Hindusthan Institute of Technology
(AN AUTONOMOUS INSTITUTION)

ICECMSN

4th International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN 2024)

28-29, November 2024

Proceedings

ELSEVIER

Organised by

Hindusthan Institute of Technology,
Coimbatore, India

Table of Contents

14	Application of LSTM-GRU Combined Model to Calculate the Reliability of Software Systems *Tamilla A Bayramova, Tofig H Kazimov*
15	Deep Learning Approach for Weather Classification using Pre-Trained Convolutional Neural Networks *Harit Tarwani, Shivang Patel, Parth Goel*
16	Logistic Regression for Enhancing Scalability of Blockchain System *Manjula K Pawar, Prakashgoud Patil*
17	Enhancing Mobile Application Security Through Android Threat Classification *Helen Josephine V L, Manikandan Rajagopal, Gayatri S, Lakshmi K*
18	Metaheuristic Feature Selection for Diabetes Prediction with P-G-S Approach *Karuppasamy M, Jansi Rani M, Poorani K*
19	Neural Architecture Search-Driven Optimization of Deep Learning Models for Drug Response Prediction *Uday Kiran G, Srilakshmi V, Padmini G, Sreenidhi G, Venkata Ramana B, Preetham Reddy G J*
20	Integrating NAS for Human Pose Estimation *Srilakshmi V, Uday Kiran G, B Moulika, G S Mahitha, G Laukya, M Ruthick*
21	Greener and Energy-Efficient Data Center for Blockchain-based Cryptocurrency Mining *Asif Mahmuda, K M Safin Kamala, Ahmed Wasif Rezaa*
22	Efficient Miner Selection in Blockchain based on Predicted Transaction Time *Manjula K Pawar, Prakashgoud Patil, Narayan D G, Vasundhara Pandey, Shorya Jain, Priyanshu Kumar*
23	A Novel Privacy Protection Technique of Electronic Health Records using Decentralized Federated Learning with Consortium Blockchain *S P Panimalar, S Gunasundari*
24	Deep Learning-based Vehicle Speed Estimation in Bidirectional Traffic Lanes *Jen Aldwayne B Delmo*
25	Adaptive Deep Reinforcement Learning-based Resource Management for Complex Decision Making in Industry Internet of Things Applications *Niharika Karne, Chandra Shekar Ramagundam, Ranga Rao Patnala, Shashishekhar Ramagundam, Sai Nitisha Tadiboina*
26	An Efficient DDoS Attack Detection in SDN using Multi-Feature Selection and Ensemble Learning *Amit V Kachavimath, Narayan D G*
27	Evaluating Energy Consumption for Routing Selection using the Multi-Routing Clustering Protocol using Timeslot Transmission in Dynamic Path Selection in Wireless Sensor Networks *M Prakash, J Abinesh, P Malarvizhi, J Jeba Emilyn, A Sam Thamburaj, D Vinod Kumar*
28	Genome Motif Discovery in Zika Virus: Computational Techniques and Validation using Greedy Method *Pushpa Susant Mahapatro, Jatinderkumar R Saini, Shraddha Vaidya*

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2025) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Evolutionary Computing and Mobile Sustainable Networks

Application of LSTM-GRU combined model to calculate the reliability of software systems

Tamilla A. Bayramova[a], Tofig H. Kazimov[b]

[a,b]*Institute of Information Technology, B. Vahabzade str., 9A, AZ1141 Baku, Azerbaijan*

Abstract

This study investigated the effectiveness of deep learning models in assessing the reliability of software systems and the application of recurrent neural network algorithms in reliability prediction. A hybrid model consisting of a combination of LSTM and GRU models is proposed to predict the reliability of software systems. Along with historical data collected during testing and implementation, several environmental factors covering the software life cycle and affecting its reliability, as well as the complexity of the software code, are taken as input. Based on these data, a new method for expert assessment of software reliability is proposed, and the calculated expert scores are taken as output. The proposed model is trained based on these values. This is a comprehensive approach to assessing the reliability of software systems.

