Building Scalable and Responsible Enterprise AI with Open-Source Foundations

Invited Talk, Business Analytics Program, Cornell University, 2026

Serdar Kadıoğlu

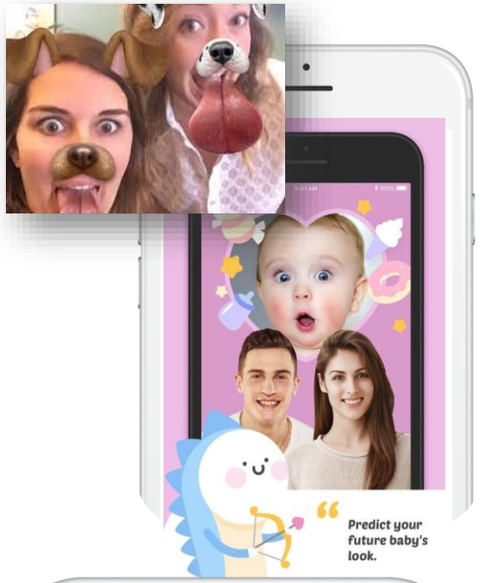
Group VP, AI Center, Fidelity Investments

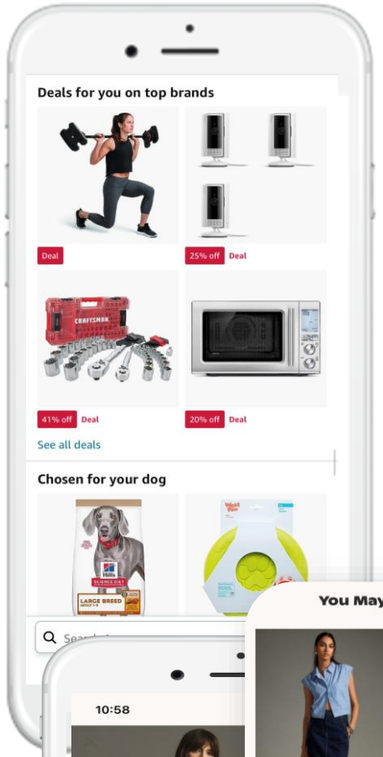
Adj. Associate Professor, Computer Science, Brown University

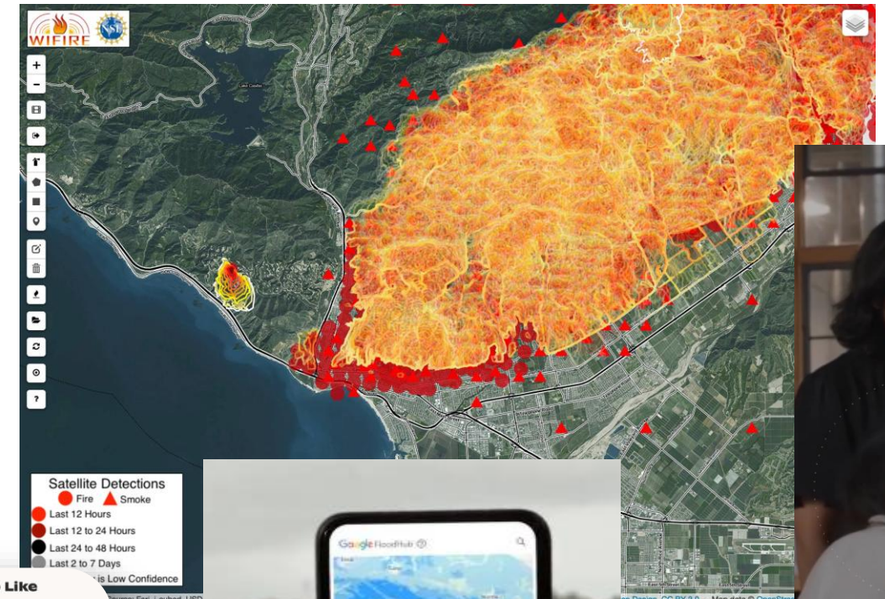
skadio.github.io

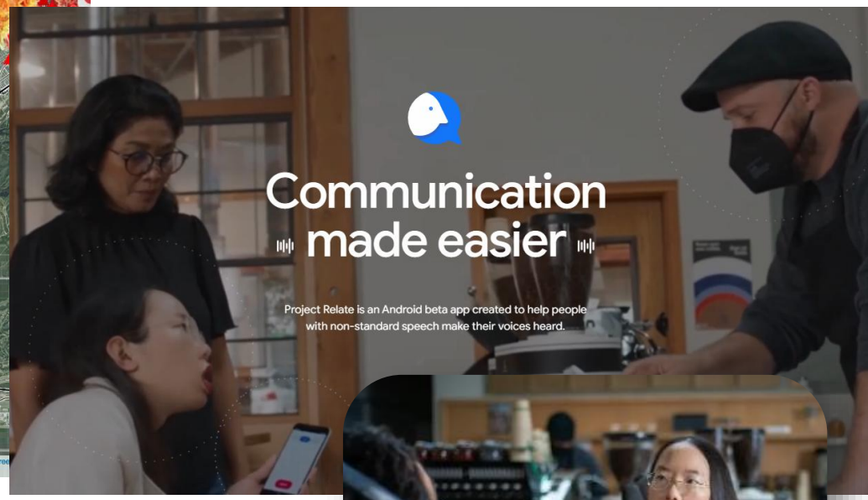
AI – it really is EVERYWHERE!

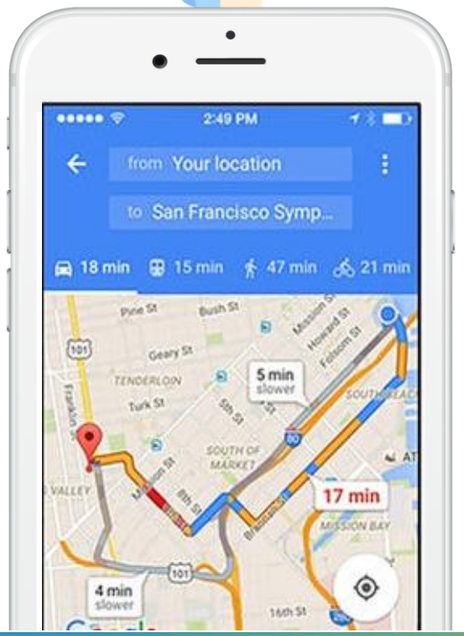
Enhancing your daily life & revolutionizing the human experience

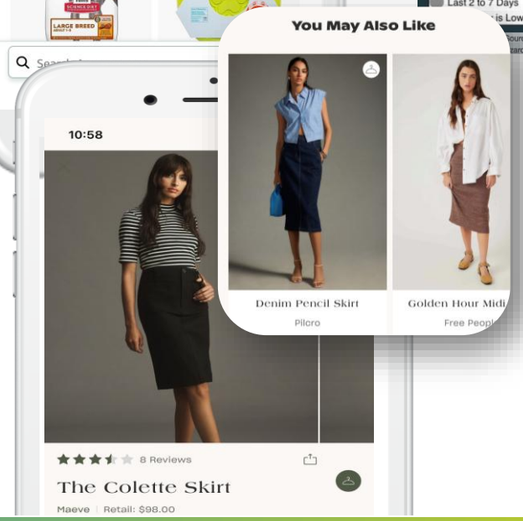












Evolution of AI Paradigms

Classical AI

expert systems (1950 – 2010)

If-Then Rules

Modern AI

supervised learning (2010 – 2020)

Deep Learning

Generative AI

self-supervision (2020+)

Large Language Models

Significant increase in investment, research, accessibility & visibility over the recent years

[The Evolution of AI Paradigms: From Classical AI to Modern and Generative AI (AAAI YouTube)]

Challenges to Build & Deploy Enterprise AI

Iteration
Must perform multiple iterations to improve features and model

Evaluation
No single metric to evaluate model performance

Algorithms
Appropriateness and performance of algorithm depends on application

Data Sparsity
Limited data on users and/or items, including cold-start problem

Scalability
Large datasets with complex processing pipelines

Representation
Finding and generating useful features of items and users

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from

Robust, scalable, reproduc
from structured, unstruct
semi-structured datasets

Selective, TextWiser,

AI for Decision Making

scale, integrated, (meta)
s for resource
ement and optimization.

ge, Balans, PathFinder

AI for Automated

Extraction and translation
tasks and intents for huma

Text2Model, Learn2Zinc

bility, evaluation, fairness,
ems.

BoolXLLM

Received: 26 July 2025 | Revised: 27 August 2025 | Accepted: 2 September 2025

DOI: 10.1002/aaai.70032

SPECIAL TOPIC ARTICLE

Open-source AI at scale: Establishing an enterprise AI strategy through modular frameworks

Serdar Kadıoğlu[1,2]

[1]AI Center of Excellence, Fidelity Investments, Boston, Massachusetts, USA

[2]Department of Computer Science, Brown University, Providence, Rhode Island, USA

Correspondence
Serdar Kadıoğlu
Email: serdark@cs.brown.edu

Abstract

We present a comprehensive enterprise AI strategy developed within the AI Center of Excellence at Fidelity Investments, emphasizing the strategic integration of open-source AI frameworks into scalable, modular, and reproducible enterprise-grade solutions. Our approach is structured around five key pillars: learning from offline data, learning from online feedback, intelligent decision-making, automated assistants, and responsible AI practices. Through a suite of 12 open-source libraries, we demonstrate how modular and interoperable tools can collectively enhance scalability, fairness, and explainability in real-world AI deployments. We further illustrate the impact of this strategy through three enterprise case studies. Finally, we distill a set of best deployment practices to guide organizations in implementing modular, open-source AI strategies at scale.

Open-Source AI at Scale: Establishing an Enterprise AI Strategy [AI Magazine'25]

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from Offline Data

Robust, scalable, reproducible features from structured, unstructured, and semi-structured datasets.

Selective, TextWiser, Seq2Pat

AI for Learning from Online Feedback

Adaptive, real-time, A/B testing systems that continuously learn from user interaction.

Mab2Rec, MABWiser, Read-Write-Learn

AI for Decision Making

Large-scale, integrated, (meta) solvers for resource management and optimization.

Forge, Balans, PathFinder

AI for Automated Assistants

Extraction and translation of natural language into downstream tasks and intents for human-computer interaction.

Text2Model, Learn2Zinc, Gala, Ner4Opt, Text2Zinc, iCBS

Responsible AI

Horizontal capabilities for explainability, evaluation, fairness, and bias mitigation across all systems.

Jurity, BoolXAI, BoolXLLM

Open-Source AI at Scale: Establishing an Enterprise AI Strategy [AI Magazine'25]

The Need for Explainable AI

Increasing Complexity

Machine Learning models are becoming **increasingly complex**, making it difficult to understand and interpret their predictions.

