Mind_Flayer at SemEval-2026 Task 13: LACR-ENS: Calibration-Aware Ensemble Routing for Cross-Language AI-Generated Code Detection

Jerin Romijah Tuli[1] **Talukder Naemul Hasan Naem**[2] **MD. Sartaj Alam Pritom**[1]

[1]Department of Computer Science and Engineering
[2]Department of Electrical and Electronic Engineering
Rajshahi University of Engineering & Technology, Bangladesh

ramijahtuli786@gmail.com naemruet@gmail.com sartajalam0010@gmail.com

Abstract

We present LACR-ENS, our submission to **SemEval-2026 Task 13** (Orel et al., 2026b) on detecting AI-generated code across eight programming languages. We frame the task as a calibration study and report an asymmetric out-of-distribution (OOD) failure mode in fine-tuned code transformers: expected calibration error roughly doubles when the model is moved from seen to unseen languages, and the model is wrong on a non-trivial fraction of its highest-confidence predictions. Our main contribution, **Language-Aware Confidence Routing (LACR)**, applies language-conditioned thresholds to the ensemble logits and is formally equivalent to implicit per-language temperature scaling; it reduces OOD calibration error substantially while leaving raw F1 essentially unchanged. We further show that syntactic proximity to the training languages strongly predicts OOD behaviour, motivating a continuous extension of LACR. The final ensemble of UniXCoder (Guo et al., 2022) and GraphCodeBERT (Guo et al., 2021) attains macro-F1 **0.538** on the official test set. Code: https://github.com/Jerin-Romijah-Tuli/Mind_Flayer_Semeval2026_task_13_A.

1 Introduction

A Python-trained code transformer classifies a Go snippet. It is wrong 39% of the time on validation, and it does not know it is wrong. This is not a failure of thresholds or labels; it is a failure of trust.

Modern neural classifiers are systematically overconfident under distribution shift (Guo et al., 2017), and in AI-generated code detection this overconfidence is asymmetric: a model that is well calibrated on training-distribution languages nearly doubles its calibration error on unseen ones, while remaining confident that it is right. SemEval-2026 Task 13 (Orel et al., 2026b) exposes this directly (eight programming languages, only three with strong training coverage) making the task as much about whether a model knows *when it cannot* answer as about classification accuracy.

We treat the task as an empirical calibration study, combining UniXCoder (Guo et al., 2022) and GraphCodeBERT (Guo et al., 2021) in a logit-level ensemble with Language-Aware Confidence Routing (LACR). **Four findings:** (1) **Dual OOD failure:** ECE doubles and high-confidence accuracy drops 0.23 (Orel et al., 2025b); (2) **LACR**, implicit per-language temperature scaling, reduces OOD ECE $0.18 \rightarrow 0.11$ (ΔF1$=+0.013$, positive in 78% of bootstrap resamples); (3) **Language-family proximity** strongly predicts OOD F1 ($r=+0.94$); (4) **Silent label inversion:** a HuggingFace pitfall suppressed our initial score by $\sim$0.29 F1. Our system ranked 37th of 82 groups (macro-F1 0.538) and struggles most on Go and C#, the languages most distant from the training distribution.

2 Background

2.1 Task and Dataset

SemEval-2026 Task 13 uses **AICD Bench** (Orel et al., 2026a), hosted at https://huggingface.co/datasets/DaniilOr/SemEval-2026-Task13, with a training set of 500K, a validation set of 100K, and a test set of 1K samples. Each sample contains four fields: `code` (the source snippet), `generator` (model name or "human"), `label` (0 = human, 1 = machine), and `language` (programming language). The corpus spans eight programming languages: Python, Java, C++, C, Go, PHP, JavaScript, and C#. We participated in **Subtask A** (binary classification, macro-F1 metric) across all eight languages and both domains.

