



Artificial Intelligence for Mission

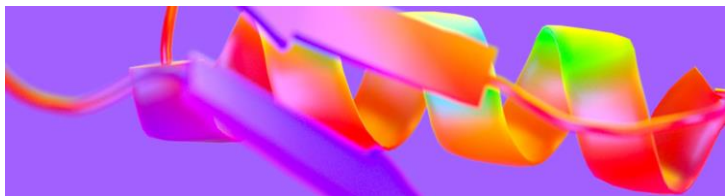
Earl Lawrence

December 19, 2024

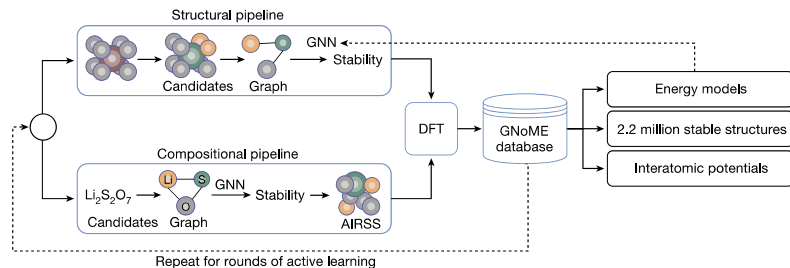


Industry has led the way on AI for science.

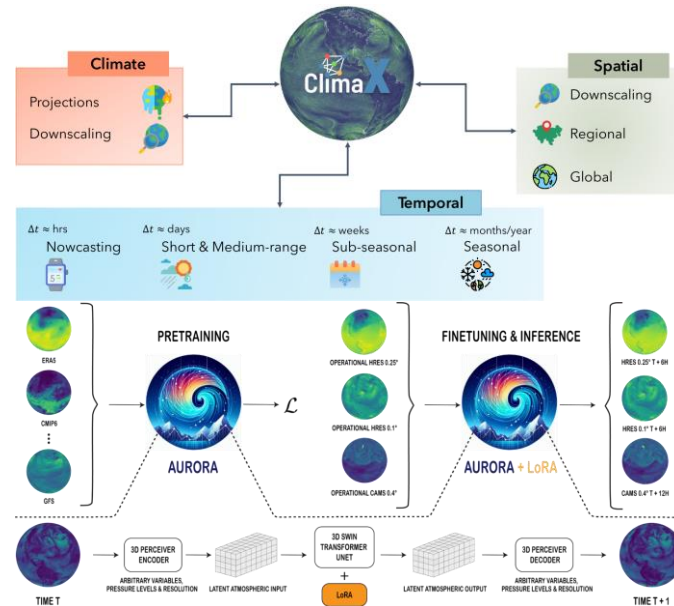
Protein Folding



Materials



Climate



We need to develop our own expertise to apply industry solutions and to solve problems they won't.

AI has become a priority across DOE and the federal government.



AI based surrogates for HPC

Climate Ensembles
Effective Zettascale on Exa

AI for software engineering and programming

Code Translation, Optimization
Quantum Compilation, QAIgs

AI for advanced properties inference and inverse design

Energy Storage
Proteins, Polymers

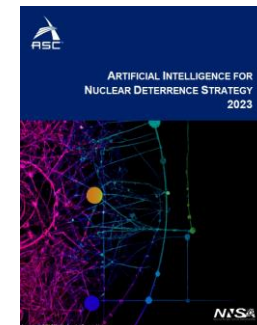
AI and robotics for autonomous discovery

Materials, Chemistry, Biology
Light-Sources, Neutrons

AI for prediction and control of complex engineered systems

Accelerators, Buildings, Cities,
Power Grid, Networks

Foundation AI for scientific knowledge

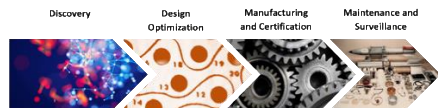


LANL began a Signature Institutional Commitment on AI.

Principles



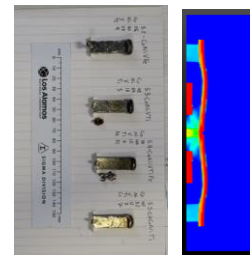
Large, coherent team.



Transformational advances.

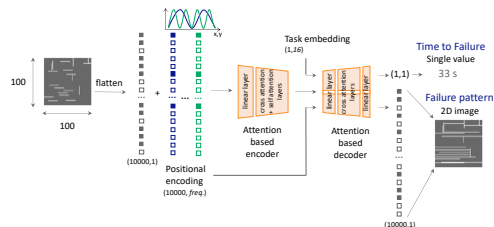


Infrastructure & partnerships



Unique LANL data

Goals



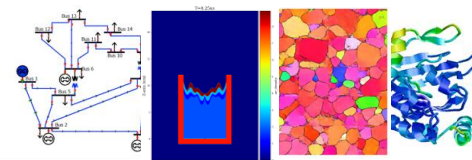
Foundation & generative AI.



MathOptAI.jl

MathOptAI.jl is a [Julia](#) extension for embedding trained AI, machine learning, and statistical learning models into a Julia optimization model.

Extensible solution engines.

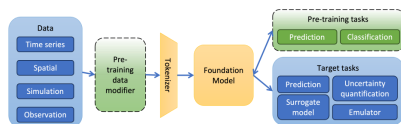


Small number of apps.

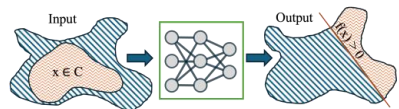
ArtIMis combines core AI development, key applications and support technology.

