Advanced use cases: training NN on HPC via a Kubernetes based interface

Speaker: Raffaele Gerosa

Institute: Università degli studi di Milano-Bicocca and INFN

Physics motivation beyond this talk …

Chasing the Higgs boson self-coupling

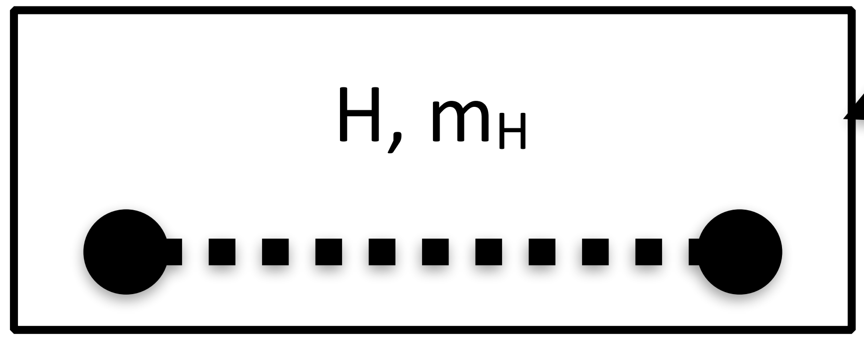
$$\boxed{V(H) = \frac{1}{2}m_{\mathrm{H}}^{2}\mathrm{H}^{2} + \lambda v \mathrm{H}^{3} + \frac{1}{4}\lambda \mathrm{H}^{4} - \frac{\lambda}{4}v^{4}}$$

** Expression of the Higgs boson potential when expanded around the VEV

Chasing the Higgs boson self-coupling

$$V(H) = \boxed{\frac{1}{2}m_{\mathrm{H}}^2\mathrm{H}^2} + \lambda v\mathrm{H}^3 + \frac{1}{4}\lambda\mathrm{H}^4 - \frac{\lambda}{4}v^4$$

** Expression of the Higgs boson potential when expanded around the VEV



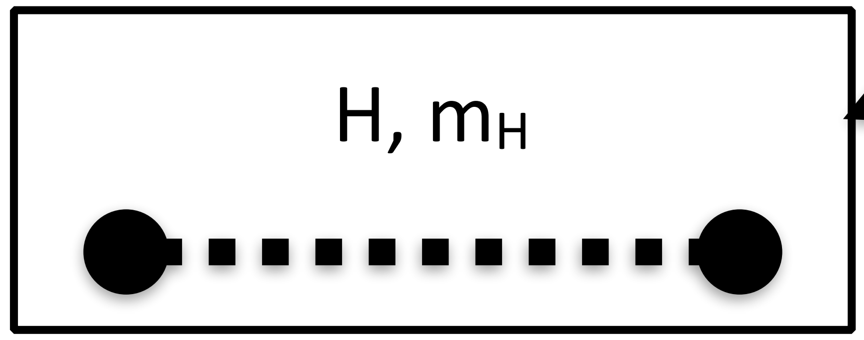
Mass term measured with O(100) MeV precision

Physics motivation beyond this talk …

Chasing the Higgs boson self-coupling

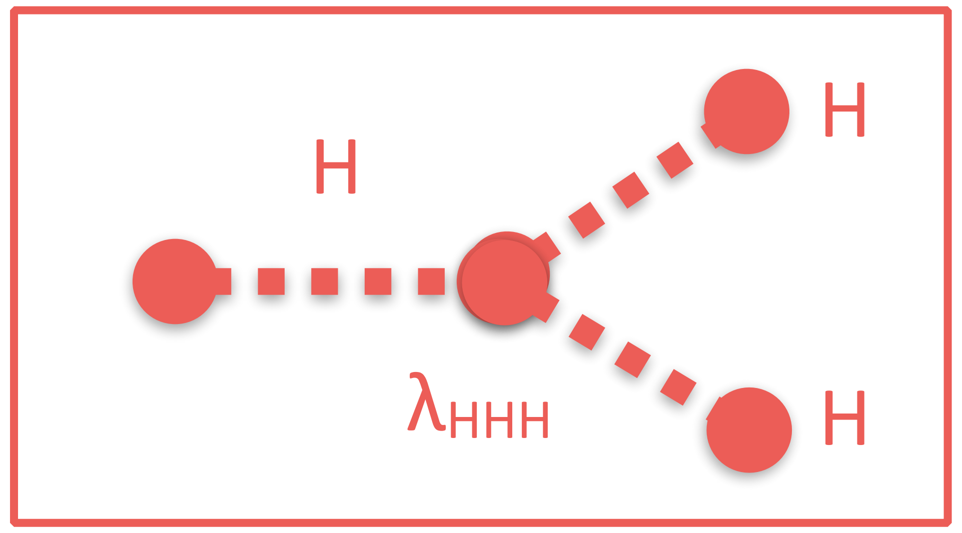
$$V(H) = \boxed{\frac{1}{2}m_{\mathrm{H}}^2\mathrm{H}^2} + \boxed{\lambda v \mathrm{H}^3} + \frac{1}{4}\lambda \mathrm{H}^4 - \frac{\lambda}{4}v^4$$

** Expression of the Higgs boson potential when expanded around the VEV

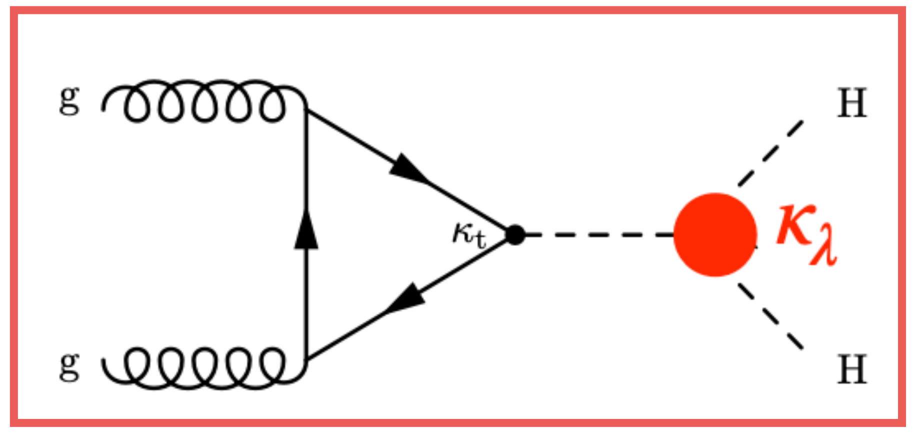


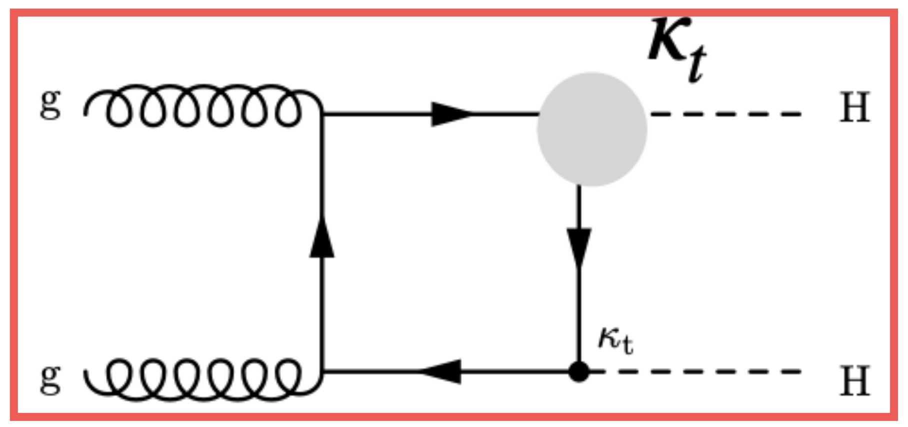
Mass term measured with O(100) MeV precision

