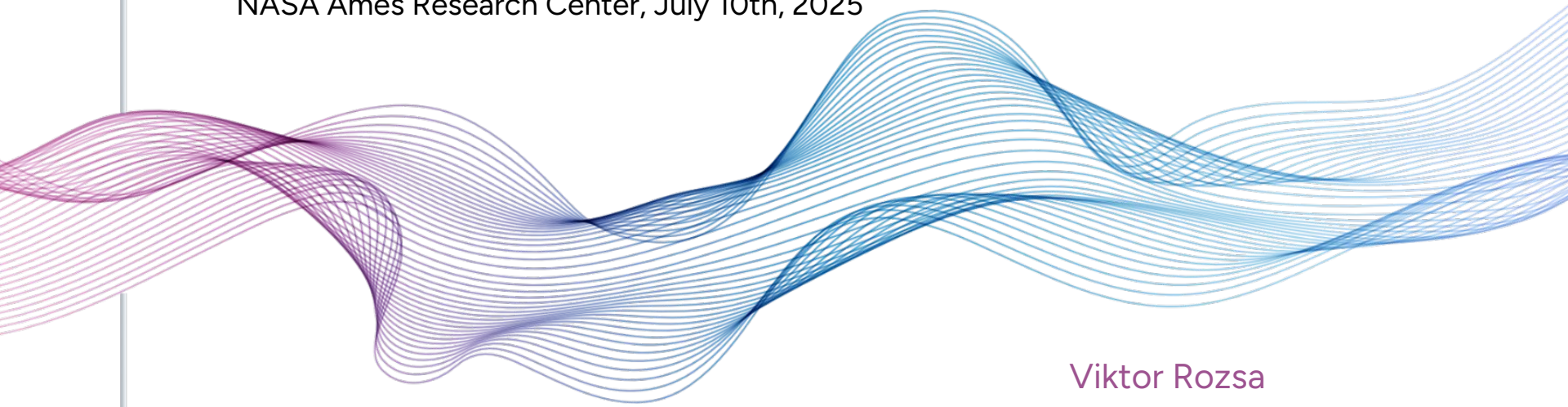
rescale

Accelerating CFD with Modular Surrogates: Insights from MeshGraphNets and Neural Operator Architectures

Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, July 10th, 2025



Viktor Rozsa
Solutions Architect, Rescale

Agenda

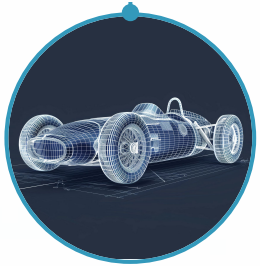
- Motivations: accelerating engineering innovation
- Graph Neural Nets and MGN overview
- Surrogate Modeling using MGNs: B787 Study
- Demo of Continuous Workflow
- Beyond mesh graph based methods: DoMINO
- Building surrogate models using NASA code stacks（Fun3D, CAPE, etc)
- Conclusions

Engineering and Scientific R&D are Undergoing a Transformation

Intuition-led Engineering

Data and AI-augmented Engineering

1960 — **1980** — **2015** — **2024**



Drawn Wireframe



Computer-Aided Design (CAD)



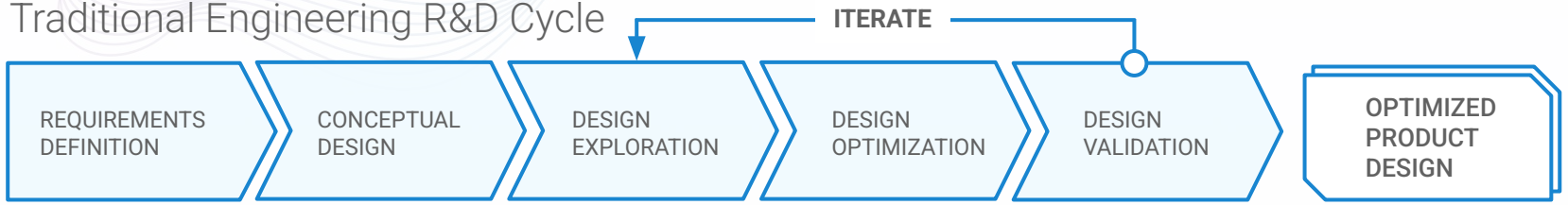
Computer-Aided Engineering (CAE)



Computational Science & Engineering (CSE)

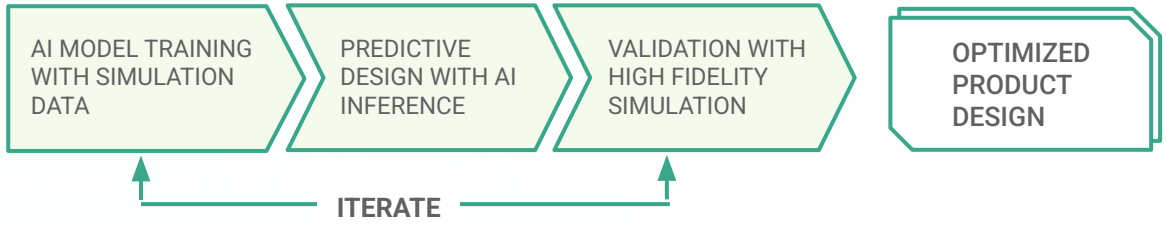
The Traditional Engineering Cycle is too Slow

Traditional Engineering R&D Cycle



DAYS OR WEEKS

AI-Augmented Digital R&D Cycle



MINUTES OR HOURS

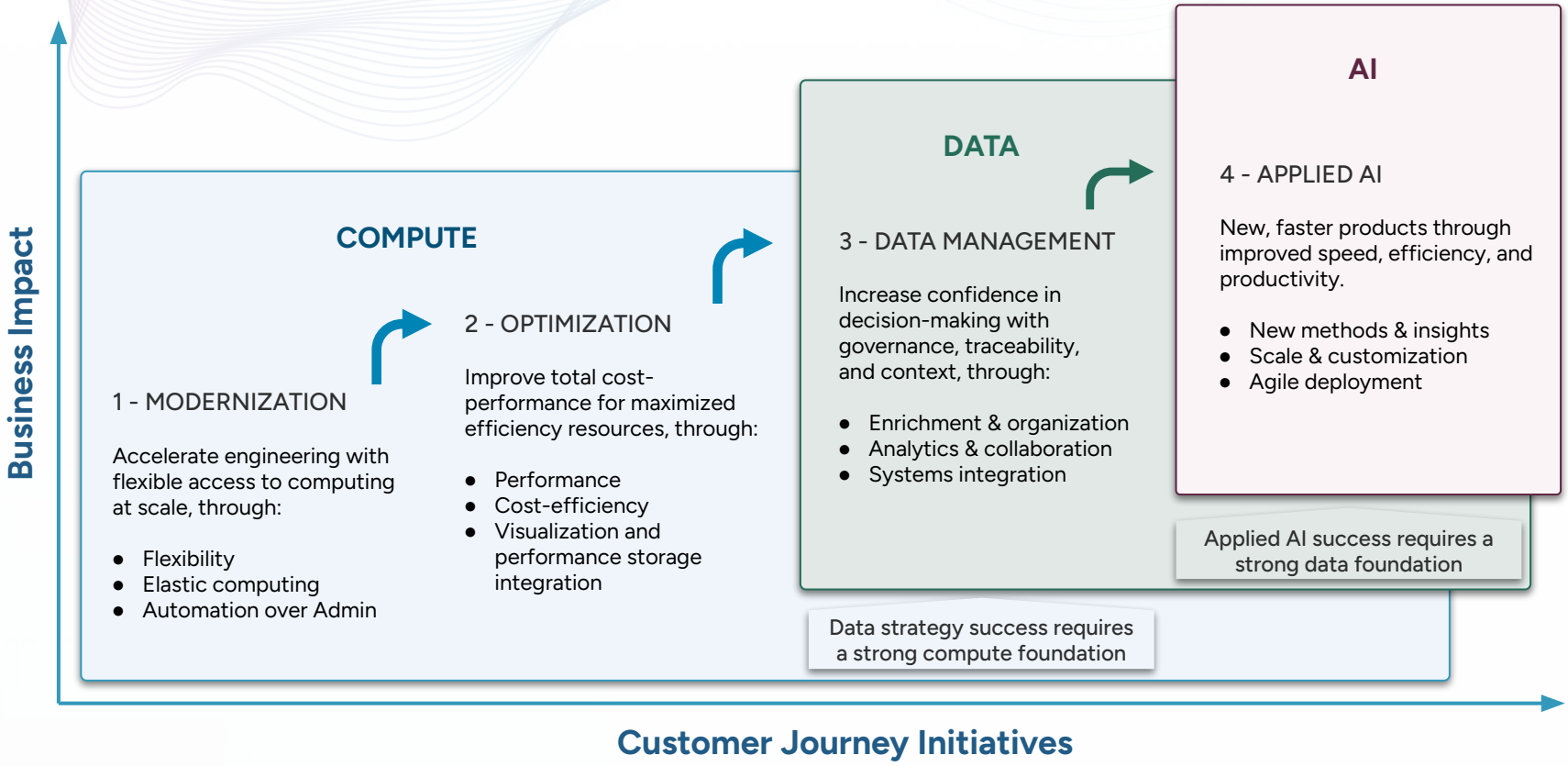
1000X+ Accelerated Results for Design Evaluation

