When Big Data and Machine Learning meet Partial Differential Equations

Marc Schoenauer
Michele Alessandro Bucci
Equipe TAU, INRIA, LRI, UP-Sud et CNRS,
Université Paris-Saclay

Statistical Modeling and Machine Learning in Meteorology and Oceanography
Brest, Feb. 10., 2020

Machine Learning

AI/Big Data is (Deep) Machine Learning

Machine Learning

AI/Big Data is (Deep) Machine Learning

FALSE

Machine Learning

although ...

AI/Big Data is (Deep) Machine Learning

What has changed :

- Data Deluge — avent of WWW
- Moore law — or continuation
- New algorithms — or better understanding of old ones

Machine Learning

Learning from examples — recognition tasks

- Supervised — all examples are labelled
- Semi-supervised — some examples are labelled
- Unsupervised — no example is labelled

Reinforcement Learning — sequential decision making

Machine Learning

Learning from examples	recognition tasks
• Supervised	all examples are labelled
• Semi-supervised	some examples are labelled
• Unsupervised	no example is labelled
Reinforcement Learning	sequential decision making

Supervised Learning

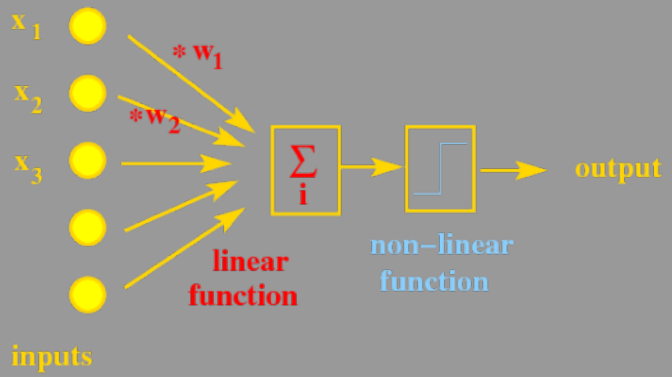
A toy case-study

- One example = (x_1,x_2, y), where **y** is the **label** (red or blue here)
- **Goal**: find a **model** $f(x_1,x_2)$ that separates the labels
- allowing to correctly label future unlabelled example from (x_1,x_2)

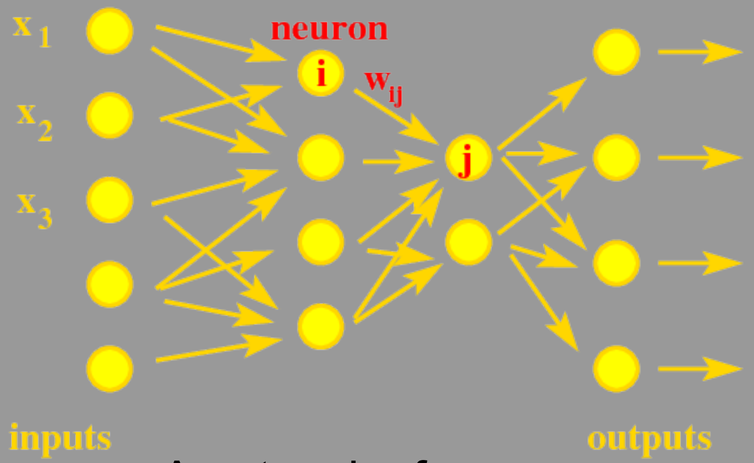
Supervised Learning

A zoology of models

- Linear and Polynomial
- Bayésiens Networks
- Decision trees and Random Forests
- Support Vector Machine (kernel machines)
- **Artificial Neural Networks**



One neuron



A network of neurons

Parameters are the **weights** w_{ij}

Deep Supervised Learning

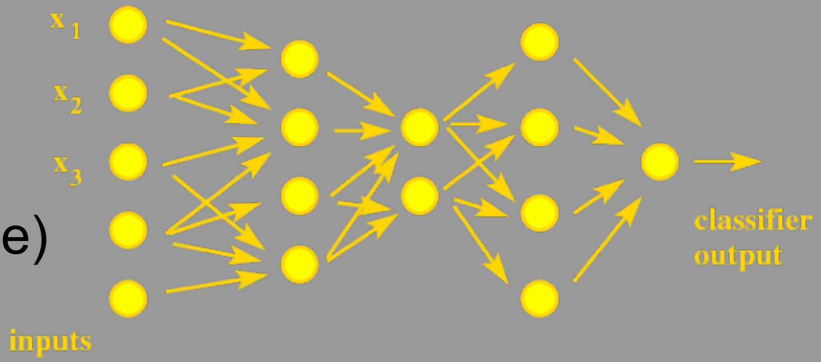
Learning Phase

Gradient **back-propagation** aka **Stochastic Gradient Descent**

- Present the examples 1 by 1
  - or mini-batch by mini-batch
- **Forward** pass: Compute the **Loss**

  e.g., $L = \sum |y(x_1,x_2) - NN(x_1,x_2)|^2$
- **Backward** pass: Compute $\nabla_w L$ (chain rule)
- Modify the weights w_{ij} from $\nabla_w L$

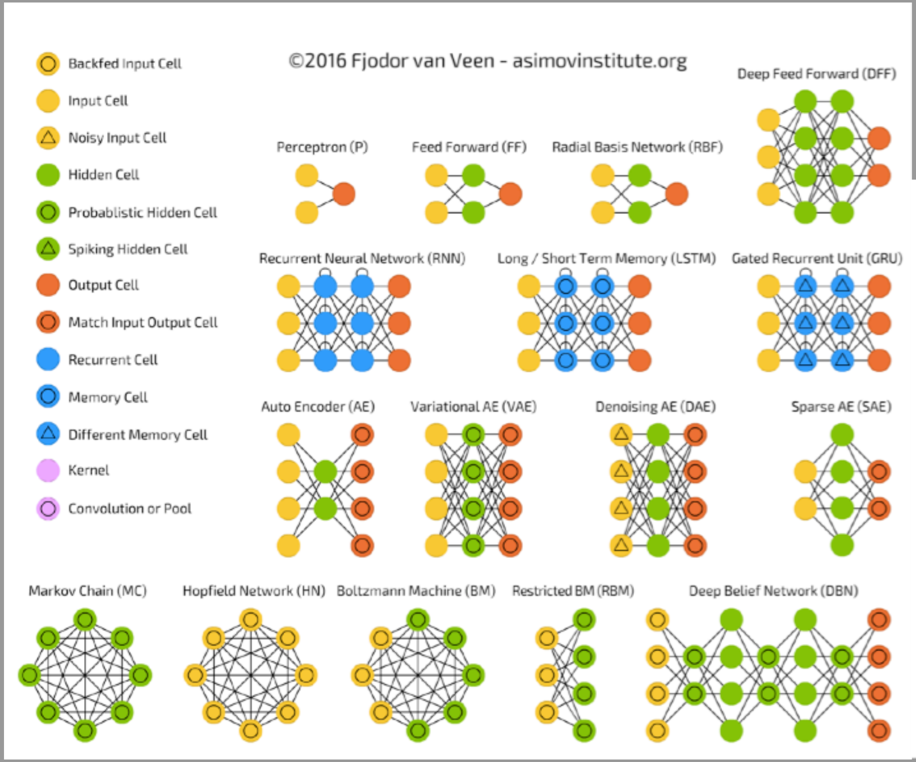
  to decrease of the loss
- Loop



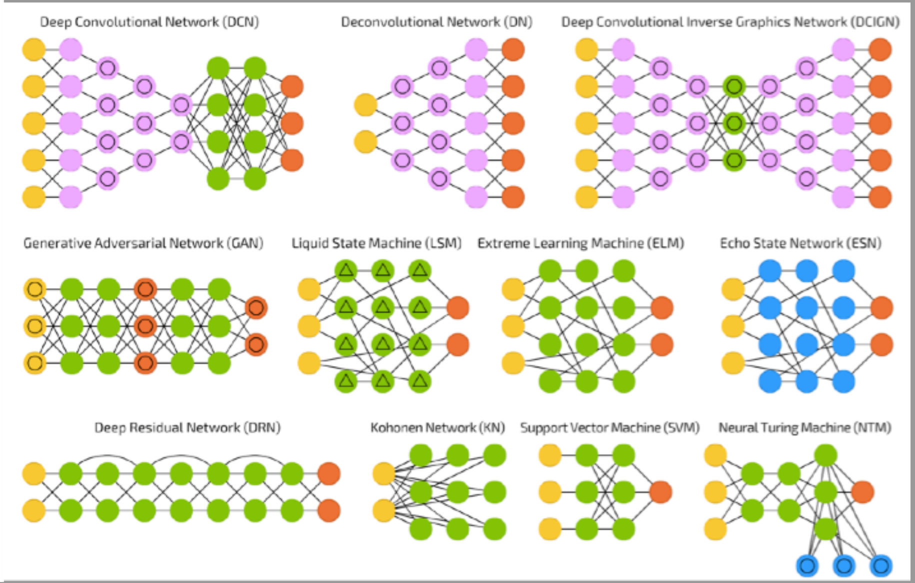
Recognition Phase aka Inference

Present an unlabelled example, the output of the network is the predicted label

Neural Networks zoology

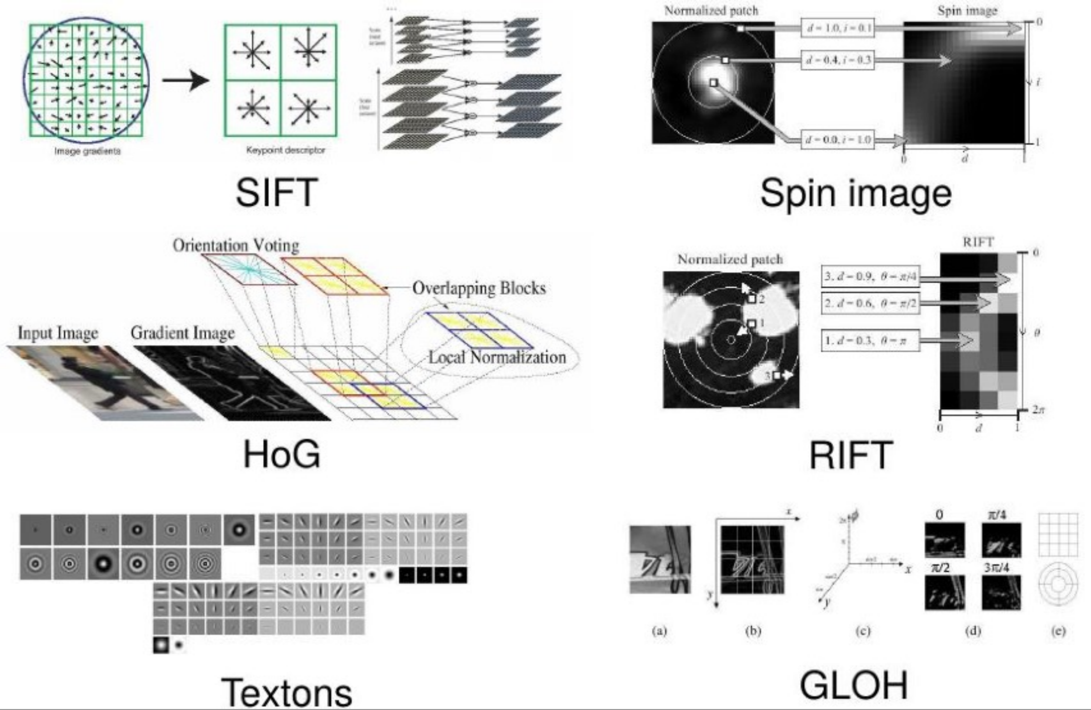


Almost complete list of
neural network architectures



Good Old Computer Vision

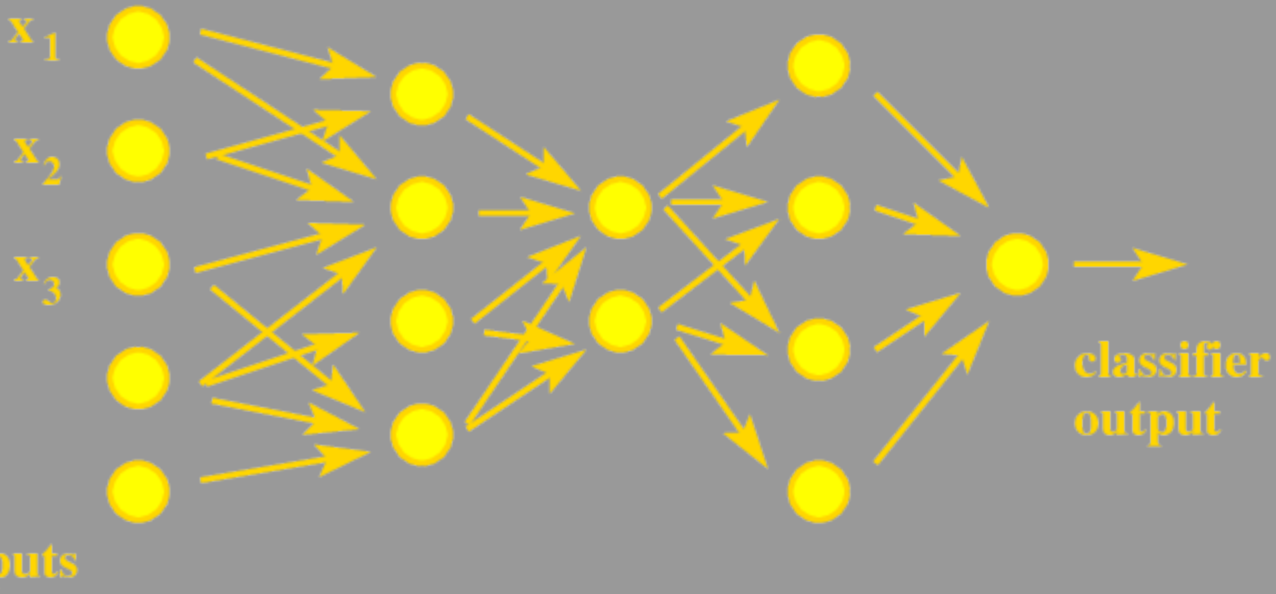
Hand-made **features** → Learned classifier → CAT



