2021 IEEE Smart Information Systems and Technologies (SIST)
28-30 April, 2021, Nur-Sultan, Kazakhstan

Neural Network Estimation Model to Optimize Timing and Schedule of Software Projects

Mohamed A. Hamada[1], Abdelrahman Abdallah[2*], Mahmoud Kasem[3], Mohamed Abokhalil[4]
[1]Department of Information System, International IT University, Kazakhstan, Dr.mmhamada@gmail.com
[2*]Department of Machine Learning & Data Science, Satbayev University, Kazakhstan, abdoelsayed2016@gmail.com
[3]Department of Machine Learning & Data Science, Satbayev University, Kazakhstan, mahmouds.192@gmail.com
[4]Department of Information Technology, Assiut University, Egypt, mohammed1994hamdyy@gmail.com

2021 IEEE International Conference on Smart Information Systems and Technologies (SIST) | 978-1-7281-7470-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/SIST50301.2021.9465887

***Abstract* — Software projects have a probability of high failure rates that appear to linger around 60% for significant IT projects. Estimating time and project schedule are crucial tasks and extremely influence the project outcomes. Artificial Intelligence now can provide multiple solutions for most problems of software projects. This article aims to develop a Neural Network estimation model to manipulate the problem of timing for software projects. The model can predict the estimation value of project time which optimizes the scheduling process, the developed model achieved high accuracy after testing through the test datasets.**

Keywords— deep learning; neural networks; project management; artificial intelligence; time optimization

I. INTRODUCTION

The growing interest in the field of software project development leads to an increase in the number of software products that are developed by IT companies for solving various business problems [1]. Estimation of timing and schedule one of the main project development activity which causes the failure of software development [2,3]. This is due to an incorrect assessment of the complexity of the project, Technical defects in software products, improper qualifications of developers, and other factors. Improving the accuracy of determining the complexity of the project and predicting the time of software development will greatly simplify and systematize the management of IT projects, as well as save resources, which enriches the value of this research.

Accordingly, improving the accuracy of estimating terms, labor intensity in man-hours, and other parameters in the future can significantly reduce financial costs and losses from the project going beyond the allotted time [1,3].

Since the 80s of the 20th century, many methods have been developed for calculating the term and laboriousness, but they have several disadvantages and are more adapted to the outdated procedural style of development.

Also, in recent years areas of neural networks, machine learning, artificial intelligence has been actively developing. Models built using these methods, with proper training and tuning on the right data sets, can generate accurate results that are approaching or even exceeding an expert estimate [4,5,6,7,8,9,10].

Therefore, this work is devoted to the research and analysis of evaluation methods, models based on neural networks, and the construction of an artificial neural network model that can effectively assess the timing and complexity of an IT project [11,12].

The purpose of the research is to test the suitability of classical estimation methods superimposed on a neural network model for evaluating the parameters of a wide range of various IT projects.

Our research objectives are:

- Analysis of classical methods for evaluating project parameters.
- Analysis of modern assessment methods
- Development of a model for the evaluation of modern IT projects
- Testing the developed model on current datasets

The subject of the research is a preliminary assessment of the properties of the project (time and labor intensity). The object of the research is IT projects, which include the development of software, databases, information systems, software, and hardware systems, as well as neural networks as an evaluation method.

The research methodological apparatus uses such methods as the research of articles and scientific sources, the analysis of neural network architectures, data collection, markup, and modeling, followed by a comparative analysis of the results.

The main hypothesis of the research is that with the help of a correctly constructed and trained neural network model, based on the parameters of COCOMO [13,14,18] systems and others, it is possible to estimate the actual terms of the project with sufficient accuracy (while analyzing the qualifications of employees, possible risks from the customer, etc.).