98%+ Accuracy Compared to Traditional CFD Simulation

85% Improvement in Computing Resource Efficiency

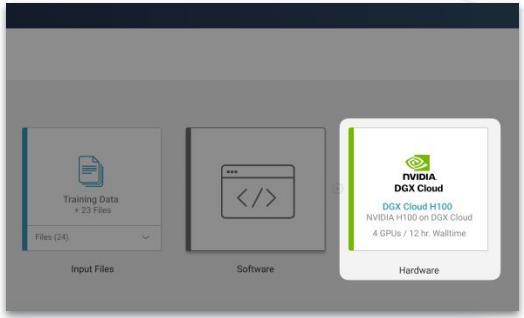
Source

The Journey to Applied AI



Predictive Physics Models: Rescale AI

1 Simulate

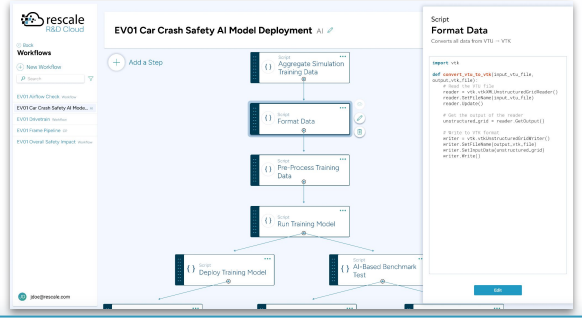


Run Physics Simulations

(Simulation Engineer)

Generate Simulation Data

2 Train



AI Model Development and Deployment

(Methods Developer)

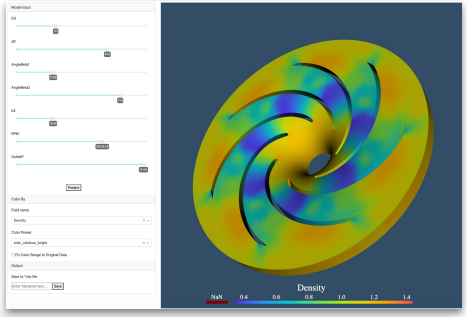
Aggregate Training Data

Pre-Process Training Data

Run Training Model

Deploy Training Model

3 Predict



Run AI Simulations

(Simulation Engineer)

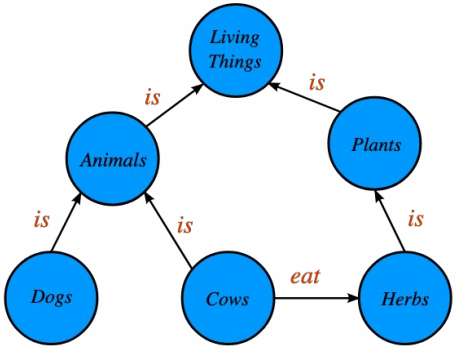
Model Selection

Run Inference

Validation & Tuning

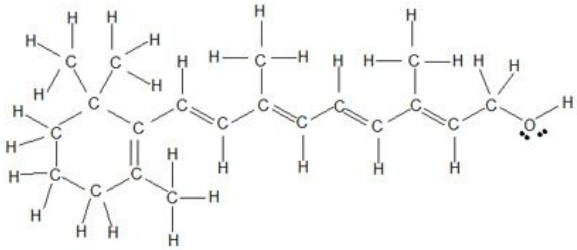
Why Graph Neural Nets?







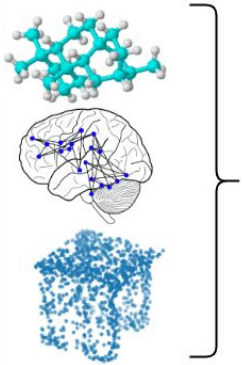
- Vertexes (nodes) and edges, with local properties, or graph level global properties
- Common tasks:
  - Node classification
  - Node clustering
  - Link prediction
  - **Regression**



GNN learning

- The physical system is represented as a graph of nodes and edges via an **adjacency matrix** (to define locality) and **feature matrix** (to define all other properties)

- The GNN learns by iteratively **passing messages** between connected nodes, updating feature representations based on local context. Transformed features are then used to **predict outcomes** such as node-level fields, edge interactions, or global system behavior.



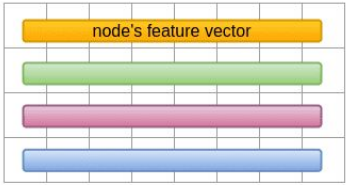
Structure =
Adjacency matrix

$A \in R^{N \times N}$

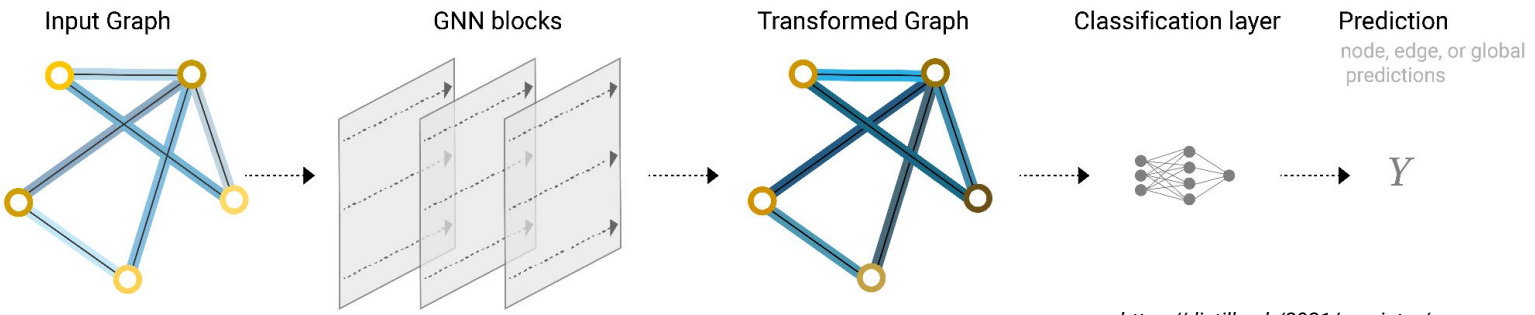
1	0	1	0
0	1	0	0
1	0	1	1
0	0	1	1

Graph Signal
(feature matrix)

$X \in R^{N \times F}$



https://theaisummer.com/graph-convolutional-networks/



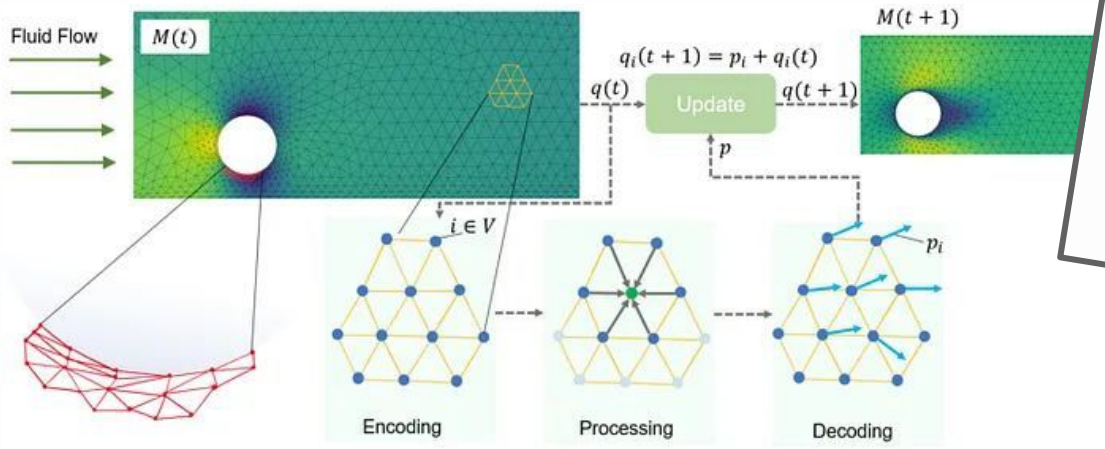
https://distill.pub/2021/gnn-intro/

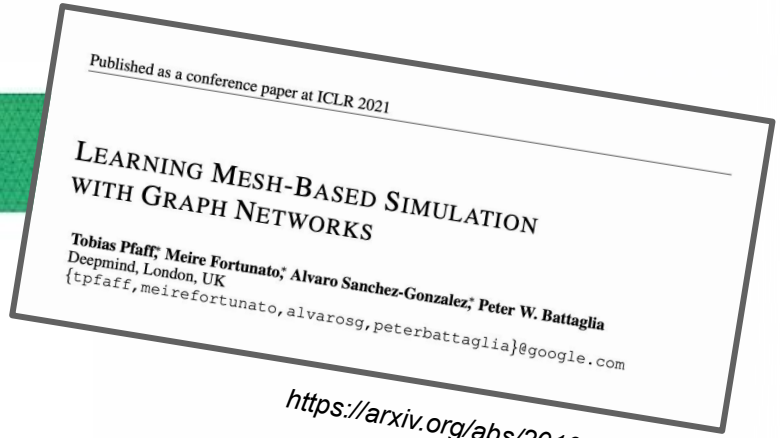
Mesh Graph Nets (MGNs): Theory

- MeshGraphNets use graph neural networks to model mesh-based physical simulations, supporting adaptivity to **capture different resolutions across the mesh**. Represent complex physical phenomena like aerodynamics, structural mechanics, and cloth dynamics.

