Deep Neural Networks for Data-Driven Turbulence Models

@ICERM 2019: Scientific Machine Learning
Outline

1. Introduction
2. Machine Learning with Neural Networks
3. Turbulence Models from Data
4. Training and Results
5. Summary and Conclusion
Introduction
Introduction

- **Numerics Research Group** @ IAG, University of Stuttgart, Germany
- Primary Focus: High Order **Discontinuous Galerkin** Methods
- **OpenSource** HPC solver for the compressible Navier-Stokes equations
DG-SEM in a nutshell

- Hyperbolic/parabolic conservation law, e.g., compressible Navier-Stokes Equations
  \[ U_t + \vec{\nabla} \cdot \vec{F}(U, \vec{\nabla}U) = 0 \]

- Variational formulation and weak DG form per element for the equation system
  \[ \langle JU_t, \psi \rangle_E + \left( \tilde{f}^* \tilde{n} \xi, \psi \right)_{\partial E} - \left( \tilde{F}, \nabla \xi \psi \right)_E = 0, \]

- Local tensor-product Lagrange polynomials, interpolation nodes equal to quadrature nodes

- Tensor-product structure in multi-D: line-by-line operations

\[
\begin{align*}
(U_{ij})_t + \frac{1}{J_{ij}} \left[ \tilde{f}^*(1, \eta_j)\hat{\psi}_i(1) - \tilde{f}^*(-1, \eta_j)\hat{\psi}_i(-1) + \sum_{k=0}^{N} \hat{D}_{ik} \tilde{F}_{kj} \right] \\
+ \frac{1}{J_{ij}} \left[ \tilde{g}^*(\xi_i, 1)\hat{\psi}_j(1) - \tilde{g}^*(\xi_i, -1)\hat{\psi}_j(-1) + \sum_{k=0}^{N} \hat{D}_{jk} \tilde{G}_{ik} \right] = 0
\end{align*}
\]

- BR1/2 lifting for viscous fluxes, Roe/LF/HLL-type inviscid fluxes, explicit in time by RK/Legendre-Gauss or LGL-nodes
Applications: LES, moving meshes, acoustics, multiphase, UQ, particle-laden flows...
Rationale for Machine Learning

“It is very hard to write programs that solve problems like recognizing a three-dimensional object from a novel viewpoint in new lighting conditions in a cluttered scene.

• We don’t know what program to write because we don’t know how its done in our brain.
• Even if we had a good idea about how to do it, the program might be horrendously complicated.”

Geoffrey Hinton, computer scientist and cognitive psychologist (h-index:140+)
An attempt at a definition:

Machine learning describes algorithms and techniques that progressively improve performance on a specific task through data without being explicitly programmed.

Learning Concepts

- Unsupervised Learning
- Supervised Learning
- Reinforcement Learning

Artificial Neural Networks

- General Function Approximators
- AlphaGo, Self-Driving Cars, Face recognition, NLP
- Incomplete Theory, models difficult to interpret
- NN design: more an art than a science
Neural Networks

- **Artificial Neural Network (ANN):** A non-linear mapping from inputs to outputs: \( M : \hat{X} \rightarrow \hat{Y} \)
- An ANN is a nesting of linear and non-linear functions arranged in a directed acyclic graph:
  \[
  \hat{Y} \approx Y = M(\hat{X}) = \sigma_L (W_L (\sigma_{L-1} (W_{L-1} (\sigma_{L-2} (...W_1(\hat{X})))))) \tag{1}
  \]
  with \( W \) being an affine mapping and \( \sigma \) a non-linear function
- The entries of the mapping matrices \( W \) are the parameters or weights of the network: improved by training
- Cost function \( C \) as a measure for \( |\hat{Y} - Y| \), (MSE / \( L_2 \) error) convex w.r.t to \( Y \), but not w.r.t \( W \):
  \( \Rightarrow \) non-convex optimization problem requires a lot of data
Advanced Architectures

- Convolutional Neural Networks
  - Local connectivity, multidimensional trainable filter kernels, discrete convolution, shift invariance, hierarchical representation
  - Current state of the art for multi-D data and segmentation

Filter Kernel $W_{mn}^{l-1,g}$,
$g = 1, \ldots, f$

Activation $A^{l-1}$

Local Receptive Fields
$\Delta_i = 3, \Delta_j = 3$

Feature Maps $A^{l,g}$
What does a CNN learn?

