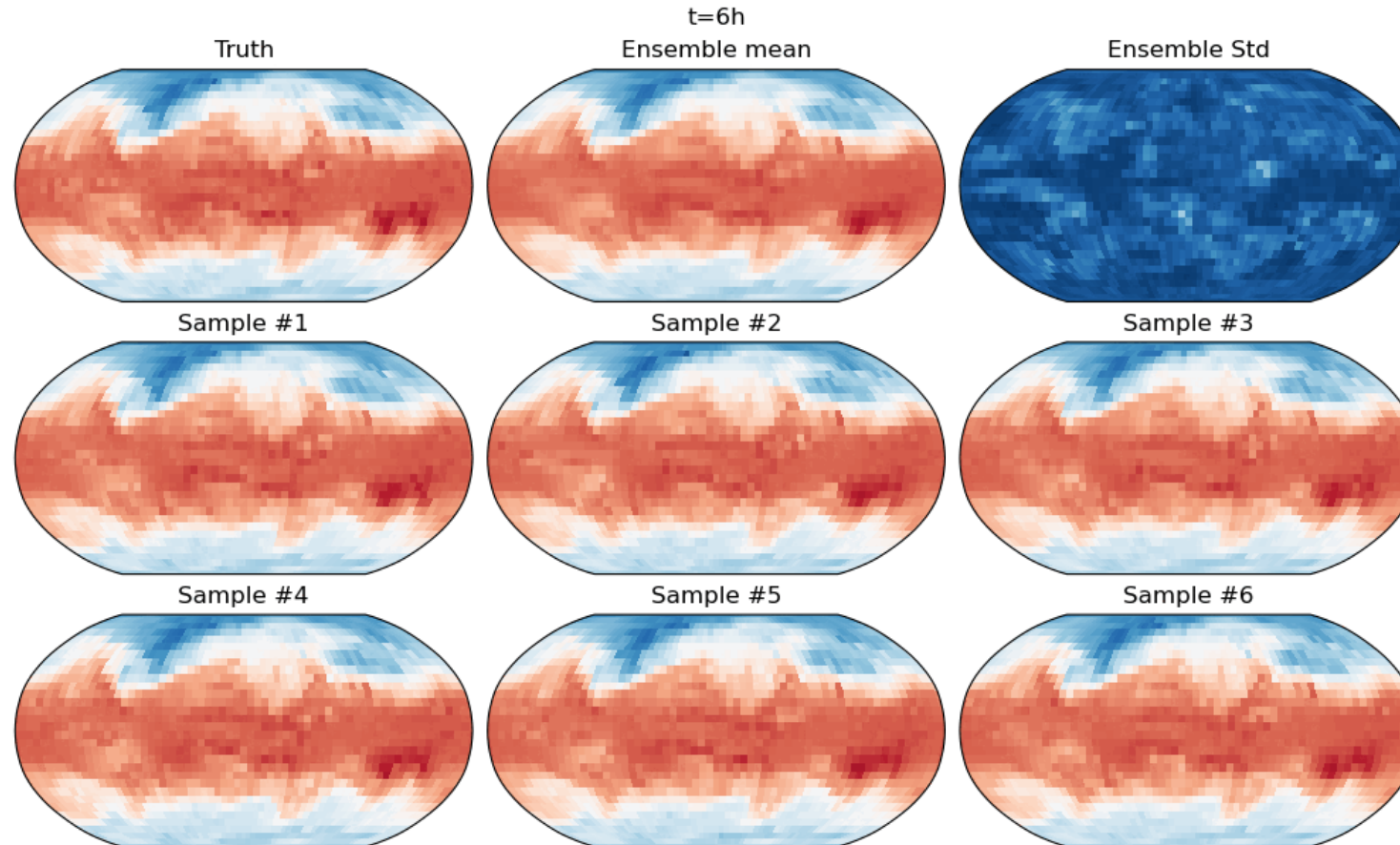


Intro to Probabilistic Deep Learning

(with focus on diffusion models)



