# Estimating software development effort based on deep learning

**Israa Akram Al-naimy[1,a], Marwa Adeeb Al-jawaherry[1,b]**

[1] **Department of Computer Science, College of Computer Sciences and Mathematics,University of Tikrit, Iraq / Tikrit**

[a] *Corresponding author: israa.a.basheer@tu.edu.iq* , [b] Marwa.A.Aljawaherry@tu.edu.iq

## Abstract:

The aim of this study is to compare two learning models used for estimating software effort. The first model includes a layer, Gated Recurrent Unit (GRU), dropout layer, flatten layer and two dense layers. In contrast the other model is an Artificial Neural Network (ANN). These models will undergo testing, on three datasets. Desharnais, ISBSG10 and Finnish. To assess their performance. The evaluation will consider metrics such as Mean Absolute Error (MAE) Mean Squared Error (MSE) Root Mean Squared Error (RMSE) and R squared ($R^2$). The result reveal that the first model exhibits performance across all datasets with $R^2$ values and high MAE and RSMA values signifying its inefficacy. Similarly the second model demonstrates performance on the Finnish dataset with negative $R^2$ values. From these findings it is apparent that hybrid deep learning models have limitations in software effort estimation, thus warranting research, in this domain.

**Keywords:** Software effort estimation, , hybrid deep learning model, Convolutional Neural Network (CNN) , Recurrent Neural Network (RNN), ANN , GRU

## 1.Introduction

Effort estimation, in software development is crucial for project planning and execution. It involves predicting the time resources and manpower needed for a task. However estimating effort can be challenging due to the variable nature of software projects. It holds significance, for reasons. Firstly it aids project managers in allocating resources establishing deadlines and pinpointing risks during the development phase to effectively plan projects. Secondly it supports in budget planning and cost estimation ensuring organizations allocate resources suitably without surpassing limits. Accurate effort estimation is crucial, for communication with stakeholders. Ensuring customer satisfaction by managing expectations regarding project timelines and deliverables. However achieving effort estimation in software development can be challenging due to uncertainties such as evolving technology and changing requirements that impact the accuracy of estimates. This research seeks to explore how well learning models work., in forecasting software effort estimation .By combining network architectures, such, as neural networks (CNNs) and recurrent neural networks (RNNs) we aim to leverage the distinct strengths of each architecture to enhance the precision of prediction models for estimating software development efforts. Specifically we evaluate two learning models. Assess their performance using three used datasets, in the field. The aim of this research is to create and assess learning models, for estimating software effort. The main emphasis is, on testing their ability to predict accurately and generalize across datasets. We will thoroughly examine performance measures, like Mean Absolute Error (MAE) Mean Squared Error (MSE) Root Mean Squared Error (RMSE) and the R R2) value to understand how well hybrid deep learning methods tackle the complexities of estimating software effort. The paper is structured as follows. Firstly, A review of the existing literature covers methods on software effort estimation techniques and deep learning approaches. This is followed by a description of the methodology used to design and evaluate the deep learning models. The experimental results are then presented and their implications are discussed. Finally, The paper ends by summarizing the discoveries and providing suggestions, for future studies.

## 2.Related Work

Khatibi, E., & Khatibi Bardsiri, V Hybrid methods involve merging models to create a powerful unified model. When compared to algorithms, like ABE, CART, MLR and CNN a combined estimation model with a weighting system surpasses them. Metrics such as MRE, MMRE and PRED(25) are used for evaluation. It was found that computational expenses pose a challenge, for approaches. [1]

Nassif et al. compared the decision tree forest model with the decision tree model and multiple linear regression. They used the ISBSG and Desharnais datasets for this evaluation. The decision tree forest model consists of decision trees growing simultaneously while the decision tree follows a partitioning method. In terms of evaluation metrics, like MMRE, MdMRE and PRED (0.25) the decision tree forest model outperformed both the decision tree model and multiple linear regression.[2]

The study conducted by Mendes and Lokan delves into the topic of replicating research studies using the ISBSG 10 datasets. The results suggest that discrepancies, in estimation accuracy are likely due to variations in data patterns across dataset releases. This discrepancy arises from the fact that using datasets like ISBSG 10 may not truly reflect the software project population despite their convenience. Authors often do not fully disclose the setup, which contributes to findings. This issue is particularly crucial for ISBSG 10 datasets to an amount of missing data. While this article discusses the importance of utilizing company data like ISBGS 10 for predicting software effort studies indicate that models within a company yield more precise forecasts. However companies lacking historical data may rely on publicly available datasets such as ISBGS 10, for estimating their effort levels.[3].

