Dynamic Feature-Aware Attention Fusion of DRN and Machine Learning Models for Software Defect Prediction

M. Murali Mohana Kumara Varma [1*], Dr.M.Giri [2]
[1] Department of CSE, BEST Innovation University, Gorantla, Sri Sathyasai District, Andhra Pradesh, India.
[2] Department of CAI, Mother Theresa Institute of Engineering and Technology, Chittoor, Andhra Pradesh, India.
E-mail: varmabestiu@gmail.com
[*]Corresponding author

Keywords: software defect prediction, feature-aware attention, Deep Residual Network (DRN), machine learning, base classifiers

Recieved: September 27, 2025

Software Defect Prediction (SDP) is significant for making sure the software developed is of quality. This is achieved by detecting error-prone components early in the development process. This allows prioritizing testing and quality assurance efforts to minimize maintenance costs, to avoid system failures, and finally to ensure the delivery of a reliable software system. Traditional machine learning (ML) models and ensemble approaches such as soft voting and stacking have shown promise, yet often rely on static weighting strategies that fail to adapt to instance-specific variations. In order to overcome this limitation, they proposed a new Feature-Aware Attention-Based Fusion Model to learn the outputs of multiple base learners and a Deep Residual Network (DRN) based on a dynamic attention mechanism conditioned on the base model predictions and the original input features that allows the model to learn adaptive, instance-specific weights for making the final prediction. The approach incorporates a PSO-CSO hybrid feature selection strategy for dimensionality reduction and uses SMOTE for handling class imbalance. In the fusion layer, a context-aware attention mechanism dynamically integrates the base predictions with the outputs of the Deep Residual Network (DRN). Extensive experiments were conducted on five NASA benchmark datasets (PC5, PC1, KC1, KC2, JM1), and evaluated in comparison to the individual classifiers, soft voting, stacking, and the DRN with static attention. The Empirical evidence suggests that the proposed model outperformed all standard approaches, achieving an accuracy of 0.99, an F1-score of 0.99, a precision of 0.98, a recall of 0.9976, and an impressive ROC (Receiver Operating Characteristic)-AUC (Area Under the Curve) of 0.9992. The results indicate that combining the feature context with the model outputs using attention can form a robust and high-accuracy framework for SDP.

Povzetek: Predlagan je fuzijski model za napovedovanje programskih napak, ki z izbiro značilk (PSO-CSO) in SMOTE ter dinamičnimi utežmi preseže klasične ansamble.

1 Introduction

Software defects represent a serious challenge in real-world software engineering because unrecognized discrepancies can severely reduce reliability and increase costs of maintenance if they are discovered near or after releasing the software [1]. In an attempt to mitigate these effects, SDP emerged as an important research area that is actively looking at ways to identify defect-prone modules in software prior to release [2]. Effective models enable developers to concentrate assessment and assurance mechanisms on the most vulnerable parts of the code, thereby decreasing the overall expensiveness of testing and allowing timely provision of high-assurance software systems. Early prediction and remediation of defects not only improve software reliability but also lead to substantial cost savings over the software life cycle. The application of traditional ML methods [3] to SDP models includes both simple classifiers and simple ensemble methods. Simple classifiers are simple to use but often limited in capture—they often perform well on specific segments of the data but do not capture all the

defect-inducing patterns that occur in more complex software projects in the data. Ensemble learning has produced better predictions, typically by taking outputs from multiple models. Previous research suggests that ensemble models tend to be preferable to single classifiers on defect prediction tasks [4]. However, there are limitations to traditional ensemble approaches; for example, simple voting or averaging ensemble methods apply a static weight to the base learners and practically assume that all of the models are equally useful for each instance [5]. Decision-making in the presence of uncertainty about base model reliability for certain subsets of modules/features has only been examined in very limited ways. More advanced ensembles like stacking [6] do have some level of a meta-learner to combine outputs from base models and typically improve upon accuracy when using uniform voting. However, stacking is traditionally employed on the basis of predictions from base models alone as inputs to the meta-learner and does not explicitly draw from the original software metrics or features for the final classification decision.

Recent advances in deep learning provide potential responses to the complexity of SDP [7]. Deep neural networks [8] can quickly learn rich, high-dimensional representations of software data, as opposed to relying on feature engineering [9]. As a result, deep learning methods are becoming more common in SDP research with a variety of different network architectures adapted to capture semantic context and structural details from code metrics or repositories [10]. Of particular interest in SDP are DO-NETs, which can be successful at deep levels to give rise to deep models. DO-NETs introduce skip connections, which result in learning tasks that can be defined in terms of residual functions that mitigate vanishing gradients. In doing so, DO-NETs allow a network to utilize accuracy from a larger depth. Therefore, the representation can capture complex non-linear dependencies in the representation that could be missed by shallow models or other forms of learning methods [11]. As such, modeling based on DO-NETs may be more appropriate for difficult prediction tasks, and practitioners are reporting experiencing superior performance using DO-NETs in SDP [12].

1.1 Research gap

Despite the robustness of deep learning and ensemble methods, an open question arises about the appropriate fusion of outputs from multiple predictive models in a context-aware manner. Rather than combining experts with fixed weights or via a black-box meta-model, an ideal combination should take into account input-instance characteristics and dynamically adjust the influence of each model. This work proposes a feature-aware, attention-based fusion method. The method is inspired by the mixture-of-experts paradigm, wherein a gating network learns to weight the various expert outputs according to the expert's special competence in different regions of the input space. An attention mechanism is used to calculate the weightings considering the original feature vector of a software module so that the results coming from different base learners can be properly amalgamated. This mechanism is adept at grasping and learning the relationships unfolding between input features and model outputs and thereby giving adaptive attention to the most pertinent predictions for each single instance. In contrast to voting schemes that suffer from a static nature, our newly introduced attention-based fusion method does compute instance-specific weights on the fly. Using the input features as context, the model may learn to trust one base predictor for modules that have a certain complexity metric and another predictor for modules with a different characteristic. Such dynamic, feature-conditioned fusion is opposed to the inflexibility of traditional ensembles and should realize more robust, accurate defect predictions across diverse software modules. To guide this study and evaluate the effectiveness of our proposed approach, we formulate the following research questions:

RQ1: Can a feature-aware dynamic attention fusion outperform static ensemble methods in SDP?
RQ2: Does the proposed model provide consistent generalization across different folds and datasets?
RQ3: Is the performance improvement statistically significant compared to state-of-the-art models?
RQ4: Is the proposed model efficient in terms of computational cost?

The rest of this paper is presented in the following order. Section 2 surveys the prior work on software defect prediction, including machine learning methods, deep learning models, ensemble methods, and attention-based methods. Section 3 presents the proposed method, including data preprocessing, PSO–CSO-based dimensionality reduction, base classifier design and the Deep Residual Network, and the feature-aware attention-based dynamic fusion strategy. Section 4 describes the experimental design, datasets, performance measures, and results, with in-depth discussion and comparison. Section 5 concludes the paper and future directions of research.

2 Related work

Recent advancements in SDP have led to amassing a wide range of methods, from traditional ML classifiers to new deep learning architectures and hybrid-style ensemble techniques. This section will review existing work under four broad categories, namely ML-based methods, deep learning models, ensemble strategies, and attention-based approaches, while emphasizing key methodologies with the recent advancements, along with performance trends and limitations.

2.1 Machine learning-based defect prediction

Traditional ML classifiers remain the cornerstone of SDP. Techniques based on logistic regression (LR), Naïve Bayes (NB), support vector machines (SVM), k-nearest neighbors (KNN), and decision tree-based models have been extensively used on traditional defect datasets. These classifiers mostly draw upon hand-crafted software metrics derived from code modules. Publicly available datasets including NASA's Metrics Data Program and the PROMISE datasets are common benchmarks. NASA designs PC1 through PC5, and others (CM1, KC, JM, etc.) are widely used in training as well as testing ML classifiers.

2.2 Deep learning techniques

Deep learning (DL) methods have become more dominant in recent years in defect prediction research. The most widely used architectures are deep feed-forward neural networks (multi-layer perceptrons), deep belief networks (DBN), convolutional neural networks (CNN), recurrent neural networks (particularly LSTM/GRU variants), and Transformer-based models [13]. These models have the capability to autonomously derive feature representations directly from raw data, bypassing manual feature extraction. CNNs have been used to take local code structures and spatial hierarchies. CNNs move filters along input data to learn local motifs that are associated with defects. CNNs have been employed on structured inputs such as AST-based representations to transform programs' ASTs into token vectors and pass them through a CNN, then merge the learned features with traditional metrics for end-to-end defect prediction [14]. CNNs perform well in learning short-range structural patterns of code. CNNs were utilized by other researchers for just-in-time defect prediction with commit data—a model employed parallel CNN channels to embed the code change and commit message, combining them in a fusion layer to predict if a commit introduces a bug [15]. SMOTE-Tomek sampling-based CNN-GRU-based model for class imbalance in SDP. Experimental results on PROMISE datasets demonstrate substantial performance gains over current state-of-the-art algorithms [16].

