Accelerating computational fluid dynamics with deep learning

Stephan Hoyer
twitter.com/shoyer

Google Applied Science
g.co/research/gas

Google Research

ECMWF ML workshop
April 1, 2022

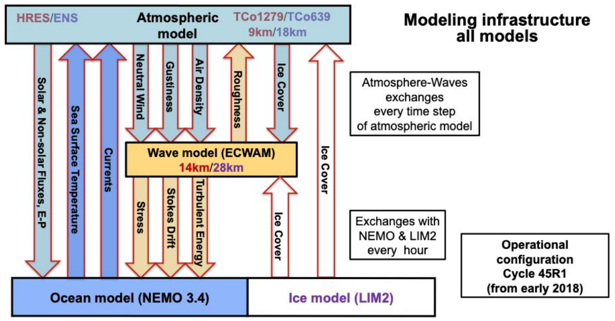
Deep learning has had a transformative impact on Google (and the tech industry) over the past decade

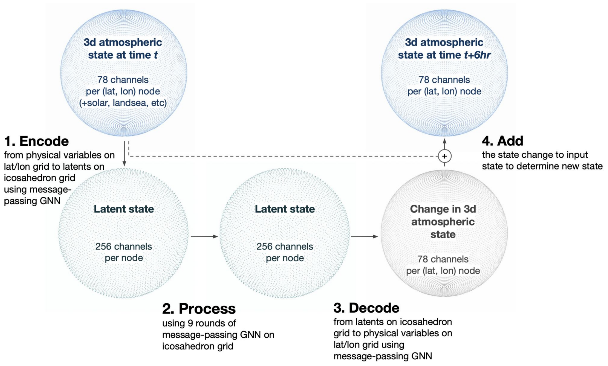
Can it also transform computational science?

“Pure” deep learning offers a drastic alternative to numerical weather prediction

ECMWF Integrated Forecast System
1,000+ person-years of effort
~1 million of lines of code
~10,000 CPU-hours per forecast
~7 days of accurate forecasts
Built upon known physical laws

Keisler 2022 Graph Neural Network
<1 person-year of effort
1000s of lines of code
1 GPU-second per forecast
~6 days of accurate forecasts
Fit to loads of data (from IFS)





Google

The “differentiable programming” paradigm of deep learning offers a potential middle path

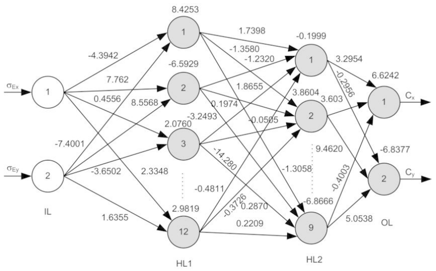
Numerical methods

for interpretability, generalization and extensibility

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = \sigma$$

Neural networks

for fast approximation



Today’s talk: What does “differentiable programming” offer for computational fluid dynamics and weather modeling?

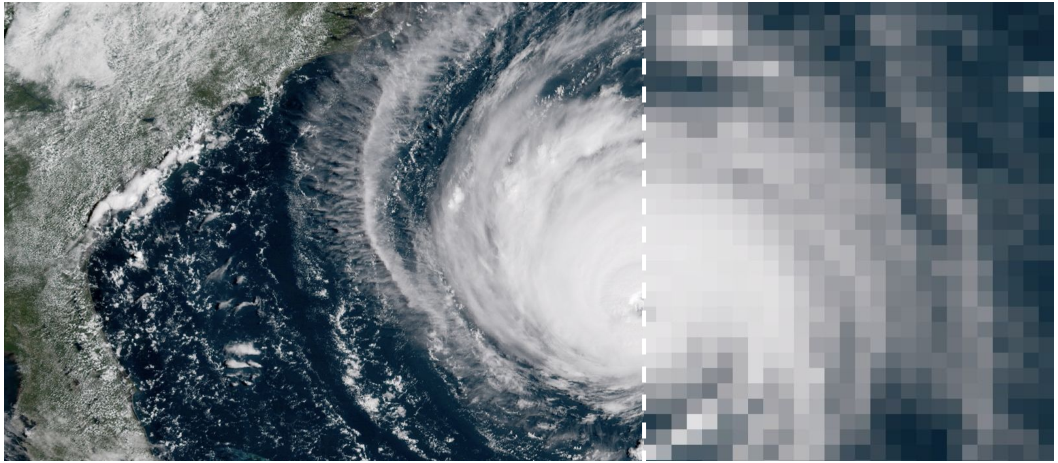
Traditional simulation methods are accurate, but slow