Core AI

Scientific Foundation Models

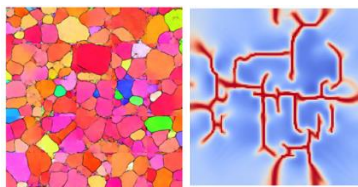


Design, Discovery & Control

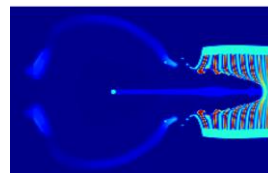


Applications

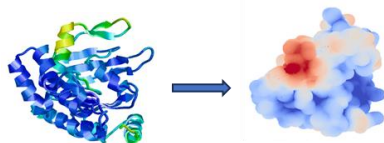
Materials Performance and Design



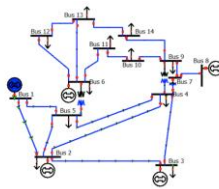
Multiphysics Systems



Biosecurity



Applied Energy



AI Support

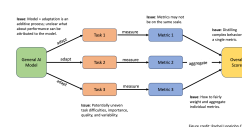
Data



Computation



Test & Evaluation

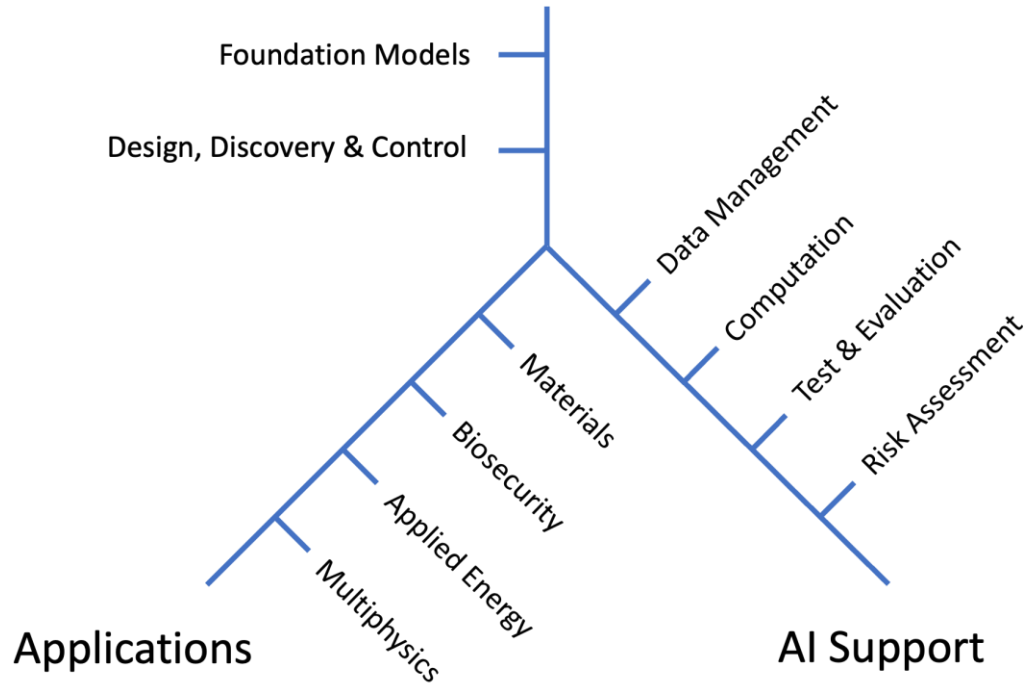


Risk Assessment



The long-term goal is tight integration and commodification of expertise.

Core AI



Project: Earl Lawrence, Nathan DeBardleben & Terry Turton

Foundation Models: Diane Oyen & Ayan Biswas

Design, Discovery & Control: Russell Bent & Mike Grosskopf

Materials Fracture: Hari Viswanathan & Dan O'Malley

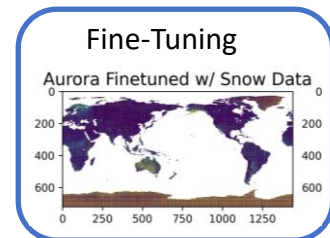
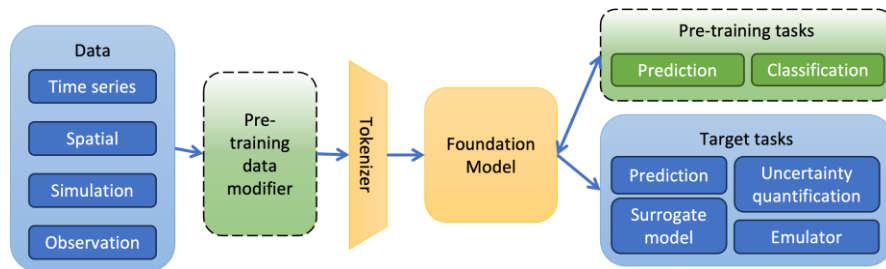
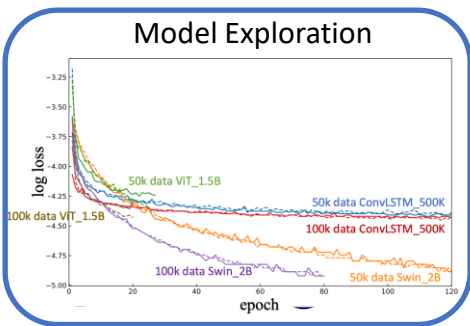
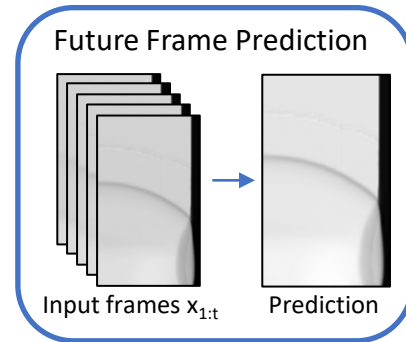
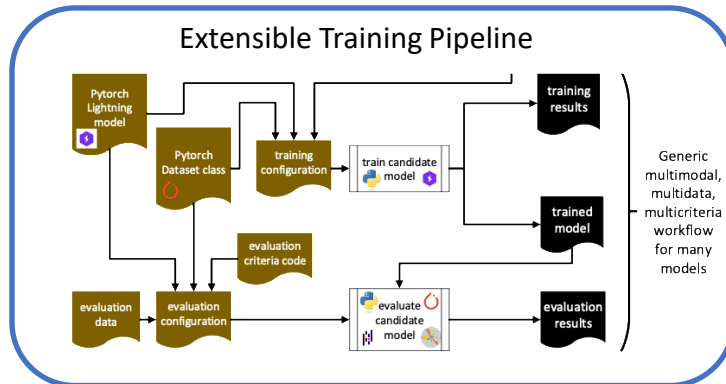
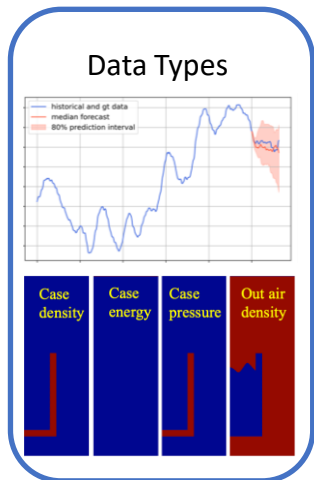
Materials Discovery: Saryu Fensin & Tim Germann

Applied Energy: Kaarthik Sundar

Multiphysics: Kyle Hickmann

AI Support: Emily Casleton & Christine Sweeney

We are building capability for foundation model construction and research on core issues.



We are developing extensible and trustworthy AI tools for design, discovery, and control.

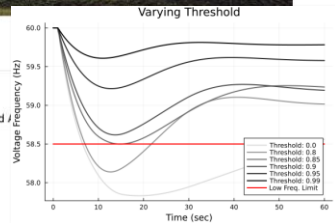
Direct optimization for AI models applied to power flow.



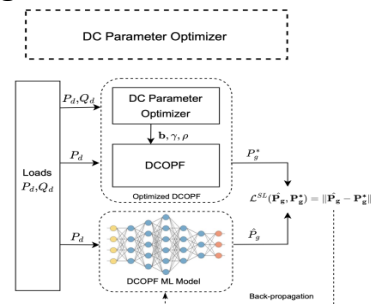
MathOptAI.jl

CI passing codecov 100%

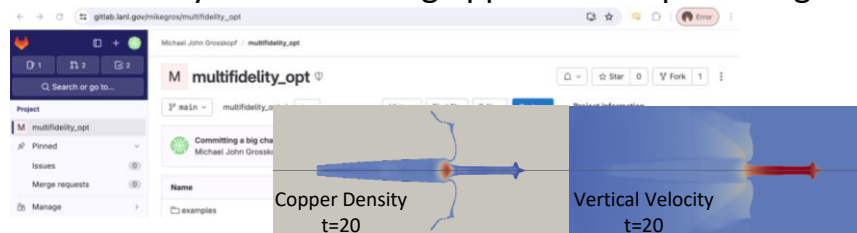
MathOptAI.jl is a JUMP extension for embedding trained a JUMP optimization model.



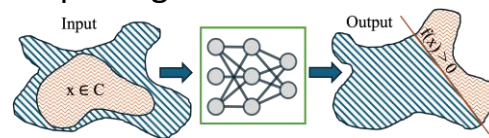
End-to-end learning applied to grid control.



