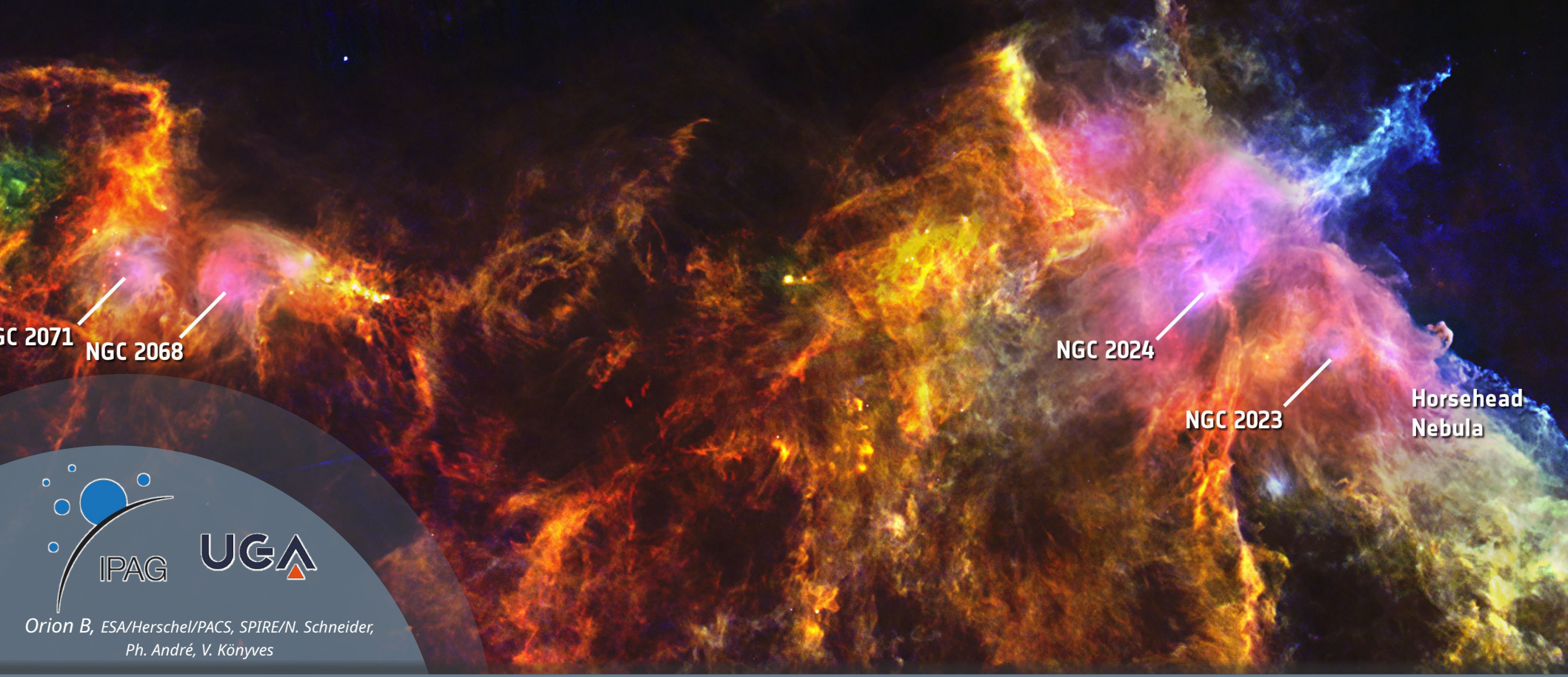


IPAG UGA

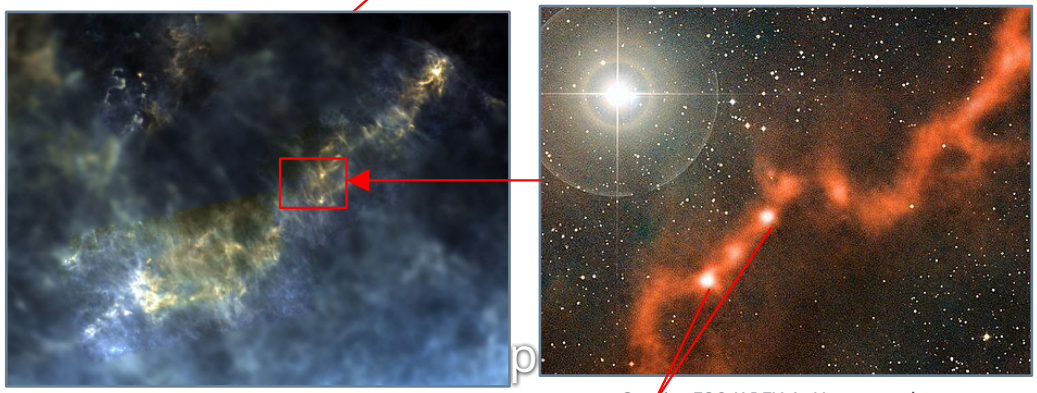
Orion B, ESA/Herschel/PACS, SPIRE/N. Schneider, Ph. André, V. Könyves

Neural network estimation of the average density in molecular clouds and prestellar cores : A benchmark of modern architectures

Zack Ribeiro & Pierre Hily-Blant

Context

Captured by Frédéric CHIOLA



(B, Hasenberger et al. 2018), Infrared emission

Credit : ESO/APEX A. Hacar et al.

Dense cores

Dense cores and star formation

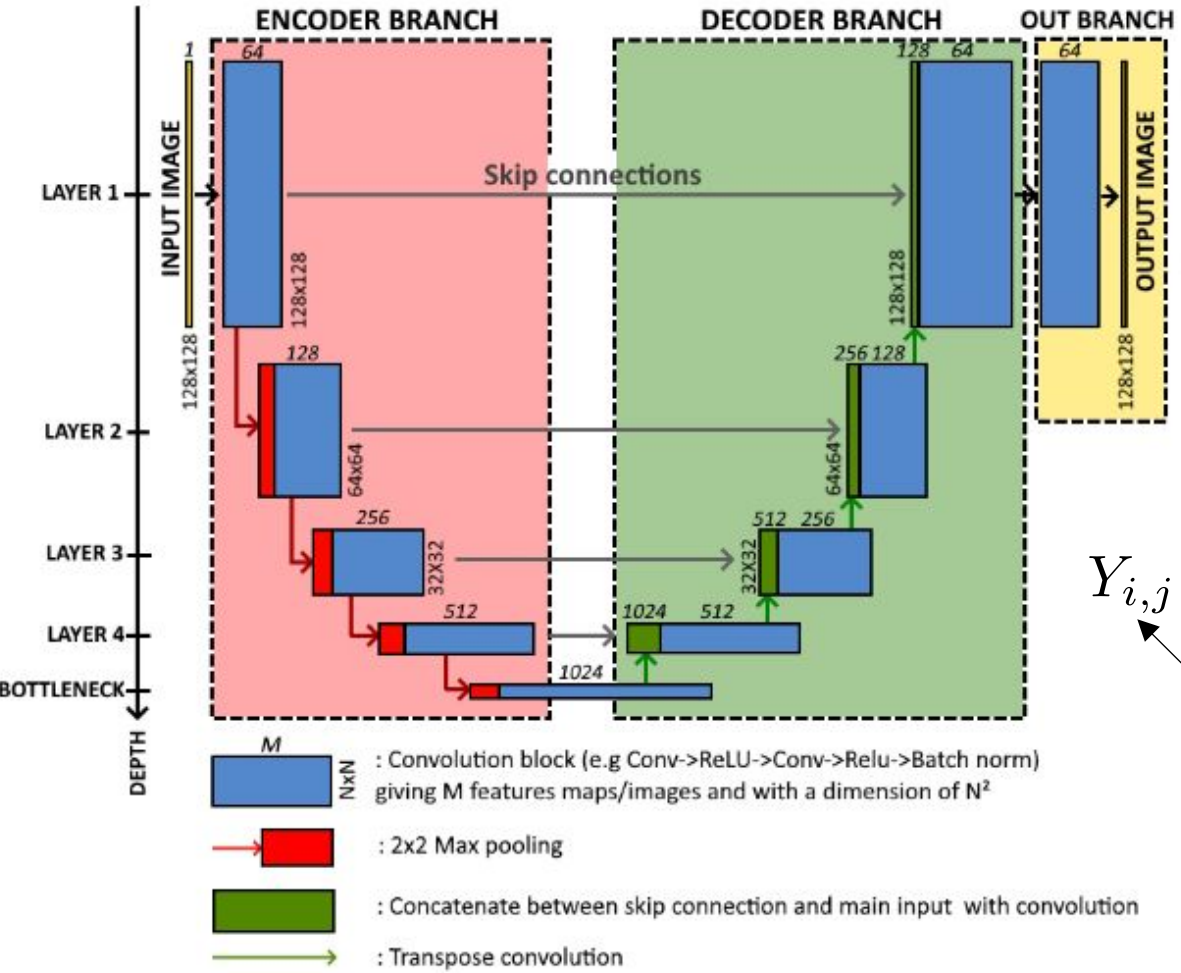
- Dense cores are the birth sites of stars.
- Critical parameters :
  - Density profil, radius, mass
- Critical questions :
  - Star Formation Rate ; Origin of the initial mass function

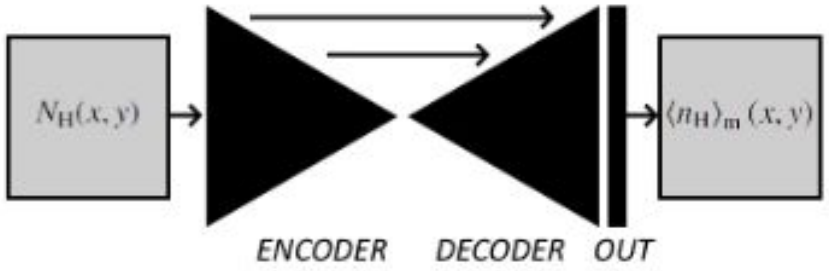
Observational challenge

- Dust observations provide column density
- Star formation depends on volume density
- Recovering volume density is a degenerate problem

$$N_{\rm H}(x,y) = \int_0^L n_{\rm H}(x,y,z)dz. \longrightarrow \langle n_{\rm H}\rangle_{\rm m}(x,y) = \frac{1}{N_{\rm H}} \int_0^L n_{\rm H}^2(x,y,z)dz.$$

Unet – Convolutional Neural Networks





(Unet, ***Ronneberger et al (2015) ; Oktay et al (2018) ; LeCun et al (1989)***)

$$Y_{i,j} = f\left(\sum_{m=1}^{M}\sum_{n=1}^{N} W_{m,n} \cdot X_{i+m,j+n} + b\right)$$

Width and Height of the used kernel (M=N=3)

Pixel at (i,j) in the resulting image Y

Element (m,n) of the kernel

Conditional Invertible Neural Networks (cINNs)

- **Generate** samples by a sequence of invertible transformations :

$$Z = f_K \cdot f_{K-1} \cdot \ldots \cdot f_1(X, c).$$

From sample from Gaussian prior, to a data sample conditionned.

