

Highlights of the Third ML_INFNO Hackathon Advanced Level

Luca Giommi

INFNO CNAF



- Progetto ML_INFNO di CSN5 nasce il 1 Gennaio 2020 con durata Triennale (+1)
- Tommaso Boccali lascia il suo posto di responsabile di ML_INFNO a Lucio Anderlini



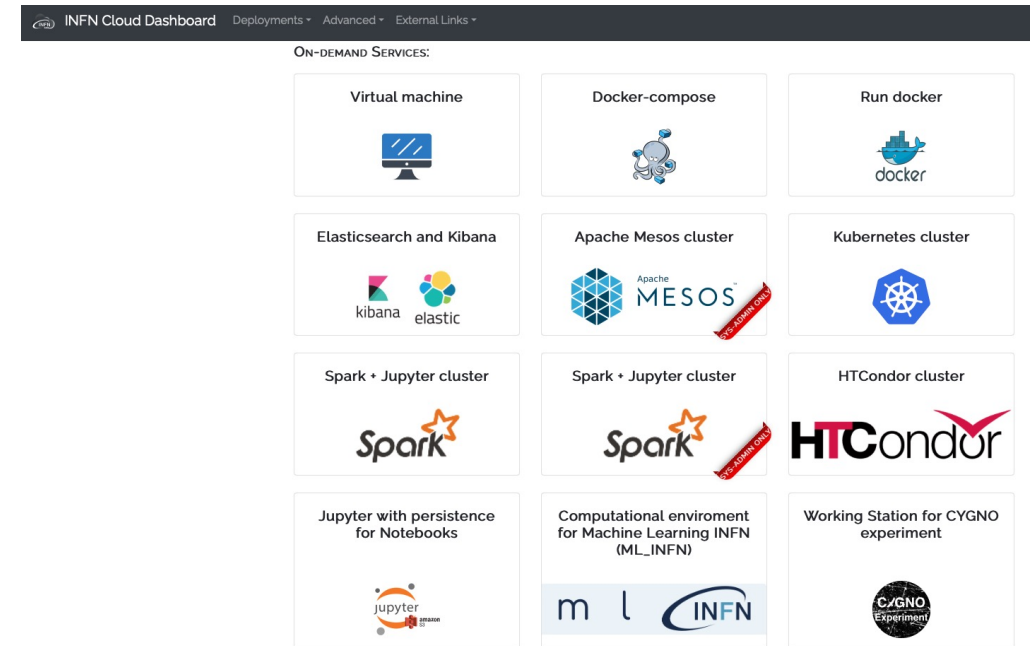
Obiettivo: Il progetto ML_INFNO nasce nel 2020 con l'obiettivo di migliorare l'utilizzo di tecnologie di frontiera AI-driven (machine learning, deep learning) da parte dei ricercatori e degli associati INFNO. Il progetto mette a disposizione sia una piattaforma hardware, comune ed espandibile, che la possibilità di condividere le conoscenze di base già acquisite dagli stessi ricercatori e tecnologi.




Struttura:

- WP1 – Infrastruttura & Provisioning (Stefano Dal Pra)
- WP2 – Formazione (Francesca Lizzi)
- WP3 – Casi scientifici (Luca Giommi)

➤ **INFN Cloud** costituisce la base infrastrutturale di ML_INF

- ML_INF ha messo in opera un cluster di risorse ad alte performance (tra cui GPU) progettato per adattarsi ai progetti dell'INFN che utilizzano tecniche di Machine Learning, e lo ha reso accessibile tramite INFN Cloud.
- ML_INF ha fin dall'inizio fornito a INFN Cloud casi di studio e personale esperto per focalizzarsi su tematiche di frontiera, quali la realizzazione di servizi specifici per calcolo ad alte prestazioni, il provisioning e l'accounting di risorse eterogenee, e la gestione di pipeline complesse per applicazioni di Machine Learning potenzialmente data intensive



 <p>7-9 Jun 2021 Zoom Europe/Rome timezone</p>	<p>First ML-INFN Hackathon</p> <p>Base level Online Only 54 “students” –3x registrations</p>
<p>Base level Online Only 60 “students”</p>	 <p>13-15 Dec 2021 Zoom Europe/Rome timezone</p> <p>Second ML-INFN Hackathon: Starting Level</p>
 <p>21-24 Nov 2022 Europe/Rome timezone</p>	<p>Third ML-INFN Hackathon: Advanced Level</p> <p>Advanced level In person 23 participants</p>

Raccolta e ampliamento della collezione di casi d'uso realistici di tecniche di Machine Learning nelle varie linee di ricerca dell'Ente

- Come entry point del progetto verso l'esterno e come catalogo degli use case viene usato Confluence
<https://confluence.infn.it/display/MLINFN/ML-INFN+Knowledge+Base>
<https://confluence.infn.it/display/MLINFN/Machine+Learning+Knowledge+Base>

Table of Use cases

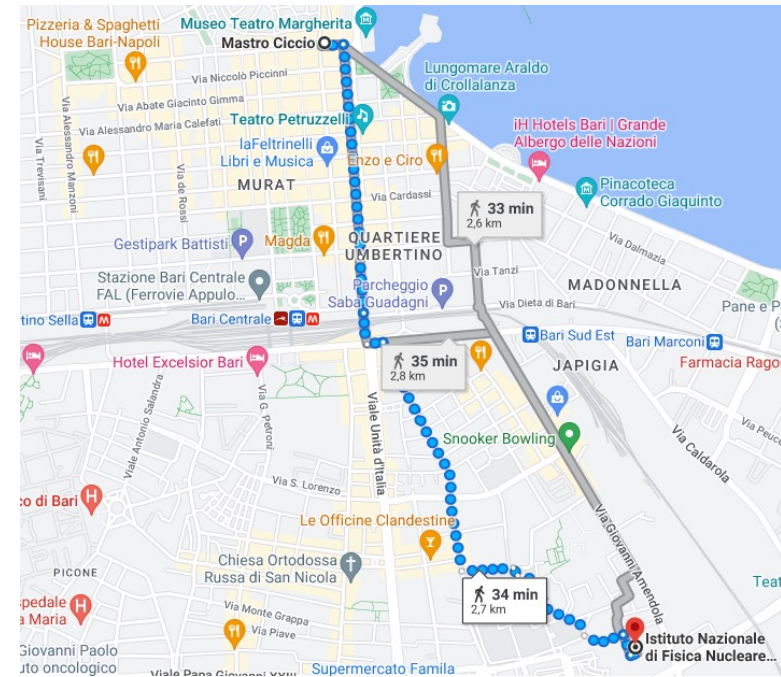
Name and Link	ML Technologies	Scientific Field	ML Tools	Comments
Btagging in CMS (templated version)	CNN, LSTM	High Energy Physics	Keras + Tensorflow	Realistic application
LHCb Masterclass, with Keras	DE, MLP	High Energy Physics	ROOT + Keras + TF	Introductory tutorial
MNIST in a C header	MLP		Keras	Free-styling tutorial
LUMIN: Lumin Unifies Many Improvements for Networks	CNN, RNN, GNN	High Energy Physics	PyTorch	Package use examples
INFERNO: Inference-Aware Neural Optimisation	NN	High Energy Physics	Keras + Tensorflow	Technique application example
An introduction to classification with CMS data	Fisher, BDT, MLP	High Energy Physics	Scikit-learn, TF2	Tutorials for Master Students
Virgo Autoencoder tutorial	Autoencoder	General Relativity	Python Keras	Tutorial for student
Distributed training of neural networks with Apache Spark	DNN	High Energy Physics	Spark + BigDL	Tutorial
FTS log analysis with NLP	NLP	High Energy Physics, Computing	Word2Vec + Rake + sklearn	
Image inpainting tutorial: how to digitally restore damaged images	CNN U-Net	Applied Physics	Keras + Sci-kit image, PIL, OpenCV, matplotlib	Tutorial
Signal/background discrimination for the VBF Higgs four lepton decay channel with the CMS experiment using Machine Learning classification techniques	ANNs, RF	High Energy Physics	Keras, TensorFlow, Scikit-learn	Tutorial
Explainability of a CNN classifier for breast density assessment	CNN	Medical Physics	Keras, Tensorflow	Tutorial
ML for smart caching	ML/RL	High Energy Physics, Computing, Cache	Keras, Tensorflow, sklearn	Demo, playground

L'hackathon

Info sull'hackathon

- **Quando:** dal 21 al 24 Novembre
- **Dove:** INFN Sezione di Bari
- **Partecipanti:** 23
- **Organization committee:** 22 (8 anche nel Local Organizing Committee)

<https://agenda.infn.it/event/32568/>



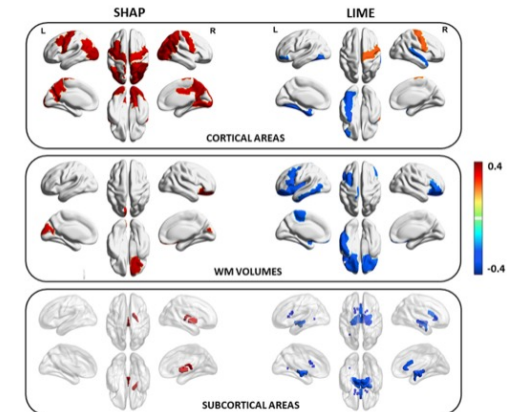
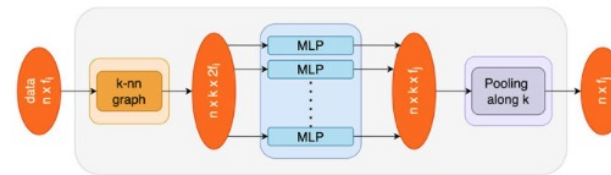
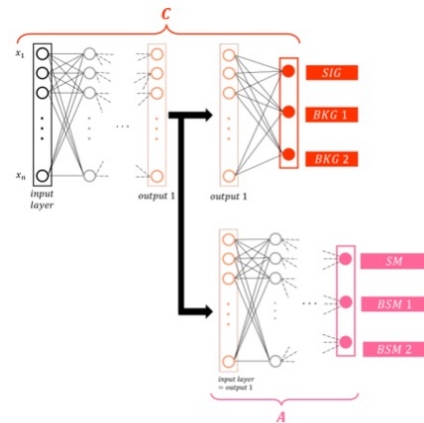
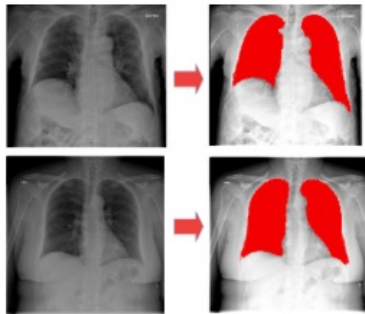
Programma dell'hackathon

- Lunedì pomeriggio: lezioni/seminari
- Martedì mattina: lezioni/seminari
- Martedì pomeriggio: esercizi hackathon
- Martedì sera: cena sociale
- Mercoledì mattina: lezioni/seminari
- Mercoledì pomeriggio: esercizi hackathon
- Giovedì mattina: lezioni/seminari



➤ Gli esercizi dell'hackathon

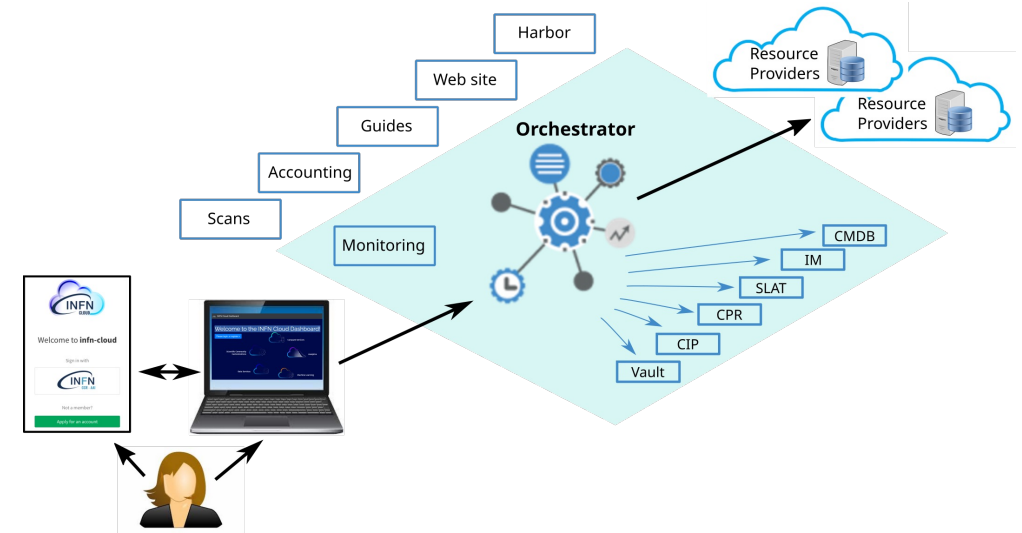
- **Domain Adaptation to train model-independent classifiers in High Energy Physics**
- Lung Sementation on Chest X-Ray images with U-Net
- **Graph Neural Networks and Transformers**
- Explainable Artificial Intelligence (XAI)



Giorno 1

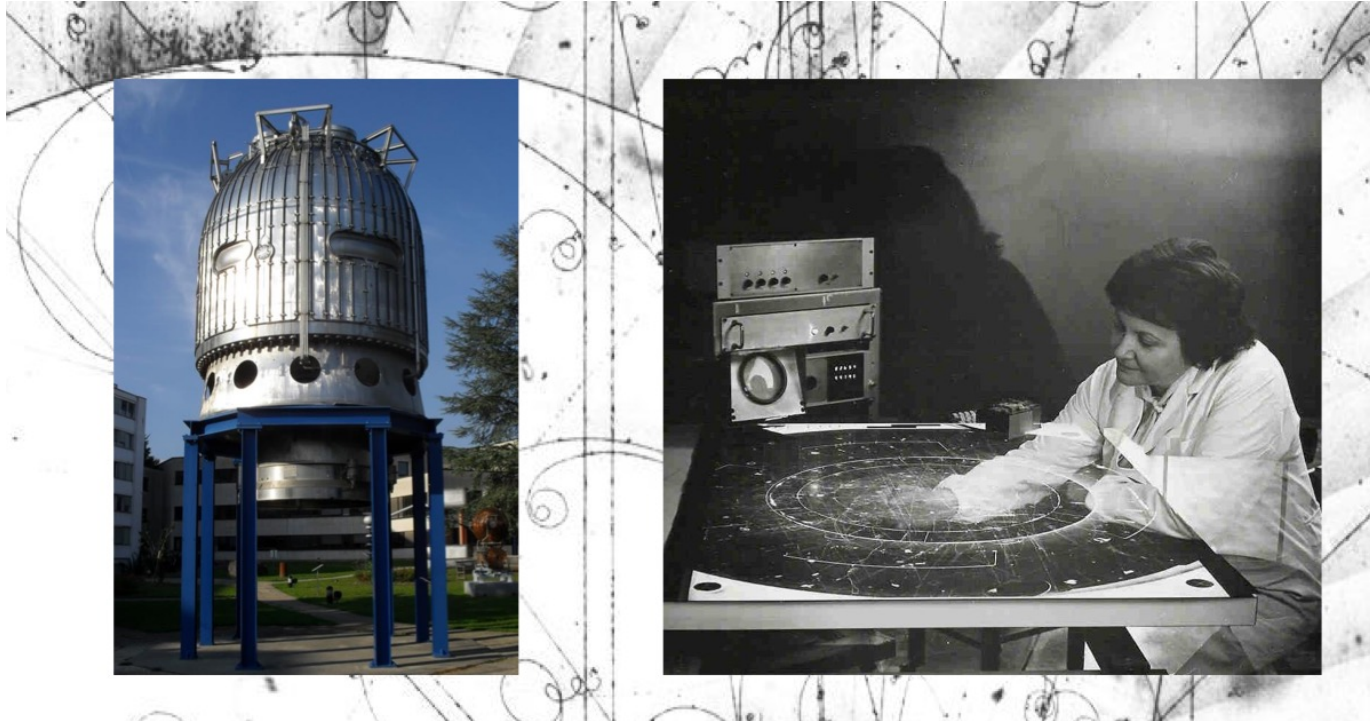
Descrizione dei task e obiettivi dei diversi WP

- **WP1:** gestione operativa dell'infrastruttura del backbone di INFN Cloud, gestione dei servizi core di INFN Cloud presenti sul backbone, gestione di servizi di tipo SaaS, gestione dell'infrastruttura di storage S3 di INFN Cloud, supporto agli altri WP, supporto di secondo livello agli utenti
- **WP2:** Documentazione, supporto utenti e di casi d'uso, comunicazione e training
- **WP3:** Monitoring (infrastruttura e servizi ancillari, servizi cloud, risorse per cloud federata, servizi utente), dashboard di progetto (visualizzazione metriche), notifiche di malfunzionamento, accounting (collezione dati su uso delle risorse per utenti e per gruppi), ...
- **WP4:** Coordinare la sicurezza di INFN-Cloud, stabilire le politiche di utilizzo e comportamento accettabile su INFN-Cloud, gestire gli incidenti e monitorare la sicurezza, design (in collaborazione con gli altri WP) di soluzioni che rispettino i requisiti di sicurezza
- **WP5:** Sviluppo, porting e co-design e R&D di soluzioni tecniche user-level per utenti e comunità scientifiche, finalizzate a facilitare l'accesso alle risorse di computing e di storage, incluso eventualmente accesso ad hardware specializzato (i.e. GPU ...)