End-to-end Learning

Features and Classifier are learned together

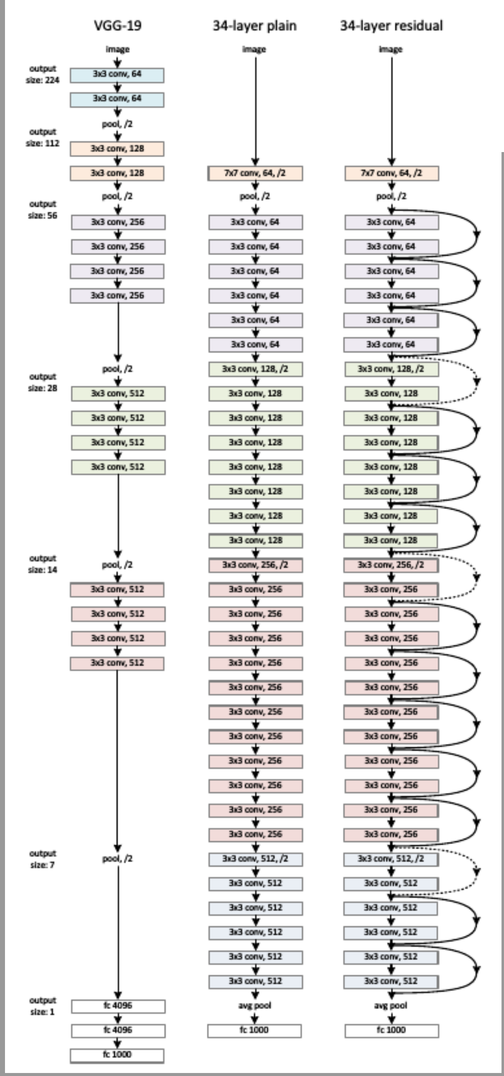
CAT



State-of-the-art

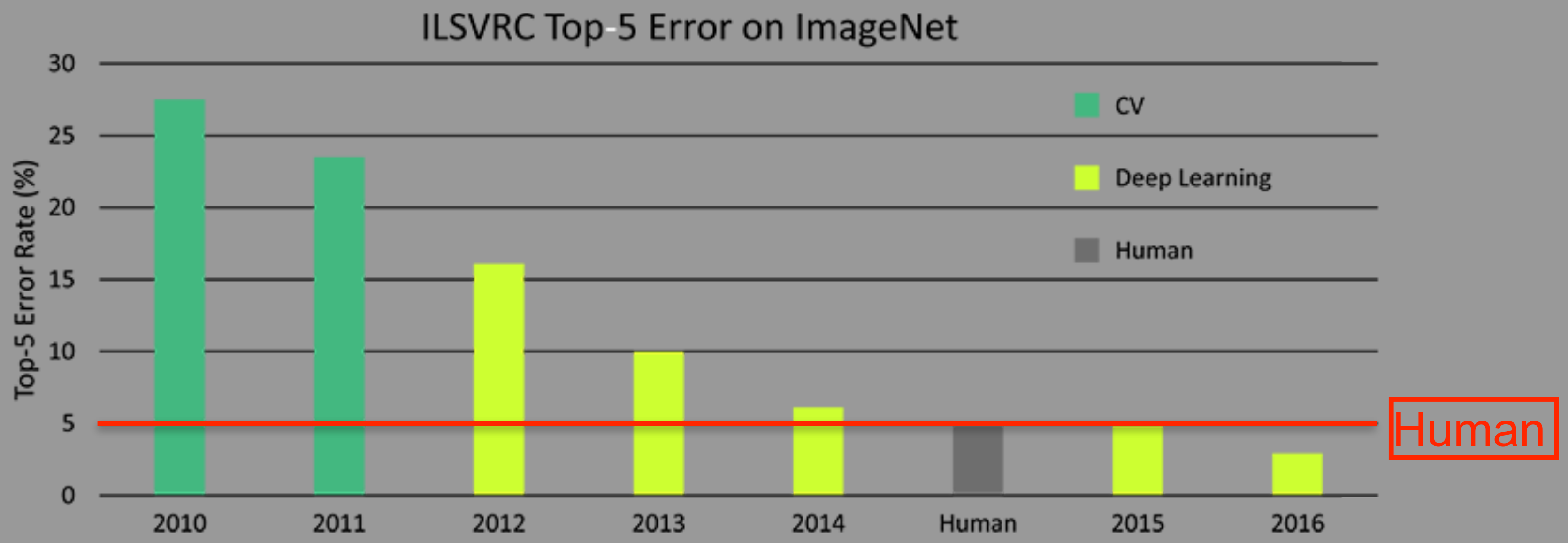
- Many datasets available
  - ImageNet : 14+ M examples, 1000 classes
- (pre-trained) networks with numerous layers
  - up to 152 !
- Millions to billions weights
  - hundreds of GPU mandatory for learning
- Several tricks of the trade
  - Dropout, residual layers, ensembles, ...

He et al., 2015



Deep Learning

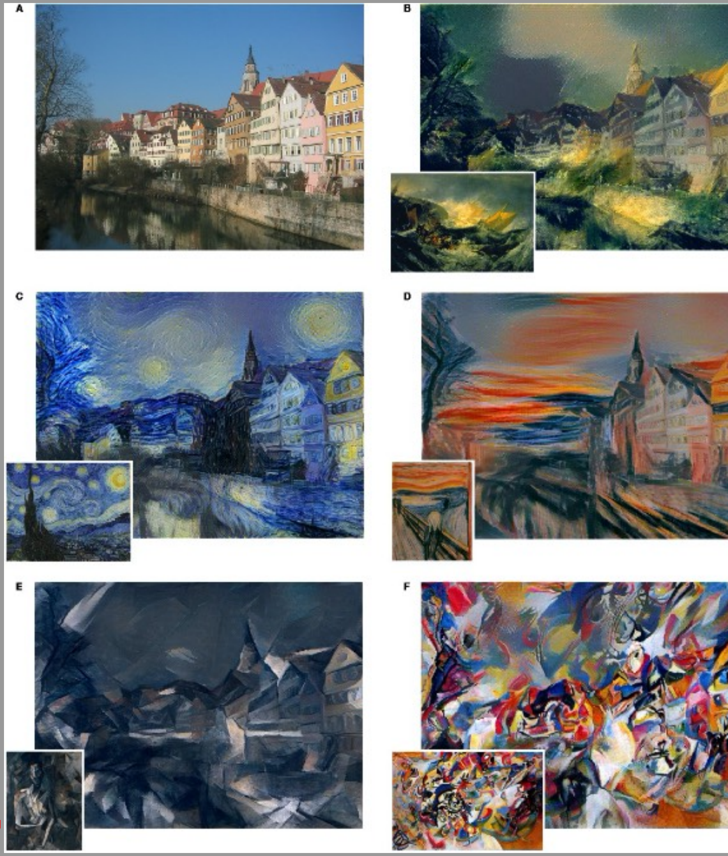
Better than human learning



Deep Learning

Outstanding performances … in well-focused domains

- Supervised learning
  - Image/videos
  - NLP / translation
- Reinforcement/sequential learning
  - Games (AlphaZero)
  - Dynamical systems (Fluid/Structure)
- Representation learning
  - generative models (GANs)
  - domain transfer (DANNs)
- Toward **Differentiable Programming**



many successes, and fun applications, **but ...**

DL configuration: more of an art

A very high number of hyperparameters to tune

- Cost function
- Topology of the network
  - nblayers, nb neurons, residual or not residual, ...
- Activation functions (sigmoid, tanh, ReLU, …)
- Batch size (and curriculum)
- Optimizer (SGD, Momentum, Adam, Adagrad, …)
  - and its parameters (e.g., learning rate)
- Initialization
- Dropout, Batch Normalisation, ...
- ...

Empirical rules, or meta-optimization (meta-costly)

Partial Differential Equations

Mathematically grounded representation of phenomena

- Poisson equation

$$\Delta\varphi = \nabla^2\varphi = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\varphi(x,y,z) = f(x,y,z)$$

- Heat equation

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

- Generally no analytical solution

Numerical solutions by discretisation of the domain

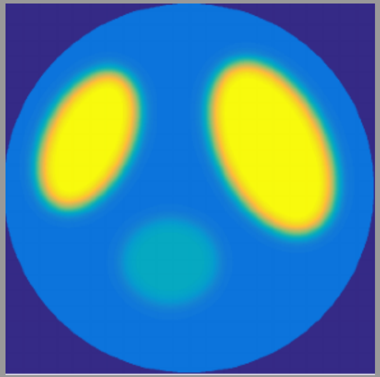
- Finite differences, finite elements, finite volumes, ...

Partial Differential Equations

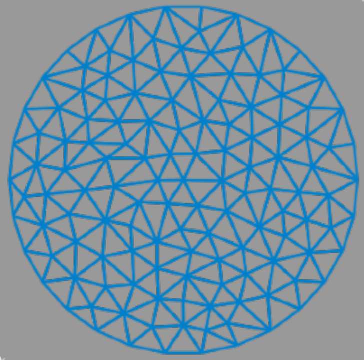
Mathematically grounded representation of phenomena: exemple

- Generalized Poisson equation

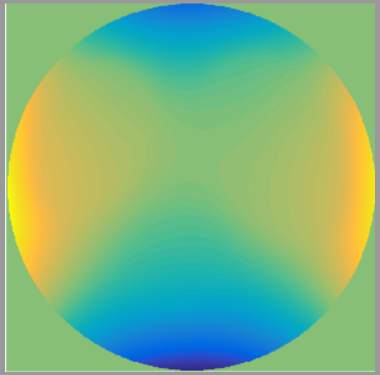
$$\begin{aligned} &\nabla \cdot \Big(\sigma(x)\nabla u(x)\Big) = 0, && x \in \Omega \subset \mathbb{R}^2 \\ &u(x) = u_0(x), && x \in \partial\Omega. \end{aligned}$$



Conductivity σ



Sample mesh



Computed current

PDEs and ML

But does “The Data Deluge Make the Scientific Method Obsolete”(*)?

The answer is of course “no” ...

This talk: survey of synergies between Machine Learning and PDE solving.

(*) C. Anderson (2008). “The End of Theory”, Wired Magazine. url: https://www.wired.com/2008/06/pb-theory/.

