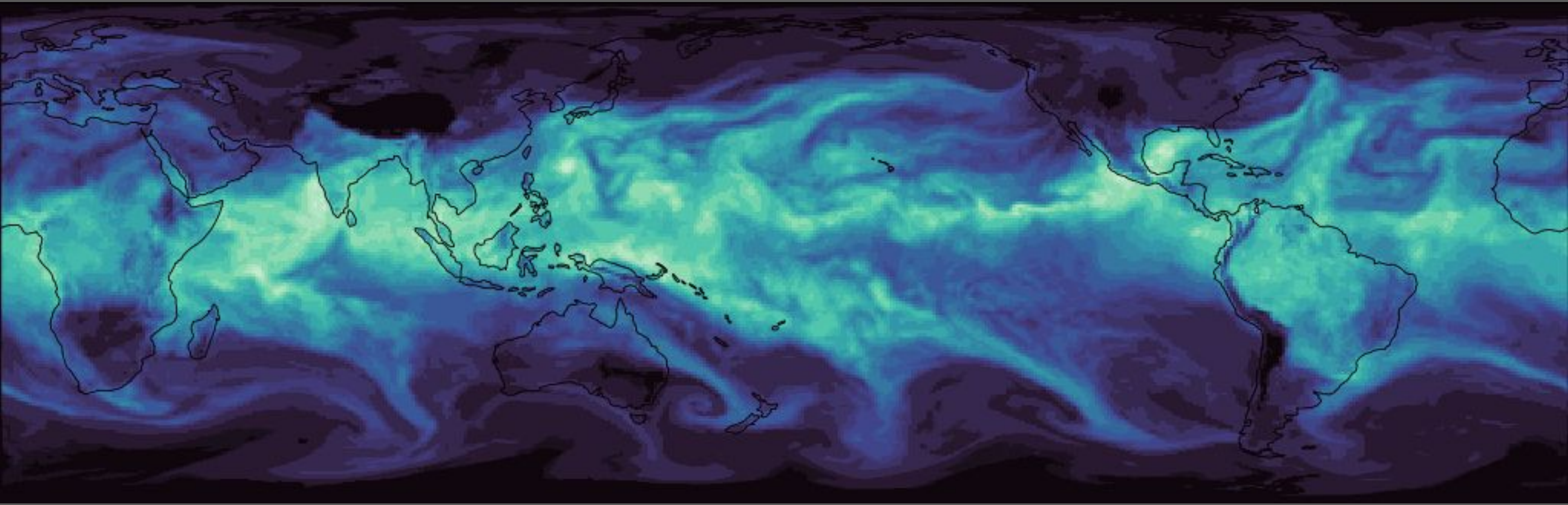
From Scoring Rules to Probabilistic ML Forecasting



Ian Langmore
Google Research

UQ Summer School
USC
August 7, 2024

Acknowledgments: the extended NeuralGCM team

Google Research

Dmitrii Kochkov

Janni Yuval

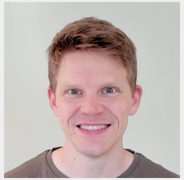
Ian Langmore

Peter Norgaard

Jamie Smith

Griffin Mooers

James Lottes



Stephan Rasp

Michael Brenner

Stephan Hoyer

ECMWF

Sam Hatfield

Peter Düben

MIT

Milan Klöwer

Google DeepMind

Peter Battaglia

Alvaro Sanchez-Gonzalez

Matthew Willson

Outline

Part 1: Evaluating probabilistic forecasts with scoring rules

Part 2: ML weather forecasting and NeuralGCM

Part 3: JAX and JAX-CFD live tutorial

Part 1

Evaluating probabilistic forecasts with scoring rules

Background and Motivation

How to train/tune/evaluate a probabilistic model?

Training a probabilistic model

The situation

- Many ground truth examples
  - I'm not going into Bayesian modeling here
- Must learn *many* parameters in a complex model
  - Need a scalable method

My personal work

- Weather forecasts

Loss functions will be used with SGD

Stochastic gradient descent

- Ground truth examples $\{Y_1, \ldots, Y_N\}$
- Model parameterized by θ
- Possible conditioning covariates $\{\alpha_1, \ldots, \alpha_N\}$
- Forecasts $\{X_1, \ldots, X_N\}$
- Loss $\mathcal{L}(X, Y; \theta)$

1. Draw random $n \in \{1, \ldots, N\}$
2. Model produces $X_n \sim P(X \mid \alpha_n)$
3. Update parameters $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(X_n, Y_n \mid \theta)$
4. Repeat until convergence

Evaluating a probabilistic model

The setup

- Given many forecasts $\{X_n\}$ from model P and ground truth $\{Y_n\}$

We want to

- Give a score to P that is minimized when

  X and Y have the same distribution

- Give interpretable evaluations

Some loss functions double as eval functions – some don't

Maximum likelihood and its shortcomings

Maximum likelihood for logistic and linear regression

Example 1: Logistic regression

- *ground truth* $Y \in \{0, 1\}$
- *covariates* α
- *forecast* X

$$\mathrm{P}[X = 1] = \frac{1}{1 + e^{-\theta \cdot \alpha}}.$$

$$\begin{aligned} \mathcal{L}(X, Y \mid \theta) &= -\log \mathrm{P}[X = Y] \\ &= Y\, \mathrm{P}[X = 1] + (1 - Y)\mathrm{P}[X = 0] \\ &= Y \,\log\left(1 + e^{-\theta \cdot \alpha}\right) + (1 - Y) \log\left(1 + e^{\theta \cdot \alpha}\right), \end{aligned}$$

$$\theta_{ML} := \arg\min \mathcal{L}(X, Y; \theta).$$

Pros

- Maximum likelihood is an efficient way to estimate θ
- Works well with stochastic gradient descent

Cons

- Doesn’t work for continuous variables
- Loss values are often difficult to interpret

Cross Entropy can give uninterpretable values

Suppose we have models P and Q

- P[Thunderstorm] = 0.001
- Q[Thunderstorm] = 0

From a consumer's perspective they are pretty similar

If there is a thunderstorm (Y=1)

- -Log(P[Y]) = 3
- -Log(Q[Y]) = infinity

Example 2: Linear regression

- *ground truth* $Y \in \{0, 1\}$
- *covariates* α
- *forecast* X

$$X = \theta \cdot \alpha + \mathcal{N}(0, \sigma^2)$$

The model implies

$$p(X \mid \theta) = \mathcal{N}(\alpha \cdot \theta, \sigma^2).$$

$$\begin{aligned}\mathcal{L}(X, Y \mid \theta) &= -\log p(X \mid \theta) \\ &= \frac{(Y - \alpha \cdot \theta)^2}{2\sigma^2} + \frac{1}{2}\log\sigma + \frac{1}{2}\log 2\pi.\end{aligned}$$

Pros

- Maximum likelihood is an efficient way to estimate θ
- Works well with stochastic gradient descent
- Squared error is easy to interpret

Cons

- Linear model often insufficient
- Additive noise often insufficient

Example 3: Deep Neural Network and Mean Square Error

Deep neural network and MSE

- *ground truth* $Y \in \mathbb{R}^d$
- *covariates* α
- *forecast* $X \in \mathbb{R}^d$

$$X = F(\alpha; \theta)$$

$$\begin{aligned} \mathcal{L}(X, Y \mid \theta) &= \|Y - X\|^2 \\ &= \|Y - F(\alpha \mid \theta)\|^2 \end{aligned}$$

The loss is maximum likelihood under the probabilistic model

$$p(X \mid \theta) = F(\alpha \mid \theta) + \mathcal{N}(0, \sigma^2 I).$$

Pros

- Maximum likelihood is an efficient way to estimate θ
- Works well with stochastic gradient descent
- Squared error is easy to interpret
- Often results in X ≈ Y

Cons

- The generative model gives silly samples
- If Y has inherent uncertainty, results in blurry forecasts

Mean squared error

...why it favors blurry forecasts

Given stochastic ground truth Y and forecast X,

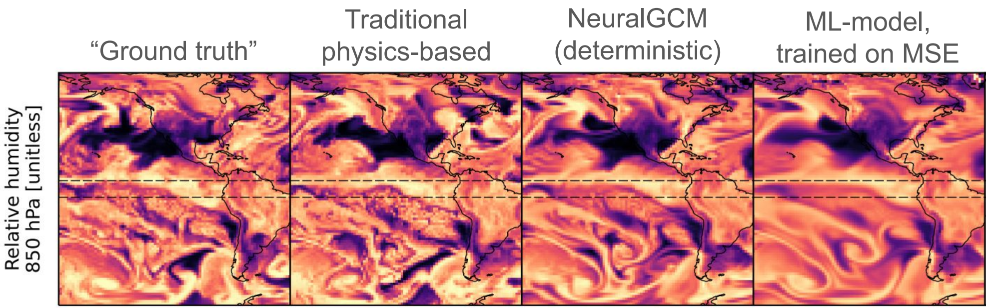
$$\begin{aligned}\mathrm{MSE} :&= \mathbb{E}\|X - Y\|^2 \\ &= \|\mathbb{E}X - \mathbb{E}Y\|^2 + \mathrm{Var}\{X\} + \mathrm{Var}\{Y\}.\end{aligned}$$

- MSE is minimized by forecasting $X \equiv \mathbb{E}Y$
- Regardless of your forecast, *any variance in X only hurts!*

Alternative explanation:

If a storm may be here or there…

you minimize MSE by forecasting a blurry cloud everywhere



Common misleading practice:

- train to minimize MSE
- show better RMSE than a physics-based model
- claim to be SOTA

Approximate maximum likelihood models

Probabilistic generative model with parameters θ. Samples are generated via

1. Sample $Z \sim p(z \mid \theta)$ (the *prior*)
2. Sample $X \sim p(x \mid Z, \theta)$ (the *likelihood*)

Ways to approximate a likelihood

- Variational autoencoder variants [C]
- Normalizing flow based models [P]
- Diffusion models [S, GC]

Estimate the marginal likelihood

$$p(x \mid \theta) = \int p(z \mid \theta) p(x \mid z, \theta) \, \mathrm{d}z,$$

then estimates θ by maximum likelihood over data $Y = (Y_1, \ldots, Y_K)$

$$\theta^* = \arg\max \log p(Y \mid \theta) \approx \arg\max \sum_{k=1}^{K} \log p(Y_k \mid \theta).$$

Difficulty:
How to approximate this integral in a realistic model?

Asymptotic efficiency of the MLE

Suppose...

- we have ground truth examples $\{Y_1, \ldots, Y_N\}$
- we have model depending on parameter θ with probability density $p(x \mid \theta)$
- a parameter estimator $\hat{\theta}$

Then

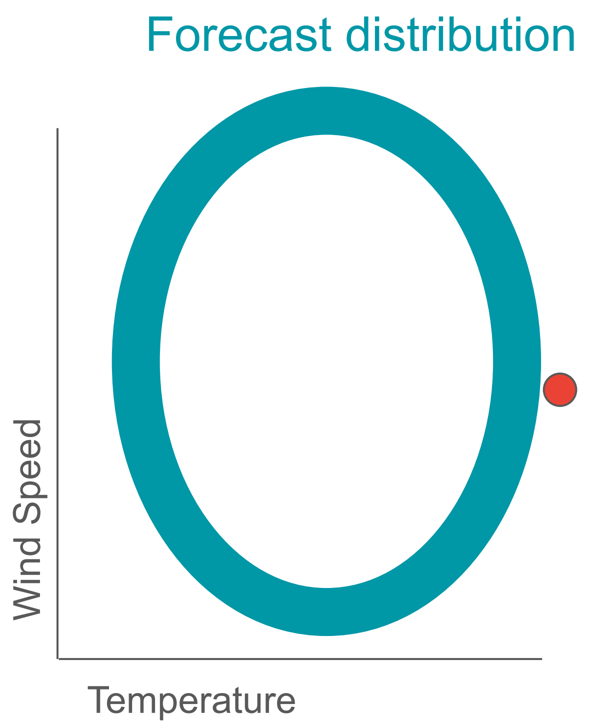
$$\mathrm{Var}\left\{\hat{\theta}\right\} \geq \frac{1}{I(\theta)},$$

where

$$\begin{aligned} I(\theta) :&= N\mathbb{E}\left\{\left(\frac{\partial}{\partial\theta}\log p(Y;\theta)\right)^2\right\} \\ &= -N\mathbb{E}\left\{\frac{\partial^2}{\partial\theta^2}\log p(Y;\theta)\right\} \end{aligned}$$

The maximum likelihood estimator (asymptotically, as N → infinity) achieves equality in this bound [notes].