Virtual machine 	Docker-compose 	Run docker
Elasticsearch and Kibana 	Apache Mesos cluster 	Kubernetes cluster
Spark • Jupyter cluster 	HTCondor cluster 	Jupyter with persistence for Notebooks
Computational environment for Machine Learning INFN (ML_INFN) 	Working Station for CYGNO experiment 	Sync&ShareaaS

- Che cos'è il **CNAF**?
- Centro Nazionale di HPC, Big Data e Quantum Computing. Il CNAF inizierà lo spostamento all'ICSC in estate 2023. ICSC ospiterà l'80% della capacità di calcolo d'Italia
- Diversi lavori fatti e in progress per integrare HPC@CNAF con HPC@CINECA (Marconi A2 e Marconi 100)

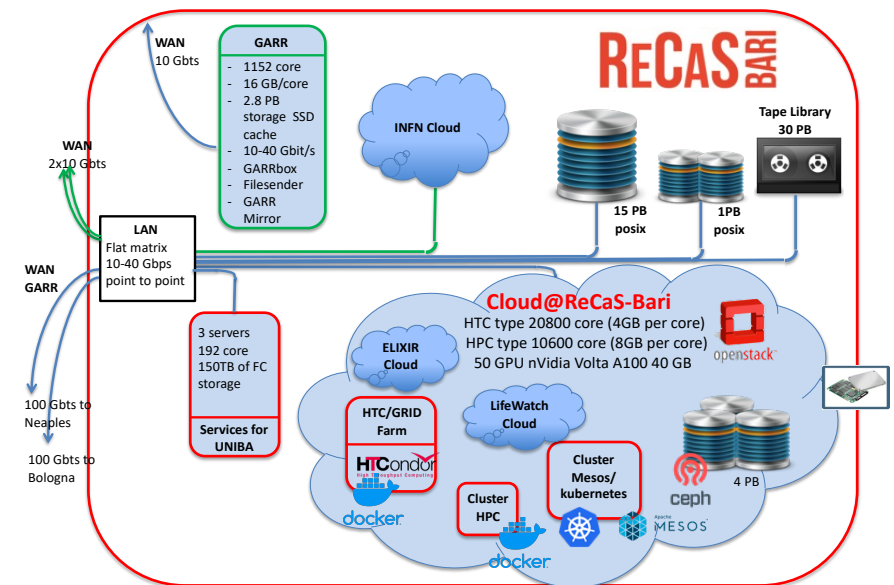


- Small HPC cluster, local submission
- 44 nodes, recently moved LSF → Slurm
- GPFS shared filesystem
- A few nodes having V100 GPUs
- ML_INFNO: Accelerated HW via CLOUD:
- GPU: (Tesla 10 x T4, 5 x RTX5000, A30, 2 x A100)
- FPGA: 2 x Xilinx Alveo U50, 1 x Xilinx Alveo U250
- Coming soon:
- FPGA: 5 more Alveo U50,
- GPU: 2 more A100

Il DataCenter ReCaS è ospitato in un edificio di due piani, appositamente realizzato, con una superficie di 430 metri quadri per piano.

Chi usa ReCaS? ReCaS supporta diverse **attività scientifiche** e non

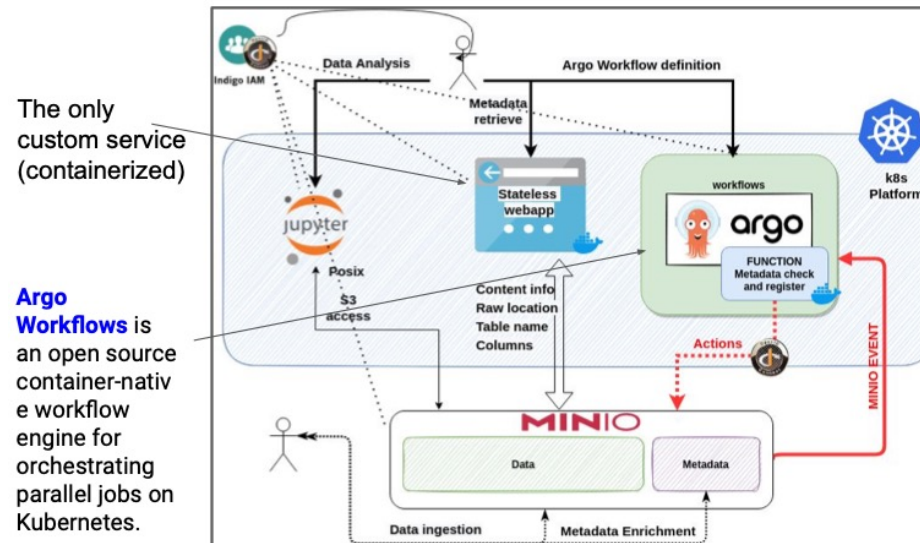
- ARPA–PUGLIA
 - Previsioni meteo e analisi di qualità dell'aria della regione Puglia (giornaliere, con risoluzione di 4 km) con un focus di 1 km intorno a Taranto (per inquinamento ILVA)
- Comune di Bari
 - Risorse Cloud@ReCaS–Bari per il progetto MUSICA
- Test con ASI e CNR su risorse Cloud@ReCaS–Bari
 - Analisi dati Sentinel e Sentinel 2
- EGI Federated Cloud
- JRU ELIXIR–IIB (sia UNIBA che INFN sono membri della JRU)
- JRU Lifewatch (sia UNIBA che INFN sono membri della JRU)
- CNR
 - Storage e analisi di dati per varie attività (p.es. ECOPOTENTIAL)
 - Analisi dati bioinformatica
- GAP
 - Telerilevamento e analisi di immagini da satellite
- Planetek:
 - Uso di risorse HPC/HTC (con GPU) per analisi dati
 - Utilizzo di risorse cloud per esportare servizi critici.
- PON: RPASinAir // CLOSE // TEBAKA



- Cosa vuol dire **Open Science Cloud** (Open Science + Cloud) e nel concreto “Allow researcher to exploit free and open services to manage workflow, build pipeline, data processing and analysis and, of course, to share/to reuse technical solutions. It also allow researchers to focus on science”.
- Modello “**Infrastructure as a code**” basato su un approccio dichiarativo che promuove soluzioni basate su container. L’utente può decidere di non preoccuparsi dell’infrastruttura che c’è sotto
- INFN Cloud utilizza questo approccio fornendo una piattaforma e servizi basati su tecnologie industry standard e open source.

- Distribuire il carico di lavoro per parallelizzare il processamento dei dati può essere complicato. Oggi diverse comunità stanno usando framework di alto livello in grado di sfruttare risorse di calcolo distribuite (i.e. RDataFrame nel mondo HEP).
- Soluzione R&D all’INFN per HEP (CMS) utilizza JupyterHub+ JupyterLab (per gestire l’interfaccia utente), DASK (che permette di scalare su un sistema altamente distribuito), XRootD (per accedere dati remoti su GRID). Risultati incoraggianti su use-case di fisica. Usando ~ 2 TB di dati si è ottenuto un miglioramento di un fattore 8 e 3 sulla preselezione e postselezione

A cloud-native platform would look like



Almost everything is implemented with **Industry Standard solutions**

Container (docker) everywhere

Container Orchestration

- Self healing
- Automate
- Easy service deployment

Architecture of **The PLANET** experiment@CSN5

Materiale usato lo potete trovare [qui](#)

- Eseguire notebooks in batch con **tmate** (tmate is a small program that connects to a remote server and provides a secure ssh access to a virtual terminal instance through that remote server) e **nbconvert**.
 - Lanciando tmate su terminale di jupyter viene restituito un comando che copiamo e lanciamo sul nostro device. In questo modo abbiamo accesso come root all'istanza di ML-INFEN da un terminale del nostro device
 - Lanciando “**jupyter nbconvert <notebook_file> --execute --inplace**” eseguiamo un notebook come se fosse un programma
- Lanciare workflow di notebook con [Snakemake](#)
- Nella lezione è stato fatto un esempio HEP in CMS dividendo i task (caricamento e selezione dati, definizione e training modello ML, plot dei risultati) in tre diversi notebook (sotto un esempio di Snakefile)

```

1 BOUNDARIES = (0, 10, 15, 20, 25, 30, 50)
2
3 rule get_data:
4     input:
5         notebook="get_data.ipynb"
6
7     output:
8         notebook="scratch/get_data.ipynb",
9         data=["scratch/psi_%s-%s.pkl" % b for b in zip(BOUNDARIES[:-1], BOUNDARIES[1:])]
10
11     shell:
12         "OUTPUT='{output.data}' "
13         "jupyter nbconvert --to notebook --output {output.notebook} --execute "
14         "{input.notebook}"
15
16
17 rule fit:
18     input:
19         notebook="fit.ipynb",
20         data="scratch/psi_{p_low}-{p_high}.pkl"
21
22     output:
23         notebook="scratch/fit_{p_low}_{p_high}.ipynb",
24         data="scratch/fit_{p_low}-{p_high}.npz"

```

```

26     shell:
27         "INPUT='{input.data}' "
28         "OUTPUT='{output.data}' "
29         "jupyter nbconvert --to notebook --output {output.notebook} --execute "
30         "{input.notebook}"
31
32
33 rule collect:
34     input:
35         notebook="collect.ipynb",
36         data=["scratch/fit_%s-%s.npz" % b for b in zip(BOUNDARIES[:-1], BOUNDARIES[1:])]
37
38     output:
39         notebook="scratch/collect.ipynb"
40
41     shell:
42         "INPUT='{input.data}' "
43         "jupyter nbconvert --to notebook --output {output.notebook} --execute "
44         "{input.notebook}"
45
46 ..

```

Giorno 2

Systems of Neural Networks in HEP – Piergiulio Lenzi

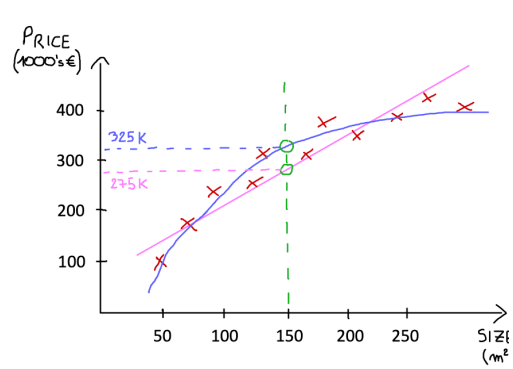


Tutto il talk incentrato su ICSC

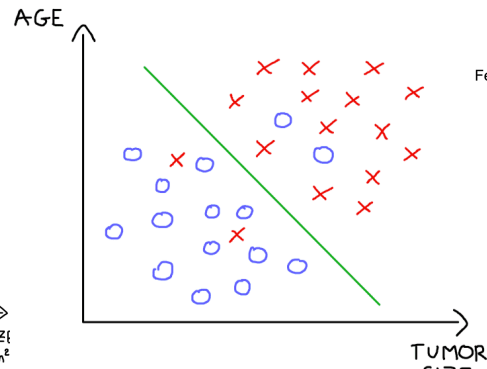
- Con il PNRR sono stati finanziati 5 centri nazionali di cui uno (CN1 o ICSC) al Tecnopolo di Bologna
 - Coinvolte 34 Università, 15 grandi aziende
 - 10 Spoke. In Spoke 2 (fundamental research and space economy) l'INFN e INAF sono leader (c'è Tommaso Boccali per l'INFN)
- In Spoke 2 le attività sono divise in 6 work packages (WP)
 - Kick-off dello Spoke 2 c'è stato il [13 Ottobre](#) mentre quello di CN1 il [25-26 Novembre](#)
- WP 2 (Design and development of science-driven tools and innovative algorithms for Experimental High Energy Physics) è incentrato su algoritmi, ML e porting su infrastrutture eterogenee
 - Esempi, ML-based fast simulation, event reconstruction, event classification, iniziative cross-domain (Explainable AI, DQM)
- WP4 propone nuove tecnologie (GPU, FPGA) e WP5 propone soluzioni che scalano per le diverse comunità



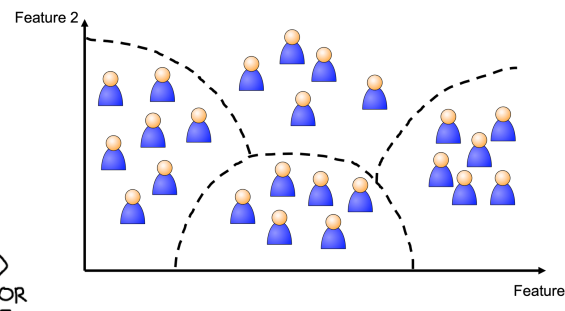
AI is the capability of a computer system to make human-like decisions and therefore mimic human cognitive functions, such as problem-solving and learning. With AI a computer system uses logic and math to simulate the reasoning that allows people to learn from new information and make decisions. **ML** is an application of AI. Indeed, ML is referred to how a computer system develops its intelligence, i.e. it is the process of using mathematical models of data to help a computer learn without direct instruction.