Deep feed-forward networks & autoencoders use multi-layer perceptrons for defect prediction from dozens of input metrics. Deep autoencoders have been utilized as unsupervised feature extractors, reducing high-dimensional metric data into lower space [17]. RNNs, LSTM networks, are well-suited to handle sequences and have been used to model source code as token sequences. LSTM-based models are capable of learning long-range contextual dependencies in code. Bahaweres et al. (2021) [18] represented code as sequences of tokens and input them into an LSTM to obtain embeddings that capture syntactic and semantic information. Khleel et al. [19] combined a bidirectional long short-term memory (Bi-LSTM) network with oversampling. A gated hierarchical LSTM (GH-LSTM) approach is proposed for semantic feature-based defect prediction. AST-based features and conventional PROMISE measures. Experiments show that GH-LSTM outperforms current models in numerous scenarios of evaluation [20]. Bandhu et al. [21] presented TML, a Transformer Meta-Learning framework that integrates adversarial domain adaptation, ensemble strategies, and Bayesian optimization to enhance CPDP accuracy. Experiments on a variety of datasets illustrate that TML substantially outperforms current CPDP methods in several performance measures.

2.3 Ensemble learning techniques

Ensemble learning has emerged as a pillar of contemporary defect prediction since it tends to produce more stable classifiers by summarizing the forecasts from diverse models. Ensemble methods may be homogeneous or heterogeneous. Techniques for ensemble learning have also become very common in SDP because they can pool the excellence of multiple classifiers and maximize predictive performance. Some common ensemble techniques are bagging, boosting, voting, and stacking [22].

Variance reduction is achieved with bagging-based methods like Random Forest by combining the predictions of different models trained on various subsets of data. Boosting methods, including AdaBoost and XGBoost [23], iteratively train models to learn from the mistakes of their previous ones to improve the accuracy of the model. Voting ensembles pool predictions of various base learners by using majority or weighted voting strategies, providing resilience to vulnerabilities of individual models. Stacking uses a meta-learner to aggregate base model predictions to enable the ensemble to learn the best combinations of predictions. Hybrid ensemble architectures that use deep learning models along with classical classifiers have also been researched more recently, further extending the scope of defect prediction accuracy and generalizability.

2.4 Attention mechanisms in defect prediction

An attention mechanism allows models to concentrate on the most important sections of an input. Attention in deep learning first appeared in sequence-to-sequence models. Self-attention was made prominent by the transformer model to handle long-range dependencies in a sequence. Attention mechanisms have been used in various novel ways in defect prediction. Self-attention mechanisms have been utilized in defect prediction to identify semantic relations in code and enhance interpretability. Models such as DPSAM and explainable transformers pinpoint defect-prone code locations by indicating influential regions of code, which leads to enhanced prediction performance, specifically F1-score. These models not only predict more accurately but also help developers identify in which locations defects can be expected within the code. [24] employed a self-attention mechanism to learn semantic features from abstract syntax trees (ASTs) for automatic defect prediction. Tested on seven open-source projects, it performs better than DBN and CNN-based approaches, with up to 16.8% and 14.4% F1-score improvement in WPDP and 23% and 60% improvement in CPDP. These findings indicate the better performance of DPSAM in both within-project and cross-project defect prediction cases.

[25] proposed the MFA (Multi Features Attention) model, which integrates deformable and self-attention mechanisms to extract and combine semantic and network features from ASTs. Tested on 21 Java projects in cross-version and cross-project settings, MFA showed significant gains, up to 41% performance improvement against previous models. Results verify that integrating various feature types improves defect prediction precision over individual features by a long margin. [26] integrated Graph Sample and Aggregate (GraphSAGE) and the Nomadic People Optimizer and applied Univariate Ensemble Feature Selection on features of the PROMISE dataset. Following

Table 1: Comparison of recent attention-based models for SDP

Model	**Authors (Year)**	**Dataset Used**	**Architecture Summary**	**Evaluation Metrics (values)**
Defect Prediction via Self-Attention mechanism (DPSAM)	Ting-Yan Yu et al. (2021) [24]	7 open-source projects	Abstract syntax trees (AST)-based model with self-attention mechanism	16.8% F1 improvement in Within Project Defect Prediction and up to 60% in Cross-Project Defect Prediction
MFA (Multi Features Attention)	Qiu et al. (2025) [25]	21 Java projects (cross-version & cross-project)	Deformable attention on AST & network features, fused via self-attention	CV: Acc=0.700, F1=0.614, AUC=0.711; CP: Acc=0.687, F1=0.575, AUC=0.696
DP-AGL	Munir et al. (2021) [27]	Code4Bench (119,989 C/C++ programs)	Bidirectional GRU + LSTM with attention on code tokens	Recall=0.980, Precision=0.617, Acc=0.750, F1=0.757
TSASS (Transformer-based Sequence Attention)	Yang et al. (2022) [28]	10 projects (ELFF dataset)	Transformer encoder on method-call sequences	Defect-density MAE (mean absolute error) 8% lower than baselines
GraphSAGE-NPO-SDP	P. Dhavakumar et al. (2025) [26]	PROMISE	GraphSAGE with Attention + Nomadic People Optimizer + UEFST	32.45–36.48% accuracy improvement

min-max normalization, the model classifies modules as defective or not, performing better in measures including accuracy, precision, and F-measure when compared with current models. Table. 1 provided summarizes state-of-the-art attention-based models for SDP and includes key comparative information across five important dimensions.

2.5 Why dynamic fusion with feature context

Prior attention-based approaches for SDP [24–28] have aimed to enhance the feature representation either through self-attention mechanisms or by enforcing static attention weights inside some deep models, including CNN and LSTM. Even if these methods are capable of enhancing feature learning, they do ignore the contextual relation of the input features and the reliability of the base learners' prediction. Our approach aims to bridge this gap by realizing a dynamic fusion mechanism in which attention weights are conditioned jointly by the original feature vector and outputs of multiple base learners since such explicit attention enables instance-specific weightings to adapt to module-level characteristics.

3 Proposed method

This study proposes a hybrid prediction framework that synergistically incorporates DRN with various classical ML models in an attention-based feature-aware fusion scheme. First, the dataset is subject to feature selection by means of a PSO-CSO hybrid optimizer, preprocessing, and class balancing with SMOTE. The six base classifiers and a specially designed DRN with a gated feature attention module and stacked residual blocks are jointly trained for deep semantic learning. The outputs of all models are then stacked and combined with the initial input features using a neural attention layer that learns instance-specific weights for every model's prediction. The end prediction is calculated through a weighted aggregation based on these attention scores. The proposed method's overall framework is depicted in Figure 1.

3.1 Data Preparation and pre-processing

Let the initial dataset be denoted as $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^{N}$. where $X_i \in \mathbb{R}^d$ represents the feature vector of d software metrics for the i^{th} instance, and $y_i \in \{0,1\}$ is the corresponding binary class label, with 1 indicating a defective module and 0 otherwise. The dataset used in this research are the NASA datasets, widely adopted for benchmarking in SDP. In order to lower the feature space and remove unnecessary or repetitive features, a hybrid metaheuristic feature selection technique, PSO-CSO, is utilized.

This is done with the intention to enhance model generalization as well as computational efficiency. After feature selection, the class imbalance problem inherent in defect datasets is tackled by Synthetic Minority Oversampling Technique (SMOTE), which creates artificial samples of the minority class like in Eq. (1),

$$|\{i : y_i = 0\}| \approx |\{i : y_i = 1\}| \qquad (1)$$

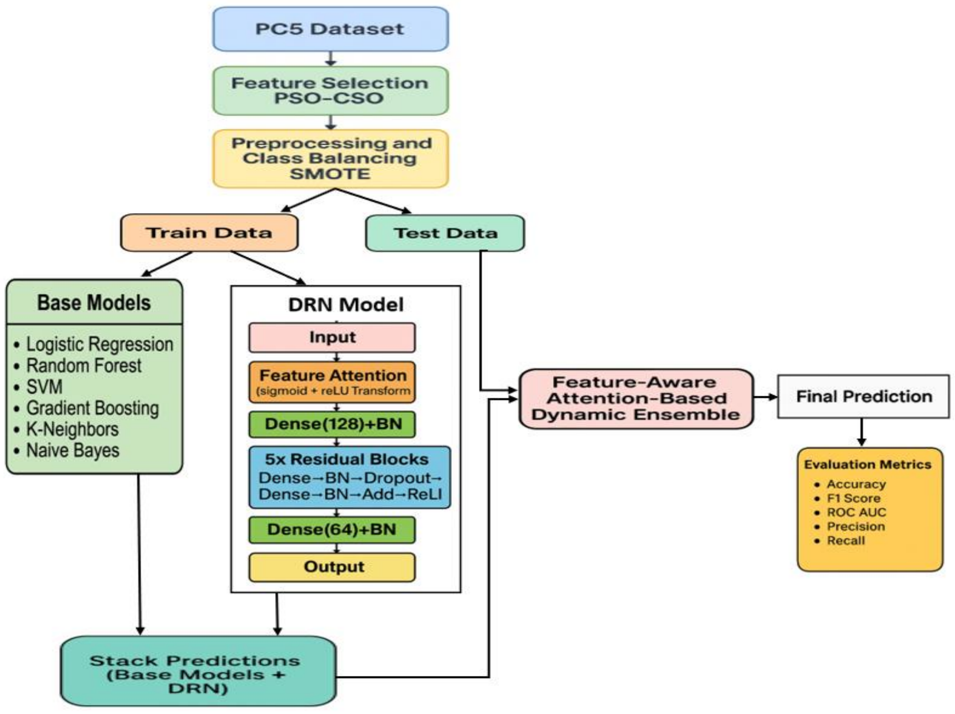

Figure 1: The architecture diagram of proposed model

Subsequently, all features are standardized using z-score normalization, defined as in Eq. (2),