Maximum likelihood doesn't "respect the metric"



Observation Y

⇒ $-\mathrm{Log}\ p(Y \mid \theta) \approx \infty$

Would prefer a small penalty

Proper Scoring Rules

Formal definition

Given model P, a *scoring rule* is a function $S(P, \cdot)$ such that

- if event $y \sim Q$ is seen, the reward is $S(P, y)$
- The expected reward is

$$S(P, Q) = \mathbb{E}_Q \{S(P, Y)\} = \int S(P, y) q(y)\, \mathrm{d}y \approx \frac{1}{N} \sum_{n=1}^{N} S(P, y_n).$$

S is *proper* if the true distribution is a minimizer

$$S(Q, Q) \leq S(P, Q), \text{ for all } P.$$

S is *strictly proper* if the true distribution is the unique minimizer

$$S(Q, Q) < S(P, Q), \text{ for all } P \neq Q.$$

Logarithmic Score (Maximum likelihood)

Suppose model P has probability density p.
The logarithmic score is

$$S(P, y) = -\log p(y \,|\, \theta).$$

This gives *maximum likelihood estimation*

$$\theta_{ML} = \arg\min_{\theta} -\frac{1}{N}\sum_{n=1}^{N} \log p(y_n \,|\, \theta)$$
$$\approx \arg\min_{\theta} \mathbb{E}_Y \left\{-\log p(Y \,|\, \theta)\right\}.$$

Pros

- Asymptotically efficient
  - (asymptotically) no parameter estimator can have lower variance [notes]
- Every *local* strictly proper scoring rule is equivalent to logarithmic score.
  - local = depends on P only at observed points

Cons

- Requires the density p(x | θ),

  cannot work if you only have samples

  X □ p

Non-local scores from losses

We will build non-local scores from loss functions $\mathcal{L}(X, Y)$.

Rather than...

- Writing $S(P, y) := \mathbb{E}_X \{\mathcal{L}(X, y)\}$
- Then minimizing $S(P, Q) := \mathbb{E}_Y \{S(P, Y)\}$ via SGD

we simply analyze the "score"

$$\mathbb{E}\{\mathcal{L}(X, Y)\}.$$

Continuously Ranked Probability Score (CRPS)

For scalar predictions X, X' and ground truth Y,

$$CRPS = \mathbb{E}|X - Y| - \frac{1}{2}\mathbb{E}|X - X'|$$
$$= \int_{-\infty}^{\infty} (\mathrm{P}[X \leq y] - \mathrm{P}[Y \leq y])^2 \, \mathbf{d}y + \mathbf{const.}$$

- Strictly proper
  ⇒ X ~ Y is the unique minimizer
- Generalizes MAE
- Does not require density p(x)!

In $\mathbb{R}^N$,

$$CRPS = \sum_{n=1}^{N} \left[\mathbb{E}|X_n - Y_n| - \frac{1}{2}\mathbb{E}|X_n - X_n'|\right]$$
$$= \mathbb{E}\|X - Y\|_1 - \frac{1}{2}\mathbb{E}\|X - X'\|_1.$$

- ~~Strictly~~ proper
  ⇒ *any* X with correct marginals is *a* minimizer
- Generalizes MAE
- We let $\{X_n\}$ be components in spatial & spectral basis [GR]

Energy Score (ES)

In $\mathbb{R}^N$, the energy score is

$$ES = \mathbb{E}\|X - Y\|_2 - \frac{1}{2}\mathbb{E}\|X - X'\|_2.$$

Given

$$F(\omega) := \mathbb{E}\left\{e^{i\omega\cdot X}\right\}, \qquad G(\omega) := \mathbb{E}\left\{e^{i\omega\cdot Y}\right\},$$

$$ES \propto \int_{\mathbb{R}^N} \frac{|F(\omega) - G(\omega)|^2}{\|\omega\|^{N+1}} \mathrm{d}\omega + \mathrm{const}.$$

- Strictly proper
  - …but the signal in correlations is *tiny*
- Generalizes RMSE
- Rotationally invariant
  - can use spectral basis
- Still strictly proper if we rescale the norm
- *Barely* penalizes incorrect correlations

[SZ]

Kernel scores

Given negative definite kernel $K(x, y)$, define a score

$$\mathbb{E}\left\{K(X,Y) - \frac{1}{2}K(X,X')\right\} \approx \frac{1}{2N}\sum_{n=1}^{N}\{K(X_n,Y_n) + K(X'_n,Y_n) - K(X_n,X'_n)\}.$$

E.g.

$$K(x,y) = \exp\left(-\|x-y\|^2/(2\sigma^2)\right),$$

or

$$K(x,y) = \frac{1}{1+\|x-y\|^2/\sigma^2}.$$

Maximum Mean Discrepancy (MMDs)

Given function class $\mathcal{F}$, define

$$MMD[\mathcal{F}] := \sup_{f \in \mathcal{F}} \left(\mathbb{E}_X \{f(X)\} - \mathbb{E}_Y \{f(Y)\} \right).$$

Think of f as distinguishing forecasts X from ground truth Y, as in a GAN.

If $\mathcal{F}$ is the unit ball in the RKHS generated by kernel $K(x, y)$, then

$$MMD[\mathcal{F}] = \mathbb{E}\left\{ K(X,Y) - \frac{1}{2}K(X,X') - \frac{1}{2}K(Y,Y') \right\}.$$

A parallel set of literature exists analyzing MMDs [G12]

Brier score for binary tail events

Given

- Binary tail event $Y > \tau$
- Forecast X
- (possibly estimated) probability $p = \mathrm{P}[X > \tau]$

The *Brier Score* is

$$BS(\tau) = \mathbb{E}\left\{|p - \mathbf{1}_{Y>\tau}|^2\right\}.$$

Note that

$$\int BS(\tau)\,\mathrm{d}\tau = \int |P[X > \tau] - Q[Y > \tau]|^2\,\mathrm{d}t + \text{const}$$
$$= CRPS + \text{const}.$$

Strictly proper for the tail event Y > τ

Subpar 😈 scores (guess why)

Square the norms in the energy score

$$\mathbb{E}\|X - Y\|^2 - \frac{1}{2}\mathbb{E}\|X - X'\|^2$$
$$= \|\mathbb{E}\{X\} - \mathbb{E}\{Y\}\|^2 + \mathrm{Var}\{Y\}.$$

Proper but not strict
Any X with the same mean as Y is a minimizer

Mean square error

$$\mathbb{E}\|X - Y\|^2$$
$$= \|\mathbb{E}\{X\} - \mathbb{E}\{Y\}\|^2 + \mathrm{Var}\{X\} + \mathrm{Var}\{Y\}.$$

Not proper
X = E[Y] (deterministic) is the minimizer

Quadratic score

$$\mathbb{E}\left\{-2p(y) + \int p(x)^2 \, \mathrm{d}x\right\}.$$

Strictly proper but not stable
p(y) ~ e^N has huge variance

Linear score

$$\mathbb{E}\{-p(y)\}.$$

Not proper (& unstable)
Prefers distributions peaked near the modes

[GR]

What compromises do these scores make?

Since our model is not perfect, we do not achieve the minimum.

- Given these restrictions, what distribution will be chosen as the minimizer?

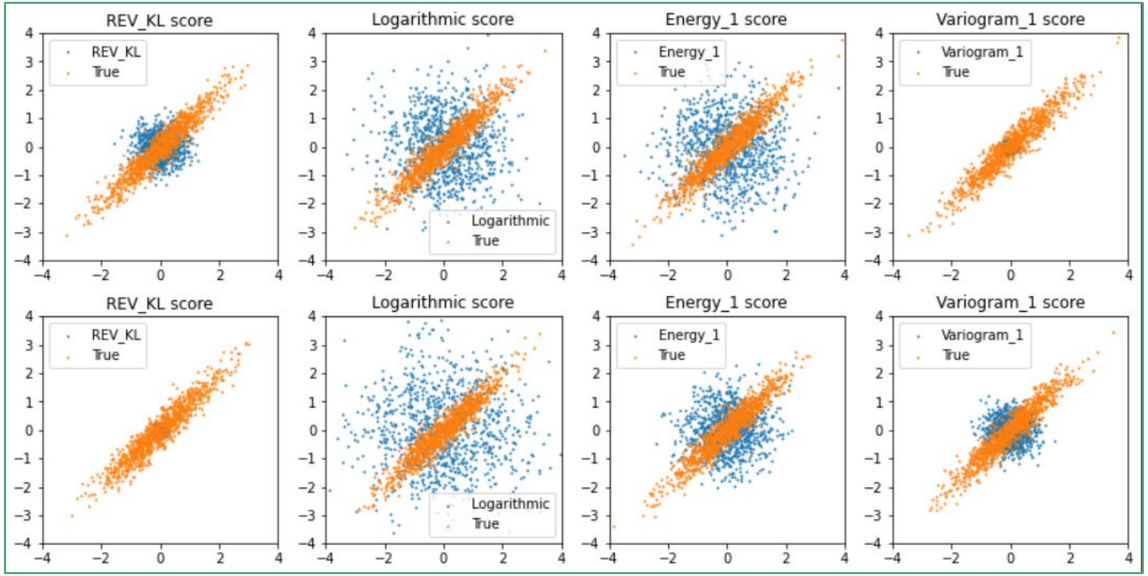
More study needed here

Correlated 2D Gaussian

Y ~ correlated 2D Gaussian

X ~ uncorrelated 2D Gaussian with variance σ^2.

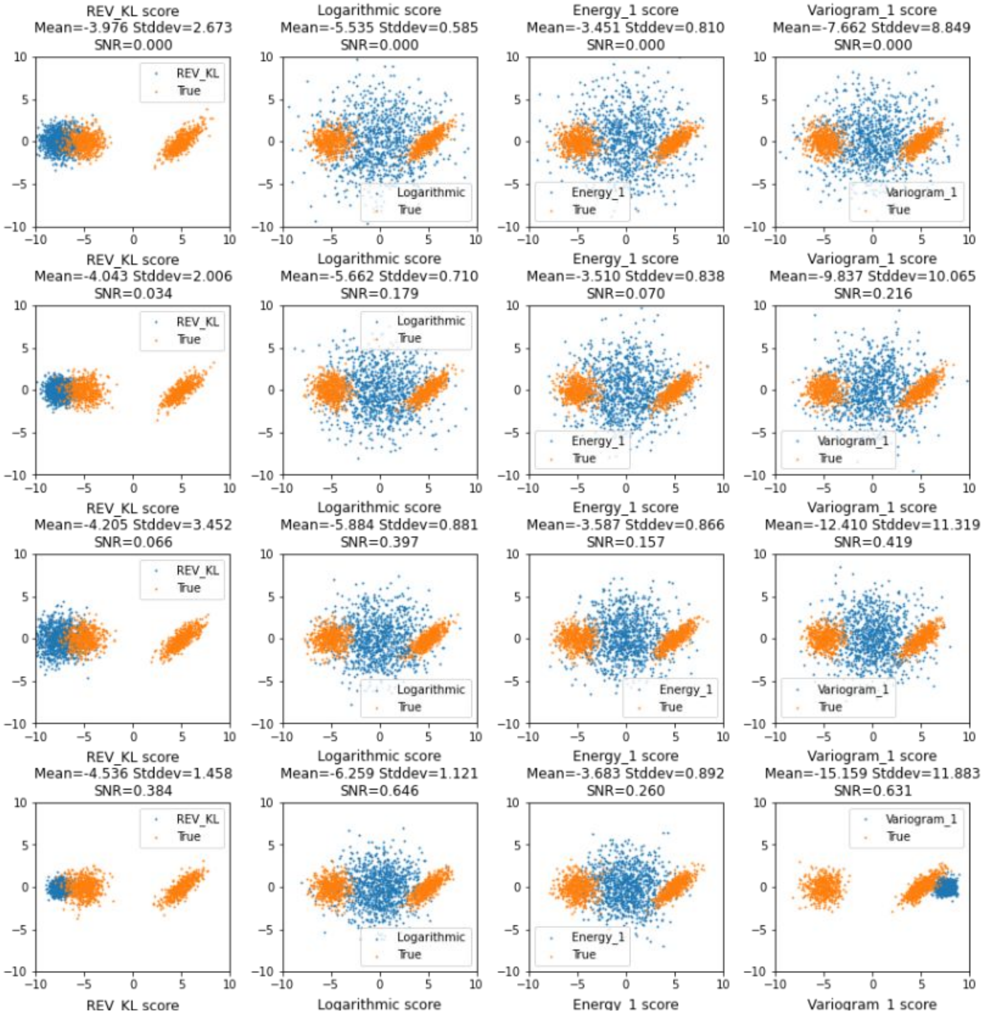
Which σ gives the best score?



