# Multi View Neural Network for Software Effort Estimation Prediction

**Boy Setiawan[1] and Agus Subekti[1]**
[1]Faculty of Computer Science, Nusa Mandiri University, Jakarta, Indonesia

**Corresponding author:** Boy Setiawan (e-mail: 14230023@nusamandiri.ac.id).

**ABSTRACT** Software Effort Estimation (SEE) is a critical challenge in software project management, dating back to the early years of software engineering. Accurate estimation of the effort required for software development is essential for project planning, resource allocation, and risk management. Incorrect effort estimates can result in poor resource distribution, cost overruns, missed deadlines, and even complete project failure. This issue is increasingly urgent today as software systems are deeply embedded in almost every product and service, amplifying the need for reliable and accurate predictions. Over the years, several methods for SEE have been proposed, ranging from algorithmic models to expert judgment. More recently, machine learning (ML) approaches such as Case-Based Reasoning (CBR), Support Vector Machines (SVM), Decision Trees (DT), and Neural Networks (NN) have gained attention for their ability to model complex, nonlinear relationships inherent in SEE tasks. In this study, we propose a novel approach based on multi-view learning with NN (MVNN), which leverages multiple views from existing datasets, thus improving performance and generalization, particularly when the available data is small and scarce. The effectiveness of the MVNN model is validated through empirical comparisons with existing SEE models, demonstrating its potential to enhance SEE accuracy and improve prediction reliability.

**KEYWORDS** Multi View, Neural Networks, Scarce Dataset, Software Effort Estimation

## I. INTRODUCTION

SEE is recognized as one of the earliest and most critical challenges in software project management, with its conceptual roots dating back to the formative years of software engineering [1]. The ability to reliably estimate the effort required for successful software development is fundamental to effective project planning, resource allocation, and risk management. Although the terms effort and cost are often used interchangeably in practice, they represent distinct constructs; nonetheless, both are integral to ensuring project feasibility and delivery. Inaccurate estimation of software effort can result in suboptimal resource distribution, leading to cost overruns, missed deadlines, and ultimately project failure. This issue becomes even more pressing in the current era, where software systems underpin a significant portion of goods and services, intensifying organizational reliance on accurate estimation. Despite its longstanding relevance, SEE continues to present considerable difficulties for software teams and project managers, necessitating its consideration from the earliest stages of project development [2]. Consequently, a substantial body of research has emerged focusing on the advancement of models, techniques, and empirical strategies aimed at enhancing the precision, efficiency, and applicability of software effort estimation in various development contexts [3].

The scope of SEE extends beyond resource allocation; it plays a critical role in facilitating communication among both internal and external stakeholders concerning planning, budgeting, financial oversight, and implementation proposals. SEE enables organizations to establish budgets and allocate funds effectively while offering insights into the anticipated costs of software development [4]. More broadly, the capacity to generate realistic effort estimates empowers executives and managers to make informed decisions, mitigate risks, and identify factors that could contribute to project failure, such as complexity, technology constraints, and team requirements. Since inaccurate effort estimation often leads to project crises, providing an accurate approximation of the resources needed to meet project objectives—while ensuring the delivery of products and services that fulfil both functional and non-functional requirements—can significantly reduce the likelihood of project failure [1].

Measuring software sophistication early in the project lifecycle and making accurate estimations is a complex task, posing significant challenges for both managerial and development roles [5]. Unlike traditional manufacturing, software engineering is primarily a human-intensive process.

Over the past 50 years, the software industry has undergone substantial evolution, with at least four generations of programming languages and three major development paradigms [1]. This progression has been further complicated by the rapid advancement of development technologies, constant paradigm shifts, and ongoing changes in methods and tools. Moreover, the nature of software development has transitioned from being the responsibility of a single contractor to distributed projects, where teams are dispersed across various companies, time zones, cultures, and even continents, further enhancing the intangible and volatile nature of software products.

Over the years, researchers have developed a variety of effort estimation methods, with each new approach generally exhibiting increased sophistication. These methods are extensively covered in the SEE literature and are typically categorized into three main groups: algorithmic, non-algorithmic, and ML methods (see Figure 1). Initially, SEE relied on non-algorithmic expert judgment, a straightforward approach to generate realistic estimates [6]. The Delphi technique and work breakdown structure (WBS) are among the most widely used expert judgment methods. In the Delphi technique, a meeting is convened with project experts, and through discussion and argumentation, a consensus estimate is derived. In contrast, the WBS method involves breaking down the entire project into smaller sub-projects or tasks, continuing until the baseline activities are reached. This hierarchical decomposition allows for more accurate effort estimations to smaller and more manageable sub-tasks.

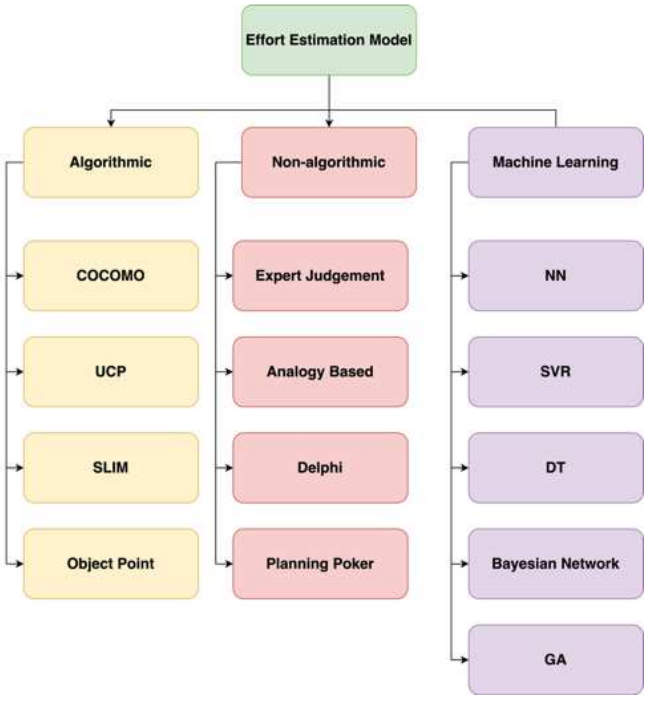

Figure 1. SEE Methods

Algorithmic approaches leverage statistical and mathematical principles for SEE. Notable examples of estimation techniques within this category include Constructive Cost Model (COCOMO)-II, Putnam Software Life Cycle Management (SLIM), SEER-SEM, and True Planning. The primary input for these models is the size of the software being estimated, which is typically measured using metrics such as function points (FP), source lines of code (LOC), or use case points [2].

ML techniques have recently demonstrated their effectiveness, particularly in estimating the effort required for software projects. Among these techniques, Case-Based Reasoning (CBR) stands out as one that leverages the history of successfully completed projects to predict solutions for new cases [3]. CBR is particularly promising for effort estimation because it emulates human reasoning by referencing past projects that were successfully implemented and using their actual effort data to predict the effort required for current projects. Recently, Bayes' theorem has garnered significant attention as a potential approach for managing estimation uncertainty and integrating quantitative data with subjective human judgment [1]. In addition to other well-known ML algorithms such as SVM, DT, Linear Regression (LR), and K-Nearest Neighbours (KNN), the growing complexity and variability of software projects have driven the adoption of NN in SEE [7]. One of the key advantages of using NNs is their capacity to model complex, nonlinear relationships that are inherent in software projects. Moreover, advancements in NN architectures have significantly enhanced the accuracy of estimations [8]. Additionally, innovations such as Genetic Algorithms (GA), Convolution, and metaheuristic techniques integrated with NNs have improved the convergence and adaptability during the training phases, leading to more accurate software effort estimates.

Several recent studies [9], [10], [11], [12] highlight the challenges associated with assembling and analysing empirical software engineering datasets. The SEE research community has recognized and prioritized issues such as noise, outliers, and missing data, where simple approaches like the “majority rule” may not be applicable. A key example of this challenge is the treatment of outliers, where common practices of exclusion must be approached with caution, as extreme estimates may sometimes be the most accurate [1]. Removing such values could also distort the dataset distribution, which is particularly problematic since many ML algorithms assume a normal distribution dataset. Additionally, issues such as poor provenance, data inconsistency, and commercial sensitivity in software estimation datasets have been largely overlooked [13]. These issues complicate the accuracy of SEE, especially for NN models, where the availability of sufficient, high-quality data is crucial for improving model performance.

When data is scarce and acquiring additional data from external sources presents significant challenges, particularly due to confidentiality concerns in software projects. It becomes essential to maximize the utility of existing datasets. Standard ML techniques used in SEE typically rely on a single input for training. However, SEE can also be approached using multiple views, where multiple feature vectors are utilized [14]. Multi-view learning is an emerging area in ML that leverages multiple perspectives or feature sets to enhance generalization, commonly referred to as data fusion or data integration [15]. The goal of multi-view learning is to model each view independently while jointly optimizing all models to improve overall generalization performance. A significant advantage of this approach is its ability to boost generalization by generating multiple views manually to enhance performance. Although various multi-view ML methods, such as sparse multi-view time SVM [16] and multi-view discriminant analysis (DA) [17], have demonstrated effectiveness in classification tasks, to the best of our knowledge this is the first study applying multi-view learning using NN within the SEE domain.

