Politecnico di Bari

DIPARTIMENTO INTERATENEO DI FISICA

Dottorato in Fisica – XXXIX ciclo - 2024

Machine Learning techniques for particle physics

Federica Maria Simone - federica.simone@poliba.it

Dense Neural Networks: hands-on!

simplilearn

KERAS VS TENSORFLOW VS PYTORCH

Keras is an effective high-level neural network **Application Programming Interface (API)** written in Python.
This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can **run on top of CNTK, TensorFlow, and Theano**.

Keras focuses on being modular, user-friendly, and extensible. It doesn’t handle low-level computations; instead, it hands them off to another library called the Backend.

TensorFlow is an end-to-end open-source deep learning framework developed by Google and released in 2015. It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android.

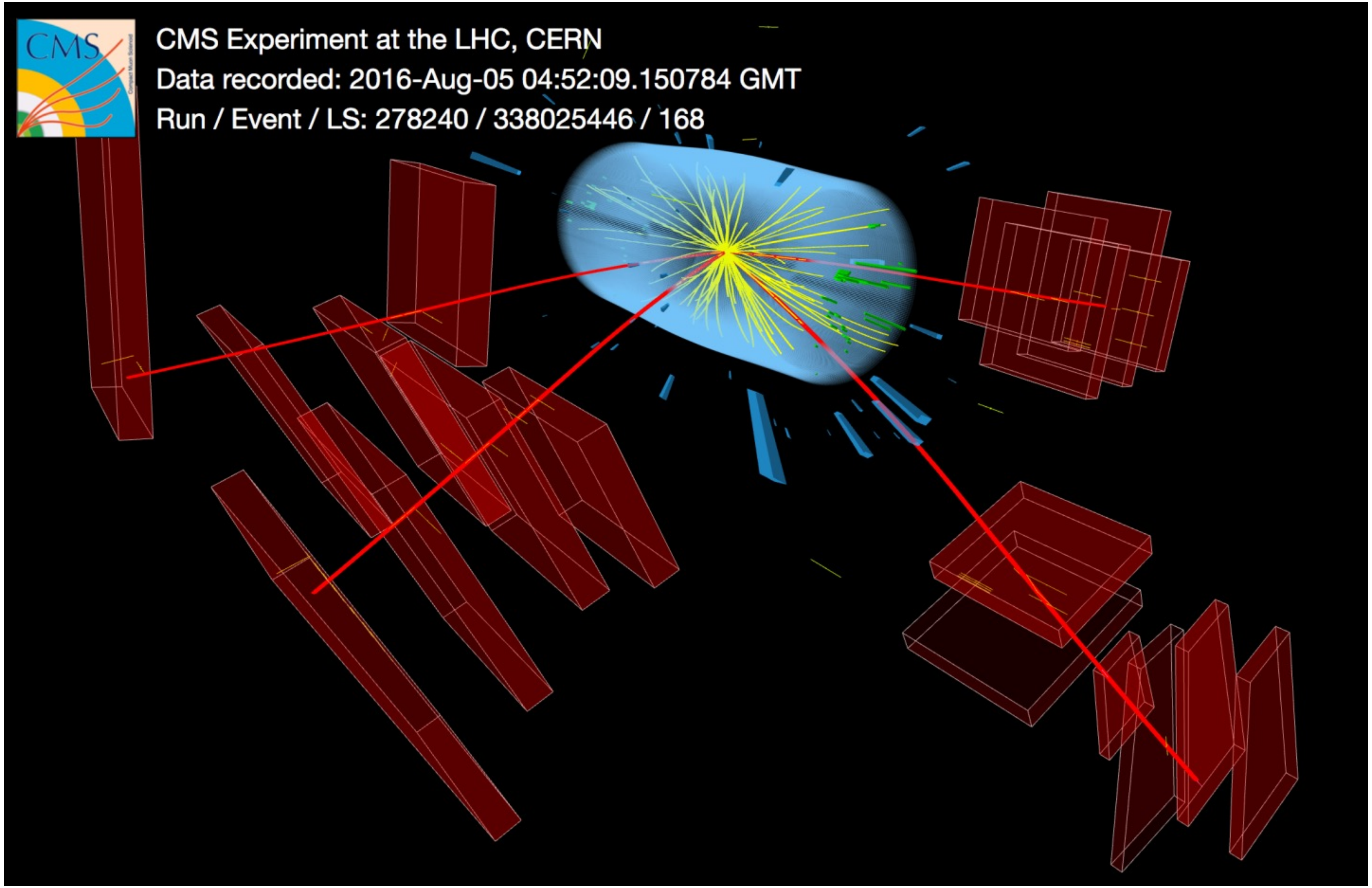
TensorFlow is a **symbolic math library** used for neural networks and is best suited for dataflow programming across a range of tasks. It offers multiple abstraction levels for building and training models.

