TOBIAS BUCK

27.9.2024 TOBIAS.BUCK@IWR.UNI-HEIDELBERG.DE

MACHINE LEARNING IN ASTROPHYSICS

WHO HAS AN ASTROPHYSICAL BACKGROUND?

WHO HAS A STRONG NUMERICAL/ CODINGE BACKGROUND?

WHO HAS EXPERIENCE WITH ML?

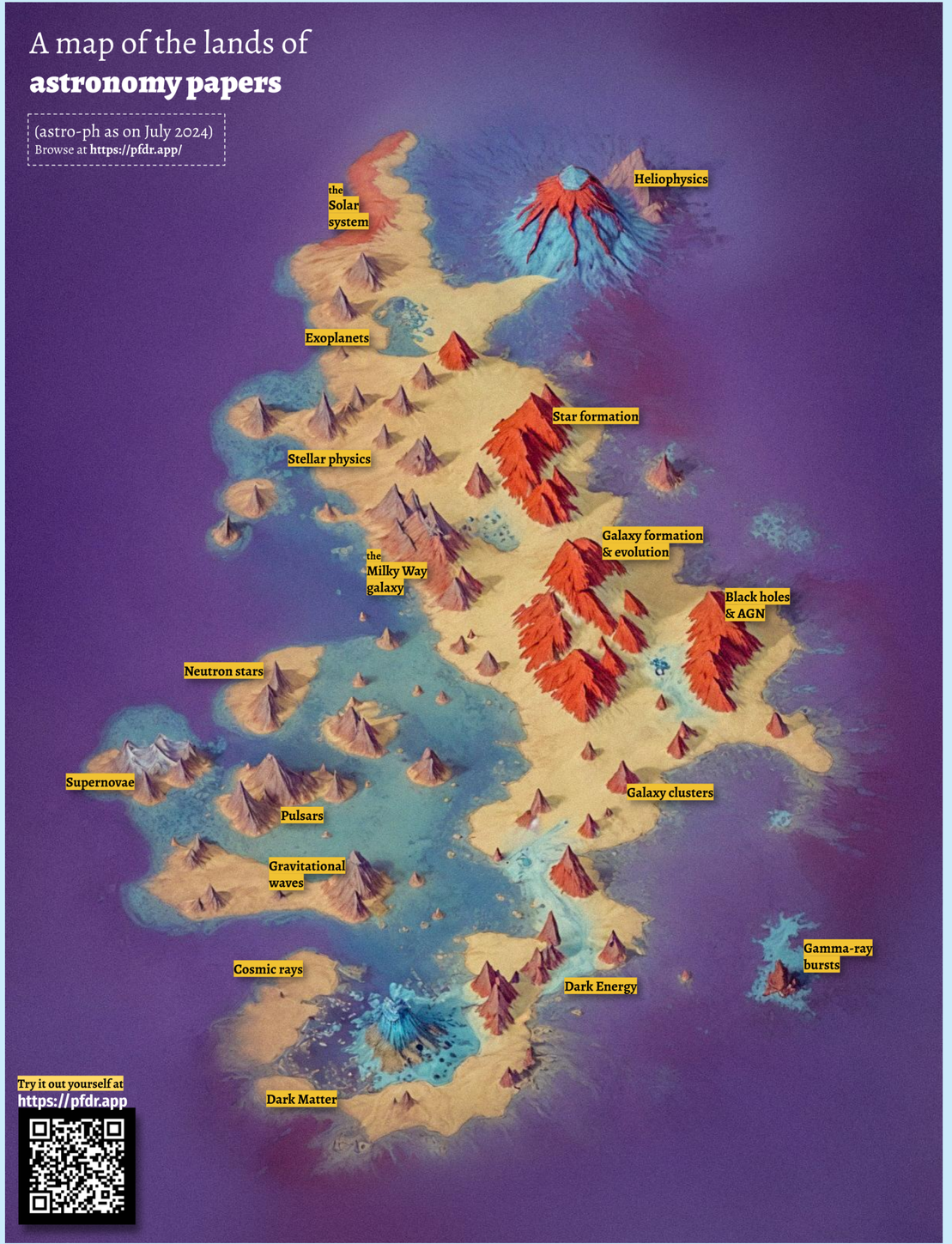
AI ASSISTED PROGRAMMING

GitHub
Copilot

WHO HAS BEEN USING LARGE LANGUAGE MODELS FOR (SCIENTIFIC) WRITING?

LARGE LANGUAGE MODELS IN ASTROPHYSICS

A public-friendly visualization of the 2d manifold of galaxy evolution papers created with UMAP+stable diffusion that shows the different areas of the astro-ph literature corpus. Following similar patterns as the heatmap, mountains indicate well-studied areas, plains indicate fields of active study, coastal regions are ‘hot topics’, and water denotes regions with no papers. Similar to a world map, the axes here do not hold a particular meaning. Regions close to each other have semantic similarity, while distant regions do not.



Iyer+2024, Pathfinder (UniverseTBD)

OUTLINE OF THIS LECTURE SERIES

1. Day 1: Overview over ML in Astro
   A. Broad overview of different techniques
   B. Intro to automatic differentiation
2. Day 2: Generative AI
   C. Normalizing Flows
   D. Variational Autoencoder
   E. Model Comparison & Model Misspecification
3. Day 3: Generative AI cont.
   F. Score matching / diffusion models
   G. Simulation-Based Inference
4. Day 3: Scientific ML I
   H. Neural ODEs

OUTLINE OF THIS LECTURE SERIES

1. Day 4: Scientific ML cont.
   A. Operator Learning, PINNs, Hamiltonian Neural Networks
2. Differentiable physics simulators
3. Day 5: Differentiable physics simulators
   B. Introduction to Differentiable Physics
   C. Gradient based optimisation of computer programs
   D. Training NNs with „the simulator in the loop“
   E. reducing numerical errors with deep learning
   F. solving inverse problems with NNs

WHERE TO FIND THE LECTURE MATERIAL

https://github.com/TobiBu/graddays

WHERE TO FIND THE LECTURE MATERIAL

In general, there will be ~1.5h lectures (14:00-15:30), a ~30min. coffee break (15:30-16:00) and then ~1h tutorials/exercises (16:00-17:00)

→ bring your notebook/laptop to work on google colab Jupiter notebooks

Today, we will have interactive exercises as I don’t expect everyone to have a notebook.

OUTLINE OF TODAY

1. Unsupervised Learning
   A. Clustering
   B. Representation Learning
2. Inference / Bayesian Modelling
   C. Modeling Complex Prob. Distributions
   D. Simulation-Based Inference
   E. Model Comparison & Model Misspecification
3. Forward Models and Emulators
   F. Neural ODEs and Operator Learning and PINNs

MODERN COSMOLOGY AND ASTROPHYSICS

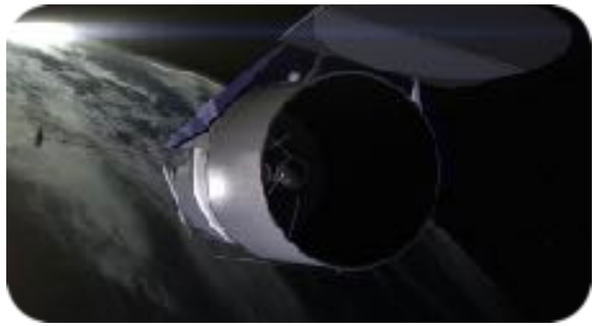
SKA 2.3 billion USD

TMT 2.4 billion USD

JWST 10 billion USD

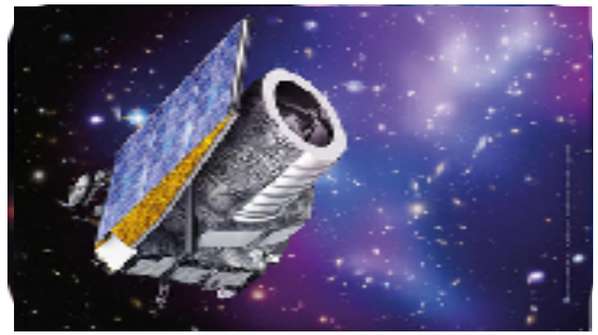
LSST 2 billion USD

ALMA 1.3 billion USD



WFIRST 3.2 billion USD

ATHENA 1 billion USD



Euclid 0.7 billion USD

DATA IS GETTING MORE AND MORE COMPLEX!

THIS MEANS: NEW OPPORTUNITIES BUT ALSO NEW CHALLENGES

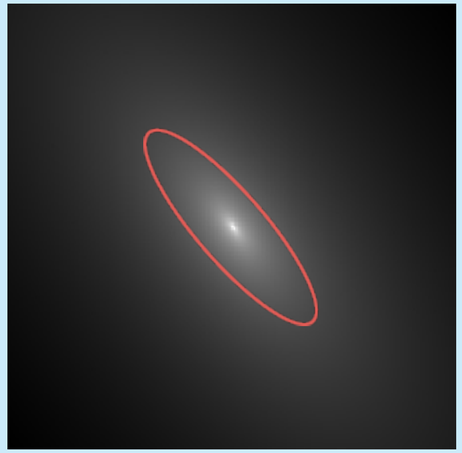
LOTS OF THE THINGS WE WANT TO DO WITH THIS DATA REQUIRE A MODEL (IMPLICIT OR EXPLICIT)

- Simulations
- Inference (aka 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



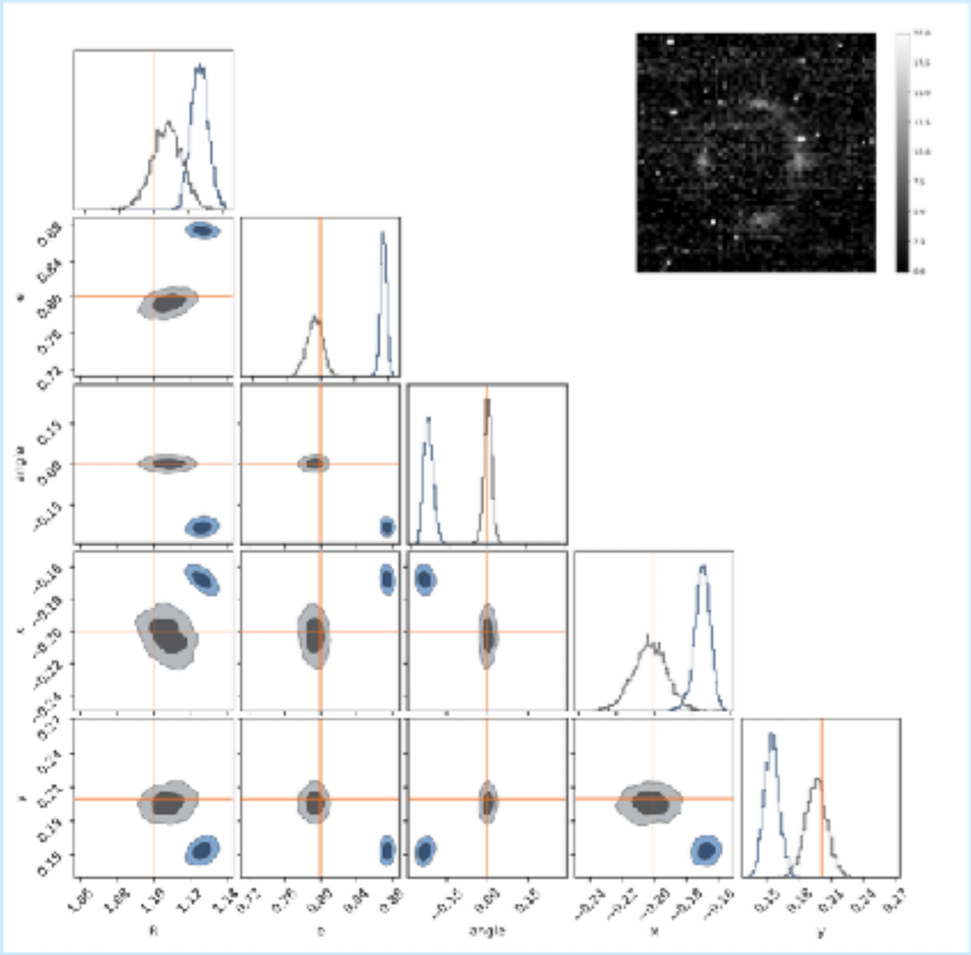
Better data -> with simplistic models we can bias the inference

?

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...




Representation Learning

REPRESENTATION LEARNING

representation learning seeks to automatically discover the representations needed for feature detection or classification from raw data.

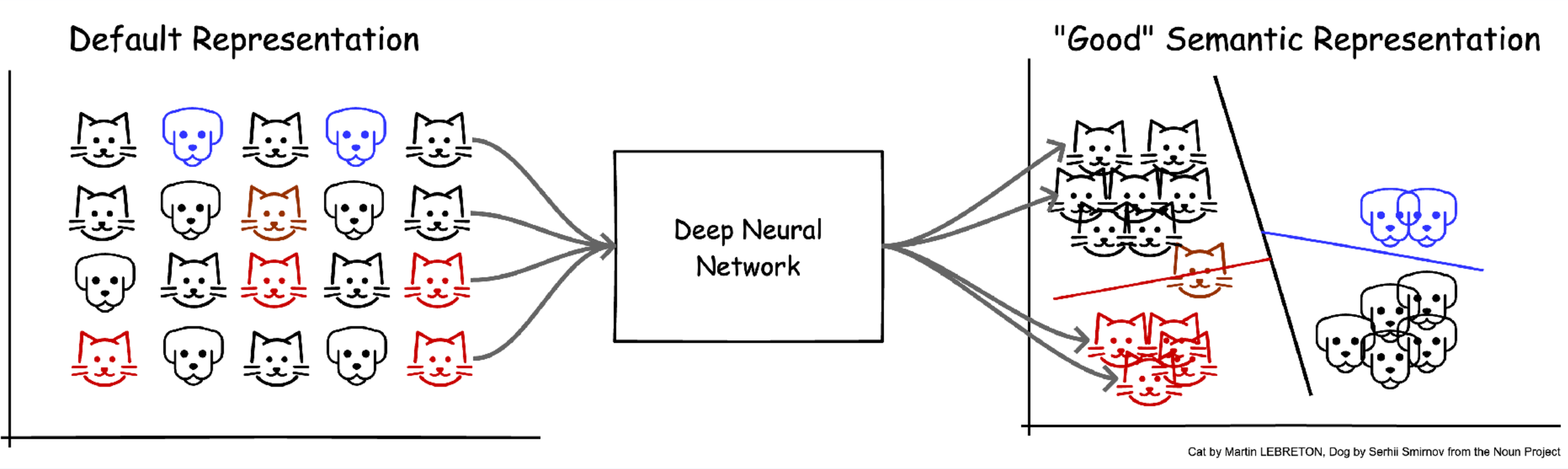
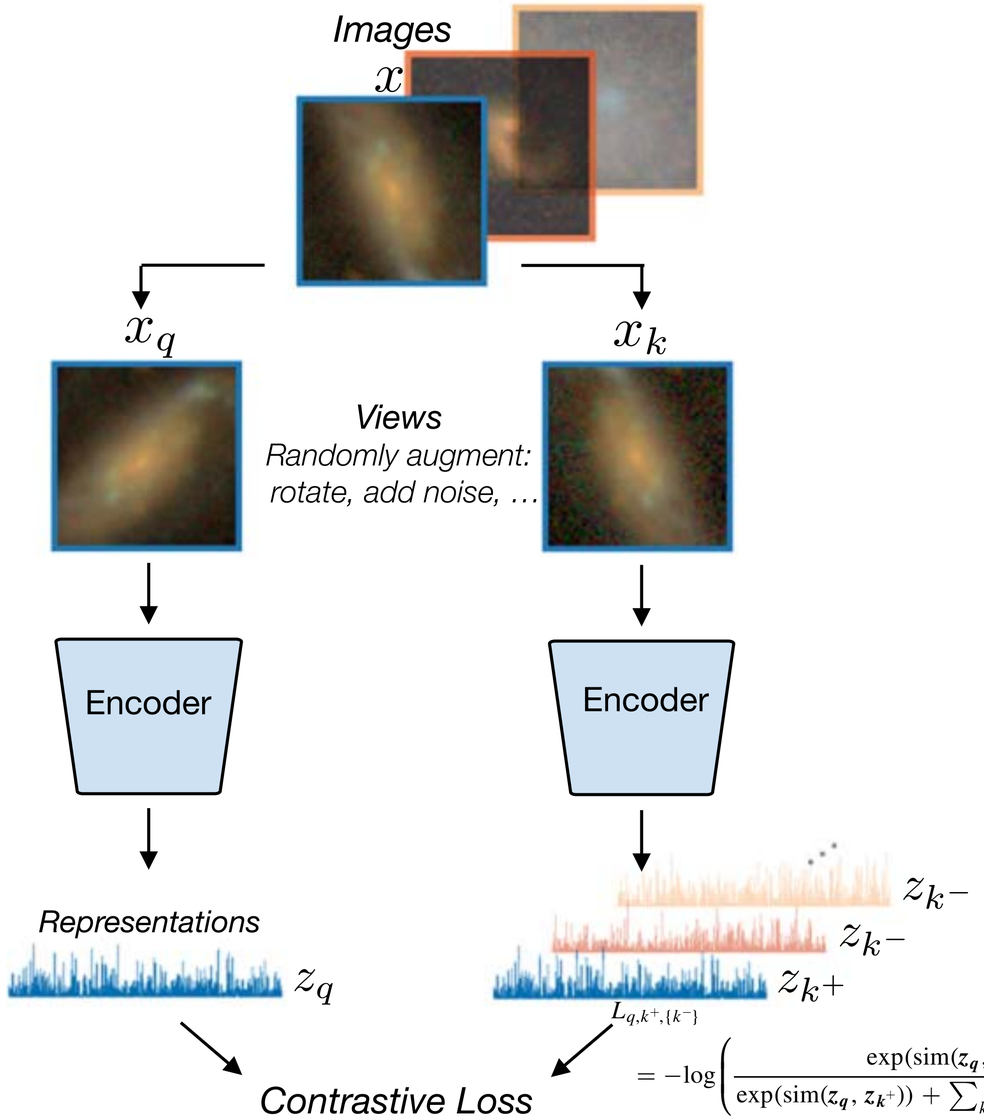


image credit: https://blog.fastforwardlabs.com/2020/11/15/representation-learning-101-for-software-engineers.html

REPRESENTATION LEARNING

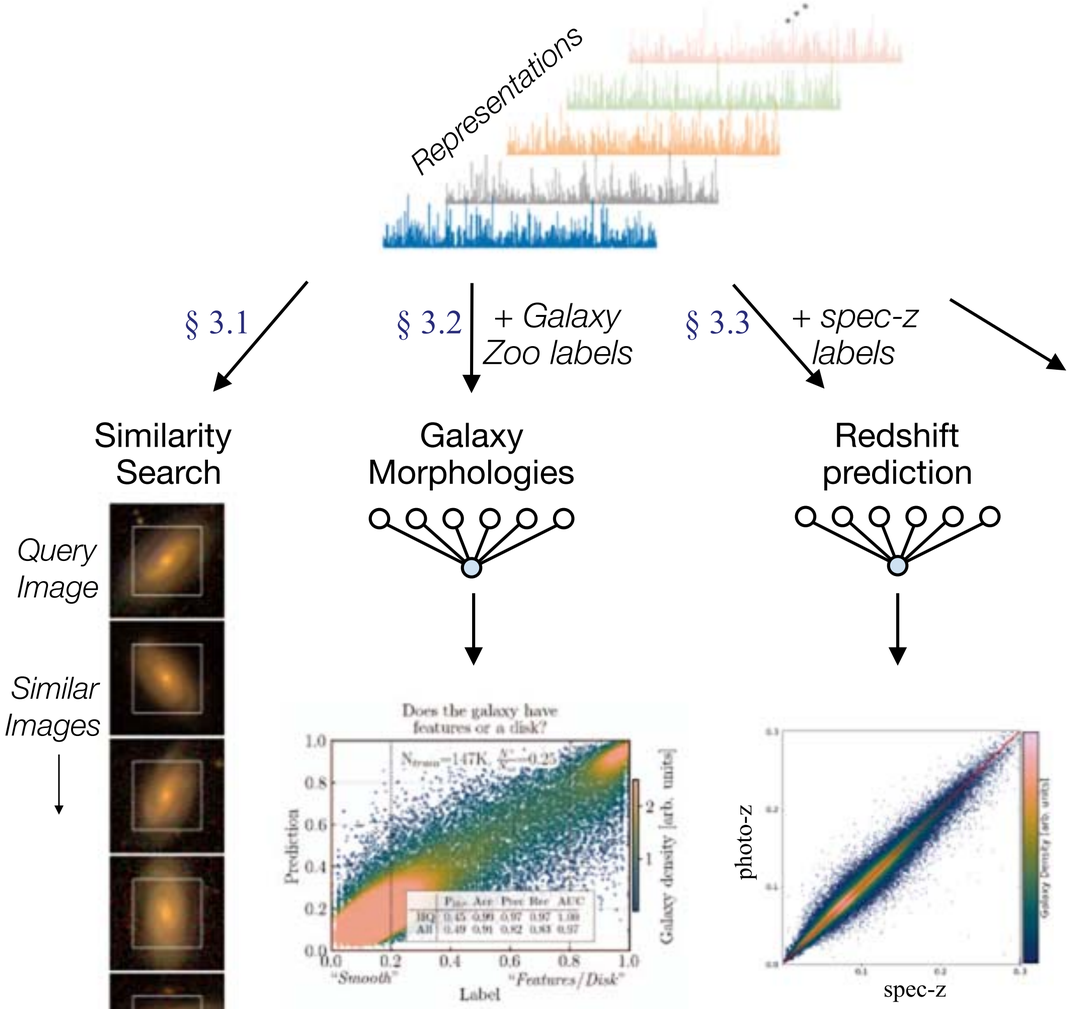
1. Self-supervised contrastive representation learning

Learn representations in an unsupervised manner



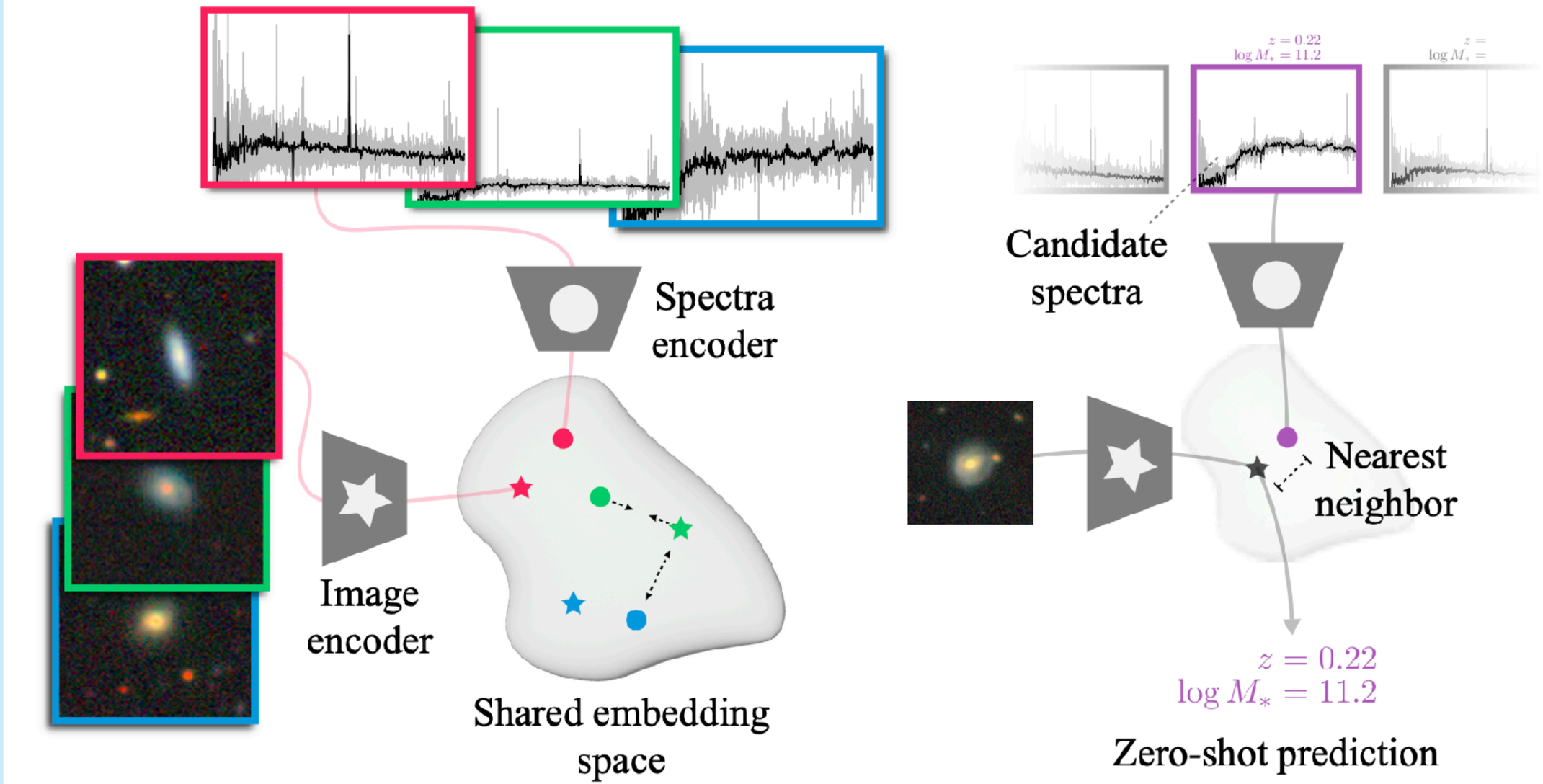
2. Downstream tasks

Use representations for a variety of applications



Hayat+2021

REPRESENTATION LEARNING

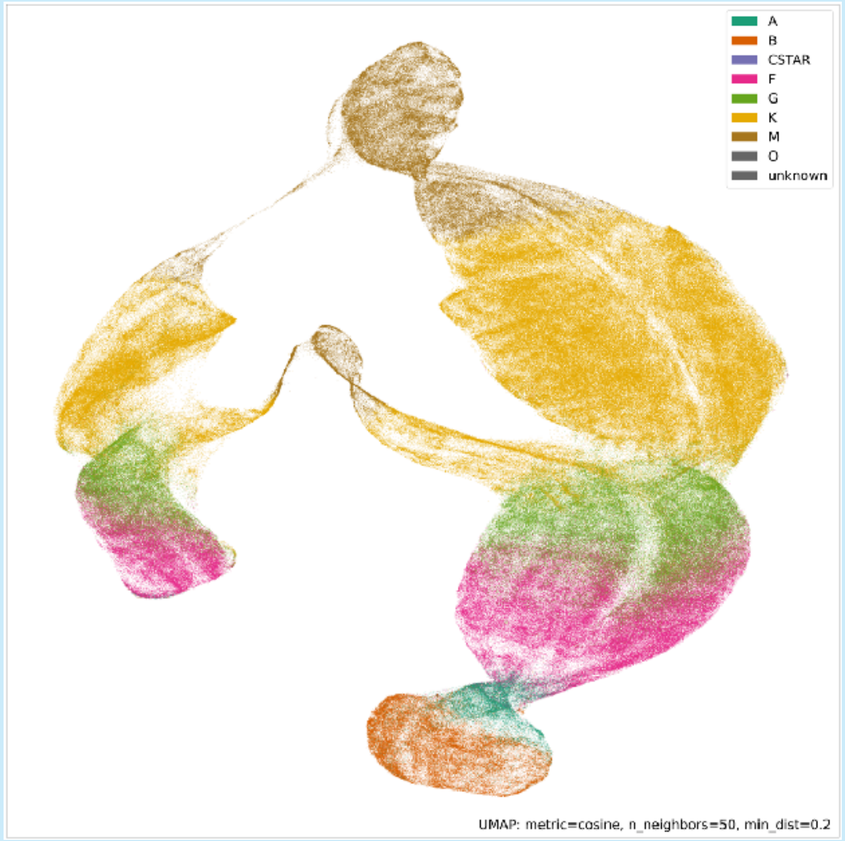


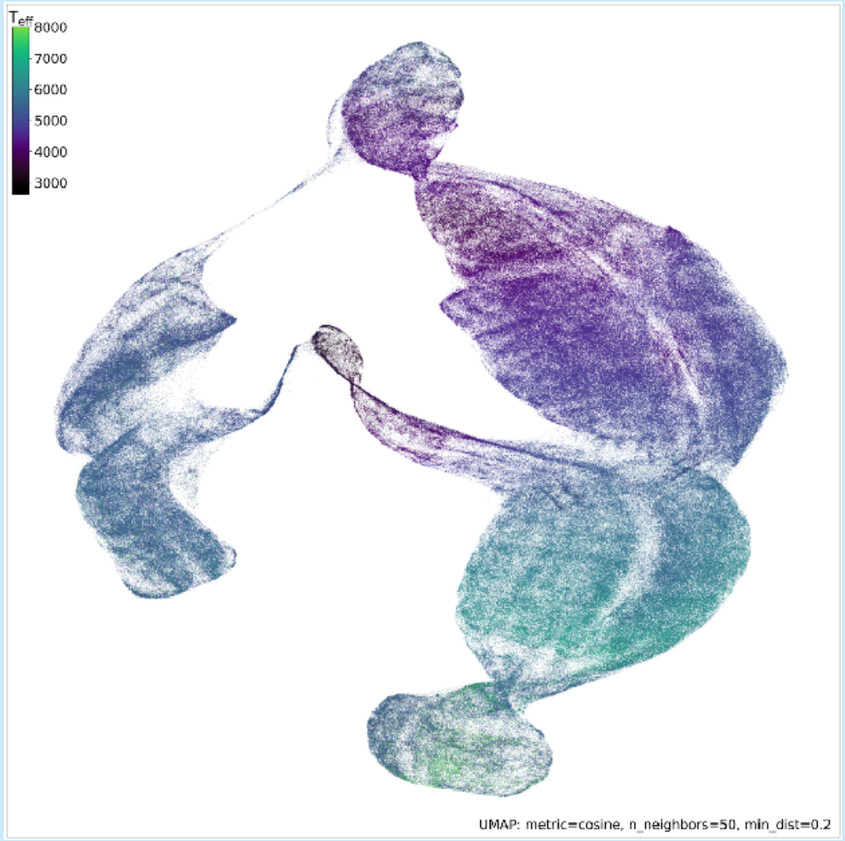
Parker+2024

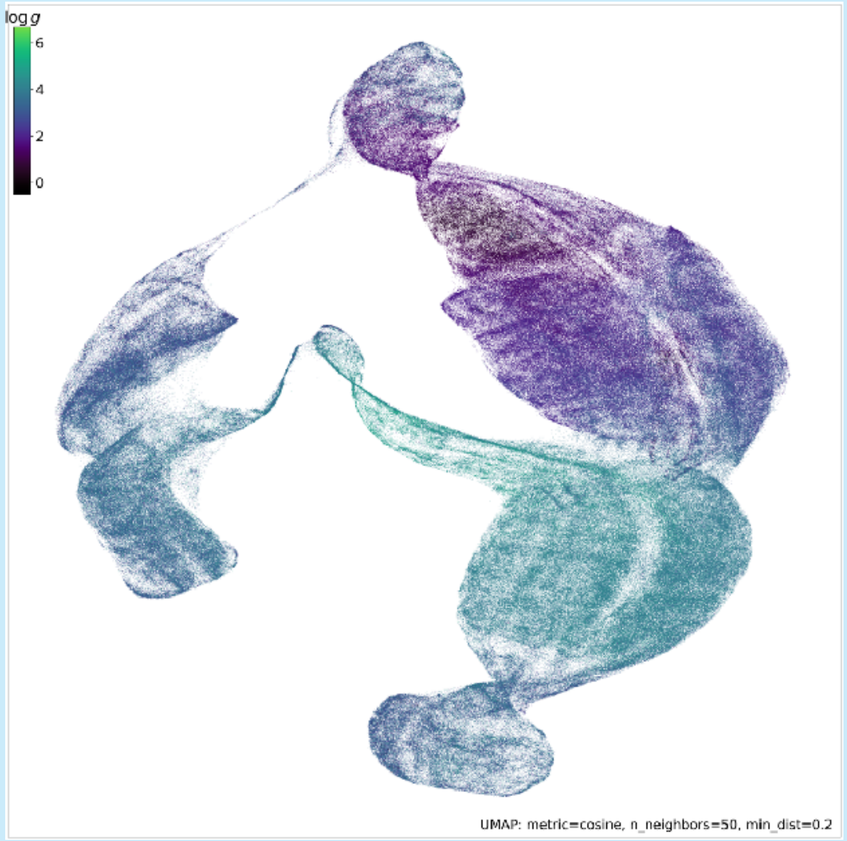
REPRESENTATION LEARNING FOR STELLAR SPECTRA

multi-modal data: Gaia RVS spectra and Gaia XP coefficients plus contrastive loss

$$\mathcal{L}_{CL} = -\frac{1}{N}\sum_{i}^{N} log\frac{exp(x_i^T y_i/\tau)}{\sum_{j=1}^{N} exp(x_i^T y_j/\tau)} - \frac{1}{N}\sum_{i}^{N} log\frac{exp(y_i^T x_i/\tau)}{\sum_{j=1}^{N} exp(y_i^T x_j/\tau)}$$



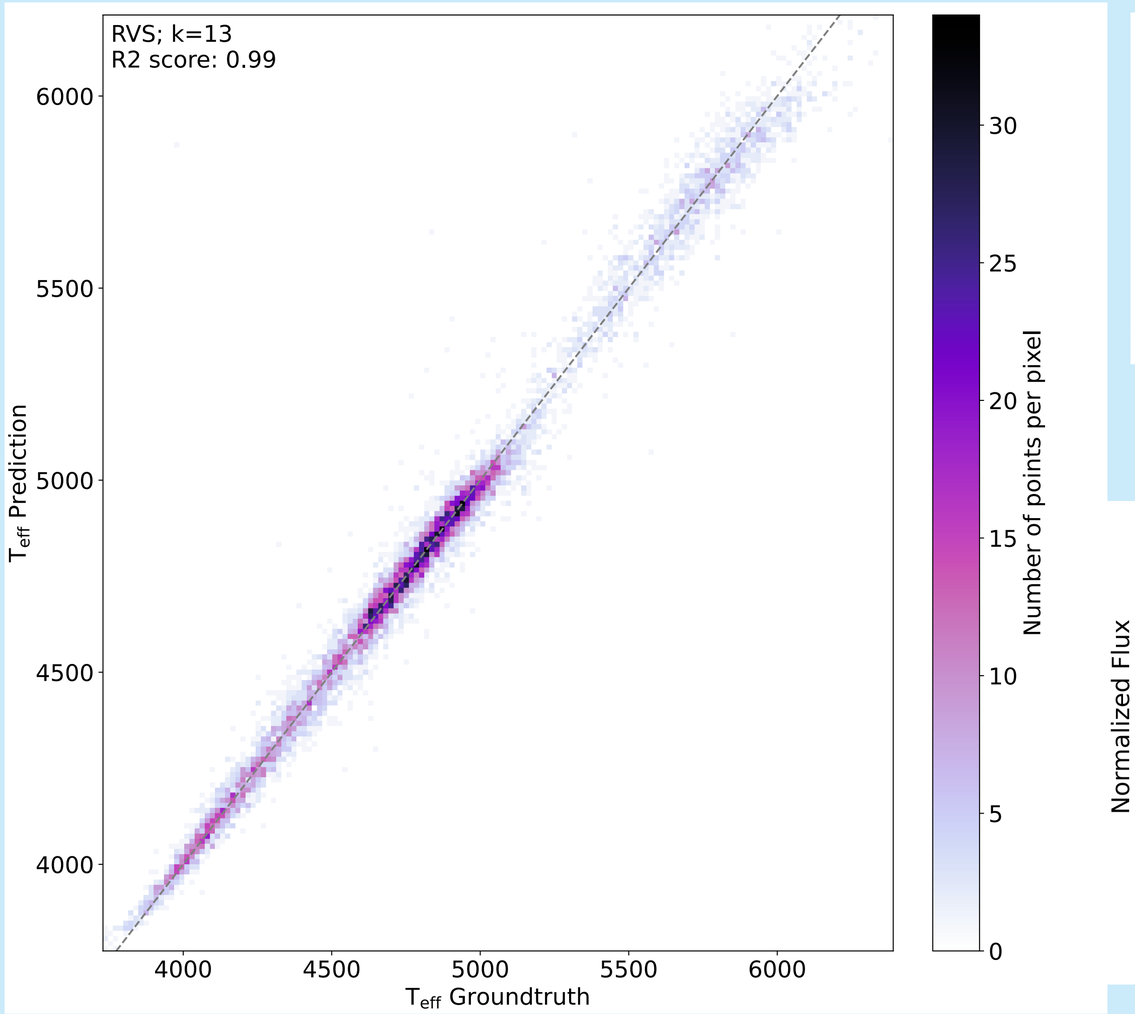




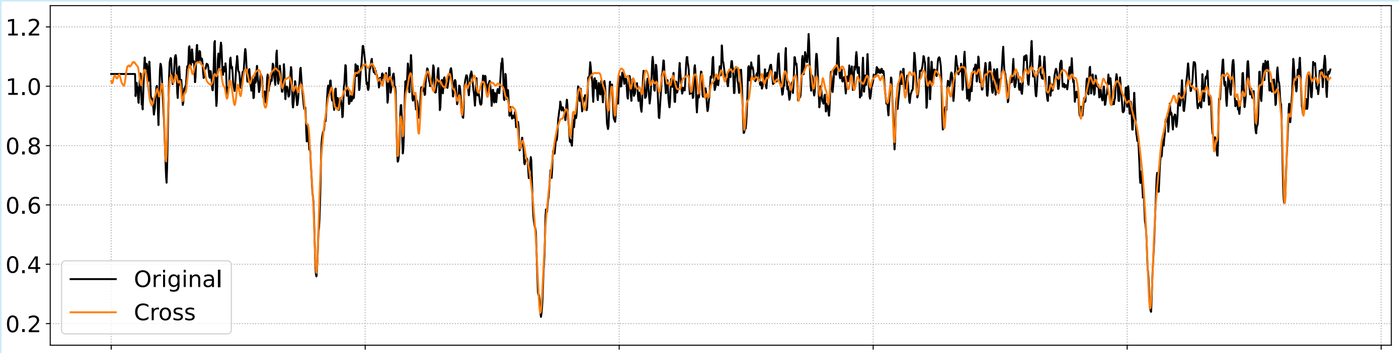
Buck+2024 in prep., see also Parker+2024 for AstroCLIP