- Representation in hierarchical basis

Residual Neural Networks

- He et al. recognized that the prediction performance of CNNs may deteriorate with depths (not an overfitting problem)
- Introduction of skip connectors or shortcuts, most often identity mappings
- A sought mapping, e.g. $G(A^{l-3})$ is split into a linear and non-linear (residual) part
- Fast passage of the linear part through the network: hundreds of CNN layers possible
- More robust identity mapping

\[ A^{l-3} \xrightarrow{\text{Conv}_{l-3}} A^{l-2} \xrightarrow{\text{Conv}_{l-2}} A^{l-1} \xrightarrow{\text{Conv}_{l-1}} A^l \]

\[ A^l = F(A^{l-3}) \]

\[ f(h(A^{l-3}) + F(A^{l-3})) \]

Turbulence Models from Data
Turbulence in a nutshell

- **Turbulent fluid motion** is prevalent in naturally occurring flows and engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of **non-linear PDEs** (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: **Coarse scale formulation** of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a **closure term / regularization** dependent on the filtered full scale solution $\Rightarrow$ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (**RANS**) or low-pass filter in space (**LES**)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model
Turbulence in a nutshell

- Turbulence is a multiscale problem in space and time.
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.).
- Fullscale resolution (DNS) is rarely feasible: Coarse scale formulation of NSE is necessary.
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term / regularization dependent on the filtered full scale solution.
- Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES).
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model.
Turbulence in a nutshell

- **Turbulent fluid motion** is prevalent in naturally occurring flows and engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term / regularization dependent on the filtered full scale solution ⇒ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (**RANS**) or low-pass filter in space (**LES**)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model
Turbulence in a nutshell

- Turbulent fluid motion is prevalent in naturally occurring flows and engineering applications:
  - multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term /
  regularization dependent on the filtered full scale solution
  \[ \Rightarrow \]
  Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model

A Beck, IAG: DNN for LES
Turbulence in a nutshell

- Turbulent fluid motion is prevalent in naturally occurring flows and engineering applications: multiscale problem in space and time
- Navier-Stokes equations: system of non-linear PDEs (hyp. / parab.)
- Fullscale resolution (DNS) rarely feasible: Coarse scale formulation of NSE is necessary
- Filtering the NSE: Evolution equations for the coarse scale quantities, but with a closure term / regularization dependent on the filtered full scale solution ⇒ Model depending on the coarse scale data needed!
- Two filter concepts: Averaging in time (RANS) or low-pass filter in space (LES)
- An important consequence: RANS can be discretization independent, LES is (typically) not!
- 50 years of research: Still no universal closure model
Idea

- Approximating an unknown, non-linear and possibly hierarchical mapping from high-dimensional input data to an output ⇒ ANN
Approximating an unknown, non-linear and possibly hierarchical mapping from high-dimensional input data to an output ⇒ LES closure

\[ \hat{Y} = M(\hat{X}) \]
Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit ⇔ explicit, linear ⇔ non-linear, discrete ⇔ continuous...
  - Numerical operator: negligible ⇔ part of the LES formulation, isotropic ⇔ non-isotropic, commutation with filter...
  - Subgrid closure: implicit ⇔ explicit, deconvolution ⇔ stochastic modelling,...
Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit $\iff$ explicit, linear
  $\iff$ non-linear, discrete
  $\iff$ continuous...

- Numerical operator: negligible $\iff$ part of the LES formulation, isotropic, non-isotropic, commutation with filter...