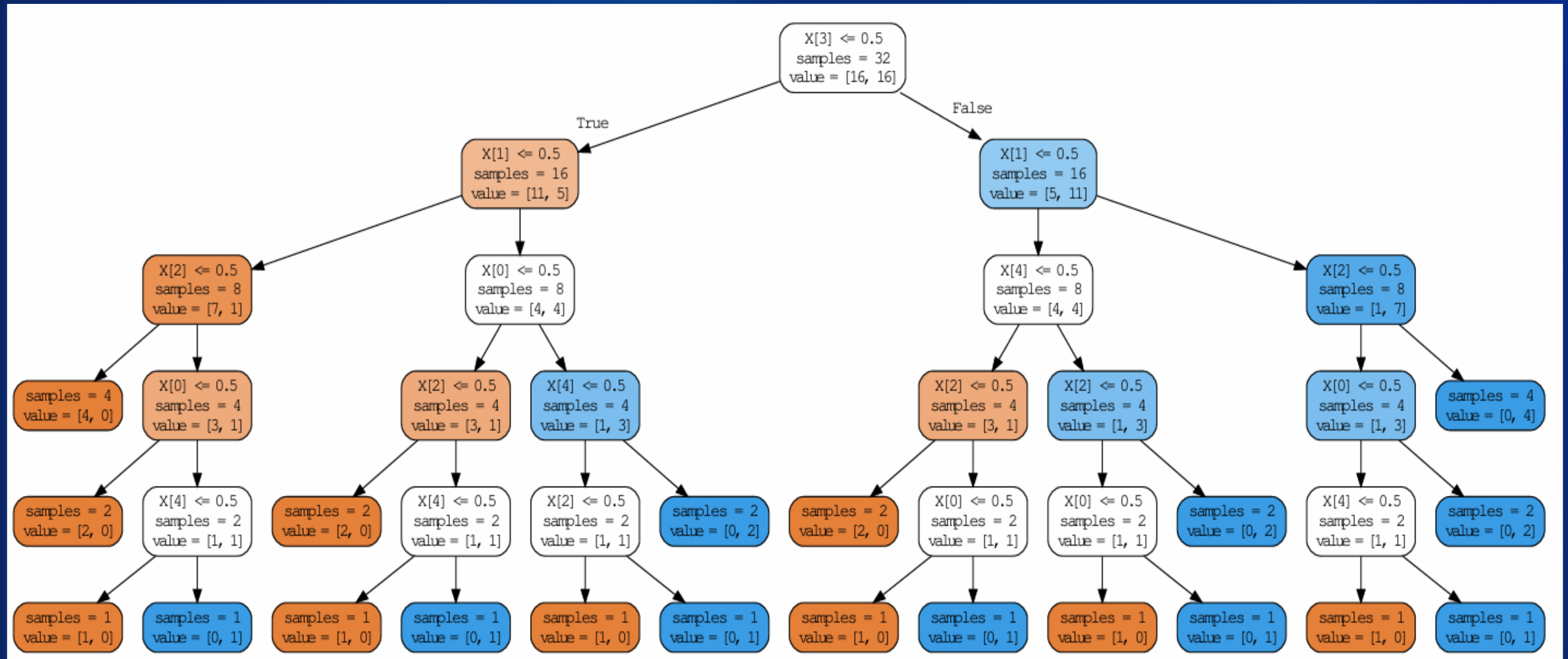
Regulatory Requirements

Explainability is mandatory in several domains such as finance and healthcare due to industry regulations.

Responsible AI

Explainable models help **discover superfluous patterns** and avoid unwanted bias in AI systems.

Limitations of Existing Models



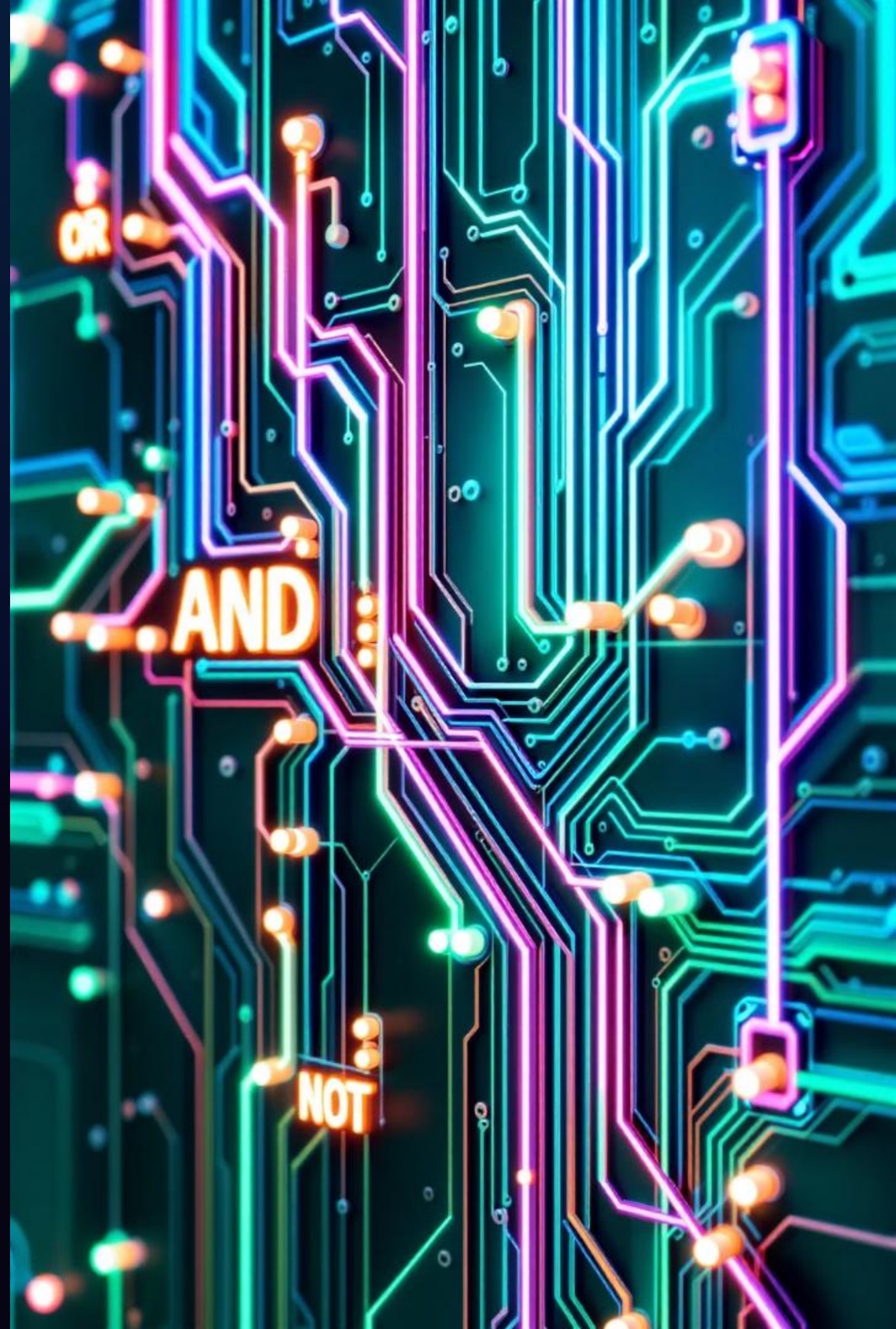
Expressiveness of Boolean Operators

Decision Trees

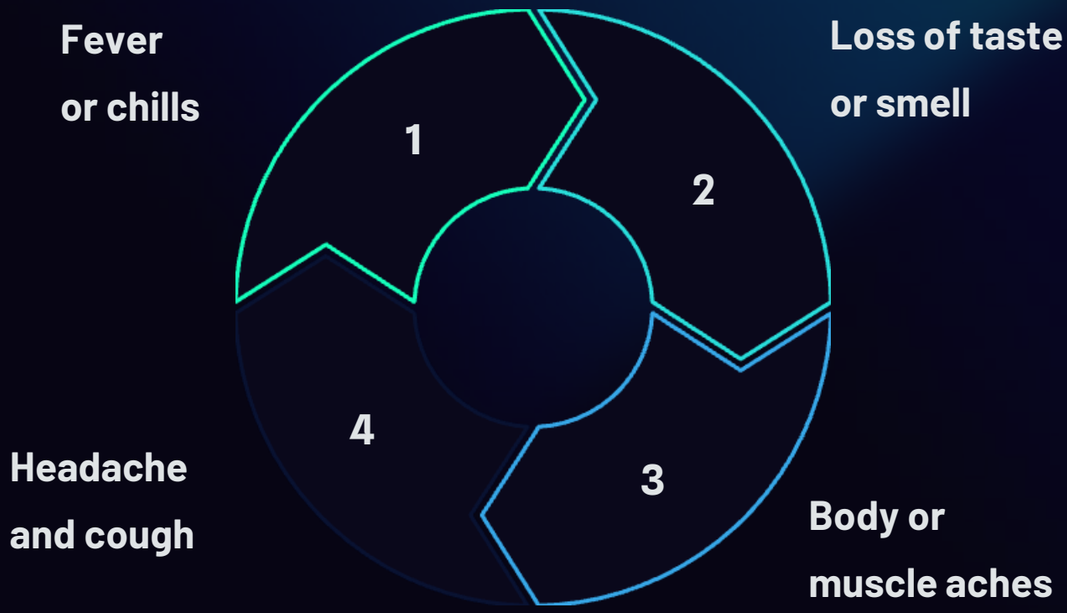
The same requires a decision tree with **19 split nodes** which is less interpretable

