e.g., features with an invalid value, missing data, etc.). Next, non-numeric data are encoded; as the ANN is built to process numeric data, inputs that are not numeric must be encoded to numeric values in order to be a valid input for the ANN model. The last stage is the normalization of the data; when using multiple inputs to an ANN model, the input variables may have different scales, as they represent different units. Differences in the scales across input variables may increase the complexity of the problem being modeled [59]. Therefore, this stage makes sure that all the input data are on the same scale by applying a normalization algorithm that scales all the input values to a scale of 0 to 1. The normalization algorithm we used is the min–max algorithm presented in Equation (1).

$$\text{Input}_{\text{Normalized}} = \frac{\text{Input} - \text{Min(input)}}{\text{Max(input)} - \text{Min(input)}} \quad (1)$$

3.1.2. ANN Model Optimization

After the data processing phase, the data are divided into three groups: training dataset, validation dataset, and testing dataset. The training dataset includes ~80% of the projects and is used for training the ANN. The validation dataset includes ~10% of the projects and is used as a validation reference during the training. The testing dataset includes the rest of the projects, and is used for testing, optimization, and scoring in each of the optimization cycles.

There are two different ways for dividing the datasets in case the dataset includes project dates or other indications for the precedence relation between the projects. The training dataset will include 80% of the earliest executed projects. The rest of the projects are randomly divided into the validation dataset and the testing dataset. The aim of this split methodology is to allow the ANN to be trained on old projects in order to predict the more recent project duration. In case there is no indication of the precedence relation between the projects, the data are randomly split into these three groups. The system also sets a lower boundary for the amount of data per group, which assures that the model will have enough data for training and testing.

The ANN model consists of an input layer, an output layer, and hidden layers. The output of the ANN model is the predicted duration which sets the layer size to a single artificial neuron. The input layer size is the number of features in the dataset which are being extracted. The chosen training optimizer for the method is Adam. Adam optimization is a

stochastic gradient descent method that is based on the adaptive estimation of first-order and second-order moments. The Adam optimizer is considered efficient in computing and memory; it is invariant to diagonal rescaling of gradients and is well suited for problems with large datasets or problems with many features [60]. The learning rate of the Adam optimizer is set to the value of 0.01. A batch size of 512 is set to allow small enough training data to fit in the GPU memory, as well as big enough data to allow utilization of the parallel calculation capability of the GPU. Then, the ANN training process begins with the training dataset using the validation dataset as training validation. The outcome of a training cycle is an ANN model able to predict a project's duration with a certain degree of accuracy. To optimize the ANN prediction, it is important to define the suitable number of hidden layers and the number of artificial neurons in each layer that yield the best prediction result. A genetic algorithm [33] is applied in order to find the most accurate ANN architecture. The entire testing dataset is routed to the trained ANN, in order to get the prediction value of the network. The differential between the ANN prediction value and the actual value (the error) is calculated. To score the method prediction accuracy, three types of errors are calculated: the mean absolute error (MAE) described by Equation (2), the root mean square error (RMSE) [61] described by Equation (3), and the mean absolute percent error (MAPE) [62], which is described by Equation (4).

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |T\_true_i - T\_prediction_i|}{n} \tag{2}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(T_{true\,i} - T_{prediction\,i}\right)^2}{n}} \tag{3}$$

$$\mathrm{MAPE} = \sum_{i=1}^{n} \frac{\left|T_{true\,i} - T_{prediction\,i}\right|}{n \cdot |T_{true\,i}|} \times 100\% \tag{4}$$

$T_{true\,i}$ represents the true duration of project i; $T_{prediction\,i}$ represents the ANN predicted duration of project i, and n represents the total number of projects in the dataset. A lower value of MAE, RMSE, and MAPE means a lower error which signifies better prediction accuracy. Using these types of error measurements allows one to have different points of view on the prediction accuracy. The MAE allocates an equal weight to all of the individual differences, and it is considered to be more accurate than the RMSE and MAPE [62]. In the RMSE, a relatively high weight is given to large errors, which is a good indication of the presence of a large error. The RMSE must be bigger or equal to the MAE; if the RMSE is equal to the MAE, then all the errors are of the same magnitude. The greater the difference between the RMSE and MAE, the greater the variance in the individual errors. The units of both MAE and RMSE are the same time units as the time units of the duration prediction, so it is easier to understand the duration error in terms of time. The MAPE is related to the percentage of the error from the true value, which allows a relative point of view on the results, but it also gives higher weight to an under-forecast value which can bias the results when compared with the MAE and RMSE.

The training cycle is repeated multiple times; each time the error is measured, and the genetic algorithm is applied in order to optimize the results. Last, the ANN model architecture with the lowest error value is selected to be saved and used as the trained model for the DPDP-based tool.