First choice for each score

Second choice for each score

Bimodal Normal



Signal to Noise Ratio

Since we only have finite data, so we compute

$$\sum_{n=1}^{N} \mathcal{L}(X_n, X_n', Y_n) = \mathbb{E}\left\{\mathcal{L}(X, X', Y)\right\} + \text{noise}.$$

Of interest is the signal to noise ratio

$$SNR = \sqrt{\frac{\mathbb{E}\left\{\mathcal{L}(X, X', Y) - \mathcal{L}(Y, Y', Y'')\right\}}{\mathrm{Var}\left\{\mathbb{E}\left\{\mathcal{L}(X, X', Y) - \mathcal{L}(Y, Y', Y'')\right\}\right\}}}.$$

$1/SNR^2$ is approximately the number of samples needed.

Signal to noise ratio test

Setup

- Fit a 1000 dimensional Gaussian
- Sweep parameter & compute scores
- SNR = (best_score - score) / stddev
- # Samples needed ∝ 1 / SNR^2

Results

- Energy (2 ensemble) has SNR worse than 5x lower

  ⇒ Needs > 25x as many samples

  ⇒ Needs > 50x as much compute

Warning: This used an older computation of SNR!

parameter	SNR[Logarithmic]	SNR[Energy_1]
0.948	0.082	0.009
0.896	0.326	0.041
0.844	0.727	0.096
0.792	1.282	0.176
0.739	1.999	0.283
0.687	2.841	0.415
0.635	3.810	0.574
0.583	4.890	0.759
0.531	6.065	0.970

Tuning a probabilistic model

The setup

- Your team is developing a probabilistic model
- You have *many many* forecasts $\{X_n\}$ and ground truth examples $\{Y_n\}$

We want to

- Help scientists answer, “did this change help or hurt?”

In my experience, you end up

- Running giant evaluations on a cluster
- Output HTML summaries

Cluster jobs using Beam

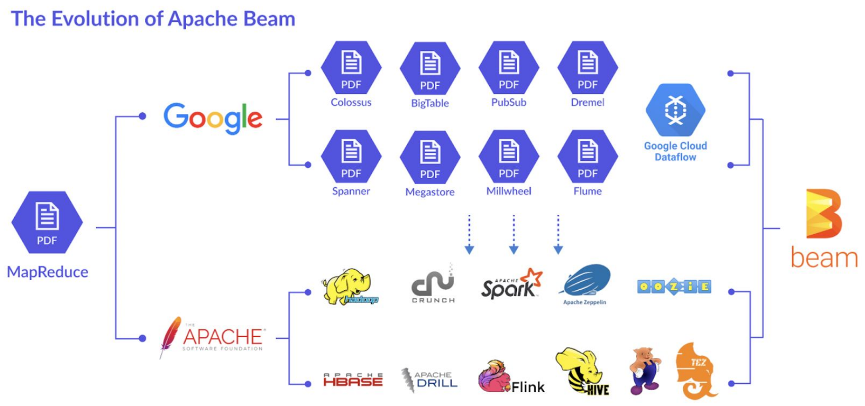


Figure 1. Evolution of Apache Beam. [Source]

Beam

- Allows robust use of 1000's of machines
- Is somewhat efficient
- Provides somewhat sane job monitoring

Maintaining Beam pipelines may not seem "Glorious", but…

- if you run the evals you're using stats to influence decisions
- building models is often "random tweak, train, check"

HTML Output : Page 1

2020-01-02 to 2022-11-03 reforecasts summary for xid=116889089, wid=6, training_step=70000, ensemble_size=2

Table of Contents

Single forecast analysis was skipped

Long forecast analysis

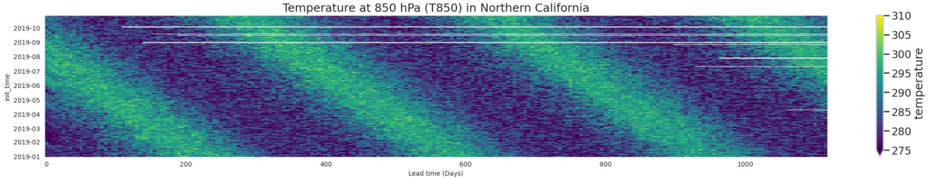
Long forecast analysis| § 240x121/probabilistic/metrics| § 240x121/probabilistic/spectrum| § resample-1w 240x121/probabilistic/metrics| § resample-1w 240x121/probabilistic/spectrum| § 240x121/ensemble binary/metrics| § resample-1w 240x121/ensemble binary/metrics| § 240x121/deterministic spatial

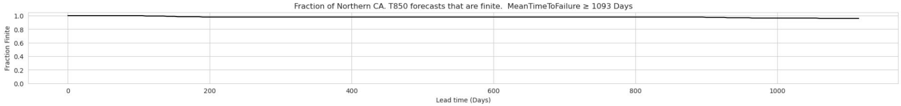
Forecasts began between 2019-01-01 and 2019-10-26

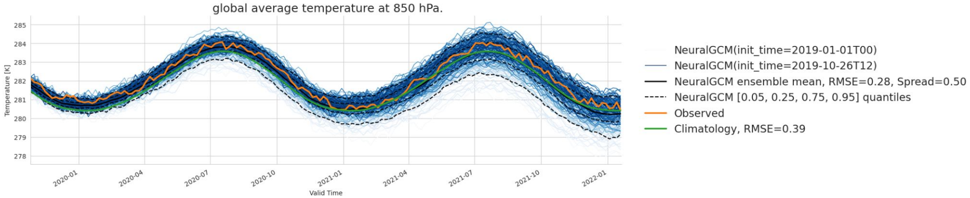
long_forecast_path: /namespace/gas/primary/whirl/models/gcm/r=3.2/langmore/20240620/d2faa2ec/PS_TL63_37_epd_rand_randomness_sweep/training/train_dataset/FieldStride=2__NConstFields=5__erand=Gaussian__peaklr=0.001__S=46/reforec

observations_path: /placer/prod/home/gas/whirl/era5/with_clouds_and_ozone/1959_to_20230111_64x32_cds_37_gauss_conservative_fillnan_nearest.zarr

climatology_path: /cns/od-d/home/gas/whirl/datasets/era5-climatology/langmore/1990_to_2020_64x32_gauss_conservative_fillnan_nearest.zarr/







HTML Output : Page 2

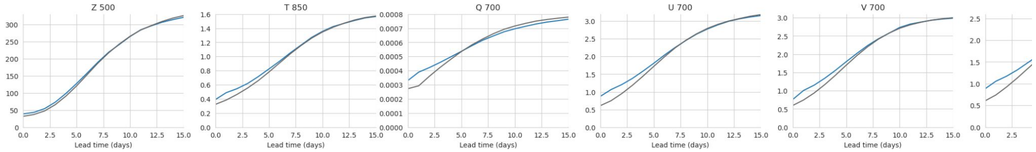
§ 240x121/probabilistic/metrics

Long forecast analysis| § 240x121/probabilistic/metrics| § 240x121/probabilistic/spectrum| § resample-1w 240x121/probabilistic/metrics| § resample-1w 240x121/probabilistic/spectrum| § 240x121/ensemble binary/metrics| § resample-1w 240x121/ensemble binary/metrics| § 240x121/deterministic s

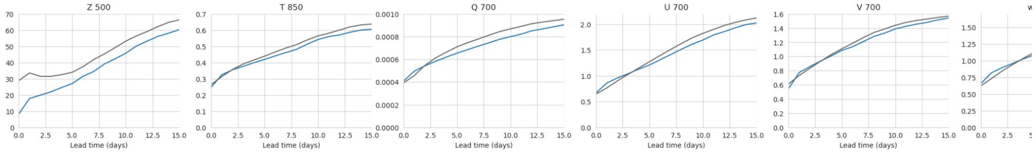
prediction_path: `/namespace/gas/primary/whirl/models/gcm/r=3.2/langmore/20240620/d2faa2ec/PS_TL63_37_epd_rand_randomness_sweep/training/train_dataset/FieldStride=2__NConstFields=5__erand=Gaussian__peaklr=0.001__S=46/refore`

baseline_path: `/namespace/gas/primary/whirl/datasets/ecmwf-ext-ens/langmore/2020-2022/evals/240x121_equiangular_with_poles_conservative/probabilistic.nc`

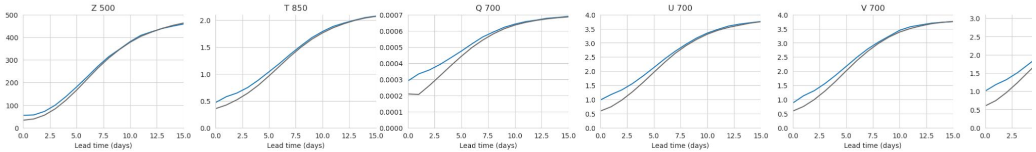
CRPS(Global Error)



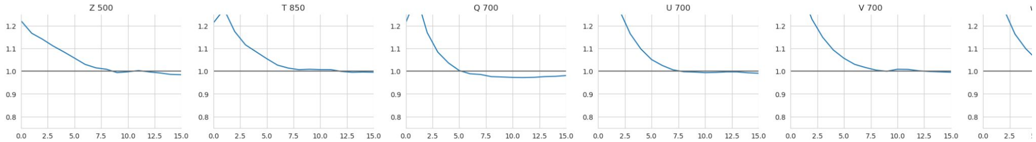
CRPS(Tropics Error)



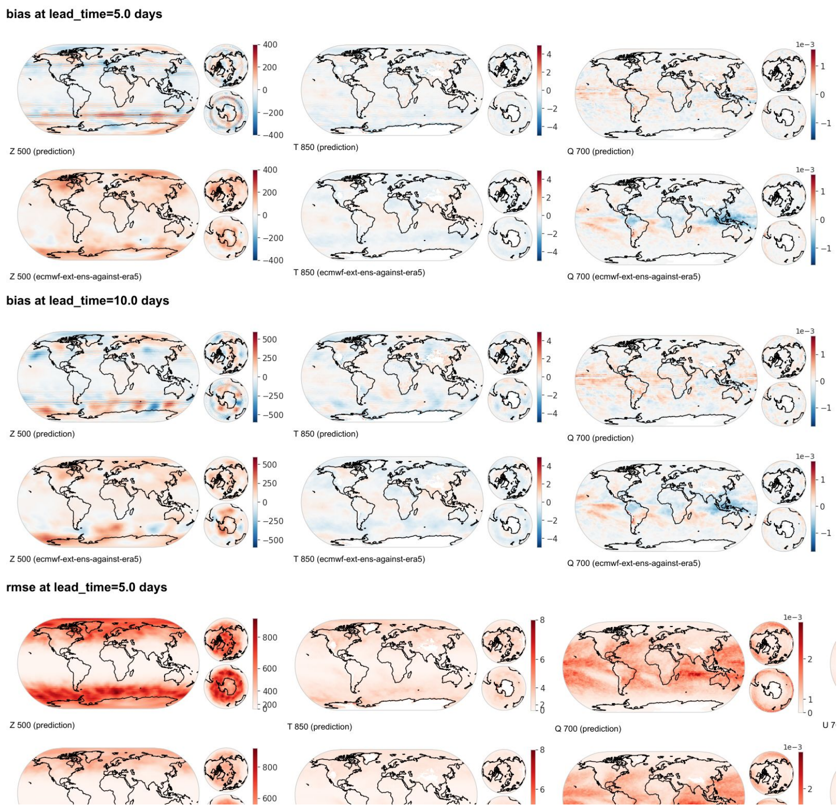
CRPS(Extra-tropics Error)



