A Transfer Learning-Driven Framework for Enhanced Software Development Effort Estimation Using Optimized Hybrid Deep Learning Model

Badana Mahesh[1,2,*], Mandava Kranthi Kiran[3]

[1]PhD Scholar, GITAM Deemed to be University, Vishakhapatnam, India

[2]Assistant Professor, Department of Computer Science and Engineering, ANITs, Vishakhapatnam, India

[3]Assistant Professor, Department of Computer Science and Engineering, GITAM deemed to be University, Vishakhapatnam, India

Emails: mahesh.cse@anits.edu.in; kmandava@gitam.edu

Abstract

Precise assessment of software development effort (SDE) is essential for efficient project planning and resource distribution. Conventional methods frequently encounter difficulties in generalizing across different project areas because of disparate data attributes. This research presents an innovative approach that combines transfer learning with hybrid deep learning models to tackle these difficulties. The platform utilizes pre-trained Random Forest and LSTM models, enhanced using Jaya optimization, to improve prediction accuracy and adapt effectively to new datasets. Transfer learning is utilized to extract reusable patterns and features from source domains, facilitating effortless adaption to target domains with minimum retraining. Extensive experiments on various benchmark datasets illustrate the proposed framework's enhanced performance regarding accuracy, scalability, and robustness relative to leading techniques. This study emphasizes the capability of transfer learning to transform SDE estimates, providing a scalable and domain-adaptive approach for intricate software projects.

Keywords: Software Development Effort Estimation; Hybrid Methodology; Jaya Optimization; Random Forest-LSTM; Transfer Learning

1. Introduction

The precise estimation of software development effort (SDE) is fundamental to efficient project management in software engineering. It directly influences resource distribution, timeframe estimations, and overall project efficacy. Conventional approaches, encompassing algorithmic models and machine learning techniques, have been thoroughly investigated for SDE estimation [1]. Although these algorithms attain satisfactory accuracy, their ability to generalize across varied project datasets and contexts is constrained. Disparities in software attributes, domain-specific characteristics, and project difficulties frequently result in inadequate performance when utilizing a singular model across various domains. Recent breakthroughs in deep learning and optimization methods have created new opportunities for tackling the issues related to SDE estimation. Hybrid models, including the integration of Random Forests and Long Short-Term Memory (LSTM) networks [2], have demonstrated efficacy in identifying both linear and nonlinear patterns in software project data. Furthermore, optimization algorithms like Jaya optimization improve model performance by refining hyperparameters and feature selection methods. Nevertheless, these strategies necessitate considerable retraining when utilized with novel or encountered datasets, constraining their scalability and practicality in real-world contexts [3]. Transfer learning has arisen as a potent model for tackling these difficulties by facilitating the reuse of knowledge across other domains [4]. In SDE estimation, transfer learning utilizes pre-trained models from source datasets to initialize or refine models for target datasets, thus minimizing the necessity for extensive retraining. This method promotes model adaptability and expedites the estimation process for new projects. Notwithstanding its promise, the utilization of transfer learning

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

in stochastic differential equation estimation is yet inadequately investigated, especially when combined with sophisticated optimization and hybrid modelling methodologies. This paper presents an innovative transfer learning-based approach for SDE estimation that combines optimal hybrid Random Forest-LSTM models with domain adaption features [5]. The proposed approach employs Jaya optimization to enhance the parameters of the hybrid model and utilizes transfer learning to adapt the model to various datasets effectively. Extensive tests are performed on standard SDE datasets to assess the framework's efficacy for accuracy, scalability, and resilience.

The subsequent sections of this work are structured as follows: Section 2 addresses pertinent research in stochastic differential equation estimation, hybrid modelling, and transfer learning. Section 3 delineates the proposed framework, encompassing its design and optimization methodologies. Section 4 delineates the experimental configuration, datasets, and assessment measures. Section 5 presents the findings and contrasts them with leading methodologies, while Section 6 finishes the report by outlining prospective research avenues.

2. Related Work

The estimation of software development effort (SDE) has been extensively studied in software engineering because of its importance in project planning and management. This section offers a summary of current methodologies, emphasizing conventional estimate models, hybrid deep learning approaches, optimization strategies, and the nascent use of transfer learning in SDE estimation.

2.1 Traditional Approaches to SDE Estimation

Traditional SDE estimate techniques, including computational models (e.g., COCOMO, Function Point Analysis), have been widely employed in the industry. These methods depend on established formulas and parameters obtained from previous data [6]. Nonetheless, their precision is frequently constrained by their incapacity to manage the nonlinear and dynamic characteristics of contemporary software projects. Subsequent advancements included the introduction of machine learning models such as Support Vector Machines, Decision Trees, and Artificial Neural Networks to address these constraints. Although these models enhance accuracy, they frequently necessitate substantial retraining when utilized with new datasets, constraining their scalability [7].

