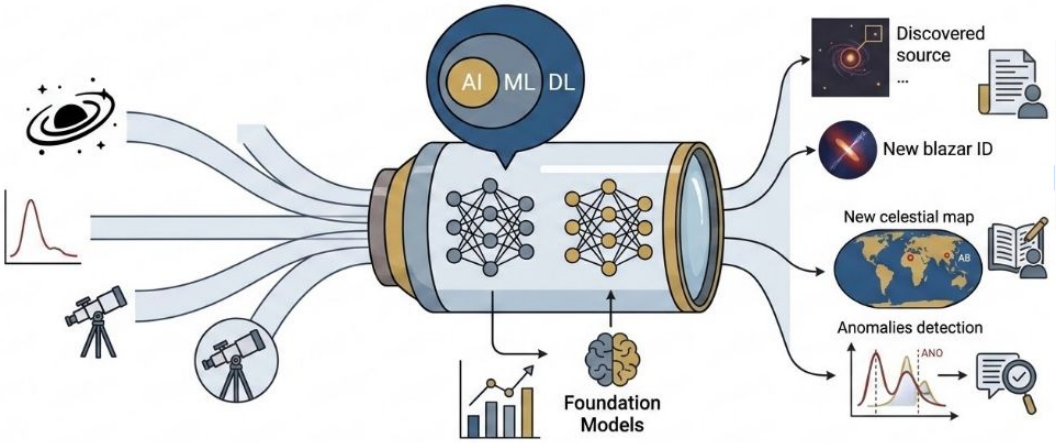
AI for Astrophysics

From Pattern Recognition to Scientific Discovery



Yvonne Becherini
Laboratoire Astroparticule et Cosmologie
Université Paris Cité

Applied Data Analytics (Usually in November)

Core data-science workflow: data preparation, supervised learning, regression, classification, imbalanced data, deep learning, CNNs, time series, unsupervised learning, clustering and latent-space exploration.

Advanced Applied Data Analytics (Usually in January)

Modern AI methods: generative models, autoencoders, transformers, foundation models, GNNs, probabilistic learning, Bayesian uncertainty, hyperparameter tuning, dimensionality reduction and knowledge-informed neural networks.

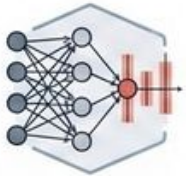
Focus of this Lecture

Foundation Models, Simulation-based inference https://arxiv.org/abs/2508.12939

Artificial Intelligence in Astrophysics

HIGH-DIMENSIONAL DATA
Large, Heterogeneous, Noisy, Incomplete Data

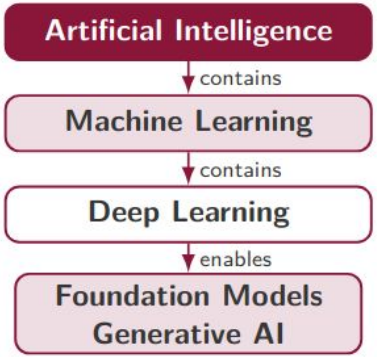
SCIENTIFIC INFERENCE
Infer Physical Parameters, Beyond Task-Specific Recognition



ACCELERATE DISCOVERY
Speed Up Simulations, Emulate Complex Physical Processes

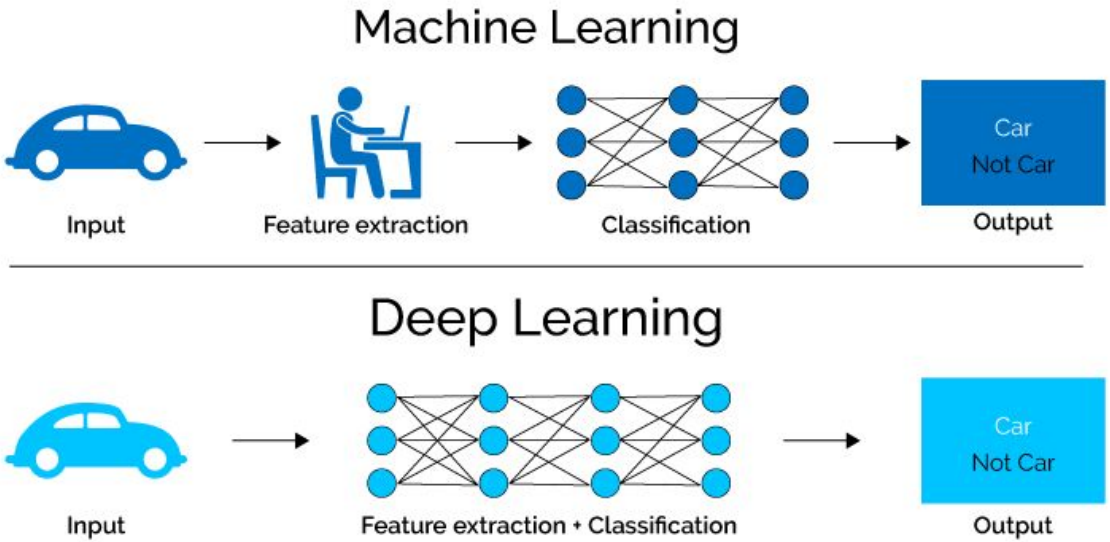
FIND NEW PHENOMENA
Anomaly Detection, Connect Archives for Research Assistance

In Astrophysics: source classification, photometric redshifts, event reconstruction, anomaly detection, simulation emulation, multimessenger association and research assistance.



- Artificial Intelligence is the broad field of building systems able to perform tasks that usually require human intelligence.
- Machine Learning is a subfield of AI where algorithms learn patterns from data rather than being explicitly programmed.
- Deep Learning is a subfield of Machine Learning based on neural networks with many layers, able to learn complex representations.
- Foundation and generative models are large deep-learning models trained on broad data sets and adapted to many downstream tasks

Machine Learning versus Deep Learning



XenonStack

Applied Data Analytics Today

Today's Data

- **High Granularity**: Observations and measurements are detailed and fine-grained in real-world phenomena.
- **Complexity**: Data is often high-dimensional, heterogeneous, and intricate.
- **Large Scale**: Data volume is significant, with some experiments generating terabytes of data per instance.

Analytical Challenges

- **Data-Driven Discovery**: Using data to drive new scientific insights.
- **Model Refinement and Development**: Improving existing models or creating new ones to better explain real-world phenomena.
- **Complexity of Analysis**: Analysis becomes challenging due to data size and complexity, necessitating advanced tools and techniques.
- **Need for New Tools**: New data analysis and intelligence techniques are essential to handle these challenges effectively.

Machine Learning

- **Efficiency in Analysis**: Machine learning enables rapid analysis using pre-existing algorithms, as long as you maintain control over the process.
- **Applied Research Focus**: In this course, we will use machine learning from an applied research perspective, focusing on practical applications.

Note:
We will not develop new machine learning algorithms but will focus on applying existing ones.

Data Science

- **Rapidly Evolving Field**: Data science is a fast-growing area, driven by scientific needs across various domains (e.g., biology, medicine).
- **Cross-Domain Applicability**: An algorithm developed for one field, like biology, could potentially be adapted for use in another domain.

Machine Learning versus Programming. And Unsupervised Learning

Programming:

In traditional programming, you start with data and a set of rules. These rules are manually defined and applied to the data, producing the desired results.

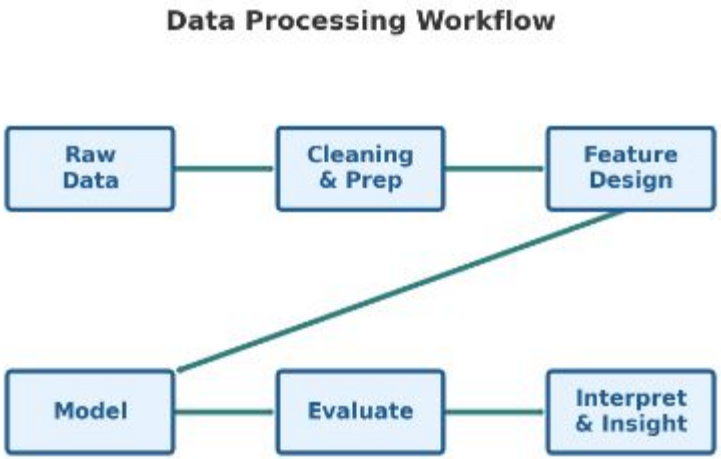
Supervised Machine Learning:

- In supervised learning, we have both the input data and the expected results (labels).
- We use this labelled data to train a machine learning model, which learns the rules that map inputs to outputs, allowing us to generalize and make predictions on new data.

Unsupervised Learning:

- In unsupervised learning, we have only the input data without any labels or predefined rules.
- The goal is to discover the **underlying structure of the data**, such as finding clusters or reducing dimensionality, to understand patterns, relationships, or similarities within the data.

The Modern Data Science Workflow



- **Raw Data → Cleaning**
  Real-world data is often imperfect. Cleaning ensures correctness, consistency, and usability.
- **Feature Design**
  We transform raw variables into informative features that help models understand patterns.
- **Model Building**
  Using machine learning methods, we learn relationships within the data.
- **Evaluation**
  We assess the model's performance on unseen data to check its generalization ability.
- **Interpretation & Insight**
  Beyond accuracy, the goal is to extract scientific understanding and meaningful conclusions.

Supervised and Unsupervised Learning

Supervised Learning

Definition: The model learns and makes inferences from **labelled data**.

How it Works: During training, the model is provided with both input data and the correct output labels, allowing it to learn the desired patterns or relationships.

Common Uses: Used primarily for **Classification** (categorizing data) and **Regression** (predicting continuous values).

Suppose we have 10,000 images of trees and houses. In supervised learning, each image is labelled as either “tree” or “house” (or as "0" and "1").

- The model uses these labels to learn the differences and similarities between the categories, allowing it to identify trees and houses based on known labels.

Unsupervised Learning

Definition: The model learns and makes inferences from **unlabelled data**.

How it Works: The model analyses the features of the input data to identify patterns or clusters, grouping similar data points without prior knowledge of labels.

Common Uses

Often used for:

- **Visualization and Clustering**: Grouping similar items for easier analysis and subsequent labelling.
- **Generative Modelling**: Learning data distributions to generate new samples.

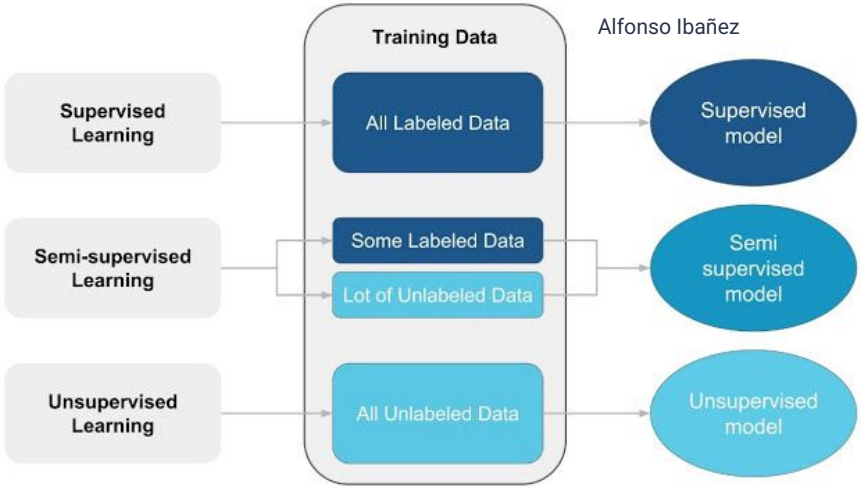
With unlabelled images of trees and houses, the model clusters similar images together based on features, such as shape and colour, without knowing the true label. It simply groups them by visual similarity.

Semi-supervised learning (SSL)

Supervised learning requires a large amount of labeled data, which is often difficult to obtain.

Semi-supervised learning (SSL) is a method that combines elements of supervised and unsupervised learning, allowing us to benefit from both approaches.
In SSL, **we use a mix of labeled and unlabeled data**:

- This approach exploits a small amount of labeled data and a large amount of unlabeled data to train the model, making it practical for situations where labeling all data would be too time-consuming or costly.



Example Workflow:

Suppose you have a large dataset with mostly unlabeled data, and labeling each item individually would be impractical:

1. **Label a small portion** of the data and use it to train an initial model.
2. Use this model to **pseudo-label** the remaining unlabeled data.
3. **Re-train the model** on both the labeled and pseudo-labeled data for improved accuracy.

This method enables effective learning even with limited labeled data, making SSL especially useful in real-world applications with large datasets.

Self-Supervised Learning

Self-supervised learning aims to mimic the way human infants learn about their environment without direct supervision.
The model learns from **unlabelled data** by creating **pretext tasks**:

- These tasks involve generating **pseudo-labels** from the data itself, such as predicting parts of an image from other parts, filling in missing words in a sentence, or identifying relationships between data points.
- **By solving these tasks, the model learns useful representations of the data** that capture meaningful patterns and structures.

In self-supervised learning, the model essentially creates its own labels, building a labelled dataset from the original unlabelled data. It then uses these labels to learn in a way similar to supervised learning, allowing it to generalize well to new tasks.

Reinforcement Learning

In reinforcement learning, an agent interacts with an environment by performing actions and receives rewards (or penalties) based on its performance.