Relative CRPS(Global Error)



HTML Output : Page 3



Part 2

ML weather forecasting
and NeuralGCM

Outline

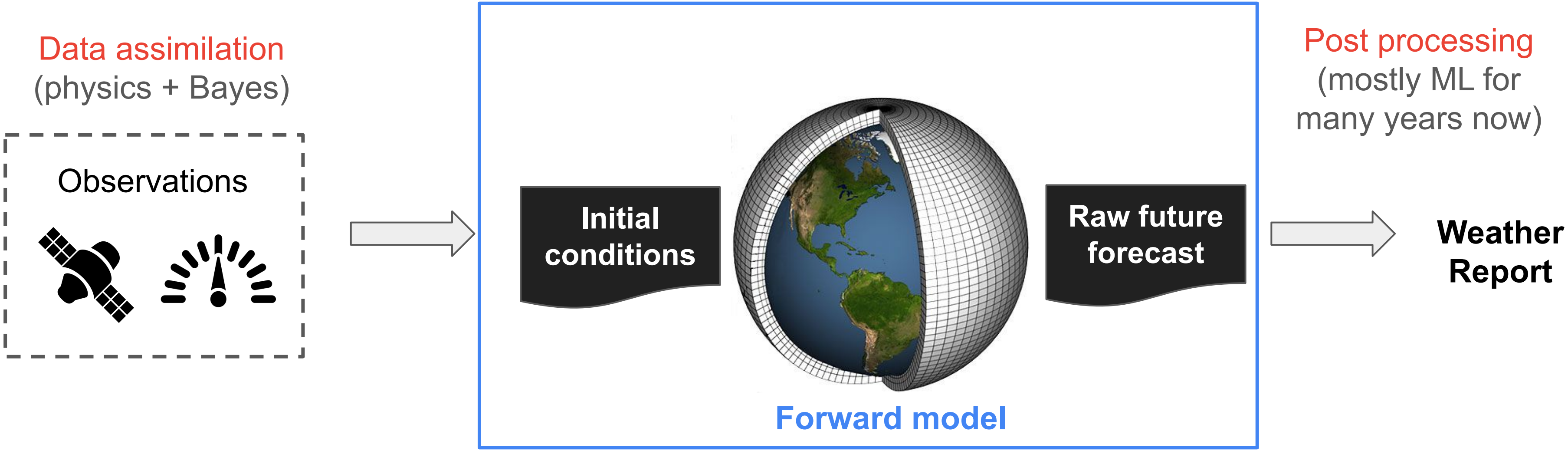
ML weather prediction is at state of the art

NeuralGCM: Neural network augmented (differentiable) fluid solvers for weather prediction

The **math** behind *probabilistic* NeuralGCM

ML Weather forecast overview

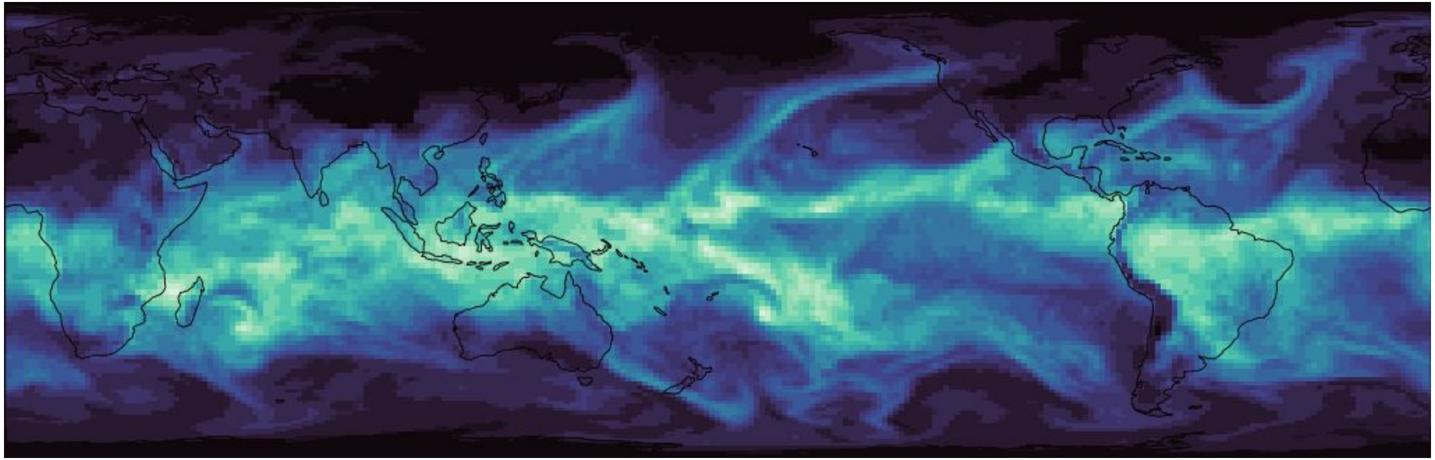
ML is used in a weather forecasting **System**



Recent headlines in ML-forecasting are primarily *Global Circulation Models*

GCM = Global Circulation Model

Models global flow of humidity, temperature, and wind



Global humidity

Question: What training data is used for ML forward models?

Answer: Reanalysis (ECMWF)

- Retrospective reconstruction of weather
  - Dense, includes all variables of interest
- Done with traditional "pure physics" models

Q: How can ML be "better than existing physics models" if the training data comes from physics models?

A: ML can forecast *future* weather that is closer to the *retrospective reconstruction*

A: ML forecasts are made orders of magnitude faster

NeuralGCM

Open source

Dycore: https://github.com/google-research/dinosaur
Model: https://github.com/google-research/neuralgcm
Paper: Neural General Circulation Models

Traditional Global Circulation Models

Traditional GCMs rely on too many hand-tuned parameters & empirical equations

"Dynamical core"

$$\frac{d\mathbf{u}}{dt} + f\mathbf{k}\times\mathbf{u} + \frac{1}{\rho}\nabla_z p = \mathbf{0}$$

$$\frac{\partial\rho}{\partial t} + \nabla_z\cdot(\rho\mathbf{u}) + \frac{\partial\rho w}{\partial z} = 0$$

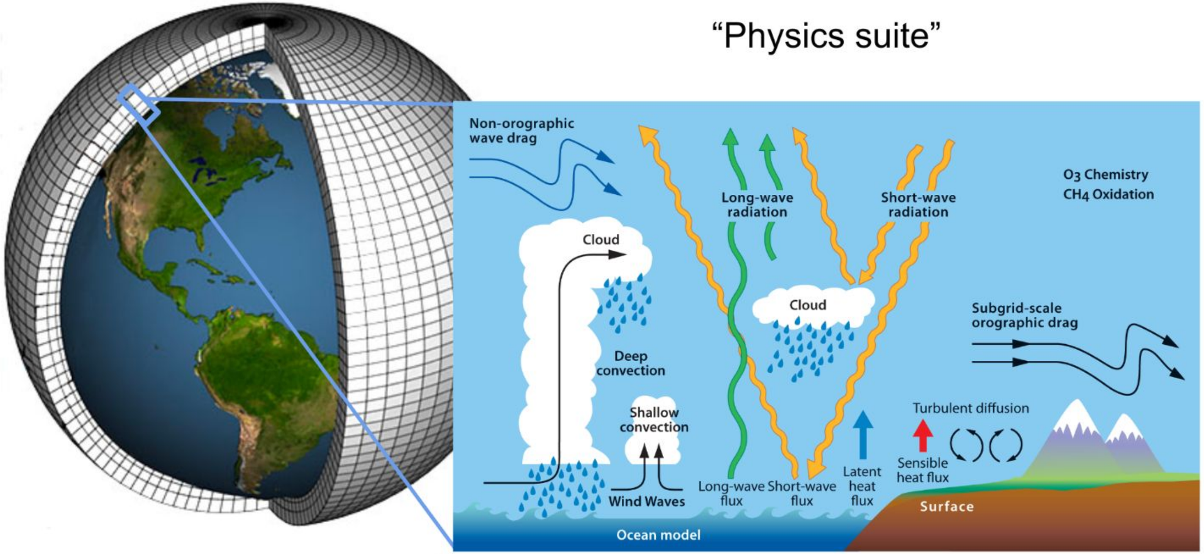
$$\frac{dT}{dt} - \frac{\omega}{c_p\,\rho} = 0$$

$$\frac{\partial p}{\partial z} = -\rho g$$

$$p = \rho R T$$

Fluid dynamics on the surface of a rotating sphere

"Physics suite"



Many PhD theses (100k-1M lines of Fortran)

NeuralGCM

Using ML to learn physical tendencies (rates of change)

“Dynamical core”

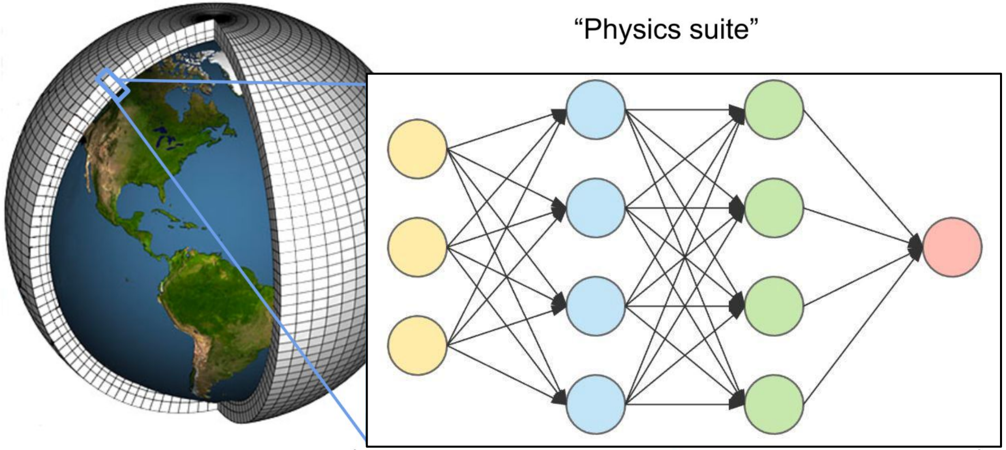
$$\frac{d\mathbf{u}}{dt} + f\mathbf{k}\times\mathbf{u} + \frac{1}{\rho}\nabla_z p = \mathbf{0}$$

$$\frac{\partial\rho}{\partial t} + \nabla_z\cdot(\rho\mathbf{u}) + \frac{\partial\rho w}{\partial z} = 0$$

$$\frac{dT}{dt} - \frac{\omega}{c_p\,\rho} = 0$$

$$\frac{\partial p}{\partial z} = -\rho g$$

$$p = \rho R T$$



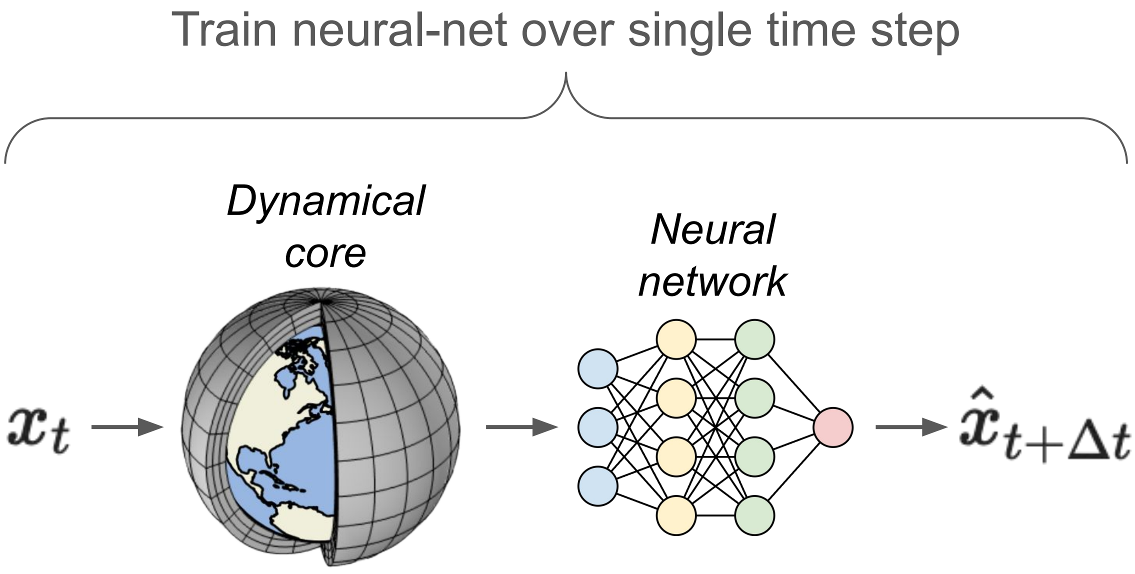
“Physics suite”

