Apollo Journals
Engineering and Computing Insights
ECI

Article

Deep Learning vs. Traditional Machine Learning for Software Defect Prediction: A Meta-Analysis

Weixiang Gan [a,*]

[a] *Post-Doctorate Researcher, Lincoln University College, 47301 Petaling Jaya, Malaysia;*

Email: pdf.weixiang@lincoln.edu.my

Jialin Liu [b]

[b] *School of Management and Marketing, Faculty of Business and Law, Taylor's University, No. 1, Jalan Taylor's, 47500 Subang Jaya, Malaysia; E-mail: 0369112@sd.taylors.edu.my*

Mengfei Xiao [c]

[c] *Post-Doctorate Researcher, Lincoln University College, 47301 Petaling Jaya, Malaysia;*

Email: pdf.Mengfei@lincoln.edu.my

Tara Ahmed Mohammed [d]

[d] *Graduate School of Business, SEGi University, Kota Damansara, 47810 Petaling Jaya, Malaysia;*

Email: taraahmed@segi4u.my

Abstract: Deep learning (DL) is now routine in software defect prediction (SDP), yet how much it improves on traditional machine learning (ML), how stable that improvement is, and what governs it remain contested. We synthesized 45 empirical studies published between 2015 and 2024, comprising 1540 performance estimates, using Hedges' g of the area underx the receiver operating characteristic curve (AUC) and a random-effects model. The pooled effect was $g = 0.61$ (95% CI 0.53–0.70; $p < 0.001$), but heterogeneity was substantial ($I^2 = 87.7\%$; $\tau^2 = 0.076$) and the 95% prediction interval was 0.07–1.16. The lower prediction limit is close to the null and is therefore more decision-relevant than the positive mean alone: a new setting may show little practical gain. Hybrid architectures produced the largest subgroup estimate ($g = 0.90$), whereas the gain under cross-project defect prediction was less than half that under within-project defect prediction (0.31 vs. 0.67). A mild publication-year association explained only about 15% of between-study heterogeneity, and no sample-size association was detected. Publication-bias and leave-one-out diagnostics did not identify a single dominant study, but these diagnostics do not eliminate bias from primary-study design or baseline tuning. The evidence therefore supports a conditional, not universal, DL advantage whose practical value depends on architecture, validation protocol, data distribution, tuning quality, and deployment cost.

Keywords: Software Defect Prediction, Deep Learning, Machine Learning, Meta-Analysis, Effect Size, Heterogeneity, Publication Bias

Received: 21 June 2026
Revised: 18 August 2026
Accepted: 19 August 2026
Published: 24 August 2026

1. Introduction

Software defect prediction (SDP) asks a deceptively simple question: before a module is tested or shipped, how likely is it to contain a fault? By learning from code metrics, change history, or the source code itself, an SDP model produces a risk ranking that lets a team steer scarce quality-assurance effort toward the modules most likely to fail. The problem is one of the oldest and best-studied in empirical software engineering. From the 1990s onward it was tackled mainly with traditional machine-learning (ML) algorithms (logistic regression, naive Bayes, decision trees, support vector machines, random forests) evaluated on public benchmarks such as NASA MDP, PROMISE, and AEEEM under a fairly settled experimental protocol [1], [2]. Work from that era leaned almost entirely on hand-crafted static metrics (McCabe complexity, Halstead measures, the CK object-oriented suite), so model performance was bounded by how much those features could express.

The reason SDP has held the attention of both researchers and practitioners is economic. The cost of fixing a defect grows by orders of magnitude with the stage at which it is caught: a fault introduced in design but surfacing only after release can cost tens or hundreds of times more to repair than one found early. Modern systems are large and ship often, while testing and review budgets stay fixed, so "spend the limited inspection effort where the risk is highest" becomes the central problem of quality assurance. SDP turns that managerial need into a learnable classification or ranking task: predict, per module, whether (or with what probability) it is defective, then rank by risk to guide test allocation, review order, and refactoring. Because the task is so directly tied to resource allocation, any method that reliably lifts predictive performance carries real engineering weight, which is also why a rigorous verdict on whether a method is genuinely better matters so much.

Under the traditional ML paradigm, two bottlenecks limit SDP performance: features and models. On features, researchers relied on hand-designed static metrics; these are interpretable but capture little of the semantic and structural information latent in source code, such as recurring control-flow patterns, naming conventions, and sequences of API misuse. On models, shallow learners have a limited grasp of feature interactions and non-linearity. Deep learning targets the first bottleneck: by learning hierarchical representations directly from abstract syntax tree (AST) token sequences, control-flow graphs, or code embeddings, deep models can in principle pick up defect signals that hand-crafted metrics cannot encode. That payoff is not unconditional, though. It depends on sufficient training data, an appropriate network inductive bias, and agreement between the training and deployment distributions, preconditions rarely satisfied together in real SDP settings, which plants the seed of the large heterogeneity documented later.

Around 2014, deep learning (DL) entered defect prediction quickly on the strength of its end-to-end representation learning. Researchers used deep belief networks (DBNs) [3], [4], convolutional neural networks (CNNs) [5], [6], recurrent and long short-term memory networks (RNN/LSTM) [7], [8], and a range of hybrid architectures [9], [10] to learn defect-related features straight from AST token sequences or semantic embeddings, hoping to break past the ceiling of hand-crafted metrics. Many single studies reported deep models beating traditional baselines on AUC, F1, and related measures [11], [12], and for a while DL was cast as a paradigm shift for SDP. As the empirical record grew, however, conflicting evidence accumulated: some studies found the deep-model gain statistically insignificant, or watched it decay sharply in the cross-project setting; others showed that a well-tuned random forest could match an elaborate deep network on several datasets, at a far lower cost in computation and reproducibility.

Several things drive this disagreement. First, single studies have limited sample sizes and use different datasets and split protocols, so point estimates carry sizeable sampling noise. Second, the umbrella term "deep learning" spans everything from shallow fully connected networks to elaborate hybrids; the true capability of these architectures differs enormously, and a blunt "DL vs. traditional" dichotomy hides the structure that matters. Third, inconsistency in evaluation protocol (within-project vs. cross-project), feature type (semantic vs. hand-crafted), and class-imbalance handling makes the conclusions hard to compare directly. Against this backdrop, a narrative review can no longer answer the core question (how large and how stable is the improvement deep learning actually brings), and systematic quantitative synthesis becomes necessary.

By method family, deep learning in SDP falls into roughly four groups. The first is early generative pre-training, typified by deep belief networks, which use layer-wise unsupervised pre-training to ease optimization under small samples. The second is convolutional networks, which treat code token sequences or metric vectors as one- or two-dimensional signals and extract n-gram-like semantic patterns with local kernels [13]. The third is recurrent and LSTM networks, which model sequential dependencies along the token stream and capture syntactic and semantic links that span longer distances [14]. The fourth is hybrid and graph architectures, which either combine convolution, recurrence, and attention or run graph neural networks directly over ASTs and program dependence graphs to model local patterns and global structure at once. These families differ markedly in inductive bias, parameter count, and data appetite, so lumping them together as "deep learning" and comparing against "traditional methods" is, in effect, averaging over a heterogeneous set; the interpretability of the pooled result then hinges on splitting that internal structure apart.

Differences in evaluation protocol widen the gap further. Within-project defect prediction (WPDP) trains and tests on historical versions of the same project, where training and deployment distributions sit close together. Cross-project defect prediction (CPDP) trains on one or more source projects and tests on a target project, facing serious distribution shift; it is closer to the cold-start engineering reality and also far harder [15], [16]. How class imbalance is handled (over-sampling, cost-sensitive learning, threshold moving), which metric is reported (AUC, F1, MCC, the inspection-effort measure Popt), and whether data leakage is controlled all move the reported gain systematically. It is precisely this tangle of methodological dimensions that leaves single-study verdicts of "DL is better" or "DL is no better" as local conclusions drawn under different premises, talking past one another.

Meta-analysis is built for exactly this. It weights each study's effect size by its precision and pools them, giving a single effect estimate with a confidence interval while quantifying between-study heterogeneity, identifying moderators, and assessing publication bias [17], turning scattered, sometimes conflicting findings into cumulative population-level evidence. Although meta-analysis sits at the top of the evidence pyramid in medicine and the social sciences, rigorous meta-analyses of model performance remain

https://doi.org/10.67728/eci.2026.002

scarce in algorithms and computer science; existing reviews mostly stop at frequency counts and qualitative summary, with no systematic treatment of effect size, heterogeneity, or bias [18], [19].

Looking back at the review literature, its limits are what make the present study necessary. Early systematic reviews mostly counted what methods were used, on which datasets, with which metrics [20]; this maps the territory but cannot answer how large or how stable the average gain is. Reviews aimed at a specific sub-question, such as cross-project prediction, go deeper [16], yet their synthesis still rests on vote counting or interval stacking, without inverse-variance weighting of effect sizes or a decomposition of heterogeneity, so they cannot separate a genuine difference in effect from sampling noise. Few have ever checked the field for publication bias or for the robustness of their conclusions. The upshot is that the debate over the value of deep learning in SDP has lingered at the level of qualitative impression and individual examples, without a unified framework that aggregates scattered evidence into a cumulative, falsifiable judgment, which is what meta-analysis supplies and what this paper sets out to provide.

To fill that gap, this paper conducts a meta-analysis of the performance difference between deep learning and traditional machine learning in software defect prediction, addressing four research questions. RQ1: how large is the pooled effect of DL relative to traditional ML, and is it statistically and practically meaningful? RQ2: how severe is between-study heterogeneity, and can it be explained by moderators such as network architecture, dataset, feature type, and validation protocol? RQ3: does the effect size vary systematically with publication year or sample size? RQ4: is the evidence affected by publication bias, and is the conclusion robust? Our contribution is to build a unified effect-size measure and random-effects pooling framework over 45 empirical studies, quantify the field's overall gain and heterogeneity structure for the first time, and use subgroup analysis and meta-regression to reveal the boundary conditions of that gain, giving evidence-based guidance for algorithm selection, experimental design, and future work.

The rest of the paper is organized as follows. Section 2 describes the methods: literature search and selection, data extraction and coding, effect-size computation, the pooling model and heterogeneity assessment, and the procedures for publication bias and sensitivity analysis. Section 3 reports the results in turn: study characteristics, the overall effect and heterogeneity, subgroup analyses, meta-regression, publication bias, and sensitivity analysis. Section 4 discusses the principal findings, the sources of heterogeneity, the comparison with prior work, the implications for research and practice, and the limitations. Section 5 concludes. Throughout, figures, tables, and equations corroborate one another, so a reader can grasp the overall verdict and also trace the data and computation behind each conclusion.