This approach is effective in scenarios where there is **continuous feedback** for the agent's actions.

Applications include:

- **Gaming** (e.g., training agents to beat world champions)
- **Robotics** (e.g., teaching robots to walk)
- **Self-driving cars**

Reproducibility: A Pillar of Research

Reproducibility is essential in scientific data analysis, as it ensures that:

- **Scientific results are credible**
  People must be able to repeat your experiment and obtain consistent conclusions.

- **Debugging is possible**
  Re-running exactly the same pipeline helps identify bugs and understand changes in model behaviour.

- **Collaboration is smoother**
  Colleagues (and your future self) must be able to re-run the notebook and reproduce your results.

- **Long-term projects remain coherent**
  PhD analyses span years. Reproducibility guarantees you can re-run older experiments without confusion.

How to Ensure Reproducibility in Practice

1. Fix random seeds
Ensures consistent train/test splits, shuffling, weight initialization, and model training behaviour.

2. Structure notebooks clearly
Keep cells in order, avoid re-running cells randomly, and place imports + seed settings at the top.

3. Keep dataset metadata
Record dataset version, preprocessing steps, feature definitions, and acquisition conditions.

4. Save model versions
Store model architecture, weights, hyperparameters, and preprocessing objects (scalers, encoders).

5. Use consistent train/test splits
Fix the random state and save or log the indices used in each split.

In **Supervised Machine Learning**,
we need three things:

Input Data Points:

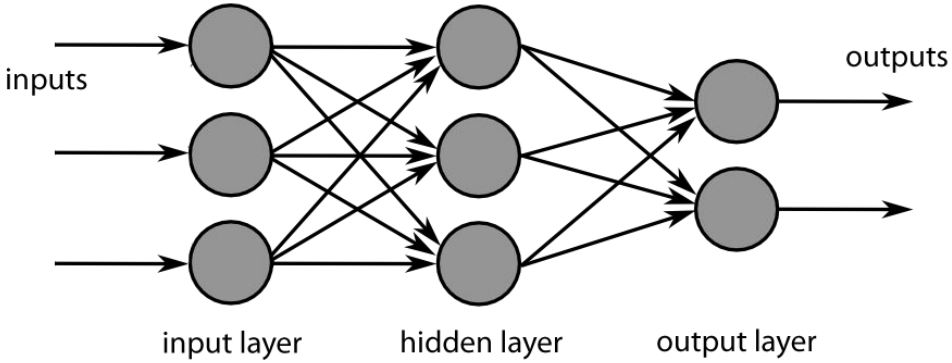
These can include tables, catalogues, time series, images, sound files, or videos.

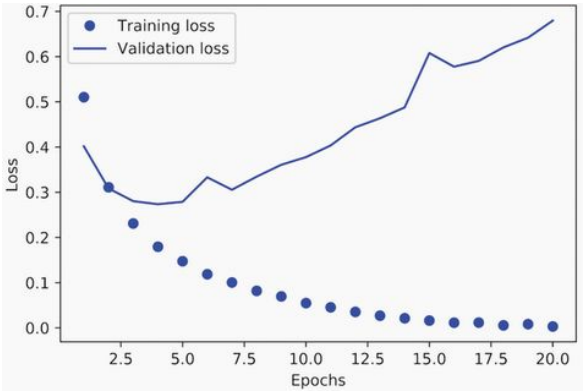
Expected Output Examples:

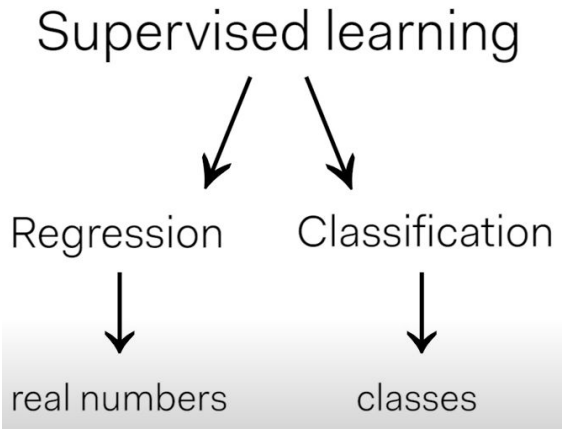
This is what the data represents, often referred to as “labels.” Labels guide the model in understanding the desired outcome.

A Measure of Performance:

To assess how well the algorithm is performing, we compare its output to the expected output. This comparison provides feedback, helping to fine-tune the model and improve its predictions. This adjustment process is what we refer to as **learning**.







Transforming Data through Machine Learning and Deep Learning

The primary challenge in machine learning and deep learning is to **transform data meaningfully**. This means learning useful representations of the input data that bring us closer to the desired output.

- **Deep Learning** focuses on creating **multiple layers of increasingly meaningful representations**. These layers allow the model to capture complex patterns and nuances within the data.

Common Goals in ML & DL:

Classification of Data Samples:

- **Binary Classification**: Extracting signals for a yes/no decision.
- **Multi-Class Classification**: Separating data into multiple categories.

Property Reconstruction:

- Predicting continuous values on independent datasets (**Scalar Regression**).

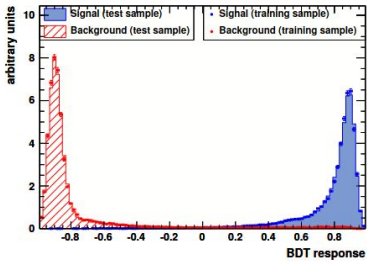
Data Generation and Representation Learning:

- Generating new data samples or learning high-quality data representations.

Supervised Learning: Binary Classification

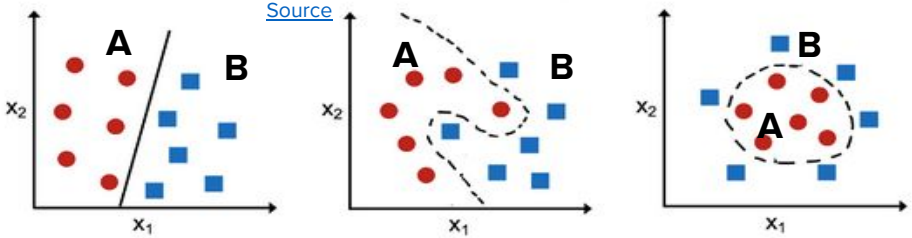
Binary classification is the task of classifying data into two distinct categories (A and B).

- **Objective**: Separate data into two classes (A and B), also known as **binary classification**.
- **Capability**: Machine learning can identify both **linear** and **non-linear relationships** in data, allowing for flexible decision boundaries.
- **Mutually Exclusive Classes**: The classes are exclusive, meaning that each data point belongs to only one class.
    - The **sum of predicted probabilities** for both classes is always 1.



Y. Becherini *et al.*, 2011 (Astrop. Phys., Vol 34, 12, 2011, 858-870)

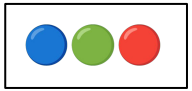
Source



Examples of Binary Classification Tasks:

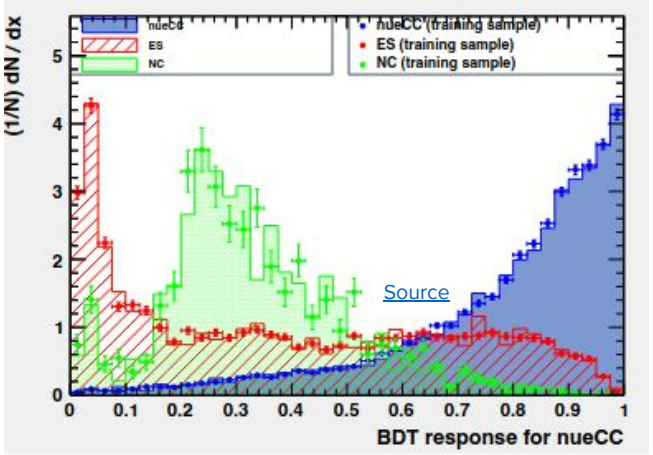
- Classifying **healthy food** vs. **junk food**
- Identifying **spam emails** vs. **non-spam emails**
- Differentiating **gamma rays** vs. **protons**
- Detecting **credit card fraud**
- Analyzing **Twitter sentiment** (e.g., Happy vs. Sad Tweets)
- Distinguishing **spiral** vs. **elliptical galaxies**

Supervised Learning: Multiclass Classification



In multiclass classification, we aim to classify data into more than two categories (e.g., classes A, B, and C).

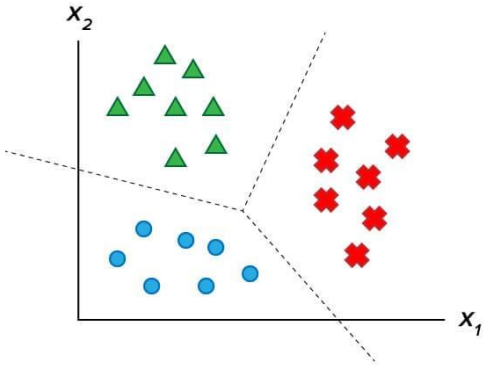
- **Objective**: Classify data into multiple distinct classes, also known as **multiclass classification**.
- **Mutually Exclusive Classes**: Each data point belongs to only one class.
    - The **sum of predicted probabilities** across all classes is equal to 1



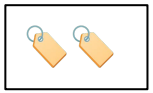
Source

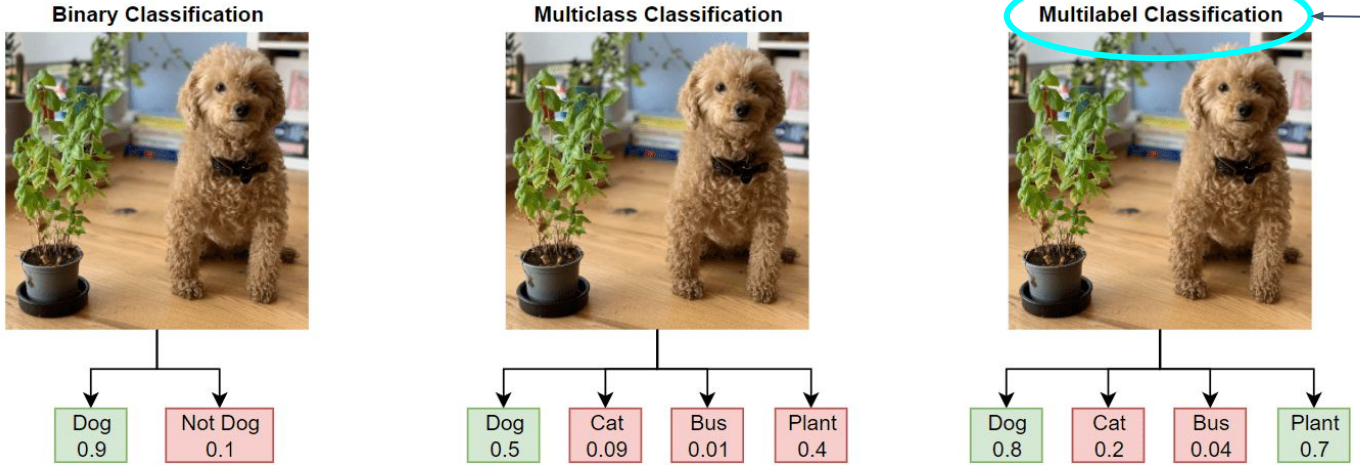
Examples:

- Recognizing digits (0 to 9)
- Identifying characters (A to Z)
- Classifying images of various objects (e.g., animals, fruits, buildings)



Supervised Learning: Multi-label Classification





Mathworks

Multi-label classification allows multiple labels to be assigned to each instance of data.

Each label is treated as a separate binary classification problem, meaning that each label is evaluated independently.

For each label, the model predicts a probability between 0 and 1.

- If the predicted probability for a label is greater than 0.5, the label is assigned.

The probability assigned to one label does not affect the probabilities of other labels, as each label prediction is independent.

Examples of Multi-Label Classification:

- **Movies**: Tags like horror, romance, adventure, and action can be assigned in various combinations (e.g., horror + action, romance + adventure).
- **Images**: Multiple objects in an image can each be identified with separate labels (as shown in the figure).
- **Research Articles**: Papers can have multiple tags, such as computing + social sciences, physics + mathematics, or machine learning + biology.

Supervised Learning: Regression

Regression is a type of problem where the output variable is a **real continuous value**.

Many models can be used for regression, with **linear regression** being the simplest. In linear regression, the model tries to fit the data with the best-fitting line (or hyperplane) that captures the relationship between the independent variables and the dependent variable.

You can perform regression to predict **a single output** or **multiple outputs** at the same time.

