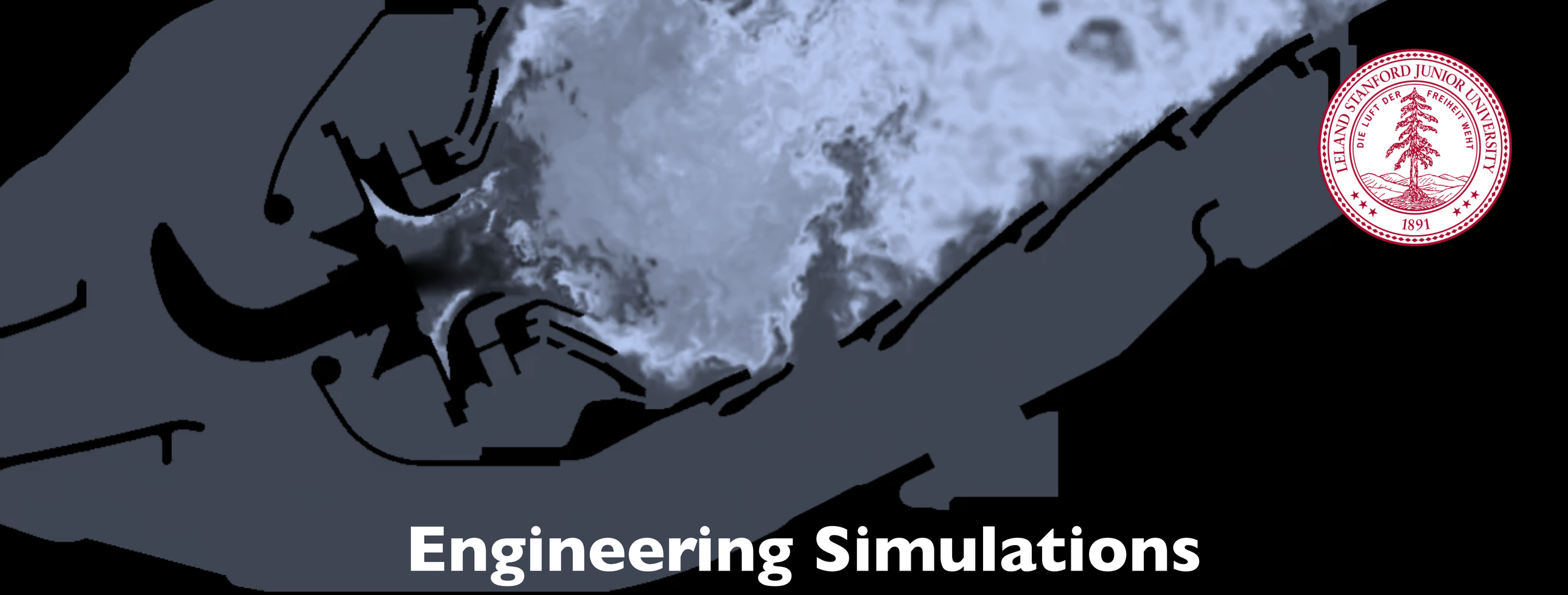


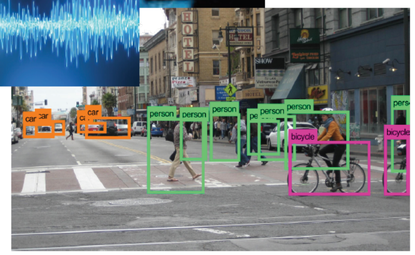
Engineering Simulations in the Age of Data

Gianluca Iaccarino

Mechanical Engineering & Institute for Computational Mathematical Engineering
Stanford University

jops@stanford.edu

The Age of Data



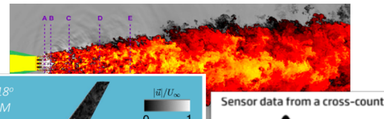
Software for Data

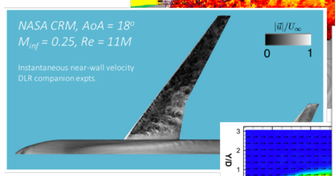
TensorFlow

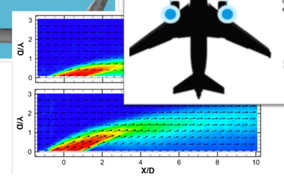
Keras

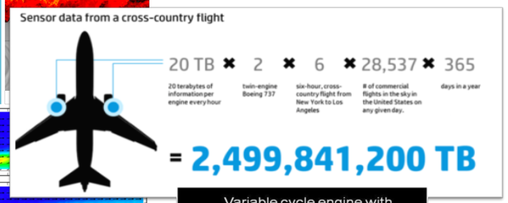
PYTORCH

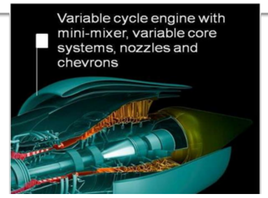
Data



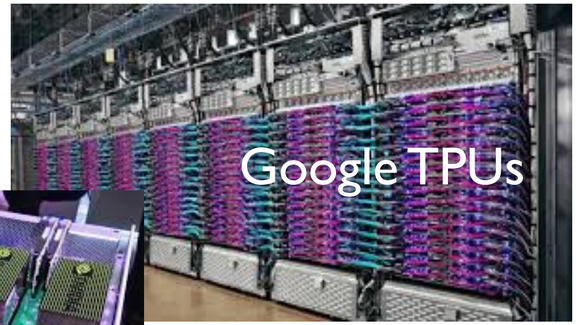


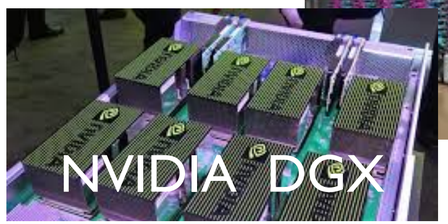






Computing for Data





Engineering Simulations
...or...
the Science of Predictions

Weather Predictions

C. Abbe, Physical Basis of Long Range Weather Forecast, 1901

24th, both preceded by heavy swell and followed by low dew-point. The north wind evidently precipitated the terrific downpour on north Hilo. There was lightning reported from Hawaii for the 8th, 9th, 13th, 24th, and 25th, and Maui on the 8th. Snow fell on Mauna Kea and Mauna Loa on the 8th and 24th; on Haleakala on the 8th or 9th. Earthquake reported at Hilo, 7:30 p. m. on the 2d.

AN AURORAL-LUNAR HALO DISPLAY.

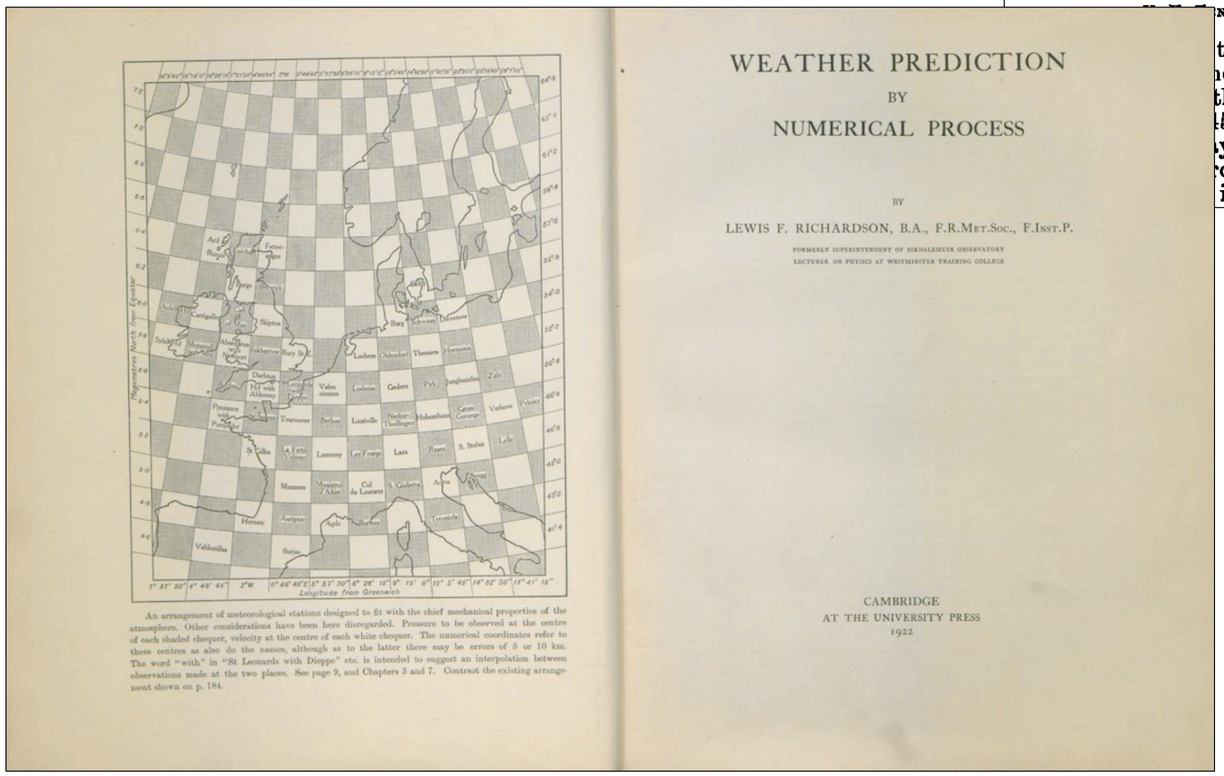
N BROECK, Braidentown, Fla., dated December 29, 1901.

t of the 28th I observed an auroral display with ding about west-northwest to east-southeast. the time an unusually brilliant halo around the 5° in diameter. The upper half was fringed on ys about 3° or 4° long, a few much longer, ra- om the moon (the center of the halo), but from it about east-southeast on the horizon. Some

THE PHYSICAL BASIS OF LONG-RANGE WEATHER FORECASTS.[1]

By PROF. CLEVELAND ABBE.

The expression "long range" must not be misunderstood. It refers only to the length of time intervening between the date of making a weather prediction and the date when we expect it to be fulfilled. At the present time, by the help of the daily weather map, the official weather forecasters of this country, and indeed of every civilized nation on the globe, publish forecasts, in detail, of approaching weather changes, and especially storms, for one and two, or possibly occasionally three days in advance. These predictions all relate to comparatively minute details for regions that have been charted and studied daily for many years. They merely represent the direct teachings of experience; they are generalizations based upon observations but into which physical theories have as yet entered in only a superficial manner if at all. They



An arrangement of meteorological stations designed to fit with the chief mechanical properties of the atmosphere. Other considerations have been here disregarded. Pressure to be observed at the centre of each shaded chequer, velocity at the centre of each white chequer. The numerical coordinates refer to these centres as also do the names, although as to the latter there may be errors of 5 or 10 km. The word "with" in "St Leonards with Dieppe" etc. is intended to suggest an interpolation between observations made at the two places. See page 9, and Chapters 3 and 7. Contrast the existing arrangement shown on p. 184.

WEATHER PREDICTION

BY

NUMERICAL PROCESS

BY

LEWIS F. RICHARDSON, B.A., F.R.MET.SOC., F.INST.P.

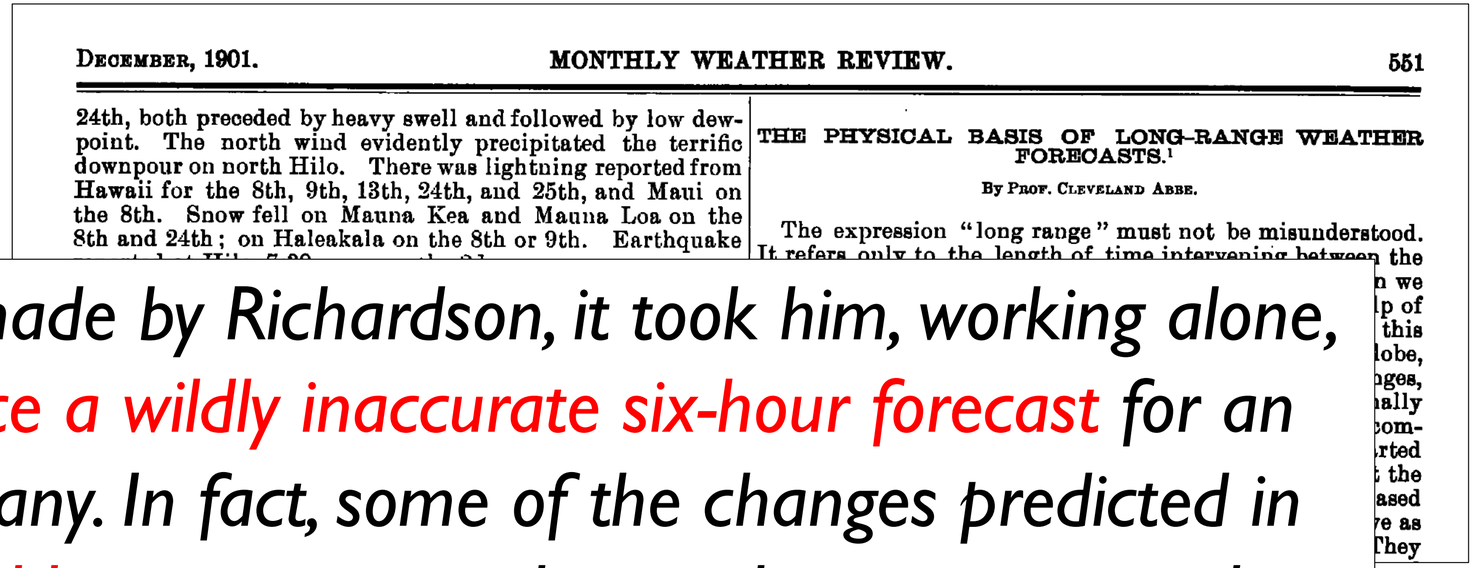
FORMERLY SUPERINTENDENT OF Eskdalemuir OBSERVATORY
LECTURER ON PHYSICS AT WESTMINSTER TRAINING COLLEGE

CAMBRIDGE
AT THE UNIVERSITY PRESS
1922

L. F. Richardson, Weather Predictions by Numerical Process 1922

Weather Predictions

C. Abbe, Physical Basis of Long Range Weather Forecast, 1901



DECEMBER, 1901. MONTHLY WEATHER REVIEW. 551

24th, both preceded by heavy swell and followed by low dew-point. The north wind evidently precipitated the terrific downpour on north Hilo. There was lightning reported from Hawaii for the 8th, 9th, 13th, 24th, and 25th, and Maui on the 8th. Snow fell on Mauna Kea and Mauna Loa on the 8th and 24th; on Haleakala on the 8th or 9th. Earthquake

THE PHYSICAL BASIS OF LONG-RANGE WEATHER FORECASTS.[1]

By Prof. Cleveland Abbe.