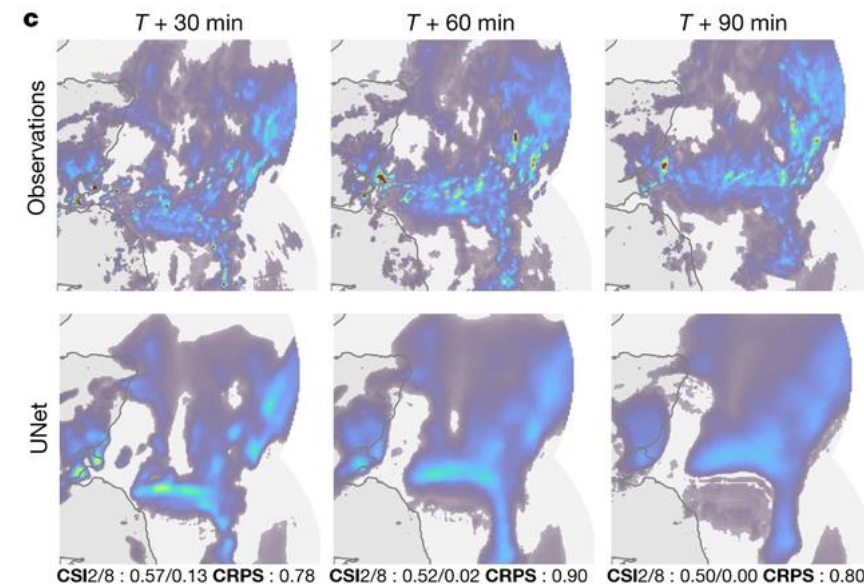
Paul Keil

Hereon; paul.keil@hereon.de

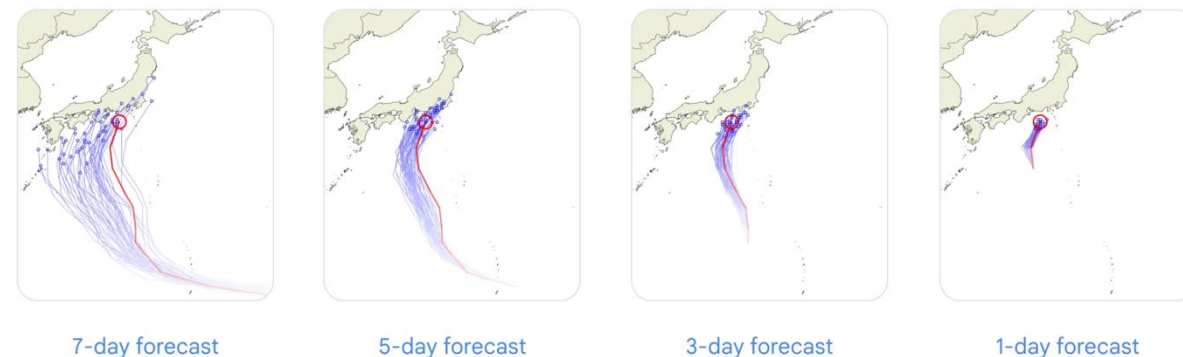
Deep Learning for Climate Scientists 2026

Why Probabilistic Deep Learning

- Overconfidence of standard NNs
- Many standard architectures trained on RMSE suffer from oversmoothing
- Probabilistic Methods are use for classical modelling (ensembles)
 - Risk communication to stakeholders
 - Deep Learning enables extremely large ensembles



Ravuri, S., Lenc, K., Willson, M. *et al.* Skilful precipitation nowcasting using deep generative models of radar. *Nature* **597**, 672–677 (2021). <https://doi.org/10.1038/s41586-021-03854-z>



—○— Cyclone track up to October 12, 2019, at 0600 UTC —○— Possible paths predicted by GenCast

Aleatoric and Epistemic Uncertainty

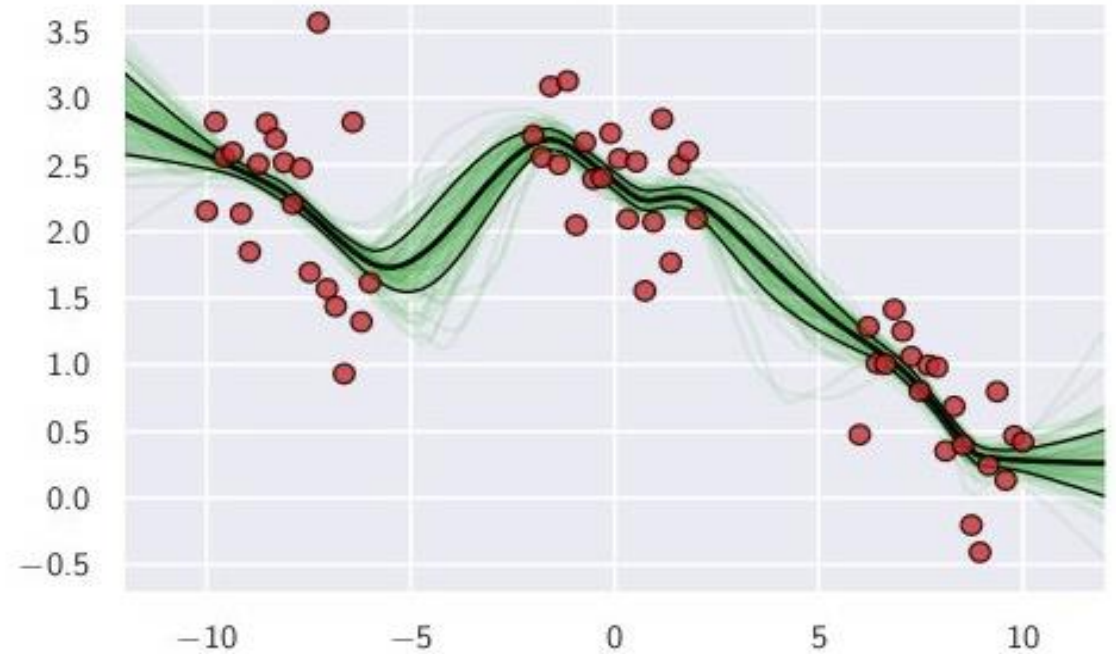
Epistemic:

- Model uncertainty
- Reducible uncertainty
- Reduces as you increase data
- Increases as you move away from your data
- Epistemic uncertainty quantification should not cover all your data!

Aleatoric:

- Data noise
- Irreducible/Intrinsic uncertainty
- Related to shortcomings in how the data is collected/produced

It's important to think about these categories for the interpretation of results. Different methods provide different uncertainty types.



$$\sigma^2 = \sigma_{epistemic}^2 + \sigma_{aleatoric}^2$$

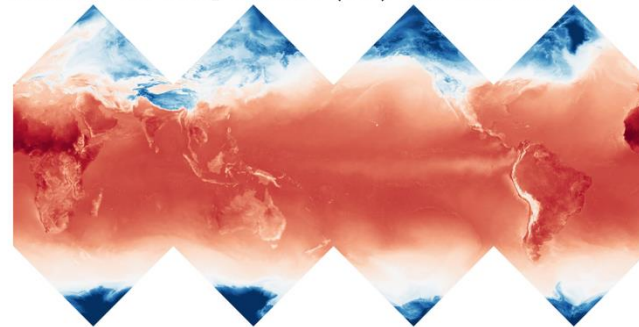
Diffusion Models

- **State-of-the-art:**
Used for high-quality image generation
- **Generative:**
Learns the distribution to sample from it
- **Conditional:**
output conditioned on input
- **Non-Deterministic**
- **Capture Epistemic and Aleatoric Uncertainty:**
Generates plausible samples

