ML Based CFD Model Reduction for Rapid Computational Screening

Phan Nguyen & Brian Bartoldson
Lawrence Livermore National Laboratory

August 17, 2022

Machine Learning Team:
Jose Cadena (LLNL)
Amar Saini (LLNL)
Yeping Hu (LLNL)
CCSI\textsuperscript{2}: Carbon Capture Simulation for Industry Impact
Machine Learning

Output

Prediction

Structure

Policy

Data in numeric representation

Input

Foundation

Math

Statistics

Optimization

Machine Learning
History of ML in CCSI

- **Unsupervised Learning**
  - Discover hidden structure in unlabeled data
  - Dimension Reduction

- **Supervised Learning**
  - Supervised Learning
  - Surrogates in Opt & UQ
  - Low-dim models with <100 input/output parameters
  - Regression

- **Reinforcement Learning**
  - Learning actions (policy) based on feedback from the environment
How do we use ML for CCSI^2?

CFD is critical for the fundamental understanding, to inform process and system level modeling.

- **Need local information on transport phenomena** to understand driving forces
- **Can be incorporated into design optimization** to optimize the device

Simulation time is a bottleneck that impedes high-level modeling.

Machine learning surrogates, such as **Deep Fluids (DF)** and **MeshGraphNets (MGN)**, can reduce the computational burden of time-consuming simulations.
Fast Surrogates for CFD Simulation Model

Raschring random packing config
Raschring element properties
Liquid flow rate
Gas flow rate
Liquid viscosity
Liquid density
Surface tension
Contact angle
Packed ring surface to volume ratio

CFD sim (high-dim)

Inputs

Machine Learning

Outputs

Interfacial area
Liquid holdup
Pressure drop
Effective mass transfer

Fast Surrogates for CFD Simulation Model

CCSI² Carbon Capture Simulation for Industry Impact
### Computational approaches to screening parameters

<table>
<thead>
<tr>
<th>Metric</th>
<th>Computational Fluid Dynamics (CFD)</th>
<th>Machine Learning (ML) Surrogates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td><em>Slow:</em> 2D model takes 1 hour to simulate</td>
<td><em>Fast:</em> 2D model takes 1 second to simulate</td>
</tr>
<tr>
<td>Effort to construct</td>
<td><em>High:</em> Equations, assumptions, numerical methods, software packages, …</td>
<td><em>Low:</em> Common architectures across problems reduce effort required for a ‘good’ model</td>
</tr>
<tr>
<td>Accuracy</td>
<td><em>Variable:</em> Depends on the model/effort</td>
<td><em>Variable:</em> Depends on accurate CFD training data and surrogate’s ability to bridge from &quot;toy&quot; problems in literature to our large-scale data</td>
</tr>
<tr>
<td>Downstream modeling/problems</td>
<td><em>Ill-suited:</em> Generally, too slow to scale to large, 3D simulations</td>
<td><em>Well-suited:</em> Scaling made feasible by speed of reduced models</td>
</tr>
</tbody>
</table>
Growth in Machine Learning and the Physical Sciences

Growth in Neural Information Processing Systems’ Workshop on Machine Learning and the Physical Sciences

Number of Papers Accepted
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: [https://github.com/CCSI-Toolset/DeeperFluids](https://github.com/CCSI-Toolset/DeeperFluids)

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: [https://github.com/CCSI-Toolset/MGN](https://github.com/CCSI-Toolset/MGN)
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: [https://github.com/CCSI-Toolset/DeeperFluids](https://github.com/CCSI-Toolset/DeeperFluids)

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: [https://github.com/CCSI-Toolset/MGN](https://github.com/CCSI-Toolset/MGN)
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: https://github.com/CCSI-Toolset/DeeperFluids

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: https://github.com/CCSI-Toolset/MGN
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: https://github.com/CCSI-Toolset/DeeperFluids

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: https://github.com/CCSI-Toolset/MGN
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: [https://github.com/CCSI-Toolset/DeeperFluids](https://github.com/CCSI-Toolset/DeeperFluids)

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: [https://github.com/CCSI-Toolset/MGN](https://github.com/CCSI-Toolset/MGN)
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: [https://github.com/CCSI-Toolset/DeeperFluids](https://github.com/CCSI-Toolset/DeeperFluids)

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: [https://github.com/CCSI-Toolset/MGN](https://github.com/CCSI-Toolset/MGN)
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: [https://github.com/CCSI-Toolset/DeeperFluids](https://github.com/CCSI-Toolset/DeeperFluids)