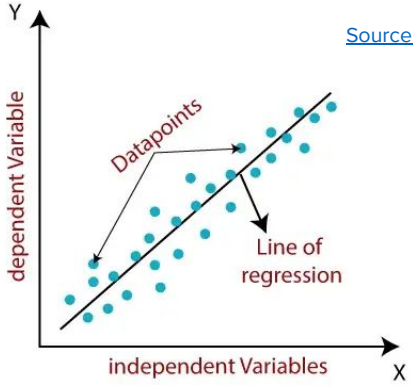
Examples of Single-Output Regression:

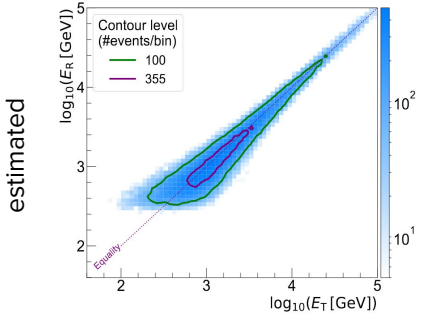
- Predicting a person's age
- Estimating house prices
- Calculating the energy of a particle
- Assessing the severity of a disease

Example of Multiple-Output Regression:

- Predicting both the degree of insulin resistance and the age of onset of Type-II diabetes in adults

Source





Y. Becherini

M. Senniappan, Y. Becherini et al, JINST (2021)

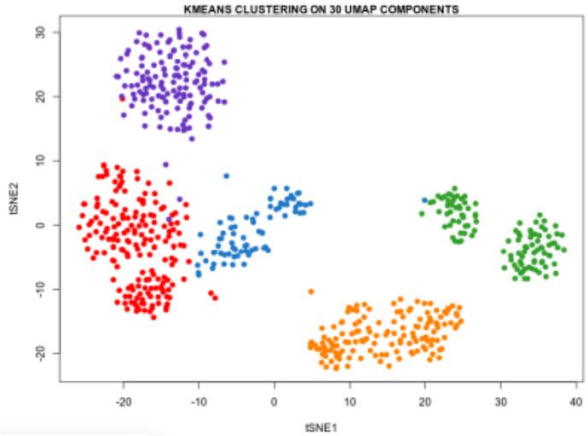
Unsupervised Learning & Generative Models

Unsupervised Learning

In unsupervised learning, the system learns patterns from **unlabelled data** without supervision.

Common Applications:

- **Clustering**: Grouping similar data points together.
- **Visualization and Dimensionality Reduction**: Simplifying data for easy visualization.
- **Anomaly Detection**: Identifying unusual or novel data points



Generative Models

Generative models aim to learn dense representations of input data, known as **latent representations**, which can be used to generate new data.

Autoencoders

- Efficiently create compressed, dense representations (latent representations) of input data.
- Useful for **dimensionality reduction** and **visualization**.
- Act as feature detectors and can generate new data that closely resembles the input data.

Generative Adversarial Networks (GANs)

GANs generate new data using two neural networks:

- **Generator**: Attempts to create data similar to the training data.
- **Discriminator**: Attempts to distinguish between real and fake data generated by the generator.

Science: Astrophysics

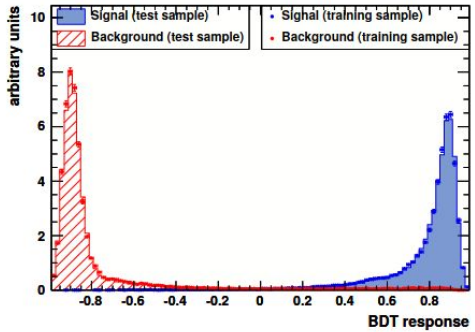
- **Machine Learning** is widely used in **Astrophysics** and **Particle/Astroparticle Physics**, with several successful applications in regression, classification, and generative models.

Applications in Astroparticle Physics:

- **Event Reconstruction**: ML is used to determine the properties of incoming "events" caused by particles from the Universe.
- **Phenomena Classification**: Helps in distinguishing between different astrophysical phenomena.
- **Synthetic Data Generation**: Generates additional data samples to enhance analyses and simulations.

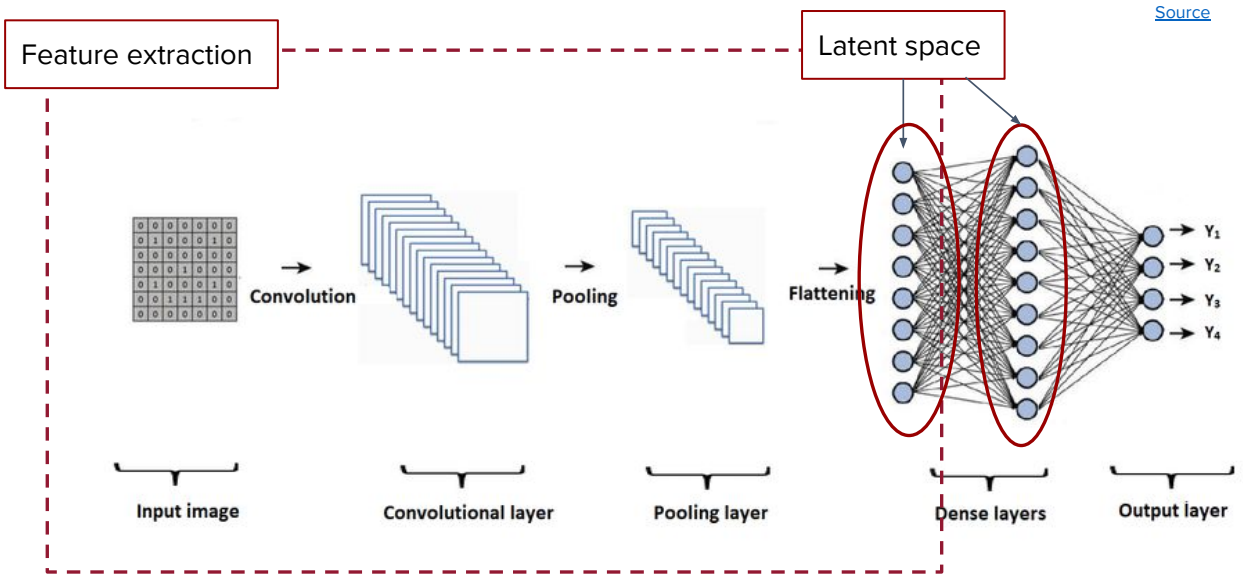
In Gamma-Ray Astronomy:

- **Advances in Classification**: Improved classification techniques have significantly enhanced detection sensitivity.
- **Increased Discoveries**: More astrophysical sources are now being identified and studied thanks to enhanced sensitivity.
- **Expanded Knowledge**: ML contributes to a deeper understanding of the energetic processes in the Universe.
- **High Classification Accuracy**: Achieves classification rates of up to **95% signal** with only **1% background** remaining.



Y. Becherini

Convolutional Neural Networks



Source

Classification

Binary Cross-Entropy

$\hat{y}$ is prediction, y is ground truth

$$f(y,\hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)]$$

Multi-Class Cross-Entropy Loss

$$f(y,\hat{y}) = -\sum_{c=1}^{M} y_c log(\hat{y}_c)$$

M: total number of class

Regression

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

(test set; predicted vaue; actual value)

Mean Absolute Error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Mean Squared Error

The latent space is an abstract multidimensional space that encodes a meaningful internal representation of externally observed events

Samples that are similar “in the external world” are positioned close to each other in the latent space.

Latent space

In the feature extraction phase, the model has captured the important patterns of the input that are needed for the image classification task.

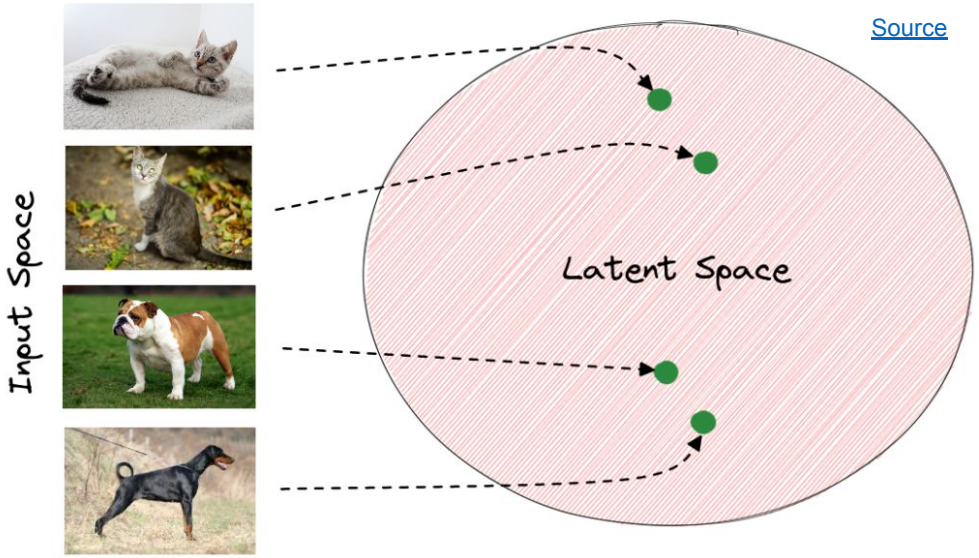
In the latent space, images that depict the same object have very close representations.

Generally, the distance of the vectors in the latent space corresponds to the semantic similarity of the raw images.

The green points correspond to the latent vector of each image extracted from the last layer of the model.

We observe that vectors of the same animals are closer to the latent space.

Therefore, it is easier for the model to classify the input images using these feature vectors instead of the raw pixel values.

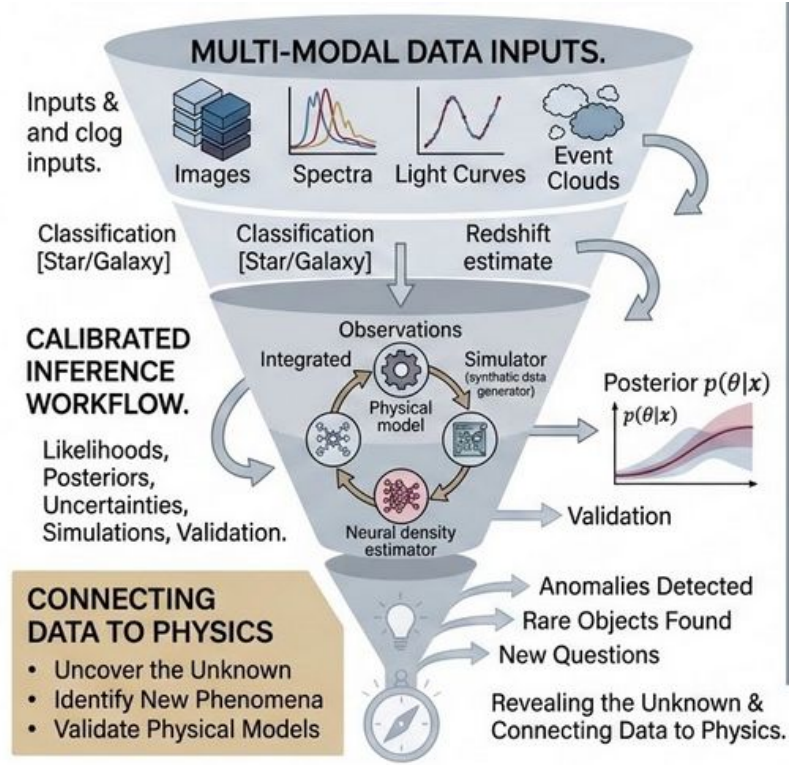


Source

But ... difficult to interpret the latent space meaning

Key concepts

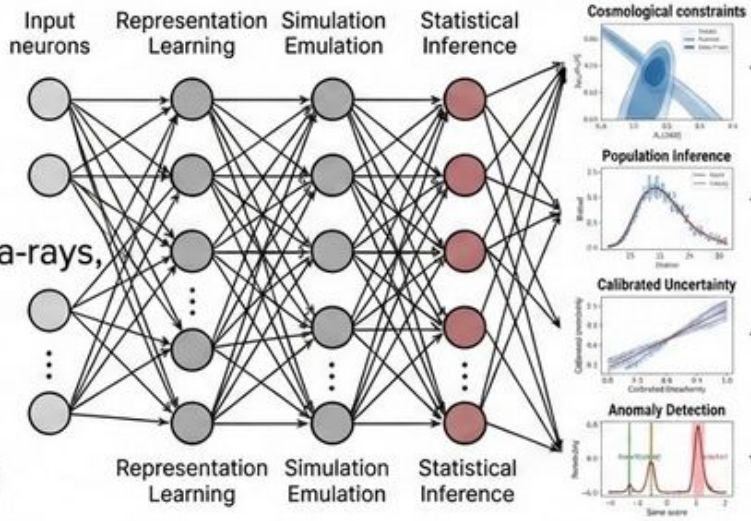
- Astrophysics now combines heterogeneous data, complex instruments and simulations.
- AI first appears as a predictive tool: classification, regression, ranking and pattern detection.
- Deep learning extends the input space: images, spectra, light curves and detector events.
- Prediction is not yet scientific inference.
- Physical claims require likelihoods, posteriors, uncertainties, simulations and validation.
- The shift is from ML as a final predictor to ML inside calibrated inference workflows.
- Foundation models aim at reusable representations across surveys, messengers and modalities.
- Discovery goes beyond classification: AI can reveal anomalies, rare objects and new questions.
- AI does not replace physical reasoning. It connects data, simulations and physical understanding



Modern Astrophysics Data Sources

- Images and sky maps
- Spectra and spectral cubes
- Light Curves and alert streams
- Catalogues and tabular data
- Event-level detector data (gamma-rays, neutrino events)
- Numerical Simulations
- Gravitational-wave signals
- Instrument response functions
- Papers/Databases and metadata