The expression "long range" must not be misunderstood. It refers only to the length of time intervening between the

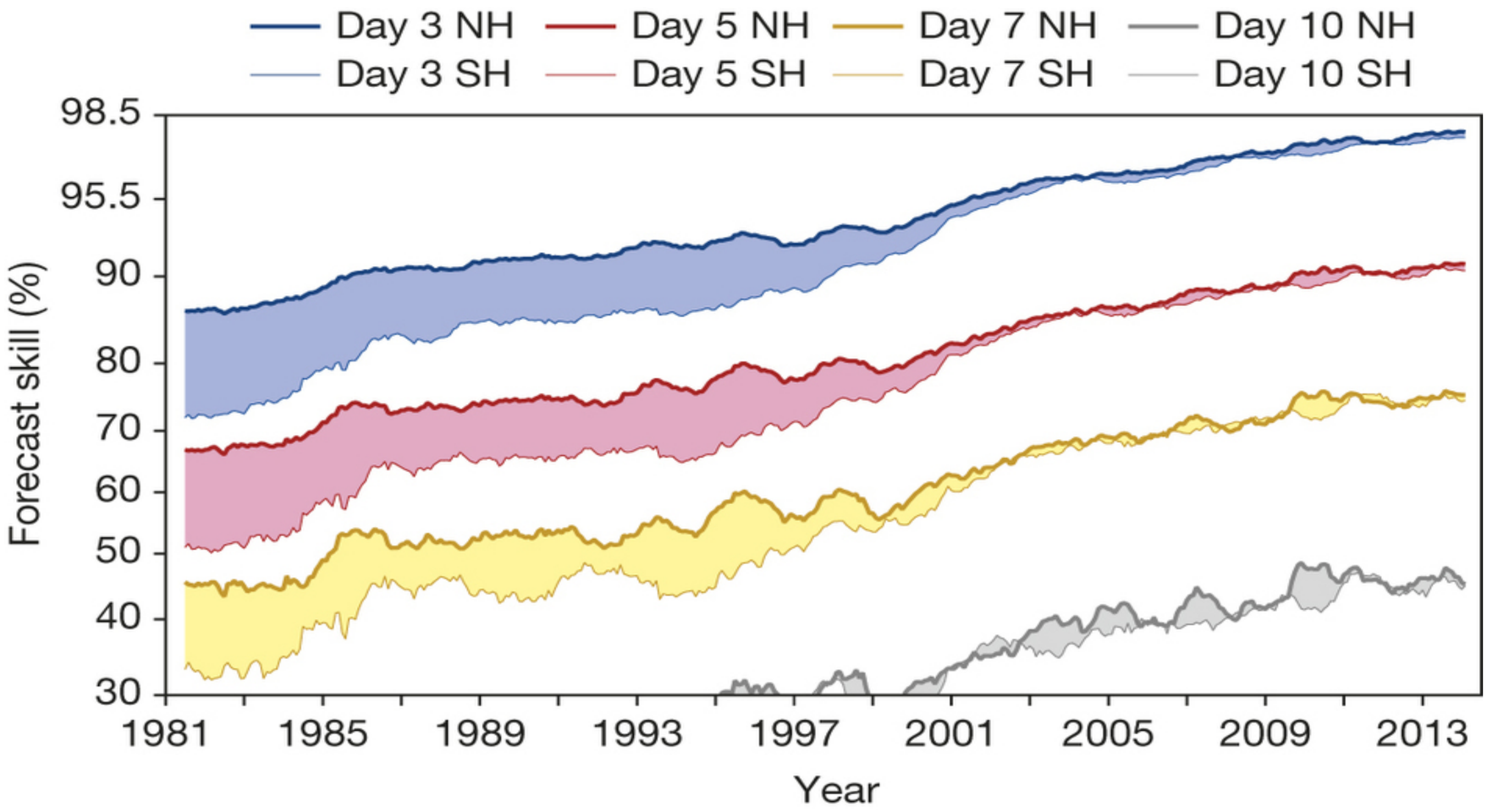
"Despite the advances made by Richardson, it took him, working alone, several months to produce a wildly inaccurate six-hour forecast for an area near Munich, Germany. In fact, some of the changes predicted in Richardson's forecast could never occur under any known terrestrial conditions. Adding to the failure of this effort, a six-hour forecast is not particularly useful if it takes weeks to produce."
"Weather Forecast through the Ages", 2007

An arrangement of meteorological stations designed to fit with the chief mechanical properties of the atmosphere. Other considerations have been here disregarded. Pressure to be observed at the centre of each shaded chequer, velocity at the centre of each white chequer. The numerical coordinates refer to these centres as also do the names, although as to the latter there may be errors of 5 or 10 km. The word "with" in "St Leonards with Dieppe" etc. is intended to suggest an interpolation between observations made at the two places. See page 9, and Chapters 3 and 7. Contrast the existing arrangement shown on p. 184.

CAMBRIDGE
AT THE UNIVERSITY PRESS
1922

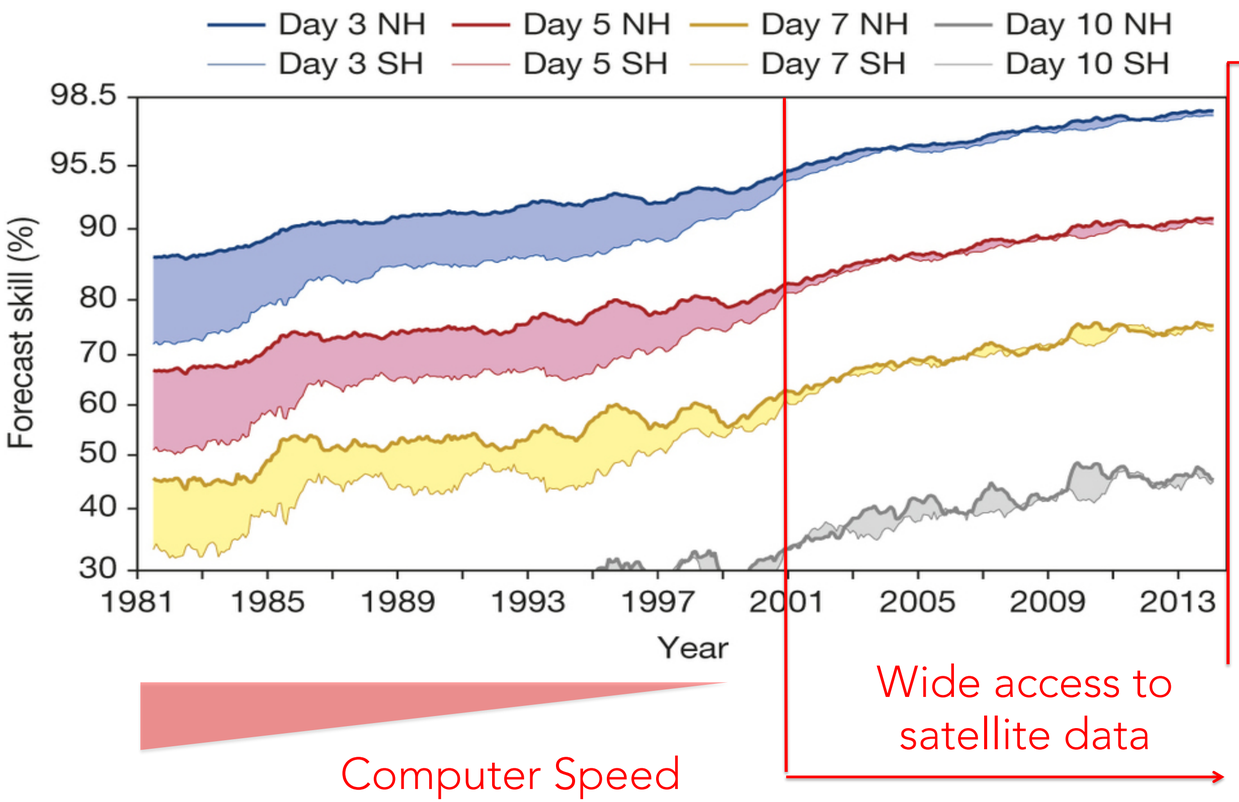
L. F. Richardson, Weather Predictions by Numerical Process 1922

Are weather predictions predictive?



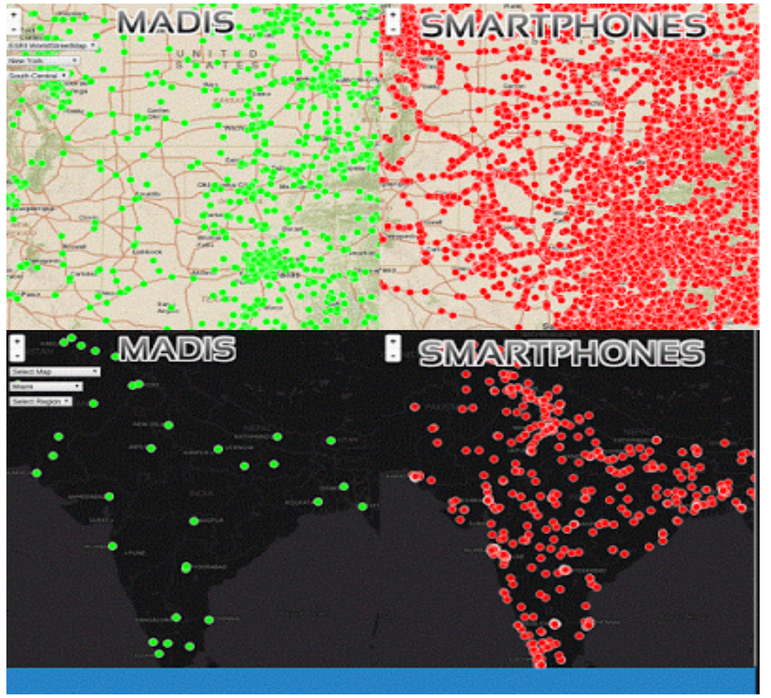
The quiet revolution of numerical weather prediction (Bauer et al., Nature 2015)

Are weather predictions predictive?



Next revolution

Smartphone Crowdsourcing?
(Big Data)



US

India

MADIS: Meteorological Assimilation Data Ingest System

The quiet revolution of numerical weather prediction (Bauer et al., Nature 2015)

(Engineering) Simulations + **Data**

Simulations + Data

Validation

The “C” word

Statistical Inference

Machine Learning

Physics + Data

Simulations + Data

Validation

The “C” word

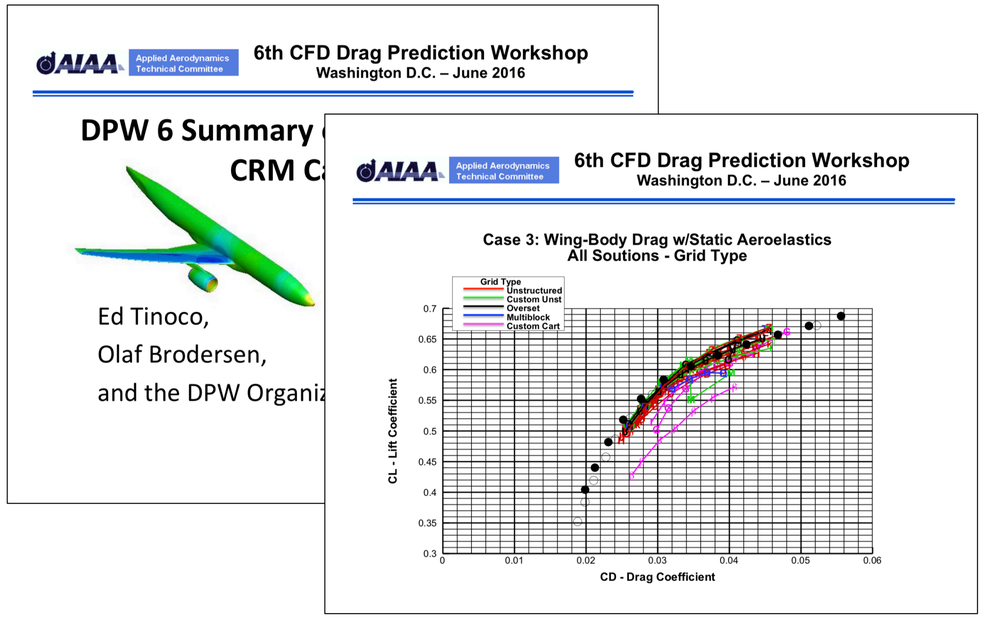
Statistical Inference

Machine Learning

Physics + Data

Blind Tests: Accuracy Evaluation

simulation → $y \neq y_{data}$ ← data



Simulations + Data

Validation

The "C" word

Statistical Inference

Machine Learning

Physics + Data

Calibration: adjust (tune) model parameters to match data

simulation → $y \sim y_{data}$ ← data

$c_1, c_2, \ldots c_M$

		Mixing Layer	Plane Jet	Round Jet	Radial Jet	Wake
	Experiment[7, 78–81]	0.103-0.120	0.100-0.110	0.086-0.096	0.096-0.110	0.320-0.400
$k-\omega$	Wilcox[82*]	0.096	0.108	0.094	0.099	0.326
$k-\omega$	Wilcox[83*]	0.105	0.101	0.088	0.099	0.339
$k-\omega$	Wilcox[84*]	0.141	0.135	0.369	0.317	0.496
SST	Menter[85†]	0.100	0.112	0.127	-	0.257
$k-\epsilon$	Launder & Sharma[77*]	0.098	0.109	0.120	0.094	0.256
	with Pope[86*]	0.098	0.109	0.086	0.040	0.256
$k-\zeta$	Robinson, et al.[87*]	0.112	0.115	0.091	0.097	0.313
$k-\tau$	Speziale, et al.[88*]	0.082	0.089	0.102	0.073	0.221
SA	Spalart & Allmaras[76*]	0.109	0.157	0.248	0.166	0.341