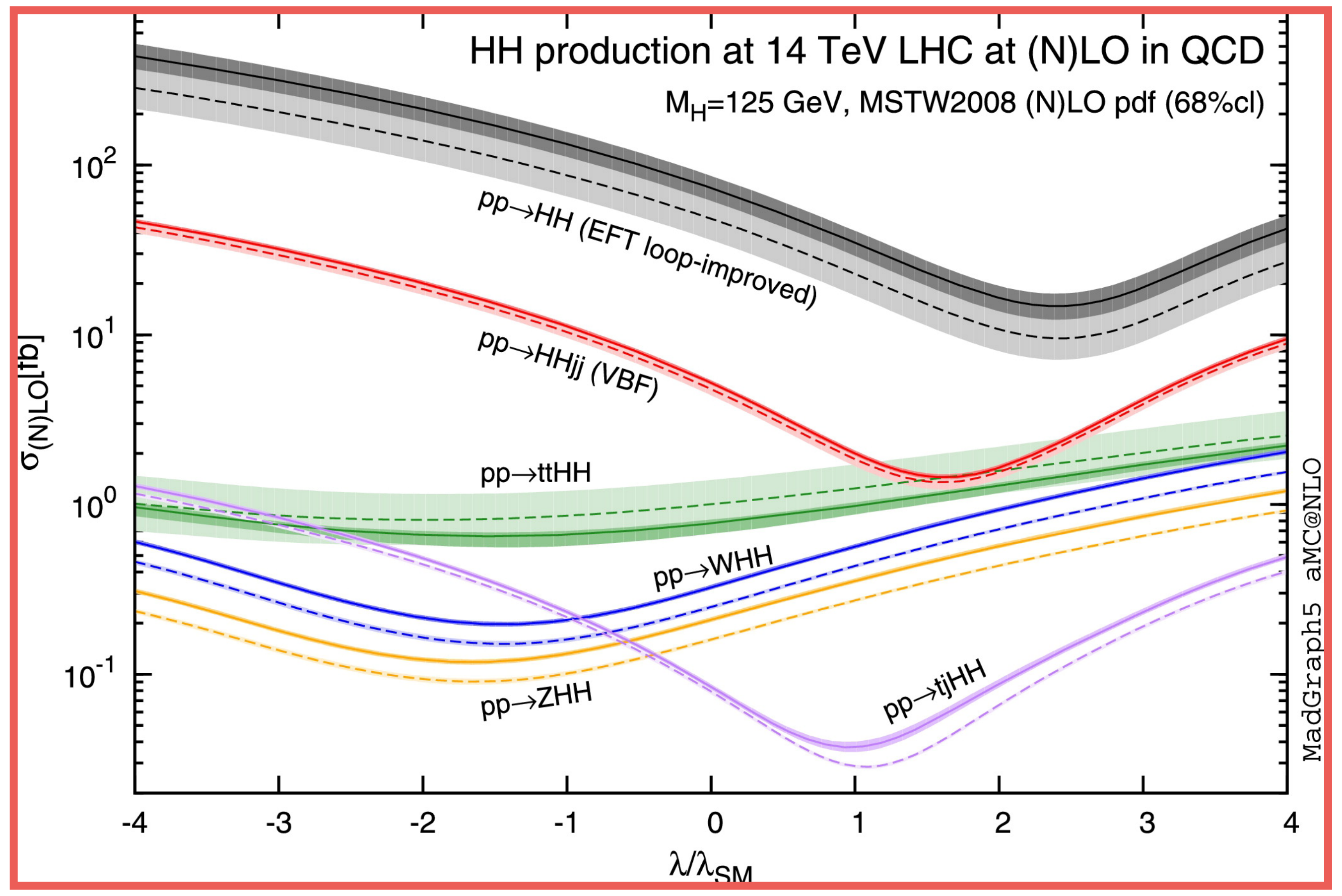
Trilinear coupling



It can be *directly probed* via the non-resonant production of *HH pairs*





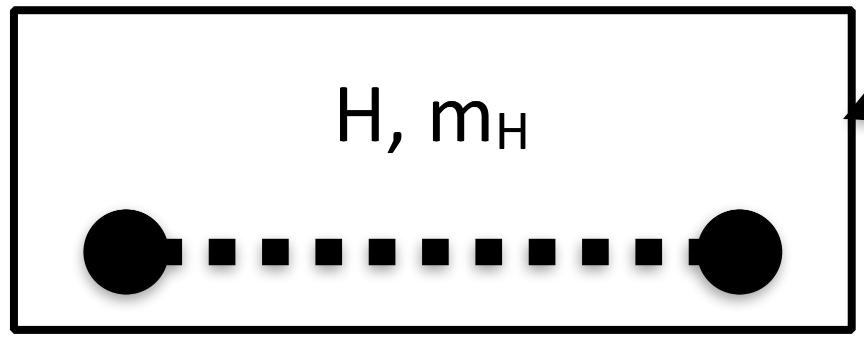


Physics motivation beyond this talk …

Chasing the Higgs boson self-coupling

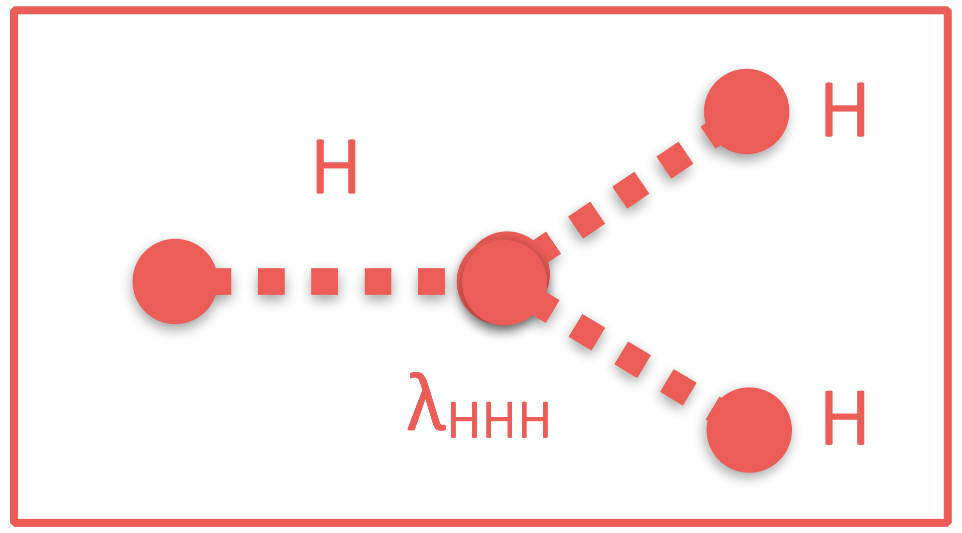
$$V(H) = \boxed{\frac{1}{2}m_{\mathrm{H}}^{2}\mathrm{H}^{2}} + \boxed{\lambda v \mathrm{H}^{3}} + \boxed{\frac{1}{4}\lambda \mathrm{H}^{4}} - \frac{\lambda}{4}v^{4}$$

** Expression of the Higgs boson potential when expanded around the VEV

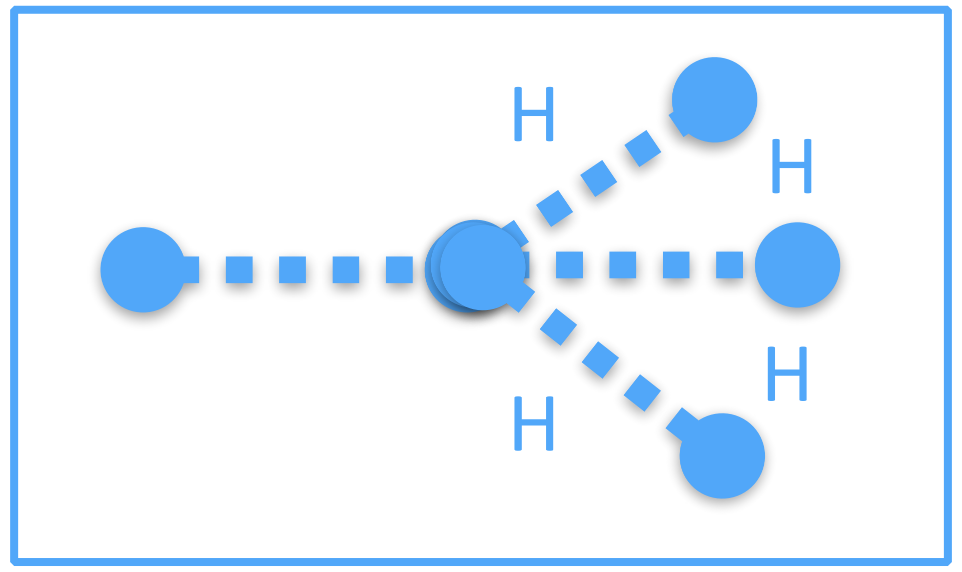


Mass term measured with O(100) MeV precision

Trilinear coupling



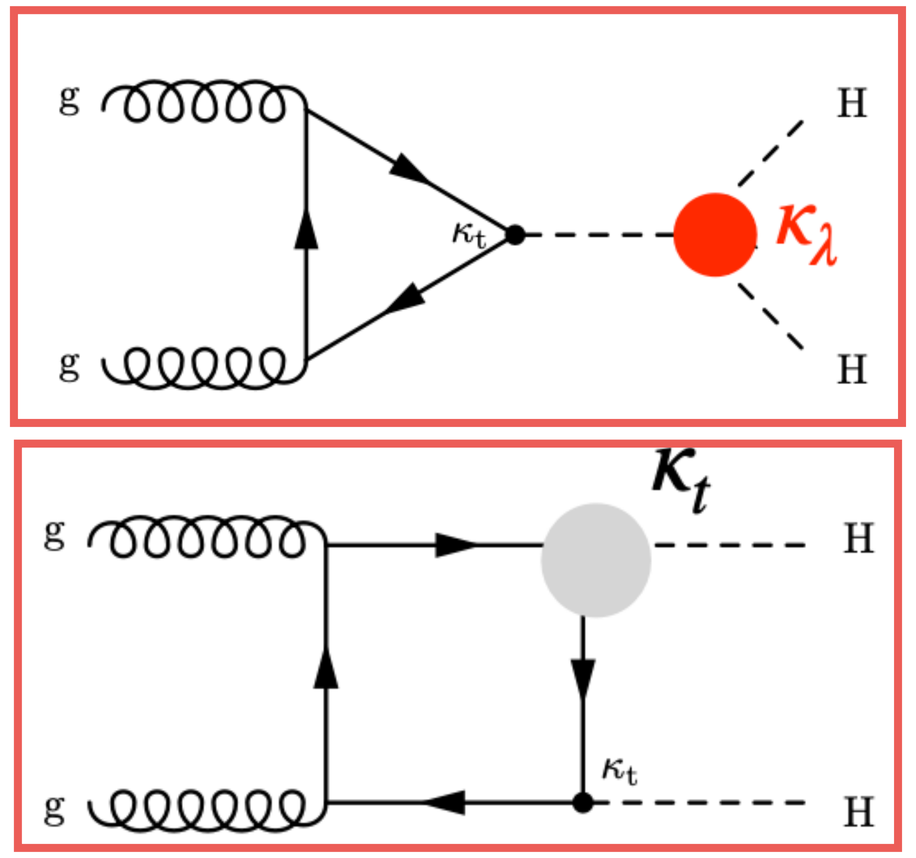
Quartic coupling

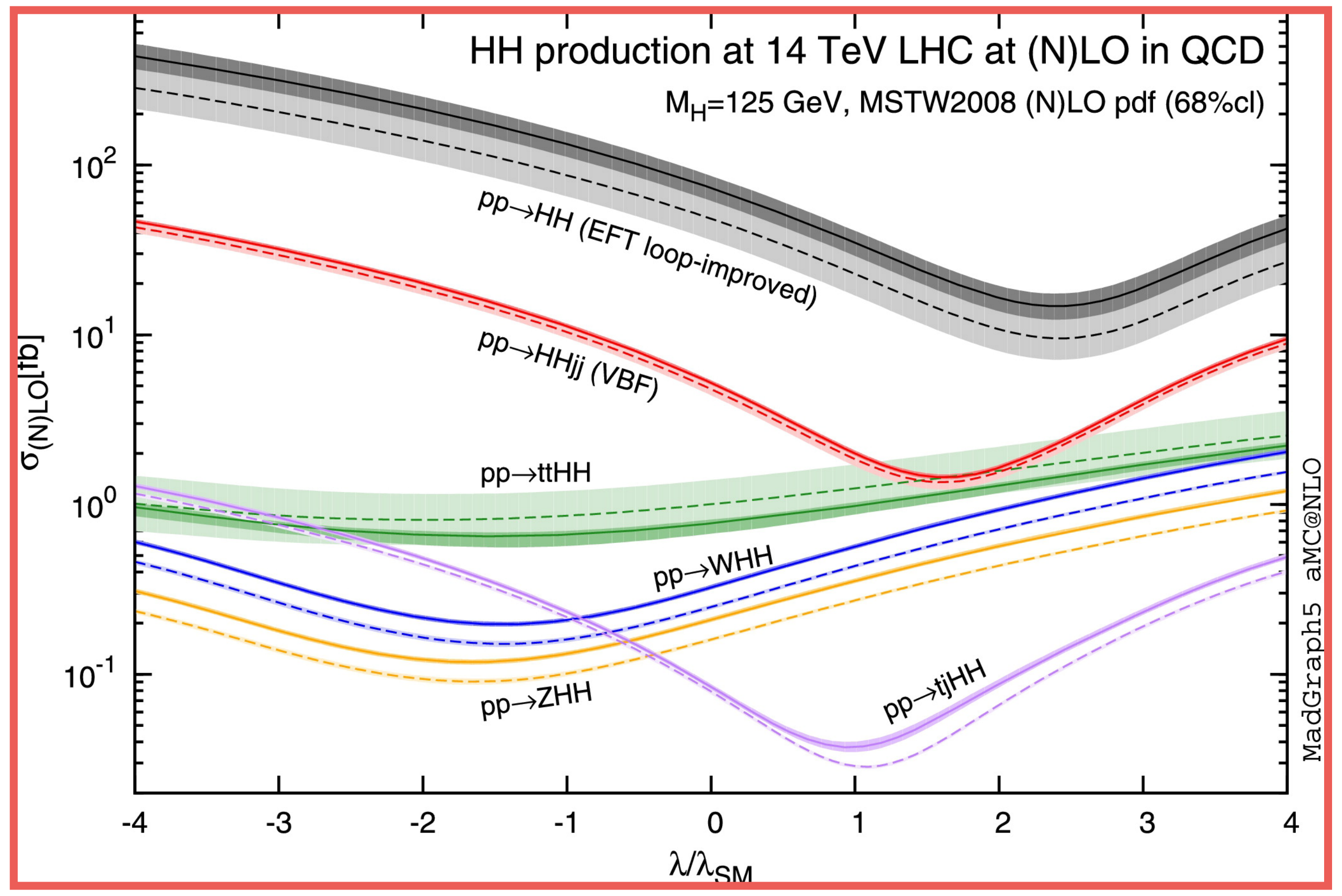


Extremely rare → *out of reach for HL-LHC*

Serves as additional probe for BSM

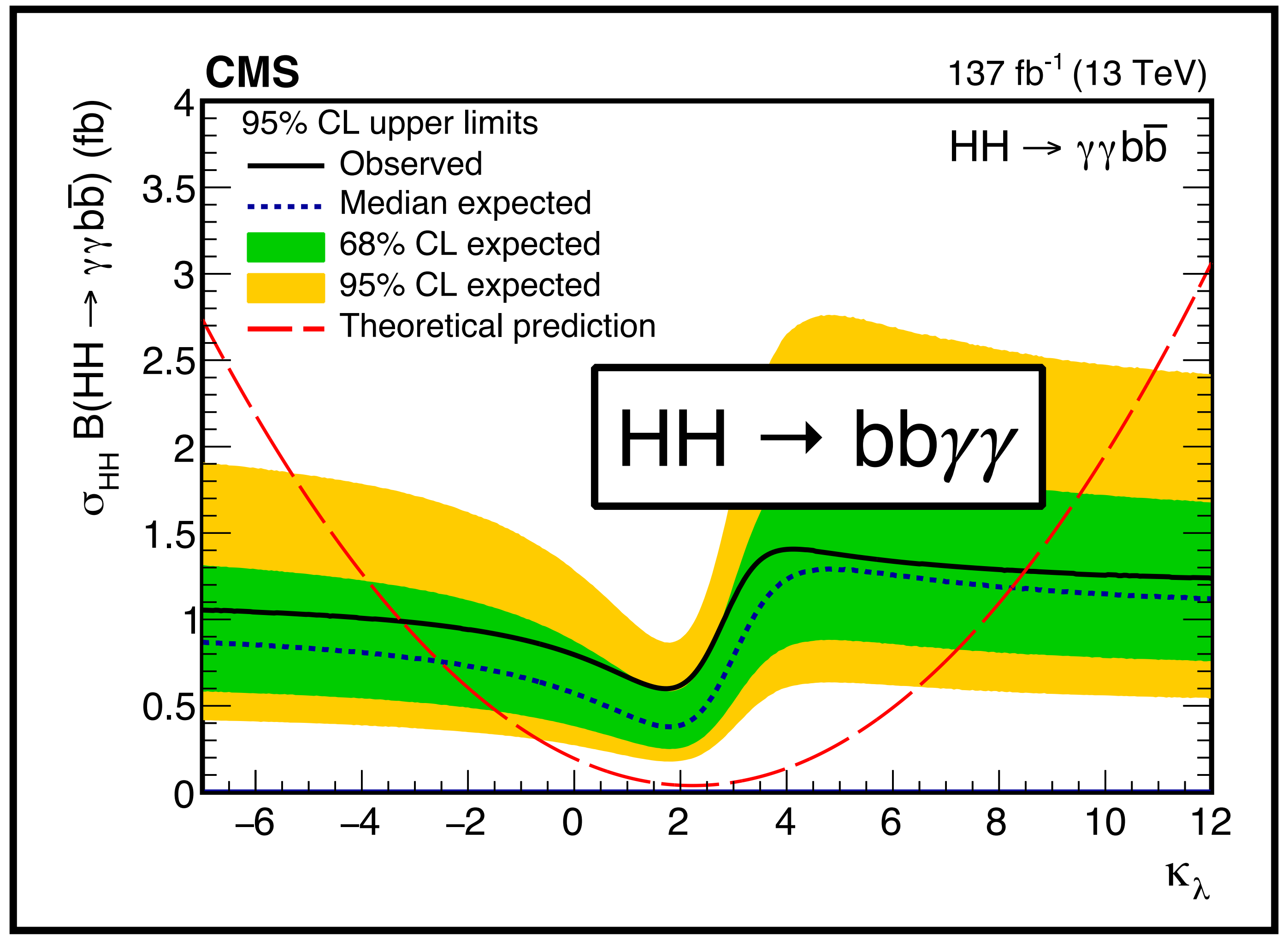
It can be *directly probed* via the non-resonant production of *HH pairs*