Based on the problems stated above, we are focussing our study to use multiple-views with NN using common datasets for SEE to construct a high-quality multi-view NN (MVNN). On the issue of outlier, we opted a pre-processing method to preserve as much information available by applying scaling to reduce the high value impact of the outlier rather than eliminating it from the dataset and apply -1 to null values . In order to maximize existing datasets, we are using the same datasets to generate a different view by using the output of a dimensionality reduction algorithm to increase performance and generalization of the proposed MVNN. At the end, we will validate our findings with an empirical comparison from previous studies to show the competitiveness of our proposed method.

The key contributions of this works are as follows:

1. This study proposes a novel SEE model based on MVNN to construct a prediction model that enhance the contribution of SEE in software engineering.
2. We proposed a novel way to generate a different view of the datasets by utilizing a dimensionality reduction algorithm to produce the same datasets in a different latent space.
3. Finally, to verify the performance of the proposed method, we conducted experiments on various SEE dataset with existing SEE models.

This paper follows the following structure. Section 1 gives introduction on the problem domain. Section 2 provides literature overview of the relevant SEE work. The presentation of our research methodology and experimental setups follow in section 3. The experimental results are presented in Section 4 along with the threats to internal, external, construct of our study and conclusions are covered in Section 5.

## II. THEORETICAL FRAMEWORK

In this section we briefly introduced theoretical review which underline our proposed MVNN method and related works on SEE from previous studies.

### A. MULTI-VIEW NEURAL NETWORK

With the growing volume and diversity of data in recent years, the interest in multi-modal and heterogeneous representations has surged, driven by the desire to enhance learning performance. MVNN have emerged as an effective approach for integrating multiple data representations into a unified predictive model [18]. MVNNs refer to NN architectures that incorporate multiple feature representations (views) from the same data instance to enhance learning performance, leveraging both redundant and complementary information across all modalities [15]. A key challenge lies in effectively representing and summarizing multimodal data to fully exploit the complementarity and redundancy of the multiple modalities in the dataset [19]. One straightforward approach to addressing multiple modalities is early fusion, which involves concatenating features from individual modalities immediately after extraction, resulting in joint representations or unimodal data [20]. This approach aligns well with NN, which excel at handling such unified representations and have become a popular method in various tasks. While several ML algorithms, such as kernel-based SVM are used for multi-view classification problems, NN have demonstrated exceptional performance in tasks such as face recognition, object detection, and classification with MVNNs [15]. The superior performance of NN-based joint representations, coupled with the ability to pre-train models in an unsupervised manner, has further fuelled their popularity. However, their performance is highly dependent on the availability of large amounts of training data. Despite their many advantages, one limitation of NN is their inability to effectively handle missing data, although strategies exist to mitigate this issue [21].

### B. SOFTWARE EFFORT ESTIMATION DATASETS

SEE plays a pivotal role in software project management, enabling accurate estimation of the effort required for a successful software project completion including project planning, budgeting, and execution of the project. Various datasets have emerged as valuable resources for researchers and practitioners in this field, specifically targeting the challenge of estimating the effort required for software development. These datasets typically consist of historical data from previous software projects, encompassing various project attributes, such as size, complexity, effort, and other relevant factors which are used for training and evaluation of SEE models. In the SEE domain, several publicly available datasets have been widely used for model training and testing. Notable examples include the COCOMO datasets, such as COCOMO-81, COCOMO NASA-V1, and COCOMO NASA-V2, which provide a detailed record of software development effort based on various attributes like lines of code, function points, and other software metrics. Other commonly used datasets include Desharnais, China, and Maxwell, which offer a diverse set of project data to evaluate the performance of estimation models. The challenge in SEE datasets lies not only in the availability of accurate data but also in addressing common issues such as missing values, noise, and outliers. Furthermore, the rapidly evolving nature of software development practices and technologies means that datasets must be continuously updated to reflect current trends and methodologies. In this context, datasets with a diverse range of software projects and environments are crucial for the generalization of SEE models across various domains. A comprehensive understanding of the available datasets is vital, as highlighted by [22], who identified 12 publicly accessible datasets including Albrecht, COCOMO-81, and COCOMO NASA-V2 among others. They specifically extracted the China and Maxwell datasets due to their structural quality and content suitability for machine learning applications in estimating software effort.

In this study, several SEE datasets were used to validate our proposed method and compared with results from previous studies. A summary description of the datasets used can be seen in Table I.

TABLE I
ESS DATASETS

| Dataset | Records | Attri-butes | Effort | Size (Unit measurement) |
|---|---|---|---|---|
| Desharnais | 81 | 11 | Person-hours | Function point |
| China | 499 | 18 | Person-hours | Function points |
| COCOMO NASA-V1 | 60 | 16 | Person-months | LOC |
| COCOMO NASA-V2 | 93 | 22 | Person-months | LOC |
| COCOMO-81 | 63 | 17 | Person-months | LOC |

## C. K-FOLD CROSS-VALIDATION

K-fold cross-validation is an effective technique for assessing the performance of ML models. This method involves partitioning the dataset into K subsets or folds. A model is trained K times, each time using K-1 folds for training and the remaining fold for validation. This process allows for a robust evaluation of the model's performance, as it mitigates issues related to overfitting and provides a better estimate of model generalizability [23]. The choice of K can significantly impact the effectiveness of cross-validation. While 10-fold cross-validation is a common choice in the literature, studies suggest that the optimal K could vary depending on the dataset and the modelling context. For instance, Okfalisa et al. argue that although 10-fold is standard, there's no one-size-fits-all solution, and K can be adjusted based on the dataset size and specific requirements [24]. Furthermore, increasing K might reduce bias but could also lead to higher variance in model performance estimates [25].

## D. MIN-MAX SCALER

The Min-Max scaler adjusts the scale of an attribute by shifting its values along the x-axis, ensuring that the transformed attribute's values fall within the interval of (0, 1) [26], according to this formula:

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

In (1), the scaling factor is determined by the attribute's range, while the translational term is set as its minimum value. This approach guarantees that the attribute's values are transformed to a minimum of zero and a maximum of one which is the ideal value for NN input.

## E. ISOMETRIC FEATURE MAPPING

Isometric Feature Mapping (Isomap) is a widely used technique for non-linear dimensionality reduction technique to overcome high dimensionality in a dataset compare to Principal Component Analysis (PCA) which excels on linear dataset. The most distinct feature of Isomap lies in its versatility tested across various applications, ranging from image processing, fault prediction in electromechanical systems, and anomaly detection in hyperspectral imagery [27]. Introduced in 2000 by Tenenbaum et al. [28] as an improvement of multidimensional scaling (MDS) by replacing geodesic distances rather than Euclidean distances, this improvement allows Isomap to capture the true manifold structure of the dataset [29]. Beside the advantages, Isomap performs sub optimally when processing data that encompasses multiple clusters or manifold structures, but this drawback has spurred the development of modifications, including extensions from the original Isomap such as FastIsomap and Landmark Isomap, aimed at enhancing computational efficiency and the ability to handle more complex datasets effectively [30].

## F. EVALUATION MEASURES

Evaluation measures typically reflect the performance of ML predictive result. In this paper, there are five main measures to validate the effectiveness of the proposed model: mean absolute error (MAE), mean square error (MSE), mean magnitude relative error (MMRE), root mean square error (RMSE), R-squared (R2) and median magnitude relative error (MdMRE) as shown in (2) until (7) respectively.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (2)$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (3)$$

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \quad (4)$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (5)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (6)$$

$$\text{MdMRE} = \text{median}\left(\frac{|y_i - \hat{y}_i|}{y_i}\right) \quad (7)$$

## III. PREVIOUS RESEARCH

Numerous studies have been conducted over time to enhance the efficacy of SEE, particularly in predicting effort estimation early in the software development lifecycle (SDLC). Researchers have explored various techniques, including ML, NN, and hybrid methods, which integrate multiple approaches to develop the most effective SEE models. One notable study by [3] combined CBR with GA to optimize key CBR parameters, such as feature selection (FS), feature weighting, similarity measures, and the number of nearest neighbours (k). The results demonstrated the effectiveness of GA in producing an accurate SEE model. Another study proposed by [31], focused on applying CBR with a bisecting k-medoids clustering algorithm to better understand datasets and identify the most relevant cases for effort estimation. This approach involved removing unrelated projects to find the best k analogies for each new project requiring estimation. Empirical experiments on different datasets indicated that the optimal k value depends on the specific characteristics of the dataset. In the study of [32], the authors explored SEE estimation based on analogies utilizing distance similarity measures such as Euclidean, Manhattan, and Minkowski. Their results showed that the Manhattan similarity measure yielded the highest accuracy, with a 50% MMRE, 28% MdMRE, and 48% prediction accuracy (PRED). In a study by [33], SEE model named DEAPS was proposed, which is based on the differential evolution algorithm using the Desharnais dataset. The model employed the Euclidean distance similarity measure to reduce the set of historical projects to a subset of similar projects, followed by the application of the differential evolution algorithm to refine and retrieve the best solutions. The results of this model showed significant improvements in analogy-based effort SEE.

