IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 June 2023, accepted 10 June 2023, date of publication 14 June 2023, date of current version 21 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3286372

RESEARCH ARTICLE

Comparing Stacking Ensemble and Deep Learning for Software Project Effort Estimation

HUYNH THAI HOC, RADEK SILHAVY, ZDENKA PROKOPOVA, AND PETR SILHAVY
Faculty of Applied Informatics, Tomas Bata University in Zlín, 76001 Zlín, Czech Republic

Corresponding author: Petr Silhavy (psilhavy@utb.cz)

This work was supported by the Faculty of Applied Informatics, Tomas Bata University in Zlín, under Project RVO/FAI/2021/002 and Project IGA/CebiaTech/2023/004.

ABSTRACT This study focuses on improving the accuracy of effort estimation by employing ensemble, deep learning, and transfer learning techniques. An ensemble approach is utilized, incorporating XGBoost, Random Forest, and Histogram Gradient Boost as generators to enhance predictive capabilities. The performance of the ensemble method is compared against both the deep learning approach and the PFA-IFPUG technique. Statistical criteria including MAE, SA, MMRE, PRED(0.25), MBRE, MIBRE, and relevant information related to MMRE and PRED(0.25) are employed for evaluation. The results demonstrate that combining regression models with Random Forest as the final regressor and XGBoost and Histogram Gradient Boost as prior generators yields more accurate effort estimation than other combinations. Furthermore, the findings highlight the potential of transfer learning in the deep learning method, which exhibits superior performance over the ensemble approach. This approach leverages pre-trained models and continuously improves performance by training on new datasets, providing valuable insights for cross-company and cross-time effort estimation problems. The ISBSG dataset is used to build the pre-trained model, and the inductive transfer learning approach is verified based on the Desharnais, Albrecht, Kitchenham, and China datasets. The study underscores the significance of transfer learning and the integration of domain-specific knowledge from existing models to enhance the performance of new models, thereby improving accuracy, reducing errors, and enhancing predictive capabilities in effort estimation.

INDEX TERMS Software effort estimation, ensemble, function point analysis, deep learning, inductive transfer learning.

I. INTRODUCTION

A successful software project development might begin with extensive analysis and preparation of relevant documents, such as explicitly outlining requirements in the early stages of projects. Technique considerations, environmental factors, and even developer experience are essential in developing a project. These could contribute to measuring effort estimates. However, software development effort estimating (SDEE) may be more challenging and the most challenging task for the project manager [1], [2], [3]. They may leverage previous project experiences or well-known methods when estimating software effort, such as Function Point Analysis (FPA) [4], [5]. They might make their decision based on the researchers' suggestions, which are based on information from sources that have already been used for the project.

FPA is a fundamental approach for estimating the size of software projects from the viewpoint of the user [4], [6]. In 1979, Allan J. Albrecht invented this approach at IBM. The International Function Point Users Group subsequently expanded it (IFPUG). Previous publications [7], [8] indicate that FPA predicts the cost of software development or maintenance regardless of the technology used. The functional size, for instance, should be consistent across domains, languages, and development approaches.

Nevertheless, this is the most difficult step for software engineering estimators. One possible reason is that there are

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Elish.

so many different project lifecycle models, and each one might have different resource needs at different stages of the project's development. The typical estimating technique [7] needs greater effort to record actions, which increases the complexity and time of the estimation. Moreover, the expertise of software engineers, the software team's project history in the same business area, and a range of other features, as well as the correlations between these aspects, are not always accounted for appropriately [9].

There might be two popular kinds of approaches to propose the predictive models, including regression-based approaches [4], [10], [11], [12], [13], machine learning-based or deep learning-based approaches [4], [5], [14], [15]. Moreover, solo vs ensemble techniques is the concepts mentioned in the publication [16]. The term "solo" might be driven by [17], which means "alone; without other people." opposite of solo is ensemble. As presented in [16], the proposal models are based on a single approach, such as a regression model or a machine learning model called solo techniques. Ensemble methodology aims to combine multiple models to create a good prediction model [18].

This article aims to suggest a way to improve SDEE by stacking ensemble techniques that use Random Forest, XGBoost, and Histogram Gradient Boost as generators. Furthermore, there might not be any research conducted to investigate the feasibility of using trained models in estimating effort for the recent dataset. The trained models are built using the oldest dataset. This paper studies how well the models already trained might be used to estimate the effort for the most recent dataset.

The following sections are organized: Section II presents the Function Point Analysis; Section III presents Related Work; Section IV proposes Research Questions; Sections V, and VI present Experimental Design and Experimental Design; Sections VII, and VIII illustrate the Result and Discussion and Threats to Validity; and Section IX conveys Conclusion - Future Work.

II. FUNCTION POINT ANALYSIS

In this study, we employ the FPA-IFPUG approach [4], [7], [19] which is widely used for quantifying the software's complexity and feature set concerning actual user requirements. This technique aims to use a number of unique transaction function types (External Inputs (EI), External Outputs (EO), External Inquiry (EQ)) and data function types (External Interface Files (EIF), Internal Logical Files (ILF)) created by software development projects to measure a size characteristic.

Table 1 shows the relative complexity of the various parts. According to the Counting Practices Manual [7], which is responsible for drafting and modifying its standards, version 4.3.1 (2010), ISO/IEC 20926:2010 standardizes the FPA created by the IFP, also known as the initial function point analysis. Additional ISO/IEC Functional Software Measurements (FSM) include MarkII, MESMA, COSMIC, and FISMA. These methods are outside the scope of this research, however they are referenced in [22].

TABLE 1. Complexity weights of FPA components [20], [21].

Size Attributes	Complexity Weight (CWs)		
	Low	Medium	Large
EI	3	4	6
EO	4	5	7
EQ	3	4	6
EIF	5	7	10
ILF	7	10	15

TABLE 2. General systems characteristics [20], [21].

GSC Factors	Characteristic	Description
F1	Data communications	Does the system require backup and recovery?
F2	Distributed Functions	Are Data Required for Communication?
F3	Performance	Does the system include a distributed processing function?
F4	Heavily Used Configuration	Is critical performance required?
F5	Transaction Rate	Will the system work during heavy loads?
F6	Online Data Entry	Does the system require direct data input?
F7	End-User Efficiency	Do data inputs require multiple screens or operations?
F8	Online Update	Are the main files up to date?
F9	Complex Processing	Are inputs, outputs, files, and queries intricate?
F10	Reusability	Is internal processing complicated and complex?
F11	Installation Ease	Is the code designed for reuse?
F12	Operational Ease	Are Conversions and installation Included in Design?
F13	Multiple Sites	Is the system designed for multiple installations in different locations?
F14	Facilitate Change	Is the application designed to make it easy for users to make changes?

The FPA has many of the characteristics necessary to provide preliminary estimates for software development projects [23]. To begin, it is possible to completely assign function points if doing so satisfies the needs or design criteria. The efforts look to be just getting started. Second, they are unrelated to any kind of data processing, including programming languages, specialist development tools, and so on citation [21]. In addition, the function points may be easier to understand for non-technical users of the program since they are based on the user's external view of the system.

To count function points, a linear combination of size characteristics with appropriate weights for three levels of complexity is built. The function count is often referred to as Unadjusted Function Points (UFP). The UFP formula is shown as Equation (1).

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} BCs_{ij} \times CWs_{ij} \tag{1}$$

where BCs_{ij} is the number of component i at level j, and CWs_{ij} is an appropriate weight from Table 1.

Multiplying the UFP by the adjustment influent factors determines the output of the function point count (GSC). These may assist in a more accurate UFP count [24]. Moreover, the formula (2) defines Value

TABLE 3. Influent factors rating [20], [21].

Influence	Rating
None	0
Insignificant	1
Moderate	2
Average	3
Significant	4
Strong significant	5

Adjustment Factor (VAF):

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{14} F_i \times Rating_{Influence} \quad (2)$$

where F represents the impact of the GSC component, and the system is affected by fourteen distinct factors. These considerations are shown in Table 2, while the ranking of influential considerations is shown in Table 3.

The Adjusted Function Points (AFP) may be calculated using the following formula (3):

$$AFP = UFP \times VAF \quad (3)$$

AFP may be utilized as a factor in the effort estimation process. Meanwhile, the amount of work will be proportional to the Productivity (PDR) divided by the Average Full-Time Productive Capacity. As demonstrated by Equation (4), the amount of effort used will equal the amount of PDR included in the AFP. However, VAF is not recorded for the majority of recently counted projects in ISBSG [25], and the VAF is assumed to be 1. It means that AFP and UFP might be used interchangeably for recent projects.

$$Effort = AFP \times PDR \quad (4)$$

III. RELATED WORK

The previous studies examined various problems with the FPA methodology. Many different kinds of studies have been done to find out how accurate effort estimation models are. Researchers and experts figure out which estimation methods give reliable results for certain data sets and other factors.

Hoc et al. [4] examined the performance of Pytorch-based deep learning, multilayer perceptron (MLP) and multiple linear regression (MLR) in terms of function point analysis-based software effort estimates. In this research, the relative size, type of business, adjusted function points, and EI, EO, EQ, EIF, ILF are all studied based on the ISBSG dataset (version 2020/R1). The effort-estimating performance of multiple models is evaluated using a prediction level of 0.3 (PRED(0.3)) and standardized accuracy (SA). The findings demonstrate that deep learning based on Pytorch and MLP performed better than MLR. Moreover, deep learning outperformed MLP. Furthermore, the authors concluded that EI, EO, EQ, EIF, ILF, and the industry sector contribute the most to the model's accuracy when compared to other factors.

