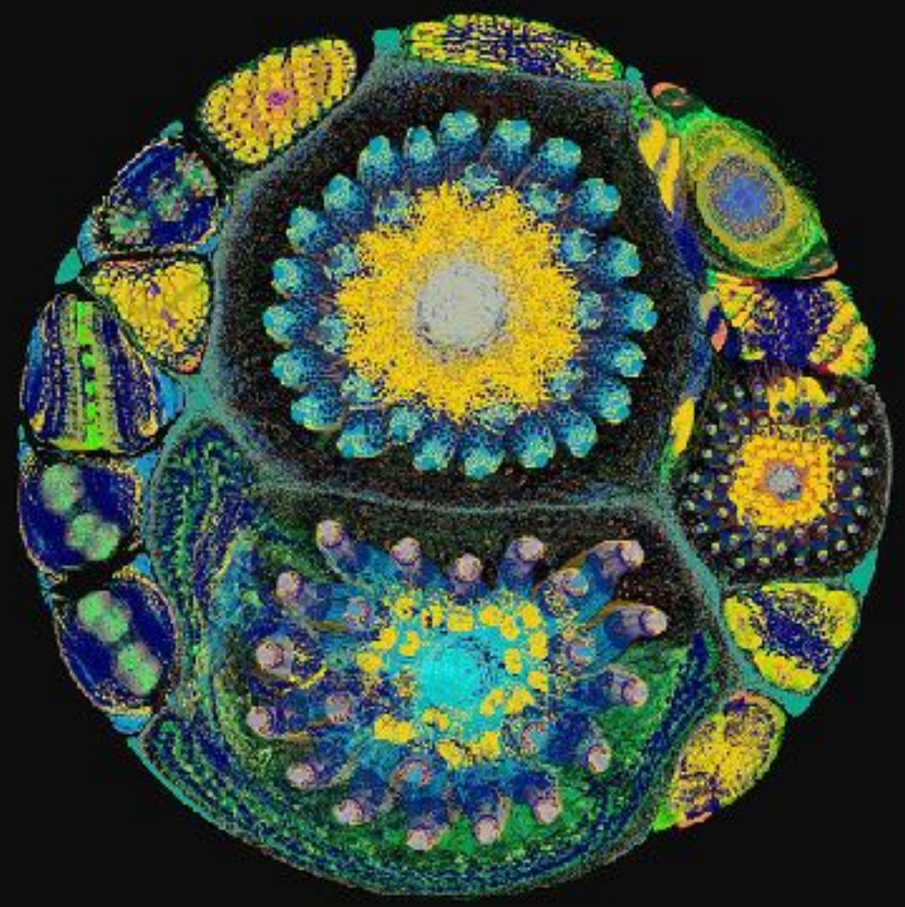
Introduction to Diffusion Models for Astrophysical Applications

Laurence Perreault Levasseur



Ciela Institute
Université de Montréal
Mila
Flatiron Institute
Center for Computational Astrophysics

Modern Cosmology+Astrophysics

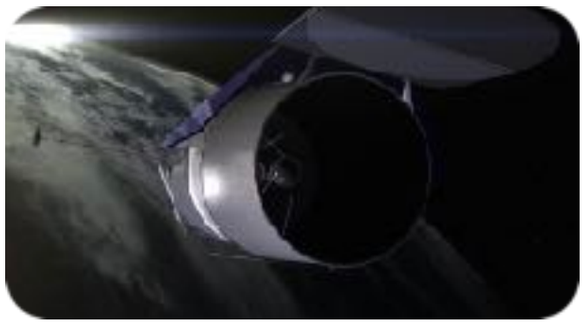
SKA 2.3 billion USD

TMT 2.4 billion USD

JWST 10 billion USD

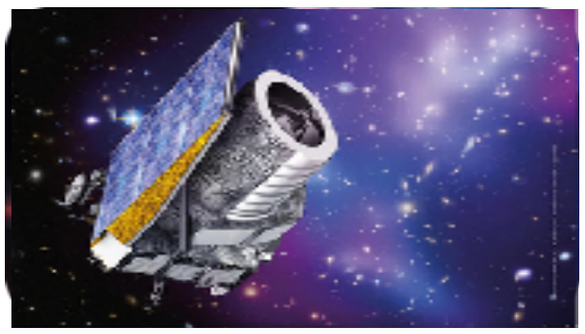
LSST 2 billion USD

ALMA 1.3 billion USD



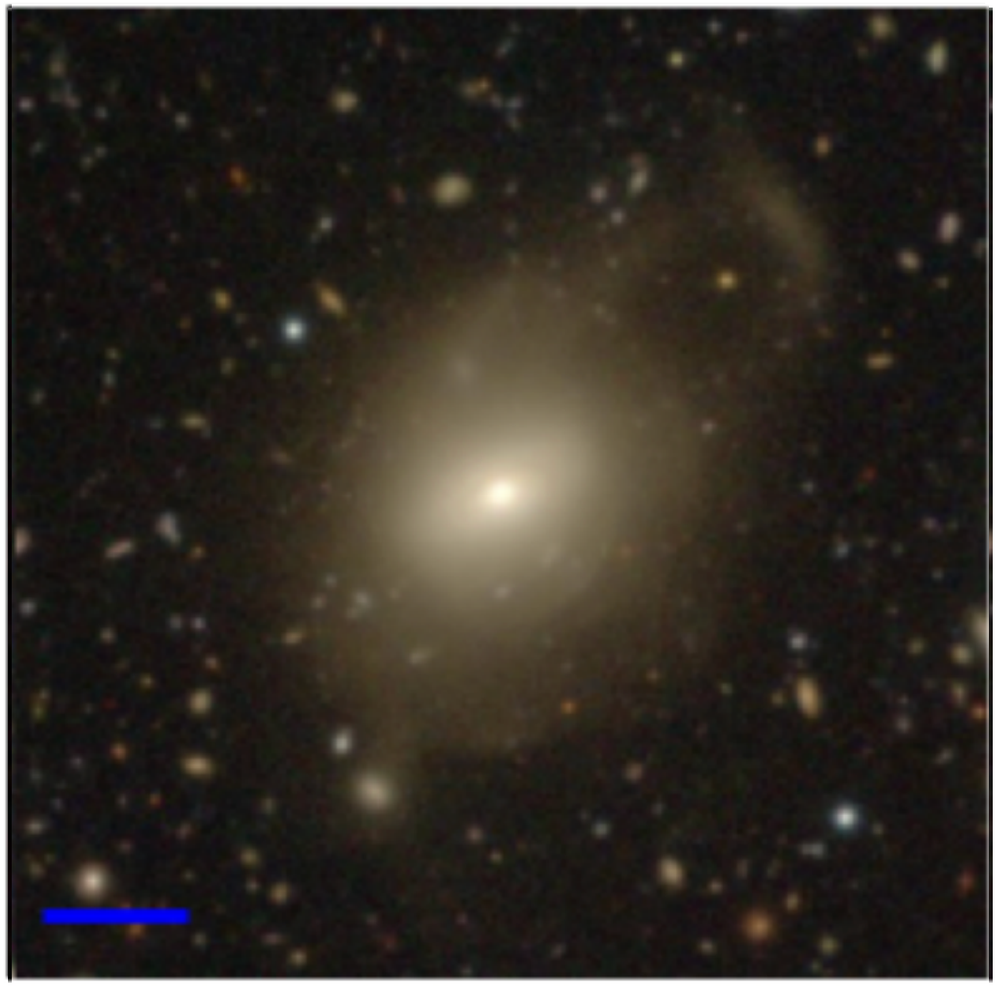
WFIRST 3.2 billion USD

ATHENA 1 billion USD



Euclid 0.7 billion USD

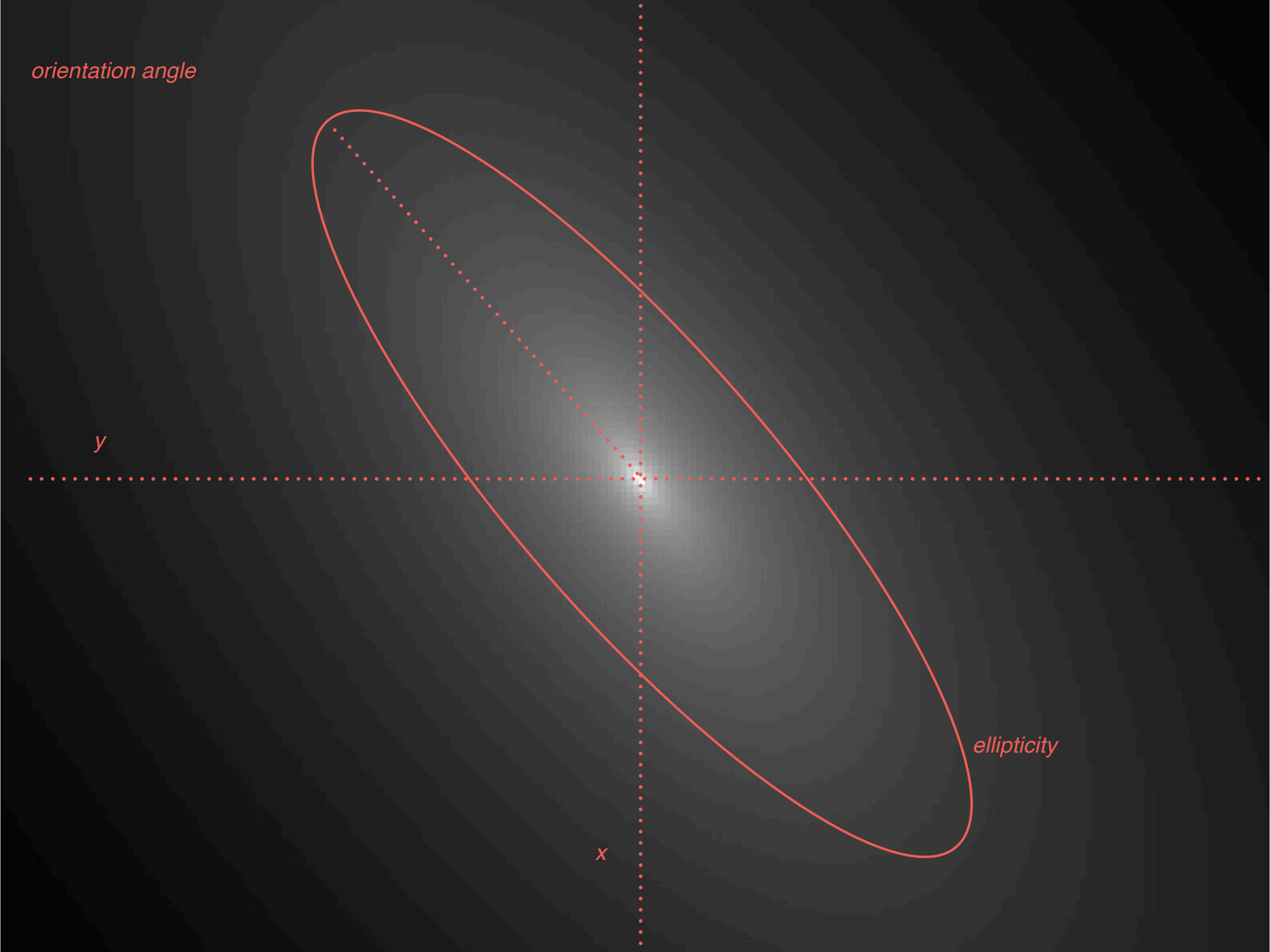
Data in astrophysics is getting more and more complex



HSC-SSP wide

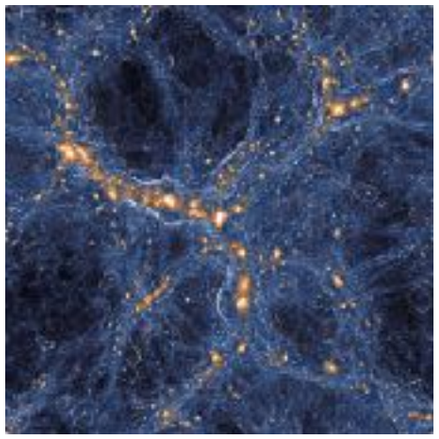
With this more complex data comes new opportunities,
But also unprecedented challenges

Jackson et al., 2022

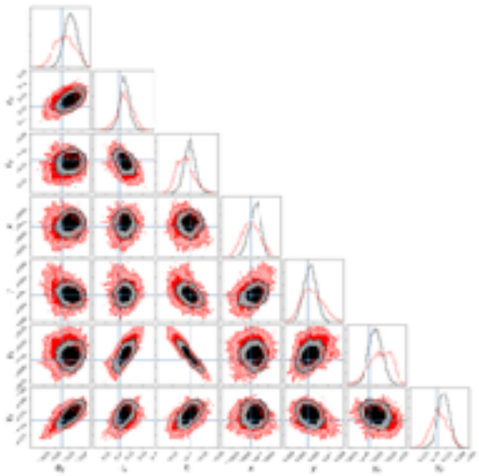

orientation angle
y
x
ellipticity

Lots of the things we want to do with this data require a *model* (implicit or explicit)

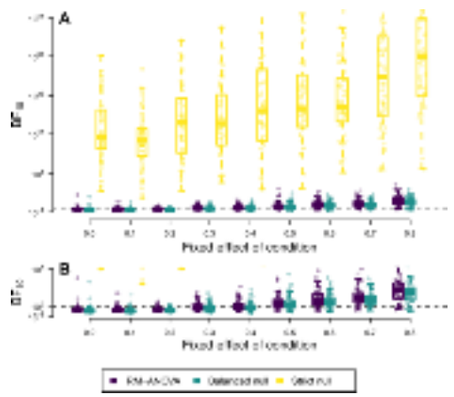
- Simulations



- Inference (a.k.a. solving inverse problems)

