June 18th 2026

Aurelio Amerio – IFIC, CSIC-UV, Spain

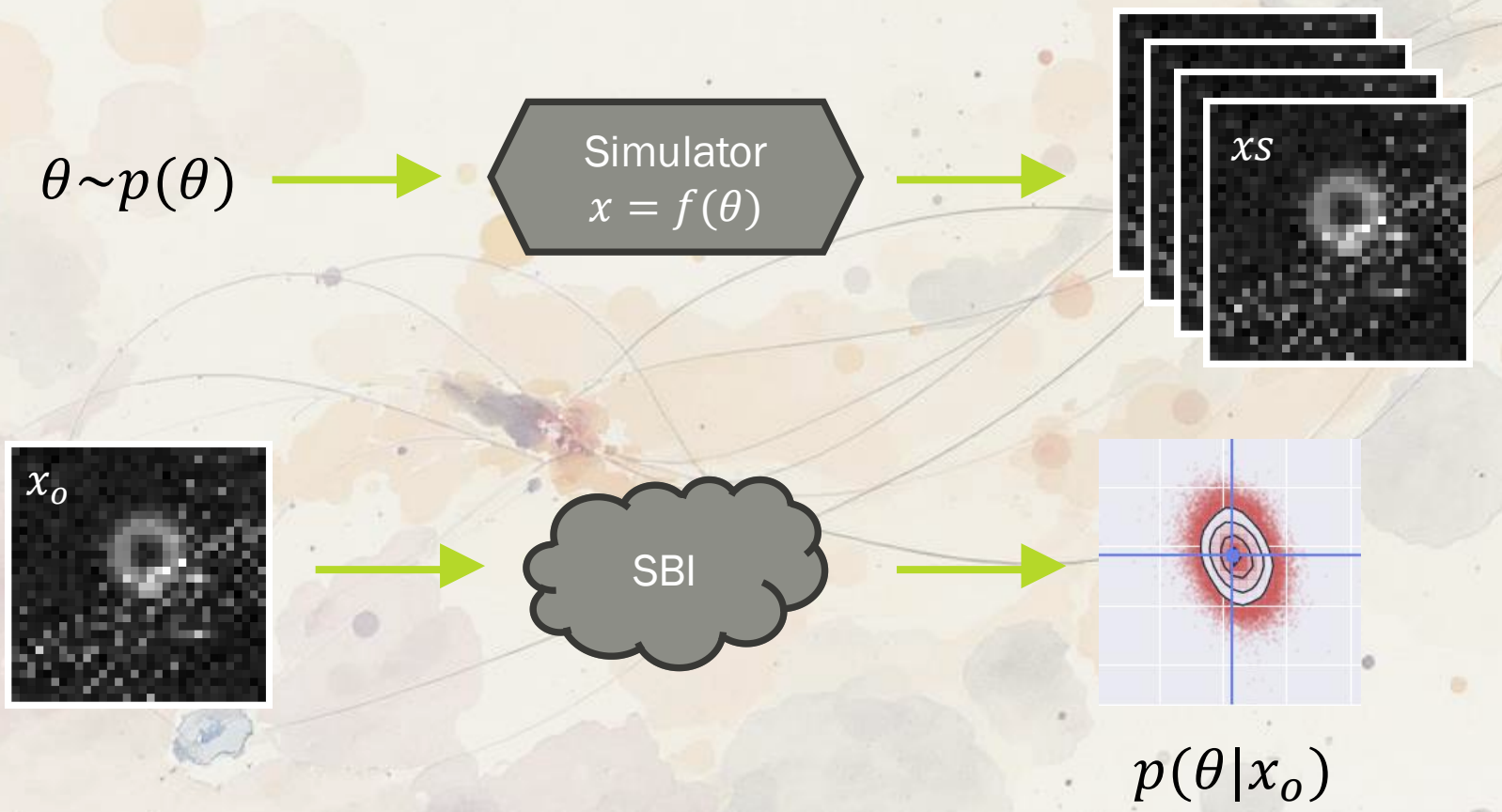
Non-Gaussian Universe

gensbi.com

AA, arXiv:2605.27499

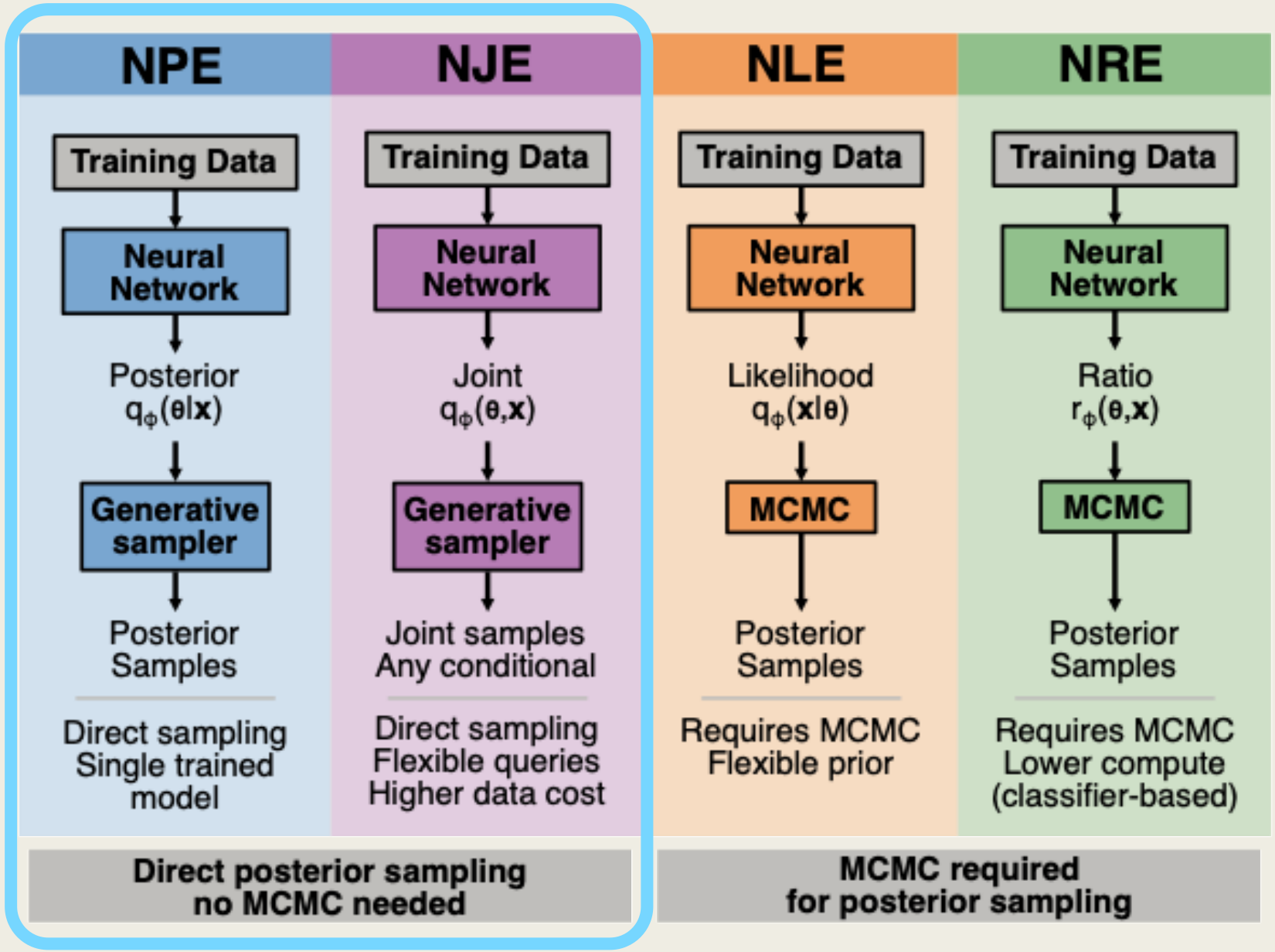
CSIC
Consejo Superior de Investigaciones Científicas
Universitat de València
IFIC Institut de Física Corpuscular
Excelencia Severo Ochoa

Simulation-Based Inference



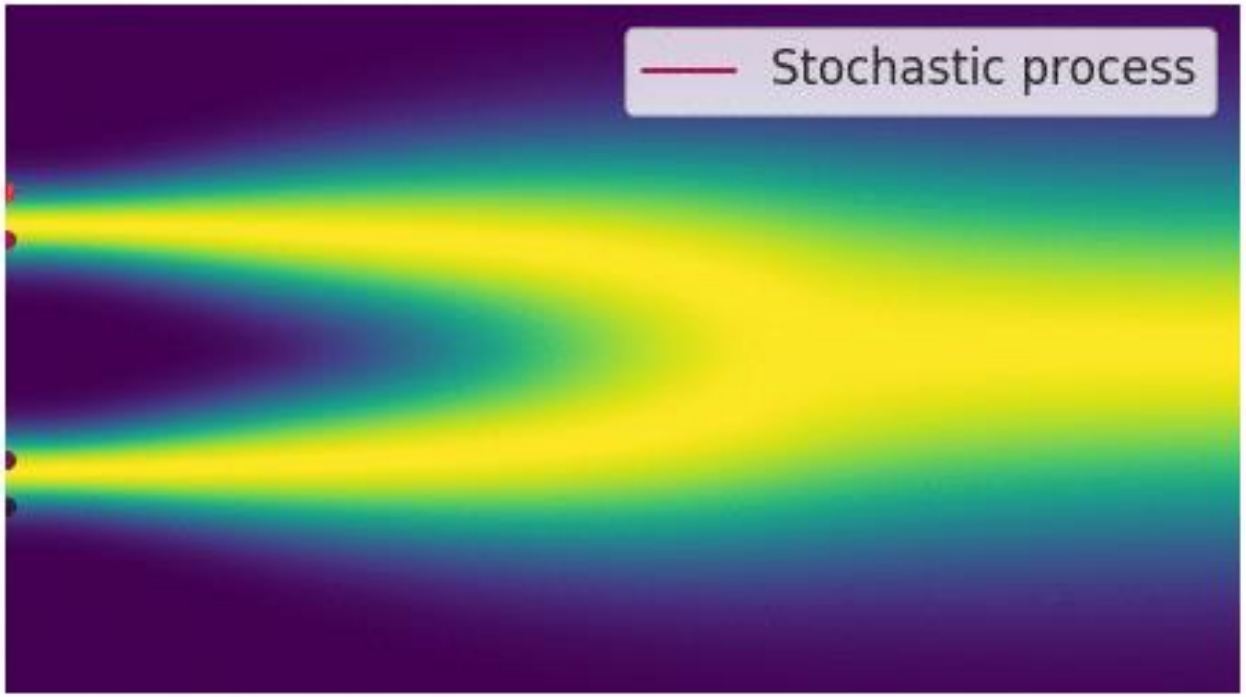
Simulation-Based Inference

- Learn to **reconstruct real-world physical parameters** using simulated data
- **Solves the inverse problem**, especially in cases of **intractable likelihoods**
- Is "**amortized**": once the model is trained, **no further simulations are required**

