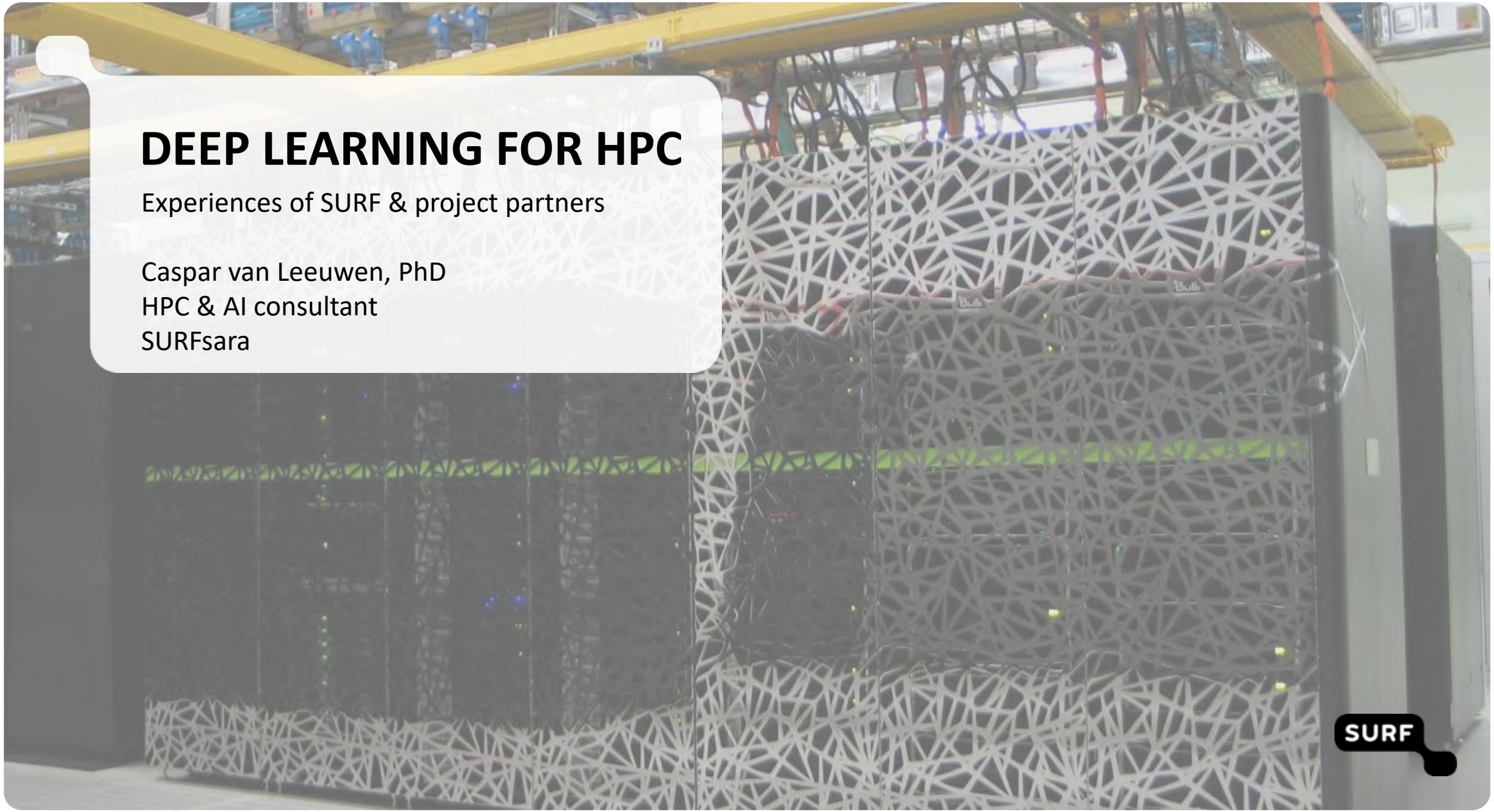


DEEP LEARNING FOR HPC

Experiences of SURF & project partners

Caspar van Leeuwen, PhD
HPC & AI consultant
SURFsara

Disclaimer

This is a workshop, *not* a training.

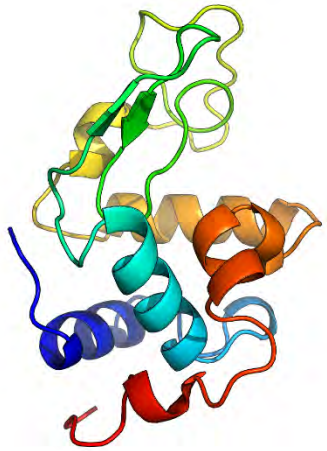
- Deep learning for HPC is a very new field of research
- We don’t have the solutions, we want to help you find them!

Overview

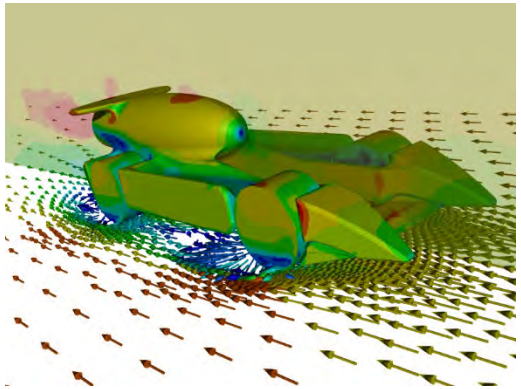
- What is deep learning for HPC?
- Deep learning for HPC projects @ SURF
- How to replace/augment an HPC simulation with deep learning?
- Conlusions
- Round tables

What is high performance computing (HPC)?

Chemistry

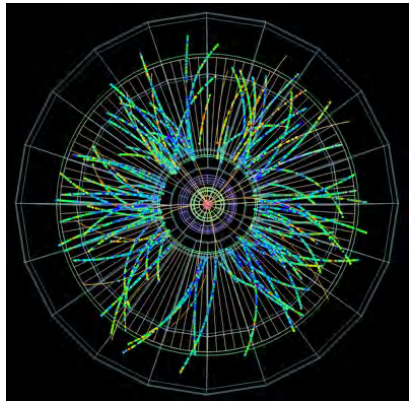


Complex Fluid Dynamics



Astrophysics

Particle physics



Meteorology

and more …

Sources: J.A. Lemkul, http://www.mdtutorials.com/gmx/ ; Symscape http://www.symscape.com

What is deep learning for HPC?