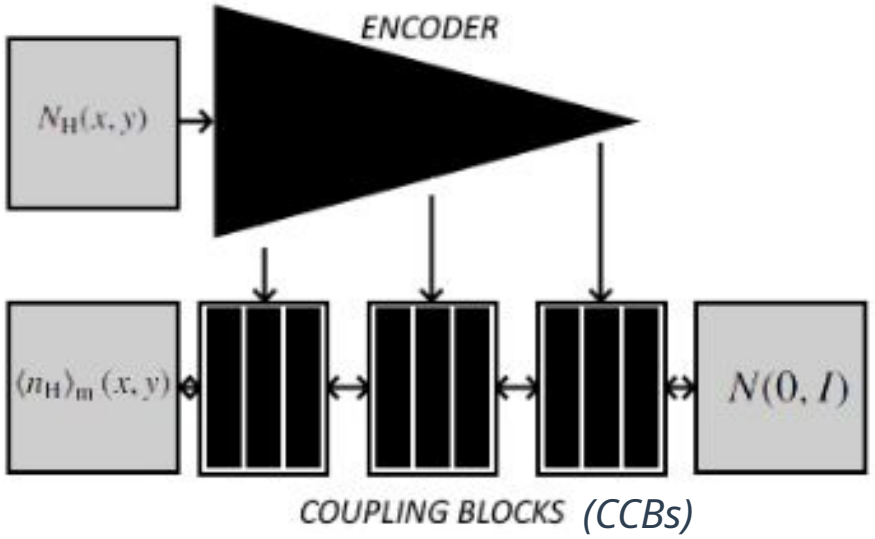
$$p_z = N(0, I) \longrightarrow x_0 \sim p_{\text{data}}.$$

- **Training :** Learn to transform the complex data distribution into a gaussian one using **invertible** transformations.

$$p_{\text{data}}(X|c) = p_z(f(X, c))|\det \frac{\partial f(X, c)}{\partial X}|.$$

- **Inference :** Network is **inverted** in order to transform a sample from the gaussian distribution to the learned data distribution

$$X = f^{-1}(Z|c).$$



(cINNs, ***Ardizzone et al. (2021) ; Rezende & Mohamed (2016)***)

transformation of a CCB is:

$$\begin{cases} v_1 = u_1 \odot \exp(s_1(u_2, c)) + t_1(u_2, c), \\ v_2 = u_2 \odot \exp(s_2(v_1, c)) + t_2(v_1, c), \end{cases} \quad \begin{cases} u_2 = (v_2 - t_2(v_1)) \odot \exp(-s_2(v_1)) \\ u_1 = (v_1 - t_1(u_2)) \odot \exp(-s_1(u_2)) \end{cases}$$

where s_i, t_i are scale and translation functions ← **Learned** by convolutional neural networks (CNNs)

Denoising Diffusion Probabilistic Models (DDPMs)

- **Generate** samples by gradually transforming white noise, i.e sample from Gaussian prior, into a data sample.

$$p_z = \bar{N}(0, I) \longrightarrow x_0 \sim p_{\text{data}}.$$

- **Training** : a U-Net learn to predict the noise added to $\langle n_{\text{H}} \rangle_{\text{m}}$

$$dx_t = f(x,t)dt + g(t)dW_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t,$$

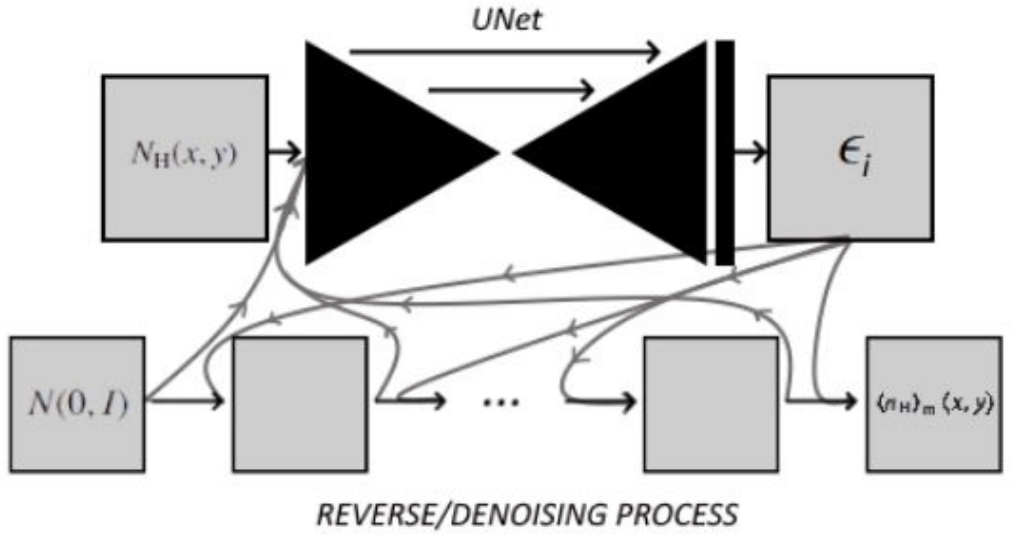
where W_t is a Wiener process. Here:

  - $g(t) = \sqrt{\beta(t)}$ is the diffusion strength,
  - $f(x,t) = -\frac{1}{2}\beta(t)x_t$ is the drift term,
  - $t \in [0,T]$ parametrized the diffusion trajectory.

- **Inference** : Sequence of learned denoising steps

$$dx_t = [f(x,t) - g(t)^2 \nabla_{x_t} \log p_t(x_t)]dt + g(t)d\overline{W}_t,$$

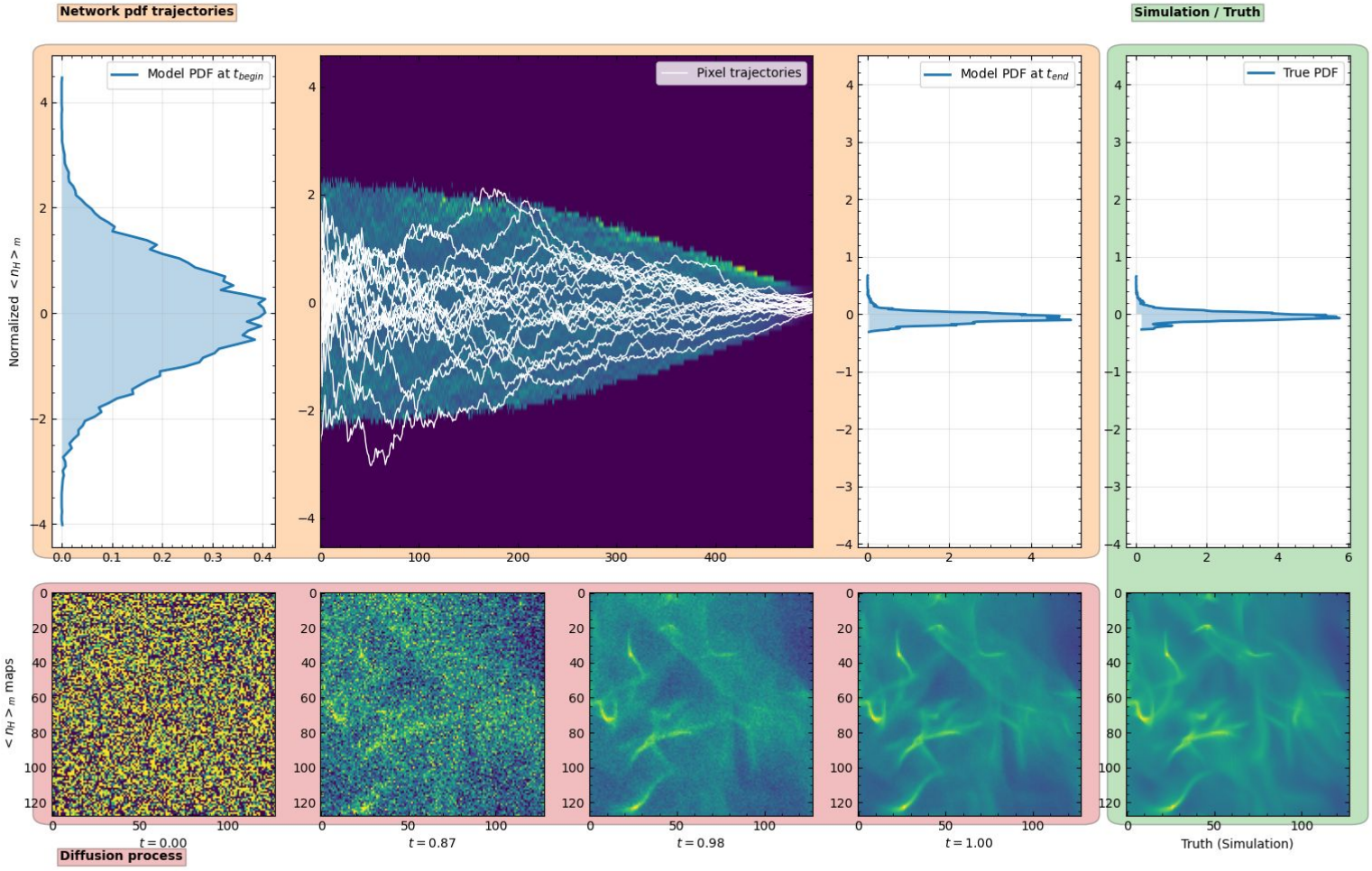
Likelihood



*(DDPMs, **Ho et al. (2020)**)*

DDPMs – Diffusion trajectories

*(DDPMs, **Ho et al. (2020)**)*



- Each element of the map follows a **diffusion process**
- Each **amplitude** of diffusion step is **predicted by a U-Net.**

Dataset & Training

Simulation (RAMSES)

Ntormousi & Hennebelle (2019 , GALACTICA database)

