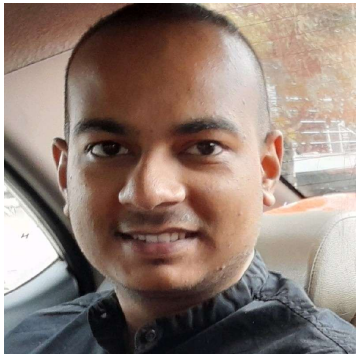
Multimodal data and model fusion using diffusion models

Romit Maulik,
Assistant Professor, The Pennsylvania State University,
Faculty Affiliate, Argonne National Laboratory.
Email: rmaulik@psu.edu

PennState

Acknowledgements



Dibyajyoti (Penn State)

Haiwen (Penn State)

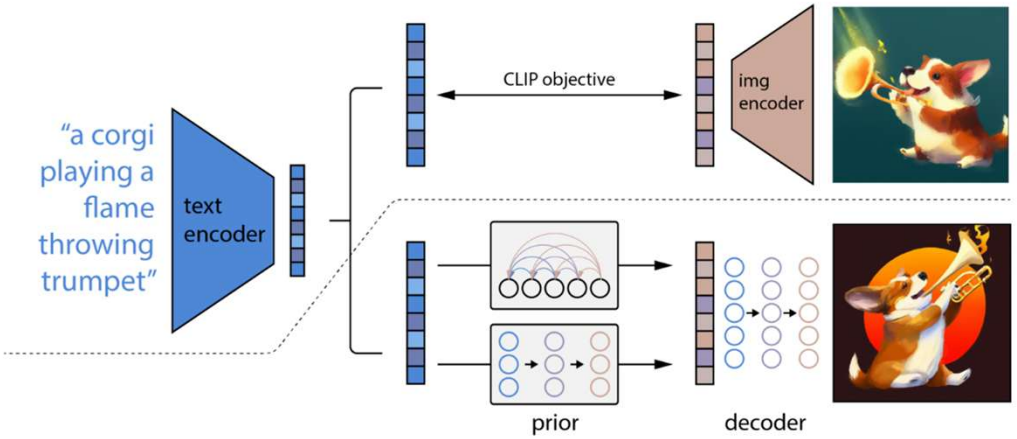
Collaborative with:
Troy Arcomano, Guido Cervone, Jason Stock

Generative machine learning

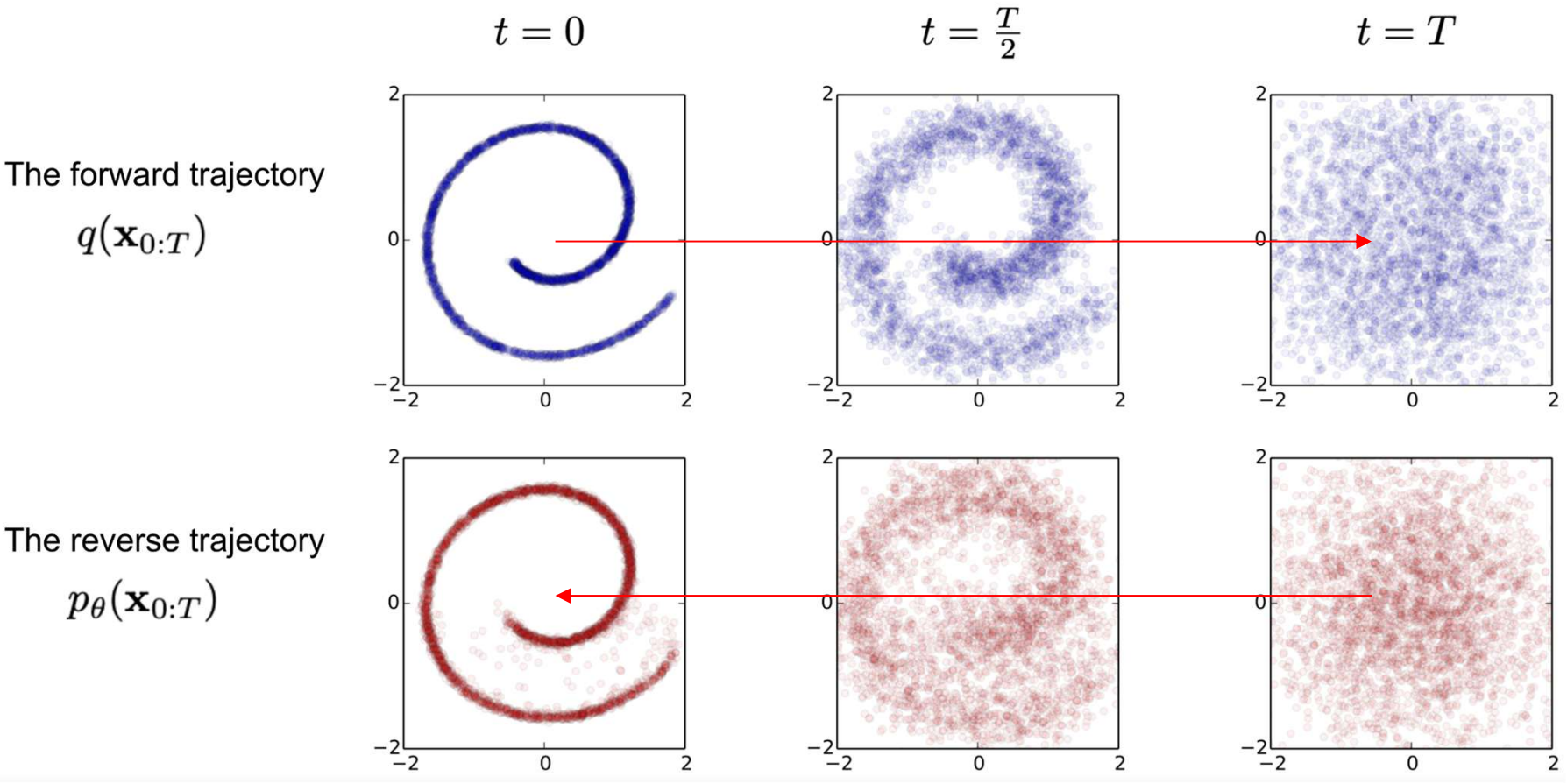
Generative machine learning is a branch of machine learning focused on **modeling data distributions so that new, realistic samples can be generated**.

Instead of only predicting labels or values (like in traditional discriminative models), generative models learn the underlying structure of the data and can create new instances that resemble the original dataset

State-of-the-art generative algorithms are all built on the so-called "diffusion modeling" approach.

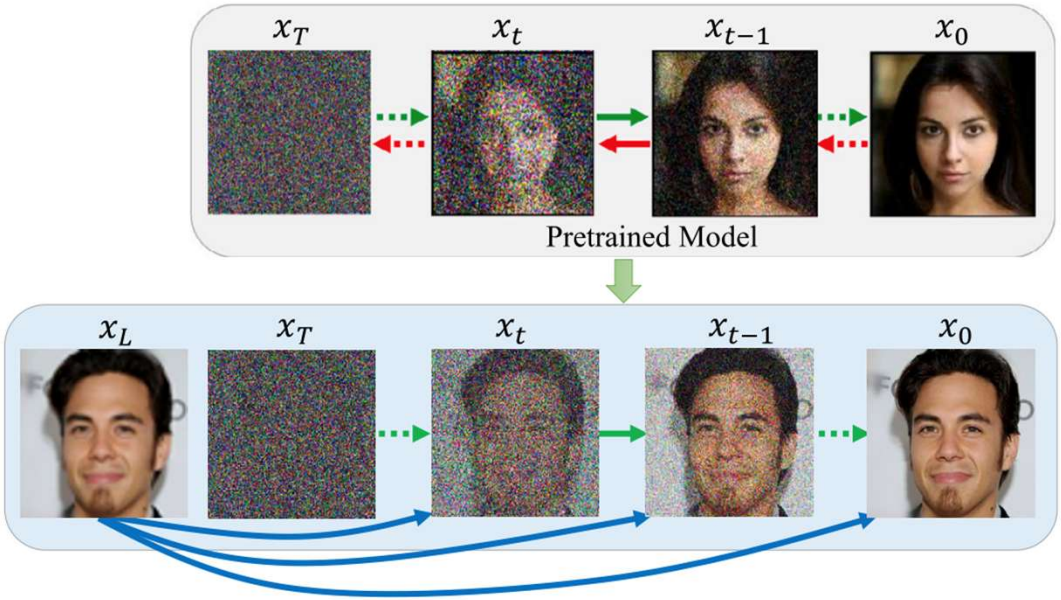


Deep denoising diffusion models



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Score-matching with partial observations



By given partial observations of ground truth – one can perform state recovery

Zero-shot score-matching takes a model that generates arbitrary samples from a distributions and "matches" it to sparse observations. This is just Bayes rule!

Liu, Huan, et al. "When guided diffusion model meets zero-shot image super-resolution." Engineering Applications of Artificial Intelligence 138 (2024): 109336.

The Forward Process

The forward diffusion process systematically corrupts an initial data sample $\mathbf{x}_0$ (drawn from the true data distribution $q(\mathbf{x})$) with noise.

This is described by a forward stochastic differential equation (SDE):

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

The drift term $\mathbf{f}(\mathbf{x}, t)$ governs the deterministic evolution of the data, while the diffusion term $g(t)$ scales the noise added via a Wiener process $d\mathbf{w}$.

The Forward Process

If the drift is zero, often referred to as the Variance Preserving (VP) SDE, the process simplifies to the addition of Gaussian noise with a specific standard deviation σ:

$$\mathbf{x}(\sigma) = \mathbf{x}_0 + \sigma \mathbf{n}$$

where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The noise level σ ranges from $\sigma_{\text{min}} \approx 0$ (no noise) to σ_{max} (pure noise), at which point the data distribution $p(\mathbf{x}(\sigma_{\text{max}}))$ converges to a simple isotropic Gaussian prior. Note that $\sigma(t)$ represents the total accumulated noise variance until time t.

The Reverse Process

The forward SDE we have designed has a corresponding reverse-time SDE. This is

$$d\mathbf{x} = \Big[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\Big] dt + g(t) d\bar{\mathbf{w}},$$

where $d\bar{\mathbf{w}}$ denotes a Wiener process run backwards in time.

This formulation shows that sampling requires knowledge of the **score function**, $\nabla_{\mathbf{x}(\sigma)} \log p(\mathbf{x}(\sigma))$, which points in the direction of increasing data density. For the Gaussian noise model:

$$S(x(\sigma), \sigma) = \nabla_{\mathbf{x}(\sigma)} \log p(\mathbf{x}(\sigma)) = -\frac{\mathbf{E}_{\mathbf{x}_0|\mathbf{x}(\sigma)}[\mathbf{x}(\sigma) - \mathbf{x}_0]}{\sigma^2}$$

The probability flow ODE

An alternative deterministic formulation, known as the probability flow ODE, evolves samples according to:

$$d\mathbf{x} = \left[-\tfrac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt,$$

where we have assumed zero drift. This ODE shares the same marginal distributions as the reverse SDE at all times t, but removes the stochastic diffusion term. With some algebraic manipulation from the original EDM paper $\left(\sqrt{\int_0^t \frac{g(\xi)^2}{s(\xi)^2} \, d\xi} = \sigma(t) \right)$, we have

$$d\mathbf{x} = \left[-\dot{\sigma}(t) \sigma(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}; \sigma(t)) \right] dt,$$

which provides a deterministic mapping between the Gaussian prior at $\sigma_{\max}$ and the data distribution at $\sigma_{\min}$.

The Score

A neural network, D_θ, that estimates the clean data for different noise levels is used to indirectly approximate this score. If the network is trained to predict the clean data, $D_\theta(\mathbf{x}(\sigma);\sigma) \approx \mathbf{x}_0$, its output can be used to estimate the score:

$$\nabla_{\mathbf{x}(\sigma)} \log p(\mathbf{x}(\sigma);\sigma) \approx \frac{D_\theta(\mathbf{x}(\sigma);\sigma) - \mathbf{x}(\sigma)}{\sigma^2} = \mathbf{s}_\theta(x(\sigma),\sigma)$$

If this network is available, generating new samples becomes the matter of sampling $\mathbf{x}(\sigma)$ at the final 'time', assuming a variance schedule $\sigma(t)$ and evolving the reverse ODE.

EDM Formulation

The Elucidating Diffusion Model (EDM, Karras et al., 2022) framework provides a principled design space for diffusion models. Here, the overall denoising model F_θ is defined as:

$$D_\theta(\mathbf{x}(\sigma);\sigma) = c_{\text{skip}}(\sigma)\mathbf{x}(\sigma) + c_{\text{out}}(\sigma)F_\theta\left(\frac{\mathbf{x}(\sigma)}{c_{\text{in}}(\sigma)}; c_{\text{noise}}(\sigma)\right)$$

where F_θ is the neural network parameterized by θ. The scaling factors $(c_{\text{skip}}, c_{\text{out}}, c_{\text{in}}, c_{\text{noise}})$ are crucial for ensuring the network operates on inputs with approximately unit variance. The training objective is a weighted mean squared error:

$$L(\theta) = \mathbf{E}_{\mathbf{x}_0,\mathbf{n},\sigma}\left[\lambda(\sigma)||\mathbf{x}_0 - D_\theta(\mathbf{x}_0 + \sigma\mathbf{n};\sigma)||^2\right]$$

where σ is sampled from a distribution $p(\sigma)$ and $\lambda(\sigma)$ is a weighting function, typically $\lambda(\sigma) = 1/c_{\text{out}}(\sigma)^2$, that balances the loss contribution across different noise levels.

EDM Formulation

Once trained, the EDM model generates samples by numerically solving the probability flow ODE,

$$\frac{d\mathbf{x}}{dt} = \frac{D_\theta(\mathbf{x}(t); t) - \mathbf{x}(t)}{t}$$

Starting from a noise sample $\mathbf{x}_{t_{max}} \sim \mathcal{N}(\mathbf{0}, t_{max}^2 \mathbf{I})$, an ODE solver iteratively refines the sample across a sequence of decreasing noise levels t_i. For example, a first-order Euler step from $\mathbf{x}(t_i)$ to $\mathbf{x}(t_{i+1})$ is:

1. Predict denoised data: $\hat{\mathbf{x}}_0 = D_\theta(\mathbf{x}(t_i); t_i)$.
2. Calculate gradient: $\mathbf{d} = (\mathbf{x}(t_i) - \hat{\mathbf{x}}_0)/t_i$.
3. Take step: $\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \mathbf{d}(t_{i+1} - t_i)$.

More complicated integration schemes are possible. The noise schedule is set to $\sigma(t) = t$.

Diffusion posterior sampling

From Bayes' theorem, the posterior score is the sum of the prior and likelihood scores:

$$\nabla_{\mathbf{x}(t)} \log p_\theta(\mathbf{x}(t)|\mathbf{y}) = \nabla_{\mathbf{x}(t)} \log p_\theta(\mathbf{x}(t)) + \nabla_{\mathbf{x}(t)} \log p(\mathbf{y}|\mathbf{x}(t))$$

For a differentiable measurement model M and Gaussian observation noise with covariance Σ_y, DPS approximates the likelihood $p(\mathbf{y}|\mathbf{x}(t))$ by first estimating the clean data $\hat{\mathbf{x}}_0(\mathbf{x}(t), t)$ from the noisy sample $\mathbf{x}(t)$ and then evaluating the likelihood at this estimate:

$$p(\mathbf{y}|\mathbf{x}(t)) \approx \mathcal{N}(\mathbf{y}|M(\hat{\mathbf{x}}_0(\mathbf{x}(t))), \Sigma_y)$$

To bypass Monte-Carlo, $\hat{\mathbf{x}}_0$ is obtained via Tweedie's formula, which connects the posterior mean to the score of the noisy data distribution:

$$\hat{\mathbf{x}}_0(\mathbf{x}(t)) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}(t)] \approx \mathbf{x}(t) + \sigma(t)^2 \mathbf{s}_\theta(\mathbf{x}(t), t)$$

Score-based data assimilation

Score-based Data Assimilation (SDA) (Rozet et al., 2023) refines this by accounting for the variance of the estimate $\hat{\mathbf{x}}_0$, yielding a more stable likelihood:

$$p(\mathbf{y}|\mathbf{x}(t)) \approx \mathcal{N}\left(\mathbf{y} \mid M(\hat{\mathbf{x}}_0), \Sigma_y + \sigma(t)^2 \mathbf{J}_M(\hat{\mathbf{x}}_0)\mathbf{\Gamma}\mathbf{J}_M(\hat{\mathbf{x}}_0)^T\right)$$

where $\mathbf{J}_M$ is the Jacobian of the measurement function and $\mathbf{\Gamma}$ is the term that depends on the eigen-decomposition of the covariance of prior $p(x)$, given by Σ_x.