In the field of ML, a study by [34] compare the performance of Random Forest (RF), SVM, DeepNet, and NN. Their findings concluded that RF outperformed the other methods when applied to the Desharnais, Maxwell, China, and Albrecht datasets. Rahman et al. [2], compared three SEE forecasting algorithms: DT, Support Vector Regression (SVR), and KNN. They processed and analysed the datasets, applying the proposed algorithms and evaluating the models based on three criteria: MAE, MSE, and $R^2$. The study demonstrated that DT outperformed the other algorithms. Alhazmi et al. [35] employed bagged learning with base learners such as LR, SMOReg, NN, RF, REPTree, and M5 rule for SEE. They also implemented FS algorithm to assess the impact of the BestFit FS algorithm and GA, using the China dataset for evaluation. The results revealed that the M5 packing rule with GA as FS achieved an average relative error size of 10%, making it more effective than the other algorithms. Varshini et al. [6] presented both single and combined techniques which included combinations of individual methods. They used RF, SVM, DT, stacking with SVM, and stacking with RF and conducted experiments on the Albrecht, China, Desharnais, Kemmerer, Kitchenham, Maxwell, and COCOMO-81 datasets. The models were evaluated using MAE, RMSE, and $R^2$, with the results showing the superiority of RF over other models, including ML algorithms and clustering techniques. Zakaria et al. [36] introduced a model based on SVM and LR for SEE, implemented through an application called SOFREST estimator. They applied RF, regression tree, LR, and SVM to the COCOMO Nasa-V1, COCOMO Nasa-V2, and COCOMO-81 datasets. The models were evaluated using multiple criteria, including MSE, RMSE, MAE, MdMRE, min–max accuracy, correlation accuracy, and P-value with the results demonstrating the superiority of the targeted algorithms across the datasets.

Fadhil et al. [37] introduced a model based on the Dolphin Swarm Algorithm (DSA) and the hybrid Bat Algorithm (DolBat) to enhance cost estimation models. The DSA is particularly effective for optimization tasks, requiring fewer individuals and fitness function calls while utilizing echolocation to more efficiently find optimal solutions. This study was conducted using the COCOMO NASA-V1 and NASA-V2 datasets. The model's performance was evaluated using the MMRE metric and was compared with other algorithms, such as GA. Vo Van et al. [38] proposed a model to assess the impact of data aggregation on SEE, aiming to identify the most effective aggregation method. This model, called Effort Estimation Using Machine Learning Applied to Clusters (EEAC), was evaluated using multiple metrics, including Mean Absolute Percentage Error (MAPE), RMSE, MAE, Mean Balance Relative Error (MBRE), and Mean Inverted Balance Relative Error (MIBRE). The experimental results demonstrated that estimation accuracy achieved through clustering consistently outperformed accuracy without clustering, for both Function Point Analysis (FPA) and the EEAC methods.

The application of NN is also widespread in SEE research. Sharma et al. [39] proposed four distinct methods for SEE prediction: Localized Neighbourhood Mutual Information-based NN (LNI-NN), Fuzzy-based NN (NFL), Adaptive GA-based NN (AGANN), and GEHO-based Neural Fuzzy Network (GEHO-NN). These models were applied to five datasets: COCOMO-81, COCOMO NASA-V1, COCOMO NASA-V2, China, and Desharnais, and evaluated using four prediction metrics: MMRE, RMSE, MdMRE, and PRED. Kassaymeh et al. [40] presented a model for SEE using a Fully Connected NN (FCNN) combined with a Gray Wolf Optimizer (GWO), termed GWO-FC. This model was tested on 12 datasets and evaluated based on several criteria, including MSE, Relative Absolute Error (RAE), MAE, Variance Accounted For (VAF), Manhattan Distance (MD), and RMSE.

## IV. RESEARCH METHOD

This section outlines the experimental procedures implemented in this study to assess the proposed SEE methods. Figure 2 illustrates a schematic representation of the experimental framework employed to validate the effectiveness of our proposed method. The framework was designed to facilitate an empirical evaluation of the models, where K-fold CV is applied to SEE datasets, and the model with the lowest MMRE from the best k-fold result is selected as the optimal model.

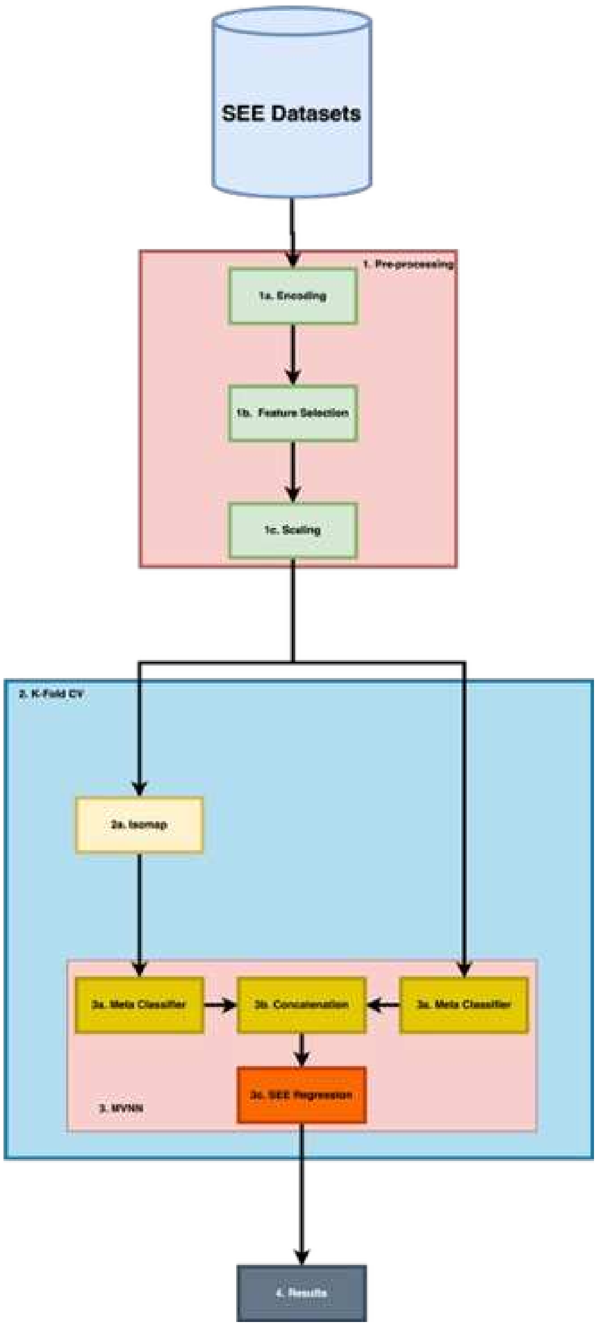

Figure 2. **Experimental Framework**

## A. PRE-PROCESSING

The pre-processing phase begins by converting any feature to numerical and dropping unnecessary features which does not add information in the dataset. In addition, the absence of null values is handled by setting it to -1. The last phase of pre-processing is utilizing scaling to bring down all input value to a same scale to avoid any outlier of affecting the final calculation. A brief result of the pre-processing is shown in Table II.

TABLE II
PRE-PROCESSED ESS DATASETS

| # | Dataset | Attributes | Drop Features | Null Value |
|---|---|---|---|---|
| 1 | Desharnais | 9 | Project, YearEnd | Yes |
| 2 | China | 18 | - | No |
| 3 | COCOMO NASA-V1 | 16 | - | No |
| 4 | COCOMO NASA-V2 | 20 | projectname, year | No |
| 5 | COCOMO-81 | 17 | - | No |

Although NN has the abilities to extract features and complex relationship in the dataset and has proven effective with or without feature engineering across various domains as detailed by [41] on comparing between ML and NN, the transformed dataset will boost NN dynamically to process input data, and learned to recognize patterns and assimilate high-level features in a hierarchical manner, effectively managing complexities in relationships among features. A summary of features used from each dataset can be seen in Table III.

TABLE III
FEATURES USED

| Dataset | Features | | | |
|---|---|---|---|---|
| Desharnais | TeamExp | Manager Exp | Language | Transactions |
| | Length | Entities | PointsNon Adjust | PointsAdjust |
| | Adjustment | | | |
| China | AFP | Input | Output | Enquiry |
| | File | Interface | Added | Changed |
| | Deleted | PDR_AFP | PDR_UFP | NPDR_AFP |
| | NPDU_UFP | Resou-rce | Dev.Type | Duration |
| | N_effort | effort | | |
| COCOMO NASA-V1 | RELY | DATA | CPLX | TIME |
| | STOR | VIRT | TURN | ACAP |
| | AEXP | PCAP | VEXP | LEXP |
| | MODP | TOOL | SCED | LOC |
| COCOMO NASA-V2 | cat2 | forg | center | mode |
| | rely | data | cplx | time |
| | stor | virt | turn | acap |
| | aexp | pcap | vexp | lexp |
| | modp | tool | sced | equivphyskloc |
| COCOMO-81 | dev_mode | rely | data | cplx |
| | time | stor | virt | turn |
| | acap | aexp | pcap | vexp |
| | lexp | modp | tool | sced |
| | loc | | | |

### B. K-FOLD CV

In this study, we employed 10 folds K-fold CV in the training phase with a constant random state for reproduction.