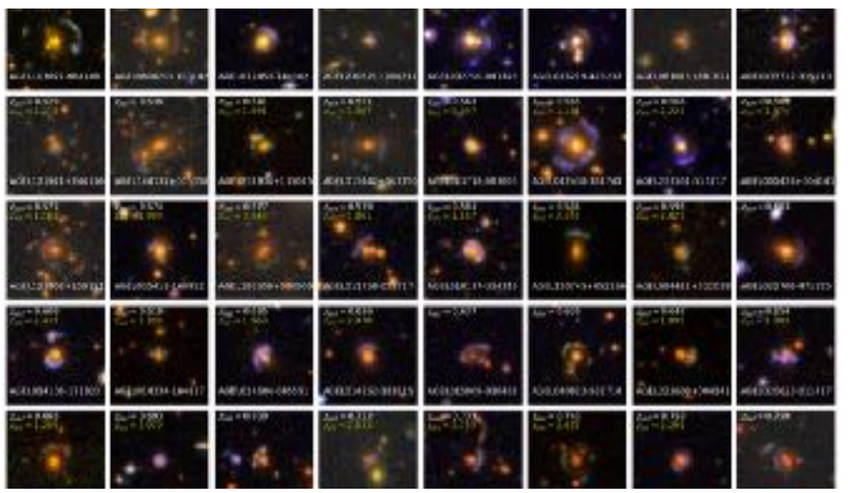


- Model comparison



- Etc…

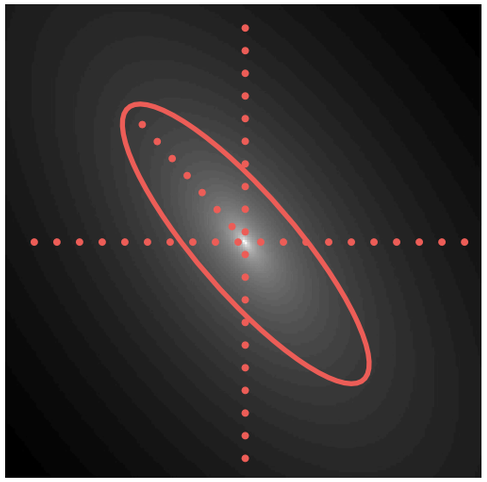
- Even (arguably) anomaly detection…



So we need models, but…

It’s becoming more and more challenging to come up with analytical models complex enough to keep up with the increasing complexity of our data

Simple/low resolution/low SNR data -> simple model might be sufficient



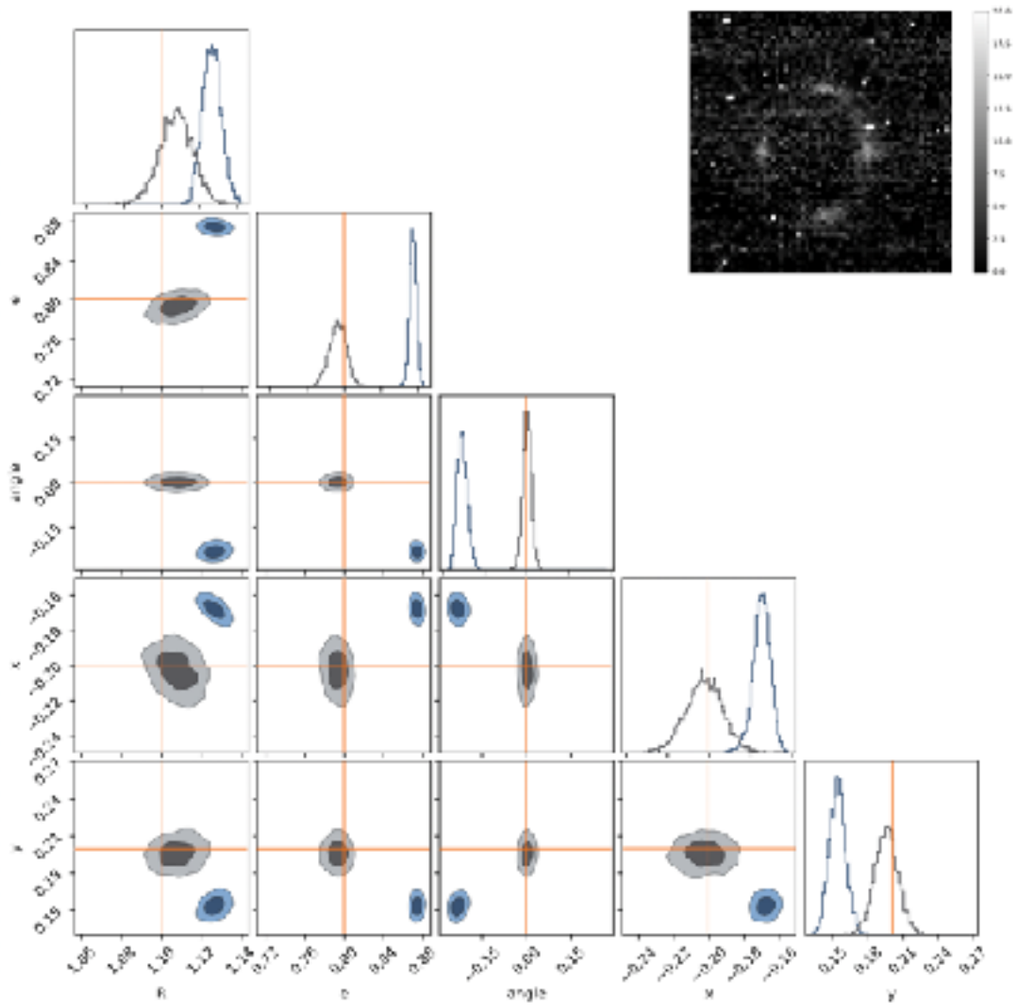
Better data -> with simplistic models we can bias the inference

?

So we need models, but…

It’s becoming more and more challenging to come up with analytical models complex enough to keep up with the increasing complexity of our data

And overly simplistic models can lead to biases…

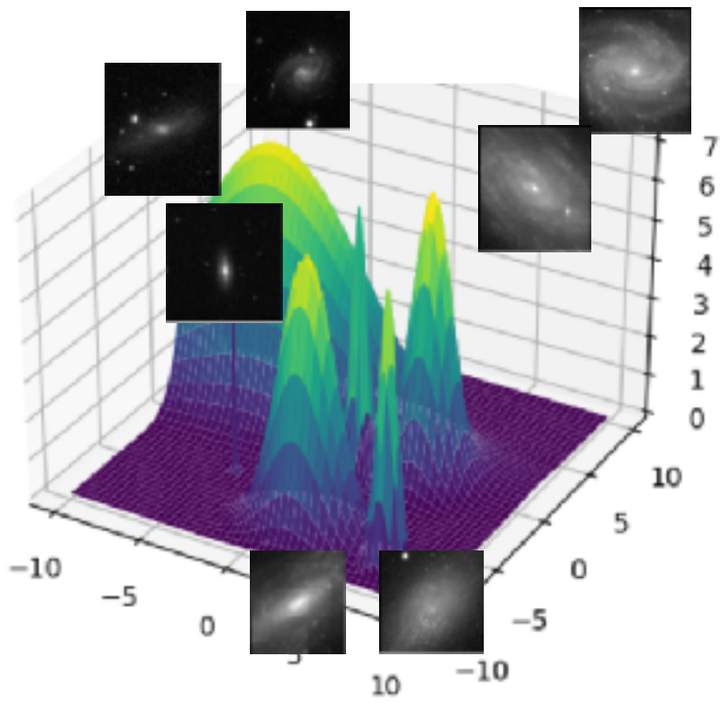


Can we learn the model from the data?
Can we have a data-driven model?

What would this mean?

E.g. image data lives in ***pixel space***, so it would mean to ***learn the distribution of the data*** in this (potentially very high dimensional) space.

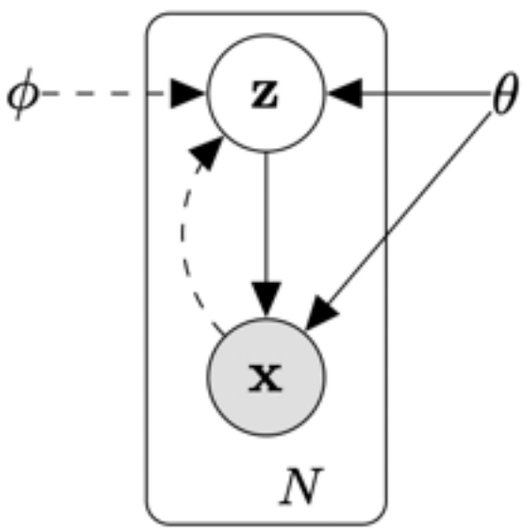
If we manage to learn this distribution, then we can sample from it to generate new examples! The way we encoded this distribution becomes our simulator!



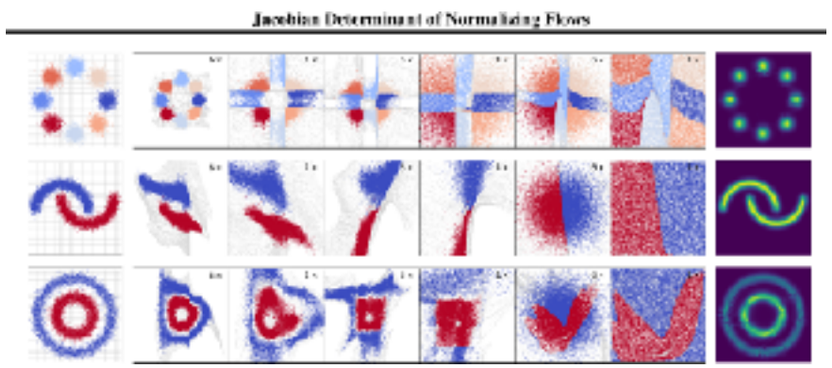
Can we learn the model from the data?
Can we have a data-driven model?

This is what generative models do.

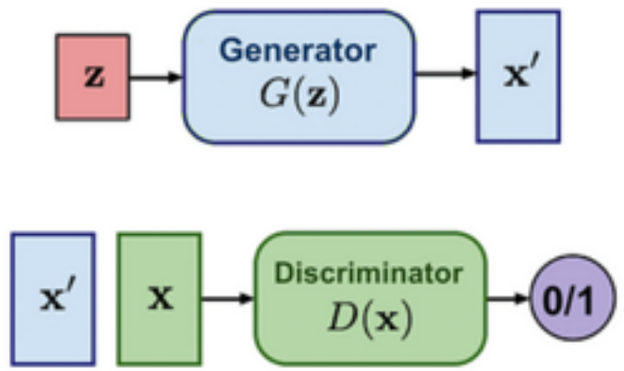
VAEs



Normalizing Flows



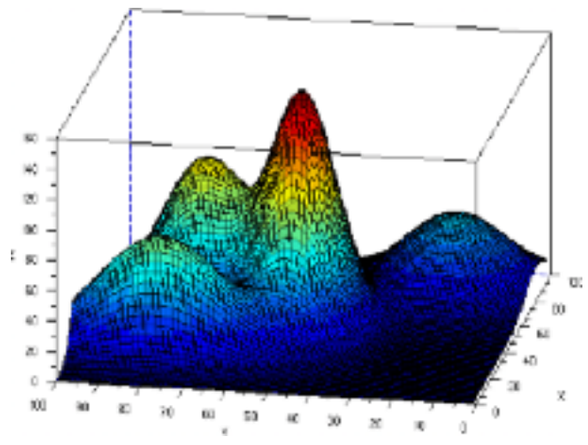
GANs



So…problem solved?
Not really…

Implicit models: like GANS: prone to mode collapse, hard to train since have to be trained adversially.

Explicit likelihood models: like flows. Lots of constraints on architecture to ensure we can calculate the normalization constant.



In the last few years there's been new generations of algorithms that have been developed to do this in high dimension. ***Diffusion models*** are such a class of algorithms, and this is what we are going to talk about today.

Game plan

1. Back to the basics physics of diffusion and Brownian motion.
2. How to turn this into a sampling algorithm
3. Score matching
4. How to learn to sample arbitrary distributions from samples
5. Example of application: data driven prior for inverse problems.

Brownian Motion in Physics…

A particle in a gas…

$$m\frac{d^2\mathbf{X}}{dt^2} + \gamma\frac{d\mathbf{X}}{dt} = \mathbf{F}$$ ⟶ This is not a Hamiltonian system (energy is not conserved!)

because of the friction term w/ coefficient γ

We know, if the particle has radius $\mathfrak{a}$, from Stokes:

$$\gamma = 6\pi\eta a$$

↑ Viscosity of gas

If we know $\mathbf{F}$, we can solve for the path of the particle exactly… doesn't this contradict that we expect the solution to be stochastic? Not if the force itself is unknown!

$$\mathbf{F} = -\nabla V + \mathbf{f}(t)$$

Deterministic force from a potential
e.g. - gravitational potential
- particle attached to spring…

Force due to collisions of all other particles in the gas => **noise**!

So we end up with…

$$m\frac{d^2\mathbf{X}}{dt^2}+\gamma\frac{d\mathbf{X}}{dt}=-\nabla V+\mathbf{f}(t)$$

Langevin equation

This is a ***stochastic differential equation*** -> we need to solve it without knowing $\mathbf{f}(t)$ exactly.

What does it mean?

If I knew the specific series of collisions a specific particle went through in a given time interval, then we would know exactly $\mathbf{f}(t)$ for this particle and we would be able to solve the equation exactly.

BUT!

- We would be clueless about what happens later, or for a different particle.
- We really don't want to have to solve for the exact dynamics of all the individual gas molecules to figure out all the collisions that lead to our motion (and, in fact, we can't).

What do we do then??

Let's say we can figure out some crude properties of $\mathbf{f}(t)$, like its average,
Then we hope it's enough to solve for crude properties of $\mathbf{X}(t)$, like its average!

That's our goal when solving a stochastic differential equation

Lets simplify…

$$m\frac{d^2\mathbf{X}}{dt^2} + \gamma\frac{d\mathbf{X}}{dt} = -\nabla V + \mathbf{f}(t)$$

First look at a system in the absence of external forces

$$=> \qquad V = 0$$

What about in a very viscous medium (γ large)?

Then the dynamics will be completely dominated by the friction term,
We ignore the inertial term => equivalent to setting $m = 0$.

We're left with:

$$\frac{d\mathbf{X}}{dt} = \frac{1}{\gamma}\mathbf{f}(t)$$

Obviously, if we knew the specific realization of $\mathbf{f}(t)$, we could trivially integrate this to give:

$$\mathbf{x}(t) = \mathbf{x}(0) + \frac{1}{\gamma}\int_0^t \mathbf{f}(t')dt'$$

But what if we know only the statistical properties of $\mathbf{f}(t)$?

What do we know about $\mathbf{f}(t)$?
Since we're looking at a particle being kicked around by gas molecules, it's safe to assume the gas is isotropic and so the average force should vanish:

$$\langle \mathbf{f}(t) \rangle = 0$$

$$=> \qquad \langle \mathbf{x}(t) \rangle = \mathbf{x}(0)$$

If on average the noise vanishes, on average the particles will stay at the same place!

But what if we know only the statistical properties of $\mathbf{f}(t)$?

What about the 2pt correlation function?

$$\langle (\mathbf{x}(t) - \mathbf{x}(0))^2 \rangle = \frac{1}{\gamma^2} \int_0^t \int_0^t \langle \mathbf{f}(t_1) \cdot \mathbf{f}(t_2) \rangle dt_1 dt_2$$

The components of the noise correlator are (i, j are the directions $1, 2, 3$ of the kick in 3D):

$$\langle \mathbf{f}(t_1) \cdot \mathbf{f}(t_2) \rangle = \delta_{ij} \langle f_i(t_1) f_j(t_2) \rangle$$

We assume that the collisions are independent and uncorrelated, so if we look at times $(t_1 - t_2) \gg \tau_{coll}$, then $\langle f_i(t_1) f_j(t_2) \rangle = 0$.