Physical Meaning Extraction



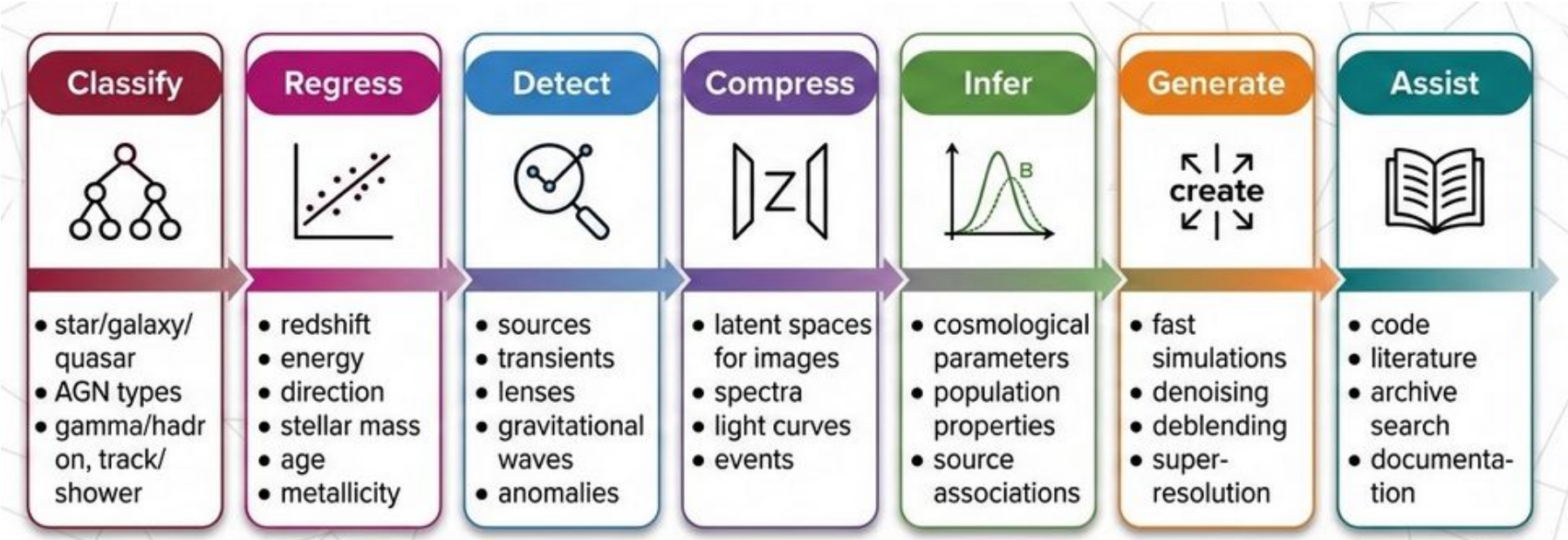
High-Dimensional Drivers

- Rubin/LSST
- Euclid, Roman
- JWST
- SKA
- CTA
- IceCube
- KM3NeT
- LIGO/Virgo/KAGRA

AI is useful when the relevant information is high-dimensional, weak, non-linear or difficult to compress by hand.

The limiting factor is increasingly the extraction of physical meaning, not data acquisition itself.

What do we ask AI to do in Astrophysics?



AI is useful when the relevant information is high-dimensional, weak, non-linear or difficult to compress by hand.

The limiting factor is increasingly the extraction of physical meaning, not data acquisition itself.

Learning paradigms: what supervision signal is available?

more explicit human labels / physical targets

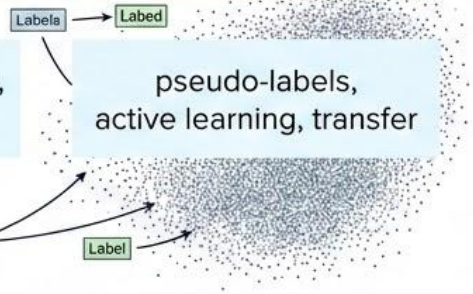
labels emerge from data, simulations or structure

Supervised learning
(x, y) → ŷ

classification, regression, calibrated predictors

Weak / semi-supervised
few labels + many x

pseudo-labels, active learning, transfer



Self-supervised learning
pretext signal from x

masked modelling, contrastive learning, next-token prediction

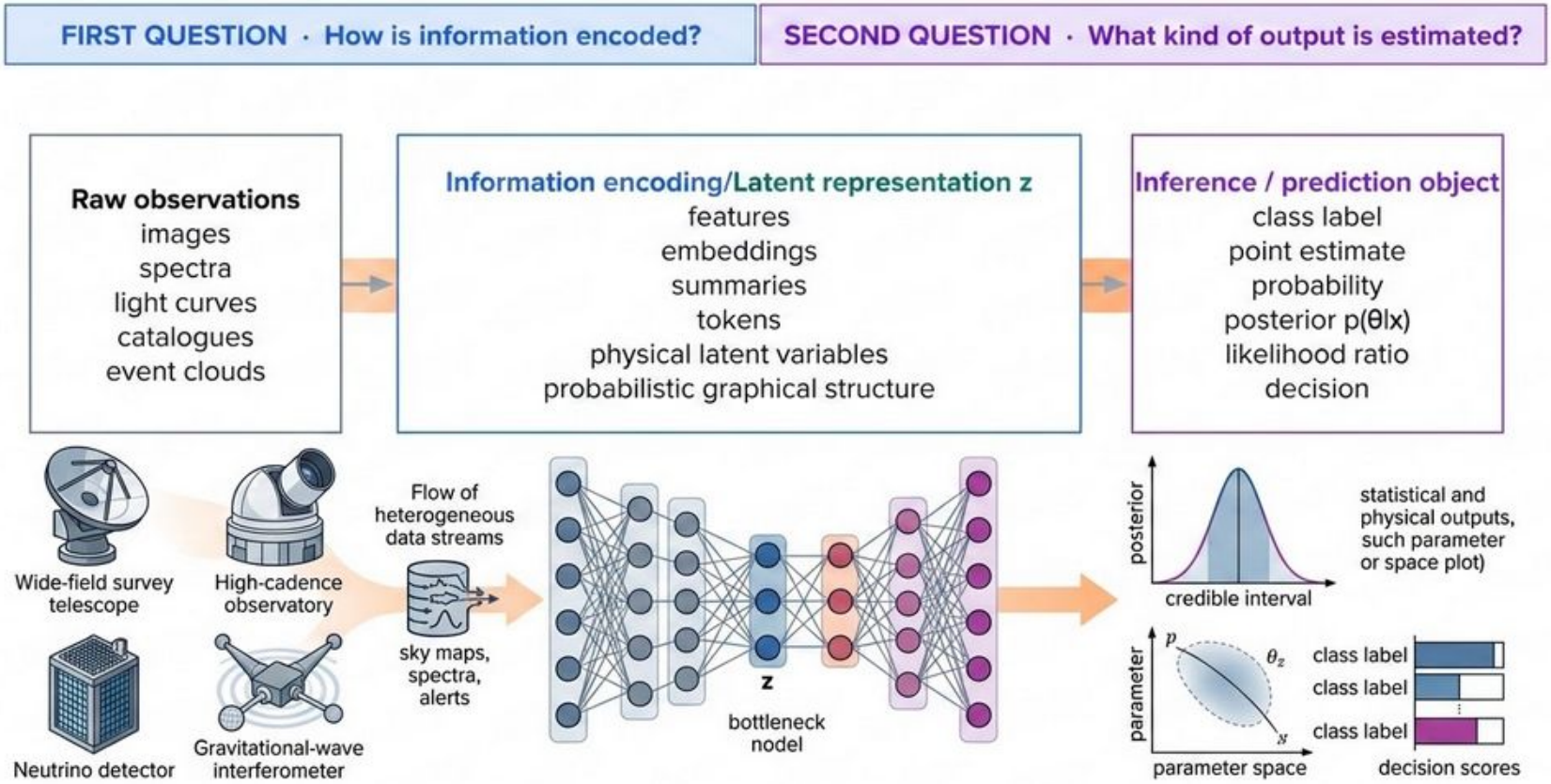
Unsupervised learning
structure in x

clustering, density estimation, anomaly search

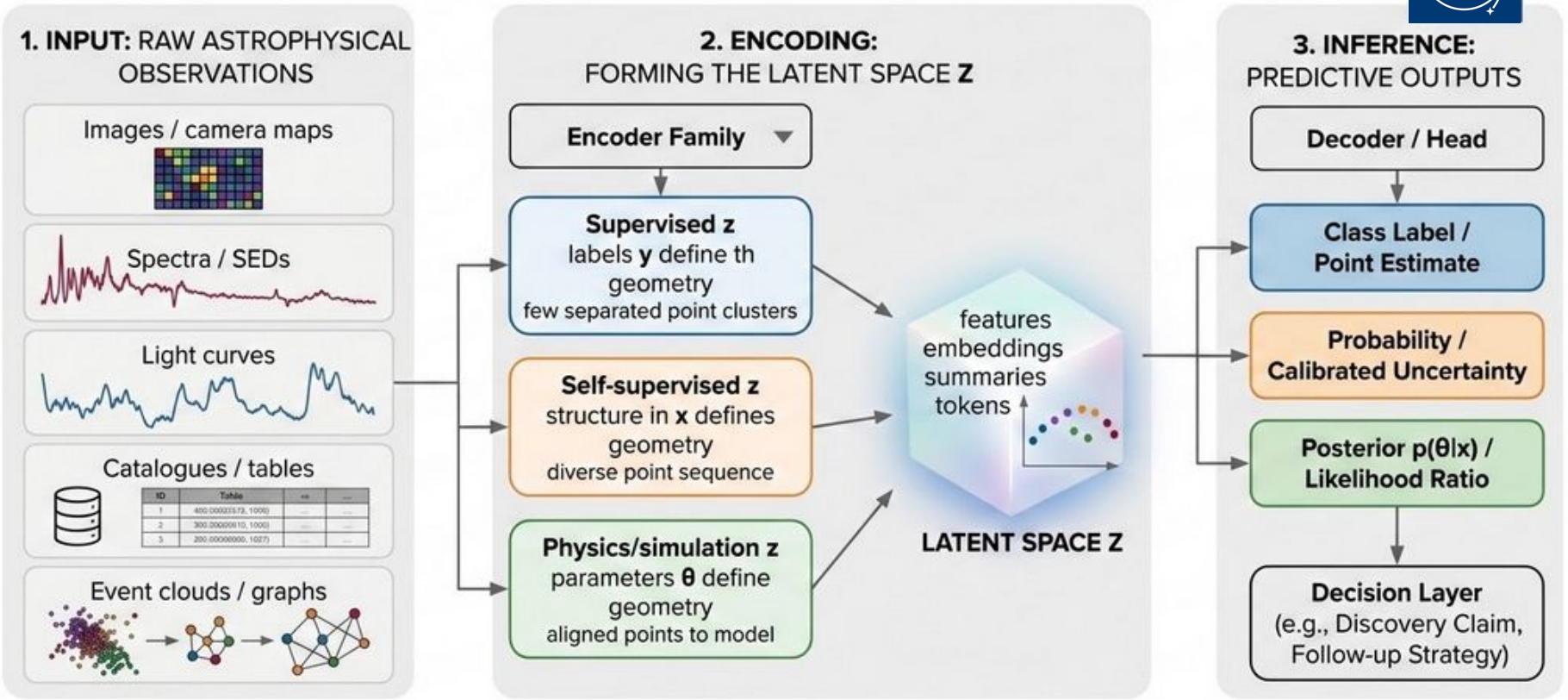
Astrophysics has traditionally used many supervised ML pipelines, as simulations and catalogues provide labels.

However, modern astrophysical data are increasingly large, heterogeneous, incomplete and weakly labelled, so self-supervised, semi-supervised and unsupervised methods are becoming central.