Peer-review under responsibility of the scientific committee of the 4th International Conference on Evolutionary Computing and Mobile Sustainable Networks

Keywords: software reliability, hybrid model, recurrent neural network, Long Short-Term Memory, gated recurrent unit

1. Introduction

In today's world, software is an integral part of most industries, including medicine, financial services, transportation, and energy. The growing complexity of systems and increasing user expectations require developers and organizations to implement high standards of software reliability. Software reliability is the ability of a system to perform its functions for a certain period of time under specified conditions without failures. It is the result of an integrated approach that covers all stages of development, testing, and operation. Focusing on the development team, documentation, code complexity management, and active work with errors will significantly improve the quality and reliability of software. Software plays a central role in the functioning of modern systems, with software reliability requirements becoming more stringent each year. The process of designing, developing, and testing software requires

Peer-review under responsibility of the scientific committee of the 4th International Conference on Evolutionary Computing and Mobile Sustainable Networks

an emphasis on preventing failures and ensuring stable operation. Modern society is increasingly dependent on computer systems. Critical applications, such as healthcare and transportation management systems, require high reliability, as a failure in such systems can lead to serious consequences, including threats to life and safety. In the era of Industry 4.0, with the increase in data volume and the introduction of new technologies such as cloud computing and the Internet of Things, software is becoming more complex. This complexity requires stricter standards and regulations to ensure reliability. More complex systems often become more vulnerable to errors. Users expect software to be not only functional but also highly reliable. In a highly competitive environment, companies must provide solutions that maintain both functionality and performance. A negative user experience can lead to a loss of customers. Many industries have strict standards and regulations regarding software reliability, which may vary depending on the application area. To meet these standards, companies must increase their efforts to ensure reliability.

To meet the growing requirements for software reliability, it is necessary to implement the following approaches:

1. Systems approach: Development of reliable software requires analysis at all levels: architecture, code, testing and operation.

2. Regular testing: The introduction of multi-level testing (unit tests, integration tests, system tests) allows you to identify and eliminate errors at early stages.

3. Process automation: The use of continuous integration and deployment (CI/CD) helps to ensure high quality and minimize the number of errors that get into the final product.

To achieve reliability of software systems, it is necessary to form a team with highly qualified developers, testers and required specialists, attend regular trainings and seminars to improve their knowledge. The team should consist of specialists from different fields - analysts, designers, test engineers and DevOps specialists. This helps to consider various aspects of reliability at each stage of development. Using collaboration tools such as Agile, Scrum or Kanban and applying automated testing methods based on artificial intelligence contribute to higher productivity and coordination. To minimize the complexity of the code, it is necessary to follow the principles of clean code, architectural patterns and design that promote simplicity. Separating the code into independent modules that can be tested and developed separately significantly reduces the risk of errors. Detailed documentation is needed to support and improve the understanding of the system. These documents include a description of the system architecture, functional requirements, APIs and other critical components, clear instructions on the development, testing and implementation processes, clear and accessible user guides so that they can effectively use the software and solve problems.

The requirements for software reliability are growing exponentially. The success of organizations depends on their ability to create reliable systems that can cope with current and future challenges. Implementing best practices and technologies aimed at improving reliability will be the key to successful operations in the context of rapid technological evolution.In recent years, various models have been developed to predict and manage software reliability. However, the reliability of software systems is mainly estimated based on data obtained during testing, and most reliability models do not take into account other factors.

Software reliability modeling is an important aspect of software development that helps ensure high-quality and stable functioning of applications. Here are some reasons why it is of great importance:

- Predictability: Modeling allows you to predict the behavior of the system under different conditions, which helps identify weak points.
- Cost reduction: By identifying potential problems early, you can reduce the time and money spent on fixing errors at later stages of development.
- Performance optimization: Modeling helps analyze and improve the performance of the system, allowing you to create more efficient algorithms and architecture.
- User satisfaction: High quality and reliability of software directly affects the user experience, which leads to increased trust and loyalty to the product.
- Standards compliance: Many industries require certain reliability standards to be met. Modeling helps ensure that your software meets them.

Overall, reliability modeling is the key to success in modern programming [1].

2. Releated Work

The study [2] employed a deep learning model based on a recurrent neural network (RNN) with an encoder-decoder architecture to predict the number of software errors and assess software reliability. This research compares the deep neural network model utilizing the RNN encoder-decoder with four other neural network models, each using five different parameters. Fourteen crash datasets were selected as the training data for the deep neural network. The experimental results indicate that the proposed model demonstrates the best predictive performance, along with superior stability, reliability, and accuracy in forecasting software reliability.