BoolXAI

BoolXAI can represent complex conditions concisely **AtLeast3(f1,..,f5)**

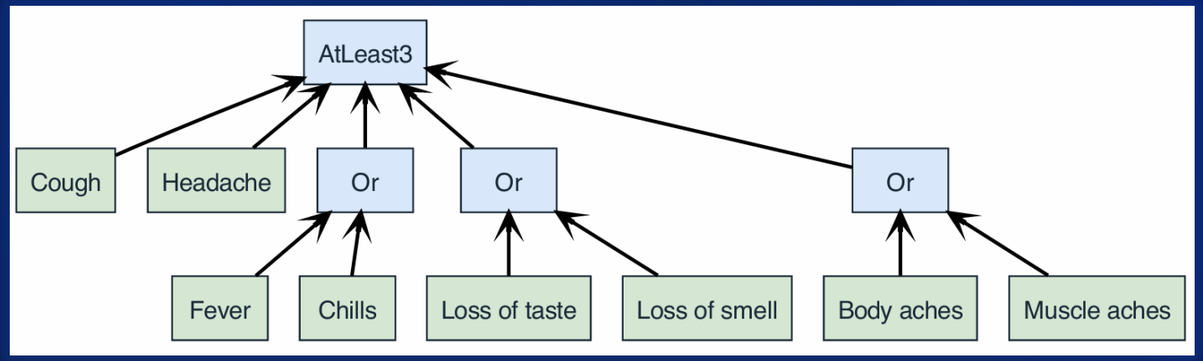


Motivation – Checklists



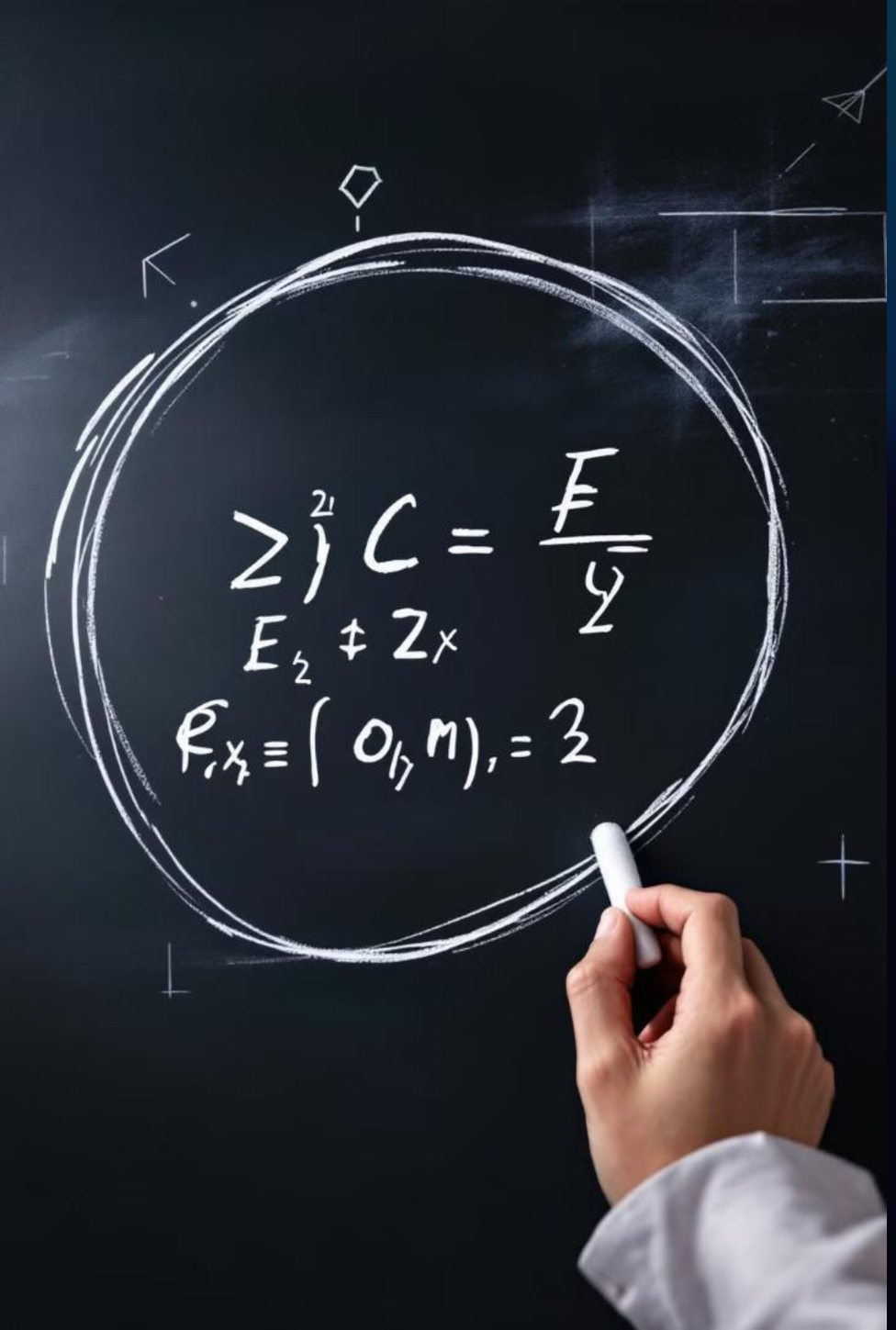
**AtLeast3 (Cough, Headache, Or(Fever, Chills),
Or(Loss of taste, Loss of smell),
Or(Body aches, Muscle aches))**

Complex decision-making processes can be expressed in a clear, interpretable manner using Boolean logic.



Parameterized operators are motivated by checklists, which are intuitive and widely used in various fields.

Example: a simplified disease checklist
Disease if you have at least three of these symptoms



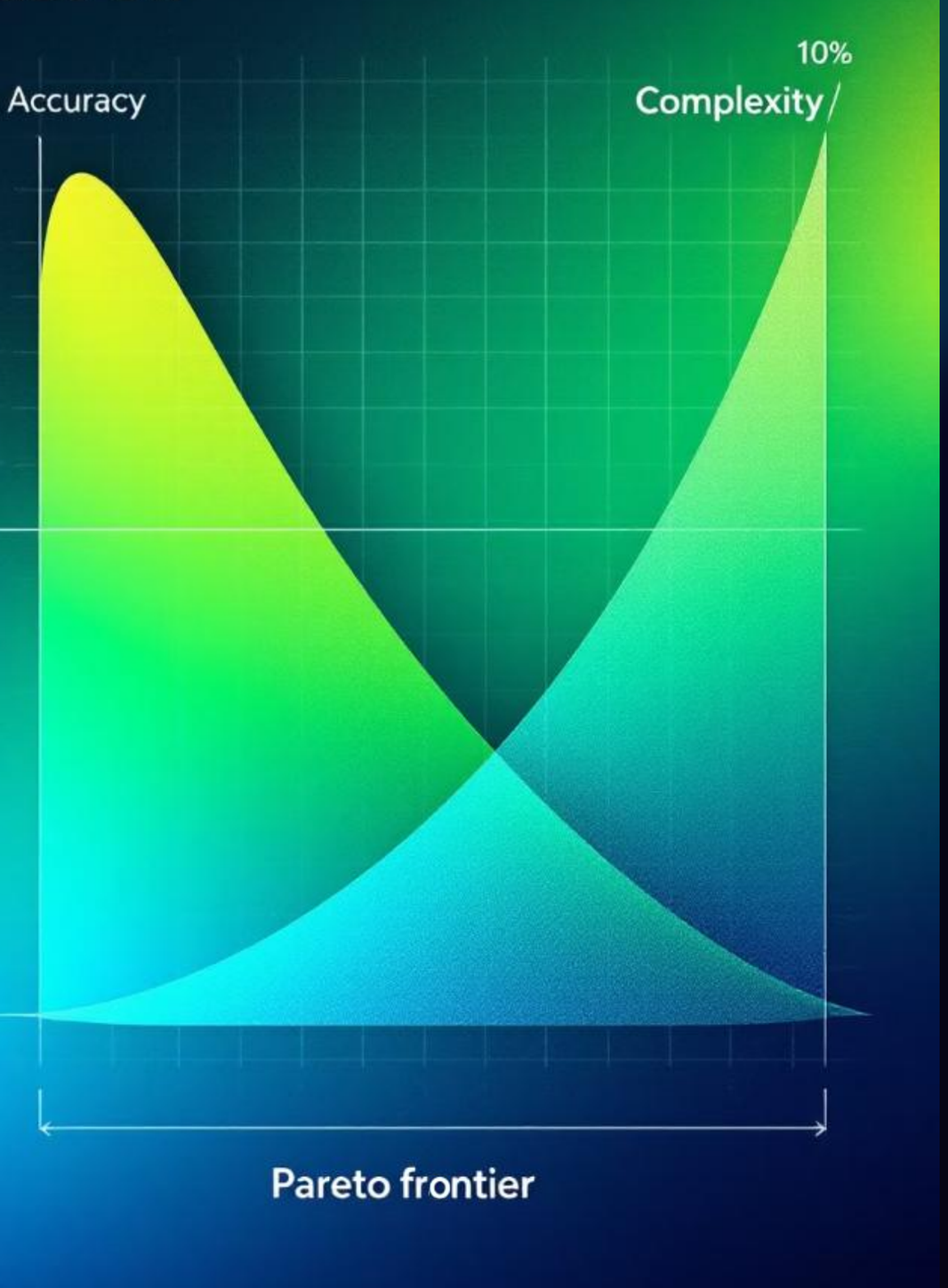
Optimization Problem Definition

Definition 1 (Rule Optimization Problem (ROP)) *Given a binary feature matrix* $\boldsymbol{X}$ *and a binary label vector* $\boldsymbol{y}$*, the goal of the Rule Optimization Problem (ROP) is to find the optimum rule* R^* *that balances the score* S *of the rule* R *on classifying the data, and the complexity of the rule* C*, which is given by the total number of features and operators in* R*, bounded by a parameter* C'*.*

Mathematically, our optimization problem can be stated at a high level as:

$$R^* = \arg\max_R [S(R(\boldsymbol{X}), \boldsymbol{y}) - \lambda C(R)] \quad \text{s.t.} \quad C(R) \leq C', \tag{1}$$

Rule optimization problem is the foundation for BoolXAI's approach to finding optimal Boolean formulas for classification tasks.



Score vs. Complexity

1. **Score:** performance metric (e.g., balanced accuracy)
2. **Complexity:** total number of operators and literals
3. **Tradeoff** between score and complexity

In our approach, we consider two key metrics: **score**, a performance metric such as accuracy, and **complexity**, total number of operators and literals in the Boolean formula. There is an **inherent tradeoff** between these two metrics.

Achieving a higher score will require sacrificing some interpretability by increasing the complexity of the formula. BoolXAI aims to find a **balance between these competing objectives**.

Collaborative Development

Academia

California Institute of Technology and **Brown University** contributed academic expertise.

Financial Technology

AI Center at Fidelity and **Fidelity Center of Applied Technology** provided industry perspective.

High-Tech Industry

Amazon Quantum Solutions and **AWS Center of Quantum Computing** offered cutting-edge technology.

BoolXAI Approach

1

Native Search Space

BoolXAI optimizes directly in the **space of all valid expressive** Boolean formulas, rather than reformulating the problem in a fixed format like MaxSAT, ILP, or QUBO.

2

Stochastic Local Search

The search space is explored via a series of **(non) local moves** that make changes to the current configuration.

3

Multi-Start Simulated Annealing

The initial rule is constructed by randomly choosing literals and operators within the complexity constraints with **multi-started simulated annealing** process.

BoolXAI User Base & Quick Start Example

pip install boolxai

100+

Data Scientists

BoolXAI is available to over 100 data scientists.

