

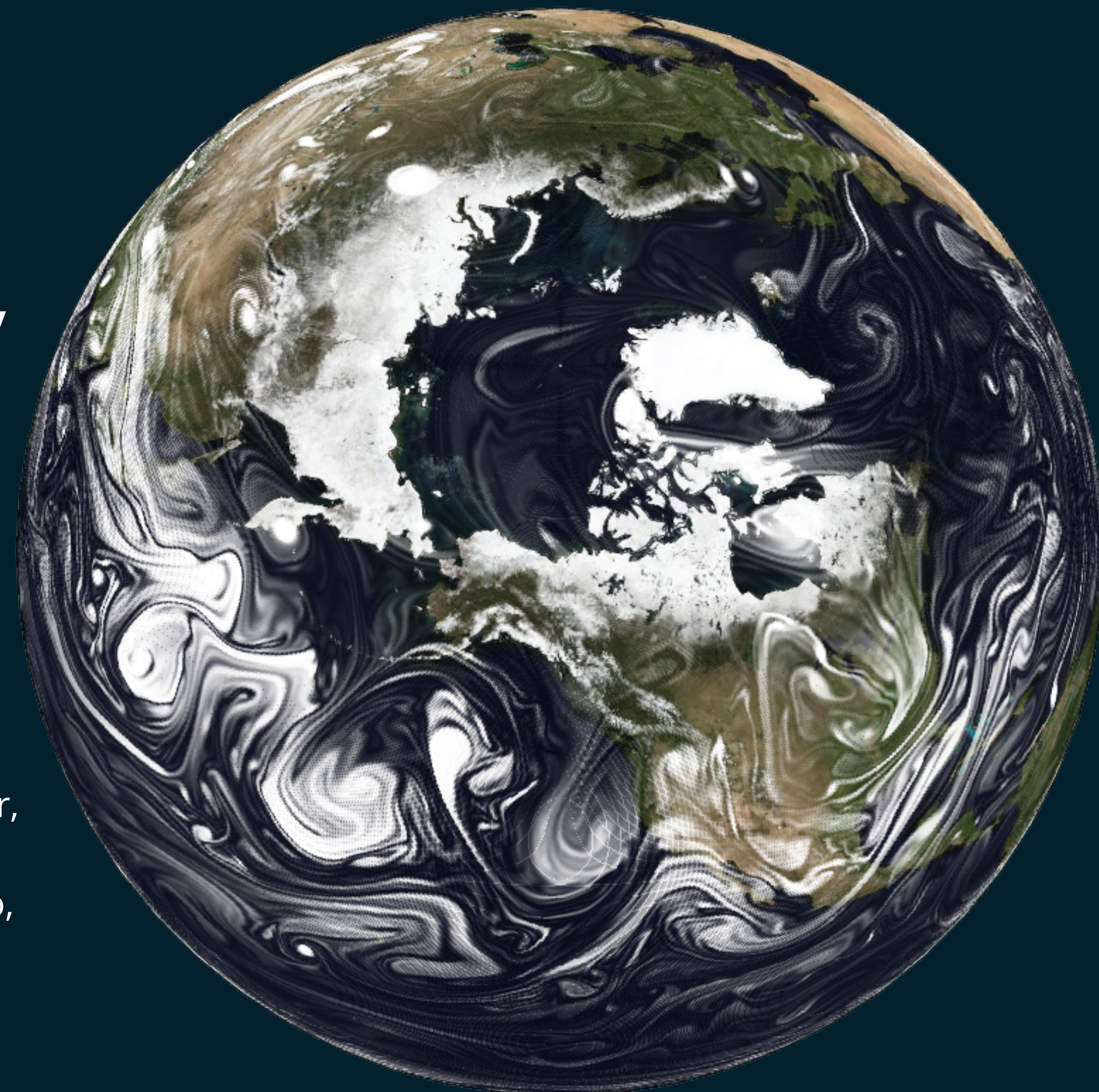
SpeedyWeather.jl:

How to build an atmospheric model
towards extensibility and interactivity
for machine learning-based climate
science

Milan Klöwer¹,

Maximilian Gelbrecht, Daisuke Hotta, Justin Willmert, Simone Silvestri, Greg Wagner,
Alistair White, Sam Hatfield, Tom Kimpson, Navid Constantinou, Yu-Kun Qian,
Nathanael Wong, Pietro Monticone, David Meyer, Valentin Churavy, Mosè Giordano,
Chris Hill