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
  - PDE-NET
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

ML and PDEs: Agenda

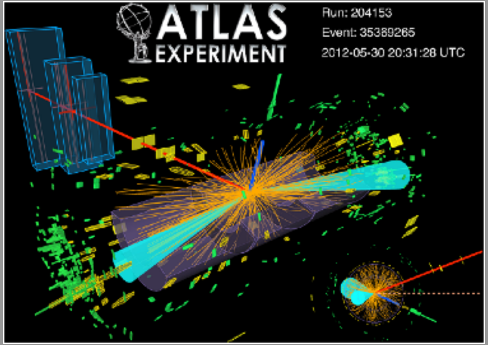
- Simulation is fine, but huge output data
    - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
    - DL as surrogate model (whole simulation, or sub-components)
    - Physic Informed Deep Learning
    - Deep Galerkin Method (high dimensions)
    - PDE-NET
- Inverse problems / calibration
    - Often ill-posed
- Mechanistic model is unknown
    - Learn an analytical model from data
    - Learn a black-box model from data

Data Analytics for Simulation Results

- Experiments, either real or simulated, produce loads of data
  - Manual interpretation not possible any more
- Particle physics experiments: several challenges

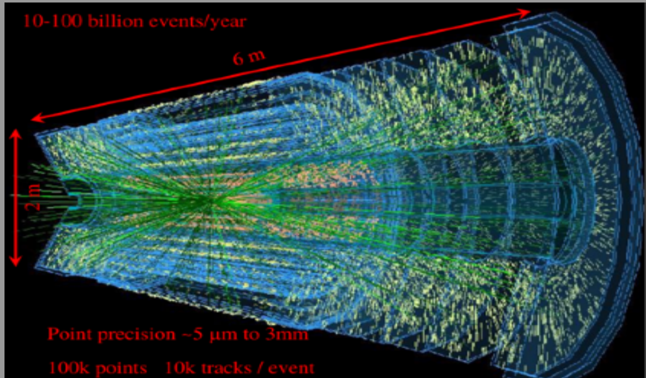
The Higgs boson Challenge (2014)

- Largest Kaggle challenge (1785 entries)
- < 100 Boson events per year (10^{10} events)
- DL winner … unpractical for physicists



The TrackML challenge (2018-2019)

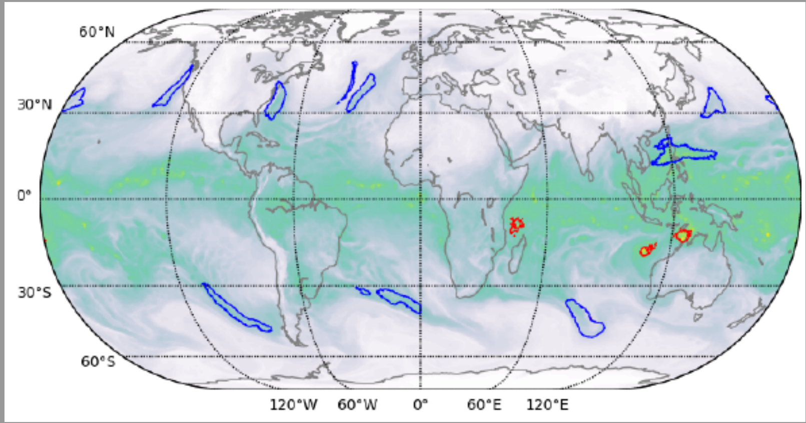
- 10-100 Billion events/year
- Reconstruct particle trajectories
- from sparse point-wise traces
- Winner is not a machine learner



Deep Learning for Climate Analytics

Recognize forthcoming Tropical Cyclones in simulation outputs

- Public Data: Community Atmosphere Model (CAM5)
- 16 variables
- 1152x768 spatial grid, 3 hours time step
- 100 years are available, ~63000 hires samples
- labeled with supervised expert heuristics
- 3 classes: TCs (0.1%), Atmospheric Rivers (1.7%) and Background



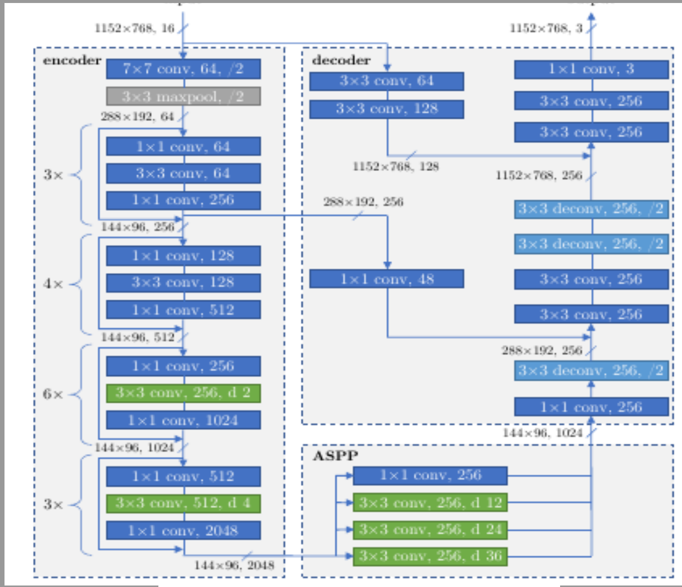
T. Kurth et al., "Exascale Deep Learning for Climate Analytics", *Super Computing* 2018

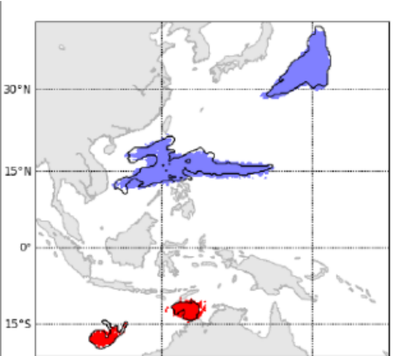
Deep Learning for Climate Analytics

- Specific architecture DeepLabV3+
- High-speed parallel data staging
    - 27 360 GPUs, 999 PF/s
    - distribution using Horovod (MPI)
    - and *all_reduced* procedure
- Ran on Summit, first in TOP500

- Weighted loss 1/sqrt(frequency)
- Layer-wise adaptive learning rate control

A new era for extreme weather prediction!





T. Kurth et al., “Exascale Deep Learning for Climate Analytics”, *Super Computing* 2018

Data Analytics for Simulation Results

- Analysis of simulation results can also be used as feedback to the simulation itself
  - to detect numerical instabilities before they take place
  - and take appropriate counter-measures (e.g., locally modify diffusion parameter, adaptative discretization).

BUT

- … need some expert labeling

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
  - PDE-NET
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

Accelerating simulations

A basic example: Regression of the complete solution

- Poisson equation: $\nabla \cdot (\varepsilon(x)\nabla\phi(x)) = -\rho(x)$, $x \in D$

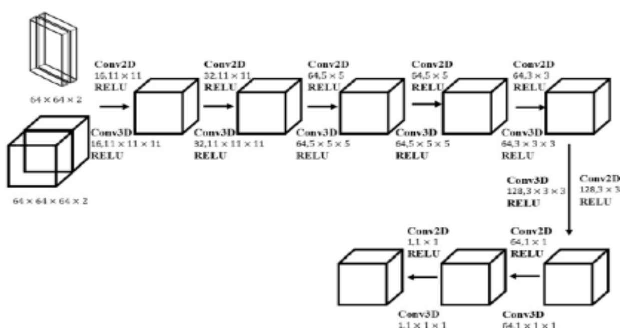
  $$\varphi|_{\partial D} = 0$$

- **Inputs**: permittivity ε and source distribution ρ
- **Outputs**: dielectric potential ϕ

- Finite differences on a fixed Cartesian grid
- Inputs assumed constant in each pixel/voxel

- “Standard” CNN architecture
- Regularized MSE loss

$$\|log_{10}(\phi) - log_{10}(\widehat{\phi})\|^2 + \frac{\lambda}{2n}\sum_{w} w^2$$

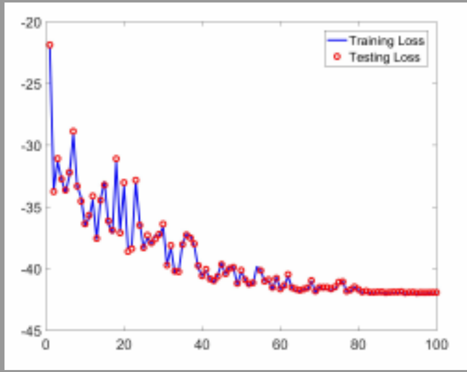


W. Tang *et al.*, "Study on a Poisson's equation solver based on deep learning technique," *2017 IEEE Electrical Design of Advanced Packaging and Systems Symp. (EDAPS)*, 2017.

Regression of the complete solution (2)

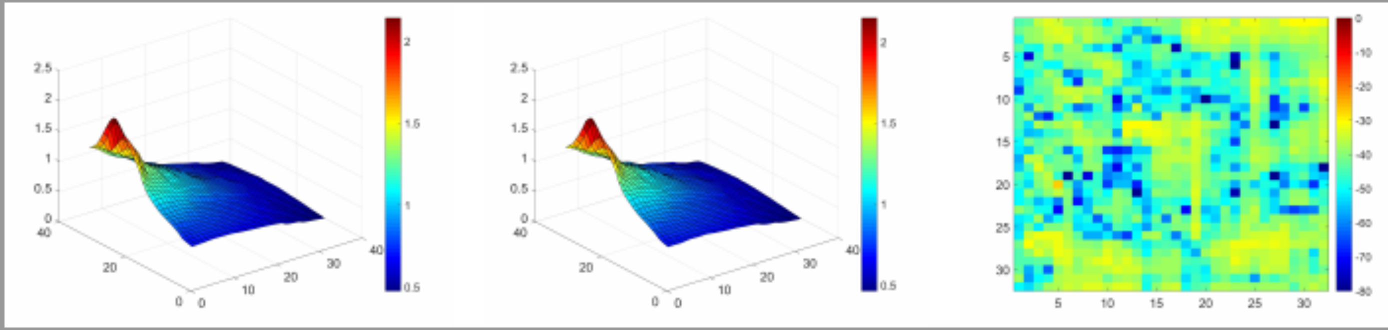
- 64x64(x64) grids
- 8000/2000 examples (finite differences)

- **Error**: 1.5% in 2D, 3% in 3D
- from **16s** to **0.13s** for 2000 simulations in 2D
  from **292s** to **1.2s** for 5 simulations in 3D



BUT

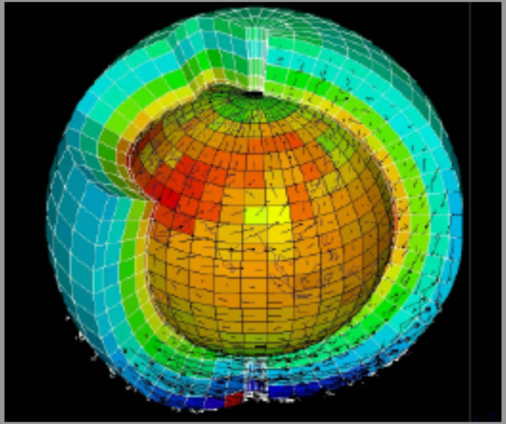
- No scaling study
- Linear model
- Fixed grid/mesh



Accelerating simulations

Learning sub-scale phenomena

- Global climate modeling
  - 2° horizontal resolution, 30 altitude levels
  - 30mn time step
- needs to solve CRMs (Cloud Resolving Models)
  - turbulence + cloud convection + ...
  - in each column (4km-wide),
  - at each time-step (20s)



- Train a DNN
- Inputs(z): temperature, humidity, wind profile, … (→ dim 94)
- Outputs(z): heating, moistening, radiative fluxes, … (→ dim 65)
- 9-layers fully connected NN, 256 neurons/layer
- Training data: one-year SPCAM simulations - 140M example

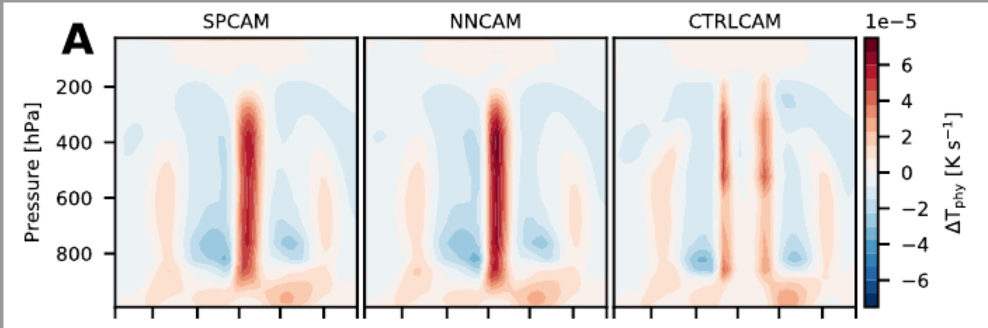
S. Rasp et al. “Deep learning to represent sub-grid processes in climate models”. *PNAS* 2018

Accelerating simulations

Learning sub-scale phenomena (2)

Results

- 20x speedup
- Means and variability OK



- Good conservation of energy (though not prescribed)
- Good interpolation between extreme values of train data
- Poor extrapolation beyond train data

BUT

- Learning from observations only? Too few available

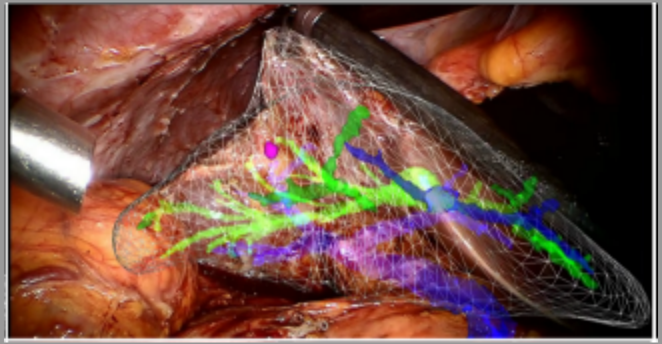
S. Rasp et al. “Deep learning to represent sub-grid processes in climate models”. *PNAS* 2018

Soft Body Deformation

Need patient-specific real-time simulation of laparoscopy

- Liver is hyper viscoelastic and anisotropic
- Several complex PDEs for soft tissues
  - anyway an approximation
- Material identification
- Patient-specific geometry
  - Not a big issue, but time consuming
- Boundary conditions are essential
  - but difficult to obtain from images

