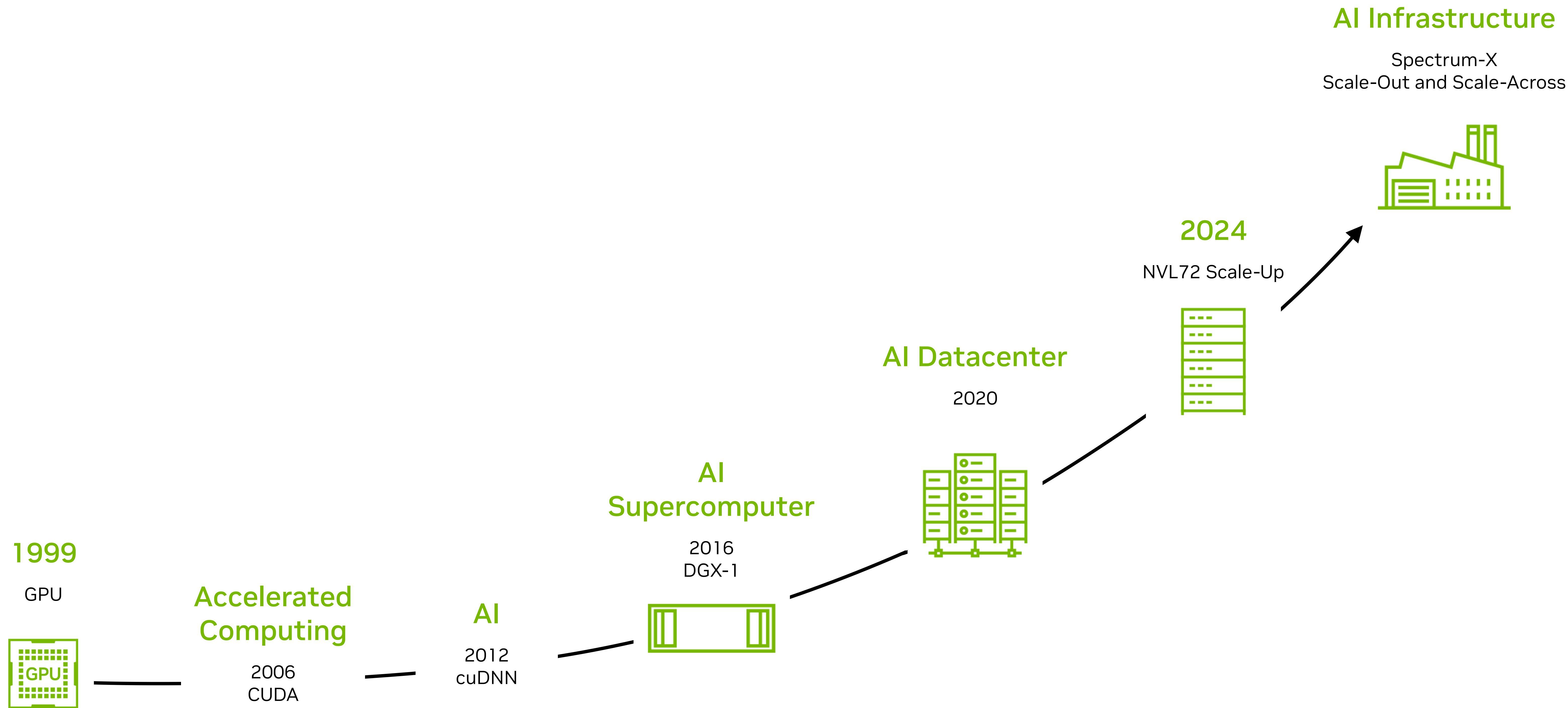


# AI Seismic at Scale: Frameworks, Mixed Precision, Foundation Models, and Physics-ML

João Paulo Navarro - Sr. Solutions Architect

[jnavarro@nvidia.com](mailto:jnavarro@nvidia.com)

# NVIDIA's Evolution From Chips to an AI Infrastructure Company



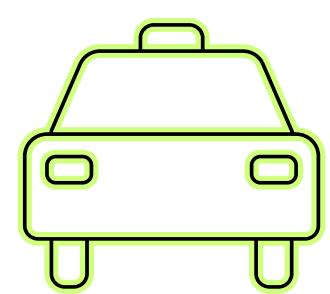
# Leading the AI Era



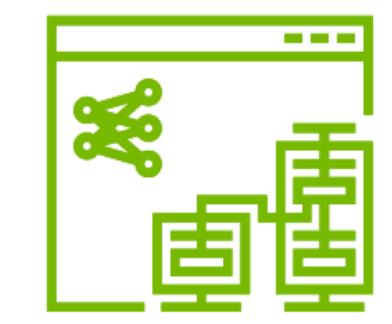
Chatbots



Digital Biology



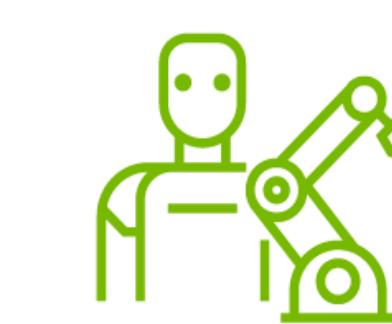
Robotaxi



AI Enterprise Agents



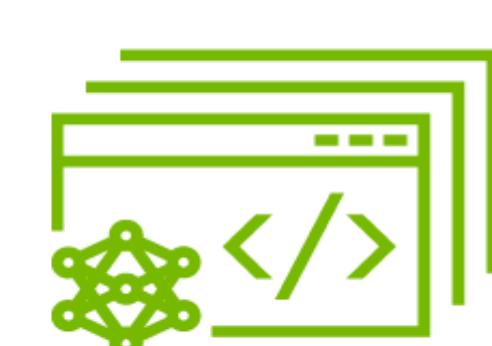
Science



Robotics



Manufacturing



AI Coder

Applications

Models

Infrastructure

Chips

Energy

# Data Preparation

# RAPIDS - cuDF

A GPU DataFrame library in Python with a pandas-like API built into the PyData ecosystem

## Pandas-like API on the GPU

```
pandas
```

```
>>> import pandas as pd
>>> df = pd.read_csv("filepath")
>>> df.groupby("col").mean()
>>> df.rolling(window=3).sum()
```

CPU

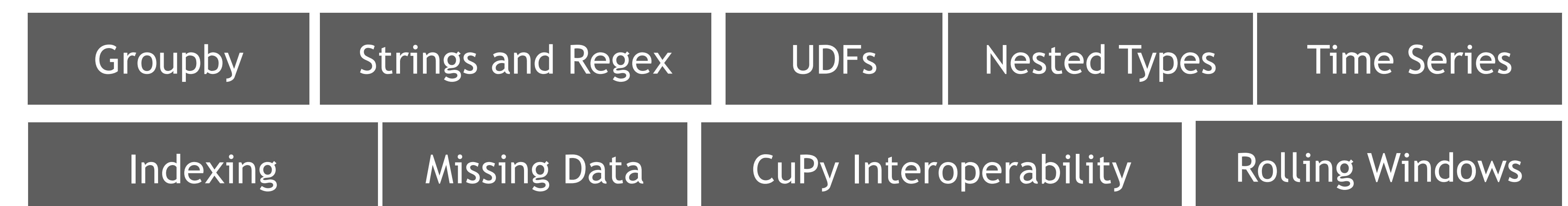
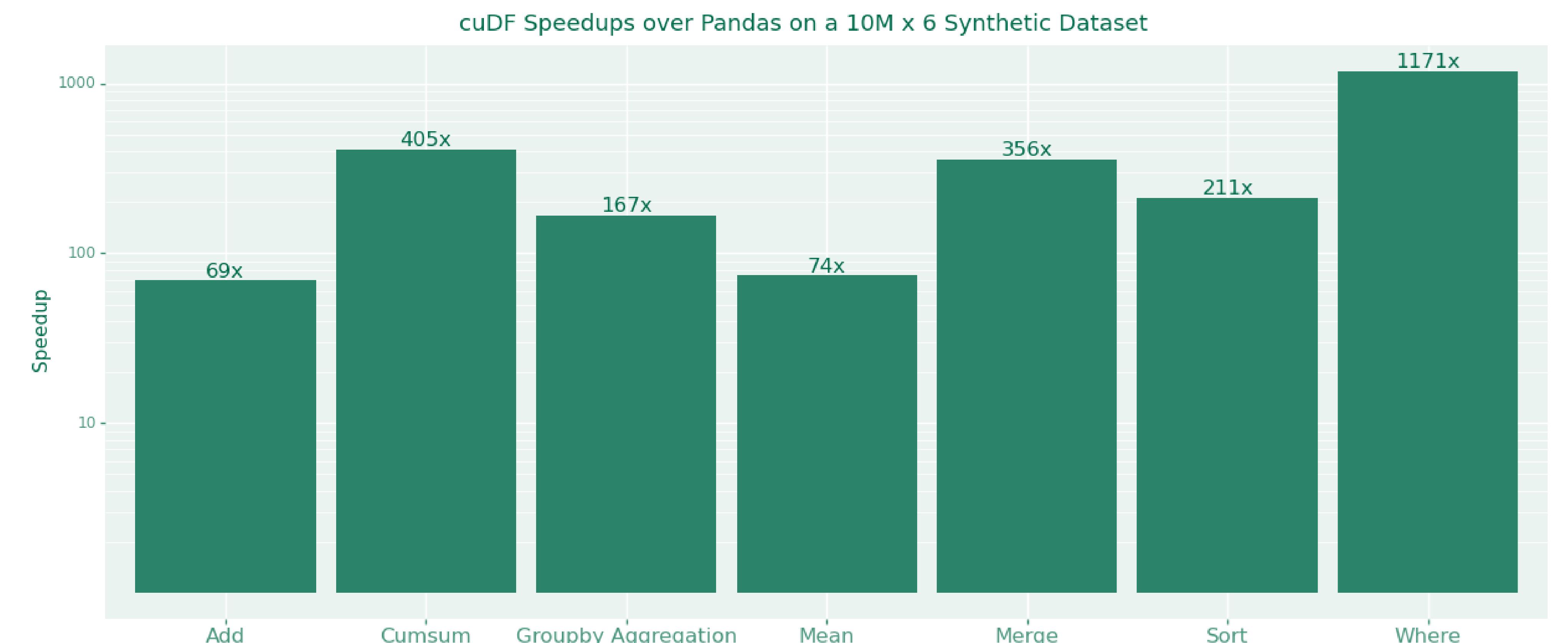
```
cuDF
```

```
>>> import cudf
>>> df = cudf.read_csv("filepath")
>>> df.groupby("col").mean()
>>> df.rolling(window=3).sum()
```

GPU

Average Speed-Ups: 10-100x

## Best-in-Class Performance ([Benchmark](#))



[10 Minutes to cuDF](#)

NVIDIA A100 vs. AMD EPYC 7642 48-Core Processor  
cuDF Python vs. Pandas

# CuPyNumeric and Legate Ecosystem

Distributed runtime framework and libraries



NumPy



SciPy



XGBoost



cuPyNumeric

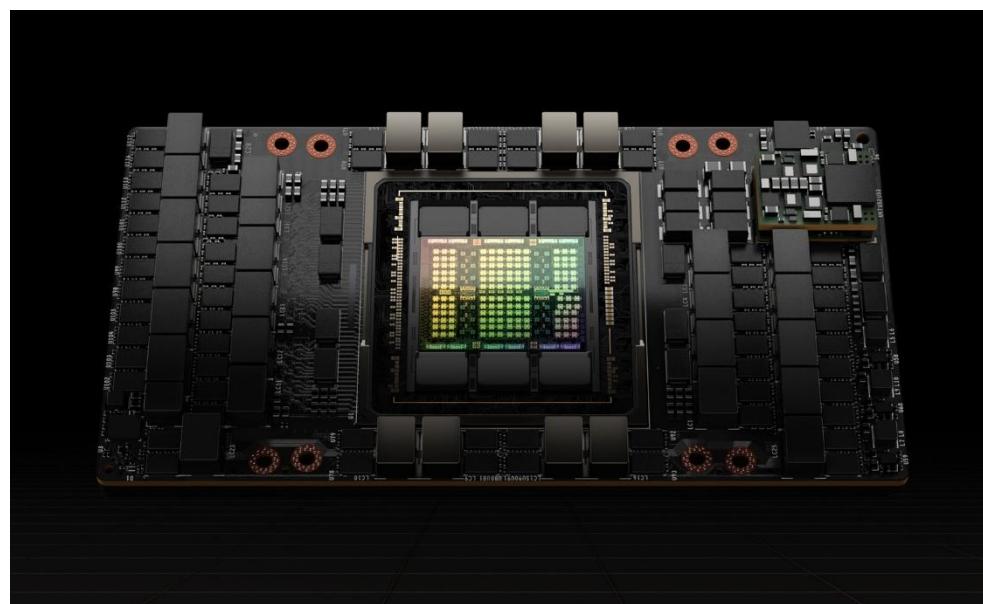
Legate Boost

Legate Dataframe

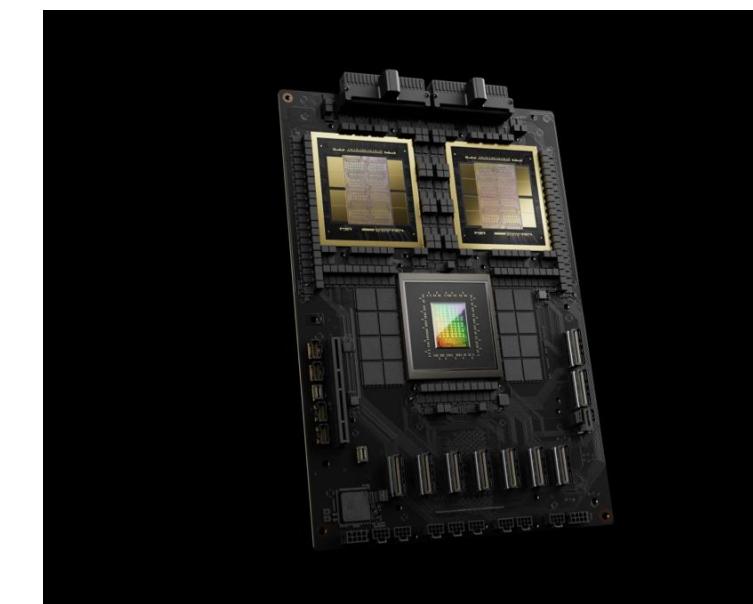
**Legate:** Productivity and Composability Layer