Solve dynamical equations on TPUs

Learn “physics suite” from data

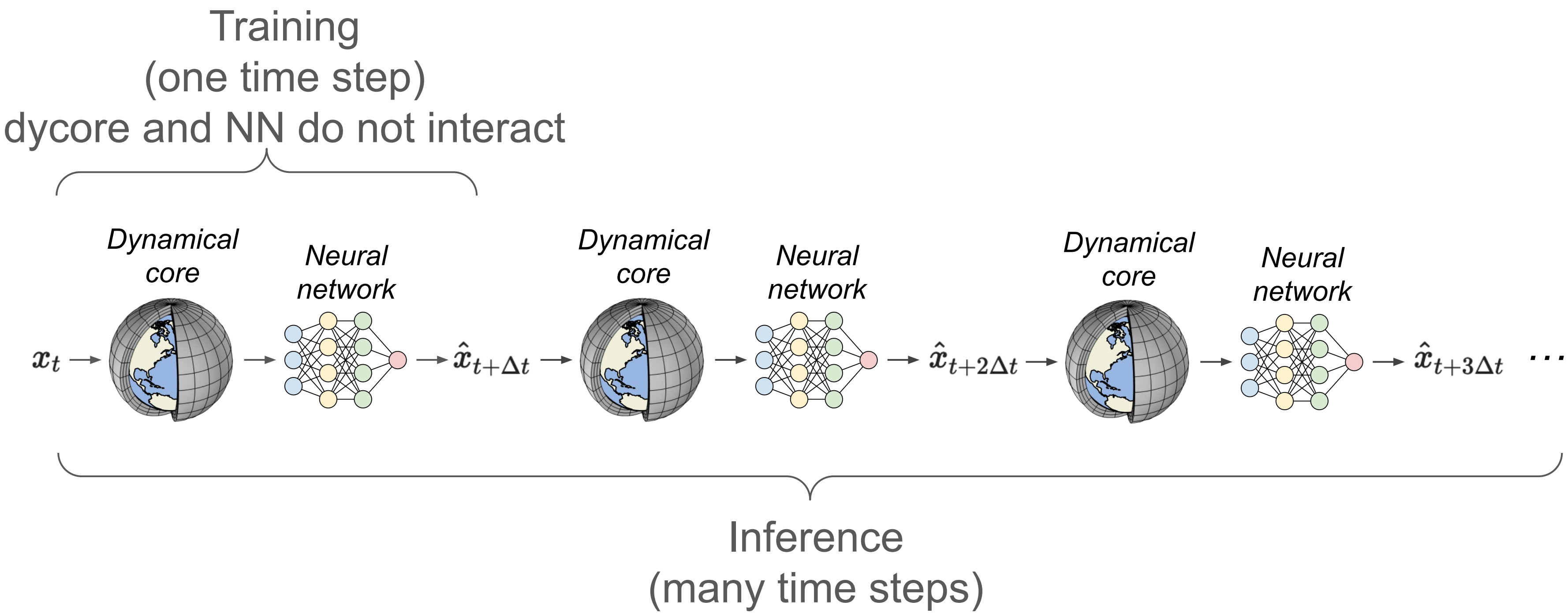
$$\frac{\partial X_t}{\partial t} + (X_t\cdot\nabla)X_t - \nu\nabla^2 X_t = \overbrace{\Psi(X_t)}^{\text{ML-tendencies}}$$

Conventional hybrid models train an ML model "offline"

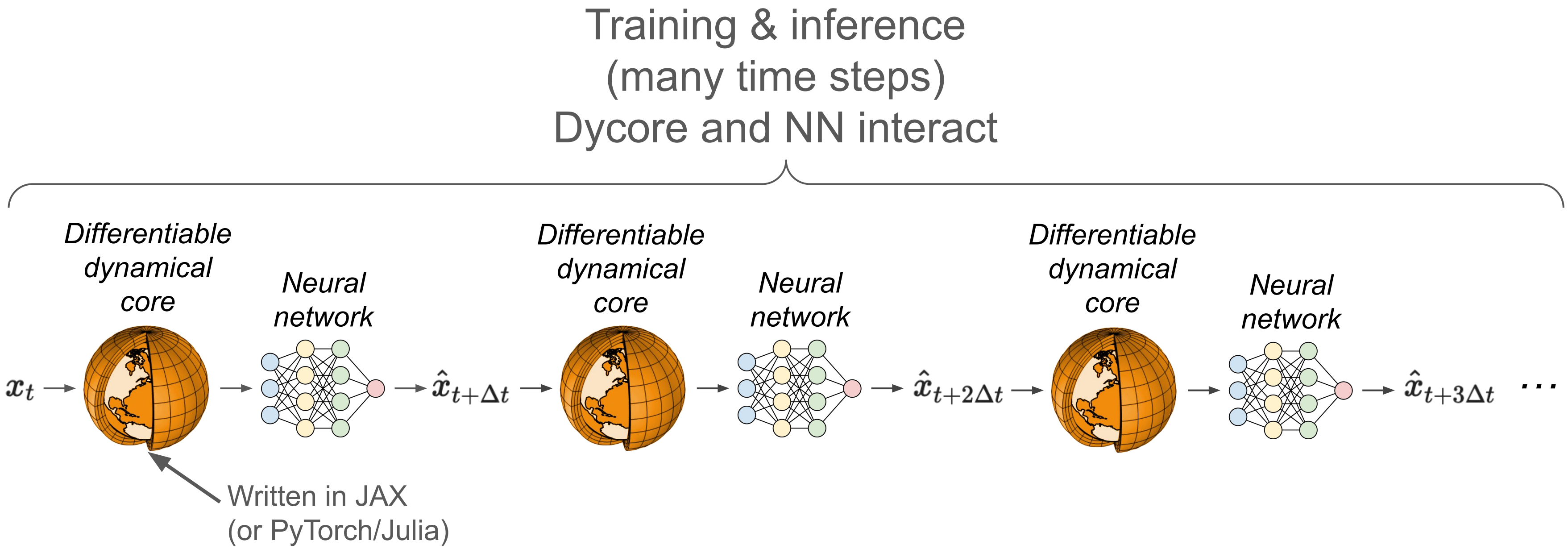


Diagnose the missing physics and build a training data set

Conventional hybrid models train an ML model "offline"

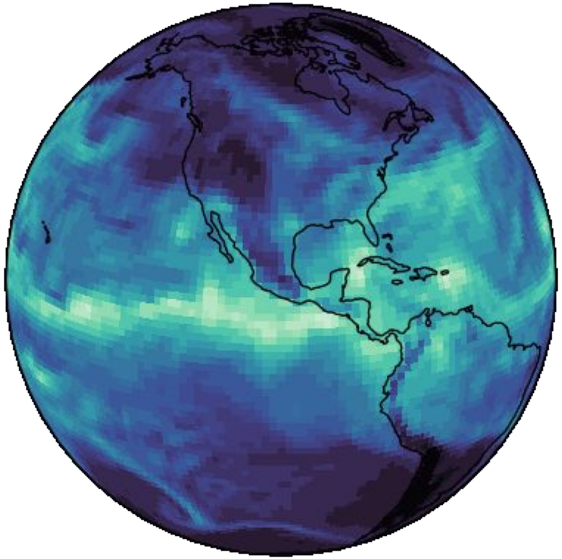


Differentiable hybrid models can be trained end-to-end for “online” performance

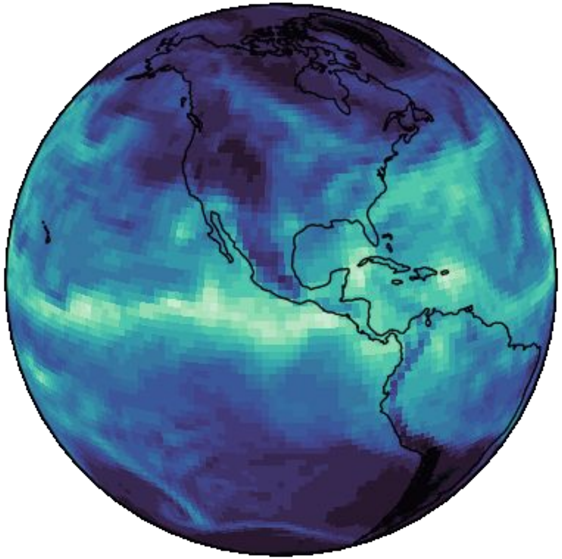


Forecasts are realistic

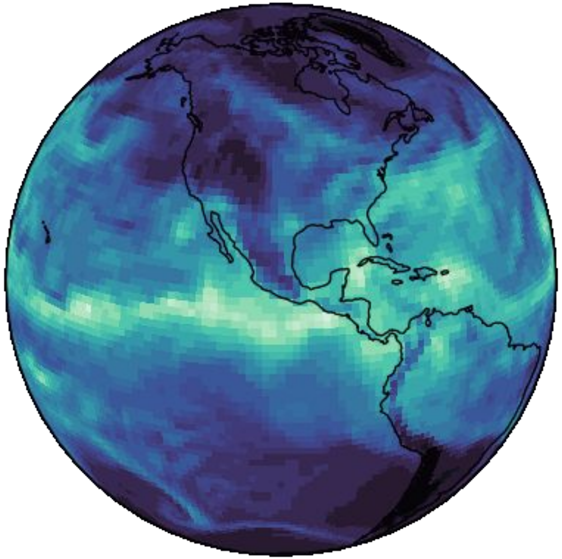
One is ground truth, the other two NeuralGCM ensemble members



