A composable machine learning approach for steady-state simulations on high-resolution grids.

Rishikesh Ranade[1], Chris Hill[2], Lalit Ghule[1], Jay Pathak[1]

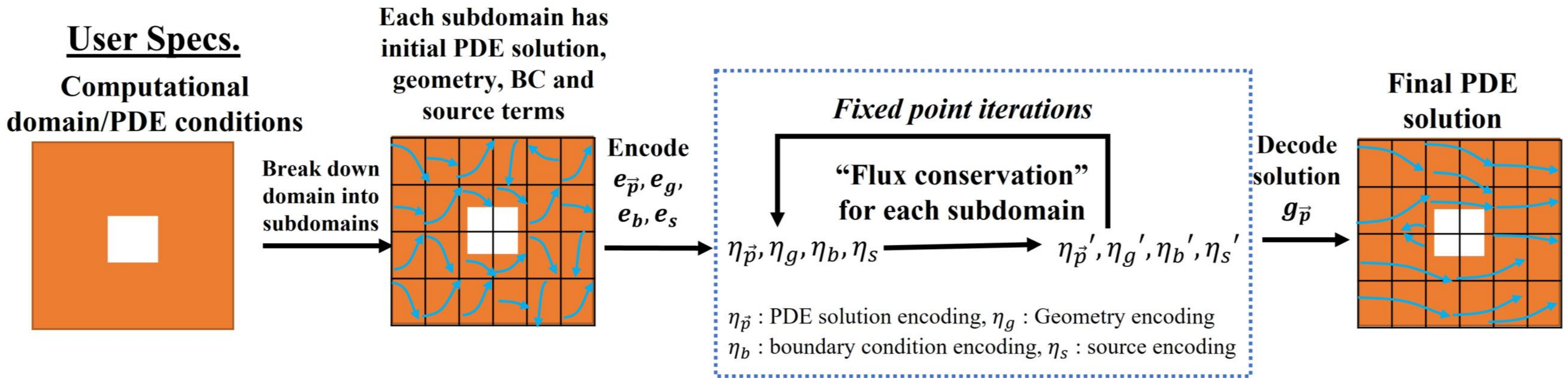
[1]Office of CTO, Ansys Inc.

[2]Fluids Business Unit, Ansys Inc.

Motivation

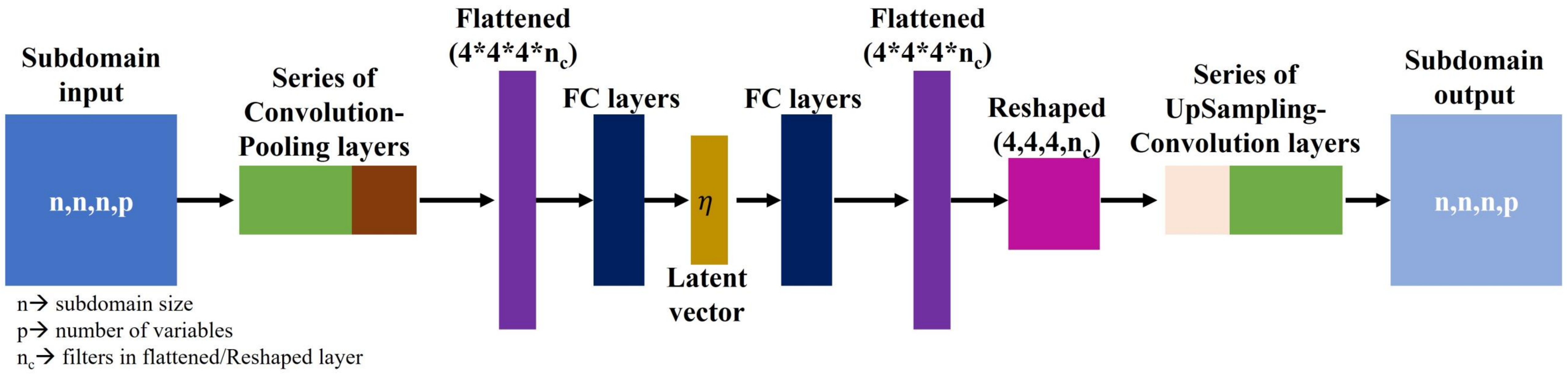
- Geometries and physics have a lot of identifiable patterns.
- Traditional FVM/FEM/FDM based PDE solvers work at the level of computational elements. They solve PDEs by iteratively conserving fluxes across neighboring elements. This can be computationally expensive.
- Existing supervised/unsupervised ML methods are faster but not accurate or generalizable.
  - Black-boxed static inferencing
  - Inefficient and inaccurate for high-resolution grids required to capture solution features
- Most ML approaches don’t use any ideas from traditional PDE solvers.
- Can we learn from PDE solver theory and build a low-dimensional ML approach?