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \forall j \in \{1,2,\dots,d'\} \qquad (2)$$

where μ_j and σ_j are the mean and standard deviation of feature $j, d' < d$ represents the number of features selected through PSO-CSO. The processed dataset is then divided into training and test subsets using an 80:20 ratio as in Eq. (3),

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}, \mathcal{D}_{train} \cap \mathcal{D}_{test} = \phi \qquad (3)$$

3.2 PSO-CSO hybrid feature selection

In order to improve prediction performance and feature dimensionality reduction, this paper presents a hybrid feature selection technique that integrates Particle Swarm Optimization (PSO) and Cuckoo Search Optimization (CSO). The hybrid method leverages the rapid convergence ability of PSO and the effective exploration property of CSO and is thus particularly apt at dealing with the complexity of high-dimensional software metrics. The appropriateness of each solution is assessed in terms of classification accuracy on a random forest model trained on the chosen subset. The objective function is given in Eq. (4),

$$\mathcal{F}(S) = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} [\hat{y}_i = y_i] \qquad (4)$$

where the indicator $[\hat{y}_i = y_i]$ equals 1 if the predicted label $\hat{y}_i$ matches the true label y_i , and 0 otherwise.

In PSO, each particle i is associated with a position vector $x_i \in [0,1]^d$ and a velocity vector $v_i \in \mathbb{R}^d$. The velocity and position updates are governed by Eq. (5) and (6),

$$v_{ij}^{t+1} = w.v_{ij}^{t} + c_1.r_1.(p_{ij} - x_{ij}^{t}) + c_2.r_2.(g_j - x_{ij}^{t}) \quad (5)$$

$$x_{ij}^{t+1} = \begin{cases} 1 \ if\ \sigma(v_{ij}^{t+1}) > rand() \\ 0 \ otherwise \end{cases} \quad (6)$$

Where w is the is the inertia weight, c_1 and c_2 are acceleration coefficients, $r_1, r_2 \sim U(0,1)$, p_{ij} is the personal best, and g_j is the global best. PSO emulates the brood parasitism behavior of cuckoo birds and leverages Lévy flight for global exploration. A new solution is generated using Eq. (7),

$$x_i^{t+1} = x_i^t + \alpha.Lévy(\lambda) \quad (7)$$
$$where\ Lévy \sim t^{-\lambda}, 1 < \lambda \leq 3$$

To integrate the feature subsets selected independently by PSO and CS, we adopt an averaging-based fusion strategy. Let S_{pso} and S_{cso} be the binary masks indicating selected features by each method. The final selection vector $\hat{S}$ is computed as Eq. (8),

$$\hat{S}_j = \begin{cases} 1, \ if\ (\frac{S_{pso_j} + S_{cso_j}}{2}) > 0.6 \\ 0, \ otherwise \end{cases} \quad (8)$$

The threshold $\theta = 0.6$ is empirically determined to strike a balance between feature relevance and compactness of the subset. This hybrid strategy ensures that only features with strong mutual agreement are retained for subsequent classification tasks.

For feature selection, PSO was configured with 30 particles and executed over 50 iterations. The inertia weight was set to 0.5, with cognitive and social coefficients both equal to 1.5. CS was also executed with 30 nests over 50 iterations, with a discovery rate (pa) of 0.25 and a Lévy flight parameter β = 1.5. The final selected feature subset was derived by averaging the binary solutions from both PSO and CS, using a threshold of 0.6.

3.3 Training of base ML models

To establish a diverse predictive foundation, six ML classifiers were employed as base models: LR, Random Forest (RF), SVC, Gradient Boosting Classifier (GBC), KNN, and Gaussian Naïve Bayes (GNB). Each of these classifiers brings unique inductive biases and decision mechanisms, enhancing the diversity required for the subsequent fusion process. Let the training dataset be represented as Eq. (9),

$$\mathcal{D}_{train} = (X_i, y_i)_{i=1}^{n} \quad (9)$$

Where $X_i \in \mathbb{R}^{d'}$ denotes the i^{th} feature vector after PSO-CSO feature selection and standardization, and $y_i \in \{0,1\}$ is its corresponding class label. Each base model M_k , $k = 1,2,\ldots,6$ is trained independently using $\mathcal{D}_{train}$. The training process for each model involves minimizing a task-specific loss function $\mathcal{L}_k$, such as binary cross-entropy for LR and GBC, hinge loss for SVC, or log-loss for GNB. This can be expressed as Eq. (10),

$$\min_{\theta_k} \mathcal{L}_k(\mathcal{D}_{train}, \theta_k) \quad (10)$$

Where θ_k denotes the set of parameters of the k^{th}. After training, each model is used to predict the class probabilities on the test dataset $\mathcal{D}_{test}$, generating a probability vector in Eq. (11),

$$P_k = M_k(\mathcal{D}_{test}) \in \mathbb{R}^m \quad (11)$$

Where $m = |\mathcal{D}_{test}|$. These predicted probabilities P_k are later combined with the DRN output in the attention-based fusion layer. By combining a diverse set of base learners with different strengths, linear decision boundaries in LR, non-linear ensembles in RF and GBC, kernel-based decisions in SVC, neighbourhood-based reasoning in KNN, and probabilistic modelling in GNB, the framework captures a broad spectrum of defect-related patterns.

3.4 DRN with feature-aware attention

To extract deep hierarchical representations from software metrics, a DRN is designed, integrated with a feature-aware attention mechanism to emphasize the most relevant features during training. This module serves as a deep learner in the hybrid prediction framework, complementing the shallow base models. The DRN architecture begins with a feature-aware attention mechanism that enhances input representation by learning dynamic feature importance weights. The attention-enhanced input X_{attn} is computed as Eq. (12),

$$X_{attn} = g(X) \odot t(X) = \sigma(W_g X + b_g) \odot ReLU(W_t X + b_t) \quad (12)$$

Where W_g, W_t, b_g and b_t are trainable parameters, and $\odot$ denotes element-wise multiplication. This is achieved by applying two parallel transformations: sigmoid activation $g(.)$ to produce a gating vector and A ReLU activation $t(.)$ to produce a transformed feature vector. This attended representation is then passed through a residual block-based architecture. Each residual unit consists of two fully connected layers followed by batch normalization, dropout, and skip connections as in Eq. (13),

$$\mathrm{h_t} = \mathrm{ReLU}(\mathrm{BN}(\mathrm{W_2}(\mathrm{Dropout}(\mathrm{ReLU}(\mathrm{BN}(\mathrm{W_1}\mathrm{h}_{\ell-1} + \mathrm{b_1}))) + \mathrm{b_2}))) + \mathrm{h}_{\ell-1} \quad (13)$$

Where $\mathrm{h}_{\ell-1}$ is the input to the ℓ^{th} residual block? In total, five stacked residual blocks are used to enable deep transformation while mitigating the vanishing gradient problem. The final layer is a dense output unit with sigmoid activation that produces a probability score as in Eq. (14),

$$\hat{y}_{drn} = \sigma(W_o \mathrm{h}_L + b_o) \quad (14)$$

Where L is the number of residual layers, and $\hat{y}_{drn} \in [0,1]$ indicates the predicted probability of defectiveness? To facilitate stable learning, the model is trained using the Adam optimizer and binary cross-entropy loss using Eq. (15),

$$\mathcal{L}_{drn} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{15}$$

This DRN model, equipped with the attention gate, enhances not just classification performance but also improves the interpretability of learned features by highlighting those contributing most to the defectiveness prediction. This architecture consisted of an initial dense layer with 128 ReLU-activated units, followed by five residual blocks, each comprising a pair of dense layers with batch normalization and dropout (rate = 0.3), combined through a skip connection. After the residual blocks, a 64-unit dense layer and a sigmoid output layer were added. The model was trained for 30 epochs with a batch size of 32 using the Adam optimizer. A learning rate scheduler (ReduceLROnPlateau) was used to adapt the learning rate dynamically based on validation loss.

3.5 Prediction aggregation from base models and DRN

After independently training the six base ML models and the DRN, the outputs from all these models are summarized to construct a rich prediction vector for each instance. This step ensures that the strengths of both shallow learners and the deep model are leveraged during the final decision-making process. The test dataset be represented as $\mathcal{D}_{test} = \{\mathrm{x}_i\}_{i=1}^{m}, \mathrm{x}_i \in \mathbb{R}^{d'}$. For each instance x_i, let $M_k(\mathrm{x}_i)$ denote the predicted probability score of defectiveness by the k^{th} base model, where $k = 1,2,\ldots 6$.