Usage of deep learning (DL) to

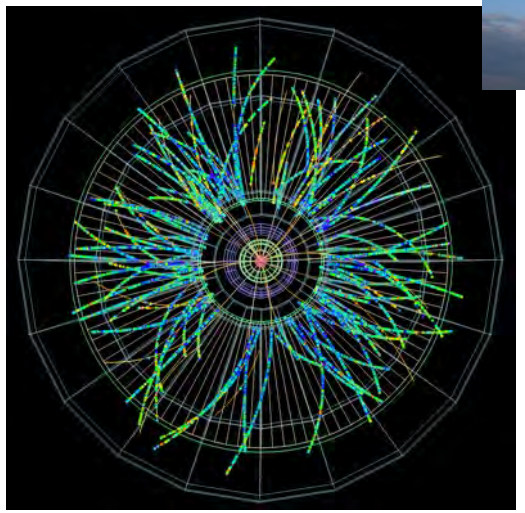
- Accelerate traditional HPC workloads / simulations (at same accuracy)
- Improve accuracy of traditional HPC workloads / simulations (at same computational cost)

Deep learning projects @ SURF

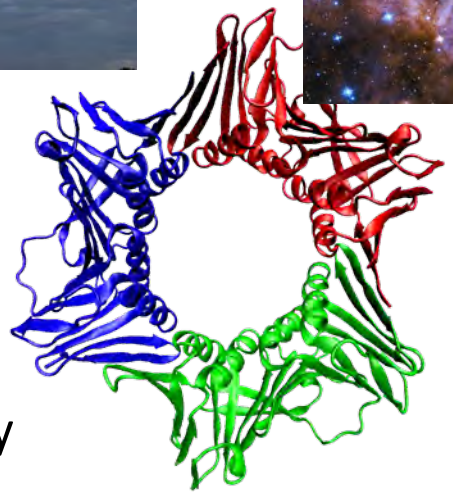
Meteorology

Astrophysics

Particle physics



Computational structural biology



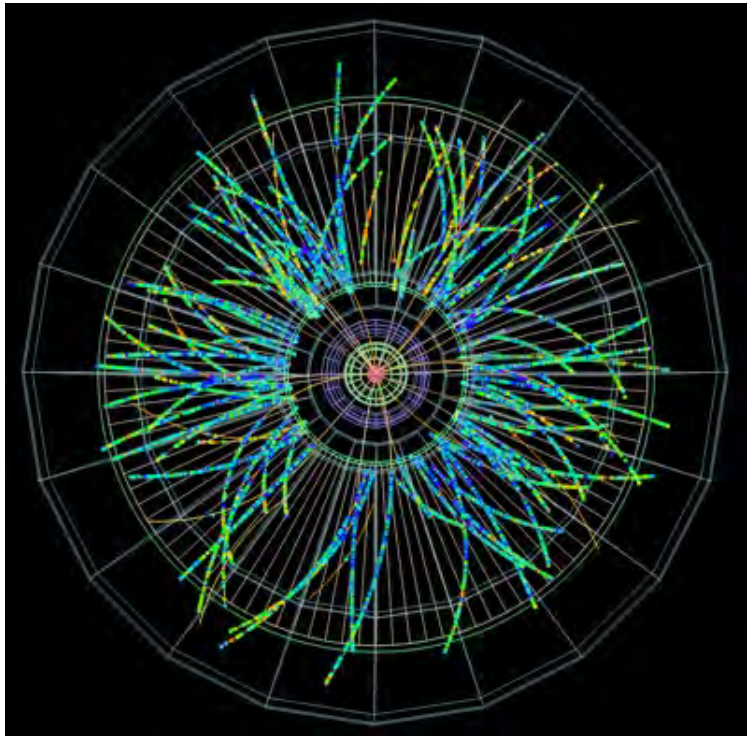
Particle physics: event generation

Radboud University: Sydney Otten, Sascha Caron (PI), *et al*

SURF: Damian Podareanu, Caspar van Leeuwen

Traditional simulation

- Generating particle events through Monte Carlo simulation
- Requires repeated sampling of probability distribution
- Simulation may take up to O(10) minutes for realistic LHC events![*]

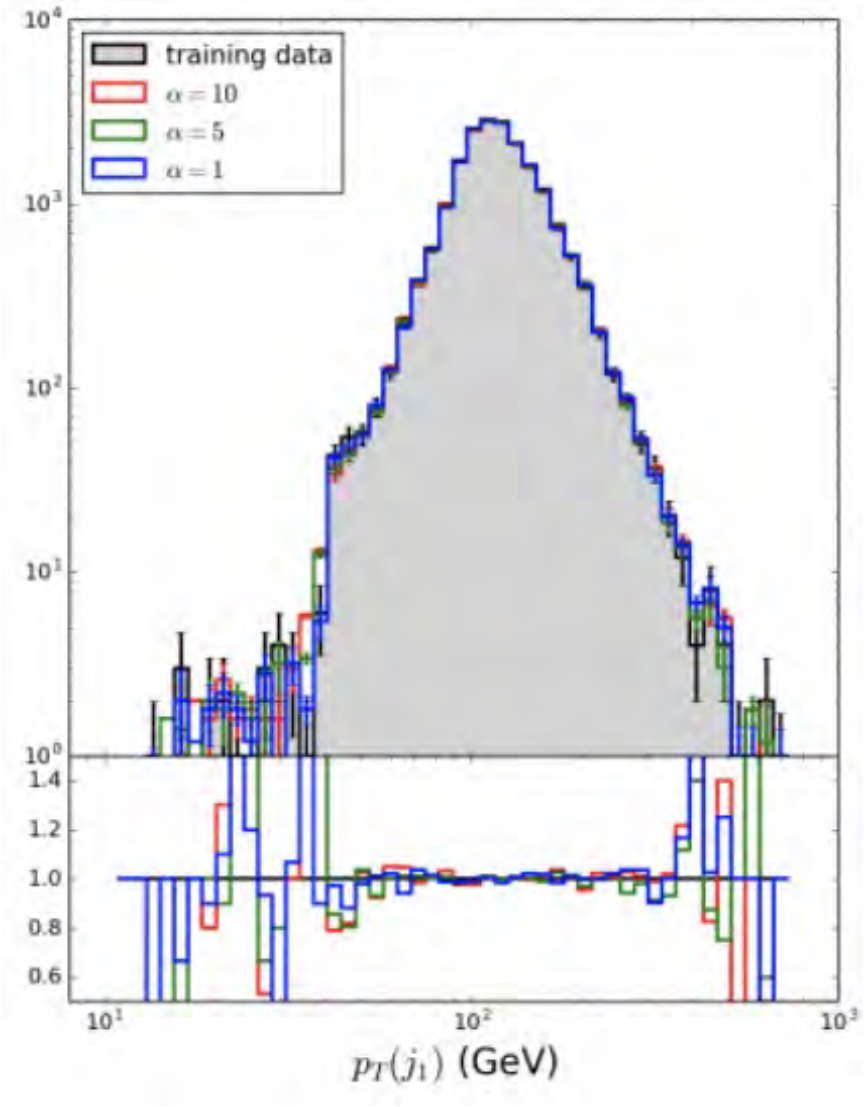


[*]Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer
S. Otten, S. Caron, W. de Swart, *et al*, arXiv, 2019

Particle physics: event generation

Deep learning approach:

- Model: Complete simulation replaced with generative neural network (B-VAE)*
- Validation: Physical parameters follow the same distribution as for the original simulation
- Speedup: O(10^8) faster



*Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer
S. Otten, S. Caron, W. de Swart, *et al*, arXiv, 2019