Option A



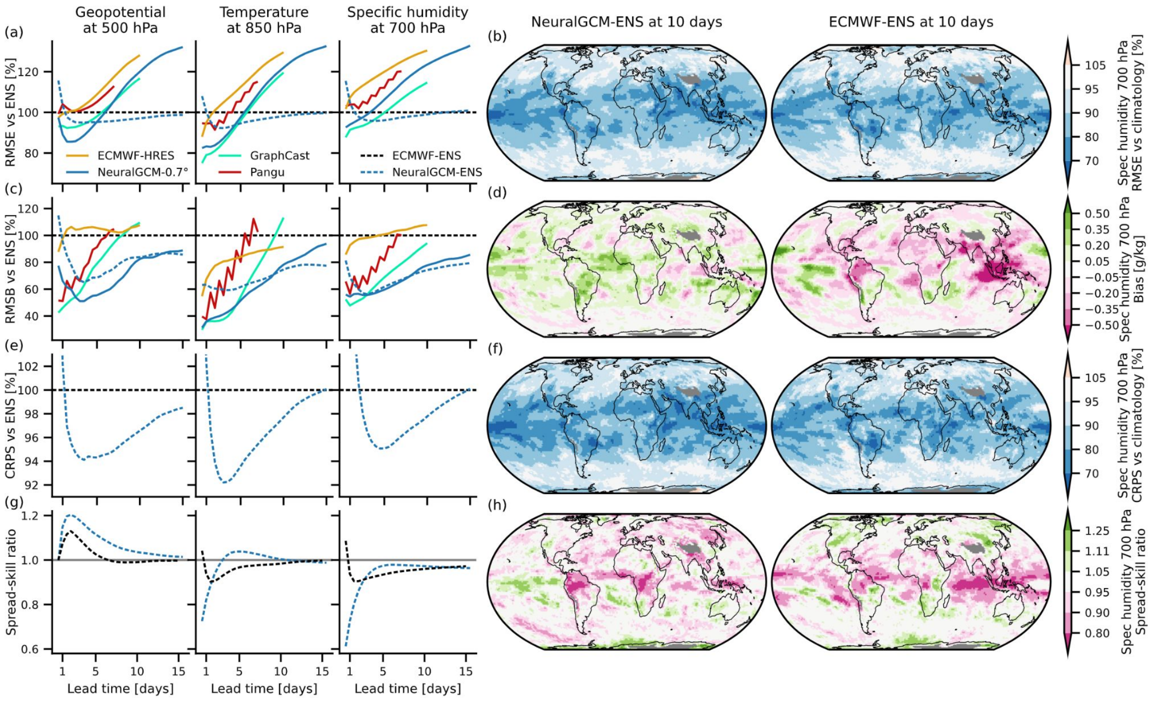
Option B



Option C

Total column water, 0-15 days

NeuralGCM was the first ML model to beat ECMWF’s ensemble on RMSE, Bias, CRPS



WeatherBench 2

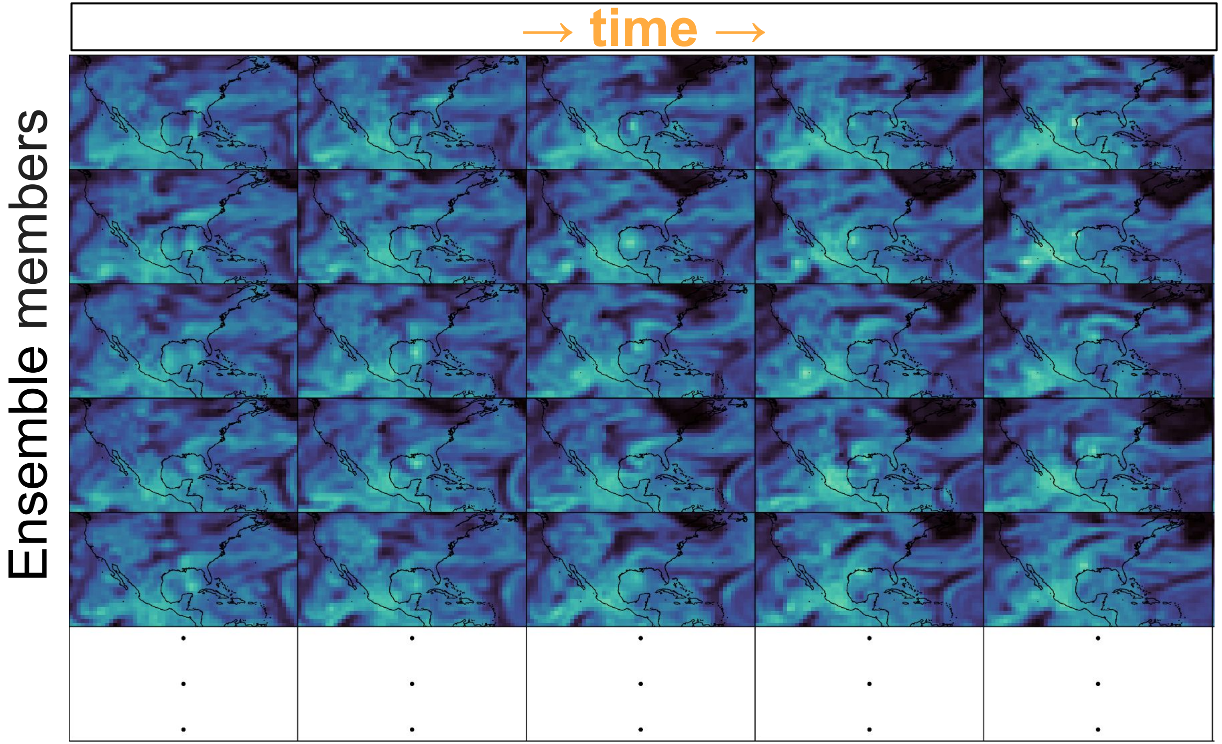
Stephan Rasp et al

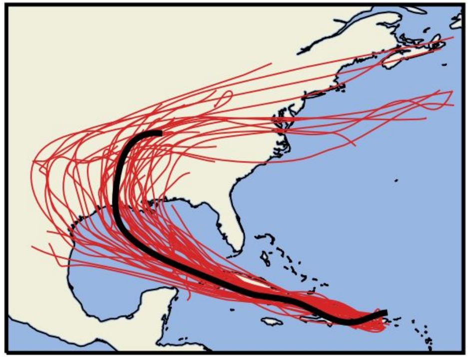
Caveat: All ML models are much lower resolution

New model (GenCast) also beats ECMWF’s ensemble

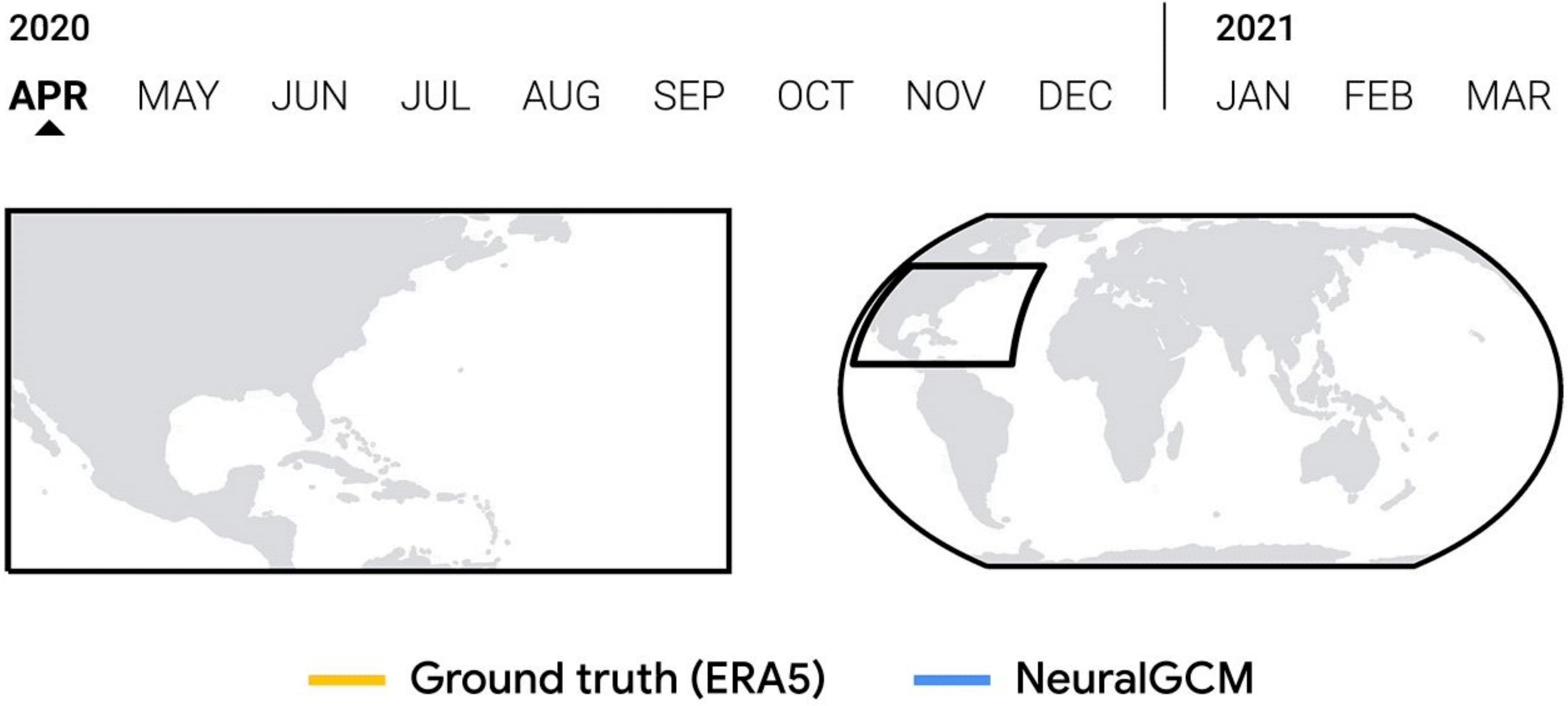
Ensembles capture uncertainty

Ensemble forecasts a realistic range **cyclone tracks**





NeuralGCM near-term climate forecasts also have realistic distributions of tropical cyclones

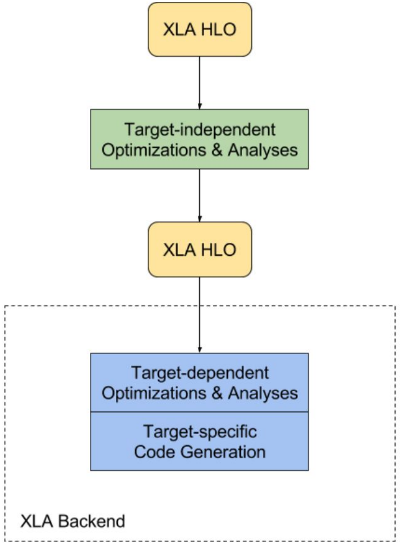


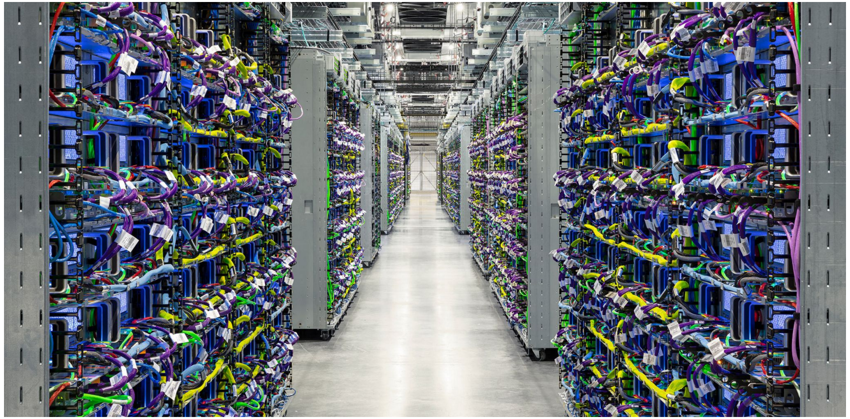
JAX, XLA, and TPUs

JAX (Python) code – **XLA** compiled – runs on **TPUs**

```
import jax
import jax.numpy as jnp

def square(x):
  return x * x

grad_square = jax.grad(square)

print(square(3.), grad_square(3.))
```

```
9.0 6.0
```





Where does the speedup come from?

ECMWF HRES

- 9 km resolution
- 15 day simulation
  - 52 minutes on 64 x 128 core CPUs

NeuralGCM

- 70 km resolution
- 15 day simulation
  - 5.4 minutes on 1 TPU ($1 / hr)
  - can easily increase number of TPUs for an ensemble

Scaling the **70 km** → **9 km** would *naively* require ≈ 7^3 more TPUs

- this scaling may not actually work

9 km is not necessary

- small-scale phenomena result in learnable patterns at larger scales

The **math** behind *probabilistic* NeuralGCM

Training generative models with scoring rules

The network learns to transform random fields

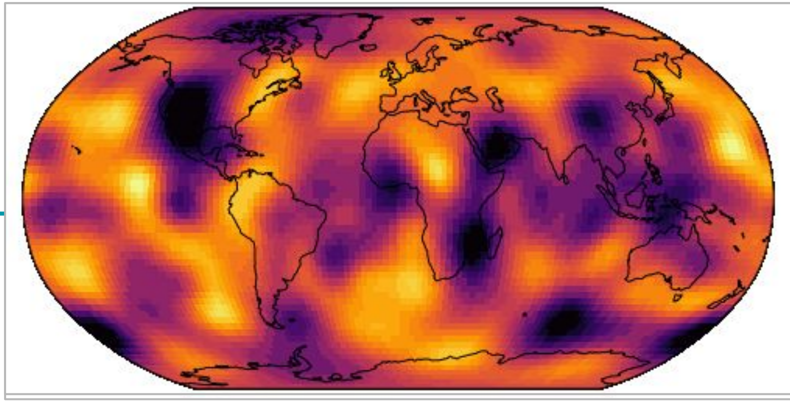
Start with K≈10 Gaussian random fields ($Z_t^{(1)}, \ldots, Z_t^{(K)}$), with

- correlation lengths ($\lambda_1, \ldots, \lambda_K$)
- correlation times ($\tau_1, \ldots, \tau_K$)

The network learns to transform the random fields

The field parameters $\{\lambda_i, \tau_i\}$ are learned as well

$$\frac{\partial X_t}{\mathrm{d}t} + (X_t \cdot \nabla)X_t - \nu\nabla^2 X_t = \overbrace{\Psi(X_t, Z_t)}^{\text{ML-tendencies}}$$



What loss function will encourage proper use of the random fields?