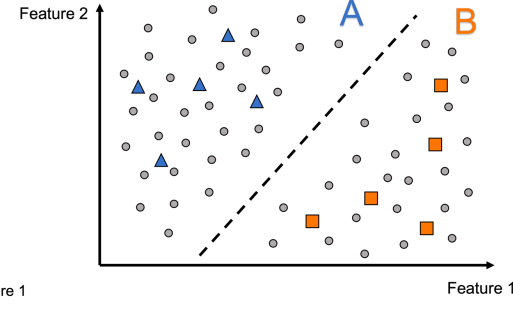
Supervised - regression



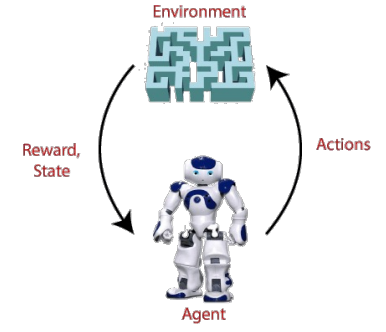
Supervised - classification



Unsupervised

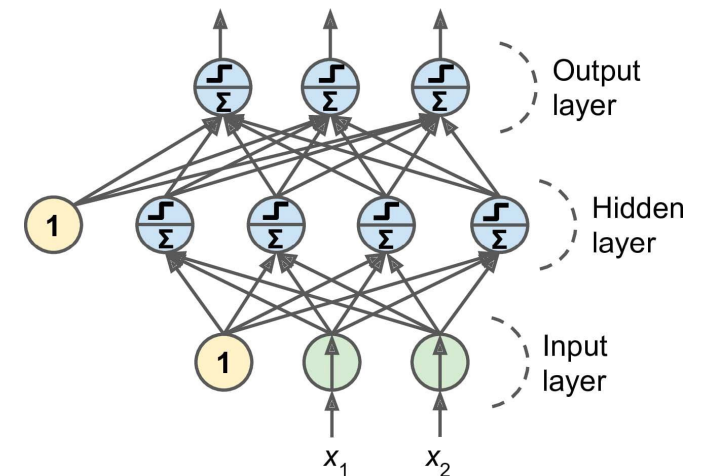


Semisupervised

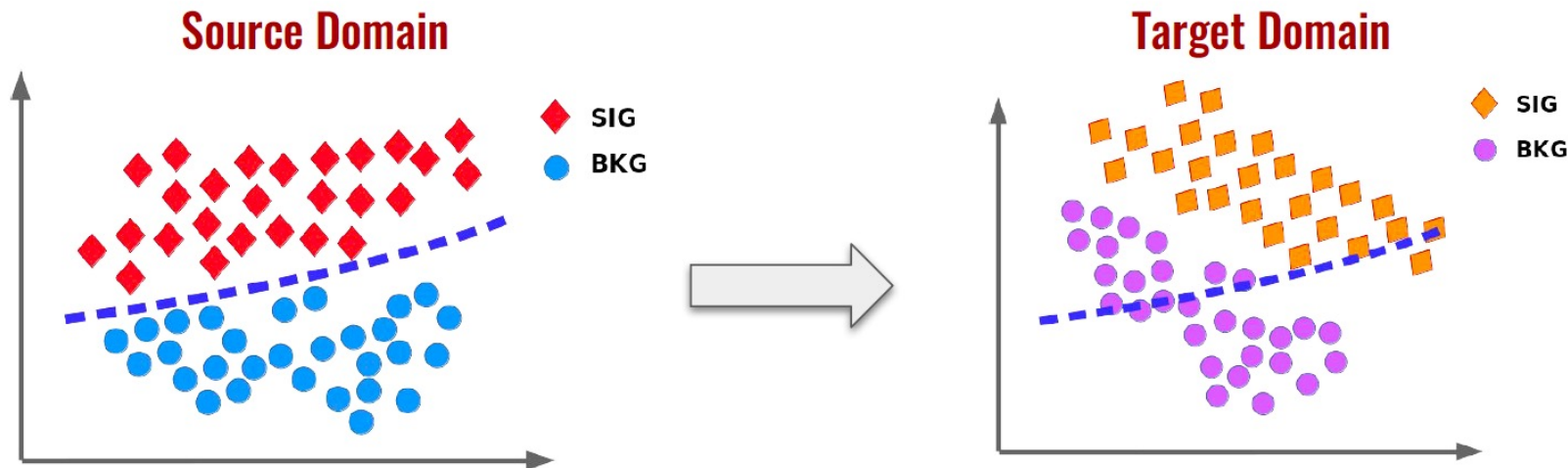


Reinforcement

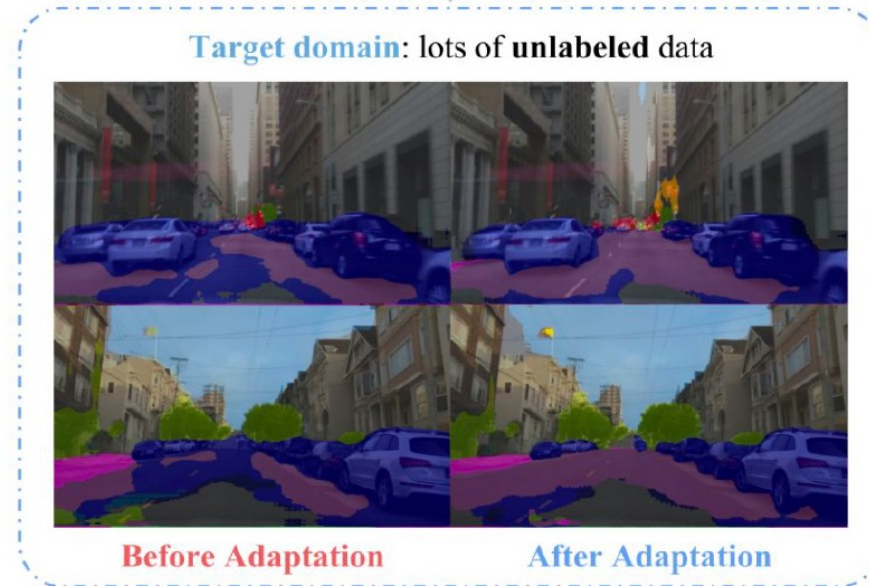
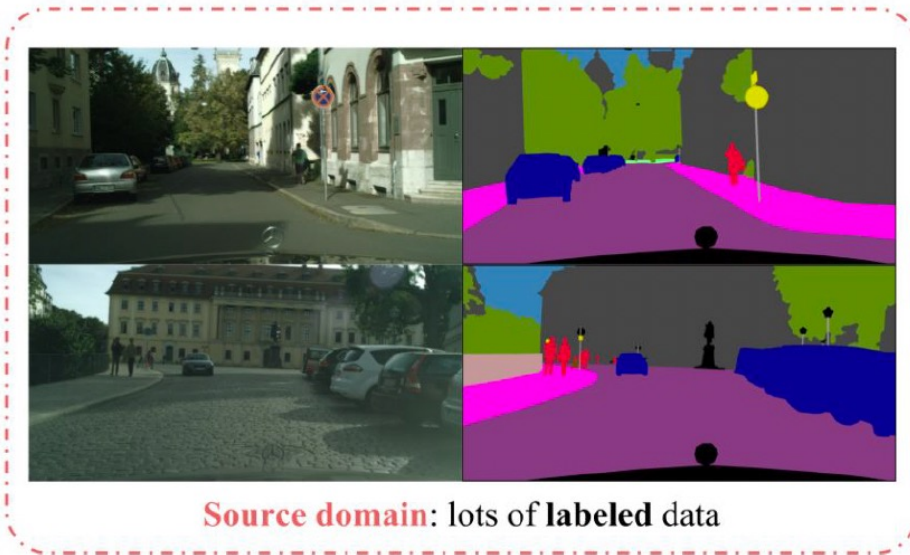
- An **ANN** is a network designed to tackle non-linear learning problems
- The Fully Connected Multilayer Perceptrons (**MLPs**) are made up of single units called Perceptrons
- Perceptrons can be stacked together to build arbitrarily deep custom networks;
- The NN learns during the **training** process by receiving **input patterns** together with the corresponding true target variable and finding the **best set of weights**
- The weights are used to **predict the output with unseen data**.



- **Domain Adaptation** (DA) is a particular case of transfer learning (TL) that leverages labeled data in one or more related source domains, to learn a classifier for unseen or unlabeled data in a target domain.
- Two different (but related) distributions D_S (source domain) and D_T (target domain) on $X \times Y$.
 - Goal: learn h from the two domains such that it commits as little error as possible on the target domain D_T , i.e. train a NN in one dataset (source), securing good performance and accuracy in a different dataset (target).
- Different ways to achieve DA, unsupervised or (semi-)supervised:
 - focus on an Adversarial Deep Learning approach;
 - main idea: find a representation space that is common to source and target domains.



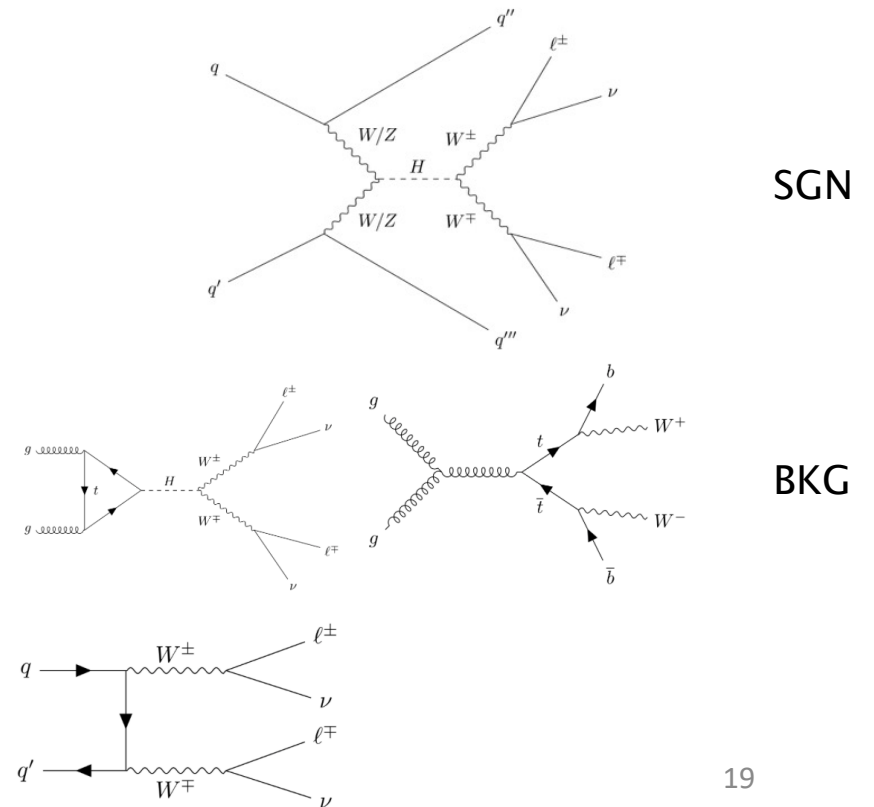
➤ Example of DA in Computer Vision



- Training Machine Learning classifiers to distinguish signal from background in *High Energy Physics* is usually based on simulated datasets, at least for the signal component. The simulation of signal relies on theoretical models that often include hypotheses that have not been completely validated, yet. As a consequence, an implicit bias is introduced in the classifier performance that will differ from model to model.
 - **Domain adaptation** can be used to mitigate the dependence of the trained classifier on the theoretical model adopted for training, forcing the classifier Neural Network to actively ignore the information necessary to distinguish different theoretical models.

- Exercise of the hackathon inspired to the published study [EPJC 82 \(2022\), arXiv:2207.09293](#)

- Higgs boson produced through *Vector Boson Fusion (VBF)* and decaying to a pair of W bosons. The final state of interest is the fully leptonic one, i.e. when both W bosons decay to leptons
- We want to build a three-classes feed-forward Deep Neural Network to categorize the events in the VBF (SM and BSM), ggH and BKG classes with the maximum achievable accuracy. At the same time, we want the output probability distributions to be independent of the particular VBF theoretical model used in the training and in the analysis.



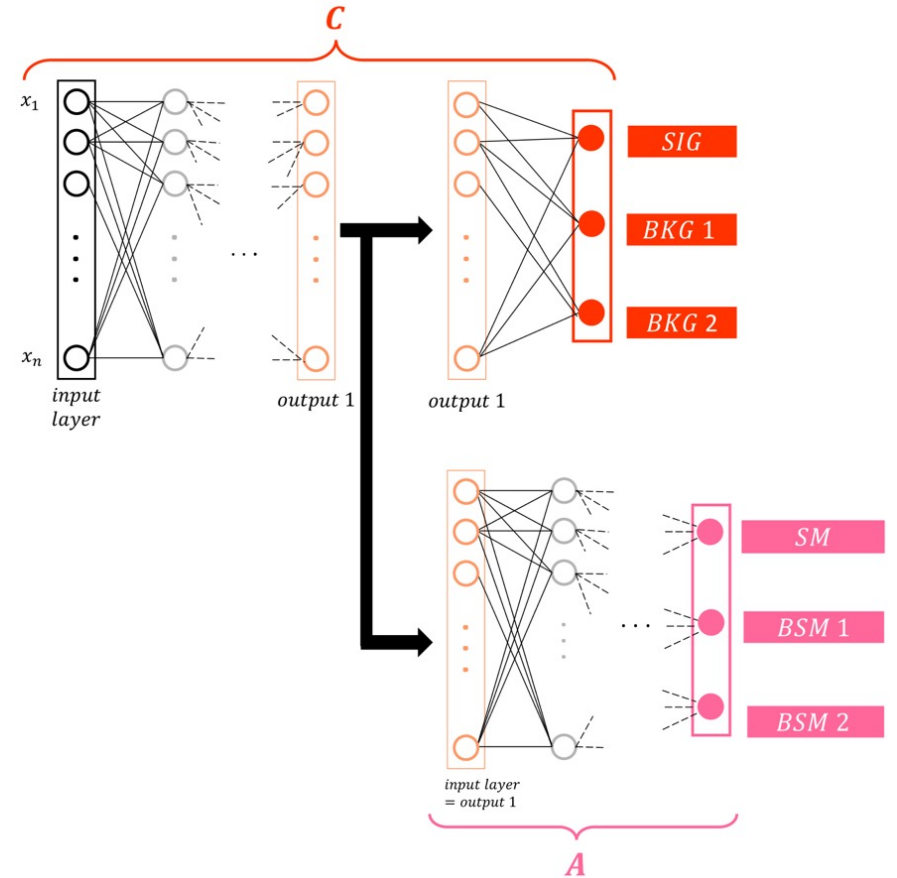
The main components of the ADNN are:

- the **Classifier (C)**: a feed-forward Deep Neural Network with the goal to classify events in 3 classes (VBF signal, ggH and BKG);
- the **Adversary (A)**: a feed-forward Deep Neural Network connected to the second-to-last layer of C (representation) with the goal to classify different signal models.

C and A are trained in a competitive way with the final goal of maximizing the performance of C but simultaneously preventing A to identify the alternative signal models.