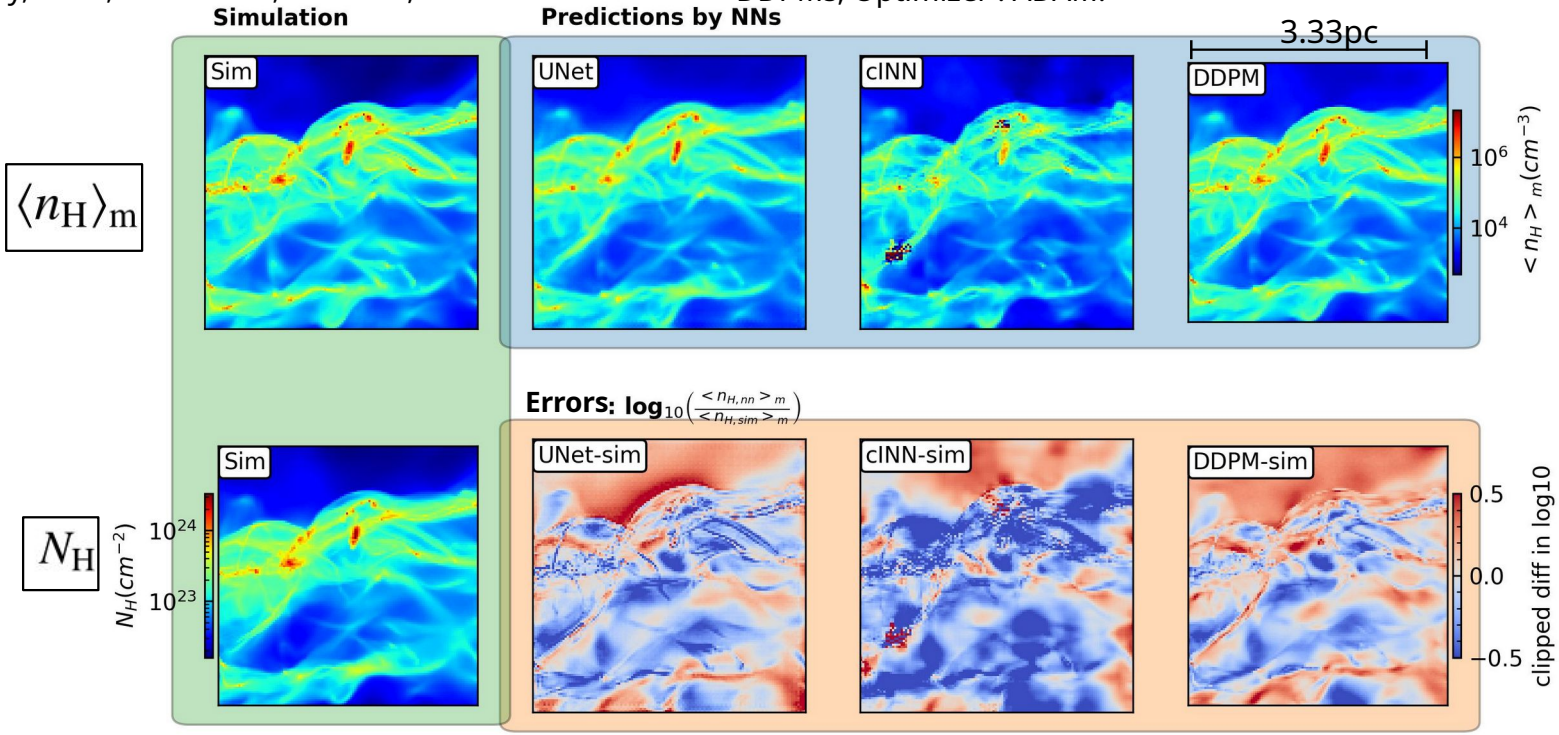
- Width: 66,1pc, resolution : AMR (down to 500au)
- Turbulence, Gravity, Magnetic field (Ideal MHD), Not isothermal

+ Simulation (IDEFIX) for validation only

- Turbulence, Gravity, MHD, Isothermal, Periodic ; res = 512

Training : 100 regions

- 128px - 3.33pc wide
- 70 % for training, 30 % for validation.
- Diffuse to dense regions
- Data augmentation, ExpMovAverage (EMA) for cINNs and DDPMs, Optimizer : ADAM.



Benchmark on validation-set (from Simulations)

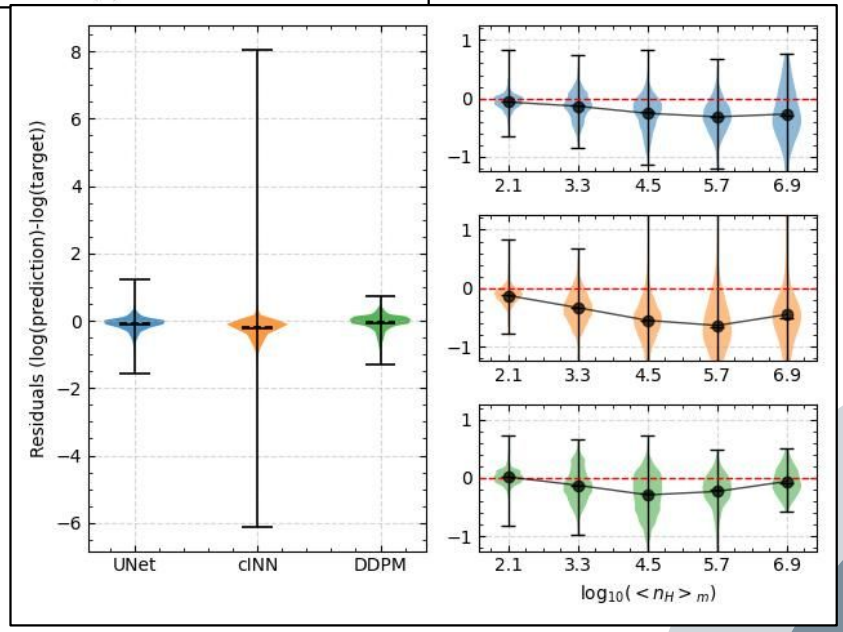
- **U-Net**: faster inference and training, great performance (**factor 1.25**)
- **DDPMs**: slower but same performance and probabilist, i.e can handle degenerate problems.
- **cINNs**: unstable and difficult to train but probabilist with exact likelihood

Metric	UNet	cINN	DDPM
MSE (log10)	0.20	0.50	0.25
Parameters ($\times 10^7$)	13	3.8	0.8
Inference speed (regions/s)	3.5	3.4	0.32
Error at 80% accuracy	0.25	0.39	0.25
Training stability	High	Low	High

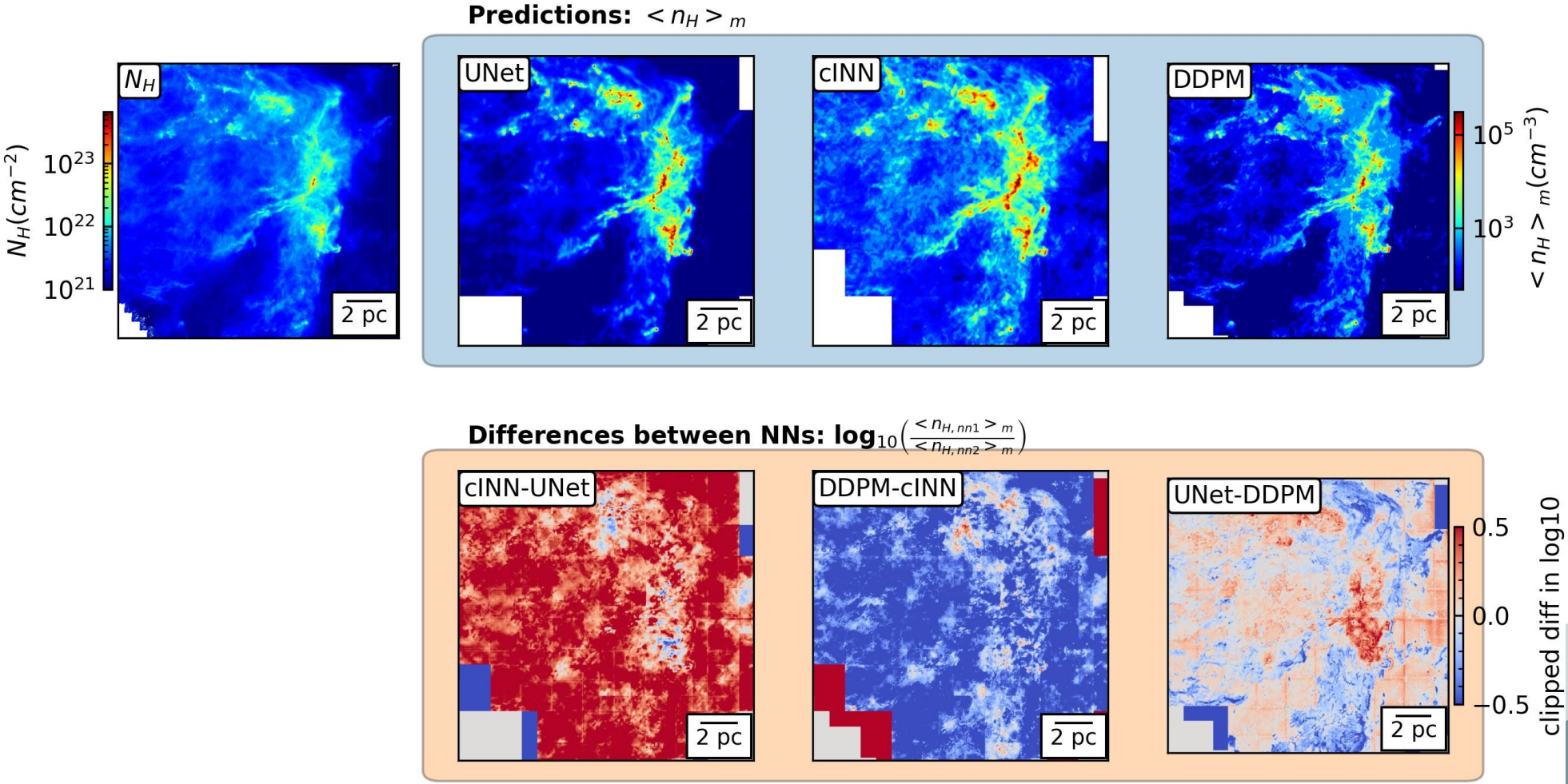
(GPU : RTX 2070 SUPER)

$$\mathrm{Acc}(\sigma) = \frac{1}{B}\sum_{i\in\{B\}} \frac{N\left(|Y_{\mathrm{pred}} - Y_{\mathrm{true}}| < \sigma\right)}{N_i},$$

$$\mathrm{MSE} = \mathbb{E}\left[\left(\log Y_{\mathrm{pred}} - \log Y_{\mathrm{true}}\right)^2\right],$$



Application on Orion B



*(in prep, **Ribeiro & Hily-Blant (2026)**)*

Benchmark on Orion B

- **Core catalog** from *Könyves et al. (2020).*

$$\frac{\langle n_{\rm H}\rangle_{\rm m}}{n_c} = \frac{n_d}{n_c} + \frac{n_c L_c}{N_{\rm H}}\left(1 - \frac{n_d}{n_c}\right)$$

- **cINNs** : best for predicting dense cores densities ;
  DDPMs, U-Net : for the diffuse environment