At macroscopic scales, we only care about timescales $(t_1 - t_2) \gg \tau_{coll}$, so we can take the limit $\tau_{coll} \to 0$, and we are left with a delta-function contribution.

$$\langle f_i(t_1) f_j(t_2) \rangle = 2D\gamma^2 \delta_{ij} \delta(t_1 - t_2)$$

But what if we know only the statistical properties of $\mathbf{f}(t)$?

What about the 2pt correlation function?

$$\langle (\mathbf{x}(t) - \mathbf{x}(0))^2 \rangle = \frac{1}{\gamma^2} \int_0^t \int_0^t \langle \mathbf{f}(t_1) \cdot \mathbf{f}(t_2) \rangle dt_1 dt_2$$

$$= \frac{1}{\gamma^2} \int_0^t \int_0^t \delta_{ij} \langle f_i(t_1) f_j(t_2) \rangle dt_1 dt_2$$

$$= \frac{1}{\gamma^2} \int_0^t \int_0^t \delta_{ij} 2D\gamma^2 \delta^{ij} \delta(t_1 - t_2) dt_1 dt_2$$

$$= 6D \int_0^t dt_2$$

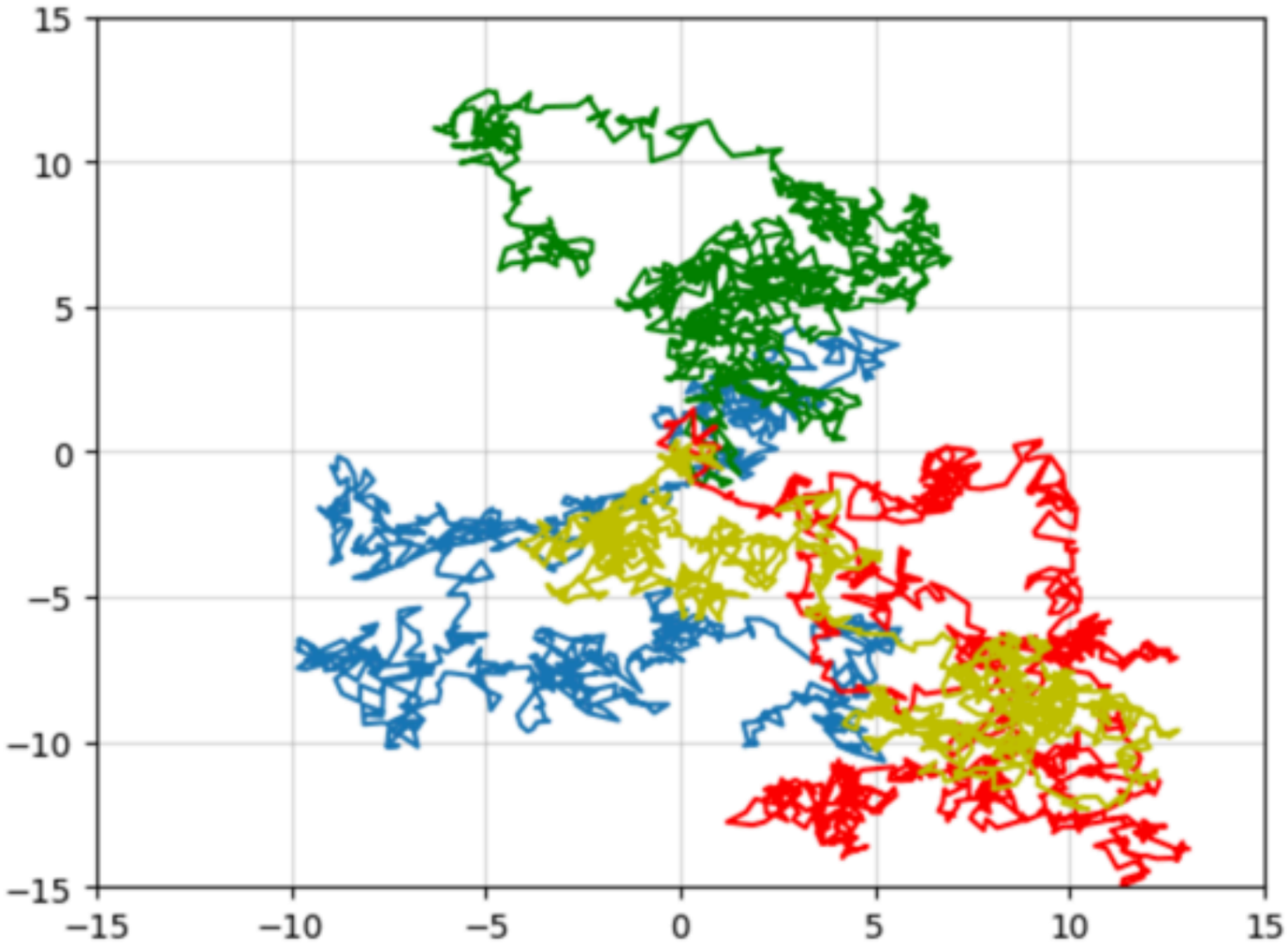
$$= 6Dt$$

The **rms** of the distance from starting point increases **as $\sqrt{t}$ with time**.

This is characteristic behaviour of diffusion.

The coefficient D is called the diffusion constant.

$$\langle \mathbf{x}(t) \rangle = \mathbf{x}(0) \qquad \langle (\mathbf{x}(t) - \mathbf{x}(0))^2 \rangle \propto t$$



To solve numerically, it's easier to write the SDE as:

$$d\mathbf{x}(t) = \sqrt{2D}\, dW(t)$$

Where $W(t)$ is a Wiener process, meaning:

- It is (almost surely) continuous;
- It has independent increments: for every $t > 0$, $W_{t+u} - W_t$ for every $u > 0$ is independent of past W
- It has Gaussian increments: $W_{t+u} - W_t \sim \mathcal{N}(0, u)$

This gives us an easy way to solve Langevin equation numerically (this is the stochastic generalization of Euler's method):

$$X_0 = x(0)$$
$$X_{n+1} = X_n + \sqrt{2D}\, \Delta W_n, \qquad n = 0, 1, \ldots, N-1$$

Where we have defined:

$$\Delta W_n = W_{n+1} - W_n \sim \mathcal{N}(0, \Delta_n) \qquad \Delta_n = t_{n+1} - t_n$$

[1]:
```
import numpy as np
from matplotlib import pyplot as plt
```

[372]:
```
def langevin_step(xn,yn, delta_t , sigma):

    xn = xn + sigma*np.random.normal(0, np.sqrt(delta_t))
    yn = yn + sigma*np.random.normal(0, np.sqrt(delta_t))
    return xn,yn
```

[469]:
```
def langevin_integrator(x0, y0, D, x, y, t ):
    x[0] = x0
    y[0] = y0
    delta_t=t[1]-t[0]
    sigma = np.sqrt(2*D)
    for i in range(len(t)-1):
        x[i+1],y[i+1] = langevin_step(x[i],y[i], delta_t , sigma)
    return x, y
```

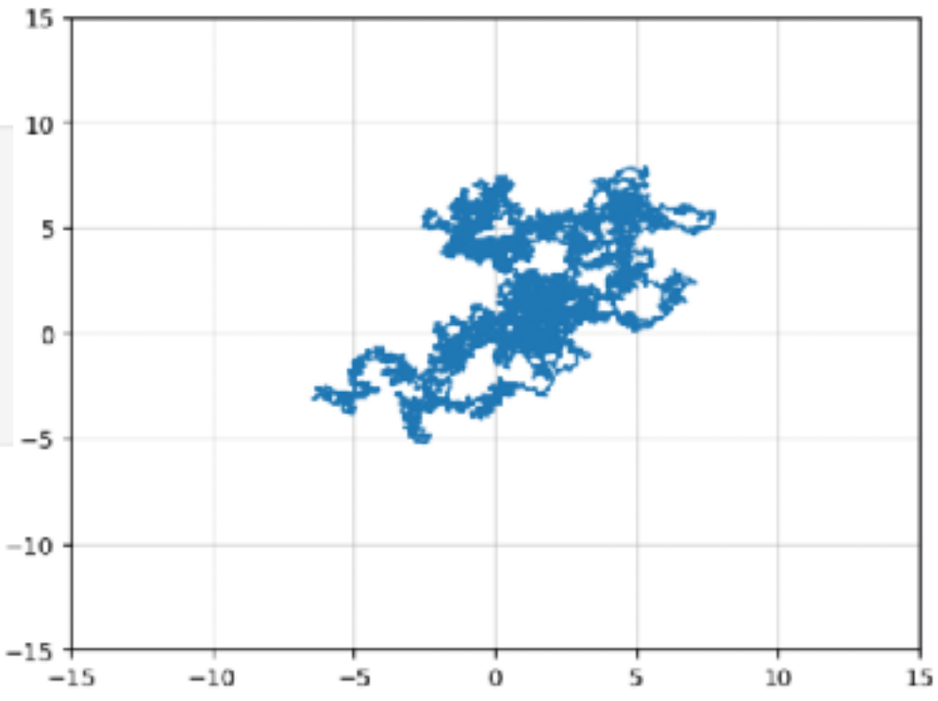
[481]:
```
X = np.zeros(10000)
Y = np.zeros(10000)
t=np.linspace(0,100, 10000)
```

[482]:
```
X, Y = langevin_integrator(0,0,0.5, X, Y, t)
```

[483]:
```
plt.plot(X[:], Y[:])
plt.xlim([-15, 15])
plt.ylim([-15, 15])

plt.grid(which='major', axis='both', alpha=0.5)

plt.show
```



The Fokker-Planck equation: an intro

When dynamics is noisy, we can't track down the future evolution of a particle, the best we can do is talk about probabilities:

$$\mathbf{x}(t) \sim P(\mathbf{x}, t; \mathbf{x}_0, 0)$$

So far we've talked about this in terms of correlation functions of a given path of the particle.

Now we want to know: what $P(\mathbf{x}, t; \mathbf{x}_0, 0)$ would give rise to the same correlations as the Langevin equation.

We are *NOT* asking about the probability distribution over *paths* (that's hard)!

We want to know the pdf of the particle being at $\mathbf{x}$ at t regardless of how it got here:

$$P(\mathbf{x}, t) = \langle \delta(\mathbf{x} - \mathbf{x}_f) \rangle$$

For simple Brownian motion:

For the Langevin equation we've been considering so far, with $\mathbf{x}(0) = 0$, we have:

$$P(\mathbf{x}, t; 0,0) = \frac{1}{(2\pi 2Dt)^{n/2}} e^{-\frac{1}{2}\cdot\frac{\mathbf{x}^2}{2Dt}}$$

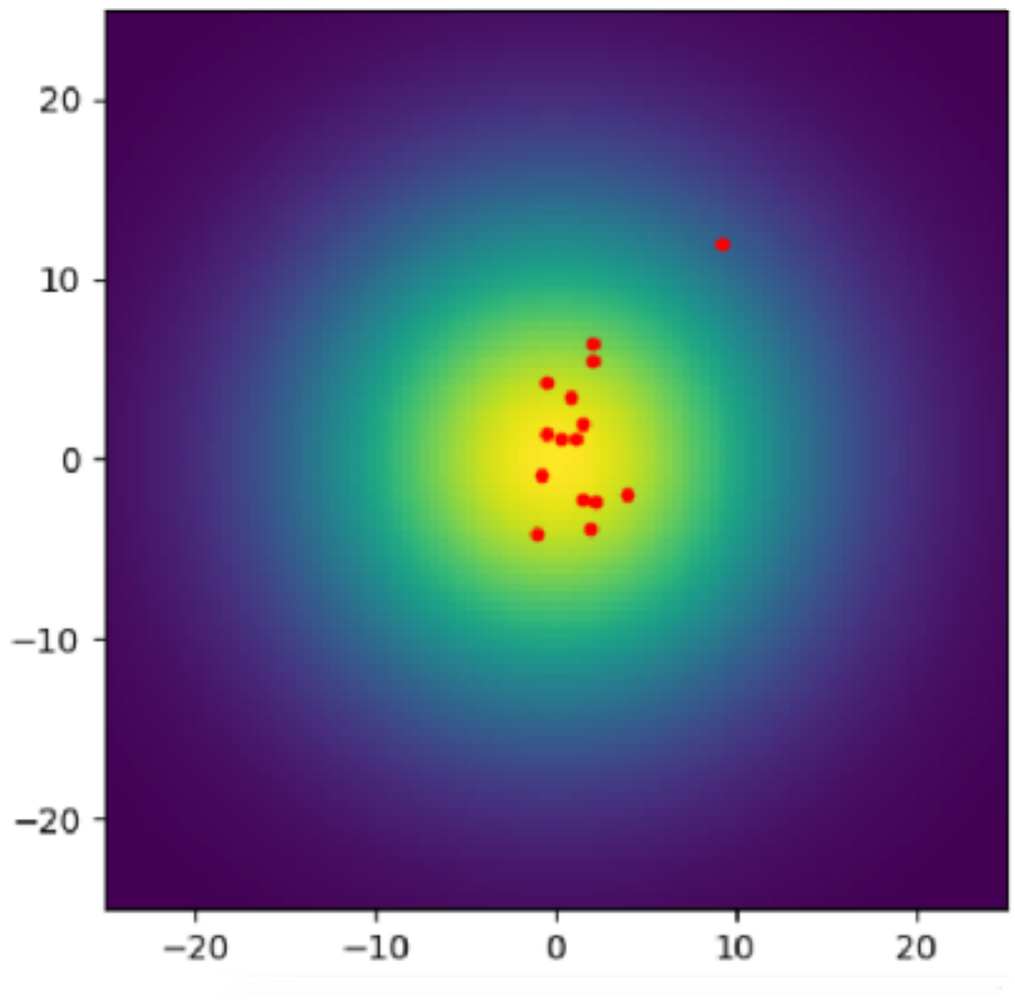
This pdf obeys the ***diffusion equation***:

$$\frac{\partial P(\mathbf{x}, t)}{\partial t} = D\,\nabla^2 P(\mathbf{x}, t)$$

This is the simplest example of a ***Fokker-Planck equation***

For the example we had above:

$$P(\mathbf{x}, t) = \frac{1}{(2\pi 2Dt)^{n/2}} e^{-\frac{1}{2} \cdot \frac{\mathbf{x}^2}{2Dt}}$$



Back to the Langevin equation:

Still considering the case of a very viscous fluid, we have $m = 0$ and

$$\gamma \frac{d\mathbf{x}}{dt} = -\nabla V + \mathbf{f}(t)$$

For some general potential $V(\mathbf{x}, t)$

Numerically, we can still solve this by re-writing this equation

$$d\mathbf{x}(t) = -\frac{1}{\gamma}\nabla V(\mathbf{x}(t), t)dt + \sqrt{2D}dW_t$$

And again integrating numerically

$$X_0 = x(0)$$

$$X_{n+1} = X_n - \frac{1}{\gamma}\nabla V(X_n, t_n)\Delta_n + \sqrt{2D}\,\Delta W_n\,, \qquad n = 0, 1, \ldots, N-1$$

$$\Delta W_n = W_{n+1} - W_n \sim \mathcal{N}(0, \Delta_n) \qquad \Delta_n = t_{n+1} - t_n$$

The Fokker-Planck equation

After some math magic

(involves looking at a small increment in time in the Langevin equation, looking at the 1 and 2pts correlators to leading order in δt, and Taylor expanding the pdf of the position of the particle)

$$\frac{\partial P(\mathbf{x},t)}{\partial t} = \frac{1}{\gamma}\nabla\cdot(P(\mathbf{x},t)\nabla V) + D\nabla^2 P(\mathbf{x},t)$$

Fokker-Planck equation, aka Kolmogorov's forward equation

We went from a ***stochastic ODE*** describing the path of a particle, to a ***deterministic PDE*** describing the probability of finding the particle at a given position

So… Why do we care about this?