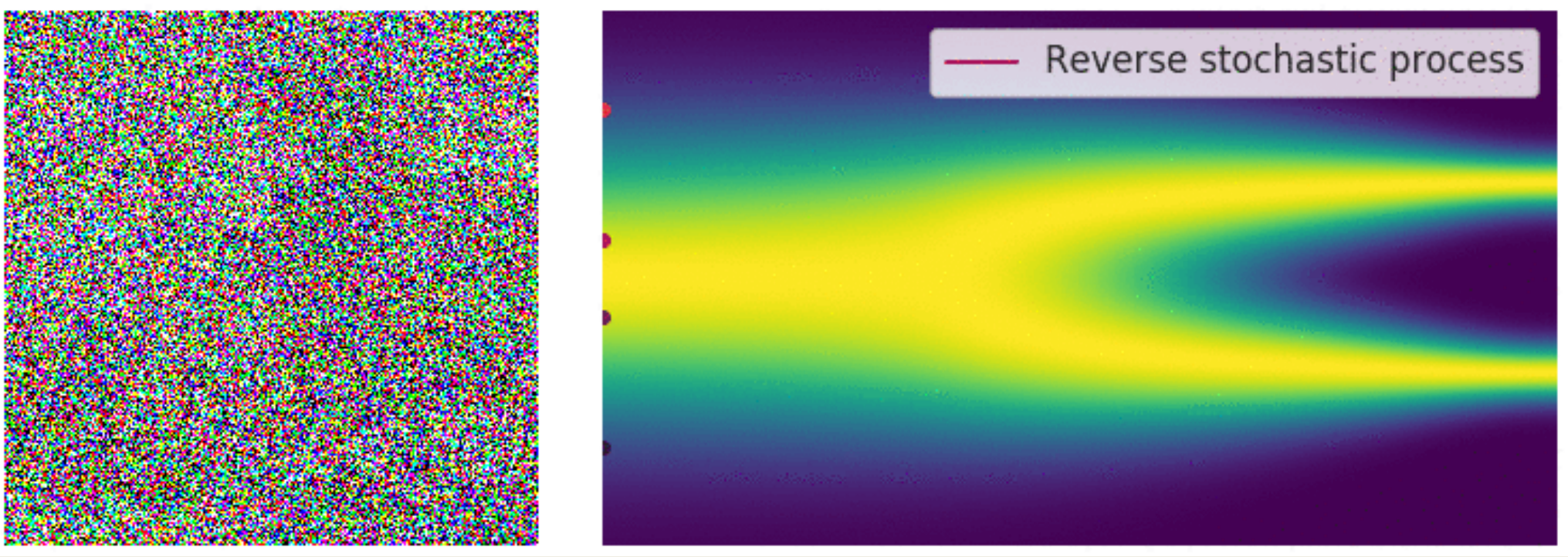


- GenSBI can be used for fast NPE and NJE
- NLE is supported, but explicit likelihood evaluation requires solving an ODE

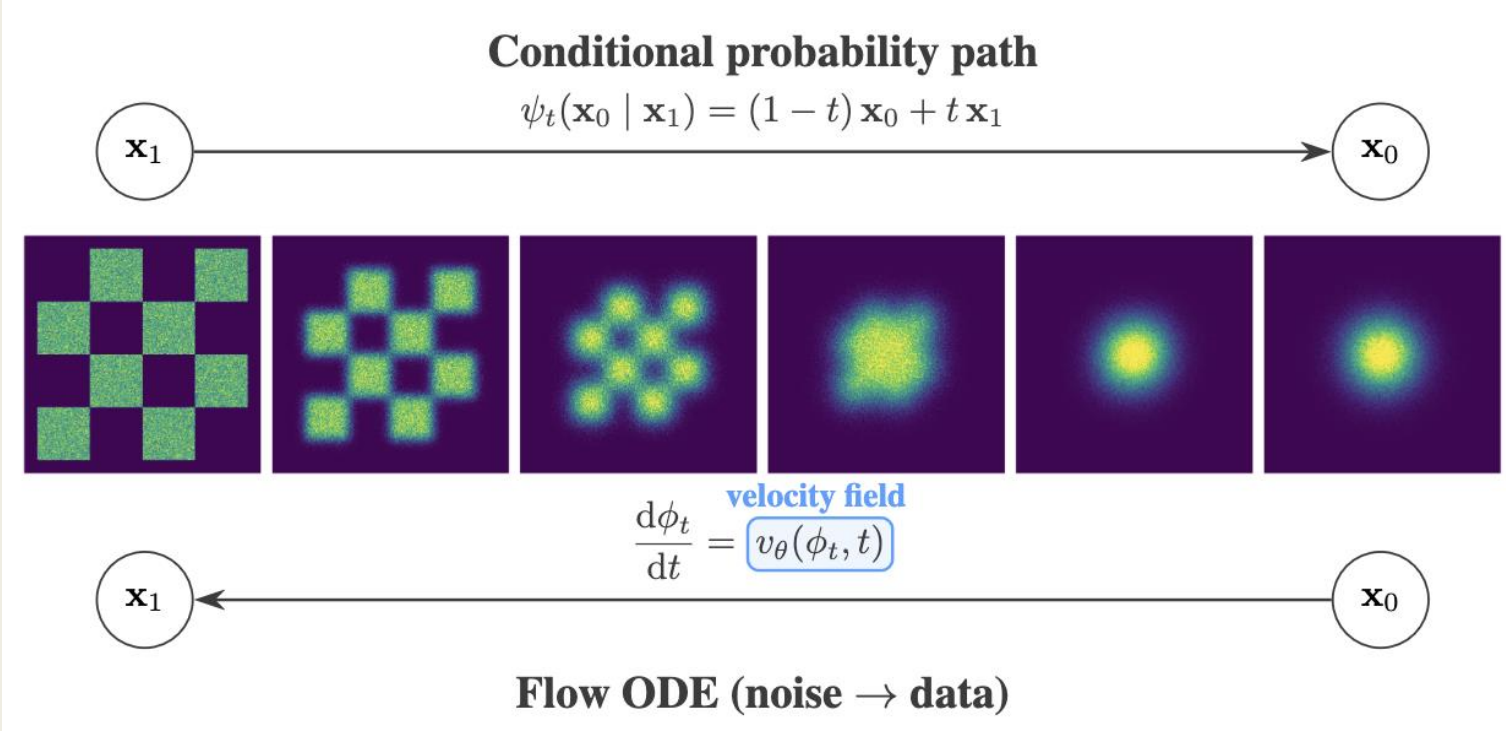
Diffusion models (in one slide)

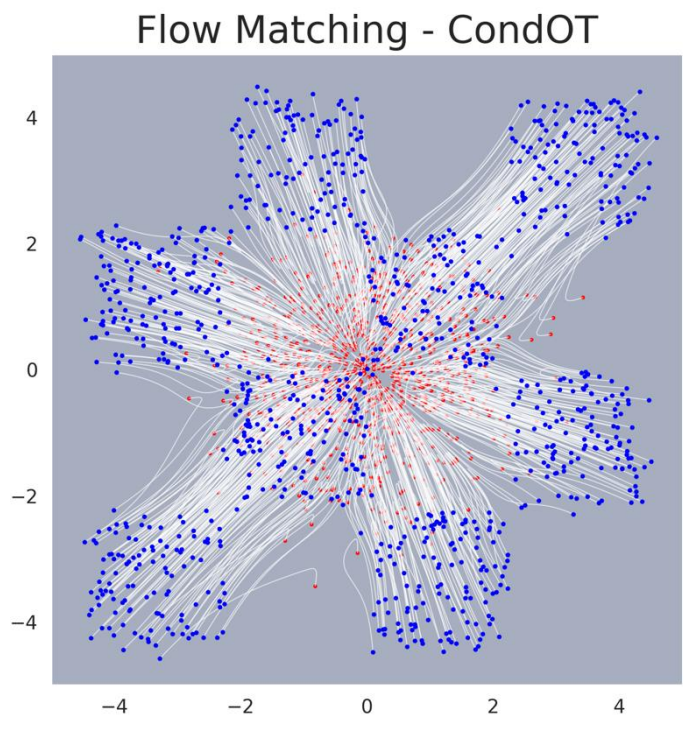


Diffusion models (in one slide)



Continuous normalizing flows





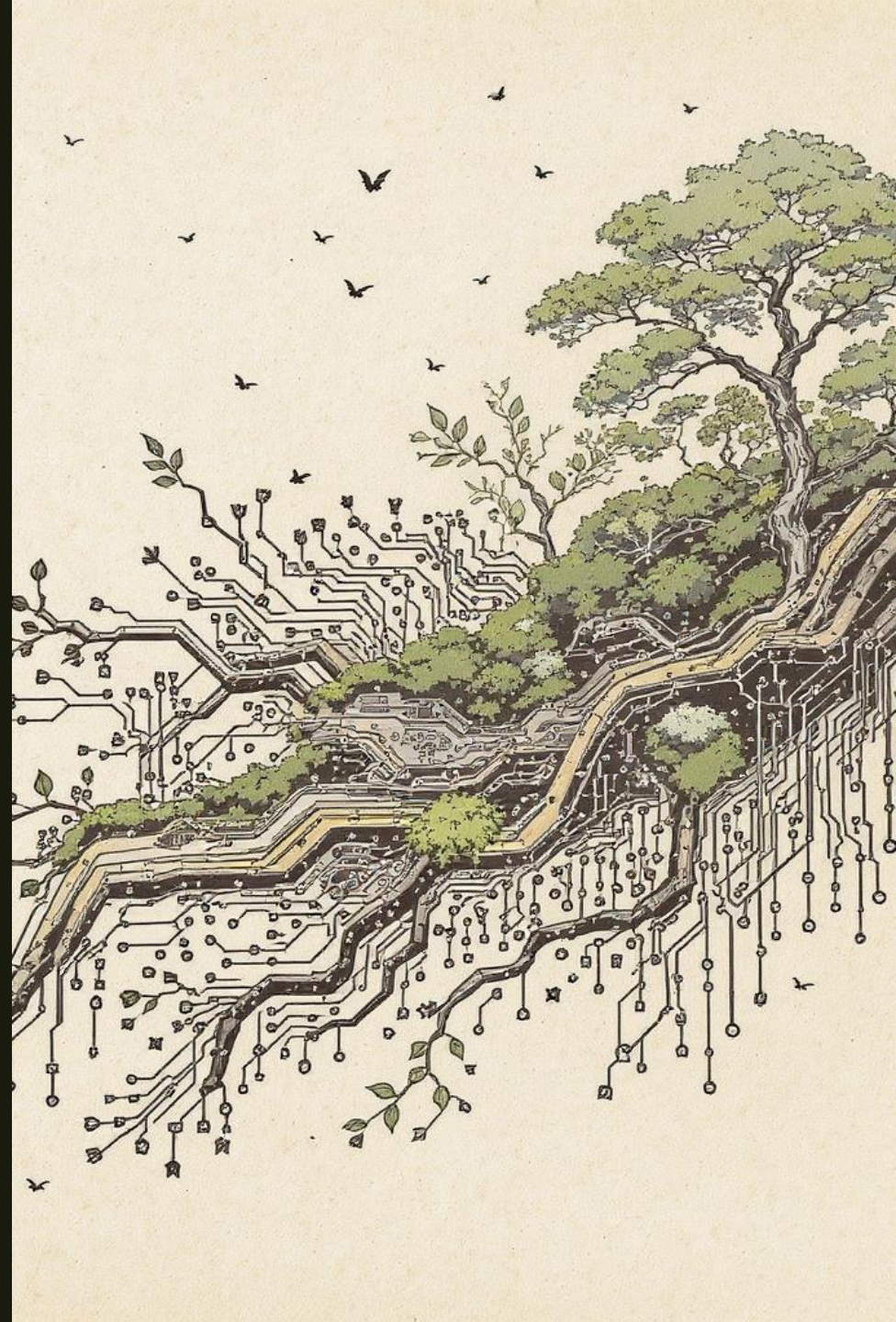
$$\mathcal{L}_{\mathcal{CFM}}(\theta) = \mathbb{E}_{\mathrm{t,\,q(x_1),\,p_t(x|x_1)}} \|v_\theta(x,t) - u_t(\,x \mid x_1\,)\|^2$$