- The framework uses a three-step **Encode-Process-Decode Architecture**
  - Encoder: Converts the mesh into a graph with features at nodes and edges.
  - Processor: Uses graph neural network layers to simulate interactions over mesh and world-space edges.
  - Decoder: Predicts the next state, updating mesh nodes dynamically to allow for adaptive remeshing.

- Adapts resolution based on the simulation domain: accurate and computationally efficient—running 1-2 orders of magnitude faster than traditional simulators.

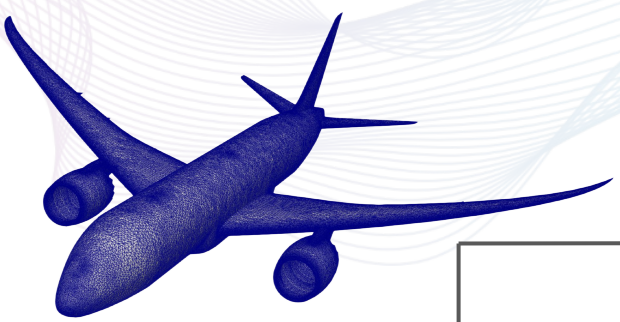



Published as a conference paper at ICLR 2021

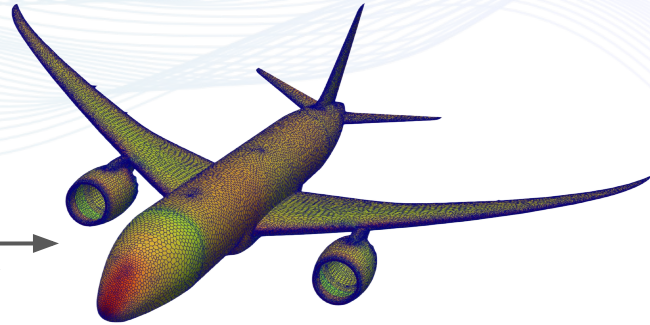
LEARNING MESH-BASED SIMULATION WITH GRAPH NETWORKS

Tobias Pfaff*, Meire Fortunato*, Alvaro Sanchez-Gonzalez*, Peter W. Battaglia
Deepmind, London, UK
{tpfaff,meirefortunato,alvarosg,peterbattaglia}@google.com

https://arxiv.org/abs/2010.03409

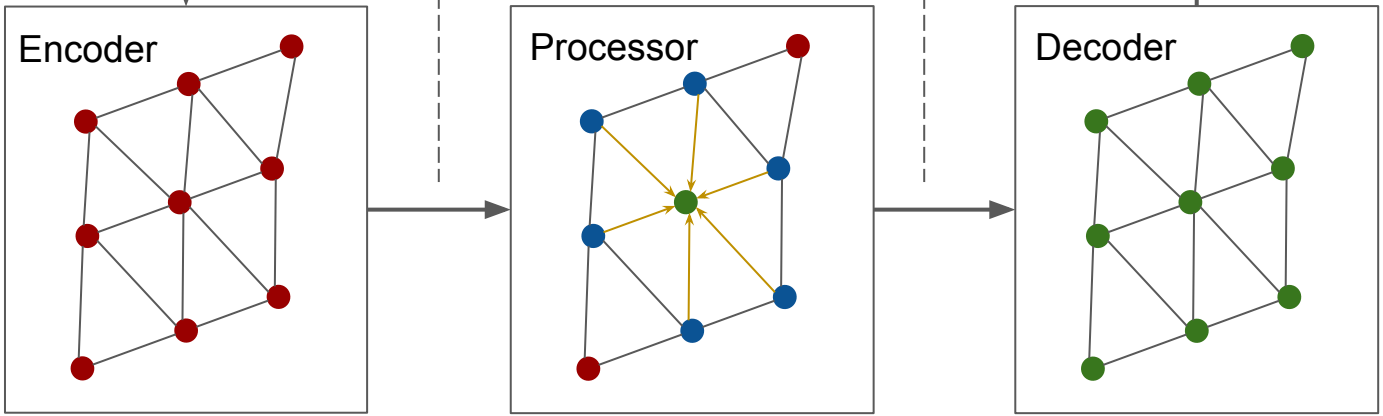


$$y_i, y_j = f(e_{ij}, x_i, x_j)$$



Sample n

Latent space

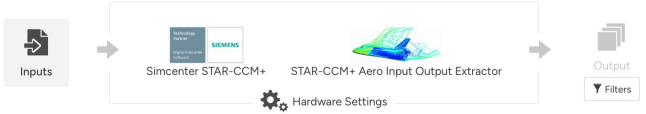


Step 1 Encoder: encode features as node and edge embeddings

Step 2 Processor: Update node and edge embeddings using message passing

Step 3 Decoder: Generate updated feature from new node embeddings

787 Surrogate Model: study workflow



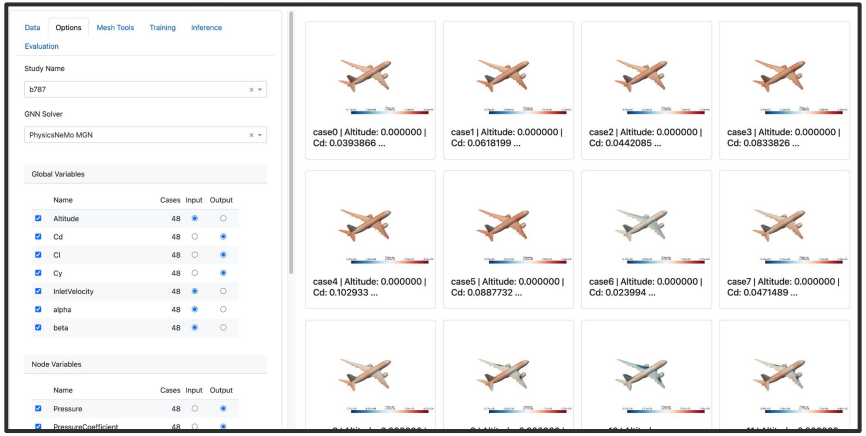
- **Simulate**
  - Design space and simulation profile:
    - Steady state CFD simulations on B787 airplane model
    - Input levers: **altitude, alpha** (angle of attack)**, beta** (sideslip angle)
    - 48 cases: combinations of input configurations
    - CFD solver: STAR-CCM+: simulations results on Cd, Cl, Cy as well as local field variables: Pressure, SkinFriction coefficient, Wall Shear stress.
  - Automation to extract all input and output files and post them as metadata associated with each simulation case
- **Train**
  - Staging of high performance storage device
  - Data preparation: Automatically gather simulation-generated result files from all 48 cases of DOE study, convert to ML-ready file formats (.vtp files), filter blocks, filter any outliers among cases, filter constant inputs (e.g. inlet velocity)
  - Mesh downsampling (if necessary)
  - Hyperparameter selection
  - 80-10-10 split (training, validation, testing)
  - Submit training jobs on GPU architectures (NVIDIA L40)
- **Predict**
  - Prediction of global targets (Cd, Cl, Cy), and node-level variables (Pressure, Skin Friction, Wall Shear Stress)
  - Vary input levers for near-realtime inference
  - Publish model as standalone software tile

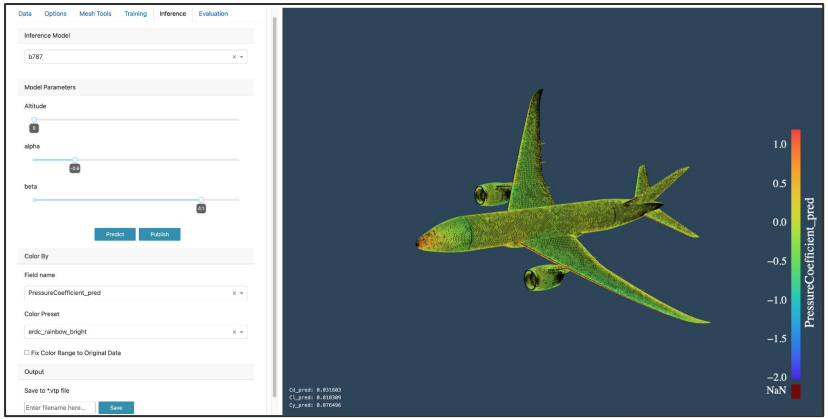
787 study: MeshGraphNet Surrogate Model

- Interactive in-platform setup:
  - Block filtering, virtual edges, decimation
  - Training parameters: train on: Pressure, Wall Shear Stress, Friction Coefficient
  - Hyperparameter configuration