3.1.3. Genetic Algorithm

The GA was applied to optimize two of the main ANN hyper-parameters: the number of hidden layers and the number of neurons in each hidden layer. An initial population of networks (chromosomes) with different sets of these parameters (genes) was randomly created. The initial setting of the GA parameters was based on previous studies [33–36,39] and computational experiences. Chromosomes in the population pool contained two types

of genes: the number of hidden layers and the artificial neuron number of each hidden layer. The range of the number of hidden layers in the network is set to 1–20. The range of artificial neuron numbers in each hidden layer is set to 1–10 times the number of features. For example, if the number of the input features of the ANN is 5, the number of artificial neurons in each layer can be in a range of 5 to 50. The GA starts with 200 randomly generated chromosomes (networks), where all chromosomes in the population pool had at least one different parameter. The GA has three primary operations: reproducing, generating, and choosing new populations according to their fitness; crossover, parent choosing, recombination, and mutation in creating new generations. The fitness value of chromosomes in generations was evaluated according to the MAE value of the network prediction; a low MAE means better fitness. Reproduction evaluation was made according to the fitness value, by extracting the chromosomes that have a fitness value above the reproduction threshold. The reproduction threshold was set to the MAE of the traditional method prediction. Since in this case the parents may not have the same number of genes, a custom crossover and mutation algorithm was used, as described by Domashova [33]. The evolving networks are iterated through 100 generations, where at the end, the networks with the best fitness are the chosen networks for the provided dataset.

4. Dataset Description

In order to test the improvement in prediction accuracy of a tool based on the DPDP method, we used real-life project datasets collected from two different organizations. The first organization is a large global IT corporation, which produces an average of 1251 new IT projects each year. The organization's project datasets consist of a total of 20,024 recorded projects that were executed during the years 2005–2021. The second organization is a startup company that has developed multidisciplinary products including mechanics, electronics, and software. The organization produces an average of nine projects a year and a dataset that consists of a record of 26 projects executed during the years 2018–2021. The two datasets represent two extremes of organization size. One is a very big organization with a large record of project data; the organization has an established and structured project management methodology, with a lot of experience in executing projects that have similar properties. The other organization is a small startup that has not yet established a project management methodology and has little historical information about its projects. The two datasets were recorded differently by the corresponding organizations and were based on each organization's methods of defining and applying project features, such as task duration, risk level, etc. It is hard to evaluate the quality of these features, how accurate their estimations are, how relevant they are to the project outcomes, and how they correspond to other features of the organization. Nevertheless, a DPDP tool can overcome these biases in data as long as the methods of defining and estimating these features per dataset are consistent, as the tool optimizes for each organization and does not generalize across organizations. The DPDP tool learns the correlation between these feature estimations and the actual results per dataset, so the biases and the errors caused by different methodologies are diminished.

Tables 1 and 2 present the details of the project datasets provided by Organizations 1 and 2 respectively. As can be seen in these tables, although both datasets include 11 features per project, the description, and the units of the features are different between the organizations. Usually, it is necessary to build different prediction tools for each organization in order to use its features and optimize the AI model according to each organization's dataset. However, as described in this paper, the DPDP tool can accept both datasets and can create a unique ANN model for each organization, which is optimized to predict future project duration.

Table 1. Project Features of Organization 1.

No.	Feature Name	Notes
1	Project type	Minor R&D involvement—19,247 (~96%) (Procurement + Delivery only)—777 (~4%)
2	Business unit	Network integration services—13,587 (~68%) Customer support—4182 (~21%) Managed service and network assurance—776 (~4%) Managed service—613 (~3%) Software company service—515 (~2.5%) Others—351 (~1.5%)
3	Project mode	The data contain 293 different project modes: Standard—16,034 (~80%) 292Others—3990 (~20%)
4	Project risk level	Four levels of scheduled risks: Class A is the highest risk and D the lowest risk
5	Start day	The day the project kicked off
6	Start month	The month when the project kicked off
7	Start year	The year when the project kicked off
8	Planned end day	The expected day for project delivery
9	Planned end month	The expected month for project delivery
10	Planned end year	The expected year for project delivery
11	Project actual duration	The actual project duration (in weeks).

Table 2. Project Features of Organization 2.