$$u_t(\,x \mid x_1\,) = x_1 - x$$

gensbi.com
AA, arXiv:2605.27499

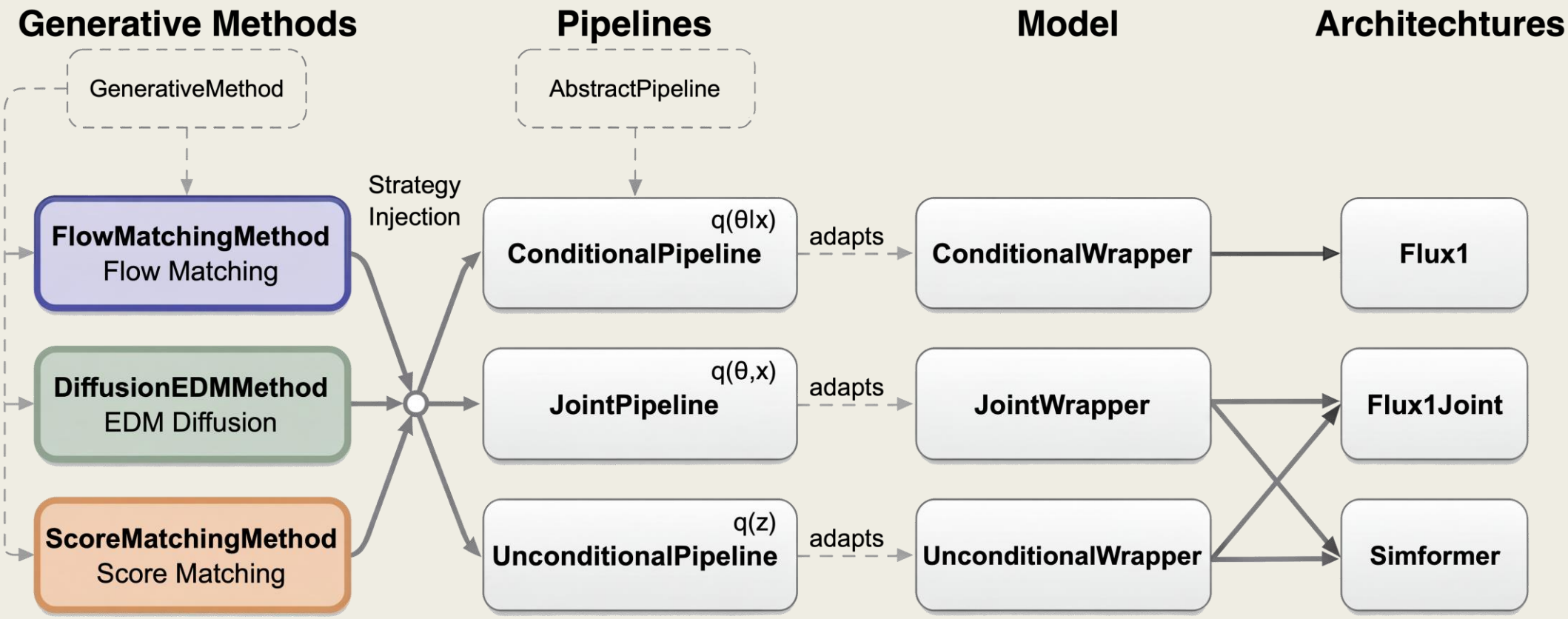
Why GenSBI?

- Few SBI libraries implementing Flow and Diffusion models* (none in pure JAX)
- FM/SM code mainly associated with research papers or proofs of concept → bridge to SBI
- Highly efficient implementations exist, but mostly in the context of image generation
- Bridge SOTA generative methodologies and physics research

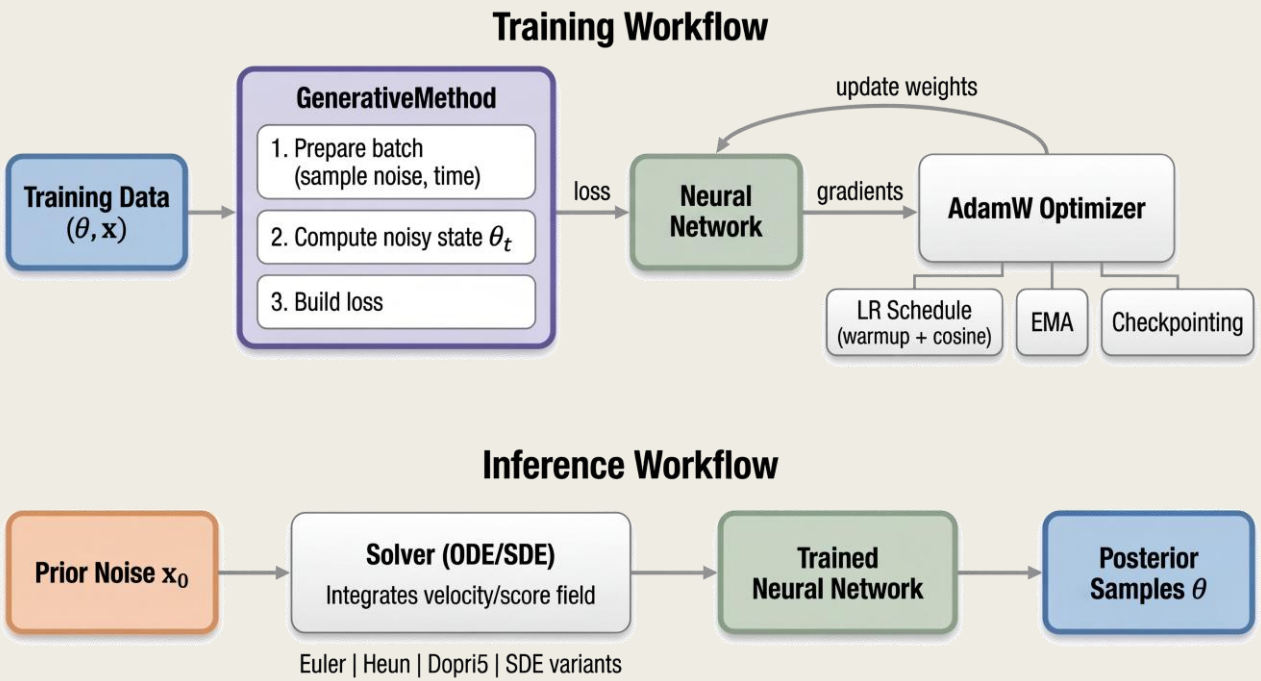


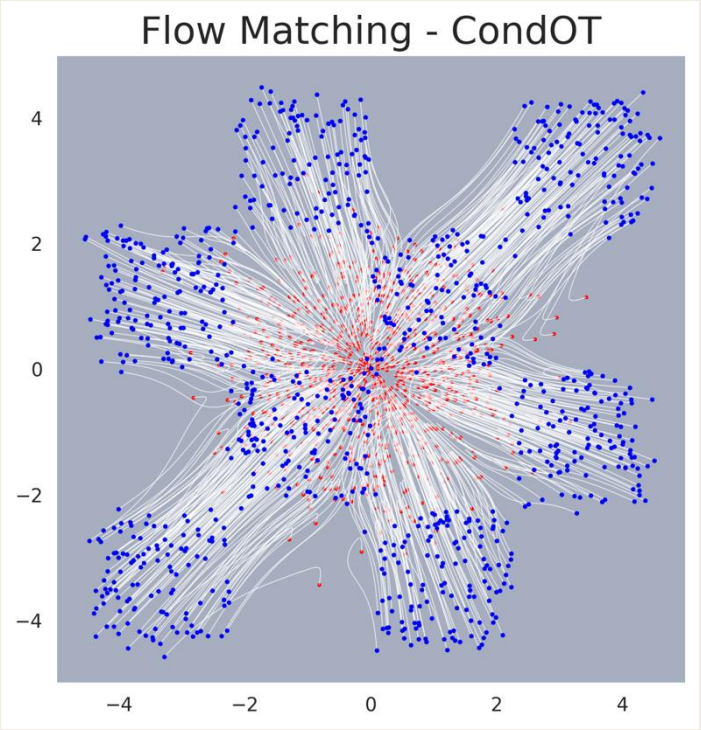
* Ask me for more info about how GenSBI sits in the SBI landscape

Modularity as a core principle



Modularity as a core principle





What's in the box

SOTA transformers*: Flux1 · SimFormer · Flux1Joint

Few-line recipes for training & sampling

Calibration built in: SBC · TARP · LC2ST · marginals

A real embedding system → batteries included!

