for details, just google:

**deepPP**

# Optimal size constraining of Deep Neural Network Models for FPGA implementation in trigger systems of experiments at future colliders


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

University of Trento, INFN TIFPA, FBK



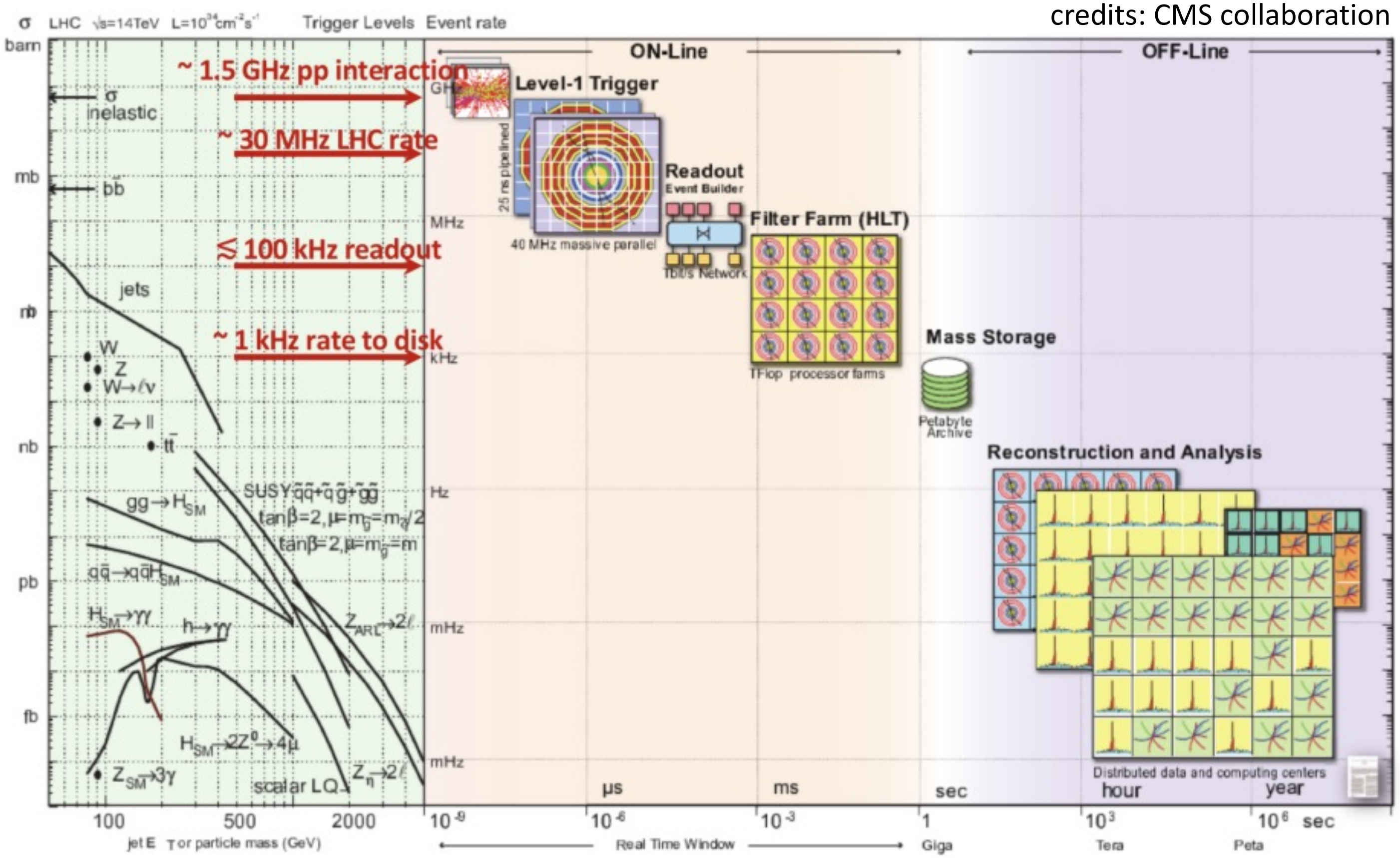
ICHEP 2022 BOLOGNA

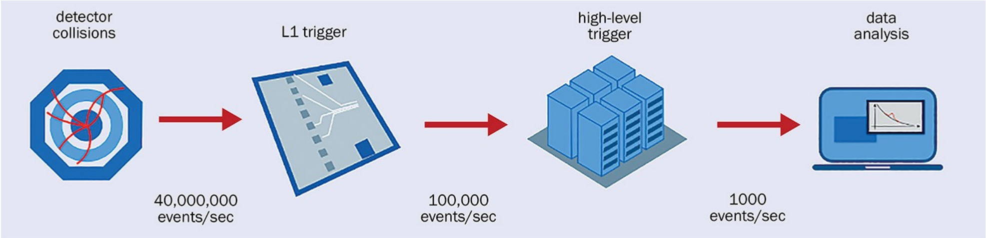
ICHEP 2022 XLI International Conference on High Energy Physics Bologna (Italy)