No.	Feature Name	Notes
1	Project type	Operational project Logistic project Combined (a project with minor R&D involvement) Combined+ (a project with significant R&D involvement)
2	Product type	The type of products delivered in the project. (Type 1, Type 2, Type 3, Type 4, Type 5, Type 6)
3	Project estimated duration	The project estimated duration (in weeks) according to the Gantt chart at kick-off.
4	Stability of the project scope	A 3-point scale (High, Medium, Low) of what is the level of the certainty that the project scope is not likely to be changed?
5	Degree of risks for project delay	A 3-point scale (High, Medium, Low) of the level of certainty that the project will stay on schedule
6	Importance of time	A 3-point scale (High, Medium, Low) of how critical it is to deliver the project on time
7	Importance of cost	A 3-point scale (High, Medium, Low) of how critical it is to keep the project cost low
8	Experience with the technology	A 3-point scale (High, Medium, Low) of how much experience the organization has with the technology being used in the project
9	Level of details in the project plan	A 3-point scale (High, Medium, Low) of the detailed and resolution level of the project plan
10	Sub-contractor dependency	A 3-point scale (High, Medium, Low) of how much the project relies on sub-contractors
11	Project actual duration	The actual project duration (in weeks)

5. Results

This section presents the results of the analysis performed in order to evaluate and validate the DPDP tool. The analysis includes accuracy measurements and processing time performance analysis. The accuracy measurement was performed by applying the prediction tool to a real-life project dataset that was presented in the previous section, measuring the accuracy of the prediction results and comparing these predictions to the

traditional prediction methods used by the organizations. The accuracy measurements of the tool were based on calculating three types of errors: MAE, RMSE, and MAPE as presented in the previous section. The error calculations of the tool outcome were compared with the error calculations of the traditional predictions made by these organizations using their traditional methods. In both cases, the organizations used a Gantt chart and CPM in order to estimate the duration of the projects. The processing time performance analysis was conducted by comparing the processing time of sections in the DPDP tool using only the CPU (Central Processing Unit) versus running it with an additional GPU. The aim of this test is to understand the benefit of using this tool on a GPU-based platform and to find the optimized size of a training batch to fit such a tool.

5.1. Prediction Accuracy Test

The results of the accuracy test are summarized in Tables 3 and 4. They reveal that the DPDP tool prediction accuracy is superior to the traditional method. The tables show a comparison between the MAE, RMSE, MAPE, and max error (the highest measured error) of the two methods for each of the dataset partitions (A, B, and C). The A, B, and C datasets were created by random partitioning of the data for the testing set and validation set. The partition was performed three times and yielded three different validation and testing sets A, B, and C. This was done in order to reduce the bias that a single test may create. Table 3 compares Organization 1's duration prediction error and Table 4 relates to Organization 2's duration prediction error. As can be seen in both tables, the DPDP tool's MAE is significantly lower than that of the traditional method on all of the testing sets in both organizations. Comparing the RMSE with the MAE of each method reveals the cases in which the variance of the error is large. For example, in Table 3, the value of the traditional method's MAE in testing set A is 54.92 while the value of the RMSE is 83.18; the differential between these values is significant (more than 50%), which implies that many large errors exist. The average on Organization 1's traditional prediction MAE is 49.18 weeks and the average RMSE is 71.69 weeks, which results in an average of ~47% difference; by comparison, the DPDP tool, which has an average MAE of 24.25 weeks and an average RMSE of 30.12 weeks, results in a 24% difference. Organization 2's results show the same trend with 37% in the traditional method vs. only 15% with the DPDP tool. The MAE and the RMSE give an absolute number in the same time units of the predicted duration. This makes it easy to understand the practical value of the error, but it does not help in understanding the magnitude of the error relative to the project length. For example, a 2-week prediction error will be significant when referring to a 5-week project, but the same error would be considered low when referring to a 200-week project. The MAPE value of each method gives a good indication of the error magnitude relative to the project length. For example, in Table 4, the MAE values are in single digits, seemingly lower than the ones in Table 3 which have two-digit errors. Nevertheless, according to the MAPE value, one can deduce that in terms of the error relative to the length of the project, there is not much of a difference as in both tables the magnitude of the MAPE is approximately the same. As can be seen in both tables, the average relative error of the DPDP is consistently lower in all instances.

The max error indication focuses on one extreme use case with the biggest prediction error. Even though this value alone is not enough for accuracy determination, presenting it near the other average errors helps to understand more about the robustness of the prediction tool and its suitability to extreme cases of variance and uncertainty that projects bring with them. The results show that the DPDP can predict uncertainties better than the traditional methods and also give better predictions in extreme cases.

Measuring the variance of the errors between the different testing datasets allows a point of view on the tool's immunity to noise; when the variance is high, it means that the error of the prediction is easily affected by different types of projects, while low variation points to a stable prediction and adaptation to different types of projects. In

both organizations, the DPDP tool shows improvement in the variation errors, while in Organization 1, with the larger dataset, the improvement is more significant.

Table 3. Comparing Organization 1's traditional vs. DPDP prediction accuracy.