Navier-Stokes equations:

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

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

Challenge: Need $\Delta x \rightarrow 0$ for accuracy, but runtime is $O(1/\Delta x^4)$

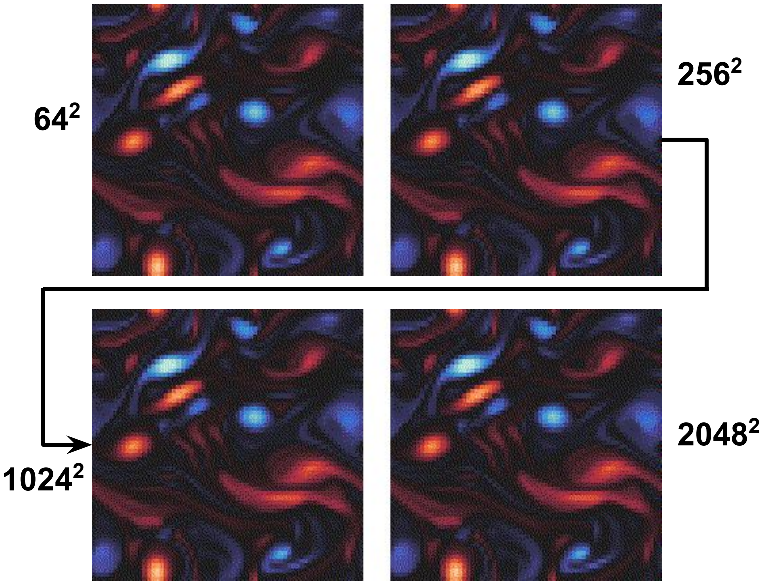


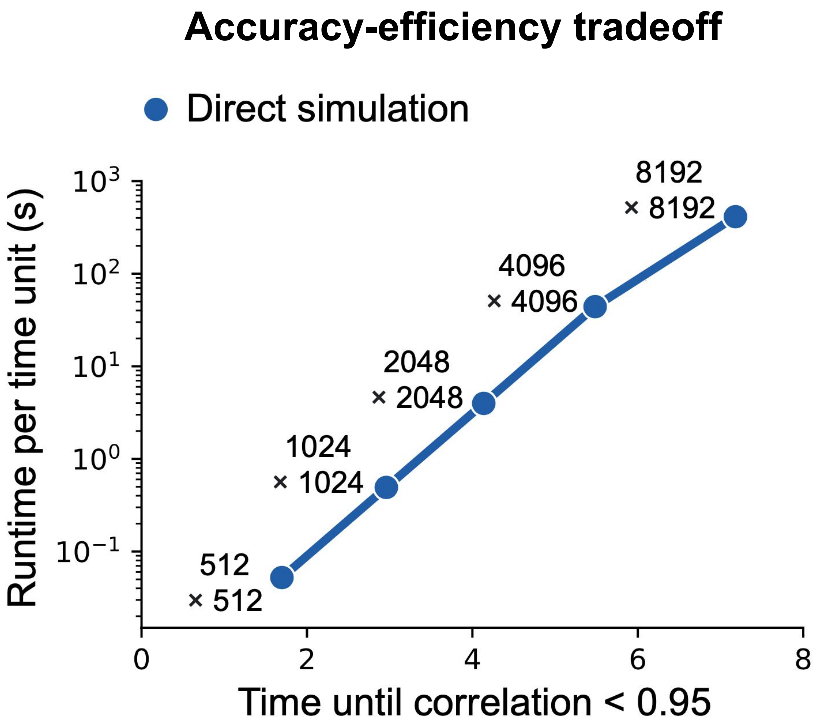
Satellite photo *Weather forecast*

Google

Quantifying effects of resolution

- Higher resolution → more accurate
- Lower resolution → faster simulation



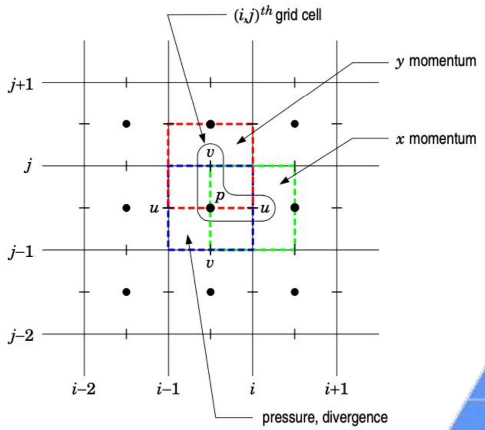


Our work is built upon the "JAX-CFD" library for computational fluid dynamics

JAX-CFD is:

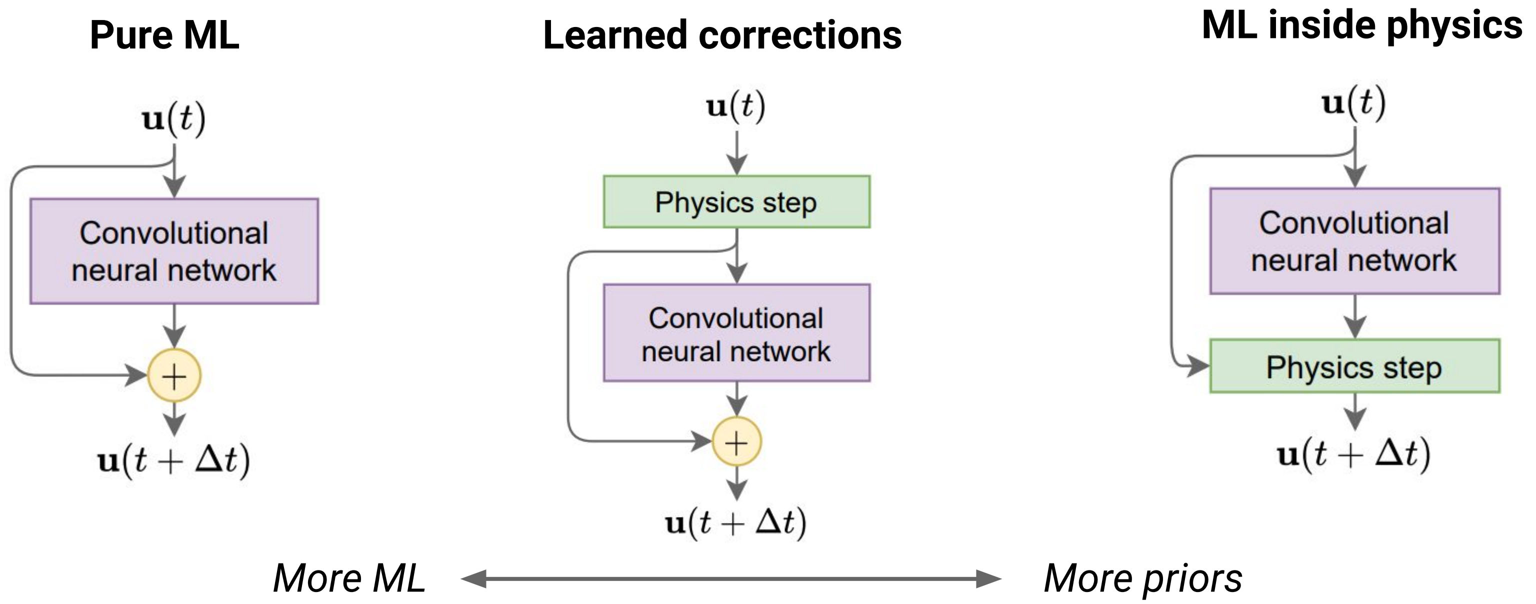
- Fully **differentiable**
- **GPU/TPU native**
- Designed for **hybrid ML/physics** models (e.g., learned interpolation)
- Based on relatively **simple numerics** (first/second order)