**Legion:** Implicit Parallelism Layer

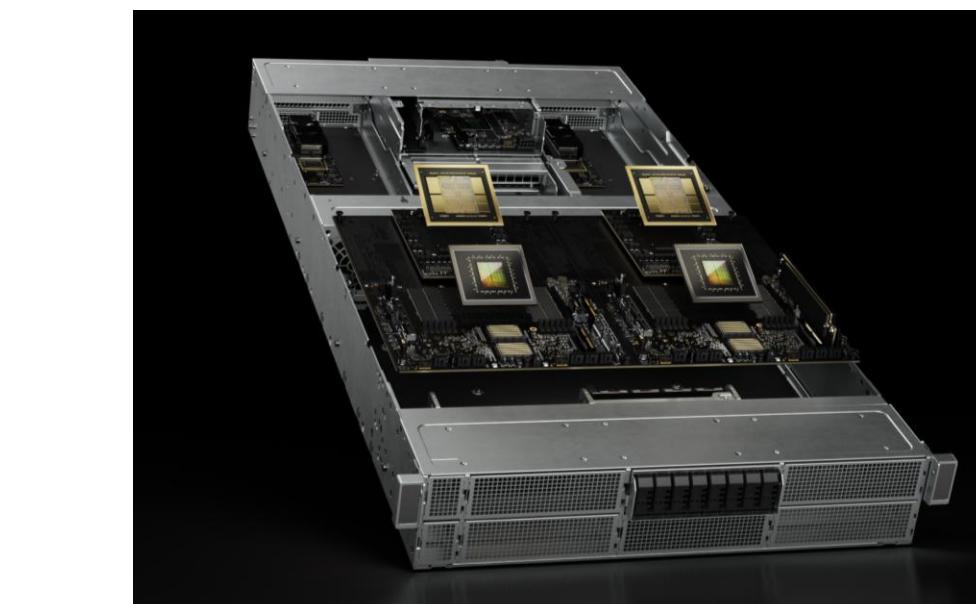
**Realm** (Runtime for Scalable and Portable Execution) Today



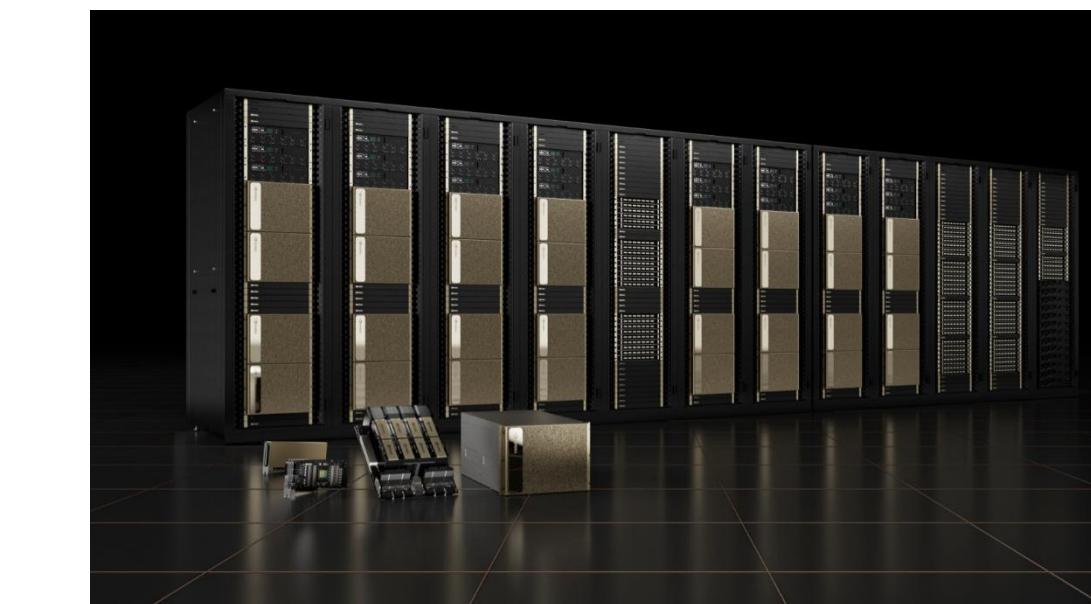
Single GPU



Mixed CPU/GPU



Single-Node Multi-GPU



Multi-Node Multi-GPU  
Cloud & Supercomputer

# Scaling NumPy Code with cuPyNumeric

```
import cupynumeric as np

m, n, k = 1000000, 1000000, 1000000 # 1Mx1M
A = np.random.rand(m,k).astype(np.float32)
B = np.random.rand(k,n).astype(np.float32)

res = A@B
```

Simple python code w/ NumPy  
No partitioning code  
No MPI code

```
$ legate --launcher mpirun --nodes 64 --gpus 8 matmul.py
```

Scales to multi-GPU multi-Node at run time

## Get your science done faster!

- Ideal for **domain scientists and researchers**.
- Prototype quickly in **the NumPy API**.
- **Scaling** from single CPU core to a multi-GPU multi-node supercomputer with thousands of GPUs.
- **Use HPC clusters without HPC programming** or expensive rewrites in e.g. MPI.
- Implements most NumPy functionality, **with growing support for SciPy, and other scientific libraries**.

# cuPyNumeric and Legate Slurm Integration

Simple Integration Enables Rapid Prototyping with cuPyNumeric on HPC Clusters

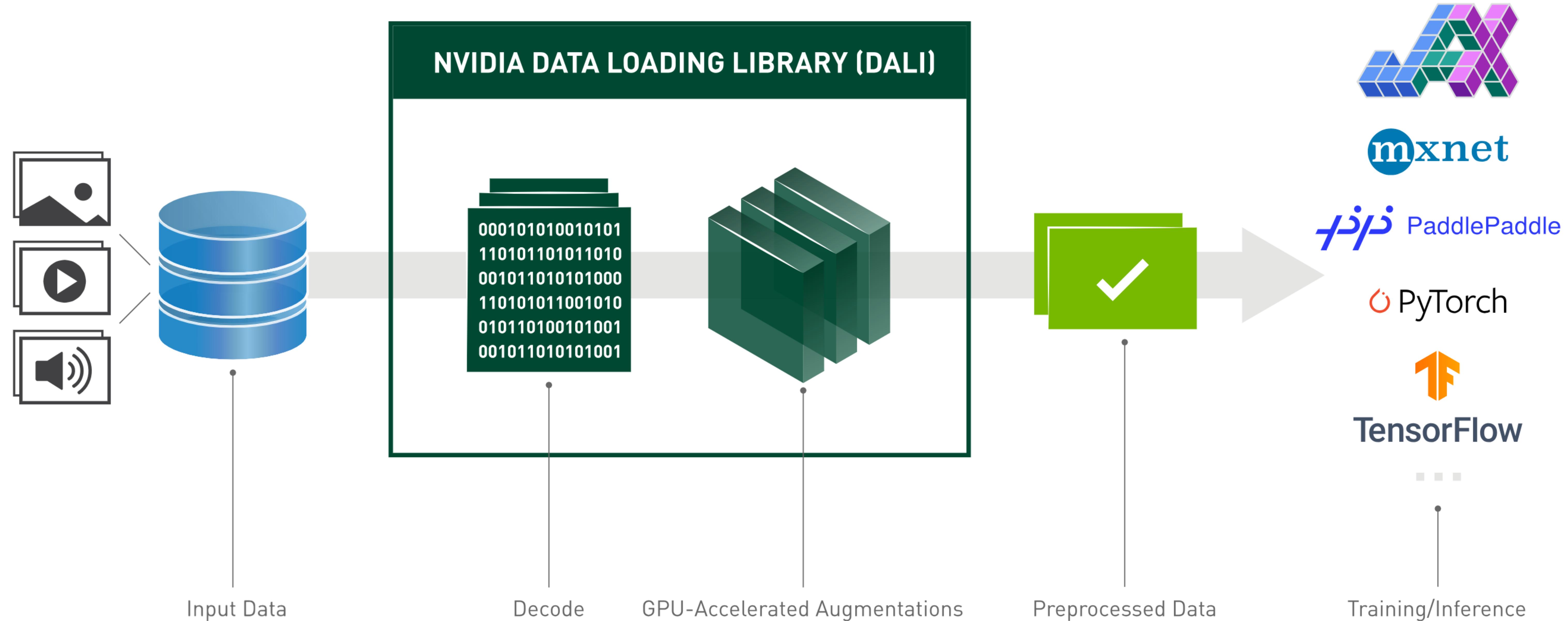
```
#!/bin/bash
#SBATCH --job-name=fwi-legate
#SBATCH --nodes=4          # 4 Compute nodes
#SBATCH --ntasks-per-node=4      # 4 GPUs per node (e.g., HGX H100)
#SBATCH --gpus-per-node=4
#SBATCH --time=02:00:00

# Load environment (Container or Module)
module load nvidia/hpc_sdk

# THE MAGIC COMMAND:
# Legate parses the variables above and scales automatically.
legate fwi_solver_3d.py --cpus 4 --gpus 4 --fbmem 60000
```

# NVIDIA DALI

## Data Loading Library

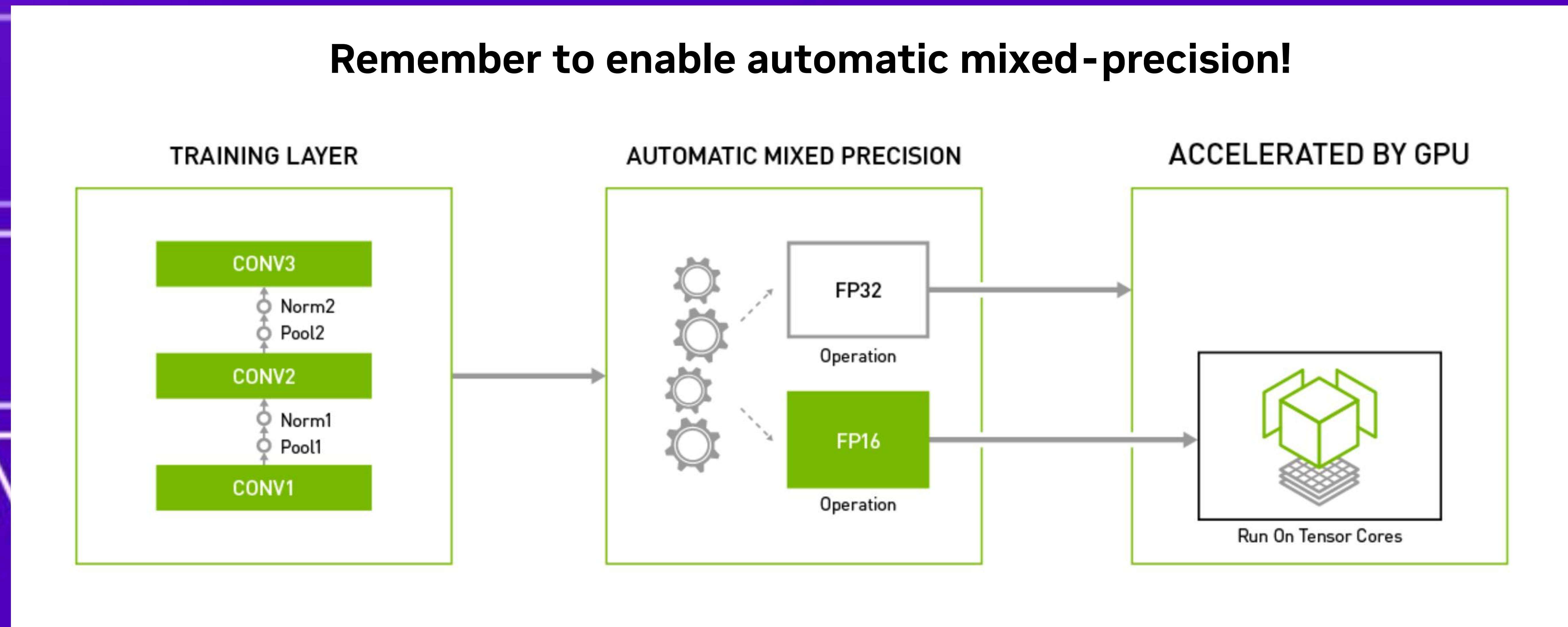


DALI is a high-performance alternative to built-in data loaders and data iterators. You can now run your data processing pipelines on the GPU, reducing the total time it takes to train a neural network. Data processing pipelines implemented using DALI are portable because they can easily be retargeted to TensorFlow, PyTorch, and MXNet.

# Training

# From CUDA and Python to PyTorch Lightning

**Remember to enable automatic mixed-precision!**



Python  
CUDA/C++

The programming languages used to build PyTorch, Lightning Fabric, and PyTorch Lightning.

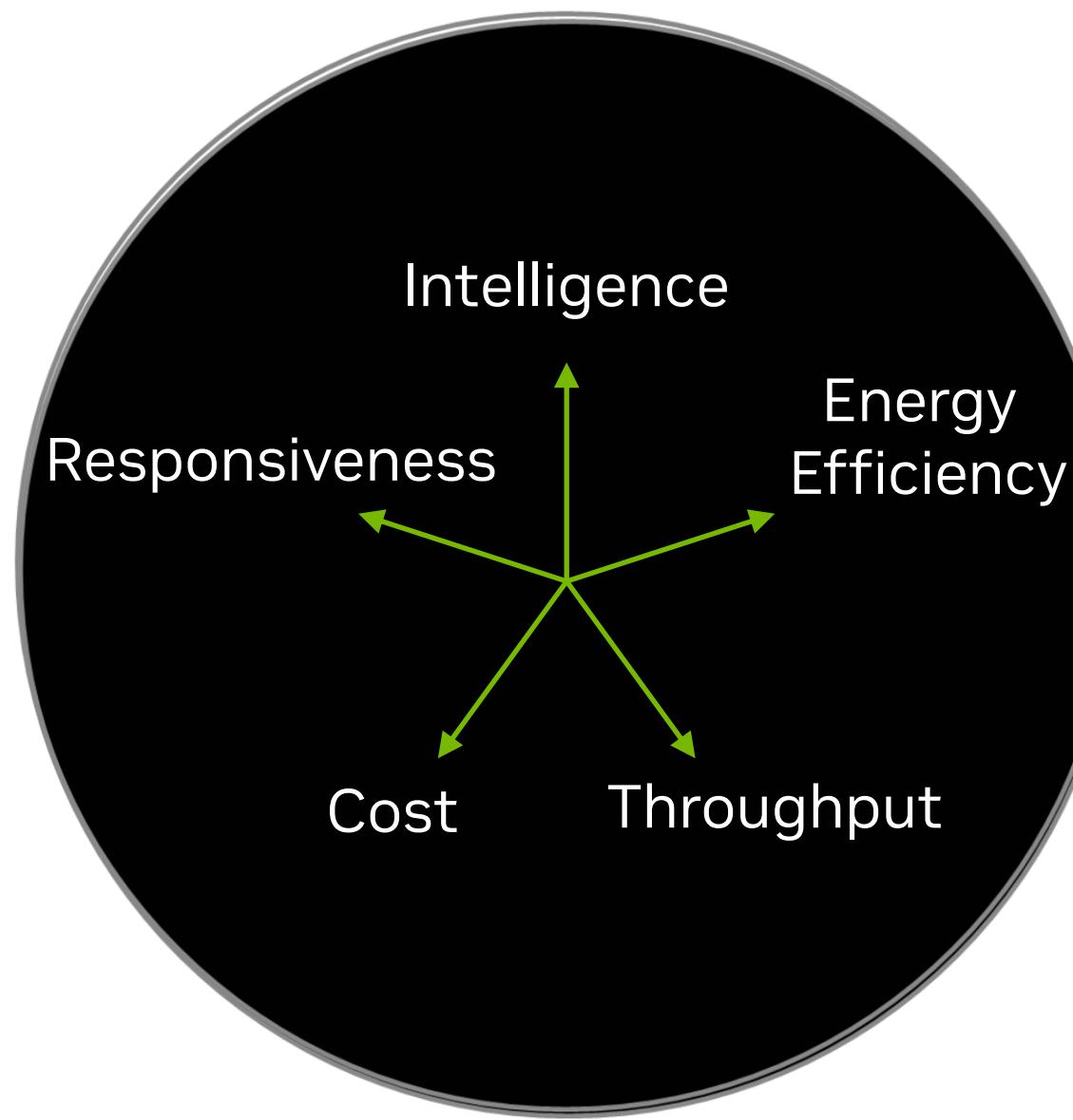
PyTorch Lightning is built on top of PyTorch, providing a high-level API for distributed training. It includes features like automatic mixed-precision, gradient accumulation, and easy deep learning workloads.