Prediction Method	Testing Set	MAE (Weeks)	RMSE (Weeks)	MAPE (%)	Max Error (Weeks)
Traditional	A	54.92	83.18	190.13%	272
DPDP	A	25.47	32.96	34.98%	92.18
Traditional	B	37.27	55.49	29.01%	237.71
DPDP	B	20.93	26.32	15.62%	68.37
Traditional	C	55.35	76.4	52.75%	248.86
DPDP	C	26.36	31.07	24.33%	71.02
Traditional Average	A/B/C	49.18	71.69	91%	252.86
DPDP Average	A/B/C	24.25	30.12	25%	77.19
Traditional Variance	A/B/C	70.95	138.88	–	203.95
DPDP Variance	A/B/C	5.65	7.80	–	113.52

Table 4. Comparing Organization 2's traditional vs. DPDP prediction accuracy.

Prediction Method	Testing Set	MAE (Weeks)	RMSE (Weeks)	MAPE (%)	Max Error (Weeks)
Traditional	A	5.75	8.13	21.67%	15.57
DPDP	A	4.48	4.77	25.7%	7.29
Traditional	B	6.11	8.74	18.75%	15.57
DPDP	B	3.02	4.15	8.99%	7.92
Traditional	C	8.54	11.11	31.14%	15.71
DPDP	C	4.56	4.99	17.53%	6.46
Traditional Average	A/B/C	6.80	9.33	24%	15.62
DPDP Average	A/B/C	4.02	4.64	17%	7.22
Traditional Variance	A/B/C	1.54	1.65	–	0.004
DPDP Variance	A/B/C	0.50	0.13	–	0.36

5.2. Generated Model Structure

The prediction models generated for each use case had different hyper-parameters, which were set using the GA, as explained in Section 3.1.3. For each organization and separately for each validation set (A, B, and C), the DPDP tool created different models with different hyper-parameters (a total of six ANNs). Each model was optimized according to the validation dataset loaded into the DPDP tool. The training optimizer in all cases was set as Adam (as presented in Section 3.1.2) and the learning rate was set as 0.01. Table 5 presents the ANN layers and artificial neurons in each layer per network and validation set.

Table 5. Number of layers and neurons per model and validation set.

Layer	Org 1—A	Org 1—B	Org 1—C	Org 2—A	Org 2—B	Org 2—C
Input	10	10	10	15	15	15
Hidden 1	154	110	66	105	285	105
Hidden 2	33	77	154	225	120	255
Hidden 3	-	198	77	-	-	255
Hidden 4	-	176	99	-	-	195
Hidden 5	-	110	187	-	-	30
Output	1	1	1	1	1	1

As can be seen in Table 5, the output layer contains a single neuron as there is only one output (project duration). The input layer is set as the number of features. Both organizations' (1 and 2) datasets contain 10 features, but the feature "Project Type" in Organization 2 was decoded using one-hot encoding [63], which increases the number of

input features to 15. The number of layers and number of artificial neurons in each layer were optimized for the best accuracy according to the GA.

5.3. CPU/GPU Performance Analysis

As GPUs can perform multiple, simultaneous computations based on their thousands of cores, this trait can be used for the distribution of training processes and significantly speed up the ANN training procedure. Thus, the dataset size and mostly the batch size which is set for the training process need to be optimized for the GPU and its characteristics. The analysis was performed using the hardware presented in Table 6. The same hardware was used for both CPU and GPU experiments, by enabling or disabling the GPU functionality. In addition, as the tool uses randomization algorithms, we overwrite the randomization with fixed values in order to allow the same execution in all experiments. The DPDP tool uses multiple training sessions in order to optimize the ANN. For the presented test, we performed only one training session as a reference to a multi- session procedure. The tested ANN architecture includes five layers with 11, 55, 110, 55, and 1 artificial neuron(s), respectively, in each layer. The activation function was set as a *rectified linear unit* [64]. In addition, a fixed number of epochs of 100 were set in order to reduce the randomization of weight selection.

Table 6. Server Hardware Properties.

