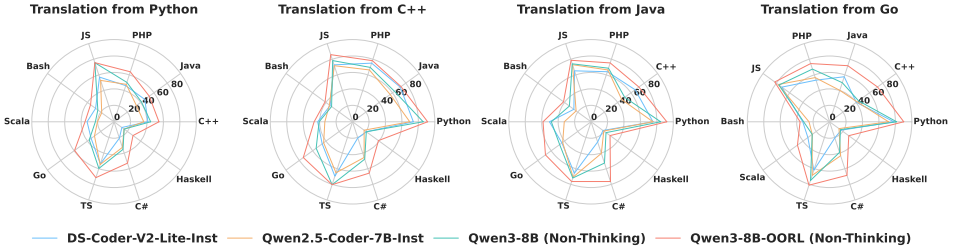
On-Policy Optimization with Group Equivalent Preference for Multi-Programming Language Understanding

Haoyuan Wu
The Chinese University of Hong Kong, Hong Kong SAR

Outline

Introduction

The Coding Capability Gap in LLMs



Code translation performance on our CrossPLEval benchmark reveals a significant capability gap between different programming languages for SOTA models.

Code Translation

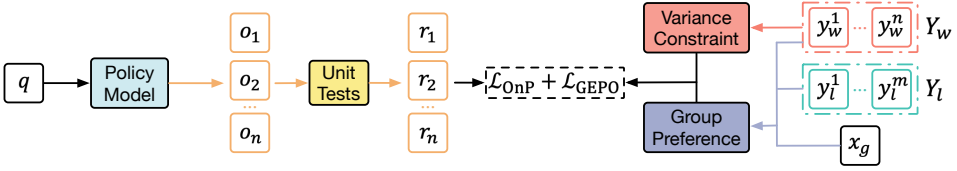
- Code Translation among different high-level programming languages.
- Code Translation between high-level programming languages and low-level intermediate representations (IRs).

Contributions

- We introduce **OORL**, a novel RL framework integrating on-policy and off-policy strategies to bridge the language capability gap via **code translation tasks**.
- We develop **GEPO**, a novel preference optimization method that extends beyond pairwise comparisons by explicitly modeling functional equivalence within groups of IRs.
- Extensive evaluations show our method achieves superior performance across various code benchmarks in multiple programming languages, including low-resource ones.

Method

OORL: A Hybrid RL Framework



Overview of OORL, which combines on-policy RL with rule-based rewards for correctness and off-policy preference optimization (GEPO) for nuanced understanding.

- We train the LLM on **code translation tasks** (e.g., Python to Java).
- A simple, **binary rule-based reward** is used:

$$R_{\text{rule}}(q, o) = \begin{cases} 1, & \text{if translation is successful (compiles + passes unit tests),} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

- This provides a strong but coarse-grained signal for task completion.

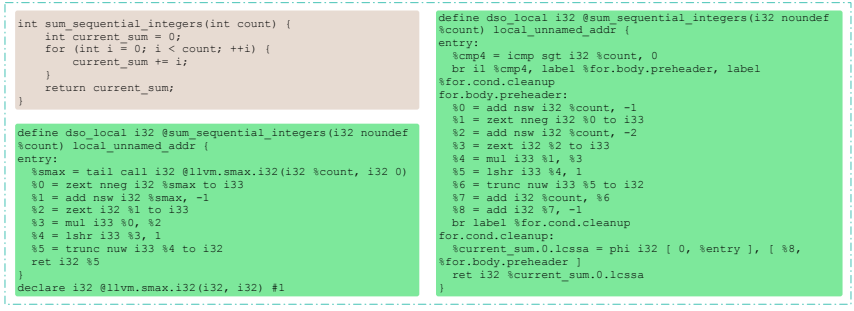
Limitation of Rule-Based Rewards

The binary reward is coarse-grained. It cannot distinguish between multiple valid implementations or provide process-level supervision on **functional equivalence**.

Our Solution: Preference Optimization with IRs

We use compiler Intermediate Representations (IRs), which are language-agnostic, to capture fine-grained code functionality. This allows for a more nuanced learning signal.