- Need less than 3mm error
- in less than 50ms per image

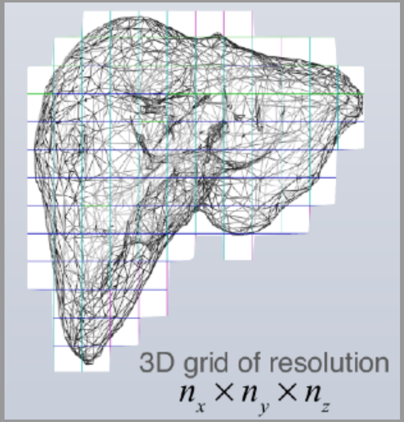


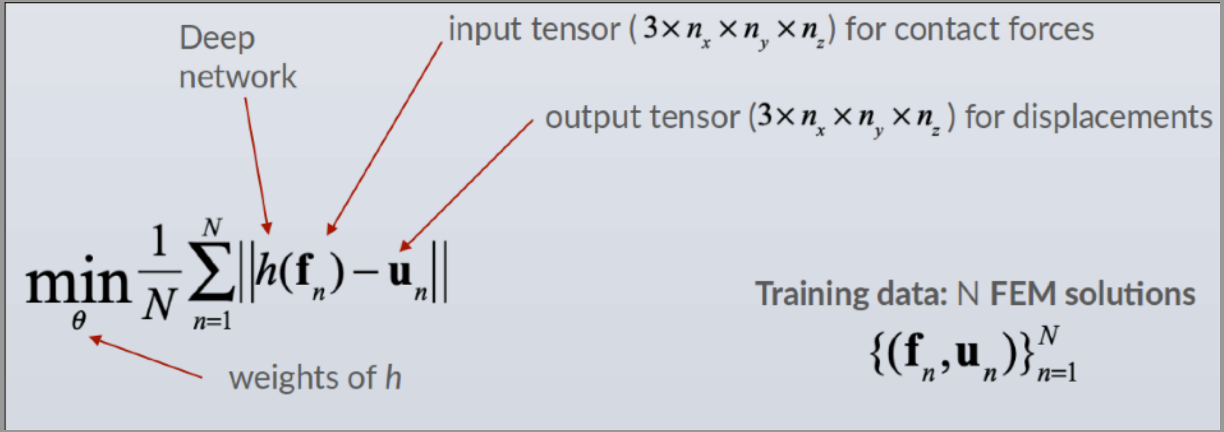
A. Mendizabal, J-N. Brunet and S. Cotin - Physics-based Deep Neural Network for Augmented Reality during Liver Surgery, MICCAI 2019.

Soft Body Deformation

Bottleneck: real-time simulation

→ Replace FEM simulation by a Deep Network
Supervised learning (regression) of simulation results

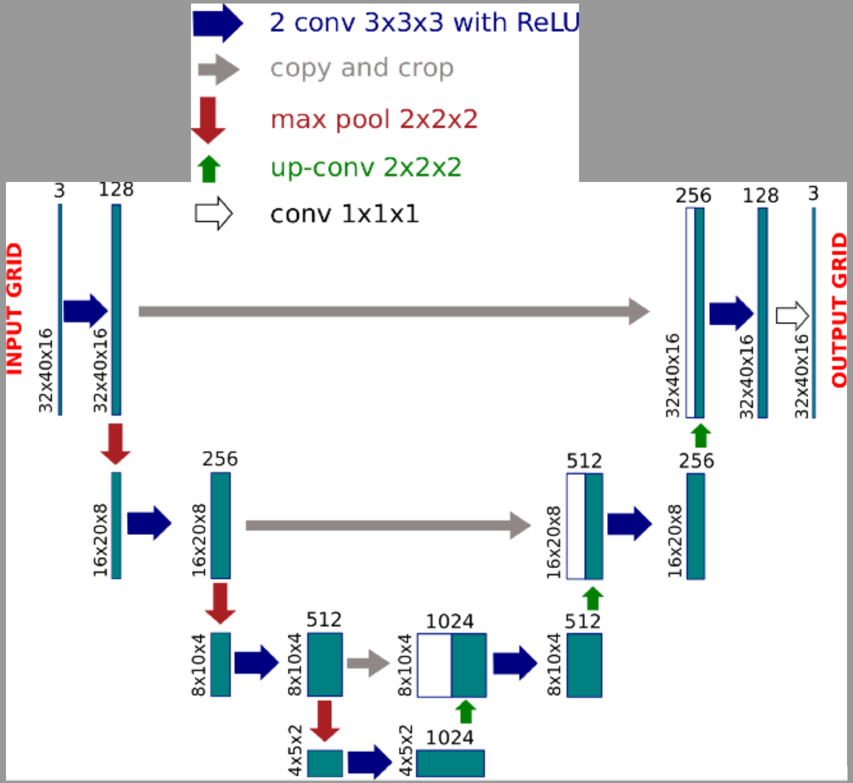




A. Mendizabal, J-N. Brunet and S. Cotin - Physics-based Deep Neural Network for Augmented Reality during Liver Surgery, MICCAI 2019.

Soft Body Deformation

Inspired by the U-net architecture* for image segmentation



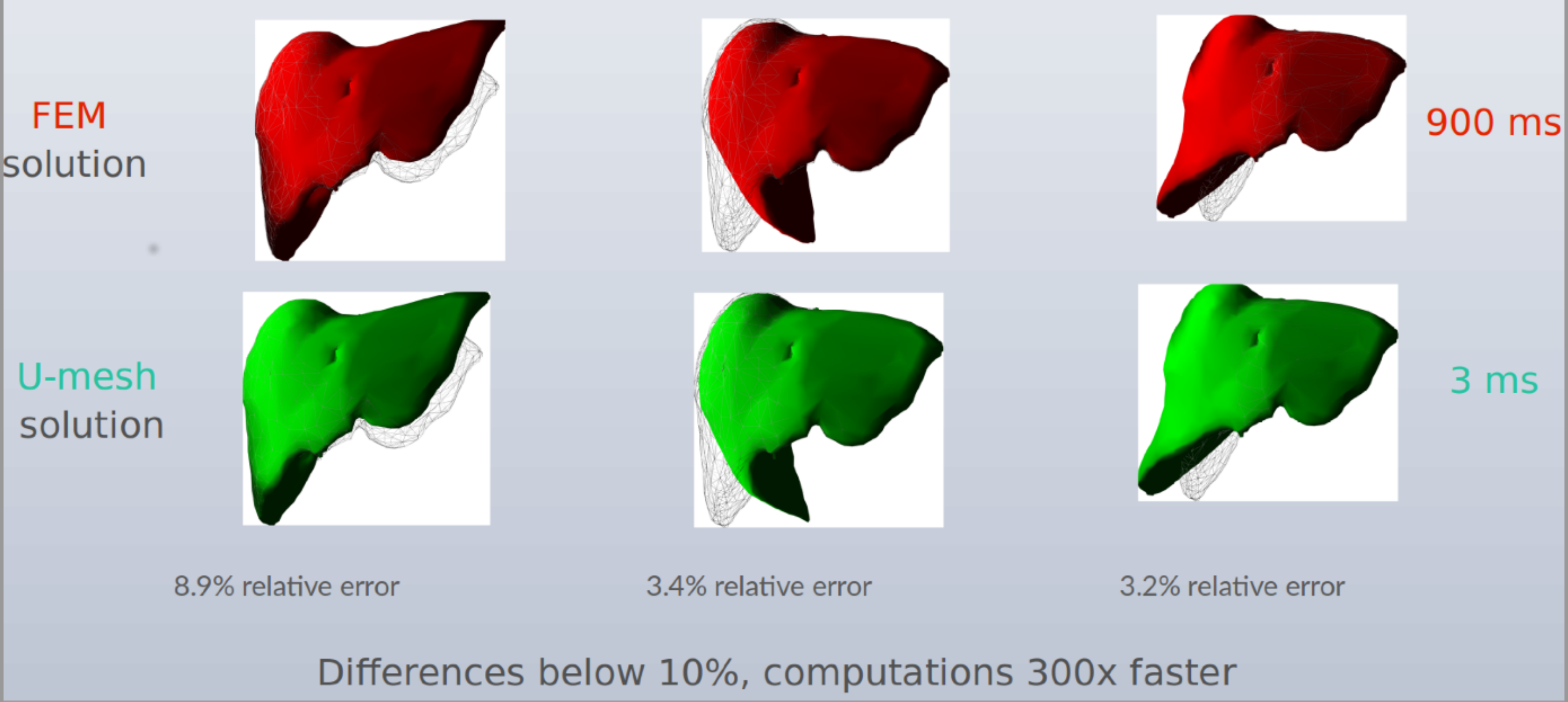
- Fully convolutional network
- Encoding:
  transforms input to reduced space
- Decoding:
  expands to original dimensions
- # steps and # channels control the accuracy.
- Each convolution kernel isolates different characteristics of u

Very similar to POD approach!

(*) Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015.

Soft Body Deformation

Results



A. Mendizabal, J-N. Brunet and S. Cotin - Physics-based Deep Neural Network for Augmented Reality during Liver Surgery, MICCAI 2019.

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
  - PDE-NET
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

Physics Informed Deep Learning

Data-driven solution of PDEs

- Define $u_t = \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots)$

$$f := u_t - \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots)$$

- and minimize $\frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 + \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$

- $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ initial and boundary training data
- $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ collocation training points