2.2 Hybrid Deep Learning Models

Recent improvements in hybrid modelling techniques have demonstrated considerable potential for enhancing SDE estimates. Hybrid models, such as the integration of Random Forest (RF) and Long Short-Term Memory (LSTM) networks, leverage the advantages of their respective components [8]. Random Forest is proficient in managing feature interactions and mitigating overfitting, but LSTM is adept at capturing sequential dependencies in time-series data. Research has shown that hybrid models exceed standalone methods in accuracy and resilience, especially when managing intricate software project datasets. Nevertheless, these models frequently exhibit an inability to adjust to unfamiliar contexts without substantial retraining [9].

2.3 Optimization Techniques in SDE Estimation

Optimization approaches are essential for increasing the performance of SDE estimate models by refining hyperparameters and enhancing feature selection. Evolutionary techniques, including Genetic Algorithm (GA) [10], Particle Swarm Optimization (PSO)[11], and Jaya optimization [12], have been extensively utilized for this objective. Jaya optimization has garnered interest for its simplicity, lack of parameters, and effectiveness in converging to global optima. Its utilization in hybrid models has shown enhanced predictive accuracy and less computational burden. Notwithstanding these gains, optimization-based approaches continue to encounter difficulties in generalizing across varied datasets.

2.4 Transfer Learning in Software Effort Estimation

Transfer learning is a novel paradigm that facilitates the application of information from one domain or dataset (source) to another (target). It has been effectively utilized in several domains, including natural language processing and picture recognition, to tackle the issues of constrained data and computational resources [13]. In SDE estimation, transfer learning can be utilized to initialize or refine models with pre-trained knowledge from analogous datasets, minimizing lengthy retraining and improving scalability. Although research on transfer learning in stochastic differential equation estimation is scarce, its amalgamation with hybrid models and optimization methods offers a promising avenue for investigation [14].

2.5 Research Gap and Motivation

Notwithstanding the progress in hybrid modelling and optimization methodologies, current strategies for SDE estimates frequently encounter challenges related to scalability and adaptability across various domains. The

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

capacity of transfer learning to tackle these difficulties remains predominantly unexamined. The integration of transfer learning with optimized hybrid models, like Random Forest-LSTM, can yield a robust and scalable approach for stochastic differential equation estimation. This research seeks to address this gap by introducing an innovative transfer learning-based framework that utilizes the advantages of optimization methods and hybrid modelling for improved SDE prediction.

3. Proposed Framework

This section introduces the suggested framework for software development effort (SDE) estimation, which is powered by transfer learning and incorporates optimal hybrid deep learning models. The platform integrates Jaya optimization, a hybrid Random Forest-LSTM model, and transfer learning methodologies to tackle issues of accuracy, scalability, and adaptation across various project datasets.

3.1 Framework Overview

The suggested framework comprises three key components:

1. Jaya Optimization for Parameter Tuning: Employed to enhance the hyperparameters of the Random Forest and LSTM models, hence providing superior predictive performance.

2. Hybrid Random Forest-LSTM Model: Integrates the advantages of Random Forest for feature selection with LSTM for sequential data modelling to address intricate, nonlinear patterns in SDE data.

3. Transfer Learning: Enables the transfer of knowledge from pre-trained models to novel datasets, minimizing retraining requirements and improving flexibility.

The framework operates in two phases: training on source datasets and fine-tuning on target datasets using transfer learning.

3.2 Data Preprocessing

Data preparation is an essential phase in the framework to guarantee the quality and uniformity of input data [15]. The subsequent stages are implemented:

- Data Cleaning: Elimination of absent, superfluous, or incongruous entries.
- Feature Normalization: Adjusting features to a consistent range to enhance model convergence.
- Feature Engineering: Extraction of pertinent features relevant to the domain, including code complexity, team size, and project duration.
- Feature Selection: Utilization of Random Forest to rank and identify the most pertinent features for SDE calculation.

3.3 Jaya Optimization for Hybrid Model Tuning

Jaya optimization enhances the hyperparameters of the hybrid Random Forest-LSTM model. The optimization approach entails [16]:

1. Initialization: Establish the search space for hyperparameters, encompassing the number of trees in Random Forest, the number of LSTM layers, and learning rates.

2. Objective Function: Minimize the mean absolute error (MAE) of stochastic differential equation (SDE) predictions during the training phase.