Group Equivalent Preference Optimization (GEPO)



```
int sum_sequential_integers(int count) {
    int current_sum = 0;
    for (int i = 0; i < count; ++i) {
        current_sum += i;
    }
    return current_sum;
}
```

```
define dso_local i32 @sum_sequential_integers(i32 noundef
%count) local_unnamed_addr {
entry:
  %smax = tail call i32 @llvm.smax.i32(i32 %count, i32 0)
  %0 = zext nneg i32 %smax to i33
  %1 = add nsw i32 %smax, -1
  %2 = zext i32 %1 to i33
  %3 = mul i33 %0, %2
  %4 = lshr i33 %3, 1
  %5 = trunc nuw i33 %4 to i32
  ret i32 %5
}
declare i32 @llvm.smax.i32(i32, i32) #1
```

```
define dso_local i32 @sum_sequential_integers(i32 noundef
%count) local_unnamed_addr {
entry:
  %cmp4 = icmp sgt i32 %count, 0
  br i1 %cmp4, label %for.body.preheader, label
%for.cond.cleanup
for.body.preheader:
  %0 = add nsw i32 %count, -1
  %1 = zext nneg i32 %0 to i33
  %2 = add nsw i32 %count, -2
  %3 = zext i32 %2 to i33
  %4 = mul i33 %1, %3
  %5 = lshr i33 %4, 1
  %6 = trunc nuw i33 %5 to i32
  %7 = add i32 %count, %6
  %8 = add i32 %7, -1
  br label %for.cond.cleanup
for.cond.cleanup:
  %current_sum.0.lcssa = phi i32 [ 0, %entry ], [ %8,
%for.body.preheader ]
  ret i32 %current_sum.0.lcssa
}
```

Source code x_g and corresponding function equivalenct IRs group Y_w

```
define dso_local i32 @sum_sequential_integers(i32 noundef
%count) local_unnamed_addr {
entry:
  %smax = tail call i32 @llvm.smax.i32(i32 %count, i32 0)
  %0 = add nsw i32 %smax, -1
  %1 = zext i32 %0 to i33
  %2 = mul i33 %0, %1
  %3 = lshr i33 %2, 1
  %4 = trunc nuw i33 %3 to i32
  ret i32 %4
}
declare i32 @llvm.smax.i32(i32, i32) #1
```

```
define dso_local i32 @sum_sequential_integers(i32 noundef
%count) local_unnamed_addr {
entry:
  %smax = tail call i32 @llvm.smax.i32(i32 %count, i32 0)
  %0 = zext nneg i32 %smax to i33
  %1 = add nsw i32 %smax, -1
  %2 = zext i32 %1 to i33
  %3 = mul i33 %0, %2
  %4 = lshr i33 %3, 1
  ret i32 %4
}
declare i32 @llvm.smax.i32(i32, i32) #1
```

Inequivalenct IRs group Y_l, augmented from equivalenct IRs group Y_w

GEPO trains on groups of "winner" (functionally equivalent) IRs and "loser" (inequivalent) IRs, learning both preference and equivalence.

$$\mathcal{L}_{\text{GEPO}} = \underbrace{-\mathbb{E}[\log \sigma(\beta(\frac{1}{n}\sum \hat{r}_w - \frac{1}{m}\sum \hat{r}_l))]}_{\text{Preference Term: Winners > Losers on average}} + \underbrace{\lambda \mathbb{E}[\text{Var}(\hat{r}_w)]}_{\text{Equivalence Term: Penalize variance within winners}}$$

- The first term pushes the model to prefer the winner group over the loser group.
- The second term explicitly encourages the model to view all IRs within the winner group as equally good.

Experiments

Experiment Settings

- **Base Model**: Qwen3-8B
- **Training Strategy**: OORL (On-policy REINFORCE++ and off-policy GEPO)
- **Baselines**:
  - DS-Coder-V2-Lite-Inst, Qwen2.5-Coder-7B-Inst, Qwen3-8B
- **Benchmarks**:
  - **MultiPL-E**: Code generation in 8 languages.
  - **CrossPLEval**: Code translation across 11 languages.

Main Results: MultiPL-E Benchmark

Table: Pass@1 performance on MultiPL-E. Our Qwen3-8B-OORL significantly outperforms all baselines.

	Python	C++	Java	PHP	Bash	JS	TS	C#	Avg.
DS-Coder-V2-Lite-Inst	81.10	32.91	68.35	72.67	19.62	81.36	83.33	41.13	60.06
Qwen2.5-Coder-7B-Inst	88.40	57.14	70.88	73.91	43.67	78.88	84.27	52.53	68.71
Qwen3-8B	82.60	71.42	70.25	50.31	36.07	83.22	84.27	42.41	65.07
Qwen3-8B-OORL	**90.06**	**83.23**	**83.54**	**78.26**	**46.20**	**89.44**	**85.53**	**54.22**	**76.31**

Main Results: CrossPLEval (Code Translation)

Table: Full Pass@1 results on CrossPLEval, showing translation from four source languages to ten target languages.