Two questions behind machine learning analysis

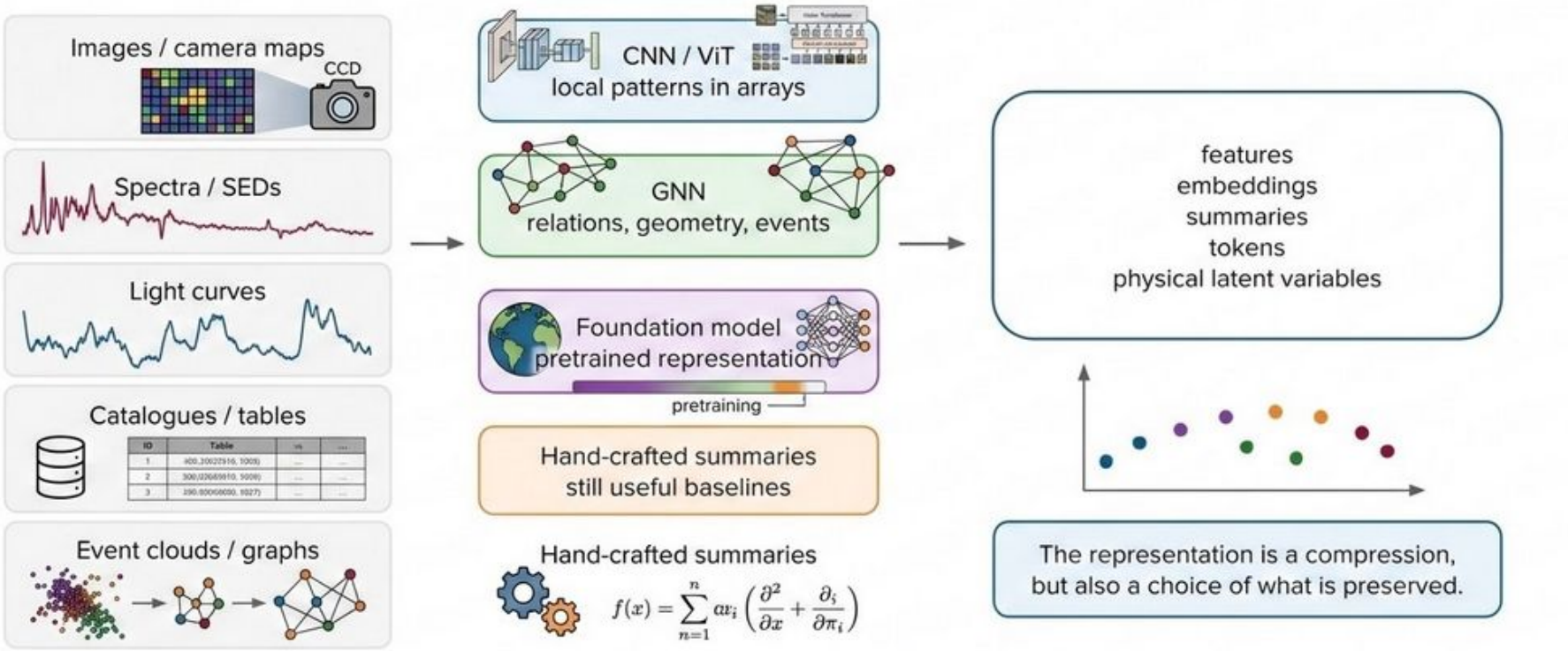


From raw data to scientific discovery



The encoding phase choice defines the representation, while the inference choice defines the scientific object

How is information normally encoded?



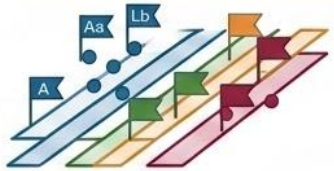
The first scientific question is: which representation contains the information needed for the physics task?

The richness of the latent space depends on the learning signal

Raw Data x

- images
- spectra
- light curves
- catalogues

Supervised z
labels y define the geometry

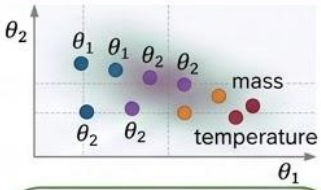


Classes or targets become axes of separation.
Excellent for the labelled task, but can hide variability not needed for y.

Self-supervised z
structure in x defines the geometry

Invariances, temporal context, masking or contrastive signals decide what is close.
Good when labels are scarce.

Physics/simulation z
parameters θ define the geometry

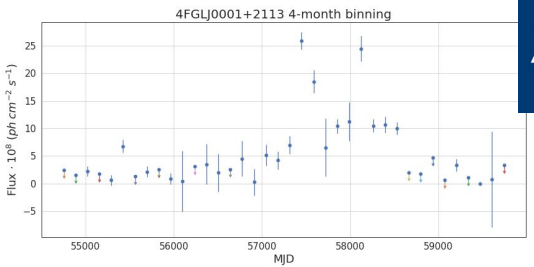


If simulations provide θ, the representation can align with physical parameters, nuisance effects or likelihood information.

No universal latent space exists: the geometry of z reflects the supervision, the augmentations, the simulator, and the loss.

A latent space can be rich but not automatically physical: interpretability comes from validation, calibration and links to models.

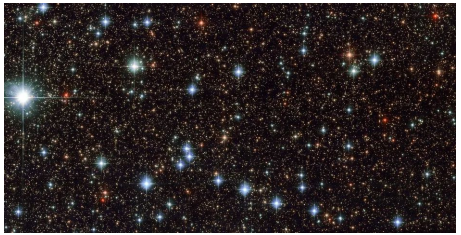
Deep Learning & Transfer Learning



Deep Learning

You use the data in your possession to train a model: light curves, images, etc

- **But the data you have might not be enough to train a good model, or you do not possess enough labelled data. Or it will require exorbitant computing times that you do not possess**
- Possible solutions: improve labelled data statistics, use GANs, take more data
- OR: Transfer Learning, especially if you use images or Time Series



Transfer Learning

Transfer Learning involves taking a **pre-trained model**, originally trained on a large dataset, and adapting it for a different but related task.

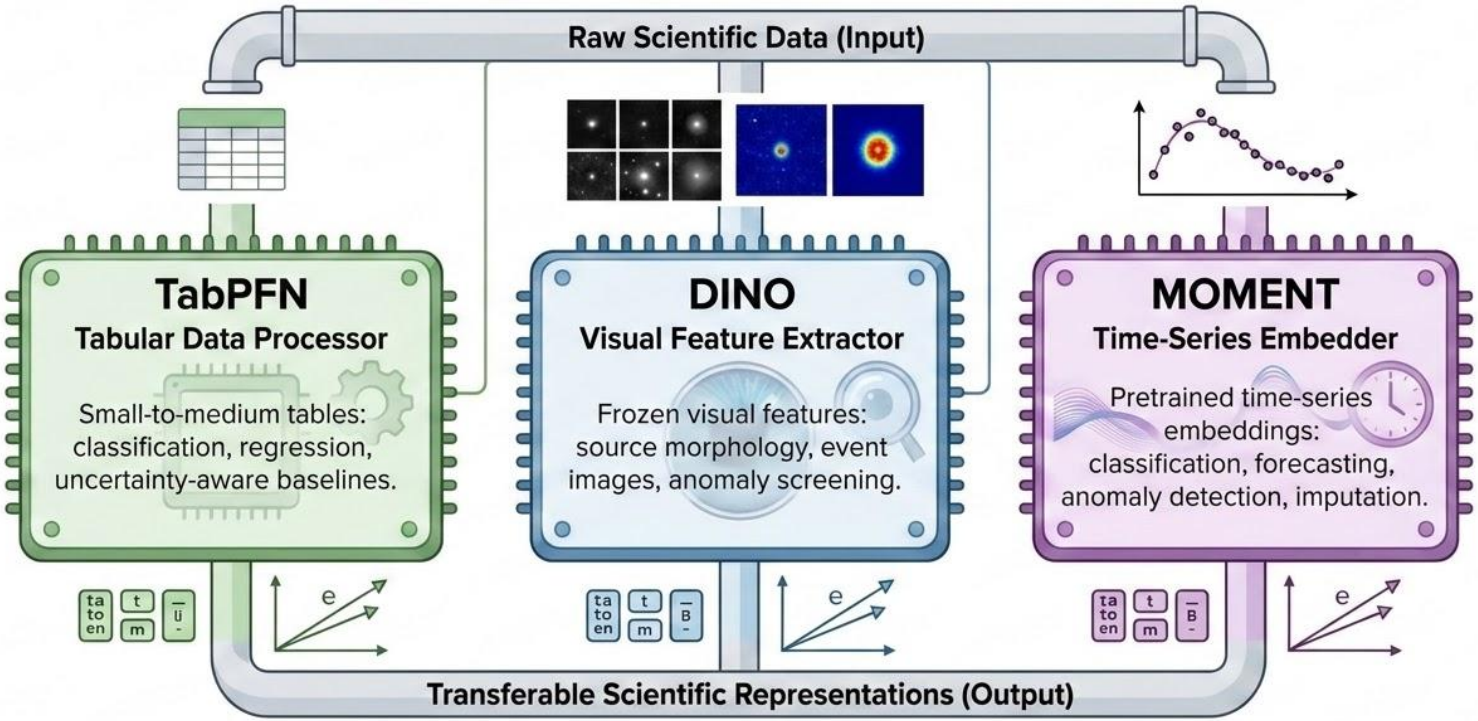
Fine-Tuning Techniques

Fine-tuning can include modifying the architecture, adjusting hyperparameters, and training with a smaller, domain-specific dataset to improve performance.

Benefits for Astrophysics

This approach significantly reduces training time and resource requirements, enabling researchers to use existing models for analyzing astrophysical data.

Foundation models as reusable encoders for scientific data



Foundation models are useful for small data samples, but they are limited to point-like predictions

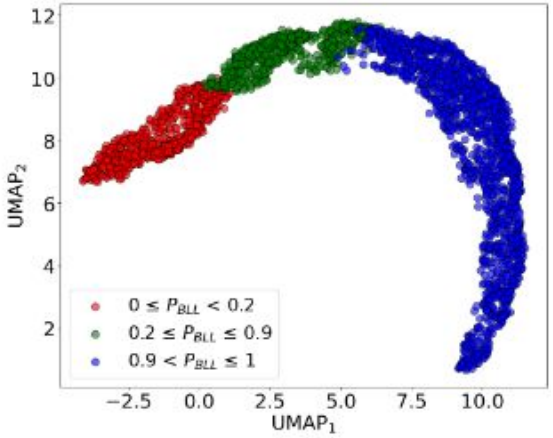
- TabPFN is a foundation model for small- to medium-sized tabular data.
- It is trained to perform supervised learning by in-context inference on synthetic tasks.
- It is especially relevant for astronomical catalogues with limited labelled samples.
- Typical uses: classification, regression, uncertainty-aware ranking and fast baselines.
- Caveat: excellent prediction does not automatically imply calibrated physical inference.

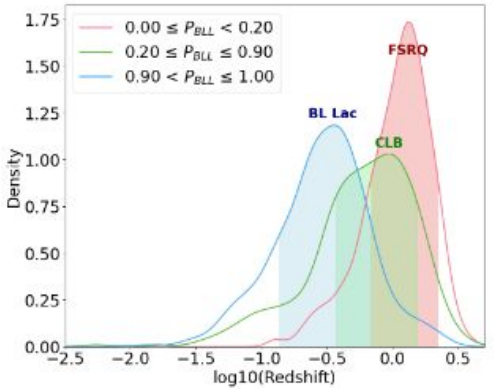
TabPFN is a good example of a foundation model outside images and text, directly relevant to catalogue-based astrophysics.

Hollmann et al., Accurate predictions on small data with a tabular foundation model, Nature 2025.

Blazar evolution beyond discrete classes

- We classify blazars from the Fermi 4LAC-DR3 catalogue, focusing on blazars of uncertain type.
- The analysis uses XGBoost as a strong supervised benchmark and TabPFN as a foundation model for tabular data.
- The high-dimensional latent space is reduced to a two-dimensional representation of the blazar population.
- The result suggests a continuum between FSRQs and BL Lacs, with Changing-Look Blazars as possible transitional objects.
- Here the ML output is not only a label. A probability score plus latent-space structure becomes a way to discuss physical evolution.
- Oukacha & Becherini,
  Towards a unified scheme of blazar evolution, A&A,
  arXiv:2507.03088.





- DINOv3 is a large self-supervised vision foundation model.
- It learns visual representations without class labels or text supervision.
- Its strength is dense visual features: useful for detection, segmentation and morphology.
- In astronomy, DINO-like models are attractive for galaxy morphology, lenses, artefacts and survey images.
- Caveat: pretraining on natural images may not encode astronomical noise, PSF, selection effects or physical units.

DINOv3 illustrates the transition from task-specific CNNs to reusable visual representations.

Siméoni et al., DINOv3, arXiv:2508.10104.

MOMENT

- MOMENT is a family of open foundation models for general-purpose time-series analysis.
- It is pretrained with masked time-series modelling on heterogeneous time-series data.
- Relevant tasks include forecasting, classification, anomaly detection and representation learning.
- Astrophysical targets: light curves, AGN variability, transients, detector rates and alert streams.
- Caveat: astronomical time series are irregular, heteroscedastic and often have informative missingness.

MOMENT is a useful bridge between general time-series foundation models and time-domain astronomy.

Goswami et al., MOMENT: A Family of Open Time-series Foundation Models, arXiv:2402.03885.

Hands-on Colab notebooks (Focus on Foundation Models and Simulated-Based Inference)

1. Classification of Fermi blazars with the TabPFN Foundation Model
2. Galaxy type through the DINOv2 foundation model

Probabilistic Learning:

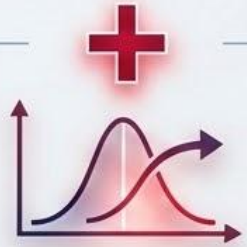
- Bayesian Neural Networks
- Knowledge-Informed NN
- Simulation-Based Inference

Bayesian Deep Learning is a field at the intersection of Bayesian statistics and Deep Learning.

It combines the strengths of deep learning, known for its ability to learn **complex patterns from large amounts of data**, with the **probabilistic** and **uncertainty modelling capabilities** of Bayesian methods.

Source

Deep Learning: **Deep Neural Networks** are particularly good at handling high-dimensional data like images, videos, text, or sound.