REPRESENTATION LEARNING FOR STELLAR SPECTRA

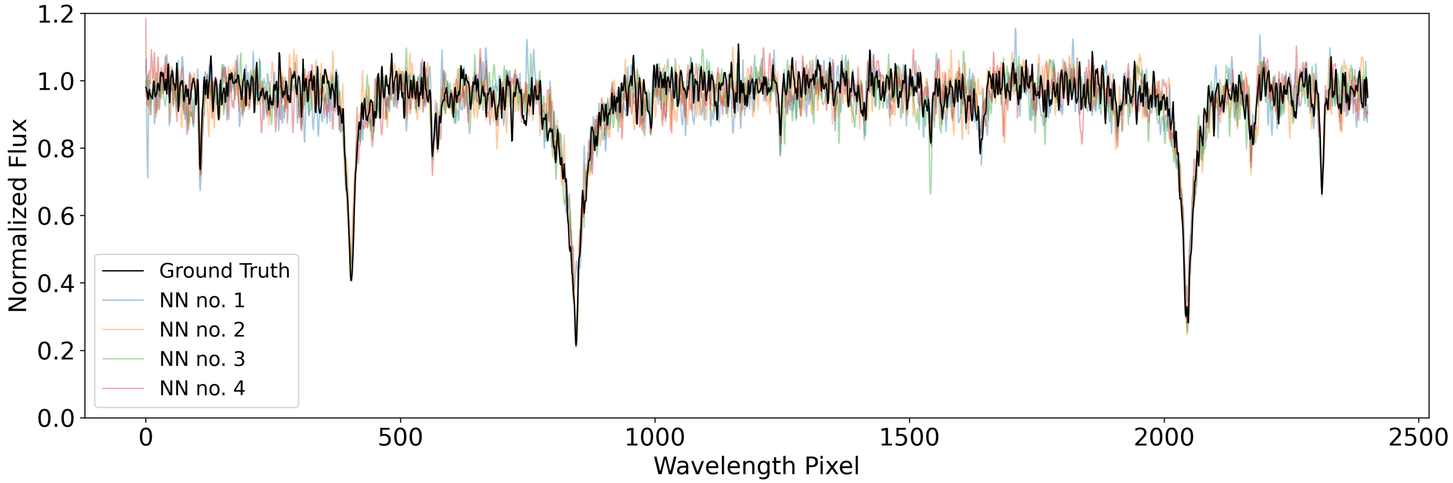
regression



cross-modal generation



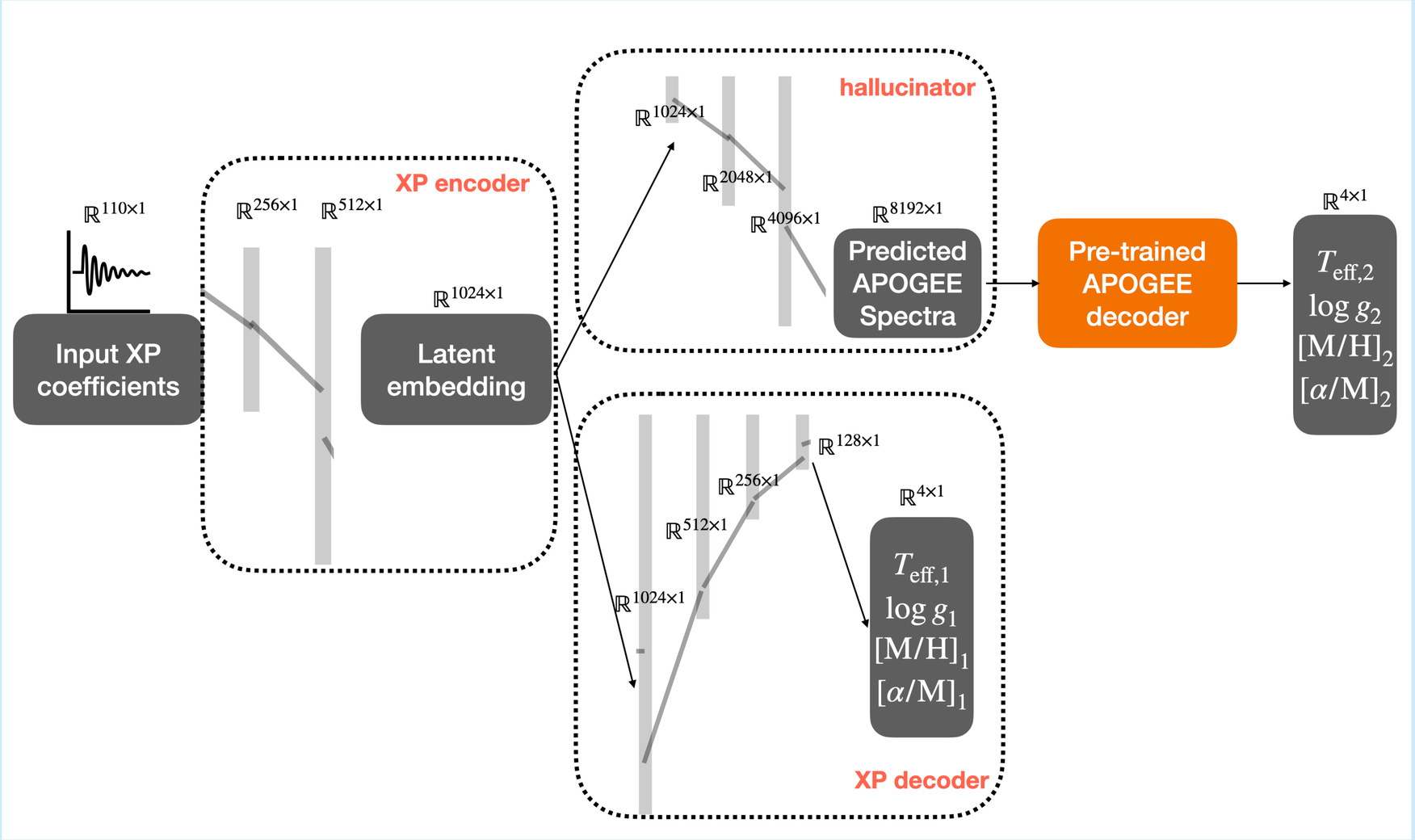
cross-modal look-up



Buck+2024 in prep.

REPRESENTATION LEARNING FOR STELLAR SPECTRA

cross-modal generation: AspGap



Li,Wong,Hogg+2024

REPRESENTATION LEARNING

Literature: Huertas-Company & Sarmiento 2023

A brief review of contrastive learning applied to astrophysics

Marc Huertas-Company[1,2,3,4,5]★, Regina Sarmiento[1,2] and Johan H. Knapen[1,2]
[1]*Instituto de Astrofísica de Canarias (IAC), La Laguna, E-38205, Spain*
[2]*Departamento de Astrofísica, Universidad de La Laguna (ULL), E-38200, La Laguna, Spain*
[3]*Observatoire de Paris, LERMA, PSL University, 61 avenue de l'Observatoire, F-75014 Paris, France*
[4]*Université Paris-Cité, 5 Rue Thomas Mann, 75014 Paris, France*
[5]*Center for Computation Astrophysics, Flatiron Institute, 162 5th avenue, 10015 New York, USA*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT
Reliable tools to extract patterns from high-dimensionality spaces are becoming more necessary as astronomical datasets increase both in volume and complexity. Contrastive Learning is a self-supervised machine learning algorithm that extracts informative measurements from multi-dimensional datasets, which has become increasingly popular in the computer vision and Machine Learning communities in recent years. To do so, it maximizes the agreement between the information extracted from augmented versions of the same input data, making the final representation invariant to the applied transformations. Contrastive Learning is particularly useful in astronomy for removing known instrumental effects and for performing supervised classifications and regressions with a limited amount of available labels, showing a promising avenue towards *Foundation Models*. This short review paper briefly summarizes the main concepts behind contrastive learning and reviews the first promising applications to astronomy. We include some practical recommendations on which applications are particularly attractive for contrastive learning.

Key words: methods: data analysis – methods: statistical – methods: miscellaneous – techniques: miscellaneous

Unsupervised Clustering

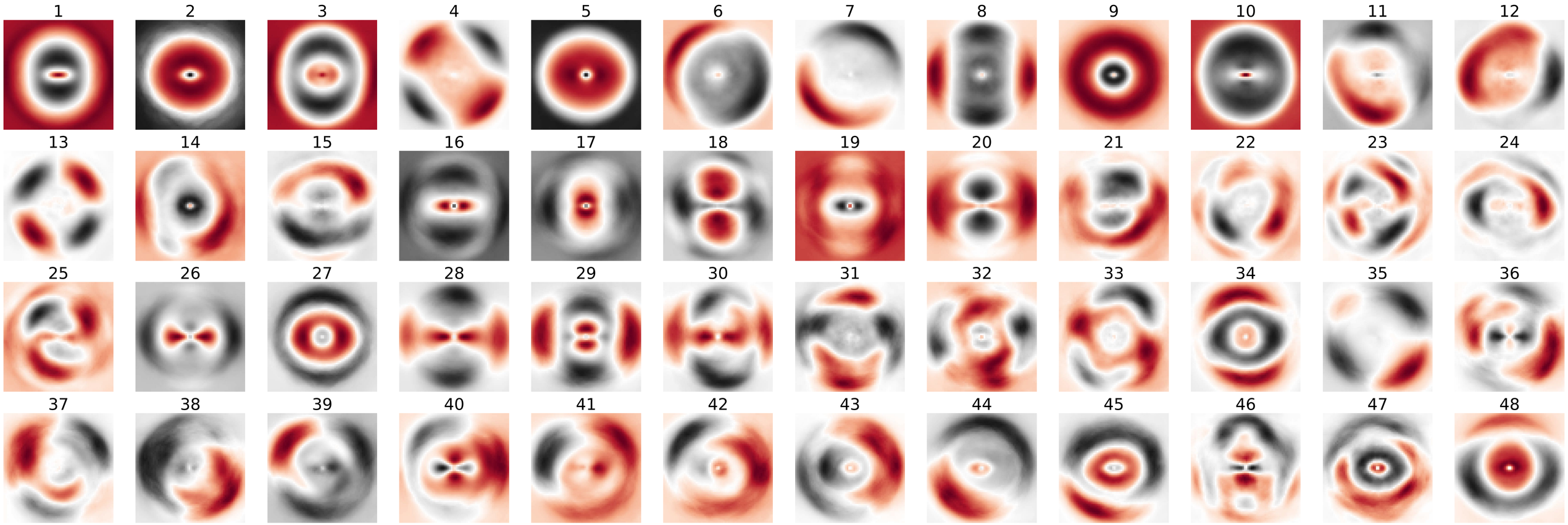
CLUSTERING AND DIMENSIONALITY REDUCTION

many data sources in astro are high-dimensional: e.g. spectra, images, IFU datacubes, …

most simple dimensionality reduction: Principal Component Analysis!

Eigengalaxies = Principle components of galaxy images

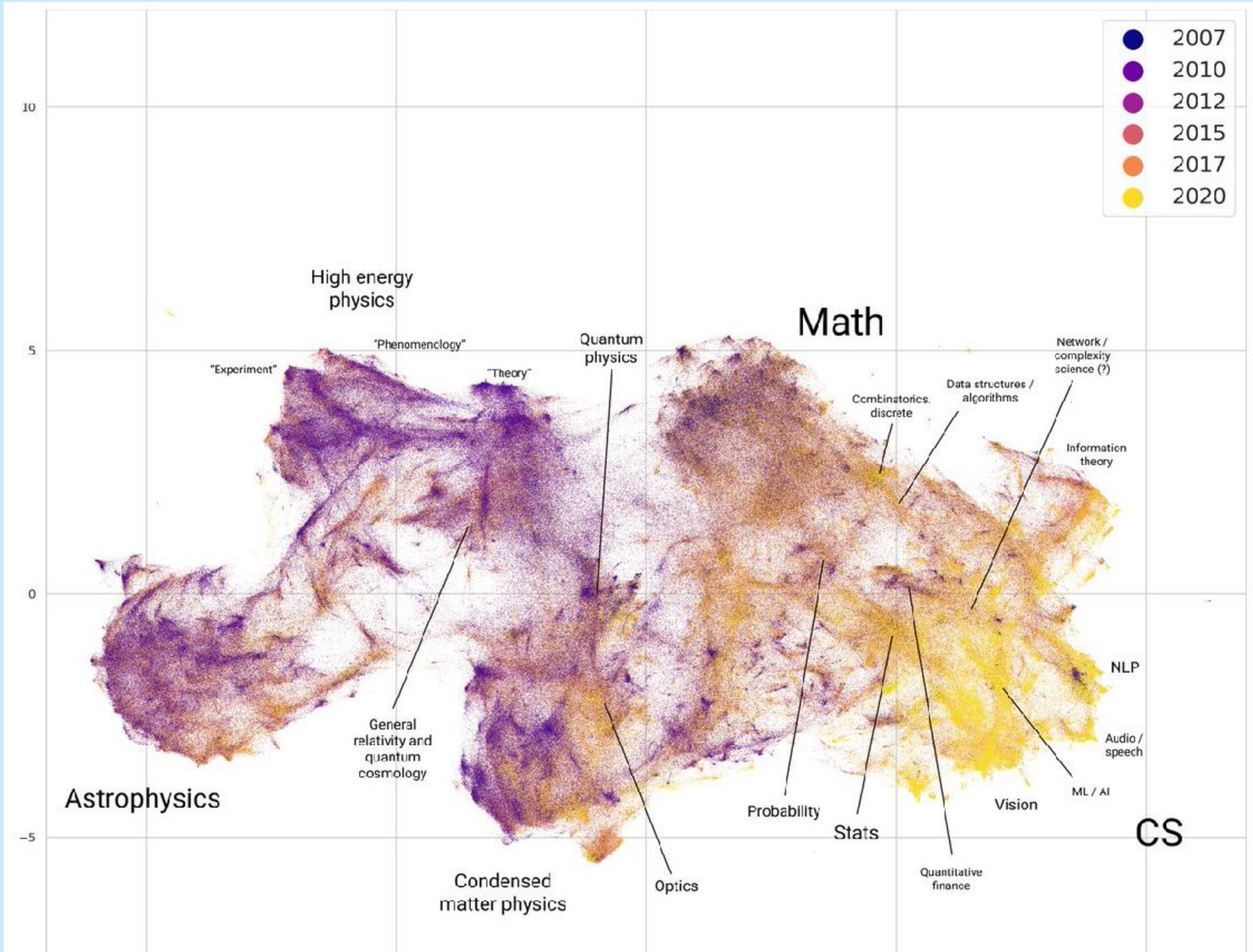
Uzeirbegovic+2020, Çakir+Buck 2024



CLUSTERING AND DIMENSIONALITY REDUCTION

many data sources in astro are high-dimensional: e.g. spectra, images, IFU datacubes, …

more complex dimensionality reduction: UMAP (Uniform Manifold Approximation Projection) and tSNE (t-distributed Stochastic Neighbour Embedding)



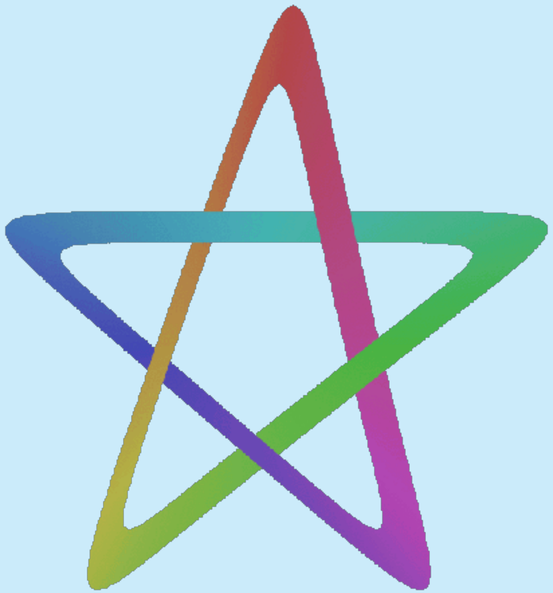
David McClure via Twitter, now X

CLUSTERING AND DIMENSIONALITY REDUCTION

most of the time we think in categories: stars vs. galaxies, elliptical vs. spirals, halos vs. filaments vs. voids

how to find/exploit structure in the data?

unsupervised clustering: Gaussian Mixture Models, k-means, HDBSCAN, … in general any halo finder



ASTROLINK

Oliver+2024

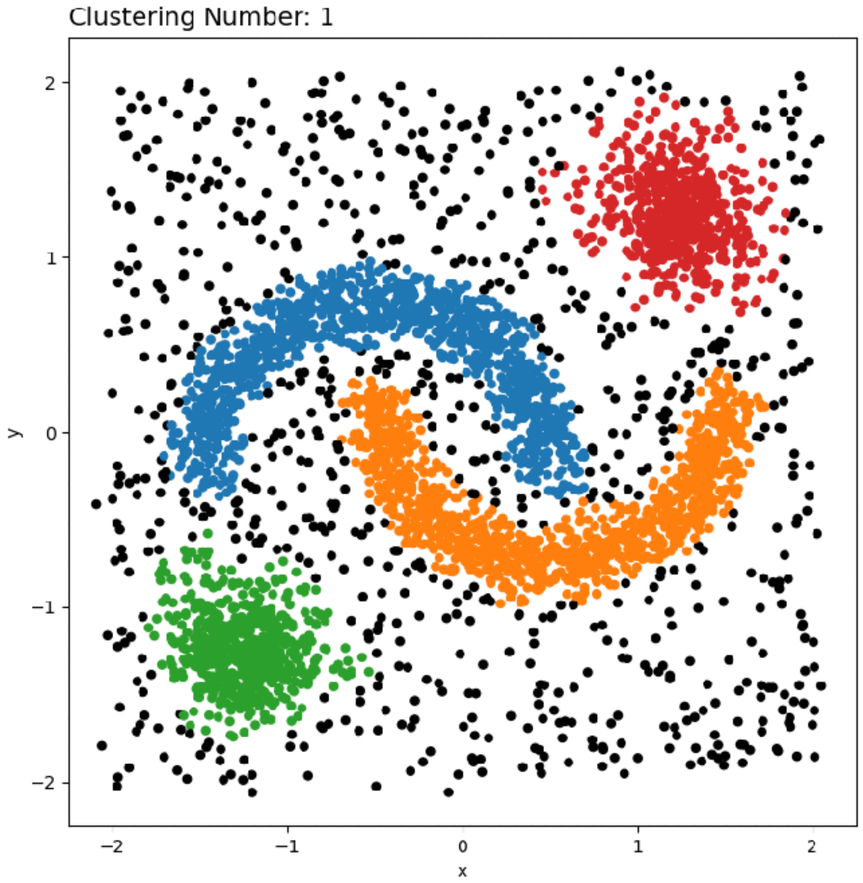
https://astrolink.readthedocs.io

CLUSTERING AND DIMENSIONALITY REDUCTION

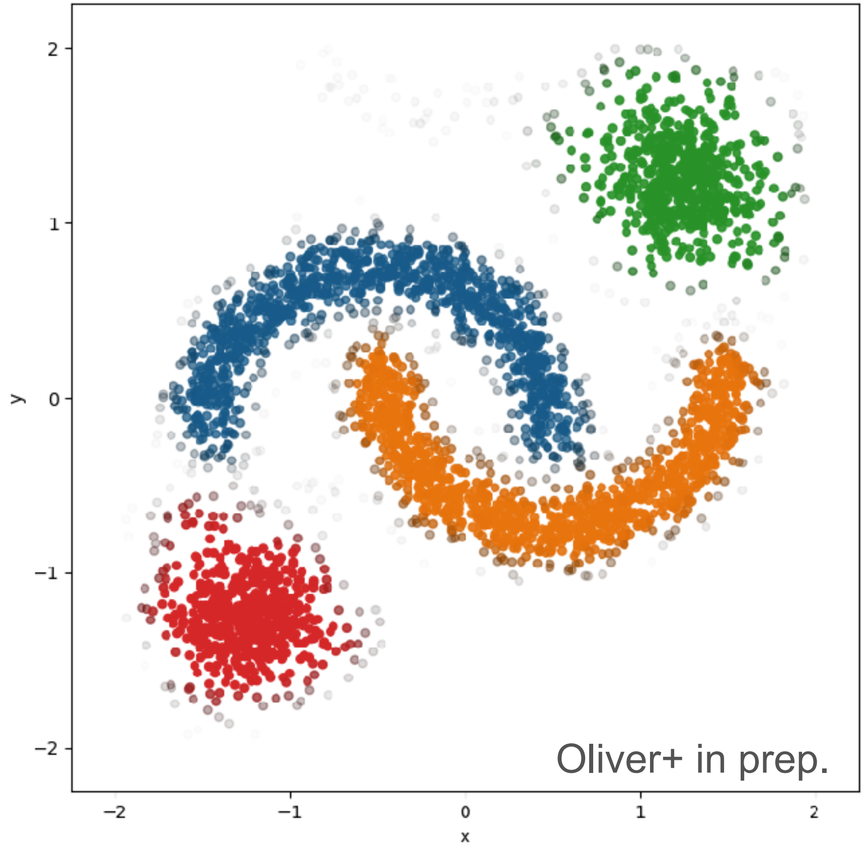
most of the time we think in categories: stars vs. galaxies, elliptical vs. spirals, halos vs. filaments vs. voids

how to find/exploit structure in the data?

What about measurement uncertainties or time evolving data?



https://fuzzycat.readthedocs.io

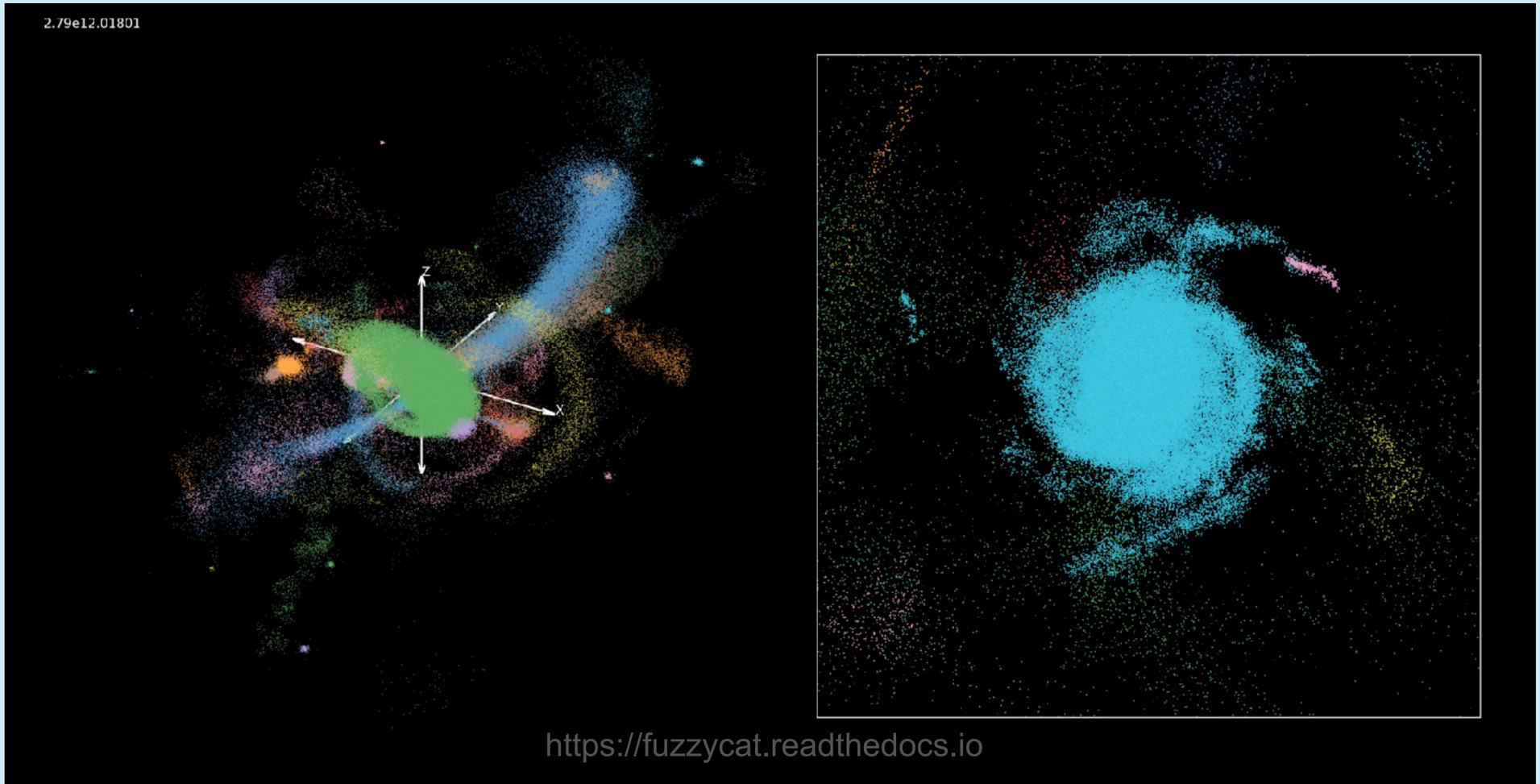


CLUSTERING AND DIMENSIONALITY REDUCTION

most of the time we think in categories: stars vs. galaxies, elliptical vs. spirals, halos vs. filaments vs. voids

how to find/exploit structure in the data?

What about measurement uncertainties or time evolving data?



https://fuzzycat.readthedocs.io

Oliver+ in prep.

Inference and Bayesian Modelling

MODELING COMPLEX PROBABILITY DISTRIBUTIONS

Bayesian inference aims at determining $p(\theta \,|\, \mathbf{x_o})$

$$p(\theta \,|\, \mathbf{x_0}) = \frac{p(\mathbf{x_0} \,|\, \theta) p(\theta)}{p(\mathbf{x_0})} \propto p(\mathbf{x_0} \,|\, \theta) p(\theta)$$

In astrophysics, $\mathbf{x_o}$ typically results from a large number of mechanisms/effects that transform the data and involve a large number of latent variables z, hence the marginal likelihood $p(\mathbf{x_o} \,|\, \theta)$ is intractable.

$$p(\mathbf{x_0} \,|\, \theta) = \int p(\mathbf{x_0} \,|\, \theta, z) p(z) dz$$

MODELING COMPLEX PROBABILITY DISTRIBUTIONS

How can we approximate high-dimensional, complex probability distributions $p(\theta \mid \mathbf{x_o})$?

Goal:

- effectively: learn a model from the data!
- model $p(\theta \mid \mathbf{x_o})$ explicitly or implicitly
- sample and evaluate $p(\theta \mid \mathbf{x_o})$

Options:

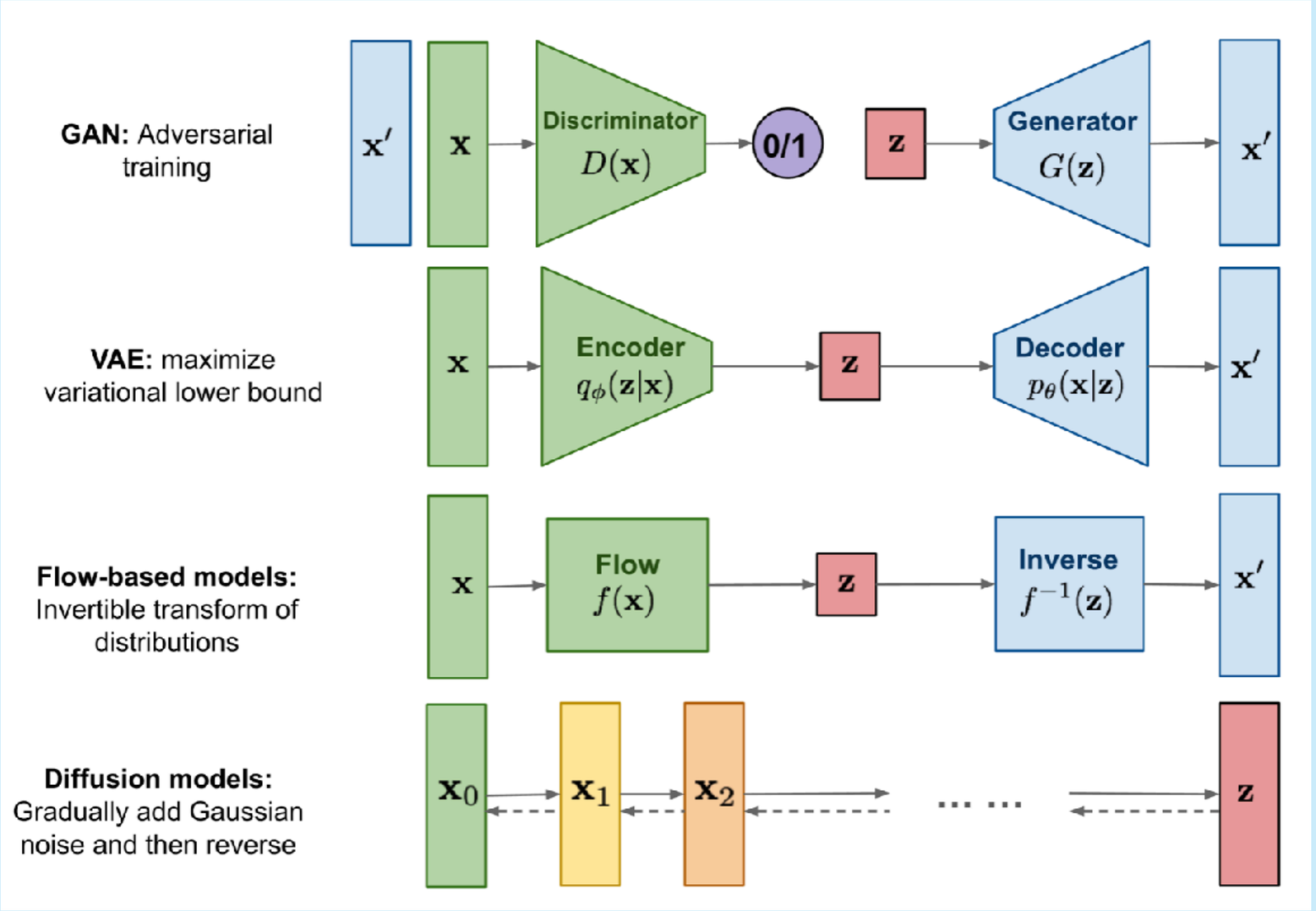
- normalizing flows
- VAEs
- GANs
- score matching / flow matching
- and possibly more

implicit model: architectural constraints

explicit model: prone to mode collapse

GENERATIVE AI FLAVOURS

"Creating noise from data is easy; creating data from noise is generative modeling.“ (Song+2020)



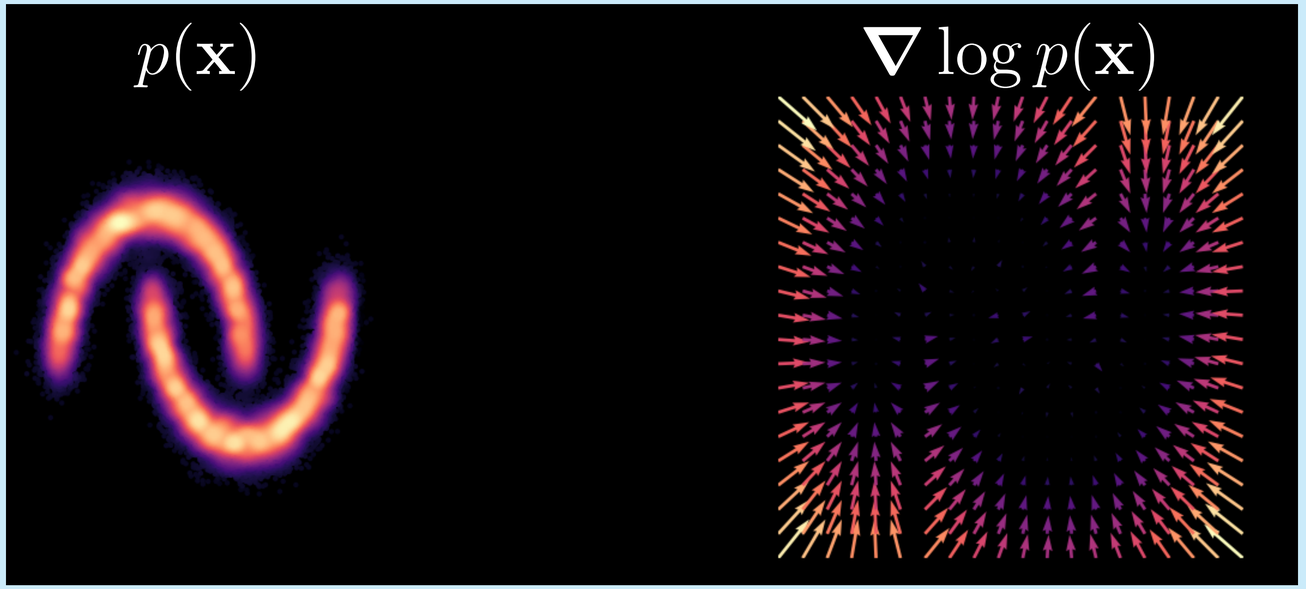
blog post by https://lilianweng.github.io/posts/2021-07-11-diffusion-models/


Score Matching

SCORE MATCHING AND DIFFUSION MODELS

What is the score $s(\mathbf{x})$ of a pdf $p(\mathbf{x})$?

$$s(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$$



Idea: Learn $p(\mathbf{x})$ solely from data samples, then sample new instances.

