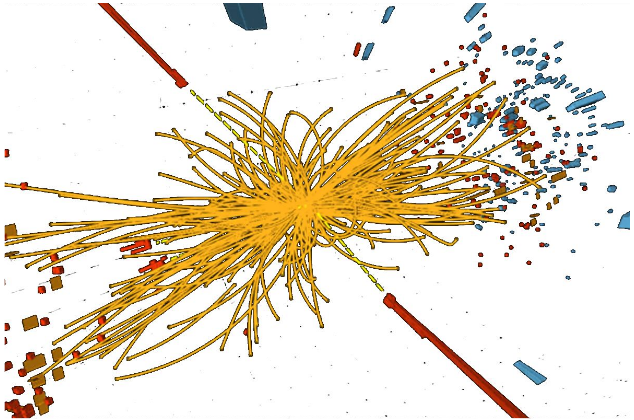
Deep neural networks resizing for online event selection in future collider experiments

M. Cristoforetti, A. Di Luca, F. M. Follega, R. Iuppa, D. Mascione

University of Trento, Fondazione Bruno Kessler, INFN TIFPA

A lot of data



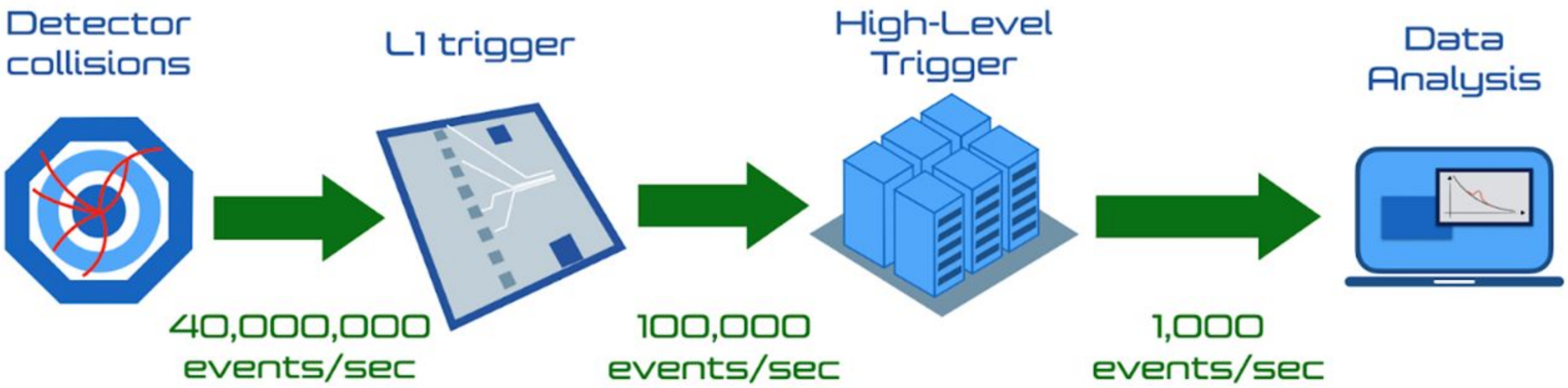
Collider experiments produce a **huge amount of data**.

At the Large Hadron Collider we have

- one collision every 25 ns (= **40 Million collisions/sec**)
- **thousands of particles** emerging from each collision
- **1 MB of data** recordered at each collision by big detectors

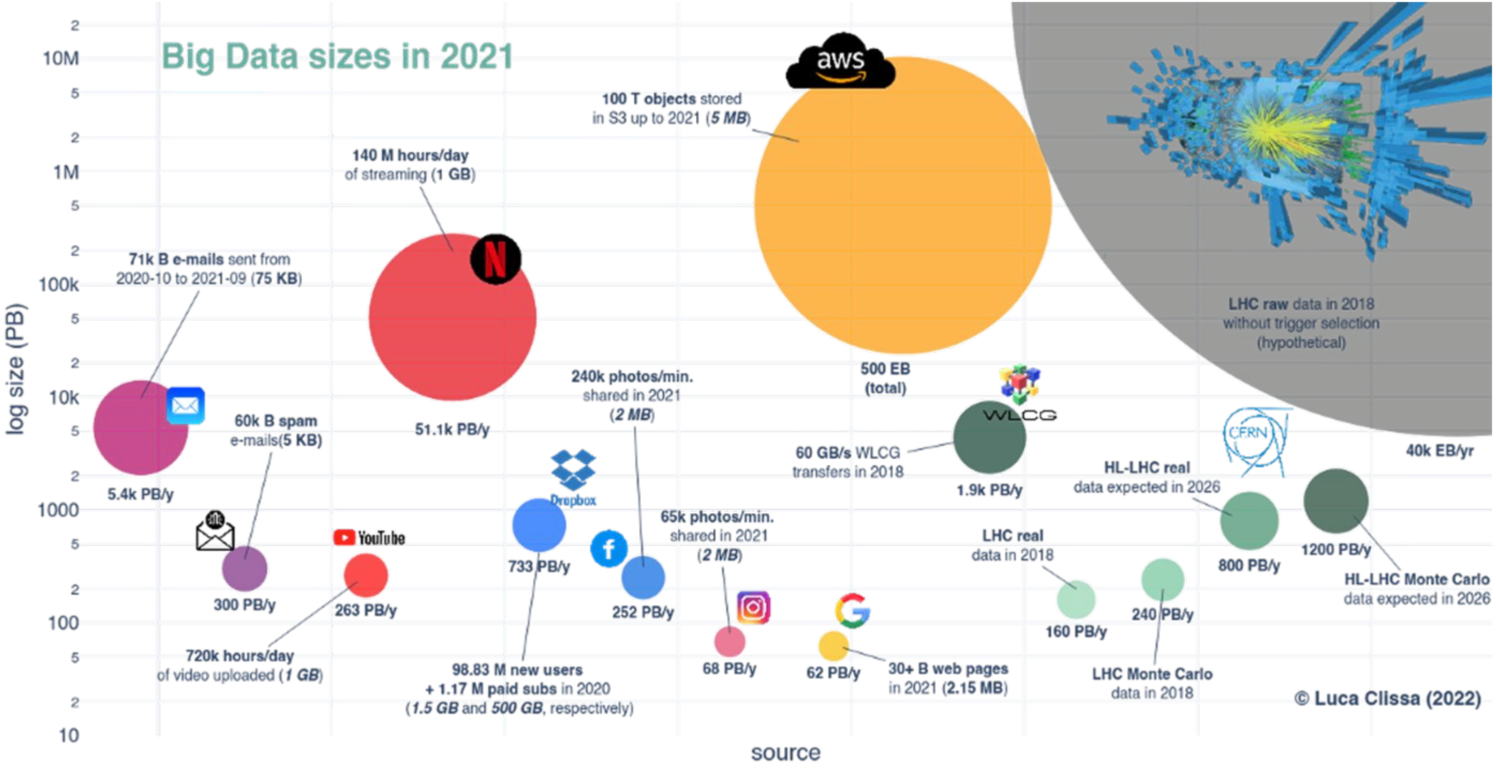
Data reduction system at the LHC

Not all data produced at the LHC are stored: they are first filtered with a trigger chain



source

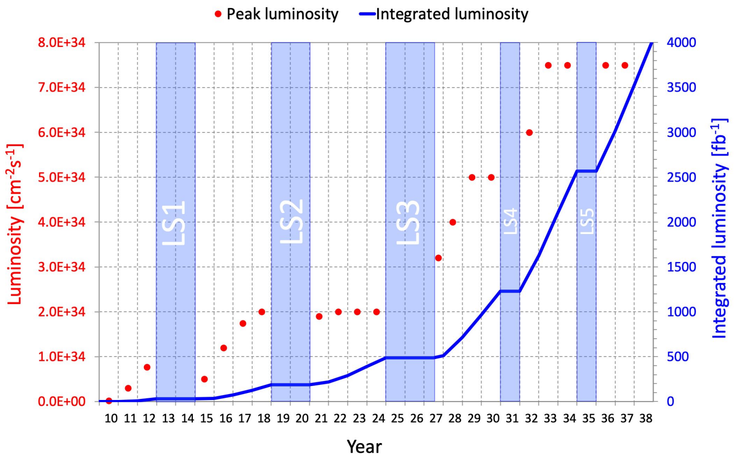
Data amount at the LHC



source

HL-LHC

Things will get even worse with HL-LHC: the HL-LHC will produce more than 250 inverse femtobarns of data per year and will be capable of collecting **up to 4000 inverse femtobarns** (1 inverse femtobarn equates to 100 million million collisions).



source

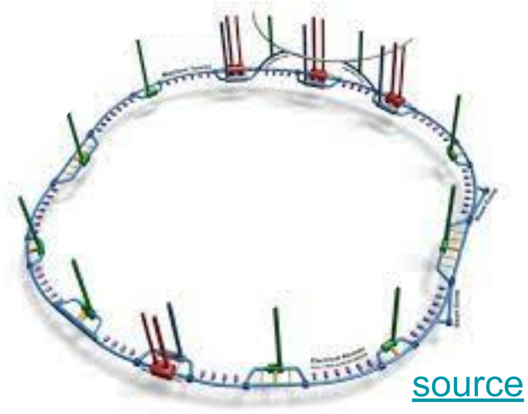
LHC → **HL-LHC**

5 interactions per beam cross
➔ ~ 40 collisions/event

40 interactions per beam cross
➔ ~ 200 collisions/event

Future colliders

At the **FCC-hh** huge amounts of data will be produced (**O(TBytes/s) expected**). We will need to make intelligent decisions as close to the detector as possible and to provide at least O(10) data reduction factors after front-end readout.



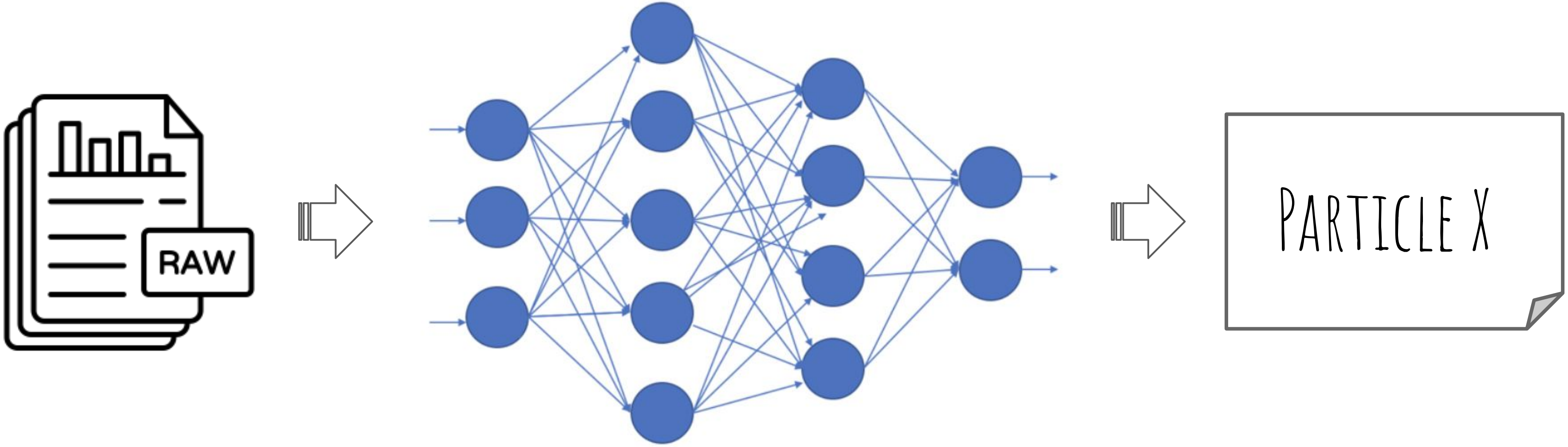
source

source

Also for **FCC-ee and ILC**, that will explore triggerless approaches, the event selection will be committed to strategies directly interfaced with the detector's front-end readout.

