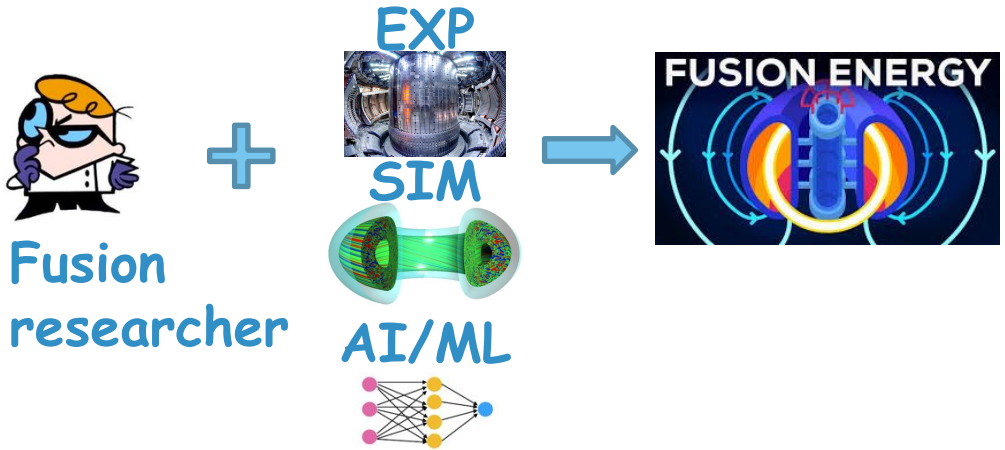
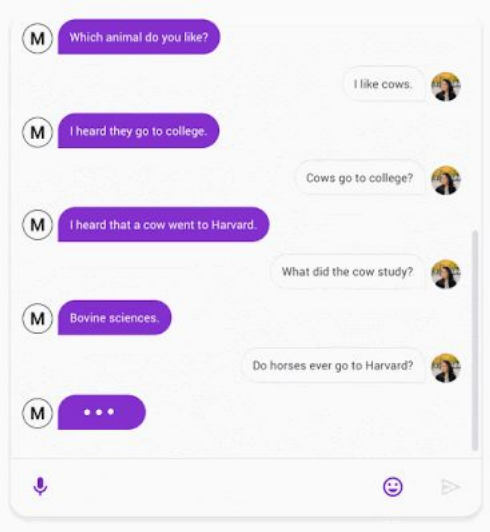
# AI/ML for Fusion - and beyond!

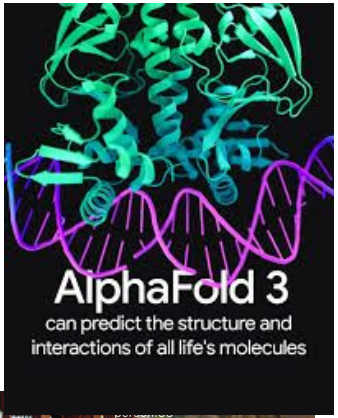
R. Michael Churchill (PPPL)



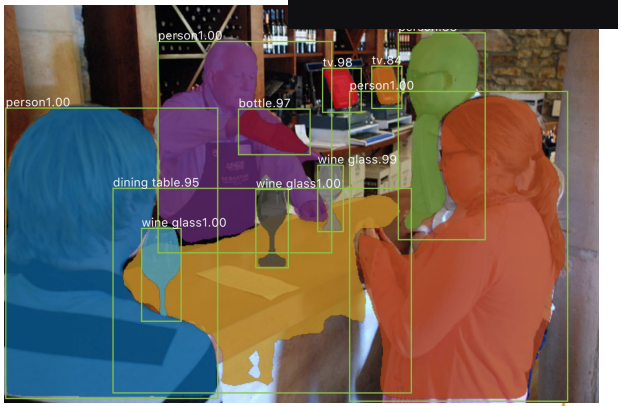
**Artificial intelligence** is the science and engineering of making computers behave in ways that, until recently, we thought required human intelligence – Andrew Moore, *Forbes Magazine 2017*





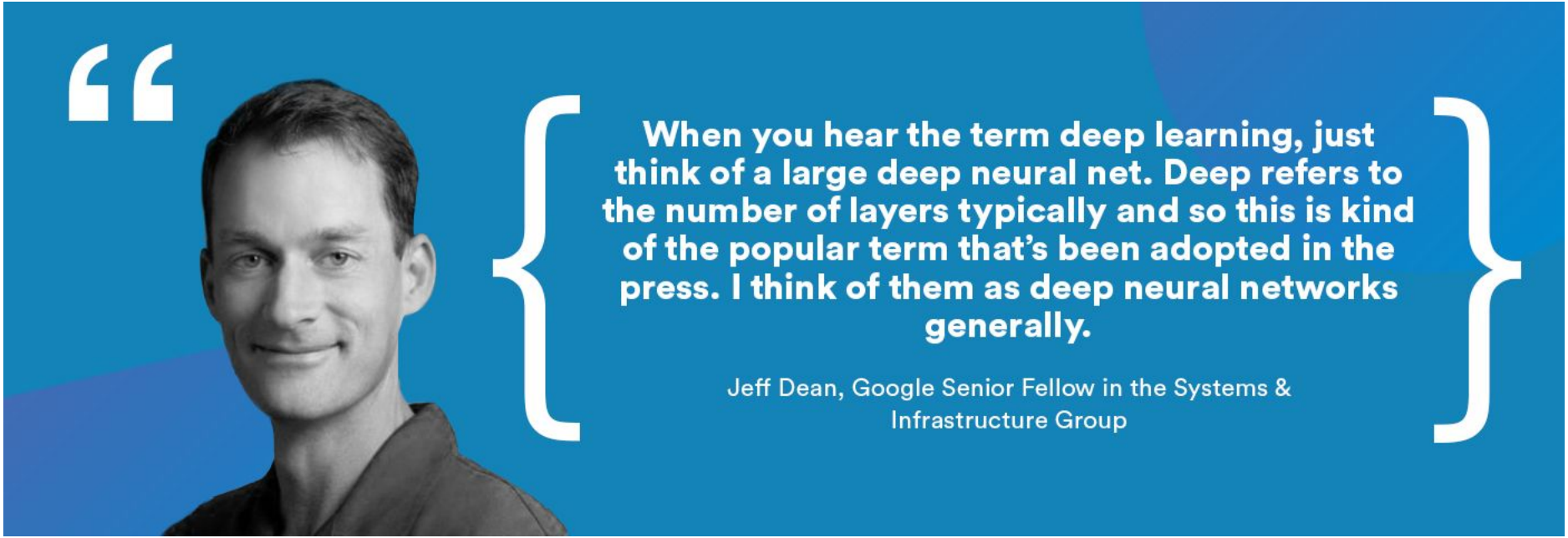






# Building Blocks

Models, Data, Compute, Training


When you hear the term deep learning, just think of a large deep neural net. Deep refers to the number of layers typically and so this is kind of the popular term that's been adopted in the press. I think of them as deep neural networks generally.
Jeff Dean, Google Senior Fellow in the Systems & Infrastructure Group

# Building blocks: model layer equations

## Layer equation

$$\mathbf{y} = \sigma(W\mathbf{x} + \mathbf{b})$$

Layer output

Nonlinear activation

Input

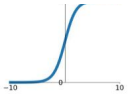
Weights/ biases

## Fully-connected Neural Network

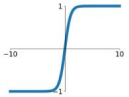
$\mathbf{w}$

## Nonlinear activation functions

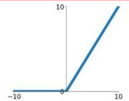
**Sigmoid**
$\sigma(x) = \frac{1}{1+e^{-x}}$



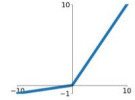
**tanh**
$\tanh(x)$



**ReLU**
$\max(0, x)$

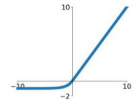


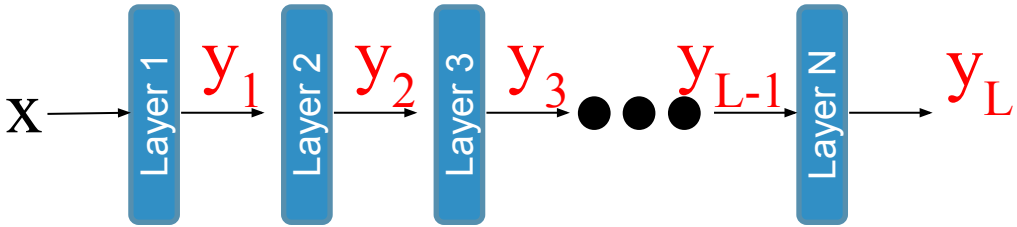
**Leaky ReLU**
$\max(0.1x, x)$



**Maxout**
$\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

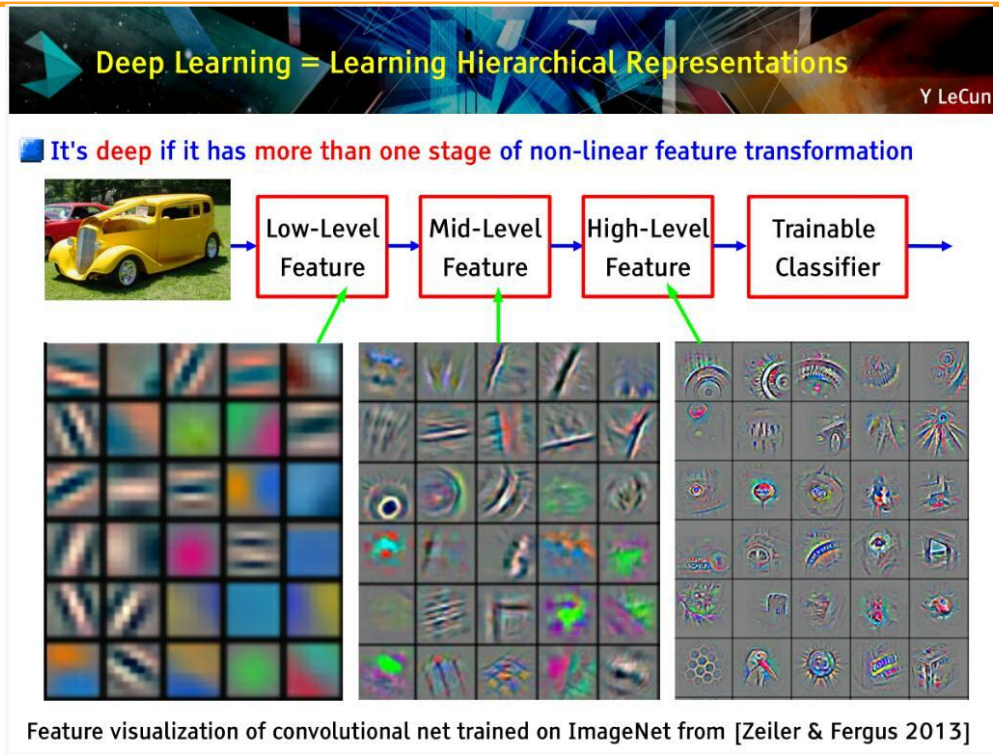







Mehta, V., et. al. (2020). http://arxiv.org/abs/2006.12682

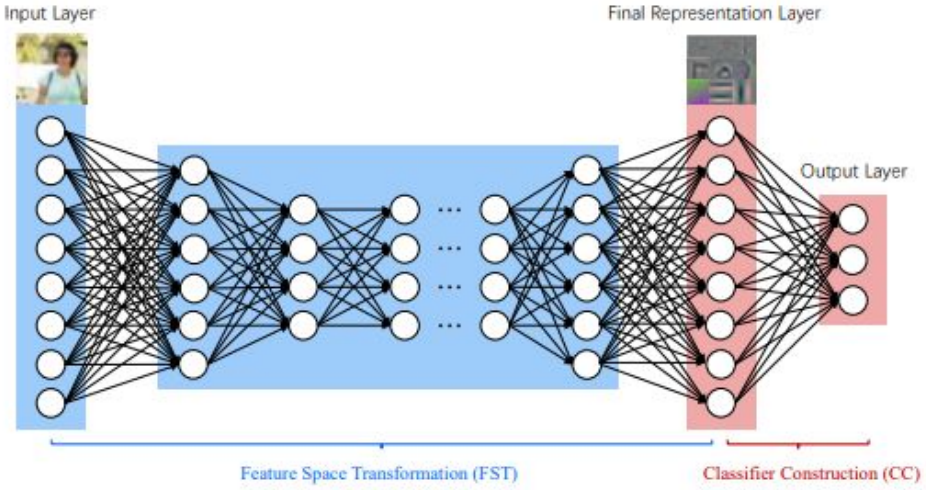
# Why go deep?
# Learn hierarchical, composable features




Important that these features are learned *jointly*, i.e. can not train layers separately and get the same result

# Features, representations, latent space

- One way to view deep NN is that they learn “features” or “representations”, with a final layer for classification or regression
- The feature space also often referred to as the latent space; data compressed to a space which latent random variables define




https://towardsdatascience.com/overparameterized-but-generalized-neural-network-420fe646c54c

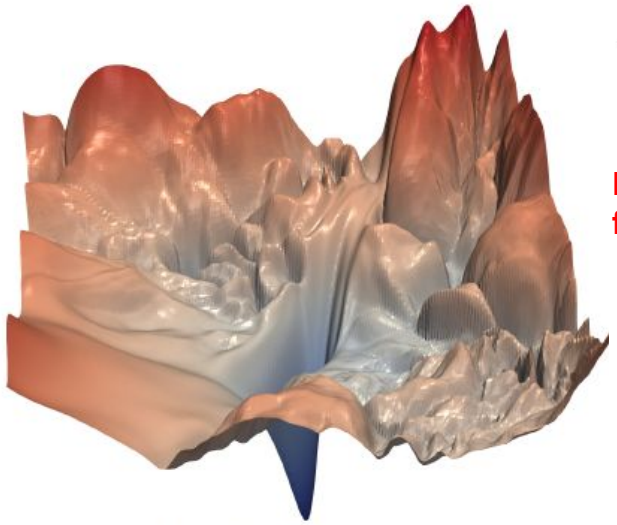
# Building blocks: data

- Due to large number of parameters, deep neural networks are data hungry.
- How much data do you need for your problem?
  - Answer is always “it depends” (complexity of the problem, size of the network, etc.), but more is (almost) always better
  - Rule of thumb ~5k examples per category for classification
- Typical “supervised learning” setup involves gathering input data and the targeted output data (e.g. input: pictures of cats/dogs; output: label for each picture whether cat/dog)

| Image | Label |
| --- | --- |
|  | Cat |
|  | Cat |
|  | Dog |
|  | Dog |

# Building blocks: Loss function for gradient-based optimization

**Goal of NN training is to minimize the loss function for the dataset**



$$f_\theta(x) = f^{(L)}\left(f^{(L-1)}\left(f^{(N-2)}\left(\ldots\left(f^{(2)}\left(f^{(1)}(x)\right)\right)\right)\right)\right)$$