This research has a high degree of novelty, because, in the field of constantly changing technologies, IT project management methods (Scrum, Agile, etc.), and approaches, the old methods and evaluation algorithms no longer work, or require many hard-to-calculate parameters (function points, SLOC) for the correct evaluation. Accordingly, they are not suitable for new IT

projects and a more adapted solution is needed. The research has sufficient theoretical significance (improving and combining existing methods, testing them on new data sets) and highly practical, since the developed model with some adaptation will improve the accuracy of the estimation, reduce the financial and reputation costs of software companies, and information systems.

The main results of the research – the developed model – were tested on the main datasets of the NASA [15] and ISBG [16] series, showing high accuracy. Tests were also conducted on a limited number of real IT projects from the areas of web projects and VR projects.

The project structure consists of an analysis of existing project evaluation methods, an overview of the data sets used, a description of the architecture of the developed model, and its testing on data sets.

II. LITERATURE REVIEW

Software development cost and duration estimation research areas have a long and interest history from the 20th century to the current day. It successively passed stages from expert judgment, simple basic algorithms, statistic regressions to machine learning and neural network.

Each of the models developed by the community has several advantages and disadvantages.

Thus, expert evaluation and the Delphi method [17] allow you to generalize and use the experience of experts and work well in conditions when projects are quite similar and put-on stream (for example, in outsourcing companies, in a well-coordinated team with assigned development processes, with experienced project managers). However, they are subject to human factors, and they also tend to systematically overestimate or underestimate the timing or complexity of the project.

The COCOMO[18] evaluation method and its improved version, COCOMO II, allow you to evaluate the timing and complexity of project development, but the accuracy of their work strongly depends on how detailed the model was selected, as well as in what units the complexity of the project is measured. Due to its age, the evaluation method does not consider the features of modern IT projects development, other design, and development principles (object-oriented model, components, microservices, reuse of code and libraries, increased number of frameworks and plugins).

Genetic algorithms [19] differ in several features and usually, the quality of their evaluation depends heavily on the implementation features. Each company needs to develop a suitable algorithm again. Also, due to its nature, evaluating the timing and complexity of a project using a genetic algorithm can take up many computing resources, as well as have a nonlinear complexity depending on the number of input parameters.

Models based on neural networks [5,6] usually do not require a thorough detailed study of the logical model (which is often virtually impossible to develop correctly for any large project). We can also select the data that is relevant to the project and sort it by priority. The technique of training a neural network on input data provides great opportunities. Many available architecture options make it possible to choose the appropriate one or test several possible architectures. Not the last place in the list of advantages of models based on neural networks is occupied by the worldwide trend to popularize neural networks, which gives a huge advantage in the ability to choose the infrastructure. Many cloud providers offer ready-made services for configuring, training, validating, and running the model. A huge number of development environments (Jupyter Notebook, Anaconda, RStudio, SAS, and others) allows you to gradually train internal specialists to work with this technology.

However, models based on neural networks and using machine learning have several disadvantages. The model can detect false correlations where there are none. For example, we feed the model the volume values of test databases [8,9,10], and the model begins to consider the performance of the command to depend on the size of the databases, which may be incorrect.

A big problem is the dependence of models on the quality and quantity of input data. It is quite difficult to collect an array of data of acceptable quantity and quality, especially in the field of software development and maintenance.

The best choice of model depends on the context project specifics, development team, and model requirements. If all processes and their development properties are standardized and do not change, and top managers need a common tool for approximate parameter estimation, the proven COCOMO and COCOMO II [13,14,18] will be a good choice. However, if the development takes place in an environment of constantly changing requirements (flexible methodologies, Scrum), technologies (new programming languages, frameworks, technologies, and methodologies (DevOps), then the classic algorithms for assessing complexity will not work here and you will need to develop your model using new technologies.

Neural networks play a key role in the developed models. It is their inputs that provide data about the project, the composition, features of the project team, requirements, customer characteristics, environmental properties, and others. The data format is adapted to the features of the neural network and its architecture.

The area of evaluating the timing, complexity, and other parameters of the project requires the development of a universal framework/information system with the possibility of changing the characteristics of the evaluated project and the project team [28,29].