Composable Machine Learning Simulator (CoMLSim)


User Specs.
Computational domain/PDE conditions
Break down domain into subdomains
Each subdomain has initial PDE solution, geometry, BC and source terms
Encode $e_{\vec{p}}, e_g, e_b, e_s$
Fixed point iterations
"Flux conservation" for each subdomain
$\eta_{\vec{p}}, \eta_g, \eta_b, \eta_s$
$\eta_{\vec{p}}', \eta_g', \eta_b', \eta_s'$
$\eta_{\vec{p}}$: PDE solution encoding, η_g : Geometry encoding
η_b : boundary condition encoding, η_s : source encoding
Decode solution $g_{\vec{p}}$
Final PDE solution

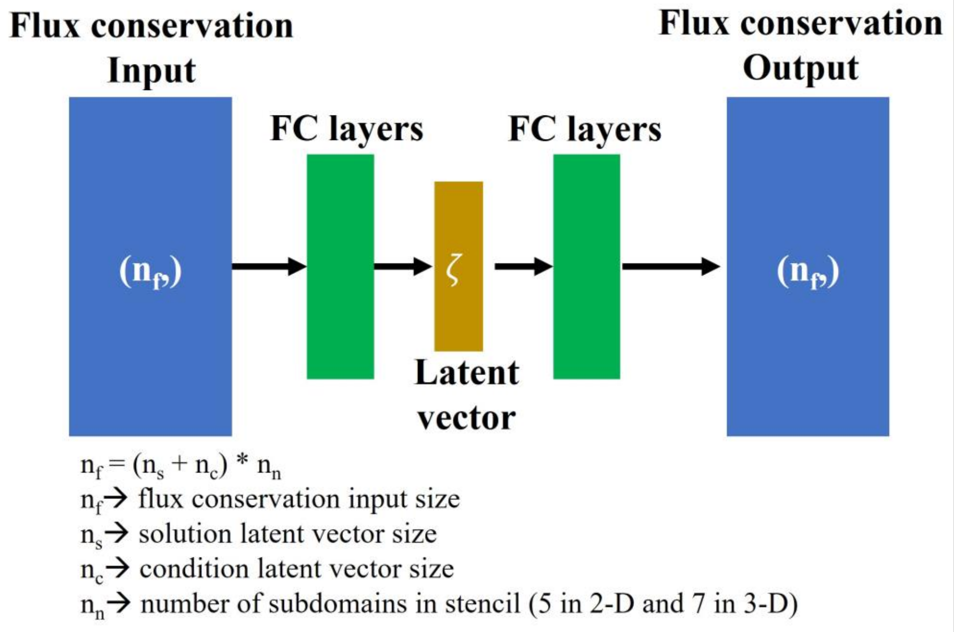
- Given a computational domain with user specified PDE conditions such as geometry, source terms etc.
- CoMLSim discretizes the domain into subdomains (collection of elements) and represents initial PDE solutions and conditions with corresponding latent vectors, $\eta_{\vec{p}}, \eta_g, \eta_s$ using pretrained autoencoders.
- Concatenated latent vectors are evaluated iteratively with a flux conservation autoencoder. In each iteration the solution latent vectors are updated while the condition latent vectors are kept constant.
- The iteration converges when $L_2\left(\eta'_{\vec{p}} - \eta_{\vec{p}}\right) < 1e^{-8}$

