Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Cross-Attention or Concatenation? A Multi-Seed Empirical Study of Code-Comment Fusion for Pull Request Outcome Prediction

Ahmed Hussen

ahmbim25@gmail.com

Addis Ababa Science and Technology University

Research Article

Keywords: Pull request outcome prediction; Code review; CodeBERT; Cross-modal attention; Multi-seed evaluation; Empirical study

Posted Date: November 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11767396/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Cross-Attention or Concatenation? A Multi-Seed Empirical Study of Code–Comment Fusion for Pull Request Outcome Prediction

1 Introduction

Code review is one of the principal quality-assurance mechanisms in modern software engineering. Platforms such as GitHub, Gerrit, and GitLab have accumulated extensive records of pull requests, code changes, reviewer comments, and outcomes (**gousios2014exploratory?**; **bacchelli2013expectations?**; **mcintosh2014impact?**). These records make it possible to study empirically which factors accompany the acceptance or rejection of a contribution (**tsay2014influence?**; **gousios2014exploratory?**) and to build models that predict the outcome of a PR from its content.

Transformer-based encoders have made it straightforward to represent both modalities that matter in a review: source code, with models such as CodeBERT (**feng2020codebert?**) and GraphCodeBERT (**guo2021graphcodebert?**), and natural-language reviewer comments, with models such as RoBERTa (**liu2019roberta?**). Prior work on code–comment fusion for review tasks, such as DeepReview (**li2019deepreview?**) and CLMN (**wu2022contrastive?**), shows that comments carry useful signal, but it typically collapses each modality to a single vector before combining them, and it rarely compares fusion strategies under identical conditions. A natural alternative is to let code tokens and comment tokens attend to each other directly (cross-modal attention). Whether this extra machinery pays off over plain concatenation is an empirical question, and it is easy to answer wrongly: fine-tuned transformers can vary by several tenths of a percentage point from seed to seed, which is comparable to the differences between competing designs.

This paper therefore treats fusion for PR outcome prediction as a controlled empirical comparison. We train every model with five random seeds on the same split, report means and standard deviations, and use paired tests where the data allow. Our study addresses four research questions:

- **RQ1:** How do fusion strategies, code encoders, and unimodal baselines compare in test accuracy across seeds?

- **RQ2:** Does combining code and comments significantly outperform unimodal baselines?
- **RQ3:** Does bidirectional cross-modal attention outperform simpler fusion (concatenation, late fusion) under identical encoders and settings?
- **RQ4:** What do uncertainty estimates and interpretability analyses reveal, and what does the dataset's composition imply for the validity of the evaluation?

Contributions.

1. A multi-seed benchmark of 22 configurations for three-class PR outcome prediction on 33,955 PRs, with accuracy, macro-F1, MCC, and macro-AUC for every configuration.
2. Evidence that adding comments to code yields a consistent gain of about 1.6–1.9 pp, whereas cross-attention adds only about 0.34 pp over concatenation on average, with an encoder-dependent sign and no significant per-encoder difference. A single-seed comparison reversed the ordering for one configuration.
3. A dataset-composition analysis showing that project, author, and reviewer identity are strongly associated with the outcome, which signals a shortcut risk under random splitting and motivates project-held-out evaluation.
4. A fully specified protocol (data flow, label construction, hyperparameters, seeds) so that results can be reproduced and extended.

Scope. The study is retrospective: all inputs, including review duration, are available in the historical record, and we do not claim that the models can predict outcomes before a review concludes. We evaluate encoder-based models only and do not compare against LLM- or agent-based approaches.

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the data, models, and evaluation protocol. Section 4 reports results for each research question. Section 6 discusses threats to validity, and Section 7 concludes.

2 Related Work

2.0.0.1Pull request acceptance and review outcomes.

Empirical studies of the pull-based development model have examined which technical and social factors accompany PR acceptance (**gousios2014exploratory?**; **tsay2014influence?**). Process-level studies relate review coverage and participation to software quality (**mcintosh2014impact?**), and Bacchelli and Bird (**bacchelli2013expectations?**) show that review serves purposes beyond defect finding. Murphy-Hill et al. (**murphyhill2022pushback?**) report that review outcomes and

friction differ across demographic groups, which motivates caution when models are trained on historical outcomes.

2.0.0.2Neural representations of code and review text.

The naturalness hypothesis (**allamanis2018survey?**) underlies the use of statistical language models for code. CodeBERT (**feng2020codebert?**) introduced bimodal pre-training, GraphCodeBERT (**guo2021graphcodebert?**) added data-flow information, UniXcoder (**guo2022unixcoder?**) unified several pre-training objectives, and CodeT5 (**wang2021codet5?**) is an identifier-aware encoder–decoder. CodeReviewer (**li2022automating?**) is pre-trained specifically on code-change and review data. RoBERTa (**liu2019roberta?**) provides strong contextual representations for the natural-language side.

2.0.0.3Automated code review and fusion of code and comments.

Surveys and empirical studies of review automation document both its promise and its weaknesses (**tufano2024strengths?**), and recent work examines how developers interact with deep-learning-based review assistants (**tufano2024deeplearning?**). For PR or review outcomes specifically, DeepReview (**li2019deepreview?**) formulates the task as multiple-instance learning over code hunks and comments, and CLMN (**wu2022contrastive?**) applies contrastive learning to code–comment pairs. These systems report results from single configurations, and controlled comparisons between fusion mechanisms across seeds and encoders are rare. Our contribution is such a comparison.

3 Study Design

3.1 Dataset

3.1.0.1Collection and filtering.