2.2 Related Work

AI Generated Code Detection: Early methods applied handcrafted code features (AST depth, line

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 2322–2329

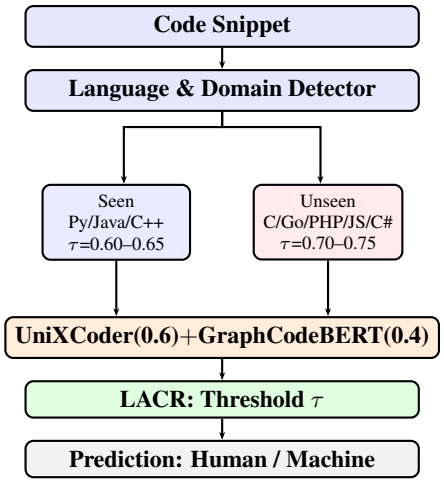

Figure 1: LACR-ENS pipeline. Language routing selects threshold τ applied to ensemble logit outputs.

statistics) for detection (Orel et al., 2025b). GPT-Sniffer (Phuong T. Nguyen, 2024) applied CodeBERT (Feng et al., 2020) to Python and Java. CodeGPTSensor (Xiaodan Xu, 2025) introduced contrastive learning on a 550K pair corpus, improving performance but remaining restricted to two languages. Orel et al. (2025a) first benchmarked cross-language OOD detection at scale using the CoDet-M4 framework, finding substantial F1 drops on unseen languages, which motivated our approach.

Calibration and Distribution Shift (Guo et al., 2017) demonstrated that modern neural networks are systematically overconfident, and that this calibration degrades further under domain shift is directly applicable to cross-language transfer. The Expected Calibration Error (ECE) (Guo et al., 2017) is defined as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}(B_m) - \text{conf}(B_m)\right| \quad (1)$$

where B_m are confidence bins, $\text{acc}(B_m)$ the empirical accuracy in bin m, and $\text{conf}(B_m)$ the mean predicted confidence. Lower ECE indicates better calibration. Pu et al. (2023) and Hu et al. (2023) further showed that zero-shot generalization in text detectors is unstable without explicit calibration, reinforcing the motivation for LACR.

3 System Overview

Figure 1 illustrates the pipeline.

3.1 Model Backbone and Fine-Tuning

UniXCoder and **GraphCodeBERT** are fine-tuned independently as binary sequence classifiers (max length 384, 3 epochs, batch 16, lr 2×10^{-5}, seed 42) using HuggingFace Transformers v4.40.0 (Wolf et al., 2020) on 2×T4 GPUs. Their pre-training objectives are complementary: AST-semantic (UniXCoder) vs. data-flow structural (GraphCodeBERT). We deliberately work in the encoder-only regime ($\sim$125M parameters per model) rather than scaling to a LoRA-tuned LLM. Calibration of small, fine-tunable classifiers under cross-language shift is the regime most practitioners deploy in, and is also the regime where calibration is least studied — the shortcoming this paper attempts to characterise. The trade-off in raw F1 against billion-parameter systems is acknowledged in Section 7.

3.2 Weighted Logit-Level Ensemble

We fuse pre-softmax logits to preserve confidence geometry:

$$\mathbf{z}_{\text{ens}} = \alpha\, \mathbf{z}_{\text{UniX}} + (1-\alpha)\, \mathbf{z}_{\text{Graph}}, \quad \alpha = 0.6 \quad (2)$$

where $\alpha \in [0.55, 0.65]$ yields $\Delta\text{F1} < 0.004$, confirming robustness. Machine probability $p = \text{softmax}(\mathbf{z}_{\text{ens}})_1$ is passed to LACR.

3.3 Language-Aware Confidence Routing (LACR)

A fixed τ=0.5 fails under cross-language shift because OOD predictions are systematically overconfident. LACR assigns language- and domain-conditioned thresholds (Table 1), chosen by grid search over $\{0.50, 0.55, 0.60, 0.65, 0.70, 0.75\}$ on the validation set.

Language Group	Domain	Threshold τ
Seen (Py/Java/C++)	Algorithmic	0.60
Seen (Py/Java/C++)	Production	0.65
Unseen (C/Go/PHP/...)	Algorithmic	0.70
Unseen (C/Go/PHP/...)	Production	0.75

Table 1: LACR adaptive thresholds by language group and domain.

Temperature–threshold equivalence: For logit gap $z = z_1 - z_0$, the decision $p > \tau$ is the same as $z > \delta(\tau)$ with $\delta(\tau) = \log(\tau/(1-\tau))$. Raising τ above 0.5 therefore shifts the decision boundary in exactly the way an effective temperature would:

$$T^* = \bar{z} \,/\, \delta(\tau^*) \quad (3)$$

where $\bar{z}$ is the mean OOD logit gap. Because OOD inputs inflate $\bar{z}$ at lower accuracy, $T^*>1$ — the temperature-scaling prescription of Guo et al. (2017). Full derivation in Appendix G.