Source: Yoder et al. NASA/TM 2013-218072

Simulations + Data

Validation

The "C" word

Statistical Inference

Machine Learning

Physics + Data

Principled Calibration: solve an optimization problem to find parameters

$$\min_{c_1, c_2, \ldots c_M} \| y - y_{data} \|^2$$

simulation → y y_{data} ← data

$c_1, c_2, \ldots c_M$

….assuming that data is noisy leads to Bayesian inversion

Simulations + Data

Validation

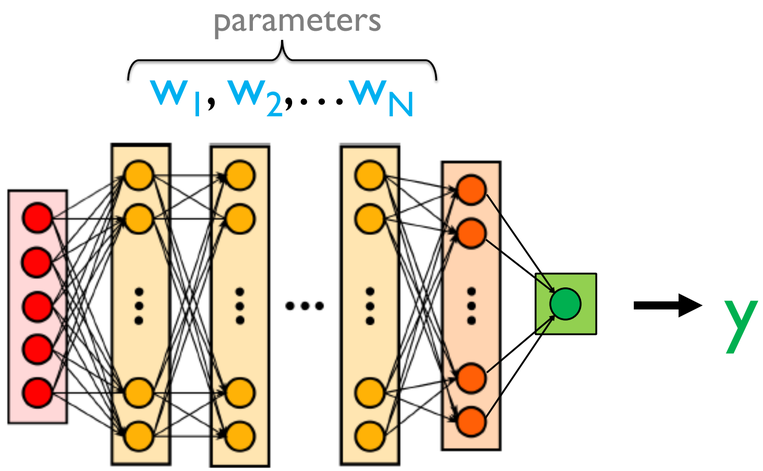
The "C" word

Statistical Inference

Machine Learning

Physics + Data

Deep Neural Network: a general input-output relationship



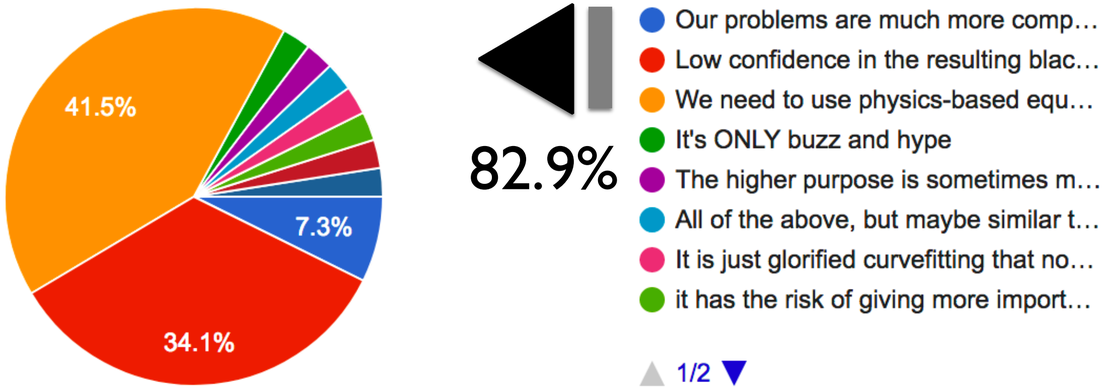
…solve an optimization problem to find the weights

$$\min_{w_1, w_2, \ldots w_N} || y - y_{data} ||^2$$

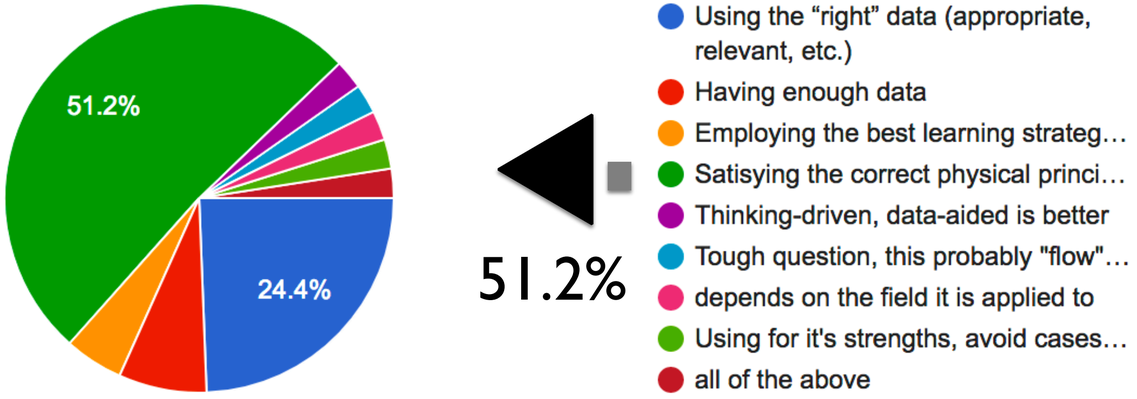
A Survey of ~80 Fluid Mechanics Researchers

2018 Stanford Center for Turbulence Research

What is your main criticism of data-driven predictions?



What do you think is the most important ingredient for data-driven predictions?



Simulations + Data

Validation

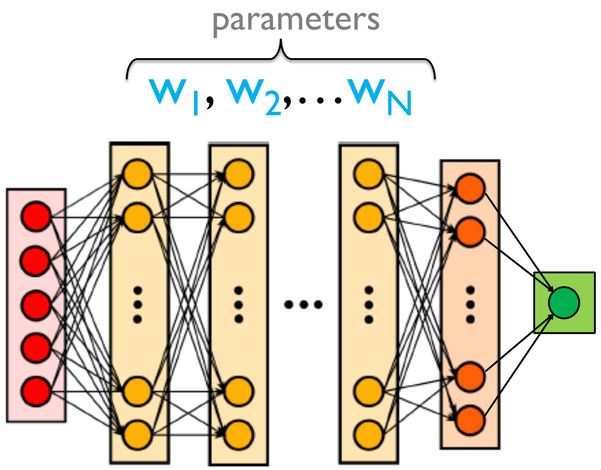
The "C" word

Statistical Inference

Machine Learning

Physics + Data

Deep Neural Network & Simulation



$y^* \leftarrow$ **simulation**

$c_1, c_2, \ldots c_M$

$\rightarrow y$

…solve an optimization problem to find the weights but constrain the acceptable outputs

$$\min_{w_1, w_2, \ldots w_N} || y - y_{data} ||^2 \quad \& \quad \min_{\substack{w_1, w_2, \ldots w_N \\ c_1, c_2, \ldots c_M}} || y - y^* ||^2$$

Simulations + Data

Validation

The “C” word

Statistical Inference

Machine Learning

Physics + Data

Deep Neural Network & Simulation

https://github.com/lululxvi/deepxde

Computer Science > Machine Learning

DeepXDE: A deep learning library for solving differential equations

Lu Lu, Xuhui Meng, Zhiping Mao, George E. Karniadakis

(Submitted on 10 Jul 2019)

Deep learning has achieved remarkable success in diverse applications; however, its use in solving partial differential equations (PDEs) has emerged only recently. Here, we present an overview of physics-informed neural networks (PINNs), which embed a PDE into the loss of the neural network using automatic differentiation. The PINN algorithm is simple, and it can be applied to different types of PDEs, including integro-differential equations, fractional PDEs, and stochastic PDEs. Moreover, PINNs solve inverse problems as easily as forward problems. We propose a new residual-based adaptive refinement (RAR) method to improve the training efficiency of PINNs. For pedagogical reasons, we compare the PINN algorithm to a standard finite element method. We also present a Python library for PINNs, DeepXDE, which is designed to serve both as an education tool to be used in the classroom as well as a research tool for solving problems in computational science and engineering. DeepXDE supports complex-geometry domains based on the technique of constructive solid geometry, and enables the user code to be compact, resembling closely the mathematical formulation. We introduce the usage of DeepXDE and its customizability, and we also demonstrate the capability of PINNs and the user-friendliness of DeepXDE for five different examples. More broadly, DeepXDE contributes to the more rapid development of the emerging Scientific Machine Learning field.

https://github.com/kailaix/ADCME.jl

Mathematics > Numerical Analysis

Predictive Modeling with Learned Constitutive Relations from Indirect Observations

Daniel Z. Huang, Kailai Xu, Charbel Farhat, Eric Darve

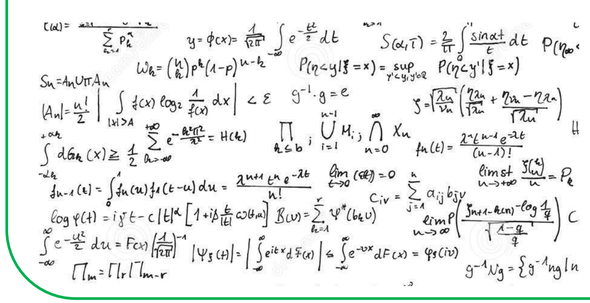
(Submitted on 29 May 2019 (v1), last revised 15 Oct 2019 (this version, v2))

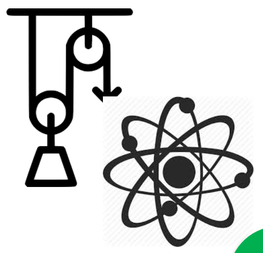
We present a new approach for predictive modeling and its uncertainty quantification for mechanical systems, where coarse-grained models such as constitutive relations are derived directly from observation data. We explore the use of neural networks to represent the unknowns functions (e.g., constitutive relations). Its counterparts, like piecewise linear functions and radial basis functions, are compared, and the strength of neural networks is explored. The training and predictive processes in this framework seamlessly combine the finite element method, automatic differentiation, and neural networks (or its counterparts). Under mild assumptions, we establish convergence guarantees. This framework also allows uncertainty quantification analysis in the form of intervals of confidence. Numerical examples on a multiscale fiber-reinforced plate problem and a nonlinear rubbery membrane problem from solid mechanics demonstrate the effectiveness of the framework.

Comments: 32 pages, 20 figures
Subjects: **Numerical Analysis (math.NA)**; Computational Physics (physics.comp-ph)

Simulations + Data

Physical Principles





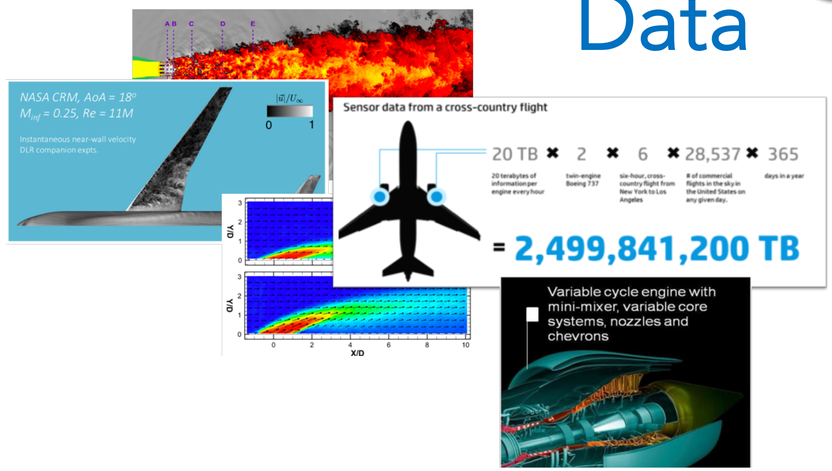
Software for Data

TensorFlow

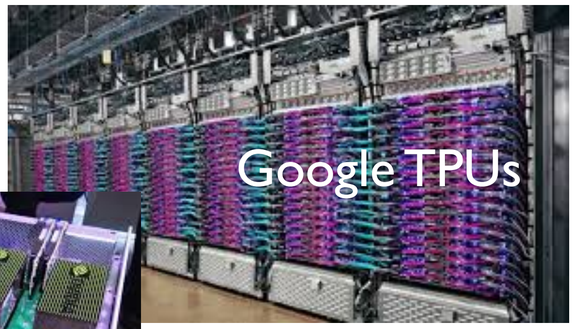
Keras

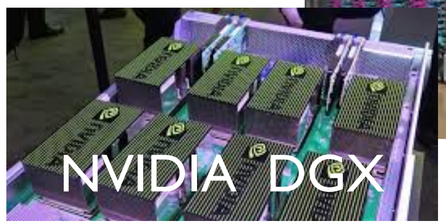
PYTORCH

Data



Computing for Data





Outline

1. Blind Experiment >> ML-driven Predictions
2. Physics First >> ML-driven Modeling
3. Reset >> ML-driven Numerical Methods

Outline

1. Blind Experiment >> ML-driven Predictions
2. Physics First >> ML-driven Modeling
3. Reset >> ML-driven Numerical Methods

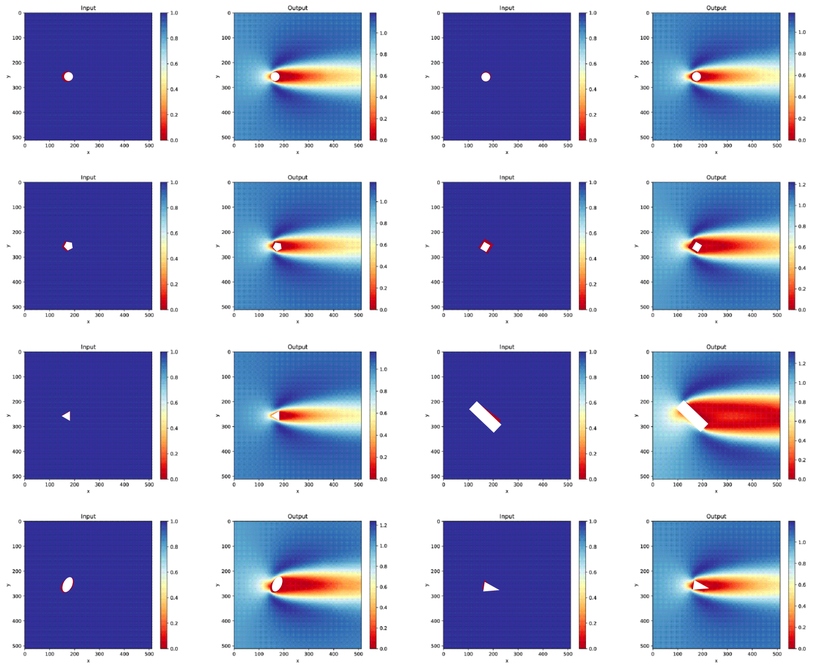
ML@Google: Synthetic Faces (deepfakes)

Increasingly realistic synthetic faces generated by ML@Google Goodfellow et al (2014), Radford et al. (2015), Liu and Tuzel (2016), Karras et al. (2017), Karras et al. (2018)

Synthetic Flow Simulations: flowfakes

Data

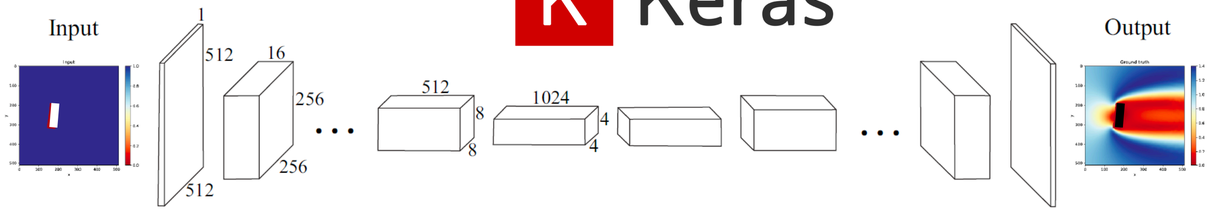
2D Geometries and Velocity Maps



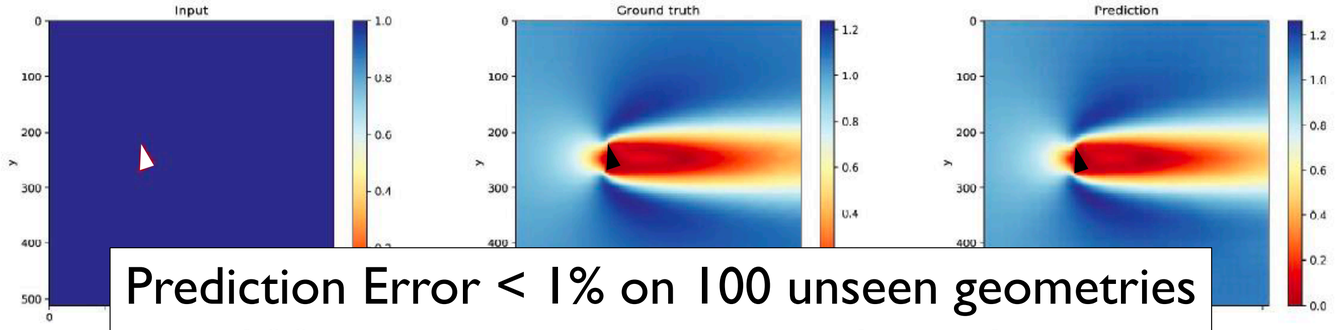
1000s of Fluent Computations

Machine Learning Training (convolutional autoencoders)