NN output

Target output

Loss function

$$\ell\left(f_\theta(x), y\right) = \begin{cases} \frac{1}{2}(f_\theta(x) - y)^2 & \text{MSE loss} \\ -\sum_i^C y_i \log f_\theta(x) & \text{Cross-entropy loss} \\ \ldots \end{cases}$$

NN weight update

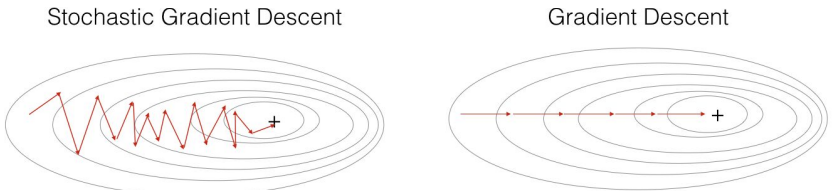
Learning rate

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial \ell}{\partial \boldsymbol{\theta}}$$

# Stochastic Gradient Descent (SGD)

- Performs optimization steps using part (or “batches”) of dataset for gradient (instead of entire dataset)
  - “stochastic” because random samples used in mini-batches
  - Spend more time processing more data instead of minimizing optimization steps
  - “the best optimization algorithms are not necessarily the best learning algorithms” [Bottou, NeurIPS 2007]

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \ell(f_\theta(x_i), y_i)}{\partial \boldsymbol{\theta}}$$



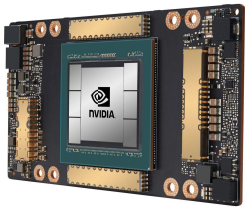

$n = N_{batch}$
$(N_{batch} < N)$

$n = N$

- Variants most commonly used:
  - SGD with momentum
    - Faster convergence
  - Adam
    - Easy default hyperparameters

# Building blocks: Compute with GPUs

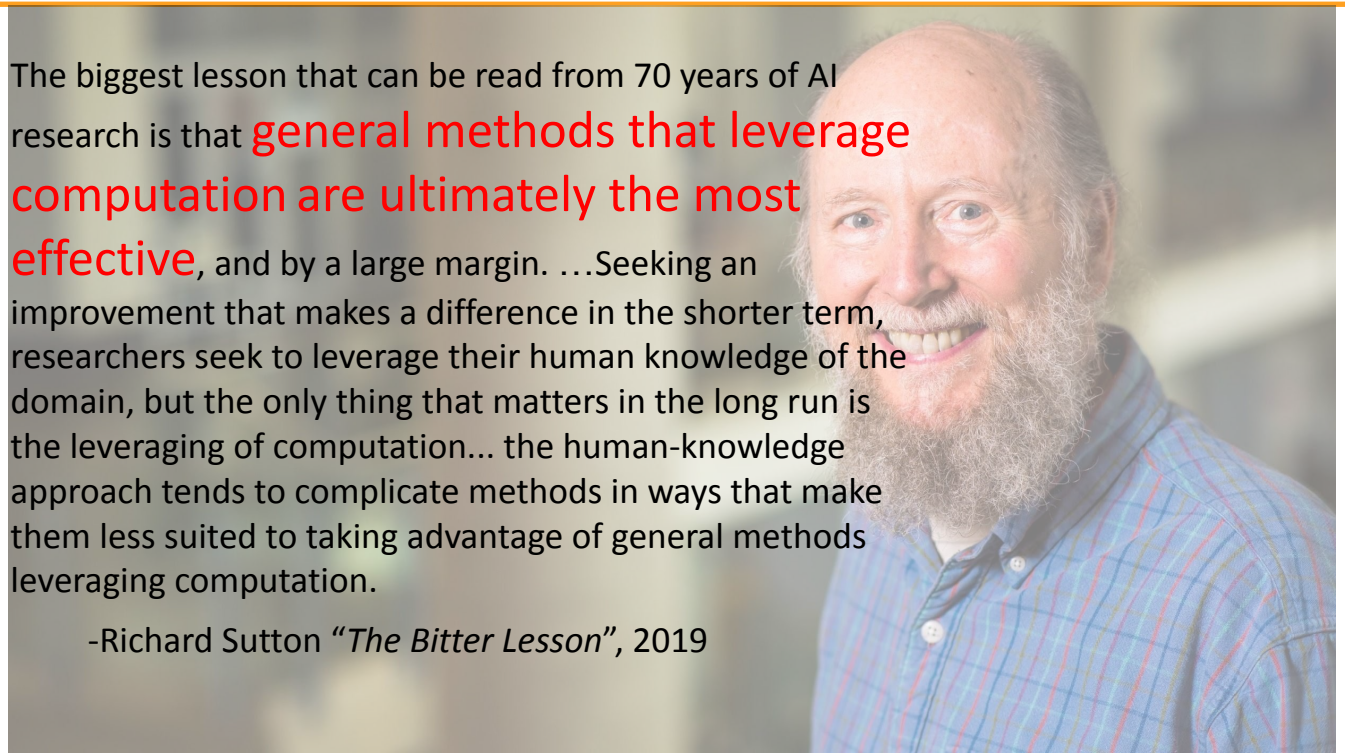
- Specifically CUDA parallel programming model made it easy to leverage GPUs for parallel processing, accelerating the tensor operations needed in neural networks
- Many frameworks exist to implement  deep neural networks; all make it seamless to leverage GPUs (no CUDA programming required)
- Key for fastest performance is pipelining the workflow to ensure GPUs don't sit idle
  - e.g. load next data batch using CPUs concurrent with GPU operations on other batch of data with pin_memory in Pytorch




NVIDIA CUDA

PyTorch

TensorFlow


The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. …Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation... the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation.
-Richard Sutton "*The Bitter Lesson*", 2019

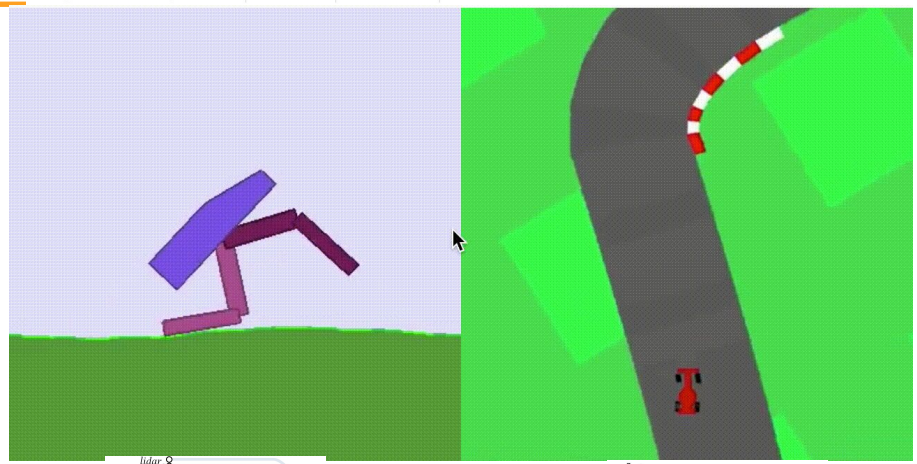
# Inductive bias

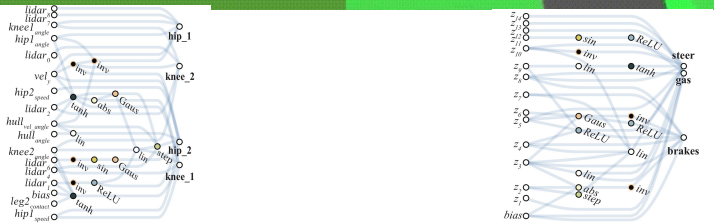
"Encode our knowledge and assumptions about the world"

# Weight Agnostic Neural Networks

ADAM GAIER Google Brain | DAVID HA Google Brain
June 12 2019 | Download PDF | NeurIPS 2019 Slides

- Instead of updating weights, modify *structure* of neural network
- Showed that with the right *structure*, learning was possible, despite single parameter weight





"Not all neural network architectures are created equal, some perform much better than others for certain tasks. But **how important are the weight parameters of a neural network compared to its architecture?**"

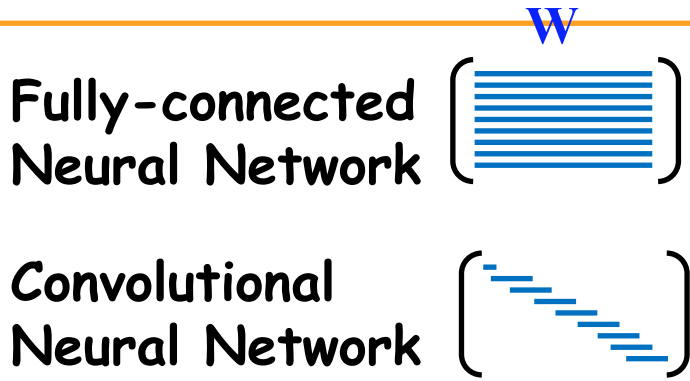
https://arxiv.org/abs/1906.04358

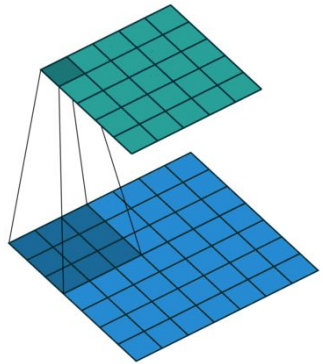
# Convolutional structure in neural networks a strong inductive bias for locality

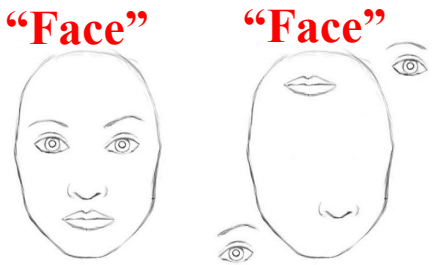
**Layer equation**

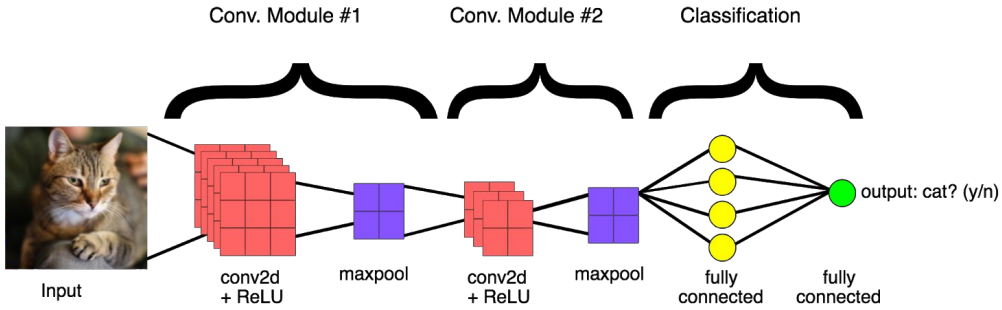
$$\mathbf{y} = \sigma(W\mathbf{x} + \mathbf{b})$$

CNN weight matrix sparse connectivity enforces **translation invariance,** useful for natural images. But also cons, e.g. one con is the ”Picasso effect”, default CNNs can’t distinguish global and relative relationships

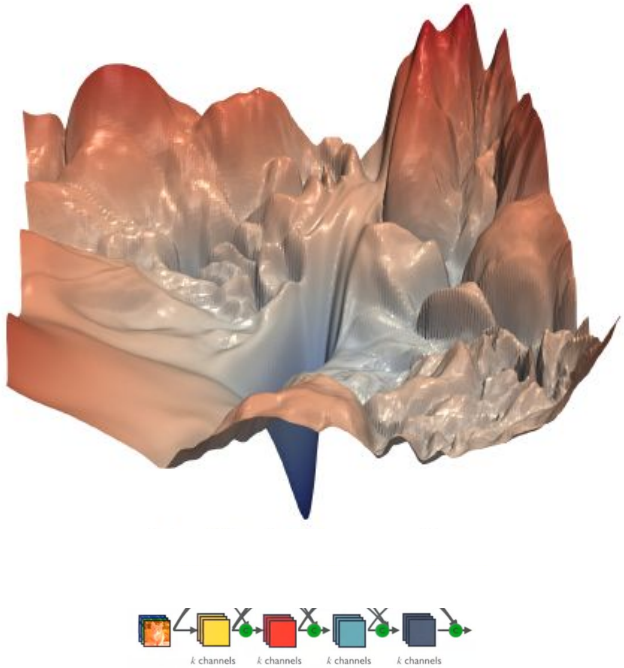


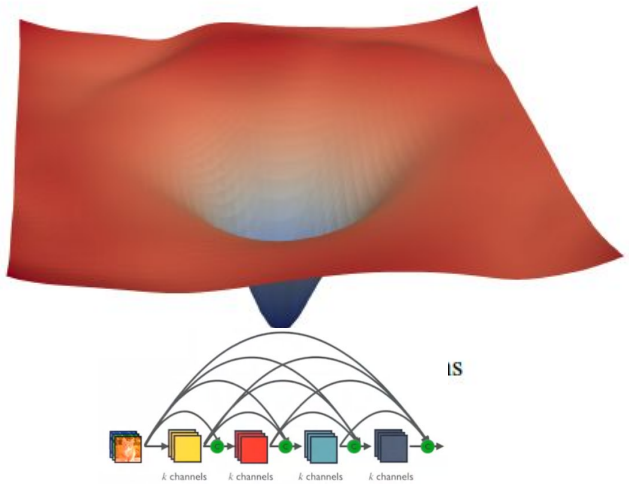


# Architecture choices have dramatic effect on loss landscape -> ease of training




No Residual connections




With Residual connections



Li, H., *et. al.*(2018). https://arxiv.org/pdf/1712.09913.pdf

# Neural network architecture can incorporate different levels of physics

- Many NN architectures offer enhancements in terms of learning capacity, and ability to follow the operator manifold