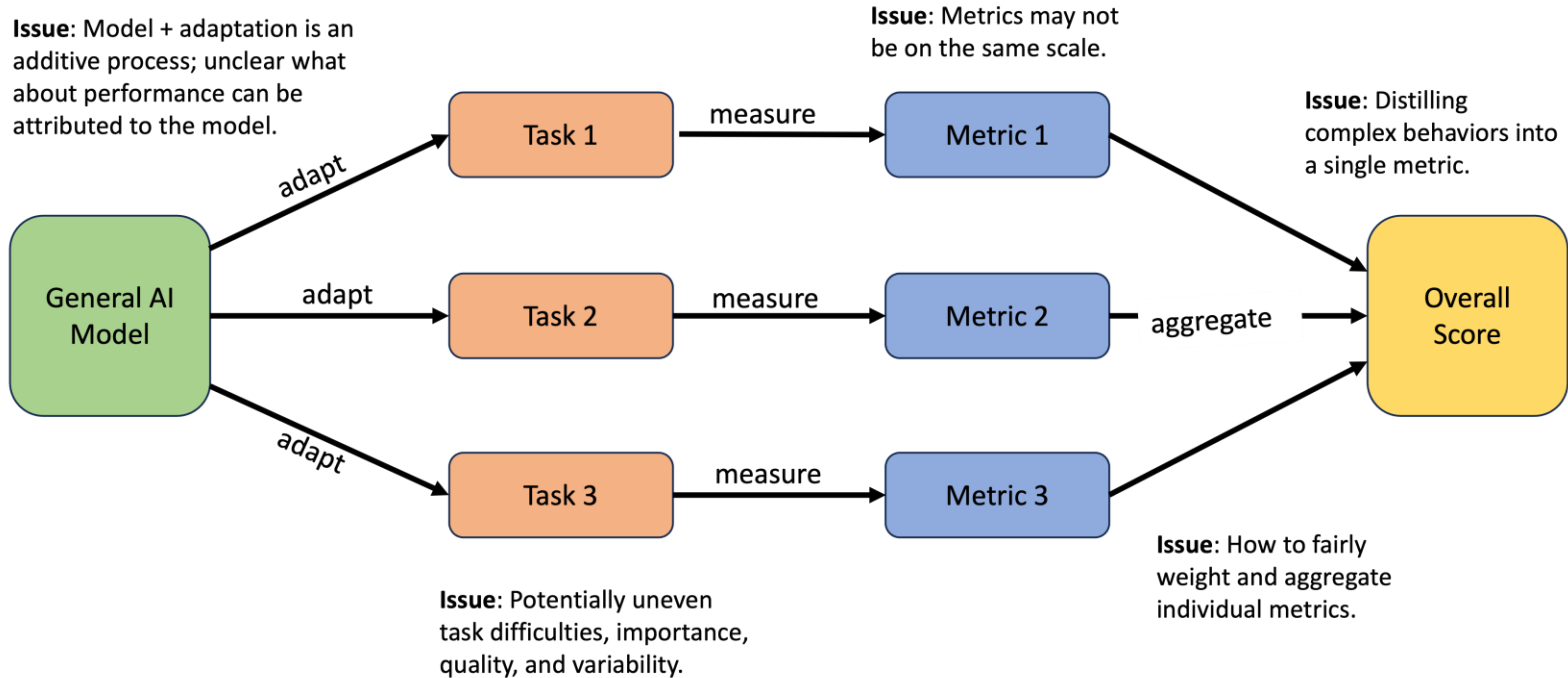
Multi-fidelity active learning applied to shaped charges.



Mathematically verifiable input regions.

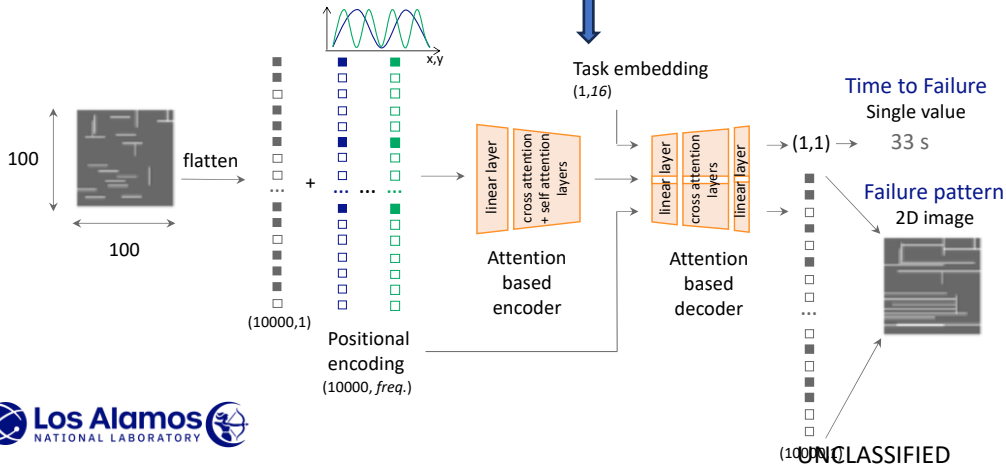


Test & evaluation is a vital open problem in a mission critical environment.

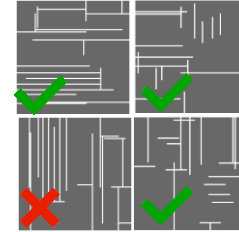


We've built a large multi-modal for material fracture.

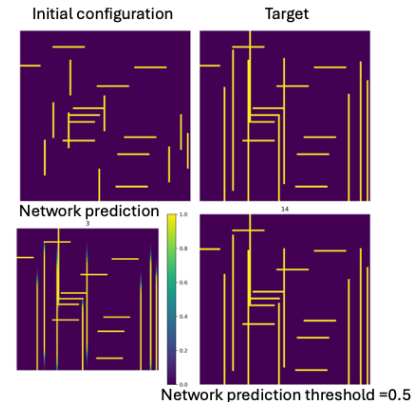
This is a simple simulation of a X/T growth where a shear-stress is applied in the **horizontal/vertical** direction. Provide a one token embedding for it.



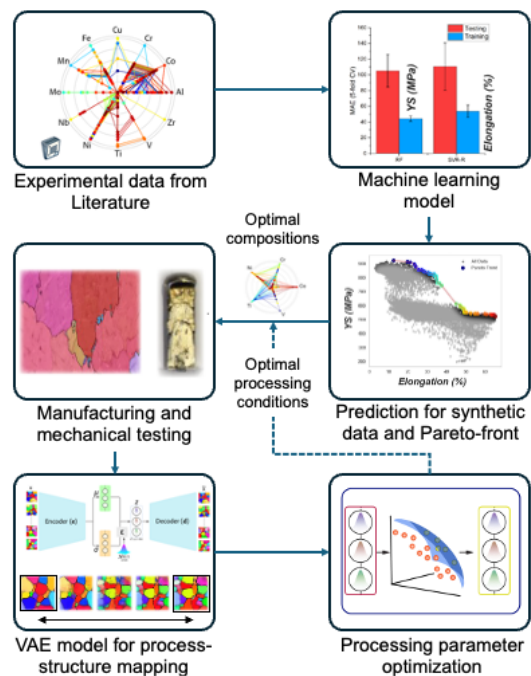
Training data includes 3 out of 4.



Model predicts vertical-X well anyway.



We've constructed a design loop for new materials, including a generative model for microstructure.

