Automated Software Engineering
ASE 2026 Munich
October 12–16, 2026

Software Research Lab

iSE LAB
Interactive Software Engineering and Analytics Lab

A Unified Model for Cross-Domain Clone Detection via Model Merging

ASE 2026 · Munich, Germany · October 12–16, 2026



Palash Ranjan Roy



Banani Roy



Kevin A. Schneider



Chanchal K. Roy

Department of Computer Science, University of Saskatchewan, Canada

University of Saskatchewan

BE WHAT THE WORLD NEEDS

Monday morning, a developer fixes one bug

```
int sumFirst(int[] a, int n) {
    int total = 0;
    for (int i = 1; i < n; i++)
        total += a[i];
    return total;
}
```

The loop starts at 1, so the first element is never counted.

The fix is one character. Test passes, ticket closed.

But this function was copied months ago, before the fix.

Where are the copies, and do they still carry the bug?

Copy one was renamed and reformatted

Copy 1 · same language
renamed and reformatted

```
int addAll(int[] v, int k) {
  int s = 0;
  for (int j = 1; j < k; j++)
    s = s + v[j];
  return s;
}
```

✓ Java detector flags it





Copy two was ported to Python

Copy 1 · same language
renamed and reformatted

```
int addAll(int[] v, int k) {
  int s = 0;
  for (int j = 1; j < k; j++)
    s = s + v[j];
  return s;
}
```

✓ Java detector flags it

Copy 2 · ported to Python
same logic, another language

```
def sum_first(a, n):
    total = 0
    for i in range(1, n):
        total += a[i]
    return total
```

✗ Java detector says not a clone

?

Copy three was rewritten by an AI assistant

Copy 1 · same language
renamed and reformatted

```
int addAll(int[] v, int k) {
  int s = 0;
  for (int j = 1; j < k; j++)
    s = s + v[j];
  return s;
}
```

✓ Java detector flags it

Copy 2 · ported to Python
same logic, another language

```
def sum_first(a, n):
    total = 0
    for i in range(1, n):
        total += a[i]
    return total
```

✗ Java detector says not a clone

Copy 3 · rewritten by AI
same behaviour, new shape

```
int sum3(int[] a, int n) {
  return IntStream
    .range(1, n)
    .map(i -> a[i])
    .sum();
}
```

✗ Looks unrelated, so it is missed

2 of 3

copies still carry the bug

Our detector only recognized the copy that looked like the original

Our lab built a benchmark for each disguise



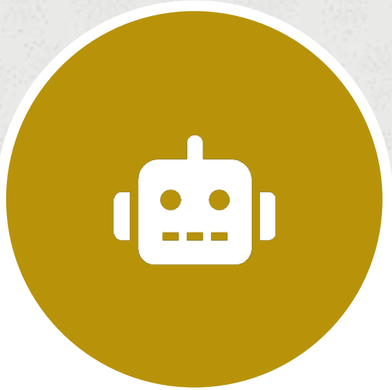
2014 · BigCloneBench

Same-language clones, like copy one



2019 · CLCDSA

Cross-language clones, like copy two



2023 · GPTCloneBench

AI-generated clones, like copy three



The obvious fix is one model trained on everything

Multi-task UniXcoder on 1.38M BigCloneBench and CLCDSA pairs reaches 0.927 combined F1 in domain.


0.927
Home Domain F1
0.151
Unseen AI Clones F1
On unseen AI-generated clones
GPTCloneBench F1, worse than every single specialist
Recall falls to 0.089
University of Saskatchewan
BE WHAT THE WORLD NEEDS
Software Research Lab
ISE LAB
Intelligent Software Engineering and Analytics Lab
ASE 2026 · CLONE MERGING

WHAT IF WE MERGE INSTEAD?

Merge task vectors, not data

Task vector $\delta_k = \theta_k - \theta_{base}$

Merged model $\theta_{merged} = \theta_{base} + \Sigma \lambda_k \delta_k$



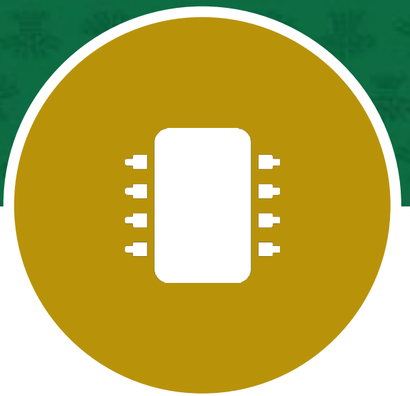
NO TRAINING DATA

Works on checkpoints only



NO GRADIENTS

Pure tensor arithmetic



MINUTES ON A CPU

Under five minutes per merge

We put merging through four questions

RQ1 How well do clone detectors generalize across domains?

