# Deep learning & differentiable simulations

## …for weather and climate modeling

Stephan Hoyer
Google Applied Science
Google Research

*ESiWACE2 Second Virtual Workshop on Emerging Technologies for Weather and Climate Modelling*
*7 October 2022*

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

Can it also transform computational science?

# Our thesis: deep learning software is also fantastically useful for computational science

1. **Neural networks** – powerful machine learning models
2. **Accelerator hardware** – fast code on GPU/TPU clusters
3. **Automatic differentiation** – gradient-based optimization
4. **Scientific computing** – write code like NumPy



# Within the “differentiable programming” paradigm, we can optimize simulations for end-to-end performance

## Numerical methods

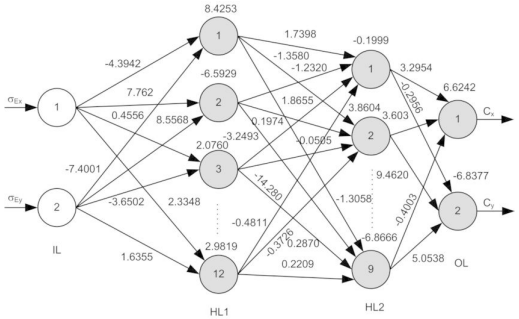
for interpretability, generalization and extensibility

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

+

## Neural networks

for data-driven approximations




**Key challenge**: this requires *differentiable simulators*

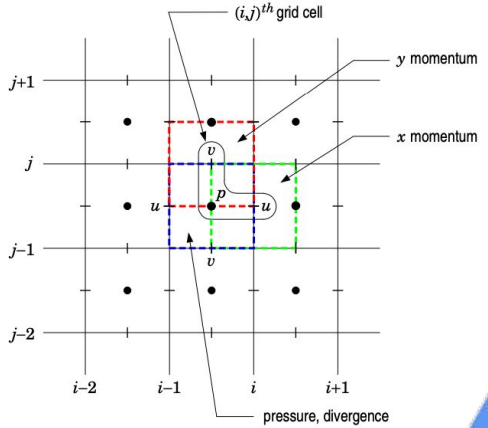
# Part 1: ML for computational fluid dynamics

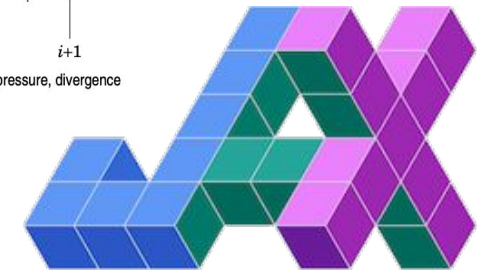
# Our work is built upon the “JAX-CFD” library for computational fluid dynamics

JAX-CFD is:

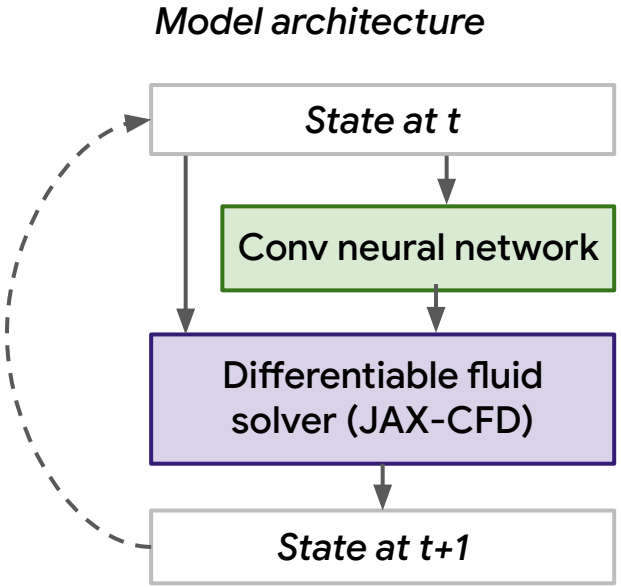
- Fully **differentiable**
- **GPU/TPU native**
- Designed for **hybrid ML/physics** models (e.g., learned interpolation)
- Based on **simple numerics** (finite volume and pseudo-spectral)

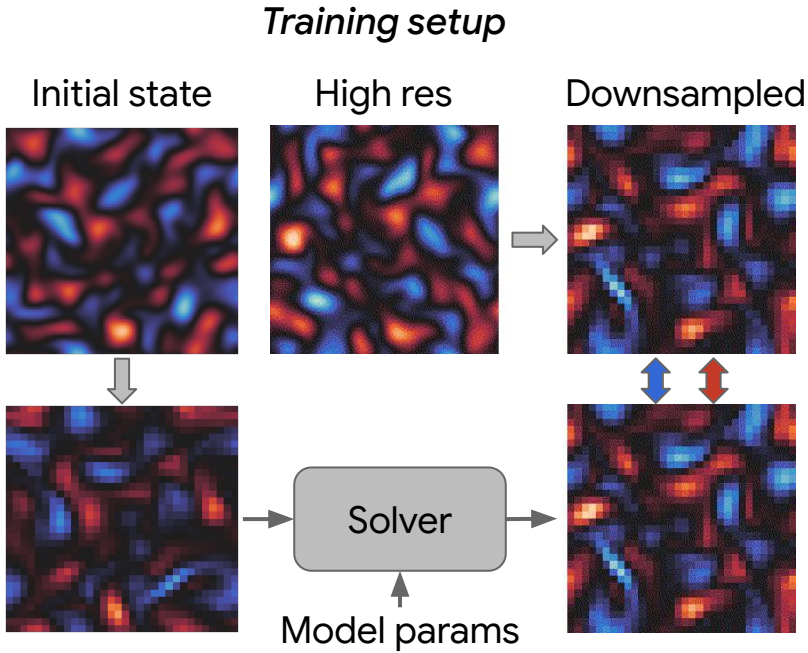
[github.com/google/jax-cfd](github.com/google/jax-cfd)