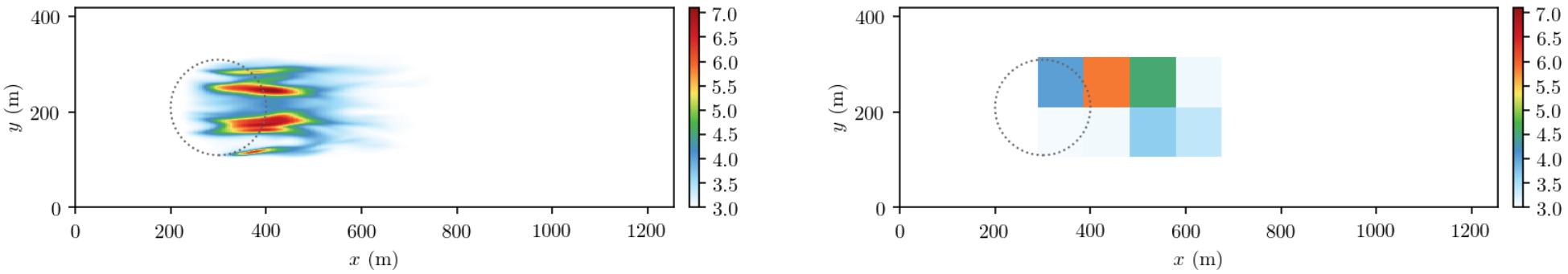
Meteorology

Wageningen University: Robin Stoffer, Menno Veerman, Chiel van Heerwaarden (PI)

SURF: Damian Podareanu, Caspar van Leeuwen

Traditional simulation

- Sub-grid model to resolve turbulent transport at scales < grid spacing
- Good sub-grid models are expensive
- Faster sub-grid models may not be very accurate or have strong assumptions

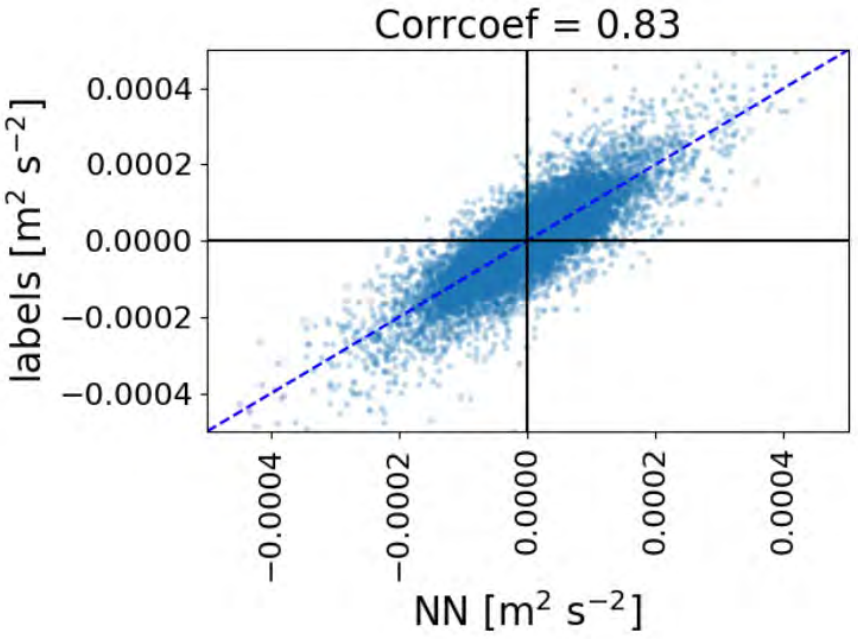


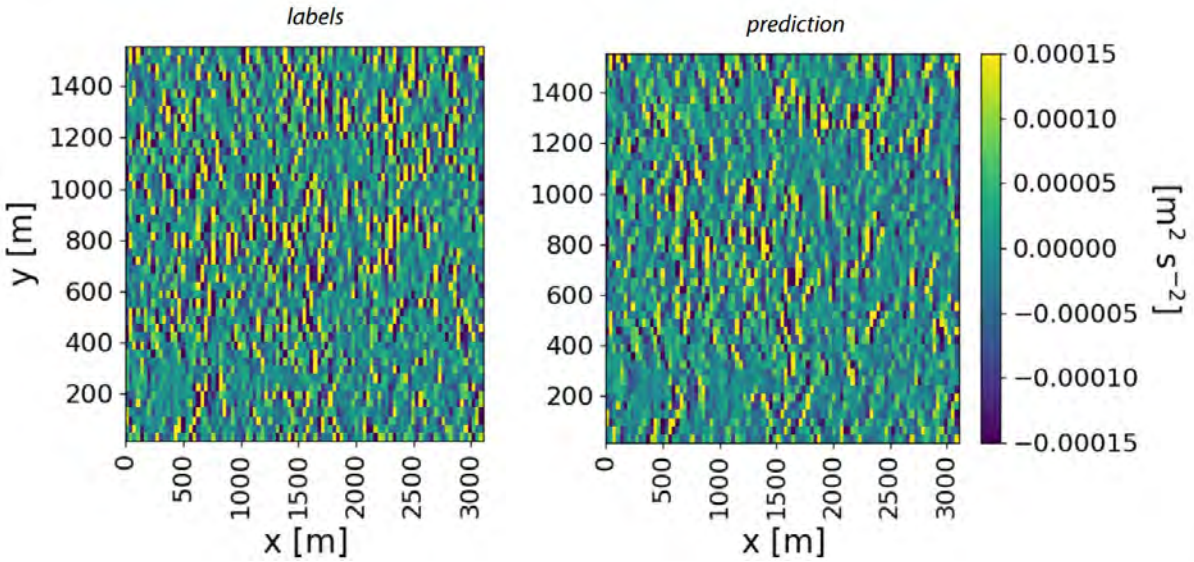
Example: Water evaporation & transport above an irrigated field, high vs low resolution

Meteorology

Deep learning approach:

- Model: Sub-grid model replaced by neural network, rest of the traditional simulation remains intact
- Validation: Correlation of 0.7 - 0.8 is excellent for stochastic quantity
- Speedup: to be determined, but main goal is increased accuarcy