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: [https://github.com/CCSI-Toolset/MGN](https://github.com/CCSI-Toolset/MGN)
We explore a range of ML surrogates

**DeeperFluids**
- Builds on Kim et al., 2019
- Interpolates CFD mesh onto regular grid
- Uses image-processing ML techniques
- Published in IAAI 2022 (Bartoldson et al., 2022)
- Included as a plugin for FOQUS
- Code: [https://github.com/CCSI-Toolset/DeeperFluids](https://github.com/CCSI-Toolset/DeeperFluids)

**MeshGraphNets**
- Builds on Pfaff et al., 2021
- Uses same mesh as CFD
- More faithful to physics than DF
- Code: [https://github.com/CCSI-Toolset/MGN](https://github.com/CCSI-Toolset/MGN)
Carbon capture systems have many parameters to tune.

Image source: Heldebrant et al., 2019
Carbon capture systems have many parameters to tune.

Image source: Heldebrant et al., 2019
Potential approach: Run a CFD simulation to understand effect of each parameter…

Image source: Heldebrant et al., 2019
Screen parameter settings to optimize efficiency---interfacial area (IA), liquid holdup, pressure drop, etc.
Problem: CFD is too slow to fully explore parameter space

Computational fluid dynamics (CFD) modeling

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0
\]

\[
\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g} + \mathbf{F}_\sigma
\]

\[
\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u} \alpha) = 0
\]

Numerical methods/software for CFD

- ANSYS FLUENT®
- OpenFOAM®
- COMSOL
- STAR-CCM+®
Problem: CFD is too slow to fully explore parameter space

Computational fluid dynamics (CFD) modeling

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \]

\[ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g} + \mathbf{F}_\sigma \]

\[ \frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u} \alpha) = 0 \]

Numerical methods/software for CFD

[Images of software logos: ANSYS FLUENT, OpenFOAM, COMSOL, STAR-CCM+]
Problem: CFD is too slow to fully explore parameter space

Computational fluid dynamics (CFD) modeling

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \]

\[ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g} + \mathbf{F}_\sigma \]

\[ \frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u} \alpha) = 0 \]

Numerical methods/software for CFD

[Images and logos of ANSYS FLUENT, OpenFOAM, COMSOL, and STAR-CCM+ are shown.]

[CCSI² logo is also present.]
Surrogates learn from CFD data, are tested against CFD data

50 2D simulations
(2D slices from 3D RCM)
Surrogates learn from CFD data, are tested against CFD data

50 2D simulations
(2D slices from 3D RCM)
Surrogates learn from CFD data, are tested against CFD data

50 2D simulations
(2D slices from 3D RCM)

Inlet Velocity (m/s)

Training Data
Test Data

Simulation Index
Deep Fluids (DF)

Encode input to low-dim latent space

Forward pass in latent space

Decide back to high-dim true space
LLNL’s Deeper Fluids Surrogates

Building on the original surrogates...

<table>
<thead>
<tr>
<th>H</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024, 512</td>
<td>0.22</td>
<td>0.53</td>
</tr>
<tr>
<td>128, 128, 128</td>
<td>0.64</td>
<td>0.53</td>
</tr>
</tbody>
</table>

We find better performance!

<table>
<thead>
<tr>
<th>LIN</th>
<th>s</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>1</td>
<td>0.12</td>
<td>0.53</td>
<td>0.09</td>
<td>0.55</td>
<td>0.12</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.15</td>
<td>0.53</td>
<td>0.14</td>
<td>0.54</td>
<td>0.15</td>
<td>0.53</td>
</tr>
<tr>
<td>LSTM</td>
<td>1</td>
<td>0.12</td>
<td>0.54</td>
<td>0.06</td>
<td>0.54</td>
<td>0.12</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.09</td>
<td>0.54</td>
<td>0.09</td>
<td>0.52</td>
<td>0.09</td>
<td>0.54</td>
</tr>
<tr>
<td>MLP</td>
<td>1</td>
<td>0.07</td>
<td>0.55</td>
<td>0.07</td>
<td>0.55</td>
<td>0.07</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.08</td>
<td>0.52</td>
<td>0.12</td>
<td>0.49</td>
<td>0.08</td>
<td>0.52</td>
</tr>
<tr>
<td>Transformer</td>
<td>1</td>
<td>0.08</td>
<td>0.55</td>
<td>0.04</td>
<td>0.52</td>
<td>0.08</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.06</td>
<td>0.80</td>
<td>0.22</td>
<td>0.53</td>
<td>0.06</td>
<td>0.80</td>
</tr>
</tbody>
</table>

And big speedups!