In this work, to simplify the approximation, the term $\mathbf{J}_M\mathbf{\Gamma}\mathbf{J}_M^T$ is replaced by a constant (diagonal) matrix.

SDA with EDM

EDM gives a direct and intuitive estimate for the clean data:

$$\hat{\mathbf{x}}_0 = D_\theta(\mathbf{x}(t); \sigma(t))$$

The likelihood becomes:

$$\mathbf{s}_l(\mathbf{x}(t), \sigma(t); \mathbf{y}) = \nabla_{\mathbf{x}(t)} \log p(\mathbf{y}|\mathbf{x}(t)) \propto \nabla_{\mathbf{x}(t)} \frac{-||\mathbf{y} - M(D_\theta(\mathbf{x}(t); \sigma(t)))||^2}{\Sigma_y + \sigma(t)^2 \hat{\Gamma}}$$

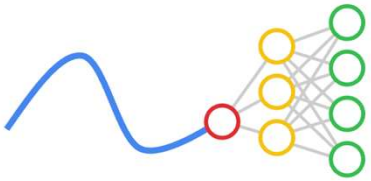
where Σ_y is the variance of observation noise and $\hat{\Gamma}$ approximates $\mathbf{J}_M \boldsymbol{\Gamma} \mathbf{J}_M^T$.
Again, an Euler-based sampling becomes:

1. **Predict clean data:** $\hat{\mathbf{x}}_0 = D_\theta(\mathbf{x}_{t_i}; t_i)$.
2. **Calculate prior gradient:** $\mathbf{d}_{\text{prior}} = (\mathbf{x}_{t_i} - \hat{\mathbf{x}}_0)/t_i$.
3. **Evaluate likelihood gradient:** $\mathbf{s}_l(\mathbf{x}_{t_i}, t_i; \mathbf{y})$.
4. **Calculate posterior gradient:** $\mathbf{d}_{\text{pos}} = \mathbf{d}_{\text{prior}} - \lambda_g \cdot t_i \cdot \mathbf{s}_l$, where λ_g is a guidance scale and t_i is multiplied to scale the drift term of the probability flow ODE.
5. **Perform sampling step:** $\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \mathbf{d}_{\text{pos}} \cdot (t_{i+1} - t_i)$ and iterate.

Problem: A reanalysis generator

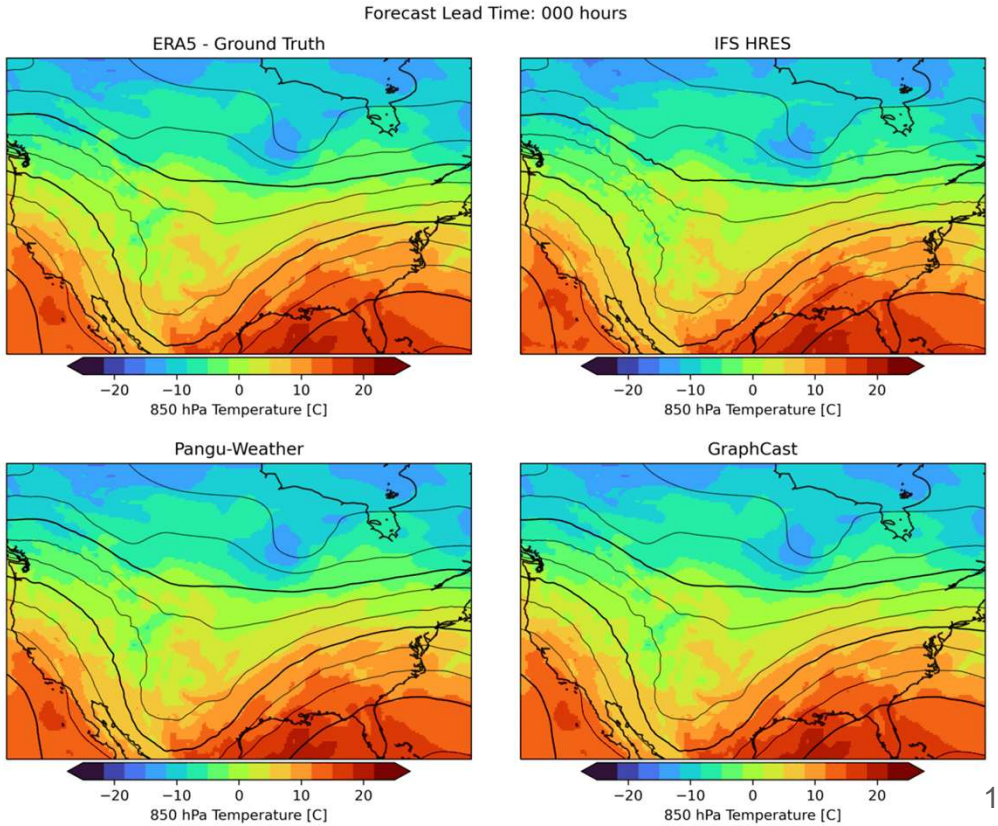
Our data is given by ERA5. For this work, we use the 1.40525-degree (128 × 256 grid points) ERA5 dataset from WeatherBench (1980 to 2015 for training, 2020 for the results). Bilinear interpolation is used to regrid from the native resolution.

WeatherBench 2



CI passing | Lint passing | docs passing | Open in Colab

WeatherBench 2 - A benchmark for the next generation of data-driven global weather models



Problem: A reanalysis generator

Most deep learning models learn from reanalysis data and build a forecaster. However, operational forecasting is much more interested in model and data fusion, i.e., fusing observations and numerical models for *getting the reanalysis at timestep 0*!

Article | Open access | Published: 20 March 2025

End-to-end data-driven weather prediction

Anna Allen, Stratis Markou, Will Tebbutt, James Requeima, Wessel P. Bruinsma, Tom R. Andersson, Michael Herzog, Nicholas D. Lane, Matthew Chantry, J. Scott Hosking & Richard E. Turner

Nature **641**, 1172–1179 (2025) | Cite this article

Position in literature

Several groups are looking into the use of diffusion models for atmospheric data assimilation.

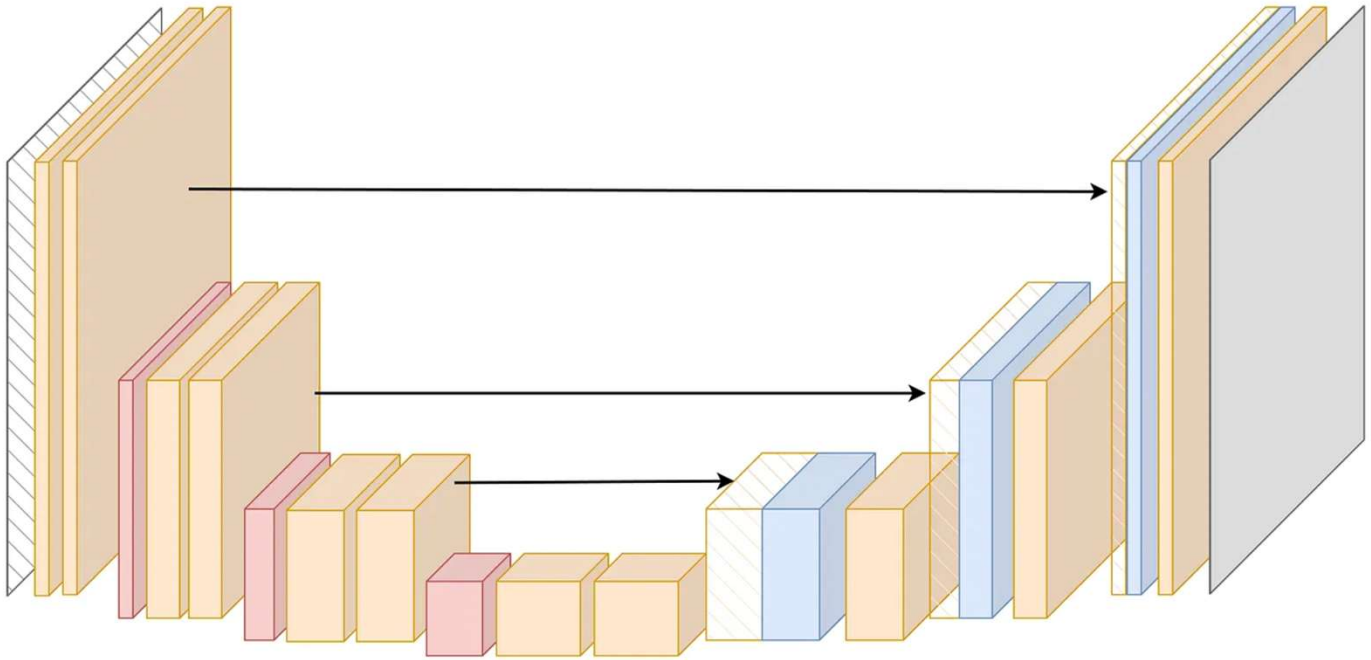
1. Huang, Langwen, Lukas Gianinazzi, Yuejiang Yu, Peter D. Dueben, and Torsten Hoefler. "Diffda: a diffusion model for weather-scale data assimilation." arXiv preprint arXiv:2401.05932 (2024). **Uses a conditional diffusion model (i.e., observations and a numerical emulator prediction go into the model before a prediction of reanalysis may be made).**
2. Andry, Gérôme, François Rozet, Sacha Lewin, Omer Rochman, Victor Mangeleer, Matthias Pirlet, Elise Faulx, Marilaure Grégoire, and Gilles Louppe. "Appa: Bending weather dynamics with latent diffusion models for global data assimilation." arXiv preprint arXiv:2504.18720 (2025). **Similar to our approach but at much higher resolution, and demonstrated success on satellite radiances. High-frequency capture is limited due to compression in the diffusion model.**

Many studies frame reanalysis prediction as a single end-to-end mapping from observations (i.e., they ditch Bayes rule). Examples are MetNet3, Aardvark Weather, GraphDOP.

Others use established EnKF, 3D/4D Var techniques where the AI emulator replaces the NWP model. Examples are FuXi-DA, Fengwu-4DVar, 4DVarNet – Note: *Use of NN surrogate adjoints are controversial (https://arxiv.org/pdf/2411.14677)*

Step 1: Score-network

Our first step is to build a diffusion model that simply generates “random” reanalysis data through sampling (i.e, denoising samples from a Gaussian). We parameterize our score learner using a U-Net architecture with cyclic convolutions.



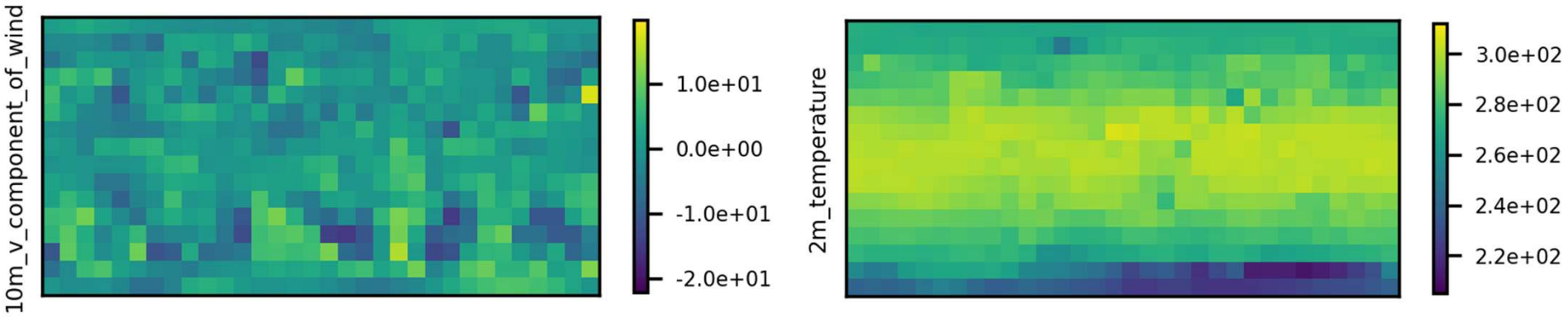
Diffusion posterior sampling scenarios

Using ERA5/Weatherbench – we can train an EDM-based sampler that is generates random snapshots of the atmosphere.

After this training is complete, we can set up various experimental scenarios for our diffusion posterior sampling:

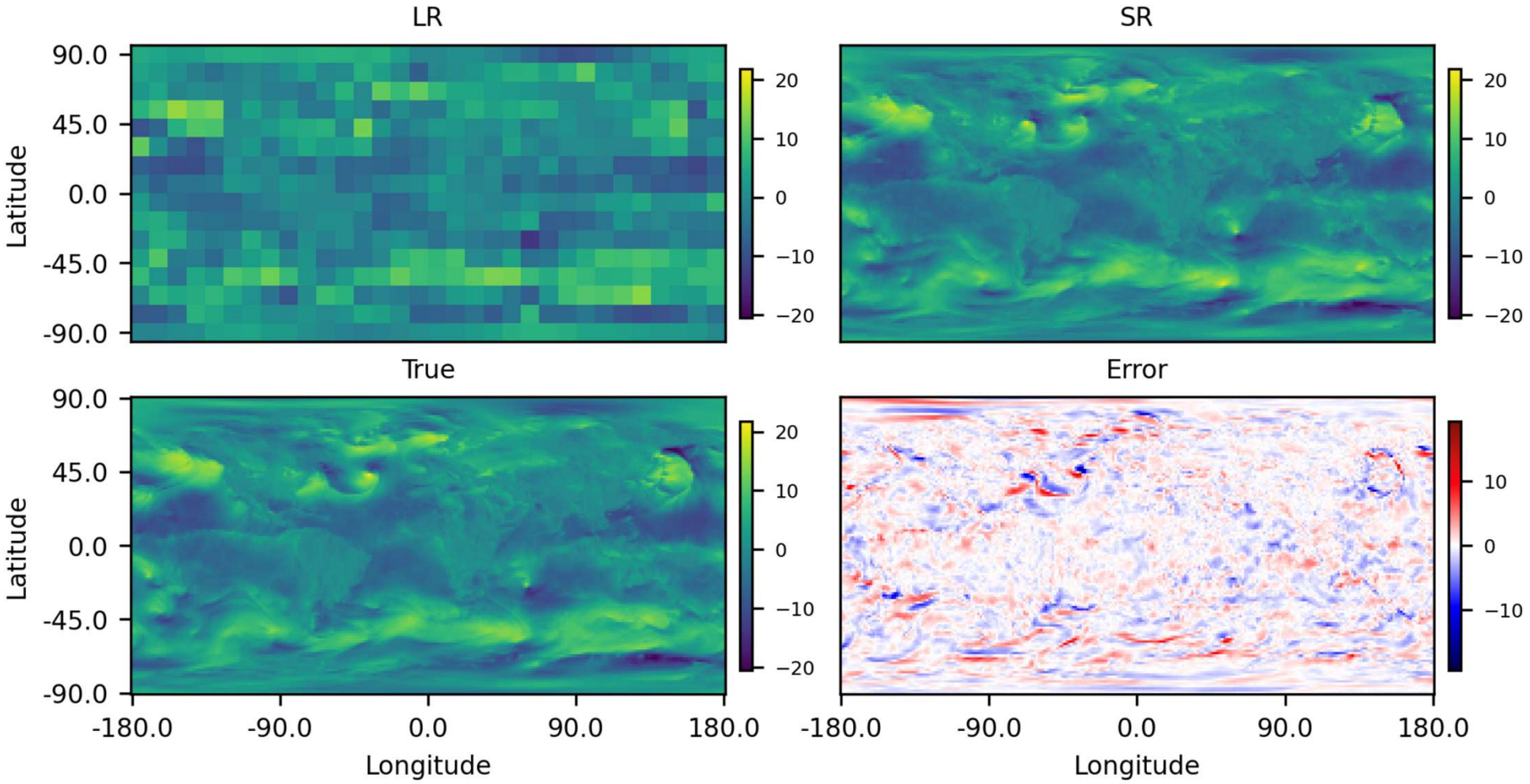
1. Observing ERA5 on a coarse-grid itself (a super-resolution/downscaling task)
2. Observing radiosonde information on sparse and unstructured grids (multimodal super-resolution).
3. Observing numerical weather prediction models (a framework that converts a numerical model to an observable!)