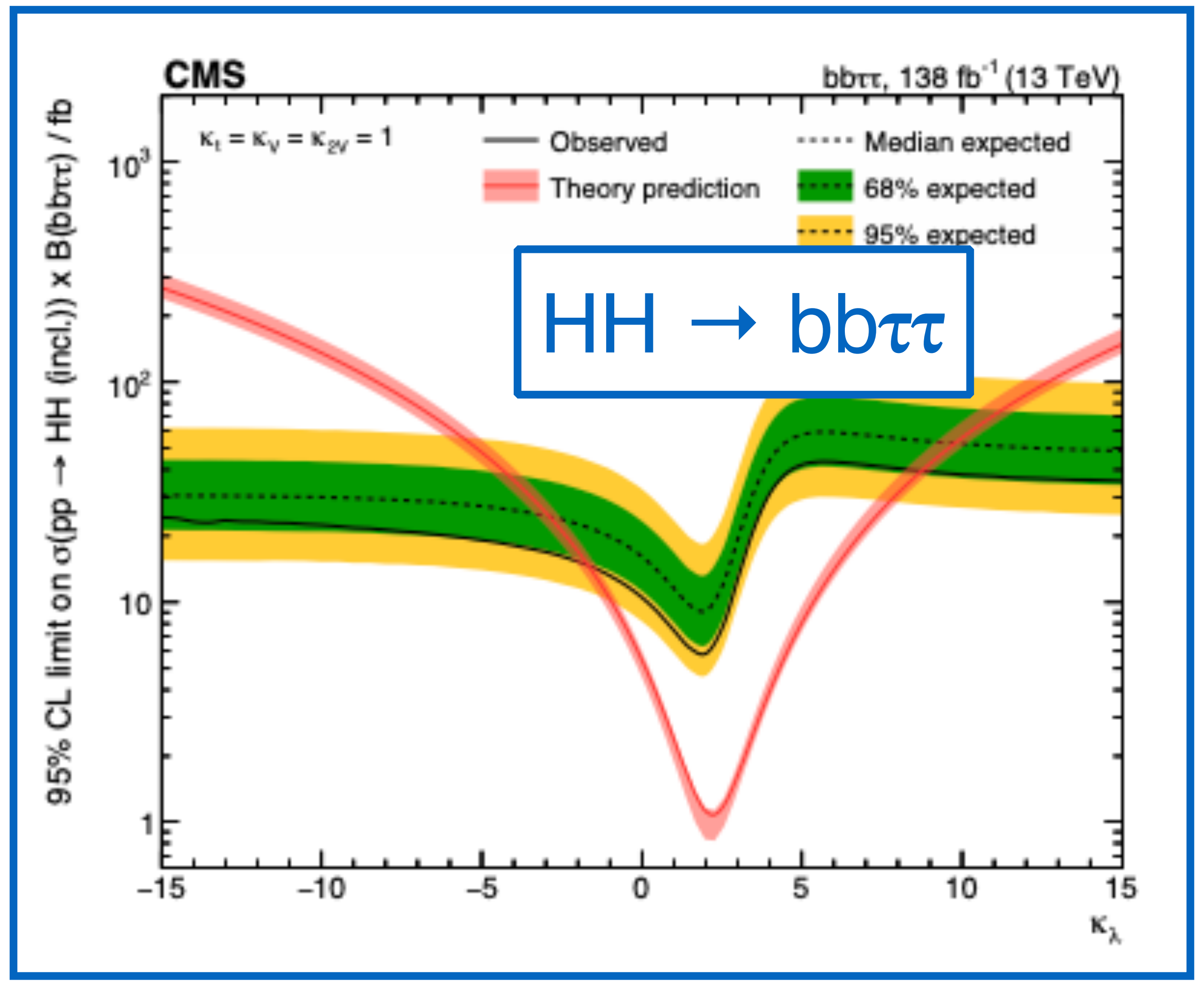


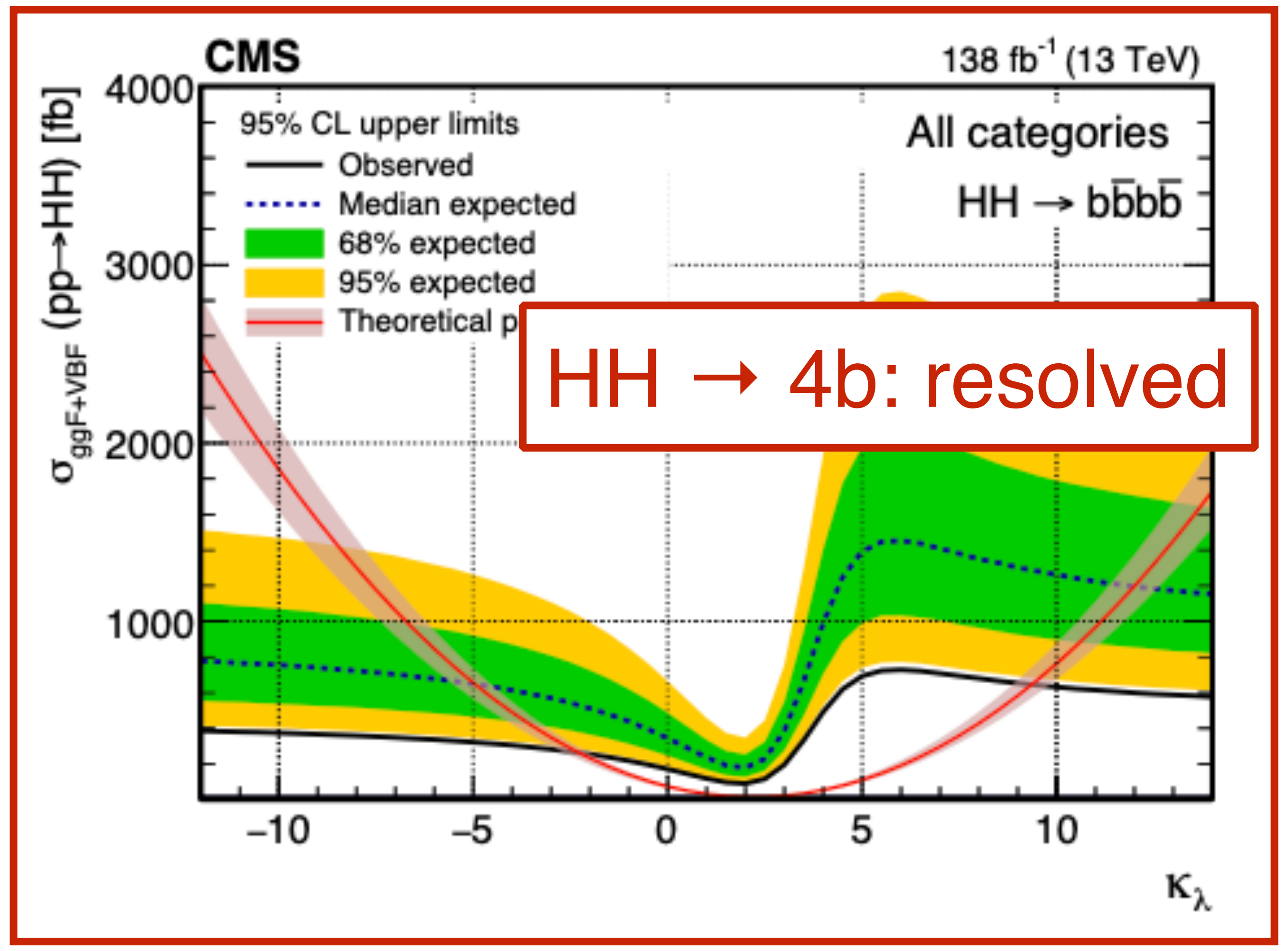


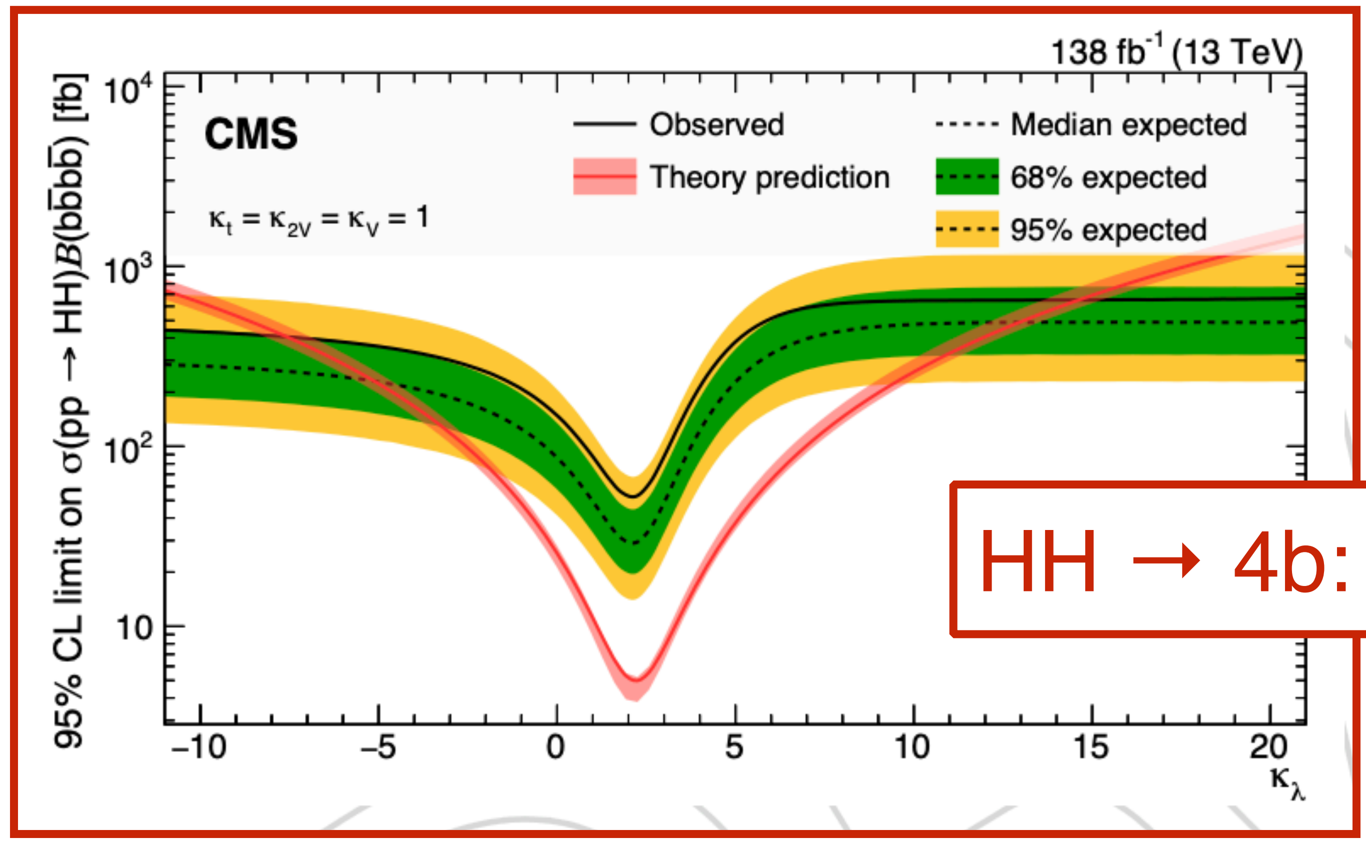
Physics motivation beyond this talk …

What are the most sensitive non-resonant HH analyses to Higgs boson self-couplings (k_λ,k_{2V})?