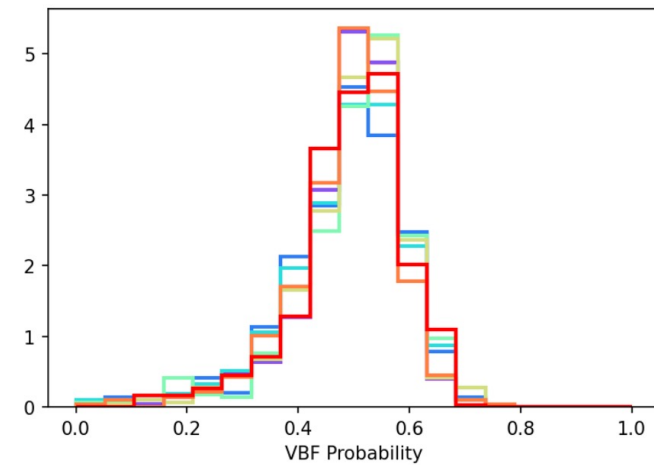
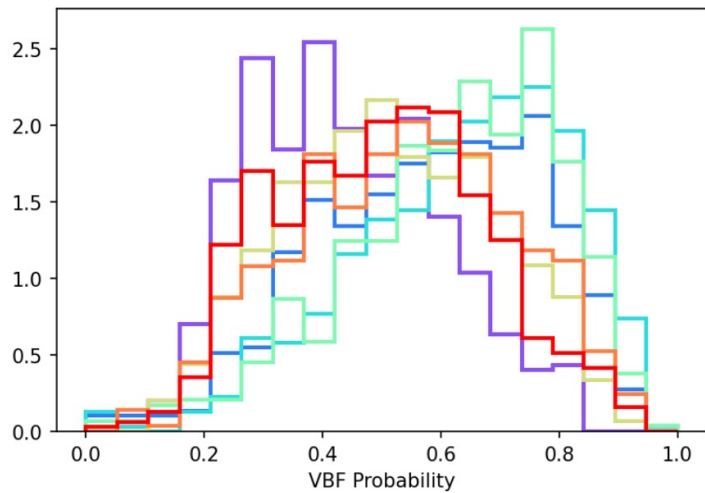
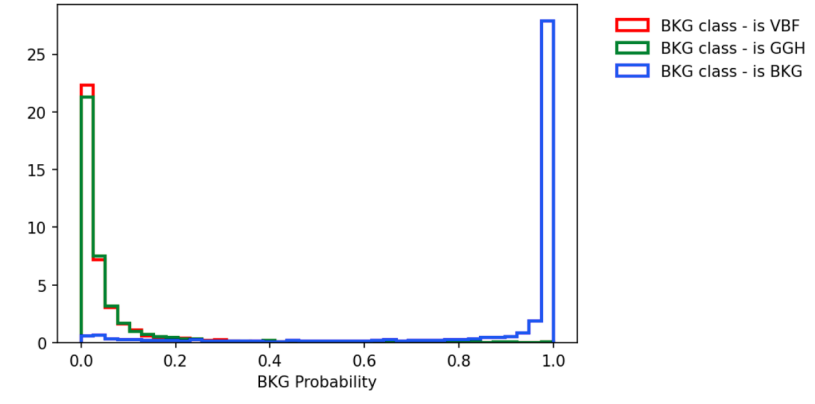
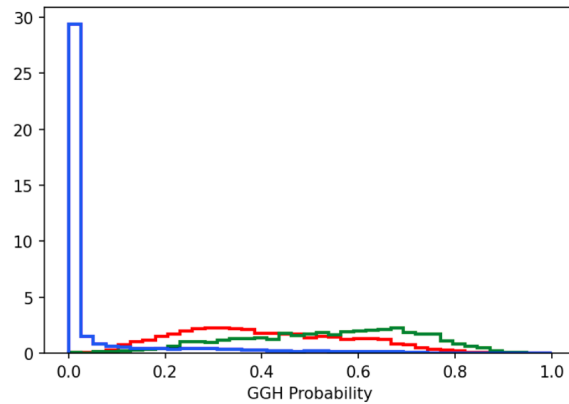
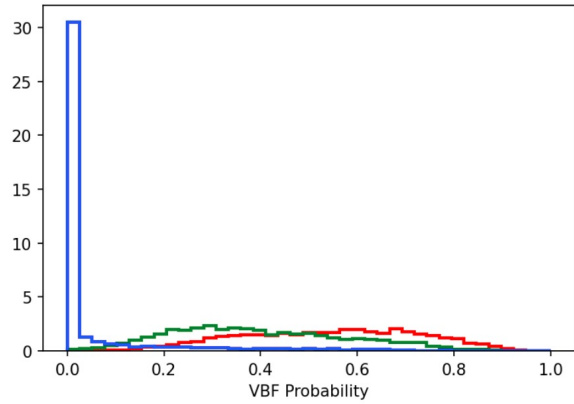
Steps performed during the exercise

- Load, read and understand the data
- Define the DA model according to the Adversarial DNN approach
 - We define 3 Keras Sequential models: model1 is the classifier network without the last output layer, i.e. the second-to-last layer is a representation of the input features; model2 is the adversary network, connected to the second-to-last layer of model1; model3 is just the output layer of the classifier that applies a softmax activation.
- Train the ADNN model and play with the α parameter
- Check plots



$$Loss = Loss(C) - \alpha \cdot Loss(A)$$

[Notebook](#)

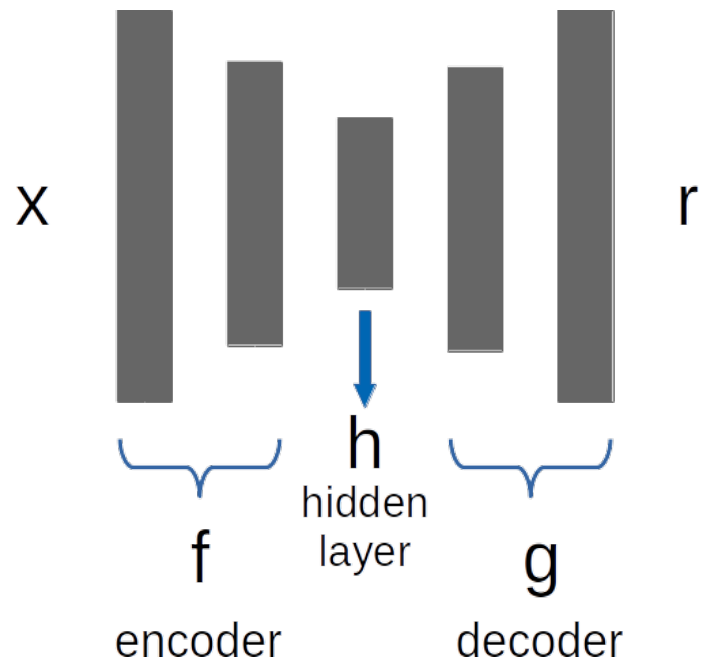


$\alpha=0 \rightarrow$ no DA

Use optuna to find the best α

The best value is a tradeoff between the accuracy of the classifier and the amount of model dependence.

An AutoEncoder Neural Network is an unsupervised learning algorithm that is trained to attempt to copy its input to its output. It is made of two main parts: an encoder and a decoder.



Autoencoders are artificial neural networks capable of learning dense representations of the input data, called latent representations or codings, without any supervision (i.e. the training set is unlabeled). These codings typically have a much lower dimensionality than the input data, making autoencoders useful for **dimensionality reduction**, especially for visualization purposes. Autoencoders also act as feature detectors (**learning features**), and they can be used for unsupervised pre-training of deep neural networks. Lastly, some autoencoders are **generative models**: they are capable of randomly generating new data that looks very similar to the training data.

- **Convolutional Sparse AutoEncoder (CSAE) for breast density**
 - CSAE to extract features and then train a classifier for breast density.

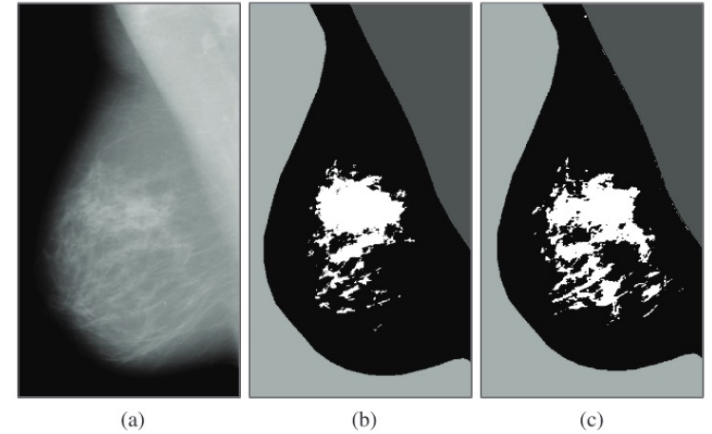
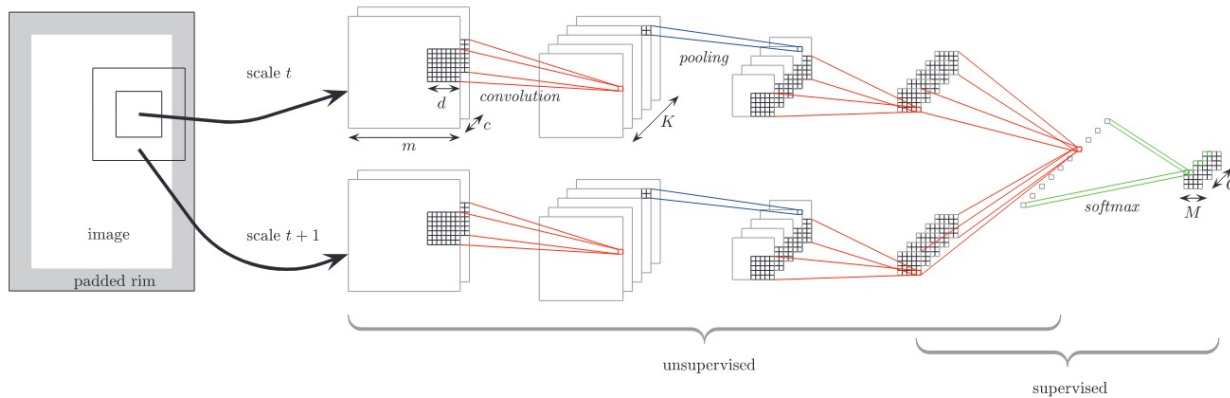
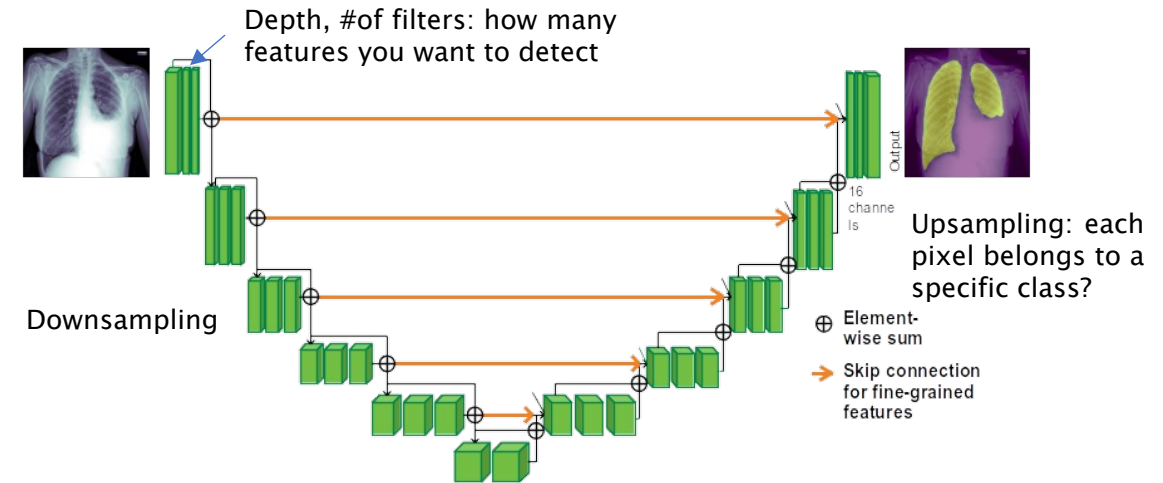
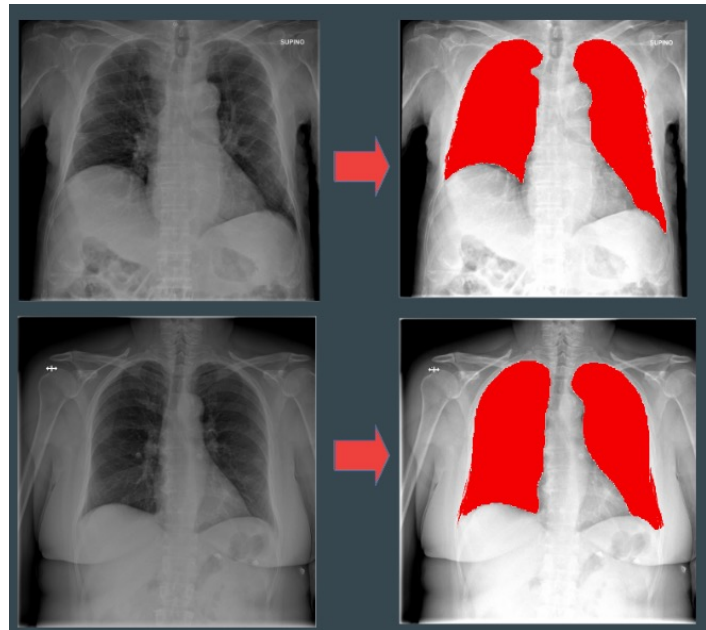


Fig. 3. Automated MD thresholding. Depicted are (a) original image, (b) dense tissue according to expert Cumulus-like threshold, and (c) dense tissue according to CSAE.

It is not often possible to make a priori considerations on distributions or constraints but we need to delete on the images those parts that are not relevant for our scopes → SEGMENTATION

Autoencoders and U-Net – Francesca Lizzi

U-Nets are Fully Convolutional Neural Networks (FCNN) and the state-of-the-art method for medical *image segmentation*. They have an encoder-decoder structure as autoencoders. U-Nets are supervised learning algorithms while AE are unsupervised. U-Nets exploit the skip connections (in orange in the Figure). They consist in connecting different layers through **addition or concatenation**.