Idri et al. conducted a study comparing the Analogy based software effort estimation (ASEE) with eight techniques, both machine learning (ML) and non ML approaches. These techniques included the COCOMO model, regression analysis, expert judgment, artificial neural networks, function point analysis, support vector regression, decision trees and radial basis function. The evaluation was performed on datasets such, as Desharnais, ISBSG (International Software Benchmarking Standards Group) Albrecht, COCOMO, Kemerer, Maxwell Abran and Telecom. The results showed that the ASEE method demonstrated accuracy compared to the techniques based on three assessment metrics; MMRE, MdMRE and Pred(25). The ASEE approach incorporates a range of methods like logic, genetic algorithms expert judgment and artificial neural networks. When combined with logic and genetic algorithms specifically yielded results in comparison to other combined methods. This integration of ASEE, with machine learning and non learning techniques was found to improve estimation accuracy [4].

Suresh Kumar et al. conducted a study, on estimating software development effort using neural network (ANN) algorithms, such as a basic neural network, a higher order neural network and a deep learning network. The main goal of their research was to compare both qualitative analyses of studies related to software effort estimation. In addition they designed a survey that focused on used datasets for estimation, popular hybrid algorithms for prediction and frequently used assessment metrics, like MMRE, MdMRE and MRE.[5]

A G Priya Varshini et al. compared the deepnet, neuralnet, support vector machine and random forest algorithms. The findings show that random forest outperforms the algorithms because of its resilience and capability to handle datasets. Mean Absolute Error, Root Mean Squared Error, Mean Square Error and R Squared are the evaluation metrics that were talked about.[6].

Akshay Jadhav and Shishir Kumar Shandilya conducted a study comparing the performance of eight machine learning models using used datasets, in Software

Development Effort Estimation. They evaluated the models based on eight metrics. Found that certain machine learning models outperformed datasets across various metrics. This suggests that machine learning improves performance in SDEE. Furthermore after comparing the three models it was concluded that Random Forest demonstrates accuracy and stability, in estimation precision compared to the other models[7].

## 3.Methodology

In this section we detail the approaches employed in this study covering the selection of datasets, preprocessing steps, building models and determining evaluation criteria.

### 3.1 deep learning

A class of ML methods known as deep learning is derived from the information processing processes of biological systems. In order to enhance the extraction of features from the input data, these models implement numerous neurons per layer in multiple layered layers (hence the term "deep"). Currently, the most advanced methodologies in the fields of image recognition and natural language processing are based on deep neural networks (DNNs).[8]

DL technology, which originated from artificial neural networks (ANNs), has become a hot topic in the field of computing due to its impressive data-driven learning capabilities. It is now extensively used in a variety of practical applications. It is a challenging task to construct a suitable deep learning model, as it is characterized by the dynamic nature of real-world problems and the variability of data. Additionally, the absence of a fundamental comprehension renders DL methods into black-box devices that impede standard-level development.[9]

#### 3.1.1 Deep ANN

The term artificial neural network (ANN) is employed. The structure and operation of the human brain are the foundation of this particular machine learning method. An artificial neural network (ANN) is composed of layers of interconnected elements, which are frequently referred to as neurons. A "deep artificial neural network" (deep ANN) is an ANN that contains multiple hidden layers, whereas traditional ANNs typically have one or two hidden levels. As a way to learn more intricate data representations, deep neural networks (ANNs) may contain significantly more hidden layers than standard ANNs[10], figure (1) shows the structure of Deep ANN model