But how does this work? → Langevin dynamics (Welling&Teh 2011)

blog post by https://jmtomczak.github.io/blog/16/16_score_matching.html

THE IDEA BEHIND SCORE MATCHING

Sohl-Dickstein et al. (2015), Ho et al. (2020), Song et al. (2021)

Forward SDE: $$\mathrm{d}\mathbf{z}_t = -\frac{1}{2}\beta(t)\mathbf{z}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}_t$$

Easy

$t = 0$

$\mathbf{z}_0 \sim p_{\text{target}}$

$t = 1$

$\mathbf{z}_1 \sim \mathcal{N}(0, I_d)$

Hard

Backward SDE: $$\mathrm{d}\mathbf{z}_t = \left[-\frac{1}{2}\beta(t)\mathbf{z}_t - \beta(t)\,\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\overline{\mathbf{w}}_t$$

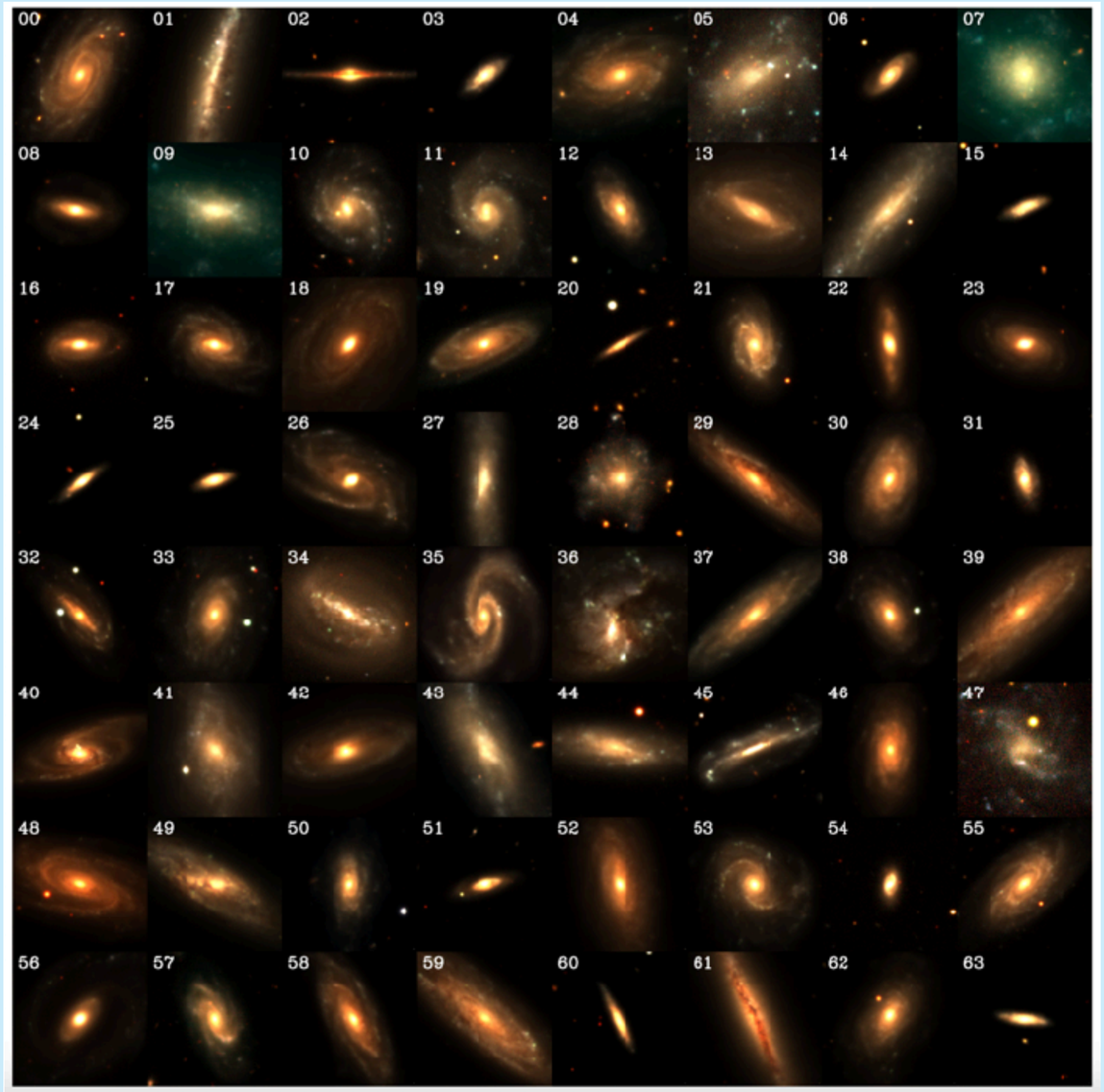
Score function

adapted from: slides by Bruno Régaldo-Saint Blancard and blog post by https://jmtomczak.github.io/blog/16/16_score_matching.html

APPLICATION OF SCORE MATCHING AND DIFFUSION MODELS

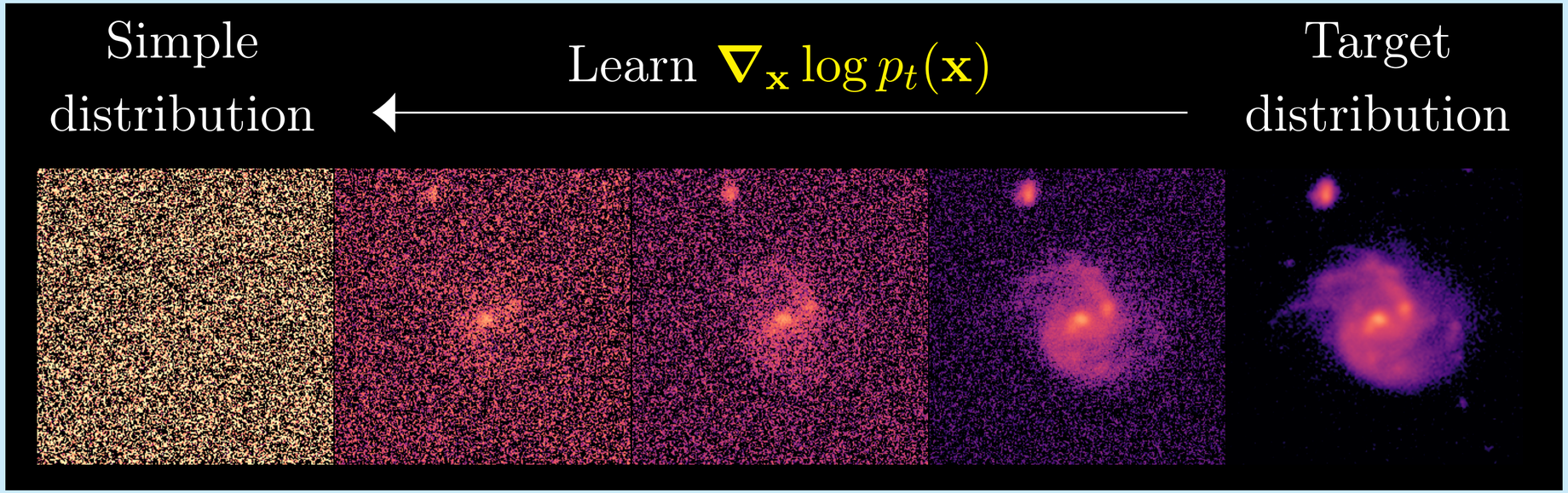
Learn generative model purely from data!

Smith+2021



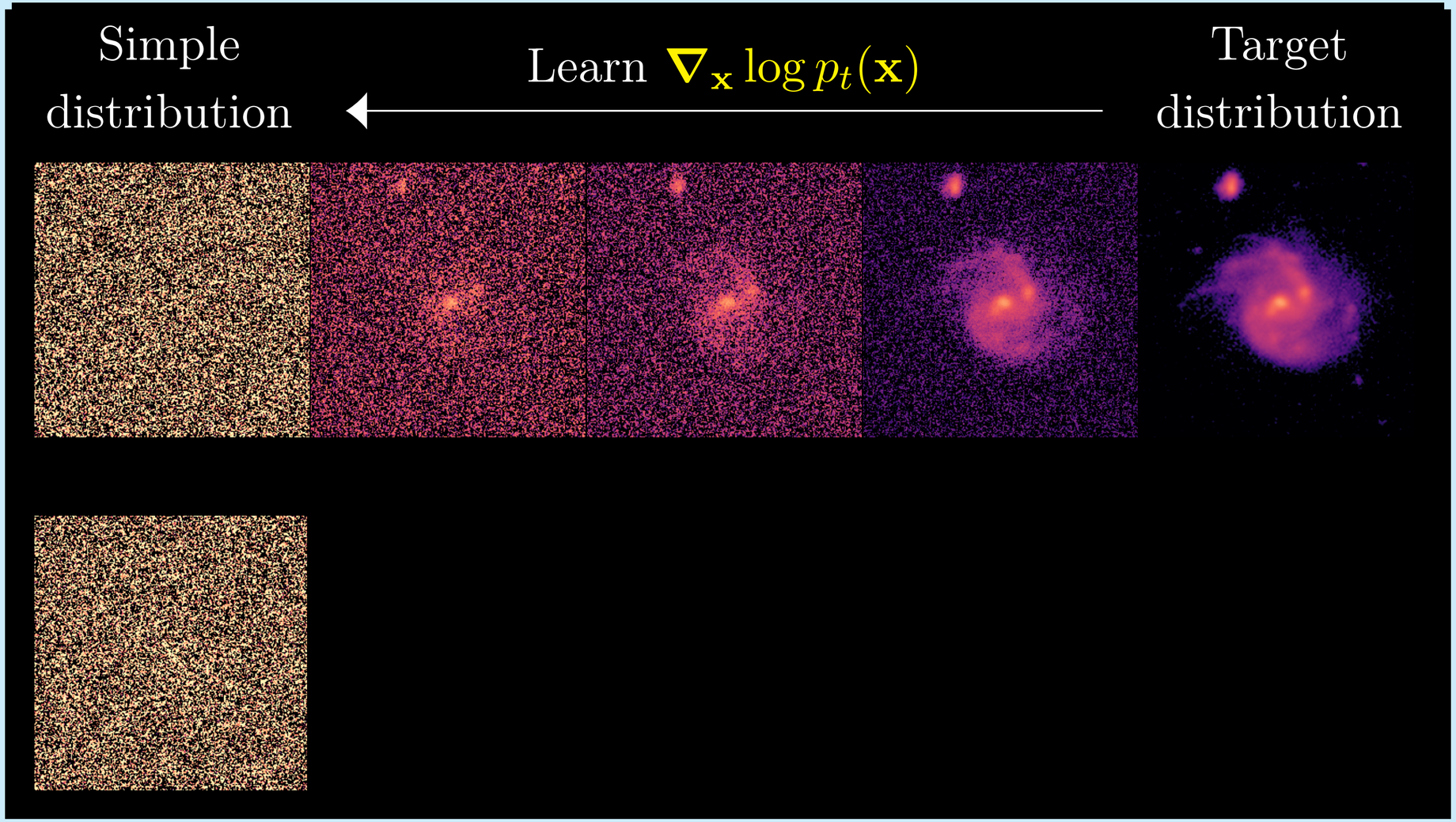
http://www.mjjsmith.com/thisisnotagalaxy/

TRAINING DIFFUSION MODELS



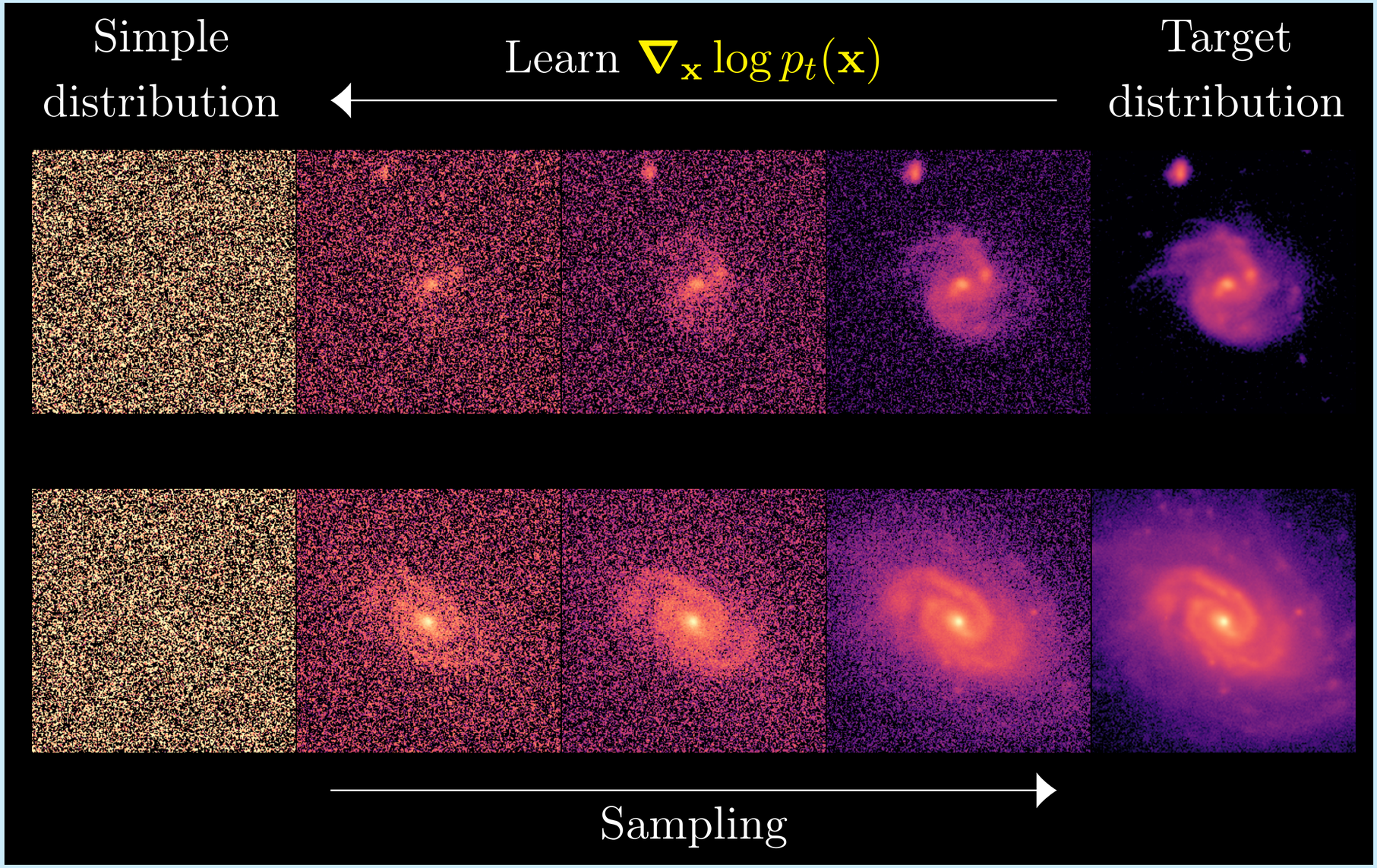
slides adapted from Alexandre Adams

TRAINING DIFFUSION MODELS



slides adapted from Alexandre Adams

TRAINING DIFFUSION MODELS



slides adapted from Alexandre Adams

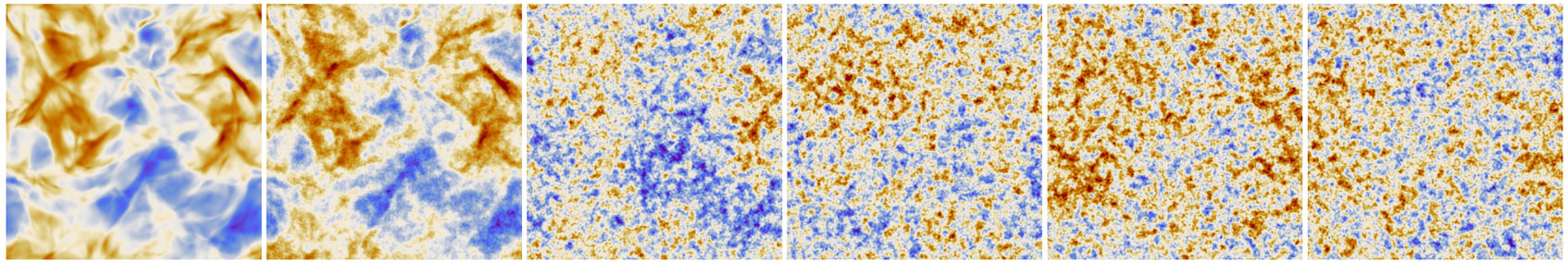
DIFFUSION MODELS AS A MAPPING $p_{\text{dust}} \leftrightarrow p_{\text{CMB}}$

Forward SDE: $$\mathrm{d}\mathbf{z}_t = -\frac{1}{2}\beta(t)\mathbf{z}_t\,\mathrm{d}t + \sqrt{\beta(t)}\boldsymbol{\Sigma}_{\phi}^{\frac{1}{2}}\,\mathrm{d}\mathbf{w}_t$$

Easy

$t = 0$

$\mathbf{z}_0 \sim p_{\text{dust}}$



$t = 1$

$\mathbf{z}_1 \sim p_{\text{CMB}}$

$\mathcal{T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$

Hard

Backward SDE: $$\mathrm{d}\mathbf{z}_t = \left[-\frac{1}{2}\beta(t)\mathbf{z}_t - \beta(t)\boldsymbol{\Sigma}_{\phi}\,\underbrace{\nabla_{\mathbf{z}_t}\log p_t(\mathbf{z}_t)}_{\text{Score function}}\right]\mathrm{d}t + \sqrt{\beta(t)}\boldsymbol{\Sigma}_{\phi}^{\frac{1}{2}}\,\mathrm{d}\overline{\mathbf{w}}_t$$

Heurtel-Depeignes+2023

slides adapted from Bruno Régaldo-Saint Blancard

DIFFUSION MODELS AS A MAPPING $p_{\text{dust}} \leftrightarrow p_{\text{CMB}}$

Component Separation ?

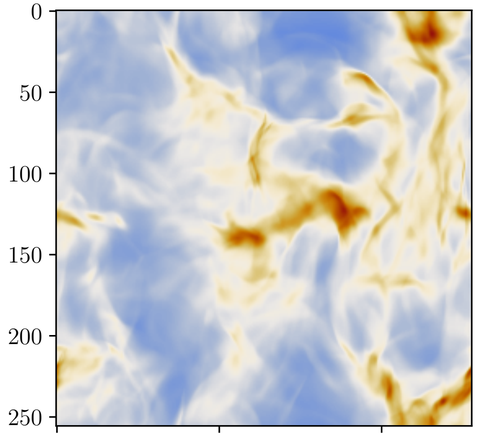
Visual assessment:

Dust $\mathbf{x}$

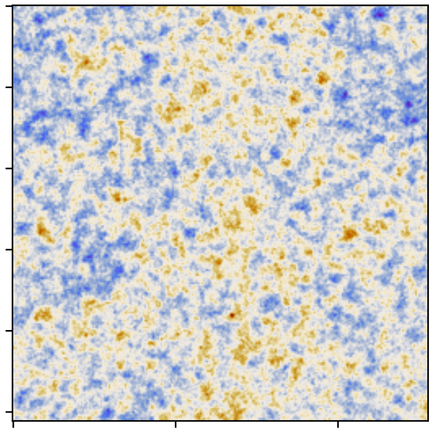
CMB ϵ

Observation y

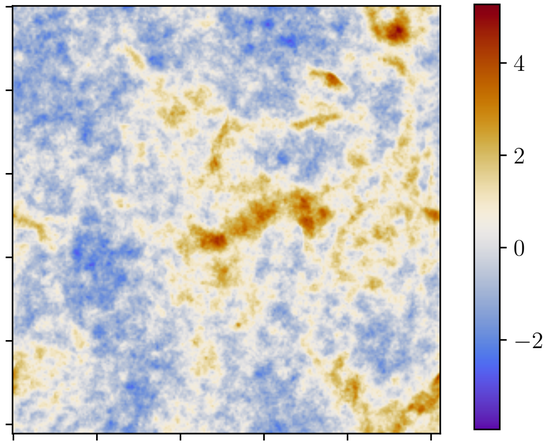
True



+



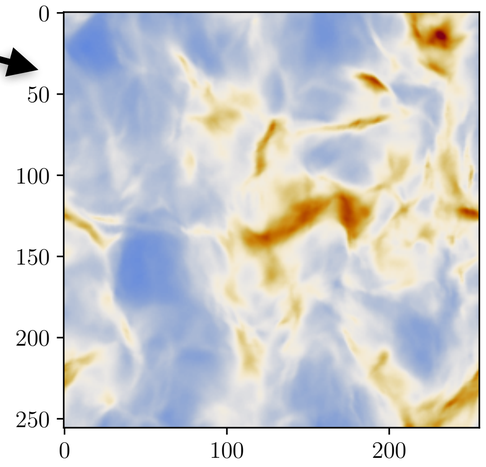
=



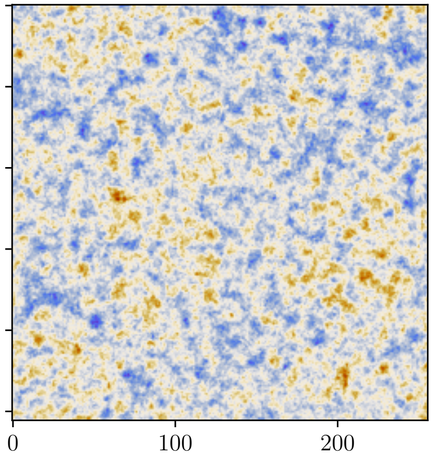
Sample from

$p(\mathbf{x} \mid \mathbf{y}, \phi)$

Reconstructed



+



=

Heurtel-Depeignes+2023

slides adapted from Bruno Régaldo-Saint Blancard

POSTERIOR SAMPLES WITH DIFFUSION MODELS

Posterior $p(x \mid y)$ with observation y is given by Bayes' theorem:

$$\log p(x \mid y) = \log p(y \mid x) + \log p(x) - \log p(y)$$

with $p(y \mid x)$ being the likelihood
and $p(x)$ the prior.

hence the score is given by:

$$\nabla_x \log p(x \mid y) = \nabla_x \log p(y \mid x) + \nabla_x \log p(x) - \nabla_x \log p(y)$$

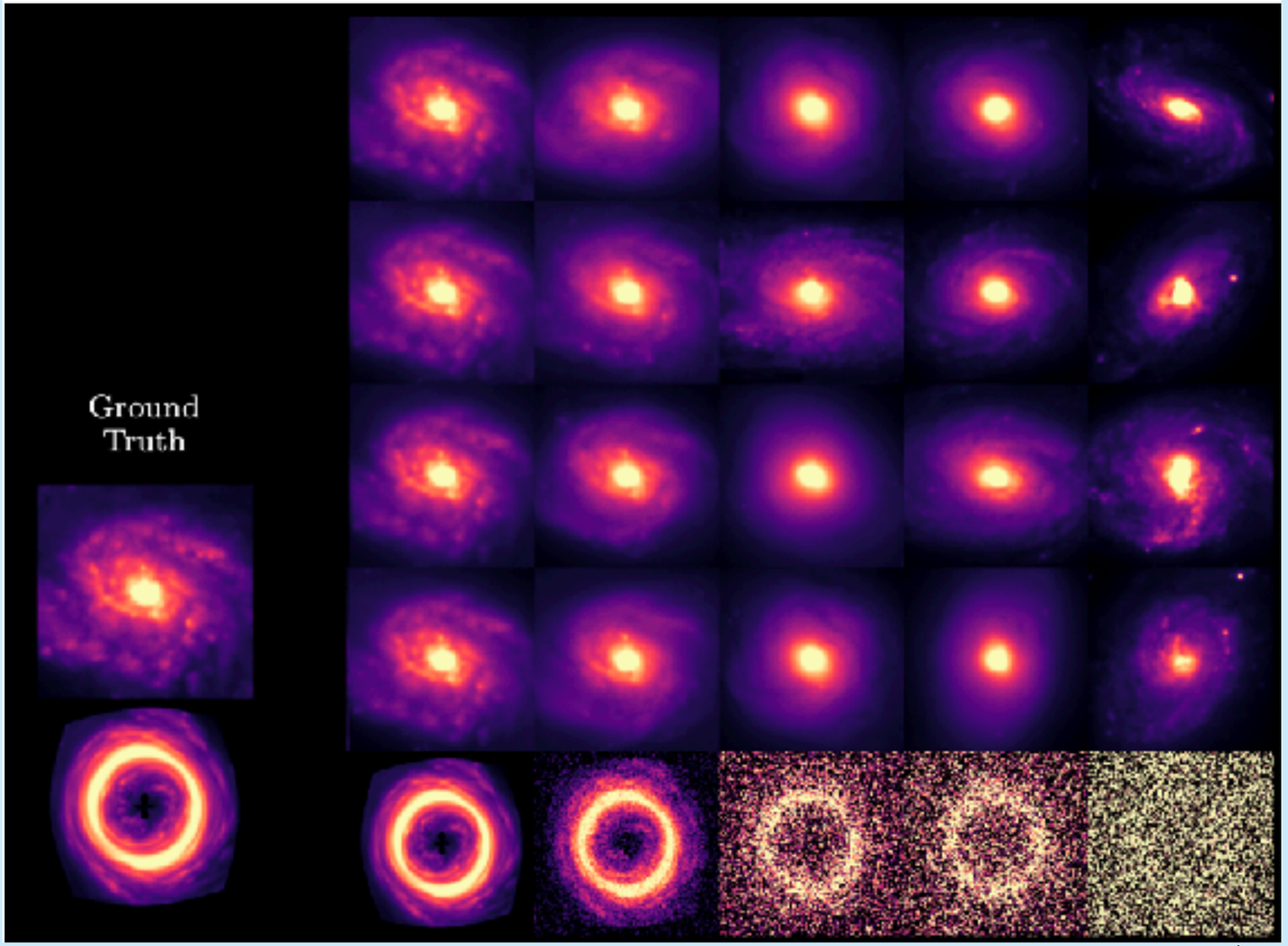
To a good approximation, we can calculate the likelihood score analytically if we assume it's Gaussian.

This is the score we learnt with Score Matching from data!

=0 because it does not depend on x.

POSTERIOR SAMPLES WITH DIFFUSION MODELS

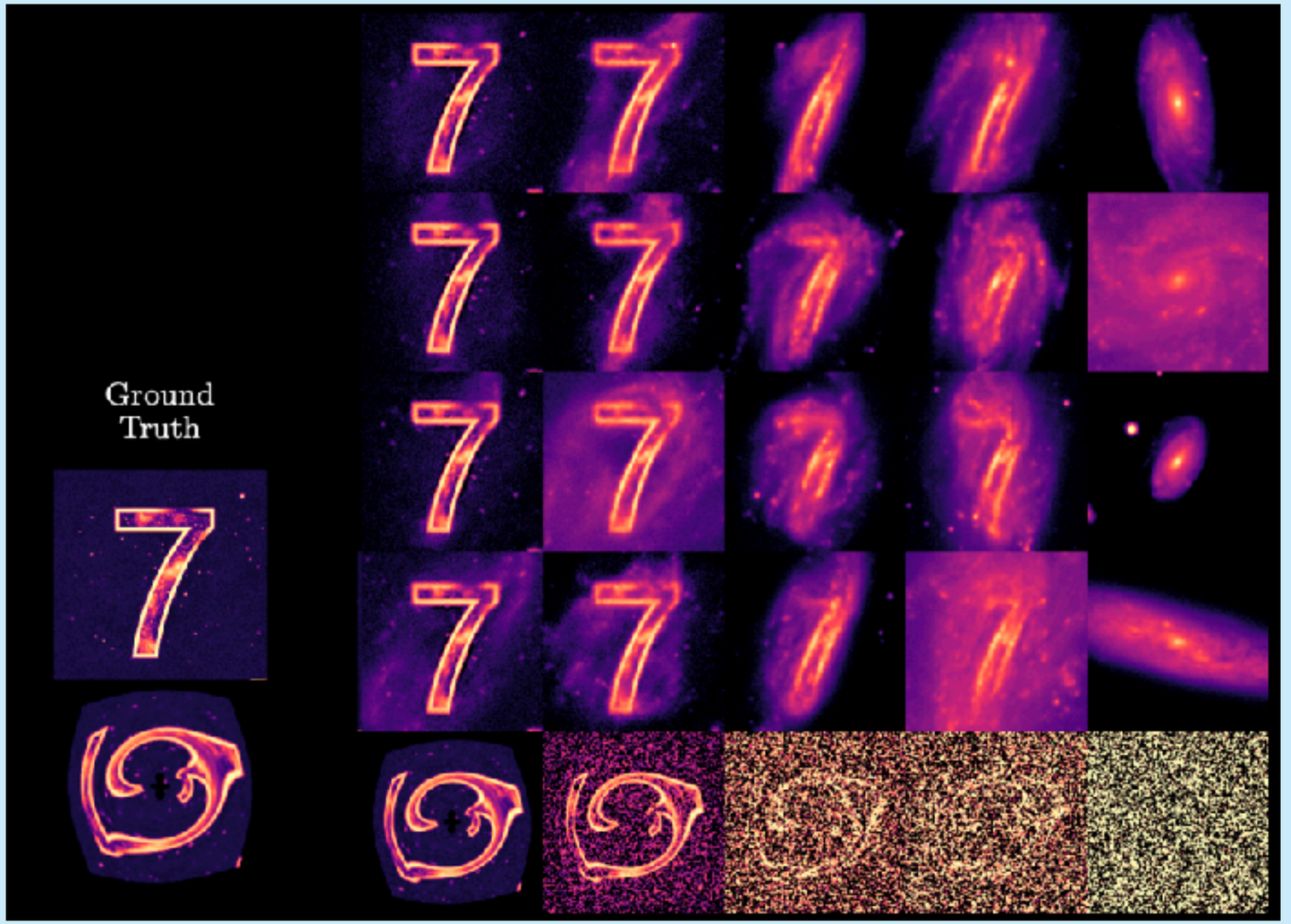
Posterior samples of source galaxies in strong gravitational lenses with score-based priors



Adam+2022

POSTERIOR SAMPLES WITH DIFFUSION MODELS

Posterior samples for out-of-distribution galaxies



posterior samples

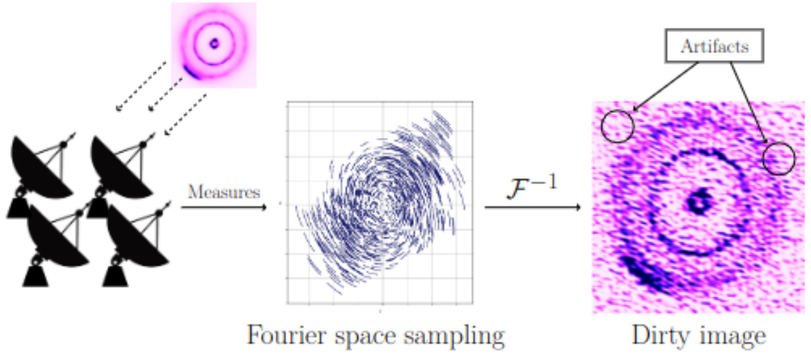
Adam+2022

less informative likelihood

MORE APPLICATIONS OF DIFFUSION MODELS

Bayesian Imaging for Radio Interferometry with Score-Based Priors

Noé Dia[1,2,4] M. J. Yantovski-Barth[1,2,4] Alexandre Adam[1,2,4] Micah Bowles[5]
Pablo Lemos[1,2,3,4] Anna M. M. Scaife[5,6] Yashar Hezaveh[1,2,3,4,7,8]
Laurence Perreault-Levasseur[1,2,3,4,7,8]
[1]Université de Montréal [2]Ciela Institute [3]Flatiron Institute [4]Mila [5]University of Manchester
[6]The Alan Turing Institute [7]Trottier Space Institute [8]Perimeter Institute



THE ASTROPHYSICAL JOURNAL LETTERS

OPEN ACCESS

Beyond Gaussian Noise: A Generalized Approach to Likelihood Analysis with Non-Gaussian Noise

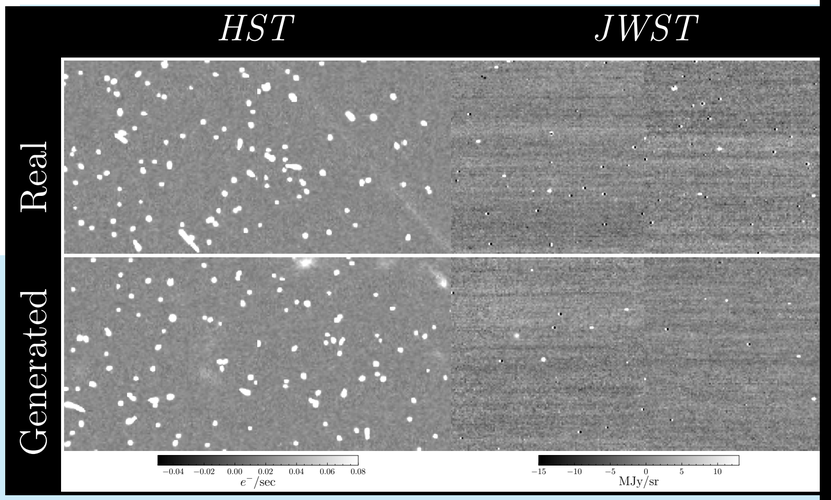
Ronan Legin[5,1,2,3], Alexandre Adam[5,1,2,3], Yashar Hezaveh[1,2,3,4], and Laurence Perreault-Levasseur[1,2,3,4]

Published 2023 June 6 • © 2023. The Author(s). Published by the American Astronomical Society.

The Astrophysical Journal Letters, Volume 949, Number 2

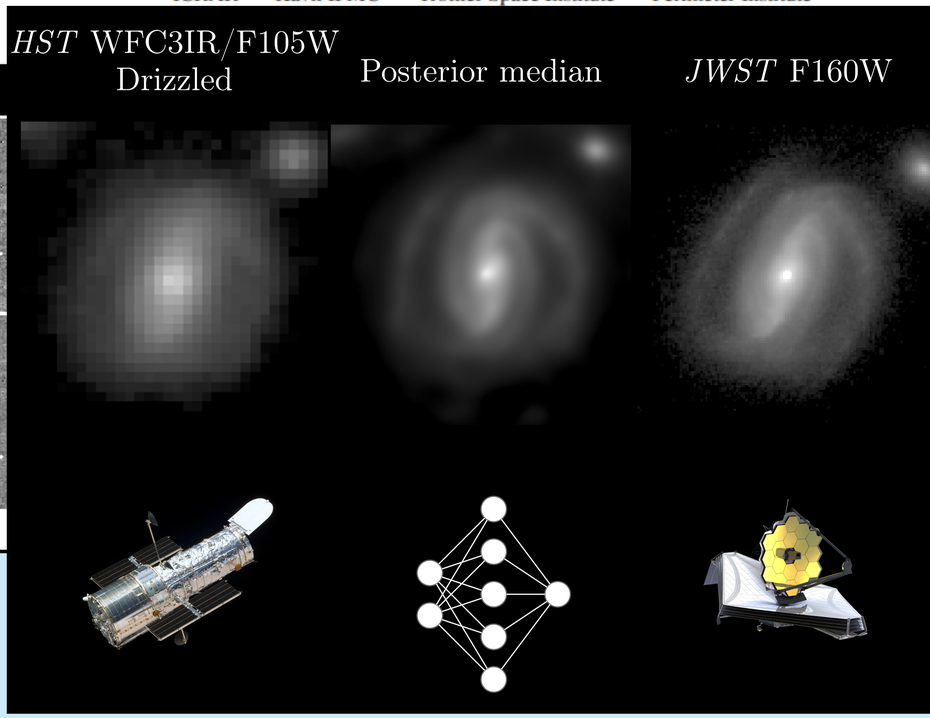
Citation Ronan Legin *et al* 2023 *ApJL* **949** L41

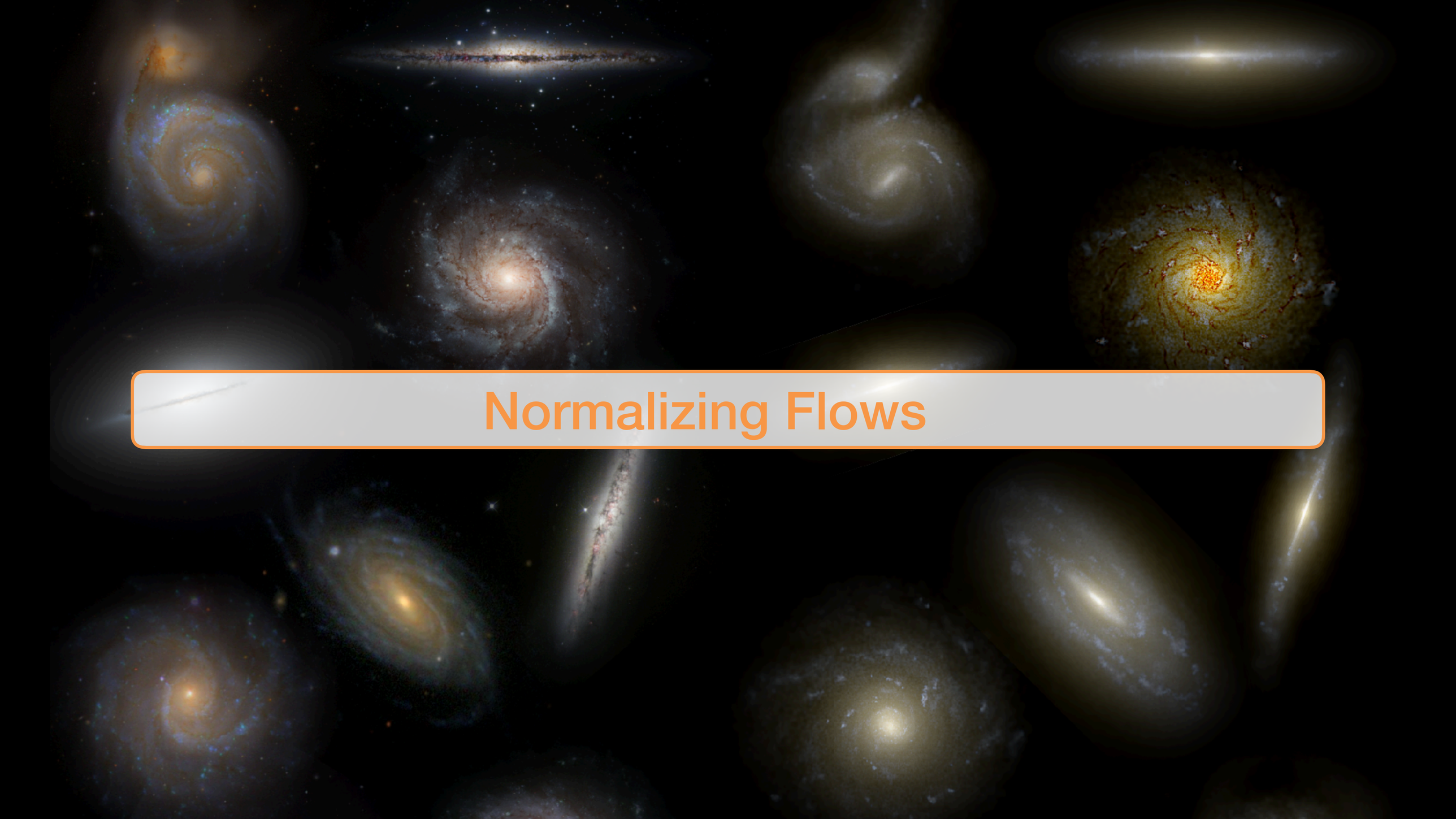
DOI 10.3847/2041-8213/acd645



Echoes in the Noise: Posterior Samples of Faint Galaxy Surface Brightness Profiles with Score-Based Likelihoods and Priors

Alexandre Adam[1,2,4] Connor Stone[1,2,4] Connor Bottrell[5,6] Ronan Legin[1,2,4]
Yashar Hezaveh[1,2,3,4,7,8] Laurence Perreault-Levasseur[1,2,3,4,7,8]
[1]Université de Montréal [2]Ciela Institute [3]CCA, Flatiron Institute [4]Mila
[5]ICRAR [6]Kavli IPMU [7]Trottier Space Institute [8]Perimeter Institute





Normalizing Flows

NORMALIZING FLOWS

How can we approximate high-dimensional, complex probability distributions $p(\theta \mid \mathbf{x_o})$?