5000+

Downloads

Recent launch and has been downloaded over 5000 times in the broader community.

```python
import numpy as np
from sklearn.metrics import balanced_accuracy_score

from boolxai import BoolXAI, Operator

# Create random toy data for binary classification. X and y must be binary!
rng = np.random.default_rng(seed=42)
X = rng.choice([0, 1], size=(100, 10))
y = rng.choice([0, 1], size=100)

# Rule classifier with maximum depth, complexity, possible operators
rule_classifier = BoolXAI.RuleClassifier(max_depth=3,
                                         max_complexity=6,
                                         operators=[Operator.And, Operator.Or, Operator.Choose,
                                         random_state=42)

# Learn the best rule
rule_classifier.fit(X, y)

# Best rule and best score
best_rule = rule_classifier.best_rule_
best_score = rule_classifier.best_score_
print(f"{best_rule=} {best_score=:.2f}")
```

Illustrative Results

Dataset	Expressive BoolXAI Formula	Balanced Accuracy
Airline Customer Satisfaction	And(Inflight entertainment ≠ 5, Inflight entertainment ≠ 4, Seat comfort ≠ 0)	86%
Breast Cancer	AtMost1(worst concave ≤ 0.15, worst radius ≤ 16.43, mean texture ≤ 15.30)	95%
Direct Marketing	Or(dur > 393, employed < 5076, mon=mar)	86%
Online Shopper Intent	AtMost1(PageValues ≤ 5.55, PageValues ≤ 0, BounceRates > 0.025)	87%
Customer Churn	AtMost1(tenure > 5, Contract ≠ Month-to-month, InternetService ≠ Fiber optic)	82%

BoolXAI rules obtained by the native local solver with **max_complexity = 4** across well-known UCI ML datasets.
On average, BoolXAI achieves **85% balanced accuracy** with a **single Boolean formula** of complexity four.

EaSe: Explainability-as-a-Service

Interactive Web Service

EaSe is a web service powered by BoolXAI that does not require programming skills.

Data Upload

Users can **upload their input data** or provide its path on Amazon S3 for analysis.

Visualization & Explanations

EaSe provides BoolXAI rules and **visualizations** for easy interpretation. BoolXLLM provides explanations.

Incremental Optimization

Users can **seed assumptions or extract base rules** from previous runs for incremental re-optimization.

User-Driven Enhancements

1 2 3

Predefined User Rules

BoolXAI now allows optimizing only part of a rule, keeping a **user-defined base** rule fixed.

Incremental Rule Generation

Users build formulas **incrementally**, starting with the single best feature and optimizing the remainder.

Controlled Complexity

Users introduce features incrementally and decide when to **stop based on complexity.**

BoolXAI Summary

1 Expressive Boolean Formulas

Logical rules with tunable complexity for classification, going beyond classical conjunction and disjunction.

2 Effective Training & Competitive Results

Native local optimization to search the space of feasible formulas efficiently. Competitive with black-box models.

3 Open-Source Library

Provides a high-level user interface for researchers and practitioners, available at https://github.com/fidelity/boolxai

pip install boolxai

skadio.github.io

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from Offline Data

Robust, scalable, reproducible features from structured, unstructured, and semi-structured datasets.

Selective, TextWiser, Seq2Pat

AI for Learning from Online Feedback

Adaptive, real-time, A/B testing systems that continuously learn from user interaction.

Mab2Rec, MABWiser, Read-Write-Learn

AI for Decision Making

Large-scale, integrated, (meta) solvers for resource management and optimization.

Forge, Balans, PathFinder

AI for Automated Assistants

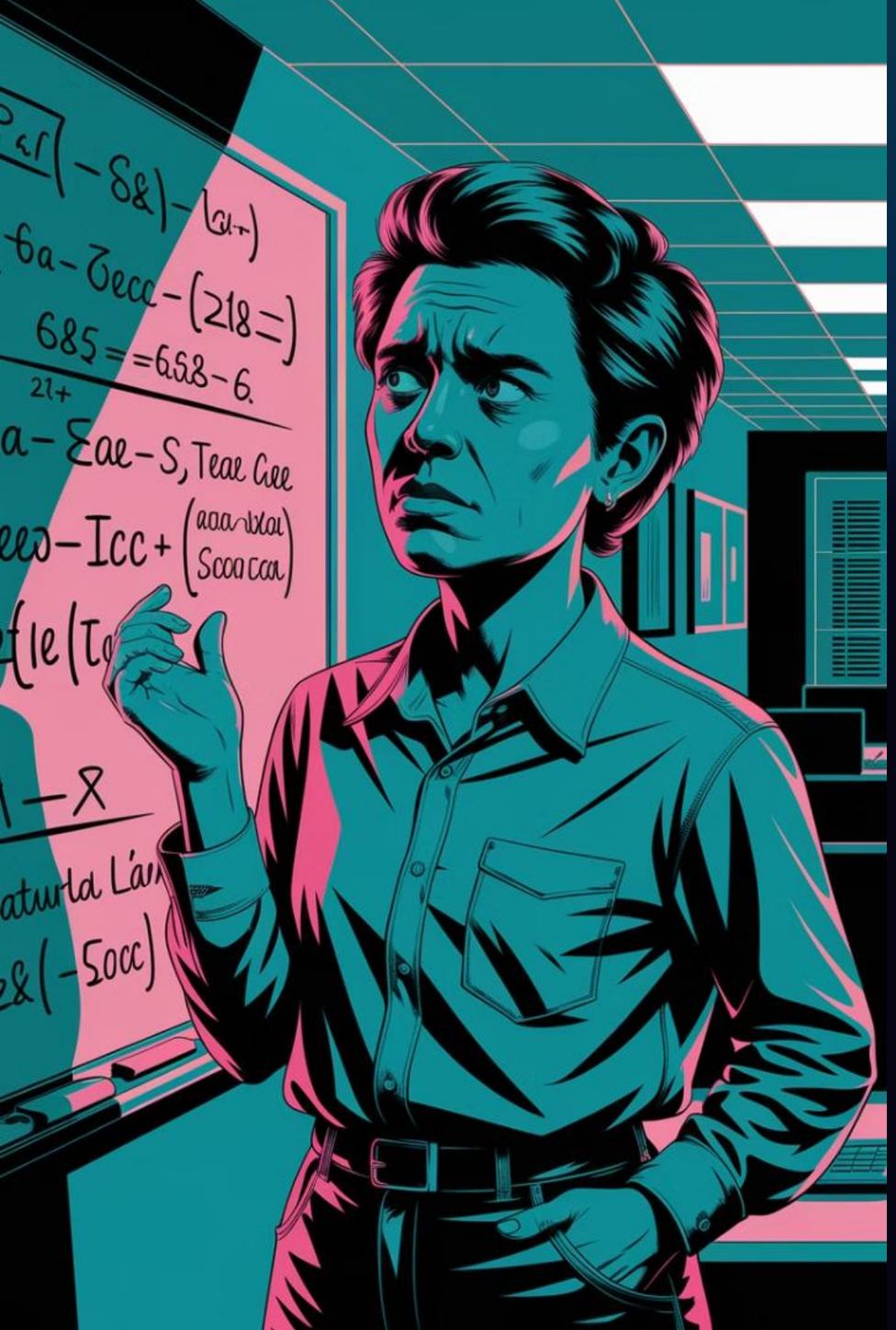
Extraction and translation of natural language into downstream tasks and intents for human-computer interaction.

Text2Model, Learn2Zinc, Gala, Ner4Opt, Text2Zinc, iCBS

Responsible AI

Horizontal capabilities for explainability, evaluation, fairness, and bias mitigation across all systems.

Jurity, BoolXAI, BoolXLLM



The De-Facto Model-and-Run Strategy

1

Problem Description

Users **describe optimization problems** in natural language, which contains ambiguous references to variables, constraints, and objectives that must be precisely identified.

2

Model Formulation

Experts must **manually transform problem descriptions** into formal mathematical models, a process that requires specialized knowledge and is prone to errors.

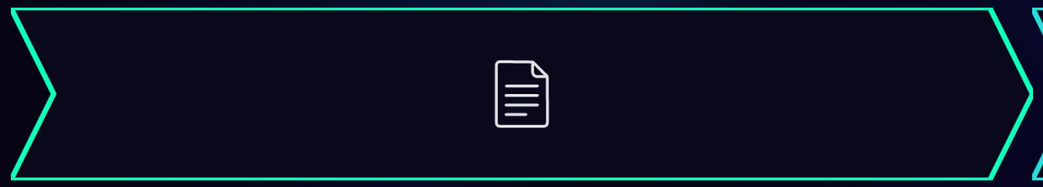
3

Solution Finding

Once properly modeled, optimization solvers can find optimal solutions, but the **modeling barrier** remains a significant obstacle to wider adoption of optimization technology.

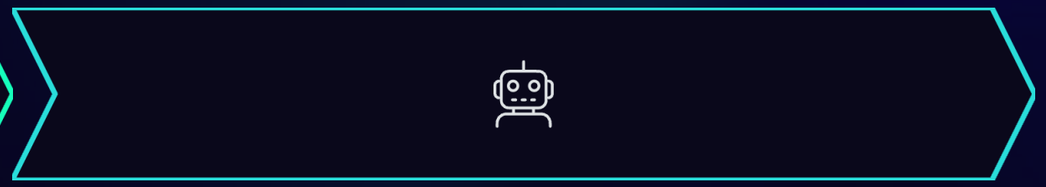
Our Vision: Modeling Co-Pilots

A paradigm shift integrating **automated modeling assistants** capable of translating natural language into formal optimization formulations.