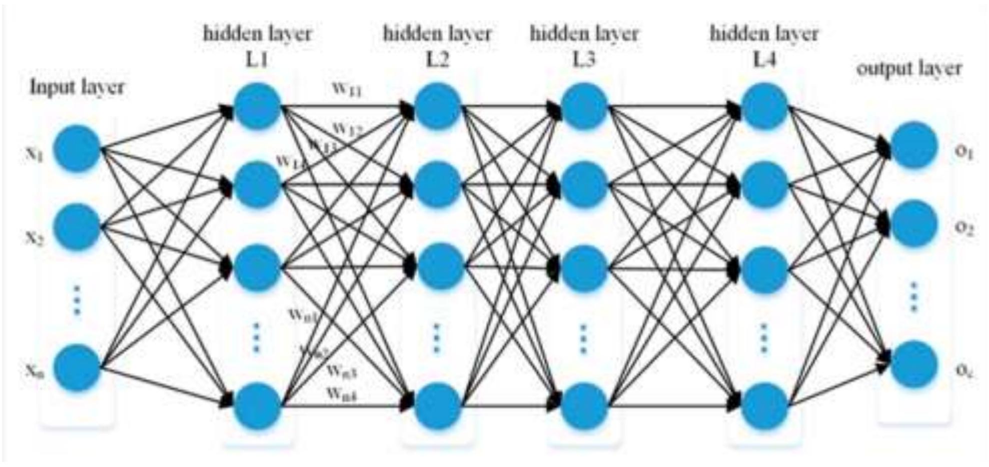


**Figure (1) Structure of Deep ANN model**

### 3.3.2 CNN

In recent years, the convolutional neural network (CNN) has become quite popular due to its ability for recognizing patterns in data. Although CNN is commonly used in image analysis it also proves handy for data making it a good fit for analyzing time related data such, as rainfall and runoff. Convolutional neural networks (CNNs) with 1D convolutions (Conv1D) are adaptable and have a broad range of applications in several kinds of domains[11]. The objective of CNN is to determine and discover correlations and patterns among data items[12] . The fundamental components of CNN are convolutional layers, pooling layers, and fully connected layers. Convolutional layers are accountable for the extraction of features from input data[13] figure (2) illustrates the structure of typical CNN model

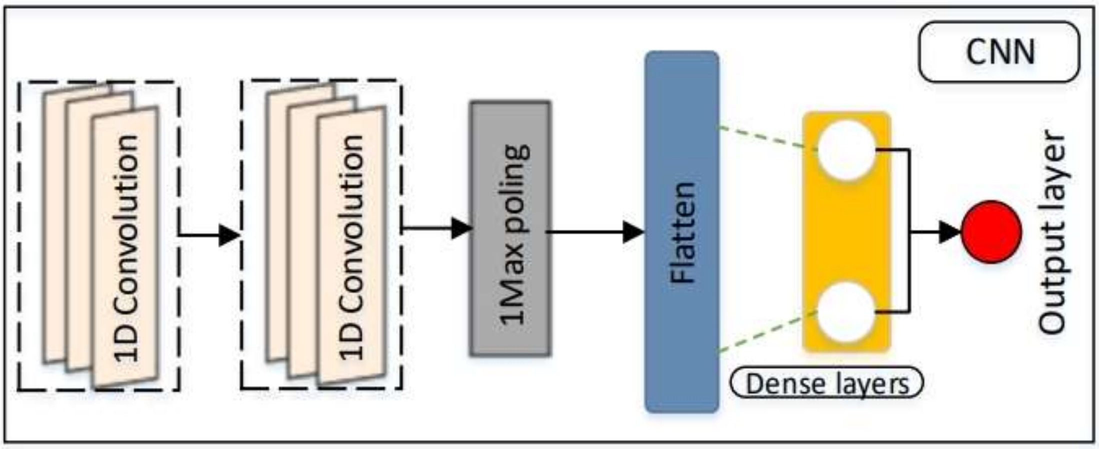


**Figure (2) Structure of basic 1D CNN[14]**

### 3.3.3 RNN

The RNN deep-based model is capable of processing sequential data. Even so, RNN experiences exploding and vanishing gradients issues when long-term dependencies are present in a given input. Consequently, to resolve this matter, a variety of RNN deep learning-based models were introduced, such as GRU networks[15] figure (3) illustrates the structure of typical RNN model

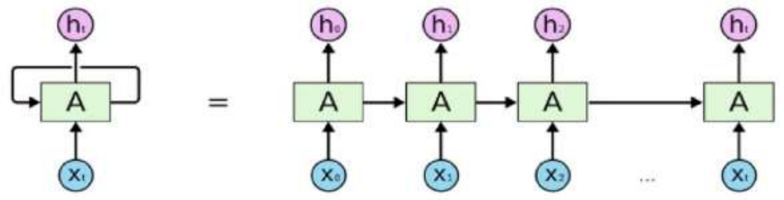


**Figure (3) Structure of typical RNN model [16]**

To resolve the memory issue associated with RNN, the GRU deep model was developed. The update gater determines the quantity of prior information that must be transferred to the current state. In addition, the reset gate Z decides if the prior hidden state should be ignored or passed on to the layers[15] as shown in figure (4)

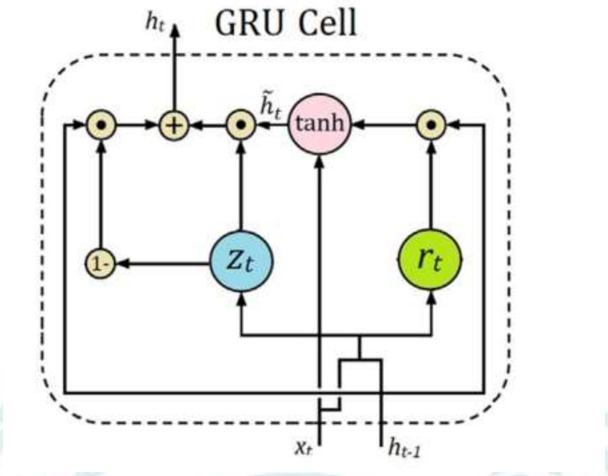


**Figure (4) Structure of basic GRU cell [17]**

In this study, we have developed a focus bidirectional CNN-RNN deep model that is motivated by the previously mentioned discussion.