¹University of Oxford



Simulations vs Satellite

Uncertainties in *dynamics* reduce with higher resolution

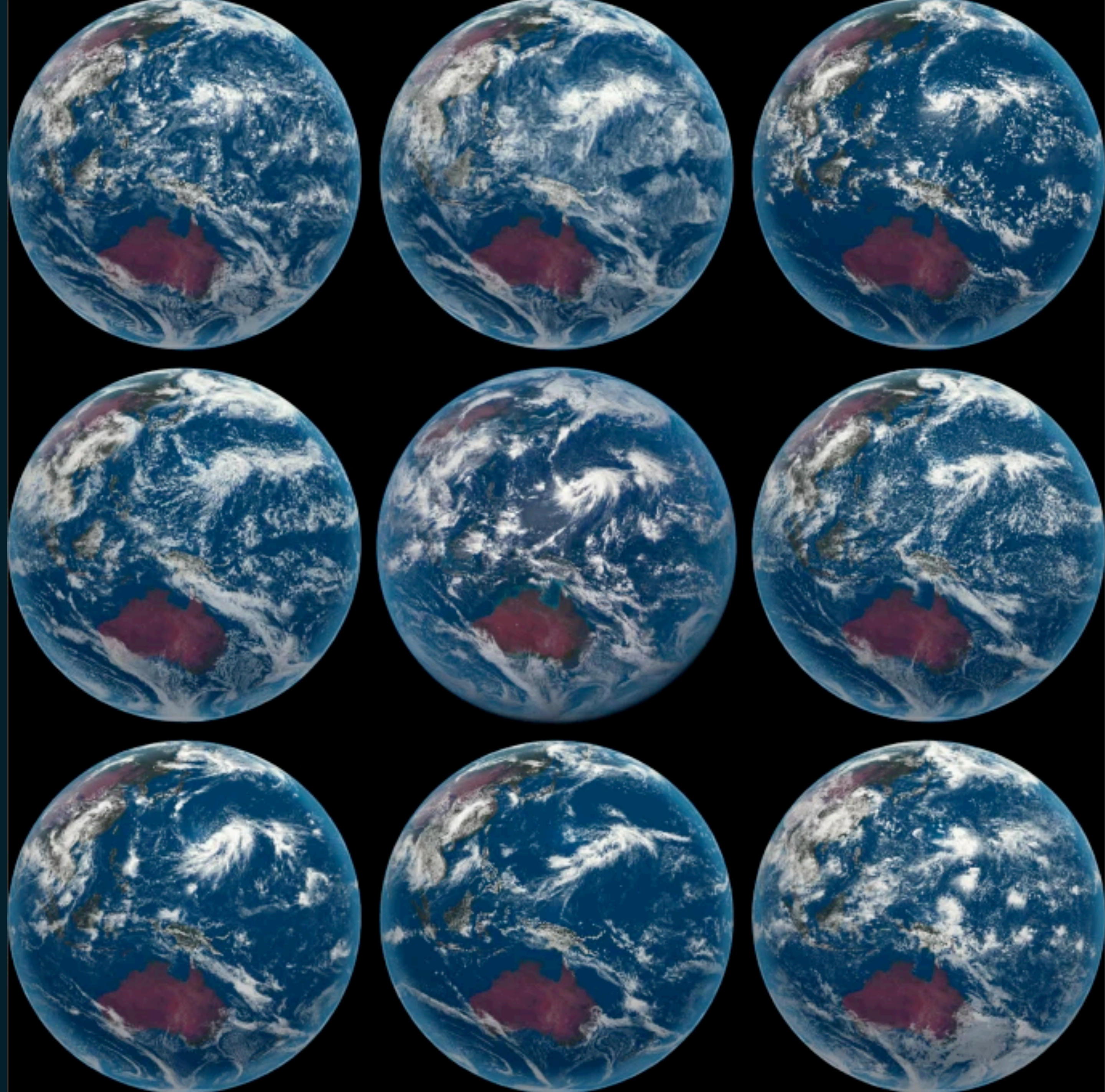
Uncertainties in *physics* often dominate weather forecasts

Dynamics \leftrightarrow (micro) physics

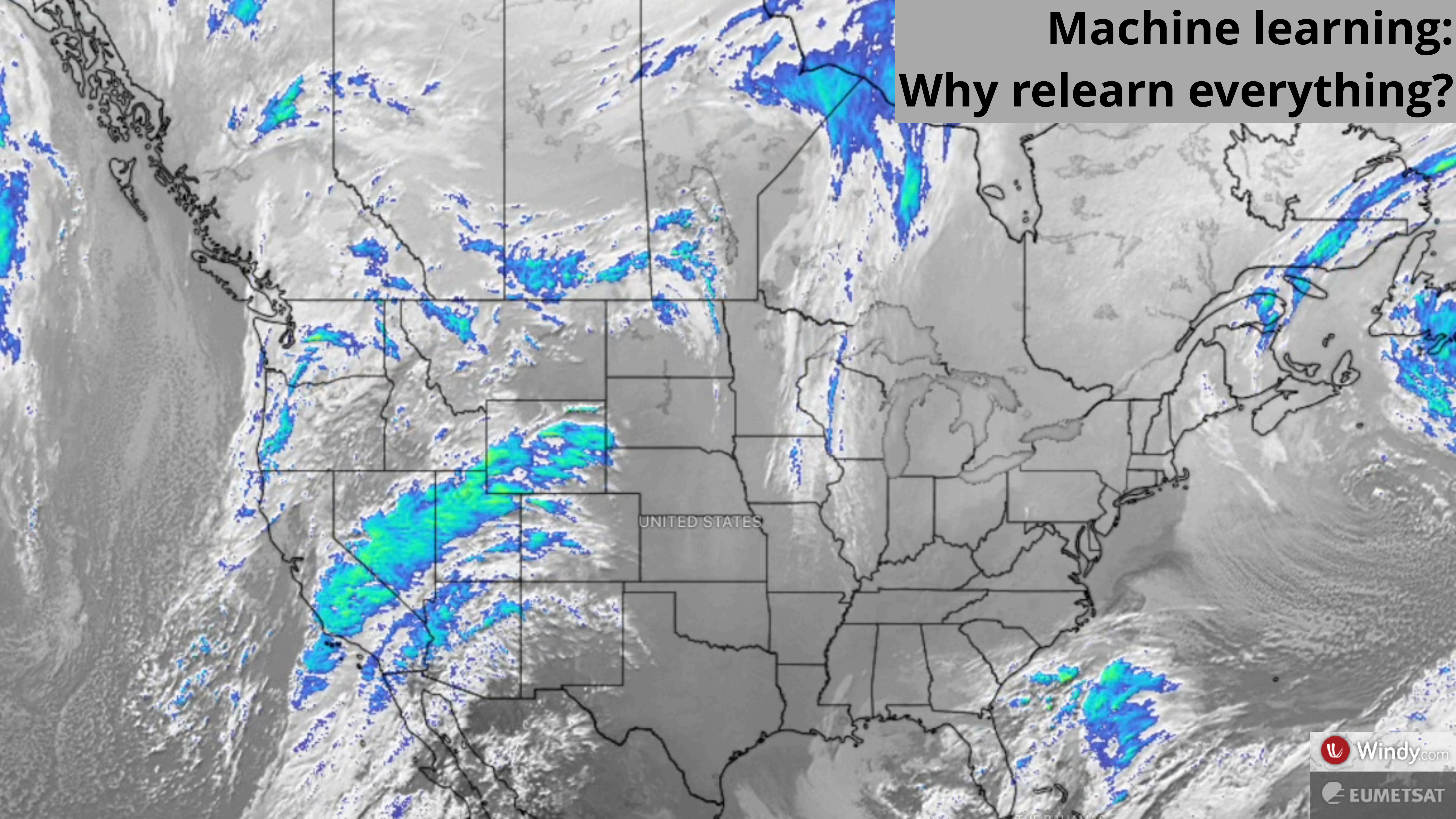
Laws \leftrightarrow heuristics

Certain \leftrightarrow uncertain

How can we constraint models better to data?



Machine learning: Why relearn everything?



GraphCast

Pure machine learning weather forecast model

GraphNN with encoder - processor - decoder architecture

6h time steps

No physical laws, not generalizable (?)
nor explainable (?)