Natural Language

Problem descriptions in free-form text



LLM Co-Pilot

Automated translation and formulation

Formal Models

Executable constraint model code

Solution | Interactivity | Feedback Loop

Verified results

Our Contributions

Text2Zinc Dataset

A **unified cross-domain dataset** curated to work with **LLM co-pilots** and **interactive editor** *Singirikonda et. al., AAAI'25*

Text2Model Copilots

A suite of **LLM Modeling copilots** with multiple strategies of varying complexity and **leaderboard** *Kadioglu et. al*

Gala: Global LLM Agents

A **global agentic approach** with specialized agents decompose modeling by global constraint. *Cai et. al. NeurIPS'25*

Learn2Zinc

Targeted **fine-tuning** to teach small language models to generate syntactically correct MiniZinc *Kadioglu et. al*

Ner4Opt

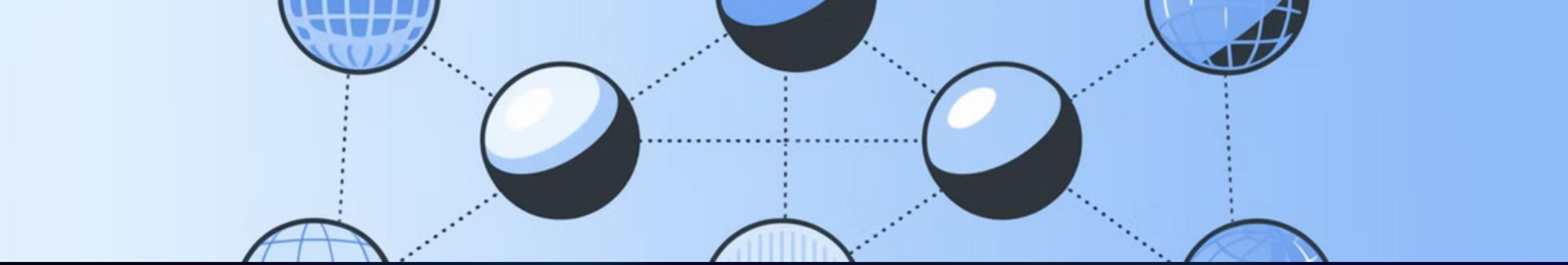
Extracting optimization components from free-form natural language text. *Kadioglu et. al, Constraints'24*

Holy Grail 2.0

Blueprint for **optimization modelling co-pilots** with feedback loop and user interactivity *Tsouros et. al., 2023*

https://skadio.github.io/text2model

pip install text2model



Text2Zinc: Addressing Dataset Gaps

1 2 3 4

Cross-Domain

- Focus on combining both **optimization & satisfaction** problems.
- **Unified** dataset to incorporate LP, MIP, and CP problems with integer, Boolean, and continuous

Unified Format

- Unifies existing datasets.
- **Clear separation** of problem description & instance data.

Solver Agnostic

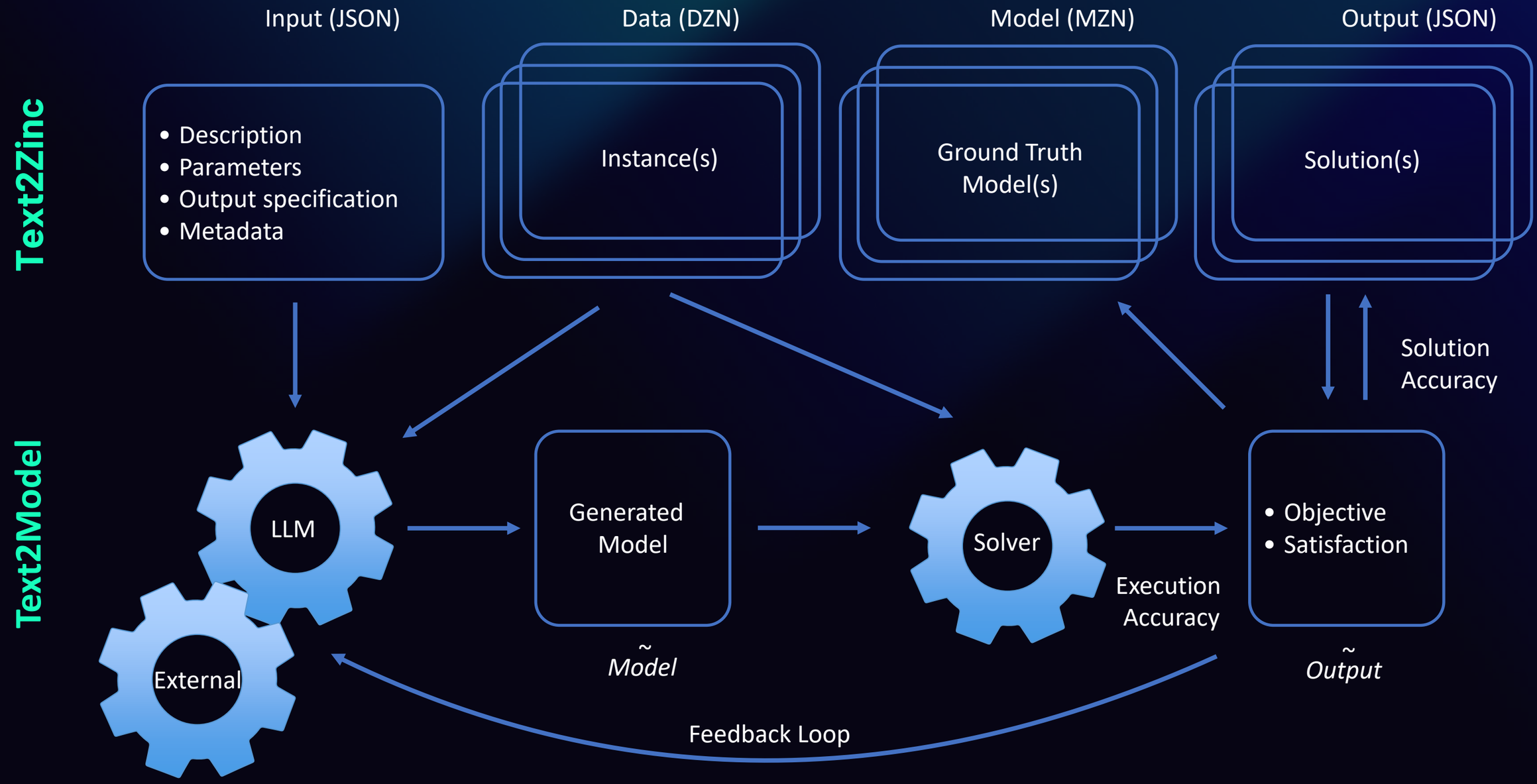
- Enables **solver agnostic** approaches.
- MIP, CP, SAT, LCG through MiniZinc.

Data Augmentation

- Clear and concise descriptions.
- Input and output specification.
- **Metadata** generation.
- Manual verification.

S. Kadıoğlu et. al. Text2Zinc: A Cross-Domain Dataset for Modeling Optimization and Satisfaction Problems in MiniZinc

From Text2Zinc Dataset to Text2Model Copilots



Text2Model Co-Pilots

Out-of-the-box LLM

Vanilla prompting, zero-shot, few-shot performance
Single vs. Multi-Call

Structured Prediction

Grammar-based model generation to enforce LLM output

Chain-of-Thought & Agentic

Improved reasoning through step-by-step problem-solving and agentic decomposition

Knowledge Graph

Leveraging structured knowledge as intermediary representation

Grammar Validation

When **syntax errors** are detected, the grammar validation prompt provides:

→ Problem Context

Problem description and data nomenclature for semantic grounding.

→ Current Code + Error

The syntactically incorrect MiniZinc code and the specific error message from execution.

→ MiniZinc Grammar Spec

The formal grammar rules (**from MiniZinc's Bison parser**) used to analyze and correct syntax.

Output: complete corrected MiniZinc code only, with all syntax errors resolved per the grammar specification.

MiniZinc Grammar Example

Grammar-constrained generation relies on formal grammar rules to validate generated code.

Consider this **simple rule for Boolean literals:**

```
<bool-literal> ::= "false" | "true"
```

What This Rule Enforces

- Boolean values have exactly two valid forms
- No other spellings (False, TRUE, 0/1) are valid
- Prevents invalid expressions like `maybe` or `yes`

Why It Matters

Prevents model from generating invalid Boolean expr
Applied across the full grammar, this post-processing step catches a wide range of structural violations **without requiring access to token probabilities**.

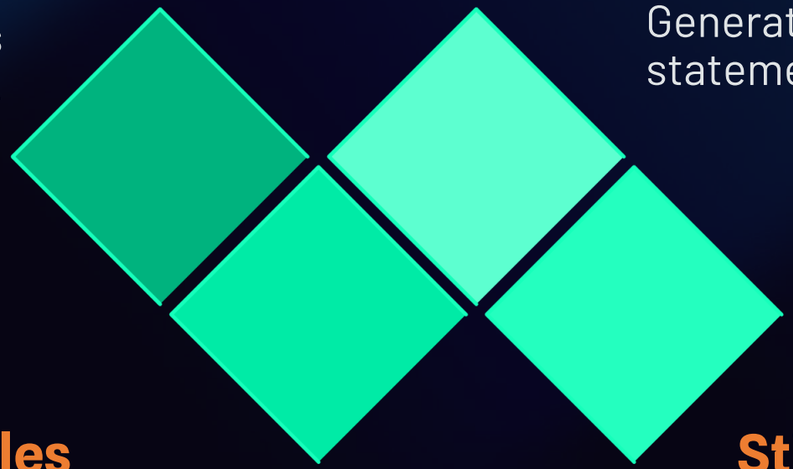
Agentic Decomposition

Constraints

Generate constraint blocks ensuring global correctness.

Objective

Generate the solve statement for optimization.



Parameters & Variables

Declare all inputs and decision variables.

Stitch

Combine sections into a complete MiniZinc model.

Each sub-task has dedicated principles: parameters/variables focus on type consistency and bounded declarations constraints emphasize global constraints and iteration correctness, objective ensures a single solve keyword stitch validates integration, resolves circular dependencies, and confirms type consistency across all sections.

Benchmark Results: GPT-5.2

Strategy	NLP4LP E/S	CSPLIB E/S	HAKANK E/S	INDUSTRYOR E/S	MAMO-EASY E/S	MAMO-COMP E/S
Zero-Shot	63.08 / 32.31	63.64 / 36.36	50.00 / 31.82	66.00 / 46.00	95.71 / 79.75	48.82 / 27.96
Chain-of-Thought	70.77 / 35.38	54.55 / 27.27	59.09 / 36.36	80.00 / 49.00	98.62 / 83.90	57.35 / 39.34
CoT + Grammar	**76.92 / 37.26**	**90.91 / 45.23**	**72.73 / 37.9**	**92.00 / 56.00**	**99.85 / 83.74**	**95.26 / 54.03**
Agentic + Code	76.92 / 29.23	81.82 / 36.36	68.18 / 36.36	86.00 / **57.00**	99.23 / **84.97**	92.89 / 54.03

CoT + Grammar achieves the best results across most datasets, potential of structured output generation.

What’s Next?

1. **Call-to-Action: Dataset & Copilot Expansion**

   Encourage community contributions to create more comprehensive resources

2. **Context Engineering**

   Explore alternative intermediate representations and semantic graphs

3. **Agentic SLM Frameworks**

   Investigate agentic approaches in capturing nuances of modeling problems

4. **Specialized SLMs**

   Develop specialized SLMs for optimization and satisfaction tasks

MiniZinc is Out-of-Distribution for SLMs

Complete Failure Without Fine-Tuning

The combination of **rare syntax** and **unfamiliar paradigm** explains the near-zero performance of untuned models.