In 2021, Hoc et al. [26] developed the Adj-Effort method to optimize effort estimates in terms of FPA on the basis of the ISBSG version 2020. In order to get the most accurate estimate possible, they used a method called MLR based on AdamOptimizer [10] with 10-fold cross-validation. PRED(0.25), Mean Absolute Error (MAE), and Maximum RMSE were used to evaluate their results against those of the baseline models (Casper-Jones, and FPA-IFPUG). Consequently, their model outperformed the benchmark models.

Using FPA, categorical variable segmentation (CVS), and stepwise regression, Silhavy et al. [27] created a new way to measure the amount of work that goes into making software. The approach of stepwise regression develops an estimation model for each segment. Observational studies relied on data from the ISBSG dataset (Release 13, 2015). The proposed model increases prediction performance in terms of Mean Absolute Percentage Error, Mean Estimation Error, and PRED compared with baseline approaches such as non-clustered FPA and clustering-based models (0.25). In terms of accuracy, the new CVS model exceeds previous methods.

Prokopova et al. [28] analyzed the effect of a few factors on the assessment of labor effort using function point techniques. In this analysis, several things were taken into account, such as the function point count method, where the businesses were located, what they did, and how big they were. Their goal was to find out if the productivity from the training dataset could be used to estimate the amount of work and if the parameters used affected how well the estimates worked. The ISBSG repository (Release 13) is made up of 1,333 finished projects that were put together for historical reasons. Using the hold-out method and a 2:1 ratio, the dataset was split into a training dataset and a testing dataset.

On the other hand, Ahmad and Ibrahim [29] studied and compared support vector machines, random forest, Lasso, Neuralnet, decision tree, ridge, ElasticNet, and Deepnet using the Kitchenham, China, Maxwell, Albrecht, Kemerer, Desharnais, and Cocomo81 datasets to improve SDEE. According to a study of multiple machine learning algorithms, the random forest approach outperforms all other measures (MAE, RMSE) in the Albrecht data source. On the other hand, the Lasso technique produced better results based on MAE in the Kitchenham dataset. The Neuralnet performed better because it produced lower RMSE and greater R-Squared values; in Cocomo81, where the Lasso algorithms outperformed others in the China dataset, it produced lower RMSE, MAE, and higher R-Squared values.

In 2019, Abdelali et al. [30] conducted empirical research aimed at estimating effort using the random forest technique. The impact of the number of trees and the number of qualities used to grow them was initially investigated. Next, the researchers compared the performance of the random forest model with that of the regression tree model using the hold-out validation method (70-30) and three datasets from COCOMO, ISBSG, and Tukutuku. The evaluation criteria employed for assessment were MMRE, MdMRE, and Pred, with a threshold of 0.25. Across all evaluation criteria, the random forest model outperformed the regression tree-based model, particularly in the case of COCOMO and ISBSG datasets.

Passakorn [31] studied whether machine-learning techniques that have excelled in recent active data science competitions will also do well when estimating software effort. Based on 13 industry-standard software effort estimation datasets, he examined 14 machine learning techniques, including the increasingly popular gradient boosting machine and deep learning (PROMISE 2015). The most widely used stable ranking evaluation method for estimating software effort was used to come to the main conclusions of this study. Combining multiple effort estimators into a stacked ensemble, for example, to just take the average of the predicted effort levels, gave more accurate results than any of the 14 estimators that were looked at. In this investigation, the estimated effort values were taken as an average of the values from the stacked ensemble that was most accurate overall. He also found that using the boosting principle to create an ensemble improved performance.

Palaniswamy and Venkatesan [32] employed the ensemble technique to improve prediction. Traditionally, hyperparameters are found through trial and error based on the problem and dataset, which takes a lot of time. Particle swarm optimization (PSO) and genetic algorithms were used to change the hyperparameters in this study. The stacking ensemble model was made with data from the ISBSG dataset. It collects data from different software projects in different countries and companies. In 2021, Anitha et al. [33] investigated software effort estimation using ensemble techniques and machine/deep-learning algorithms. They compared different ensemble techniques and evaluated several stacking models. The authors conducted experiments on multiple datasets, including Albrecht, China, Desharnais, Kemerer, Maxwell, Kitchenham, and Cocomo81, to evaluate the performance of the models. Their findings suggest that the proposed random forest stacking method outperforms SVM, decision trees, and neural nets when applied to various datasets.

Moreover, Suresh Kumar et al. [34] proposed a robust approach using a gradient-boosting regressor model. They compare the performance of this model with various other regression models, including stochastic gradient descent, K-nearest neighbour, decision tree, bagging regressor, random forest regressor, Ada-boost regressor, and gradient boosting regressor. They used Cocomo81 and China datasets to evaluate the models. Their findings demonstrated that the gradient-boosting regressor model performs exceptionally well.

Last but not least, Pan and Yang [35] conducted a study on transfer learning, focusing on the relationship between traditional machine learning and different transfer learning settings. They classified transfer learning into three subsetting: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning, based on the variations in situations between the source and target domains and tasks. Inductive transfer learning applies when the target task differs from the source task, regardless of whether the source and target domains are the same. In transductive transfer learning, the source and target tasks remain the same, while the domains differ. Lastly, in unsupervised transfer learning, similar to inductive transfer learning, the target task differs from but shares a relationship with the source task.

In 2015, Kocaguneli et al. [36] studied transfer learning techniques in effort estimation. The authors employed a transfer learning methodology to address both conventional cross-company learning challenges and data set shift challenges. This demonstrates the potential benefits of integrating research techniques from two distinct but interconnected strands within the field of software engineering, namely ''data set shift'' introduced by Turhan [37] and ''cross-company learning'' proposed by Kitchenham et al. [38]. The results of their investigation challenge two prevalent assumptions: firstly, that information obtained from one organization is unsuitable for informing local decisions; and secondly, that historical data within an organization lacks relevance in the present context. The authors highlight the significance of their findings by emphasizing the success of their transfer learning strategies in effectively managing data transitions, underscoring the importance of not disregarding valuable insights derived from past experiences.

IV. RESEARCH QUESTION

Three research questions (RQs) must be answered:

- RQ1: Which of the prior combinations in the ensemble model could contribute to improved performance?
- RQ2: Is the ensemble model estimation more accurate than the deep learning model?
- RQ3: Is it feasible to use the trained model to estimate effort for recent projects? How effective is the trained model compared to one trained based on a new dataset, assuming both models use the same approach?
- RQ4: What approach (EnsEffort or PytEffort) will improve the accuracy of effort estimation? What happens if we employ transfer learning for PytEffort model?

V. RESEARCH METHODOLOGY

Our research methodology is designed to achieve four primary goals. The methodological steps undertaken to achieve these objectives are outlined below.

1) **Develop an Ensemble Approach:** The study aims to create efficient prediction models that combine multiple regression techniques for enhanced predictability over solo models. The process includes:
   - Selecting robust regression techniques, including XGBoost, Random Forest, and HGBoost.
   - Creating an ensemble model that employs a generator from these techniques as the final regressor, with the others serving as prior generators, ensuring impartiality and optimal use of the regression techniques.
   - Experimenting with different arrangements of these techniques, such as using Random Forest as the final regressor and XGBoost and HGBoost as prior predictors.
2) **Employ Trained Models on New Datasets:** Given potential limitations in accessing or training on new

datasets, we propose using the robust performance of trained models to improve effort estimation. This involves:
- Adopting the model trained in phase 1 for use in the second phase.
- Examining two different scenarios based on two subsets of the second dataset.

3) **Compare Ensemble and Deep Learning Approaches:** The study also aims to contrast the performance of the ensemble approach with that of a deep learning method. This is done by:
- Implementing transfer learning in a deep learning model.
- Conducting preliminary assessments to compare the performance of the transfer learning approach with the ensemble method.

4) **Apply Transfer Learning:** The research further explores the potential benefits of leveraging results from previously trained models for improving the performance on new datasets. This includes:
- Applying transfer learning to continue training on a new set of databases.
- Conducting experiments on four different datasets - Desharnais, Albrecht, Kitchenham, and China, which share the same target feature ('effort') with the ISBSG dataset.

5) **Evaluate Performance:** The final step involves assessing the performance of the methods employed, including:
- Comparing the performances in different scenarios.
- Evaluating the accuracy, error reduction, and enhanced predictive capabilities of the methods used.

This research methodology underscores the potential of using ensemble approaches, deep learning, and transfer learning techniques in developing efficient prediction models.

A. ENSEMBLE APPROACH

In practice, aggregating the predictive effort estimation of different predictors (such as regressors or classifiers) might typically achieve better findings than the best individual predictor. In 1990, Hansen et al. [39] claimed that employing an ensemble of neural networks with a majority agreement technique outperforms using a single. The ensemble is a term to indicate a group of predictors. A method that integrates from a group of predictive models is ensemble learning; the ensemble learning algorithm is an ensemble method. According to a publication [18], bagging, boosting, and stacking are the most common ensemble methods. David Wolpert proposed stacking [40] in 1992, taking prior predictions as feed to determine the final prediction (blender/meta learner). The training dataset is separated into two subsets; the first will be used as a training dataset for predictors. The predictions made by those predictors are based on the second subset as inputs (blending training set) to make the blended predictor. This guarantees the forecasts are "clean," as the predictors have never observed these events throughout training.

Based on the conclusion of Anitha et al. [33], we build the proposed models to estimate software effort estimation by employing a staking ensemble approach (EnsEffort). We choose two among the random forest, extreme gradient boosting, and histogram gradient boosting as generators, and the rest is the final regressor.

B. RANDOM FOREST

Random Forest (RF), introduced in 2001 by Breiman [41], is a kind of ensemble of decision trees trained via the bagging method (or sometimes the pasting method). Several poor models are joined to build a superior model. Each tree categorises the attributes of a new entity. The forest chooses the category with the most votes and averages the outputs of the different trees. According to Mustapha et al. [30], it outperformed several other classification models and was also resistant to over-fitting and relatively user-friendly [42].