In this paper, we have developed a model that aims to provide a universal evaluation of software projects, regardless of the scope and type of project, with

the ability to adapt to a specific project team and environment.

Further development and research in this area, improvements, and adaptation of neural networks in the future can significantly change the area of project management related to software development. Regular and widespread use of adaptive models and frameworks for estimating deadlines and costs can reduce the frequency of budget overruns or delays. This step will increase the profitability of the IT industry and make it almost as predictable as the more conservative technical sectors – construction, aviation, and mechanical engineering, which will have an exceptionally positive impact on the global economy.

III. RESEARCH METHODOLOGY

The methods used in this work relate both to work with data processing and preparation, as well as to the development and verification of the model itself for estimating the time and complexity of software.

a. Data processing

Properly processed and systematized data should be used as the data that serves as the basis for the development and construction of the model. Since this work is devoted to estimating the timing and complexity of software development, the main source data here are the properties and parameters of existing projects.

Before bringing existing and collected data to a general view, we need to formulate a list of variables and their format, i.e., in essence, to form a sample to which all external experimental data will be converted.

The main common data that the model should have at the input to correctly estimate the timing and complexity of software development are the following:

1. Project complexity
2. The number, structure, and experience of developers
3. Experience of the project manager and his assistants
4. «Quality» of the customer – how well he or she sets tasks, evaluates deadlines, sends requirements with changes, frequency, and quality of communications.
5. The ability to reuse the code or modules.
6. Used technologies, languages, frameworks

Based on these requirements, we can create a general list of variables-columns that must be in the dataset to ensure that they can be used in the development or validation of the model:

1. Project year
2. Platform
3. The number of development languages
4. The number and experience of the development team
5. Project duration in months
6. The size of the project
7. The design
8. Type of development (new project or support/revision of an existing one)
9. Percentage of reusing code and modules
10. Reliability requirements
11. Architecture.

b. Research methods

Theoretical methods [20] are intended for researching without direct interaction with the object. The theoretical methods used in this paper are listed below.

1. *The analysis* is the mental division of an object into its parts, elements, and attributes.
2. Synthesis is the opposite process of combining the parts and features of an object selected during analysis into a single whole. In this paper, the analysis was used to find ways to solve the main goal of the study – to estimate the timing and complexity of software development. To do this, the evaluation process itself was divided into a set of features that should be evaluated (Complexity, Duration, and Laboriousness)

Empirical methods [20] used in this work include the study of literature, documents, and results of activities, measurement, and expert assessments. The study of literature, documents, and research on the topic is the main empirical method of this work. Since the topic of this research is on the one hand quite extensive (due to the intersection with the research on neural networks and machine learning), and on the other hand – small (due to the small number of serious works on this topic), it becomes obvious that a special protocol for systematic study and review of literature should be developed. A diagram of the literature selection and sorting process is shown in Fig 1.

c. Datasets

Existing datasets (Kemerer, Maxwell, Albrecht, and others) [21,22,23] have a low relevance and low compatibility with each other, and this work has collected data about projects based on the author's own experience as a developer. All information was grouped into 13 main features during analysis:

- Name is the name of the project. The information characteristic for distinguishing project during debugging, configuration, and testing of the model. For some projects, the real name was changed to a conditional one for data protection reasons.
- Year is the year when the project started developing. Information characteristic.
- Platform is the main (and only one in the case of non-cross-platform development) platform of the project, where it was first tested and passed the customer acceptance procedures. During the analysis, the following possible options were identified: Desktop, Mobile, and Web. For cross-platform projects, as well as those where there

was no unambiguous dominance of any option, the Generic option was left.

- Language means the main development language. It is a programming language that was used to develop the main modules and plugins for the project. However, the Generic option is left for the microservice architecture (see below) and for some projects.