We can turn this into a **sampling** algorithm!

How? If we can engineer a potential $V(\mathbf{x})$ such that the corresponding Langevin equation produces paths that follow a steady-state (stationary) distribution, the pdf P that solves the Fokker-Planck equation will be time-invariant, and if we initialize $\mathbf{X}_0$ according to that distribution, then we can simulate the SDE to generate samples of P.

So… Why do we care about this? Sampling!

The FP equation can be rewritten as a continuity equation:

$$\frac{\partial P}{\partial t} = \frac{1}{\gamma}\nabla \cdot (P \nabla V) + D \nabla^2 P$$

$$\frac{\partial P}{\partial t} = \nabla \cdot \mathbf{J}$$

With

$$\mathbf{J} = \frac{1}{\gamma} P \nabla V + D \nabla P$$

$\mathbf{J}$ is the probability ‘flow’…

So… Why do we care about this? Sampling!

$$d\mathbf{x}(t) = -\nabla V(\mathbf{x}(t), t)dt + \sqrt{2}dW_t$$

$$\frac{\partial P}{\partial t} = \nabla \cdot \mathbf{J} \qquad \text{with} \qquad \mathbf{J} = P\nabla V + \nabla P$$

The steady state distribution is given by $\frac{\partial P(\mathbf{x}, t)}{\partial t} = 0$ therefore $\nabla_{\mathbf{x}}\mathbf{J} = 0$ and $\mathbf{J}$ must be a constant in space.
Moreover, $\mathbf{J}$ must satisfy the boundary condition that it must vanish at $\mathbf{x} = \infty$, so $\mathbf{J}(\infty, t) = 0$ (there cannot be any probability flow at infinity since P is a probability density and must vanish at infinity)

But since $\mathbf{J}$ must be a constant, this means $\mathbf{J} = 0$ everywhere, and

$$P\nabla V + \nabla P = 0$$

So… Why do we care about this? Sampling!

$$P\nabla V + \nabla P = 0$$

In 1D it's obvious that this equation can easily be solved:

$$P_{stdy}(x)\frac{\partial V}{\partial x} = -\frac{P_{stdy}(x)}{\partial x}$$

$$=> \quad P_{stdy}(x) \propto \exp\left(-V(x)\right)$$

Therefore, this means we can sample pdfs of the form $\pi(x) = \frac{\exp\left(-E(x)\right)}{Z}$, by choosing $V(x) = E(x)$ in the Langevin equation.

Moreover, by writing $\pi(x) = \exp\left(\log \pi(x)\right)$, we can also sample ***any*** distribution $\pi(x)$ by choosing $V(x) = -\log(\pi(x))$

Important: we don't need the normalization constant Z since only $\nabla V(x) = \boxed{-\nabla \log(\pi(x))}$ appears in the SDE and $\nabla Z = 0$

Score

Unadjusted Langevin Monte Carlo Sampling

To produce samples from $\pi(x)$, we want to simulate trajectories that solve the SDE:

$$d\mathbf{x}(t) = \nabla \log(\pi(\mathbf{x}))dt + \sqrt{2}dW_t$$

Discretizing, we get:

$$X_{n+1} = X_n + \Delta_n \nabla \log\left(\pi(x)\right) + \sqrt{2}\,\Delta W_n, \qquad n = 0, 1, \ldots, N-1$$

$$\Delta W_n = W_{n+1} - W_n \sim \mathcal{N}(0, \Delta_n) \qquad \Delta_n = t_{n+1} - t_n$$

And time steps don't need to be equal.

It's also very common to write it as:

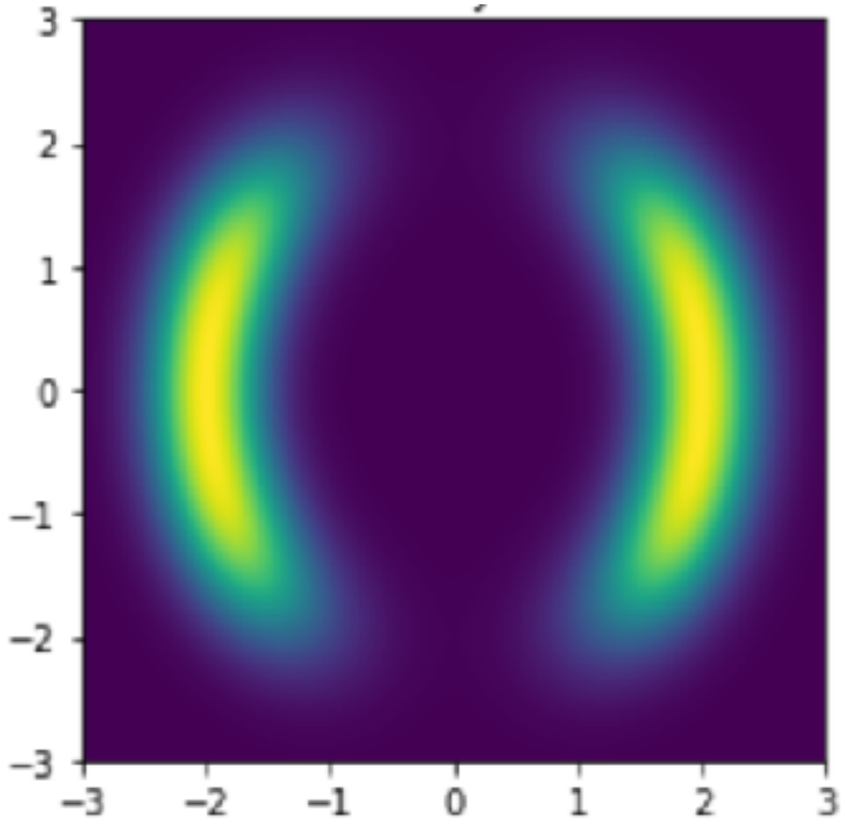
$$X_{n+1} = X_n + \Delta_n \nabla \log\left(\pi(x)\right) + \sqrt{2\Delta_n}\,\xi, \qquad n = 0, 1, \ldots, N-1$$

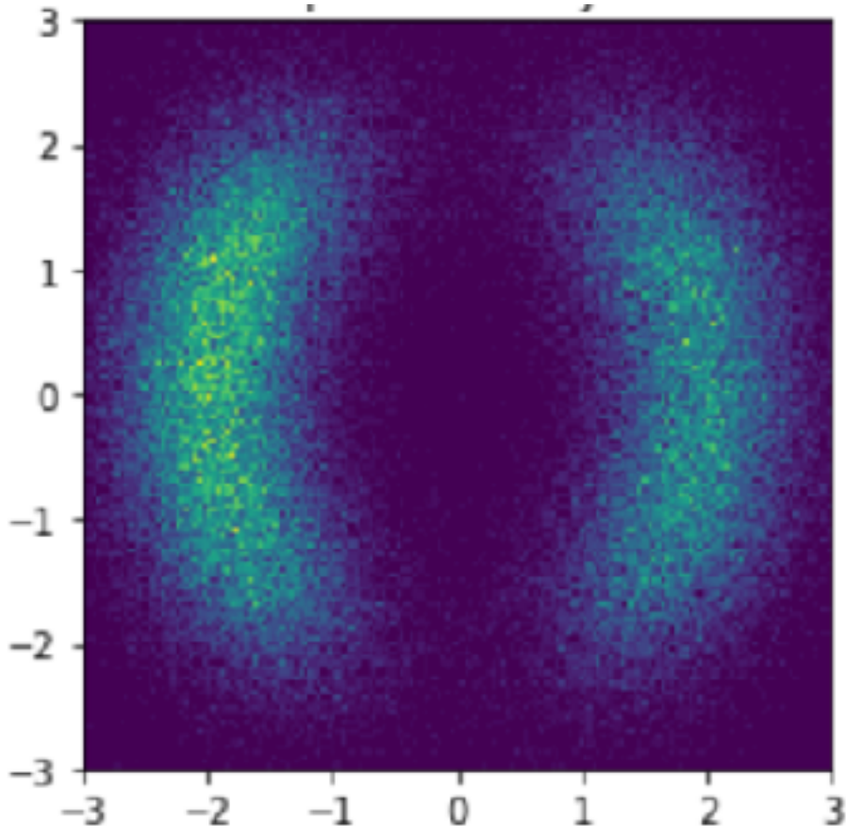
$$\xi \sim \mathcal{N}(0,1) \qquad \Delta_n = t_{n+1} - t_n$$

Example of ULA

$$V(\mathbf{x}) = \left(\frac{|\mathbf{x}|-2}{0.4}\right)^2 - \log\left(\exp\left[-\frac{1}{2}\left(\frac{x_1-2}{0.6}\right)^2\right] + \exp\left[-\frac{1}{2}\left(\frac{x_1+2}{0.6}\right)^2\right]\right)$$

$$p(\mathbf{x}) = e^{-V(x)}$$





Game plan

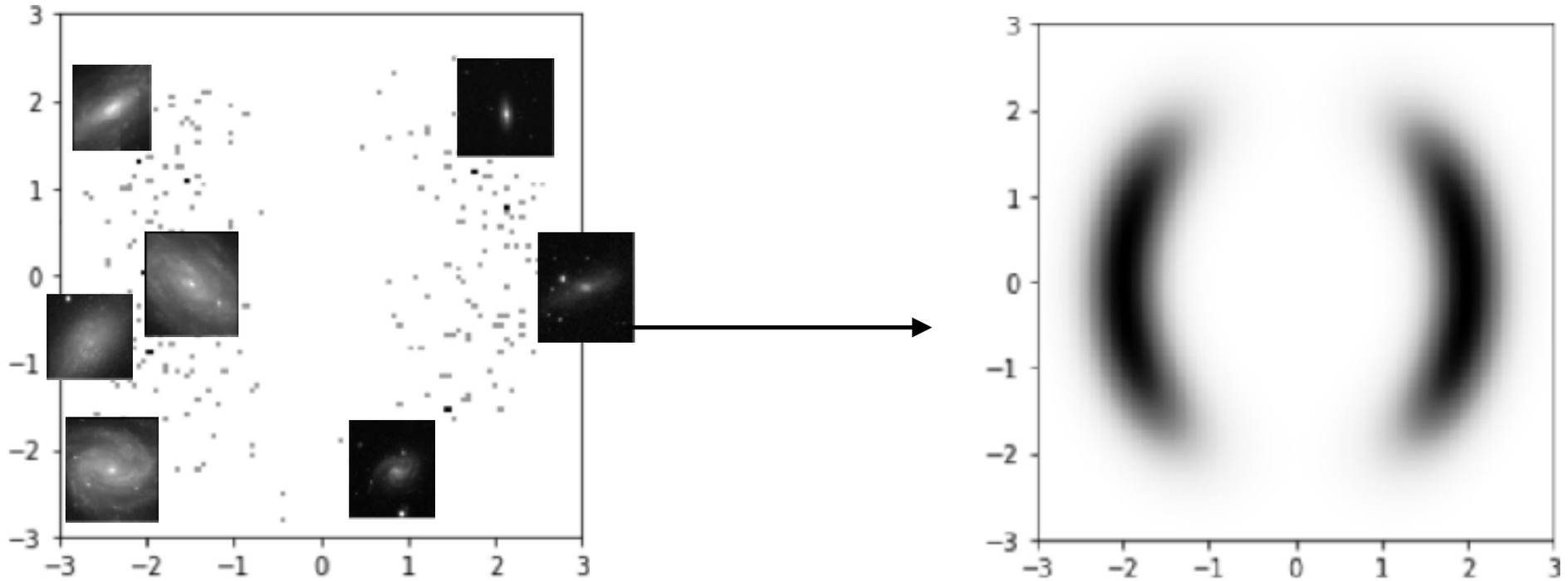
1. Back to the basics physics of diffusion and Brownian motion.
2. How to turn this into a sampling algorithm
3. **Score matching**
4. How to make this work to learn arbitrary distributions from samples (denoising diffusion)
5. Example of application: data driven prior for inverse problems.

Score Modeling

Recap: We have a way to sample any distribution $\pi(\mathbf{x})$ that does not require us to know Z, the normalization constant.
We only need to be able to evaluate its ***score***

$$\mathbf{s}(\mathbf{x}) = \nabla \log(\pi(\mathbf{x}))$$

How can we do this from samples (e.g. data)?