github.com/google/jax-cfd



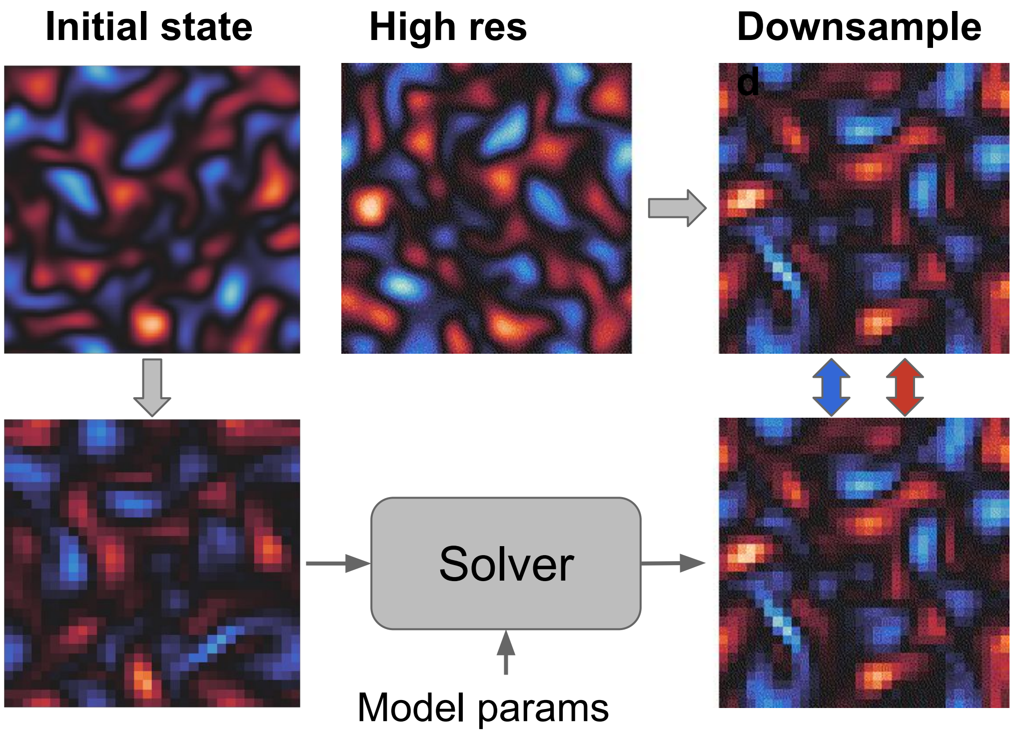
Google

We consider a range of hybrid ML/physics models

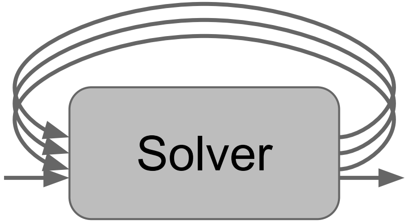


All of these models have two key “inductive biases”: (1) locality and (2) translation invariance