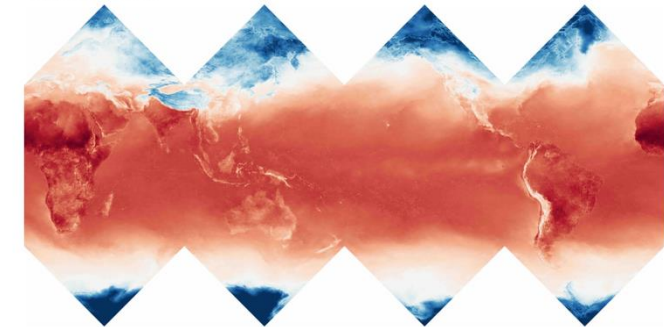


<https://stable-diffusion.io>

Surface air temperature (tas) Ground truth



Generated



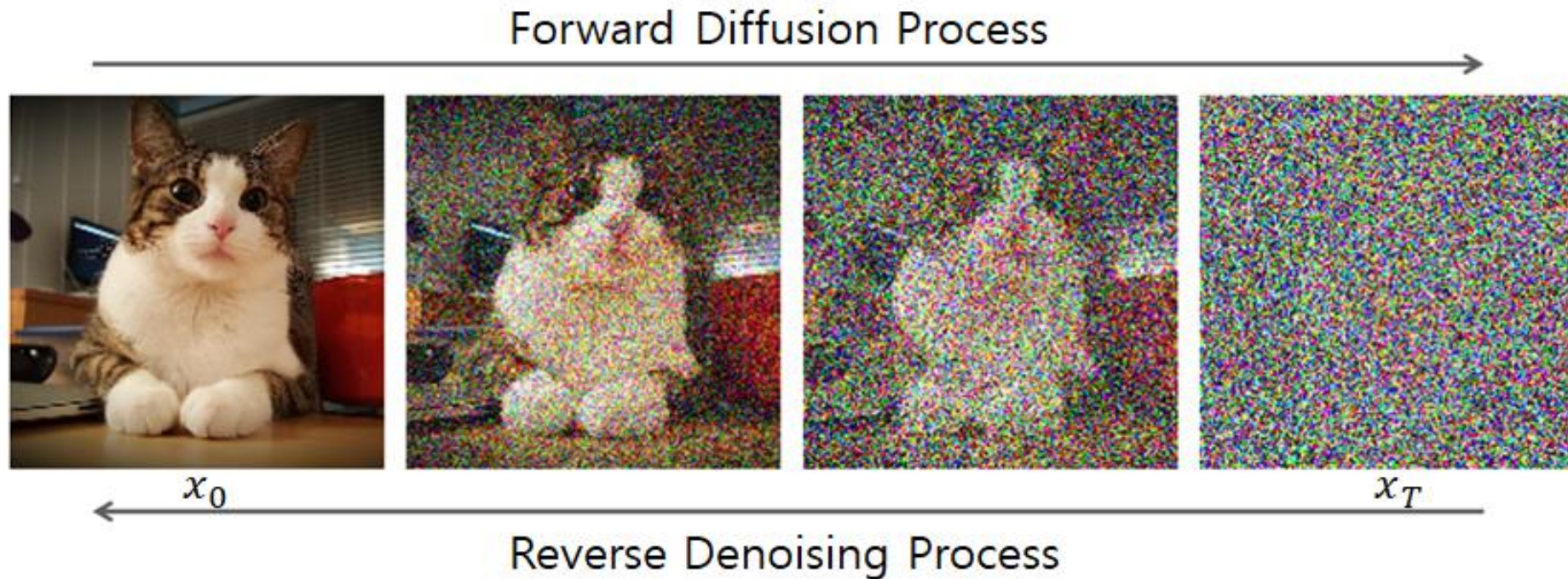
Brenowitz, Noah D., et al. "Climate in a bottle: Towards a generative foundation model for the kilometer-scale global atmosphere." *arXiv preprint arXiv:2505.06474* (2025).

Diffusion Models

- **Many different implementation options, ongoing development**
- **We will look at:**
 - Denoising Diffusion Probabilistic Models (DDPM; Ho et al. 2020)
 - EDM (Karras, 2023)
 - Flow Matching (Lipmann, 2022)
 - Latent Diffusion (Rombach, 2022)
- **Derivation of equations require time and detailed stochastics knowledge**
- **We focus on high level concepts and implementation**

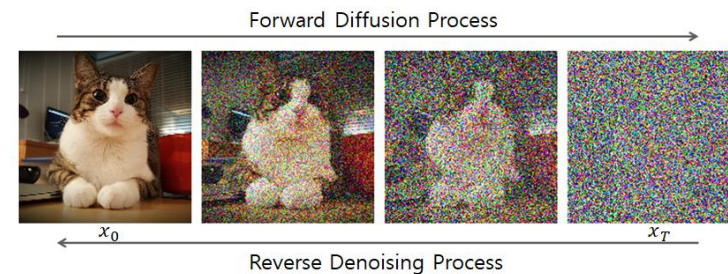
Denoising Diffusion Probabilistic Models (DDPM; Ho et al. 2020)

- **Core idea:**
 - Artificially introduce stepwise noise along time t in your training data
 - Let a NN learn how to denoise for a random t



Denoising Diffusion Probabilistic Models (DDPM; Ho et al. 2020)

- **Core idea:**
 - Artificially introduce stepwise noise along time t in your training data
 - Let a NN learn how to denoise for a random t



- **Introduce noise epsilon**
 - Alpha(t) is the noise scheduler
 - Alpha=0: pure noise
 - Alpha=1: clean image
 - Important hyperparameter!