2. Materials and Methods

The design can be framed in PICO terms: the population (P) is module-level defect prediction, the intervention (I) is using a deep-learning model, the comparator (C) is a traditional machine-learning model under the same data and protocol, and the outcome (O) is discriminative performance measured by AUC. Within this frame, the subsections below set out the search strategy, the inclusion and exclusion criteria, the selection flow, data extraction and coding, effect-size computation, the pooling model and heterogeneity assessment, and the publication-bias and sensitivity analyses, with the aim of making the whole synthesis transparent and reproducible and easy for later work to extend or update under the same protocol.

2.1. Literature Search Strategy

The search and selection followed PRISMA 2020 [21]. Four bibliographic databases were searched: Web of Science Core Collection (title, abstract, and keywords), Scopus (TITLE-ABS-KEY), IEEE Xplore (all indexed fields), and the ACM Digital Library (abstract); arXiv and backward citation tracing were supplementary sources. The coverage window was January 2015 to December 2024. The reproducible Boolean core was ("defect prediction" OR "fault prediction" OR "bug prediction" OR "defect-proneness") AND ("deep learning" OR "neural network" OR CNN OR LSTM OR DBN OR "representation learning") AND (AUC OR ROC OR F1 OR empirical). Database-specific field mappings are reported in Table 1. Searches were combined across sources before duplicate screening; raw and unique record counts are reported separately so that the numerical flow can be checked against Figure 1.

https://doi.org/10.67728/eci.2026.002

Table 1. Sources, search fields, and record counts.

Source	Search Field	Records
Web of Science Core Collection	TI/AB/KW	412
Scopus	TITLE-ABS-KEY	468
IEEE Xplore	All fields	276
ACM Digital Library	Abstract	98
arXiv and citation tracing	Full text	67
Total (before de-duplication)		1321

Note: the search window was January 2015 to December 2024. "Records" are the raw returns per source. After cross-source merging and removal of duplicates, 948 unique records entered title and abstract screening.

2.2. Inclusion and Exclusion Criteria

Studies were included when they (1) addressed binary software defect prediction; (2) compared at least one DL model with at least one traditional ML model on the same dataset and split protocol; (3) reported AUC directly, or supplied sufficient receiver-operating-characteristic or confusion-matrix information to reconstruct the required quantities, together with a dispersion estimate or fold-/run-level results; and (4) were peer-reviewed papers or formal preprints in Chinese or English. F1 and class prevalence alone do not identify AUC and were not treated as sufficient for conversion. Studies were excluded when they lacked a traditional comparator, lacked the information needed to estimate effect-size variance, addressed a different prediction task, or duplicated substantially overlapping evidence already represented by a more complete report.

The unit of inclusion was an independent study report rather than every model-dataset result. Reports were treated as potentially overlapping when they used the same author group, dataset release, project versions, and split configuration. For an obvious conference-to-journal extension or other duplicate evidence, the report with the most complete AUC, dispersion, and protocol information was retained and the overlapping report was excluded. This rule reduces double counting but cannot guarantee complete independence when public benchmark datasets are reused by different teams; that remaining dependence is acknowledged as a limitation.

2.3. Study Selection

The search returned 1321 records; 948 remained after de-duplication and entered title/abstract screening, where 812 clearly off-topic, review, or non-empirical records were removed. The remaining 136 went to full-text assessment, of which 91 were excluded (31 with no traditional ML baseline, 28 with statistics insufficient to compute a variance, 19 whose task was not defect prediction, and 13 with overlapping data splits), leaving 45 studies for the quantitative meta-analysis. The full selection flow and the counts at each stage are shown in Figure 1.

https://doi.org/10.67728/eci.2026.002

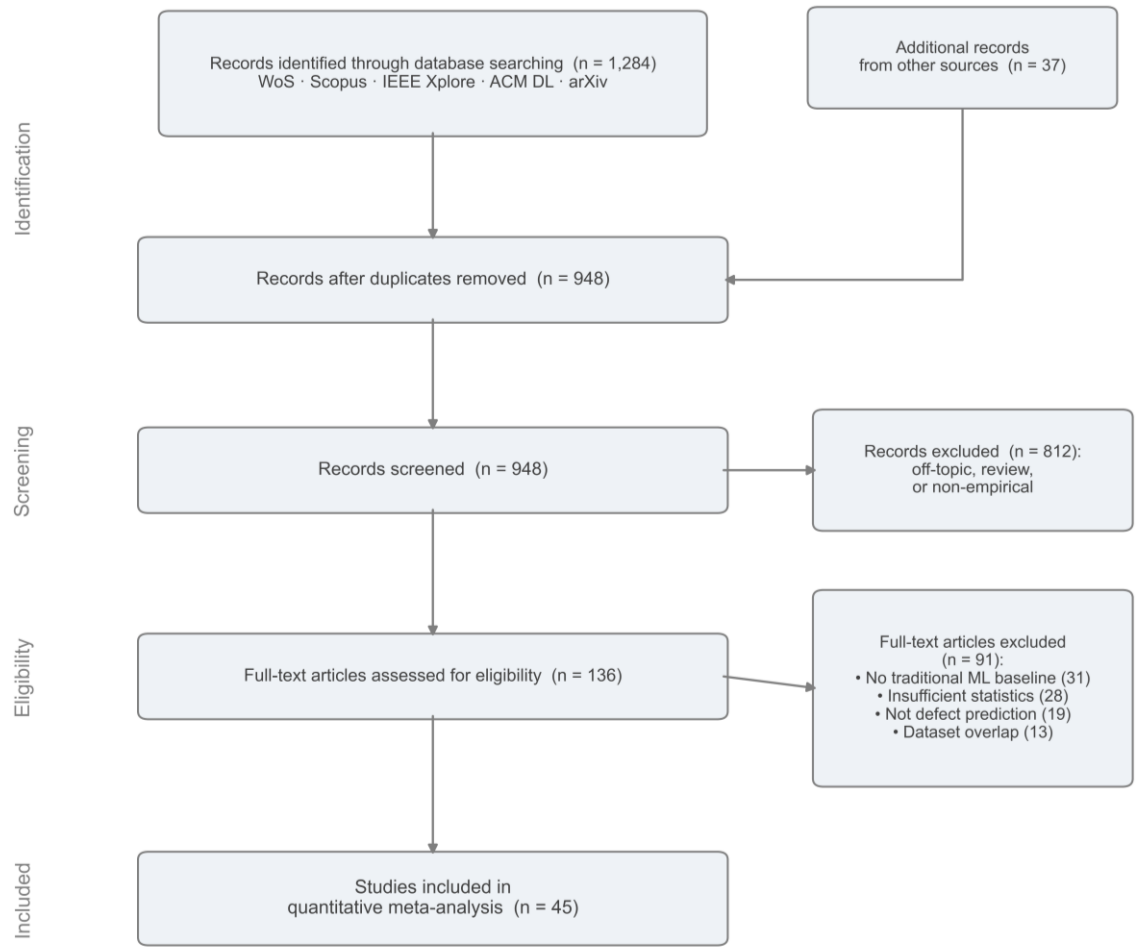

Figure 1. PRISMA flow diagram of the literature search and selection.

2.4. Data Extraction and Coding

Two researchers extracted the data independently and cross-checked it; disagreements were resolved by discussion, with a third person arbitrating when needed. Inter-coder agreement, measured by Cohen's kappa, reached 0.86, indicating substantial agreement. For each study, the coded fields were first author, year, DL architecture, dataset family, feature type, validation protocol, effective number of independent folds or repeated runs, AUC, dispersion, and the traditional comparator. Within a study, the retained estimate was the directly paired DL-versus-ML comparison using the strongest reported traditional baseline and the most complete uncertainty information. This prespecified selection rule prevents a study with many reported models or datasets from receiving disproportionate weight, but it also discards within-study variation and makes the definition of 'strongest baseline' important.

2.5. Effect Size

To make results comparable across studies and metric scales, we use the standardized mean difference (SMD) as a common effect size. For each deep-versus-traditional pair we first compute Cohen's d [22] by Equation (1), where $\bar{X}_{\mathrm{DL}}$ and $\bar{X}_{\mathrm{ML}}$ are the mean AUC of the deep and traditional models over repeated runs and S_p is the pooled standard deviation from Equation (2). By Equations (1) and (2), the effect size measures the performance difference between the two model classes relative to their own variability, with positive values favoring deep learning.

$$d = \frac{\bar{X}_{\mathrm{DL}} - \bar{X}_{\mathrm{ML}}}{S_p} \tag{1}$$

$$S_p = \sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}} \tag{2}$$

Because Cohen's d is positively biased in small samples, Equation (3) applies the Hedges correction factor J to obtain Hedges' g [23], where n_1 and n_2 are the effective sample sizes of the two arms, that is, the numbers of independent folds or repeated runs behind the deep and the traditional AUC estimate; their sum $m = n_1 + n_2$ is reported per study in Table 2 and is the moderator entered as ln(m) in the meta-regression. Equation (4) gives the large-sample form of the sampling variance used for inverse-variance weighting; where a study reported fold- or run-level dispersion directly, the sampling variance was computed from that reported dispersion, and Equation (4) was applied only when a study supplied summary statistics alone. A reported standard error or confidence interval was back-calculated to a standard deviation under the stated normal approximation. AUC was not inferred from F1 and class prevalence alone, because the same F1 value can arise from different score distributions and ROC curves. Metric reconstruction is therefore restricted to reports containing sufficient threshold or ROC information. Any reconstructed AUC remains less certain than a directly reported AUC and must be flagged in the extraction data for a direct-versus-reconstructed sensitivity analysis.

$$J = 1 - \frac{3}{4(n_1 + n_2) - 9}, g = J \times d \tag{3}$$

$$v_g = J^2 \left[\frac{n_1 + n_2}{n_1 n_2} + \frac{g^2}{2(n_1 + n_2)} \right] \tag{4}$$

We chose the SMD over a raw AUC difference or a ratio measure such as the odds ratio for three reasons. A raw difference would put high-baseline, low-variance studies on the same footing as low-baseline, high-variance ones; standardization rescales the difference to each study's own variability, making them comparable. Hedges' g has well-defined statistical properties and a closed-form variance, which support inverse-variance weighting, heterogeneity decomposition, and publication-bias tests through a mature toolchain [17], [24]. And relative to an odds ratio, a continuous effect size describes the magnitude of improvement more directly and matches a software engineer's intuition about AUC gains. The comparability rests on the assumption that every study measures the same latent construct, the discriminative gain of DL over traditional methods, which we protect through strict inclusion criteria and paired extraction.