Downscaling via diffusion



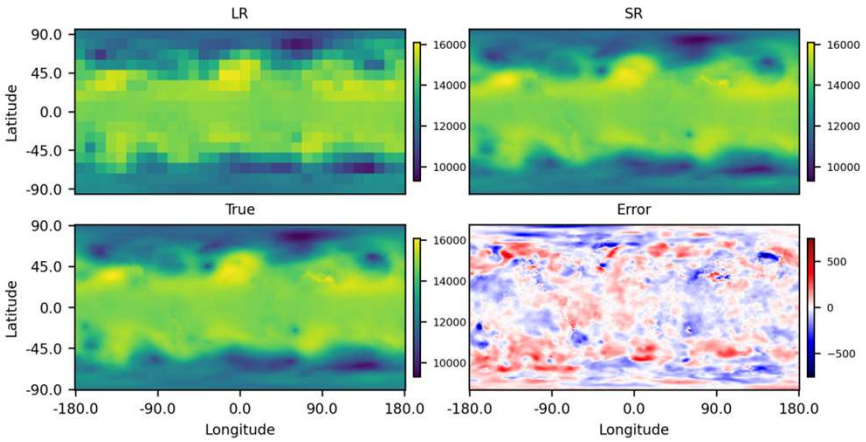
We show our pretrained diffusion model ERA5 on a 32x64 resolution (69 variables)

Downscaling via diffusion

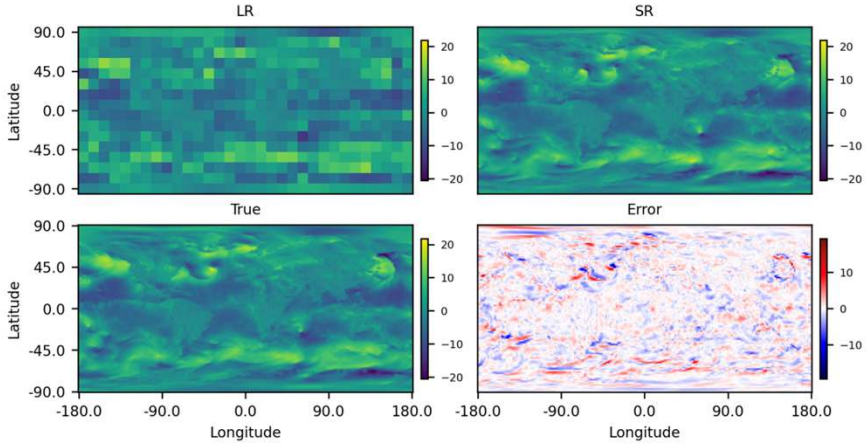


Reconstructions via score-matching based super-resolution

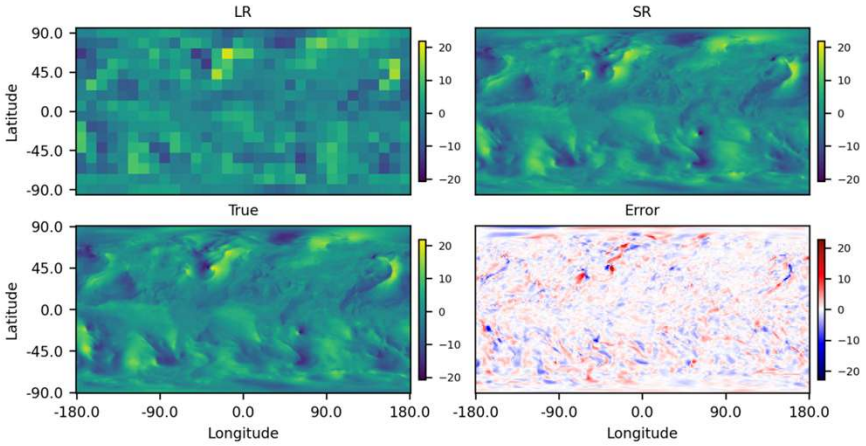
Downscaling via diffusion



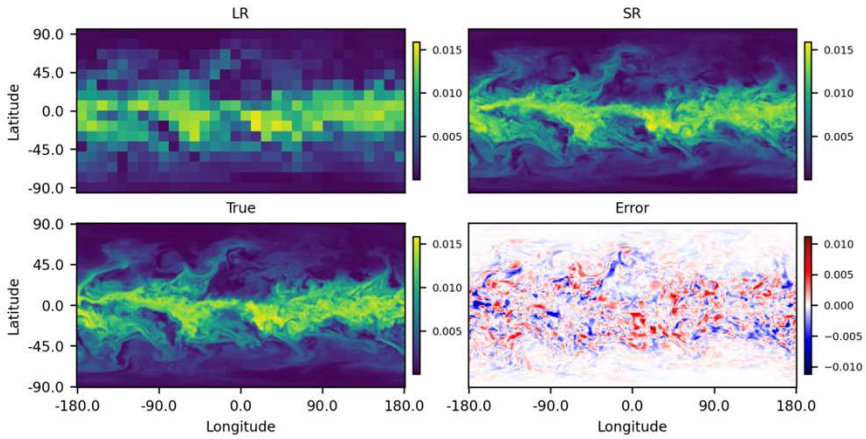
(a) 500mb Geopotential (m^2/s^2)



(b) 10m U-Wind (m/s)

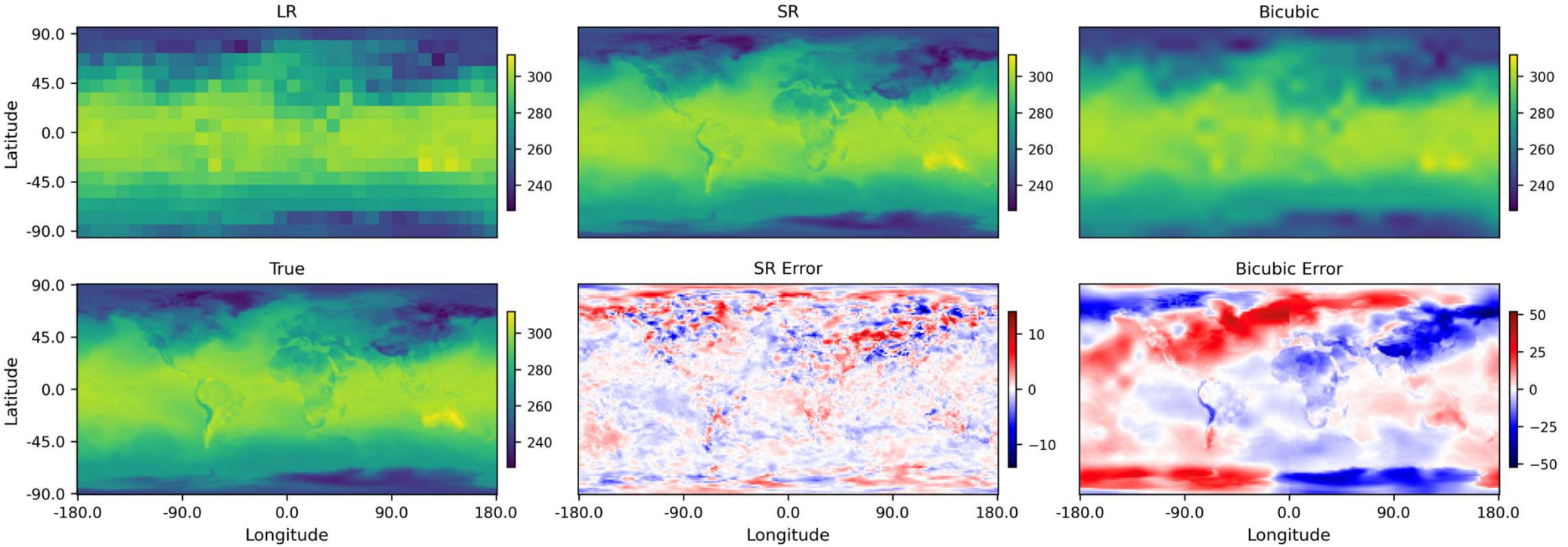


(c) 10m V-Wind (m/s)



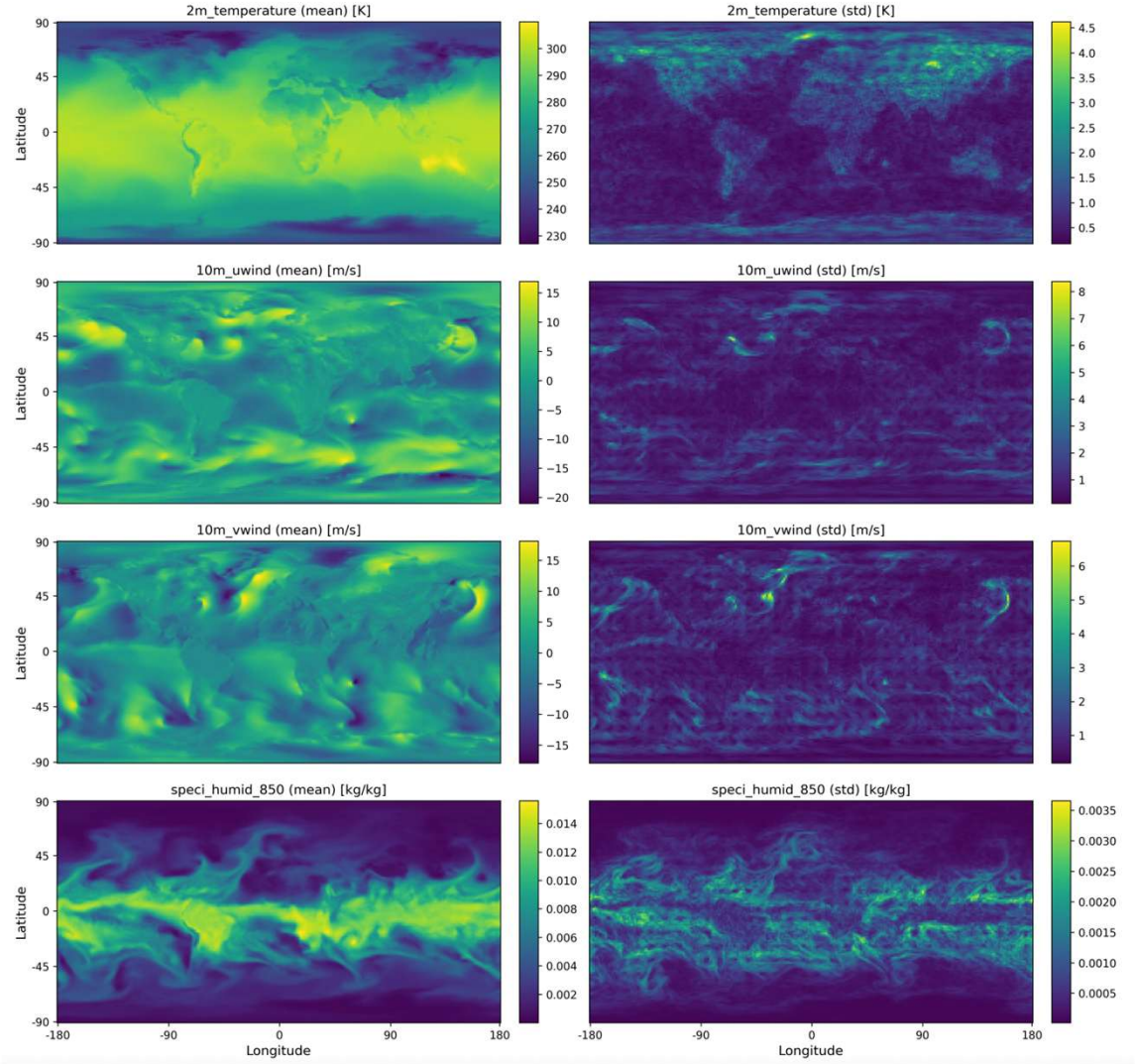
(d) 850mb Specific humidity (kg/kg)

Downscaling via diffusion



Comparison with bicubic interpolation (note scale)

Downscaling via diffusion

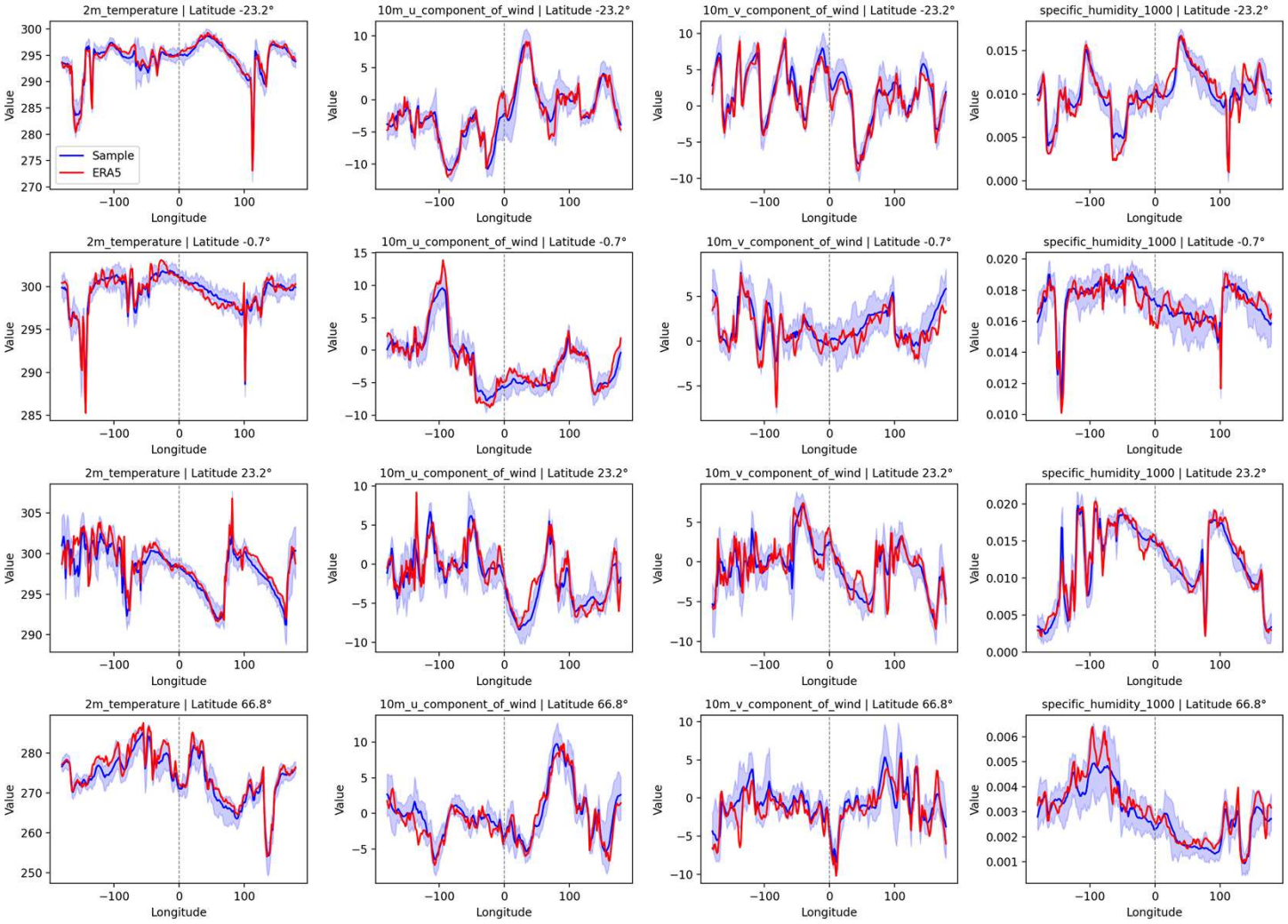


Estimates of uncertainty. Notice uniform "pattern" of low variance.

A question you might have at this moment – how do you decide the likelihood covariances? Answer: Validation data (surface temperature based)

Σ_y	$\hat{\Gamma}$	2m Temp RMSE (Mean $\pm$ Std)
1×10^{-1}	1×10^{-2}	2.3511 ± 0.1246
1×10^{-1}	1×10^{-3}	1.7273 ± 0.0529
1×10^{-1}	5×10^{-3}	$\mathbf{1.6883 \pm 0.0224}$
5×10^{-1}	1×10^{-3}	2.3673 ± 0.0834
5×10^{-1}	5×10^{-3}	2.3359 ± 0.1129
5×10^{-2}	5×10^{-3}	54.4904 ± 7.0866

Downscaling via diffusion

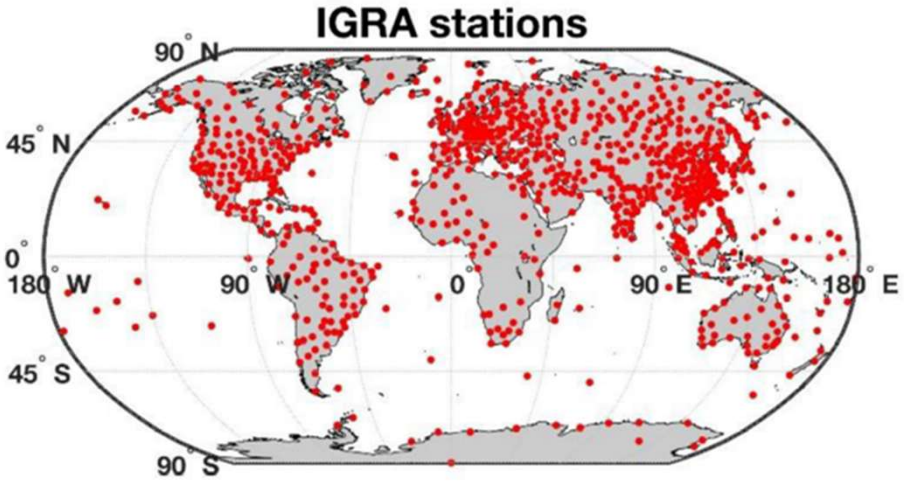


Ensembles for uncertainty estimates from diffusion-based sampling

Observing the IGRA radiosonde data

We use radiosonde observations from the NOAA Integrated Radiosonde Archive (IGRA) version 2.2. This is a much smaller set of observations than used for making ERA5.