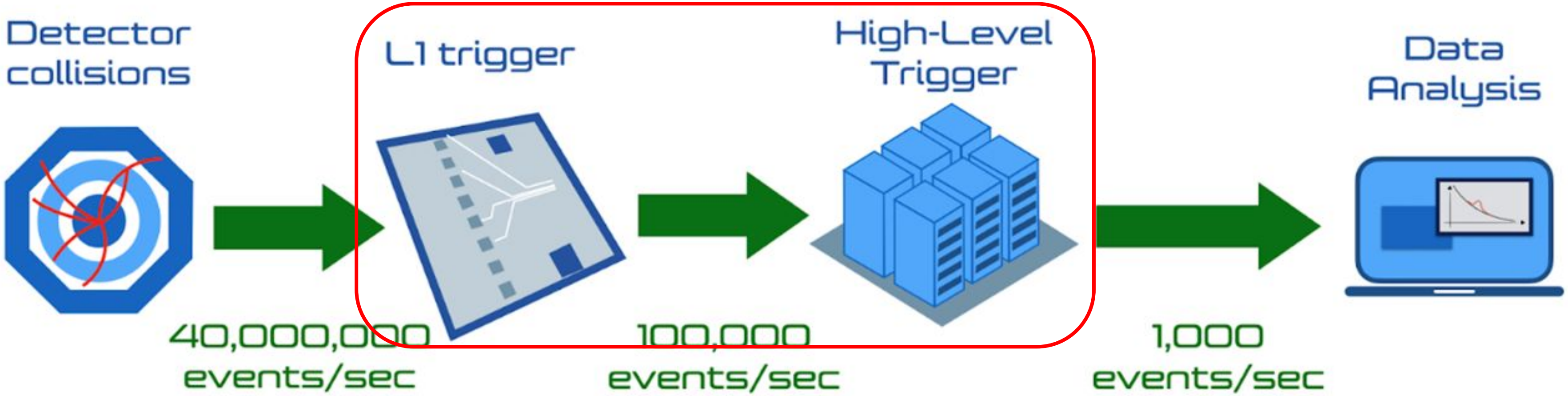
Deep Learning at rescue

We know how to get from the data the answers we want, but the process is **slow**. Deep Neural Networks can help us making the process **faster** by giving us those answers directly from raw data.

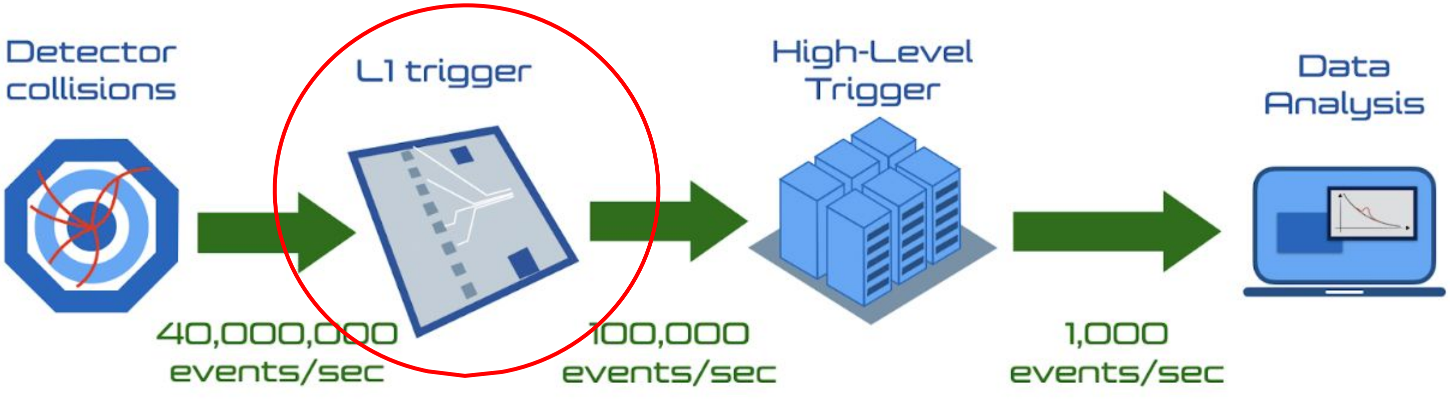


Deep Learning at trigger level

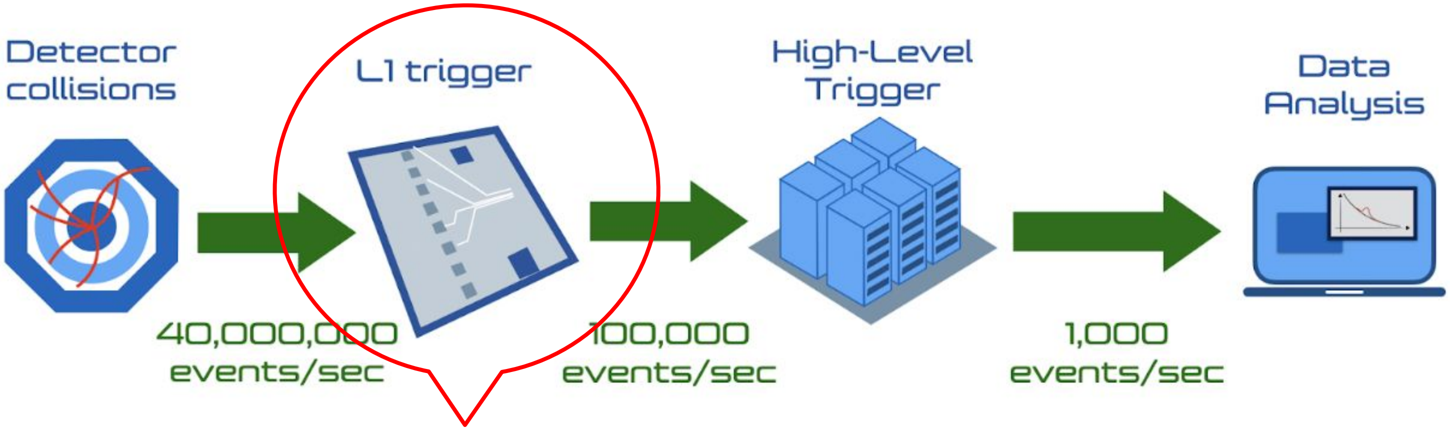
Deep Learning need to be used in between collisions and data analysis, where the event selection happens



Deep Learning at L1 trigger

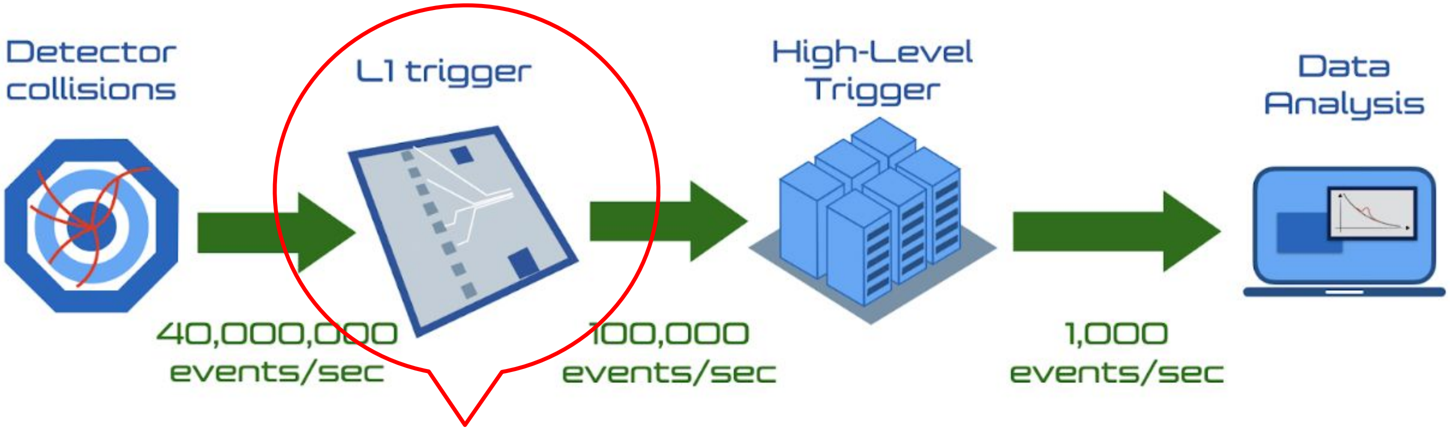


Deep Learning at L1 trigger



L1 of data processing typically uses custom hardware with FPGAs

Deep Learning at L1 trigger



L1 of data processing typically uses custom hardware with FPGAs

Let's run Deep Neural Networks in real-time on FPGAs to improve event selection!

FPGAs

FPGAs (Field-Programmable Gate Arrays) are programmable integrated circuits.

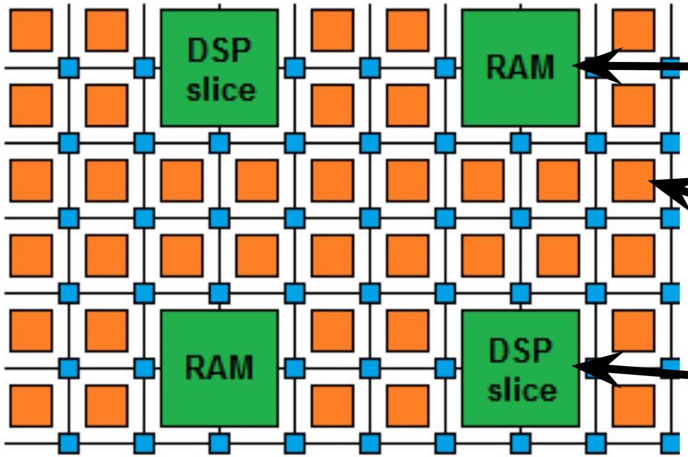


Image source

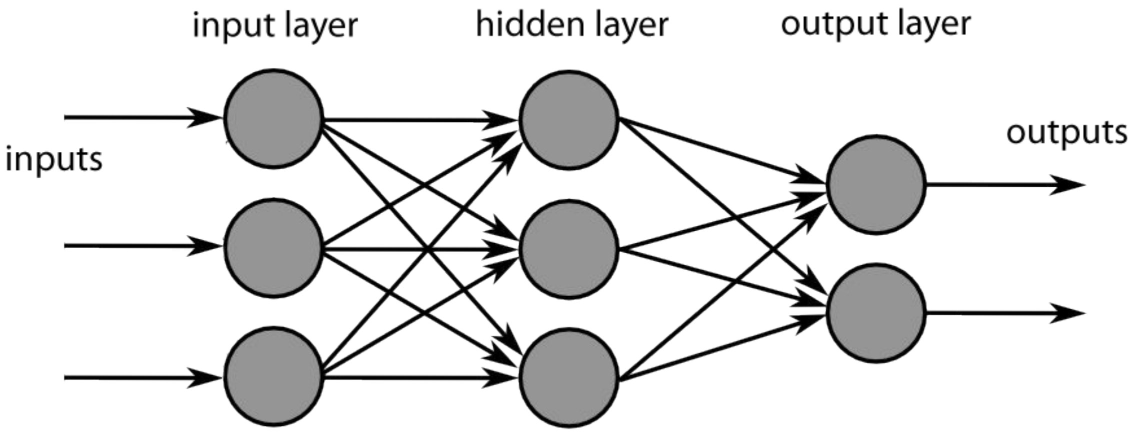
Random Access Memories to store constant values

Logic cells for simple arithmetic operations

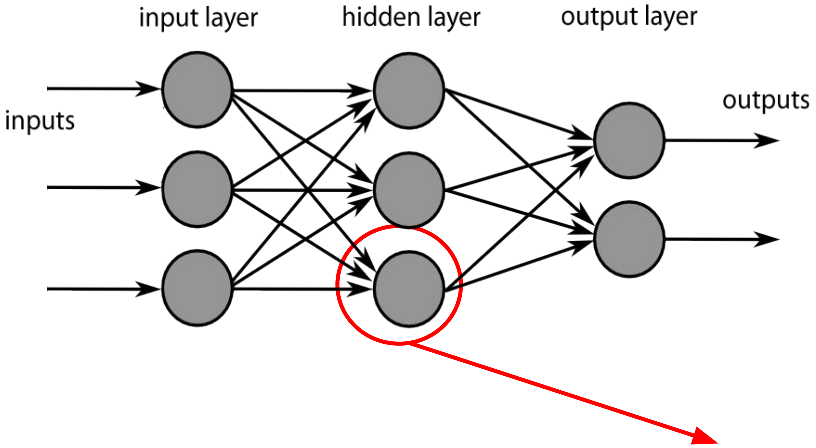
Digital Signal Processors to perform multiplications

Deep Neural Networks

An Artificial Neural Network is a **computational model** that has layers of interconnected nodes.
A Deep Neural Network has more than one hidden layer.