In [3], time series forecasting methods were effectively utilized for predicting software failures. This paper investigates and compares the performance of software failure prediction using various types of neural networks, specifically the Radial Basis Function (RBF) neural network and recurrent neural networks (GRNN and LSTM). The experiments were conducted using the Chromium browser crash dataset, which contains information on 11,001 error reports collected over 759 days. The analysis of the results indicated that the RBF NN demonstrated good predictive accuracy and learning efficiency for both short and medium time series. However, the simple RBF neural network configuration did not achieve satisfactory accuracy in approximating long software crash time series. Consequently, the authors implemented GRNN and LSTM models because of their ability to retain memory. Both recurrent neural networks effectively predicted long software crash time series. Jindal et al. employed artificial intelligence algorithms, including artificial neural networks (ANN), recurrent neural networks (RNN), gated recurrent units (GRU), and long short-term memory (LSTM) networks to predict software reliability. Based on their experiments, it was found that LSTM produced excellent results in predicting software failure trends, as it effectively captures both long-term and short-term trends within the software failure dataset [4]. Munir et al. proposed a new framework named DP-AGL, which utilizes attention-based GRU-LSTM to predict statement-level defects. By utilizing Clang to construct an abstract syntax tree (AST), the authors defined a set of 32 statement-level metrics. They labeled each statement, created a three-dimensional vector, and applied it as an automated learning model, then used a supervised recurrent unit (GRU) combined with long short-term memory (LSTM). As a benchmark, compared to the state evaluation method, the recall, accuracy, precision, and F1-score of the well-trained DP-AGL under normal conditions improved by 1%, 4%, 5%, and 2%, respectively [5].

Research [6] demonstrates the effectiveness of using the LSTM model, a type of recurrent neural network, for assessing the reliability of software systems. In recent years, hybrid models that integrate recurrent neural networks and their various combinations have gained popularity in forecasting applications [7,8].

This paper proposes:

1. *A combined LSTM-GRU model for software reliability assessment*. A comprehensive approach to reliability assessment is proposed. Historical data on errors and failures, complexity of software code, evaluation of the development team, and evaluation of the documentation level are used as input data.

2. *Expert evaluation of program reliability based on this data*, which will act as output data. The model is trained with this data.

3. Software reliability parameters

To identify weaknesses, detect potential errors and improve quality, it is necessary to conduct code reviews and apply code analysis tools. Using automation tools to create a reliable set of tests allows you to quickly and effectively check the software after changes. Implementing monitoring tools to track the performance and status of the system in real time allows you to detect flaws in the software and work on improving it. For a comprehensive reliability assessment that takes into account external factors, the following approach can be used:

1. Evaluation of the development team: Assess the qualifications and experience of the team members, as well as their knowledge of best practices for developing reliable software.

2. Evaluation of the documentation level: Assess the completeness and relevance of technical documentation, including requirement specifications, architectural diagrams, installation instructions, etc.

3. Evaluation of the number and type of errors detected during testing (unit tests, integration tests, system tests) and their impact on the reliability of the program.

4. Evaluation of the number and type of errors detected during operation: Analyze the causes of errors and the

measures taken to correct them.

5. Evaluation of the complexity of the code: Use appropriate metrics (for example, Halstead metrics, cyclomatic complexity) [9].

These parameters can be assessed by specialists or experts in the field of software reliability based on their experience and knowledge. Considering the professional opinions of experts regarding the current state and prospects for improving software reliability introduces a subjective element to the assessment. Therefore, this paper proposes a hybrid model based on recurrent neural networks.The method of expert assessment of reliability of software systems.

If several experts have assessed the reliability of a software system, the following expert assessment method is proposed for a more reliable assessment:

$$R = \frac{1}{5}\sum_{i=1}^{5} \omega_i f_i , \quad (j = 1,...,5), \tag{1}$$

f_1 -- development team assessment;

f_2 - number of faults during testing;

f_3 - documentation assessment;

f_4 - number of errors during operation;

f_5 - complexity of the program code (with a negative sign);

ω_i - weights of parameters f_i (value coefficient in reliability).

The parameter f_5 can be calculated using the Halstead metric, the cyclomatic complexity metric, and finally estimated by experts in the interval [0, 1].

The parameters $f_1, f_2,..., f_5$ and weights $\omega_i (i = 1,2,...,5)$ will be calculated based on expert assessment, the obtained results will be entered into formula (1) and will allow us to assess the degree of reliability of the software system.