Most of the pixels in this image do not belong to the lungs. So they are not useful if we want to analyze lungs and the U-Nets can be used to delete them → exercise of the hackathon

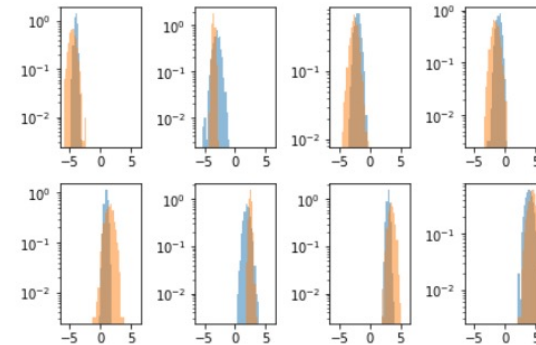
Keras using fashion MNIST dataset



```
class MyModel(Model):
    def __init__(self):
        super(MyModel, self).__init__()
        self.flatten = Flatten(input_shape=(28, 28))
        self.d1 = Dense(128, activation='relu')
        self.drop1 = Dropout(.2, input_shape=(28,28))
        self.d2 = Dense(100, activation='relu')
        self.drop2 = Dropout(.2, input_shape=(28,28))
        self.d3 = Dense(80, activation='relu')
        self.drop3 = Dropout(.2, input_shape=(28,28))
        self.d4 = Dense(60, activation='relu')
        self.drop4 = Dropout(.2, input_shape=(28,28))
        self.d5 = Dense(40, activation='relu')
        self.d6 = Dense(10)

    def call(self, x):
        x = self.flatten(x)
        x = self.d1(x)
        x = self.drop1(x)
        x = self.d2(x)
        x = self.drop2(x)
        x = self.d3(x)
        x = self.drop3(x)
        x = self.d4(x)
        x = self.drop4(x)
        x = self.d5(x)
        return self.d6(x)
```

PyTorch using datasets with gaussian distributions with different means and variance



```
class MyMLP(nn.Module):
    def __init__(self):
        super(MyMLP, self).__init__()

        self.layer1 = nn.Linear(8, 20)
        self.activ1 = nn.ReLU()
        self.layer2 = nn.Linear(20, 50)
        self.activ2 = nn.ReLU()
        self.layer3 = nn.Linear(50, 100)
        self.activ3 = nn.ReLU()
        self.layer4 = nn.Linear(100, 1)
        self.activ4 = nn.Sigmoid()

    def forward(self, x):
        out = self.layer1(x)
        out = self.activ1(out)
        out = self.layer2(out)
        out = self.activ2(out)
        out = self.layer3(out)
        out = self.activ3(out)
        out = self.layer4(out)
        out = self.activ4(out)
        return out
```

- What is generative modeling? Overview of state-of-the-art-algorithms
 - The interest of Scientific Community towards **Generative Models** has grown in recent years thanks to Deep Learning developments.

Some applications

Generative modeling algorithms have been mainly developed for applications in **Computer Vision** to create, transform or improve a broad set of images composed of human faces, animals, landscapes, cartoons, sketches, photos and much more.

Generative models can be mainly used for:

- **Data augmentation**
 - Creation of new and never-seen instances according to the reference dataset
- **Super resolution**
 - Enhancing the resolution of an input image keeping its quality as high as possible
- **Inpainting**
 - Reconstruction of missing pixels in the input images keeping realism and consistency
- **Denoising**
 - Removing noise from input instances minimizing the loss of information
- **Translation**
 - Creation or transformation of images from a domain to another (i.e. text-to-image translation)

Text-to-image with DALL·E 2

"Teddy bears mixing sparkling chemicals as mad scientists as a 1990s Saturday morning cartoon"

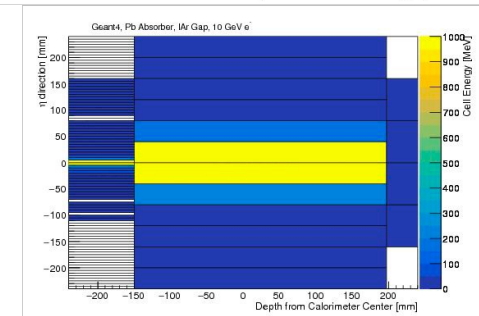
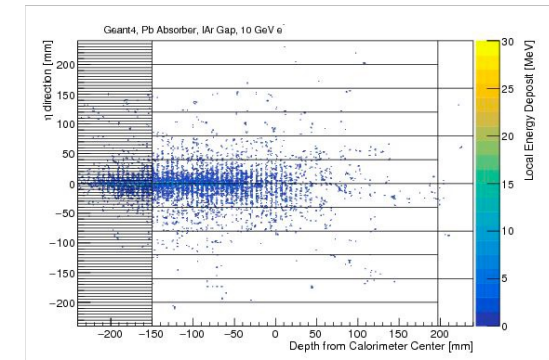
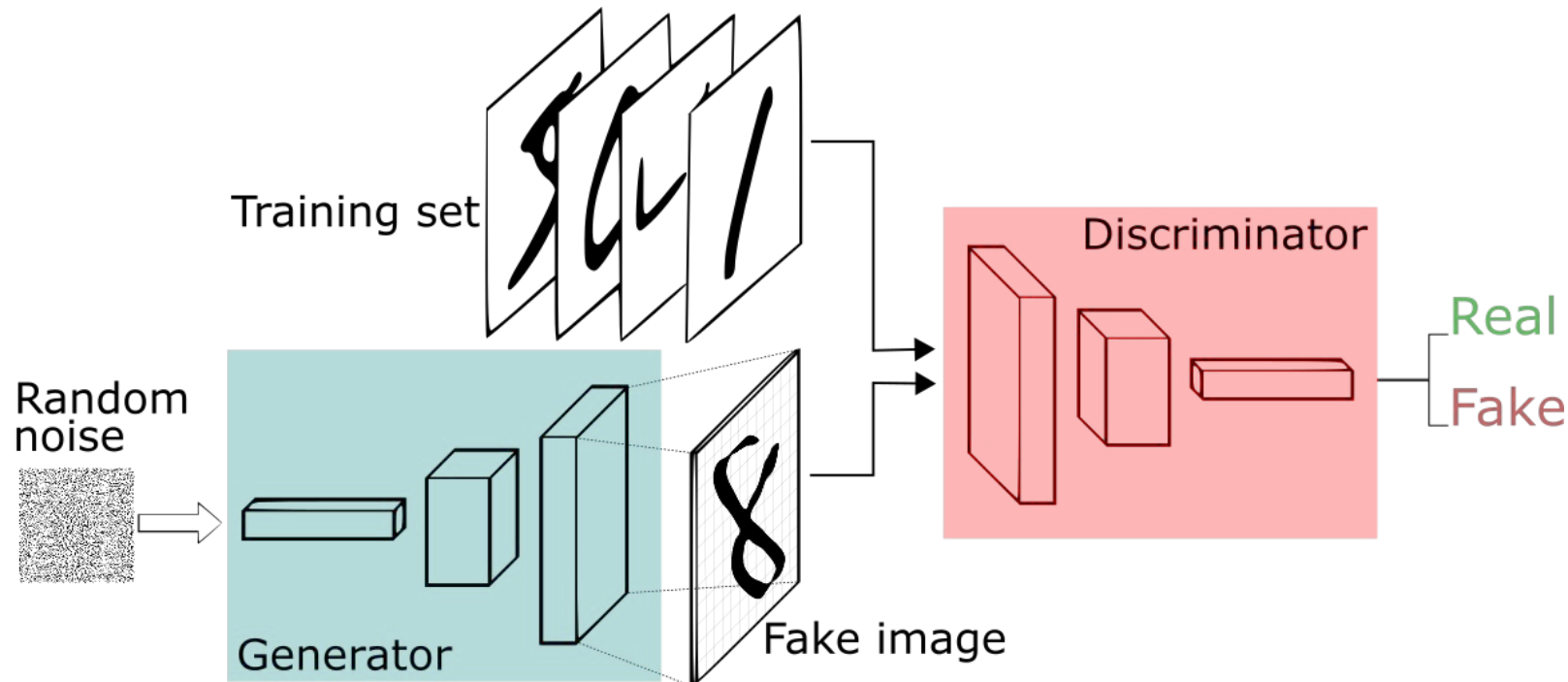


Images taken from the OpenAI [blog post](#)

Generative Adversarial Networks (GAN) rely on the simultaneous training of two neural nets:

- the **discriminator network** (D) is trained by a classification task to separate the generator output from the reference dataset;
- the **generator network** (G) is trained by a simulation task to reproduce the reference dataset trying to fake the discriminator.

The discriminator goal is to **maximize** the separation between the generator output and the real data, while the generator, driven by the discriminator errors, aims to **minimize** the differences with the data.



CaloGAN fast simulation
from arXiv:1712.10321

A **Variational AutoEncoder** (VAE) is an *autoencoder* (AE) whose training is regularized to avoid overfitting and ensure that the latent space has good properties that enable generative process. Also VAEs are based on the simultaneous training of two NNs, i.e. encoder and decoder.

- Contrary to GANs, VAEs training doesn't rely on an adversarial procedure and the system goal is to minimize the distance between encoded-decoded data and the initial data (**reconstruction error**).
- Contrary to AEs, VAEs encode an input as a **distribution** (with good properties) over the latent space rather than a single point to enable generative capabilities.

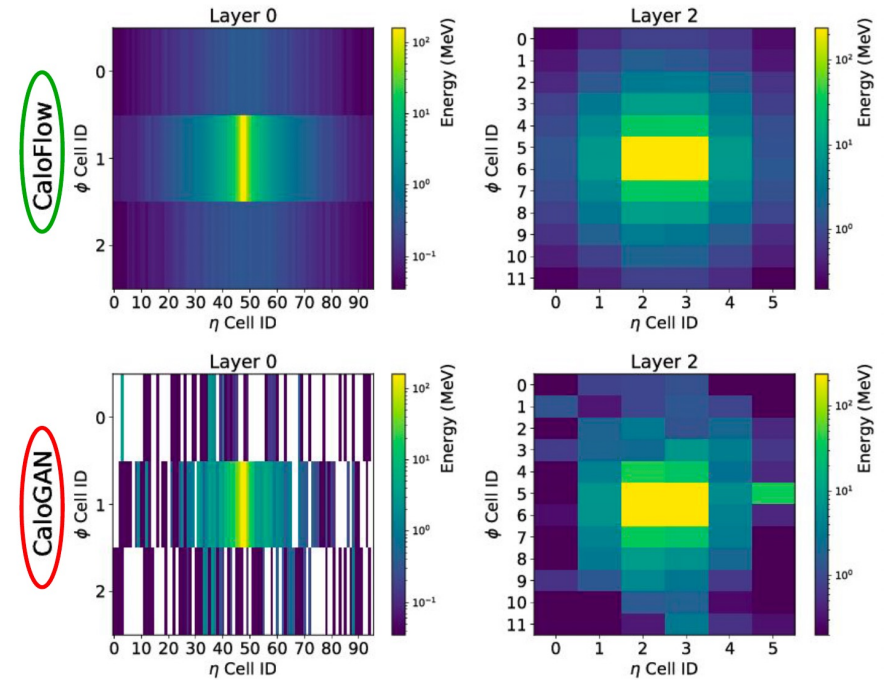
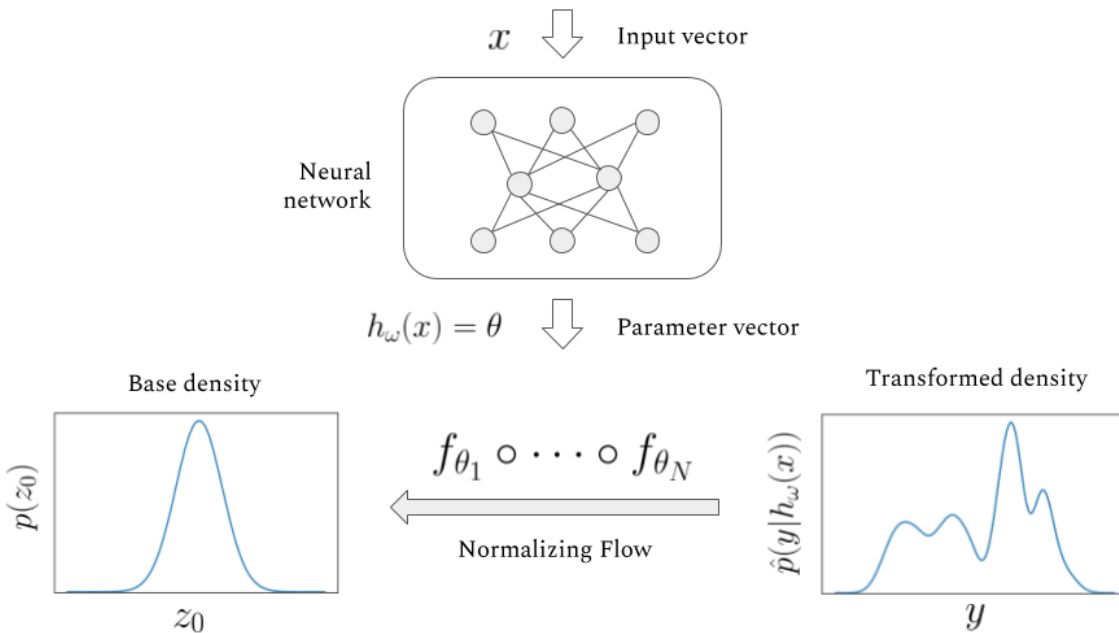
The VAE training is driven by the following **loss function**:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) = \|x - d_{\theta}(z)\|^2 + D_{KL}(\mathcal{N}(\mu(x; \phi), \sigma(x; \phi)) \parallel \mathcal{N}(0, I))$$

where the first term is the **reconstruction error**, while the second one is a **regularization term** that forces the distribution induced by the encoder to have good properties for the **generative process**.

VAEs have been less used than GANs since the **limited capabilities** of the generative model provided, e.g. it generates fuzzy data

- A **Normalizing Flow** is a transformation of a simple probability distribution (e.g., a standard normal) into a more complex distribution by a sequence of invertible and differentiable mappings.



the GAN did not learn to cover the full available phase space

Giorno 3

- State of art and future goals of AI in Medical Image Analysis
 - **Radiologists** can guide the introduction of AI into healthcare. They **will not be replaced by AI**, which, in turn will:
 - standardize the level of reporting across different clinical centres
 - speed up the diagnosis process and allow radiologists to perform more value-added tasks
 - **AI algorithms** for medical imaging **must be effectively evaluated** before they are used in clinical practice. The performance obtained in the R&D stage is difficult to maintain in the clinical use. → Both the generalizability of AI algorithms and the benefits of AI-assisted care relative to conventional care should be proved