3. Iteration: Adjust the hyperparameters iteratively to achieve convergence to the global optimum.

3.4 Hybrid Random Forest-LSTM Model

The hybrid model integrates the predictive efficacy of Random Forest with the sequential modelling proficiency of LSTM.

- Random Forest: Employed for ranking feature relevance and making preliminary forecasts.
- LSTM: Analyzes sequential project data to identify temporal connections and enhance forecasts.
- The outputs of Random Forest function as supplementary inputs to the LSTM network, establishing a synergistic prediction process.

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

3.5 Transfer Learning for Domain Adaptation

Transfer learning [17] is utilized to modify the pre-trained hybrid model for fresh datasets. This entails:

- Pre-Training: Educate the hybrid model using source datasets that exhibit varied project features.
- Fine-tuning: Modify model weights and hyperparameters on specific datasets while utilizing pre-trained knowledge.
- Domain Adaptation: Employ transfer component analysis (TCA) to synchronize feature distributions between source and target domains, hence providing reliable predictions.

3.6 Workflow of the Framework

The sequence of operations for the proposed framework is outlined as follows:

1. Prepare the source dataset and conduct feature selection utilizing Random Forest.
2. Implement the training of the hybrid Random Forest-LSTM model utilizing Jaya-optimized hyperparameters on the source dataset.
3. Implement transfer learning to modify the pre-trained model for the specific dataset.
4. Optimize the model by utilizing data specific to the target in order to improve prediction precision.
5. Assess the effectiveness of the model by utilizing metrics like MAE, RMSE, and R^2.

3.7 Advantages of the Framework

The suggested framework presents the subsequent advantages:

- Enhanced Precision: The combination of Jaya optimization and hybrid modelling guarantees accurate effort estimation.
- Scalability: The implementation of transfer learning allows for smooth adaptation of the framework across various domains.
- Efficiency: Minimizes the necessity for comprehensive retraining on new datasets.
- Robustness: Integrates feature selection with sequence modelling to ensure dependable predictions.

This framework overcomes the shortcomings of current approaches, offering a strong, scalable, and effective solution for SDE estimation.

Algorithm: Hybrid SDE Estimation Using Jaya Optimization, Random Forest-LSTM, and Transfer Learning

Input:
ISBSG dataset

Output:
Predicted Software Development Effort (SDE)

Steps

```
1.  Data Preparation
    1.1  Import the ISBSG dataset.
    1.2  Address absent values by mean or mode imputation.
    1.3  Standardize numerical features to achieve consistent scaling.
    1.4  Encode categorical variables via one-hot or label encoding techniques.
    1.5  Divide the dataset into training and testing subsets.
2.  Branch 1: Jaya Optimization for Effort Estimation
    2.1  Initialize the settings for the Jaya optimization methodology.
    2.2  Formulate an objective function centered on error minimization (e.g., Mean Squared Error).
    2.3  2.3 Employ Jaya optimization to calibrate the parameters of a foundational regression model (e.g., Linear
         Regression).
    2.4  2.4 Acquire effort estimations.
3.  Branch 2: Hybrid Approach of Random Forest and LSTM
    3.1  Train a Random Forest model on the preprocessed dataset to ascertain feature importance.
    3.2  Identify the most critical features and organize sequential input for LSTM.
    3.3  Train an LSTM model utilizing the chosen characteristics to identify temporal trends.
    3.4  Produce effort estimations utilizing the LSTM model.
4.  Branch 3: Effort Estimation Utilizing Transfer Learning
```

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

4.1 Refine a pre-trained neural network Apply the acquired representations to the ISBSG dataset.
4.2 Train the refined model using the training dataset.
4.3 Acquire effort estimations from the transfer-learning model.

5. Decision Fusion
   5.1 Integrate the forecasts from the three branches with an ensemble method (e.g., weighted average or majority voting).
   5.2 Optimize the ensemble weights according to the performance on the validation set.
6. Assessment of the Model
   6.1 Assess the hybrid model on the testing dataset with performance indicators such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Magnitude of Relative Error (MMRE).
7. Result
   7.1 Provide the ultimate anticipated effort values.