Through training, the neural network **learns** to recognize a **pattern** in the input data.



Nodes convert weighted inputs to outputs. The **weights keep getting updated** in the process of learning.

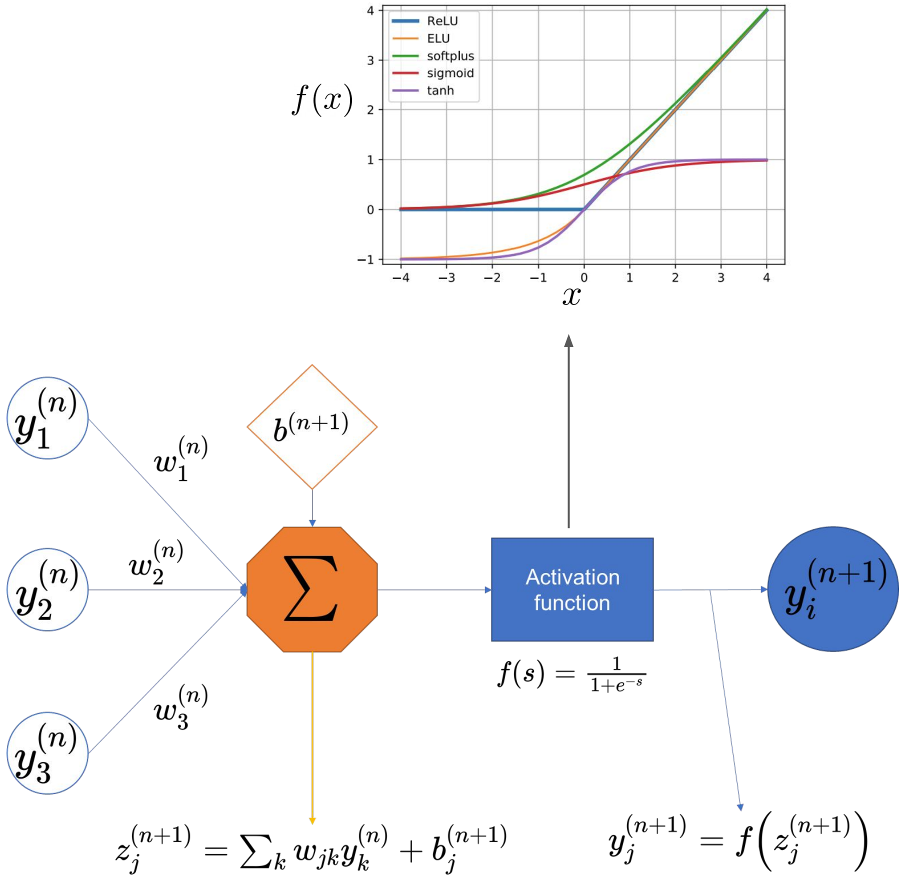


Image: F. Di Clemente

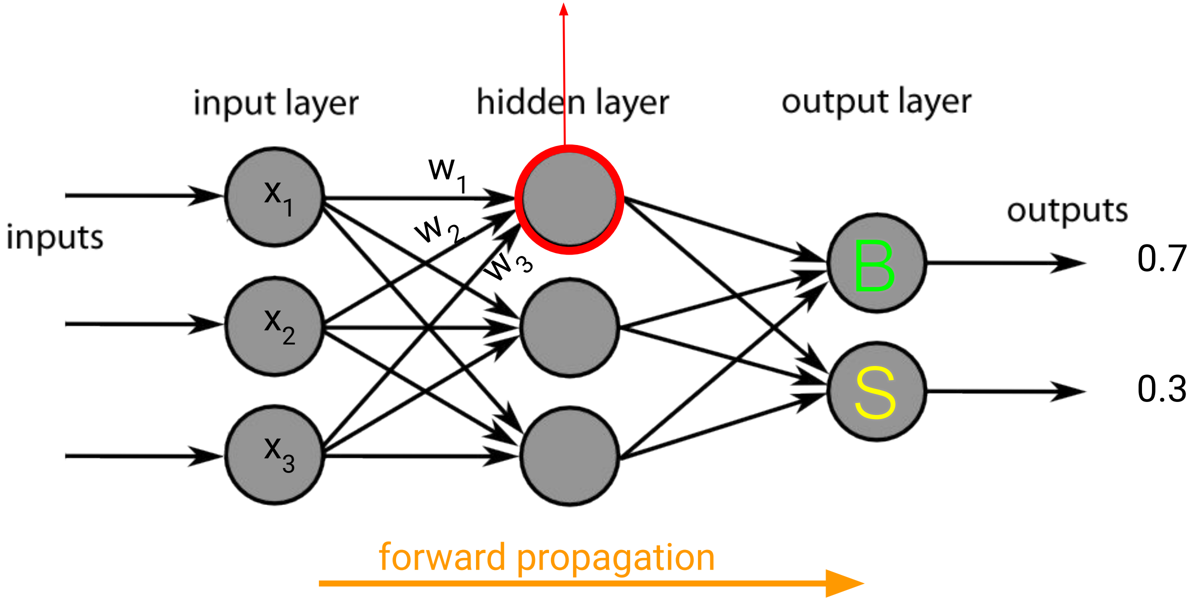
Example

$$(x_1 * w_1 + x_2 * w_2 + x_3 * w_3) + b_1$$ activation function

Back-ground

or

Signal



actual output
0
1

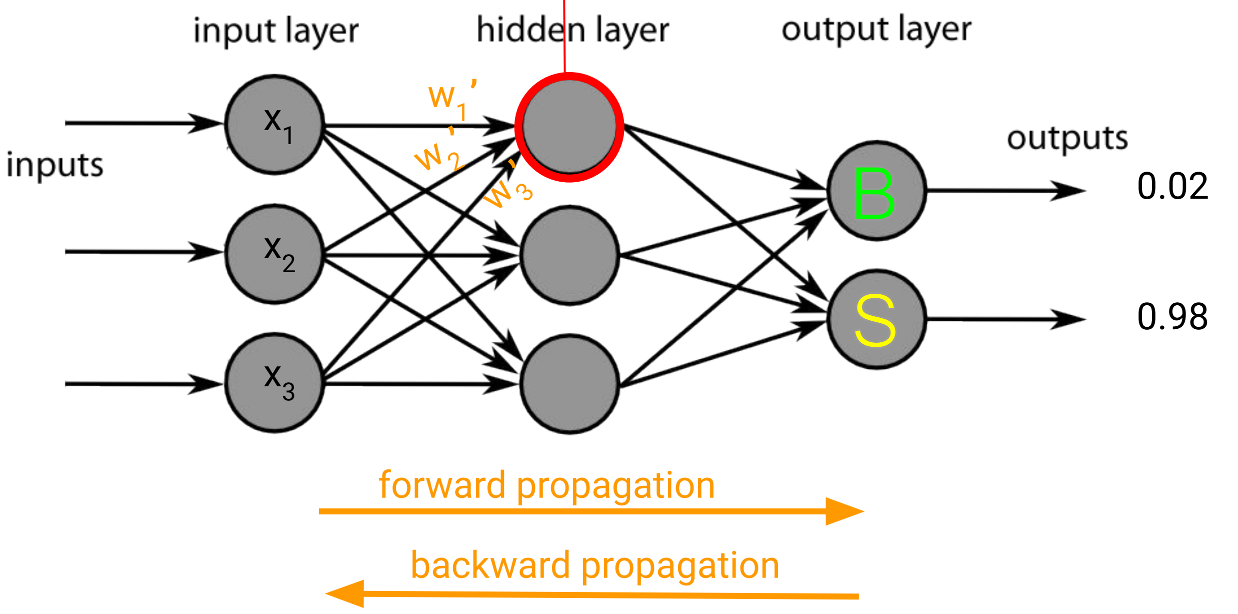
Example

$$(x_1 * w_1' + x_2 * w_2' + x_3 * w_3') + b_1 \Rightarrow \text{activation function}$$

Back-ground

or

Signal



actual output
0
1

Running Deep Neural Networks on FPGAs

Although extremely powerful, **FPGAs have limited resources**: before being implemented into FPGAs, *neural networks have to be suitably optimized*. Optimization is usually organized in two steps:

COMPRESSION/REDUCTION

reduce the DNN size ”as much as possible”, reducing the number of neurons and synapses

- Very first contribution to resource optimization
- Little or no dependence on the FPGA model

TUNING

optimize the DNN implementation for the available FPGA resources, acting upon precision of parameters and thread parallelization

- Procedure strongly dependent on the FPGA model
- Little or no dependence on the actual model implemented (differences among model families)

Very active field: see
J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, JINST 13.07 (2018), P07027
S. Francescato et al., *Model compression and simplification pipelines for fast deep neural network inference in FPGAs in HEP*, Eur.Phys.J.C 81 (2021) 11, 969

Running Deep Neural Networks on FPGAs

Although extremely powerful, **FPGAs have limited resources**: before being implemented into FPGAs, *neural networks have to be suitably optimized*. Optimization is usually organized in two steps:

focus of this talk

COMPRESSION/REDUCTION

reduce the DNN size ”as much as possible”, reducing the number of neurons and synapses

- Very first contribution to resource optimization
- Little or no dependence on the FPGA model

TUNING

optimize the DNN implementation for the available FPGA resources, acting upon precision of parameters and thread parallelization

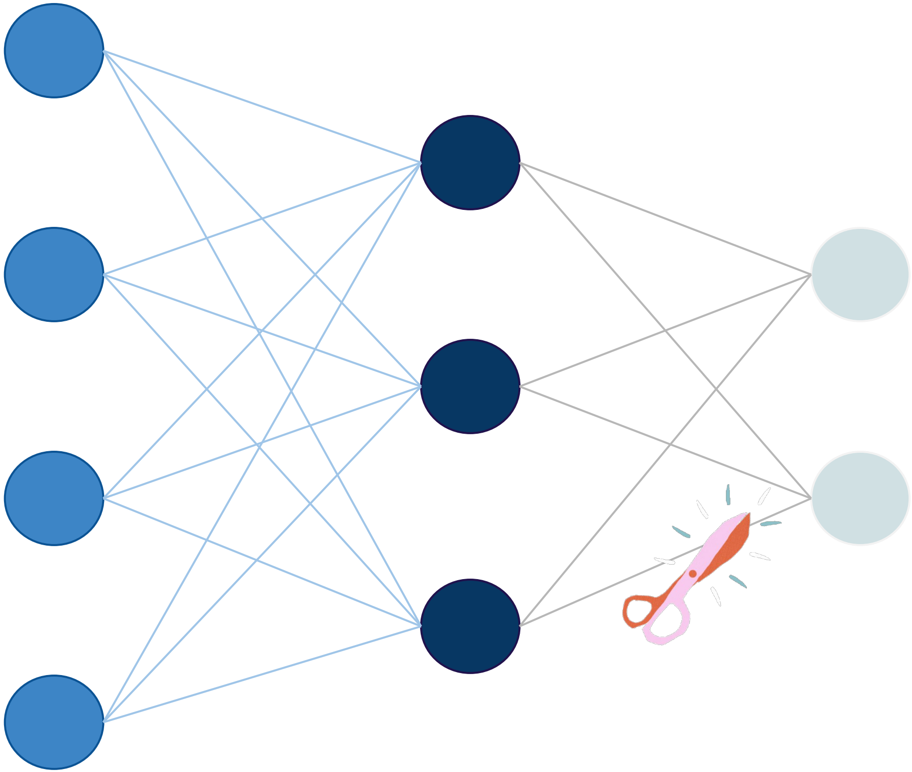
- Procedure strongly dependent on the FPGA model
- Little or no dependence on the actual model implemented (differences among model families)

Very active field: see
J. Duarte et al., *Fast inference of deep neural networks in FPGAs for particle physics*, JINST 13.07 (2018), P07027
S. Francescato et al., *Model compression and simplification pipelines for fast deep neural network inference in FPGAs in HEP*, Eur.Phys.J.C 81 (2021) 11, 969

Pruning

One way of **reducing** the size of a neural network is **pruning**.

