# Inferring Volume Density in Molecular Clouds Using Deep Learning

Intern : Zack Ribeiro[*]
Under the supervision of Pierre Hily-Blant[†]

May 27, 2025

**Abstract**

**Context.** Molecular clouds, seen as dark patches on the sky, are complex 3D structures forming filaments and dense cores where stars are born. Yet, as Earth-bound observers, we only access 2D, quasi-static projections. Reconstructing their 3D structure is crucial for understanding star formation, turbulence, and chemistry, but doing so from a single 2D map is highly non-invertible.

**Aims.** This work explores deep learning, particularly neural networks, to infer physical quantities such as the average volume density along the line of sight. It also initiates the use of architectures capable of handling PPV cubes, which offer additional kinematic information.

**Methods.** Neural networks are trained on synthetic data to predict volume density from column density and extended to use molecular emission maps (e.g., from $^{13}$CO).

**Results.** We developed models capable of predicting average volume density and explored new multi-input architectures. We also implemented Kolmogorov-Arnold networks and began developing POLARIS[1][2], a software tool for future experimentation.

**Conclusion.** Though still exploratory, these results mark an early step toward reconstructing the 3D structure of molecular clouds using deep learning.

*This document was written as part of a Master 2 internship carried out in 2025 at IPAG (Institut de Planétologie et d'Astrophysique de Grenoble), France.*

[*]zackribeiro@univ-grenoble-alpes.fr
[†]pierre.hily-blant@univ-grenoble-alpes.fr
[1]Software code available here: https://github.com/EazyEnder/POLARIS
[2]Python code available here: https://github.com/EazyEnder/POLARIScore

# Contents

# Part I
# Context

The subject of this document is the prediction of a physical quantity using a machine learning model. To facilitate its reading, it is therefore essential to have some basic understanding of both the underlying physics of the system and the principles of neural networks. In addition to the global context provided here, relevant concepts and reminders will be given in the opening section of each part, to ensure a smooth and coherent progression alongside the application.

## Molecular clouds

The objects studied in this work are molecular clouds. These clouds are not static, they have both a history and a future. To understand why we study molecular clouds, it is therefore essential to consider their evolution.

As will become clear throughout this document, physics plays a central role in every stage of the model development process, from the design of the architecture, where a link must be established between the operations performed and the underlying physics, to the training, validation (i.e., error evaluation), and application of the model to real observations. It is therefore not sufficient to build a model blindly and expect it to perform well; physical understanding is required at every step. Moreover, without any underlying physical structure, a model would have nothing meaningful to learn, and such an approach would be fundamentally misguided.

Here are several papers that provide clear and comprehensive introductions to key topics such as star formation and interstellar turbulence: Girichidis et al. (2020), P. Hennebelle and M. Grudić (2024), Hopkins (2013), and Guszejnov and Hopkins (2016).

### Interstellar medium

The interstellar medium (ISM) is the matter that exists in the space between the stars within a galaxy. Far from being empty, this medium is composed of gas (both atomic and molecular), dust grains, cosmic rays, and magnetic fields. Though tenuous in comparison to Earth's atmosphere, with typical densities ranging from less than one to several thousand particles per cubic centimeter, the ISM plays a fundamental role in galactic evolution. It is the reservoir of material from which stars are born and into which they return mass and energy at the end of their life cycles, making it both the cradle and the graveyard of stars.

The ISM is not uniform but highly structured, exhibiting variations in temperature, density, ionization state, and chemical composition. These different conditions give rise to a classification into several distinct phases, which coexist in a complex, dynamically interacting balance. The two main neutral atomic phases are the Warm Neutral Medium (WNM) and the Cold Neutral Medium (CNM), while the molecular phase marks the densest regions of the ISM where star formation can occur. These phases are not static. The WNM can cool and condense into the CNM, which under further compression and shielding can transition into a molecular cloud. Within these molecular clouds, regions may collapse gravitationally to form new stars, which in turn inject energy, momentum, and material back into the ISM through radiation, stellar winds, and supernova explosions. This feedback can heat and disperse the surrounding gas, transforming it once again into WNM, thereby completing a cyclical process that drives the continuous evolution of the interstellar medium.

| Component | Temperature (K) | Density ($cm^{-3}$) |
|---|---|---|
| Molecular gas | $10 - 20$ | $> 10^2$ |
| Cold neutral medium (CNM) | $50 - 100$ | $20 - 50$ |
| Warm neutral medium (WNM) | $10^4$ | $10^{-1}$ |
| Warm ionized medium (WIM) | $\sim 8000$ | $10^{-1}$ |
| Hot ionized medium (HIM) | $\sim 10^6$ | $10^{-2}$ |

Table 1: Phases of the ISM

### Physics behind

Turbulence, bases de la formation stellaire: temps caractéristiques, viriel -> jeans...

### Dense cores

Avec CMF -> IMF...

## Machine learning

Machine learning, a subset of artificial intelligence, focuses on algorithms that enable computers to learn patterns from data and make decisions without being explicitly programmed. The foundational ideas trace back to the 1950s, notably with Alan Turing's work on machine intelligence and Arthur Samuel's pioneering research in self learning systems.

Neural networks, inspired by the structure and function of the human brain, were first conceptualized in 1943 by McCulloch and Pitts (McCulloch and Pitts 1943). The introduction of the perceptron by Rosenblatt in 1958 (Rosenblatt 1958) marked an early attempt at mimicking neural processes. However, due to computational limitations and theoretical criticisms, interest waned during the 1970s. The resurgence came in the 1980s with the development of the backpropagation algorithm (Rumelhart, Hinton, and Williams 1986), enabling the training of multi-layer networks.

The modern era of machine learning was catalyzed by advances in computing power, the availability of large datasets, and algorithmic innovations, leading to the rise of deep learning in the 2010s. Architectures such as convolutional and recurrent neural networks have since achieved state-of-the-art performance in domains including computer vision, natural language processing, and bioinformatics.

## Multi Layer Perceptrons

The fundamental building block of neural networks is the perceptron (Rosenblatt 1958). Analogous to a biological neuron and its synapses, the perceptron receives $N$ inputs $x_i$, each of which is weighted by a corresponding learnable parameter $w_i$. It computes a weighted sum of the inputs, which is then passed through a nonlinear function known as the activation function. This nonlinearity is what gives the perceptron its expressive power.

Mathematically, the output $y$ of a perceptron is given by:

$$y = f(\sum_{i}^{N} w_i x_i + b) \tag{1}$$

where $f$ is the activation function, and $b$ is the bias term, which allows the model to shift the activation threshold.

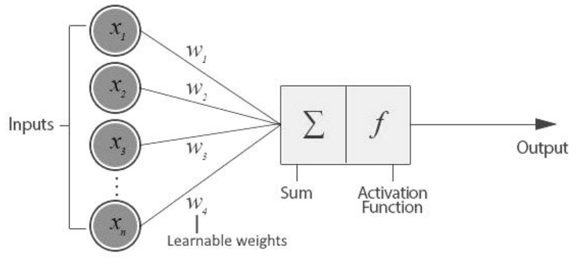


Figure 1: Perceptron

What gives Multilayer Perceptrons (MLPs) their expressive power are several key elements. First, the activation function, commonly a sigmoid, or more recently, the ReLU, introduces non-linearity into the model, allowing it to approximate complex functions beyond simple linear mappings.

Second, the number of neurons in each layer can be arbitrarily large, and more importantly, the number of trainable parameters, which corresponds to the number of connections (weights), can reach thousands or even millions. This high parameter count gives the model substantial flexibility and representational capacity.

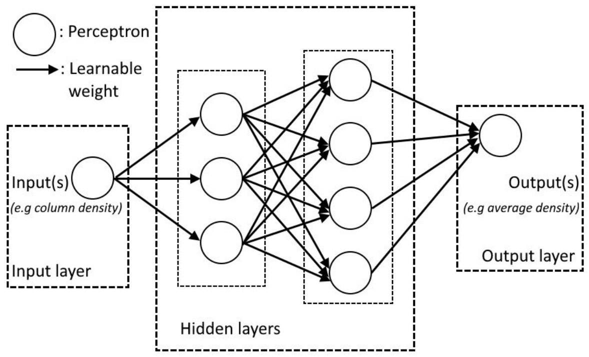


Figure 2: Multi Layer Perceptrons (MLPs)

However, this very richness in parameters was historically a major obstacle. Training such large networks was computationally expensive and often infeasible. Only in recent years, thanks to the development of more efficient algorithms, optimization techniques, and the rise of powerful hardware (e.g., GPUs), has it become possible to train deep networks at scale. These advancements have led to the resurgence and widespread application of neural networks in numerous fields.

## Convolutional Neural Network

## Denoising Diffusion Probabilistic Models

Part II

# From the column density to the volume density using neural networks

## 1 Introduction

The first part of this work focuses on predicting the average density $< n >$ along the line of sight (l.o.s.) from the column density $N_C = \int_L n(x, y, z)dz$ where $z$ is the line of sight axis. This remains a 2D-to-2D prediction, meaning we are estimating a new 2D map from an existing 2D map. This objective is, in principle, more achievable than recovering the full 3D structure of a molecular cloud, while still offering valuable scientific insights, such as highlighting dense cores.

Recently, several new methods have emerged to estimate this average density (Xu et al. 2023; Orkisz and Kainulainen 2024; Gaches and M. Y. Grudić 2024), including one that leverages machine learning and a diffusion model (Xu et al. 2023). In this work, we aim to reconstruct a neural network model capable of this task, and to explore various potential improvements, while intentionally avoiding diffusion models at this stage.

The goal is to proceed step by step, to build a clean and understandable foundation, to learn and analyze the process, and to prepare for more complex architectures in the future.

There are few examples of why a 3D reconstruction of molecular clouds would be highly valuable:

- Lifting projection degeneracies: For instance, multiple dense cores may overlap along the same line of sight. A 3D view would allow us to disentangle them and more accurately estimate their individual masses, essential for determining the dense core mass function.
- Estimating the star formation rate (SFR).
- Astrochemistry: The UV and cosmic-ray ionization rates depend strongly on the local depth within the cloud, i.e., the effective column density shielding a given position $(x, y, z)$. A 3D reconstruction would provide a more realistic view of the material distribution, directly impacting chemical modeling.
- Studying turbulent processes: Understanding the 3D structure of turbulence, including its anisotropy and scaling, requires volumetric data. A 2D projection washes out key features of the velocity and density fields, limiting insight into the physics of turbulence in the interstellar medium.

### 1.1 Neural networks

Neural networks are machine learning models that process data through layers of interconnected neurons. Two key types are Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs).

- MLPs are fully connected networks that apply weighted sums and non-linear activations. They work well for structured data but struggle with spatial dependencies.
- CNNs are designed for spatial data like images, using convolutional layers to extract features and pooling layers to reduce dimensionality (see Appendix A).

#### 1.1.1 U-Net

U-Net (Ronneberger, Fischer, and Brox 2015) is a specialized convolutional neural network architecture primarily designed for biomedical image segmentation. It features a symmetric encoder-decoder structure, where the encoder captures high-level features through convolution and max pooling operations, while the decoder gradually reconstructs the image using upsampling layers. A key feature of U-Net is its skip connections, which allow direct information flow between corresponding encoder and decoder layers, preserving spatial details lost during downsampling. (See Figure 3)

Even though it was originally designed for image segmentation, its ability to capture spatial information makes it suitable for our study of molecular clouds.

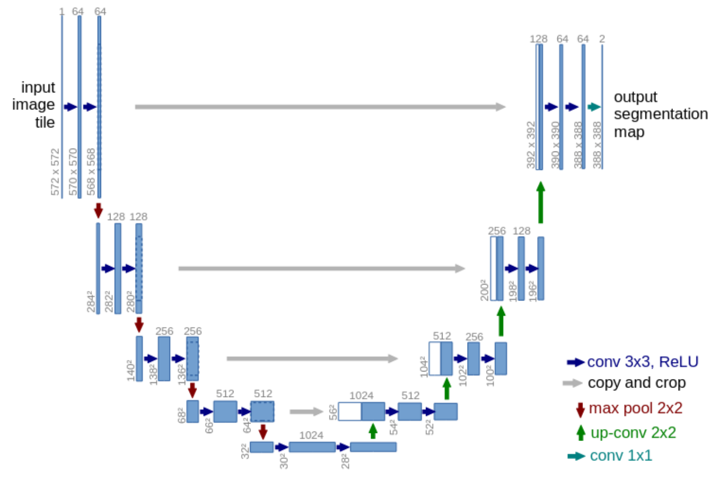
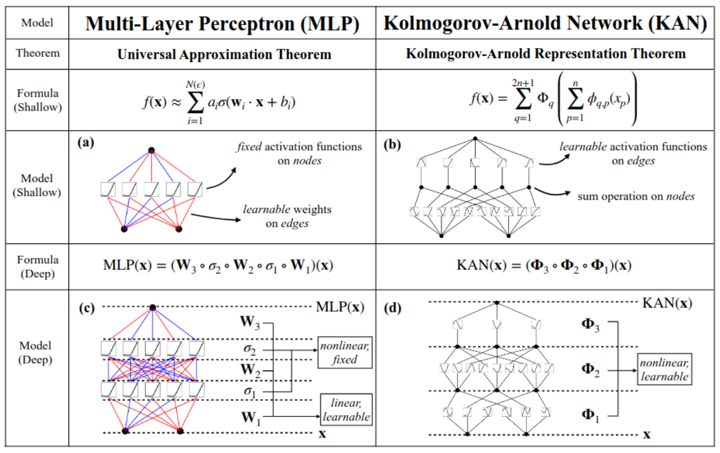


Figure 3: U-net architecture Ronneberger, Fischer, and Brox 2015 (example for 32x32 pixels in the lowest resolution). Each bluebox corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. Whiteboxes represent copied feature maps. The arrows denote the different operations

#### 1.1.2 KANs

Kolmogorov–Arnold Networks (KANs) (Liu et al. 2024) are a novel class of neural networks inspired by the Kolmogorov–Arnold representation theorem. Unlike traditional deep learning architectures, KANs replace fixed activation functions with learnable, parameterized univariate functions (see Figure 4 ).