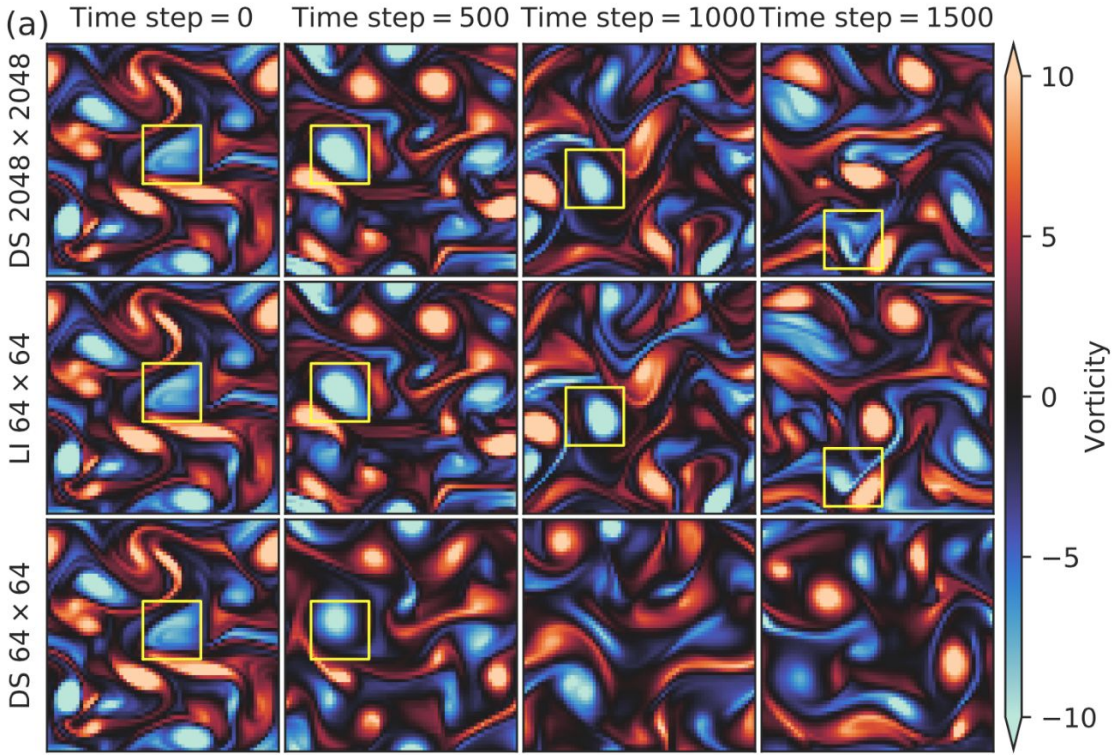
# Our approach: train ML-physics models to accurately simulate on coarse grids

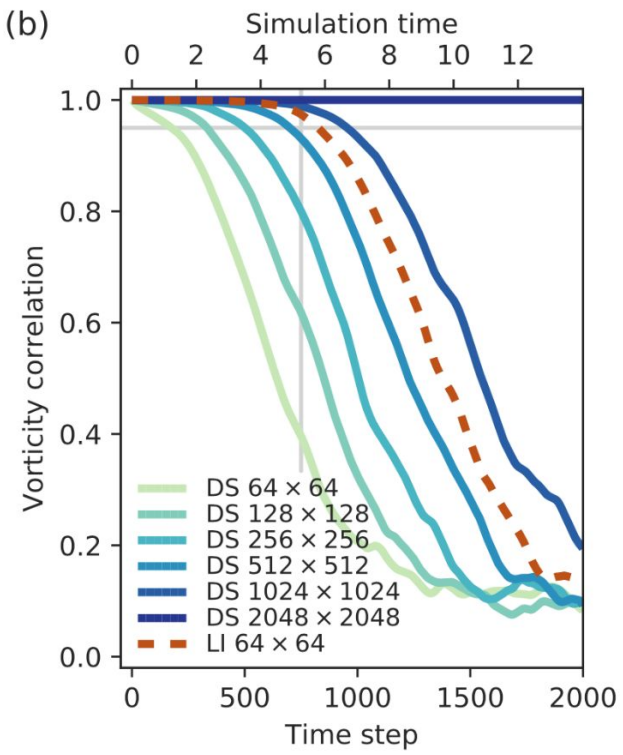



Kochkov et al (PNAS, 2021)