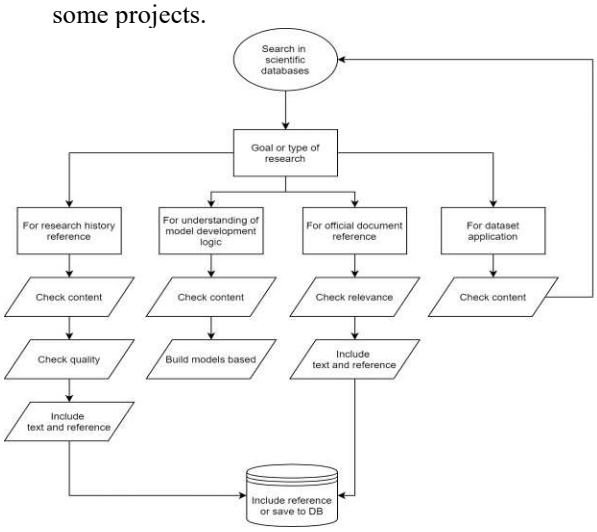

Figure 1. Literature review processing stages

- Team is the composition of the project team. In this work, the following formula of estimating the project team was used:

$$T = \sum_{i=1}^{n} R_i * t_i \quad (1)$$

where n is the number of developers; Ri is the rank of developer, rated on the generally accepted Junior, Middle, and Senior scale.

t_i is the time spent by the developer on the project. It means a mode of operation:

$$t_i = \frac{hours}{8} \quad (2)$$

That is, the ratio of the number of hours that the developer works on the average project per day to 8 (the standard daily norm). For full time developers, this indicator will be equal to 1.

- Size Metric is the project size estimation metric. Since estimation using the Use Case points or Feature points methodologies is associated with a large amount of work and is not always possible, and the sources of all projects were available, the LOC (count of lines of code) metric was chosen for compiling the dataset. Size is the project size in LOC.
- Development Mode is the type of work – a completely new project or reworking/fixing an existing one.
- Code Reuse means an indicator of re-use of code, modules, or libraries. It is estimated from 0% to 100%.
- Architecture is the software architecture of the project. The analyzed projects were mostly based on a multi-layer or micro-service architecture.
- Customer Quality is an indicator of the customer's «quality». Despite the absence of this parameter in most existing works, the frequency and quality of communication with the customer, the number of additional requests and requirements for the project, put forward after the start of development, directly affects the final period of project implementation. This parameter varies from 1 to 3.
- Project Management Quality is an indicator of the experience and suitability of the project manager, his/hers assistants, administrators (if available), and the maturity of project management in the team and company.
- Duration means the actual period of project implementation from initiation to delivery to the customer.

Fig. 2 show the dataset features correlation with project's duration.

d. Model

Based on the collected dataset we have built a deep learning model to predict the duration of the project. Deep neural networks can re-use the features computed in each hidden layer in higher hidden layers. This enables a deep neural network to exploit compositional structure in a function, and to approximate many natural functions with fewer weights and units.

Our models contain 6 layers: the first and fifth layers contain 64 features with SELU [24] and linear activation function [25], the second and fourth layers contain 128 features with RELU activation function, the third layer contains 256 features, and the output layer to predict the duration of the project.

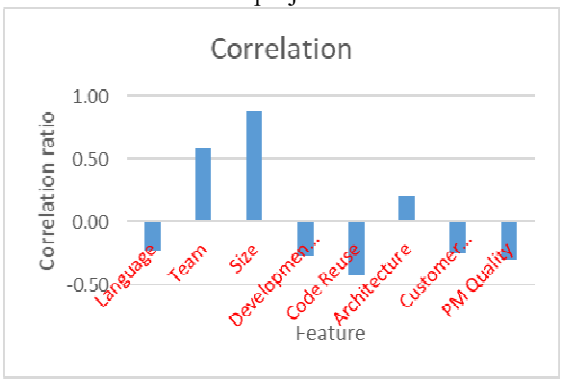

Figure 2. Dataset features correlation with project's duration

The training of the networks was achieved with the Stochastic Gradient Descent algorithm [26] using mini batches of size 16. The data were divided into training (70%) and testing (30%), and the networks were

all regularized using early stopping and gradient norms, and the dropout mechanism and weight decay for some of the networks. Network parameters have all been randomly initialized from a uniform distribution, which is a good initialization technique for deep networks.