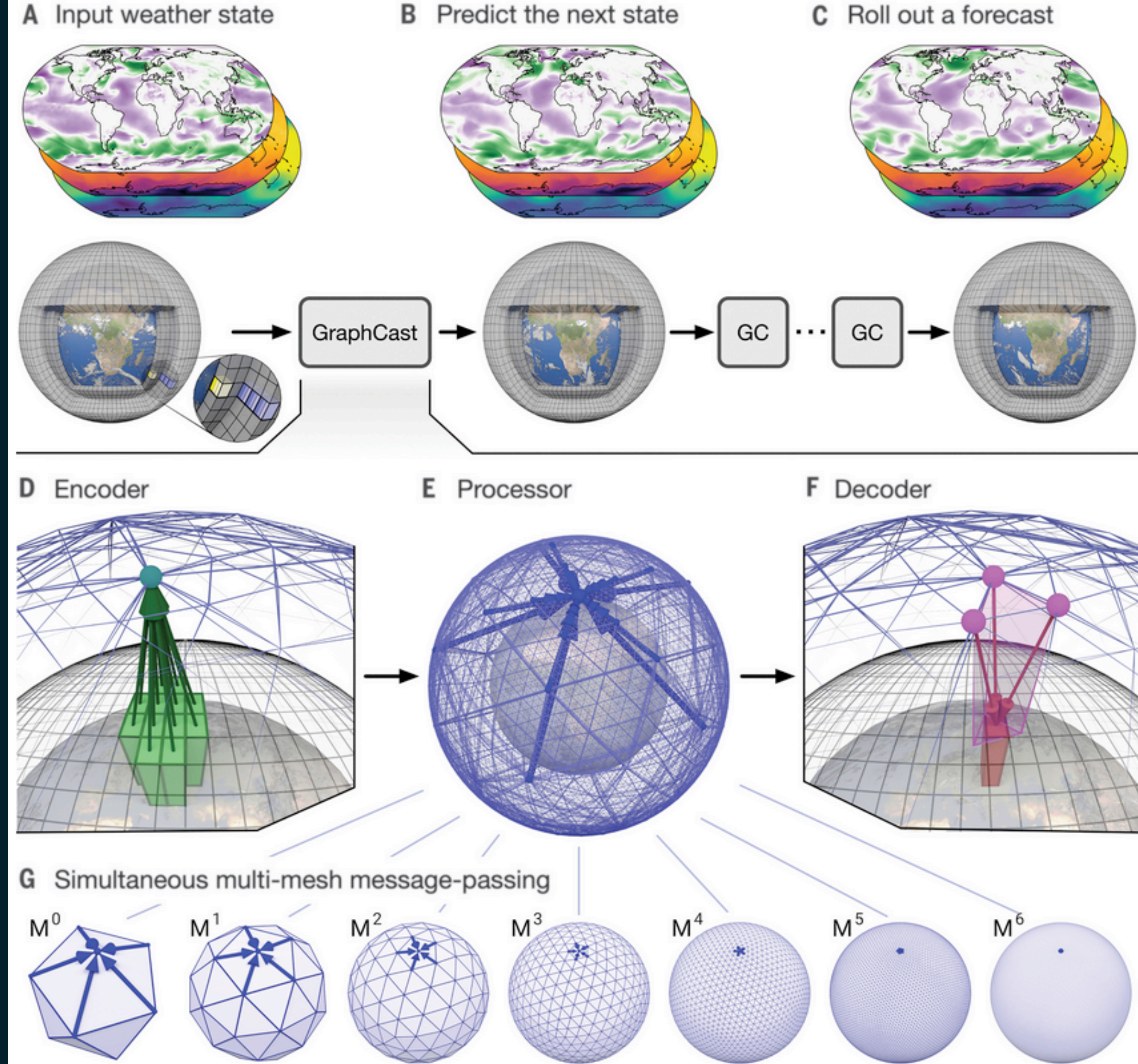
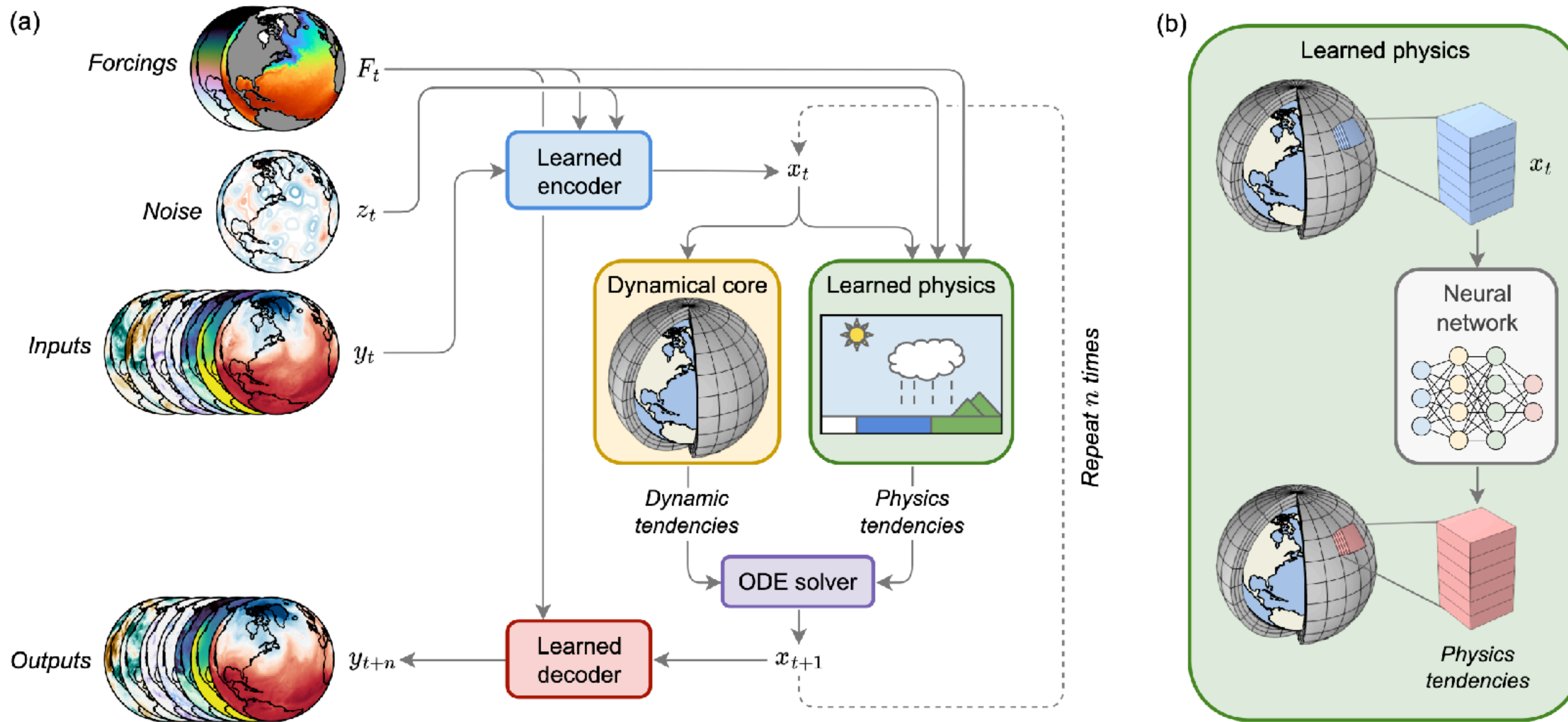