- Subgrid closure: implicit $\iff$ explicit, deconvolution $\iff$ stochastic modelling,...
Problem Definition

- Choice of LES formulation:
  - Scale separation filter: implicit $\Leftrightarrow$ explicit, linear $\Leftrightarrow$ non-linear, discrete $\Leftrightarrow$ continuous...
  - Numerical operator: isotropic $\Leftrightarrow$ non-isotropic, commutation with filter...
  - Subgrid closure: implicit $\Leftrightarrow$ explicit, deconvolution $\Leftrightarrow$ stochastic modelling,...
Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit $\Leftrightarrow$ explicit, linear $\Leftrightarrow$ non-linear, discrete $\Leftrightarrow$ continuous...
  - Numerical operator: negligible $\Leftrightarrow$ part of the LES formulation, isotropic $\Leftrightarrow$ non-isotropic, commutation with...
  - Subgrid closure: implicit $\Leftrightarrow$ explicit, deconvolution $\Leftrightarrow$ stochastic modelling,...

![Graph showing velocity profiles](image)
Problem Definition

- Choice of LES formulations:
  - Scale separation filter: implicit ⇔ explicit, linear ⇔ non-linear, discrete ⇔ continuous...
  - Numerical operator: negligible ⇔ part of the LES formulation, isotropic ⇔ non-isotropic, commutation with filter...
  - Subgrid closure: implicit ⇔ explicit, deconvolution ⇔ stochastic modelling,...

- Essential for ML methods: Well-defined training data (both input and output)
- Is $U$ known explicitly? ⇒ For practical LES, i.e. grid-dependent LES, it is not most of the time!
Problem Definition

- **Choice of LES formulations:**
  - Scale separation filter: implicit ⇔ explicit, linear ⇔ non-linear, discrete ⇔ continuous...
  - Numerical operator: negligible ⇔ part of the LES formulation, isotropic ⇔ non-isotropic, commutation with filter...
  - Subgrid closure: implicit ⇔ explicit, deconvolution ⇔ stochastic modelling,...

- **Essential for ML methods:** *Well-defined* training data (both input and output)
- **Is \( \overline{U} \) known explicitly?** ⇒ For practical LES, i.e. *grid-dependent* LES, it is not most of the time!

**Definition: Perfect LES**

- All terms must be computed on the coarse grid
- Given \( \overline{U}(t_0, x) = \overline{U^{DNS}}(t_0, x) \quad \forall \, x \), then \( \overline{U}(t, x) = \overline{U^{DNS}}(t, x) \quad \forall \, x \) and \( \forall \, t > 0 \)
Turbulence Closure

- Filtered NSE:
  \[
  \frac{\partial U}{\partial t} + R(F(U)) = 0
  \]  

- Imperfect closure with \( \hat{U} \neq U \):
  \[
  \frac{\partial \hat{U}}{\partial t} + R(F(\hat{U})) = M(\hat{U}, C,k)
  \]  

- Perfect closure with:
  \[
  \frac{\partial U}{\partial t} + R(F(U)) = R(F(U)) - R(F(U)) = 0
  \]
Turbulence Closure

- Filtered NSE:

\[
\frac{\partial \overline{U}}{\partial t} + \overline{R(F(U))} = 0
\]  \hspace{1cm} (2)

- Imperfect closure with \( \hat{U} \neq \overline{U} \):

\[
\frac{\partial \hat{U}}{\partial t} + \tilde{R}(F(\hat{U})) = \tilde{M}(\hat{U}, C_k)
\]

imperfect closure model

(3)

- Perfect closure with \( \overline{U} \)

\[
\frac{\partial \overline{U}}{\partial t} + \tilde{R}(F(\overline{U})) = \tilde{R}(F(\overline{U})) - \overline{R(F(U))}
\]

perfect closure model

(4)

- Note \( \tilde{R}(F(\overline{U})) \) is necessarily a part of the closure, but it is known