# Inference

# Inference Requires a SMART Platform



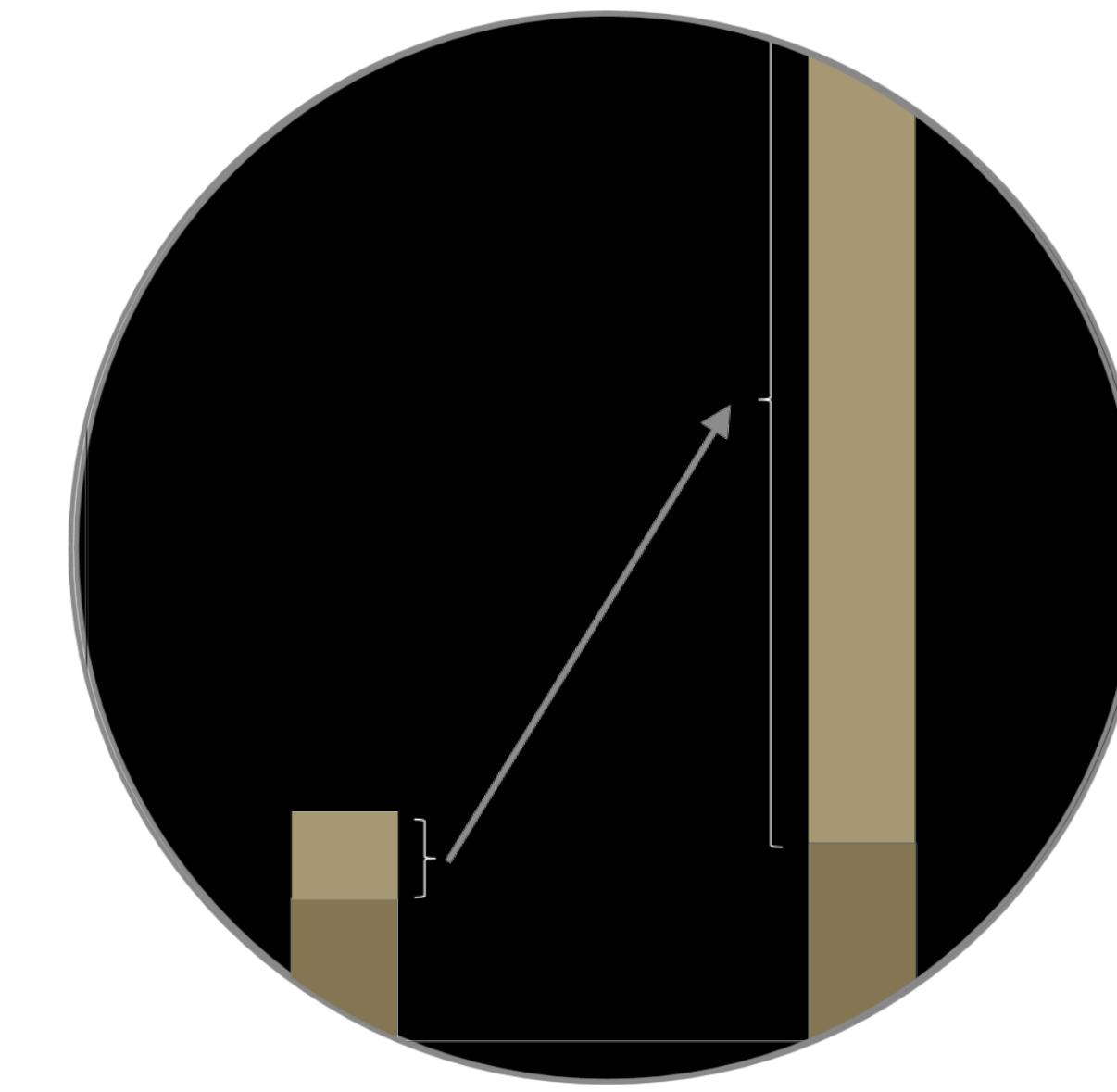
**S**cale and Complexity



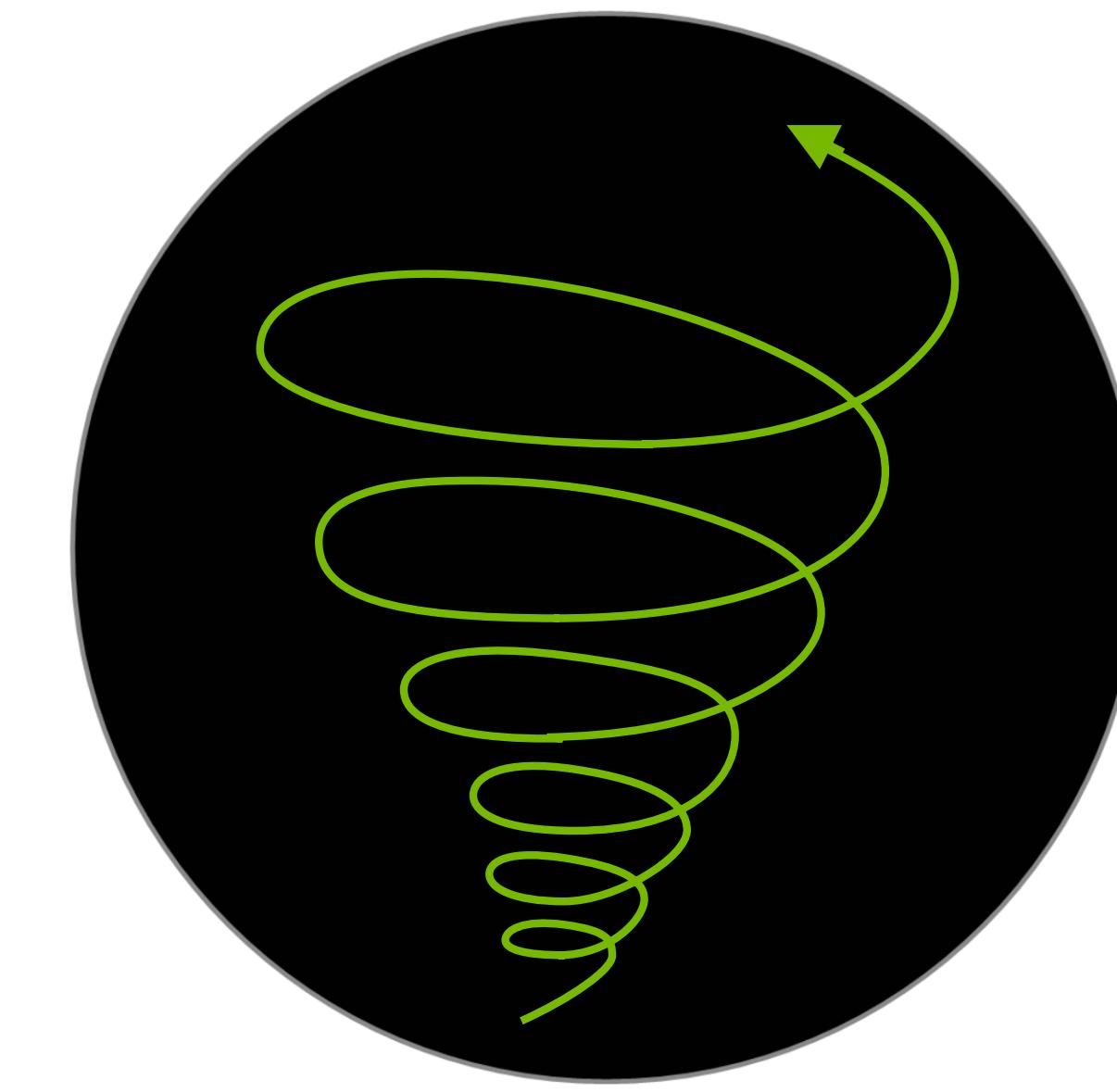
**M**ulti-Dimensional Performance



**A**rchitecture and Software

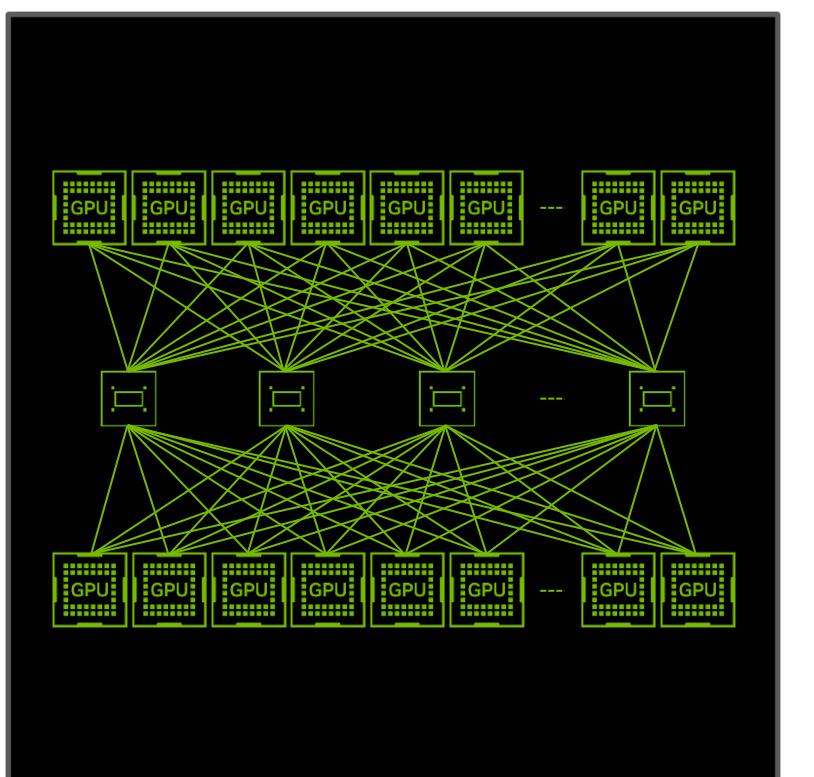


**R**OI Driven by Performance



**T**echnology Ecosystem and Install Base

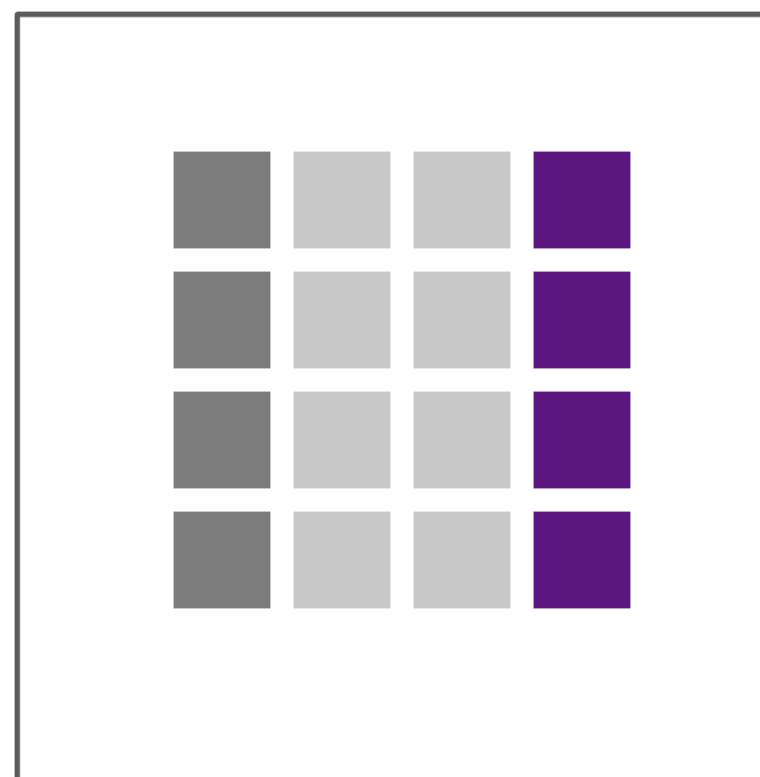
NVLINK 72



Spectrum-X Ethernet



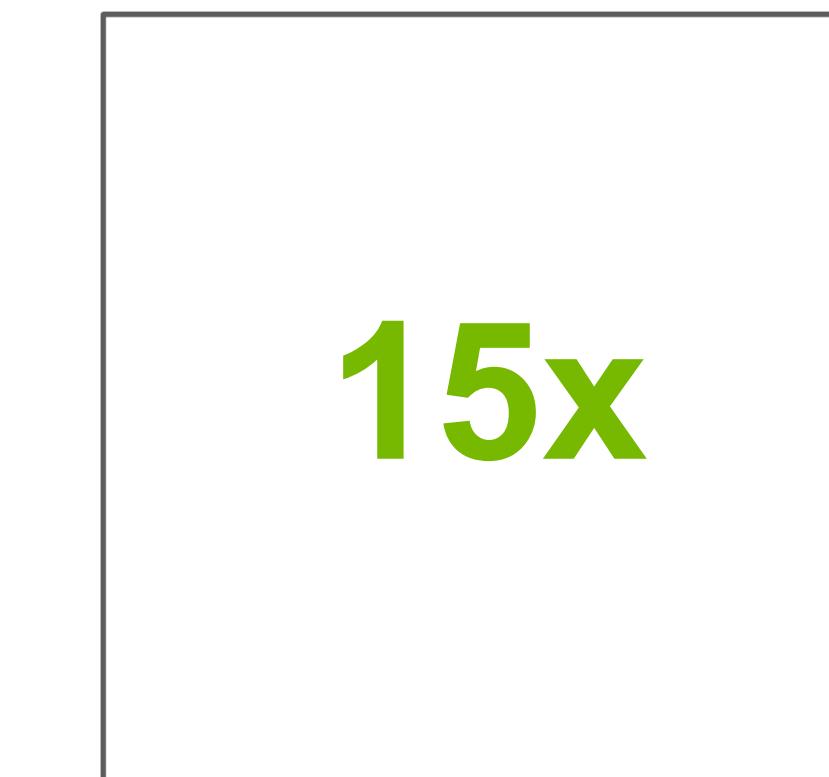
NVFP4



Dynamo



Blackwell ROI



Open-Source Software

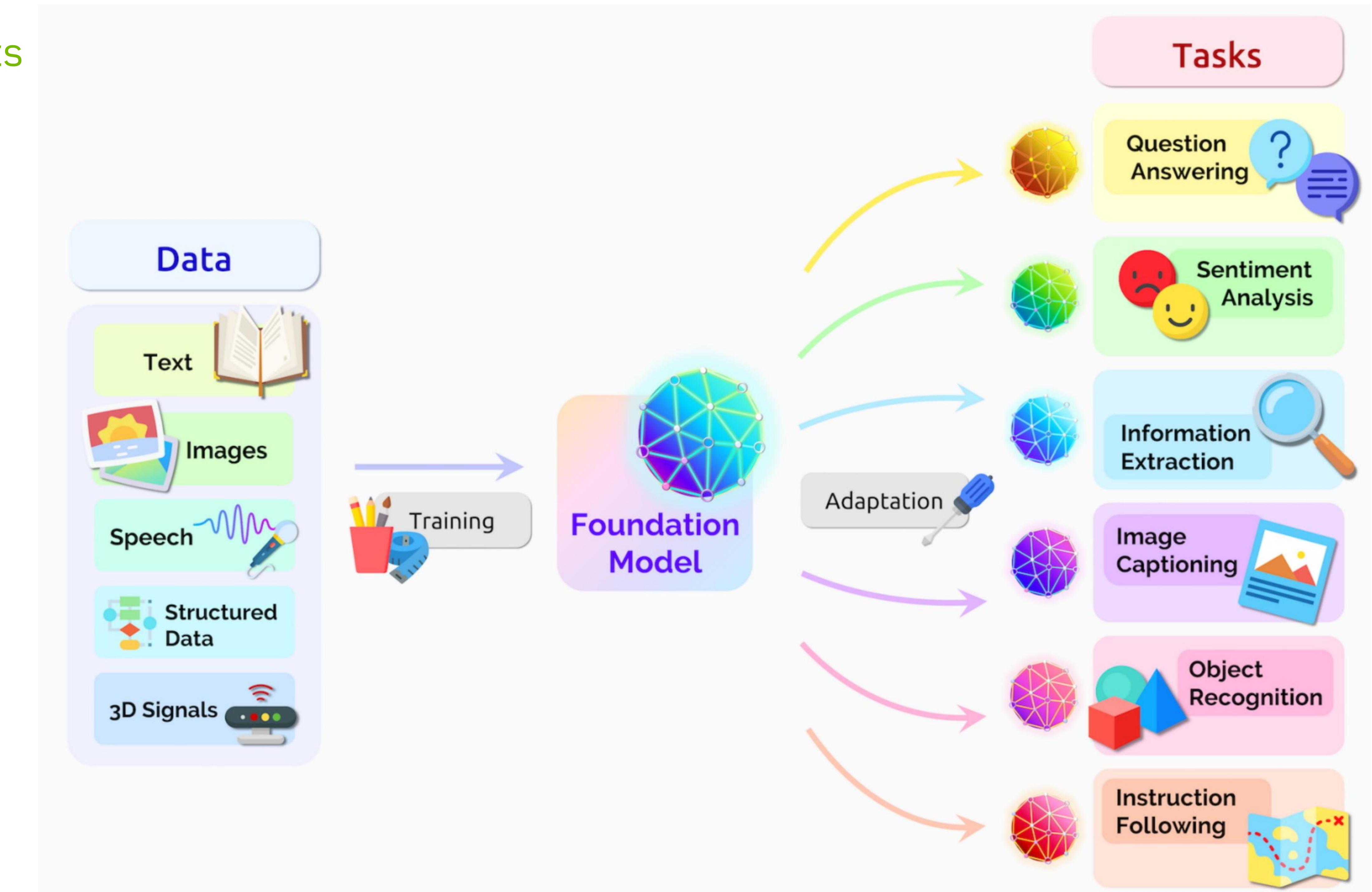


# AI Use Cases

# What are Foundation Models

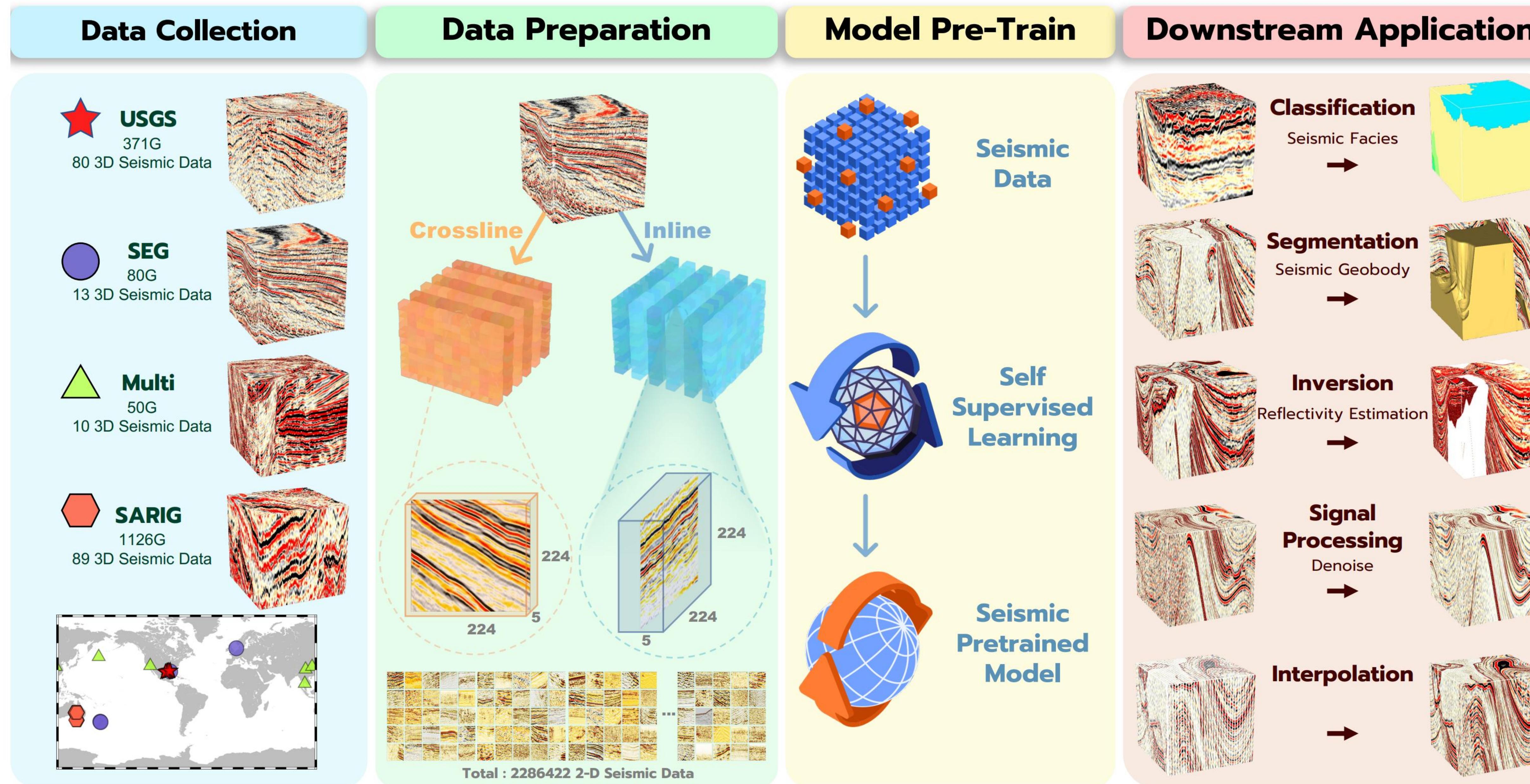
General purpose machine learning models

- Trained on massive unlabeled datasets to handle a wide variety of jobs from translating text to analyzing seismic images.
- General purpose adaptable models that can be fine-tuned for diverse tasks like computer vision, natural language processing with minimal task-specific training.