Suppose that $n \in N$ experts have given an assessment of $f_1, f_2,..., f_5$ and $\omega_i (i = 1,2,...,5)$ in an $\lambda \in N$ point system. Then we can calculate the weights ω_i using the following formula:

$$\omega_j = \frac{1}{2n\lambda}\sum_{i=1}^{n} P_{ij} + \frac{1}{2\lambda}\left(\prod_{i=1}^{n} P_{ij}\right)^{\frac{1}{n}}, \quad (j = 1,...,5), \quad \omega_j \in [0,1], \tag{2}$$

Here P_{ij} is the score of the i-th expert on weight ω_j. Note that the parameters $f_2,..., f_4$ can be calculated as follows:

$$f_l = \frac{1}{nk\lambda}\sum_{i=1}^{k} \gamma_{jl} a_{lij}, \quad (l = 2,3,4), \ \gamma_{jl} \in [0,1] \tag{3}$$

Here a_{lij} is the value given by the *i*-th expert to the *j*-th factor of the parameter f_l in the λ -point system.

γ_{jl} - is the weight of the *j*-th factor of parameter f_l,

k - is the number of factors in parameter f_l.

The weights γ_{jl} can be calculated based on the formula (2).

It is clear that in the proposed approach the number of factors influencing software systems can be significantly increased. There is an opportunity to increase the number of experts and change the evaluation system accordingly.

4. Reliability assessment of software systems using recurrent neural network model

To achieve high accuracy, it is necessary to determine which metrics (for example, probability of failure, mean time to fix, mean time between failures) are best suited to assess the reliability of a given software, and establish clear criteria for measurement. Reliability modeling is one of the main areas of reliability management. Statistical and probabilistic models are used to assess and predict the reliability of software systems. Reliability models help to continuously improve the system and adapt to changes. It is necessary to conduct stress and load tests to determine

how resilient the system is to errors and failures, train the team in reliability analysis methods and tools so that they can apply these skills in practice. Focusing on the use of reliable metrics and adapting models based on real data will help improve the reliability and accuracy of software.

The use of recurrent neural networks (RNN) for software reliability prediction is an innovative approach, a powerful deep learning tool that allows modeling complex temporal dependencies between various software quality and reliability metrics. The selection of appropriate metrics and correct data preparation are key factors for the successful implementation of this approach in real-world settings [10]. RNNs are able to efficiently process sequential data, taking into account the relationships between past and future events. Long-term memory mechanisms such as LSTM and GRU allow RNNs to capture long-term dependencies in data, which is important for reliability prediction. RNNs can adapt to changing conditions and identify new patterns in data (fig.1) [11].
Choice of RNN architecture:

- LSTM (Long Short-Term Memory): An architecture that solves the vanishing gradient problem, having special cells and control mechanisms for remembering and forgetting information.
- GRU (Gated Recurrent Unit): A simplified version of LSTM with fewer parameters, but similar functionality. A simpler and more efficient alternative to LSTM.
- Other architectures: Depending on the specific task, other RNN architectures can be used.

Assessing software reliability is a complex task that requires analyzing various factors including crash history, code changes, system load, etc. Recurrent neural networks such as LSTM and GRU are particularly well suited for analyzing time series and discovering hidden dependencies in data.

Making different predictions using a combination of LSTM and GRU models is one of the new and actively researched areas of deep learning and has the following advantages:

- Combined strengths: LSTM is good at remembering long-term data dependencies thanks to its memory cells. GRU has a simpler architecture and can be trained quickly. By combining them, you can create a model that can handle both long-term and short-term dependencies.
- More flexible architecture: The combined model can better adapt to different types of data and tasks, since it allows you to choose the most appropriate type of computation (LSTM or GRU) for each layer and even each neuron.
- Improved accuracy: In most cases, the accuracy of such models is higher than that of LSTM and GRU models taken separately. This is due to the fact that the model can better model complex non-linear dependencies.

LSTM and GRU models can be combined in different ways. In the most common case, LSTM and GRU layers are combined alternately in a network. The use of a hybrid model is effective when the data has both long-term and short-term dependencies, the data is in different formats (e.g. text and images), and when the amount of data is limited, a combined model can help avoid overfitting.