- Launch training jobs across menu of GPUs
- Monitor model convergence: Training loss, validation error

- **See: live in platform**

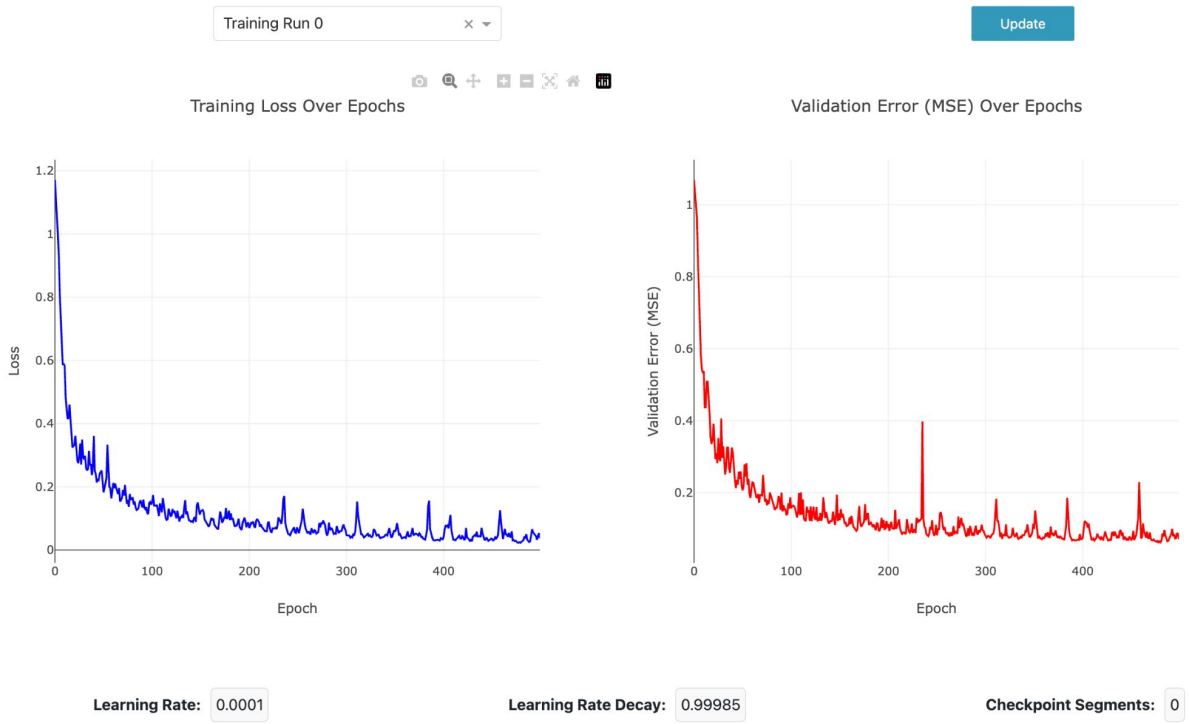




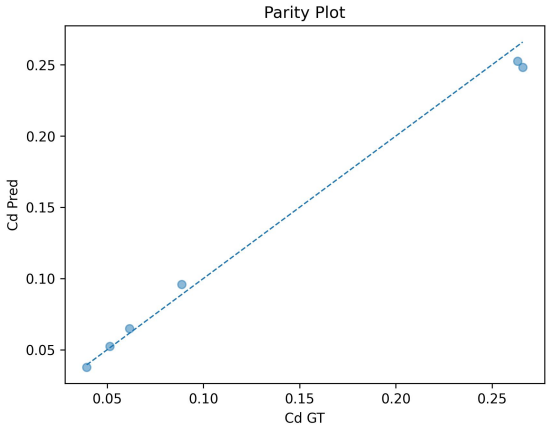


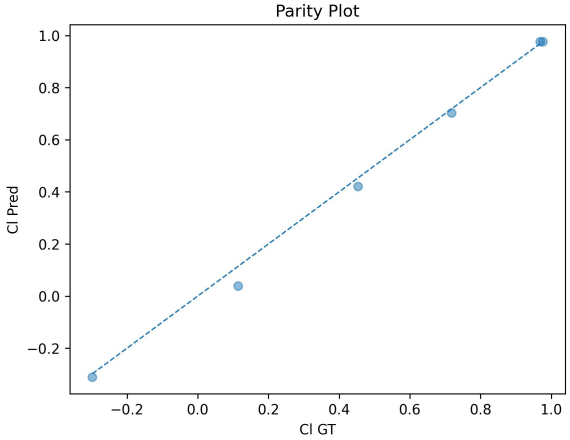
787 study: Baseline training

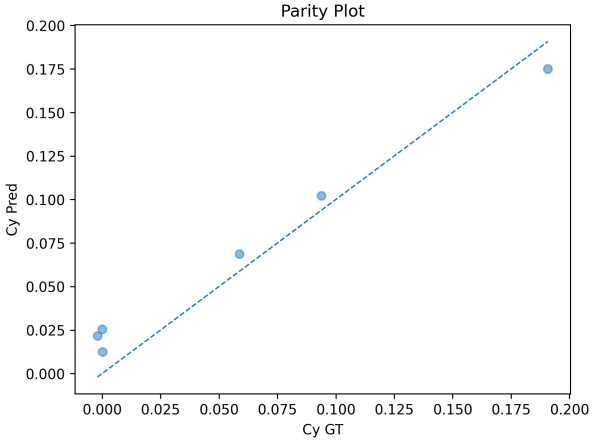
- Hyperparemeters
  - 500 epochs
  - Learning rate: 0.0001
  - Hidden dim variables: 256
- Training loss and MSE error convergence
- Inference outcomes on:
  - Pressure, Shear Stress, Friction Coefficient
- Publishing model



787 study: Summary of global and node level accuracy







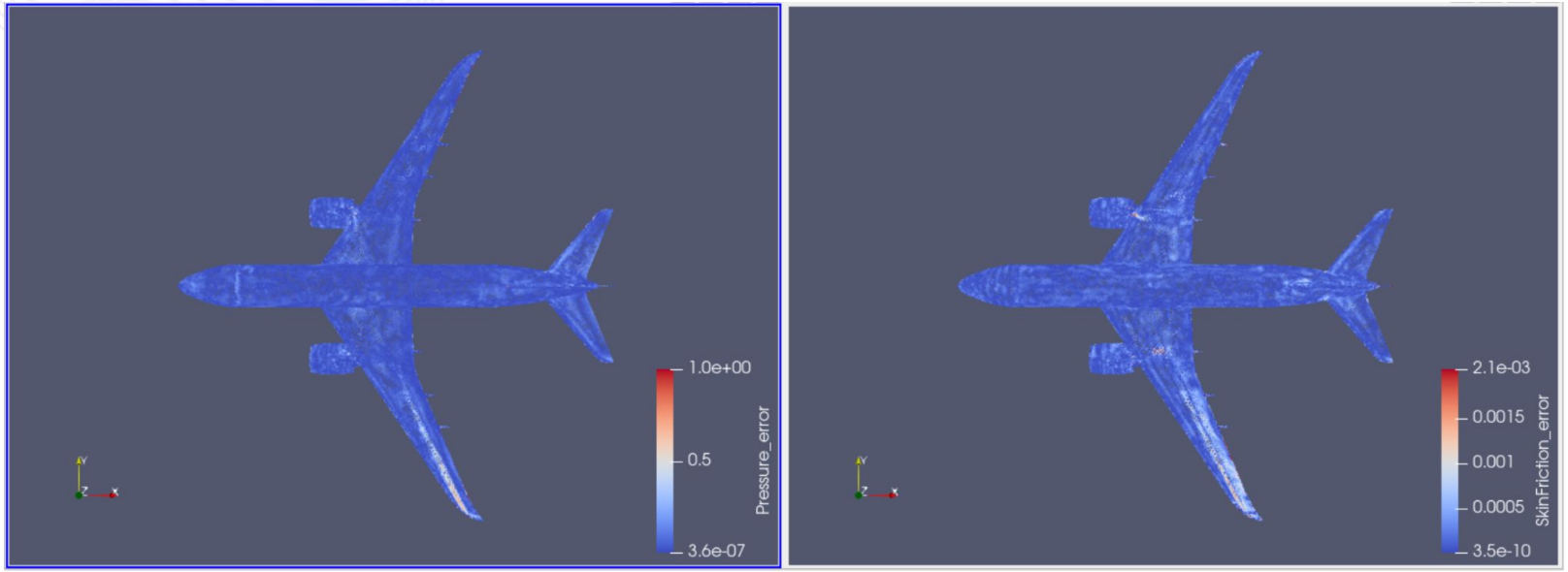
Cd:
MAE=6.9050e-03
RMSE=9.0385e-03
R2=0.991

Cl:
MAE=2.3972e-02
RMSE=3.4748e-02
R2=0.994

Cy:
MAE=1.5923e-02
RMSE=1.7223e-02
R2=0.939

Avg. global error across Cd, Cl, Cy: **9.3%**
Avg. Node level error across pressure, skin friction, wall shear stress: **16.8%**