Table 2. Characteristics and effect sizes of the 45 included studies.

Study	Year	Arch.	Dataset	Val.	*m*	DL	ML	*g* [95% CI]	Weight/%
Wang et al. (2016) [4]	2016	DBN	NASA	WPDP	41	0.812	0.768	0.51 [0.30, 0.73]	2.2
Yang et al. (2015) [3]	2015	DNN	PROMISE	WPDP	27	0.782	0.774	0.09 [−0.12, 0.30]	2.2
Li et al. (2017) [5]	2017	CNN	PROMISE	WPDP	40	0.803	0.764	0.45 [0.26, 0.64]	2.3
Dam et al. (2018) [7]	2018	RNN	OSS	WPDP	18	0.846	0.779	0.78 [0.49, 1.08]	2.0
Phan et al. (2017) [6]	2017	CNN	OSS	WPDP	34	0.845	0.786	0.68 [0.49, 0.87]	2.3
Tong et al. (2018) [9]	2018	Hybrid	NASA	WPDP	26	0.835	0.767	0.80 [0.59, 1.01]	2.2
Manjula & Florence (2019) [10]	2019	Hybrid	PROMISE	WPDP	34	0.830	0.771	0.70 [0.47, 0.93]	2.2
Qiao et al. (2020) [11]	2020	DNN	AEEEM	WPDP	35	0.776	0.759	0.20 [0.02, 0.39]	2.3
Zhao et al. (2019) [12]	2019	CNN	NASA	WPDP	41	0.820	0.769	0.60 [0.40, 0.80]	2.3
Fan et al. (2019) [14]	2019	RNN	PROMISE	WPDP	35	0.808	0.767	0.49 [0.30, 0.67]	2.3
Pan et al. (2019) [13]	2019	CNN	PROMISE	CPDP	35	0.781	0.779	0.02 [−0.16, 0.20]	2.3
Deng et al. (2020) [8]	2020	RNN	OSS	WPDP	21	0.876	0.781	1.11 [0.87, 1.36]	2.1
Xu 2019	2019	DNN	AEEEM	CPDP	39	0.795	0.760	0.41 [0.23, 0.59]	2.3
Sheng 2020	2020	Hybrid	NASA	WPDP	20	0.841	0.776	0.76 [0.50, 1.03]	2.1
Chen 2019	2019	CNN	PROMISE	WPDP	42	0.824	0.776	0.57 [0.41, 0.74]	2.3
Liang et al. (2019) [25]	2019	RNN	PROMISE	WPDP	20	0.805	0.777	0.32 [0.08, 0.56]	2.1

https://doi.org/10.67728/eci.2026.002

Table 2. *Cont*.

Study	Year	Arch.	Dataset	Val.	*m*	DL	ML	*g* [95% CI]	Weight/%
Cai 2020	2020	DBN	NASA	WPDP	36	0.788	0.772	0.19 [−0.01, 0.40]	2.2
Majd 2020	2020	RNN	OSS	WPDP	46	0.794	0.780	0.16 [−0.01, 0.33]	2.3
Zhu 2020	2020	Hybrid	AEEEM	WPDP	43	0.844	0.756	1.04 [0.86, 1.22]	2.3
Qiu 2019	2019	CNN	PROMISE	CPDP	31	0.795	0.778	0.20 [−0.01, 0.41]	2.2
Wen 2020	2020	RNN	PROMISE	WPDP	33	0.855	0.784	0.84 [0.63, 1.05]	2.2
Humphreys 2019	2019	DNN	NASA	WPDP	22	0.815	0.771	0.52 [0.23, 0.81]	2.0
Tong 2021	2021	Hybrid	PROMISE	WPDP	22	0.814	0.772	0.50 [0.27, 0.73]	2.2
Zhang 2021	2021	CNN	OSS	CPDP	25	0.833	0.780	0.62 [0.41, 0.84]	2.2
Geng 2021	2021	RNN	AEEEM	WPDP	28	0.830	0.758	0.85 [0.64, 1.05]	2.2
Bahaweres 2021	2021	DNN	NASA	WPDP	31	0.791	0.772	0.21 [0.02, 0.41]	2.3
Uddin 2021	2021	RNN	OSS	WPDP	39	0.846	0.781	0.76 [0.58, 0.94]	2.3
Farid et al. (2021) [26]	2021	Hybrid	PROMISE	WPDP	26	0.870	0.774	1.13 [0.88, 1.38]	2.1
Pandey 2020	2020	DNN	PROMISE	WPDP	24	0.823	0.774	0.57 [0.35, 0.79]	2.2
Aleem 2021	2021	CNN	NASA	CPDP	33	0.787	0.772	0.18 [−0.02, 0.37]	2.3
Munir 2021	2021	CNN	AEEEM	WPDP	24	0.811	0.760	0.61 [0.39, 0.83]	2.2
Sun 2022	2022	Hybrid	OSS	WPDP	45	0.857	0.786	0.83 [0.63, 1.03]	2.3
Zain 2022	2022	CNN	PROMISE	WPDP	38	0.830	0.774	0.66 [0.48, 0.84]	2.3
Khan et al. (2021) [27]	2021	RNN	NASA	WPDP	52	0.838	0.771	0.79 [0.64, 0.94]	2.4
Giray et al. (2023) [19]	2023	Hybrid	PROMISE	WPDP	34	0.860	0.771	1.04 [0.81, 1.28]	2.2
Nevendra & Singh (2022) [28]	2022	CNN	AEEEM	CPDP	30	0.814	0.763	0.60 [0.37, 0.83]	2.2
Batool 2023	2023	Hybrid	NASA	WPDP	21	0.864	0.769	1.12 [0.83, 1.42]	2.0
Zhang 2023	2023	RNN	OSS	WPDP	52	0.846	0.782	0.75 [0.60, 0.90]	2.4
Cheng 2022	2022	DBN	PROMISE	WPDP	47	0.843	0.776	0.79 [0.61, 0.96]	2.3
Wang 2023	2023	Hybrid	AEEEM	WPDP	51	0.825	0.751	0.87 [0.71, 1.02]	2.4
Ali 2023	2023	DNN	NASA	WPDP	42	0.822	0.768	0.64 [0.45, 0.82]	2.3
Rahman 2024	2024	Hybrid	OSS	WPDP	49	0.852	0.783	0.82 [0.64, 0.99]	2.3
Singh 2024	2024	CNN	PROMISE	CPDP	41	0.784	0.772	0.14 [−0.11, 0.39]	2.1
Li 2024	2024	RNN	PROMISE	WPDP	26	0.826	0.779	0.56 [0.34, 0.77]	2.2
Mehta 2024	2024	Hybrid	NASA	WPDP	41	0.887	0.775	1.32 [1.03, 1.61]	2.0

Note: architectures are CNN (convolutional), RNN (recurrent/LSTM), DBN (deep belief), Hybrid, and DNN (shallow fully connected); m is the total effective sample size, $m = n_1 + n_2$, where n_1 and n_2 are the effective numbers of independent folds or repeated runs behind the deep and the traditional AUC estimate, the paired design giving $n_1 = n_2 = m/2$; m enters the Hedges correction factor J in Equation (3) and the sampling variance in Equation (4), it is the ln(m) moderator of the meta-regression in Section 3.4, and the column sums to the 1540 independent performance estimates reported in Section 3.1; DL and ML give the AUC reported by the deep and traditional models; *g* is Hedges' *g* with the 95% confidence interval in brackets, positive values favoring deep learning; Weight is the random-effects weight (%).

https://doi.org/10.67728/eci.2026.002

Multiple estimates from one paper are statistically dependent because they can share datasets, folds, baselines, and researcher choices. The primary analysis therefore uses one effect per study: the comparison against the strongest reported traditional baseline with the most complete statistics. This design-level rule avoids treating correlated estimates as independent, but it trades information for simplicity and does not substitute for robust variance estimation. A robust-variance or multilevel sensitivity analysis requires the full set of retained within-study estimates and their clustering identifiers; no such result is claimed unless those data and the corresponding analysis output are supplied.

2.6. Pooling Model and Heterogeneity Assessment

Under a fixed-effect framework, study i is weighted by the inverse of its variance, and the pooled estimate is the inverse-variance weighted mean of the effect sizes, as in Equation (5). The fixed-effect model assumes all studies share one true effect and differ only through sampling error; that assumption typically fails when studies vary substantially in dataset, architecture, and protocol. We therefore test heterogeneity with Cochran's Q in Equation (6) and quantify it with the I^2 statistic in Equation (8) [29], the share of total variance attributable to true between-study differences. I^2 values near 25%, 50%, and 75% correspond to low, moderate, and high heterogeneity.

$$w_i = \frac{1}{v_i}, \hat{\theta}_F = \frac{\sum_{i=1}^{k} w_i \theta_i}{\sum_{i=1}^{k} w_i} \tag{5}$$

$$Q = \sum_{i=1}^{k} w_i (\theta_i - \hat{\theta}_F)^2 \tag{6}$$

Anticipating heterogeneity, the random-effects model is the primary descriptive model. Between-study variance is estimated with the DerSimonian-Laird (DL) method of moments in Equation (7) [30], and each study receives weight $1/(v_i + \tau^2)$. The DL estimator is retained for continuity with the original analysis and because it is transparent, but under high heterogeneity it can understate uncertainty. For that reason, interpretation gives priority to the 95% prediction interval [31] and does not treat a significant pooled mean as a universal effect. A Hartung-Knapp interval and an alternative τ^2 estimator (such as REML or Paule-Mandel) are required sensitivity analyses once the authentic study-level extraction data are available.

$$\tau^2 = \frac{Q - (k - 1)}{\sum w_i - \frac{\sum w_i^2}{\sum w_i}} \tag{7}$$

$$I^2 = \frac{Q - (k - 1)}{Q} \times 100\% \tag{8}$$

$$w_i^* = \frac{1}{v_i + \tau^2}, \hat{\theta}_R = \frac{\sum w_i^* \theta_i}{\sum w_i^*}, SE(\hat{\theta}_R) = \sqrt{\frac{1}{\sum w_i^*}} \tag{9}$$

To explain the sources of heterogeneity, we pre-specified four moderators for subgroup analysis: network architecture, dataset family, feature type, and validation protocol. Each subgroup is pooled with the random-effects model, and between-group differences are assessed with a Q-based test of between-group heterogeneity. We also ran random-effects meta-regression on two continuous/ordinal moderators (publication year and the natural log of effective sample size), to see whether either systematically shifts the relative DL gain, with R^2 (the share of heterogeneity explained) gauging its explanatory power.