M. Raissi. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations." ArXiv 1711.1056.

Physics Informed Deep Learning

Automatic differentiation

- Thanks to Differentiable Programming
- e.g., in TensorFlow

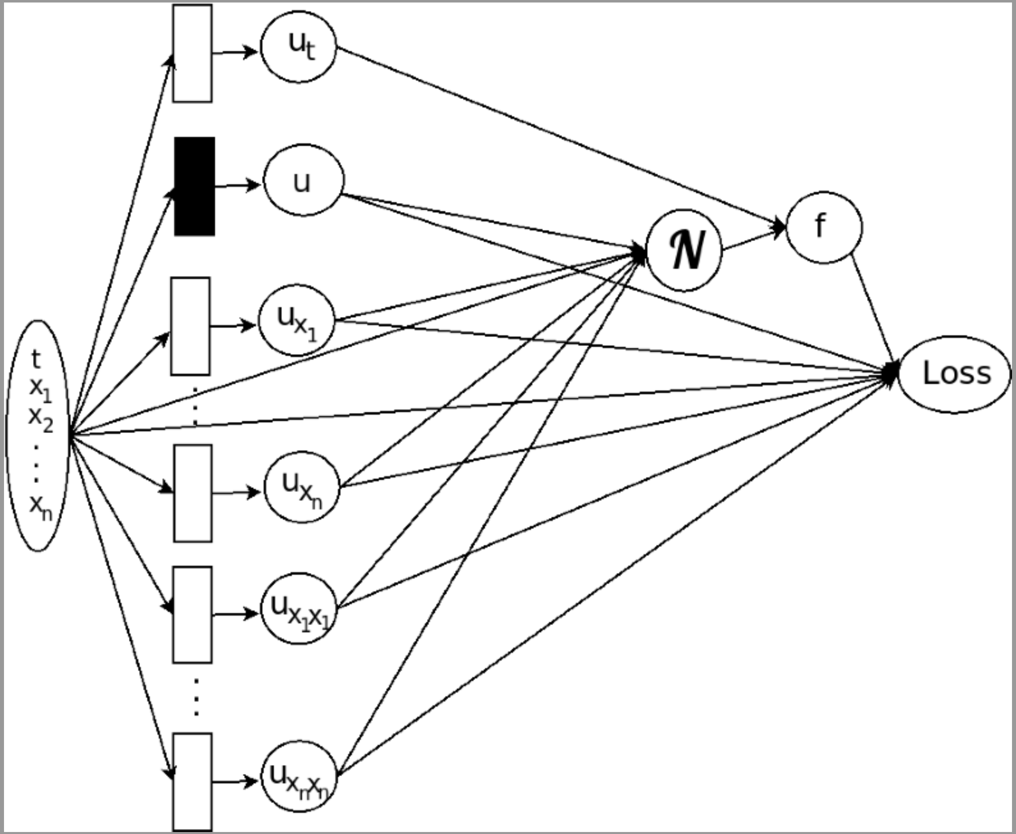
```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

M. Raissi, “Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations,” ArXiv 1711.1056,

Physics Informed Deep Learning

A meshless approach

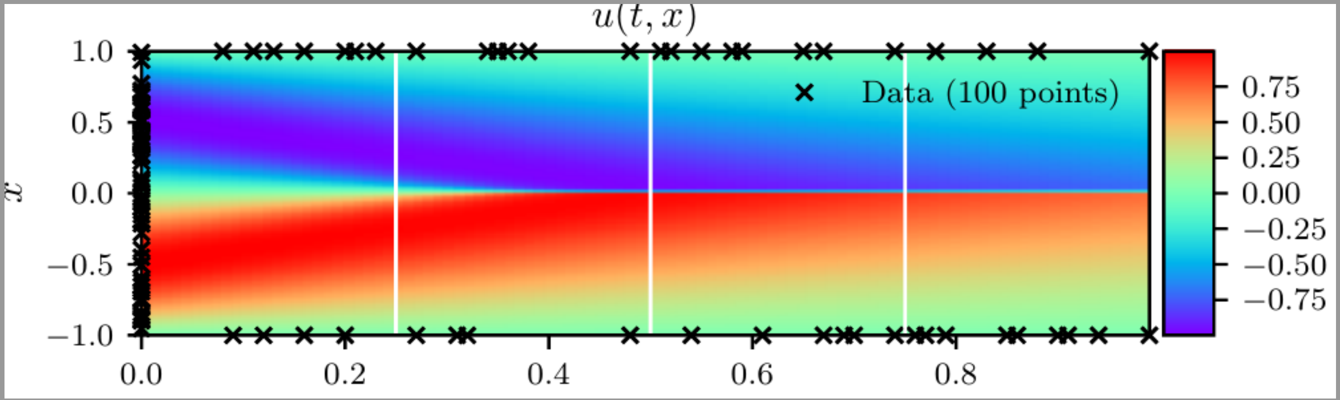


M. Raissi. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations." ArXiv 1711.1056.

Physics Informed Deep Learning

Burger's equation

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1]$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$



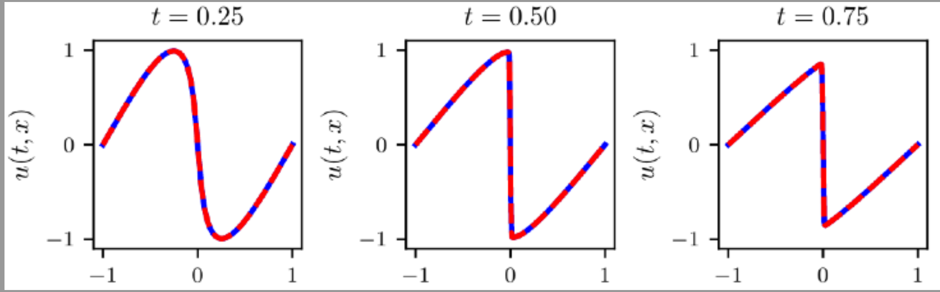
Data: training boundary points X + 10000 collocation points (not shown).
Predicted dynamics, **MSE** = 6.7 10^{-4}.

M. Raissi. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations." ArXiv 1711.1056.

Physics Informed Deep Learning

Burger's equation

Exact vs predicted



N_u \ N_f	2000	4000	6000	7000	8000	10000
20	2.9e-01	4.4e-01	8.9e-01	1.2e+00	9.9e-02	4.2e-02
40	6.5e-02	1.1e-02	5.0e-01	9.6e-03	4.6e-01	7.5e-02
60	3.6e-01	1.2e-02	1.7e-01	5.9e-03	1.9e-03	8.2e-03
80	5.5e-03	1.0e-03	3.2e-03	7.8e-03	4.9e-02	4.5e-03
100	6.6e-02	2.7e-01	7.2e-03	6.8e-04	2.2e-03	6.7e-04
200	1.5e-01	2.3e-03	8.2e-04	8.9e-04	6.1e-04	4.9e-04

Influence of amount of data on accuracy

M. Raissi. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations." ArXiv 1711.1056.

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
  - PDE-NET
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

Deep Galerkin Method

- **Original motivations**: High-dim PDEs, parameterized PDEs
- **Context**: parabolic PDEs

$$\partial_t u(t,x) + \mathcal{L}u(t,x) = 0, \qquad (t,x) \in [0,T] \times \Omega$$
$$u(0,x) = u_0(x), \qquad x \in \Omega$$
$$u(t,x) = g(t,x), \qquad x \in [0,T] \times \partial\Omega,$$

- **Baseline**: minimize

$$\|\partial_t f + \mathcal{L}f\|^2_{2,[0,T]\times\Omega} + \|f - g\|^2_{2,[0,T]\times\partial\Omega} + \|f(0,\cdot) - u_0\|^2_{2,\Omega}$$

- **Convergence**: the solution learned by the 1-layer NN converges toward the solution of the PDF when #neurons → ∞
- **Derivatives**: all ∂_t and ∂_x can be computed analytically
- **Specific trick** to avoid computing all 2nd order derivatives
  - based on Monte-Carlo approximation
  - but introduces bias and variance

J. Sirignano,K. Spiliopoulos. “DGM: A deep learning algorithm for solving partial differential equations”. *J. Comp. Physics* 375:1339-1364, 2018

Deep Galerkin Method

The Neural Network

- **Inputs**: **x** = (t,x)
- **Outputs**: f(**x**),
- **Loss**: for (t_n,x_n):

$$\left(\frac{\partial f}{\partial t}(t_n, x_n; \theta_n) + \mathcal{L} f(t_n, x_n; \theta_n)\right)^2 + \left(f(\tau_n, z_n; \theta_n) - g(\tau_n, z_n)\right)^2 + \left(f(0, w_n; \theta_n) - u_0(w_n)\right)^2$$

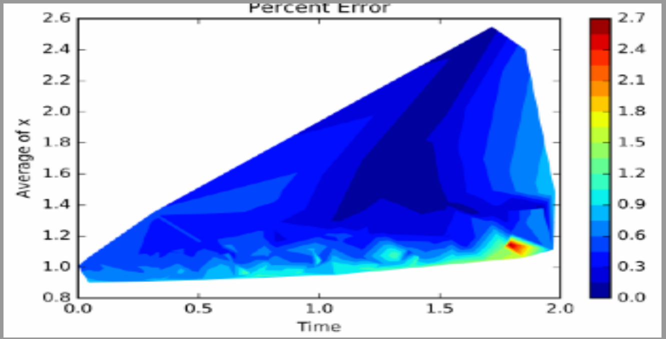
  - no regularization, “infinite” sample set
- **Architecture**: inspired by LSTM topology
  - L hidden layers (typically 4),
    - 4 sub-layers per layer,
    - **x** fed into all sublayers
  - M units per layer (typically 50)
  - *tanh* transfer functions
- **Parallel** asynchronous stochastic gradient on 5 GPUs
- 100000 iterations, 1000 samples on each GPU per iter.
  - **500 M points** altogether
- Adam optimizer, with complex **ad hoc schedule** to decrease the learning rate

J. Sirignano,K. Spiliopoulos, “DGM: A deep learning algorithm for solving partial differential equations”, *J. Comp. Physics* 375:1339-1364, 2018

Deep Galerkin Method: Results

Results

Number of dimensions	Error
3	0.05%
20	0.03%
100	0.11%
200	0.22%



- "American Options" PDE with semi-analytic solution
- Parameterized Burger equation
  - **Inputs**: (t, x, a, b, α, υ)
  - **Results**: Indistinguishable from the finite differences solutions

- … without analytical solutions:
  - theoretical bounds
  - results are within bounds

$$
\begin{aligned}
&\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} - \alpha u \frac{\partial u}{\partial x}, \qquad (t,x) \in [0,1] \times [0,1],\\
&u(t, x=0) = a,\\
&u(t, x=1) = b,\\
&u(t=0, x) = g(x), \qquad x \in [0,1].
\end{aligned}
$$

J. Sirignano,K. Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations". *J. Comp. Physics* 375:1339-1364, 2018

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
  - PDE-NET
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

PDE-NET

- **Original motivations**: learn PDEs improve flexibilty
- **Context**: 2D convection-diffusion equation $u_t(t,x,y) = F(x,y,u,u_x,u_y,u_{xx},u_{xy},u_{yy},\ldots)$

$$\begin{cases} \frac{\partial u}{\partial t} & = a(x,y)u_x + b(x,y)u_y + cu_{xx} + du_{yy} \\ u|_{t=0} & = u_0(x,y), \end{cases}$$

- **Baseline**: minimize