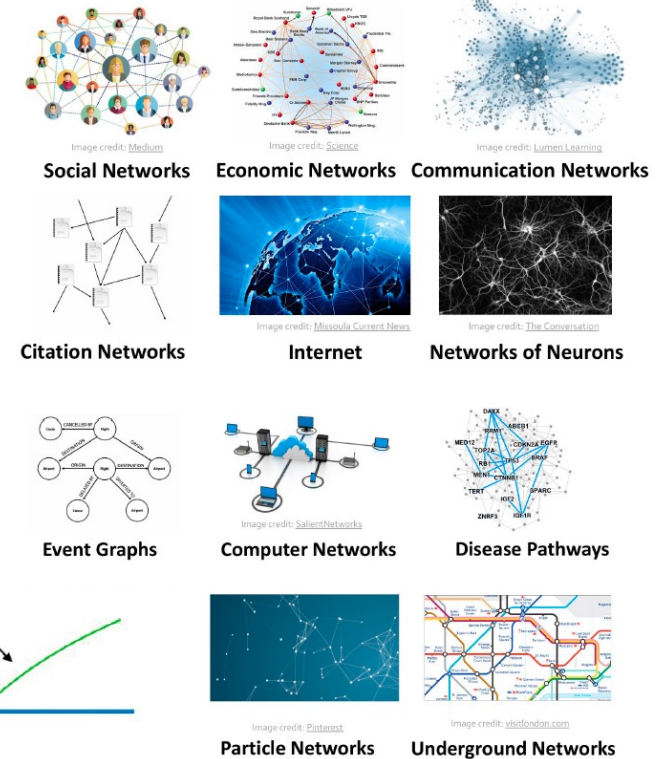
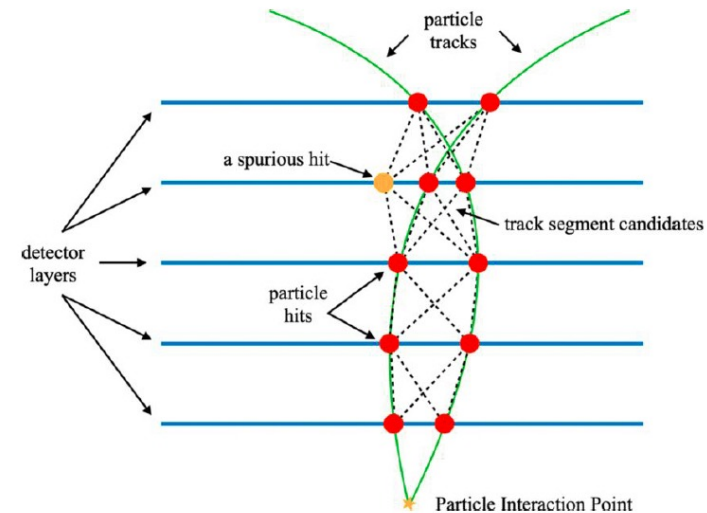
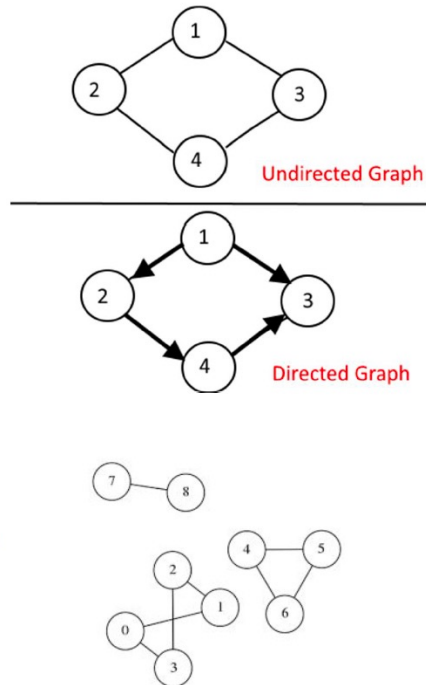
- Open Issues and Challenges:
 - **Limited availability of annotated data** → samples of limited size.
 - Data annotation is an extremely time consuming task. It involves collection of additional information stored in other storing system, expertise in segmenting meaningful regions in images, specific knowledge to assign class labels
 - **Solutions** are: Transfer Learning, data augmentation, use of GAN
 - **Mining data from multiple sources**
 - Integration of the complementary information encoded in omics data, electronic health records (HER), imaging data is expected
 - Missing values → data imputation techniques (replacing with 0, mean/median, most frequent, kNN approach)
 - **Reliability of AI-based systems, i.e robustness to inappropriate inputs**
 - Image type/quality can be evaluated by another AI algorithm, and possibly discarded if not appropriate
 - **Explainability (XAI)**

Graph Neural Networks – Andrea Rizzi

- Data in many case is not easily fitting tensor structure
- Plenty of datasets can be represented with graphs. Such a representation is especially important when there are links between objects
- Graphs are useful also when your data is a list of values that would result in a sparse format if you try to bin it and put it on a grid

Some language dictionary

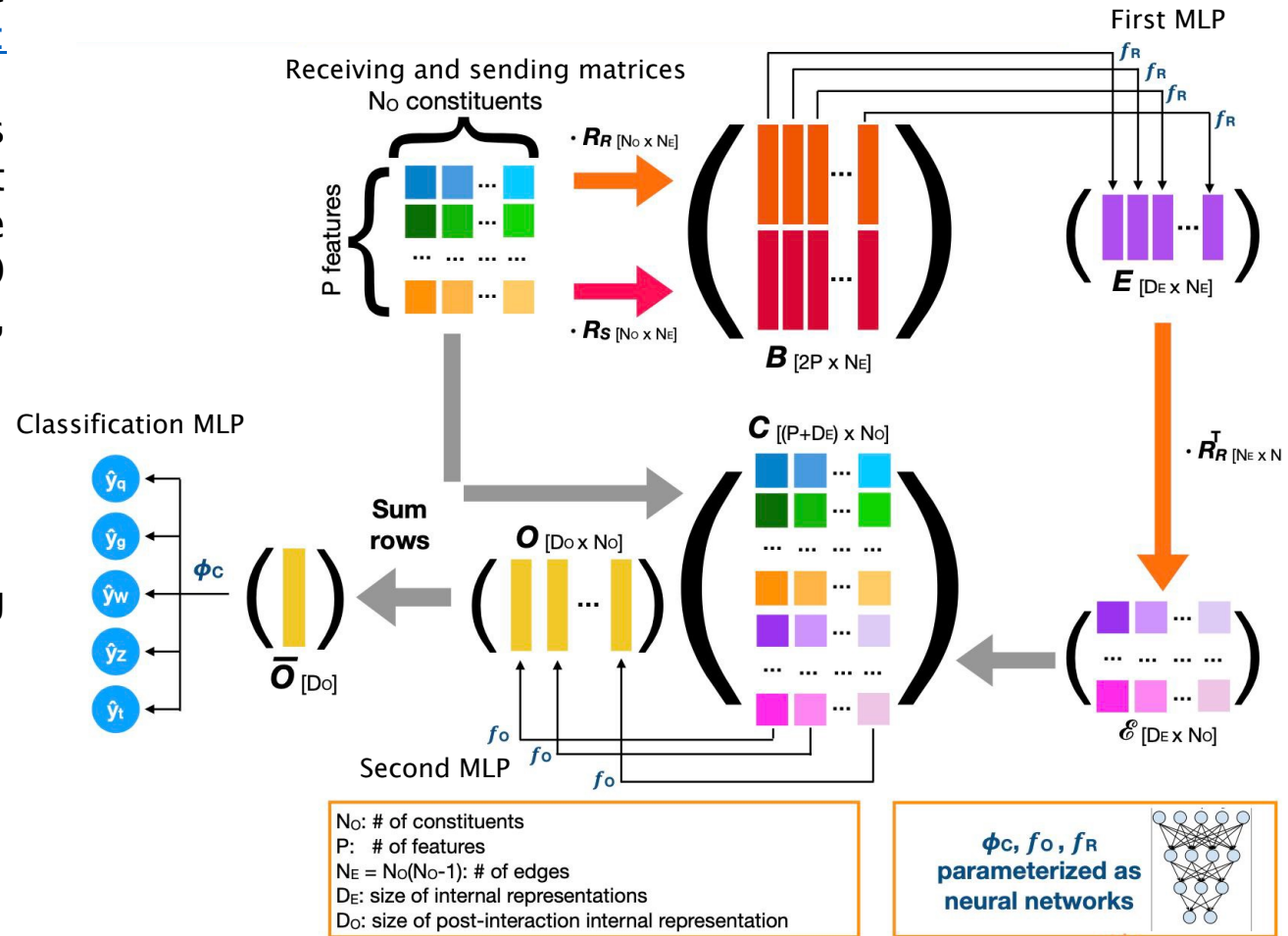
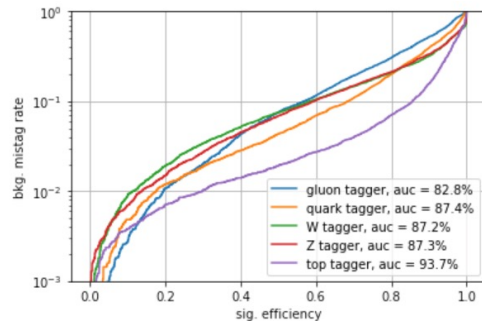
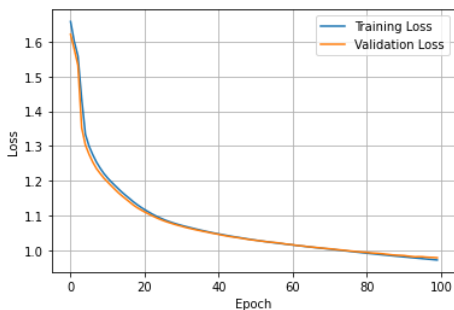
- Vertices
 - the nodes of the graph
- Edges
 - the connections between the vertices
- Degree
 - the number of edges of a given vertex (i.e. the number of vertices connected to it)
- Directed vs undirected
 - edges having a specific direction or being bi-directional
- Connected component
 - a subset of the graph where the vertices are linked to each other by a path (i.e. a set of edges)



Exercise of the hackathon: Interaction networks for the identification of boosted $H \rightarrow b\bar{b}$ decays ([DOI: 10.1103/PhysRevD.102.012010](https://doi.org/10.1103/PhysRevD.102.012010))

- The problem consists in identifying a given jet as a quark, a gluon, a W, a Z, or a top, based on a jet image, i.e., a 2D histogram of the transverse momentum (pt) deposited in each of 100x100 bins of a square window of the (η, ϕ) plane, centered along the jet axis.

- Steps performed during the hackathon
 - Download the data
 - Write in PyTorch the GNN architecture
 - Train the model and check if it is learning correctly
 - Plot the AUC



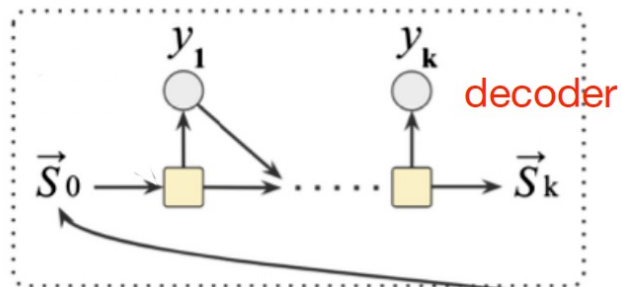
[Notebook](#)

Even if the model may need to draw upon information from the entire input, however some parts of it will be more relevant than others. The **attention mechanism** provides a way to identify such parts

REMINDER: SEQ-TO-SEQ LSTM/GRU

Encoder-Decoder RNNs

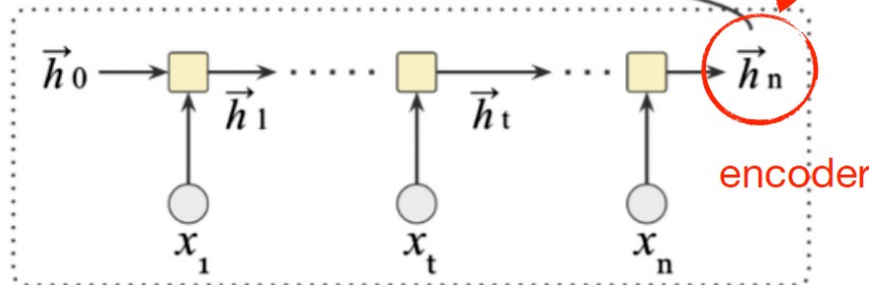
Vector to sequence



seq2seq models (used for example in machine translation tasks) represents a first example of such attempt to create a **context vector** (the cell state of the LSTM/GRU) from the input

use of LSTM or GRU cells allows to “memorise” relevant terms that are far from the current element in the sequence and that are crucial to solve the task

Sequence to vector



limitation: LSTM/GRU becomes ineffective for very long sequence, unless implemented in complex StackedRNN architectures that are impossible to train in an acceptable time due to the recursive (i.e. non parallelizable) intrinsic structure of a RNN

intuitive idea of the attention mechanism: one forms a representation for the entire input, but different parts of the input are weighted differently according to the task at hand. By making the weights a learnable component, the network can learn to attend to the relevant parts of the input

Transformers – Stefano Giagu

Transformers are recent DNN architectures based on the attention mechanism that have gradually replaced RNNs in mainstream NLP tasks, and that can also often compete/surpass (when trained with very large datasets) CNNs and RNNs in vision and in time-domain related tasks

- A. Vaswani et al. “Attention is All You Need” (2017) arXiv:1706.03762

➤ there is a strong link between Graph Neural Networks and Transformers (a Transformer is a GNN with a multi head attention as aggregation function)

Solution	# of parameters
OpenAI – GPT-1 (2018)	117M
OpenAI – GPT-3 (2020)	175B
Google – BERT (2019)	340M
Google – PaLM (2022)	540B
OpenAI – DALL-E (2021)	12B
OpenAI – DALL-E 2 (2022)	3.5B
Meta – Galactica	125B

PaLM

```

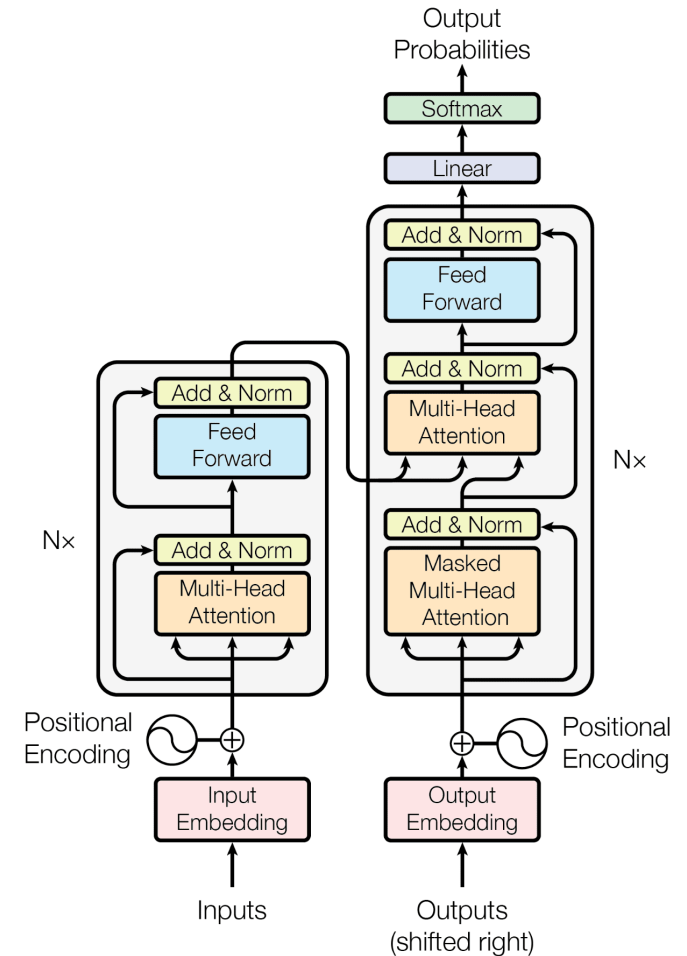
Prompt → Model Response
// Translate from C to Python
int add_one ( int x ){
  int m = 1;
  while ( x & m ) {
    x = x ^ m;
    m <<= 1;
  }
  x = x ^ m;
  return x; }

def add_one(x: int):
  m = 1
  while (x & m):
    x = (x ^ m)
    m <<= 1
  x = (x ^ m)
  return x

Prompt → Model Response
def exercise5():
  """Marie ordered one chicken meal that costs $12, 5 packs of milk that cost $3 each, 4 apples that cost $1.50 each, and some boxes of pizza. Marie paid a total of $50. How many boxes of pizza did Marie order if box costs $8.50?"""

  total_paid = 50
  cost_of_pizza_per_box = 8.50
  cost_of_meal = 12
  cost_of_milk_per_pack = 3
  cost_of_apples_per_apple = 1.50
  packages_of_milk = 5

  return float((total_paid - cost_of_meal - cost_of_milk_per_pack * packages_of_milk - cost_of_apples_per_apple * 4) / cost_of_pizza_per_box)
  
```



Input:

Question: Translate the following Math formula:

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s}$$

into plain English.