### C. MVNN

NN with two views as inputs will be used to predict SEE with the same architecture as shown in Figure 2. Both meta models consist of four layers of 256, 128, 64 and 32 nodes respectively with ReLu activation. The first meta-model input is the output of the pre-processing phase, while the output of the input transformation will be feed to the second meta-model. The outputs of both meta-models will be concatenated and feed to a layer of 2048 nodes and dropout (set at 0.1) for regularization before being feed to the final SEE regressor which will predict the final result. A depiction on the proposed MVNN is shown on Figure 3.

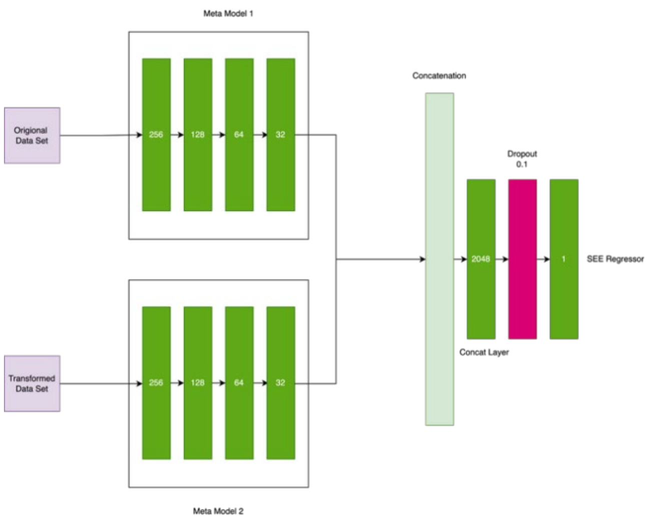

Figure 3. **Our proposed MVNN**

When data is scarce and acquiring additional data from external sources presents significant challenges—particularly due to confidentiality and privacy concerns, which are highly prevalent in software engineering projects—it becomes imperative to focus on maximizing the utility of the limited available datasets. In many real-world software engineering environments, especially those involving SEE, datasets are often small, incomplete, or imbalanced. This limitation is exacerbated by the sensitive nature of project data, which may include historical data or resource allocation records that organizations are unwilling or legally unable to share. As a result, conventional data-hungry approaches such as NN faces severe limitations in their applicability and generalization in SEE domain.

Traditional ML techniques commonly employed in SEE typically operate under a single-view paradigm. In contrast, multi-view learning—and more specifically, MVNN—presents a robust and scalable solution to this challenge. Multi-view learning leverages the idea that different "views" or feature groups, even if derived from the same underlying data instance, can offer both redundant and complementary information, which introduce new aspects or perspectives that a single view cannot provide alone. By simultaneously learning from all available views, MVNNs are capable of constructing a more holistic and discriminative representation of the data. Moreover, MVNNs help mitigate overfitting, which is a prevalent concern when training deep learning models on small datasets. By distributing the learning burden across multiple feature spaces, the model is less prone to memorizing noise or spurious correlations from any single view. Instead, it learns more robust and generalizable patterns that are supported by evidence across several data modalities. This becomes a particularly valuable property in SEE tasks, where the cost of misestimation can significantly impact project planning and resource allocation.

### A. INPUT TRANSFORMATION

The correlation analysis summarized in Table IV provides a comprehensive overview of the linear relationships between input features and the effort target variable across multiple SEE datasets used in this study. This statistical analysis is a crucial step to understand the predictive power of individual attributes and identifying whether traditional linear modeling assumptions hold in the context of SEE. Upon close inspection, it becomes evident that most of the features across these datasets exhibit low or even negative correlation coefficients with the effort target variable, strongly suggesting the presence of non-linear or complex relationships that cannot be captured effectively through simple linear models.

For the China dataset, a few features such as AFP (0.68), Added (0.69), and File (0.61) demonstrate moderately high positive correlations with effort, indicating that these variables may linearly contribute to SEE. This aligns with expectations, as Added functionality and function point metrics like AFP and File often scale with the complexity and size of the project, naturally increasing development effort. However, other features such as Deleted (0.07), Changed (0.11), and Interface (0.33) show very low correlations, suggesting they provide little to no linear explanatory power. The metric Dev.Type returns NaN, which might indicate missing or unprocessable data in this context. Furthermore, the moderately positive correlation of Duration (0.48) with effort is intuitive, though not strong enough. Notably, N_effort (0.99) shows an almost perfect correlation, likely because it is either a derived or a target-like variable.

The COCOMO NASA-V2 dataset presents a more nuanced picture. The majority of features show negative or weak positive correlations with the effort target. For example, forg (-0.37), mode (-0.33), and sced (-0.31) are negatively correlated with effort, suggesting either inverse relationships or the presence of non-linear dependencies. The relatively low positive correlations for center (0.42) and cplx (0.41) suggest that complexity and organizational structure have some influence, but not dominantly so. Interestingly, equivphyskloc (0.59) shows one of the higher correlations, reinforcing the long-standing view that code size (KLOC) remains a strong, albeit imperfect, predictor of effort. However, other typical COCOMO drivers like acap (-0.21) and aexp (-0.11) show negative or negligible relationships, raising questions about the linear assumptions often applied in parametric models such as traditional COCOMO.

The COCOMO NASA-V1 and COCOMO 81 mirror the trend of weak correlations. In COCOMO NASA-V1, LOC (0.92) shows a very strong positive correlation with effort, again affirming size as a core effort driver. However, other features—such as TURN (-0.18), AEXP (-0.18), and MODP (-0.18)—display low or negative correlations, underscoring the possibility that their relationship with effort is non-linear or context-dependent. In COCOMO-81, feature correlations generally range between -0.15 and +0.66, with data (0.44) and loc (0.66) being the highest. This reinforces the general pattern that software size and data complexity tend to have higher linear correlations, while more abstract or qualitative attributes (e.g., tool usage, experience levels, scheduling constraints) demonstrate weaker associations.

The DESHARNAIS dataset stands out due to several features showing moderate to strong positive correlations with effort. Notably PointsAjust (0.74) and PointsNonAdjust (0.71) show high correlation, suggesting that adjusted and raw function points are highly indicative of effort in this dataset. Length (0.69), Transactions (0.58), and Entities (0.51) further reinforce this observation, indicating that functional decomposition and feature complexity strongly drive effort in this particular dataset. Adjustment (0.46) and ManagerExp (0.16) follow a similar trend, although the correlation is weaker. The only notable negative correlation is with Language (-0.26), which may imply that the choice of programming language influences effort inversely—possibly due to productivity differences across languages.

The overarching insight drawn from Table IV is the prevalence of weak and negative correlations across multiple datasets and feature sets, especially in the COCOMO and CHINA datasets. This pattern strongly suggests that simple linear models may fail to capture the true complexity of relationships in SEE. Specifically features that are traditionally assumed to be strong drivers (e.g., programmer capability, experience, scheduling pressure) show poor linear alignment with effort, challenging their effectiveness in linear regression-based models. Many features likely interact in non-linear, conditional, or hierarchical ways—for example, the impact of complexity may depend on developer experience or tool support, which linear correlations cannot detect. The consistent correlation of size-based metrics (AFP, LOC, PointsAdjust) with effort supports their inclusion, but highlights a potential over-reliance in models that don't incorporate richer, multi-dimensional feature representations.

TABLE IV
FEATURES CORRELATION WITH EFFORT

| CHINA | | COCOMO NASA-V2 | | COCOMO NASA-V1 | | COCOMO-81 | | DESHARNAIS | |
|---|---|---|---|---|---|---|---|---|---|
| AFP | 0.68 | projectname | 0.11 | RELY | -0.13 | dev_mode | -0.13 | TeamExp | 0.12 |
| Input | 0.58 | cat2 | -0.14 | DATA | -0.03 | rely | 0.21 | ManagerExp | 0.16 |
| Output | 0.56 | forg | -0.37 | CPLX | 0.16 | data | 0.44 | Length | 0.69 |
| Enquiry | 0.51 | center | 0.42 | TIME | -0.14 | cplx | 0.01 | Transactions | 0.58 |
| File | 0.61 | mode | -0.33 | STOR | -0.16 | time | 0.15 | Entities | 0.51 |
| Interface | 0.33 | rely | 0.24 | VIRT | -0.12 | stor | 0.10 | PointsNonAdjust | 0.71 |
| Added | 0.69 | data | -0.25 | TURN | -0.18 | virt | 0.02 | Adjustment | 0.46 |
| Changed | 0.11 | cplx | 0.41 | ACAP | -0.05 | turn | 0.21 | PointsAjust | 0.74 |
| Deleted | 0.07 | time | 0.38 | AEXP | -0.18 | acap | -0.15 | Language | -0.26 |
| PDR_AFP | 0.24 | stor | 0.26 | PCAP | 0.10 | aexp | -0.04 | | |
| PDR_UFP | 0.26 | virt | 0.19 | VEXP | 0.00 | pcap | 0.16 | | |
| NPDR_AFP | 0.22 | turn | 0.13 | LEXP | 0.26 | vexp | 0.07 | | |
| NPDU_UFP | 0.24 | acap | -0.21 | MODP | -0.18 | lexp | 0.09 | | |
| Resource | 0.22 | aexp | -0.11 | TOOL | 0.12 | modp | 0.27 | | |
| Dev.Type | NaN | pcap | -0.16 | SCED | 0.03 | tool | 0.00 | | |
| Duration | 0.48 | vexp | -0.23 | LOC | 0.92 | sced | 0.02 | | |
| N_effort | 0.99 | lexp | -0.05 | | | loc | 0.66 | | |
| | | modp | -0.14 | | | | | | |
| | | tool | 0.10 | | | | | | |
| | | sced | -0.31 | | | | | | |
| | | equivphyskloc | 0.59 | | | | | | |

This correlation analysis motivates the need for more sophisticated, non-linear modeling that can capture complex feature interactions and non-additive effects. Additionally, it opens opportunities for multi-view learning, enabling better utilization of both weakly and strongly correlated features across diverse views.