$$L = \sum_{i,j} l_{ij}, \text{where} \quad l_{ij} = ||u_j(t_{i+n},\cdot) - \tilde{u}_j(t_{i+n},\cdot)||_2^2$$

- **Idea**: use convolution neural network to learn derivative operators

$$\tilde{u}(t_{i+1},\cdot) = D_0u(t_i,\cdot) + \Delta t \cdot F(x,y,D_{00}u,D_{10}u,D_{01}u,D_{20}u,D_{11}u,D_{02}u,\ldots).$$

$$D_{ij}u \approx \frac{\partial^{i+j}u}{\partial^i x \partial^j y}.$$

- **Specific trick** to constrain convolution operation
  - filters constrained using their associate moment matrices
  - multiple filters to approximate a given differential operator

Long, Zichao, et al. "PDE-net: Learning PDEs from data." *arXiv preprint arXiv:1710.09668* (2017)

PDE-NET

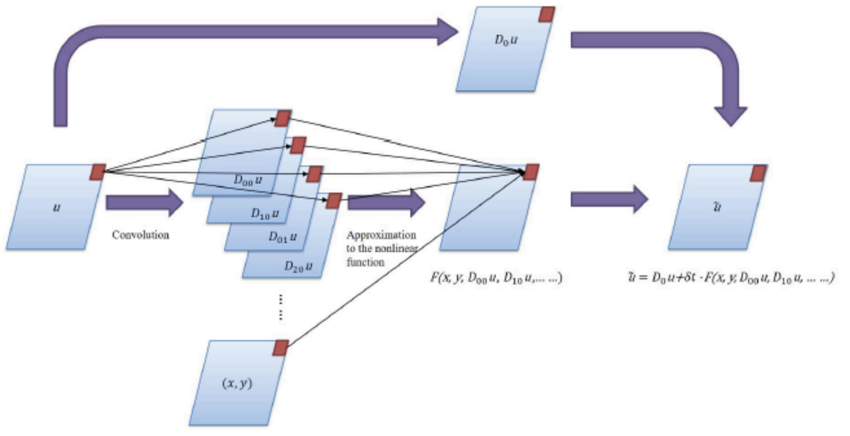


Figure 1: The schematic diagram of a δt-block.

δt block implementation

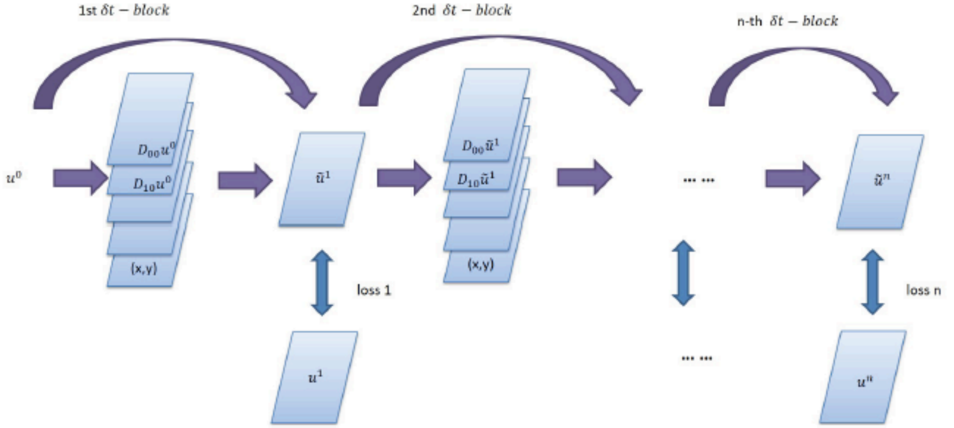


Figure 2: The schematic diagram of the PDE-Net: multiple δt-blocks.

Multiple steps procedure

Long, Zichao, et al. "PDE-net: Learning PDEs from data." *arXiv preprint arXiv:1710.09668* (2017)

PDE-NET

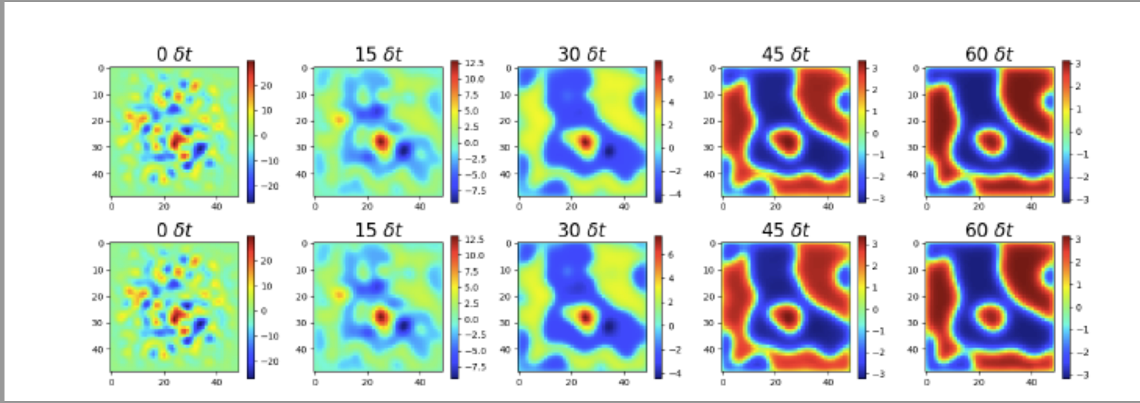
$$\begin{cases} \frac{\partial u}{\partial t} &= c\Delta u + f_s(u) \\ u|_{t=0} &= u_0(x,y), \end{cases} \quad \text{with } (t,x,y) \in [0, 0.2] \times \Omega,$$

$$f_s(u) = 15\sin(u)$$

- 50 x 50 mesh
- 7 x 7 filters
- 1,2k parameters in each δt-block

True

Prediction



Long, Zichao, et al. "PDE-net: Learning PDEs from data." *arXiv preprint arXiv:1710.09668* (2017)

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
  - PDE-NET
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

Inverse problems

Coupling forward and inverse problems

- Generalized Poisson equation: $$\begin{aligned} \mathcal{L}u &= 0, & x &\in \Omega \\ u(x) &= u_0(x), & x &\in \partial\Omega. \end{aligned}$$

  with $$\mathcal{L}u = \partial_i(a^{ij}(x)\partial_j u) + b^j(x)\partial_j u + c(x)u, \quad i,j = 1,\ldots,$$
- **Forward problem:** find u knowing a, b, c
- **Inverse problem:** find a (b, c) knowing a few measurements of u
- Use two neural networks, one for u and one for a
- input **x**, standard fully connected
- **Losses** (meshless approach):

  Forward $$\mathcal{F}(u) = \lambda\|\mathcal{L}u\|_2^2 + \mu\|\mathcal{L}u\|_\infty + \|u - u_0\|_{1,\partial\Omega} + \mathcal{R}^F(u)$$

  Inverse $$\mathcal{I}(a^{ij}) = \lambda\|\mathcal{L}u\|_2^2 + \mu\|\mathcal{L}u\|_\infty + \|a^{ij} - a_0^{ij}\|_{1,\partial\Omega} + \mathcal{R}^I(a^{ij})$$

L. Bar et al., “Unsupervised Deep Learning Algorithm for PDE-based Forward and Inverse Problem”. ArXiv, april 2019.

Inverse problems

Coupling forward and inverse problems: results

- **Example**: Electrical Impedance Tomography (2D), Poisson equation
- Data (forward): random points on Ω for σ, on $\partial\Omega$ for u_0
- 4 layers (26, 26, 26, 10), Adam, bs 1000, decay learning rate
- Discretized loss:
  (Forward)

$$\mathcal{F}(u(x;w_u)) = \frac{\lambda}{N_s}\sum_{i=1}^{N_s}|\mathcal{L}_i|^2 + \frac{\mu}{K}\sum_{k\in\mathrm{top}_K(|\mathcal{L}_i|)}|\mathcal{L}_k| + \frac{1}{N_b}\sum_{b=1}^{N_b}\Big|u(x_b)-u_0(x_b)\Big| + \alpha\|w_u\|_2^2.$$

- Parameters: N_S=45000, N_b=1200, λ=0.01, $\alpha=10^{-8}$, K=40, $\mu=10^{-2}$
- Excellent results. In particular
  MSE = ($1.72\ 10^{-3}$, $1.22\ 10^{-3}$, $2.35\ 10^{-4}$) for 1, 2 or 3 boundary cond.

L. Bar et al., “Unsupervised Deep Learning Algorithm for PDE-based Forward and Inverse Problem”. ArXiv, april 2019.

Inverse problems

Coupling forward and inverse problems: results

- **Example**: Electrical Impedance Tomography (2D), Poisson equation
- Data (inverse): random points on Ω for u, on $\partial\Omega$ for u_0 and σ_0
- Discretized loss: (Inverse)

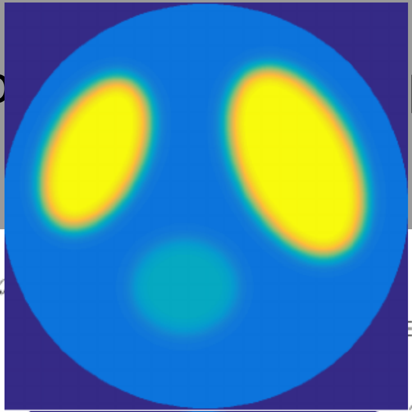
$$\mathcal{I}(\sigma(x;w_\sigma)) = \frac{\lambda}{N_s}\sum_{i=1}^{N_s}|\mathcal{L}_i|^2 + \frac{\mu}{K}\sum_{k\in \mathrm{top}_K(|\mathcal{L}_i|)}|\mathcal{L}_k| + \frac{1}{N_b}\sum_{b=1}^{N_b}\Big|\sigma(x_b)-\sigma_0(x_b)\Big| + \alpha\|w_\sigma\|_2^2 + \frac{\beta}{N_s}\sum_{i=1}^{N_s}|\nabla\sigma(x_i)|^p$$

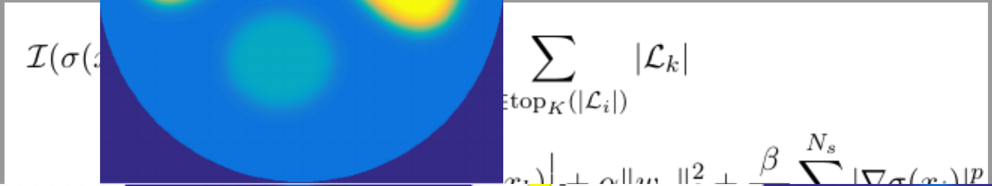
- Parameters: same, except $\beta=10^{-3}$, $\mu=10^{-2}$
- MSE = ($1.72\ 10^{-3}$, $1.22\ 10^{-3}$, $2.35\ 10^{-4}$) with 1, 2 or 3 measurements

L. Bar et al., “Unsupervised Deep Learning Algorithm for PDE-based Forward and Inverse Problem”. ArXiv, april 2019.

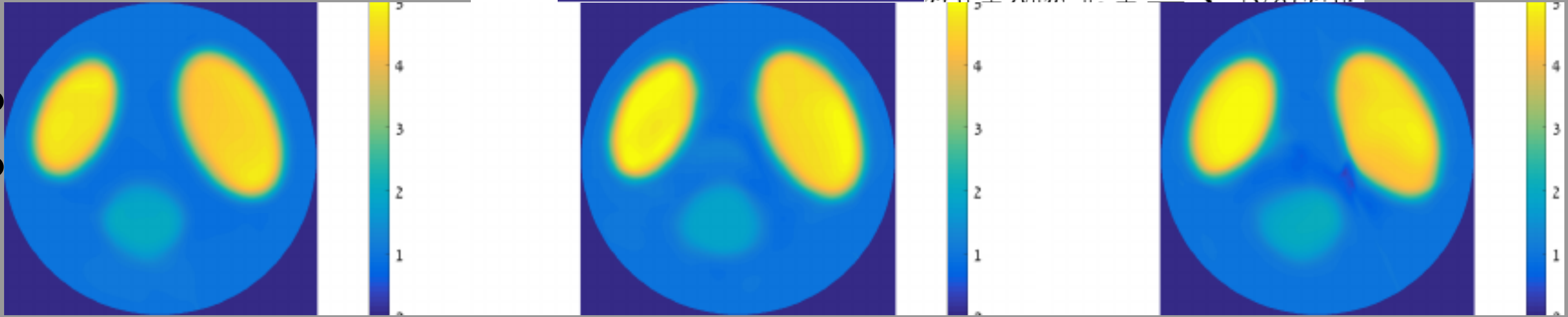
Inverse problems

Coupling forward and inverse problems: results

- **Example**: Electrical Impphy (2D), Poisson equation
- Data (inverse): random on $\partial\Omega$ for u_0 and σ_0
- Discretized loss:
  (Inverse)



-
-



L. Bar et al., “Unsupervised Deep Learning Algorithm for PDE-based Forward and Inverse Problem”. ArXiv, april 2019.

ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, ...)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