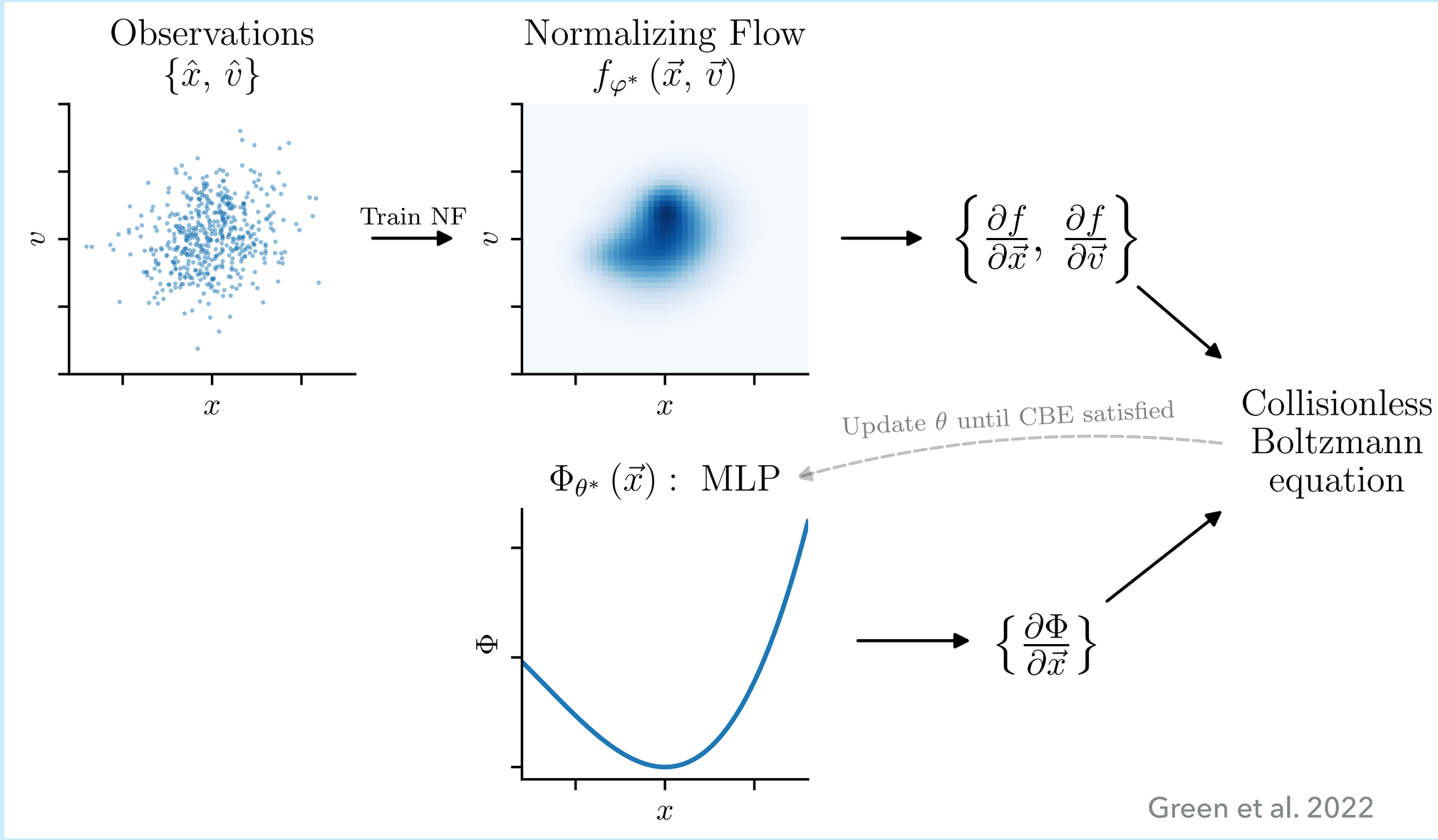
Goal:

- model $p(\theta \mid \mathbf{x_o})$ explicitly
- sample and evaluate $p(\theta \mid \mathbf{x_o})$

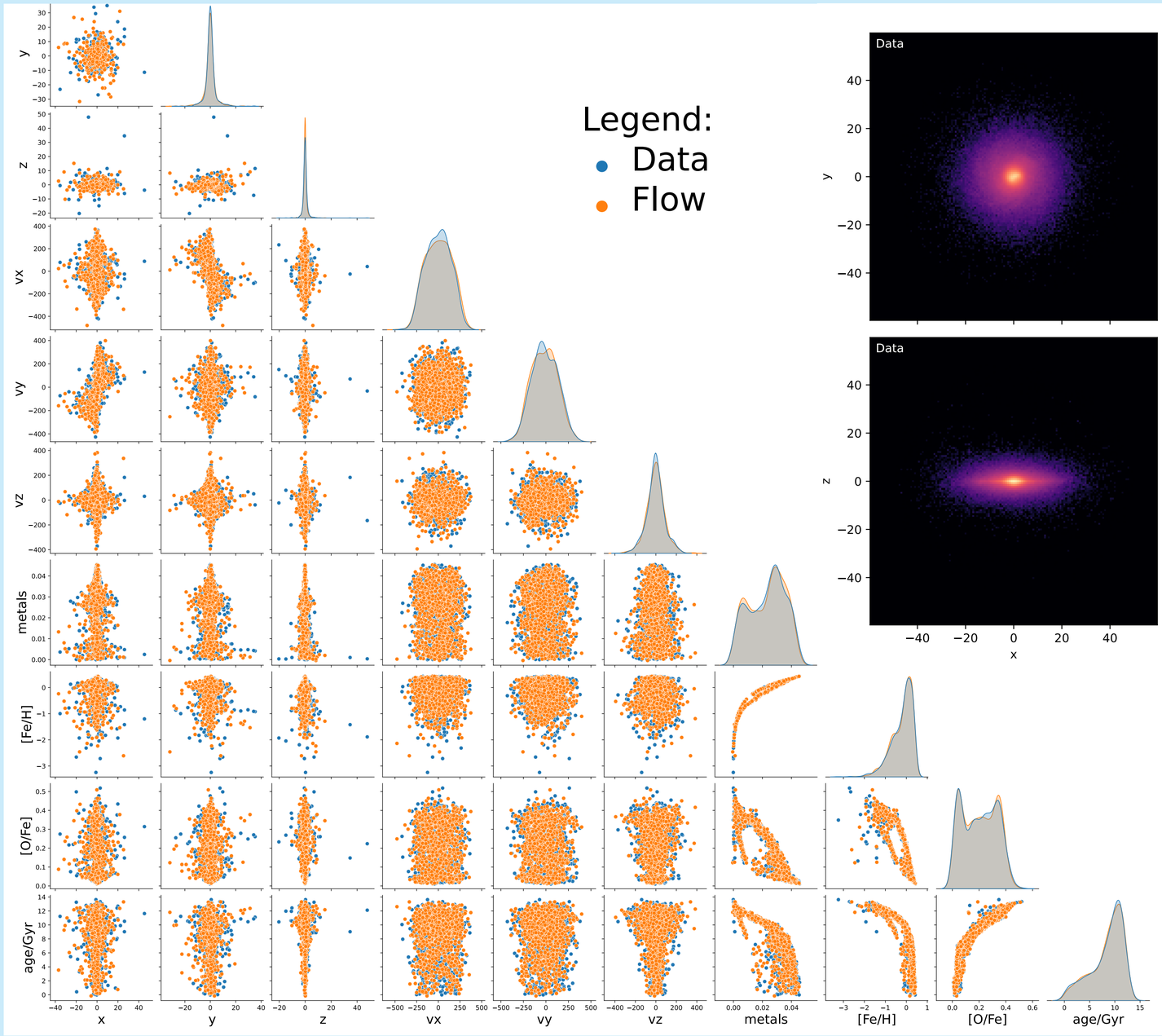
Idea:

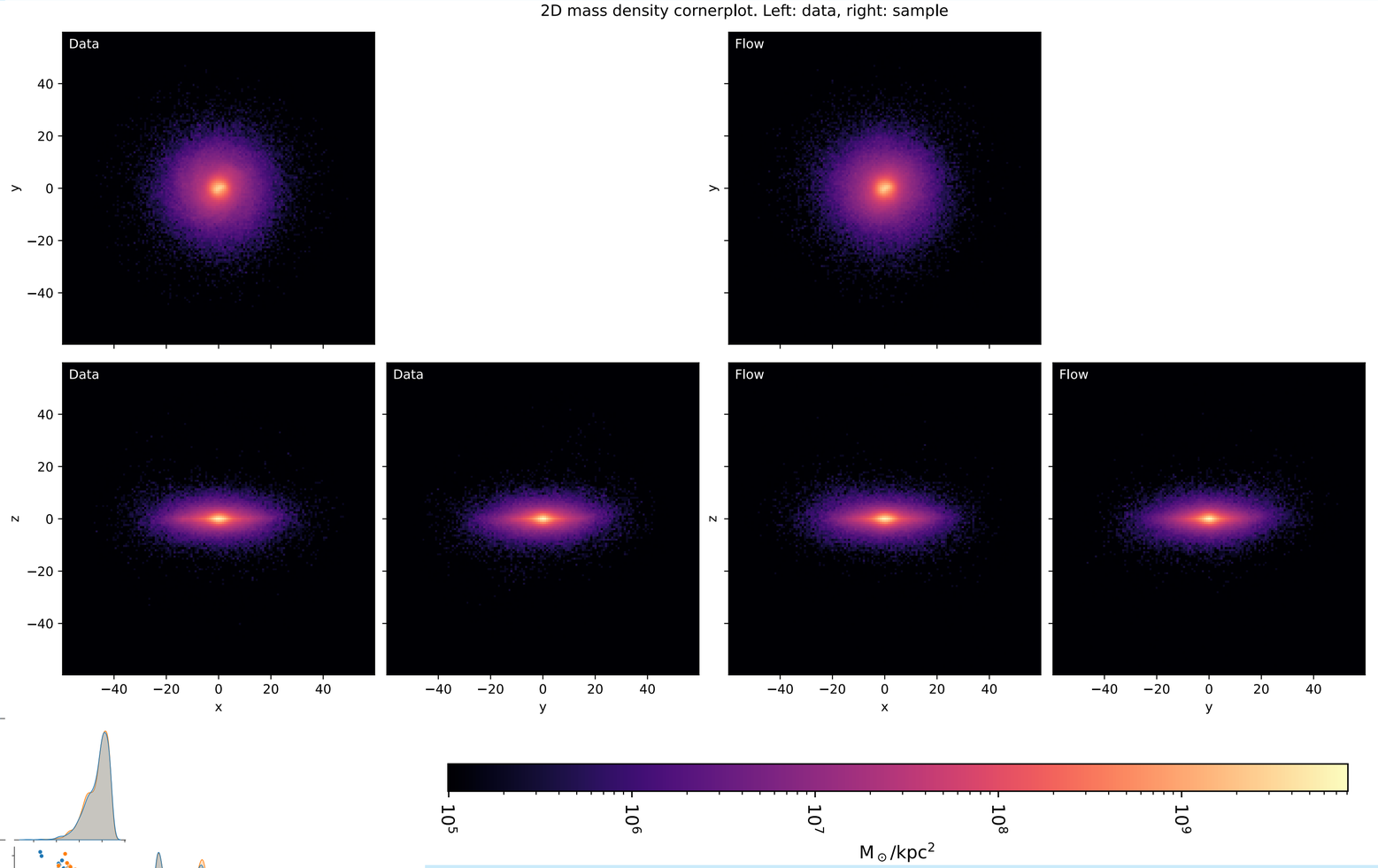
Transform a simple base distribution through a series of invertible transformations.

NORMALISING FLOW: APPLICATION I



NORMALISING FLOW: APPLICATION II



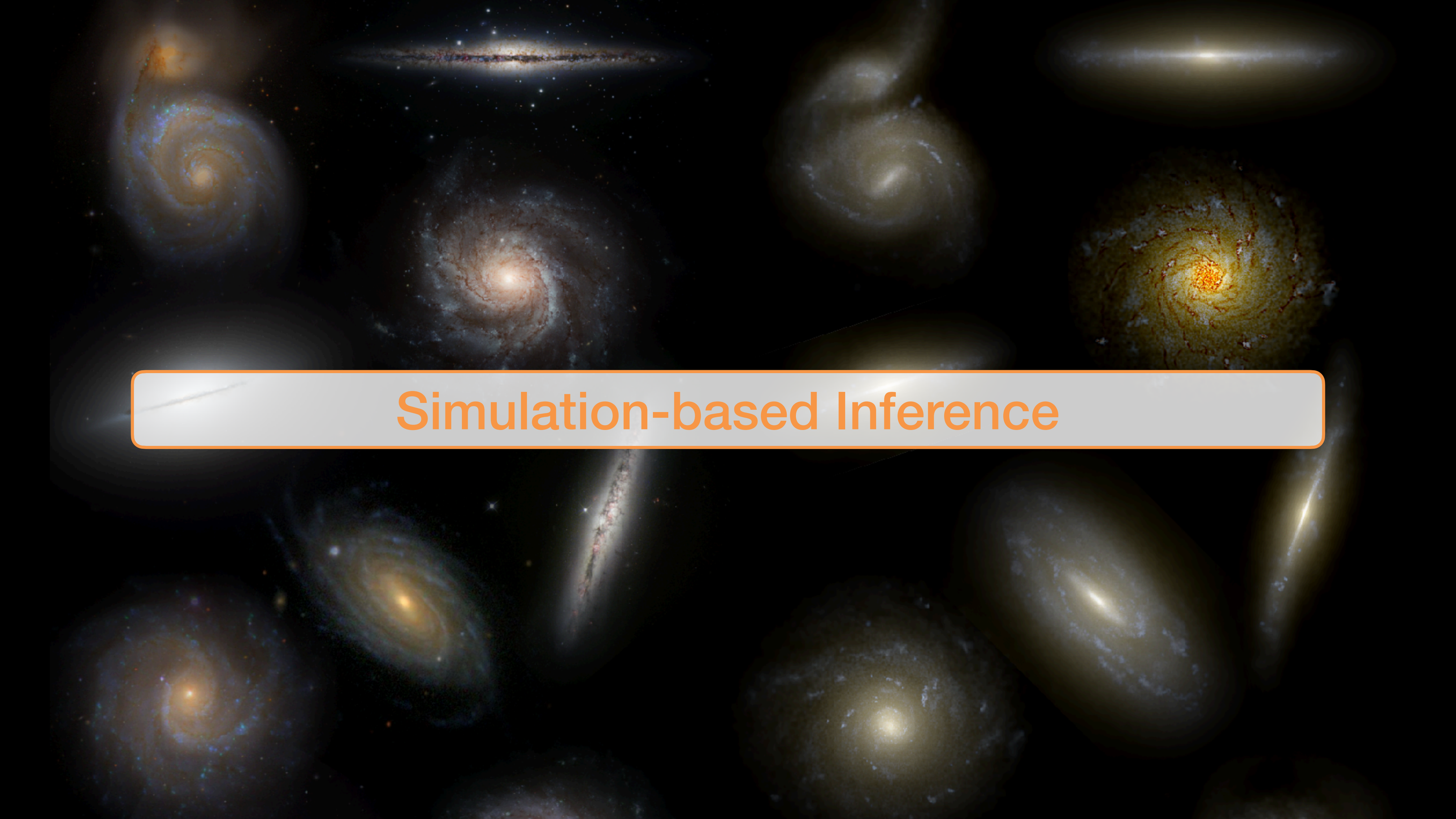


Wolf+Buck 2023

NORMALISING FLOW: APPLICATION III

- Normalizing flows for random fields in cosmology (Rouhiainen+2021)
- Bayesian Stokes inversion with normalizing flows (Baso+2022)
- A Hierarchy of Normalizing Flows for Modelling the Galaxy-Halo Relationship (Lovell+2023)
- HIFlow: Generating Diverse Hi Maps and Inferring Cosmology while Marginalizing over Astrophysics Using Normalizing Flows (Hassan+2022)
- Normalizing Flows as an Avenue to Studying Overlapping Gravitational Wave Signals (Langendorff+2023)
- Charting Galactic Accelerations: When and How to Extract a Unique Potential from the Distribution Function (An+2021)
- Charting galactic accelerations II: how to ‘learn’ accelerations in the solar neighbourhood (Naik+2021)

- many many more…


Simulation-based Inference

SIMULATION-BASED INFERENCE - SBI

$$p(\theta \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \theta) p(\theta)}{p(\mathbf{x})} \propto p(\mathbf{x} \,|\, \theta) p(\theta)$$

- Insight: running a **stochastic simulator** with input θ gives an output $\mathbf{x}$ that is drawn from an implicit likelihood $p(\mathbf{x} \,|\, \theta)$
- „simulation-based inference“ or „likelihood-free inference“ or „implicit likelihood inference“ or … (review: Cranmer+2020)
- recent progress thanks to deep learning algorithms, e.g. conditional normalizing flows (Papamarkios+2019, Greenberg+2019, Hermans+2020, …)

SBI: NEURAL X ESTIMATION

- Use neural networks to approximate some quantities in Bayes' formula

$$p(\theta \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \theta)}{p(\mathbf{x})} p(\theta) = \frac{p(\mathbf{x} \mid \theta)}{p(\mathbf{x})} p(\theta)$$

  - Neural Posterior Estimation (NPE)
  - Neural Likelihood Estimation (NLE)
  - Neural Ratio Estimation (NRE)

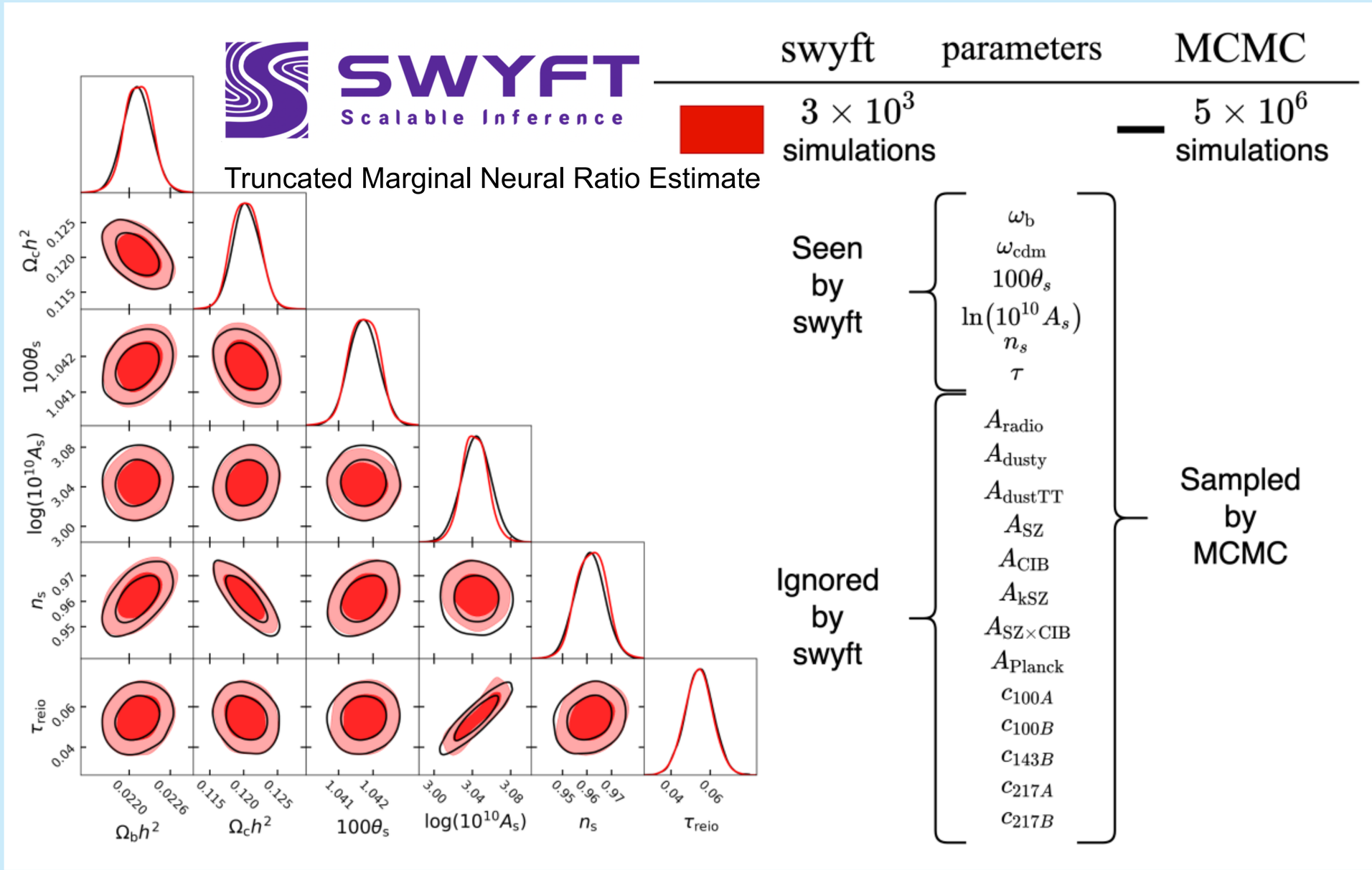
SBI: (CONDITIONAL) DENSITY ESTIMATION

- NLE and NPE both estimate normalised probability densities, hence:
  - restricted network architectures, e.g. normalizing flows or mixture density models. potentially difficult to train (Papamarkios+2021)
  - for high-dimensional data, compression/embedding network needed.
- but: restriction can be a good inductive bias, especially if posterior or likelihood is “perturbation around Gaussian distribution”
- automatic marginalization possible

c.f. pydelfi Alsing+2018,2019; moment networks Jeffrey+Wandelt 2020, SBI Jakob Macke, ltU-ili Ho+2024, Bayesflow Radev+2020,2023, swyft Miller+2021,2022

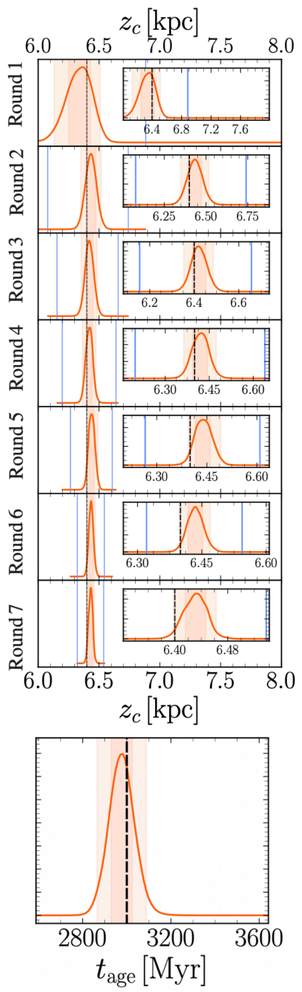
https://simulation-based-inference.org/
https://github.com/smsharma/awesome-neural-sbi for references to software and applications

SBI: APPLICATION IN COSMOLOGY



Cole+2022

SBI: APPLICATION FOR STELLAR STREAMS



TRUNCATED MARGINAL NEURAL RATIO ESTIMATION (TMNRE) FOR STELLAR STREAMS

STEP 1: (RE-) SIMULATE

- Sample parameters θ from (truncated) prior $p(\theta)$
- Simulate data $x \sim p(x \mid \theta)$

$$\theta \equiv (t_{\text{age}}, \sigma_v, \ldots) \rightarrow x = \text{stream} + \text{bkg}\,.$$

STEP 2: RATIO ESTIMATION

- Train ratio estimators $r(x; \theta_i)$ on simulated data to approximate the posterior-to-prior ratio $r(x; \theta_i) \sim p(\theta_i \mid x)/p(\theta_i)$ for each parameter of interest θ_i

sstrax
FAST STELLAR STREAM MODELLING IN JAX

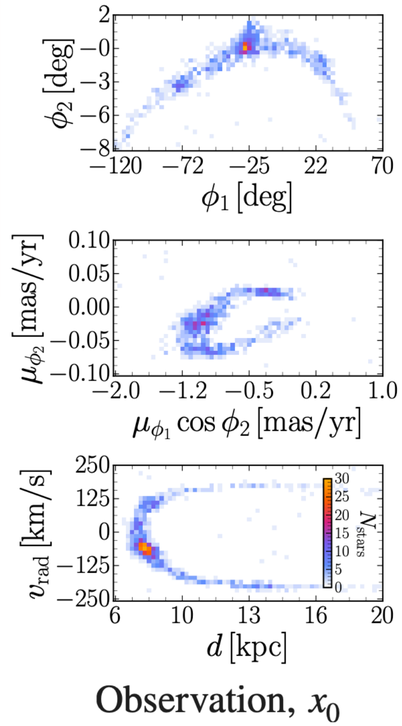
ALBATROSS
STELLAR STREAM PARAMETER INFERENCE WITH NEURAL RATIO ESTIMATION

STEP 3: INFERENCE

- Obtain a prior sample from $p(\theta_i)$
- Target a specific observation x_0 and compute the ratios $r(x_0; \theta_i)$ across the prior sample
- Weight the samples according to this ratio

STEP 4: TRUNCATION

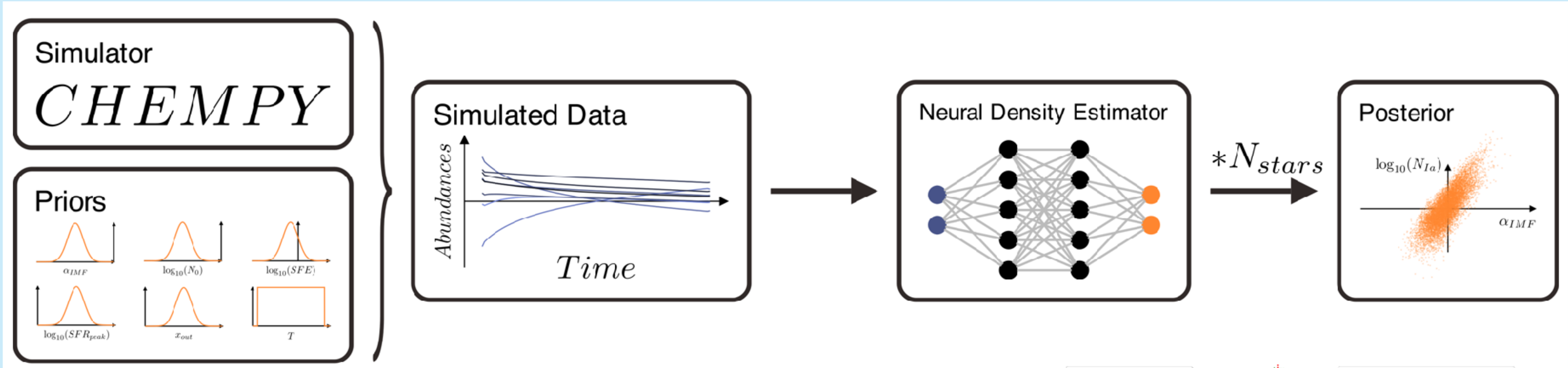
- Use ratios $r(x_0; \theta_i)$ to remove regions with extremely low posterior density
- If this leads to a reduction in prior volume, re-simulate from **Step 1** with truncated prior
- Repeat until converged across all parameters, then obtain resulting posterior

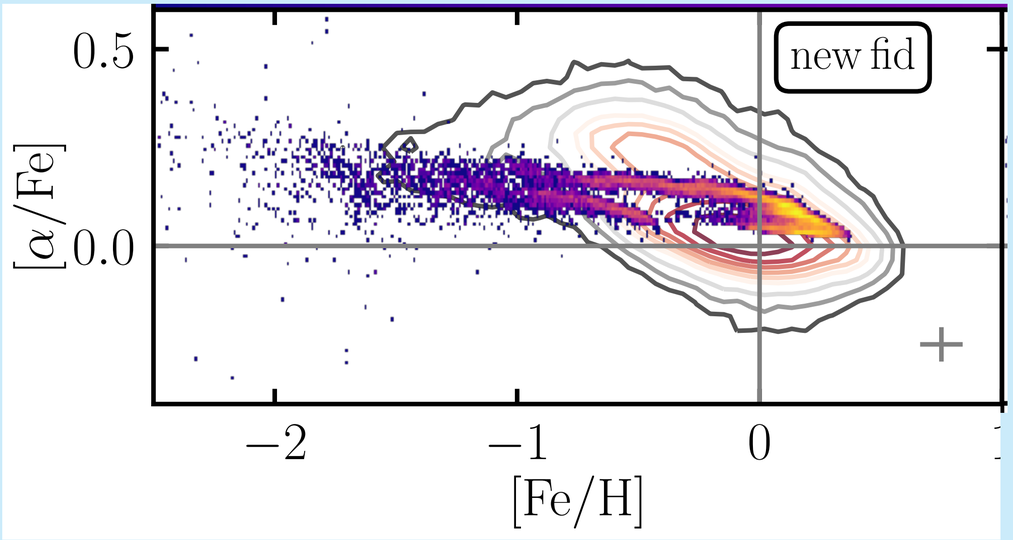


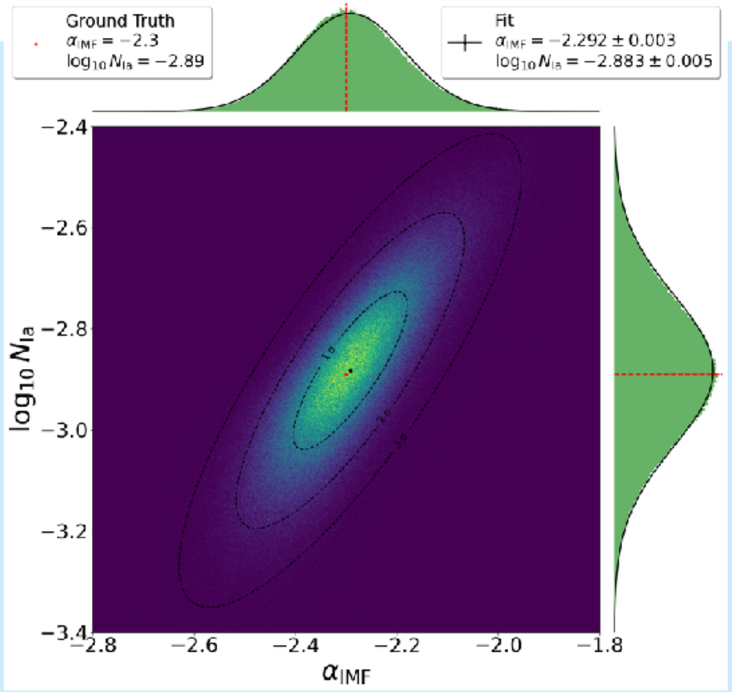
Observation, x_0

Alvey+2024

SBI: APPLICATION FOR GALACTIC CHEMICAL ENRICHMENT



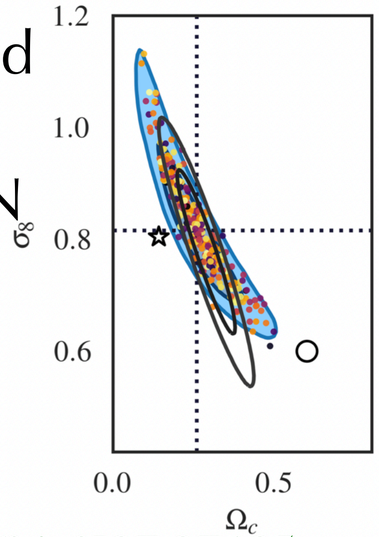




Günes+Buck subm.