Keras



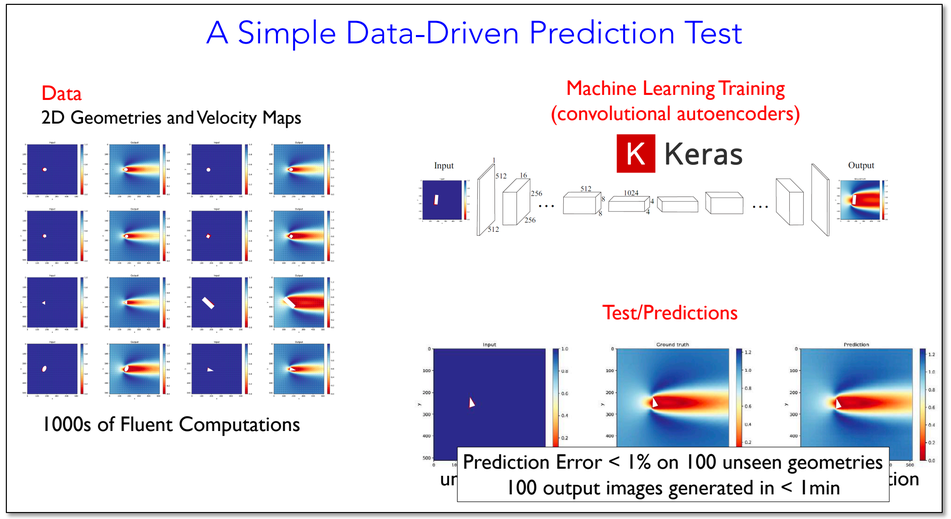
Test/Predictions



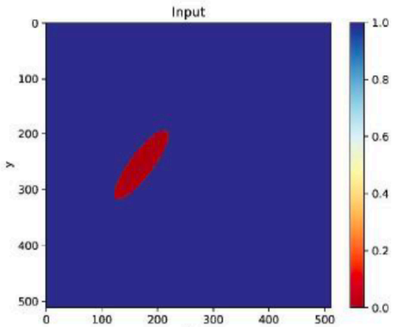
Prediction Error < 1% on 100 unseen geometries
100 output images generated in < 1min

Work with Ali Kashefi @ Stanford

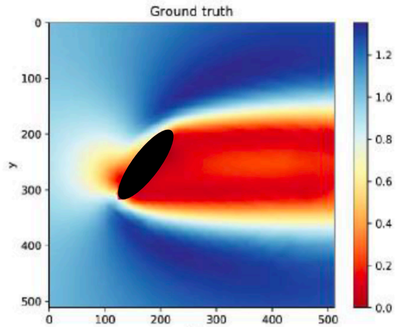
Synthetic Flow Simulations: flowfakes



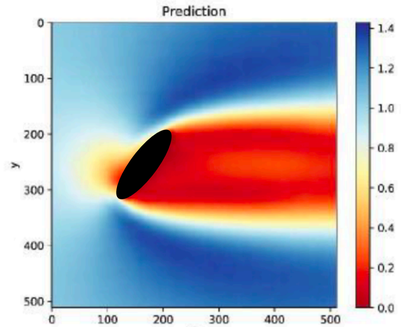
- Building the Dataset is expensive and requires extensive and robust automation
- Training the network is expensive and requires knowledge and expertise in selecting options
- Predictions accuracy is measured in average… in some cases the (local) error is as high as 5%



unseen input



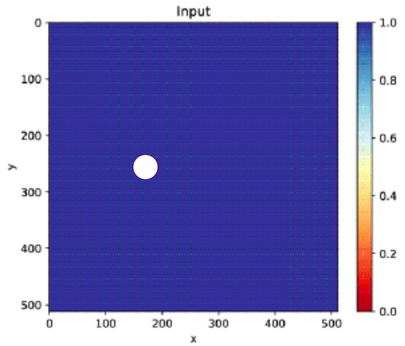
predicted output

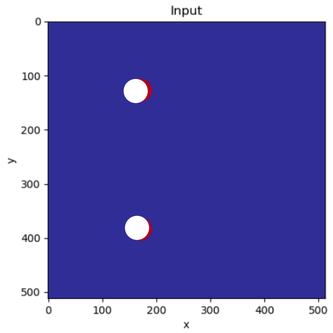


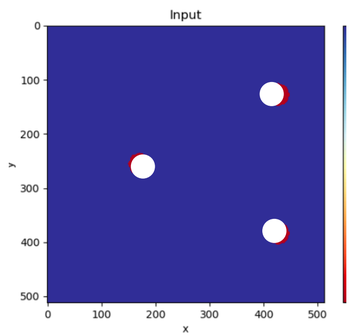
true solution

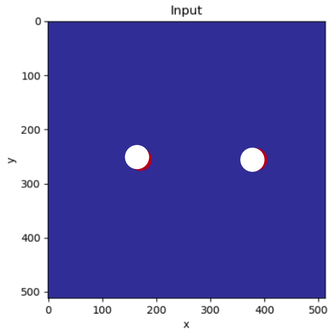
Synthetic Flow Simulations: flowfakes

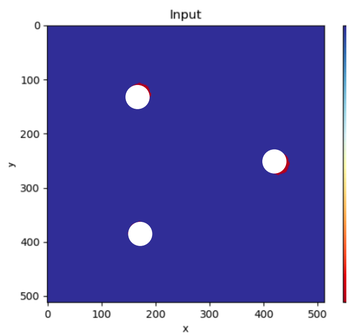
What is the accuracy expectation?





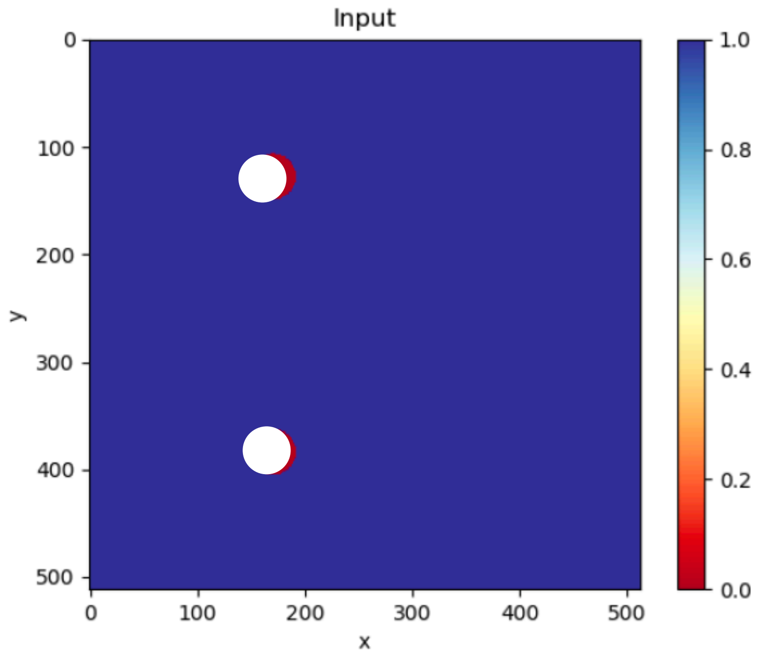




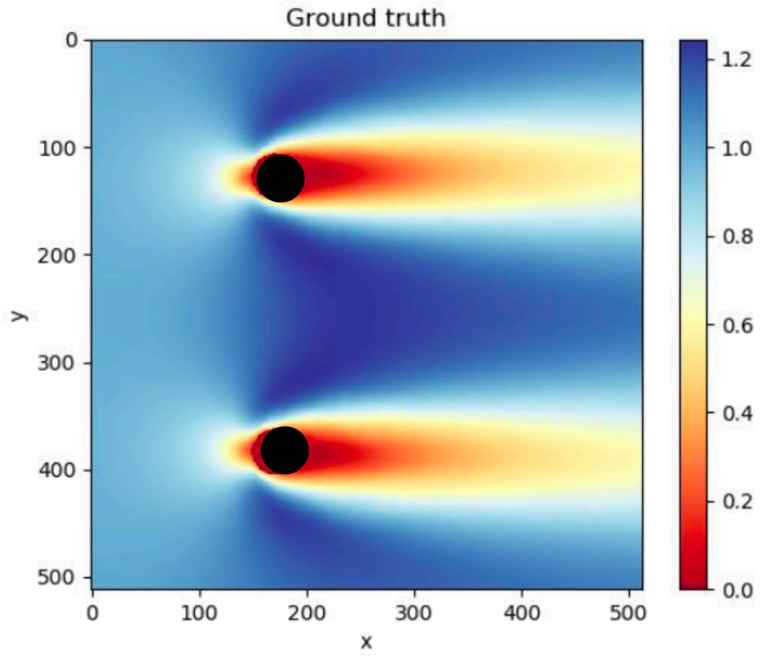


Synthetic Flow Simulations: flowfakes

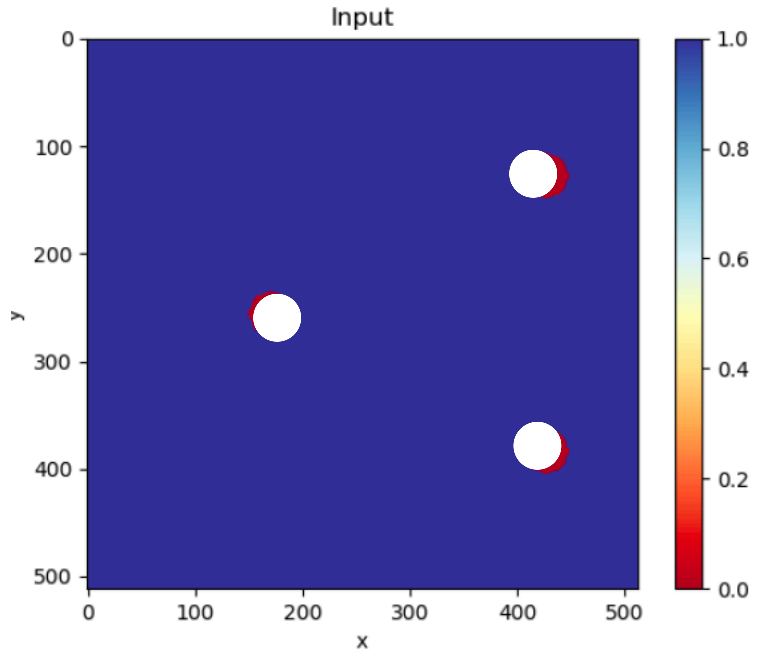
What is the accuracy expectation?