Answer:

GALACTICA Output:

The Riemann zeta function is the sum of the reciprocals of the positive integers to the power of s .

Answer: The Riemann zeta function is the sum of the reciprocals of the positive integers to the power of s .

Exercise of the hackathon: same goal of the GNN exercise but now with transformers → jet tagging (e.g. classification of jets originating from different particles: *gluons, quarks, Ws, Zs, and top quarks*).

- Better explanation of the physics problem and inspection of the data

```
def __init__(self, embed_dim, hidden_dim, num_heads, dropout=0.0):
    super(AttentionBlock, self).__init__()

    self.layer_norm_1 = nn.LayerNorm(embed_dim)
    self.attn = nn.MultiheadAttention(embed_dim, num_heads)
    self.layer_norm_2 = nn.LayerNorm(embed_dim)

    self.mlp = nn.Sequential(
        nn.Linear(embed_dim, hidden_dim),
        nn.GELU(),
        nn.Dropout(dropout),
        nn.Linear(hidden_dim, embed_dim),
        nn.Dropout(dropout)
    )

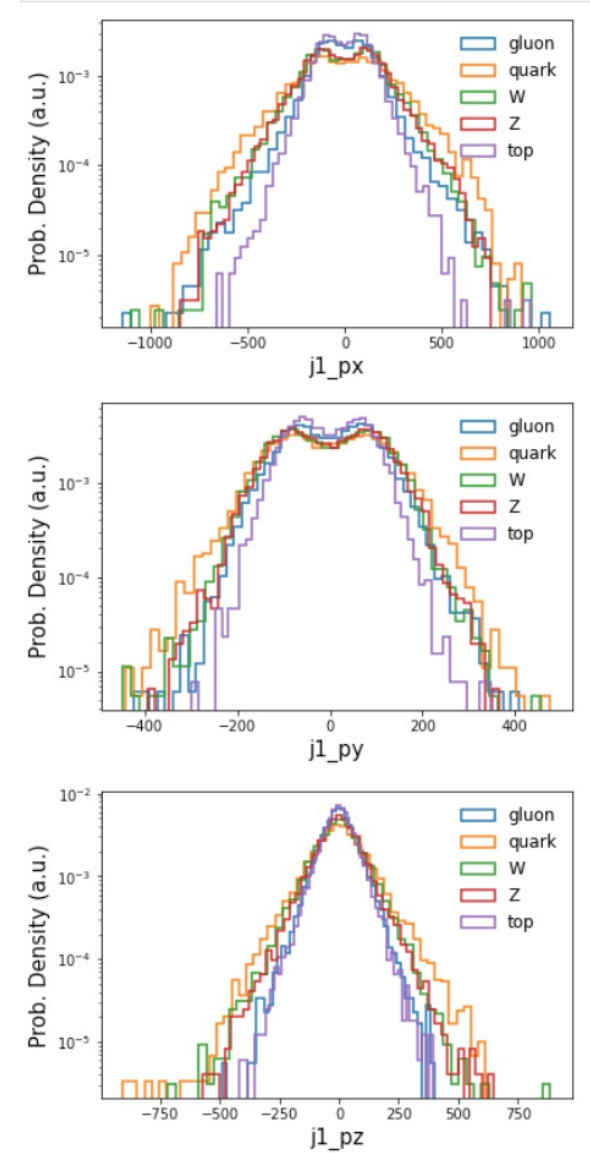
def __init__(self, input_dim, embed_dim, hidden_dim, num_heads, num_layers, num_classes, dropout=0.0):
    super(MyTransformer, self).__init__()

    # input layer (a dense MLP projecting the input in a embed_dim embedding space)
    self.input_layer = nn.Sequential(
        nn.Linear(input_dim, 128),
        nn.GELU(),
        nn.Linear(128, 512),
        nn.GELU(),
        nn.Linear(512, embed_dim),
        nn.GELU()
    )

    # Transformer encoder: stack of num_layers Attention Blocks (embed_dim -> embed_dim)
    self.transformer = nn.Sequential(*[AttentionBlock(embed_dim, hidden_dim, num_heads, dropout=dropout) for _ in range(num_layers)])

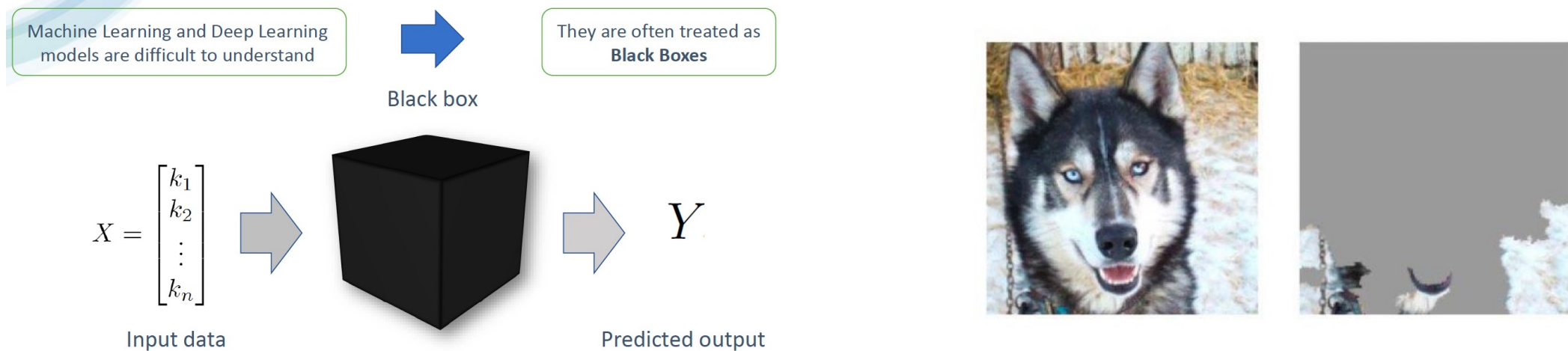
    # MLP Classifier (embed_dim -> num_classes)
    self.mlp_head = nn.Sequential(
        nn.LayerNorm(embed_dim),
        nn.Linear(embed_dim, num_classes)
    )

    self.dropout = nn.Dropout(dropout)
```



[Notebook](#)

Explainability is the degree to which a human can understand the cause of a decision



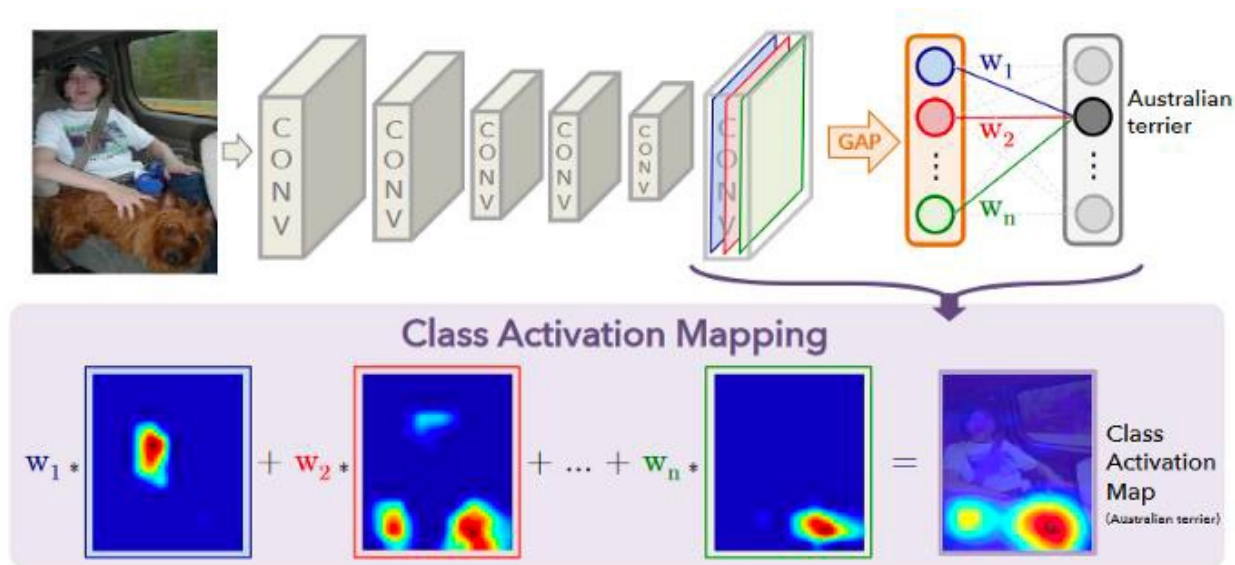
Is feature importance a good solution? It should have the following properties: consistency, accuracy, insightfulness.

- It is not always possible to meet this properties
- It gives global information, not on the single prediction

Feature importance not the best solution to promote explainability → **eXplainable Artificial Intelligence (XAI)** models.

Class Activation Maps (CAM) are a technique to get the discriminative image regions used by a CNN to identify a specific class in the image.

- Use the feature maps of a CNN model as weight to explain a certain prediction

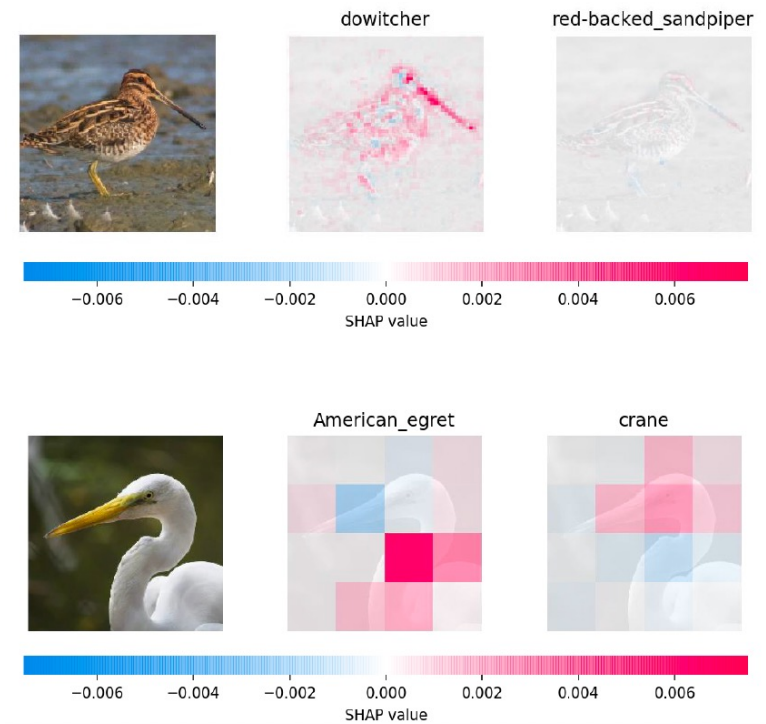
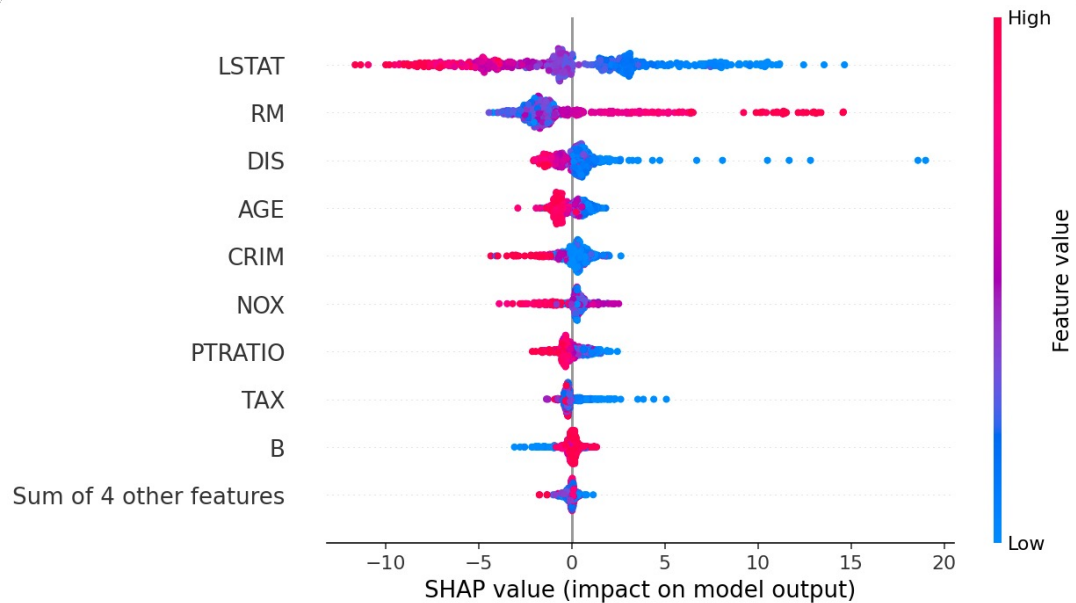


-Area of the image that explains a prediction "Dog"

GAP: Global Average Pooling
Evolution: Grad-CAM. It uses backpropagation to calculate the weights of the maps)

SHAP is a model agnostic explainer

- Its purpose is to "imitate" the model used.
- It gives an understandable explanation of a local prediction of a model by assigning to each feature a value.

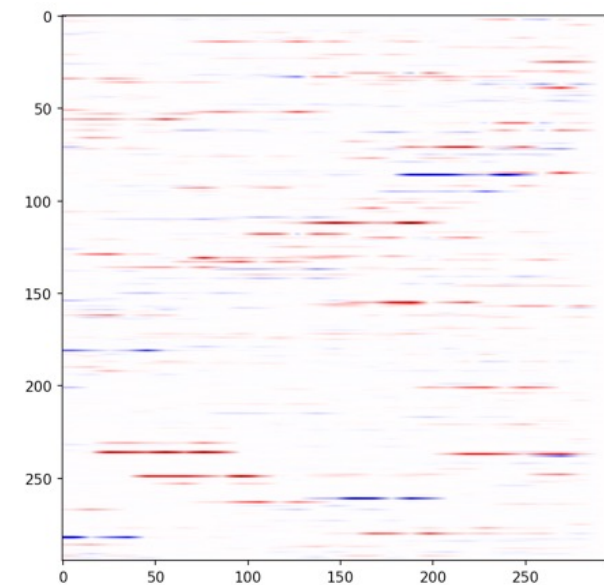
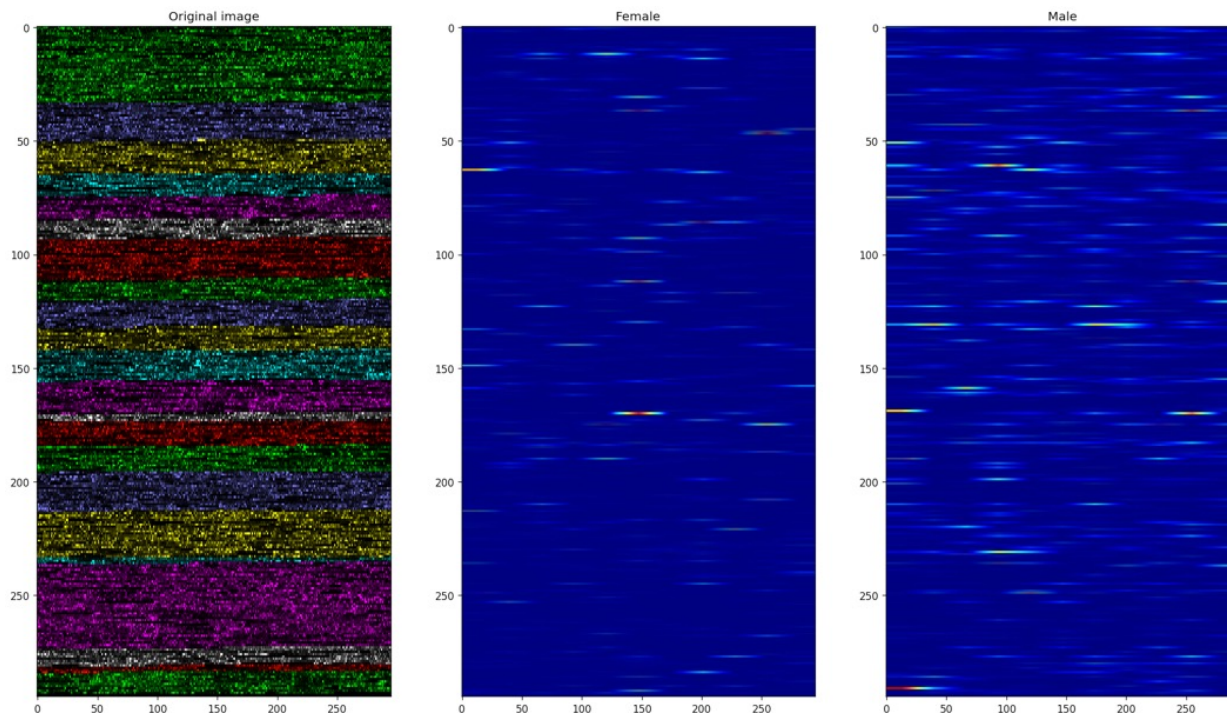


Explainable AI in Medical Physics – Alessandro Fania

In the exercise of the hackathon, the students tried to use explainable AI models to identify patterns in a very long sequence of RNA editing sites sufficient to discriminate men and women.