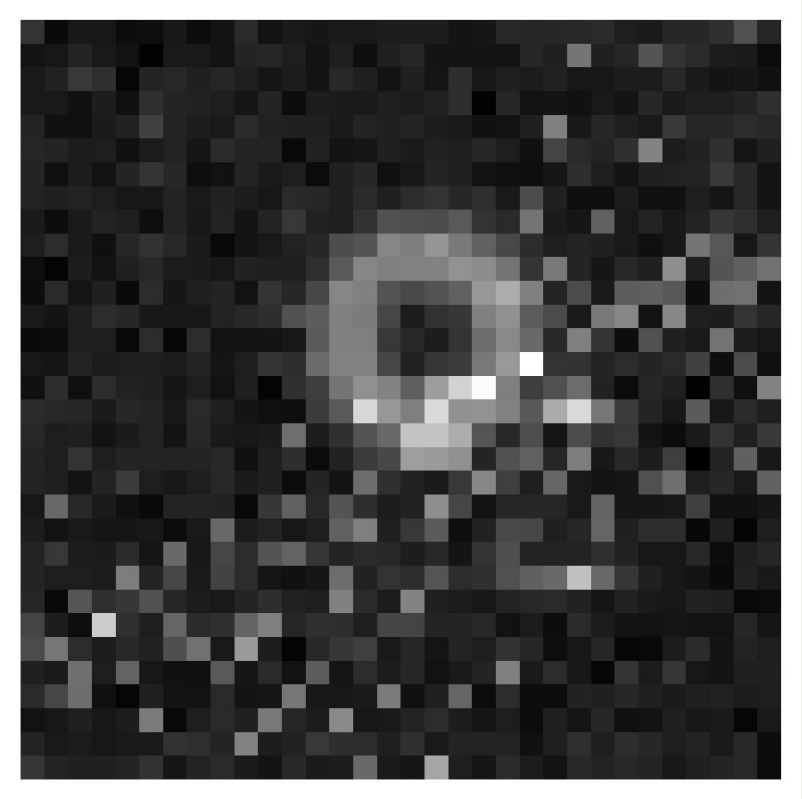
Interchangeable solvers + post-training SDE↔ODE swap

GenSBI-examples: companion repo of solved, ready-to-run examples

* Ask me about what makes them *special*

A TOY EXAMPLE: STRONG LENSING

A toy example: strong lensing



The task

- A toy **32×32 image** of a strong-lensing system → infer **2 physical parameters** of the lens
  (radius and width of the ring)

Why it matters

- Small enough to train and sample in minutes
- A clean stand-in for **any image / map → parameters** inference

The dataset

gensbi — python

```python
# grain: fast, JAX-native data loading
import grain
from gensbi_examples.tasks import GravitationalLensing

task = GravitationalLensing()

df_train = task.df_train  # any dataset; we use HuggingFace and Parquet
df_val = task.df_val

train_dataset = (
    grain.MapDataset.source(df_train)
    .shuffle(42).repeat().to_iter_dataset()
    .batch(batch_size)
)

val_dataset = (
    grain.MapDataset.source(df_val)
    .shuffle(42).repeat().to_iter_dataset()
    .batch(256)
```

The model

gensbi — python

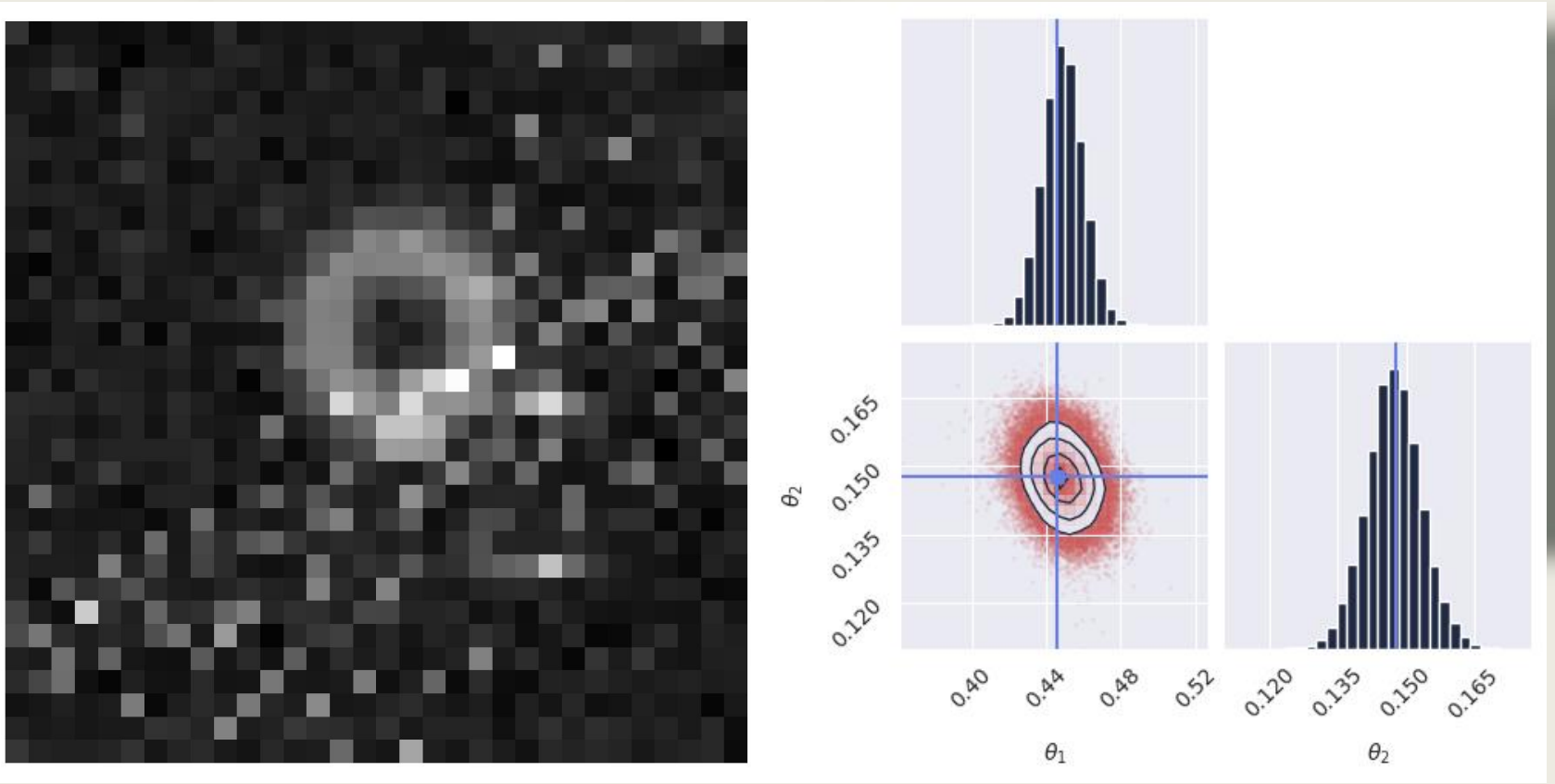
```python
from gensbi.recipes import ConditionalPipeline
from gensbi.core import FlowMatchingMethod
from gensbi.experimental.models.autoencoders import AutoEncoder2D
from gensbi.experimental.models.embedder import Embedded2DModel
from gensbi.models import Flux1

# build the component models
model_sbi = Flux1(params_flux)
vae_model = AutoEncoder2D(ae_params)

# small wrapper: CNN encoder compresses the image, Flux1 does inference
model = Embedded2DModel(vae_model, model_sbi)     # trained end-to-end

pipeline = ConditionalPipeline(
    model, train_dataset, val_dataset,
    dim_obs=2,                    # 2 lens parameters
    dim_cond=(8, 8), ch_cond=64, # patchified VAE latent
    method=FlowMatchingMethod(),
    id_embedding_strategy=("absolute", "rope2d"),  # keep 2D structure
)
```

Training and sampling

gensbi — python

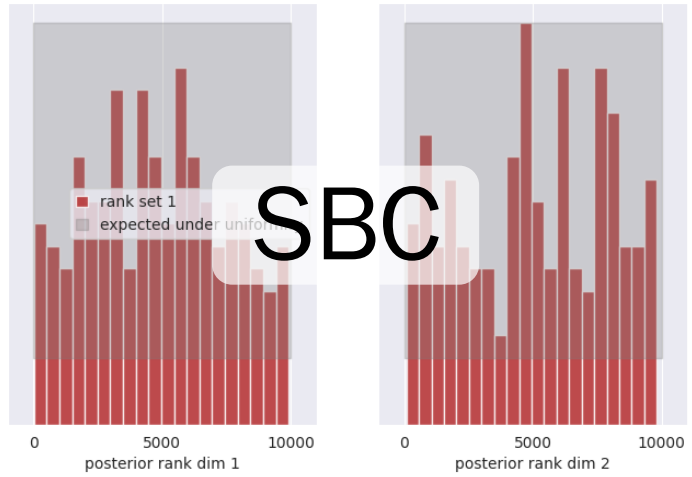
```python
from gensbi.utils.plotting import plot_marginals

# train the model
pipeline.train(rngs=nnx.Rngs(0))

# x_observed is some observation for which we want to sample from p(θ|x)
samples = pipeline.sample(rng, x_observed, 100_000)

# plot the samples
plot_marginals(samples, true_param=theta_true, gridsize=30)
```