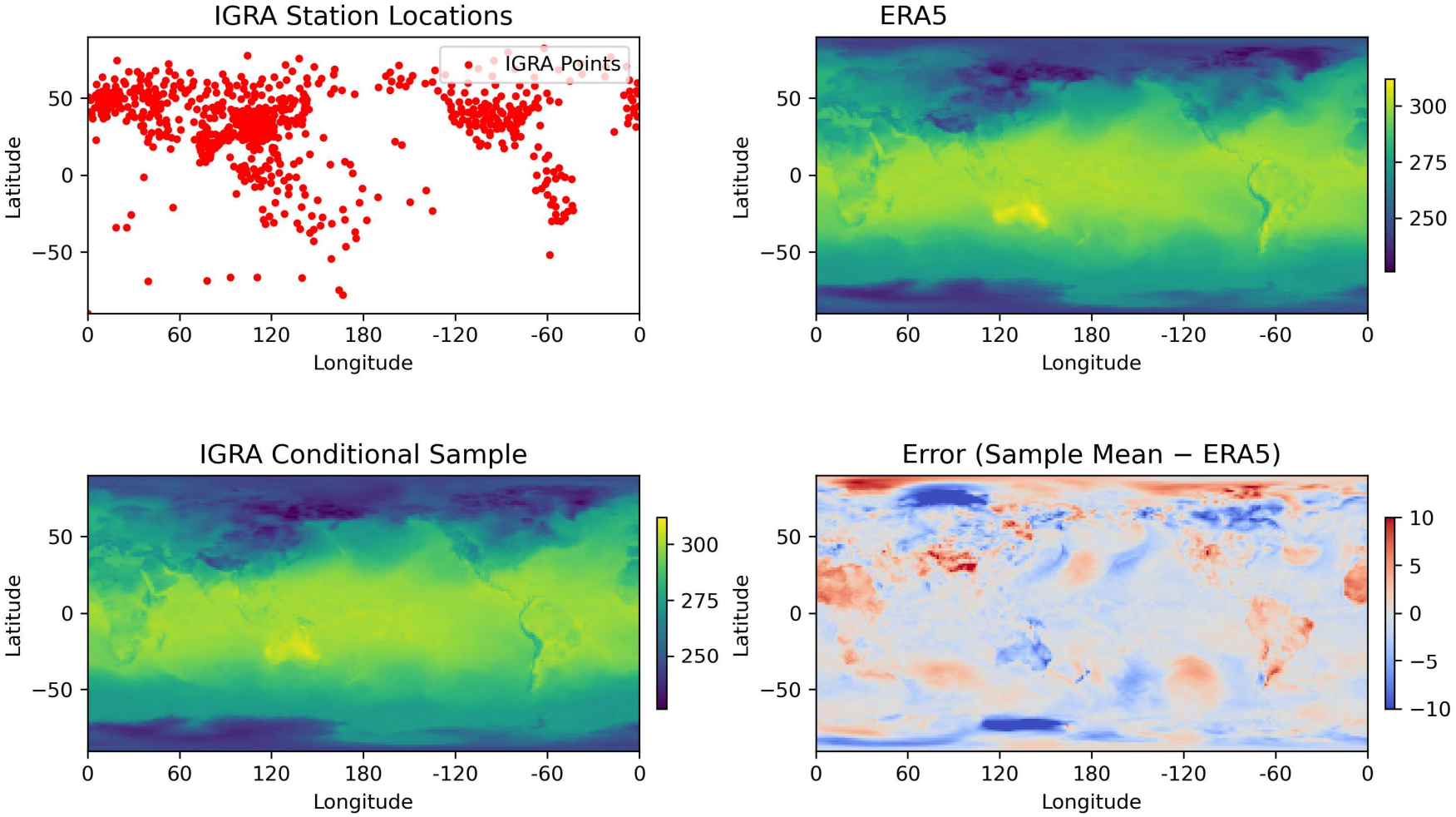
"The Integrated Global Radiosonde Archive (IGRA) consists of radiosonde and pilot balloon observations from more than 2,800 globally distributed stations. The earliest data date back to 1905, and recent data become available in near real time from about 800 stations worldwide."



harmonics

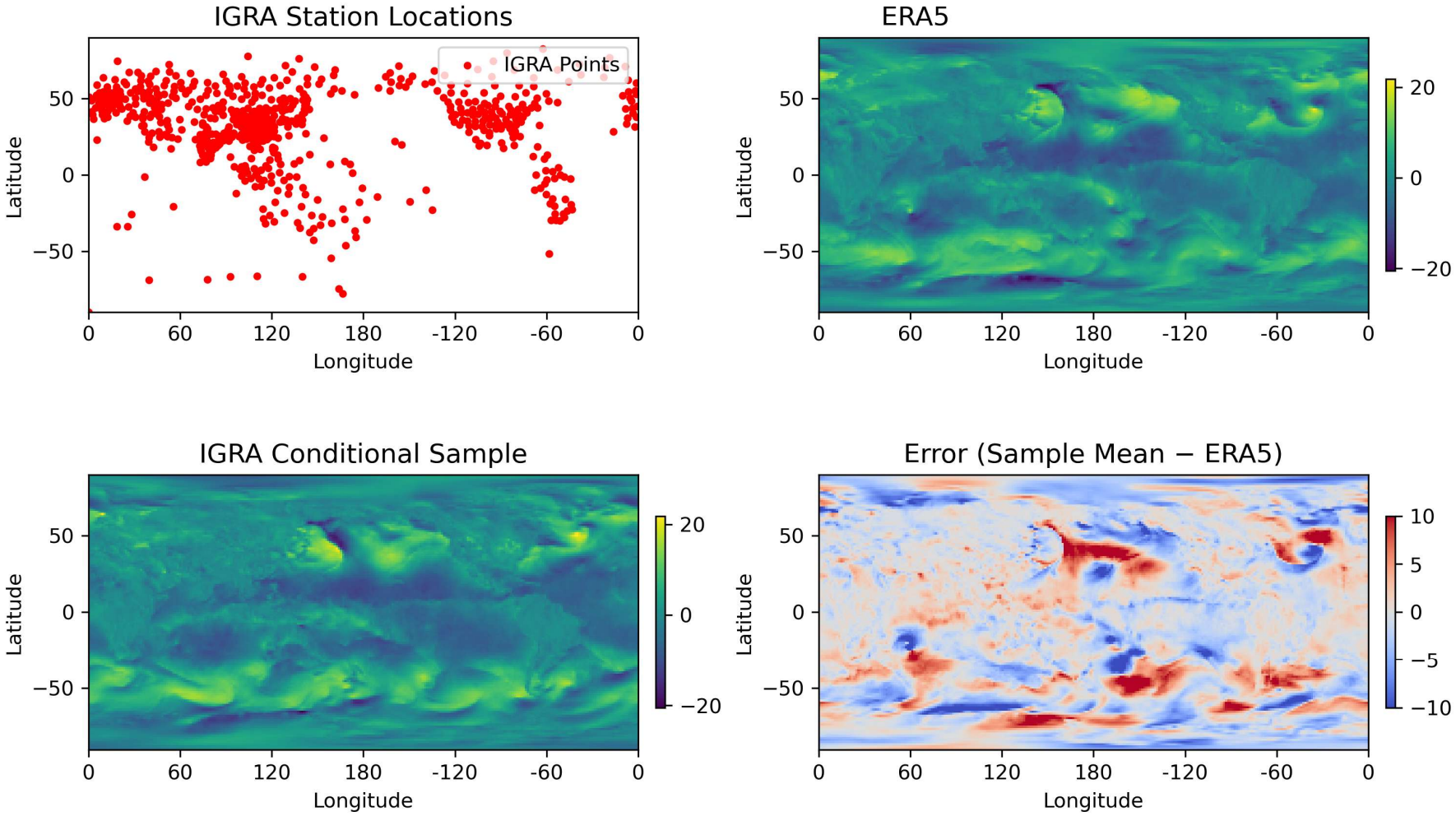
https://www.ncei.noaa.gov/products/weather-balloon/integrated-global-radiosonde-archive

Observing the IGRA radiosonde data



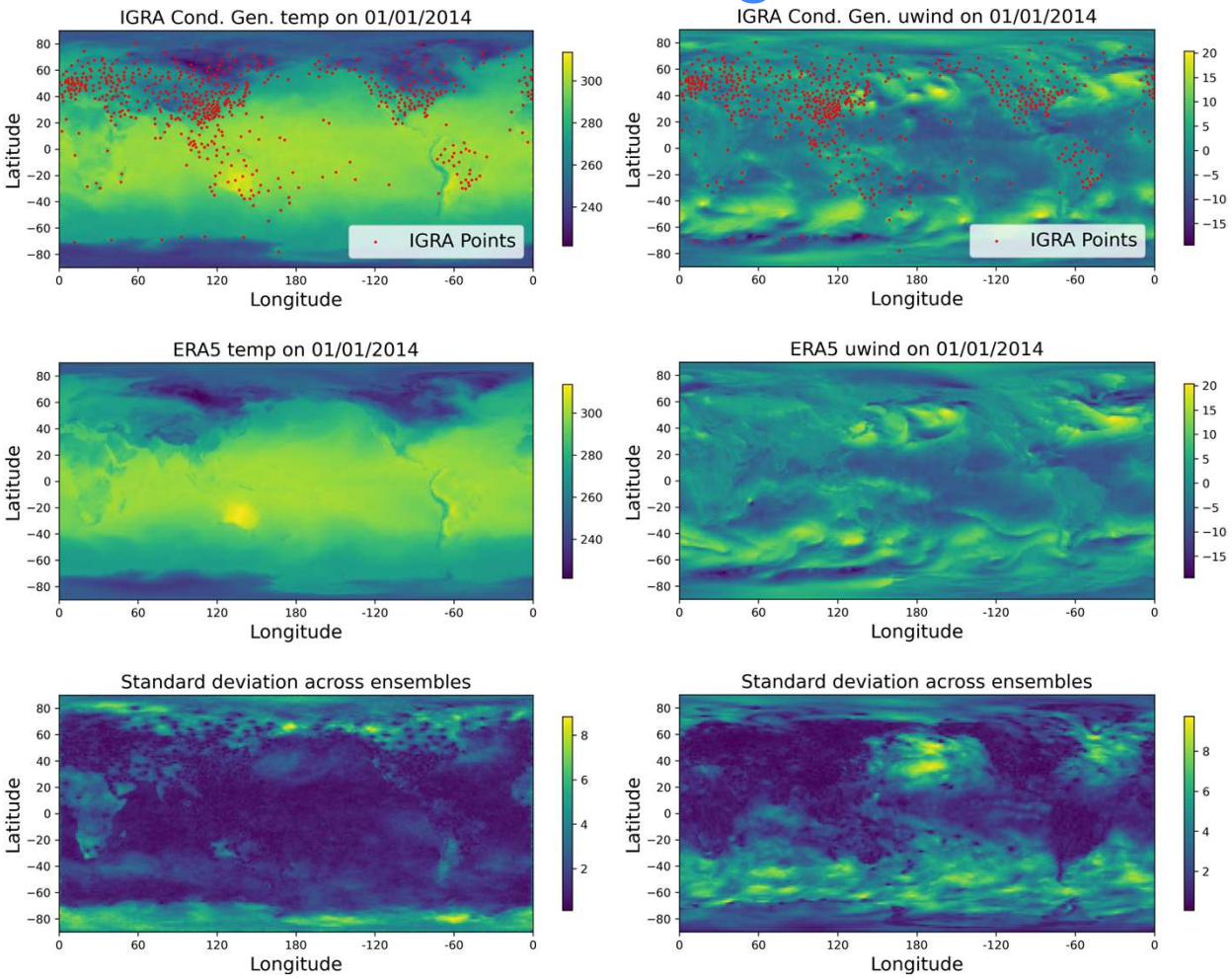
Temperature reconstruction using IGRA observations