	Model	Target Language											
		Python	C++	Java	PHP	JS	Bash	Scala	Go	TS	C#	Haskell	Avg.
Py	DS-Coder-V2-Lite-Inst	-	41.17	46.21	46.21	57.14	27.73	32.77	34.45	53.78	21.00	11.76	37.22
	Qwen2.5-Coder-7B-Inst	-	36.13	33.61	47.89	52.94	18.54	20.16	29.41	54.62	32.77	15.12	34.12
	Qwen3-8B	-	44.53	41.17	50.42	75.54	24.36	22.68	37.25	59.66	35.29	15.13	40.60
	Qwen3-8B-OORL	**-**	**54.62**	**52.94**	**63.94**	**75.63**	**35.29**	**35.29**	**59.32**	**71.42**	**52.94**	**28.48**	**52.99**
C++	DS-Coder-V2-Lite-Inst	73.91	-	71.73	75.00	72.82	33.69	42.39	45.65	69.56	20.65	20.65	52.61
	Qwen2.5-Coder-7B-Inst	69.56	-	56.52	66.30	71.73	34.78	33.69	43.47	65.21	45.65	17.39	50.43
	Qwen3-8B	85.86	-	63.04	69.56	78.26	31.52	40.65	54.71	80.43	47.82	20.65	57.25
	Qwen3-8B-OORL	**91.30**	**-**	**72.82**	**78.15**	**85.86**	**41.30**	**46.73**	**74.02**	**80.43**	**65.65**	**38.91**	**67.52**
Java	DS-Coder-V2-Lite-Inst	75.28	66.29	-	64.04	65.16	25.84	50.56	43.82	65.82	25.84	17.97	50.06
	Qwen2.5-Coder-7B-Inst	75.28	46.06	-	66.29	73.03	22.47	32.58	42.69	70.78	40.44	19.10	48.87
	Qwen3-8B	84.64	50.56	-	68.53	74.15	32.58	48.31	47.19	71.91	52.80	22.47	55.31
	Qwen3-8B-OORL	**92.13**	**71.91**	**-**	**75.28**	**78.65**	**41.34**	**58.42**	**68.53**	**76.40**	**76.40**	**28.53**	**66.76**
Go	DS-Coder-V2-Lite-Inst	78.26	40.21	57.60	53.26	71.73	32.60	32.60	-	61.95	20.65	17.39	46.63
	Qwen2.5-Coder-7B-Inst	71.73	41.30	40.21	56.52	77.17	25.00	26.08	-	68.47	43.47	14.13	46.41
	Qwen3-8B	81.52	43.47	50.00	67.39	76.08	31.52	26.08	-	75.00	39.13	16.30	50.65
	Qwen3-8B-OORL	**90.21**	**72.06**	**71.73**	**74.45**	**82.39**	**35.86**	**48.26**	**-**	**81.08**	**68.47**	**28.80**	**65.33**

Ablation Studies

Table: Contribution of each component in OORL.

Model	On-Policy RL	Off-Policy RL	MultiPL-E (Avg)	CrossPLEval (Avg)
Qwen3-8B (Base)	-	-	65.06	50.96
+ REINFORCE++	✓	-	73.60 (+8.54)	58.46 (+7.50)
+ REINFORCE++	✓	DPO	73.48	58.70
+ REINFORCE++ (OORL)	✓	**GEPO**	**76.31 (+11.25)**	**63.15 (+12.19)**

- **On-policy RL** provides a major boost over the base model.
- **GEPO** provides significant additional gains over standard DPO, proving the value of modeling functional equivalence.

Conclusion

Conclusion

- We proposed **OORL**, a hybrid RL framework to improve the multi-language capabilities of LLMs via code translation.
- Our key innovation, **GEPO**, leverages groups of Intermediate Representations to teach the model about functional equivalence, providing a richer, more nuanced training signal than standard methods.
- Our model, **Qwen3-8B-OORL**, establishes a new state-of-the-art on both code generation (MultiPL-E) and code translation (CrossPLEval) benchmarks, demonstrating effective generalization even to unseen and low-resource languages.

THANK YOU!