Combining LSTM (Long Short-Term Memory) and GRU (Lightweight Memory Unit) can offer several benefits in software reliability assessment. Here are some of them:

- Better data representation - LSTM and GRU can capture long-term and short-term dependencies in the data. This allows for a better understanding of patterns and dependencies, which is critical for reliability analysis.
- Speed and efficiency - GRUs are less computationally intensive than LSTMs, which can speed up model training. This is important when working with large amounts of data [12];
- Balanced - By using both architectures, it is possible to create a hybrid model that captures the benefits of both LSTMs and GRUs. This can result in higher prediction accuracy [13];
- Adaptive learning - Combining these models allows for an adaptive approach to different types of data, which can improve the overall reliability of the estimates [14];
- Reduced overfitting - The interaction between LSTMs and GRUs can reduce the risk of overfitting, as each model can compensate for the weaknesses of the other [15];
- Improved time handling - Both types of neural networks are excellent for time series, making them ideal for analyzing changes in software reliability over time [16].

This combination can lead to more powerful and robust models that can identify problems and predict potential failures in software systems.

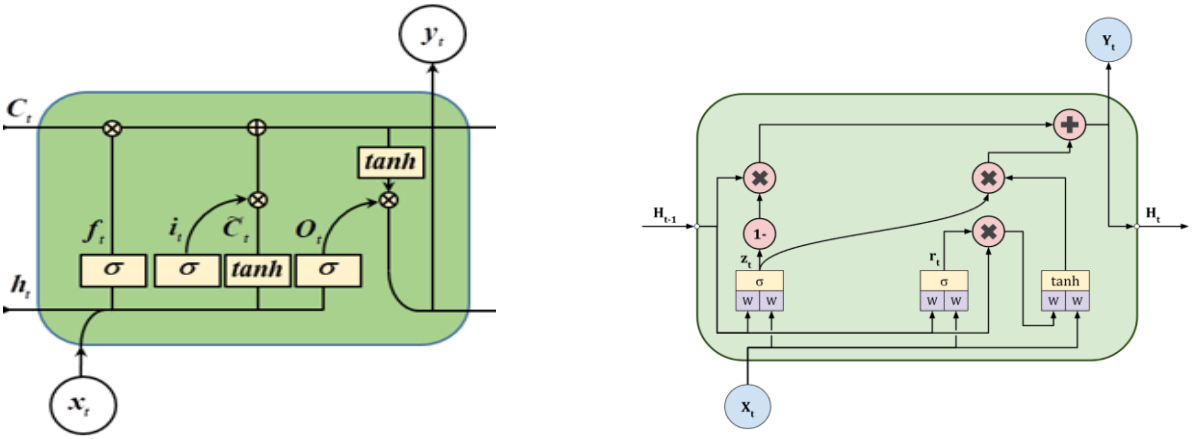

Fig. 1. LSTM & GRU models

4.1. ***Proposed model***

Problem Statement

The problem is to create a machine learning model that, based on five input variables characterizing software, predicts an expert assessment of its reliability. Each of the five input values corresponds to:

- Number of errors during testing,
- Number of errors during operation,
- Program complexity,
- Documentation assessment,
- Development team assessment.

Output data: One output for training - software reliability expert assessment data.

Model Architecture

- Input Layer: Five neurons corresponding to five input variables.
- Hidden Layer: 3 layers of LSTM and GRU cells.
- Output Layer: One neuron that outputs the expert reliability assessment prediction.

Additional Components:

- Embeddings: Some input variables are categorical (e.g., rating documentation on a scale of 1 to 10), embeddings can be used to transform them into dense vector representations.
- Dropout: Dropout can be applied on hidden layers to prevent overfitting.
- Batch Normalization: Can speed up training and stabilize the optimization process.
- Attention: If it is necessary to highlight the most important input features, an attention mechanism can be used.

To prevent overfitting, it is necessary to reduce the complexity of the model, the number of layers and neurons in the model. A simpler model with fewer parameters is less prone to overfitting. You can monitor the validation loss and stop training when the validation error starts to increase (fig 3.). This helps to avoid overfitting, when the model starts to memorize data noise. Reducing overfitting in LSTM and GRU models requires the use of various methods and techniques, as well as experimentation with architecture and hyperparameters. Regularization, early termination, data augmentation and reducing model complexity are all important steps to create more robust and generalizable models. GRU solves the problem of gradient vanishing and exploding in classical recurrent neural networks (RNNs) when learning long-term dependence.