As the SEE datasets are non-linear in nature, the process of transforming the original dataset with a non-linear method such as Isomap will produce a better solution compared with a linear method such as PCA. The dimensionality reduction and transformation process described in the proposed method reflects a critical pre-processing step aimed at enhancing the representational quality of the input data used in the MVNN architecture. Specifically, the procedure involves applying the Isomap algorithm to each fold of the training dataset during CV. Isomap, a well-established non-linear manifold learning technique, is employed here with n_neighbors set to half of the dataset dimension, which defines the local neighbourhood size used to construct the geodesic distance graph—a key component in preserving the intrinsic geometry of the data in the lower-dimensional space. The use of Isomap as a pre-processing transformation in this stage is particularly valuable for revealing latent non-linear structures in the feature space. Beside introducing diversity in input representations, which enhances the learning behaviour of MVNN. By setting n_neighbors to a half the dataset dimension, the Isomap transformation becomes highly sensitive to local curvature and fine-grained manifold structure, potentially capturing non-linear relationships that are otherwise obscured in high-dimensional Euclidean space. This configuration is particularly useful in SEE datasets, where feature interdependencies often exhibit non-linear interactions due to the complex nature of software artifacts, such as code metrics, effort estimations, and process attributes. As a result of this transformation, each fold of the transformed dataset is effectively reduced to half of its original size, not in terms of instances, but in feature dimensionality, making the subsequent learning process more efficient while preserving relevant structural information.

Figure 4 provides a visual TSNE transformed description of the original and Isomap-transformed datasets in each parts of the MVNN. The upper-left quadrant illustrates the post-Isomap transformation output. When compared to the original data—where a clear quadratic trend is observable, allowing for a smooth approximation curve or regression line—the Isomap-transformed data appears spread out, disordered, and random in its geometric structure. This visual manifestation is typical of Isomap in scenarios where local neighbourhood preservation dominates over global structure, especially with a low n_neighbors parameter. This seemingly erratic pattern does not imply a degradation in data quality but instead reflects a reprojection of the data manifold into a space where latent non-linear relationships are more linearly separable. In essence, Isomap flattens the manifold, unfolding hidden non-linearities that are not linearly apparent in the original high-dimensional feature space.

Following transformation, both the original dataset and the Isomap transformed version are independently fed into their respective meta-models within the MVNN architecture. These meta-models, implemented as parallel NN, learn distinct representations from their input view. Notably, because of the stark difference in feature space geometry, the activation patterns and learned representations within each meta-model diverge significantly. Consistent with the visual outputs in Figure 4, the internal activations from both meta-models are characterized by random-like scatter patterns, reflecting the absence of obvious global trends in the transformed feature spaces. This also indicate that there is no dominant view which could affect the final prediction. Despite this initial irregularity, the concatenation layer, which aggregates the outputs from both views, also exhibits a composite randomness, with no immediate coherent structure emerging at this intermediate level. However, this multimodal noise is not a sign of model failure but rather a reflection of the heterogeneity and complementary nature of the multi-view inputs.

The most critical insight emerges at the final SSE regressor layer of the MVNN. It is at this terminal stage that a clear, structured pattern becomes visible. Despite the randomness seen in earlier stages, the final layer manages to synthesize the multi-view representations into a coherent mapping that aligns with the underlying regression task. A distinct solid line emerges in the output, indicative of a strong predictive signal and successful integration of both views. This progression—from scattered and disjointed feature spaces to a unified and interpretable output—is a hallmark of deep NN architectures capable of abstract feature fusion and non-linear regression modelling. This behaviour illustrates one of the primary strengths of MVNN, which is their ability to learn hierarchical representations from disparate feature spaces, gradually integrating them through multiple layers of abstraction and non-linear transformations. The earlier layers focus on local view-specific encoding, while the deeper layers are tasked with capturing cross-view synergies, ultimately culminating in a more structured and fine-grained and robust prediction.

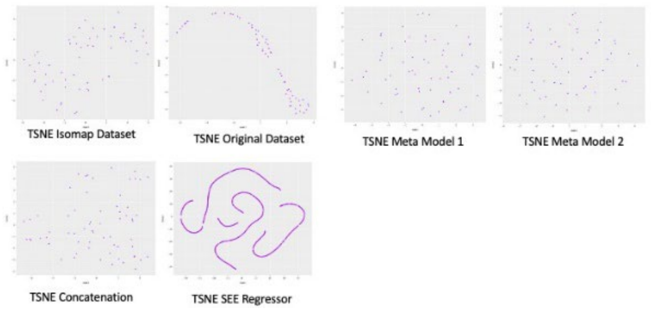

**Figure 4.** Original, Isomap, Meta Model 1 and 2, Concatenation and SSE Regressor Transformation on the dataset

## V. RESULT AND DISCUSSION

In this section, the results of the study are presented and summarizing the performance parameters used such as MAE, MSE, MMRE, RMSE, MdMRE and $R^2$ as shown in Table V.

### A. RESULTS FROM SEVERAL SEE DATASETS

TABLE V
OUR PROPOSED MVNN RESULTS ON VARIOUS SEE DATASETS

| MAE | MSE | MMRE | RMSE | MdRME | $R^2$ |
|---|---|---|---|---|---|
| 2.38E-03 | 2.30E-05 | 3.34E-02 | 4.79E-03 | 3.76E-02 | 0.998 |
| 5.47E-03 | 5.70E-05 | 4.44E-02 | 7.54E-03 | 7.21E-02 | 0.999 |
| 1.25E-02 | 8.31E-04 | 1.66E-01 | 2.88E-02 | 2.47E-01 | 0.956 |
| 9.52E-03 | 1.85E-03 | 1.60E-01 | 4.30E-02 | 3.48E-01 | 0.927 |
| 1.04E-02 | 3.69E-04 | 5.40E-02 | 1.92E-02 | 3.44E-02 | 0.990 |

The results presented a detailed narrative of how the proposed MVNN performs across a variety of widely recognized SEE datasets in the following order: China, COCOMO NASA-V1, COCOMO NASA-V2, COCOMO-81 and Desharnais. Each dataset represents a distinct context with different data characteristics, feature sets, and distributions. Despite this diversity, the MVNN consistently demonstrates strong predictive performance, reinforcing its adaptability and effectiveness in complex, real-world scenarios where SEE remains a challenging task.

Since no universal baseline exists across these datasets—and because each dataset is inherently unique and lacks correlation with the others—the evaluation of model performance focuses primarily on the $R^2$ (coefficient of determination) metric. This measure reflects how much of the variation in the target (effort) can be explained by the model. A high $R^2$ value indicates that the model captures the underlying relationships between features and the target variable very well.

The MVNN achieves an $R^2$ of 0.998 on the CHINA dataset, suggesting that nearly all of the variability in software effort can be explained by the model's predictions. The error values are remarkably low—MAE of 0.00238 and RMSE of 0.00479—indicating both accuracy and consistency. The MMRE, which stands at just 3.34%, further supports this conclusion. In practice, this means that the MVNN effectively learns from different feature views and integrates them to generate accurate effort predictions.

Similarly, for the COCOMO NASA-V1 dataset, the model achieves an exceptionally high $R^2$ of 0.999, indicating nearly perfect alignment between predicted and actual effort values. The error rates are similarly low, with an MAE of 0.00547 and MMRE of 4.44%, showing that the MVNN captures both large-scale and nuanced relationships within the dataset.

The COCOMO NASA-V2 dataset yields a lower $R^2$ value of 0.956, but this still indicates that more than 95.6% of the variance in the target variable is explained by the MVNN. The model's performance in terms of error—MAE of 0.0125, RMSE of 0.0288, and MMRE of 16.6%—is slightly worse than in the previous datasets, which may be due to the noisier nature of the dataset or missing contextual features that are not captured. Nevertheless, the performance is still excellent by regression standards, especially in the domain of SEE, where achieving high $R^2$ values is notoriously difficult due to the uncertainty and inconsistency in human estimation and project documentation.

On the older and perhaps less structured COCOMO-81 dataset, the model achieves an $R^2$ of 0.927, the lowest performance compared with other datasets, but it still manages to capture over 92% of the effort variance. The result is still impressive given the dataset's age and potential inconsistencies in its feature definitions. The MMRE of 16% and MdRME of 34.8% suggest that prediction accuracy varies more in this dataset—possibly because the influence of human and organizational factors in historical data is harder to quantify or was less rigorously measured. Even so, the MVNN maintains strong performance by adapting to the available feature views.

Lastly, for the Desharnais dataset, the MVNN achieves an $R^2$ of 0.990, a level of accuracy that again demonstrates its robustness. With an MAE of 0.0104, MMRE of 5.4%, and RMSE of 0.0192, the model shows both precision and consistency. This dataset contains structured features like function points and adjustment factors, which are well-suited to multi-view processing. The MVNN takes advantage of these structured views to accurately map input features to effort values.