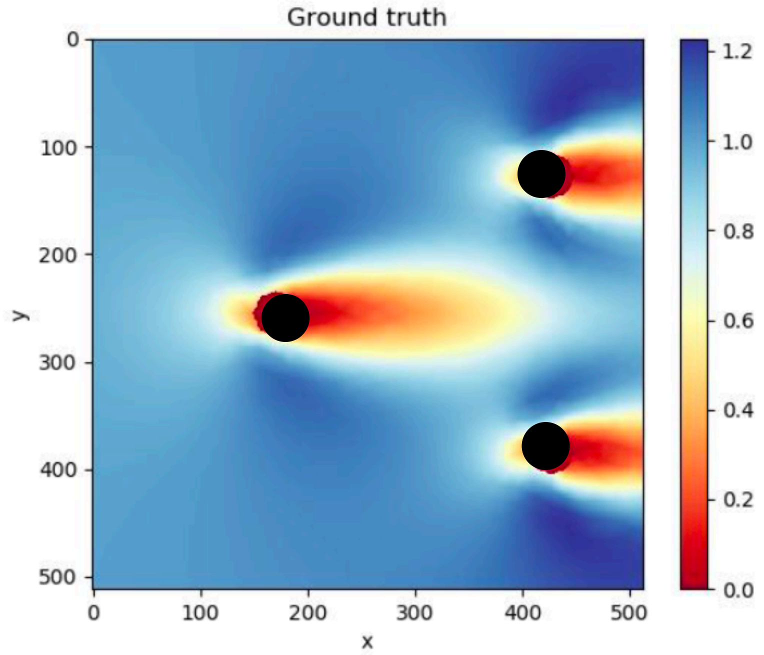
input



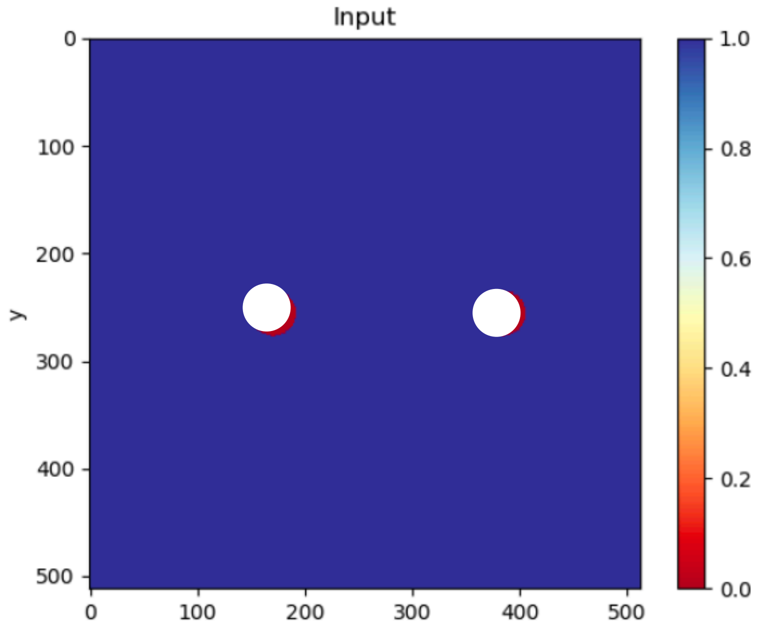
predicted output



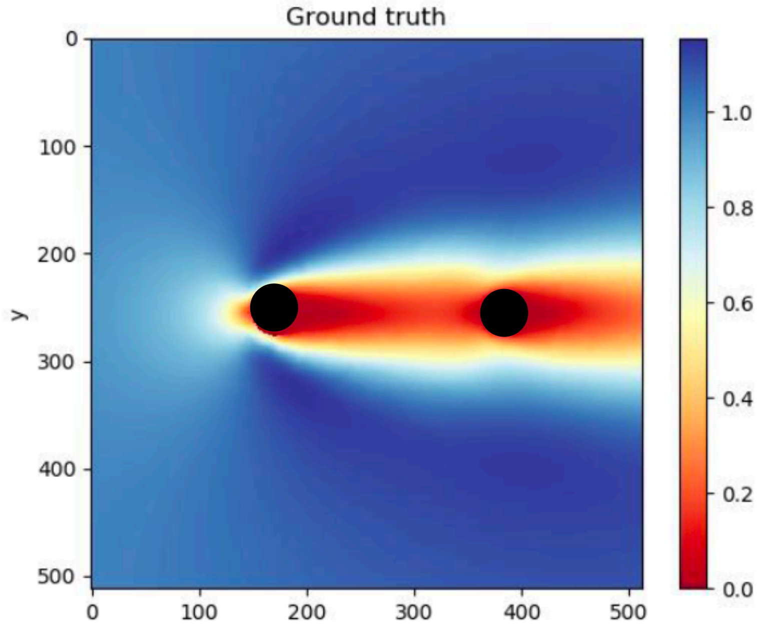
input



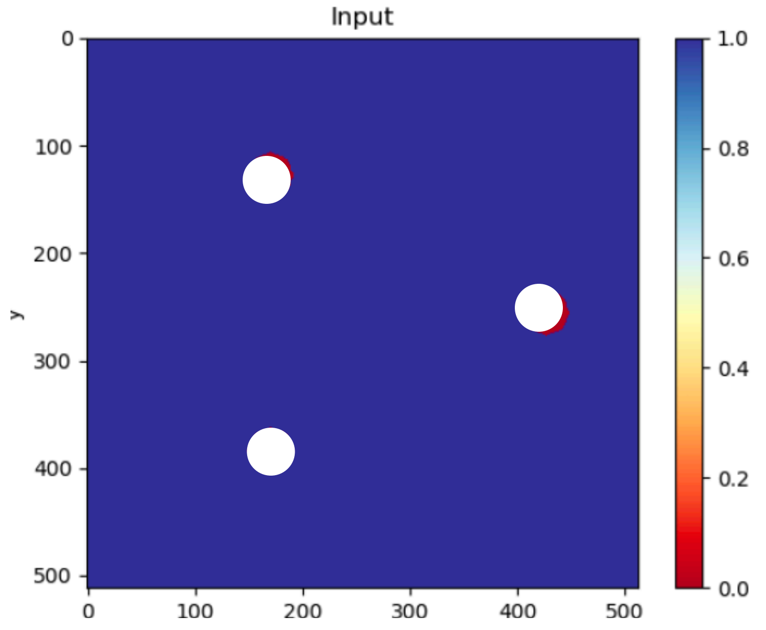
predicted output



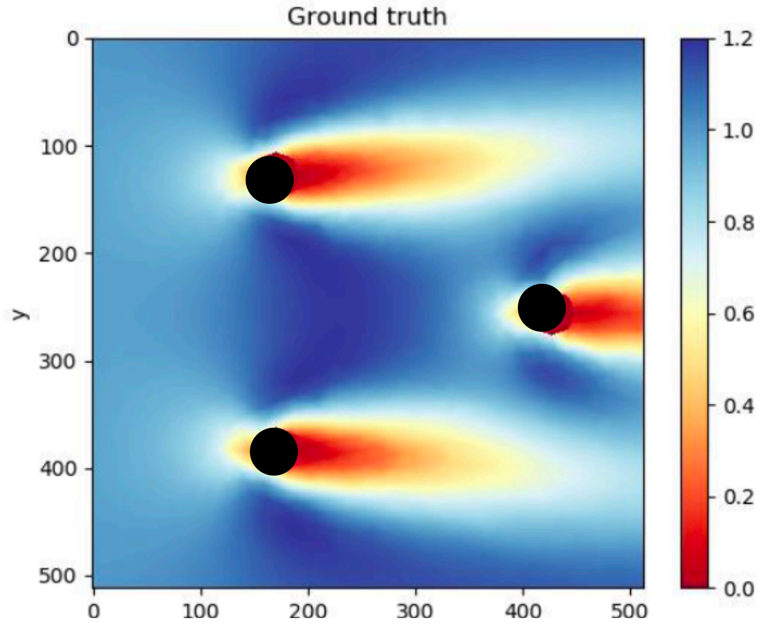
input



predicted output



input



predicted output

Synthetic Flow Simulations: flowfakes

What is the accuracy expectation? …difficult to say

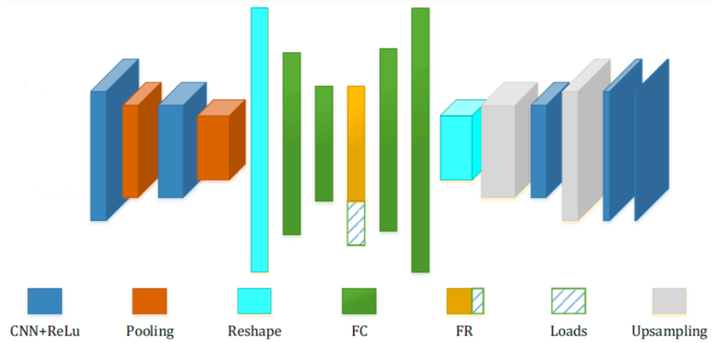
input | predicted output | true solution | input | predicted output | true solution

input | predicted output | true solution | input | predicted output | true solution

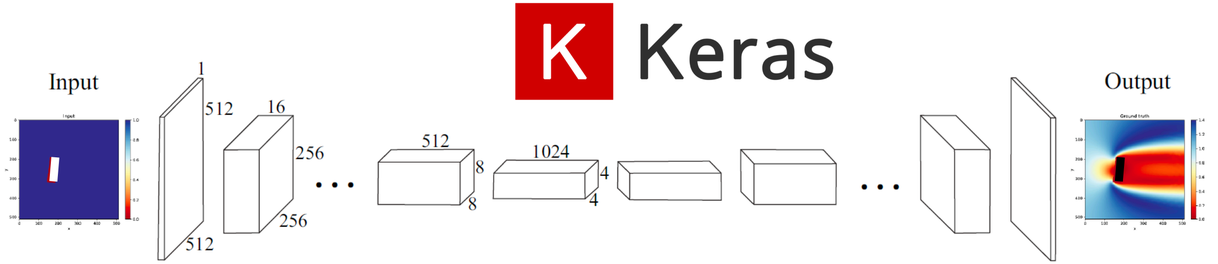
Synthetic Flow Simulations: flowfakes

CNN for Linear Stress Predictions

Nie, Jiang, Kara @ Carnegie Mellon, 2018



Machine Learning Training (convolutional autoencoders)



Some Details of the present DNN

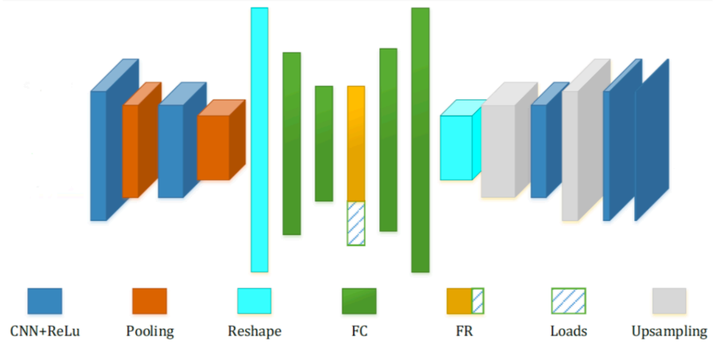
- ~7000 solutions in dataset
- 39,158,080 hyper-parameters
- 1000 Epochs
- No fully connected layers
- No Max-pooling/Upsampling
- CNN with Leaky ReLU
- DCNN with ReLU

Layer#1	= Input
Layer#2	C($f=16, k=4, s=2$), Leaky ReLU
Layer#3	C($f=32, k=4, s=2$), Leaky ReLU, Batch Normalization
Layer#4	C($f=64, k=4, s=2$), Leaky ReLU, Batch Normalization
Layer#5	C($f=128, k=4, s=2$), Leaky ReLU, Batch Normalization
Layer#6	C($f=256, k=4, s=2$), Leaky ReLU, Batch Normalization
Layer#7	C($f=512, k=4, s=2$), Leaky ReLU, Batch Normalization
Layer#8	C($f=1024, k=4, s=2$), Leaky ReLU, Batch Normalization
Layer#9	CT($f=1024, k=4, s=1$), ReLU, Batch Normalization
Layer#10	CT($f=512, k=4, s=2$), ReLU, Batch Normalization
Layer#11	CT($f=256, k=4, s=2$), ReLU, Batch Normalization
Layer#12	CT($f=128, k=4, s=2$), ReLU, Batch Normalization
Layer#13	CT($f=64, k=4, s=2$), ReLU, Batch Normalization
Layer#14	CT($f=32, k=4, s=2$), ReLU, Batch Normalization
Layer#15	CT($f=16, k=4, s=2$), ReLU, Batch Normalization
Layer#16	Output = CT($f=1, k=4, s=2$), ReLU

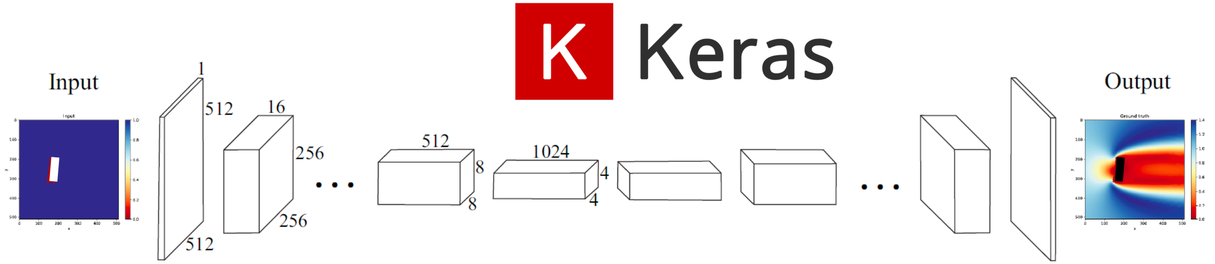
Synthetic Flow Simulations: flowfakes

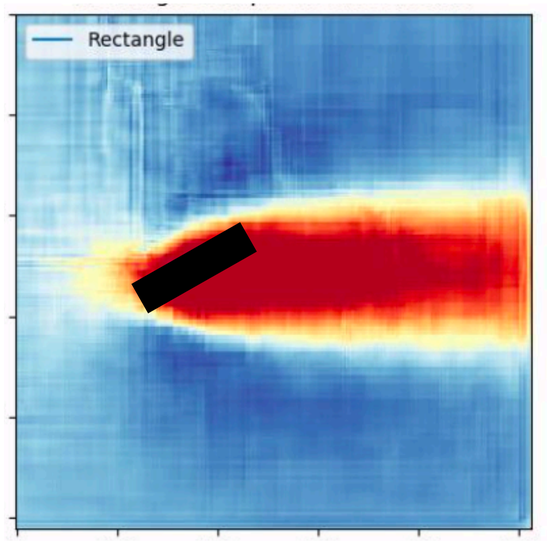
CNN for Linear Stress Predictions

Nie, Jiang, Kara @ Carnegie Mellon, 2018

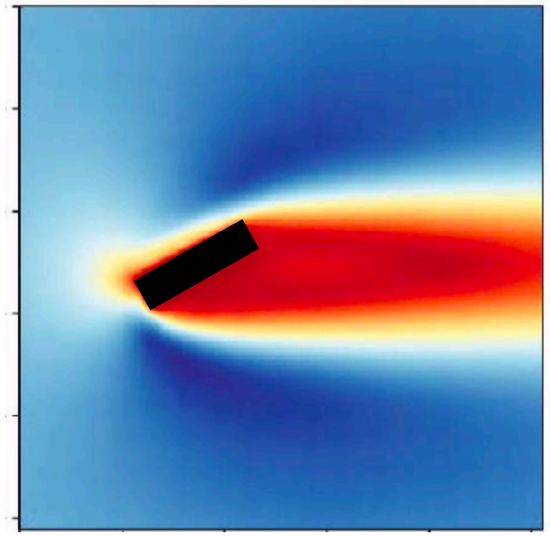


Machine Learning Training (convolutional autoencoders)



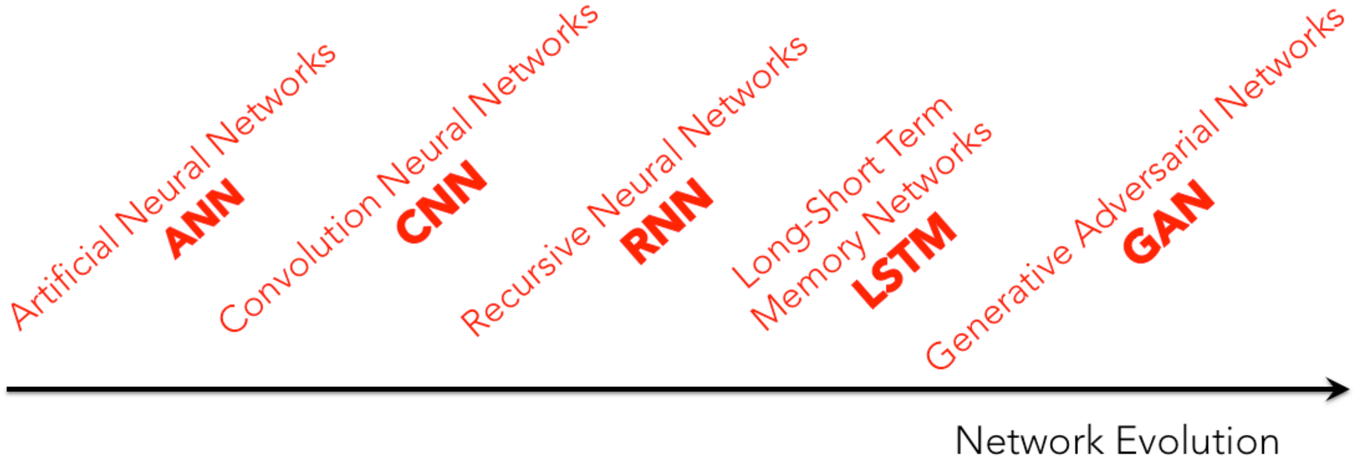


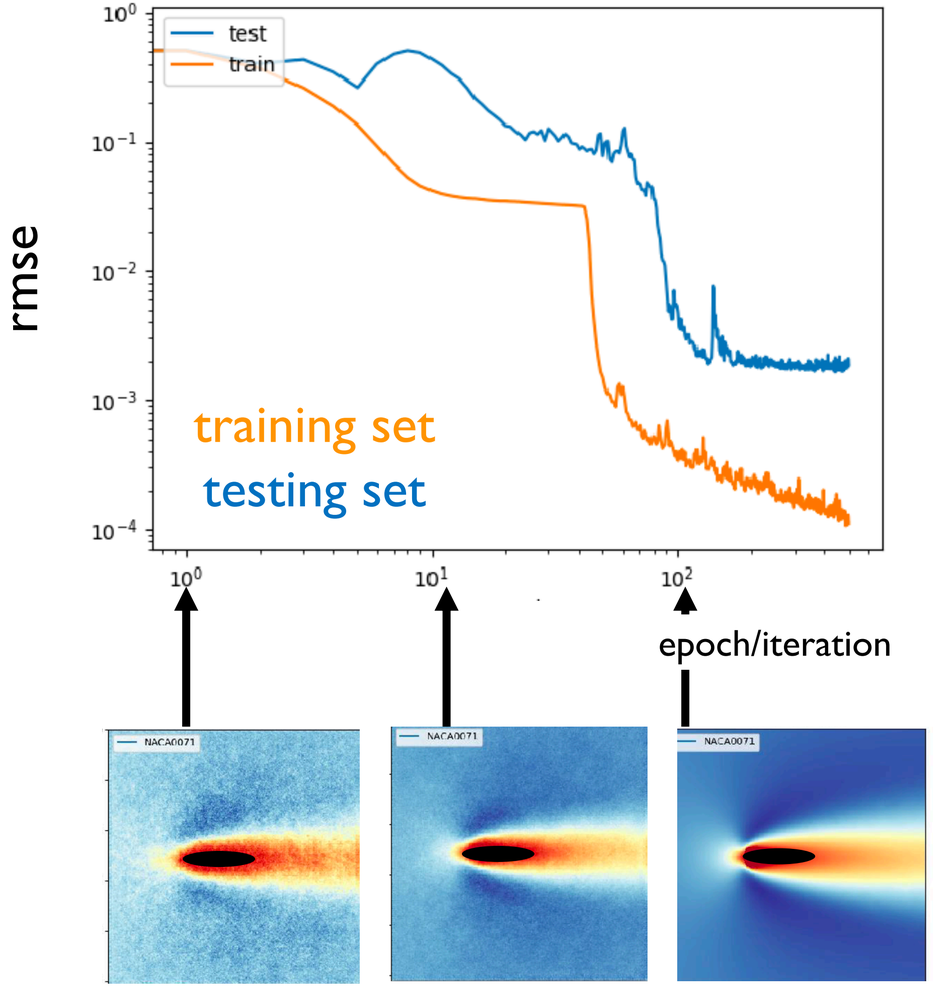
@ fixed training cost



Pacing Items

99% of time to construct the dataset and train the network





The choice of network architecture has strong influence on the success metric

Outline

1. Blind Experiment >> ML-driven Predictions
2. **Physics First >> ML-driven Modeling**
3. Reset >> ML-driven Numerical Methods