Larger models/more data

**Large capacity, little physics**

**E.g. EfficientNet, ViT**

**Large capacity with sufficient data can learn very general functions, but unclear if can reach the numerical precision levels for PDE solving**

**General physics incorporation**

**E.g. Hamiltonian NN , Equivariant NN**

**NN with physics symmetries inherent promise to stay closer to operator manifold, but incorporating multiple symmetries challenging**

**Operator learning**

**E.g. FNO, DeepOnet**

**Specialized neural network structure in principle allows learning the general collision operator, data efficient,**

More inductive bias/less data

# Architectures and Techniques Related to Scientific Deep Neural Networks

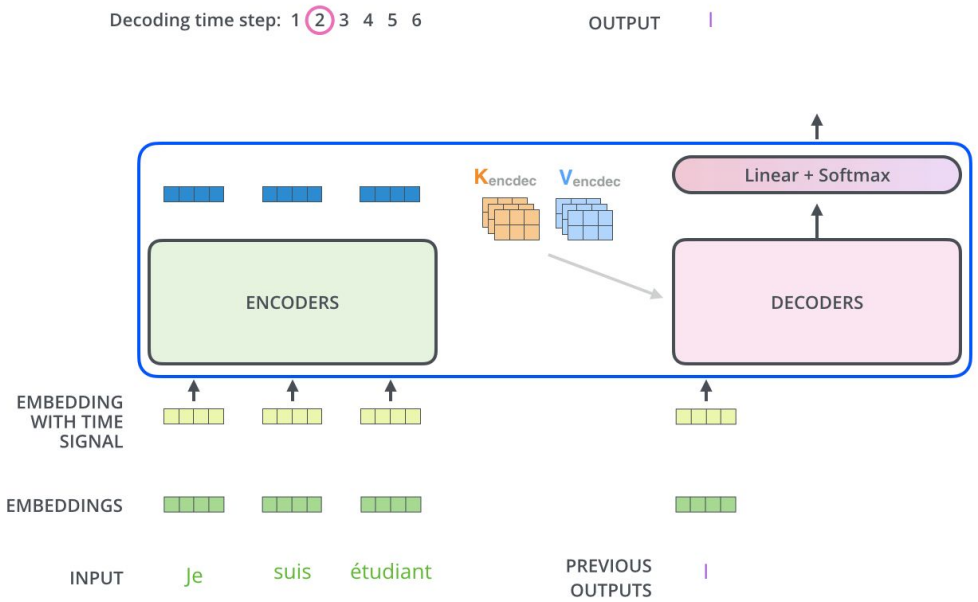
Resource for exploring current models:

Papers With Code

https://paperswithcode.com/

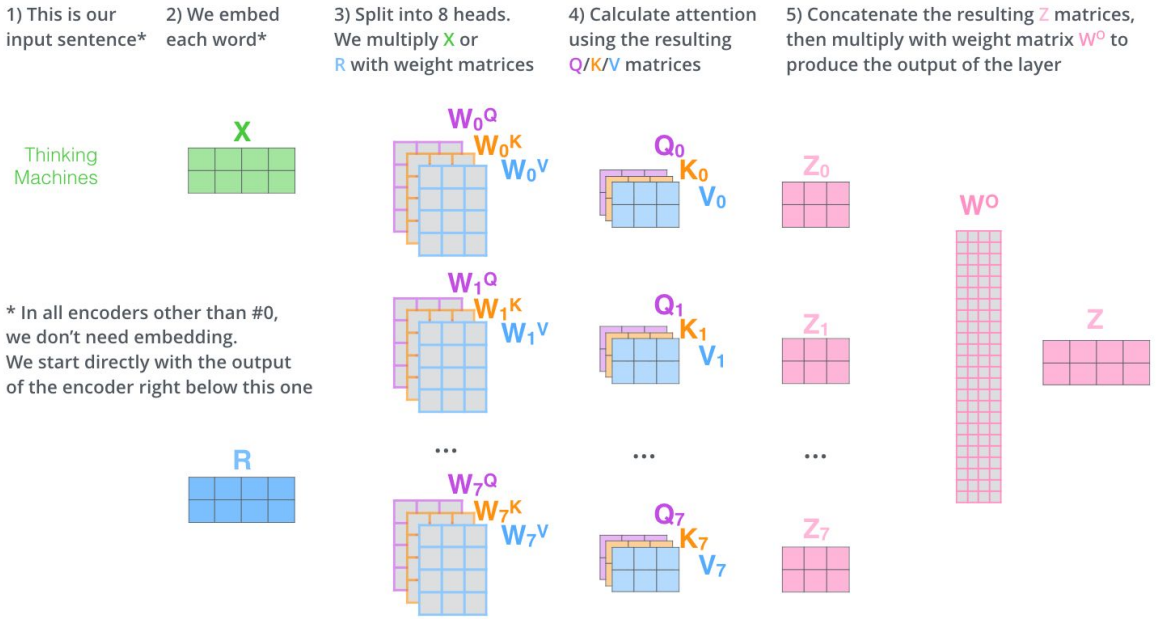
# Sequential Models

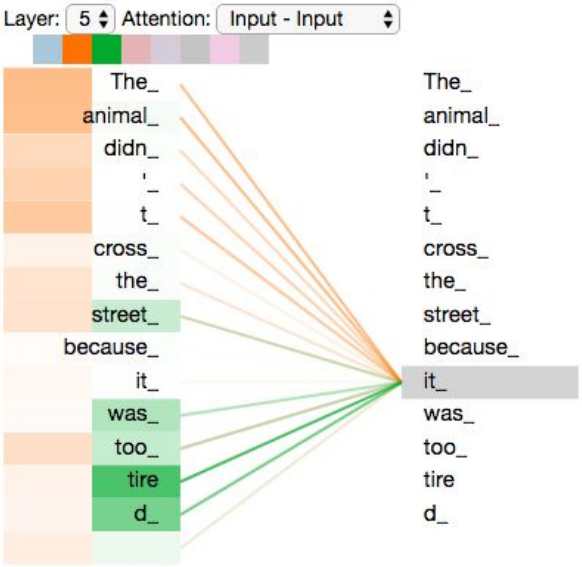
# Sequential models: Transformer




A **Transformer** is a model architecture that eschews recurrence and instead relies entirely on an attention mechanism to draw global dependencies between input and output. Before Transformers, the dominant sequence transduction models were based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The Transformer also employs an encoder and decoder, but removing recurrence in favor of attention mechanisms allows for significantly more parallelization than methods like RNNs and CNNs.

# Sequential models: Transformer cont., the attention mechanism



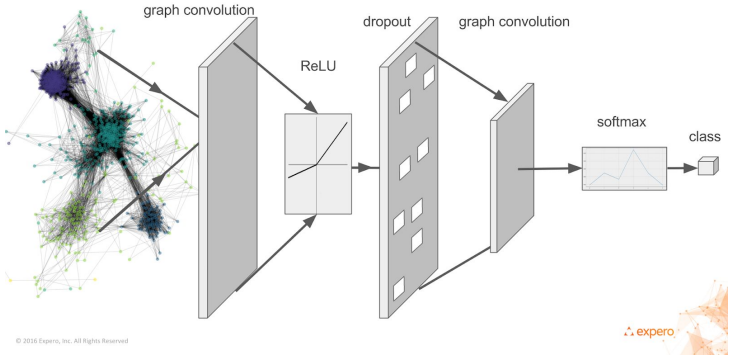



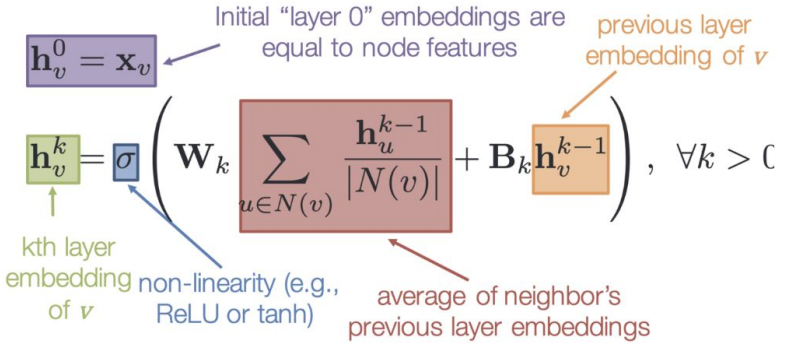

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Graph Neural Networks

# Graph Neural Networks

- GNNs operate on graph structures with nodes/edges
  - Perform better with fewer layers

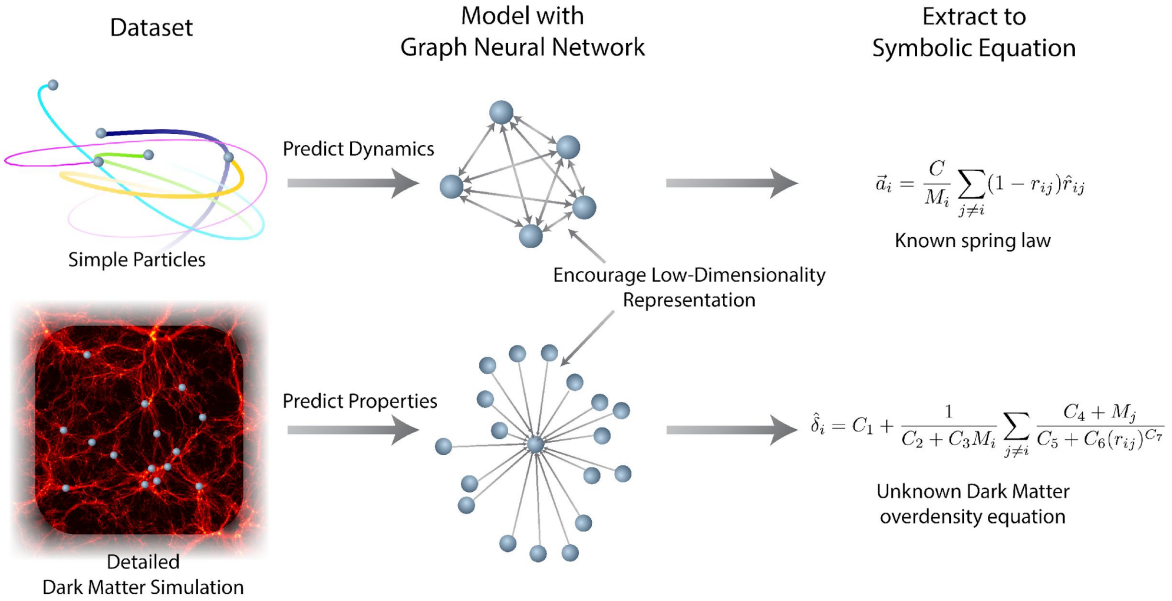



Savannah Thais, Graph Neural Networks

https://ericmjl.github.io/essays-on-data-science/machine-learning/graph-nets/
https://theaisummer.com/graph-convolutional-networks/
https://theaisummer.com/gnn-architectures/

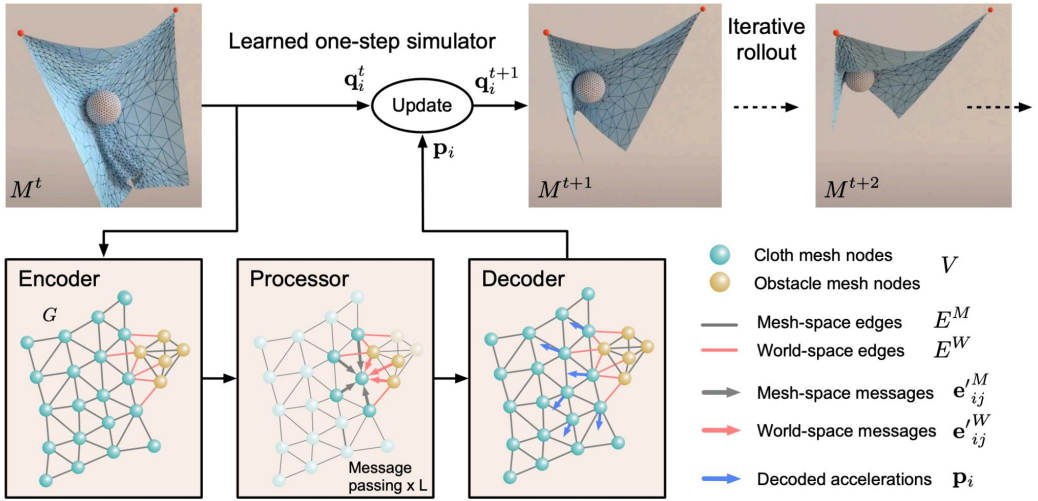
# Graph Neural Networks for learning N-body problems and dark matter is cosmology




M. Cranmer, https://astroautomata.com/paper/symbolic-neural-nets/

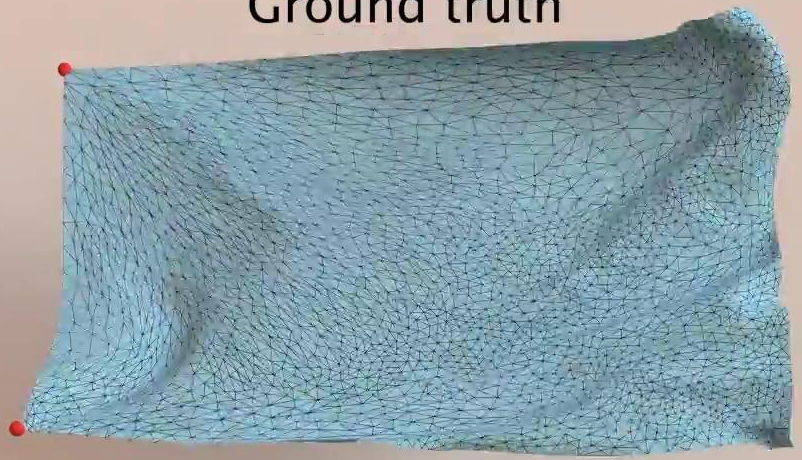
# Learning on complicated mesh structures

- Encodes information from mesh onto lower-dimensional graph structure in order to simplify the prediction process
- Some capability shown for beginning with simple mesh and transferring to larger mesh



