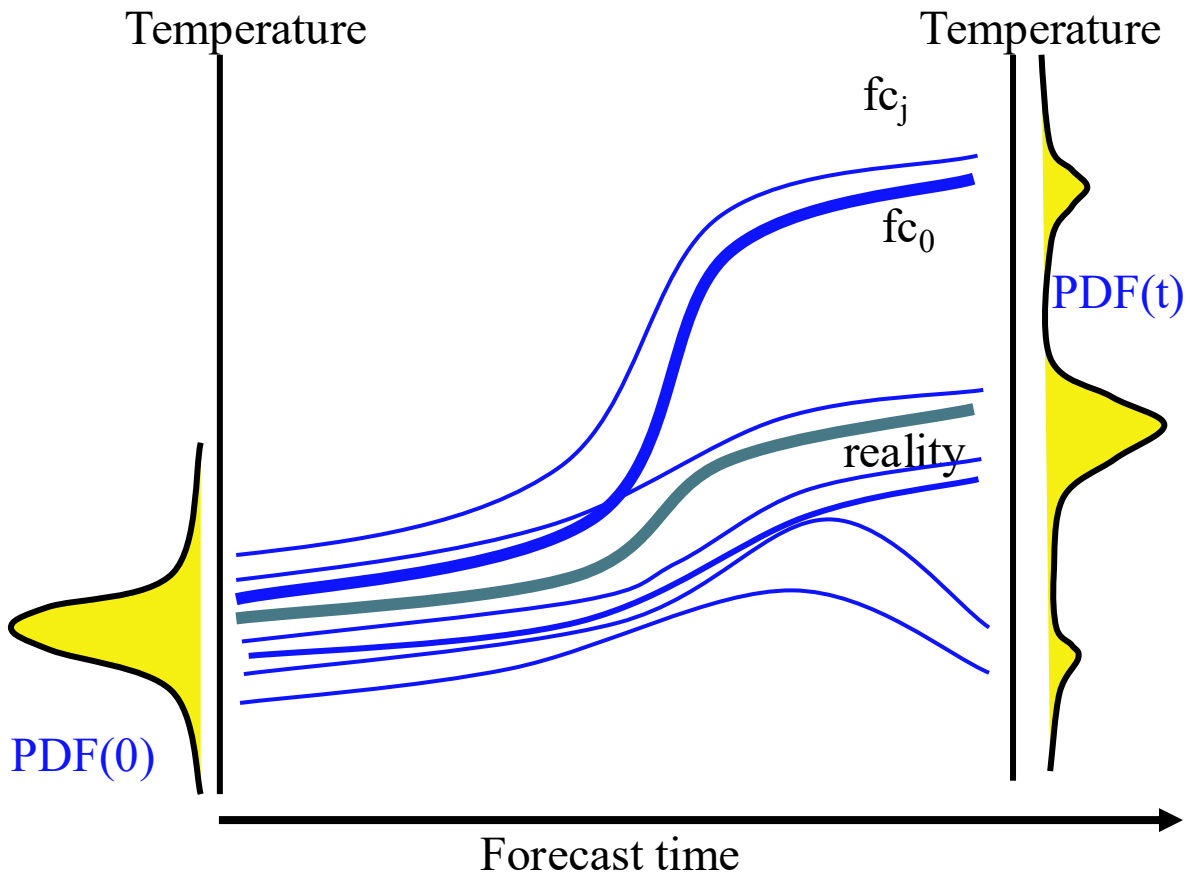
Generative models for weather forecasting

Simon Lang

ECMWF



Ensemble prediction systems

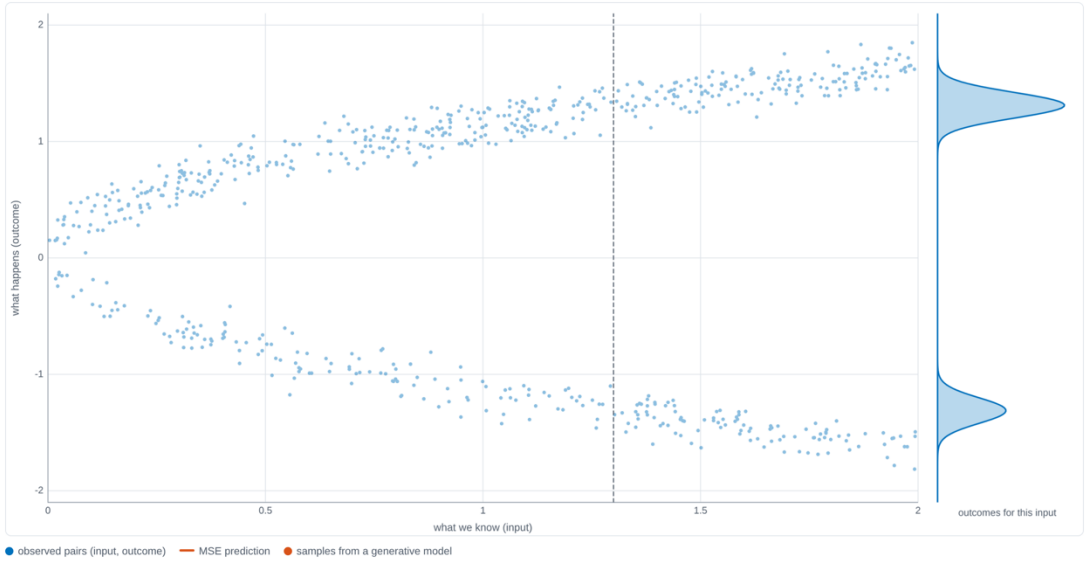


Predicting realistic samples from the target distribution

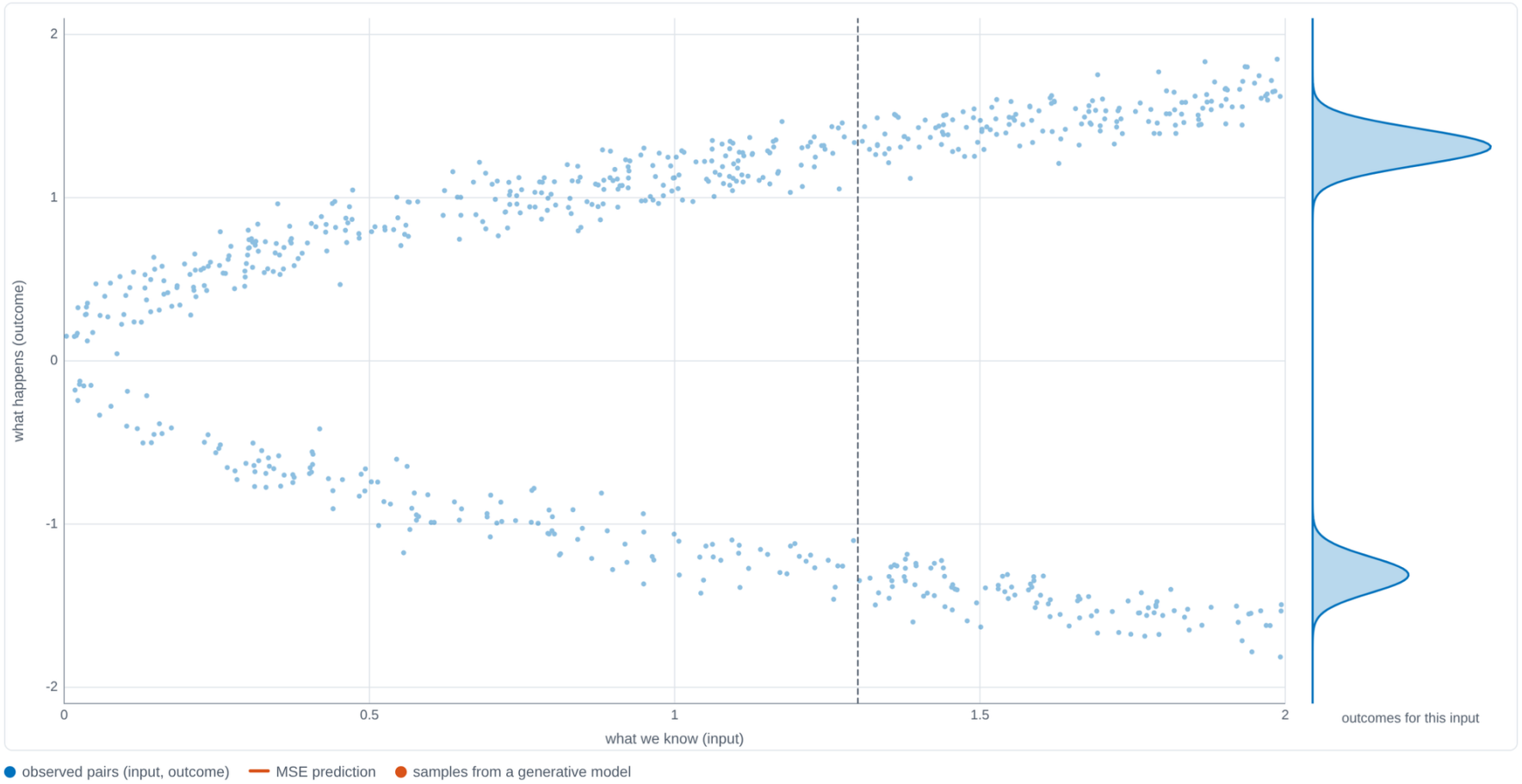
A network trained with **mean squared error** must give a single answer. The best single answer is the conditional **mean**:

```
loss = | prediction - truth |^2
best prediction = mean of truth given input
```

- The mean can be a state that never happens.
- A **generative model** returns *samples* from the target distribution

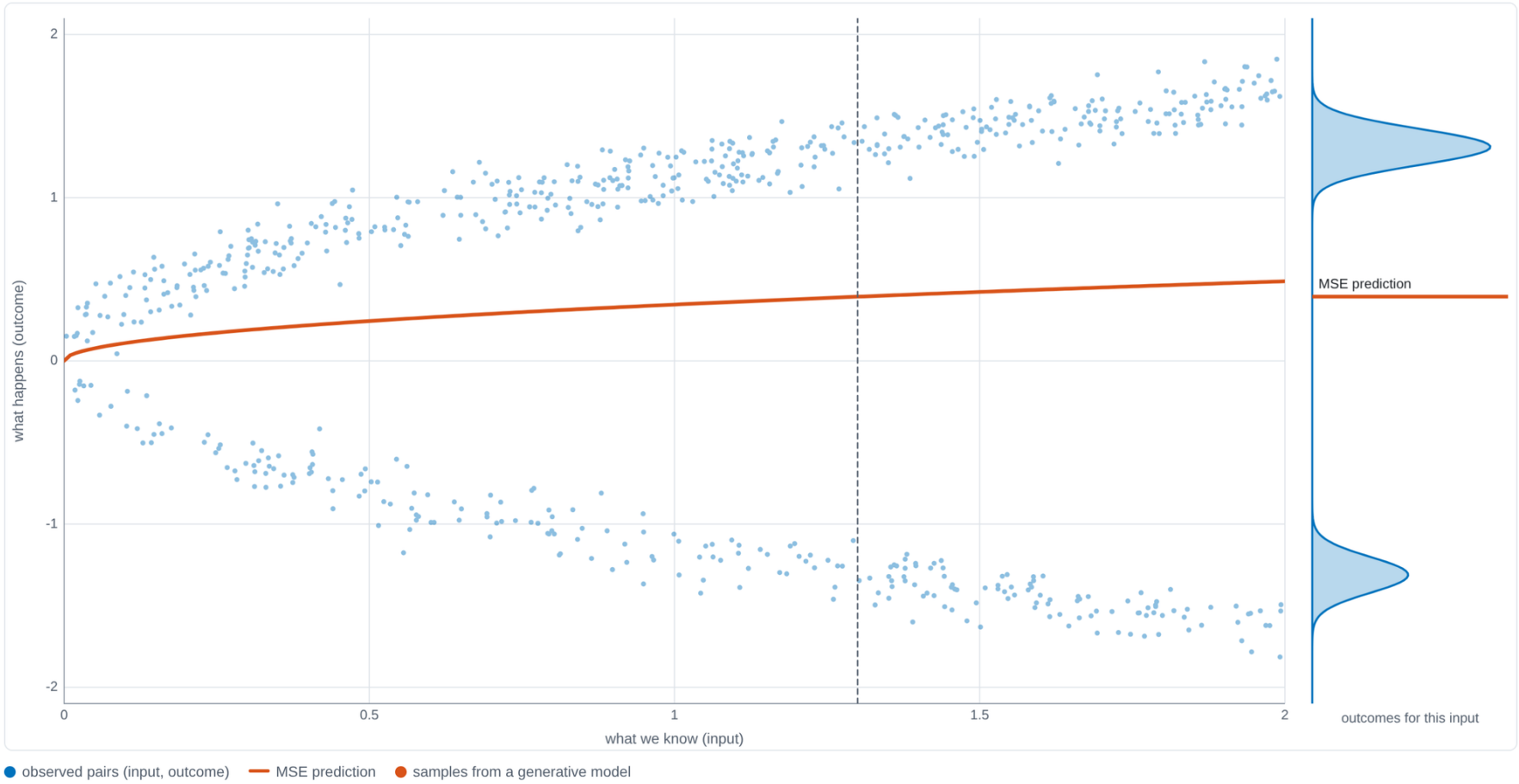


Predicting realistic samples from the target distribution



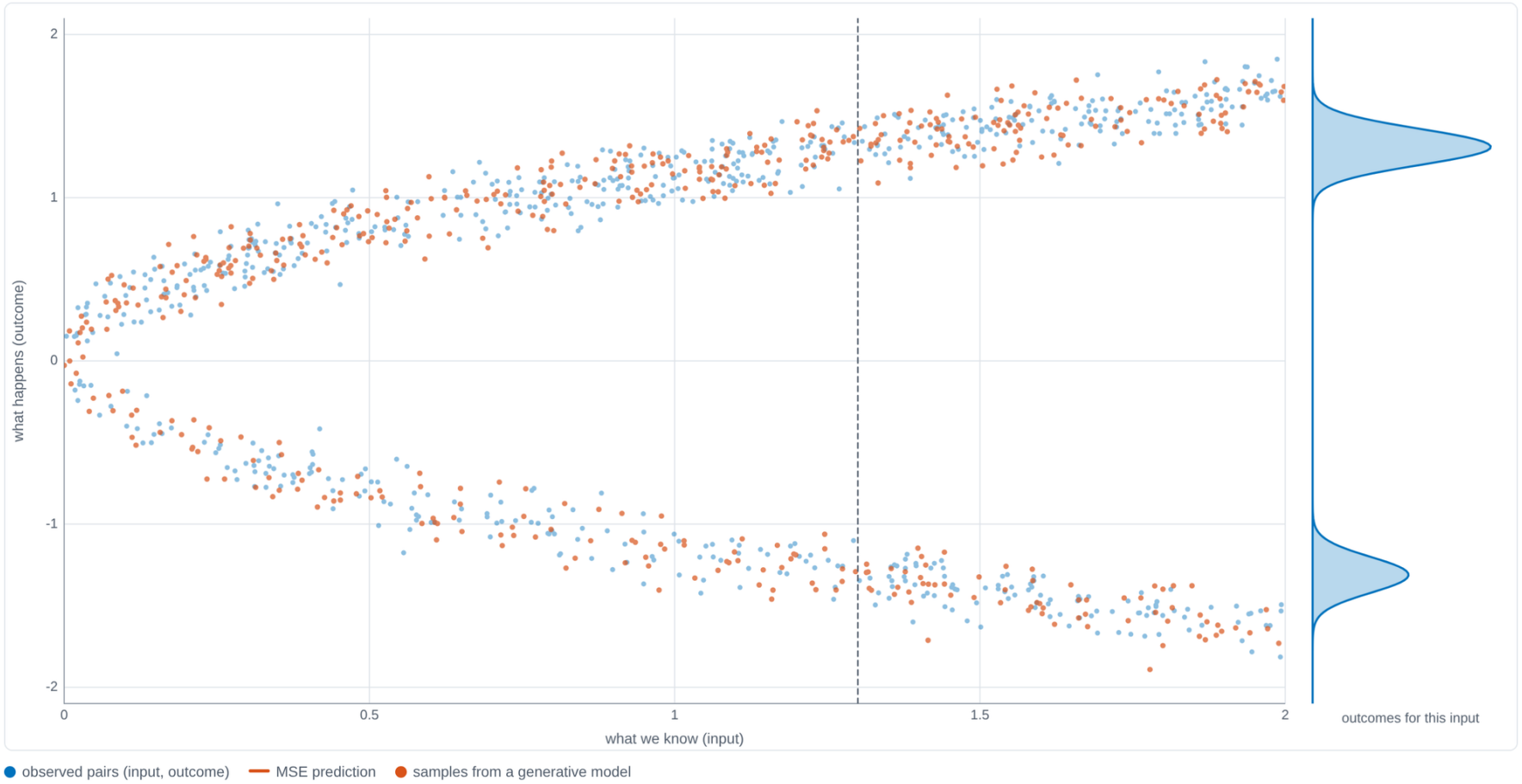
Observed pairs. For each input the outcome follows one of two branches.

Predicting realistic samples from the target distribution



The MSE prediction is the mean of the outcomes. It falls between the branches, off the attractor.