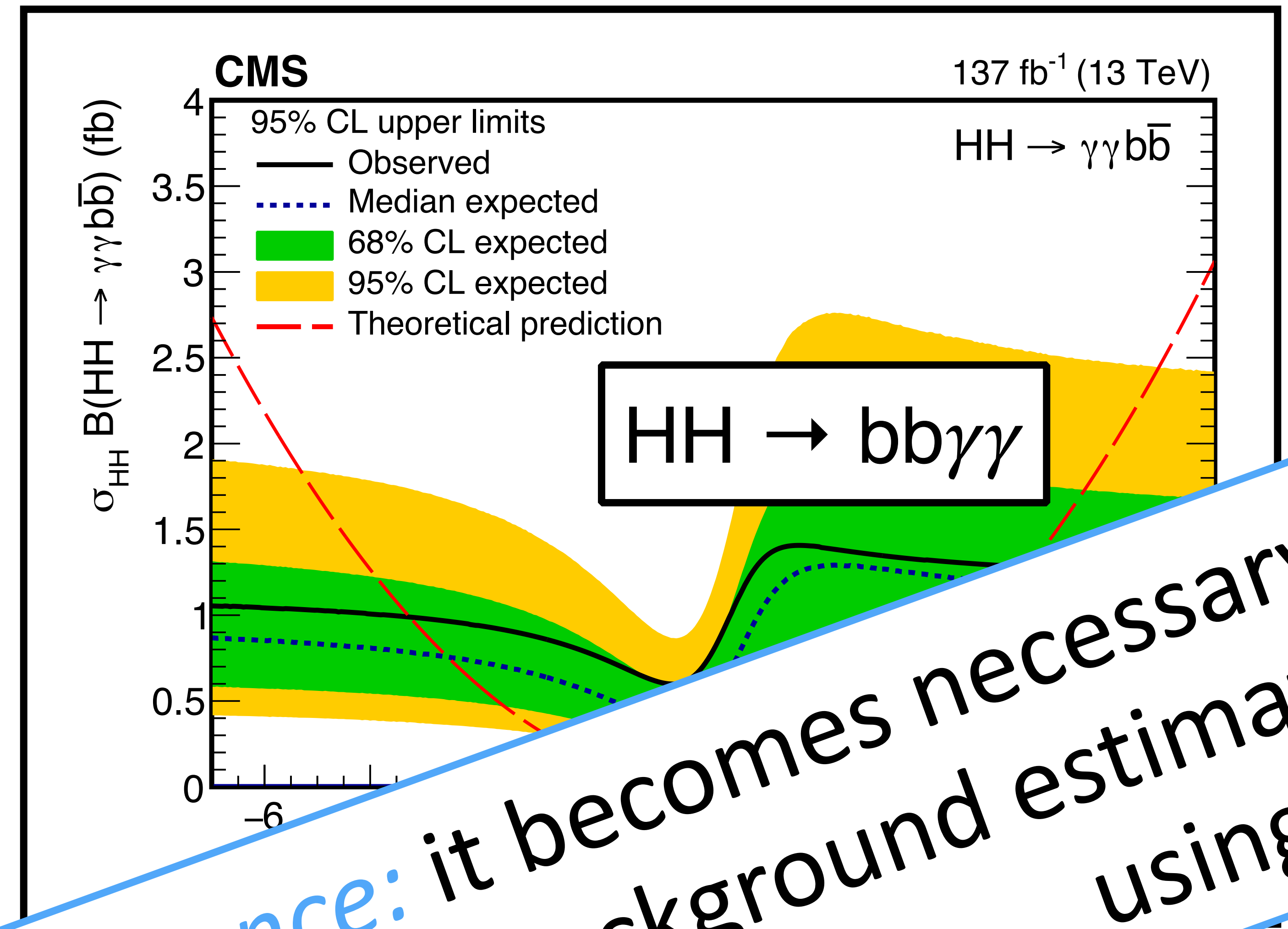


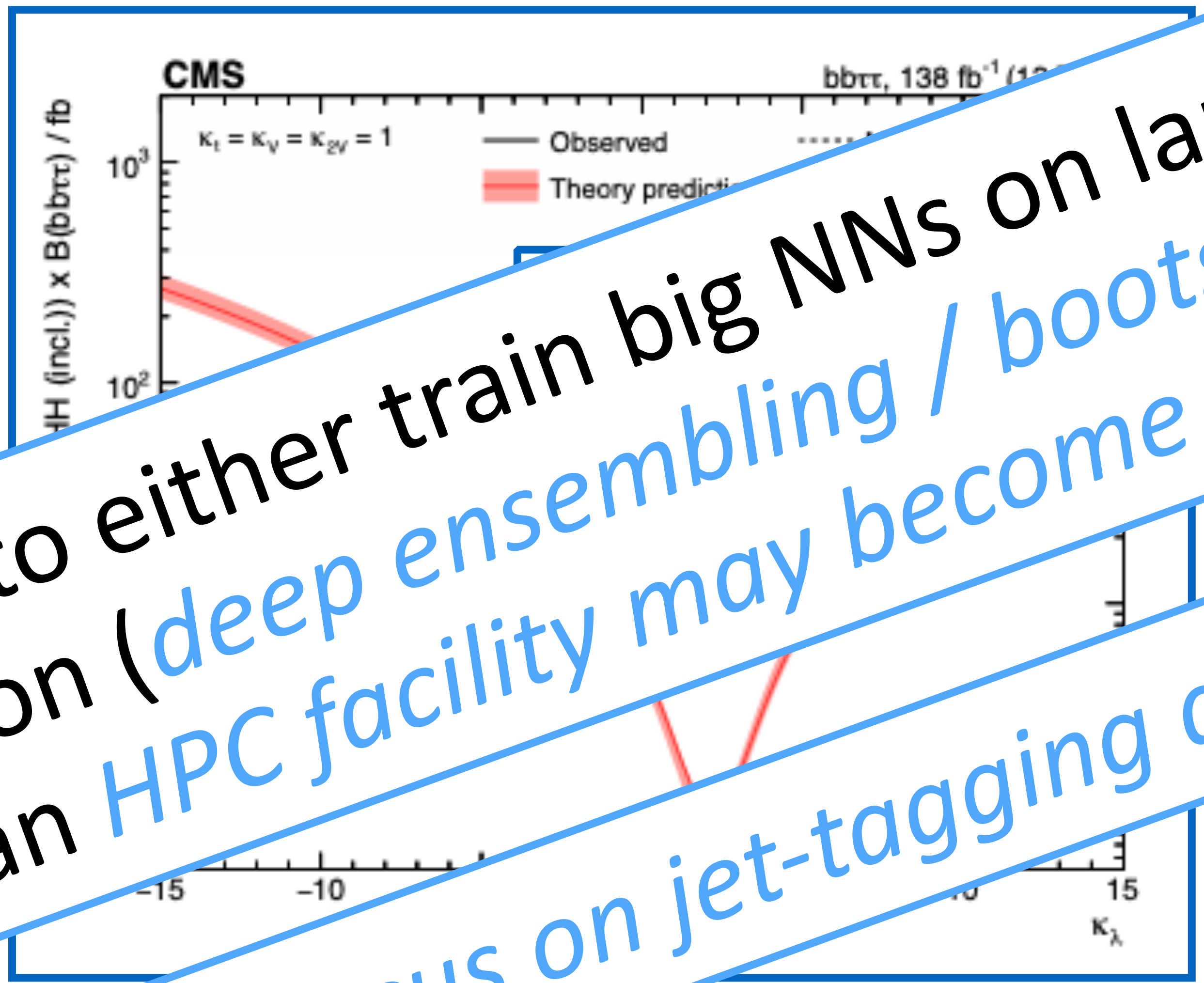


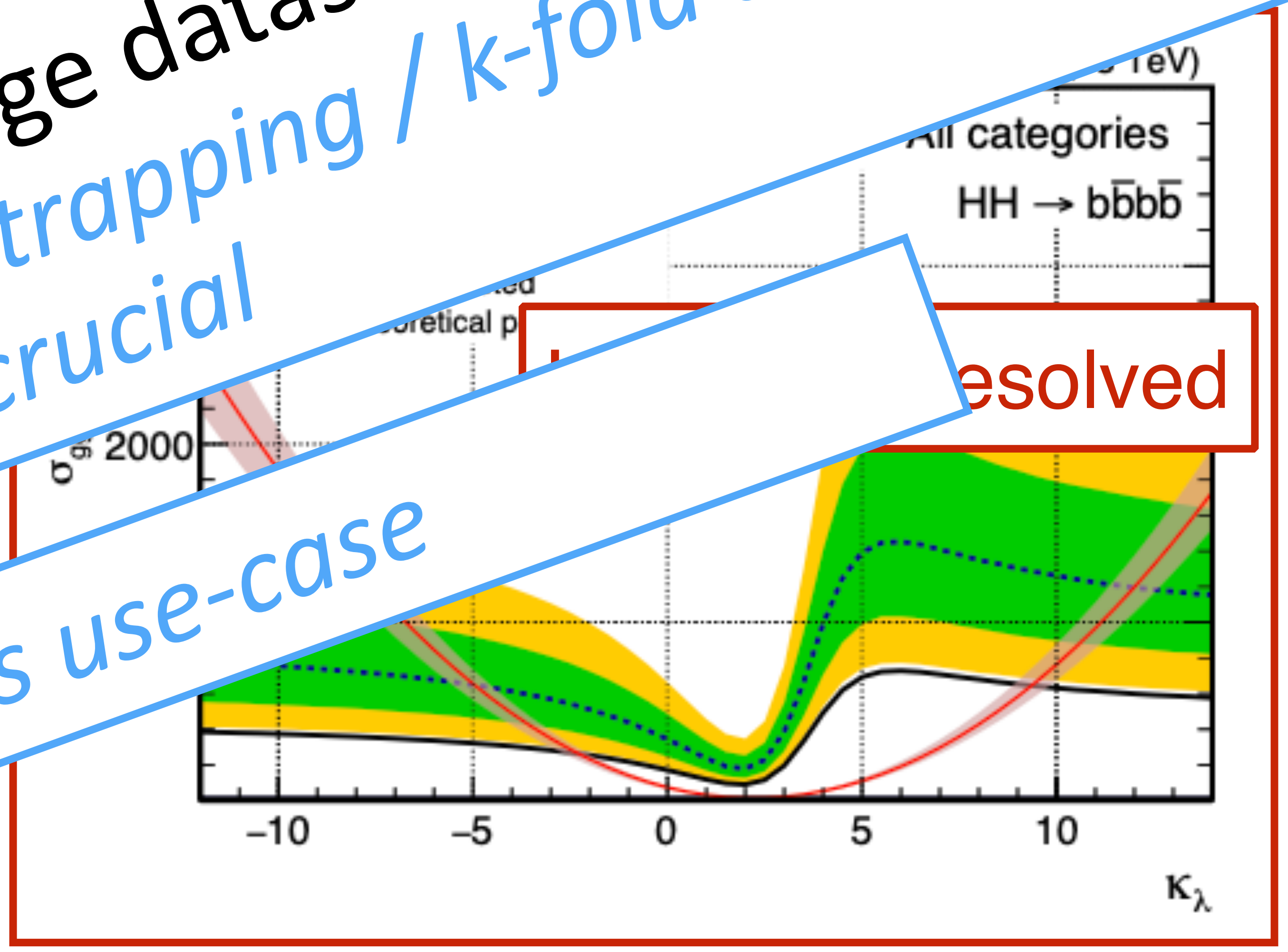
- **Sensitivity of hadronic channels HH→4b and HH→bb$\tau_h\tau_h$** limited by a combination of trigger acceptance, jet tagging performance, signal-to-background ratio, and background estimation
- These are all areas where ML makes a difference