This design provides greater flexibility in function approximation while maintaining interpretability. KANs have demonstrated strong potential in various applications, including scientific computing and physics-informed learning, where explainability and efficient function representation are essential. However, they are more computationally expensive, and their interpretability is lost when incorporated into networks such as U-Net.



| Model | Multi-Layer Perceptron (MLP) | Kolmogorov-Arnold Network (KAN) |
|---|---|---|
| Theorem | Universal Approximation Theorem | Kolmogorov-Arnold Representation Theorem |
| Formula (Shallow) | $f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$ | $f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$ |
| Model (Shallow) | (a) *fixed* activation functions on *nodes*; *learnable* weights on *edges* | (b) *learnable* activation functions on *edges*; sum operation on *nodes* |
| Formula (Deep) | $\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$ | $\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$ |
| Model (Deep) | (c) | (d) |

Figure 4: Multi-Layer Perceptrons (MLPs) vs. Kolmogorov-Arnold Networks (KANs)

## 2 Method

### 2.1 Datasets

In order to train our future model, two datasets are required: one for the training phase and the other for testing, i.e., the validation phase. The preparation of these datasets is a crucial factor in the model's performance.

These datasets are created using MHD simulations of molecular clouds: ORION (Ntormousi and Patrick Hennebelle 2019). These simulations model MHD turbulence and have a size of 66.1 pc with an initial resolution of 0.01 pc, reaching a few hundred AU using an AMR grid. The simulations were performed using RAMSES (Teyssier 2002), with self-gravity applied, star-formation treated using Lagrangian sink particles and ideal MHD resolved and no chemistry. However, to facilitate data manipulation, an initial approach consists of using a $512^3$ density cube. This cube has a width of 25.8 pc, centered at the simulation's midpoint, with a resolution of 0.05 pc per pixel.

To increase the amount of training data, projections along the three axes are used. In other words, we generate three column density images by summing the densities along different axes as shown in Figure 5.

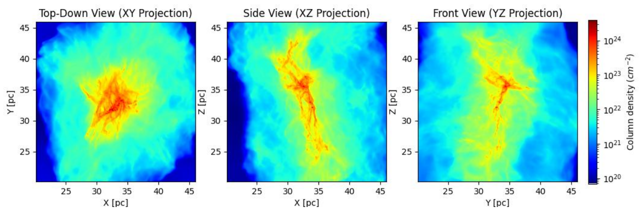


Figure 5: Column density for 3 axes of the same molecular cloud from ORIONNtormousi and Patrick Hennebelle 2019 simulation, with low magnetic field

Then, images with a resolution of 128×128 pixels and a physical size of 6.4 pc are randomly cropped from the three faces. An image is retained in the dataset if it meets several conditions:

- It is spatially distant enough from an already explored region, meaning it does not overlap with an area covered by an existing image.
- It does not contain a disproportionately large fraction of very low-density regions, i.e., the edges of the simulation.
- It exhibits significant density variations within the explored region. A score is defined based on the smoothness of the image using the Laplacian of the density, favoring regions with filaments over those with nearly homogeneous density. (See appendix B)

The generated dataset is then randomly shuffled and split between training set (80%) and validation set (20%). For example, using only the simulation shown in Figure 5, we obtain a total of 37 covered regions, each represented by a pair of images: (column density, volume density). These are distributed as 29 training (pair of) images (78%) and 8 validation (pair of) images (22%). In subsection 3.4, a model will be trained with a larger dataset of 100 covered regions generated using the same ORION simulation but with a cube of $1024^3$ and so a resolution of 0.026pc per pixel.

In this part, our goal is not to reconstruct the full density distribution along the line of sight, but rather to determine a single representative value for each region. There are several ways to compute such an average density, including:

$$< n >_V = \frac{\sum_i^{N_{cells}} \Delta V_i n_{H,i}}{\sum_i^{N_{cells}} \Delta V_i} \overset{\Delta V = \text{cst}}{=} \frac{1}{N_{cells}} \sum_i^{N_{cells}} n_{H,i} \quad (2)$$

$\Delta V_i$ and $n_i$ are respectively the differential volume element and the number density of the ith cell. For an uniform grid, as in our initial approch, $\Delta V$ is constant.

Equation 2 is the volume-weighted density and has the problem that to get this, one need the cloud length.For our simulation, if we compute the volume-weighted density, we find that it simplifies to the column density $N_C$ divided by the box length $L$:

$$N_C = \sum_i n_{H,i} (\Delta V_i)^{\frac{1}{3}} = N_{cells} (\Delta V)^{\frac{1}{3}} \sum_i n_{H,i} = L \sum_i n_{H,i}$$

where we assume a uniform grid, but the principle remains valid even for a non-uniform grid. From this, we obtain:

$$<n>_V = \frac{1}{N_{cells}} \sum_{i}^{N_{cells}} n_{H,i} = \frac{N_C}{L}$$

This shows that the volume-weighted density computed on the simulation is directly proportional to the column density.

Another density which can be used is the mass-weighted average number density:

$$<n>_M = \frac{\sum_i^{N_{cells}} m_i n_{H,i}}{\sum_i^{N_{cells}} m_i} \overset{\Delta V = \text{cst}}{=} \frac{\sum_i^{N_{cells}} n_{H,i}^2}{\sum_i^{N_{cells}} n_{H,i}} \quad (3)$$

This density highlights high density components along the line of sight. One can also want to use the max density along the l.o.s: $n_{\max} = \max(n_i)$ but this is much harder .

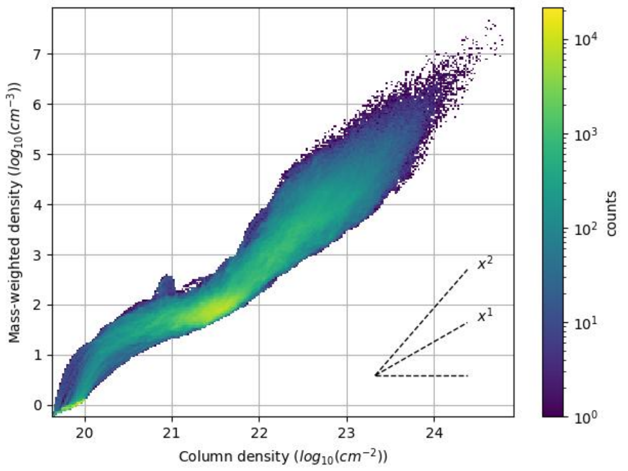


Figure 6: Density correlation of ORION

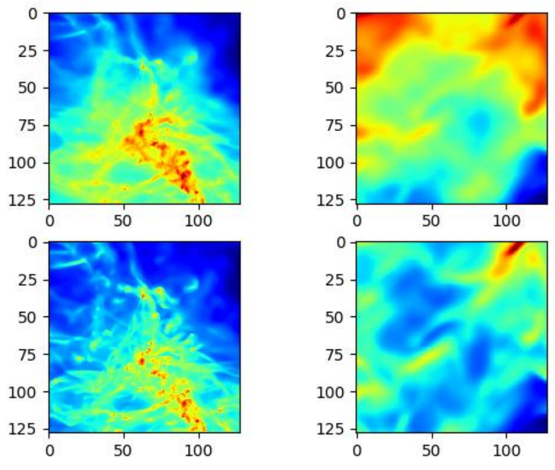


Figure 7: Two examples from the ORION training dataset. Top row is the column density, bottom row is the mass-weighted density. Left is a dense region, Right is a diffuse region.

## 2.2 Network architecture

The default network architecture used in this part is a U-Net architecture with 4 layers, a first convolution layer which go from 1 to 64 features maps and attention blocks (Oktay et al. 2018). The implementation of KAN in the architecture can occurs at the output convolution layer, where the KAN layer replaces the classic convolution layer (named K-Net if that's the case), thanks to the work of X. Gao[3] on how to implement a convolution layer using KAN. The implementation is done in Python using PyTorch. It is also possible to only use KAN convolution layers instead of the classic ones (U-Kan).

## 2.3 Training

Training a model is a crucial step in achieving good performance. There are multiple choices that we, as users, need to make to obtain optimal results.

- We want the model, with $N$ free parameters (weights), to converge to a global minimum of the loss function rather than getting stuck in a local well. Therefore, the choice of optimizer, which performs backward error propagation (adjusting the model's weights), is important. Here, we use ADAM (Kingma and Ba 2017) or stochastic gradient descent (SGD).

- Loss function: A well-chosen loss function is essential for accurately representing the problem. Here, we generally use MSELoss (see eq: 4), but other loss methods can be considered.

- Learning rate: The learning rate is another critical factor—it determines the step size the optimizer takes in the loss function during each update (epoch). Ideally, it should decrease when the loss reaches a plateau. To manage this, we use a scheduler.

Each epoch, i.e., learning step, the model processes all the training batch[4] column density images and predicts their mass-weighted density.

A way to artificially increase the size of the training batch is to apply random transformations to the images at each epoch (Yang et al. 2023), such as random rotations and random vertical or horizontal flips. Afterward, the prediction loss is computed using the chosen loss method, and backward error propagation is performed. Then, a new epoch begins, repeating the same steps.

[3]https://github.com/XiangboGaoBarry/KA-Conv
[4]A batch is a set of paired images: column density/input & volume density/target

## 3 Model performance

### 3.1 U-Net global performance

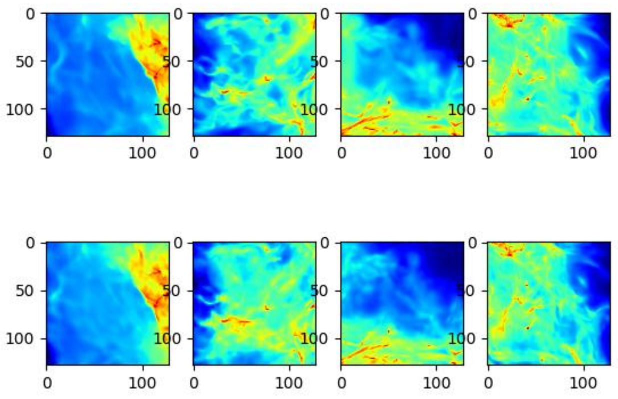


Figure 8: U-Net prediction example, top row is the true mass-weighted density, bottom row is the predicted mass-weighted density.

There are many ways to view if a model works or not. First, there are the losses, which are computed during training. Traditionally for regression problems, we use MSELoss ("Mean Squared Error") :

$$\text{MSELoss} = \mathbb{E}((\text{prediction} - \text{target})^2) \tag{4}$$

In our case, this shows the mean squared error in log10 for the batch. In Figure 9, the validation loss converges to ~0.25 and doesn't go lower. This could mean that the model cannot generalize more the training data due to a network that is too shallow or insufficient images in the training set.

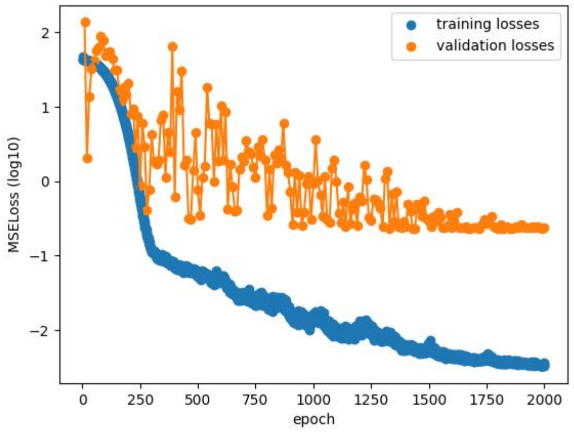


Figure 9: U-Net training losses

However, MSELoss is an average of all the batch errors. To analyze how the error varies as a function of density, we plot the correlation between the target and the predictions made by the trained model on the validation set (Figure 10). An ideal result would be a correlation that is infinitely narrow along the $y = x$ line.

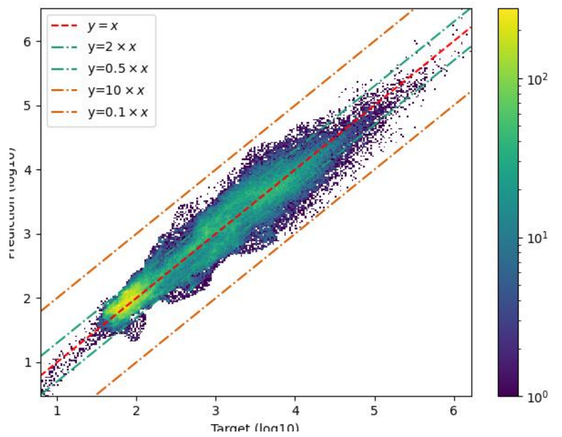


Figure 10: U-Net Prediction correlation

Or, instead, we can directly analyze the residuals. This allows us to determine whether the model overestimates or underestimates lower or higher densities. Figure 11 shows that U-Net tends to underestimate higher densities.