Fig. 1. Model schematic.

(A) The input weather state(s) are defined on a 0.25° latitude-longitude grid comprising a total of $721 \times 1440 = 1,038,240$ points. Yellow layers in the close-up pop-out window represent the five surface variables, and blue layers represent the six atmospheric variables that are repeated at 37 pressure levels ($5 + 6 \times 37 = 227$ variables per point in total), resulting in a state representation of 235,680,480 values. (B) GraphCast predicts the next state of the weather on the grid. (C) A forecast is made by iteratively applying GraphCast (GC) to each previous predicted state, to produce a sequence of states that represent the weather at successive lead times. (D) The encoder component of the GraphCast architecture maps local regions of the input (green boxes) into nodes of the multimesh graph representation (green, upward arrows that terminate in the green-blue node). (E) The processor component updates each multimesh node using learned message-passing (heavy blue arrows that terminate at a node). (F) The decoder component maps the processed multimesh features (purple nodes) back onto the grid representation (red, downward arrows that terminate at a red box). (G) The multimesh is derived from icosahedral meshes of increasing resolution, from the base mesh (M^0 , 12 nodes) to the finest resolution (M^6 , 40,962 nodes), which has uniform resolution across the globe. It contains the set of nodes

Neural General Circulation Models

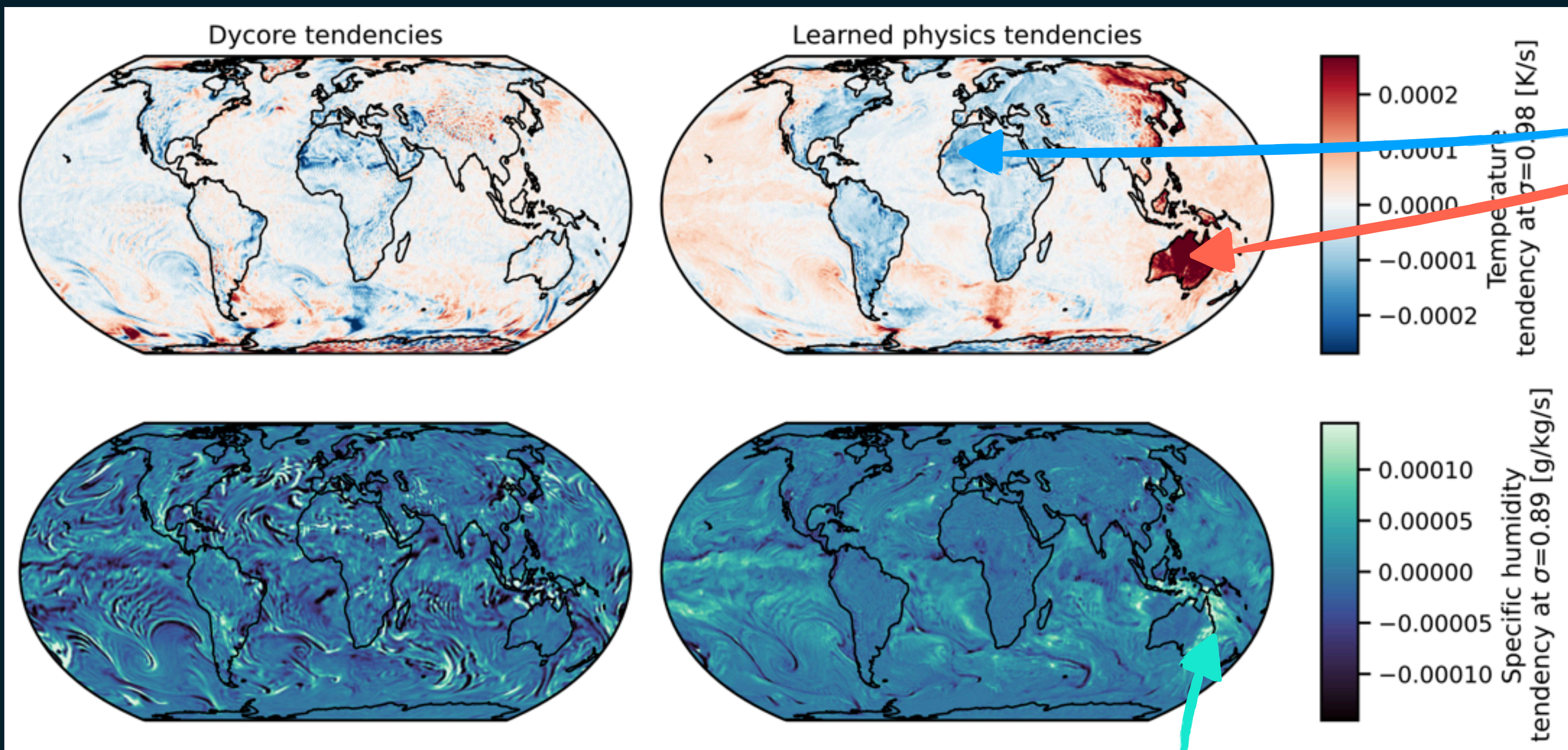


Kochkov et al., 2023

- Differentiable model (dynamics + physics) based on Python+JAX
- More explainable than pure ML