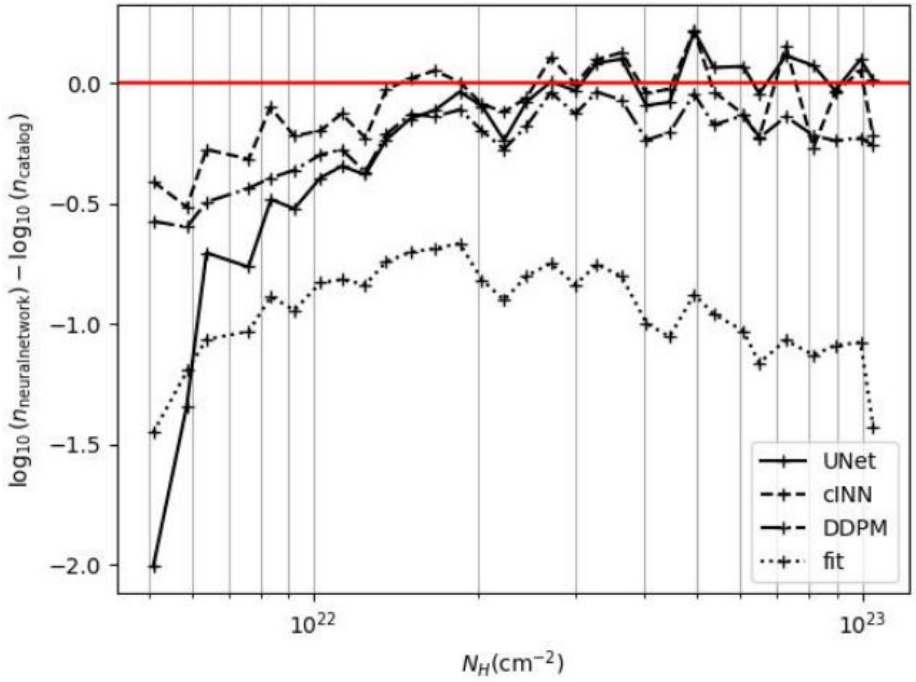
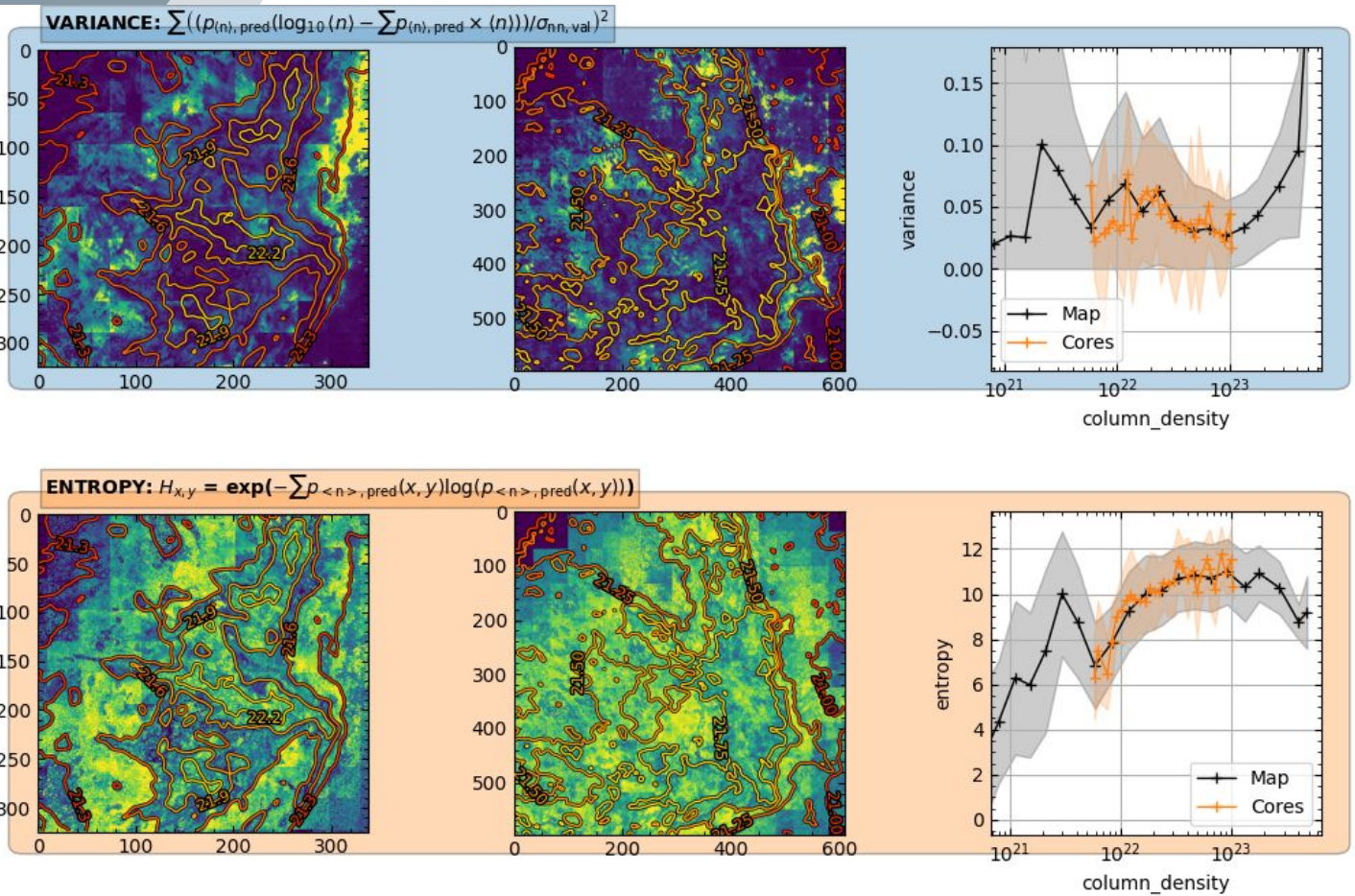


Fig. 10. Log-binned averages of differences in logspace between predicted average core densities and catalogued values. (≈ 400 cores)

Advantages of using probabilistic models



- **Probabilistic** models predict **distributions**.
  → Possible to :
  - **measure model error**
  - **difficulty** of predicting certain regions.
- **Degeneracy is not uniform** across the cloud and column density.

Conclusion

Results

- **Neural networks can estimate physical properties** (as the volume density) in molecular clouds by extracting and combining information from observational maps.
- **Probabilistic** neural networks can address **degenerate problems**.

Limitations

- **Dust emission only is limited**, an architecture capable of also using molecular emission to leverage some degeneracy (e.g separation of structures along the line of sight) is needed.
- **Missing simulations** to achieve peak performance of cINN and to validate using external simulations.

Future

- **New target : Catalog of cores**, multi-modal using molecular emissions and dust emission. → New measurement of CMF by taking into account the multiplicity along l.o.s

Benchmark on Orion B

- **Core catalog** from ***Könyves et al. (2020).***
- Radii and core list are **unchanged**; densities are **replaced** with our estimates.

$$\frac{\langle n_{\rm H}\rangle_{\rm m}}{n_c} = \frac{n_d}{n_c} + \frac{n_c L_c}{N_{\rm H}}\left(1 - \frac{n_d}{n_c}\right)$$

- cINNs : best for predicting dense cores densities ; DDPMs, Unet : for the diffuse environment

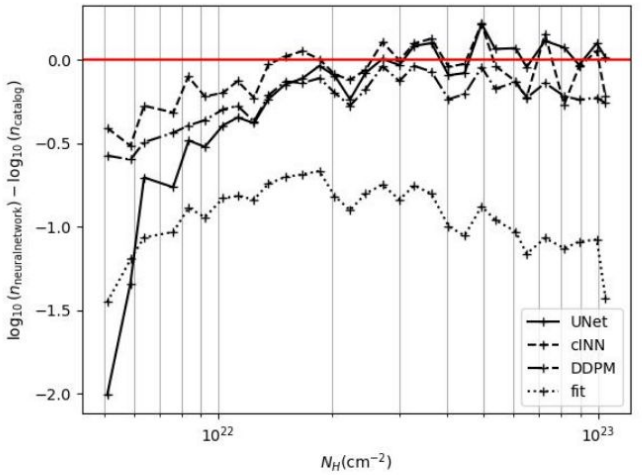
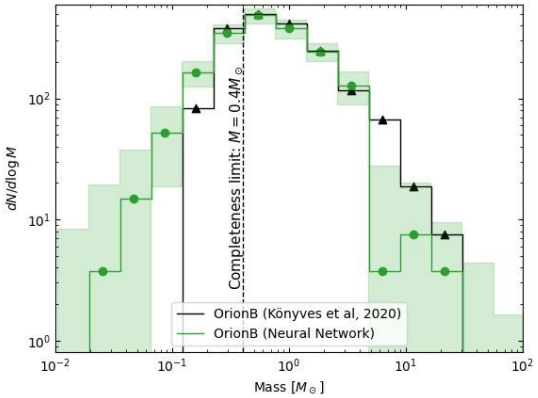
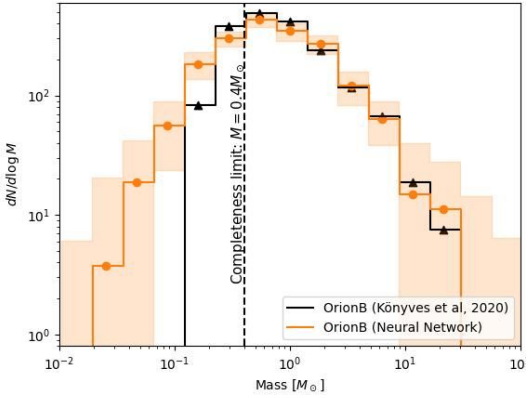


Fig. 10. Log-binned averages of differences in logspace between predicted average core densities and catalogued values. (≈ 400 cores)