6 13 07 2022

# Increased collision rates at future colliders

credits: CMS collaboration




credit: M. Pierini

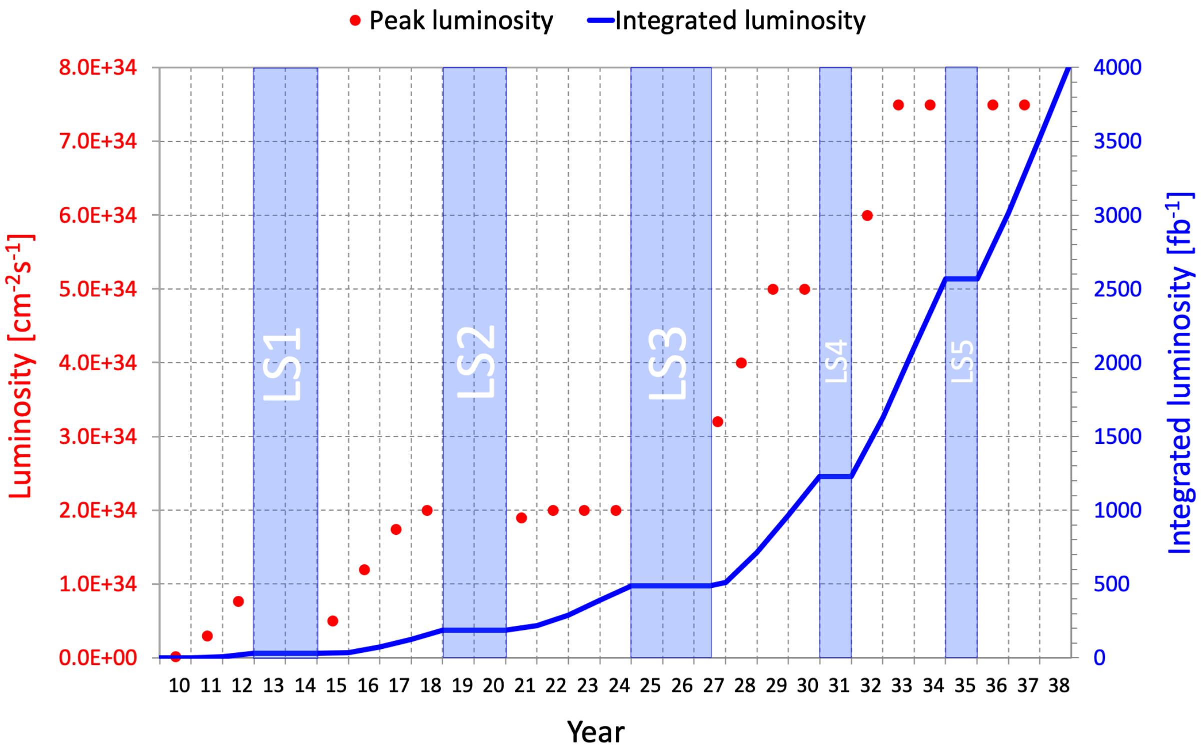
Roberto Iuppa

Istituto Nazionale di Fisica Nucleare

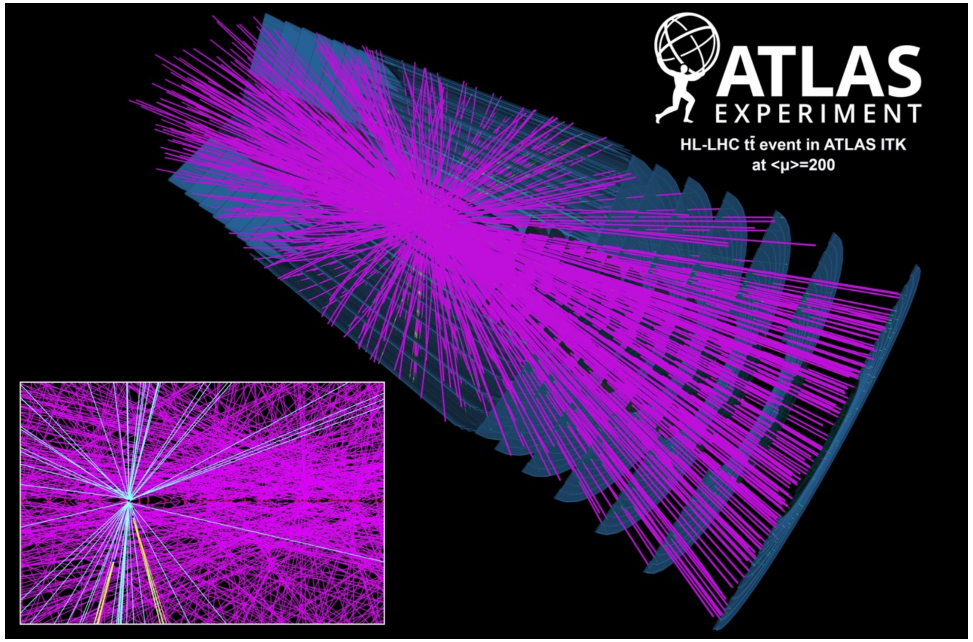
ICHEP 2022 BOLOGNA

July 8

# Increased collision rates at future colliders




https://lhc-commissioning.web.cern.ch/schedule/images/LHC-ultimate-lumi-projection.png




credits: ATLAS collaboration

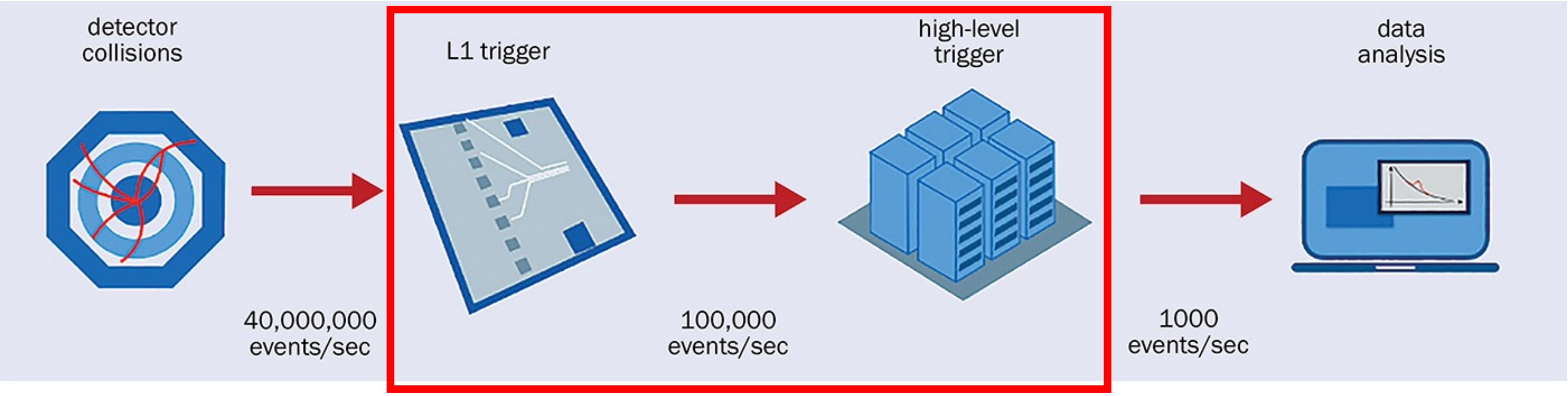
Also for **FCC-ee and ILC**, triggerless approaches are explored where the event selection are largely committed to Machine Learning models directly interfaced with detector's front-end readout.

At the **FCC-hh** huge amounts of data will be produced (O(TBytes/s) expected), Machine Learning or Artificial Intelligence algorithms will play an important role 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.


Roberto Iuppa

# Machine learning models on FPGAs

Complex event selection in “real-time”:

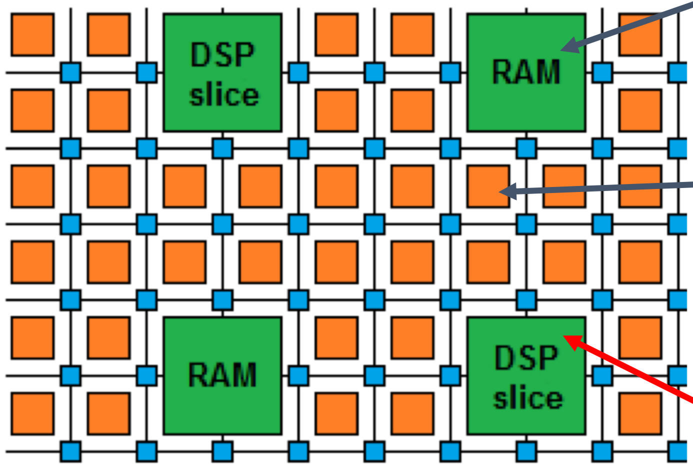



credit: M. Pierini

“real-time” means no later than a few $\mu$s after the collision

FPGAs (Field-Programmable Gate Arrays) are programmable integrated circuits. They can offer low latency and high throughput. Although extremely powerful, FPGAs have limited resources.

How to optimize the software (ML models) for the hardware at disposal (FPGA)?




**RAMs** for fast-access memory

**Logic cells** for any function and simple arithmetic.

**DSPs** (Digital Signal Processors) are designed to perform multiplications.


Istituto Nazionale di Fisica Nucleare

Roberto Iuppa

# Neural networks on FPGA: resource optimization

Very active field of technology development, driven by AI software upgrades for smartphones.
Optimization is usually organized in two steps:

*FOCUS OF THIS TALK*

## COMPRESSION/REDUCTION

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

- Very first contribution to resource optimization
- Iterative ”pruning” is the common approach
- Little or no dependence on the FPGA model

## TUNING

optimize the NN 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)

How often will you have to update your model on your hardware? How fast will you be to optimize it?
Once FE/DAQ boards for trigger are installed, no major upgrades are possible for long periods of data-taking: need for a fast and robust way to update NN models on FPGAs, being sure the implementation optimally exploits the available resources.

Istituto Nazionale di Fisica Nucleare

Roberto Iuppa

# Pruning

The 'Iterative Pruning' paradigm: Prune / Train / Repeat

*Optimal Brain Damage*

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

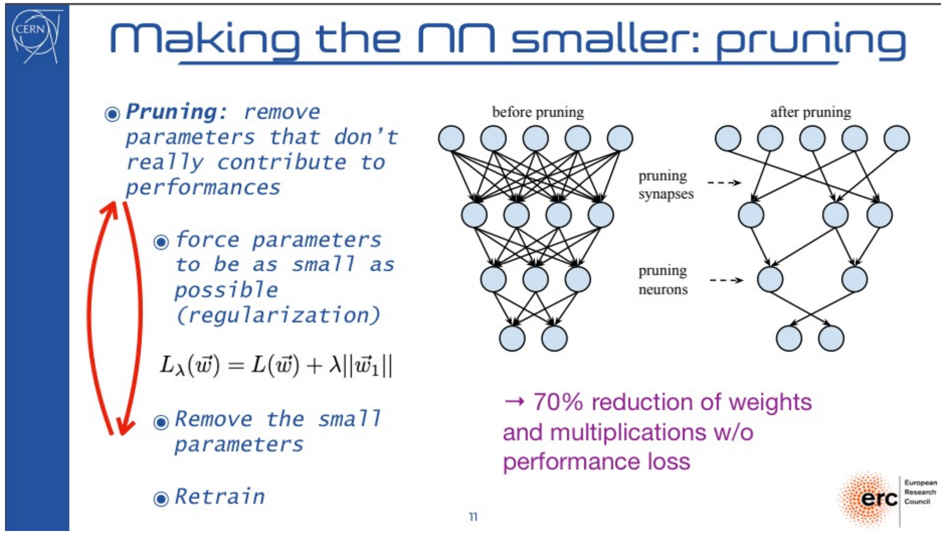
**ABSTRACT**

We have used information-theoretic ideas to derive a class of practical and nearly optimal schemes for adapting the size of a neural network. By removing unimportant weights from a network, several improvements can be expected: better generalization, fewer training examples required, and improved speed of learning and/or classification. The basic idea is to use second-derivative information to make a tradeoff between network complexity and training set error. Experiments confirm the usefulness of the methods on a real-world application.