- Scalable and reusable architectures which leverage learned patterns to perform new tasks efficiently, reducing the need for extensive retraining or custom model development.



Source: [What Are Foundation Models? | NVIDIA Blogs](#)

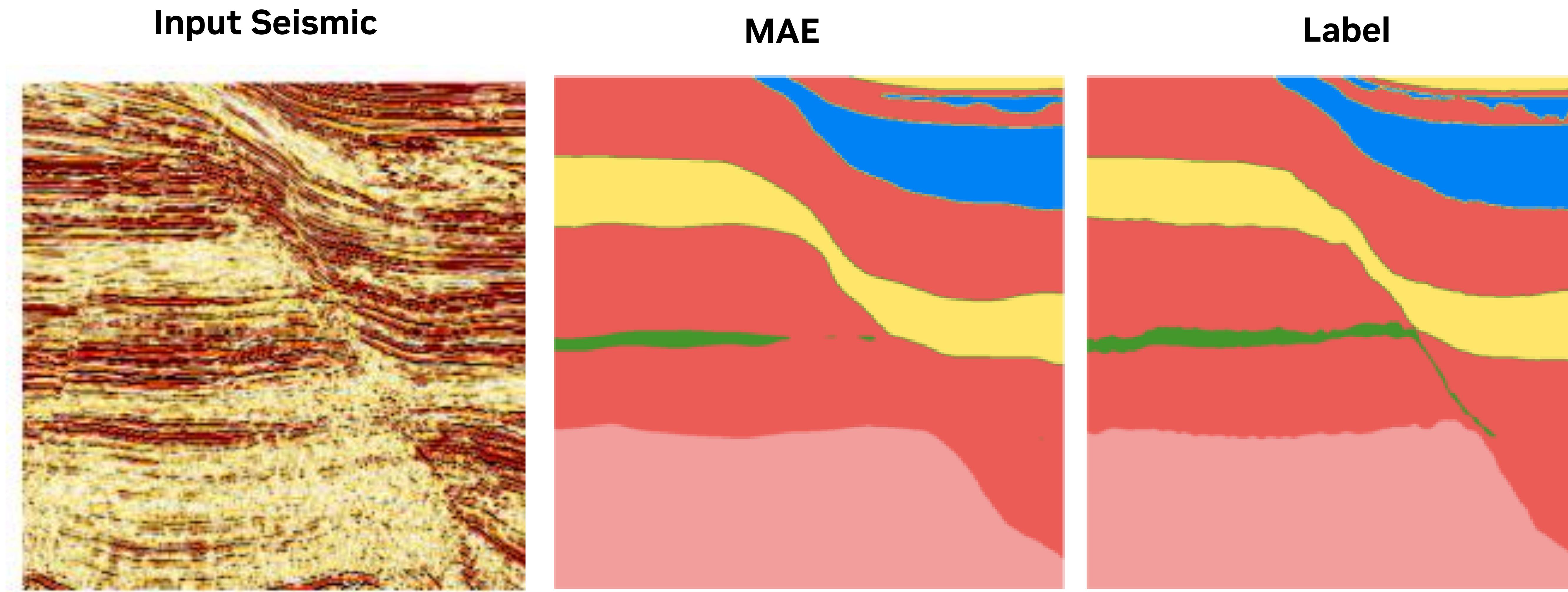
# Seismic Foundation Model (SFM)



**SFM** represents the first successful large-scale foundational model trained exclusively within the seismic domain [[reference](#)].

# Masked Auto-Encoder fine-tuned on 200 samples

Trained with self-supervised learning on 2.2 million seismic images utilizing NVIDIA TAO



Facies classification

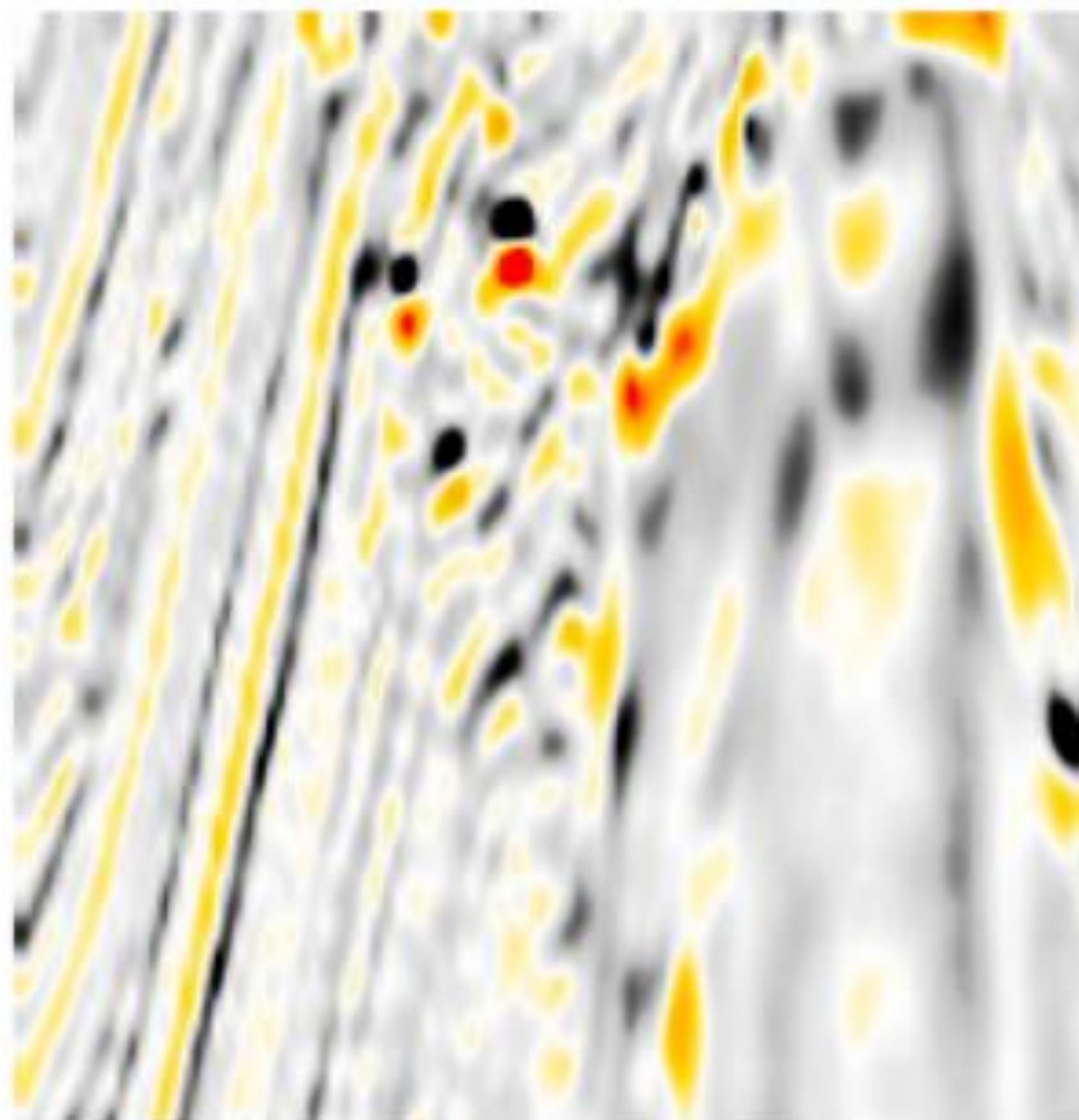
(Results reproduced from [Sheng et al., 2025](#))

“Seismic Foundation Model (SFM): a new generation deep learning model in geophysics

# Masked Auto-Encoder fine-tuned on 3000 samples

Trained with self-supervised learning on 2.2 million seismic images utilizing NVIDIA TAO