Predicting realistic samples from the target distribution

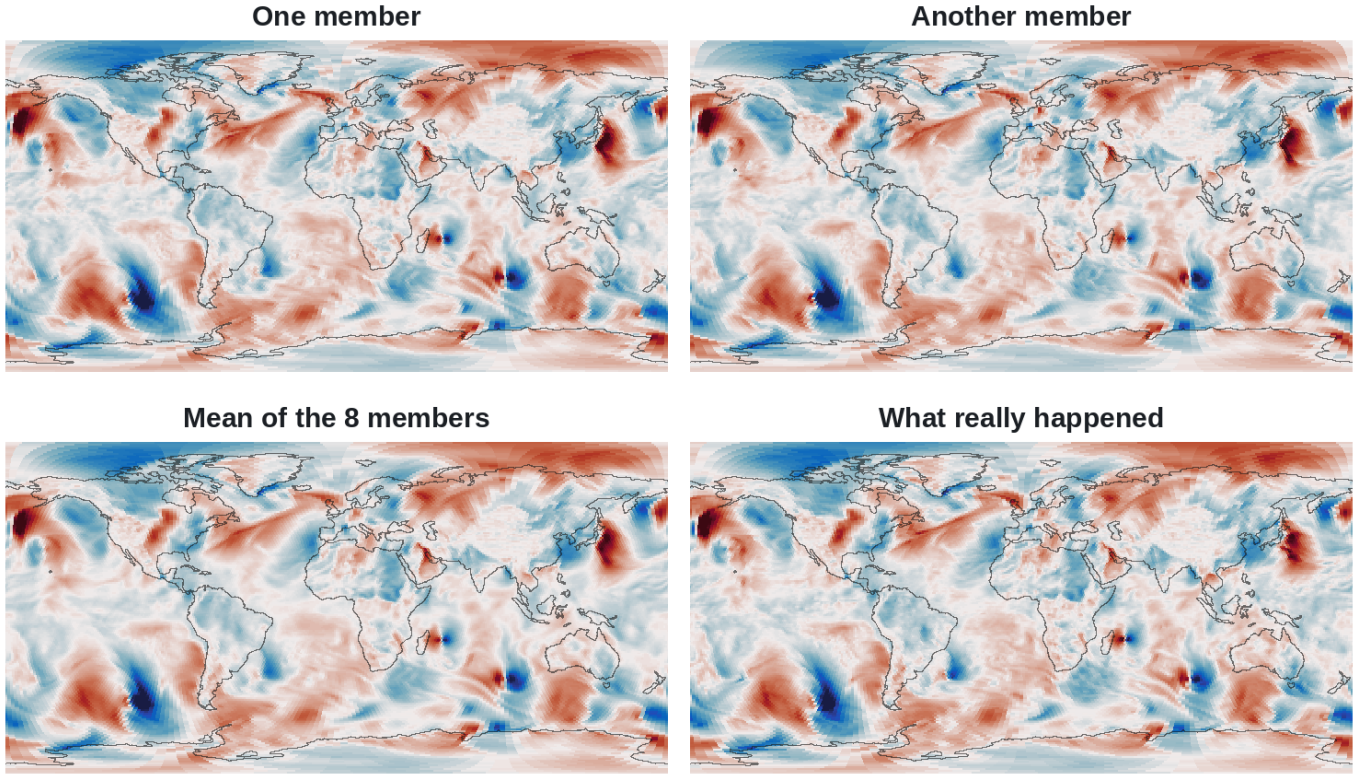


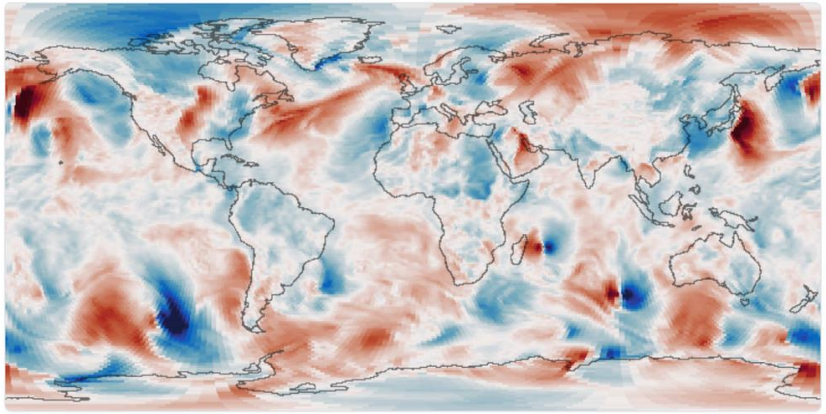
A good model draws samples from both branches.

Ensemble mean of a forecast model

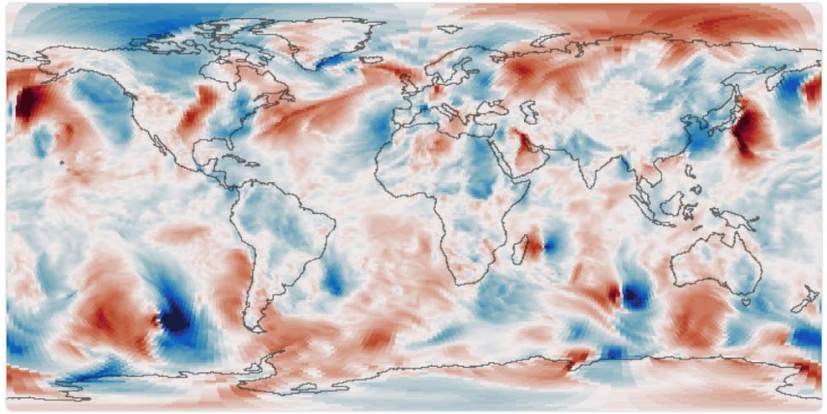
Eight forecasts from the same start (18 February 2022), at 2, 5 and 10 days lead-time, from a diffusion / flow-matching forecast model. They differ only in the random noise provided to the model.

- Each member has realistic variability, consistent with the target distribution (analysis).
- The ensemble mean has the smaller error at every lead time.
- The ensemble mean works as a **non-linear filter**: it filters out the scales according to their predictability.

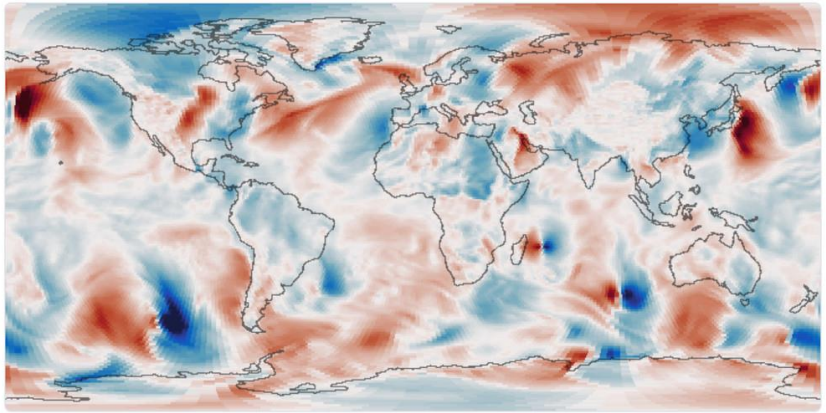




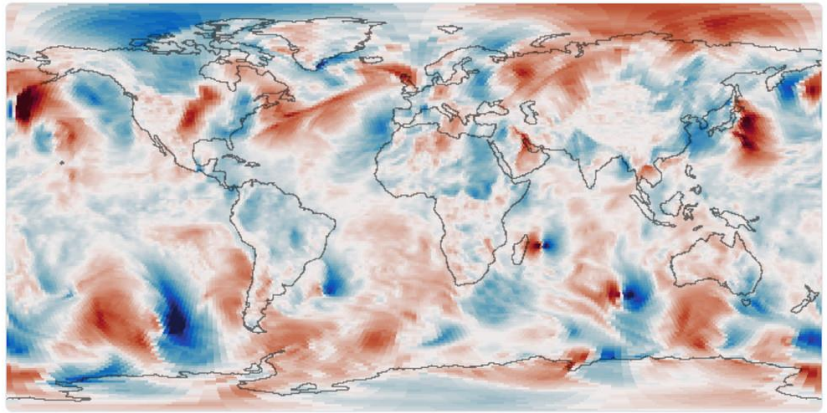
One member



Another member

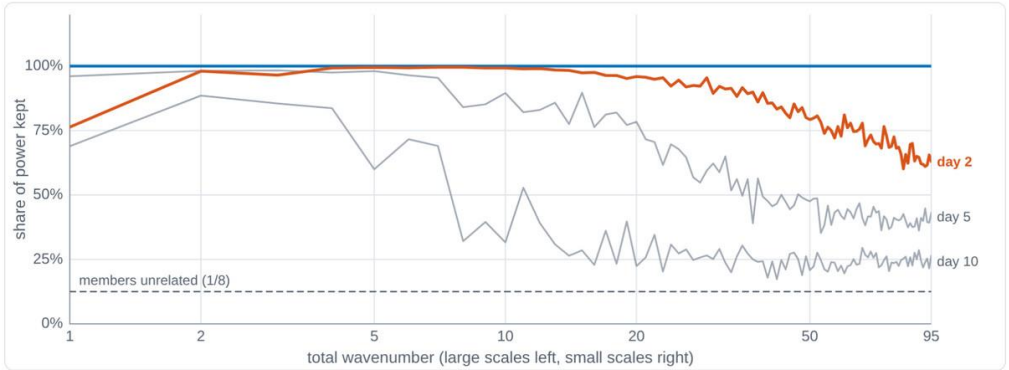


Mean of the 8 members

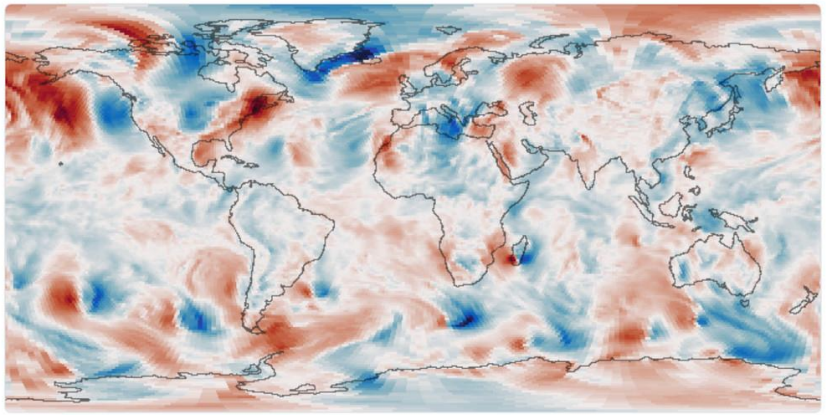


What really happened

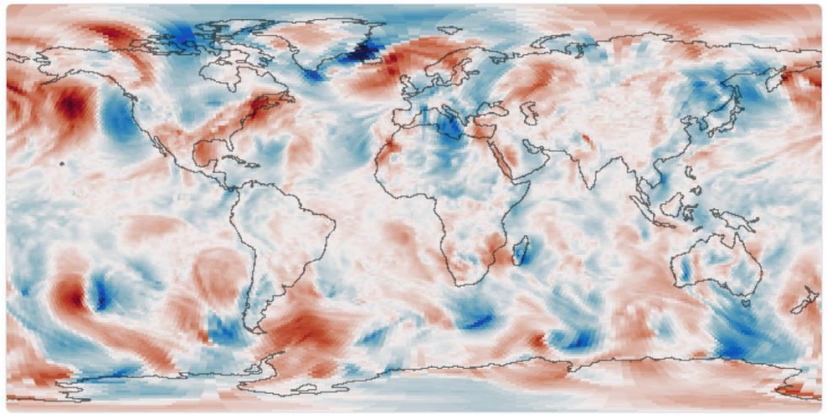
Power of the mean of 8 members, relative to a single member



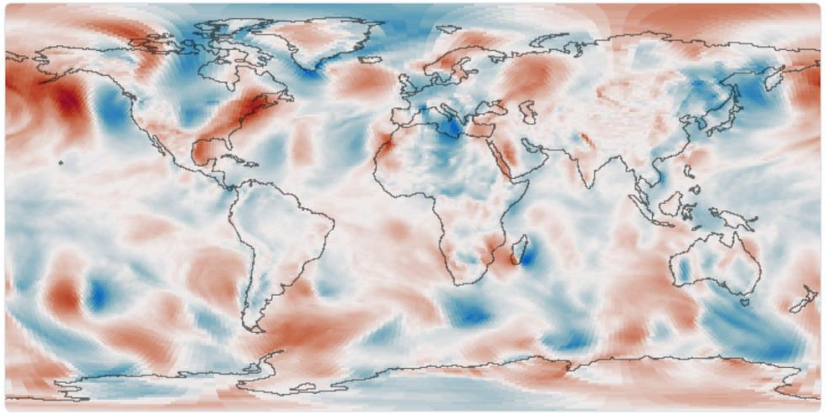
- a single member (= 1)
- mean of the 8 members, this lead time
- the other lead times



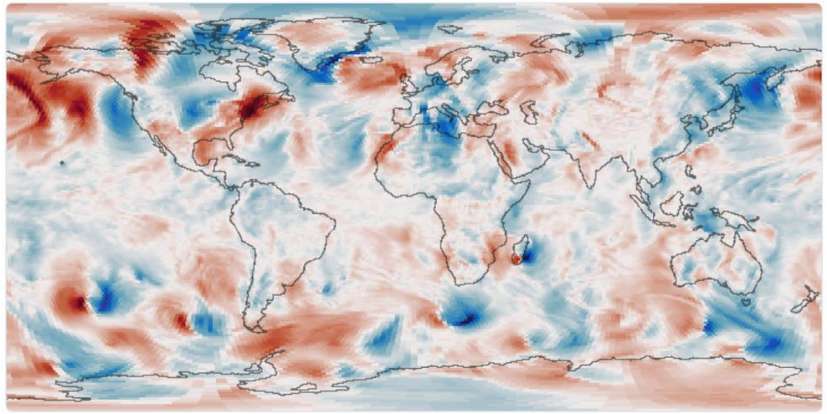
One member



Another member

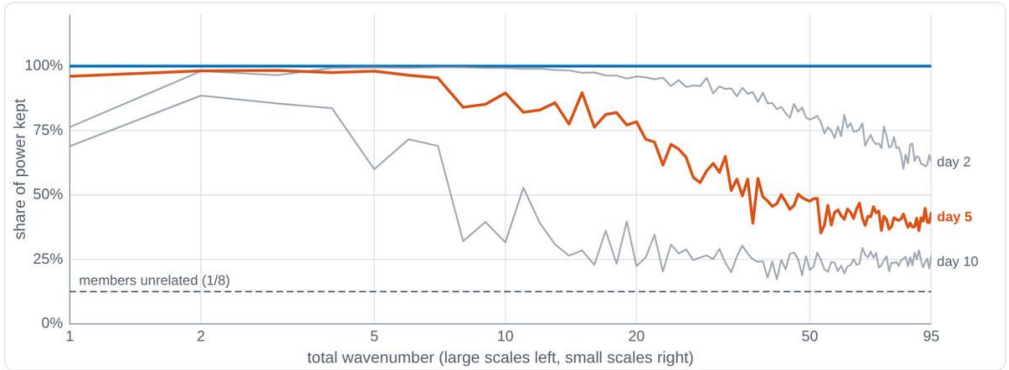


Mean of the 8 members

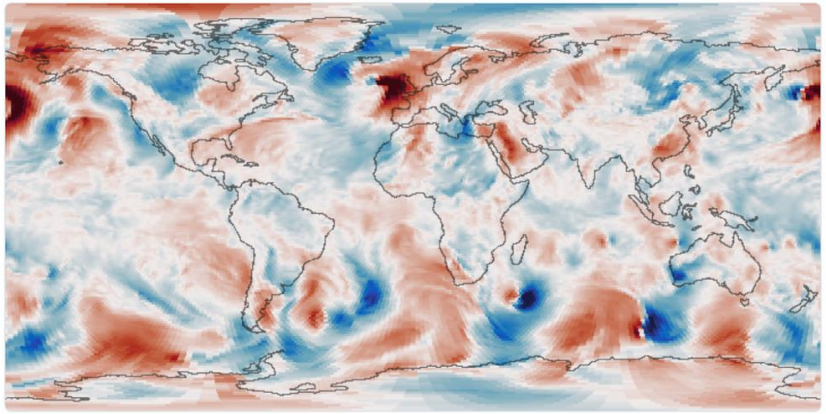


What really happened

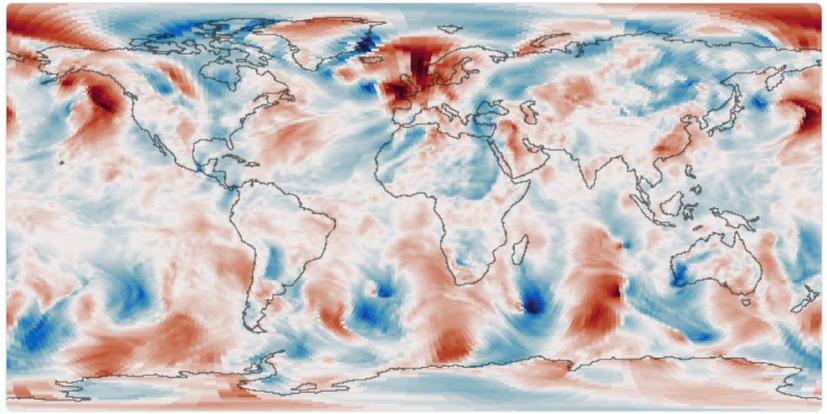
Power of the mean of 8 members, relative to a single member



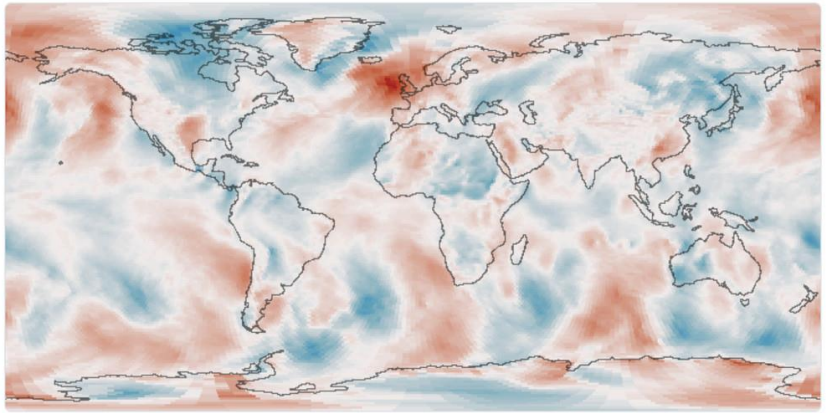
- a single member (= 1)
- mean of the 8 members, this lead time
- the other lead times