SBI: APPLICATION IN COSMOLOGY

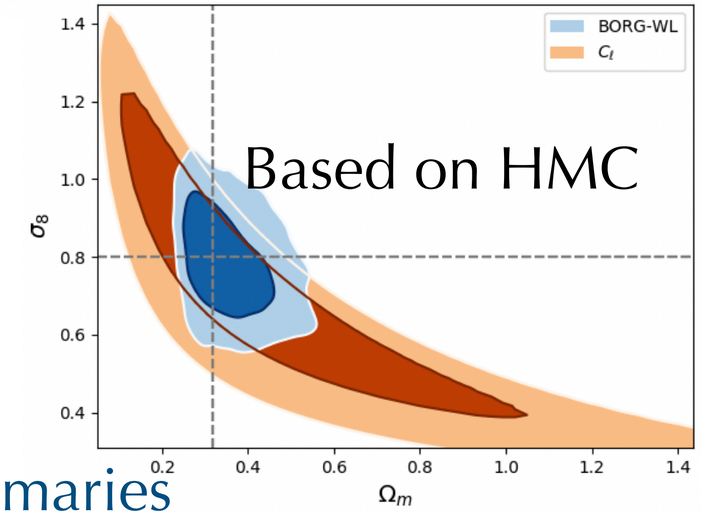
Based on CNN



Makinen+ 2107.07405

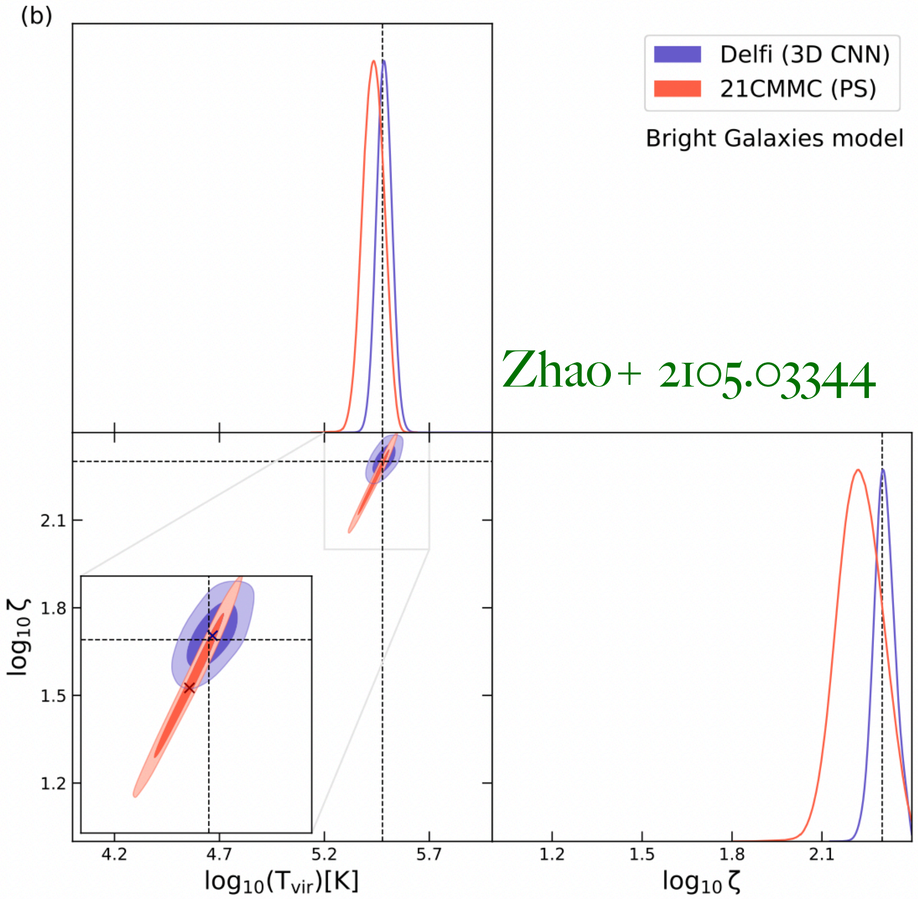
Breaking degeneracy between DM density and power-spectrum amplitude

Porqueres+ 2108.04825



Alternative to: Hand-crafted summaries

slide from Cole



Zhao+ 2105.03344

Breaking degeneracy between ionisation parameters T_{vir} and ζ

SBI: APPLICATION IN STRONG LENSING

Searching light DM halos

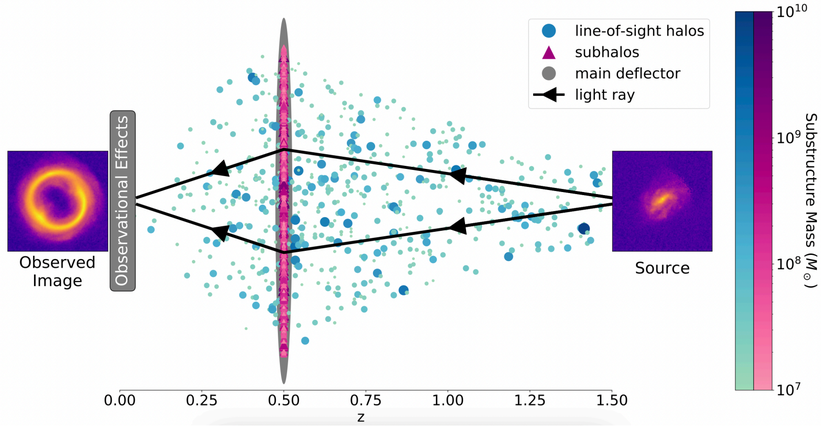
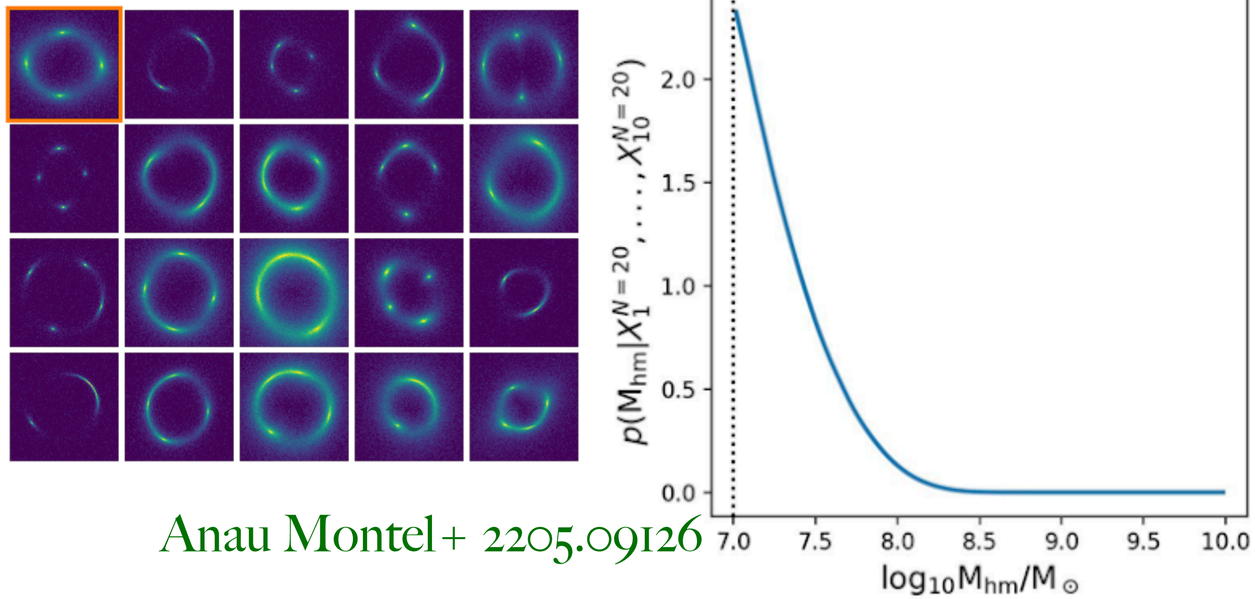


Image credit: Wagner-Carena+ 2203.00690

Halo mass function cutoff

Probing **population effects of light dark matter halos** rather than individual detections



Anau Montel+ 2205.09126

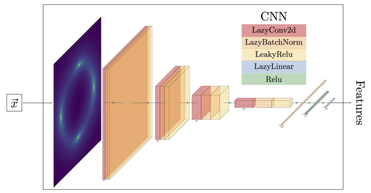
Alternative to:
HMC, parameter reduction, ABC, …

Related work: He+ 2010.13221 (similar in spirit, using ABC)

Wagner-Carena+ 2203.00690 (constraining subhalo mass function normalization)

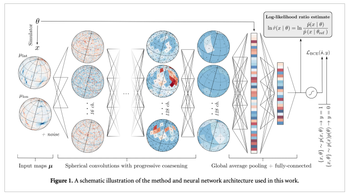
slide from Cole

SBI: APPLICATION IN STRONG LENSING



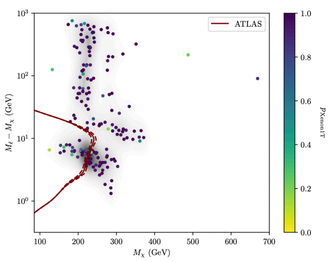
Strong lensing

Brehmer+ 1909.02005, Coogan+ 2010.07032, Legin+ 2112.05278, Wagner-Carena+ 2203.00690, Anau Montel+ 2205.09126, Coogan+ 2207.xxxxx



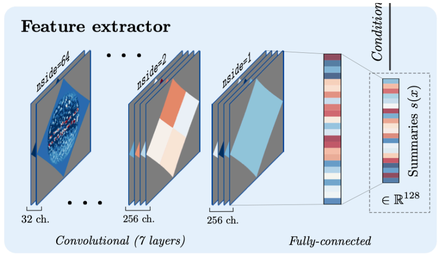
Astrometry

Mishra-Sharma+ 2110.01620



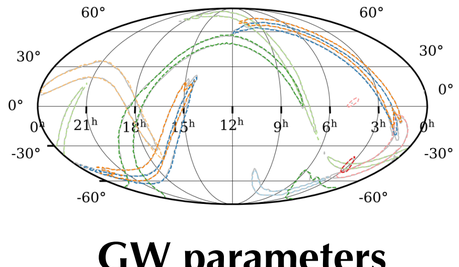
Effective field theory

Morrison+ 2203.13403



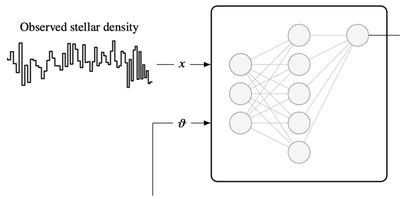
Fermi GeV excess

Mishra-Sharma+ 2110.06931



GW parameters

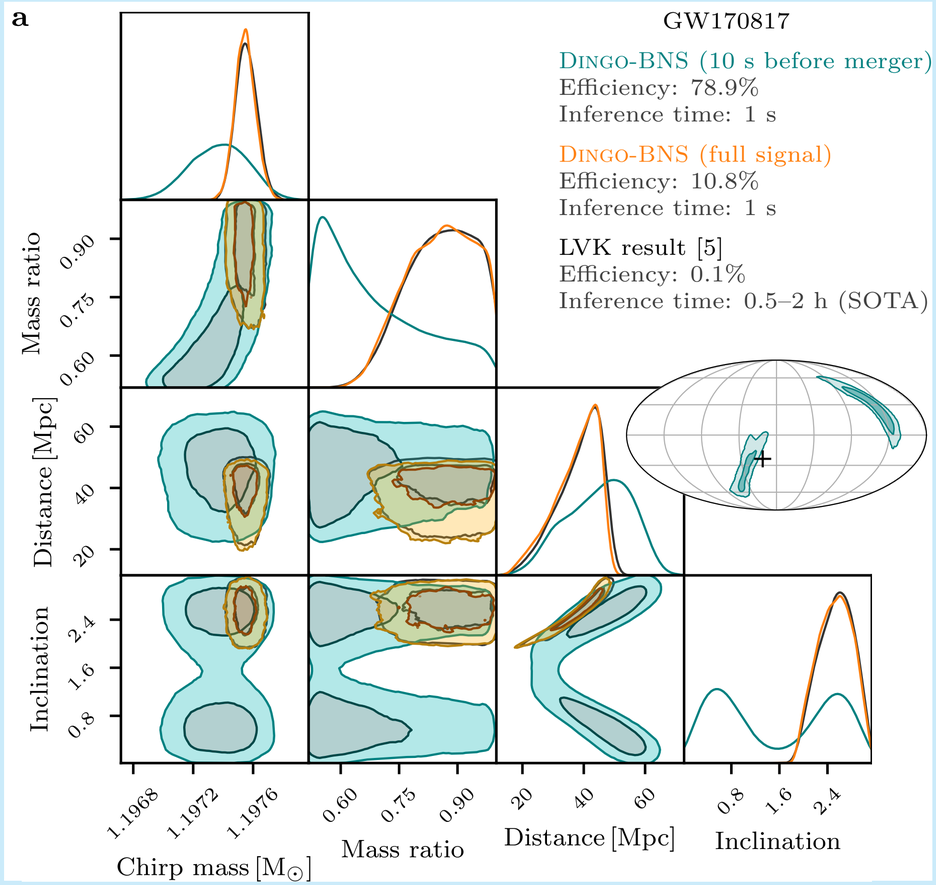
Delaunoy+ 2010.12931, Dax+ 2106.12594, ...



Stellar streams

Hermans+ 2011.14923

Gravitational wave parameter estimation

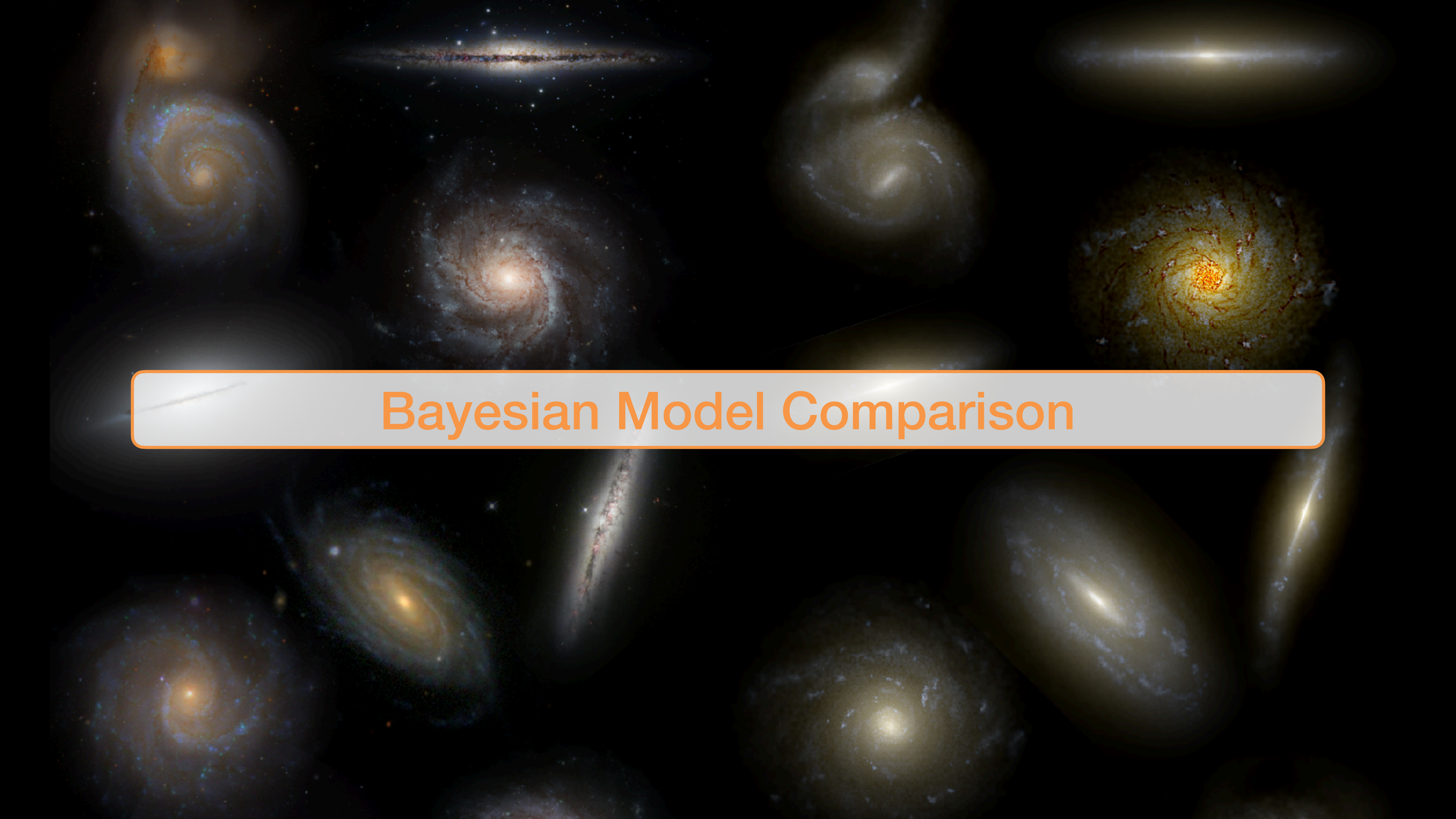


Dax+2021,2023,2024

Single frequency CMB B-mode inference with realistic foregrounds from a single training image

Niall Jeffrey,[1,2]★ François Boulanger,[1] Benjamin D. Wandelt,[3,4] Bruno Regaldo-Saint Blancard,[1,5] Erwan Allys,[1] François Levrier[1]

slide from Cole



Bayesian Model Comparison

MODEL COMPARISON & MODEL MISSPECIFICATION

Bayesian model selection assigns posterior probabilities $\mathrm{p}(\mathcal{M}_k \,|\, \mathbf{d})$ to models $\mathcal{M}_k \in \{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$ (instead of to values of their parameters $\boldsymbol{\theta}_k$), conditional on observed data $\mathbf{d}$. The conventional approach is to compute the marginal likelihood (or *evidence*) $\mathrm{p}(\mathbf{d} \,|\, \mathcal{M}_k)$, which is the average likelihood $\mathrm{p}(\mathbf{d} \,|\, \boldsymbol{\theta}_k)$ of parameters distributed according to the prior $\mathrm{p}(\boldsymbol{\theta}_k)$:

$$\mathrm{p}(\mathbf{d} \,|\, \mathcal{M}_k) = \int \mathrm{p}(\mathbf{d} \,|\, \boldsymbol{\theta}_k)\, \mathrm{p}(\boldsymbol{\theta}_k)\, \mathrm{d}\boldsymbol{\theta}_k \tag{1}$$

(where the presence of $\mathcal{M}_k$'s parameters $\boldsymbol{\theta}_k$ implies conditioning on $\mathcal{M}_k$ in the right-hand side). The prior belief in the model, $\mathrm{p}(\mathcal{M}_k)$, is then updated to its posterior probability in accordance with Bayes' theorem: $\mathrm{p}(\mathcal{M}_k \,|\, \mathbf{d}) \propto \mathrm{p}(\mathbf{d} \,|\, \mathcal{M}_k)\, \mathrm{p}(\mathcal{M}_k)$, normalised over all models considered.

MODEL COMPARISON & MODEL MISSPECIFICATION

Methods

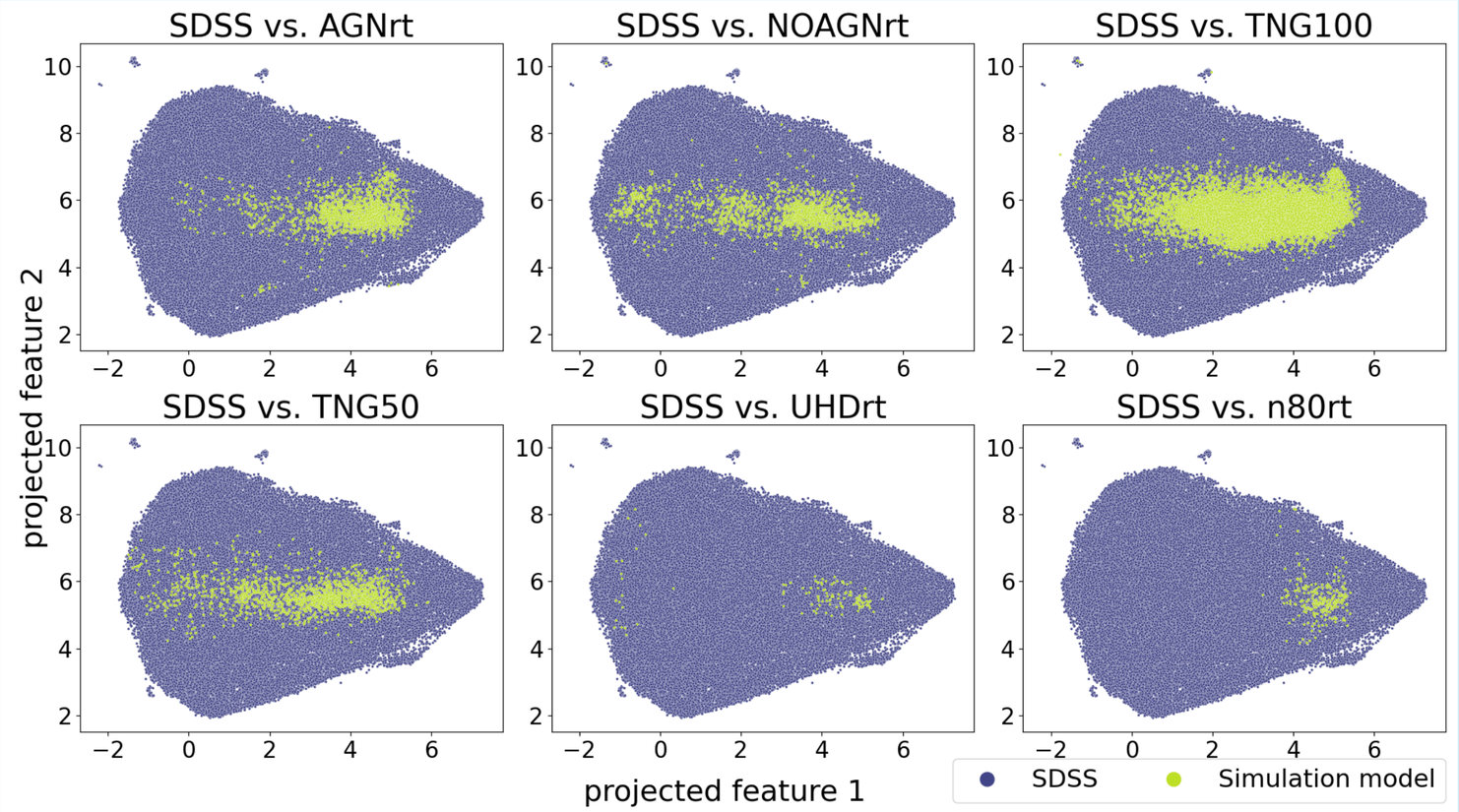
- Jeffrey+Wandelt 2023: loss functions for two-way model comparison with an emphasis on recovering accurate extreme Bayes factors
- Radev+2021: estimate a Dirichlet distribution over an arbitrary number of models using a NN and variational optimisation
- Elsemüller+2023 and Karchev+2024: use a cross-entropy loss for multi-class posterior probabilities
- Macciò+2022: Model selection for star formation prescriptions in cosmo sims
- Zhou+2024: Model misspecification plus model comparison for low simulation budget applications
- Jin+2024: Model comparison of cosmo sims via GANomaly scores

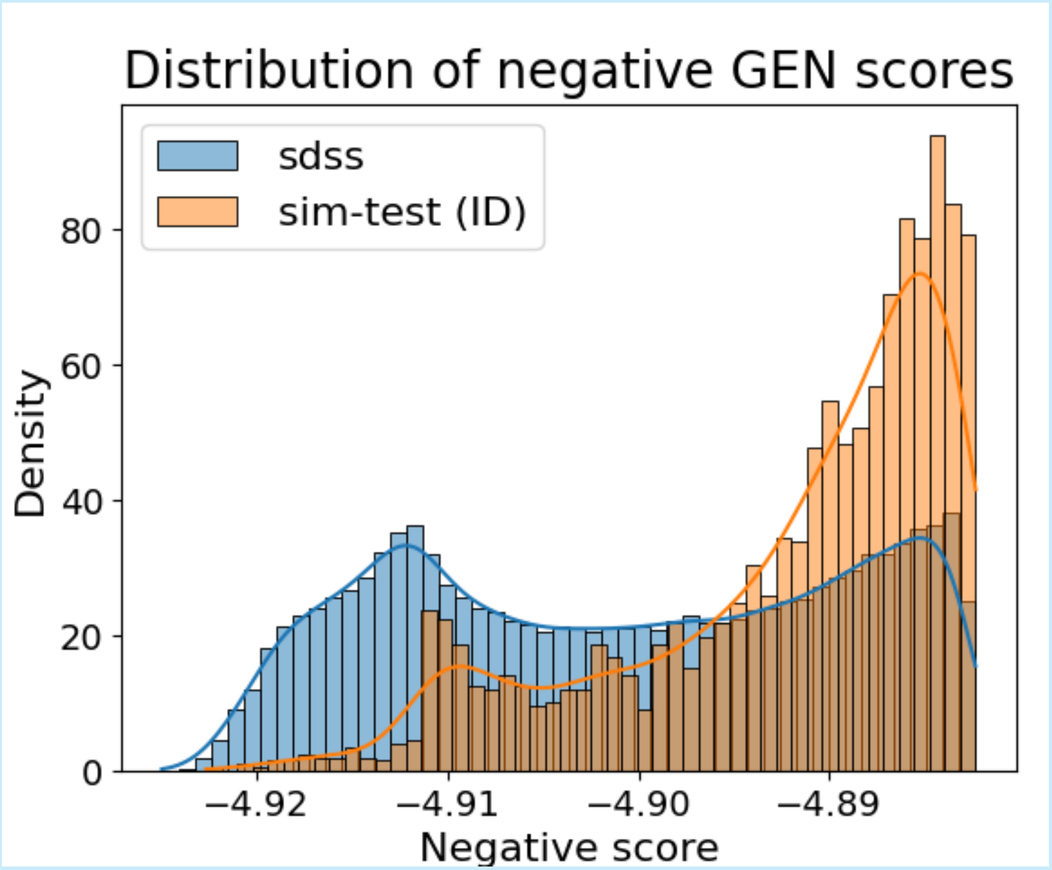
MODEL COMPARISON & MODEL MISSPECIFICATION

Idea:

- Train embedding network on ~600.000 SDSS images, then encode simulated SDSS images
- Train simulation classifiers on embeddings, apply to real SDSS images

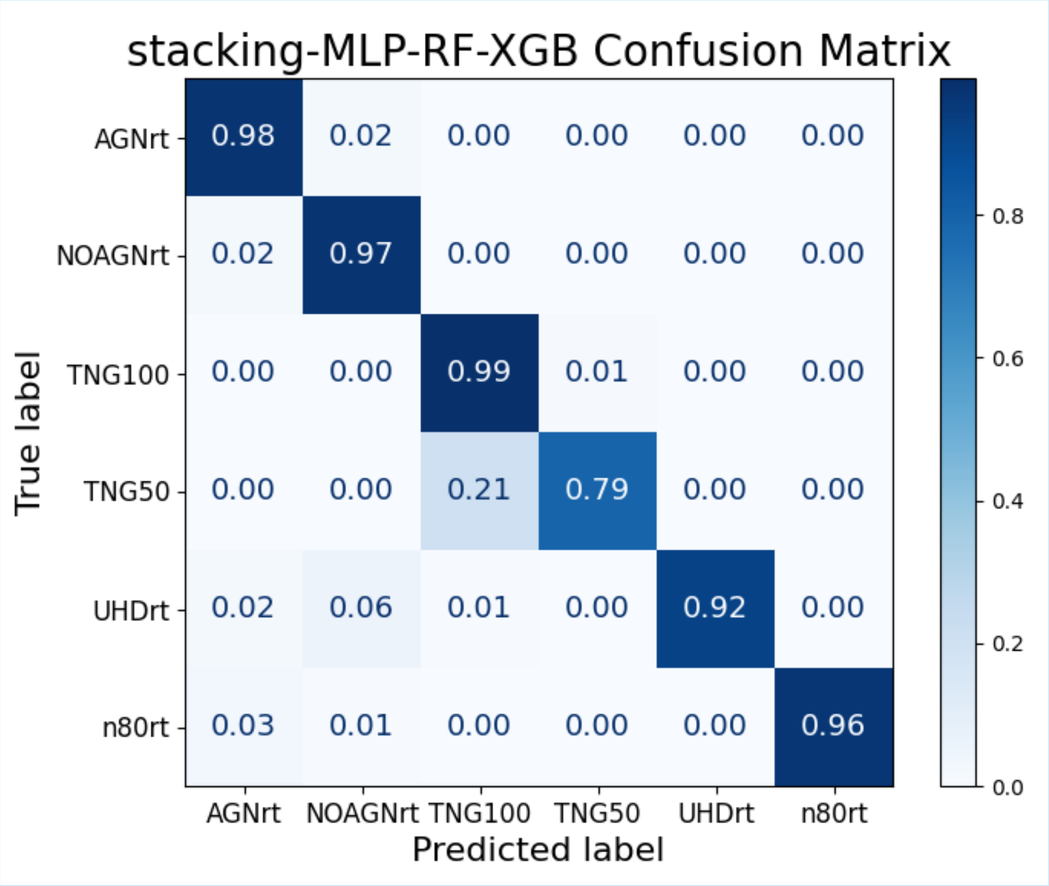
UMAP projection of k-sparse encoding of sims and Jobs

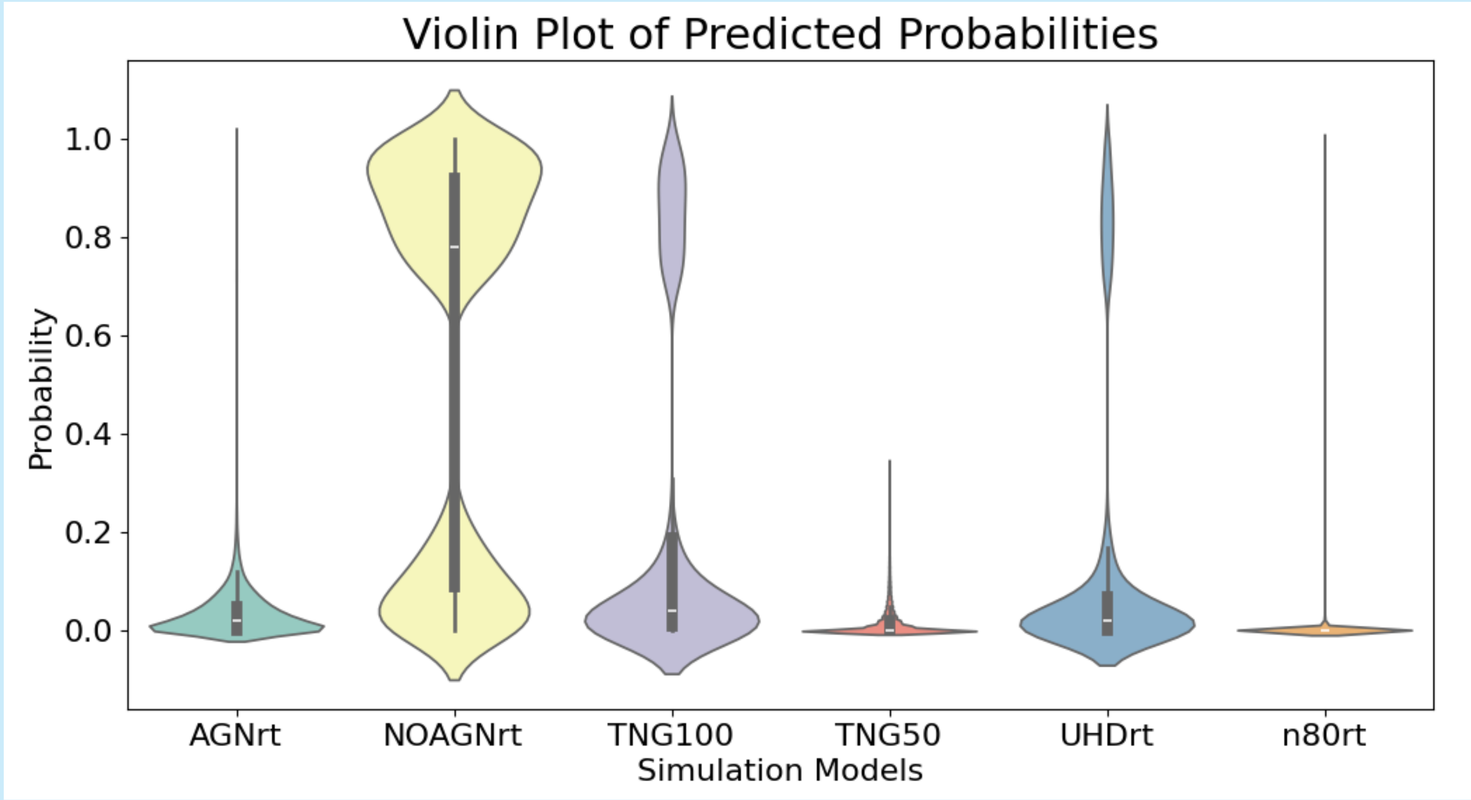




Zhou,Buck+2024 subm.

MODEL COMPARISON & MODEL MISSPECIFICATION



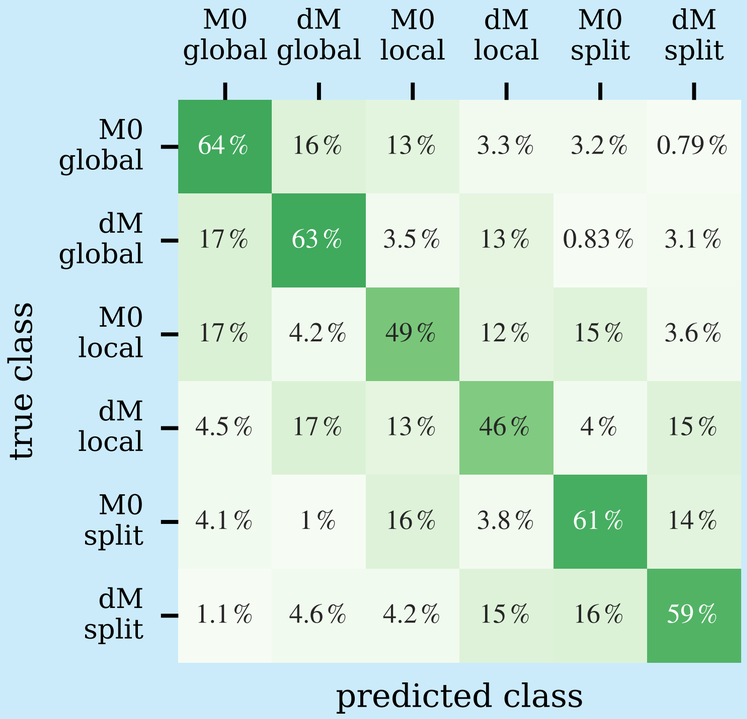


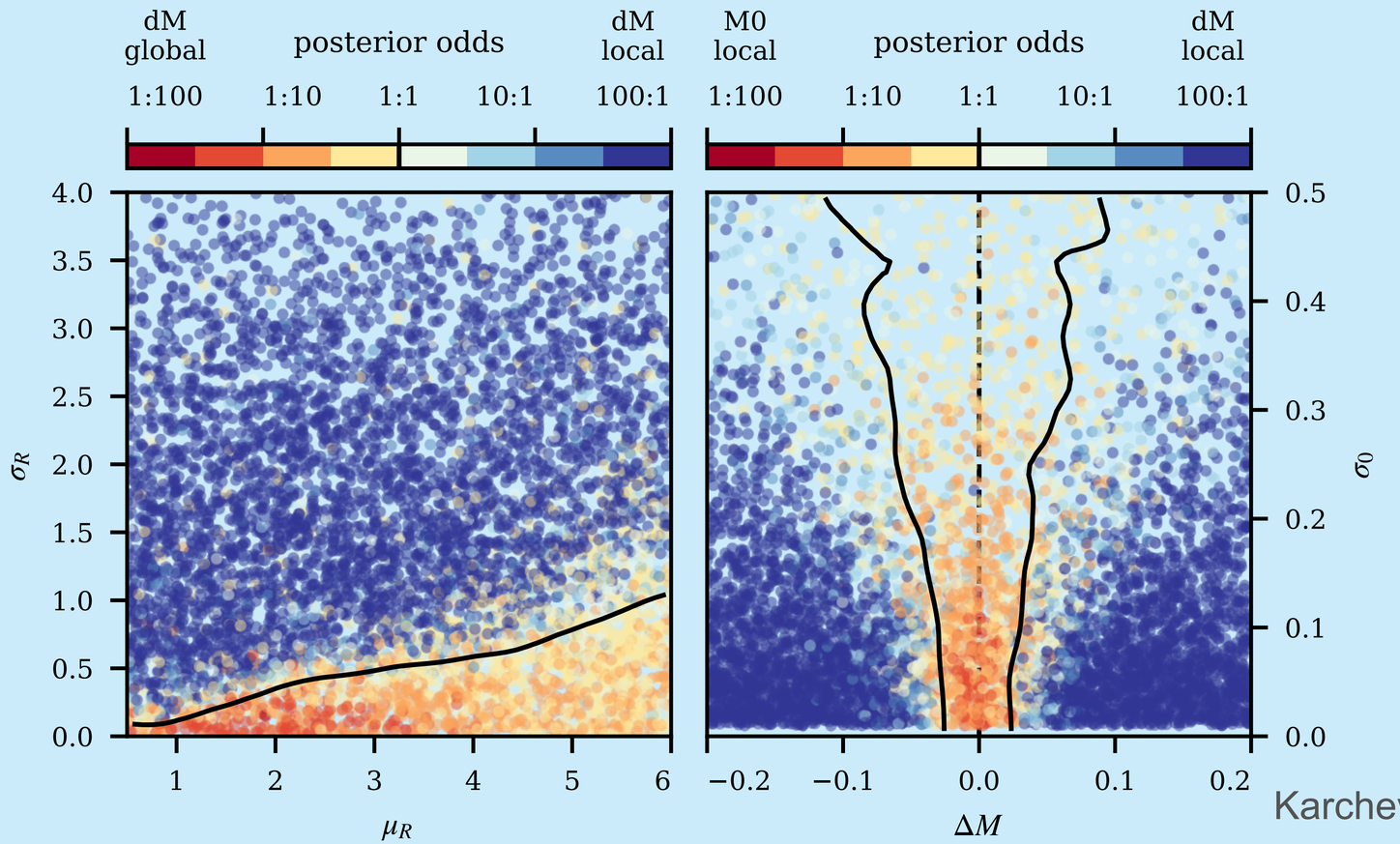
Zhou,Buck+2024 subm.