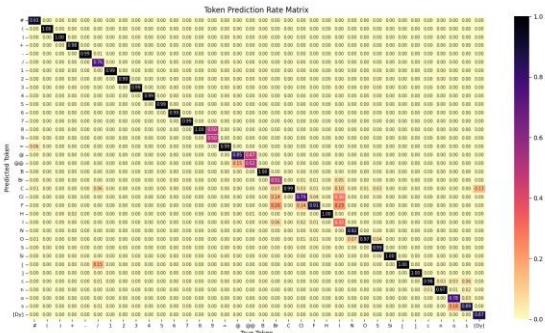
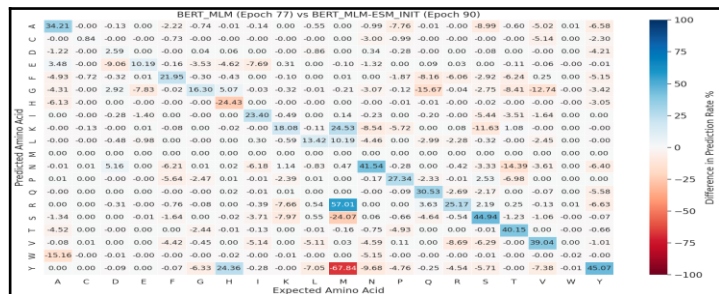


Includes our first steps toward an automated Lab.

We are developing a multi-modal bio-foundation model for protein interactions.

Leveraged an industry model to build a protein language model.

Fine-tuned the MIST model from UM.



Evaluated models for small molecule interactions.

Model	# P	Epoch	Acc	Train Score	Eval Score	Performance Drop
ResNet50	23,514,179	51:19	0.998333	0.226220	0.119410	0.878923
Wide_ResNet50_2	66,840,387	1:16:27	0.998851	0.231380	0.128400	0.870451
Edge Feature GNN	21,956	42:55	0.914060	0.206010	0.123750	0.79031
GAT	901,379	1:27:52	0.952214	0.315820	0.188130	0.764084
GAT-ChemBERTa-77M-MTR	4,353,395	6:22:40	0.947519	0.36018	0.21769	0.729829

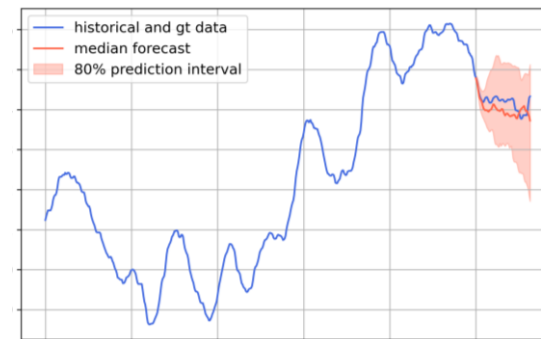
We are building Bayesian solutions for power grid control and a foundation model for grids.

118 node system with 1500 labeled samples and 600 seconds of training time

Method	Gap%	Max Eq.	Mean Eq.	Max Ineq.	Mean Ineq.
Sandwich BNN SvP (Ours)	1.484	0.089	0.018	0.008	0.000
Sandwich BNN (Ours)	1.485	0.100	0.016	0.008	0.000
Supervised BNN SvP (Ours)	1.568	0.147	0.022	0.013	0.000
Supervised BNN (Ours)	1.567	0.205	0.020	0.013	0.000
Naïve MAE	1.638	2.166	0.187	0.000	0.000
Naïve MSE	1.622	3.780	0.242	0.000	0.000
MAE + Penalty	1.577	1.463	0.102	0.000	0.000
MSE + Penalty	1.563	2.637	0.125	0.000	0.000
LD + MAE	1.565	1.284	0.083	0.000	0.000

With 30000 labeled training data & 9200 seconds state-of-art can match our results

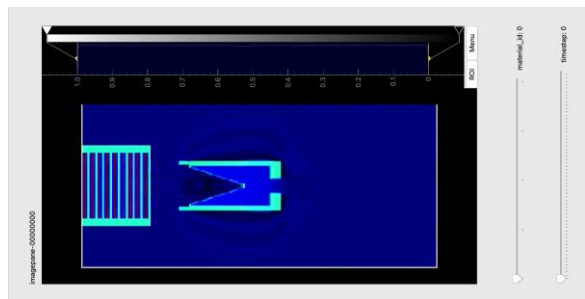
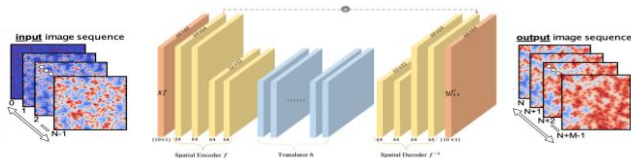
Our approach produces accurate results with a fraction of the training data required by others.



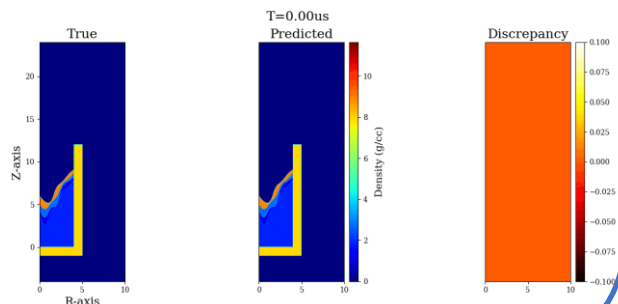
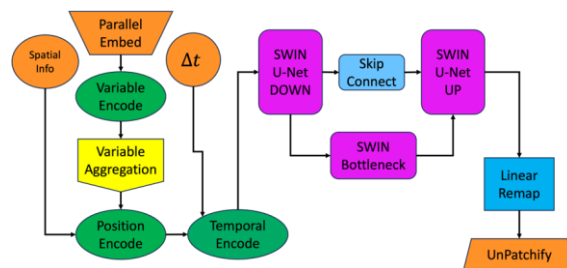
We are exploring time series foundation models to predict behavior using 60TB of power grid data.

We are prototyping physics foundation models and exploring automated design for shaped charges.