Feature	CPU	GPU
Model	Intel(R) Xeon(R) CPU	NVIDIA Tesla-K80
No. of Cores	2	2496 CUDA
CPU Clock	2.3 GHz	820 MHz
RAM	12 GB	12 GB
Cache Memory	39,424 KB	–

The experiment includes measuring the effect of changing the *batch size* parameter of the ANN training. Increasing the *batch size* will cause a larger matrix multiplication; since the matrix multiplications are parallelizable operations, increasing the *batch size* is expected to shorten the calculation time of the GPU compared to the CPU. Above a certain *batch size* value, a full batch of training data cannot fit into the GPU memory, which might cause a deceleration in the GPU performance. Figure 2 presents the results of the CPU vs. GPU experiment. As can be seen in Figure 2, the small *batch size* has a longer running time; as the *batch size* increases, the running time of the training gets shorter until a certain point where the CPU and the GPU act differently. With a small batch size, there is no advantage of using GPU over CPU; the suggested reason for that is the fact that the CPU clock is higher than the GPU clock, and because the amount of data is small, the overhead of loading the data to the GPU memory is greater than the improvement in performance by the parallel calculation method. As the *batch size* increases, the advantage of the GPU is manifested and the GPU reaches a 250% faster processing time compared to the CPU. It can be seen in Figure 2 that above a *batch size* of 1024, the running time of the GPU stays steady (10 s for 100 epochs) even though the *batch size* increases, while the CPU running time gets longer. One suggested explanation for this phenomenon is that with a *batch size* above 1024, the data loading time to the memory is negligible compared to the calculation time, and as the calculation process by the GPU is performed in parallel; as long as the GPU's cores are not fully loaded, there is no major change in the calculation duration even when the amount of data increases. However, in the CPU, it seems that above a *batch size* of 512 the serial calculation takes longer than the parallel one on the GPU, even though the clock speed of the CPU is higher.

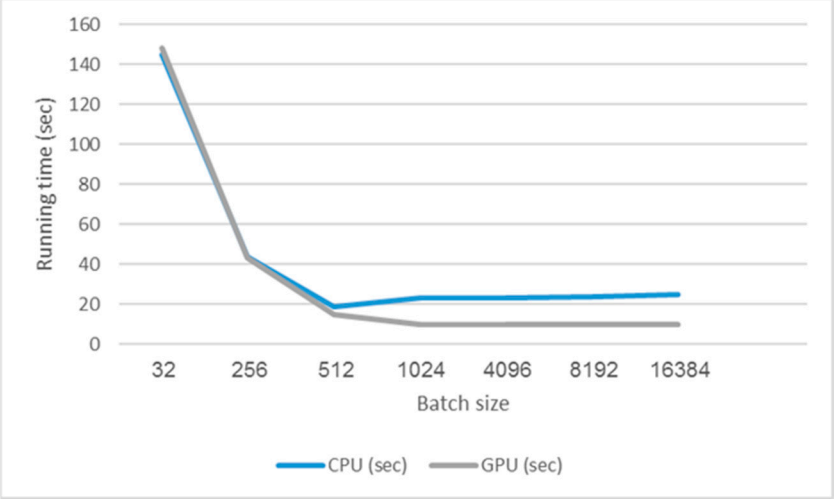

Figure 2. GPU vs. CPU running time.

6. Discussion

The DPDP method showed that it fits both small and large organizations with different sizes of datasets. It showed that it is flexible enough to get different types of features, with different scales and units, and in all the tested cases it out-performed the other traditional prediction methods. Nevertheless, the small organization had a less significant improvement compared to the large organization. The reason is that the amount and the quality of the data directly affect the ANN training and the accuracy of the DPDP results. The small organization had provided a very small number of recorded projects, which provided fewer examples for the tool to learn from. Because of this, an organization with larger historical data will more likely derive a greater benefit from this tool.

An important aspect that stood out in the results was the immunity of the DPDP method to noises. The differences between datasets hardly affected the average error of the DPDP prediction. The three datasets A, B, and C were randomly divided and used for training, validation, and accuracy tests. The variance between the three groups is perceptible by looking at the traditional method prediction results; group A in Organization 1 seems to include many projects that experienced issues that were not anticipated by the traditional prediction method. This is shown by the values of max error, MAPE, and the differences between MAE and RMSE. However, the DPDP method's variance in errors between the groups is quite small, and the results were not much different than in the other dataset groups. This fact indicates the capability of the DPDP method to provide a stable average error even with projects that are outliers in some way. In all of the performed tests, the variance of the errors had significantly decreased compared to the other traditional methods. This is an important feature for a prediction tool, as it helps to have more certainty in the prediction error range, which is important when evaluating the project risks.