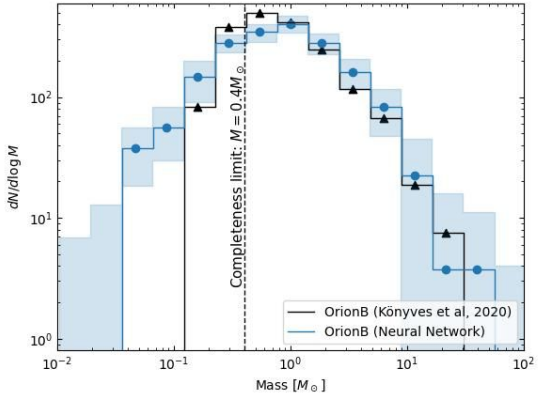
DDPMs



cINNs



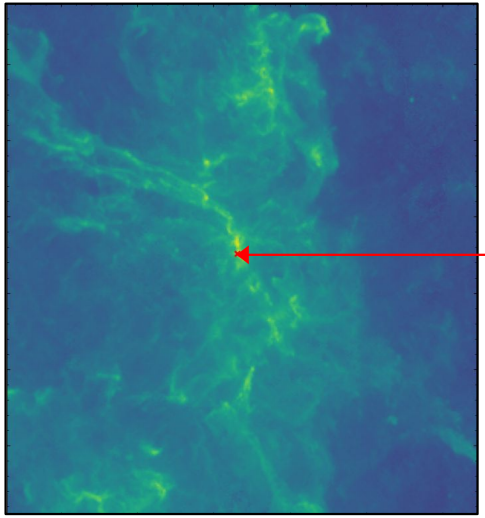
U-Net

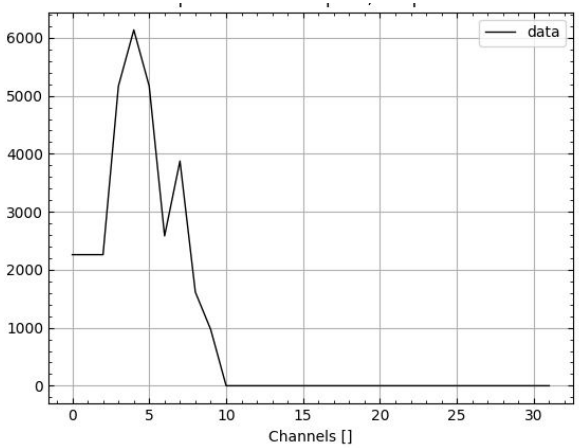


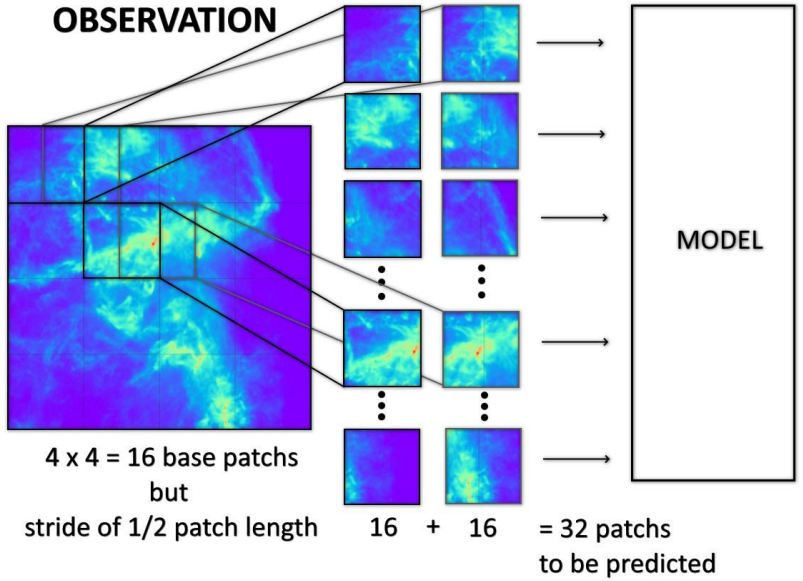
Color bars : 95 % confidence

Method to apply the neural networks on observations

- Neural network **applied repeatedly** to small observation **patches**
- Used for **probabilistic** modeling of line-of-sight densities
- Model predicts a **distribution** of average densities
- Outputs can be summarized using:
  - Mean predicted density
  - Most likely density value



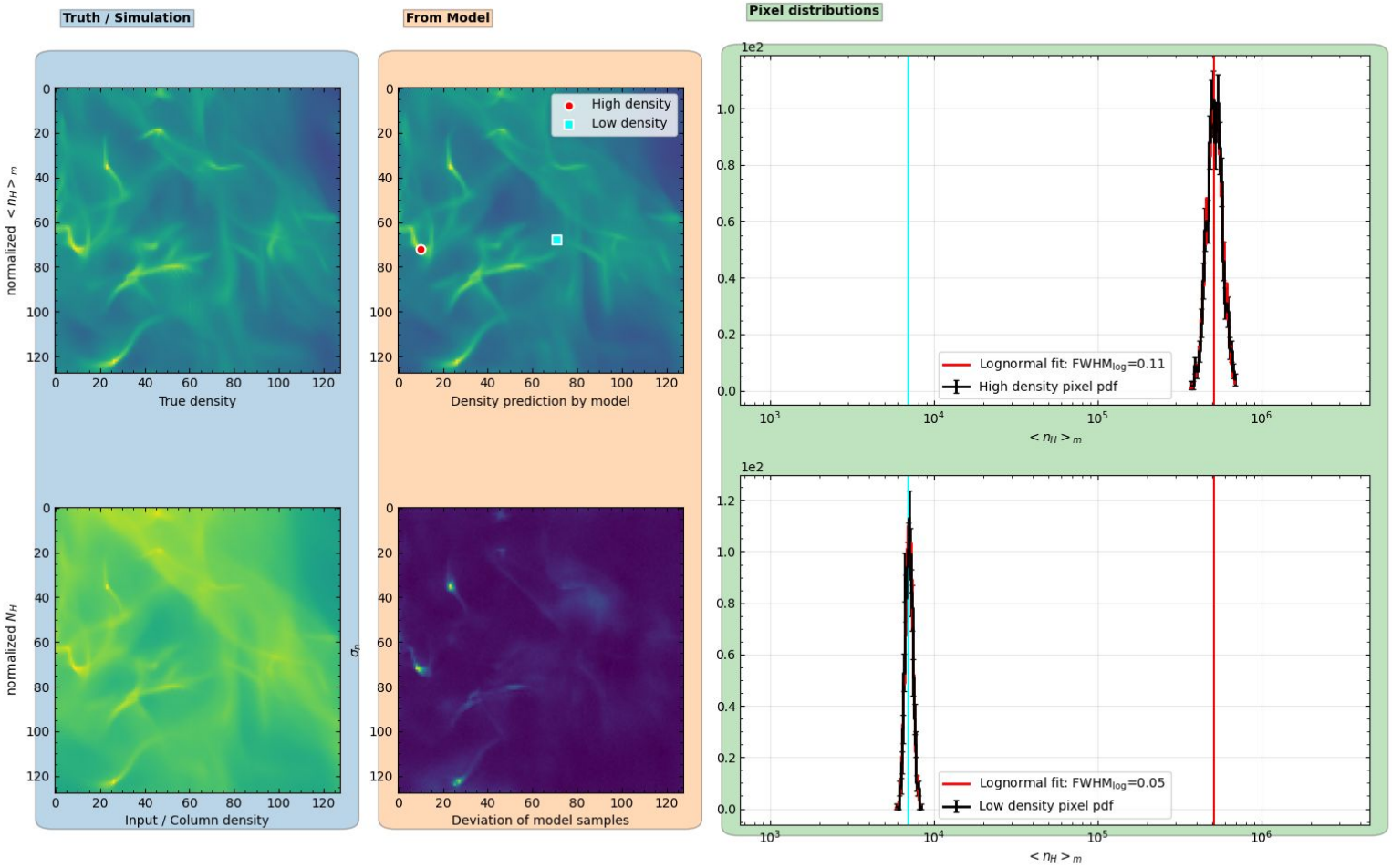




~Predicted mass-weighted average density along the l.o.s

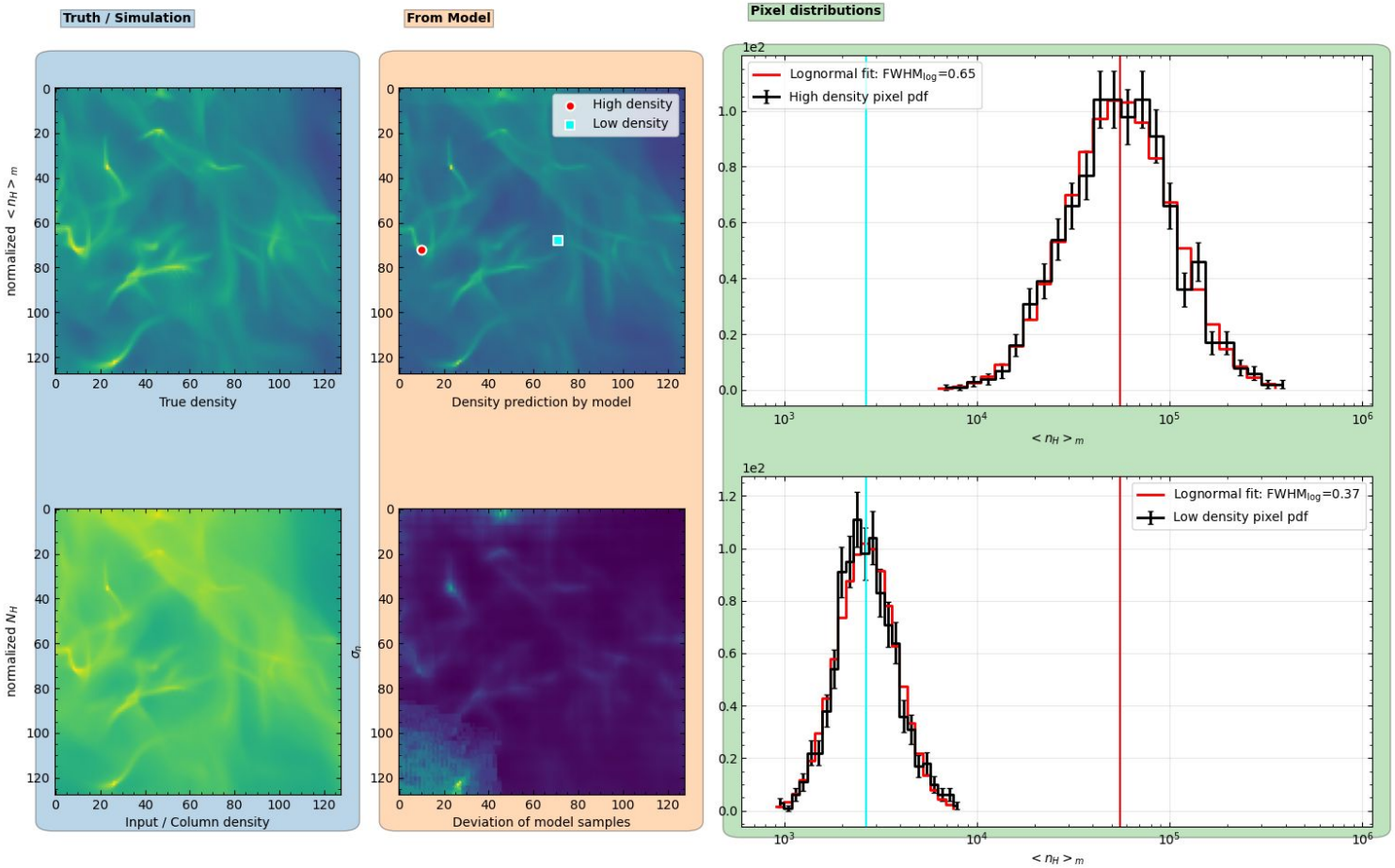
Denoising Diffusion Probabilistic Models (DDPMs) – III