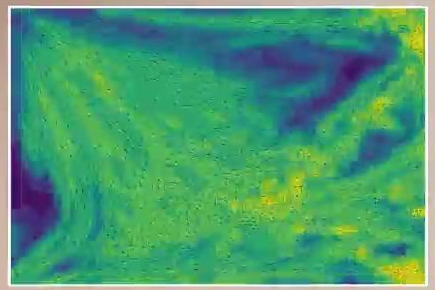

Pfaff 2021

Ground truth

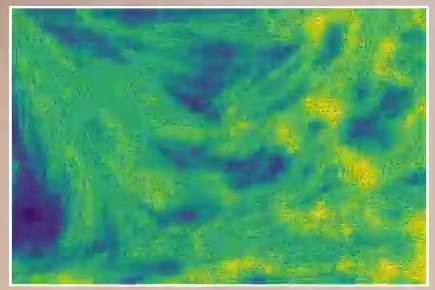


Prediction
(with learned remeshing)

Sizing field



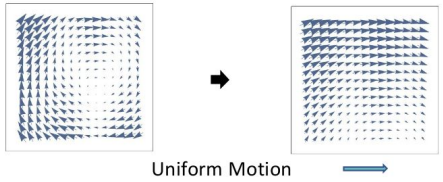
Learned sizing field



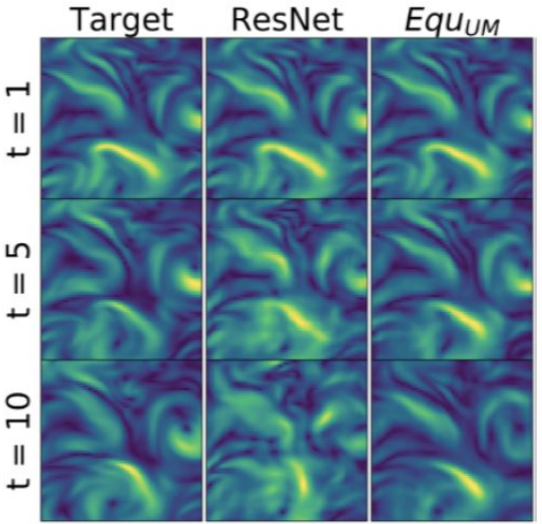
We can learn to adapt the mesh during rollout

# PDE solving

# Equivariant Neural Networks




- Modeling PDEs such as Navier-Stokes can be made more accurate by using NN architecture which guarantee symmetries of the underlying PDE are satisfied
- Ex: Ocean data flow prediction enforcing Uniform Motion performs much better over long time



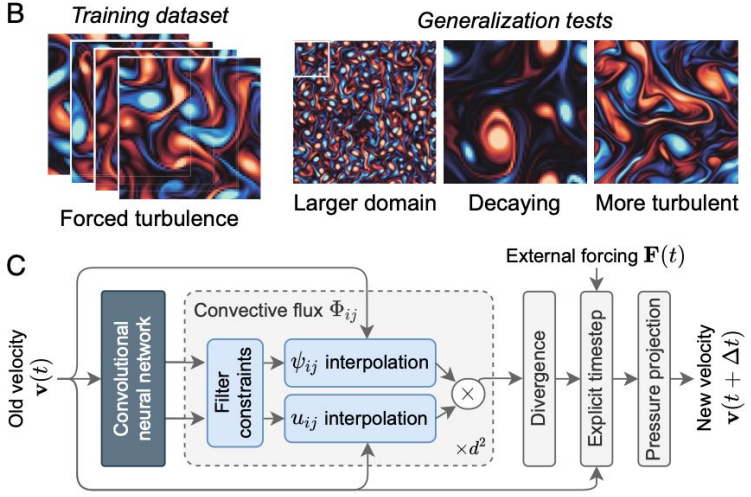

Robin Walters, "Incorporating Symmetry into Deep Dynamics Models for Improved Generalization." ICLR 2021.

# Solving on coarse grain grids, leveraging differentiable simulators

- Hybrid approaches can use NN to target specific parts of numerical PDE algorithms (e.g. local operator for convective fluxes)
- Learn to replicate high-res simulations on coarse, limited grid, inference on fine, expanded grid
- With a fully differentiable simulator, can optimize end-to-end through multiple steps of simulation. Help stability.

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \frac{1}{Re}\nabla^2 \mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0,$$



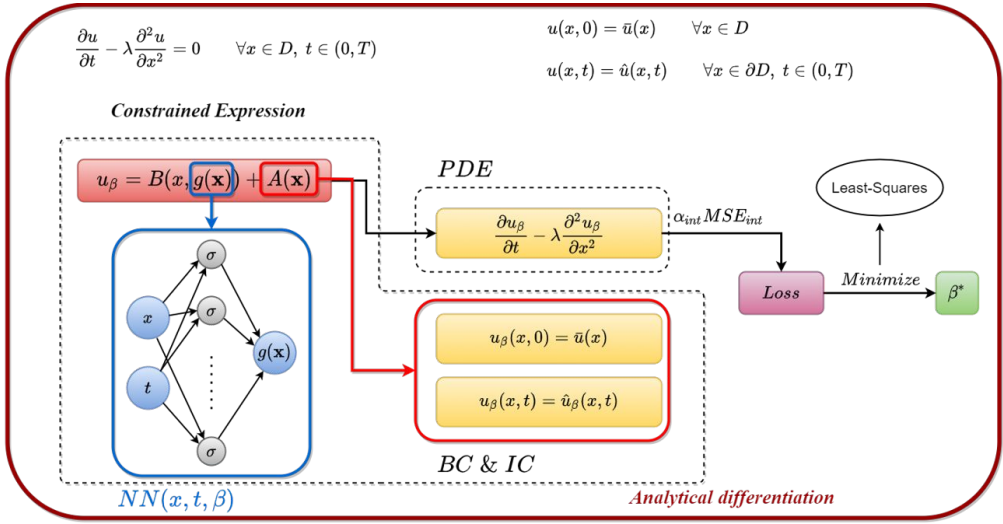

Kochkov, PNAS 2021

# Using trained neural networks allow gradient based design optimization


Allen 2022

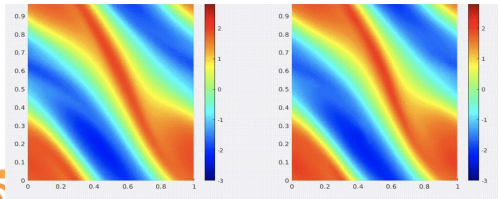
# Physics-informed Neural Network (PINN)

- Originally designed to solve single instance of PDE
  - Didn't require many data solutions, only evaluations at certain points
  - …but often traditional numerical solver would be faster
- Extensions modify to make more generalizable:
  - Put BC/IC as inputs instead of loss
  - Add some data examples to train on
- Still, PINN most utility for inverse parameter extraction from PDE (not forward solve)



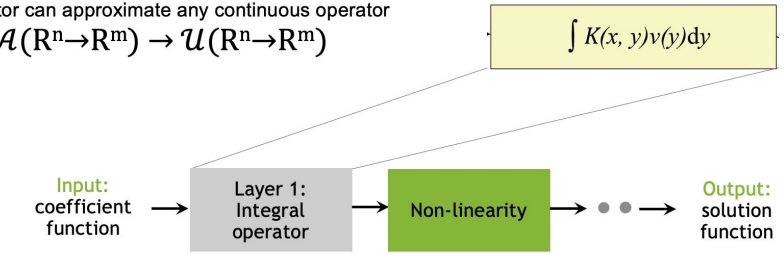
# Learning operators instead of functions

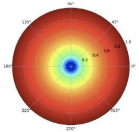
- Replacing linear function with an integral operator enables better generalization to unseen data
- e.g. Fourier Neural Operator (FNO) for fluid flow, 1000x faster



Neural Operator can approximate any continuous operator

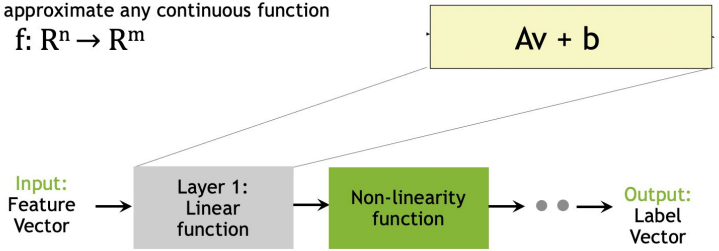
$\mathcal{F} : \mathcal{A}(\mathrm{R}^n \to \mathrm{R}^m) \to \mathcal{U}(\mathrm{R}^n \to \mathrm{R}^m)$




Neural Network can approximate any continuous function

$f: \mathrm{R}^n \to \mathrm{R}^m$




Anima Anandkumar, GTC2021

"A Digital Twin is more than a 3D model or a simulation. It is an evolving, virtual representation of an object or process that uses data from, and observations of, its physical counterpart."

Allows doing "what if?" analysis:

-Design optimization

-Operational optimization

-Predictive maintenance

-etc.

Machine learning for:

a. Surrogate model/accelerating simulation
b. Connecting physical asset to digital asset

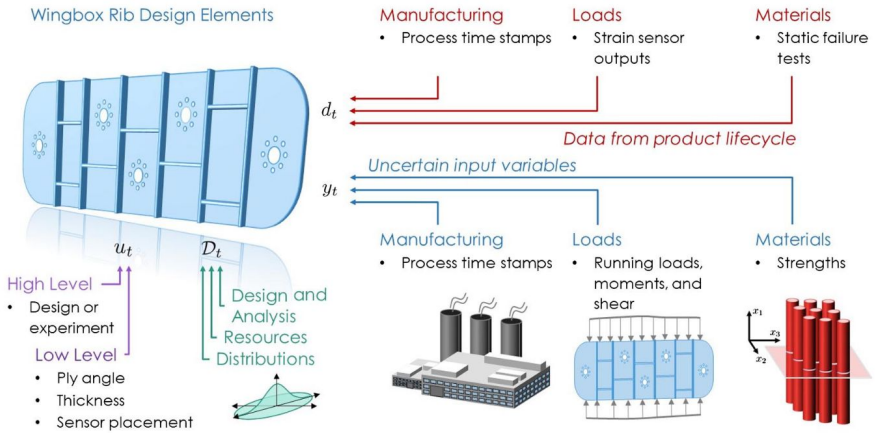



Fig. 3 Illustration of the four key elements of the design problem for the wingbox rib example.

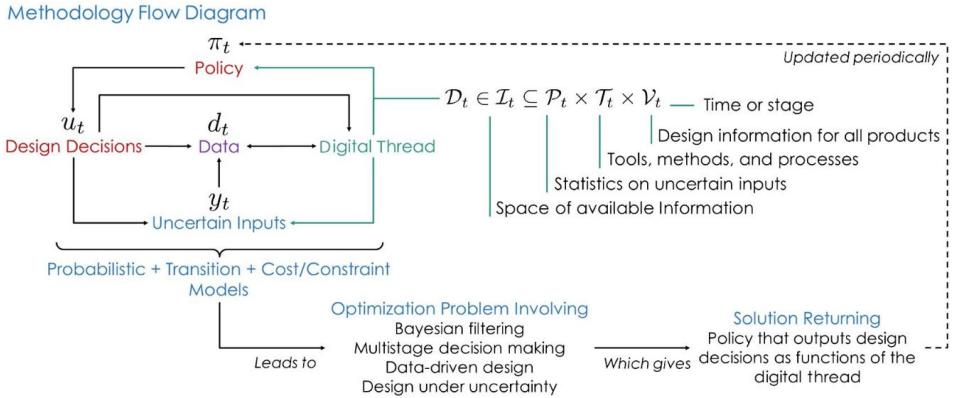



Fig. 5 Flow diagram of complete methodology.

$$V_t^*(\mathcal{D}_t) = \min_{u_t \in \mathcal{U}_t} \mathbb{E}\left[r_t(\mathcal{D}_t, u_t, y_t) + \gamma V_{t+1}^*(\Phi_t(\mathcal{D}_t, u_t, d_t)) | \mathcal{D}_t, u_t\right]$$

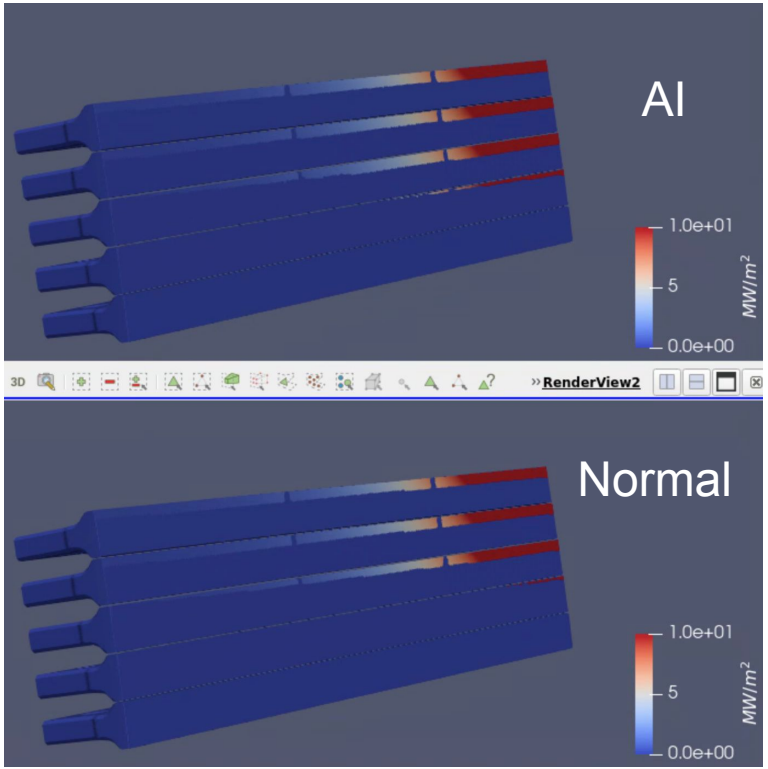
$$\text{s.t. } \mathbb{E}[g_t(\mathcal{D}_t, u_t, y_t) | \mathcal{D}_t, u_t] \leq 0, \ t \in \{0, \ldots, T\}, \ V_{T+1}^* = 0$$

Does one invest early in small-scale experiments to reduce the uncertainties for a subset of the uncertain input variables? Or, does one proceed to manufacturing and deployment to gain revenue through sales and gather other sources of data including specific manufacturing time stamps and operating conditions? When is one decision favored over the other and under what conditions?