Training and sampling

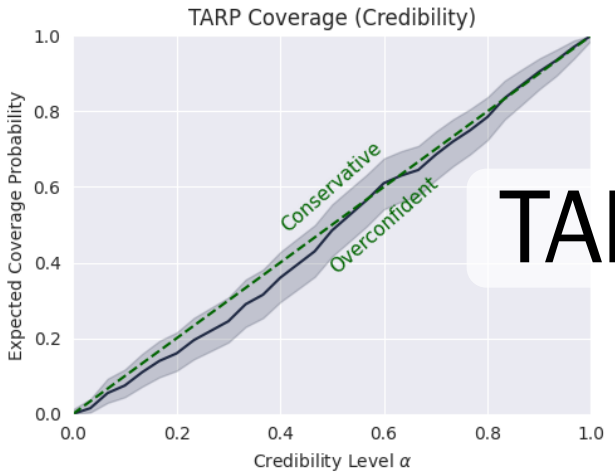


Check the full example

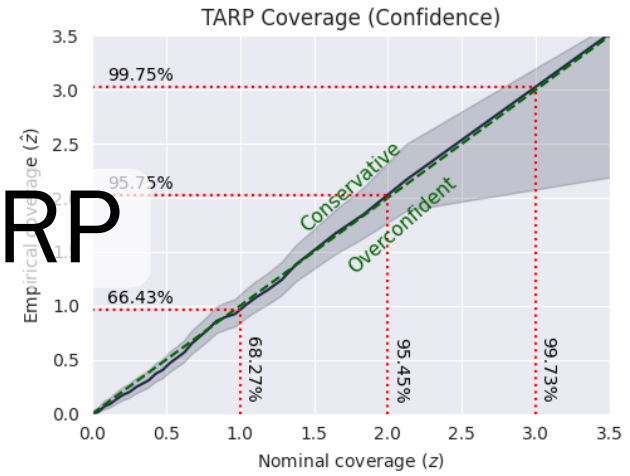
Open in Colab

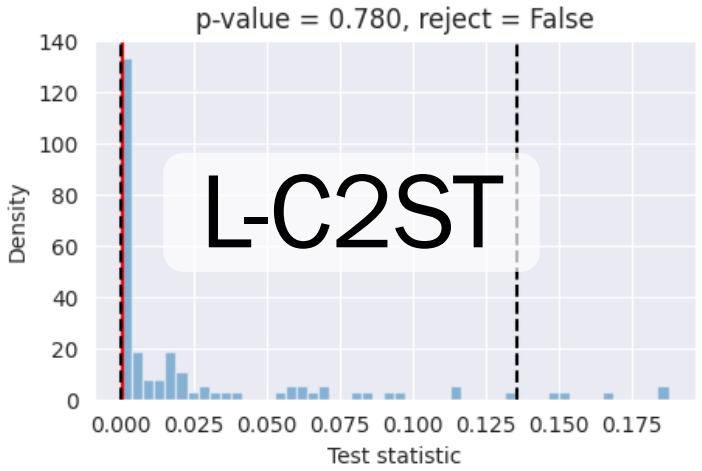


SBC

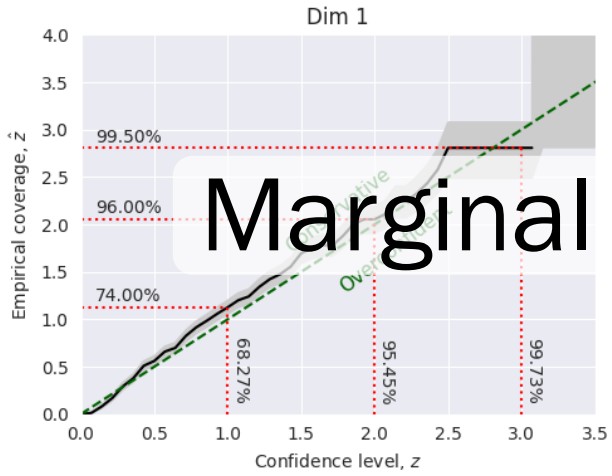


TARP

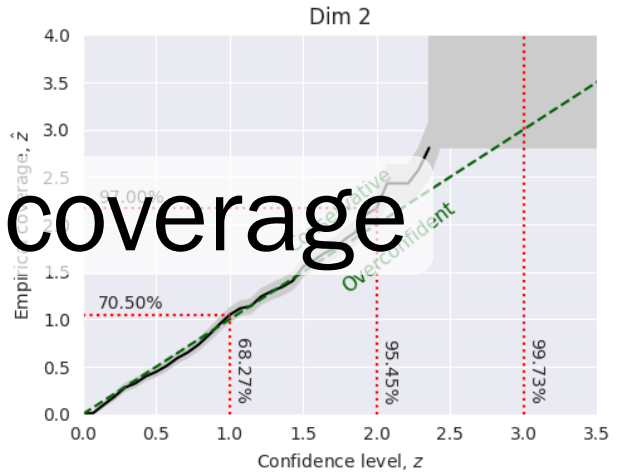




L-C2ST



Marginal coverage



AA, arXiv:2605.27499

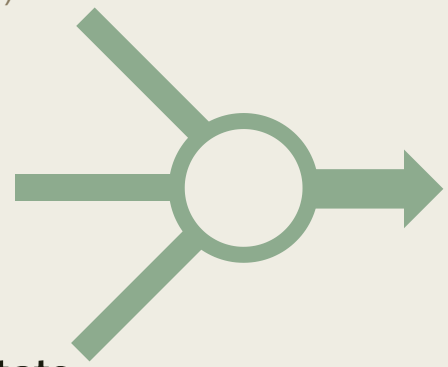
Batteries included: any data, out of the box

Works out of the box on images, time series, unstructured data & summary statistics — same pipeline, just swap the embedding.

Images / maps
patchify + 2D **RoPE** (ViT-style*)

Time series
1D-CNN + **sinusoidal PE**

Unstructured / summary stats
per-feature tokens and **learned embedding**



GenSBI engine

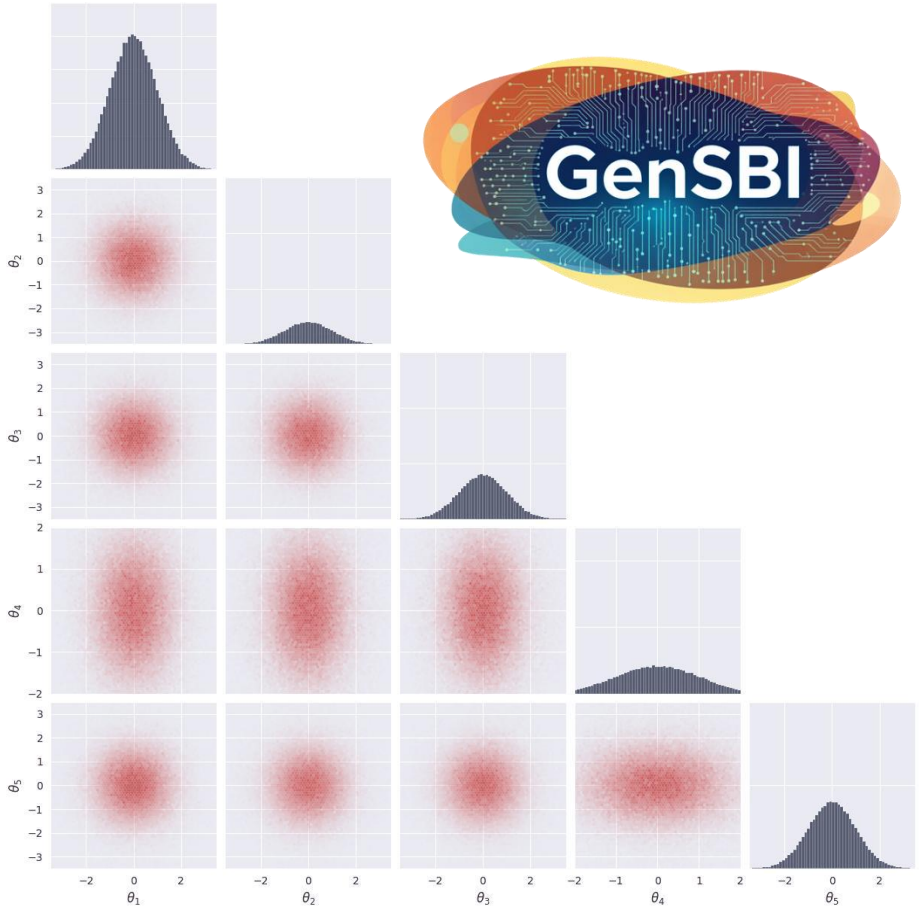
Transformer density estimation

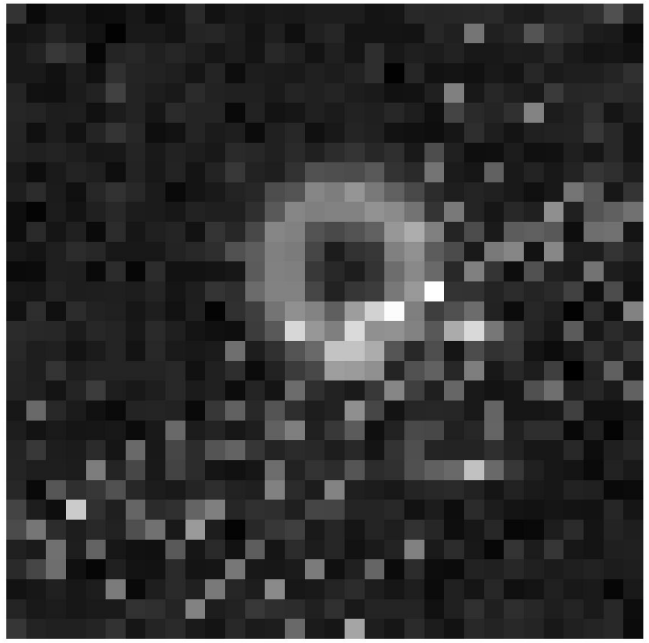
flow & score matching

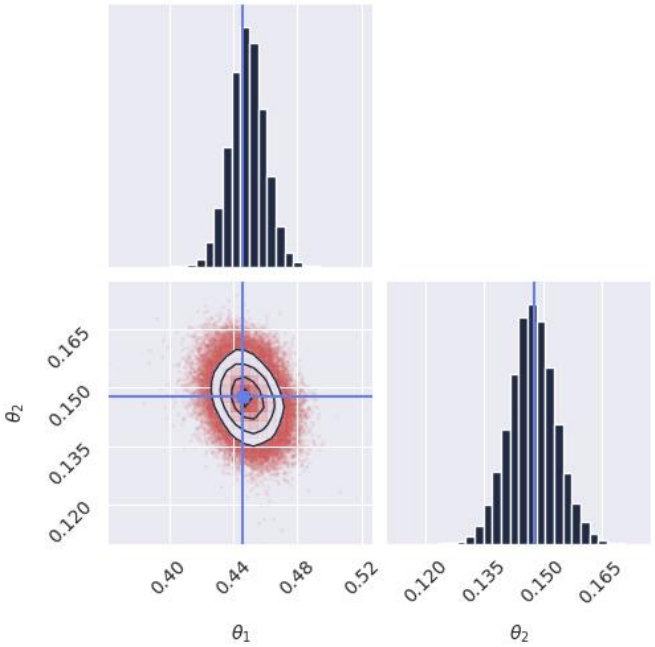
Flux1 · Simformer · Flux1Joint

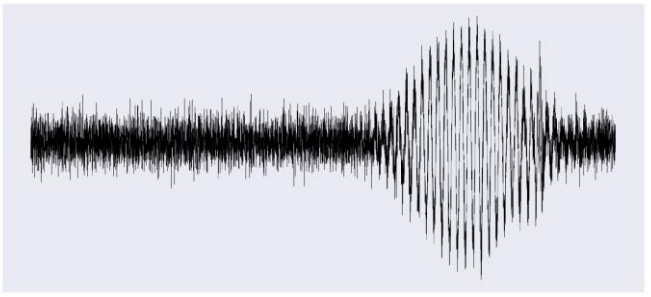
Specialized embeddings: keep/learn the feature correlation structure

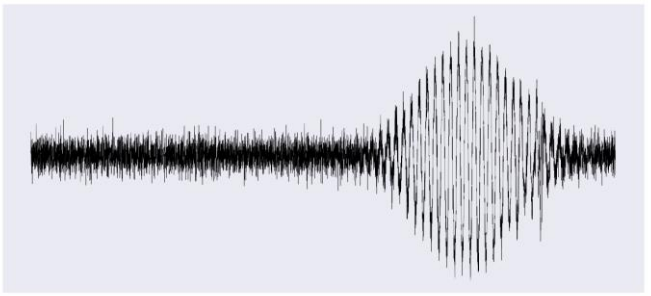
* Ask me about how we make ViT efficient