Also, TensorFlow has adopted Keras API.

PyTorch is a relatively new deep learning framework based on Torch. Originally developed by Meta AI and now part of the Linux Foundation umbrella, it is an optimized tensor library for deep learning using GPUs and CPUs.

PyTorch has a reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs. It also feels native, making coding more manageable and increasing processing speed.

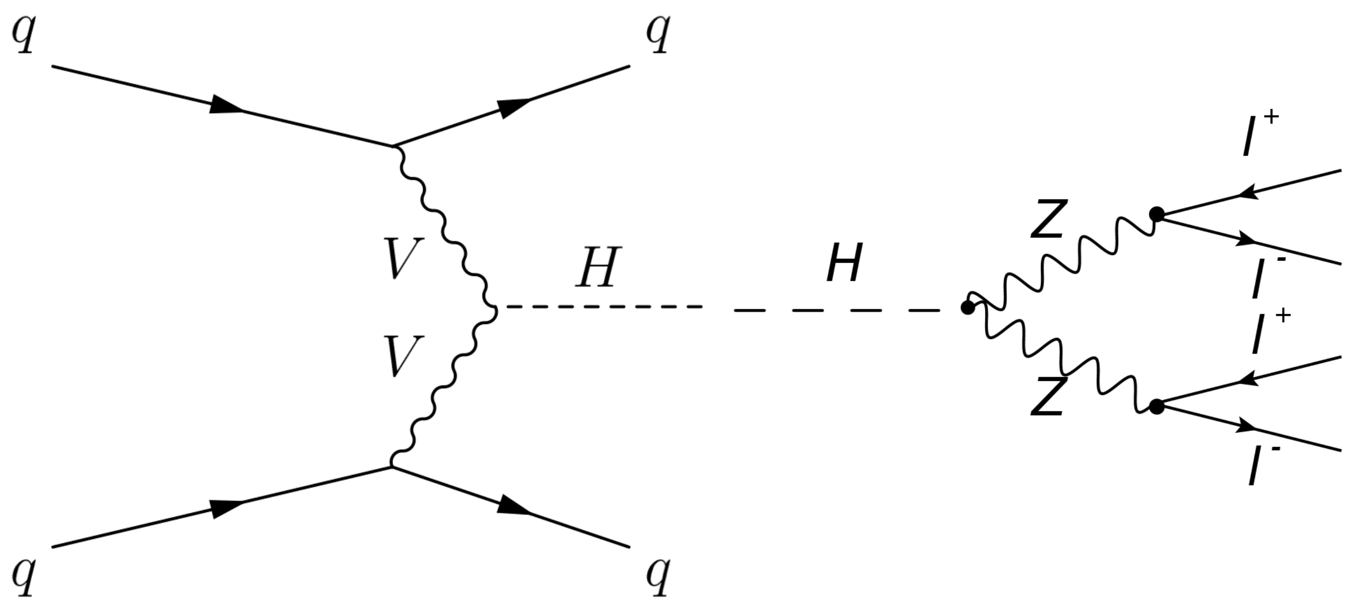
First exercise: H→ZZ→4l analysis

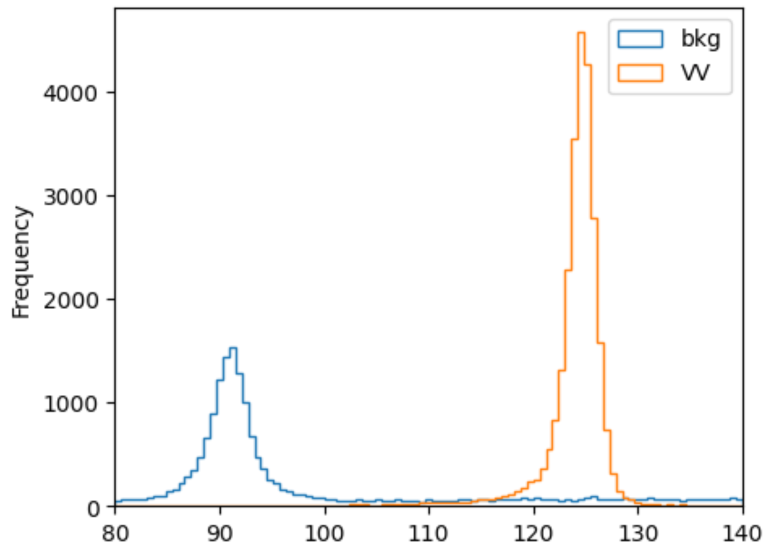


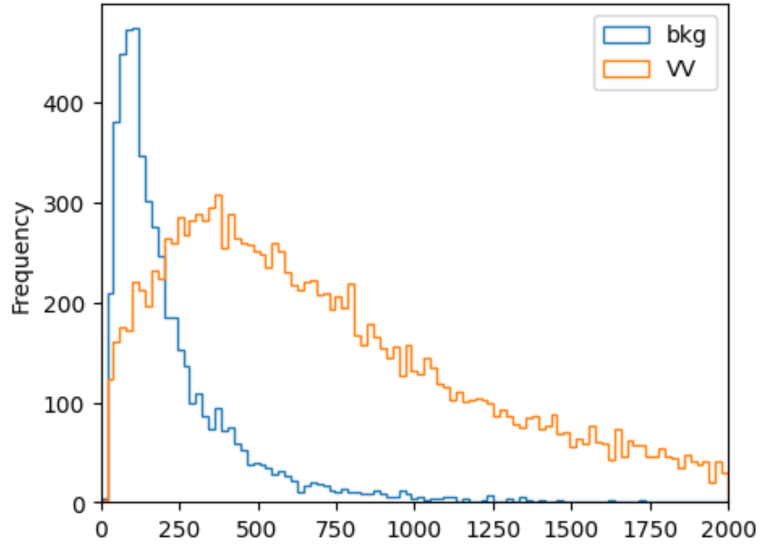
First exercise: H→ZZ→4l analysis

Higgs production mechanism: VBF
Higgs decay: H→ZZ→4l

Goal: discriminate VBF Higgs and standard model background 4 muon events.