Similarly, let $M_{drn}(\mathrm{x}_i)$ denote the probability predicted by the DRN model. The aggregated prediction vector for the i^{th} sample is then:

$$\mathrm{z}_i = [M_1(\mathrm{x}_i), M_2(\mathrm{x}_i), \ldots.. M_6(\mathrm{x}_i), M_{drn}(\mathrm{x}_i)] \in \mathbb{R}^7$$

All such prediction vectors from the test set are stacked into a matrix as in Eq. (16),

$$Z = \begin{bmatrix} \mathrm{z}_1 \\ \mathrm{z}_2 \\ . \\ . \\ . \\ \mathrm{z}_m \end{bmatrix} \in \mathbb{R}^{m\times 7} \tag{16}$$

Here, each row of Z represents the combined prediction outputs from all base models and the DRN for a given software module. These predictions serve as meta-features that capture diverse decision perspectives ranging from probabilistic to kernel-based and tree-based inferences, as well as deep learned feature transformations.

This aggregated matrix Z along with the original feature vectors $X_{test} \in \mathbb{R}^{m\times d'}$, is passed into the feature-aware attention fusion model to perform final prediction. By combining the model outputs with the original features, the fusion model learns to adaptively assign weights to the prediction of each model based on the properties of the input instance. This stage ensures that instead of treating model predictions equally or statically, the framework uses both model output diversity and instance-specific feature context for informed and adaptive decision fusion.

3.6 Feature-aware attention-based dynamic fusion

To combine the predictive strengths of the base classifiers and the DRN adaptively, a Feature-Aware Attention-Based Dynamic Fusion mechanism is proposed. Unlike static ensemble techniques, this dynamic model learns instance-specific weights for each model output based on both the original feature representation and the model prediction scores. $\mathrm{z}_i \in \mathbb{R}^7$ denote the prediction scores of 6 base models and DRN for instance i. $\mathrm{x}_i \in \mathbb{R}^{d'}$ denote the original feature vector for instance i. These two vectors are concatenated to form a unified representation in Eq. (17). where ‖ denotes vector concatenation.

$$\mathrm{u}_i = [\mathrm{z}_i \parallel \mathrm{x}_i] \in \mathbb{R}^{7+d'} \tag{17}$$

This concatenated vector is passed through two dense layers to extract nonlinear feature interactions Eq. (18),

$$\mathrm{h}_1 = ReLU(W_1\mathrm{u}_i + b_1)\,, \mathrm{h}_2 = ReLU(W_2\mathrm{h}_1 + b_2) \tag{18}$$

Then, a softmax-based attention layer computes the attention weights over the 7 prediction values as in Eq. (19),

$$\alpha_i = \mathrm{softmax}(W_a\mathrm{h}_2 + b_a) \in \mathbb{R}^7 \tag{19}$$

Each element $\alpha_{ij} \in [0,1]$ represents the learned importance of the j^{th} model for instance i, with $\sum_{j=1}^{7}\alpha_{ij} = 1$. The final prediction is computed using Eq. (20) as a weighted sum of model outputs:

$$\hat{y}_i = \sum_{j=1}^{7}\alpha_{ij}.z_{ij} = \alpha_i^{\mathrm{T}}\mathrm{z}_i \tag{20}$$

This dynamic attention-based aggregation allows the model to prioritize different base models depending on the context provided by x_i. The fusion model is trained end-to-end using binary cross-entropy loss. By dynamically adapting to both input features and base model behavior, the proposed fusion strategy significantly improves prediction accuracy and robustness across varying software modules and defect patterns. This architecture is visually illustrated in Fig. 2.

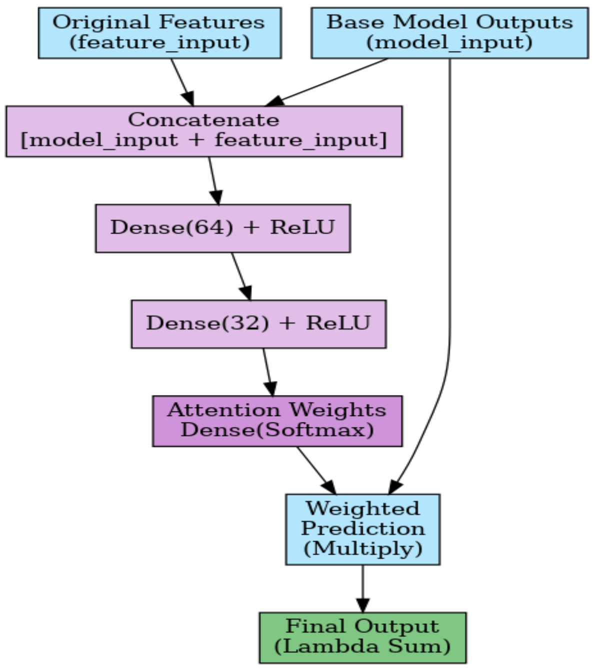

Figure 2: Architecture of feature-aware attention module for final prediction

This incorporated the predictions from all base classifiers and the original features. The fusion layer architecture included two dense layers with 64 and 32 ReLU-activated units, followed by a softmax layer to generate attention weights. These weights were used to compute a weighted sum of base model predictions. This ensemble model was trained for 40 epochs with a batch size of 16, using binary cross-entropy loss and the Adam optimizer.

3.7 Comparison with adaptive control analogies

Our fusion model's dynamic feature-aware attention mechanism can be intuitively compared with adaptive control methods from control theory. In control theory, adaptive control systems have controller parameters that update in real time to accommodate uncertainties and varying environments, similar to how our model's attention weights update instance-wise. This explains a theoretical explanation that our mechanism acts like an "adaptive controller" that adjusts the contribution of each base learner based on the input features and prediction scenario, seeking optimal performance on every software module. By extension, in the same way that an adaptive controller repeatedly updates its control law through feedback, our attention mechanism repeatedly rescales the predictors' weights by the feature-based context, which may be thought of as a type of feedback in feature space.

In adaptive fuzzy control systems, fuzzy rules in adaptive fuzzy control systems are tuned online to deal with system uncertainties. Boulkroune et al. (2025) [29] illustrate this through practical fixed-time synchronization in fractional-order chaotic systems using an adaptive fuzzy sliding-mode controller, where the controller learns to adjust for complicated uncertainties using fuzzy estimators. This is akin to our methodology in that the attention fusion layer "learns" to assign weights to classifiers based on the feature patterns of each example, similar to how an adaptive fuzzy controller assigns greater weight to some rules depending on conditions. Similarly, an output-feedback adaptive controller for chaotic systems by Boulkroune et al. (2017) [30] involves online learning of unseen dynamics using adaptive fuzzy systems. In that framework, the controller adaptively adjusts itself from the synchronization error observed, which is similar to our attention mechanism adjusting weights from the output differences between base models. Additionally, adaptive control methods in control theory also provide another analogy. Zouari et al. (2012) [31] proposed an adaptive control algorithm for uncertain multivariable systems based on a neural network for online-disturbance/model-error compensation that is ensured to be system robustness with Lyapunov-stable updates for the parameters. Conceptually, it is comparable to our attention-based fusion network, which is itself a miniature neural network learning to down-weight noisy base learners and up-weight the good ones per input. In each instance, a learning module is integrated into the decision process in order to impart adaptability and insensitivity to uncertainty.

Adaptive backstepping control is another appropriate analogy. It is a recursive design method of control where the controller is constructed in layers ("steps"), and each step adapts to uncertainties to ensure stability. This philosophy is captured in our multi-level fusion process. Zouari et al. (2013) [32] proposed an adaptive backstepping controller for a family of uncertain SISO nonlinear systems that ensures uniform ultimate boundedness of the closed-loop signals and forces the tracking error to zero in spite of uncertainties. In a different work [33], they applied an adaptive backstepping strategy to a single-link flexible robotic manipulator actuated by a DC motor, successfully accommodating the flexible dynamics via online parameter adaptation. These controllers tune their internal gains at every step of backstepping according to Lyapunov analysis in order to be stable and perform well. The concept of stability in adaptive control is applied here to predictability and consistency. The fact that our model can keep high accuracy across varied modules indicates that the attention gating has a stabilizing influence so that no one base learner's mistakes overwhelm the prediction. As future research, it is possible to use backstepping's theory tools to examine and possibly ensure the stability of the attention weight adaptation process within our neural mode

The proposed mechanism is similar to nonlinear optimal control strategies. In optimal control, the optimized controller re-optimizes the control input at each timestep to minimize a cost function. Each timestep, it essentially determines what is the best choice if the control inputs can maximize the usefulness of the current state of the system. Rigatos et al. (2023) [34] developed a nonlinear optimal controller for a gas compressor–induction motor. A disadvantage is that each iteration of the optimal control algorithm repeatedly solves an algebraic Riccati equation to continually optimize the control law in real time. The result is that every moment, the actuation decision is optimal for whichever current operating point the nonlinear system finds itself at. This analogy opens up an interesting direction for future work because it will allow us to suggest that similar concepts from nonlinear optimal control, such as stability guarantees and performance bounds, can be used to improve our attention mechanism. Investigating such cross-disciplinary strategies is a potentially productive future direction to improve the adaptivity and reliability of these dynamic attention-based software defect predictors for software systems.