Across all datasets, the $R^2$ values range from 0.927 to 0.999, reflecting that the MVNN captures between 92.7% and 99.9% of the variance in software effort. Even in the most challenging case (COCOMO NASA-V2), the unexplained variance is less than 4.4%, which could be attributed to random noise, missing features, or non-observable project management factors. Such high $R^2$ values are rare in SEE tasks and indicate a very strong model fit across diverse datasets.

This outcome supports the key strength of the MVNN with its ability to process and learn from multiple feature representations—or views—of the same data instance. By combining complementary and redundant information from different views, the MVNN forms a richer, more nuanced internal representation of the data, enabling it to model complex, non-linear relationships far better than traditional single-view or linear models.

The performance results demonstrate that the proposed MVNN offers exceptional predictive capability in the SEE domain. Its ability to generalize across varied datasets, capture complex relationships, and maintain low error rates makes it a highly effective solution—particularly in domains where effort is influenced by numerous interdependent and heterogeneous factors. The consistently high $R^2$ values reaffirm that MVNN not only models the data accurately but also understands the underlying structure of software project characteristics, leading to predictions that are both reliable and interpretable.

## B. RESULTS FROM MULTI VIEW AND SINGLE VIEW

The comparative results presented in Table VI offer a detailed performance analysis between the proposed MVNN architecture and two variations of single-view meta-models across five prominent SEE datasets. This comparison aims to empirically validate the impact of incorporating multiple data perspectives (views) during learning, as facilitated by the MVNN architecture, in contrast to learning from a single representation of the data.

From a technical standpoint, the MVNN consistently demonstrates superior predictive performance across most evaluation metrics. This consistent superiority is particularly evident when analyzing the $R^2$ metric, which represents the proportion of variance in the target variable that is predictable from the input features. MVNN achieves $R^2$ values above 0.99 in the China and COCOMO NASA-V1 datasets, 0.956 in COCOMO NASA-V2, 0.927 in COCOMO-81, and 0.990 in Desharnais—each indicative of excellent model fit and high reliability in capturing the underlying functional relationship.

The results highlight that the multi-view architecture of MVNN provides a significant performance boost over both single-view meta-models. For instance, in the COCOMO NASA-V2 dataset, which is considered more complex and noisy, MVNN achieves an $R^2$ of 0.956—whereas the two single-view models achieve considerably lower $R^2$ values of 0.657 and 0.698, respectively. This stark contrast clearly illustrates the advantage of multi-view learning in capturing richer and more abstract representations of the data, especially in complex estimation scenarios where a single perspective may fail to account for latent dependencies or nonlinear interactions.

In terms of MAE and RMSE, the MVNN also maintains lower values across most datasets, indicating reduced average error and deviation. For the China dataset, for example, MVNN produces a MAE of 0.00238 and an RMSE of 0.00479, compared to 0.00263 and 0.00607 in Meta Model 1, and 0.011 and 0.0172 in Meta Model 2. This pattern holds across multiple datasets, suggesting the MVNN's robustness and consistent capability to generalize better, despite the inherent variance and distributional shifts in the dataset characteristics.

A notable observation is the performance on the COCOMO-81 dataset, where Single View Meta Model 2 slightly outperforms MVNN with an $R^2$ of 0.998 versus 0.927 and a lower MAE (0.00405 vs. 0.00952). This exception also apply with Single View Meta Model 1 which also outperforms our proposed MVNN on MdRME using the same dataset. But overall, the multi-view approach embodied in the MVNN framework demonstrates a clear and consistent improvement in SEE when compared to traditional single-view models. This improvement is attributed to the MVNN's ability to integrate diverse feature subspaces into a unified representation, enabling it to model complex, nonlinear, and multidimensional relationships that are often missed in single-view approaches. The comparative results confirm the hypothesis that leveraging multiple views enhances learning capability and prediction fidelity, particularly in real-world SEE problems where data can be sparse, noisy, and high-dimensional.

## C. COMPARISON WITH PREVIOUS SEE METHODS

To validate our proposed method, we verify it with previous SEE studies on the same performance metric and datasets to compare the results in Table VII.

The experimental evaluation of our proposed MVNN demonstrates a consistent and notable improvement across a variety of benchmark datasets compared with LNI-based NN [39], Neuro-fuzzy logic [39], Adaptive GA-based NN [39], GEHO-based NFN [39], FCNN [40], GWO-FC [40] and others.

The comparison reveals that our MVNN achieves lower error rates in most cases. For instance, on the China dataset,

TABLE VI
COMPARISON BETWEEN SINGLE AND MULTI-VIEW

| Model | Dataset | MAE | MSE | MMRE | RMSE | MdRME | R2 |
|---|---|---|---|---|---|---|---|
| MVNN | China | **2.38E-03** | **2.30E-05** | **3.34E-02** | **4.79E-03** | **3.76E-02** | **0.998** |
| | COCOMO NASA-V1 | **5.47E-03** | **5.70E-05** | **4.44E-02** | **7.54E-03** | **7.21E-02** | **0.999** |
| | COCOMO NASA-V2 | **1.25E-02** | **8.31E-04** | **1.66E-01** | **2.88E-02** | 2.47E-01 | **0.956** |
| | COCOMO-81 | 9.52E-03 | 1.85E-03 | 1.60E-01 | 4.30E-02 | 3.48E-01 | 0.927 |
| | Desharnais | **1.04E-02** | **3.69E-04** | **5.40E-02** | **1.92E-02** | **3.44E-02** | **0.990** |
| Single View Meta Model 1 | China | 2.63E-03 | 3.70E-05 | 3.68E-02 | 6.07E-03 | 4.56E-02 | 0.997 |
| | COCOMO NASA-V1 | 6.54E-03 | 1.02E-04 | 5.31E-02 | 1.01E-02 | 1.07E-01 | 0.997 |
| | COCOMO NASA-V2 | 1.84E-02 | 6.51E-03 | 2.45E-01 | 8.07E-02 | **2.05E-01** | 0.657 |
| | COCOMO-81 | 5.92E-03 | 2.19E-04 | 9.96E-02 | 1.48E-02 | **2.83E-01** | 0.991 |
| | Desharnais | 1.20E-02 | 6.83E-04 | 6.23E-02 | 2.61E-02 | 3.83E-02 | 0.981 |
| Single View Meta Model 2 | China -CLEANED | 1.10E-02 | 2.96E-04 | 1.54E-01 | 1.72E-02 | 2.38E-01 | 0.979 |
| | COCOMO NASA-V1 | 2.60E-02 | 7.92E-03 | 2.11E-01 | 8.90E-02 | 1.86E-01 | 0.805 |
| | COCOMO NASA-V2 | 2.68E-02 | 5.73E-03 | 3.57E-01 | 7.57E-02 | 3.44E-01 | 0.698 |
| | COCOMO-81 | **4.05E-03** | **5.80E-05** | **6.81E-02** | **7.63E-03** | 3.36E-01 | **0.998** |
| | Desharnais | 2.55E-02 | 2.34E-03 | 1.33E-01 | 4.84E-02 | 1.09E-01 | 0.933 |