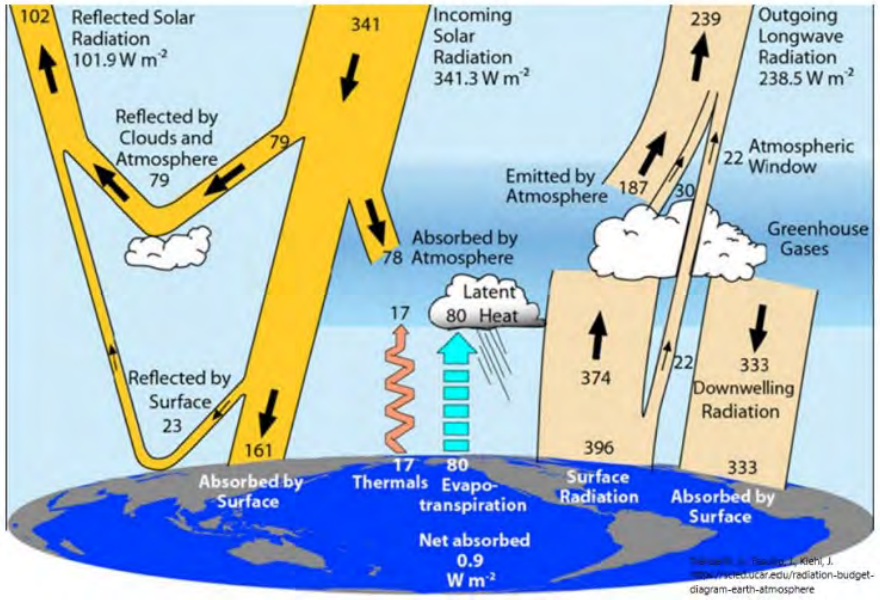


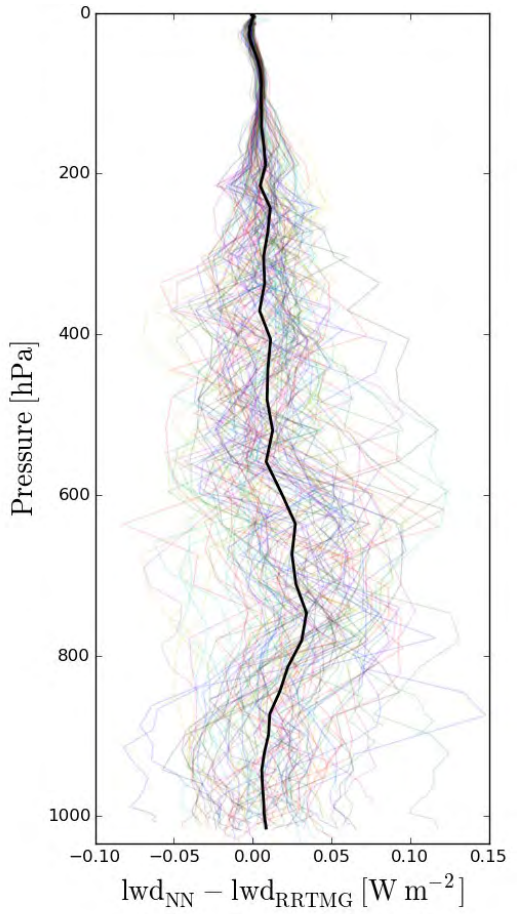


Meteorology

2nd project: similar approach for radiation transport

- Traditional model: estimates physical properties (e.g. optical depth) based on air composition, pressure, etc
- Deep learning approach: neural network with the same inputs
- Validation: difference with original parametric model negligible
- Speedup: At least 3 times faster





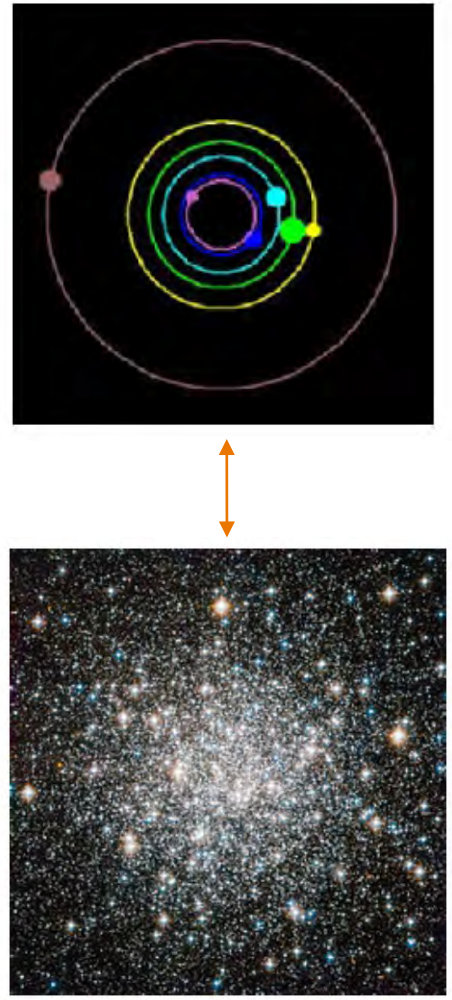
Astrophysics

Leiden Observatory: Maxwell Cai, Simon Portegies Zwart (PI)

SURF: Maxwell Cai, Valeriu Codreanu, Caspar van Leeuwen

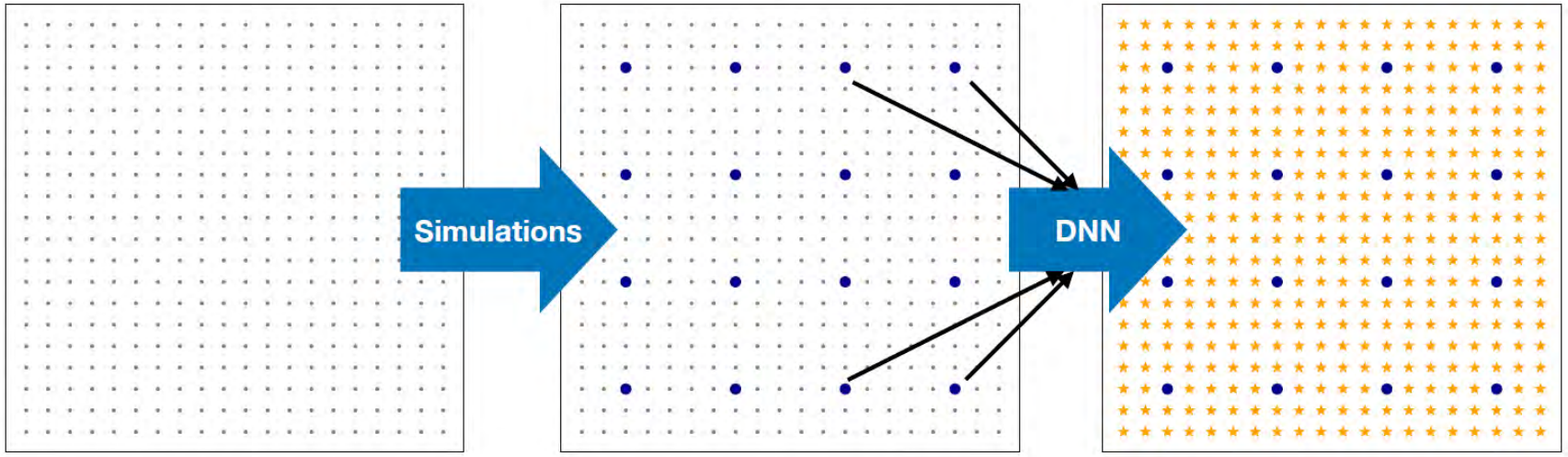
Traditional simulation

- Numerical integration of planetary systems & containing star cluster
- Large difference in scale: computationally intensive!
- Many ways to map the problem to deep learning…