### 3.3.4 Hybrid CNN-GRU Model

A CNN and GRU hybrid model was proposed in this study to ascertain the software implementation effort of an estimation tool. This approach is implemented due to the advantages of both models, specifically CNNs' capacity to capture spatial features and GRUs' reduced training time. The combination of these models allows for the calculation of a result with minimal processing time, utilizing fewer parameters[11, 18] steps of model shown in figure (5)

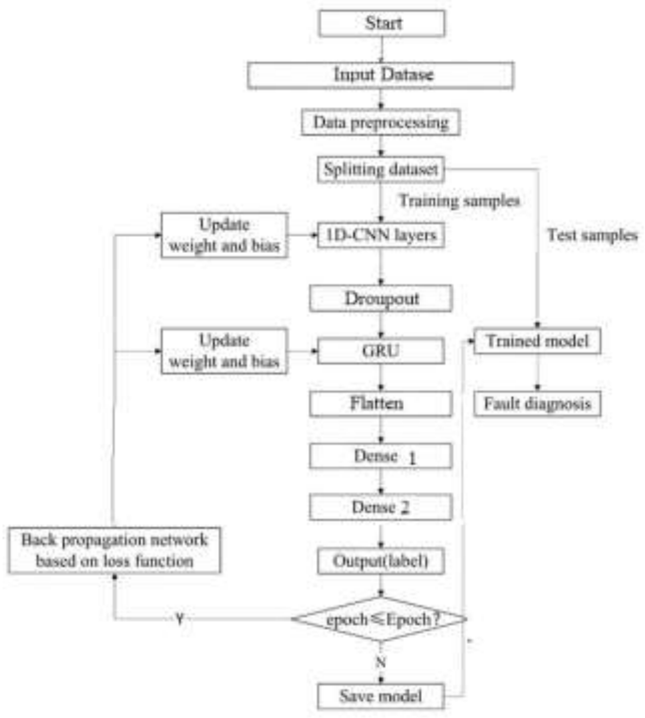


**Figure (5) Flowchart of Proposed model**

In numerous classification and regression tasks, the combination of CNN-RNN has been demonstrated to be successful, as it is capable of capturing both local and sequential aspects of input data. At this point, the input is processed and local features that are located at the text level are extracted using a CNN layer of Conv1D. Input for the RNN layer of RGU units/cells that follows is the output of the CNN layer[19].The structure of proposed model is shown in figure (6)

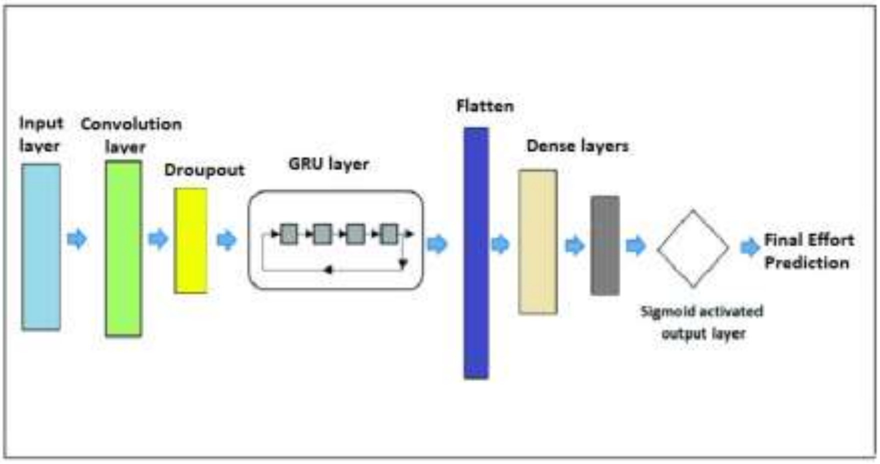


**Figure (6) Structure of Proposed model (CNN-GRU)**

## 3.3 Dataset Selection:

In this study, three datasets selected are distinic commonly utilized in software effort estimation research: Desharnais,ISBSG10,and Finnish. These datasets represent diverse project characteristics, sizes, and contexts, enabling comprehensive evaluation of ensemble models' performance across varied scenarios.