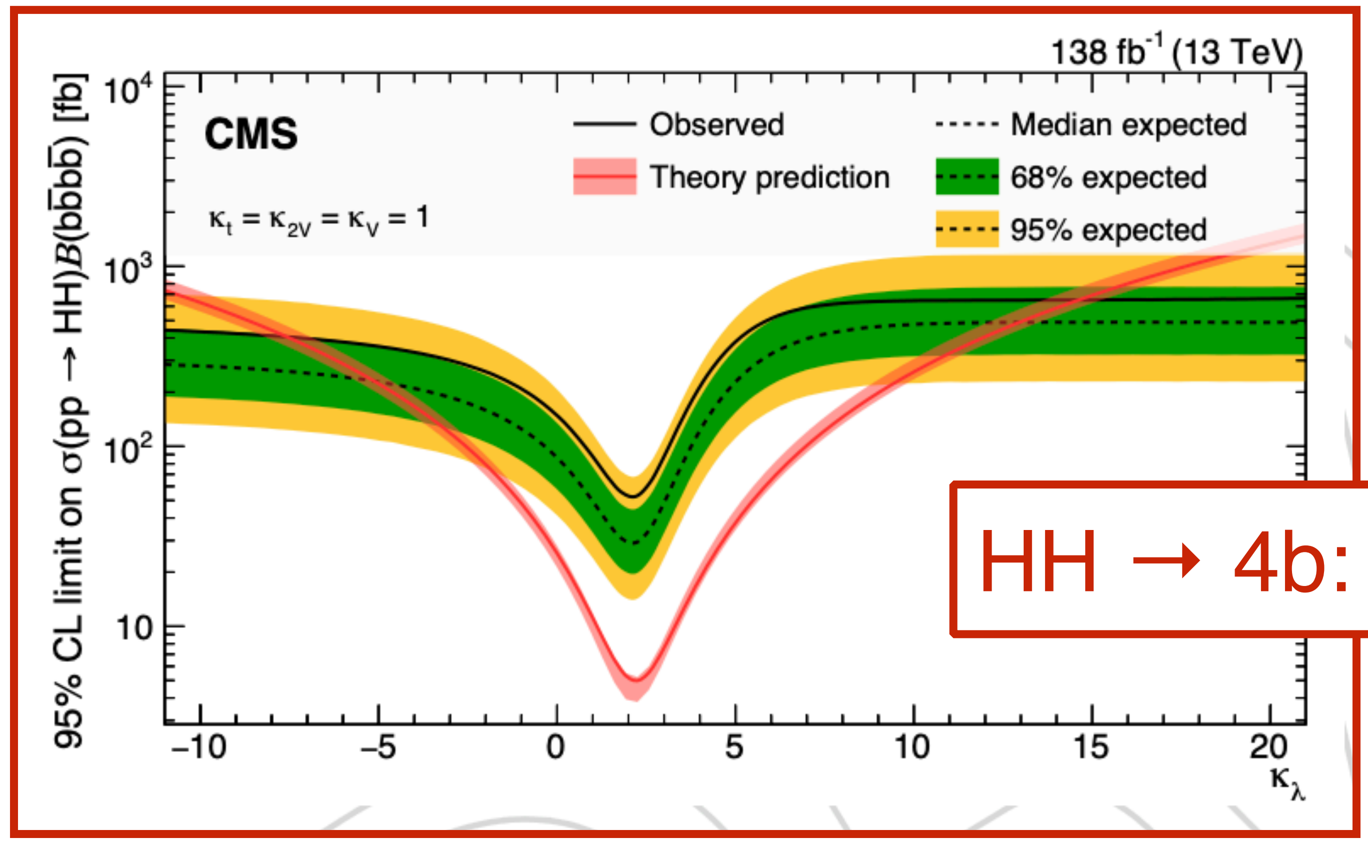
Physics motivation beyond this talk …

What are the most sensitive non-resonant HH analyses to Higgs boson self









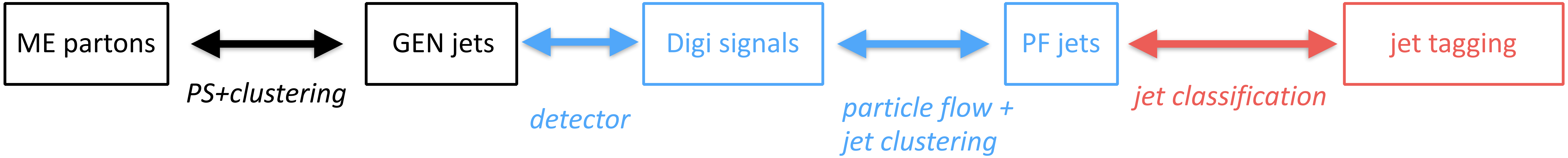
- …channels … H→bb$\tau_h\tau_h$ limited … ation of tri… …e, jet tagging performance, …l-to-backg… …d background estimation
- These are all a… where ML makes a difference

Consequence: it becomes necessary to either train big NNs on large datasets (*jet tagging*) or train several NN for background estimation (*deep ensembling / bootstrapping / k-fold cross validation*) using an *HPC facility may become crucial*

In this talk I'll focus on jet-tagging as use-case

Jet classification based on anti-k_T R=0.4 jets

- **Jet classification:** from cluster of particles infere the jet nature i.e. the original parton or the resonance who produced the jet

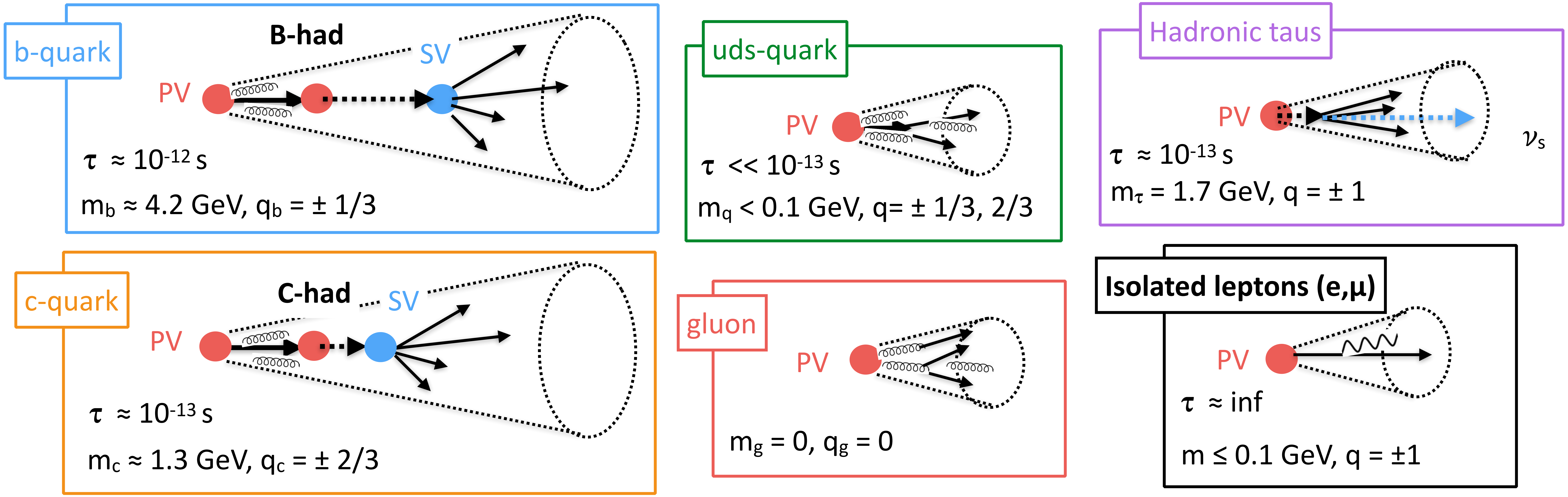


- In CMS the **anti-k_T clustering** with **R=0.4** is meant/used to cluster the fragmentation + hadronization products of a **single parton**

Jet classification based on anti-k_T R=0.4 jets

- **Jet classification:** from cluster of particles infere the jet nature i.e. the original parton or the resonance who produced the jet
- In CMS the **anti-k_T clustering** with **R=0.4** is meant/used to cluster the fragmentation + hadronization products of a **single parton**

Which classes (types) of jets we want to resolve and identify?



Jet-tagging with ML: representation and state-of-art

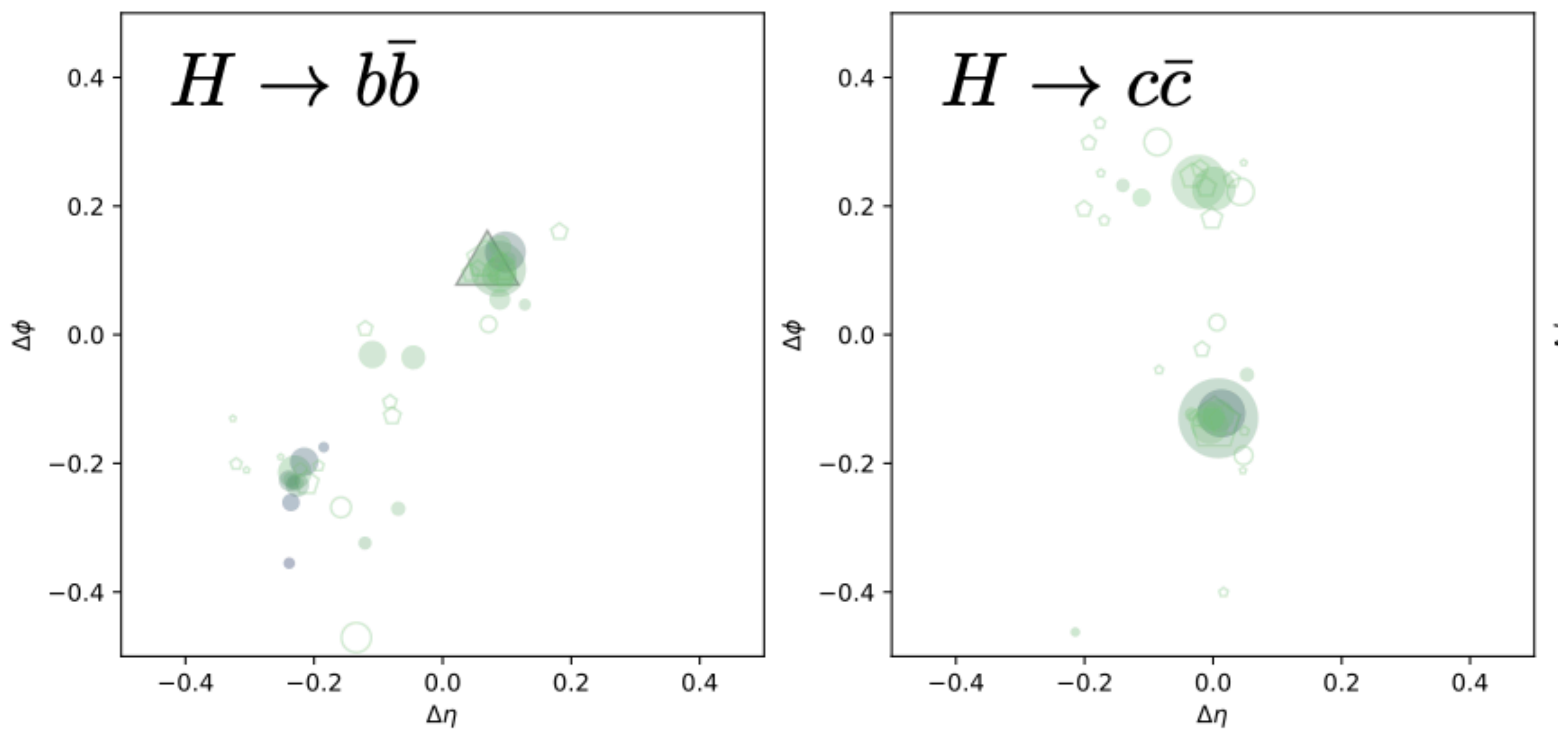
- ***Disclaimer:*** *there is a long history and evolution of jet-tagging algorithm that will be neglected in this talk*

Jet-tagging with ML: representation and state-of-art

- ***Disclaimer:*** *there is a long history and evolution of jet-tagging algorithm that will be neglected in this talk*

Jet as an image ?