Our training setup: learn to simulate on coarse grids

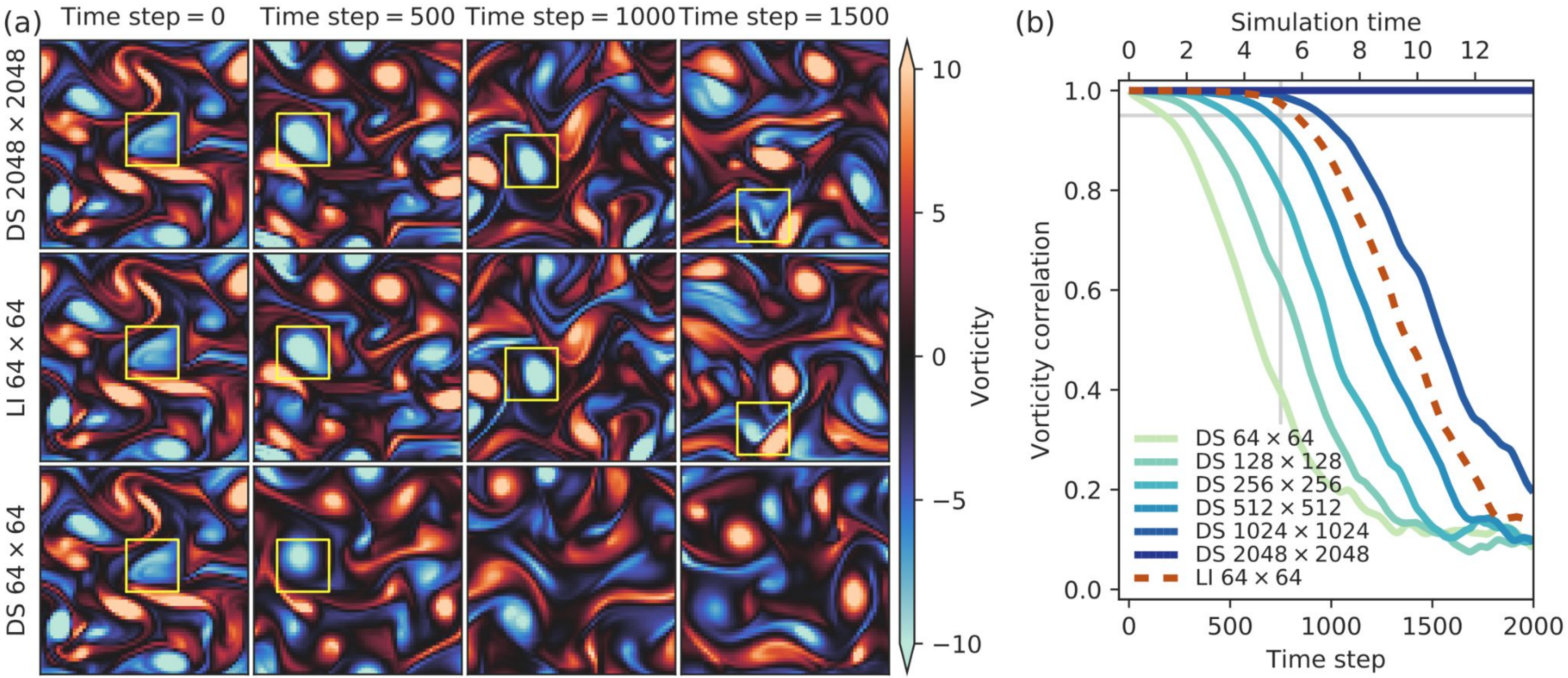


Unrolled in time for end-to-end training:



$$L(x, y) = \sum_{i=1}^{T} \mathrm{MSE}\left(\mathbf{u}_{\mathrm{exact}}(t_i), \mathbf{u}_{\mathrm{pred}}(t_i)\right)$$

Performance on the "test dataset"



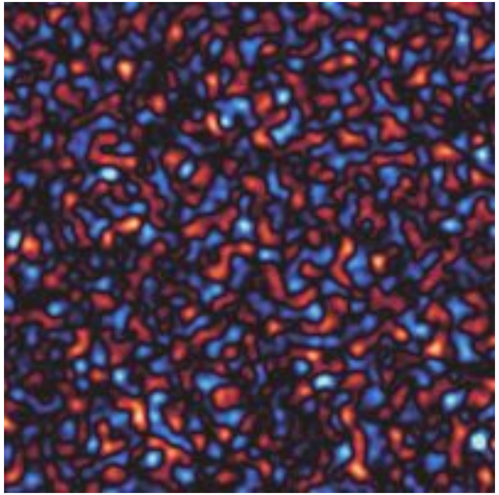
Learned Interpolations model matches CFD at ~12x higher resolution