RQ2 What conditions make parameter-level merging succeed?

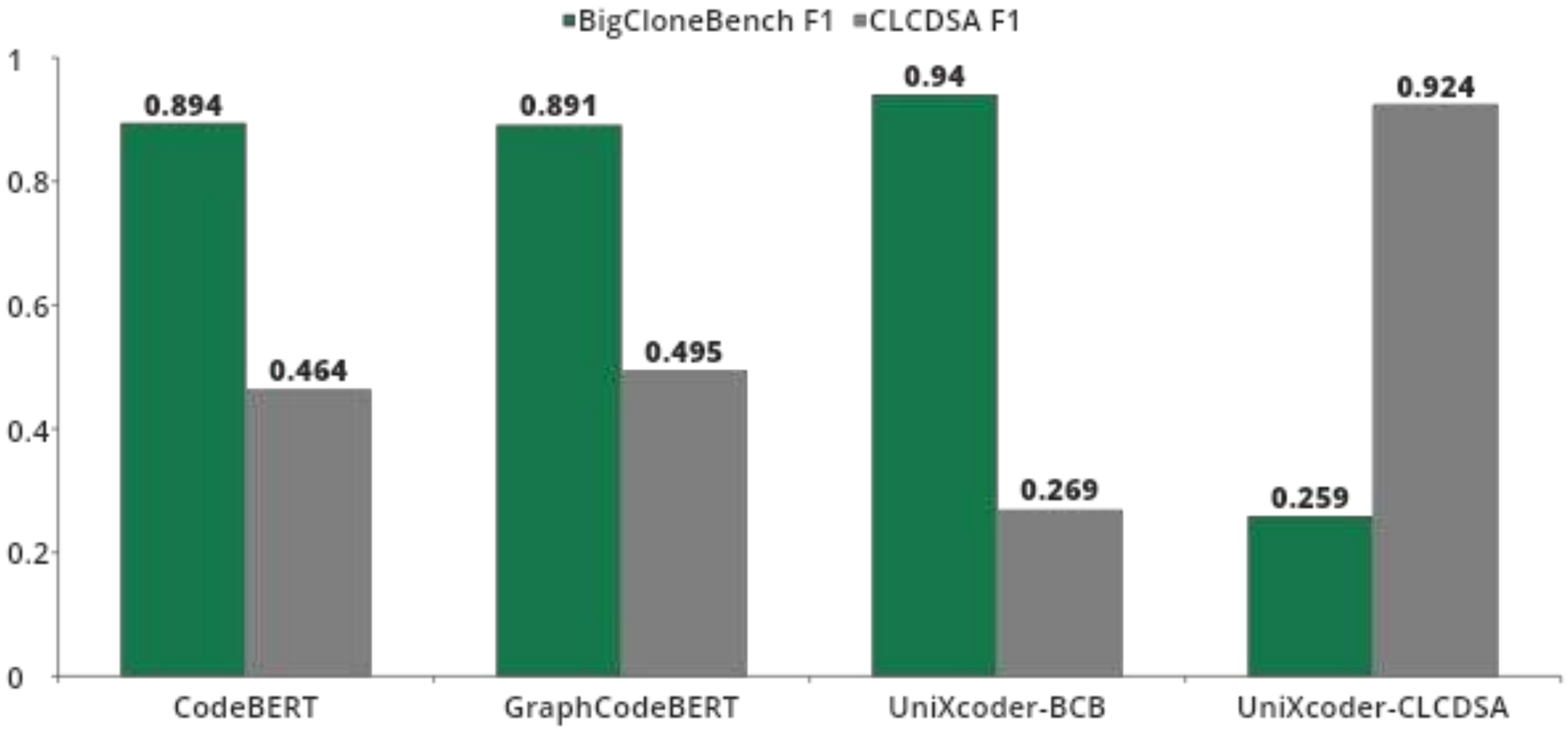
RQ3 What do layers capture, and can layer stitching exploit it?