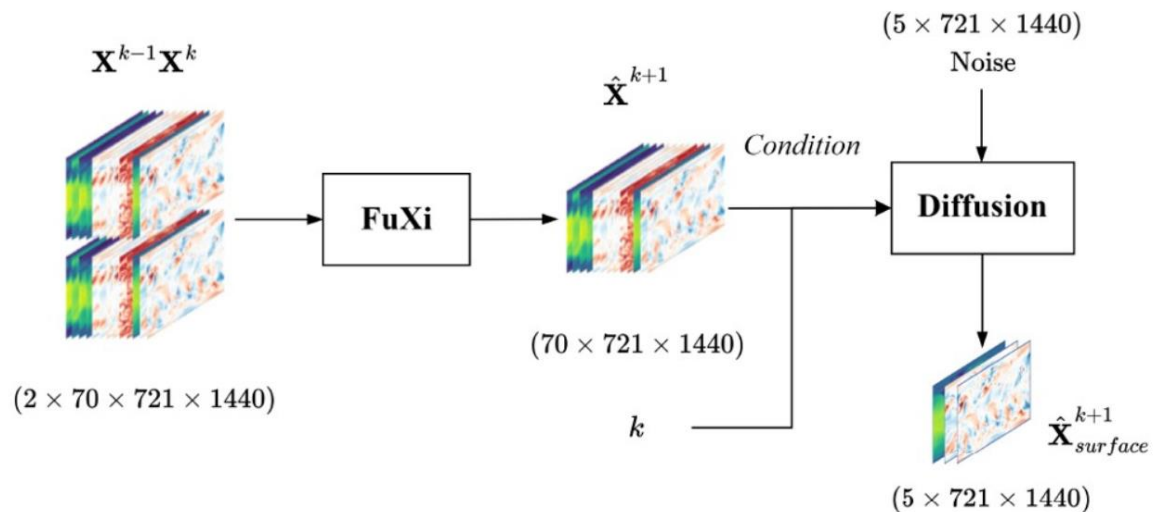
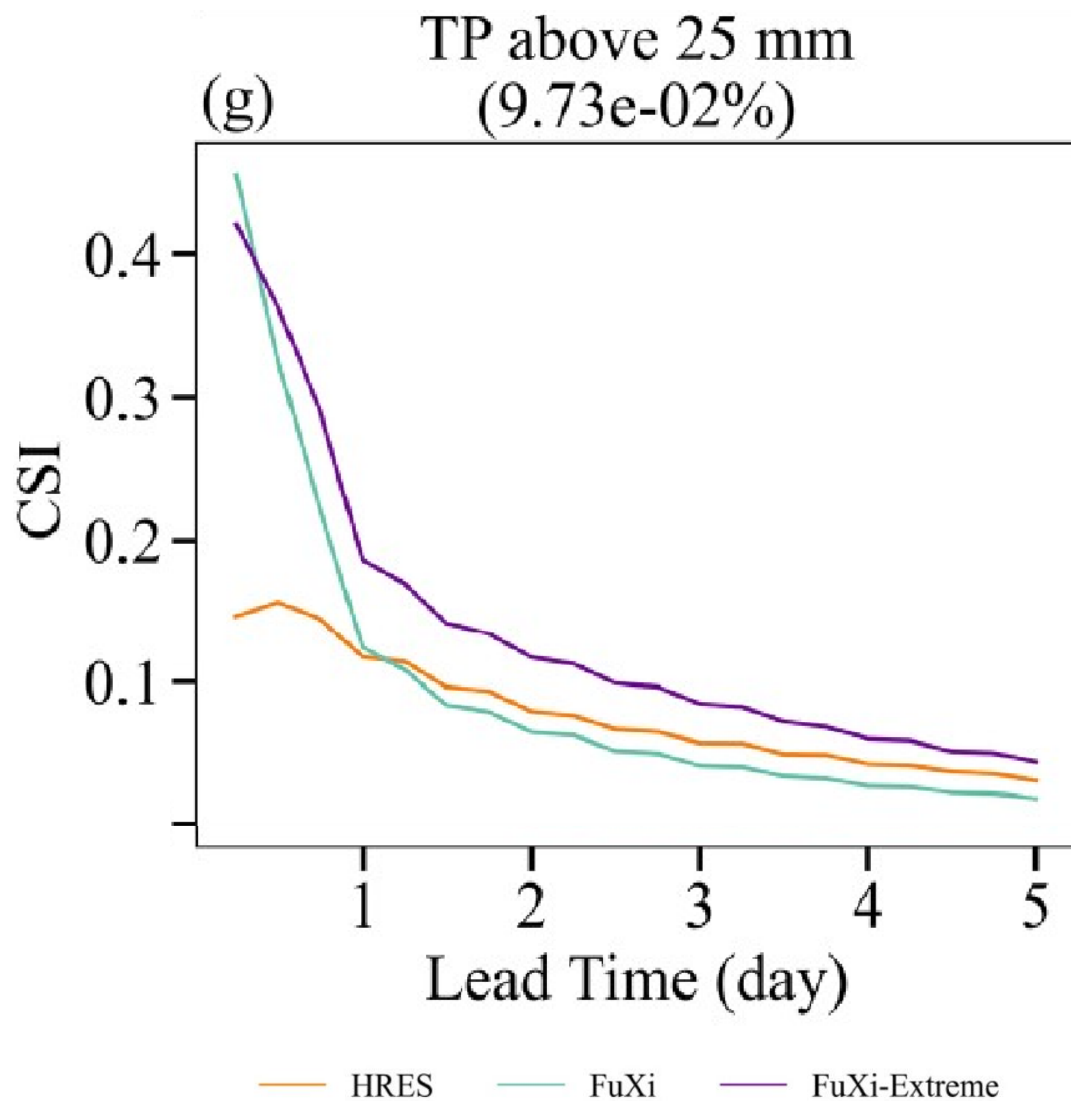
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

- **Loss function: RMSE of predicted noise vs actual noise**

$$\mathcal{L} = \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2$$

- **One NN (typically a UNET) for all noise levels t !**
- **Iterative sampling over t makes it expensive (more than 1000 sampling steps necessary sometimes)**
- **New noise every epoch and during inference → stochastic**

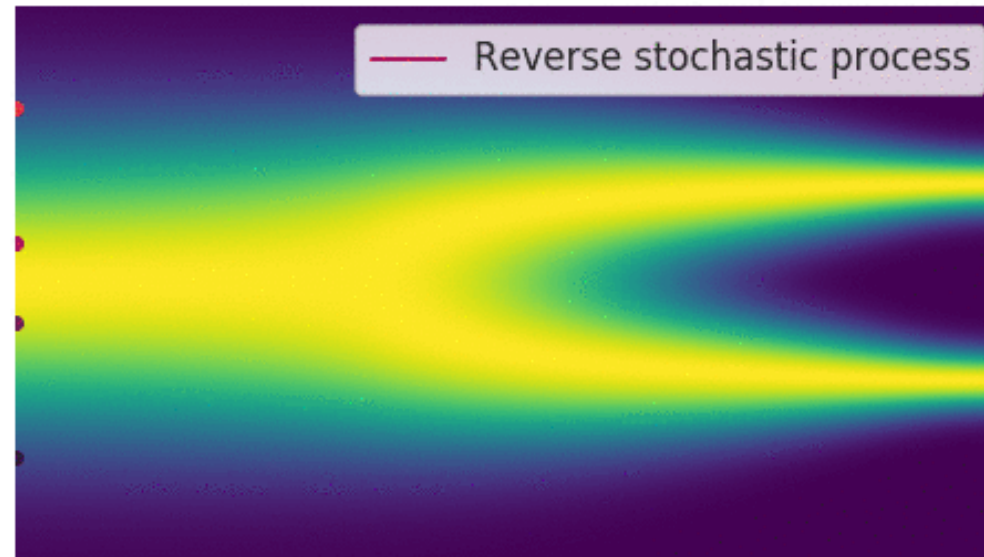
DDPM for weather forecasting: FuXi-Extreme