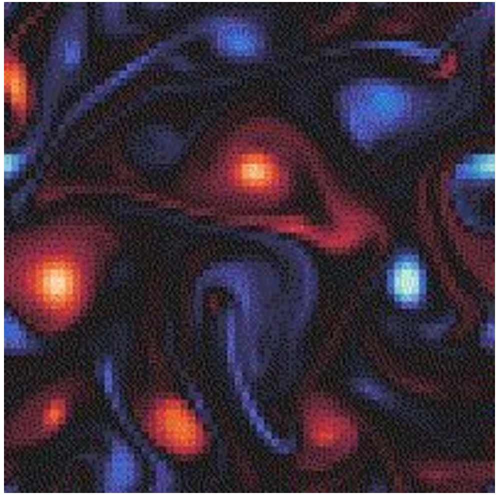
Google

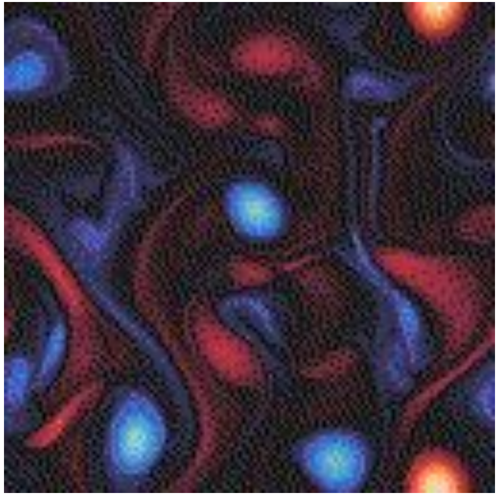
Evaluation setup

Metrics:
- Pointwise accuracy
- Statistical consistency
- Numerical stability

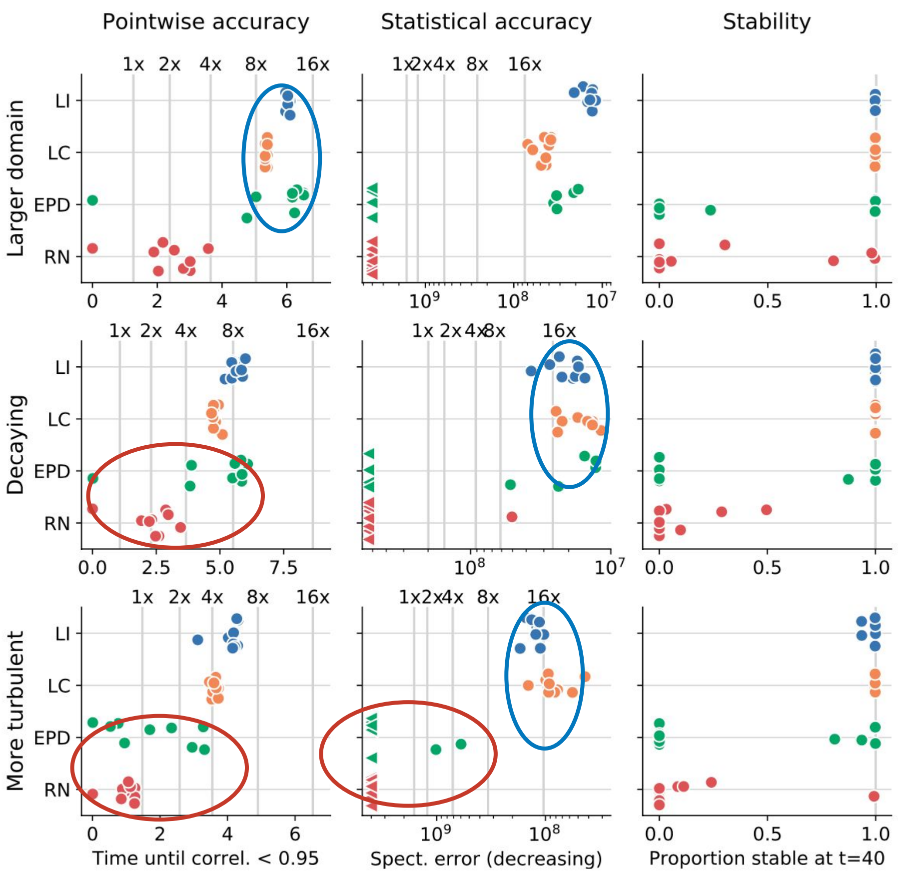
Generalization datasets:
1. Larger domain
2. Decaying turbulence
3. Higher reynolds numbers (Re=4k)







Comparing models along ML axis on eval datasets



Metrics:

1. Pointwise accuracy (T until correlation < 0.95)
2. Statistical accuracy (Error in energy spectrum)
3. Stability (Proportion of stable)