1. Choose a reasonable network architecture
2. Train the network until a reasonable solution is obtained
3. Compute the second derivatives $h_{kk}$ for each parameter
4. Compute the saliencies for each parameter: $s_k = h_{kk}u_k^2/2$
5. Sort the parameters by saliency and delete some low-saliency parameters
6. Iterate to step 2

Advances in Neural Information Processing Systems 2 (NIPS 1989)

Credit: M. Pierini, AI@INFN workshop, May 2022, Bologna (IT)




But:

- procedure quite long and resource demanding
- relative importance of parameters changes along iterations → risk to converge to sub-optimal configurations
- the "reasonable" architecture to start from is chosen according to (i) developer's experience and (ii) grid search in the hyperparameter space

Roberto Iuppa

ICHEP 2022 BOLOGNA

Is it possible to optimally tune neural networks under constraints of latency and size?

Is iterative pruning the only option?

Does there exist a mathematically robust method to choose the best network architecture among all the (infinite) ones compatible with the FPGA resources available?

We present here an innovative method to approach these problems, efficiently pruning NN nodes.

We refer to the related software tool as **AutoPruner**.

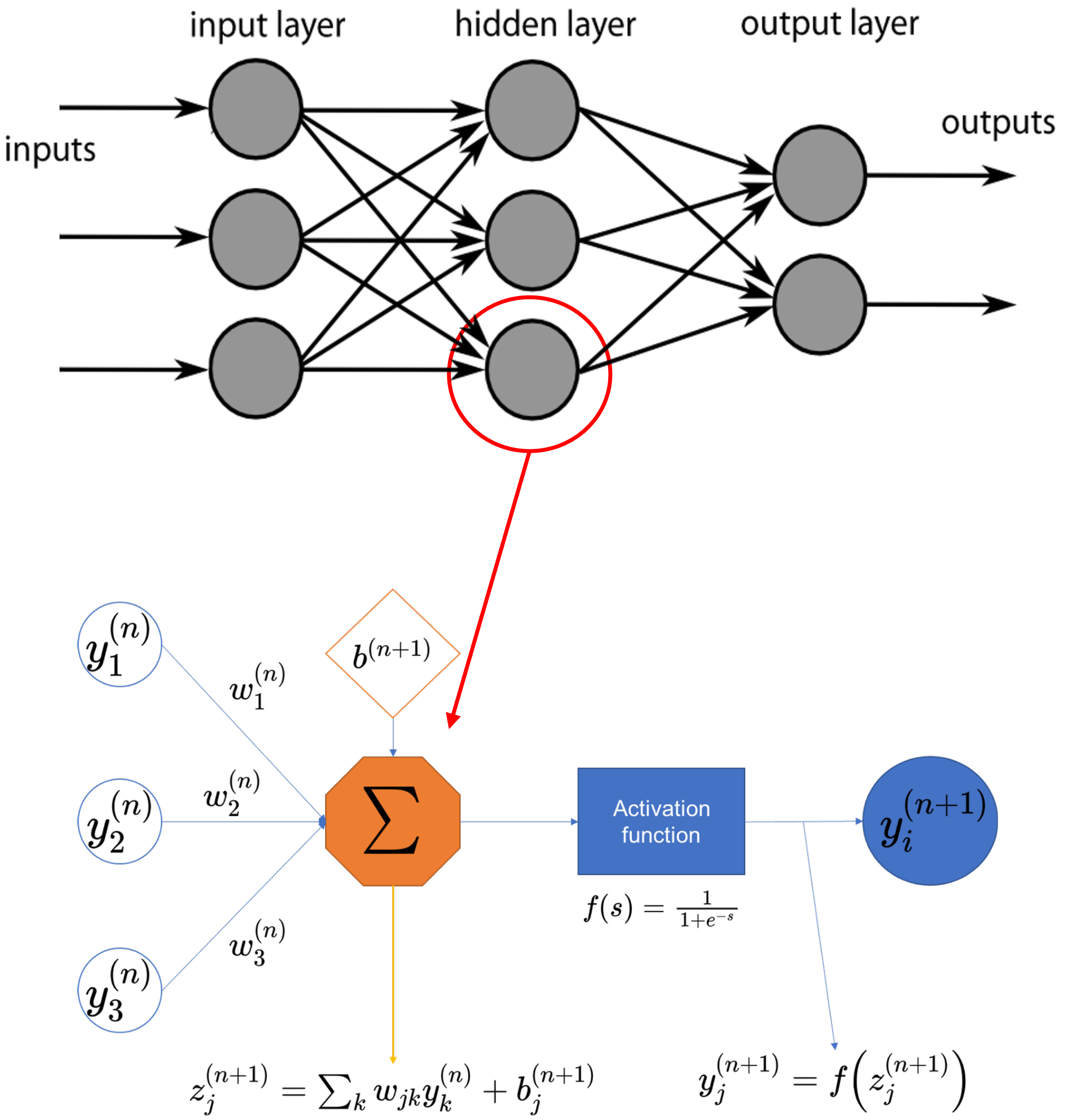
We present here the results of an application of **AutoPruner** to Fully Connected Neural Networks, but conclusions have general validity.

# DNN: quick recall

An Artificial Neural Network is a computational model that has layers of interconnected nodes called artificial neurons.

A Deep Neural Network has more than one hidden layer.

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




Roberto Iuppa

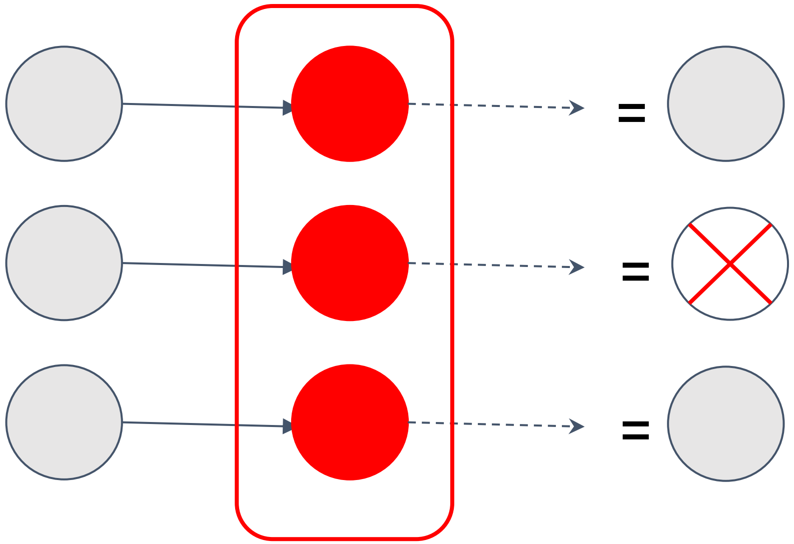
ICHEP 2022 BOLOGNA

July 8

# AutoPruner

In its simplest form, **AutoPruner** is a layer for deep neural networks that acts as a filter for a selected number of nodes.

The main idea behind this layer is to update the nodes' weights **during the training stage** so that only the fraction of nodes that contribute to the learning process will tend to a pre-set value, while the remaining ones are progressively excluded from training.



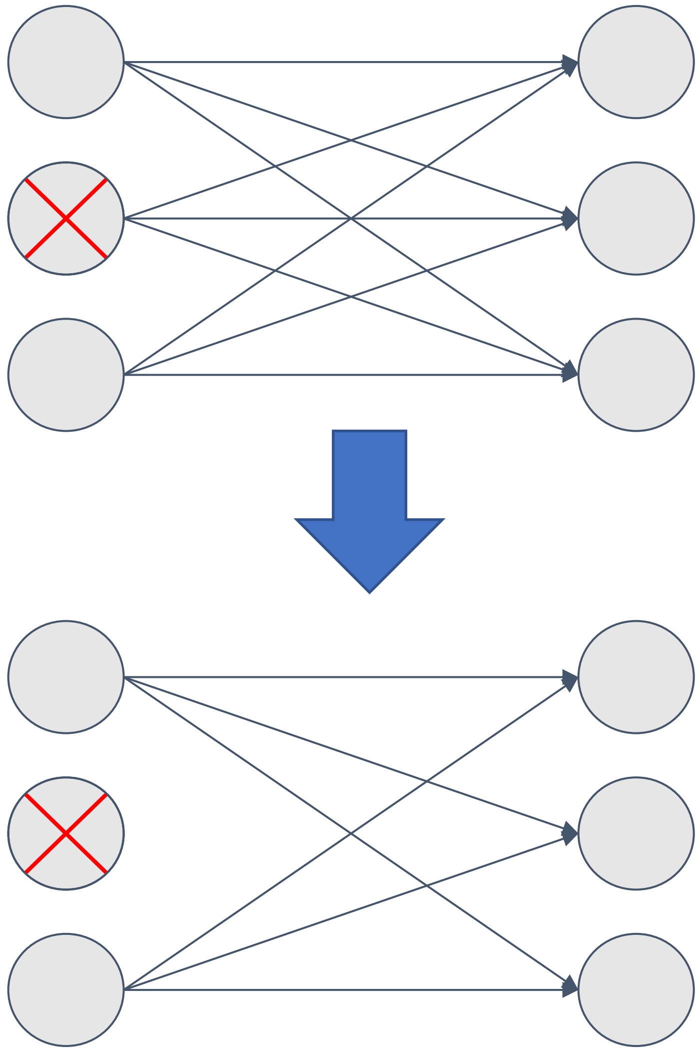

Roberto Iuppa

ICHEP 2022 BOLOGNA

# AutoPruner

In its simplest form, **AutoPruner** is a layer for deep neural networks that acts as a filter for a selected number of nodes.