*(DDPMs, **Ho et al. (2020)**)*

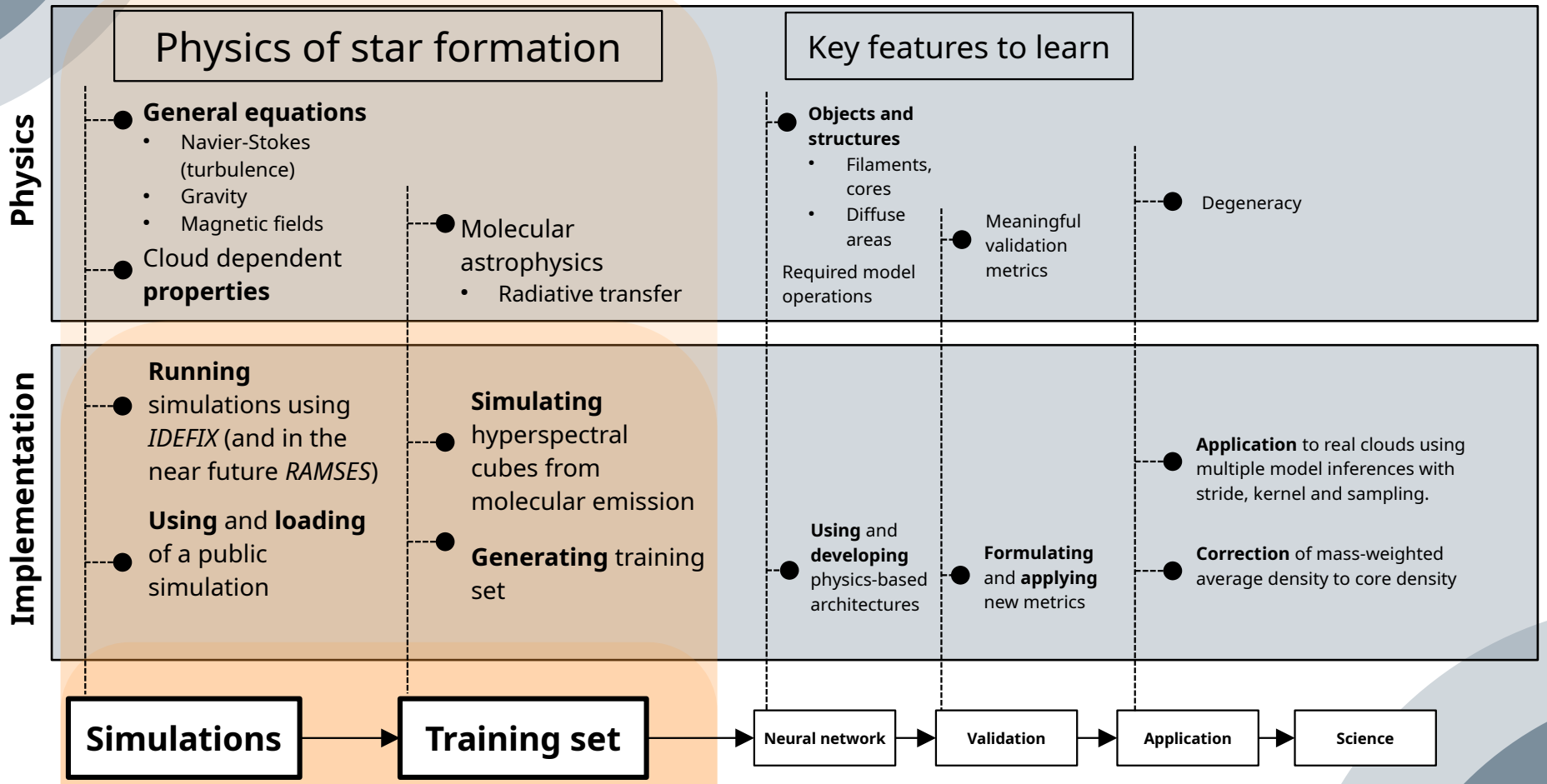


Conditional Invertible Neural Networks (cINNs) – II

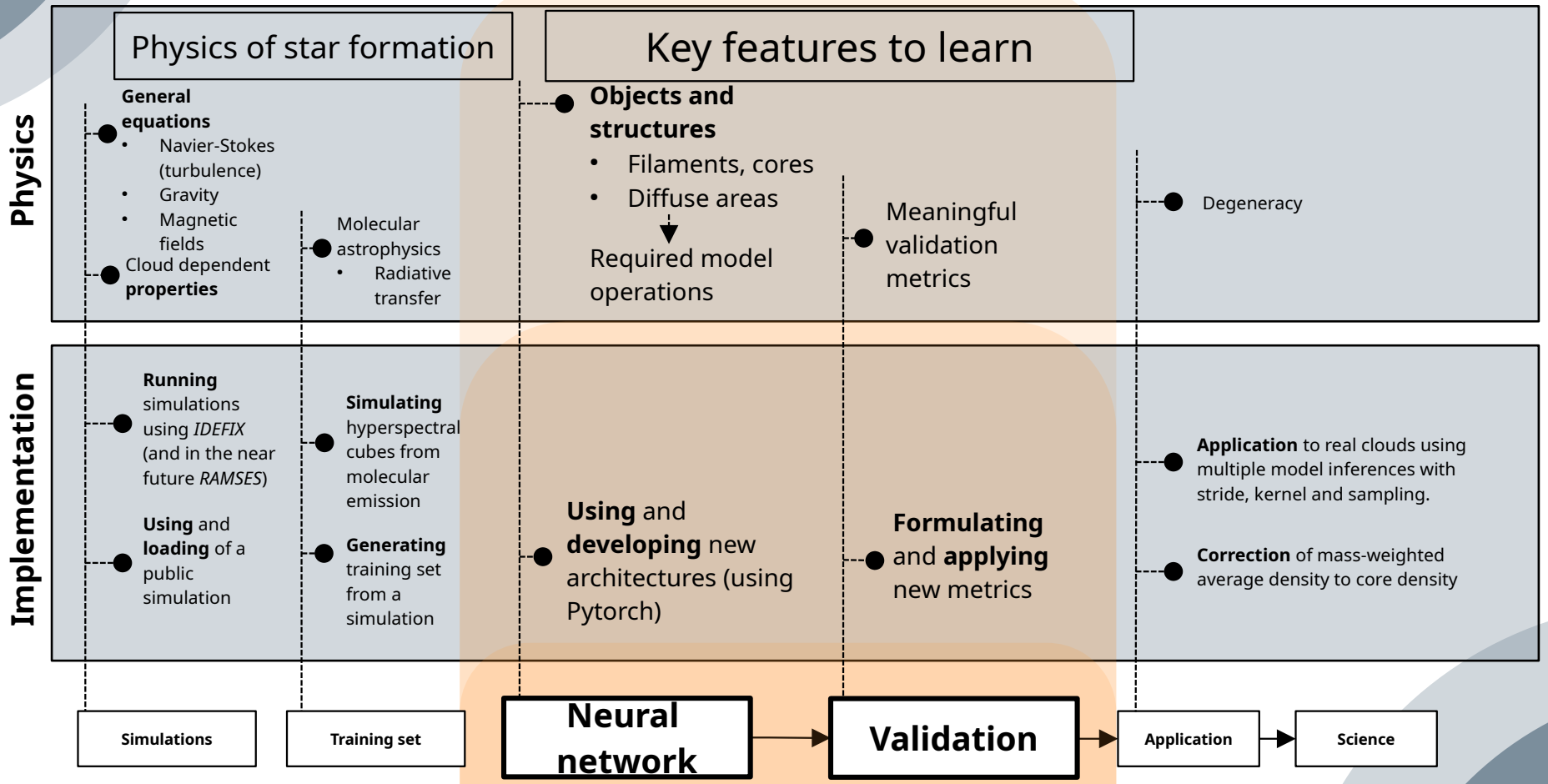
(cINNs, *Ardizzone et al. (2021) ; Rezende & Mohamed (2016)*)



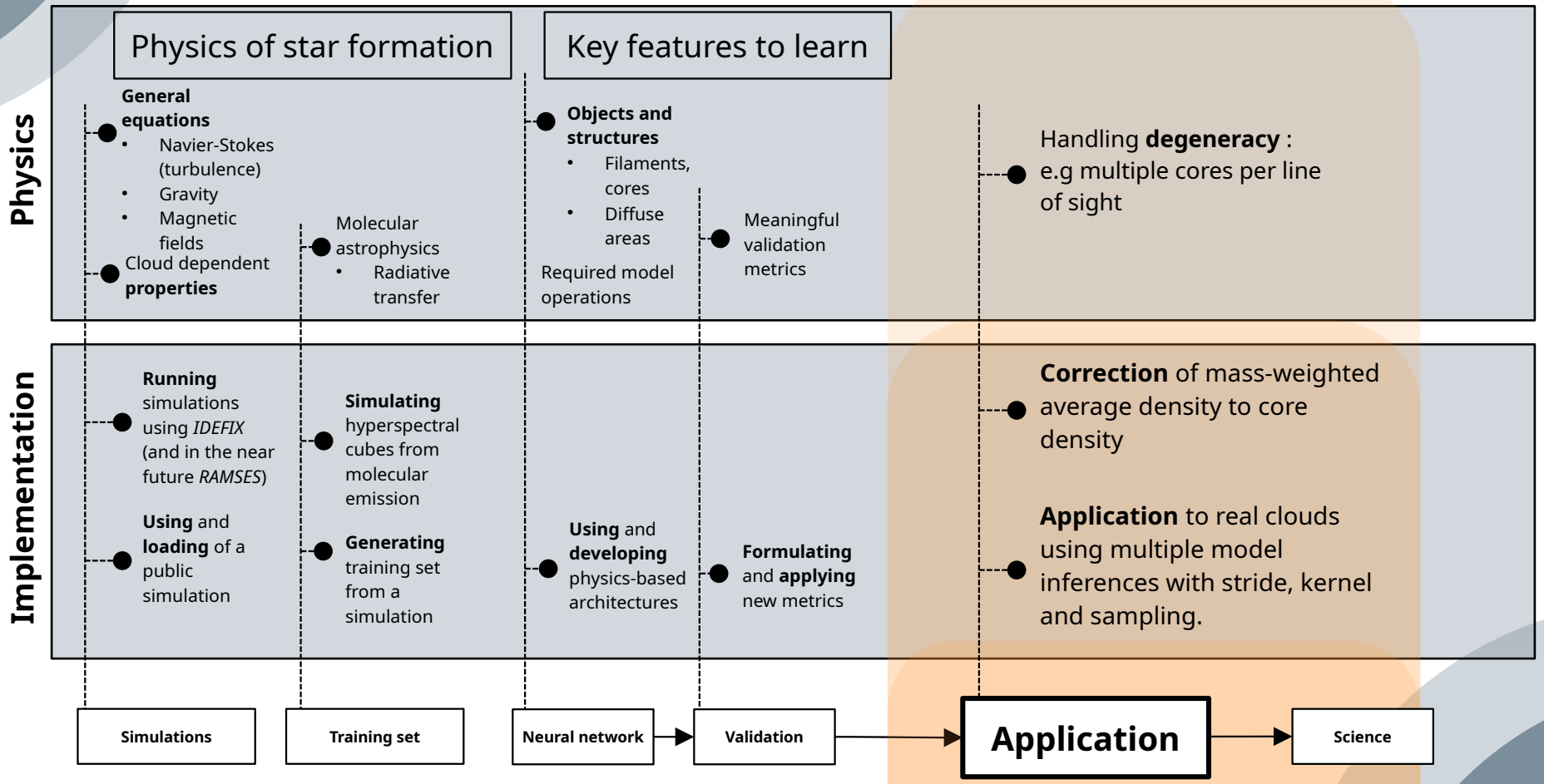
Pipeline - I


Physics of star formation
Key features to learn
Physics
Implementation
General equations
Navier-Stokes (turbulence)
Gravity
Magnetic fields
Cloud dependent properties
Molecular astrophysics
Radiative transfer
Objects and structures
Filaments, cores
Diffuse areas
Required model operations
Meaningful validation metrics
Degeneracy
Running simulations using IDEFIX (and in the near future RAMSES)
Using and loading of a public simulation
Simulating hyperspectral cubes from molecular emission
Generating training set
Using and developing physics-based architectures
Formulating and applying new metrics
Application to real clouds using multiple model inferences with stride, kernel and sampling.
Correction of mass-weighted average density to core density
Simulations
Training set
Neural network
Validation
Application
Science