Mean squared error

...why it favors blurry forecasts

Given stochastic ground truth Y and forecast X,

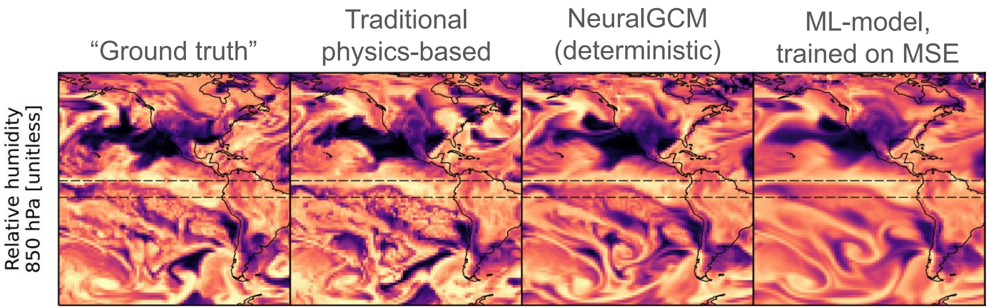
$$\begin{aligned} \mathrm{MSE} :&= \mathbb{E}\|X - Y\|^2 \\ &= \|\mathbb{E}X - \mathbb{E}Y\|^2 + \mathrm{Var}\{X\} + \mathrm{Var}\{Y\}. \end{aligned}$$

- MSE is minimized by forecasting $X \equiv \mathbb{E}Y$
- Regardless of your forecast, *any variance in X only hurts!*

Alternative explanation:

If a storm may be here or there…

you minimize MSE by forecasting a blurry cloud everywhere



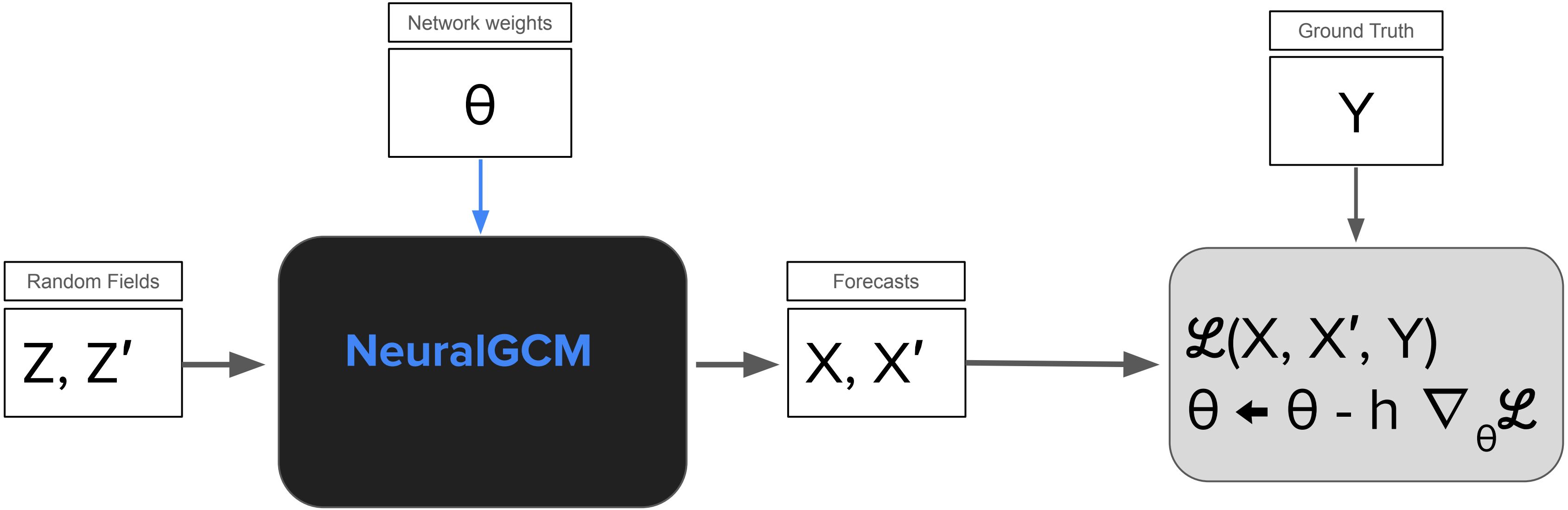
Common misleading practice:

- train to minimize MSE
- show better RMSE than a physics-based model
- claim to be SOTA

Generative-only + scoring rule

1. Transforms random Z, Z' into forecasts X, X'
2. Evaluates a loss based on a proper scoring rule
3. Takes a gradient step to minimize the loss

No likelihood estimate!



Continuously Ranked Probability Score (CRPS)

For scalar predictions X, X' and ground truth Y,

$$CRPS = \mathbb{E}|X - Y| - \frac{1}{2}\mathbb{E}|X - X'|$$
$$= \int_{-\infty}^{\infty} (\mathrm{P}[X \leq y] - \mathrm{P}[Y \leq y])^2 \, \mathbf{d}y + \mathbf{const.}$$

- Strictly proper
  ⇒ X ~ Y is the unique minimizer
- Generalizes MAE

In $\mathbb{R}^N$,

$$CRPS = \sum_{n=1}^{N} \left[\mathbb{E}|X_n - Y_n| - \frac{1}{2}\mathbb{E}|X_n - X_n'| \right]$$
$$= \mathbb{E}\|X - Y\|_1 - \frac{1}{2}\mathbb{E}\|X - X'\|_1.$$

- ~~Strictly~~ proper
  ⇒ *any* X with correct marginals is *a* minimizer
- Generalizes MAE
- We let $\{X_n\}$ be components in spatial & spectral basis

[GR]

How we trained with CRPS loss

Given neural network $\Psi(\cdot;\theta)$, parameterized by θ, ground truth Y_0

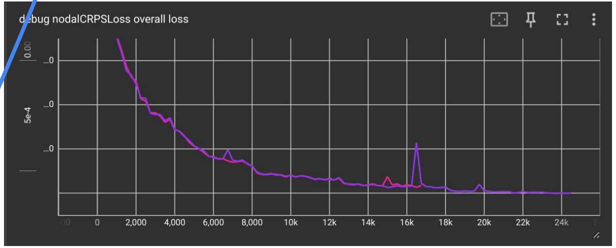
- Draw random initial perturbation ξ
- Initialize forecast $X_0 = Y_0 + \xi$
- Initialize random field Z_0
- Generate forecast X_t for $0 < t < N\tau$ by solving

$$\frac{\partial X_t}{\partial t} + (X_t \cdot \nabla)X_t - \nu\nabla^2 X_t = \Psi(X_t,\, Z_t;\theta),$$
$$\mathbf{d}Z_t = -BZ_t\,\mathbf{d}Z_t + \sigma\,\mathbf{d}W_t.$$

- ...Similarly for X'_t, Z'_t.
- Update parameters with gradient descent: $\theta \leftarrow \theta - h\nabla_\theta\mathcal{L}(\theta)$, where

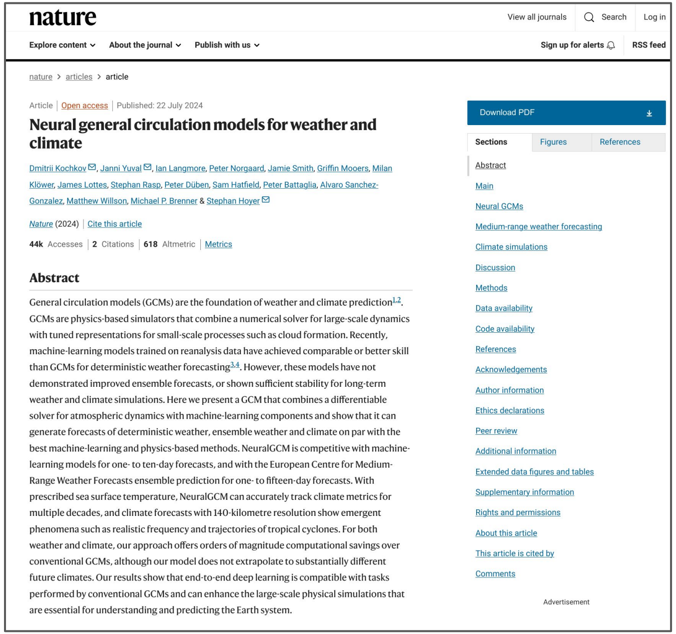
$$\mathcal{L}(\theta) = \sum_{n=1}^{N}\left[\|X_{n\tau} - Y_{n\tau}\|_1 + \|X'_{n\tau} - Y_{n\tau}\|_1 - \|X_{n\tau} - X'_{n\tau}\|_1\right]$$

Repeat with a new minibatch (SGD)



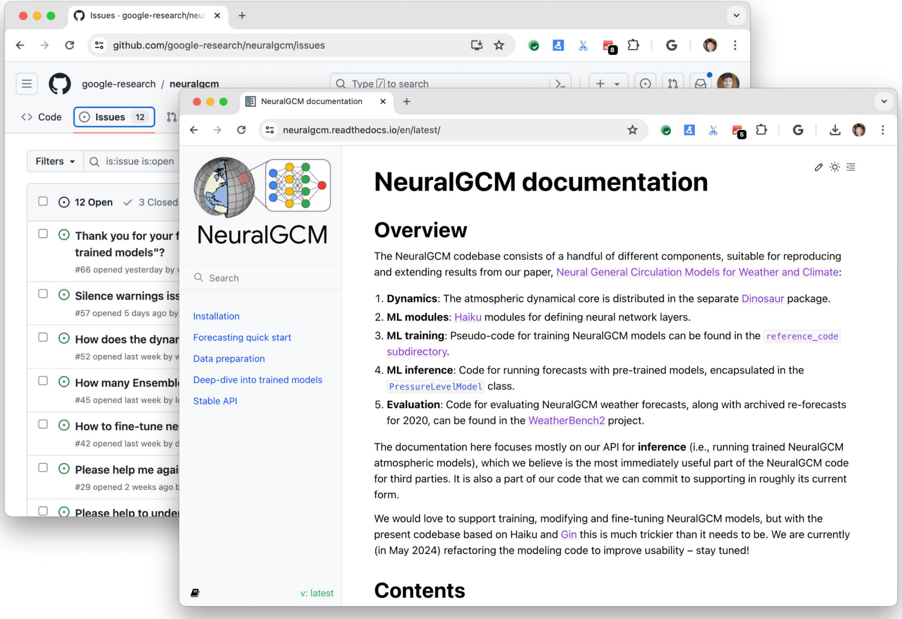
Learn more about NeuralGCM

Read the paper



nature.com/articles/s41586..

Run the open source code



github.com/google-research/neuralgcm

Part 3

JAX and JAX-CFD live tutorial

Using this colab notebook

Thank You!