In the learning process, it is important to choose the right hyperparameters of the model. The experiments were conducted in the Python environment.

Table 1. Вычисление весов для каждого параметра

	Название	Эксперт 1	Эксперт2	Эксперт3	ω
1.	f_1	0.1	0.2	0.1	0,1
2.	f_2	0.4	0.3	0.2	0,3
3.	f_3	0.1	0.1	0.1	0,1
4.	f_4	0.3	0.2	0.3	0,3
5.	f_5	0.1	0.2	0.3	0,2

Таблица 2. Экспертная оценка надежности по 5 параметрам.

	P1	P2	P3	P4	P5	P6
f_1	5	8	6	8	9	3
f_2	5	4	8	9	9	4
f_3	6	7	5	9	9	2
f_4	6	4	7	8	9	3
f_5	-5	-6	-8	-9	-8	-4
R	0.67	0.57	0.79	1.00	0.94	0.36

Table 1 shows the weight estimates for 5 parameters by 3 independent experts, and the weights ω for each

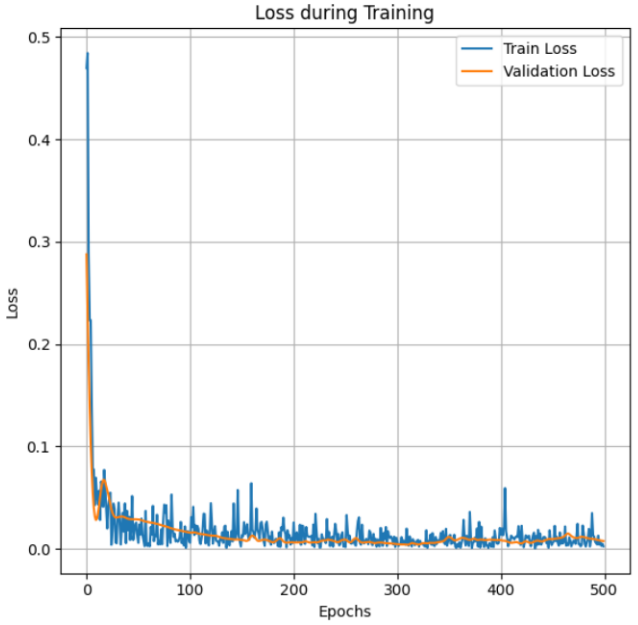

Fig 3. Loss during training

parameter were calculated using formula (2). Table 2 evaluates 6 applications by 5 parameters on a 10-point scale. The expert reliability estimate R for each application was calculated using formula (1). The input values for training the model are f_1, f_2, f_3, f_4, f_5. The output is the expert reliability estimate – R. 2 metrics were used to evalute the performance of the model: Coefficient of determination - R^2, Mean absolute error – MAE (table 3).

Table 3. Performance of model

	LSTM	GRU	LSTM-GRU
MAE	0.0532	0.0631	0.0312
R^2	0.895	0.89	0.91

Fig. 4 shows the actual and predicted values. It can be seen from the graph that the hybrid model has good predictive ability.

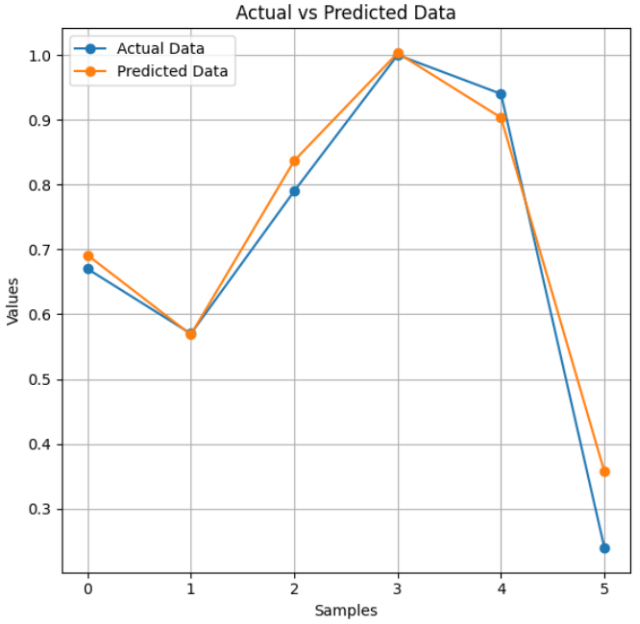

Fig 4. Actual and predicted data

5. Conclusion

Combining LSTM and GRU is a promising approach for creating more powerful and flexible recurrent neural networks. The choice of a specific architecture and hyperparameters depends on the specific characteristics of the data and the requirements for the model. The choice of a specific combination method and the evaluation of its effectiveness require a thorough experimental study.