Zhong, Xiaohui, et al. "FuXi-Extreme: Improving extreme rainfall and wind forecasts with diffusion model." *Science China Earth Sciences* 67.12 (2024): 3696-3708.

EDM

Karras, Tero, et al. "Elucidating the design space of diffusion-based generative models." *Advances in neural information processing systems* 35 (2022): 26565-26577.



<https://yang-song.net/blog/2021/score/>

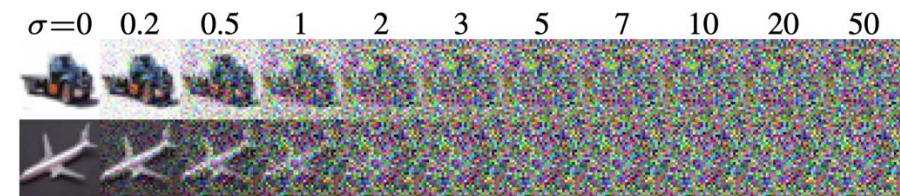
EDM

- Denoising can be framed as solving an ordinary differential equation (ODE) or a stochastic differential equation (SDE)
- **Noise schedule simpler: sigma is noise dial**
 - noise level sigma and time t interchangeable
 - Continuous noise regime
- **Network D learns denoising function in ODE**
 - You find an optimal path from pure noise to the data distribution
 - Here D outputs best guess for x_0 . No ODE solving during training (“simulation free”)!
 - Problem: Scaling differences between different levels of noise. Scaling needed within D (“preconditioning”)
 - Inference: done step by step. An ODE Solver (Euler, Runge-Kutta...) is used for an accurate prediction, important design choice!
- **ODE: random initial noise; SDE: inject noise at every timestep**
- **Sampling more efficient than DDPM, only 30-40 sampling steps**

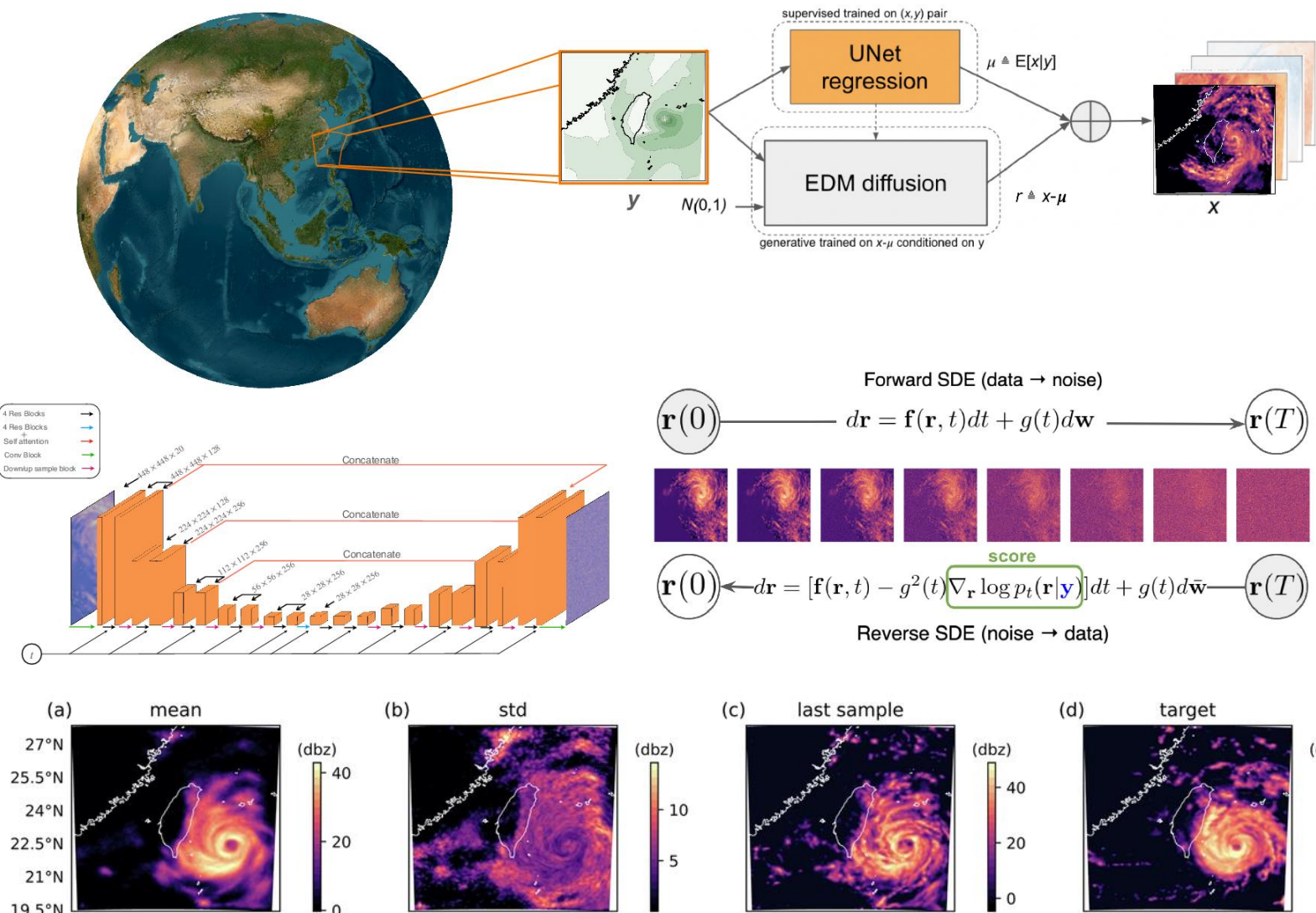
$$x = x_0 + \sigma \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