- Perfect LES and perfect closure are not new concepts: introduced by R. Moser et al in a series of papers\(^*\), termed ideal / optimal LES

Perfect LES

\[ \frac{\partial \tilde{U}}{\partial t} + \tilde{R}(F(\tilde{U})) = \tilde{R}(F(\tilde{U})) - \tilde{R}(F(U)) \]

- The specific operator and filter choices are **not relevant** for the perfect LES.
- Note that the coarse grid operator is part of the closure (and cancels with the LHS).
- We choose:
  - DNS-to-LES operator \( \tilde{\cdot} \): \( L_2 \) projection from DNS grid onto LES grid: We choose a discrete scale-separation filter.
  - LES operator \( \tilde{\cdot} \): 6\(^{th}\) order DG method with split flux formulation and low dissipation Roe flux.
Perfect LES

- Perfect LES runs with closure term from DNS
- Decaying homogeneous isotropic turbulence
- DNS grid: $64^3$ elements, $N = 7$; LES grid: $8^3$ elements, $N = 5$;

Left to right: a) DNS, b) filtered DNS, c) computed perfect LES d) LES with Smagorinsky model $C_s = 0.17$
Perfect LES

- Perfect LES runs with closure term from DNS
- Decaying homogeneous isotropic turbulence
- DNS grid: $64^3$ elements, $N = 7$; LES grid: $8^3$ elements, $N = 5$;

$$E(k)$$

⇒ Perfect LES gives well-defined target and input data for supervised with NN
Training and Results
Data Acquisition: Decaying Homogeneous Isotropic Turbulence

- Ensemble of DNS runs of decaying homogeneous isotropic turbulence with initial spectrum defined by Chasnov (1995) initialized by Rogallo (1981) procedure and $Re_\lambda = 180$ at start
- Data collection in the range of exponential energy decay: 25 DHIT realizations with 134 Mio DOF each computed on CRAY XC40 (approx. 400,000 CPUh, 8200 cores)
- Compute coarse grid terms from DNS-to-LES operator
Features and Labels

- Each sample: A single LES grid cell with $6^3$ solution points
- Input features: velocities and LES operator: $\overline{u}_i, \widetilde{R}(F(U))$
- Output labels: DNS closure terms on the LES grid $\overline{R}(F(U))$

$$\hat{X} = \left\{ \hat{x} \in \mathbb{R}^{6 \times p \times p \times p} \mid \hat{x} = (\overline{u}_{ijk}, \overline{v}_{ijk}, \overline{w}_{ijk}, \widetilde{R}(F(U^1))_{ijk}, \widetilde{R}(F(U^2))_{ijk}, \widetilde{R}(F(U^3))_{ijk}, \text{ with } i, j, k = 0, \ldots, p - 1 \right\}$$

$$\hat{Y} = \left\{ \hat{y} \in \mathbb{R}^{3 \times p \times p \times p} \mid \hat{y} = \overline{R}(F(U))^n_{ijk}, \text{ with } n = 1, \ldots, 3; i, j, k = 0, \ldots, p - 1 \right\}$$
Networks and Training

- CNNs with skip connections (RNN), batch normalization, ADAM optimizer, data augmentation
- Different network depths (no. of residual blocks)
- For comparison: MLP with 100 neurons in 1 hidden layer
- Implementation in Python / Tensorflow, Training on K40c and P100 at HLRS
- Split in training, semi-blind validation and blind test DHIT runs

Training Results I: Costs

- Cost function for different network depths
- RNNs outperform MLP, deeper networks learn better
- The approach is data-limited! NNs are very data-hungry!
## Training Results II: Correlation