PRs were mined from GitHub (via PyGithub), Gerrit (REST API), and GitLab (REST API) for the period January 2020 to March 2025. Starting from 402,661 raw PR records, we removed bot-created items, PRs with empty reviews, inactive projects, and duplicate PR records, and aggregated review activity to the PR level. Table [tab:dataflow] reports the sequential filtering flow. The final dataset has 33,955 PRs from 155 projects, with 231,880 review comments in total (mean 6.83 per PR, median 6). Intermediate per-filter counts are in the released construction scripts.

@rLrrr@ **#** & **Step** & **Removed** & **Remaining** & **Retained**
& Raw PRs scraped (GitHub, Gerrit, GitLab) & — & 402,661 & 100.0%
2 & Quality filters (bots, empty reviews, inactive projects) & 337,914 & 64,747 & 16.1%
3 & Duplicate removal (same PR scraped more than once) & 20,013 & 44,734 & 11.1%
4 & Platform merge and final label filter & 10,779 & 33,955 & 8.4%

Final platform composition: GitHub 32,191 (94.8%), GitLab 1,380 (4.1%), Gerrit 384 (1.1%).

3.1.0.2Languages.

Table 1 gives the language distribution. Python and Java account for 80.4% of PRs. Minority languages contribute at most 300 PRs each (a per-language cap in the mining scripts). The `language` field is the repository's primary language as reported by the platform; two values (LLVM and Jupyter Notebook) are repository-level tags rather than programming languages and are kept as distinct categories because the models use the field as provided.

Language distribution of the 33,955 PRs.

Language	**PRs**	**Share**
Python	15,633	46.0%
Java	11,691	34.4%
C++	4,523	13.3%
Scala, JavaScript, LLVM	300 each	0.9% each
Groovy	295	0.9%
C	289	0.9%
Jupyter Notebook	261	0.8%
Kotlin	199	0.6%
HTML, Rust (combined)	164	0.5%
Total (12 languages)	**33,955**	**100%**

3.1.0.3Labels.

Each PR receives exactly one label from its final recorded state, harmonised across platforms in the released scripts. *Merge* ($n = 13{,}984$, 41.2%) denotes a PR merged into the target branch. *Reject* ($n = 11{,}174$, 32.9%) denotes a PR explicitly closed or abandoned without merging. *Revise* ($n = 8{,}797$, 25.9%) denotes a PR that was not merged and did not receive an explicit rejection: about 68% of these were closed after a period of inactivity and about 32% were still open when the data were collected. Revise is therefore best read as "outcome unresolved" rather than as an explicit request for changes, and the open PRs are right-censored (they might have been merged later). We keep the label name *revise* for consistency with the released data.

3.1.0.4Model inputs.

Each instance has (i) the code diff, (ii) the concatenated review comments, (iii) the repository language, and (iv) four numerical metadata features: lines added, lines deleted, number of files changed, and review duration (the time from creation to merge or close, in seconds). No other features, such as reviewer experience or comment sentiment, are used. Review duration is undefined for PRs without a close time; all of these ($\approx 8.2\%$ of PRs) belong to the revise class and were imputed with the training-set median. Metadata were standardised with training-set statistics only. Median review duration differs sharply by

class (merge 98,192 s, about 27 h; reject 78 s; revise 53 s), so it is a strong correlate of the label; we return to this in Section 6.

3.1.0.5Split.

The dataset was split 70/15/15 with a two-stage stratified split on the class label only (`random_state` = 42), giving 23,768 training, 5,093 validation, and 5,094 test instances. The same split is used for every model. The split is random: it does not group by project, author, reviewer, or time.

3.2 Models

3.2.0.1Proposed hybrid.

Figure 1 shows the architecture. CodeBERT encodes the diff (at most 512 tokens) and RoBERTa encodes the comments (at most 128 tokens), each with its own tokeniser, producing token-level states $\boldsymbol{H}_c \in \mathbb{R}^{L_c \times 768}$ and $\boldsymbol{H}_r \in \mathbb{R}^{L_r \times 768}$. Bidirectional cross-modal attention lets code tokens query the comment sequence and vice versa, using 8-head multi-head attention (**vaswani2017attention?**) with padding masks:

$$\widetilde{\boldsymbol{H}}_c = \mathrm{LayerNorm}\big(\boldsymbol{H}_c + \mathrm{Dropout}(\mathrm{MHA}(\boldsymbol{H}_c, \boldsymbol{H}_r, \boldsymbol{H}_r))\big),$$
$$\widetilde{\boldsymbol{H}}_r = \mathrm{LayerNorm}\big(\boldsymbol{H}_r + \mathrm{Dropout}(\mathrm{MHA}(\boldsymbol{H}_r, \boldsymbol{H}_c, \boldsymbol{H}_c))\big).$$

The attended sequences are reduced by masked mean pooling to $\boldsymbol{e}_c, \boldsymbol{e}_r \in \mathbb{R}^{768}$. The language is mapped to a 16-dimensional embedding $\boldsymbol{e}_l$ and the metadata to a 32-dimensional projection $\boldsymbol{e}_m$ (linear layer with ReLU). The concatenation $\boldsymbol{h} = [\boldsymbol{e}_c; \boldsymbol{e}_r; \boldsymbol{e}_l; \boldsymbol{e}_m] \in \mathbb{R}^{1584}$ is classified by an MLP ($1584 \to 256 \to 128 \to 3$, ReLU, dropout 0.2 and 0.1). The model has about 254 million parameters.

Proposed hybrid model. CodeBERT and RoBERTa encode the code diff and the review comment into token-level states. Bidirectional cross-modal attention (solid arrows: queries; dashed arrows: keys and values) is followed by residual connections and layer normalisation. The attended sequences are mean-pooled and concatenated with a language embedding and a projected metadata vector into a 1,584-dimensional representation that an MLP maps to the three outcome classes.