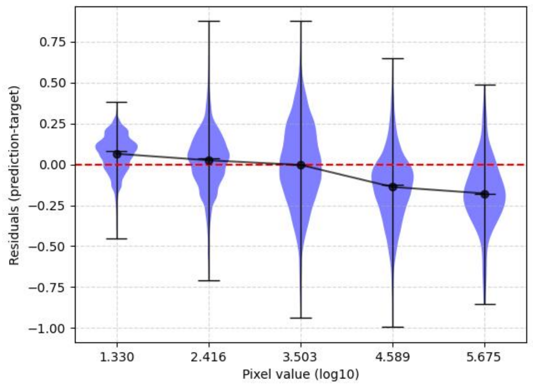


Figure 11: U-Net residuals , Pixel value means in our case the mass-weighted density, blue is the residuals distribution in the density bin with black lines for min,max and mean of the residuals.

Molecular clouds form spatial structures such as cores and filaments, so it is also important to assess whether the spatial error is "homogeneous" (i.e., does not exhibit obvious structures—see Figure 12 for example) or if the model struggles to predict certain features, such as densities within a filament. If this is the case, an alternative loss function that emphasizes spatial structures could improve the model's performance.

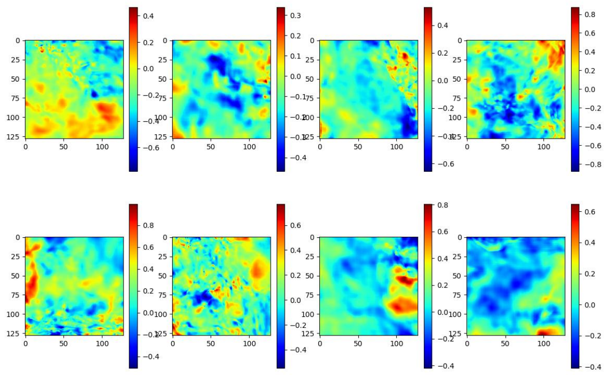


Figure 12: U-Net Validation spatial errors

Finally, we can also define an error metric that we will refer to as "accuracy." The accuracy of the predicted image is the ratio of correct predictions to the total number of elements (i.e., pixels, here $128 \times 128$). A predicted pixel is considered correct if$|\text{target} - \text{prediction}| < \sigma$ ,where $\sigma$ represents the allowed error in the prediction in log10.

$$Acc_i = \frac{\sum_i^{N_{pixels}} N(|\text{target} - \text{prediction}| < \sigma)}{N_{pixels}} \tag{5}$$

And for the entire validation batch, the accuracy is computed as the average accuracy across all images in the batch.

$$Acc =< Acc_i >_{\text{batch}}= \frac{1}{B} \sum_i^B Acc_i \tag{6}$$

Where $B$ is the batch size,i.e the number of images in the batch.

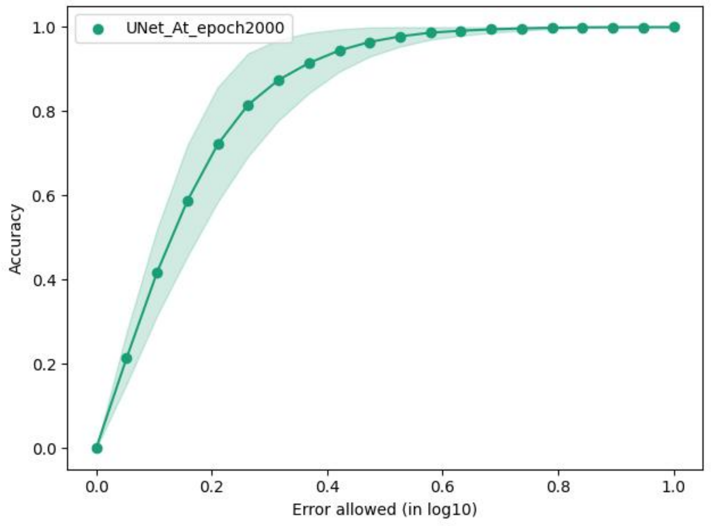


Figure 13: U-Net Accuracy. Transparent area is the standard deviation of the accuracy over the batch images.

## 3.2 U-Net complexity

Our problem is a regression problem, similar to function fitting. As with function fitting, our model has $N$ free parameters. A higher $N$ increases the likelihood of capturing important features, whereas a lower $N$ may lead to missing details such as high-frequency variations or structural connections.

In the U-Net architecture, the number of free parameters is influenced by:

- The number of layers: This determines how many times the network applies pooling, affecting the depth of feature extraction.

- The base filters: The first convolutional layer creates $n$ feature maps from the input (column density), i.e. $1 \to n$. Then, each subsequent layer and its convolutional operations generally double this number. For example, a convolutional layer in the $i$th U-Net encoder stage transforms $n \times 2^{i-1}$ feature maps into $n \times 2^i$.

So, when the number of base filters or the number of layers increases, we say that the model complexity increases. And generally, as the complexity increases, the memory usage, training time, and the number of epochs required to converge to a validation plateau will also be higher.

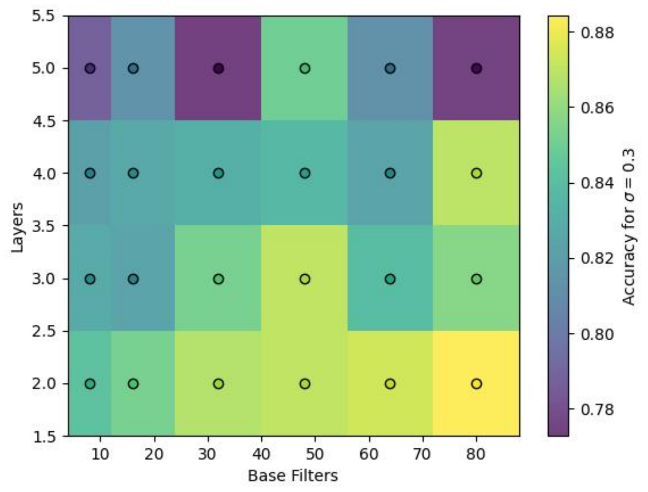


Figure 14: U-Net complexity map . Each point is a trained U-Net with attention blocks and with X Base Filters and Y Layers. Yellow means higher accuracy and purple lower accuracy.

Figure 14 shows the variation of the model accuracy $Acc$ with the number of base filters and the number of layers. This highlights that a model with higher complexity does not always guarantee better accuracy, probably because of the lack of training data and longer convergence time. However, Figure 14 is not sufficient because the accuracy, as defined by 5 can be higher even for a blurred prediction. This is illustrated in Figure 15, where a model with fewer layers fails to capture high-frequency features (sharp edges), resulting in a blurry output.

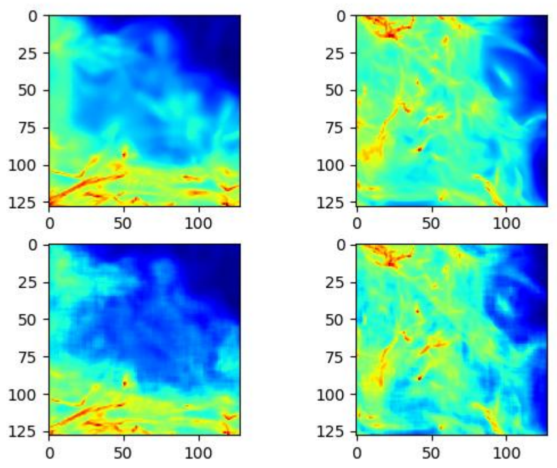


Figure 15: U-Net 2 layers with 80 initial features maps, validation images. Top row is the true mass-weighted density, bottom row is the predicted mass-weighted density.

Note that in U-Net, each layer typically performs two convolution operations rather than just one, as shown in Figure 3. This further increases the model complexity. However, in our case, using two convolutions per block does not lead to better performance (see figure 16). Therefore, our default model is designed with just one convolution operation per block to reduce computational cost while maintaining accuracy. That said, our implementation is flexible, making it easy to modify the architecture and use alternative blocks, such as double convolution per block, if needed.

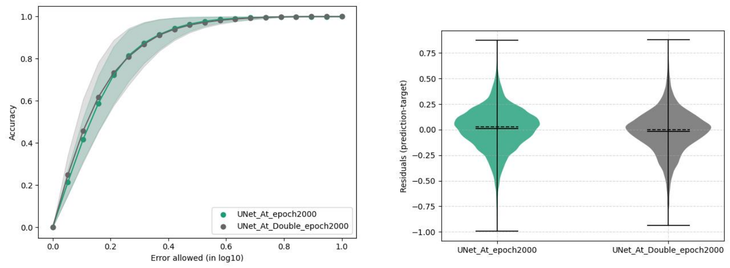


Figure 16: U-Net Accuracy and Residuals for double conv per block (black) and for one conv per block (green)

## 3.3 U-Net with KAN parts

Kolmogorov Arnold Networks(Liu et al. 2024) are news and promising alternatives of MLPs. They replace fixed activation functions with learnable sum of univariates functions like B-splines or Radial Basis functions (Z. Li 2024). Some recent papers have shown that using KAN layers in traditional architectures can improve performance and accuracy (C. Li et al. 2024).

### 3.3.1 KAN Layer as an network convolutionnal output

Our first idea is to simply replace the output convolutionnal layer of U-Net by a KAN convolutionnal layer, creating a new architecture referred to as "K-Net". The goal is to evaluate whether the KAN layer improves training when the network has more layers.

Since KAN layers are more computationally expensive, training takes longer. However, replacing only one layer keeps the training time per epoch manageable, allowing us to analyze the complexity map (Figure 17).

The results show that K-Net's accuracy is slightly lower than U-Net's, and the same trend persists: accuracy decreases as the number of layers increases. Thus, this small modification does not provide significant benefits over the standard U-Net architecture.

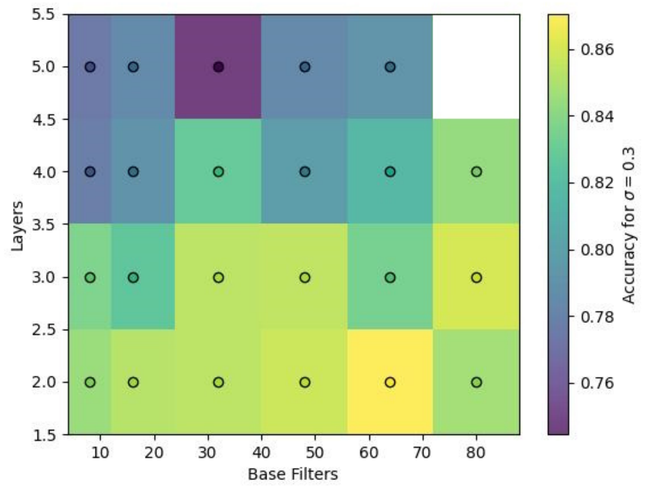


Figure 17: K-Net complexity map . Each point is a trained U-Net with KAN conv output with attention blocks and with X Base Filters and Y Layers. Yellow means higher accuracy and purple lower accuracy.

### 3.3.2 KAN Layer as a symbolic use

Another approach is to fit the function $N_{col}(n_H)$ using a KAN layer that operates on the flattened image matrix—that is, with one number input (the column density) and one number output (the number density). In parallel, a U-Net network can be used to incorporate spatial information as shown in Figure 18.

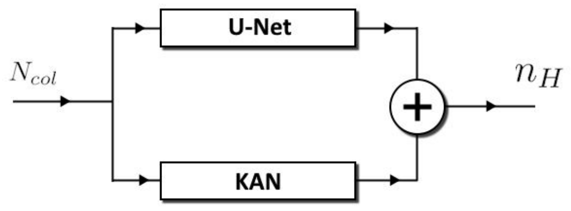


Figure 18: KAN as a symbolic use: Architecture

The comparison of the residuals between this modified architecture (with neuron layers $1 \rightarrow 10 \rightarrow 10 \rightarrow 10 \rightarrow 1$) and the U-Net, as shown in Figure 19, highlights a slight improvement in performance: the residuals are less scattered and there is reduced underestimation at high densities. This improvement may either be due to the increased complexity of the model, or it suggests that adding a parallel KAN layer to the standard U-Net does indeed provide a small performance boost.

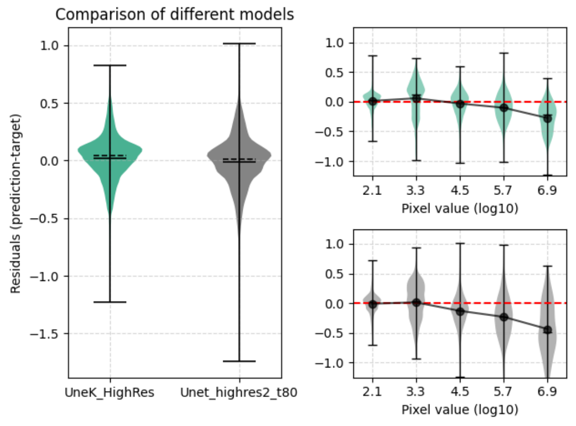


Figure 19: KAN as a symbolic use: Residuals. Green is the model using KAN, Gray is the model using just UNet.

### 3.3.3 KAN Layer as a replacement of convolutionnal layers

It is possible to construct convolutional layers using KANs: instead of relying on traditional ReLU activation functions, a KAN convolutional layer learns its activation function, for example using radial basis functions. KAN convolutional layers have been shown to perform better, although they come with a trade-off of slower prediction throughput and a higher number of model parameters. Therefore, we can replace some of the traditional convolutional layers with KAN convolutional layers. Following the approach used in C. Li et al. (2024), we will only replace the deeper layers with KAN. This architecture is named U-Kan.