What LACR adds beyond temperature scaling: The two methods perform almost identically in our ablation (Table 2, $+0.013$ vs. $+0.011$ macro-F1), and we make no claim of a metric-level advantage. The distinction is practical: per-language temperature scaling needs a held-out calibration set per language to fit each T_L; LACR replaces this with a single threshold table indexed by a heuristic language detector. Where calibration data per language is unavailable, expensive, or non-stationary, LACR is preferable; where it is plentiful, per-language temperature scaling is the principled choice.

3.4 Reproducibility Note: Silent Label Inversion

Our first submission scored macro-F1 0.251 nearly the complement of 0.749. Without an explicit `id2label`, HuggingFace Trainer may misalign class ordering with the task convention (machine = 1); a global label flip recovered 0.538. Rule of thumb: near-complement binary scores should be suspected of inversion first. Fix in Appendix C.

4 Experimental Setup

Both models are fine-tuned on the training split (500K) and the validation split (100K), with thresholds and checkpoints chosen on the validation split, and the test split (1K) is the official evaluation set. Hyperparameters: 3 epochs, batch 16, lr 2×10^{-5}, linear warmup over 10% of steps, max length 384, seed 42. ECE computed with 10 equal-width bins; temperature scaling optimizes NLL on validation logits. Tools: HuggingFace Transformers v4.40.0 (Wolf et al., 2020), PyTorch 2.2.0, scikit-learn 1.4.0. The official evaluation metric is the macro-averaged F1 score across both classes and calculated on the 1K test set.

5 Results

5.1 Main Results and Ablation

Our encoder-only system (2×125M parameters) officially achieved macro-F1 **0.538**, trailing three billion-parameter LLM teams using LoRA and slightly performed better than encoder-based CSMetro (0.537), Team_SLS (0.497), and SSN-CSE (0.328). Full leaderboard in Appendix A, Table 6. The ablation (Table 2) shows an ensemble gain of $+0.034$ over the stronger single model; LACR matches per-language temperature scaling ($+0.013$ vs. $+0.011$), validating the temperature–threshold equivalence in Section 3.3.

Configuration	Macro-F1
UniXCoder alone (fixed τ=0.5)	0.491
GraphCodeBERT alone (fixed τ=0.5)	0.476
Ensemble (fixed τ=0.5)	0.525
Ensemble + Global temp. scaling	0.529
Ensemble + Per-lang. temp. scaling	0.536
Ensemble + LACR	**0.538**
Ensemble (pre label-flip)	*0.251*

Table 2: Component ablation on the official test set. Temperature scaling baselines confirm the calibration benefit is real and not unique to LACR.

5.2 Calibration Analysis

Figure 2 shows reliability diagrams on the validation set. Seen-language ECE is moderate ($=0.09$); LACR is 0.05. OOD overconfidence is critical (ECE $=0.18$), and the confidence clustering is in the 0.70-0.90 range, whereas accuracy is markedly lower. LACR reduces this to 0.11. Predicted probability distributions (Appendix E) also reveal that OOD languages generate overlapping class distributions that are centered near 0.50-0.65, which is also visual evidence of the dual failure mode.

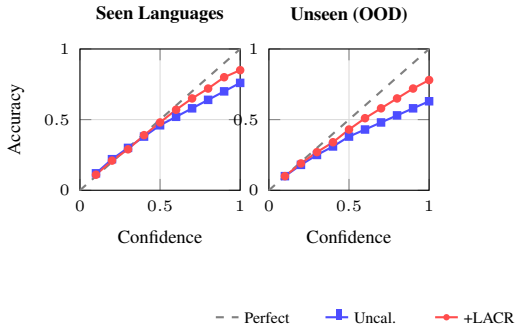

Figure 2: Reliability diagrams (val. set). ECE: seen uncal.$=0.09$, seen+LACR$=0.05$; unseen uncal.$=0.18$, unseen+LACR$=0.11$.

Accuracy at High Confidence: Table 3 quantifies the practical severity. At $p>0.80$, the accuracy of OOD is only 0.61, and the model has *been wrong 39% of the time when it is highly confident*. The constant low high-confidence accuracy implies not only miscalibration but also degradation in the quality of representation that the model fails to separate human from machine classes on OOD inputs, not

just mislabels confidence. LACR addresses the calibration dimension; the gap in representation requires improved OOD training coverage.