TABLE VII
PERFORMANCE COMPARISON

| Method | Metrics | MAE | MSE | MMRE | RMSE | MdMRE |
|---|---|---|---|---|---|---|
| LNI-based NN [39] | China | - | - | 2.40E-01 | 1.48E-01 | 2.55E-01 |
| | COCOMO NASA-V1 | - | - | 2.43E-01 | 1.83E-01 | 2.49E-01 |
| | COCOMO NASA-V2 | - | - | 2.25E-01 | 3.83E-01 | 2.49E-01 |
| | COCOMO-81 | - | - | 2.24E-01 | 2.61E-01 | 2.56E-01 |
| | Desharnais | - | - | 3.20E-01 | 3.12E-01 | 3.36E-01 |
| Neuro-fuzzy logic [39] | China | - | - | 2.20E-01 | 7.50E-02 | 2.40E-01 |
| | COCOMO NASA-V1 | - | - | 2.36E-01 | 1.31E-01 | 2.15E-01 |
| | COCOMO NASA-V2 | - | - | 1.96E-01 | 2.90E-01 | 2.15E-01 |
| | COCOMO-81 | - | - | 2.13E-01 | 1.78E-01 | 2.56E-01 |
| | Desharnais | - | - | 2.96E-01 | 1.73E-01 | 2.23E-01 |
| Adaptive GA-based NN [39] | China | - | - | 1.92E-01 | 5.60E-02 | 2.18E-01 |
| | COCOMO NASA-V1 | - | - | 2.31E-01 | 6.50E-02 | 1.72E-01 |
| | COCOMO NASA-V2 | - | - | 1.74E-01 | 2.32E-01 | 1.72E-01 |
| | COCOMO-81 | - | - | 1.99E-01 | 1.30E-01 | 2.35E-01 |
| | Desharnais | - | - | 1.97E-01 | 1.11E-01 | 1.81E-01 |
| GEHO-based NFN [39] | China | - | - | 1.67E-01 | 3.90E-01 | 1.68E-01 |
| | COCOMO NASA-V1 | - | - | 2.20E-01 | 6.00E-02 | 1.30E-01 |
| | COCOMO NASA-V2 | - | - | **1.28E-01** | 9.60E-01 | **1.30E-01** |
| | COCOMO-81 | - | - | 1.74E-01 | 5.50E-02 | 2.23E-01 |
| | Desharnais | - | - | 1.12E-01 | 6.00E-02 | 1.00E-01 |
| FCNN [40] | China | 3.10E-02 | 2.88E-03 | - | 5.36E-02 | 5.71E-01 |
| | COCOMO NASA-V1 | 1.06E-01 | 2.53E-02 | - | 1.59E-01 | 7.87E-01 |
| | COCOMO NASA-V2 | 9.67E-02 | 2.10E-02 | - | 1.42E-01 | 8.25E-01 |
| | COCOMO-81 | 1.52E-01 | 3.19E-02 | - | 1.79E-01 | 1.73E-01 |
| | Desharnais | 1.27E-01 | 3.47E-02 | - | 1.86E-01 | 4.39E-01 |
| GWO-FC [40] | China | 2.18E-02 | 1.34E-03 | - | 3.66E-02 | 3.82E-01 |
| | COCOMO NASA-V1 | **4.80E-03** | **4.58E-05** | - | **6.80E-03** | **6.30E-03** |
| | COCOMO NASA-V2 | 6.53E-02 | 1.17E-02 | - | 1.08E-01 | 6.31E-01 |
| | COCOMO-81 | 1.30E-02 | **2.82E-04** | - | **1.68E-02** | **1.31E-02** |
| | Desharnais | 3.21E-02 | 1.85E-03 | - | 4.30E-02 | 8.43E-02 |
| KNN [42] | China | 3.32E-02 | 6.30E-03 | | 7.92E-02 | 3.48E-01 |
| | COCOMO NASA-V1 | 7.80E-02 | 1.16E-02 | | 1.08E-01 | 1.98E-01 |
| | COCOMO NASA-V2 | 1.10E-01 | 4.38E-02 | | 2.09E-01 | 3.37E-01 |
| | COCOMO-81 | 1.54E-02 | 9.00E-04 | | 3.00E-02 | 4.27E-01 |
| | Desharnais | 9.94E-02 | 2.84E-02 | | 1.69E-01 | 2.10E-01 |
| LR [42] | China | 4.33E-02 | 9.90E-03 | | 9.95E-02 | 4.75E-01 |
| | COCOMO NASA-V1 | 1.68E-02 | 4.00E-04 | | 2.00E-02 | 1.01E-01 |
| | COCOMO NASA-V2 | 1.11E-01 | 3.38E-02 | | 1.84E-01 | 6.11E-01 |
| | COCOMO-81 | 1.00E-01 | 1.90E-02 | | 1.38E-01 | 4.69E+00 |
| | Desharnais | 8.96E-02 | 1.38E-02 | | 1.17E-01 | 3.10E-01 |
| NB [42] | China | 4.15E-02 | 6.80E-03 | | 8.26E-02 | 4.55E-01 |
| | COCOMO NASA-V1 | 1.64E-02 | 4.00E-04 | | 1.92E-02 | 1.06E-01 |
| | COCOMO NASA-V2 | 1.05E-01 | 3.34E-02 | | 1.83E-01 | 6.63E-01 |
| | COCOMO-81 | 3.26E-02 | 2.10E-03 | | 4.53E-02 | 3.19E+00 |
| | Desharnais | 8.78E-02 | 1.28E-02 | | 1.13E-01 | 3.14E-01 |
| SBG [42] | China | 4.36E-02 | 8.50E-03 | | 9.23E-02 | 5.50E-01 |
| | COCOMO NASA-V1 | 1.42E-01 | 3.14E-02 | | 1.77E-01 | 3.38E-01 |
| | COCOMO NASA-V2 | 8.92E-02 | 3.56E-02 | | 1.89E-01 | 6.89E-01 |
| | COCOMO-81 | 4.87E-02 | 5.10E-03 | | 7.13E-02 | 3.91E+00 |
| | Desharnais | 1.12E-01 | 2.27E-02 | | 1.51E-01 | 2.99E-01 |
| Our | China | **2.38E-03** | **2.30E-05** | **3.34E-02** | **4.79E-03** | **3.76E-02** |
| | COCOMO NASA-V1 | 5.47E-03 | 5.70E-05 | **4.44E-02** | 7.54E-03 | 7.21E-02 |
| | COCOMO NASA-V2 | **1.25E-02** | **8.31E-04** | **1.66E-01** | **2.88E-02** | 2.47E-01 |
| | COCOMO-81 | **9.52E-03** | 1.85E-03 | **1.60E-01** | 4.30E-02 | 3.48E-01 |
| | Desharnais | **1.04E-02** | **3.69E-04** | **5.40E-02** | **1.92E-02** | **3.44E-02** |

our MVNN achieved an exceptionally low MAE of 0.00238, outperforming other models such as the FCNN (MAE 0.0310) and KNN (MAE 0.0332). Similarly, on the COCOMO NASA-V1 dataset, our model obtained an MAE of 0.00547, which is significantly better than traditional learning approaches such as Linear Regression (MAE 0.0168) and Naive Bayes (MAE 0.0164). Even on more challenging datasets like COCOMO NASA-V2 and COCOMO-81, which tend to produce relatively higher error margins, our MVNN still performs competitively. On COCOMO NASA-V2, our model achieved an MAE of 0.0125, compared to FCNN with an MAE of 0.0967. Likewise, in the Desharnais dataset, where many classical methods show significant errors and variability, our model reached an MAE of 0.0104.

These results demonstrate the robustness and generalizability of our approach across datasets of varying characteristics. Unlike many baseline models that either rely on linear assumptions or are limited by single-view input representations, our MVNN benefits from its architecture that integrates multiple views through a structured DL pipeline. This allows it to adapt to different data distributions and capture complex nonlinear relationships that are often present in software effort estimation tasks.

### *D. THREATS TO VALIDITY*

As with every empirical experiment, the results of our works are subject to some threats to validity.

### *E. CONSTRUCT VALIDITY*

We admit that during our experiments, only a subset of various SEE datasets were used and not all datasets were included in the case of PROMISE. Although it would be best to include all of them, but the limitation of resources hinders us to take this step. For objectivity, we reserved ourself from modifying unless it is necessary to conduct the experiment. Since most studies on SEE uses an open and public datasets, we consider the datasets is complete and adequately fixed and reliable to be used in our study.

### *F. INTERNAL VALIDITY*

Although there are variations of the same dataset in some repositories. We found it to be constructive and the necessary adjustment have been made and verified by previous studies. Therefore, the validity of the datasets should be minor and will cause little effect on the results.

### *G. EXTERNAL VALIDITY*

We validated our findings using open and public datasets from different sources and different software metrics to gain more confidence in the external validity of our study. By doing so, we hope to achieve generalization with our proposed method, and any replicated studies with our method will be a step to improve our method.

## VI. CONCLUSION

In this article, we propose MVNN for SEE which shows to be reliable compared with previous studies. The challenges of SEE lies in the small amount of examples and the different software metrics used universally among software projects. Although there are methods and techniques to overcome the challenges of SEE, but the complex nature of software projects still prove to be a challenging field in the future to improve software engineering, by finding an efficient tools to find and predict defect during the life cycle of software development. In this method, we used multiple steps of pre-processing prior of training ranging from features selection, imputation, and scaling to overcome the different software metrics, and to avoid any dominant value in the outlier. Besides the original dataset as the primary view, we opted to create a different view from the dataset by utilizing Isomap reliable dimensionality reduction algorithm. The use of Isomap as a dimensionality reduction will reduce the size of the MVNN input, so a smaller yet effective NN regressor can be trained using K-fold CV. Empirical studies with some notably SEE datasets show the effectiveness of our proposed method compared with previous methods. In the future, we would like to extend our research towards more deep layers NN to further improve the performance of our proposed MVNN.

## AUTHORS CONTRIBUTION

**Boy Setiawan:** Conceptualization, Methodology, Research, Investigation, Formal Analysis, Resources, Software, Visualization, Original Draft Writing.
**Agus Subekti:** Supervision, Validation, Original Draft Writing Preparation, Review Writing & Editing.