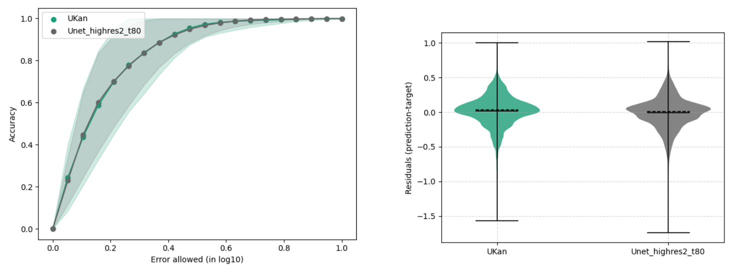


Figure 20: U-Kan Accuracy and Residuals, U-Kan(green and U-Net (black)

In our case, for an architecture with five layers, three of which use KAN convolutional layers, we do not observe any improvement in performance, as shown in Figure 20.

Finally, we can conclude that, in the case of estimating the mass-weighted number density using only the column density, KAN provides little to no improvement. However, its symbolic capabilities are interesting and worth considering for potential future use.

## 3.4 U-Net working scale

Real observations and derived column density maps span a large range of sizes, such as the Herschel column density map of Taurus B213, which will be seen in the next section and is a few parsecs long. However, our model is trained on a single large scale of approximately 6 pc (width of the $128 \times 128$ image). Therefore, to apply our model to observed maps, we train a U-Net model on a dataset generated from the same simulation but at a higher resolution: $1024^3$ instead of $512^3$. Thus, no physical parameters are changed—only the working resolution of the model is improved, from 0.05pc per pixel to 0.025pc per pixel. This new dataset contains 100 pairs of images instead of the previously 37. Therefore, the previously discussed trends regarding model complexity and performance remain unchanged. While a higher-resolution dataset provides more detailed information, the relationship between model capacity, accuracy, and computational cost follows the same principles.

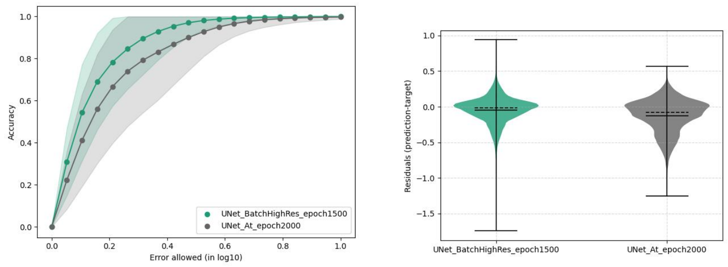


Figure 21: U-Net Accuracy and Residuals on a high resolution validation set for a model trained on this set (green) and the previous model(black)

As we can see in Figure 21, the model previously trained at a lower resolution performs worse than a U-Net model trained at this new resolution. Even though the residuals span a wider range of values, the majority of the residuals distribution remains centered around 0.0.

We can conclude that, to make an accurate prediction for an observation of a certain spatial size, the model must have been trained on that specific size. Otherwise, it is necessary to downsample the observation map before applying the prediction.

## 3.5 Size of the training set

In previous sections, we already discussed the effect of training set size on model performance, specifically, that a larger training set can lead to better results. This short section aims to validate that assumption. Note that we are still working within the same parameter range; this is not about generalization to unseen parameters.

Figure 22 clearly shows this performance improvement. However, there's a slight drop in performance for a training set size of around 60. This is explained by our training method: to limit memory usage, the dataset is split into N subsets that are processed sequentially within a single epoch.

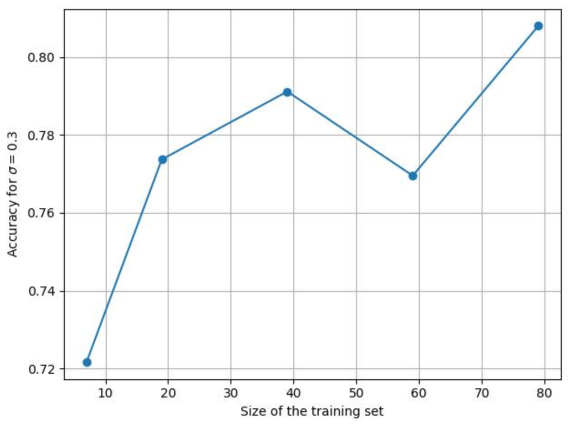


Figure 22: U-Net Effect of the training set size on the model accuracy . Each point is a model trained on respectively a set size of 7,19,39,59 and 79 images with a virtual batch size of 32.

To avoid noisy gradients, it's possible that the model does not see the entire dataset during a single epoch. For instance, if batches are divided into chunks of 32 images, and the total batch contains 60 images (as shown in the figure), 28 images will not be seen in that epoch. Since sampling is randomized at each epoch, they may be seen in the next one. If the model had been trained with 64 images, this drop in performance would likely not have occurred. Therefore, it's important to consider how the training set size interacts with the chosen batch subdivision. Later, when we introduce additional information, the dataset will become more memory-intensive, and we won't be able to load the entire set onto the GPU at once.

Figure 23 highlights the importance of ensuring that the training set size is a proper multiple of the batch size.

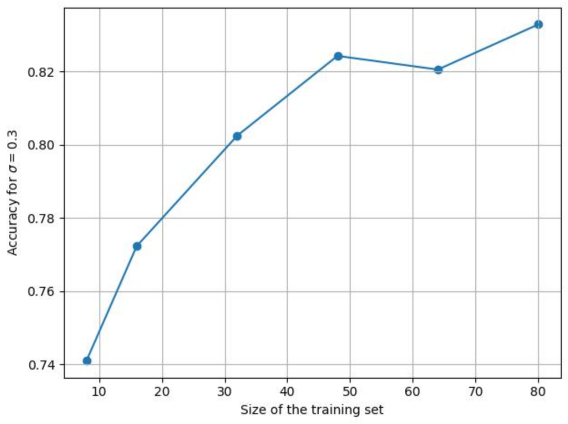


Figure 23: U-Net Effect of the training set size on the model accuracy . Each point is a model trained on respectively a set size of 8,16,32,48 and 64 images with a virtual batch size of 16.

## 4 Results

In this section, we apply our U-Net model fused with KAN as a symbolic role and using residuals fitting (see Appendix C) to Herschel observations (see Appendix D for method), allowing us to compare it with a few recent methods:

- A machine learning approach (Xu et al. 2023) that uses a Denoising Diffusion Probabilistic Model to predict molecular cloud number density.
- A method based on fitting the log-PDF of the column density combined with a probabilistic model (Gaches and M. Y. Grudić 2024).
- A novel inverse modeling approach (Orkisz and Kainulainen 2024).

### 4.1 Taurus L1495

The Herschel column density map of Taurus L1495, one of the closest star-forming filamentary structures, is obtained from Palmeirim et al. (2013) with a resolution of 18.2". The column density map is derived based on an optically thin graybody assumption. By adopting a power-law radial density profile to fit the column density, they found the central density of the filament to be $n_{H,c} = 7.5 \times 10^4 cm^{-3}$.

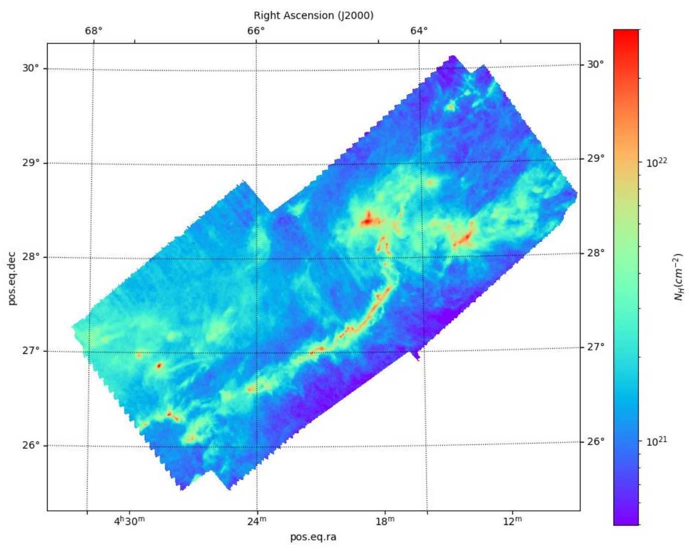


Figure 24: Taurus L1495 Column density

Xu et al. (2023) predicted a line of sight mass-weighted number density of $4.7 \times 10^4 cm^{-3}$. Gaches and M. Y. Grudić (2024) on the other hand, predicted a density of $2 \times 10^3 cm^{-3}$ for diffuse regions and $10^5 cm^{-3}$ for dense regions, such as the filament core. Not to mention D. Li and Goldsmith (2012), who, by using cyanoacetylene transitions, measured an average volume density of $1.8 \times 10^4 cm^{-3}$.

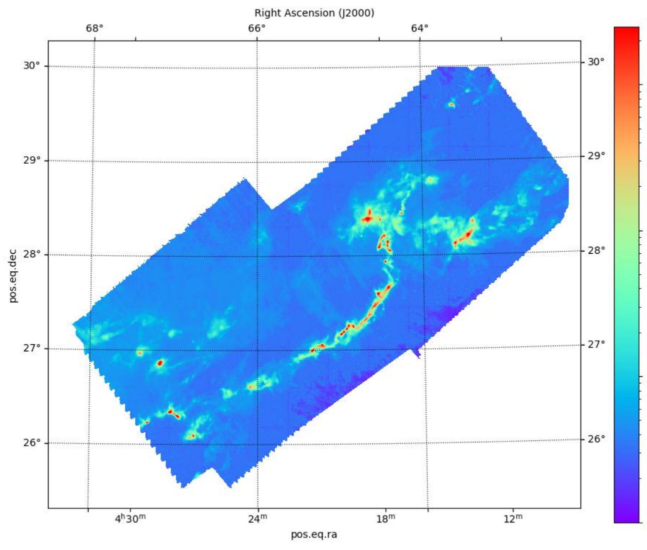


Figure 25: Taurus L1495 Predicted mass-weighted density

Our prediction is shown in Figure 25. We predict a maximum mass-weighted density of $10^5 cm^{-3}$ and density at the center of the filament of $2 \times 10^4 cm^{-3}$, with diffuse regions ranging between $0.5 \times 10^2 cm^{-3}$ and $1.2 \times 10^2 cm^{-3}$(corresponding to the 50% and 95% quantiles, respectively). The correlation between number density and column density is shown in Figure 26.

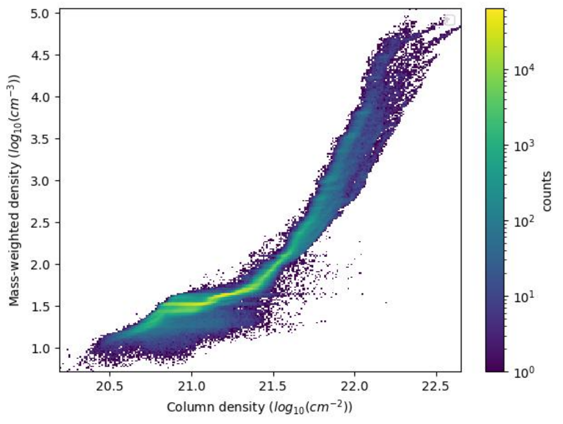


Figure 26: Taurus L1495 Correlation for the Predicted mass-weighted density

Our prediction is lower than previous estimates, likely even when accounting for the fact that we predict a mass-weighted average rather than the maximum along the line of sight. This difference may be due to an underestimation of high densities, as highlighted in Section 3.1, or in a bigger problem in how we apply the model on observations.

## 4.2 Polaris Flare

The Polaris Flare is a diffuse cloud that exhibits little to no signs of ongoing star formation, making it a quiescent star-forming region. However, Ward-Thompson et al. (2010) identified five possibly bound prestellar cores within the "saxophone" region with a mean density of $n_{H,c} = 5 \times 10^4 cm^{-3}$ and a mean radius near $r_c = 0.03$pc. The Polaris Flare is characterized by diffuse gaseous structures and striations, with a strong magnetic field threading the cloud (Panopoulou, Psaradaki, and Tassis 2016).

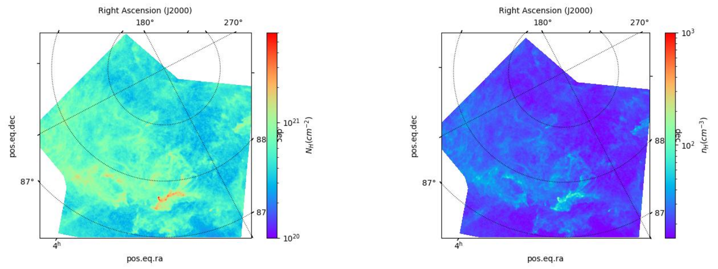


Figure 27: Polaris Flare: Left: column density; Right: predicted mass-weighted density

Using their probabilistic method, Gaches and M. Y. Grudić (2024) found a diffuse density of $n_{H,\text{diffuse}} \sim 10^2 cm^{-3}$ and a dense density $n_{H,\text{dense}}$ reaching up to a few $10^3 cm^{-3}$. Our model prediction, shown in Figure 27, gives a mean diffuse density of $n_H = 3 \times 10^1 cm^{-3}$ and a maximum mass-weighted density of $n_H = 3 \times 10^2 cm^{-3}$ in the "saxophone" region. This suggests our prediction is an order of magnitude lower for the diffuse region and likely several orders lower for the saxophone region, where our basic model don't show any massive cores.