Please send questions to: langmore@google.com

References

NeuralGCM: Neural General Circulation Models

GenCast: Diffusion based ensemble forecasting for medium-range weather

GR: Strictly Proper Scoring Rules, Prediction, and Estimation

P: Normalizing Flows for Probabilistic Modeling and Inference

S: Maximum Likelihood Training of Score-Based Diffusion Models

C: Online Variational Filtering and Parameter Learning

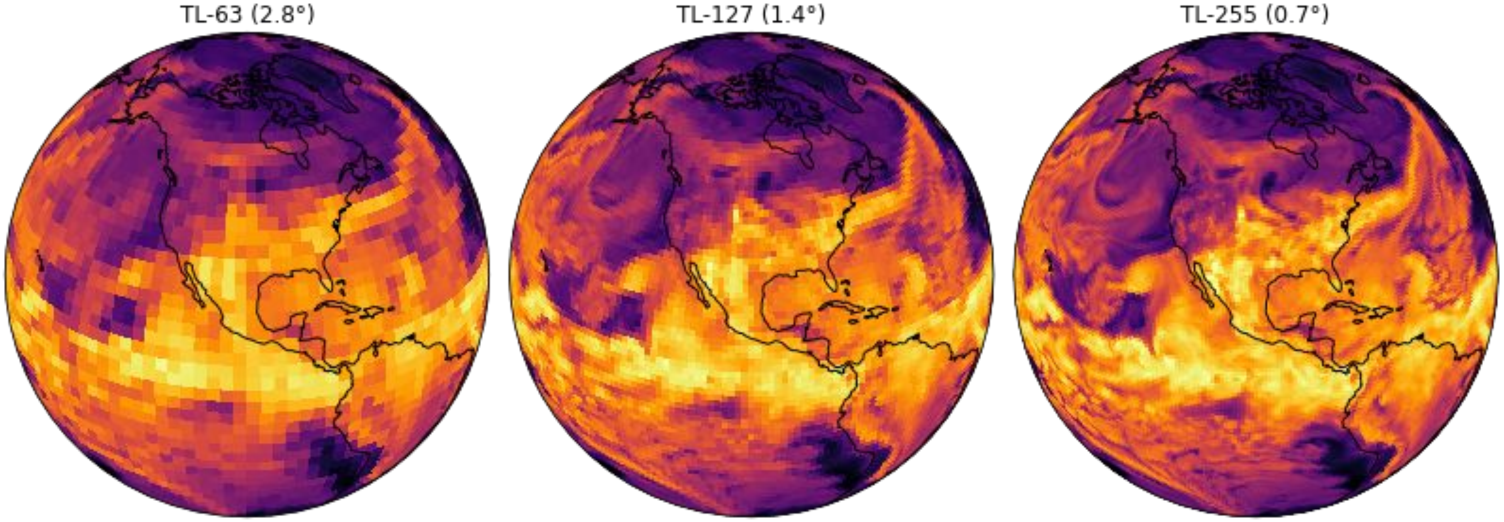
Pa: Lecture notes on properties of MLE

SZ: Energy statistics: A class of statistics based on distances

Appendix

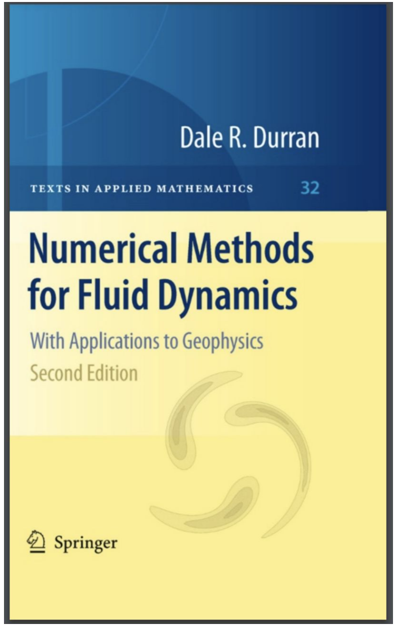
We train a hierarchy of models at different resolutions

We also train both deterministic & stochastic models (~2x more expensive)

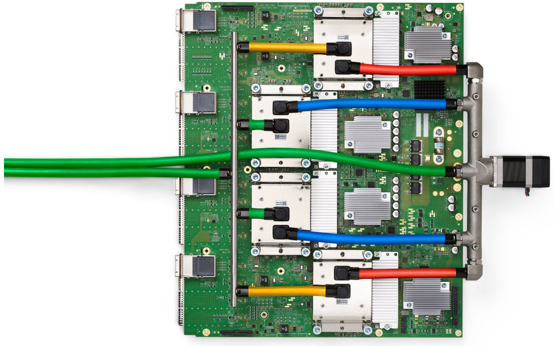


	TL-63	TL-127	TL-255
Inference:	350,000 sim days / day	69,000 sim days / day	7,300 sim days / day
Training:	1 day on 16 TPUs	1 week on 16 TPUs	3 weeks on 256 TPUs (16x model parallelism)

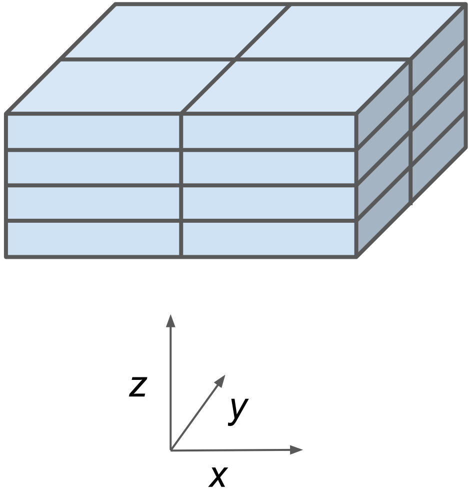
Our dynamical core solves the moist hydrostatic primitive equations with spectral methods



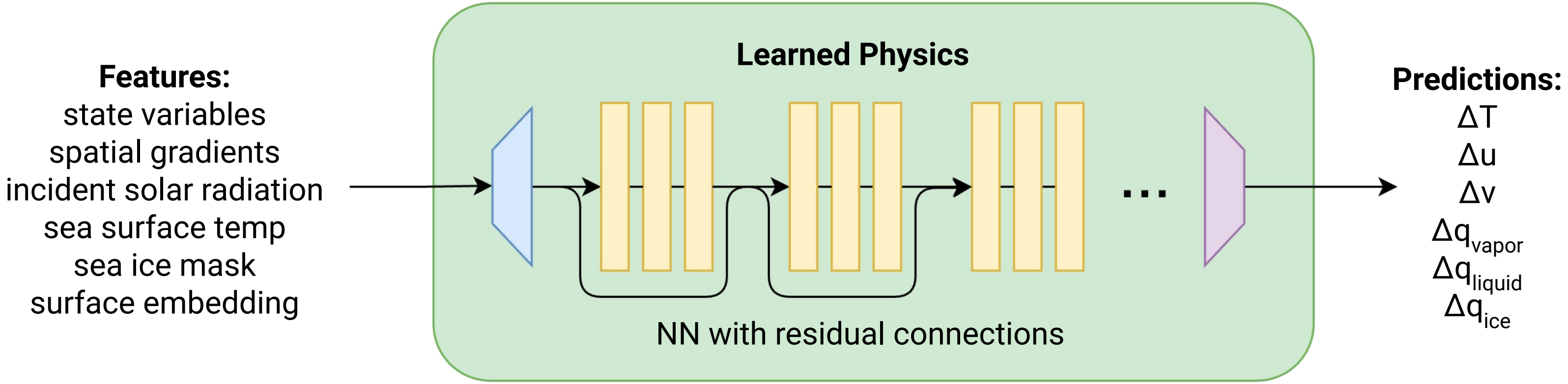
Written in JAX and runs fast on Google TPUs (transforms use 24 bit precision matmul)

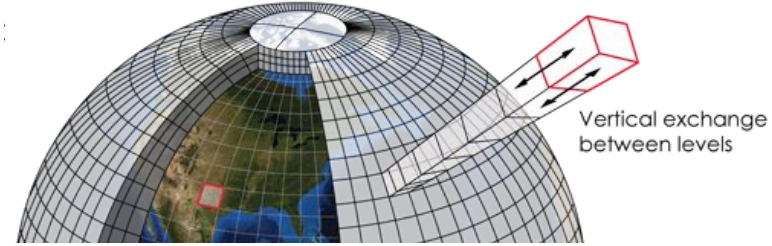


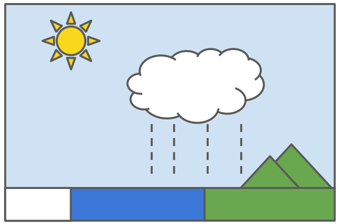
Up to 16x model parallelism



Our physics is a fully-connected neural net that acts on a single vertical column of the atmosphere

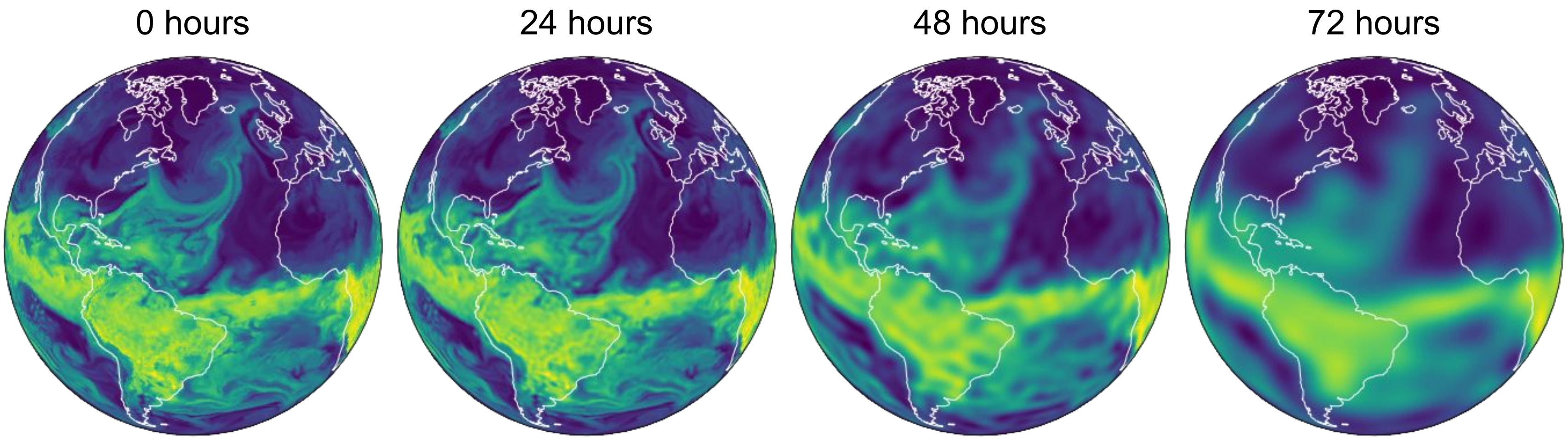






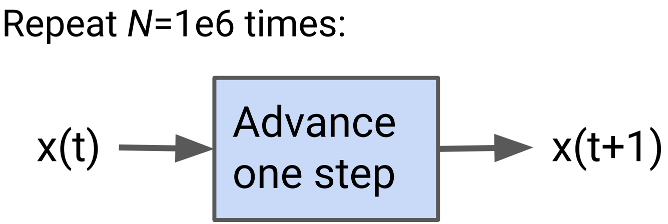
Deterministic Neural GCM loss terms

1. Squared error with spatial filtering by lead-time
2. Spectral loss
3. Bias loss



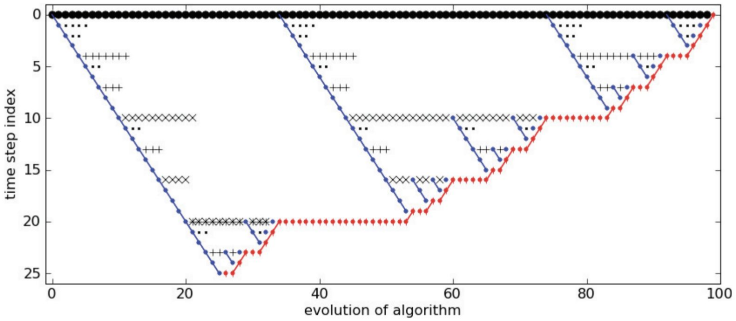
How differentiating simulations can go wrong: part 2

Problem: Storing every intermediate result can use a ridiculous amount of memory.



O(N) compute
O(N) memory

Solution: Gradient checkpointing (i.e., jax.remat)



O(N log N) compute
O(log N) memory