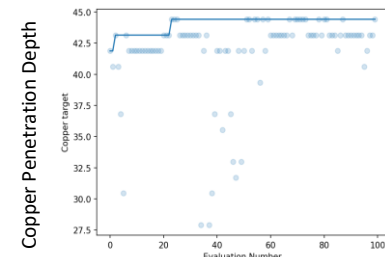
Model Exploration



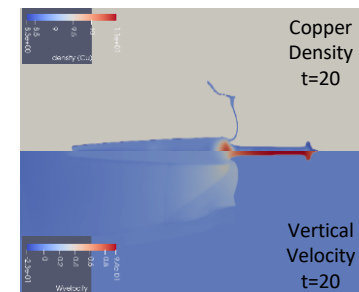
“Production” Model



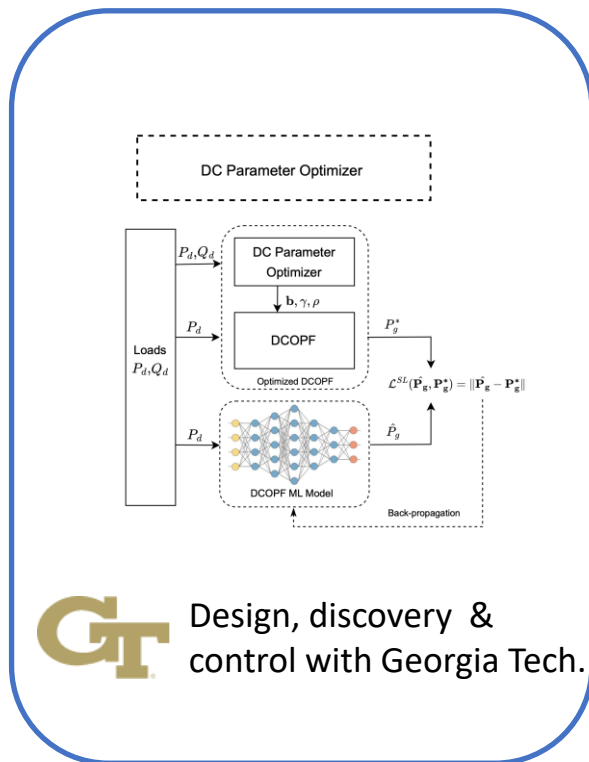
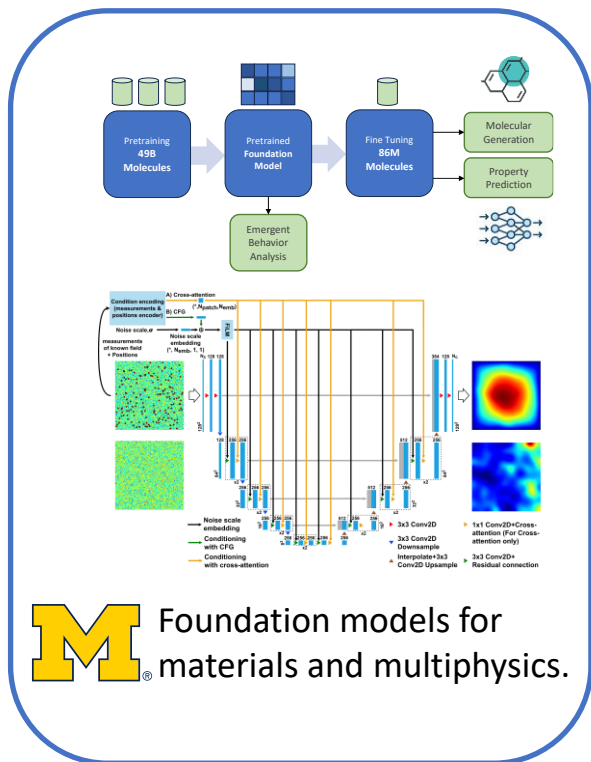
Design Optimization



3D Sim Iteration Guided by 2D Surrogate



University partnerships have expanded our ability to make progress on fundamentals.



LANL Home / Engage / Collaboration / NNSC /

AI Science at Scale

Opportunity for University of California System Researchers and National Laboratory Collaborators to Advance Artificial Intelligence Science at Scale

The University of California (UC) System, in partnership with NNSA laboratories, is seeking proposals from UC researchers for collaborative projects that advance the frontiers of AI and Machine Learning science at scale.

UNIVERSITY OF CALIFORNIA Applications.

Industry partnerships have provided access to tools and accelerated progress.



Training and support for model building.



OpenAI and Los Alamos National Laboratory announce bioscience research partnership

OpenAI and Los Alamos National Laboratory are developing evaluations to understand how multimodal AI models can be used safely by scientists in laboratory settings.

Strawberry Evaluation Results

Project	Novelty	Productivity	Solution	Strength	Importance
Subcritical limit calculations	3	4	4	3	5
High entropy alloy design	2	2	3	2	4
CPU to GPU translation		5	4	4	5
Phasor measurement unit placement	4	5	4	4	4
Adaptive control for dynamic systems	3	4	2	2	4
Quantum algorithms for fracture	3	4	4	4	4
Novel protein design	3	5	3	4	5
ICF Target Design	2	4	2	3	4
Bayesian optimization properties		4	3	3	4
Geodesic embedding in color space	3	4	3	4	5
System reliability modeling	4	5	4	3	4
Combustion science	3	2	3	3	5
Sparse sensing	2	3	2	3	3
Cybersecurity	3	5	4	4	5
Expected value proof	3	4	5	4	5
Endogenous models	3	4	3	4	5

Early access for test and evaluation.

Deeper partnerships are providing us with a new opportunity to build agentic discovery systems.

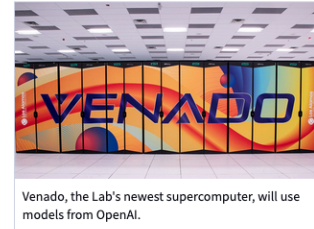
Lab partners with OpenAI to advance national security

OpenAI's latest reasoning models will be used on Venado supercomputer

Los Alamos National Laboratory has entered a partnership with OpenAI to install its latest o-series models — capable of expert reasoning for a broad span of complex scientific problems — on the Lab's [Venado](#) supercomputer, which uses NVIDIA GH200 Grace Hopper Superchips, to conduct national security research.

"As threats to the nation become more complex and more pressing, we need new approaches and advanced technologies to preserve America's security," said Laboratory Director **Thom Mason**. "Artificial intelligence models from OpenAI will allow us to do this more successfully, while also advancing our scientific missions to solve some of the nation's most important challenges."

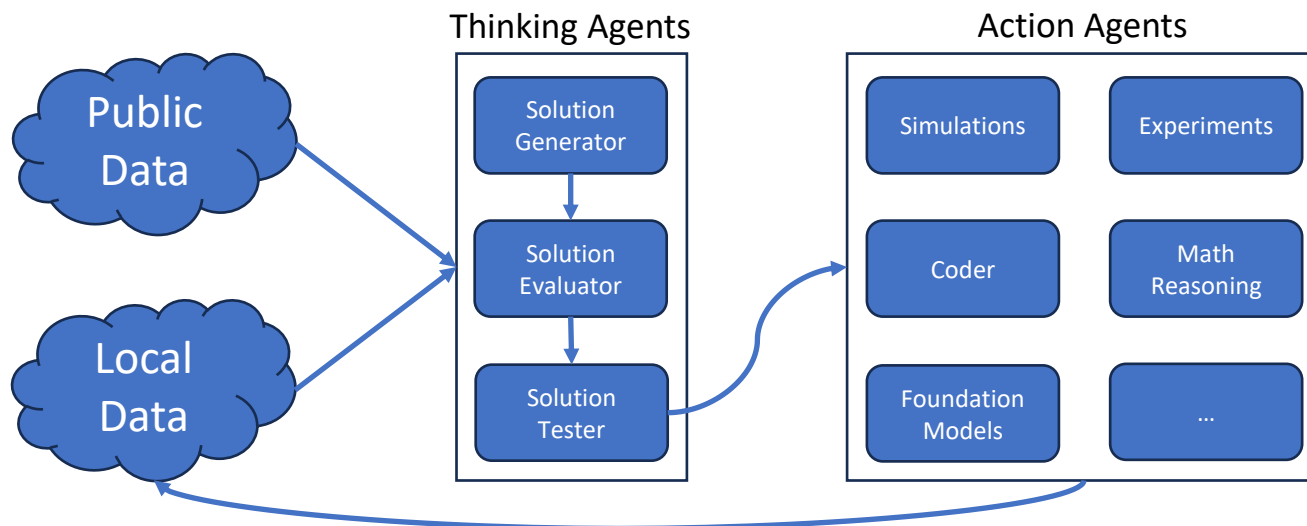
The Venado machine will be moved to a secure, classified network where it will be a shared resource for researchers from Los Alamos, Lawrence Livermore, and Sandia national labs.