We can also perform a quick calculation to verify this. Suppose there are two components along the line of sight: a diffuse component with a length $L_D$ and density $n_D$, and a massive component (the core) with a length $L_c = 2r_c$ and density $n_c$. Then, the predicted number density $n_M$ can be written as:

$$< n >_M = \frac{\sum n_H^2}{\sum n_H} = \frac{L_D n_D^2 + L_c n_c^2}{L_D n_D + L_c n_c}$$

We can then estimate the order of magnitude of the cloud length along the line of sight $L = L_D + L_C$:

$$L = L_c \frac{n_c^2 - n_D^2 - < n >_M (n_c - n_D)}{n_D(< n >_M - n_D)} \tag{7}$$

Using the bound prestellar cores found in the Saxophone region, we estimate a cloud length $L$ of $10^5$pc, which is much larger than the distance to the Polaris Flare (150 pc, Heithausen et al. (1998)). This suggests that our predicted density is indeed incorrect.

This bad performance for this cloud could be due to a lack of diversity in the training dataset, i.e., the absence of training data featuring a strong magnetic field and/or physical properties similar to those of the Polaris Flare cloud like for diffuse clouds.

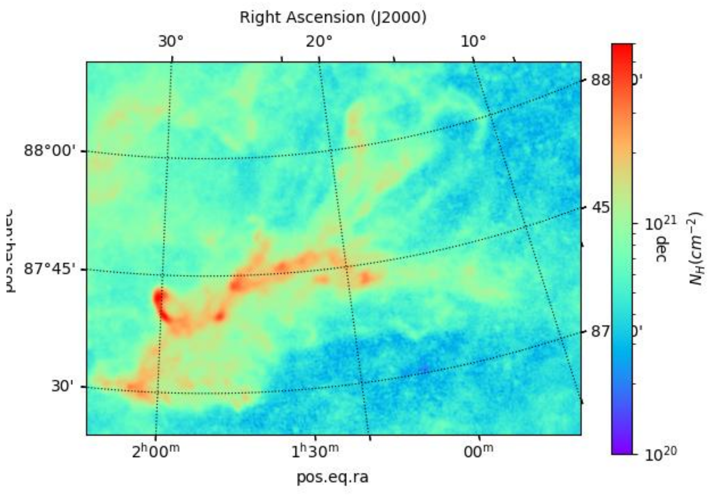


Figure 28: Polaris Flare Saxophone Region: Column density

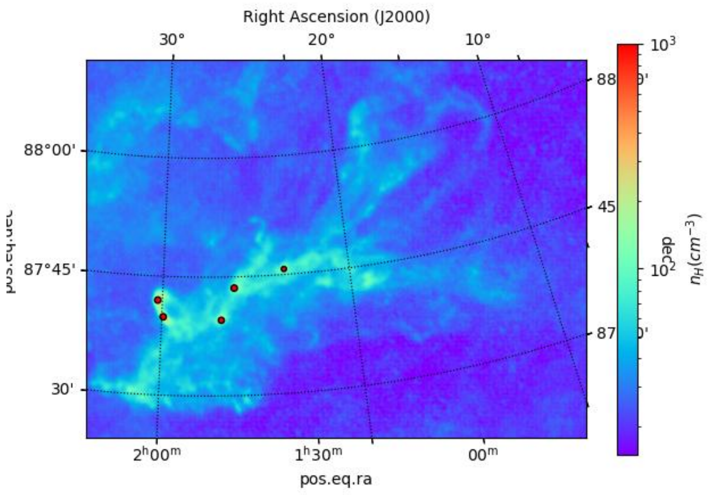


Figure 29: Polaris Flare Saxophone Region Predicted mass-weighted density; The dots are the dense cores.

## 4.3 Orion

Orion is one of the most prominent nearby giant molecular clouds (GMCs) actively forming stars, located at a distance of approximately 414 pc (Menten et al. 2007).

### Orion A

Orion A is a part of the Orion group. It hosts a range of environments from quiescent regions to dense, active star-forming sites such as the Integral Shaped Filament (ISF). Using Herschel observations. The Orion A cloud is notable for its complex interplay of gravity, turbulence, and magnetic fields, driving its ongoing star formation and evolution (Bally 2008). The column density is obtained from Roy et al. (2013).

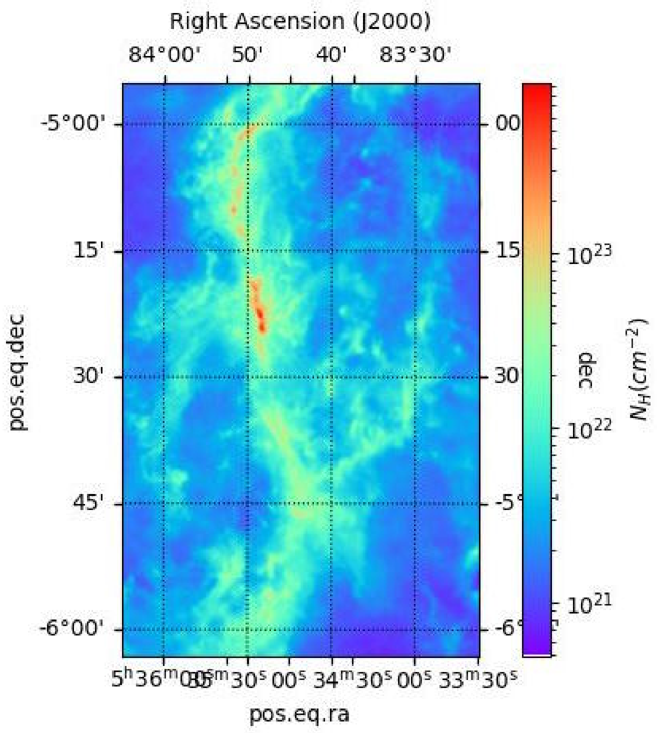


Figure 30: Orion A Column density

Using our model, shown in Figure 31, we found a mass-weighted number density of $n_{H,\text{diffuse}} = 50 cm^{-3}$ for diffuse regions and a maximum of $n_{H,c} = 3 \times 10^5 cm^{-3}$ for dense areas, such as cores. These results are consistent with the cloud properties of Orion A.

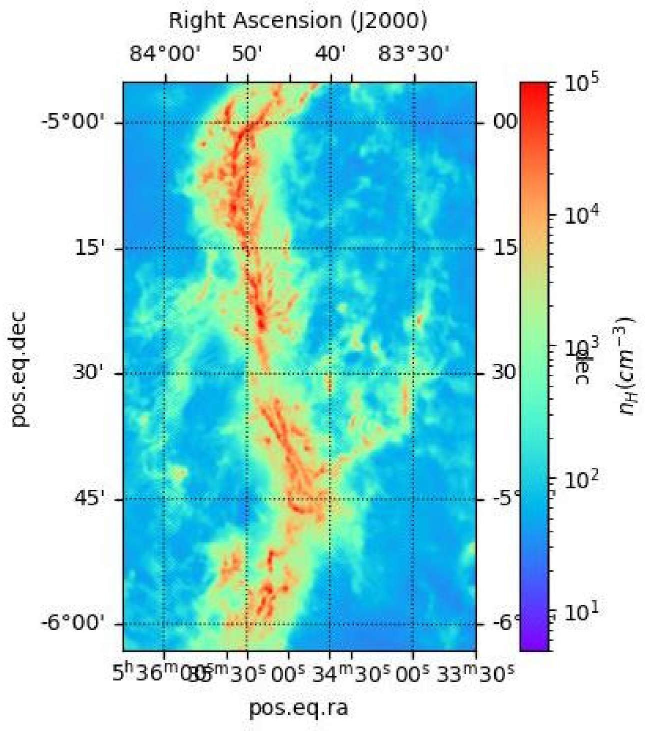


Figure 31: Orion A Filament Predicted mass-weighted density

**Orion B**

Orion B is another nearby giant molecular cloud located situated northeast of Orion A. It presents a mix of quiescent and active star-forming regions. Herschel observations (Könyves et al. 2020) reveal a fragmented filamentary structure, with star formation primarily concentrated in dense clumps.

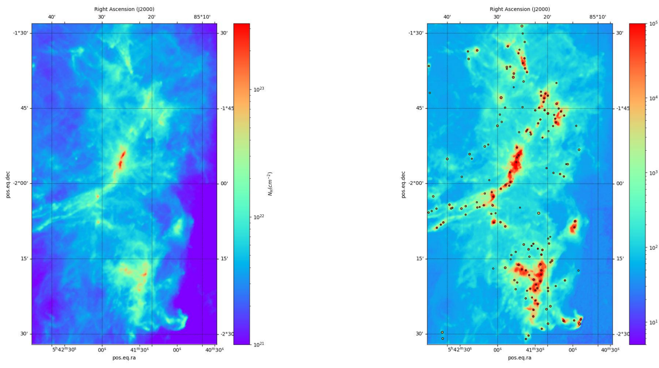


Figure 32: Orion B: NGC2023 & NGC2024 Column density (left) and Predicted mass-weighted density (Right). The dots are the dense cores.

A mass-weighted number density of $n_{H,\text{diffuse}} = 50cm^{-3}$ is predicted in the more diffuse areas, while the densest regions reach values up to $2.5 \times 10^6 cm^{-3}$."As shown in Figures 32 & 33, the cores identified byKönyves et al. (2020) seem to exhibit densities of the same order of magnitude.

Thus, our prediction for the mass-weighted density is consistent with both the overall cloud properties and the characteristics of the derived dense cores.

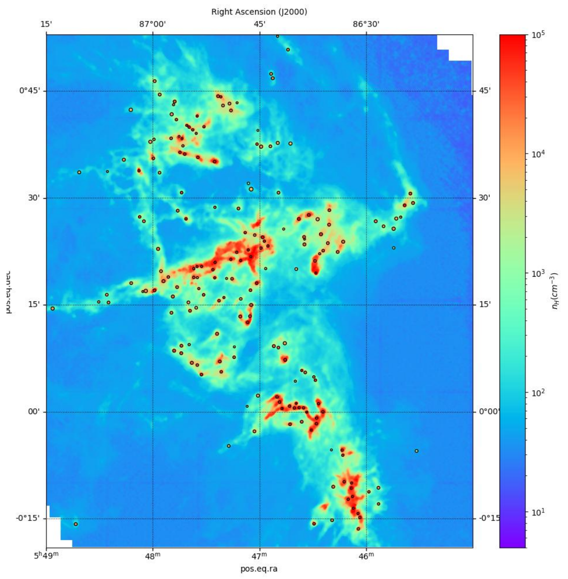


Figure 33: Orion B: NGC2071 & NGC2068 Predicted mass-weighted density. The dots are the dense cores.

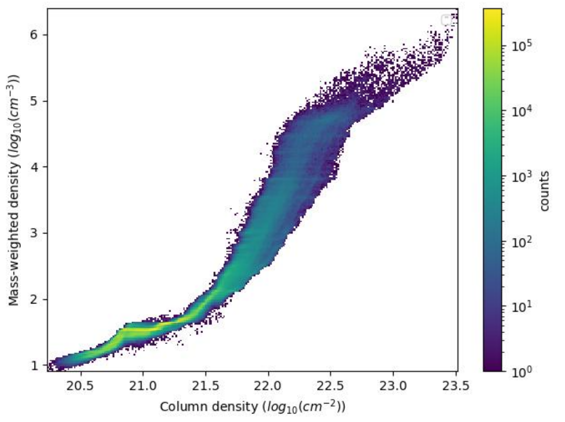


Figure 34: Orion B Correlation for the Predicted mass-weighted density

## 5 Discussion

In this first part, we tried to reproduce the results of Xu et al. (2023)(comparison on Taurus), while also exploring new possibilities for the U-Net architecture, including the use of the recently proposed KAN networks (Liu et al. 2024). Using only the ORION simulation (Ntormousi and Patrick Hennebelle 2019), the model performs significantly better in predicting dense clouds, consistent with the simulation's properties. As a result, we obtained a good prediction of the mass-weighted average density along the line of sight for dense regions like Orion filament. However, accurately predicting dense regions within the more diffuse areas (i.e high density for low column density) remains very challenging as shown for the Polaris Flare or in the diffuse regions of the previous studied clouds.

Figure 35 clearly highlights a significant underestimation of the average density in dense cores located in regions of low column density. However, beyond a certain column density threshold, the model performs well.

We currently have several hypotheses to explain this behavior, as well as potential solutions that could be explored in more detail during a future PhD or in subsequent work.

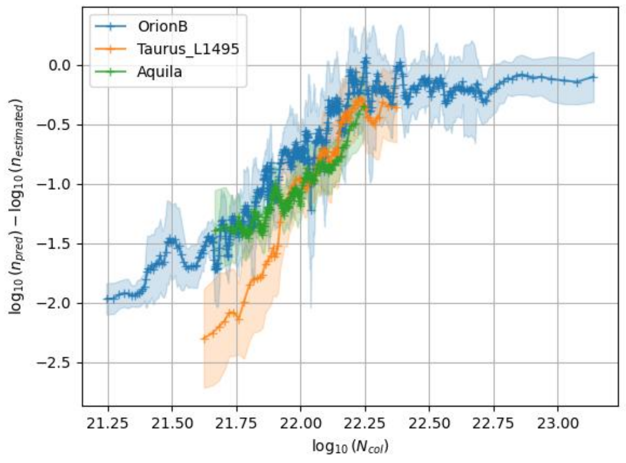


Figure 35: Baseline of model error on observations: difference between density $n_{\mathrm{estimated}}$ estimated using the dense cores by previous papers and the density $n_{\mathrm{pred}}$ found by our model. The baseline is found by using a moving average of $N = 10$. The transparent area is the standard deviation from the $N$ points used in the moving average.