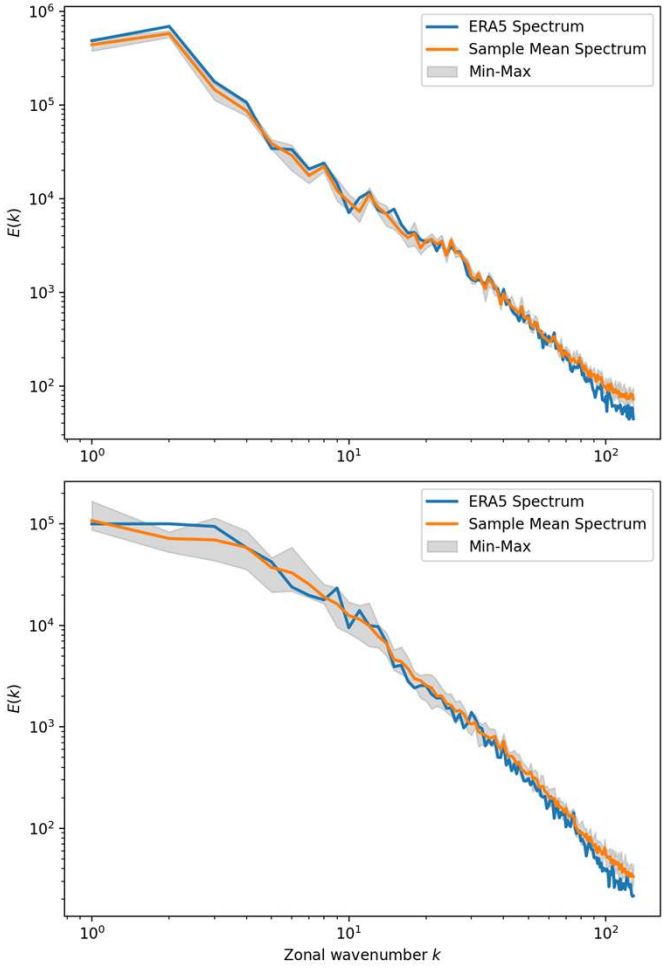
Observing the IGRA radiosonde data



10m U-Wind speed reconstruction using IGRA observations

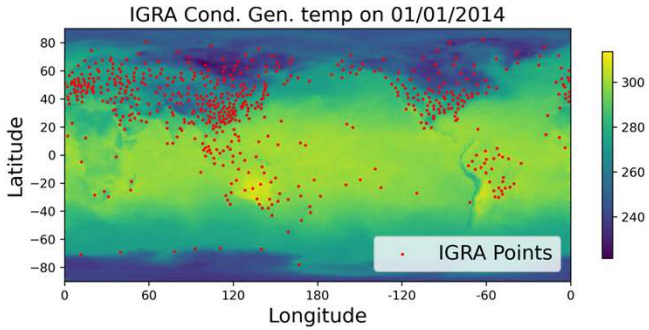
Observing the IGRA radiosonde data

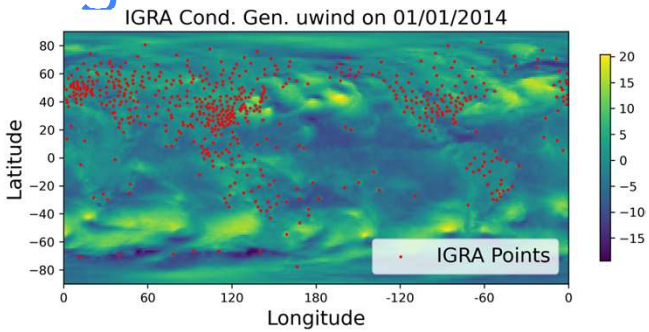


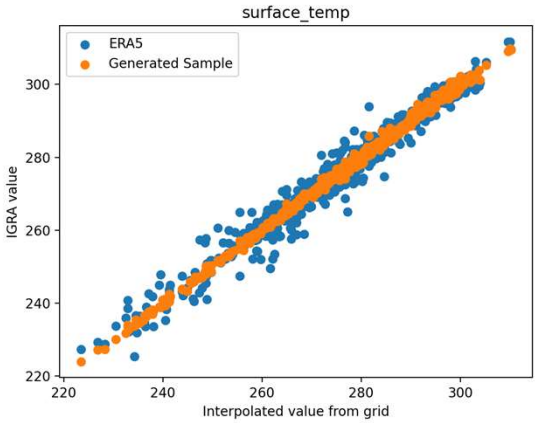


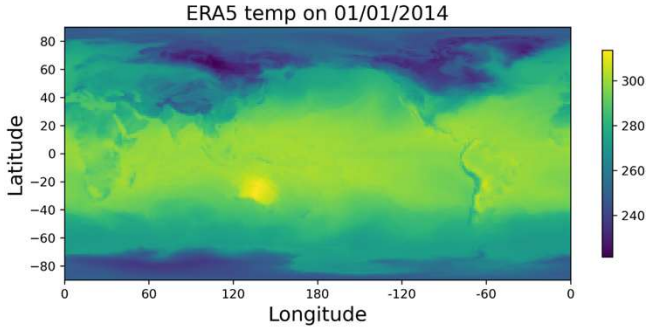
Reconstruction using IGRA observations

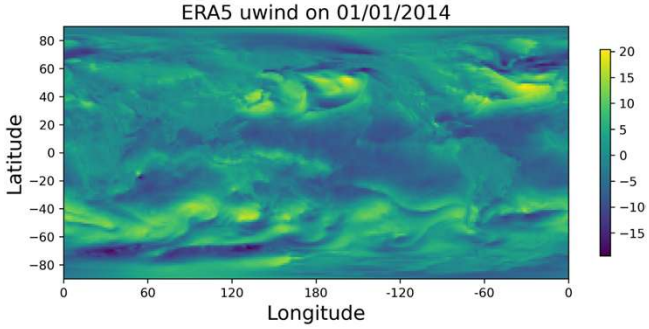
Observing the IGRA radiosonde data

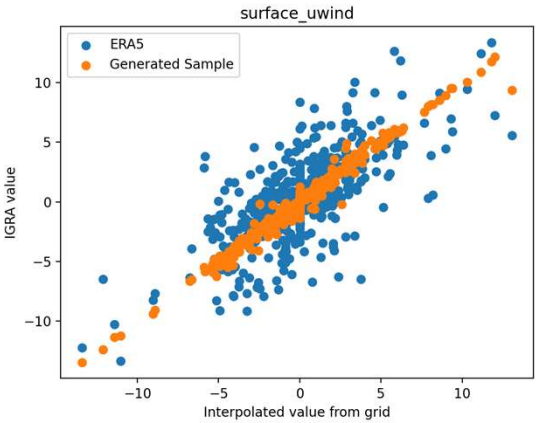


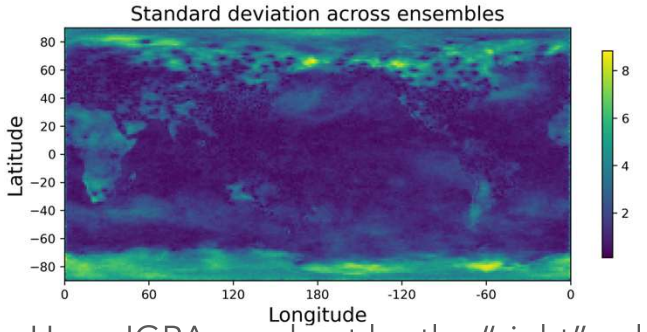


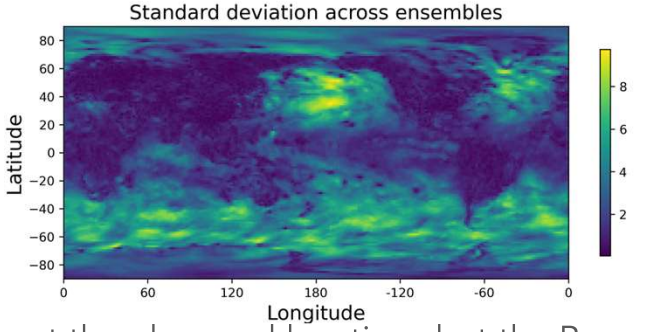






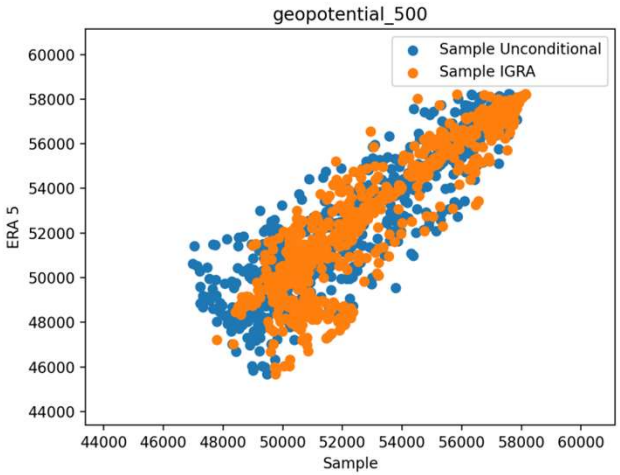




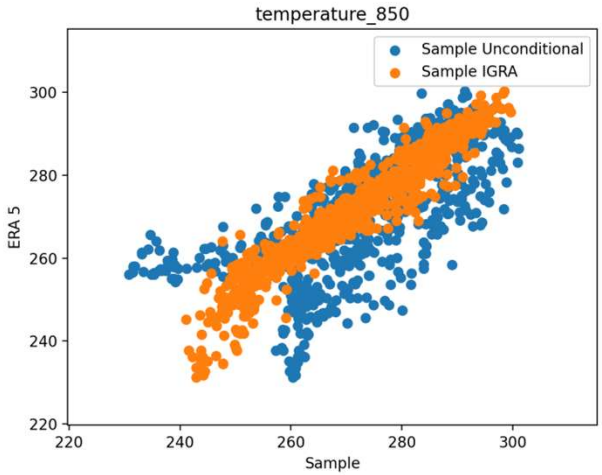


Here, IGRA need not be the “right” value at the observed locations but the Bayesian approach pushes the generated samples in that direction. Classical problems of likelihood/covariance are still present.

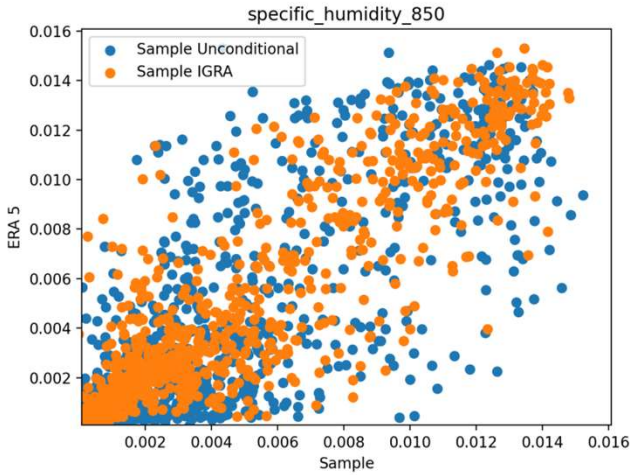
Observing the IGRA radiosonde data



(a) Geopotential 500



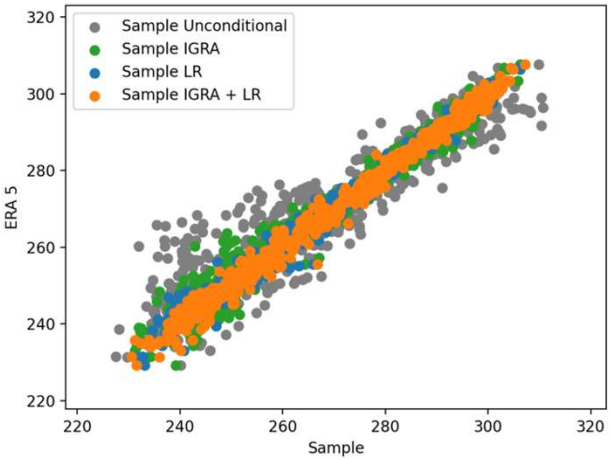
(b) Temperature 850



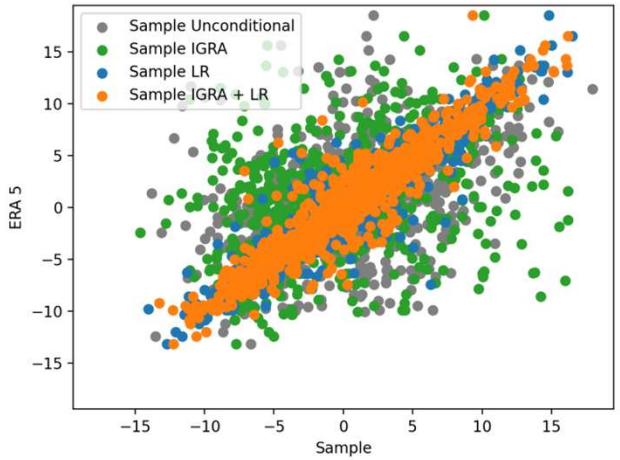
(c) Humidity 850

For several variables, IGRA leads to an improved recovery of reanalysis (at non-IGRA locations) in comparison with the "random reanalysis generator". However, this is not true for *all* variables.

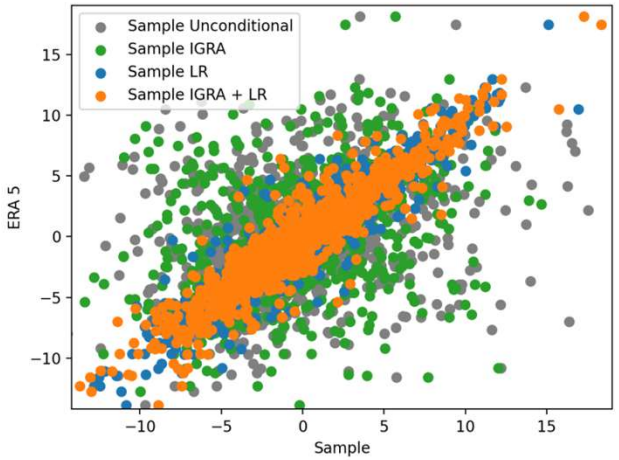
Observing IGRA and ERA5-Low Res



(a) Surface Temperature



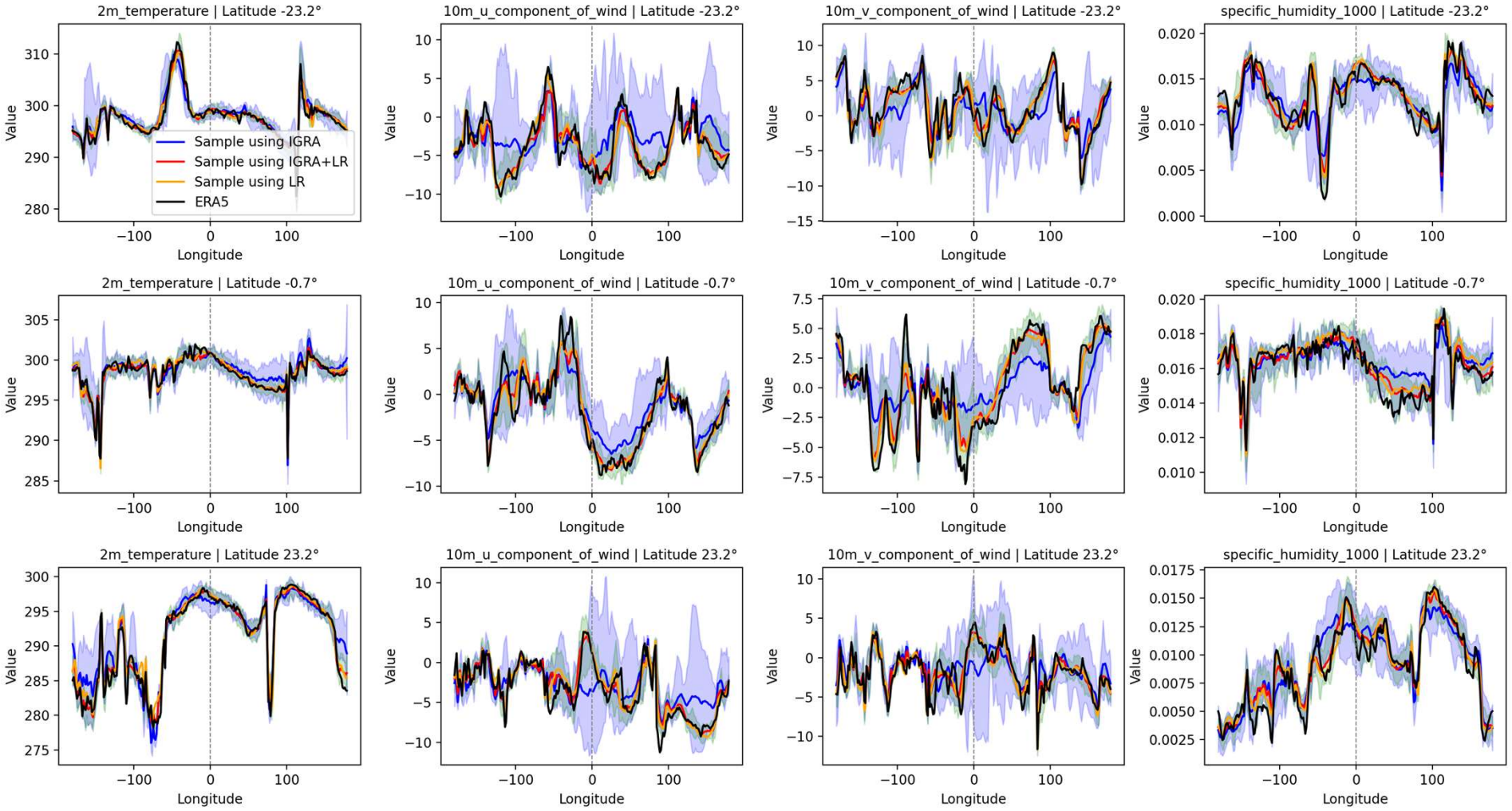
(b) U-Wind



(c) V-Wind

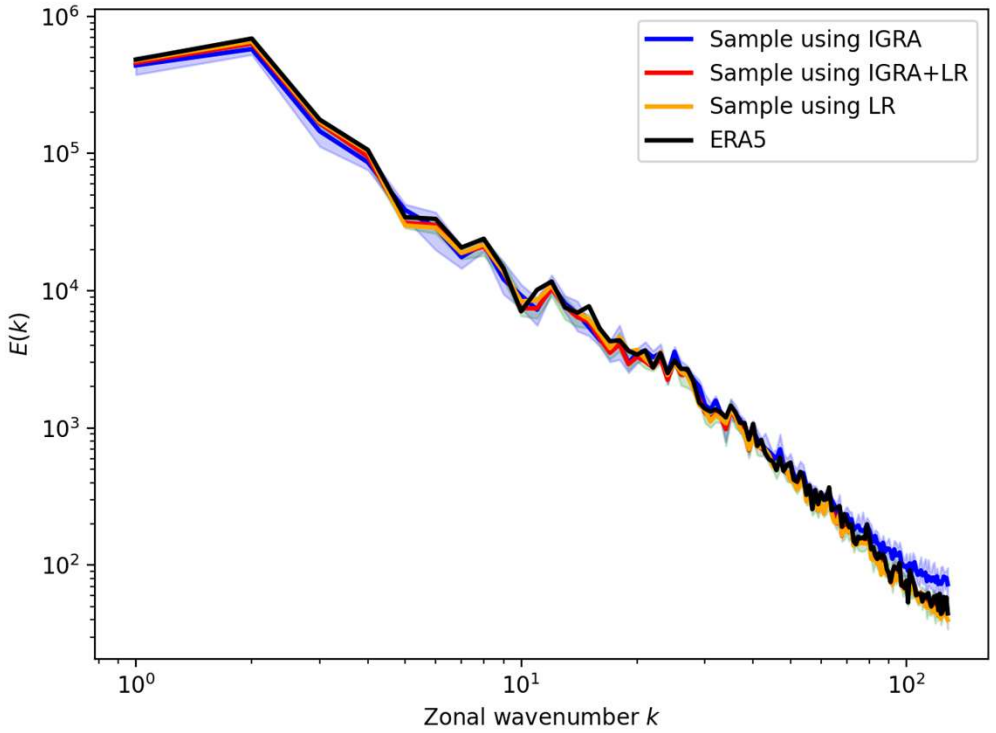
What if we observe *both* IGRA and ERA5 Low Res (a completely fictional situation). Here at non-IGRA locations. Low-res info improves reconstructions.