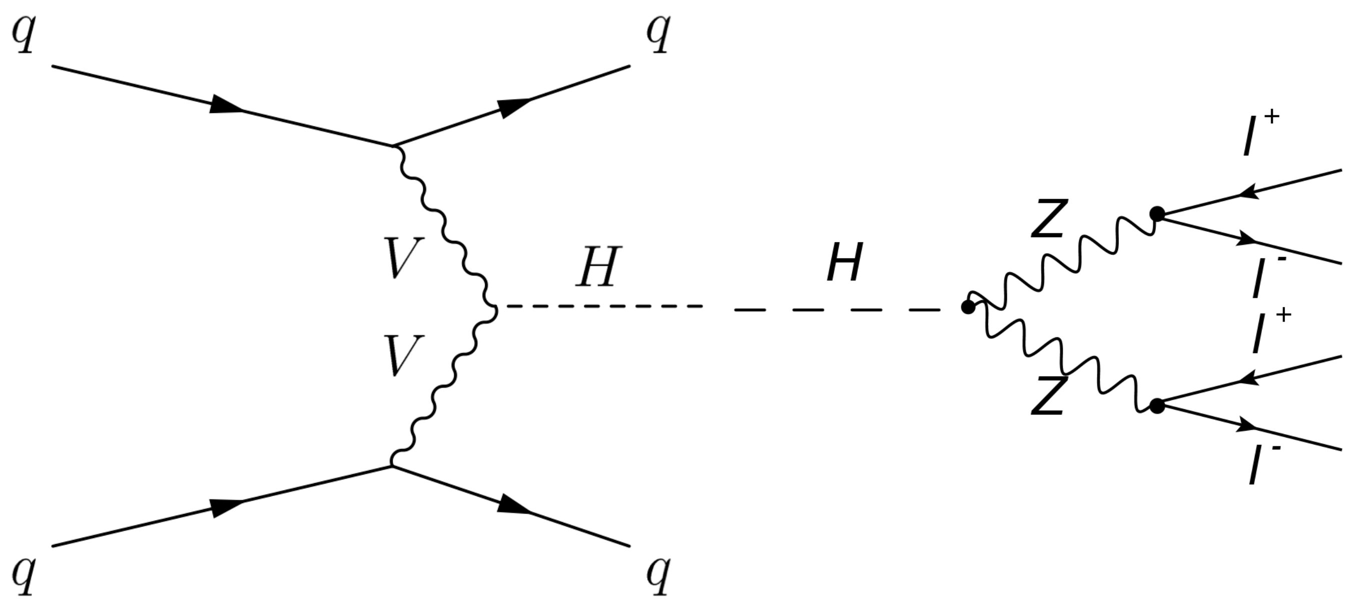


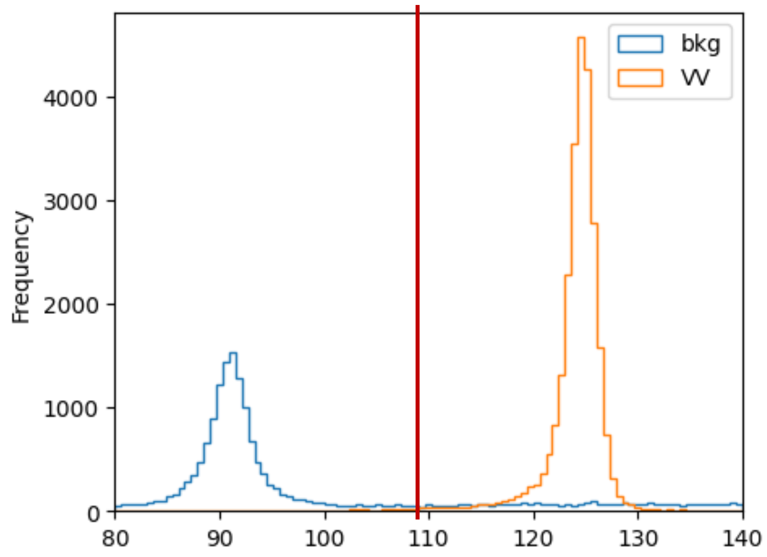
First exercise: H→ZZ→4l analysis

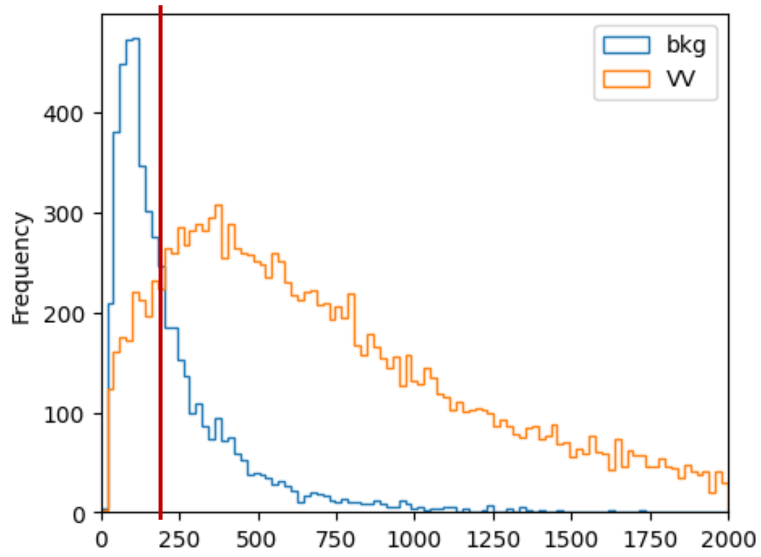
Higgs production mechanism: VBF
Higgs decay: H→ZZ→4l

Goal: discriminate VBF Higgs and standard model background 4 muon events.