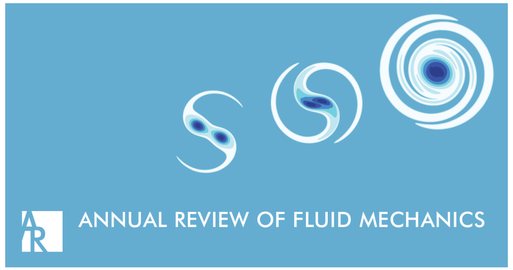
Turbulence Modeling in the Age of Data

Annual Review of Fluid Mechanics

Vol. 51:357-377 (Volume publication date January 2019)
First published as a Review in Advance on September 19, 2018
https://doi.org/10.1146/annurev-fluid-010518-040547

Karthik Duraisamy,[1,*] Gianluca Iaccarino,[2,*] and Heng Xiao[3,*]

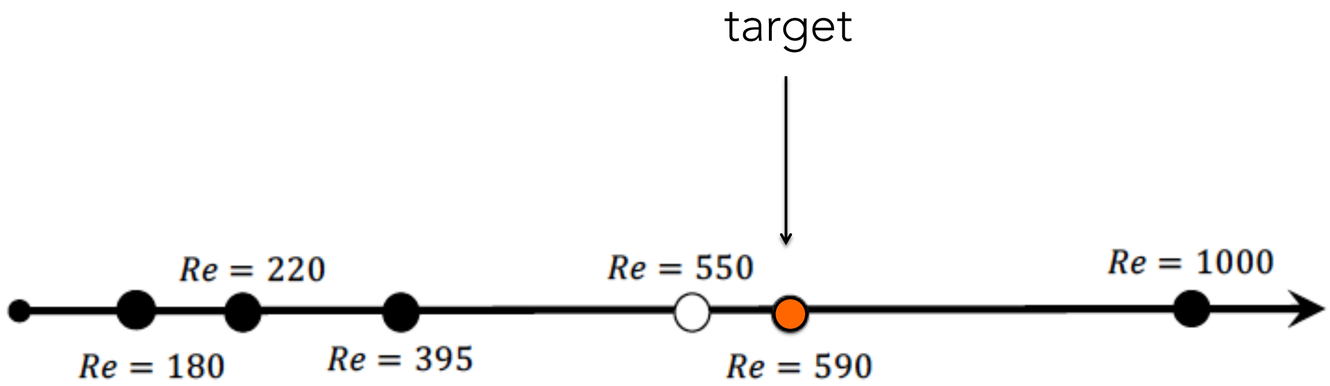
[1]Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109, USA; email: kdur@umich.edu
[2]Department of Mechanical Engineering, Stanford University, Stanford, California 94305, USA; email: jops@stanford.edu
[3]Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, Virginia 24060, USA; email: hengxiao@vt.edu



Random Forests and Turbulence Models

- Goal: learn a correction for the turbulence kinetic energy computed by the k-ε model @ Re_τ=590

$$k_{DNS} = k_{k\text{-}\varepsilon} + \delta k(\eta)$$



- Data: DNS of channel flows from J. Jimenez et al.

- Decisions: dictionary of **28 known turbulence features**
  - Turbulent Reynolds number (based on k)
  - Time scale ratio
  - Wall distance
  - Ratio of rotation/strain
  - Ratio of production and dissipation
  - […]

- Forest: collection of **25** decision trees
  - Max Depth: 10
  - Max Features: 6 (with repetition)
  - Decision split based on variance reduction
  - Training dataset: Re_τ= 180, 220,395,1000
  - Test dataset: Re_τ=590

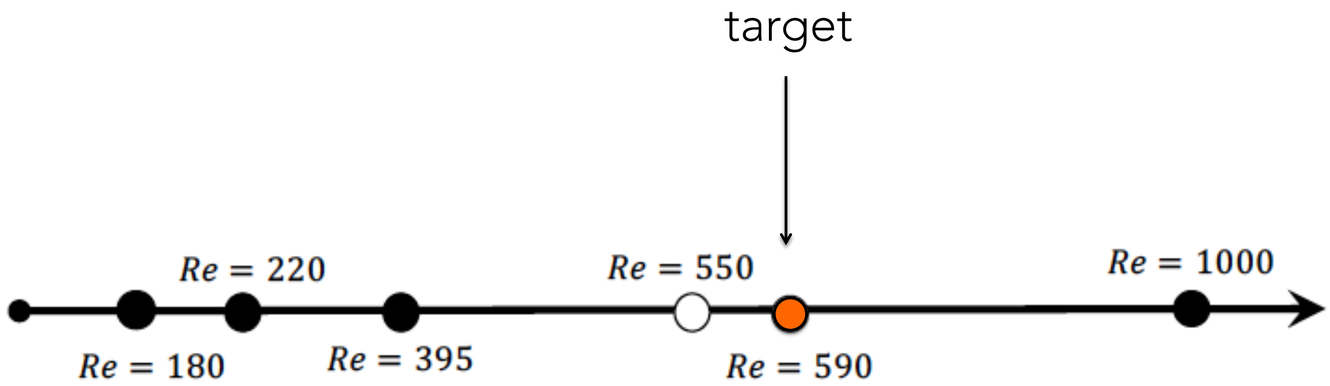
Work with Aashwin Mishra @ Stanford

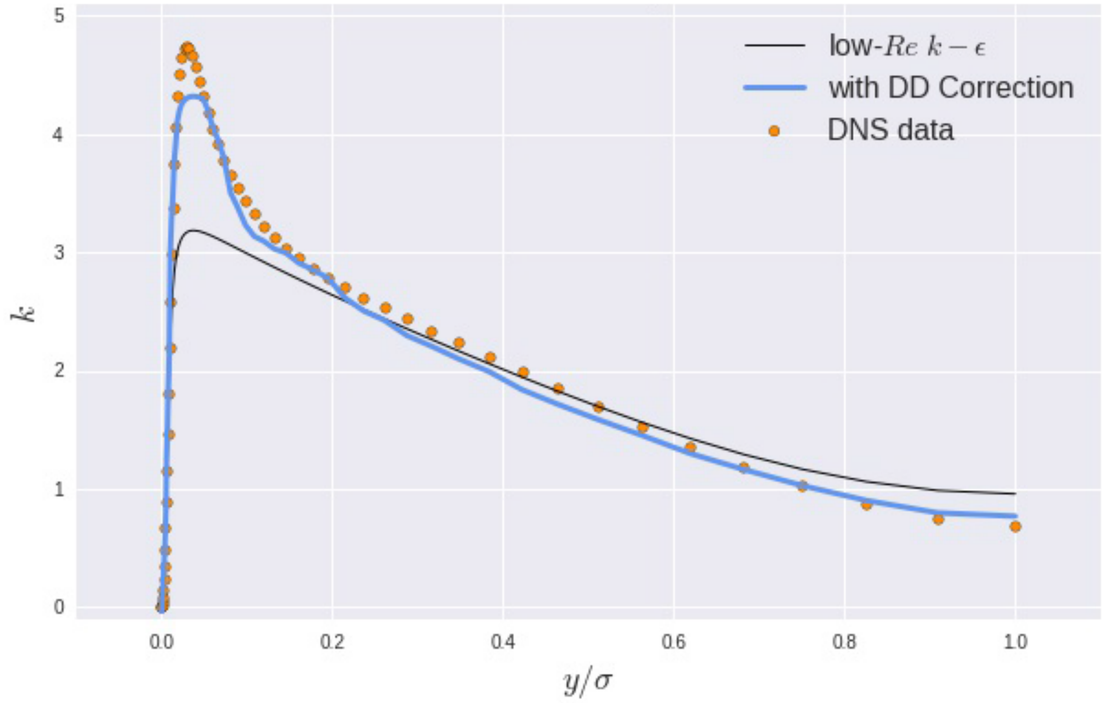
Random Forests and Turbulence Models

- Goal: learn a correction for the turbulence kinetic energy computed by the k-ε model @ Re_τ=590

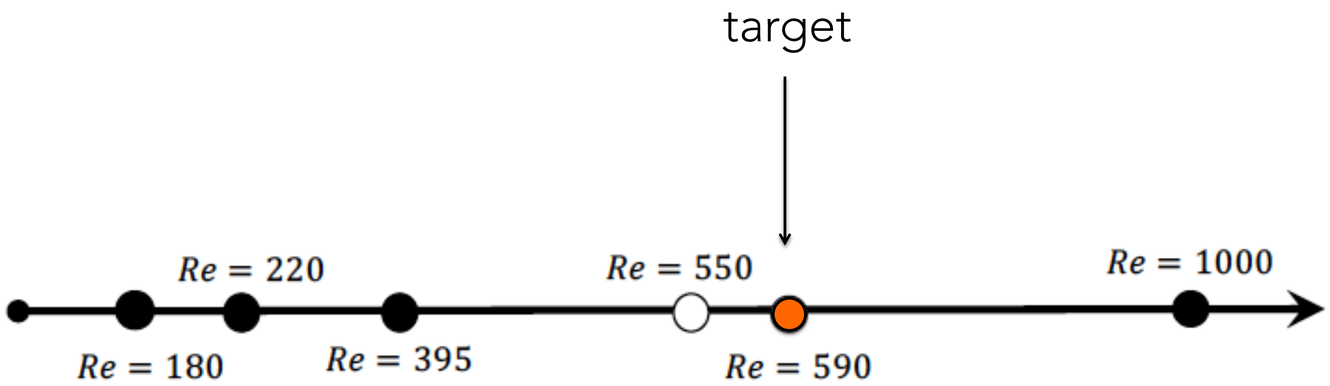
$$k_{DNS} = k_{k\text{-}\varepsilon} + \delta k(\eta)$$

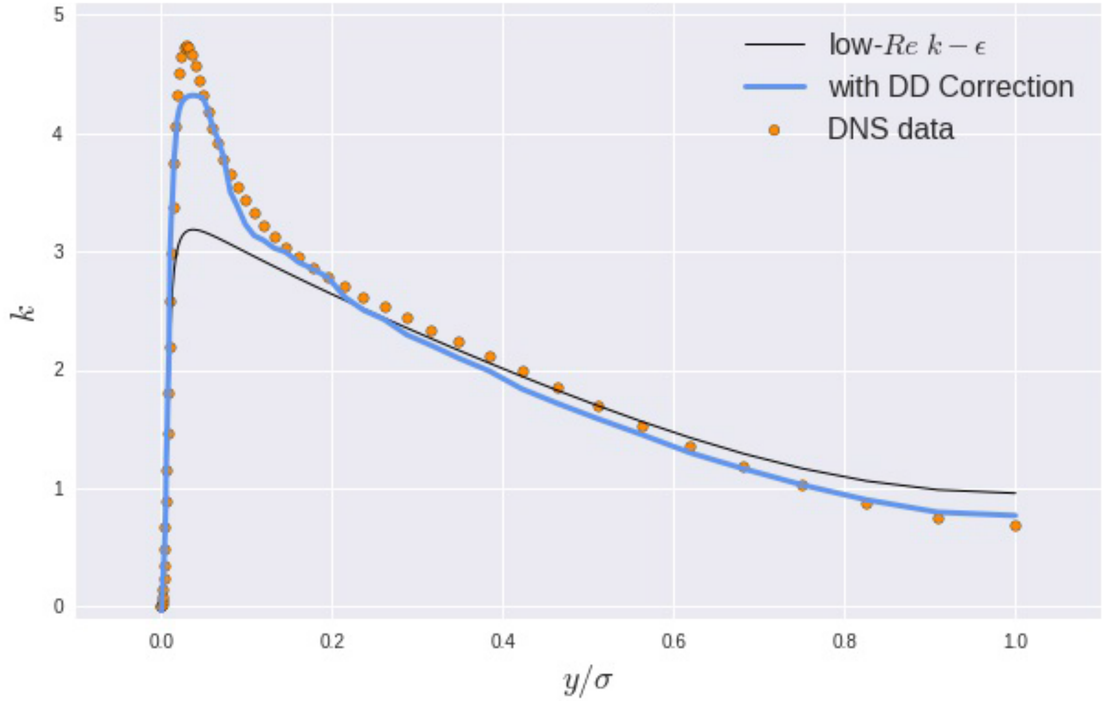
- Data: DNS of channel flows from J. Jimenez et al.