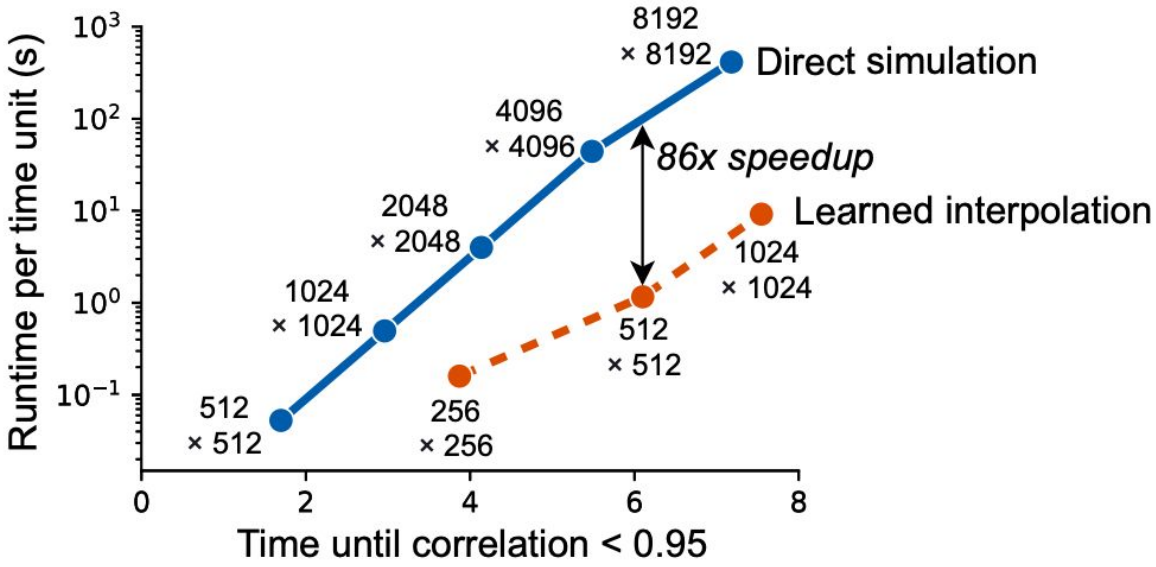
# Our hybrid model have comparable accuracy to 10x higher resolution simulations




Kochkov et al (PNAS, 2021)

# Our hybrid ML-physics models can improve both accuracy *and* speed

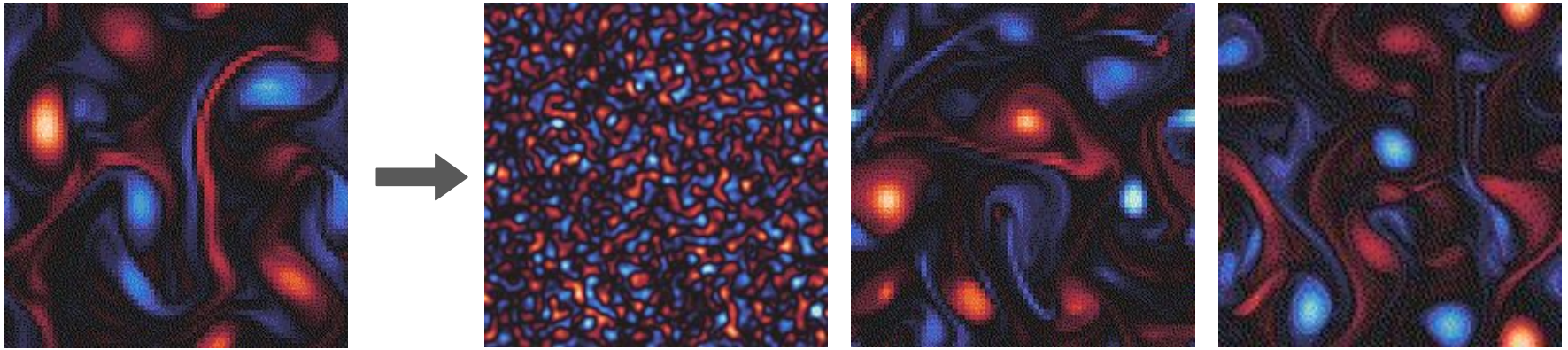



Kochkov et al (PNAS, 2021)