<table>
<thead>
<tr>
<th>Network</th>
<th>$a, b$</th>
<th>$CC(a, b)$</th>
<th>$CC_{inner}(a, b)$</th>
<th>$CC_{surf}(a, b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN0</td>
<td>$\frac{R(F(U))^1}{\text{ANN}}$, $\frac{R(F(U))^1}{\text{ANN}}$</td>
<td>0.347676</td>
<td>0.712184</td>
<td>0.149090</td>
</tr>
<tr>
<td></td>
<td>$\frac{R(F(U))^2}{\text{ANN}}$, $\frac{R(F(U))^2}{\text{ANN}}$</td>
<td>0.319793</td>
<td>0.663664</td>
<td>0.134267</td>
</tr>
<tr>
<td></td>
<td>$\frac{R(F(U))^3}{\text{ANN}}$, $\frac{R(F(U))^3}{\text{ANN}}$</td>
<td>0.326906</td>
<td>0.669931</td>
<td>0.101801</td>
</tr>
<tr>
<td>RNN4</td>
<td>$\frac{R(F(U))^1}{\text{ANN}}$, $\frac{R(F(U))^1}{\text{ANN}}$</td>
<td>0.470610</td>
<td>0.766688</td>
<td>0.253925</td>
</tr>
<tr>
<td></td>
<td>$\frac{R(F(U))^2}{\text{ANN}}$, $\frac{R(F(U))^2}{\text{ANN}}$</td>
<td>0.450476</td>
<td>0.729371</td>
<td>0.337032</td>
</tr>
<tr>
<td></td>
<td>$\frac{R(F(U))^3}{\text{ANN}}$, $\frac{R(F(U))^3}{\text{ANN}}$</td>
<td>0.449879</td>
<td>0.730491</td>
<td>0.269407</td>
</tr>
</tbody>
</table>

- High correlation achievable with deep networks
- For surfaces: one-sidedness of data / filter kernels
### Training Results III: Feature Sensitivity

<table>
<thead>
<tr>
<th>Set</th>
<th>Features</th>
<th>(CC^1)</th>
<th>(CC^2)</th>
<th>(CC^3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(u_i, \tilde{R}(F(U^i)), i = 1, 2, 3)</td>
<td>0.4706</td>
<td>0.4505</td>
<td>0.4499</td>
</tr>
<tr>
<td>2</td>
<td>(u_i, i = 1, 2, 3)</td>
<td>0.3665</td>
<td>0.3825</td>
<td>0.3840</td>
</tr>
<tr>
<td>3</td>
<td>(\tilde{R}(F(U^i)), i = 1, 2, 3)</td>
<td>0.3358</td>
<td>0.3066</td>
<td>0.3031</td>
</tr>
<tr>
<td>4</td>
<td>(\rho, p, e, u_i, \tilde{R}(F(U^i)), i = 1, 2, 3)</td>
<td>0.4764</td>
<td>0.4609</td>
<td>0.4580</td>
</tr>
<tr>
<td>5</td>
<td>(u_1, \tilde{R}(F(U^1)))</td>
<td>0.3913</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Feature sets and resulting test correlations. \(CC^i\) with \(i = 1, 2, 3\) denotes the cross correlation between the targets and network outputs \(CC(\tilde{R}(F(U)^i), \tilde{R}(F(U))^i_{ANN})\). Set 1 corresponds to the original feature choice; Set 5 corresponds to the RNN4 architecture, but with features and labels for the \(u\)–momentum component only.

- Both the coarse grid primitive quantities as well as the coarse grid operator contribute strongly to the learning success
- Better learning for 3D cell data than pointwise data
Training Results IV: Visualization

- "Blind" application of the trained network to unknown test data
- Cut-off filter: no filter inversion / approximate deconvolution

\[
R(F(U))^1, \quad R(F(U))^1_{\text{ANN}_4}, \quad R(F(U))^1_{\text{ANN}_0}
\]

\[CC \approx 0.47, \quad CC \approx 0.34\]
LES with NN-trained model I

\[
\frac{\partial \tilde{U}}{\partial t} + \tilde{R}(F(\tilde{U})) = \tilde{R}(F(U)) - \tilde{R}(F(U)) \quad \text{(ANN closure)}
\]