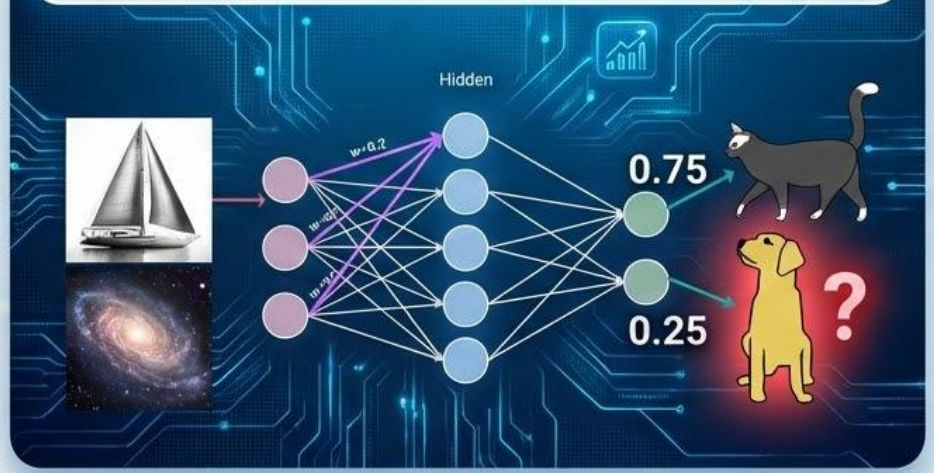


Bayesian Statistics: A statistical paradigm involving the use of **probability distributions** to **express uncertainty** about the state of knowledge using the **Bayes' theorem**, which describes how to update the probabilities of hypotheses when given more evidence.

The importance of having confidence on the Model

The Problem: Traditional Deterministic Approach

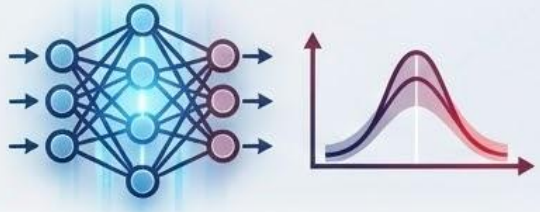
- In the traditional deterministic approach, when we train a model, we often **clean and pre-process the data** leading to a sample with minimal noise or ambiguity.
- Then, when we deploy the model, it sees edge data samples, samples with a high noise, etc so our model is completely unequipped to handle these cases.
- How much can we be confident on the output of my model?



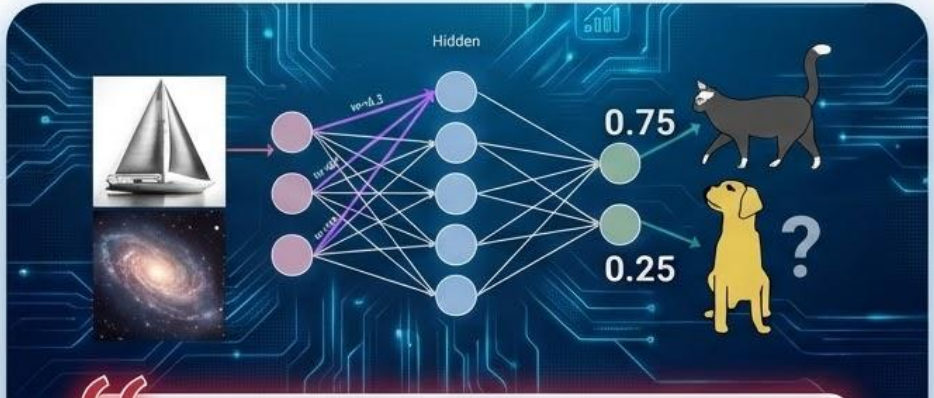
The Goal & Solution

- **The Goal:** Learn if we can trust the output
- Traditional deep learning models:
  - Tend to propagate biases from training
  - Are often susceptible to failures on brand new out-of-distribution data instead.
- We need:
  - Models that can reliably and quickly estimate uncertainty in the data that they are seeing as well as the outputs that they're predicting.

- **The Solution: Probabilistic Learning**
  Goal: Teach our neural networks to predict probability distributions instead of purely deterministic point outputs and how formulating our neural networks like this can allow us to model one type of uncertainty.



The Problem: Traditional DL & Overconfidence



> The model should have answered us **"In this case, I do not know"**

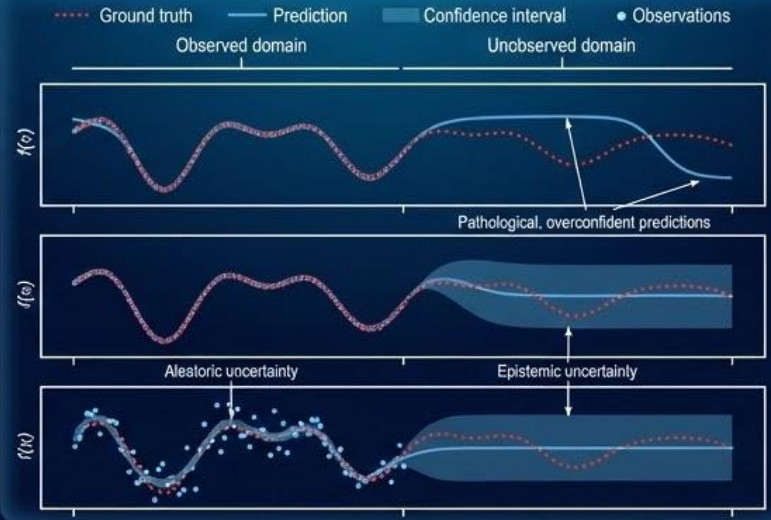
What are the different types of uncertainty?

Our traditional DL model captures the uncertainty in data, it does not capture the uncertainty in the predictions themselves

The Solution: Understanding Uncertainty Types

Types of uncertainty

1. Uncertainty in data (Aleatoric or Statistical Uncertainty)
2. Uncertainty in the model (Epistemic Uncertainty)



Source

Aleatoric and Epistemic Uncertainty

Aleatoric Uncertainty

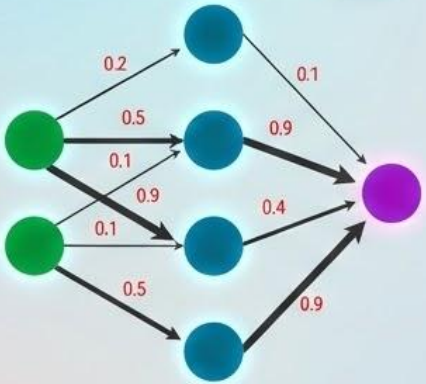
- Describes the statistical uncertainty arising from the inherent randomness or variability in the data
- It is characterized by the uncertainty that comes from the noise or natural variability in the system being modelled
  ➔ High uncertainty when the input data is noisy
- Cannot be reduced by adding more data
- Can be learned by using NN

Understanding these uncertainties is crucial for designing robust models and for interpreting model predictions in deep learning.

Epistemic Uncertainty

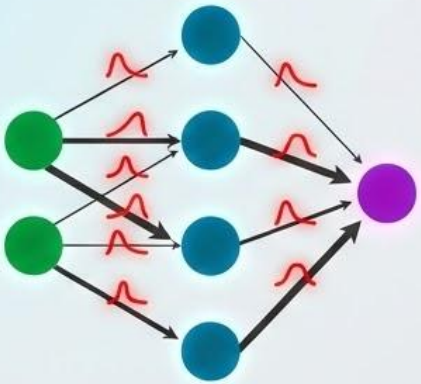
- Epistemic uncertainty, or model uncertainty, stems from a lack of knowledge and the inherent limitations of a model in capturing data complexity.
- Describes the confidence in the prediction
- It can be reduced with more data or a more suitable model, as it's rooted in the model's structural limitations, parameter estimates, and built-in assumptions
  ➔ High uncertainty when missing training data
- Can be reduced by adding more data (adding boat samples)
- Much more challenging to estimate

Deterministic Neural Network



- A given set of weights passing in one input to the model multiple times will yield the same output.
- Each weight is a single deterministic number.

Bayesian Neural Network



- Every single weight is represented by a probability distribution.
- When we pass an input to our NN we will sample from a point in this distribution for every single weight.
- Every time we feed in an input to the model, we're going to have a slightly different output depending on the samples of our weights.
- Learn the distributions of each and every weight (and bias) to then learn the uncertainty of our model (the epistemic uncertainty).

Key Concepts

- Deterministic neural networks have fixed weights and biases, while Bayesian Deep Learning uses distributions for these parameters.
- Bayesian Deep Learning allows for uncertainty in predictions and prevents overconfidence in model outcomes.
- Bayes' Theorem is employed to update beliefs about model parameters as new data is observed, impacting model calibration and uncertainty reduction.
- Bayes' Theorem is used to incorporate prior knowledge, allowing models to learn effectively even with limited data and avoid learning spurious patterns.
- The prior in Bayes' Theorem acts as a regularization factor, helping the model to a prefer simpler explanations and avoid overfitting to the data.

Priors & Guidance

The choice of prior distributions is crucial and should be guided by

- Domain knowledge
- Model characteristics/complexity
- Computational considerations

The goal is to select priors that reflect a balance between model flexibility and data-informed constraints.

Traditional vs. Bayesian Approach

Traditional approach

- Weights and biases are fixed
- Do not account for uncertainty in data
- Not ideal for small datasets

Bayesian approach

- Weights and biases have distributions
- Can account for uncertainty in data
- Suitable for small datasets due to uncertainty modelling

Bayes's Theorem in Supervised Learning

During Training, we adjust the weights to maximize the posterior distribution. Direct computation of this posterior is computationally intractable, so this It, so this requires approximation techniques like Markov Chain Monte Carlo (MCMC) and Variational Inference (VI)

Posterior Distribution

This is the posterior distribution of the weights given the input features and the output targets. This is what we're interested in finding out. It represents our updated belief about the weights after observing the data.

Likelihood Function

The likelihood function. This is the probability of observing the targets **Y** given the input features **X** and a specific set of weights **W**. In a neural network, this is determined by the model's architecture and the activation functions. It quantifies how well the model with certain parameters explains the observed data.

$$P(\mathbf{W}|\mathbf{X},\mathbf{Y}) = \frac{P(\mathbf{Y}|\mathbf{X},\mathbf{W})P(\mathbf{W})}{P(\mathbf{Y},\mathbf{X})}$$

Prior Distribution

The prior distribution of the weights. This represents our initial belief about the distribution of the weights before observing any data. The choice of the prior can be informed by domain knowledge or other considerations.
The training starts with these prior distributions, representing initial beliefs about the weights before observing any data.

Marginal Likelihood (Evidence)

The marginal likelihood or evidence. This is the probability of observing the targets Y given the input features X, marginalized over all possible weight configurations. It's often a complex integral over all possible values of W and typically intractable for neural networks. In practice, it's usually treated as a normalizing constant.

Posterior ∝ Likelihood × Prior

Predictions are made by averaging over multiple samples from the posterior distribution of the weights, allowing the BNN to estimate the uncertainty of the prediction.

Posterior approximation through sampling

Posterior ∝ Likelihood × Prior

The Problem: Intractability

Direct computation of the posterior distribution is often intractable due to high dimensionality and complexity.

The Solution: Sampling Methods

- Markov chain Monte Carlo (MCMC)
- Variational Inference (VI)

Loss function: Evidence lower bound (ELBO)

$$\mathcal{L}(\phi, \theta; \mathbf{x}) := \mathbb{E}_{\mathbf{z} \sim q_\phi(\cdot|\mathbf{x})} \left[\ln \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]$$

- $\mathbb{E}$ represents the expectation
- $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational approximation of the posterior distribution, a simpler distribution parameterized by θ that approximates the true posterior distribution
- $p(\mathbf{x}, \mathbf{z})$ is the likelihood of the observed data x and the latent variable z quantifying how well the model explains the data

Predictions & Uncertainty Estimation

To make predictions, the BNN uses the sampled posterior distributions of the parameters. Each sample represents a possible set of parameter values. Predictions are made by averaging over the predictions from multiple samples, providing a measure of uncertainty.

The sampled posterior in a BNN represents many possible instances of the model, each reflecting a different plausible set of parameters conditioned on the data. This leads to a range of predictions, capturing the uncertainty inherent in the model's understanding of the data.

From weights posteriors to the model output

After training, we do not have a single set of weights but a distribution over plausible weight configurations. Each of these corresponds to a different function mapping X to Y.
The predictive distribution is obtained by averaging the predictions of all these functions, weighted by how probable they are under the posterior.

The weights posterior

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, W)\, p(W|X, Y)\, dW$$

$p(y^*|x^*, W)$ is the probability of observing a value y value y^* given that the input is x^* and the data-generating mechanism is defined by the model parameters W.

- X, Y Observed training data
- x^* New input
- y^* Unknown

Choice of the prior

Common prior distributions include Gaussian priors, Laplace priors, and more, each suitable for different scenarios.

Informative priors incorporate existing knowledge into the prior distribution, while uninformative priors assume minimal initial knowledge.

Why are Priors Important?

Priors capture existing knowledge and impact model outcomes.

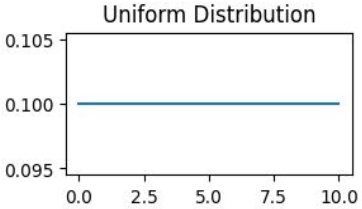
Guidance for Prior Selection

Priors should be informed by domain knowledge, model characteristics, and data variability.

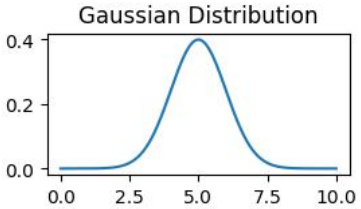
Crucial Impact on Model Performance

Inappropriate priors can lead to biased results and poor predictions.