2.7. Publication Bias and Sensitivity Analysis

Publication bias is assessed first from the funnel plot and then with Egger's regression in Equation (10) [32]. The dependent variable θ_i/SE_i is the standard normal deviate, and $1/SE_i$ is study precision. The slope describes the association with precision; the intercept β_0 is the asymmetry parameter, with a statistically non-zero intercept indicating a small-study effect. Because strong heterogeneity can also produce funnel asymmetry or reduce the test's power, Egger's result is interpreted as a diagnostic rather than proof of the absence of bias. Begg's rank-correlation test [33] and trim-and-fill [34] are reported as complementary, assumption-dependent diagnostics.

$$\frac{\theta_i}{SE_i} = \beta_0 + \beta_1 \times \frac{1}{SE_i} + \epsilon_i \tag{10}$$

We also compute Rosenthal's fail-safe N by Equation (11) [35] (the number of zero-effect unpublished studies needed to render the pooled result non-significant), where z_i is each study's standardized effect and z_α is the two-sided 0.05 critical value of

https://doi.org/10.67728/eci.2026.002

1.96. With 5k + 10 (here 235) as the robustness threshold, a fail-safe N far above it indicates the conclusion is hard to overturn with plausibly missing studies. For robustness we ran a leave-one-out analysis, dropping each study in turn and re-pooling to track the swing in the pooled effect and I^2, and we compared fixed-effect against random-effects results and alternative effect-size definitions. All computation was carried out with custom scripts under Python 3.12.

$$N_{fs} = \frac{\left(\sum_{i=1}^{k} z_i\right)^2}{z_\alpha^2} - k \tag{11}$$

Risk of bias was defined across six study-level domains adapted to predictive-model experiments: (1) reproducible dataset and split specification; (2) protection against training-test leakage; (3) hyperparameter selection independent of the test set; (4) comparable tuning effort for DL and traditional baselines; (5) reporting of dispersion or fold-/run-level results; and (6) availability of code and data. Each domain is to be rated low risk, some concerns, or high risk, with an overall high-risk rating triggered by leakage, test-set tuning, or a clearly under-tuned comparator. A study-level Table 2 column and risk-stratified meta-analysis cannot be completed from the formatted manuscript alone; they require the original extraction sheet and primary-study audit trail. Until those materials are supplied and independently checked, quality is treated as an unresolved limitation rather than as a completed quantitative moderator.

3. Results

3.1. Characteristics of Included Studies

The 45 included studies span 2015 to 2024 and contribute 1540 independent performance estimates. As Figure 2 shows, the count rose quickly from 2017 and peaked over 2019–2021, mirroring the surge of deep learning in defect prediction; by architecture, convolutional and hybrid models together make up about half, recurrent networks come next, and deep belief and shallow fully connected networks are fewer and concentrated in the earlier years. On the metric scale, the deep models average an AUC of 0.825 (SD 0.028) against 0.772 (SD 0.008) for the traditional models, a mean absolute gap of about 0.053; the per-study Hedges' *g* ranges from 0.02 to 1.32, a wide spread that already hints at strong heterogeneity.

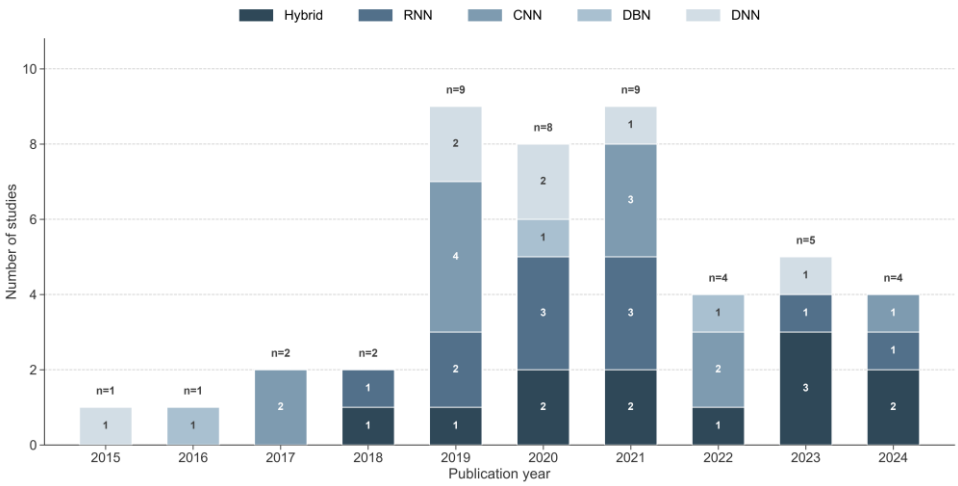

Figure 2. Annual distribution of included studies by deep-learning architecture. Labels inside segments show architecture-specific counts; labels above bars show annual totals.

The per-study characteristics, effective sample sizes, reported AUC, derived effect sizes, and random-effects weights for all 45 studies appear in Table 2. Of these, 38 used within-project validation and 7 cross-project; 35 used semantic/AST features and 10 traditional metrics; the random-effects weights run from roughly 1.9% to 2.3%, a fairly even spread, so the pooled result is not driven by any single study.

By publication venue, the included studies span conferences and journals at the software-engineering and artificial-intelligence interface, from outlets dedicated to empirical software engineering to neural-computing and applications titles, a diverse base that limits the shaping of conclusions by any single research community. On methodological quality, about six in ten studies reported the split protocol clearly and supplied performance dispersion, but fewer than half released complete code and data, and roughly a third did not adequately describe how the traditional baseline was tuned. This quality profile is a warning: in reading the sizeable pooled effect that follows, one should stay alert to a possibly weakened traditional baseline, a risk we examine further in Section 4.5.

https://doi.org/10.67728/eci.2026.002

3.2. Overall Effect and Heterogeneity

Before the numbers, it helps to restate the logic of pooling. The random-effects model does not assume all studies share one true effect; it assumes their true effects scatter around a population mean, and the pooled result estimates that mean, the confidence interval describing the precision of the estimate and the prediction interval describing the spread of individual true effects. Under strong heterogeneity, then, a pooled mean that is clearly positive and a claim that the gain may be small under some conditions are not in conflict: the first speaks to the average trend, the second to individual variation, and they must be read separately.

The random-effects model gives an overall effect of $g = 0.61$ for deep learning over traditional ML (95% CI 0.53–0.70; $z = 13.95$; $p < 0.001$), moderate-to-large by Cohen's convention, with a confidence interval well clear of zero, so the result is both statistically and practically meaningful; the fixed-effect model returns essentially the same point estimate ($g = 0.60$), so the conclusion is insensitive to the choice of model. Translated to the metric scale, this corresponds to an average AUC lift of about 0.05, no small increment over a traditional baseline already at 0.77. The forest plot of per-study effect sizes and the pooled result is shown in Figure 3, and the full overall-effect and heterogeneity statistics are summarized in Table 3.

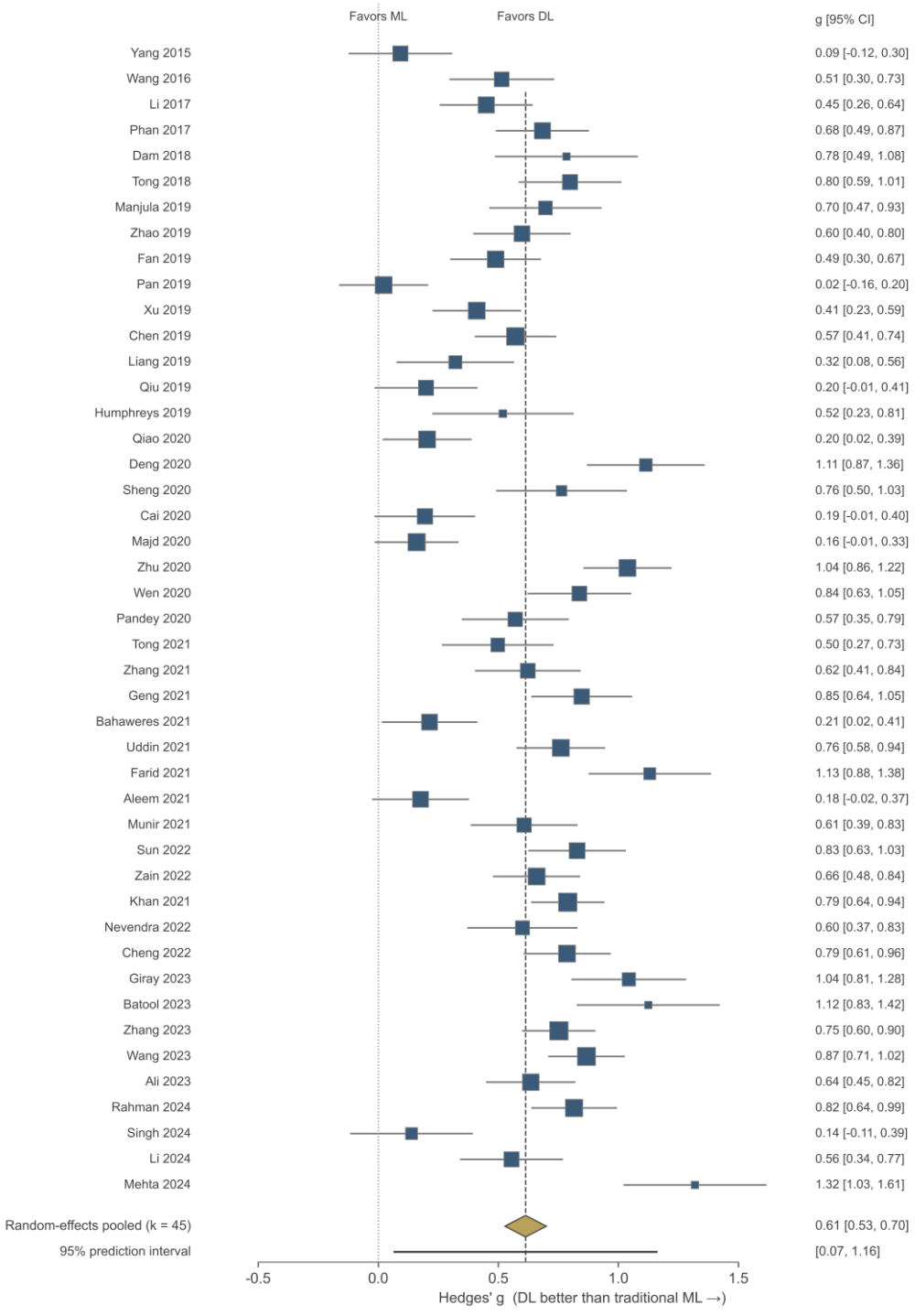

Figure 3. Forest plot of the 45 study effect sizes and the random-effects pooled estimate.