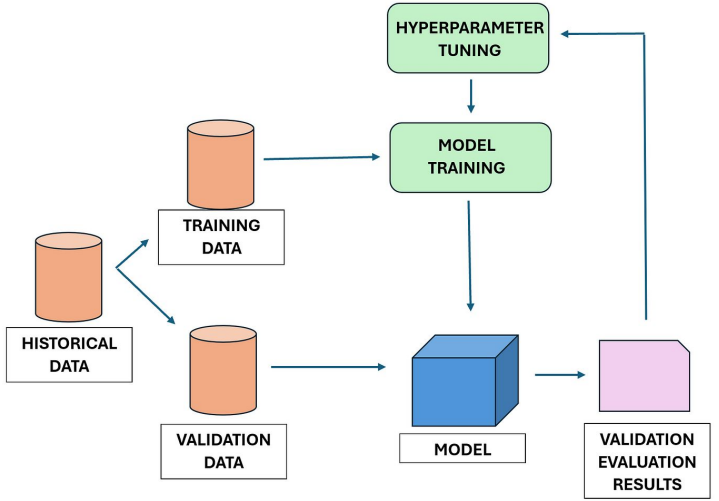
787 study: Physical significance of node-level local error



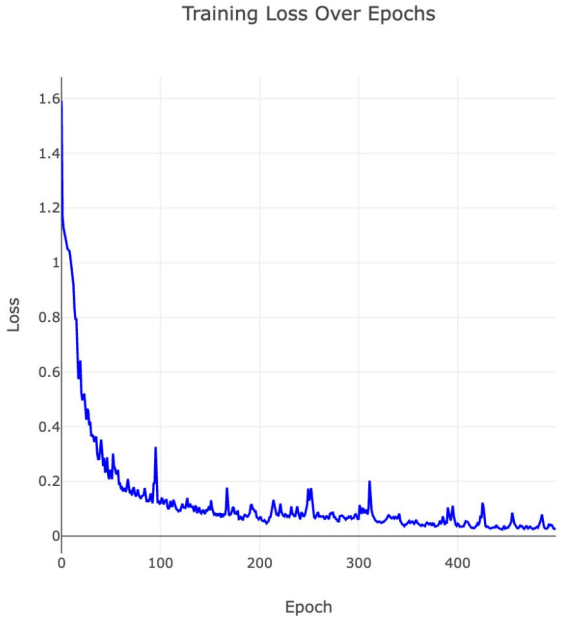
- Error of surrogate model vs simulation ground truth for test case with highest relative error
- Output files write global variables and all local field functions

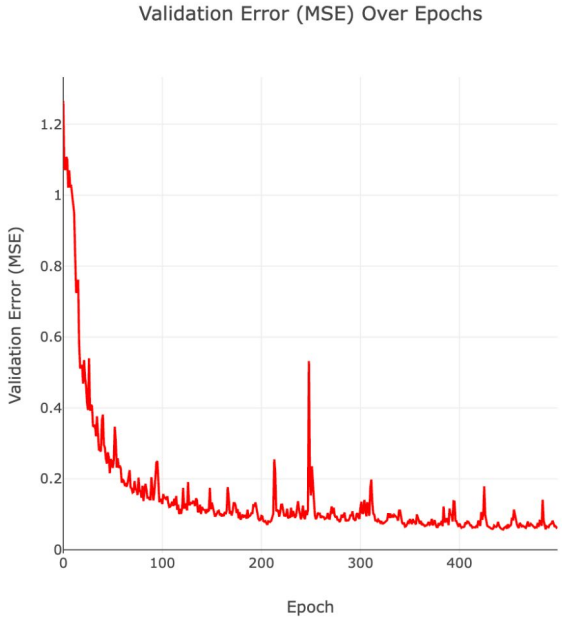
737 study: MGN Results: Hyperparameter tuning

- Hyperparameter tuning, variables of interest:
  - **Learning rate: c**ontrols size of each update step is when adjusting the model's weights to minimize the loss during training.
    - Baseline: **0.0001**; → Test: **0.001**
    - Upside: faster model convergence
    - Downside: overshoot learning, training divergence
  - **Hidden edge/node encoder/decoder dimensions**
    - Baseline: hidden_dim_node_encoder: **256**, hidden_dim_edge_encoder: **256**, hidden_dim_node_decoder: **256**.
    - Upside: Lower memory usage and faster training
    - Downside: underfitting complex physics
- **Within Rescale AI,** hyperparameters tuned manually for new training jobs, or scripted to launch a hyperparameter DOE



787 study: lr = 0.001



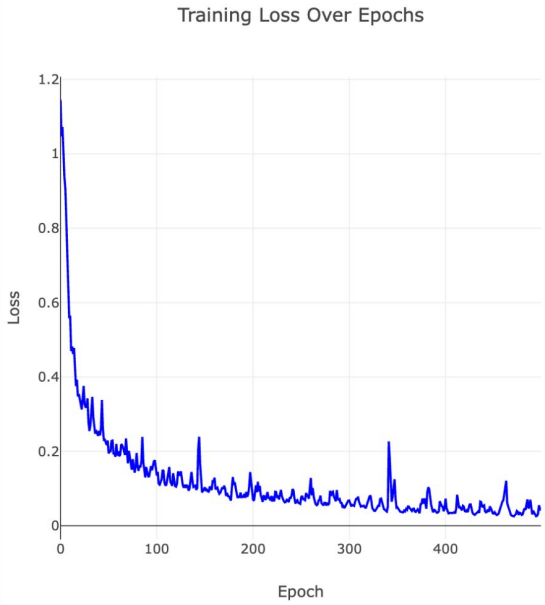


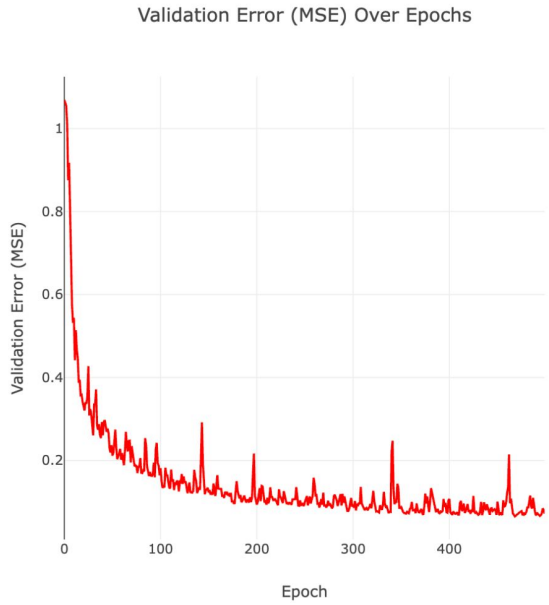
Learning Rate: 0.001 **Learning Rate Decay:** 0.99985 **Checkpoint Segments:** 0

Hyperparameters

- 500 epochs
- Learning rate: 0.001
- Learning Rate Decay 0.99985
- Hidden dim params: 256
- Batch size: 1

787 study: hidden dims: 128



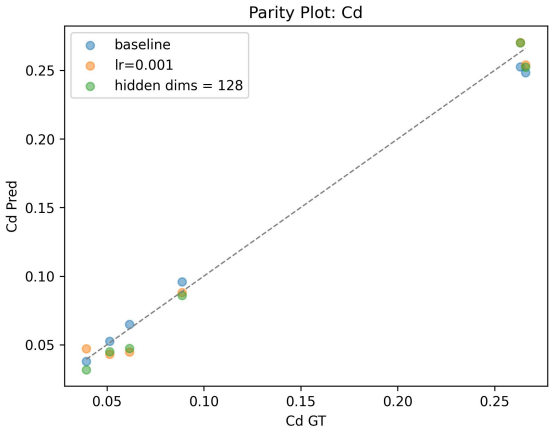


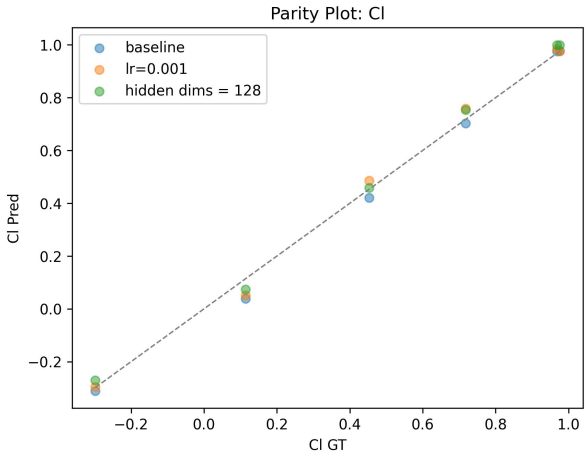
Learning Rate: 0.0001 **Learning Rate Decay:** 0.99985 **Checkpoint Segments:** 0