Data-driven Parametric Model Identification

Dictionary-based Learning

$$u_t = \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots)$$

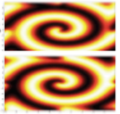
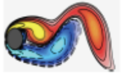
- Sparse optimisation from a set of primitives

$$\begin{aligned}
\mathcal{N}(t, x, u, u_x, u_{xx}, \ldots) \quad = \quad & \alpha_{0,0} + \alpha_{1,0}u + \alpha_{2,0}u^2 + \alpha_{3,0}u^3 + \\
& \alpha_{0,1}u_x + \alpha_{1,1}uu_x + \alpha_{2,1}u^2u_x + \alpha_{3,1}u^3u_x + \\
& \alpha_{0,2}u_{xx} + \alpha_{1,2}uu_{xx} + \alpha_{2,2}u^2u_{xx} + \alpha_{3,2}u^3u_{xx} + \\
& \alpha_{0,3}u_{xxx} + \alpha_{1,3}uu_{xxx} + \alpha_{2,3}u^2u_{xxx} + \alpha_{3,3}u^3u_{xxx}
\end{aligned}$$

- Pros: intelligibility
- Cons:
  - Numerical differentiation is unstable
  - Completeness of the set of primitives

S.L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems” PNAS, 113(15):3932–3937, 2016

Data-driven Parametric Model Identification

PDE	Form	Error (no noise, noise)	Discretization
 KdV	$u_t + 6uu_x + u_{xxx} = 0$	$1\%\pm0.2\%, 7\%\pm5\%$	$x\in[-30, 30], n{=}512,\ t\in[0, 20], m{=}201$
 Burgers	$u_t + uu_x - \epsilon u_{xx} = 0$	$0.15\%\pm0.06\%, 0.8\%\pm0.6\%$	$x\in[-8, 8], n{=}256,\ t\in[0, 10], m{=}101$
 Schrödinger	$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0$	$0.25\%\pm0.01\%, 10\%\pm7\%$	$x\in[-7.5, 7.5], n{=}512,\ t\in[0, 10], m{=}401$
 NLS	$iu_t + \frac{1}{2}u_{xx} + \|u\|^2u = 0$	$0.05\%\pm0.01\%, 3\%\pm1\%$	$x\in[-5, 5], n{=}512,\ t\in[0, \pi], m{=}501$
 KS	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$	$1.3\%\pm1.3\%, 70\%\pm27\%$	$x\in[0, 100], n{=}1024,\ t\in[0, 100], m{=}251$
u v  Reaction Diffusion	$u_t = 0.1\nabla^2 u + \lambda(A)u - \omega(A)v$ $v_t = 0.1\nabla^2 v + \omega(A)u + \lambda(A)v$ $A^2 = u^2 + v^2, \omega{=}-\beta A^2, \lambda{=}1-A^2$	$0.02\% \pm 0.01\%,\ 3.8\% \pm 2.4\%$	$x, y\in[-10, 10], n{=}256,\ t\in[0, 10], m{=}201$ subsample 1.14%
 Navier Stokes	$\omega_t + (\mathbf{u}\cdot\nabla)\omega = \frac{1}{Re}\nabla^2\omega$	$1\% \pm 0.2\%$, $7\% \pm 6\%$	$x\in[0, 9], n_x{=}449,\ y\in[0, 4], n_y{=}199,$ $t\in[0, 30], m{=}151$, subsample 2.22%

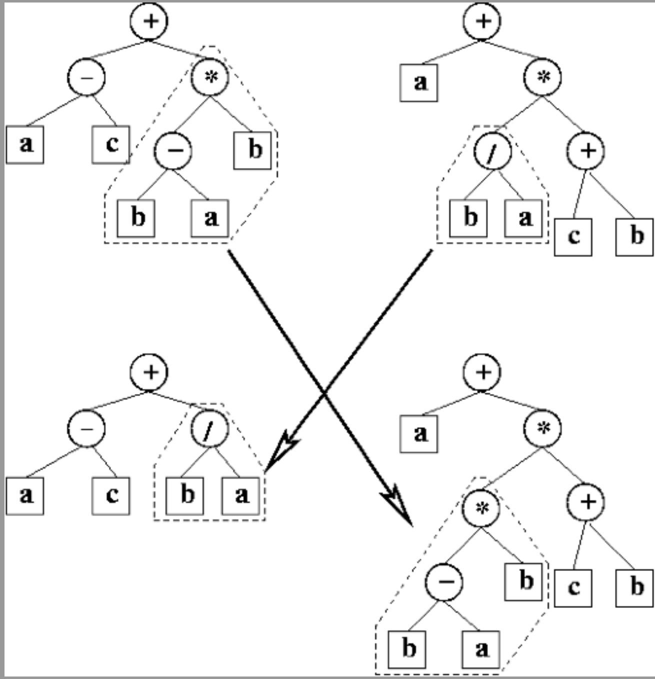
Rudy, Samuel H., et al. "Data-driven discovery of partial differential equations." *Science Advances* 3.4 (2017): e1602614.

Data-driven Free-form Model Identification

Genetic Programming

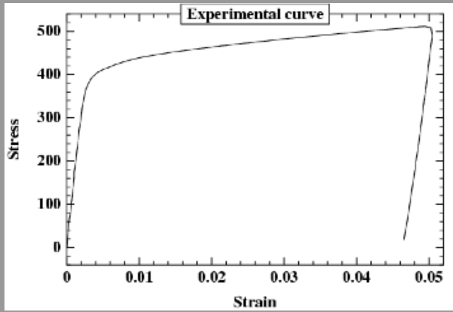
- An Evolutionary Algorithm
- evolving tree structures
- representing programs, functions, …

- Can explore huge unstructured search spaces
- and discover innovative solutions
- without a template



Data-driven Free-form Model Identification

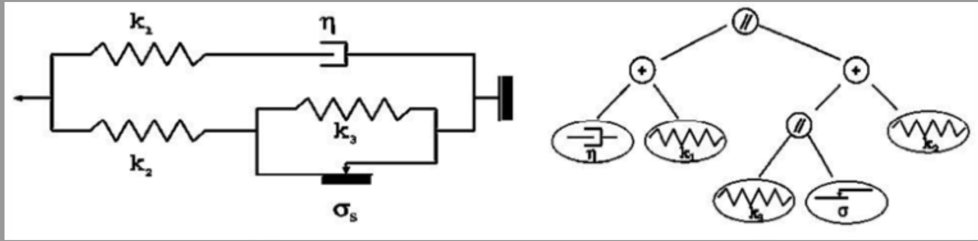
Identification of 1-D rheological models



- Data: Strain measures at discrete time steps

- Use of GP for rheological models

$$\sigma_{sim}(t) = \mathcal{F}(\epsilon(t), \dot{\epsilon}(t); k_1, k_2, k_3, \eta, \sigma_S)$$



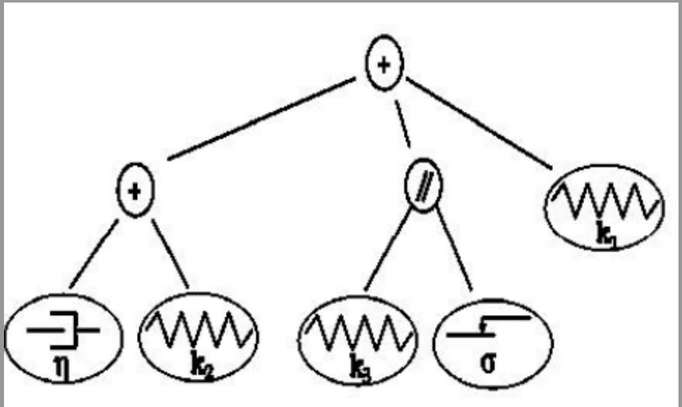
Difficulties

- Need an **interpreter** of rheological models
- Computational cost: gradually take into account the experiments

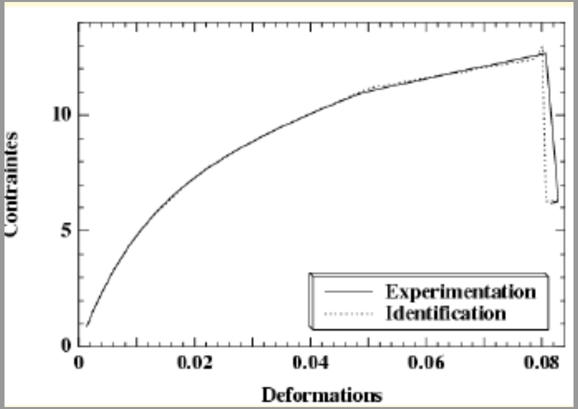
M.Schoenauer, M.Sebag, F.Jouve, B.Lamy, H.Maitournam. “Evolutionary identification of macro-mechanical models”. in *Advances in Genetic Programming II*, MIT Press, pp.467-488, 1996.

Data-driven Free-form Model Identification

Identification of 1-D rheological models: Results



Frequently identified model



with its "experimental" curve

- → **Active Learning**: lack of creep in the experiment
- Also, identification of 3D hyper-elastic law

physical constraints issues

M.Schoenauer, M.Sebag, F.Jouve, B.Lamy, H.Maitournam. "Evolutionary identification of macro-mechanical models". in *Advances in Genetic Programming II*, MIT Press, pp.467-488, 1996.

Data-driven Free-form Model Identification

Learning Dynamical Systems with Genetic Programming

- Direct identification of dynamical systems from time series
- with several computational tricks
    - Partitioning the variables
    - Snipping (anti-bloat)
    - Active learning (unrealistic)

- Good results on synthetic and real systems
- but many trials were unsuccessful

Single pendulum	
Target	$d\theta/dt = \omega$
	$d\omega/dt = -9.8\sin(\theta)$
Best model	$d\theta/dt = \omega$
	$d\omega/dt = -9.79987\sin(\theta)$
Median model	$d\theta/dt = \omega$
	$d\omega/dt = -9.8682\sin(\theta)$
Lotka–Volterra interspecific competition between two species	
Target	$dx/dt = 3x - 2xy - x^2$
	$dy/dt = 2y - xy - y^2$
Best model	$dx/dt = 3.0014x - 2xy - x^2$
	$dy/dt = 2.0001y - xy - y^2$
Median model	$dx/dt = 2.9979x - 2.0016xy - x^2$
	$dy/dt = 1.999y - 0.917xy - 1.005y^2$

J. Bongard and H. Lipson “Automated reverse engineering of nonlinear dynamical systems” PNAS, 104(24):9943–9948, 2007.

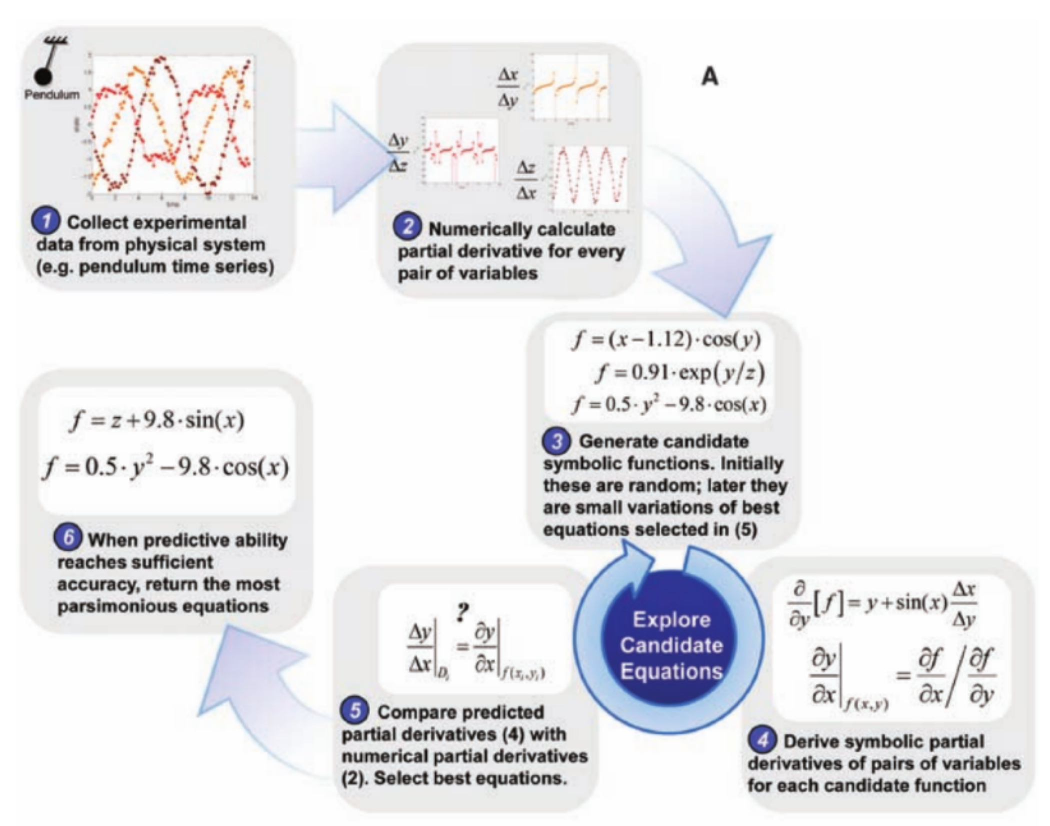
Data-driven Free-form Model Identification