Solution/Condition autoencoder

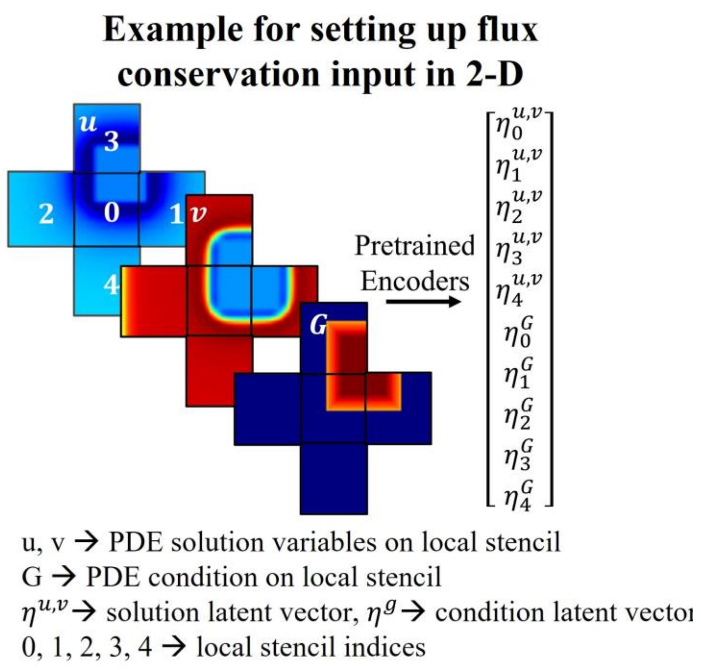


- PDE solutions and input conditions on local subdomains can be represented in a lower-dimensional vector η.
- Other encoder/decoder networks can be used based on the input data representation: graph encoders, PCA etc.
- Higher compression reduces time-per-iteration and iteration count, as well as memory usage.
- Field autoencoders for solutions are trained with samples generated from FEM/FDM/FVM solvers for a given PDE.

Flux conservation autoencoder



$n_f = (n_s + n_c) * n_n$
$n_f \rightarrow$ flux conservation input size
$n_s \rightarrow$ solution latent vector size
$n_c \rightarrow$ condition latent vector size
$n_n \rightarrow$ number of subdomains in stencil (5 in 2-D and 7 in 3-D)



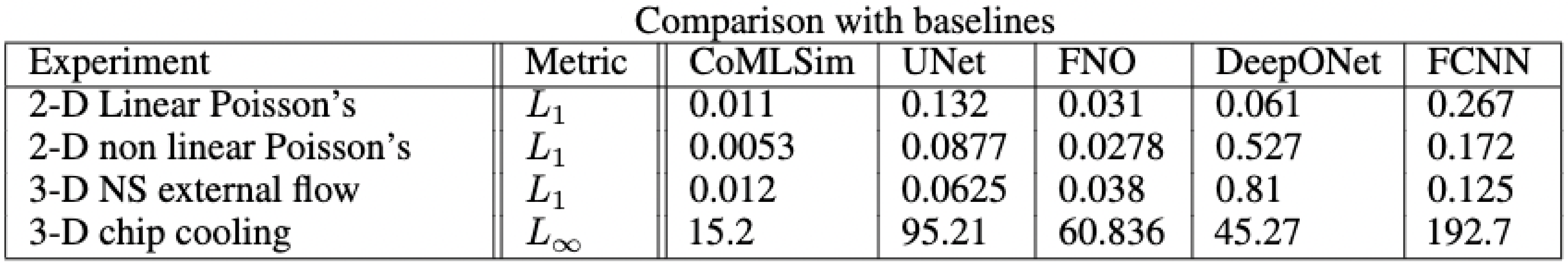
u, v $\rightarrow$ PDE solution variables on local stencil
G $\rightarrow$ PDE condition on local stencil
$\eta^{u,v} \rightarrow$ solution latent vector, $\eta^{g} \rightarrow$ condition latent vectoı
0, 1, 2, 3, 4 $\rightarrow$ local stencil indices

- Flux conservation autoencoder learns local consistency relationships between neighborhood subdomains in the latent space. This process is very similar to flux conservation in traditional PDE.
- Solution and condition latent vectors on neighboring subdomains are concatenated together and used as input to the flux conservation autoencoder.
- Since flux conservation autoencoder is trained on encodings of actual locally consistent PDE solutions/conditions it always converges to locally consistent solutions during inference. The uniqueness of solution is determined by the fixed condition latent vectors.

Key takeaways from CoMLSim approach

1) Local learning enables accurate and generalizable learning on highly-resolved grids.

- Higher accuracy over ML baselines ✓
- Extending to bigger physical domains with larger resolutions.
- Better generalization to out-of-distribution conditions such as geometries, source terms etc.