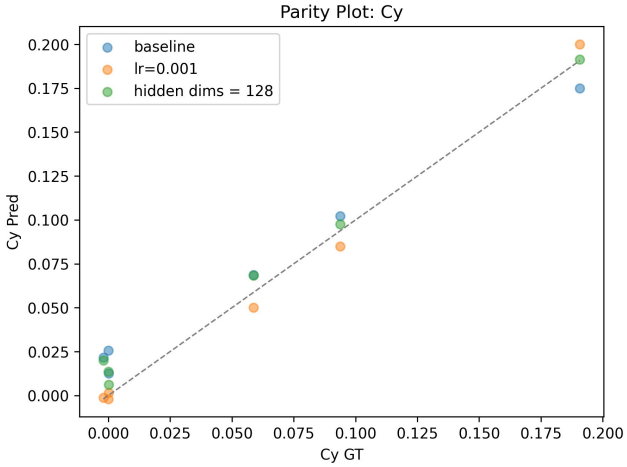
Hyperparameters

- 500 epochs
- Learning rate: 0.0001
- Learning Rate Decay 0.99985
- Hidden dim params: 128
- Batch size: 1

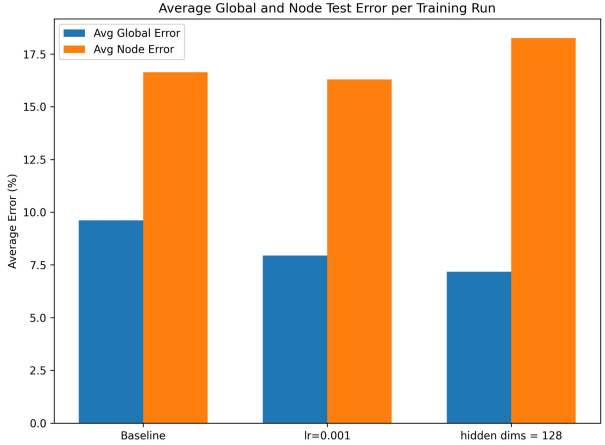
787 study: Global accuracy metrics across hyperparameter







	Cd	Cl	Cy
Baseline	0.991	0.994	0.939
lr=0.001	0.989	0.995	0.992
Hidden dims = 128	0.990	0.996	0.972



Beyond GNN methods: Decomposable Multi-scale Iterative Neural Operator (DoMINO)

- Limitations of MeshGraphNet (GNN) based approaches
  - Require explicit mesh connectivity; retraining needed for new mesh topologies or resolutions.
  - Message-passing for long-range effects is computationally expensive.
  - Downsampling meshes to reduce cost often sacrifices accuracy, especially near critical features.

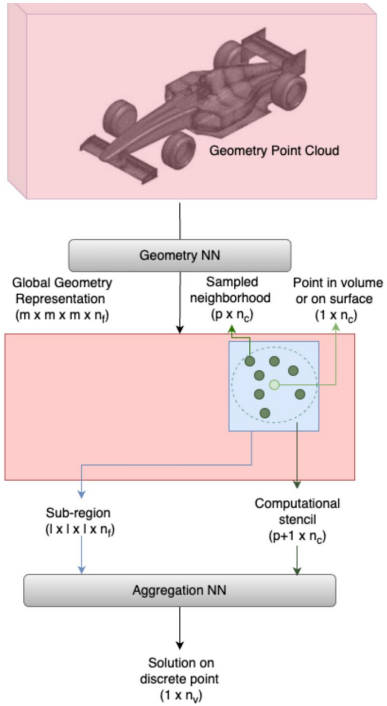
- DoMINO is a **neural operator-based framework** within NVIDIA PhysicsNeMo to model complex fluid dynamics problems, including high-fidelity external aerodynamics simulations. It generalizes well across varying flow conditions and geometries by learning **mesh-independent operators** directly in the frequency domain.

  - **Global encoding**: Learn a multi-scale, iterative global geometry representation from **point clouds**, enriched with signed distance fields and positional encodings to capture both short- and long-range dependencies.

  - **Local extraction**: For each prediction point, sample a sub-region and extract local geometry features from the global encoding using dynamic point convolution kernels.

  - **Stencil aggregation**: Dynamically construct a computational stencil by sampling neighboring points, then aggregate local geometry and stencil information to predict solution variables at each discrete point.

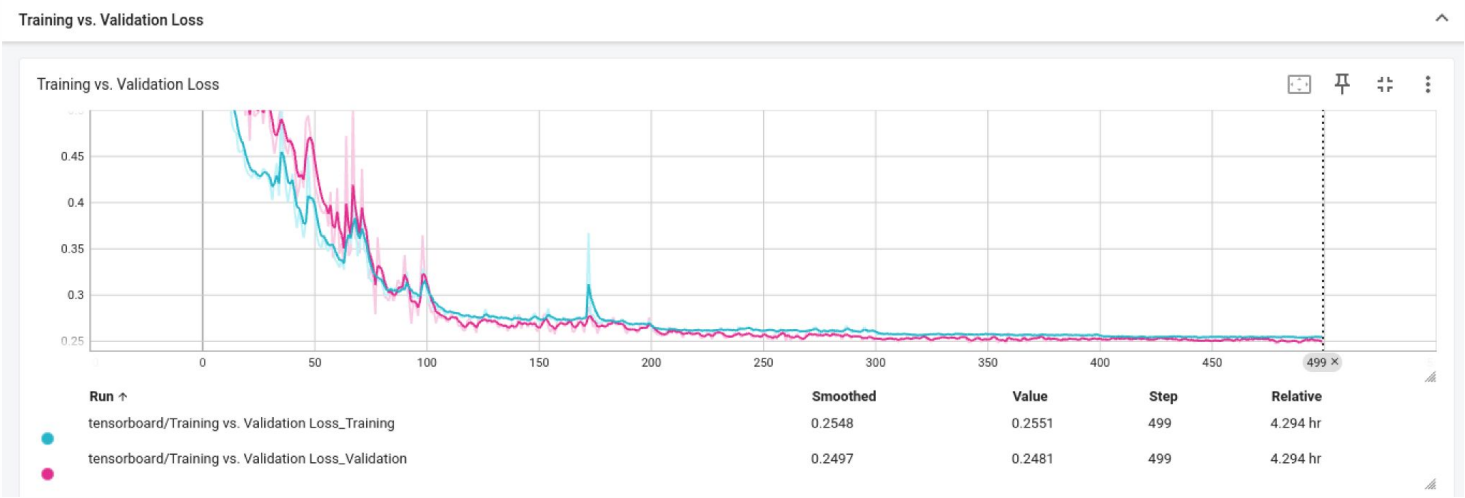
- DoMINO supports **mesh-free inference** and achieves substantial speed-ups (1–3 orders of magnitude) over traditional CFD solvers while preserving physical realism.



https://arxiv.org/pdf/2501.13350

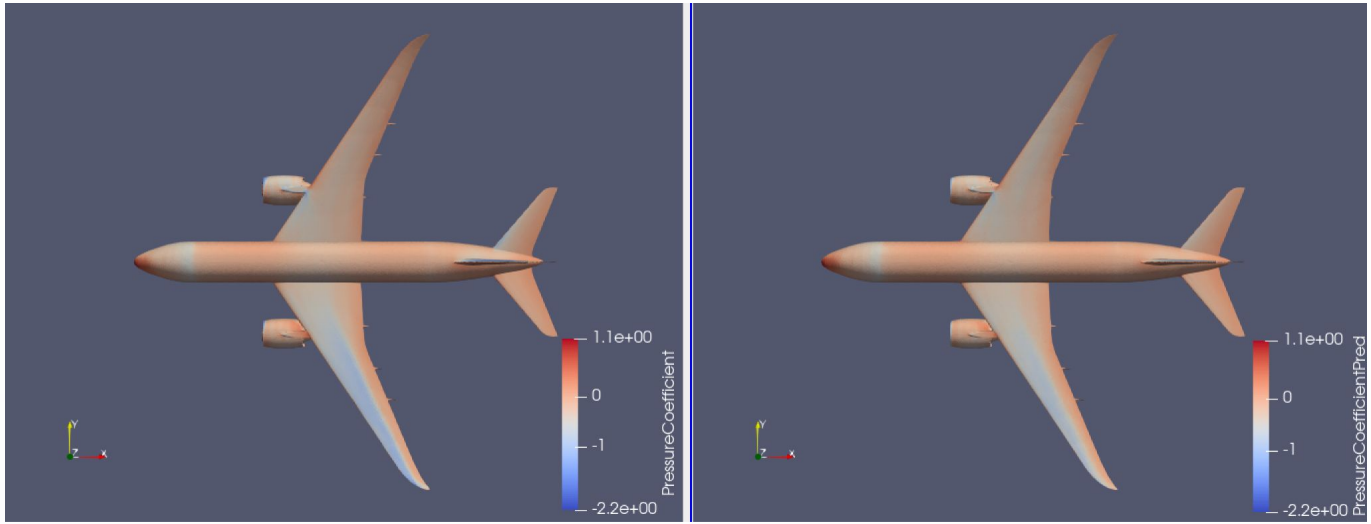
787 study: Preliminary DoMINO surrogate model results