- **Jets** can be interpreted **as pixel-like images** taken by a camera called particle detector
- **Cons:** images are sparse and we have multiple detectors providing heterogeneous informations
- **Conclusion:** we cannot easily use technique for image classification to solve jet tagging

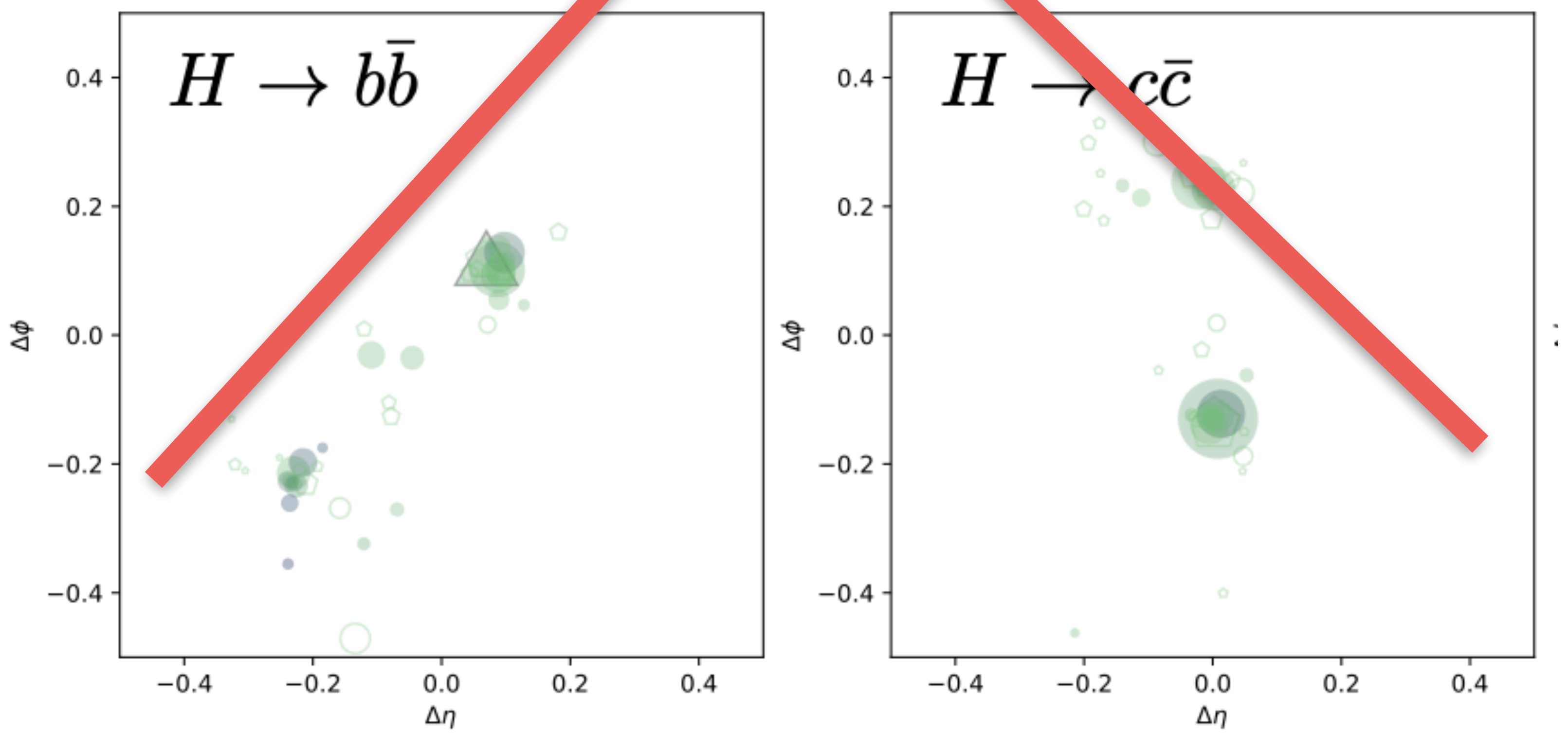


Jet-tagging with ML: representation and state-of-art

- ***Disclaimer:*** *there is a long history and evolution of jet-tagging algorithm that will be neglected in this talk*

Jet as an image ?

- **Jets** can be interpreted **as pixel-like images** taken by a camera called particle detector
- **Cons:** images are sparse and we have multiple detectors providing heterogeneous informations
- **Conclusion:** we cannot easily use technique for image classification to solve jet tagging



Jet as a sequence ?

- A jet is a set of O(10-60) particles → sequence
- Each particle has O(50) features
- A jet is intrinsically an un-ordered set of particles with certain relation due to shower and hadronization structure
- Therefore operations in jet-tagging must be permutation invariant

Jet-tagging with ML: representation and state-of-art

- ***Disclaimer:*** *there is a long history and evolution of jet-tagging algorithm that will be neglected in this talk*

Jet as a sequence ?

- A jet is a set of O(10-60) particles → sequence
- Each particle has O(50) features
- A jet is intrinsically an un-ordered set of particles with certain relation due to shower and hadronization structure
- Therefore operations in jet-tagging must be permutation invariant

Jet as a graph of particles → GCNN or Transformers

- A jet is a sparse set of particles → graph nodes
- *Number of nodes varies* from graph to graph
- Nodes can be related via physics-inspired pairwise features → graph edges
- Particles in the jet are product of the shower algorithm which respects conservation of 4-momentum, Lorentz-invariance, etc
- Transformers networks, which are equivalent to fully connected graphs, are state-of art architectures for jet tagging
- Examples of modern jet-tagging networks:
  - Particle-Transformer (ParT) [Link]
  - Lorentz Geometric-Algebra-Transformer (LGATr) [Link]

A real-example: jet-tagging with ParT .. step-1

- ***Disclaimer:*** *this setup reflects studies I was doing until 1.5 years ago … now BTV-JME are working on a single-framework based on* ***DeepNTuples*** *for producing the training dataset and* ***B-hive [Link]*** *to run the NN trainings*

Production of training Ntuples

- **Dataset** must be large **O(100)M jets** as **ParT model** for **AK4-tagging** contains about **2-2.5M** hyper-parameters
- Jets from a **diverse set** of simulated **physics processes**
- **Processes:** ttbar, V+jets, QCD multijet, VBF-H, VH, high mass Z', etc
- **Input features for each jet:**
  - PF-candidates information → PackedPFCandidates
  - (μ,e,γ) specific features → match with slimmedMuons, etc
  - Lost tracks matched to the jet with p_T > 1 GeV
  - Secondary, kaon, and lambda vertexes matched to the jet
  - If PF-candidate is used to build an HPS-τ

Technical implementation

- Input data-tier: MiniAOD in Run2-UL
- Processing: CMSSW job using native multithread (stream producers/filters) with a lock for thread safety on the output EDAnalyzer
- Inference: this step contains also example on how to run ParT-inference via the ONNXRuntime engine of CMSSW
- Execution on the grid via crab3
- See "producing tuples" in the README of

https://gitlab.cern.ch/rgerosa/particlenetstudiesrun2/-/tree/cmssw_13X_new_features/TrainingNtupleMakerAK4?ref_type=heads#producing-training-ntuples

A real-example: jet-tagging with ParT .. step-2

- ***Disclaimer:*** *this setup reflects studies I was doing until 1.5 years ago … now BTV-JME are working on a single-framework based on* ***DeepNTuples*** *for producing the training dataset and* ***B-hive [Link]*** *to run the NN trainings*

Skimming + preparation of training Ntuples

- **Goals:**
  - Produce a **jet-library** for every sample stored in a ROOT TTree in which every feature of a node (PF/SV/track/lepton) is in a std::vector with length equal to the number of nodes of that type in the jet
  - Define **truth class** and **truth kinematics** from GEN info
- **Skim options:**
  - **jet selection** to be applied
  - PF/SV/track/lepton candidate **minium p_T**
  - Selection for data-domain adaptation: Zμμ, DY(μτ), dijet, tt(eμ)
  - **Caveat:** not all parameters can be override via command line [Link]

Technical implementation

- Standalone compiled C++ via scram (CMSSW) using std::thread for parallel event loop
- Python script to launch skim step on CondorHT CERN batch, a job for all files in a list
- By default I/O on CERN EOS either via XROOTD or local mount-point
- Reduced compression w.r.t. default NanoAOD for faster I/O
- Instructions:

  https://gitlab.cern.ch/rgerosa/particlenetstudiesrun2/-/tree/cmssw_13X_new_features/TrainingNtupleMakerAK4?ref_type=heads#skim-training-ntuples

A real-example: jet-tagging with ParT .. step-2

- ***Disclaimer:*** *this setup reflects studies I was doing until 1.5 years ago … now BTV-JME are working on a single-framework based on* ***DeepNTuples*** *for producing the training dataset and* ***B-hive [Link]*** *to run the NN trainings*

Skimming + preparation of training Ntuples

- **Goals:**
  - Produce a **jet-library** for every sample stored in a ROOT TTree in which every feature of a node (PF/SV/track/lepton) is in a std::vector with length equal to the number of nodes of that type in the jet
  - Define **truth class** and **truth kinematics** from GEN info
- **Skim options:**
  - **jet selection** to be applied
  - PF/SV/track/lep
  - Selection for d ptation: Zμμ, DY(μτ), dijet, tt(eμ)
  - **Caveat:** not all p meters can be override via command line [**Link**]

This produced a jet-library requiring several GB of disk space …
in my case it was up to 1.5 TB split across O(500) ROOT files

Technical implementation

- Standalone c ++ via scram (CMSSW) usin llel event loop
- ch skim step on RN batch, a job for all files ist
- By default I/O on CERN EOS either via XROOTD or local mount-point
- Reduced compression w.r.t. default NanoAOD for faster I/O
- Instructions:

https://gitlab.cern.ch/rgerosa/particlenetstudiesrun2/-/tree/cmssw_13X_new_features/TrainingNtupleMakerAK4?ref_type=heads#skim-training-ntuples

A real-example: training framework

- **Training framework** based on a **customised** version of the **weaver-core** package developed by Huilin Qu for ParticleNet

  - Fully based on **python3** and **PyTorch** as machine-learning engine
  - **ROOT** to **awkward** array conversion via **uproot**, then **awkward** to **numpy / torch tensors**
  - Interface to internal operation via **YAML configuration file**
    - **Training** and **test** dataset **selection** via cut string
    - **Definition** of **new variables** (columns)
    - **Definition** of input **feature transformation** (variable standardisation)
    - Definition of **class weights** and **per-jet sampling weights** based on kinematics
    - **Padding mode** (wrap or zero-padding) and **length** for each input block
    - Definition of **truth labels** (classification labels) and **targets** (regression targets)
  - **NN architecture** and **loss function** are given as external modules

Technical preparation of weaver

- Setup pip-python libraries in a conda environment → [Instructions]
- Example of a weaver configuration needed to run a classification + regression task → [YAML config]
- Example of a ParT architecture config and loss definition → [YAML config]

A real-example: pre-training aspects

- ***Disclaimer:*** *most of the time, once the input dataset is built, some choices need to be taken before feeding a NN with tensor-like inputs*

Class balance

- It is a physics inspired action depending on the final scope
- Higher class weight to b and c will enhance heavy-flavour tagging w.r.t. quark vs gluon
- Class weight used in weaver to up-sample (w>1) or down-sample (w<1) a class of jets

A real-example: pre-training aspects

- ***Disclaimer:*** *most of the time, once the input dataset is built, some choices need to be taken before feeding a NN with tensor-like inputs*

Class balance

- It is a physics inspired action depending on the final scope
- Higher class weight to b and c will enhance heavy-flavour tagging w.r.t. quark vs gluon
- Class weight used in weaver to up-sample (w>1) or down-sample (w<1) a class of jets

Jet kinematic sampling

- **Jet tagging** performance **depend** on **jet p_T**, **η**, as well as on the global event configuration
- The way **jets are sampled could bias** the performance towards a specific phase-space
- A quite **natural strategy** is to **sample uniform** in **p_T** and **η** every class
- This is done by defining a **1D or 2D matrix of weights** binned in (X,Y) observables **for every class** of jets
- These **weights** are used in the **internal jet sampling strategy** not applied in the loss

A real-example: pre-training aspects

- ***Disclaimer:** most of the time, once the input dataset is built, some choices need to be taken before feeding a NN with tensor-like inputs*

Class balance

- It is a physics inspired action depending on the final scope
- Higher class weight to b and c will enhance heavy-flavour tagging w.r.t. quark vs gluon
- Class weight used in weaver to up-sample (w>1) or down-sample (w<1) a class of jets