4 Results and analysis

To assess the efficacy of the proposed model, a thorough comparison was carried out against several baseline classifiers, as well as ensemble methods and DRN variants.

4.1 Comparison of proposed model with base learners

This subsection provides a comparative study between the suggested Dynamic Feature-Aware Attention-based Fusion Model and six ML classifiers: LR, random forest, SVC, gradient boosting, KNN, and naive Bayes in Fig.3 and 4. The suggested approach performed best in all the metrics tested (Accuracy: 0.9900, F1 Score: 0.9900, Precision: 0.9826, Recall: 0.9976, and ROC AUC: 0.9992). Among base learners, Random Forest performed the best overall (accuracy: 0.9844, ROC AUC: 0.9985). Naive Bayes, on the other hand, produced relatively weaker performance, particularly in recall (0.4248), showing its weak ability in correctly identifying defective modules. The above results are especially clear in showing the superiority of ensemble models in a dynamic attention mechanism combining multiple models, which performs clearly much better than individual classifiers.

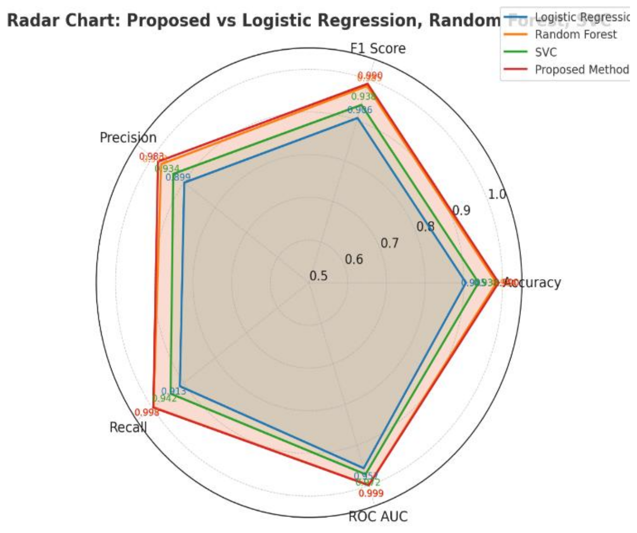

Figure 3: Proposed vs (LR, RF, SVC)

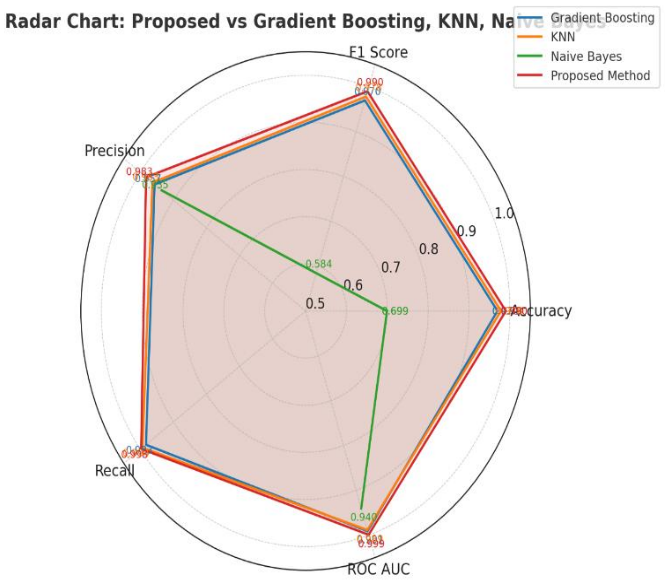

Figure 4: Proposed vs (GB, KNN, NB)

4.2 Comparison with traditional ensemble methods (voting and stacking)

In Fig.5, the model is contrasted against conventional ensemble techniques, which are Voting Classifier and Stacking Classifier. Voting (Accuracy: 0.9672, ROC AUC: 0.9894) and Stacking (Accuracy: 0.9886, ROC AUC: 0.9986) techniques both demonstrated great predictive performance. Importantly, the Stacking Classifier had a performance very close to the proposed model, but the proposed dynamic fusion method was still better, with improvements in Accuracy (+0.14%), F1 Score (+0.14%), Precision (+0.11%), Recall (+0.40%), and ROC AUC (+0.06%). This shows the value added from using feature-aware dynamic weighting, as opposed to static or predefined aggregation methods.

4.3 Comparison with DRN and DRN with Static Attention

Here the new dynamic fusion model and two deep learning-based models: a baseline DRN and a DRN variant with static (non-adaptive) attention weights are compared in Fig.6. The baseline DRN registered an accuracy of 0.9699 and a ROC AUC of 0.9902, which increased dramatically with the addition of static attention weights (accuracy: 0.9847, ROC AUC: 0.9990). Even with this significant performance improvement, the introduced dynamic attention mechanism still surpassed these findings, reaching the highest ever recorded performance figures (accuracy: 0.9900, ROC AUC: 0.9992). This indicates the superiority of dynamically adapting attention weights in consideration of both prediction and input feature context, ensuring better adaptability and accuracy.

4.4 Performance on NASA datasets

The model has been evaluated using five NASA benchmark datasets (PC5, PC1, KC2, KC1, and JM1). Classification performance is detailed in Table.2. The model has shown very strong performance across all five NASA datasets, with ROC AUC values of above 0.95 in every single case. For the PC5 dataset, the model gave almost perfect results (accuracy = 0.9900, F1 = 0.9900, ROC AUC = 0.9992), showing the effectiveness of the dynamic feature-aware attention mechanism. On PC1 and JM1, the framework continued to have high levels of accuracy (0.9174 and 0.9258, respectively) and good precision–recall balance, thereby confirming strong defect detection ability in larger projects. Likewise, KC1 performed competitively (accuracy = 0.9174, F1 = 0.9197, ROC AUC = 0.9731), giving a bit more weight to recall, thereby reflecting a model that tries to capture every possible defect-prone module. Whereas KC2 posed the hardest challenge, good results were obtained for this dataset as well (accuracy = 0.8855, ROC AUC = 0.9554), with recall (0.9302) being greater than precision (0.8602), indicating high sensitivity to detecting defects even if the cost was to endure some false positives. These results, on the whole, indicate that the proposed methodology generalizes across heterogeneous datasets with high predictive power and recall-oriented behavior, which is desirable for software defects, as it is worse to miss defective modules than to raise a false alarm.