Physics discovery and generalisation

Temperature
Humidity



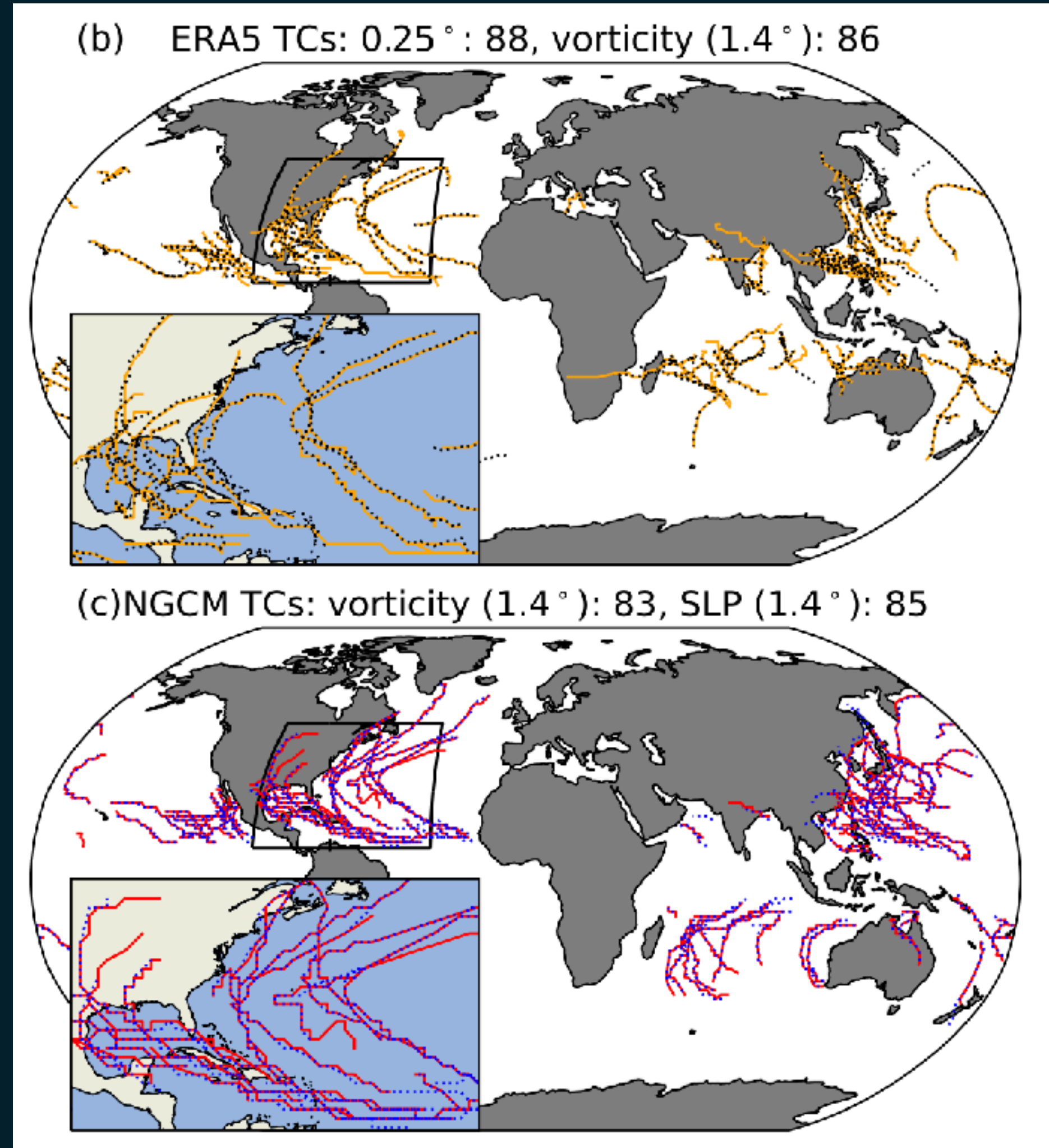
Radiative surface cooling (night) and heating (day)