Comparison with baselines

Experiment	Metric	CoMLSim	UNet	FNO	DeepONet	FCNN
2-D Linear Poisson's	L_1	0.011	0.132	0.031	0.061	0.267
2-D non linear Poisson's	L_1	0.0053	0.0877	0.0278	0.527	0.172
3-D NS external flow	L_1	0.012	0.0625	0.038	0.81	0.125
3-D chip cooling	L_∞	15.2	95.21	60.836	45.27	192.7

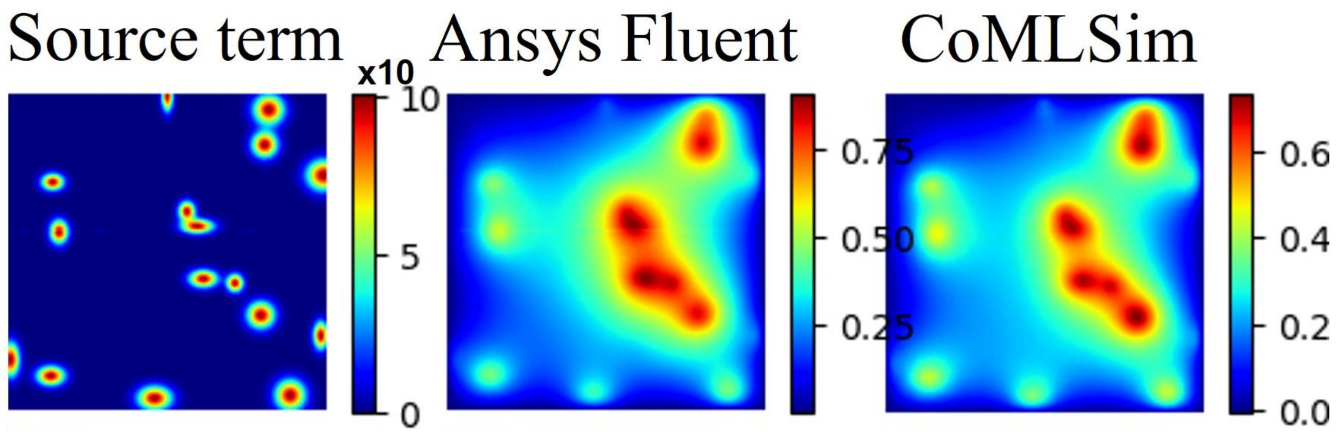
All experiments are carried out on highly-resolved grids with limited training samples. The errors are averaged over a large number of unseen testing samples. CoMLSim outperforms all the ML baselines.

Key takeaways from CoMLSim approach

1) **Local learning** enables accurate and generalizable learning on highly-resolved grids.

- Higher accuracy over ML baselines ✓
- Extending to bigger physical domains with larger resolutions ✓
- Better generalization to out-of-distribution conditions such as geometries, source terms etc.

2D Poisson's equation

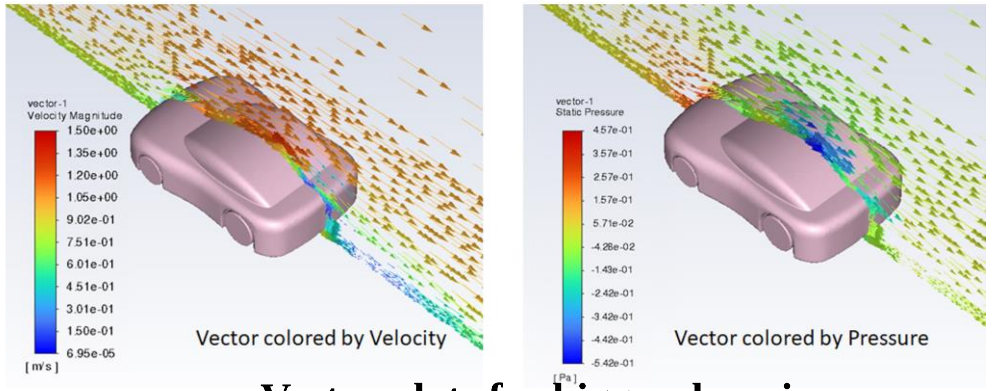


Contour comparisons for bigger domain (resolution 2048x2048)