3.2.0.2Fusion variants (same encoders).

Concatenation (early fusion) pools $\boldsymbol{H}_c$ and $\boldsymbol{H}_r$ independently without cross-attention and concatenates them with $\boldsymbol{e}_l$ and $\boldsymbol{e}_m$. *Late fusion* classifies each modality separately and combines the per-modality predictions with learnable weights.

3.2.0.3Alternative code encoders.

To test whether conclusions depend on the code encoder, we replace CodeBERT with GraphCodeBERT, UniXcoder, CodeT5 (encoder), or CodeReviewer (encoder), each with both cross-attention and concatenation fusion, keeping RoBERTa for comments. Code-only variants of each encoder are also trained.

3.2.0.4Unimodal baselines.

CodeBERT-only (code diff only), plain RoBERTa (comments only), and a BiLSTM with additive attention over comments (text only; 300-dimensional embeddings, two bidirectional layers with 256 hidden units per direction, dropout 0.2, RoBERTa tokeniser).

3.2.0.5Prior systems (re-implemented).

A CLMN-style model (**wu2022contrastive?**) that trains a CodeBERT-based encoder with a contrastive loss on code–comment pairs (temperature 0.07) and uses a three-way instead of a binary classification head, and a DeepReview-style model (**li2019deepreview?**) that keeps the multiple-instance attention pooling over comments but uses a RoBERTa encoder in place of the original recurrent encoder, so that the comparison isolates the pooling mechanism. Both are adaptations to our three-class task and split, not reproductions of the original evaluations.

3.3 Training Protocol

All models minimise cross-entropy with inverse-frequency class weights, using AdamW (learning rate 2×10^{-5}, weight decay 0.01), gradient clipping (norm 1.0), at most 10 epochs, and early stopping with patience 3 on validation accuracy; the checkpoint with the best validation accuracy is evaluated on the test set. Encoder parameters are fine-tuned jointly with the fusion and classification layers. The batch size is 16, with two exceptions: CodeT5 with cross-attention uses 8 (GPU memory), and the BiLSTM uses 32. Because the BiLSTM is trained from scratch, it uses a learning rate of 10^{-3}, the conventional range for recurrent models, while the pre-trained transformers use 2×10^{-5}. Every model is trained with five seeds $\{42,123,456,789,101112\}$ on the same split. For the proposed model, one epoch takes about 58 minutes (1,486 optimisation steps).

3.4 Evaluation Protocol

We report accuracy (primary), macro-F1, the multiclass Matthews correlation coefficient (MCC) (**chicco2020advantages?**; **gorodkin2004comparing?**), and macro one-vs-rest AUC, as mean $\pm$ sample standard deviation ($n - 1$) over seeds. A single accuracy estimate on $n = 5{,}094$ test instances has a 95% confidence interval of about ± 0.9 pp; this describes one measurement and is not a criterion for distinguishing two models. We therefore compare models with (i) paired t-tests over seeds on per-seed accuracy, with Holm correction over the family of comparisons against the proposed model, and (ii) McNemar's test (**dietterich1998approximate?**) on paired per-instance predictions at a single checkpoint (seed 42). Dietterich documents that McNemar's test can be anti-conservative when classifiers are trained on the same data, and a paired t-test over five seeds has low power; we interpret both with caution. The seeds vary initialisation and data order but not the train/test split.

3.5 Environment

The proposed hybrid, the fusion ablations on CodeBERT+RoBERTa, and the CodeBERT-only baseline were trained with an earlier notebook environment (Python 3.9, PyTorch 2.0, Transformers 4.30) on a single NVIDIA T4 GPU in Google Colab; all other models were trained with Python 3.13, PyTorch 2.11, and Transformers 5.x. This difference is a confound that we discuss in Section 6.

4 Results

4.1 RQ1: Overall Performance Across Seeds

Table [tab:overall] reports mean and sample SD over seeds for every configuration, grouped by input type and sorted by accuracy within each group.

@Lcccccc@ **Model** & **Seeds** & **Accuracy** & **Macro-F1** & **MCC** & **AUC** &

CodeBERT+RoBERTa, cross-attention (proposed) & 5 & 0.8811 ± 0.0039 & 0.8789 & 0.8195 & 0.9680 & $+1.85$
GraphCodeBERT, cross-attention & 5 & 0.8845 ± 0.0036 & 0.8822 & 0.8242 & 0.9683 & $+2.19$
UniXcoder, cross-attention & 5 & 0.8830 ± 0.0046 & 0.8800 & 0.8228 & 0.9668 & $+2.04$
GraphCodeBERT, concatenation & 5 & 0.8808 ± 0.0038 & 0.8788 & 0.8181 & 0.9678 & $+1.82$
CodeReviewer, cross-attention (Run 2) & 5 & 0.8805 ± 0.0045 & 0.8781 & 0.8176 & 0.9660 & $+1.79$
UniXcoder, concatenation & 5 & 0.8803 ± 0.0036 & 0.8778 & 0.8178 & 0.9671 & $+1.77$
CodeReviewer, concatenation (Run 2) & 5 & 0.8768 ± 0.0065 & 0.8747 & 0.8124 & 0.9642 & $+1.42$
CodeBERT+RoBERTa, concatenation & 5 & 0.8722 ± 0.0076 & 0.8723 & 0.8055 & 0.9631 & $+0.96$
CodeReviewer, cross-attention (Run 1) & 5 & 0.8693 ± 0.0071 & 0.8671 & 0.8015 & 0.9591 & $+0.67$
CodeT5, concatenation & 5 & 0.8679 ± 0.0049 & 0.8660 & 0.7986 & 0.9586 & $+0.53$
CodeT5, cross-attention & 5 & 0.8660 ± 0.0054 & 0.8640 & 0.7962 & 0.9584 & $+0.34$
CodeReviewer, concatenation (Run 1) & 5 & 0.8640 ± 0.0078 & 0.8624 & 0.7930 & 0.9573 & $+0.14$
CodeBERT+RoBERTa, late fusion & 5 & 0.8608 ± 0.0088 & 0.8596 & 0.7905 & 0.9270 & -0.18