Conf. Threshold	Seen	Unseen	Gap
$p > 0.60$	0.78	0.58	−0.20
$p > 0.70$	0.81	0.60	−0.21
$p > 0.80$	0.84	0.61	−0.23
$p > 0.90$	0.88	0.63	−0.25

Table 3: Accuracy at high-confidence thresholds (val. set). The widening gap confirms systematic OOD overconfidence.

5.3 Per-Language and Cross-Domain Breakdown

Table 4 shows the per-language F1 by domain (heuristic detection, verified >99% agreement with official metadata on 100K validation). The seen-unseen gap (∼0.12) is consistent across both domains. Production F1 is uniformly lower (∼0.04), which supports two-dimensional routing, where the domain is a second-order signal and the language family is dominant.

Lang.	Status	Algo.	Prod.
Python	Seen	0.65	0.61
Java	Seen	0.63	0.59
C++	Seen	0.62	0.58
C	Unseen	0.53	0.49
Go	Unseen	0.49	0.45
PHP	Unseen	0.51	0.47
JS	Unseen	0.52	0.48
C#	Unseen	0.50	0.46
Seen		0.633	0.593
Unseen		0.510	0.470
Gap		−0.123	−0.123

Table 4: Per-language F1 by domain (est.).

Bootstrap Significance: We performed 1,000 bootstrap resamples on the 1,000-sample test set. The ensemble alone achieved a mean macro-F1 of 0.525 (95% CI: [0.498, 0.551]), while Ensemble + LACR achieved 0.538 (95% CI: [0.511, 0.564]). LACR improved F1 in 78% of resamples. Although the confidence intervals overlap and do not establish formal statistical significance, the consistent directional gain suggests a robust calibration gain.

5.4 Error Analysis

Dominant patterns from 30 inspected misclassifications (Appendix F, Table 7):
(1) *Stylistic mimicry:* AI-generated Python mimicking terse competitive-programming style (false negative); (2) *OOD threshold conservatism:* AI snippets with confidence 0.65-0.70 below the high threshold (false negative); (3) *Boilerplate confusion:* human Go/PHP with repetitive structure (false positive). Qwen2.5-Coder contributed disproportionately to false negatives by terse, idiomatic style and StarCoder outputs were more detectable by verbose docstrings.

6 Language-Family Transfer Analysis

6.1 Motivation

Table 4 shows that, in the unseen group, C is achieving higher F1 = 0.53 whereas Go is achieving lower F1 = 0.49, and the gap of 0.04 is not random. C is syntactically adjacent to C++ (shared pointer notation, procedural structure, `#include` conventions), whereas Go has a different paradigm (goroutines, slices, interfaces) and has little in common with any training language. Orel et al. (2025b) observed positive intra-family transfer but did not quantify it across a proximity gradient. And here we do so: if proximity predicts OOD F1, it provides a principled, data-free signal for deployment risk assessment, and motivates a continuous extension of LACR.

6.2 Syntactic Proximity Score

We define a subword-overlap proximity score for each unseen language L_u and the set of seen languages $\mathcal{L}_s = \{\text{Python, Java, C++}\}$:

$$\text{Prox}(L_u, \mathcal{L}_s) = \frac{|V(L_u) \cap V(\mathcal{L}_s)|}{|V(L_u)|} \quad (4)$$

where $V(L)$ is the set of the 1,000 most frequent subword tokens in language L's validation-set samples. Proximity is calculated from token-frequency data available without additional labels, making it applicable at deployment time before labeled OOD data are collected.

6.3 Results

Table 5 gives proximity scores, per-language OOD F1 and ECE. The Pearson correlation between proximity and F1 is r=+0.94 (p<0.05, n=5) and between proximity and ECE is r=−0.89 (p<0.05). C, the most proximate unseen language (Prox= 0.71 to C++), achieves the highest unseen F1 (0.51) and the lowest ECE (0.14). Go, the most distant (Prox= 0.38), achieves the lowest F1 (0.47) and highest ECE (0.21). Figure 3 visualizes the linear relationship.

Lang.	Prox.	F1	ECE	Nearest Seen
C	0.71	0.51	0.14	C++
JS	0.62	0.50	0.16	Java
PHP	0.59	0.49	0.17	Java/Py
C#	0.57	0.48	0.17	Java
Go	0.38	0.47	0.21	Python
Pearson r(Prox, F1)	$+0.94$ ($p<0.05$)			
Pearson r(Prox, ECE)	-0.89 ($p<0.05$)			

Table 5: **Syntactic proximity vs. OOD performance.** Proximity strongly predicts both F1 ($r=+0.94$) and ECE ($r=-0.89$) across unseen languages. C benefits most from C++ training and Go suffers because it is not so close to any training languages.