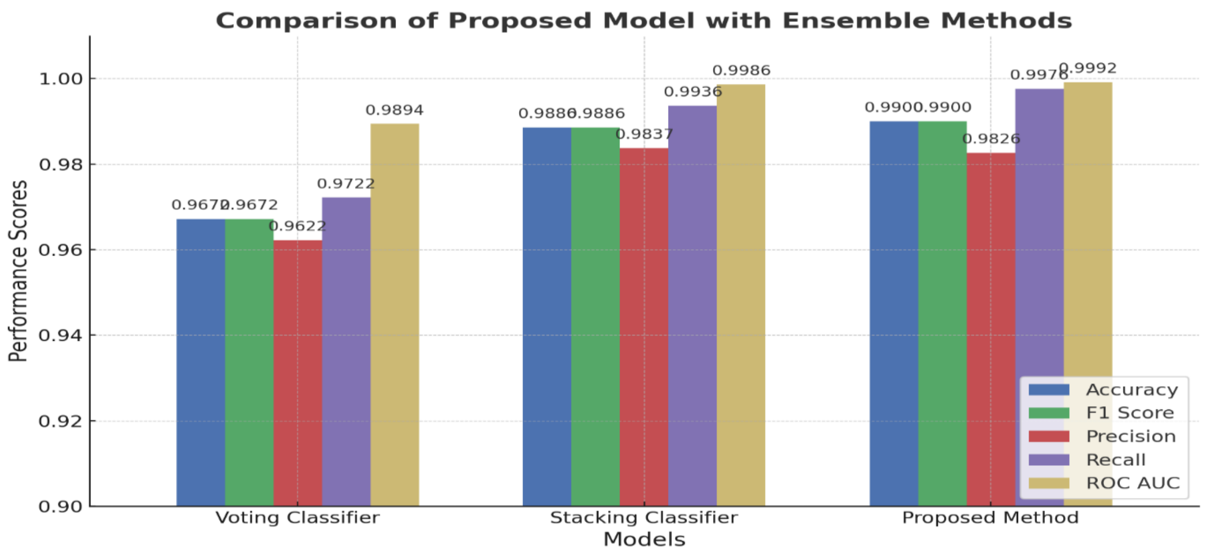

Figure 5: Performance comparison of proposed model with ensemble methods

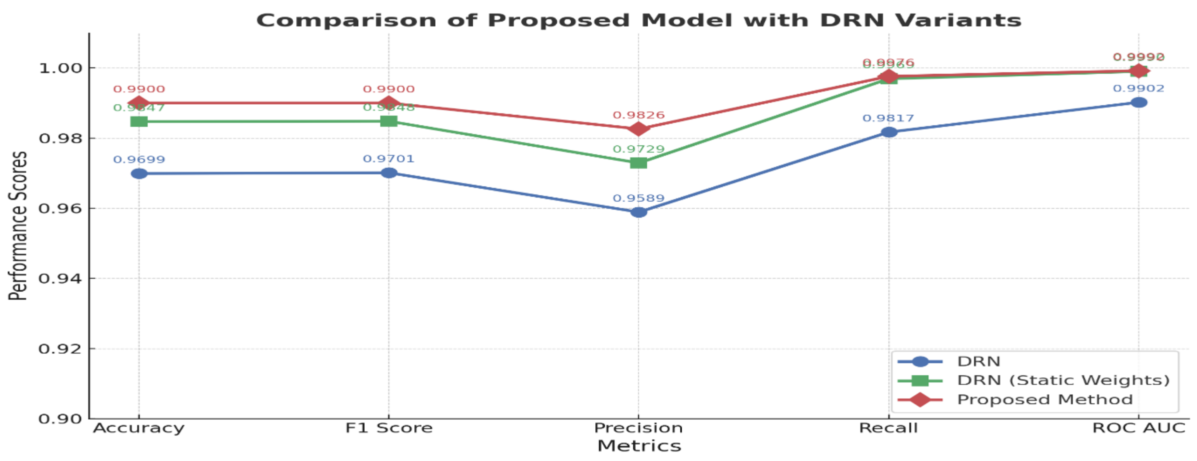

Figure 6: Comparison of Proposed model with DRN variants

Table 2: Performance of the model on multiple datasets

Dataset	Accuracy	F1 Score	Precision	Recall	ROC AUC
PC5	0.9900	0.9900	0.9826	0.9976	0.9992
PC1	0.9174	0.9142	0.9296	0.8993	0.9751
KC2	0.8855	0.8939	0.8602	0.9302	0.9554
KC1	0.9174	0.9197	0.9086	0.9311	0.9731
JM1	0.9258	0.9277	0.9189	0.9366	0.9705

4.5 Ablation study on feature selection approaches

In order to assess the performance of the hybrid PSO-CSO feature selection method, an ablation study was carried out comparing four different setups: (i) no feature selection, (ii) PSO-based selection, (iii) CSO-based selection, and (iv) a combination of the two strategies, PSO-CSO. The outcomes of the experiments are illustrated in Fig. 7. The model that used no feature selection at all managed to get a very high score (F1 Score: 0.9776, AUC: 0.9994), which means that the dataset was quite easy to separate. On the other hand, when only one of the two algorithms, either PSO or CSO, was used, the results were lower than that of the non-selection model (F1 Scores: 0.9572 and 0.9605, respectively), which meant the classification was less precise and accurate. Nevertheless, the hybrid PSO-CSO selection method gave a very good result and improved all metrics considerably (Accuracy: 0.9900, F1 Score: 0.9900, AUC: 0.9992). It is thus assured that the combining method takes advantage of the powerful sides of both methods while controlling their disadvantages. This indisputably prioritizes the hybrid method as the most powerful in terms of obtaining better classification results.

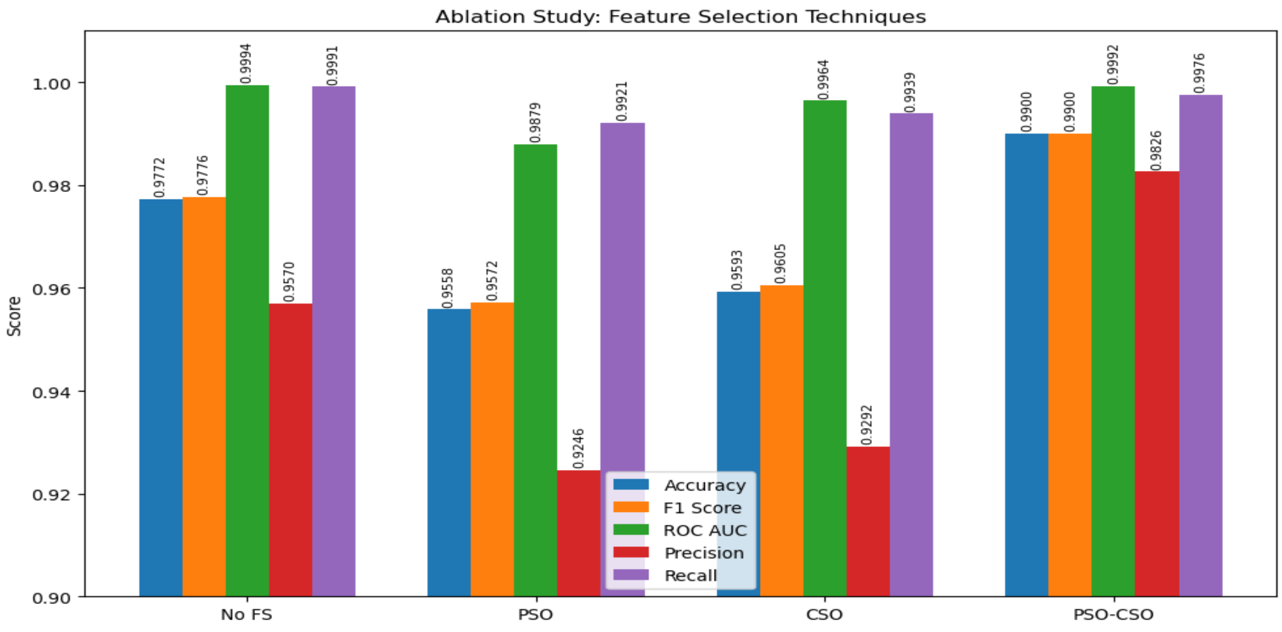

Figure 7: Performance Impact of Feature Selection Techniques in SDP

4.6 Comparative analysis and computational efficiency

The superiority of the proposed model is attributed to adaptively combining base learner predictions with original input features, unlike various state-of-the-art ensembling approaches such as soft voting, stacking, and DRN, which rely on static aggregation. This provides instance-specific weighting, which supports better generalization across varying characteristics of defects, as demonstrated by consistent high ROC AUC scores (>0.95) across all NASA datasets. Compared to the recent models such as DPSAM, MFA, and TML, our model allows for improved defect identification without requiring manual configuration of attention. Although DPSAM improves F1 scores in both WPDP and CPDP, our approach yields a nearly perfect classification on PC5 (F1 = 0.99, AUC = 0.9992) without architectural complications or additional feature engineering.

Table 3: Summary of computational efficiency metrics for the model

Metric	Value
DRN Training Time	268.90 seconds
DRN Model Size	2.25 MB
DRN Total Parameters	1,81,505
Memory Usage	613.70 MB
Inference Time (Test Set) for the ensemble model	1.18 seconds

The computational cost of the model was calculated to identify its viability for deployment in real-world scenarios. As can be seen in Table 3, the DRN backbone had a complete training time of 268.90 seconds and produced a small model size of just over 2.25 MB, demonstrating efficiency in resource-limited settings. The number of total trainable parameters in the model was 181,505, indicating an efficient and expressive architecture. The ensemble model had a low inference time of 1.18 seconds, as it processed the entire test set with low latency, which indicates a viable real-time prediction environment. The memory consumption during the implementation of the model was 613.70 MB, indicating ease of integration into continuous integration and automated defect detection environments. Although the model has been evaluated on NASA datasets, its architecture indicates great potential for generalization to real-world industrial datasets and a diverse range of software domains, hence motivating future studies in broader cross-project and cross-domain contexts.

4.7 Performance validation and SOTA comparison

Results from the proposed model strongly indicate that it fits into the main research goal of improving SDP accuracy. The performance of the model was almost perfect, with Accuracy = 0.9910 ± 0.0008, F1 Score = 0.9910 ± 0.0008, and ROC AUC = 0.9994 ± 0.0003 in 5-fold cross-validation.

The low standard deviations seen in the bar chart in Fig.8 confirm stability and consistency in the performance across the folds. This helps ease the worries of overfitting issues, even with the small and similar nature of the NASA datasets.