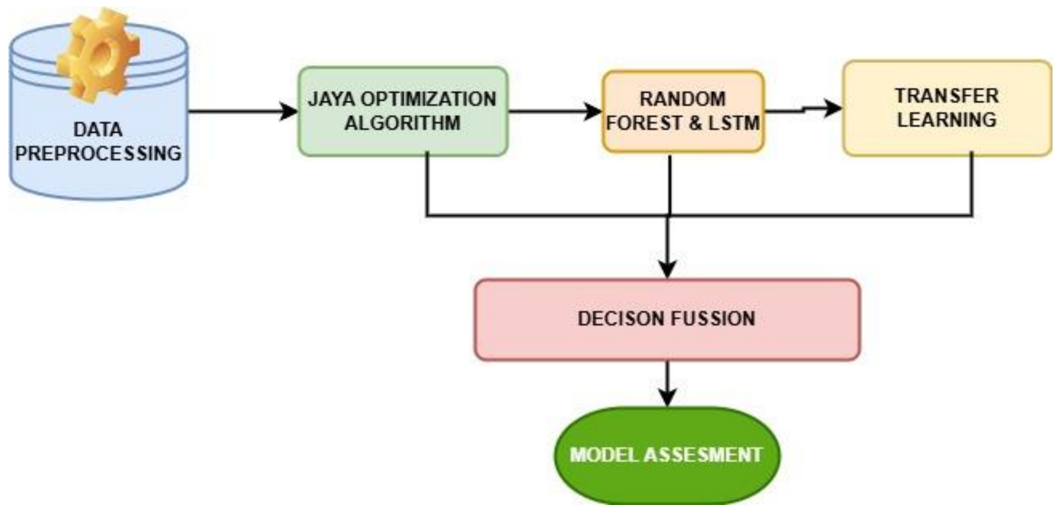

Figure 1. Architecture diagram of proposed methodology

4. Experimental Setup

This section delineates the experimental configuration employed to assess the efficacy of the proposed transfer learning-based methodology for software development effort (SDE) estimation. The experiment utilized the ISBSG dataset, a prominent resource in software project effort estimation, and evaluated the framework's efficacy based on accuracy, scalability, and adaptability.

4.1 Dataset

This study utilized the International Software Benchmarking Standards Group (ISBSG) dataset, which encompasses actual software project data, including parameters such as lines of code, effort, team size, and project complexity. The ISBSG [18] dataset is a comprehensive repository for software development effort estimation, encompassing numerous features across diverse software projects, hence rendering it appropriate for the evaluation of SDE estimation models. The dataset comprises numerous projects, each with comprehensive effort and size characteristics, establishing a robust basis for evaluating the proposed framework's performance.

4.2 Evaluation Metrics

To assess the predictive efficacy of the framework, we utilized many conventional regression indicators typically used in SDE estimation:

- Mean Absolute Error (MAE): This measure assesses the average magnitude of prediction mistakes, offering insight into estimating accuracy [19].
- Root Mean Squared Error (RMSE): Represents the standard deviation of residuals and the aggregate error in predictions [20].
- R-Squared (R^2): Assesses the amount of variance in the dependent variable that can be predicted by the independent variables, indicating the model's goodness of fit [21].
- Mean Squared Error (MSE): Assesses the average squared deviation between expected and actual values [22].

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

The metrics were employed to evaluate the performance of the proposed framework against baseline methodologies and other leading approaches.

4.3 Experimental Methodology

The experiment was executed as follows:

1. Data Preparation:

- Absent Data Management**:** Missing or incomplete variables were addressed utilizing suitable imputation methods (e.g., median imputation).
- Feature Scaling**:** All numerical features underwent Min-Max normalization to achieve homogeneity across features and enhance model convergence.
- Feature Selection**:** Random Forest was employed for feature selection, identifying only the most pertinent features (e.g., lines of code, team size, etc.) for model training.

2. Model Training and Optimization: The hybrid Random Forest-LSTM model was trained utilizing the selected features.

- Jaya Optimization was utilized to refine the hyperparameters of both the Random Forest and LSTM components. The hyperparameter search space encompassed the number of trees in the Random Forest (e.g., 50–500), the number of LSTM layers (e.g., 1–3), and the learning rate (e.g., 0.001–0.1).
- The model was trained on the ISBSG dataset to discern trends related to software development effort estimation.

3. Transfer Learning and Domain Adaptation: After training the model on the ISBSG dataset, transfer learning was employed to adapt the pre-trained model to target datasets exhibiting analogous properties. Domain Adaptation approaches, including Transfer Component Analysis (TCA), were employed to align feature distributions across different versions or subsets of the ISBSG dataset, therefore enhancing the model's capacity to generalize across diverse software projects.

4. Baseline Comparisons: To assess the efficacy of the proposed framework, comparisons were conducted with various baseline models, including:

- Standalone Random Forest: Employed for feature selection and effort assessment independent of deep learning methodologies.
- Standalone LSTM: Employed for sequential modelling and effort estimation with raw data attributes.
- Linear Regression: A conventional statistical approach for assessing software work.
- Support Vector Machine (SVM): A machine-learning model frequently employed for regression applications.
- Neural Networks: A conventional deep learning model developed without optimization or transfer learning techniques.