- Converting the dataset in a sequence of images
- a simple CNN was designed to perform a binary classification between male and female candidates.
- how does the network manage to achieve this conclusion? To explain it GradCAM and SHAP were used

	chr1_136052	chr1_136915	chr1_564461	chr1_564467	chr1_567459
rownames					
SRR1313426	0.465523	0.208592	0.337590	0.229242	0.005040
SRR1433088.2	0.465523	0.208592	0.337590	0.229242	0.007031
SRR1343115.1	0.465523	0.200000	0.337590	0.229242	0.007031
SRR660895.1	0.324680	0.193550	0.337590	0.229242	0.007031
322	0.250647	0.125343	0.337590	0.229242	0.007031
SRR1432123	0.733330	0.250000	0.337590	0.142860	0.007031
88	0.463340	0.208592	0.328383	0.225002	0.005138
SRR1396659	0.465523	0.208592	0.337590	0.229242	0.007031
SRR1436444	0.465523	0.208592	0.337590	0.229242	0.010870
SRR1404969	0.611110	0.133330	0.337590	0.229242	0.007031
(1472, 86683)					



Giorno 4

Let X and Y be respectively the input and output datasets such that $y=f(x)$ with f unknown.
Using Machine Learning (ML), we want to find the best parameterization for f , namely

$$\hat{f}(x; \theta, \varphi) \doteq \hat{f}_{\theta}(x; \varphi)$$

where θ are the **parameters** of our ML model (automatically set during the training process),
and φ are the **hyperparameters** that define such training procedure.

To push the performance of our ML model we can define some score function S (i.e. accuracy, AUC, MSE, BCE, KS-test, ...) over a set of controllable parameters (namely, the hyperparameters) that we typically want to optimize (min/max). This optimization problem is named **hyperparameter optimization**.

Standard methods to search for good hyperparameter candidates are:

- **Manual**;
- **Grid Search** – exhaustive search over pre-specified range;
- **Random Search** – randomized hyperparameter search.

None of these methods exploit the whole history of tests!

Bayesian optimization exploits all the tests performed through the **prior/posterior evolution**.

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

$$p(x) = \int p(x|\theta)p(\theta)d\theta$$

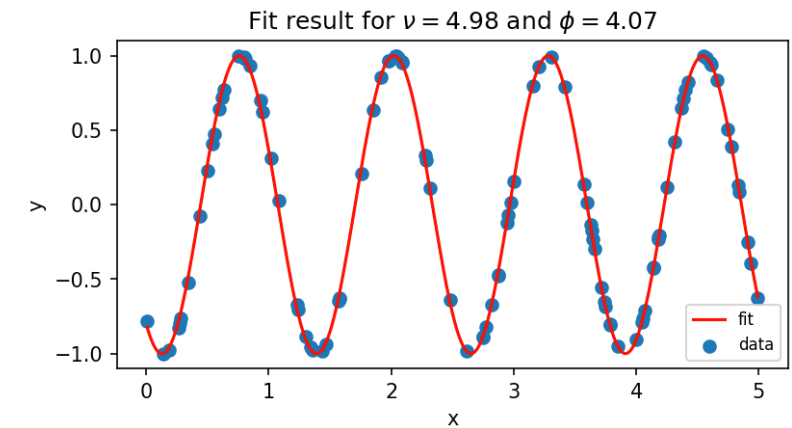
Bayesian optimization can be easily implemented in Python using, for instance, one of the following tools: **Optuna**, **Weights & Biases**, **Ray Tune**, **BoTorch**, **scikit-optimize**

Optuna is an open-source framework designed for automatic hyperparameter optimization.

- Optuna is entirely written in Python and has few dependencies
 - The hyperparameter search space is defined taking into account the **Python syntax**, including conditionals and loops
- Optuna implements efficient optimization algorithms
 - The user has access to state-of-the-art algorithms for **sampling** set of hyperparameters and **pruning** efficiently unpromising trials
- Optuna enables **framework-agnostic** optimization campaigns
 - The user is free to use the preferred Machine Learning framework (i.e. Scikit-Learn, TensorFlow, PyTorch, JAX, ...)

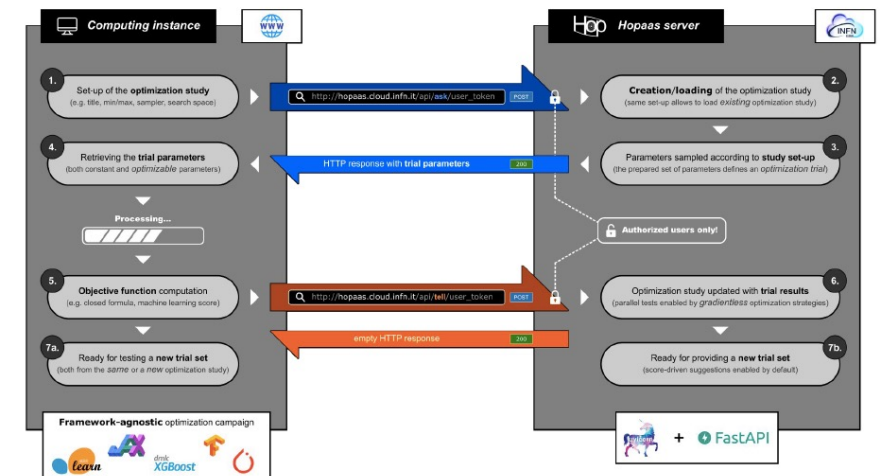


Matteo performed a tutorial



Hopaas (*Hyperparameter Optimization As A Service*) is a service hosted by INFN Cloud that allows orchestrating optimization studies across multiple computing instances via **HTTP requests**. Hopaas provides a set of REST APIs and a web interface to enable **user authentication** and monitor the status of the user studies.

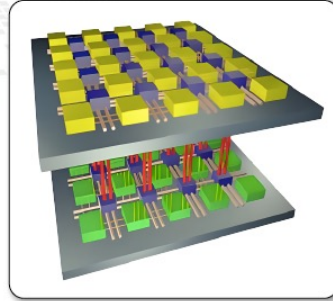
- **Back-end** based on FastAPI;
- Underlying databases powered by PostgreSQL;
- **Bayesian optimization** powered by Optuna;
- Service provided by Uvicorn and NGINX;
- Python **front-end** available (hopaas_client).



Machine Learning on FPGA – Mirko Mariotti

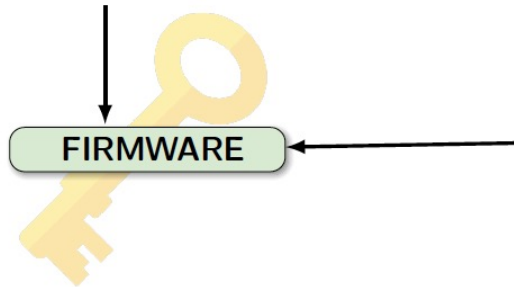
FPGA

A field programmable gate array (FPGA) is an integrated circuit whose logic is re-programmable.

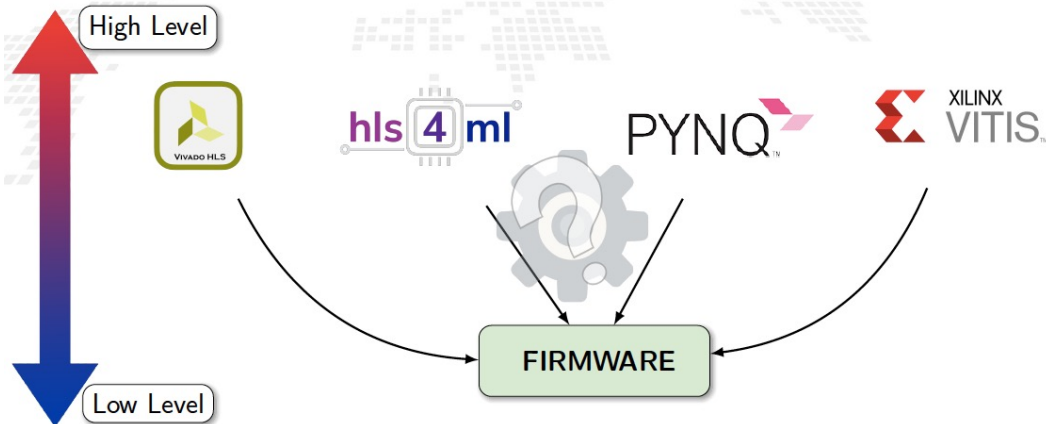


- Parallel computing
- Highly specialized
- Energy efficient

- Array of programmable logic blocks
- Logic blocks configurable to perform complex functions
- The configuration is specified with the hardware description language



Many projects have the goal of abstracting the firmware generation and use process.



CPU	GPU	FPGA
Fixed architecture	Fixed architecture	Adaptable Architecture
Predefined instruction set	Predefined instruction set	No fixed instruction set
Fixed memory hierarchy	Fixed memory hierarchy	Customizable memory hierarchy
Thread-level parallelism	SIMD parallelism	Excels at all types of parallelism

	CPU	GPU	FPGA Sim	FPGA Hw
Engineering time	short	medium	medium	medium
Compilation time	short	short	short	Very long
Debugging	easy	medium	medium	medium
Maintainability	easy	medium	medium	medium

With High Level Synthesis tools

- Programmables to perform a wide range of tasks;
- Low-level/Near-metal implementation of algorithms → low latency;
- Blend the benefits of both hardware and software;

Hardware Description Language High Level Synthesis **BondMachine**

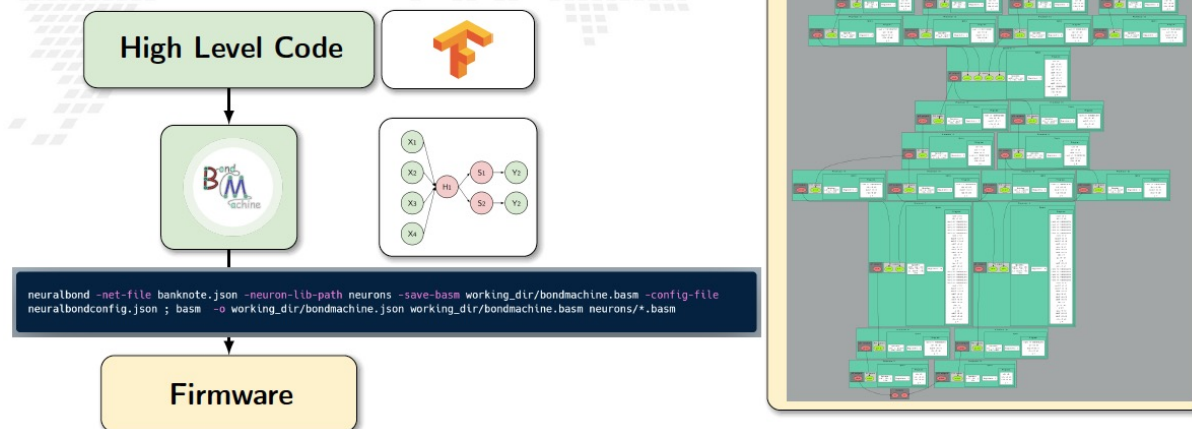
The **BondMachine** is a software ecosystem for the dynamic generation of computer architecture. Architectures with multiple interconnected processors like the ones produced by the BondMachine Toolkit are a perfect fit for Neural Networks and Computational Graphs. Several ways to map this structures to BondMachine has been developed:

- A native Neural Network library
- A Tensorflow to BondMachine translator
- An NNEF based BondMachine composer

The tool **neuralbond** allow the creation of BM-based neural chips from an API go interface. Tensorflow Graphs can be converted to BondMachines with the **tf2bm** tool.

From idea to implementation

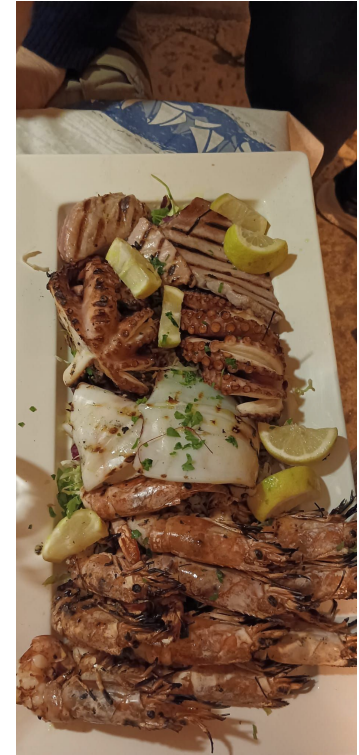
Starting from High Level Code, a NN model trained with **TensorFlow** and exported in a standard interpreted by **neuralbond** that converts nodes and weights of the network into a set of heterogeneous processors.



website: <http://bondmachine.fisica.unipg.it>
code: <https://github.com/BondMachineHQ>

Closing remarks – Lucio Anderlini

- An intensive week where we saw:
 - Cloud and computing technologies
 - Machine Learning algorithms and applications to INFN research
 - Ongoing and future developments
 - and very good food!!!
- The notebooks of the hackathon are available [here](#)
- We have ML_INFN groups in most INFN units that can support researchers and coordinate the access to resources and support!
- Hackathon plans for 2023
 - Basic-Level hackathon in June
 - Advanced-Level hackathon in person in autumn with a good GPU per participant and a high tutors-to-student ratio



If you are working on a project involving the use of Machine Learning and you want to present at one of the weekly ML_INFN meetings (every Monday at 16:00), please contact me (luca.giommi@cnaif.infn.it). You are welcome!

Questions?