MODEL COMPARISON/SELECTION

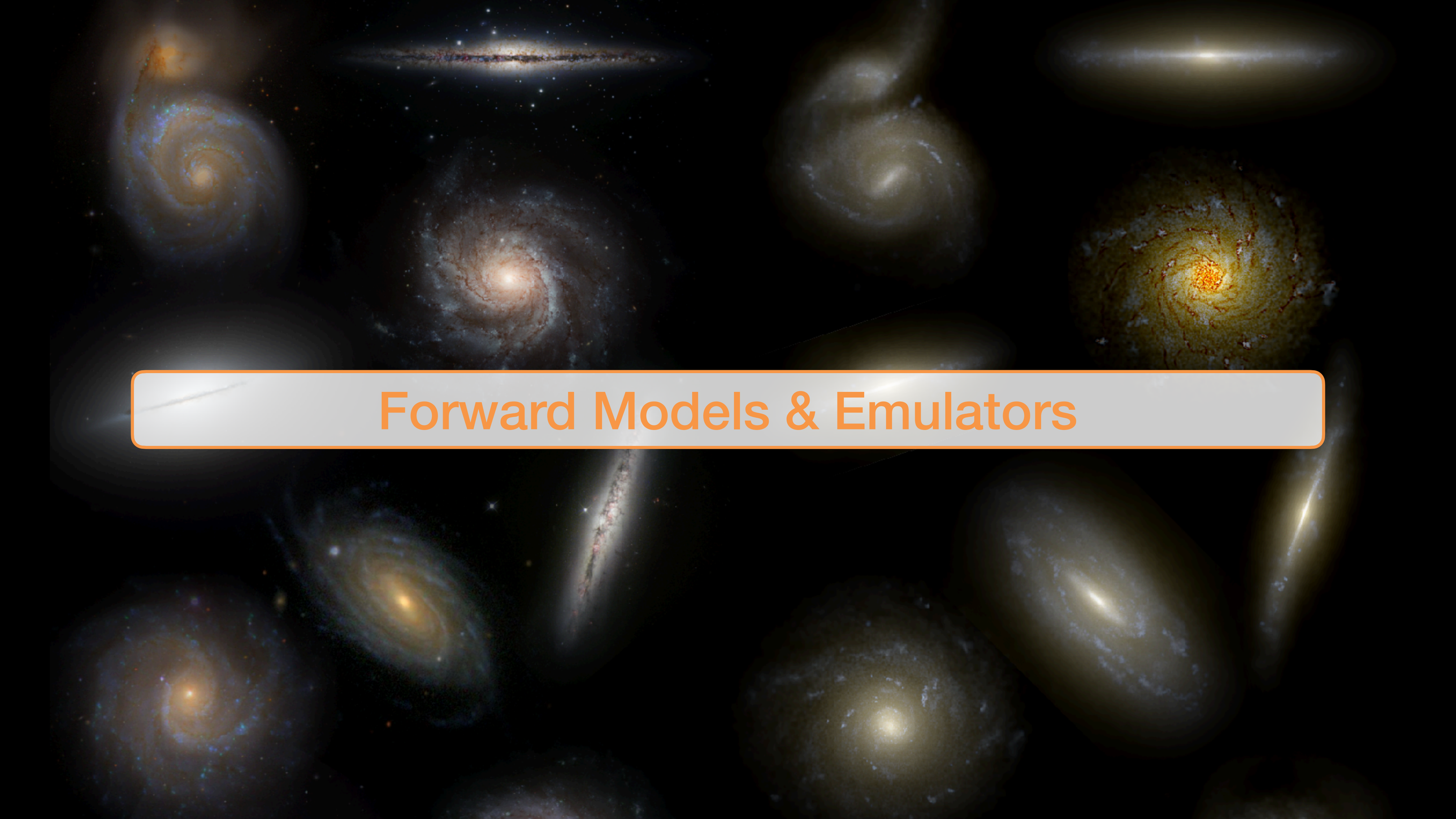
SimSIMS: Simulation-based Supernova Ia Model Selection with thousands of latent variables

Konstantin Karchev[1] **Roberto Trotta**[1,2,3,4] **Christoph Weniger**[5]





Karchev+2024



Forward Models & Emulators

LEARNING THE SOLUTION OF ODES AND PDES — NEURAL ODES, OPERATOR LEARNING AND PINNS

- ODEs are good for:
  - population models
  - motion of the planets
  - structural integrity of a bridge
  - fluid dynamics
  - …

ODEs are kind of easy — only derivatives with respect to one variable

PDEs are more complicated — derivatives with respect to many variables and differential equations are local while solutions exhibit non-local properties

- Traditional solution: discretisation (in time and space) and iterative solution

NEURAL ODES, OPERATOR LEARNING AND PINNS

Neural ODE: $\frac{df}{dt} = h_\theta(x_0, t, p)$

(neural net = right hand side of diff eq.

solution: integrate entire neural net.)

Neural Operator: $G_\theta : X \to Y \quad u \mapsto G_\theta(u)$ with X, Y function spaces (infinite dimensional)

(neural net approximates the operator
i.e. the map between function space)

PINN: $f(x,t) = h_\theta(x,t,p)$ with $\frac{df}{dt} = \frac{dh_\theta}{dt}, \quad \frac{df}{dx} = \frac{dh_\theta}{dx}$ need to fulfil the diff eq.

(solution is given by neural net,
autodiff and diff eq. are used in loss)

NEURAL ODES

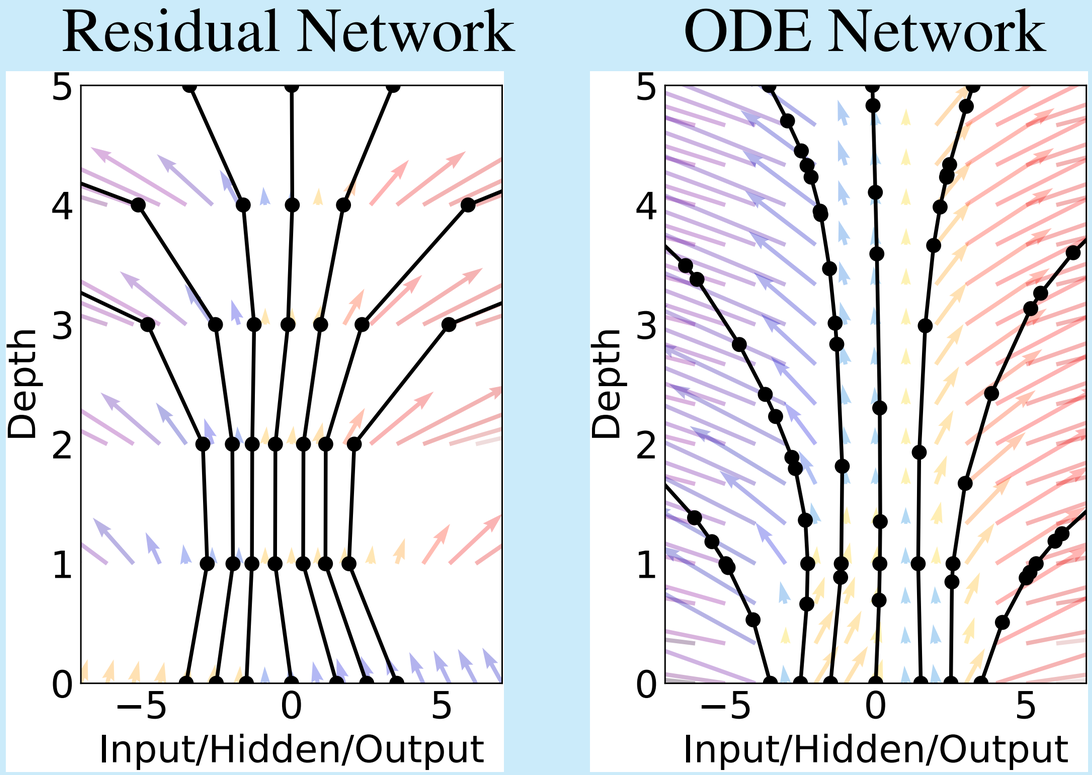
- Traditional solution: discretisation (in time and space) and iterative solution

Euler discretization

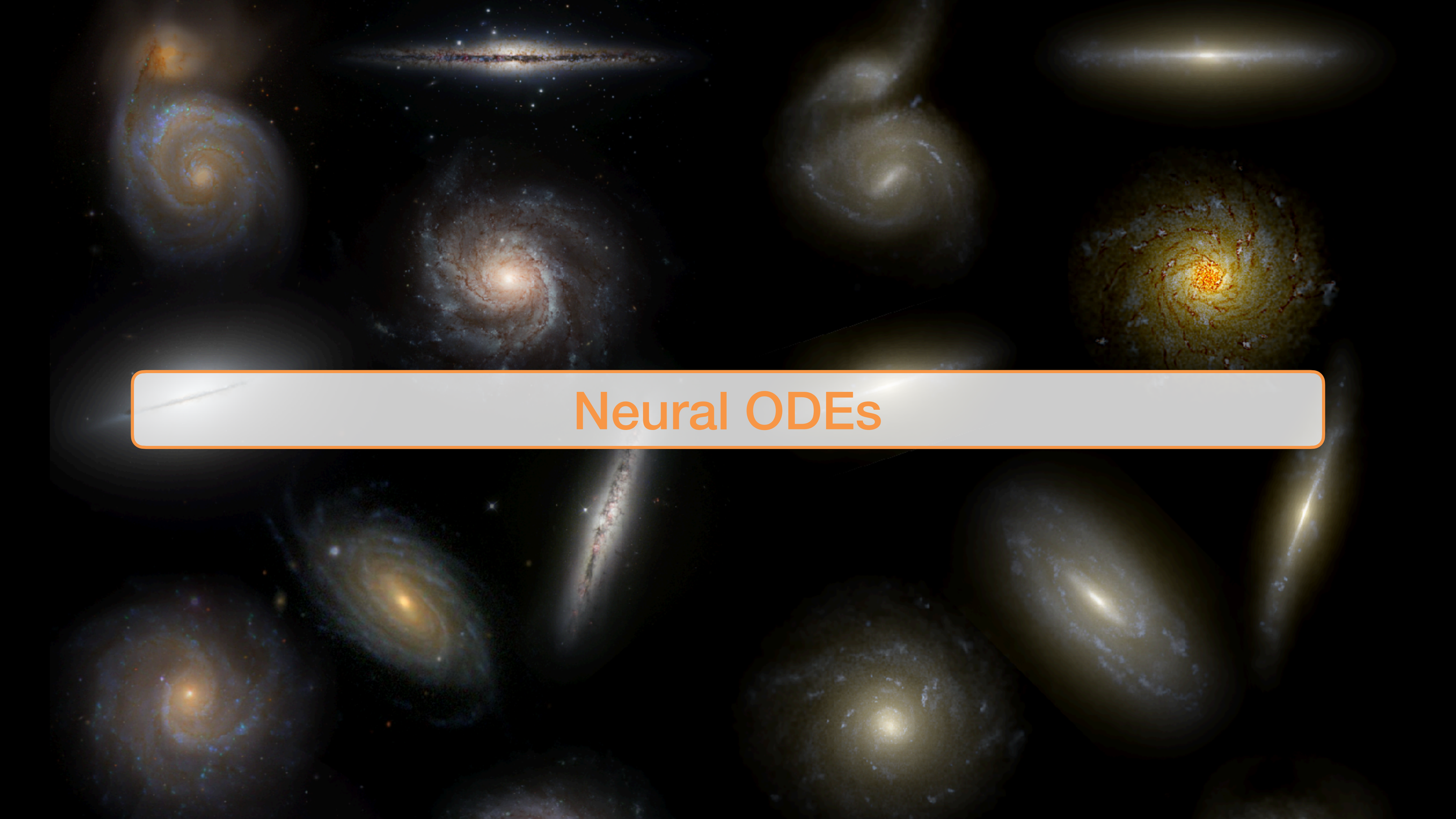
$$h_{t+1} = h_t + f_t(h_t, \theta_t)$$

$$h(t + \Delta t) = h(t) + \Delta t \cdot f(t, h(t), \theta)$$

$$\frac{h(t + \Delta t) - h(t)}{\Delta t} = f(t, h(t), \theta)$$



Chen+2019


Neural ODEs

NEURAL ODES

What do we need to train a NeuralODE? → Reverse-mode automatic differentiation of ODE solutions

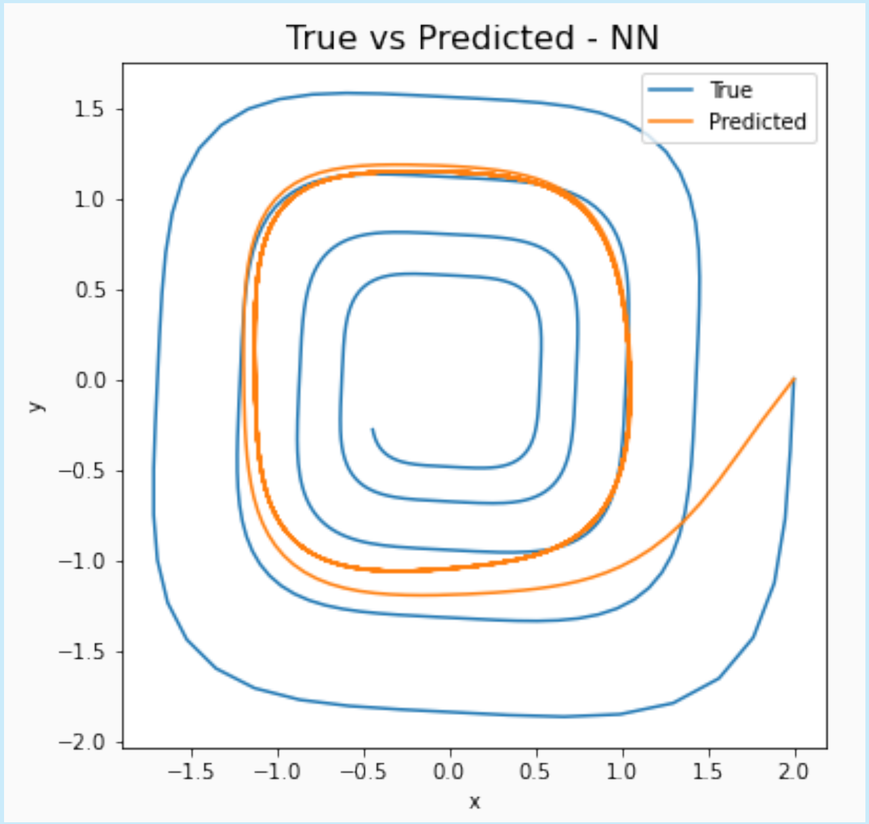
adjoint sensitivity method (Pontryagin et al., 1962)

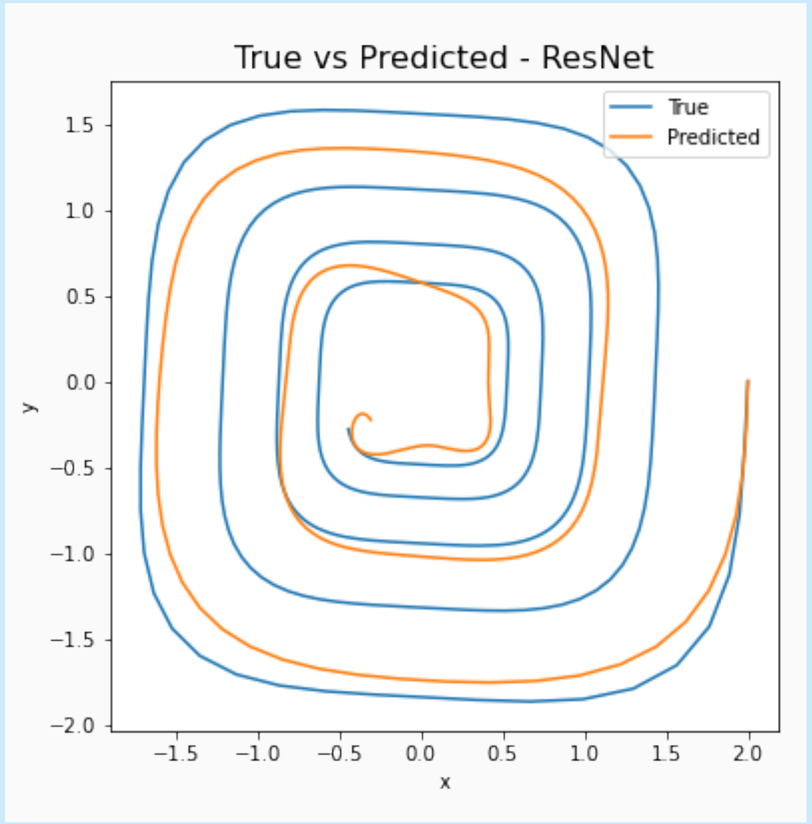
$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right)$$

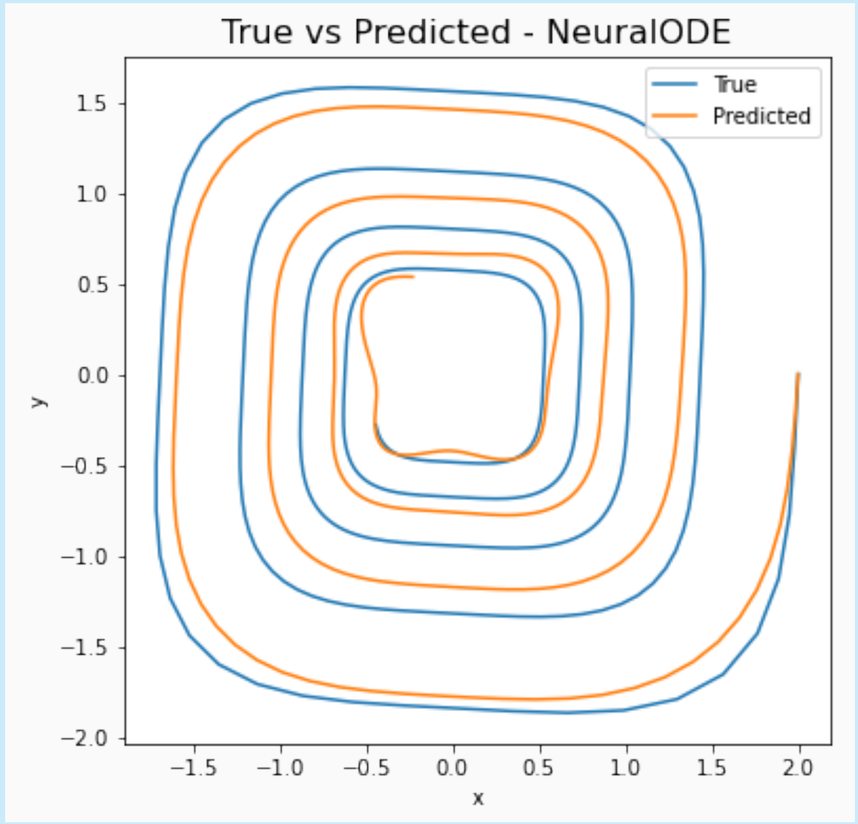
Don’t worry about the details… diffrax (Patrick Kidger), torchdiffeq (Ricky Chen), DiffEqFlux.jl got you convert!

NEURAL ODES

Simple application of NeuralODEs to a supervised learning task

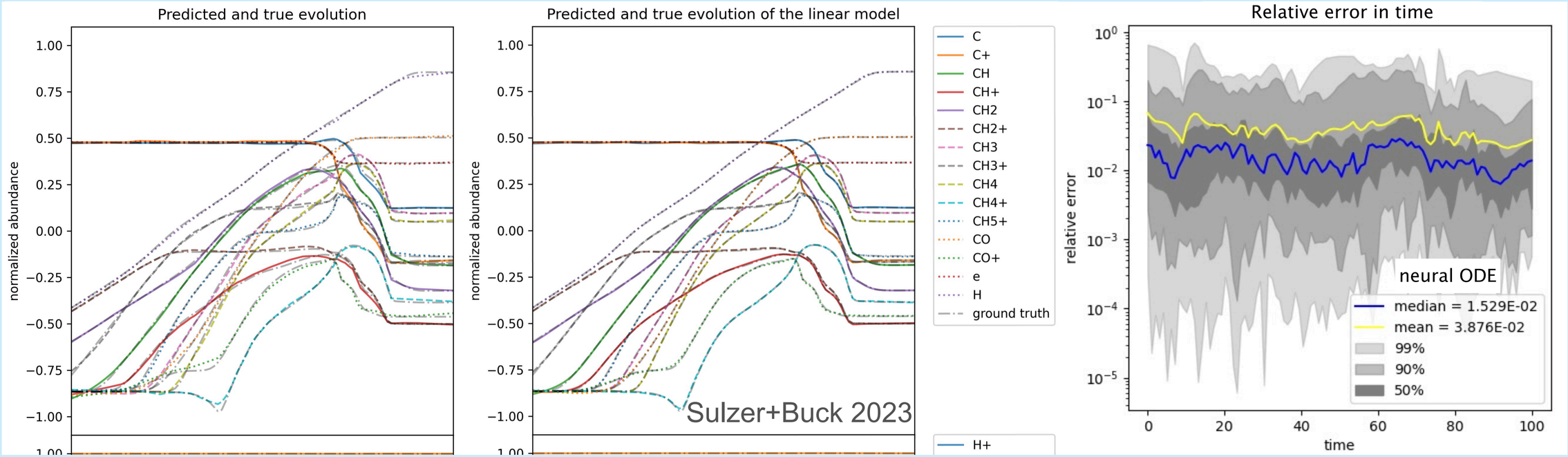


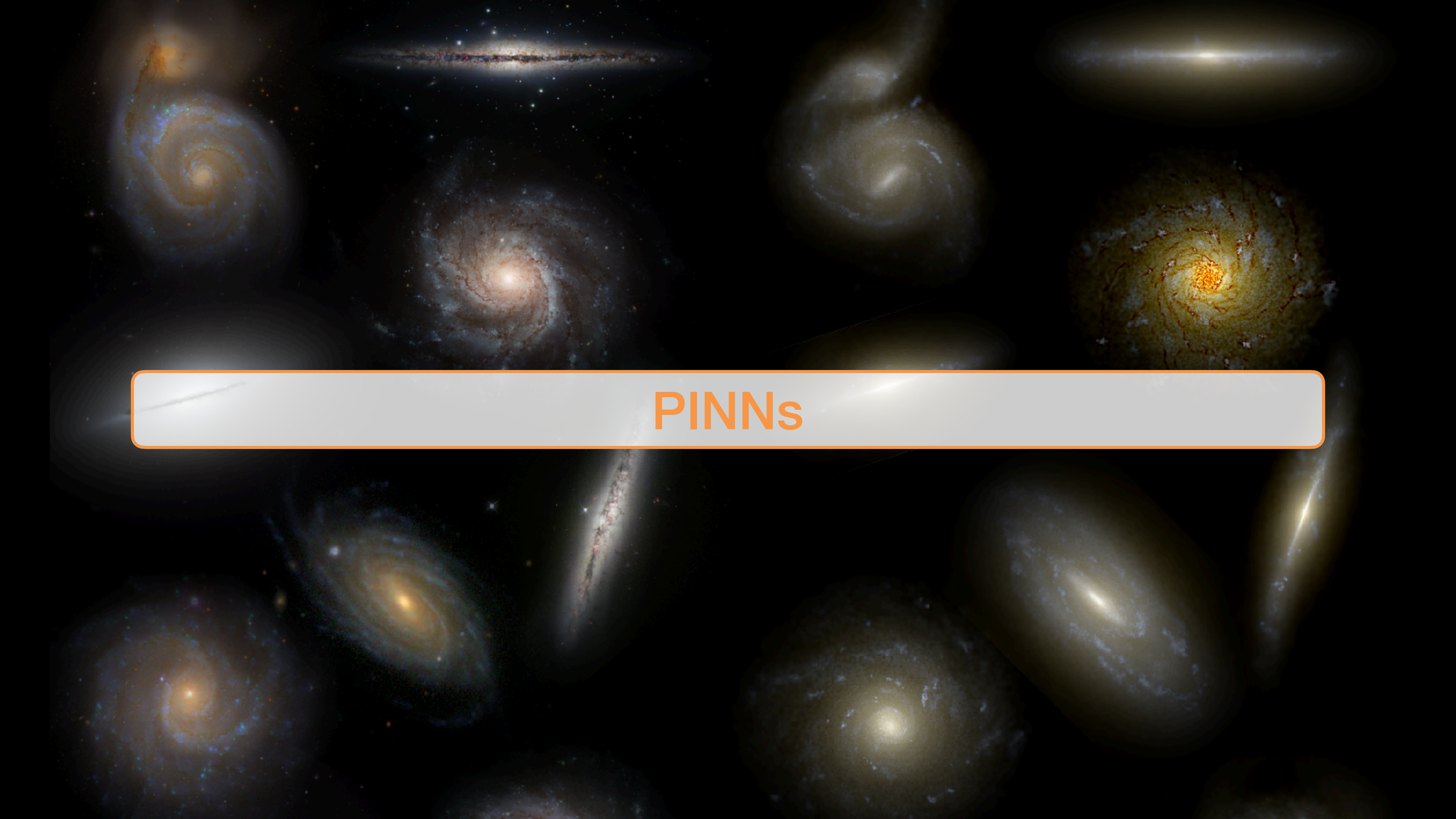




NEURAL ODES IN ASTROPHYSICS

- Neural Astrophysical Wind Models (Nguyen 2023)
- Neural ODEs as a discovery tool to characterize the structure of the hot galactic wind of M82 (Nguyen+2023)
- Speeding up astrochemical reaction networks with autoencoders and neural ODEs (Sulzer+Buck 2023)




PINNs

PHYSICS INFORMED NEURAL NETS

Loss function for PINNs

Differential Equation: $\mathcal{F}[u(x,y)] = f(x,y)$

Dataset: $(x_i, y_i, u_i);\ i = 1, \dots, N_{data}$

Collocation points: $(x_j, y_j); j = 1, \dots, N_C$

Initial Condition: (x_o, y_0, u_0)

$$L_{DiffEq} = \frac{1}{N_c}\sum_{j=1}^{N_c}\left(\mathcal{F}[u(x_j, y_j)] - f(x_j, y_j)\right)^2$$

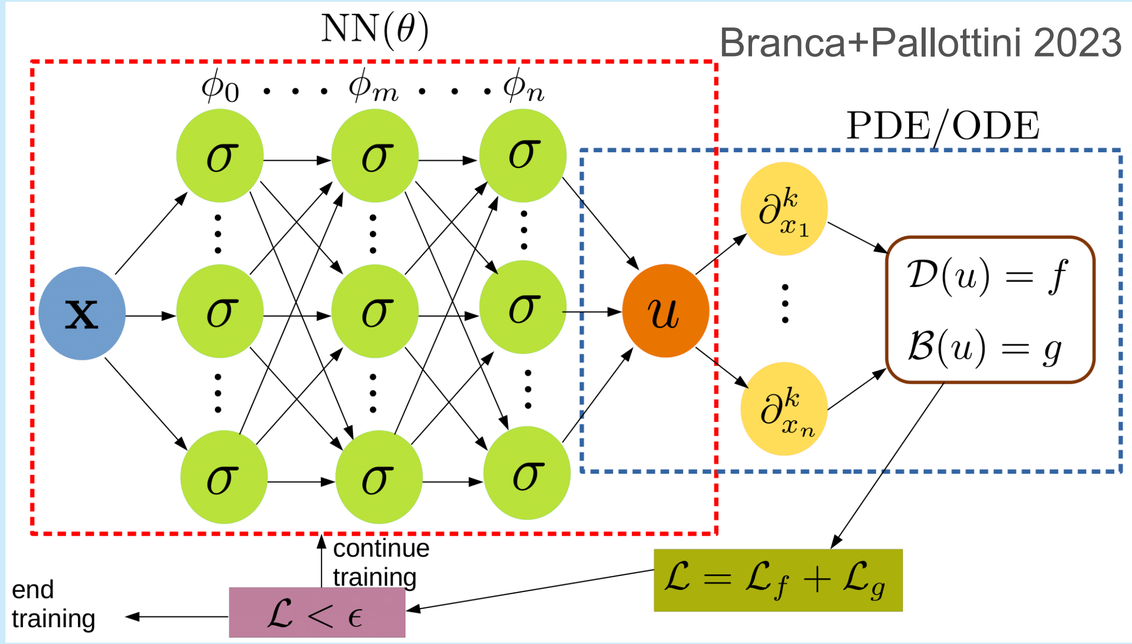
$$L_{total} = \omega_{data} \cdot L_{data} + \omega_{DiffEq} \cdot L_{DiffEq} + \omega_{IC} \cdot L_{IC}$$

$$L_{data} = \frac{1}{N_{data}}\sum_{i=1}^{N_{data}}(u(x_i, y_i) - u_i)^2$$

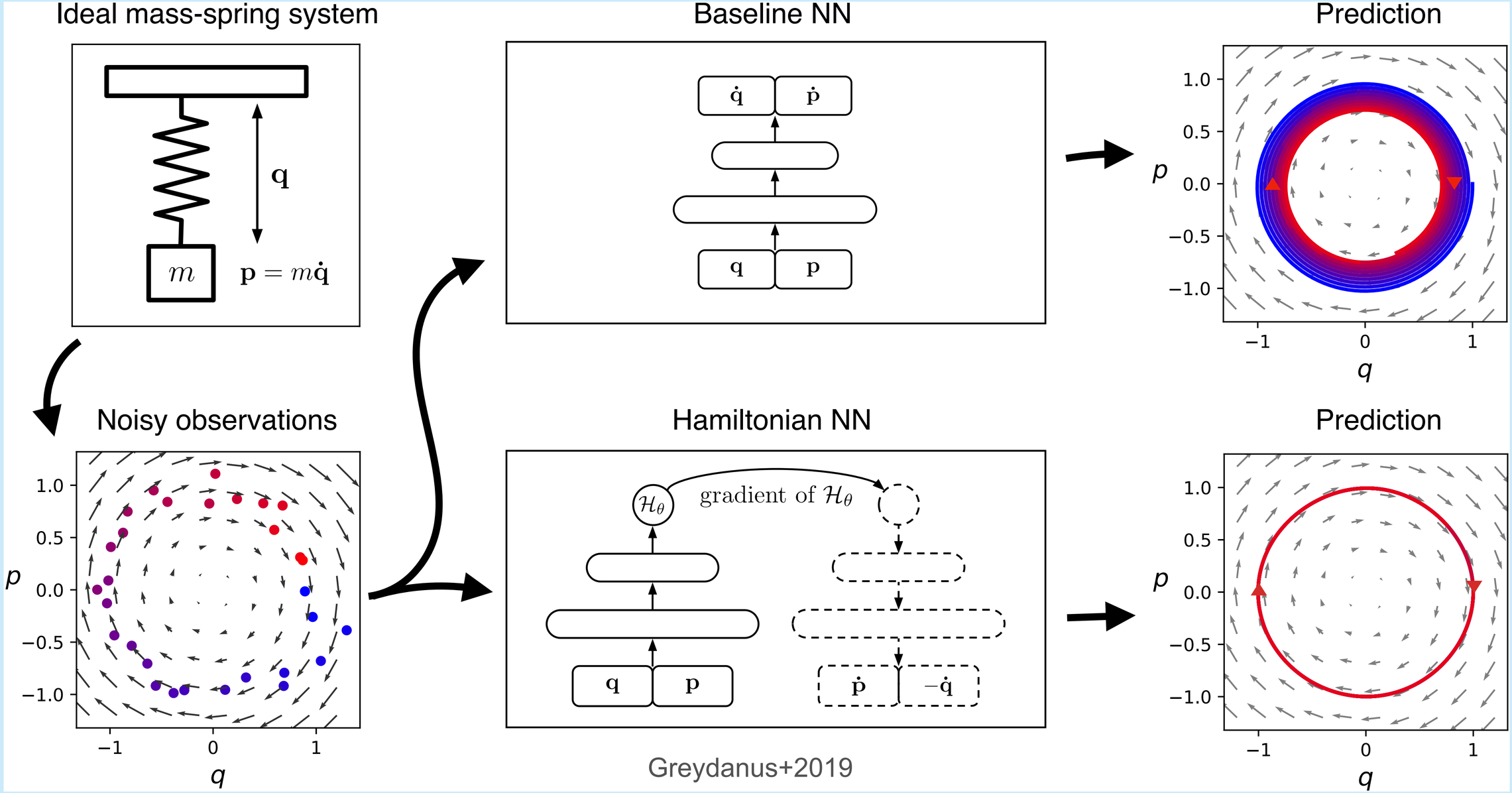
$$L_{IC} = (u(x_0, y_0) - u_0)^2$$

PINNS IN ASTROPHYSICS

- Physics-informed neural networks for modeling astrophysical shocks (Moschou+2023)
- Probing the solar coronal magnetic field with physics-informed neural networks (Jarolim+2022)
- Physics-informed neural networks in the recreation of hydrodynamic simulations from dark matter (Dai+2023)
- Physics informed neural networks for simulating radiative transfer (Mishra+Molinaro 2021)
- Neural networks: solving the chemistry of the interstellar medium (Branca+Pallottini 2023)



HAMILTONIAN NEURAL NETWORKS




Operator Learning

UNIVERSAL APPROXIMATION THEOREM FOR NONLINEAR OPERATORS

UAT – Operators

Chen & Chen, Universal Approximation to Nonlinear Operators..., 1995

Let $\mathrm{G}: \mathrm{V} \to \mathrm{W}, \mathrm{u} \mapsto \mathrm{G}(\mathrm{u})$ be a non-linear continuous Operator.
$G(u)(y)$ denotes the value of $G(u)$ at point y.

There is a function $f_{c,\xi,\theta,\zeta}(\boldsymbol{x}, y)$ such that for every $\epsilon > 0$

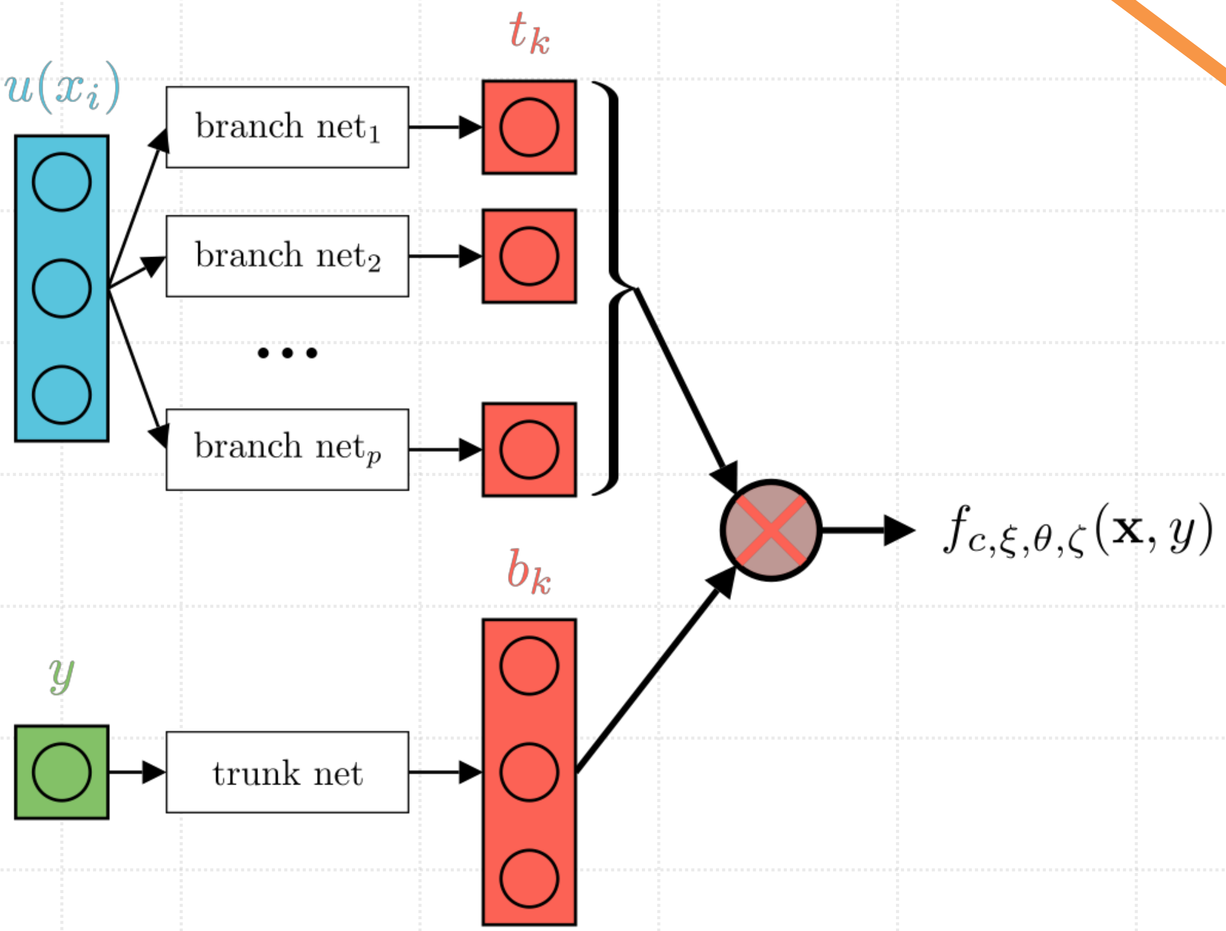
$$\left|G(u)(y) - f_{c,\xi,\theta,\zeta}(\boldsymbol{x}, y)\right| < \epsilon$$

for all $u \in V, y \in Y$ $(G(u): Y \to Z)$.

$$f_{c,\xi,\theta,\zeta}(\boldsymbol{x}, y) = \sum_{k=1}^{p} \sigma(w_k \cdot y + \zeta_k) \sum_{i=1}^{n} c_i^k \cdot \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k \cdot u(x_j) + \theta_i^k\right)$$

DEEP OPERATOR ARCHITECTURE

$$f_{c,\xi,\theta,\zeta}(\boldsymbol{x}, y) = \sum_{k=1}^{p} \sigma(w_k \cdot y + \zeta_k) \sum_{i=1}^{n} c_i^k \cdot \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k \cdot u(x_j) + \theta_i^k\right)$$



Theory – Approximation Theorem:

1. Branch Net:

$$t_k = \sum_{i=1}^{n} c_i^k \cdot \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k \cdot u(x_j) + \theta_i^k\right)$$

2. Trunk Net:

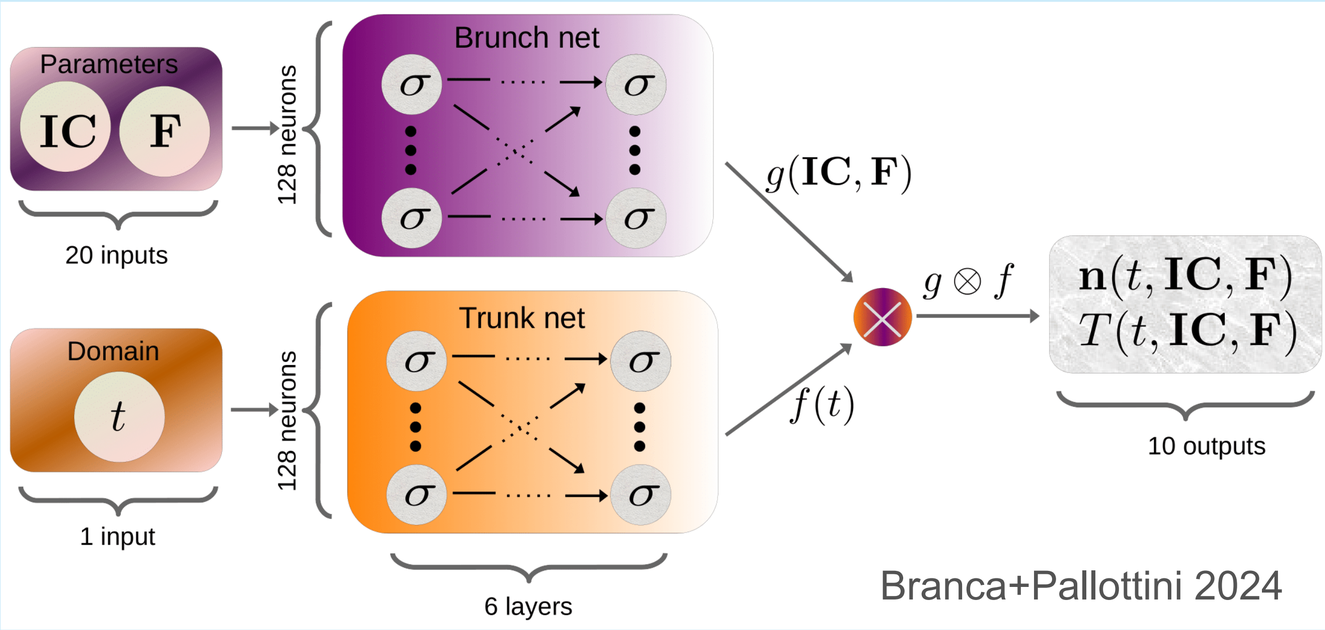
$$b_k = \sigma(w_k \cdot y + \zeta_k)$$

3. Output: $G(u)$

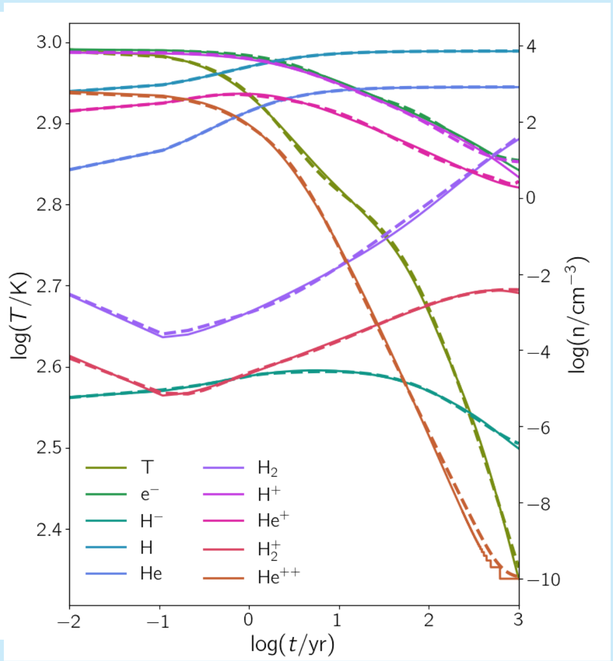
$$f_{c,\xi,\theta,\zeta}(\boldsymbol{x}, y) = \sum_{k=1}^{p} t_k b_k$$

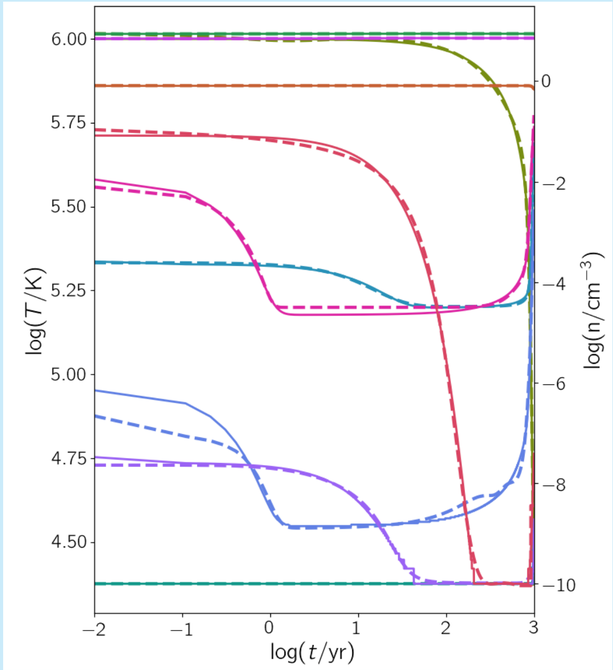
DEEP OPERATORS IN ASTROPHYSICS

- PPDONet: Deep Operator Networks for Fast Prediction of Steady-State Solutions in Disk-Planet Systems (Mao+2023)
- Emulating the interstellar medium chemistry with neural operators (Branca+Pallottini 2024)
- CODES Benchmark for neuralODEs and Operator Learning for astrochemistry (Janssen,Sulzer+Buck 2024)



Branca+Pallottini 2024






Differentiale Programming

DIFFERENTIABLE PROGRAMMING

Why do we want to calculate gradients of compute programs automatically and efficiently?