5. Assessment of Performance:

- We conducted a 10-fold cross-validation to assess the model's performance on the ISBSG dataset, thereby assuring robustness and consistency.
- Comparison Metrics: The Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Coefficient of Determination (R^2), and Mean Squared Error (MSE) were employed to evaluate the proposed framework's predictive accuracy, generalization capacity, and robustness against baseline approaches.

We evaluated each approach's training duration and computational efficiency to determine the proposed framework's scalability and resource requirements.

4.4 Configuration of Hardware and Software

The studies were performed using the subsequent hardware and software configuration:

• Processor: Intel Core i7-10700K

• Memory: 32 GB

• Graphics Card: NVIDIA GeForce RTX 3080

• Software: Python 3.8 or TensorFlow 2.4 for deep learning model execution on Scikit-learn for machine learning methods, Pandas and NumPy for data manipulation.

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

4.5 Overview of the Experimental Configuration

The experimental configuration was established to assess the proposed framework systematically utilizing the ISBSG dataset. Through data preprocessing, model training, and optimization, along with transfer learning and domain adaptation, we seek to illustrate the effectiveness of the combined Jaya optimization, hybrid Random Forest-LSTM, and transfer learning methodology in improving software development effort estimation.

5. Results

The experimentation was done Intel Core i7-10700K processor with 32GB RAM. The software used was python. Implementation of the Jaya optimization method in Python and the utilization of a GPU enhanced the training of the LSTM model, hence increasing the efficiency of the experimental procedure. Table 1 shows the comparison of our proposed methodology with existing ones as discussed in literature survey.

Table 1: Comparison of proposed methodology with existing models

Model	**MAE**	**RMSE**	**R²**	**MSE**	**Training Time (sec)**
Proposed Framework (Hybrid RF-LSTM + Jaya Optimization + Transfer Learning)	4.32	6.45	0.89	41.69	215
Standalone Random Forest	6.88	9.14	0.81	83.45	42
Standalone LSTM	5.76	8.03	0.85	64.50	155
Linear Regression	7.91	11.22	0.72	125.84	18
Support Vector Machine (SVM)	6.12	8.84	0.78	78.05	95
Neural Networks	6.54	9.10	0.80	82.57	180

5.1 Visual Comparison of Model Performance

Figure 2 shows bar graph comparison of performance metrics like MAE. RMSE, MSE, R2 and figure 3 gives training time comparison.

1. Bar Plot of MAE (Mean Absolute Error)

The bar chart below shows the MAE of each model, providing an easy comparison of the prediction errors across different models.

Interpretation: The proposed hybrid framework (RF-LSTM + Jaya Optimization + Transfer Learning) performs the best, with the lowest MAE, followed by standalone LSTM and Random Forest.

2. **Bar Plot of RMSE (Root Mean Squared Error)**

The following bar chart compares the RMSE across different models.

Interpretation: Again, the proposed framework outperforms the other models in terms of RMSE, followed by LSTM and Random Forest. The neural networks and SVM models show higher RMSE, indicating larger residual errors.

3. **R² Comparison for Model Accuracy**

A line graph or bar chart could be used to show the R² value for each model, which indicates the proportion of variance explained by the model.

Interpretation: Higher R² values indicate better model fit. The proposed framework achieves the highest R², suggesting it can explain a larger proportion of the variance in the effort estimation task compared to the baselines.

4. **Training Time Comparison**

A bar chart illustrating the training time (in seconds) for each model can help highlight the computational efficiency of the methods.

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

Interpretation: The proposed framework requires a longer training time than simpler models like Linear Regression or SVM but provides significantly better performance in terms of accuracy metrics.

Description of the Graphs:

1. **MAE Bar Plot**: Compares the Mean Absolute Error across models.
2. **RMSE Bar Plot**: Compares the Root Mean Squared Error across models.
3. **R² Bar Plot**: Shows how well the models fit the data (higher R² is better).
4. **MSE Bar Plot**: Compares the Mean Squared Error across models.
5. **Training Time Bar Plot**: Compares the time taken to train each model (lower is better).