### 3.3.1 Desharnais dataset

The Desharnais dataset was gathered from 81 software projects conducted by software companies, in Canada. It consists of ten characteristics, which involve two factors that rely on each other (time and effort quantified in 'person hours), Eight separate attributes. Regrettably, four projects out of the 81 contained missing values, prompting their removal to prevent potential influence on the estimation process. As a result, the data preprocessing phase resulted in a total of 77 completed software projects, it’s taken from the PROMISE repository[20, 21].

### 3.3.2 ISBSG10 dataset

The International Software Benchmarking Standards Group Version 1 dataset is a small subset of the ISBSG data downloaded from zenodo website ,consist of 37 projects[22].

### 3.3.3 Finnish datasets

The TIEKE group purchased the Finnish dataset from nine Finnish companies. At first, forty documents were collected. 38 records remained after two projects' data were removed from the study (Kitchenham and Kansala 1993) owing to missing values in certain of their characteristics. There are nine features in this dataset, and each one represents the size expressed in function points[20, 21, 23].

### Preprocessing

Before model training, preprocessing steps conducted to prepare the datasets for analysis. This involved data type conversion, handling missing values, and removal of duplicate entries , converting data type to numerical type. Additionally, performed feature selection based on correlation analysis by plotting heat map to identify the most informative features for effort estimation as below for each dataset, In Desharnais dataset, converting the data type and delete duplicated, plotted the dataset correlation heat map, as shown in the figure(7).

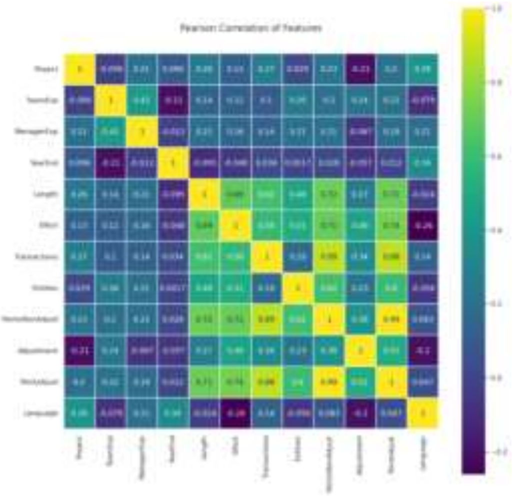
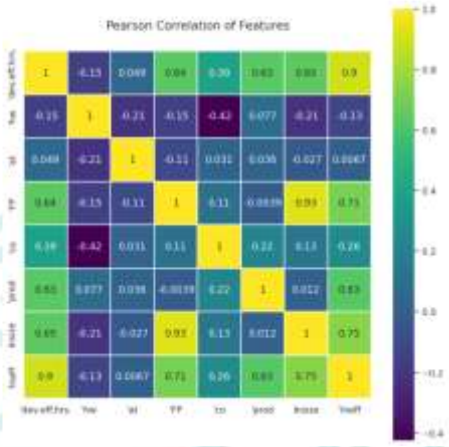


**Figure (7) Desharnais dataset heat map**

According to the heat map the feature “yearend” deleted from the dataset, and used other features as input for the model.at the same way the Finnish and ISBSG10 datasets processed, the feature “prod” deleted from finnish dataset, while in ISBSG10 all features deleted which have less than 10% correlation to the effort from the dataset, and used other features as input for the model, as shown in the figures (8) and (9).



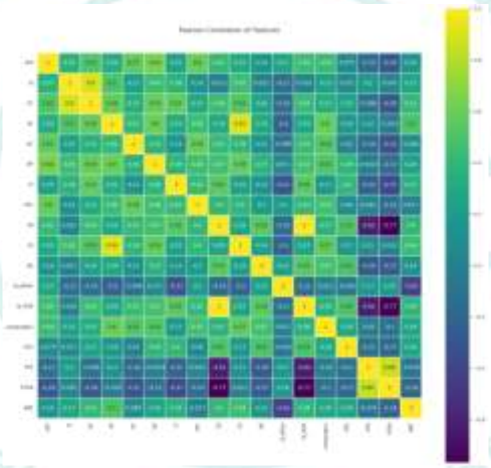


Figure (8) Finnish dataset heat map Figure (9) ISBSG heat map

## 3.5 Model Design

The proposed models are designed and tested using the Google COLAB platform and using Python programming language , where the models are designed and tested on each dataset separately. each dataset divided into 80% as a training set and 20% as a test set. work was done according to two scenarios: In the first scenario, hybrid CNN-GRU model is used, and the second scenario a deep ANN model, Model performance is then evaluated.

## 3.6 Evaluation Criteria

Regression measures that are commonly used to evaluate model performance were Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2). These metrics offer a comprehensive understanding of the models' accuracy, precision, and the fit quality across a variety of datasets. A model that is effective has a lower MAE and RMSE, as well as a higher R2[24].