“Cut-based” analysis: classify signal and background events based on thresholds on some observables



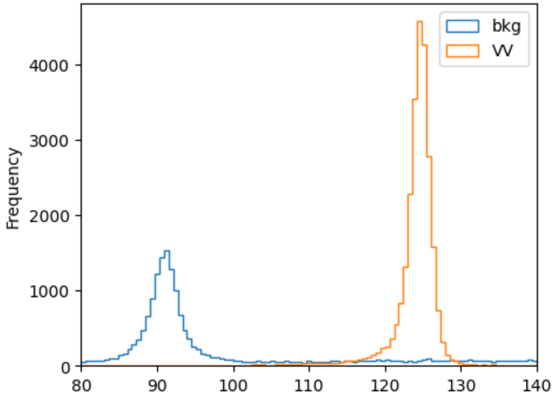


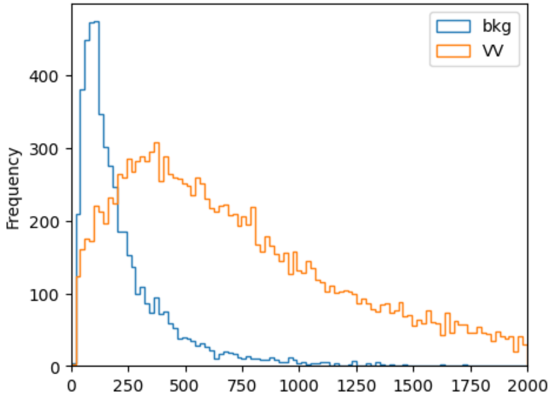


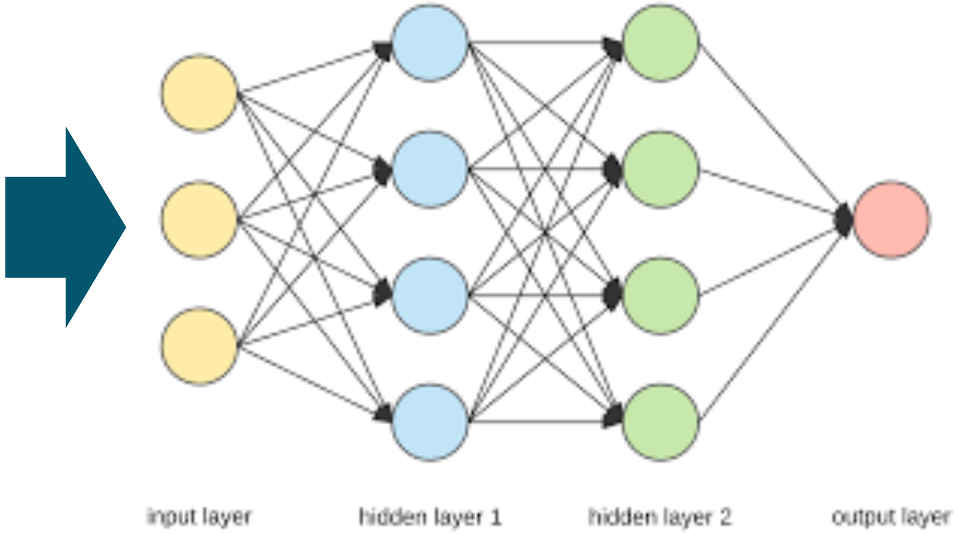
First exercise: H→ZZ→4l analysis

Let's solve this with ML!