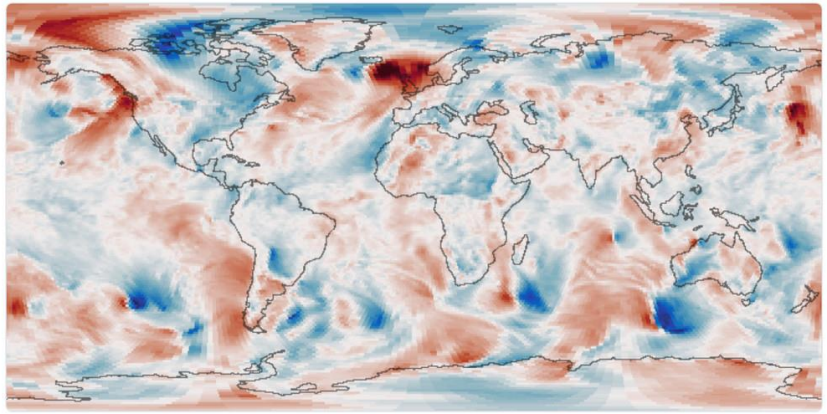
One member



Another member

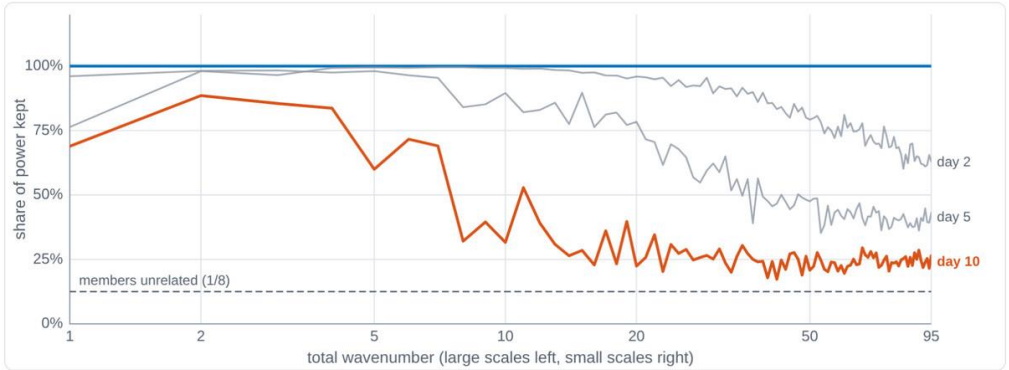


Mean of the 8 members



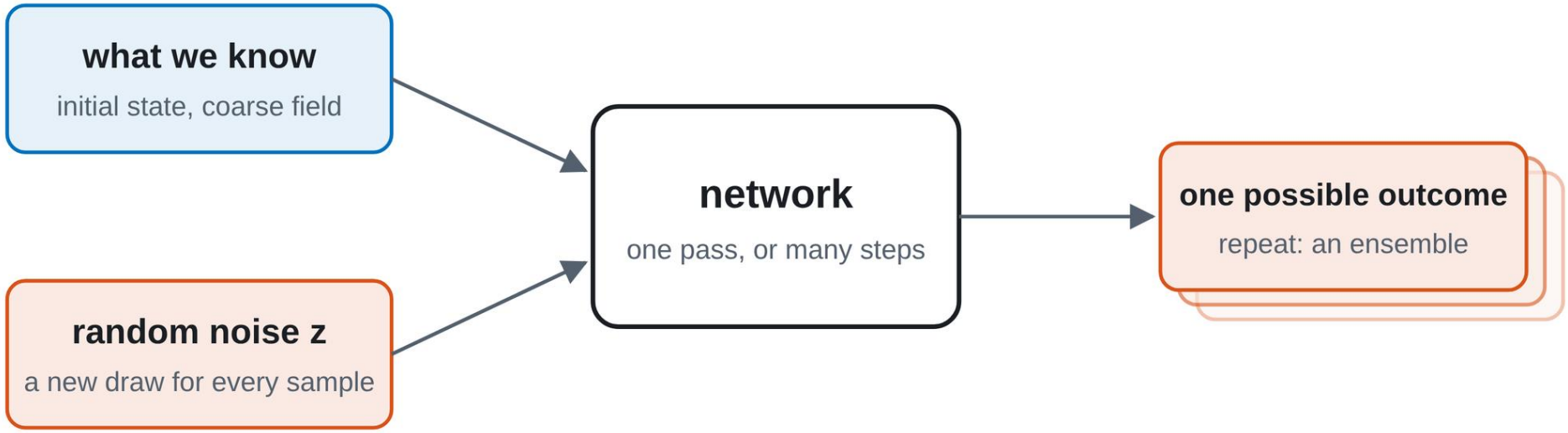
What really happened

Power of the mean of 8 members, relative to a single member



- a single member (= 1)
- mean of the 8 members, this lead time
- the other lead times

Generative model: noise to samples



VAE
compress, then decode a code

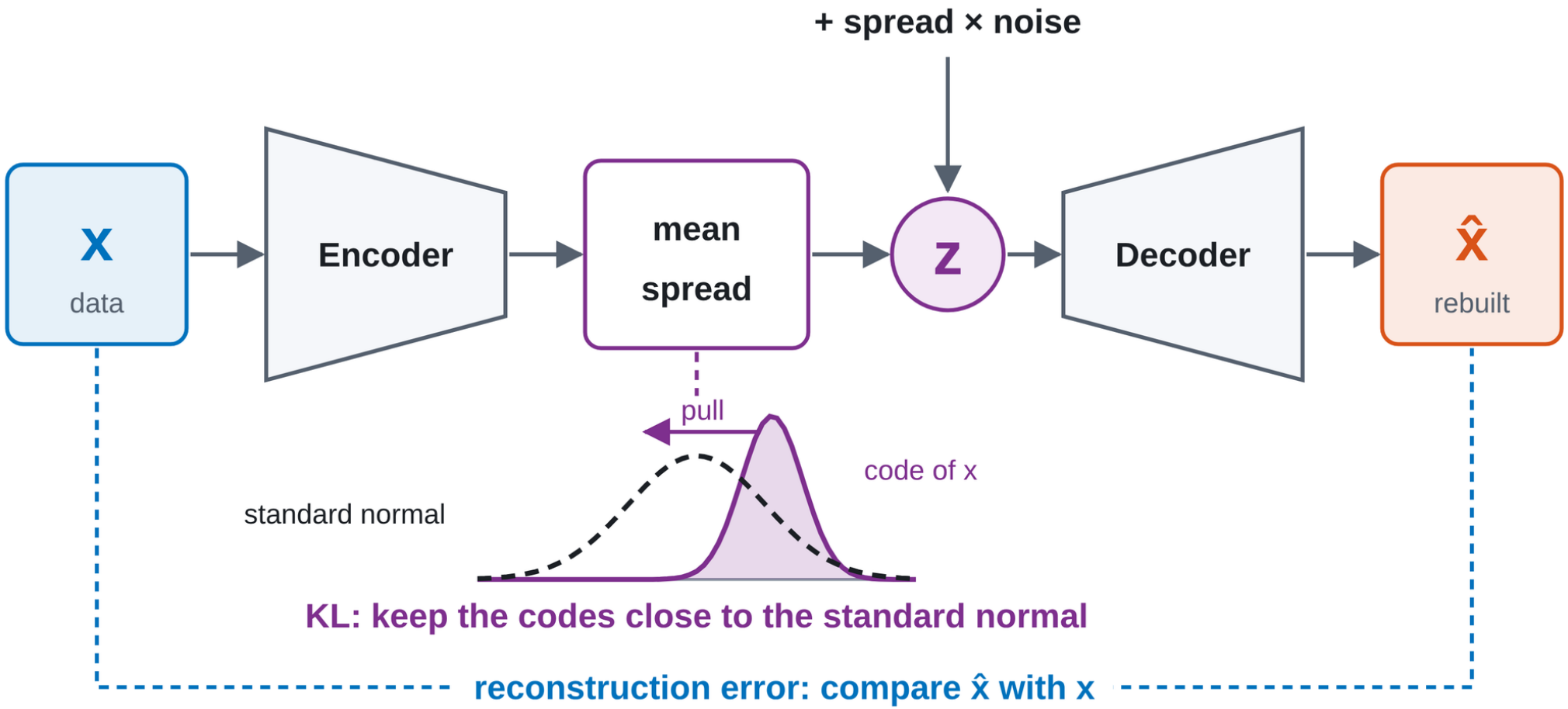
GAN
a second network judges samples

Diffusion / flow
many small denoising steps

Proper scores
score the ensemble directly

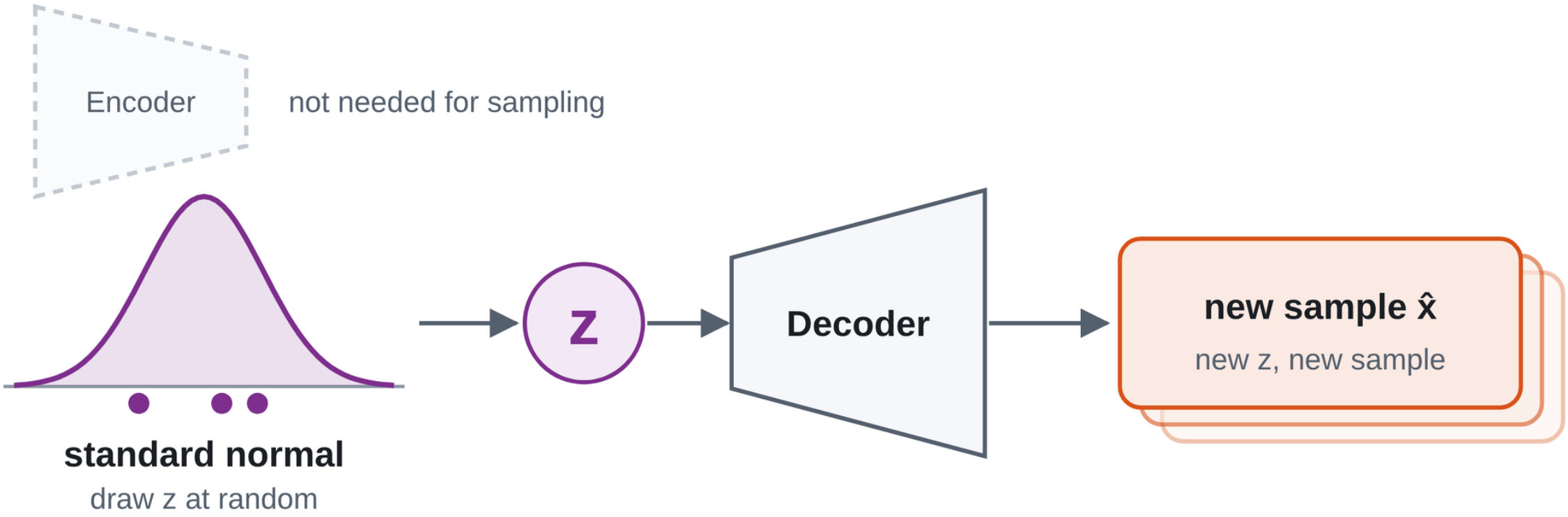
We only have samples from the target distribution (reanalyses, analyses, forecast fields, observations)

VAE: encoder, latent code, decoder



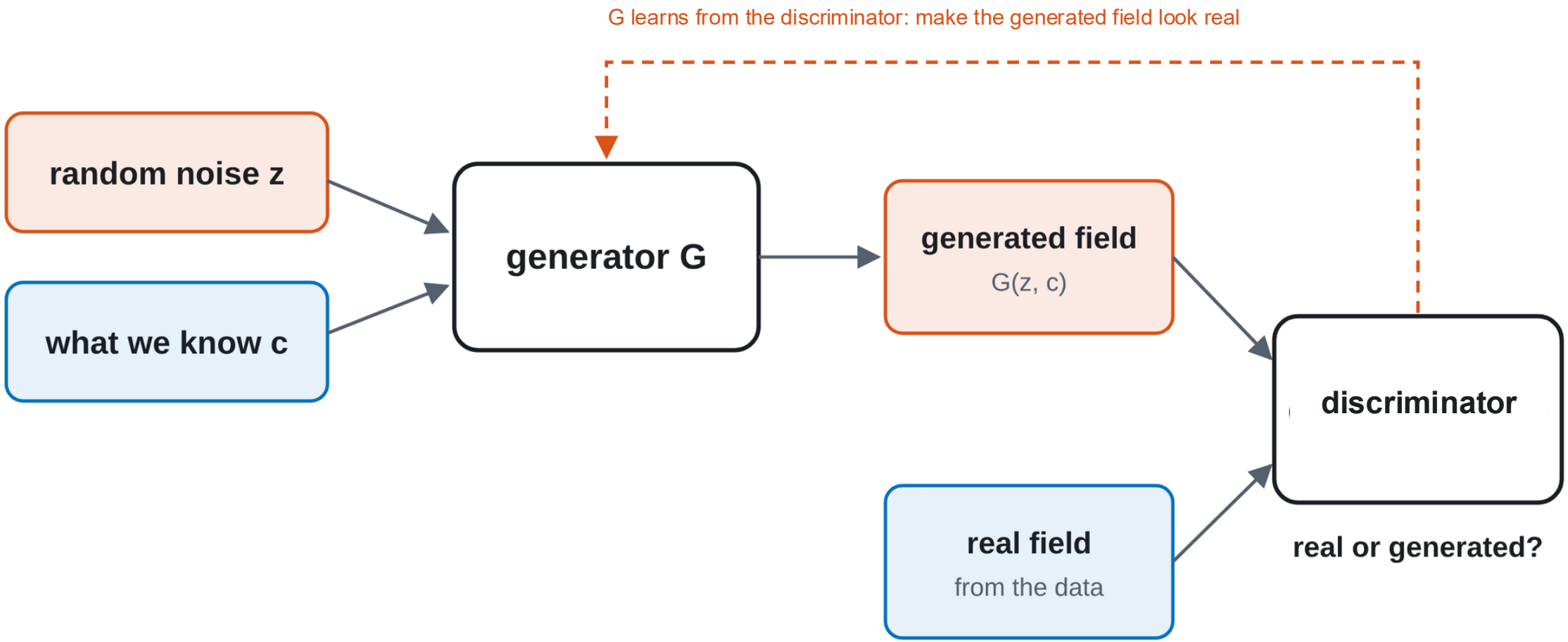
Training: encoder, code with mean and spread, decoder. The KL term keeps the codes close to the standard normal; the reconstruction error compares x-hat with x.

VAE: generating new samples



one network pass per sample; repeat for an ensemble

Generative adversarial networks



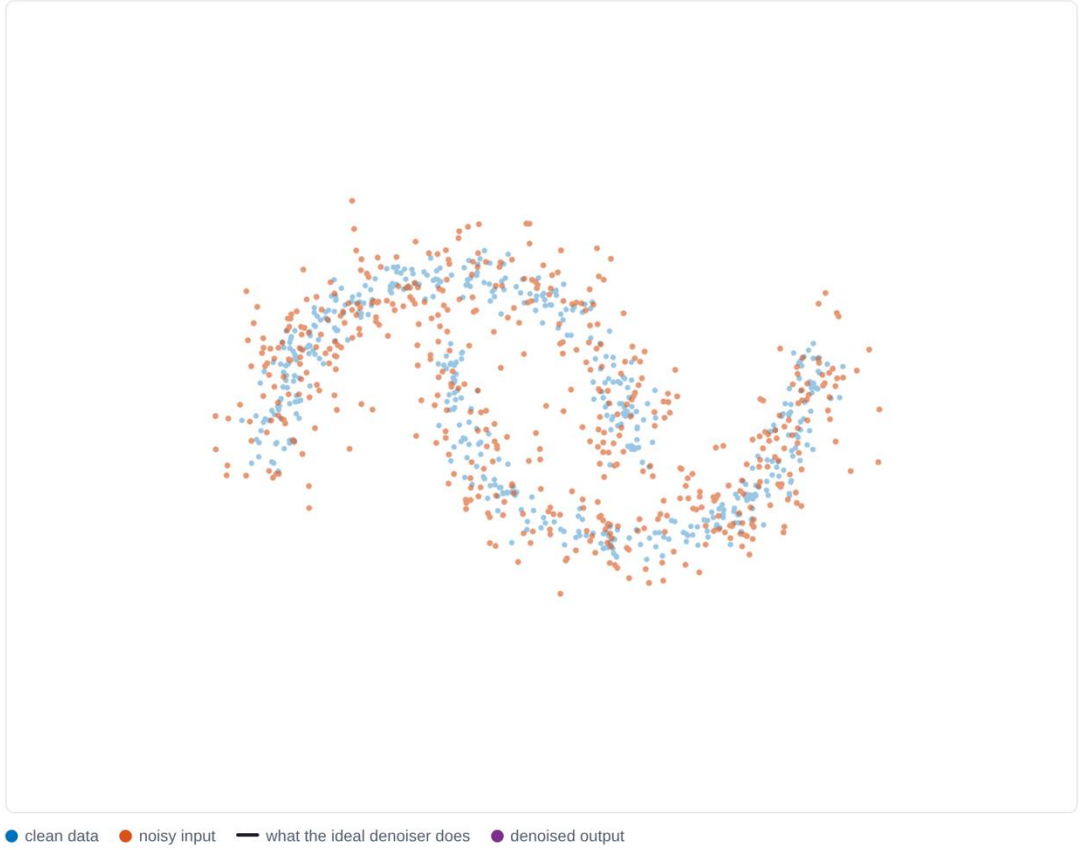
The generator turns noise and what we know into a field; the discriminator tells real from generated; the generator learns from the discriminator.