To keep the DPDP tool up to date and to maintain its accuracy, it is recommended to retrain the tool often with new records of project data. In some organizations, it can get to a point where this training is being done daily or even a couple of times a day. Smaller datasets will probably need frequent retraining, as every record is more significant. In large datasets adding a small number of records will probably not affect the results too much, so less frequent retraining is required. As training ANN with a large dataset consumes a lot of computing resources and may take a long time, the use of a GPU with this tool is recommended in order to produce an outcome in a reasonable timeframe. The reduction in time can be effective with the use of a GPU as the model is being trained hundreds of times until the optimal architecture is found. The CPU vs. GPU experiment results provided insights into how to optimize the ANN training parameters in order to take advantage of and maximize the GPU capabilities. The GPU performance analysis had shown an improvement of 250% in processing time compared to the CPU. Nevertheless, as today's

tools are moving to cloud computing, the cost of running time is also a factor to take into account when considering which hardware to use. In cloud computing, the GPU server cost is significantly higher than the CPU cost. At the time of conducting this study, the cost of a cloud GPU can be more than 1000% higher than a cloud CPU [65]. The organization that chooses to use a tool based on the DPDP method will need to find the balance between the time required versus the cost of GPU cloud computing. An important limitation of this tool is its optimization per project. Although the DPDP method significantly improved the prediction accuracy, it is not the optimal solution for each project.

7. Conclusions

The DPDP method is built to fit different types of projects and different datasets; it cannot be optimal for all of them. A dedicated model designed for a specific project or type of dataset can potentially produce better results than the presented tool but will require an additional effort and redesign to fit other types of data, which is not always practical for an organization. Future work can include more optimization of the DPDP tool with additional validation on more datasets, finding the upper and lower boundaries of the amount of data this tool has and improving its performance. In addition, a future study can compare the performance of the presented DPDP tool with other AI prediction tools built and optimized according to the project type, data type, or organization. Such a study would measure the expected improvement in prediction accuracy and try to find the point of equivalence in resources, time, cost, etc. of building a dedicated prediction model vs. using an existing general prediction tool.

Author Contributions: Conceptualization, I.L. and A.S.; methodology, I.L. and A.S.; software, I.L.; validation, I.L. and A.S.; formal analysis, I.L.; investigation, I.L.; resources, I.L. and A.S.; data curation, I.L.; writing—original draft preparation, I.L.; writing—review and editing, A.S.; visualization, I.L.; supervision, A.S.; project administration, A.S.; funding acquisition, A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data are presented in the main text.

Acknowledgments: We would like to thank all who gave us support to complete this paper.

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

Abbreviations

ANN	Artificial Neural Network
GPU	Graphics Processing Unit
CPM	Critical Path Method
PERT	Program Evaluation and Review Technique
AI	Artificial Intelligence
ML	Machine Learning
GA	Genetic Algorithm
DPDP	Dynamic Project Duration Prediction
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percent Error

References

1. *Project Management Body of Knowledge (PMBOK® Guide 7th Edition)*; Project Management Institute: Newton Square, PA, USA, 2021.
2. Lishner, I.; Shtub, A. Measuring the success of Lean and Agile projects: Are cost, time, scope and quality equally important? *J. Mod. Proj. Manag.* **2019**, *7*, 139–145.
3. Barlow, G.; Tubb, A.; Riley, G. Driving business performance: Project Management Survey 2017. *Wellingt. N. Z. KPMG N. Z.* **2017**.
4. The Standish Group. CHAOS Manifesto Report. 2015.
5. PWC. The Third Global Survey on the Current State of Project Management. 2012.