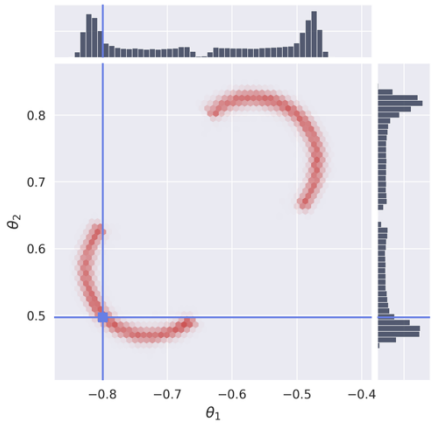


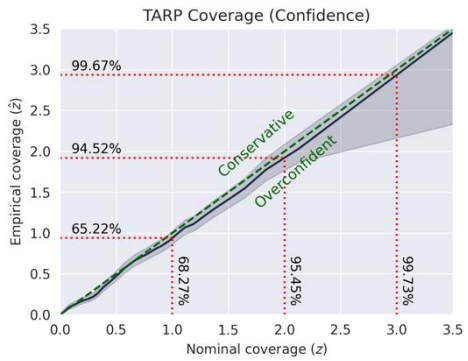


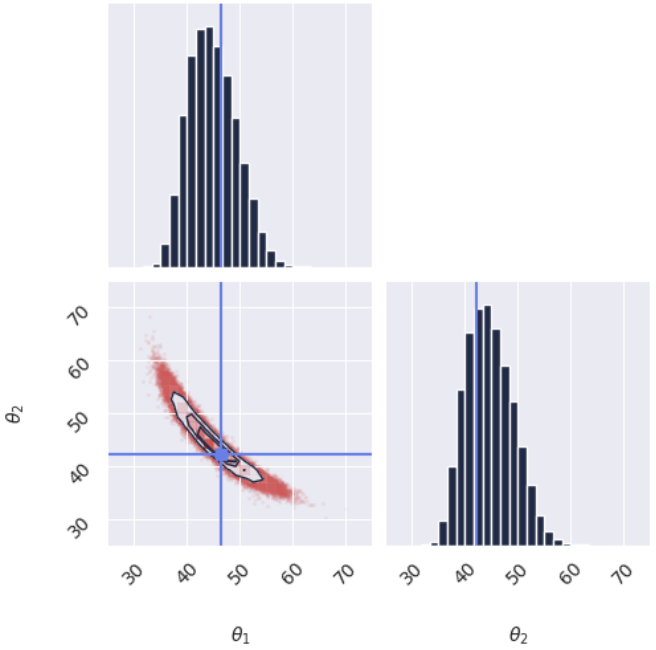








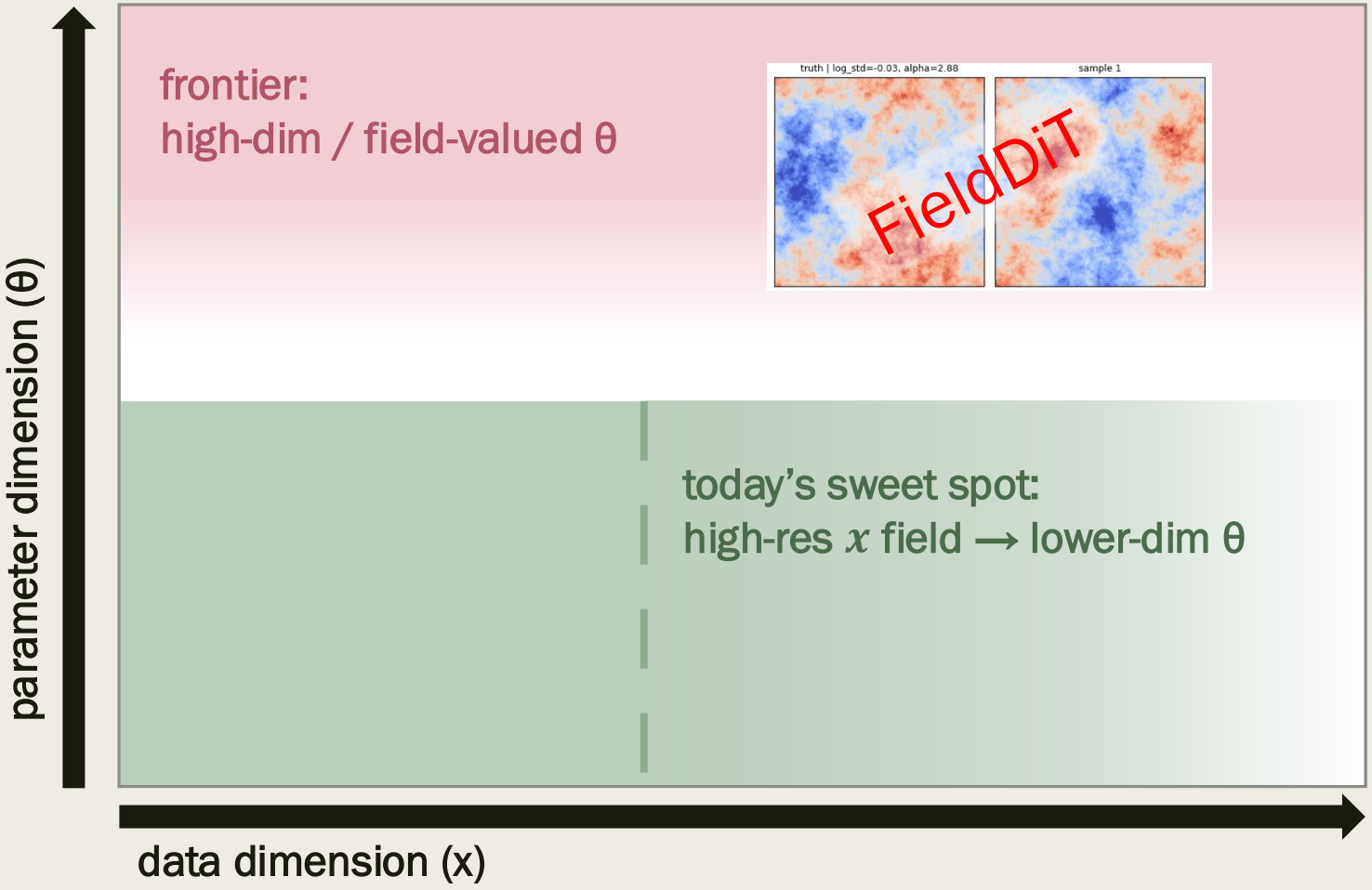




* Ask me about performance benchmarks

AA, arXiv:2605.27499

Where does your problem live?



Where GenSBI lives (today)

- Scales to high-dimensional data (x)
- Tested in lower parameter dimension (θ)
- FM/SM with transformers scales to 1000s of parameters, stay tuned!

**Flux1, 3072 tokens, image generation mode, see https://github.com/black-forest-labs/flux*

The road ahead

NF backbones for NLE
single-pass exact likelihood ⟶ MCMC ⟶ NPE

Field-level / high-dim θ
hard: per-component tokens ⟶ $O(n^2)$ attention

Multimodal conditioning
condition on >1 observable at once
(e.g. summary stats and a field map)

Coming next:

Scaling & deployment
distributed training / inference; HuggingFace serving (safetensors)

New generative methods
sequential NPE, MeanFlow, Consistency models

More architectures
additional models (PixelDiT) + MLP alternatives for simpler problems

Science-ready I/O
HEALPix-map inference; more physics-related examples

SBI benchmarks repo
take a sneak peek at tinyurl.com/SBI-benchmarks

What GenSBI can do for you

- **Modern Generative methods**: OT Flow Matching (Meta) and Diffusion Models (Nvidia)
- **High-Performance JAX Stack:** based on Flax NNX (CPU, GPU & TPU)
- **SOTA models**: Flux1, Simformer + embedders
- **High-Level "recipes":** plug-and-play for easy experimentation
- **Modular & Extensible Design:** easy to use low-level API
- **Documented & Tested**: looking for feedback!

gensbi.com
arXiv:2605.27499
`pip install gensbi`

CSIC Consejo Superior de Investigaciones Científicas | Universitat de València | IFIC Institut de Física Corpuscular | Excelencia Severo Ochoa

Thank you!

aure.amerio@gmail.com

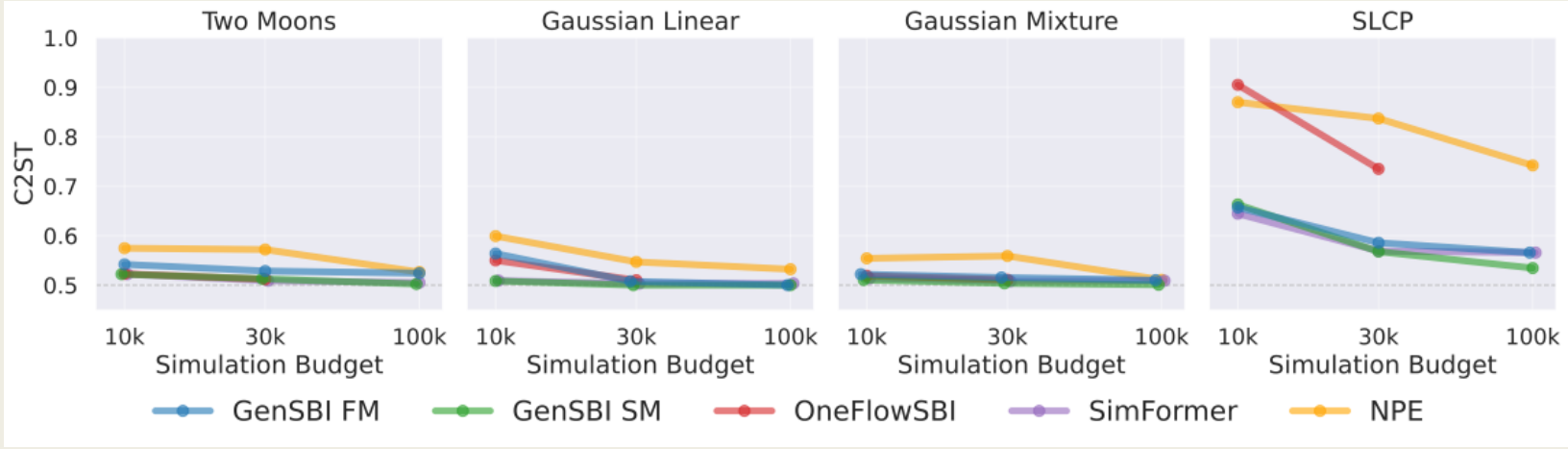
BACKUP

Core advantages

- **Likelihood free**
- **Stable training** (unlike GANs)
- **Flexible model architecture** (unlike Normalizing Flows)
- **Strong theoretical foundation** (ODEs and SDEs guarantee convergence)
- **Iterative sampling procedure** (stable sampling, easier training)
- Empirically, **less prone to overfitting, better sample quality**
- **State of the Art** for generative tasks, NPE, and Emulation**

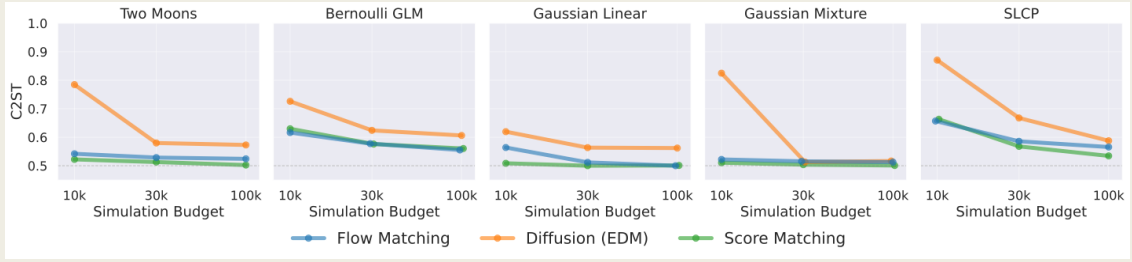
Benchmarks: competitive / SOTA & calibrated

C2ST near 0.5 across SBIBM tasks · matches / surpasses SimFormer, OneFlowSBI, NPE at 10^5 sims (esp. SLCP) · near-uniform config across tasks.