<table>
<thead>
<tr>
<th>LIN</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
<th>SV</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>0.07 (0.00)</td>
<td>0.49 (0.00)</td>
<td>4800</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.08 (0.01)</td>
<td>0.49 (0.01)</td>
<td>2700</td>
</tr>
<tr>
<td>MLP</td>
<td>0.06 (0.00)</td>
<td>0.47 (0.00)</td>
<td>5400</td>
</tr>
<tr>
<td>Transformer</td>
<td>0.08 (0.04)</td>
<td>0.51 (0.00)</td>
<td>4300</td>
</tr>
</tbody>
</table>
LLNL’s Deeper Fluids Surrogates

Building on the original surrogates...

<table>
<thead>
<tr>
<th>H</th>
<th>Error_{IA}</th>
<th>50</th>
<th>150</th>
<th>200</th>
<th>300</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024, 512</td>
<td>0.33</td>
<td>0.08</td>
<td>0.09</td>
<td>0.10</td>
<td>0.11</td>
<td>2.53</td>
</tr>
<tr>
<td></td>
<td>(0.14)</td>
<td>(0.00)</td>
<td>(0.00)</td>
<td>(0.00)</td>
<td>(0.00)</td>
<td>(0.20)</td>
</tr>
<tr>
<td>128, 128, 128</td>
<td>0.64</td>
<td>0.08</td>
<td>0.08</td>
<td>0.07</td>
<td>0.06</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>(0.11)</td>
<td>(0.01)</td>
<td>(0.00)</td>
<td>(0.00)</td>
<td>(0.00)</td>
<td>(0.18)</td>
</tr>
</tbody>
</table>

We find better performance!

<table>
<thead>
<tr>
<th>LIN</th>
<th>s</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$L_{RE}$</td>
<td>RMSE</td>
</tr>
<tr>
<td>ARC</td>
<td>1</td>
<td>0.12</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td>LSTM</td>
<td>1</td>
<td>0.12</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>MLP</td>
<td>1</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.08</td>
<td>0.12</td>
</tr>
<tr>
<td>Transformer</td>
<td>1</td>
<td>0.08</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.06</td>
<td>0.22</td>
</tr>
</tbody>
</table>

And big speedups!

<table>
<thead>
<tr>
<th>LIN</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
<th>Sw</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>0.07 (0.00)</td>
<td>0.49 (0.00)</td>
<td>4500</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.08 (0.01)</td>
<td>0.49 (0.01)</td>
<td>2700</td>
</tr>
<tr>
<td>MLP</td>
<td>0.06 (0.00)</td>
<td>0.47 (0.00)</td>
<td>5400</td>
</tr>
<tr>
<td>Transformer</td>
<td>0.08 (0.04)</td>
<td>0.51 (0.00)</td>
<td>4300</td>
</tr>
</tbody>
</table>
LLNL’s Deeper Fluids Surrogates

Building on the original surrogates...

<table>
<thead>
<tr>
<th>H</th>
<th>Error_{IA}</th>
</tr>
</thead>
<tbody>
<tr>
<td>w=20</td>
<td>0.33</td>
</tr>
<tr>
<td>50</td>
<td>0.08</td>
</tr>
<tr>
<td>150</td>
<td>0.06</td>
</tr>
<tr>
<td>200</td>
<td>0.10</td>
</tr>
<tr>
<td>300</td>
<td>0.11</td>
</tr>
<tr>
<td>499</td>
<td>2.53</td>
</tr>
<tr>
<td>1024, 512</td>
<td>(0.14)</td>
</tr>
<tr>
<td>128, 128, 128</td>
<td>(0.06)</td>
</tr>
<tr>
<td>0.64</td>
<td>(0.00)</td>
</tr>
<tr>
<td>0.88</td>
<td>(0.00)</td>
</tr>
<tr>
<td>0.08</td>
<td>(0.00)</td>
</tr>
<tr>
<td>0.07</td>
<td>0.06</td>
</tr>
<tr>
<td>2.25</td>
<td>(0.18)</td>
</tr>
</tbody>
</table>

We find better performance!