RQ4 Do merged models generalize to unseen AI-generated clones?

5	merging methods	**4**	code models
3	benchmarks	**12**	configurations
3	seeds per same-base result		

Each specialist only recognizes its own disguise

F1 of each specialist at home and away



0.940 → 0.269

71% DROP

UniXcoder-BCB on cross-language clones. Every specialist loses 44 to 72% outside its home domain.

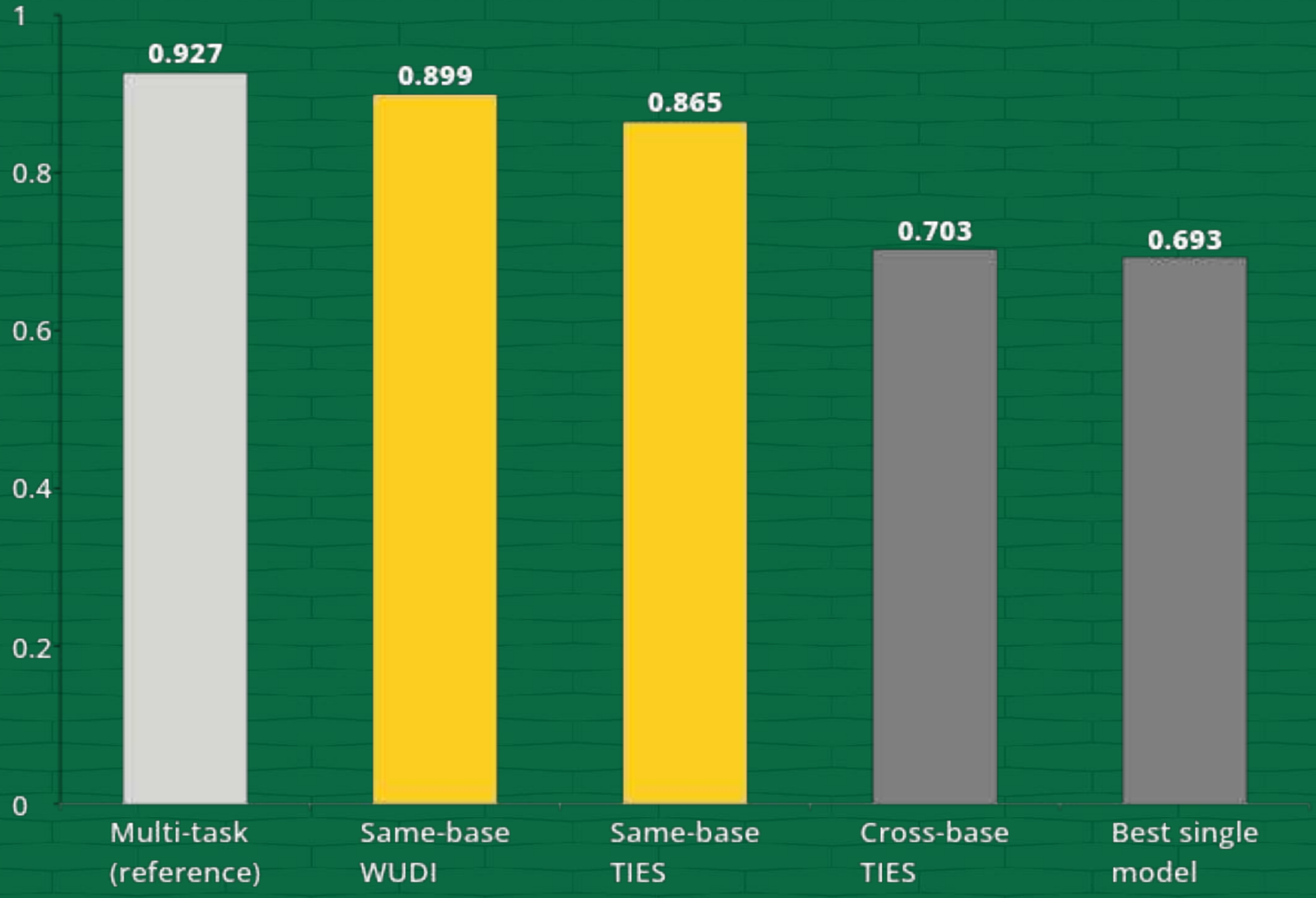
No single model reaches a combined F1 above 0.693.