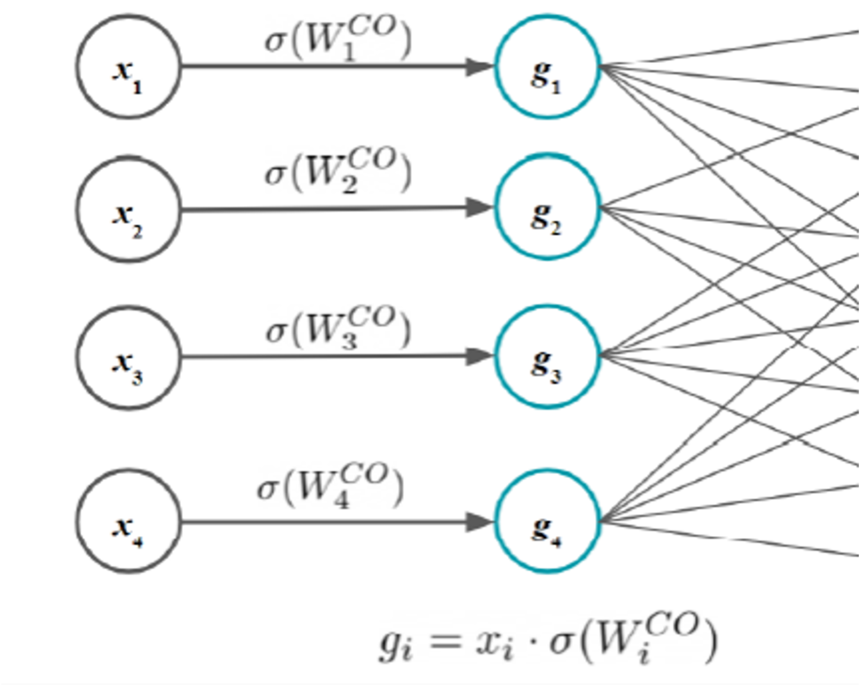
The main idea behind this layer is to update the nodes' weights **during the training stage** so that only the fraction of nodes that contribute to the learning process will tend to a pre-set value, while the remaining ones are progressively excluded from training.

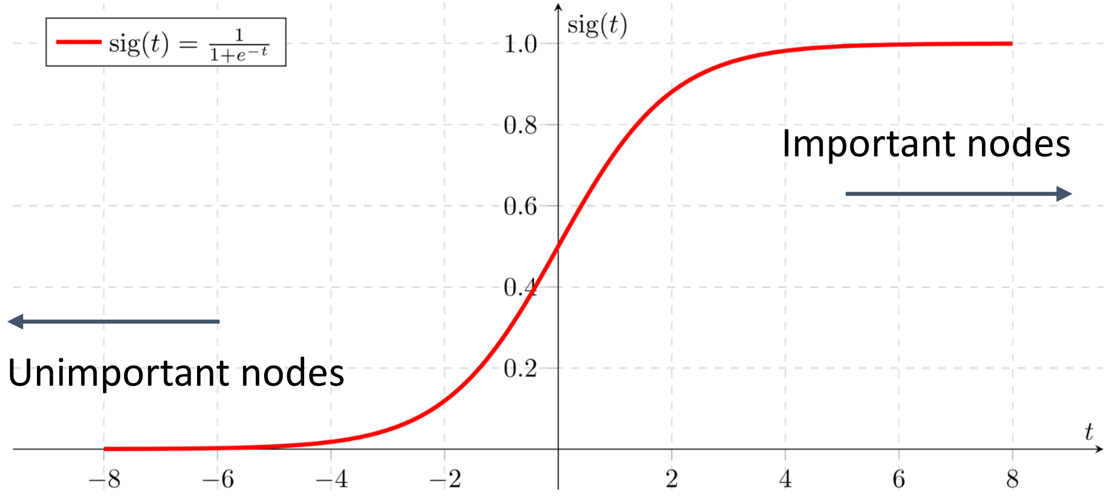


Roberto Iuppa

# AutoPruner

AutoPruner layers contribute to the training process, because the loss includes a term devised on purpose. Along epochs, training is not optimized only for learning, but also to make the NN containing the exact number of nodes required.




$$\mathcal{L}_{TOT} = \mathcal{L}_{classifier}(\vec{x}) + \mathcal{L}_{pruner}(\vec{w})$$

Roberto Iuppa

ICHEP 2022 BOLOGNA

# 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

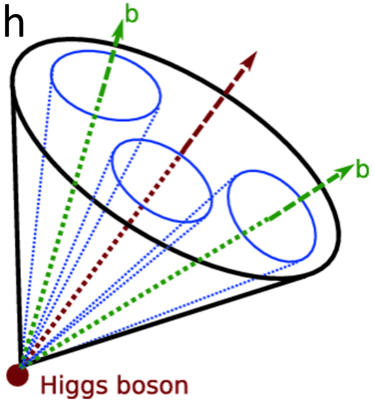
Microsoft Azure

Run on **Azure VM**
Ubuntu 18.04-LTS
Standard NC6_Promo
[**6 vcpus**, **56 GiB memory, 1 GPU**]

$4 \times 10^6$ simulated events of pp-collision at 14 TeV with ATLAS-like detector geometry

Signal: g+b →H+b
Background: QCD

FCNN trained to tag boosted Higgs decays




Reconstructed objects:

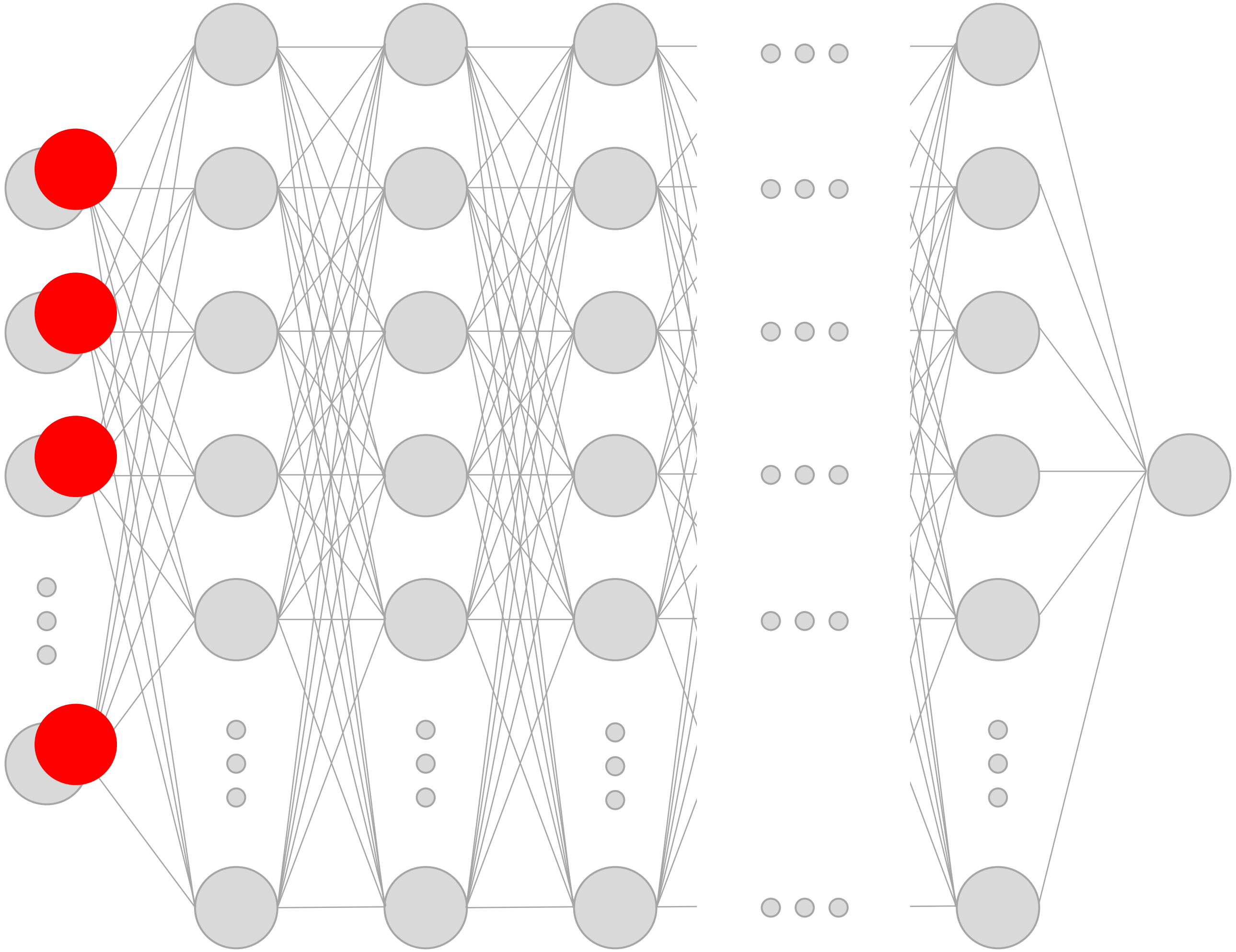
- Large radius jet (Large R jet)
- Variable radius track jets