Singh, Wilcox 2021

# AI based digital twins for engineering operations

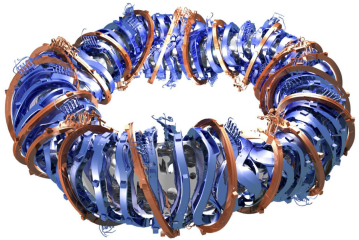
- AI models for accelerating HEAT code (divertor heat flux modeling code), for fast comparison to experimental conditions
- Based on low(er)-fidelity physics combined with accurate 3d CAD geometries
  - Faster for generating training data, but needs to be modified by experiment data. Not clear if can adjust incorrect simulation with experiment sufficiently
- Tension between low- and high-fidelity for creation of digital twin (can low-fidelity based be corrected?)

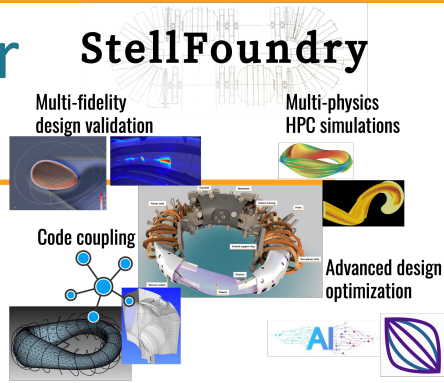



D. Rivera 2024 Fus. Eng.

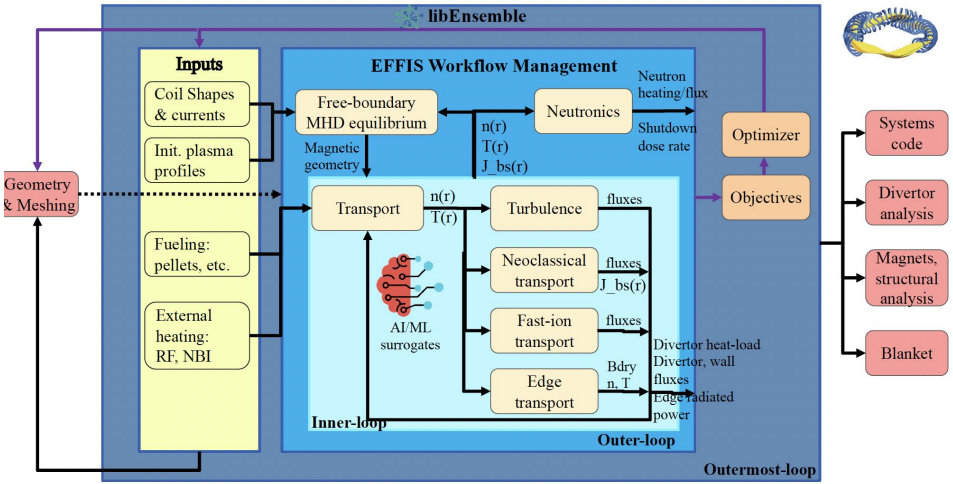
# Coupling multiple fidelity codes for stellarator design optimization and verification

- Design optimization traditionally makes use of simplified metrics for speed
  - Direct optimization using high-fidelity simulation *when needed* can make more robust designs
- Consistent plasma state from simulation by coupling multiple codes
  - Leverage AI surrogates/differentiable simulations for design optimization
  - High-fidelity simulation for design verification






SciDAC-5 project



# Accelerating stellarator integrated modeling and optimization with AI/ML

- Current integrated modeling of stellarator transport dominated by GPU-accelerated turbulence solver (GX, 24 hrs total), even though simplified physics

t3d

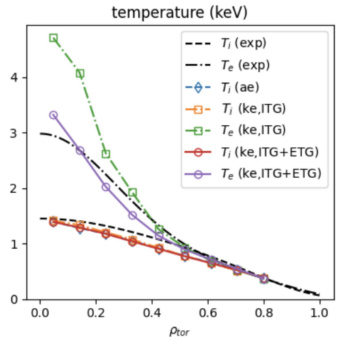
ESC

~24 hrs
(~2 hrs per iteration)

64 nodes
(256 GPUs)

GX

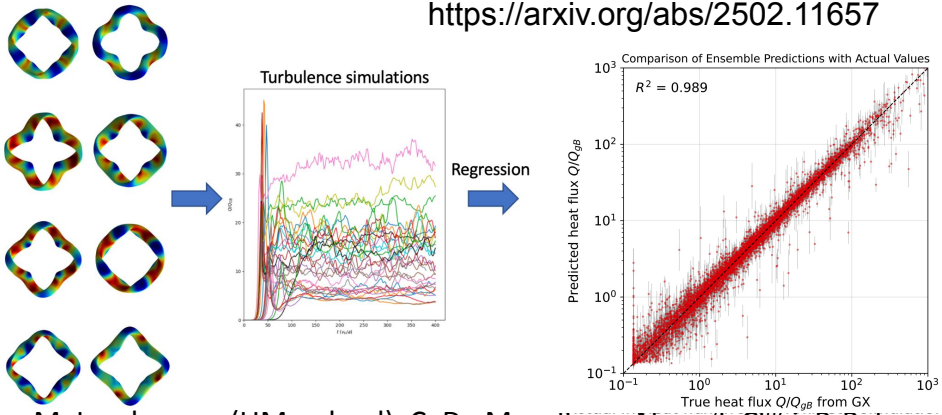
KNOSOS




N. Mandell (PPPL, now Type One), T. Qian (PPPL), J. Sachdev (PPPL), Bill Dorland (UMaryland, PPPL), M. Zarnstorff (PPPL)

https://arxiv.org/abs/2502.11657

DeepHyper

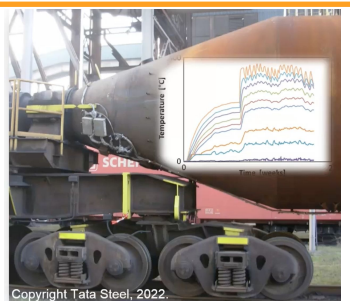


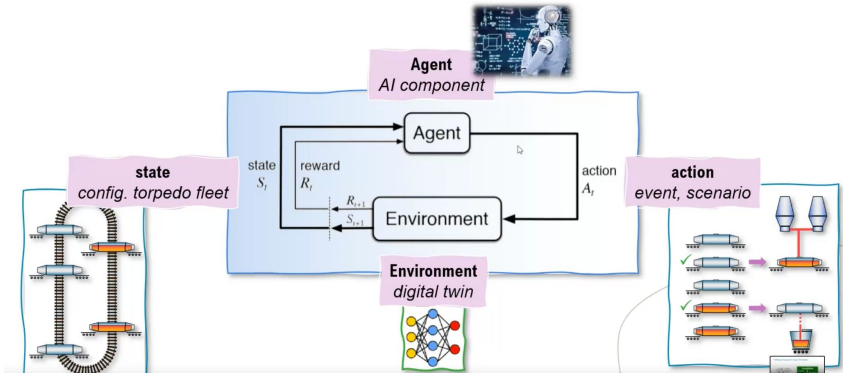

M. Landreman (UMaryland), C. De Magalhaes Alves, J. Choi, P. Balaprakash (ORNL)

- AI/ML surrogate model (CNN created by DeepHyper) predicting turbulent heat flux from magnetic geometry will accelerate 10-1000x
- Ensemble for uncertainties, call out to full GX model when errors large

# Digital twin in Nala Steel Plant

- FEM for thermal simulations of steel plant transport torpedoes
- Created reduced order models (ROM) based on neural networks with ANSYS Twin Builder
- Utilized in a planning scenario similar to reinforcement learning for planning with the environment



# How can we best use AI for scientific discovery with experiment?

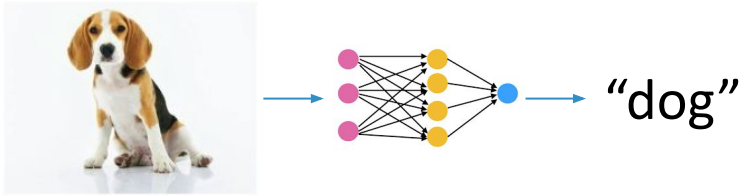
This talk explores two frontiers:

1. **Connecting Models & Reality:** Using generative AI to rigorously compare simulations with experimental data.
2. **Automating the Workflow:** Using agentic AI to assist in the process of research itself.

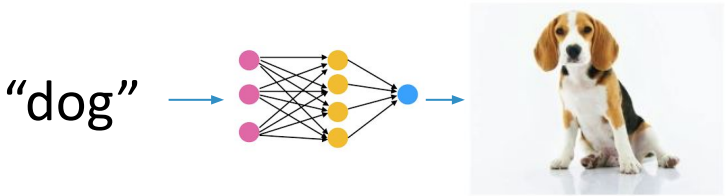
# Generative AI

# Generative AI

- Simplify life to two kinds of AI:
  - Predictive AI:
    - Takes dataset of paired inputs {θ} and output {x}, and learns **p(x|θ)**, i.e. how to **predict** e.g. classification or regression for some output x

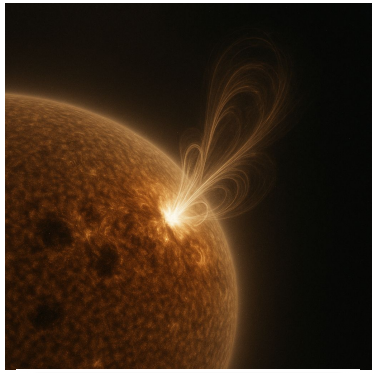



  - Generative AI:
    - Takes dataset of inputs {θ}, possibly with output pairs {x}, and learns entire data distribution **p(θ,x)**, often conditioning on x so **p(θ|x)**, i.e. how to **generate** a new input data sample given (sometimes also called "probability density estimators")

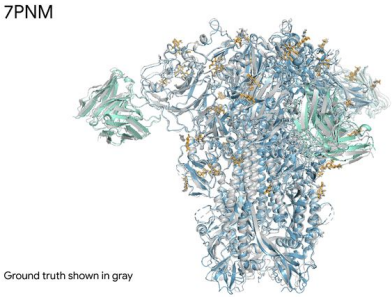


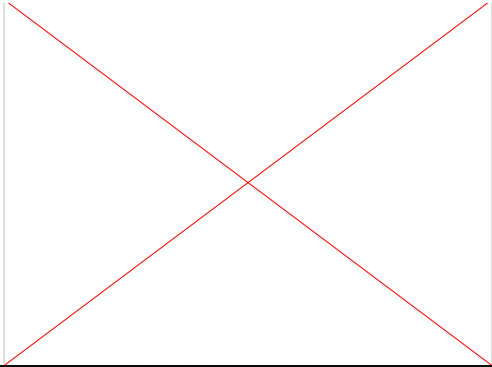
# Generative AI in industry

- Significant advances in a short few years using generative AI for:
  - Chatbots (e.g. ChatGPT)
  - Image generation (e.g. Midjourney, ChatGPT,
  - Video generation (e.g. Veo3, Sora)
  - Protein (e.g. AlphaFold, RFDiffusion)
  - Novel materials (e.g. MatterGen)
- In common are AI architectures which excel at learning complex data distributions



Generate an image of a solar flare on the sun

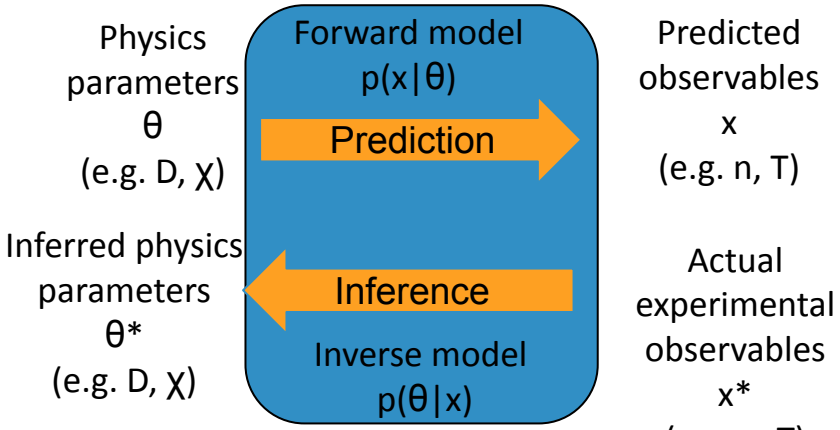


**Prompt:** A medium shot frames an old sailor, his knitted blue sailor hat casting a shadow over his eyes, a thick grey beard obscuring his chin. He holds his pipe in one hand, gesturing with it towards the churning, grey sea beyond the ship's railing. "This ocean, it's a force, a wild, untamed might. And she commands your awe, with every breaking light"

# Motivation: Utilize simulation for fast statistical inference of physics parameters from experiment

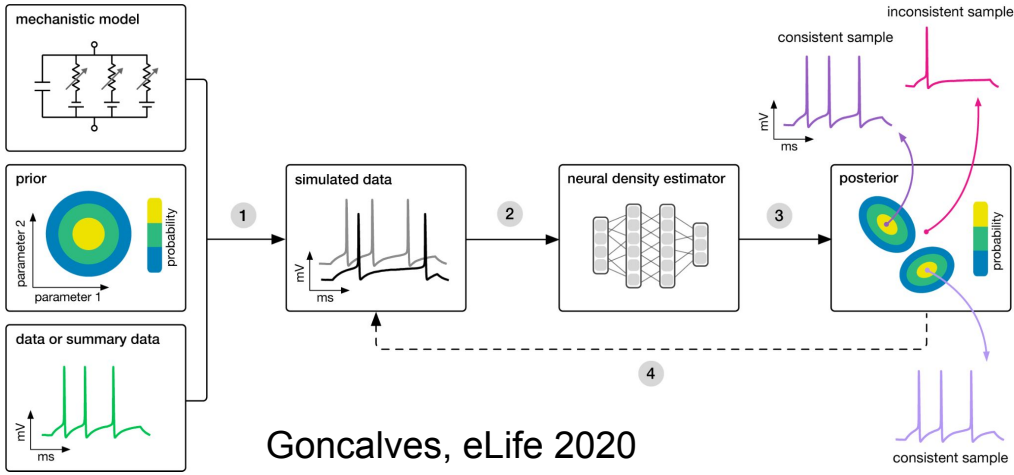
- Bayesian inference can extract physics parameters from experiment data, but typically explicitly specify likelihood $p(x|\theta)$
  - Some simulators with randomness can't explicitly define a likelihood
- Traditional methods (e.g. MCMC) for statistical inference are sequential, require ~hours, don't scale to number of experimental points in fusion experiments