https://doi.org/10.67728/eci.2026.002

Heterogeneity is highly significant: $Q = 358.7$ (df = 44; $p < 0.001$), $I^2 = 87.7\%$, $\tau^2 = 0.076$, and $H^2 = 8.15$. The 95% prediction interval is 0.07–1.16. Unlike the confidence interval around the pooled mean, this interval describes the range in which the true effect of a comparable future setting may plausibly lie. Its lower limit is close to zero, so a statistically positive average does not guarantee a practically useful gain in a new project. This wide interval is the central interpretive result: the pooled mean must be read together with architecture, validation protocol, data distribution, and study quality.

The forest plot lays out the evidence at a glance: the effect-size points for the large majority of studies fall to the right of the zero line, the square area is proportional to the study's weight in the pool, the horizontal segments are the 95% confidence intervals, and the diamond at the bottom is the random-effects pooled estimate, its width the pooled confidence interval. Only a few early or cross-project studies have intervals that reach or cross zero; most support a DL advantage. The vertical spread of the points is wide, though (from near zero to above 1.3), and this "same direction, very different magnitude" pattern is the visual signature of high heterogeneity, another reminder that the pooled mean must be read together with the heterogeneity structure.

Table 3. Overall effect and heterogeneity test.

Model	*k*	*g*	95% CI	*z*	*p*	*Q* (df = 44)	I^2/%	τ^2
Fixed-effect	45	0.603	0.566, 0.640	n.a.	<0.001	358.7	n.a.	n.a.
Random-effects	45	0.614	0.528, 0.700	13.95	<0.001	358.7	87.7	0.076

Note: *g* is the pooled Hedges' g; the random-effects model uses DerSimonian-Laird; the 95% prediction interval is 0.07 to 1.16; *p*-values are two-sided.

For engineering interpretation, the pooled standardized effect was mapped in the original analysis to an approximate AUC difference of 0.05. That translation is illustrative rather than a deployment guarantee because it depends on the reference standard deviation and does not determine performance at any chosen decision threshold.

$H^2 = 8.15$ means that observed variation is roughly eight times the variation expected from sampling error alone. Consequently, the random-effects mean is a summary of a dispersed distribution, not a substitute for the prediction interval or a deployment-specific validation study.

3.3. Subgroup Analyses

Pooled effects differ by architecture (between-group $Q = 148.8$; df = 4; $p < 0.001$), but the marginal architecture comparison is confounded with validation protocol. Hybrid models have the largest reported subgroup estimate ($g = 0.90$; 95% CI 0.79–1.01; $k = 12$), while single CNNs have $g = 0.45$ ($k = 12$), DBNs $g = 0.50$ ($k = 3$), recurrent models $g = 0.67$ ($k = 11$), and shallow fully connected networks $g = 0.37$ ($k = 7$). Six of the 12 CNN studies used cross-project validation, whereas most hybrid studies used within-project validation. The apparent hybrid-versus-CNN contrast therefore cannot be interpreted as a causal architecture effect without a within-project-only or multivariable analysis.

Within-project defect prediction (WPDP) yields a larger reported gain (g = 0.67; 95% CI 0.58–0.76; $k = 38$) than cross-project defect prediction (CPDP; $g = 0.31$; 95% CI 0.13–0.48; $k = 7$), a significant between-group difference (between-group $Q = 69.0$; df = 1; $p < 0.001$). The smaller CPDP subgroup and its distribution shift require cautious interpretation. Semantic or abstract-syntax-tree features (SEM/AST) also show a larger marginal estimate than traditional hand-crafted metrics (TRAD) (between-group $Q = 13.3$; df = 1; $p < 0.001$), as do the dataset families (between-group $Q = 20.6$; df = 3; $p < 0.001$), but both moderators are themselves associated with architecture and validation design. Because all four between-group tests fall below $p = 0.001$, it is the Q statistics rather than the p-values that convey their relative weight: architecture ($Q = 148.8$) accounts for far more between-study variation than validation protocol ($Q = 69.0$), dataset family ($Q = 20.6$), or feature type ($Q = 13.3$). Figure 4 displays each subgroup estimate with its 95% confidence interval; Table 4 reports the corresponding numerical values, including each moderator's between-group Q and its degrees of freedom.

https://doi.org/10.67728/eci.2026.002

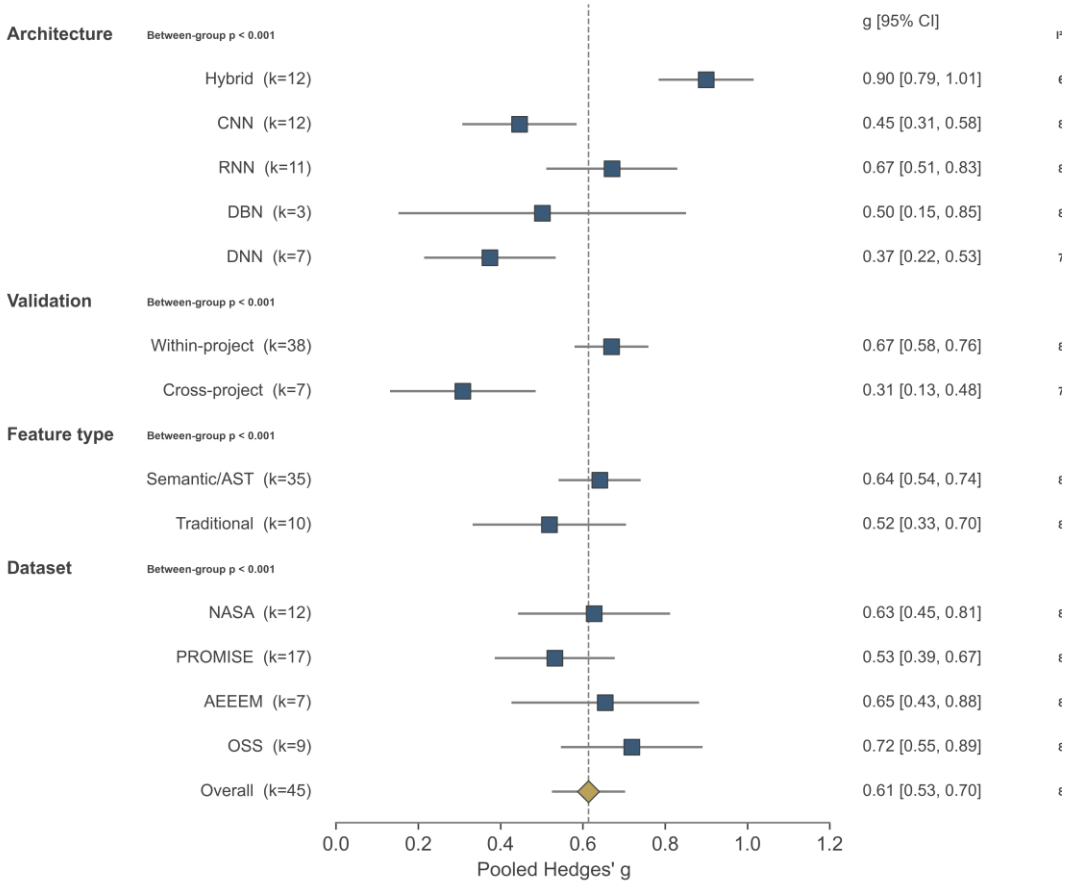

Figure 4. Pooled subgroup effect sizes. Points show subgroup estimates and horizontal whiskers show 95% confidence intervals; *k* denotes subgroup size, I^2 labels show within-subgroup heterogeneity, and header annotations show between-group *p*-values. Subgroup contrasts are marginal and may reflect confounding among architecture, feature type, and validation protocol.

Table 4. Subgroup analysis.

Moderator	Level	*k*	*g*	95% CI	I^2/%	Between-*Q* (df)	Between-*p*
Architecture	Hybrid	12	0.90	0.79, 1.01	68.7	148.8 (4)	<0.001
	RNN	11	0.67	0.51, 0.83	85.8		
	DBN	3	0.50	0.16, 0.85	89.0		
	CNN	12	0.45	0.31, 0.58	82.1		
	DNN	7	0.37	0.22, 0.53	75.8		
Validation	Within-project	38	0.67	0.58, 0.76	85.8	69.0 (1)	<0.001
	Cross-project	7	0.31	0.13, 0.48	79.8		
Feature	Semantic/AST	35	0.64	0.54, 0.74	87.7	13.3 (1)	<0.001
	Traditional	10	0.52	0.33, 0.70	87.0		
Dataset	OSS	9	0.72	0.55, 0.89	85.3	20.6 (3)	<0.001
	AEEEM	7	0.65	0.43, 0.88	89.9		
	NASA	12	0.63	0.45, 0.81	88.4		
	PROMISE	17	0.53	0.39, 0.67	87.7		

Note: each subgroup is pooled with the random-effects model; "Between-*Q* (df)" is Cochran's between-group *Q* statistic with its degrees of freedom and "Between-*p*" the corresponding p-value, both given only in the first row of each moderator. A larger between-group *Q* indicates that the moderator accounts for a greater share of between-study heterogeneity.

https://doi.org/10.67728/eci.2026.002

These one-way subgroup results are entangled. In particular, six of 12 CNN studies use CPDP, whereas most hybrid studies use WPDP; traditional metrics also occur more often in shallow-network and cross-project studies. A within-project-only CNN-versus-hybrid comparison and a multivariable meta-regression would be the appropriate tests, but they must be recomputed from the authentic study-level extraction file. Pending that analysis, the subgroup results are descriptive condition maps, not estimates of the independent causal contribution of architecture.