C. GRADIENT BOOSTING

The main idea behind boosting is to add new models to an existing group in a logical order. Leo Breiman proposed it [41]. Each iteration entails training a new weakly based learner model based on the errors of the entire ensemble previously learned. The first widely used boosting techniques were entirely algorithm-driven. A gradient-descent-based formulation of boosting approaches (gradient boosting machines) was developed to adapt boosting methods with the statistical framework [43], [44], [45]. The learning method fits new models one after the other to measure the response variable more accurately. The basic idea behind this method is to build new base learners most similar to the negative gradient of the loss function, which is related to the whole ensemble. The learning process will lead to sequential error-fitting if the error is the traditional squared error loss.

- Extreme gradient boosting (XGBoost) is a GB ensemble that leverages the second-order derivatives of the loss function to determine the most accurate and efficient base classifier [18], [46], [47]. XGBoost employs second-order gradients, whereas gradient boosting uses gradients to fit a new base classifier.
- Histogram Gradient Boosting (HGBoost), sometimes known as histogram-based gradient boosting, is a boosting ensemble that uses feature histograms to identify the optimal splits quickly and accurately [46], [48]. It is more efficient than gradient boosting regarding processing speed and memory use.

D. DEEP LEARNING APPROACH

Deep learning enables computational models with several processing layers to discover data representations with different degrees of abstraction [49]. These techniques have significantly advanced the state-of-the-art in many fields, such as object detection, segmentation, and classification. Deep learning uncovers detailed structure in enormous

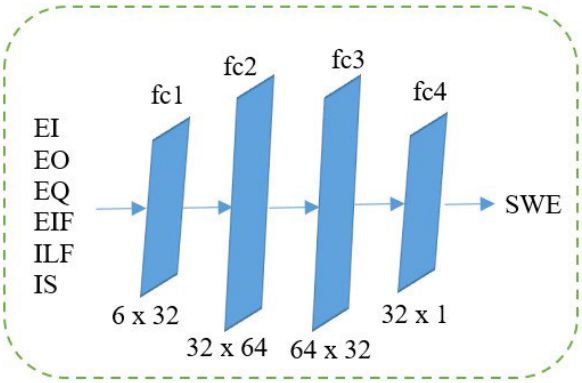

FIGURE 1. The diagram of deep learning with four fully connected layers.

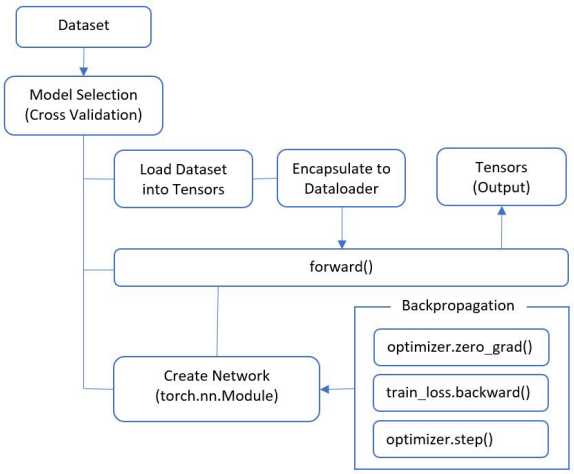

FIGURE 2. The flow diagram of pytorch model.

data sets by utilizing the backpropagation technique to determine how a machine's internal parameters used to calculate each layer's representation from the previous layer's representation should be altered [50].

Figure 1 shows the flow diagram for the deep learning-based proposed model with four fully connected layers. It includes fc1, fc2, fc3, fc4. The first layer (fc1) takes an input tensor of size six and produces an output tensor of size 32. The subsequent layers have increasing output sizes: fc2 outputs 64 features, fc3 outputs 32 features, and fc4 outputs a single scalar value. The forward method specifies the forward pass of the model. It takes an input tensor x and sequentially passes it through the defined layers. Each hidden layer contains a nonlinear activation function applied to the previous layer's output. During the training phase, the weights assigned to each neuron connection are adjusted based on the difference between the predicted and actual outputs for each example in the training set. The output of the last layer (fc4) is returned as the final output of the model.

In 2016, Facebook's AI team launched Pytorch [4], [51]. It is an open-source platform for deep learning. It incorporates dynamic computing, enabling more flexibility in creating complicated structures. Figure 2 presents the proposed model to estimate the effort estimation based on the Pytorch platform (PytEffort). There are several steps in this process. Firstly, the training data set is loaded into torch.utils.data.Dataset. The samples and their related labels are stored inside that dataset, while DataLoader encapsulates an iterable over Dataset to provide simple access to the samples. Secondly, the dataset is converted to a torch.Tensor before loading into data loader. Following that, the proposed model is defined based on Pytorch's nn.Module, the number of hidden layers is identified by experimental. Last but not least, the training loop is executed. The model parameters are updated by looping over the training data and doing forward and backward passes.

E. TRANSFER LEARNING APPROACH

Transfer learning is a method in machine learning designed to enhance the effectiveness of a specific task by utilizing the knowledge acquired while addressing a similar yet distinct problem [35], [36], [52]. This technique entails employing a pre-existing model as a foundation for training a novel model on a separate task. Ordinarily, the pre-trained model is trained on an extensive dataset, and the acquired features are utilized to initialize or refine the parameters of the new model. As mentioned in Section III, the instance of inductive transfer learning is when the source and target tasks are not equivalent regardless of whether the source and target domains are the same or not [35]. To tackle this issue, we might replace the last layer of the pre-trained model with a new layer that matches the size of the input features in different datasets. Subsequently, only this last layer is trained on the respective datasets, while keeping the rest of the model's parameters frozen. The details of those steps are presented as follows:

- Pre-trained model loading: The initial step involves loading a pre-trained model that has been trained on a big dataset. The best-performing model from the training process is selected to serve as the pre-trained model for further utilization.
- Freezing layers except the last one: In this step, all layers of the pre-trained model, except for the last layer, are frozen. By setting the requires_grad attribute of each parameter to False, the learned features of the pre-trained model are preserved. This allows for fine-tuning only the last layer to adapt it to the new task.
- New optimizer creation: A new optimizer is created specifically for the last layer of the model, which was set to require gradients in the previous step. In this case, we choose an optimizer as the same optimizer used for the pre-trained model.
- Continuing model training: The model is further trained using a new dataset through the standard PyTorch training loop. In each iteration, the input is passed through the model, the loss is calculated, gradients are computed, and the weights are updated using the optimizer. This training process is repeated for a specific number of epochs or until the model reaches convergence.

TABLE 4. The brief information of five datasets: ISBSG (2020), Desharnais, Albrecht, Kitchenham, and China.

Datasets	Source	No.Records	No.Features	Unit Effort
ISBSG(2020)	ISBSG [25]	9,592	251	person-hours
Desharnais	Promise [53]	81	12	person-hours
Albrecht	Promise [21]	24	8	person-hours
Kitchenham	Promise [54]	145	10	person-hours
China	Zenodo [55]	499	19	person-hours

VI. EXPERIMENTAL DESIGN

A. DATA DESCRIPTION

This research draws its observational projects from the ISBSG projects repository (version R1/2020) [25]. The dataset consists of 9,592 finished software projects, with 251 recorded characteristics. These characteristics are categorized into subsets such as rating, year, development type, and size. The sizing team represents the AFP and VAF. The Size group stores data about sizes, and Effort is divided into Summary Work Effort (SWE) and Normalized Work Effort (NWE). The Rating section uses letter grades (A through D) to rate the quality of a product. Other main types of information, such as industry sector, group size, and development type, are also recorded.

Additionally, to expand the scope of the study, four additional datasets (Desharnais, Albrecht, Kitchenham, and China) are incorporated to examine the necessity of using a pre-trained model. These datasets share the ''effort'' feature with the ISBSG dataset and are included to validate the potential improvement in prediction performance when using a pre-trained model on a similar dataset. The brief information on these datasets is presented in Table 4.

The primary objective of this inclusion is to thoroughly examine the necessity of using a pre-trained model (answer for RQ4). The ISBSG and those datasets are not equivalent. However, they share the same 'effort' feature, which indicates a direct relationship between the target tasks and the source task [35]. Thus, we might retrain the last layer of the pre-trained model (originally trained on the ISBSG dataset) on those datasets. This study aims to validate whether training a pre-trained model on a similar dataset would improve prediction performance compared to using the pre-trained model alone.

B. DATA PROCESSING

The ISBSG dataset is processed according to the following standards:

- Only projects with high-quality ratings (A and B) are considered, following recommendations from ISBSG and other publications [4], [25]. This reduces the number of projects in the dataset to 8,619, excluding projects with insufficient legitimacy or a combination of variables [56].
- Only the methods developed by IFPUG for counting Function Point Analysis (FPA) are used in this investigation. Other methods such as MarkII, NESMA, COSMIC, and FISMA are ignored, resulting in 6,365 records.

TABLE 5. Data Description - Dataset 1 (ranging from 1998 to 2000).