Numerical Optimisation

Variational Inference

Hamilton Monte Carlo Sampling

Training neural networks

CLASSICAL NUMERICAL GRADIENTS

Finite differences:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Easy to implement!

```
def finite_diff(f, x, h=1e-6):
    return (f(x+h)-f(x)) / h
```

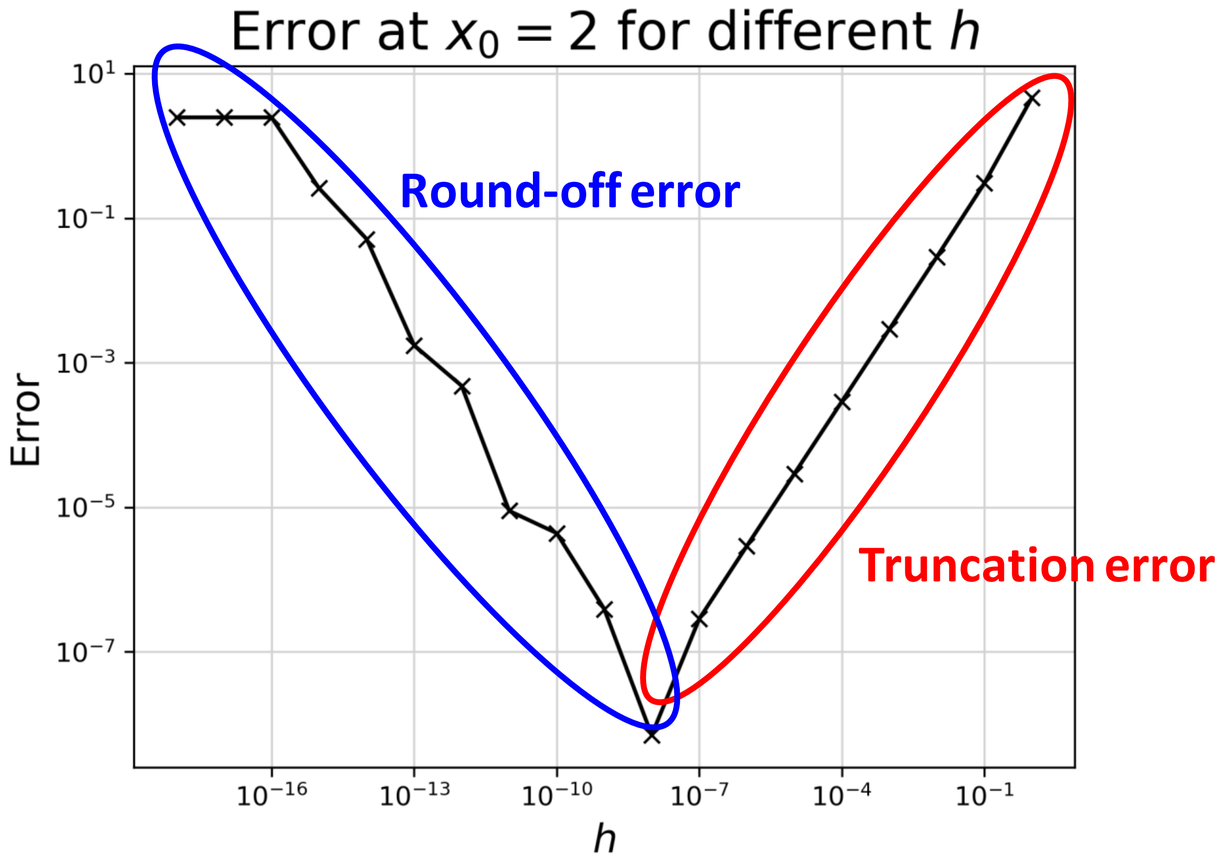
But: imprecise and costly

CLASSICAL NUMERICAL GRADIENTS

- Error approximation

$$Error(h, x_0) = \left| \frac{f(x_0 + h) - f(x_0)}{h} - f'(x_0) \right|$$

- Truncation error
  - $h \to \infty$
- Round-off error
  - Inaccuracy of floating-point numbers of computer



Slow computation and doesn't scale to large dimensions

HOW WOULD YOU DIFFERENTIATE THE FOLLOWING FUNCTION?

$$f(x) = \sin(2x) + x^2$$

SYMBOLIC DIFFERENTIATION

Expression manipulation

Reduction rules

Evaluation of exact gradients

$$\frac{dc}{dx} = 0 \qquad (C \text{ independent of } x)$$

$$\frac{dx}{dx} = 1$$

$$\frac{d(u \circ v)}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx} \qquad \textit{(chain rule)}$$

$$\frac{d(uv)}{dx} = u \cdot \frac{dv}{dx} + v \cdot \frac{du}{dx} \qquad \textit{(product rule)}$$

SYMBOLIC DIFFERENTIATION

$$f(x) = \sin(2x) + x^2$$

Build a computational Graph

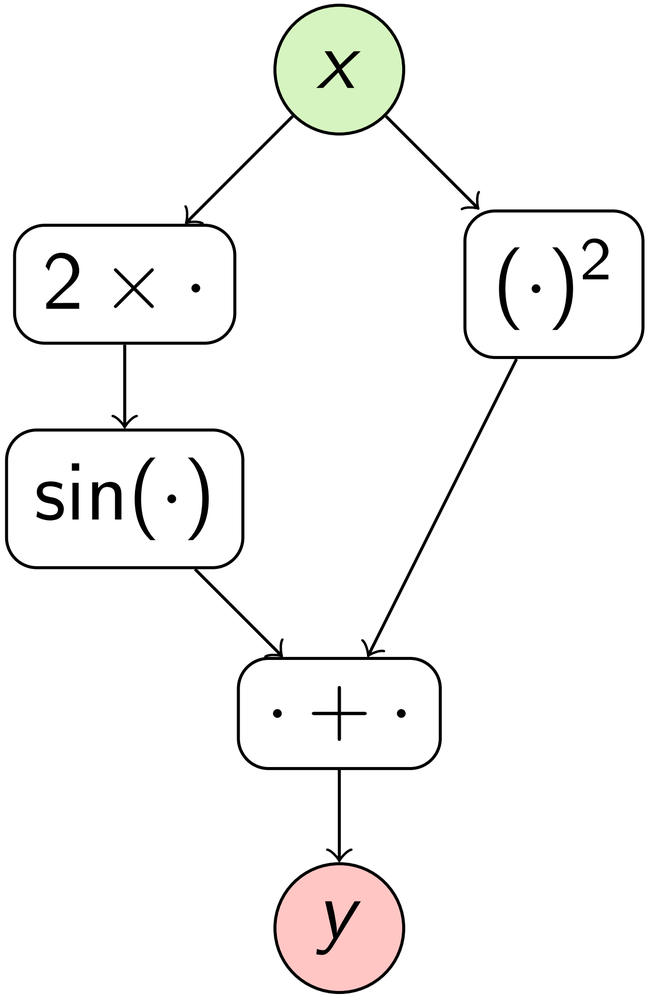
Exact derivatives (down to machine precision)

hard to implement and inflexible (control flow)

memory intensiv

„Expression swell“

SD is implemented e.g. in sympy



AUTOMATIC DIFFERENTIATION

Calculates derivatives *numerically* through *symbolic* differentiation rules

same accuracy as symbolic differentiation, i.e. exact derivatives (down to machine precision)

inflexible for control flows

How does this work? → Computational Graph for $F : \mathbb{R}^n \mapsto \mathbb{R}^m$

$$F = D \circ C \circ B \circ A \qquad y = F(x) = D(C(B(A(x))))$$

$$a = A(x), b = B(a), c = C(b), y = d = D(c)$$

$$F'(x) = \frac{\partial y}{\partial x} = \frac{\partial y}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}$$

see also: lecture by Matthew J. Johnson. http://videolectures.net/deeplearning2017_ johnson_automatic_differentiation/

AUTOMATIC DIFFERENTIATION FORWARD VS. REVERSE MODE

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\underbrace{\left(\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}\right)}_{\frac{\partial b}{\partial x}}$$

Forward mode

$$\frac{\partial y}{\partial x} = \underbrace{\left(\frac{\partial y}{\partial d}\frac{\partial d}{\partial c}\right)}_{\frac{\partial y}{\partial c}}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial x}$$

Reverse mode

Computational graph

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

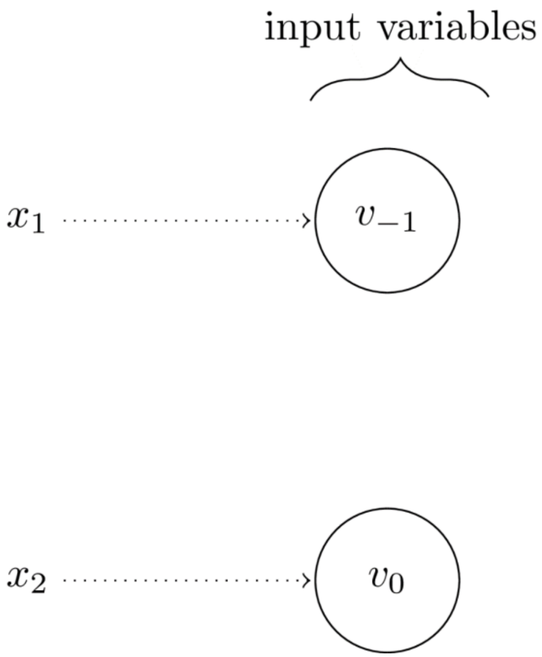
x_1

x_2

$$v_{-1} = x_1$$
$$v_0 = x_2$$

Computational graph

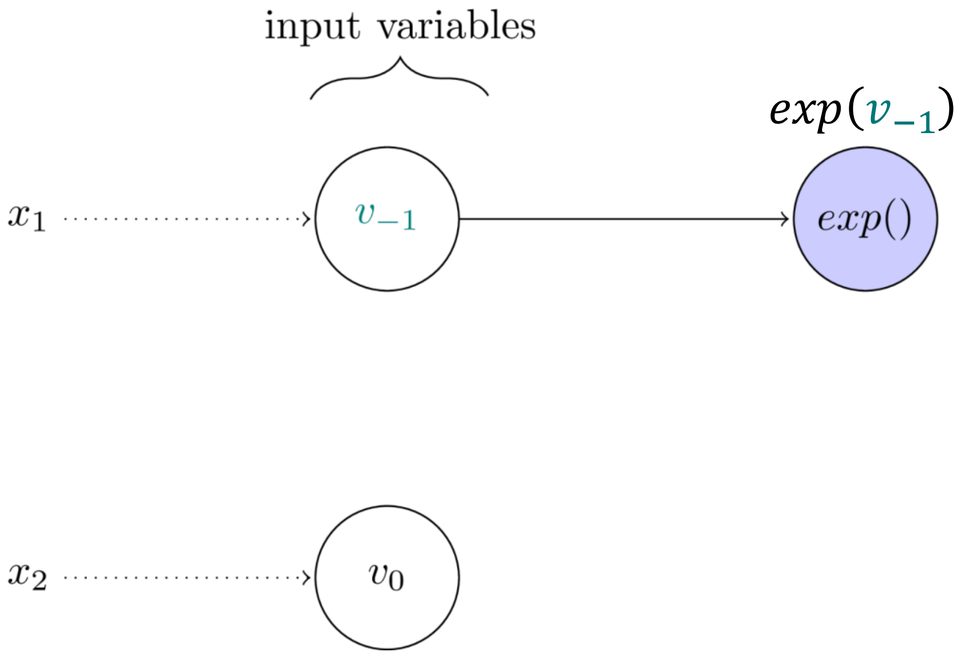
$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$
$$v_0 = x_2$$

Computational graph

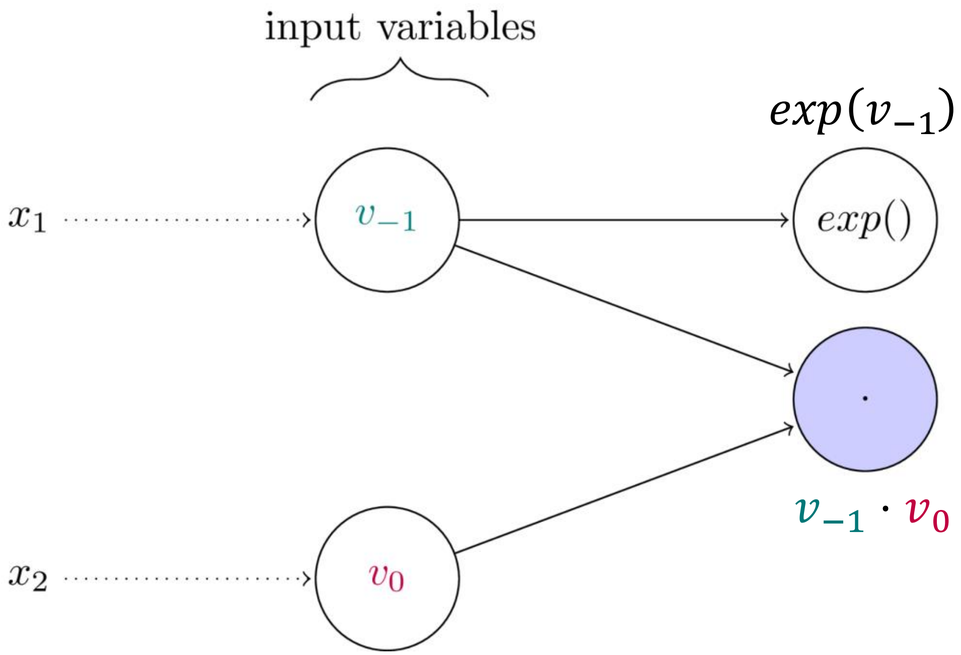
$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$
$$v_0 = x_2$$

Computational graph

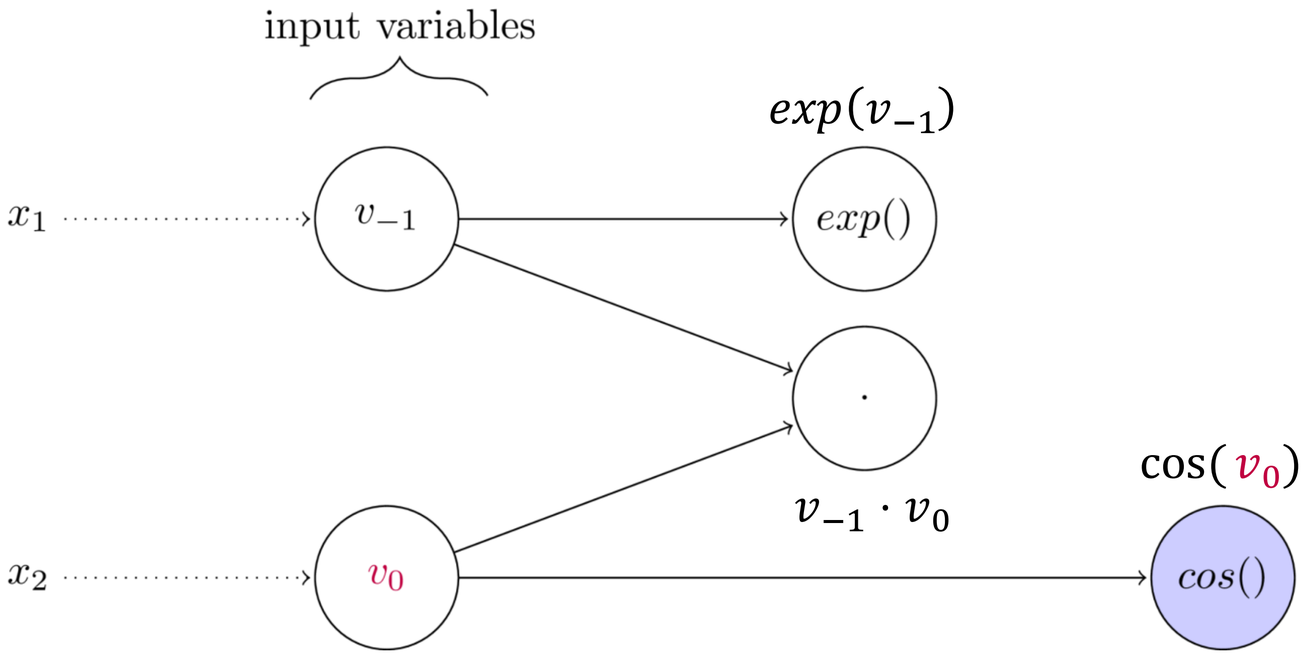
$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$
$$v_0 = x_2$$

Computational graph

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

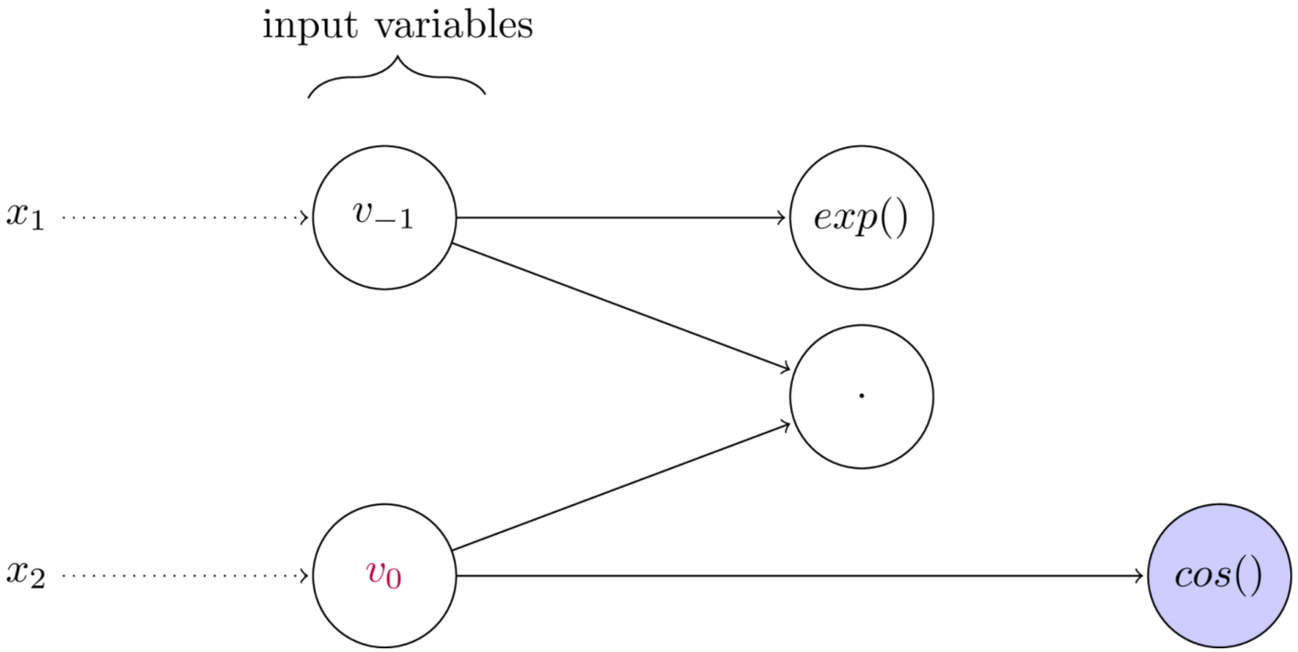


$$v_{-1} = x_1$$
$$v_0 = x_2$$

Computational graph

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

$$exp(v_{-1}) \qquad v_{-1} \cdot v_0 \qquad \cos(v_0)$$



Computational graph

$$v_{-1} = x_1$$

$$v_0 = x_2$$

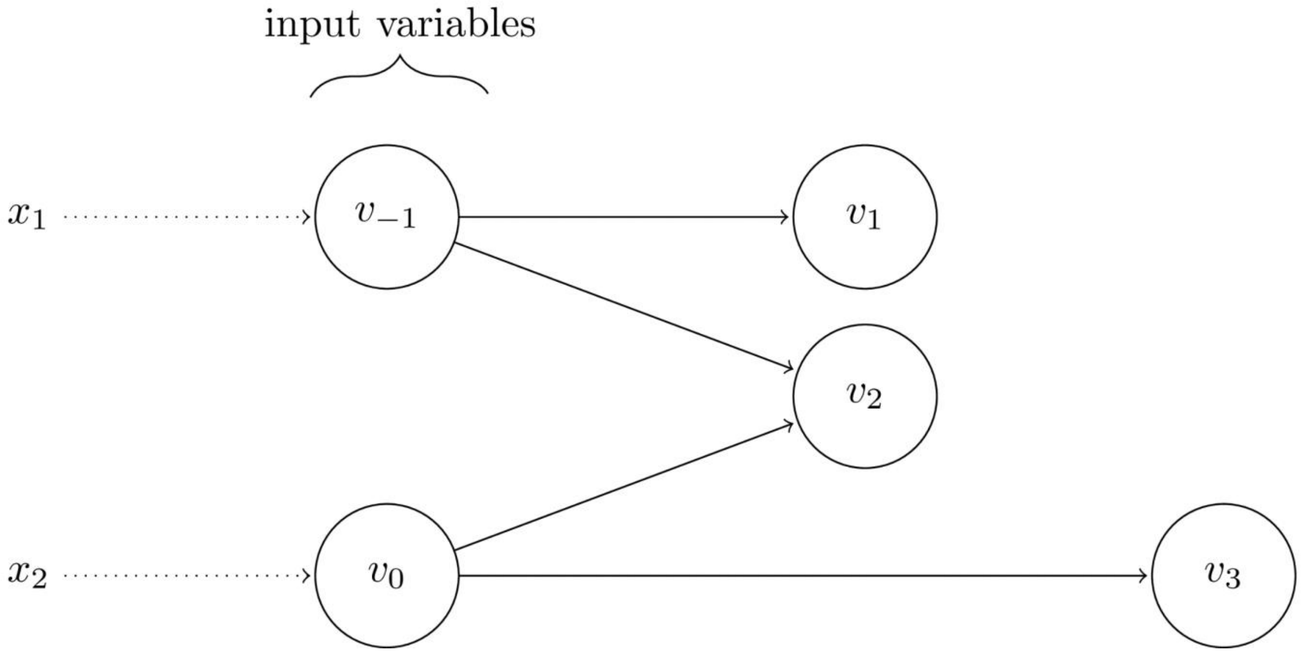
$$v_1 = exp(v_{-1})$$

$$v_2 = v_{-1} \cdot v_0$$

$$v_3 = cos(v_0)$$

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

v_1 v_2 v_3

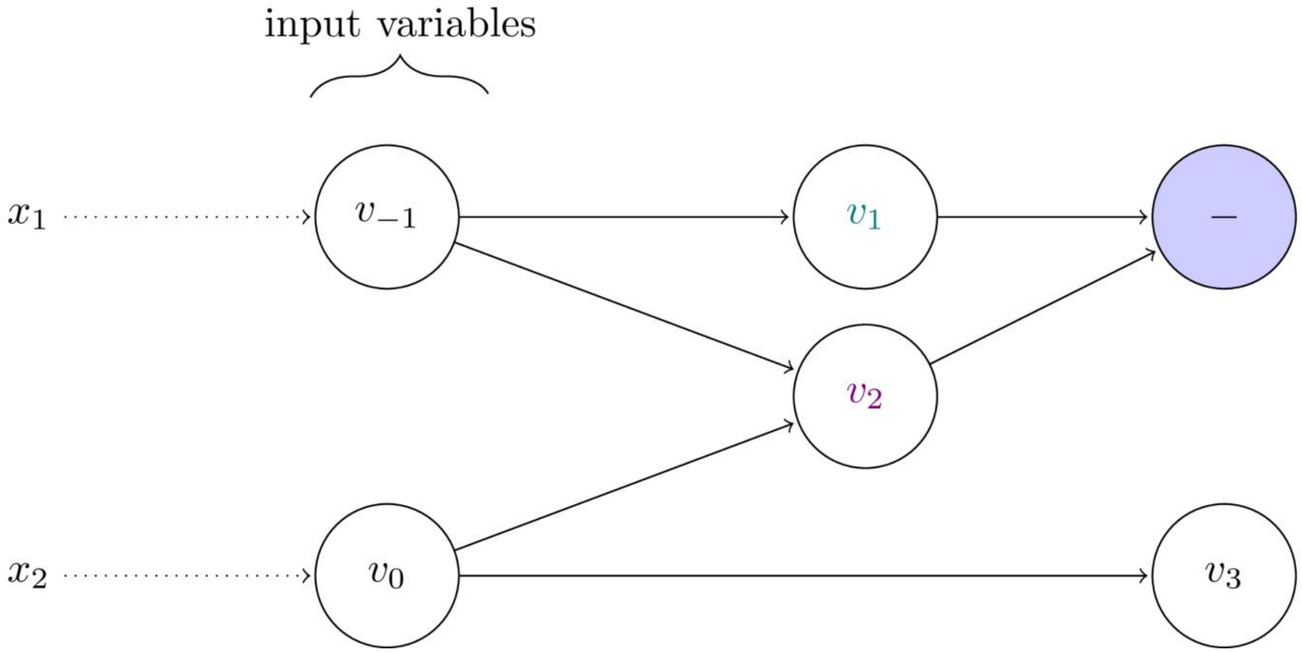


Computational graph

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

$v_1 \quad - \quad v_2 \quad v_3$

$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$

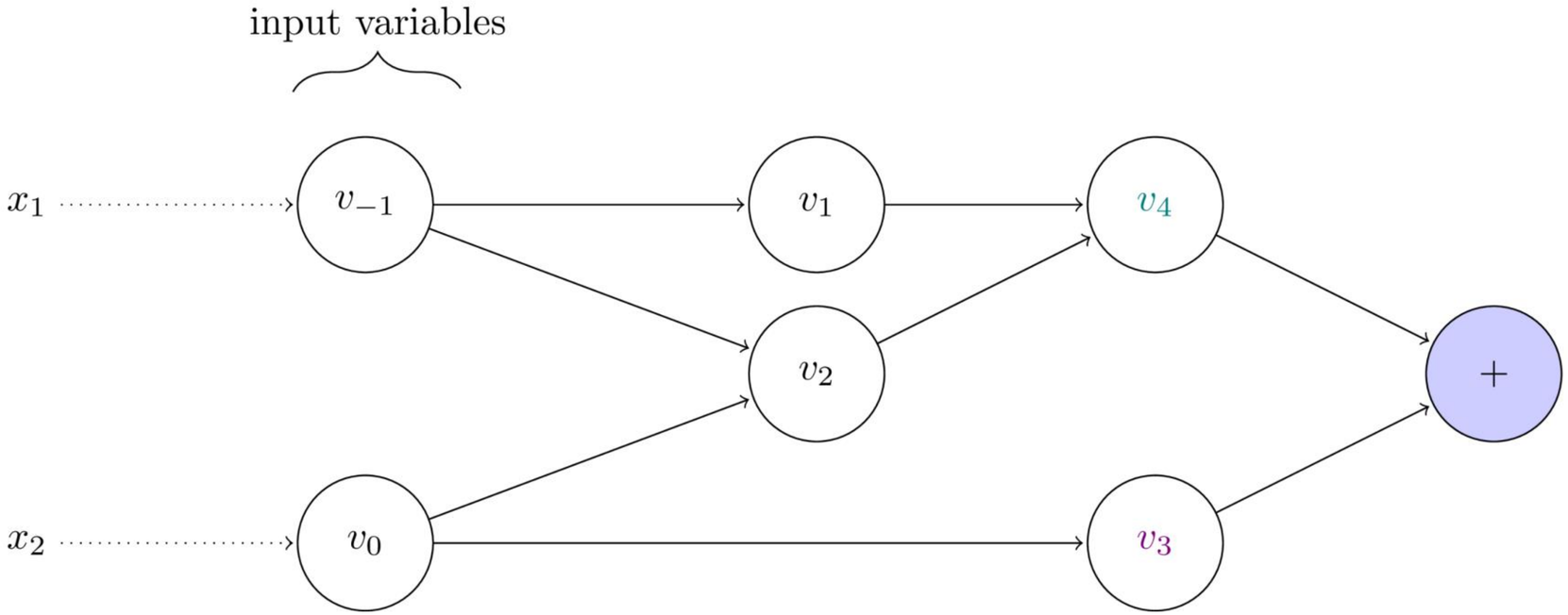


Computational graph

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$

v_4 + v_3



Computational graph

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

$$v_{-1} = x_1$$

$$v_0 = x_2$$

$$v_1 = exp(v_{-1})$$

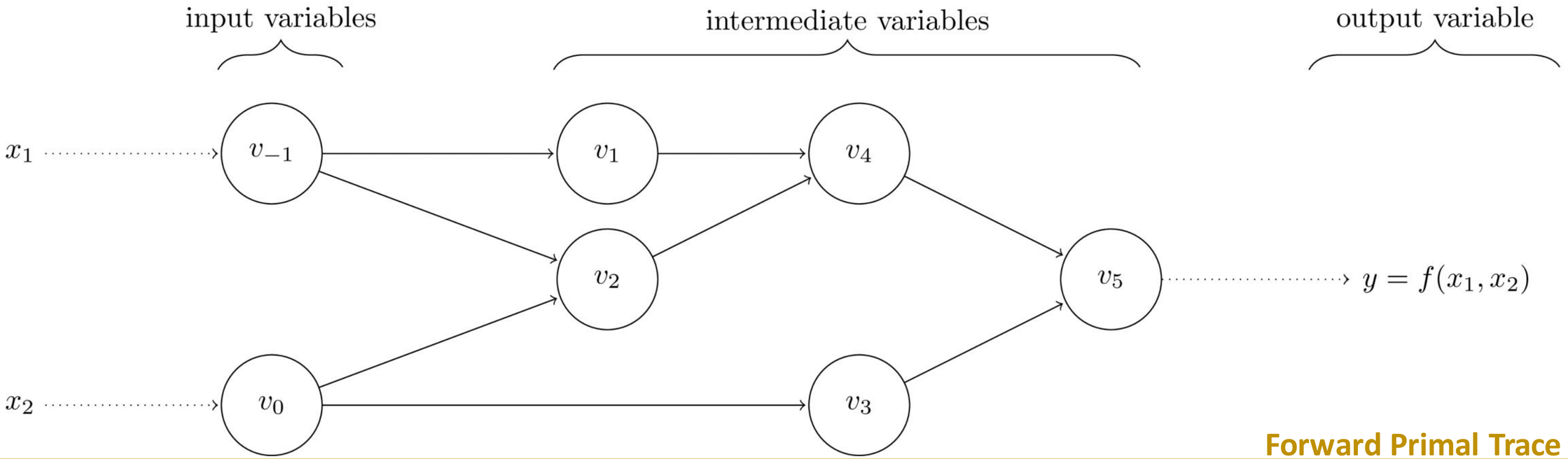
$$v_2 = v_{-1} \cdot v_0$$

$$v_3 = cos(v_0)$$

$$v_4 = v_1 - v_2$$

$$v_5 = v_4 + v_3$$

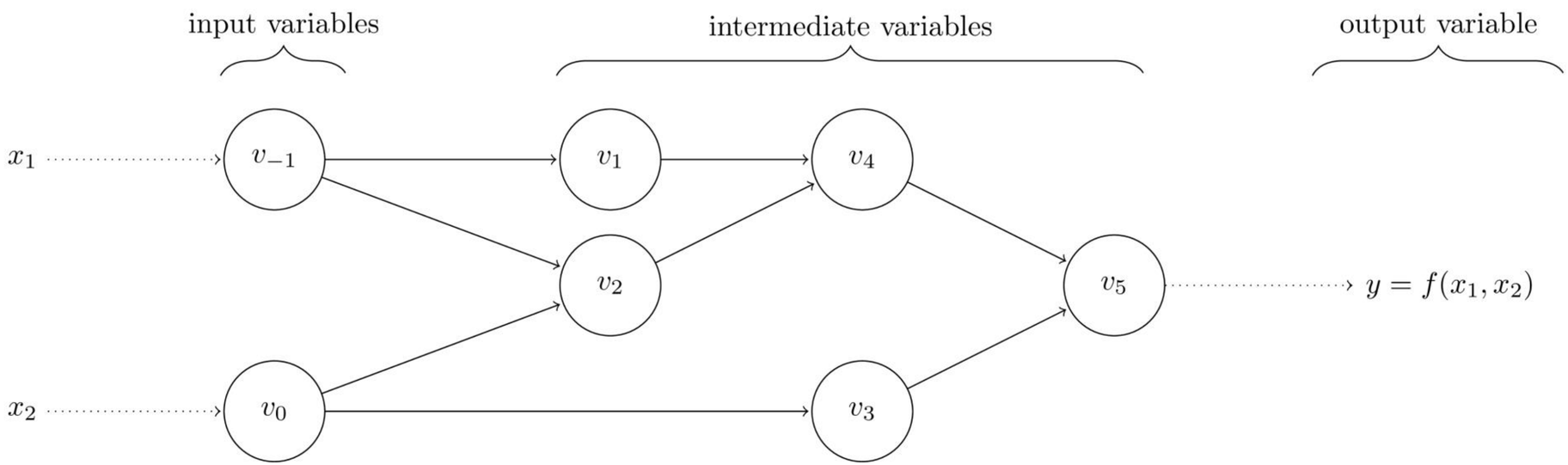
$$y = v_5$$



EXAMPLE: REVERSE MODE AD

Goal: Find the derivative of a function $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ with respect to the inputs x_n for the outputs y_m with specific values $\mathbf{y} = \mathbf{b}$