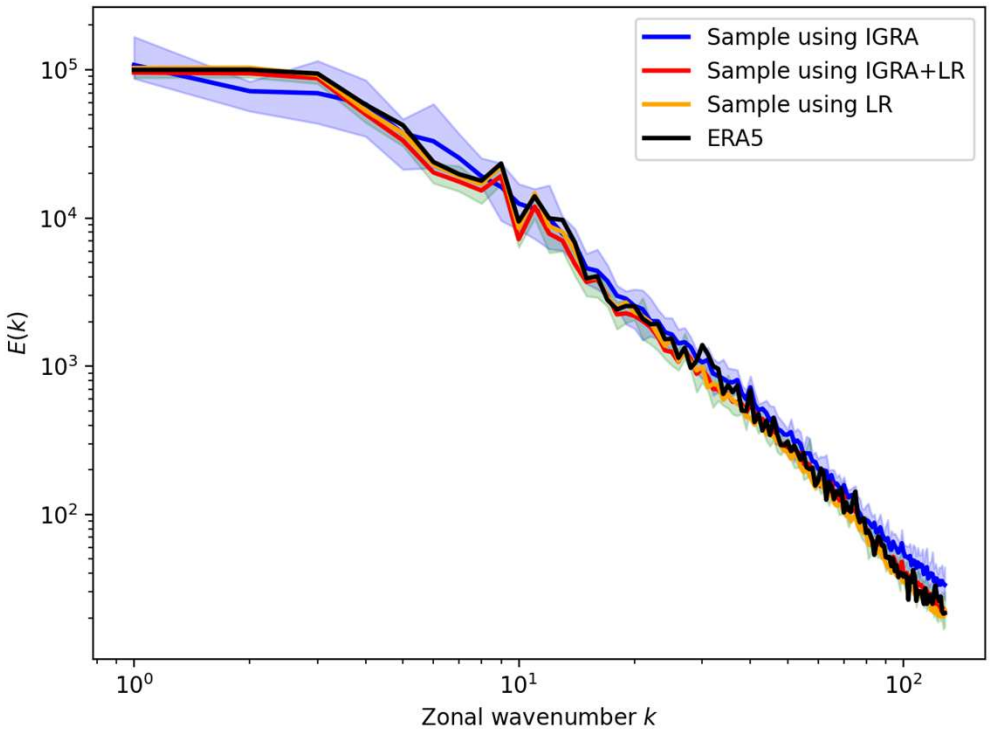
Observing IGRA and ERA5-Low Res



What if we observe *both* IGRA and ERA5 Low Res (a completely fictional situation).

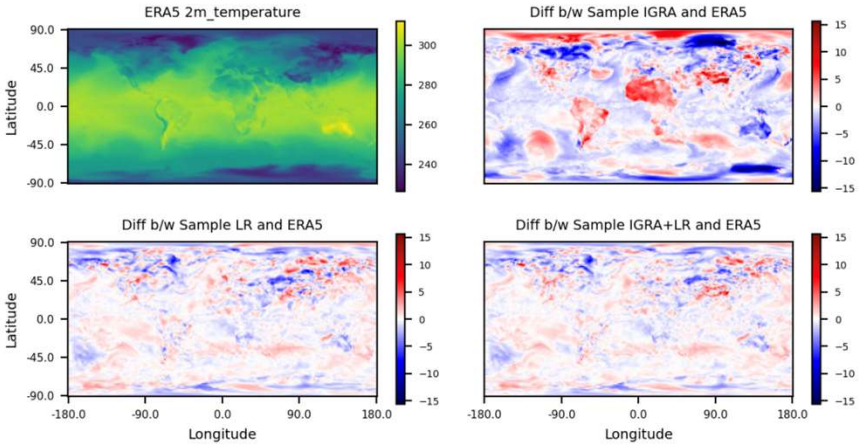
Observing IGRA and ERA5-Low Res



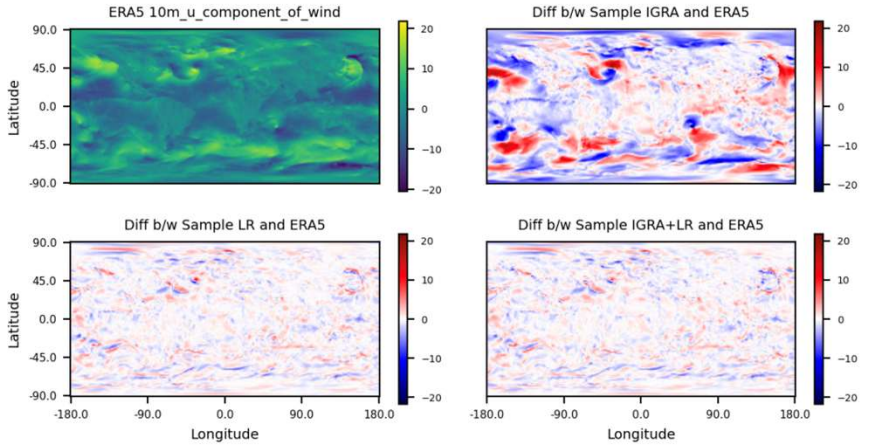


What if we observe *both* IGRA and ERA5 Low Res (a completely fictional situation).
Note spectral pileup at grid cut-off because of IGRA observations.

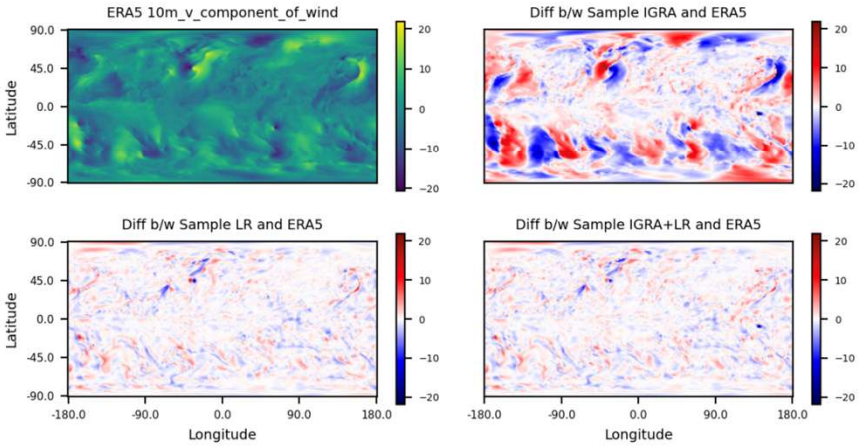
Observing IGRA and ERA5-Low Res



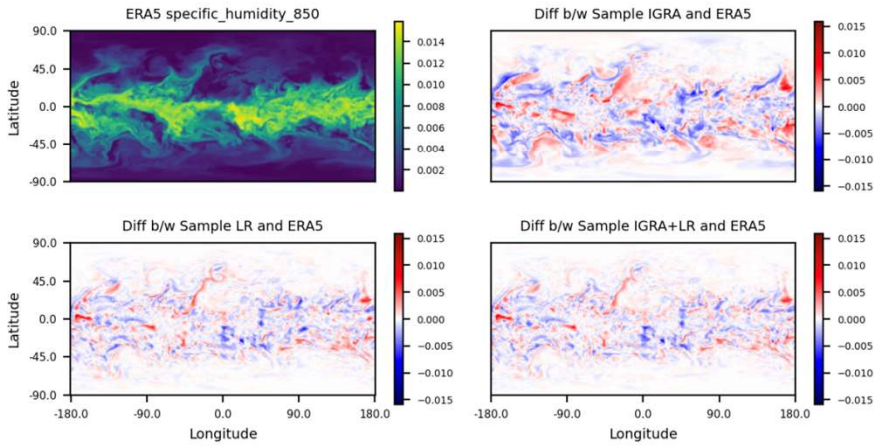
(a) 2m Temperature (K)



(b) U-Wind (m/s)

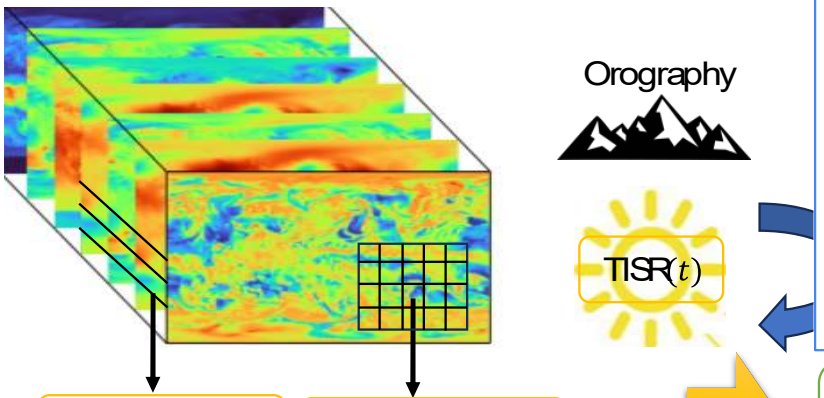


(c) V-Wind (m/s)



(d) Specific humidity (kg/kg)

Observing an AI emulator: LUCIE

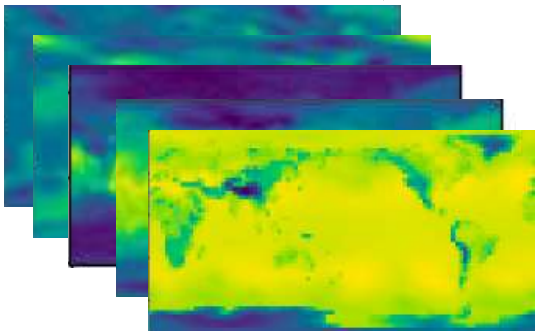


Orography

- Low-resolution, minimal HPC infrastructure
- 2.4h of training on a single A100 GPU
- O(1000) ensembles of climate emulation
- As low as 2 years of training data

Regridding on model ! levels

Regridding on T30 Gaussian grid

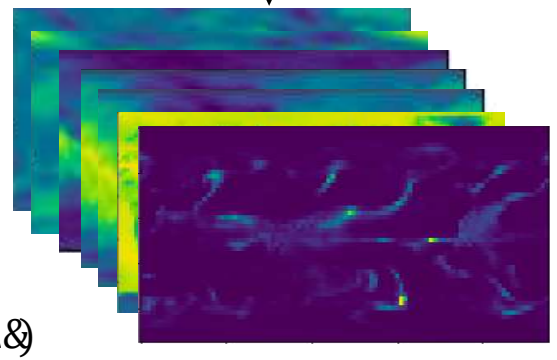


$U_{!.\#\$}$
$V_{!.\#\$}$
$SH_{!.\%\#}$
$T_{!.\%\#}$
$LogP$

$\$(\&)$

SFNO ●●● SFNO

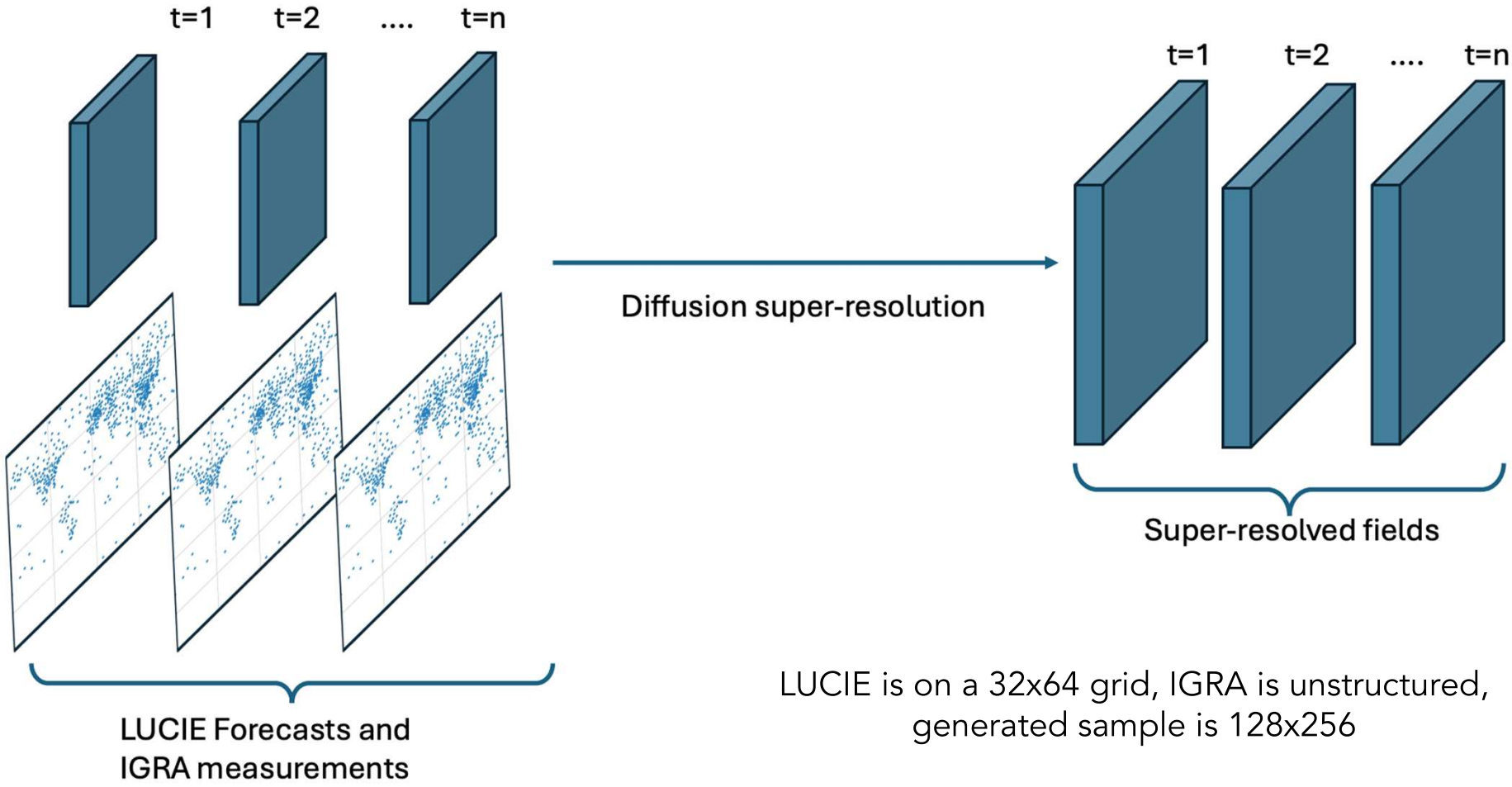
$U_{!.\#\$}$
$V_{!.\#\$}$
$SH_{!.\%\#}$
$T_{!.\%\#}$
$LogP$
TP



$\$(\&+\Delta\&)$

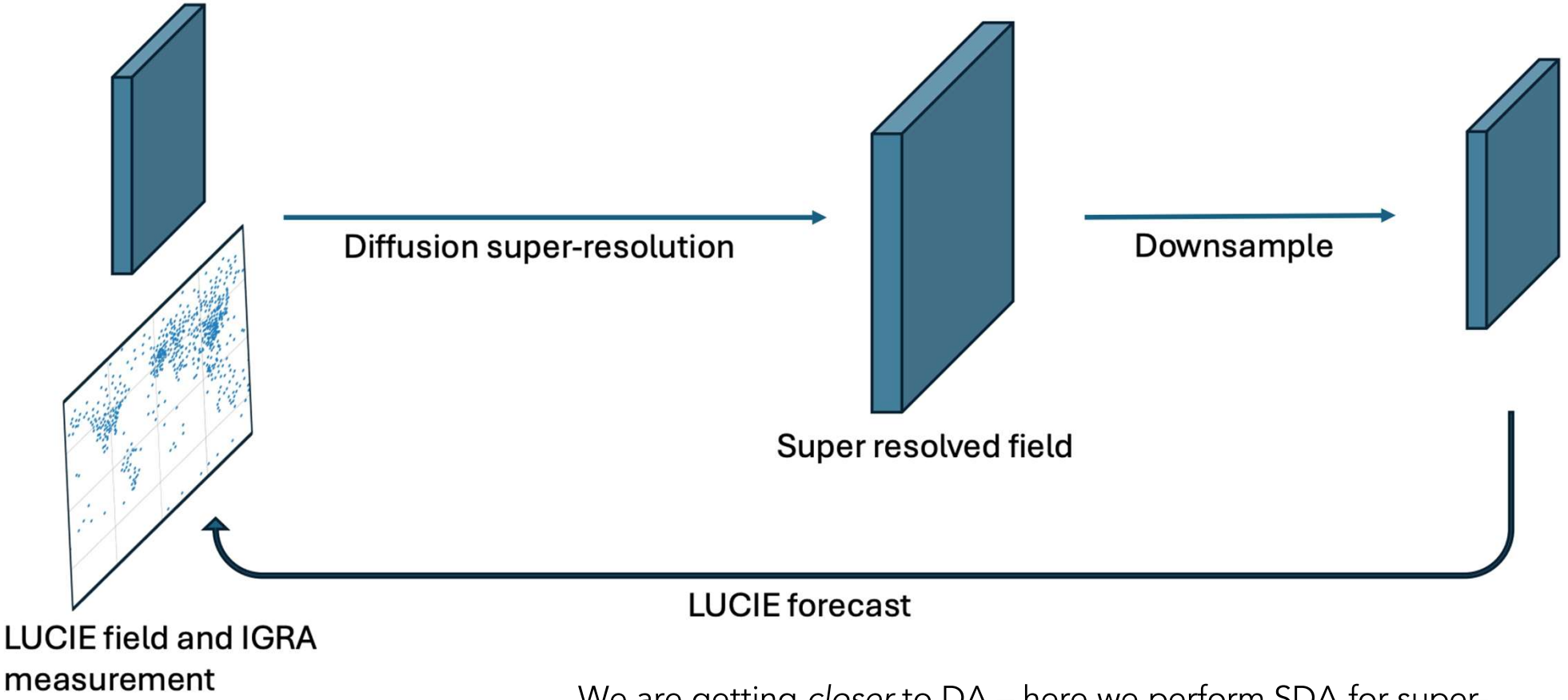
ERA-5 atmospheric data from Jan 1, 2000 to June 1, 2009, June 2009-2010 for testing (subsampled to 3.75 degrees) ~ @2GB of data