	SWE	EI	EO	EQ	ILF	EIF
Training Data (70%)						
Mean	5697.91	192.87	154.500	104.56	146.37	47.16
Std	5863.78	254.53	191.86	144.72	175.64	64.83
Minimum	529.00	3.00	4.00	3.00	7.00	5.00
Q1	1700.00	45.25	37.250	24.50	45.00	10.00
Q2	3634.50	106.50	84.50	57.00	80.00	25.00
Q3	7738.25	237.00	164.75	132.00	167.50	55.00
Maximum	27000.00	2221.00	1216.00	893.00	1137.000	572.00
Testing Data (30%)						
Mean	6989.08	220.38	136.70	129.44	223.63	39.25
Std	6359.29	301.62	169.94	205.34	326.12	48.60
Minimum	749.00	6.00	4.00	3.00	21.00	5.00
Q1	2217.50	33.50	25.00	23.50	37.00	10.00
Q2	4389.00	78.00	67.00	48.00	91.00	25.00
Q3	11264.50	288.50	172.50	124.00	284.00	53.00
Maximum	22960.00	1184.00	616.00	952.00	1252.00	300.00

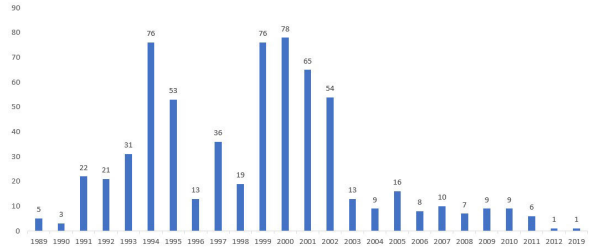

FIGURE 3. The number of selected projects from 1989 to 2019.

- The study focuses on specific variables such as SWE (person-hours), EI, EO, EQ, ILF, ELF (number of function points), and industry sector (IS), as proposed by Hoc et al. [4]. Projects not falling into these categories are excluded, leaving 1,515 projects for observation. The industry sector is classified into sixteen categories: Banking, Communication, Construction, Defence, Education, Electronics & Computers, Financial, Government, Insurance, Manufacturing, Medical Health Care, Mining, Professional Services, Service Industry, Utilities, and Wholesale Retail.
- The interquartile range (IQR) method is used to eliminate extreme cases. Values of SWE, EI, EO, EQ, ILF, and ELF that fall outside the range of Q1 (first quartile) - 1.5 * IQR to Q3 (third quartile) + 1.5 * IQR may be removed [57].

The selected dataset is divided into two groups: dataset 1, which includes completed projects from 1989 to 2000, accounting for two-thirds of the total dataset, and dataset 2, which includes completed projects after 2000 (from 2001 to 2019), accounting for one-third of the total dataset. Data descriptions for dataset 1 and dataset 2 are provided in Tables 5 and 6. Figure 3 shows the number of selected projects from 1989 to 2019.

Furthermore, the Pearson correlation method [58] was employed in this study to identify the attributes suitable in the transfer learning model (using a threshold greater than 0.5 to determine the strength of associations). A careful selection

TABLE 6. Data Description - Dataset 2 (ranging from 2001 to 2019).

	SWE	EI	EO	EQ	ILF	EIF
Training Data (70%)						
Mean	5299.93	153.39	137.60	123.96	174.39	83.03
Std	3900.97	195.65	154.20	149.47	191.83	113.28
Minimum	679.00	3.00	4.00	3.00	7.00	5.00
Q1	2165.250	42.00	42.00	21.50	55.50	20.00
Q2	4716.50	89.00	69.50	87.00	99.50	43.50
Q3	7104.75	186.50	175.25	159.25	220.00	86.250
Maximum	18600.00	1327.00	932.00	797.00	995.00	644.00
Testing Data (30%)						
Mean	6422.73	122.04	114.69	136.78	98.43	112.39
Std	5054.07	148.23	154.76	170.01	71.883635	213.43
Minimum	742.00	6.00	7.00	3.00	10.00	5.00
Q1	2983.00	43.00	34.00	36.50	40.00	31.00
Q2	4918.00	62.00	67.00	76.00	80.00	45.00
Q3	7821.50	129.50	129.00	138.50	151.50	87.00
Maximum	18825.00	584.00	698.00	653.00	231.00	977.00

TABLE 7. Lists used input and out variables among studied datasets.

Datasets	Input		Output
	Numerical	Categorical	
ISBSG	EI, EO, EQ, EIF, ILF	IS	SWE
Desharnais	Length, Transactions, Entities, PointsNonAdjust, PointsAjust	–	Effort
Albrecht	Input, Output, Inquiry, File, FPAdj, RawFPCount, AdjFP	–	Effort
Kitchenham	duration, AFP, Estimate	–	Effort
China	AFP, Input, Output Enquiry, File, Added	–	Effort

process was conducted to identify a subset of attributes that exhibited high correlation coefficients. In the case of the Desharnais dataset, the attributes 'Length', 'Transactions', 'Entities', 'PointsNonAdjust', and 'PointsAjust' were chosen due to their pronounced correlation. Similarly, for the Albrecht dataset, the attributes 'Input', 'Output', 'Inquiry', 'File', 'FPAdj', 'RawFPCount', and 'AdjFP' were identified as having significant correlation. Likewise, the China dataset highlighted 'AFP', 'Input', 'Output', 'Enquiry', 'File', and 'Added' as highly correlated attributes. Finally, the Kitchenham dataset revealed 'duration', 'AFP', and 'Estimate' as attributes with noteworthy correlations. As a result, the list of input/output variables used for this paper is mentioned in Table 7.

C. DATA PRE-PROCESSING

Data pre-processing techniques are applied to ensure the dataset is in a suitable format for analysis. The following steps are performed:

- One-hot encoding is used to transform the IS into a numerical format. Each possible category is represented by its own dummy variable. For example, if a project belongs to the "Banking" category, the corresponding dummy variable for "Banking" will be set to 1, while the dummy variable for others will be set to 0. This binary representation allows machine learning models to handle categorical variables effectively. Table 8 presents an example of one-hot encoding for software projects that belong to "Banking/Communication". In this binary representation, only the element corresponding to the "Banking/Communication" category is set to 1, indicating that the software project belongs to the "Banking/Communication" industry. This representation allows machine learning models to handle categorical variables and effectively learn patterns and relationships between different industry categories.
- Max-min normalization is applied to address the wide range of scales used by the dataset's features. The formula used for normalization is:

$$X_i = \frac{x_i - \min(X)}{\max(X) - \min(X)} \tag{5}$$

This ensures that all features contribute equally to the analysis and are on a similar scale. The sklearn.preprocessing.MinMaxScaler function in Python is used for this normalization process.

TABLE 8. An example of transforming the IS into a numerical format for Banking and Communication.

IS	Banking	Communication
Banking	1	0
Communication	0	1
Construction	0	0
Defence	0	0
Education	0	0
Electronics Computers	0	0
Financial	0	0
Government	0	0
Insurance	0	0
Manufacturing	0	0
Medical Health Care	0	0
Mining	0	0
Professional Services	0	0
Service Industry	0	0
Utilities	0	0
Wholesale Retail	0	0

D. PERFORMANCE METRICS

The Magnitude of Relative Error (MRE) (6) and Mean Magnitude of Relative Error (MMRE) (7) proposed by [59] are common indicators to measure the performance of effort estimation. Despite the fact that Myrtveit, Stensrud, and Shepperd believe these measures have some significant drawbacks because of the widespread usage of MMRE, they are still frequently employed in the validation of real effort estimation [60], [61], [62]. We will use MRE and MMRE to validate effort estimation accuracy in this research. Other metrics used to assess effort estimation accuracy include PRED(x) (8), MAE (9), and standardised accuracy (SA) [62] (10), Mean Balance Relative Error (MBRE) (11), and Mean Inverted Balance Relative Error (MIBRE) (12). Furthermore, this study includes helpful information on MMRE and PRED (0.25) [63], those are, sig_{Left} (14), sig_{Right} (15), and sig (16).

Sign values are determined using (13).

$$MRE_i = \frac{|y_i - \hat{y}_i|}{y_i} \tag{6}$$

$$MMRE = \frac{\sum_{i=1}^{N} MRE_i}{N} \tag{7}$$

$$PRED(x) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 1 & \text{if } MRE_i \le x \\ 0 & otherwise \end{cases} \tag{8}$$

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{9}$$

$$SA = 1 - \frac{MAE}{\overline{MAE}_{rguess}} \tag{10}$$

$$MBRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{min(y_i, \hat{y}_i)} \tag{11}$$

$$MIBRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{max(y_i, \hat{y}_i)} \tag{12}$$

$$sign(y_i, \hat{y}_i) = \begin{cases} 1 & \text{if } \hat{y}_i > y_i \\ 0 & \text{if } \hat{y}_i = y_i \\ -1 & \hat{y}_i < y_i \end{cases} \tag{13}$$

$$sig_{Left}(y, \hat{y}) = \frac{2}{N(N+1)}\sum_{k=1}^{N}\sum_{i=1}^{k} sign(y_i, \hat{y}_i) \tag{14}$$

$$sig_{Right}(y, \hat{y}) = \frac{2}{N(N+1)}\sum_{k=1}^{N}\sum_{i=1}^{k} sign(y_{N-i+1}, \hat{y}_{N-i+1}) \tag{15}$$

$$sig(y, \hat{y}) = \frac{sig_{Left}(y, \hat{y}) + sig_{Right}(y, \hat{y})}{2} \tag{16}$$

where $\hat{y}_i$ is the predicted and y_i is the i-th observed value. $\overline{MAE}_{rguess}$ is the MAE of a large number of random guesses (e.g., 1000 runs) [62]. MAE_{rguess} will converge on simply using the sample mean after a number of runs [64].

E. FLOW DIAGRAM

Figure 4 presents the flow diagram to build the EnsEffort model. As can be seen, there are two major phases. First, two of three models (XGBoost, HGBoost, RF) are chosen as generators, namely models 1 and 2, and the rest is the final regressor. 70% of dataset 1 are fed into models 1 and 2. Then, those generate predictions 1, and 2, respectively. Those predictions are combined into two feature sets, which are then used as inputs for the final regressor. The final predictions are then evaluated for performance using a performance evaluation method. The combination between XGBoost, HGBoost, and RF is presented in Table 9. That combination leads to 3 groups of study (group 1, group 2, and group 3). This combination aims to find the best-proposed model for the EnsEffort.