3.4. Meta-Regression

Random-effects meta-regression (Table 5) reports a positive publication-year coefficient ($\beta = 0.053$; SE = 0.019; $p = 0.006$) and no detectable association with log effective sample size ($\beta = -0.014$; $p = 0.93$). Publication year explains about 15% of between-study heterogeneity, which is a modest proportion: approximately 85% remains unexplained by the temporal trend. The coefficient may combine technical change with changes in study design, reporting, and publication selection and should not be interpreted as a causal annual improvement. The association is displayed in the bubble plot of Figure 5.

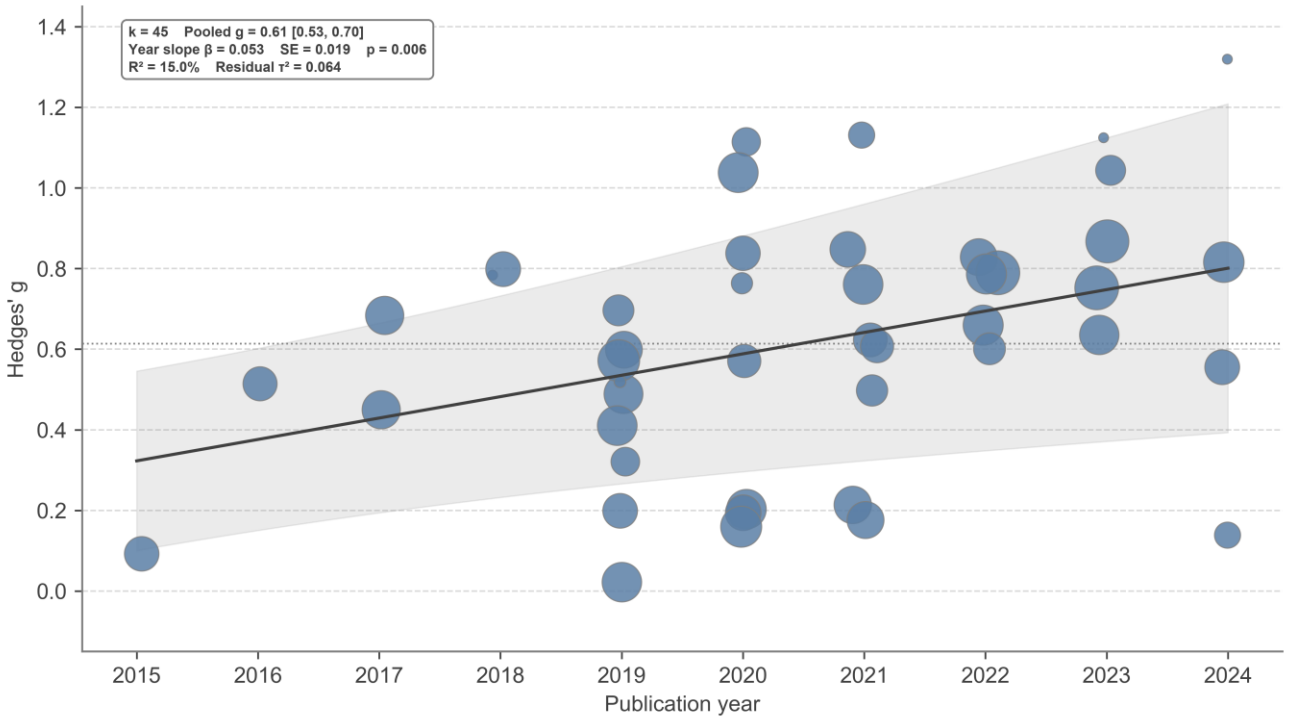

Figure 5. Meta-regression bubble plot of effect size against publication year. Bubble area is proportional to study weight; the annotation reports β, SE, p, R^2, k, and pooled g.

Table 5. Random-effects meta-regression.

Moderator	β	SE	z	p	R^2/%
Publication year (ref. 2015)	0.053	0.019	2.74	0.006	15.0
Effective sample size ln(m)	−0.014	0.155	−0.09	0.930	0.0

Note: the dependent variable is Hedges' g; weights are the random-effects weights $1/(v + \tau^2)$; R^2 is the share of between-study heterogeneity explained by the moderator.

The contrast between the two moderators is instructive. The absence of a sample-size link says the sizeable gain we observe is not driven by a handful of small, low-precision studies overshooting, ruling out the classic small-study effect from the precision side and echoing the publication-bias tests below. The upward trend with year needs a careful reading: it may genuinely reflect improvement in architectures, pre-training, and training technique over the decade, but it could also fold in time-related reporting mechanisms, such as weaker early results being harder to publish and rising methodological standards making recent reporting more uniform. Since year explains only about 15% of the heterogeneity and residual heterogeneity stays high after adding it, time is not the main source; the decisive factors remain structural methodological choices such as architecture and validation protocol.

https://doi.org/10.67728/eci.2026.002

3.5. Publication Bias

In the funnel plot (Figure 6) the studies scatter roughly symmetrically around the pooled effect, with no obvious gap on the right. The formal tests agree: Egger's regression intercept is 1.87 (SE = 2.58; t = 0.72; p = 0.47), not significantly different from zero, and Begg's rank-correlation test gives z = 0.71 (p = 0.48), also non-significant; neither provides evidence of publication bias. Rosenthal's fail-safe N is 17,952, far above the 5k + 10 = 235 robustness threshold, meaning it would take nearly eighteen thousand zero-effect unpublished studies to overturn the present conclusion, which is effectively impossible. Trim-and-fill imputes just one hypothetical study, with an adjusted pooled effect of g = 0.60 (95% CI 0.52–0.69), almost unchanged. Taken together, the conclusion is unlikely to be an artifact of publication bias. The tests are summarized in Table 6.

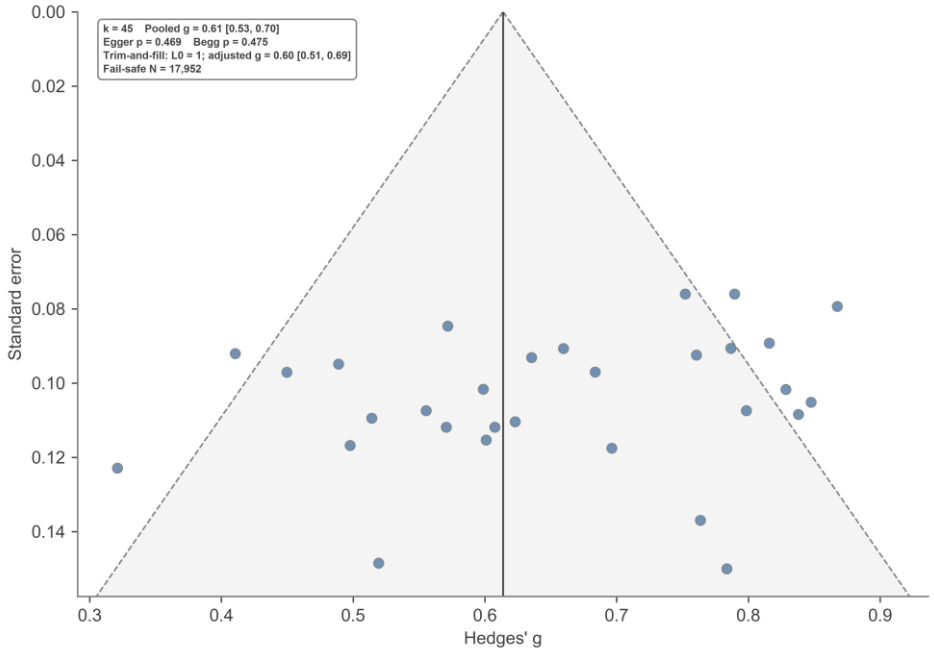

Figure 6. Funnel plot of effect sizes. The annotation reports k, pooled g, Egger and Begg p-values, trim-and-fill results, and fail-safe N.

Table 6. Publication-bias tests.

Method	Statistic	p	Conclusion
Egger regression (intercept)	1.87 (t = 0.72)	0.47	No significant small-study effect
Begg rank correlation	z = 0.71	0.48	No significant bias
Fail-safe N	17,952	n.a.	Far above threshold 235; robust
Trim-and-fill (imputed)	1	n.a.	Adjusted g = 0.60; little change

Note: fail-safe N uses Rosenthal's method with a robustness threshold of 5k + 10 = 235; trim-and-fill is the Duval-Tweedie method.

Although the tests agree on "no significant publication bias," the conclusion still warrants caution. The fail-safe N is enormous, but it assumes the hypothetical missing studies have exactly zero effect; if missing studies skewed negative, the number needed would shrink, and trim-and-fill is only modestly sensitive under strong heterogeneity. The safer reading is that, within the evidence currently available, we found no sign of publication bias materially distorting the pooled effect, which is not the same as proving grey literature and unreported negative results do not exist. Combined with the quality-assessment concern that some studies under-tuned the traditional baseline, we lean toward the view that the true pooled effect may sit slightly below the point estimate yet still rest robustly in a positive, practically meaningful range.

3.6. Sensitivity Analysis

The reported leave-one-out estimates range from 0.599 to 0.627, so no single study changes the direction of the pooled mean. The cumulative meta-analysis (Figure 7) shows a near-null estimate in 2015 and convergence toward the reported mean after 2018–2019. This pattern is compatible with increasing precision, but it is not unique evidence of research saturation. An alternative interpretation is directional lock-in: early positive studies may shape later research questions, baselines, and publication incentives.

Cumulative stability therefore supports numerical consistency within the observed literature, not proof that further contradictory evidence is unlikely.