**Possibility 1: A lack of relevant examples in the training simulation**

The simulation used for training may simply not contain regions that exhibit high volume densities combined with low column densities. In that case, the model is unable to predict configurations it has never encountered during training , a limitation inherent to data-driven methods.

That said, the model does learn spatial relationships, and should in principle be able to infer that regions near filaments or dense structures tend to have elevated densities. This may explain why the model performs well above a certain column density threshold, where it is clearly within the filamentary structures, but struggles just outside those regions.

To quantify the degree of similarity between the simulation and real observed clouds, we could define simple statistical indicators, such as the mean and standard deviation of the column density distribution, and compare these between the training data and target regions. (See Appendix D for an overview of the observational regions being predicted.)

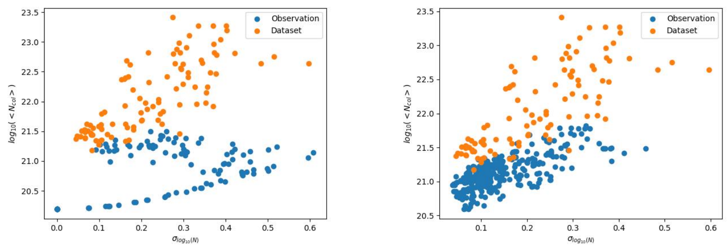


Figure 36: Mean and Std of the column density in predicted and training regions. Left is for Taurus L1495; Right for Orion B. Orange is for dataset, Blue for observations.

Figure 36 suggests that the observations and the training set share few similar regions, at least in terms of key statistical properties of the column density distribution. However, this comparison is limited to global quantities, and it remains possible that subregions within the observational regions may still be well represented in the simulation. Nonetheless, the mismatch is concerning.

To address this, one potential solution would be to make new simulations specifically designed to reproduce the physical conditions and statistical properties of the observed cloud, assuming the issue does not lie in the training set construction process itself.

**Possibility 2: Lack of context in low column density regions**

As shown in Appendix D, the observational data is processed by dividing it into smaller regions or patches. When predicting such a small sub-region, the model no longer has access to the broader spatial context, for example, it doesn't "know" that the region lies close to a filament.

Take for instance the area shown in Figure 37: it is clear that the model has no way of inferring from the patch alone that it is located near a filamentary structure. It therefore lacks crucial contextual information.

Initially, our hypothesis was that the model could learn to recognize local structures, and associate them with high-density environments. However, this does not appear to be sufficient in practice.

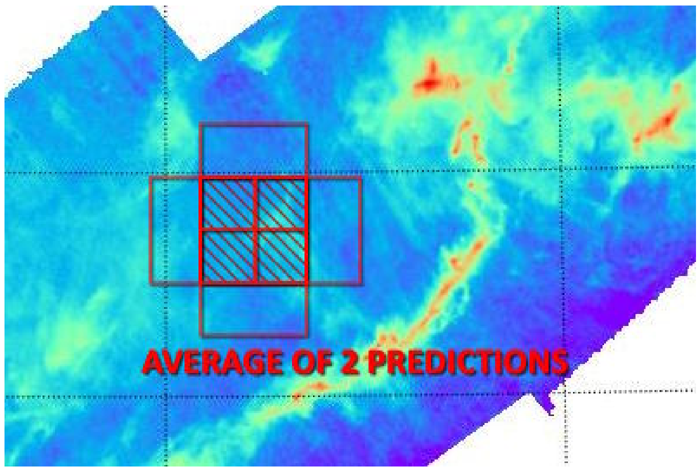


Figure 37: Column density map of Taurus L1495 with an example of the input area passed to the model. Here with 2 predictions per pixel.

A possible solution to this limitation would be to design a new architecture that includes an additional input: for example, a low-resolution version of the full cloud, along with the position of the current high-resolution patch being predicted. This would provide the model with global contextual information while preserving the local detail needed for accurate prediction.

This idea is conceptually related to graph-based learning, where nodes might represent different structures or regions of the cloud. In this context, Graph Neural Networks (GNNs) could be an interesting avenue to explore, as they are designed to capture both local interactions and global structure.

**Possibility 3: Architecture problem & Diffusion**
In their paper, Xu et al. applied a diffusion model that significantly outperforms ours, especially in diffuse regions. Interestingly, their simulations seem to span a similar parameter space as ours, although they benefit from a larger dataset (~7000 images). Yet, even their base U-Net architecture fails to achieve comparable performance to the diffusion-based model.

They attribute the superior results of the diffusion model to its Markovian nature, which allows the model to reconstruct the signal iteratively, layer by layer. However, this explanation remains somewhat vague, and the true underlying mechanism behind this improvement deserves deeper investigation.

We chose not to implement a diffusion model in this work for several reasons. Primarily, it would have required significantly more development time, and we initially believed the performance gain, while real, would not be dramatic enough to justify the cost, especially given the relatively simple objective at this stage. However, the errors exhibited by our model have revealed something more fundamental, which is important to understand.

Therefore, it is crucial to take the time to explore why diffusion-based models succeed in overcoming these limitations. This deeper understanding will likely be developed in the context of a PhD project or in subsequent work.

Part III

# Volume density from diverse observational Inputs

## 6 Introduction

Up to this point in the document, the prediction of average density has relied solely on column density. However, it may be valuable to incorporate additional information, such as molecular emission data (e.g., $^{13}$CO cubes), which can enrich the input and provide new insights to the model. Beyond simply enhancing the input features, this may also be necessary for more advanced tasks, such as reconstructing the 3D structure of the cloud or predicting the peak density along the line of sight.

Current research on 3D reconstruction from 2D data has focused primarily on dense cores, which are often assumed to be easier to model due to their quasi-spherical shapes. On the side of inverse methods, the AVIATOR algorithm, from Hasenberger and Alves (2020), enables 3D morphological reconstruction from 2D maps. In the field of deep learning, Ksoll et al. (2024) propose a model based on conditional invertible neural networks (cINNs), trained on multi-wavelength dust emission maps.

Nevertheless, such approaches remain largely limited to dense cores. More recently, the availability of precise stellar distances from Gaia has enabled low-resolution 3D reconstructions of entire molecular cloud complexes (Dharmawardena et al. 2022).

While our immediate objective is not to perform full 3D reconstruction, which remains the most ambitious and technically demanding goal, we focus here on adapting the architecture to handle multiple inputs. This will serve as a foundation for future efforts toward full 3D reconstruction, or alternatively, the still challenging task of disentangling distinct components along the line of sight.

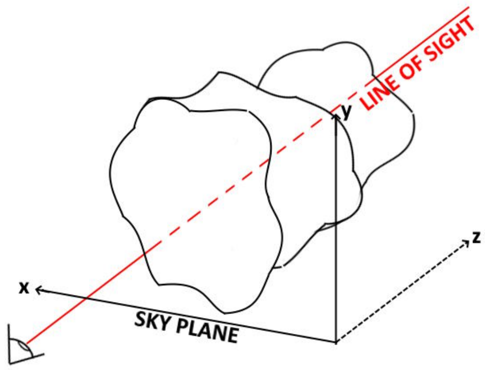


Figure 38: Line-of-sight (Ce graphe est moche...)

Moreover, $^{13}$CO and CO isotopologues in general are commonly observed in molecular clouds, meaning that existing observational data, such as that available for the Orion B region (Roueff et al. 2021), can be used for direct applications.

Additionally, one could consider incorporating emissions from other molecular tracers sensitive to specific density regimes. However, obtaining such emissions from simulations requires advanced radiative transfer calculations, which can be computationally intensive and complex, particularly compared to the relatively easier case of CO isotopologues.

Indeed, simulating molecular emission maps from numerical simulations involves radiative transfer modeling, which must be carried out using dedicated radiative transfer codes or, in some cases, a simple custom ray-tracing approach, particularly for CO molecules.

A major limitation with CO, however, is that its abundance in gas-phase decreases in high-density regions due to freeze-out onto dust grains (Panessa et al. 2023).

That said, since the simulation we are using (ORION, see subsection 2.1) does not include chemistry and thus does not track molecular abundances, we can assume a uniform CO abundance throughout the cloud. This simplification should be sufficient to evaluate whether the model is capable of processing and learning from this idealized scenario. In future work, more sophisticated simulations that include chemical networks could be used, alongside a neural network trained on emission maps from various molecular lines.

## 7 Methods

### 7.1 Emission maps

First, if we aim to incorporate additional inputs, we must begin by identifying which ones to use and how to extract them from simulations in order to construct the training dataset.

Since our goal is to gain insight into the structure along the line of sight, emission maps of various molecular species appear to be a promising addition. For instance, using the emission from a relatively scarce molecule such as $^{13}$CO, which tends to be optically thin, meaning the observed emission is not limited to outer cloud layers, can provide valuable information. Due to the Doppler effect, the spectral lines shift depending on the velocity of the gas at a given (x, y, z) position. As a result, the observed intensity is spread across a spectral profile, revealing distinct components at different velocities. This leads to the construction of what is known as a position-position-velocity (PPV) cube, offering a crucial piece of information along the third spatial dimension, namely, the line of sight.

#### 7.1.1 Computing the maps

To illustrate how an emission map can be computed, we consider the example of $^{13}$CO, though the same approach applies to other CO isotopologues.

The abundance of CO in molecular clouds is typically assumed to be on the order of $\chi_{\mathrm{CO}} = \frac{n_{CO}}{n_H} = 10^{-4}$(Fuente et al. 2019), with abundances expressed with respect to the total hydrogen density. The isotope $^{13}$CO is approximately 70 times less abundant than $^{13}$CO so $\chi_{13CO} = 1.4 \times 10^{-6}$.

The computation relies on a ray-tracing algorithm. Since the physical quantities involved (e.g., temperature, density) are not constant along the line of sight, the radiative transfer equation must be integrated step-by-step through each cell intersected by the ray, rather than over the entire line of sight at once. The ray thus propagates through the cloud toward the observer, accumulating the emerging intensity incrementally.

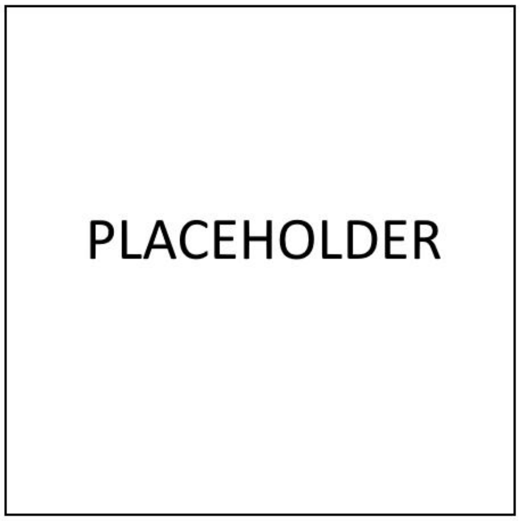


Figure 39: Ray-tracing

The governing equation is the standard radiative transfer equation:

$$\frac{dI_\nu}{dz} = j_\nu - \alpha_\nu I_\nu \tag{8}$$

where $I_\nu$ is the specific intensity at frequency $\nu$ $j_\nu$ is the emission coefficient, and $\alpha_\nu$ is the absorption coefficient.

Because the simulation is discrete, i.e. the line of sight is composed of N cells, and physical parameters are assumed constant within each cell, the change in intensity as the ray passes through a single cell is then given by:

$$I_{\nu,i+1} = I_{\nu,i} e^{-\tau_i} + B_{\nu,i}(1 - e^{-\tau_i}) \tag{9}$$

where $I_{\nu,i}$ is the incoming intensity, i.e the intensity computed on the previous cell, $B_{\nu,i} = \frac{j_\nu}{\alpha_\nu}$ is the source function and we'll take the black-body emission, and $\tau = \alpha dz$ is the optical depth across the cell of width $\Delta z$.

The equation 9 directly comes from the integration of equation 8 by making all the coefficients constants inside the cell.

La diffuclté est donc de calculer tau_i.

...bla bla démo

CMB

maping nu -> V

Convertion température

Resultats: Integré, Spectres & Channel maps

#### 7.1.2 Preprocessing

### 7.2 Architectures

## 8 Performance

### 8.1 For mass-weighted average density

### 8.2 For max density

### 8.3 For structure reconstruction

Part IV

# Polaris : In-development tool for creating neural networks for science purposes

## 9 Introduction

This internship and paper first led to the development of a Python module named Polaris-Core, after the cloud where predictions are the most challenging. This module allows, in just a few lines:

- The creation of training datasets with the desired data, including the option to generate emission maps if needed.
- The training of CNN models.
- The visualization of various error metrics.
- The application of trained models to observational data.

To further simplify and accelerate the process of model creation, testing, and application, I initiated the development of a user interface application. This way, users can choose to work directly with the Python module, through the interface, or by combining both approaches.

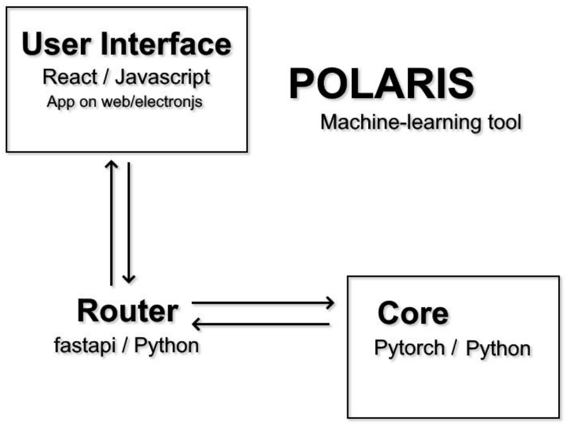


Figure 40: POLARIS Architecture

As shown in Figure 40,this tool is based on an architecture composed of three parts:

- The core module (Polaris-Core): responsible for training and calculations, built using PyTorch for machine learning.
- The communication code (Polaris-Routers): responsible for communication between the core and the interface, using FastAPI to create the communication routes.
- The interface (Polaris-Frontend): developed in JavaScript using React, with Three.js for 3D rendering.