39 input features reconstructed with same algorithms used in ATLAS

INFN
Istituto Nazionale di Fisica Nucleare

Roberto Iuppa

# Pruning the input layer: feature selection

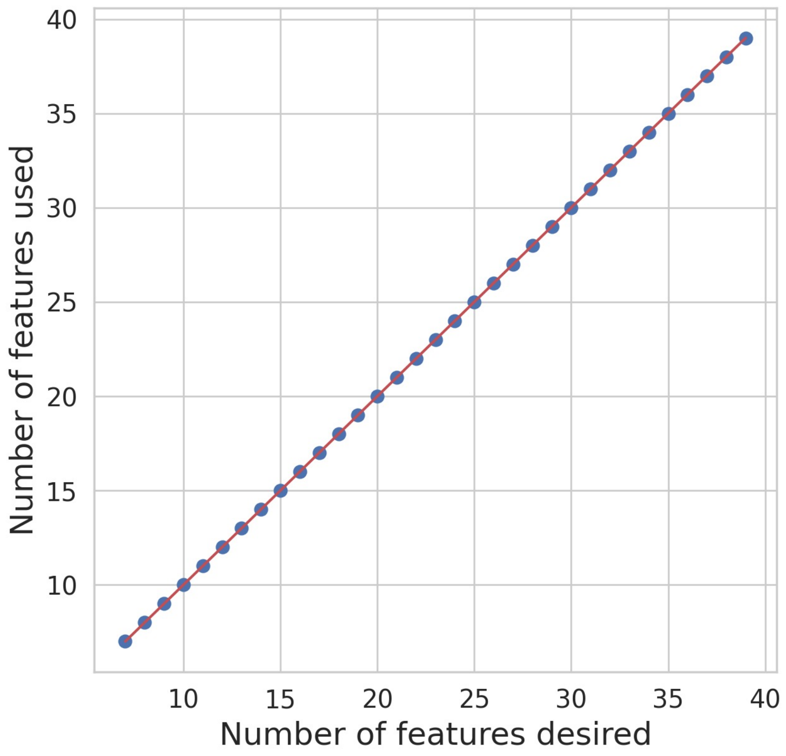


Istituto Nazionale di Fisica Nucleare

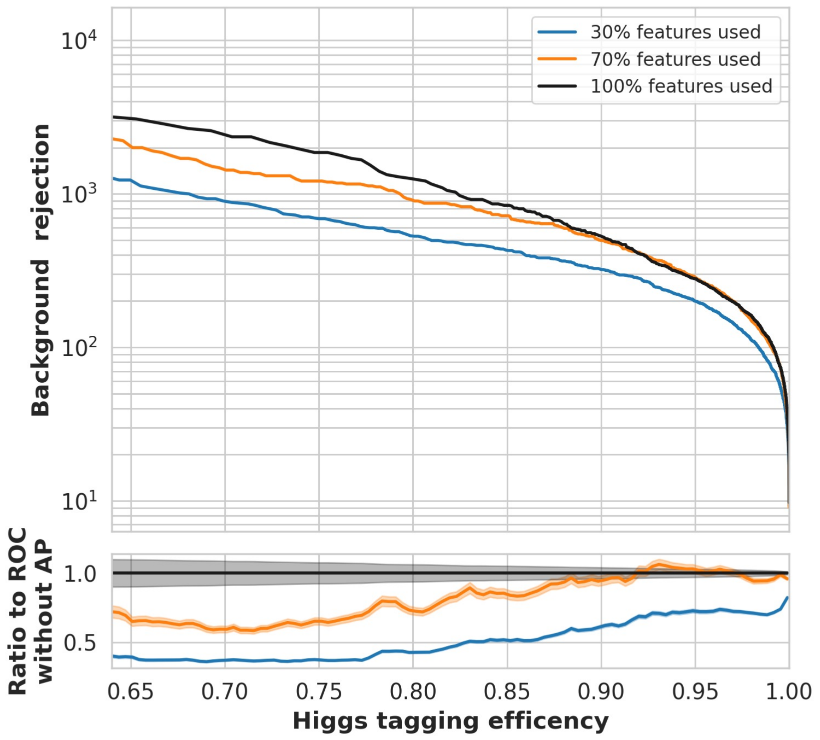
Roberto Iuppa

# Pruning the input layer: feature selection

the number of features used is equal to the requirement




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

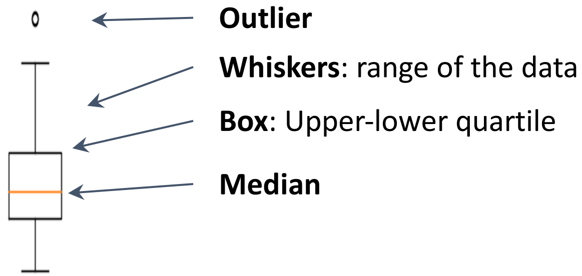


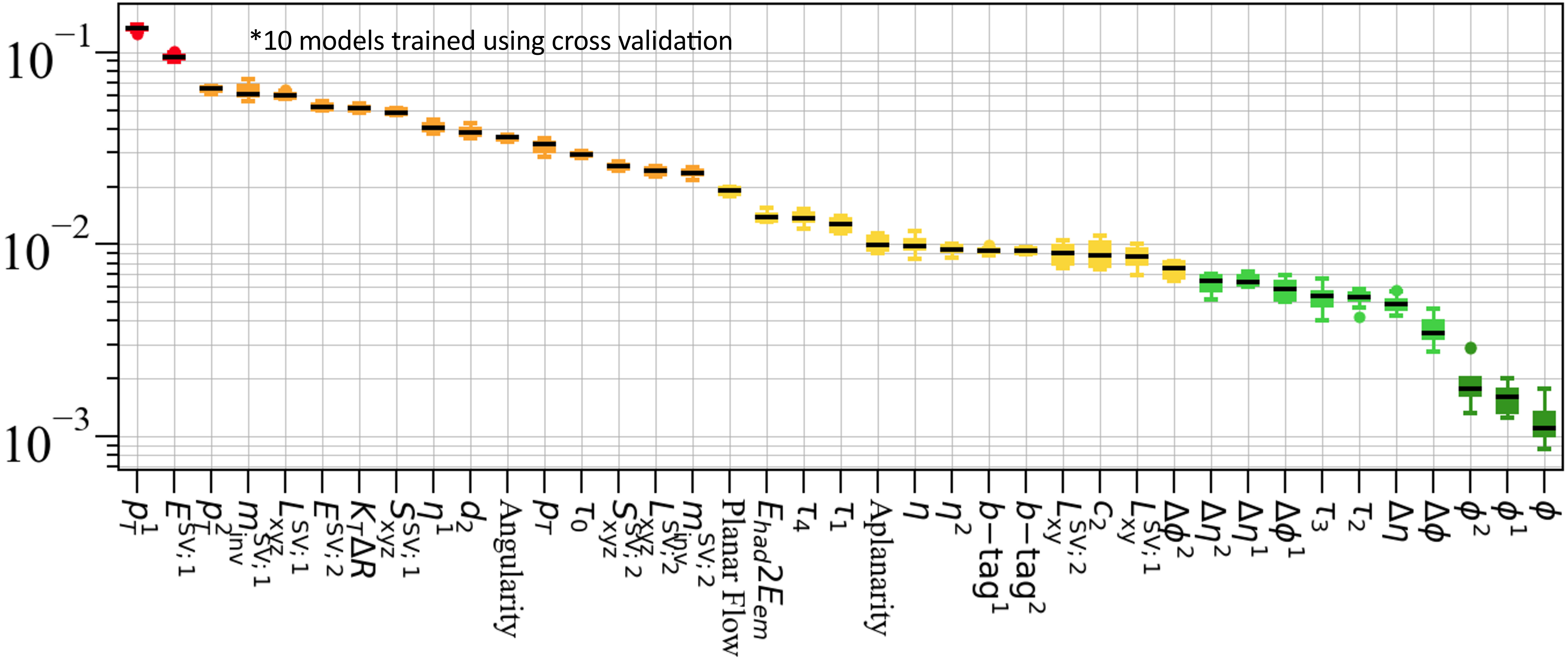

FCNN = IL(39)+4HL(64)+OL
Input feature ranking and selection independent of the hidden layout