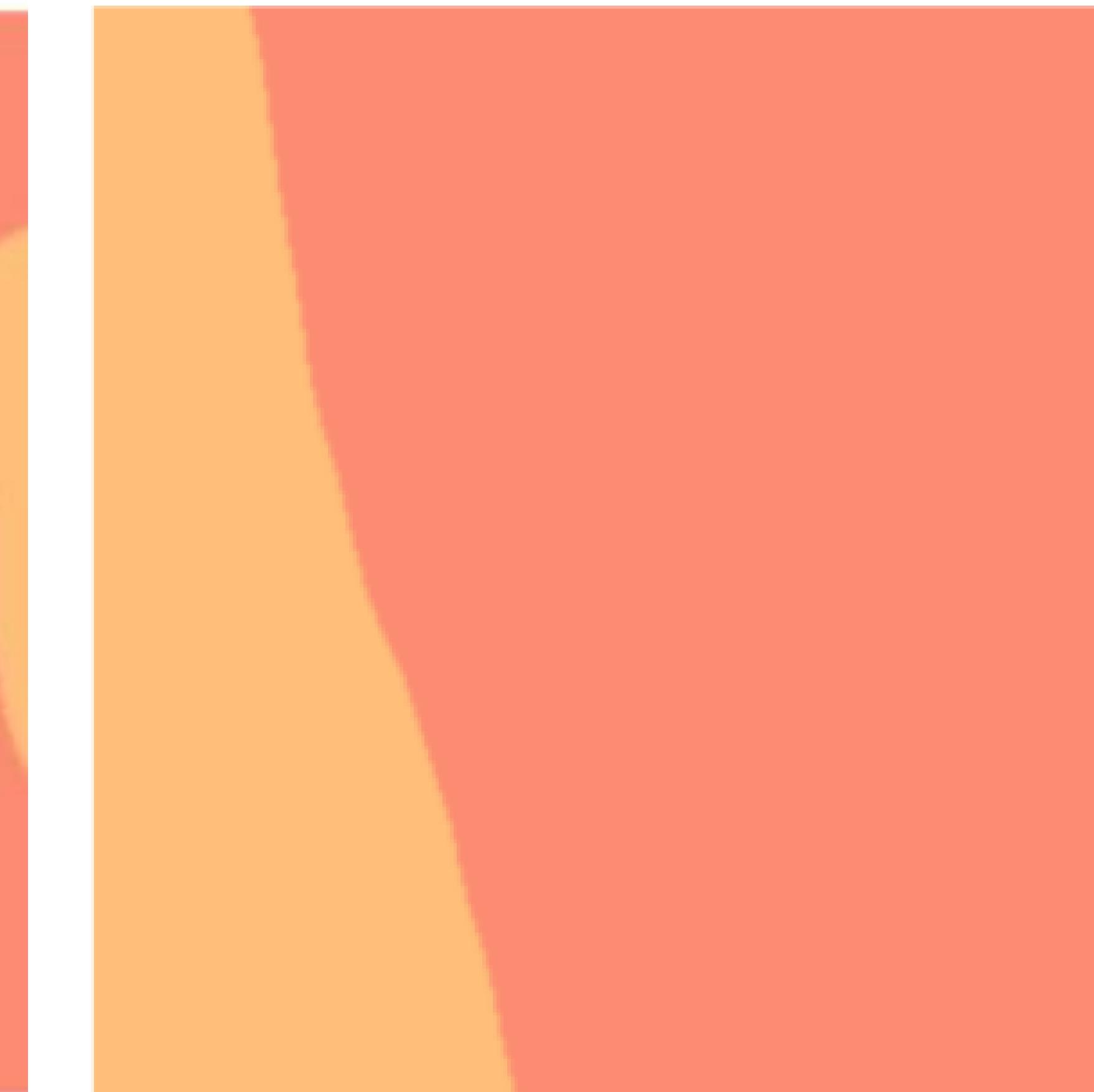
**Input Seismic**



**U-Net**



**MAE**



**Label**



Salt Segmentation

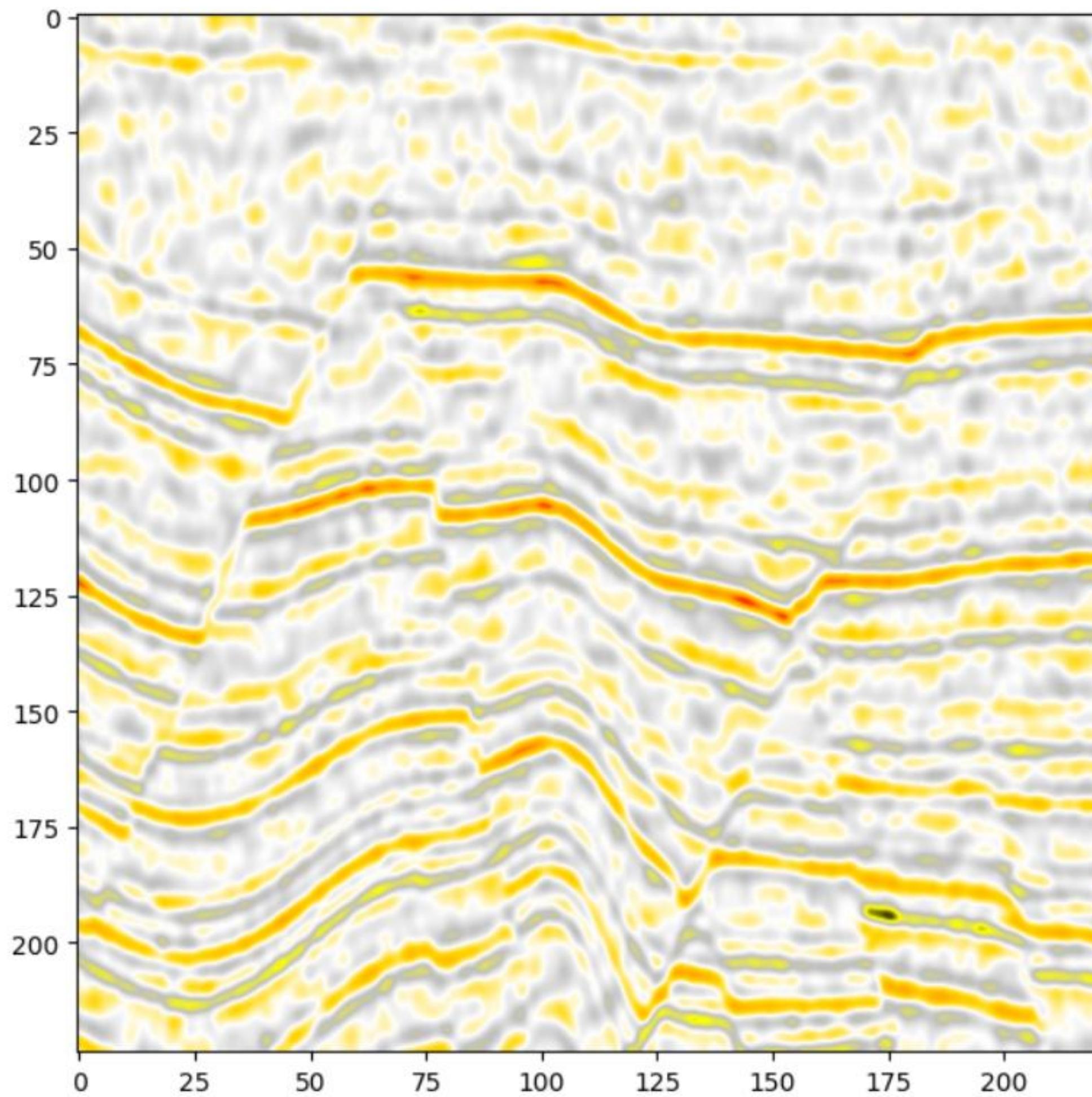
(Results reproduced from [Sheng et al., 2025](#))

“Seismic Foundation Model (SFM): a new generation deep learning model in geophysics

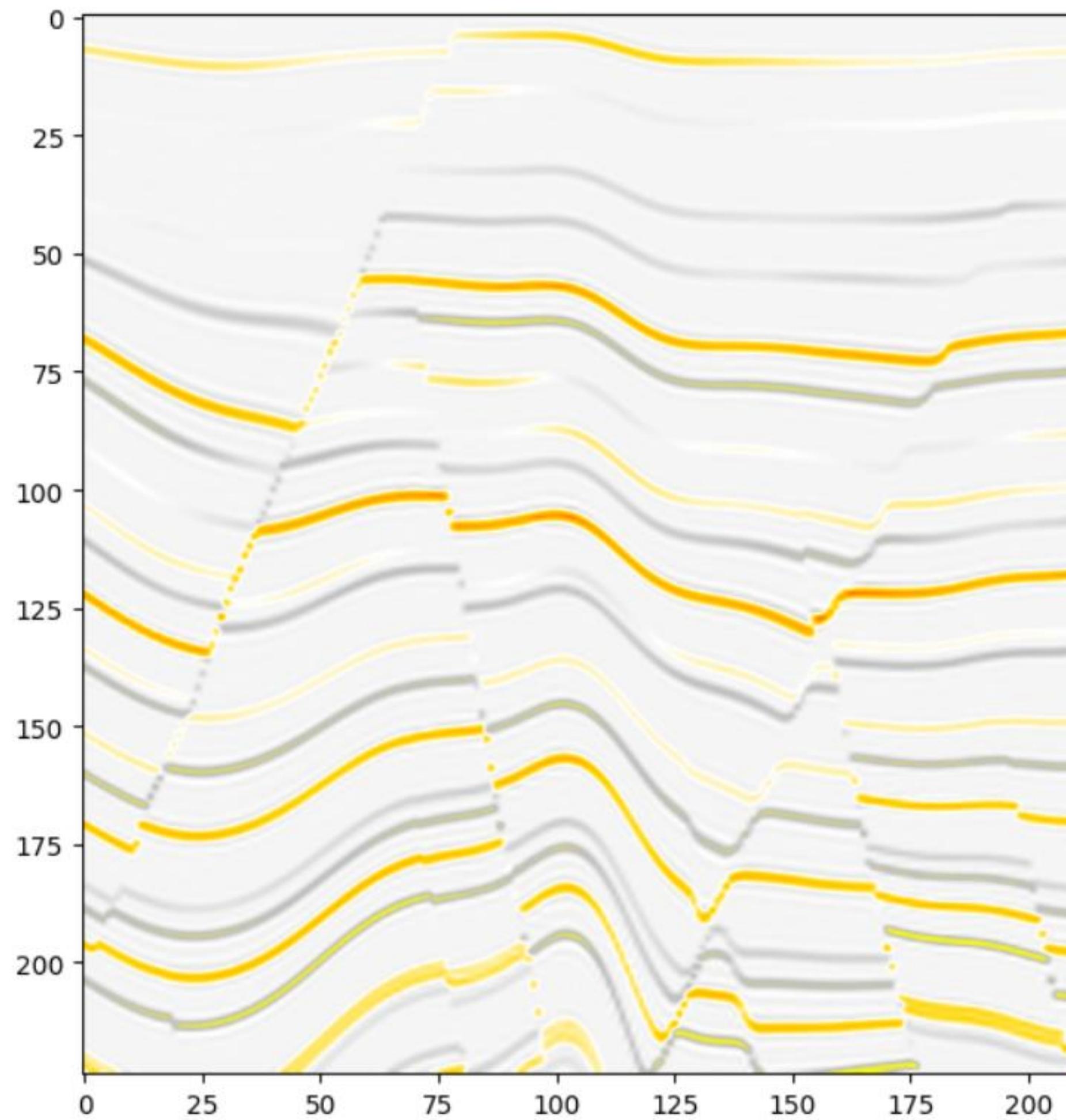
# Masked Auto-Encoder fine-tuned on 1000 samples

Trained with self-supervised learning on 2.2 million seismic images utilizing NVIDIA TAO