UniXcoder & 5 & 0.8629 ± 0.0041 & 0.8600 & 0.7938 & 0.9509 & $+0.03$
CodeBERT & 5 & 0.8626 ± 0.0034 & 0.8607 & 0.7950 & 0.9500 & $+0.00$
CodeT5 & 5 & 0.8614 ± 0.0016 & 0.8593 & 0.7920 & 0.9479 & -0.12
GraphCodeBERT & 5 & 0.8613 ± 0.0016 & 0.8588 & 0.7925 & 0.9492 & -0.13

CodeReviewer & 5 & 0.8382 ± 0.0022 & 0.8359 & 0.7557 & 0.9363 & -2.44

Plain RoBERTa & 5 & 0.8655 ± 0.0048 & 0.864 & 0.795 & 0.958 & $+0.29$
BiLSTM + attention & 5 & 0.8488 ± 0.0060 & 0.848 & 0.770 & 0.954 & -1.38

CLMN-style (CodeBERT) & 5 & 0.8838 ± 0.0015 & 0.8817 & 0.8232 & 0.9680 & $+2.12$
DeepReview-style & 5 & 0.8233 ± 0.0014 & 0.8235 & 0.7470 & 0.9183 & -3.93

The macro-F1, MCC, and AUC values for Plain RoBERTa and the BiLSTM are computed per seed from the released console logs (macro-F1 taken from each run's `macro avg` row; MCC and AUC as printed by the evaluation script) and averaged over the five seeds. Values for the CodeBERT+RoBERTa concatenation and late-fusion rows are computed from the released per-seed predictions. All values are released with the paper's artifacts and may differ from single-run console output by at most rounding.

Three observations follow. First, the seven best models (0.8803–0.8845) lie within a 0.42 pp band, comparable to the seed-to-seed SD of the individual models (0.15–0.46 pp; about 0.4 pp for most). The proposed hybrid ranks fourth (0.8811); the best entry is GraphCodeBERT with cross-attention (0.8845), followed by the CLMN-style model (0.8838). The study therefore supports the usefulness of combining code and comments, not the superiority of a particular encoder pairing. Second, adding a second modality helps: CodeBERT-only (0.8626) to the hybrid (0.8811) is $+1.85$ pp, and comment-only RoBERTa (0.8655) to the hybrid is $+1.56$ pp. Comment-only and code-only models are within 0.3 pp of each other, so the two modalities carry comparable single-modality signal on this task. Third, differences among the code encoders are small when comments are included (0.40 pp across GraphCodeBERT, UniXcoder, CodeBERT, and CodeReviewer Run 2) but larger for other configurations: CodeReviewer as a code-only encoder is 2.4 pp below CodeBERT-only, and CodeT5 with fusion is 1.7–1.9 pp below the best model. The CodeReviewer runs differ in input formatting (29.4% of diffs are truncated at 512 tokens in Run 1 versus 40.2% in Run 2, the same truncation rate as the CodeReviewer code-only model), which accompanies a 1.1–1.3 pp accuracy difference between the runs and illustrates how sensitive these comparisons are to preprocessing.

4.2 RQ2: Combined Modalities Versus Unimodal Baselines

@Lcccc c@ **Baseline** & **Seeds** & **Mean diff.** & **Range** & & **Holm** p
CodeBERT-only & 5 & $+1.85$ & $+1.22$ to $+2.27$ & < 0.001 & 0.008
CLMN-style & 5 & -0.28 & -0.79 to $+0.27$ & 0.193 & 1.000
GraphCodeBERT, cross-attention & 5 & -0.34 & -1.00 to $+0.39$ & 0.207 & 1.000
UniXcoder, cross-attention & 5 & -0.19 & -0.88 to $+1.08$ & 0.615 & 1.000
GraphCodeBERT, concatenation & 5 & $+0.02$ & -0.76 to $+0.94$ & 0.943 & 1.000
UniXcoder, concatenation & 5 & $+0.08$ & -0.77 to $+0.49$ & 0.751 & 1.000
CodeReviewer, cross-attention (Run 2) & 5 & $+0.05$ & -0.21 to $+0.37$ & 0.629 & 1.000
CodeBERT+RoBERTa, concatenation & 5 & $+0.88$ & -0.06 to $+1.83$ & 0.057 & 0.400

CodeBERT+RoBERTa, late fusion & 5 & $+2.03$ & $+0.98$ to $+4.04$ & 0.018 & 0.140
CodeT5, cross-attention & 5 & $+1.50$ & $+0.73$ to $+2.18$ & 0.007 & 0.063
DeepReview-style & 5 & $+5.78$ & $+5.32$ to $+6.32$ & < 0.001 & < 0.001

4.2.0.1Paired comparisons over seeds.