Roberto Iuppa

ICHEP 2022 BOLOGNA

# Pruning the input layer: feature ranking



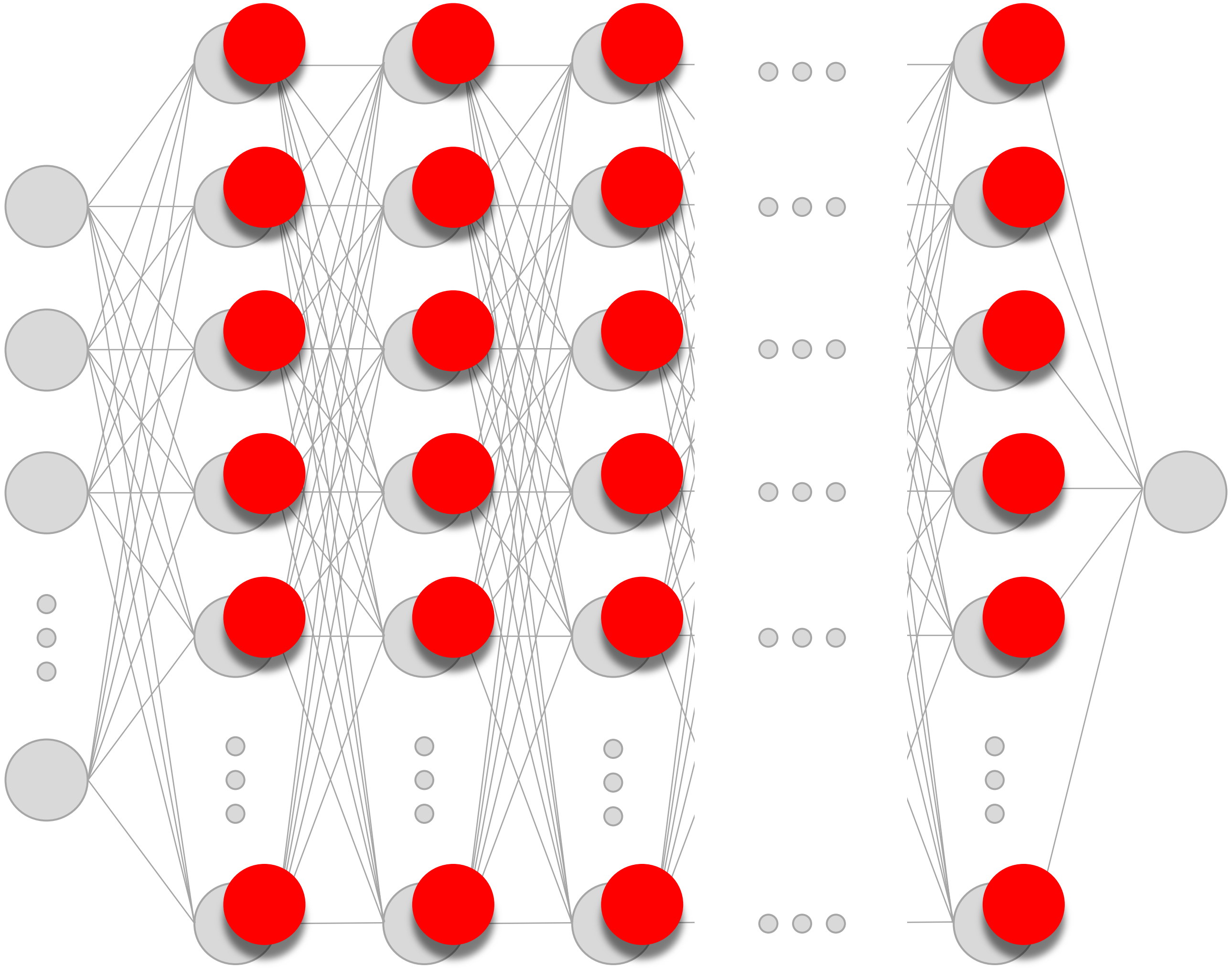




Roberto Iuppa

ICHEP 2022 BOLOGNA

July 8

# Pruning hidden layers



Roberto Iuppa

ICHEP 2022 BOLOGNA

July 8

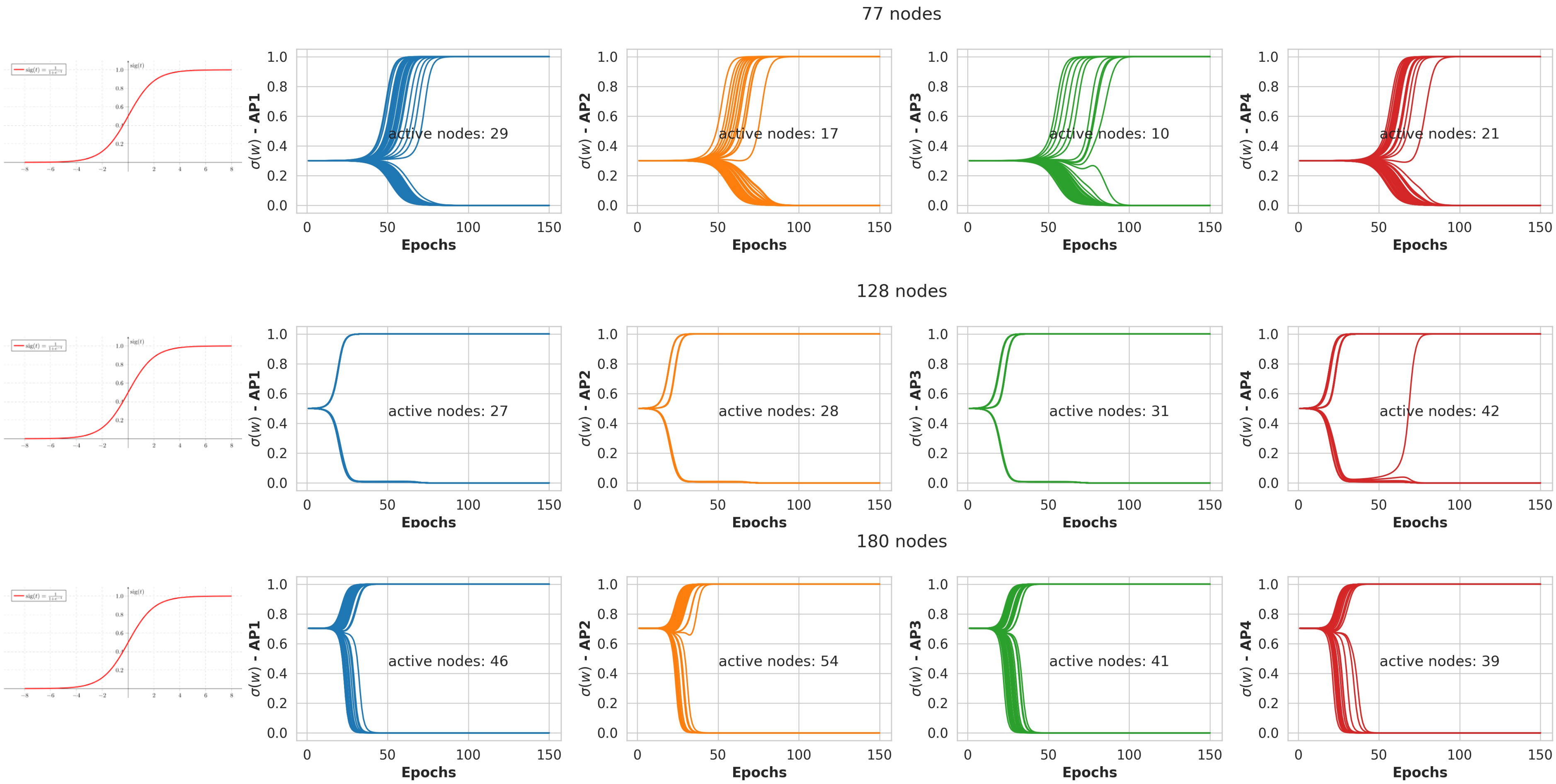
# Pruning hidden layers

FCNN = IL(39)+4HL(64)+OL

Input layer left unpruned to focus on results related to hidden layer pruning.