# ...and unlike similarly accurate pure ML models, our hybrid models *generalize*

*Training dataset*

*Generalization tests*



Forced turbulence

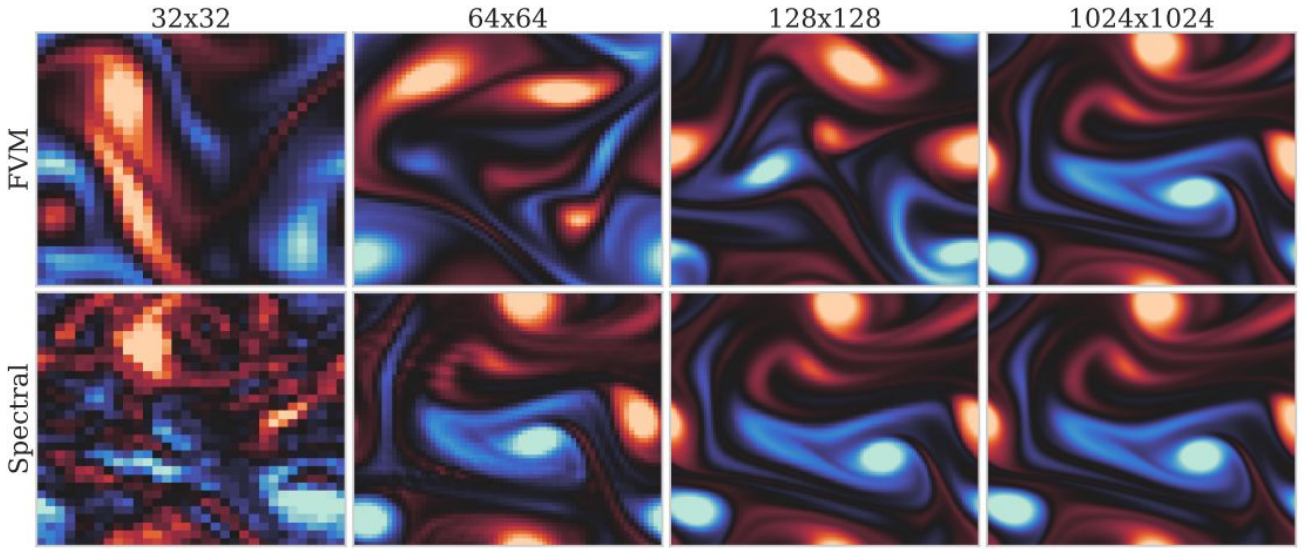
Larger domain

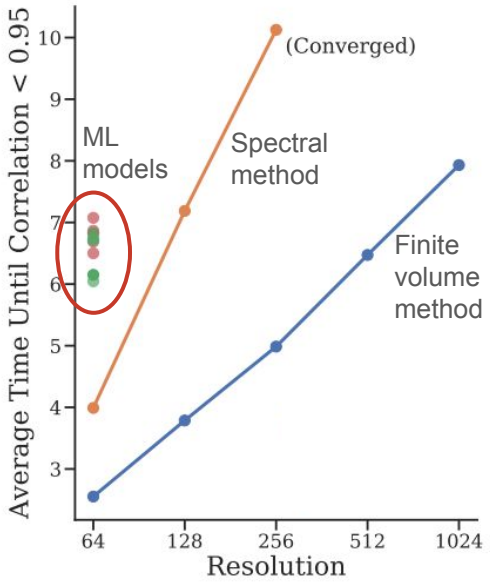
Decaying

More complex

Kochkov et al (PNAS, 2021)