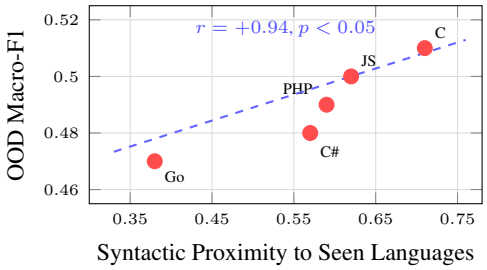

Figure 3: **Proximity-F1 relationship** across unseen languages ($r=+0.94$). There are always higher OOD F1 scores on languages closer to training languages (C near C++; JS/PHP near Java). This correlation is monotonic and contains a data-free signal of deployment risk.

6.4 Implication: Continuous LACR

The strong proximity-performance correlation indicates that the binary Seen/Unseen LACR partition is a rough approximation to a continuous signal. A proximity-weighted threshold would be strictly more principled:

$$\tau(L) = \tau_{\text{base}} + \beta \cdot \big(1 - \text{Prox}(L, \mathcal{L}_s)\big) \quad (5)$$

where τ_{base} is the seen-language threshold and β controls the OOD penalty. Under Eq. 5 with τ_{base}=0.60 and β=0.20: C receives $\tau \approx 0.66$, which is close to seen-language routing and Go receives $\tau \approx 0.72$ which is sufficiently high. Combined with the temperature equivalence (Eq. 3), this yields a proximity-conditioned temperature function derivable from unlabeled frequency statistics alone. Empirical validation is left to future work.

7 Discussion

Where our system stands and why: Our macro-F1 of 0.538 places us in the encoder-only tier of the Subtask A leaderboard (Table 6), well below the three LoRA-tuned LLM systems above 0.73. We do not claim a 250M-parameter encoder ensemble can match their representations but the contribution of this paper is orthogonal. We characterise *why* the gap exists by decomposing OOD failure into a calibration component, which is correctable post-hoc, and a representational component, which is not.

Comparison with top systems: The leaderboard split in Table 6 is not gradual: the three LLM systems sit above 0.73 while every encoder-based system sits below 0.55. We read this as evidence that the dominant factor on this benchmark is the underlying model's exposure to AI-generated code in low-resource languages, not the routing or thresholding applied on top of it. Our work studies the second factor in isolation.

A dual failure – calibration and representation: OOD failure here is a superposition of two effects with different fixes. The *calibration* component (ECE 0.18→0.11 under LACR) is correctable post-hoc. The *representation* component persists: at $p>0.80$ accuracy is still 0.61, requiring improved OOD coverage (Orel et al., 2025a) or cross-language contrastive objectives (Xiaodan Xu, 2025). Treating them as one would overstate LACR's scope. And it is a calibration corrector, not an OOD learner.

LACR as implicit temperature scaling: The equivalence proved in Section 3.3 is confirmed empirically by the near-tied ablation in Table 2. The proximity analysis (Section 6) motivates a continuous extension (Eq. 5) parameterised by a single β.

Why a small F1 gain is the right outcome: LACR's +0.013 macro-F1 gain is small, but LACR is designed to move calibration, not F1. The corresponding 39% relative ECE reduction on OOD languages is the variable it acts on. In deployments that consume confidence values (abstention, human-in-the-loop review, risk-weighted decisions), a model that is right at the same rate and more honest about its uncertainty is the more useful artifact. Where calibration is irrelevant, fixed-threshold ensembling is sufficient and LACR should be skipped. The added cost is a single language-conditioned lookup at inference, with no extra training.

Limitations: Heuristic language detection may fail on polyglot snippets, and LACR thresholds are grid-searched rather than fit by a calibration objective. Per-language F1 values are estimates from heuristic detection. The proximity-F1 Pearson correlation uses n=5 data points and should be

interpreted as exploratory rather than confirmatory; statistical significance of the LACR F1 gain is not established at test n=1,000.