Notebook 1:
- Download the rootfiles
- Convert Ttree into numpy dataframe
- Plot input features using matplotlib





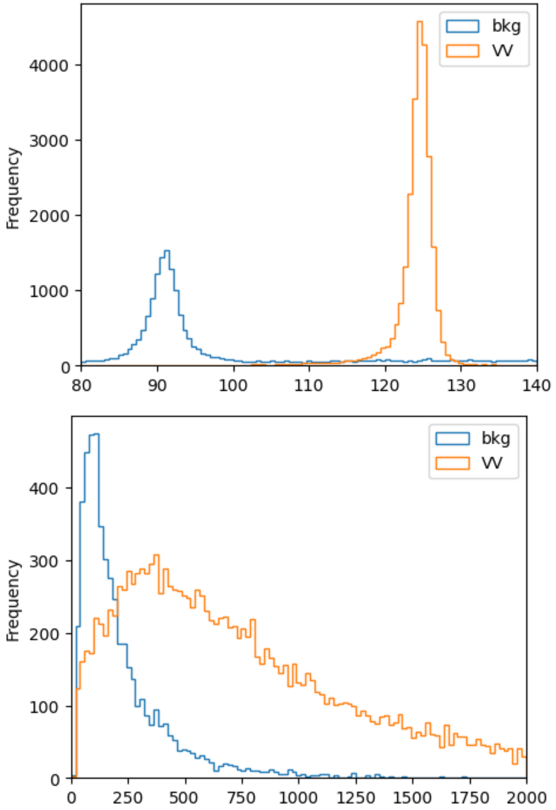


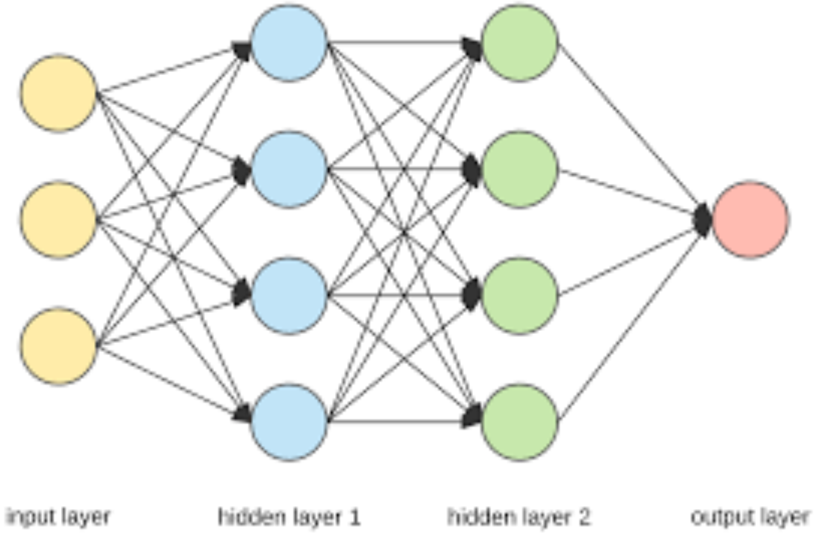
https://github.com/fsimone91/course_ml4hep/tree/2024/notebooks/2024/1-datasets-uproot.ipynb

First exercise: H→ZZ→4l analysis

Let's solve this with ML!