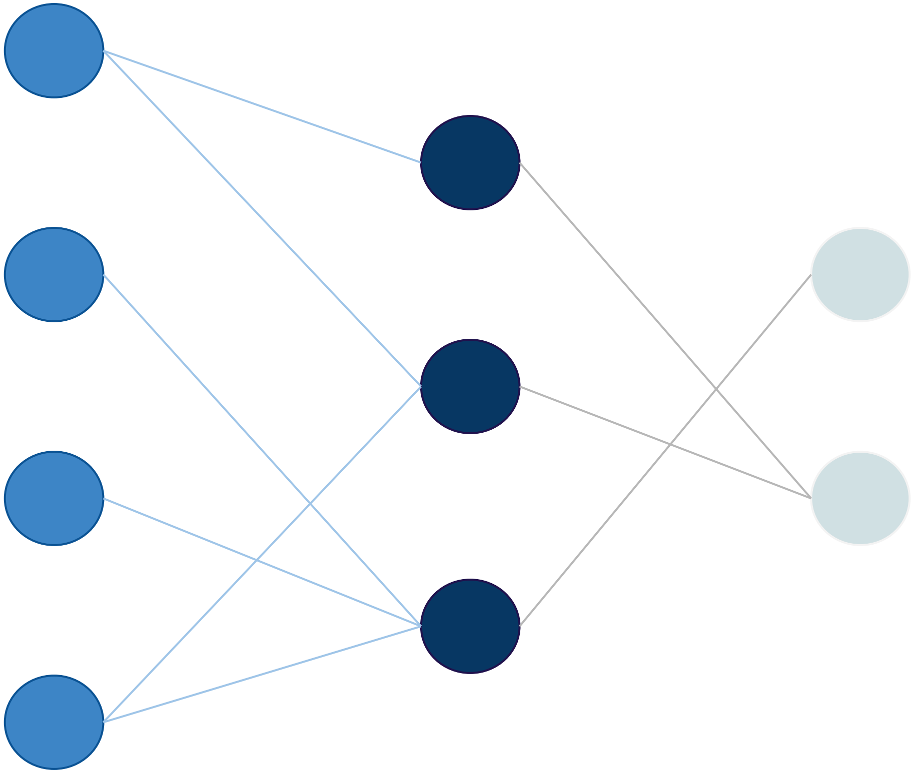
Pruning = **removing** superfluous structure



Pruning

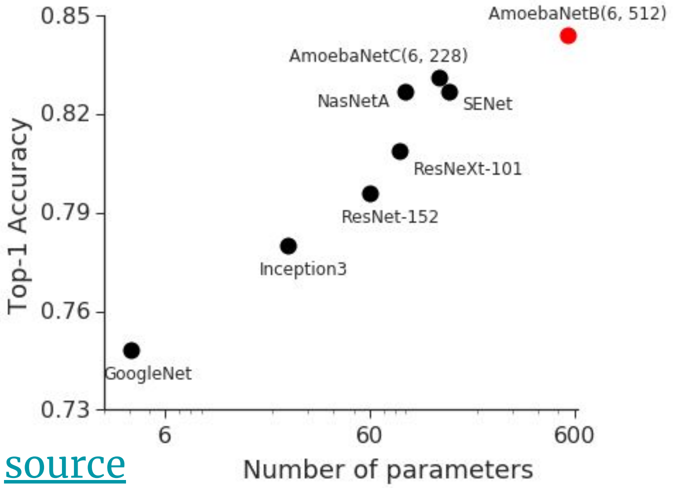
One way of **reducing** the size of a neural network is **pruning**.

Pruning = **removing** superfluous structure



Why pruning?

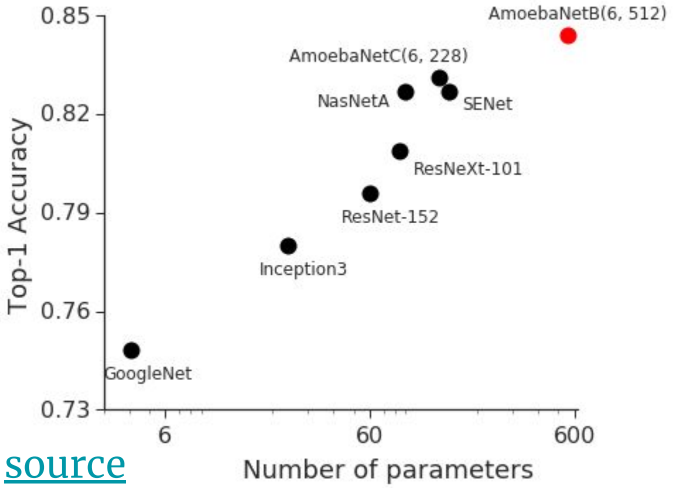
Bigger networks are usually more **accurate**



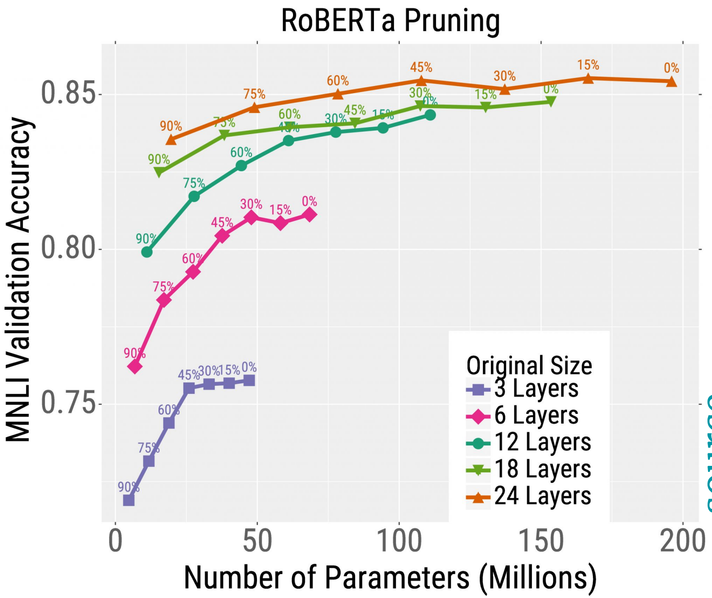
source

Why pruning?

Bigger networks are usually more **accurate**



source

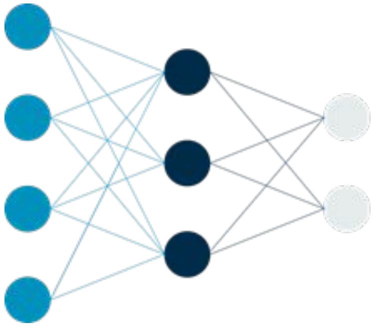


source

➔ Best to start out with very large models and prune with **minimal** performance penalty

Usual pruning scheme

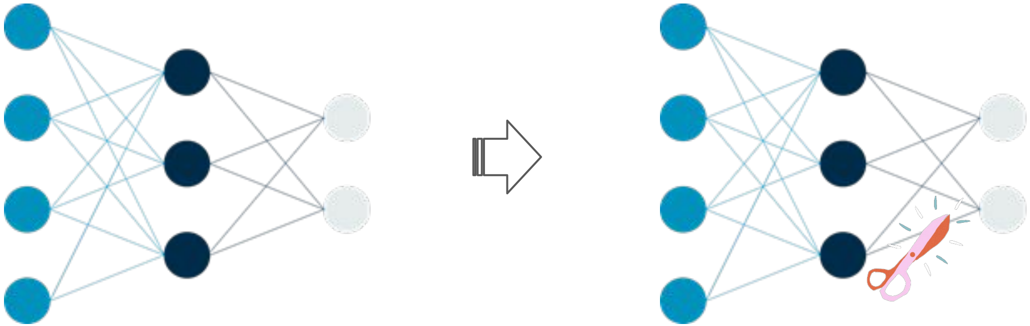
1. Train



Davis Blalock et al., *What is the state of neural network pruning?*, Proceedings of machine learning and systems 2 (2020), pp. 129–146

Usual pruning scheme

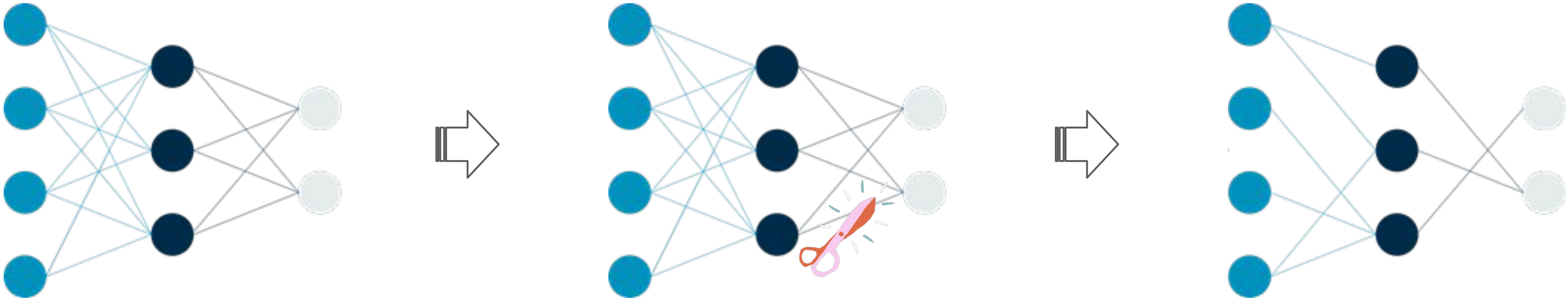
1. Train
2. Prune weights



Davis Blalock et al., *What is the state of neural network pruning?*, Proceedings of machine learning and systems 2 (2020), pp. 129–146