TODAY'S TOP NEWS

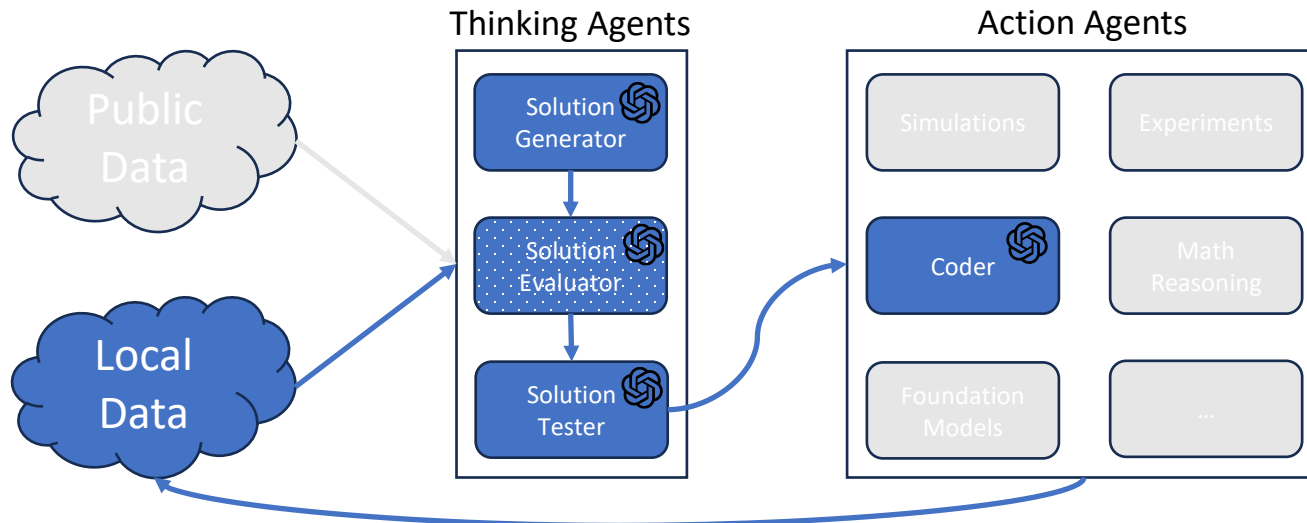
We're also have a new collaboration with NVIDIA to fine-tune a model for ICF.

We've begun an agentic discovery system with two tiers of agents that utilize frontier AI.



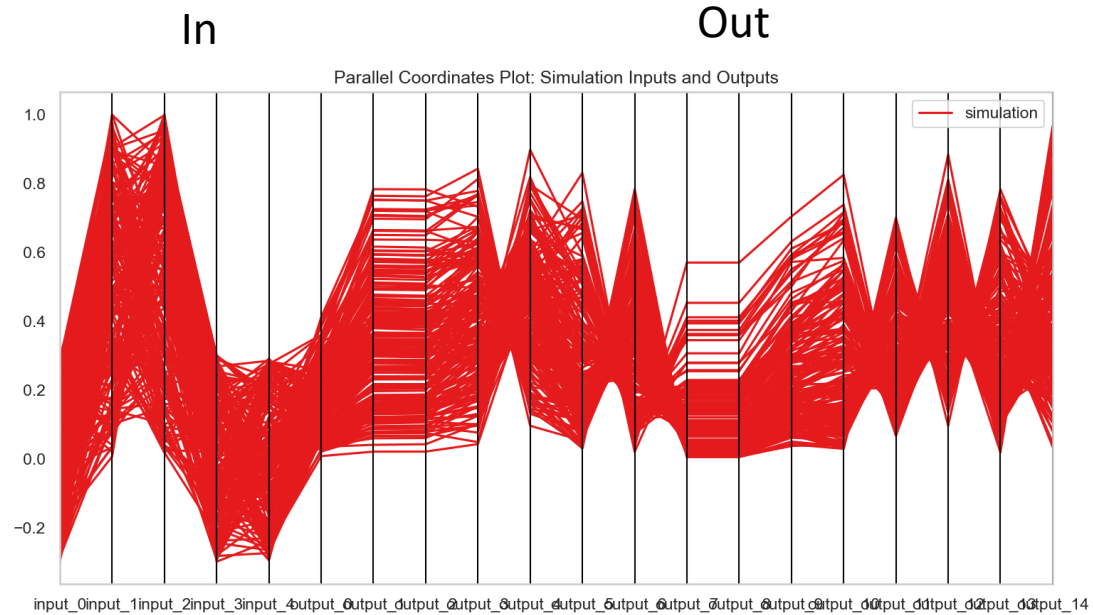
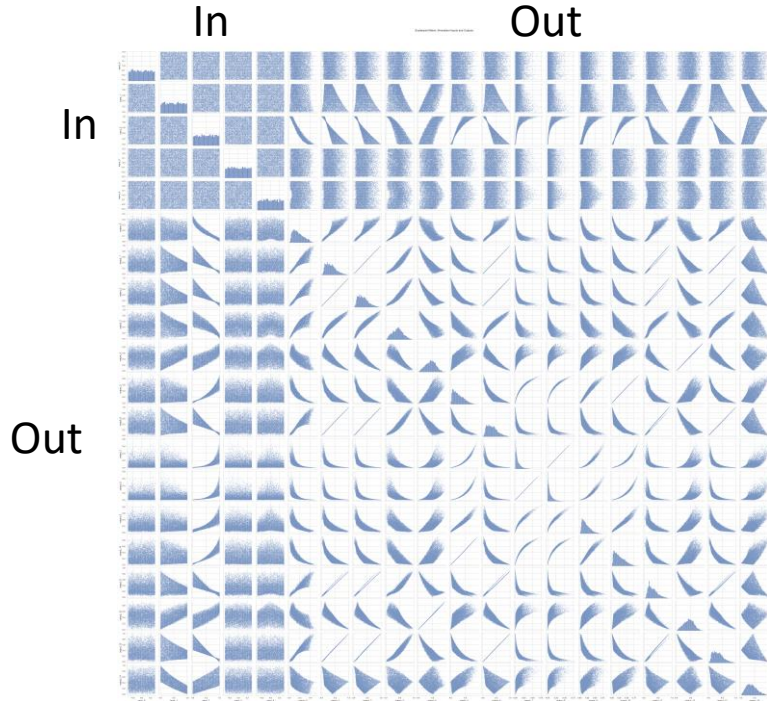
We've built a coding and ML system as a proof-of-principle.

🌀 Agents are powered by 03-mini or 4o.



This is all we need to replace me!

We have enough tools to solve a toy-ish inverse problem.



We start by asking it instructions.

```
specification.yaml ×
1 provide_directory: {"local_directory" : "./data/icf-jag-10k", "as_dir_name": "data" }
2 task: |
3   Your task is:
4
5   1. You have access to a directory ./data/ - please do not write or create anything in that directory (leave it
6   | pristine).
7   2. in there, you will find 3 files:
8   |   1. input parameters to an experiment (scalars) - jag10K_params.npy
9   |   2. output parameters from that experiment (again, scalars) - jag10K_0_scalars.npy
10  | 3. Build a forward model and test that it gives good results.
11  | 4. I have some results from an experiment. Using the forward model, can you find all inputs to the model
12  | that could have resulted in this output (below, marked "Experiment Output")
13  | NOTE: There could be more than one.
14
15  Experiment Output: 0.11582408, 0.2535939 , 0.25252718, 0.2916376 , 0.507854,
16  |                   0.25274113, 0.25506145, 0.05349015, 0.05349026, 0.16656016,
17  |                   0.24080992, 0.22401391, 0.5013491 , 0.2566557 , 0.29012832
```

First, it proposes a possible solution.

```
[ robust quokka Foreman ] We will create  
The files considered are 'jag10K_params.npy' for in  
shape (N, D_in) and (N, D_out) respectively. 2. But  
to build a model with one or two hidden layers. After  
well. 3. With the trained model, we will solve the  
0.25252718, 0.2916376, 0.507854, 0.25274113, 0.2550  
will use gradient based optimization (treating the  
solutions. We then filter out nearly identical solu  
This approach complies with the requirement not to
```