Notebook 2:
- Define a dense NN in keras+tensorflow
- Train it
- Look at performance using test set
- Play with the hyperparameters





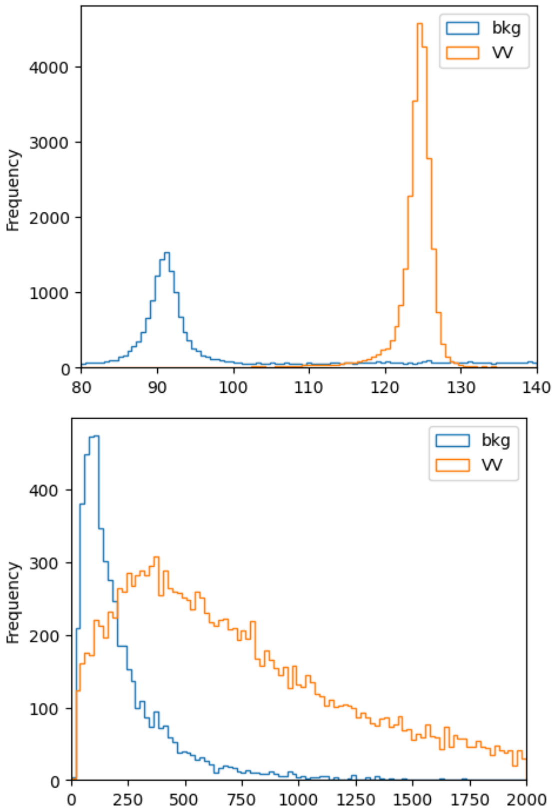
https://github.com/fsimone91/course_ml4hep/tree/2024/notebooks/2024/2.1-dense-keras.ipynb

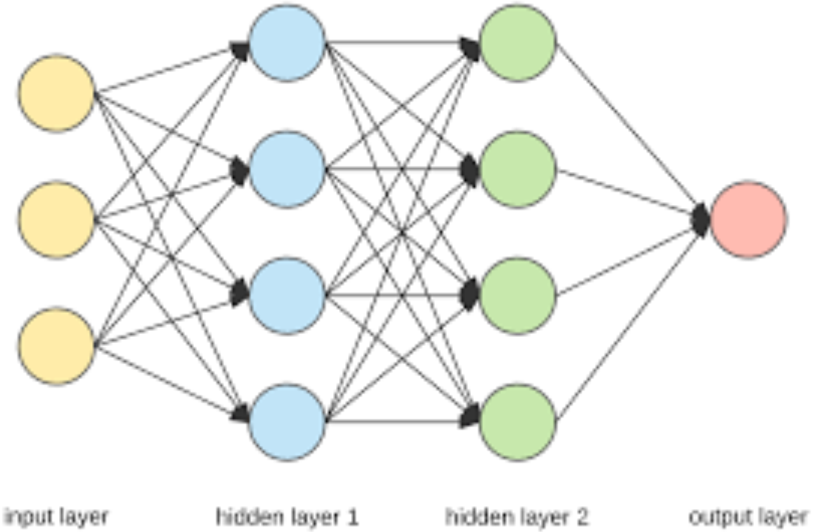
First exercise: H→ZZ→4l analysis

Let's solve this with ML!

Notebook 3:

- Define a dense NN in pytorch
- Train it
- Look at performance using test set
- Play with the hyperparameters





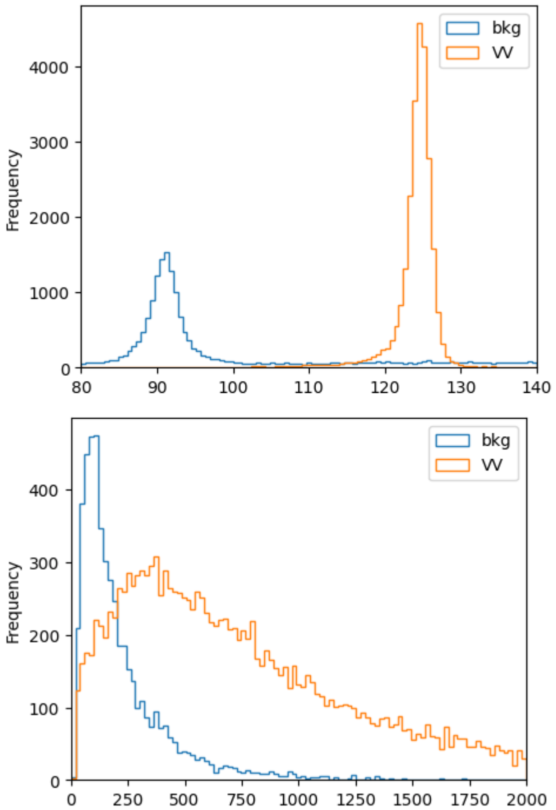
https://github.com/fsimone91/course_ml4hep/tree/2024/notebooks/2024/2.2-dense-pytorch.ipynb