Denoising auto-encoder

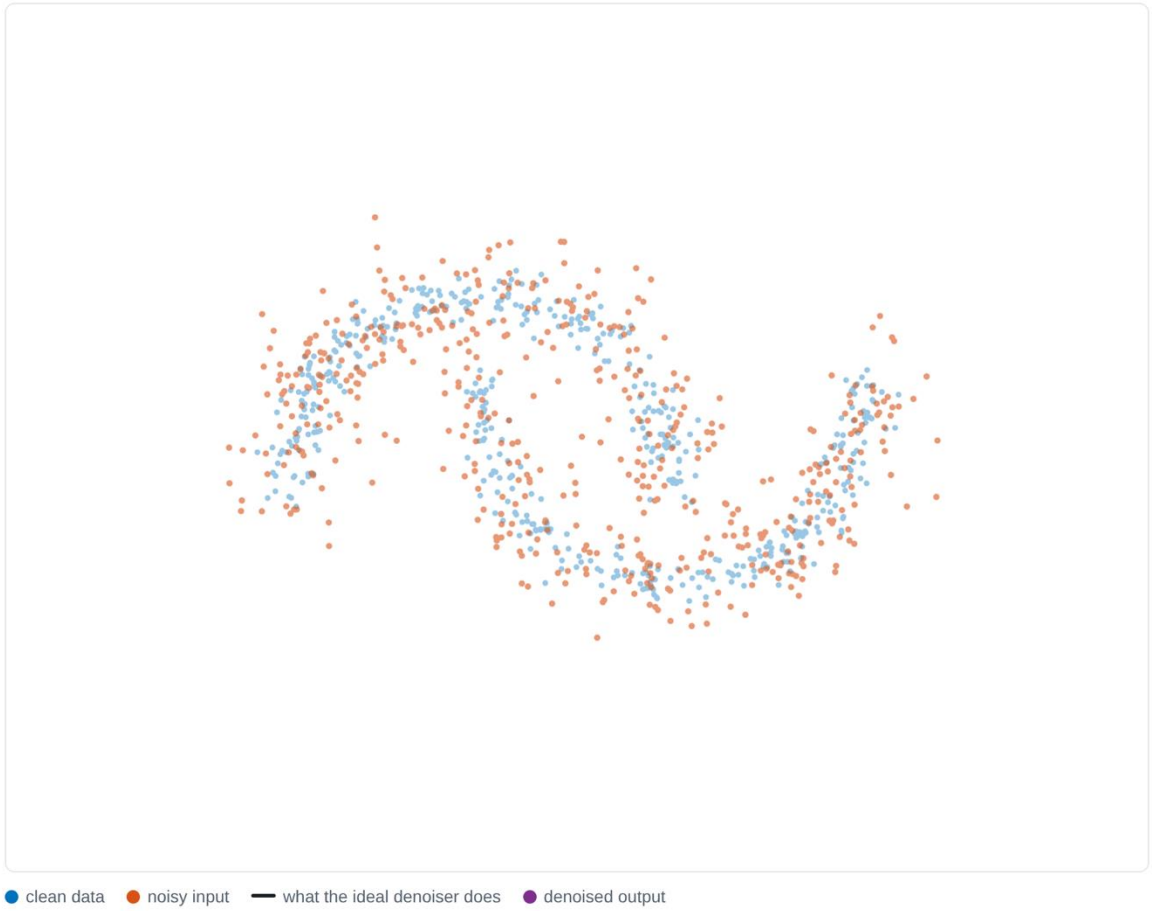
Remove noise from data samples:

```
noisy = clean + sigma * noise
loss  = | D(noisy, sigma) - clean |^2
```

- **Small noise:** each point is pushed back onto the data. To do that the network must know where the data is.
- **Large noise:** the best repair is again an average. One big step gives the blurry answer.

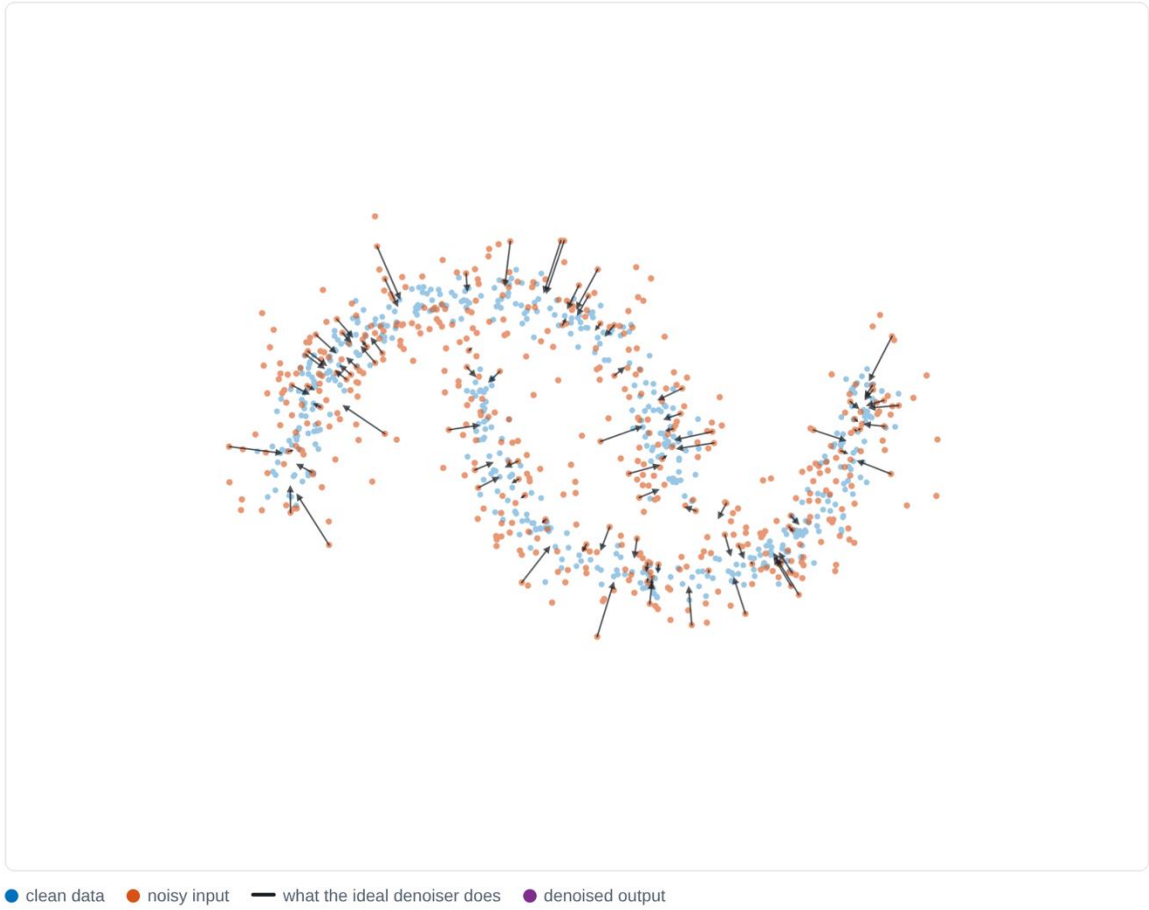


Denoising auto-encoder



Corrupt: noise moves the clean points around and between the moons.

Denoising auto-encoder



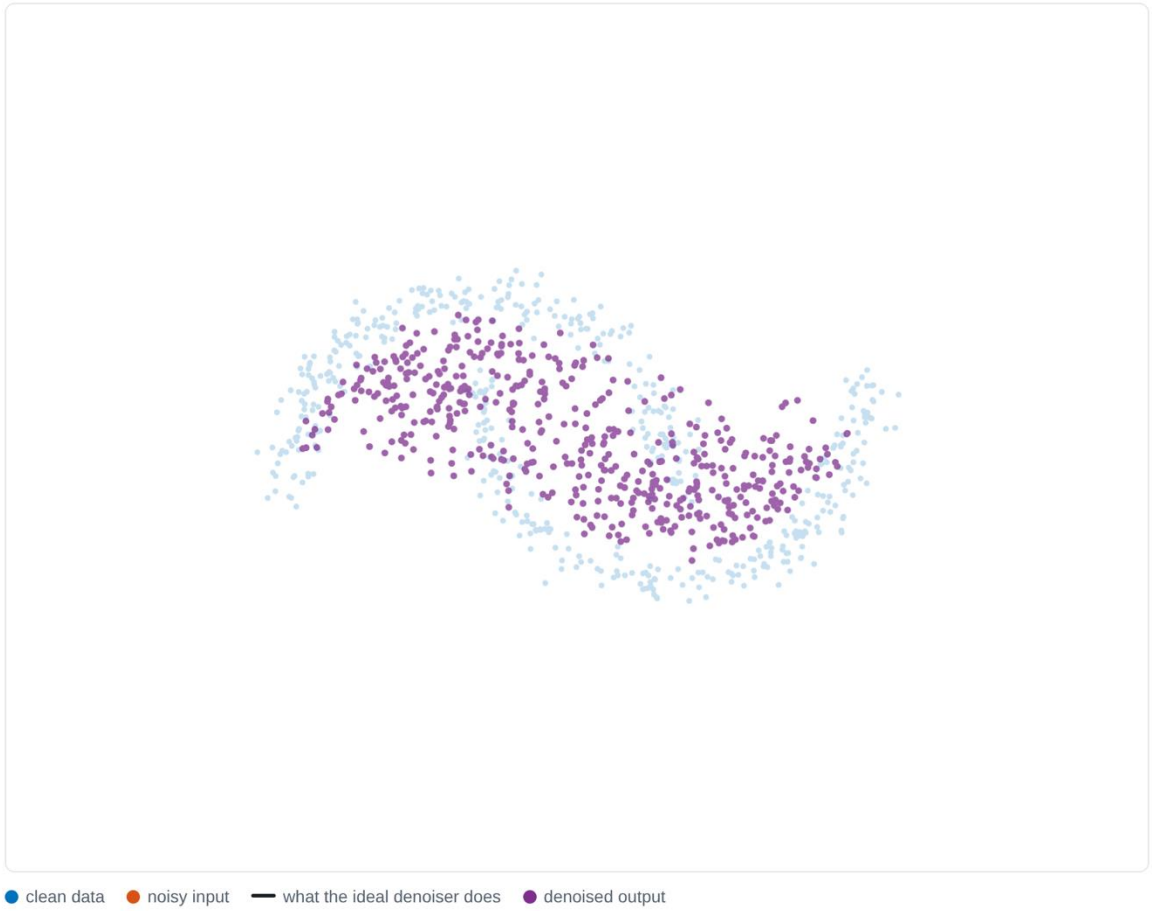
Repair: the best denoiser points each noisy point back towards the data.

Denoising auto-encoder



Small noise: almost all denoised points lie on the moons.

Denoising auto-encoder



Large noise: the denoised points collapse towards the middle, the blurry average.

Denoiser and score

The score is the direction in which a state becomes more likely.

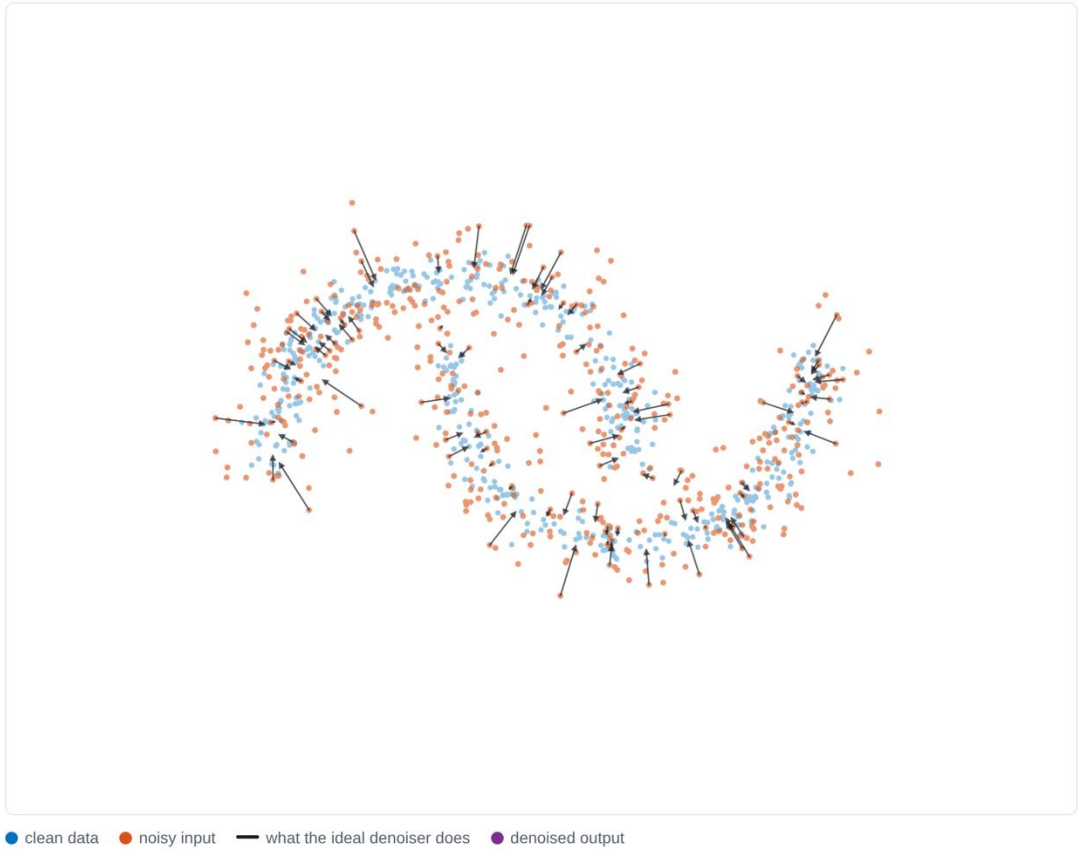
```
score(x) = gradient of log p(x)
```

Why the score? The probability p(x) of a weather state cannot be computed. The score can be learned from samples

How to learn it? A denoiser D trained with plain MSE can learn it. Its correction to a noisy state x is the score, scaled by the noise level:

```
D(x) - x = sigma^2 * score(x)
```

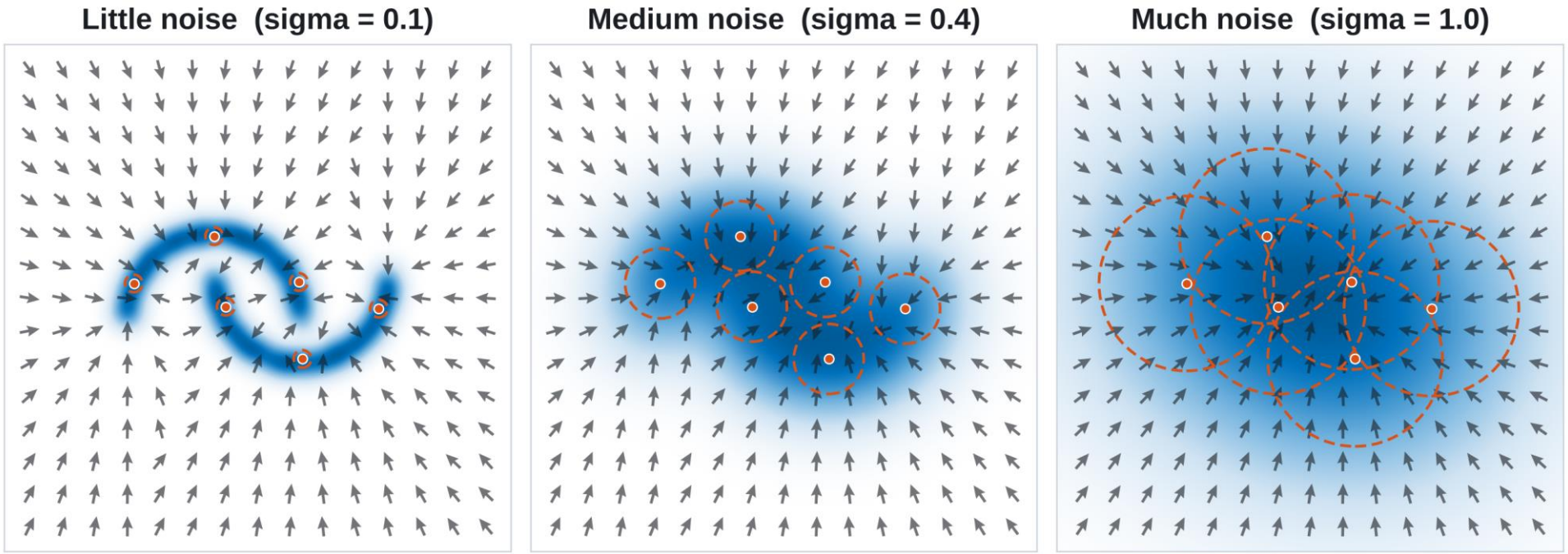
Diffusion models are built on this principle: learn to denoise, then follow the score from noise to data.



The denoiser's corrections point from each noisy point towards the data.

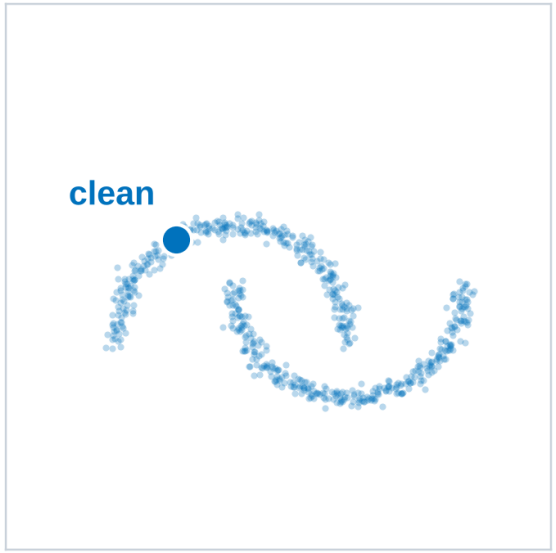
Denoising score matching minimises the expected Hyvärinen score of the noisy data, a proper scoring rule.