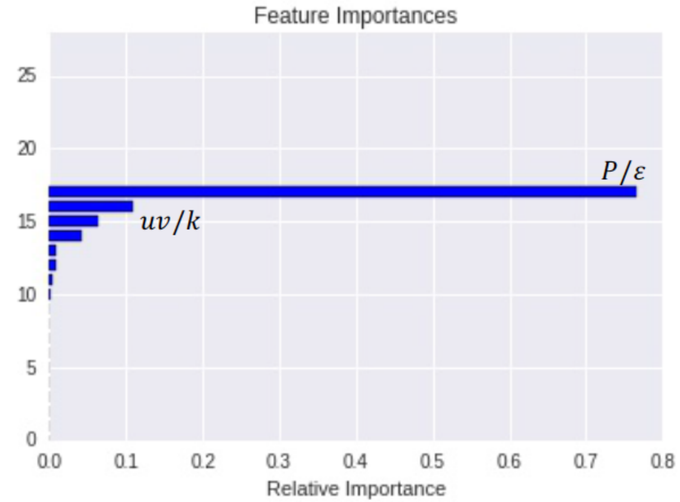




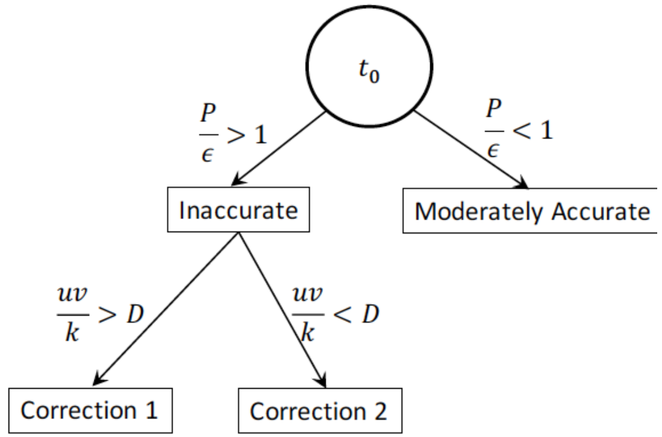
Random Forests and Turbulence Models







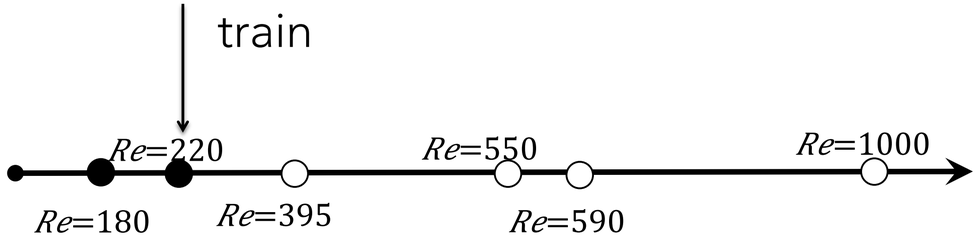
Data-driven rationale & insight

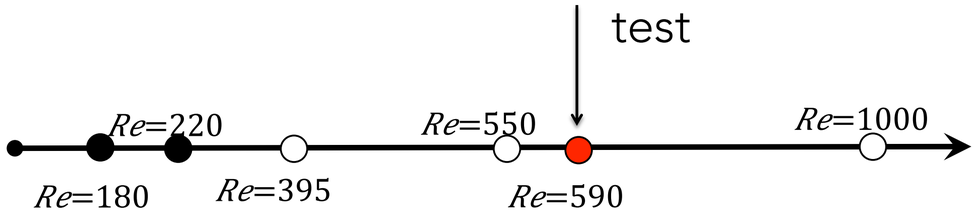


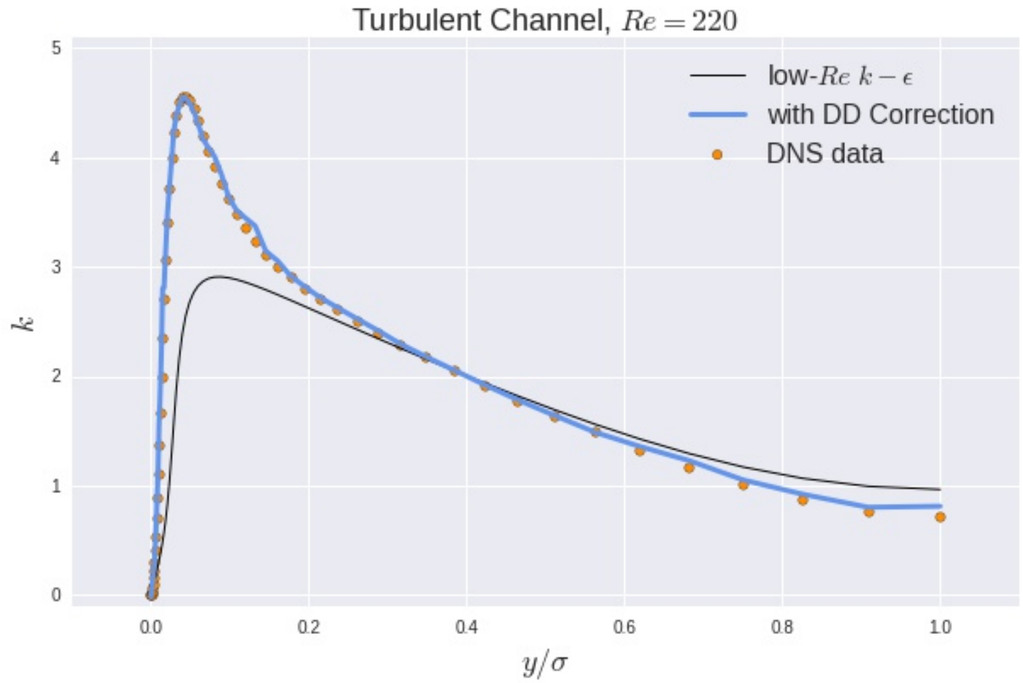
Data Driven Rationale

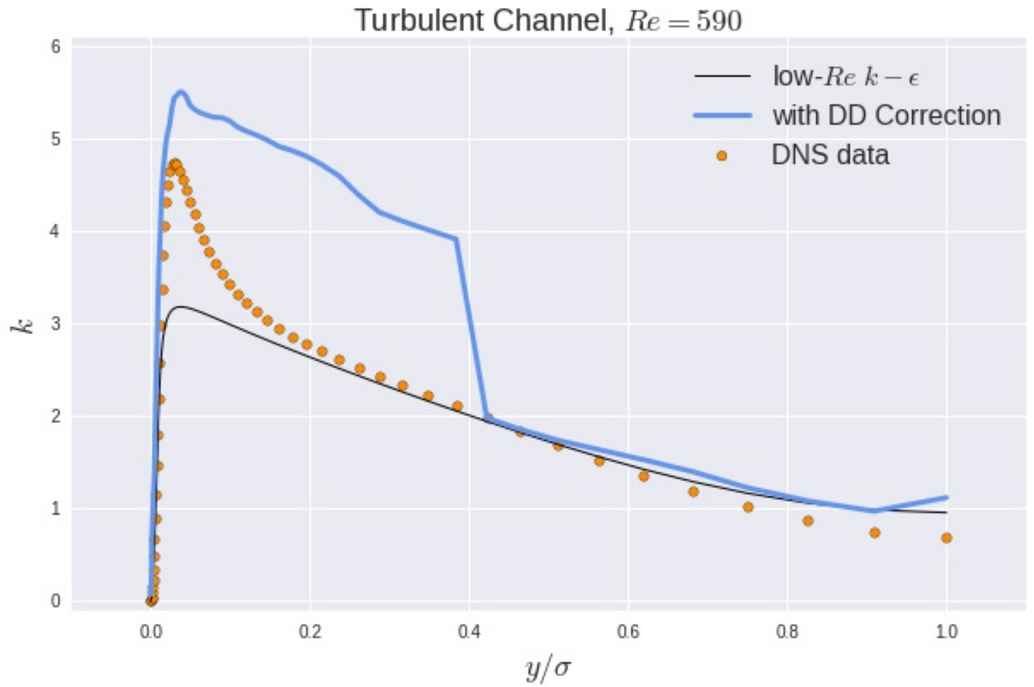
Random Forests and Turbulence Models

Insufficient size of the dataset: overfitting
Inappropriate choice of dataset range: extrapolation









Outline

1. Blind Experiment >> ML-driven Predictions
2. Physics First >> ML-driven Modeling
3. Reset >> ML-driven Numerical Methods

Pacing Items

Which one is the right ML (data-driven) strategy?

What (and how much) data is really necessary?

How can we systematically improve the prediction of ML algorithms?

How to construct network architectures that embed prior knowledge (e.g. physical principles) rather than enforcing them as constraints in the optimization problem?

(Machine) Learning to Differentiate

Oskar Alund, Jan Nordstrom
Department of Mathematics
Linköping University

oskar.alund@liu.se jan.norstrom@liu.se

Gianluca Iaccarino
Mechanical Engineering Department &
Institute for Computational Mathematical Engineering
Stanford University

jops@stanford.edu

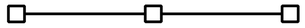
NSF Workshop on Exuberance of Machine Learning in Transport Phenomena
February 10 — 11, 2020
Dallas, TX

(Machine) Learning to Differentiate

Can we build an ML-based strategy to construct a numerical method for solving Partial Differential Equations?

Differentiate Polynomials

Consider a 3-points mesh $\Omega_h = \{-1, 0, 1\}$



and polynomials evaluated on the mesh points

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top, \quad \mathbf{x} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^\top, \quad \mathbf{x}^2 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^\top$$

We can write the differentiation process as a matrix operation

$$\mathbf{D1} = \mathbf{0} \quad \mathbf{Dx} = \mathbf{1} \quad \mathbf{Dx}^2 = 2\mathbf{x}$$

Leading to:

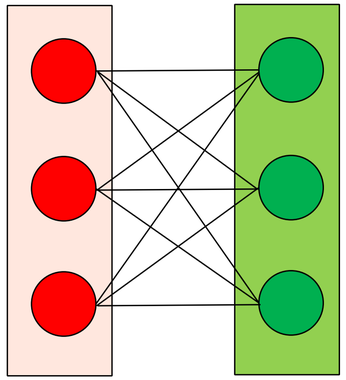
$$\mathbf{D} \begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -2 \\ 0 & 1 & 0 \\ 0 & 1 & 2 \end{bmatrix} \implies \mathbf{D} = \begin{bmatrix} -1.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 \\ 0.5 & -2 & 1.5 \end{bmatrix}$$

ML to Differentiate Polynomials

The simplest network that can accomplish the same task is:

$\mathbf{p}_i$

function values at nodes



$\mathbf{p}'_i$

derivative at nodes

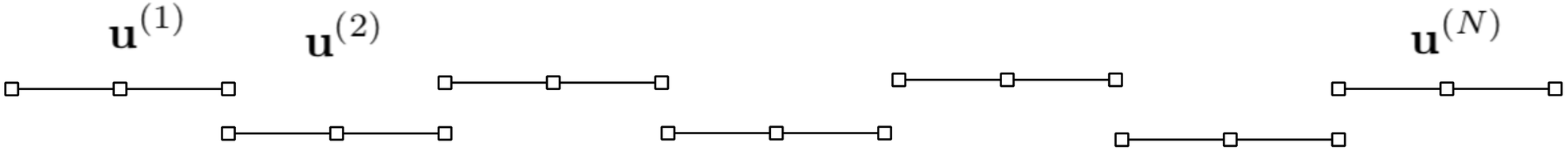
with weights $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{3\times 3}$ obtained minimizing $\sum_{i=0}^{2} \|\mathbf{p}'_i - \mathbf{W}\mathbf{p}_i\|_2^2$

If we assume linear activation functions and no-bias $\mathbf{W} \implies \mathbf{D}$

A Numerical Technique for PDEs

Consider linear transport $u_t + u_x = 0, \quad x \in (0,1), \quad t > 0,$

and a mesh with 3-point stencil in each mesh element (i.e. a nodal DG scheme)



$$\mathbf{u}_t^{(1)} + \mathbf{D}^{(1)}\mathbf{u}^{(1)} = \sigma_1(u_0^{(1)} - g)\mathbf{e}_0$$
$$\mathbf{u}_t^{(2)} + \mathbf{D}^{(2)}\mathbf{u}^{(2)} = \sigma_2(u_0^{(2)} - u_2^{(1)})\mathbf{e}_0$$
$$\vdots$$
$$\mathbf{u}_t^{(N)} + \mathbf{D}^{(N)}\mathbf{u}^{(N)} = \sigma_N(u_0^{(N)} - u_2^{(N-1)})\mathbf{e}_0$$

the RHS corresponds to coupling terms between elements and the boundary conditions with coefficient chosen to achieve stability

A Stable Numerical Technique for PDEs

if $\mathbf{D} = \mathbf{P}^{-1}\mathbf{Q}$

with $\mathbf{1}^\top \mathbf{P}\mathbf{x}^k = \int_{-1}^{1} x^k dx$ and $\mathbf{Q} + \mathbf{Q}^\top = \mathrm{diag}(-1, 0, 1)$

the discretization satisfies the summation-by-part property (SBP) and the scheme is provably stable!

H. O. Kreiss and J. Oliger, Comparison of accurate methods for the integration of hyperbolic equations, Tellus, 24 (1972), pp. 199-215.

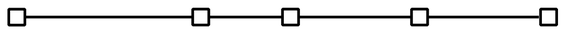
We can train 2 networks, one for $\mathbf{P}$ and one for $\mathbf{Q}$ and build $\mathbf{D}$ combining weights

A Stable Numerical Technique for PDEs

We can train 2 networks, one for $\mathbf{P}$ and one for $\mathbf{Q}$ and build $\mathbf{D}$ combining weights

Remarks:

- We can use any function to train the networks for $\mathbf{P}$ and $\mathbf{Q}$ as long as we can compute integrals (for $\mathbf{P}$) and derivatives (for $\mathbf{Q}$)
- Enforcing $\mathbf{Q} + \mathbf{Q}^\top = \mathrm{diag}(-1, 0, 1)$ is not straightforward we can use a constraint on the loss function or design a network that automatically enforces a constraint on the weights
- We can use any point-distribution within each mesh element and achieve high-order accuracy, e.g. $\Omega_h = \{-1, -0.3, 0, 0.5, 1\}$



A Stable Numerical Technique for PDEs

learning_to_diff.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Reconnect

Learning to Differentiate

In this notebook we show the basic procedure of constructing SBP operators using Tensorflow/Keras.

Let us first import a few necessary libraries.

```
[ ]  import numpy as np
     import tensorflow as tf
     import numpy.polynomial.polynomial as poly
     import matplotlib.pyplot as plt
     tf.keras.backend.set_floatx('float64')
     np.set_printoptions(precision=4,suppress=True)

     num_copies=20000
     noise_variance=0.0
```

Training the quadrature.