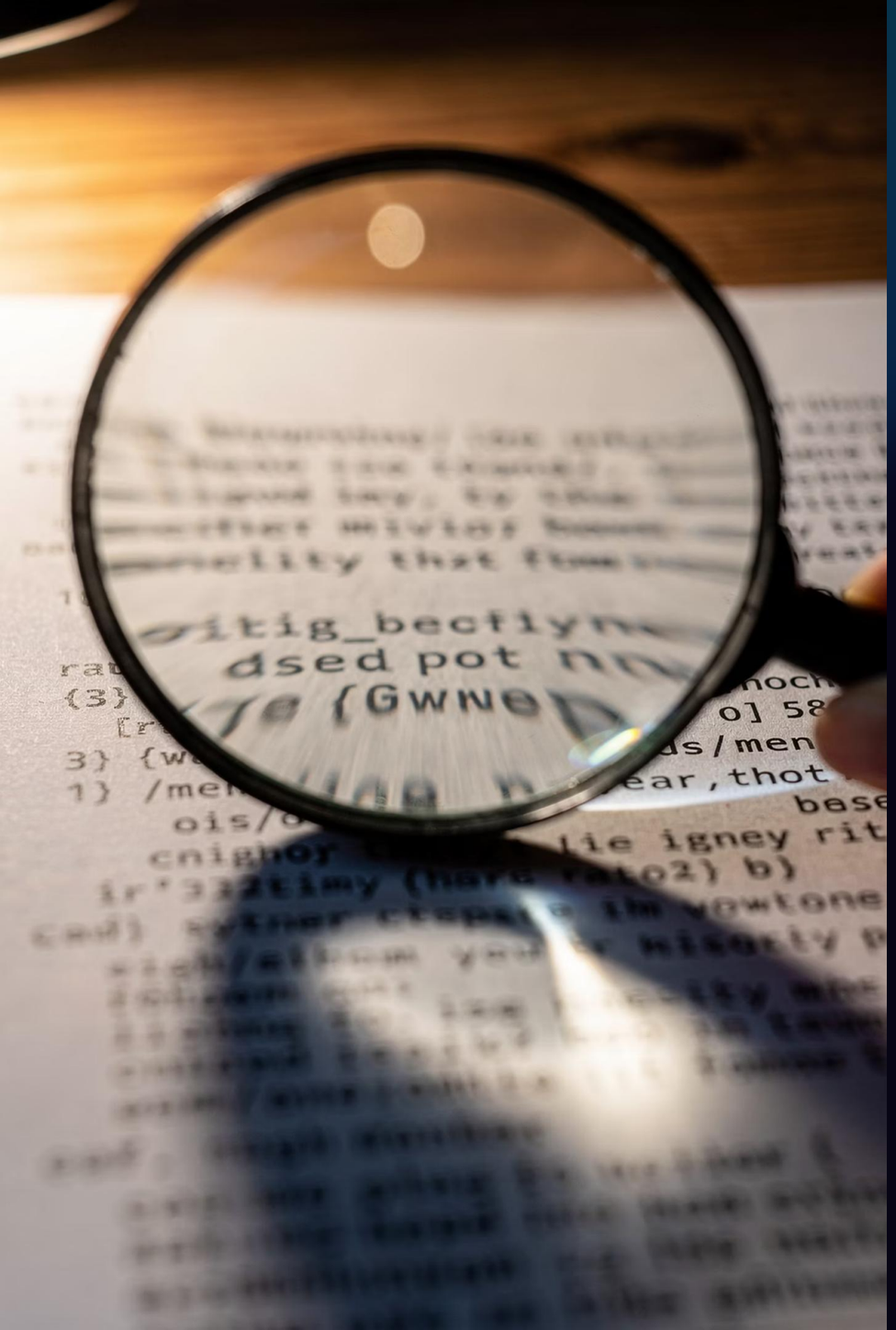
0%

Qwen3, LLaMa3, Gemma2, GPT-OSS

Execution accuracy out-of-the-box

First Systematic Study

Fine-tuning SLMs for MiniZinc code generation + Blueprint



The Core Insight: Verification Enables Fine-Tuning

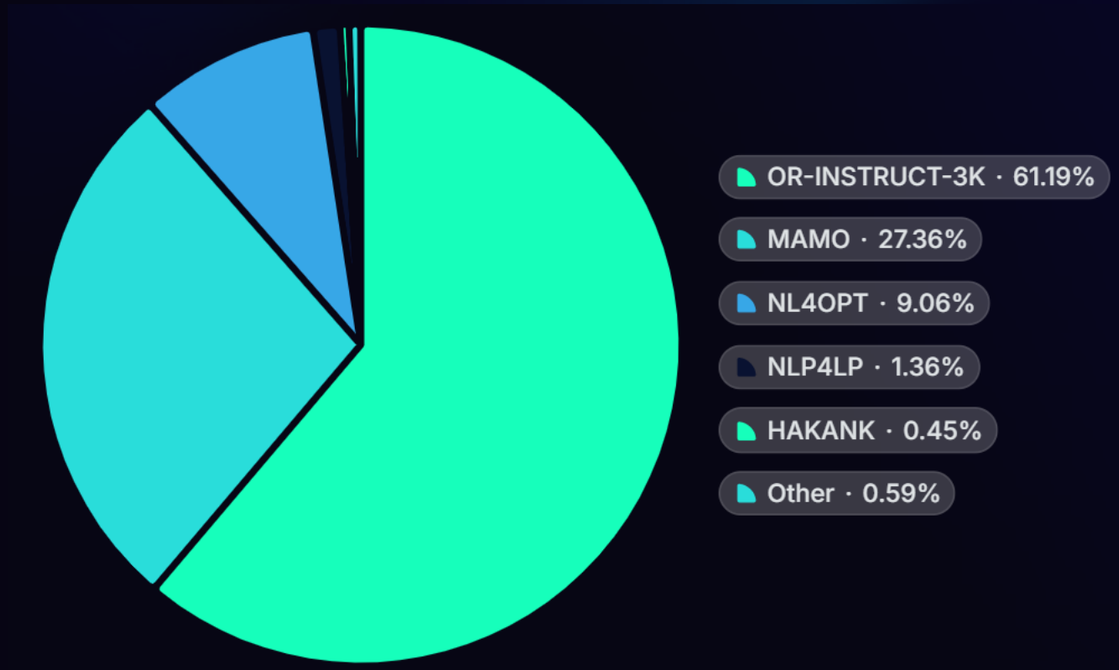
Text2Model copilots on hundreds of **Text2Zinc** instances generates a **large volume of MiniZinc models** even if not manually written.

The critical advantage is the **verification loop**:

- Optimization problems: verified against known objective values
- Satisfaction problems: feasibility asserted

This yields a large pool of **verified** **(text, model)** **pairs** that makes fine-tuning possible *without requiring manual annotation.*

Fine-Tuning Dataset: 2,208 Problems with 8,014 Pairs



Summary

2,208 unique problems across **7** sources.

Instances may have **multiple alternative MiniZinc models** from different Text2Model copilot strategies.

This yields **8,014** **instruction-tuning** **(text, model) pairs** for fine-tuning small language models.

OR-INSTRUCT dominates at **61.2%**, with MAMO contributing **27.4%**.

Fine-Tuning Methodology

Our methodology combines **multiple SLM families** at varying parameter sizes with **two fine-tuning objectives**: generation fine-tuning and error-correction fine-tuning.

1) Generation Fine-Tuning

Teach SLMs to produce valid MiniZinc from natural language descriptions.

2) Error-Correction Fine-Tuning

Teach SLMs to identify and **fix syntax mistakes** in broken MiniZinc code.

Small Language Models

Qwen3-0.6B

Smallest model — tests **lower bound** of SLM capability.

LLaMA-3.2-1B & 3B

Mid-range models from Meta's LLaMA family.

Gemma-2-9B

Google's 9B parameter model.

GPT-OSS-20B

Largest model — tests **upper bound** of SLM capability.

Three Fine-Tuning Datasets

Learn2Zinc-Base

8,014 pairs of
(text, model)

Direct **code generation** training.

Learn2Zinc-CoT

8,014 pairs of
(text, reasoning, model)

GPT4o generates **reasoning chains** identifying variables, parameters, constraints, and objectives.

Learn2Zinc-Base+CoT

16,028 pairs
combining both Base and CoT

Mixed
training strategy.

Benchmark: Text2Zinc-IndustryOR

100 Text2Zinc-IndustryOR instances as test set

the most challenging benchmark as reported across multiple studies.

86%

Best Exec Accuracy

CoT + Grammar GPT-5.2

(Kadioğlu et al. 2026)

57%

Best Sol Accuracy

CoT + Grammar GPT-5.2

(Kadioğlu et al. 2026)