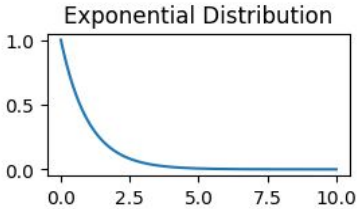
Prior Distribution Shape and Input Data Relationship



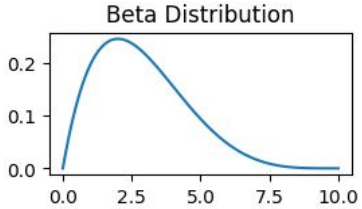
Indicates equal belief in all values within the range.
Suitable for scenarios where there's no initial preference for any value within a certain bound.



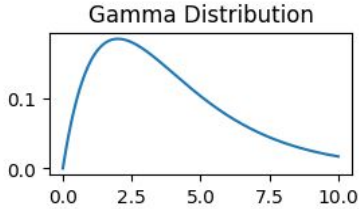
Indicates that values close to the mean are more likely.
Suitable for data that clusters around a central value with possible deviations.



Useful for modelling the time until an event occurs.
Suitable for data where large values are less likely than smaller values.



Versatile shape, can be symmetric or skewed.
Can represent a wide range of beliefs.

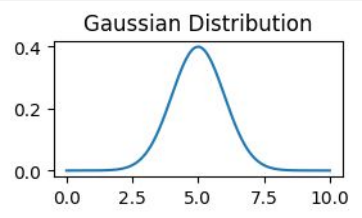


Good for modelling continuous, positive-valued data.
Skewed shape, can adjust with parameters.

The shape of the prior distribution for weights should reflect the scale and variability of the data and should align with the characteristics of the data.

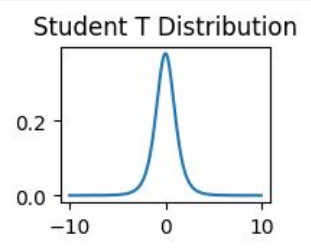
Prior Distribution Shape and Network Behaviour

Overfitting Control



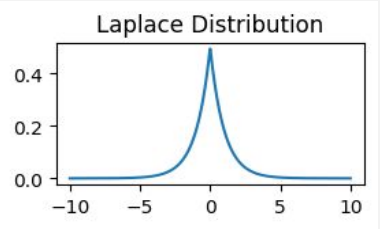
Choice of Gaussian Priors with small variance as a regularization term to penalize large weights and prevent overfitting. More on this on next slide.

Uncertainty Quantification



Using Heavy-Tailed Priors like the Student's t-distribution to allow the network to express higher uncertainty in predictions.

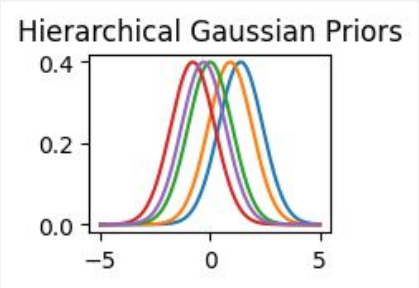
Sparse Feature Selection



Laplace Priors encourage sparsity in the network weights.

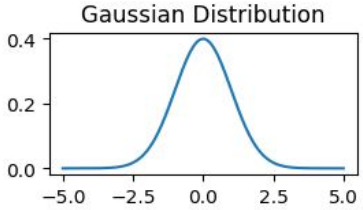
This property is particularly beneficial in scenarios involving high-dimensional data, where it aids in feature selection, enhancing both the model's performance and interpretability.

Robustness to Data Shifts



Hierarchical Priors or Mixture-of-Gaussian Priors allow for more flexible modelling of weight distributions and can adapt to shifts in data distribution.
A simple case involves a Gaussian distribution for weights, where the mean of this Gaussian is itself drawn from another Gaussian distribution.

Gaussian Priors



$$J_{regularized} = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right)\right)}_{\text{binary cross-entropy cost}} + \underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l=1}^{L}\sum_{i=1}^{n^{l}}\sum_{j=1}^{n^{l-1}} W_{j,i}^{[l]2}}_{\text{L2 regularization cost}}$$

Nature of Gaussian Priors

Gaussian priors are centred around zero, with smaller weights being more probable. The variance of the bell-shaped curve determines how quickly probabilities decrease as we move away from the mean.

Regularization Effect

Using a Gaussian prior is equivalent to adding an L2 regularization term to the loss function. It penalizes large weights by discouraging the model from fitting the training data too closely.

Gaussian priors implicitly regularize the model, encouraging weight configurations that favour simpler, more generalizable patterns.

Overfitting Prevention

Gaussian priors encourage the model to learn simpler, more general patterns rather than complex patterns that fit the training data's noise. They help in maintaining a balance between fitting the training data well and not becoming overly complex.

Impact on Learning Dynamics

Gaussian priors act as a smoothing operator, leading to smoother decision boundaries or function approximations. This makes the model more robust to small variations in the training data.

Prior Distribution Shape and Input Data Relationship

Experimentation

Experimentation with different priors is encouraged to observe their impact on model behaviour.

Comparing

Comparing the performance of models trained with different priors can provide valuable insights into the influence of priors on the learning process.

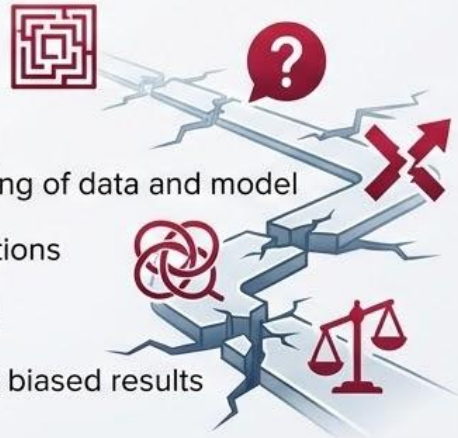
Exploring

Exploring a range of prior distributions helps in understanding which priors are more suitable for specific data and problem domains.

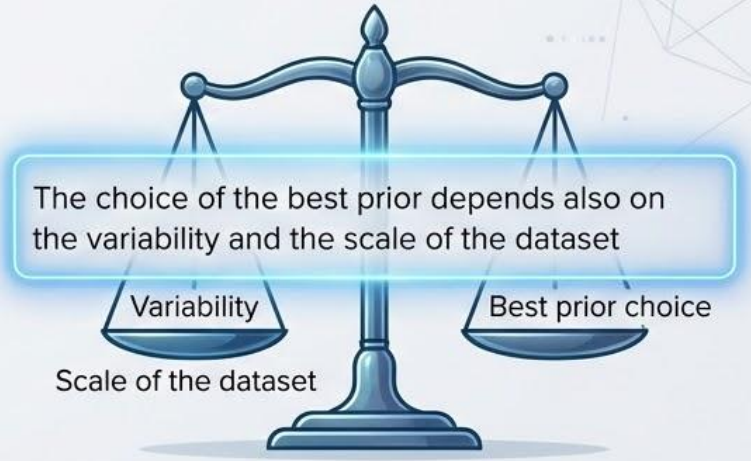
Adjusting

Adjusting priors based on new insights and findings leads to a deeper understanding of Bayesian modelling and its applications.

Challenges in prior selection



- Complex process
- Requires deep understanding of data and model
- Sensitivity to initial assumptions
- Risk of subjective influence
- Inappropriate priors lead to biased results



Why models might fail even when data are abundant

More data and larger models do not automatically improve learning.

Persistent issues:

- poor generalization
- sensitivity to noise
- unstable training

Reason: the learning objective may encode incorrect assumptions.

Weights decide what function is learned
Loss decides which function is preferred among all possible ones

Training = minimizing a loss = assuming a probability model.

- Loss functions encode:
  - the noise model (how data are corrupted)
  - the aleatoric uncertainty structure
  - which errors matter most (symmetry, tails, weighting)
- Same architecture + different loss ⇒ different model

Changing the loss changes what the model assumes about the data.

Aleatoric uncertainty (data uncertainty)

- Inherent randomness or measurement noise in the data
- Cannot be reduced by adding more data
- **Modelled through the loss / likelihood**
- Can be learned if explicitly parameterized (e.g. heteroscedastic models)

Epistemic uncertainty (model uncertainty)

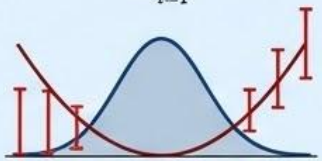
- Uncertainty due to limited data or model capacity
- Can be reduced with more data or better models
- **Encoded in the weights and inference procedure**
- Requires Bayesian or ensemble methods to estimate

Implicit Likelihoods in Standard Machine Learning

In supervised learning, we assume that the observed target y is generated from the model prediction y* plus noise. This data-generation assumption is formalized by introducing a likelihood.

Mean Squared Error → Gaussian Likelihood

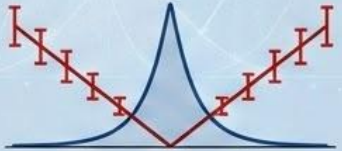
$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$



(Quadratic growth with error
→ Large errors are penalized very strongly)

Mean Absolute Error → Laplace Likelihood

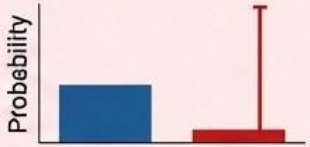
$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$



(Linear growth with error
→ Large errors are penalized less aggressively)

Cross-Entropy → Bernoulli / Categorical Likelihood

$$f(y,\hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$



(Strong penalty for confident wrong predictions
→ Asymmetric behaviour depending on confidence)

Implicit Assumptions & Consequences

Implicit likelihoods typically assume:

- a uniform level of stochastic variability across the dataset,
- symmetric deviations around the model prediction,
- equal reliability for all observations.

When the assumed uncertainty structure does not match the data:

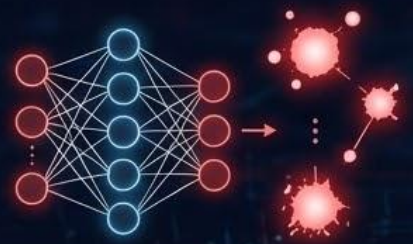
- observations with large variability dominate the gradients,
- extreme deviations distort the learned mapping,
- uncertainty is misinterpreted as structure.

As a result, learning failures are frequently misattributed to the model rather than to the learning objective.

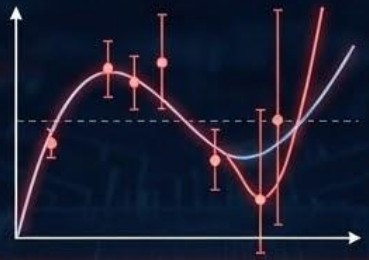
When Likelihood Assumptions Do Not Match Reality

Consequences of mismatched uncertainty assumptions

High-variability observations dominate learning



Extreme deviations distort the learned mapping



Variability is interpreted as structure

From Implicit to Knowledge-Informed Likelihoods

If uncertainty structure is known (or partially known), it should be encoded explicitly in the likelihood.

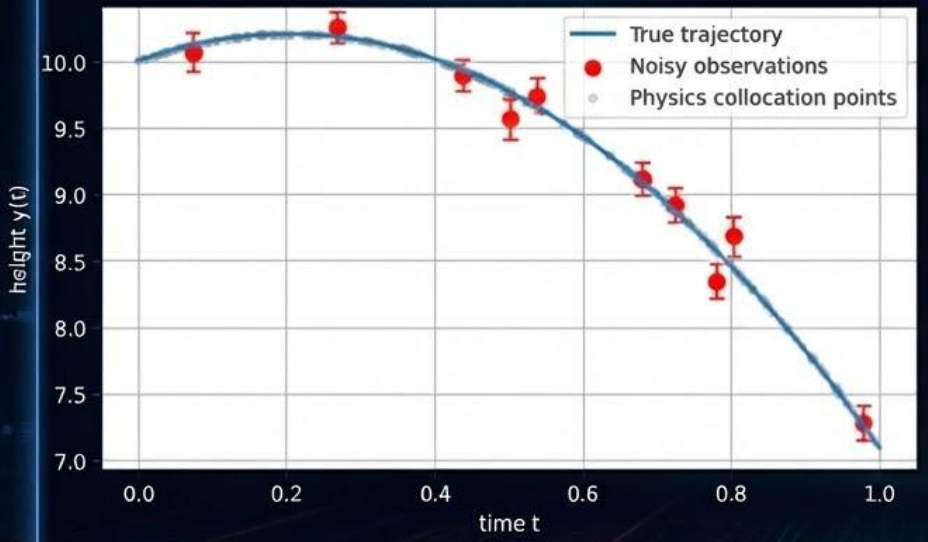
Examples of knowledge

- measurement-dependent variability
- energy-dependent resolution
- heteroscedastic noise
- asymmetric errors
- class-dependent reliability

Knowledge-informed machine learning replaces default likelihoods with structured ones.

Sparse data: the need of more informative loss functions

The Problem: Data-Only Learning with Sparse Observations



Consequences & The Physics-Informed Solution

In many physical problems, only a few noisy observations are available for each phenomenon.

A standard neural network trained with a data-only loss learns to interpolate these points, but has no incentive to behave correctly in regions without data.

As a result, the learned function can match the observations while violating known physical constraints between them.