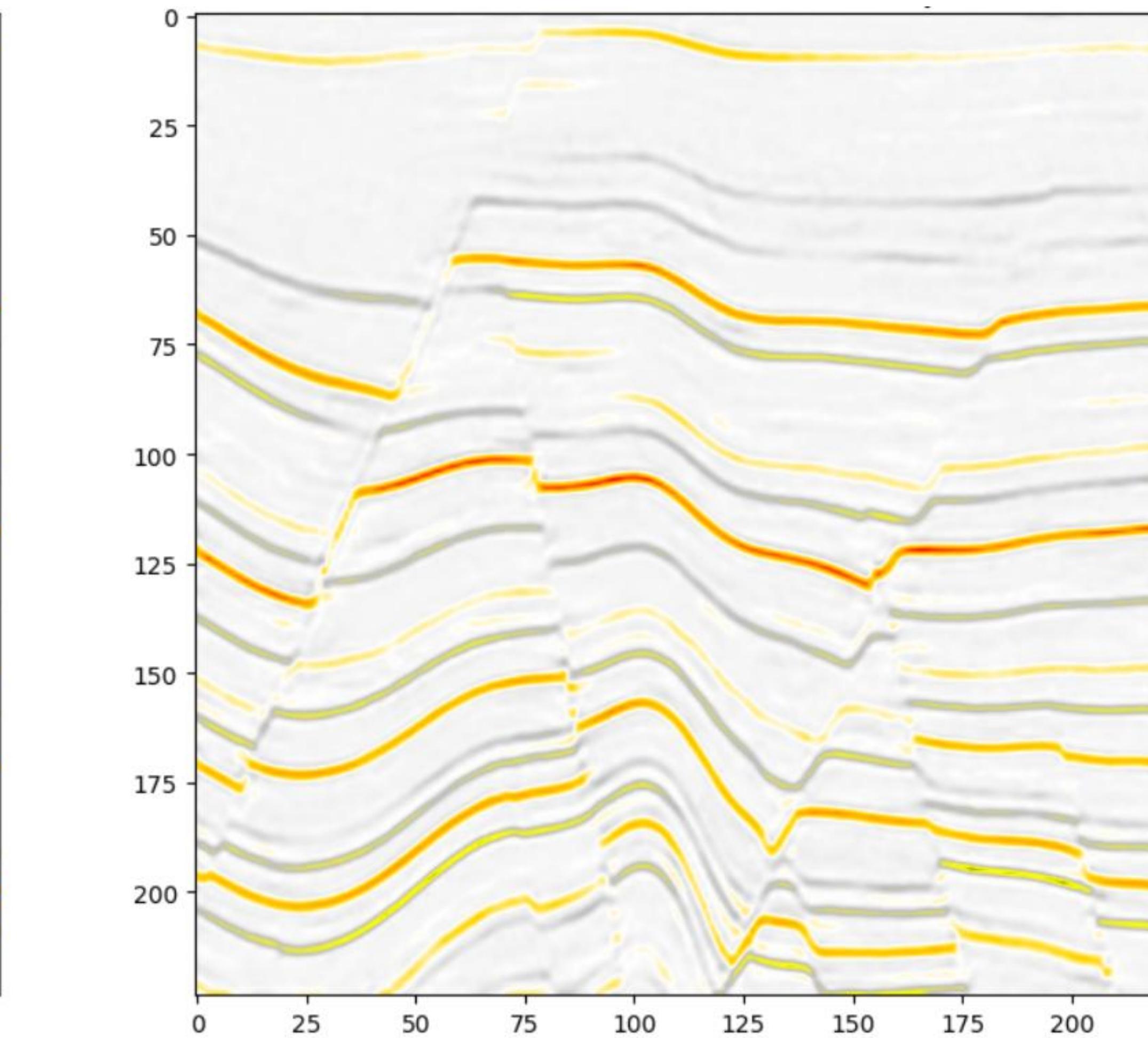
**Input Seismic**



**MAE**



**Label**



**Reflectivity inversion**

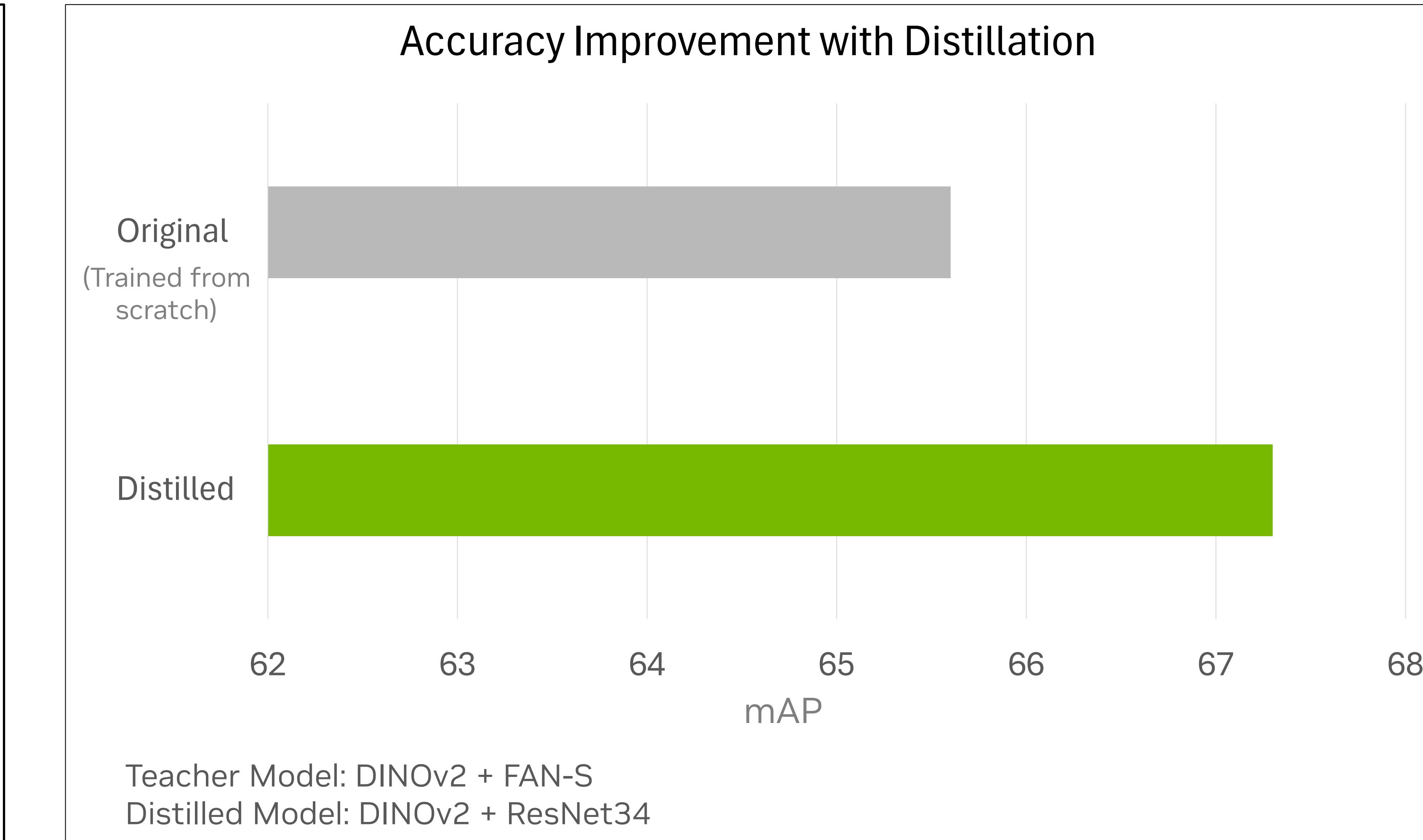
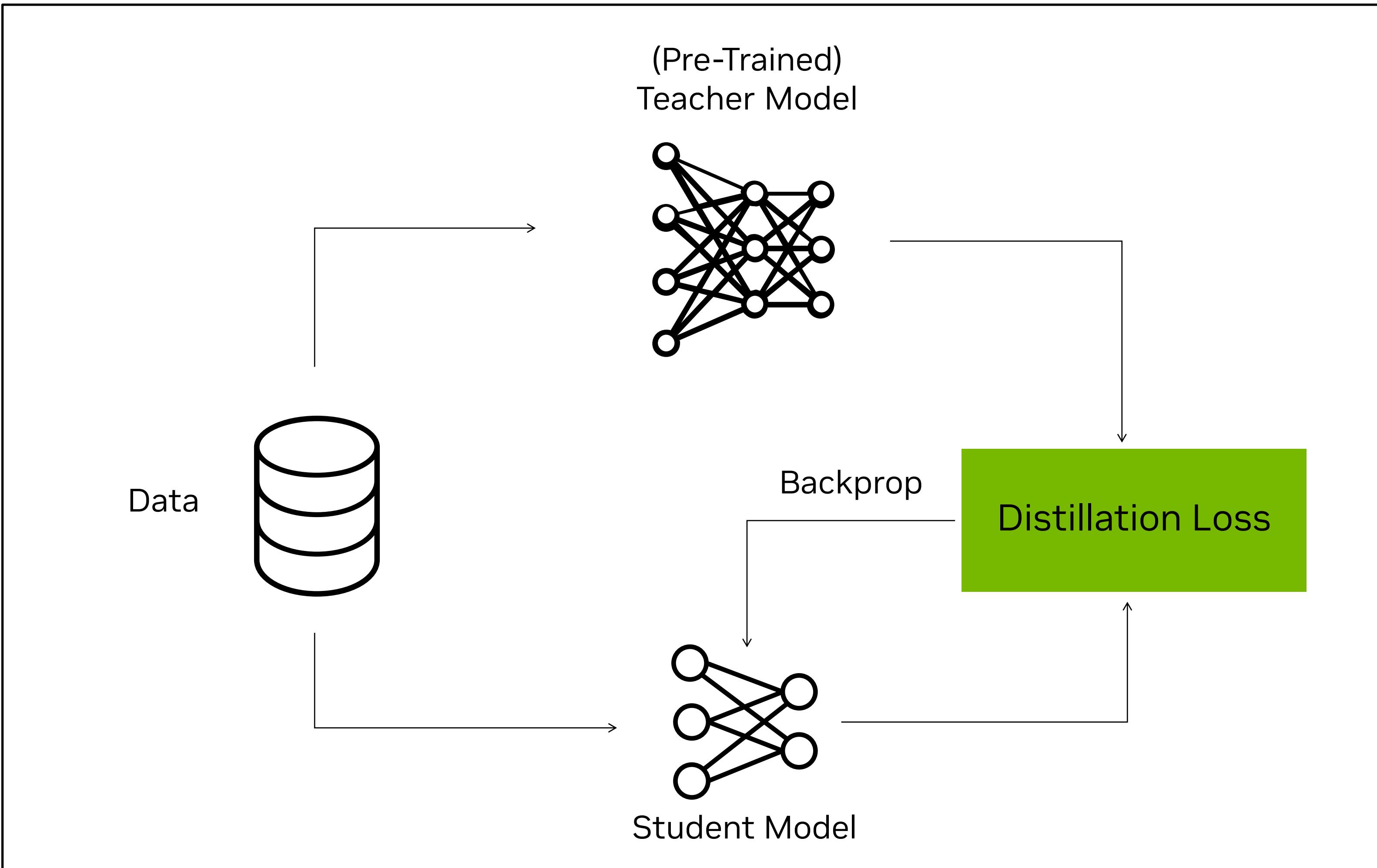
(Results reproduced from [Sheng et al., 2025](#))

“Seismic Foundation Model (SFM): a new generation deep learning model in geophysics

# Knowledge Distillation

Improved accuracy with distillation + task finetuning

Distill knowledge from larger teacher network to smaller student model



## Key Benefits:

Reduce Model Size

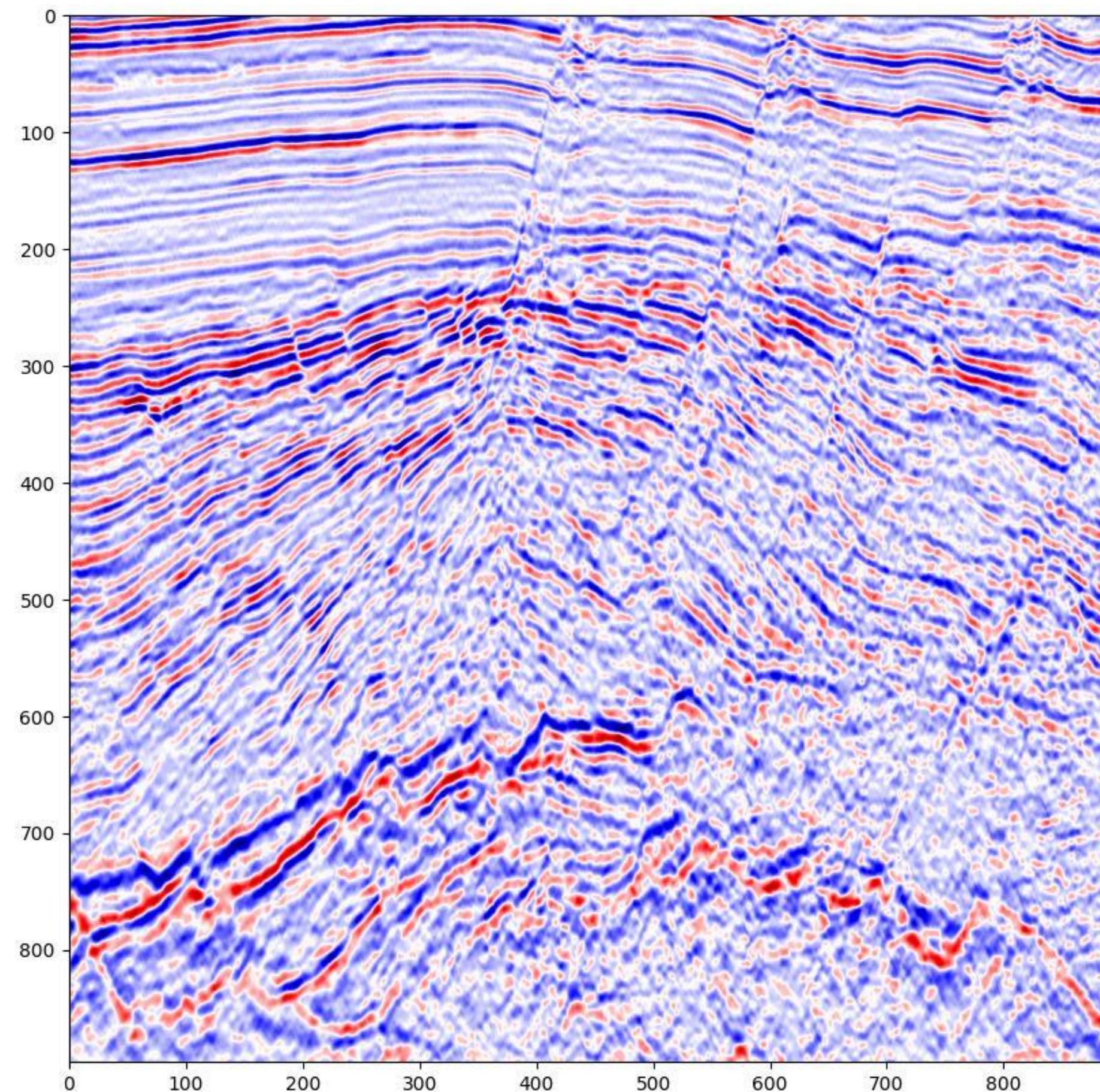
Increase Model Accuracy

Faster Training

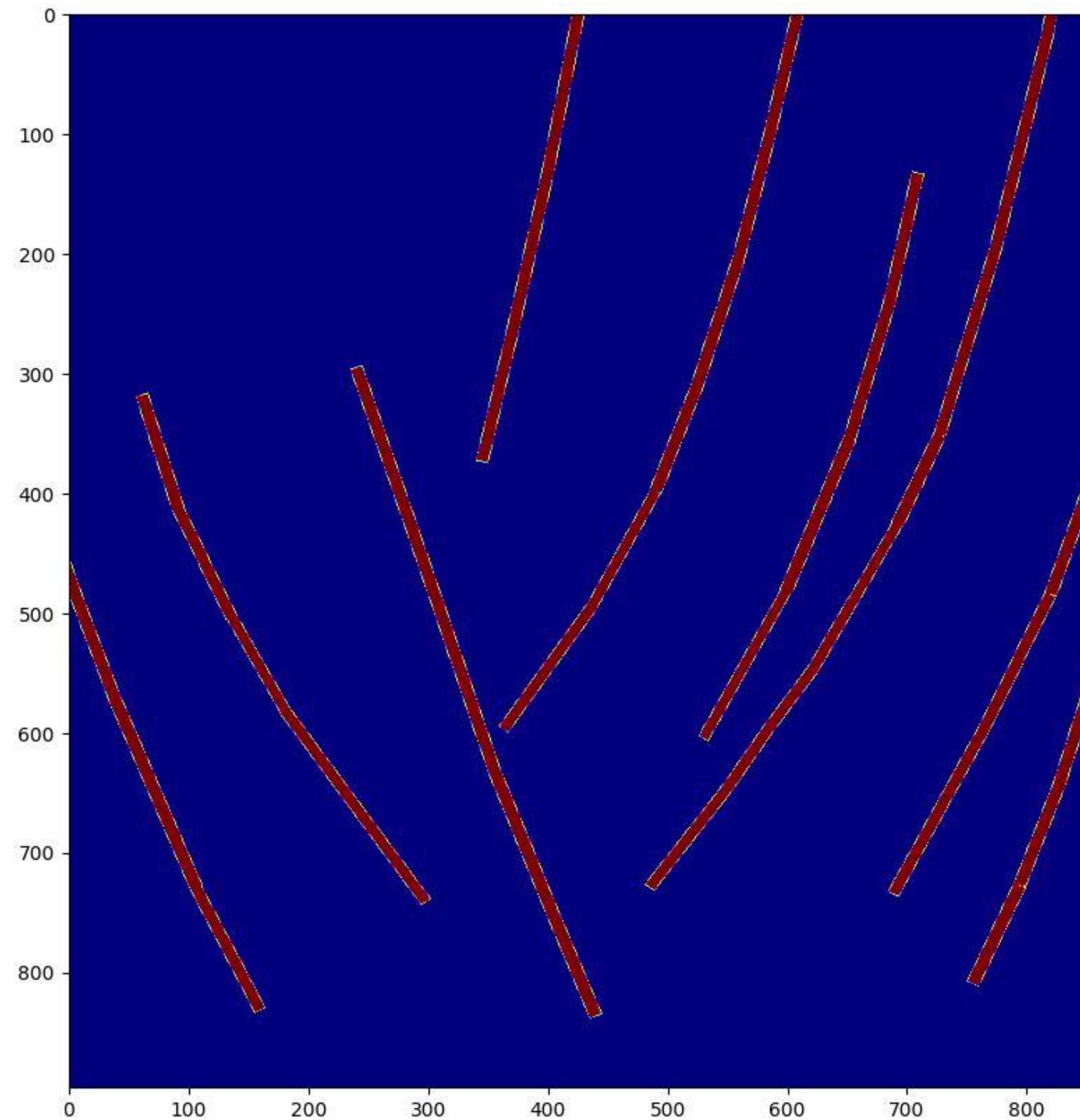
# DINOv2 Self-Supervised Learning Enhanced by Knowledge Distillation and Task-Specific Fine-Tuning

Initially trained on 2.2 million images, followed by distillation and fine-tuning with 100 samples.