In the next step, we will assess the performance of EnsEffort and PytEffort in several scenarios. There are three scenarios to compare the performance of models, named Case 1, Case 2 and Case 3, where:

TABLE 9. The combination of three estimators.

Group	Model 1	Model 2	Final Regressor
Group 1	XGBoost	HGBoost	RF
Group 2	XGBoost	RF	HGBoost
Group 3	HGBoost	RF	XGBoost

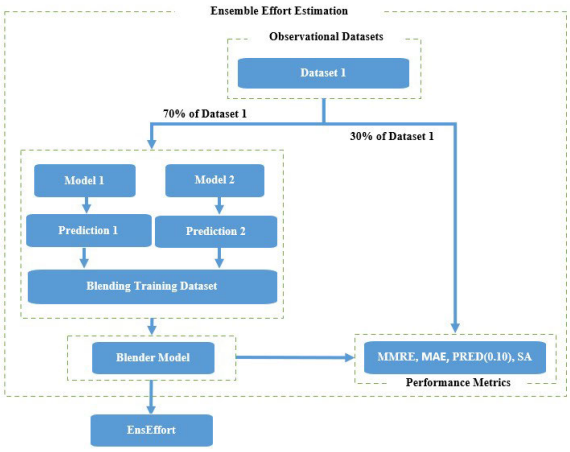

FIGURE 4. The flow diagram of EnsEffort model.

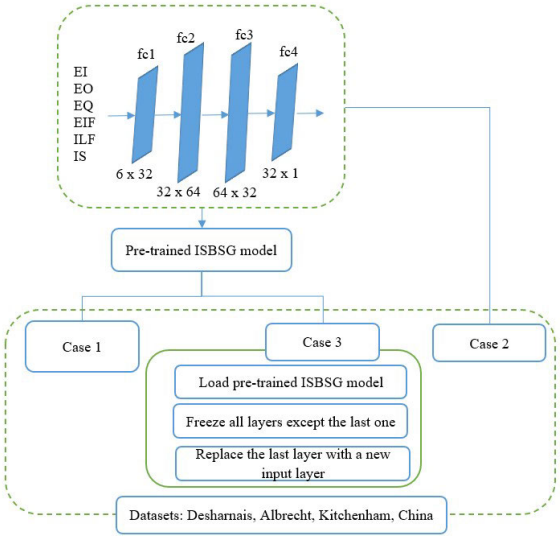

FIGURE 5. A scenario consisting of three cases (Case 1, Case 2, Case 3) aimed at evaluating the effectiveness of the trained model.

- Case 1: Using EnsEffort and PytEffort models trained based on dataset 1 to validate the performance of effort estimation based on 30% of dataset 2.
- Case 2: Using EnsEffort and PytEffort models, train them based on 70% of dataset 2 and validate the performance of effort estimation based on 30% of dataset 2.
- Case 3: This is a transfer learning approach (see Figure 5). Using PytEffort models trained by dataset 1 are called pre-trained models, and continue to train based on 70% of dataset 2 and validate the performance of effort estimation based on 30% of dataset 2.

The settings for XGBoost, HGBoost, and RF are determined by experimentation. The learning rate selected is 0.1;

further parameters such asÂ n_estimators and max_depth are determined through experimental. In PytEffort, every neuron gathers input from the previous layer's neuron. It generates output delivered to the following layer by employing an activation function to the weighted sum of its inputs. On the one hand, we adopt the rectified linear unit (ReLU) as the activation function. According to Jason Brownlee, ReLU [65] is easy to compute and consumes fewer computational resources. It also eliminates the problem of disappearing gradients, allowing models to train rapidly and increase performance [66]. On the other hand, optimization aims to discover the optimum set of parameters for a model that minimizes the loss function, which is the difference between the expected and actual outputs of the model. As stated in the paper [67], RMSProp and Adam use adaptive moment estimates to improve outcomes between Adam, RMSProp, Adaptive Gradient Algorithm (Adagrad), and a more robust version of Adagrad (Adadelta).

The EnsEffort will be compared with a PytEffort model and a baseline model as described in Equation (4) (FPA-IFPUG). A PytEffort model is installed based on the support of the Pytorch library [51]. The dataloader supported by torch.utils.data is used with a batch size of 64. As mentioned in the publication [4], the Adam optimizer is used with a learning rate of 0.01, and the loss function is a mean squared error (nn.MSELoss). Softmax with one dimension is used as the activation function for the last layer. In both approaches, cross-validation with 10-folds is used.

Figure 5 presents a scenario consisting of three cases (Case 1, Case 2, Case 3) aimed at evaluating the effectiveness of the trained model. The purpose is to assess whether the trained model might be directly used to make predictions on a relatively independent dataset compared to the data used for training or if it is necessary to continue training when using it to predict different datasets. To accomplish this, I will use the entire ISBSG dataset by incorporating Dataset 1 and Dataset 2 as training data for the pre-trained model. This approach allows me to leverage a broader range of data and increase the model's ability to generalize to different datasets.

The architecture of the scenarios involves three distinct cases for validating and building performance models. In each case, the four datasets, namely Albrecht, Desharnais, Kitchenham, and China, are divided into two parts. 70% of each dataset is allocated for training the models, while the remaining 30% is used for validation. In Case 1, a pre-trained model developed using the ISBSG dataset is loaded. This model is then validated using the respective validation subsets of the four datasets. Case 2 focuses on building a new model using the training subsets (70%) of the four datasets, followed by its validation using the corresponding validation subsets (30%).

Finally, in Case 3, a pre-trained model is loaded and further trained using the training subsets (70%) of the four datasets. As mentioned in Sections III, and V-E, due to the pre-trained model was initially built using the ISBSG dataset, which has different input features compared to the other datasets (Desharnais, Albrecht, Kitchenham, and China), we substitute the final layer of the pre-trained model with a freshly added layer designed to correspond with the dimensions of the input characteristics present in those datasets. Subsequently, only this last layer is trained on the respective datasets, while keeping the rest of the model's parameters frozen. This approach allows the pre-trained model to learn the mapping between the inputs of these datasets and the output. The resulting model is then validated using the respective validation subsets (30%). This architecture ensures that the models are trained on a majority of the data while still being evaluated on a separate portion, providing a reliable assessment of their performance and generalization capabilities.

F. VALIDITY EVALUATION

This study used a ''validity assessment'' method to examine the proposed models. This is not a good way to evaluate something, and it could make the experiment's results less reliable. In this case, we are discussing the procedure for validating statistical samples. To deal with this validity problem and ensure the proposed technique is thoroughly tested, a 10-fold cross-validation strategy was adopted. One further kind of assessment that could influence the reliability of the results is the choice of parameters in the PytEffort, EnsEffort. In this investigation, both the EnsEffort parameters and the PytEffort parameters found by trying things out are used.

External validity, or the question of whether the results can be used in a different setting, is at the heart of the research questions for this paper. ISBSG 2020 (release R1), Albrecht, Desharnais, Kitchenham and China were used to assess the accuracy of the predictions. Evaluation measures are used to determine how effective the proposed approach is. Evaluation measures such as MBRE, MIBRE, MAE, SA, and MMRE, PRED(0.25) with their useful information were utilized, as previously reported [4], [59], [62], [63]. Because of this, the results of the experiments in this research can be applied to a much larger group of projects.

VII. RESULTS AND DISCUSSION

Figure 6 illustrates the charts of MAE, MBRE, MIBE, SA, MMRE, and PRED(0.25) among three regressions (XGBoost, HGBoost, and RF). As can be seen, SA and PRED(0.25) attained from XGBoost reached the maximum, 0.49 and 0.45, respectively (see Table 10), while its MAE achieved the minimum (3147.78) compared with those gained from HGBoost, and RF. Moreover, p-values computed by Mann-Whitney U-Test [68] between XGBoost, HGBoost, and RF are less than 0.05. These results demonstrate that XGBoost yields the best performance compared with the others.

RQ1: Which of the prior combinations in the ensemble model could contribute to improved performance?

The performance of three groups (groups 1, 2, 3) is presented in Table 11 and Figure 7. As can be seen, the values of MBRE, MIBRE, MAE, and MMRE obtained from group 1 reach the minimum, while its PRED(0.25) and SA

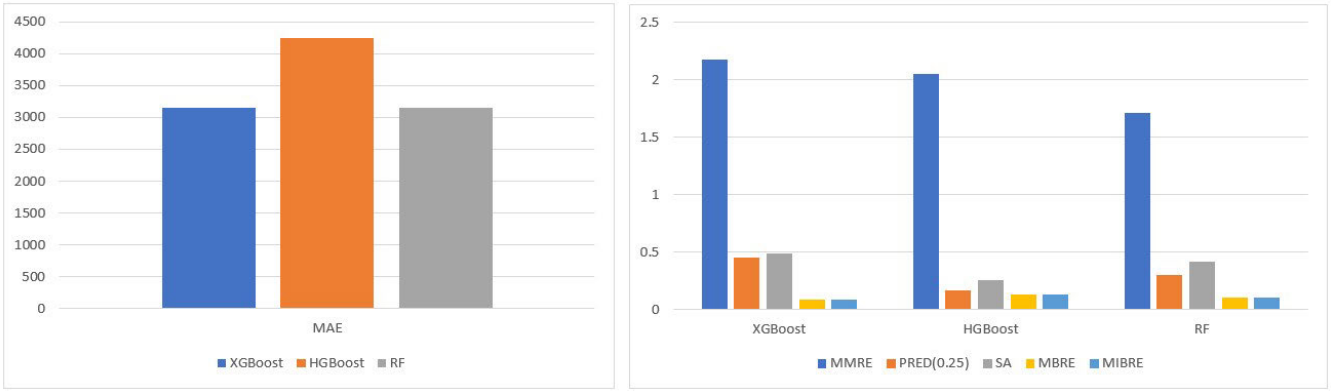

FIGURE 6. The statistical results between XGBoost, HGBoost, RF.

