SAIBench: Benchmarking AI for Science

Yatao Li, Jianfeng Zhan
Institute of Computing Technology
Chinese Academy of Sciences
Project homepage

- https://www.computercouncil.org/SAIBench/

Publication

- Yatao Li, Jianfeng Zhan, SAIBench: Benchmarking AI for Science, BenchCouncil Transactions on Benchmarks, Standards and Evaluations, Volume 2, Issue 2, 2022, 100063, https://doi.org/10.1016/j.tbench.2022.100063 (https://www.sciencedirect.com/science/article/pii/S2772485922000503)
Agenda

• The Landscape of AI for Science
• The Definition of Scientific AI Benchmarking
• Challenges and Methodology
• System Design
• Case Studies
The Landscape of Scientific AI

- This “AI wave” is propagating into scientific research communities, as researchers are gaining interest in leveraging state-of-the-art AI solutions to tackle difficult tasks.

---

**Mathematical Problem Solving**

\[ \frac{Du}{Dt} = -\nabla p + \nabla \cdot \tau + \rho g \]

**Pattern Matching**

**Prediction**

**Artifact Enhancement**

**Control**

**Hypothesis and Confirmation**

Species Classification, Event Identification, Climate Analysis, Anomaly Detection, ...

High-Energy Particle Simulation
- Molecular Dynamics
- Fluid Dynamics
- Protein Folding
...

Automatic Physics Laws Discovery
- Symbolic Regression
...

Tokamak Plasma Control, Robotics, Sensor Triggering, ...

---

Partial derivative equations
- General matrix multiplication
- Matrix decomposition
- Integration
- Monte Carlo methods
...

Genome Sequence Alignment
- Astronomy Image Enhancement
- Medical Image Enhancement
- MRI Reconstruction
...

---
Tractable Scientific Tasks

- A scientific research activity can be “creative” or “tractable”

Creative = Open-ended (yet hard to quantify)
- A new problem to solve
- A vision
- …

Tractable = Computationally verifiable
- “0.5% average error against golden standard”
- “9998 cases out of 10000 predicted correctly”
- “45.53dB Peak Signal-to-Noise Ratio”

Mathematical Problem Solving
\[
\frac{\partial U}{\partial t} = -\nabla p + \nabla \cdot \tau + \rho g
\]

Pattern Matching

Prediction

Artifact Enhancement

Hypothesis and Confirmation
Traditional View: Vertical Scientific Fields

- Each community has its own topics and AI technologies
- Isolated “Research Islands”
Our View: Reusable Building Blocks

Scientific Research Problems

Tractable Scientific Tasks

Generic AI Components

- Mathematical Problem Solving
- Pattern Matching
- Prediction
- Artifact Enhancement
- Control
- Hypothesis and Confirmation

Artificial Intelligence
- Decision Trees
- Symbolic AI
- Neural Networks
- Optimizers
- Big Data
- Expert Systems
Agenda

• The Landscape of AI for Science
• The Definition of Scientific AI Benchmarking
• Challenges and Methodology
• System Design
• Case Studies
The Definition of Scientific AI Benchmarking

• The Goal: to evaluate an **AI method** applied to a **scientific task**, based on well-defined **metrics**.
Scientific Tasks: Problem Definition

- Training/Testing AI models boils down to data generation.
- Scientific tasks can be categorized into the following types:
  - Defined by Problem Class.
  - Defined by Problem Setting.
  - Defined by Problem Cases.

| Problem Class | Enumerate | Problem Setting | Generate | Problem Case | Dataset |
|---------------|-----------|-----------------|----------|--------------|---------|
| Generic mathematical definitions | Example: Partial Differential Equations | One particular instance of a problem class | Example: 1ns of Molecular Dynamics of one CH₄ | One particular case of a problem instance | Example: one electron microscopic image |

Purely Computation-Driven Purely Data-Driven
Evaluation **Metrics** in Scientific Tasks

• Different research communities have different interests in evaluation!

- **High Energy Physics, Climate Research, …**
  - Zettabytes of data
  - Prefer **Throughput**

- **Medical Imaging, Genomics …**
  - Very few data (hundreds of images!)
  - Prefer **Sample Efficiency**

- **Molecular Dynamics, Thermodynamics …**
  - Very precise (and expensive) non-AI methods exist
  - Prefer **Cost/Error Ratio**

- **Objective Function**
- **Performance Metrics**
Agenda

• The Landscape of AI for Science
• The Definition of Scientific AI Benchmarking
• Challenges and Methodology
• System Design
• Case Studies
Challenge: Unifying AI4Science and Benchmarking Efforts

• To build an inclusive and interconnecting environment for all the relevant research efforts
  - Scientific task definition
  - AI method, training algorithm, software and hardware environment
  - Metric definition and ranking definition
  - Deliver benchmarking result efficiently with given computation resources

• To follow the FAIR principle
  - Findability
  - Accessibility
  - Interoperability
  - Reuse
Our Methodology

• Modular Design: Define interfaces for various components so they can be mix-and-matched.
  • Available module types: Solver, Model, Problem, Metric, Ranking, Software, Hardware...