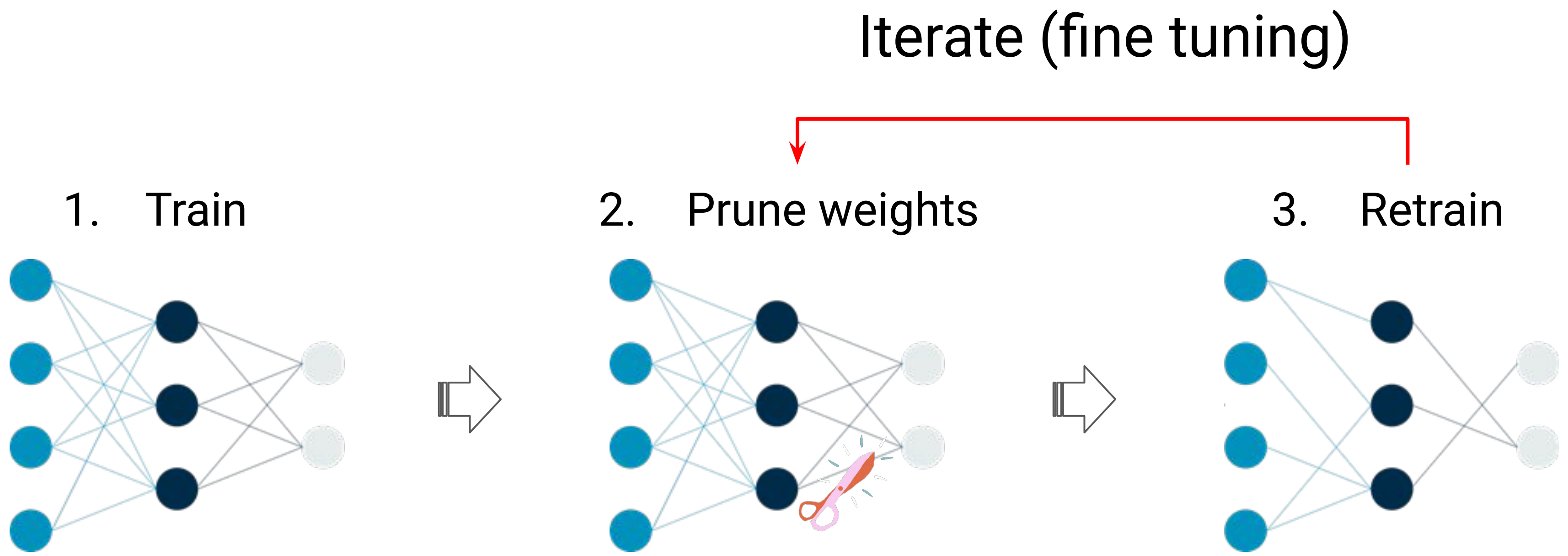
Usual pruning scheme

1. Train
2. Prune weights
3. Retrain



Davis Blalock et al., *What is the state of neural network pruning?*, Proceedings of machine learning and systems 2 (2020), pp. 129–146

Usual pruning scheme

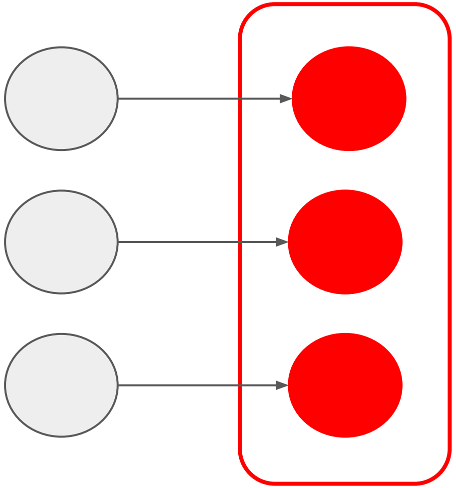


Davis Blalock et al., *What is the state of neural network pruning?*, Proceedings of machine learning and systems 2 (2020), pp. 129–146

A different pruning strategy

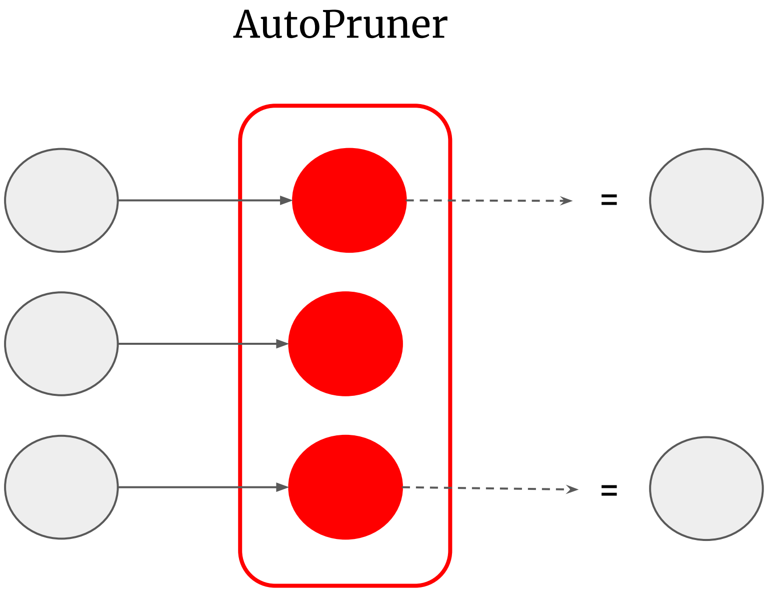
- it can prune **nodes**
- it prunes **during training**
- the number of nodes to be pruned can be determined by the **user**
- it can determine the most suitable **network architecture**

AutoPruner



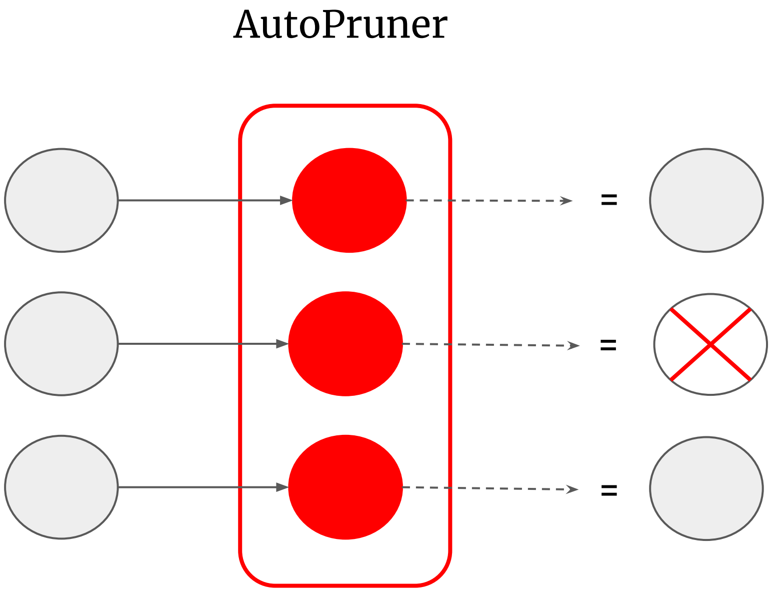
A different pruning strategy

- it can prune **nodes**
- it prunes **during training**
- the number of nodes to be pruned can be determined by the **user**
- it can determine the most suitable **network architecture**



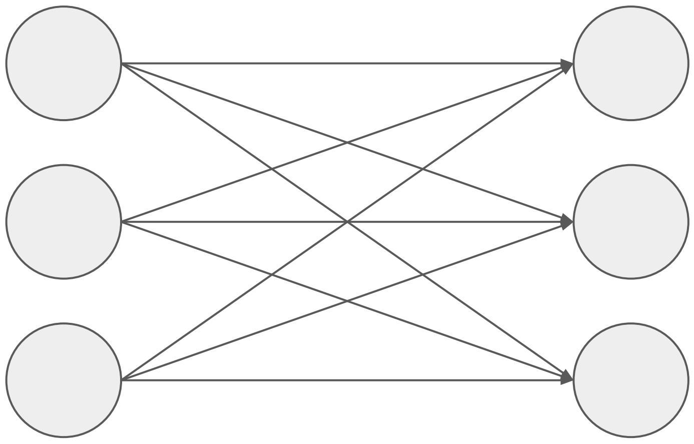
A different pruning strategy

- it can prune **nodes**
- it prunes **during training**
- the number of nodes to be pruned can be determined by the **user**
- it can determine the most suitable **network architecture**



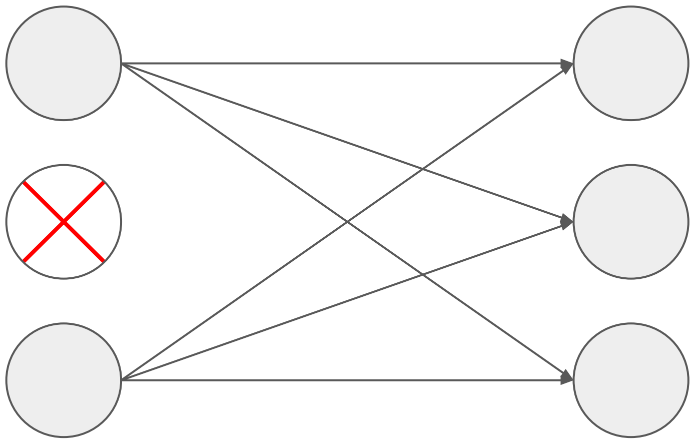
A different pruning strategy

- it can prune **nodes**
- it prunes **during training**
- the number of nodes to be pruned can be determined by the **user**
- it can determine the most suitable **network architecture**



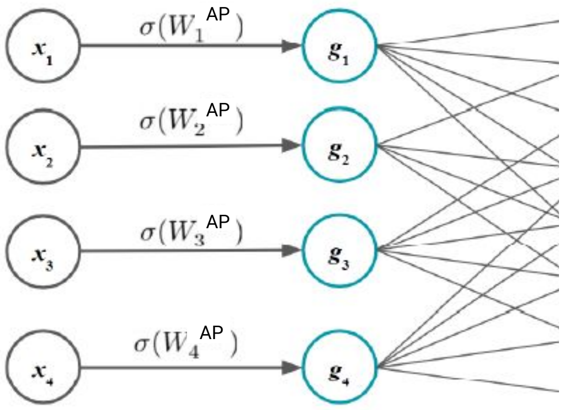
A different pruning strategy

- it can prune **nodes**
- it prunes **during training**
- the number of nodes to be pruned can be determined by the **user**
- it can determine the most suitable **network architecture**

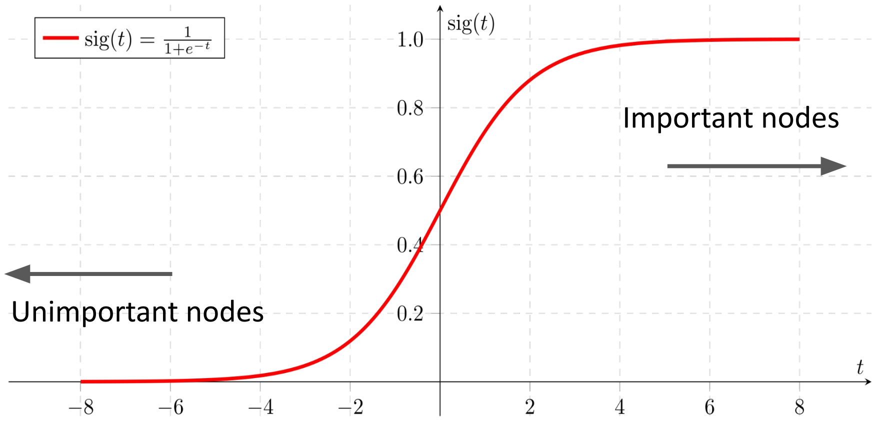


AutoPruner

AutoPruner layers contribute to the training process: along epochs, training is not optimized only for learning, but also to make the neural network containing the exact number of nodes required.