$$\frac{dx}{d\sigma} = \frac{x - D(x; \sigma)}{\sigma}$$

$$\mathcal{L} = \lambda(\sigma) \|D(x; \sigma) - x_0\|^2$$

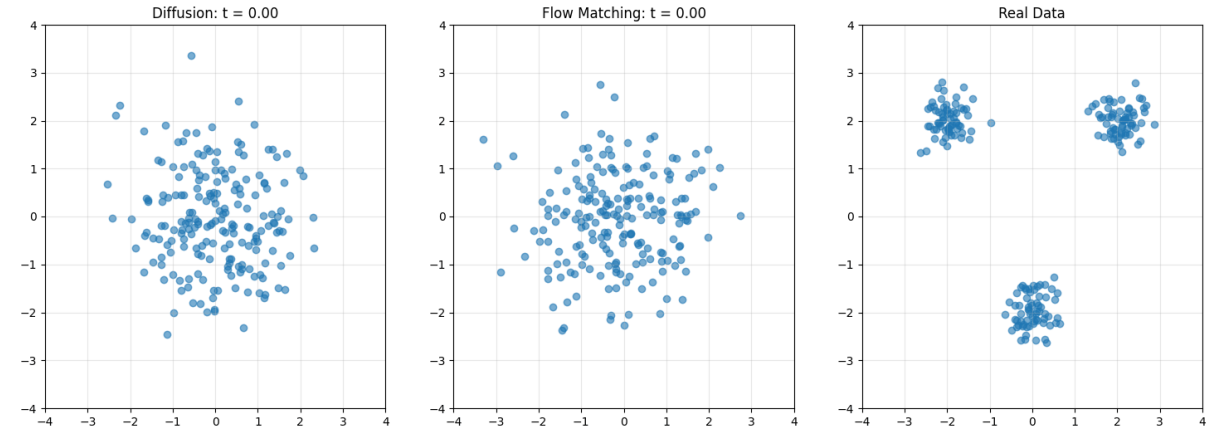


EDM for Downscaling: CorrDiff, Mardani et al., 2025

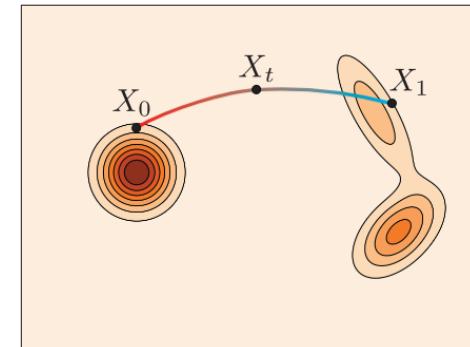


Conditional Flow Matching

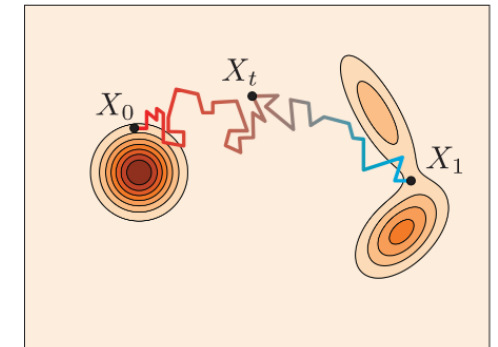
- **Core idea: Learn a vector field to define a deterministic flow**
 - Transport samples along a learned path from noise to data
 - Problem with EDM: Non-Linear probability path: careful numerical integration at inference time
 - Flow matching allows to optimize the path in the velocity field (“generate desired probability path”)
- **Related: “Continuous Normalising Flows”**
- Lipman, Yaron, et al. "Flow matching for generative modeling." arXiv preprint arXiv:2210.02747 (2022).



<https://harshm121.medium.com/flow-matching-vs-diffusion-79578a16c510>



(a) Flow



(b) Diffusion

Lipman, Yaron, et al. "Flow matching guide and code." arXiv preprint arXiv:2412.06264 (2024).

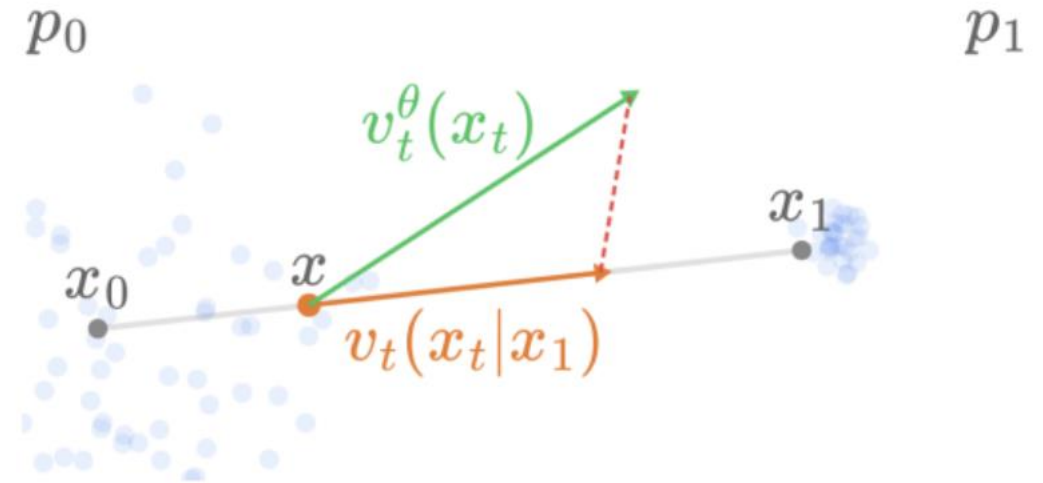
Conditional Flow Matching

- **Find velocity:**
 - Ground truth: Straight Line with constant velocity between noise and data
 - Calculate loss at arbitrary noise level t
 - You can add noise to interpolation path
 - Simulation free during training
- **Conceptually simpler and efficient inference**
 - Less hyperparameters (e.g. noise scheduling)
 - Less integration steps during inference
 - Can be used with arbitrary (non-gaussian) initial noise
- (with the right noise scheduling and loss weighting, diffusion models can be formulated equivalently)
- **Rectified Flow (Liu, 2022), Consistency/Shortcut Models for even faster sampling**

$$\frac{dx}{dt} = v_{\theta}(x, t)$$

$$x_t = t x_1 + (1 - t) x_0,$$

$$\mathcal{L} = \mathbb{E}_{t, x_0, x_1} [\|v_{\theta}(x_t, t) - u_t(x_t | x_1)\|^2]$$