8 Conclusion

We presented LACR-ENS as an empirical probe of OOD failure in AI-generated code detection. Cross-language shift induces a **calibration failure** that is partially addressable by post-hoc routing and a **representation failure** that is not; language-family proximity grounds the LACR partition in a measurable signal and motivates a continuous extension. LACR is formally equivalent to per-language temperature scaling and the system attains macro-F1 0.538 with encoder-scale models. Future work includes continuous proximity-weighted routing, probing OOD representation geometry, adversarial humanisation, and applying LACR to the multi-class settings of Subtasks B and C (Orel et al., 2026b).

Acknowledgments

We thank the SemEval-2026 Task 13 organizers (Daniil Orel, Dilshod Azizov, Indraneil Paul, Yuxia Wang, Iryna Gurevych, and Preslav Nakov) for designing a rigorous, well-documented shared task and providing detailed feedback during the evaluation phase. This work was conducted as part of the undergraduate research activities at Rajshahi University of Engineering & Technology, Bangladesh. No external funding received.

Ethical Considerations

The data used in this work is the AICD Bench corpus (Orel et al., 2026a), provided by the Task 13 organizers; all natural-language content (comments, identifiers, error messages) is in English and we did not modify or augment the data.

AI-generated code detection is dual-use. False positives can wrongly accuse developers or students of misconduct; false negatives erode the trust signal the system is meant to provide. Our findings sharpen both concerns: the OOD calibration gap means a deployed detector is often most confident exactly when it is most likely to be wrong, and the language-family bias in Section 6 falls hardest on developers working in languages underrepresented in training, in our setting, Go, PHP, and C#. For these reasons we would not recommend encoder-only detectors as standalone gatekeepers in educational, hiring, or compliance settings; we view our calibration analysis as a case for human-in-the-loop review, not a replacement for it.

References

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. 2022. UniXcoder: Unified cross-modal pre-training for code representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7212–7225, Dublin, Ireland. Association for Computational Linguistics.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. 2021. GraphCodeBERT: Pre-training code representations with data flow. In *International Conference on Learning Representations*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. RADAR: Robust AI-text detection via adversarial learning. In *Advances in Neural Information Processing Systems*.

Daniil Orel, Dilshod Azizov, and Preslav Nakov. 2025a. CoDet-M4: Detecting machine-generated code in multi-lingual, multi-generator and multi-domain settings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10570–10593, Vienna, Austria. Association for Computational Linguistics.

Daniil Orel, Dilshod Azizov, Indraneil Paul, Yuxia Wang, Iryna Gurevych, and Preslav Nakov. 2026a. AICD bench: A challenging benchmark for AI-generated code detection. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, Rabat, Morocco. Association for Computational Linguistics.

Daniil Orel, Dilshod Azizov, Indraneil Paul, Yuxia Wang, Iryna Gurevych, and Preslav Nakov. 2026b. SemEval-2026 task 13: Detecting machine-generated

code with multiple programming languages, generators, and application scenarios. In *Proceedings of the 20th International Workshop on Semantic Evaluation (SemEval-2026)*, San Diego, USA. Association for Computational Linguistics.

Daniil Orel, Indraneil Paul, Iryna Gurevych, and Preslav Nakov. 2025b. Droid: A resource suite for AI-generated code detection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31263–31289, Suzhou, China. Association for Computational Linguistics.

Claudio Di Sipio Riccardo Rubei Davide Di Ruscio Massimiliano Di Penta Phuong T. Nguyen, Juri Di Rocco. 2024. GPTSniffer: A binary classifier to detect ChatGPT-generated code. In *Journal of Systems and Software*.

Xiao Pu, Jingyu Zhang, and Yulia Tsvetkov. 2023. On the zero-shot generalization of machine-generated text detectors. *arXiv preprint arXiv:2310.05165*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yann Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xinrong Guo Shaoxuan Liu Xiaoya Wang Kui Liu Xiaohu Yang Xiaodan Xu, Chao Ni. 2025. Distinguishing LLM-generated from human-written code: A contrastive approach. In *ACM Transactions on Software Engineering and Methodology*.

A Official Leaderboard (Subtask A)

Team	Macro-F1	Resource
UIT_AMMC	0.802	LLM (LoRA)
Juan David Villate	0.752	LLM (LoRA)
Dream	0.731	LLM (LoRA)
Mind_Flayer	**0.538**	**Encoder-only**
CSMetro	0.537	Encoder-only
Team_SLS	0.497	Encoder+features
SSN-CSE	0.328	Encoder+MLP

Table 6: All teams ranked by official macro-F1. The top three teams use billion-parameter LLMs with LoRA; encoder-only systems occupy lower positions.