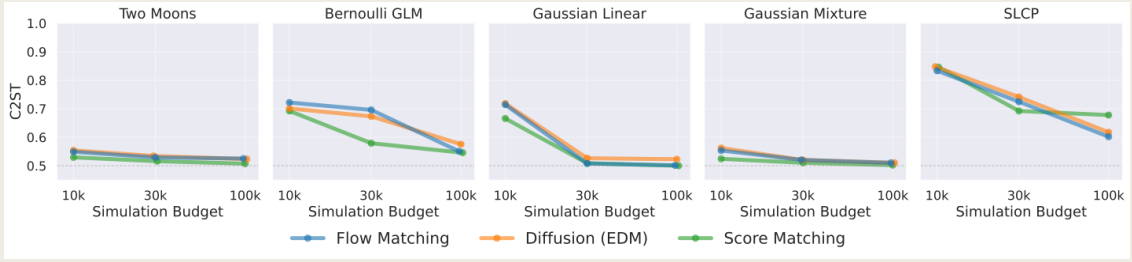


C2ST comparison vs. SimFormer, OneFlowSBI & NPE across SBIBM tasks

Flux1Joint — FM vs. Diffusion vs. Score Matching

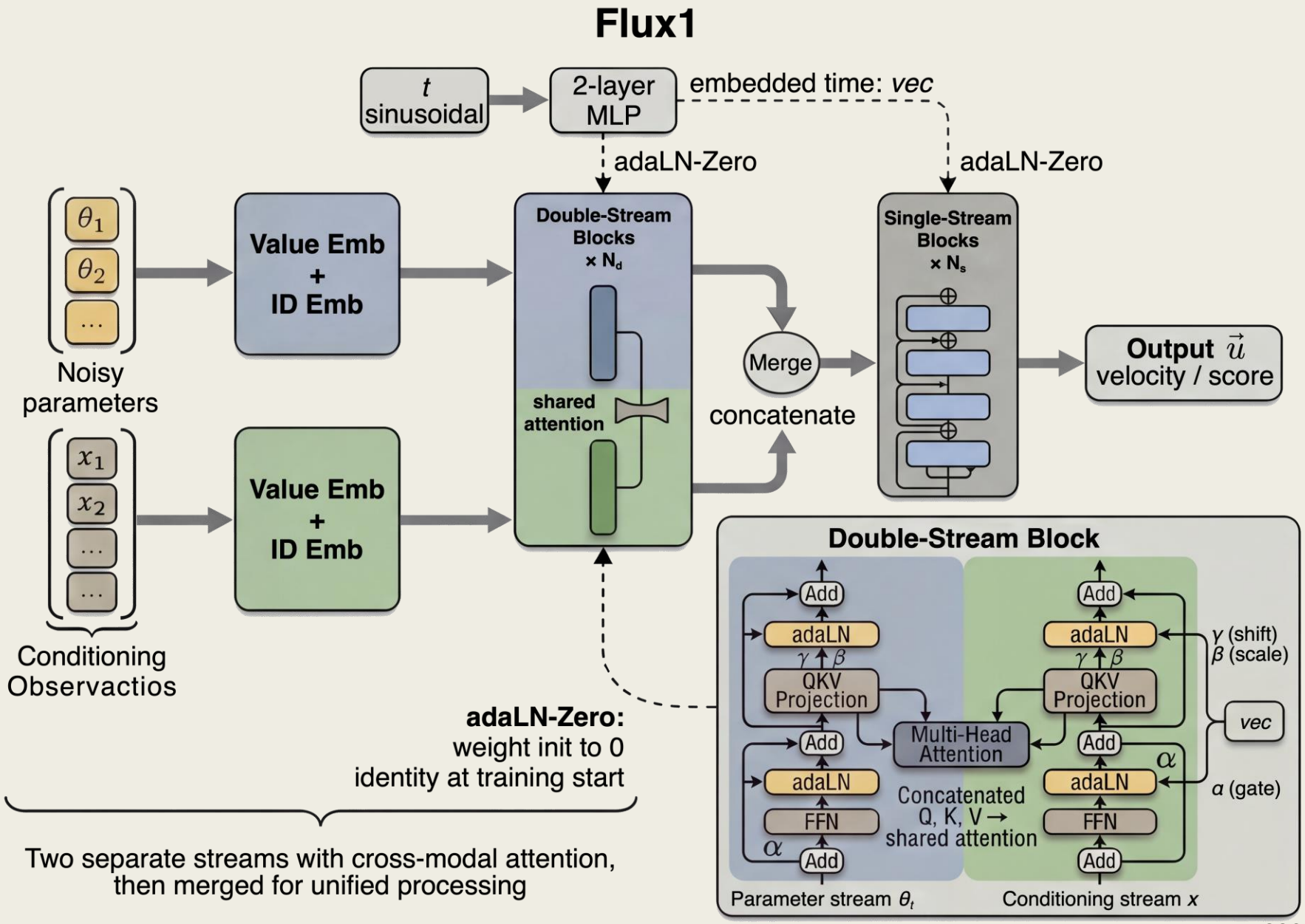


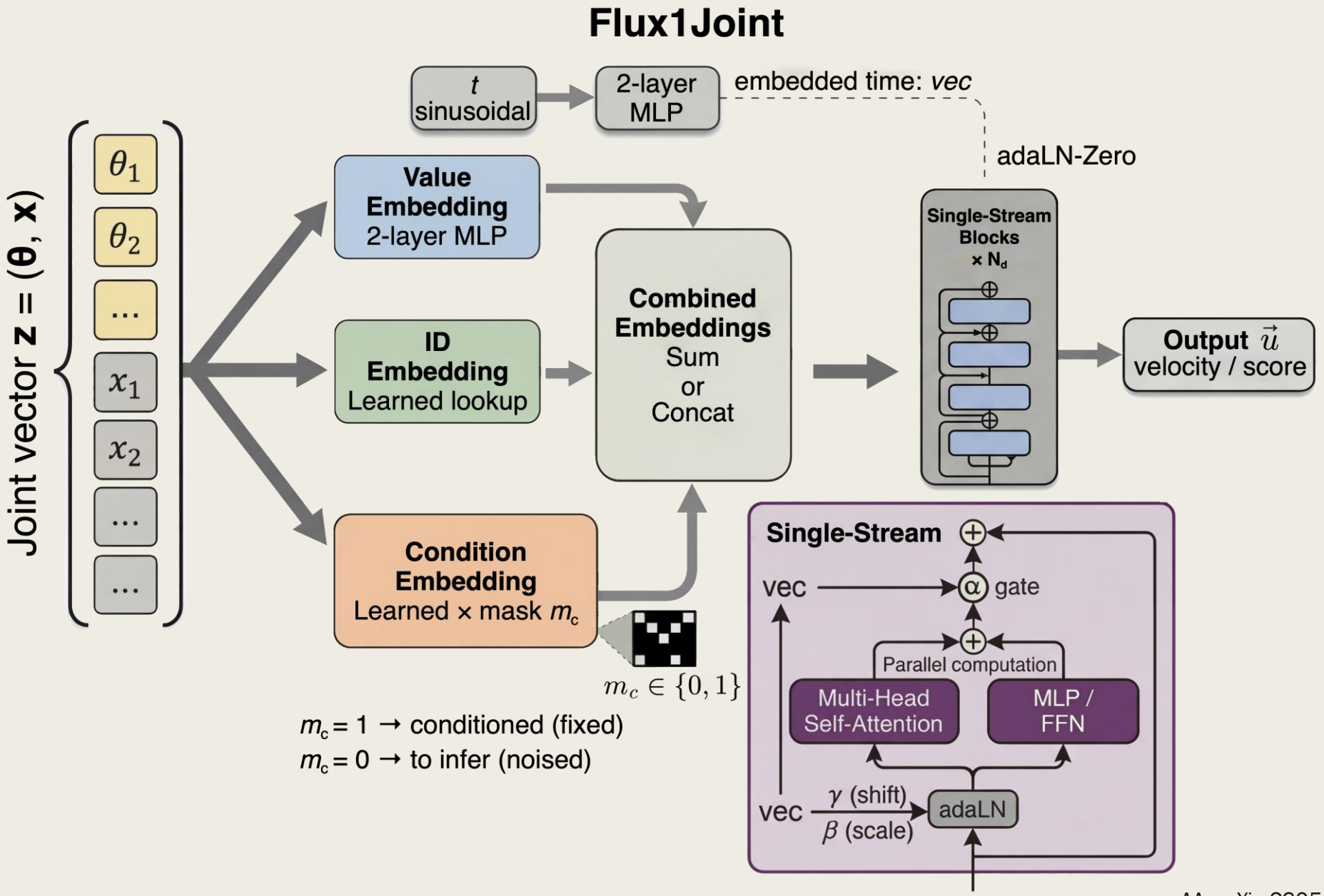
Flux1 — FM vs. Diffusion vs. Score Matching