*This tool is still under active and heavy development and has been created entirely on my own free time.*

## 10 Showcase

### 10.1 Models creation

First and foremost, the main purpose of the software is to assist in the design and diagnosis of neural network architectures. To achieve this, I opted for a blueprint-style interface model. In this setup, an architecture is built by connecting modular blocks, such as: Input → KAN → Output.

Each block represents a specific operation or layer, and the user can construct complex models by linking these elements together in a visual and intuitive way.

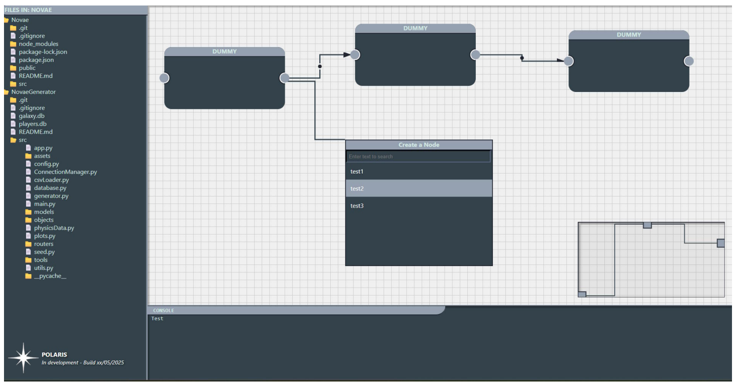


Figure 41: Dummy blueprint interface

The goal is also to enable quick diagnostics, for example by allowing one-click access to the learned weights of each block, or by visualizing intermediate feature maps directly within the interface.

### 10.2 3D Viewer

Another goal of the software is to make the analysis process more intuitive and user-friendly. For instance, it allows real-time 3D visualization of a typical molecular emission, using a simple radiative transfer calculation similar to the one used in section 7.1.1. Currently, any .npy or .fits file containing a 3D matrix can be loaded and opened with a single click.

The user can then navigate around or through the cloud or structure using the mouse. This 3D view helps to better understand the underlying morphology and is especially useful for quickly assessing the output of models attempting to reconstruct 3D structures, e.g., checking at a glance whether a predicted filament appears coherent or is unnaturally truncated.

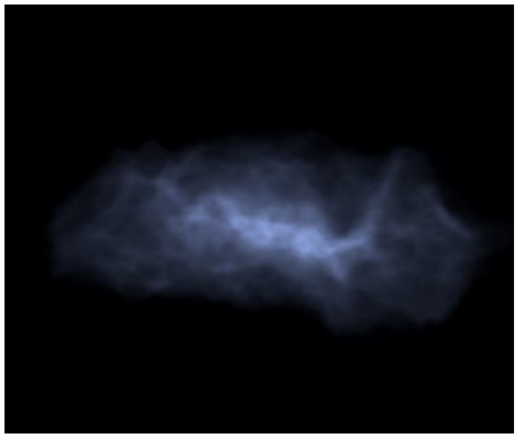


Figure 42: 3D Viewer - ORION Sim cloud, emission map mode

## 11 Conclusion

Ultimately, this software is designed to serve three main purposes:

- Assist in the design and analysis of neural network architectures, helping reduce the time spent on development and debugging, so that more time can be dedicated to the underlying physics during my future PhD (if it happens. . . ).

- Facilitate reproducibility of results. At present, reproducing results from papers using deep learning often requires not only access to the model architecture and code, but also all the training parameters (known as hyperparameters). Without these, performance can diverge significantly, even when using the same training set and method.

- Make deep learning more accessible to a broader community interested in studying the interstellar medium (ISM), and potentially beyond.

Achieving these goals still requires substantial development, which I hope to continue during the PhD.

## References

Bally, John (Nov. 2008). *Overview of the Orion Complex.* arXiv:0812.0046 [astro-ph]. DOI: 10.48550/arXiv.0812.0046. URL: http://arxiv.org/abs/0812.0046 (visited on 04/28/2025).

Dharmawardena, T. E. et al. (Dec. 2022). "The three-dimensional structure of Galactic molecular cloud complexes out to 2.5 kpc." In: *Monthly Notices of the Royal Astronomical Society* 519.1. arXiv:2210.03615 [astro-ph], pp. 228–247. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stac2790. URL: http://arxiv.org/abs/2210.03615 (visited on 05/15/2025).

Fuente, A. et al. (Apr. 2019). "Gas phase Elemental abundances in Molecular cloudS (GEMS): I. The prototypical dark cloud TMC 1." In: *Astronomy & Astrophysics* 624, A105. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201834654. URL: https://www.aanda.org/10.1051/0004-6361/201834654 (visited on 05/15/2025).

Gaches, Brandt A. L. and Michael Y. Grudić (2024). *A Probabilistic Model to Estimate Number Densities from Column Densities in Molecular Clouds.* Version Number: 1. DOI: 10.48550/ARXIV.2412.16290. URL: https://arxiv.org/abs/2412.16290 (visited on 02/24/2025).

Girichidis, Philipp et al. (2020). "Physical Processes in Star Formation." In: Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2005.06472. URL: https://arxiv.org/abs/2005.06472 (visited on 02/26/2025).

Guszejnov, Dávid and Philip F. Hopkins (June 2016). "Star formation in a turbulent framework: from giant molecular clouds to protostars." en. In: *Monthly Notices of the Royal Astronomical Society* 459.1, pp. 9–20. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stw619. URL: https://academic.oup.com/mnras/article-lookup/doi/10.1093/mnras/stw619 (visited on 02/26/2025).

Hasenberger, B. and J. Alves (Jan. 2020). "AVIATOR: Morphological object reconstruction in 3D. An application to dense cores." In: *Astronomy & Astrophysics* 633. arXiv:1912.01005 [astro-ph], A132. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201936095. URL: http://arxiv.org/abs/1912.01005 (visited on 05/15/2025).

Heithausen, A. et al. (Mar. 1998). "The IRAM key project: small-scale structure of pre-star forming regions. Combined mass spectra and scaling laws." In: *Astronomy and Astrophysics* 331. ADS Bibcode: 1998A&A...331L..65H, pp. L65–L68. ISSN: 0004-6361. URL: https://ui.adsabs.harvard.edu/abs/1998A&A...331L..65H (visited on 04/17/2025).

Hennebelle, P. and M.Y. Grudić (Sept. 2024). "The Physical Origin of the Stellar Initial Mass Function." en. In: *Annual Review of Astronomy and Astrophysics* 62.1, pp. 63–111. ISSN: 0066-4146, 1545-4282. DOI: 10.1146/annurev-astro-052622-031748. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-astro-052622-031748 (visited on 02/26/2025).

Hopkins, Philip F. (Apr. 2013). "A general theory of turbulent fragmentation." en. In: *Monthly Notices of the Royal Astronomical Society* 430.3, pp. 1653–1693. ISSN: 1365-2966, 0035-8711. DOI: 10.1093/mnras/sts704. URL: http://academic.oup.com/mnras/article/430/3/1653/977851/A-general-theory-of-turbulent-fragmentation (visited on 02/26/2025).

Kingma, Diederik P. and Jimmy Ba (Jan. 2017). *Adam: A Method for Stochastic Optimization.* arXiv:1412.6980 [cs]. DOI: 10.48550/arXiv.1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 03/19/2025).

Könyves, V. et al. (Mar. 2020). "Properties of the dense core population in Orion B as seen by the *Herschel* Gould Belt survey." In: *Astronomy & Astrophysics* 635, A34. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201834753. URL: https://www.aanda.org/10.1051/0004-6361/201834753 (visited on 04/28/2025).

Ksoll, Victor F. et al. (Mar. 2024). "A deep-learning approach to the 3D reconstruction of dust density and temperature in star-forming regions." In: *Astronomy & Astrophysics* 683. arXiv:2308.09657 [astro-ph], A246. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/202347758. URL: http://arxiv.org/abs/2308.09657 (visited on 04/22/2025).

Li, Chenxin et al. (Aug. 2024). *U-KAN Makes Strong Backbone for Medical Image Segmentation and Generation.* arXiv:2406.02918 [eess]. DOI: 10.48550/arXiv.2406.02918. URL: http://arxiv.org/abs/2406.02918 (visited on 03/27/2025).

Li, Di and Paul F. Goldsmith (Sept. 2012). "Is the Taurus B213 Region a True Filament?: Observations of Multiple Cyanoacetylene Transitions." In: *The Astrophysical Journal* 756.1. arXiv:1207.0044 [astro-ph], p. 12. ISSN: 0004-637X, 1538-4357. DOI: 10.1088/0004-637X/756/1/12. URL: http://arxiv.org/abs/1207.0044 (visited on 03/25/2025).

Li, Ziyao (May 2024). *Kolmogorov-Arnold Networks are Radial Basis Function Networks.* arXiv:2405.06721 [cs]. DOI: 10.48550/arXiv.2405.06721. URL: http://arxiv.org/abs/2405.06721 (visited on 03/20/2025).

Liu, Ziming et al. (2024). *KAN: Kolmogorov-Arnold Networks.* Version Number: 5. DOI: 10.48550/ARXIV.2404.19756. URL: https://arxiv.org/abs/2404.19756 (visited on 03/17/2025).

McCulloch, Warren S. and Walter Pitts (Dec. 1943). "A logical calculus of the ideas immanent in nervous activity." en. In: *The Bulletin of Mathematical Biophysics* 5.4, pp. 115–133. ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259. URL: http://link.springer.com/10.1007/BF02478259 (visited on 05/26/2025).

Menten, K. M. et al. (Nov. 2007). "The distance to the Orion Nebula." en. In: *Astronomy & Astrophysics* 474.2, pp. 515–520. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361:20078247. URL: https://www.aanda.org/articles/aa/abs/2007/41/aa8247-07/aa8247-07.html (visited on 04/28/2025).

Ntormousi, Evangelia and Patrick Hennebelle (May 2019). "Core and stellar mass functions in massive collapsing filaments." In: *Astronomy & Astrophysics* 625, A82. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201834094. URL: https://www.aanda.org/10.1051/0004-6361/201834094 (visited on 03/17/2025).

Oktay, Ozan et al. (May 2018). *Attention U-Net: Learning Where to Look for the Pancreas.* arXiv:1804.03999 [cs]. DOI:

10.48550/arXiv.1804.03999. URL: http://arxiv.org/abs/1804.03999 (visited on 03/17/2025).

Orkisz, Jan H. and Jouni Kainulainen (2024). *On volume density and star formation in nearby molecular clouds.* Version Number: 1. DOI: 10.48550/ARXIV.2412.07595. URL: https://arxiv.org/abs/2412.07595 (visited on 02/24/2025).

Palmeirim, P. et al. (Feb. 2013). "*Herschel* view of the Taurus B211/3 filament and striations: evidence of filamentary growth?" In: *Astronomy & Astrophysics* 550, A38. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201220500. URL: http://www.aanda.org/10.1051/0004-6361/201220500 (visited on 03/25/2025).

Panessa, M. et al. (June 2023). "The evolution of HCO\$^{+}\$ in molecular clouds using a novel chemical post-processing algorithm." In: *Monthly Notices of the Royal Astronomical Society* 523.4. arXiv:2210.06251 [astro-ph], pp. 6138–6161. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stad1741. URL: http://arxiv.org/abs/2210.06251 (visited on 04/17/2025).

Panopoulou, G. V., I. Psaradaki, and K. Tassis (2016). "The magnetic field and dust filaments in the Polaris Flare." In: Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.1607.00005. URL: https://arxiv.org/abs/1607.00005 (visited on 03/27/2025).

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (May 2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation.* arXiv:1505.04597 [cs]. DOI: 10.48550/arXiv.1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 03/17/2025).

Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." en. In: *Psychological Review* 65.6, pp. 386–408. ISSN: 1939-1471, 0033-295X. DOI: 10.1037/h0042519. URL: https://doi.apa.org/doi/10.1037/h0042519 (visited on 05/22/2025).

Roueff, Antoine et al. (Jan. 2021). "$C^{18}$ $O,^{13}$ CO, and$^{12}$ CO abundances and excitation temperatures in the Orion B molecular cloud: Analysis of the achievable precision in modeling spectral lines within the approximation of the local thermodynamic equilibrium." In: *Astronomy & Astrophysics* 645, A26. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/202037776. URL: https://www.aanda.org/10.1051/0004-6361/202037776 (visited on 05/15/2025).

Roy, Arabindo et al. (Jan. 2013). "CHANGES OF DUST OPACITY WITH DENSITY IN THE ORION A MOLECULAR CLOUD." In: *The Astrophysical Journal* 763.1, p. 55. ISSN: 0004-637X, 1538-4357. DOI: 10.1088/0004-637X/763/1/55. URL: https://iopscience.iop.org/article/10.1088/0004-637X/763/1/55 (visited on 04/28/2025).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (Oct. 1986). "Learning representations by back-propagating errors." en. In: *Nature* 323.6088, pp. 533–536. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0 (visited on 05/22/2025).