TABLE 10. The performance metrics obtained from XGBoost, HGBoost, and RF.

	XGBoost	HGBoost	RF
R-Square	0.69	0.51	0.67
MMRE	2.177	2.055	1.712
PRED(0.25)	0.450	0.170	0.298
sig_{Left}	0.078	0.078	0.071
sig_{Right}	-0.546	-0.376	-0.539
sig	-0.234	-0.149	-0.234
MAE	3147.78	4236.12	3147.78
SA	0.49	0.26	0.42
MBRE	0.090	0.129	0.102
MIBRE	0.079	0.109	0.088
p-value (Mann-Whitney U-test)			
XGBoost vs. HGBoost, RF	–	0.00	0.00
HGBoost vs. XGBoost, RF	0.00	–	0.00
RF vs. XGBoost, HGBoost, RF	0.00	0.00	–

get the maximum compared with group 2 and group 3. Furthermore, the p-value obtained from group 1 compared with groups 2 and 3 is smaller than 0.05; it is strong evidence to conclude that the performance of group 1 is better than groups 2 and 3. By contrast, there is no solid evidence for concluding whether group 2 is better than group 3 due to its p-value equal to 0.05. The results may answer RQ1: when XGBoost and HGBoost were used as generators, and RF was used as the final regressor, the accuracy was the best among the three groups. The findings also show that models would be more accurate if they used ensemble approaches instead of the single method (XGBoost, HGBoost, RF) in terms of FPA. In the future, we recommend using the ensemble technique with RF as the final regressor for effort estimation.

RQ2: Is the ensemble model estimation more accurate than the deep learning model?

Furthermore, we also study the accuracy of the EnsEffort/Group 1 compared with the PytEffort. As seen in Table 11, PRED(0.25), SA, MAE, MBRE, and MIBRE obtained from EnsEffort/Group 1 are better than those obtained from PytEffort. However, the MMRE-EnsEffort/Group 1 is slightly higher than the MMRE-PytEffort. Besides that, the absolute valuable information related to MMRE and PRED(0.25) (sig_{Left}, sig_{Right}, sig) attained from PytEffort might be slightly smaller than that from EnsEffort/Group 1. As mentioned in [63], PytEffort might be more stable than EnsEffort/Group 1. Additionally, the p-value computed by Mann-Whitney U-test between EnsEffort/Group 1 and PytEffort is less than 0.05, demonstrating that there is a statistically significant difference in the medians between the two methodologies.

RQ3: Is it feasible to use the trained model to estimate effort for recent projects? How effective is the trained model compared to one trained based on a new dataset, assuming both models use the same approach?

In the next step, we use trained models that were introduced with the oldest dataset to estimate the SDEE for new data sets. This is a crucial step in figuring out if a model that has already been trained might be used to estimate the effort for a new dataset (Case 1). How accurate is the model that has been trained compared to the model that was made using only the new data (Case 2)? If a model that has already been trained gives good results, it is helpful to train a model using a large set of historical data, and we will use that model to make predictions in the future.

Table 12 shows the performance of EnsEffort/Group 1 and PytEffort acquired based on Case 1 against Case 2. As can be seen, both approaches outperform FPA-IFPUG. PRED(0.25), SA obtained from two methods reach the maximum, and the other criteria reach the minimum compared with the FPA-IFPUG. This finding concludes that we might even use the trained model (Case 1) to estimate effort estimation for a new dataset.

However, the statistical results (MMRE, PRED(0.25), MAE, MBRE, MIBRE, SA) obtained from the EnsEffort/Group1 in Case 1 produce better outcomes than those in Case 2. The finding may be comparable to the conclusion reached by Anitha et al. [33]. However, the sig-EnsEffort value attained from Case 1 shows that the predicted effort might be less stable than in Case 2 because sig-EnsEffort-Case 2 is more significant than sig-EnsEffort-Case 1. This means that a model should be trained when predicting an entirely new dataset to get higher performance. In other words, modifying the trained model before making the prediction might yield better outcomes.

TABLE 11. The performance metrics obtained from Group 3.

	EnsEffort/Group 1	EnsEffort/Group 2	EnsEffort/Group 3	PytEffort
R-Square	0.856	0.405	0.601	0.71
MMRE	1.233	1.720	1.354	1.209
PRED(0.25)	0.574	0.149	0.260	0.540
sig_{Left}	0.372	0.183	0.401	-0.082
sig_{Right}	0.011	-0.395	-0.188	-0.270
sig	0.191	-0.106	0.106	-0.175
MAE	2528.88	4458.23	3680.85	2694.96
SA	0.63	0.21	0.38	0.55
MBRE	0.065	0.137	0.106	0.086
MIBRE	0.058	0.113	0.090	0.073
p-value (Mann-Whitney U-test)				
Group 1 vs. Group 2, 3, and PytEffort	–	0.00	0.00	0.00
Group 2 vs. Group 1, 3	0.00	–	0.05	0.00
Group 3 vs. Group 2, 3	0.00	0.05	–	0.00
PytEffort vs. Group 1, Group 2, 3	0.00	0.00	0.00	–

TABLE 12. The performance metrics obtained from EnsEffort/Group 1, PytEffort vs FPA-IFPUG.

	MMRE	PRED(0.25)	sig_{Left}	sig_{Right}	sig	MAE	SA	MBRE	MIBRE
EnsEffort/Group 1									
Case 1	1.017	0.348	0.145	0.290	0.217	3761.86	0.18	0.138	0.112
Case 2	1.558	0.304	-0.029	0.290	0.130	3870.35	0.15	0.143	0.118
PytEffort									
Case 1	0.735	0.247	-0.289	-0.206	-0.247	4521.34	0.19	0.171	0.135
Case 2	1.10	0.323	-0.468	0.189	-0.140	4213.93	0.21	0.152	0.122
Case 3	1.04	0.348	-0.167	-0.094	-0.130	3557.14	0.22	0.133	0.107
FPA-IFPUG									
	1.613	0.0	-0.601	-0.529	-0.565	5624.41	-0.33	0.247	0.182

RQ4: What approach (EnsEffort or PytEffort) will improve the accuracy of effort estimation? What happens if we employ transfer learning for PytEffort model?

As shown in Table 12, the MBRE, MIBRE and MAE values obtained from EnsEffort/Group1 may be better than those obtained from PytEffort. In contrast, the remaining values show that PytEffort gives more accurate prediction results. Due to the incomplete evaluation criteria, it is difficult to conclude which approach is better between EnsEffort and PytEffort. However, transfer learning techniques in deep learning technologies successfully handle information disparities across datasets and improve the model's prediction performance for the current data [69]. Using the transfer learning advantage, we suggest an additional transfer learning (Case 3) and compare deep learning outcomes using the transfer technique to the ensemble approach in this study.

As shown in Table 12, MMRE, MAE, MBRE, and MIBRE obtained from Case 3 are significantly smaller than those obtained from EnsEffort/Group 1 in both cases; meanwhile, SA and PRED(0.25) reach the maximum. In addition, statistical values (p-values) obtained from PytEffort/Case 3 are significantly less than 0.05 compared with EnsEffort/Group 1 (see Table 13). It reveals that employing transfer learning in the deep learning model might outperform than EnsEffort approach.

To answer the question "What happens if we employ transfer learning for the PytEffort model?", we will examine the results in Table 14. As can be seen, comparing Case 3 with Case 2 and Case 1 across the four datasets, we observe some notable trends. In general, Case 3 consistently outperforms Case 2 and Case 1 regarding various performance metrics. Firstly, considering the MMRE and MAE metrics,

TABLE 13. The p-values results obtained from the Mann-Whitney U-test in two cases based on EnsEffort/Group 1, PytEffort, and FPA-IFPUG.

	EnsEffort		FPA-IFPUG
	Case 1	Case 2	
EnsEffort/Group1			
Case 1	–	0.86	0.00
Case 2	0.86	–	0.00
PytEffort			
Case 1	0.02	0.11	0.00
Case 2	0.01	0.02	0.00
Case 3	0.00	0.02	0.00

Case 3 consistently achieves lower values than the other cases. It suggests that utilizing a pre-trained model and continuing training on additional datasets in Case 3 leads to a more accurate performance model than building a new model in Case 2 or using only the ISBSG dataset in Case 1. Secondly, when examining the PRED(0.25), Case 3 consistently achieves higher values than Cases 1 and 2. It indicates that leveraging a pre-trained model and further fine-tuning it with additional data in Case 3 enhances the predictive capability of the model.

Moreover, Table 15 shows the p-values obtained from the Mann-Whitney U test for each pairwise comparison between the cases within each dataset. These results might indicate that the performance metrics in those cases are statistically distinct.