Tropical cyclone statistics well simulated

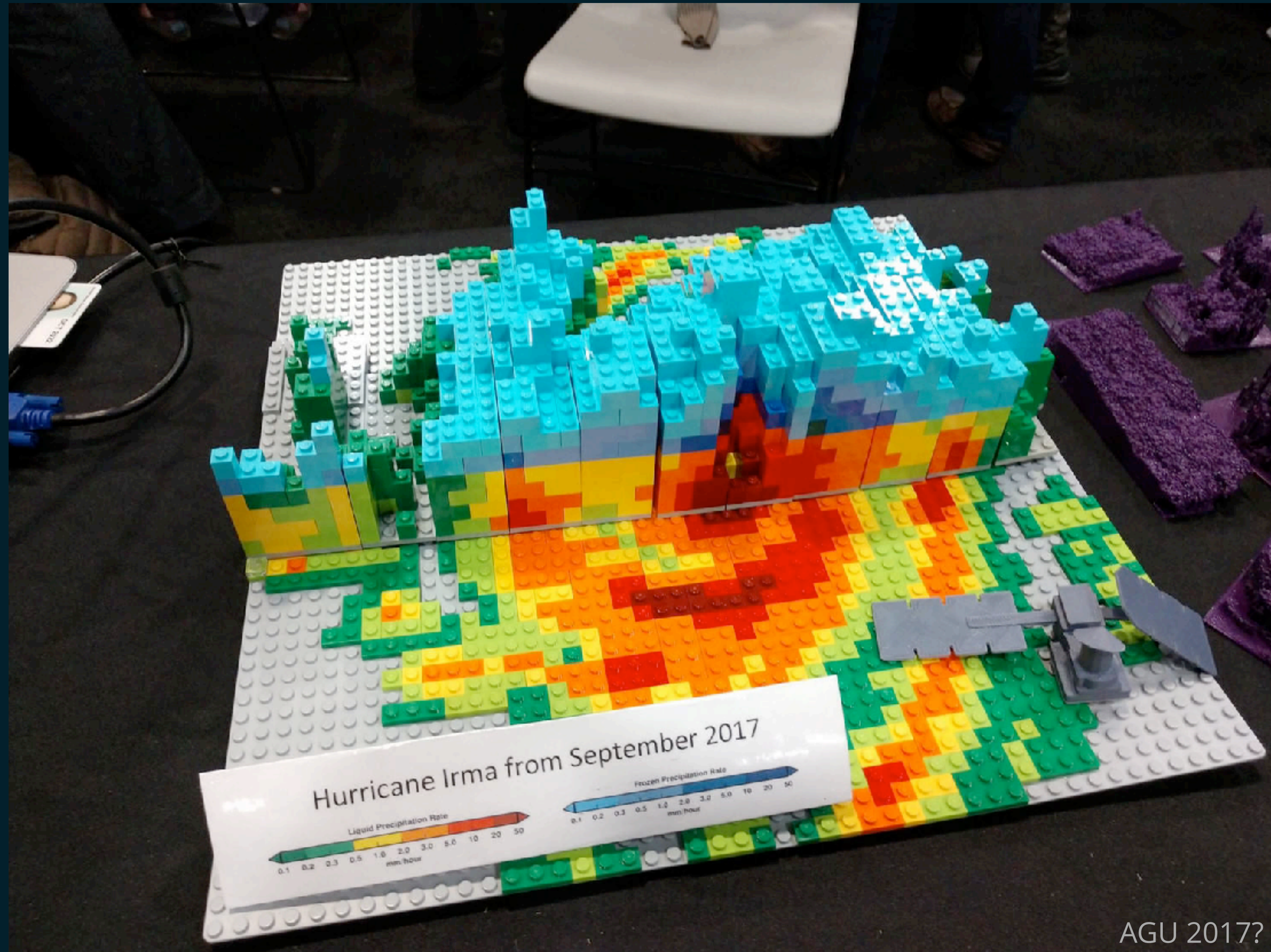
Kochkov et al., 2024

Daytime evaporation + convection over tropical oceans

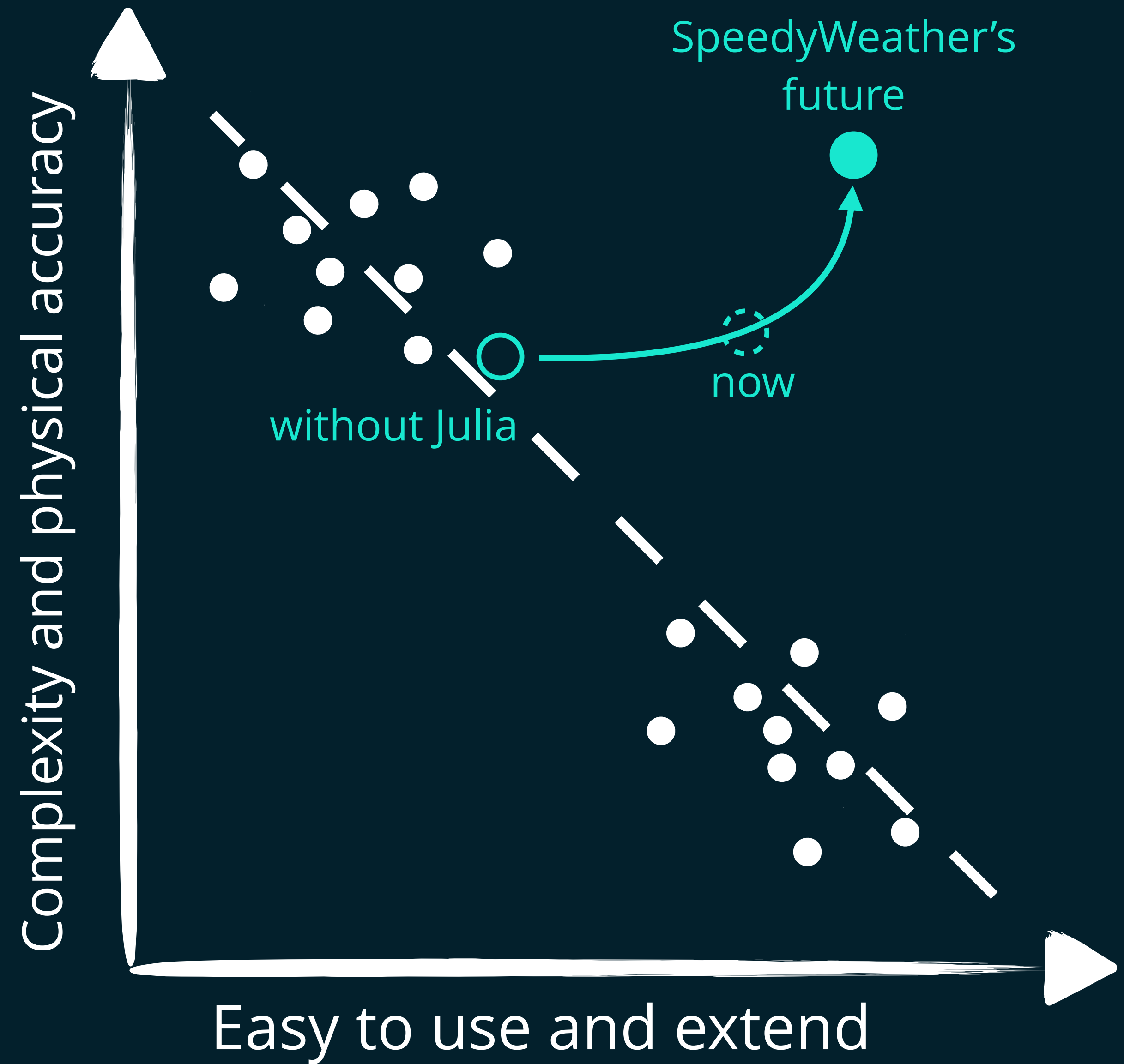
Trained on weather but skilful for climate