Table [tab:paired] compares the proposed model with every baseline for which five-seed logs are available. The proposed model beats CodeBERT-only in all five seeds (mean $+1.85$ pp, range $+1.22$ to $+2.27$ pp; Holm-adjusted $p = 0.008$) and DeepReview in all five seeds ($+5.78$ pp, adjusted $p < 0.001$). Its advantage over CodeT5 with cross-attention ($+1.50$ pp) is not significant after Holm correction ($p = 0.063$). The cross-attention-versus-concatenation comparison on CodeBERT+RoBERTa gives a mean gap of $+0.88$ pp (unadjusted $p = 0.057$, adjusted $p = 0.40$), and the late-fusion comparison gives $+2.03$ pp (unadjusted $p = 0.018$, adjusted $p = 0.14$). Against the six other models of the top-seven band, differences are within ± 0.34 pp in either direction and far from significant (all adjusted $p = 1.0$).

4.2.0.2McNemar's test at seed 42.

Table 2 reports McNemar's test on per-instance predictions of the seed-42 checkpoints. The proposed model (0.8787) significantly outperforms CodeBERT-only (0.8571), plain RoBERTa (0.8640), and the BiLSTM (0.8408); all three remain significant after Holm correction. This single-seed test addresses test-sampling noise for one trained model. The claim that the advantage is stable across training randomness rests on the seed-level analysis above, which is available for CodeBERT-only (all five seeds favour the hybrid) but not for plain RoBERTa and the BiLSTM, for which we report seed 42 only in this test.

McNemar's test at the seed-42 checkpoint (continuity-corrected). b: proposed model correct and baseline wrong; c: proposed wrong and baseline correct. The proposed model's accuracy is 0.8787.

Baseline	**Acc.**	b	c		
CodeBERT-only	0.8571	280	170	26.40	2.77×10^{-7}
Plain RoBERTa	0.8640	243	168	13.32	2.62×10^{-4}
BiLSTM + attention	0.8408	335	142	77.28	1.48×10^{-18}

4.3 RQ3: Cross-Attention Versus Simpler Fusion

Per-seed test accuracy of the three fusion strategies on CodeBERT+RoBERTa. The last column gives mean $\pm$ sample SD over five seeds.

Fusion	**42**	**123**	**456**	**789**	**101112**	**Mean $\pm$ SD**
Cross-attention (proposed)	0.8787	0.8787	0.8865	0.8838	0.8777	0.8811 ± 0.0039

Fusion	42	123	456	789	101112	Mean $\pm$ SD
Concatenation	0.8793	0.8740	0.8726	0.8759	0.8594	0.8722 ± 0.0076
Late fusion	0.8604	0.8626	0.8461	0.8671	0.8679	0.8608 ± 0.0088

Table 3 lists per-seed accuracy for the three fusion strategies on CodeBERT+RoBERTa. Across five seeds, cross-attention leads concatenation by $+0.88$ pp on average (paired $p = 0.057$), with per-seed gaps ranging from -0.06 pp at seed 42 to $+1.83$ pp at seed 101112. A seed-42-only comparison would have placed concatenation marginally ahead (0.8793 vs. 0.8787, a difference of three test instances net), whereas a seed-456-only comparison would have given a 1.39 pp cross-attention advantage. A single-seed comparison would therefore have reported anywhere between a small concat win and a large cross-attention win, neither of which reflects the pooled estimate; this spread is a practical caution against single-seed fusion comparisons.

Late fusion is lower than cross-attention on every seed (mean gap $+2.03$ pp; unadjusted paired $p = 0.018$, Holm-adjusted $p = 0.14$ over the eleven comparisons in Table [tab:paired]). Late fusion at seed 456 is an outlier: its test accuracy (0.8461) is 2.8 pp below its validation accuracy (0.8744), which we could not explain; excluding that seed, the mean gap to cross-attention is $+1.52$ pp.

@LccXc@ **Encoder** & **Seeds** & **Mean** & **Per-seed differences** &
CodeBERT+RoBERTa & 5 & $+0.88$ & $-0.06, +0.47, +1.39, +0.79, +1.83$ & 0.057
GraphCodeBERT & 5 & $+0.37$ & $-0.31, +0.10, +0.55, +1.02, +0.47$ & 0.18
UniXcoder & 5 & $+0.26$ & $+1.18, +0.31, -0.59, +0.46, -0.04$ & 0.42
CodeReviewer (Run 2) & 5 & $+0.37$ & $+1.06, +0.50, +0.36, +0.06, -0.12$ & 0.14
CodeT5 & 5 & -0.18 & $-0.19, +0.39, -0.27, -0.96, +0.11$ & 0.46

Table [tab:enc] repeats the comparison for each code encoder. Cross-attention is ahead of concatenation for four of the five encoders (CodeBERT, GraphCodeBERT, UniXcoder, CodeReviewer Run 2) and behind for CodeT5. The mean gap is $+0.34$ pp over encoders (one-sample t-test over the five encoder means, $p = 0.11$; sign test $p = 0.38$) and $+0.34$ pp pooled over the 25 seed pairs, which are not independent. No single encoder shows a significant difference at $\alpha = 0.05$; the closest is CodeBERT+RoBERTa at unadjusted $p = 0.057$. In short, cross-attention provides at most a small advantage over concatenation on this task, and our data cannot exclude zero.

4.4 Per-Class Behaviour

Table 4 gives the confusion matrix and per-class metrics of the proposed model at seed 456 (accuracy 0.8865).

Seed-456 confusion matrix (rows: true; columns: predicted) and per-class metrics of the proposed model. The grand total is 5,094 test instances. Macro averages: precision 0.8977, recall 0.8761, F1 0.8843.

	Predicted						
2-4 **True**	Reject	Merge	Revise	**Total**	**Prec.**	**Rec.**	**F1**
Reject	1,463	172	41	1,676	0.9360	0.8729	0.9034
Merge	61	1,984	53	2,098	0.8378	0.9457	0.8885
Revise	39	212	1,069	1,320	0.9192	0.8098	0.8611

Recall is highest for the majority class (merge, 0.946) and lowest for revise (0.810): 16.1% of true revise PRs are predicted as merge. This is consistent with the label definition, since revise PRs did not reach a decision and borderline cases fall near the merge boundary.