# We can also improve spectral solvers, but it's harder

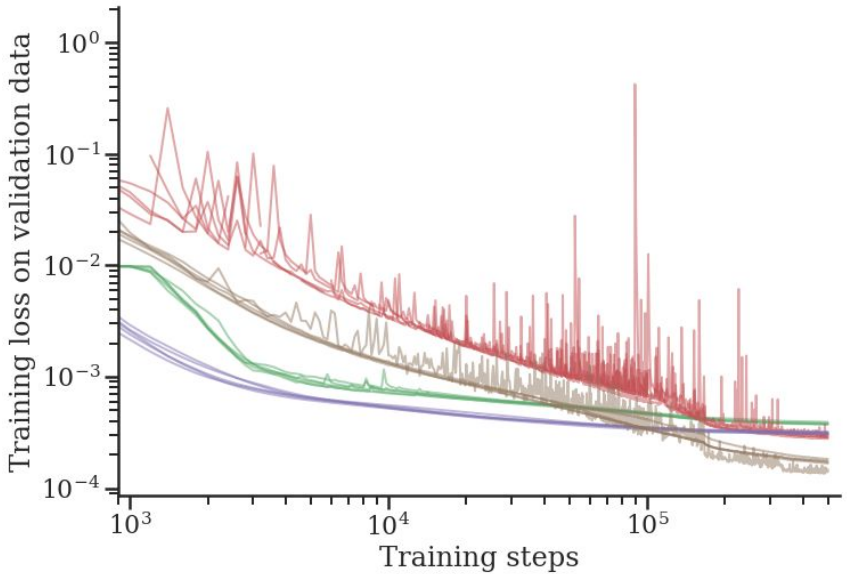


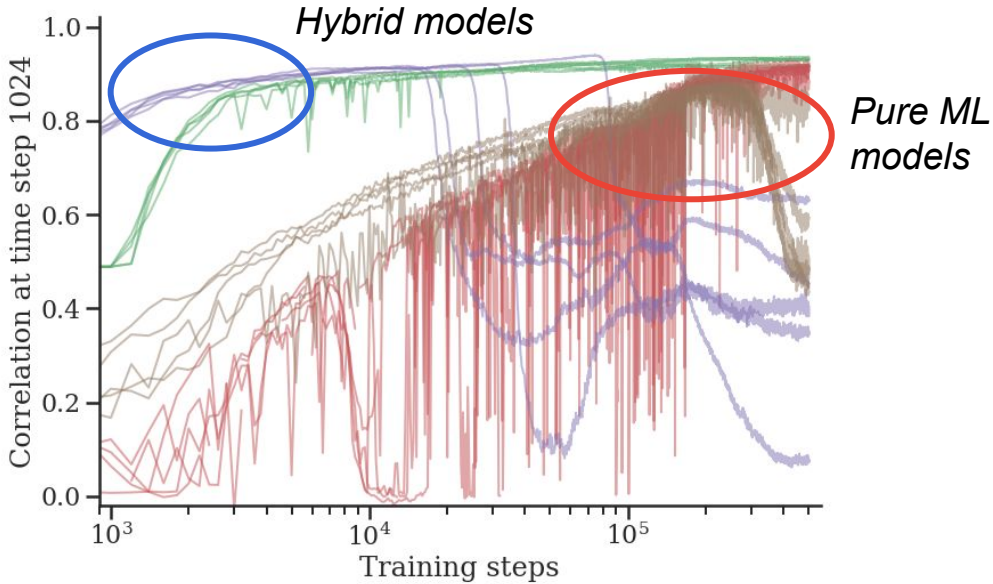

Dresdner et al, arXiv:2207.00556

# Hybrid models are faster to train than pure ML, but still finicky

Training loss (32 time steps)




Online performance (1024 time steps)



# Part 2: ML for global weather

# Our current focus: hybrid modeling with a differentiable atmospheric General Circulation Model

**“Dynamics”**
Accelerate this stuff

*Primitive equations*

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

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

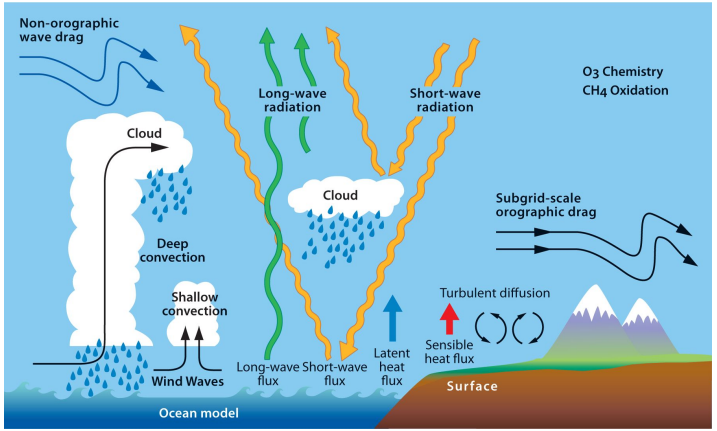
$$\frac{D}{Dt}(c_v T) + p\frac{D\alpha}{Dt} = -\alpha \nabla \cdot (\mathbf{R} + \mathbf{F}_S) + LC + \delta$$

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

*Discretization*

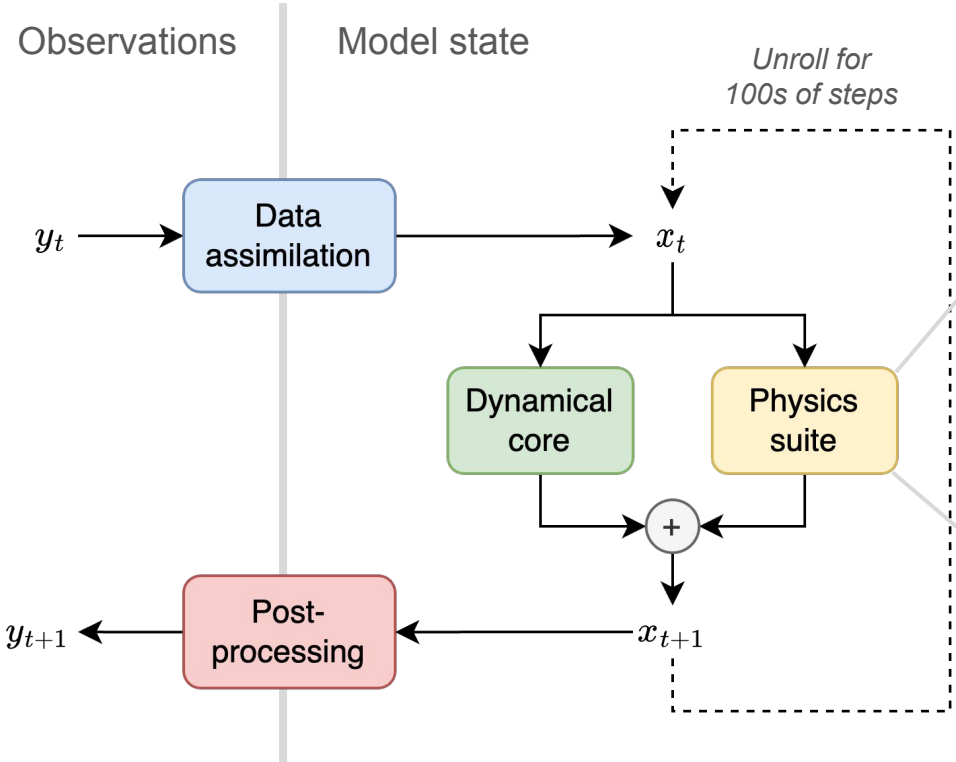
Moves stuff across space.