e. Experiments

The proposed and tested models have all been implemented using the Pytorch library for Python, which enables the straightforward use of highly optimized mathematical operations on GPUs via Python. PyTorch [27] is a Python package with two high-level features: tensor computing (like NumPy) with heavy GPU acceleration and deep neural networks built on a tape-based auto-grade framework. The tests were performed on a computer with 2x-Intel(R) Xeon(R) E-5-2680 and 4x "NVIDIA Tesla k20x" CPUs.

IV. RESULT

The average error value and standard deviation were recorded as parameters that reflect the quality of the model and the achievement of results.

Our model achieves an average error is 0.38 and a standard deviation is 3.35. The results of model validation on a test sample are presented in Table.1.

TABLE 1 MODEL VALIDATION ON TEST DATASET

Real duration	12	68	12	24	16	32	3	36
Predicted duration	12	67	18	28	16	26	2	37

Project Evaluation Survey is base part for the stage of selecting problems and methods of its resolution. During the survey, 50 respondents were interviewed and answers to 7 questions were collected.

72% of respondents work in information technology, 16% in Finance and Economics, 4% in Healthcare, and 2% in Education. Most respondents participated in projects as Project manager/Assistant (46.7%), Analyst (42.2%), Developer (62.2%), Teamlead (20%), see Fig. 3.

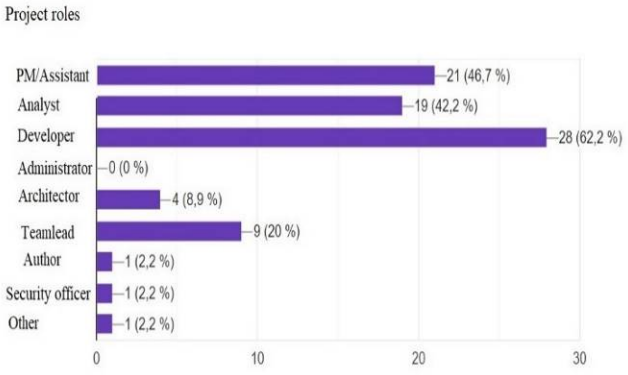

Figure 3. Project roles

According to the respondents' answers, most projects used expert estimation methods by a large margin (82.2%), to a lesser extent – Delphi or algorithmic evaluation shown in Fig 4. Whereas the average accuracy of the preliminary estimation according to the opinions of the respondents was 3.22 (see Fig. 5).

According to the survey's data, the majority (55.6%) of respondents believe that the use of automated tools, artificial intelligence models and machine learning can significantly improve the accuracy of the preliminary estimation of the duration and complexity of the project.

Based on the analysis of the survey results, the following conclusions can be made:

1. The vast majority of IT professionals have participated in project implementation one or more times.

2. During the course of a career, a person performs different roles, sometimes even within the same project (for example, simultaneously the role of team leader and analyst, or project manager and administrator, developer and analyst, etc.)

3. Despite the rapidly developing industry and general trends towards automation, many components of project management still use classical methods. For example, most projects used top-down evaluation or expert evaluation. Proven methods such as the COCOMO algorithm, parametric estimation, and others are rarely used in current projects.

4. The accuracy of classical methods is susceptible to systematic errors.

5. Applying new and improving old methods of preliminary project estimation can significantly improve the accuracy of the assessment. The use of methods based on neural network models and machine learning, according to the results of the survey, can give a good boost in this area.