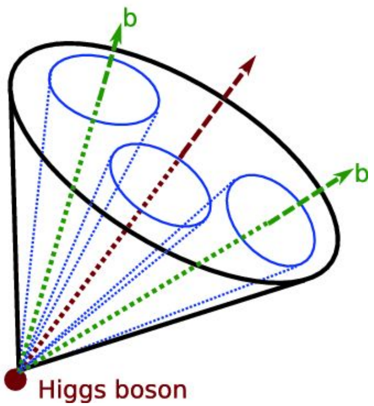


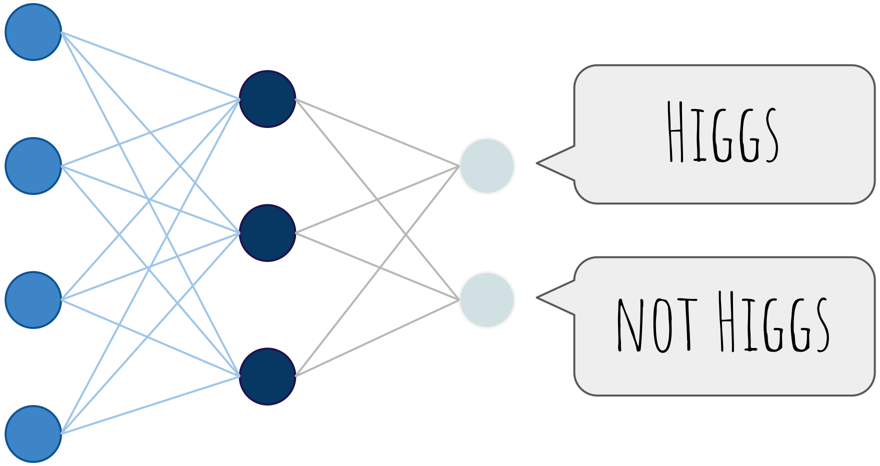
$$g_i = x_i \cdot \sigma(W_i^{AP})$$



Use case

Identify jets that contain both the *b* quarks from boosted Higgs decay in *pp* collision experiments using Deep Neural Networks





Bench-test dataset

A fast and reliable framework to make pseudo-experiments has been developed for tests.

DELPHES
fast simulation

PYTHIA8
generation of high-energy physics events

DELPHES
detector response fast simulation

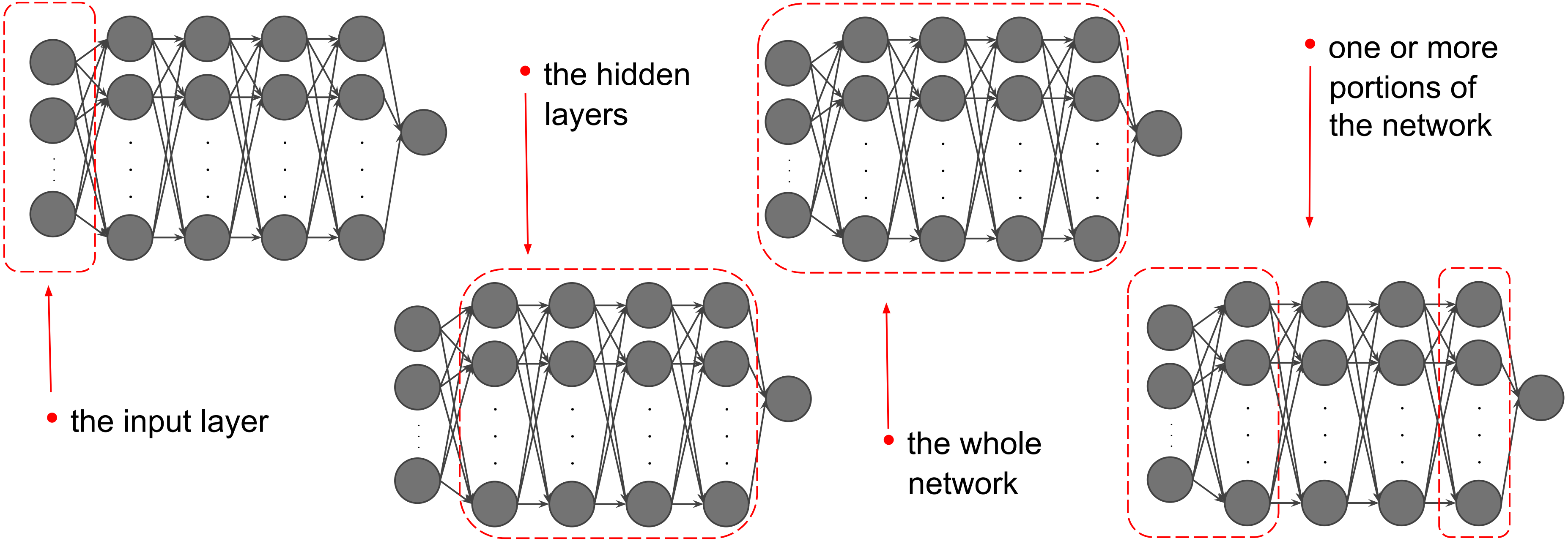
RAVE
secondary vertex reconstruction

4×10^6 simulated events of *pp*-collision at 14 TeV with ATLAS-like detector geometry

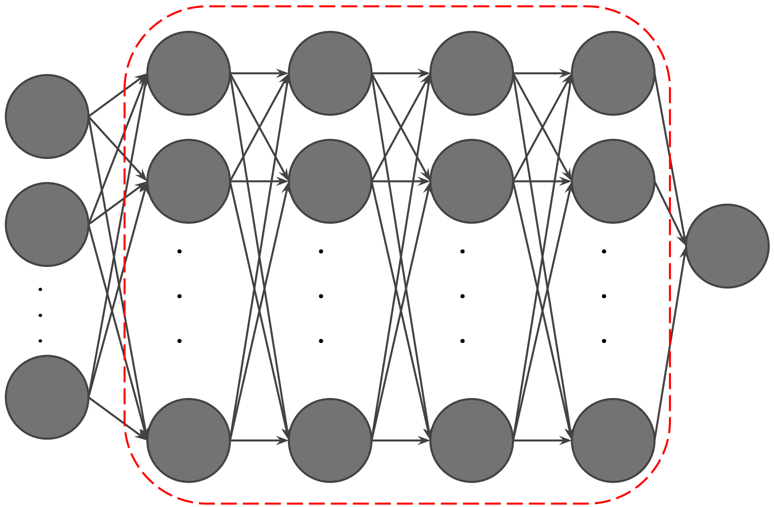
Signal: g+b → H+b Background: QCD

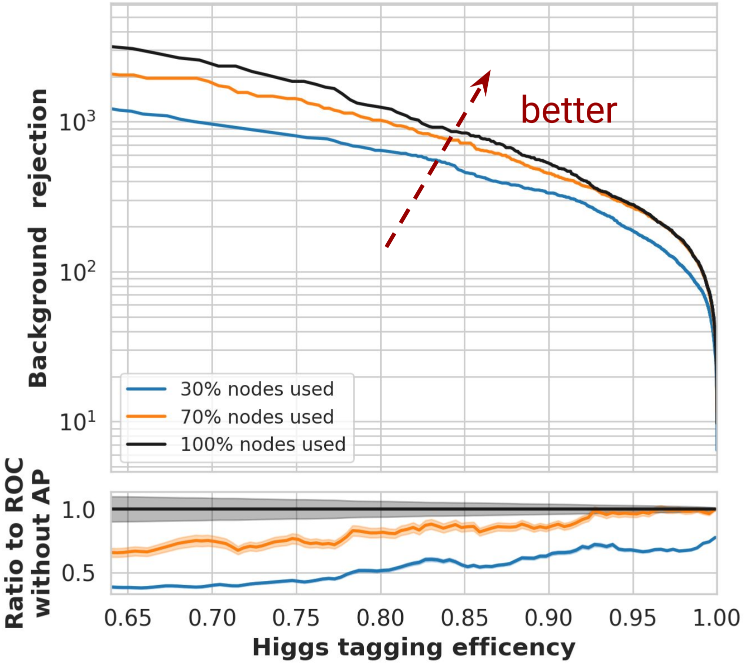
Pruning with AutoPruner

With AutoPruner you can **choose** which part of the network you want to prune



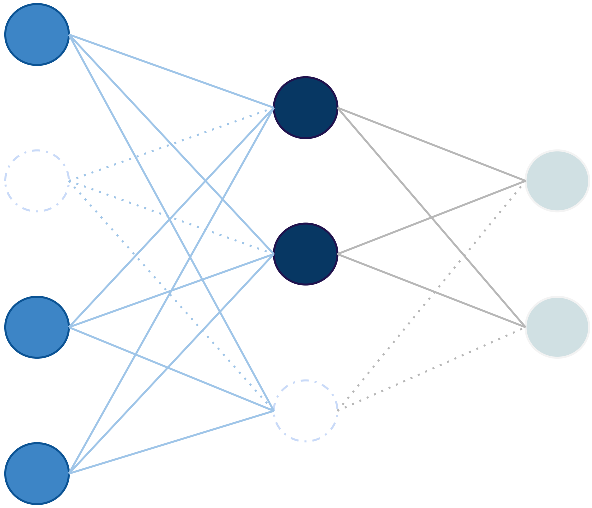
Performance evaluation



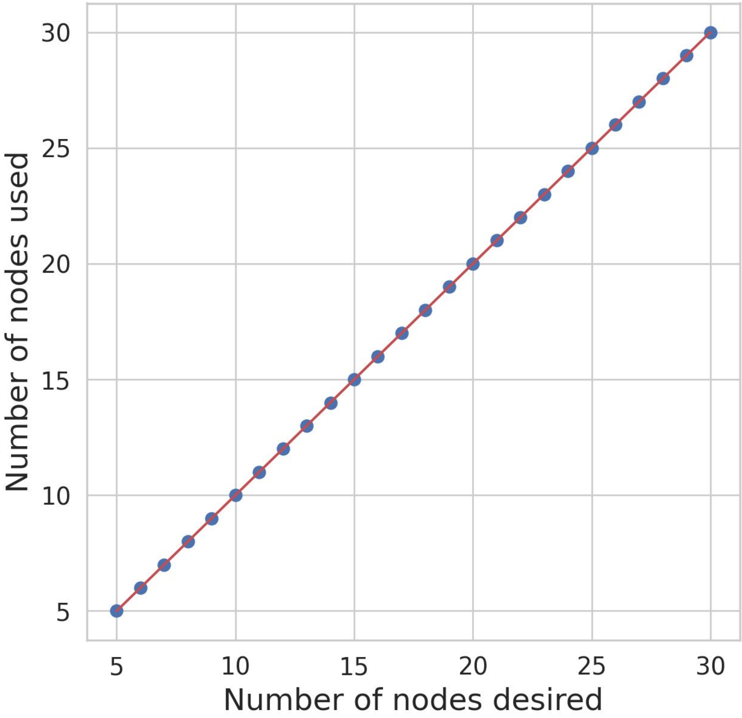


The performance increases with the percentage of nodes used, as expected: AutoPruner is really **switching off** nodes