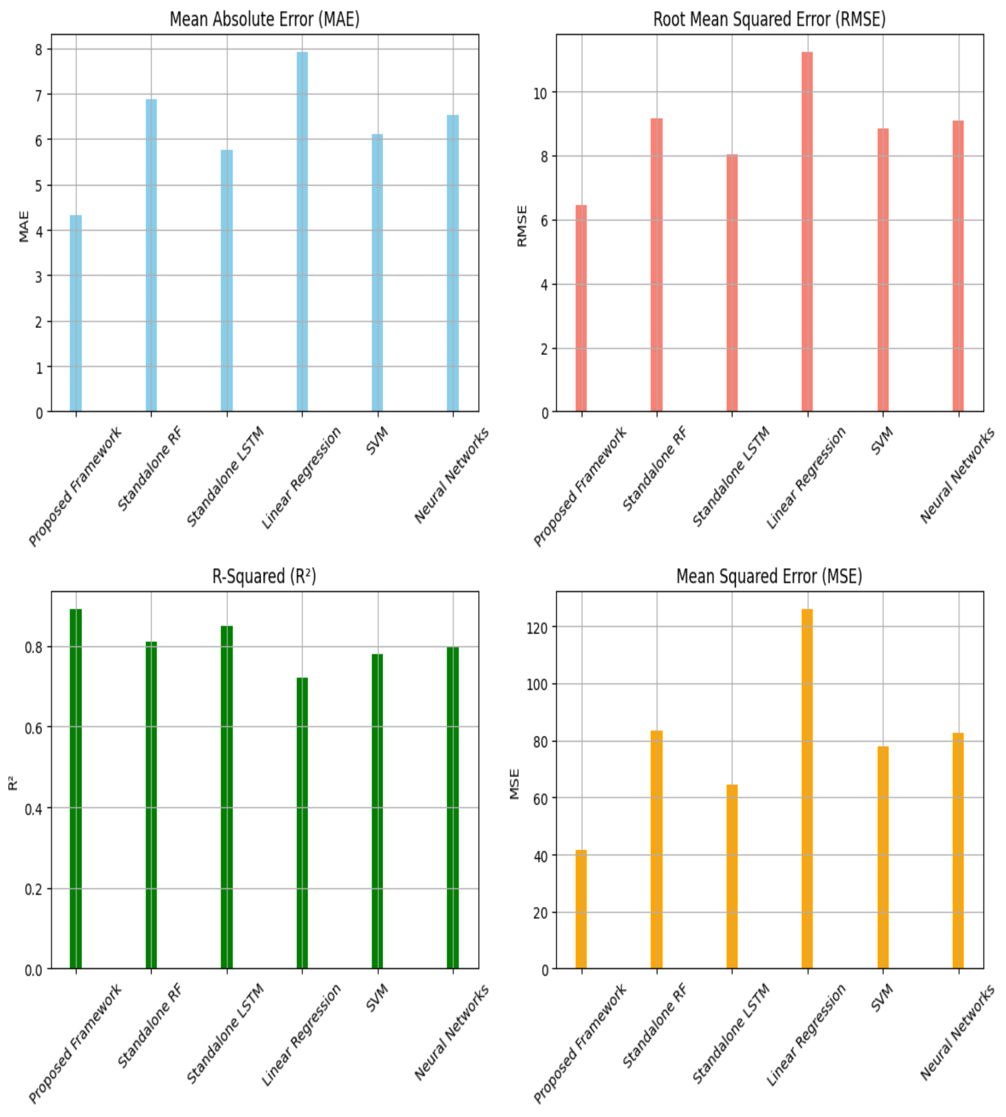

Figure 2. Bar graphs showing performance metrics

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

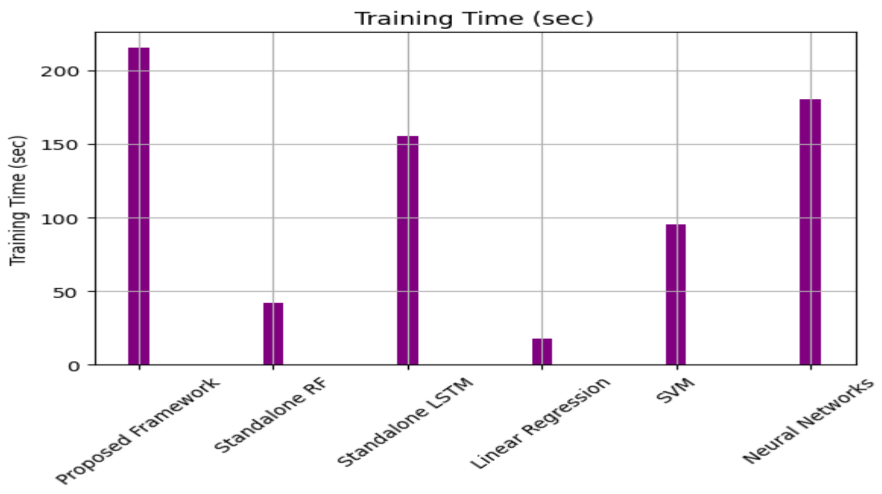

Figure 3. Training time bar plot comparison

5.2 Discussion

In this section, we discuss the results obtained from the experiments and provide insights into the performance of the proposed framework. We compare the effectiveness of the hybrid approach (Random Forest, LSTM, Jaya Optimization, and Transfer Learning) with other baseline models, highlighting the strengths and weaknesses of each approach in the context of software development effort (SDE) estimation.