<table>
<thead>
<tr>
<th>LIN</th>
<th>s</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>L_{RE}</td>
<td>RMSE</td>
</tr>
<tr>
<td>ARC</td>
<td>1</td>
<td>0.12</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td>LSTM</td>
<td>1</td>
<td>0.12</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>MLP</td>
<td>1</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.08</td>
<td>0.12</td>
</tr>
<tr>
<td>Transformer</td>
<td>1</td>
<td>0.08</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.06</td>
<td>0.22</td>
</tr>
</tbody>
</table>

And big speedups!

<table>
<thead>
<tr>
<th>LIN</th>
<th>Error_{IA}</th>
<th>Error_{VF}</th>
<th>Sw</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>0.07 (0.00)</td>
<td>0.49 (0.00)</td>
<td>4800</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.08 (0.01)</td>
<td>0.49 (0.01)</td>
<td>2700</td>
</tr>
<tr>
<td>MLP</td>
<td>0.06 (0.00)</td>
<td>0.47 (0.00)</td>
<td>5400</td>
</tr>
<tr>
<td>Transformer</td>
<td>0.08 (0.04)</td>
<td>0.51 (0.00)</td>
<td>4300</td>
</tr>
</tbody>
</table>
MeshGraphNets (MGN)

$t = 0$
MeshGraphNets (MGN)
MeshGraphNets (MGN)

Input
A patch within the original frame

Nodes and edges are non-uniformly distributed.

Frame:
150073 Nodes
439935 Edges

Patch:
406 Nodes
1084 Edges
MeshGraphNets (MGN)

**Input**
A mesh within the original frame

**Encoding E**
Each node and edge has its own embedding.

**Message Passing M**
Neighboring edges and nodes exchange info to update embeddings.

**Decoding D**
Updated embeddings are decoded, which represent the gradient in physical space.

**Forward pass F**
Via forward Euler

\[
\hat{x}_t^* = M^n(E(\hat{x}_t)) \\
\hat{x}_t' = D(\hat{x}_t^*) \\
\hat{x}_{t+1} = \hat{x}_t + \hat{x}_t' \Delta t
\]
LLNL’s MGN Surrogates

Implemented the first public version of MGN in PyTorch

CCSI-Toolset/MGN repository is popular!
LLNL’s MGN Surrogates

Implemented the first public version of MGN in PyTorch

MGN:

CFD:

CCSI-Toolset/MGN repository is popular!
LLNL’s MGN Surrogates

Implemented the first public version of MGN in PyTorch

CCSI-Toolset/MGN repository is popular!
LLNL’s MGN Surrogates

Implemented the first public version of MGN in PyTorch

MGN:

CFD:

CCSI-Toolset/MGN repository is popular!
LLNL’s MGN Surrogates

Implemented the first public version of MGN in PyTorch

MGN:

CFD:

CCSI-Toolset/MGN repository is popular!
**MGN Surrogate Innovations for Scaling to PNNL Data**

**Scaled MGN**: Trained MGN on PNNL data via **graph partitioning** and **distributed computing**.
**MGN Surrogate Innovations for Scaling to PNNL Data**

**Scaled MGN:** Trained MGN on PNNL data via *graph partitioning* and *distributed computing*

- Momentum\(_i\):
- Momentum\(_j\):
- Volume fraction:
MGN Surrogate Innovations for Scaling to PNNL Data

**Scaled MGN:** Trained MGN on PNNL data via **graph partitioning** and **distributed computing**

Momentum\_i:

Momentum\_j:

Volume fraction:
Physics-informed surrogates

\[
\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho uu) = -\nabla p + \mu \nabla^2 u + \rho g + F_o
\]

Velocity Model Input

Difficult to predict next velocity state!

Desired Output

Momentum Model Input

Easy to predict next momentum state!

Desired Output
Work in progress

- MGN extensions/improvements
  - Improving scalability
  - Architectural improvements
  - Training/optimization improvements
    - Curriculum learning
    - Incorporating physics constraints/expert knowledge
  - Explainability methods
- Surrogate models with CMU/PNNL data
- Design optimization
  - Integrate MGN into FOQUS

https://github.com/CCSI-Toolset/MGN
Acknowledgements

CCSI² Machine Learning Team

- Phan Nguyen, LLNL
- Brian Bartoldson, LLNL
- Jose Cadena, LLNL
- Amar Saini, LLNL
- Yeping Hu, LLNL

Subtask Collaborators

- Jay Xu, PNNL
- Jie Bao, PNNL
- Yucheng Fu, PNNL
- Grigoris Panagakos, CMU
- Charles Finney, ORNL
For more information
https://www.acceleratecarboncapture.org/

Contact Info:
Phan Nguyen
nguyen97@llnl.gov
Questions