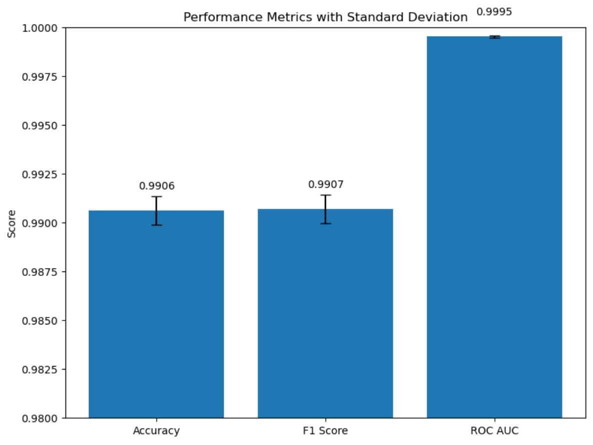

Figure 8: Average performance with variance across folds

To substantiate the statistical reliability and to highlight the advantages of the new ensemble model over the SOTA baselines, a thorough process of statistical significance tests was done. The first step was the conducting of the paired t-test, which was performed between the ensemble and each of the individual base learners. The test showed p-values < 0.05 for all the comparisons, as can be seen in Fig. 9, so it was confirmed that there were significant improvements from the statistical point of view. Then, the Friedman test was used to find out how the F1 scores varied across all models. The result ($\chi^2 = 35.0000$, p = 0.0000) indicated that the differences among the models were not only significant but also statistically so. The examination of disagreement over predictions between the ensemble and the top-calibrated baselines through the McNemar test was also done. The ensemble and random forest comparison gave rise to a test statistic of 3245.0881 together with a p-value of 0.0000, which is an indication of the very strong statistical support for the ensemble's superiority. All these sophisticated statistical tests demonstrate that our model has a significantly better predictive performance than the traditional individual classifiers combined with the DRN.

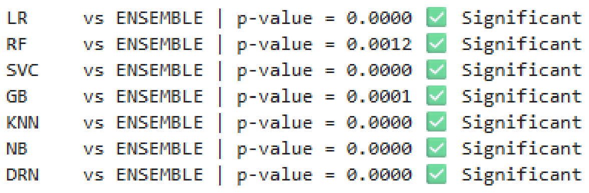
```
LR     vs ENSEMBLE | p-value = 0.0000 ✅ Significant
RF     vs ENSEMBLE | p-value = 0.0012 ✅ Significant
SVC    vs ENSEMBLE | p-value = 0.0000 ✅ Significant
GB     vs ENSEMBLE | p-value = 0.0001 ✅ Significant
KNN    vs ENSEMBLE | p-value = 0.0000 ✅ Significant
NB     vs ENSEMBLE | p-value = 0.0000 ✅ Significant
DRN    vs ENSEMBLE | p-value = 0.0000 ✅ Significant
```

Figure 9: Paired t-Test Summary

4.8 Model interpretability via attention visualization

Fig.10 shows a heatmap displaying the attention weights assigned to various base models using the dynamic attention for the first 10 test instances. A row corresponds to each of these individual test samples, while a column corresponds to each of the ensemble's base learners, including the deep residual network. The color intensity signifies the attention weight, with darker or more saturated colors representing higher weights.

The model does not uniformly weight all base learners. It changes the weight of each learner based on the current instance. Consider Sample 4 and Sample 10. They show very high attention to one learner. The weight values are nearly 1, which means that the model relied on that learner to make the prediction. This sign shows confidence in that model's prediction for these instances. Sample 1 has a more dispersed attention pattern, represented by two or more learners being assigned middle-range weights. That means it has combined several sources into a balanced decision. Samples 6 and 7 have extremely focused attention towards a single learner, indicating that in certain cases, there might be only one base learner that could be trusted enough to have significant influence over the final decision.

Overall, there is great variation in attention distribution among instances, validating the dynamic fusion capability of the model regarding instance awareness. Instead of considering all the learners equally for information, the mechanism identifies which learner or combination is most informative for a given sample. This not only improves predictive accuracy but also enhances interpretability to explain which models contribute to specific predictions and to what degree. Such insight is important in many applications where understanding the decision-making process is as important as the outcome itself.

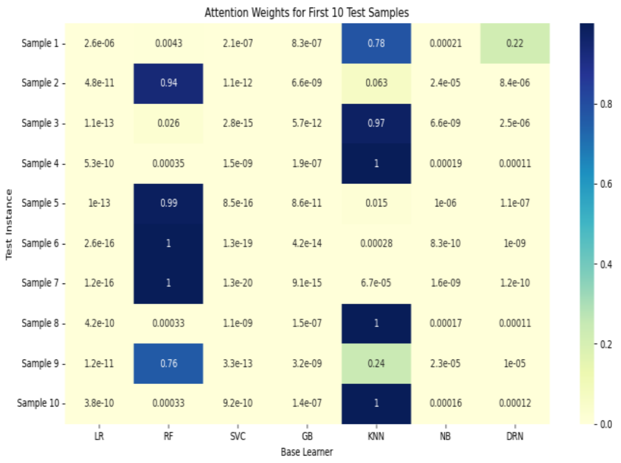

Figure 10: Heatmap of attention weights for first 10 test instances

4.9 Discussion aligned with research Questions

RQ1: Can a feature-aware dynamic attention fusion outperform satic ensemble methods in SDP?

Yes. As seen in Sections 4.1 to 4.3, the proposed dynamic fusion model achieved superior performance over individual classifiers, static ensembles (Voting, Stacking), and DRN variants. It achieved an F1 Score of 0.9900 and ROC AUC of 0.9992 on PC5, and consistently outperformed others across all evaluation metrics. Even against strong static attention DRNs and stacking classifiers, it showed measurable improvements (e.g., +0.14% in F1 Score), validating its instance-specific aggregation advantage.

RQ2: Does the proposed model provide consistent generalization across different folds and datasets?

Yes. As shown in Section 4.7 (Fig. 8), 5-fold cross-validation produced nearly perfect results (F1 Score = 0.9910 ± 0.0008, ROC AUC = 0.9994 ± 0.0003), indicating low variance and strong generalization. Additionally, Section 4.4 (Table 2) demonstrates robust performance across five diverse NASA datasets (PC1, PC5, KC1, KC2, JM1), confirming adaptability across varying defect patterns.

RQ3: Is the performance improvement statistically significant compared to state-of-the-art models?

Yes. Section 4.7 presents extensive statistical validation through Paired t-tests, the Friedman test (χ^2 = 35.0, p = 0.0000), and McNemar's test (statistic = 3245.0881, p = 0.0000), all confirming statistically significant superiority of the proposed ensemble model over baseline learners and DRNs (see Fig. 9). These outcomes underscore the model's reliability and its advancement over prior approaches.

RQ4: Is the proposed model efficient in terms of computational cost?

Yes, as presented in Section 4.6: Comparative Analysis and Computational Efficiency, the model is computationally efficient—requiring only 268.90 seconds for DRN training, consuming 613.70 MB of memory, and completing inference in 1.18 seconds—making it suitable for real-time and resource-constrained environments.

5 Conclusion

This study introduced a new dynamic feature-aware attention mechanism that aimed to boost the prediction by synergistically incorporating predictions from a conventional ML classifier and a deep residual neural network (DRN). The proposed approach employed a hybrid PSO-CSO feature selection technique for dimensionality reduction and data quality enhancement, mitigating the regular problems of feature redundancy and class imbalance that are rampant in software metrics datasets. Experimental performance carried out on the NASA datasets evidently illustrated that the suggested attention-based fusion technique evidently outperformed individual base learners, traditional ensemble methods (voting and stacking classifiers), and traditional and static-attention DRN implementations. More precisely, the suggested technique attained excellent predictive accuracy (0.9900), F1 score (0.9900), ROC AUC (0.9992), precision (0.9826), and recall (0.9976), signifying a significant improvement in correctly and reliably identifying defect-prone modules.

In addition, the dynamic attention method brought in instance-level adaptability, implicitly prioritizing the predictions of models best equipped for every given instance of data. This adaptability served not just to drive predictive accuracy but also contributed to better interpretability of the model decision process. The results support the significance of integrating dynamic models and attention mechanisms in SDP applications. Possible future research directions are examining the applicability of this dynamic attention-based approach to other software projects and repositories. Examining how explainable AI techniques can be integrated into further understanding model decisions and applying this framework to multi-class or multi-output defect prediction problems also provide interesting areas for developing software quality assurance practice.

Data and code availability

All datasets used in this study are publicly available from NASA's PROMISE repository. The implementation code and trained models are currently with the authors and will be provided upon request.

References

[1] A. Ali, N. Khan, M. Abu-Tair, J. Noppen, S. McClean and I. McChesney, "Discriminating features-based cost-sensitive approach for software defect prediction." *Automated Software Engineering*, Vol.28, no. 2, p.11, 2021. https://doi.org/10.1007/s10515-021-00289-8.

[2] S. Haldar and L.F. Capretz, "Interpretable software maintenance and support effort prediction using machine learning." *In Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pp. 288-289. 2024. https://doi.org/10.1145/3639478.3643069.

[3] K. Tanaka, A. Monden and Z. Yücel, "Prediction of software defects using automated machine learning." *In 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 490-494. IEEE, 2019. doi: 10.1109/SNPD.2019.8935839.

[4] F. Matloob, T.M. Ghazal, N. Taleb, S. Aftab, M. Ahmad, M.A. Khan, S. Abbas and T.R. Soomro, "Software defect prediction using ensemble learning: A systematic literature review." *IEEE Access*, Vol. 9, pp. 98754-98771, 2021. doi: 10.1109/ACCESS.2021.3095559.