Each model trained 9 times

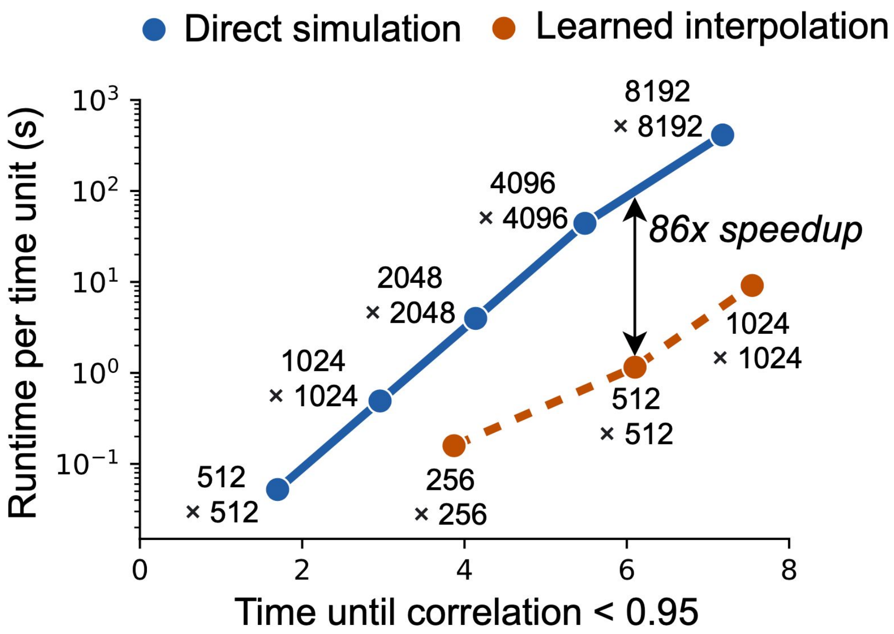
Takeaways:

- Physics-ML hybrids perform best overall
- Pure ML models don’t generalize well

2D turbulence: efficiency and the pareto frontier

Benchmarking results:

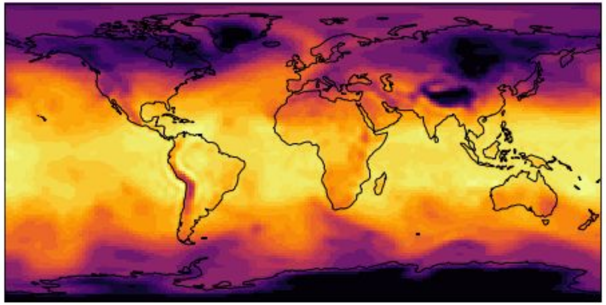
1. Our method uses 150x more FLOPS
2. But is only 12x slower at same resolution on Google TPUs
3. Overall: ~80x faster for the same accuracy (10^3 / 12)



Our current focus: a differentiable, accelerator-native GCM

Can **deep learning** inside an atmospheric GCM improve global weather forecasts?

What is the **optimal combination** of numerical methods and machine learning?



“Dynamics”
Accelerate this stuff

Primitive equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0$$

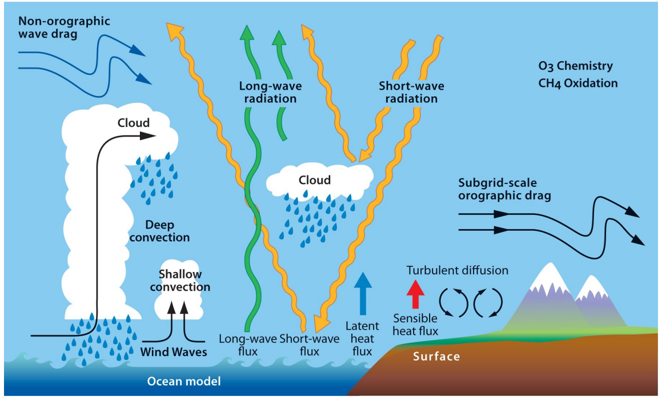
$$\frac{D\mathbf{V}}{Dt} = -2\mathbf{\Omega} \times \mathbf{V} - \mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{r}) - \nabla \phi_a - \alpha \nabla p - \alpha \nabla \cdot \mathbf{F}$$

$$\frac{D}{Dt}(c_v T) + p\frac{D\alpha}{Dt} = -\alpha \nabla \cdot (\mathbf{R} + \mathbf{F}_S) + LC + \delta$$

$$\frac{Dq_v}{Dt} = g\frac{\partial F_{q_v}}{\partial p} - C$$

Discretization

“Physics”
Learn this stuff



Thanks for your attention!

Summary: **numerical methods** + **auto diff** + **accelerators** + **deep learning** = an amazing toolkit for scientific computing!

To learn more:

- *Paper*: Kochkov et al, Machine learning accelerated computational fluid dynamics (PNAS, 2021)
- *Code*: github.com/google/jax-cfd


Dmitrii Kochkov


Jamie Smith


Peter Norgaard


Yohai Bar-Sinai


Jason Hickey


Michael Brenner


Jiawei Zhuang


Ayya Alieva


Qing Wang