Astrophysics

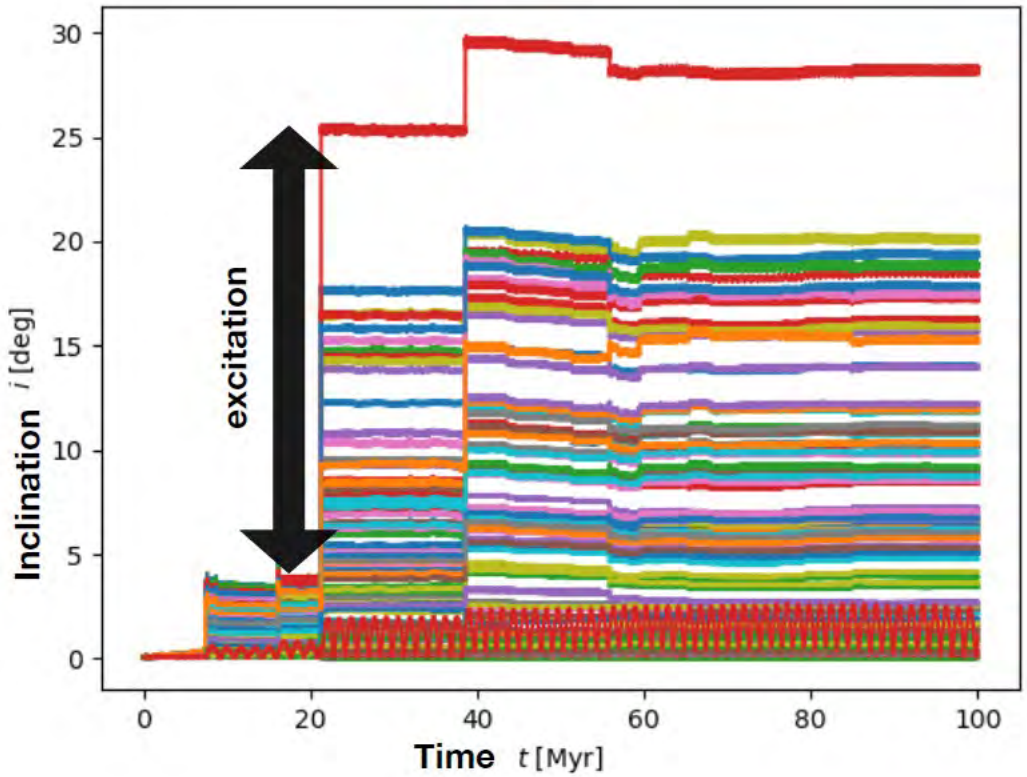
- Many starting parameters to vary (e.g. planetary masses, stellar masses)
- Simulation can be done, but only for a couple of starting parameters
- Use DNN to predict for other points in the parameter space
- Can be any type of DNN: we've tried many!



Astrophysics

As a time series (LSTM)

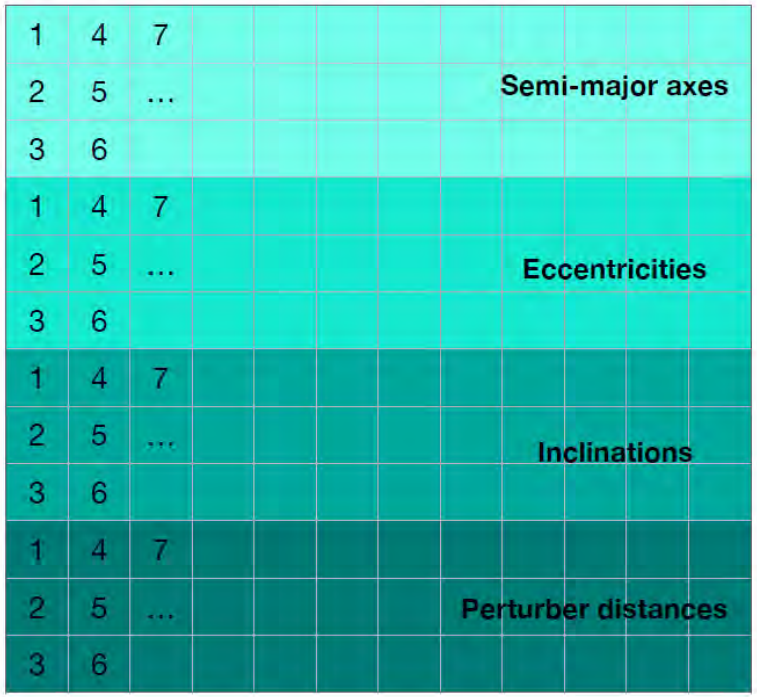
- Physical parameters (position, velocity, etc) as function of time
- Can only forecast for short time scales (for non-linear systems), few kyr



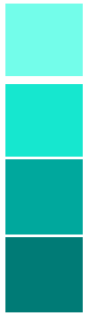
Astrophysics

As pattern recognition problem

- Physical parameters as function of time organized as array
- Regression to predict change in next time step
- Can predict further into the future (up to 1 Myr), but not further



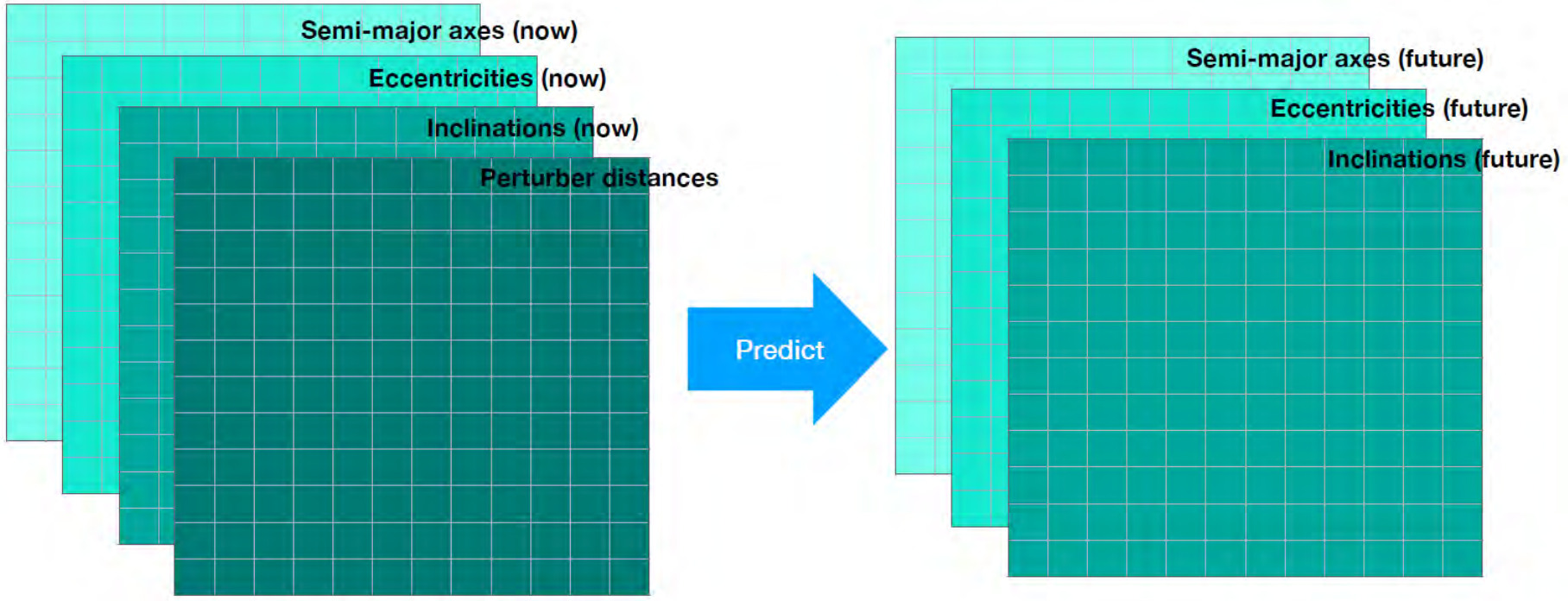
Regression: predict for next 1 Myr



Astrophysics

As image-to-image translation problem

- Physical parameters as function of time organized as array
- Image-to-image translation to predict whole series of future time properties
- Will accumulate errors, so again, limited prediction into the future is possible



Astrophysics

As reinforcement learning problem

- Reinforcement learning: an agent acting in a changing environment to optimize a reward
- A planet (agent) moving (acting) under the influence of a nearby star (environment) following a path as close as possible to the original simulation (reward)
- Error accumulation in time is mitigated
- Works ok for individual systems, but doesn’t result in correct distribution

Astrophysics

Solutions enabled further and further prediction in the future, but problem not fully solved yet.