- Perfect LES is possible, but the NN-learned mappings are approximate
- No long term stability, but short term stability and dissipation
LES with NN-trained model II

\[
\frac{\partial \overline{U}}{\partial t} + \tilde{R}(F(\overline{U})) = \tilde{R}(F(\overline{U})) - \overline{R}(F(U))
\]

\hspace{1cm} \text{data-based eddy viscosity model}

- Simplest model: \textbf{Eddy viscosity approach} with \( \mu_{ANN} \) from

\[
\tilde{R}(F(\overline{U}^i)) - \overline{R}(F(U^i)) \approx \mu_{ANN} \tilde{R}(F^{visc}(U^i, \nabla U^i))
\]  

\[5\]
Summary and Conclusion
Summary

- Perfect / optimal LES framework: well-defined target quantities for learning
- Learning the exact closure terms from data is possible
- Deeper RNNs learn better
- High order methods are a natural fit to CNN: volume data
- Our process is data-limited, i.e. learning can be improved with more data
- Achievable $CC \approx 45\%$, with up to $\approx 75\%$ for inner points
- Both the coarse grid velocities and the coarse grid operator contribute strongly to learning
- The resulting ANN models are dissipative
- No long term stability due to approximate model
- Simplest way to construct a stable model: Data-informed, local eddy-viscosity
- Other approaches to construct models from prediction of closure terms under investigation
flexi-project.org

Thank you for your attention!
History of ANNs

- Some important publications:
  - McCulloch-Pitts (1943): First compute a weighted sum of the inputs from other neurons plus a bias: the perceptron
  - Rosenblatt (1958): First to generate MLP from perceptrons
  - Rosenblatt (1962): Perceptron Convergence Theorem
  - Minsky and Papert (1969): Limitations of perceptrons
  - Rumelhart and Hinton (1986): Backpropagation by gradient descent
  - Ioffe (2015): Batch normalization
  - He et al. (2016): Residual networks
  - AlphaGo, DeepMind...
Closure Terms for LES

- For grid dependent LES: coarse grid operator is part of the closure
- Dual role of closure: cancel operator effects and model unknown term
- DNS grid: $64^3$ elements, $N = 7$; LES grid: $8^3$ elements, $N = 5$;

Figure: Left to right: a) DNS, b) filtered DNS, c) computed perfect LES d) LES with Smagorinsky model $C_s = 0.17$
Some thoughts on data-informed models, engineering and HPC

- Machine Learning is not a silver bullet
- First successes: ML can help build subscale models from data, not just for turbulence
- A lot of representative data is needed... maybe we already have the data? Computations, experiments...
- In this work, the computational times were: DNS: $O(10^5)$ CPUh, data preparation $O(10^3)$, Training the RNN: $O(10^1 - 10^2)$: Is it worth it?
- Incorporating physical constraints (e.g. realizability, positivity) field of research
- Self-learning algorithms: Reinforcement learning
- "Philosophical aspects": Interpretability of the models and "who should learn what?"
- HPC: Training has to done on GPUs (easy for supervised learning, bit more complicated for reinforcement learning), but ...
- What about model deployment? GPU (native) or CPU (export model)?
- Coupling of CFD solver (Fortran) to Neural Network (python): In our case, f2py is a very cumbersome solution
- Hybrid CPU/GPU codes, or rewrite it all for the GPU?
- Data storage policy: where to compute/store the data (reproducibility)
### Definitions and Concepts

**An attempt at a definition:**

Machine learning describes algorithms and techniques that progressively improve performance on a specific task through data without being explicitly programmed.

### Learning Concepts

- Unsupervised Learning
- Supervised Learning
- Reinforcement Learning

### Artificial Neural Networks

- General Function Approximators
- AlphaGo, Self-Driving Cars, Face recognition, NLP
- Incomplete Theory, models difficult to interpret