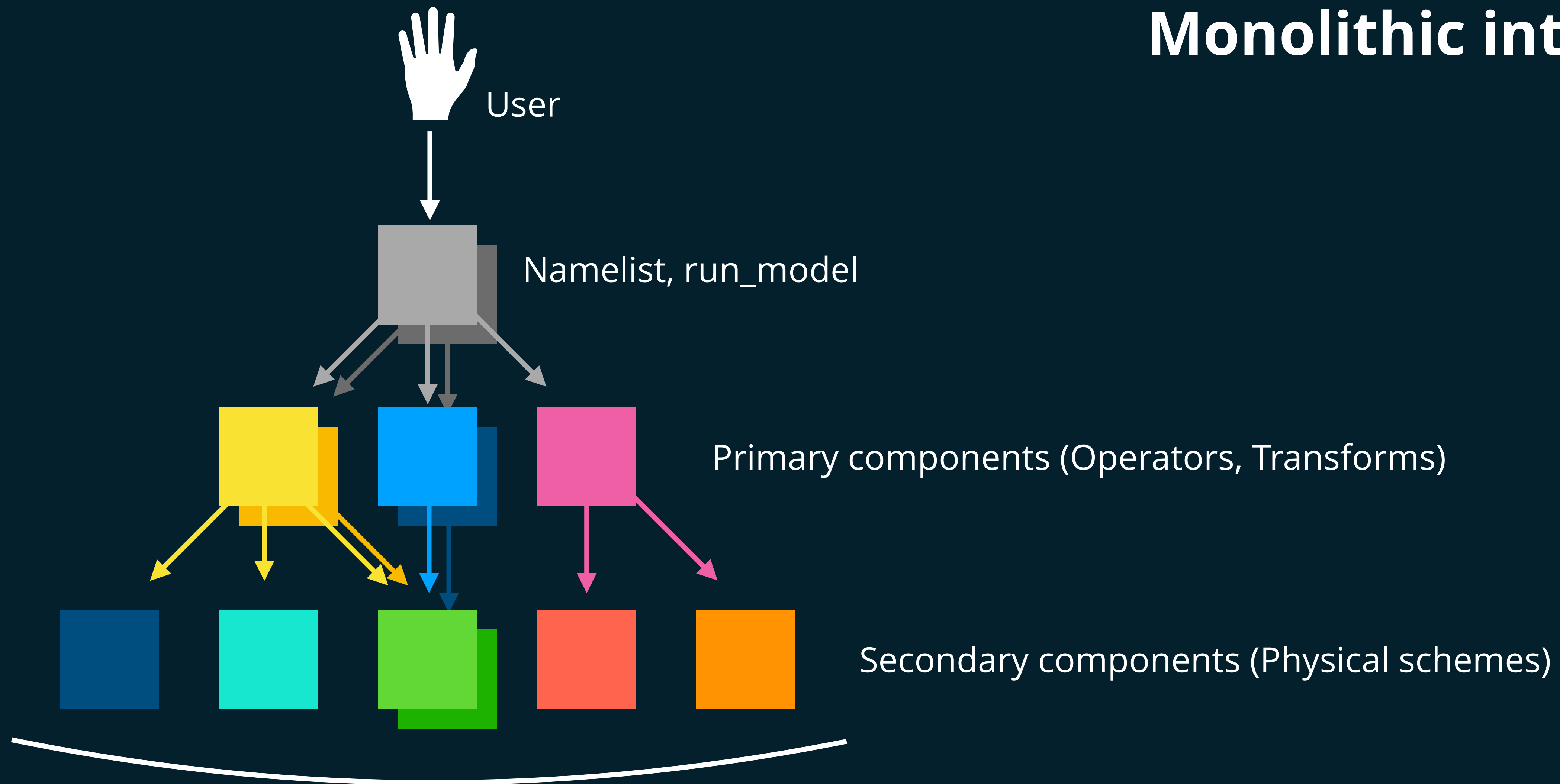
What do we want climate models to be?



Two language-problem of weather and climate models

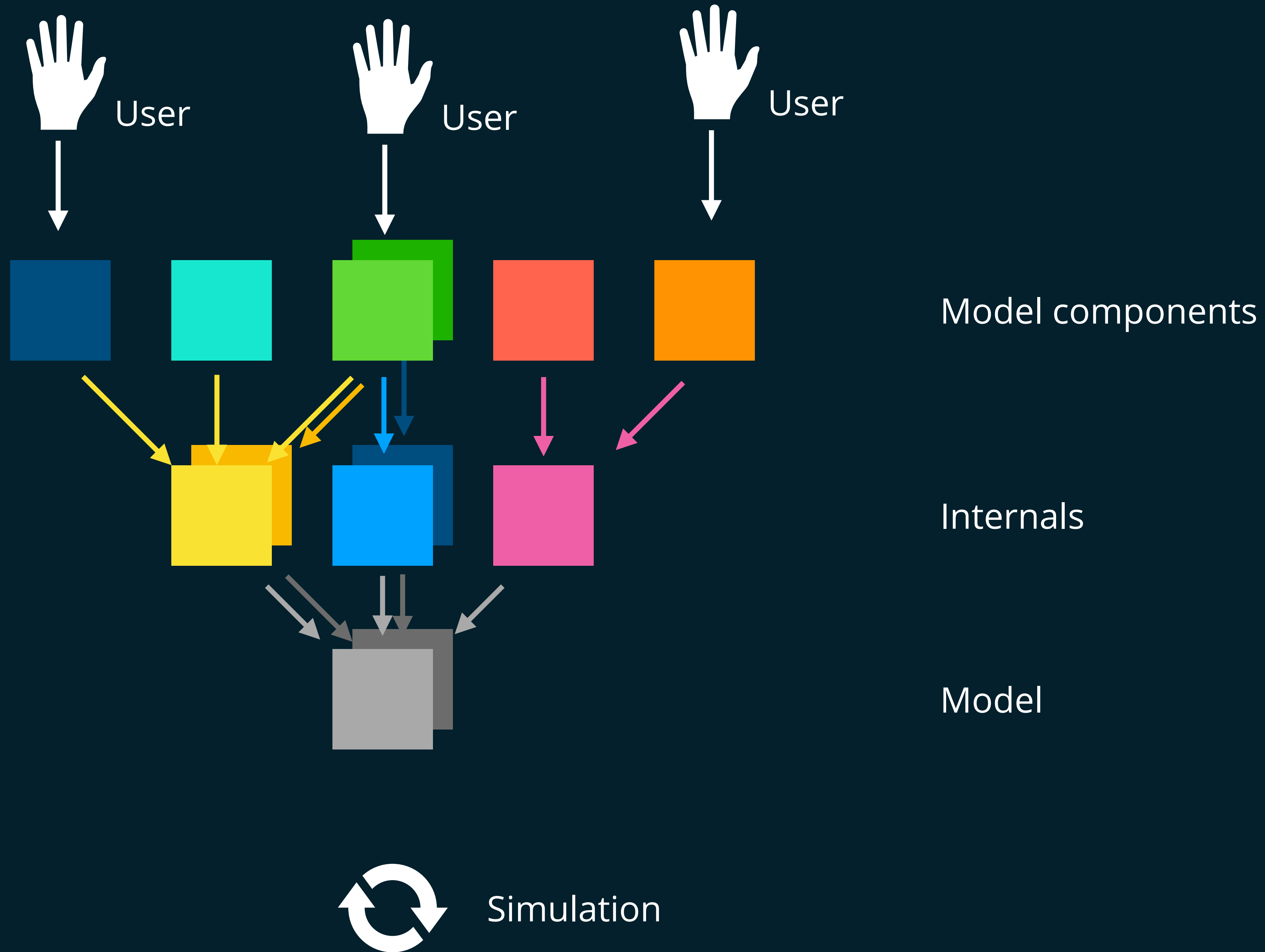


Monolithic interface?