Training data

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n,\} \overset{iid}{\sim} \pi_{data}(\mathbf{x})$$

Density function

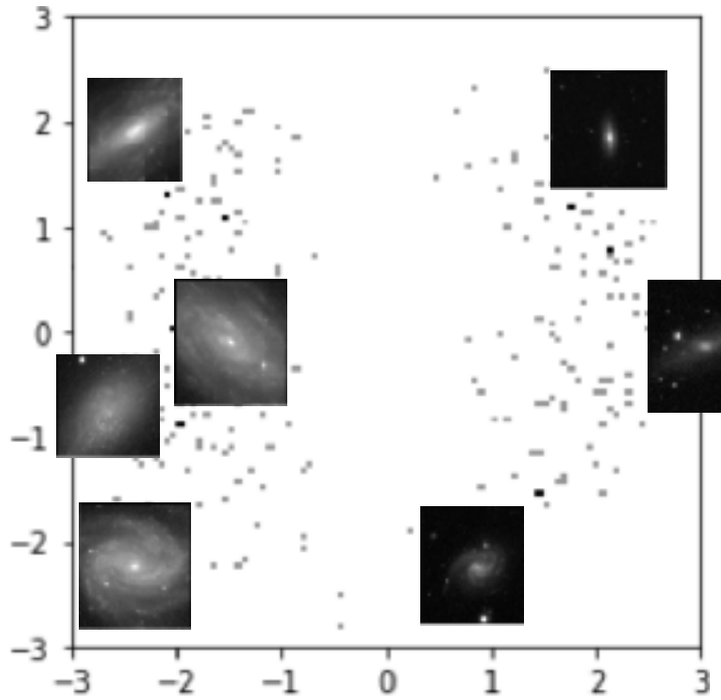
$$p_\theta(\mathbf{x}) \approx \pi_{data}(\mathbf{x})$$

Score Modeling

Recap: We have a way to sample any distribution $\pi(\mathbf{x})$ that does not require us to know Z, the normalization constant.
We only need to be able to evaluate its ***score***

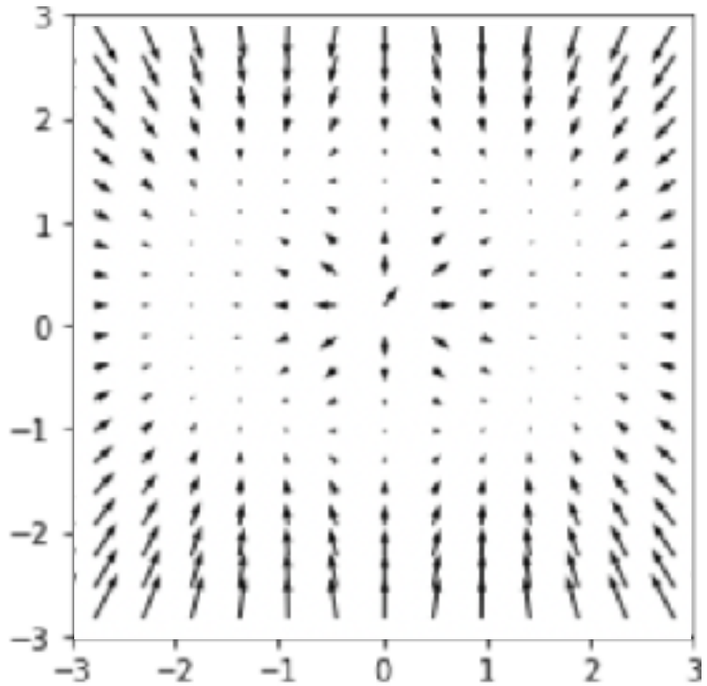
$$\mathbf{s}(\mathbf{x}) = \nabla \log(\pi(\mathbf{x}))$$

How can we do this from samples (e.g. data)?



Training data

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n,\} \overset{iid}{\sim} \pi_{data}(\mathbf{x})$$



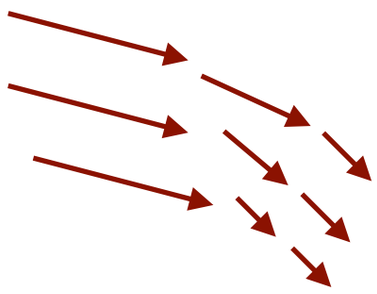
Score function

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla \log(\pi(\mathbf{x}))$$

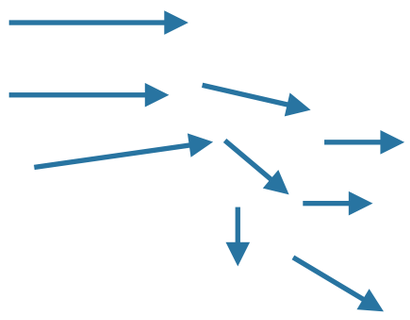
Score Modeling

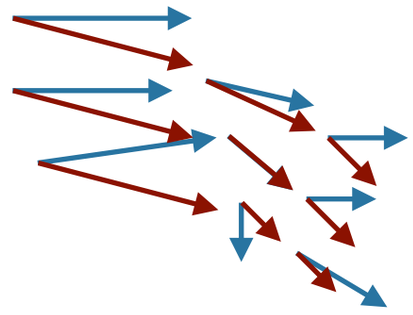
- **Given:** *i.i.d.* samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n,\} \overset{iid}{\sim} \pi_{data}(\mathbf{x})$
- **We want:** estimate the scores $\nabla \log(\pi_{data}(\mathbf{x}))$
- **Model:** A vector-valued NN with parameters $\theta : \mathbf{s}_\theta(\mathbf{x}) : R^n \rightarrow R^n$
- **Objective:** Write a (scalar) loss to compare vector fields?

$\nabla \log(\pi_{data}(\mathbf{x}))$



$\mathbf{s}_\theta(\mathbf{x})$





Difference and average the norm of the difference

Score Modeling

- **Given:** *i.i.d.* samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n,\} \overset{iid}{\sim} \pi_{data}(\mathbf{x})$
- **We want:** estimate the scores $\nabla \log(\pi_{data}(\mathbf{x}))$
- **Model:** A vector-valued NN with parameters $\theta : \mathbf{s}_\theta(\mathbf{x}) : R^n \rightarrow R^n$
- **Objective:** Loss to compare vector fields:

Mathematically, this is:

$$\frac{1}{2} E_{p_{data}(\mathbf{x})} \left[|| \nabla_x \log ? \pi_{data}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}) ||_2^2 \right]$$

BUT, using integration by parts and some more math magic, we can derive ***score matching***: (Hyvärinen 2005)

$$E_{p_{data}(\mathbf{x})} \left[\frac{1}{2} || \mathbf{s}_\theta(\mathbf{x}) ||_2^2 + \text{trace}(\nabla_x \mathbf{s}_\theta(\mathbf{x})) \right]$$

$$\approx \sum_{i=1}^{N} \left[\frac{1}{2} || \mathbf{s}_\theta(\mathbf{x}_i) ||_2^2 + \text{trace}(\nabla_x \mathbf{s}_\theta(\mathbf{x}_i)) \right]$$

Trace of the Jacobian of s: Not scalable, but can be efficiently approximated in high d

Score Matching

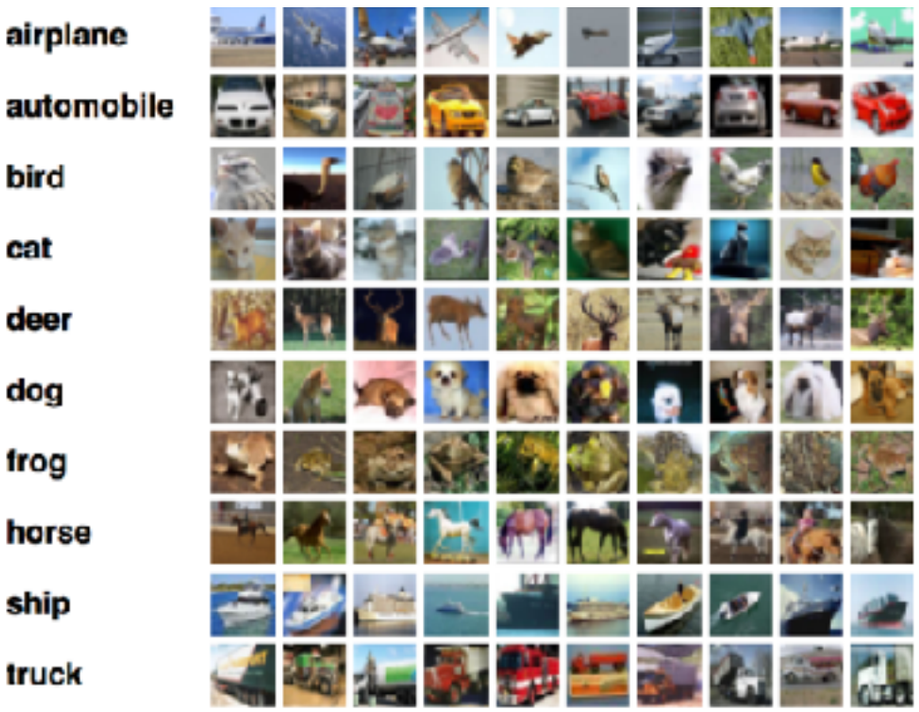
We are golden: we have a way to estimate the scores from data (samples of the data generating process) and with Langevin sampling we can sample from the underlying distribution without knowing anything else!!

Game plan

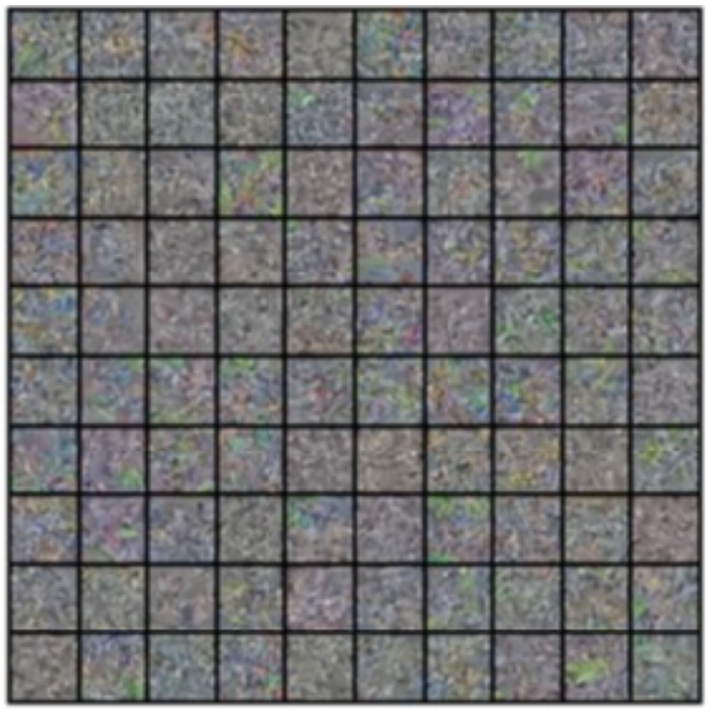
1. Back to the basics physics of diffusion and Brownian motion.
2. How to turn this into a sampling algorithm
3. Score matching
4. **How to actually make this work (denoising diffusion)**
5. Example of application: data driven prior for inverse problems.

Why aren’t we done??

CIFAR-10

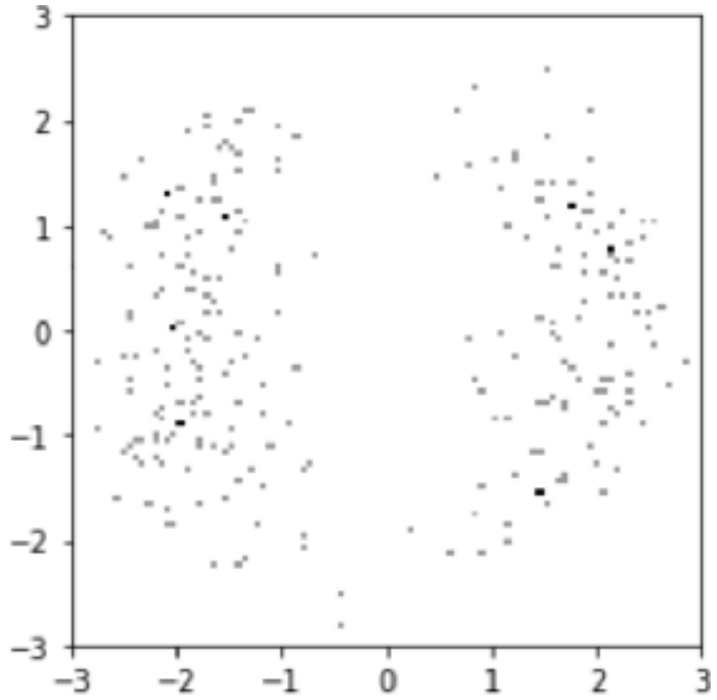


Samples

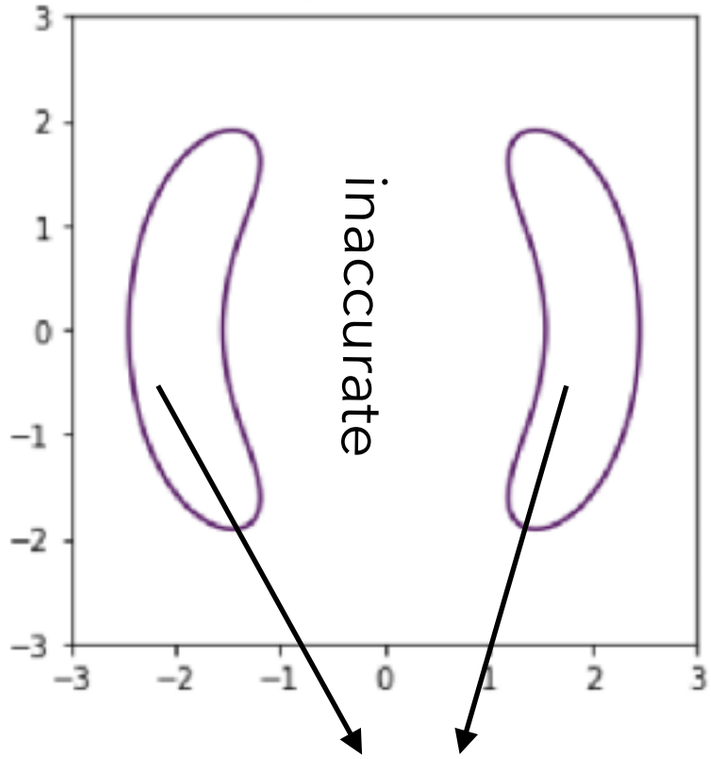


The scores we learn are only accurate in high density regions!