Score at different noise levels

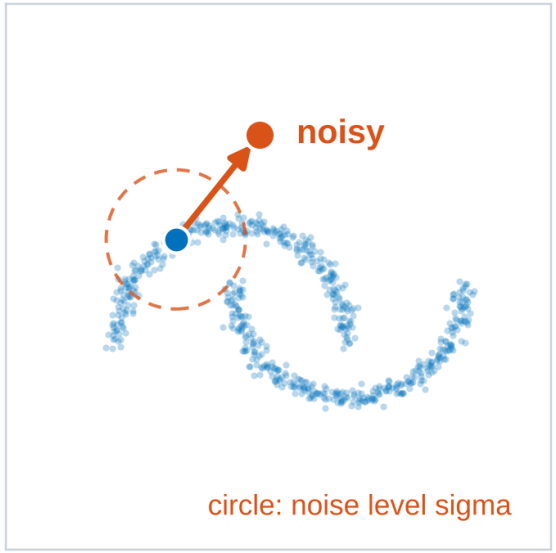


Training a diffusion model by predicting clean samples

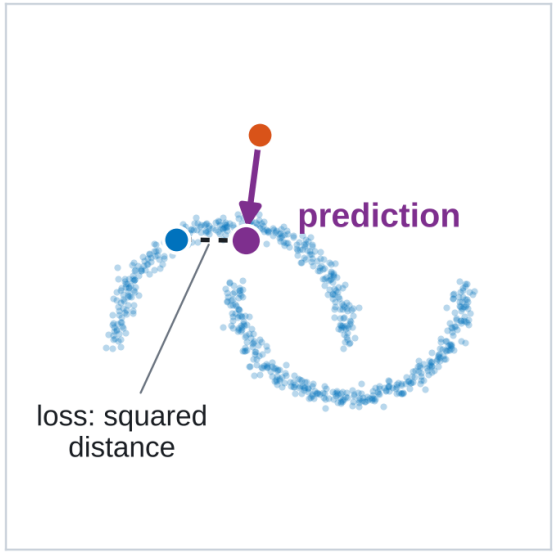
1 Pick a training example



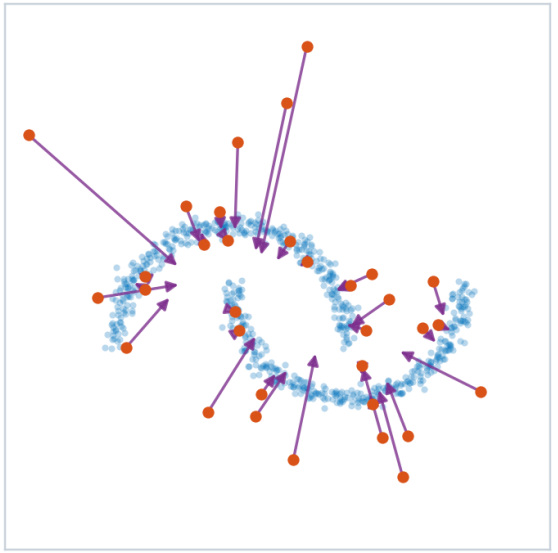
2 Add noise



3 Network predicts clean



4 Repeat many times

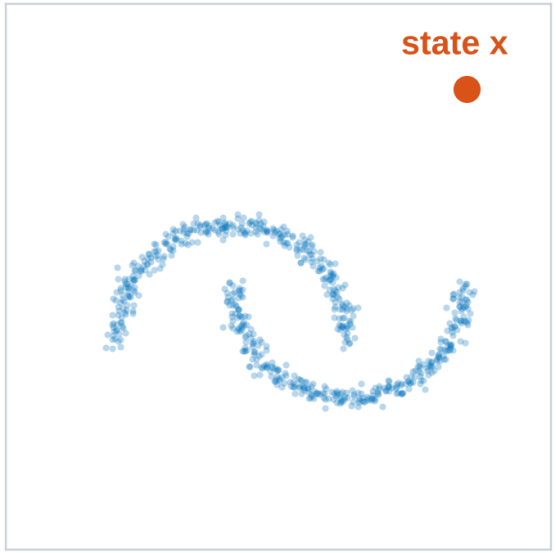


```
repeat:
    clean  = a random training example
    sigma  = a random noise level
    noisy  = clean + sigma * noise
    loss   = | D(noisy, sigma) - clean |^2
    update the network D to reduce the loss
```

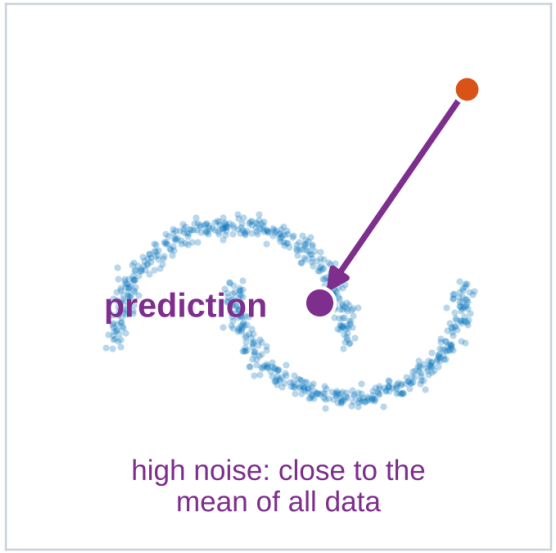
Or predict noise or combination of noise and clean data ...

Sampling from a diffusion model

1 Start from pure noise

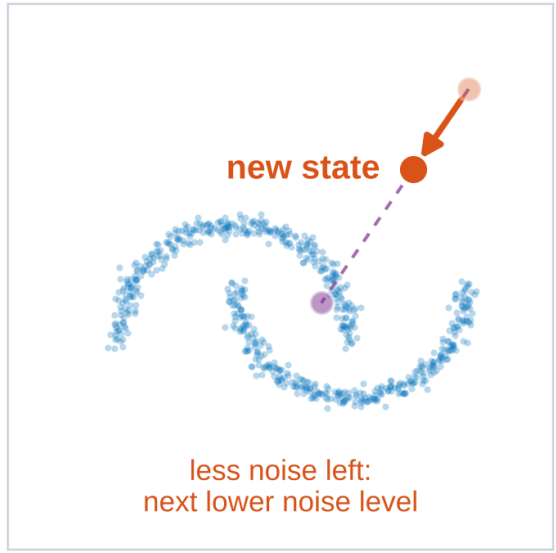


2 Network predicts clean



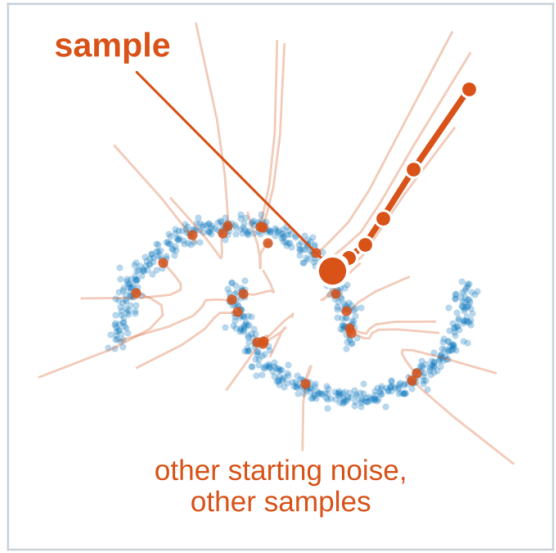
high noise: close to the mean of all data

3 Move part of the way



less noise left: next lower noise level

4 Repeat down to zero noise



other starting noise, other samples

```
x = pure noise
repeat, from high to low noise level:
    x_clean = D(x, sigma)
    x = part of the way from x to x_clean
return x
```

Sampling with the Euler method

Sampling means solving a differential equation: how the noisy state x changes as the noise level sigma drops. This is the **probability flow ODE**.

```
dx / dsigma = (x - D(x, sigma)) / sigma
            = - sigma * score(x)
```

Euler method: take the slope at the current point and follow it in a straight line to the next noise level.

```
x = sigma_max * noise
for each noise level, from high to low:
    x_clean = D(x, sigma)
    x = x_clean + (sigma_next / sigma)
                * (x - x_clean)
```

Keep the clean prediction, and only the fraction sigma_next / sigma of the noise. One network call per step.

- **Deterministic:** the only randomness is the starting noise. Same noise, same sample; new noise, new ensemble member.
- **One step** from pure noise returns the clean prediction: the blurry mean.
- **More steps,** smaller error: better follows the curved path.
- **Heun method:** an Euler step, then the slope again at the end point, and the average of the two. Two network calls per step, but fewer steps.

Flow matching: the same with the learned velocity v, in steps of dt from noise (t = 0) to data (t = 1):

```
x_next = x_t + dt * v(x_t, t)
```

The sampler is separate from the network: train once, then choose the solver and the number of steps.

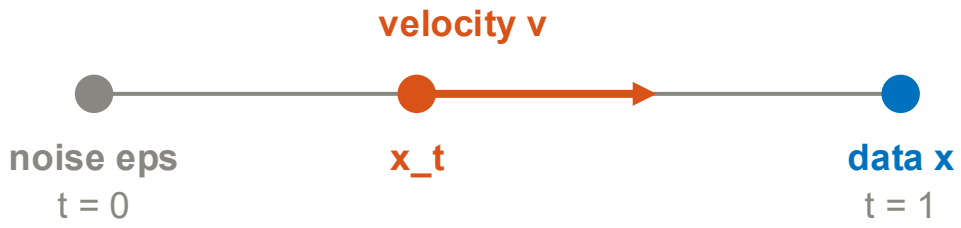
Flow matching

One training example is a straight line. Take a real field x and a Gaussian noise field eps, and mix them:

```
x_t = t * x + (1 - t) * eps
t = 0: noise,  t = 1: data
```

The target is the velocity. Along the line the field changes at a constant rate:

```
v = x - eps
```



Closely related to diffusion. The same idea with different mixing weights. With Gaussian noise, velocity and score can be converted into each other. But also allows for more general cases.

The network can predict x, noise or velocity

We have different options as prediction targets:

- clean state, x
- noise, eps
- velocity, v

They give the same information. Given x_t, any one of them can be transformed to the other two:

```
x     = x_t + (1 - t) * v
eps   = x_t - t * v
score = - eps / (1 - t)
```

Does it matter? Each target weights the noise levels differently in the loss.

A network has limited capacity, hence the weighting changes what the network learns

Noise schedule and weighting

Training: noise schedule and weighting. Each training example gets a random noise level. Here, three choices influence training:

- **Noise schedule:** how often each noise level is drawn during training.
- **Weighting per noise level:** how much the error at each level counts in the loss.
- The prediction target itself influences the weighting. Written as an error in x:

```
|v - v_pred|^2     = |x - x_pred|^2 / (1 - t)^2
|eps - eps_pred|^2 = |x - x_pred|^2 * t^2 / (1 - t)^2

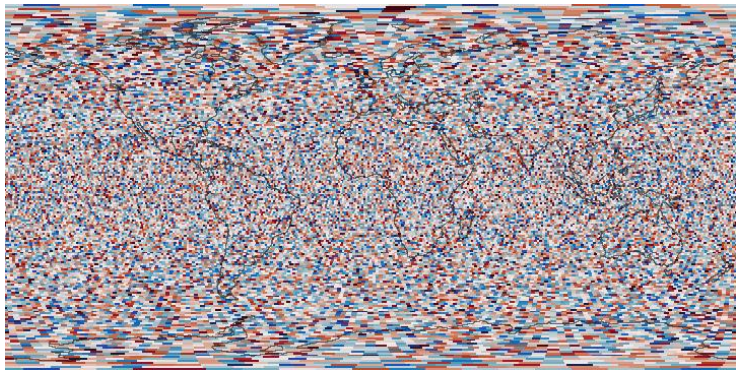
t^2 / (1 - t)^2 = signal-to-noise ratio
```

- High noise levels can influence the large scales, low noise levels the small scales. Schedule and weighting therefore decide how much model capacity is spent at which scales.
- Sampling has its own schedule.

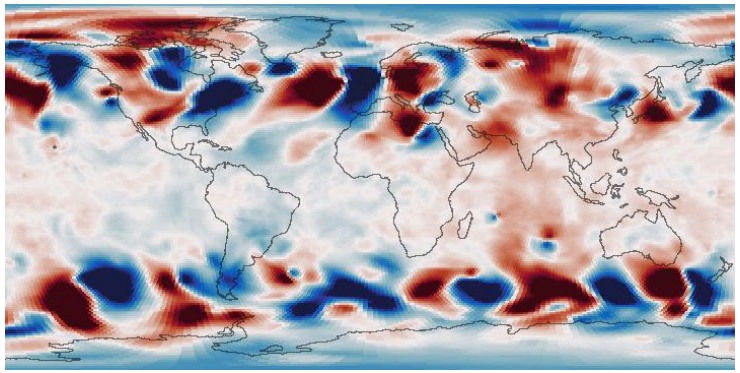
Note: higher resolution / more points can make de-noising easier and this can need a shift to higher noise levels during training

Sampling loop of a flow-matching forecast model

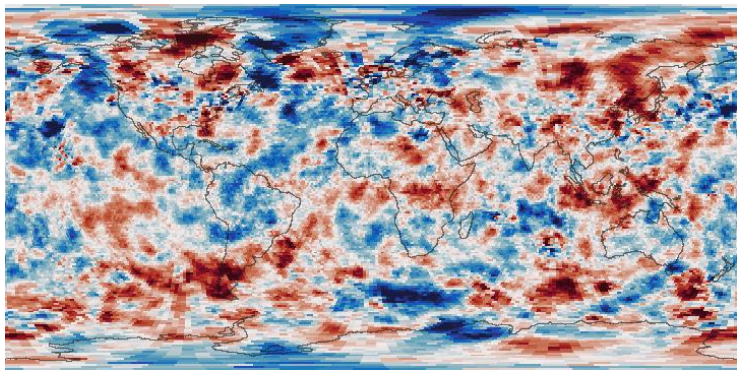
Noisy state



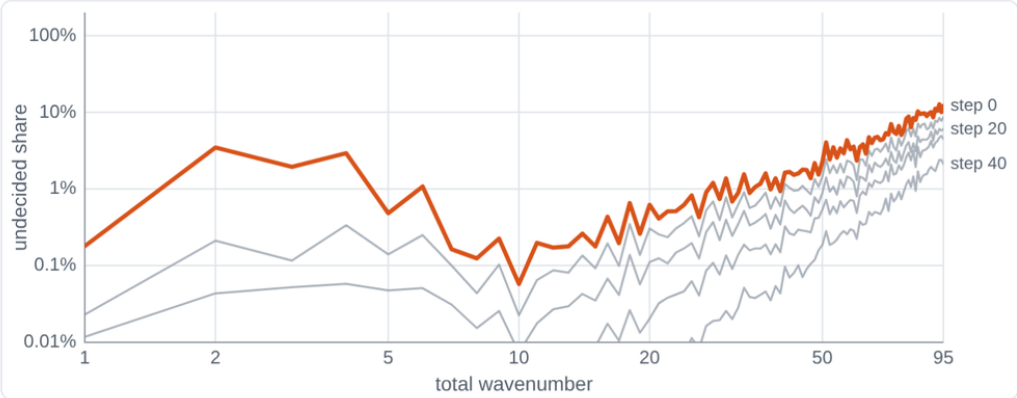
Clean prediction at this step



Prediction − final sample (amplified)

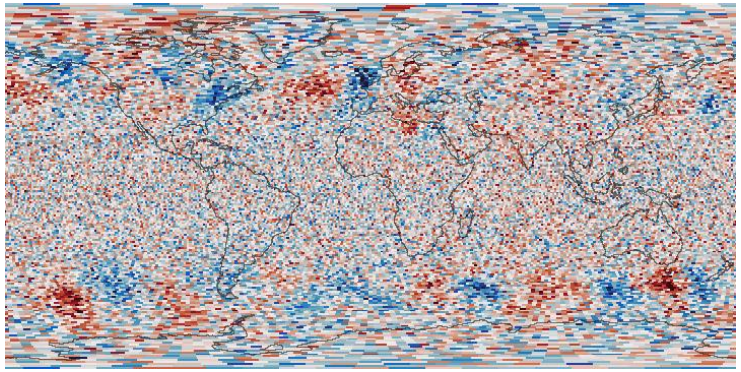


Clean prediction: share still undecided, this step and steps 0 to 40

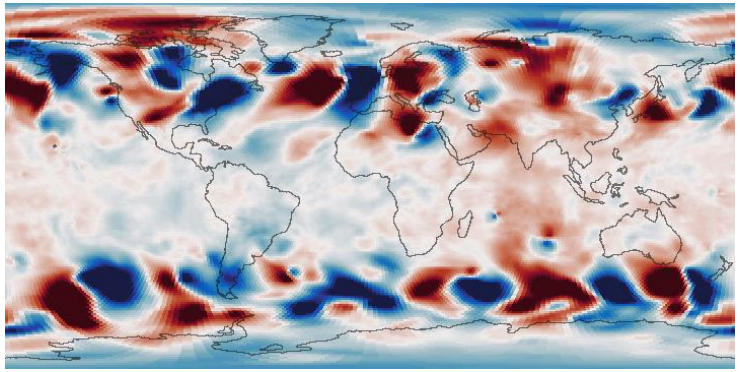


Sampling loop of a flow-matching forecast model

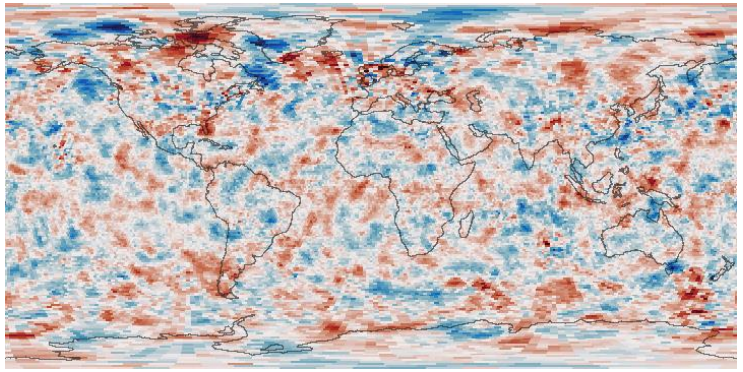
Noisy state



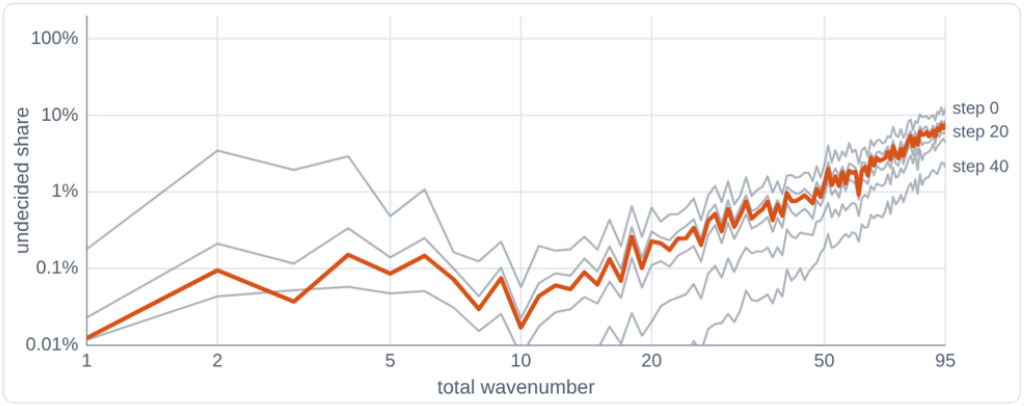
Clean prediction at this step



Prediction − final sample (amplified)