***Modularity, extendability,
composability very hard!***

Model constructor interface



SpeedyWeather.jl v0.10: Where are we?

The screenshot shows the SpeedyWeather.jl documentation website. It features a navigation menu on the left with categories like 'Extending SpeedyWeather', 'Dynamics', 'Physics', 'Large-scale condensation', 'Convection', 'Radiation', 'Vertical diffusion', 'Surface fluxes', 'Discretization', 'RingGrids', 'LowerTriangular', and 'SpeedyTransformations'. The main content area is divided into several sections:

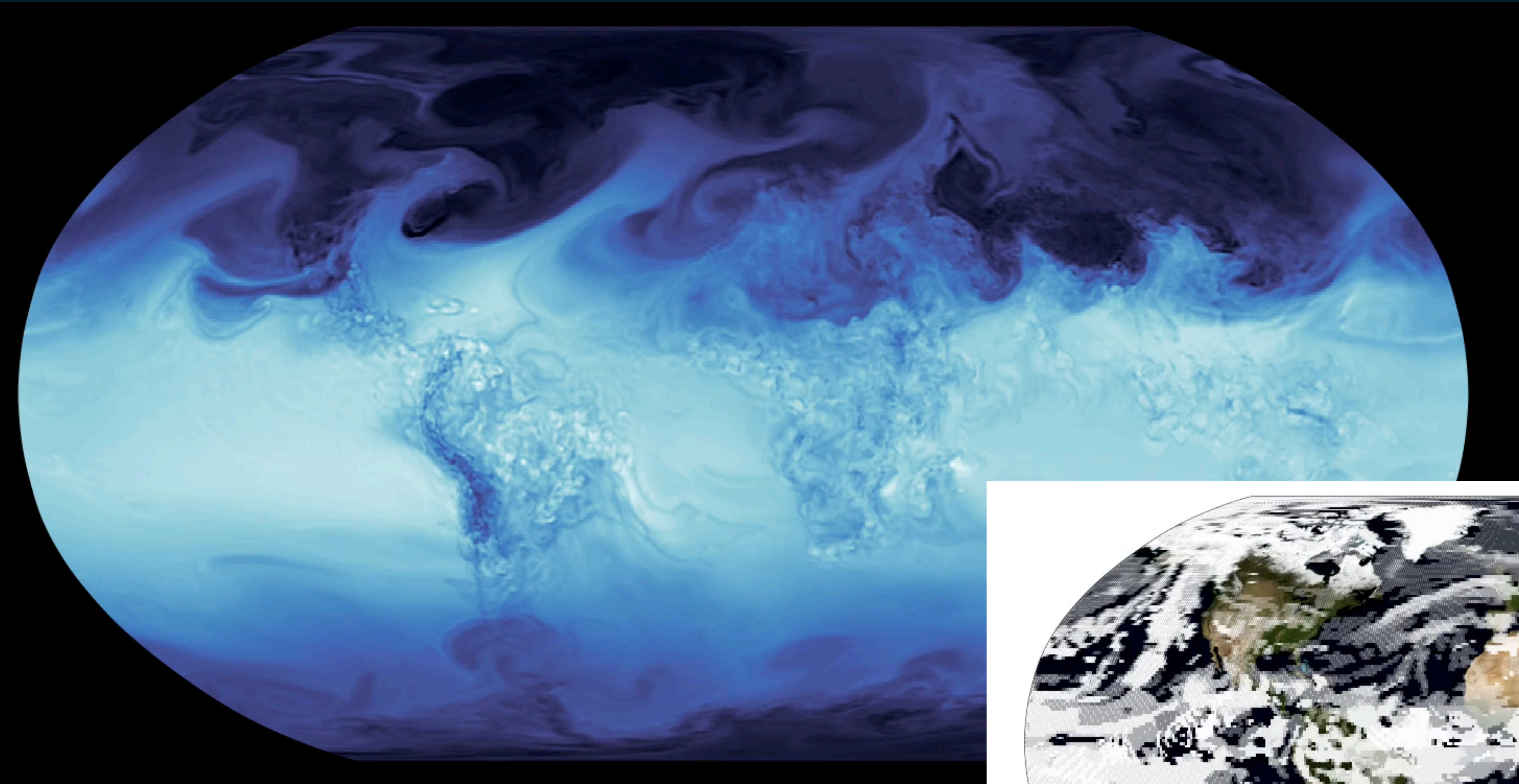
- Implicit large-scale condensation:** Explains the process of condensing towards the new saturation humidity $q^*(T_{i+1})$ at $i+1$. It includes a Taylor expansion and the resulting equation:
$$q_{i+1} - q_i = q^*(T_{i+1}) - q_i = q^*(T_i) + (T_{i+1} - T_i) \frac{\partial q^*}{\partial T}(T_i) + O(\Delta T^2) - q_i$$
 and a linear approximation:
$$\Delta q = q^*(T_i) - \frac{L_v}{c_p} \Delta q \frac{\partial q^*}{\partial T}(T_i) - q_i$$
- Held-Suarez forcing:** Shows a code snippet for setting up a spectral grid and a model with Held-Suarez forcing and drag.
- Surface relative vorticity:** A 3D heatmap plot showing vorticity distribution over a globe, with a color scale from -2.0×10^{-5} to 2.0×10^{-5} .

The image shows the cover of a paper in The Journal of Open Source Software (JOSS). The title is "SpeedyWeather.jl: Reinventing atmospheric general circulation models towards interactivity and extensibility". The authors listed are Milan Klöwer, Maximilian Gelbrecht, Daisuke Hotta, Justin Willmert, Simone Silvestri, Gregory L. Wagner, Alistair White, Sam Hatfield, Tom Kimpson, and Navid C. Constantinou. The paper is associated with DOI: 10.21105/joss.06323. The summary states: "SpeedyWeather.jl is a library to simulate and analyze the global atmospheric circulation on the sphere. It implements several 2D and 3D models which solve different sets of equations: the primitive equations with and without humidity, the shallow water equations, and the barotropic vorticity equation." The paper is edited by Kristen Thyng and reviewed by @vavrines.

JOSS paper is out!

Extensive documentation, with examples + "textbook" on global atmospheric modelling

Higher resolution examples



Humidity at 40km resolution

Cloud cover