In low density regions, we don't have any training data, so it's as if there was no support in our underlying true distribution, and so the scores we learn there will be random.



$$\frac{1}{2}E_{p_{data}(\mathbf{x})}\left[\left|\left|\nabla_x \log \pi_{data}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\right|\right|_2^2\right]$$

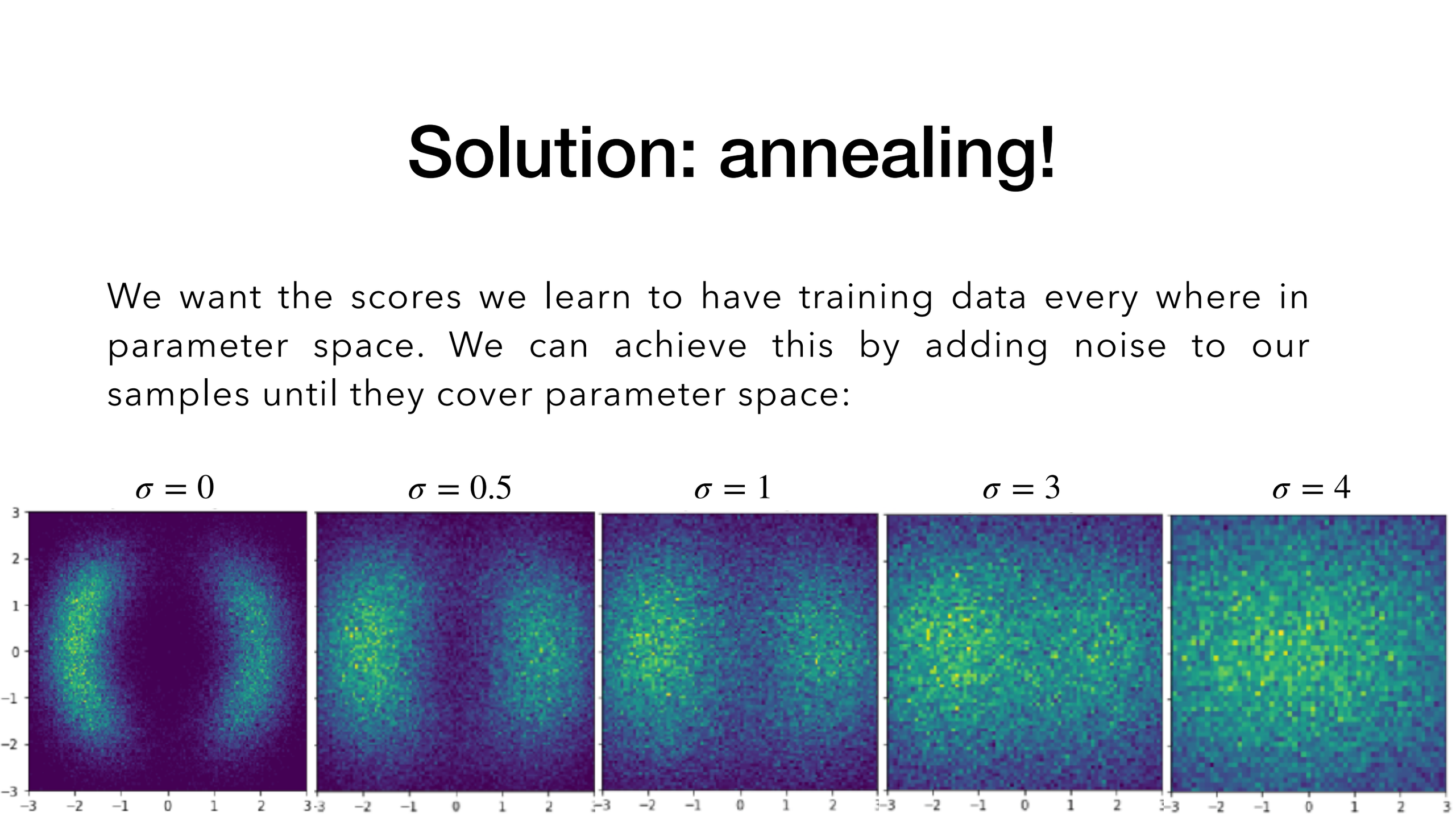


Accurate score gradients

The problem is that during the burn in phase, we don't know where the accurate regions are and we sample everywhere!!

Solution: annealing!

We want the scores we learn to have training data every where in parameter space. We can achieve this by adding noise to our samples until they cover parameter space:



Now if we learn $\mathbf{s}_\theta(\mathbf{x}, \sigma)$, a joint score model for all the different noise levels, we can get accurate scores everywhere that will lead to good, high probability samples!

Denoising Diffusion

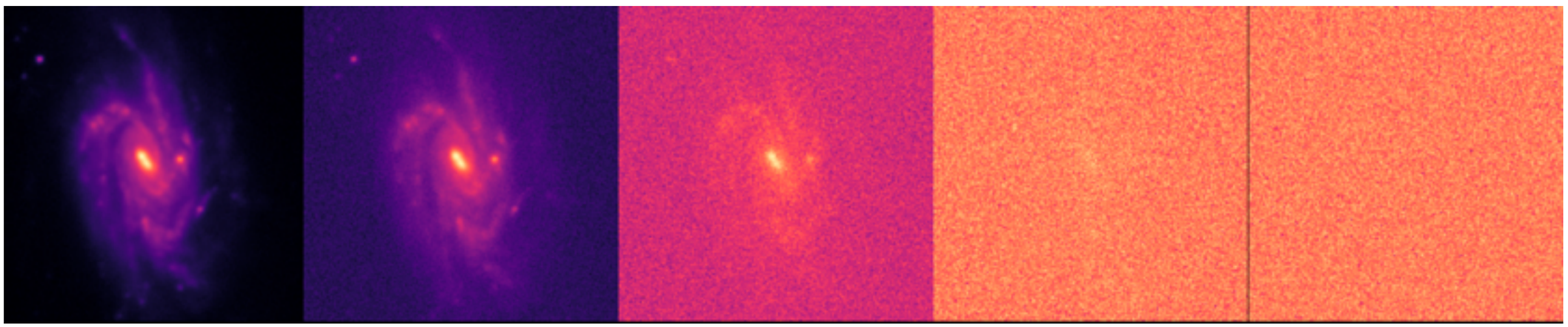
How do we know how many noise levels we should include?
This sounds like the best thing to do would be to include infinitely many
to ensure good transitions

But from what we've done before, we know exactly how to represent the process of noising up our samples with (almost certainly) continuous kicks:

With the Langevin equation!!

$$d\mathbf{x}(t) = \sigma(t)\, dW(t)$$

Target distribution → Add noise according to the forward Langevin equation → **Simple (Gaussian) distribution**



Use this to train a score model for every noise level, basically a UNet conditioned on σ

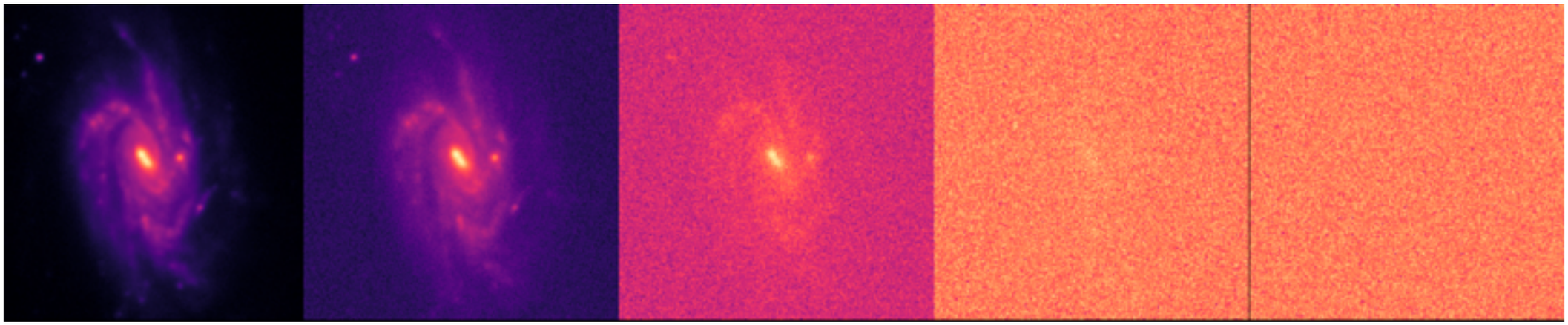
Denoising Diffusion

$$d\mathbf{x}(t) = \sigma(t)\, dW(t)$$

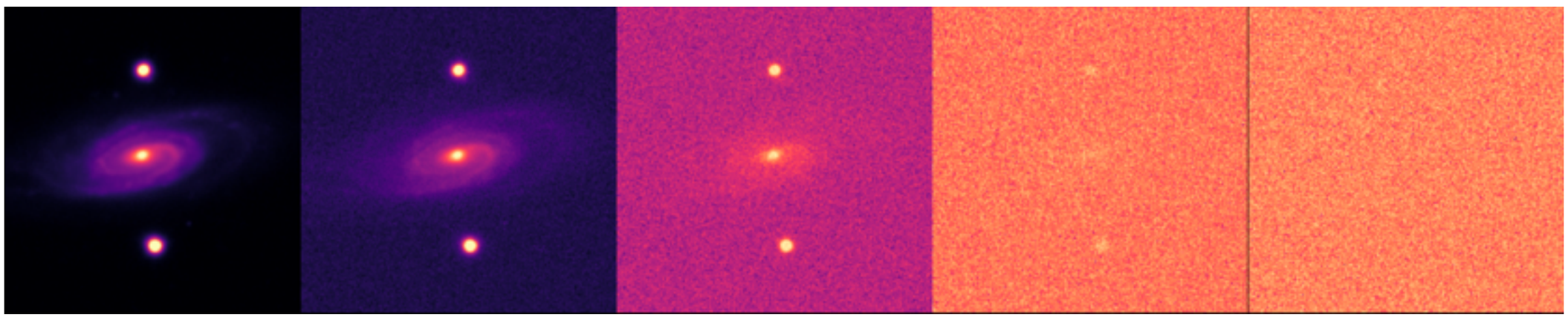
Target distribution

Add noise according to the forward Langevin equation

Simple (Gaussian) distribution



$$\frac{\partial P(\mathbf{x}, t)}{\partial t} = \frac{1}{2} \nabla^2 \left(\sigma(t)^2 P(\mathbf{x}, t) \right)$$



Target distribution

De-noise according to the backward Langevin equation

Simple distribution

$$d\mathbf{x}(t) = -\sigma(t)^2 \nabla_{\mathbf{x}} \log \left(\pi_{data}(x) \right) dt + \sigma(t)\, d\bar{W}(t)$$

Game plan

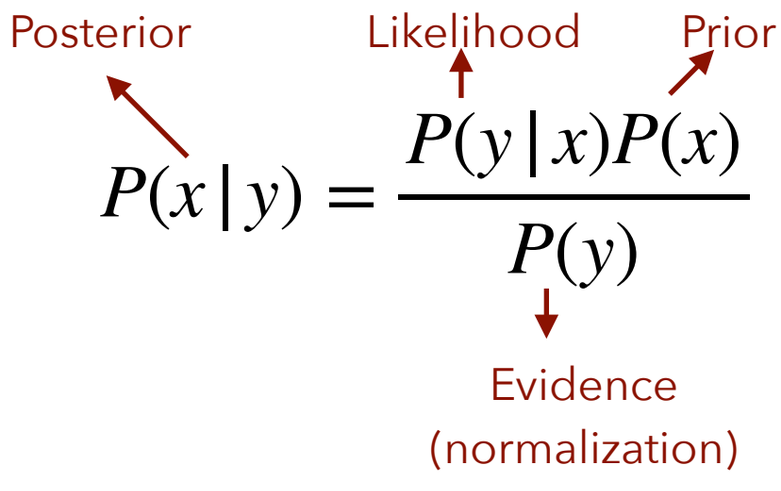
1. Back to the basics physics of diffusion and Brownian motion.
2. How to turn this into a sampling algorithm
3. Score matching
4. How to make this work (denoising diffusion)
5. **Example of application: data driven prior for inverse problems.**

Inverse problems

An inverse problem is a problem of the form

$$\mathbf{y} = f(\mathbf{x}) + z$$

Where we are interested in finding x given some observation y. Because the observations are noisy, and maybe also because f might not be invertible, some information has been destroyed and the best we can do it solve this equation probabilistically, using Bayes Theorem:

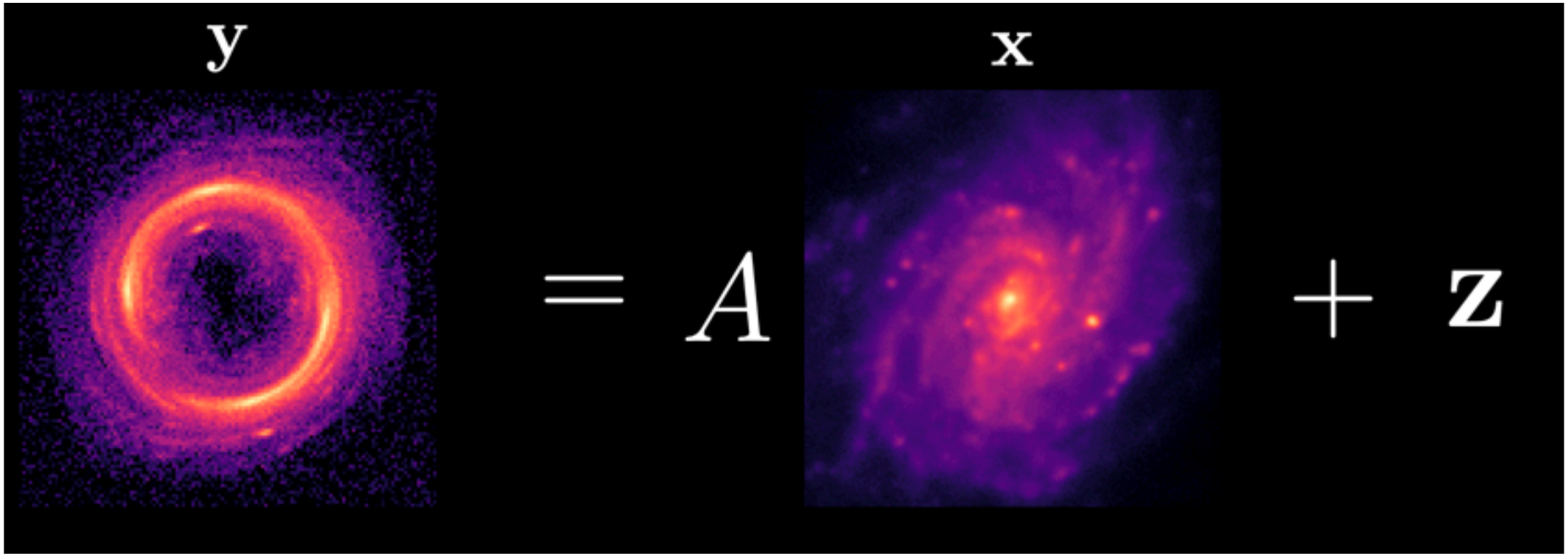


Linear inverse problems


Alexandre Adam

Let’s look specifically at linear inverse problem:

$$\mathbf{y} = A\mathbf{x} + z$$



Linear inverse problems

$$P(x\,|\,y) = \frac{P(y\,|\,x)P(x)}{P(y)}$$

Likelihood:

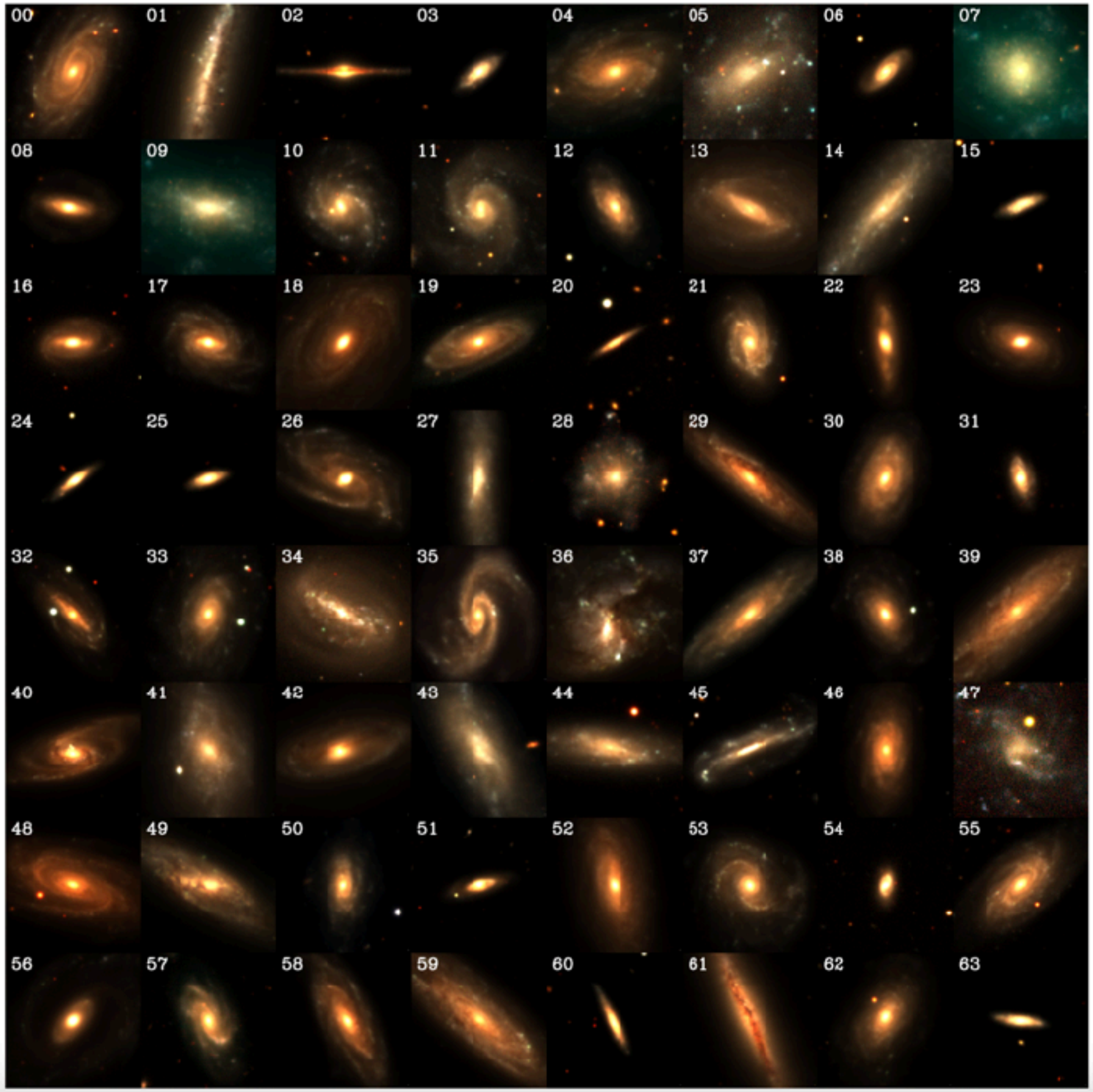
If we know the physical model, A, and know the noise statistics $p(z)$, we can write the likelihood since $\mathbf{y} - A\mathbf{x} \sim p(z)$.

Prior:

When we have low resolution data, a simple parametric model is sufficient. As data gets more complex, we need a more complex representation of what an unlensed galaxy looks like => learn it from data! We can learn the score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$!

We know how to use this to sample from $p(\mathbf{x})$, solving:

$$d\mathbf{x}(t) = -\,\sigma(t)^2\,\nabla_{\mathbf{x}} \log\left(p(x)\right)\,dt + \sigma(t)\,d\bar{W}(t)$$



Connor Stone

http://www.mjjsmith.com/thisisnotagalaxy/

Posterior sampling

Alexandre Adam

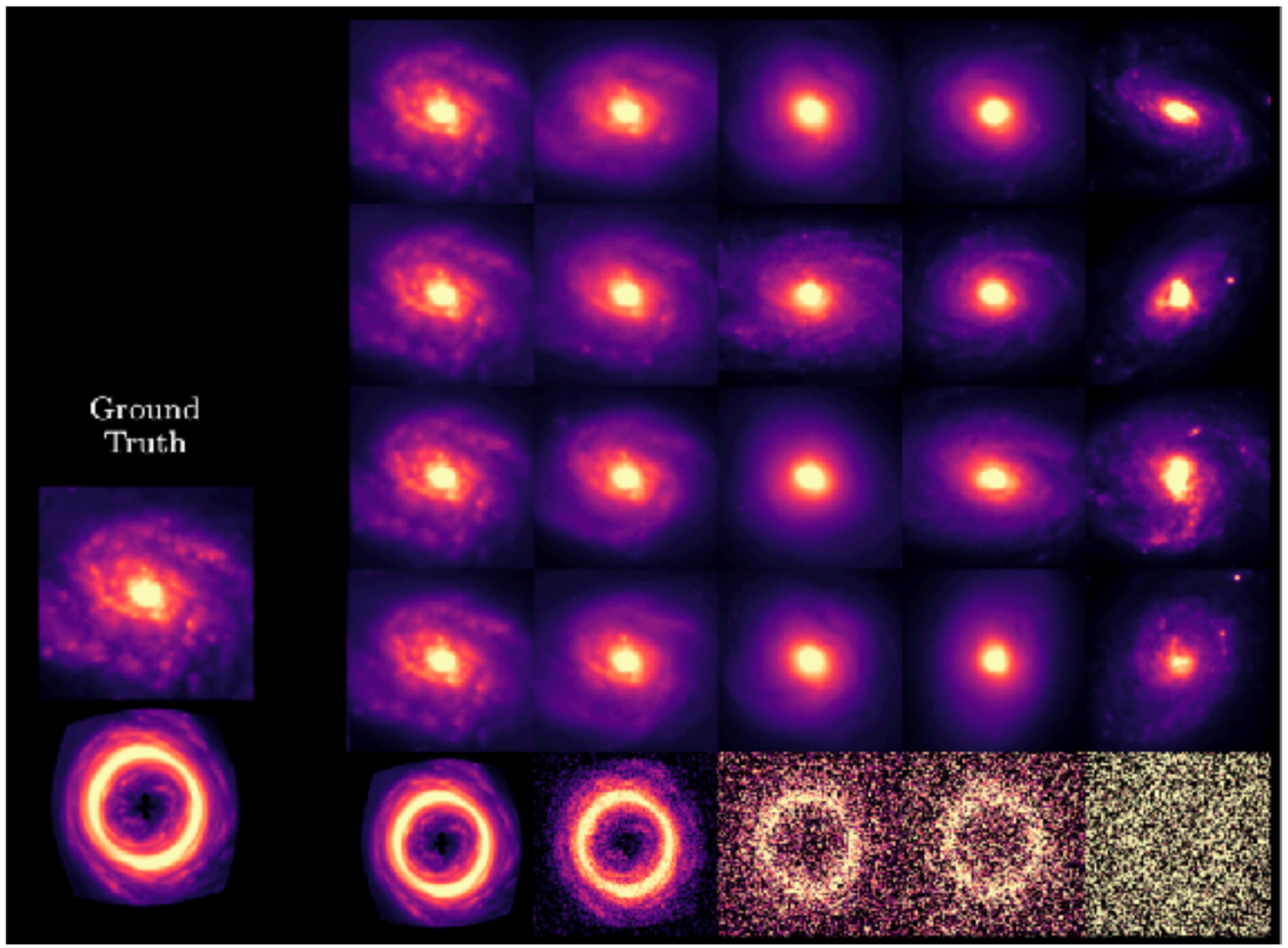
But what if we wanted to sample from the posterior instead…

$$\nabla_{\mathbf{x}} \log P(x \mid y) = \nabla_{\mathbf{x}} \log P(y \mid x) + \nabla_{\mathbf{x}} \log P(x) - \nabla_{\mathbf{x}} \log P(y)$$

To a good approximation, we can calculate the likelihood score analytically if we assume it's Gaussian and we know A.

This is the score we learnt with Score Matching!

=0 because it does not depend on x, as before

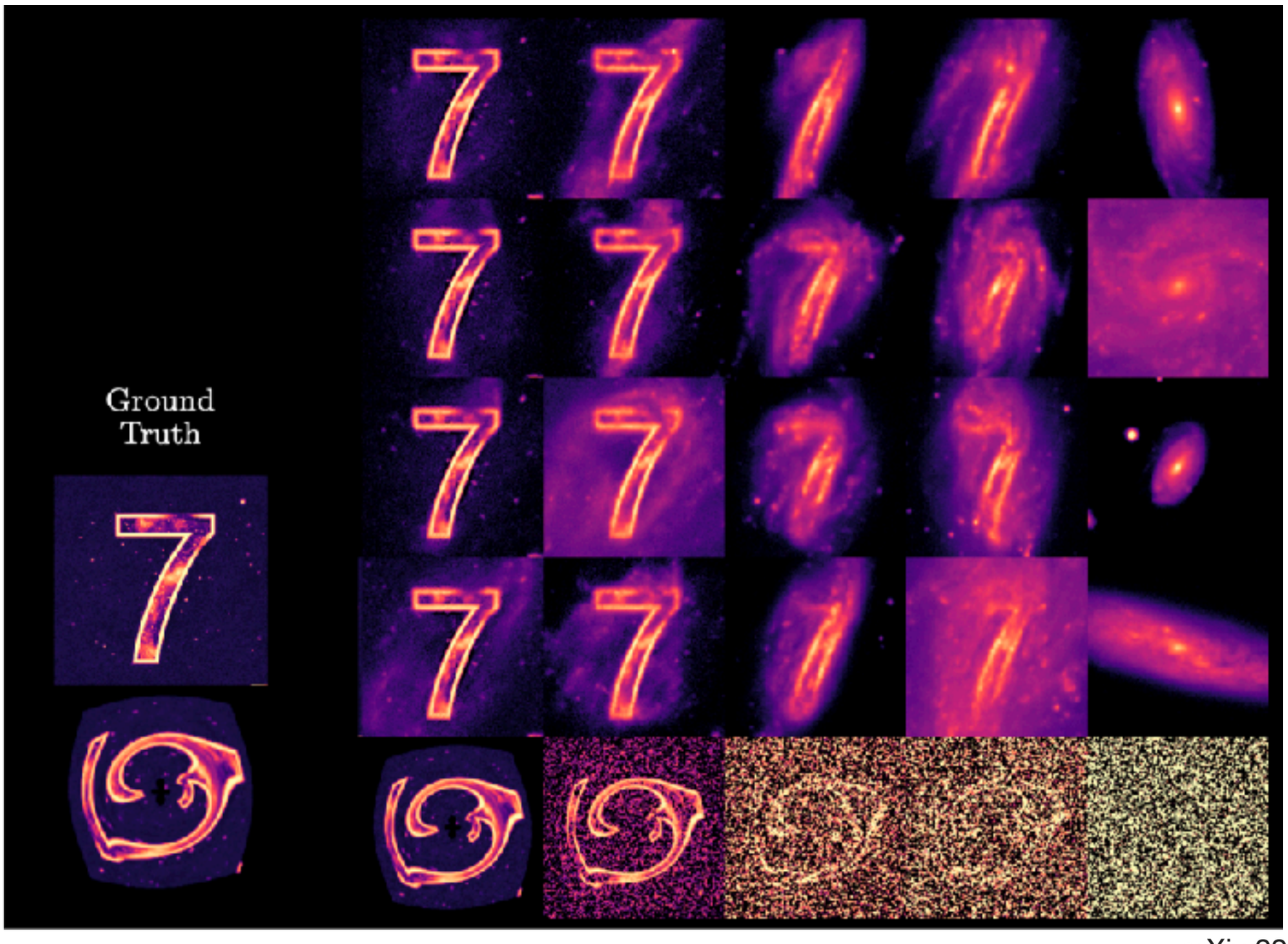


arXiv:2211.03812 [astro-ph.IM]

Out of Distribution Tests

Alexandre Adam

$$\nabla_{\mathbf{x}} \log P(x \mid y) = \nabla_{\mathbf{x}} \log P(y \mid x) + \nabla_{\mathbf{x}} \log P(x)$$