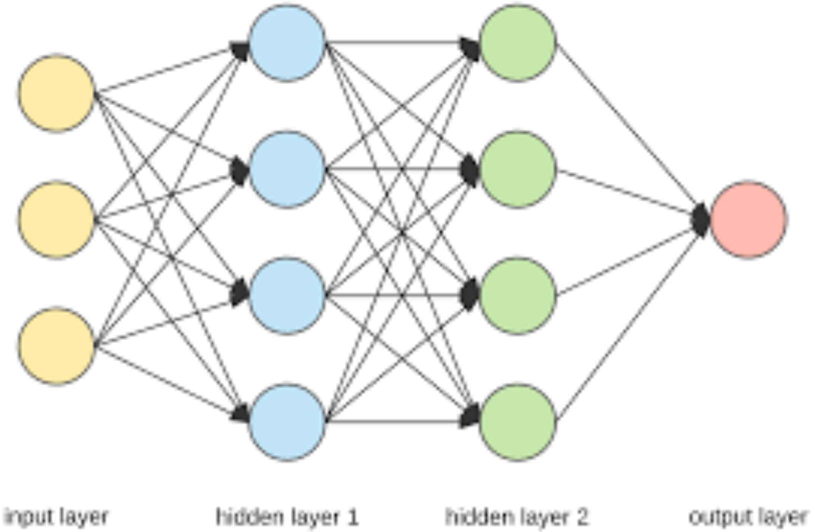
First exercise: H→ZZ→4l analysis

Let's solve this with ML!

Extra: notebook 4

- Define a dense NN in keras+tensorflow
- Optimise some hyperparameters using [Scikit-Optimize](https://scikit-optimize.github.io)





https://github.com/fsimone91/course_ml4hep/tree/2024/notebooks/2024/2.3-dense-bayesian-optimization.ipynb

Bayesian optimization with `skopt`

Gilles Louppe, Manoj Kumar July 2016. Reformatted by Holger Nahrstaedt 2020

Problem statement

We are interested in solving

$$x^* = arg$$
$$min_x f(x)$$

under the constraints that

- f is a black box for which no closed form is known (nor its gradients);
- f is expensive to evaluate;
- and evaluations of $y = f(x)$ may be noisy.

Disclaimer. If you do not have these constraints, then there is certainly a better optimization algorithm than Bayesian optimization.

Bayesian optimization loop

For $t = 1 : T$:

1. Given observations $(x_i, y_i = f(x_i))$ for $i = 1 : t$, build a probabilistic model for the objective f. Integrate out all possible true functions, using Gaussian process regression.
2. optimize a cheap acquisition/utility function u based on the posterior distribution for sampling the next point. $x_{t+1} = argmin_x u(x)$ Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Acquisition functions

Acquisition functions $u(x)$ specify which sample x: should be tried next:

- Expected improvement (default): $-EI(x) = -mathbbE[f(x) - f(x_t^+)]$
- Lower confidence bound: $LCB(x) = mu_{GP}(x) + kappasigma_{GP}(x)$
- Probability of improvement: $-PI(x) = -P(f(x)geqf(x_t^+) + kappa)$

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., $kappa$) for controlling the exploration-exploitation trade-off. - Search in regions where $mu_{GP}(x)$ is high (exploitation) - Probe regions where uncertainty $sigma_{GP}(x)$ is high (exploration)