3D External NS flow

Norm. MAE on velocity = 0.039

Norm. MAE on pressure = 0.106



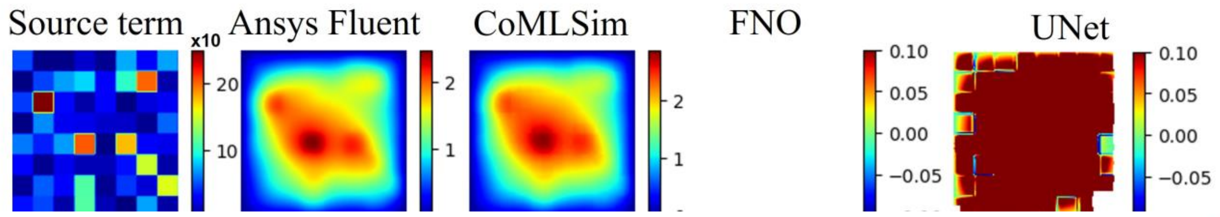
Vector plots for bigger domain (resolution 562x128x128)

Key takeaways from CoMLSim approach

1) **Local learning** enables accurate and generalizable learning on highly-resolved grids.

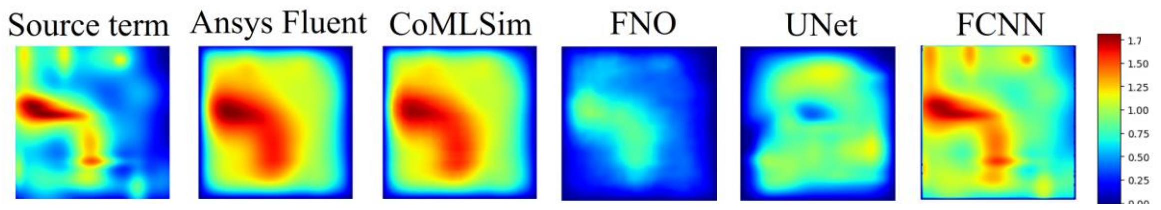
- Higher accuracy over ML baselines ✓
- Extending to bigger physical domains with larger resolutions ✓
- Better generalization to out-of-distribution conditions such as geometries, source terms etc. ✓

2D Poisson's equation

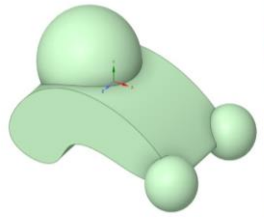


Training source distribution is Gaussian mixture but testing is on discontinuous/tiled distribution

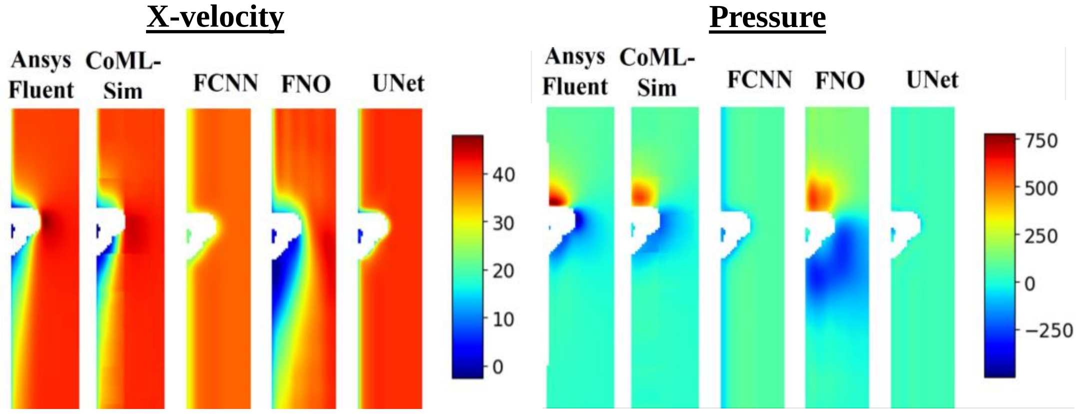
2D Non-linear coupled Poisson's equation



Training source distribution is Gaussian mixture but testing is on out-of-distribution Gaussian mixture