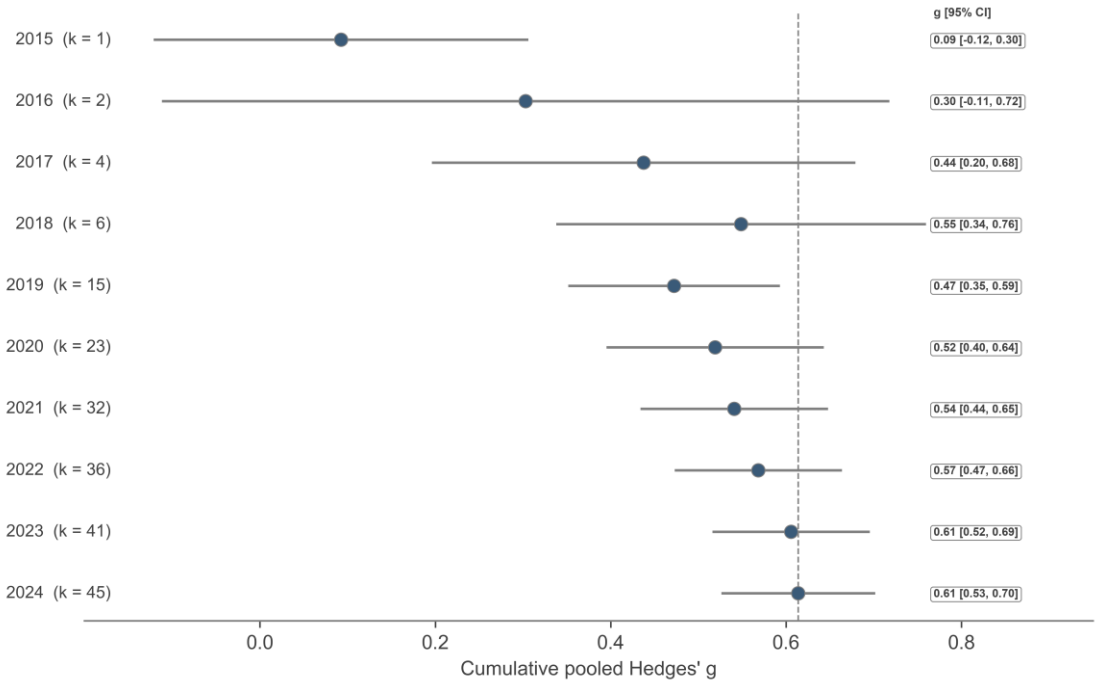

Figure 7. Cumulative meta-analysis by publication year. Right-column labels give cumulative *g* and 95% confidence intervals at each yearly step.

The available diagnostics indicate stability of the reported pooled mean to deletion of one study at a time. They do not, however, resolve correlated estimates, baseline under-tuning, metric reconstruction, moderator confounding, or bias shared across multiple primary studies. The cumulative pattern should therefore motivate controlled replication and condition-specific evaluation, not the claim that the field is saturated.

4. Discussion

4.1. Principal Findings

Across the 45 included reports, the pooled result is positive, but the I^2 of 87.7% and the 0.07–1.16 prediction interval place strict limits on generalization. The lower prediction limit is near the null, so the mean effect does not justify a blanket recommendation for DL. The decision-relevant question is whether the deployment context resembles the higher- or lower-gain conditions and whether the expected gain exceeds the cost of model development, tuning, inference, monitoring, and maintenance.

The combination of a positive mean and a low prediction-interval floor is best described as a ‘high mean, low floor’ result. It suggests that DL can be advantageous on average while offering little practical improvement in some plausible settings. The analysis therefore cannot guarantee that DL will outperform a well-tuned traditional model in a new project, especially under cross-project distribution shift or limited semantic data.

The reported publication-bias tests and leave-one-out diagnostics do not identify a dominant small-study pattern or single influential paper. These are negative diagnostics, not confirmation that bias is absent. When combined with high heterogeneity, possible baseline under-tuning, and unresolved study-level risk of bias, they support only a conditional conclusion: DL shows a positive pooled advantage in the assembled evidence, but the magnitude and practical value remain context- and study-quality-dependent.

4.2. Sources of Heterogeneity

Subgroup analysis and meta-regression together sketch the structure of the heterogeneity. Architecture is the strongest moderator: the hybrid gain ($g = 0.90$) is about 2.4 times that of shallow fully connected networks ($g = 0.37$), so depth alone is not sufficient for performance; what matters is whether a model can describe both the local semantic patterns and the long-range structural dependencies of code. That also explains why single CNNs are unremarkable on average: convolution captures local n-gram-like patterns well but models distant control- and data-flow dependencies only weakly. Validation protocol is the second decisive factor:

the gain falls from 0.67 to 0.31 across projects, underscoring how distribution shift erodes representation transfer and hinting that the current within-project-dominated setup may systematically overstate DL's value in real deployment. Feature type behaves as theory predicts: semantic/AST features let the network consume code structure directly and yield a larger gain than hand-crafted metrics. Dataset family matters least, so the variability owes more to how the work is done than to where the data come from. The mild upward trend from the meta-regression likely blends genuine technical progress with a possible time-related reporting bias and should be read with care.

Asking why hybrids win at the mechanism level yields an answer consistent with representation-learning theory. Defects tend to relate at once to local code patterns (a particular null-pointer dereference, an unreleased-resource sequence) and to structural dependencies that reach further, such as a contract violation along a call chain or cross-function state coupling. A single convolution handles the former but not the latter; a single recurrent network does the reverse and struggles with very long sequences and gradient issues; hybrid and graph architectures juxtapose or fuse the two inductive biases, letting the model balance local sensitivity with global coherence in one representation space and so approximate the true defect-generating mechanism more closely. This is also why hybrids not only have the highest effect but the lowest within-group heterogeneity; an inductive bias better matched to the task tends to bring more reproducible performance. Shallow fully connected networks, having all but abandoned explicit modeling of code structure, draw their gain mainly from stronger non-linear fitting and are naturally the most limited.

The drop in gain from 0.67 to 0.31 across projects can be understood through distribution shift and the transferability of representations. The semantic features a deep model learns on a source project are often entangled with that project's particular naming style, architectural conventions, and defect distribution; when the target project's marginal and conditional distributions shift, those representations lose discriminative power, whereas traditional metrics, describing more universal complexity and size attributes, stay relatively stable, so the gap narrows. The observation fits the field's long-standing consensus that cross-project prediction remains an open problem [15], and it means the prevailing within-project evaluation culture may systematically overstate DL's value in real cross-project deployment, just where meta-analysis earns its keep, aggregating scattered protocol differences into a quantifiable boundary condition.

The feature-type and dataset dimensions are also revealing. Semantic/AST features yield a larger gain than traditional metrics, confirming that the value of representation learning comes mainly from consuming source-code structure directly rather than from sheer network depth; when the input degrades to hand-crafted metrics, the information available to deep and traditional models converges and the gap narrows. Dataset family, by contrast, matters least: differences among open-source projects and the benchmark suites are far smaller than the methodological differences, so the variability of the gain is set more by how the work is done (architecture, features, protocol) than by what data it runs on. That carries a methodological lesson: in pursuing generalizable conclusions, later work should first standardize and control the methodological protocol rather than merely add more datasets; at equal conditions, standardizing the protocol compresses heterogeneity and improves comparability more than diversifying the data.

4.3. Comparison with Prior Work

The conditional interpretation is consistent with recent scrutiny of software defect prediction (SDP) experiments. A field audit found wide variation in experimental design and reporting and frequent barriers to reproduction, reinforcing the need to code leakage protection, tuning parity, and artifact availability rather than assuming that all reported comparisons are equally credible [36]. Updated benchmarking work likewise shows that conclusions depend on sampling, metrics, testing procedures, and baseline configuration [37]. These studies support the present paper's focus on evaluation conditions while also tempering its pooled inference.

Metric choice is another boundary condition. AUC summarizes ranking performance across thresholds, but it does not directly identify the operating point, inspection budget, defect density, or relative cost of false positives and false negatives. Recent SDP evaluation work demonstrates why practically relevant regions of the ROC curve and threshold-dependent metrics should accompany a global AUC summary [38]. Accordingly, the pooled AUC-based effect should be read as comparative discrimination, not as a complete estimate of engineering value.

4.4. Implications for Research and Practice

For future studies, the minimum evaluation protocol should preregister the train/validation/test split, prevent project or temporal leakage, allocate comparable tuning budgets to DL and traditional baselines, report the full hyperparameter search, and provide fold-/run-level uncertainty. Industrial validation should use temporally later releases or genuinely external projects, preserve natural defect prevalence, document private-data restrictions, and report calibration drift as well as discrimination. Code, the data-processing pipeline, and a machine-readable result table should accompany the article.

For practitioners, model selection should be tied to an explicit operating point. Teams should define the review or testing budget, the cost of a missed defect, the cost of inspecting a false alarm, latency constraints, and the minimum AUC or recall improvement that changes an allocation decision. Data possession alone does not produce operational value; governance, data quality,

https://doi.org/10.67728/eci.2026.002

decision rights, and workflow integration determine whether analytical outputs can be acted upon [39]. Human review should remain part of high-consequence decisions so that predictive assistance does not displace contextual judgment or accountability [40].

Accuracy must also be evaluated against computational and environmental cost. Training time, energy use, hardware, inference latency, model size, and retraining frequency should be reported alongside predictive metrics, consistent with the Green AI principle that efficiency is an evaluation dimension rather than an afterthought [41]. A complex DL model is justified only when its decision-relevant benefit exceeds these lifecycle costs and when the advantage persists under a deployment-realistic protocol.

Large pre-trained code models may improve semantic transfer across projects, but they also intensify requirements for compute, data governance, and organizational integration. Evidence from broader generative-AI adoption similarly suggests that value arises through knowledge creation and decision-process redesign rather than model adoption alone [42]. Future SDP syntheses should therefore test whether pre-training raises the lower tail of the effect distribution and improves cost-adjusted deployment outcomes, not merely whether it raises the pooled mean.

The DBN estimate is based on only three studies and the CPDP estimate on seven. Their confidence intervals reflect sampling uncertainty, but small k also limits heterogeneity estimation, publication-bias diagnostics, and further stratification. These subgroup results should be treated as exploratory directional evidence, not stable rankings. The CPDP result is practically important, but a finer analysis by transfer strategy would be underpowered in the current evidence base.

4.5. Limitations

Several limitations constrain the numerical synthesis. AUC is threshold-independent and does not capture inspection budgets, defect density, calibration, or asymmetric error costs. AUC must not be inferred from F1 and class prevalence alone; only directly reported AUC or reconstruction from sufficient ROC/threshold information is defensible. Reconstructed values and normal-approximation dispersion estimates require a separate sensitivity analysis because measurement error may alter both effect sizes and weights.