30%, 50%, 70% of the 256 hidden nodes required.

**HIDDEN LAYER 1** **HIDDEN LAYER 2** **HIDDEN LAYER 3** **HIDDEN LAYER 4**




77/256=30%

128/256=50%

180/256=70%

Roberto Iuppa

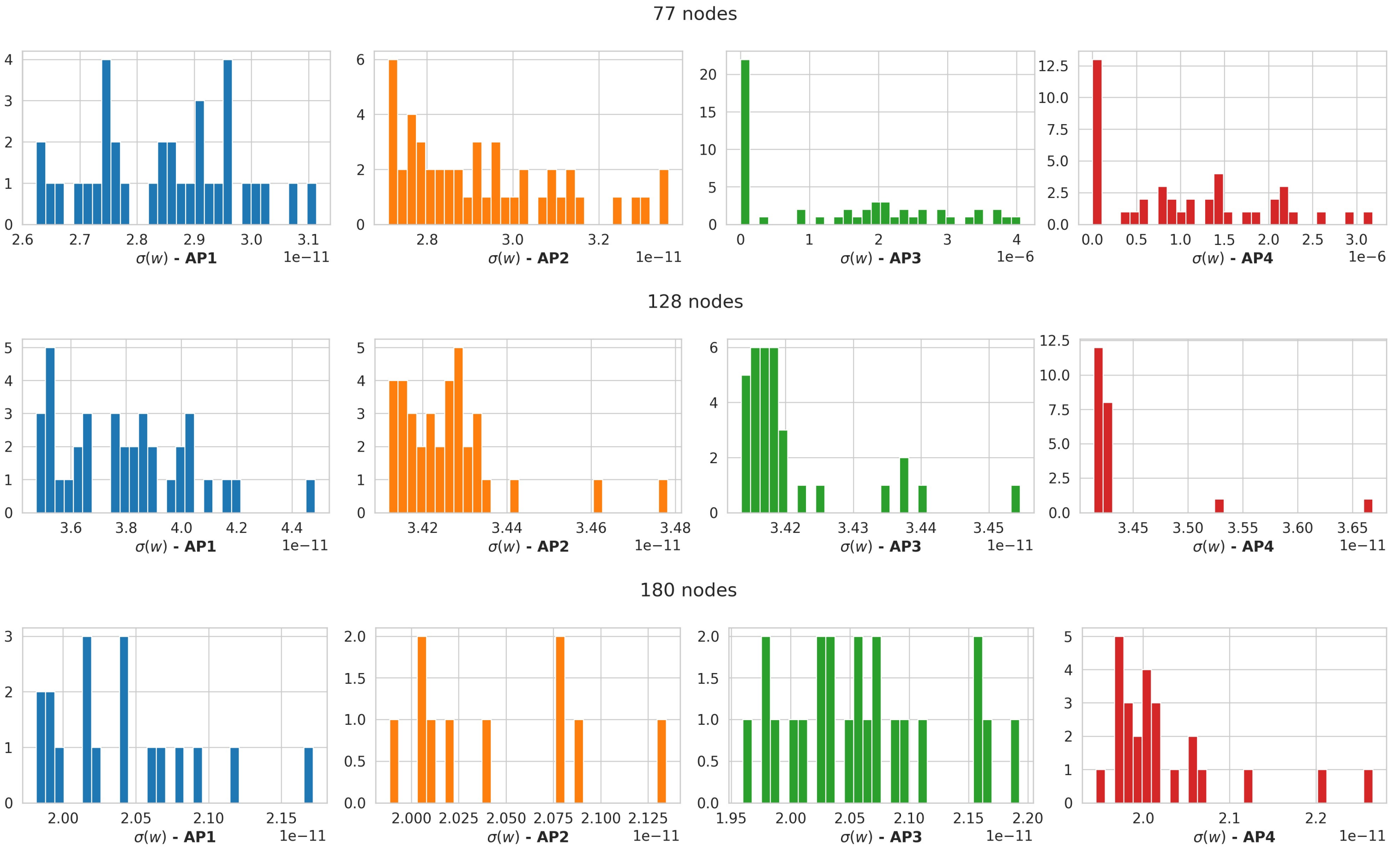
# Pruning hidden layers

FCNN = IL(39)+4HL(64)+OL

Input layer left unpruned to focus on results related to hidden layer pruning.

30%, 50%, 70% of the 256 hidden nodes required.