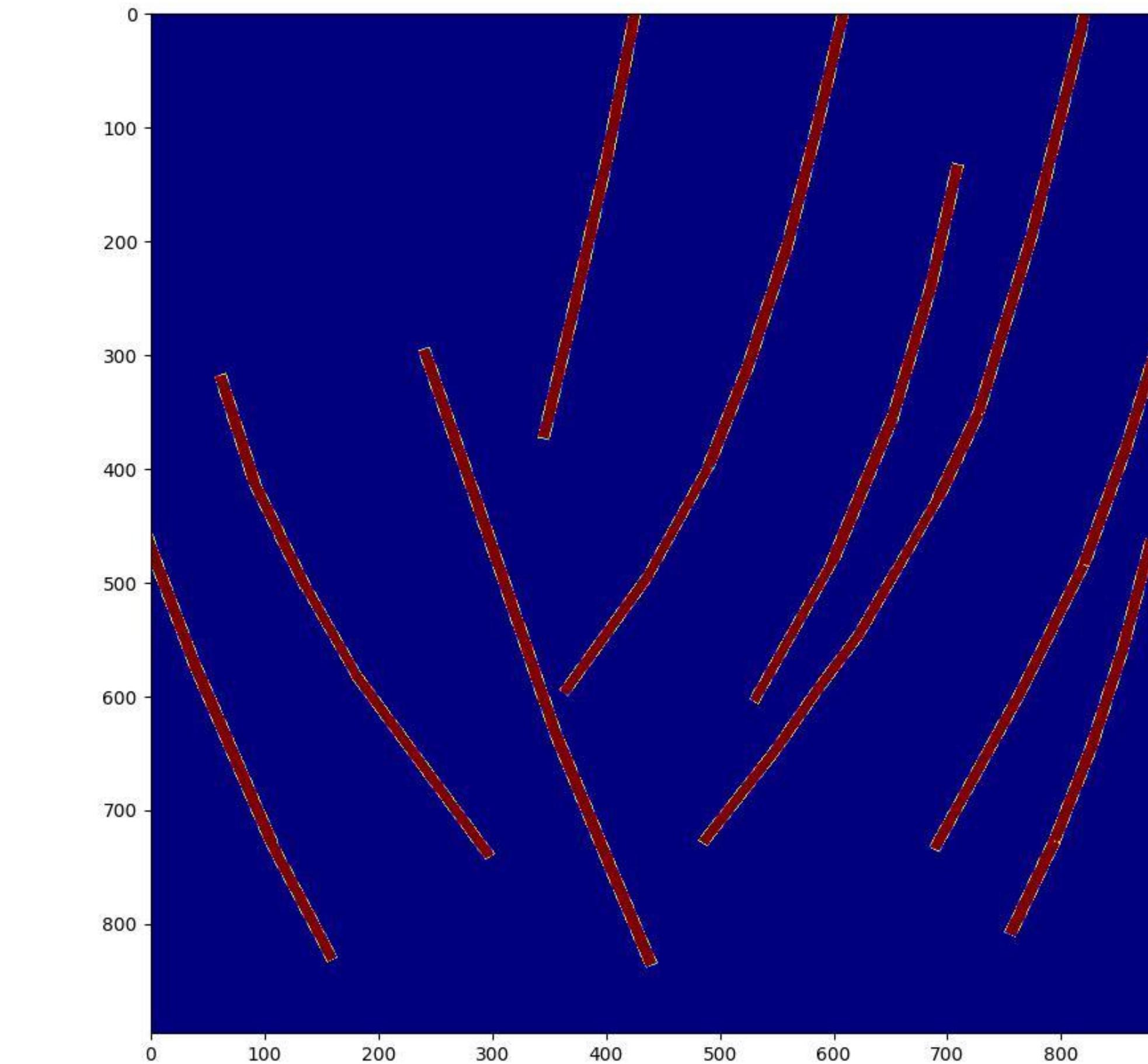
**Input Seismic**



**DINOv2 (distilled)**



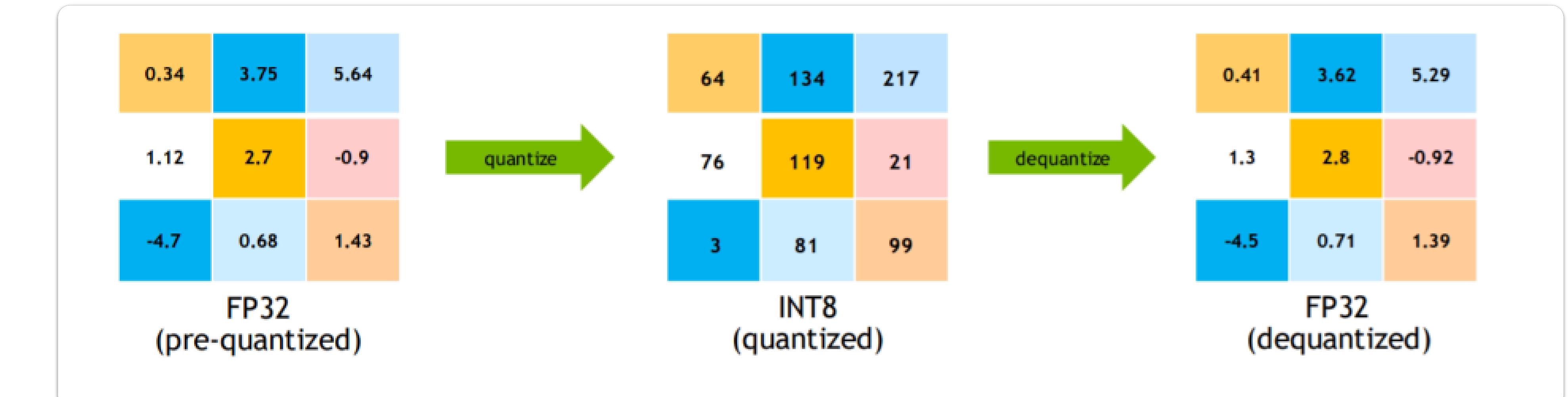
**Label**



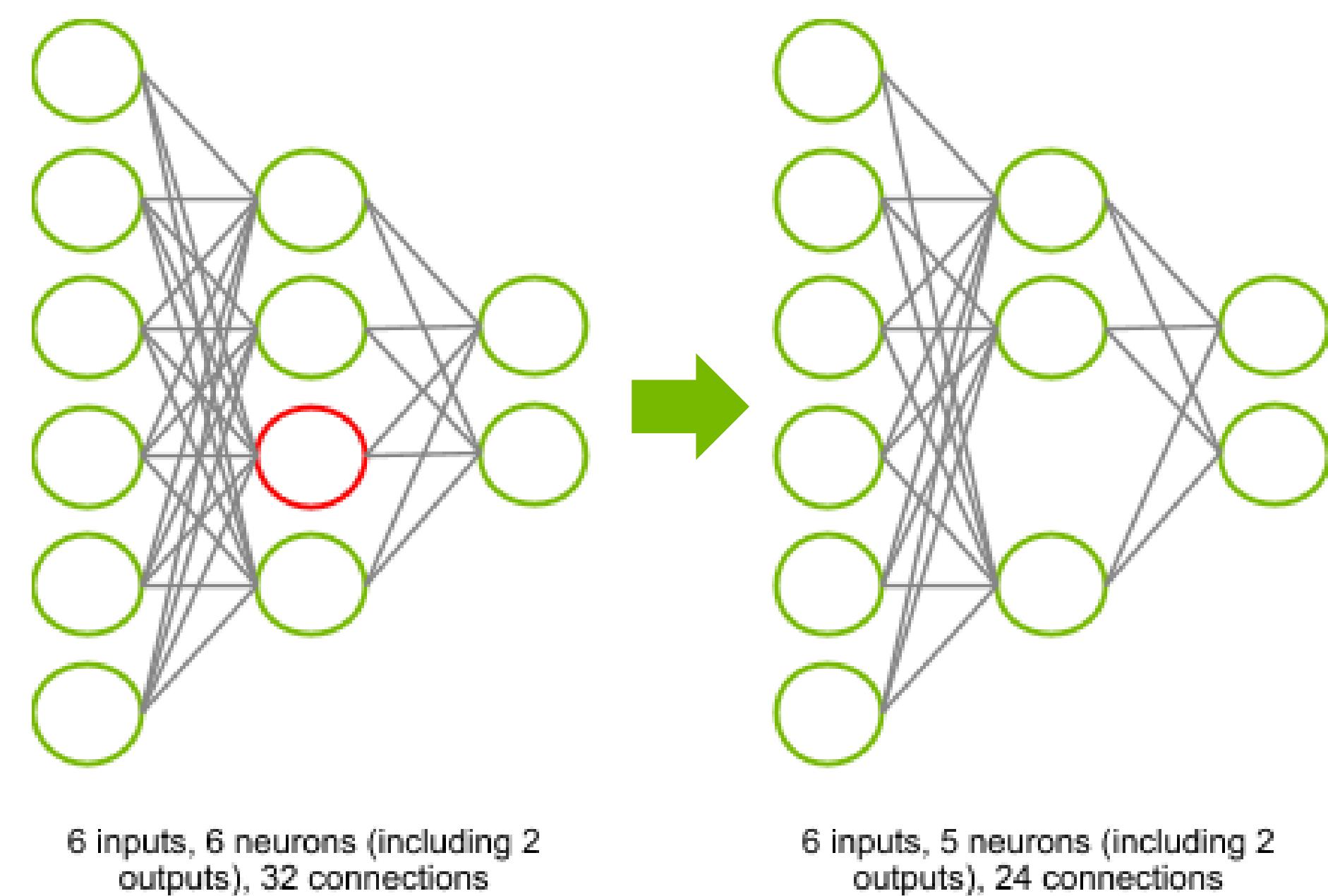
# Quantization and Pruning

Improved accuracy with distillation + task finetuning

- TAO will support quantization and pruning for foundation model
- FP8, INT8/INT4 Weight only, INT8 Smooth Quant, AWQ, GPTQ quantization will be supported
- Scaling laws and "Train large and compress more" strategy.



## Quantization

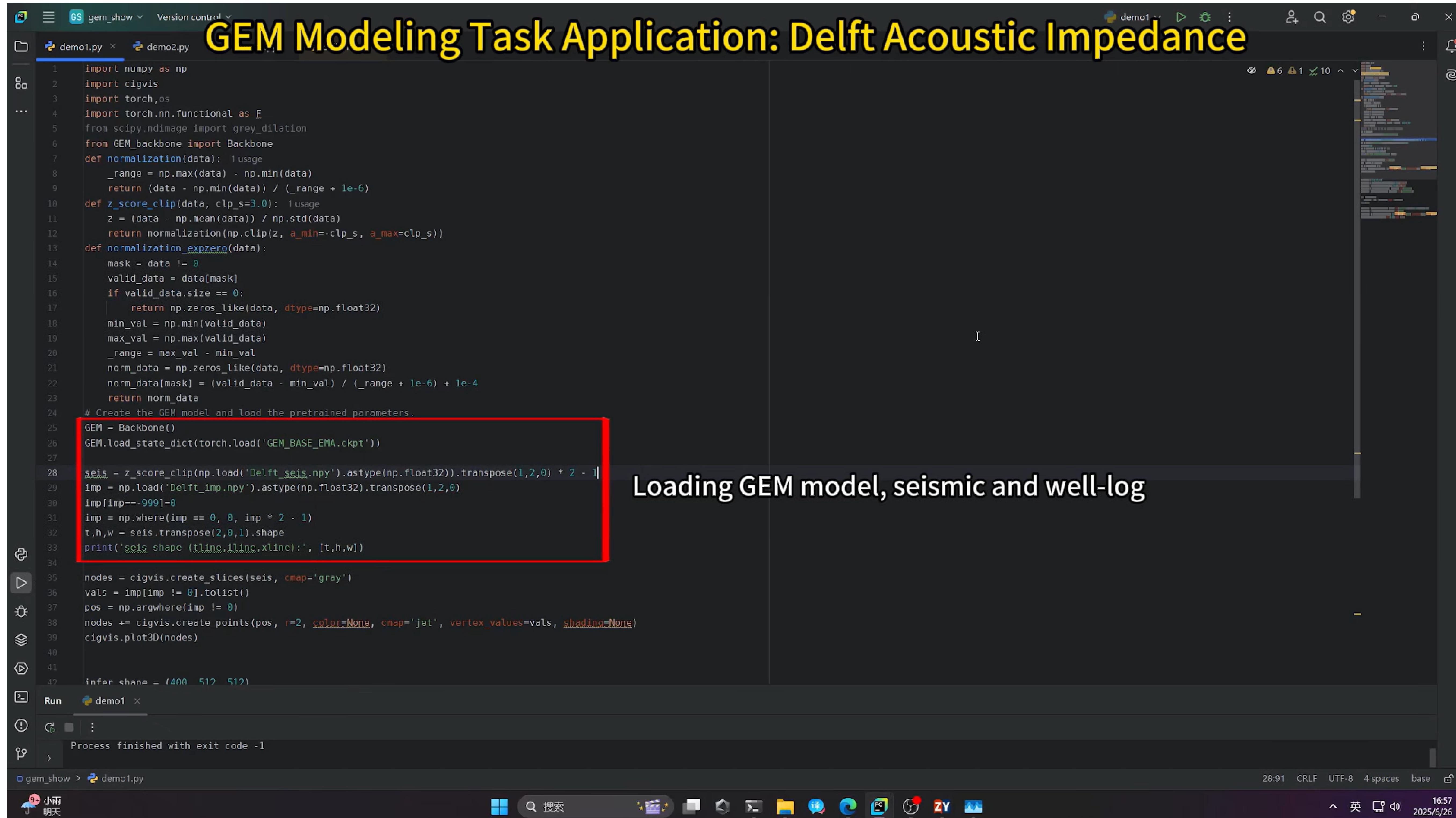


## Pruning

# **Geological Everything Model 3D**

# Geological Everything Model 3D

A Promptable Foundation Model for Unified and Zero-Shot Subsurface Understanding



The screenshot shows a code editor window titled "GEM Modeling Task Application: Delft Acoustic Impedance". The code in the editor is as follows:

```
import numpy as np
import cigvis
import torch,os
import torch.nn.functional as F
from scipy.ndimage import grey_dilation
from GEM_backbone import Backbone
def normalization(data): 1 usage
    _range = np.max(data) - np.min(data)
    return (data - np.min(data)) / (_range + 1e-6)
def z_score_clip(data, clp_s=3.0): 1 usage
    z = (data - np.mean(data)) / np.std(data)
    return normalization(np.clip(z, a_min=-clp_s, a_max=clp_s))
def normalization_exzero(data):
    mask = data != 0
    valid_data = data[mask]
    if valid_data.size == 0:
        return np.zeros_like(data, dtype=np.float32)
    min_val = np.min(valid_data)
    max_val = np.max(valid_data)
    _range = max_val - min_val
    norm_data = np.zeros_like(data, dtype=np.float32)
    norm_data[mask] = (valid_data - min_val) / (_range + 1e-6) + 1e-4
    return norm_data
# Create the GEM model and load the pretrained parameters.
GEM = Backbone()
GEM.load_state_dict(torch.load('GEM_BASE_EMA.ckpt'))
seis = z_score_clip(np.load('Delft_seis.npy').astype(np.float32).transpose(1,2,0) * 2 - 1)
imp = np.load('Delft_imp.npy').astype(np.float32).transpose(1,2,0)
imp[imp==0]=0
imp = np.where(imp == 0, 0, imp * 2 - 1)
t,h,w = seis.transpose(2,0,1).shape
print('seis shape (tline,iline,xline):', [t,h,w])
nodes = cigvis.create_slices(seis, cmap='gray')
vals = imp[imp != 0].tolist()
pos = np.argwhere(imp != 0)
nodes += cigvis.create_points(pos, r=2, color=None, cmap='jet', vertex_values=vals, shading=None)
cigvis.plot3D(nodes)
infer_shape = (400 512 512)
```

A red box highlights the following lines of code:

```
GEM = Backbone()
GEM.load_state_dict(torch.load('GEM_BASE_EMA.ckpt'))
```

To the right of this highlighted code, the text "Loading GEM model, seismic and well-log" is displayed.