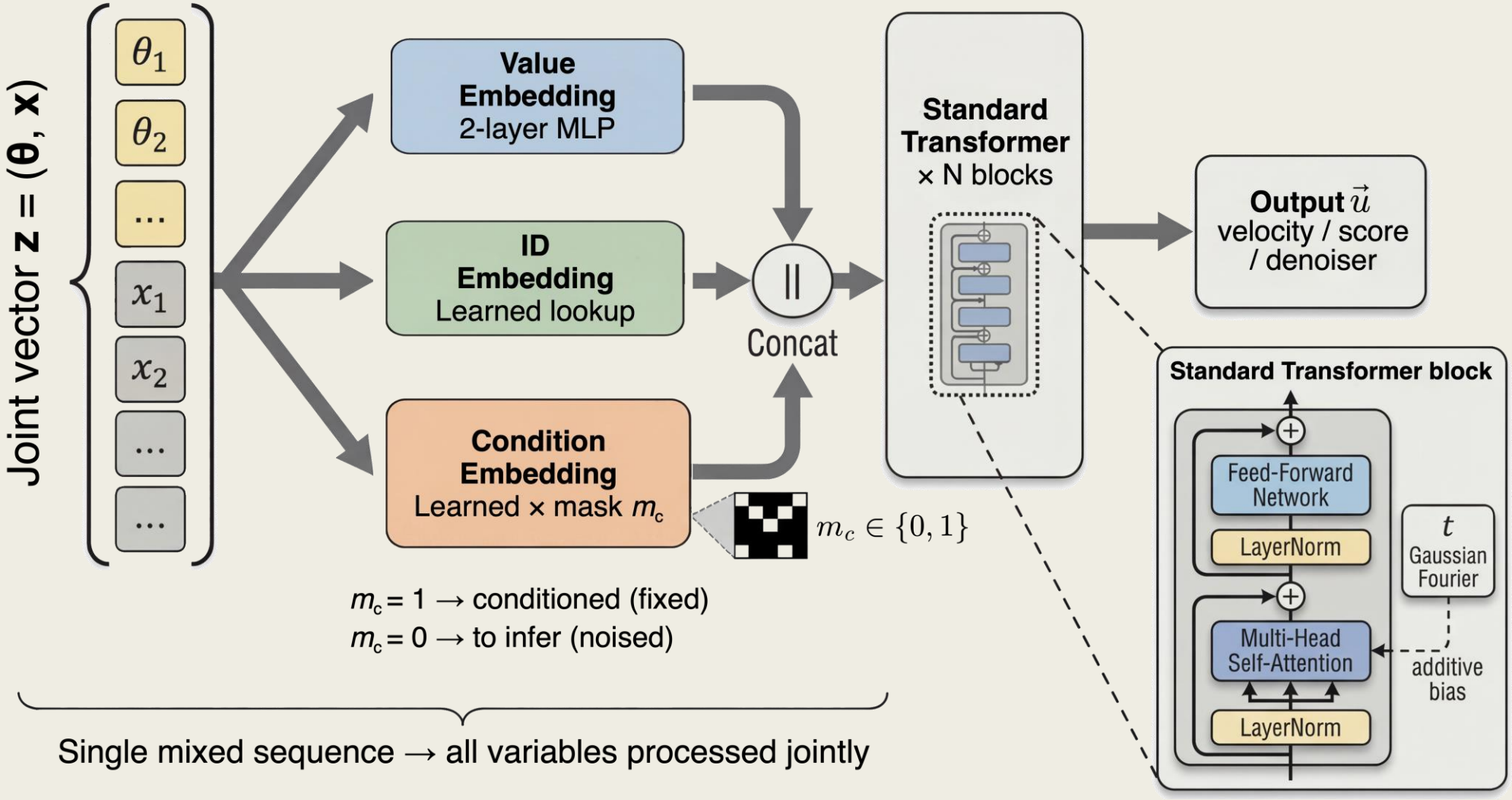
Which estimator when? NF vs flow / diffusion

	Normalizing flows	Flow matching / diffusion
Exact likelihood	Single pass ✓	Needs ODE solve ✗
Sampling cost	Single pass	Iterative (ODE / SDE)
Architecture	Invertible only	Any (transformers)
Best scope	**NLE + MCMC**	**NPE / joint**


Flux1
t sinusoidal
2-layer MLP
embedded time: vec
adaLN-Zero
adaLN-Zero
θ_1
θ_2
...
Noisy parameters
Value Emb + ID Emb
Double-Stream Blocks × N_d
shared attention
Merge
concatenate
Single-Stream Blocks × N_s
Output $\vec{u}$ velocity / score
x_1
x_2
...
...
Conditioning Observactios
Value Emb + ID Emb
adaLN-Zero: weight init to 0 identity at training start
Two separate streams with cross-modal attention, then merged for unified processing
Double-Stream Block
Add
adaLN
γ β
QKV Projection
Multi-Head Attention
Concatenated Q, K, V → shared attention
FFN
α
γ (shift)
β (scale)
vec
α (gate)
Parameter stream θ_t
Conditioning stream x

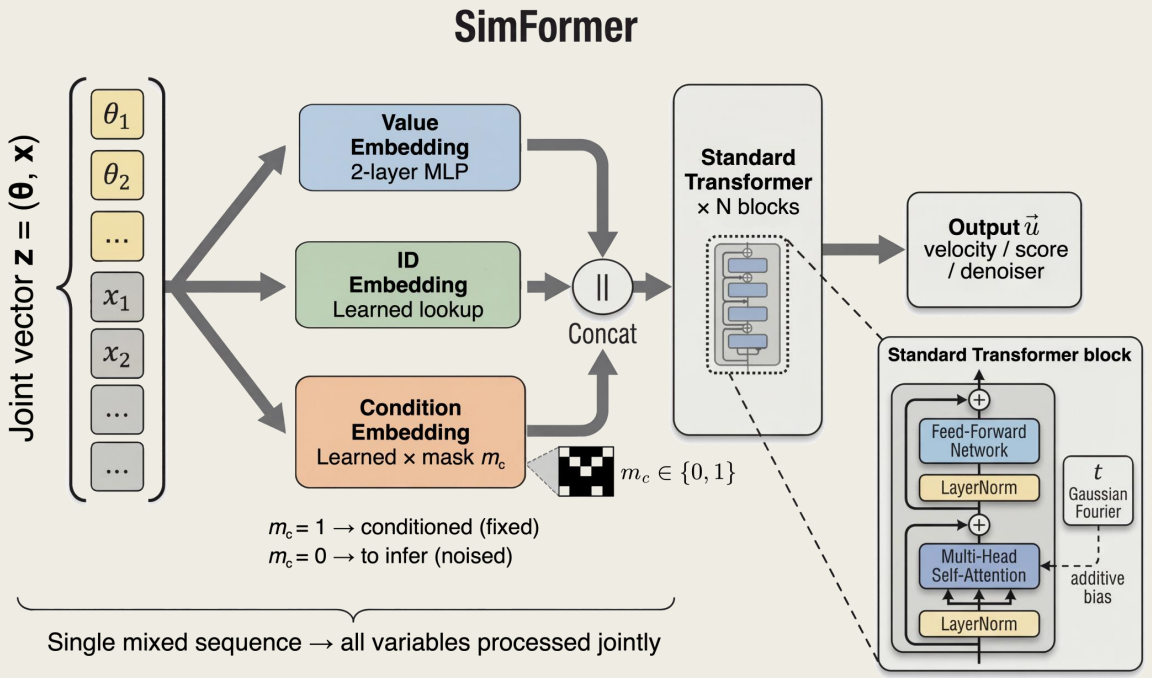

Flux1Joint
t sinusoidal
2-layer MLP
embedded time: vec
adaLN-Zero
Joint vector z = (θ, x)
θ1
θ2
...
x1
x2
...
...
Value Embedding
2-layer MLP
ID Embedding
Learned lookup
Condition Embedding
Learned × mask mc
Combined Embeddings
Sum
or
Concat
Single-Stream Blocks
× Nd
Output u⃗
velocity / score
mc ∈ {0, 1}
mc = 1 → conditioned (fixed)
mc = 0 → to infer (noised)
Single-Stream
vec
α gate
Parallel computation
Multi-Head Self-Attention
MLP / FFN
vec
γ (shift)
β (scale)
adaLN

SimFormer


Joint vector z = (θ, x)
θ_1
θ_2
...
x_1
x_2
...
...
Value Embedding
2-layer MLP
ID Embedding
Learned lookup
Condition Embedding
Learned × mask m_c
||
Concat
$m_c \in \{0, 1\}$
m_c = 1 → conditioned (fixed)
m_c = 0 → to infer (noised)
Single mixed sequence → all variables processed jointly
Standard Transformer
× N blocks
Output $\vec{u}$
velocity / score
/ denoiser
Standard Transformer block
Feed-Forward Network
LayerNorm
Multi-Head Self-Attention
LayerNorm
t
Gaussian Fourier
additive bias

Joint estimation from one model

- Train the joint p(θ, x) with a random condition mask
- At inference, fix the mask to recover any conditional from a single trained model:
  - *Posterior p(θ | x)*
  - *Likelihood samples p(x | θ)*
  - *Arbitrary marginals*



Solvers & SDE ↔ ODE swap

- Solvers: Euler / Heun / Dopri5 + SDE variants
- Post-training swap between deterministic ODE and stochastic SDE — no retraining
- ZeroEnds / NonSingular SDE solvers trade speed vs mass coverage
- Typical step counts: EDM ~18 Heun · flow matching ~100 ODE · score matching ~1000

Compute cost (it's modest)

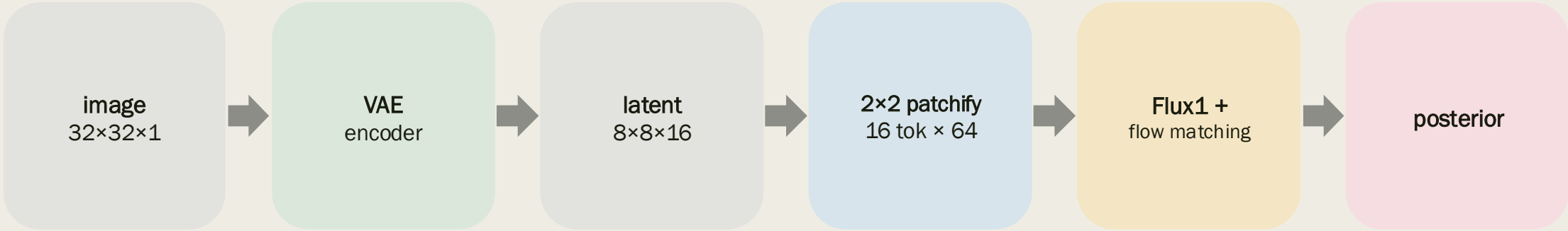
Reference: Two Moons on a V100 GPU.

Metric	Cost
Training (50k steps)	~1–3 hours
Draw 10k posterior samples	≤ ~42 s (worst case)
Minimum hardware	Consumer RTX 4070 (12 GB)

Why field-level (high-dim θ) is hard

- Per-component tokens ⟶ sequence length ∝ field size ⟶ $O(n^2)$ attention becomes computationally expensive very
- Posterior-over-a-field is huge: sampling and calibration both harder
- Directions: patchify θ (ViT-style) · local / structured attention · latent diffusion
- Latent-diffusion caveat: only as good as your autoencoder — a poor VAE ⟶ over-smoothed posteriors that erase the small-scale non-Gaussian information ⟶ need field (pixel) level transformers ⟶ lot's of cool new stuff in 2026 like PixelDiT
- GenSBI already has an experimental VAE training pipeline, FieldDiT and PixelDiT, but more experimentation is needed

The pipeline — and the embedding payoff



- ID embedding (**“absolute”, “rope2d”**) keeps each patch’s spatial position in the transformer
- The VAE encoder (GenSBI’s **AutoEncoder2D**) is trained **end-to-end** with the inference model
- GenSBI ships the encoder **and** the patchify + RoPE wiring — the part other libraries leave to you

Multimodal conditioning (future)

- Condition on more than one observable at once (e.g. summary stats and a field map)
- Per-stream embedder + ID embedding, fused via attention (MM-DiT style)
- Precedent: Flux Kontext conditions on image and text simultaneously