**HIDDEN LAYER 1** **HIDDEN LAYER 2** **HIDDEN LAYER 3** **HIDDEN LAYER 4**

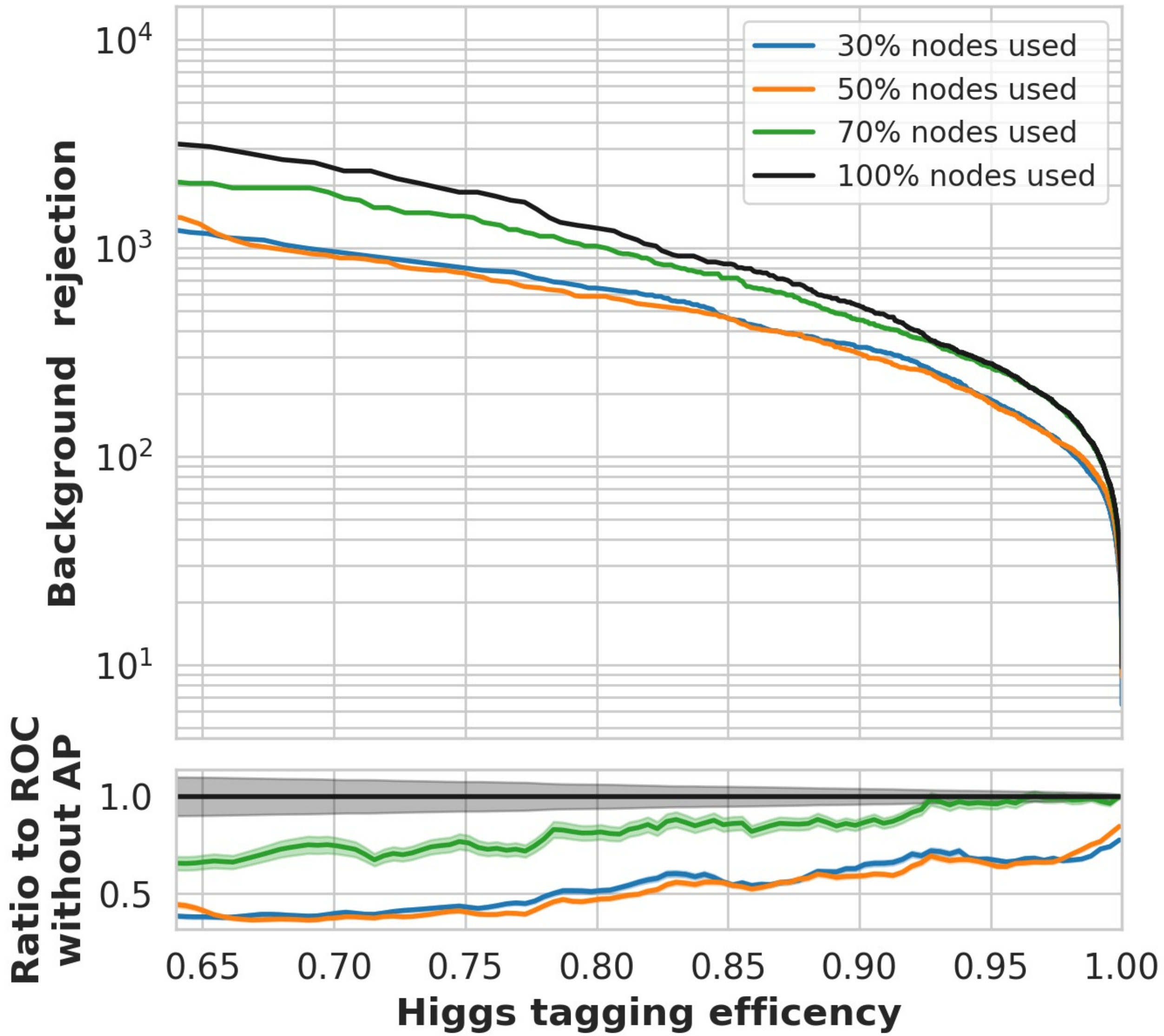


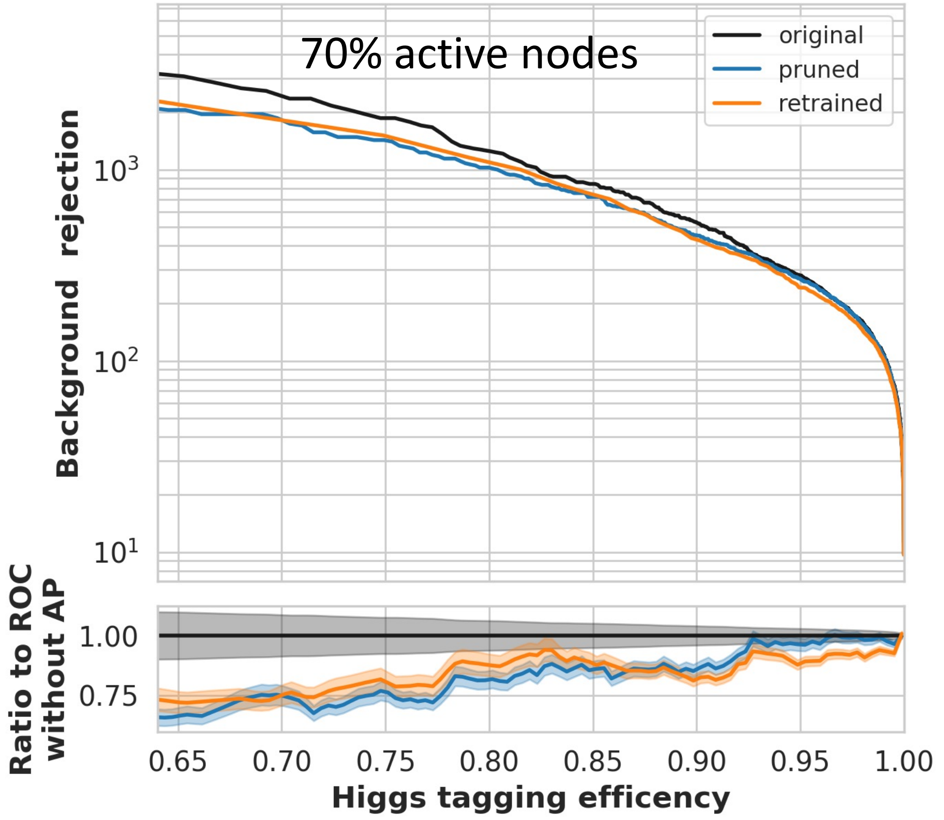

Checking that rejected nodes have AutoPruner filters really set to zero.

Roberto Iuppa

ICHEP 2022 BOLOGNA

# Pruning hidden layers




After finding the optimal network layout with AutoPruner, the reduced network can be retrained as a new independent model, with performance compatible with the pruned one within the uncertainties.

Roberto Iuppa

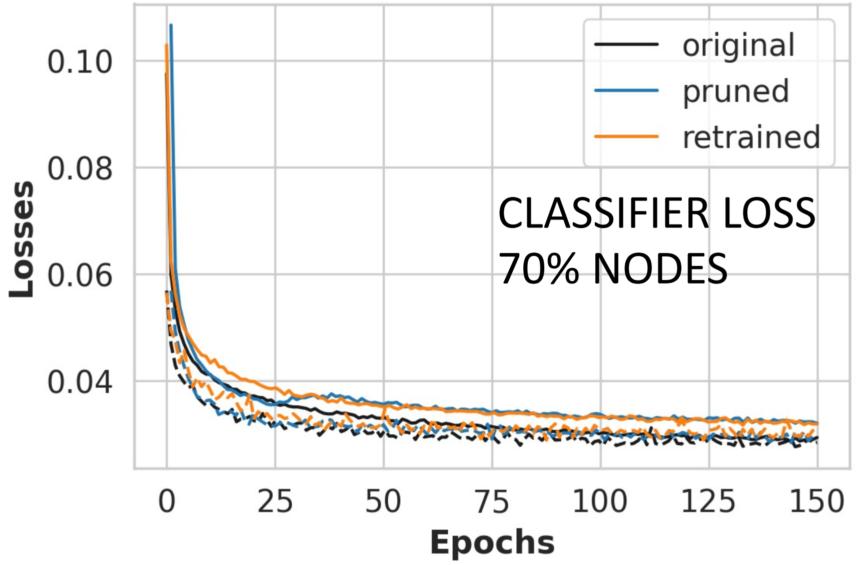
# Pruning hidden layers

FCNN = IL(39)+4HL(64)+OL
30%, 50%, 70% of the 256 hidden nodes required.

$$\mathcal{L}_{TOT} = \mathcal{L}_{classifier}(\vec{x}) + \mathcal{L}_{pruner}(\vec{w})$$

The training process minimizes the classifier and the pruner loss **at the same time**. Every change in the Network determined by AutoPruner determines a temporary increase of the classifier loss.

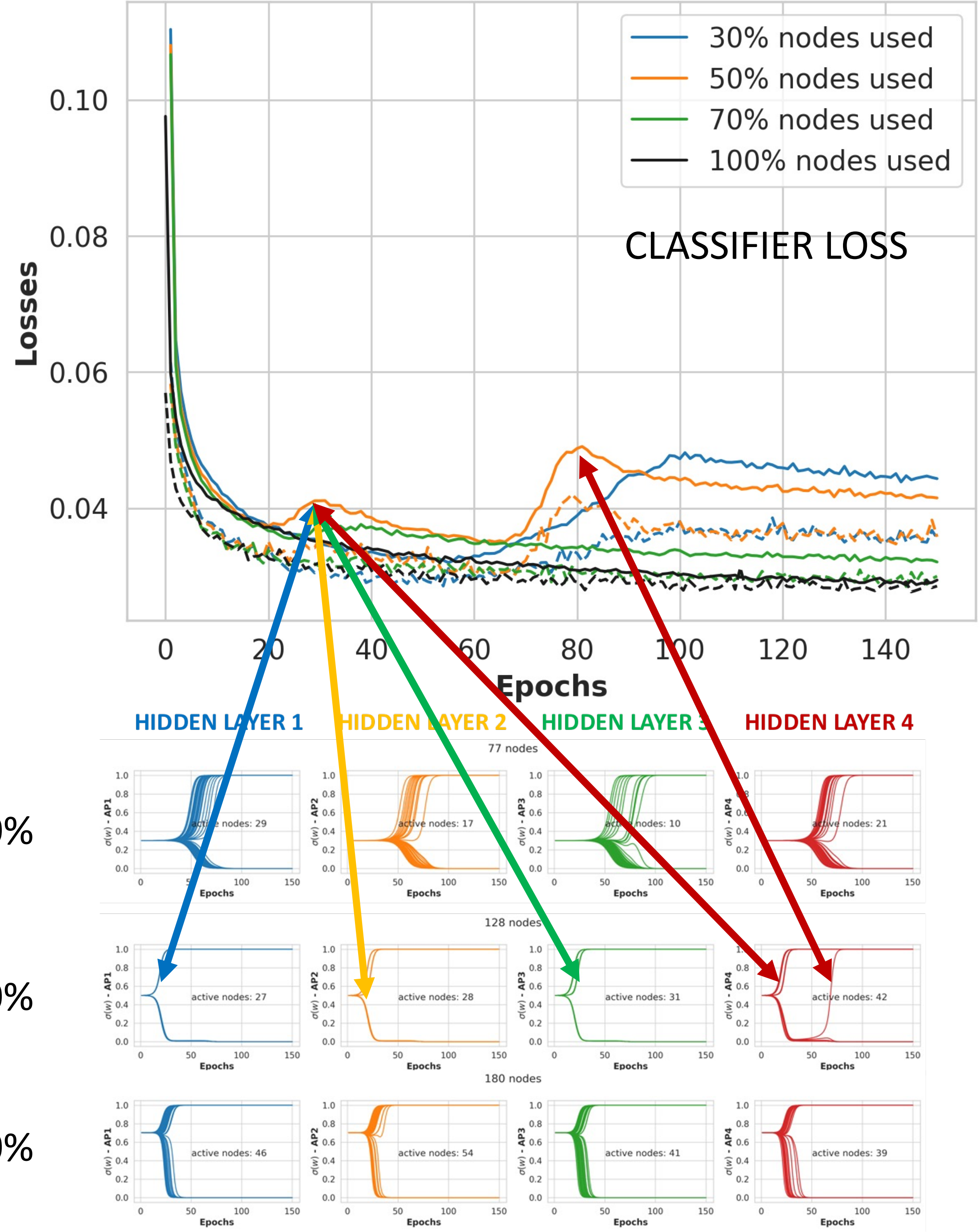
An effect invisible in the reduced&retrained model:




30%

50%

70%




Roberto Iuppa

ICHEP 2022 BOLOGNA

# Conclusions

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

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

Roberto Iuppa