The status bar at the bottom of the code editor shows the following information:

- gem\_show > demo1.py
- 28:91 CRLF UTF-8 4 spaces base
- 16:57 2025/6/26

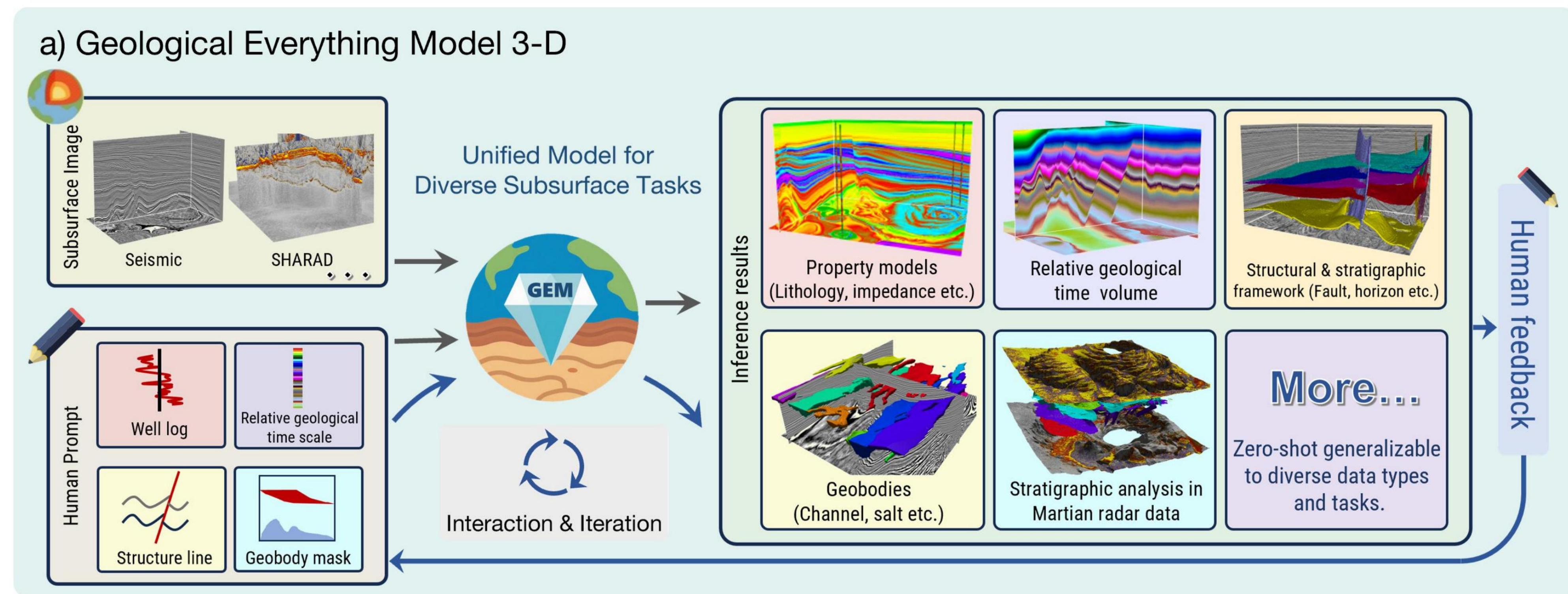
# Geological Everything Model 3D

A Promptable Foundation Model for Unified and Zero-Shot Subsurface Understanding

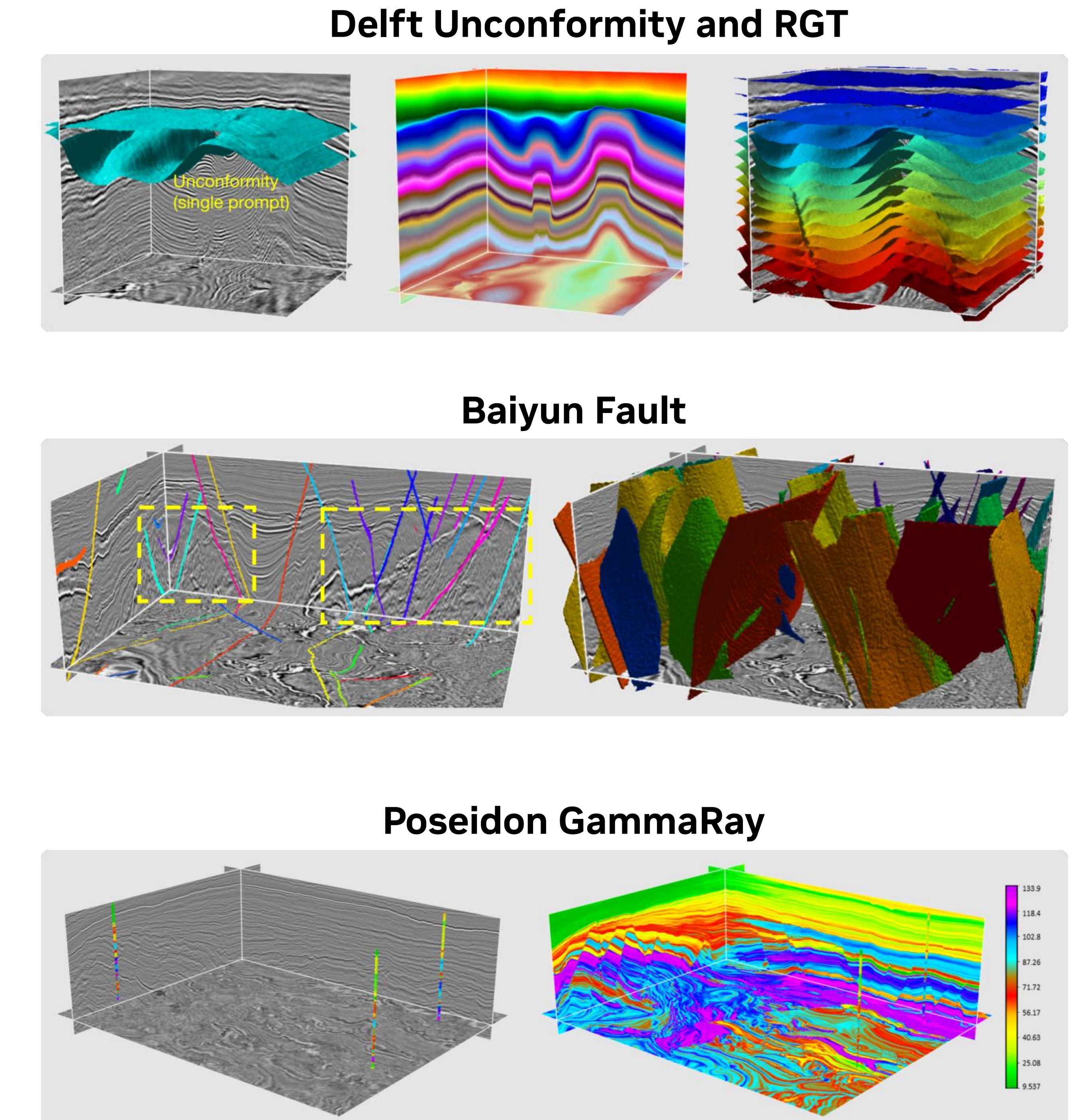


# Geological Everything Model 3D

A Promptable Foundation Model for Unified and Zero-Shot Subsurface Understanding

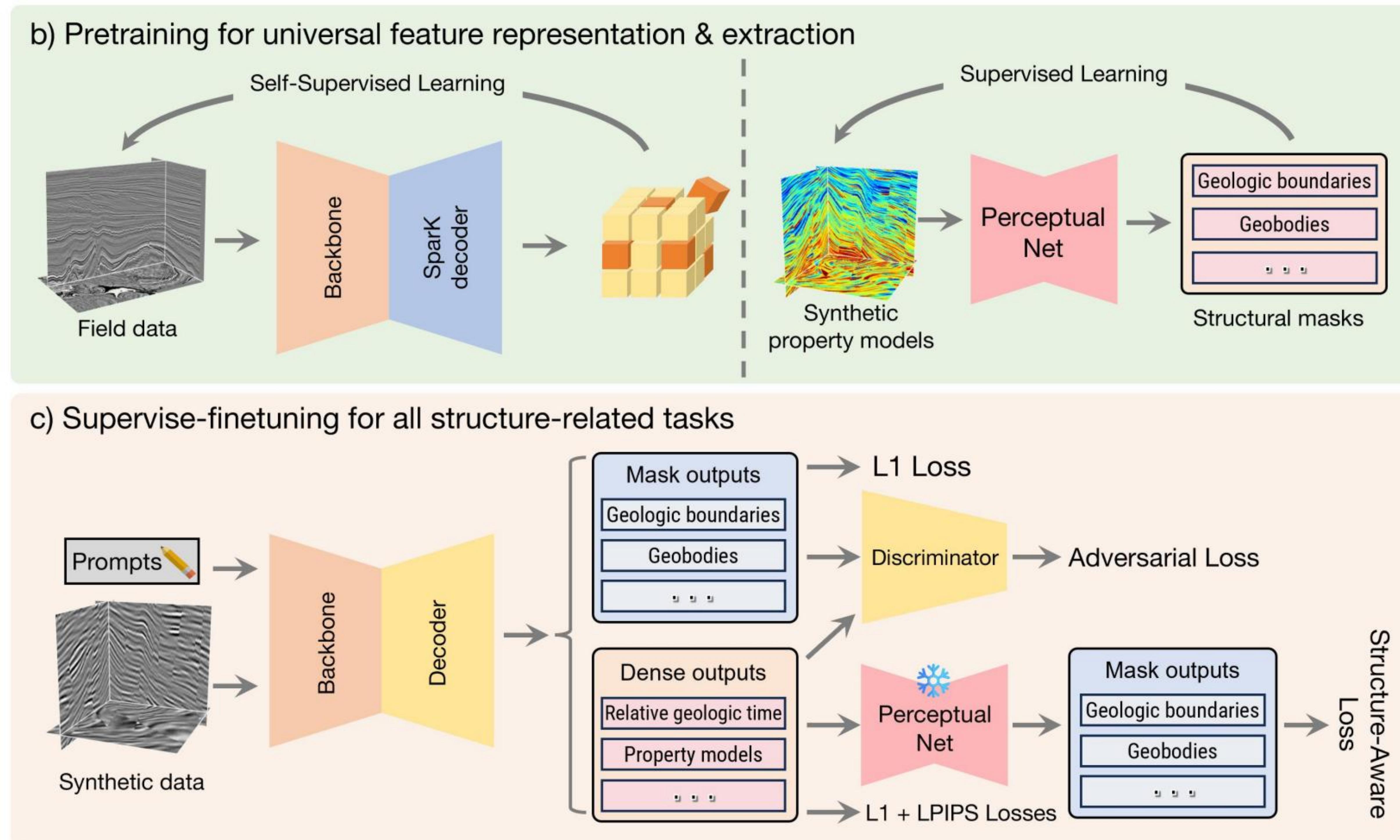


**Inference ->**  $\hat{Y} = p(Y | X_{img}, Y_{part})$



# Geological Everything Model 3D

A Promptable Foundation Model for Unified and Zero-Shot Subsurface Understanding



# Try Diffusion Model for Full-Waveform Inversion Using NVIDIA PhysicsNeMo

NVIDIA PhysicsNeMo Framework latest ▾

Overview

**Getting Started**

- System Requirements
- Install Guide

**User Guide**

- Training Recipe
- Logging and Checkpointing
- Model Architectures
- PhysicsNeMo Distributed
- Physics-guided
- Performance
- Data Curation
- Model Evaluation and Inference
- Symbolic Abstractions

**Examples**

- PhysicsNeMo Examples Catalog

**Library Documentation**

**PhysicsNeMo**

- API Reference

**Examples**

- Introductory Examples
- CFD Examples
- Weather and Climate

Diffusion Model for Full-Waveform Inversion (FWI)

## Problem Overview

In the context of geophysics, Full Waveform Inversion (FWI) is a seismic imaging technique that reconstructs subsurface properties, also called velocity model, by fitting the recorded seismic waveform. It underpins a range of applications, including:

- Hydro-carbon exploration and production, where an accurate velocity model guides drilling decisions.
- CO<sub>2</sub> storage, ensuring the integrity of underground reservoirs used for carbon capture and sequestration.
- Global and regional seismology, helping characterise tectonic processes and earthquakes.
- Analogous elastic/acoustic imaging modalities such as medical ultrasound and non-destructive testing.

The present example is tailored to the elastic wave equation in the context of hydro-carbon exploration, but the same framework can be applied to other wave equations and applications.

The following introduces a few key concepts that are essential to FWI in the context of hydro-carbon exploration:

- Velocity model  $\mathbf{x}(r) = [V_p, V_s, \rho]$  – a 3-D image over coordinates  $r = (z, x, y)$ , where  $z$  is the depth, and  $x$  and  $y$  are the surface coordinates. The P-wave velocity is denoted by  $V_p$ , the S-wave velocity by  $V_s$ , and  $\rho$  is the density. The velocity model spans several kilometres and is discretised at metre-scale resolution.

[https://docs.nvidia.com/physicsnemo/latest/physicsnemo/examples/geophysics/diffusion\\_fwi/README.html](https://docs.nvidia.com/physicsnemo/latest/physicsnemo/examples/geophysics/diffusion_fwi/README.html)