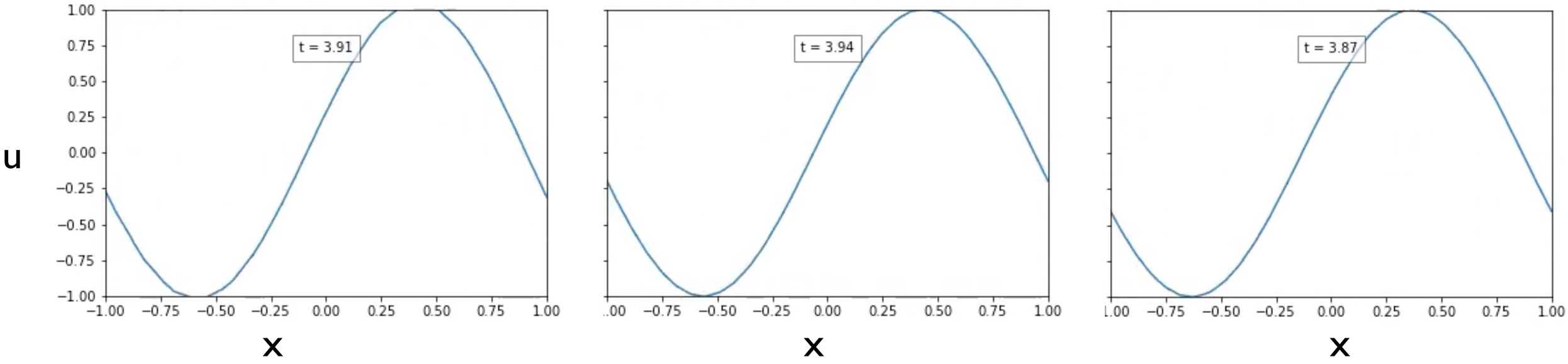
(Machine) Learning to Differentiate

Discretization for Linear Advection
N=15 Mesh Elements

No Constraints

SBP Constraint via Loss Function

SBP Constraint via Network Structure



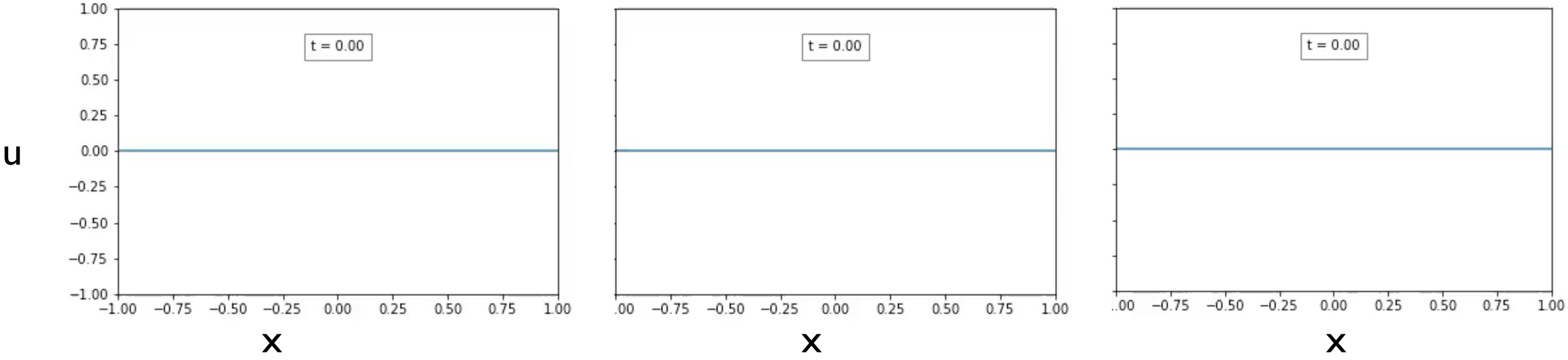
(Machine) Learning to Differentiate

Discretization for Linear Advection
N=50 Mesh Elements

No Constraints

SBP Constraint via Loss Function

SBP Constraint via Network Structure



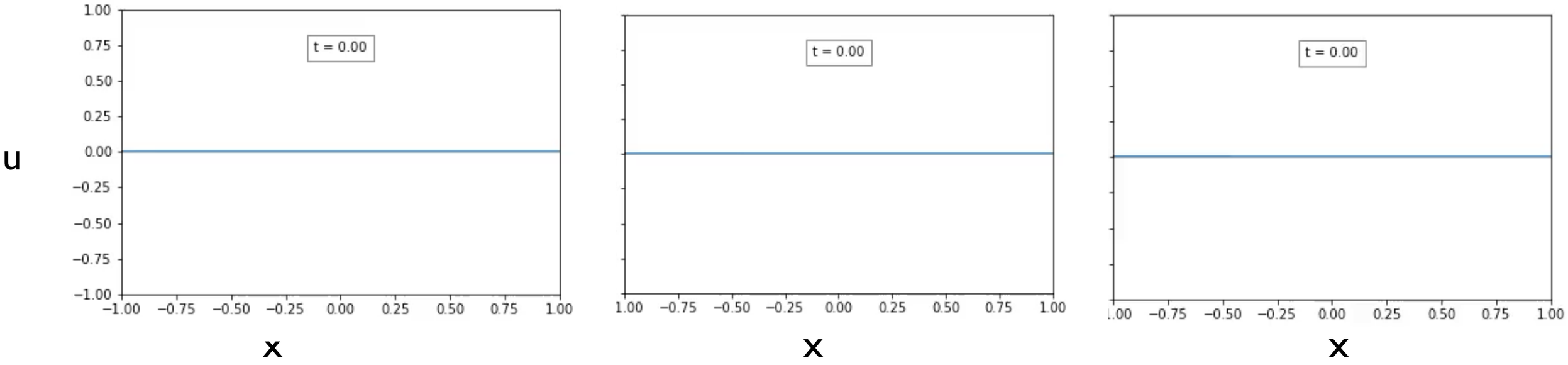
(Machine) Learning to Differentiate

Discretization for Linear Advection - training with noisy functions
N=15 Mesh Elements

No Constraints

SBP Constraint via Loss Function

SBP Constraint via Network Structure



(Machine) Learning to Differentiate

Discretization for Linear Advection

No Constraints

unstable

SBP Constraint via Loss Function

Regularization		
N	error	rate
10	4.3548e-02	-
20	4.4266e-03	3.30
40	8.5910e-04	2.37
80	2.0676e-04	2.05
160	4.4616e-05	2.21

SBP Constraint via Network Structure

Weight restriction		
N	error	rate
10	3.7098e-02	-
20	3.4367e-03	3.43
40	3.8367e-04	3.16
80	4.6386e-05	3.05
160	5.7488e-06	3.01

A Numerical Technique for PDEs

Consider non-linear transport $u_t + uu_x = 0, \quad x \in (-1, 1), \quad t > 0$

building $\mathbf{D} = \mathbf{P}^{-1}\mathbf{Q}$ using ML we cannot ensure stability.!

However we can use the transformation $u_t + \frac{1}{3}(uu_x + (u^2)_x) = 0,$

and the operator $\boldsymbol{\mathcal{L}}(\mathbf{u}) = \frac{1}{3}\mathbf{P}^{-1}\boldsymbol{\mathcal{Q}}(\mathbf{u})$ with $\mathbf{U} = \mathrm{diag}(\mathbf{u})$ and $\boldsymbol{\mathcal{Q}}(\mathbf{u}) = \mathbf{UQ} + \mathbf{QU}$

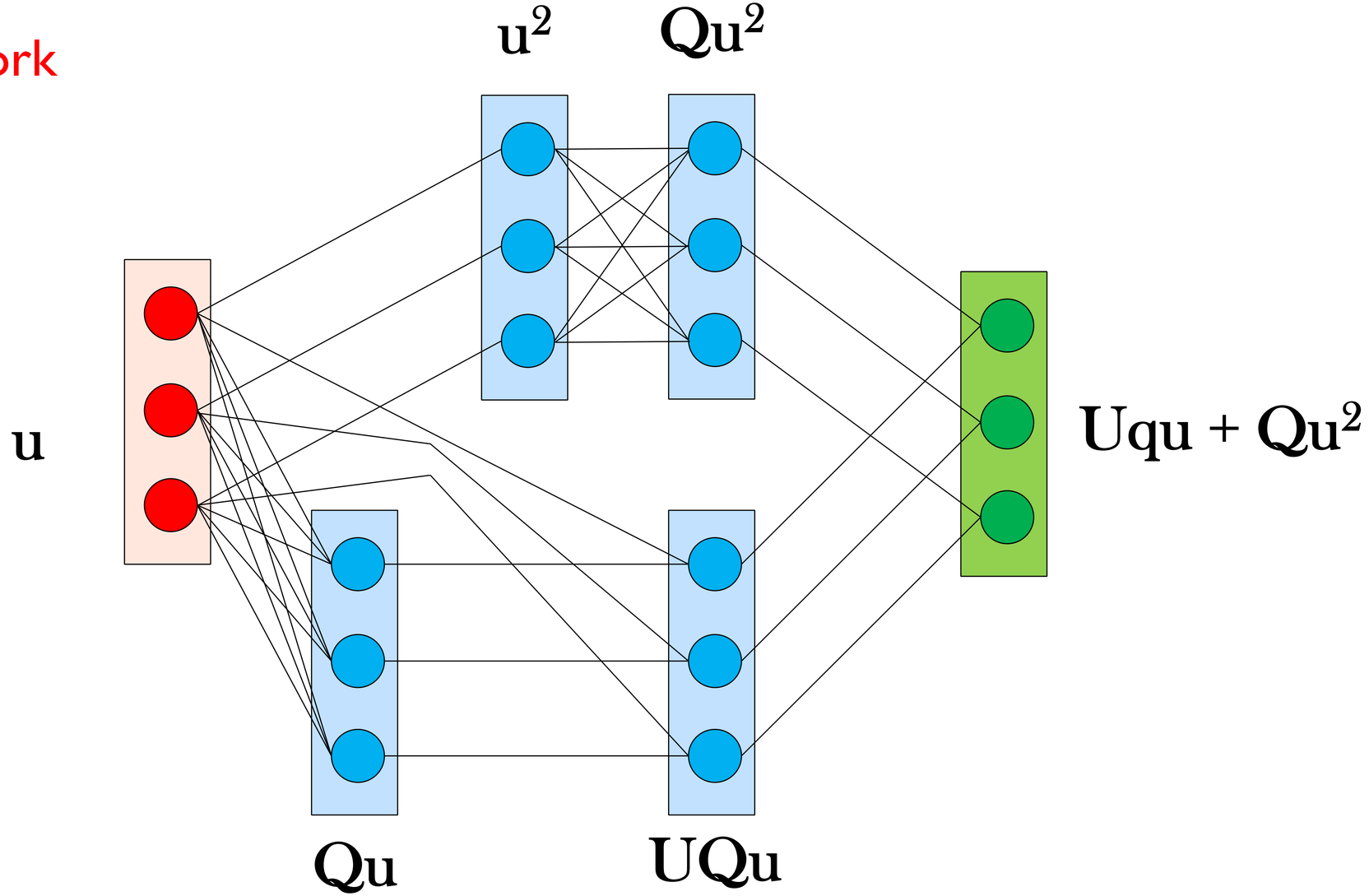
$$
\begin{aligned}
\mathbf{u}_t^{(1)} + \boldsymbol{\mathcal{L}}(\mathbf{u}^{(1)})\mathbf{u}^{(1)} &= \sigma_1 \mathbf{P}^{-1}(u_0^{(1)} - g)\mathbf{e}_0 \\
\mathbf{u}_t^{(2)} + \boldsymbol{\mathcal{L}}(\mathbf{u}^{(2)})\mathbf{u}^{(2)} &= \sigma_2 \mathbf{P}^{-1}(u_0^{(2)} - u_2^{(1)})\mathbf{e}_0 \\
&\vdots \\
\mathbf{u}_t^{(N)} + \boldsymbol{\mathcal{L}}(\mathbf{u}^{(N)})\mathbf{u}^{(N)} &= \sigma_N \mathbf{P}^{-1}(u_0^{(N)} - u_2^{(N-1)})\mathbf{e}_0 .
\end{aligned}
$$

…similar to the linear case we can prove stability!

A Numerical Technique for PDEs

We can build a new network
to construct the operator

$$\mathcal{L}(\mathbf{u}) = \frac{1}{3}\mathbf{P}^{-1}\mathcal{Q}(\mathbf{u})$$



(Machine) Learning to Differentiate

Discretization for non-Linear Advection

No Constraints

unstable

SBP Constraint via Loss Function

Regularization		
N	error	rate
10	1.8121e-02	-
20	3.0915e-03	2.55
40	4.0387e-04	2.94
80	6.0469e-05	2.74
160	1.0986e-05	2.46

SBP Constraint via Network Structure

Weight restriction		
N	error	rate
10	2.6565e-02	-
20	2.4962e-03	3.41
40	2.7021e-04	3.21
80	3.2516e-05	3.05
160	4.1301e-06	2.98

Summary & Conclusions

We built discretization schemes (nodal discontinuous Galerkin) using neural networks

We enforced stability constraints using the SBP property

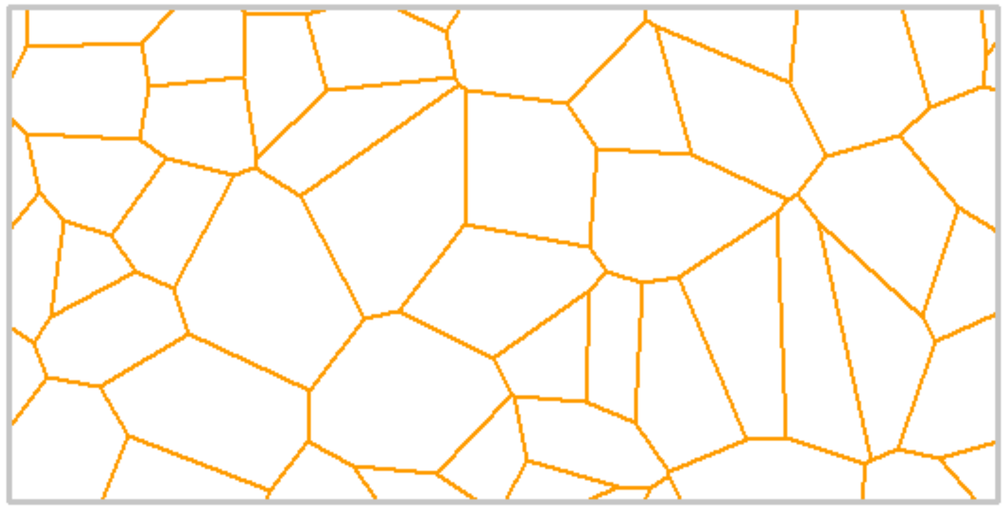
We investigated the possibility of enforcing (physical) properties either through regularization (loss function) or via network constraints (weights)

We constructed a specific network architecture for non-linear operators

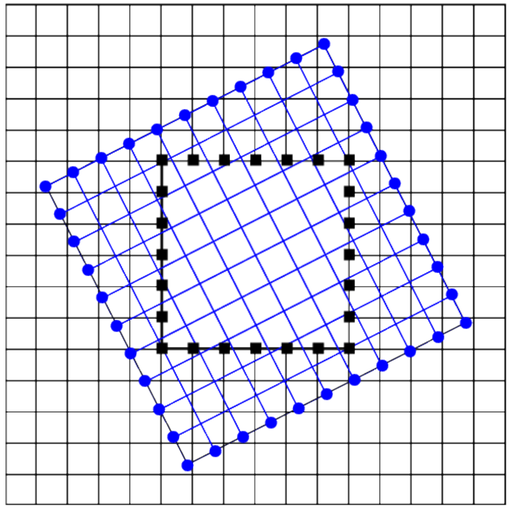
We observe that training with specific function sets (polynomials) accelerates learning

What's Next?

Genuine Multi-dimensional and Provably Stable Discretization Schemes for



Unstructured Polyhedral Grids



Irregular/Overlapping Grids

Final Thoughts

LINKÖPING UNIVERSITY

Data-driven techniques offer great potential, and we are just at the beginning…

Great diversity in techniques and objectives create difficulties in assessing where we are and how far we are moving

Low-hanging fruits: ML techniques are powerful generator of heuristics

Long term: need to consider different paradigm that blend numerical techniques and physics modeling to take full advantage of both HPC and ML

Questions?

thank you

Ali Kashefi, Aashwin Mishra (Part 1 & 2)

Oskar Ålund, Gianluca Iaccarino, Jan Nordström,
“Learning to Differentiate”, submitted to JCP 2020