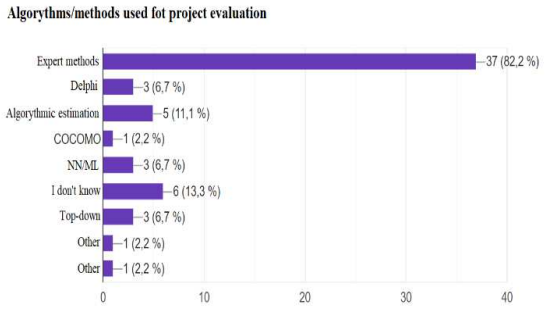

Figure 4. Methods used for project evaluation

V. CONCLUSION

The main research papers in the field of estimating the time and complexity estimation of the project were analyzed in this work. In the process of the analysis, as well as based on the results of the survey, the need to improve estimating methods and to develop new models was identified.

Neural network architecture was chosen as the basis for the model as a promising technology. The developed model, consisting of 6 layers, showed fairly high results on a test sample from the dataset compiled by the author.

Of course, the developed model needs further improvement, training, and validation on large amounts of real data from projects. However, it can be used for parallel estimation of real future projects in experimental mode already now.

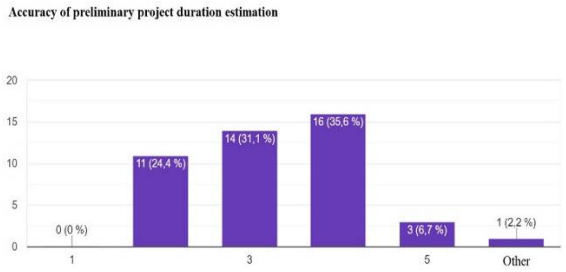

Figure 5. Accuracy of preliminary project estimation

The big advantage of the developed architecture is that input data can be easily collected from completed projects if there is a certain amount of project documentation, logs, and version control system data. This makes it easier to integrate the model into the pipeline for estimating and implementing software development projects in teams and companies.

Two scientific articles that revealed the differences and features of existing datasets, as well as recommendations on the process of collecting data on the project, were written during the research process.

This work makes a large contribution and has high scientific novelty in the area of project management and information technology.

ACKNOWLEDGEMENT

I'd like to express my gratitude to Buravov Alexey who participated in the section of model development using python programming.

REFERENCES

[1] Al-Qutaish, Rafa & Abran, Alain. (2010). Halstead Metrics: Analysis of their Design. 10.1002/9780470606834.ch7.

[2] Albrecht, A.J. and Gaffney, J.E., Jr. (1983) Software Function, Source Lines of Code, and Development Effort Prediction: A Software Engineering, IEEE Transactions on Software Engineering, 9(6):639-648

[3] Nageswaran, Suresh. "Test effort estimation using use case points." Quality week. Vol. 6. 2001.

[4] D. Nurseitov, K. Bostanbekov, M. Kanatov, A. Alimova, A. Abdallah, G. Abdimanap "Classification of Handwritten Names of Cities and Handwritten Text Recognition using Various Deep Learning Models", Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 5, pp. 934-943 (2020).

[5] Abdallah A, Hamada M, Nurseitov D. Attention-Based Fully Gated CNN-BGRU for Russian Handwritten Text. Journal of Imaging. 2020; 6(12):141. https://doi.org/10.3390/jimaging6120141

[6] Abdallah, Abdelrahman, et al. "Automated Question-Answer Medical Model based on Deep Learning Technology." Proceedings of the 6th International Conference on Engineering & MIS 2020. 2020.

[7] Hamada, Mohamed A., et al. "Sentimental text processing tool for Russian language based on machine learning algorithms." Proceedings of the 5th International Conference on Engineering and MIS. 2019.

[8] Hamada, Mohamed Ahmed, and Lyazat Naizabayeva. "Decision Support System with K-Means Clustering Algorithm for Detecting the Optimal Store Location Based on Social Network Events." 2020 IEEE European Technology and Engineering Management Summit (E-TEMS). IEEE, 2020.