- Invoking DoMINO architecture within NVIDIA Physics Nemo
- Approach: Surface model for DoMINO architecture (no training on volume yet)
- Key hyperparameters
  - **Learning rate**: 0.001, **geom_points_sample**: 300,000
- Error convergence:
  - Achieving loss convergence: no under/overfit



787 study: Preliminary DoMINO surrogate model results

- Invoking DoMINO architecture within NVIDIA Physics Nemo
- Approach: Surface model for DoMINO architecture (no training on volume yet)
- Key hyperparameters
  - **Learning rate**: 0.001, **geom_points_sample**: 300,000
- Error convergence:
  - Achieving loss convergence: no under/overfit



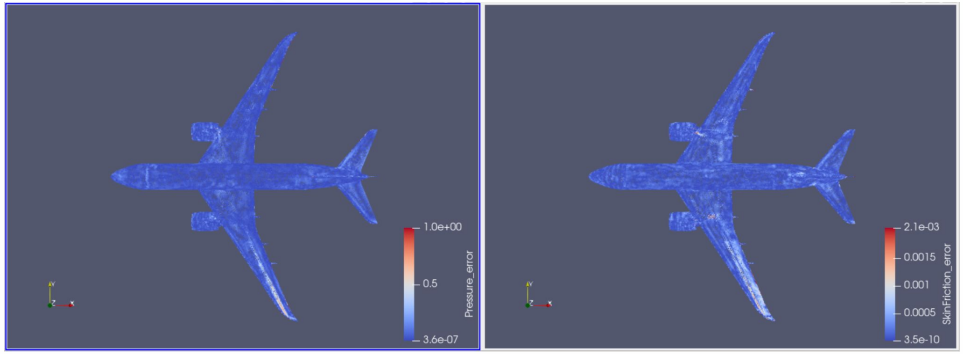
787 study: MGN vs DoMINO methods

- Similar convergence: ~500 epochs
- Early indications that DoMINO model training (4 hr @ 300,000 cloud points) is faster than MGN training (13 hours @ 600,000 surface mesh size)

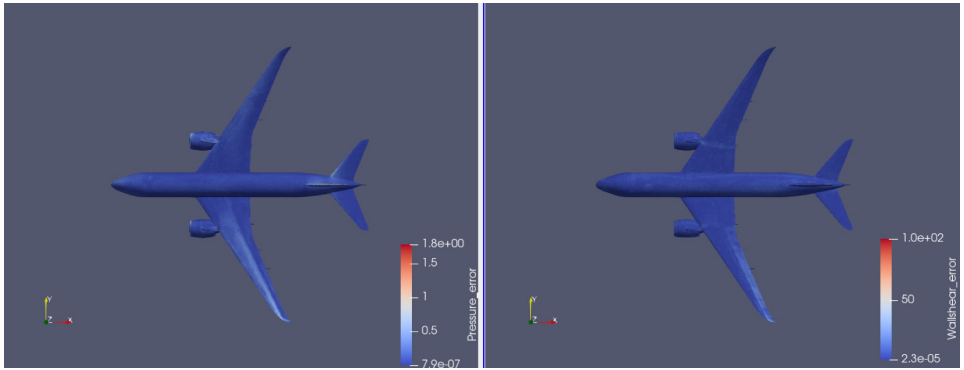
- DoMINO currently with higher RMSE (DoMINO: 0.26 vs MGN: 0.1). Average node error = ~35% (vs MGN ~17%)

- Local error comparison:
  - Similar local error zone on the left wing surface
  - More local error near engines on MGN

- DoMINO is more flexible to newer geometries and inference outside the surface mesh



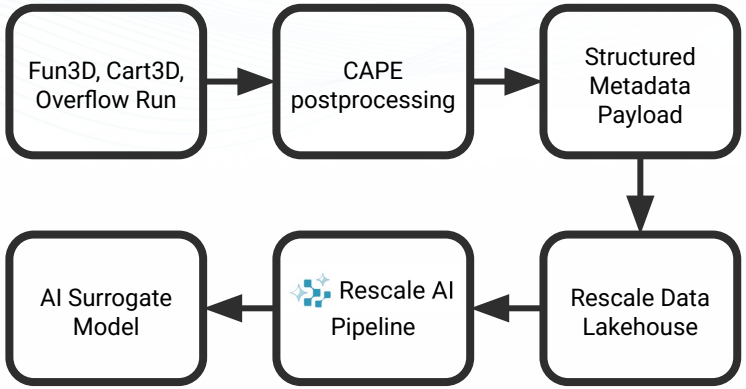
MeshGraphNet surrogate model node errors



DoMINO surrogate model node errors

Rescale AI in NASA context

- **CAPE Invocation**: Integrated alongside FUN3D, OVERFLOW or Cart3D and triggered automatically post-run via scheduler hooks.
- **Automated Metadata Extraction:** Parses solver logs and namelists to pull inputs (Mach, angle-of-attack, geometry IDs, solver settings) and outputs (lift, drag, moments, residual histories, Cp, local Mach).
- **Standardized Metadata Payload**: Formats each case's inputs and outputs into a structured record
- **Lakehouse Ingestion:** Records are sent via API into Rescale Data Lakehouse, preserving full provenance (solver version, timestamps) alongside metadata payloads.
- **Non-Disruptive Deployment:** Existing CFD workflows remain unchanged; CAPE's post-process and lakehouse agent execute transparently within each job.
- **Rapid Training-Set Assembly**: Rapidly assemble training datasets for Rescale AI without manual CSV wrangling




```python
import numpy as np

from cape.gruvoc.umesh import Umesh

def main(mesh_file, mapbc_file, flow_file, output_plt="output.plt"):
    # 1. Load mesh + BC map
    mesh = Umesh(mesh_file, mapbc=mapbc_file)

    # 2. Read FUN3D flow solution
    mesh.read_fun3d_flow(flow_file)

    # 3. Print mesh topology info
    print(f"Nodes:       {mesh.nodes.shape[0]}")
    print(f"Triangles:   {mesh.tris.shape[0]}")
    print(f"Quads:       {mesh.quads.shape[0]}")
    print(f"Tets:        {mesh.tets.shape[0]}")
    print(f"Pyramids:    {mesh.pyrs.shape[0]}")
    print(f"Prisms:      {mesh.pris.shape[0]}")
    print(f"Hexes:       {mesh.hexs.shape[0]}")

    # 4. List flow-state variables and compute average Mach if present
    print("Flow variables:", mesh.qvars)
    if "Mach" in mesh.qvars:
        m_idx = mesh.qvars.index("Mach")
        print(f"Average Mach: {np.mean(mesh.q[:, m_idx]):.4f}")

    # 5. Write Tecplot file for visualization/inspection
    mesh.write_tecplot(output_plt)
    print(f"Wrote Tecplot file: {output_plt}")
```

Conclusions

- **Surrogate AI models based on modern graph neural net and neural operator methods can transform innovation in aerodynamic engineering**
    - MeshGraphNets (MGN) vs. Neural Operator method (DoMINO):
        - MGN: Lower local errors on structured meshes; best for mesh-consistent workflows. Global errors < node-level errors (e.g., 10% vs. 16%).
        - DoMINO: Scalable, mesh-free, flexible for new geometries; current RMSE higher than MGNs, but model training is faster
    - Next steps: Increasing size of ground truth simulation DOE, hyperparameter tuning, introducing geometry changes to surrogate models
- **Technical design considerations:**
    - Simulation DOE design, standard automations for posting input/outputs, metadata for simulation provenance and scripting, ML training setup: (block filtering, hyperparemeters, AI solver), error/convergence monitoring
    - Modular machine learning workflows supporting both new physics solvers and new machine learning architectures codes



rescale

AI
DATA
COMPUTE

Innovation
Elevated

Viktor Rozsa
Sr. Solutions Architect, Rescale

vrozsa@rescale.com
@rescale
#rescale

Appendix

787 Surrogate Model study workflow

Simulation DOE

- Define design space, geometry
- Output files write global variables and all local field functions

Preprocessing

- Collect inputs and outputs: mesh, findings, metadata
- Stage results on persistent high performance storage device
- Save as filetypes (.vtp) file ready for training