## COPYRIGHT

## REFERENCES

[1] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-03629-8.
[2] M. Rahman, P. P. Roy, M. Ali, T. Gonc¸alves, and H. Sarwar, “Software Effort Estimation using Machine Learning Technique,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 4, 2023, doi: 10.14569/IJACSA.2023.0140491.
[3] S. Hameed, Y. Elsheikh, and M. Azzeh, “An optimized case-based software project effort estimation using genetic algorithm,” *Inf. Softw. Technol.*, vol. 153, p. 107088, Jan. 2023, doi: 10.1016/j.infsof.2022.107088.
[4] Moatasem. M. Draz, O. Emam, and Safaa. M. Azzam, “Software cost estimation predication using a convolutional neural network and particle swarm optimization algorithm,” *Sci. Rep.*, vol. 14, no. 1, p. 13129, Jun. 2024, doi: 10.1038/s41598-024-63025-8.
[5] M. S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, and W. Abdul, “Metaheuristic Algorithms in Optimizing Deep Neural Network Model for Software Effort Estimation,” *IEEE Access*, vol. 9, pp. 60309–60327, 2021, doi: 10.1109/ACCESS.2021.3072380.
[6] P. V. A G, A. K. K, and V. Varadarajan, “Estimating Software Development Efforts Using a Random Forest-Based Stacked Ensemble Approach,” *Electronics*, vol. 10, no. 10, p. 1195, May 2021, doi: 10.3390/electronics10101195.
[7] J. Rashid, S. Kanwal, M. Wasif Nisar, J. Kim, and A. Hussain, “An Artificial Neural Network-Based Model for Effective Software Development Effort Estimation,” *Comput. Syst. Sci. Eng.*, vol. 44, no. 2, pp. 1309–1324, 2023, doi: 10.32604/csse.2023.026018.
[8] N. Rankovic, D. Rankovic, M. Ivanovic, and L. Lazic, “A New Approach to Software Effort Estimation Using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays,” *IEEE Access*, vol. 9, pp. 26926–26936, 2021, doi: 10.1109/ACCESS.2021.3057807.
[9] M. F. Bosu, S. G. MacDonell, and P. A. Whigham, “Analyzing the Stationarity Process in Software Effort Estimation Datasets,” 2021, doi: 10.48550/ARXIV.2107.01616.

[10] M. Rahman, T. Goncalves, and H. Sarwar, "Review of Existing Datasets Used for Software Effort Estimation," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 7, 2023, doi: 10.14569/IJACSA.2023.01407100.

[11] M. Azzeh, "Dataset Quality Assessment: An extension for analogy based effort estimation," 2017, *arXiv*. doi: 10.48550/ARXIV.1703.04575.

[12] S. S. Gautam and V. Singh, "Adaptive Discretization Using Golden Section to Aid Outlier Detection for Software Development Effort Estimation," *IEEE Access*, vol. 10, pp. 90369–90387, 2022, doi: 10.1109/ACCESS.2022.3200149.

[13] M. F. Bosu and S. G. MacDonell, "Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation," 2020, doi: 10.48550/ARXIV.2012.10836.

[14] E. O. Kiyak, D. Birant, and K. U. Birant, "Multi-view learning for software defect prediction," *E-Inform. Softw. Eng. J.*, vol. 15, no. 1, pp. 163–184, Nov. 2021, doi: 10.37190/e-Inf210108.

[15] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Inf. Fusion*, vol. 38, pp. 43–54, Nov. 2017, doi: 10.1016/j.inffus.2017.02.007.

[16] X. Xie and S. Sun, "Multi-view twin support vector machines," *Intell Data Anal*, vol. 19, no. 4, pp. 701–712, Jul. 2015, doi: 10.3233/IDA-150740.

[17] Y. Makihara, A. Mansur, D. Muramatsu, M. Uddin, and Y. Yagi, "Multi-view discriminant analysis with tensor representation and its application to cross-view gait recognition," Jul. 2015, doi: 10.1109/FG.2015.7163131.

[18] X. Yan, S. Hu, Y. Mao, Y. Ye, and H. Yu, "Deep multi-view learning methods: A review," *Neurocomputing*, vol. 448, pp. 106–129, Aug. 2021, doi: 10.1016/j.neucom.2021.03.090.

[19] T. Baltrusaitis, C. Ahuja, and L.-P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," *IEEE Trans Pattern Anal Mach Intell*, vol. 41, no. 2, pp. 423–443, Feb. 2019, doi: 10.1109/TPAMI.2018.2798607.

[20] S. K. D'mello and J. Kory, "A Review and Meta-Analysis of Multimodal Affect Detection Systems," *ACM Comput Surv*, vol. 47, no. 3, Feb. 2015, doi: 10.1145/2682899.

[21] D. Wang, P. Cui, M. Ou, and W. Zhu, "Deep multimodal hashing with orthogonal regularization.," in *IJCAI*, 2015, pp. 2291–2297.

[22] A. Jadhav and S. Kumar Shandilya, "Towards effective feature selection in estimating software effort using machine learning," *J. Softw. Evol. Process*, vol. 36, no. 5, p. e2588, May 2024, doi: 10.1002/smr.2588.

[23] E. Hadjisolomou, K. Stefanidis, H. Herodotou, M. Michaelides, G. Papatheodorou, and E. Papastergiadou, "Modelling Freshwater Eutrophication with Limited Limnological Data Using Artificial Neural Networks," *Water*, vol. 13, no. 11, p. 1590, Jun. 2021, doi: 10.3390/w13111590.

[24] Okfalisa, I. Gazalba, Mustakim, and N. G. I. Reza, "Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification," in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta: IEEE, Nov. 2017, pp. 294–298. doi: 10.1109/ICITISEE.2017.8285514.

[25] J. Li, "Asymptotics of K-Fold Cross Validation," *J. Artif. Intell. Res.*, vol. 78, pp. 491–526, Nov. 2023, doi: 10.1613/jair.1.13974.

[26] L. B. V. de Amorim, G. D. C. Cavalcanti, and R. M. O. Cruz, "The choice of scaling technique matters for classification performance," *Appl. Soft Comput.*, vol. 133, p. 109924, 2023, doi: https://doi.org/10.1016/j.asoc.2022.109924.

[27] F. J. O. Gómez, G. O. López, E. Filatovas, O. Kurasova, and G. E. M. Garzón, "Hyperspectral Image Classification Using Isomap with SMACOF," *Informatica*, vol. 30, no. 2, pp. 349–365, 2019, doi: 10.15388/Informatica.2019.209.

[28] J. B. Tenenbaum, V. D. Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000, doi: 10.1126/science.290.5500.2319.

[29] T. Qu and Z. Cai, "An improved Isomap method for manifold learning," *Int. J. Intell. Comput. Cybern.*, vol. 10, pp. 30–40, Mar. 2017, doi: 10.1108/IJICC-03-2016-0014.

[30] M. Yousaf, T. U. Rehman, and L. Jing, "An Extended Isomap Approach for Nonlinear Dimension Reduction," *SN Comput. Sci.*, vol. 1, no. 3, p. 160, May 2020, doi: 10.1007/s42979-020-00179-y.

[31] M. Azzeh and Y. Elsheikh, "Learning best K analogies from data distribution for case-based software effort estimation," 2017, *arXiv*. doi: 10.48550/ARXIV.1703.04567.

[32] A. Ardiansyah, M. M. Mardhia, and S. Handayaningsih, "Analogy-based model for software project effort estimation," *Int. J. Adv. Intell. Inform.*, vol. 4, no. 3, p. 251, Nov. 2018, doi: 10.26555/ijain.v4i3.266.

[33] I. Thamarai and S. Murugavalli, "An Evolutionary Computation Approach for Project Selection in Analogy based Software Effort Estimation," *Indian J. Sci. Technol.*, vol. 9, no. 21, Jun. 2016, doi: 10.17485/ijst/2016/v9i21/95286.

[34] A. G. Priya Varshini, K. Anitha Kumari, D. Janani, and S. Soundariya, "Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation," *J. Phys. Conf. Ser.*, vol. 1767, no. 1, p. 012019, Feb. 2021, doi: 10.1088/1742-6596/1767/1/012019.

[35] O. H. Alhazmi and M. Z. Khan, "Software Effort Prediction Using Ensemble Learning Methods," *J. Softw. Eng. Appl.*, vol. 13, no. 07, pp. 143–160, 2020, doi: 10.4236/jsea.2020.137010.

[36] N. A. Zakaria, A. R. Ismail, A. Y. Ali, N. H. M. Khalid, and N. Z. Abidin, "Software Project Estimation with Machine Learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, 2021, doi: 10.14569/IJACSA.2021.0120685.

[37] A. A. Fadhil, R. G. H. Alsarraj, and A. M. Altaie, "Software Cost Estimation Based on Dolphin Algorithm," *IEEE Access*, vol. 8, pp. 75279–75287, 2020, doi: 10.1109/ACCESS.2020.2988867.

[38] V. Van Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "Toward Improving the Efficiency of Software Development Effort Estimation via Clustering Analysis," *IEEE Access*, vol. 10, pp. 83249–83264, 2022, doi: 10.1109/ACCESS.2022.3185393.

[39] S. Sharma and S. Vijayvargiya, "Modeling of software project effort estimation: a comparative performance evaluation of optimized soft computing-based methods," *Int. J. Inf. Technol.*, vol. 14, no. 5, pp. 2487–2496, Aug. 2022, doi: 10.1007/s41870-022-00962-5.

[40] S. Kassaymeh, M. Alweshah, M. A. Al-Betar, A. I. Hammouri, and M. A. Al-Ma'aitah, "Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques," *Clust. Comput.*, vol. 27, no. 1, pp. 737–760, Feb. 2024, doi: 10.1007/s10586-023-03979-y.

[41] Y. L. Alemu, T. Lahmer, and C. Walther, "Damage Detection with Data-Driven Machine Learning Models on an Experimental Structure," *Eng*, vol. 5, no. 2, pp. 629–656, 2024, doi: 10.3390/eng5020036.

[42] A. Jadhav and S. K. Shandilya, "Reliable machine learning models for estimating effective software development efforts: A comparative analysis," *J. Eng. Res.*, vol. 11, no. 4, pp. 362–376, Dec. 2023, doi: 10.1016/j.jer.2023.100150.