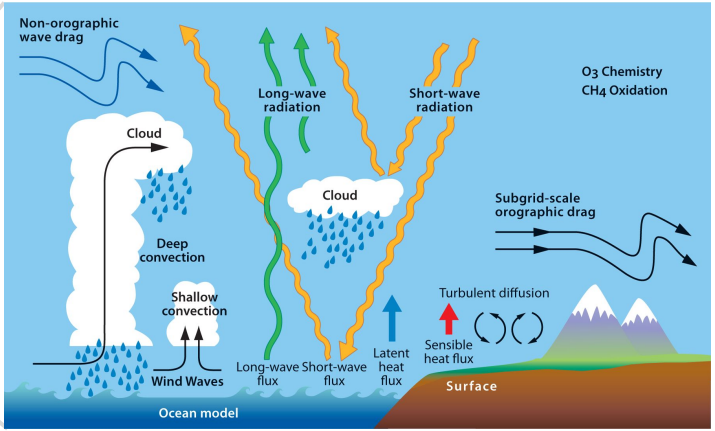
**“Physics”**
Learn this stuff



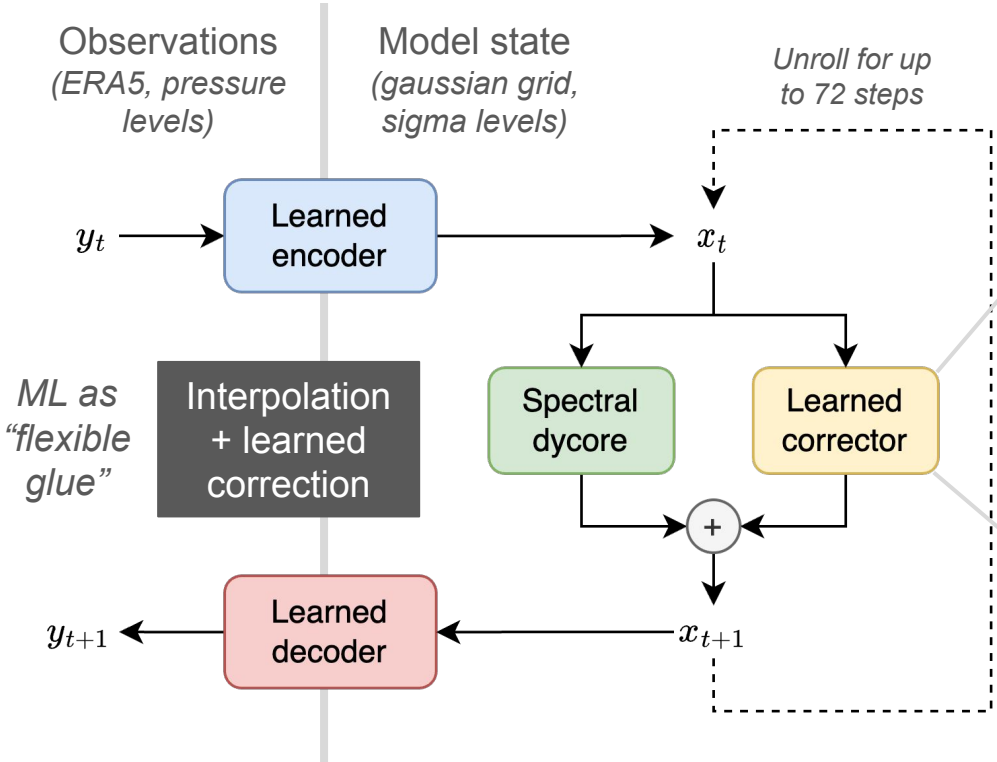

Independent in each vertical column.

# Structure of standard weather & climate models



# Structure of our hybrid deep learning model




## Single-column neural network

*Inputs*: T, u, v, q, log(sp), vor, div, gradients, surface features, incident solar radiation…