This motivates the use of more informative loss functions that encode prior physical knowledge and guide learning beyond the observed data.

Concept: Physics-Informed Loss

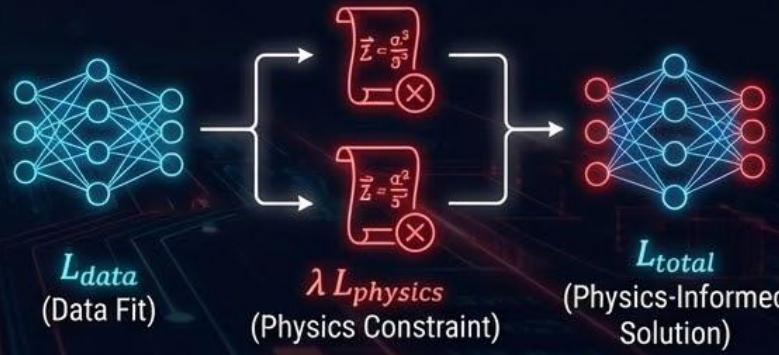
When governing equations are known, they are directly incorporated into the training objective. A physics-based loss term is added to the data loss to penalize violations of physical laws.

This is achieved by evaluating the equation's residual at selected domain points and enforcing it to be small. The total loss thus combines data agreement with physical consistency.

Enforcing these constraints requires computing derivatives of the model's prediction with respect to its inputs.

$$L_{\text{total}} = L_{\text{data}} + \lambda L_{\text{physics}}$$

The weight λ controls the relative importance of physical consistency.

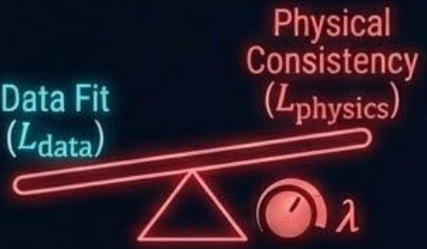


Soft Constraint

$$L_{\text{total}} = L_{\text{data}} + \lambda L_{\text{physics}}$$

$$p(\text{data}, \text{physics} \mid \theta) = p(\text{data} \mid \theta)\, p(\text{physics} \mid \theta)$$

Soft Constraint & The Role of λ



- Constraint can be violated (Discouraged, not forbidden)
- Strength of enforcement is controlled by a weight λ
- The model trades off data fit and physical consistency

Contrast & Benefits

Soft Constraint
(Physics-Informed)
Violations allowed but penalized

Hard Constraint
Exact satisfaction by construction

- Flexibility
- Robustness to noise
- Compatibility with gradient-based optimization

Physics-informed learning does not hard-code physics; it biases learning toward physically consistent solutions.

Simulation-Based Inference: learning the inverse problem from simulations

Nature's Forward Model *(the easy direction for Nature)*

Physical Parameters (θ)

e.g., source properties, particle distribution, magnetic field, environment, …

Emission → Propagation (cosmic & instrumental) → Detection → Reconstruction (instrument response)

Gamma-Ray Spectrum (x)

What we observe

Nature realizes the forward process:
physical parameters generate observable spectra.

The Inverse Problem *(the hard direction for science)*

Unknown Parameters (θ)

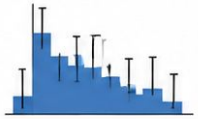
What physical conditions and parameters could have produced the observed spectrum?

Science solves the inverse problem:
from noisy, incomplete observations,
infer the parameters that could have produced them.

Goal: estimate the posterior $p(\theta \mid x_{\text{obs}})$

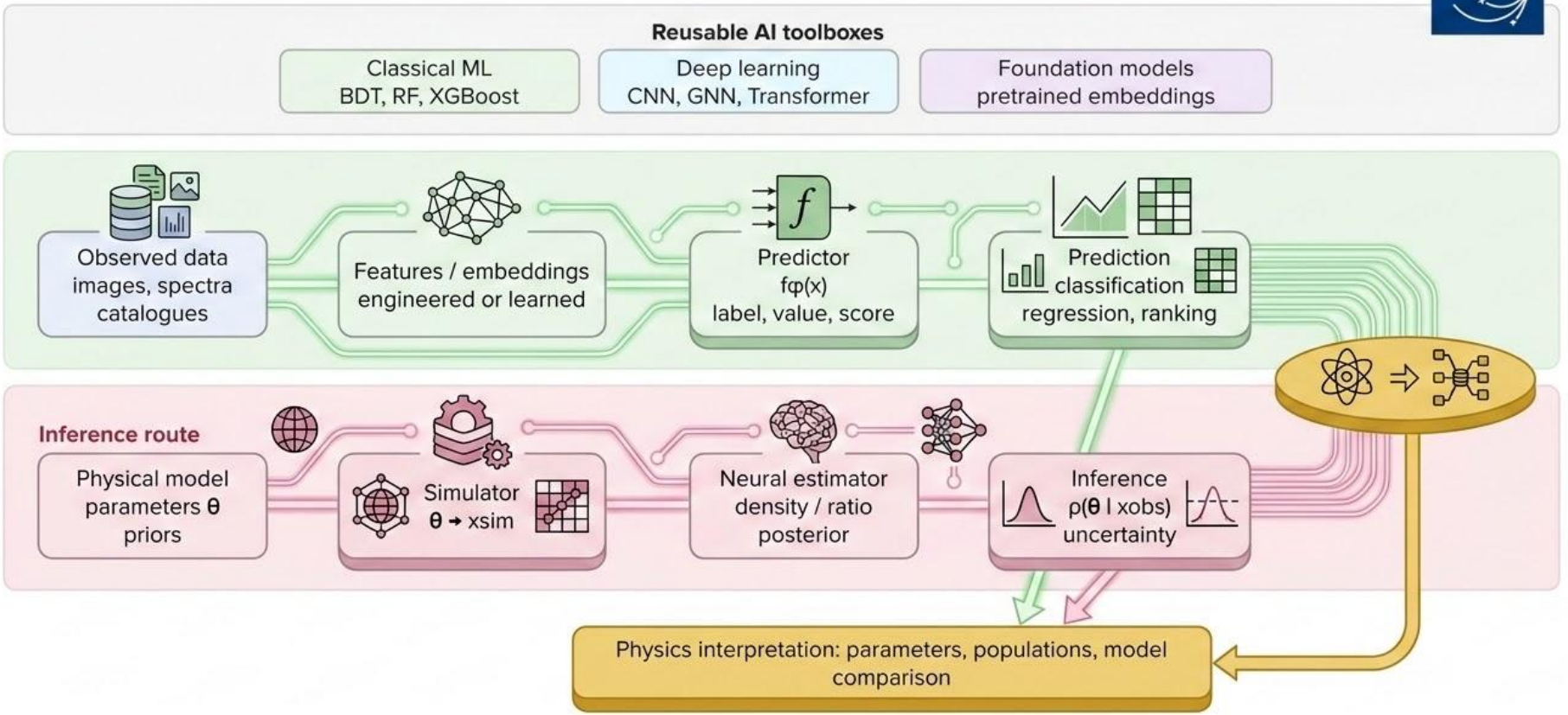
i.e., the set of parameters most compatible with the data, with quantified uncertainties.

Observed Spectrum (x_{obs})



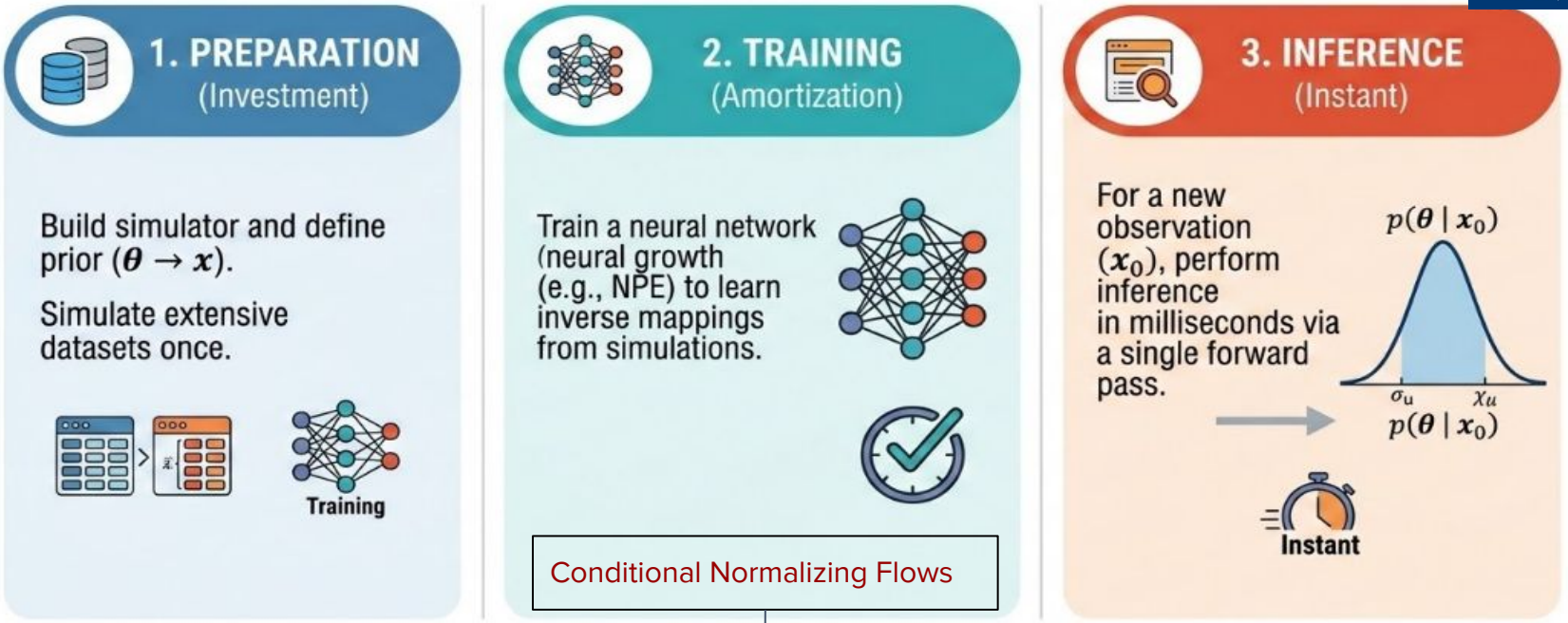
Noisy, binned data

Predictive tools versus statistical inference



The key distinction is the statistical object being learned: a prediction, an embedding, a likelihood ratio, or a posterior.
SBI is a simulation-driven inference workflow.

Learning the inverse probabilistic map with SBI



NPE training learns the inverse probabilistic map: $x \rightarrow \theta$

$$\phi^{\star} = \arg\max_{\phi} \sum_{i} \log q_{\phi}(\theta_i \mid x_i)$$

The normalizing flow is the flexible density model used to represent that inverse map.

How can we build simulations for SBI?

1. Data-driven simulator

Real observations define the distribution of x.

Use resampling, bootstrapping, empirical noise models, measured backgrounds, or synthetic signal injection into real baselines.

Useful when realism is high but the full physical model is incomplete.

2. Direct model simulator

Write an explicit model for the observables:

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

Examples: flare profiles, spectra, SEDs, source populations, simplified detector response.

Useful when the model is interpretable and fast.

3. Monte-Carlo simulator

Run a full numerical forward chain:

$$\theta \rightarrow \text{source} \rightarrow \text{detector} \rightarrow x.$$

Examples: event-level gamma-ray or neutrino simulations, exposure, reconstruction, selection effects.

Useful when instrument and analysis effects are essential.

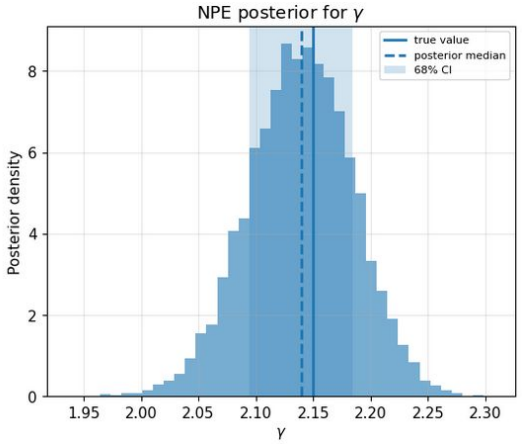
Important: SBI can combine these choices. A hybrid simulator may use physical parameters, empirical backgrounds, and Monte-Carlo detector effects in the same forward model.

Uncertainty in the SBI posterior

The SBI posterior represents uncertainty about the parameters after conditioning on the observed data and on the assumed simulator, prior, and trained neural estimator.

So:

- If the simulator is correct and the neural estimator is well trained, the posterior mainly reflects **aleatoric uncertainty** plus genuine parameter degeneracies.
- If the simulator, prior, training set, or density estimator is imperfect, the posterior also contains **epistemic uncertainty**.



Hands-on Colab notebooks (Focus on Foundation Models and Simulated-Based Inference)

1. Simulation-based inference (SBI) for a toy gamma-ray spectrum (model-driven simulation)
2. SBI-inspired posterior classification of Fermi AGN (data-driven simulation)
3. SBI flare-parameter inference from a Fermi-LAT light curve (semi-empirical parametric Monte Carlo simulator)
4. SDSS photometric redshift with SBI / NPE (data-driven simulation)