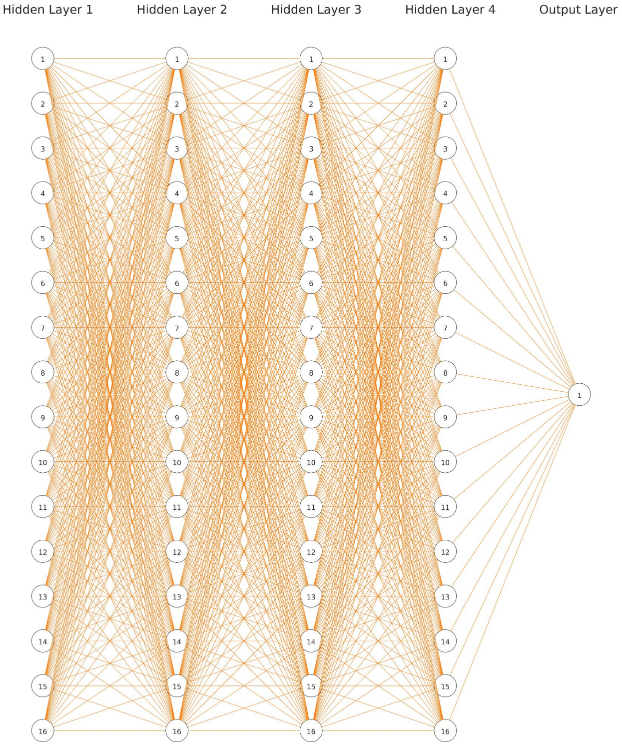
Nodes used



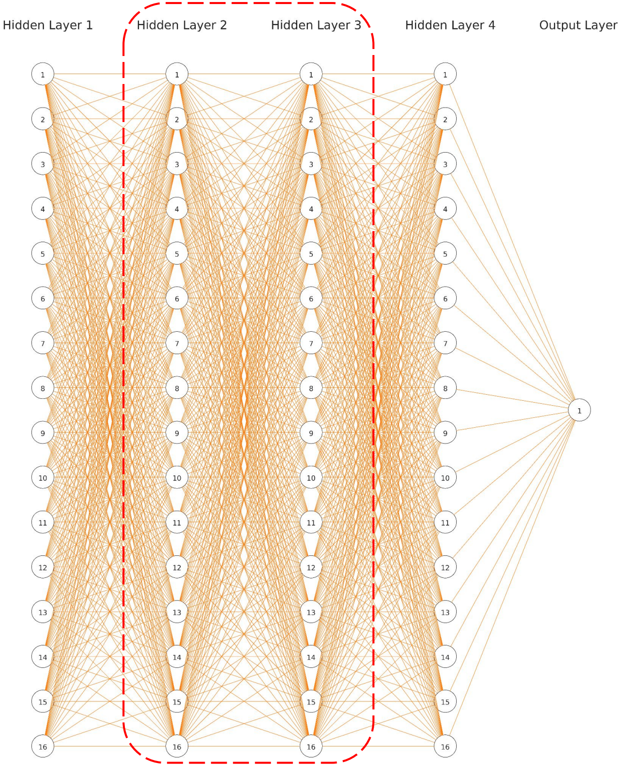
The total number of nodes used is
always equal to the required number