VIII. THREATS TO VALIDITY

Internal threats. In our view, internal validity and the ability to make inferences about the optimal parameter setting for EnsEffort and PytEffort estimates pose the most significant dangers. Appropriate parameter selection is a potential

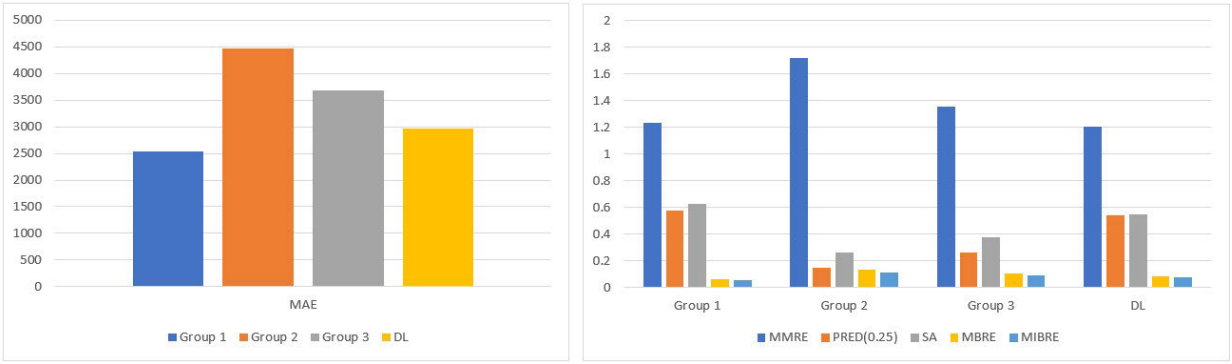

FIGURE 7. The statistical results between Group 1, 2, and 3.

TABLE 14. The performance metrics obtained from PyDL in three cases (case 1, 2, and 3) based on Albrecht, Desharnais, Kitchenham and China datasets.

	MMRE	PRED(0.25)	MAE	SA	MBRE	MIBRE
Albrecht						
Case 1	560.53	0.00	8143.68	0.00	574.64	6.48
Case 2	0.31	0.80	1.44	0.92	0.32	0.19
Case 3	0.08	0.91	0.56	0.96	0.09	0.07
Desharnais						
Case 1	0.55	0.25	3232.26	0.15	0.98	0.41
Case 2	0.29	0.68	1245.51	0.67	0.33	0.19
Case 3	0.21	0.81	1023.66	0.73	0.23	0.16
Kitchenham						
Case 1	33.58	0.00	65306.69	0.00	33.61	0.92
Case 2	0.61	0.53	547.89	0.60	0.64	0.31
Case 3	0.49	0.61	486.17	0.65	0.52	0.24
China						
Case 1	1.20	0.18	2769.88	0.24	2.34	0.68
Case 2	0.76	0.57	1074.54	0.75	0.81	0.38
Case 3	0.75	0.57	530.15	0.87	0.79	0.29

TABLE 15. The p-value obtained from the Mann-Whitney U-test in cases 1, 2, and 3 on the Albrecht, Desharnais, Kitchenham, and China datasets.

Datasets	Case 1	Case 2	Case 3
Albrecht			
Case 1	-	0.00	0.00
Case 2	0.00	-	0.00
Case 3	0.00	0.00	-
Desharnais			
Case 1	-	0.02	0.02
Case 2	0.02	-	0.03
Case 3	0.02	0.03	-
Kitchenham			
Case 1	-	0.00	0.00
Case 2	0.00	-	0.04
Case 3	0.00	0.04	-
China			
Case 1	-	0.00	0.00
Case 2	0.00	-	0.03
Case 3	0.00	0.03	-

danger to internal validity. No precise guidelines exist for selecting these parameters for each dataset.

Although it is commonly accepted that suitable parameters significantly influence identifying excellent fitness models, we used experiments to determine the parameter values for deep learning and proposed models in this work. We feel this choice is reasonable despite its time-consuming nature. Although the efficacy of the performance measure employed as the preventive criteria has been underlined, absolute certainty in this respect has been questioned, and we are obliged to depend on conventional performance measures; MMRE, MBRE, MIBRE,PRED(0.25), MAE, and SA, and the useful information related to MMRE, PRED(0.25) [4], [59], [62], [63]. We do not consider this a problem since our research was driven by prior studies that employed MMRE, MBRE, MIBRE, PRED(0.25), MAE, and SA as optimization criteria. We used 10-fold cross-validation to compare different adaption strategies. The primary reason is because 10-fold cross-validation has been used in prior research and is suggested by [4], [26], and [27] for comparing effort estimate models.

External threats. The ISBSG dataset 2020, Desharnais, Albrecht, Kitchenham, and China were used. In addition, we consider that certain datasets are too old for estimating software effort estimates since they reflect diverse software development methodologies and technologies.

IX. CONCLUSION

This study aims to develop prediction models using an ensemble approach. This is expected to improve the model's efficiency over the solo model. This paper's regression methods (XGBoost, RF, and HGBoost) offer good predictability. A combination of regression models is provided to ensure impartiality and accomplish the best choice of the regression techniques participating in the technical ensemble. A generator is chosen as the final regressor, and the others are selected as the prior generators. The generators are blended using the stacking method. As mentioned in the section above, group 1, where Random Forest is the final regressor while XGBoost and HGBoost are the prior predictions, might outperform the two other combined techniques based on dataset 1.

Furthermore, using the robust performance of the trained model to improve the effort estimation on a recent dataset might be a fantastic approach. It might be employed when the amount of new data sets required to train the model is insufficient, or we can not be privileged to access those datasets [36]. We adopted the model introduced in Phase 1 as the trained model for the second phase to clarify this claim. In this phase, we studied two scenarios (Case 1, Case 2) based on two subsets of dataset 2, as presented in section V-D.

The second goal is to compare the performance of effort estimation obtained from the ensemble against that obtained from the deep learning method. The findings produced in Cases 1 and 2 do not evaluate the strength of the deep learning approach compared to the ensemble method. However, we continue to explore by implementing transfer learning in a deep learning model. As a result, the transfer learning approach outperforms the ensemble method. In the future, we might inherit the best results from a previously trained model and apply transfer learning to continuously improve the model's performance based on the new dataset. It is the process of using the results of models that have already been trained on a specific set of databases to continue training on a new set of databases [36].

To be more specific, the results were experimentally validated on four datasets: Desharnais, Albrecht, Kitchenham, and China. Those datasets have a relationship with the ISBSG because they have the same target ('effort' feature). Notably, Case 3 exhibited superior performance compared to Cases 1 and 2, highlighting the advantages of utilizing a pre-trained model and further training it on additional datasets. This approach significantly improved accuracy, reduced errors, and enhanced predictive capabilities. These findings emphasize the importance of transfer learning and the incorporation of domain-specific knowledge from existing models to enhance the performance of new models.

Limitation These findings are based on limited data, which may not represent all possible scenarios. A trained model development should usually be done on a larger dataset. We will endeavour to collect additional data in the future to ensure data diversity.

REFERENCES

[1] M. Azzeh and A. B. Nassif, "Project productivity evaluation in early software effort estimation," *J. Softw., Evol. Process*, vol. 30, no. 12, p. e2110, Dec. 2018.

[2] H. T. Hoc, V. V. Hai, and H. L. T. K. Nhung, "A review of the regression models applicable to software project effort estimation," in *Proc. Comput. Methods Syst. Softw.*, 2019, pp. 399–407.

[3] A. Trendowicz and R. Jeffery, "Software project effort estimation," in *Software Project Effort Estimation Foundations and Best Practice Guidelines for Success*, vol. 12, 2014, pp. 277–293.

[4] H. T. Hoc, R. Silhavy, Z. Prokopova, and P. Silhavy, "Comparing multiple linear regression, deep learning and multiple perceptron for functional points estimation," *IEEE Access*, vol. 10, pp. 112187–112198, 2022.

[5] H. T. Hoc, V. Van Hai, H. L. T. K. Nhung, and R. Jasek, "Improving the performance of effort estimation in terms of function point analysis by balancing datasets," in *Software Engineering Application in Systems Design*, R. Silhavy, P. Silhavy, and Z. Prokopova, Eds. Cham, Switzerland: Springer, 2023, pp. 705–714.

[6] C. A. Behrens, "Measuring the productivity of computer systems development activities with function points," *IEEE Trans. Softw. Eng.*, vol. SE-9, no. 6, pp. 648–652, Nov. 1983.

[7] *Function Point Counting Practices Manualversion 4.3. 1*, International Function Point User Group (IFPUG), Princeton, NJ, USA, 2010.

[8] P. S. Kumar, H. S. Behera, 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, Nov. 2020, Art. no. 100288.

[9] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, vol. 137, pp. 184–196, Mar. 2018.

[10] H. T. Hoc, V. Van Hai, and H. Le Thi Kim Nhung, "Adamoptimizer for the optimisation of use case points estimation," in *Proc. Comput. Methods Syst. Softw.* Cham, Switzerland: Springer, 2020, pp. 747–756.

[11] R. Ferdiana and A. E. Permanasari, "Use case points based software effort prediction using regression analysis," in *Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, Oct. 2019, pp. 15–20.

[12] R. Silhavy, P. Silhavy, and Z. Prokopova, "Algorithmic optimisation method for improving use case points estimation," *PLoS ONE*, vol. 10, no. 11, Nov. 2015, Art. no. e0141887.

[13] R. Silhavy, P. Silhavy, and Z. Prokopova, "Analysis and selection of a regression model for the use case points method using a stepwise approach," *J. Syst. Softw.*, vol. 125, pp. 1–14, Mar. 2017.

[14] A. B. Nassif, L. F. Capretz, and D. Ho, "Estimating software effort using an ANN model based on use case points," in *Proc. 11th Int. Conf. Mach. Learn. Appl.*, vol. 2, Dec. 2012, pp. 42–47.

[15] R. Silhavy, P. Silhavy, and Z. Prokopova, "Evaluating subset selection methods for use case points estimation," *Inf. Softw. Technol.*, vol. 97, pp. 1–9, May 2018.

[16] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 1–39, Feb. 2010.