4.5 RQ4: Interpretability, Uncertainty, and Dataset Composition

4.5.0.1Uncertainty.

Monte Carlo Dropout (**gal2016dropout?**) with $T = 30$ stochastic forward passes over the 5,094 test instances gives a mean predictive standard deviation of 0.0192, which is 0.0162 for correctly classified and 0.0408 for misclassified instances. Used as an error detector, this uncertainty has an AUROC of 0.857, which is moderate discrimination. We report no calibration metrics (for example, expected calibration error).

4.5.0.2Interpretability.

We trained an XGBoost surrogate (**chen2016xgboost?**) to reproduce the neural model's predictions from its 1,584-dimensional fused embeddings (fidelity 99.80% on test-set embeddings) and applied SHAP (**lundberg2017unified?**). Globally, code and comment dimensions carry the weight, and language and metadata contribute negligibly. Because cross-attention mixes information between the two pooled vectors, the "code" and "comment" blocks are partially confounded. A local explanation for one merge-predicted PR (Figure 2) shows the comment block dominating (SHAP magnitude 4.74 versus 1.28 for code; language 0.00; metadata 0.001). This is a single illustration, not evidence for a general trend, and the surrogate's fidelity was not tested on embeddings from other splits or seeds.

Local SHAP modality contributions for one merge-predicted PR (single-instance illustration).

4.5.0.3Dataset composition.

We tested the association between the outcome label and four grouping variables on the full dataset (Table 5). This measures dataset-level association, not model-induced bias, and it cannot detect the demographic effects documented by Murphy-Hill et al. (**murphyhill2022pushback?**), because usernames are not protected attributes. All

associations are significant; project, author, and reviewer identity show large effects by Cohen's thresholds for Cramér's V at $df = 2$ (small ≈ 0.07, medium ≈ 0.21, large ≈ 0.35). Cramér's V is biased upward for sparse tables, so the author and reviewer values likely overstate the true effect (**bergsma2013bias?**).

Association between outcome and grouping variables on the full dataset (categories with very few PRs are merged into "other"; thresholds are in the released code).

Variable		**df**	**Cramér's V**	**Effect**
Project	24,686.83	286	0.603	Large
Author	22,846.39	1,756	0.580	Large
Reviewer	13,863.32	1,374	0.452	Large
Language	2,587.14	22	0.195	Small–medium

The strong project effect ($V = 0.60$) means that project identity alone predicts much of the outcome. Under a random split, projects, authors, and reviewers appear in both training and test sets, so models can exploit these shortcuts, and reported accuracies probably overstate performance on unseen projects.

5 Discussion

Our results suggest three practical lessons for work on PR outcome prediction. First, the main gain comes from modelling both modalities: adding comments to code, or code to comments, is worth about 1.6–1.9 pp, whereas the choice among cross-attention, concatenation, and several code encoders matters by tenths of a point. Practitioners with limited compute can therefore start with concatenation of pooled encoder outputs, which is simpler than cross-attention and statistically indistinguishable from it in most of our configurations. Second, fusion comparisons need multiple seeds: for CodeBERT+RoBERTa, the per-seed gap between cross-attention and concatenation ranges from -0.06 pp to $+1.83$ pp around a pooled mean of $+0.88$ pp, so a single seed can report anything from a near tie to a two-point cross-attention win. One of our seeds (42) happens to favour concatenation by a hair, while four favour cross-attention; the pooled comparison itself does not reach significance after Holm correction. Third, evaluation design may matter more than architecture. The strong association between project and outcome, near-duplicate texts across splits, and post-hoc features such as review duration can all inflate scores, and none of the architectural choices we compare addresses them. Whether the observed ordering of models survives project-held-out or chronological evaluation is an open question that this paper cannot answer.

6 Threats to Validity

6.0.0.1Internal validity.

The proposed hybrid, its fusion ablations, and CodeBERT-only were trained in an earlier software environment than the other models (Section 3.5), so part of any difference between these groups may reflect the environment. Input preprocessing also differs across some configurations: the share of diffs truncated at 512 tokens ranges from 26% to 46% across runs, and CodeReviewer Run 1 and Run 2 differ in formatting. We report Run 2 as the primary CodeReviewer result because its code representation matches that of the other code-only runs, and report Run 1 for transparency. The XGBoost surrogate may not capture all of the neural model's behaviour, so SHAP attributions computed on it are only an approximation.

6.0.0.2External validity.

The data are open-source, English-language, and 94.8% from GitHub, with Java and Python making up 80.4% of PRs. Results may not transfer to Gerrit- or GitLab-centred workflows or to under-represented languages. The 402,661 raw records were reduced to 33,955 (8.4%), and we cannot rule out selection bias in the filtering. The split is random over the whole period without project, author, reviewer, or time grouping; about 1.0–1.2% of test diffs and 2.5–2.7% of test review texts are exact matches (after whitespace normalisation) of training items, which is distinct from the duplicate-PR filter. Given $V = 0.60$ for project, we regard this as the most important limitation: absolute accuracies are likely optimistic for unseen projects.

6.0.0.3Construct validity.

The revise class is a mixture of closed-after-inactivity and still-open PRs, so a revise prediction means "outcome not determined" and not a distinct review verdict, and the open PRs are right-censored. Review comments may reveal the outcome (for example, a request for changes before a revise label), and comment-only RoBERTa reaches 0.8655 accuracy, which is consistent with either genuine signal or partial label leakage. Review duration is known only after the review ends and differs sharply by class (median 98,192 s for merge versus 78 s and 53 s for reject and revise); its contribution to accuracy was not isolated by an ablation. This study should therefore be read as a retrospective analysis, not as a model that could predict outcomes in real time.