Reverse Derivative Trace

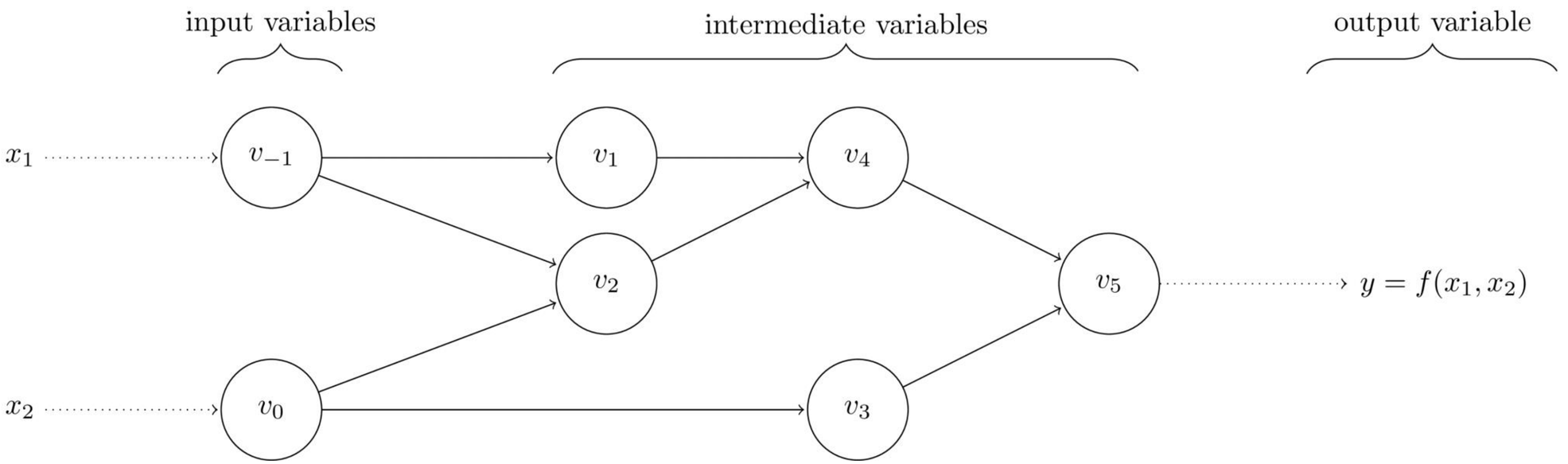


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

Forward Primal Trace

Reverse mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

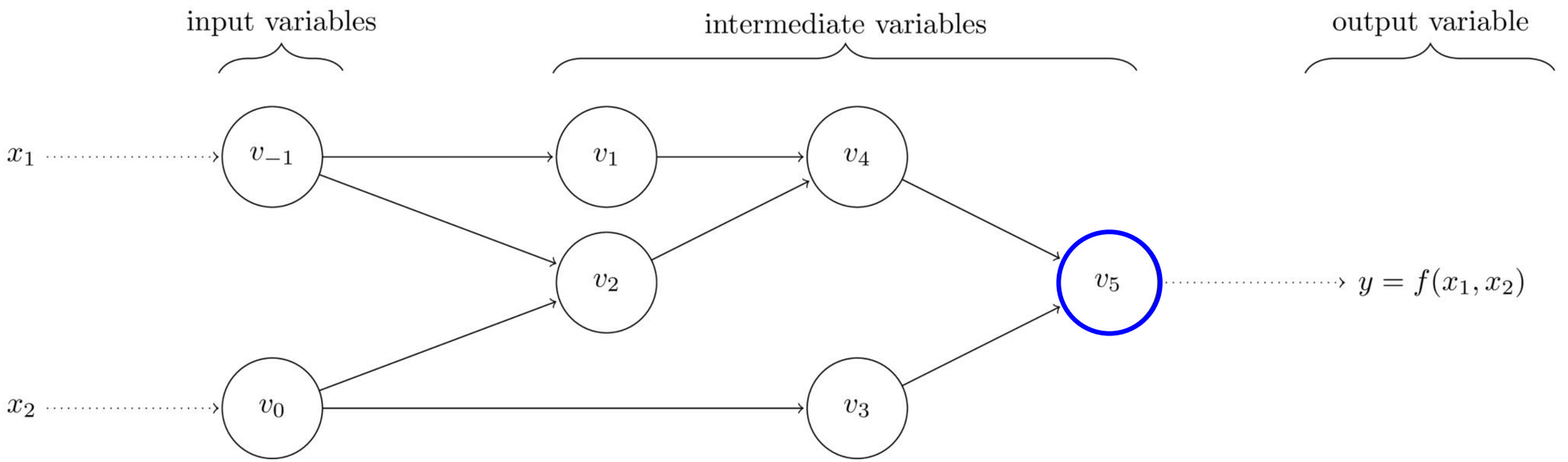


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

Adjoint: $$\nabla f = \frac{\partial y}{\partial x_i}$$

Reverse mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

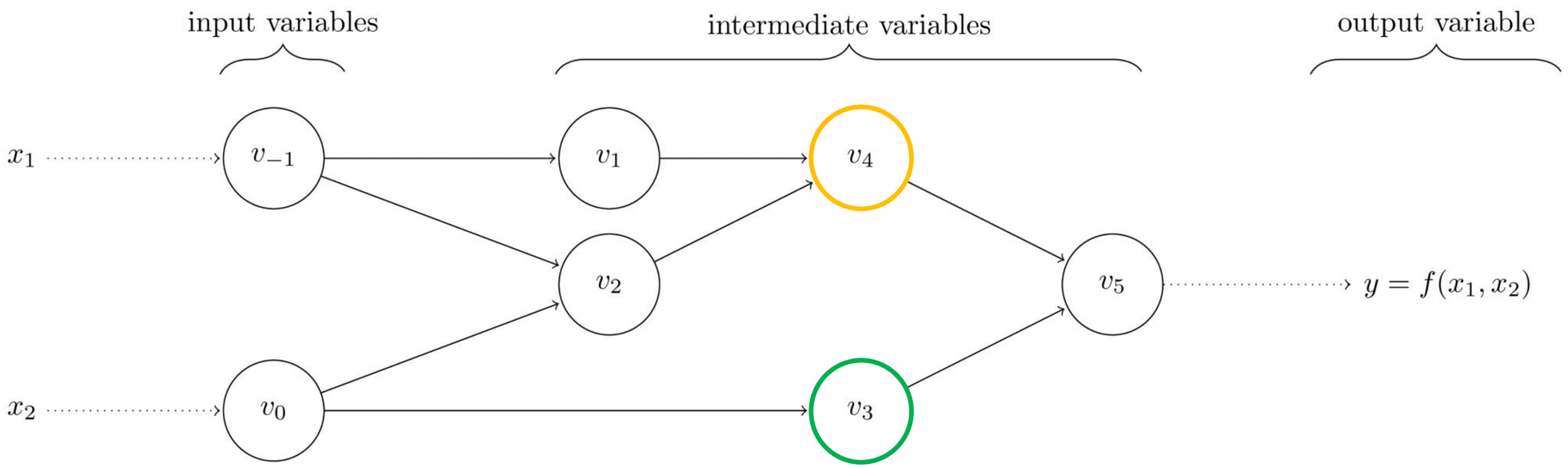


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

$$\nabla f = \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial x_i}$$

Reverse mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

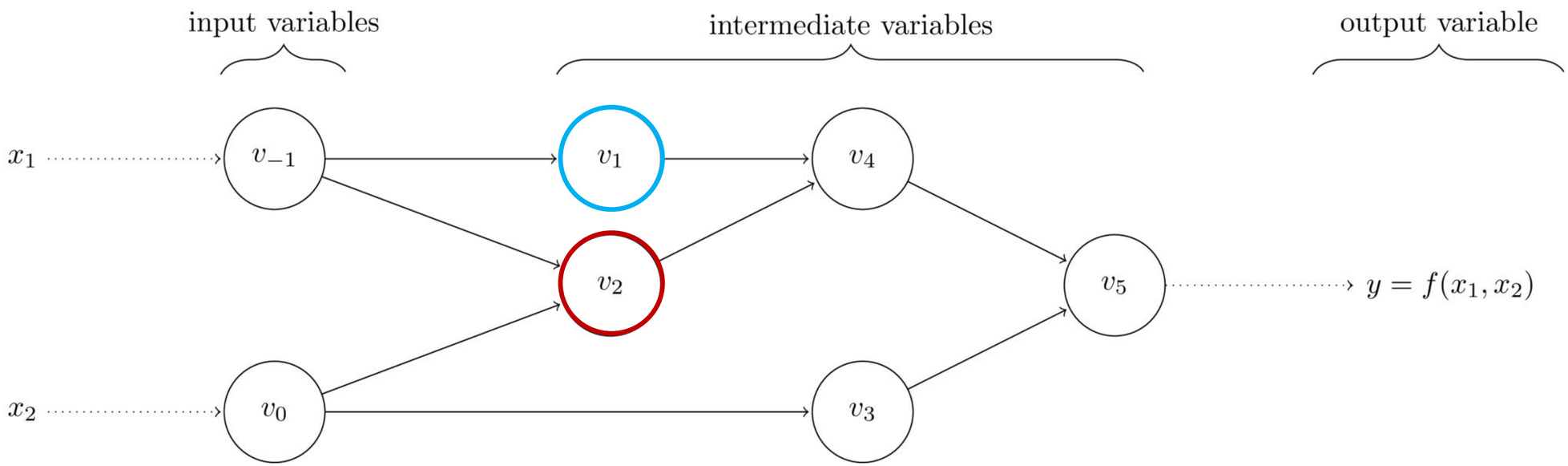


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

$$\nabla f = \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial x_i} + \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_3}\frac{\partial v_3}{\partial x_i}$$

Reverse mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

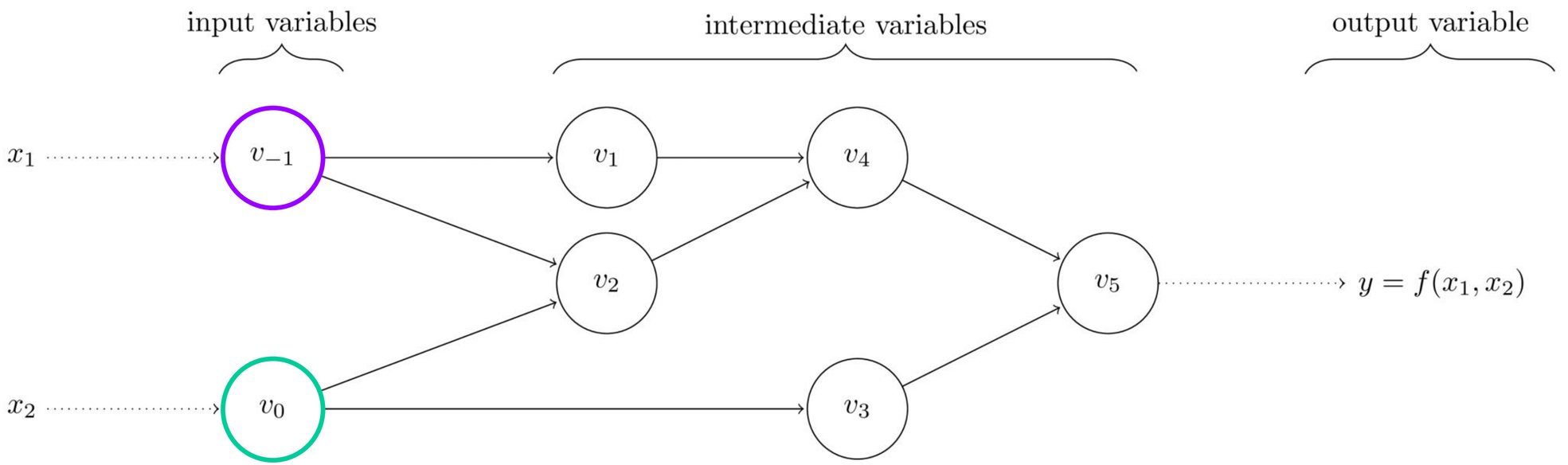


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

$$\nabla f = \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial v_1}\frac{\partial v_1}{\partial x_i} - \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial v_2}\frac{\partial v_2}{\partial x_i} + \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_3}\frac{\partial v_3}{\partial x_i}$$

Reverse mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

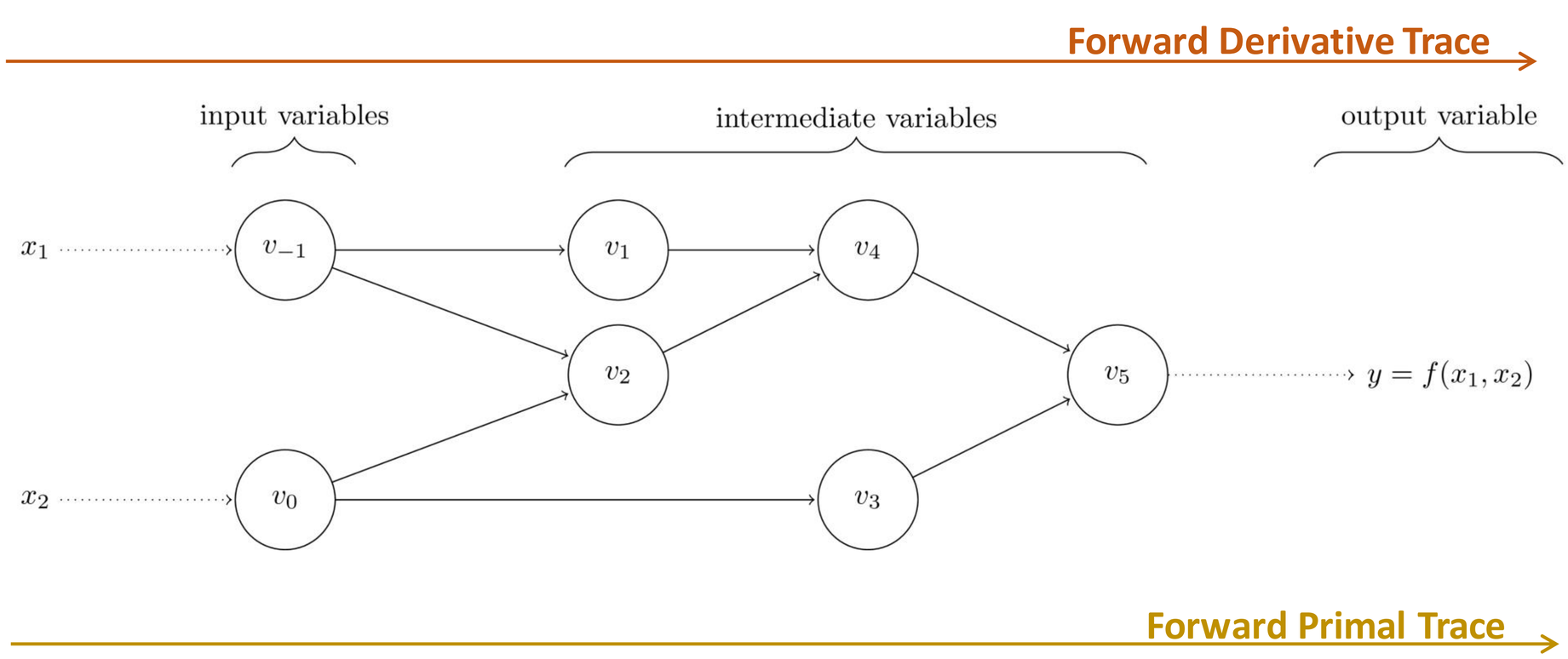


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(\boldsymbol{v_{-1}})$$
$$v_2 = \boldsymbol{v_{-1}} \cdot \boldsymbol{v_0}$$
$$v_3 = cos(\boldsymbol{v_0})$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

$$\nabla f = \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial v_1}\frac{\partial v_1}{\partial v_{-1}}\frac{\partial v_{-1}}{\partial x_i} - \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial v_2}\frac{\partial v_2}{\partial v_{-1}}\frac{\partial v_{-1}}{\partial x_i} - \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_4}\frac{\partial v_4}{\partial v_2}\frac{\partial v_2}{\partial v_0}\frac{\partial v_0}{\partial x_i} + \frac{\partial y}{\partial v_5}\frac{\partial v_5}{\partial v_3}\frac{\partial v_3}{\partial v_0}\frac{\partial v_0}{\partial x_i}$$

EXAMPLE: FORWARD MODE AD

Goal: Find the derivative of a function $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ for the outputs y_m with respect to the inputs x_n with specific values $\mathbf{x} = \mathbf{a}$



$$v_{-1} = x_1$$

$$v_0 = x_2$$

$$v_1 = exp(v_{-1})$$

$$v_2 = v_{-1} \cdot v_0$$

$$v_3 = cos(v_0)$$

$$v_4 = v_1 - v_2$$

$$v_5 = v_4 + v_3$$

$$y = v_5$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

x_1

x_2

$$\nabla f = \frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial x_2}$$

Forward mode

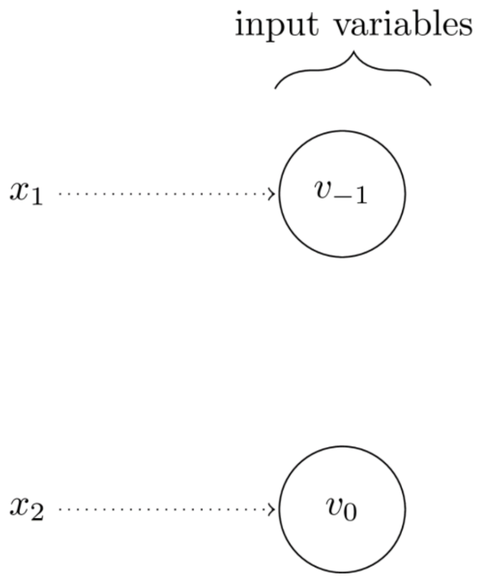
$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

x_2

$$\frac{\partial f}{\partial x_1}$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

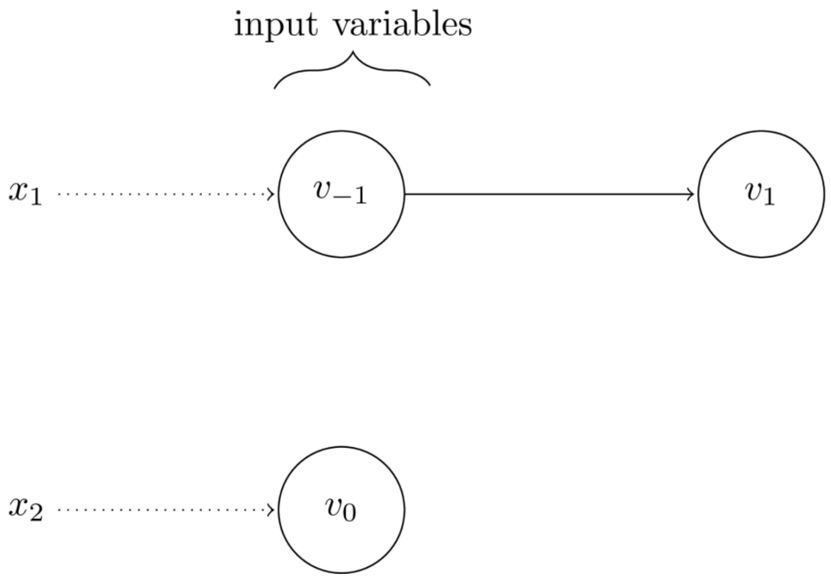


$$v_{-1} = x_1$$
$$v_0 = x_2$$

$$\dot{v}_{-1} = \frac{\partial v_{-1}}{\partial x_1} \quad \text{and} \quad \dot{v}_0 = \frac{\partial v_0}{\partial x_1}$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$

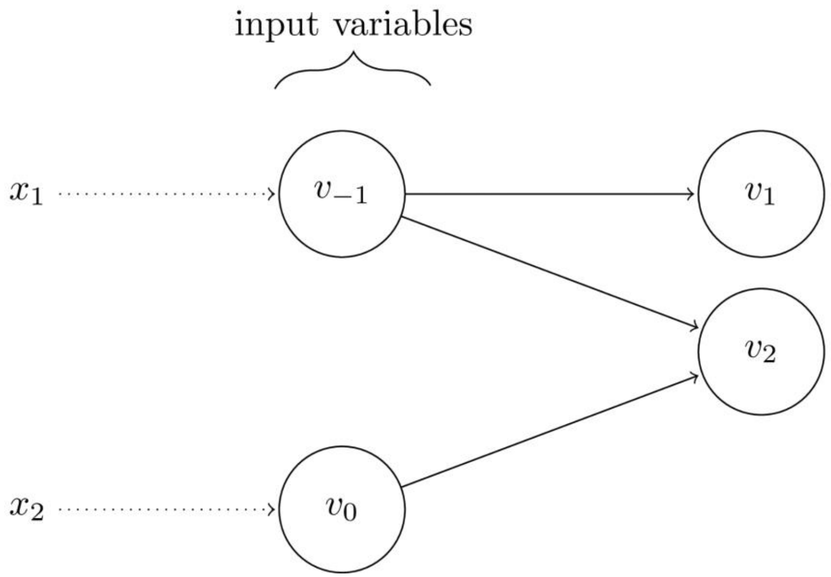
$$v_0 = x_2$$

$$v_1 = exp(v_{-1})$$

$$\dot{v}_1 = \frac{\partial v_1}{\partial v_{-1}} \dot{v}_{-1}$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$

$$v_0 = x_2$$

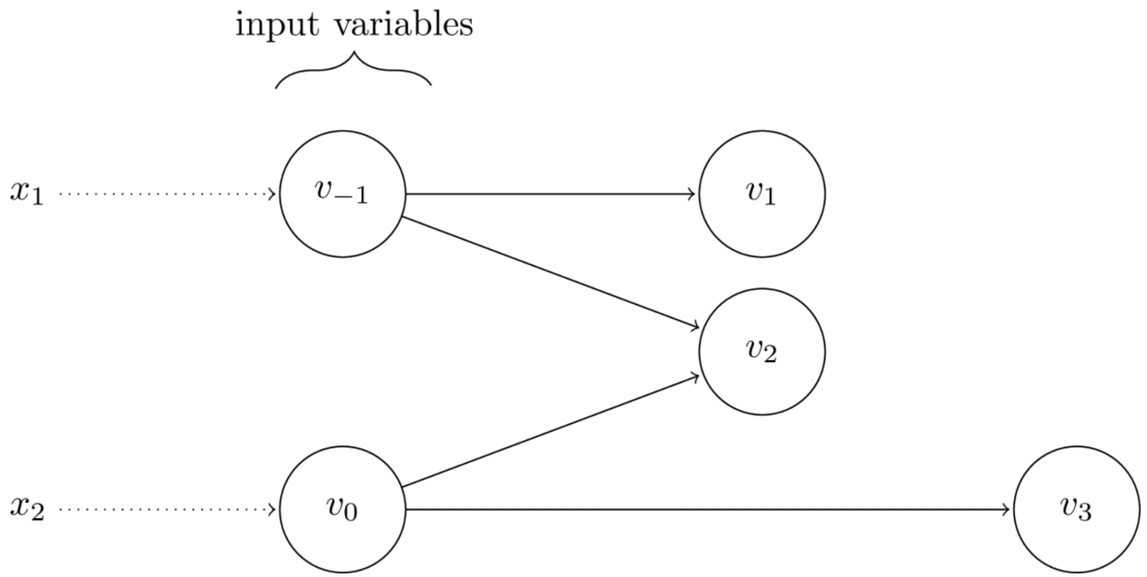
$$v_1 = exp(v_{-1})$$

$$v_2 = v_{-1} \cdot v_0$$

$$\dot{v}_2 = \frac{\partial v_2}{\partial v_{-1}} \dot{v}_{-1}$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$

$$v_0 = x_2$$

$$v_1 = exp(v_{-1})$$

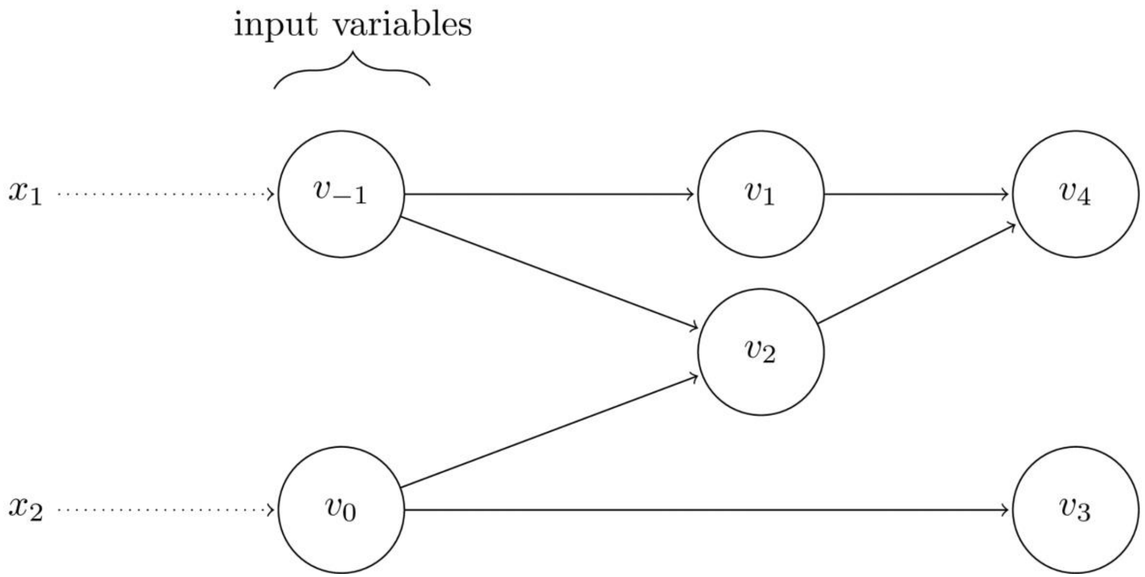
$$v_2 = v_{-1} \cdot v_0$$

$$v_3 = cos(v_0)$$

$$\dot{v}_3 = \frac{\partial v_3}{\partial v_0} \dot{v}_0$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

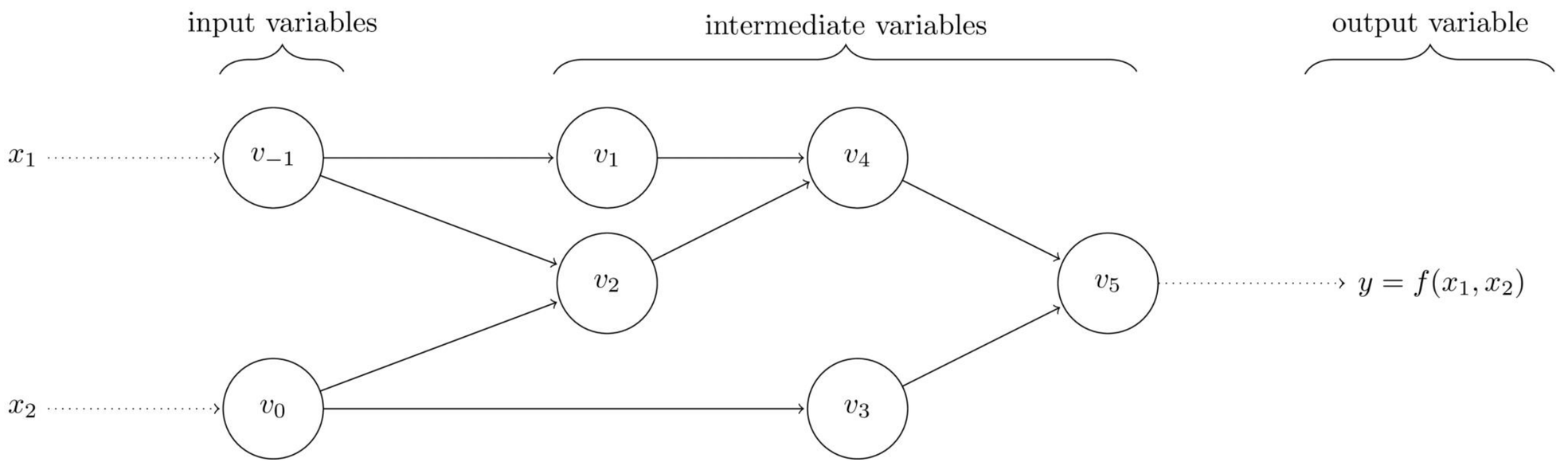


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$

$$\dot{v}_4 = \frac{\partial v_4}{\partial v_1}\dot{v}_1 + \frac{\partial v_4}{\partial v_2}\dot{v}_2$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$

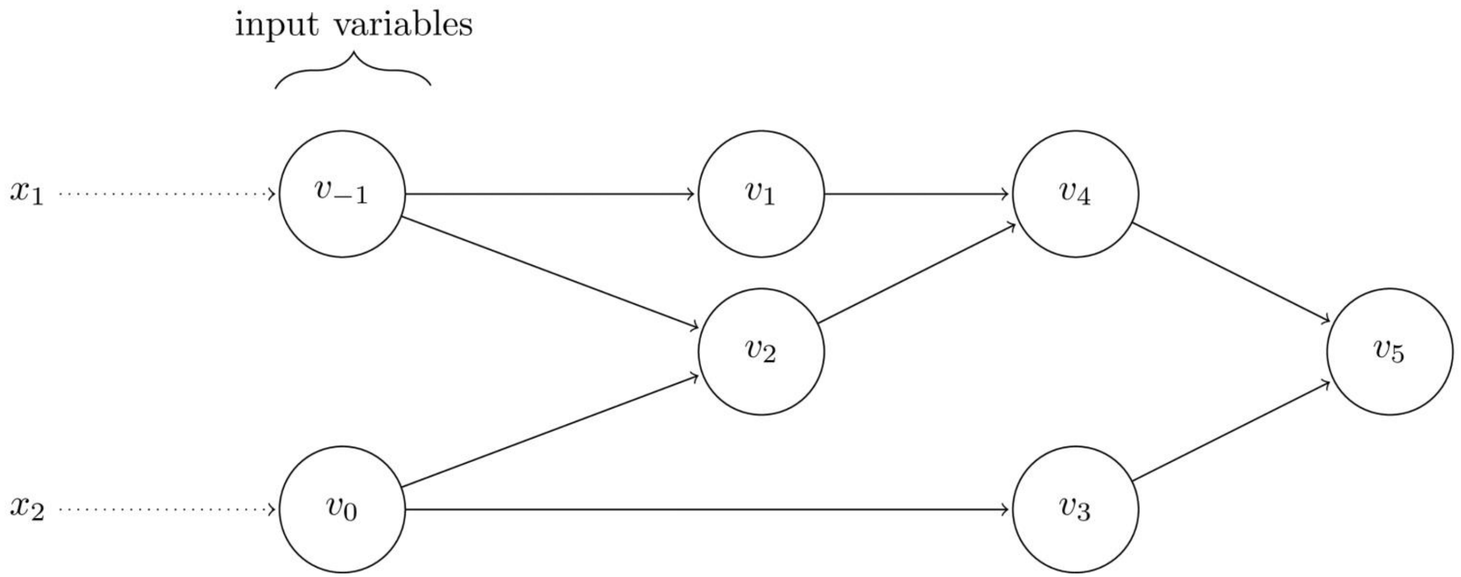


$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$
$$y = v_5$$

$$\dot{y} = \dot{v}_5$$

Forward mode

$$f(x_1, x_2) = exp(x_1) - x_1 \cdot x_2 + cos(x_2)$$



$$v_{-1} = x_1$$
$$v_0 = x_2$$
$$v_1 = exp(v_{-1})$$
$$v_2 = v_{-1} \cdot v_0$$
$$v_3 = cos(v_0)$$
$$v_4 = v_1 - v_2$$
$$v_5 = v_4 + v_3$$

$$\dot{v}_5 = \frac{\partial v_4}{\partial v_1}\dot{v}_1 + \frac{\partial v_4}{\partial v_2}\dot{v}_2$$

IMPLEMENTATION OF AD

dynamic ------ static

tape ------ graph

operator overloading ------ source code transformation

sublanguage ------ language differentiation

IMPLEMENTATION OF AD

AD using dual numbers

Define $x + \epsilon$ with $\epsilon^2 = 0 \wedge \epsilon \neq 0$

$f(x + \epsilon) = f(x) + \epsilon f'(x)$.

Consider

$$\begin{aligned} f(x+\epsilon) &= (x+\epsilon)^2 \\ &= \underbrace{x^2}_{f(x)} + \underbrace{2x}_{f'(x)}\epsilon + \underbrace{\epsilon^2}_{=0} \end{aligned}$$

- **Algebra**

$$(g \cdot f)(x + 1\epsilon) = g(x) \cdot f(x) + \epsilon(g(x) \cdot f'(x) + g'(x) \cdot f(x))$$

- **Primitives**

$$exp(x + \epsilon\, a) = \exp(x) + \epsilon a \exp(x)$$

COMPUTATIONAL COST OF AD

High dimensional functions

Derivatives become jacobians $J^f_{ij} := \frac{df_i}{dx_j}$

Be $f : \mathbb{R}^n \to \mathbb{R}^n \to \mathbb{R}^n \to \mathbb{R}$, $f(x) = (f_3 \circ f_2 \circ f_1)(x) \to J^f = \underbrace{J^{f_3}}_{1\times n}\underbrace{J^{f_2}}_{n\times n}\underbrace{J^{f_1}}_{n\times n} =: \mathbf{v}^T AB$

Forward Mode: $\mathbf{v}^T(AB) \Rightarrow \mathcal{O}(n^3)$ operations, $\mathcal{O}(n^2)$ memory, $\mathcal{O}(n^2)$ operations
Reverse Mode: $(\mathbf{v}^T A)B \Rightarrow \mathcal{O}(n^2)$ operations, $\mathcal{O}(n)$ memory, $\mathcal{O}(n^2)$ operations

Be $f : \mathbb{R} \to \mathbb{R}^n \to \mathbb{R}^n \to \mathbb{R}^n$, $f(x) = (f_3 \circ f_2 \circ f_1)(x) \to J^f = \underbrace{J^{f_3}}_{n\times n}\underbrace{J^{f_2}}_{n\times n}\underbrace{J^{f_1}}_{n\times 1} =: CD\mathbf{w}$

Forward Mode: $C(D\mathbf{w}) \Rightarrow \mathcal{O}(n^2)$ operations, $\mathcal{O}(n)$ memory, $\mathcal{O}(n^2)$ operations
Reverse Mode: $(CD)\mathbf{w} \Rightarrow \mathcal{O}(n^3)$ operations, $\mathcal{O}(n^2)$ memory, $\mathcal{O}(n^2)$ operations

EXAMPLES OF AUTOMATIC DIFFERENTIATION

Google Colab notebooks available here:

AD python demo: https://colab.research.google.com/drive/1LX1wJWvJXaZDfZTThbyIxYBNRhEPWH0D?usp=sharing

AD Juli demo: https://colab.research.google.com/drive/1itEcpqcjLhL8kuM_LrdcZdsIN1E-y9Eq?usp=sharing

great HEP tutorial on very general autodial:
https://theoryandpractice.org/stats-ds-book/autodiff-tutorial.html

SUMMARY & CONCLUSION

Main take away:

scientific motivated inductive bias helps to be more robust, more data efficient and better interpretable

My personal message:

Write better code!
Share more data!
Build more open-source software!

This will accelerate research cycles and lets you engage with peers early on!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS