2023 수치상대론 및 중력파 겨울학교
계산 천체물리 경진대회

NVIDIA

AI for Science

Hyungon Ryu | NVAITC Korea

NVIDIA AI TECHNOLOGY CENTER (NVAITC)

Catalyse AI transformation through research-centric integrated engagements

- **GPU Acceleation**
- **AI for Science**
    - DATA DRIVEN APPROACH
    - PINN APPROACH
    - NVIDIA MODULUS

NVIDIA HPC SDK

Download at developer.nvidia.com/hpc-sdk

NVIDIA HPC SDK

DEVELOPMENT

Compilers
- nvcc
- nvc
- nvc++
- nvfortran

Math Libraries
- cuBLAS
- cuTENSOR
- cuSPARSE
- cuSOLVER
- cuFFT
- cuRAND

Communication Libraries
- Open MPI
- NVSHMEM
- NCCL

Programming Models
- Standard C++ & Fortran
- OpenACC & OpenMP
- CUDA

ANALYSIS

Profilers
- Nsight
  - Systems
  - Compute

Debugger
- cuda-gdb
  - Host
  - Device

DEPLOYMENT

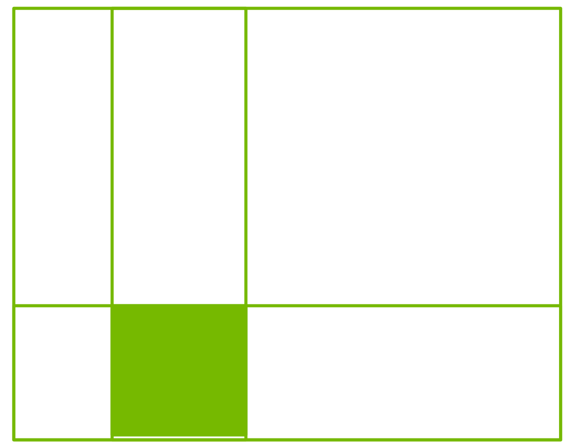
Container
- HPC Container Maker / NVIDIA Container Runtime

Develop for the NVIDIA HPC Platform: GPU, CPU and Interconnect

HPC Libraries | GPU Accelerated C++ and Fortran | Directives | CUDA

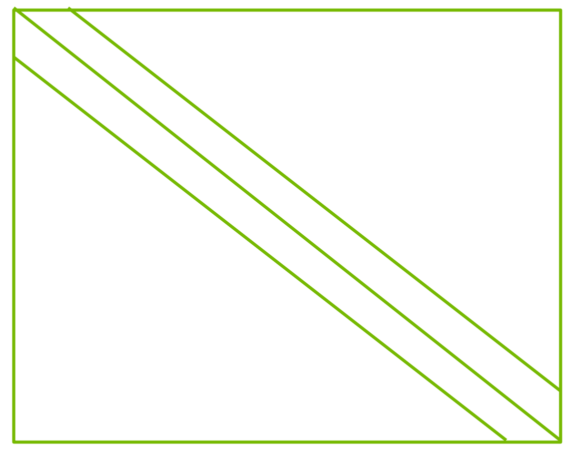
HOW GPU ACCELERATION WORKS

GPU ACCELERATED MATH LIBRARIES



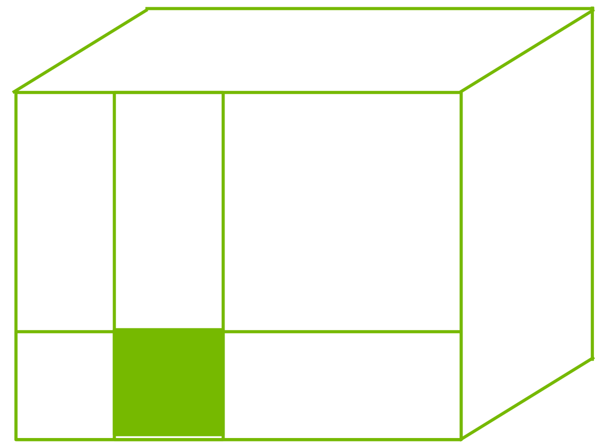
cuBLAS

BF16, TF32 and FP64
Tensor Cores



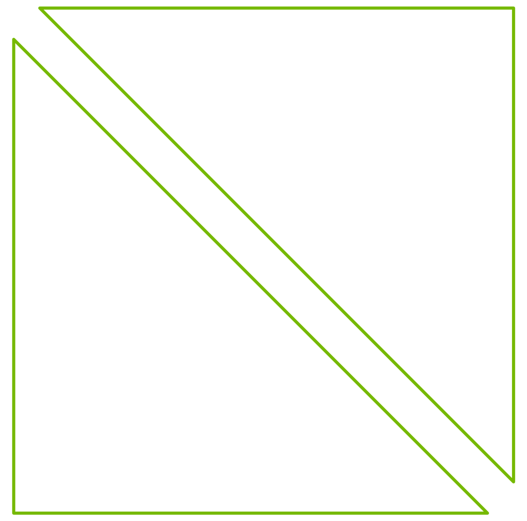
cuSPARSE

Increased memory BW,
Shared Memory & L2



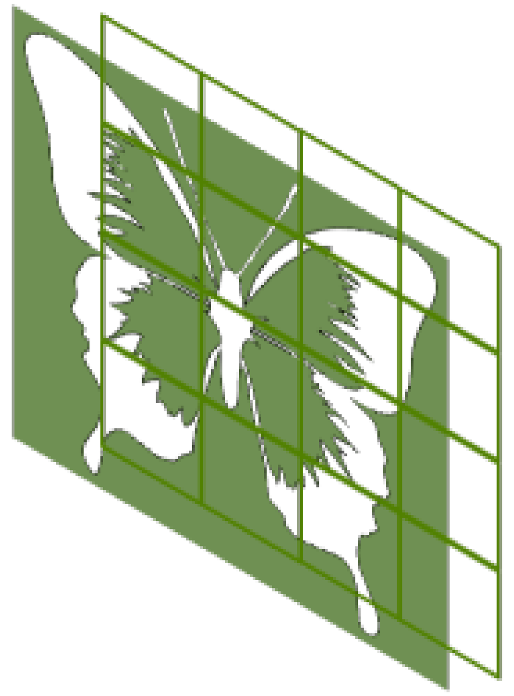
cuTENSOR

BF16, TF32 and FP64
Tensor Cores



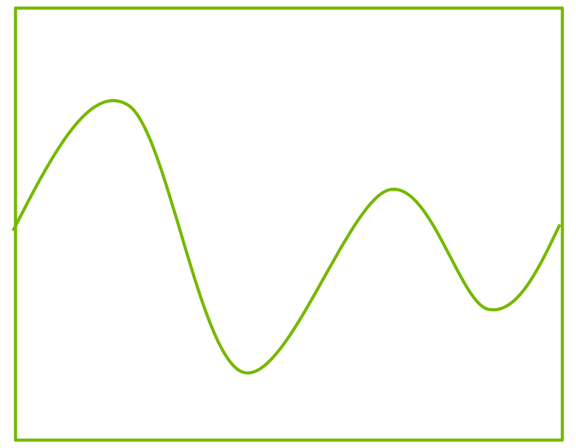
cuSOLVER

BF16, TF32 and FP64
Tensor Cores



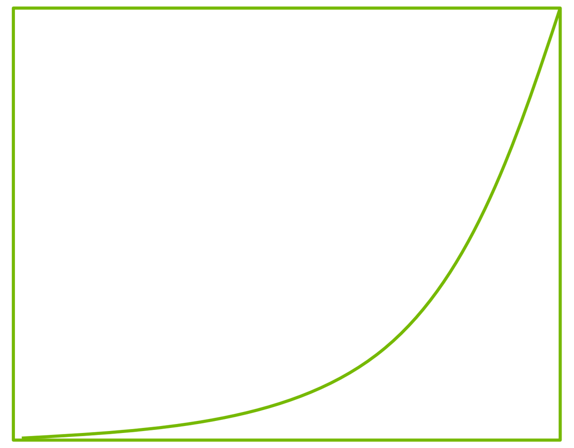
nvJPEG

Hardware Decoder



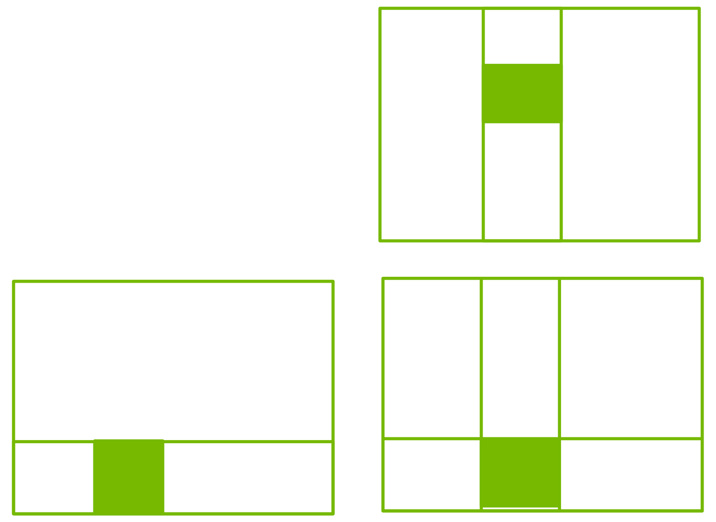
cuFFT

BF16, TF32 and FP64
Tensor Cores



CUDA Math API

Increased memory BW,
Shared Memory & L2



CUTLASS

BF16 & TF32 Support

N-WAYS TO GPU PROGRAMMING

Math Libraries | Standard Languages | Directives | CUDA

```
std::transform(par, x, x+n, y, y,
    [=](float x, float y) {
        return y + a*x;
});
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

GPU Accelerated
C++ and Fortran

```
#pragma acc data copy(x,y)
{

...

std::transform(par, x, x+n, y, y,
    [=](float x, float y) {
        return y + a*x;
});

...

}
```

Incremental Performance
Optimization with Directives

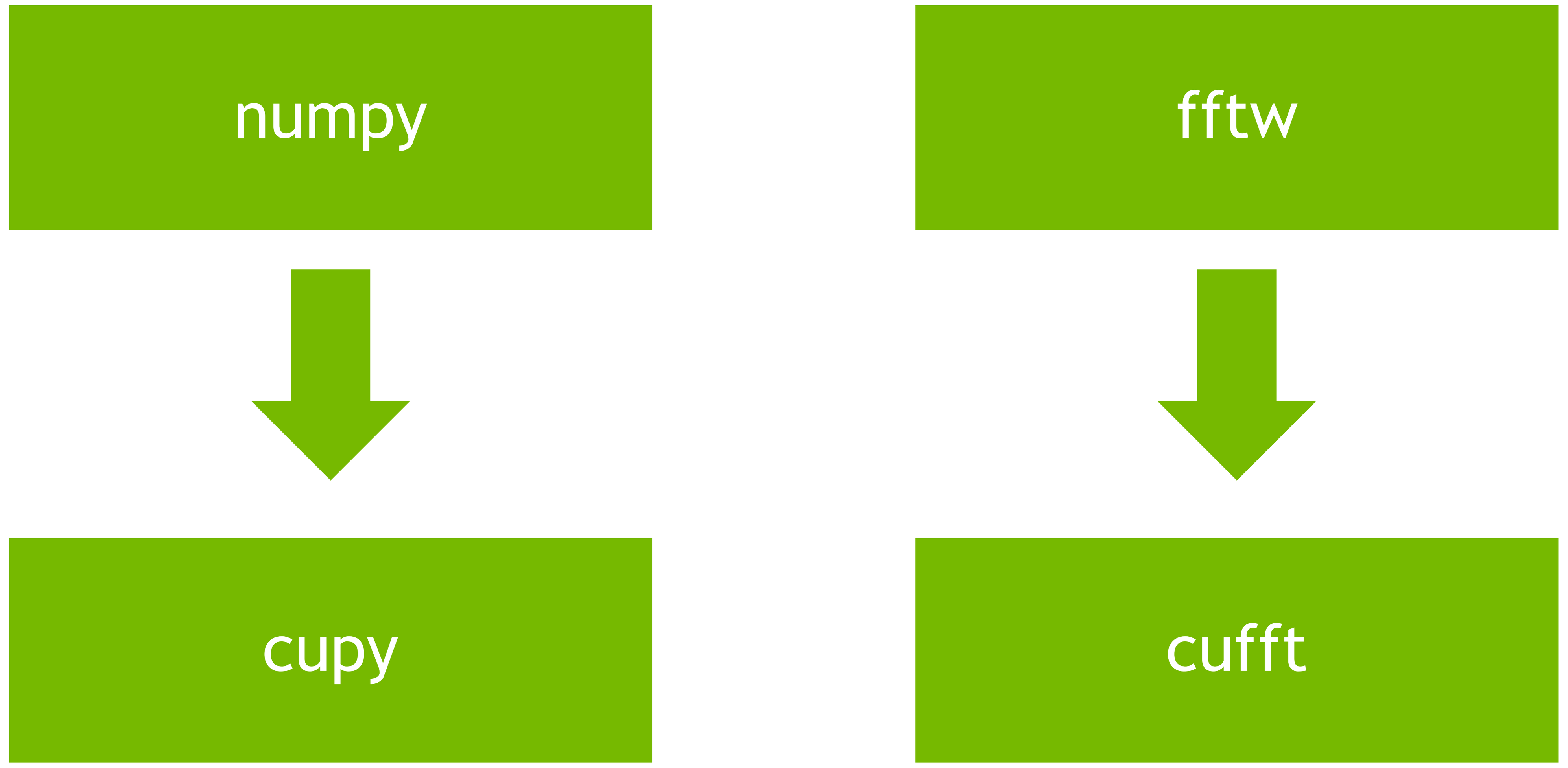
```
__global__
void saxpy(int n, float a,
           float *x, float *y) {
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] += a*x[i];
}

int main(void) {
  cudaMallocManaged(&x, ...);
  cudaMallocManaged(&y, ...);
  ...
  saxpy<<<(N+255)/256,256>>>(...,x, y)
  cudaDeviceSynchronize();
  ...
}
```

Maximize GPU Performance with
CUDA C++/Fortran

GPU Accelerated Math Libraries

CUPY (GPU ACCELERATED PYTHON)

correlation

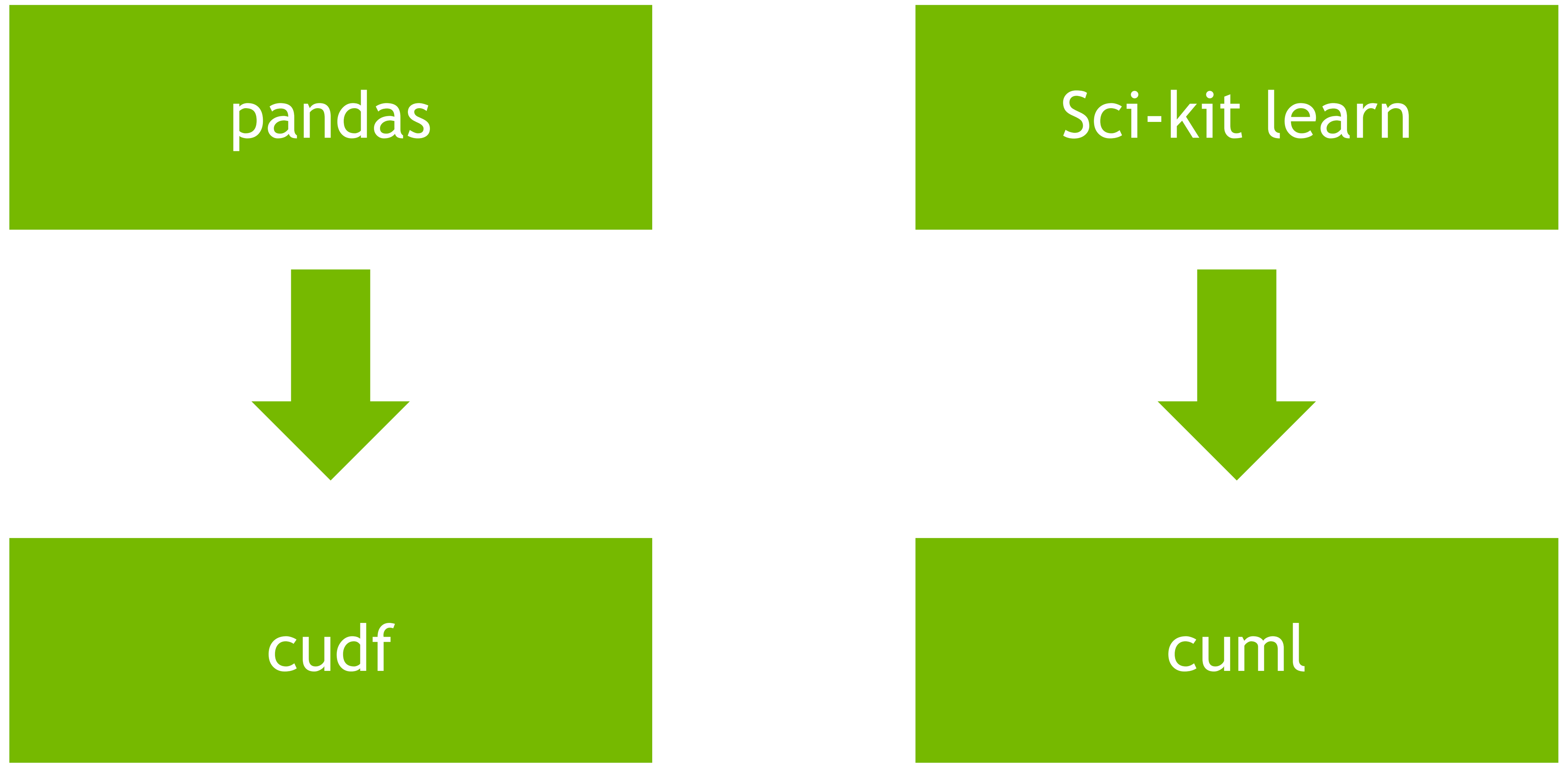


https://colab.research.google.com/drive/1zohf3Y-8g7Sv-2UkmDjIPW-eMMJYtgng?usp=sharing#scrollTo=ZdygwcMmlwH6

```
%%file main_cupy.py

import nvtx
import numpy as np
import cupy as cp
from numpy.random import rand
from cupyx.scipy.fft import rfft, irfft
#from pyfftw.interfaces.numpy_fft import rfft, irfft
import nvtx
import time

from numpy import deg2rad
from h5py import File as h5_File


def haversine_cupy(lon1, lat1, lon2, lat2):
    """
    Return the great circle distance (degree) between two points.
    """
    # convert decimal degrees to radians
    import cupy as cp
    from cupy import deg2rad

    lon1, lat1, lon2, lat2 = deg2rad(lon1), deg2rad(lat1), deg2rad(lon2), deg2rad(lat2)
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    s1 = cp.sin(dlat*0.5)
    s2 = cp.sin(dlon*0.5)
    a =  s1*s1 + cp.cos(lat1) * cp.cos(lat2) * s2 * s2
    c = cp.rad2deg( 2.0 * cp.arcsin(cp.sqrt(a))  )
    return c # degree
```

RAPIDS

GPU accelerated Data Science



Scikit-learn Model

Fit

In [ ]:

```
%%time
knn_sk = skNearestNeighbors(algorithm="brute",
                            n_jobs=-1)
knn_sk.fit(host_data)
```

In [ ]:

```
%%time
D_sk, I_sk = knn_sk.kneighbors(host_data[:n_query], n_neighbors)
```

cuML Model

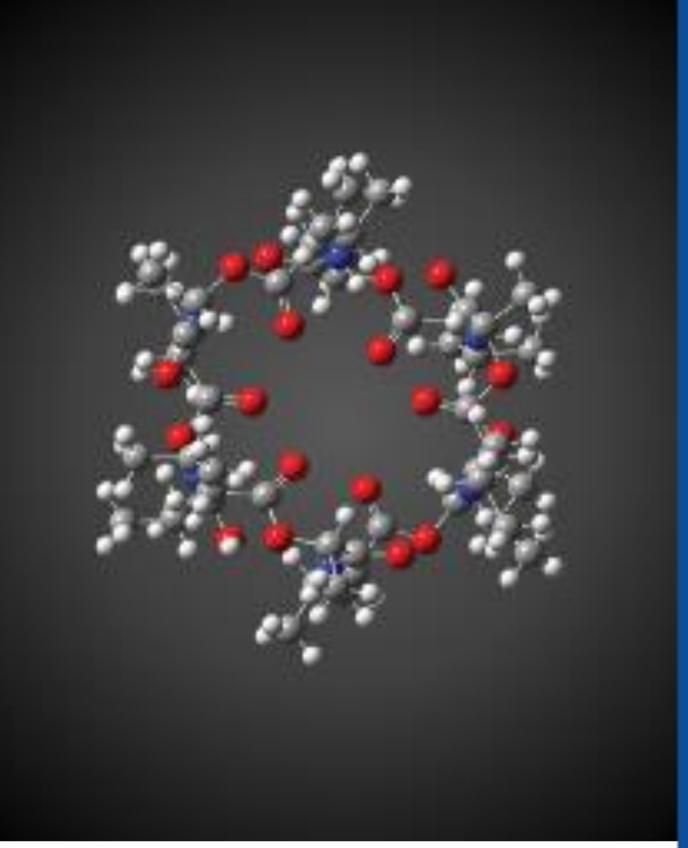
Fit

In [ ]:

```
%%time
knn_cuml = cuNearestNeighbors()
knn_cuml.fit(device_data)
```

In [ ]:

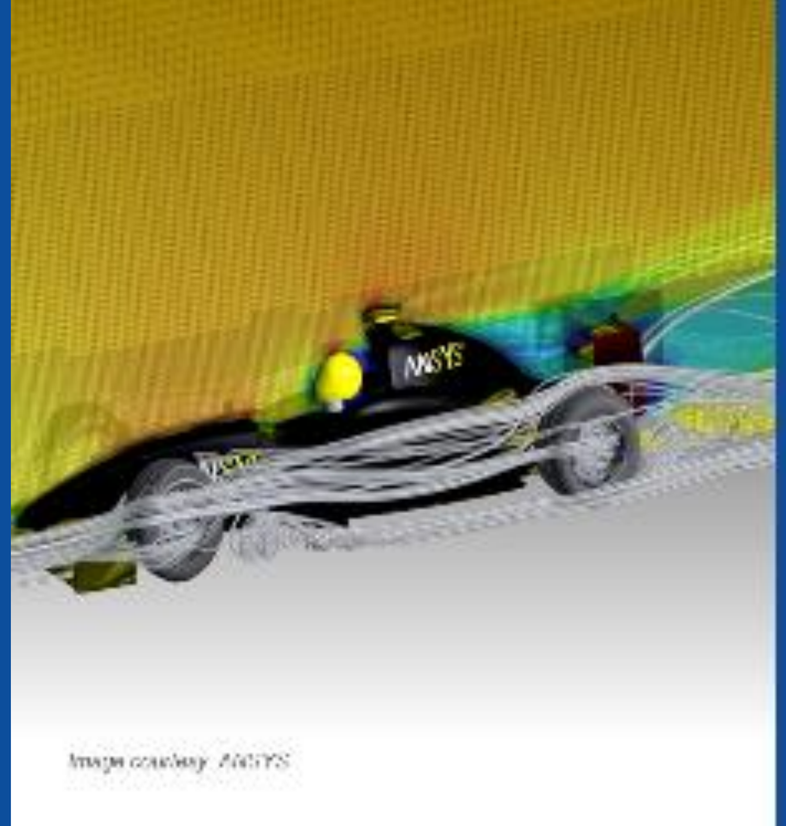
```
%%time
D_cuml, I_cuml = knn_cuml.kneighbors(device_data[:n_query], n_neighbors)
```



GAUSSIAN 16

Mike Frisch, Ph.D.
President and CEO
Gaussian, Inc.

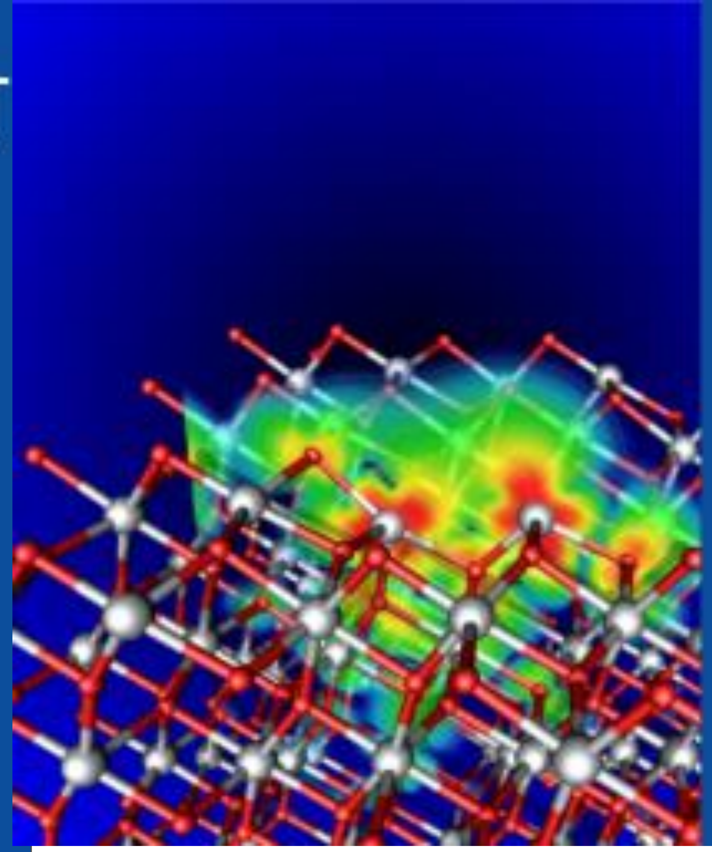
"Using OpenACC allowed us to continue development of our fundamental algorithms and software capabilities simultaneously with the GPU-related work. In the end, we could use the same code base for SMP, cluster/network and GPU parallelism. PGI's compilers were essential to the success of our efforts."



ANSYS FLUENT

Sunil Sathe
Lead Software Developer
ANSYS Fluent

"We've effectively used OpenACC for heterogeneous computing in ANSYS Fluent with impressive performance. We're now applying this work to more of our models and new platforms."



VASP

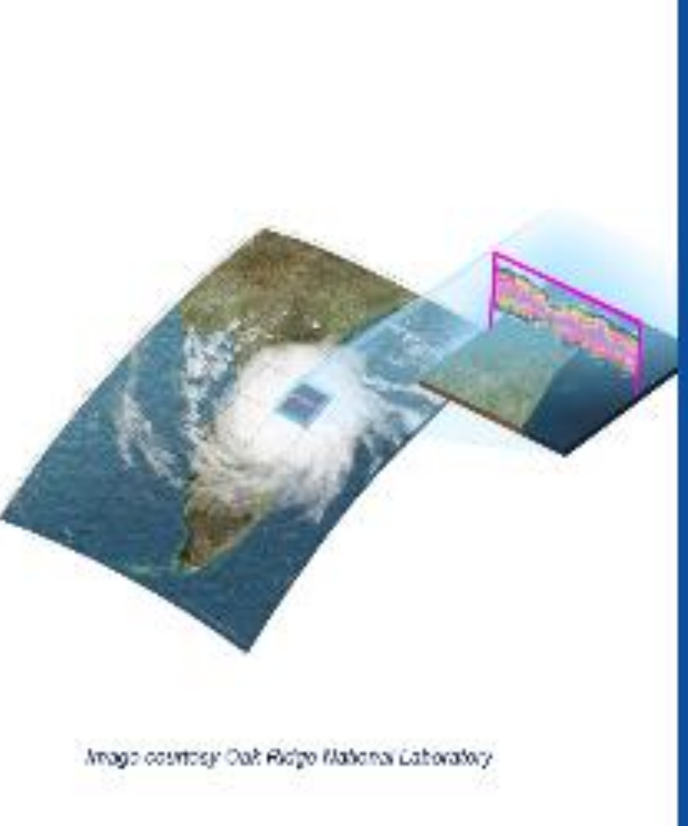
Prof. Georg Kresse
Computational Materials Physics
University of Vienna

"For VASP, OpenACC is *the* way forward for GPU acceleration. Performance is similar and in some cases better than CUDA C, and OpenACC dramatically decreases GPU development and maintenance efforts. We're excited to collaborate with NVIDIA and PGI as an early adopter of CUDA Unified Memory."

COSMO

Dr. Oliver Fuhrer
Senior Scientist
Meteoswiss

"OpenACC made it practical to develop for GPU-based hardware while retaining a single source for almost all the COSMO physics code."



E3SM

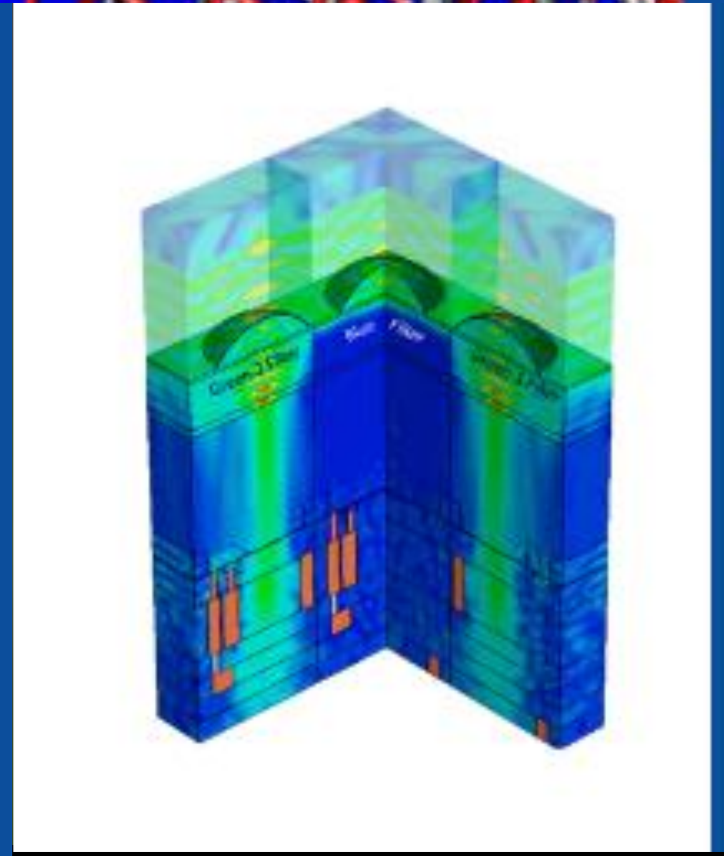
Mark A. Taylor
Multiphysics Applications
Sandia

"The CAAR project provided us with early access to Summit hardware and access to PGI compiler experts. Both of these were critical to our success. PGI's OpenACC support remains the best available and is competitive with much more intrusive programming model approaches."

NUMECA FINE/Open

David Gutzwiller
Lead Software Developer
NUMECA

"Porting our unstructured C++ CFD solver FINE/Open to GPUs using OpenACC would have been impossible two or three years ago, but OpenACC has developed enough that we're now getting some really good results."



SYNOPSYS

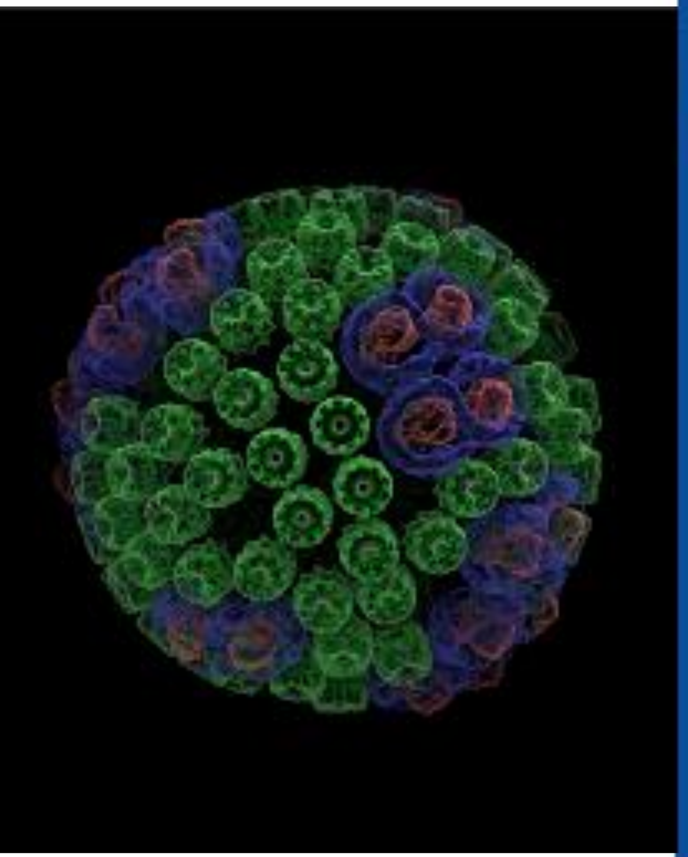
Dr. Lutz Schneider
Senior R&D Engineer
Synopsys Inc.

"Using OpenACC, we've GPU-accelerated the Synopsys TCAD Sentaurus Device EMW simulator to speed up optical simulations of image sensors. GPUs are key to improving simulation throughput in the design of advanced image sensors."

MPAS-A

Richard Loft
Director, Technology Development
NCAR

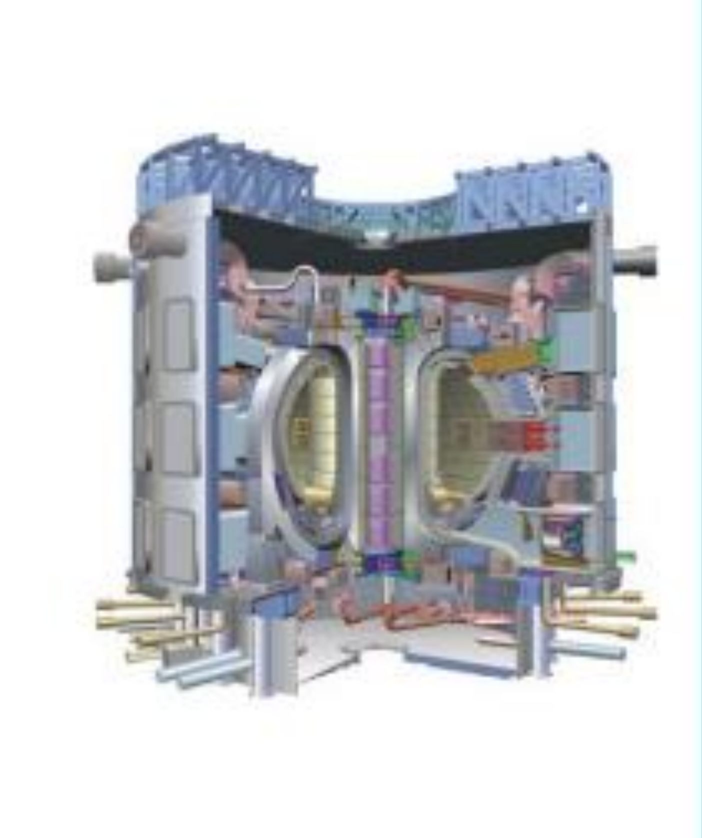
"Our team has been evaluating OpenACC as a pathway to performance portability for the Model for Prediction (MPAS) atmospheric model. Using this approach on the MPAS dynamical core, we have achieved performance on a single P100 GPU equivalent to 2.7 dual socketed Intel Xeon nodes on our new Cheyenne supercomputer."



VMD

John Stone
Senior Research Programmer
Beckman Institute
University of Illinois

"Due to Amdahl's law, we need to port more parts of our code to the GPU if we're going to speed it up. But the sheer number of routines poses a challenge. OpenACC directives give us a low-cost approach to getting at least some speed-up out of these second-tier routines. In many cases it's completely sufficient because with the current algorithms, GPU performance is bandwidth-bound."



GTC

Zhihong Lin
Professor and Principal Investigator
UC Irvine

"Using OpenACC our scientists were able to achieve the acceleration needed for integrated fusion simulation with a minimum investment of time and effort in learning to program GPUs."

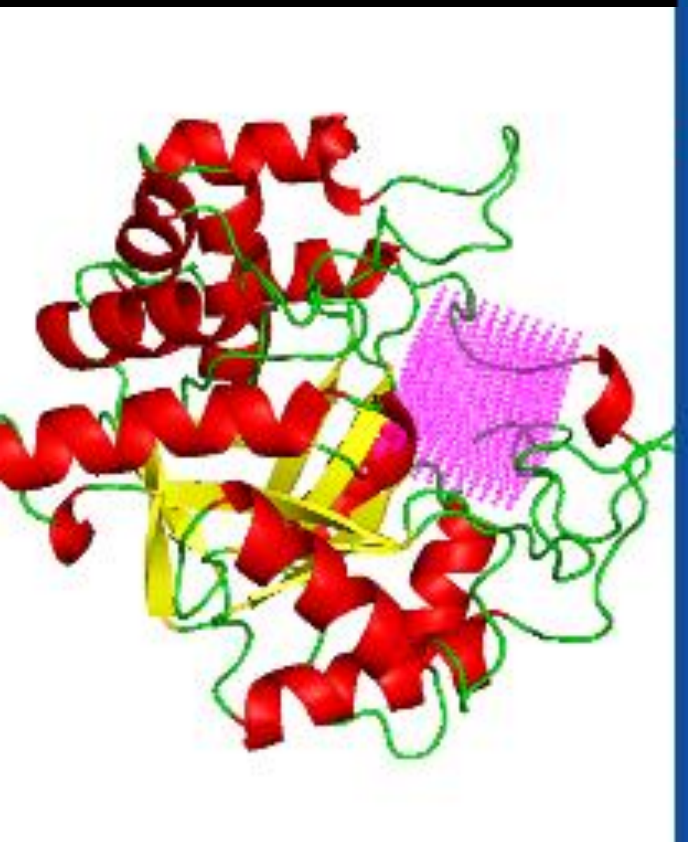
OpenACC

More Science. Less Programming

GAMERA

Takuma Yamaguchi, Kohei Fujita, Tsuyoshi Ichimura, Muneo Hori, Lalith Wijerathne,
The University of Tokyo

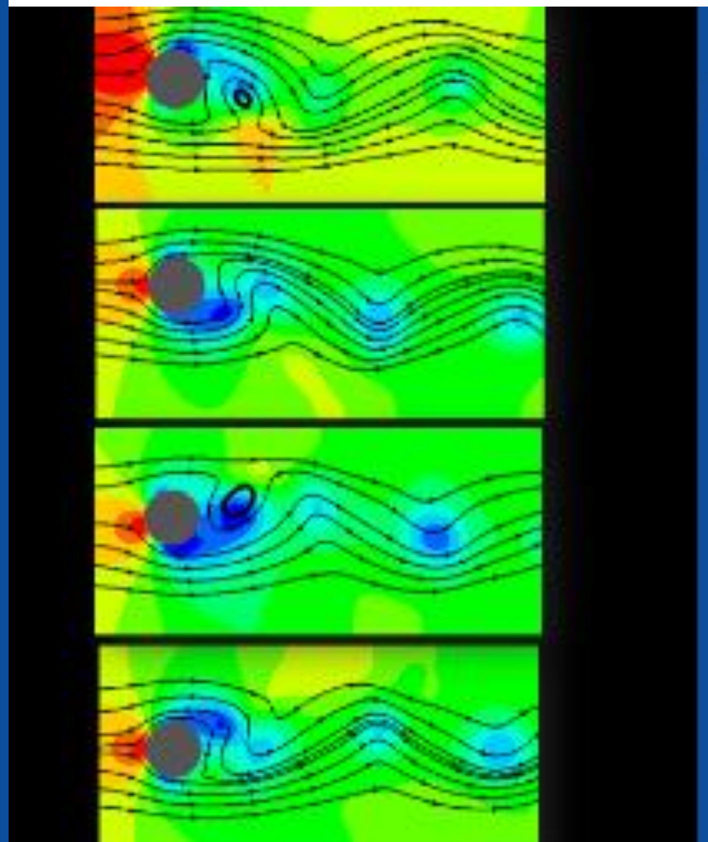
"With OpenACC and a compute node based on NVIDIA's Tesla P100 GPU, we achieved more than a 14X speed up over a K Computer node running our earthquake disaster simulation code"



SANJEEVINI

Abhilash Jayaraj
Project Scientist
Indian Institute of Technology
New Delhi

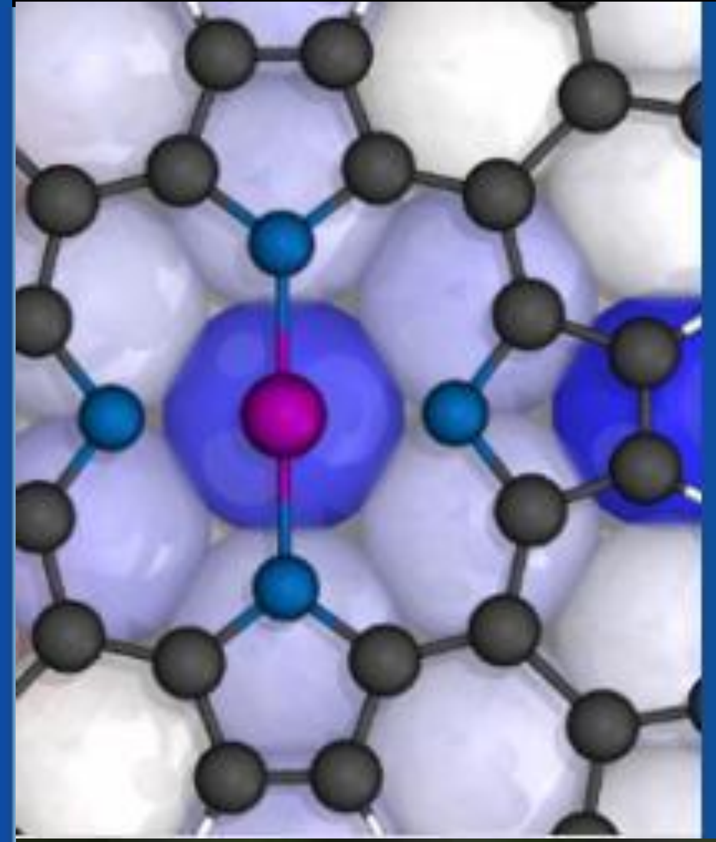
"In an academic environment maintenance and speedup of existing codes is a tedious task. OpenACC provides a great platform for computational scientists to accomplish both tasks without involving a lot of efforts or manpower in speeding up the entire computational task."



IBM-CFD

Somnath Roy
Assistant Professor
Mechanical Engineering Department
Indian Institute of Technology Kharagpur

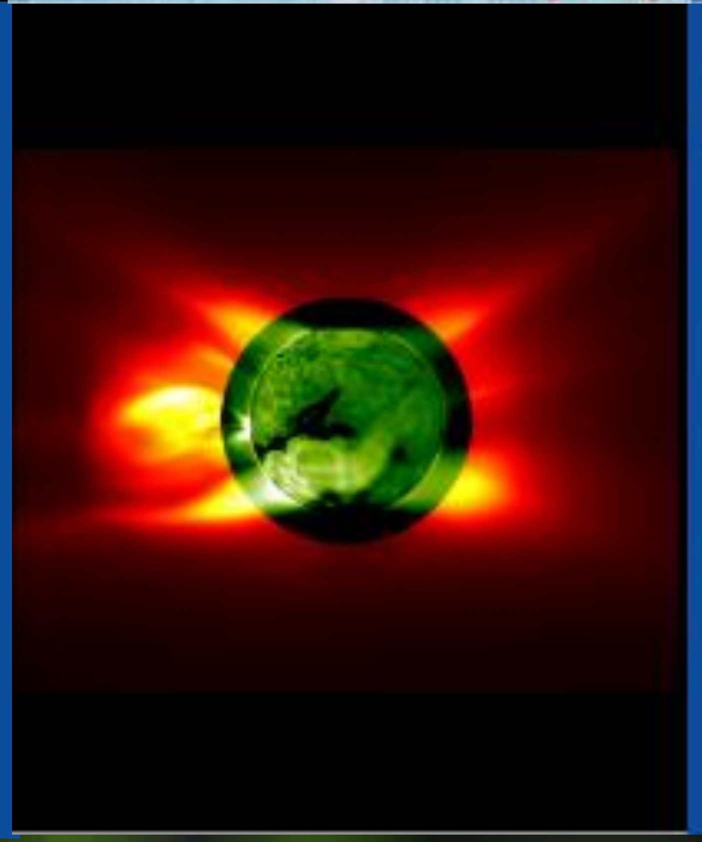
"OpenACC can prove to be a handy tool for computational engineers and researchers to obtain fast solution of non-linear dynamics problem. In immersed boundary incompressible CFD, we have obtained order of magnitude reduction in computing time by porting several components of our legacy codes to GPU. Especially the routines involving search algorithm and matrix solvers have been well-accelerated to improve the overall scalability of the code."



PWscf (Quantum ESPRESSO)

Filippo Spiga
Senior Contributor
Quantum ESPRESSO group

"CUDA Fortran gives us the full performance potential of the CUDA programming model and NVIDIA GPUs. While leveraging the potential of explicit data movement, !$CUF KERNELS directives give us productivity and source code maintainability. It's the best of both worlds."



MAS

Ronald M. Caplan
Computational Scientist
Predictive Science Inc.

"Adding OpenACC into MAS has given us the ability to migrate medium-sized simulations from a multi node CPU cluster to a single multi-GPU server. The implementation yielded a portable single-source code for both CPU and GPU runs. Future work will add OpenACC to the remaining model features, enabling GPU-accelerated realistic solar storm modeling."

OpenACC Directives

Manage Data Movement

Initiate Parallel Execution

Optimize Loop Mappings