[17] (2022). *Cambridge*. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/solo

[18] A. Geron, "Ensemble learning and random forests," *Hands-on Machine Learning With Scikit-Learn & TensorFlow*. Sebastopol, CA, USA: O'Reilly Median, 2019, pp. 189–211, 2019.

[19] *International Function Point Users Group (IFPUG) Function Point Counting Practices Manual*, IFPUG, Milford, MA, USA, 2023.

[20] A. J. Albrecht, "Measuring application development productivity," in *Proc. Joint Share, Guide, IBM Appl. Develop. Symp.*, 1979, pp. 1–10.

[21] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Softw. Eng.*, vol. SE-9, no. 6, pp. 639–648, Nov. 1983.

[22] C. Gencel and O. Demirors, "Conceptual differences among functional size measurement methods," in *Proc. 1st Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2007, pp. 305–313.

[23] G. C. Low and D. R. Jeffery, "Function points in the estimation and evaluation of the software process," *IEEE Trans. Softw. Eng.*, vol. 16, no. 1, pp. 64–71, May 1990.

[24] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, "Software development cost estimation using function points," *IEEE Trans. Softw. Eng.*, vol. 20, no. 4, pp. 275–287, Apr. 1994.

[25] Int. Softw. Benchmarking Standards Group, Melbourne, VIC, Australia, 2020.

[26] H. T. Hoc, V. Van Hai, and H. L. T. K. Nhung, "An approach to adjust effort estimation of function point analysis," in *Software Engineering and Algorithms*, R. Silhavy, Ed. Cham, Switzerland: Springer, 2021, pp. 522–537.

[27] P. Silhavy, R. Silhavy, and Z. Prokopova, "Categorical variable segmentation model for software development effort estimation," *IEEE Access*, vol. 7, pp. 9618–9626, 2019.

[28] Z. Prokopova, P. Silhavy, and R. Silhavy, "Influence analysis of selected factors in the function point work effort estimation," in *Proc. Comput. Methods Syst. Softw.* Cham, Switzerland: Springer, 2018, pp. 112–124.

[29] F. B. Ahmad and L. M. Ibrahim, "Software development effort estimation techniques: A survey," *J. Educ. Sci.*, vol. 2, p. 23, Jan. 2022.

[30] Z. Abdelali, H. Mustapha, and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," *Proc. Comput. Sci.*, vol. 148, pp. 343–352, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919300420

[31] M. P. P. Kenichi, "Model-based software effort estimation—A robust comparison of 14 algorithms widely used in the data science community," *Int. J. Innov. Comput., Inf. Control*, vol. 15, pp. 1–15, Apr. 2019.

[32] S. K. Palaniswamy and R. Venkatesan, "Hyperparameters tuning of ensemble model for software effort estimation," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 6, pp. 6579–6589, Jun. 2021, doi: 10.1007/s12652-020-02277-4.

[33] V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," *Electronics*, vol. 10, no. 10, p. 1195, May 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/10/1195

[34] P. S. Kumar, H. S. Behera, J. Nayak, and B. Naik, "A pragmatic ensemble learning approach for effective software effort estimation," *Innov. Syst. Softw. Eng.*, vol. 18, no. 2, pp. 283–299, 2022, doi: 10.1007/s11334-020-00379-y.

[35] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Jan. 2021.

[36] E. Kocaguneli, T. Menzies, and E. Mendes, "Transfer learning in effort estimation," *Empirical Softw. Eng.*, vol. 20, pp. 813–843, Jan. 2015, doi: 10.1007/s10664-014-9300-5.

[37] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Softw. Eng.*, vol. 14, no. 5, pp. 540–578, Oct. 2009.
[38] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Trans. Softw. Eng.*, vol. 33, no. 5, pp. 316–329, May 2007.
[39] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.
[40] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992.
[41] L. Breiman, "Arcing the edge," Statist. Dept., Univ. California, Los Angeles, CA, USA, Tech. Rep., 486, 1997.
[42] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, Dec. 2002.
[43] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
[44] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1–12, Oct. 2001.
[45] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, Apr. 2000.
[46] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proc. 16th ACM Int. Conf. Predictive Models Data Anal. Softw. Eng.*, New York, NY, USA, Nov. 2020, pp. 1–10, doi: 10.1145/3416508.3417114.
[47] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[48] A. Guryanov, "Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees," in *Proc. Int. Conf. Anal. Images, Social Netw. Texts*. Cham, Switzerland: Springer, 2019, pp. 39–50.
[49] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[51] L. A. E. Stevens and T. Viehmann, *Deep Learning With PyTorch*, 2020.
[52] N. Agarwal, A. Sondhi, K. Chopra, and G. Singh, "Transfer learning: Survey and classification," in *Smart Innovations in Communication and Computational Sciences*, 2021, pp. 145–155.
[53] J. Desharnais, "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function," M.S. Thesis, Univ. Montreal, Montreal, QC, Canada, 1989.
[54] B. Kitchenham, S. Lawrence Pfleeger, B. McColl, and S. Eagan, "An empirical study of maintenance and development estimation accuracy," *J. Syst. Softw.*, vol. 64, no. 1, pp. 57–77, Oct. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121202000213
[55] F. H. Yun, "China: Effort estimation dataset [data set]," Zenodo, CERN, Eur. Org. Nucl. Res., IT Dept., Digit. Repositories Section, Geneva, Switzerland, 2010, doi: 10.5281/zenodo.268446.
[56] C. Jones, "By popular demand: Software estimating rules of thumb," *Computer*, vol. 29, no. 3, pp. 116–118, 1996. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/MC.1996.485905, doi: 10.1109/MC.1996.485905.
[57] P. J. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," *Data Mining Knowl. Discovery*, vol. 1, no. 1, pp. 73–79, 2011.
[58] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*, 2009, pp. 1–4.
[59] S. D. Conte, H. E. Dunsmore, and Y. Shen, *Software Engineering Metrics and Models*. San Francisco, CA, USA: Benjamin-Cummings, 1986.
[60] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure [software estimation]," *IEE Proc.-Softw.*, vol. 148, no. 3, pp. 81–85, Jun. 2001.
[61] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 380–391, May 2005.
[62] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012.
[63] H. H. Thai, P. Silhavy, M. Fajkus, Z. Prokopova, and R. Silhavy, "Propose-specific information related to prediction level at X and mean magnitude of relative error: A case study of software effort estimation," *Mathematics*, vol. 10, no. 24, pp. 1–14, 2022.
[64] T. Xia, R. Shu, X. Shen, and T. Menzies, "Sequential model optimization for software effort estimation," *IEEE Trans. Softw. Eng.*, vol. 48, no. 6, pp. 1994–2009, Jun. 2022.
[65] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018, *arXiv:1811.03378*.
[66] J. Brownlee, "A gentle introduction to the rectified linear unit (ReLU)," in *Machine Learning Mastery*, vol. 6, 2019.
[67] E. Okewu, S. Misra, and F.-S. Lius, "Parameter tuning using adaptive moment estimation in deep learning neural networks," in *Proc. 20th Int. Conf. Comput. Sci. Appl.* Cham, Switzerland: Springer, 2020, pp. 261–272.
[68] N. Nachar, "The Mann–Whitney U: A test for assessing whether two independent samples come from the same distribution," *Tuts. Quant. Methods Psychol.*, vol. 4, no. 1, pp. 13–20, Mar. 2008.
[69] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning—ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham, Switzerland: Springer, 2018, pp. 270–279.

HUYNH THAI HOC was born in Trà Vinh, Vietnam, in 1980. He received the B.S. degree in mathematics and computer science from the University of Science (HCMUS), Vietnam, in 2002, and the M.S. degree in geographic information system from the University of Technology (HCMUT), Vietnam, in 2007. He is currently pursuing the Ph.D. degree in software engineering with Tomas Bata University in Zlín, Czech Republic.

He was a GIS Developer with DITAGIS, HCMUT, from 2002 to 2007. From 2007 to 2014, he was a Lecturer with the Faculty of Information Technology, University of Natural Resources and Environment (HCMUNRE). From 2011 to 2018, he was also a Lecturer with the Faculty of Information Technology, Industrial University of Ho Chi Minh City (IUH), Vietnam. Since 2018, he has been a Lecturer with the Faculty of Information Technology, School of Technology, Van Lang University, Ho Chi Minh City, Vietnam. His research interests include software effort estimation and data science.

RADEK SILHAVY is an Associate Professor and a Senior Researcher with the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, and an Associate Professor of System Engineering and Informatics with a demonstrated history of working in research, higher education, project management, and software analysis. He is also involved in academic publishing as the editor-in-chief, editor, or reviewer. He received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlin, in 2009. His research interests are predictive analytics for software engineering, empirical methods in software engineering, or prediction models focused on cost, size, and effort estimations in system/software engineering.

ZDENKA PROKOPOVA was born in Rimavska Sobota, Slovak Republic, in 1965. She received the master's degree in automatic control theory and the Ph.D. degree in technical cybernetics from Slovak Technical University, in 1988 and 1993, respectively.

She was an Assistant with Slovak Technical University, from 1988 to 1993. From 1993 to 1995, she was a programmer of database systems in the data-lock business firm. From 1995 to 2000, she was a Lecturer with the Brno University of Technology. Since 2001, she has been with the Faculty of Applied Informatics, Tomas Bata University in Zlín, where she is currently an Associate Professor with the Department of Computer and Communication Systems. Her research interests include programming and applications of database systems, mathematical modeling, computer simulation, and the control of technological systems.

PETR SILHAVY is an Associate Professor with the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic. He is a Senior Research and Associate Professor of System Engineering and Informatics with a demonstrated history of working in research and higher education. He has expertise as a CTO and a Software Developer in database programming, database design, data management, and data science. He received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlin, in 2009. His research interests are prediction and empirical methods for software engineering.