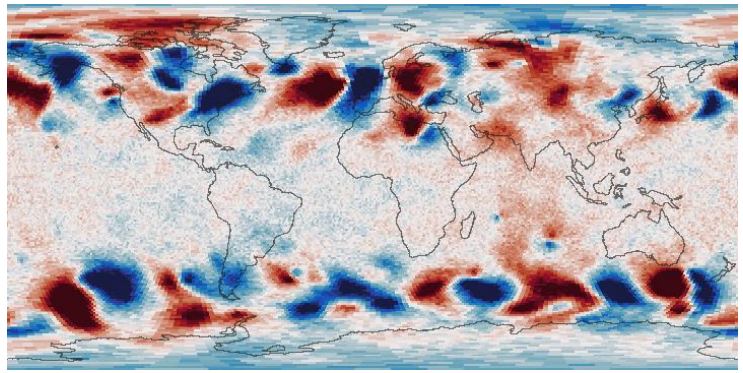


Clean prediction: share still undecided, this step and steps 0 to 40

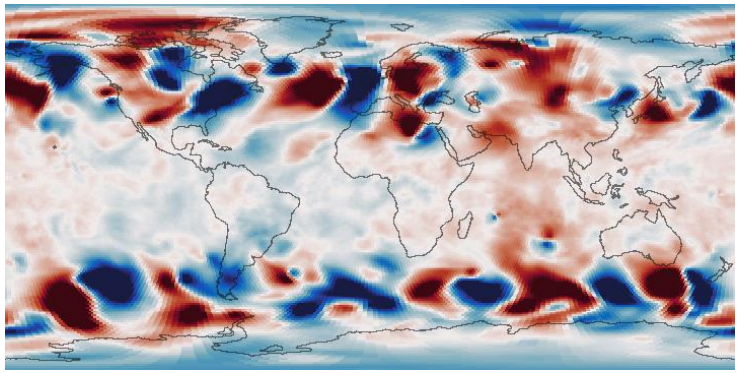


Sampling loop of a flow-matching forecast model

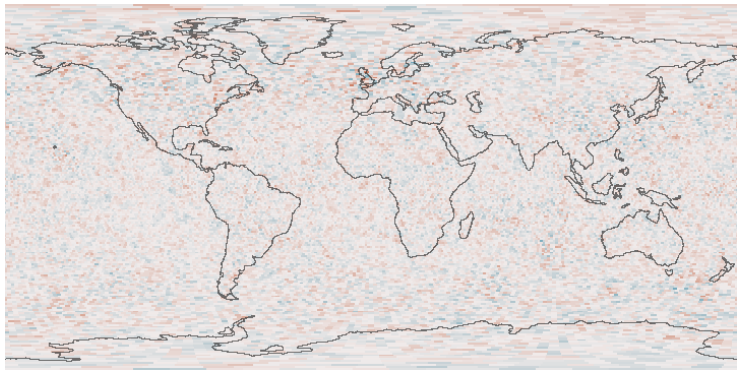
Noisy state



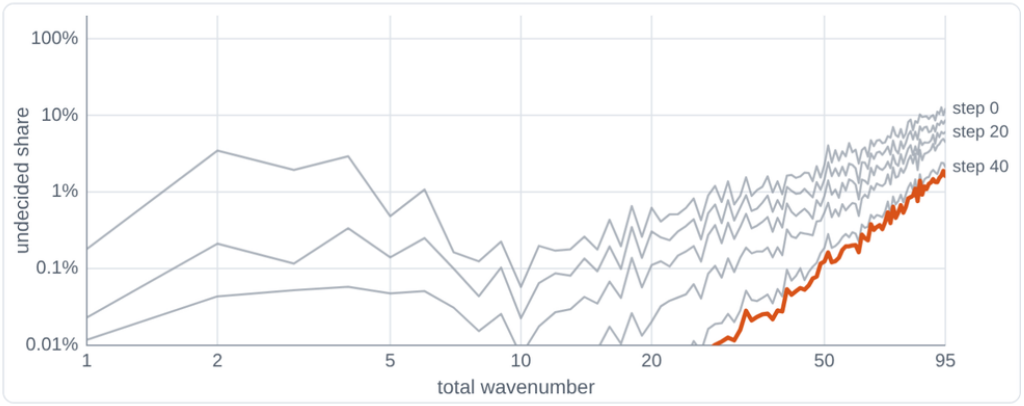
Clean prediction at this step



Prediction − final sample (amplified)

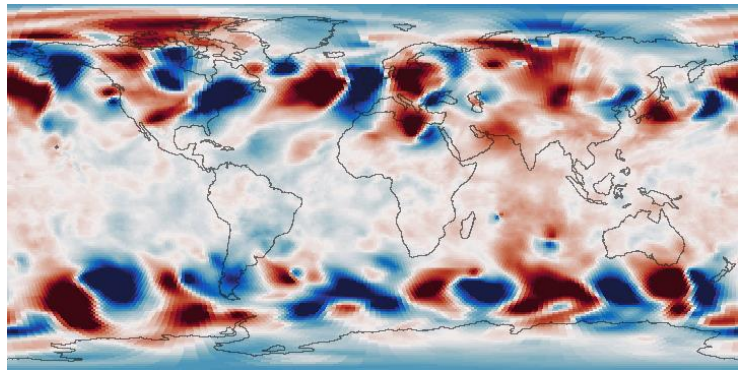


Clean prediction: share still undecided, this step and steps 0 to 40

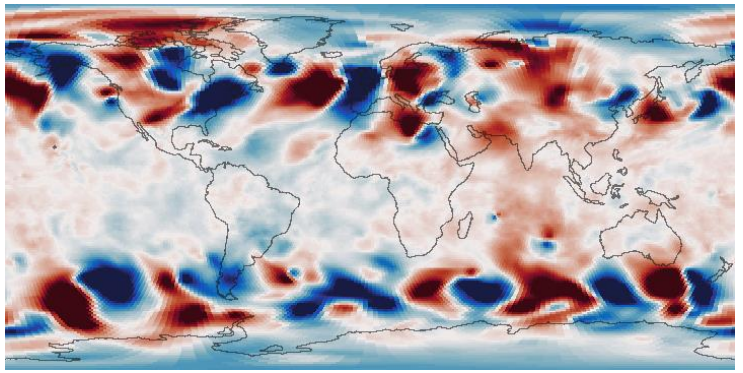


Sampling loop of a flow-matching forecast model

Noisy state

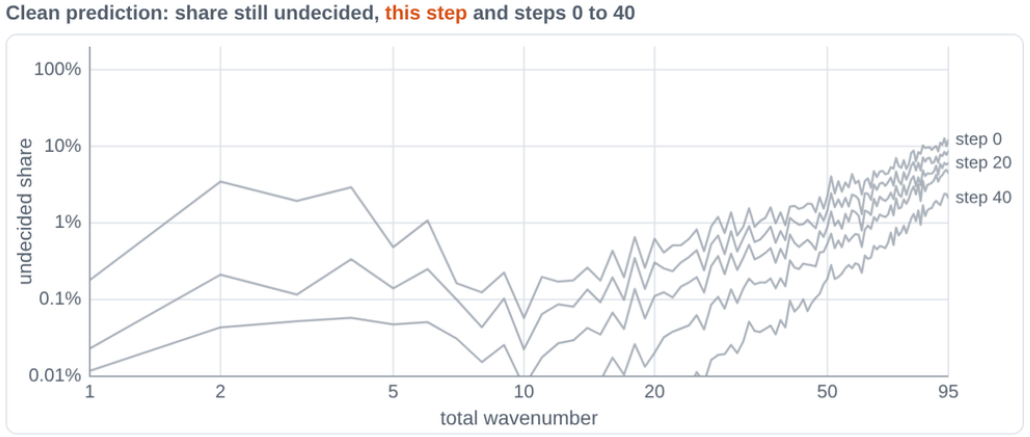


Clean prediction at this step



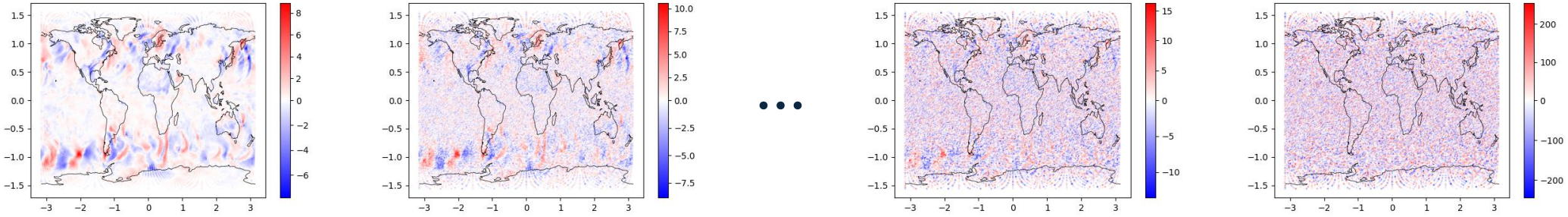
Prediction − final sample (amplified)





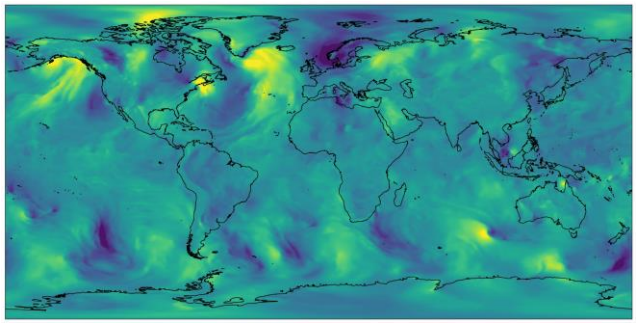
Diffusion for weather forecasting

Forward process: add noise

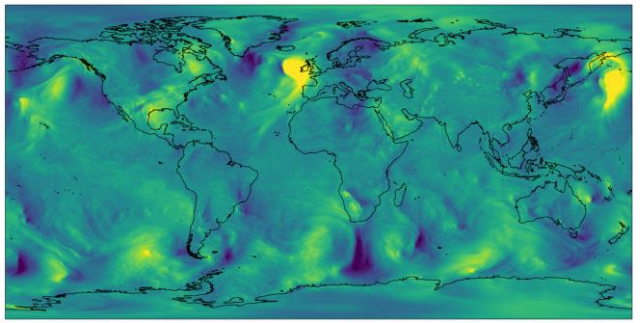


Reverse process (learned): remove noise

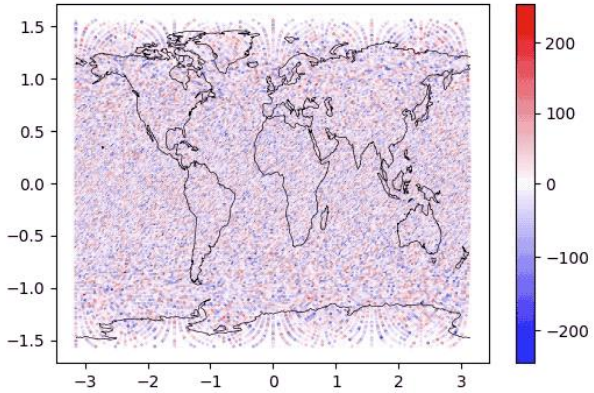
To generate a forecast (single member): start from pure noise with initial conditions, run the model, e.g. 40 times, to slowly create a forecast tendency step by step ; repeat for the next forecast step etc. -> 12 h step, 360 h lead time, call the model 1,200 times



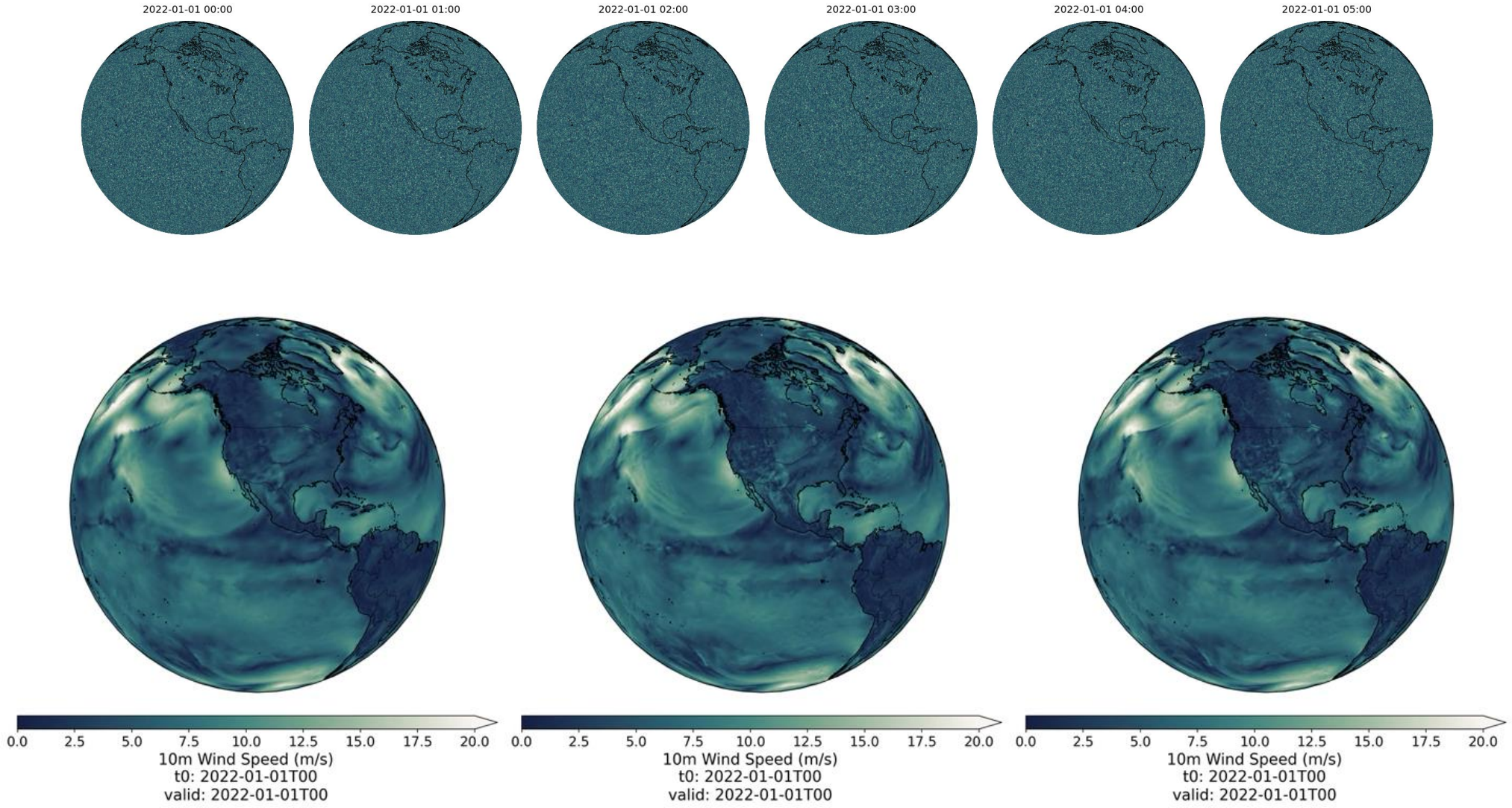
AIFS Diffusion



AIFS MSE



Example: Spatio-Temporal Downscaler via flow matching



ERA5, 6 h, 110 km

ERA5, 1 h, 30 km (target)

ERA5, 1 h, 30 km (prediction)

Proper scoring rules

Idea: let the network produce samples in **one pass**, and train it with a scoring rule that compares an ensemble with the target.