### 3.6.1 Mean Absolute Error (MAE)

It is determined by aggregating the absolute discrepancies among the predicted and actual values. Smaller MAE values suggest improved predictive model performance or accuracy. It does not demonstrate any bias or an unbalanced distribution. Equations (4) may be employed to calculate it[25] [26]

$$\boldsymbol{MAE_i = \frac{1}{n}\sum_{i=0}^{n-1}|Actual\ \ effort_i - Predicted\ \ effort_i|} \quad \dots \quad (1)$$

### 3.6.2 Mean Squared Error (MSE)

The average squared error between the actual effort and predicted effort is the metric used to evaluate quality. Its value is non-negative, and a value close zero suggests a high level of estimator quality. The MSE for each unique observation (i) over an amount of (n) samples can be calculated using the following formula[27]:

$$.\ \boldsymbol{MAE_i = \frac{1}{n}\sum_{i=0}^{n-1}(Actual\ \ effort_i - Predicted\ \ effort_i)^2} \quad \dots \quad (2)$$

### 3.6.3 Root Mean Squared Error (RMSE)

Provides the sample standard deviation of the variations between predicted effort and actual effort, places a high value on big errors as the errors are squared before they are averaged. Since fewer values indicate greater performance, it is thus most often employed when big errors are desired.The following formula is used to compute it:[11, 28]

$$\textbf{RMSE} = \sqrt{\boldsymbol{\frac{1}{n}\sum_{i=1}^{n}\left(p_i - A_i\right)^2}} \quad \dots \quad (3)$$

### 3.6.4 R-squared ( $\boldsymbol{R^2}$)

A statistical metric called R-squared is used to calculates the percentage of the variation in the dependent variable can be predicted based on the independent variable ,the value of R-squared falls within the range of 0 to 1. If the R-squared value is closer to or equal to 1, the model is better. Conversely, a negative R-value means that the data and the model do not correlate. It is sometimes referred to as the determination coefficient. It is computed in this way: [29, 30]

$$\boldsymbol{R^2 = 1 - \frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{\sum_{i=1}^{n}(\bar{Y} - Y_i)^2}} \quad \dots \quad (4)$$

## 4. Proposed models

In this study, work was done according to two scenarios, the first proposed, we designed a set of hybrid deep learning models (CNN-GRU) and evaluated the performance, and in the second scenario Deep ANN are proposed and evaluated the performance.

## 5. Results and Discussion

This section presents the outcomes of the proposed models for predicting the effort that were developed through deep learning. which is composed of numerous layers. In contrast to prior research, the data is loaded into the algorithm after being divided into two categories: 80% is used for training and 20% is used for testing.

### 5.1 CNN-GRU Model

The CNN model is composed of six layers.The initial layer is a 1D convolutional layer with 23 filters of size 2 kernals and an activation function of "tanh" then follows by a Gated Recurrent Unit (GRU) layer with 64 units. To prevent overfitting, a dropout mechanism with a rate of 0.2 is implemented [31], After the flatten layer, the data will be implemented to two fully connected layers with 64 and 1 hidden unit for additional processing before it is completed by the single unit sigmoid-triggered output layer. The activation function of the fully connected layer (Dense1) is a rectified linear unit ("RELU"), which is linear and learns more quickly than nonlinear functions like tanh and sigmoid. Reducing the vanishing gradient is facilitated by RELU[32] The activation function used in the fully connected layer (Dense2) is "sigmoid." When the fully connected estimates an output, the actual value is compared to the predicted value (in supervised learning, the objective values of the samples are already known), and the error is calculated using a loss function. The loss function for regression issues is typically binary cross-entropy, although it may vary depending on the implementation of deep networks. The loss function is optimized by a variety of methods, including the gradient descent family, stochastic optimizers, and adaptive learning rate methods. In our model, we implemented the Adam optimizer with 1000 epochs as shown in table (1)

**Table (1) Hybrid CNN-GRU Model parameters**

| Layer type | Output shape | Parameters |
|---|---|---|
| **Conv1D_1** | (None, Features number, 32) | 96 |
| **GRU 1** | (None, Features number, 64) | 18816 |
| **Dropout 1** | (None, Features number, 64) | 0 |
| **Flatten 1** | (None, 576) | 0 |
| **Dense1** | (None, 64) | 36928 |
| **Dense 2** | (None, 1) | 65 |
| **Total Parameters** | 55905 (218.38 KB) | |
| **Trainable Parameters** | 55905 | |
| **Non-trainable Parameters** | 0 | |

When we applied the Hybrid model to the datasets we got the results shown in table (2).