ML Training

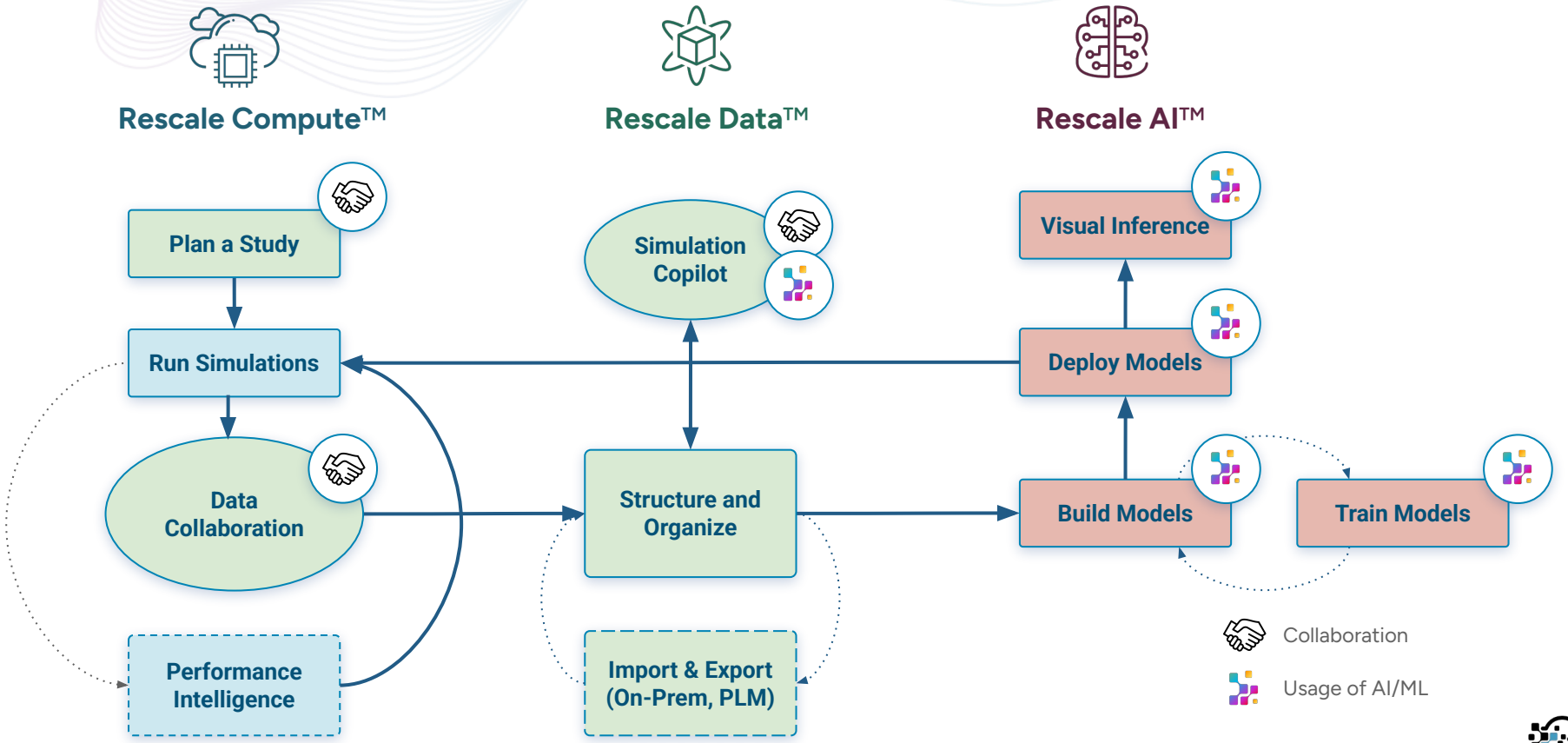
- Configure model hyperparameters
- Submit training job on GPUs
- Monitor/evaluate model error
- Save model weights

Inference

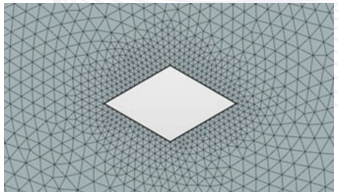
- Realtime prediction of global and local properties from selectable inputs
- Visualize results and errors of surrogate model
- Publish model for future use

Unified control plane: Rescale Compute, Rescale Data, Rescale AI

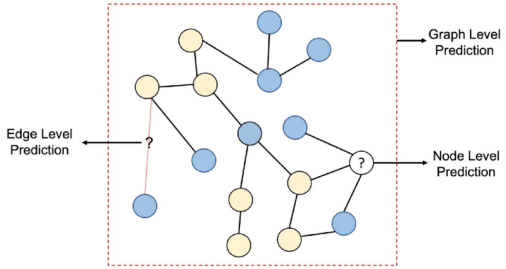
Rescale Platform | A Continuous Workflow



[option 2] MGN Methodology



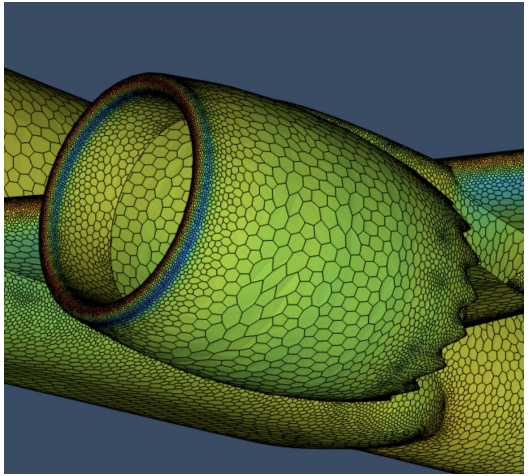
Mesh



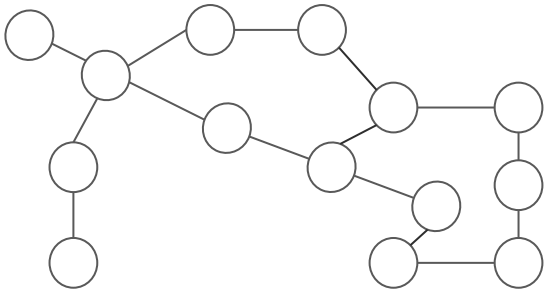
Graph

We use GNN to learn and pass messages along graph edges.

Our current GNN model restricts message passing to existing edges, so any disconnected parts cannot exchange information.



Convert mesh to graph by mesh connectivity and edges



Turnkey HPC-as-a-Service for Any Size Organization

1,200+ Leading Engineering & Scientific Applications

VASP · QUANTUMESPRESSO · Schrödinger · DASSAULT SYSTEMES · MATLAB · Ansys · PYTORCH

- Software and Licensing
- Tuned Hardware Configurations
- Flexible Job Submission
- Data Sharing & Collaboration
- Workflows Automations

Full-stack **Engineering & Scientific** Computing



Unified **Visibility & Control for IT** Management

- End-to-End Security & Compliance
- Total Spend and Consumption
- Multi-Team & Access Controls
- Data Platform Integration
- Multi-Cloud, Region, & Cost-Model

100+ High Performance Computing Architectures

aws · Google Cloud · Microsoft Azure · ORACLE CLOUD Infrastructure · CoreWeave · AMD · arm · intel · NVIDIA

Specialized Computing at Cloud Scale Delivers Faster Results

Job 1
FEA - high clock speed requirement

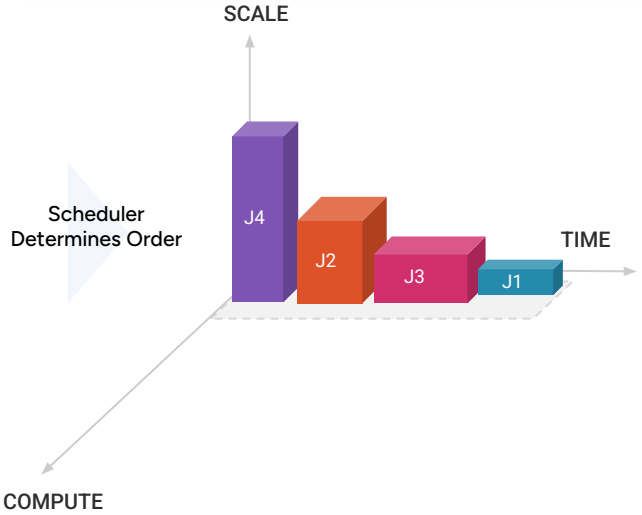
Job 2
CFD - high interconnect requirement

Job 3
Multi-physics - large memory requirement

Job 4
EDA - Highly Parallelized Large core count requirement

On-Premises & DIY Cloud

Queued Job Submission

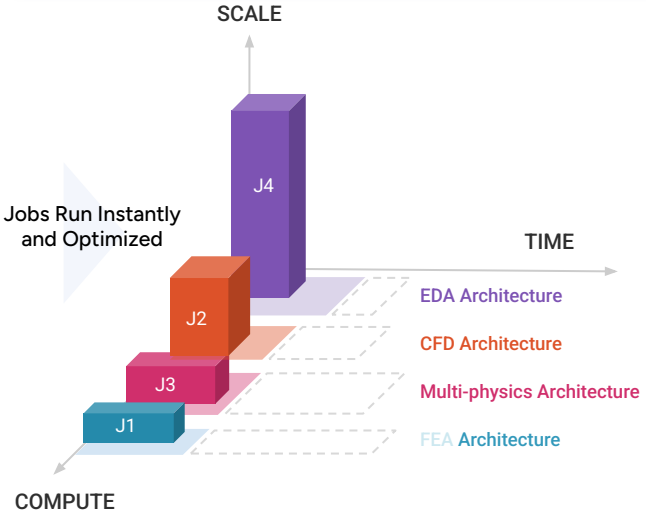


Fixed Infrastructure = Delayed Time-to-Answer
Limited by general purpose architectures and scheduler bottlenecks

VS

HPC Built for the Cloud

Concurrent & Optimized Job Execution



Flexible Infrastructure = Accelerated Time-to-Answer
Specialized compute for each workload