Observing an AI emulator: LUCIE



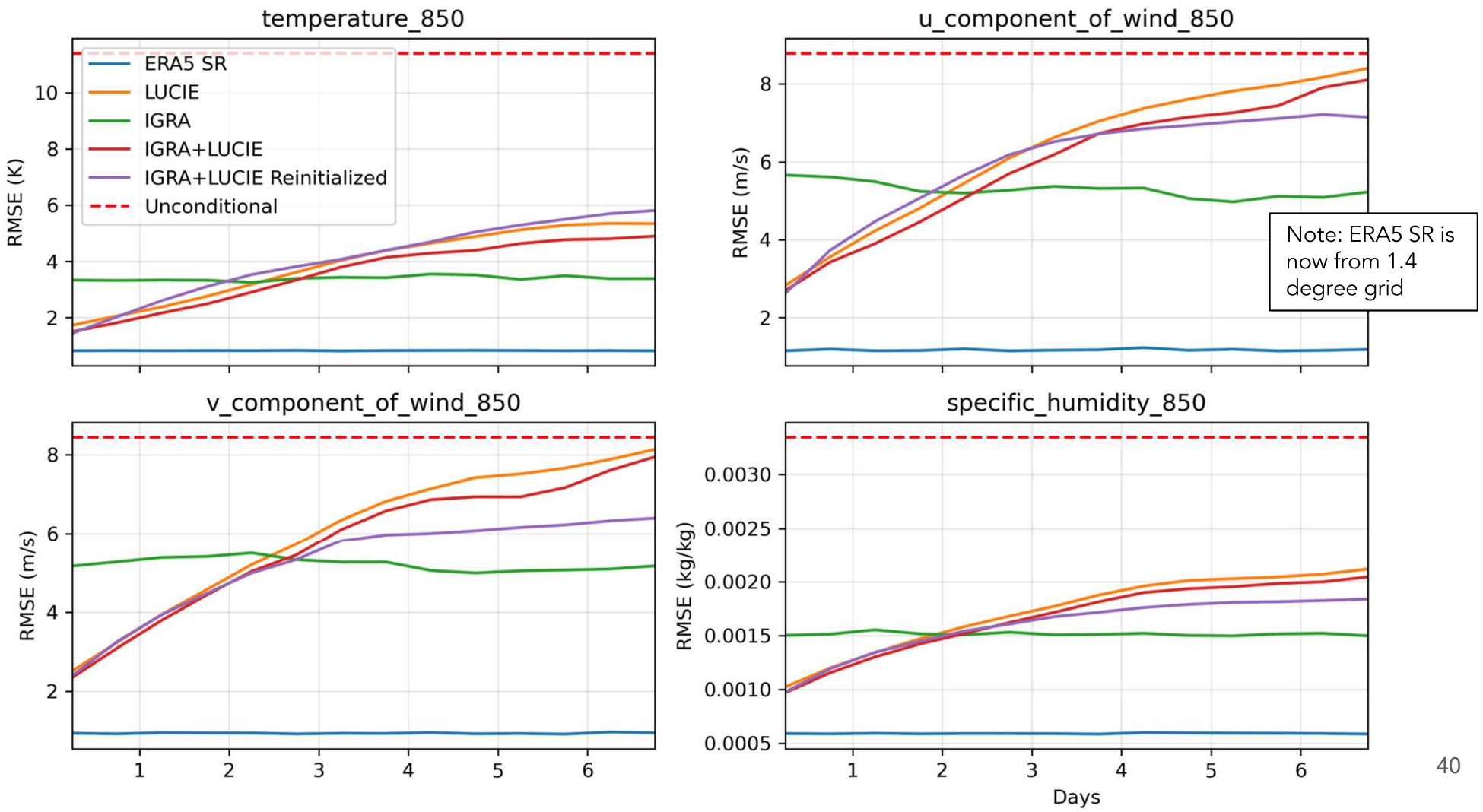
LUCIE is on a 32x64 grid, IGRA is unstructured, generated sample is 128x256

Observing LUCIE and IGRA and reinitializing LUCIE



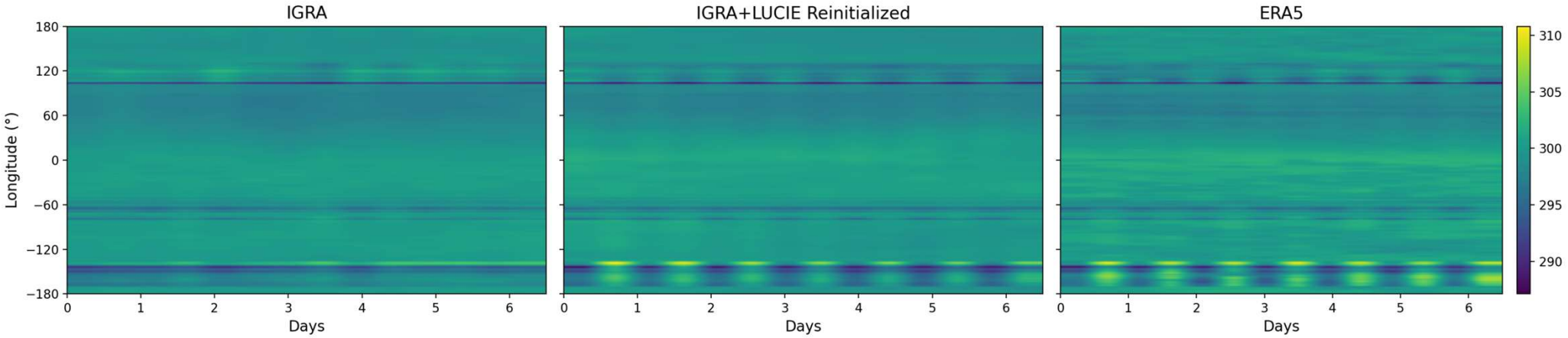
We are getting *closer* to DA – here we perform SDA for super-resolution sequentially but update "observations" of LUCIE by using previous timestep as initial conditions

Observing IGRA and LUCIE



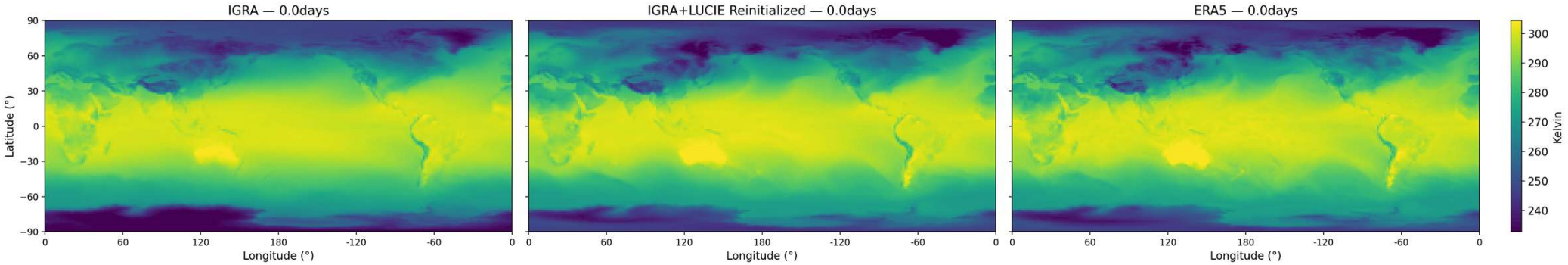
Note: ERA5 SR is now from 1.4 degree grid

Observing IGRA and LUCIE



Time series of ensemble mean surface temperature. The presence of a dynamical core during observations recovers diurnal patterns more accurately!

Observing IGRA and LUCIE



Time series of ensemble mean surface temperature. The presence of a dynamical core during observations recovers diurnal patterns more accurately!

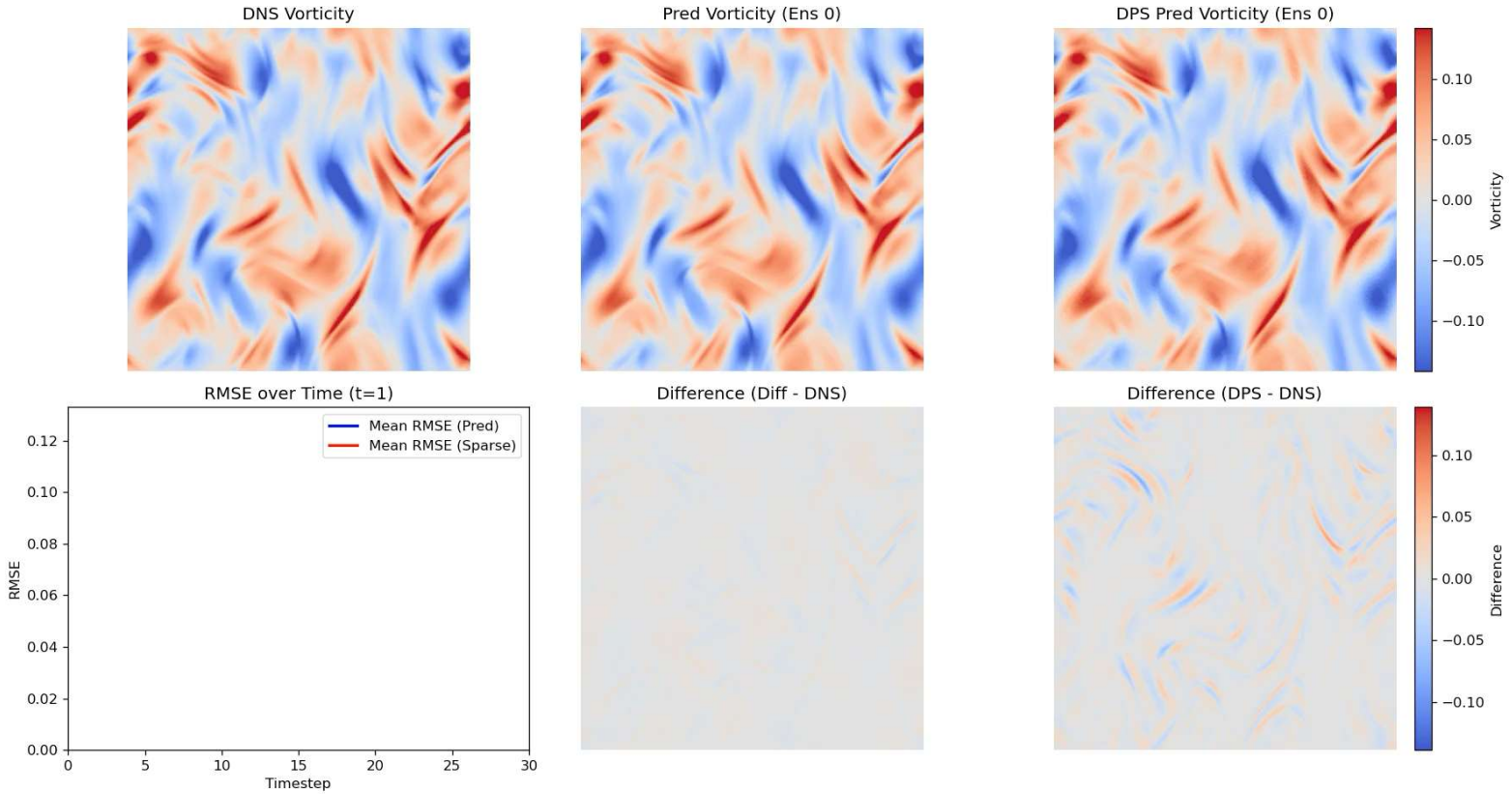
Observing IGRA and LUCIE

Method	Mean Time (s)	Std (s)
LUCIE	1.995	0.00007
IGRA	2.298	0.00504
IGRA+LUCIE	2.298	0.00073
IGRA+LUCIE Reinitialized	2.293	0.00061

More importantly – model and data fusion is FAST. Each ensemble member takes ~2 seconds to generate. This may change once we make resolutions finer.
Note, however, that generation is embarrassingly parallel).

Chakraborty, Dibyajyoti, Haiwen Guan, Jason Stock, Troy Arcomano, Guido Cervone, and Romit Maulik. "Multimodal Atmospheric Super-Resolution With Deep Generative Models." arXiv preprint arXiv:2506.22780 (2025).

Upcoming – conditional forecasting and DPS



For forecasting applications, one can take the state at a previous timestep as an input to the score and rollout into the future. This corresponds to an emulator that bypasses a system of equations. Adding DPS (here 1.5% of points are observed) is another approach to DA (in a more conventional sense) – but again no EnKF or adjoints.

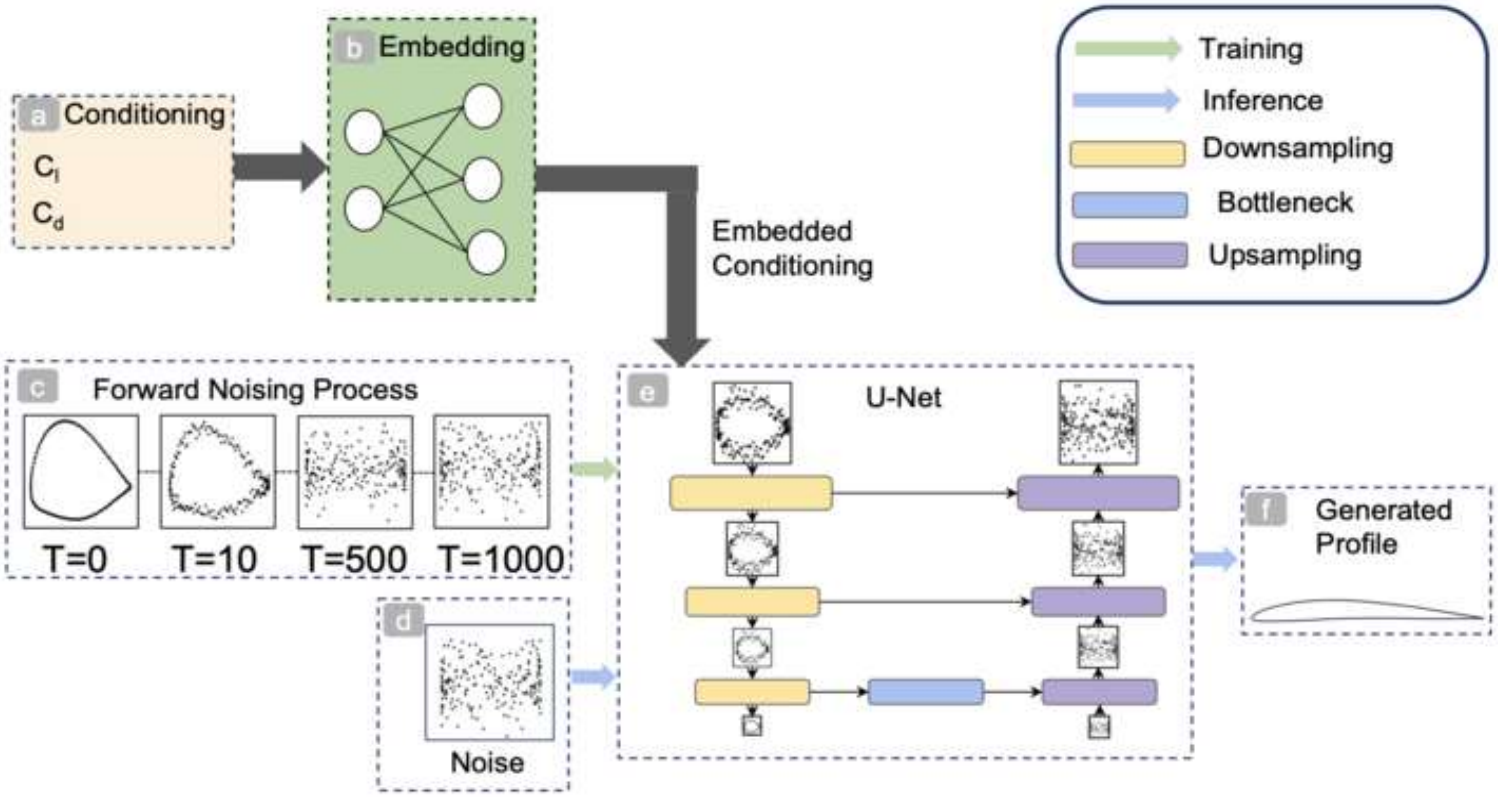
Parting thoughts

- We have swept the topic of covariances under the rug (these were manually tuned) – DA experts are needed there.
- Our framework is *not* variational – but can be. However, should it?
- We have proposed something that can consume observations from various models/data at various resolutions (multiple NWP models can be assimilated).
- This is all just Bayes rule!