The coding agent produces code to train and invert the model.

```

1 import os
2 import numpy as np
3 import torch
4 import torch.nn as nn
5 import torch.optim as optim
6
7 # Define the model
8 class ForwardModel(nn.Module):
9     def __init__(self, input_dim=5, output_dim=15):
10         super(ForwardModel, self).__init__()
11         self.net = nn.Sequential(
12             nn.Linear(input_dim, 64),
13             nn.ReLU(),
14             nn.Linear(64, 64),
15             nn.ReLU(),
16             nn.Linear(64, output_dim)
17         )
18     def forward(self, x):
19         return self.net(x)
20
21
22 def get_device():
23     if torch.cuda.is_available():
24         device = torch.device('cuda')
25     elif hasattr(torch.backends, 'mps') and torch.backends.mps.is_available():
26         device = torch.device('mps')
27     else:
28         device = torch.device('cpu')
29     print(f'Using device: {device}')
30     return device
31

```

```

32
33 def train_model(model, X_train, y_train, device, epochs=1000, lr=0.001, target_loss=1e-4):
34     model.to(device)
35     criterion = nn.MSELoss()
36     optimizer = optim.Adam(model.parameters(), lr=lr)
37     X_train = torch.tensor(X_train, dtype=torch.float32).to(device)
38     y_train = torch.tensor(y_train, dtype=torch.float32).to(device)
39     for epoch in range(epochs):
40         optimizer.zero_grad()
41         outputs = model(X_train)
42         loss = criterion(outputs, y_train)
43         loss.backward()
44         optimizer.step()
45         if (epoch+1) % 100 == 0:
46             print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.6f}')
47         if loss.item() < target_loss:
48             print(f'Training loss reached target at epoch {epoch+1}')
49             break
50     return model
51
52
53 def inverse_search(model, target_output, initial_inputs, device, candidates_needed=50, steps=300, lr=0.01, tol=
54     '\n Given a trained model and a target output, try to find candidate inputs that produce outputs close to
55     model.eval()
56     candidate_inputs = []
57     criterion = nn.MSELoss()
58
59     # We can get the min and max from initial inputs for good random initialization bounds
60     mins = initial_inputs.min(axis=0)

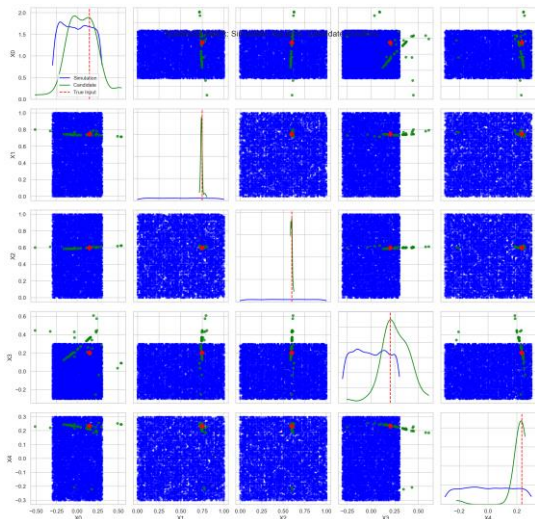
```

It executes the code.

```
[ robust quokka Foreman ] [REASONING] -->  
[ robust quokka Foreman ] Now that the Python file has been created, the next step is to run the script to train the forward model and then solve the inverse problem. I will now invoke the RUN_PYTHON command to execute the 'main' function in forward_inverse.py.
```

```
Running main in forward_inverse.py . . .  
Running main...: Elapsed: 00:00 Training forward model...  
Running main...: Elapsed: 00:01 Epoch 1/500, Loss: 0.140454  
Epoch 100/500, Loss: 0.001451  
Epoch 200/500, Loss: 0.000656  
Running main...: Elapsed: 00:02 Epoch 300/500, Loss: 0.000296  
Epoch 400/500, Loss: 0.000206  
Running main...: Elapsed: 00:03 Epoch 500/500, Loss: 0.000157
```

It produces a list of candidate solutions, including the correct answer.



inputs

array([0.14676411, 0.74648325, 0.60003832, 0.20279975, 0.2339081])

```

Found candidate from init 49 with loss 2.083950e-05
Candidates found:
Candidate 1:\nInput: [-0.26702017  0.76822275  0.5979348  0.28140718 -0.24466452]\nAchieved error: 2.692693e-05\n
Candidate 2:\nInput: [-0.04819451  0.7364068  0.6091959 -0.01907833 -0.24594866]\nAchieved error: 3.390869e-05\n
Candidate 3:\nInput: [-0.7596585  0.7552812  0.5640864 -0.1511701 -0.2412163]\nAchieved error: 9.411368e-05\n
Candidate 4:\nInput: [-0.18713363  0.7441142  0.5837594 -0.59588146  0.16615286]\nAchieved error: 2.952054e-05\n
Candidate 5:\nInput: [-0.60025966  0.7455011  0.5816006 -0.03267683  0.258708 ]\nAchieved error: 2.203939e-05\n
Candidate 6:\nInput: [-0.08578689  0.75339586  0.6029525  0.2711217 -0.24235773]\nAchieved error: 3.345796e-05\n
Candidate 7:\nInput: [-0.5930684  0.7460606  0.5825953 -0.04246255  0.25821146]\nAchieved error: 2.174766e-05\n
Candidate 8:\nInput: [-0.44195276  0.7544062  0.5919973 -0.6037008  0.07833531]\nAchieved error: 4.028691e-05\n
Candidate 9:\nInput: [ 0.5285942  0.7374647  0.5983983 -0.20526485 -0.22138026]\nAchieved error: 3.608155e-05\n
Candidate 10:\nInput: [ 0.42960927  0.7405772  0.577673  0.06881108 -0.24098127]\nAchieved error: 2.070985e-05\n
Candidate 11:\nInput: [-0.1296529  0.7505203  0.5877726 -0.31634033  0.23509973]\nAchieved error: 9.723576e-06\n
Candidate 12:\nInput: [ 0.20371872  0.7463747  0.5842964 -0.4813294  0.2212381 ]\nAchieved error: 2.510764e-05\n
Candidate 13:\nInput: [ 0.45778593  0.7401218  0.5788551  0.04210383 -0.243077 ]\nAchieved error: 2.196692e-05\n
Candidate 14:\nInput: [-0.1370684  0.7499877  0.5925709 -0.1480623  0.24014984]\nAchieved error: 8.230691e-06\n
Candidate 15:\nInput: [ 0.6341094  0.7558946  0.5712222  0.21614937 -0.2149637 ]\nAchieved error: 5.080463e-05\n
Candidate 16:\nInput: [ 0.47907782  0.74031365  0.5779905 -0.19055253 -0.21975738]\nAchieved error: 2.176614e-05\n
Candidate 17:\nInput: [-0.5056341  0.748141  0.58577865 -0.08380464  0.252931 ]\nAchieved error: 1.800514e-05\n
Candidate 18:\nInput: [-0.46701553  0.7483792  0.5861031 -0.0941631  0.25116783]\nAchieved error: 1.705307e-05\n
Candidate 19:\nInput: [ 0.6249961  0.73666346  0.58718497 -0.3131483 -0.19905834]\nAchieved error: 3.380855e-05\n
Candidate 20:\nInput: [-0.6561109  0.76220953  0.57058454  0.00978289  0.26265973]\nAchieved error: 4.139952e-05\n
Candidate 21:\nInput: [ 0.26017573  0.74340427  0.5945556 -0.02430945 -0.24587028]\nAchieved error: 2.020815e-05\n
Candidate 22:\nInput: [-0.3588749  0.7464171  0.6015088 -0.21096362 -0.24303496]\nAchieved error: 1.446839e-05\n
Candidate 23:\nInput: [-0.5356471  0.74783236  0.5856682 -0.06503789  0.25471708]\nAchieved error: 1.959911e-05\n
Candidate 24:\nInput: [-0.2632309  0.738221  0.59877944  0.14184625  0.24295413]\nAchieved error: 2.427159e-05\n
Candidate 25:\nInput: [-0.42427287  0.74905986  0.58685905 -0.10160045  0.2494304 ]\nAchieved error: 1.556828e-05\n
Candidate 26:\nInput: [-0.47807682  0.7329501  0.5905652 -0.05303667  0.2512845 ]\nAchieved error: 2.645153e-05\n
Candidate 27:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 28:\nInput: [-0.6616330  0.74030375  0.593330  0.00700053  0.23031093]\nAchieved error: 1.049373e-05\n
Candidate 29:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 30:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 31:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 32:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 33:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 34:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 35:\nInput: [-0.50143093  0.75023021  0.5947030  0.20100001 -0.22020371]\nAchieved error: 1.604834e-05\n
Candidate 36:\nInput: [0.15673074  0.73619366  0.6109944  0.1936809  0.24072999]\nAchieved error: 4.121630e-05\n
Candidate 37:\nInput: [ 0.55520080  0.7380919  0.5814030  0.2209976 -0.2441710 ]\nAchieved error: 1.7095353e-05\n
Candidate 38:\nInput: [ 0.4456304  0.7407227  0.5778576  0.10703648 -0.23392908]\nAchieved error: 2.083036e-05\n