Why is this problem so difficult?

- Underlying systems chaotic
- High dynamic range
- Extremely imbalanced training samples (significant ‘jumps’ in the time series)
- Extremely long term prediction needed

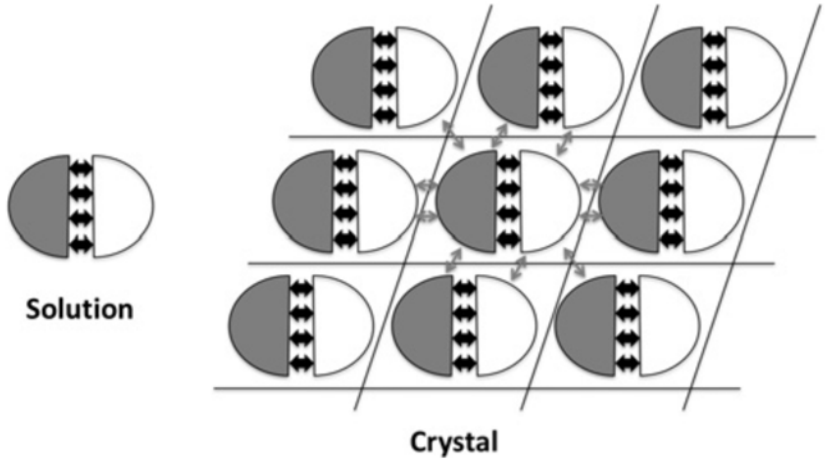
Computational Structural Biology

Utrecht University: Cunliang Geng, Alexandre Bonvin (PI)

SURF: Valeriu Codreanu, Caspar van Leeuwen

Traditional problem: distinguish biological interfaces from crystallographic interfaces

- Traditional model: classical classification approaches
- Deep learning: 3D convolutional classification task for deep learning
- Validation: accuracy quickly on par with traditional methods



doi:10.1042/BST0361438

How to replace/augment an HPC simulation with deep learning?

There is no fixed ‘recipe’! Let’s try to provide some structure:

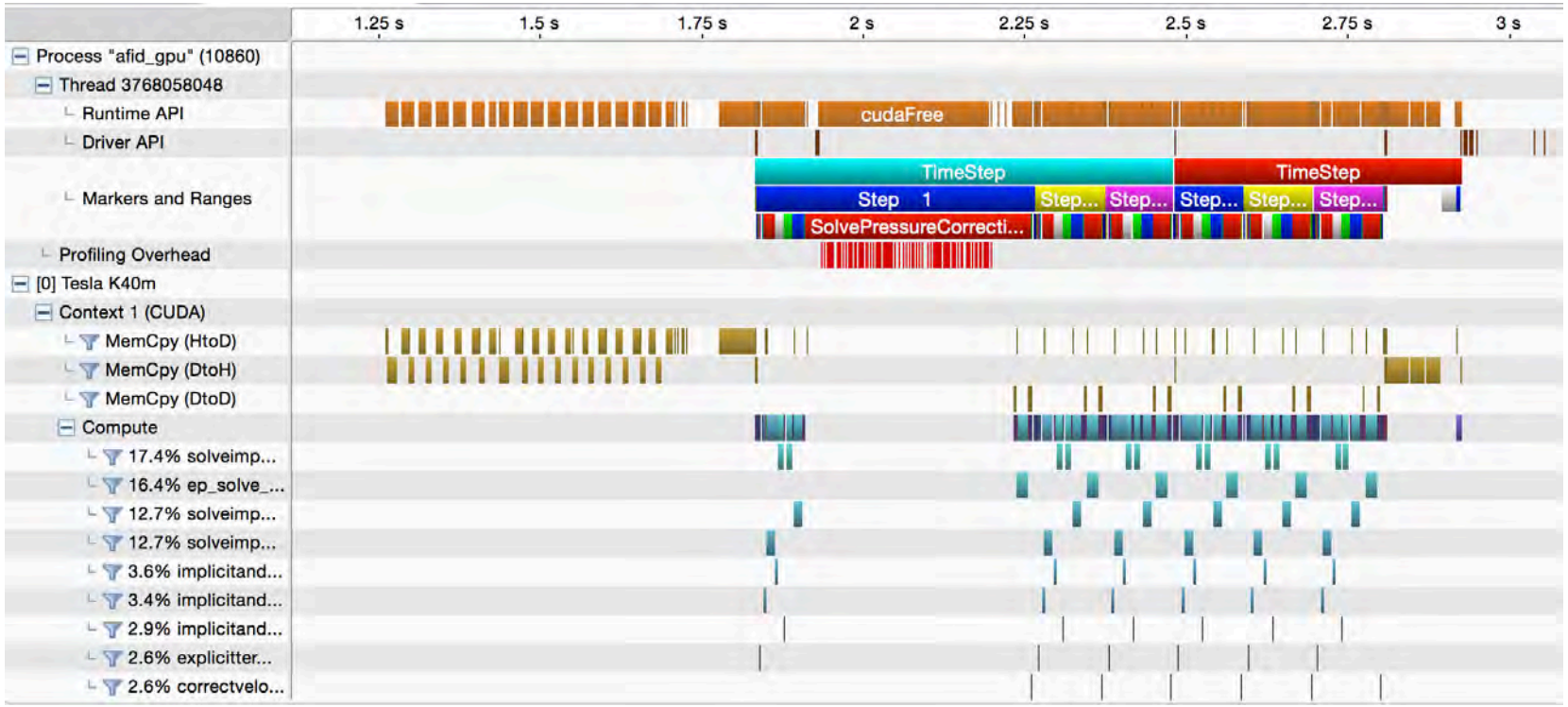
- Why is my traditional simulation computationally heavy?
- How will I enhance/speedup my code?
- How do I encode my problem?
- What deep learning model fits that encoding naturally?
- How will I validate my trained model?
- Try it! You may have too loop through these questions multiple times as you go…
- Improve inference speed

Deep learning for HPC is *pioneering*. Lot of exploration and adaptation needed. You may need to loop through the above questions multiple times.

Why is my traditional simulation computationally heavy?

Profile original application

- Identify if there are heavy kernels, that are responsible for the majority of computation time



How will I enhance/speedup my code?