RQ2 · PARAMETER MERGING

Merging works when the models share a parent

Same base reaches 93 to 97% of multi-task, with no data at the merge step.

Cross base barely beats one model, for all five methods.



Same-base task vectors point in the same direction

Same base



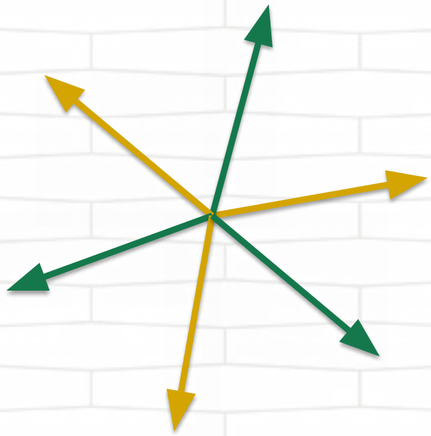
59.5%

sign agreement

0.032

cosine similarity

Cross base



50.4%

sign agreement

0.001

cosine similarity

Cross-base sign agreement is a coin flip, so TIES has no majority sign to elect. A shared base gives it one.

Layer stitching reveals what each specialist learned

Greedy layer selection for the same-base UniXcoder merge



cross-language layer (CLCDSA)

same-language layer (BigCloneBench)