- Identifies invariants
- from videos of experiments

- To avoid trivial invariants
  - check partial derivatives
  - for all pairs of variables
  - w.r.t. numerical derivatives

- Generate Pareto front
  - accuracy vs parsimony

- Keep best-looking equations :-)



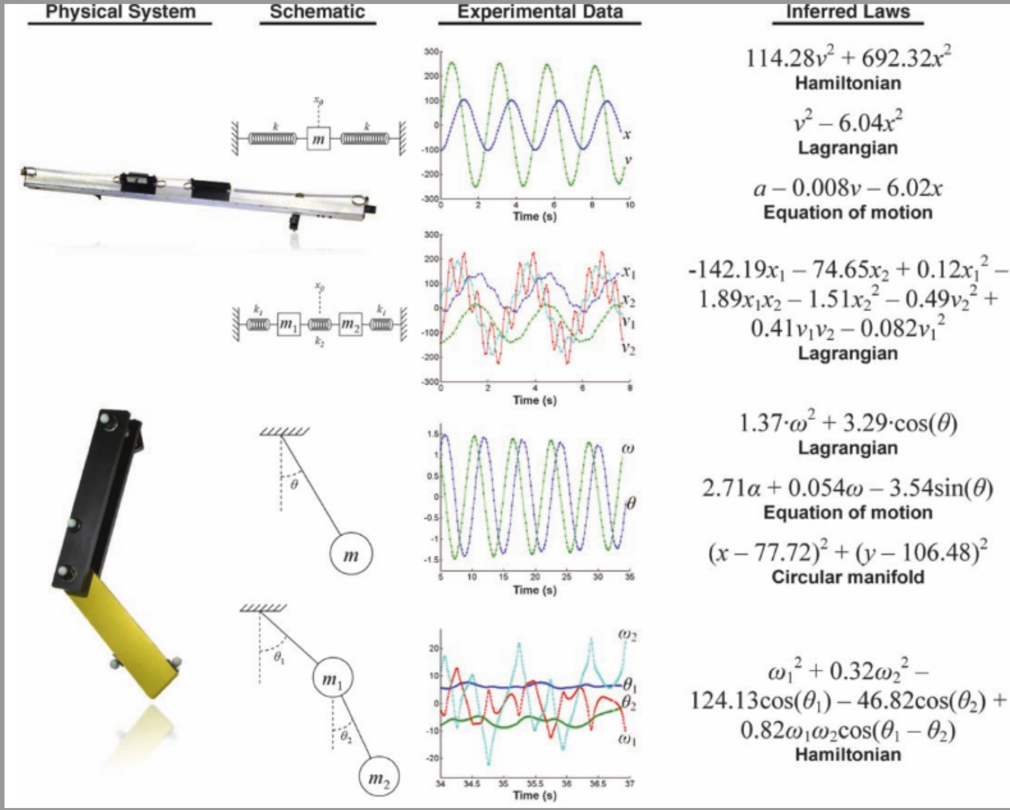
M. Schmidt and H. Lipson “Distilling free-form natural laws from experimental data” Science, 324(5923):81–85, 2009.

Data-driven Free-form Model Identification

- Impressive results
- Units of constants by varying the parameters (e.g., mass)

- But requires human knowledge
  - Choice of variables
  - Choice of operators
  - Choice of Pareto solution

- and GP poorly scales up



M. Schmidt and H. Lipson “Distilling free-form natural laws from experimental data” Science, 324(5923):81–85, 2009.

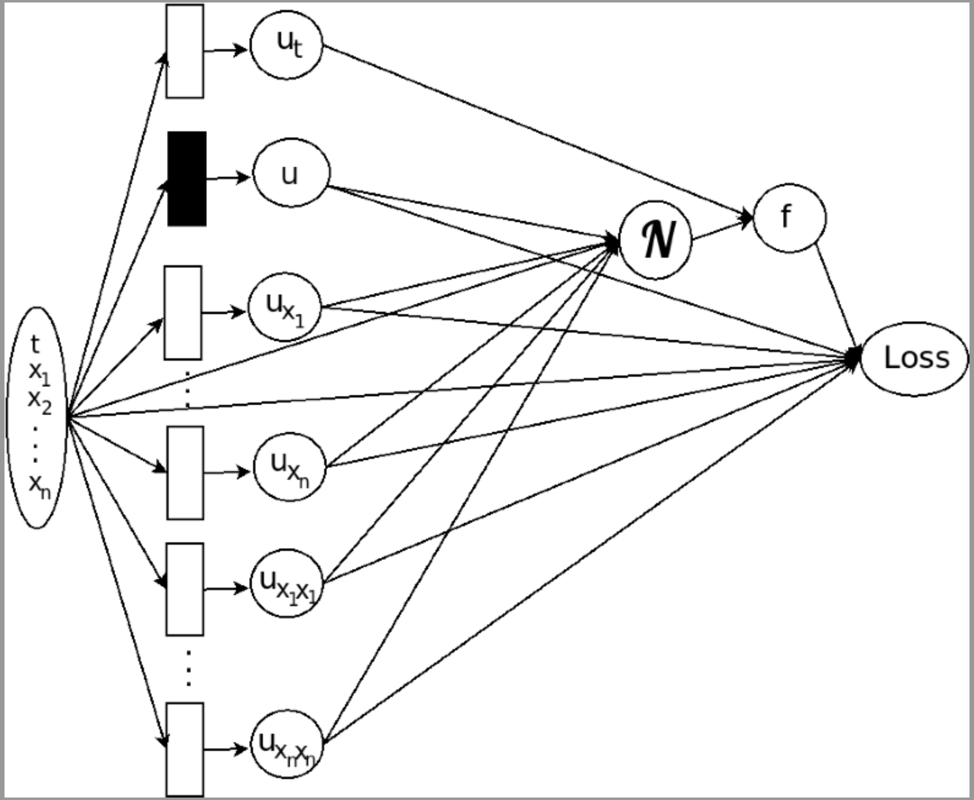
ML and PDEs: Agenda

- Simulation is fine, but huge output data
  - DL for Data Analytics (Climate, Particle Physics, …)
- Simulation is fine, but
  - DL as surrogate model (whole simulation, or sub-components)
  - Physic Informed Deep Learning
  - Deep Galerkin Method (high dimensions)
- Inverse problems / calibration
  - Often ill-posed
- Mechanistic model is unknown
  - Learn an analytical model from data
  - Learn a black-box model from data

Data-driven Black-box Model Identification

Identifying both the model and the solution

from

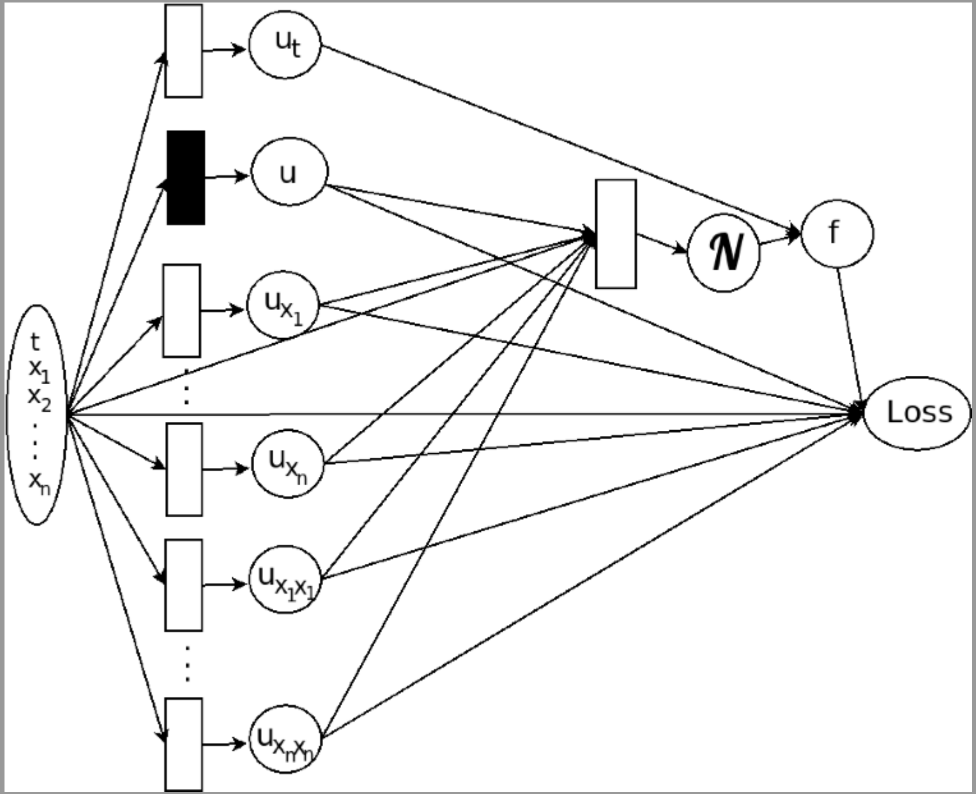


M. Raissi. "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations", JMLR 19:1-24, 2018

Data-driven Black-box Model Identification

Identifying both the model and the solution

to



M. Raissi. "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations", JMLR 19:1-24, 2018

Data-driven Model Identification

Identifying both the model and the solution (2)

$$u_t = \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots)$$

- A Deep Network for both u and N
- Deduced DN for $$f := u_t - \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots)$$
- Sample (many) data points (t_i, x_i, u_i) i=1,..., N
- Goal: minimize $$\sum_{i=1}^{N} \left(|u(t^i, x^i) - u^i|^2 + |f(t^i, x^i)|^2 \right)$$

M. Raissi, "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations", JMLR 19:1-24, 2018

Data-driven Model Identification

Identifying both the model and the solution (3)

Example: Burgers' equation $u_t = -uu_x + 0.1u_{xx}$

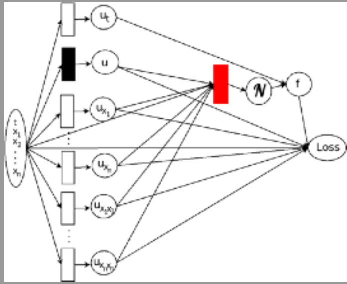
- DN_u: 5 layers, 50 neurons/layer, sine activation function
- DN_N: 2 layers, 100 neurons/layer, sine a.f.

- "**Exact**" solution from 4th order Runge-Kutta, time step 10^{-4}
- **Examples**: 201 snapshots in time
- Train set: 10000 random points for $t \in [0,6.67]$
- Tested for $t \in [6.67,10]$

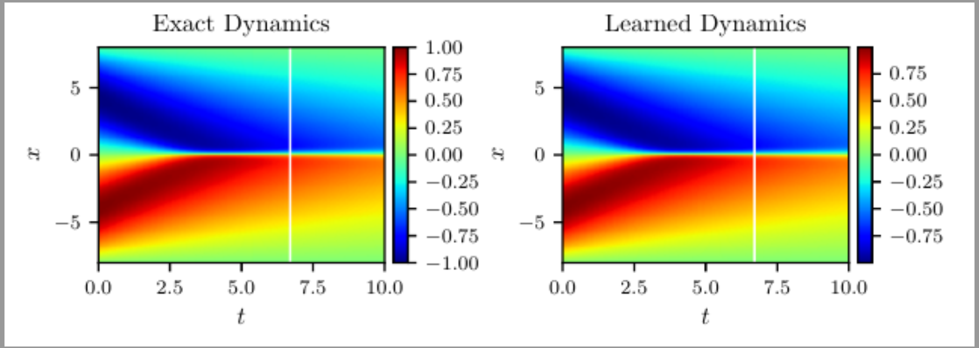
	Clean data	1% noise	2% noise	5% noise
Relative L^2-error	4.78e-03	2.64e-02	1.09e-01	4.56e-01

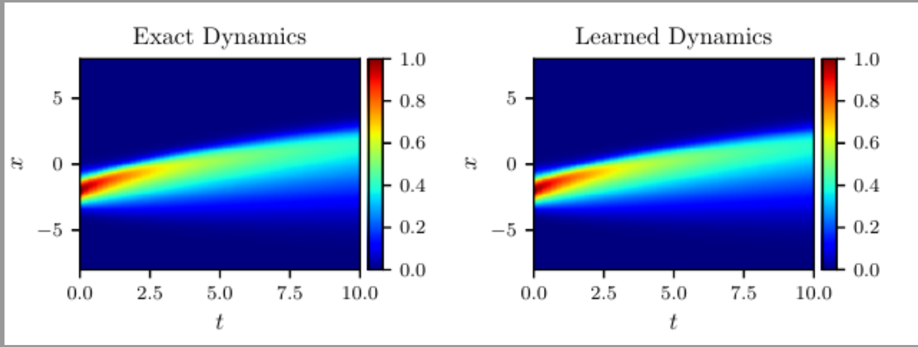
M. Raissi. "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations", JMLR 19:1-24, 2018

Data-driven Model Identification



Assessing the learned dynamics





On the learned equation

$$u(0,x) = -\sin(\pi x/8)$$

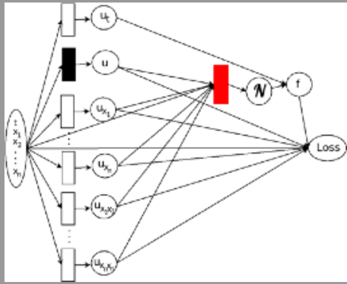
with a different initial condition

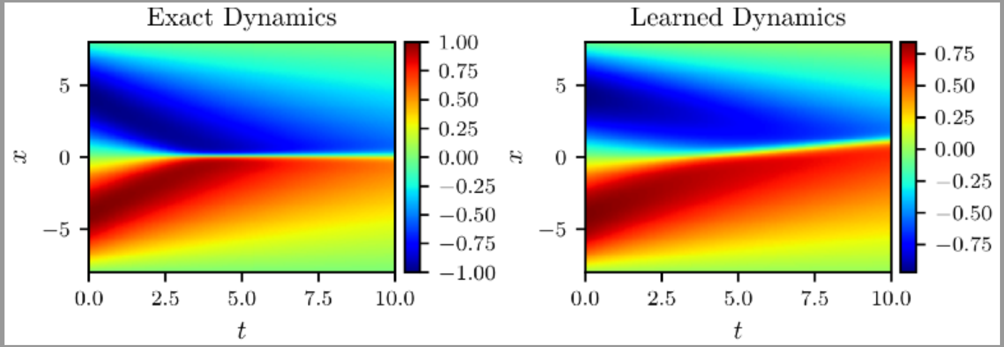
$$u(0,x) = -\exp(-(x+2)^2)$$

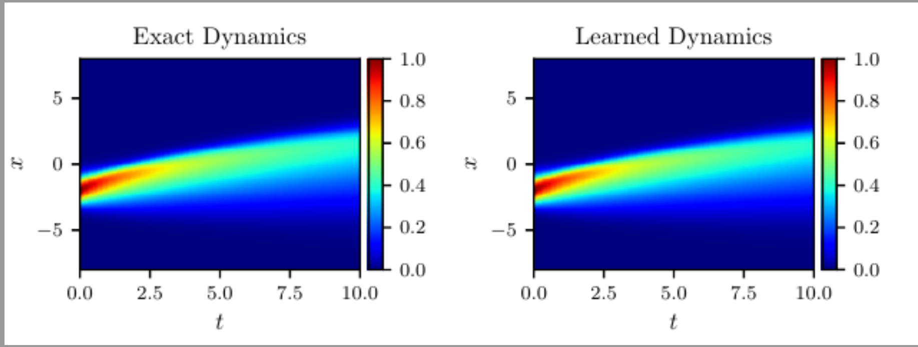
M. Raissi. “Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations”, JMLR19:1-24, 2018

Data-driven Model Identification

Assessing the learned dynamics (2)







with a different initial condition

$$u(0, x) = -\sin(\pi x/8)$$

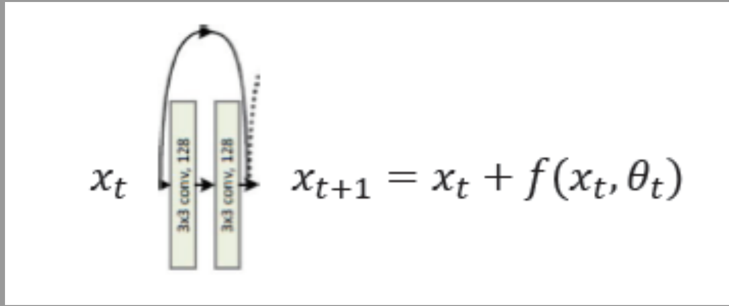
On the learned equation

$$u(0, x) = -\exp(-(x+2)^2)$$

M. Raissi. "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations", JMLR19:1-24, 2018

From PDE to NNs

- ResNet module: x is progressively modified by the residual $f(x, \theta)$

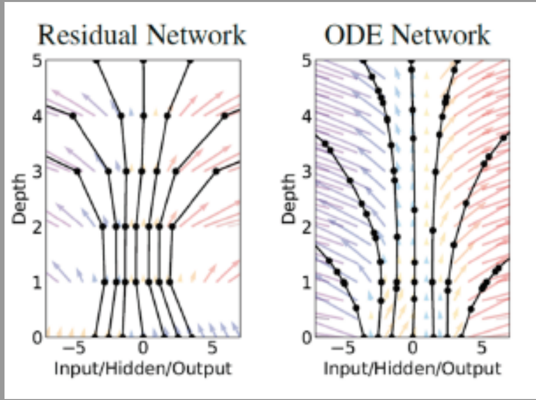


$$x_{t+1} = x_t + hf(x_t, \theta_t)$$

for small h this is the forward **Euler scheme**

Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." *T*2017.

- ODEnet:



Optimization problem:

$$\min_{\theta} L(F(x, \theta), y) \text{ s.t. } \dot{x} = F(x(t), \theta)$$

use **Lagrangian** optimization

$$L(F(x(T), \theta), y) - \int_0^T \lambda(t)(\dot{x} - F(x(t), \theta))$$

Chen, Tian Qi, et al. "Neural ordinary differential equations." *Advances in neural information processing systems*. 2018.

Conclusions

- Some impressive results (even if on small regular problems)
  - Synergy with HPC
  - Surrogate modeling
  - Meshless simulations
- Still underexploited
  - The generative power of DNNs (GANs)
  - Transfer learning and domain adaptation (DANNs)
  - Graph networks
- Open issues
  - Where do the data come from?
  - How noisy are they?
  - Small data: PDEs as constraints?

How to hybridize Machine Learning and Mechanistic Knowledge?