**Table (1) Hybrid CNN-GRU Model Results on three datasets**

| Metric\Dataset | DESHARNAIS | Finnish | ISBSG10 |
|---|---|---|---|
| **MAE** | 4496.21 | 7.798 | 2114 |
| **MSE** | 33322844.41 | 61.861 | 12574377.71 |
| **RMSE** | 5772.59 | 7.865 | 3546.03 |
| **R2** | -1.61 | -58.31 | -0.55 |

We note from the table above that the first hybrid deep learning model performs poorly on the three datasets, as negative R² value indicates that the model isn't capturing the variability in the target variable effectively making it unsuitable, for purposes.

## 5.2 Deep ANN Model

The input layer, hidden layers, and output layer constitute the network architecture. The layers are three dense layers and two dropout layers. Every node are connected to other nodes in the subsequent layer. Back propagation is naturally implemented by the Keras API. Table (3) displays the parameters, while Figure 1 illustrates the network layout. The activation function is a rectified linear unit (RELU), and the kernel initializer is normal for regression, as is consistent with previous research. Adam optimization is implemented for the enhancement algorithm. The memory requirements are comparatively low, and it typically functions satisfactorily with minimal hyperparameter tuning. Our loss function, which is binary cross-entropy, is based on MSE in our regression issue. The goal is to determine the architecture with the lowest Mean Squared Error. This is the most frequently employed loss function in regression[33]. The average of the distances, between the predicted and actual values is used to calculate the Mean Squared Error (MSE).

**Table (2) Second Deep Learning Model parameters**

| Layer type | Output shape | Parameters |
|---|---|---|
| **Input layer** | (None, Features number) | 0 |
| **Dense1** | (None, 64) | 1152 |
| **Dropout 1** | (None, 64) | 0 |
| **Dense2** | (None, 32) | 2080 |
| **Dropout 2** | (None, 32) | 0 |
| **Dense3** | (None, 1) | 33 |
| **Total Parameters** | 3265 (12.75 KB) | |
| **Trainable Parameters** | 3265 | |
| **Non-trainable Parameters** | 0 | |

When we applied the first deep learning model to the datasets we got the results shown in table (2).

**Table 2 Second Deep learning Model Results on three datasets**

| Metric\Dataset | DESHARNAIS | Finnish | ISBSG10 |
|---|---|---|---|
| **MAE** | 4501.13 | 551.68 | 2114.84 |
| **MSE** | 32934440.67 | 409556.63 | 12577674.89 |
| **RMSE** | 5738.85 | 639.96 | 3546.5 |
| **R2** | -1.58 | -392729.41 | -0.55 |

We note from the table that the first hybrid deep learning model performs poorly on the three data sets, as the negative value for $R^2$ means that the model is not efficient.

## 6. Conclusion

Deep Artificial Neural Networks (Deep ANNs) and Hybrid Convolutional Neural Network Gated Recurrent Unit (Hybrid CNN GRU) models have shown promising outcomes, in machine learning and deep learning tasks. However these models might encounter challenges in achieving accuracy rates across diverse datasets. One significant drawback of Deep ANN models is their dependence on large annotated datasets for training which leads to increased model complexity and higher storage and computational demands during the training process. Deep artificial neural network models can easily fall into the trap of overfitting especially when working with datasets that contain inconsistencies or noise. Conversely neural network (CNN) gated recurrent unit (GRU) models might face challenges, in capturing extended dependencies and complex temporal patterns within data particularly when the dataset shows significant temporal fluctuations. Furthermore integrating CNN and GRU components can introduce hyperparameters making model optimization an intricate and time consuming task. Researchers may face constraints when applying these models to three datasets.

**Acknowledgments:** The authors would like to thank [Tikrit University/Computer science and Mathematics college] for their support in conducting this research.

## References

1. Khatibi Bardsiri, V., et al., *A PSO-based model to increase the accuracy of software development effort estimation.* 2013. **21**: p. 501-526.

2. Nassif, A.B., et al. *A comparison between decision trees and decision tree forest models for software development effort estimation*. in *2013 Third International Conference on Communications and Information Technology (ICCIT)*. 2013. IEEE.

3. Barb, A.S., et al., *A statistical study of the relevance of lines of code measures in software projects.* 2014. **10**: p. 243-260.

4. Idri, A., et al., *Analogy-based software development effort estimation: A systematic mapping and review.* 2015. **58**: p. 206-230.

5. Suresh Kumar, P. and H. Behera. *Estimating software effort using neural network: An experimental investigation*. in *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2020*. 2020. Springer.

6. Varshini, A.P., et al. *Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation*. in *Journal of Physics: Conference Series*. 2021. IOP Publishing.