Augmentation

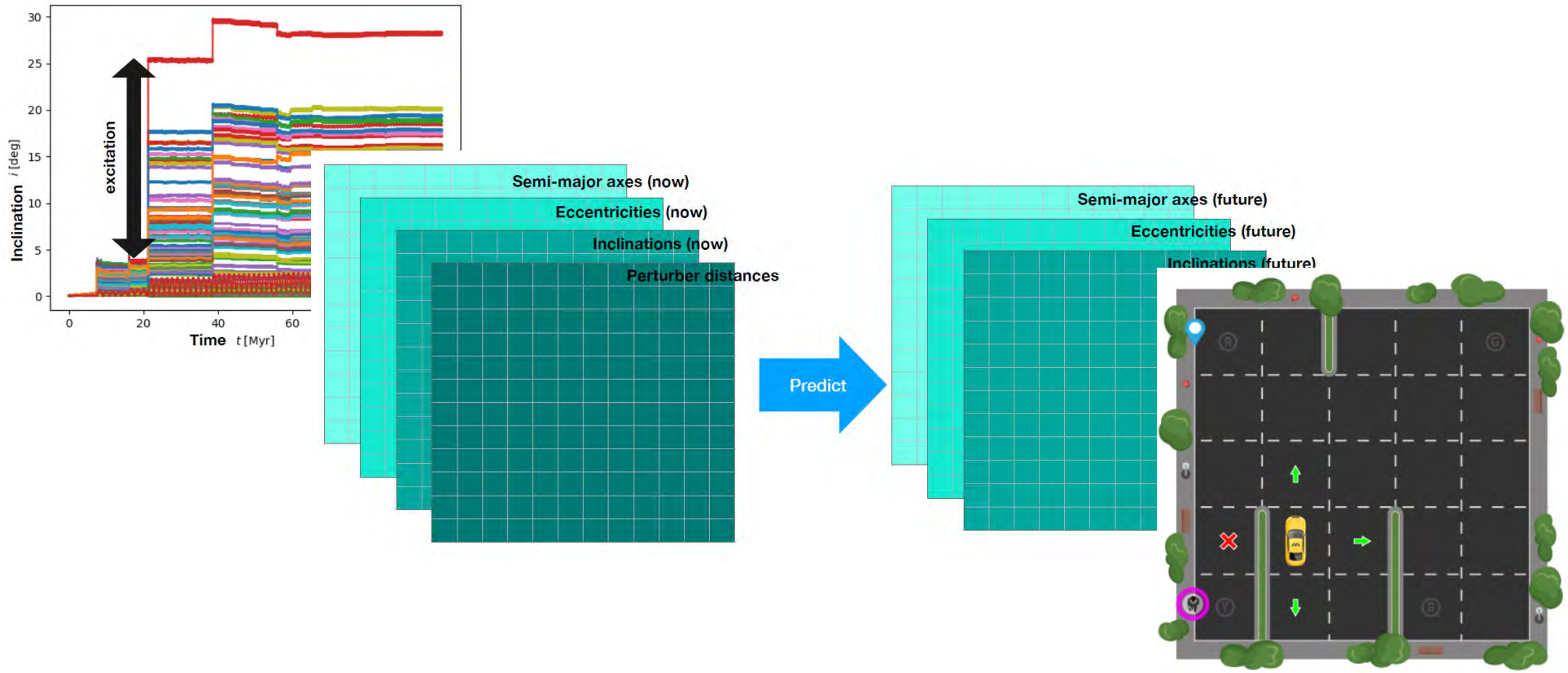
- Increases the efficiency of / reduce computation needed in your traditional simulation through DL
- Example: in a parameter sweep, use DL to skip irrelevant points (smart ‘pre-processing’)
- Example: use DL to interpolate / otherwise increase accuracy of your output (smart ‘post-processing’)

Replacement

- Replaces (part of your) traditional simulation with a DL model
- Example: replace a computationally heavy kernel in your simulation with a DL model.
- Example: replace the full simulation with a DL model

How do I encode my problem?

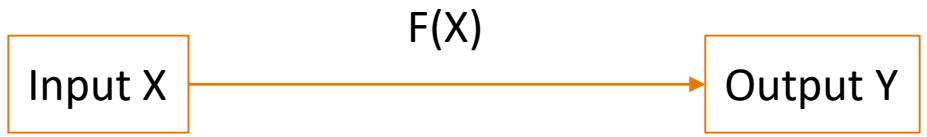
- Many ways to encode the same problem!
- Example: astrophysics project was encoded as time series, image, etc…



How do I encode my problem?

Does a relation between input and output *exist*? Use your domain knowledge to…

- Example: predicting unresolved turbulence based on coarse scale input *can* be done, because turbulence at large scales (that are resolved) are related to turbulence at small scales.



A neural network can approximate F(X), but only if *it exists!*

How do I encode my problem?

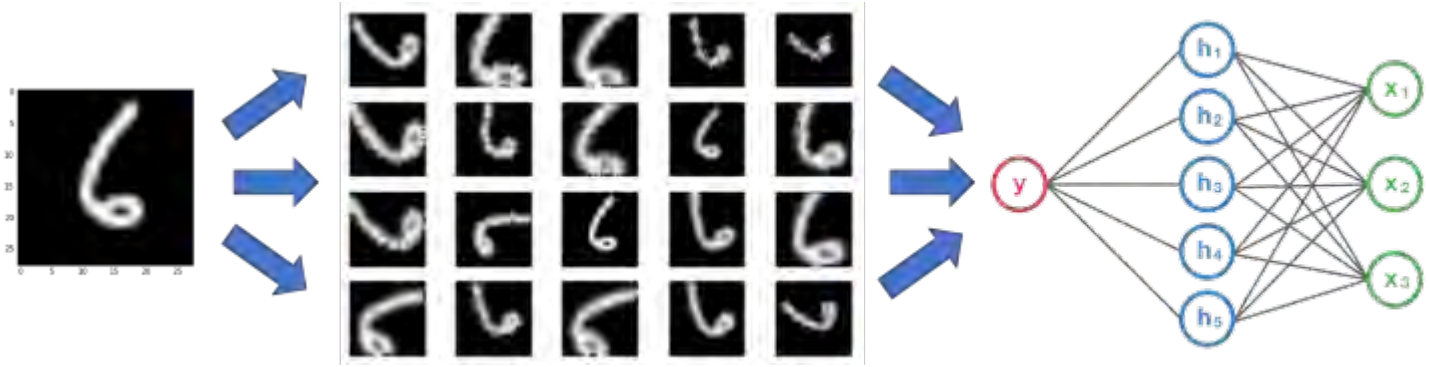
Do I have the correct data for this encoding?

- Can I generate data with my original simulation method?
- Is my data of sufficient quality?
- Is my data of sufficient quantity?
- Example: in turbulence modelling, data at both the coarse and fine resolution was needed. Since the simulation is stochastic, we could not simply run the simulation at two different resolutions => ran it at high resolution, and downsampled to obtain coarse resolution.

How do I encode my problem?

What do I know about my input and output data?

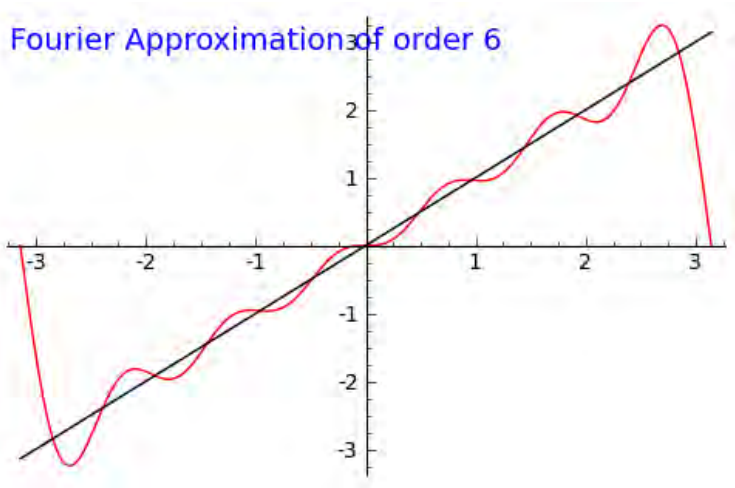
- Understand correlations between inputs, relationship between input and output data
- Is everything that would determine my output data within my input data?
- Example: if you're trying to replace an equation that is rotation invariant, you'll need a model that either incorporates this explicitly, or learns it from the data.