```

The system can evaluate its solutions and take new actions.

```
Epoch 1992, Train MSE: 0.149624, Holdout MSE: 0.355328
Epoch 1993, Train MSE: 0.149623, Holdout MSE: 0.355326
Epoch 1994, Train MSE: 0.149622, Holdout MSE: 0.355324
Epoch 1995, Train MSE: 0.149621, Holdout MSE: 0.355322
Epoch 1996, Train MSE: 0.149620, Holdout MSE: 0.355320
Epoch 1997, Train MSE: 0.149619, Holdout MSE: 0.355318
Epoch 1998, Train MSE: 0.149618, Holdout MSE: 0.355317
Epoch 1999, Train MSE: 0.149617, Holdout MSE: 0.355315
Epoch 2000, Train MSE: 0.149616, Holdout MSE: 0.355313
```

ion to improve based on output from executed command.

M with conversation state...

's reply to the conversation state.

```
[ khaki caribou Foreman ] REASONING: Our simple linear model did not converge sufficiently to match the held-out input. Let us create a new Python file with a more flexible model (e.g., a small neural network) and train further until we get closer to the holdout input.
```

```
[ khaki caribou Foreman ] ACTION: CREATE_PYTHON --> {filename: advanced_inverse_model.py, python_code_content: [ import numpy as np, import random, import tensorflow as tf, 'def train_advanced_inverse_model():', ' # 1) Load the data and holdout row from the previous step.', " input_data = np.load('jag10K_params.npy')", " output_data = np.load('jag10K_0_scalars.npy')", " # We'll read the holdout.txt to figure out the holdout row index or just recompute it.", " # For simplicity, let's just choose the same random approach (though we won't overwrite holdout.txt).", '', ' n_samples = input_data.shape[0]', ' holdout_index = random.randint(0, n_samples - 1)', ' holdout_input = input_data[holdout_index].copy()', ' holdout_output = out...']
```

STDERR:

Exception Traceback:

Traceback (most recent call last):

File "/Users/ndebard/lunar-crucible/Smith.py", line 250, in run_dynamic_call

result = dynamic_call()

^^^^^^^^^^^^^^^^

File "/Users/ndebard/lunar-crucible/Smith.py", line 241, in spec.loader.exec_module(module)

File "<frozen importlib._bootstrap_external>", line 9

File "<frozen importlib._bootstrap>", line 241, in _call_spec.loader.exec_module(module)

File "/Users/ndebard/lunar-crucible/projects/khaki-caribou/Smith.py", line 250, in run_dynamic_call

import tensorflow as tf

ModuleNotFoundError: No module named 'tensorflow'

```
[ khaki caribou Smith ] Modifying conversation to improve based on output from executed command.
```

```
[ khaki caribou Smith ] Round Trip: 7
```

```
[ khaki caribou Smith ] Sending query to LLM with conversation state...
```

```
[ khaki caribou Smith ] Appending assistant's reply to the conversation state.
```

```
[ khaki caribou Foreman ] TODO need to evaluate this action, for now passing
```

```
[ khaki caribou Foreman ] REASONING: TensorFlow is not installed, so we'll switch to using scikit-learn's MLPRegressor to create a small neural net to predict inputs from
```

```
[ khaki caribou Foreman ] ACTION: CREATE_PYTHON --> {filename: 'sklearn_inverse_model.py', python_code_content': ['import numpy as np', 'import random', 'from sklearn.neural_network import MLPRegressor', 'from sklearn.metrics import mean_squared_error', '', 'def train_sklearn_inverse_model():', ' # Load the data', " input_data = np.load('jag10K_params.npy')", " output_data = np.load('jag10K_0_scalars.npy')", ' n_samples = input_data.shape[0]', '', ' # pick a random row to hold out (like before)', ' holdout_index = random.randint(0, n_samples - 1)', ' holdout_input = input_data[holdout_index].copy()', ' holdout_output = out...']
```

In the end, it produces a human-understandable evaluation and grades itself.

```
[ robust quokka   Forgemaster ] Critique of Approach:
The task undertaken involved building and testing a forward model to establish a relationship between input parameters and scalar outputs from an experiment. The solution required training a neural network model using PyTorch to replicate this relationship and then leveraging it to solve an inverse problem: identifying input configurations that could have resulted in a given experimental output array. The process involved data loading, model training, and iterative optimization to find multiple valid input sets corresponding to the experiment output.

Critique:
1. Data Handling: The data loading was executed properly without writing to the specified pristine directory, adhering to the constraints. The use of NumPy for handling .npy files was efficient.
2. Model Architecture: The choice of PyTorch for constructing a neural network as the forward model was appropriate given its flexibility and GPU acceleration benefits. The network architecture with two hidden layers and ReLU activations is a sensible balance between complexity and computational feasibility for a scalar output regression task.
3. Training Procedure: The training of the forward model was efficient as indicated by the steadily decreasing loss over epochs. This suggests that the model learnt the mapping well.
4. Inverse Problem Solving: Gradient-based optimization was appropriately used to tackle the inverse problem. By running multiple initializations and checking for unique solutions, a diverse set of candidate inputs was derived, which fit the experimental outputs well. This method demonstrates robustness and adaptability in capturing potential solutions.
5. Result Presentation: The results were displayed with achieved errors, indicating the quality of each candidate solution. The solution addresses the task requirements by not only identifying one but several potential input configurations.

Overall, the process was executed well, demonstrating both technical proficiency and adherence to the task's constraints. However, the approach could be enhanced by potentially exploring different model architectures or by clustering solutions to identify dominant patterns among candidates if further interpretation of results is required.

Grade: A** - The steps undertaken comprehensively met the task requirements with effective use of tools and methodologies. There was room for extending interpretations or exploring further optimization strategies to refine the inverse solutions, but the main objectives were achieved successfully.
```