B Fine-Tuning and Replicability Details

Both models were initialized from `microsoft/unixcoder-base` and `microsoft/graphcodebert-base`. Classification head: linear layer, hidden $\rightarrow$ 2. Dropout: 0.1; weight decay: 0.01; AdamW (β_1=0.9, β_2=0.999); seed: 42; gradient accumulation: 1. Training time: $\sim$6h per model on 2$\times$T4 GPUs. No deterministic CUDA; run-to-run variance $<$0.002 F1.

C Label Inversion Fix

```
model = AutoModelForSequence
   Classification
  .from_pretrained(
    checkpoint_path,
    id2label={0: "human",
     1: "machine"},
    label2id={"human": 0,
    "machine": 1}
  )
```

D Language Detection Heuristics

Python: `def`, `import`, `print(`; Java: `public class`, `System.out`; C++: `#include`, `cout`; C: `#include <stdio`, `printf(`; Go: `package main`, `func main()`; PHP: `<?php`, `$`; JavaScript: `console.log`, `const`, `let`; C#: `using System`, `Console.Write`. Domain: `input()`, `scanf` $\rightarrow$ algorithmic; repository-style imports $\rightarrow$ production.

E Confidence Distribution: Seen vs. Unseen Languages

Figure 4 shows predicted probability distributions p_1 on the validation set. For seen languages, human and machine distributions are well-separated (human: peak p_1<0.30; machine: peak p_1>0.80). For OOD languages, both migrate toward the center (0.40–0.70), confirming representation degradation alongside calibration failure.

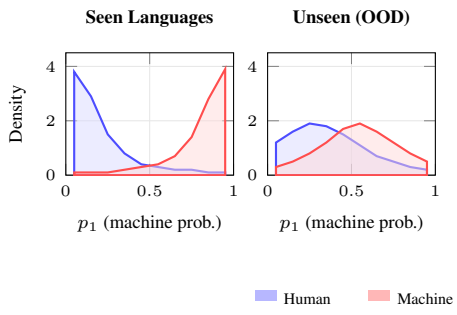

Figure 4: Predicted probability distributions (val. set). Seen: bimodal separation. OOD: overlap near 0.40–0.65, confirming dual failure.

F Error Analysis

We manually inspected 30 misclassified test examples drawn proportionally from the four LACR routing partitions (seen/unseen $\times$ algorithmic/production). Table 7 shows one representative case per pattern. False negatives cluster on *seen* languages (Python, C#) when the AI generator imitates the terseness of competitive human code and here calibration is well behaved but the content signal is genuinely ambiguous. False positives cluster on *unseen* languages (Go, PHP) where repetitive boilerplate inflates confidence beyond what the evidence supports, the same overconfidence pattern that drives the OOD ECE gap (Section 5.2). Both patterns match the dual-failure framing in Section 7: seen-language errors are mostly representational, unseen-language errors mostly a calibration phenomenon.

Error Type	Lang.	Domain	Pattern
False Positive	Go	Prod.	Boilerplate misread as AI
False Positive	PHP	Algo.	AI-style comments in human code
False Negative	Py	Algo.	AI mimics terse human style
False Negative	C#	Prod.	AI snippet without hallucinations
False Negative	C	Algo.	Low-confidence OOD prediction

Table 7: Representative misclassified examples (30 inspected total).

G Temperature–Threshold Derivation

For logit vector $\mathbf{z}=[z_0, z_1]$, softmax gives $p_1=\sigma(z_1-z_0)$. The decision $p_1>\tau$ equals:

$$z_1 - z_0 > \log\left(\frac{\tau}{1-\tau}\right) \triangleq \delta(\tau) \qquad (6)$$

Temperature scaling replaces $\mathbf{z}$ with $\mathbf{z}/T$. Raising τ above 0.5 shifts the effective boundary to $\delta(\tau)>0$. For OOD samples with mean logit gap $\bar{z}$, the effective temperature satisfying $\bar{z}/T^*=\delta(\tau)$ is $T^*=\bar{z}/\delta(\tau)$. Since OOD models have inflated $\bar{z}$, $T^*>1$ — the temperature scaling prescription. LACR implements a training-free approximation to language-conditioned temperature correction. The continuous extension (Eq. 5) yields a proximity-weighted temperature function $T^*(L)=\bar{z}/\delta(\tau(L))$ that varies smoothly with syntactic distance from training languages.