Data augmentation: rotation

What deep learning model fits that encoding naturally?

- A fully connected network *can* learn a time series problem, but doesn't map the problem as 'naturally' as a recurrent neural network
- A model that maps the problem better is likely to be more accurate / need less data / need fewer degrees of freedom
- Example: I can approximate a non-periodic signal with a Fourier series, but will need many degrees of freedom & a lot of data. Polynomials would be a more 'natural' fit.



What deep learning model fits that encoding naturally?

Rough indications:

Type of data	Architecture
Flat unstructured list	Fully connected
2D / 3D spatial	Convolutional architecture
Time series	Recurrent neural network / 1D Conv

Output of interest	Architecture
Predict single number / average	Regression
Sample distribution of outcomes	Generative networks (GAN, VAE)

Example meteorology: multiple unresolved transports possible for the same coarse grid. Sufficient if we are right *on average*.

Example particle physics: not interested in *average* particle properties (e.g. momentum), the whole point is we want *samples* from the full distribution.

How will I validate my trained model?

Hard to give general indication, depends on your acceleration approach

Augmentation / replacement of full simulation

- Compare to classical simulation outcomes (particle physics, astronomy, computational structural biology)

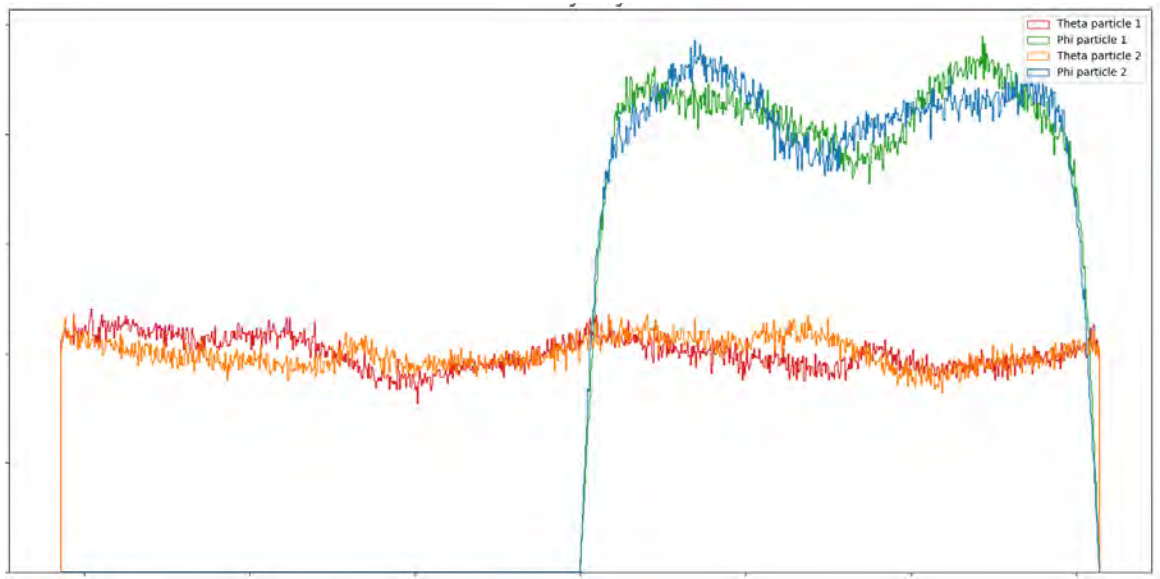
Replacement of part of simulation (kernel):

- A priori validation: compare that predicted output matches output of the classical kernel (meteorology example)
- A posteriori validation: integrate the DL-based kernel and check output of full simulation against classical numerical simulation

Try it! Iterate through the steps…

Try it! You may have too loop through these questions multiple times as you go…

- Developing a good deep learning approach is very much an *experimental* effort.
- If your results are poor, why could that be?
- Example (particle physics): GAN's resulted in realistic *individual* particle events. However their *frequency distribution* didn't match reality. Why? GAN's mostly check if individual events are 'realistic'!



Angular frequency distribution (GAN)

Improve inference speed

- Inference speed of *training* frameworks is often suboptimal
- Can you code it at a low level? Is there a framework you could use that is optimized for inference (e.g. Tensor-RT, Intel OpenVINO)
- Can you prune your network? (cut out unimportant connections to reduce computation)
- Can you redo your network in *reduced precision*? GPUs have increased throughput for e.g. FP16, mixed precision, INT8, etc. Reduces memory bandwidth bottlenecks (CPU & GPU).

Tensor core operations of Nvidia V100 GPUs

Lessons learned

- New paradigm (especially when replacing full simulations): instead of traditional numerical algorithms, you can simulate your theory through DL
- Scientific domains would have very different tools and software for numerical simulations. Now, we suddenly all have a very similar problem, with very similar tools. Opportunity for multi-disciplinary collaboration!
- Deep learning models rarely work ‘out of the box’, must often be tuned to the problem. This requires both expertise from the scientific domain and the deep learning domain.

Lessons learned

- Replacing small parts of a simulation, e.g. an existing parametric approximation, may more easily be accepted by community
- Replacing small parts will limit you in the same way as scaling through parallelization is limited: maximum speedup depends on how large that part is as fraction from the total compute time (Amdahl's law).
- Replacing full simulation can give *many* orders of magnitude speedup

Lessons learned

- Use your domain knowledge to solve problems you encounter! E.g. do you know that your replaced kernel should be invariant to rotation? Incorporate that knowledge!
- Invariance can be included in various ways: augmenting the inputs (e.g. rotating), choosing different inputs (e.g. inputs that are rotation invariant, such as the magnitude of a vector) or built into the network architecture. The latter two options have the advantage that the network doesn't have to 'learn' to invarience.
- The same problem can often be mapped in different way. A weather simulation can be viewed as a spatial problem (3D grid-based), but also as a time problem (e.g. wind velocity as function of time for a given voxel).
- In generative models, there is a tradoff between coverage (how varied are the samples I can generated) and if the samples are 'physically' correct.

Conclusion

- Very promising approach, large speedups are possible!
- New paradigm, especially when replacing full simulations. May meet resistance from scientific community!
- Pioneering work, requires a lot of exploration & adaptation
- Requires both deep learning & domain specific expertise (collaboration)

Round table

Goal

- Explore some of *your* use cases, help you formulate an initial approach

Setup

- Won’t be able to discuss all cases, but you can learn from other cases!
- Advise to stick with the same group, but you can switch after 45 min.
- Broad categories to increase the change of synergy with fellow participants
- But: feel free to discuss anything!

Round table

Topic	Chair	Where	Card
Problem encoding	Valeriu Codreanu	Plenary room	Red
Problem encoding	Damian Podareanu	Plenary room	Orange
2D/3D grid-based / mesh-based (convolutional networks)	Caspar van Leeuwen	VK2, SURFsara	Blue
Generation of samples, density estimation (GANs, VAEs)	Sydney Otten	Nano	Yellow
Time series (RNNs, reinforcement learning)	Maxwell Cai	Innovation lab	Green