<https://alechebling.com/blog/rectified-flow/>

Weather Prediction using Flow Matching

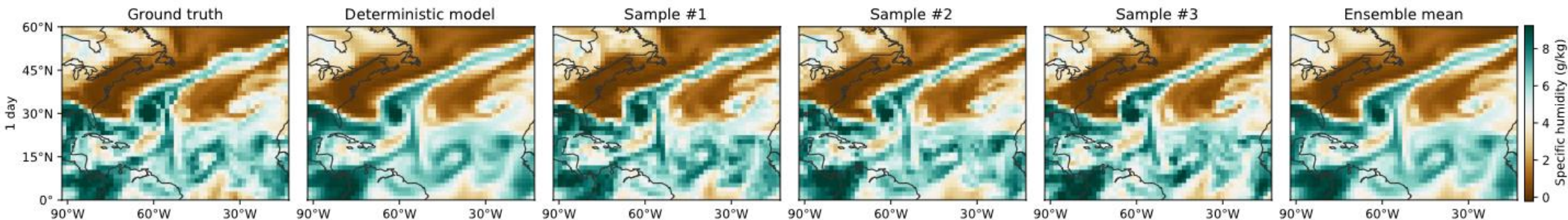
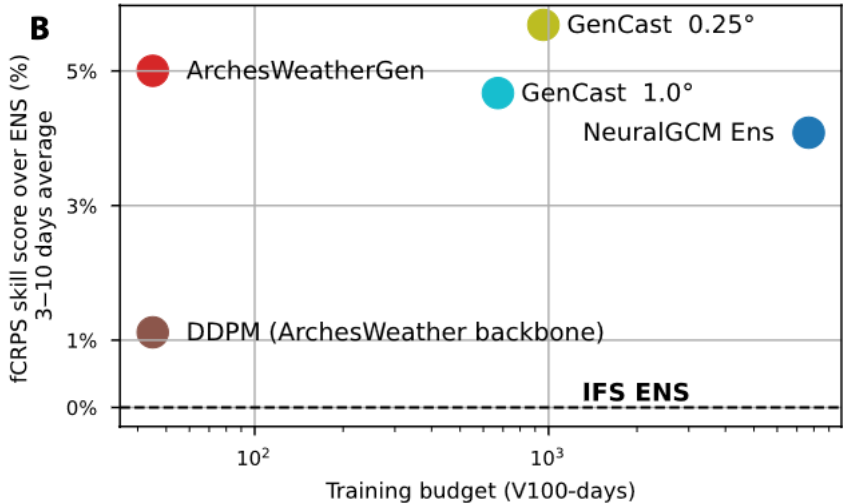
SCIENCE ADVANCES | RESEARCH ARTICLE

MACHINE LEARNING

ArchesWeatherGen: Skillful and compute-efficient probabilistic weather forecasting with machine learning

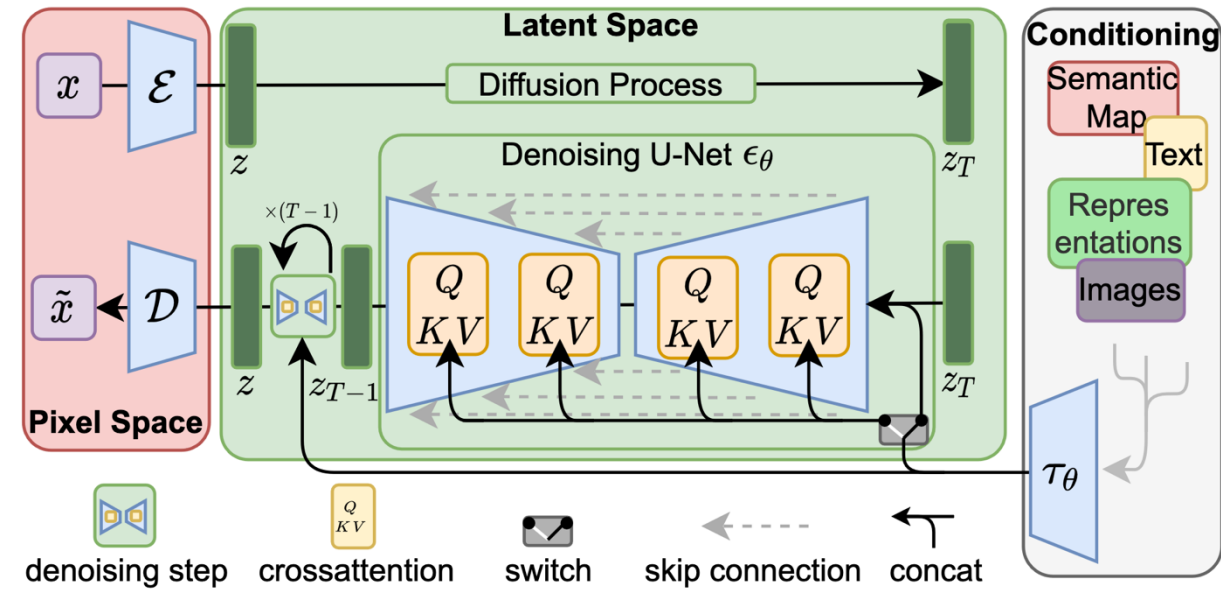
Guillaume Couairon^{1*†}, Renu Singh^{1†}, Anastase Charantonis¹, Christian Lessig², Claire Monteleoni^{1,3}

Weather forecasting plays a vital role in today's society, from agriculture and logistics to predicting the output of renewable energies and preparing for extreme weather events. Deep learning weather forecasting models trained with the next state prediction objective on ERA5 have shown great success compared to numerical global circulation models. Here, we propose a methodology to leverage deterministic weather models in the design of probabilistic weather models, leading to improved performance and reduced computing costs. We design a probabilistic weather model based on flow matching, a modern variant of diffusion models, that is trained to project deterministic weather predictions to the distribution of ERA5 weather states. Our model ArchesWeatherGen surpasses IFS ENS and NeuralGCM on all WeatherBench headline variables (except for NeuralGCM's geopotential). Our work also aims to democratize the use of generative machine learning models in weather forecasting research.