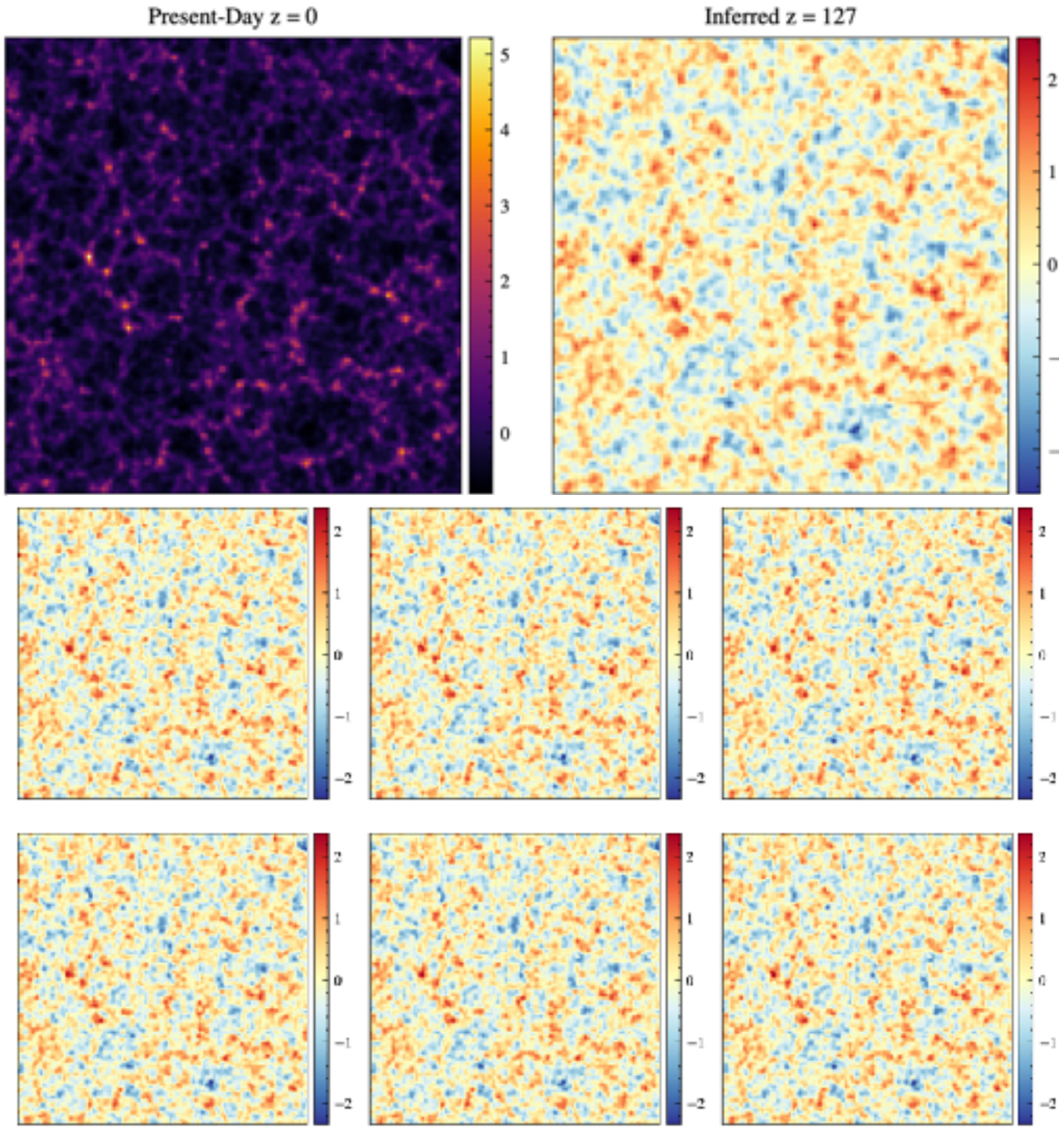


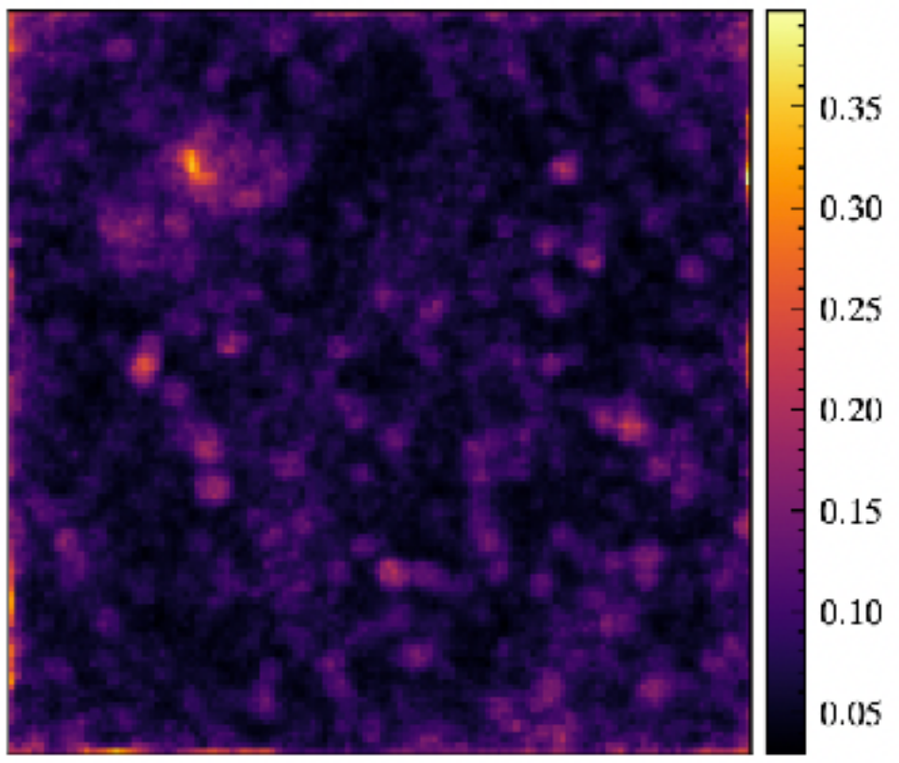
arXiv:2211.03812 [astro-ph.IM]

Inferring the Cosmological Parameters and Initial Conditions:

With Score-Based Diffusion Models

Ronan Legin



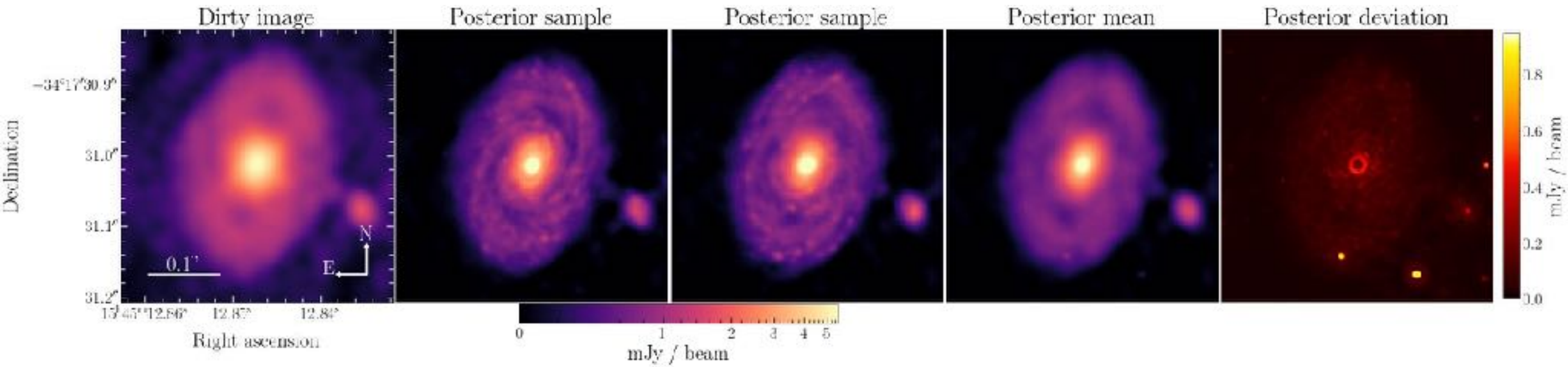


R Legin et al. 2023, arXiv:2304.03788

PSF-DECONVOLUTION AND 'DE-NOISING' FOR INTERFEROMETRIC DATA (PRELIMINARY)

With Score-Based Diffusion Models

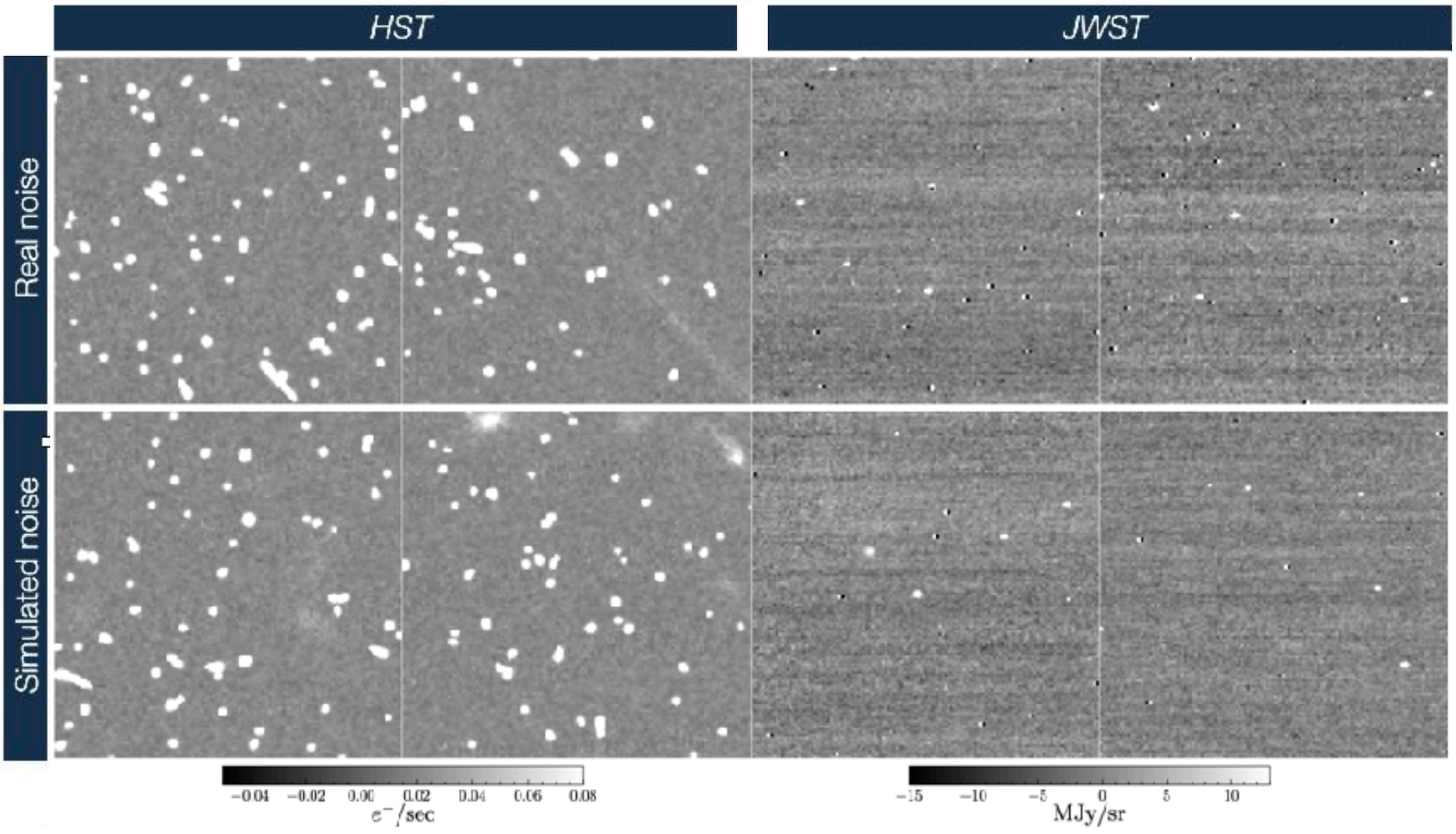
Noé Dia



Dealing With Realistic Noise: Beyond Gaussianity

Alexandre Adam

Ronan Legin



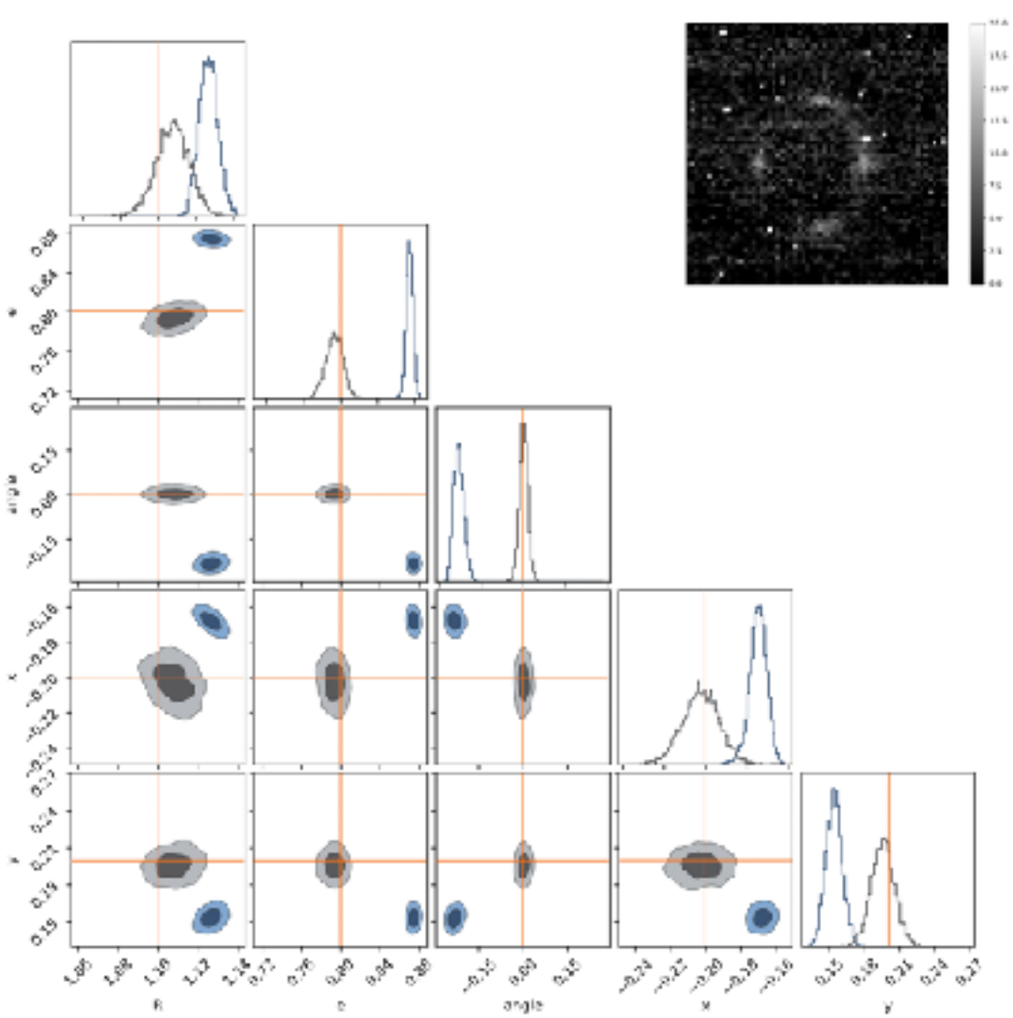
Legin, Adam, et al. 2302.03046

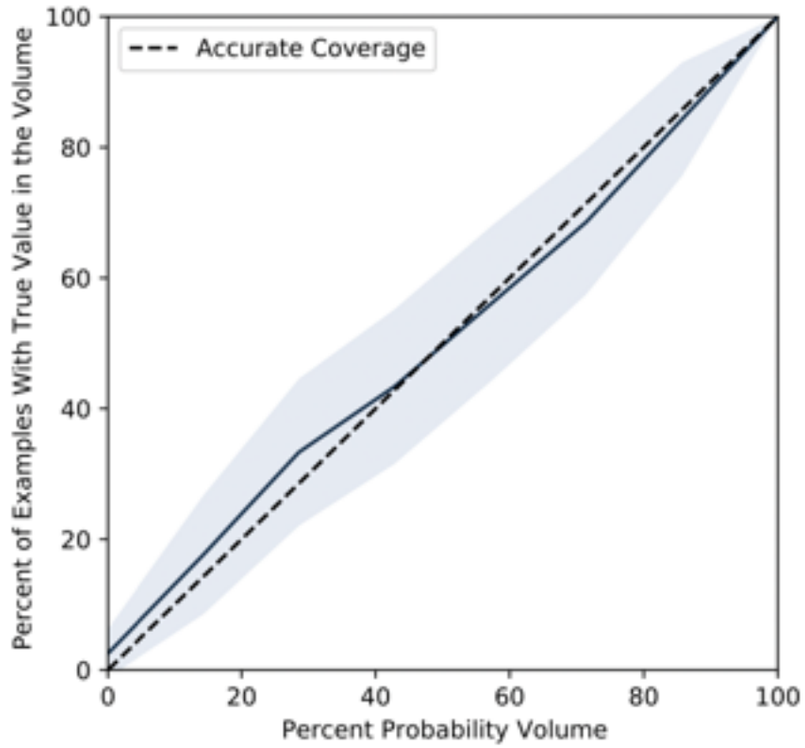
Dealing With Realistic Noise: Beyond Gaussianity

$$\mathbf{s}(\mathbf{x}_0) = \partial \log Q(\mathbf{x}_0)/\partial \mathbf{x}$$

$$P(\mathbf{x}_O|\eta) = Q(\mathbf{x}_O - \mathbf{M}(\eta))$$

$$\eta_{i+1} = \eta_i + \tau \boldsymbol{\nabla}_{\mathbf{x}} \log Q(\mathbf{x}_o - \mathbf{M}(\eta)) \boldsymbol{\nabla}_{\eta} M(\eta_i) + \sqrt{2\tau}\xi$$





Legin, Adam, et al. 2302.03046

Thank You!