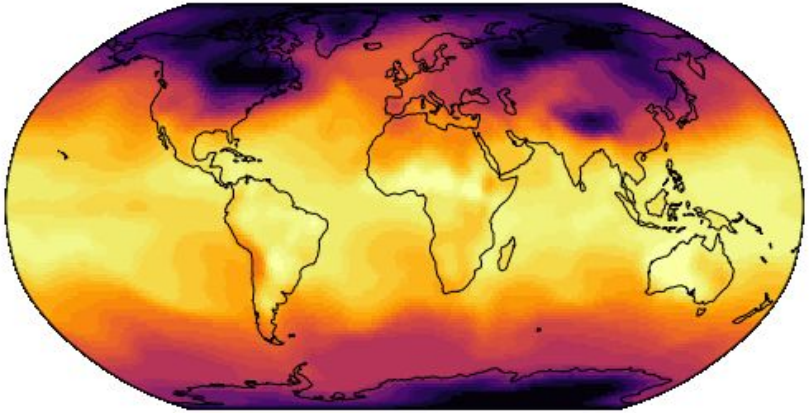
*Outputs*: ΔT, Δu, Δv, Δq

*"End-to-end" learning is key for stable physics emulation*

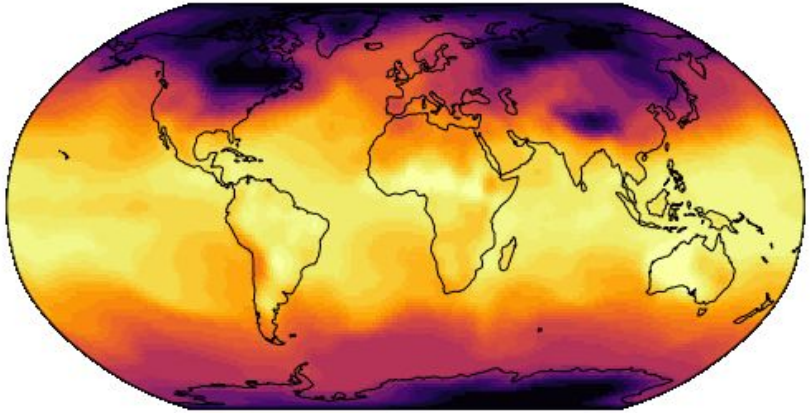
# Preliminary results with our hybrid deep learning model

Surface temperature (5 days)

ERA5 (smoothed)



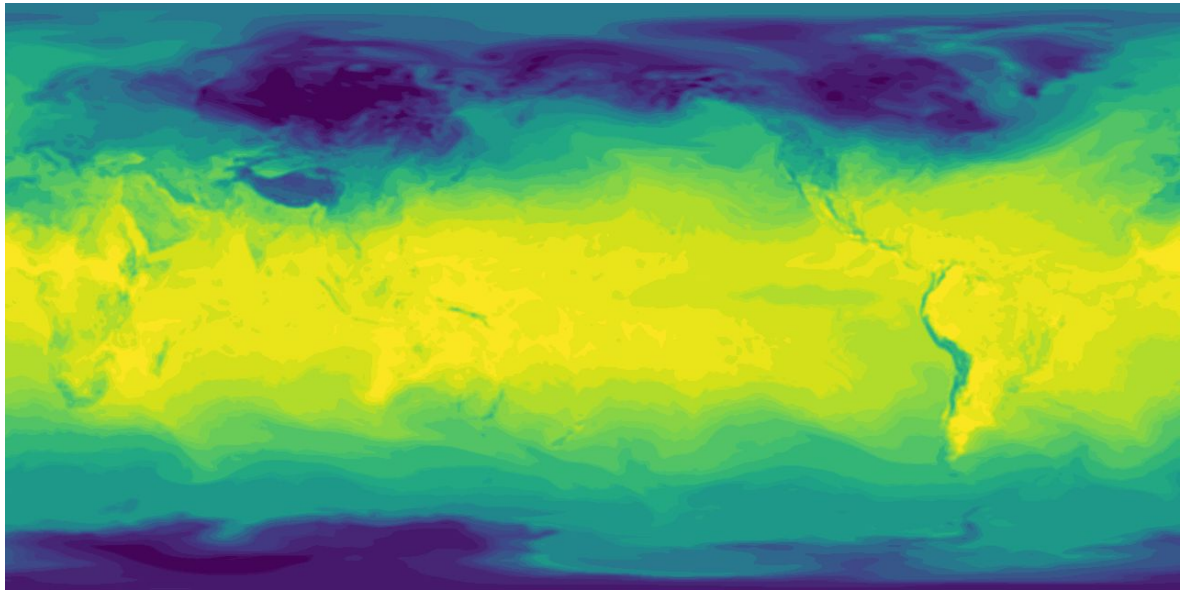
Hybrid ML GCM (T85 resolution)



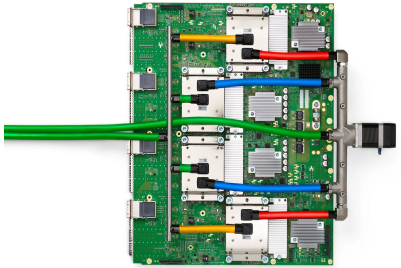
Notice how the model learned to predict the diurnal cycle.

# ML hardware is fantastic for spectral dycores

~3 minutes/day at T425 L137 (1280x640x137, ~25km)