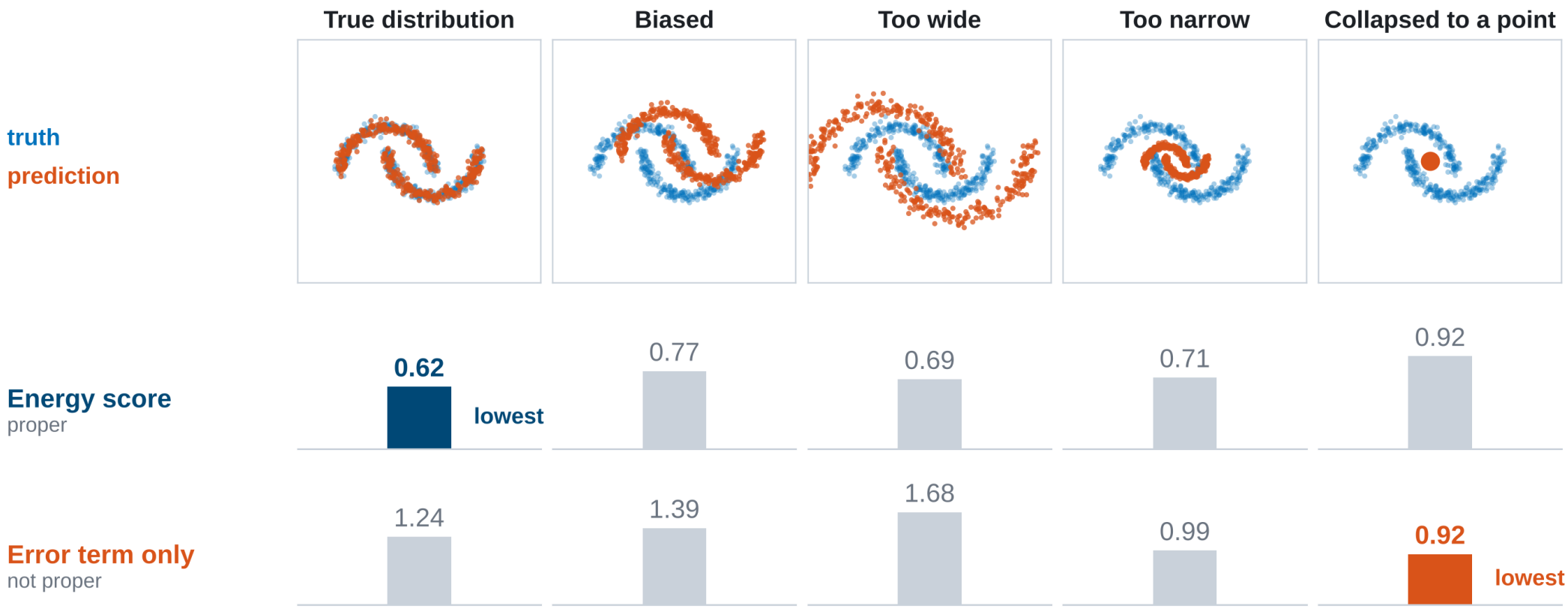
- A **scoring rule** S(forecast, y) says how good a forecast distribution was once the observation y is known. Here, lower is better.
- **Proper**: on average over many cases, the score has its minimum when the forecast distribution is equal to the target distribution.

- **First term**: attraction to the target. **Second term**: repulsion between members.
- Finite members: use fair form.
- **CRPS**: one dimensional case -> one grid point and variable.

Energy score of a forecast distribution:

```
ES = E|X - y| - 1/2 E|X - X'|
X, X': two independent draws of the forecast
|.| : distance over all grid points and variables together
```

Proper scoring rules: the energy score

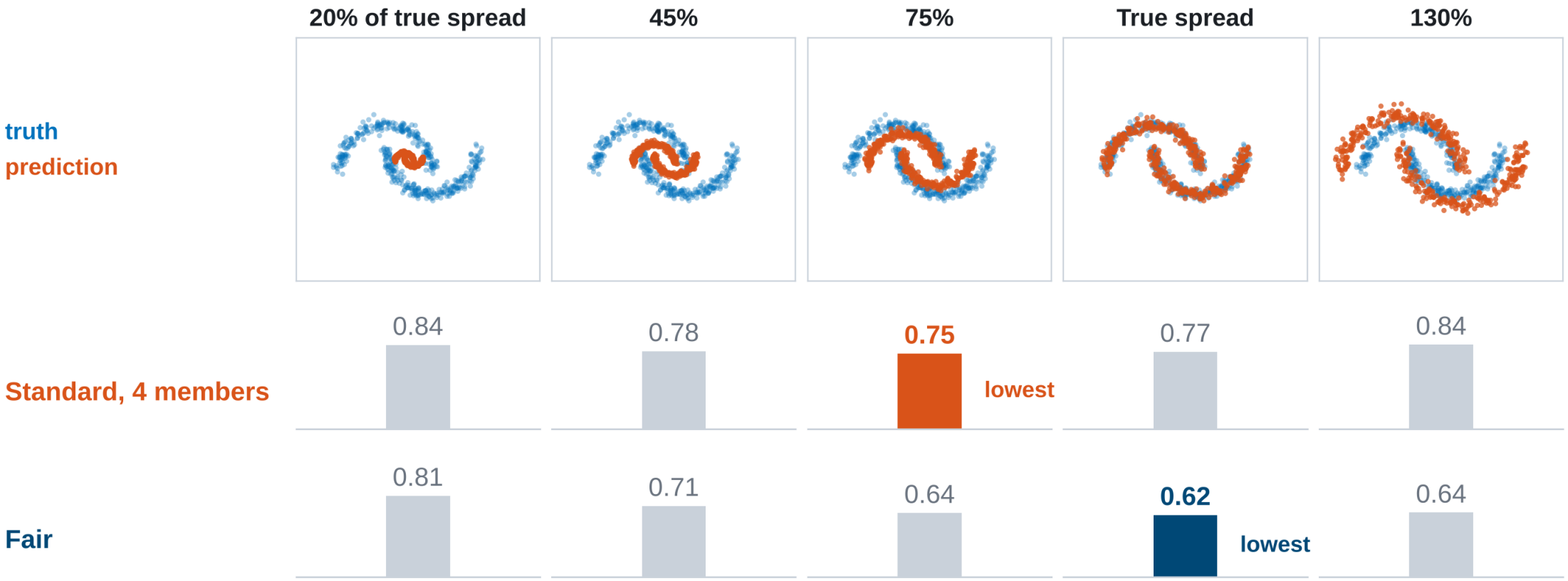



```
energy score = mean |member - obs|
  - 1/2 * mean |member - other member|
```

Error term
Spread term

The sensitivity of the score to different aspects of the distribution matters!

Fair scores




```
standard: - sum|x_i - x_j| / (2 M^2)
fair:     - sum|x_i - x_j| / (2 M (M-1))
sum over all i and j
```

CRPS: spread, ensemble size and fair scores

Fair CRPS: corrects for the ensemble size.

```
CRPS = mean|x_i - y| - sum|x_i - x_j| / (2M^2)
fair = mean|x_i - y| - sum|x_i - x_j| / (2M(M-1))

almost fair = 0.95 * fair + 0.05 * CRPS
sum over all i and j, each pair counted twice
```

Why almost fair: the fair CRPS has a degeneracy. With a 2-member ensemble, one member on each side of the observation, it is exactly zero, however far apart the members are. A little of the standard CRPS removes this.

CRPS-based training

- The members are scored together against the analysis with the **almost fair CRPS** per grid point and variable.
- One network pass per member and forecast step.
- Can be trained over several forecast steps (rollout training)

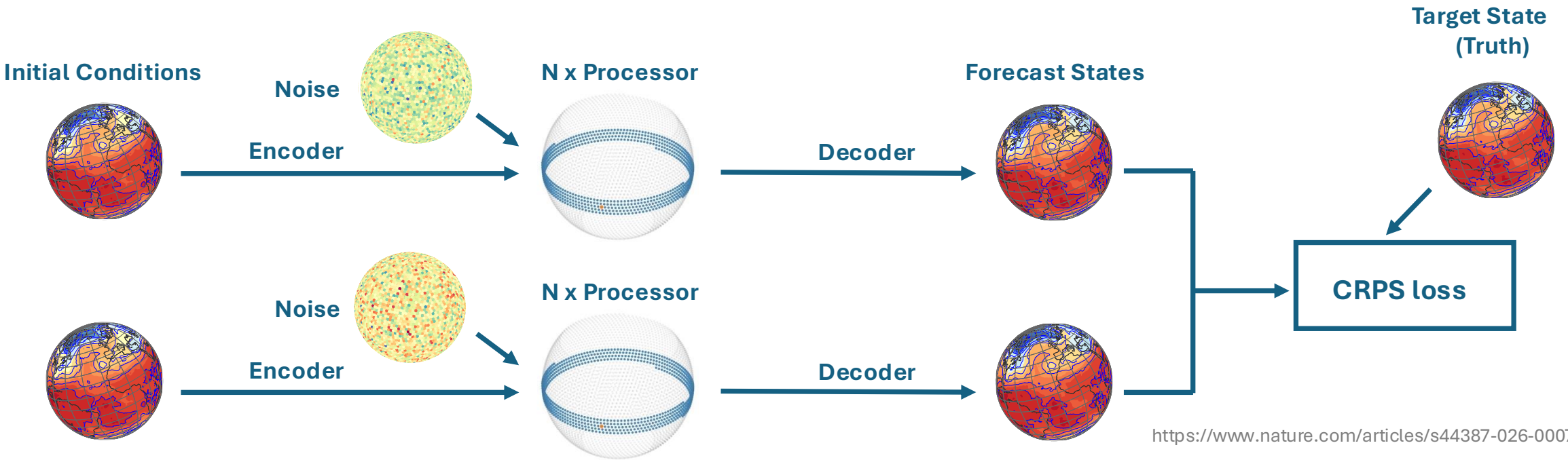
$$\mathrm{afCRPS}_\alpha := \alpha\,\mathrm{fCRPS} + (1-\alpha)\mathrm{CRPS}$$

$$= \frac{1}{M}\sum_{j=1}^{M}|x_j - y| - \frac{M-1+\alpha}{2M^2(M-1)}\sum_{j=1}^{M}\sum_{k=1}^{M}|x_j - x_k|$$

$$= \frac{1}{M}\sum_{j=1}^{M}|x_j - y| - \frac{1-\epsilon}{2M(M-1)}\sum_{j=1}^{M}\sum_{k=1}^{M}|x_j - x_k| \qquad \epsilon = (1-\alpha)/M$$