7. Jadhav, A. and S.K.J.J.o.E.R. Shandilya, *Reliable machine learning models for estimating effective software development efforts: A comparative analysis.* 2023. **11**(4): p. 362-376.

8. Benning, L., A. Peintner, and L.J.C. Peintner, *Advances in and the applicability of machine learning-based screening and early detection approaches for cancer: A primer.* 2022. **14**(3): p. 623.

9. Sarker, I.H.J.S.C.S., *Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions.* 2021. **2**(6): p. 420.

10. Priya Varshini, A., et al. *Deep Artificial Neural Network for Software Effort Estimation*. in *International Conference on Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication*. 2023. Springer.

11. Shekar, P.R., et al., *A combined deep CNN-RNN network for rainfall-runoff modelling in Bardha Watershed, India.* 2024: p. 100073.

12. Alabsi, B.A., M. Anbar, and S.D.A.J.S. Rihan, *CNN-CNN: Dual Convolutional Neural Network Approach for Feature Selection and Attack Detection on Internet of Things Networks.* 2023. **23**(14): p. 6507.

13. Kumar, A., et al., *Analysis of spectrum sensing using deep learning algorithms: CNNs and RNNs.* 2024. **15**(3): p. 102505.

14. Jafari, S. and Y.-C.J.C. Byun, *A CNN-GRU Approach to the Accurate Prediction of Batteries' Remaining Useful Life from Charging Profiles.* 2023. **12**(11): p. 219.

15. Cho, K., et al., *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* 2014.

16. Sharfuddin, A.A., M.N. Tihami, and M.S. Islam. *A deep recurrent neural network with bilstm model for sentiment classification*. in *2018 International conference on Bangla speech and language processing (ICBSLP)*. 2018. IEEE.

17. Basiri, M.E., et al., *ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis.* 2021. **115**: p. 279-294.

18. Azyus, A.F., S.K. Wijaya, and M.J.S.I. Naved, *Prediction of remaining useful life using the CNN-GRU network: A study on maintenance management.* 2023. **17**: p. 100535.

19. Nasir, J.A., O.S. Khan, and I.J.I.J.o.I.M.D.I. Varlamis, *Fake news detection: A hybrid CNN-RNN based deep learning approach.* 2021. **1**(1): p. 100007.

20. Rahman, M., et al., *Review of Existing Datasets Used for Software Effort Estimation.* 2023. **14**(7).

21. Kocaguneli, E., T.J.J.o.S. Menzies, and Software, *Software effort models should be assessed via leave-one-out validation.* 2013. **86**(7): p. 1879-1890.

22. ISBSG Limited. (2012). isbsg10 [Data set]. Zenodo. https://doi.org/10.5281/zenodo.268485

23. Mittas, N. and L. Angelis. *Comparing software cost prediction models by a visualization tool*. in *2008 34th Euromicro Conference Software Engineering and Advanced Applications*. 2008. IEEE.

24. Marco, R., et al., *Bayesian hyperparameter optimization and Ensemble Learning for Machine Learning Models on software effort estimation.* 2022. **13**(3).

25. Azzeh, M., et al., *An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation.* 2015. **103**: p. 36-52.

26. Nassif, A.B., et al., *Towards an early software estimation using log-linear regression and a multilayer perceptron model.* 2013. **86**(1): p. 144-160.

27. Prabhakar, M.D., M.J.I.J.o.A.R.i.C.S. Dutta, and S. Engineering, *Prediction of software effort using artificial neural network and support vector machine.* 2013. **3**(3): p. 40-46.

28. Marapelli, B.J.I.J.o.I.T. and E. Engineering, *Software development effort duration and cost estimation using linear regression and k-nearest neighbors machine learning algorithms.* 2019. **9**(2): p. 1043-1047.

29. Labidi, T. and Z.J.T.J.o.S. Sakhrawi, *On the value of parameter tuning in stacking ensemble model for software regression test effort estimation.* 2023. **79**(15): p. 17123-17145.

30. AG, P.V., A.K. K, and V.J.E. Varadarajan, *Estimating software development efforts using a random forest-based stacked ensemble approach.* 2021. **10**(10): p. 1195.

31. Hawkins, D.M.J.J.o.c.i. and c. sciences, *The problem of overfitting.* 2004. **44**(1): p. 1-12.

32. Raziani, S. and M.J.N.I. Azimbagirad, *Deep CNN hyperparameter optimization algorithms for sensor-based human activity recognition.* 2022. **2**(3): p. 100078.

33. Al-Mahasneh, A.J., S.G. Anavatti, and M.A.J.a.p.a. Garratt, *Review of applications of generalized regression neural networks in identification and control of dynamic systems.* 2018.