Combining LSTM and GRU is a powerful tool for assessing software reliability. This technology allows you to build more accurate and reliable models that can help prevent failures and improve the quality of the software product.

The model can help determine which changes in the code or system configuration have the greatest impact on reliability, can be used to optimize resource allocation and prevent overloads that can lead to failures. Accurate reliability assessment allows you to increase trust in the software and make more informed decisions about its use.

- **References**

[1] Vishal Pradhan, Ajay Kumar, Joydip Dhar, (2023), “Chapter 7 - Emerging trends and future directions in software reliability growth modeling”, *In Advances in Reliability Science, Engineering Reliability and Risk Assessment*, Elsevier, Pages 131-144, https://doi.org/10.1016/B978-0-323-91943-2.00011-3.

[2] Wang, J., & Zhang, C. (2018). “Software reliability prediction using a deep learning model based on the RNN encoder–decoder”. *Reliability Engineering & System Safety*, 170, 73-82

[3] Yakovyna, V., & Shakhovska, N. (2022). “Software failure time series prediction with RBF, GRNN, and LSTM neural networks*”. Procedia Computer Science, 207*, 837-847.)

[4] Jindal, A., & Gupta, A. (2022). “Comparative Analysis of Software Reliability Prediction Using Machine Learning and Deep Learning”. *In Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, pp. 389-394.

[5] Munir, H. S., Ren, S., Mustafa, M., Siddique, C. N., & Qayyum, S. (2021). “Attention based GRU-LSTM for software defect prediction”. Plos one, 16(3), e0247444.

[6] Bayramova, Tamilla. (2024), “Predicting the reliability of software systems using recurrent neural networks: LSTM model”. *Problems of Information Technology*, vol. 15, no. 1, 52-61.

[7] Oveisi, S., Moeini, A., & Mirzaei, S. (2021). “LSTM Encoder-Decoder Dropout Model in Software Reliability Prediction”. *International Journal of Reliability, Risk and Safety: Theory and Application*, 4(2), 1-12.

[8] Gür, Y. E. (2024). “Comparative Analysis of Deep Learning Models for Silver Price Prediction: CNN, LSTM, GRU and Hybrid Approach”. *Akdeniz İİBF Dergisi*, 24(1), 1-13. https://doi.org/10.25294/auiibfd.1404173.

[9] Bayramova, Tamilla. (2023). “Development of a Method for Software Reliability Assessment using Neural Networks”. *Procedia Computer Science*, 230, 445-454.

[10] Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). “Recurrent neural networks for time series forecasting: Current status and future directions”. *International Journal of Forecasting*, 37(1), 388-427. https://doi.org/10.1016/j.ijforecast.2020.06.008

[11] Agarap, A. F. M. (2018, February). “A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data”. *In Proceedings of the 2018 10th international conference on machine learning and computing*, pp. 26-30.

[12] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). “A review of recurrent neural networks: LSTM cells and network architectures”. *Neural computation*, 31(7), 1235-1270..

[13] Gao, S., Huang, Y., Zhang, S., Han, J., Wang, G., Zhang, M., & Lin, Q. (2020). “Short-term runoff prediction with GRU and LSTM networks without requiring time step optimization during sample generation”. *Journal of Hydrology*, 589, 125188.

[14] Ghani Rizky Naufal, Antoni Wibowo, "Time Series Forecasting Based on Deep Learning CNN-LSTM-GRU Model on Stock Prices," *International Journal of Engineering Trends and Technology*, vol. 71, no. 6, pp. 126-133, 2023. Crossref, https://doi.org/10.14445/22315381/IJETT-V71I6P215

[15] Dar, Y., Muthukumar, V., & Baraniuk, R. G. (2021). “A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning”. *arXiv preprint arXiv*:2109.02355.

[16] Schmidt, R. M. (2019). “Recurrent neural networks (rnns): A gentle introduction and overview”. arXiv preprint arXiv:1912.05911.