Different bases

Stitching beats parameter merging

0.738 vs 0.703

Same base

Parameter merging beats stitching

smooth beats hard layer choices

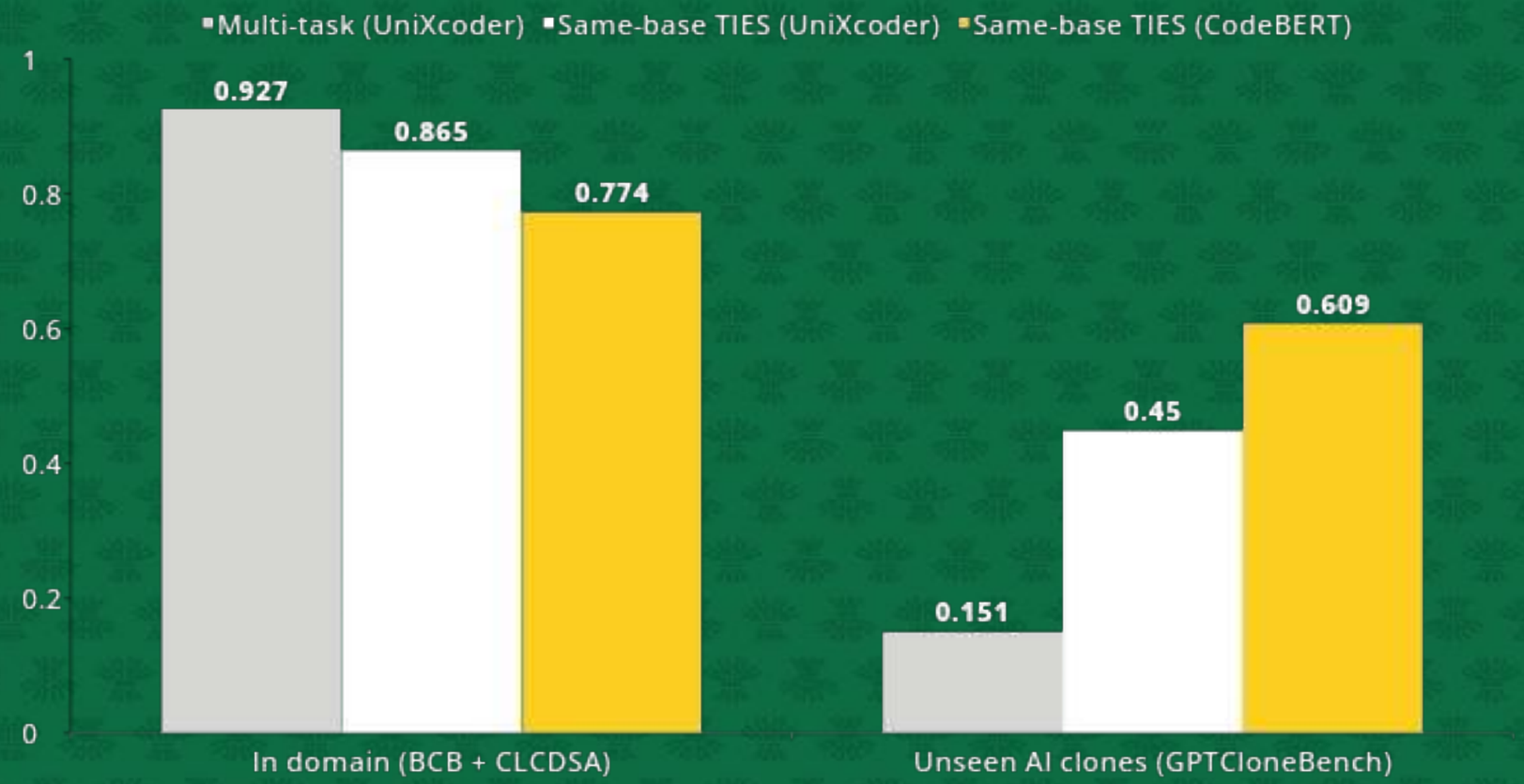
Different tokenizers

Linear alignment is never selected

a barrier deeper than vocabulary

RQ4 · UNSEEN AI CLONES

The twist is that merging wins where it matters most



4×

better than multi-task on unseen AI clones (0.609 vs 0.151)