~50% FLOP utilization (matrix-multiplication) on Google TPU v4

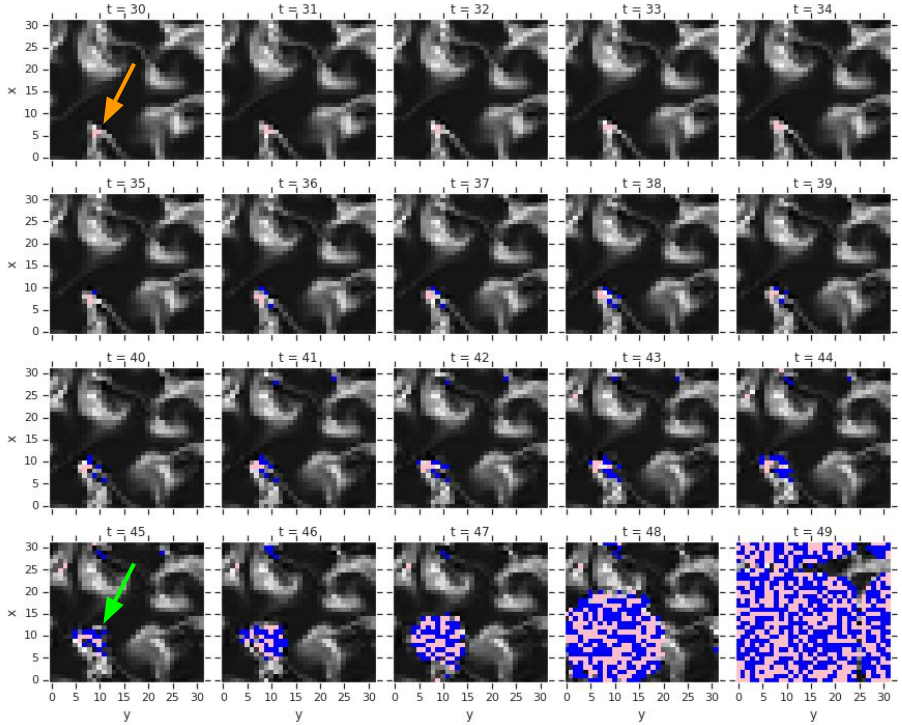


# Part 3: current challenges for ML NWP

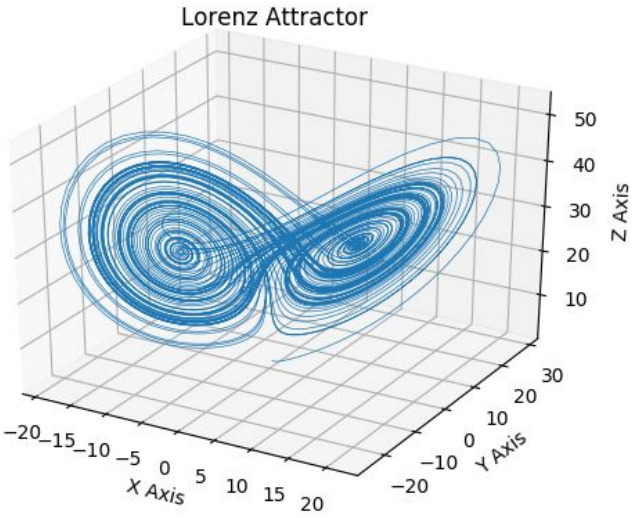
# Numerical stability with learned numerics is still hard

It's also hard for standard numerical solvers, but it's worse with ML.

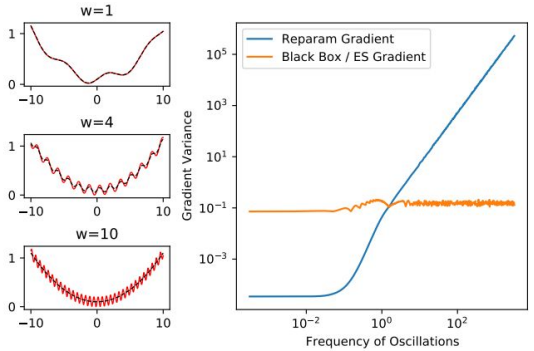
***Possible solutions**:*
Slowly unroll for more steps? Adversarial noise? Reinforcement learning? Stability analysis?



# Gradient-based optimization through chaos is ill-posed




"Gradients are not all your need"
Luke Metz, Daniel Freeman, Sam Schoenholz, Tai Kachman (2021), arXiv:2111.05803




Modeling both convective and synoptic scales -> chaos is likely unavoidable for learning global weather/climate

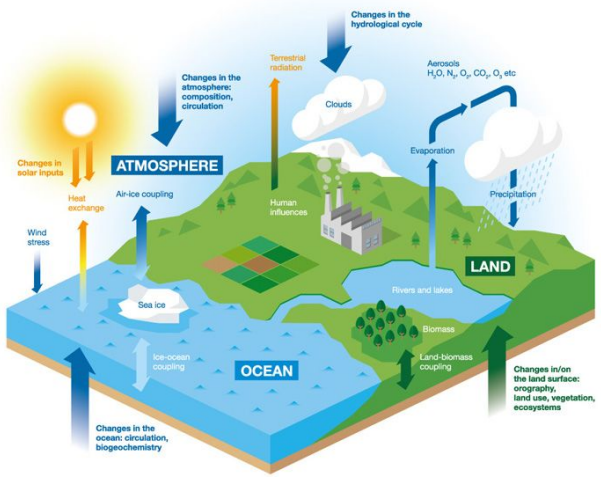
***Possible solution***: learn stochastic models, instead of using L2 loss?

# Weather and climate models are incredibly complex

ECMWF's IFS is ~1 million lines of code.

Even if ML-augmented code bases can be much smaller, that's lot of components to (re)write.

***Possible solutions***:
Replace the entire thing with black-box ML?
Figure out how to use ML in existing codes without modifications?
Large-scale efforts to make existing codes differentiable?



# Thanks for your attention!

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

To learn more: **github.com/google/jax-cfd**

Stay tuned for our new atmospheric GCM! (to be open sourced)

*My amazing collaborators at Google:*


Dmitrii Kochkov


Jamie Smith


Peter Norgaard


Griffin Mooers


Gideon Dresdner


Ayya Alieva


Qing Wang


Leonardo Zepeda-Núñez


Michael Brenner