NOT included in fine-tuning

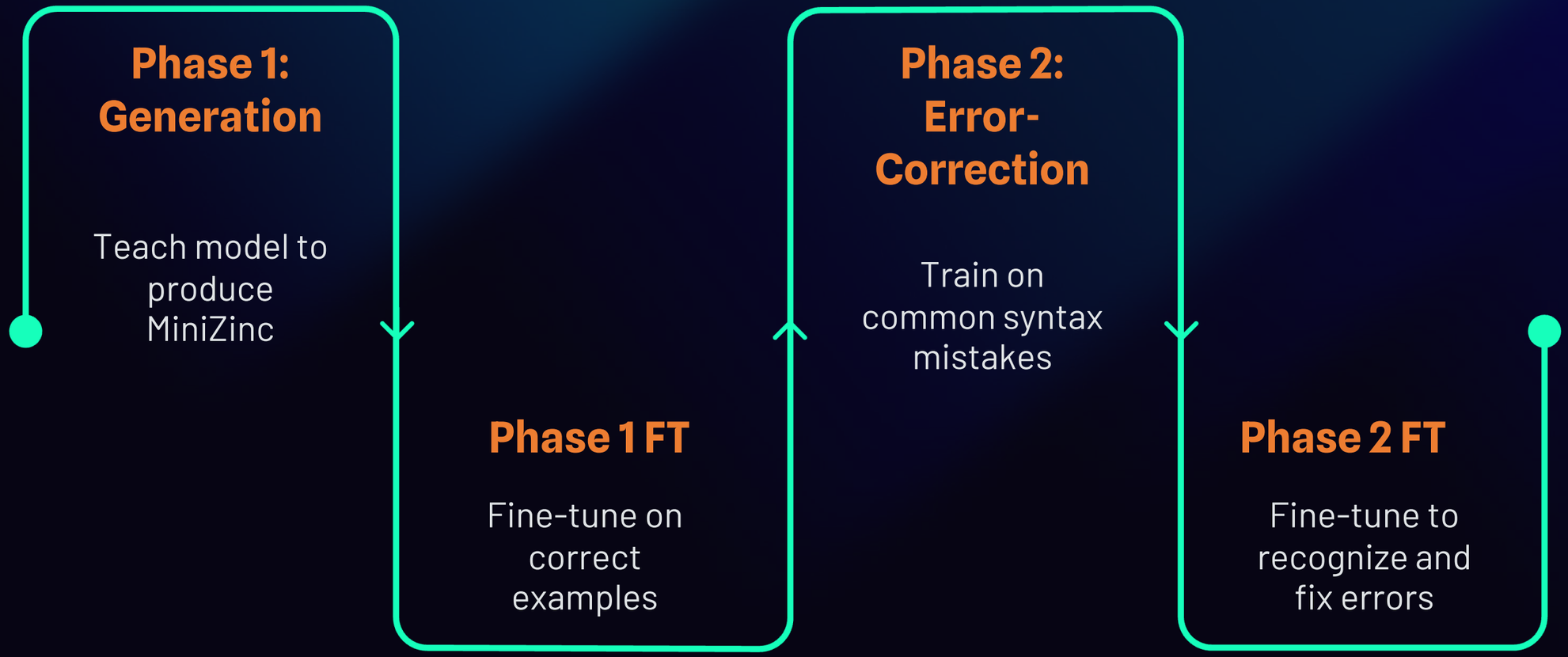
Model	Strategy	Exec Acc (%)	Sol Acc (%)
Qwen3-0.6B	Original-8bit	**0.0**	**0.0**
	Learn2Zinc-Base	**51.0**	9.0
	Learn2Zinc-CoT	44.0	10.0
	Learn2Zinc-Base+CoT	51.0	12.0
LLaMA-3.2-1B	Original-8bit	**0.0**	**0.0**
	Learn2Zinc-Base	**44.0**	4.0
	Learn2Zinc-CoT	22.0	1.0
	Learn2Zinc-Base+CoT	46.0	4.0
LLaMA-3.2-3B	Original-8bit	**0.0**	**0.0**
	Learn2Zinc-Base	**62.0**	10.0
	Learn2Zinc-CoT	47.0	9.0
	Learn2Zinc-Base+CoT	58.0	12.0
Gemma-2-9B	Original-8bit	**0.0**	**0.0**
	Learn2Zinc-Base	**74.0**	**22.0**
	Learn2Zinc-CoT	49.0	15.0
	Learn2Zinc-Base+CoT	70.0	21.0
GPT-OSS-20B	Original-4bit	**6.0**	**5.0**
	Learn2Zinc-Base	**66.0**	**27.0**
	Learn2Zinc-CoT	52.0	21.0
	Learn2Zinc-Base+CoT	63.0	27.0

Near-Zero Baseline

Remarkable Boost
COT underperforms

SLMs vs. Frontier (86%-57%)

Shifting to Error-Correction Fine-Tuning



Design an augmented training strategy that **explicitly teaches syntax correction**. The key insight: models need exposure to common errors *and* ***their fixes***, not just correct outputs. This requires a dedicated error-correction dataset built from **two complementary sources.**

Error Correction Dataset: Synthetic Corruptions

We derive **20 corruption rules** from MiniZinc's BNF grammar, organized into three difficulty levels, to generate realistic syntax errors and their fixes.

Easy (7 rules)

Delimiters and punctuation: missing semicolons, dropped commas, unbalanced brackets.

Medium (6 rules)

Keywords and types: invalid solver directives, dropped `var` or `of` keywords, split compound tokens like `endif`.

Hard (7 rules)

Structural elements: swapped declaration order, invalid operators, misplaced semicolons.

Sampling: 30% easy, 30% medium, 30% hard, 10% identity (no fix needed).

This process yields **4,452 verified instances** with **⟨text, corrupt_model, model⟩** triples.

Cross-Model Error Bootstrapping

Synthetic corruptions may not capture actual LLM failure modes. We address this by collecting real errors from model runs:

01

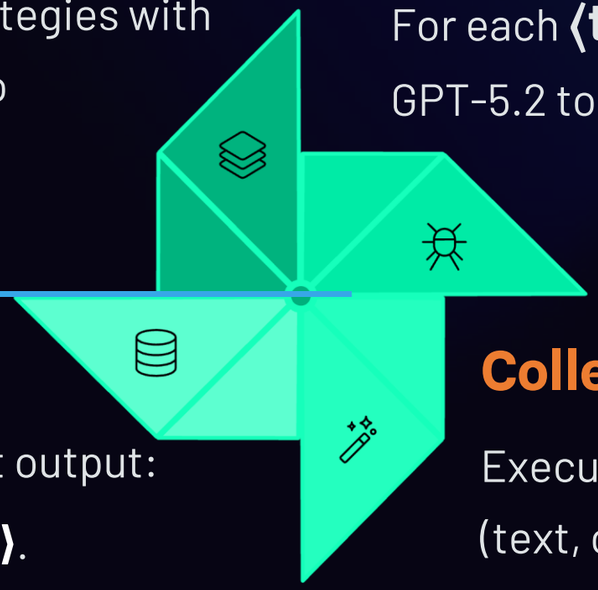
Run All 15 SLMs

Across all 5 SLMS and 3 fine-tuning strategies with **sampling temperatures** (0.2, 0.5, 0.8) to maximize error variety.

02

Collect Execution Failures

For each **⟨text, corrupt_model, error_message⟩**, GPT-5.2 to make **minimal targeted fixes** preserving original



03

Verify & Add to Dataset

If the fix compiles and produces correct output: add **⟨text, corrupt_model, model⟩**.

Yields **2,286 verified correction instances**.

04

Collect Successes Too

Execution and solution successes yield new positive (text, correct_model) pairs — extending the base dataset to **8,911 generation instances**.

Augmented Dataset with Dual Fine-Tuning Objective

4,452

Synthetic Corruptions

Grammar-based error-correction pairs

2,286

Cross-Model Corrections

Real LLM failure corrections via GPT-5.2

8,911

Generation Examples

Verified (text, model) pairs for code generation

15,649

Total Instances

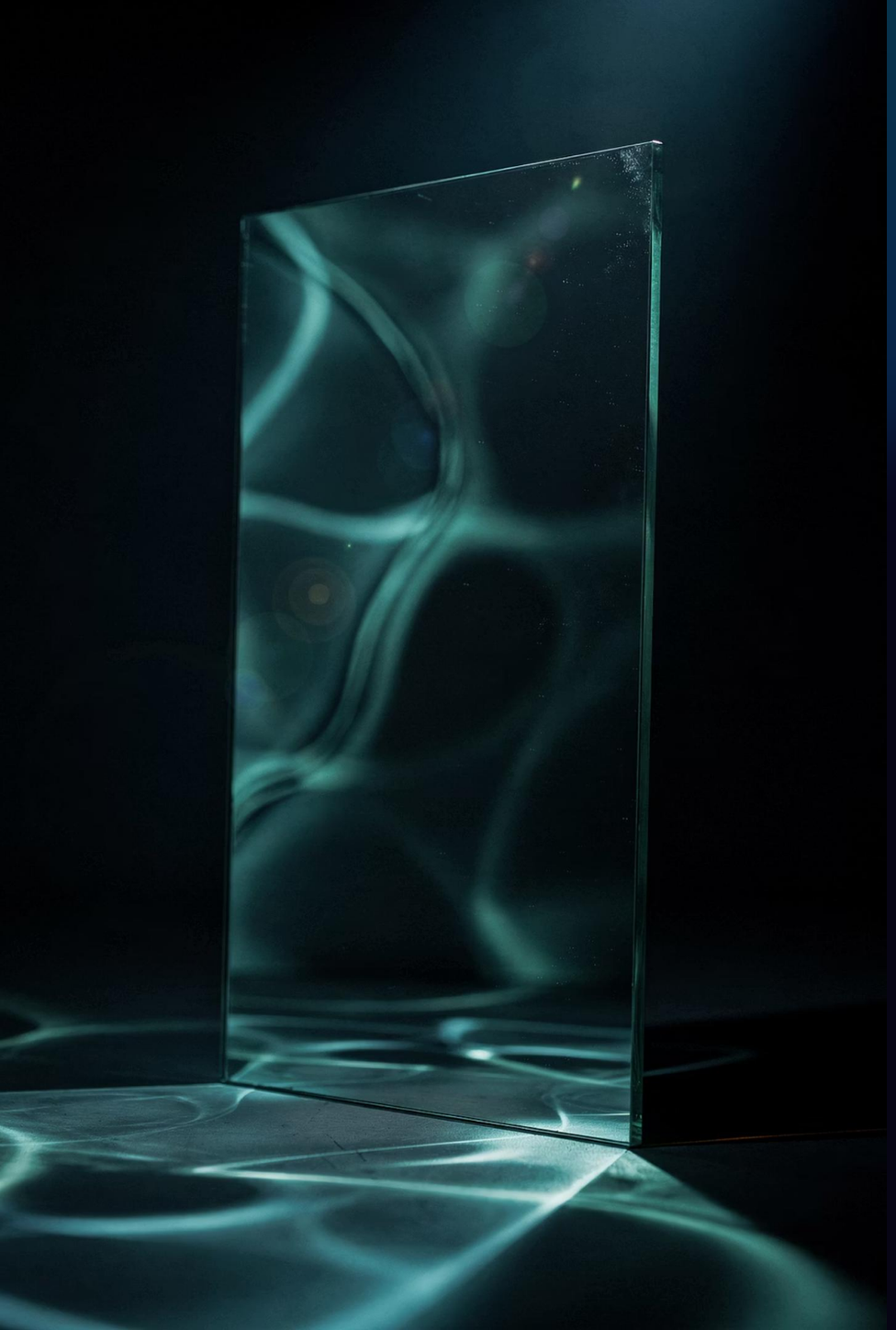
Combined augmented fine-tuning dataset

Learn2Zinc Augmented

Fine-tuning all five SLMs on the 15,649-instance augmented dataset with **dual objectives** (generation + error-correction) yields **consistent improvements** across all model sizes.

Model	Orig Exec (%)	Base Exec (%)	Aug Exec (%)	Aug Sol (%)
Qwen3-0.6B	0.0	51.0	**64.0**	13.0
LLaMA-1B	0.0	44.0	**57.0**	8.0
LLaMA-3B	0.0	62.0	**70.0**	17.0
Gemma-9B	0.0	74.0	72.0	22.0
GPT-OSS-20B	6.0	66.0	**76.0**	**32.0**

Validating that training for error correction directly addresses the syntax bottleneck.