**How can we utilize simulators without explicit likelihoods, and ensure fast inference?**

# Simulation-based inference (a.k.a. likelihood-free inference) using neural density estimator

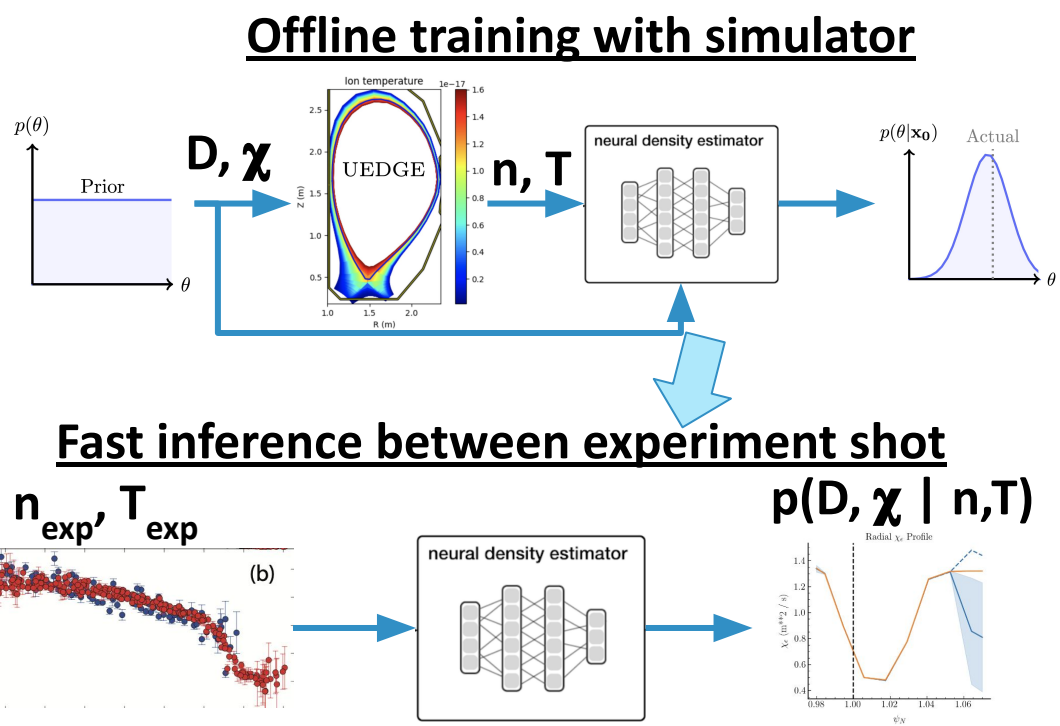
- Procedure to use:
  - Generate sample physics parameters $\theta_i \sim p(\theta)$ from prior
  - Run through simulator ("forward model") to generate experimental observations $x_i \sim \text{Simulator}(\theta_i)$
  - Train neural network with dataset $\{\theta_i, x_i\}$'s to learn conditional density, e.g. the posterior $p(\theta|x)$
  - Can train amortized model for generic experimental targets (x), or sequentially update prior in rounds for a specific instance of $(x_0)$




Goncalves, eLife 2020

# Simulation-based inference (SBI) enables between-shot extraction of physics parameters from experiment

- Train normalizing-flow based neural network to learn posterior from many simulation
- Extract for example anomalous transport coefficients from experiment, consistent with SOL simulator (UEDGE)
- **Make routine and robust experiment comparison to simulation**
- **Use in digital twins for robust uncertainties**






R. Michael Churchill, GSS 2025

C. Furia, R.M. Churchill, PPCF 2022

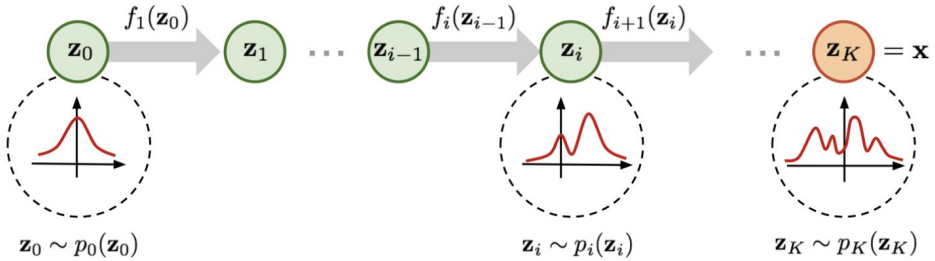
# Normalizing Flows

- Normalizing flows are a bijective (invertible) transformation
  - Allows transforming a known distribution $p_Z(z)$ e.g. a Gaussian to a more complicate one $p_X(x)$

$$f : R^n \to R^n \quad \text{such that} \quad x = f(z) \quad \text{and} \quad z = f^{-1}(x)$$

$$p_X(x) = p_Z\left(f^{-1}(x)\right)\left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right| = p_Z(z)\left|\det\left(\frac{\partial f}{\partial z}\right)\right|^{-1}$$

$$\log p_X(x) = \log p_Z(z) - \log\left|\det\left(\frac{\partial f}{\partial z}\right)\right|$$

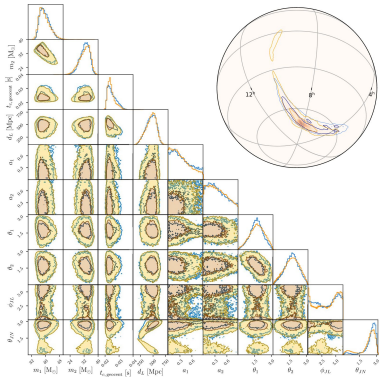



- Chaining bijective transforms also leads to a total bijective transform
- For discrete number of layers, effectively invertibility *hard-coded* into structure
- Can be used for conditional density estimation of posterior $p(\theta|x)$

https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

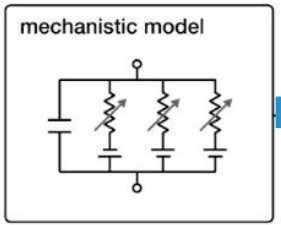
# Simulation-based inference in gravitational-wave observatories

- Simulation-based inference (sbi) using normalizing flows used to infer binary black hole merger GW150914, successfully applied in other fields (e.g. gravitational waves with LIGO, see Green, *Mac. Learn Sci. Tech*, 2021)
  - Use forward model *IMRPhenomPv2* to generate gravitational waveforms produced by quasi-circular, precessing binary black hole systems, include realistic noise
  - Inference parameters masses of black holes, spin magnitudes, angles, inclination, distance to the system, time of coalescence, and sky position.
  - Compared to MCMC method, showed good agreement
    - MCMC take days to weeks, SBI generates 5k samples per second

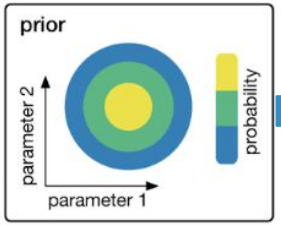


# Example applying SBI to plasma physics

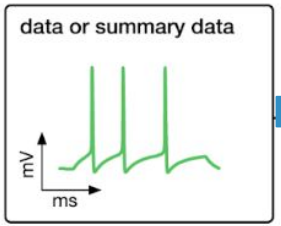
https://github.com/cfuria/ppcf_lfi




UEDGE, a simulator based on a fluid model for scrape-off layer transport



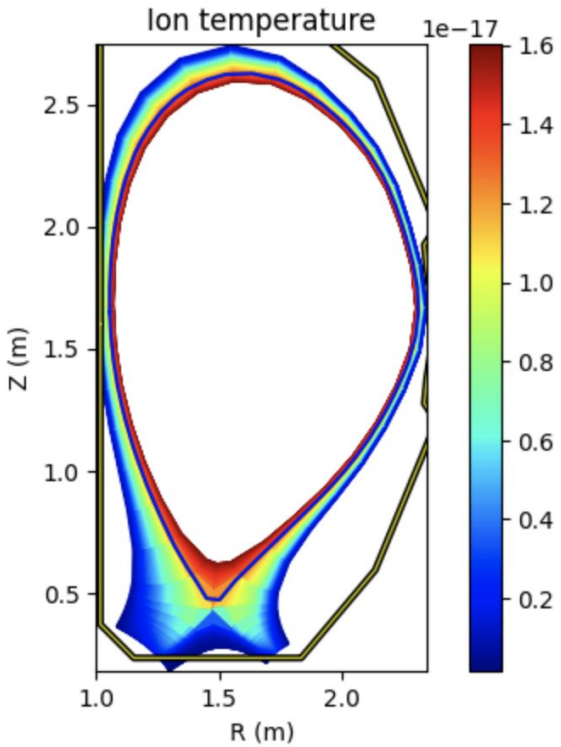

Physics parameters anomalous transport coefficients representing turbulence, $\theta = \{D, \chi e, \chi i\}$, 1D profiles vs $\psi_N$. Prior is uniform distribution.




Generated toy problem "experiment" data, kinetic profiles $n_e$, $T_e$, $T_i$ from a ground truth $\theta_0$

# Simulator: UEDGE

- UEDGE is a fluid SOL code from LLNL, similar to SOLPS
  - Most often fluid neutrals used
- Utilizes ad-hoc inputs of anomalous perpendicular transport:
  - *D* particle diffusion, $\chi_e$ electron energy diffusion, $\chi_i$ ion energy diffusion
- Relatively fast on small grids, ~10 s using pyUEDGE
- Here used 18 x 10 grid

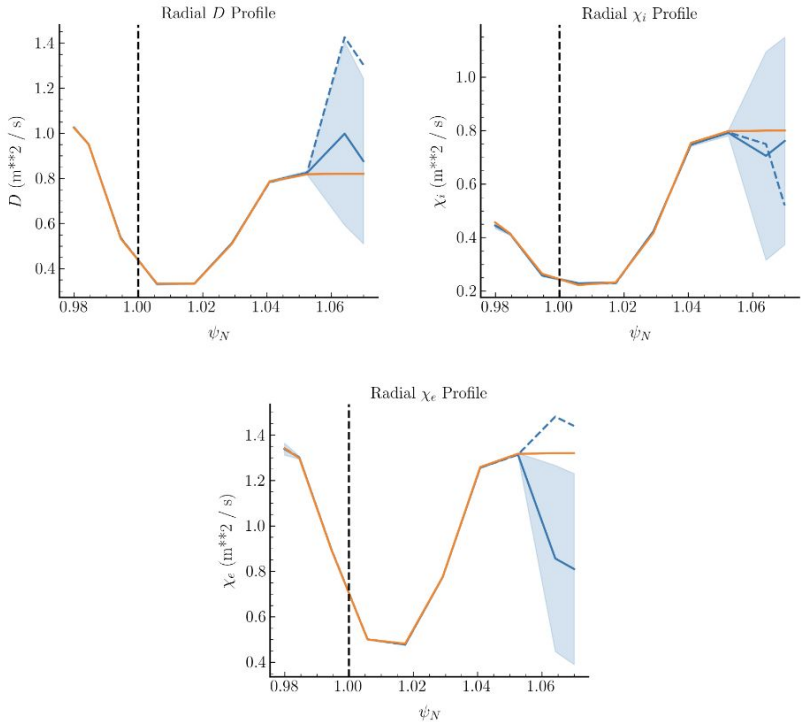


# Results for NPE

$N$ = 10,000 simulations total
120 hidden features, 10 transforms

**Orange:** Actual ($\boldsymbol{\theta}_o$)
**Dashed:** SNPE maximum-a-posteriori

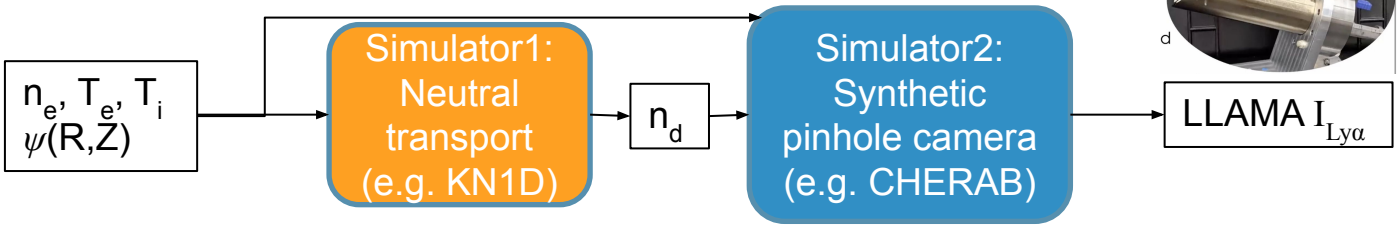
**Solid:** SNPE expected value (95% C.I., 10^5 samples)

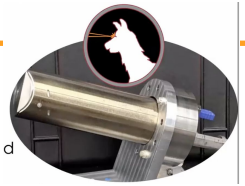


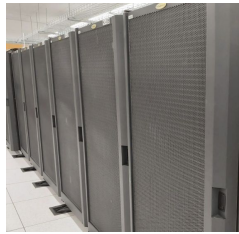
T. Khare, 2024, to be submitted

# Simulation-based Inference for diagnostic interpretation

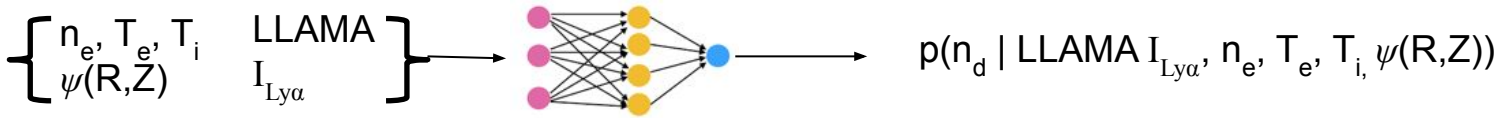
**(1) Generate 1000's synthetic LLAMA signals, with added noise**

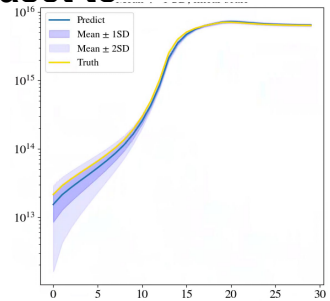




**(2) Train specially designed neural networks (e.g. normalizing flows) on the synthetic dataset to learn the inverse mapping, extracting $n_d$ from LLAMA measurements including uncertainties (posterior)**



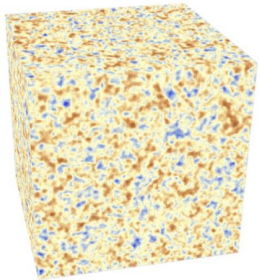

**(3) During operations, "inference" with neural network to extract $n_d$ from LLAMA measurements is very fast (~ms - s) and run locally, visualize between discharges $n_d$ over entire discharge**



**KEY INSIGHT: NN can enable fast simulation to experiment comparison**

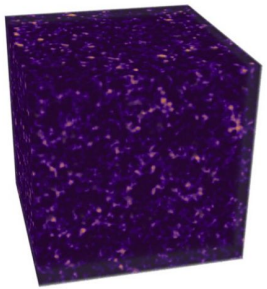
# High dimensional data poses difficulties for SBI with discrete normalizing flows

- Strength but also drawback of discrete normalizing flows is that the bijective nature *is hardcoded in the model,* requiring clever architecture choices
- In practice, Jacobian between layers leads to input dimension constraints to ~O(30) due to memory and computation constraints
- Example: Infer the 3-D density field of the early universe θ given observations **x** of the dark matter distribution at low redshift, where both are 3-D on a $128^3$ voxel grid [Legin, arxiv 2023]
  - How to overcome the dimension limitations of discrete normalizing flows?