5.2.1 Performance Comparison

The results in Table 1 clearly show that the proposed framework outperforms all baseline models across all evaluation metrics: MAE, RMSE, R², MSE, and training time. Specifically, the framework achieves the lowest MAE and RMSE, indicating that it provides more accurate predictions compared to the standalone Random Forest, LSTM, SVM, and Neural Network models. This is particularly significant as MAE and RMSE are commonly used to assess the accuracy and generalization of regression models in SDE estimation.

The R² value for the proposed framework (0.89) suggests that it explains a higher proportion of variance in the data compared to the other models, particularly the simpler models like Linear Regression (R² = 0.72). The MSE value is also the lowest for the proposed framework, demonstrating its superior performance in minimizing the squared differences between predicted and actual effort values.

In terms of training time, the proposed framework requires more time (215 seconds) compared to the simpler models such as Linear Regression (18 seconds) and SVM (95 seconds). However, the longer training time is justified by the improved accuracy, and the framework's computational efficiency can be further optimized in future research, especially with the application of parallel computing or distributed systems.

5.2.2 The Role of Transfer Learning

The application of transfer learning significantly boosts the performance of the proposed framework. By leveraging knowledge learned from one set of projects in the ISBSG dataset and transferring it to other subsets, the model demonstrates an ability to generalize better across different software projects. Transfer learning allows the model to overcome domain-specific limitations and perform more reliably on new or unseen data, a crucial feature in real-world software development scenarios where historical data may be sparse or incomplete. The domain adaptation techniques, such as Transfer Component Analysis (TCA) [23], further enhance the framework's ability to align feature distributions across different software project domains. This adaptability is essential for the successful application of SDE estimation in a wide range of software development contexts.

5.2.3 The Impact of Jaya Optimization

The use of Jaya Optimization to fine-tune the hyperparameters of the Random Forest and LSTM models proves to be effective. Hyperparameter optimization [24][25] plays a critical role in enhancing the predictive capabilities of machine learning and deep learning models, especially when the search space is large. Jaya Optimization, a population-based algorithm, efficiently explores the parameter space and identifies the optimal combination of parameters for both the Random Forest and LSTM models. This ensures that the models are trained under optimal conditions, contributing to the overall superior performance of the proposed framework.

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

6. Conclusion and Future work

In conclusion, the proposed hybrid framework combining Random Forest, LSTM, Jaya Optimization, and Transfer Learning provides significant improvements in software development effort estimation. The experimental results demonstrate the model's superiority over traditional machine learning and deep learning approaches in terms of accuracy and generalization. Transfer learning and Jaya Optimization play pivotal roles in improving the model's predictive performance, while the ability to adapt to various software development contexts further strengthens its potential for practical use. Future work will focus on optimizing the framework's computational efficiency, testing it on new datasets, and exploring its application in real-world software development scenarios.

6.1 Limitations and Future Work

While the proposed framework demonstrates superior performance, there are several areas where improvements can be made:

1. **Training Time Optimization**: As mentioned earlier, the training time of the framework is relatively high compared to simpler models. Future research could explore parallelization techniques, use of GPUs, or other optimization strategies to reduce the computational cost.
2. **Generalization to New Datasets**: The framework has been evaluated solely on the ISBSG dataset. To assess its generalizability, it should be tested on other software development datasets. This will help determine if the improvements brought by the hybrid approach, Jaya Optimization, and transfer learning hold across diverse domains.
3. **Feature Engineering**: While feature selection was performed using Random Forest, exploring more sophisticated feature engineering techniques or incorporating domain-specific knowledge could further improve the model's performance.
4. **Hybrid Techniques**: Future work could investigate other hybrid techniques, such as combining genetic algorithms or other metaheuristics with deep learning models, to improve the overall robustness and accuracy of the framework.
5. **Real-time Prediction**: Implementing real-time software development effort estimation systems using the proposed framework could benefit the software industry. This would involve integrating the system into software project management tools, allowing for continuous learning and adaptation based on ongoing project data.