Teyssier, R. (Apr. 2002). "Cosmological hydrodynamics with adaptive mesh refinement: A new high resolution code called RAMSES." In: *Astronomy & Astrophysics* 385.1, pp. 337–364. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361:20011817. URL: http://www.aanda.org/10.1051/0004-6361:20011817 (visited on 04/23/2025).

Ward-Thompson, D. et al. (July 2010). "A *Herschel* study of the properties of starless cores in the Polaris Flare dark cloud region using PACS and SPIRE." In: *Astronomy and Astrophysics* 518, p. L92. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201014618. URL: http://www.aanda.org/10.1051/0004-6361/201014618 (visited on 03/27/2025).

Xu, Duo et al. (June 2023). "Denoising Diffusion Probabilistic Models to Predict the Density of Molecular Clouds." In: *The Astrophysical Journal* 950.2. arXiv:2304.01670 [astro-ph], p. 146. ISSN: 0004-637X, 1538-4357. DOI: 10.3847/1538-4357/accae5. URL: http://arxiv.org/abs/2304.01670 (visited on 03/25/2025).

Yang, Suorong et al. (Nov. 2023). *Image Data Augmentation for Deep Learning: A Survey.* arXiv:2204.08610 [cs]. DOI: 10.48550/arXiv.2204.08610. URL: http://arxiv.org/abs/2204.08610 (visited on 03/19/2025).

# A Operations used in U-Net

The U-Net Ronneberger, Fischer, and Brox 2015 architecture (see figure 3) is built upon a combination of a few basic operations. To fully grasp how the network functions and later adapt it to our specific problem, it is crucial to understand these operations.

Since these operations are already implemented and optimized in popular libraries like PyTorch, it is easy to fall into the trap of constructing an architecture without understanding its underlying mechanics.

## A.1 Convolution

Convolution is the fundamental operation in deep learning architectures like U-Net. It applies a set of $N$ learnable filters (learnable kernels) to an input image, extracting features such as edges and patterns. Each filter slides across the image, computing weighted sums to create $N$ feature maps, which are then passed to the next layers. (see note for image author & details [5])

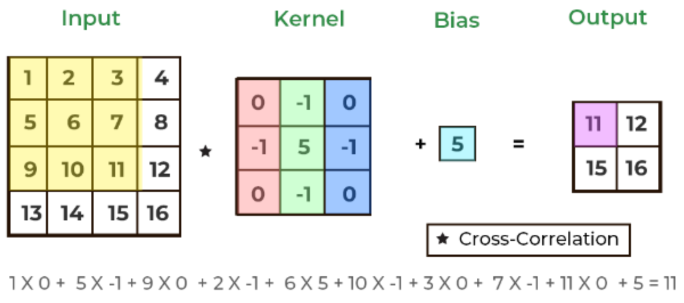

Input
Kernel
Bias
Output
★ Cross-Correlation
1 X 0 + 5 X -1 + 9 X 0 + 2 X -1 + 6 X 5 + 10 X -1 + 3 X 0 + 7 X -1 + 11 X 0 + 5 = 11

Figure 43: Network Convolution

## A.2 Max pooling

Max pooling is a downsampling operation used to reduce the spatial dimensions of feature maps while preserving the most important information. It works by selecting the maximum value from a small region (e.g., 2×2) of the feature map, improving computational efficiency and making the network more robust to small spatial variations.

It is also possible to use other pooling operations, such as Average Pooling, where instead of selecting the maximum value in a region, the average of all values is taken. However, Average Pooling is generally less efficient in capturing important features because it smooths out details, whereas Max Pooling preserves the most prominent features by selecting the highest value. (see note for image author & details[6])

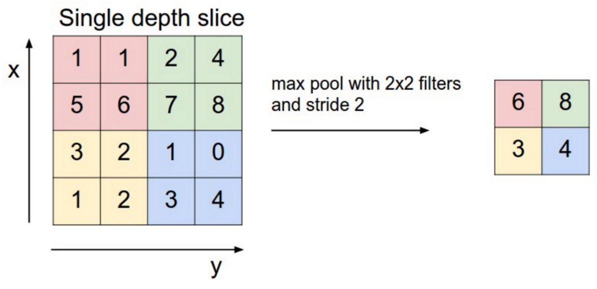


Figure 44: Network Max Pooling

## A.3 Conv Transpose

Also known as upsampling convolution, this operation increases the spatial dimensions of feature maps. It is used in the U-Net decoder to reconstruct high-resolution outputs from lower-resolution feature maps, helping to recover lost spatial details. (see note for image author & details[7])

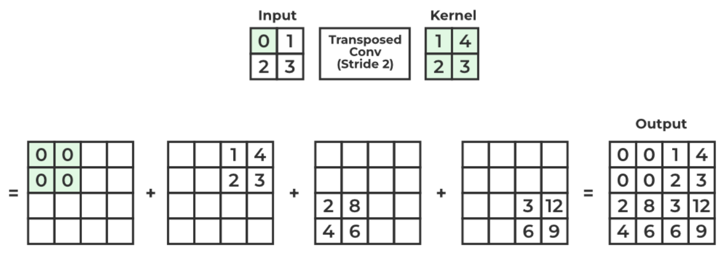


Figure 45: Network Convolutionnal Transpose

## A.4 Concatenate

Concatenation is a key operation in U-Net, allowing feature maps from different layers to be combined. In the skip connections, feature maps from the encoder are concatenated with corresponding decoder layers, ensuring that fine spatial information is retained during reconstruction.

## A.5 ReLU

ReLU is an activation function applied after convolution to introduce non-linearity. It sets negative values to zero while keeping positive values unchanged:

$$\text{ReLU}(x) = \max(0, x) \tag{10}$$

Which helps accelerate training and prevent the vanishing gradient problem.

[5]https://www.geeksforgeeks.org/apply-a-2d-convolution-operation-in-pytorch/
[6]https://www.geeksforgeeks.org/apply-a-2d-max-pooling-in-pytorch/
[7]https://www.geeksforgeeks.org/what-is-transposed-convolutional-layer/

## B How the score is computed

Creating a training dataset is a crucial step in building an effective model. In our case, we want to avoid overtraining on the borders of the simulation, as these regions contain less valuable information compared to the dense regions with filaments and cores.

To achieve this, we implement an additional filtering step:

1. We compute a "score" $s(\text{image})$ for each new candidate area before adding it to the dataset.

2. This score is calculated using a custom function that aligns with our training objectives.

3. We then apply a filter function $f(s)$. If a randomly generated number ($random$) in the range [0,1] satisfies $random < f(s)$, the region is accepted and added to the training set. I.e $f(s)$ is the probability to accept and add an area with a score of $s$ in the dataset.

This approach ensures that the dataset focuses more on important structures (filaments and cores) rather than sparse or less informative regions.

We can also use multiple scoring functions, each capturing different aspects of the dataset, and combine them using weighted ($w_i$) sums to account for various criteria. Mathematically, the final score $s$ can be expressed as:

$$s = \sum_i s_i w_i \tag{11}$$

This score can also be made for the column density image $s(N_C)$ and the volume density image $s(n_H)$.

### B.1 Smoothness score

The score function used when generating our training set needs to be based on the smoothness of the selected area. This is because if an area is entirely within a diffuse region, it will exhibit low spatial frequency variation (i.e., only large-scale structures) and lack filaments.

To ensure that each selected image contains both diffuse regions and some filaments, we use the variance of the smoothness as a criterion. The smoothness itself is defined as the Laplacian of the image, which highlights areas with sharp density changes (such as filaments).

$$s_\Delta(x) = \text{Var}[\Delta(\log(1+x) - \min[\log(1+x)])] \tag{12}$$

The variance in equation 12 is to get a mix of dense and high-density regions.

### B.2 Difference score

A basic criterion that can be chosen is the difference between the normalized column density and the normalized volume density. This helps identify regions where the local density distribution differs significantly between the two representations.

For example, when using mass-weighted density, this criterion will highlight dense regions, since areas with high volume density but relatively lower column density indicate localized dense structures along the l.o.s.

$$s_{\text{res}} = \text{Var}[\frac{N_C - \min(N_C)}{\max(N_C) - \min(N_C)} - \frac{n_H - \min(n_H)}{\max(n_H) - \min(n_H)}] \tag{13}$$

### B.3 Filter function

As a filter function $f(s)$, we use a sigmoid :

$$f(s) = \frac{1}{1 + \exp(-A(s - B))} \tag{14}$$

where $\begin{cases} A: & \text{step length} \\ B: & \text{offset} \end{cases}$ and are chosen empirically.

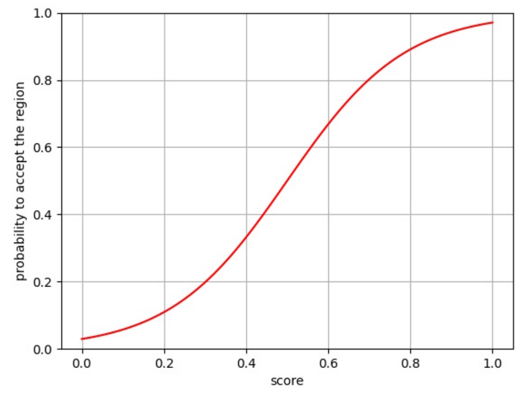


Figure 46: Sigmoid as a filter function

## C Residuals Correction

It is possible, and actually common in our case, that the model struggles to predict the mean values within certain density ranges. For example, the model tends to underestimate high densities (see Figure 47). This could indicate that the loss function is poorly formulated or that the architecture is not well-suited to the task.

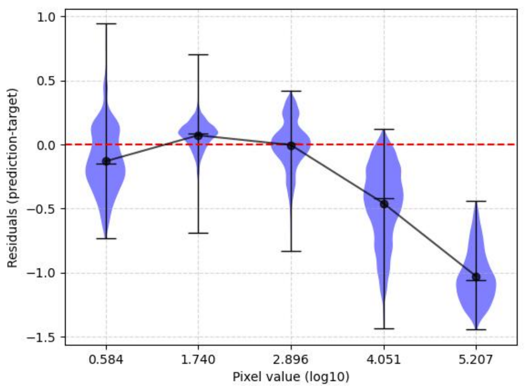


Figure 47: Residuals for a UNet model fused with KAN

For instance, while the training set is constructed using a score (see Appendix B) to avoid including too many diffuse regions, such regions still dominate overall, since they are inherently prevalent in a cloud. As a result, if we use the Mean Squared Error (MSE) as the loss function, accurately predicting diffuse regions becomes more important for the model than predicting high-density regions.

This is why one can introduce weighted loss functions, where the weight depends on the density. For example, the MSE loss (see Formula 4) can be modified as follows:

$$\text{MSELoss} = \mathbb{E}(w(n_{H,\text{target}}) \times (\text{prediction} - \text{target})^2) \qquad (15)$$

where $w(n_{H,\text{target}})$ is the weighting function, which in a simple case could follow a power law, such that $w \propto n^{\alpha}$, with $\alpha$ being the chosen exponent.

However, implementing this in practice is quite challenging, as the model may quickly develop undesirable behavior, for instance, predicting a dense background everywhere, since errors in low-density regions are given less importance.

Nevertheless, what matters more in model error is the distribution of residuals, not their average. If we know that the model systematically underestimates high densities, we can correct this bias uniformly across all observations using a baseline inferred from the validation set (not the observations themselves, otherwise, there would be no point in applying the model in the first place).

To determine this baseline, we apply a moving average: residuals are first binned (e.g., averaged every 50 values), and the resulting smoothed curve serves as the correction function. During inference, this baseline is then removed from each prediction via linear interpolation between the two nearest bin values. This allows the model's outputs to be corrected in a continuous and data-driven way, without altering the model architecture or retraining.

### C.1 Results

The result of this operation is shown in Figure 48. However, both the linear interpolation and the moving average approach present several major issues. For instance, they do not perfectly preserve the distribution of residuals. Moreover, the corrections can be abrupt from one bin to the next, which may introduce visible artifacts in the predictions. This can either make it harder to accurately identify a real structure or, conversely, highlight a feature that is in fact just an artifact caused by the residual correction.

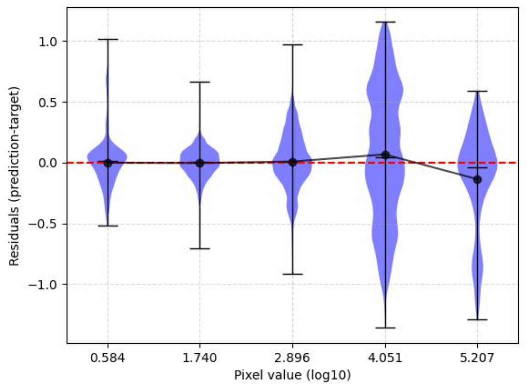


Figure 48: Residuals after the baseline is removed.

Moreover, it is clearly preferable that the model learns to accurately estimate all density ranges on its own. If it systematically underestimates high densities, for instance, this also indicates that it has failed to learn the underlying physics and conditions associated with these high-density regions. As a result, the residuals will remain widely distributed, even after applying the correction.

Ultimately, it is important to understand that while such a correction can compensate for part of the avoidable error, it does not fix the underlying model deficiencies. On the contrary, it can obscure them, potentially hiding systematic under- or over-estimations across certain density ranges and making diagnostic evaluation more difficult.

## D How the model is applied on observations

## E Archives of some tested architectures

Although these neural network architectures do not currently lead to significant improvements, and sometimes even degrade performance, they represent initial explorations of promising concepts.

### E.1 UNet with multi-outputs

### E.2 Using custom loss function

### E.3 Pos-Pos-Vel to Pos-Pos-Pos