3D Ext. NS Flow



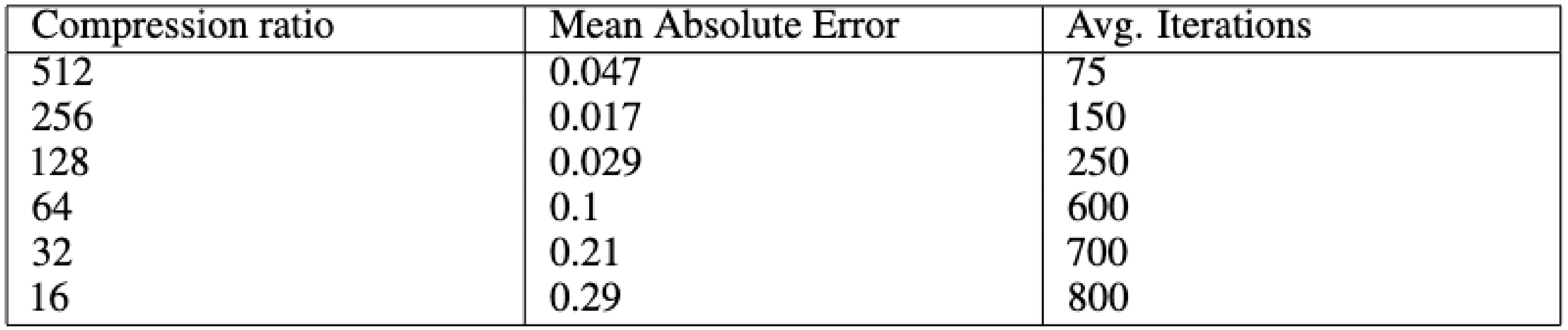
Training on primitive shapes but testing on non-trivial shapes

Key takeaways from CoMLSim approach

2) Latent space computations provide computational speed-ups during inference ✓

- Latent space is a lower-dimensional representation of the physical computational space.

Solution/Condition Encoding size vs accuracy and number of convergence iterations



Compression ratio	Mean Absolute Error	Avg. Iterations
512	0.047	75
256	0.017	150
128	0.029	250
64	0.1	600
32	0.21	700
16	0.29	800

Smaller latent sizes result in faster convergence and have a smaller per iteration cost. As the latent size increases the autoencoders overfit and negatively impacts the accuracy and convergence iterations.

Key takeaways from CoMLSim approach

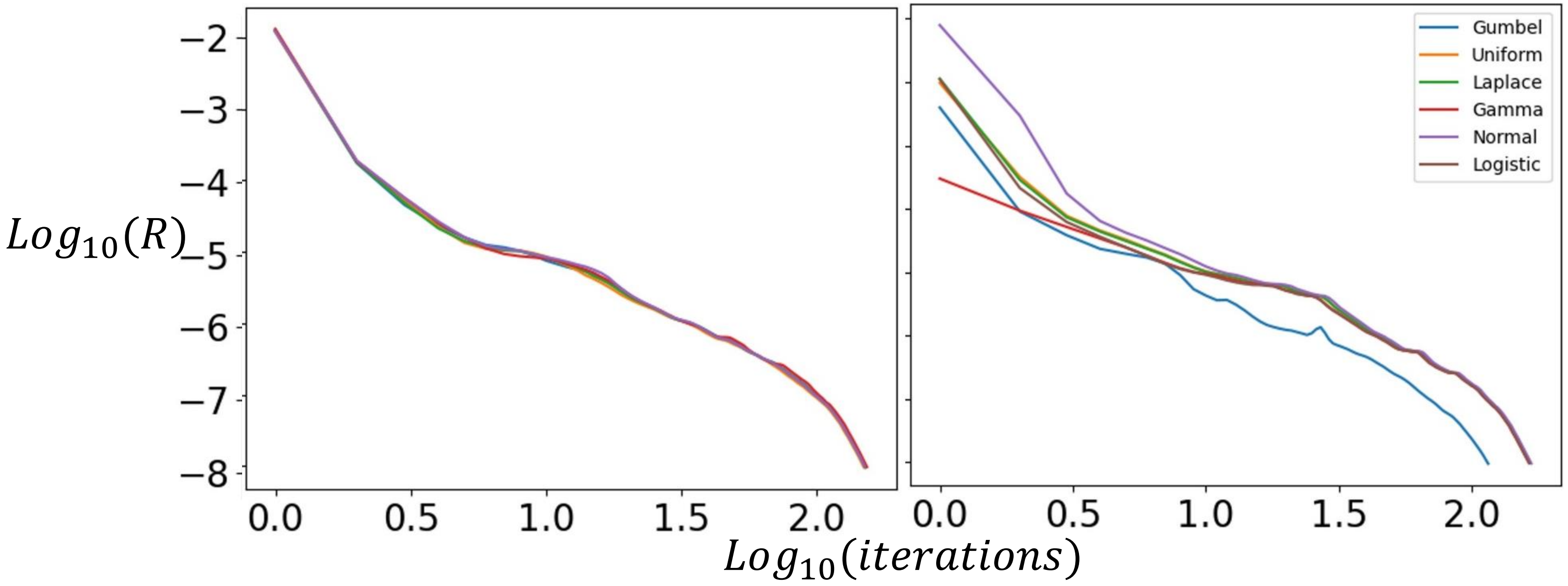
3) Iterative solution algorithm at inference:

- Self-supervised learning algorithm: Just need to learn solution representations through field and flux autoencoders. ✓
- Stable and robust convergence at inference
- Ease of coupling with traditional (Ansys) solvers
- Better explanability: solution evolution during iteration aligns with nature of the PDE solved

Key takeaways from CoMLSim approach

3) Iterative solution algorithm at inference:

- Self-supervised learning algorithm: Just need to learn solution representations through field and flux autoencoders. ✓
- Stable and robust convergence at inference ✓
- Ease of coupling with traditional (Ansys) solvers
- Better explainability: solution evolution during iteration aligns with nature of the PDE solved

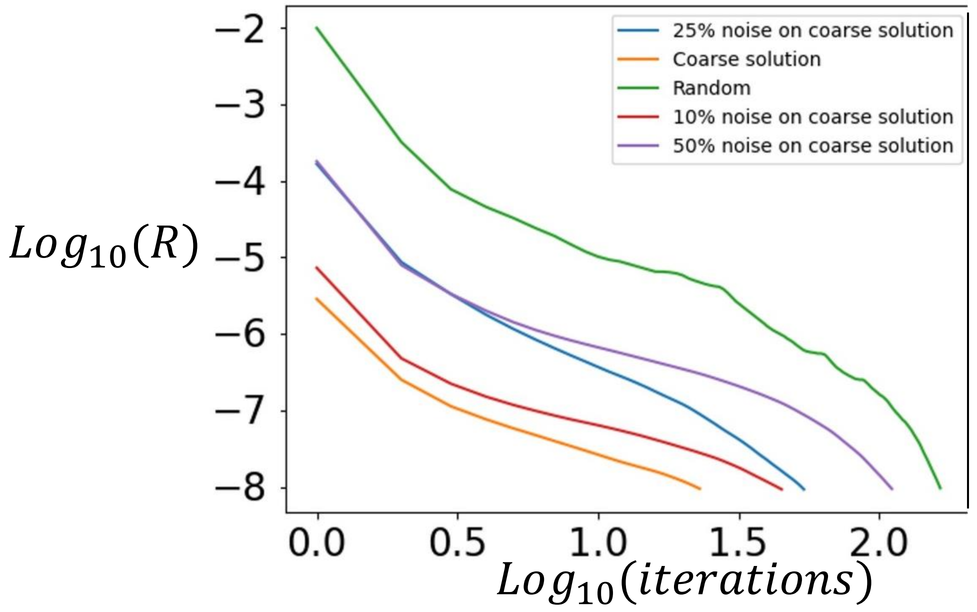


Shows robust convergence of CoMLSim during inference starting from 25 uniform random solutions (left) and 6 solutions from different distributions (right)

Key takeaways from CoMLSim approach

3) Iterative solution algorithm at inference:

- Self-supervised learning algorithm: Just need to learn solution representations through field and flux autoencoders. ✓
- Stable and robust convergence at inference ✓
- Ease of coupling with traditional (Ansys) solvers ✓
- Better explanability: solution evolution during iteration aligns with nature of the PDE solved

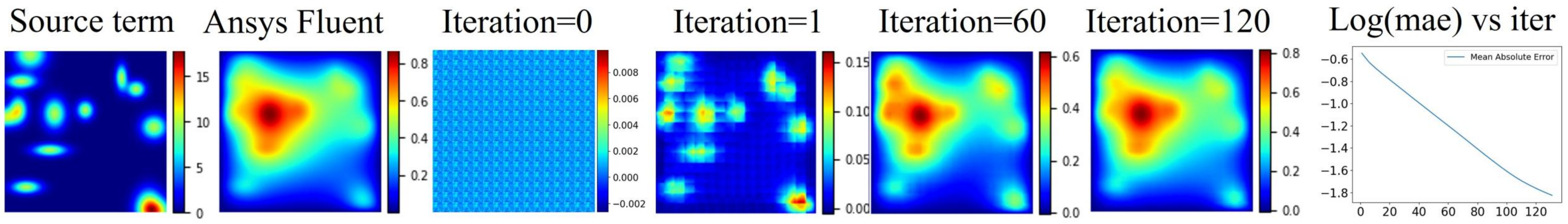


Shows faster convergence of CoMLSim during inference starting from coarse solutions generated by traditional solvers.