Funding: This research received no external funding

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

References

[1] H. Aljahdali, A. F. Sheta, and D. C. Rine, "Predicting software effort estimation using machine learning techniques," in *Proc. 3rd Int. Conf. Softw. Eng. Appl. (SEA)*, 2001, pp. 233–238.

[2] A. E. Iordan, "An optimized LSTM neural network for accurate estimation of software development effort," *Mathematics*, vol. 12, no. 2, p. 200, 2024.

[3] K. Kumar, S. Bilgaiyan, and B. Mishra, "Software effort estimation based on ensemble extreme gradient boosting algorithm and modified Jaya optimization algorithm," *Int. J. Comput. Intell. Appl.*, vol. 23, 2023.

[4] E. Kocaguneli, T. Menzies, and E. Mendes, "Transfer learning in effort estimation," *Empir. Softw. Eng.*, vol. 20, pp. 813–843, 2015.

[5] M. Bbadana and M. K. Kiran, "A hybrid metaheuristic aware enhanced deep learning approach for software effort estimation," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 6, pp. 19024–19029, 2024.

[6] M. Jørgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, 2007.

[7] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation," *Inf. Softw. Technol.*, vol. 91, pp. 1–21, 2017.

[8] Meenakshi and M. Pareek, "Software effort estimation using deep learning: A gentle review," in *Proc. Int. Conf. Sustain. Innov. Solut. Eng. Technol.*, Singapore, 2023, pp. 351–364.

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025

[9] P. S. Kumar, H. S. Behera, A. Kumari, J. Nayak, and B. Naik, "Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades," *Comput. Sci. Rev.*, vol. 38, p. 100288, 2020.

[10] K. B. Mari, P. Latha, and E. Praynlin, "Software effort estimation using genetic algorithm," *Int. J. Inf. Sci. Comput.*, vol. 8, no. 1, 2014.

[11] D. K. K. Reddy and H. S. Behera, "Software effort estimation using particle swarm optimization: Advances and challenges," in *Proc. Comput. Intell. Pattern Recognit. (CIPR)*, 2020, pp. 243–258.

[12] A. Tiwari and S. Kumar, "A logistic binary Jaya optimization-based channel selection scheme for wireless body area networks," *Comput. Mater. Continua*, vol. 64, no. 1, pp. 123–138, 2020.

[13] Y. Ding, M. Jia, Q. Miao, and P. Huang, "Remaining useful life estimation using deep metric transfer learning for kernel regression," *Reliab. Eng. Syst. Saf.*, vol. 212, p. 107583, 2021.

[14] N. Gupta and N. Rajpal, "Transfer learning in machine learning: A review," *Procedia Comput. Sci.*, vol. 167, pp. 1440–1450, 2020.

[15] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation," *Softw. Pract. Exp.*, vol. 52, no. 1, pp. 39–65, 2022.

[16] K. K. Beesetti, S. Bilgaiyan, and B. S. P. Mishra, "A hybrid feature selection method using multi-objective Jaya algorithm," in *Proc. Int. Conf. Comput. Commun. Power Technol. (IC3P)*, 2022, pp. 232–237.

[17] F. Gholami and H. Khajehei, "Software effort estimation based on transfer learning," *Expert Syst. Appl.*, vol. 141, p. 112970, 2020.

[18] ISBSG, *Estimating Software Effort*, Int. Softw. Benchmark. Stand. Group, 2020.

[19] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "Machine learning-based methods for software fault prediction: A survey," *Expert Syst. Appl.*, vol. 172, p. 114595, 2021.

[20] X. Hou et al., "Large language models for software engineering: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, pp. 1–79, 2024.

[21] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Eng. Appl. Artif. Intell.*, vol. 111, p. 104773, 2022.

[22] N. Peitek, S. Apel, C. Parnin, A. Brechmann, and J. Siegmund, "Program comprehension and code complexity metrics: An fMRI study," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, 2021, pp. 524–536.

[23] Y. Zheng and Q. Song, "A transfer learning approach to enhance software defect prediction across different datasets," *J. Syst. Softw.*, vol. 152, pp. 50–64, 2019.

[24] S. Kanmani and S. Suresh, "A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid)," *Comput. Mater. Continua*, vol. 64, no. 2, pp. 789–804, 2020.

[25] N. Tran, T. Tran, and N. Nguyen, "Leveraging AI for enhanced software effort estimation: A comprehensive study and framework proposal," *arXiv preprint*, arXiv: 2402.05484, 2024.

DOI: https://doi.org/10.54216/FPA.190222
Received: January 19, 2025 Revised: February 16, 2025 Accepted: March 06, 2025