6. Szwarcfiter, C.; Herer, Y.T.; Shtub, A. Project scheduling in a lean environment to maximize value and minimize overruns. *J. Sched.* **2022**, *25*, 177–190. [CrossRef]
7. Hanzalek, Z.; Hanen, C. The impact of core precedences in a cyclic RCPSP with precedence delays. *J. Sched.* **2014**, *18*, 275–284. [CrossRef]
8. Ashtiani, B.; Leus, R.; Aryanezhad, M.-B. New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing. *J. Sched.* **2009**, *14*, 157–171. [CrossRef]
9. Gantt, H.L. A Graphical Daily Balance in Manufacture. *Trans. Am. Soc. Mech. Eng.* **1903**, *24*, 1322–1336.
10. Petersen, P.B. The evolution of the Gantt chart and its relevance today. *J. Manag. Issues* **1991**, *3*, 131–155.
11. Wilson, J.M. Gantt charts: A centenary appreciation. *Eur. J. Oper. Res.* **2003**, *149*, 430–437. [CrossRef]
12. Moore, D.A.; Healy, P.J. The trouble with overconfidence. *Psychol. Rev.* **2008**, *115*, 502–517. [CrossRef]
13. König, C.J. Anchors distort estimates of expected duration. *Psychol. Rep.* **2005**, *96*, 253–256. [CrossRef]
14. Hill, J.; Thomas, L.C.; Allen, D.E. Experts' estimates of task durations in software development projects. *Int. J. Proj. Manag.* **2000**, *18*, 13–21. [CrossRef]
15. Josephs, R.A.; Hahn, E.D. Bias and accuracy in estimates of task duration. *Organ. Behav. Hum. Decis. Process.* **1995**, *61*, 202–213. [CrossRef]
16. Burt, C.D.B.; Kemp, S. Construction of activity duration and time management potential. *Appl. Cogn. Psychol.* **1994**, *8*, 155–168. [CrossRef]
17. White, R.W.; Awadallah, A.H. Task duration estimation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, Australia, 11–15 February 2019; pp. 636–644.
18. König, C.J.; Wirz, A.; Thomas, K.E.; Weidmann, R.-Z. The effects of previous misestimation of task duration on estimating future task duration. *Curr. Psychol.* **2014**, *34*, 1–13. [CrossRef]
19. López-Martin, C.; Chavoya, A.; Meda-Campaña, M.E. Use of a feedforward neural network for predicting the development duration of software projects. In Proceedings of the 12th International Conference on Machine Learning and Applications, Miami, FL, USA, 4–7 December 2013; Volume 2, pp. 156–159.
20. Berlin, S.; Raz, T.; Glezer, C.; Zviran, M. Comparison of estimation methods of cost and duration in IT projects. *Inf. Softw. Technol.* **2009**, *51*, 738–748. [CrossRef]
21. Anderson, J.A. *An Introduction to Neural Networks*; MIT Press: Cambridge, MA, USA, 1995.
22. Reed, R.; Marks, R.J., II. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*; MIT Press: Cambridge, MA, USA, 1999.
23. Tetko, I.V.; Livingstone, D.J.; Luik, A.I. Neural network studies. 1. Comparison of overfitting and overtraining. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 826–833. [CrossRef]
24. Lippmann, R.P. Multi-style training for robust isolated-word speech recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, TX, USA, 6–9 April 1987; Volume 4.
25. Heaton, J.; Goodfellow, I.; Bengio, Y.; Courville, A. Deep learning. *Genet. Program. Evolvable Mach.* **2017**, *19*, 305–307. [CrossRef]
26. Kolmogorov, A.N. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*; Russian Academy of Sciences: Moscow, Russia, 1957; Volume 114, pp. 953–956.
27. Weigend, A. On overfitting and the effective number of hidden units. In Proceedings of the 1993 Connectionist Models Summer School, Lawrence Erlbaum Associates, NJ, USA, 1994; Volume 1, pp. 335–342.
28. Sarle, W.S. Stopped training and other remedies for overfitting. *Proc. 27th Symp. Interface Comput. Sci. Stat.* **1996**, 352–360.
29. Berry, M.J.A.; Linoff, G.S. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*; John Wiley & Sons: Hoboken, NJ, USA, 2004.
30. Swingler, K. *Applying Neural Networks: A Practical Guide*; Morgan Kaufmann: San Francisco, CA, USA, 1996.
31. Boger, Z.; Guterman, H. Knowledge extraction from artificial neural network models. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; Volume 4, pp. 3030–3035.
32. Zhong, Z.; Yan, J.; Wu, W.; Shao, J.; Liu, C.-L. Practical block-wise neural network architecture generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2423–2432.
33. Domashova, J.V.; Emtseva, S.S.; Fail, V.S.; Gridin, A.S. Selecting an optimal architecture of neural network using genetic algorithm. *Procedia Comput. Sci.* **2021**, *190*, 263–273. [CrossRef]
34. Mahajan, R.; Kaur, G. Neural networks using genetic algorithms. *Int. J. Comput. Appl.* **2013**, *77*, 6–11. [CrossRef]
35. Idrissi, M.A.J.; Ramchoun, H.; Ghanou, Y.; Ettaouil, M. Genetic algorithm for neural network architecture optimization. In Proceedings of the 2016 3rd International Conference on Logistics Operations Management (GOL), Fez, Morocco, 23–25 May 2016; pp. 1–4.
36. Abbasi, H.; Seyedain Ardabili, S.M.; Mohammadifar, M.A.; Emam-Djomeh, Z. Comparison of trial and error and genetic algorithm in neural network development for estimating farinograph properties of wheat-flour dough. *Nutr. Food Sci. Res.* **2015**, *2*, 29–38.
37. Majdi, A.; Beiki, M. Evolving neural network using a genetic algorithm for predicting the deformation modulus of rock masses. *Int. J. Rock Mech. Min. Sci.* **2010**, *47*, 246–253. [CrossRef]