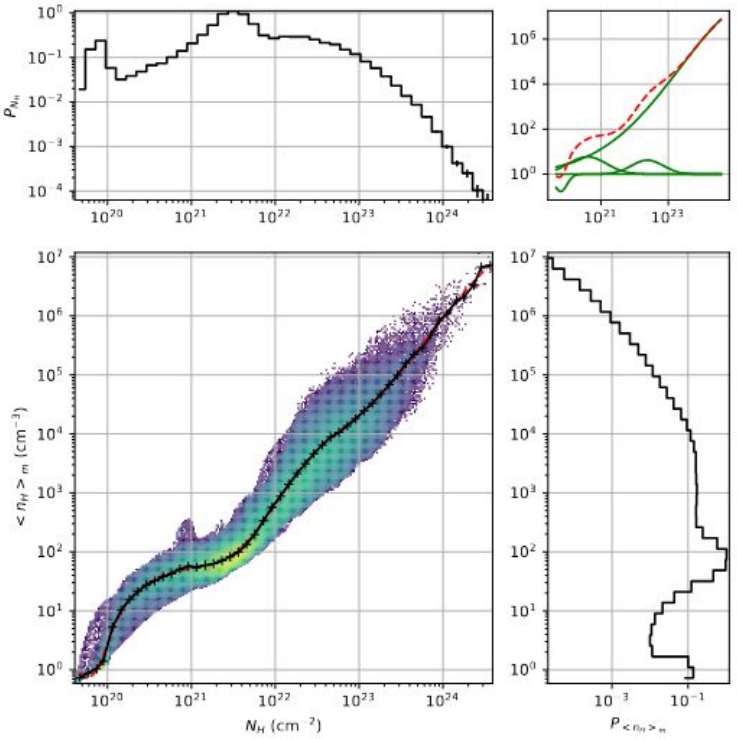
Pipeline - II

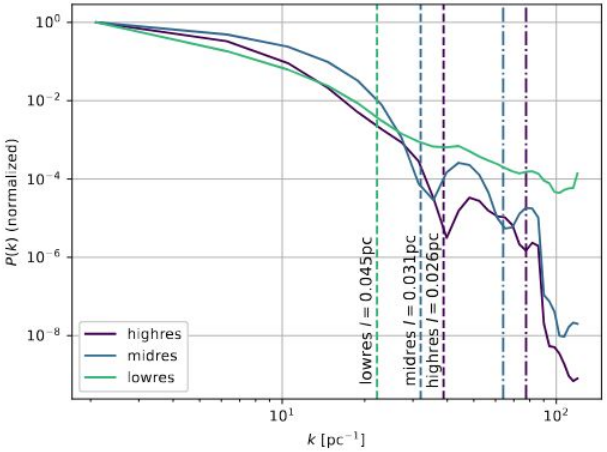

Physics of star formation
Key features to learn
Physics
Implementation
General equations
Navier-Stokes (turbulence)
Gravity
Magnetic fields
Cloud dependent properties
Molecular astrophysics
Radiative transfer
Objects and structures
Filaments, cores
Diffuse areas
Required model operations
Meaningful validation metrics
Degeneracy
Running simulations using IDEFIX (and in the near future RAMSES)
Using and loading of a public simulation
Simulating hyperspectral cubes from molecular emission
Generating training set from a simulation
Using and developing new architectures (using Pytorch)
Formulating and applying new metrics
Application to real clouds using multiple model inferences with stride, kernel and sampling.
Correction of mass-weighted average density to core density
Simulations
Training set
Neural network
Validation
Application
Science

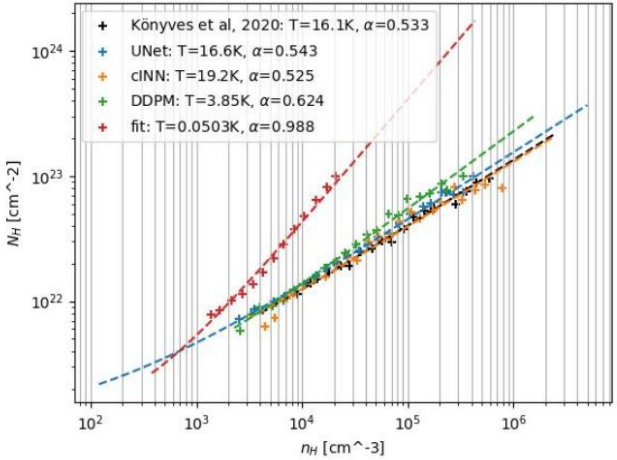
Pipeline - III



Extra figures







Correction

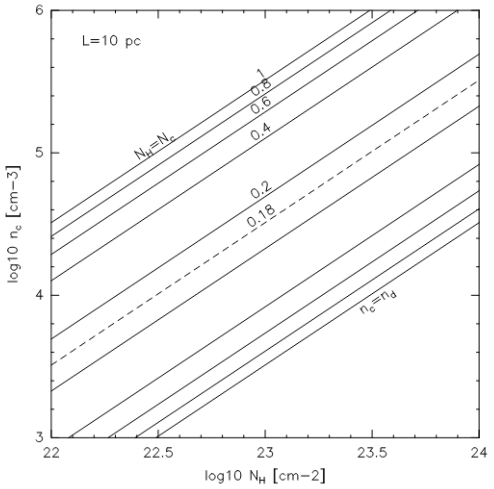
$$\frac{\langle n_{\rm H}\rangle_{\rm m}}{n_c} = \frac{n_d}{n_c} + \frac{n_c L_c}{N_{\rm H}}\left(1 - \frac{n_d}{n_c}\right).$$

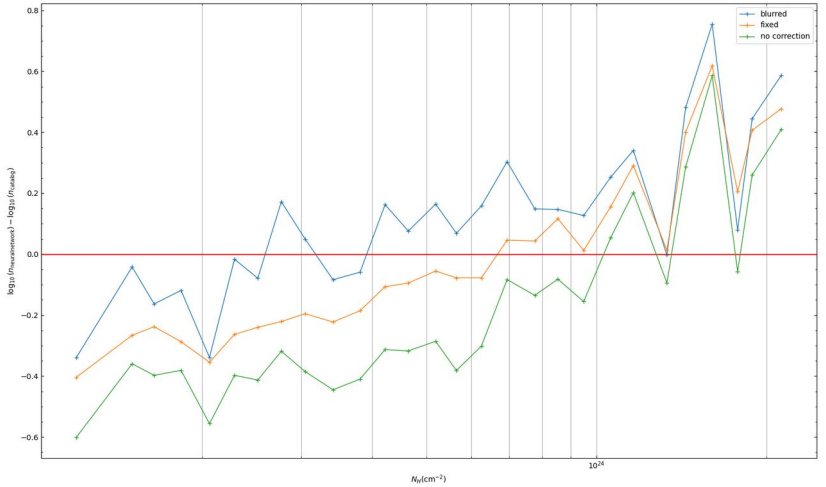
$$n_c = \frac{N_{\rm H}}{L_c + L_d}\left\{1 + \left[1 - \left(\frac{L_d}{L_c} + 1\right)\left(1 - \frac{\langle n_{\rm H}\rangle_{\rm m} L_d}{N_{\rm H}}\right)\right]^{1/2}\right\}, \qquad \text{(A.3)}$$

while if instead n_d is prescribed, then

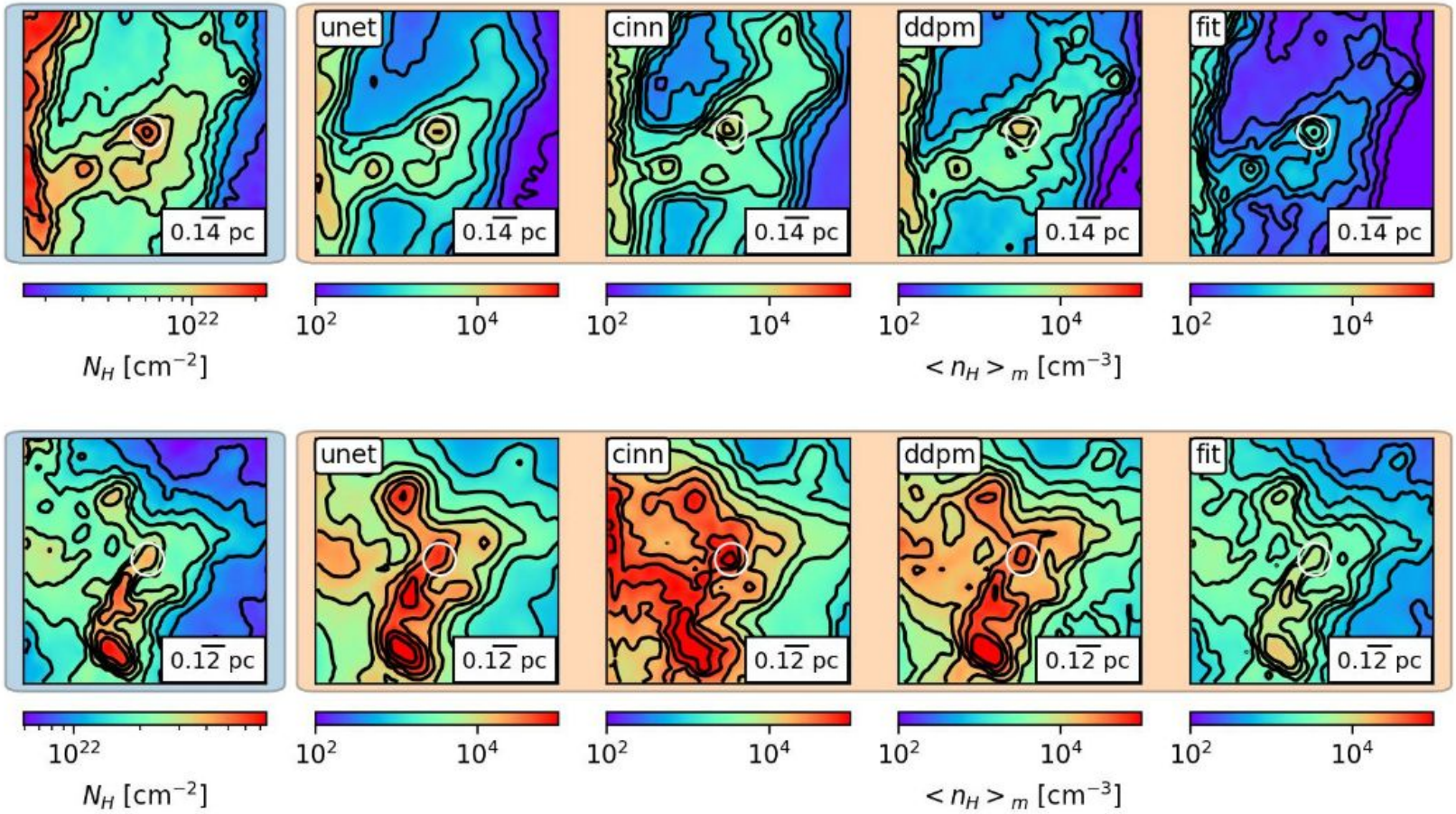
$$n_c = \frac{n_d}{2}\left(1 + \sqrt{1 - \frac{4N_{\rm H}}{L_c n_d}(1 - \frac{\langle n_{\rm H}\rangle_{\rm m}}{n_d})}\right) \qquad \text{(A.4)}$$

$$\lim_{L_d \to +\infty} n_c = \lim_{n_d \to 0} n_c = \sqrt{\frac{\langle n_{\rm H}\rangle_{\rm m} N_{\rm H}}{L_c}},$$