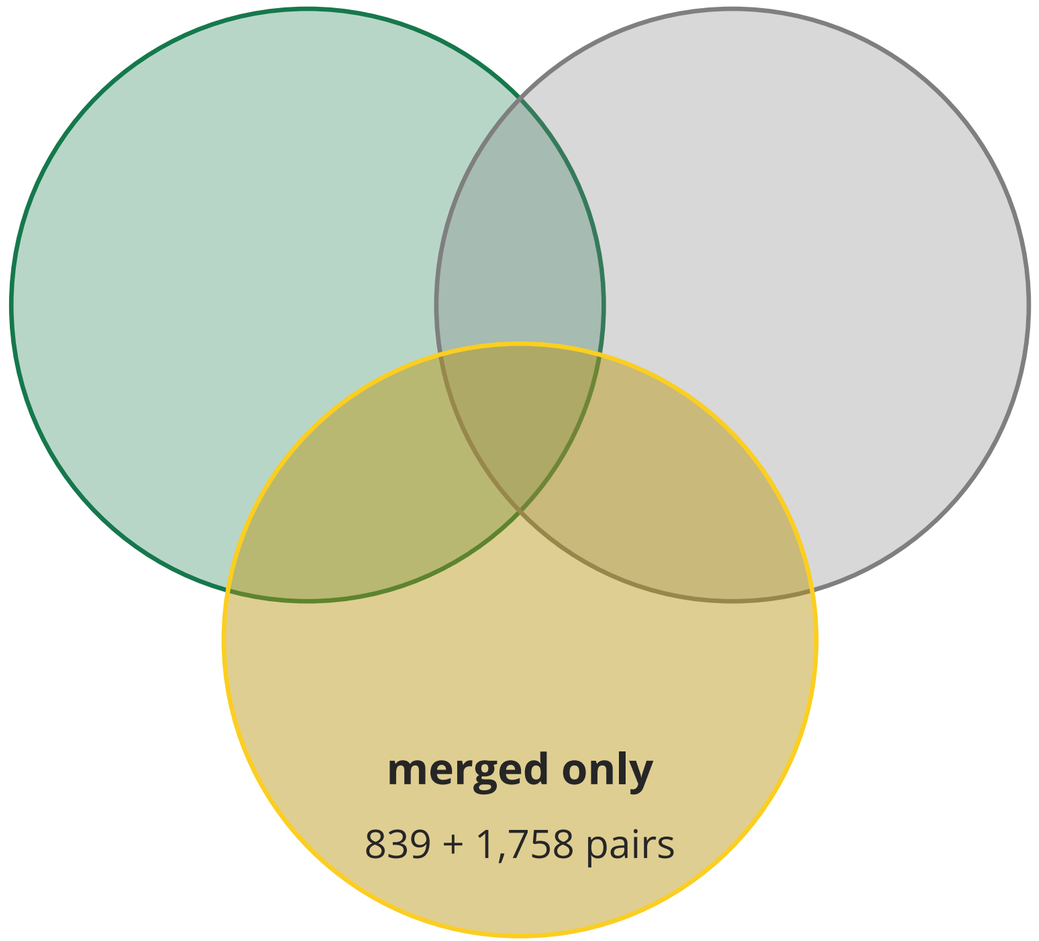
0.454

strongest zero-shot code LLM we tested, at far higher cost per pair

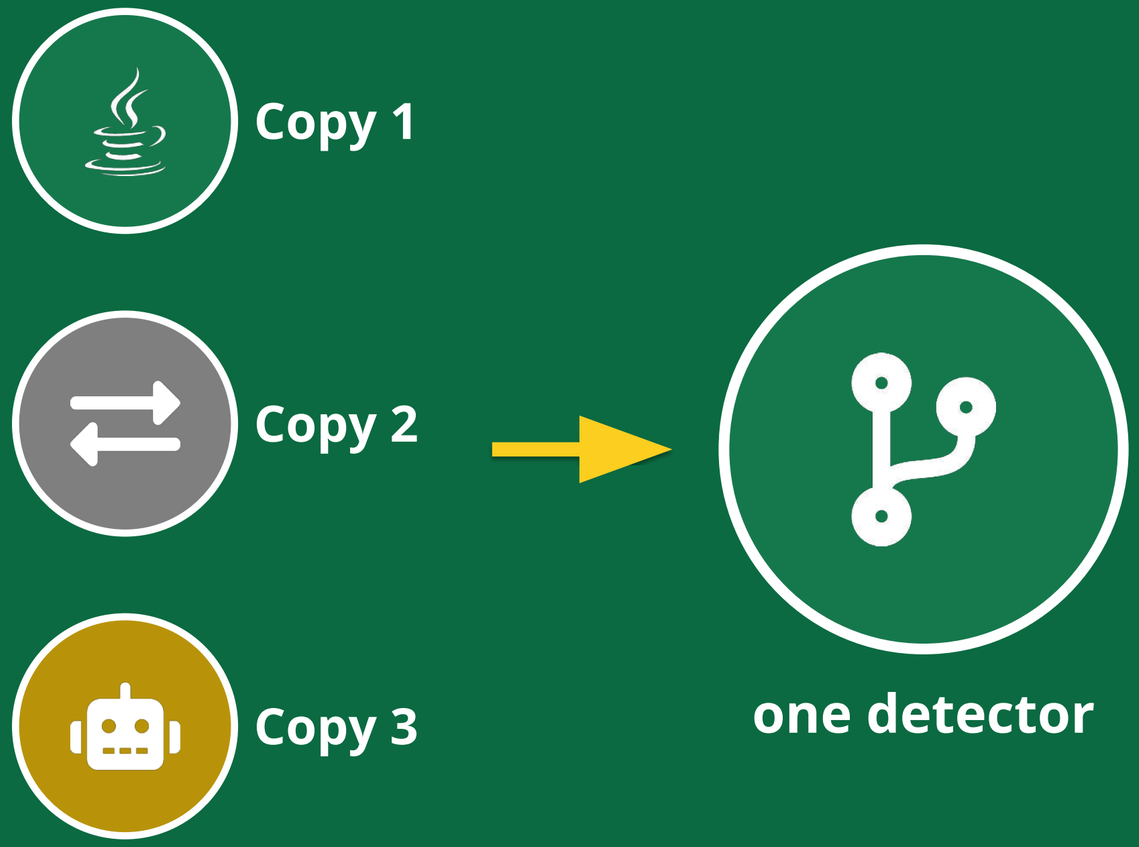
WHY MERGING GENERALIZES

The merged model knows things neither parent knew

- 839 BigCloneBench pairs and 1,758 CLCDSA pairs only the merged model gets right
- Every comparison significant under McNemar's test, $p < 0.001$
- Same cost as one model at inference time



Illustrative, region sizes not to scale



Back to Monday's bug

1. Pick one shared base model
2. Fine-tune one copy per domain
3. Merge with TIES, in minutes on a CPU
4. Add the next domain by re-merging

Try it live at the MergeSE tool demo
Wed 14 Oct, 15:35 to 15:45 · Forum 8

Three things to take home



THE BASE BINDS

Same base works for all five methods. Cross base barely beats one model.



HOME VS UNSEEN

Multi-task wins at home. Merging wins on unseen AI clones, 0.609 vs 0.151.



DATA-FREE RECIPE

Fine-tune per domain, merge with TIES, add domains later.

palashroy.me

Thank you

Read the paper

Palash R. Roy · palash.roy@usask.ca

Software Research Lab

iSE LAB
Interactive Software Engineering and Analytics Lab

Automated Software Engineering
ASE 2026 Munich
October 12–16, 2026

BACKUP

Complete results across the three benchmarks

Model	BCB F1	CLCDSA F1	Combined F1	GPTCloneBench F1
CodeBERT-BCB	0.894	0.464	0.679	0.450
GraphCodeBERT-BCB	0.891	0.495	0.693	0.376
UniXcoder-BCB	0.940	0.269	0.605	0.390
UniXcoder-CLCDSA	0.259	0.924	0.592	0.410
Multi-task (UniXcoder)	0.933	0.922	0.927	0.151
Cross-base TIES	0.841	0.566	0.703	n/a
Same-base TIES (UniXcoder)	**.890 ± .010**	**.839 ± .027**	**.865 ± .012**	**.450 ± .042**
Same-base TIES (CodeBERT)	**.801 ± .018**	**.746 ± .012**	**.774 ± .014**	**.609 ± .049**
Layer stitching (CB + GCB)	0.856	0.620	0.738	0.528
Layer stitching (UX + UX)	0.842	0.743	0.793	0.339

Same-base TIES rows are means ± std over three seeds (42, 2, 3). WUDI reaches the highest in-domain combined F1 (0.899). Full tables with all five methods are in the paper.

BACKUP

Multi-task fails on AI clones because recall collapses

Model	BCB P	BCB R	CLCDSA P	CLCDSA R	GPT P	GPT R
UniXcoder-BCB	.942	.937	.563	.177	.505	.318
UniXcoder-CLCDSA	.155	.788	.949	.900	.483	.356
Multi-task	**.949**	**.919**	**.947**	**.898**	**.517**	**.089**
Same-base TIES	.905	.896	.871	.813	.516	.468
Layer stitching (CB + GCB)	.820	.895	.509	.793	.507	.551

Precision stays near 0.5 for every model on GPTCloneBench. The difference is recall.

The multi-task model finds only 9 of every 100 AI-generated clones. Same-base TIES finds about 47.

BACKUP

Same-base merging also works for CodeBERT

Model	BCB F1	CLCDSA F1	Combined F1	GPTCloneBench F1
CodeBERT-BCB	0.894	0.464	0.679	0.450
CodeBERT-CLCDSA	0.000	0.924	0.462	0.000
CodeBERT TIES merge	**.801 ± .018**	**.746 ± .012**	**.774 ± .014**	**.609 ± .049**

The CodeBERT-CLCDSA specialist collapses completely on BigCloneBench (0.000 F1). It predicts every pair as non-clone.

Merging still recovers both domains, 14% above the best individual CodeBERT in combined F1.

It also gives the highest GPTCloneBench score in the study, 0.609.

BACKUP

Common questions about our design choices

Why recommend TIES if WUDI scores higher?

WUDI has the best in-domain combined F1 (0.899), but TIES generalizes better to unseen AI-generated clones. Robustness to the unknown is the point of the recipe.

Why not just ensemble the specialists?

An ensemble averages answers, so when both specialists are wrong it is wrong too. It also needs both models at inference, twice the parameters and compute. The merged model is one model.

What does merging cost?

A one-time step of under five minutes on a CPU. No training data and no GPU are needed.

Can I weight one domain more?

Yes. The merge weights can be changed without retraining to favour either domain.

BACKUP

Threats to validity

Internal validity

Greedy layer search may not find the global optimum, and results may depend on random seeds. We report three seeds (42, 2, 3). Every seed beats multi-task on GPTCloneBench, though variance there is higher.

External validity

We study RoBERTa-style encoder models on Java and Java to Python. Results on two model families suggest the finding generalizes, but other languages and decoder-only models remain open. BigCloneBench label concerns are known.

Construct validity

Combined F1 weights both domains equally. GPTCloneBench uses 10,000 sampled Java pairs. Any overlap between BCB and CLCDSA would favour the multi-task baseline, since merging never sees the data.