Latent Diffusion

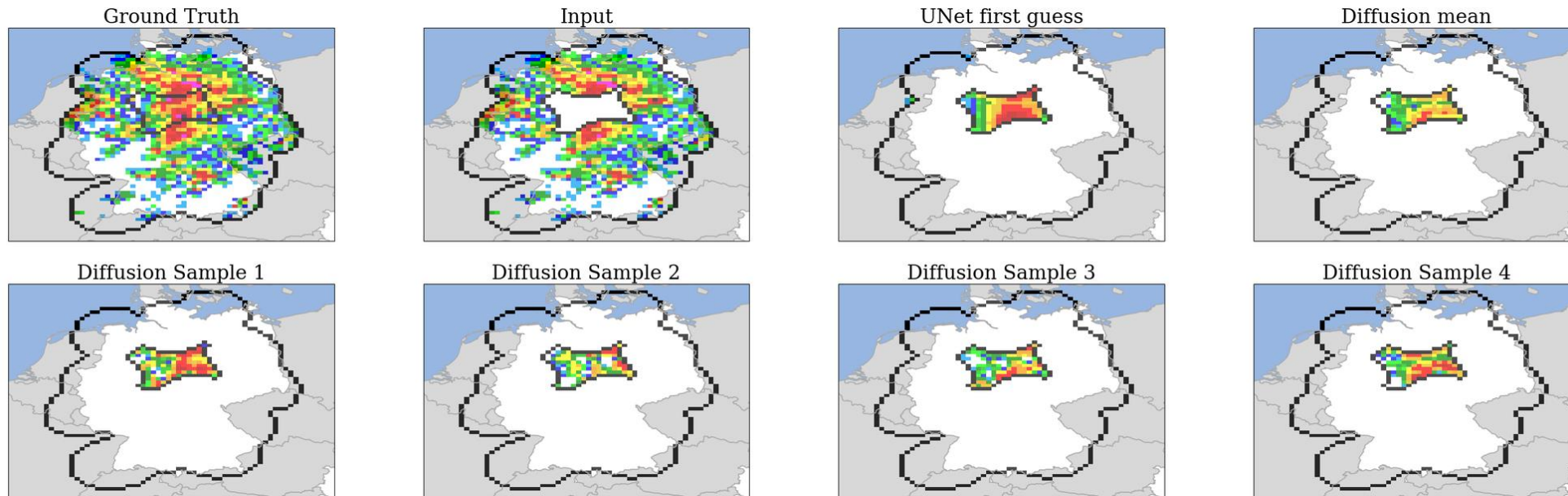
- **Problem: Diffusion Modelling is expensive**
 - costly integration steps in the high dimensional physical/pixel space
 - Attention scales quadratically with data amount
 - High resolution modelling extremely costly
- **Latent Diffusion moves the costly integration to the latent space**
 - Encoder: Downsample image
 - Latent Diffusion in low-dimensional space
 - Denoise the low-dimensional image
 - Decoder: Upsample back to physical space
- **Compatible with different diffusion methods (DDPM, EDM, Flow Matching)**



Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." 2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR). iee, 2022.

Exercise

- Builds on infilling exercise
- You are implementing a simple Flow Matching Model
- Analyse results



torch.rand

`torch.rand(*size, *, generator=None, out=None, dtype=None, layout=torch.strided, device=None, requires_grad=False, pin_memory=False) → Tensor`

Returns a tensor filled with random numbers from a uniform distribution on the interval $[0, 1)$

The shape of the tensor is defined by the variable argument `size`.

```
>>> a = torch.rand(2,2)
>>> print(a)
tensor([[0.2842, 0.7267],
        [0.6119, 0.1798]])
>>> b = torch.randn(2,2)
>>> print(b)
tensor([[ 0.6054, -0.1861],
        [-0.9877,  1.5263]])
```

torch.randn_like

`torch.randn_like(input, *, generator=None, dtype=None, layout=None, device=None, requires_grad=False, memory_format=torch.preserve_format) → Tensor`

Returns a tensor with the same size as `input` that is filled with random numbers from a normal distribution with mean 0 and variance

```
>>> c = torch.randn_like(b)
>>> print(c)
tensor([[ -1.1858,  1.3560],
        [-0.7387,  0.2088]])
```

Additional Packages

- **Diffusers**

<https://huggingface.co/docs/diffusers/index>

- **flow_matching**

https://facebookresearch.github.io/flow_matching/

- **torch_cfm**



```
scaled_input = scheduler.precondition_inputs(noisy_target, sigma)
model_input = torch.cat([source, scaled_input], dim=1)
c_noise = scheduler.precondition_noise(sigma).flatten()

raw_output = model(model_input, c_noise).sample
denoised = scheduler.precondition_outputs(noisy_target, raw_output, sigma)
```

Additional Resources (Blogs)

- **Intro to diffusion modelling**
 - <https://lilianweng.github.io/posts/2021-07-11-diffusion-models>
 - <https://yang-song.net/blog/2021/score>
 - <https://towardsdatascience.com/a-visual-guide-to-how-diffusion-models-work/>
 - <https://diffusion.csail.mit.edu/2026/index.html>
- **Flow Matching vs Diffusion Modelling:**
 - <https://harshm121.medium.com/flow-matching-vs-diffusion-79578a16c510>
 - <https://diffusionflow.github.io/>