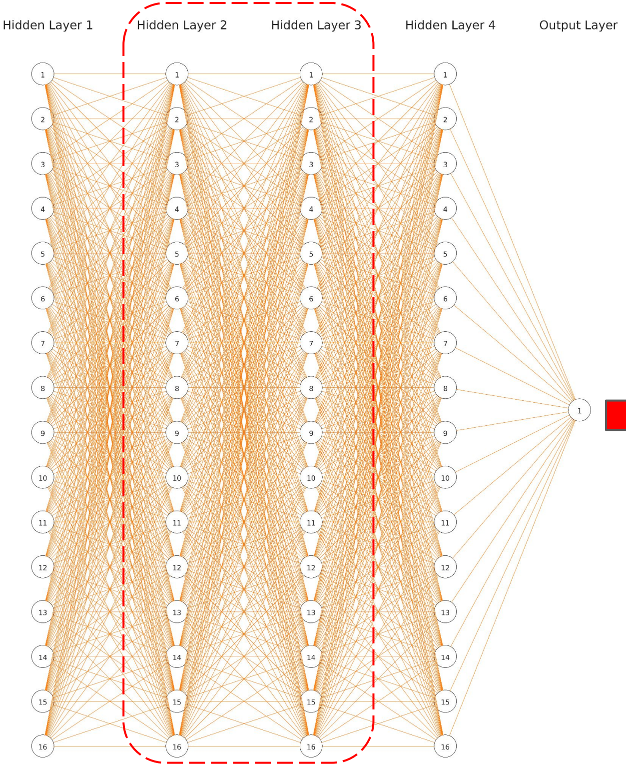
Models' comparison

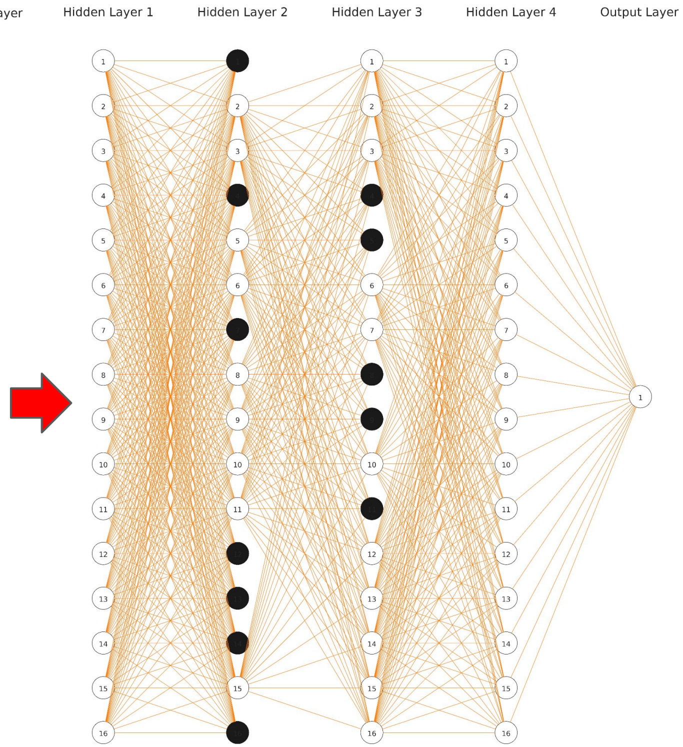


Models’ comparison

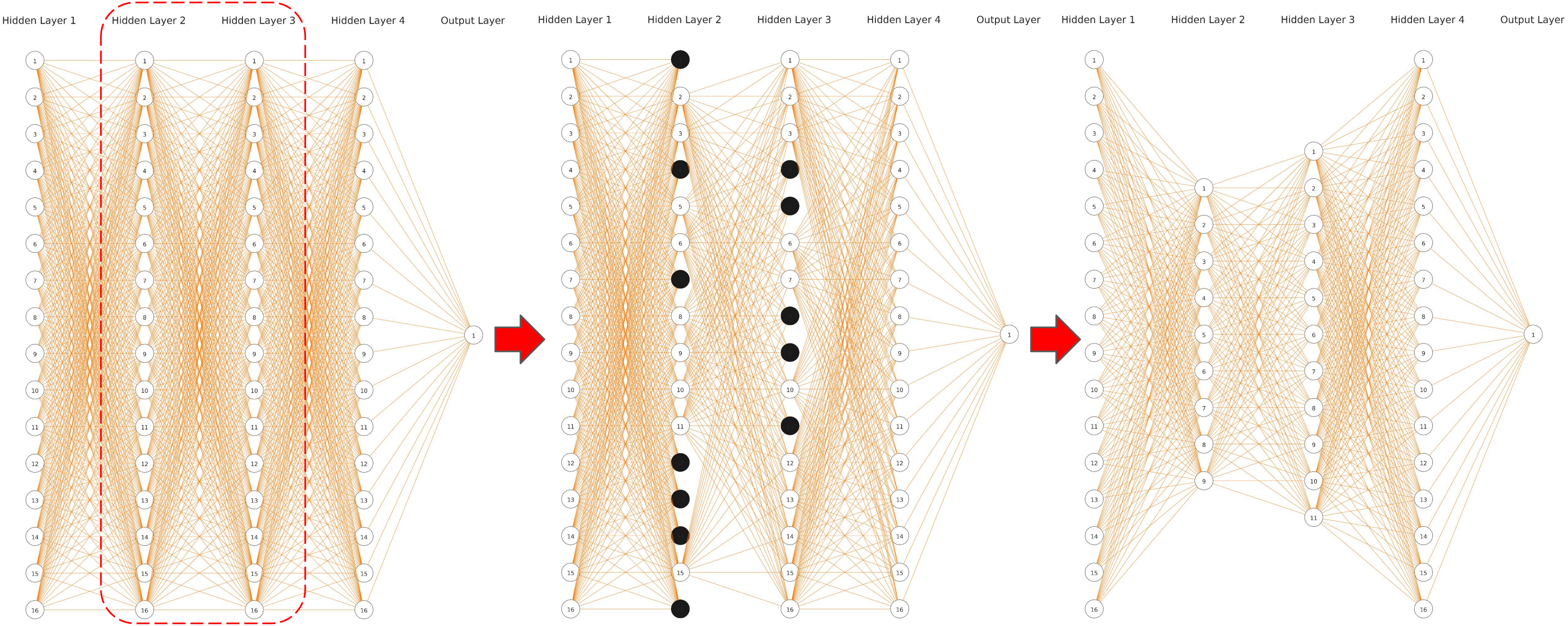


Models' comparison

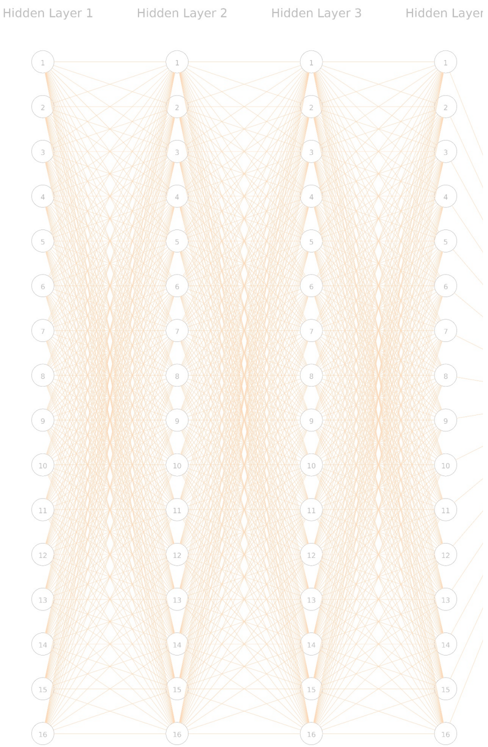


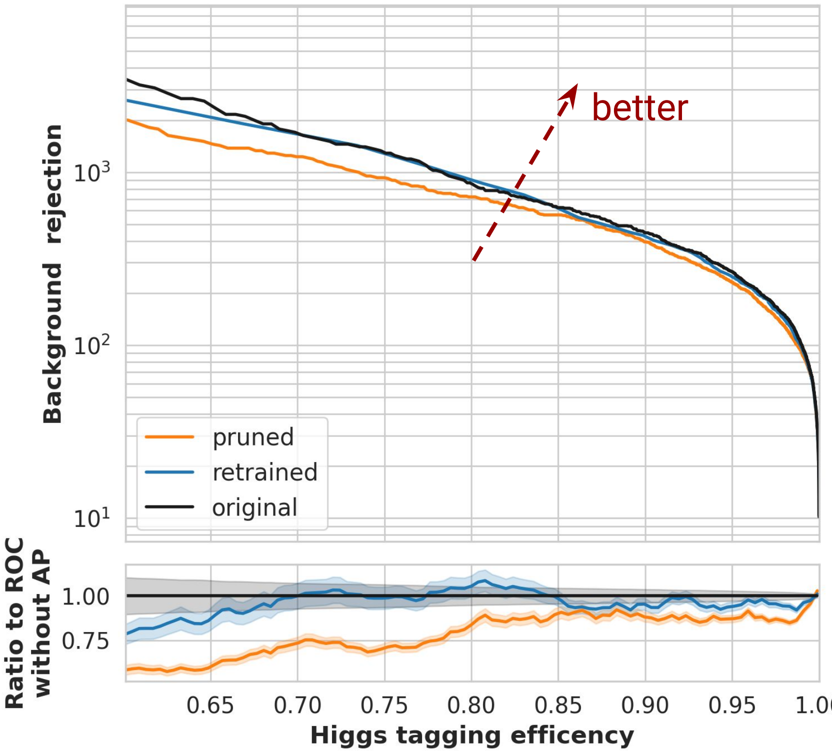


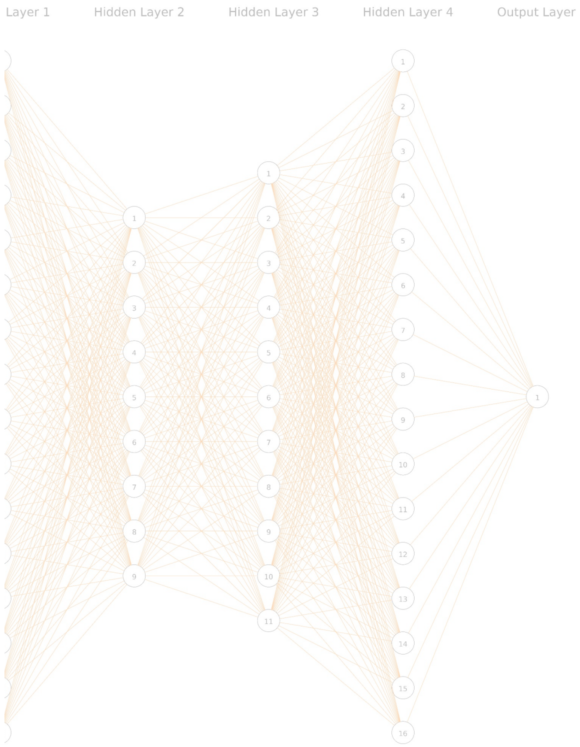
Models’ comparison



Models’ comparison

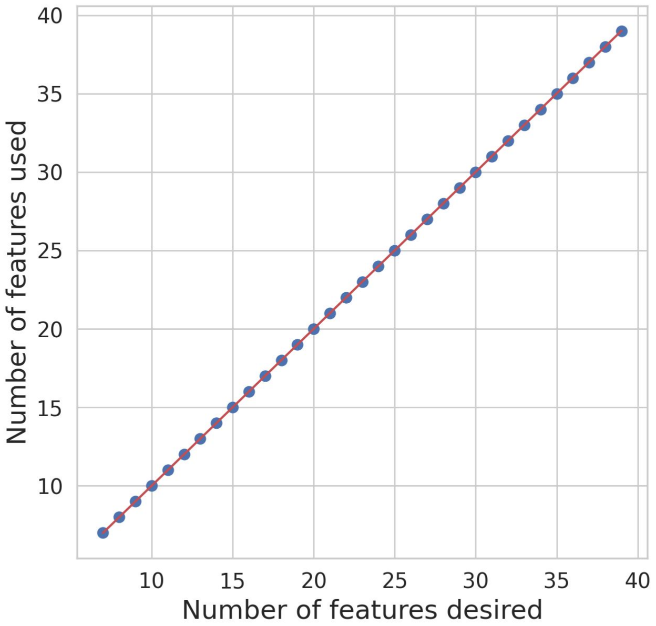




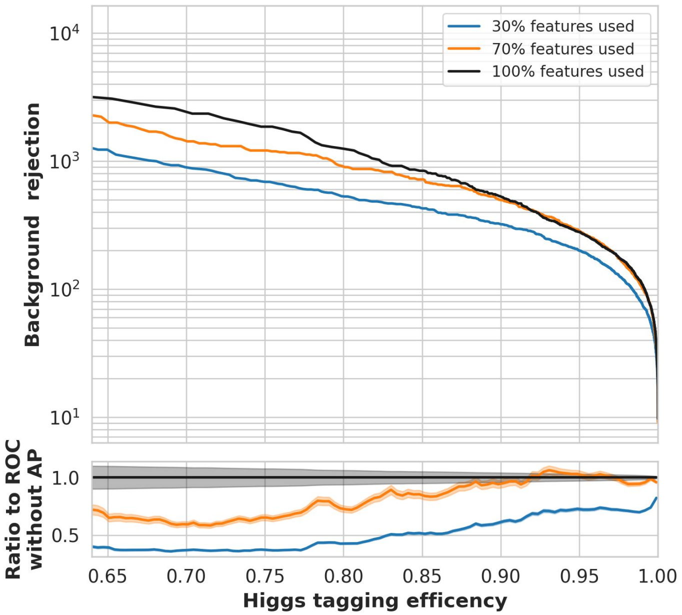


Feature selection

the number of features used
is equal to the requirement

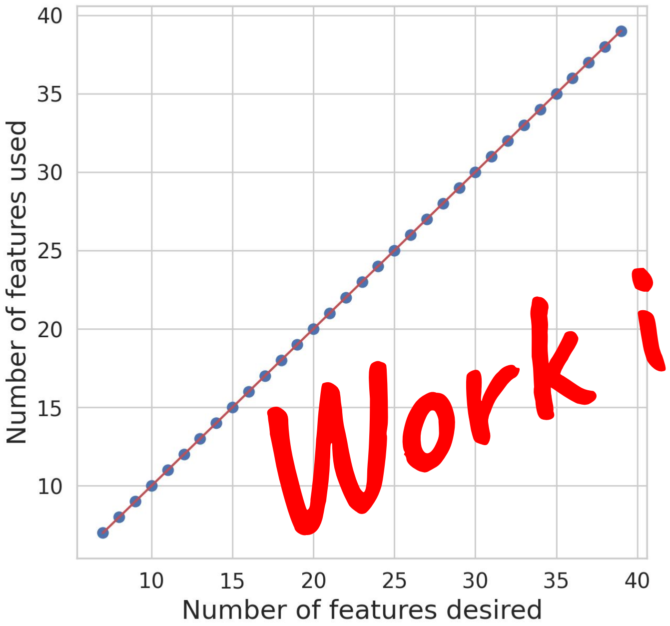


the performance worsens as expected as long
as the number of used features diminishes

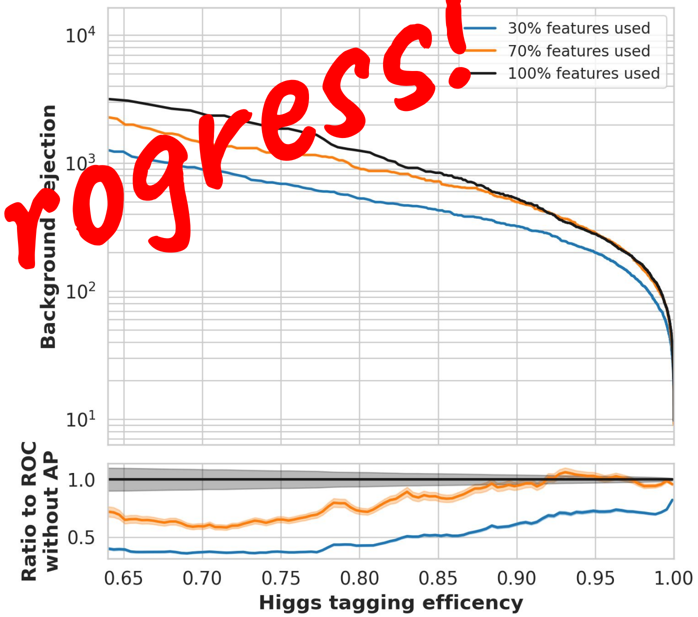


Feature selection

the number of features used is equal to the requirement



the performance worsens as expected as long as the number of used features diminishes



Work in progress!

Conclusions

The **problem** of effectively and optimally prune/tune Deep Neural Networks is ubiquitous in experiments at future colliders.

We introduced the AutoPruner approach to **effectively prune** Deep Neural Networks during training.

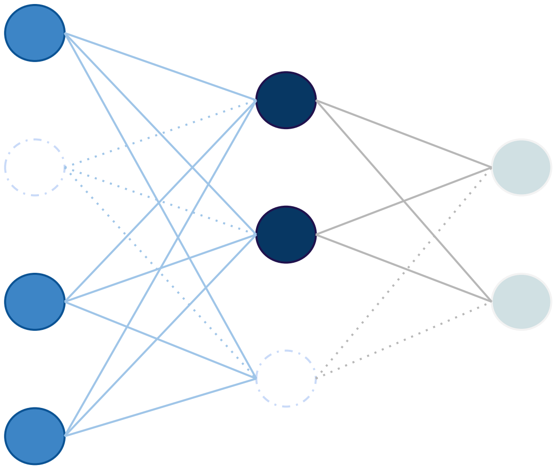
We applied the derived tool to a simulated dataset that we constructed on purpose.

AutoPruner proved to be:

- **simple** to incorporate
- **effective and successful** in reducing the networks' size
- very **understandable**

Further developments are focusing on:

- quantify stability against initial conditions
- characterize optimality



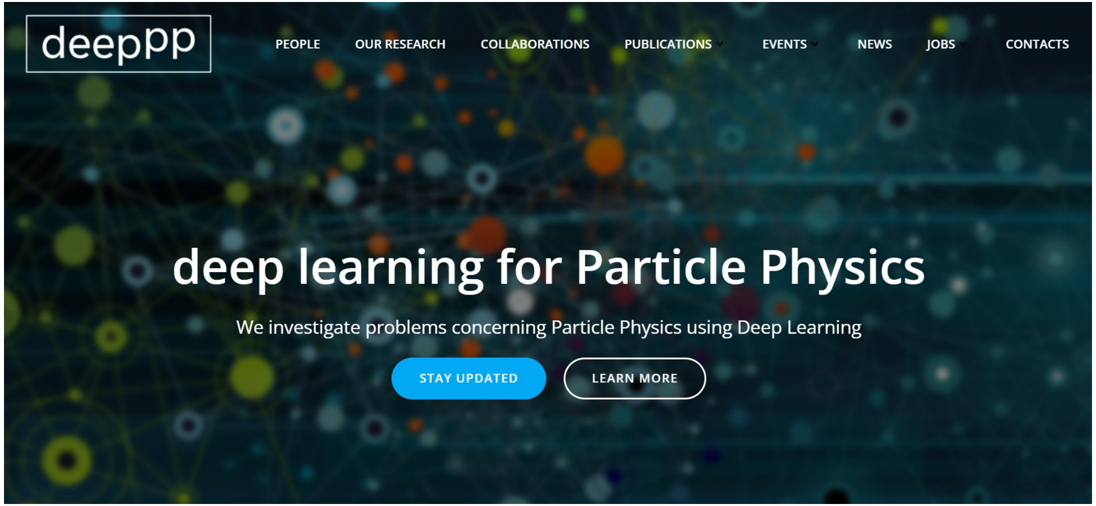
Thanks!

Want to know more about Deep Learning applications in Particle Physics?

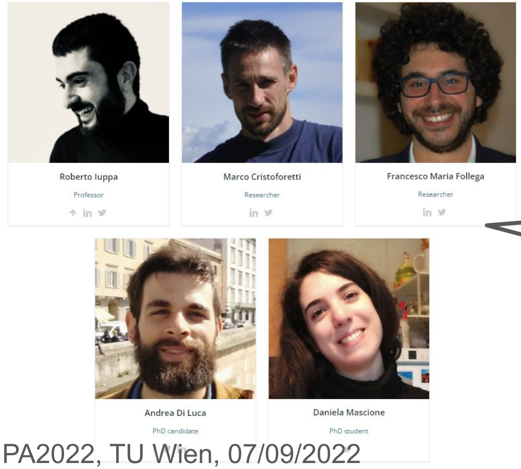
Awesome!

Visit
https://www.deeppp.eu/

deepPP



Our work



Us

Who we work with

TIFPA
Trento Institute for Fundamental Physics and Applications

ATLAS EXPERIMENT
ATLAS Flavour Tagging Working Group