38. Mohebbi, A.; Taheri, M.; Soltani, A. A neural network for predicting saturated liquid density using genetic algorithm for pure and mixed refrigerants. *Int. J. Refrig.* **2008**, *31*, 1317–1327. [CrossRef]
39. Saemi, M.; Ahmadi, M.; Varjani, A.Y. Design of neural networks using genetic algorithm for the permeability estimation of the reservoir. *J. Pet. Sci. Eng.* **2007**, *59*, 97–105. [CrossRef]
40. Kim, G.-H.; Yoon, J.-E.; An, S.-H.; Cho, H.-H.; Kang, K.-I. Neural network model incorporating a genetic algorithm in estimating construction costs. *Build. Environ.* **2004**, *39*, 1333–1340. [CrossRef]
41. Gupta, T.K.; Raza, K. Optimization of ANN architecture: A review on nature-inspired techniques. *Mach. Learn. Bio-Signal Anal. Diagn. Imaging* **2019**, 159–182. [CrossRef]
42. Alibrahim, H.; Ludwig, S.A. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June 2021–1 July 2021; pp. 1551–1559.
43. Acharya, R.Y.; Charlot, N.F.; Alam, M.M.; Ganji, F.; Gauthier, D.; Forte, D. Chaogate parameter optimization using bayesian optimization and genetic algorithm. In Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 7–9 April 2021; pp. 426–431.
44. Trotter, M.; Liu, G.; Wood, T. Into the storm: Descrying optimal configurations using genetic algorithms and bayesian optimization. In Proceedings of the 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W), Tucson, AZ, USA, 18–22 September 2017; pp. 175–180.
45. Roman, I.; Ceberio, J.; Mendiburu, A.; Lozano, J.A. Bayesian optimization for parameter tuning in evolutionary algorithms. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4839–4845.
46. Masters, D.; Luschi, C. Revisiting small batch training for deep neural networks. *arXiv* **2018**, arXiv:1804.07612.
47. Mosavi, A.; Ozturk, P.; Chau, K.W. Flood prediction using machine learning models: Literature review. *Water* **2018**, *10*, 1536. [CrossRef]
48. Neu, D.A.; Lahann, J.; Fettke, P. A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artif. Intell. Rev.* **2021**, *137*, 106024. [CrossRef]
49. Bertolini, M.; Mezzogori, D.; Neroni, M.; Zammori, F. Machine Learning for industrial applications: A comprehensive literature review. *Expert Syst. Appl.* **2021**, *175*, 114820. [CrossRef]
50. Van Klompenburg, T.; Kassahun, A.; Catal, C. Crop yield prediction using machine learning: A systematic literature review. *Comput. Electron. Agric.* **2020**, *177*, 105709. [CrossRef]
51. Guo, Z.; Chen, L.; Gui, L.; Du, J.; Yin, K.; Do, H.M. Landslide displacement prediction based on variational mode decomposition and WA-GWO-BP model. *Landslides* **2020**, *17*, 567–583. [CrossRef]
52. Wen, J.; Li, S.; Lin, Z.; Hu, Y.; Huang, C. Systematic literature review of machine learning based software development effort estimation models. *Inf. Softw. Technol.* **2012**, *54*, 41–59. [CrossRef]
53. Hsu, M.-W.; Dacre, N.; Senyo, P.K. Applied algorithmic machine learning for intelligent project prediction: Towards an AI framework of project success. *Adv. Proj. Manag.* **2021**, 21. [CrossRef]
54. Ling, F.Y.Y.; Liu, M. Using neural network to predict performance of design-build projects in Singapore. *Build. Environ.* **2004**, *39*, 1263–1274. [CrossRef]
55. de Barcelos Tronto, I.F.; da Silva, J.D.S.; Sant'Anna, N. An investigation of artificial neural networks based prediction systems in software project management. *J. Syst. Softw.* **2008**, *81*, 356–367. [CrossRef]
56. Pospieszny, P.; Czarnacka-Chrobot, B.; Kobylinski, A. An effective approach for software project effort and duration estimation with machine learning algorithms. *J. Syst. Softw.* **2018**, *137*, 184–196. [CrossRef]
57. Alami, A. Why do information technology projects fail? *Procedia Comput. Sci.* **2016**, *100*, 62–71. [CrossRef]
58. Majid, I.A. *Causes and Effects of Delays in ACEH Construction Industry*; Universiti Teknologi Malaysia: Johor Bahru, Malaysia, 2006.
59. Bishop, C.M. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
60. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
61. Willmott, C.J.; Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Clim. Res.* **2005**, *30*, 79–82. [CrossRef]
62. Goodwin, P.; Lawton, R. On the asymmetry of the symmetric MAPE. *Int. J. Forecast.* **1999**, *15*, 405–408. [CrossRef]
63. Brownlee, J. Why One-Hot Encode Data in Machine Learning; Machine Learning Mastery. 2017. Available online: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (accessed on 1 November 2022).
64. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* **2018**, arXiv:1803.08375.
65. Google Cloud GPU Pricing. Available online: http://cloud.google.com/compute/gpus-pricing (accessed on 1 July 2022).