Jet kinematic sampling

- **Jet tagging** performance **depend** on **jet p_T, η**, as well as on the global event configuration
- The way **jets are sampled could bias** the performance towards a specific phase-space
- A quite **natural strategy** is to **sample uniform** in **p_T** and **η** every class
- This is done by defining a **1D or 2D matrix of weights** binned in (X,Y) observables **for every class** of jets
- These **weights** are used in the **internal jet sampling strategy** not applied in the loss

Technical example

- **This is done once** for the entire jet-library **provided** that the definition of truth labels and sampling binning isn't changed
- **RDF python code** that computes the sampling weights **[Link]**
- **Weights stored** in the configuration file, see for example **[YAML config weights]**

A real-example: pre-training aspects

- ***Disclaimer:*** *most of the time, once the input dataset is built, some choices need to be taken before feeding a NN with tensor-like inputs*

Class balance

- It is a physics inspired action depending on the final scope
- Higher class weight to b and c will enhance heavy-flavour tagging w.r.t. quark vs gluon
- Class weight used in weaver to up-sample (w>1) or down-sample (w<1) a class of jets

Jet kinematic sampling

- **Jet tagging** performance **depend** on **jet p_T, η**, as well as on the global event configuration
- The way **jets are sampled could bias** the performance towards a specific phase-space
- A quite **natural strategy** is to **sample uniform** in **p_T** and **η** every class
- This is done by defining a **1D or 2D matrix of weights** binned in (X,Y) observables **for every class** of jets
- These **weights** are used in the **internal jet sampling strategy** not applied in the loss

Feature engineering

- Basic feature standardisation can help in maximising NN performance
- Implementation in weaver: for every input feature in each block one can define a vector of parameters
  - Mean value to shift to 0
  - Variance value to transform to variance 1
  - Clipping values for outliers
- RDF python code that computes standardisation parameters [Link]
- Weights stored in the configuration file, see for example [YAML config weights]

Running training: challenges

- ***Goal:*** run the training until convergence in order or 4-5 days, a week at maximum for the more complex cases

High CPU throughput needed

- **Large dataset size** O(1TB) that needs to analysed at every epoch → ***cannot be loaded in memory***
- **Iterative PyTorch loader** that loads in memory a chunk of every file, converts to torch tensors, apply transformation, defines new columns, and build batches
- **The data-loader** need to span across different **parallel workers** to keep the pace with the **training loop**

Order of 16 workers needed

Large RAM needed

- Load cyclically a small chunk of each file, pad to a fix dimension, create new columns, etc
- Instantiate multiple iterative workers for I/O is more demanding on the memory side
- Uncompress files and decided whether to use lower floating point precision

Order of 64-72 Gb per training

GPU memory and speed

- Mini-batch not ideal cause of larger loss fluctuations and more backward passes
- Larger batches required lot of GPU memory

Parallel on 2 GPUs 24 Gb each or single GPU with 48 Gb

Solution is to use an HPC facility

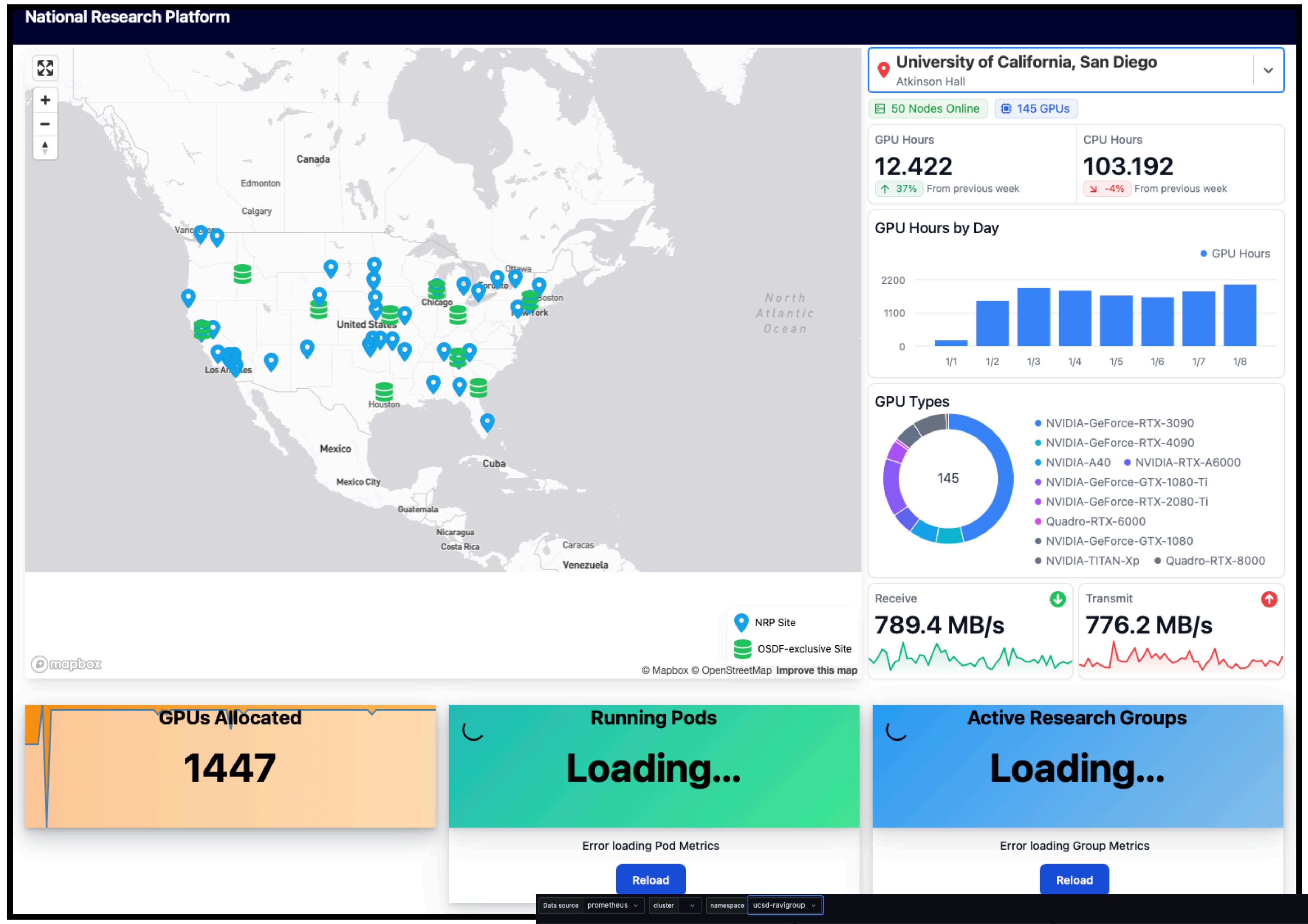
- ***Disclaimer:*** *this is what I was doing up to 1.5 years ago → work ongoing to replicate it at Cineca/Leonardo*

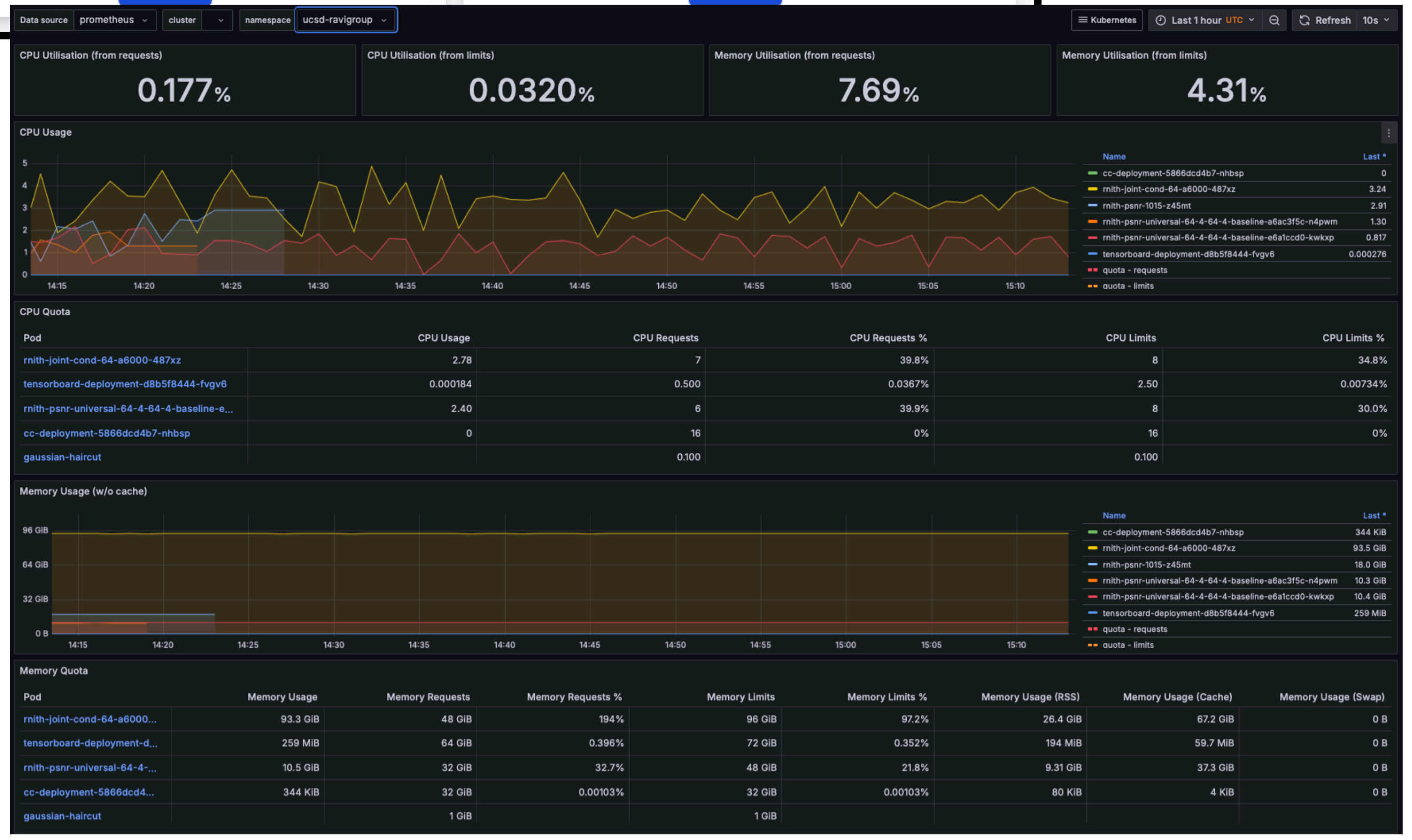
National research platform (NRP)

- **US network of computing cent**ers lead by UCSD **[Link]**
- Sites are linked together under a common **Kubernetes cluster**
- **GPUs available ~ 1.5k**
- Five **CEPH** clusters with S3 access allowing for **O(10) Pb storage**
- **Typical machine**: O(128) CPUs, O(540) Gb RAM, 5 TB SSD, 8 GPU
- **Description of cluster and its use:**

  https://docs.nationalresearchplatform.org/

- **Monitor** job / pod resources via **Grafana webpages**





Running over a Kubernetes cluster

- ***Disclaimer:*** *I am old enough to avoid modern solutions like jupyter-notebooks … thus I still work with python scripts and use manual job submission / description*

Training environment → docker image

- **Docker image:** it starts from default centos/RHEL/ ubuntu + cuda images from NVIDIA
- **Additional software:** git, rsync, krb5, + xrootd (in case I/O with CERN is needed)
- **Conda env** with packages needed by weaver-core
- **Docker file [Link]**
- **Docker image uploaded to gitlab registry** of a repository interfaced with the cluster
- Commands needed to create and upload the docker image from a local computer **[Link]**

Persistent volume and data transfer from CERN

- Persistent volume of O(2.5) TB on the CEPH cluster [Link]
- Generate CERN kerberos keytab and upload as KUBE secret
- Copy files without pwd in the KUBE job [Link]

Example of training job submission

- weaver-core not included in the docker image: it is pulled from github at job starting → allows to update packages
- weaver train.py script takes several input parameters to customise the job
- Training submission command [Link]

Running over a Kubernetes cluster: example

- ***Kubectl config YAML file*** is made by several sections

job name and docker container to pull

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: weaver-job-ak4-transformer-cont-ch
spec:
  template:
     spec:
       containers:
        - name: process
          image: gitlab-registry.nrp-nautilus.io/rgerosa/particlenetrun2ul/weaver_cuda121_torch212_alma9:latest
          command: ["/bin/sh","-c"]
```

node affinity: type of hardware, etc. → 48 Gb GPUs