[5] R.J. Jacob, R.J. Kamat, N.M. Sahithya, S.S. John and S.P. Shankar, "Voting based ensemble classification for software defect prediction." *In 2021 IEEE Mysore Sub Section International Conference (MysuruCon)*, pp. 358-365. IEEE, 2021. doi: 10.1109/MysuruCon52639.2021.9641713.

[6] L. Chen, C. Wang and S. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection." *Complex & Intelligent Systems*, Vol. 8, no. 4, pp. 3333-3348, 2022. https://doi.org/10.1007/s40747-022-00676-y.

[7] G. Giray, K.E. Bennin, Ö. Köksal, Ö. Babur and B. Tekinerdogan, "On the use of deep learning in software defect prediction." *Journal of Systems and Software*, Vol.195, p. 111537, 2023. https://doi.org/10.1016/j.jss.2022.111537.

[8] D.P. Gottumukkala, P.R. PVGD and S.K. Rao, " Optimizing Software Defect Prediction using a Hybrid Neural Network with Weighted Modified Cuckoo Search", *Proceedings on Engineering Sciences*, Vol.7, no. 3, pp. 2127-2136, 2025. DOI: 10.24874/PES07.03B.013.

[9] M. Ali, T. Mazhar, A. Al-Rasheed, T. Shahzad, Y.Y. Ghadi and M.A. Khan, "Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning." *PeerJ Computer Science*, Vol. 10, p. e1860, 2024. https://doi.org/10.7717/peerj-cs.1860

[10] W. Albattah and M. Alzahrani, "Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach." *AI*, Vol. 5, no. 4, pp. 1743-1758, 2024. https://doi.org/10.3390/ai5040086.

[11] Á.J. Sánchez-García, X. Limon, S. Dominguez-Isidro, D.J. Olvera-Villeda and J.C. Perez-Arriaga, "Class Balancing Approaches to Improve for Software Defect Prediction Estimations: A Comparative Study." *Programming and Computer Software*, Vol. 50, no. 8, pp. 621-647, 2024. https://doi.org/10.1134/S036176882470066X.

[12] V. Yakovyna and O. Nesterchuk, "Cross-Project Software Defect Prediction Using Ensemble Model with Individual Data Balancing and Feature Selection." *In International Conference on Advances in Mobile Computing and Multimedia Intelligence*, pp. 161-175. Cham: Springer Nature Switzerland, 2024. https://doi.org/10.1007/978-3-031-78049-3_15.

[13] E.N. Akimova, A.Y. Bersenev, A.A. Deikov, K.S. Kobylkin, A.V. Konygin, I.P. Mezentsev and V.E. Misilov, "A survey on software defect prediction using deep learning." Vol. 9, No.11, p. 1180, 2021. https://doi.org/10.3390/math9111180.

[14] A.T. Elbosaty, W.M. Abdelmoez and E. Elfakharany, "Within-Project Defect Prediction Using Improved CNN Model via Extracting the Source Code Features." *In 2022 International Arab Conference on Information Technology (ACIT)*, pp. 1-8. IEEE, 2022. doi: 10.1109/ACIT57182.2022.9994220.

[15] W. Zhuang, H. Wang and X. Zhang, "Just-in-time defect prediction based on AST change embedding." *Knowledge-Based Systems*, Vol. 248, p.108852, 2022. https://doi.org/10.1016/j.knosys.2022.108852.

[16] N.A.A Khleel and K. Nehéz, "A novel approach for software defect prediction using CNN and GRU based on SMOTE Tomek method." *Journal of Intelligent Information Systems*, Vol. 60, no. 3, pp. 673-707, 2023. https://doi.org/10.1007/s10844-023-00793-1.

[17] K. Rajnish and V. Bhattacharjee, "A cognitive and neural network approach for software defect prediction." *Journal of Intelligent & Fuzzy Systems*, Vol. 43, no. 5, pp. 6477-6503, 2022. https://doi.org/10.3233/JIFS-220497.

[18] R.B. Bahaweres, D. Jumral, I. Hermadi, A.I. Suroso and Y. Arkeman, "Hybrid software defect prediction based on LSTM (long short-term memory) and word embedding." *In 2021 2nd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*, pp. 70-75. IEEE, 2021. doi: 10.1109/ICON-SONICS53103.2021.9617182.

[19] N.A.A Khleel and K. Nehéz, "Software defect prediction using a bidirectional LSTM network combined with oversampling techniques." *Cluster Computing*, Vol. 27, no. 3, pp. 3615-3638, 2024. https://doi.org/10.1007/s10586-023-04170-z.

[20] H. Wang, W. Zhuang and X. Zhang, "Software defect prediction based on gated hierarchical LSTMs." *IEEE Transactions on Reliability*, Vol. 70, no. 2, pp. 711-727, 2021. doi: 10.1109/TR.2020.3047396.

[21] H. Bandhu, A. Ali, S. McClean, H. Ullah, M. Abu-Tair, A. Ziolkowski and J. Noppen, "TML: A Transformer-Based Meta-Learning Framework for Cross-Project Software Defect Prediction." (2024). https://doi.org/10.21203/rs.3.rs-5382592/v1.

[22] F. Matloob, T.M. Ghazal, N. Taleb, S. Aftab, M. Ahmad, M.A. Khan, S. Abbas and T.R. Soomro, "Software defect prediction using ensemble learning: A systematic literature review." *IEEE Access*, Vol. 9, pp. 98754-98771, 2021. doi: 10.1109/ACCESS.2021.3095559.

[23] H. Yang and M. Li, "Software defect prediction based on SMOTE-Tomek and XGBoost." *In International Conference on Bio-Inspired Computing: Theories and Applications*, pp. 12-31. Singapore: Springer Singapore, 2021. https://doi.org/10.1007/978-981-19-1253-5_2.

[24] T.Y. Yu, C.Y. Huang, and N.C. Fang, "Use of deep learning model with attention mechanism for software fault prediction." *In 2021 8th international conference on dependable systems and their applications (DSA)*, pp. 161-171. IEEE, 2021. doi: 10.1109/DSA52907.2021.00025.

[25] S. Qiu, B.E and J. He, "Features extraction and fusion by attention mechanism for software defect prediction." *PloS one*, Vol. 20, no. 4, p. e0320808, 2025. https://doi.org/10.1371/journal.pone.0320808.

[26] P. Dhavakumar and S. Vengadeswaran, "Software Defect Prediction using Graph Sample and Aggregate-Attention Network optimized with Nomadic People optimizer for Enhancing the Software Reliability." *Computer Standards & Interfaces*, Vol.95, p.104033, 2025. https://doi.org/10.1016/j.csi.2025.104033.

[27] H.S.Munir, S. Ren, M. Mustafa, C.N. Siddique and S. Qayyum, "Attention based GRU-LSTM for software defect prediction." *Plos one*, Vol.16, no. 3, p.e0247444, 2021. https://doi.org/10.1371/journal.pone.0247444.

[28] F.Yang, Y. Huang, H. Xu, P. Xiao and W. Zheng, "Fine-Grained Software Defect Prediction Based on the Method-Call Sequence.",*Computational intelligence and neuroscience,* no. 1, p. 4311548, 2022. https://doi.org/10.1155/2022/4311548.

[29] A. Boulkroune, F. Zouari and A. Boubellouta, “Adaptive fuzzy control for practical fixed-time synchronization of fractional-order chaotic systems.”, *Journal of Vibration and Control*, p.10775463251320258, 2025. https://doi.org/10.1177/10775463251320258.

[30] A. Boulkroune, S. Hamel, F. Zouari, A. Boukabou and A. Ibeas, “Output-Feedback Controller Based Projective Lag-Synchronization of Uncertain Chaotic Systems in the Presence of Input Nonlinearities”, *Mathematical Problems in Engineering*, no. 1, p.8045803, 2017. https://doi.org/10.1155/2017/8045803

[31] F. Zouari, K.B. Saad and M. Benrejeb, “Robust neural adaptive control for a class of uncertain nonlinear complex dynamical multivariable systems”, *International Review on Modelling and Simulations*, Vol.5, no. 5, pp. 2075-2103, 2012.

[32] F. Zouari, K.B. Saad and M. Benrejeb, “Adaptive backstepping control for a class of uncertain single input single output nonlinear systems”, *In 10th International Multi-Conferences on Systems, Signals & Devices 2013 (SSD13)*, pp. 1-6. IEEE, 2013. doi: 10.1109/SSD.2013.6564134.

[33] F. Zouari, K.B. Saad and M. Benrejeb, “Adaptive backstepping control for a single-link flexible robot manipulator driven DC motor”, *In 2013 International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 864-871. IEEE, 2013. doi: 10.1109/CoDIT.2013.6689656.

[34] G. Rigatos, M. Abbaszadeh, B. Sari, P. Siano, G. Cuccurullo and F. Zouari, "Nonlinear optimal control for a gas compressor driven by an induction motor.", *Results in Control and Optimization*, Vol.11, p.100226, 2023. https://doi.org/10.1016/j.rico.2023.100226.