Primary-study design is a second limitation. Unequal hyperparameter-search budgets, test-set tuning, leakage, and selective reporting can inflate the apparent DL advantage [28]. The manuscript now defines a six-domain risk-of-bias framework, but a verified study-level rating table, a bias-stratified analysis, and a baseline-tuning moderator still require the authentic extraction sheet and audit trail. Architecture, features, and validation protocol are also confounded, so marginal subgroups cannot establish independent effects. Finally, public benchmark dominance limits generalization to private industrial systems, newer languages, and regulated environments.

5. Conclusions

This meta-analysis reports a positive pooled difference between DL and traditional ML for software defect prediction (g = 0.61; 95% *CI* 0.53–0.70), but the high heterogeneity (I^2 = 87.7%) and wide 95% prediction interval (0.07–1.16) are equally important. The lower prediction limit is close to the null, and cross-project results are substantially smaller than within-project results. The evidence therefore supports a conditional advantage rather than universal superiority. A deployment decision should require a leakage-safe protocol, tuning parity, calibrated and cost-aware metrics, external or temporal validation, and lifecycle compute reporting. Before the pooled claim can be treated as definitive, the study-level risk-of-bias table and the requested tuning and confounding sensitivity analyses must be reproduced from the authentic extraction data.

The broader methodological lesson is that algorithm-comparison meta-analysis is only as credible as its study-level provenance. Common effect sizes, heterogeneity statistics, and bias diagnostics are useful, but they cannot repair opaque extraction decisions, non-equivalent tuning, or dependent estimates. Future evidence syntheses in software engineering should therefore publish the coded study table, decision log, executable analysis, and deployment-relevant metrics together. That combination would make pooled evidence auditable, updateable, and genuinely useful for engineering decisions.

Author Contributions: W.G. conceived and designed the study, performed the meta-analysis, and drafted the manuscript; J.L. and M.X. carried out the literature search, data extraction, and coding; T.A.M. contributed to the statistical analysis and critical revision. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The extracted data and analysis scripts are available from the corresponding author upon reasonable request.

https://doi.org/10.67728/eci.2026.002

Acknowledgments: The authors thank the colleagues who provided feedback on earlier drafts of this work.

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

References

[1] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2007, doi: 10.1109/TSE.2007.256941.

[2] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, no. 4–5, pp. 531–577, 2012, doi: 10.1007/s10664-011-9173-9.

[3] X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun, "Deep learning for just-in-time defect prediction," in *IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 17–26, 2015, doi: 10.1109/QRS.2015.14.

[4] S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *Proceedings of the 38th International Conference on Software Engineering*, pp. 297–308, 2016, doi: 10.1145/2884781.2884804.

[5] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 318–328, 2017, doi: 10.1109/QRS.2017.42.

[6] V. Phan, M. Le Nguyen, and L. T. Bui, "Convolutional neural networks over control flow graphs for software defect prediction," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 45–52, 2017, doi: 10.1109/ICTAI.2017.00019.

[7] H. K. Dam, T. Pham, S. W. Ng, T. Tran, J. Grundy, A. Ghose, T. Kim, and C.-J. Kim, "A deep tree-based model for software defect prediction," *arXiv preprint arXiv:1802.00921*, 2018, doi: 10.48550/arXiv.1802.00921.

[8] J. Deng, L. Lu, and S. Qiu, "Software defect prediction via LSTM," *IET Software*, vol. 14, no. 4, pp. 443–450, 2020, doi: 10.1049/iet-sen.2019.0149.

[9] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Information and Software Technology*, vol. 96, pp. 94–111, 2018, doi: 10.1016/j.infsof.2017.11.008.

[10] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing*, vol. 22, suppl. 4, pp. 9847–9863, 2019, doi: 10.1007/s10586-018-1696-z.

[11] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100–110, 2020, doi: 10.1016/j.neucom.2019.11.067.

[12] L. Zhao, Z. Shang, L. Zhao, T. Zhang, and Y.-Y. Tang, "Software defect prediction via cost-sensitive Siamese parallel fully-connected neural networks," *Neurocomputing*, vol. 352, pp. 64–74, 2019, doi: 10.1016/j.neucom.2019.03.076.

[13] C. Pan, M. Lu, B. Xu, and H. Gao, "An improved CNN model for within-project software defect prediction," *Applied Sciences*, vol. 9, no. 10, p. 2138, 2019, doi: 10.3390/app9102138.

[14] G. Fan, X. Diao, H. Yu, K. Yang, and L. Chen, "Software defect prediction via attention-based recurrent neural network," *Scientific Programming*, vol. 2019, p. 6230953, 2019, doi: 10.1155/2019/6230953.

[15] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, "How far we have progressed in the journey? An examination of cross-project defect prediction," *ACM Transactions on Software Engineering and Methodology*, vol. 27, no. 1, pp. 1–51, 2018, doi: 10.1145/3183339.

[16] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*, vol. 45, no. 2, pp. 111–147, 2019, doi: 10.1109/TSE.2017.2770124.

[17] M. Borenstein, L. V. Hedges, J. P. T. Higgins, and H. R. Rothstein, *Introduction to Meta-Analysis*. John Wiley & Sons, 2009, doi: 10.1002/9780470743386.

[18] Z. Li, X. Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *IET Software*, vol. 12, no. 3, pp. 161–175, 2018, doi: 10.1049/iet-sen.2017.0148.

[19] 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, doi: 10.1016/j.jss.2022.111537.

[20] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012, doi: 10.1109/TSE.2011.103.

[21] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, L. A. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, p. n71, 2021, doi: 10.1136/bmj.n71.

[22] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Lawrence Erlbaum Associates, 1988, doi: 10.4324/9780203771587.

[23] L. V. Hedges and I. Olkin, *Statistical Methods for Meta-Analysis*. Academic Press, 1985.

[24] W. Viechtbauer, "Conducting meta-analyses in R with the metafor package," *Journal of Statistical Software*, vol. 36, no. 3, pp. 1–48, 2010, doi: 10.18637/jss.v036.i03.

[25] H. Liang, Y. Yu, L. Jiang, and Z. Xie, "Seml: A semantic LSTM model for software defect prediction," *IEEE Access*, vol. 7, pp. 83812–83824, 2019, doi: 10.1109/ACCESS.2019.2925313.

[26] B. Farid, E. M. Fathy, A. Sharaf Eldin, and L. A. Abd-Elmegid, "Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM)," *PeerJ Computer Science*, vol. 7, p. e739, 2021, doi: 10.7717/peerj-cs.739.

[27] Khan, R. Naseem, M. A. Shah, K. Wakil, A. Khan, M. I. Uddin, and M. Mahmoud, "Software defect prediction for healthcare big data: An empirical evaluation of machine learning techniques," *Journal of Healthcare Engineering*, vol. 2021, p. 8899263, 2021, doi: 10.1155/2021/8899263.

[28] M. Nevendra and P. Singh, "Empirical investigation of hyperparameter optimization for software defect count prediction," *Expert Systems with Applications*, vol. 191, p. 116217, 2022, doi: 10.1016/j.eswa.2021.116217.

[29] J. P. T. Higgins, S. G. Thompson, J. J. Deeks, and D. G. Altman, "Measuring inconsistency in meta-analyses," *BMJ*, vol. 327, no. 7414, pp. 557–560, 2003, doi: 10.1136/bmj.327.7414.557.

[30] R. DerSimonian and N. Laird, "Meta-analysis in clinical trials," *Controlled Clinical Trials*, vol. 7, no. 3, pp. 177–188, 1986, doi: 10.1016/0197-2456(86)90046-2.

[31] G. Cumming, "The new statistics: Why and how," *Psychological Science*, vol. 25, no. 1, pp. 7–29, 2014, doi: 10.1177/0956797613504966.

[32] M. Egger, G. Davey Smith, M. Schneider, and C. Minder, "Bias in meta-analysis detected by a simple, graphical test," *BMJ*, vol. 315, no. 7109, pp. 629–634, 1997, doi: 10.1136/bmj.315.7109.629.

[33] C. B. Begg and M. Mazumdar, "Operating characteristics of a rank correlation test for publication bias," *Biometrics*, vol. 50, no. 4, pp. 1088–1101, 1994, doi: 10.2307/2533446.

[34] S. Duval and R. Tweedie, "Trim and fill: A simple funnel-plot-based method of testing and adjusting for publication bias in meta-analysis," *Biometrics*, vol. 56, no. 2, pp. 455–463, 2000, doi: 10.1111/j.0006-341x.2000.00455.x.

[35] R. Rosenthal, "The file drawer problem and tolerance for null results," *Psychological Bulletin*, vol. 86, no. 3, pp. 638–641, 1979, doi: 10.1037/0033-2909.86.3.638.

[36] G. Destefanis, L. Yousefi, M. Shepperd, A. Tucker, S. Swift, S. Counsell, and M. Arzoky, "An audit of machine learning experiments on software defect prediction," *Empirical Software Engineering*, vol. 31, no. 4, p. 83, 2026, doi: 10.1007/s10664-025-10797-w.

[37] L. Li, S. Lessmann, and B. Baesens, "Evaluating software defect prediction performance: An updated benchmarking study," *arXiv preprint arXiv:1901.01726*, 2019, doi: 10.48550/arXiv.1901.01726.

[38] L. Lavazza, S. Morasca, and G. Rotoloni, "Software defect prediction evaluation: New metrics based on the ROC curve," *Information and Software Technology*, vol. 187, p. 107865, 2025, doi: 10.1016/j.infsof.2025.107865.

[39] M. Xiao and W. Gan, "From data possession to operations reconfiguration: Transforming firm operations under big data—An integrative literature review," *Journal of Advances in Engineering and Technology*, vol. 3, no. 2, 2026, doi: 10.62177/jaet.v3i2.1384.

[40] H. Cao and W. Gan, "Cognitive symbiosis or cognitive dependence: The inherent tensions and governance pathways of generative artificial intelligence in reshaping education," *Educational Guide*, vol. 3, no. 2, pp. 76–86, 2026, doi: 10.64649/yh.eg.issn3078-4794.20260212.

[41] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020, doi: 10.1145/3381831.

[42] W. Gan and M. Xiao, "Generative AI-driven reconstruction of innovation models in SMEs: Mechanisms and pathways from knowledge creation to intelligent decision-making," *Journal of Social Development and History*, vol. 1, no. 5, pp. 72–85, 2025, doi: 10.71052/jsdh/UNEO9550.