$\theta \in \mathbb{R}^{128^3}$

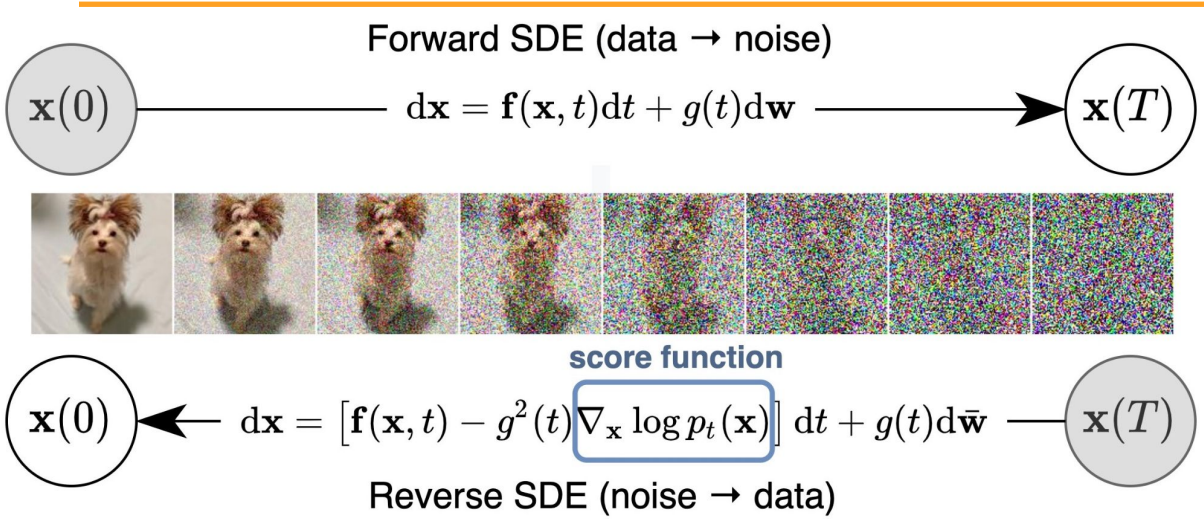
(density early universe)

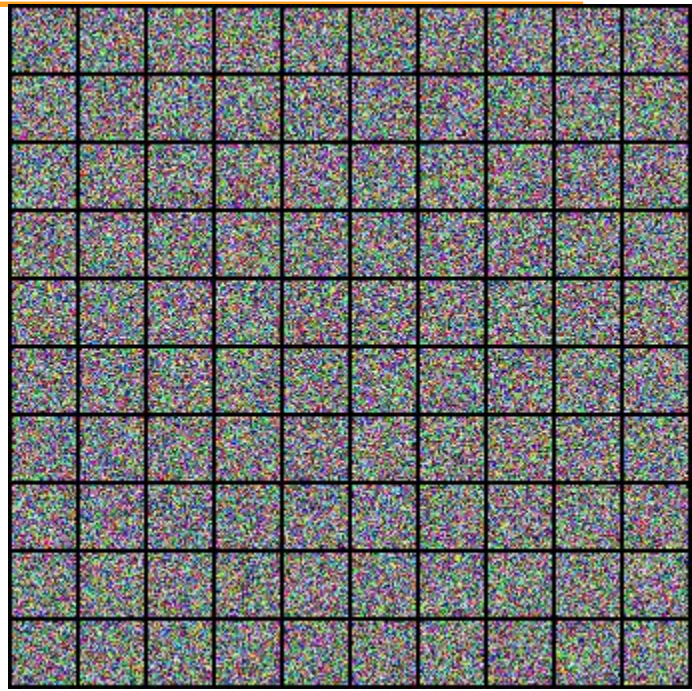


$x \in \mathbb{R}^{128^3}$

(density today)

# Diffusion models



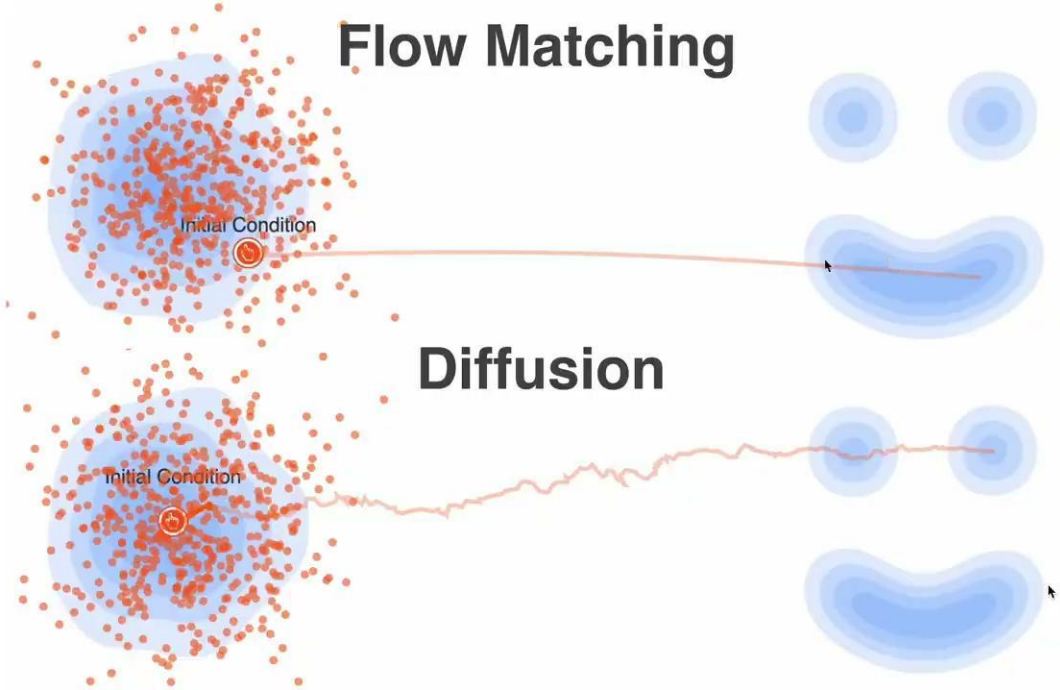
Y. Song, 2021

Learn score function, use Langevin dynamics to generate new samples. Overcomes need to have Jacobian calc at each layer, allows using wide variety of architectures (usually UNet, Transformers)

# Flow matching similar to diffusion, but learns deterministic flow field, “simulation-free” (a.k.a no ODE solve)

$$\frac{dx_t}{dt} = v_\theta(x_t, t), \qquad x_{t=0} \sim p_0$$

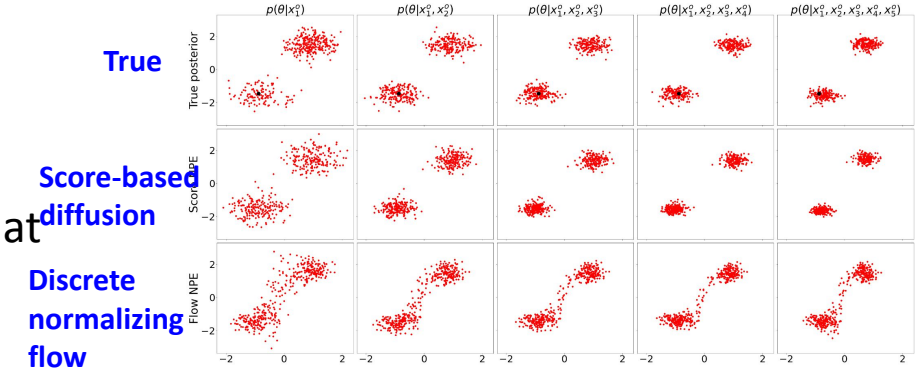
$$\mathbf{dx} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

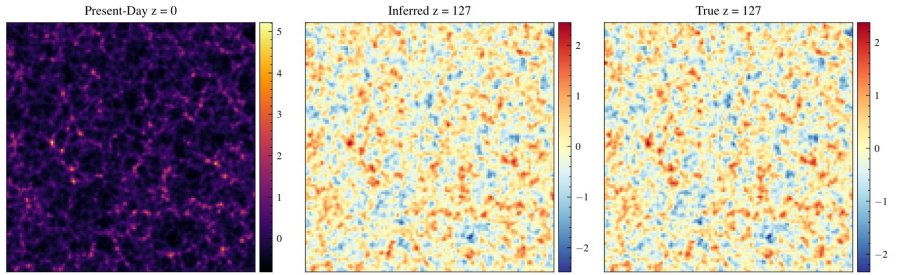


# Score-based models in simulation-based inference (SBI)

T. Geffner, arxiv 2022

- Score-based diffusion models learn the exact likelihood, and therefore can be used in SBI [Geffner 2022]
- Applied to inference of $128^3$ 3-D density field at beginning of universe [Legin 2023] (though synthetic sanity check, no real observation used), used 2,000 $(1\ \mathrm{Gpc/h})^3$ dark matter N-body simulations



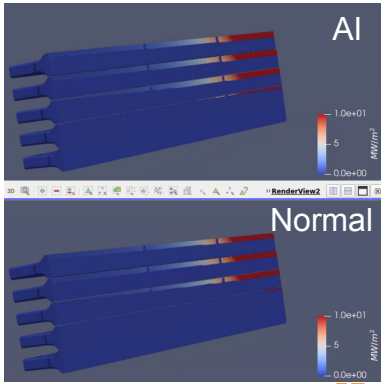




R. Legin, arxiv 2023

# Digital twin extension to SBI based on diffusion/flow matching

Rotu arxiv 2024

- Digital twin is a “hybrid” data+sim model, supplementing modsim with physical device data for better predictive capabilities [NASEM 2024]
- Diffusion/flow matching very flexible models, with steering capability due to semantic concept learning
  - Can be applied in a similar way to update models in digital twin manner
    - Potentially sacrifice some extrapolation capability to new devices for gains in predictive capability for current (control, planning, etc.)

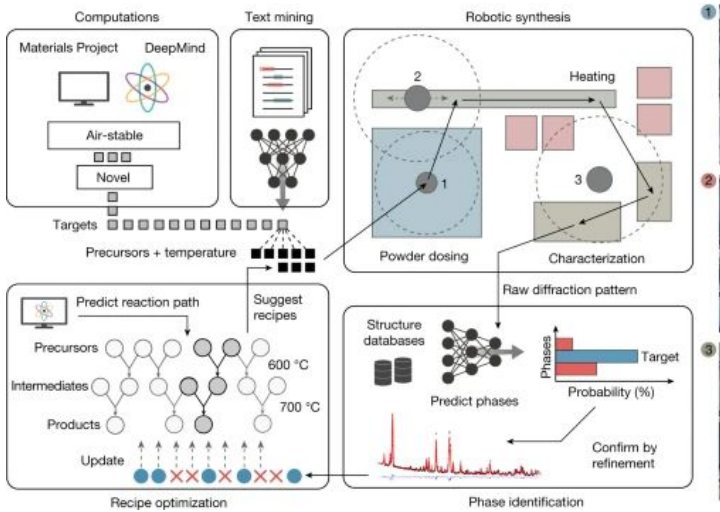
# How can we best use AI for scientific discovery with experiment?

This talk explores two frontiers:

1. **Connecting Models & Reality:** Using generative AI to rigorously compare simulations with experimental data.
2. **Automating the Workflow:** Using agentic AI to assist in the process of research itself.

# Automating discovery process

- Berkeley A*lab fully automated robotic laboratory for materials discovery, connected with active learning and LLM to suggest novel materials and synthesis autonomously
- Initial claim of 41 novel compounds synthesized
  - However, heavily disputed in the community on novelty of materials
- Cautionary tale on agentic AI driving discovery. Can these tools truly discover novelty?