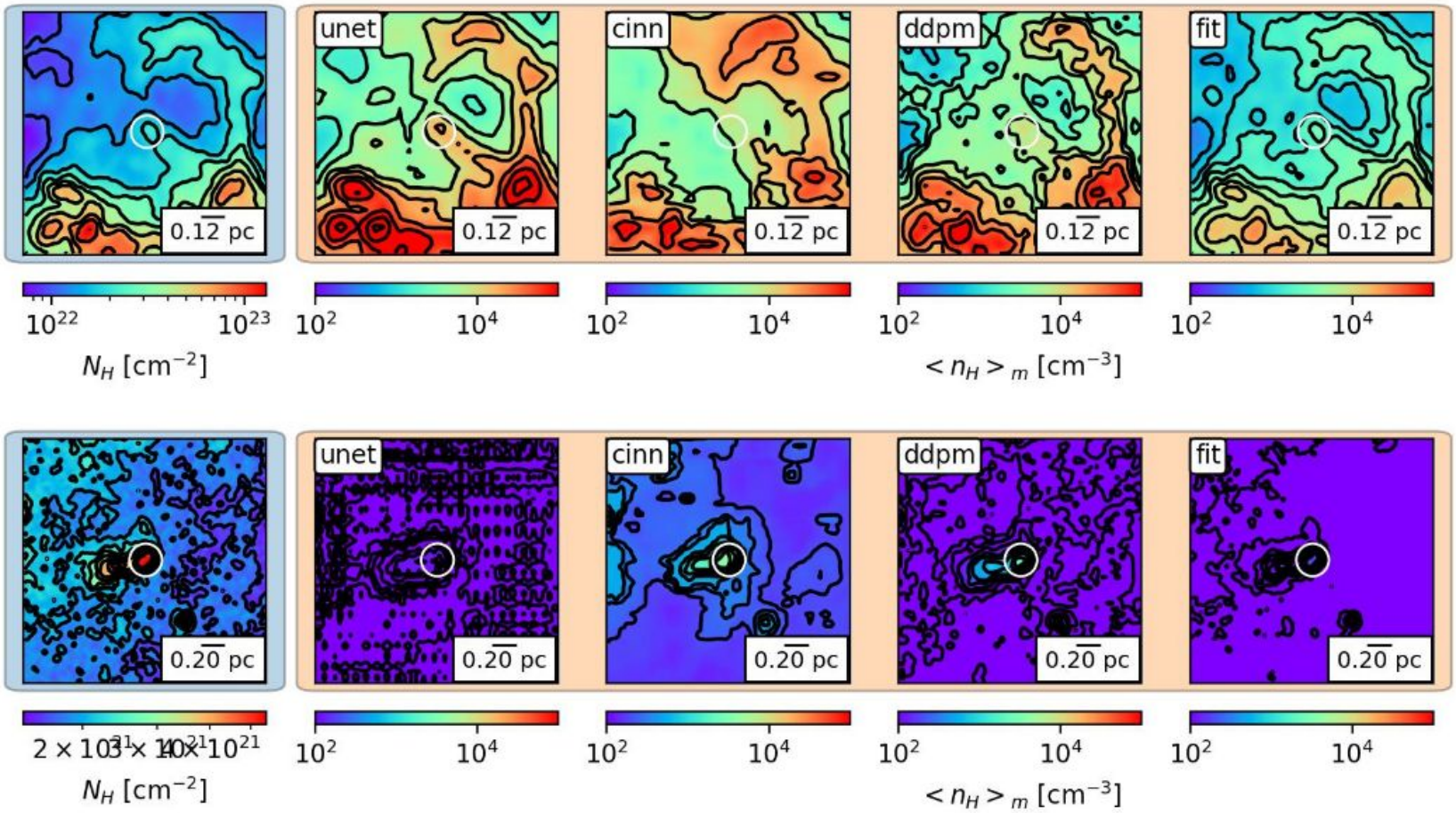




Cores



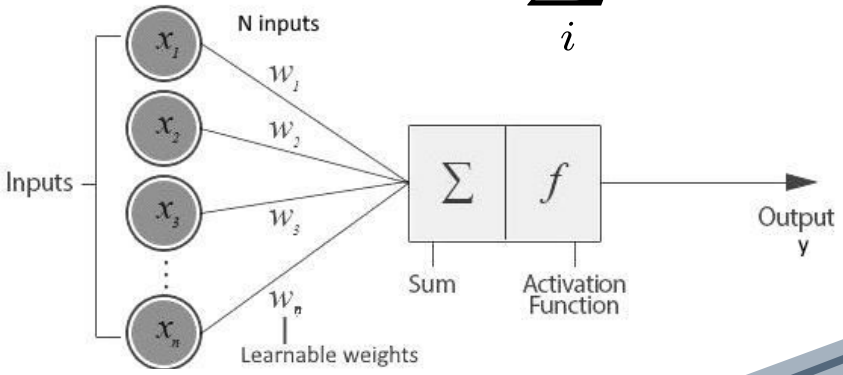
Cores

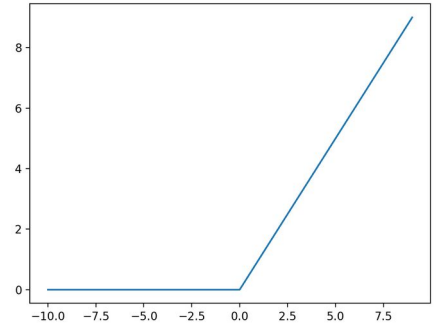

unet
cinn
ddpm
fit
0.12 pc
0.20 pc
10^{22}
10^{23}
N_H [cm^{-2}]
10^2
10^4
$< n_H >_m$ [cm^{-3}]

Neural networks

MLPs: Multi Layer perceptrons

$$y = f(\sum_{i}^{N} w_i x_i + b)$$





e.g $f(x) = \text{ReLU} = max(0, x)$

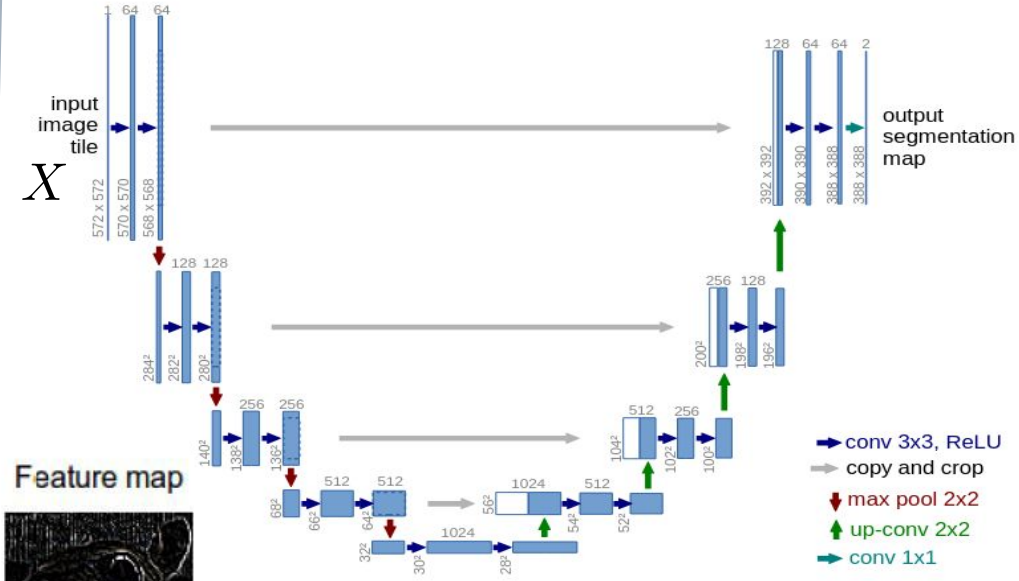
CNNs - UNet

$$Y_{i,j} = f(\sum_{m=1}^{M}\sum_{n=1}^{N} W_{m,n} \cdot X_{i+m,j+n} + b)$$

Width and Height of the used kernel

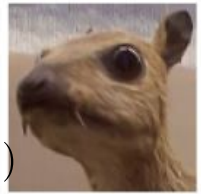
Pixel at (i,j) in the result image Y

Element (m,n) of the kernel



[Ronneberger, O., Fischer, P., & Brox, T. ; 2015]

Input image



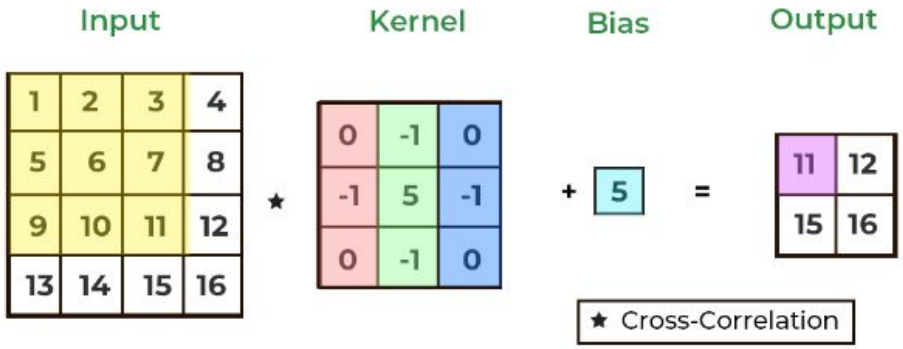
Convolution Kernel W

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

Feature map

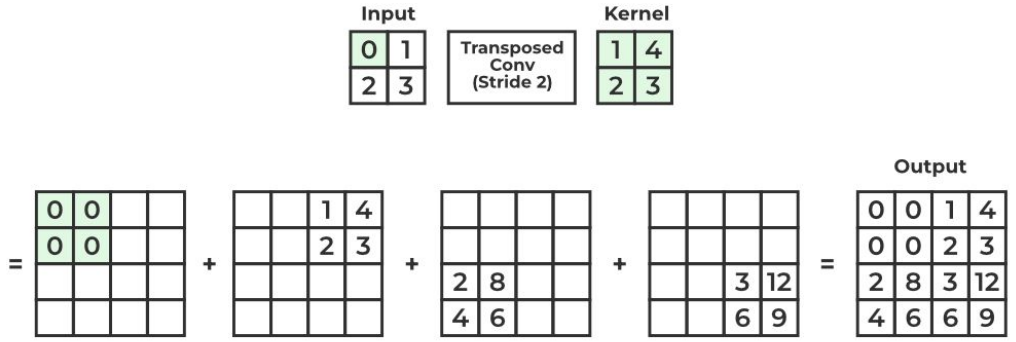
U-Net Operations

Convolution



1 X 0 + 5 X -1 + 9 X 0 + 2 X -1 + 6 X 5 + 10 X -1 + 3 X 0 + 7 X -1 + 11 X 0 + 5 = 11

Transp Convolution



Max pooling (2x2)