```
#pragma acc data copyin(a,b) copyout(c)
{
  ...
  #pragma acc parallel
  {
  #pragma acc loop gang vector
      for (i = 0; i < n; ++i) {
          c[i] = a[i] + b[i];
          ...
      }
  }
  ...
}
```

- Incremental
- Single source
- Interoperable
- Performance portable
- CPU, GPU, Manycore

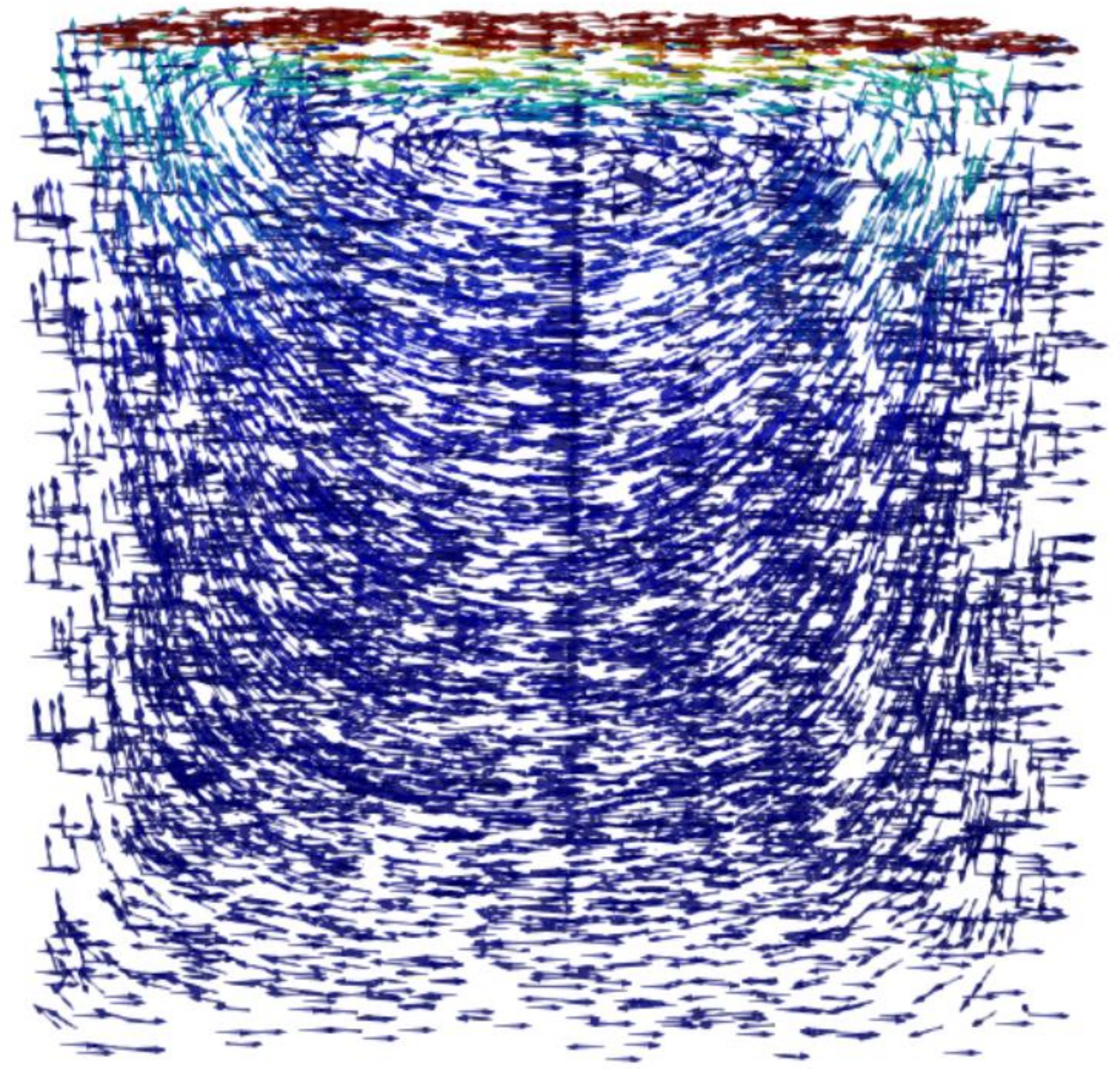
OpenACC

Directives for Accelerators

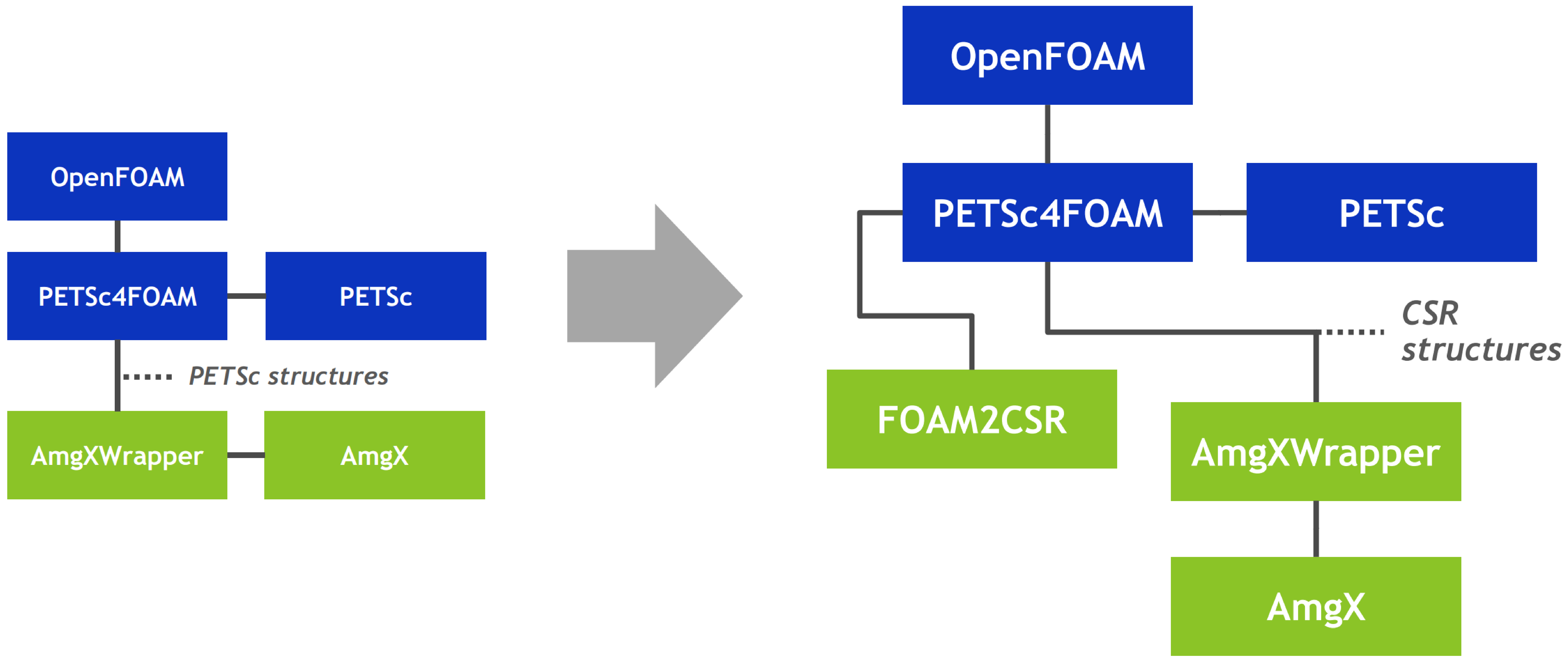
GPU Accelerated CFD

OpenFOAM + PETSc + AmgX

- Early results of the AmgX solver library used to accelerate the OpenFOAM pressure solve on GPUs achieved ~4x to ~8x speedups



Lid Driven cavity
(M, 200x200x200, 20 steps) solution,
accelerated with AmgX

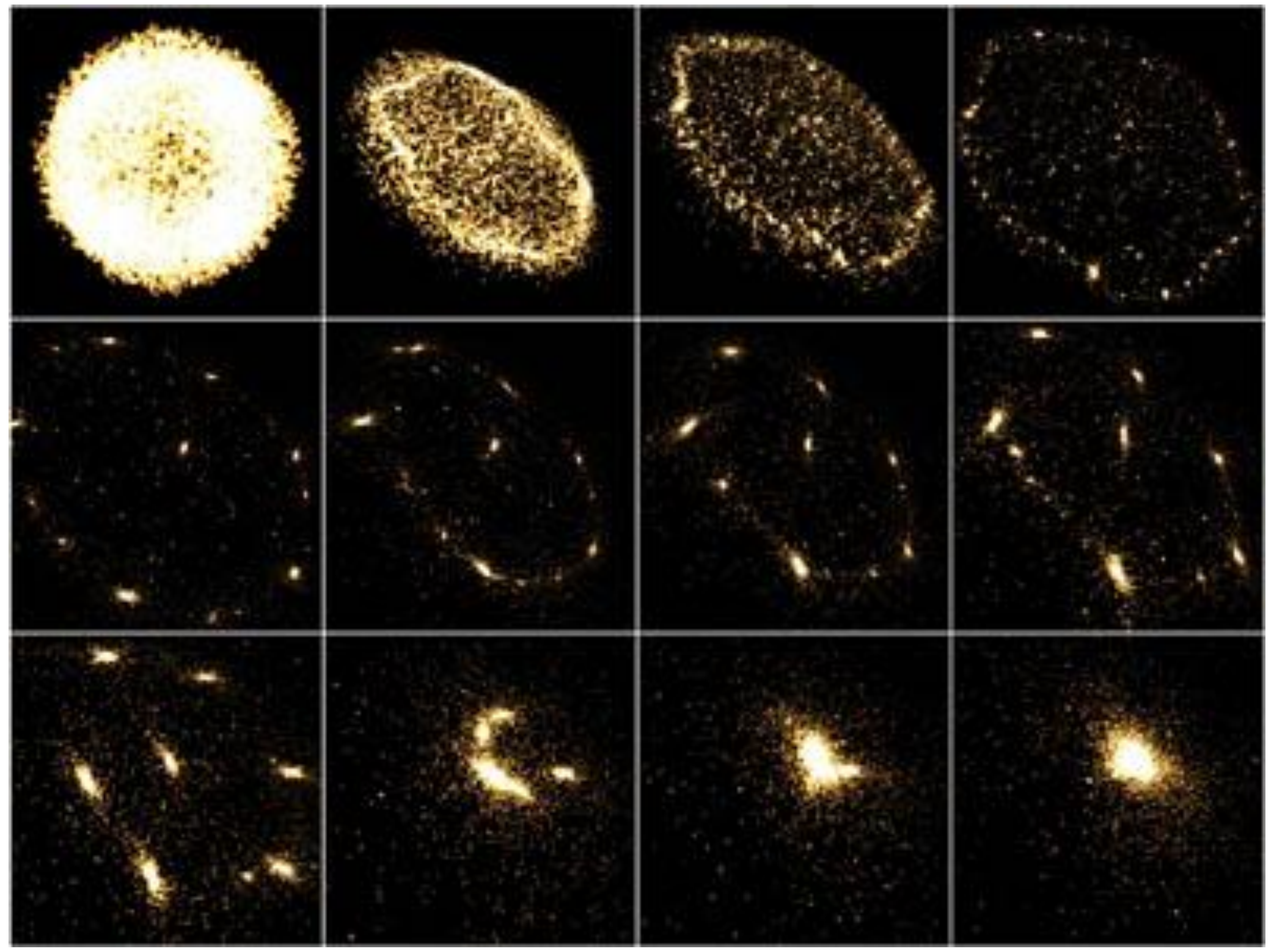


NBODY SIMULATION

MD simulation COSMOS

```
!pair calculation
call nvtxStartRange("Pair Calculation")
do iconf=1,nframes
   if (mod(iconf,1).eq.0) print*,iconf
   do i=1,natoms
      do j=1,natoms
         dx=x(iconf,i)-x(iconf,j)
         dy=y(iconf,i)-y(iconf,j)
         dz=z(iconf,i)-z(iconf,j)

         dx=dx-nint(dx/xbox)*xbox
         dy=dy-nint(dy/ybox)*ybox
         dz=dz-nint(dz/zbox)*zbox

         r=dsqrt(dx**2+dy**2+dz**2)
         ind=int(r/del)+1
         !if (ind.le.nbin) then
         if(r<cut)then
            g(ind)=g(ind)+1.0d0
         endif
      enddo
   enddo
enddo
```

```
!pair calculation
call nvtxStartRange("Pair Calculation")
do iconf=1,nframes
   if (mod(iconf,1).eq.0) print*,iconf
   !$acc parallel loop
   do i=1,natoms
      do j=1,natoms
         dx=x(iconf,i)-x(iconf,j)
         dy=y(iconf,i)-y(iconf,j)
         dz=z(iconf,i)-z(iconf,j)

         dx=dx-nint(dx/xbox)*xbox
         dy=dy-nint(dy/ybox)*ybox
         dz=dz-nint(dz/zbox)*zbox

         r=dsqrt(dx**2+dy**2+dz**2)
         ind=int(r/del)+1
         if(r<cut)then
            !$acc atomic
            g(ind)=g(ind)+1.0d0
         endif
      enddo
   enddo
enddo
```



AI FOR SCIENCE[DATA DRIVEN APPROACH]

LLM(LARGE LANGUAGE MODEL)

420 node DGX-1(8EA A100)

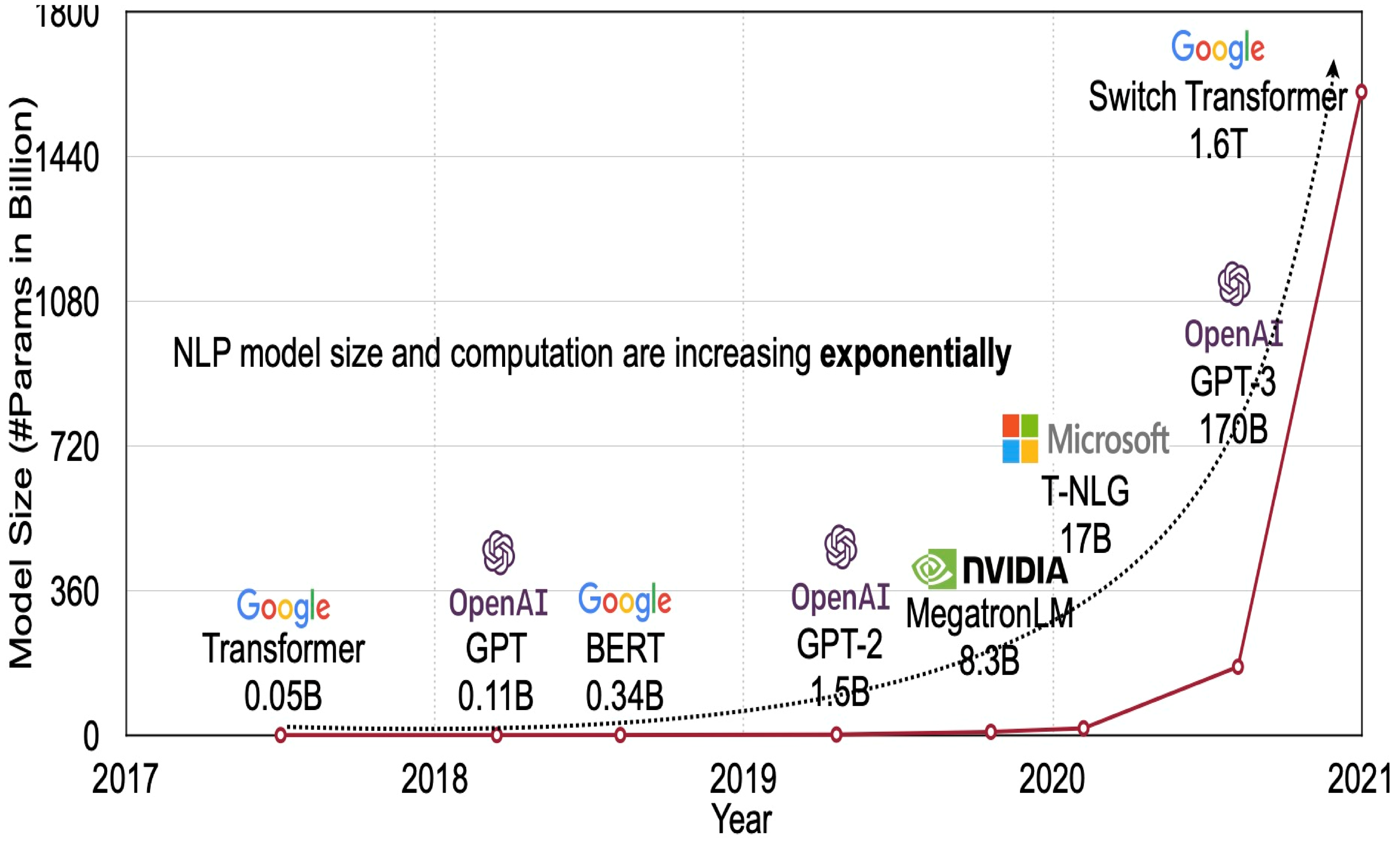


Image from https://hanlab.mit.edu/projects/efficientnlp_old/

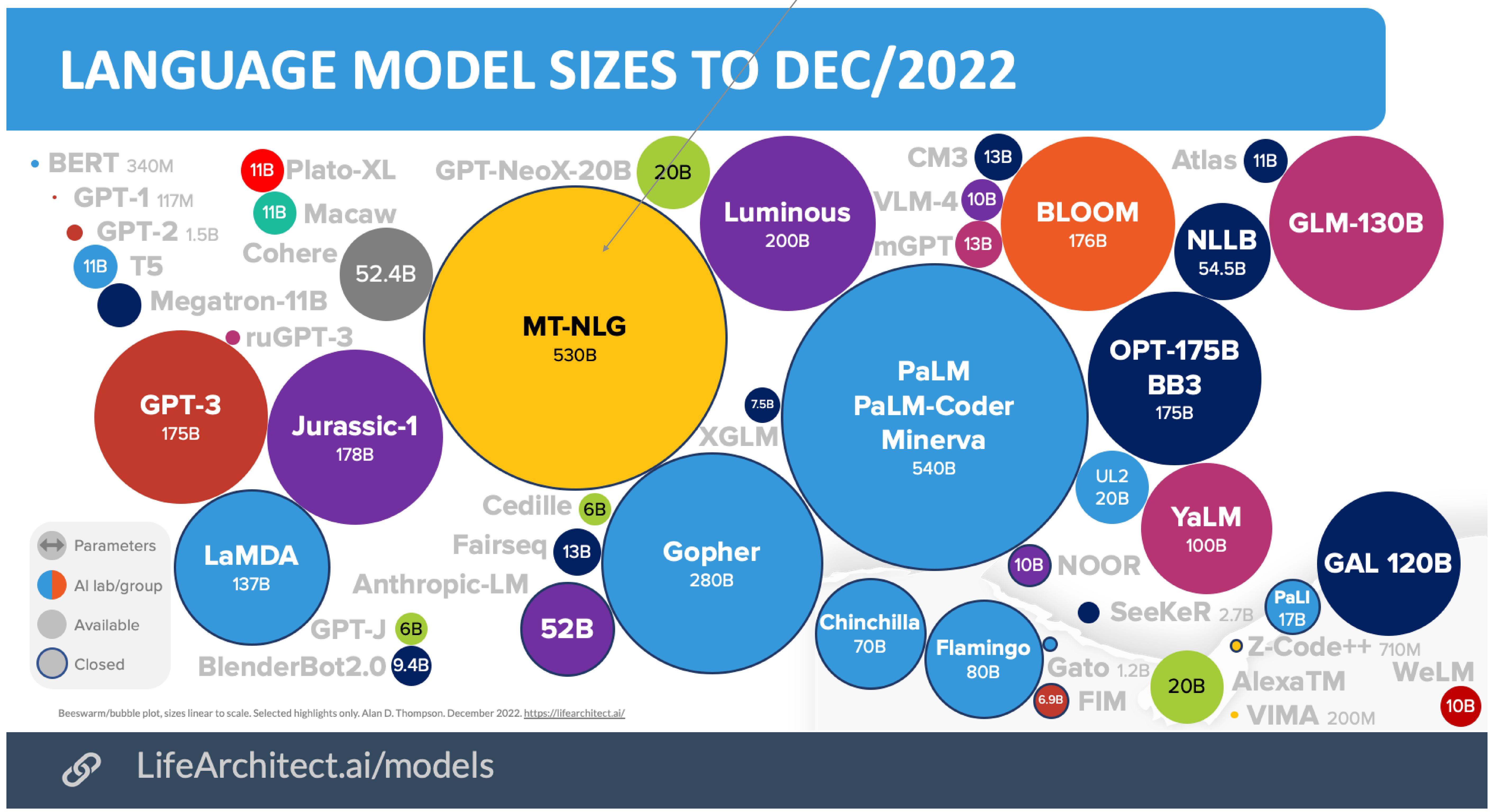
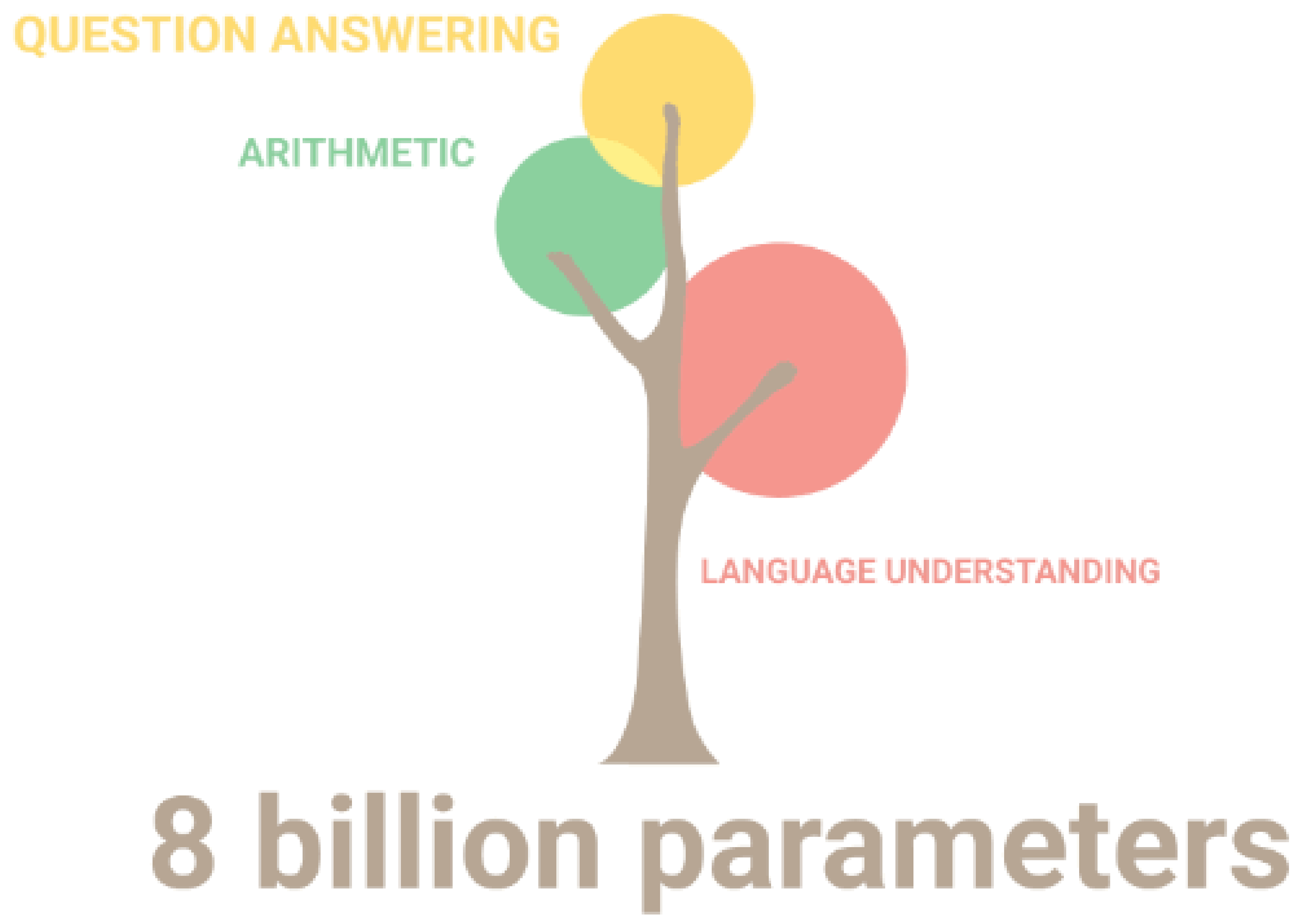


Image from https://lifearchitect.ai/models/

MODEL CAPABILITIES WITH SCALES



Compute Resource

4 Epochs

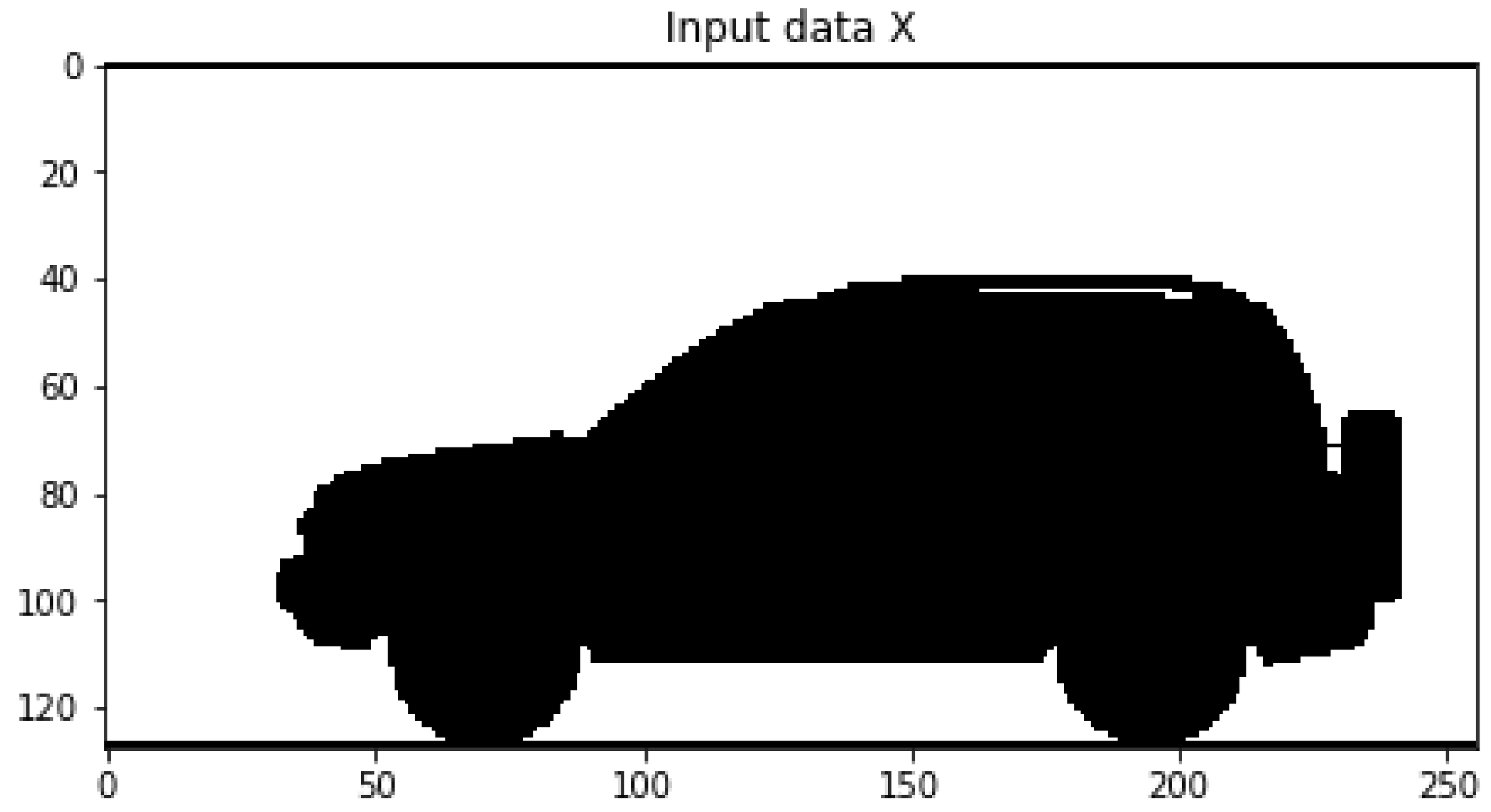
Model Param

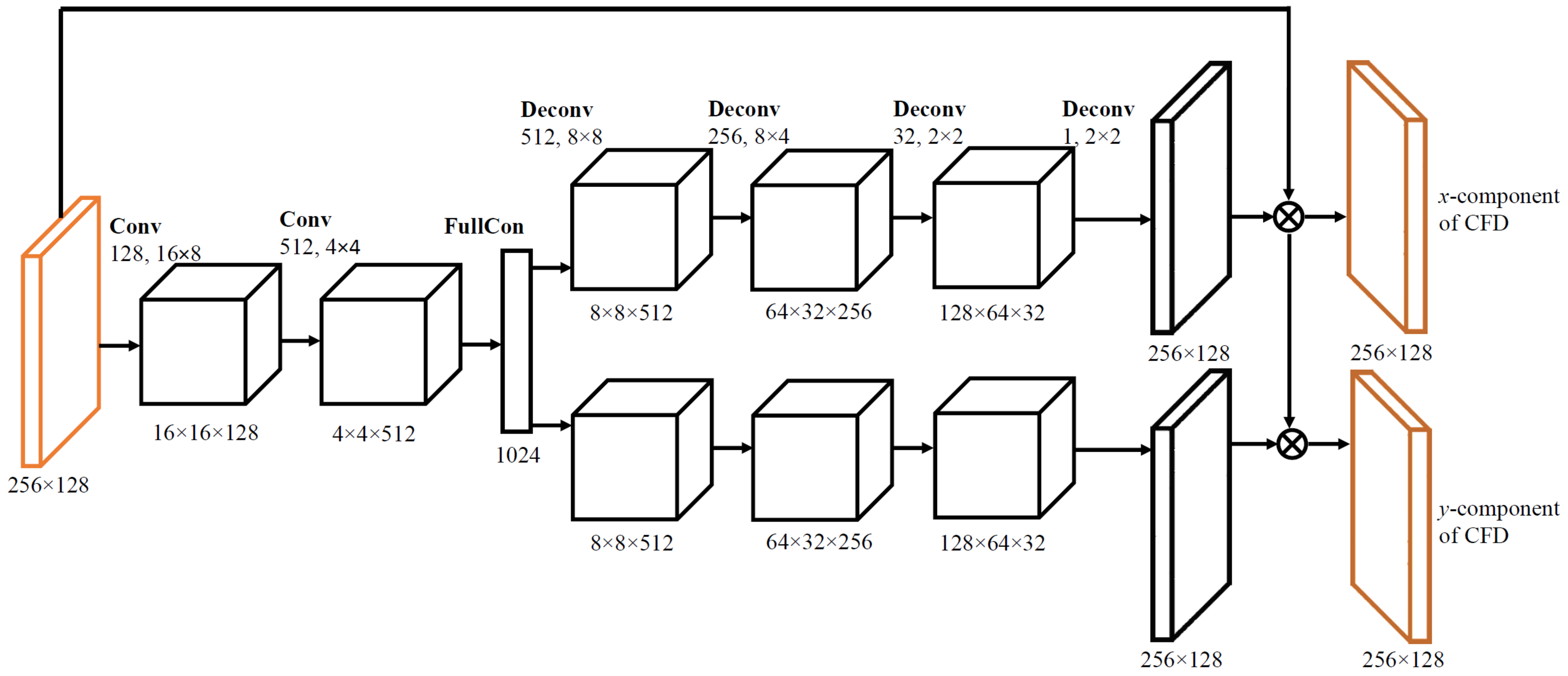
DataToken

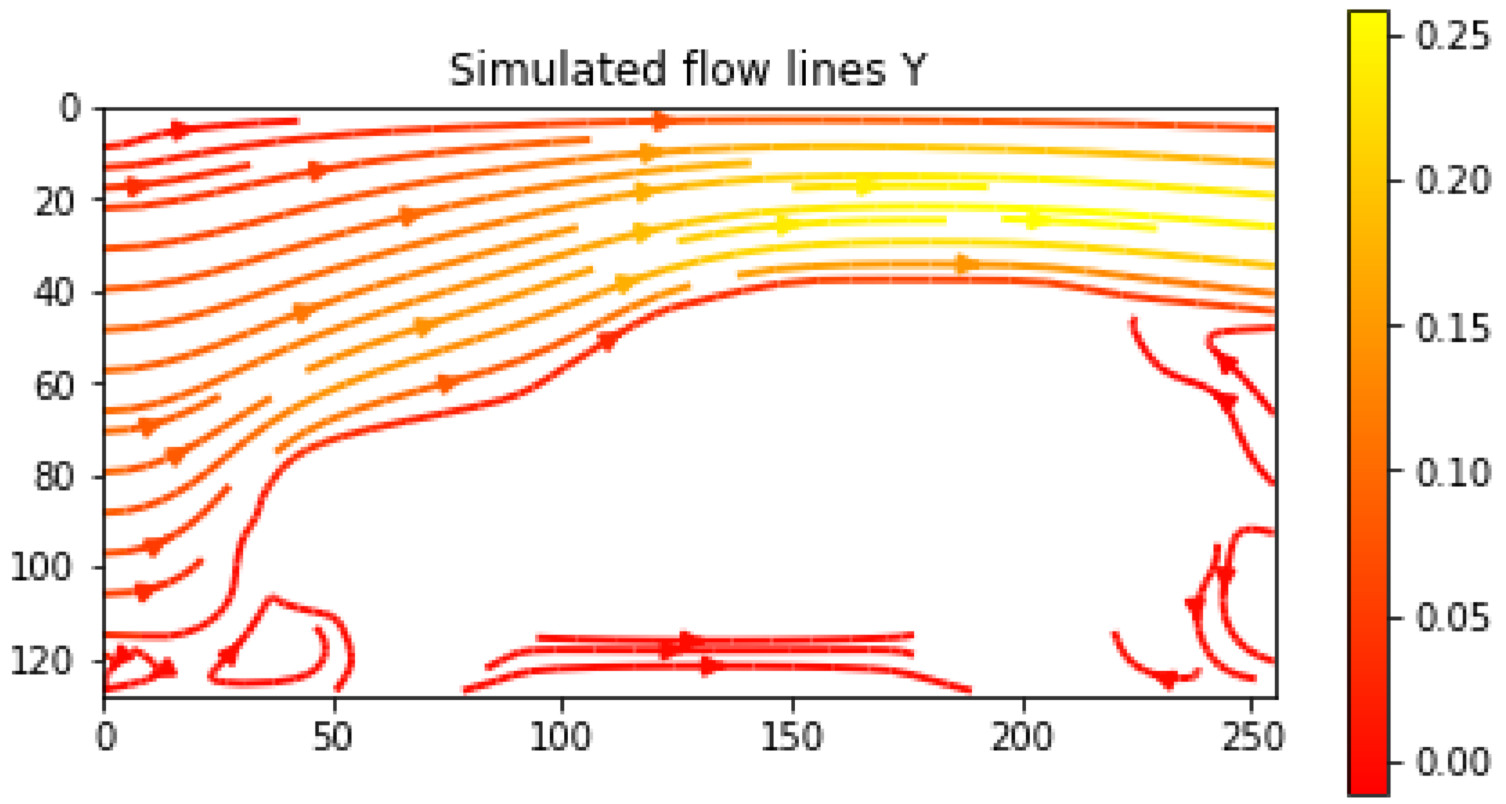
NVIDIA

2d Steady State Flow with Neural Network

Xiaoxiao Guo, Wei Li, Francesco Iorio, Convolutional Neural Networks for Steady Flow Approximation , ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2016
https://www.autodeskresearch.com/publications/convolutional-neural-networks-steady-flow-approximation

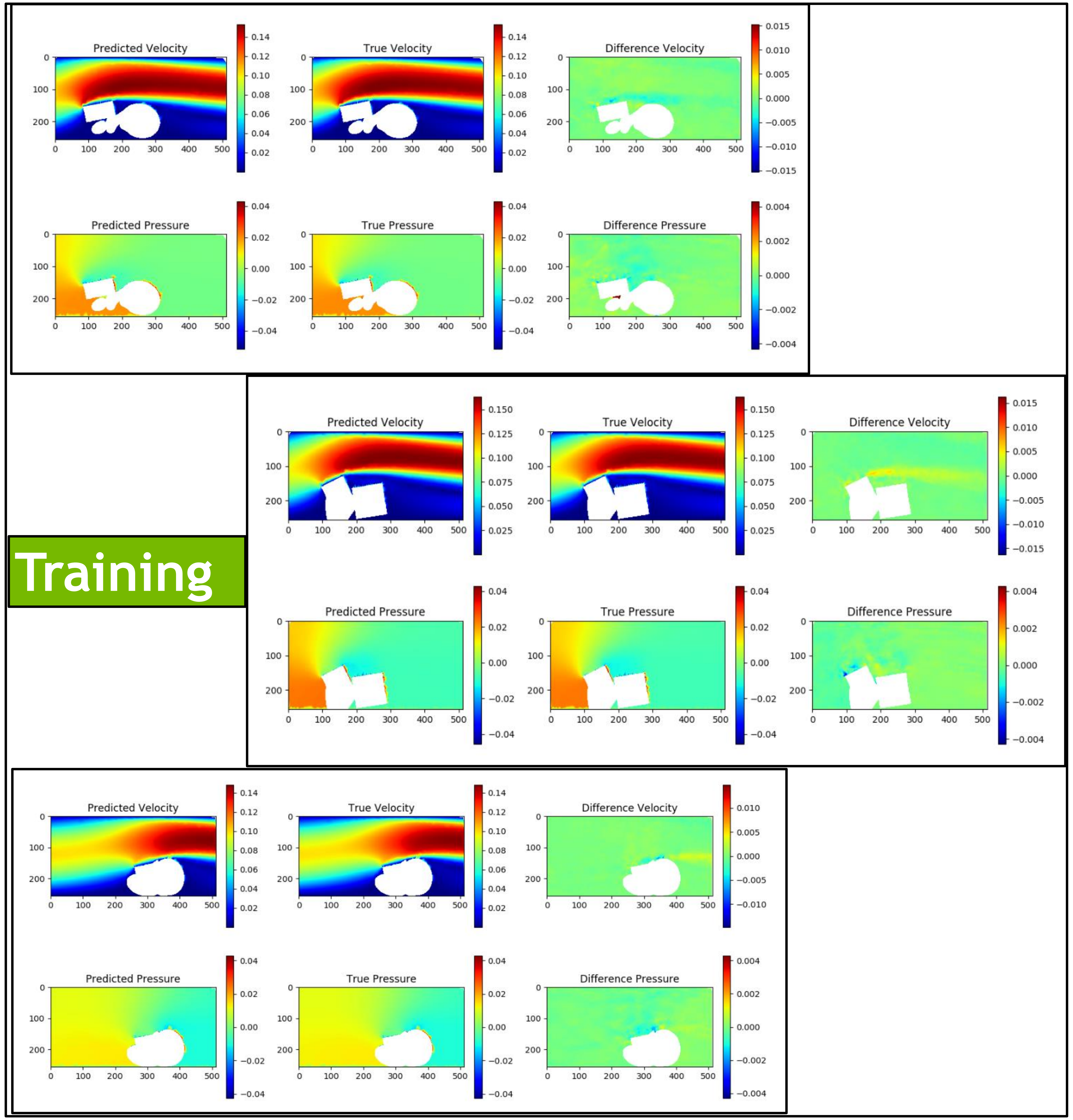




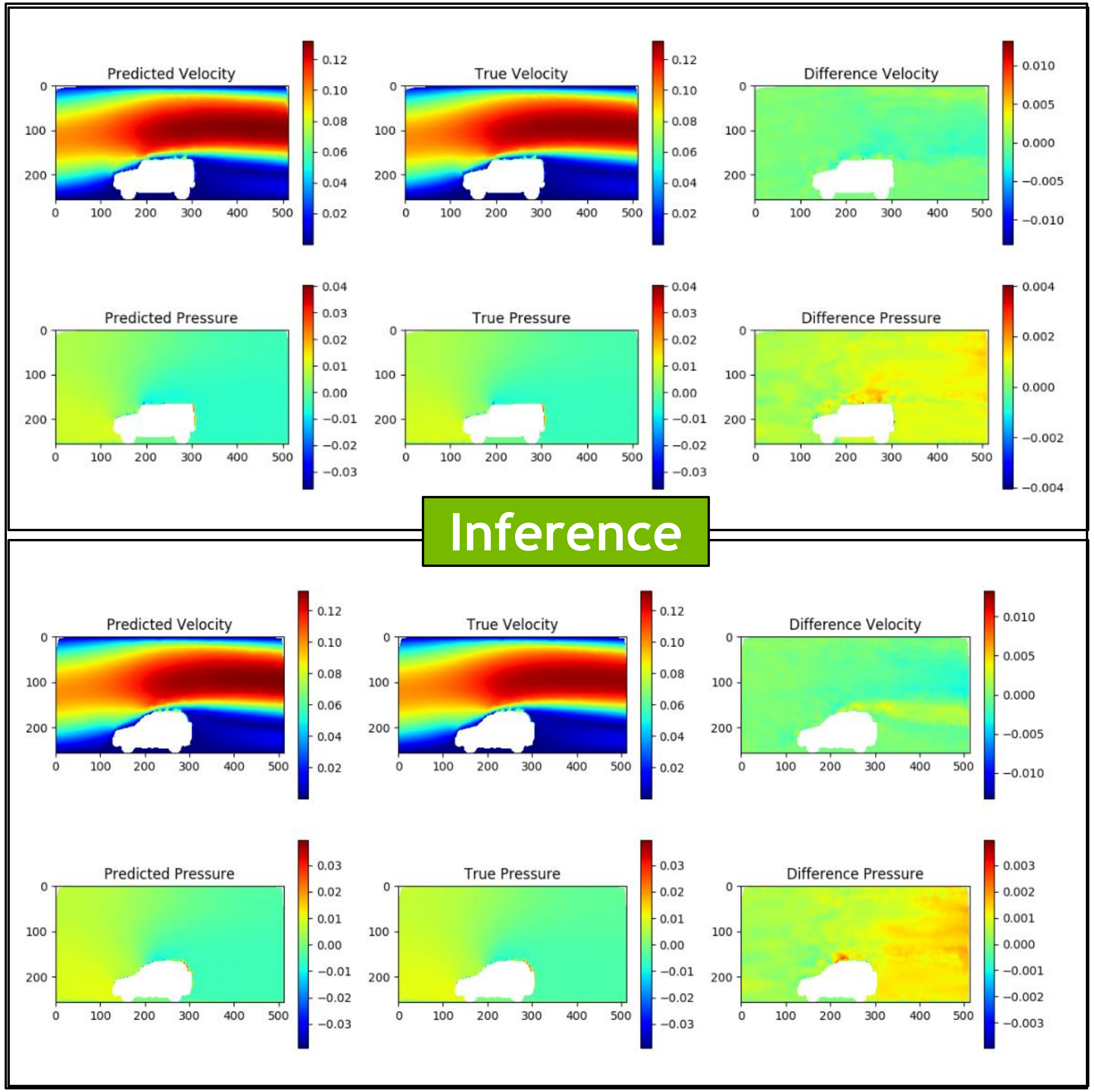


Pair of (2D domain,
Simulated CFD flow)

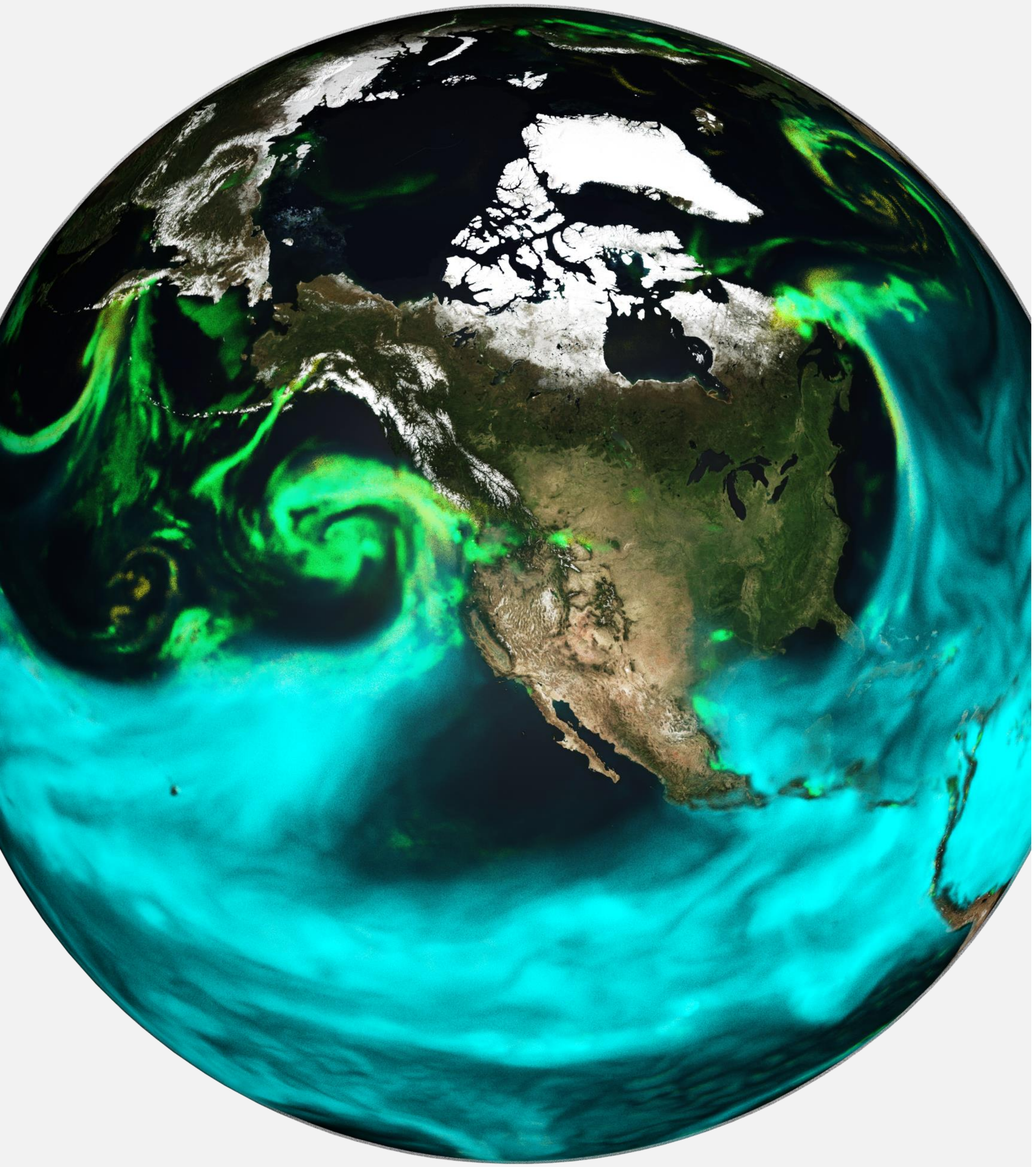
AUTOMOTIVE AERODYNAMICS



Training



Inference



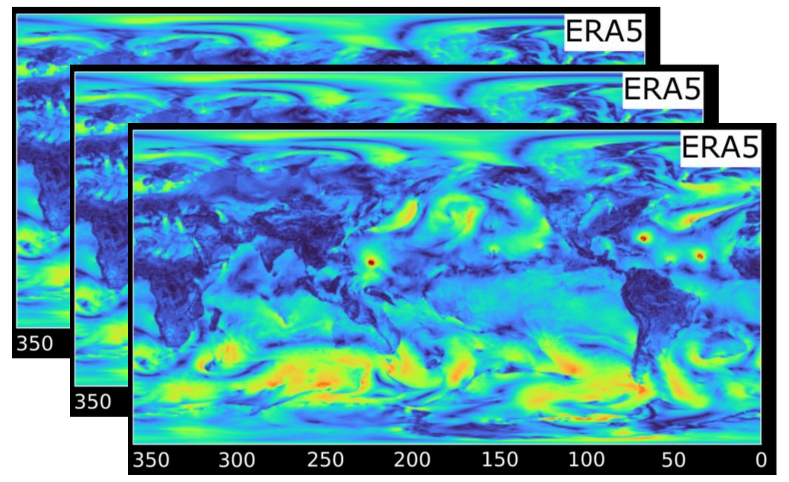
EARTH-2 BEGAN BY EXPLORING DATA-DRIVEN WEATHER PREDICTION

- **FourCastNet**
  - Scope — Global, Medium Range
  - Model Type — Full-Model AI Surrogate
  - Architecture — AFNO (Adaptive Fourier Neural Op.)
  - Resolution: — 25km
  - Training Data: — ERA5 Reanalysis
  - Initial Condition — GFS / UFS
  - Inference Time — 0.25 sec (2-week forecast)
  - Speedup vs NWP — $O(10^4\text{-}10^5)$
  - Power Savings — $O(10^4)$

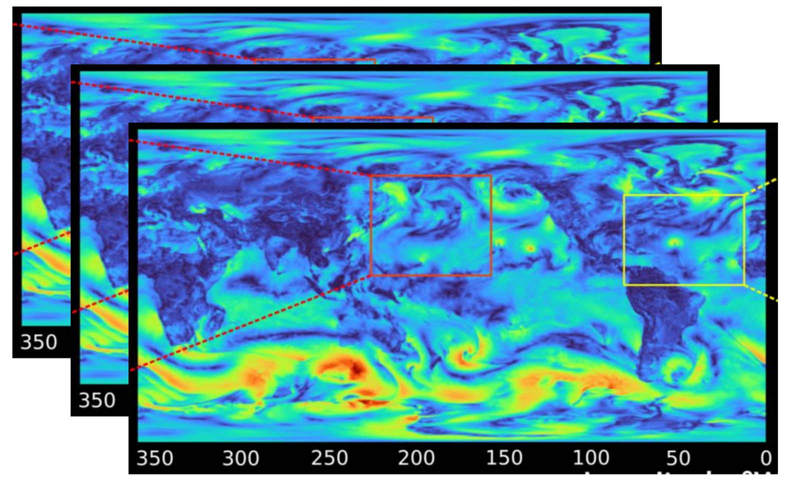
FourCastNet

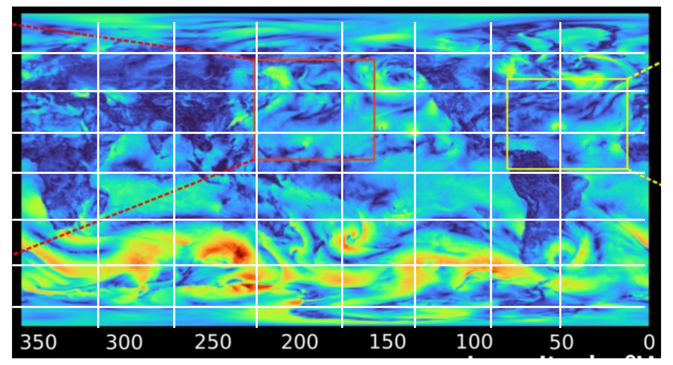
Pair of (input, GT)

(GT : K + 6hr)
20 variables



(input : K)
20 variables





8x8 patch

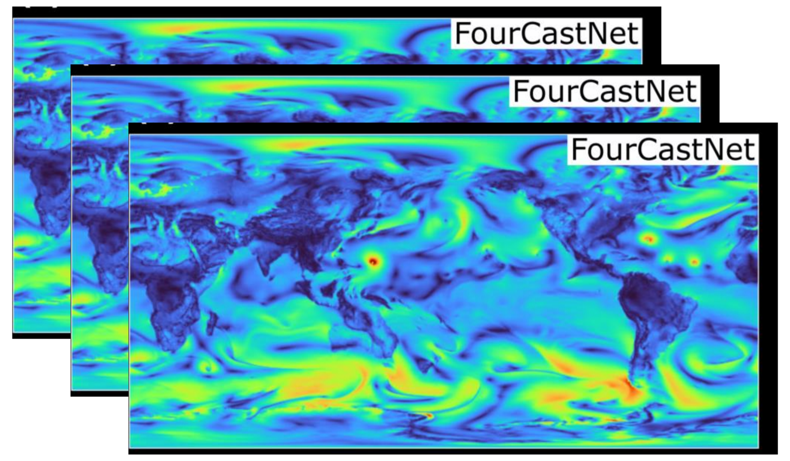
5TB
ERA5

loss

GT

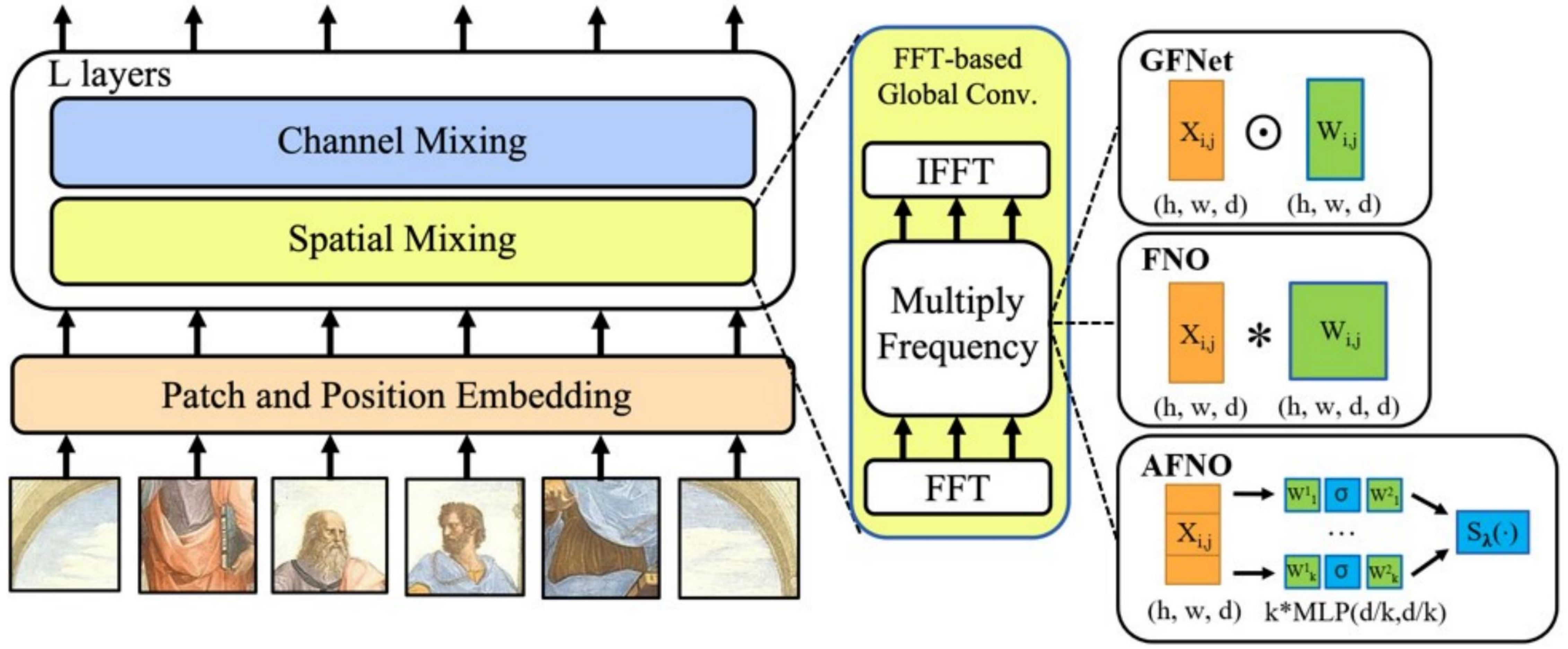
input → AFNO backbone → predict

predict
20 variables



20 variables

20ch x 720H x 1440W
20ch x (8x8)p x 90H x180W



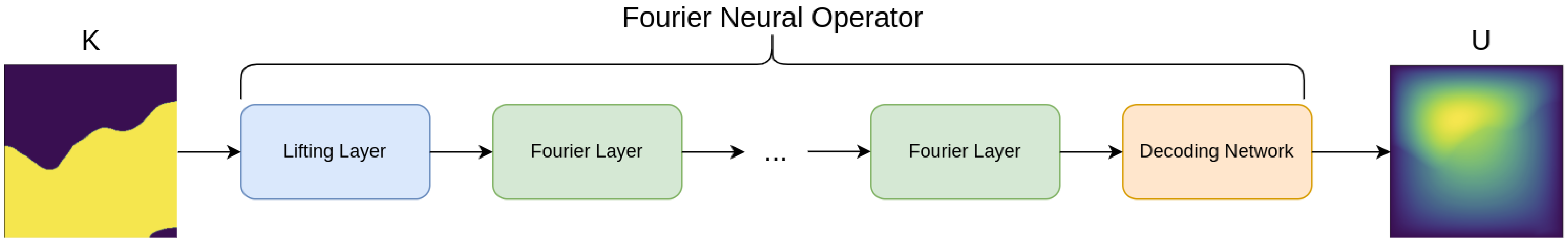
FOURIER NEURAL OPERATOR

This tutorial sets up a data-driven model for a 2D Darcy flow using the Fourier Neural Operator (FNO) architecture inside of Modulus. It covers these details:

1. Loading grid data and setting up data-driven constraints
2. How to create a grid validator node
3. How to use Fourier Neural Operator architecture in Modulus

This problem develops a surrogate model that learns the mapping between a permeability field and the pressure field of a Darcy system governed by the elliptic PDE:

$$-\nabla \cdot (k(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in D,$$



FOURIER NEURAL OPERATOR

Results

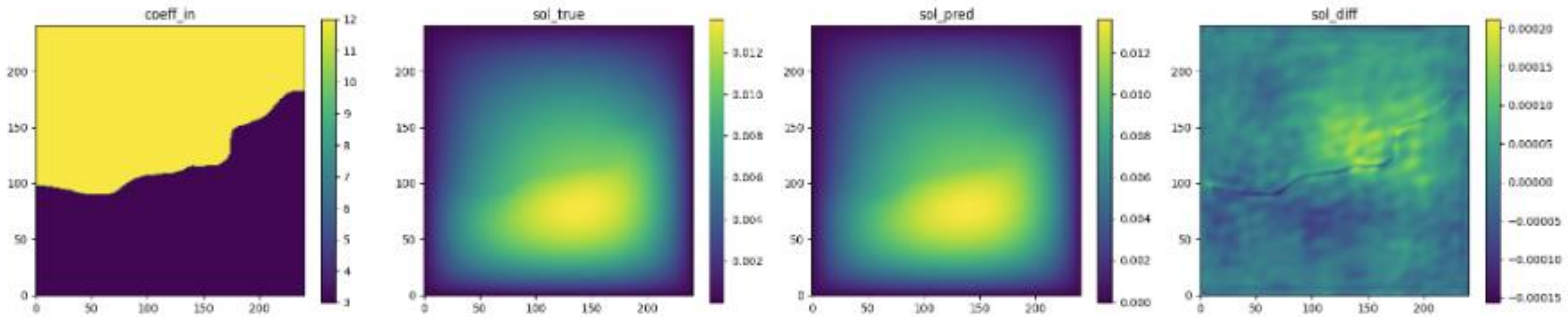


Fig. 61 FNO validation prediction 1. (Left to right) Input permeability, true pressure, predicted pressure, error.

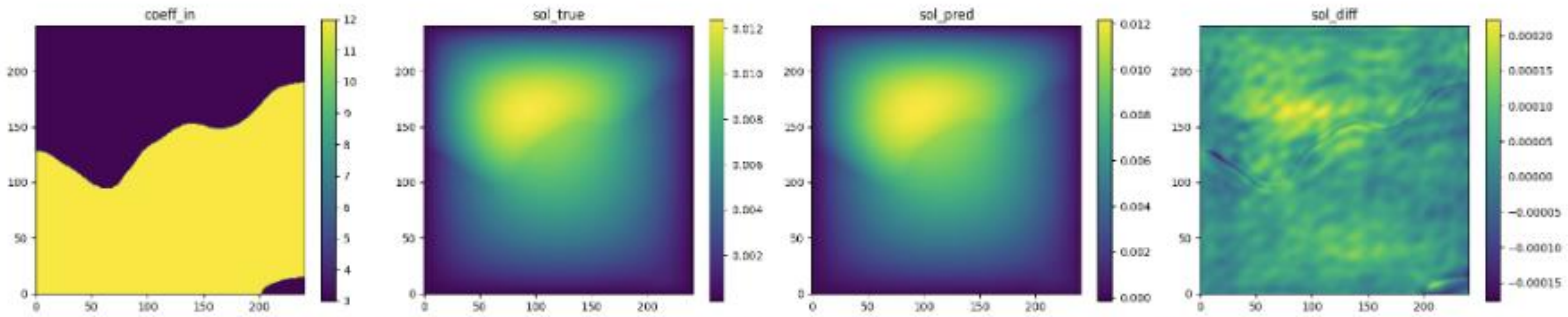


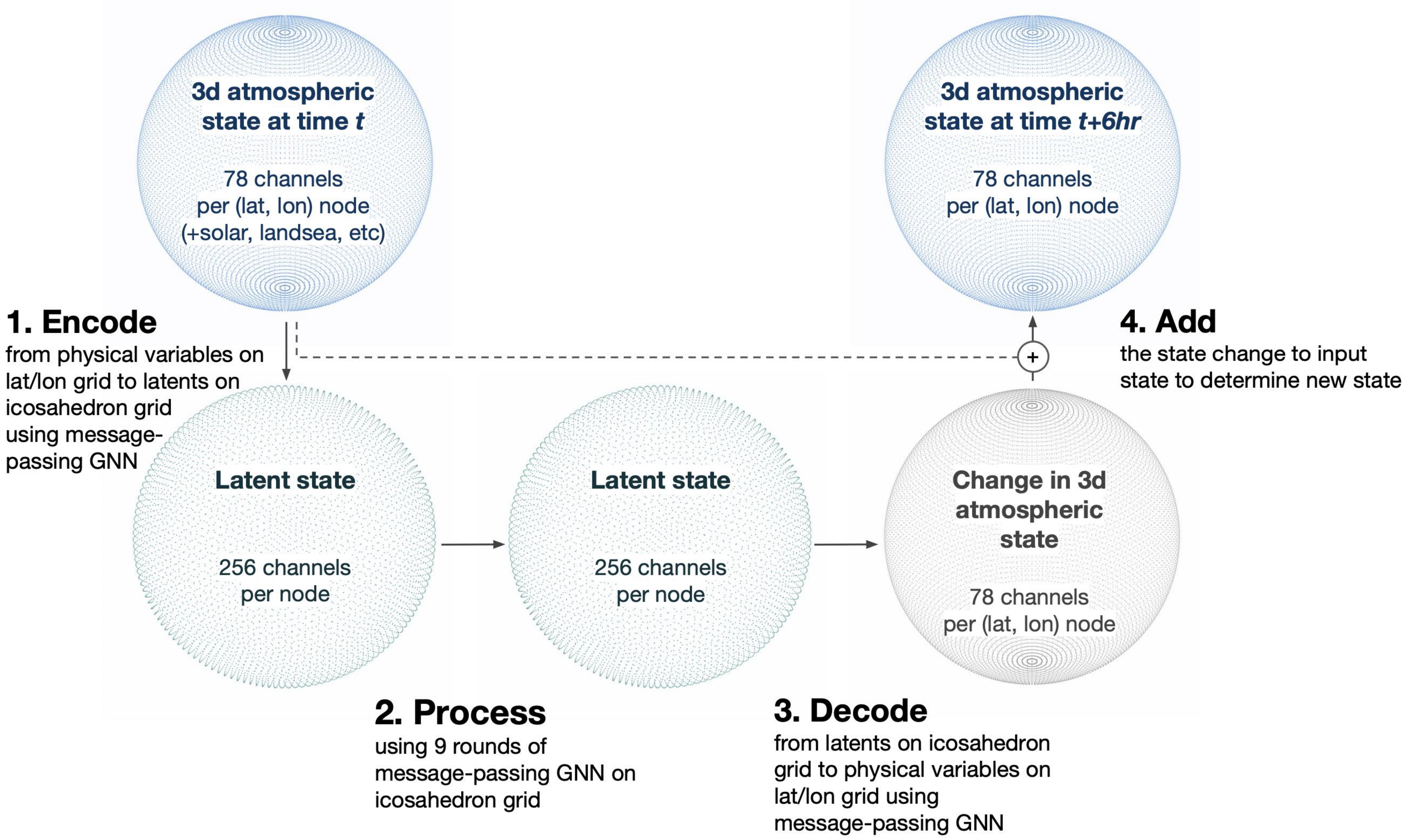
Fig. 62 FNO validation prediction 2. (Left to right) Input permeability, true pressure, predicted pressure, error.

FNO accurately learns the solution of this system.

Modulus supports the visualization of results through images (matplotlib), Tensorboard, VTK files and Omniverse for select problems.

For more information, please refer to the official Modulus user guide example.

Ryan Keisler's GNN model



GNN

enc-dec arch.
2 enc + 6 dec layer

2d rec → graph

Multilevel(1d~3d)

ERA5 dataset
1979~2020(6yr test), 3hr interval
1d(360x180)
full variables
- 6 var 13 pres, [TZQUVW]
- 4 surf variable

6.7M params
5.5 day 1ea GPU

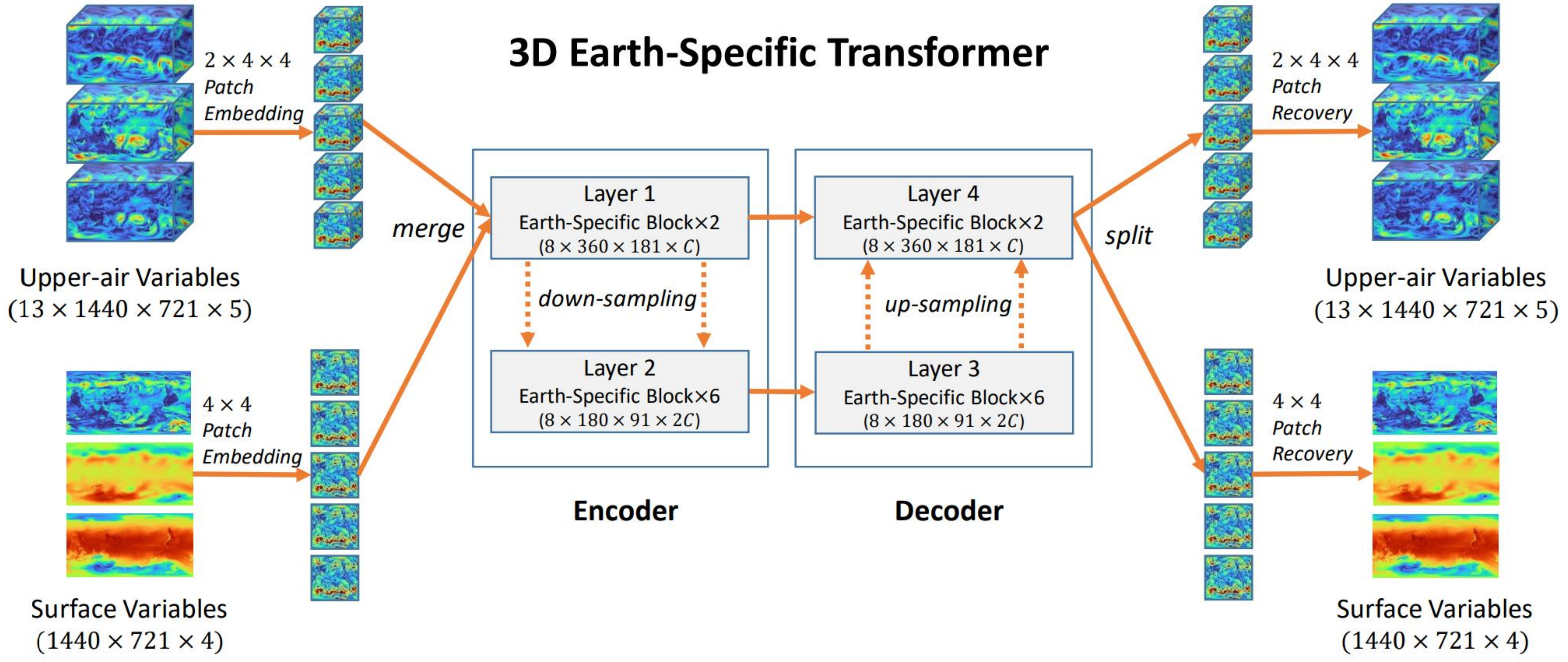
Huawei Pangu-Weather

https://arxiv.org/pdf/2211.02556.pdf

TF

2d rec

Swin transformer
enc-dec arch.
2 enc + 6 dec layer



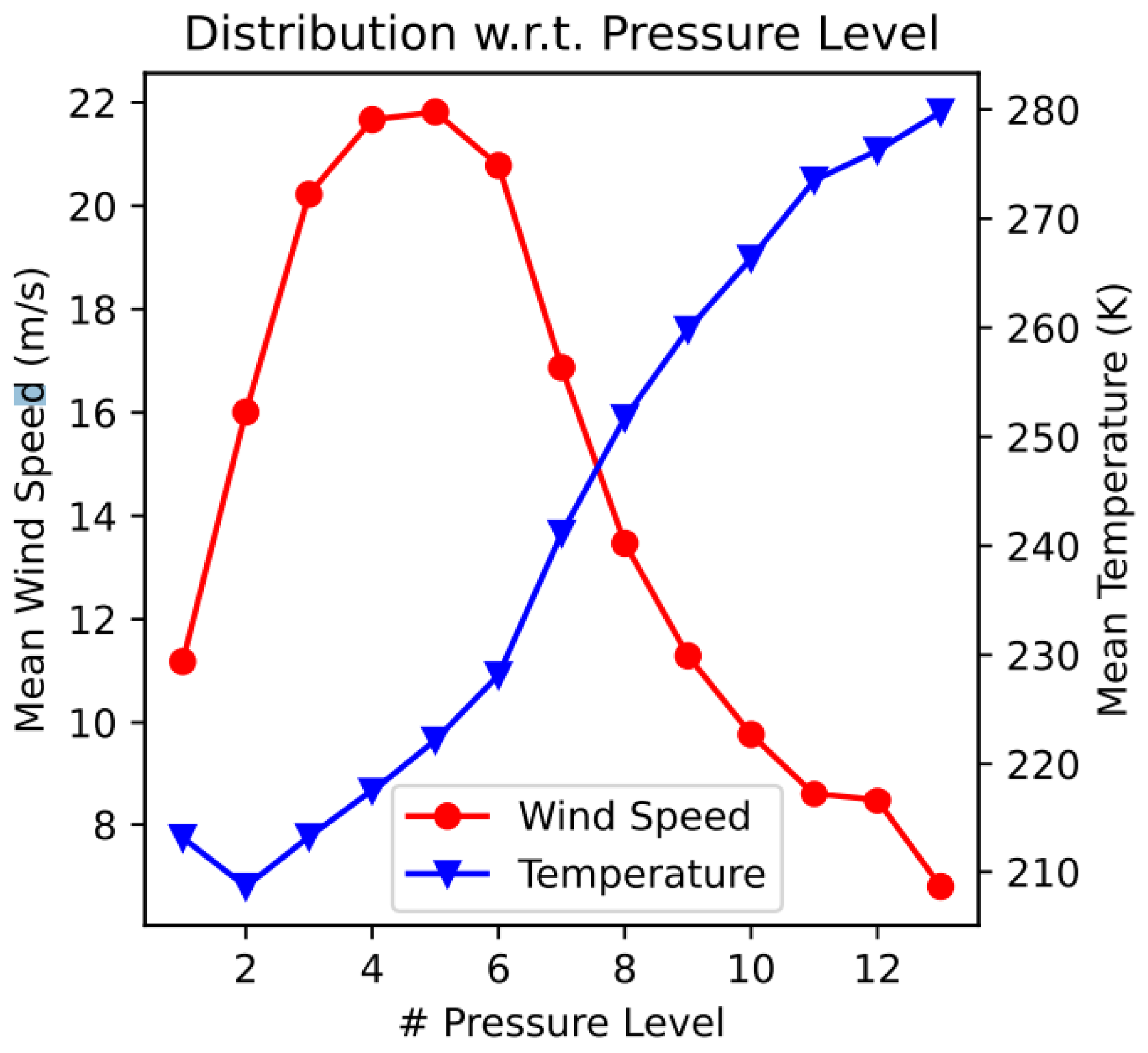
ERA5 dataset
1979~2017(39yr), 6hr interval(leadtime)
0.25d(1440x721)
full variables

- 5 var 13 pres,
- 4 surf variable

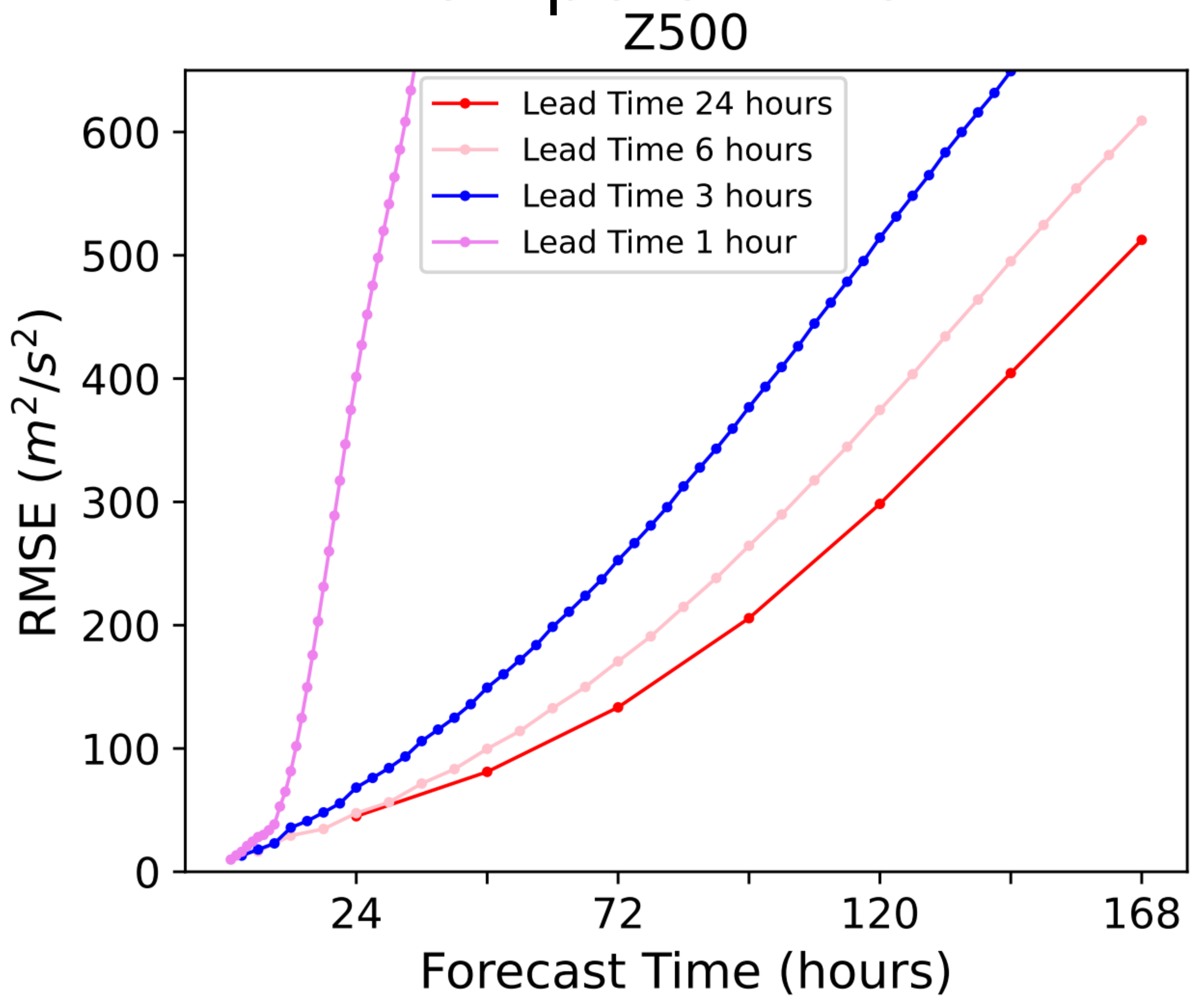
Train : 15 day, 192EA V100

Huawei Pangu-Weather Result and insight

Pressure level



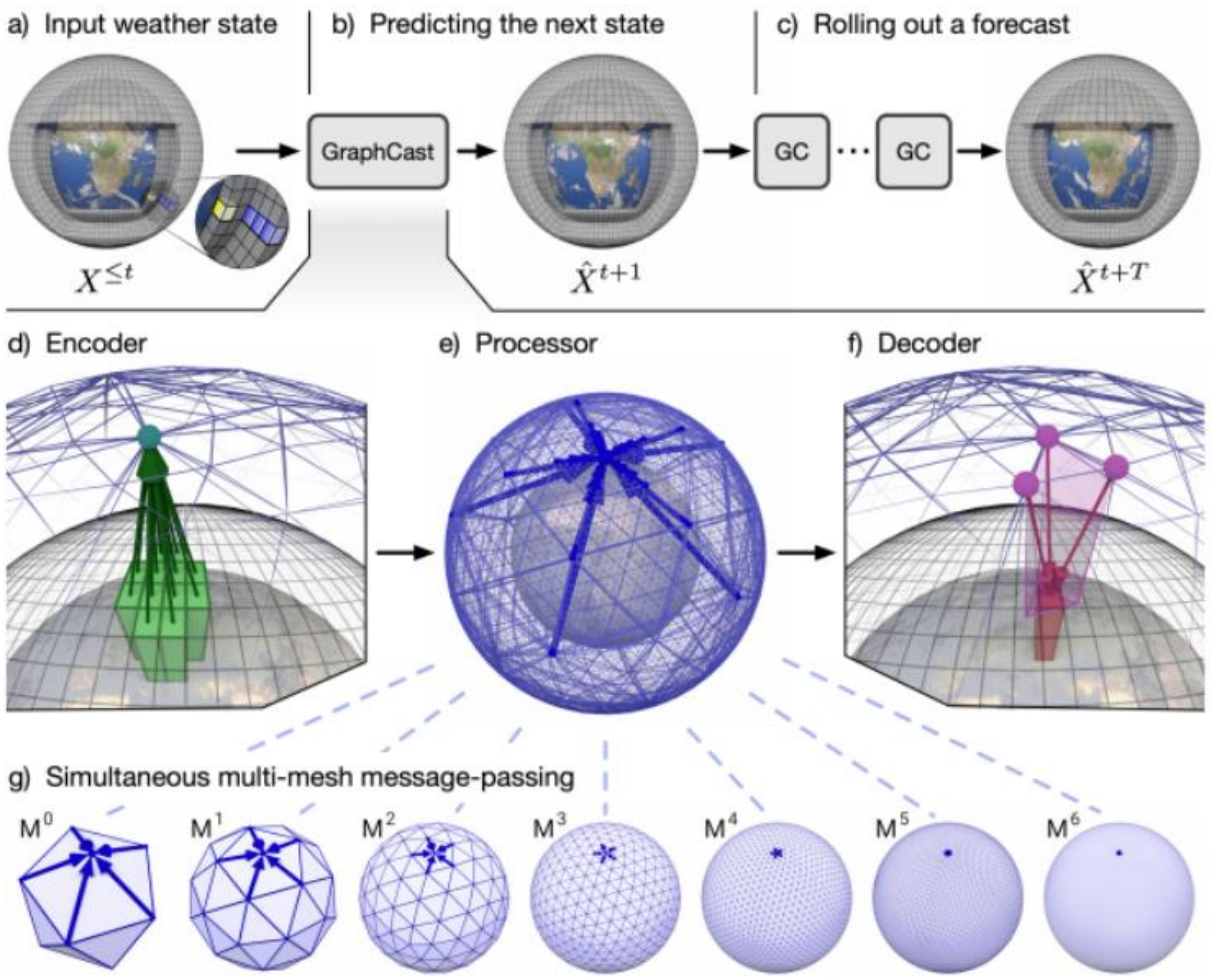
Temporal Error



2018 Kong-rey

Google DeepMind GraphCast

https://arxiv.org/abs/2212.12794



GNN

Enc-dec arch

2d rec → graph

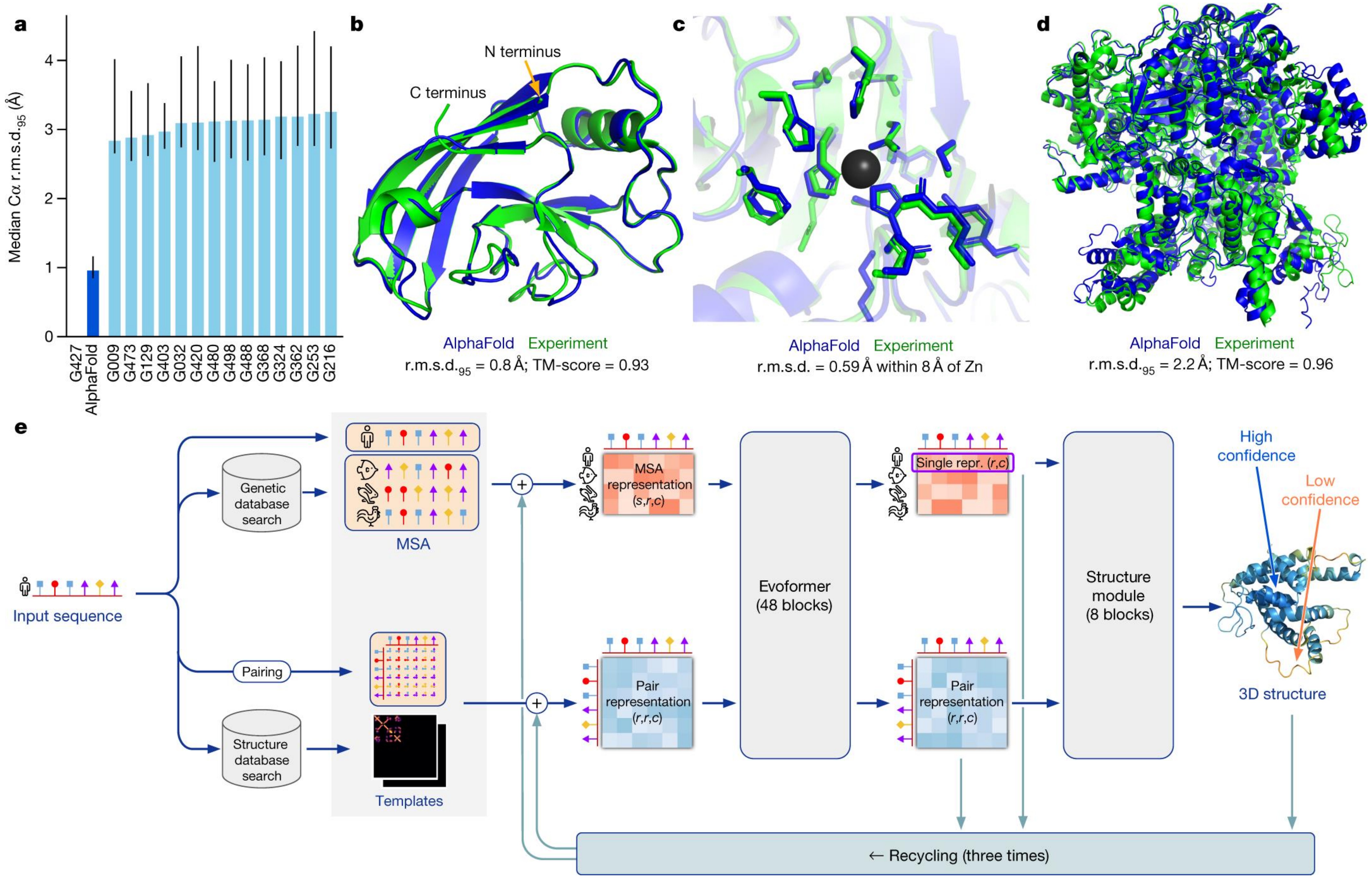
multigrid

1979~2018, 6hr interval
0.25d(1440x721)
full variables
- 6 var 37 pres,
- 5 surf variable

10-day forecast (at 6-hour steps) in under 60 seconds
TPU based, no opensource

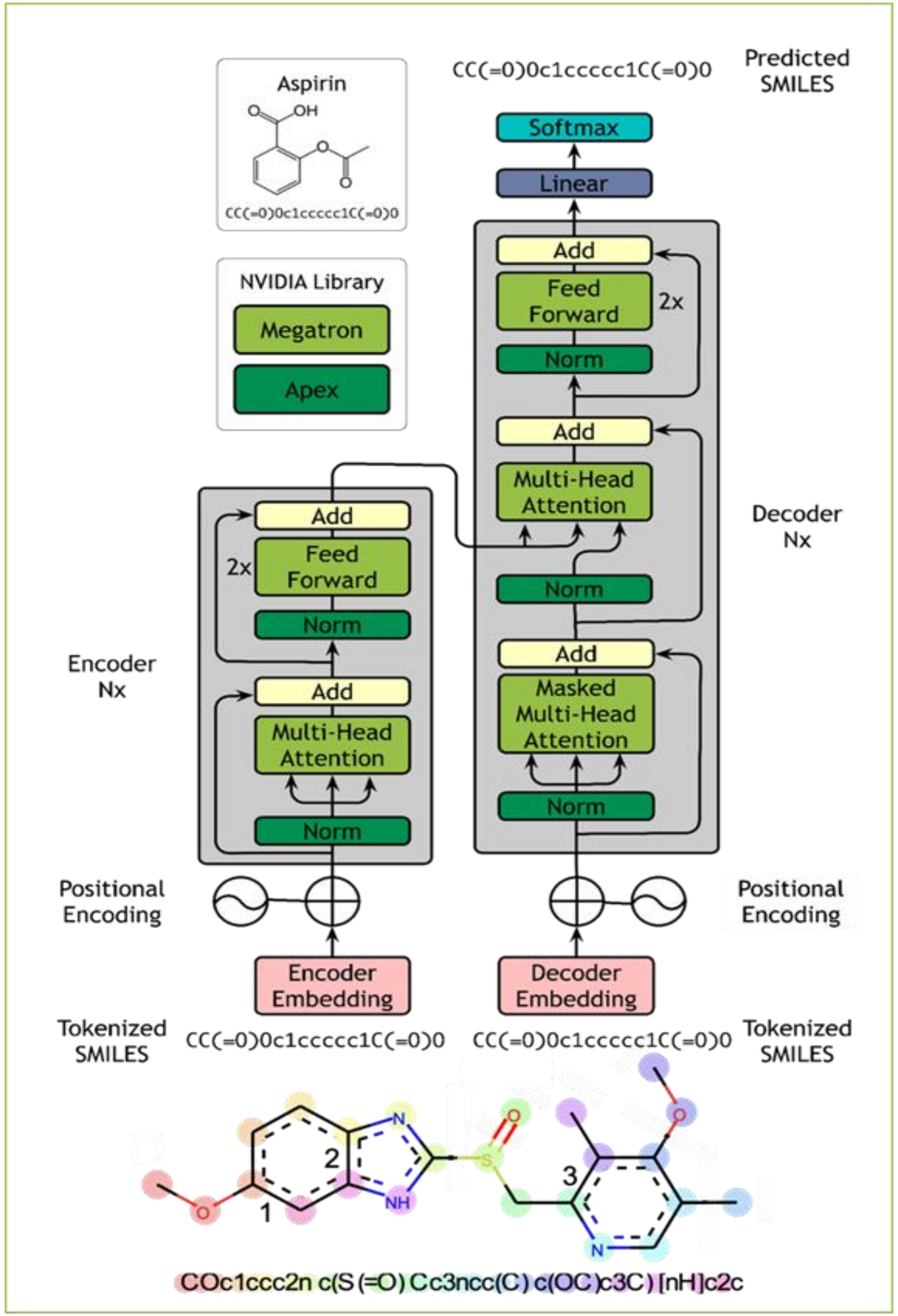
ALPHAFOLD2

Predict 3D Structure of Protein



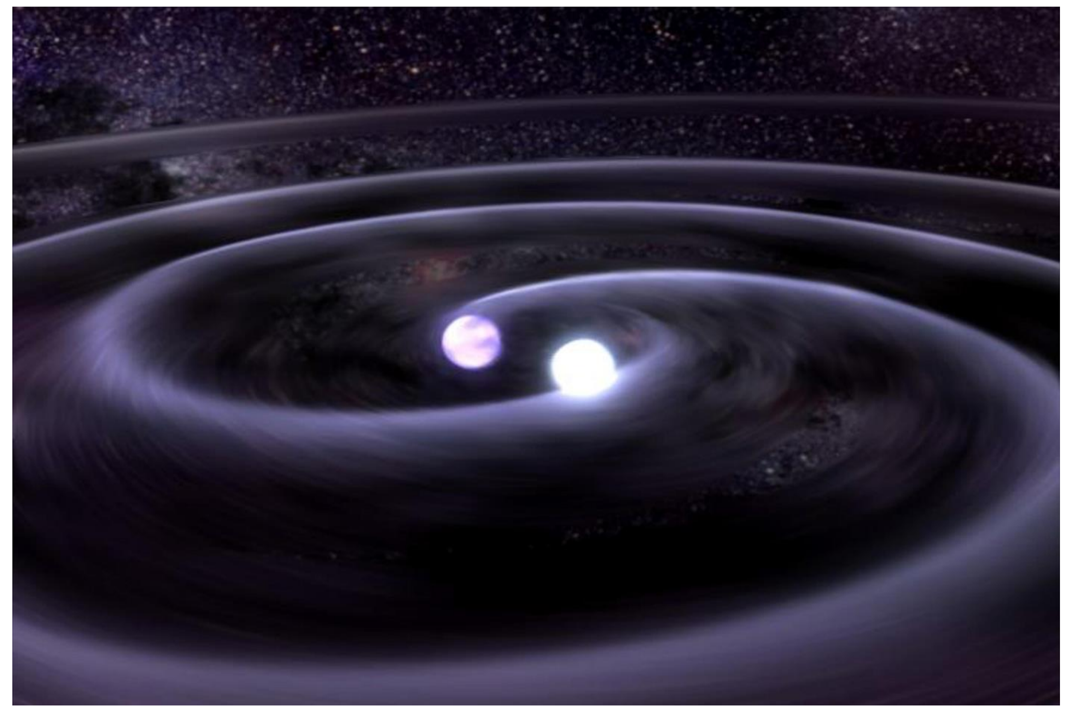
MEGAMOLBART

- MegaMolBART is a deep learning model for small molecule drug discovery and cheminformatics based on SMILES. MegaMolBART uses NVIDIA's Megatron framework, designed to develop large transformer models.
- The ZINC-15 database is used for pre-training. Approximately 1.45 Billion molecules (SMILES strings) were selected from tranches meeting the following constraints: molecular weight <= 500 Daltons, LogP <= 5, reactivity level was "reactive", and purchasability was "annotated". SMILES formats, including chirality notations, are used as-is from ZINC.

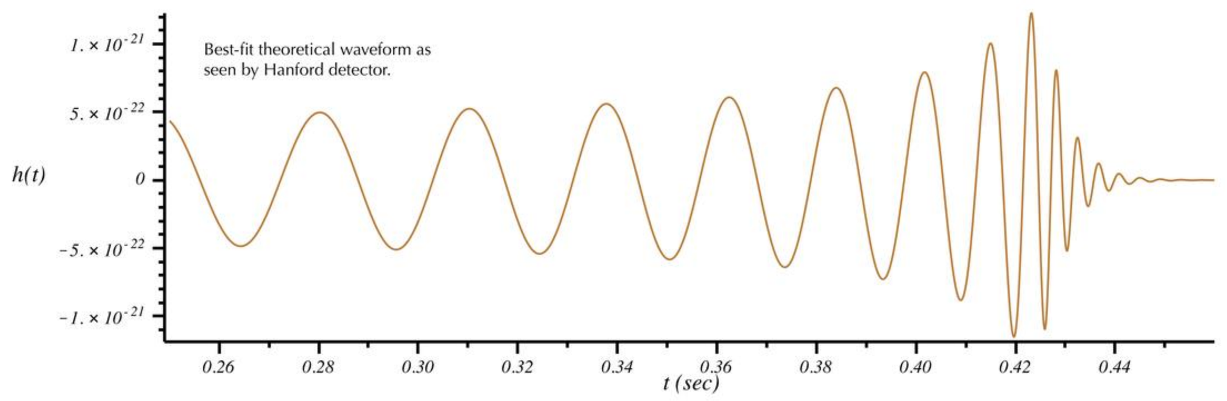


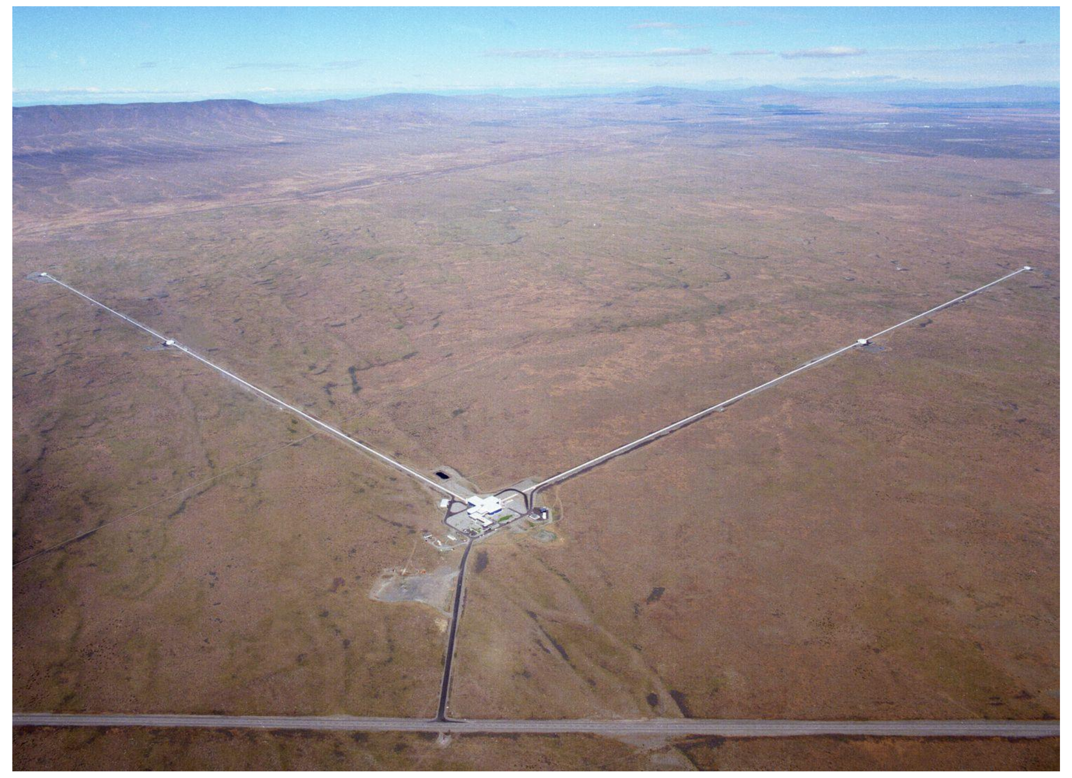
G2NET GRAVITATIONAL WAVE DETECTION

https://www.kaggle.com/c/g2net-gravitational-wave-detection



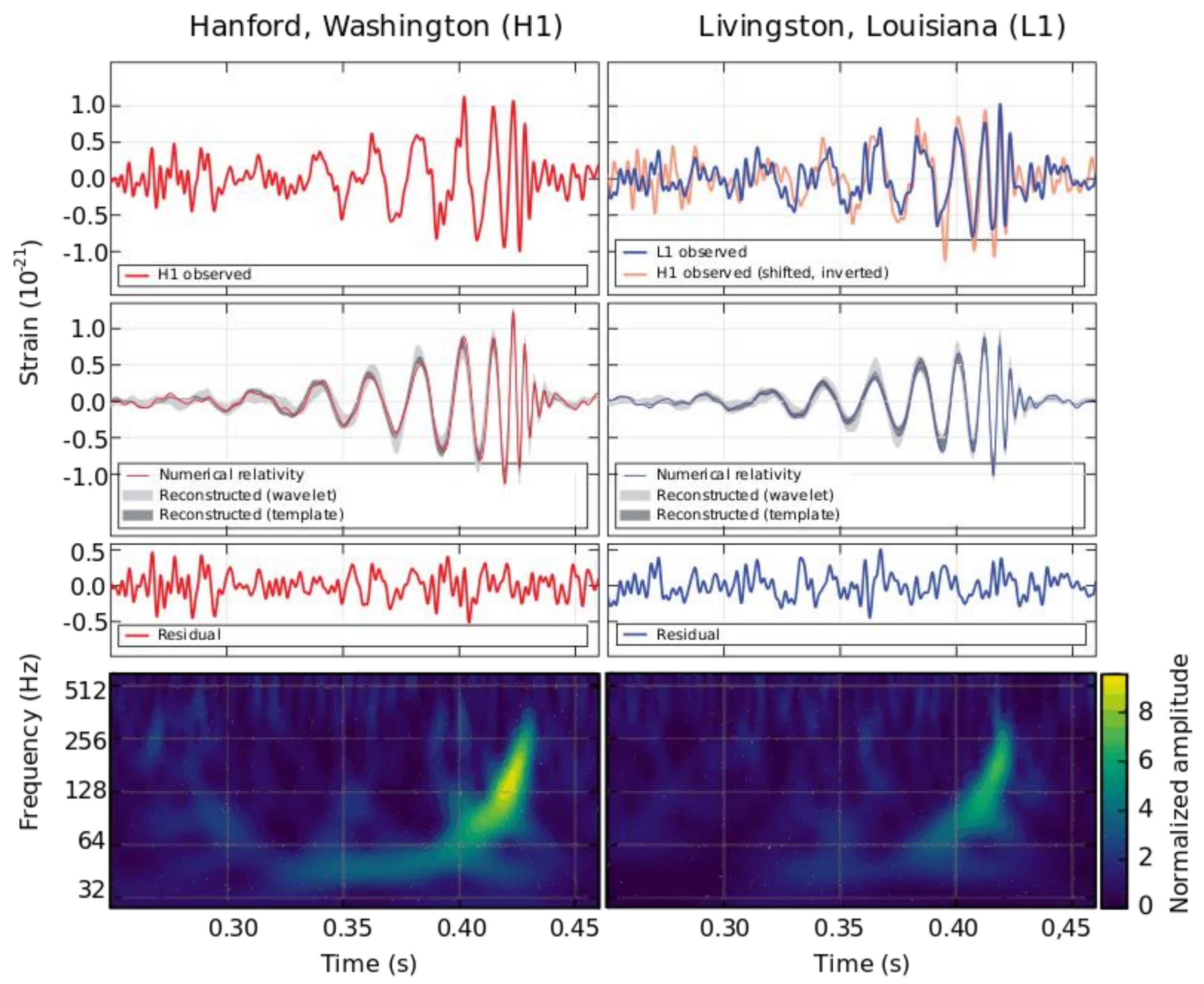
Simulated chirp signal





GW Detector

Multi variate obversation



Noisy signal

Feature engineering

Correlation(DTW)
Denoising
MFCC/MEL Spectrogram
Augmentation

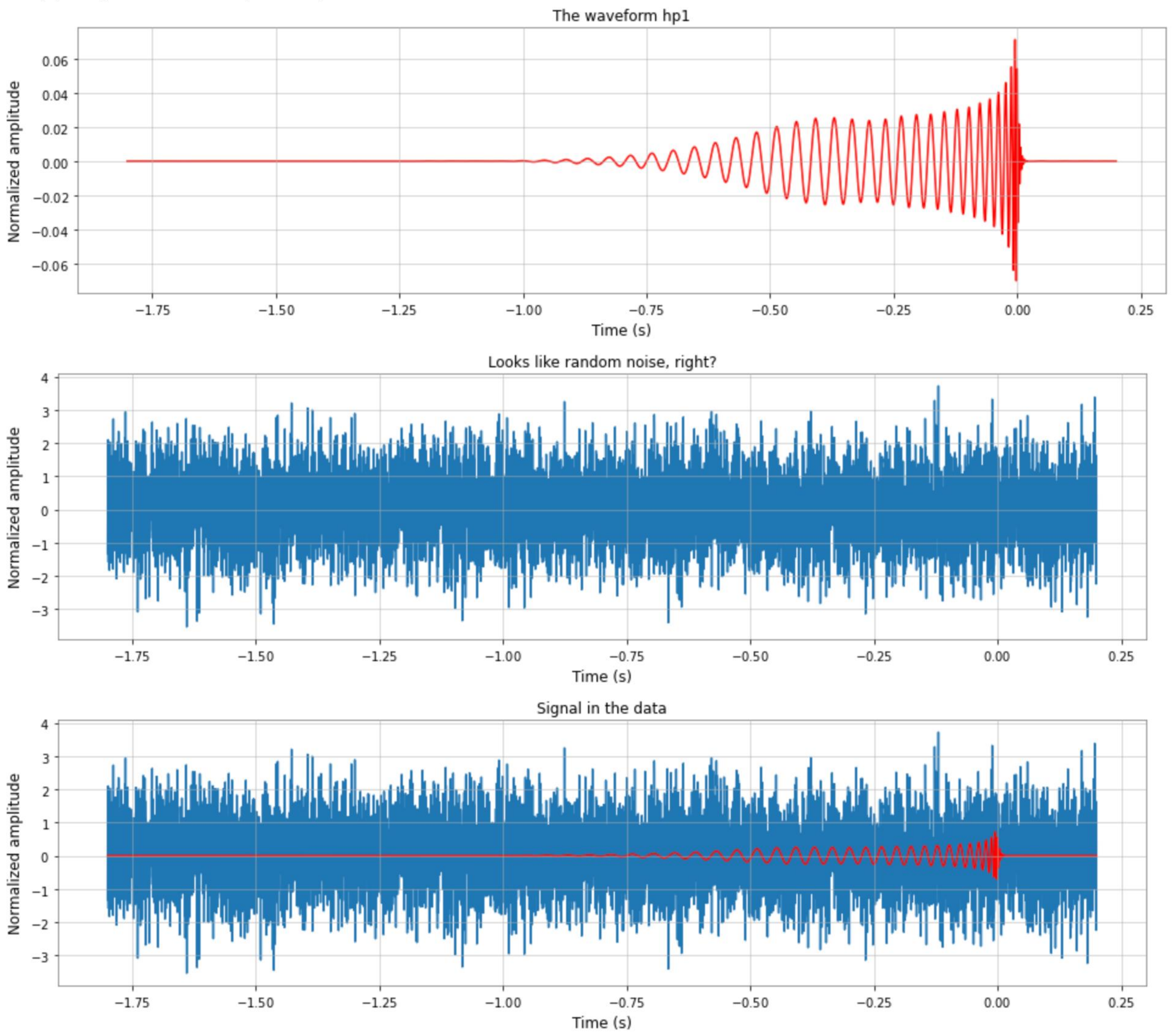
DNN

GW or not

GW_ODW_2019 EXAMPLE

Synthetic GW

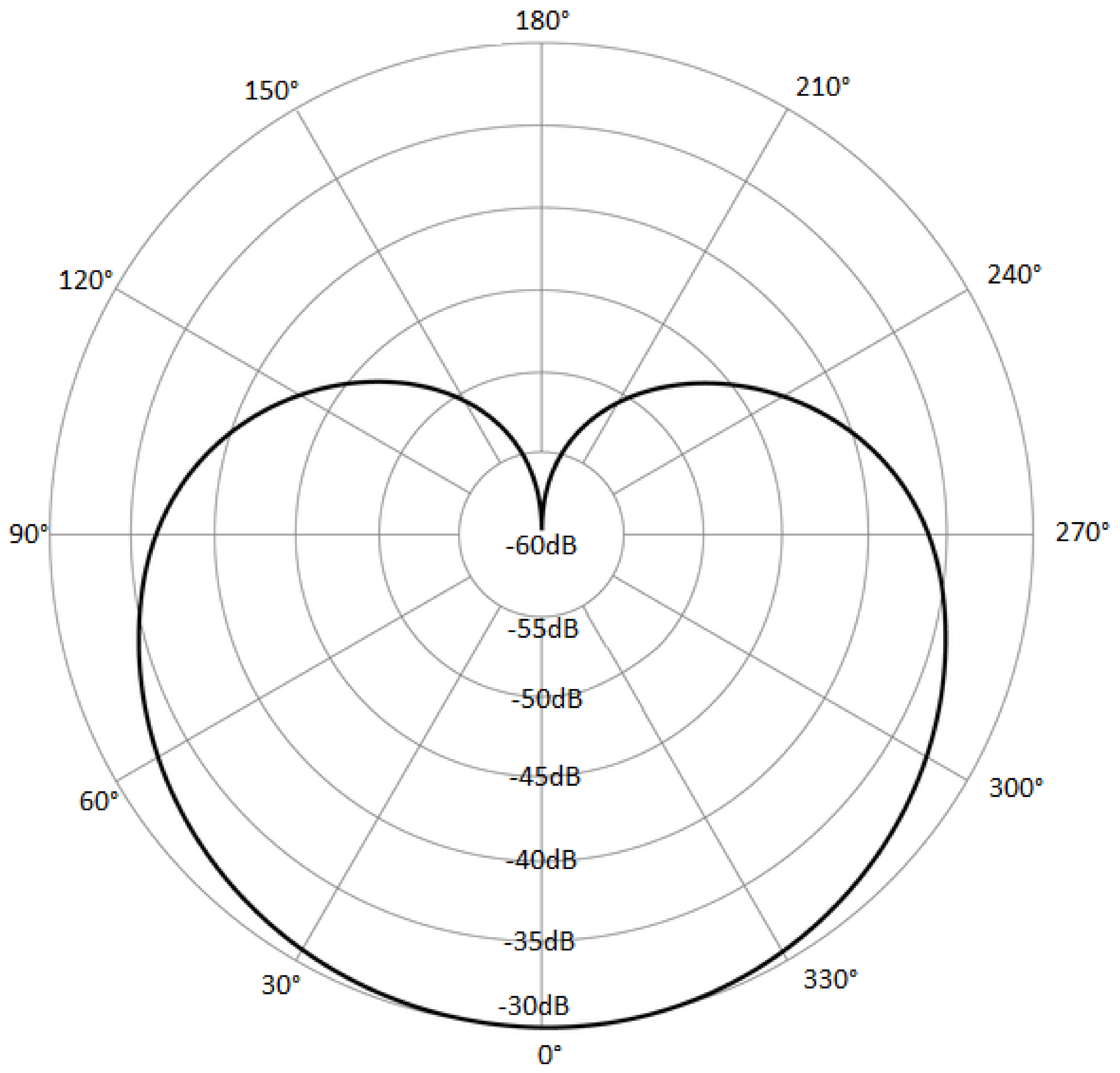
```
from pycbc.waveform import get_td_waveform


sample_rate = 4*1024 # samples per second
data_length = 32 # seconds

apx = 'IMRPhenomD'

# GW170809
hp1, _ = get_td_waveform(approximant=apx,
                         mass1=35.0,
                         mass2=23.8,
                         delta_t=1.0/sample_rate,
                         f_lower=25)


hp1 = hp1 / max(numpy.correlate(hp1,hp1, mode='full'))**0.5


pylab.figure( figsize=(16,4) )
pylab.title("The waveform hp1")
pylab.plot(hp1.sample_times, hp1, color='red')
pylab.xlabel('Time (s)')
pylab.ylabel('Normalized amplitude')

waveform_start = numpy.random.randint(0, len(data) - len(hp1))
data[waveform_start:waveform_start+len(hp1)] += 10 * hp1.numpy()
```



COMPARE TO AUDIO PROCESSING

RIR(simulator)

Cardioid Microphone Polar Plot

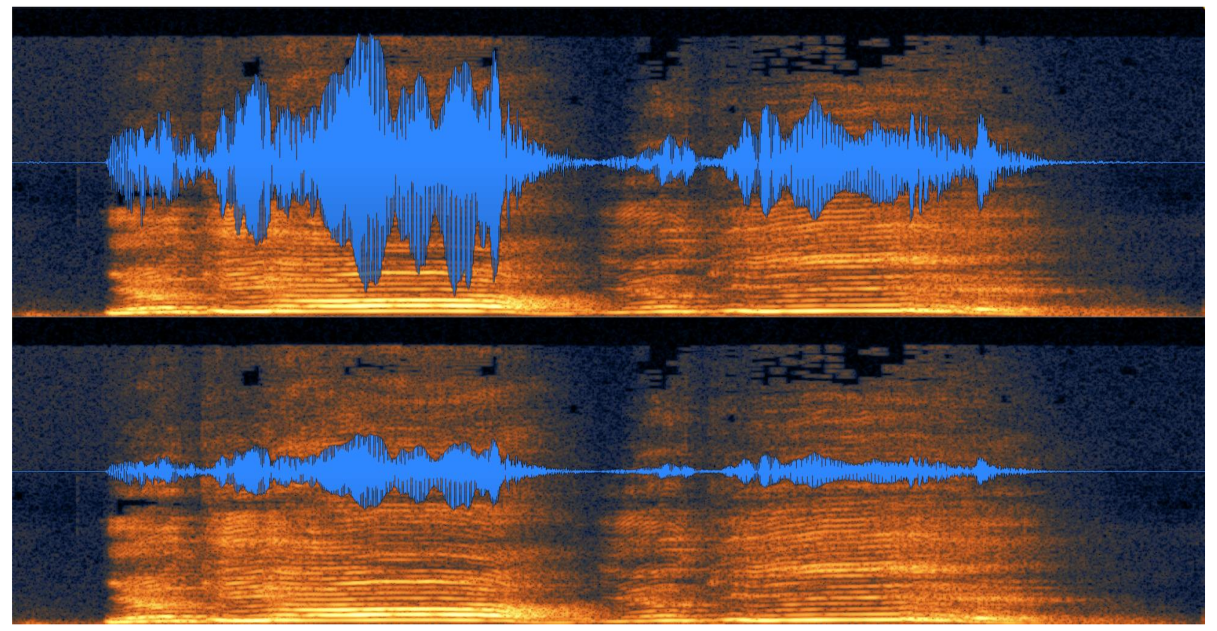



```python
def volume_slider(signal, dB):
    signal = signal*gain_scaler(dB)
    return signal

def mic_angle(theta=0, m1=-45, m2=45, dis_mic=0.039):
    recv_angle_m1 = theta+m1
    recv_angle_m2 = theta-(180-m2)
    return recv_angle_m1, recv_angle_m2

def cardiod_2d(alpha=0.5, angle=5):
    radians = np.deg2rad(angle)
    alpha=0.5
    result=1
    result = alpha * (1. + np.cos(radians))
    return round(result,4)
```

mono recording

Cardiac Simulator

L



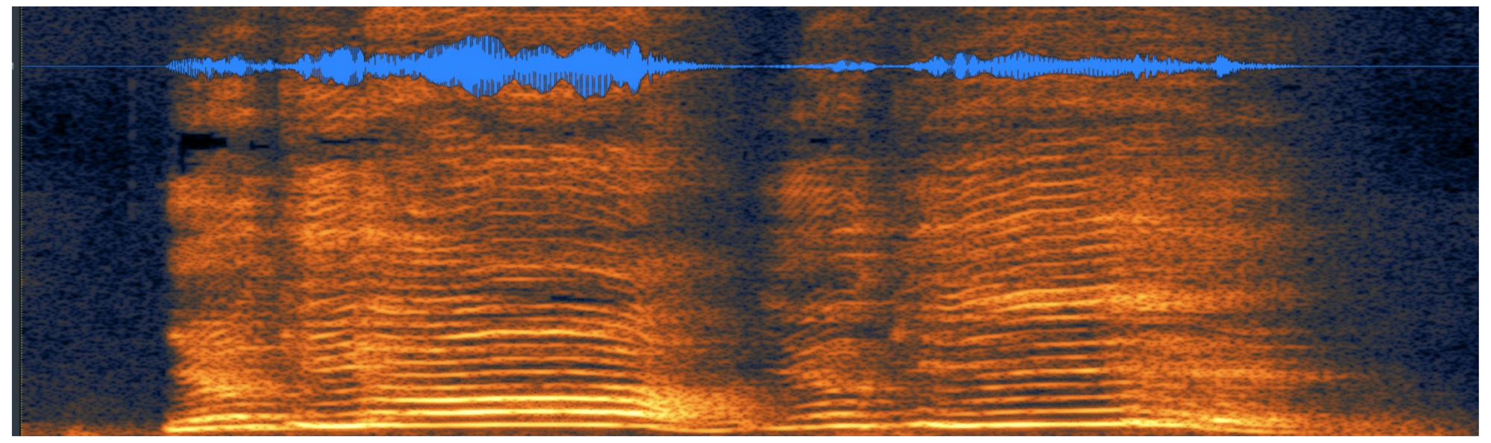
R

```python
def do_rir_generator(file_name, target_path, save_filename, srt, distance, theta, jitters=0):
    from librosa.core import load as wfload
    data, sr = wfload(file_name, sr = srt, mono=True)

    #adjust distance
    distance_adj = dB_distance_diff(60,4.99,distance)
    data = volume_slider(data,distance_adj)

    #adjust angle
    m1_angle, m2_angle=mic_angle(theta=theta)
    left_adj  =  cardiod_2d(alpha=0.5, angle=m1_angle+jitters )
    right_adj =  cardiod_2d(alpha=0.5, angle=m2_angle+jitters )
    data_left =data *  left_adj
    data_right=data * right_adj

    data_left=float_to_pcm16(data_left)
    data_right=float_to_pcm16(data_right)

    data_stereo=np.vstack((data_left, data_right))
    save_wave_file_rir(data_stereo, srt,  target_path , save_filename, distance, theta, jitters )

    return data_stereo
```

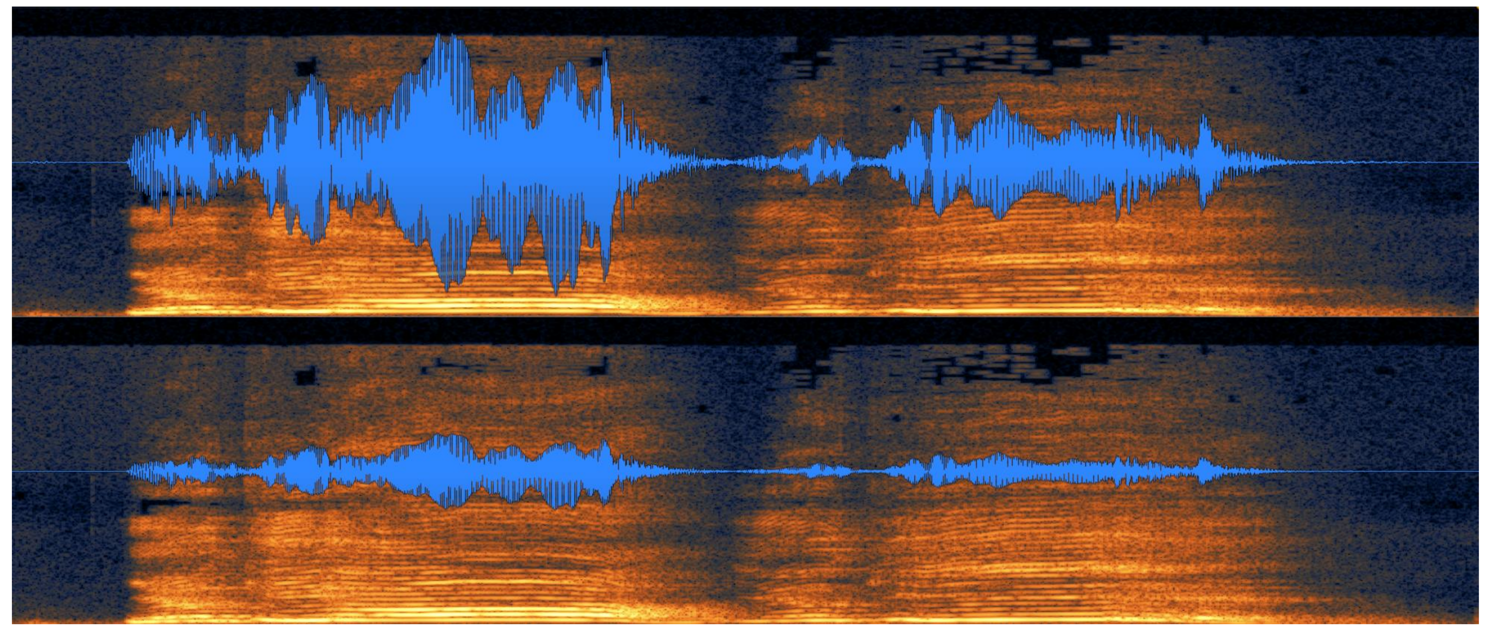
COMPARE TO AUDIO PROCESSING

add noise

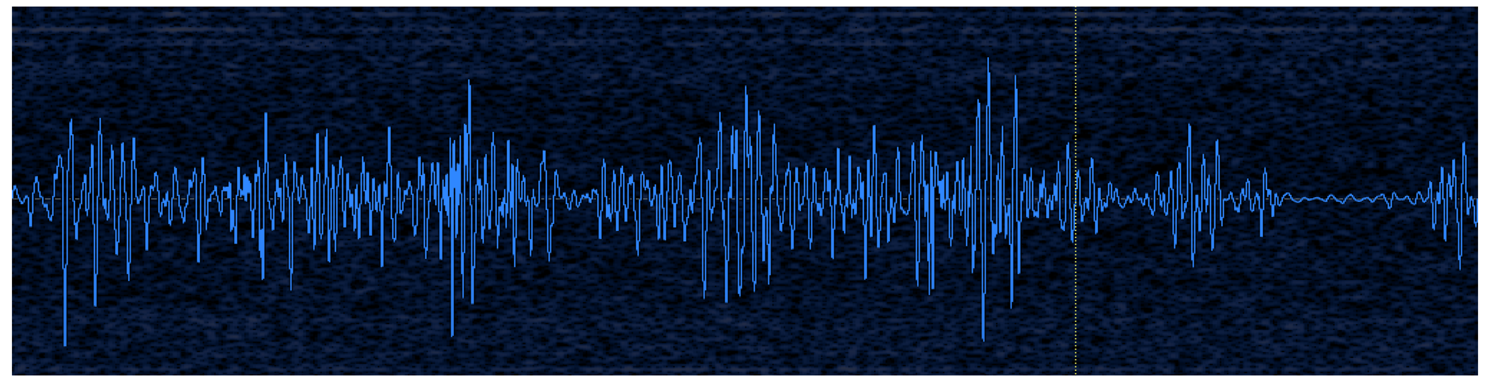
Voice



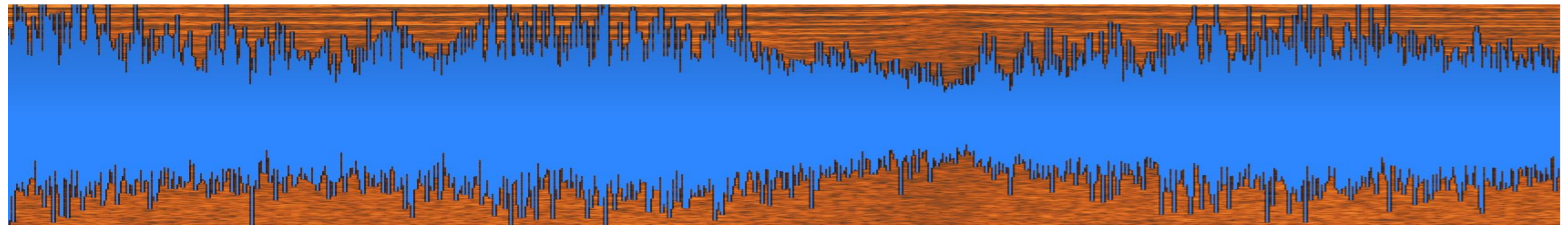
cardioid



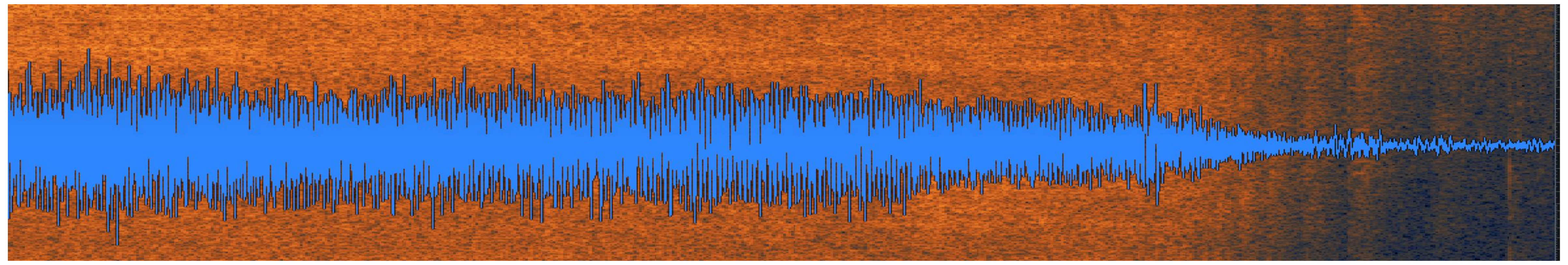
Windy
effect



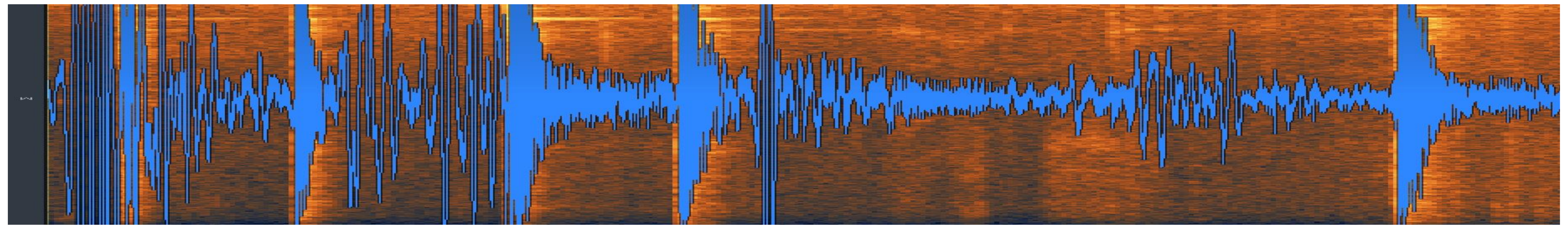
Drone noise



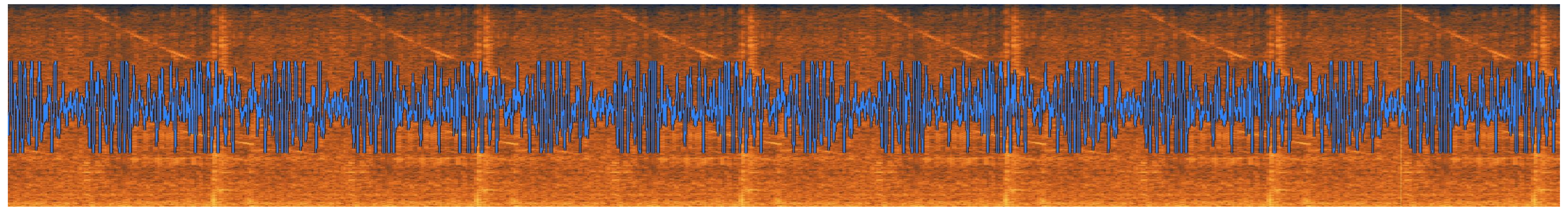
drill noise



Hammer noise



engine noise



COMPARE TO AUDIO PROCESSING

ASR Pipeline

MODULES FOR DL

DL Model
- Demo only
- Paper only
  - With sample
  - With code
  - With dataset
  - With Checkpoint

application
Paperwithcode, github
NEMO, RIVA, MONAI, Hugginface
timm, einops,

Pair of (Input,Output)
(Image, Optical Flow)
(text, image), (image, cls)
(audio, text)

Data Loader
Dali, stream
Augment, patch

preprocessing
Tokenizer, normalizer

Dataset
Image, WSI, X-ray/MRI,
Lanauge(audio,text), video, 3D, stereo,
Chemical, Protein, CFD

Technique
AMP, Data Parallel, Model Parallel, Quantization, hash, parameter sharing, checkpointing, ZeRO,

Train recipe
Learning rate schedule(Cosine, warm up), early stopping
Optimizer(Adam), accumulation

Task
Multistage, multi modal, end2end, Pretrain/finetune, distill, quantization
Regression, CLS, AE, GAN, Prompt, LM, AR, MLM, denoising, jigsaw, SuperRes

Objective
MSE, Cross Entropy, Dice, triplet, contrative

Model
Model : ResNet, EfficientNet, Unet, Hifi-GAN, transformer, BERT, BART, GPT-2, GPT-3 , NERF
Module : Pool, Conv, LSTM, GRU, FCN, MLA, GNN, softmax, GeLU, ReLU, Residual, Skip
Variation : Prenorm, postnorm,

DLFW
Pytorch, TF, Keras, DGL, PyG, JAX, pennylane, TorchANI
WanDB, ignite, torchlightening,

DevOps
OS(Ubuntu,WSL2), PIP, Conda, Singularity, Docker,
slurm/PBS/LSF, jupyter, NFS, Baremetal/Virtual, Ansible

Resource
GPU, TensorCore, multiGPU, MultiNode, IB,

EXAMPLES

Healthcare

Task : lung CT segmentation
Data pair : In:CT raw, Out : Segmentation
Dataset : COVID19-CT-Dataset
Augmentation : none
DataLoader : nefti reader(MONAI)

Task : 3D segmentation
Model : Unet(MONAI)
Optimizer : Adam
Recipe : train with warm up

System : 1 node (2EA RTX8000 40GB)
OS : Ubuntu
DLFW : pytorch on NGC docker

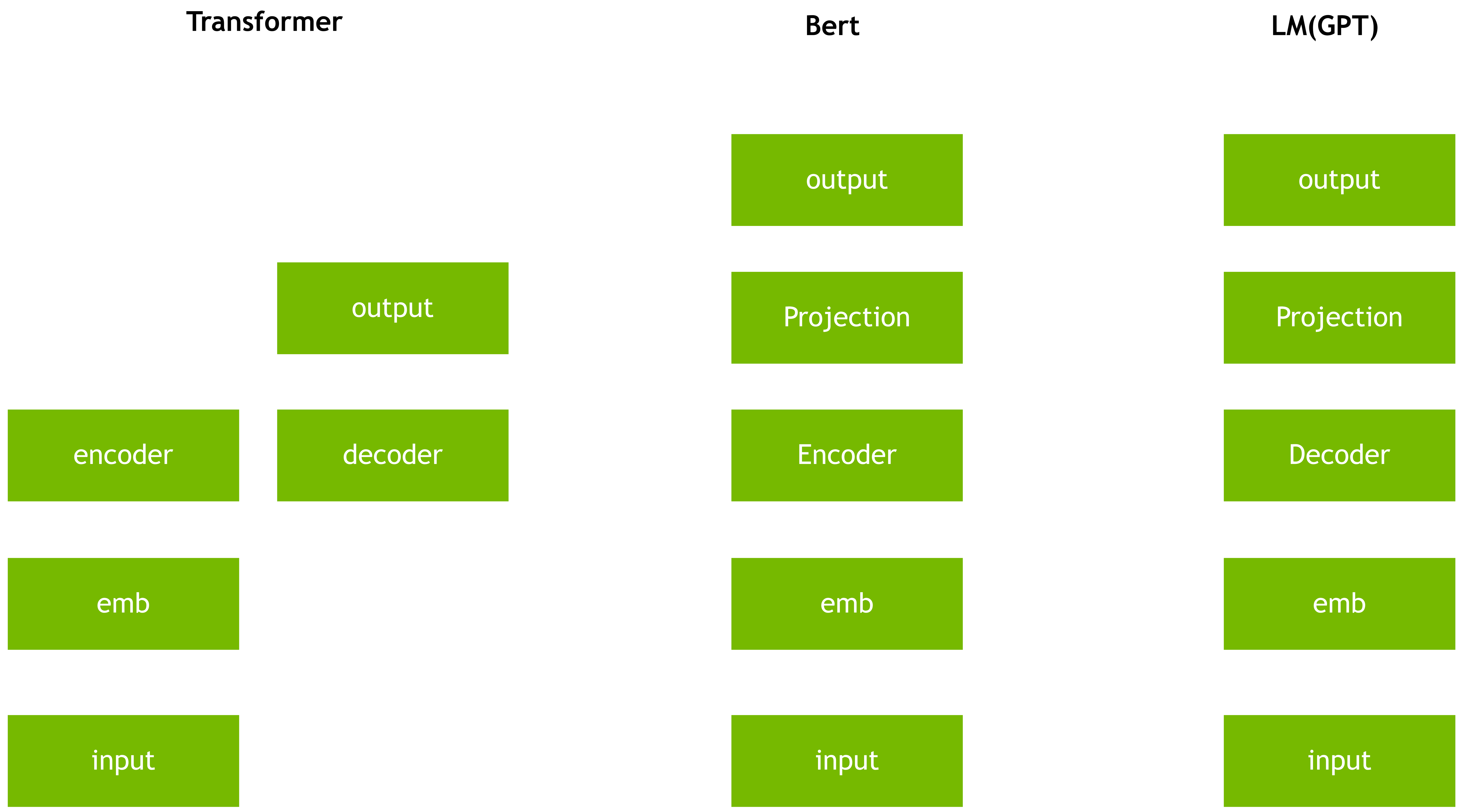
Audio

Task : ASR
Data pair : In:audio, Out : text
Dataset : LibriLight
Augmentation : SpecAug
DataLoader : Nemo

Task : ASR
Model : ContextNet(Conv, SELayer)(NEMO)
Recipe : train with warm up

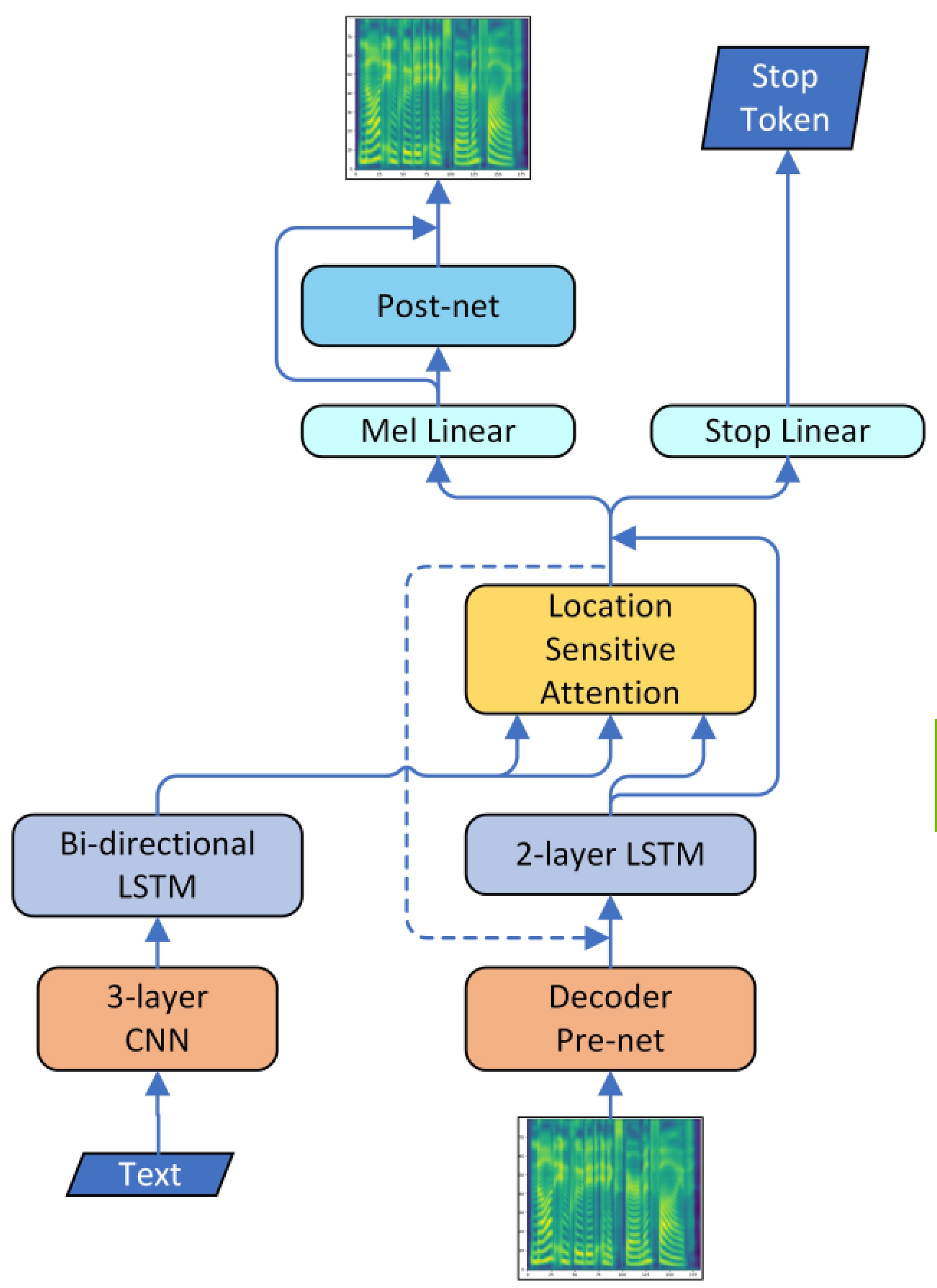
System : 2 node DGX-1 (8EA A100 80GB)
OS : Ubuntu
DLFW : pytorch on singularity, slurm

TRANSFORMERS

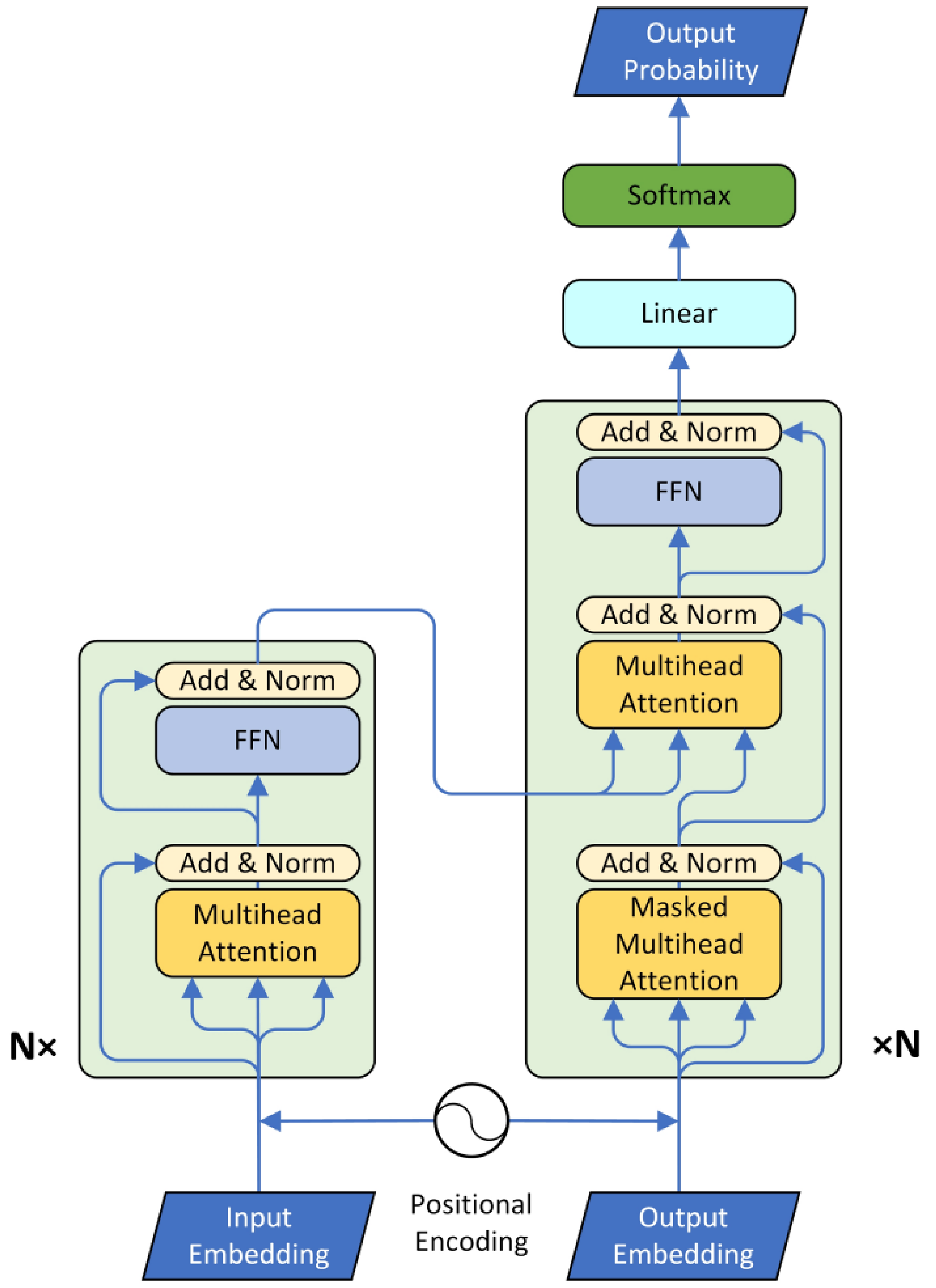


Transformer IN Various Domain

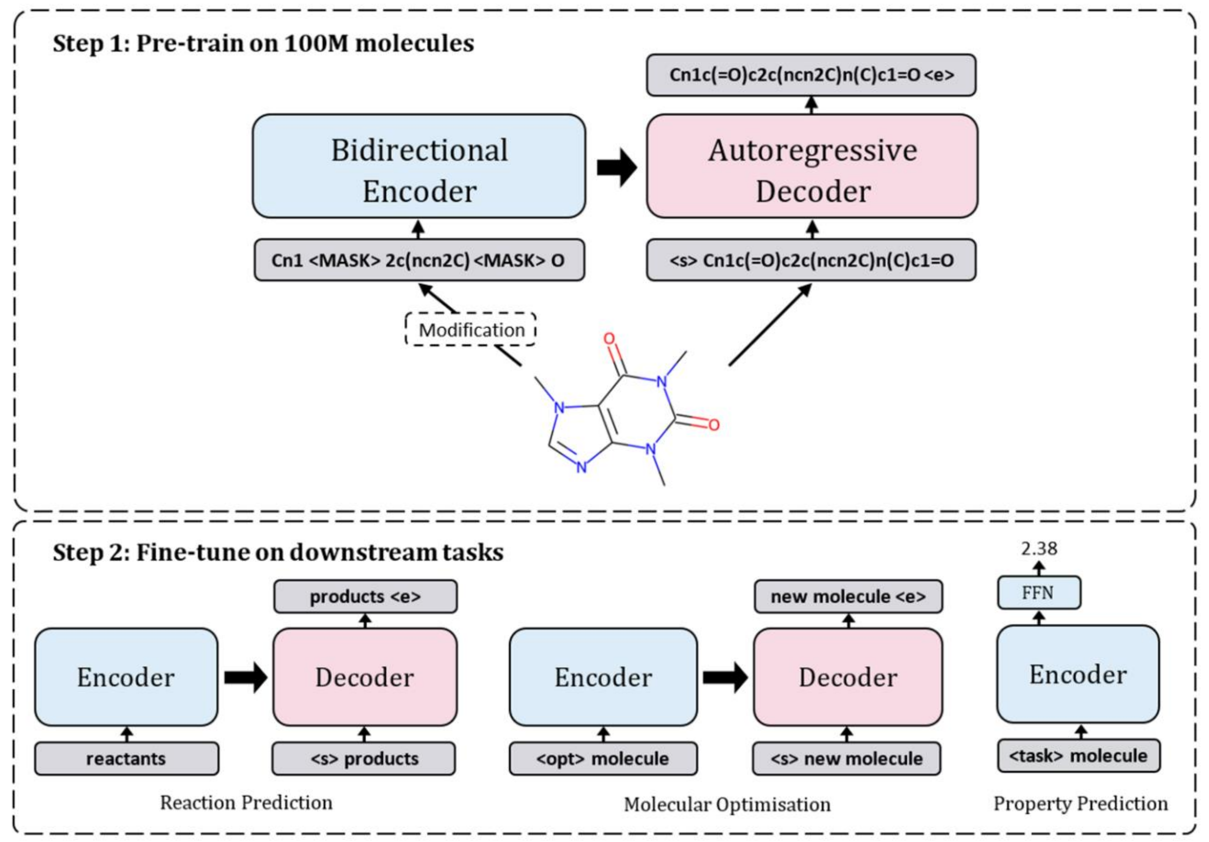
Neural Speech Synthesis with Transformer Network(2019)
https://arxiv.org/pdf/1809.08895.pdf



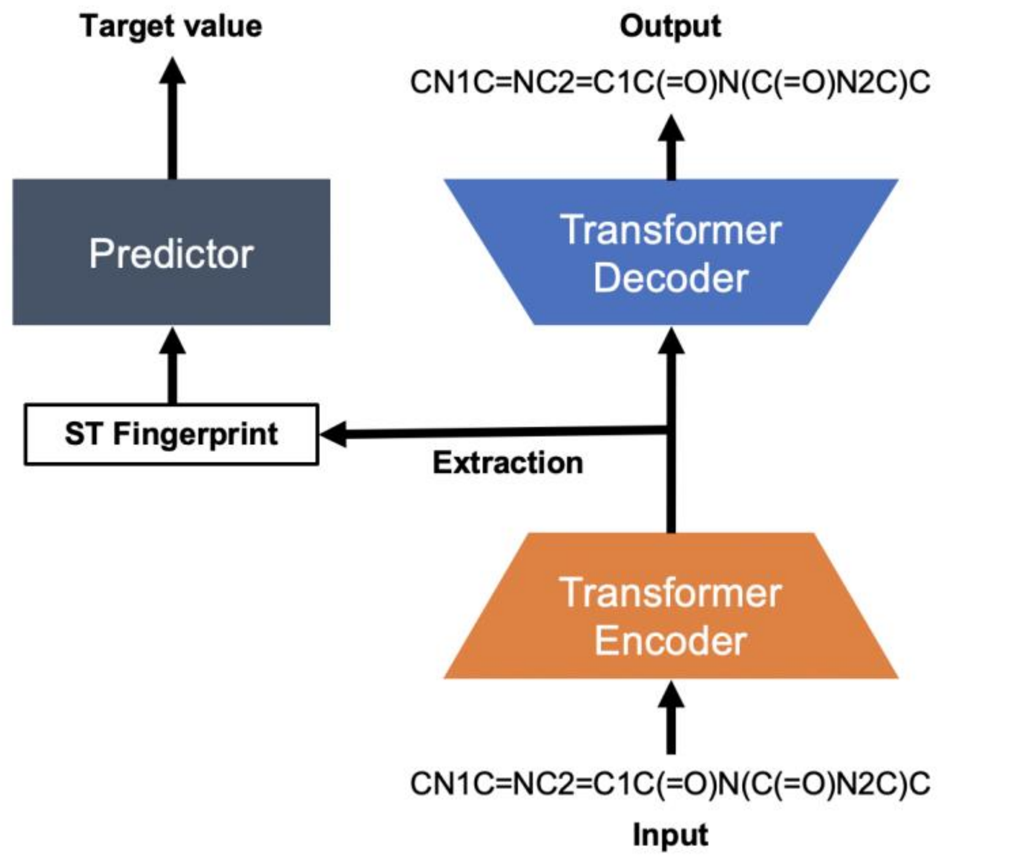
TTS(LSTM)



TTS(transformer)

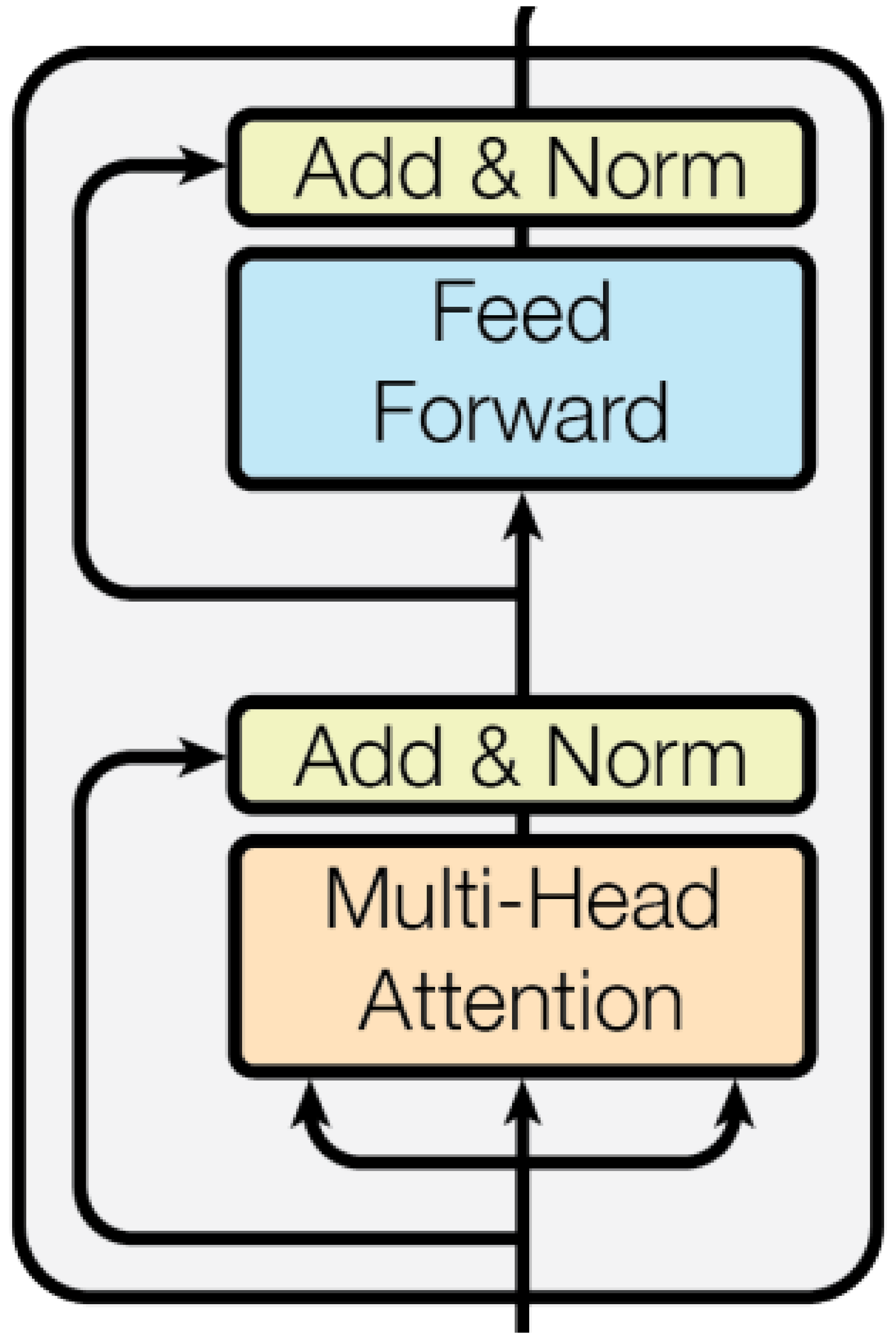


MolBART



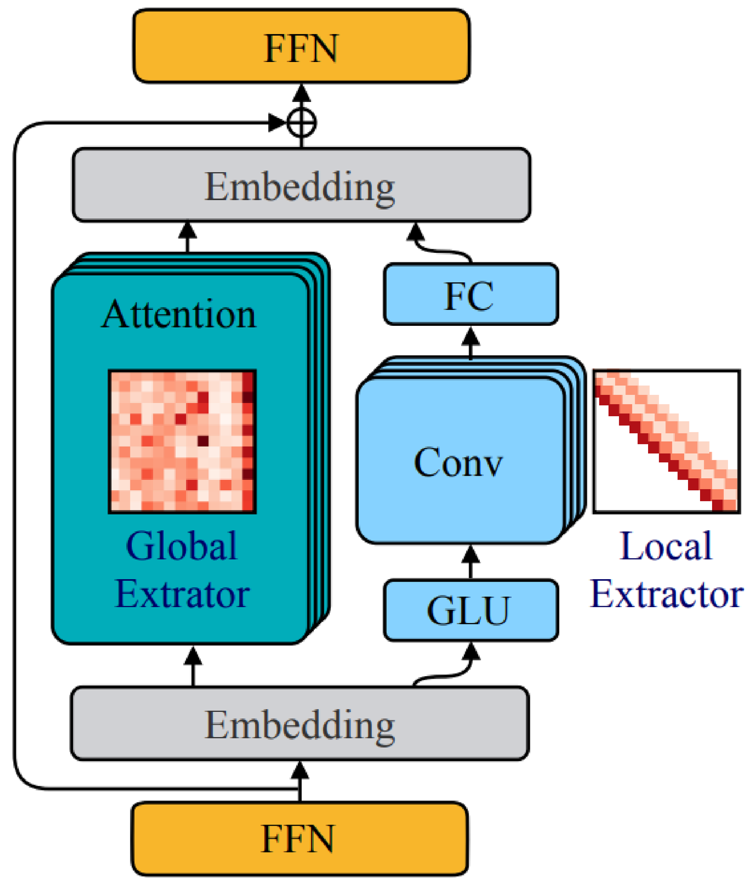
Chemical(transformer)

Various Transformer Layers



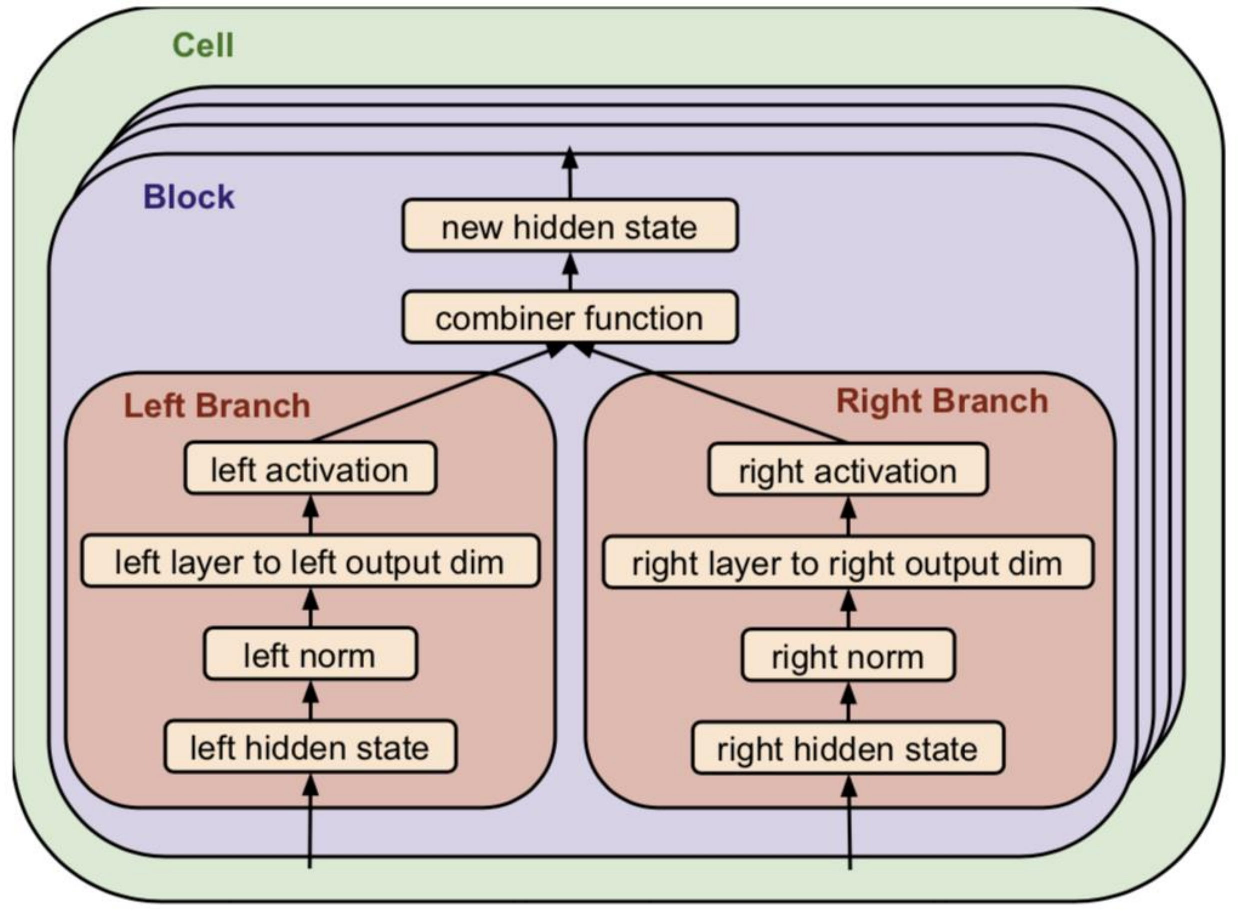
Replace
FF, MHA
Change order

Lite Transformer



(a) Lite Transformer block

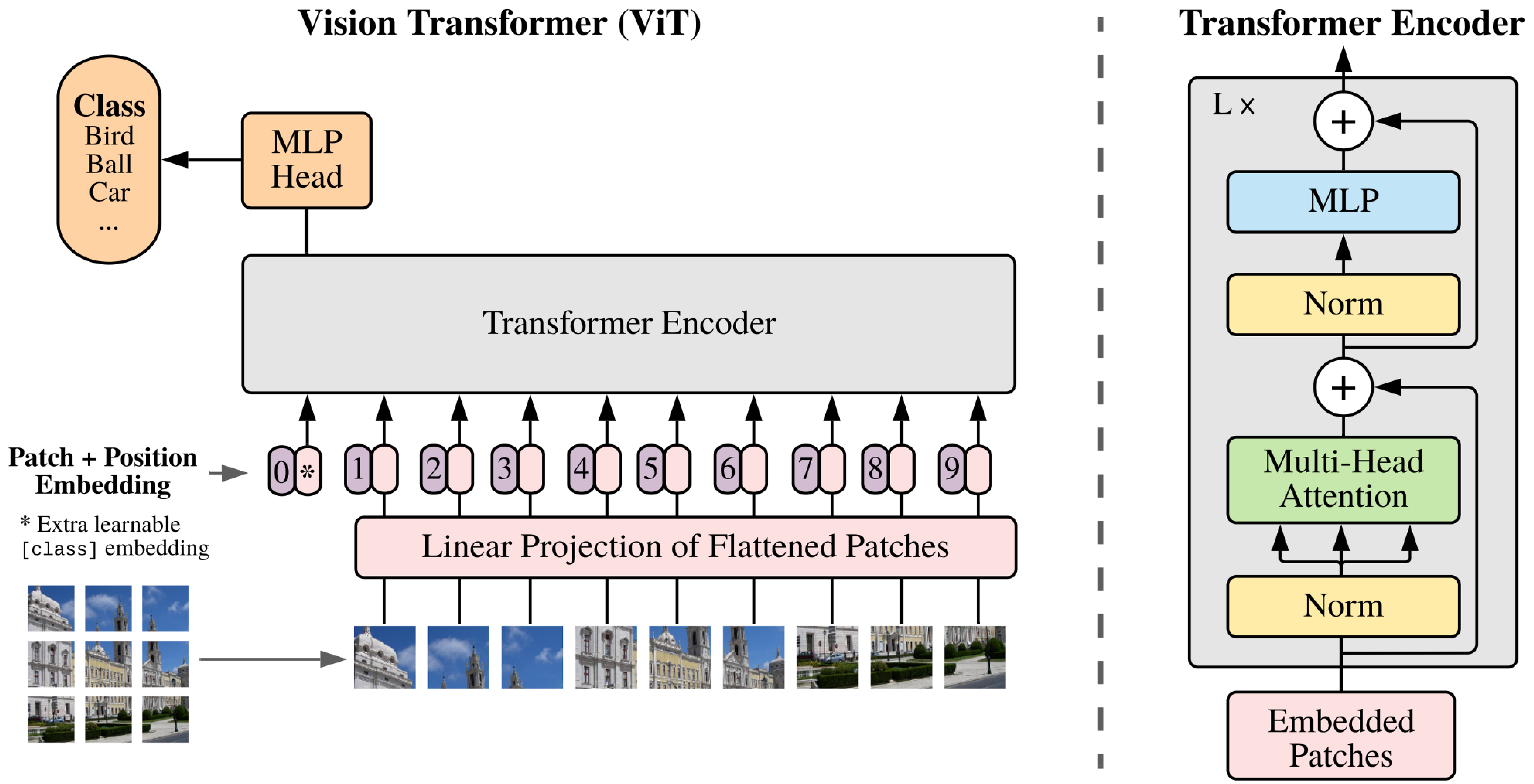
Evolved Transformer(NAS)



Sparse Attention
Axial Attention
Graph Attention
Quaternion Transformer

Longformer
Linformer
Reformer
Performer

Vision Transformer(ViT) ICLR2021



TRANSFORMERS

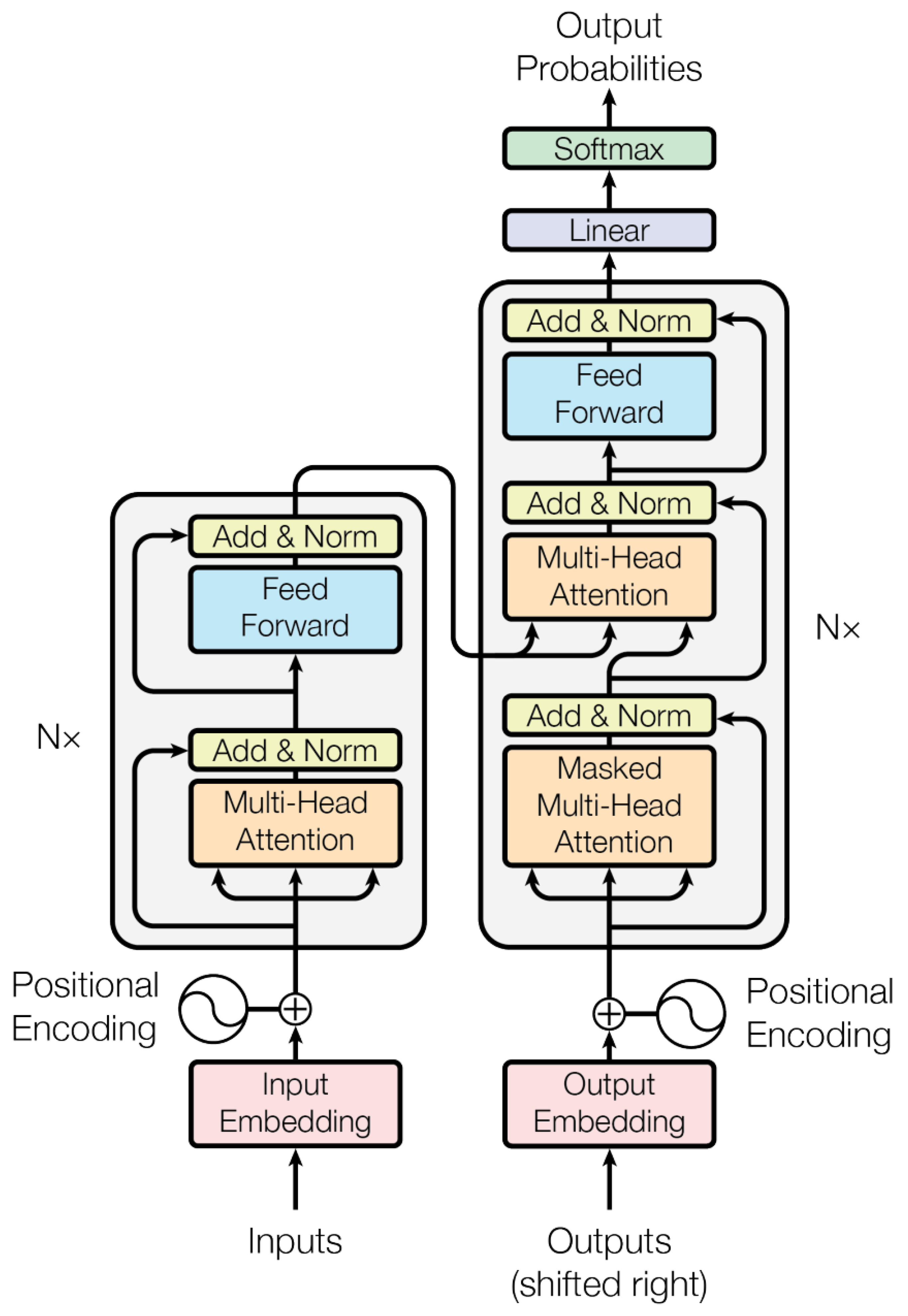
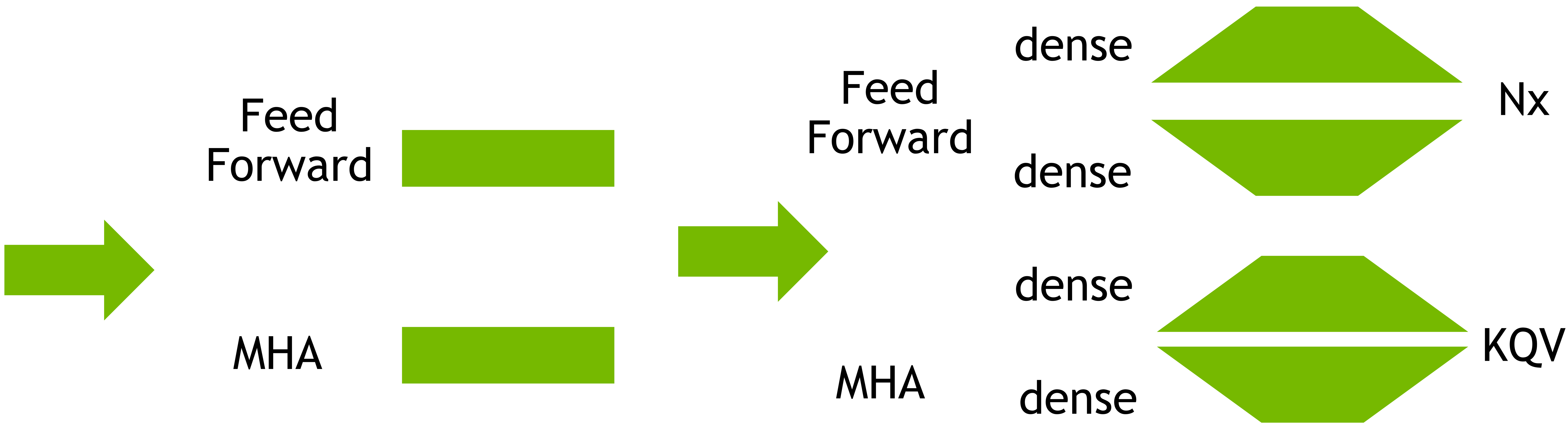


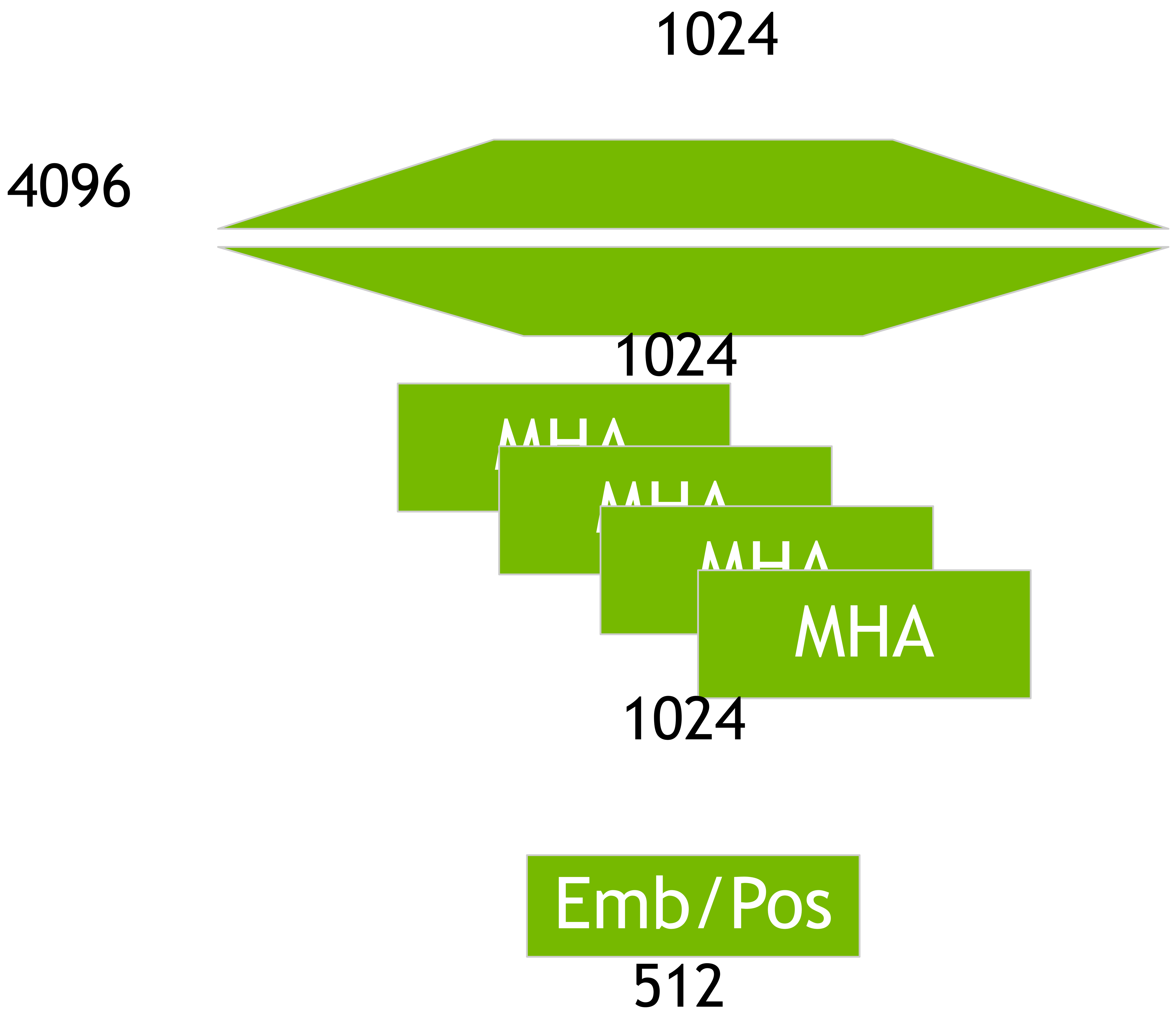
Figure 1: The Transformer - model architecture.

Attention Is All You Need



BERT BASE

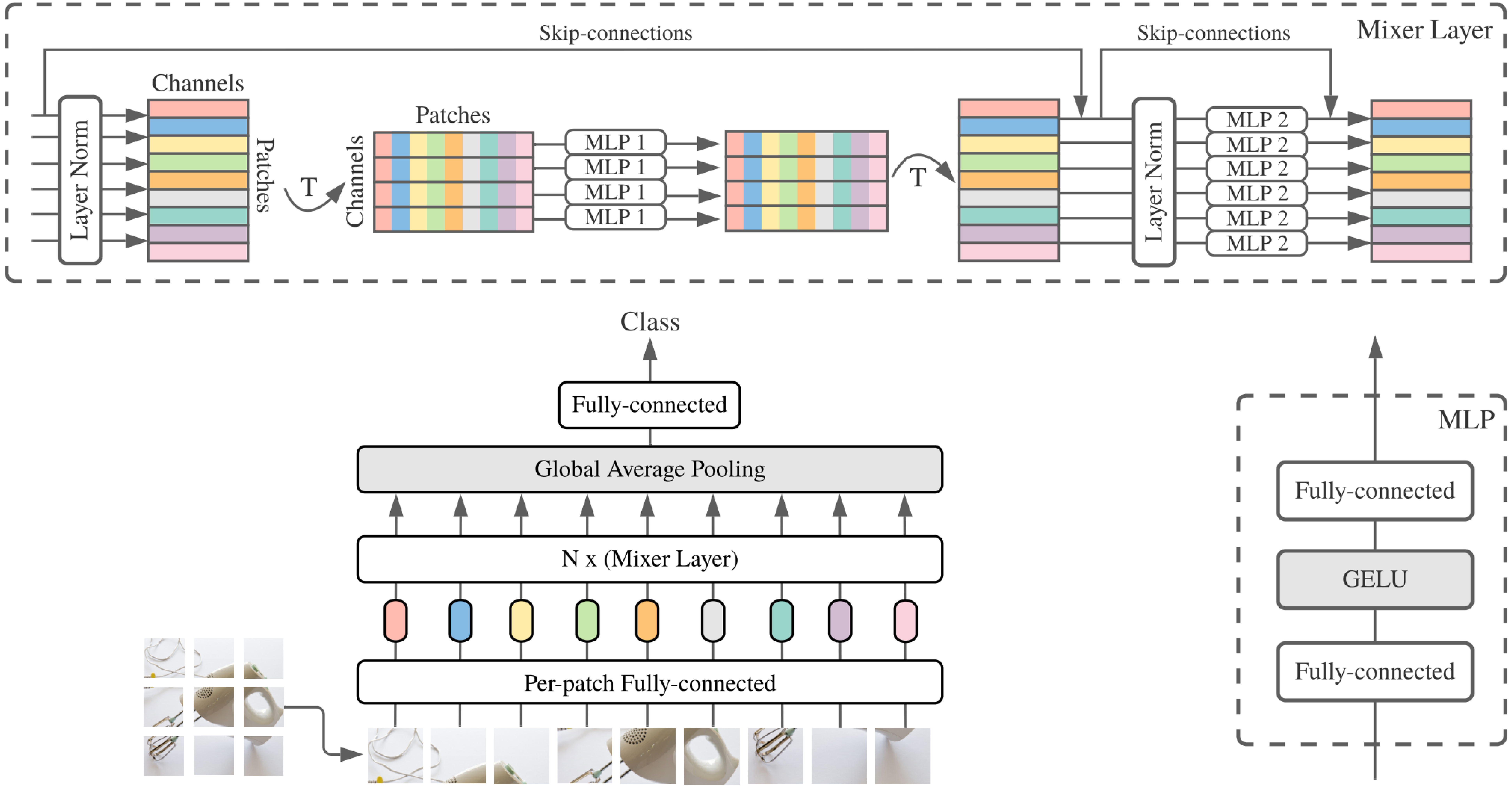
BERT BASE	BERT LARGE
Pos : 512	Pos : 512
numVOCA= 2^15	numVOCA= 2^15
NumLayers: 12	NumLayers: 24
dimModel : 768	dimModel 1024
dimHead :64	dimHead :64
NumHeads : 12	NumHeads : 16
Act : gelu	Act : gelu
Dropout : 0.1	Dropout : 0.1
FF scale : 4	FF scale : 4
110M Param	340M Param



NVIDIA

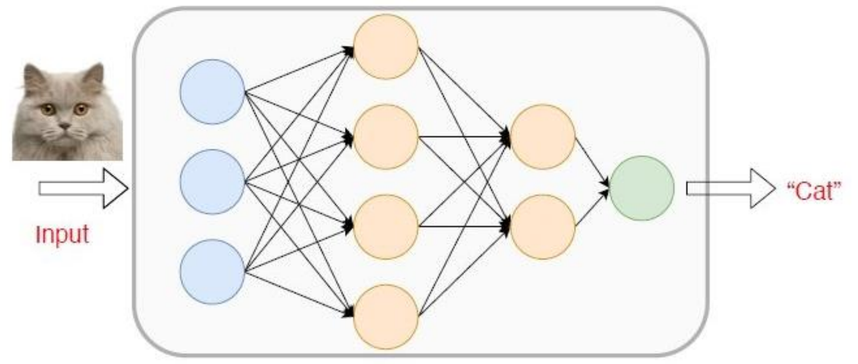
MLP-Mixer

MLP-Mixer: An all-MLP Architecture for Vision
https://arxiv.org/pdf/2105.01601.pdf



REVISIT MLP

MLP

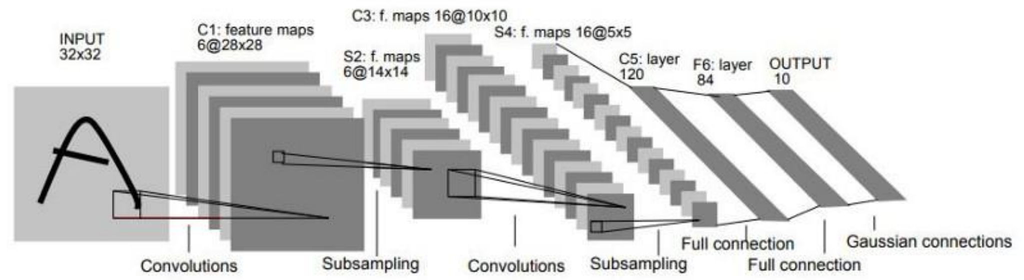


flatten
raw
1d input

MLP
sigmoid

softmax

CNN



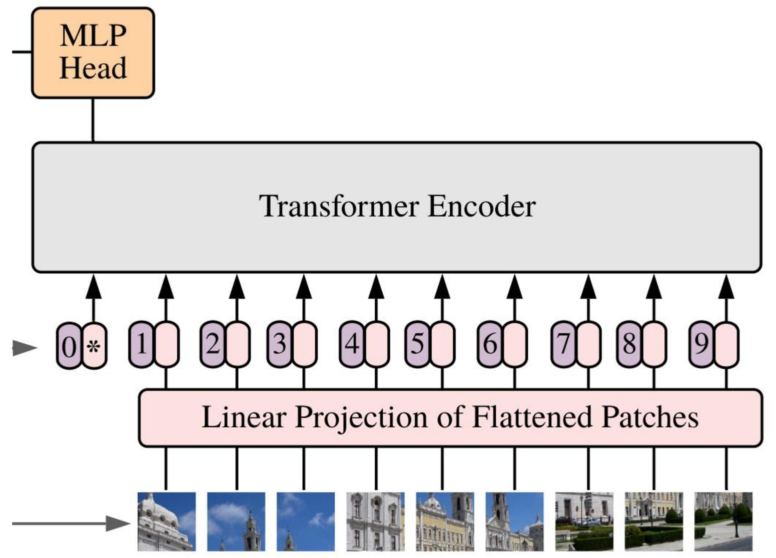
2d input

features
2d conv
relu

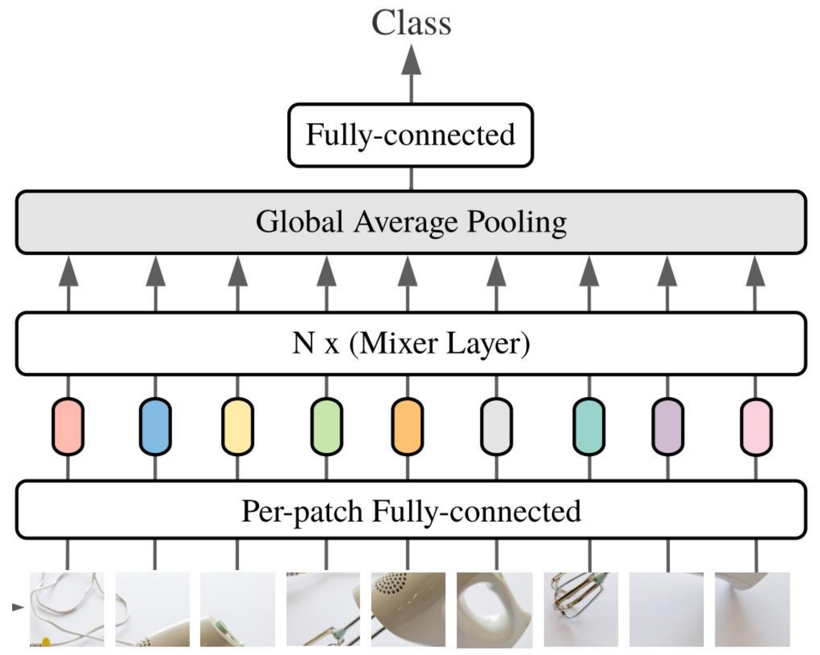
Onehot
encoding

Residual
SELayer

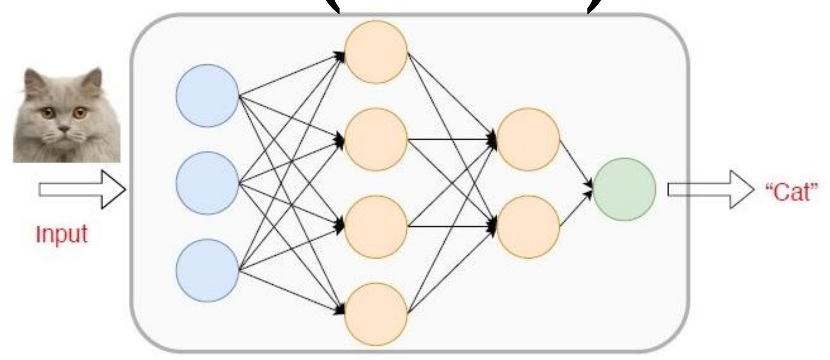
Transformer



MLP-Mixer



MLP(new)



encoded
1d input

repeat n
residual

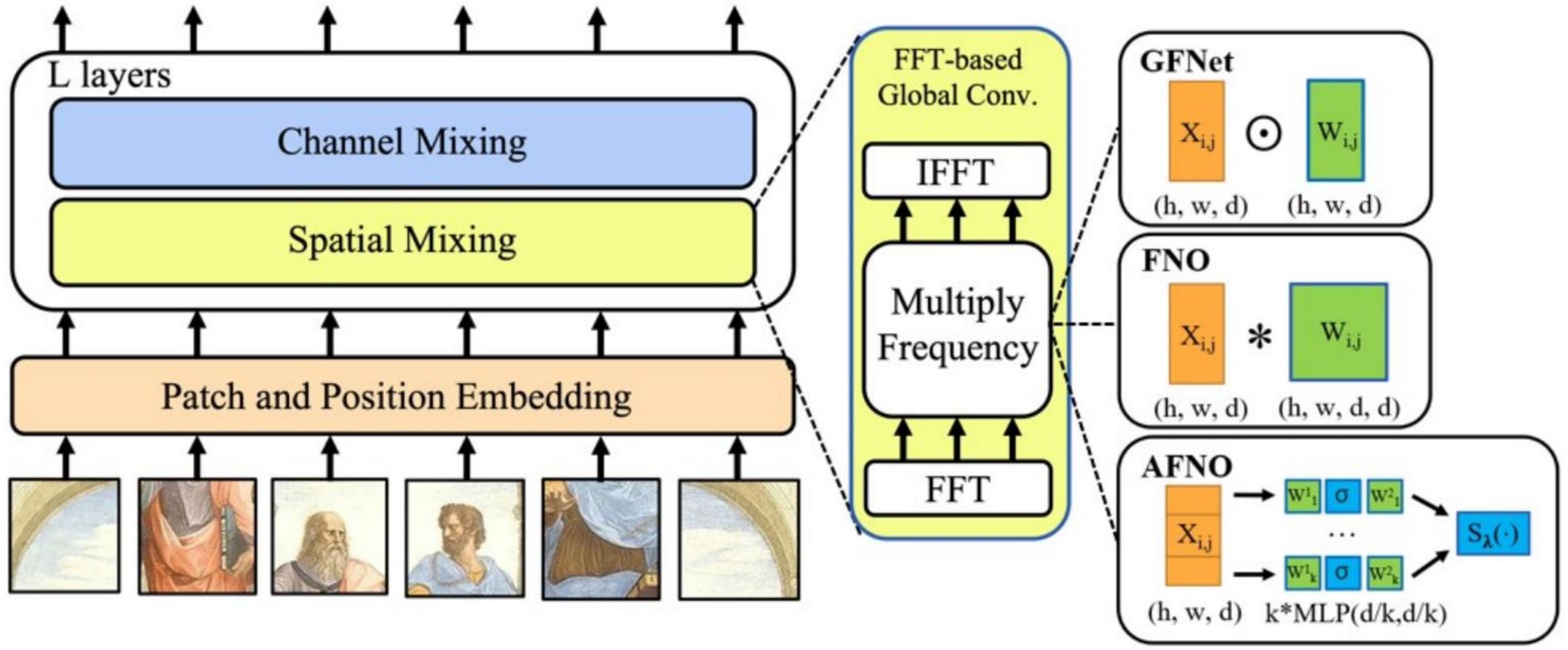
relu/gelu

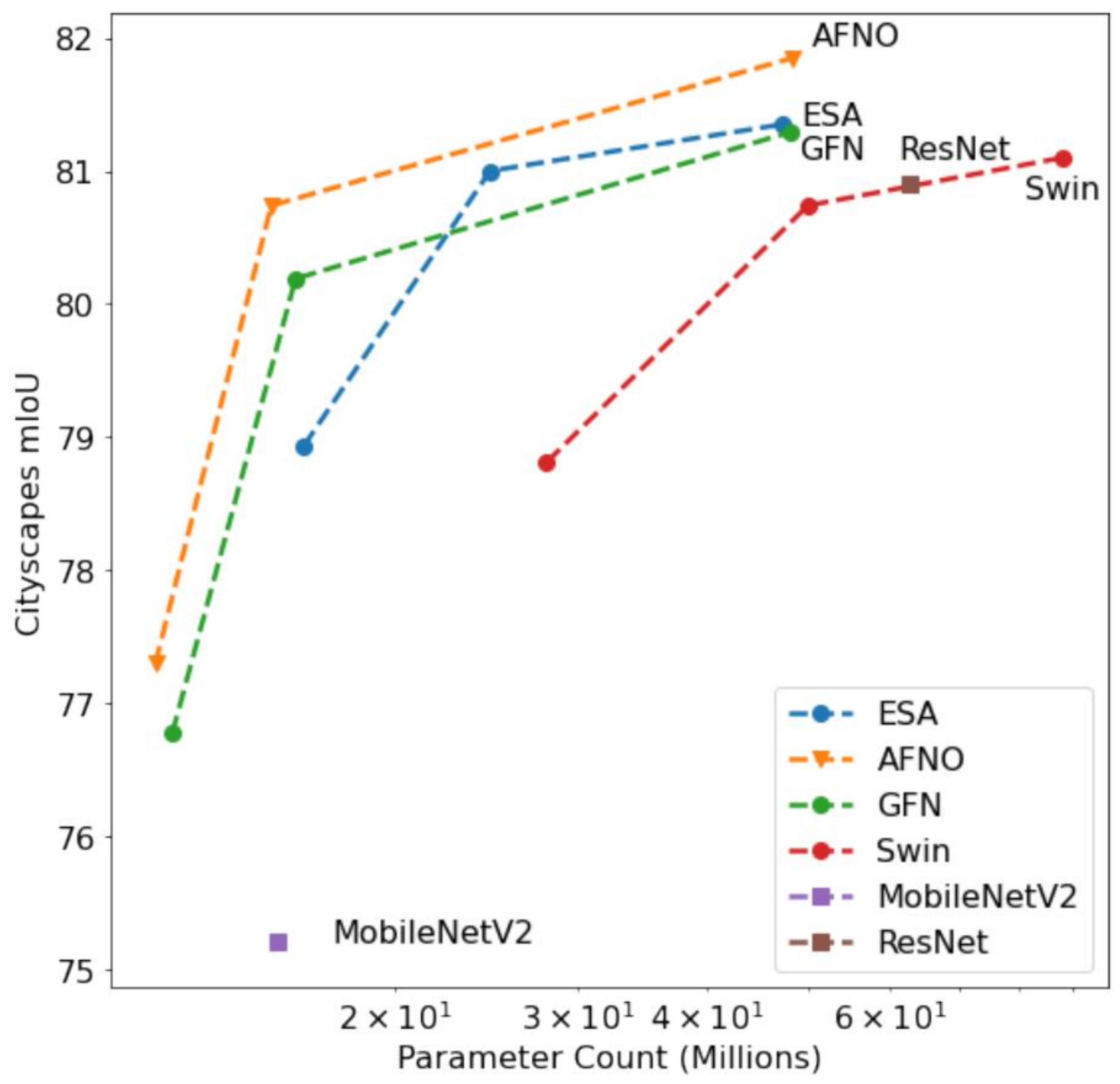
layernorm
dropout

Softmax
Onehot
encoding

AFNO (ICLR 2022)
Adaptive Fourier Neural Operators

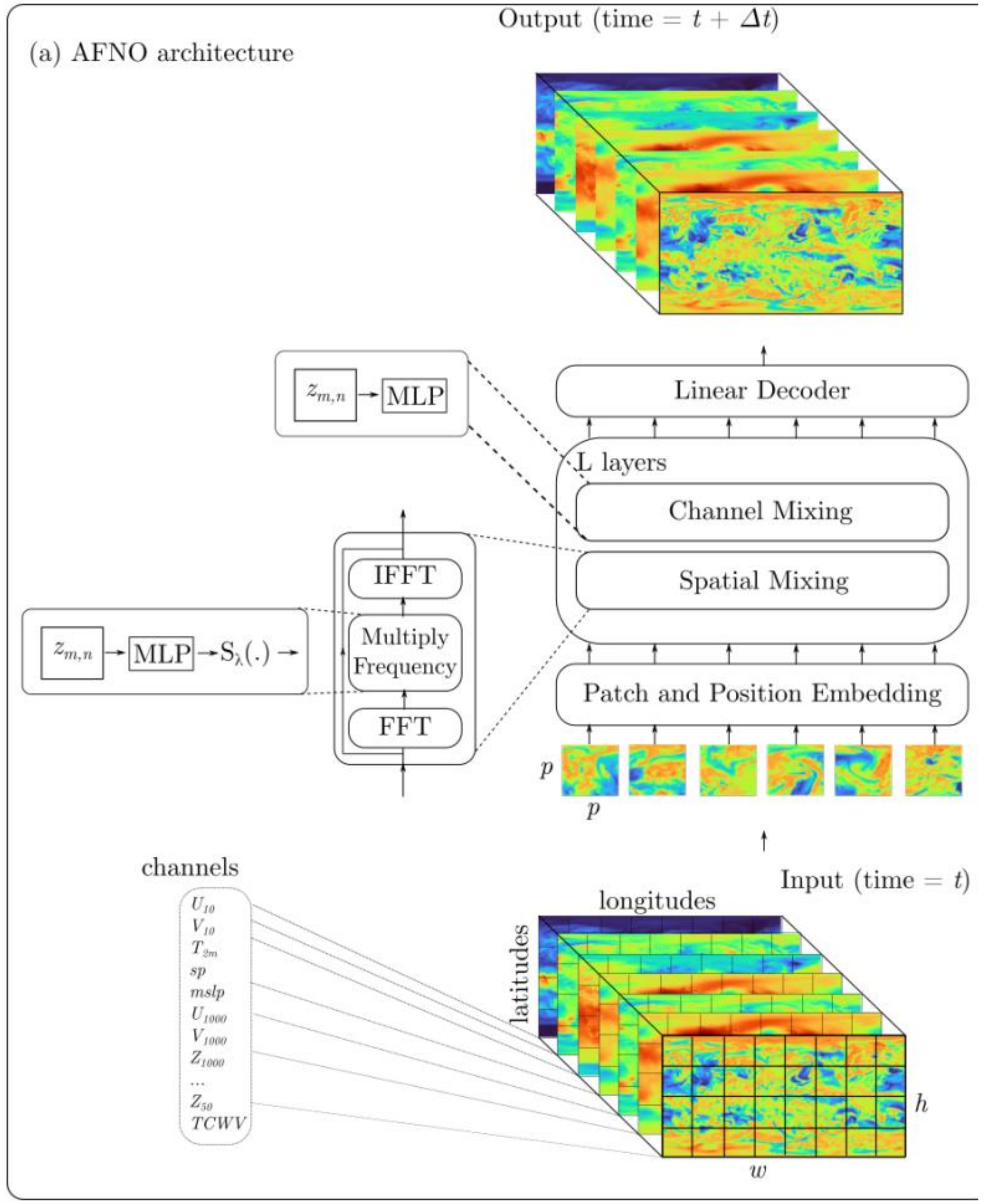
MLP-Mixer with FFT





FourCastNet

https://arxiv.org/pdf/2202.11214.pdf



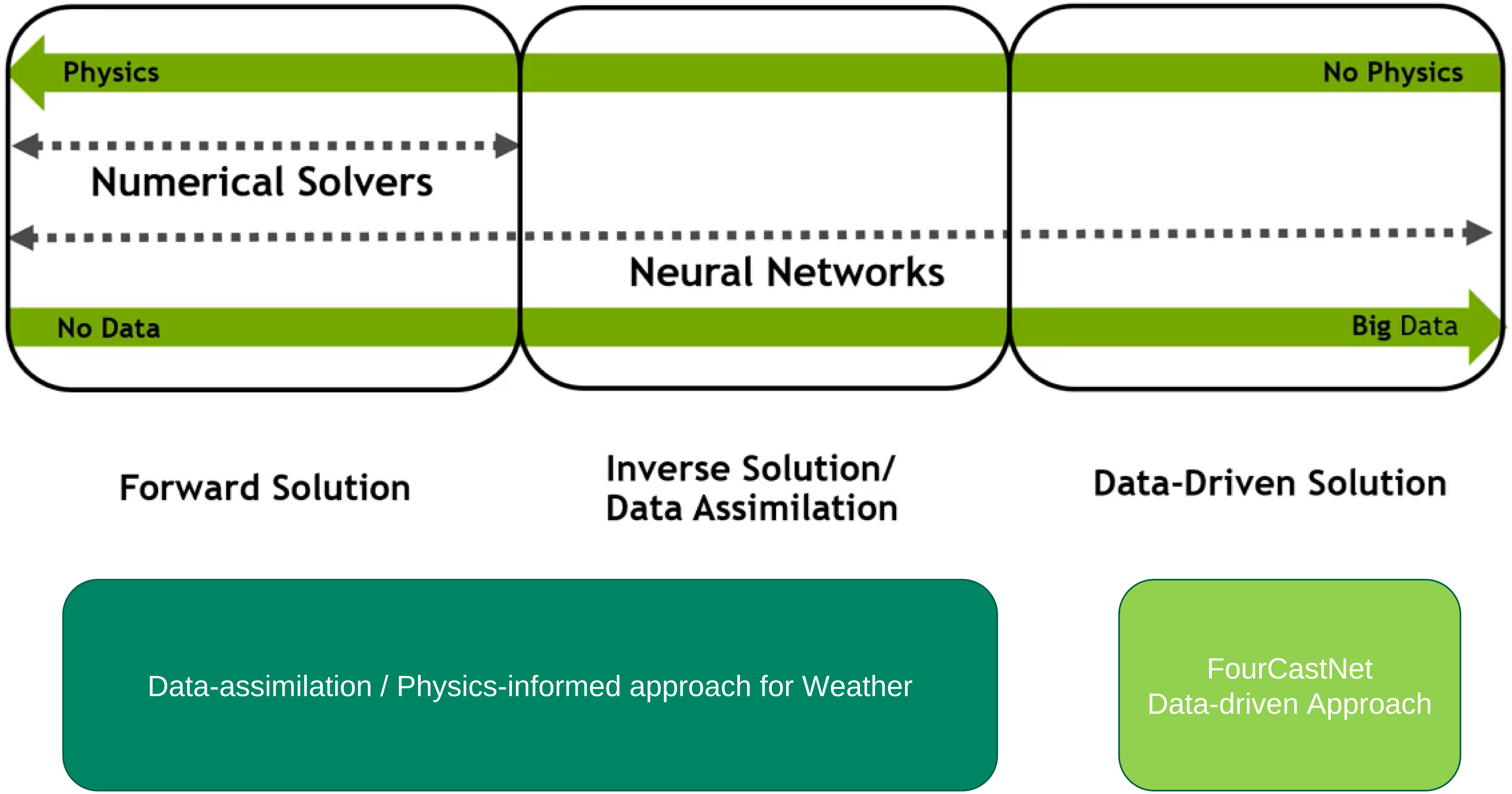
Use AFNO for weather modeling(NWP)

FourCastNet generates a week-long forecast in less than 2 seconds

FourCastNet is about 45,000 times faster than traditional NWP models on a node-hour basis

PINN

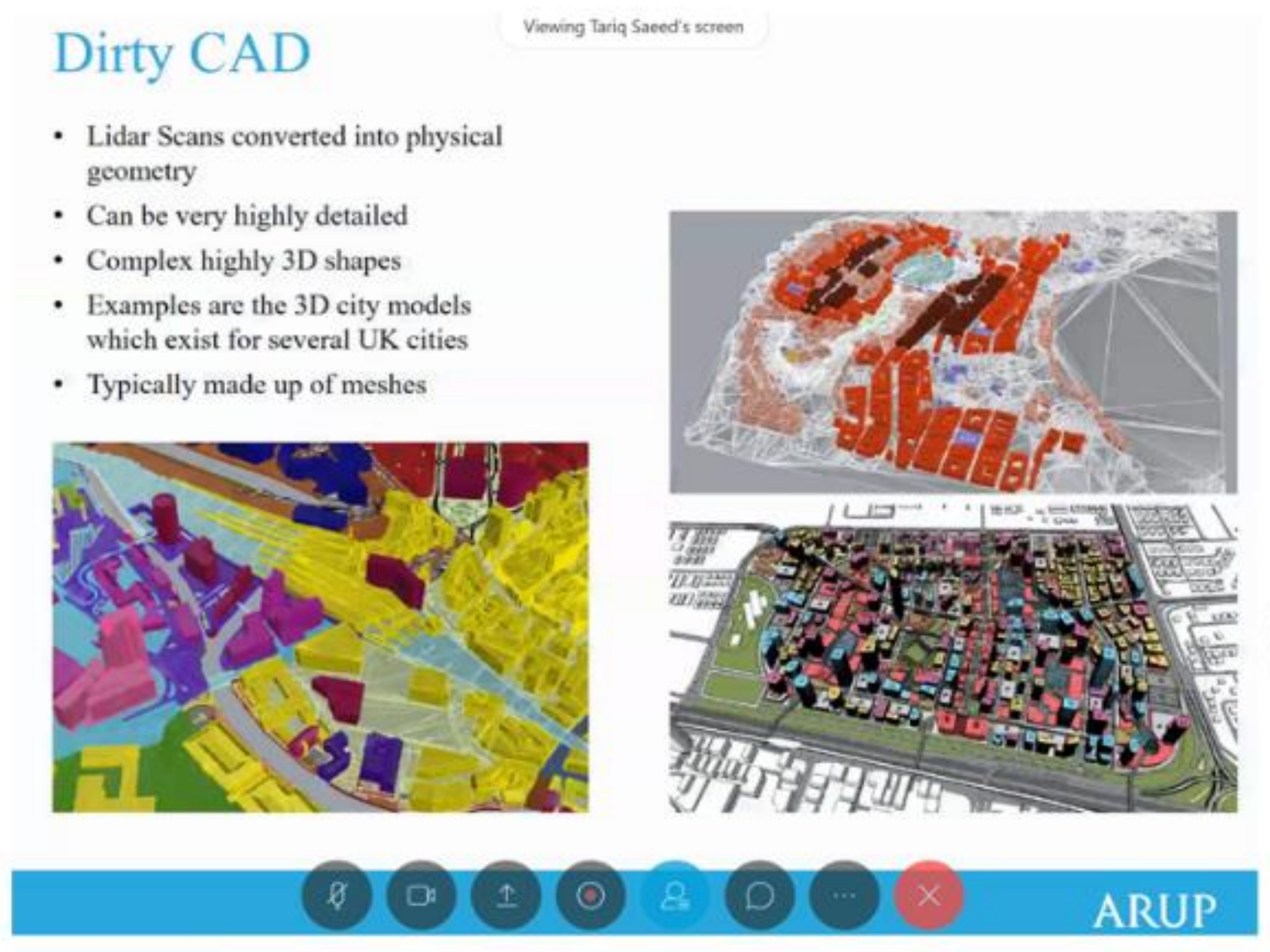
MODULUS IN A GLANCE.

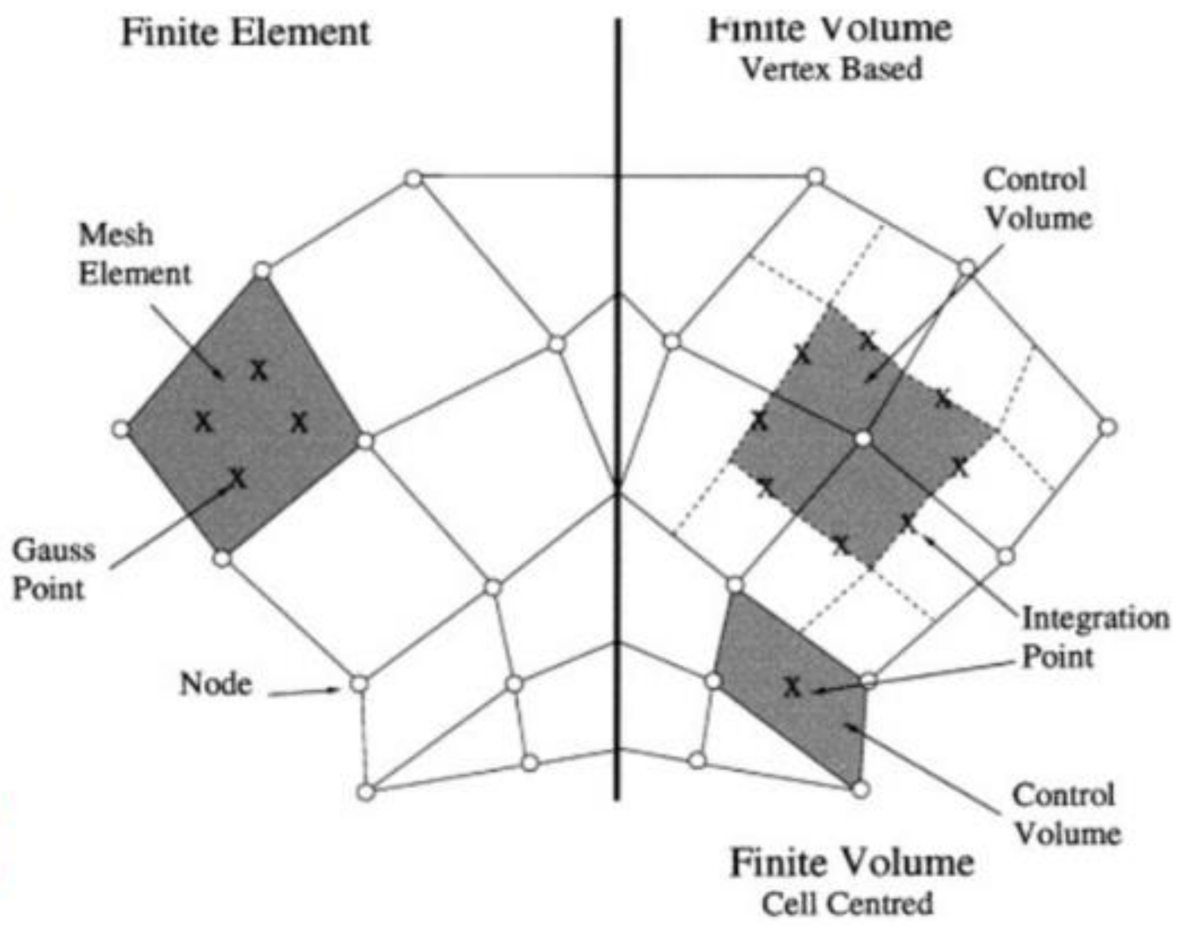


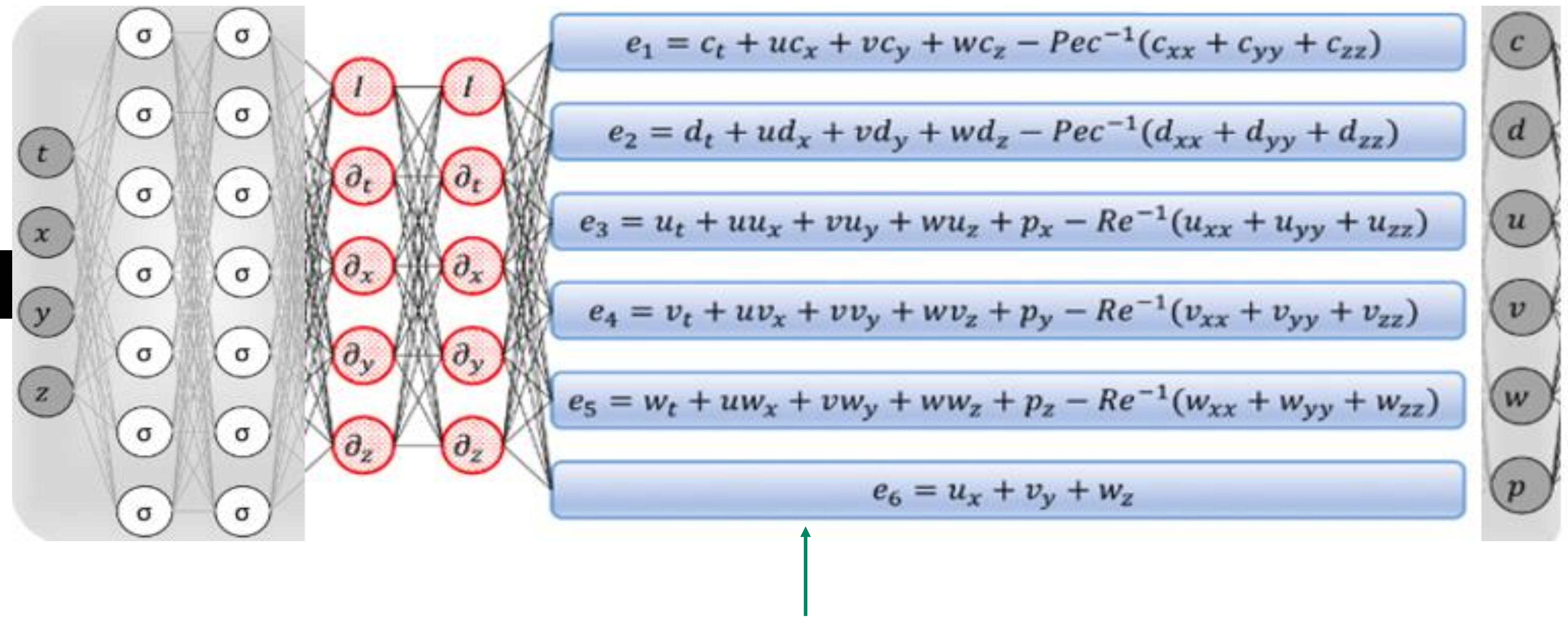
PHYSICS INFORMED NEURAL NETS: ARCHITECTURE

A Neural Network Architecture for Computational Mechanics/Physics problems

- ❑ Point Cloud for 3D Geometries
- ❑ Physics Driven & Physics Aware Networks (respects the governing PDEs, Multi-disciplinary)
- ❑ Performance optimized for GPU tensor cores

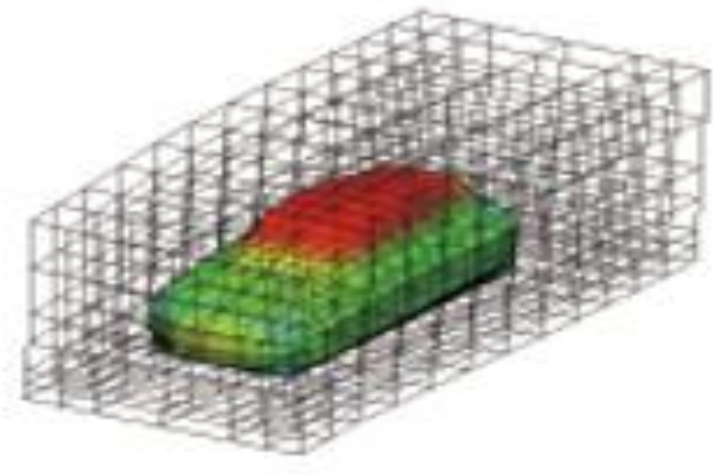






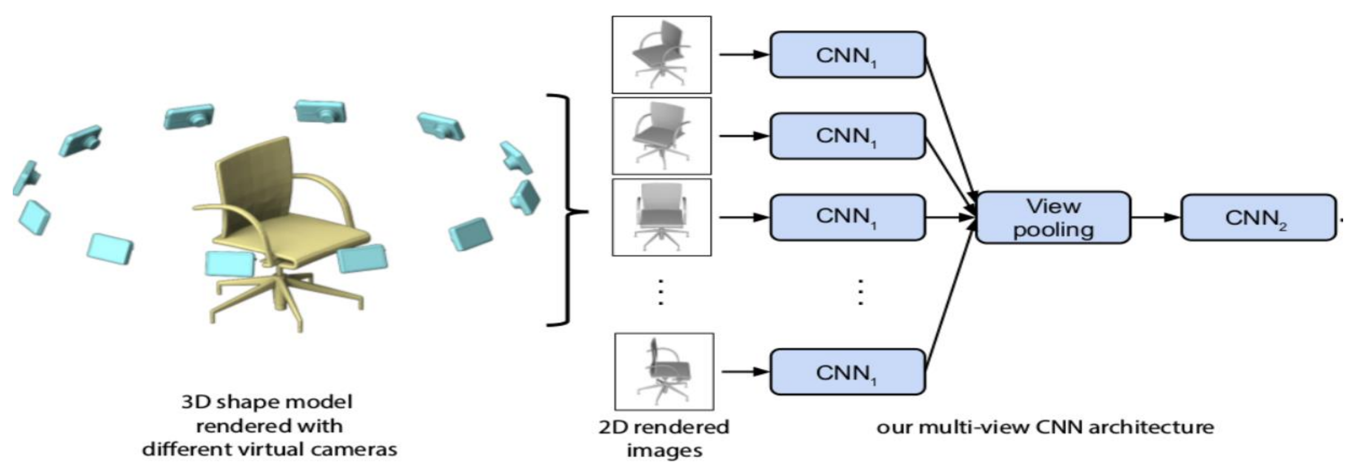
SHAPE PARAMETERIZATION

- **Voxels**



- Good for CNNs but memory intensive for high resolution, cannot represent geometry well and has quantization effects

- **Multi-View**



- Unable to capture fine geometry details & gradients and completely unsuitable for Physics problems

✓ **Point Cloud**

Input: 3D Scene
Point cloud

- 1:1 correspondence with analysis data format
- Works for uneven density and unstructured meshes. Perfect for Physics problems

- **Poly Cube**

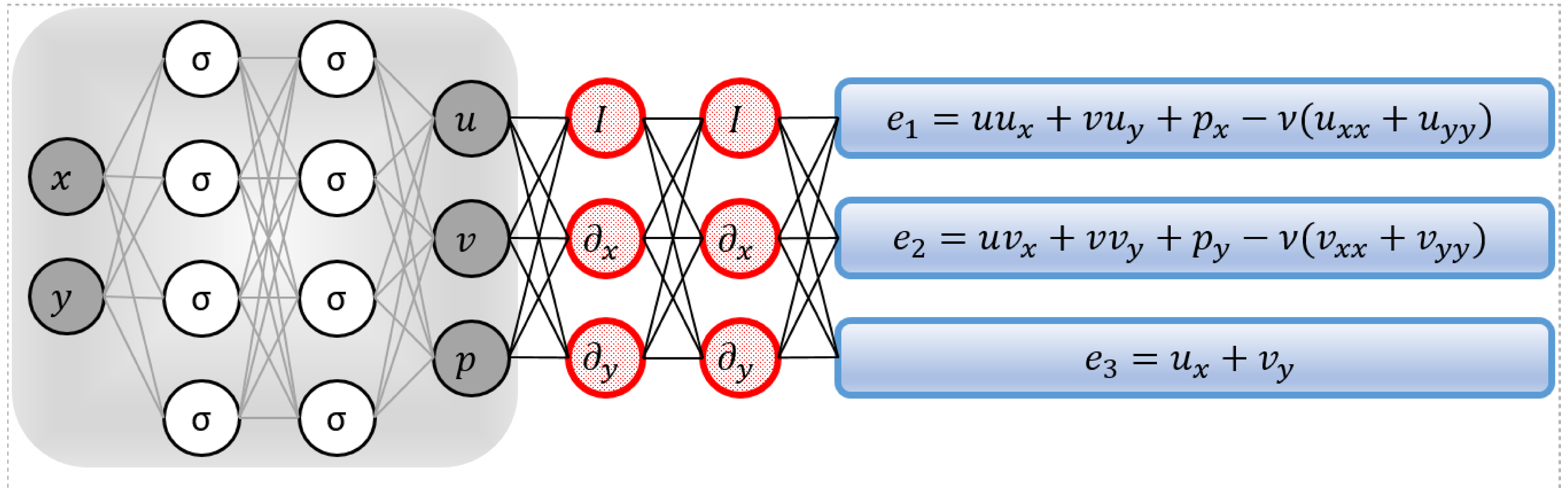
level-1

- Will require integration into CAD tools in order to regenerate uniform mesh and then invokes CNN
- Will retain the deficiencies of Voxel based CNNs
- Does not address legacy analysis results

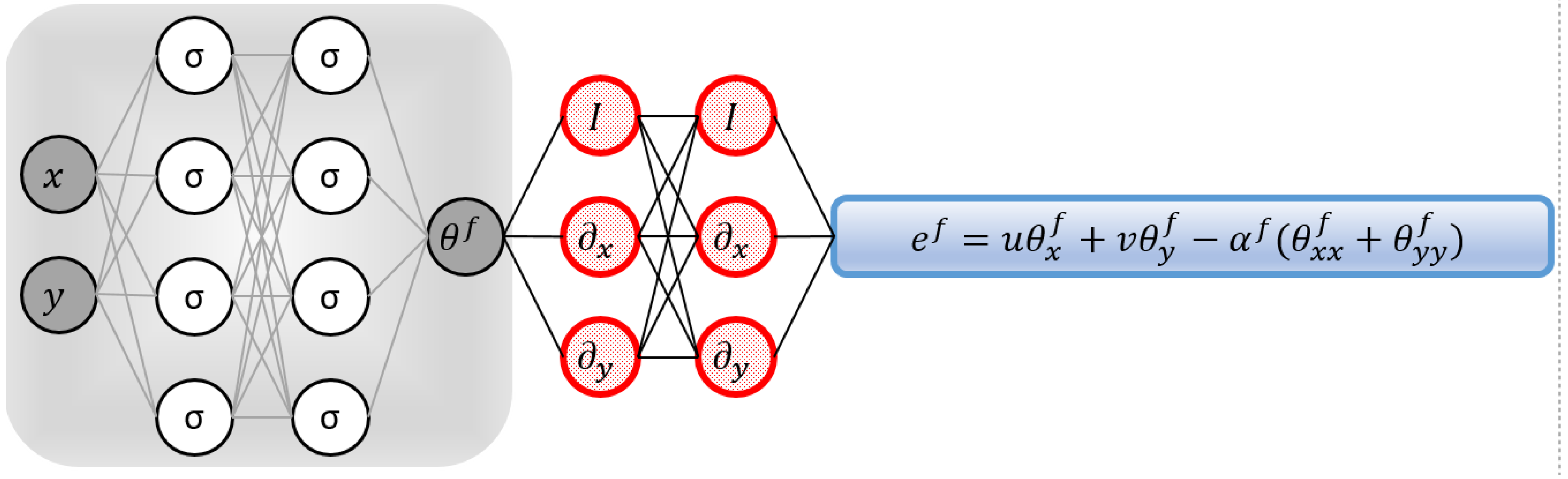
AI TRAINING ENGINE

Multi-Physics Neural Networks

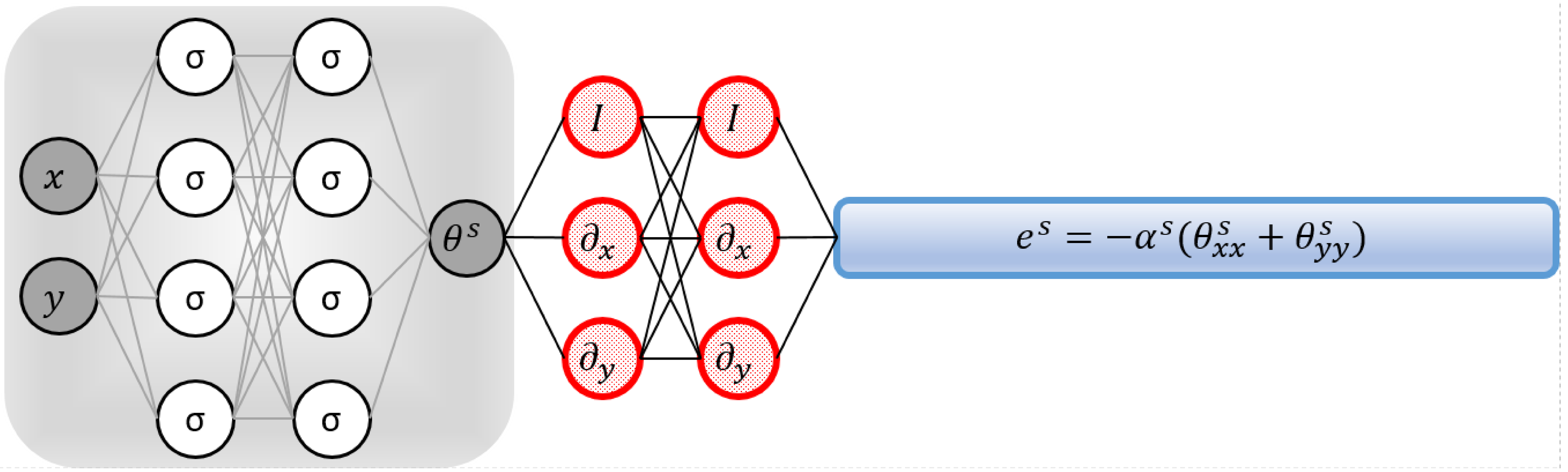
CFD (turbulent)



Heat Transfer in Fluid



Heat Transfer in Solid



Multi-Physics PDEs

CFD (with turbulence) – 2nd Order PDE
Heat Transfer in Solids & Fluid

Fluid-Solid Interface Conditions

$\theta^f = \theta^s$ Temperature

$\kappa^f(\theta^f_x n_x + \theta^f_y n_y) = \kappa^s(\theta^s_x n_x + \theta^s_y n_y)$ Heat Flux

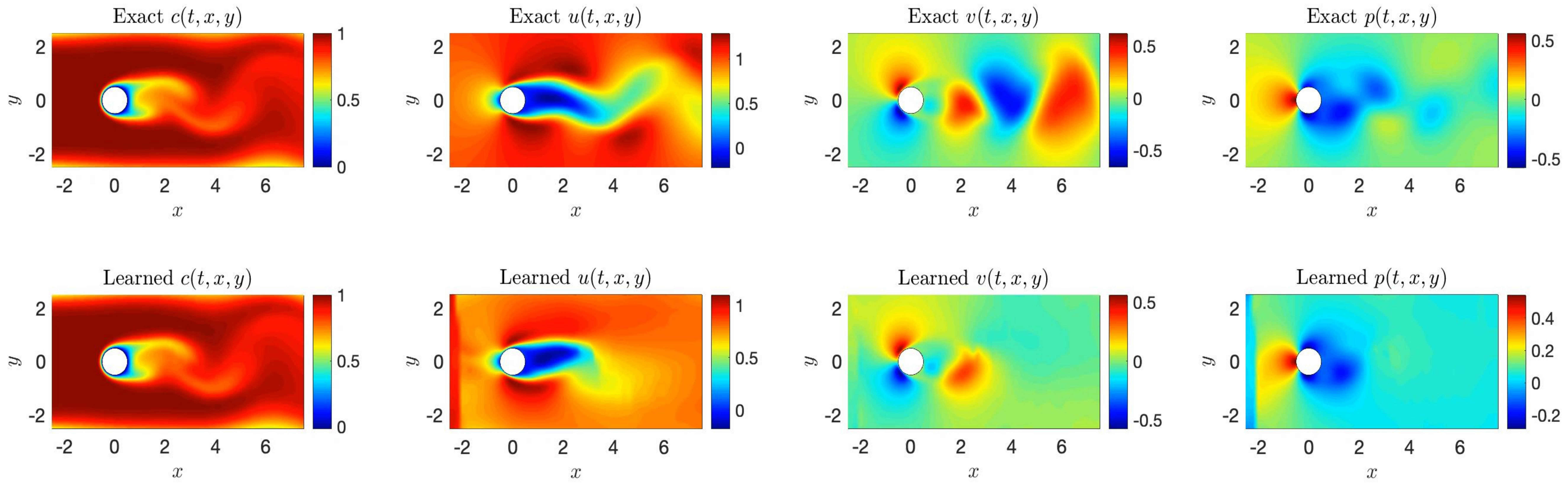
PINN Network Architecture

10 layers for non-Physics Informed Network
10 x 2n layers for n^{th} order PDEs
50 neurons per layer
Swish Activation Function

EXTERNAL FLOW PAST A CYLINDER – LEARNT VS. GROUND TRUTH

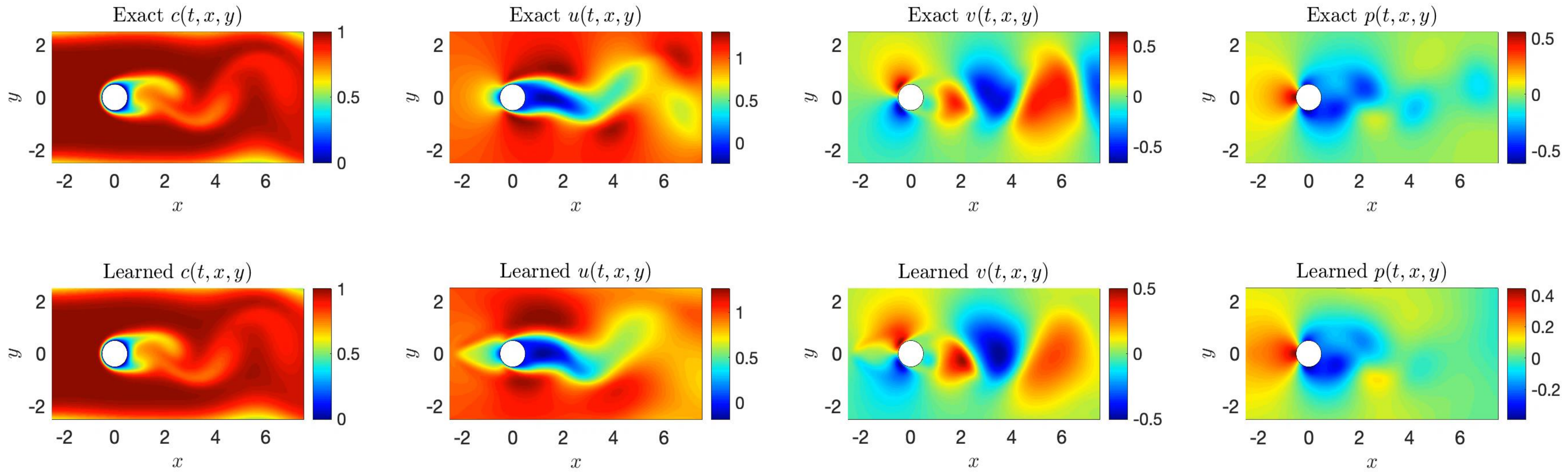
CFD Simulation of an **External Flow over a Cylinder** with OpenFOAM –

A user error was incidentally discovered by the PINNs that presented itself as a mismatch between the Simulation & AI result !!!

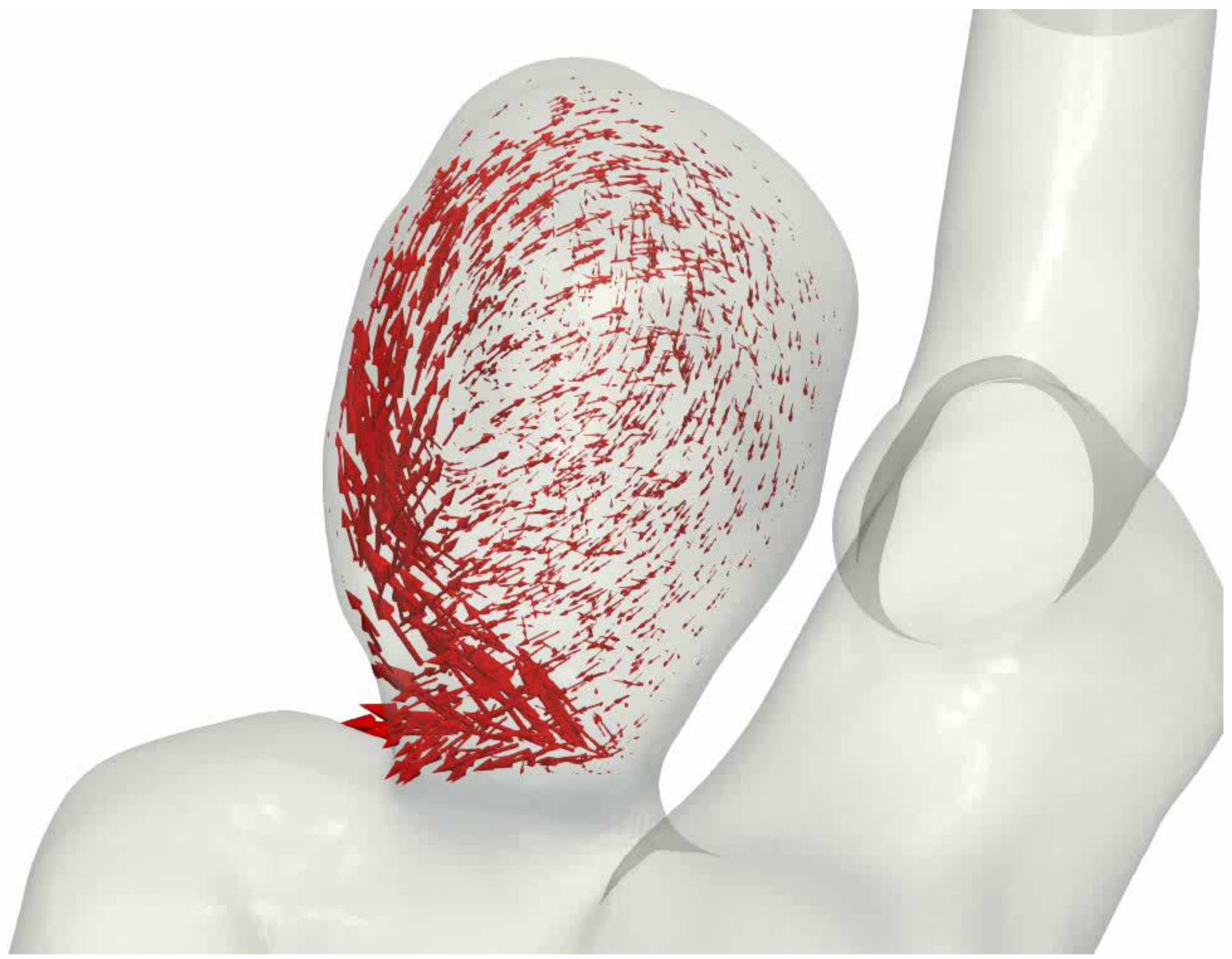


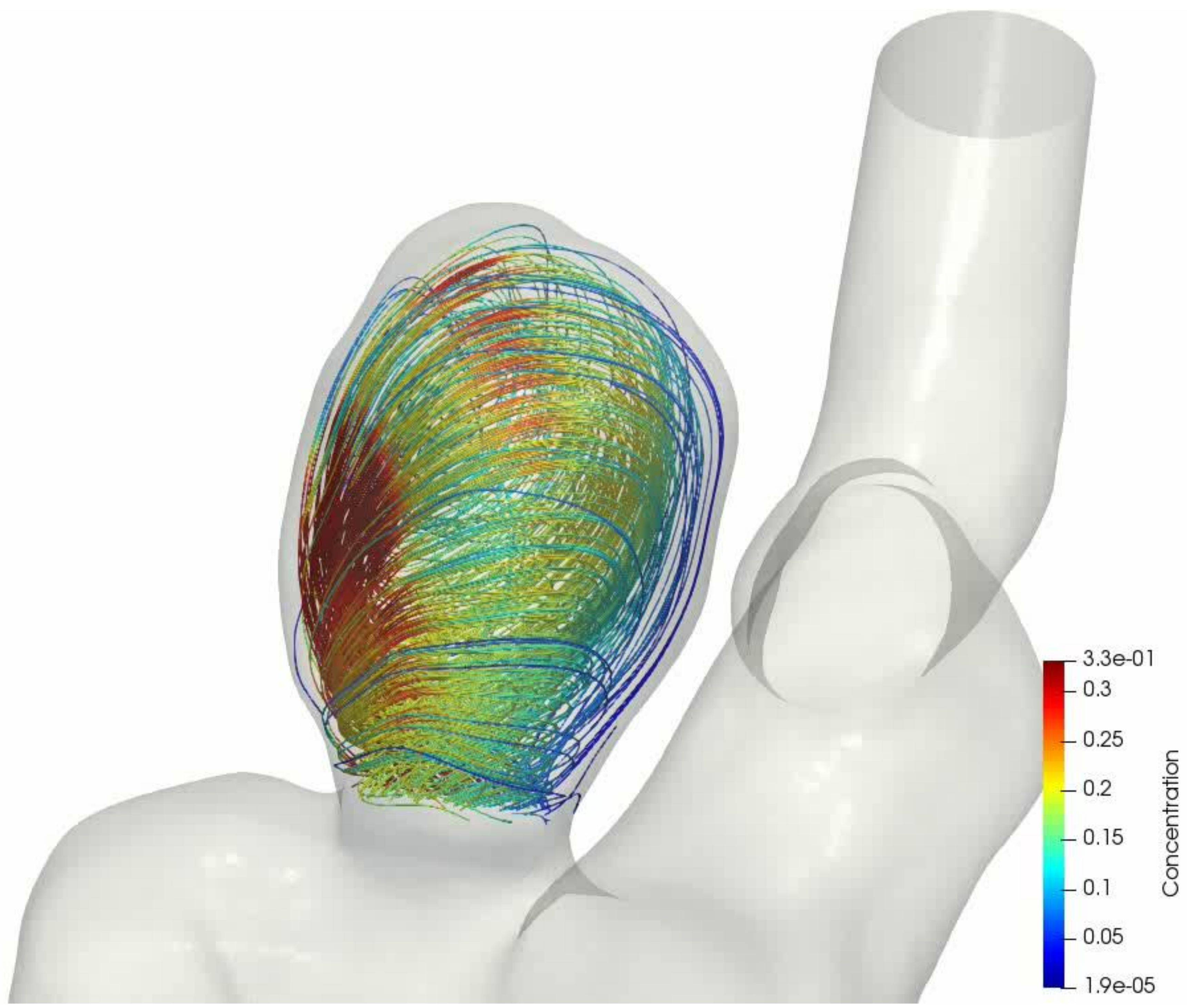
Correct CFD Simulation Results with OpenFOAM (Ground Truth)

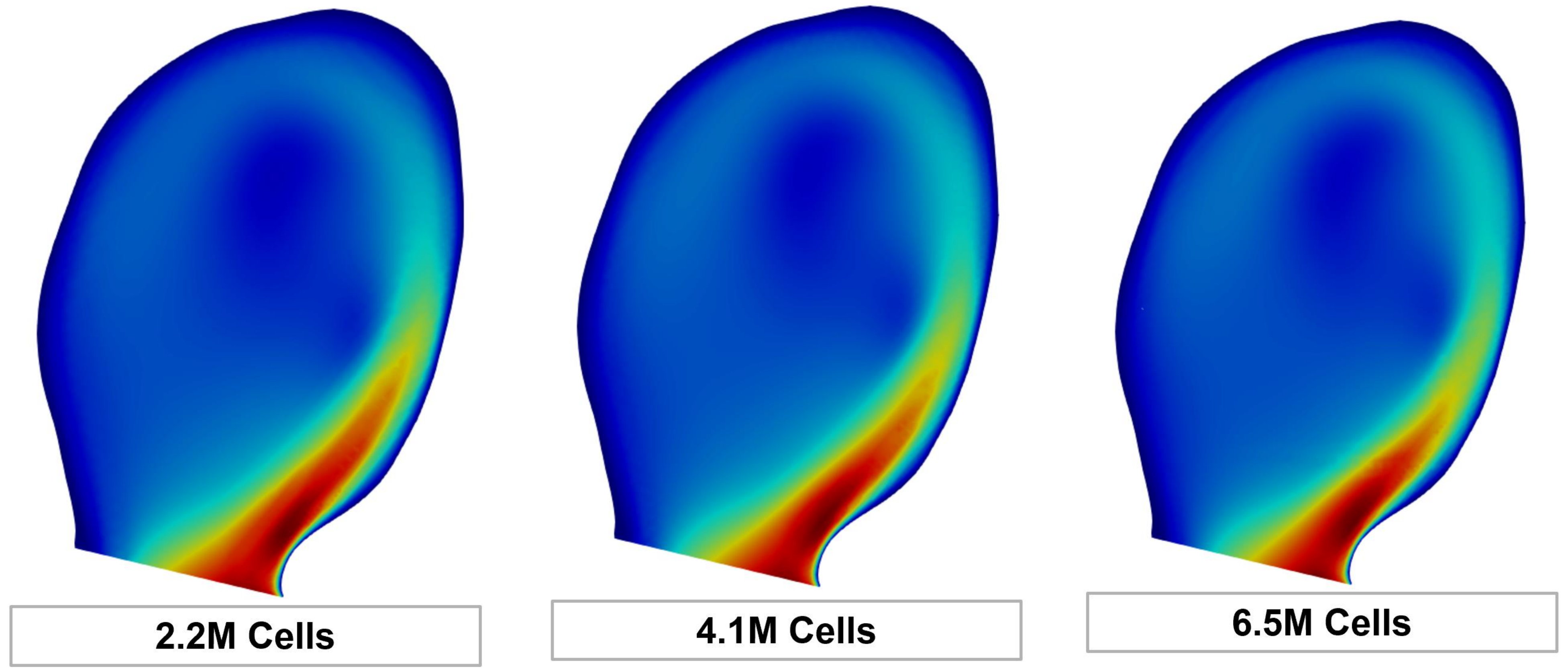
Correct Predictions



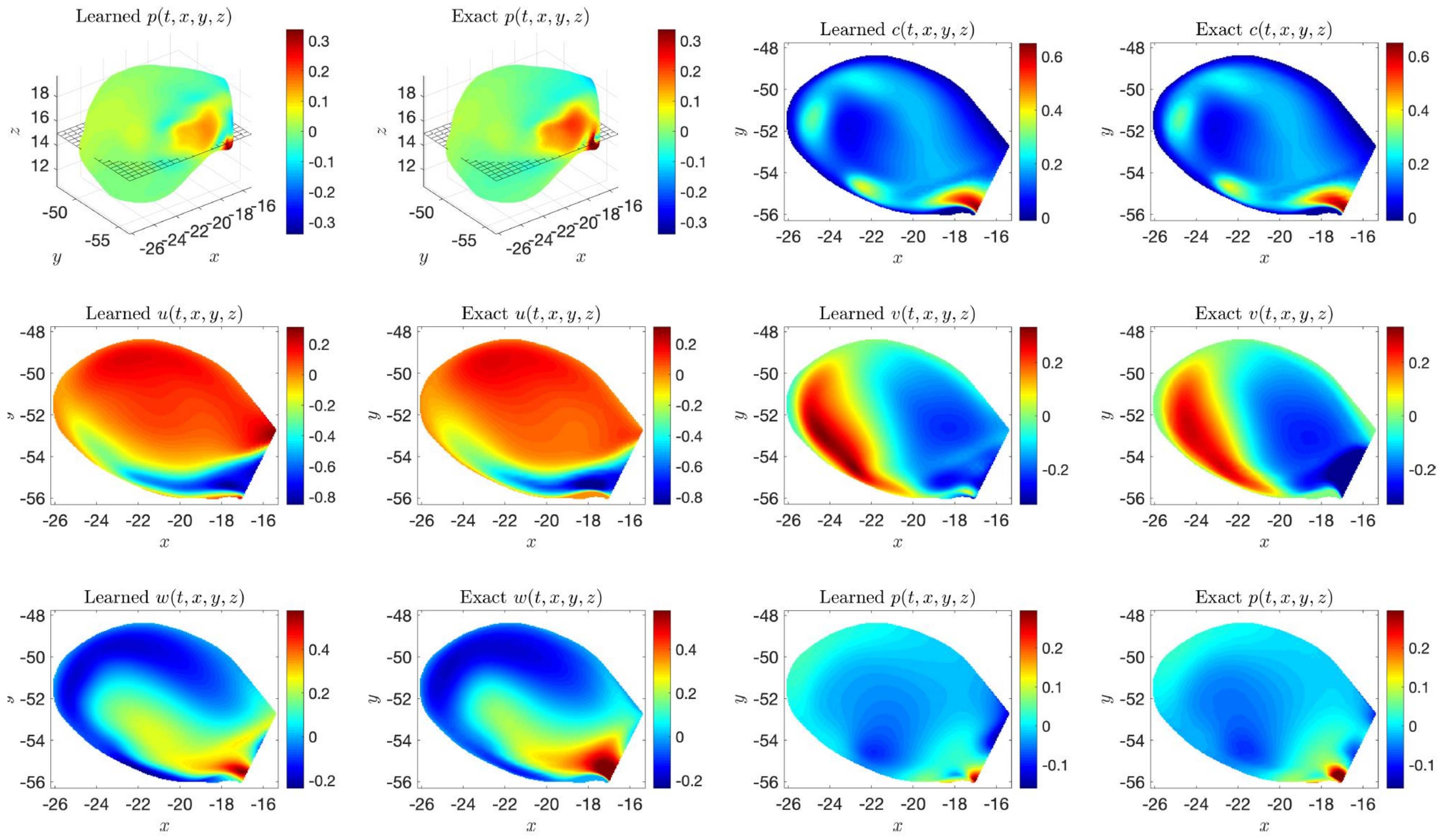
MEDICAL IMAGING: INTRACRANIAL CEREBRAL ANEURYSM (ICA)



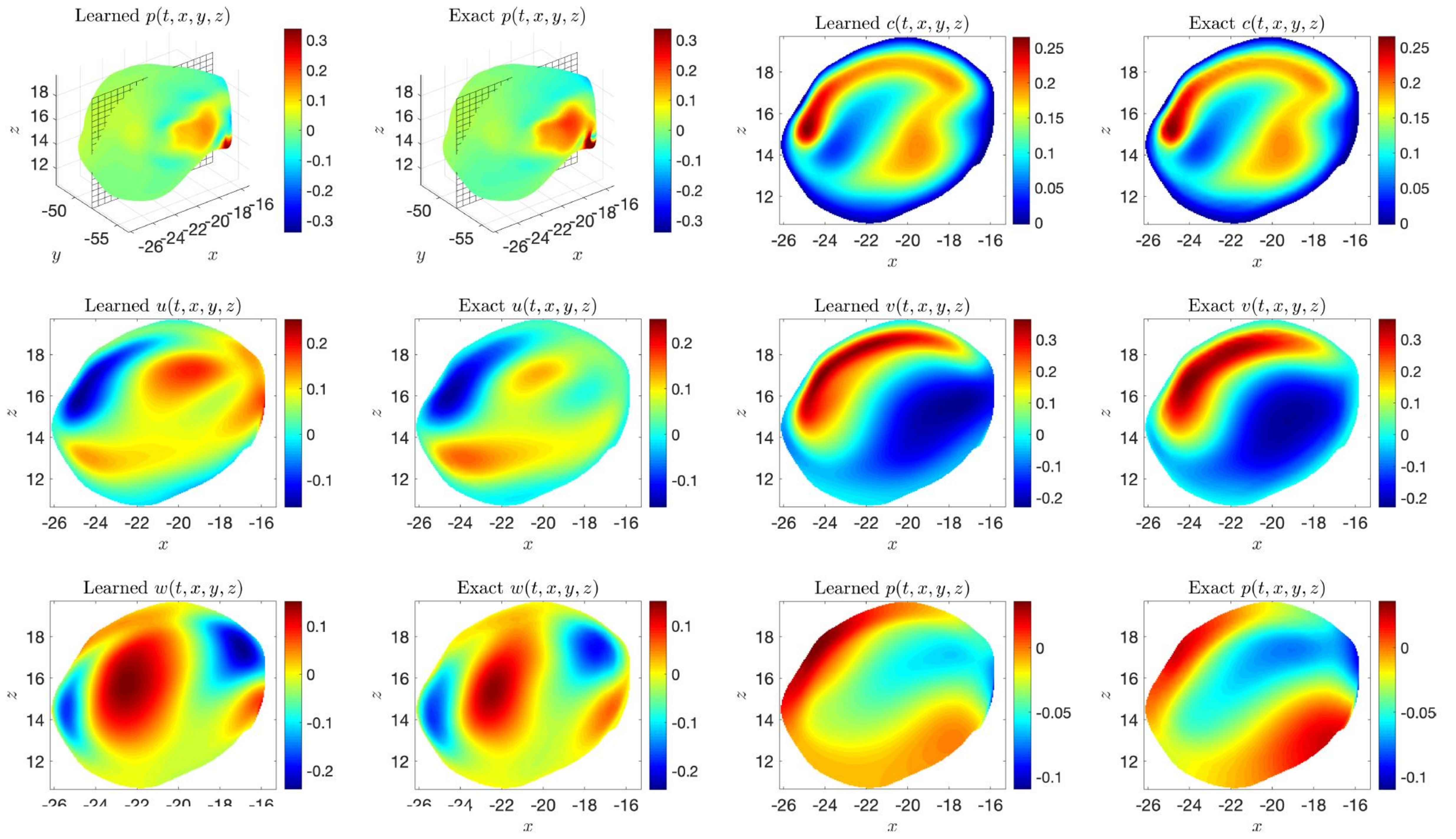




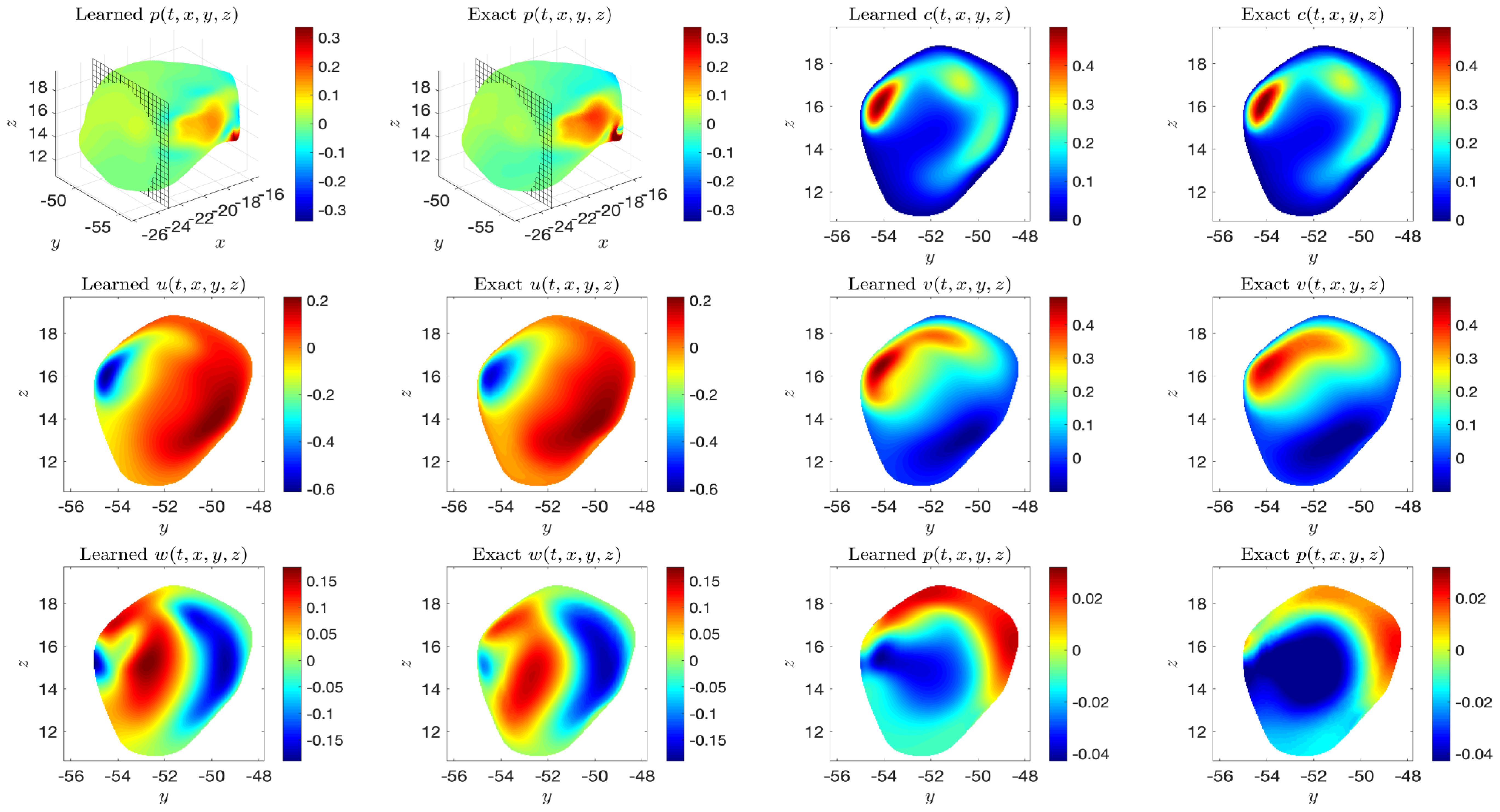
ICA – COMPARISON BETWEEN SIMULATION & NN



Cut along Z-Plane

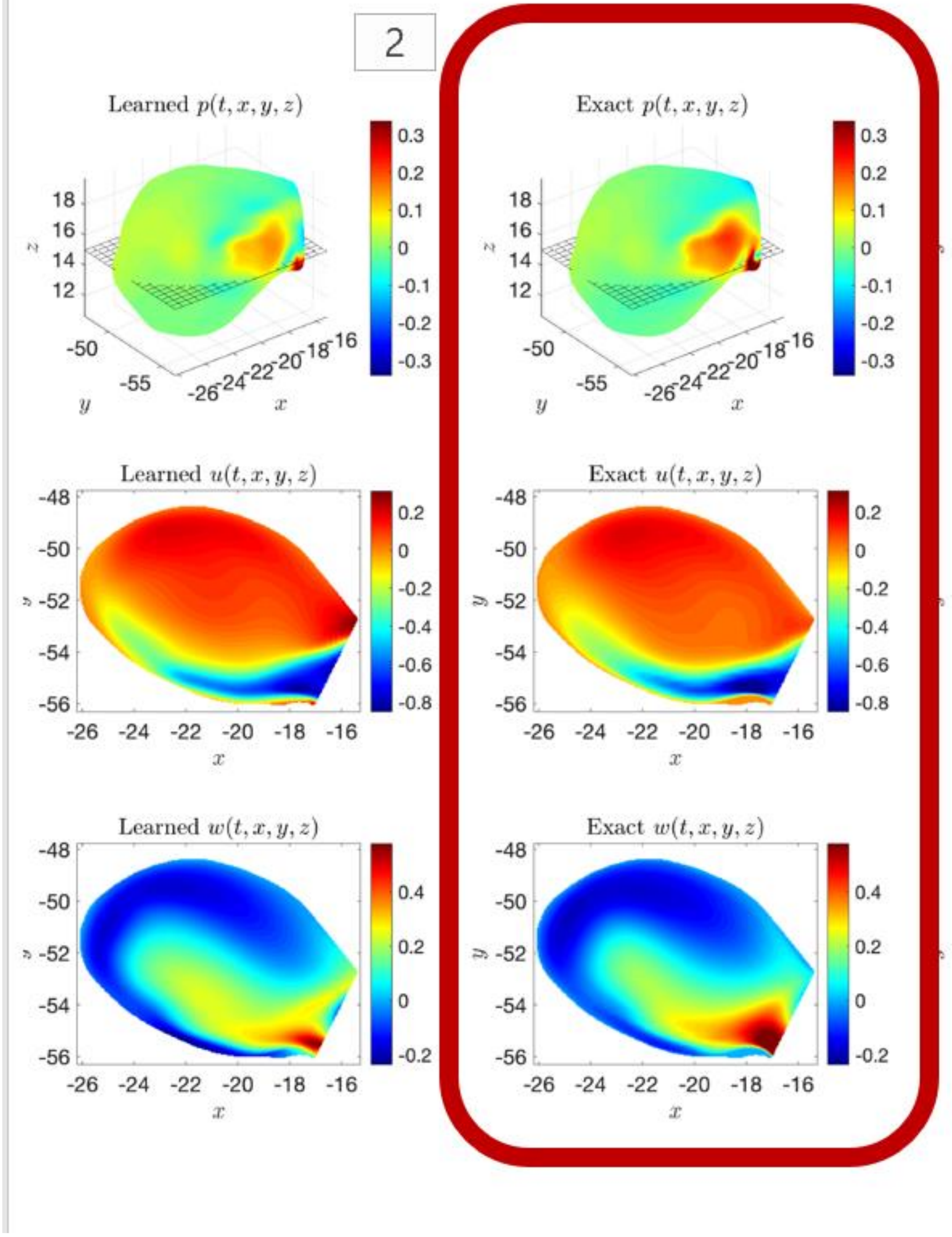


Cut along Y-Plane

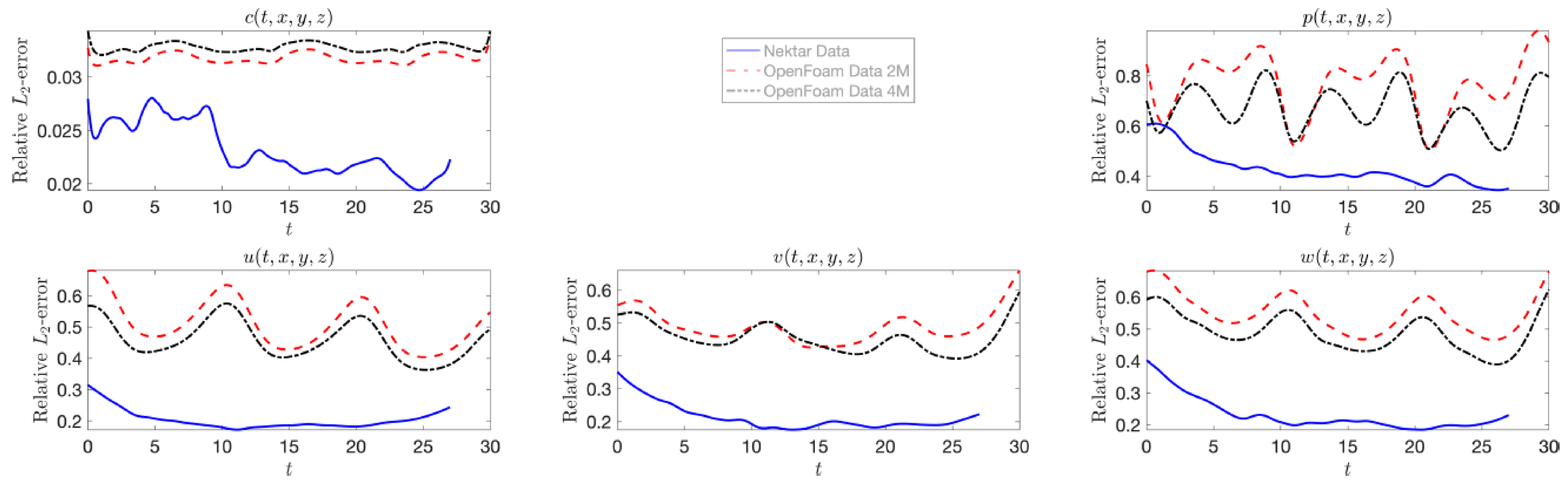


Cut along X-Plane

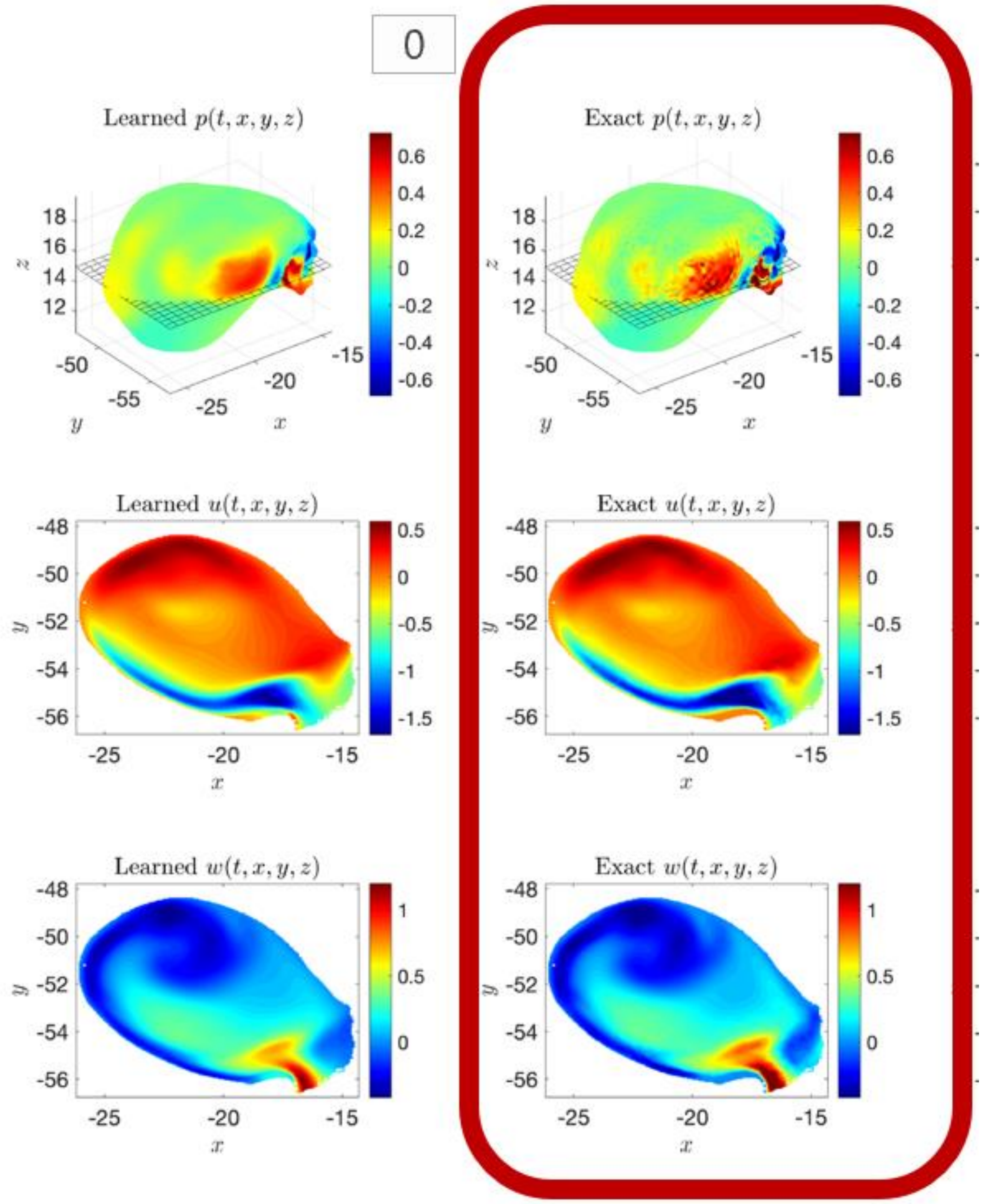
ICA – COMPARISON BETWEEN TWO CFD SOLVERS



OpenFOAM v/s Neural Networks



- Nektar++ is a higher fidelity solver (implicit, h- & p- method based finite element CFD code) and provides higher quality results with less diffusion



Nektar++ v/s Neural Netwo

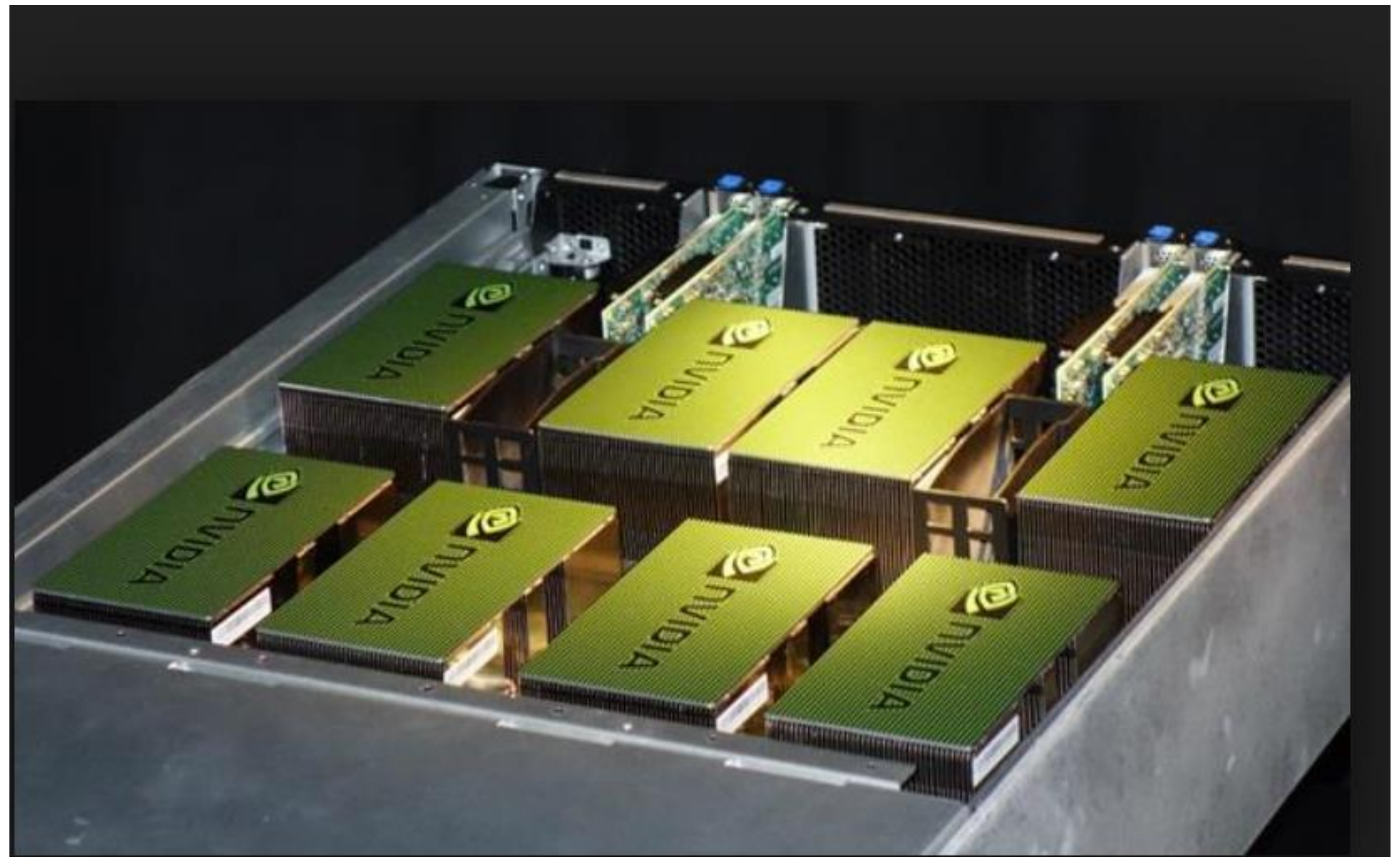
HEAT SINK

Heat Sink -

* Temperatures to not exceed the design criteria

Objectives -

* Similar accuracy as the Solver
* Geometry representation with Point Clouds
* Multiple simultaneous parametrized & unparametrized geometries



Physics involved – CFD & Heat Transfer

*Ansys IcePack used for Simulation (** we kindly acknowledge Ansys's support **)*

HEAT SINK – CONJUGATE HEAT TRANSFER

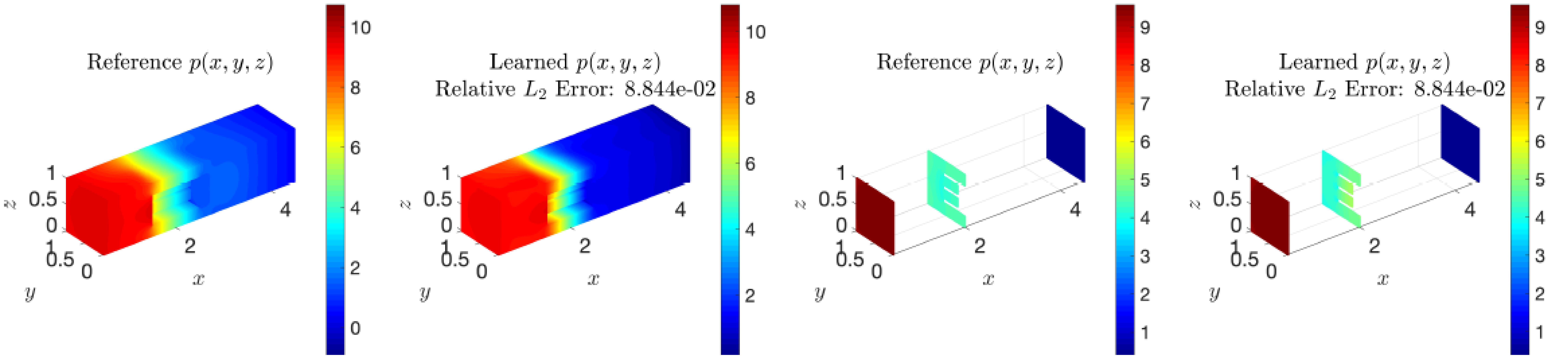
$$MSE = \frac{1}{N}\sum_{i=1}^{N} |d(x_i, y_i) - d_i|^2.$$

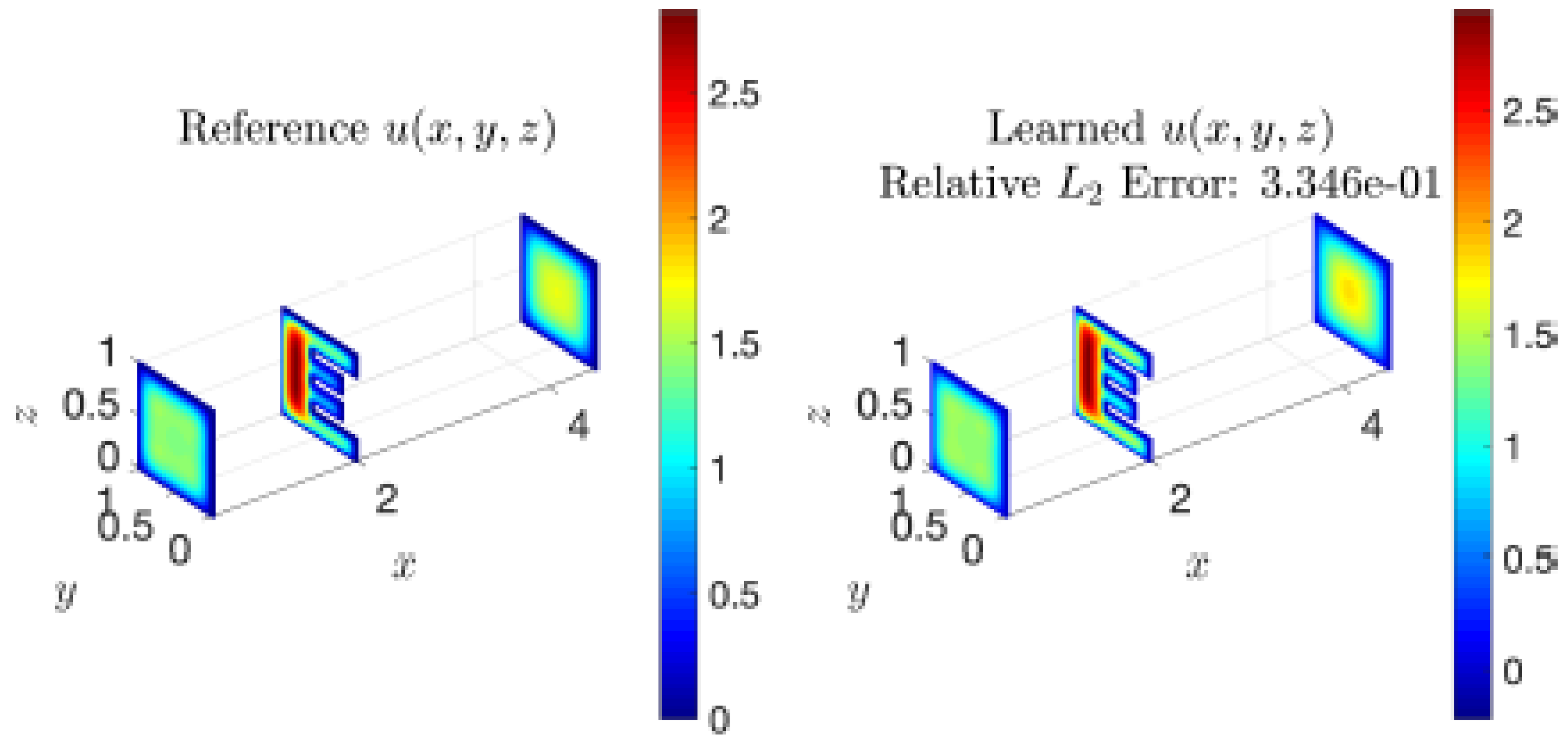
Mean Square Error

$$
\begin{aligned}
e_1 &:= uu_x + vu_y + p_x - (\nu + \nu^t)(u_{xx} + u_{yy}) - 2(\nu_x^t s^{xx} + \nu_y^t s^{xy}),\\
e_2 &:= uv_x + vv_y + p_y - (\nu + \nu^t)(v_{xx} + v_{yy}) - 2(\nu_x^t s^{xy} + \nu_y^t s^{yy}),\\
e_3 &:= u_x + v_y,\\
e^f &:= u\theta_x^f + v\theta_y^f - (\kappa^f/c_p^f + \kappa^t/c_p^f)(\theta_{xx}^f + \theta_{yy}^f) - (1/c_p^f)(\kappa_x^t\theta_x^f + \kappa_y^t\theta_y^f),\\
e^s &:= -\alpha^s(\theta_{xx}^s + \theta_{yy}^s).
\end{aligned}
$$

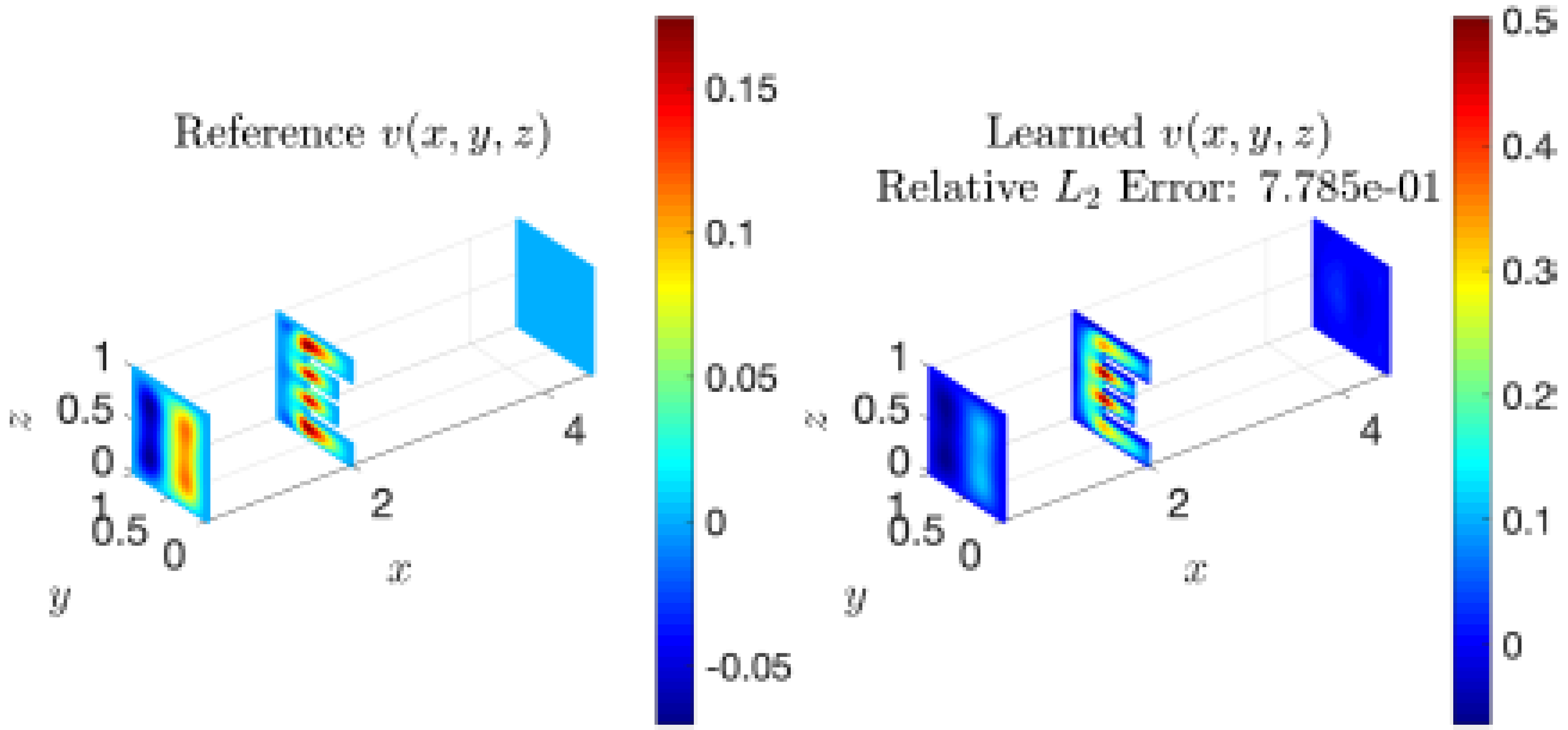
Loss

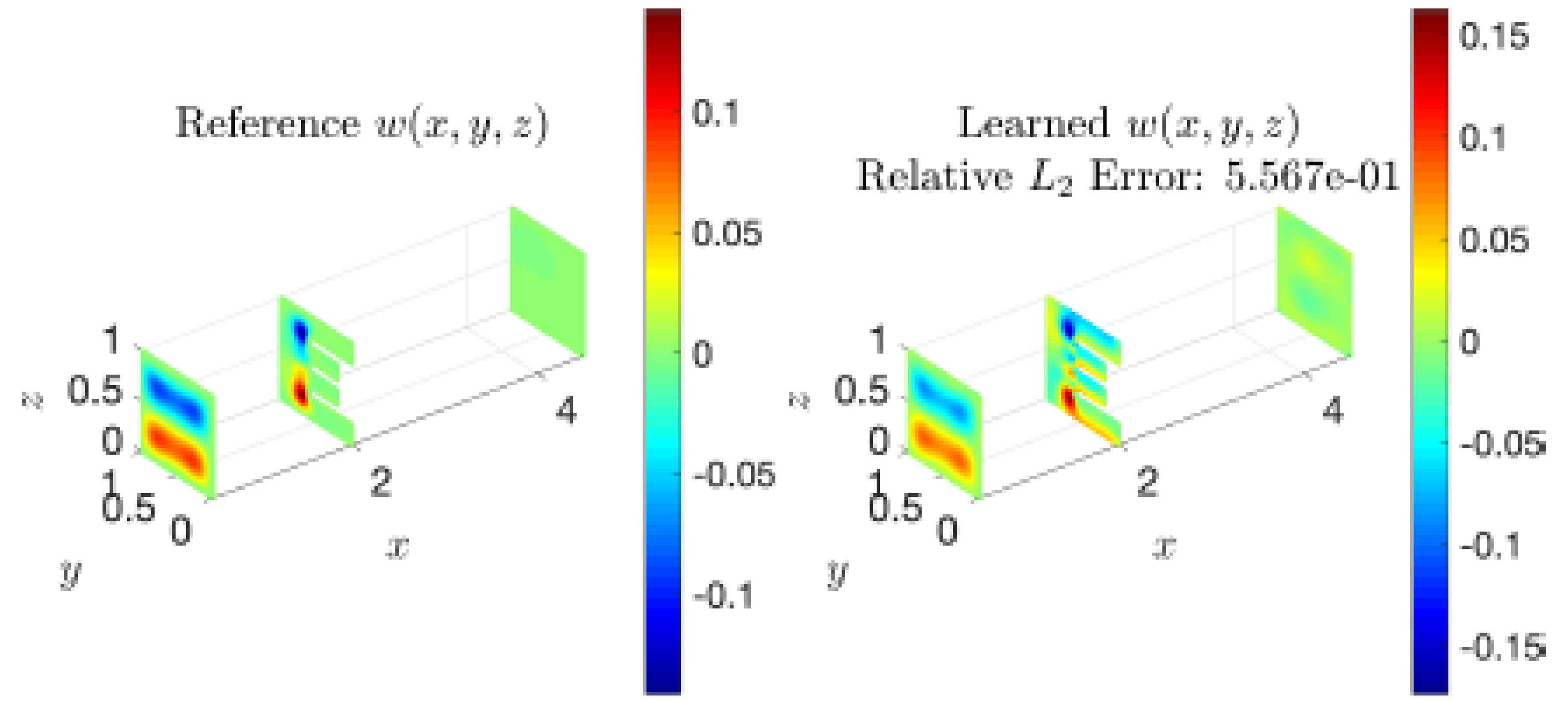
HEAT SINK – CONJUGATE HEAT TRANSFER





Turbulence modeled

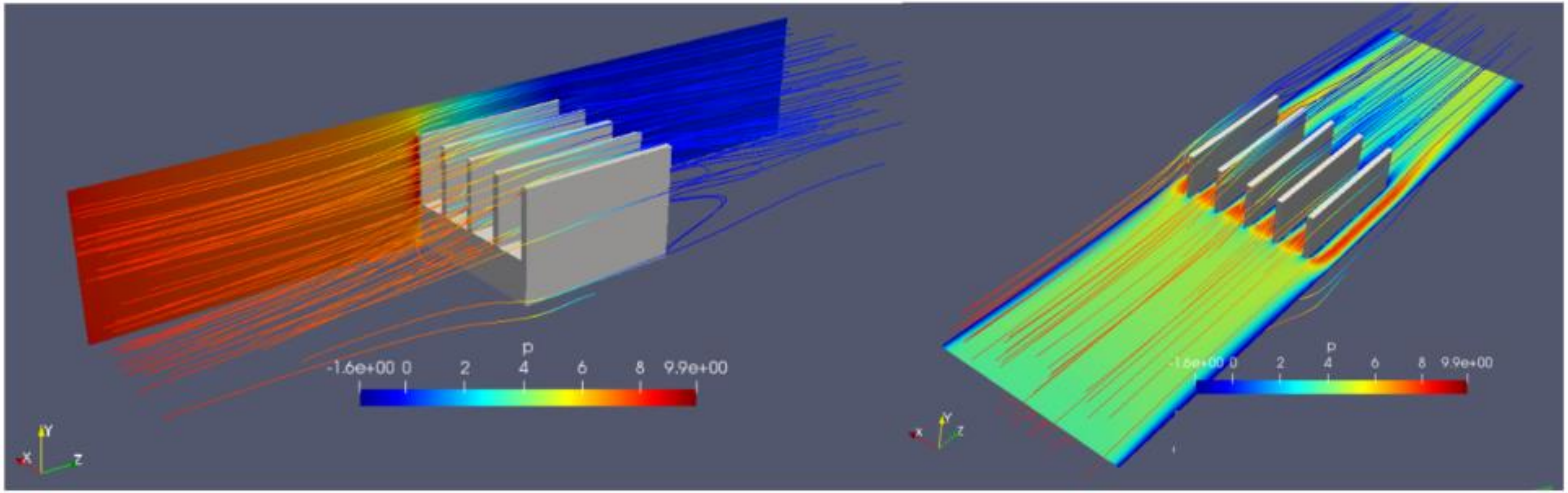




VISUALIZATION

Trained Model Generates Interactive Design Feedback

A 5-Fin Heat Sink solved using AI Workflow



FPGA HEAT SINK

Interactive Design Space Exploration with AI

- Interactive design space exploration is enabled using AI based on Physics informed Neural Networks,
- Multi-Physics (involving CFD & Heat Transfer) heat sink problem solved using end-to-end AI approach
- No training dataset required, only parameterized geometry and boundary conditions

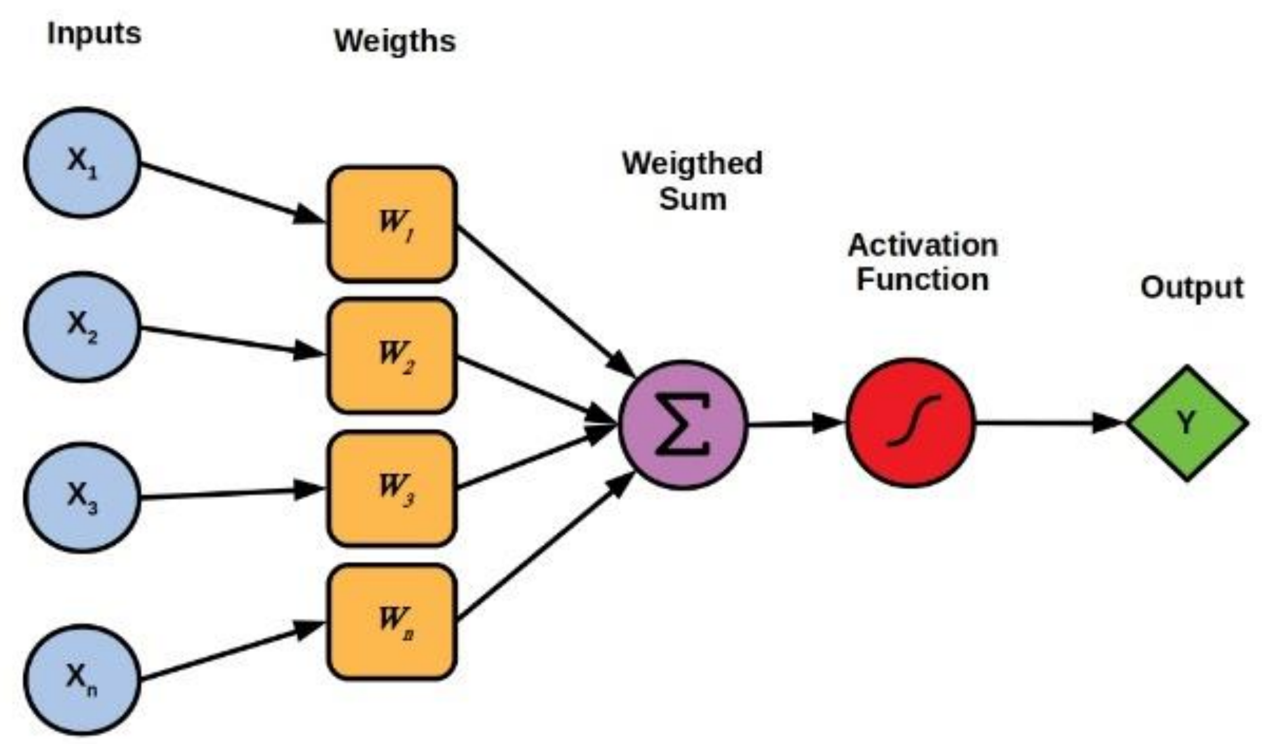
	SimNet Simulation	**Ansys Icepack Simulation**
Total compute time for 2500 cases (design evaluation)	~2 hours (3 secs for each evaluation on a Volta GPU)	>100 days (60 mins on 12 Intel Xeon Gold 6128 CPU cores @ 3.40GHz)
Memory (each case)	216 MB	64 GB
Results file size (each case)	~ 0.5 GB	< 2 GB

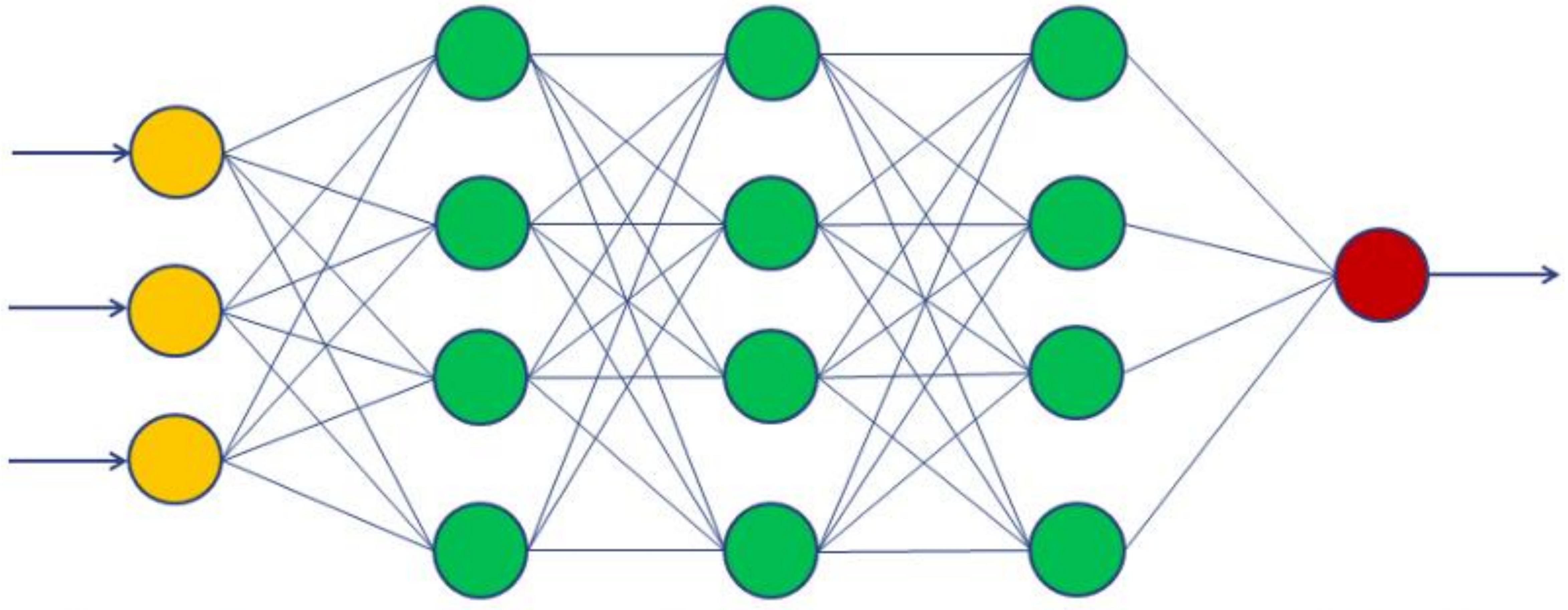
Results - The difference in max. temperature at the heat source between SimNet and Ansys Icepack is similar to the difference between solvers

MODULUS: Promise of PINNs

A PROBLEM WITH NNS AND THE PROMISE OF PINNS

- Neural Networks are functions that can be modified to represent almost any other function
  - Target function: $f(x)$
  - NN to approximate it: $u(x; W) \cong f(x)$
  - **Training:** find weights W that minimize mismatch at selected data points
- Given enough data, Neural Networks can approximate almost any function to any degree of accuracy





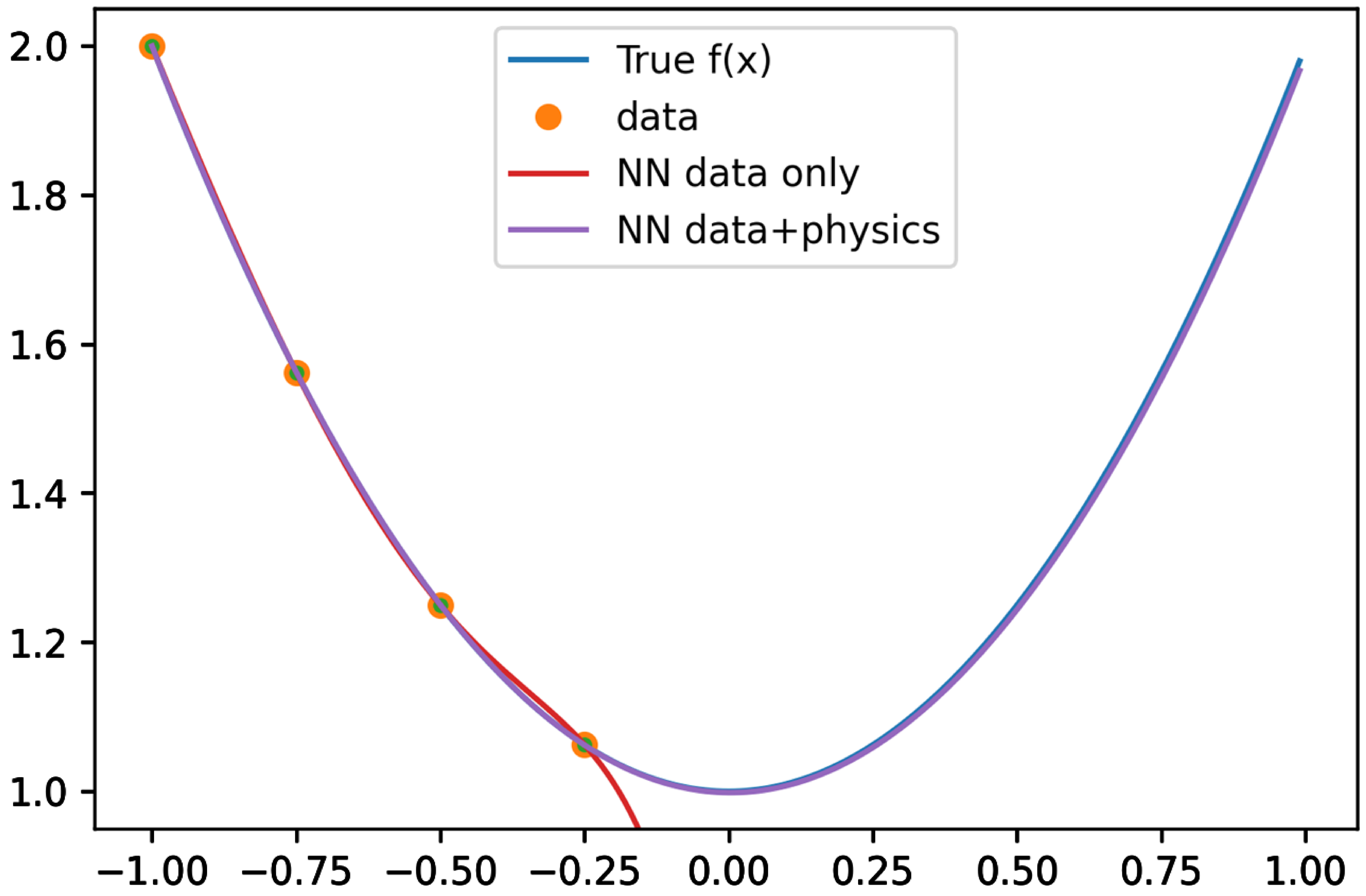
A PROBLEM WITH NNS AND THE PROMISE OF PINNS

Data Only vs PINN: Solving The Data Problem

- Neural Networks are functions that can be modified to represent almost any other function
  - Target function: $\mathrm{f(x)}$
  - NN to approximate it: $\mathrm{u(x;W)} \cong \mathrm{f(x)}$
  - **Training:** find weights W that minimize mismatch at selected data points
- Given enough data, Neural Networks can approximate almost any function to any degree of accuracy
- But… collecting field data may not always be possible
- If we understand the physical laws behind the data, then we can generate enough

$$f(x) = x^2 + 1$$

$$\frac{d^3}{dx^3}u(x) = 0$$



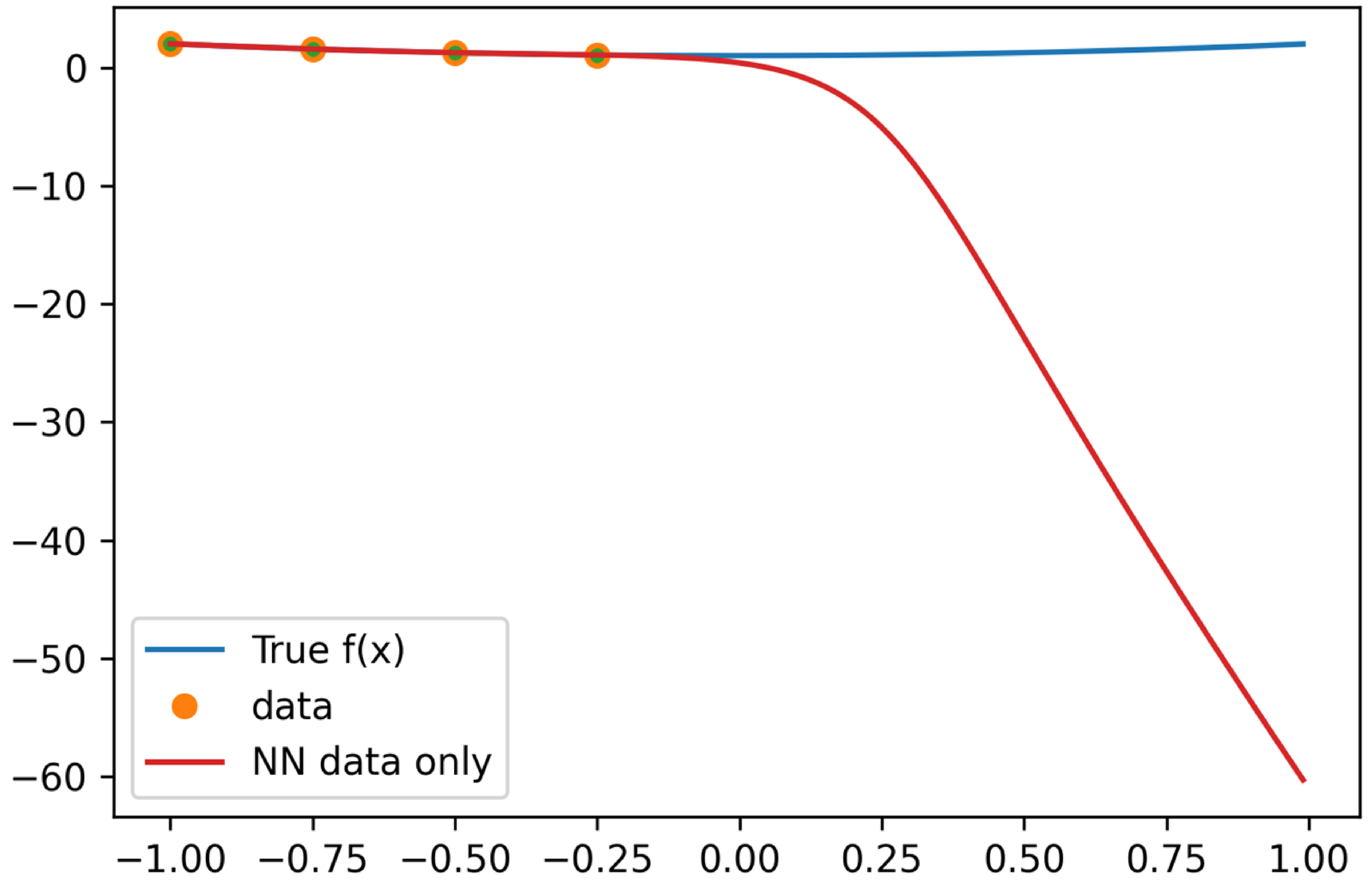
A PROBLEM WITH NNS AND THE PROMISE OF PINNS

Data Only vs PINN: Loss Function

Field Data Only

$$L_{data} \sum_{x_i \in Field\ Data} (u(x_i) - f(x_i))^2$$

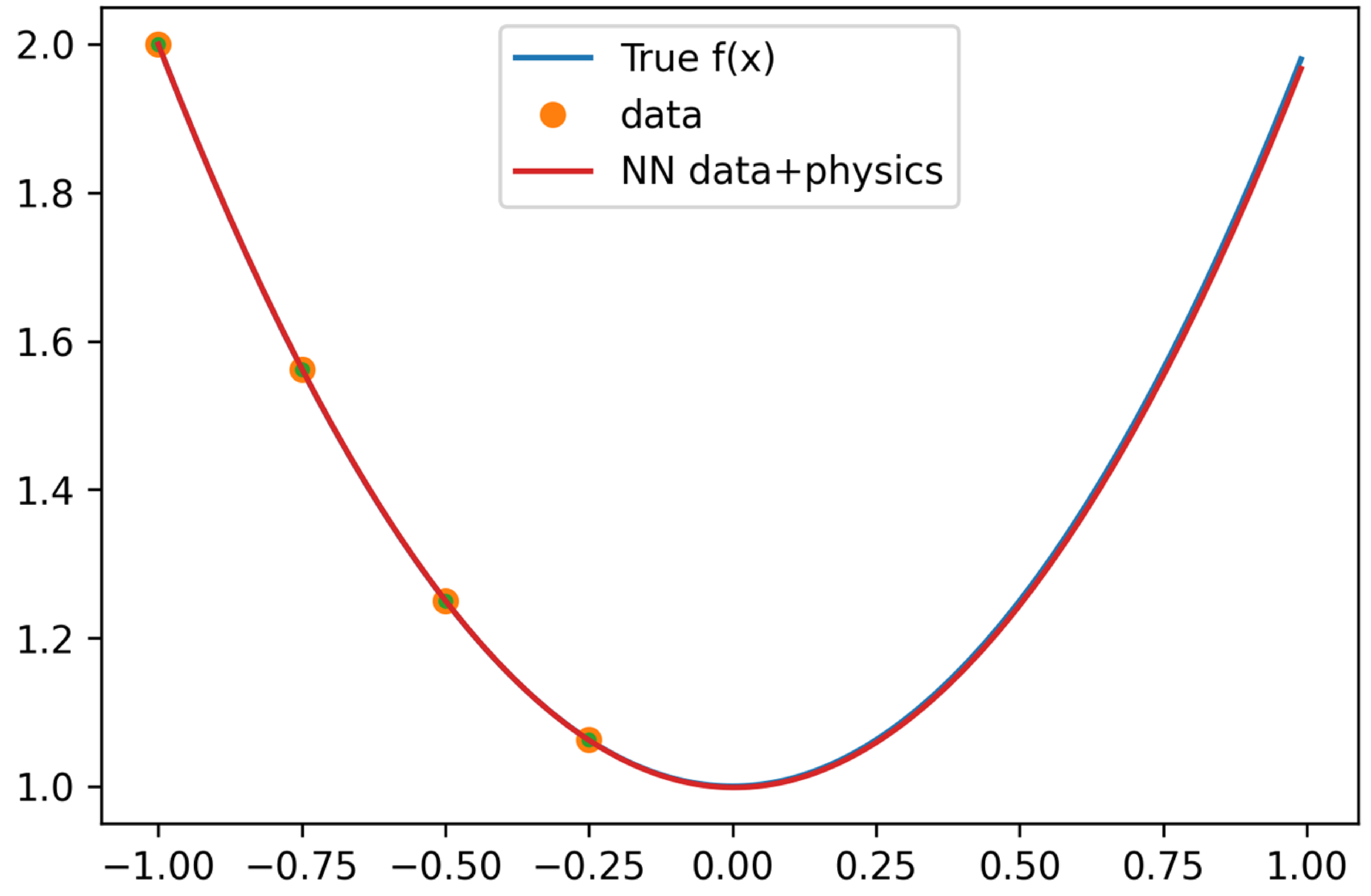
$$L_{total} = L_{data}$$



Field Data + Physics $\frac{d^3}{dx^3} u(x) = 0$

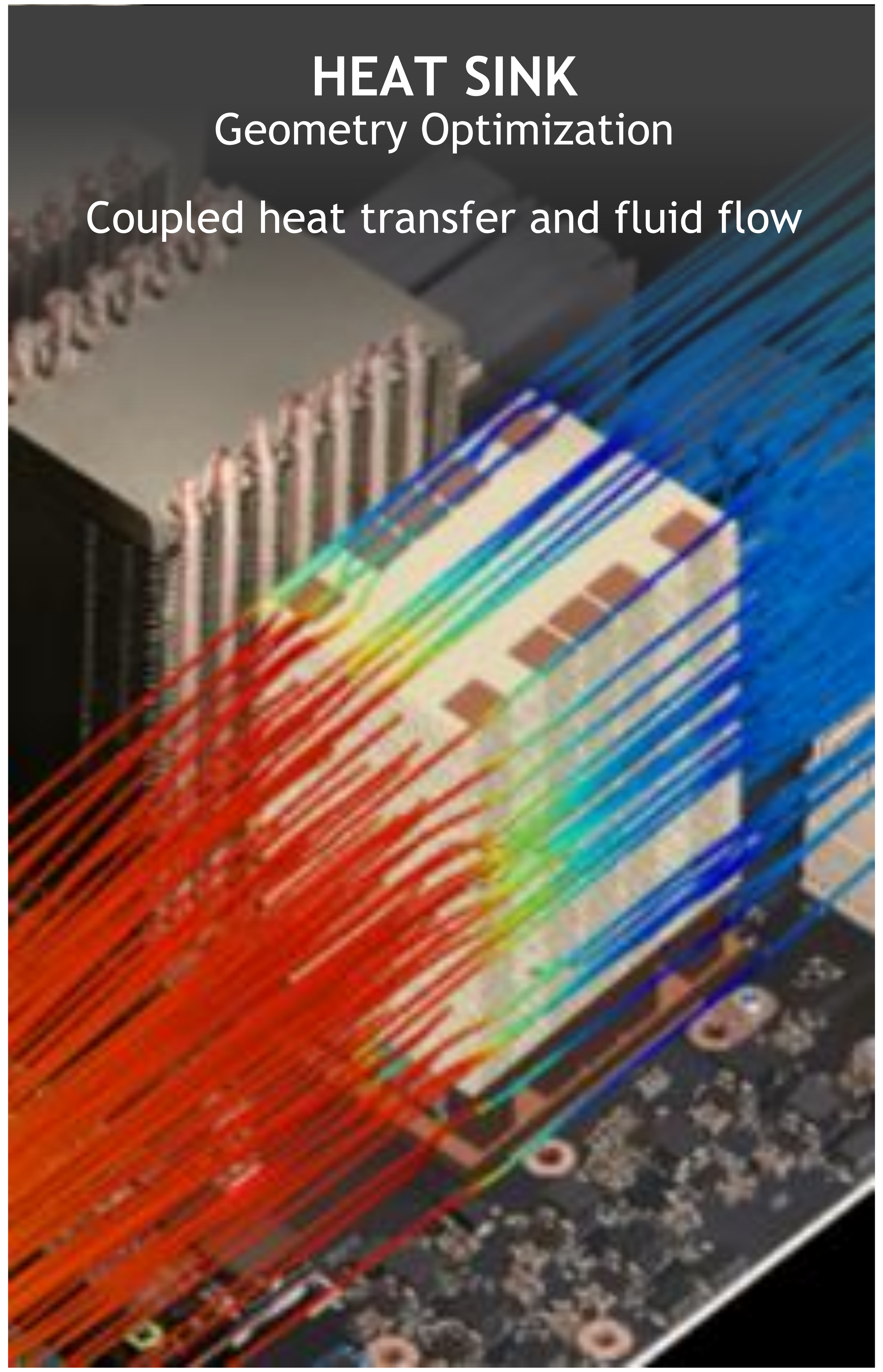
$$L_{physics} = \underbrace{\sum_{x_j \in Domain\ samples}}_{\text{Point Cloud}} \left(\frac{d^3}{dx^3} u(x_j) \right)^2$$

$$L_{total} = L_{data} + L_{physics}$$



A PROBLEM WITH NNS AND THE PROMISE OF PINNS

Sample Applications of PINNs



HEAT SINK
Geometry Optimization

Coupled heat transfer and fluid flow

SIEMENS ENERGY
Heat Recovery Steam Generation

Computational Fluid Dynamics

Coupled flows/physics

SIEMENS GAMESA
Turbine Placement and Life

Computational Fluid Dynamics

NETL
Power Plant Boiler

Computational Fluid Dynamics

Heat Transfer

Chemical Reactions

A PROBLEM WITH NNS AND THE PROMISE OF PINNS

Ongoing Physics-ML Use Cases + Personas: Energy Only

- **Pavel Dimitrov**
  - Siemens Gamesa (**Akshay Subramaniam**, Modulus)
  - Siemens Energy T&D: Bushing
  - RTE / SystemeX: Michelin Tire …
- Shell (Farah Hariri) CFD for Wind Turbines
- **Shourya Otta**
  - Siemens Energy FMS (Fatigue…)
  - GE Research
    - Stenosis
  - Baker Hughes
    - Turbo machinery
    - Additive manufacturing (**Mohammad Nabian**, Modulus)
  - BMW
    - Design optimization: cabin flow
- **Oliver Hennigh** (Modulus team): NETL (power plant boiler)

- (Mostly) Internal Projects
  - Clement Etienam
    - Reservoir Simulation and Inversion (PINNs)
  - Harpreet Sethi
    - FNOs for seismic processing: wave equation “solver” and inversion
  - Jihyun Yang
    - FNOs for brain imaging: wave equation + inversion
- Partner/Customer Personas
  - Researcher LinkedIn (SGRE: Greg Oxley)
  - Research Manager LinkedIn (SE: Georg, Stefan, Shell: Mohammed)

MODULUS: ANATOMY OF A PROJECT

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

- Modulus is a tool to build (differentiable!) Python functions that satisfy constraints such as
  - Adherence to field data
  - Partial Differential Equations
  - Etc.
- Modulus works by:
  - Writing functions (models) as symbolic expressions which include at least one adaptable function (a NN)
  - Writing objective functions as a combination of these models
  - Describing the geometry where the models should be evaluated
  - Minimizing the objective functions by using the provided data, by sampling the geometry, or both
  - Running the models to obtain the desired effect
- The following (partial) list of problems can be solved with this workflow as a side-effect:
  - Train a Neural Network model from data alone
  - Obtain a (differentiable!) function that satisfies a PDE with no field data
  - Obtain best-fit (differentiable!) function that satisfies a PDE using field data
  - Represent PDE boundary conditions through data loosely or exactly
  - Parameterize the solutions of a PDE
  - Inverse problems—e.g., solve for parameters of a function or PDE
  - Etc.

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

- Modulus is a tool to build (differentiable!) Python functions that satisfy constraints such as
  - Adherence to field data
  - Partial Differential Equations
  - Etc.

- Modulus works by:
  - Writing functions (models) as symbolic expressions which include at least one adaptable function (a NN)
  - Writing objective functions as a combination of these models
  - Describing the geometry where the models should be evaluated
  - Minimizing the objective functions by using the provided data, by sampling the geometry, or both
  - Running the models to obtain the desired effect

1. Function Declarations

2. Domain Geometry

3. Loss / Constraint Declarations

4. Auxiliary Validation / Inference

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

$$L_{data} = \sum_{x_i \in Field\ Data} (u(x_i) - f(x_i))^2$$

$$L_{physics} = \sum_{x_j \in Domain\ samples} \left(\frac{d^3}{dx^3} u(x_j) \right)^2$$

$$L_{total} = L_{data} + L_{physics}$$

1. Function Declarations

NN: $u(x)$ Constraint: $\frac{d^3}{dx^3} u(x)$

2. Domain Geometry

Point Cloud Generator over [-1,1]

3. Loss / Constraint Declarations

L_{data}

$L_{physics}$

$L_{total} = L_{data} + L_{physics}$

4. Auxiliary Validation / Inference

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

Step 1. Problem Definition 1/2: Function Declarations

- Declare all Neural Networks
  - Specify input names and output names
    - $u(x)$, $Q_{FC}(x)$, $u(x, bc_{left}, bc_{right})$
  - The NN architecture and parameters (config file)
- Declare functions/equations using the NNs
  - Auxiliary functions: $g(x) = z(x)u(x) + v(x)$
  - Constraint equations:
    - $\frac{d^3}{dx^3}u(x) = 0$ with name "eq"
    - $D\frac{d^2}{dx^2}u(x) - Q = 0$ with name "`diffusion_u`"
    - $D\frac{d^2}{dx^2}g(x) - Q_{FC}(x) = 0$ with name "`diffusion_g`"
- Any declared function can be differentiated using SymPy and Pytorch
- Any declared function can be evaluated provided all inputs are defined (e.g., in an inference stage)

1. Function Declarations

```
# NN declarations
net = instantiate_arch(
        input_keys=[Key("x")],
        output_keys=[Key("u")],
        cfg=cfg.arch.fully_connected,
    )

# Symbolic Function Declarations
x = Symbol('x')

# writing directly
eq = Function("u")(x).diff(x).diff(x).diff(x)

# using PDE library
diff = Diffusion(T="v", D=1.0, Q=-1, dim=1, time=False)

# Aggregate all function declarations in nodes list (required)
# used below in Constraint Declarations
nodes = diff.make_nodes()
nodes += [net.make_node(name=f"diff_net0", jit=cfg.jit) ]
```

$\frac{d^3}{dx^3}u(x) = 0$

2. Domain Geometry

3. Loss / Constraint Declarations

4. Auxiliary Validation / Inference

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

Step 2. Domain Definition: Geometry

- Modulus provides Constructive Solid Geometry tools to describe the geometry by hand
- Modulus can import STL files for complex 3D geometries (e.g., aneurysm example)
- The geometry objects can sample both interior and boundaries (1-D less than interior) to generate the physics-informed point cloud for training or inference

1. Function Declarations

2. Domain Geometry

```
from modulus.geometry.csg.csg_2d import Rectangle
from modulus.geometry.csg.csg_1d import Line1D
from modulus.geometry.csg.csg_3d import Box

# STL geometry
from modulus.geometry.tessellation.tessellation import Tessellation

# read stl files to make geometry
point_path = to_absolute_path("./stl_files")
inlet_mesh = Tessellation.from_stl(
        point_path + "/aneurysm_inlet.stl", airtight=False
)
outlet_mesh = Tessellation.from_stl(
        point_path + "/aneurysm_outlet.stl", airtight=False
)
```

3. Loss / Constraint Declarations

4. Auxiliary Validation / Inference

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

Step 3. Build the Objective Function to Minimize

- The final objective function is created by adding **constraints** to the problem domain; there are many types
  - PointwiseBoundaryConstraint
  - PointwiseInteriorConstraint
  - PointwiseConstraint.from_numpy – field data
  - IntegralConstraint
  - Etc.
- Each pointwise constraint requires:
  - The function declarations from Step 1
  - The geometry object to generate the point cloud
  - The name of the equation from Step 1 and its required value(s) (e.g., `diffusion_u`)
  - Optionally, the type of pointwise aggregation (L2 norm by default, but Lp for any p available)
- Modulus sums all loss functions by default, but that can be modified

1. Function Declarations

2. Domain Geometry

3. Loss / Constraint Declarations

```
# make domain
domain = Domain()
# define data constraints -- at least one type needed
a, b = 1, 2
tt = np.array([-1,-1, 1, 1])
yy = np.array([a, a, b, b])

# supervised = PointwiseConstraint.from_numpy(
    nodes=nodes,
    invar={"x": tt.reshape(-1,1)}, outvar={"u": yy.reshape(-1,1)},
    batch_size=4
)
domain.add_constraint(supervised, "supervised")

# interior (Physics) constraint
interior = PointwiseInteriorConstraint(
    nodes=nodes, geometry=line,
    outvar={"diffusion_u": 0},
    batch_size=cfg.batch_size.interior,
    bounds={x: (-1.0,1.0)},
)
domain.add_constraint(interior, "interior")
```

4. Auxiliary Validation / Inference

MODULUS: ANATOMY OF A PROJECT

What is Modulus?

Step 4. Do Something With The Model(s)

- Validate model performance by comparing model output to expected behavior.
  - E.g., useful to compare PINN solution to an existing numerical solution stored in a data file
- Inference: generate model output given a set of input values; i.e., evaluate $u(x)$ given values for x
  - PointwiseInferencer takes
    - a dict of inputs
    - a dict of desired outputs (Modulus expression)
    - the function declarations (nodes)
  - The function declarations define a compute graph
  - The compute graph allows differentiation of any function in the graph w.r.t any input of said function
    - Example: if a graph defines $g(x)$ then putting outputs=['g', 'g__x'] will compute both $g(x)$ and its first derivative

1. Function Declarations

2. Domain Geometry

3. Loss / Constraint Declarations

4. Auxiliary Validation / Inference

```
xx = np.arange(-1,1, 1/100)
in_vars = {"x": xx.reshape(-1,1)}

inferencer = PointwiseInferencer(
        in_vars,
        ['g', 'g__x'],
        nodes,
        batch_size=256,
               plotter=Plotter(), # Plot results in Tensorboard
    )
    domain.add_inferencer(inferencer)
```

OMNIVERSE – TOOL FOR BUILDING METAVERSE APPLICATIONS

NVIDIA