• Self-Descriptive Modules
  • A module not only executes, but also provides metadata for automatic discovery

• Automatic Benchmark Construction
  • Selecting tuples from the Cartesian product of compatible modules and build benchmarking cases
  • Example: when a new research community is onboarding, implement its ProblemDefinition module, and take advantage of automated benchmarking against existing AI methods, software/hardware configurations, metrics and ranking.
Agenda

• The Landscape of AI for Science
• The Definition of Scientific AI Benchmarking
• Challenges and Methodology
  • System Design
• Case Studies
System Design
System Design

Modules are implemented with our domain-specific language, SAIL.
System Design

The Planner conducts inner-join like operation to build benchmarking tuples and send them to the elastic computing platform.

Algorithm 1: Benchmarking Task Discovery.

```
Input: eDSL source files [src]
Output: Test scenarios [test]

repo ← φ;
// 1. Module discovery
foreach s : src do
    ast ← parse(s);
    foreach m : methods(ast) do
        if decorated(m, ProblemDefinition) then
            repo[0].append(m);
        if decorated(m, MetricDefinition) then
            repo[1].append(m);
        // Scan for other modules...
    ctx ← φ; test ← φ; iters ← iterators(repo);
i ← 0;
// 2. Task discovery
while i ≥ 0 do
    if i = N then
        test.append(ctx);
        ctx.pop();
i ← i − 1;
    else if next(iters[i]) then
        m ← get(iters[i]);
        metadata ← execute(m, ctx);
        if not Failed(metadata) then
            ctx.push(metadata);
i ← i + 1;
        else
            ctx.pop();
            rewind(iters[i]);
i ← i − 1;
return test
```
SAIL: Scientific AI domain-specific Language

• Previous AI benchmarking systems either implicitly define a series of built-in modules, or expose a markup language schema to define modules.

• Our approach: an embedded domain-specific language (eDSL) in Python, providing primitives for model construction, module self description, compatibility checks etc.
  • Better programmability
  • More discoverable
  • User friendly
SAIL Language Constructs

• The SAIL eDSL is built upon the language constructs of Python, to provide Scientific AI Benchmarking primitives.

• Primitive of different roles:
  • Metadata – e.g. module entry points
  • Input/Output, data type description – e.g. Tensor(28,28)
  • Computation graph construction – e.g. ReLU, FCLayer
  • Compatibility check – e.g. FAIL(“reason”)

| Feature               | Construct     | Instances                     |
|-----------------------|---------------|-------------------------------|
| Module Entry Points   | Decorators    | @ProblemDefinition            |
|                       |               | @MetricDefinition ...         |
| Type Descriptors      | Classes       | class Tensor                  |
|                       |               | class Scalar ...              |
| Concepts and Primitives | Well-known Global Objects | Train.Classify |
|                       |               | Model.Predict                 |
|                       |               | Test.Compare ...              |
| AI Models             | Declarative Methods | Pipeline |
|                       |               | Linear                        |
|                       |               | Relu                           |
|                       |               | Softmax ...                   |

Helps the Planner to “inner-join” the modules
SAIL Examples

- A Software definition module
- Collected by the Planner (Algorithm 1)
- Executed to validate the HW+SW configuration
- If successfully executed (without Fail), the Planner accepts the current module, and moves on to the next module type
SAIL Examples

- The Planner “Dry-run” the Model definition module to extract computation graph
- The SAIL Codegen generates actual benchmarking code against the model + software (e.g. Tensorflow)
- Executed to validate the HW+SW configuration
- If successfully executed (without Fail), the Planner accepts the current module, and moves on to the next module type

```python
@ModelDefinition
def MLP(task, inputType, outputType, hiddenSize):
    Suggest(hiddenSize, 128, 256, 512, 1024)
    if task == Classify:
        f = Softmax(outputType.dims)
    elif task == Enhance:
        f = id
    else:
        Fail("don't know how to solve")
    return Pipeline([Convert(inputType, Tensor),
                     Linear(hiddenSize), Relu(),
                     Linear(outputType.dims), f,
                     Convert(Tensor, outputType)])
```
Agenda

• The Landscape of AI for Science
• The Definition of Scientific AI Benchmarking
• Challenges and Methodology
• System Design
• Case Studies
Case Study: Molecular Dynamics

- An End-to-End Problem definition
  - Data I/O and format ("XYZ" atom descriptors)
  - Calling external software (MolecularDynamicsSoftware)
  - Train/Test phases

- Unique MD-specific metric: accumulated error
  - During testing, two independent trajectories are maintained (external MD software “golden”, and AI model “pred”)
  - In contrast to traditional “single step loss”, examines whether the trained model does not generate bias and accumulate error across many steps.
Thank You!