Szymanski, Nature 2023

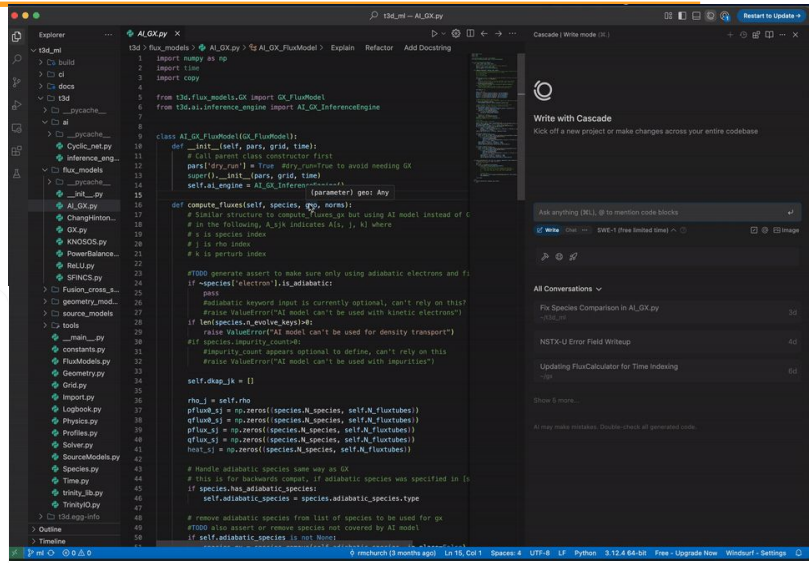
# LLMs expanding from chatbots to include reasoning, agentic capabilities for research and code

Windsurf



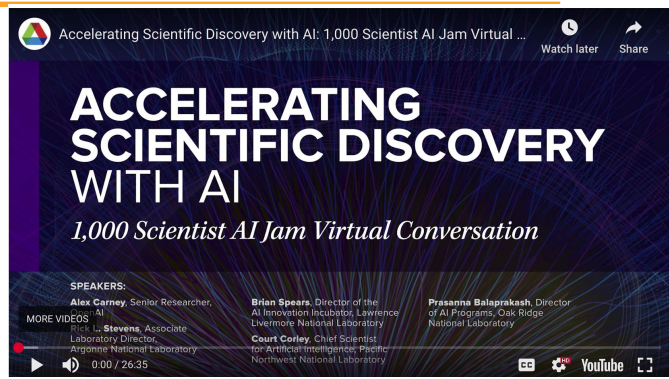




# DOE AI-jam session: a tale of two cities

- DOE held a 1000 scientist jam session at national lab sites in Feb 2025
  - OpenAI provided access to newest (at the time) AI reasoning models: o3, o4-mini-high, and agentic Deep Research
  - Scientist brought their problems to probe advanced AI reasoning models on scientific problems important to DOE
- For certain classes of problems, it was the best of times
  - **AI models scored an A- on PhD qualifying exam for plasma physics students in the Princeton Astrophysics department**
- For other classes of problems, it was the worst of times
  - **AI models struggled with open-ended problems requiring high degree of creativity (not necessarily knowledge)**




A toroidal plasma with a pressure gradient in the radial direction and a parallel electric field E_parallel will relax via Coulomb collisions. In this problem, consider the associated "neoclassical" transport in the long mean-free-path "banana regime," and ignore, for simplicity, any temperature gradients.

What is the trapped-particle flux in terms of these driving forces? Specifically, provide a simplified (heuristic) derivation of the coefficients for the pressure gradient (-Te * dn/dr) and -n*E_parallel driving forces. Use the conservation of canonical angular momentum to estimate the inward radial velocity of the trapped particles.

In the long mean-free-path "banana regime" of neoclassical transport, the flux of trapped particles is driven by both the radial pressure gradient and the parallel electric field $E_\parallel$. The key steps to estimate the trapped-particle flux involve using conservation of canonical angular momentum and estimating the radial drift of the trapped particles.

**1. Conservation of Canonical Angular Momentum**

For a toroidal plasma, the canonical angular momentum $P_\phi$ of a particle in the presence of a toroidal magnetic field $B_\phi$ and a poloidal magnetic field $B_\theta$ is:

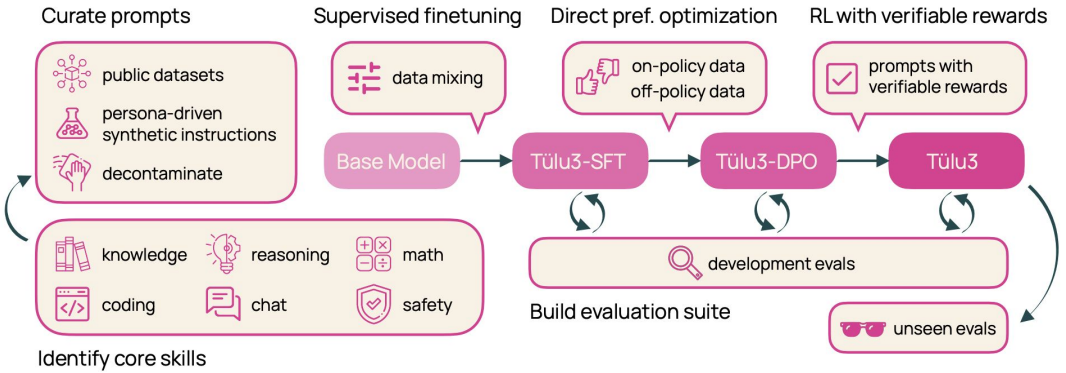
$$P_\phi = mRv_\parallel + q\psi$$

where:

- $R$ is the major radius,

# Updating models for specific capabilities such as reasoning, tool use, etc.

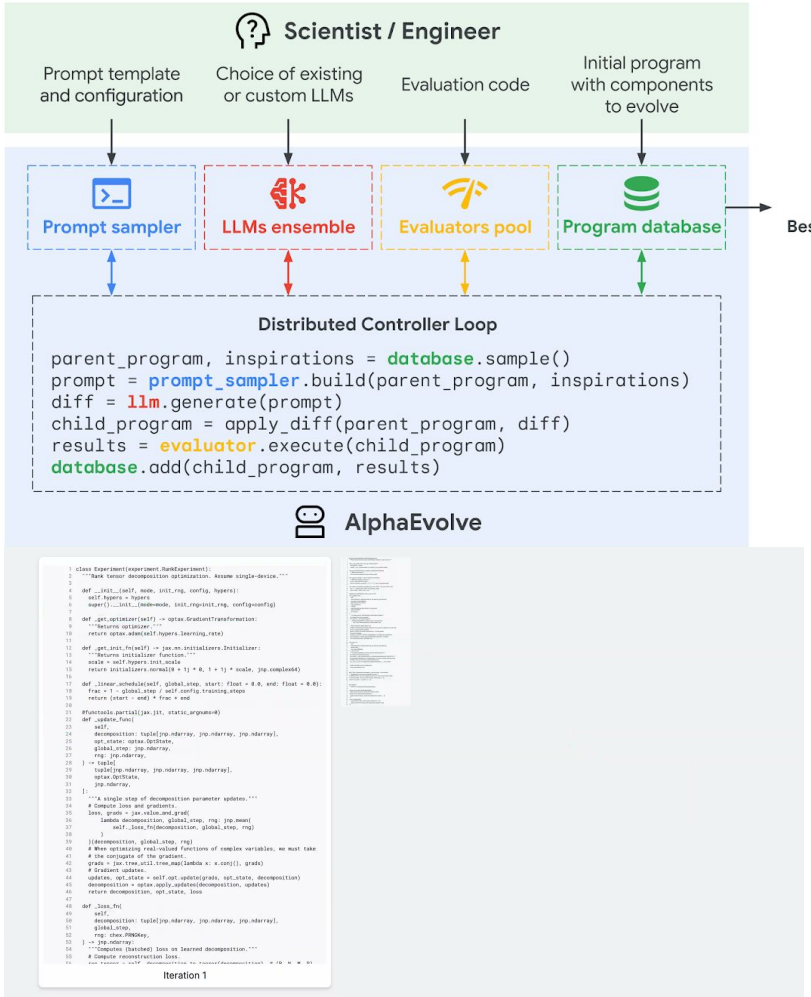
- Increasing capabilities of LLMs have come with larger trained base models AND post-training updates
- Reinforcement Learning with Verifiable Rewards (RLVR) and similar techniques enabling reasoning capability on domains with deterministic, verifiable objectives (math, coding)
  - But RL is usually sample hungry, and difficult to build up posttraining datasets necessary for specific skills




N. Lambert 2025

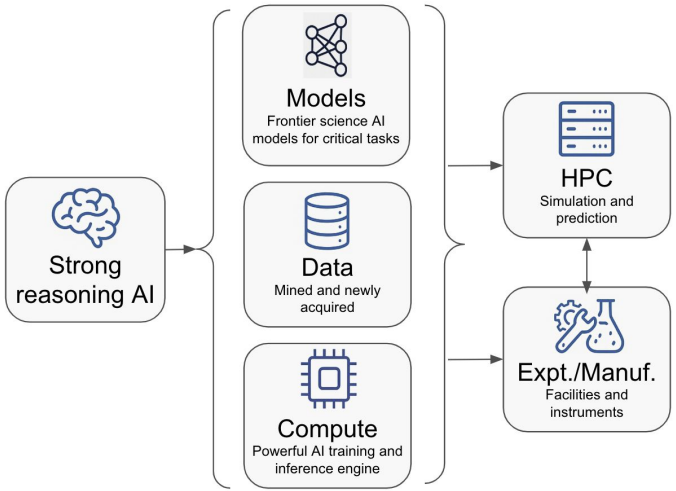
# AlphaEvolve

- AI agents coupled with Google LLM (Gemini) and evolutionary algorithms to generate novel algorithms
  - Found algorithm to reduce number of operations in 4 x 4 matrix multiply, datacenter
- "While AlphaEvolve is currently being applied across math and computing, its general nature means it can be applied to any problem whose solution can be described as an algorithm, and automatically verified."
- Searches space of programs much more intelligently, enables novel discovery



# How to leverage these capabilities?

- First, AlphaEvolve not open (yet) but there is Sign Up list (sign up!)
- Many more tools from different vendors will be available (e.g. Microsoft Discovery recently released)
- Key need is defining correctly reward signal from fuzzy real-world reward functions
  - Domain scientist in a natural position to do
- Connecting strong reasoning AI with our tools of HPC simulation and experimental facilities can accelerate science discovery




Adopted from B. Spears

# Conclusion

- “The proliferation of **AI** tools in science **risks** introducing a phase of scientific enquiry in which **we produce more but understand less.**” - Messeri, Nature 2024
- “Those who trust AI tools to overcome their own cognitive limitations become susceptible to illusions of understanding” - Messeri Nature 2024

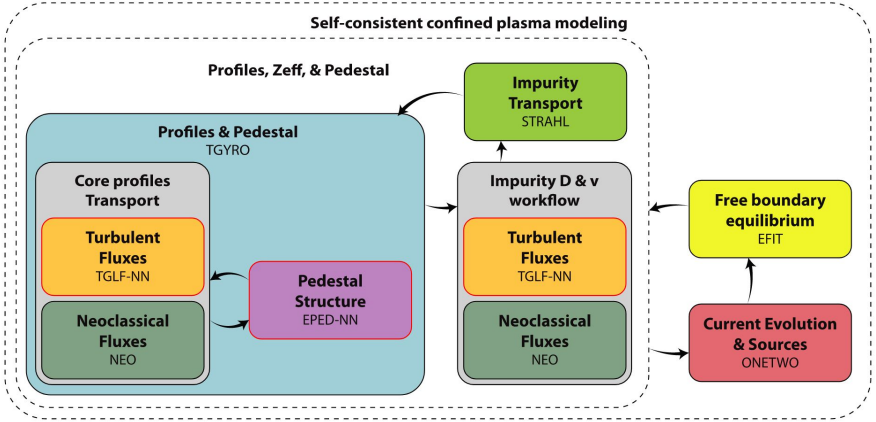
- AI tools offer the promise of aiding researchers in multiple science fields to improve the process of scientific discovery
- Using AI with simulation-based inference to broadly compare and verify mechanistic simulation with experiment can greatly expand our ability to discover regions of mismatch, point to areas to improve -> code validation accelerated
- Future looking capabilities of agentic AI semi-automating discovery is closer, need domain scientist to engage in rewards signals, connecting tools, and human-in-the-loop for finding novelty

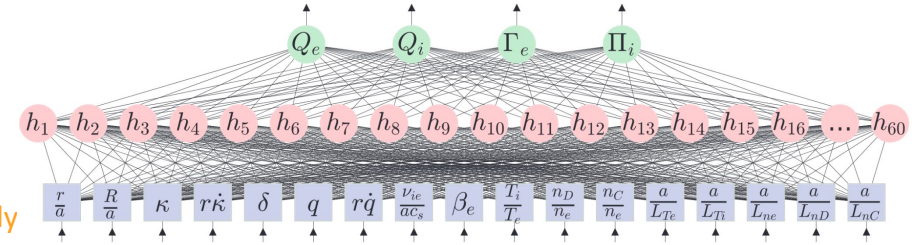
# References

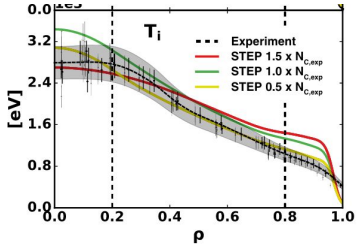

- https://arxiv.org/pdf/2202.00728.pdf
- https://sites.google.com/view/optimizing-designs

# Basic fully-connected networks for profile transport

- Sometimes all you need (throw a neural network at it), especially when scalar inputs/outputs desired or possible
- If you can reduce what the ML model needs to predict, try first
  - When higher-fidelity needed, go a little further…



# Backpropagation: the learning algorithm

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial \ell}{\partial \boldsymbol{\theta}}$$

- Backpropagation uses chain rule to determine how weights should change given an output loss. Propagate error backwards, calculating weight updates
- Most deep learning frameworks use autodifferentiation to accurately calculate gradients, no need to specify by hand

$$\boldsymbol{\theta} = \left\{ W^{(1)}, \mathbf{b}^{(1)}, ..., W^{(L)}, \mathbf{b}^{(L)} \right\}$$

$$\mathbf{z}^{(k)} = W^{(k)} \mathbf{y}^{(k-1)} + \mathbf{b}^{(k)}$$

$$\frac{\partial \ell}{\partial W^{(k)}} = \frac{\partial \ell}{\partial \mathbf{z}^{(k)}} \cdot \left[ \mathbf{y}^{(k-1)} \right]^T$$

$$\frac{\partial \ell}{\partial \mathbf{b}^{(k)}} = \frac{\partial \ell}{\partial \mathbf{z}^{(k)}}$$

$$\frac{\partial \ell}{\partial \mathbf{z}^{(k-1)}} = \left[ W^{(k)} \right]^T \cdot \frac{\partial \ell}{\partial \mathbf{z}^{(k)}} \odot \sigma^{(k-1)\prime}(\mathbf{z}^{(k-1)})$$

Rumelhart, “Learning representations by back-propagating errors”, Nature 1986