[9] Hamada, Mohamed A., Yeleussiz Kanat, and Adejor Egahi Abiche. "Multi-Spectral Image Segmentation Based on the K-means Clustering." Int. J. Innov. Technol. Explor. Eng 9 (2019): 1016-1019.

[10] Nurseitov, Daniyar, et al. "Hkr for handwritten kazakh & russian database." arXiv preprint arXiv:2007.03579 (2020).

[11] Hidmi, Omar, and Betul Erdogdu Sakar. "Software development effort estimation using ensemble machine learning." International Journal of Computing, Communication and Instrumentation Engineering 4.1 (2017): 1-5.

[12] Braga, Petrônio L., Adriano LI Oliveira, and Silvio RL Meira. "Software effort estimation using machine learning techniques with robust confidence intervals." 7th international conference on hybrid intelligent systems (HIS 2007). IEEE, 2007.

[13] Subandri, M. A. and Sarno, R. (2017) "Cyclomatic Complexity for Determining Product Complexity Level in COCOMO II," *Procedia Computer Science*. Elsevier B.V., 124, pp. 478–486. doi: 10.1016/j.procs.2017.12.180.

[14] Boehm, B. et al. (1995) "Cost models for future software life cycle processes: COCOMO 2.0," Annals of Software Engineering, 1(1), pp. 57–94. doi: 10.1007/BF02249046.

[15] Hihn, Jairus; Menzies, T. (2006) NASA93 PROMISE data repository. Available at: http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa_2.arff.

[16] ISBSG (2020) ISBSG Project Data. Available at: https://www.isbsg.org/software-project-data/.

[17] Valerdi, R. (2011) "Convergence of expert opinion via the wideband delphi method: An application in cost estimation models," 21st Annual International Symposium of the International Council on Systems Engineering, INCOSE 2011, 2, pp. 1238–1251.

[18] Boehm, B. (1981) COCOMO 81 Dataset. Available at: http://promise.site.uottawa.ca/SERepository/datasets/cocomo81.arff.

[19] Kuldin, S. P. (2010) "Genetic approach to the problem of estimation the timing and complexity of software development with specified quality requirements."

[20] Gonzalez-Ladron-De-Guevara, F. and Fernández-Diego, M. (2014) "ISBSG variables most frequently used for software effort estimation: A mapping review," International Symposium on Empirical Software Engineering and Measurement, pp. 0–3. doi: 10.1145/2652524.2652550.

[21] Kemerer, C. F. (1987) *Kemerer dataset*. Available at: https://zenodo.org/record/268464.

[22] Maxwell, K. . (2002) *Maxwell dataset*, *2002*. Available at: https://zenodo.org/record/268461.

[23] Albrecht, A. J. and Gaffney, J. E. (1983) *Albrecht dataset*. Available at: https://zenodo.org/record/268467.

[24] Klambauer, Günter, et al. "Self-normalizing neural networks." arXiv preprint arXiv:1706.02515 (2017).

[25] Nwankpa, Chigozie, et al. "Activation functions: Comparison of trends in practice and research for deep learning." arXiv preprint arXiv:1811.03378 (2018).

[26] Bottou, Léon. "Stochastic gradient descent tricks." Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012. 421-436.

[27] Paszke, Adam, et al. "Automatic differentiation in pytorch." (2017). URL:https://pytorch.org

[28] Hamada, Mohamed A., and Abdelrahman Abdallah. "Estimate The Efficiency Of Multiprocessor's Cash Memory Work Algorithms." arXiv preprint arXiv:2102.03848 (2021).

[29] Abdelhalim, Ibrahim Saad Aly, Mamdouh Farouk Mohamed, and Yousef Bassyouni Mahdy. "Data augmentation for skin lesion using self-attention based progressive generative adversarial network." Expert Systems with Applications 165 (2021): 113922.