Beyond Zero-Shot: Self-Reflection & Ensemble

Having fine-tuned for error correction, we now leverage this capability at **inference time**. Two complementary strategies

Self-Reflection Loop

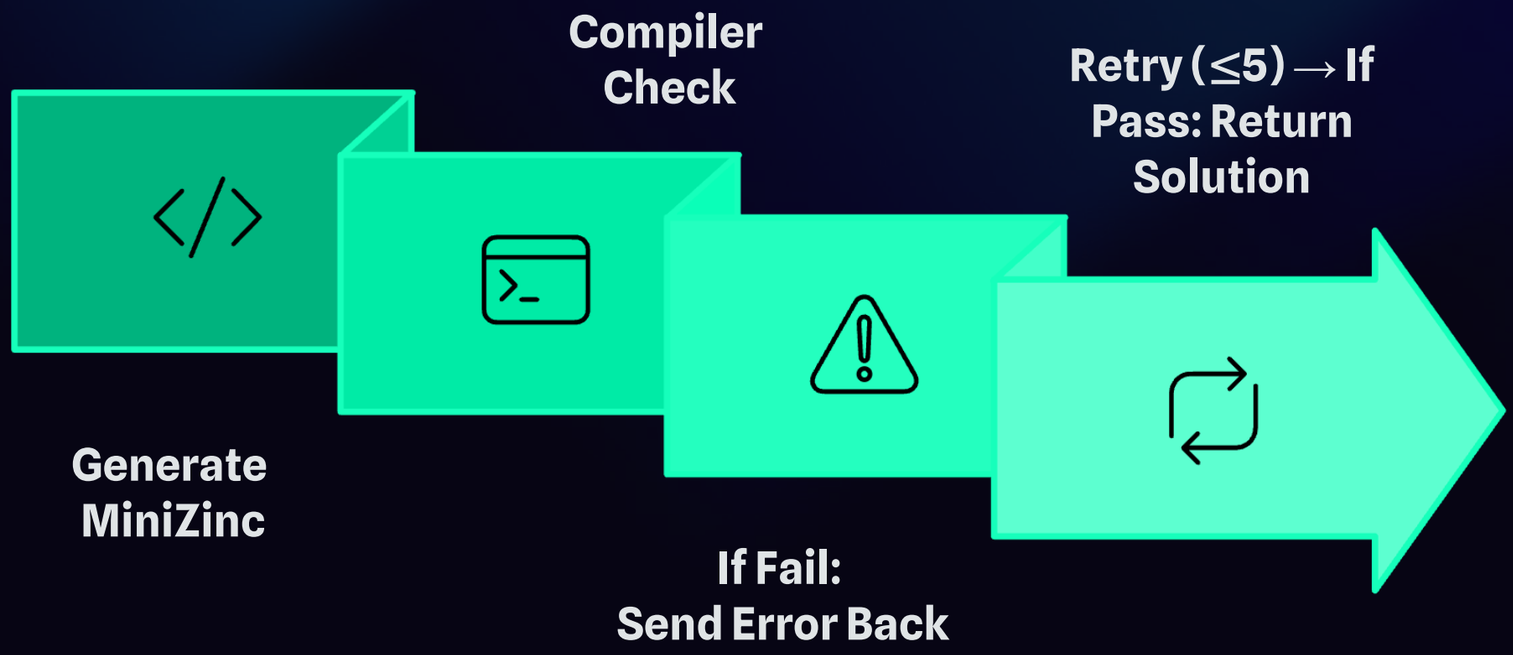
Model receives its broken code + compiler error message and retries — up to 5 attempts.

Top-Down Ensemble

Models tried in descending capability order: GPT-OSS-20B → Gemma-9B → LLaMA-3B → LLaMA-1B → Qwen-0.6B.

Self-Reflection up to 5 Attempts

On each failed attempt, the model receives its broken code and the compiler error message, then tries again



Reflections reach a running model **in less than two trials on average**, with larger SLMs requiring fewer attempts.

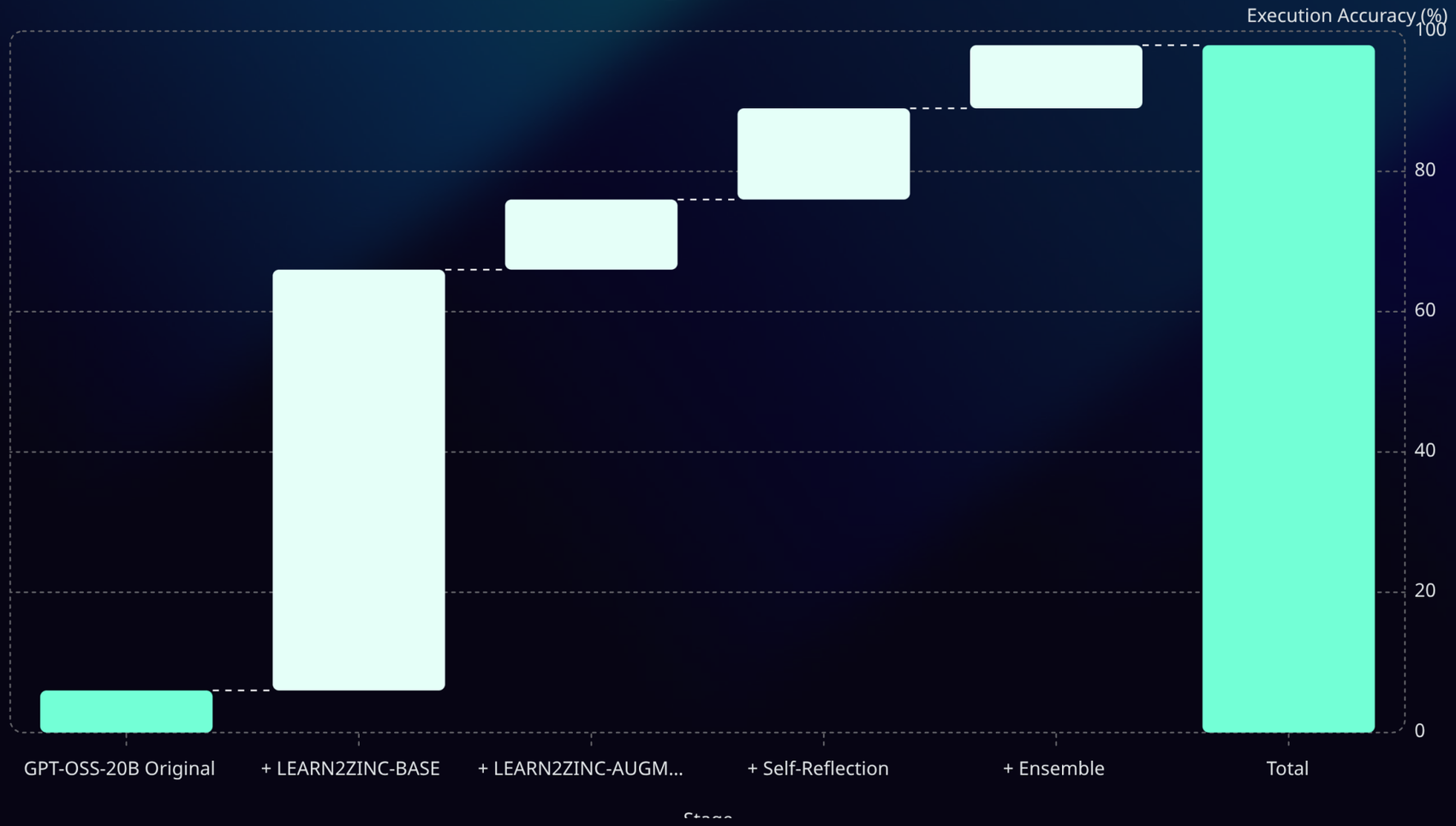
Key Milestone: Self-Reflection 89% – GPT-5.2 86%

Model	Single Exec (%)	Single Sol (%)	Reflection Exec (%)	Reflection Sol (%)	Avg Tries
Qwen3-0.6B-Aug	64.0	13.0	72.0	13.0	2.22
LLaMa-3.2-1B-Aug	57.0	8.0	71.0	8.0	2.35
LLaMa-3.2-3B-Aug	70.0	17.0	80.0	17.0	1.90
Gemma-2-9B-Aug	72.0	22.0	84.0	24.0	1.79
GPT-OSS-20B-Aug	76.0	32.0	**89.0**	**34.0**	**1.62**

98% Execution Accuracy with Ensemble

Model	Approach	Exec (%)	Sol (%)
GPT-OSS-20B	Original-4bit	6.0	5.0
GPT-OSS-20B	Learn2Zinc-Base	66.0	27.0
GPT-OSS-20B	Learn2Zinc-Augmented	76.0	32.0
GPT-OSS-20B	Aug + Self-Reflection	89.0	34.0
Our Fine-Tuned SLMs	**Learn2Zinc - Ensemble**	**98.0**	35.0
GPT-5.2	CoT + Grammar (Kadioglu et. al.)	86.0	**57.0**

Learn2Zinc: Overall Progression



Each stage of Learn2Zinc **builds on the previous, steadily improving** execution accuracy from **6%** to a remarkable **98%** — effectively **solving the MiniZinc syntax problem** for small language models.

Learn2Zinc: Key Take Aways

Novelty

Cross-model error bootstrapping constructs realistic syntax error-correction datasets from LLM failures, enabling models to learn both code generation and error correction with **dual fine-tuning**

Result

Learn2Zinc-Augmented + Reflection Ensemble achieves **98% execution accuracy** effectively resolving the syntax bottleneck. Solution accuracy saturates at 34–35%.

Insight

Unlike LLMs, **COT fine-tuning does not improve performance** when foundational syntax knowledge is absent in SLMs. Syntax can be learned; reasoning remains the primary challenge.

→ **Distillation Reasoning from Larger Models**

Distilling **constraint reasoning capabilities** from frontier models into smaller, more efficient SLMs.

https://skadio.github.io/text2model

pip install text2model

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from Offline Data

Robust, scalable, reproducible features from structured, unstructured, and semi-structured datasets.

Selective, TextWiser, Seq2Pat

AI for Learning from Online Feedback

Adaptive, real-time, A/B testing systems that continuously learn from user interaction.

Mab2Rec, MABWiser, Read-Write-Learn

AI for Decision Making

Large-scale, integrated, (meta) solvers for resource management and optimization.

Forge, Balans, PathFinder

AI for Automated Assistants

Extraction and translation of natural language into downstream tasks and intents for human-computer interaction.

Text2Model, Learn2Zinc, Gala, Ner4Opt, Text2Zinc, iCBS

Responsible AI

Horizontal capabilities for explainability, evaluation, fairness, and bias mitigation across all systems.

Jurity, BoolXAI, BoolXLLM

Open-Source AI at Scale: Establishing an Enterprise AI Strategy [AI Magazine'25]

skadio.github.io