PennState
Institute for Computational and Data Sciences

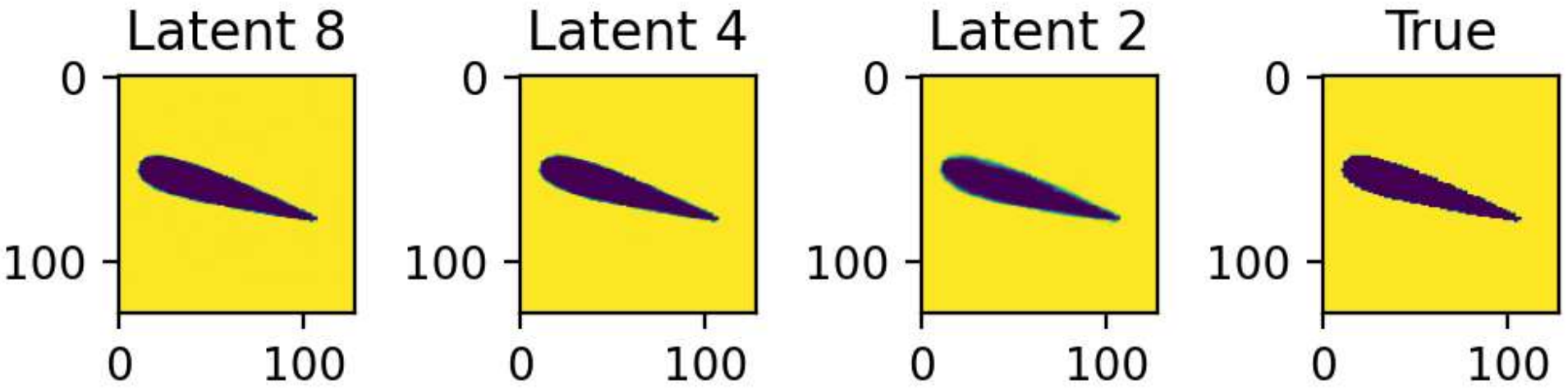
Funding and capability acknowledgements:
Department of Energy (ASCR), Army Research Office (YIP, Modeling of Complex Systems),
Computing Facilities: NERSC Perlmutter, ALCF Polaris, and Penn State ICDS (Roar Collab)

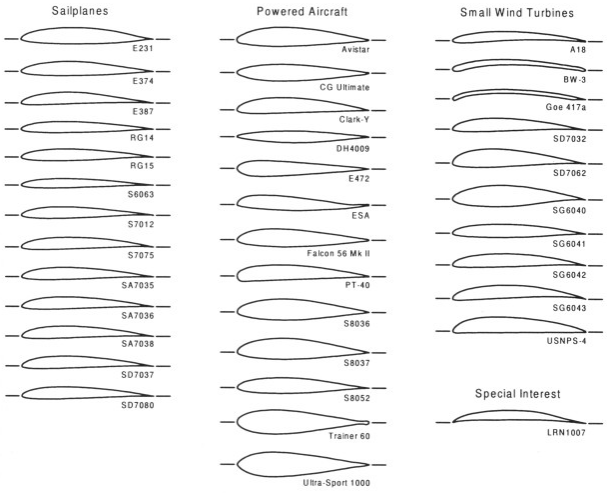
Generative models to bypass Bayesian inference



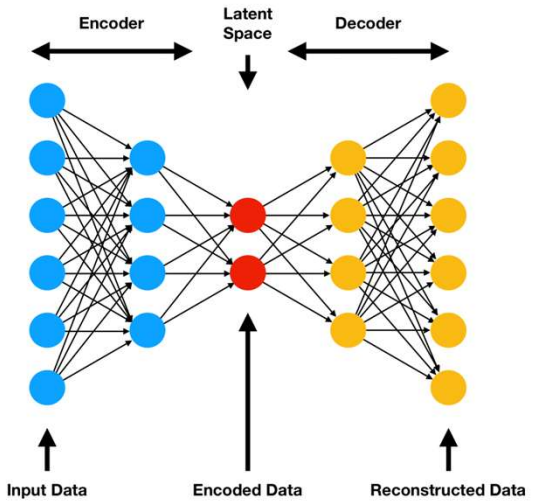
Conditional Denoising Diffusion Model
Graves et. al. 2024 (https://arxiv.org/pdf/2408.15898)

Generative models to bypass Bayesian inference

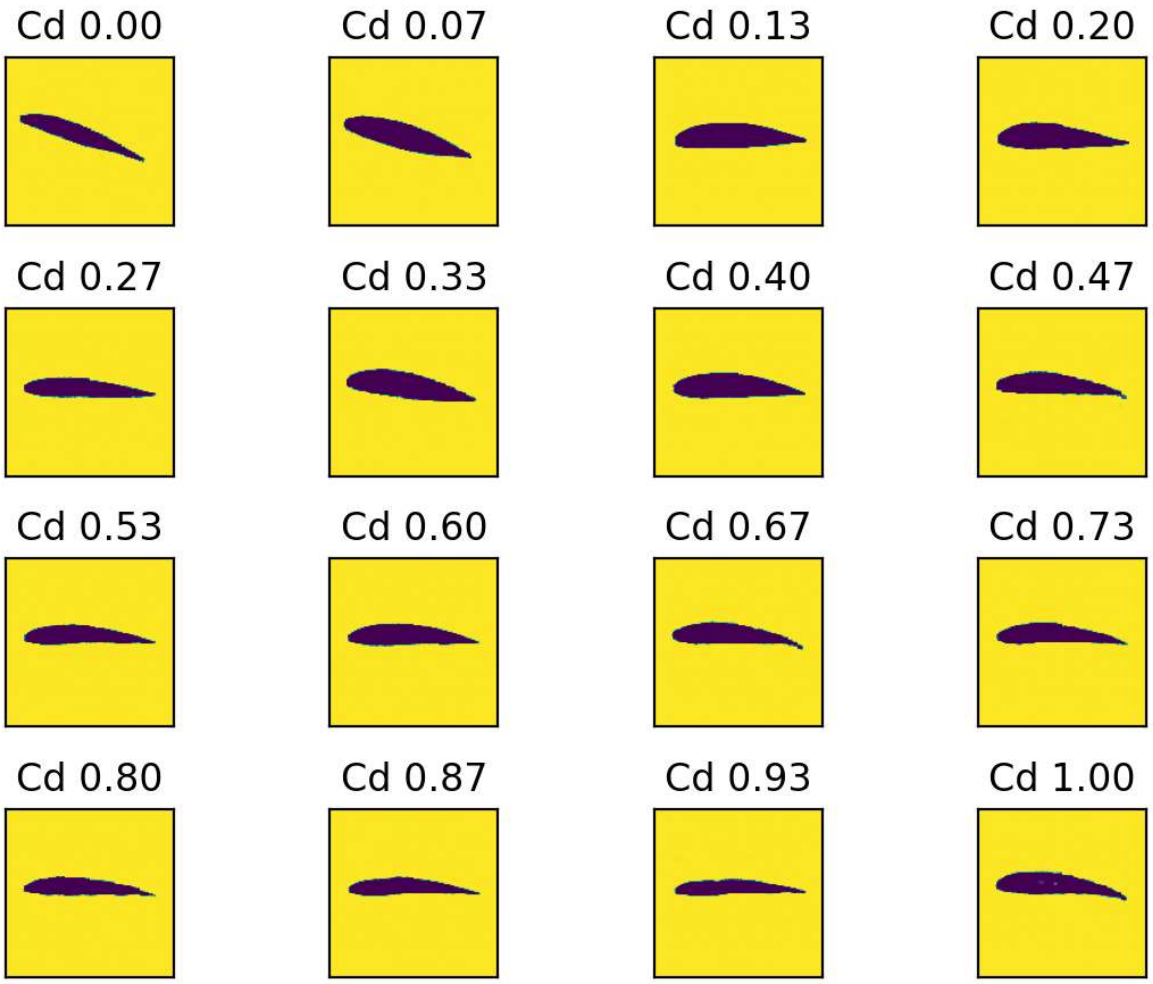




Using an autoencoder to compress airfoil data from the UIUC airfoil dataset

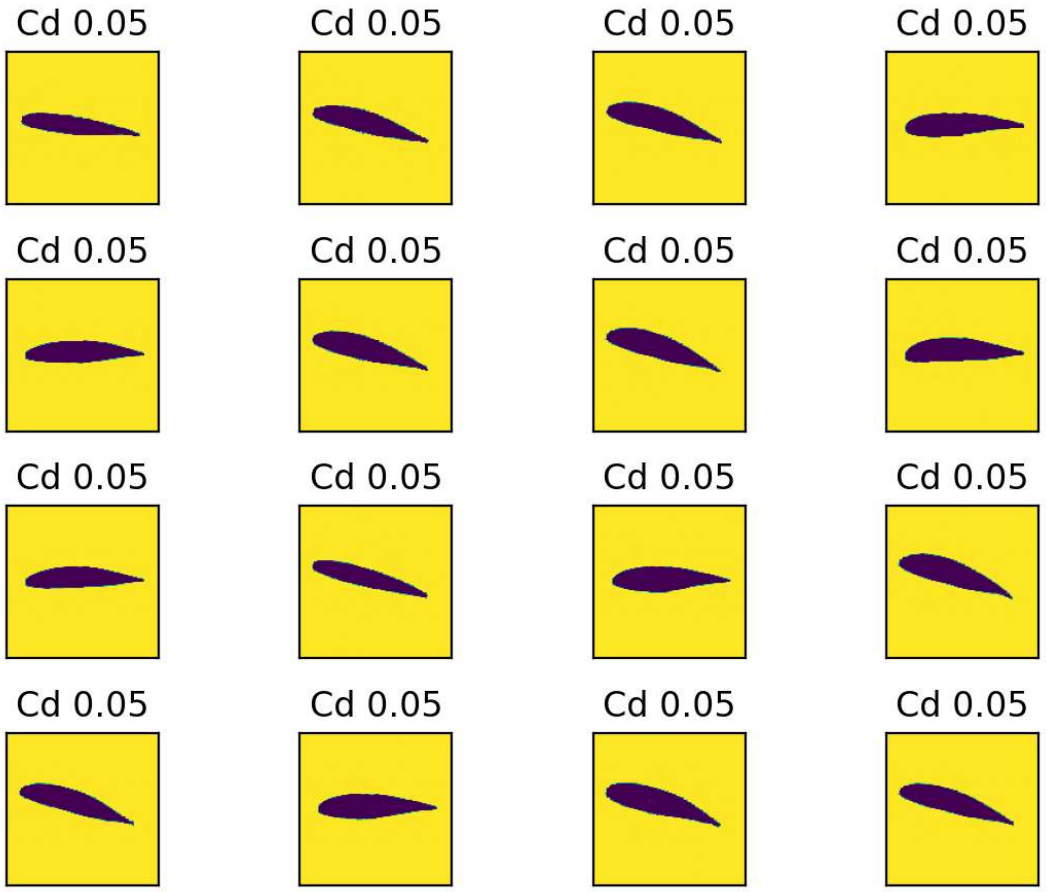


Generative models to bypass Bayesian inference

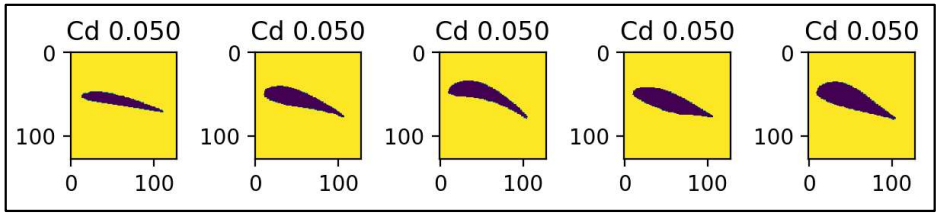


A diffusion model that is able to generate different airfoil profiles, condition on the drag coefficient

Generative models to bypass Bayesian inference

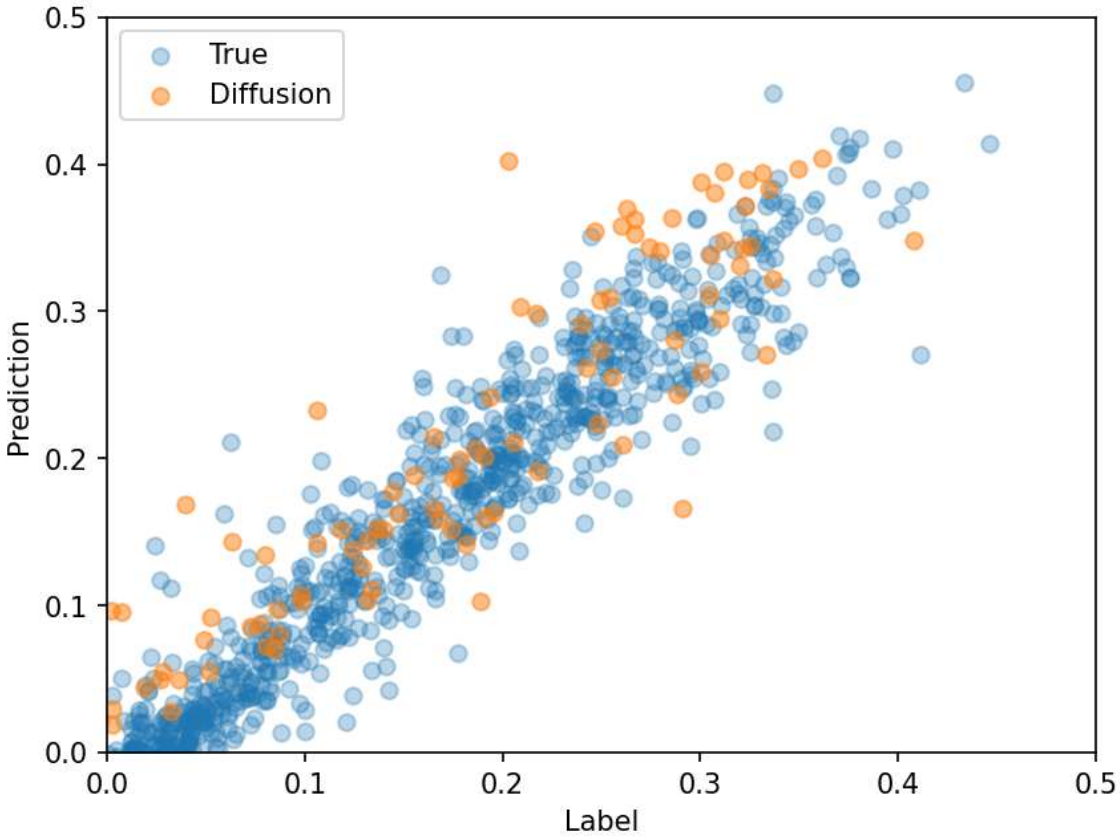


Samples from diffusion model



Samples from training data

Generative models to bypass Bayesian inference



An approximate verification of the conditional generation process

Ideally each generated airfoil shape should be propagated to an 'experiment' for computing Cd.

Here we build a CNN based regression model for predicting Cd (R^2 value of 0.9) on the training data set and check for if conditional input is recovered.