Key takeaways from CoMLSim approach

3) Iterative solution algorithm at inference:

- Self-supervised learning algorithm: Just need to learn solution representations through field and flux autoencoders. ✓
- Stable and robust convergence at inference ✓
- Ease of coupling with traditional (Ansys) solvers ✓
- Better explanability: solution evolution during iteration aligns with nature of the PDE solved ✓



Contour plots show the CoMLSim prediction vs Ansys solver ground truth solution at different iterations. The solution evolution is analogous to the PDE (Poisson's equation) representing this problem. Initially, solution magnitude builds around source peaks, which diffuse through the domain as iterations progress.

Key takeaways from CoMLSim approach

4) **Based on key ideas from traditional PDE solvers.**

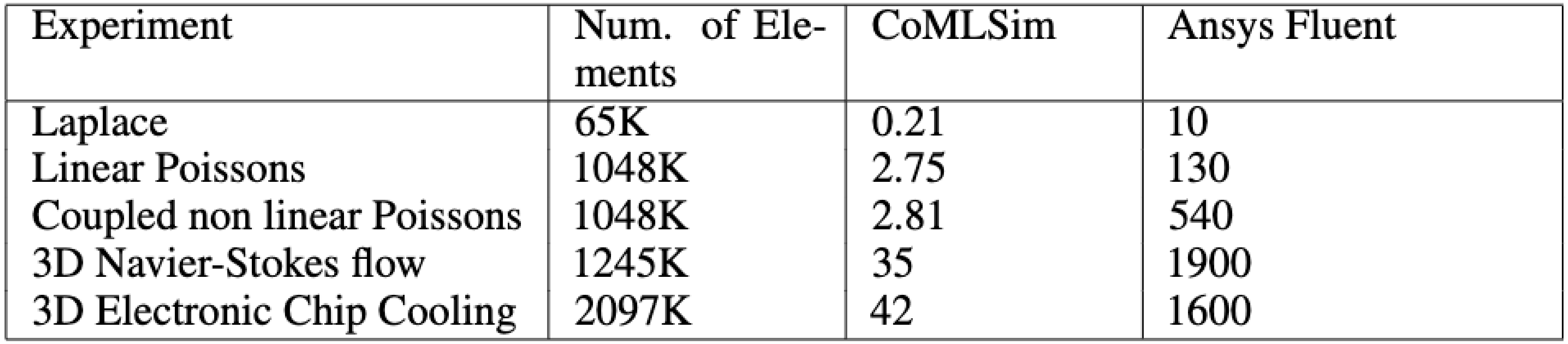
- Discretization,
- linear/non-linear solvers,
- evaluation/convergence metrics etc.

5) Can be **trained accurately with limited training data (~300).**

6) Converges **50-100x faster** than traditional PDE solvers on a **single CPU**.

7) Can be parallelized to multi-CPU and GPU architectures.

Simulation time (in seconds) comparison with Ansys Fluent



Experiment	Num. of Elements	CoMLSim	Ansys Fluent
Laplace	65K	0.21	10
Linear Poissons	1048K	2.75	130
Coupled non linear Poissons	1048K	2.81	540
3D Navier-Stokes flow	1245K	35	1900
3D Electronic Chip Cooling	2097K	42	1600

CoMLSim Summary

- Data efficient learning on high-resolution grids (~300 samples)
  - Most other baselines require a lot more data.
- Scaling to large physical domains
  - Can be parallelized to more CPUs and GPUs
- Better generalization to out-of-distribution settings
  - 3D complex geometric shapes
  - High dimensional source terms
- Scales to any grid resolution
  - Can be trained and evaluated on very fine resolutions
- Iterative inferencing enables coupling with traditional PDE solvers

Future work

- Extension to transient PDEs.
- Handling unstructured/multi-resolution representations within subdomains.
- Further speedup of iterative inference with better algorithms and parallel processing.

Ansys