6.0.0.4Conclusion validity.

Five seeds on one split capture initialisation and ordering variance, not data-split variance. Paired t-tests over five seeds have low power, McNemar's test is applied to one checkpoint per model, and the pooled cross-attention-versus-concatenation estimate uses non-independent pairs; "no significant difference" should not be read as equivalence. Plain RoBERTa and BiLSTM are tested at seed 42 only.

6.0.0.5Scope.

We do not evaluate LLM- or agent-based review systems. Such systems can be prompted or fine-tuned as classifiers, and a comparison is an important next step; we did not run one because of compute limits.

7 Conclusions and Future Work

We compared encoder-based fusion strategies for retrospective PR outcome prediction across five seeds on 33,955 PRs. A CodeBERT–RoBERTa hybrid with bidirectional cross-attention reaches 88.11% mean accuracy and beats its code-only counterpart in every seed ($+1.85$ pp), but cross-attention is statistically indistinguishable from concatenation and from several alternative encoder pairings, which together occupy a 0.42 pp band. The consistent gain comes from combining code and comments; the cross-attention–concatenation gap, by contrast, is small (approximately 0.9 pp) and seed-dependent, with per-seed values ranging from -0.06 pp to $+1.83$ pp, so a single-seed comparison could easily overstate or reverse the ordering. Because project identity is strongly associated with the outcome and the split is random, our absolute numbers probably overstate generalisation.

Future work should (i) repeat the comparison with project-held-out and chronological splits and after deduplicating texts across PRs, (ii) evaluate fine-tuned and prompted LLM classifiers, and (vi) measure surrogate fidelity out of sample and calibration of the uncertainty estimates.

Declarations

Not applicable.

The authors declare no competing interests.

This study used publicly available open-source review data. Usernames of authors and reviewers were used only for aggregate statistical analysis, and no individual-level inferences are drawn.

Not applicable.

Not applicable.

The curated dataset (33,955 PRs), the train/validation/test split indices, the per-instance model predictions, and the dataset-construction scripts are available at https://github.com/TimBroAhm/Code-Review-Model-Using-Deep-Learning.

All training code, evaluation scripts, and model weights are available at the same repository.

Both authors contributed equally to study design, implementation, analysis, and manuscript preparation.

99 Allamanis M, Barr ET, Devanbu P, Sutton C (2018) A survey of machine learning for big code and naturalness. ACM Comput Surv 51(4):1–37. https://doi.org/10.1145/3212695

Bacchelli A, Bird C (2013) Expectations, outcomes, and challenges of modern code review. In: Proceedings of the 35th International Conference on Software Engineering (ICSE), pp 712–721. https://doi.org/10.1109/ICSE.2013.6606617

Bergsma W (2013) A bias-correction for Cramér's V and Tschuprow's T. J Korean Stat Soc 42(3):323–328. https://doi.org/10.1016/j.jkss.2012.10.002

Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 785–794. https://doi.org/10.1145/2939672.2939785

Chicco D, Jurman G (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21(1):6. https://doi.org/10.1186/s12864-019-6413-7

Dietterich TG (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput 10(7):1895–1923. https://doi.org/10.1162/089976698300017197

Feng Z, Guo D, Tang D, Duan N, Feng X, Gong M, Shou L, Qin B, Liu T, Jiang D, Zhou M (2020) CodeBERT: a pre-trained model for programming and natural languages. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp 1536–1547. https://doi.org/10.18653/v1/2020.findings-emnlp.139

Gal Y, Ghahramani Z (2016) Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: Proceedings of the 33rd International Conference on Machine Learning (ICML), pp 1050–1059

Gorodkin J (2004) Comparing two K-category assignments by a K-category correlation coefficient. Comput Biol Chem 28(5–6):367–374

Gousios G, Pinzger M, van Deursen A (2014) An exploratory study of the pull-based software development model. In: Proceedings of the 36th International Conference on Software Engineering (ICSE), pp 345–355. https://doi.org/10.1145/2568225.2568260

Guo D, Ren S, Lu S, Feng Z, Tang D, Liu S, Zhou L, Duan N, Svyatkovskiy A, Fu S, Tufano M, Deng SK, Clement C, Drain D, Sundaresan N, Yin J, Jiang D, Zhou M (2021) GraphCodeBERT: pre-training code representations with data flow. In: Proceedings of the 9th International Conference on Learning Representations (ICLR)

Guo D, Lu S, Duan N, Wang Y, Zhou M, Yin J (2022) UniXcoder: unified cross-modal pre-training for code representation. In: Proceedings of the 60th Annual Meeting of the

Association for Computational Linguistics (ACL), pp 7212–7225. https://doi.org/10.18653/v1/2022.acl-long.499

Li HY, Shi ST, Thung F, Huo X, Xu B, Li M, Lo D (2019) DeepReview: automatic code review using deep multi-instance learning. In: Proceedings of the 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp 318–330. https://doi.org/10.1007/978-3-030-16145-3_25

Li Z, Lu S, Guo D, Duan N, Jannu S, Jenks G, Majumder D, Green J, Svyatkovskiy A, Fu S, Sundaresan N (2022) Automating code review activities by large-scale pre-training. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), pp 1035–1047. https://doi.org/10.1145/3540250.3549081

Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692

Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems 30 (NeurIPS), pp 4765–4774