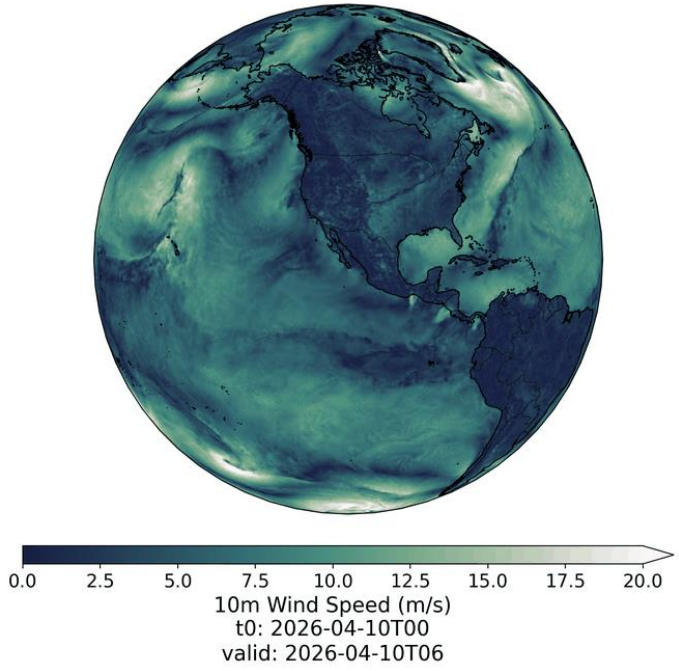
Members should be close to the truth

Differences between ensemble members should represent the forecast uncertainty

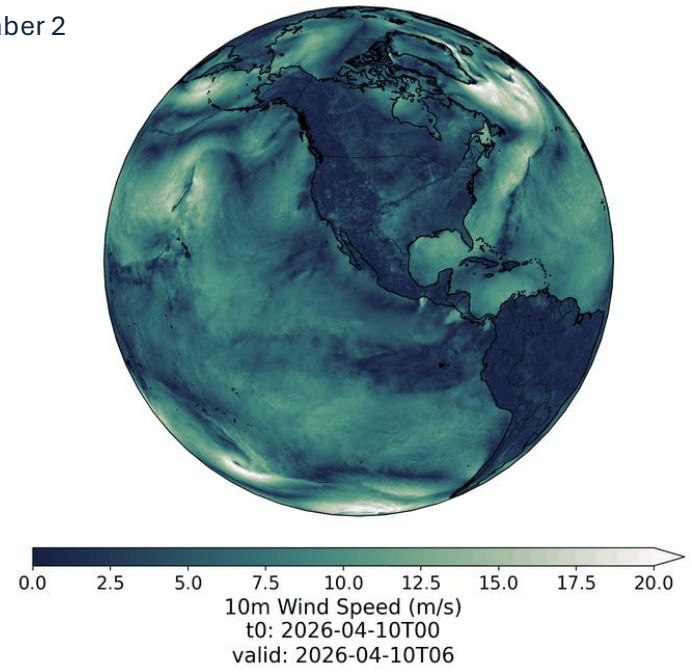


https://www.nature.com/articles/s44387-026-00073-7

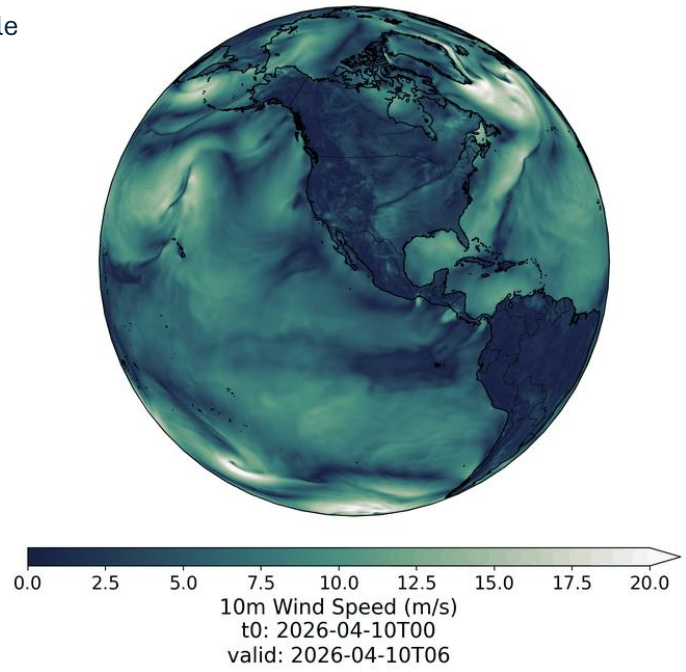
Member 1



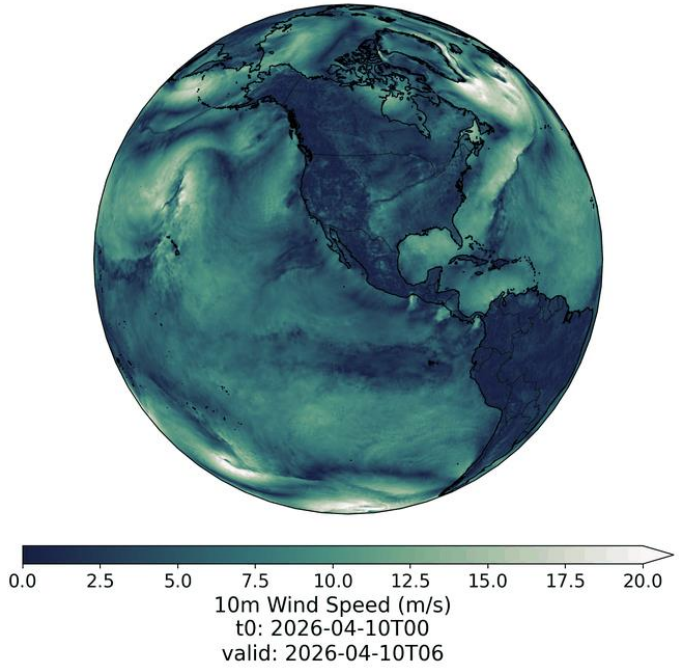
Member 2



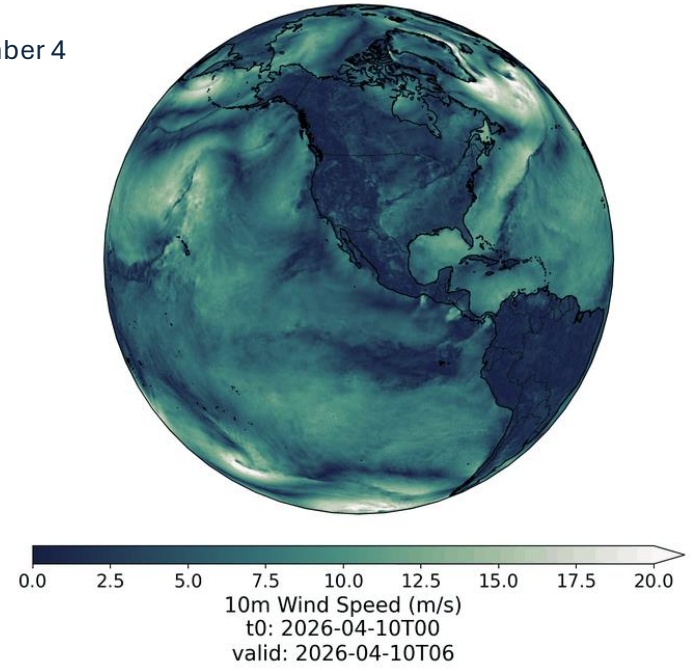
Single



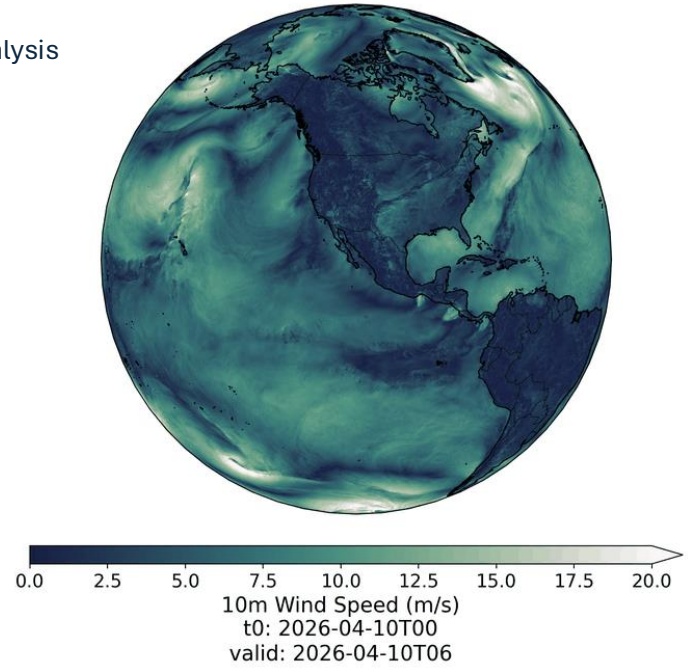
Member 3



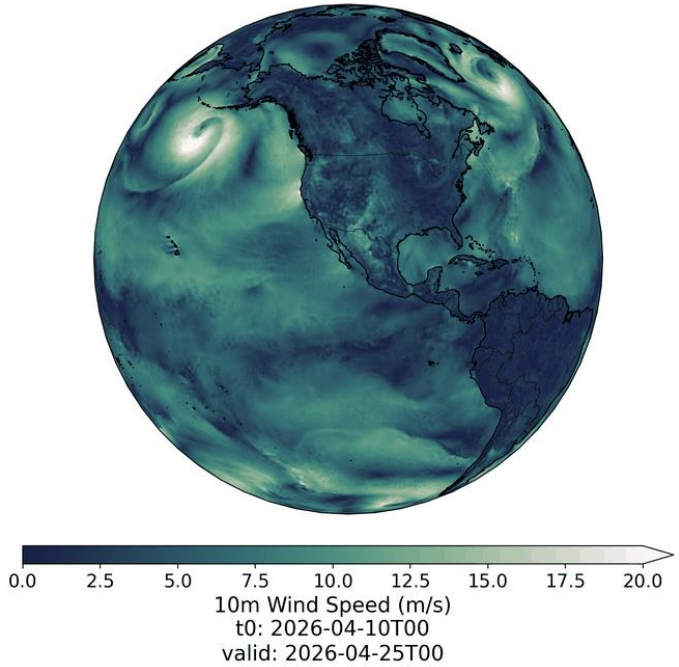
Member 4



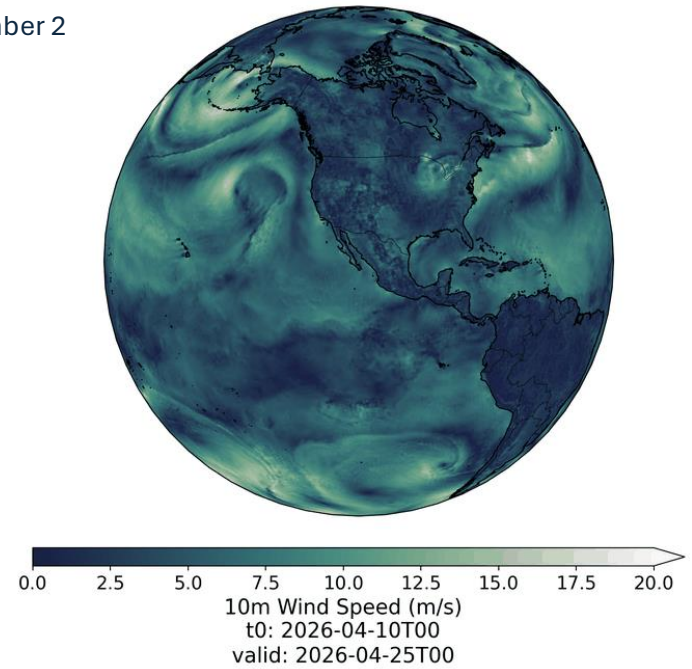
Analysis



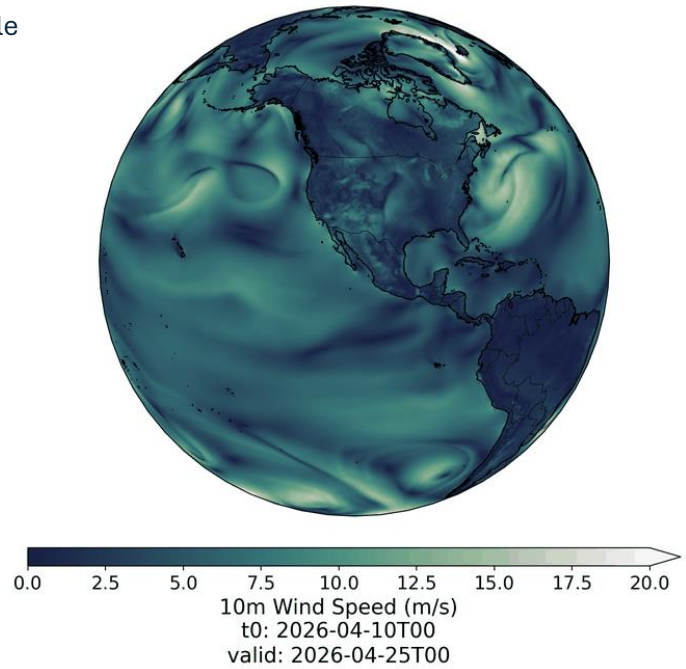
Member 1



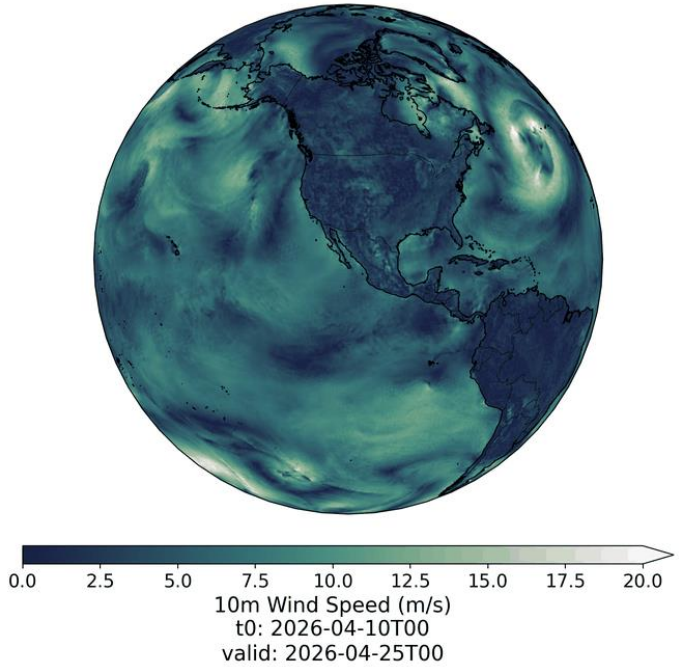
Member 2



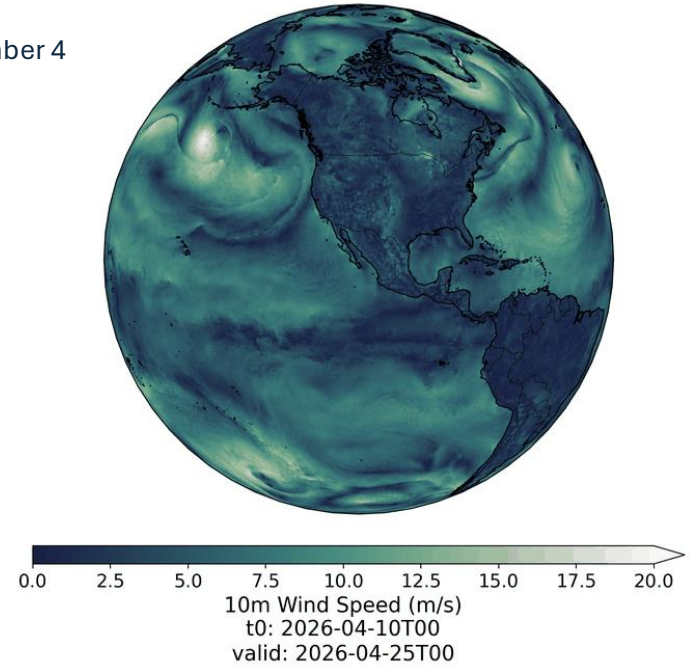
Single



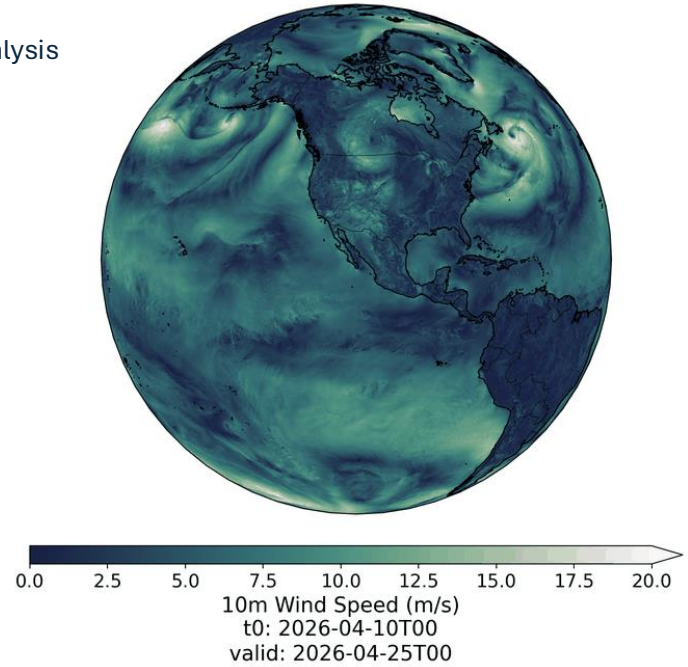
Member 3



Member 4



Analysis



Scores for spatial structure

The **CRPS** is computed grid point by grid point. It does not know that neighbouring points belong together, and different scales contribute very differently to it.

The **energy score** over the whole field sees the joint structure in principle. In high dimensions it struggles: members with the right point values but the wrong spatial pattern can score almost the same.

Alternatives:

- **CRPS / energy score in a feature space:** score transformed fields point by point, for example spectral coefficients, gradients or neighbourhood averages.
- **Multi-scale loss:** split each field into scales, score every scale separately. This removes small-scale noise from the members.
- **Proper scores on graphs:** score each grid point together with its neighbours (energy score, variogram score, differences along edges).

Make the loss scale-aware:

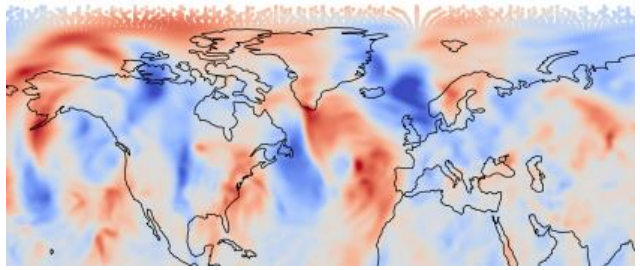
CRPS is computed point-wise on the full output field. However, atmospheric processes are inherently multi-scale, and different scales contribute to a different degree to the loss function -> introduce a scale-aware loss

Scale-unaware: $\mathcal{L} = c \int_{\mathcal{M}} \mathcal{S}([x_j \mid j = 1, \dots M], y)\, \mathrm{d}\mu$ scoring rule $\mathcal{S}$ for scalars

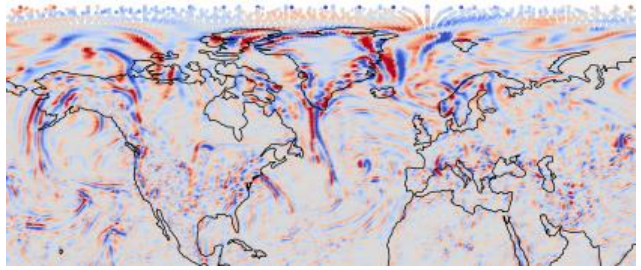
Scale-aware: $\mathcal{L}_{n\text{-scale}} = \sum_{i=1}^{n} \zeta_i\, c \int_{\mathcal{M}} \mathcal{S}([x_{j,\text{scale}\,i} \mid j = 1, \dots M], y_{\text{scale}\,i})\, \mathrm{d}\mu$

With ordered smoothing operators D_i that partition of a function ϕ on the manifold M into n scales:

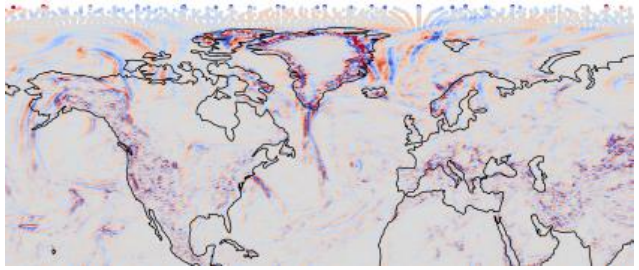
$$\begin{aligned}
\phi_{\text{scale}\,1} &= D_1(\phi) \\
\phi_{\text{scale}\,2} &= D_2(\phi) - D_1(\phi) \\
&\vdots \\
\phi_{\text{scale}\,n} &= \phi - D_{n-1}(\phi)
\end{aligned}$$



Scale 1



Scale 2



Scale 3

Scores for spatial structure

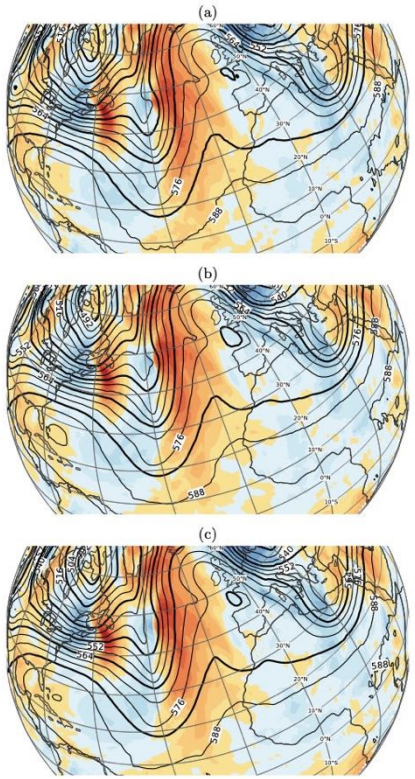
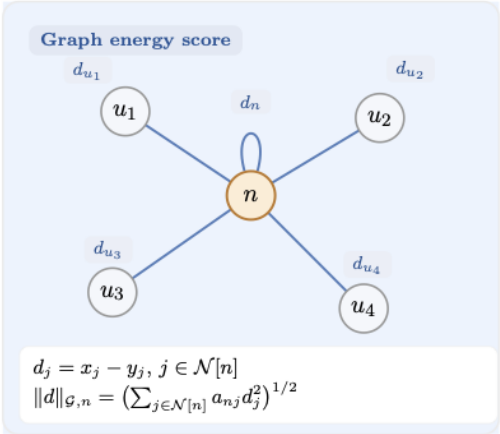
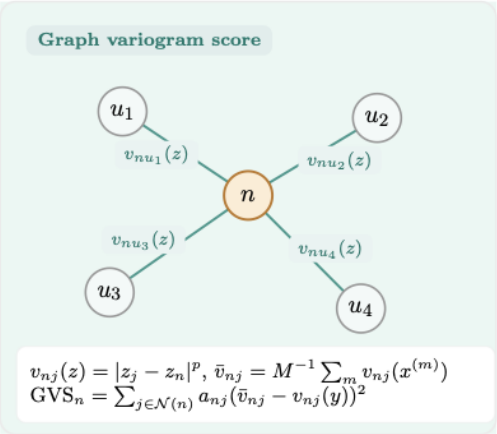
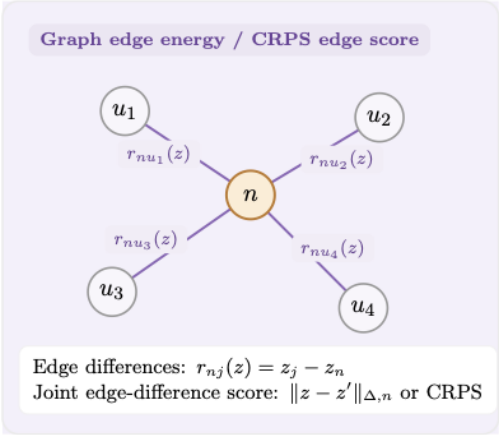


Figure 6: Geopotential at 500 hPa (dam, contours) and v-component of wind at 850 hPa (shaded) of (a) the scale-unaware loss experiment, (b) the multi-scale loss experiment and (c) the ERA5 analysis. Shown are 24 h forecasts of member 1, initialised on 2019-01-01 00 UTC (a, b) and verifying analysis on 2019-01-02 00 UTC (c).

(a) plain point-wise loss, (b) multi-scale loss, (c) reanalysis. 24-hour forecast, one member.







Proper scores defined on a grid point and its neighbours.

Comparison of model families

Family	Trained by	Network calls per forecast step	Good at	Watch out for
VAE	reconstruction error + KL term on the latent	1 pass	fast; gives a compact latent space	smooth, blurry samples
GAN	a second network tries to tell samples from data	1 pass	realistic fields (skill?)	unstable training; can ignore parts of the distribution
Diffusion	denoising with MSE at many noise levels	many passes	sharp, realistic fields; stable training	slow sampling; choose noise schedule and weights
Flow matching	velocity along straight paths from noise to data, MSE	many passes	simple to set up; potentially straighter paths, fewer steps	same family as diffusion, same cost in kind
Proper scores	CRPS or energy score of a small ensemble	1 pass	cheap to generate ensembles; forecast skill	The score matters: e.g. point-wise scores do not see spatial structure

Examples, not a complete list.

A lot of literature now about few step generation and diffusion models: distillation, flow maps …

Questions



www.ecmwf.int