McIntosh S, Kamei Y, Adams B, Hassan AE (2014) The impact of code review coverage and code review participation on software quality: a case study of the Qt, VTK, and ITK projects. In: Proceedings of the 11th Working Conference on Mining Software Repositories (MSR), pp 192–201. https://doi.org/10.1145/2597073.2597076

Murphy-Hill E, Jaspan C, Egelman C, Cheng L (2022) The pushback effects of race, ethnicity, gender, and age in code review. Commun ACM 65(3):52–57. https://doi.org/10.1145/3474097

Tsay J, Dabbish L, Herbsleb J (2014) Influence of social and technical factors for evaluating contribution in GitHub. In: Proceedings of the 36th International Conference on Software Engineering (ICSE), pp 356–366. https://doi.org/10.1145/2568225.2568315

Tufano R, Dabic O, Mastropaolo A, Ciniselli M, Bavota G (2024) Code review automation: strengths and weaknesses of the state of the art. IEEE Trans Softw Eng 50(2):338–353. https://doi.org/10.1109/TSE.2023.3348172

Tufano R, Martin-Lopez A, Tayeb A, Dabic O, Haiduc S, Bavota G (2024) Deep learning-based code reviews: a paradigm shift or a double-edged sword? arXiv preprint arXiv:2411.11401

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Advances in Neural Information Processing Systems 30 (NeurIPS), pp 5998–6008

Wang Y, Wang W, Joty S, Hoi SCH (2021) CodeT5: identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In: Proceedings of the

2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 8696–8708. https://doi.org/10.18653/v1/2021.emnlp-main.685

Wu B, Zhang X (2022) Contrastive learning for multi-modal automatic code review. In: Proceedings of the 34th International Conference on Software Engineering and Knowledge Engineering (SEKE), pp 54–59. https://doi.org/10.18293/SEKE2022-022

Figures

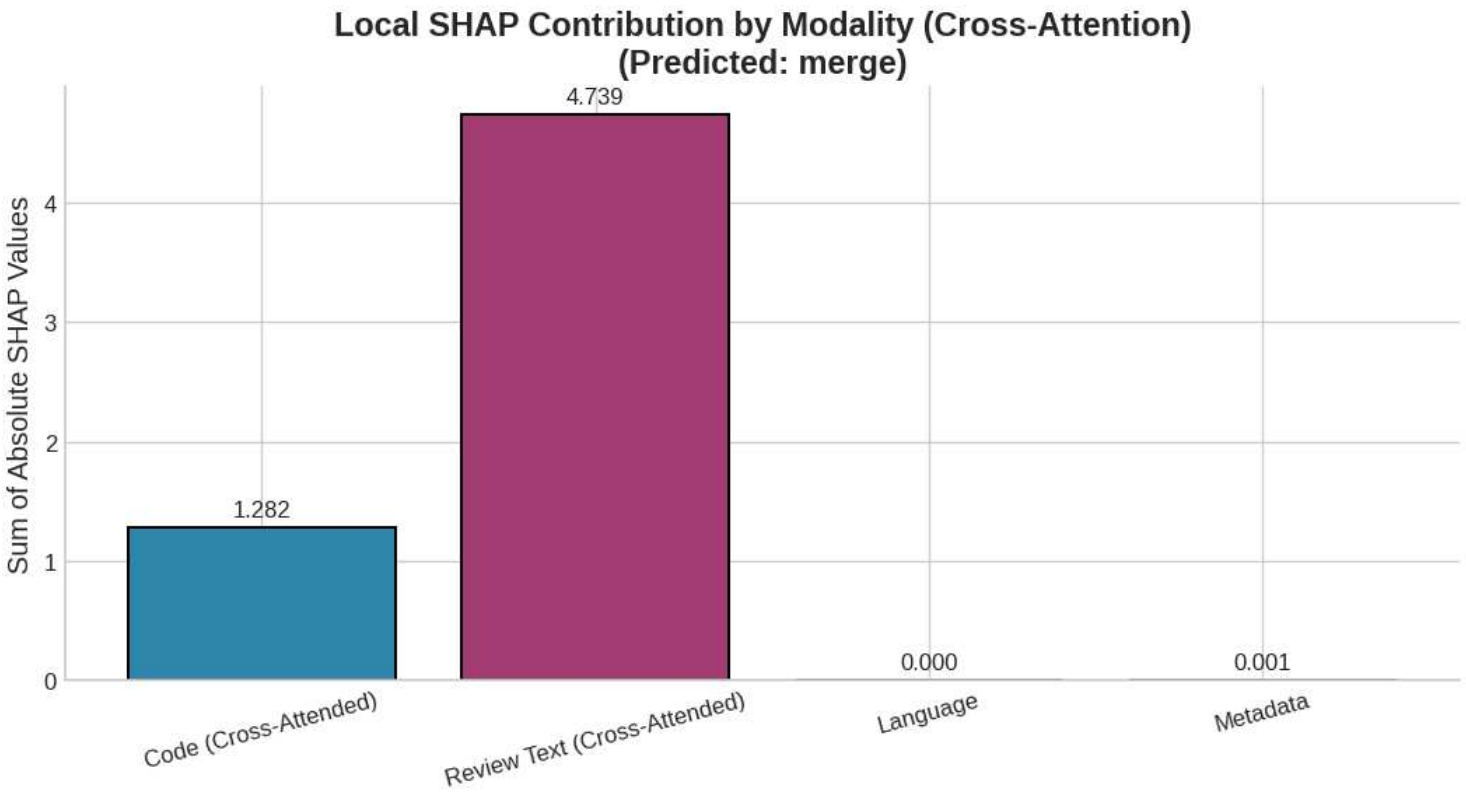


Figure 1

- Blue: Code (Cross-Attended)
- Purple: Review Text (Cross-Attended)