```yaml
        affinity:
          nodeAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
              nodeSelectorTerms:
               - matchExpressions:
                 - key: nvidia.com/gpu.product
                   operator: In
                   values:
                   - NVIDIA-A40
                   - NVIDIA-RTX-A6000
                   - Quadro-RTX-8000
                   - NVIDIA-L40
                   - NVIDIA-A100-PCIE-40GB
```

job resources, volumes, shared memory

```yaml
           volumeMounts:
           - mountPath: /mnt/weavervolume
             name: weavervolume
           - mountPath: /dev/shm
             name: dshm
           resources:
            limits:
             nvidia.com/gpu: 1
             memory: 72Gi
             cpu: 18
            requests:
             nvidia.com/gpu: 1
             memory: 72Gi
             cpu: 18
        volumes:
        - name: weavervolume
          persistentVolumeClaim:
           claimName: weavervolume
        - name: dshm
          emptyDir:
           medium: Memory
           sizeLimit: "32Gi"
```

args section of the config express the commands to be executed

https://gitlab.nrp-nautilus.io/rgerosa/particlenetrun2ul/-/blob/main/ak4_training_latest/config_1gpu_48gb/weaver-job-ak4-transformer-cont-ch.yaml?ref_type=heads#L12-L53

Final results: heavy flavour tagging

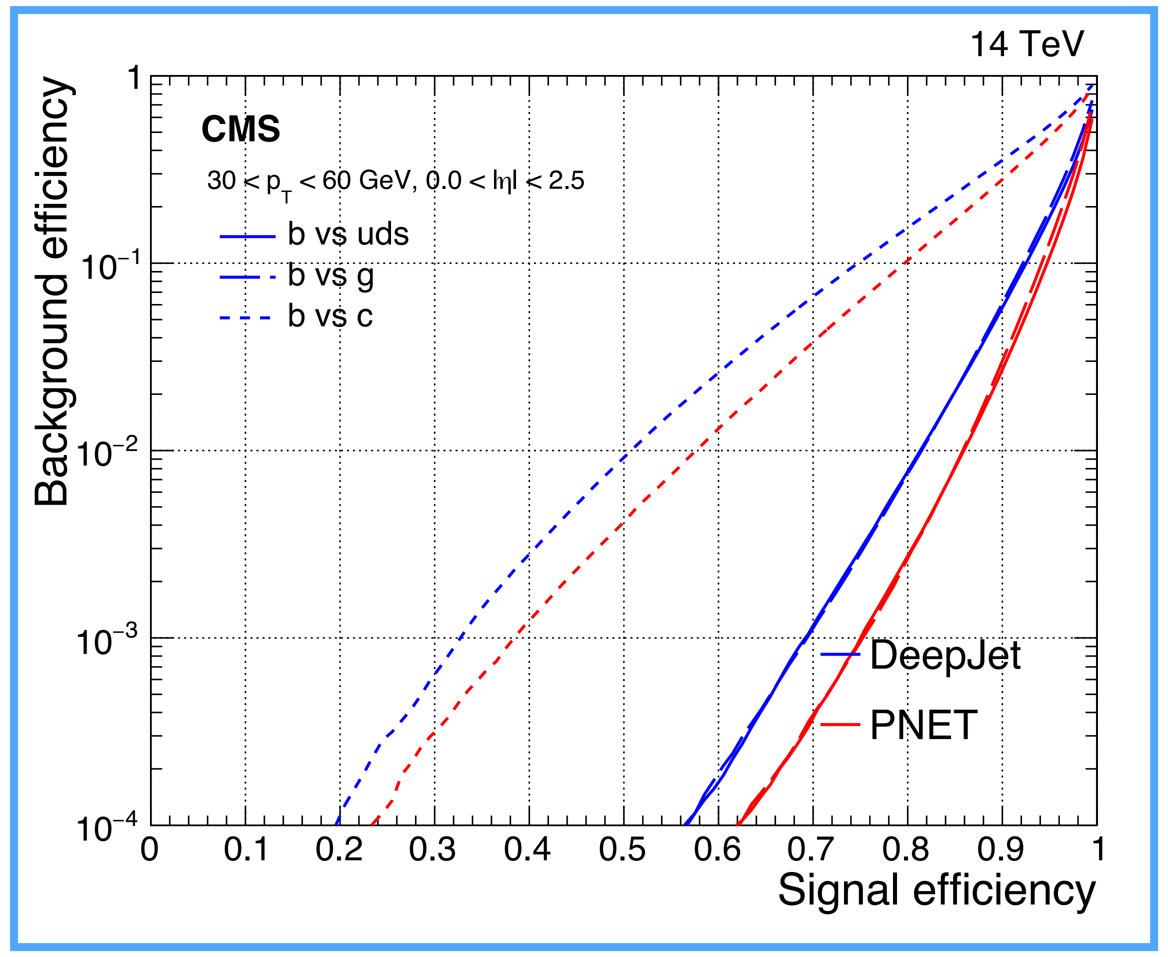
Disclaimer: *in the plots shown, performance is not necessarily the best to-date → they are meant to provide a glance of the performance*

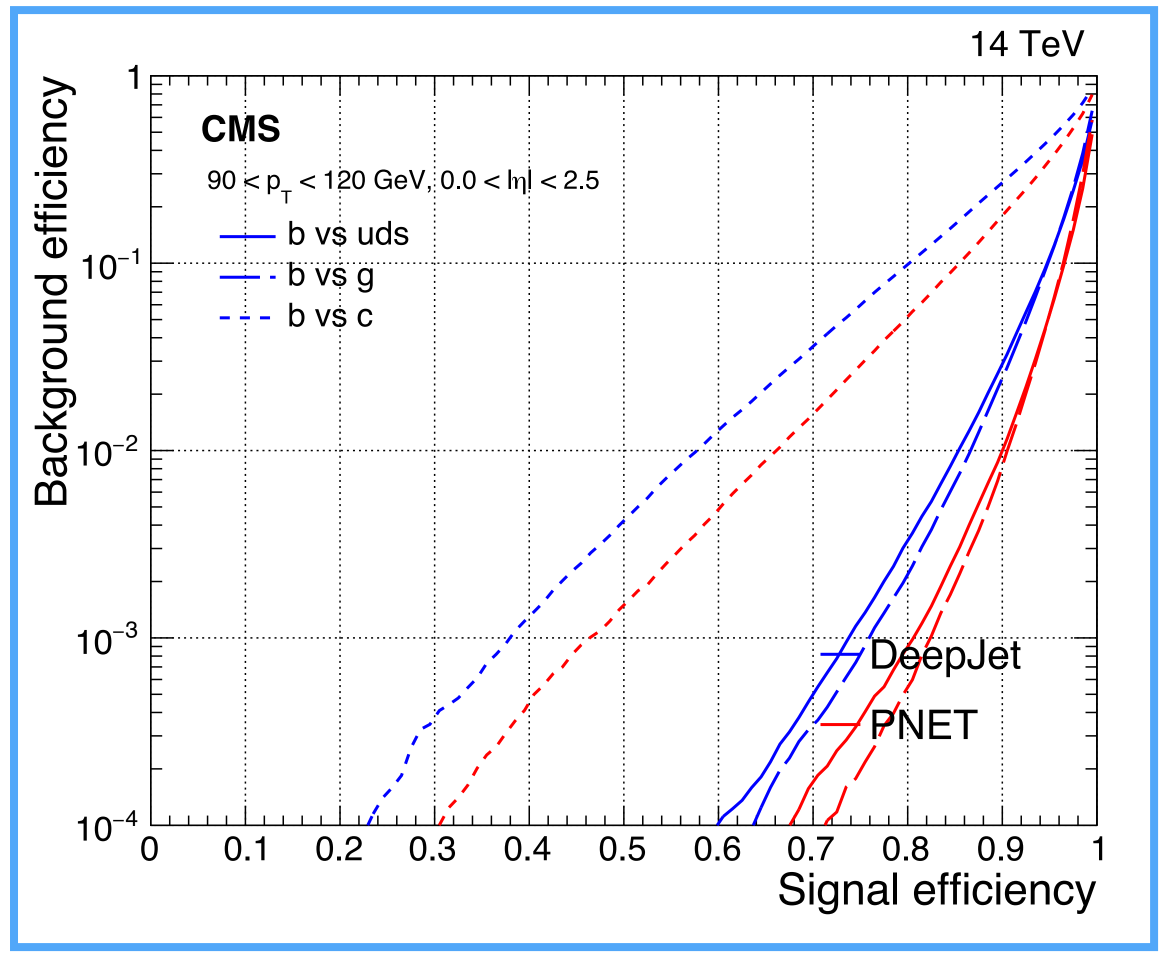
Bkg. efficiency

P(b vs c) = P(b)/(P(b)+P(c))

P(b vs uds) = P(b)/(P(b)+P(uds))

P(b vs g) = P(b)/(P(b)+P(g))





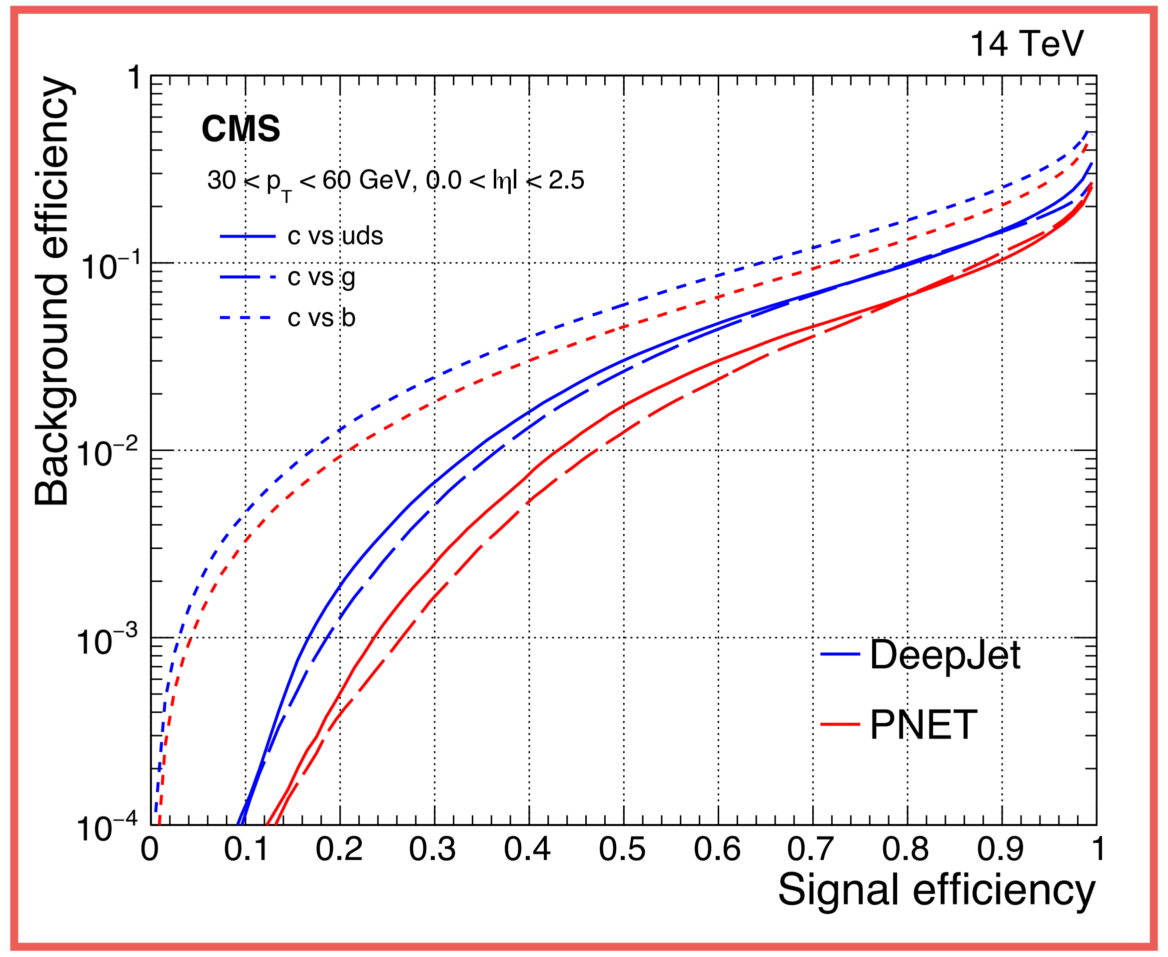
Tagging performance improves with p_T and better in central barrel

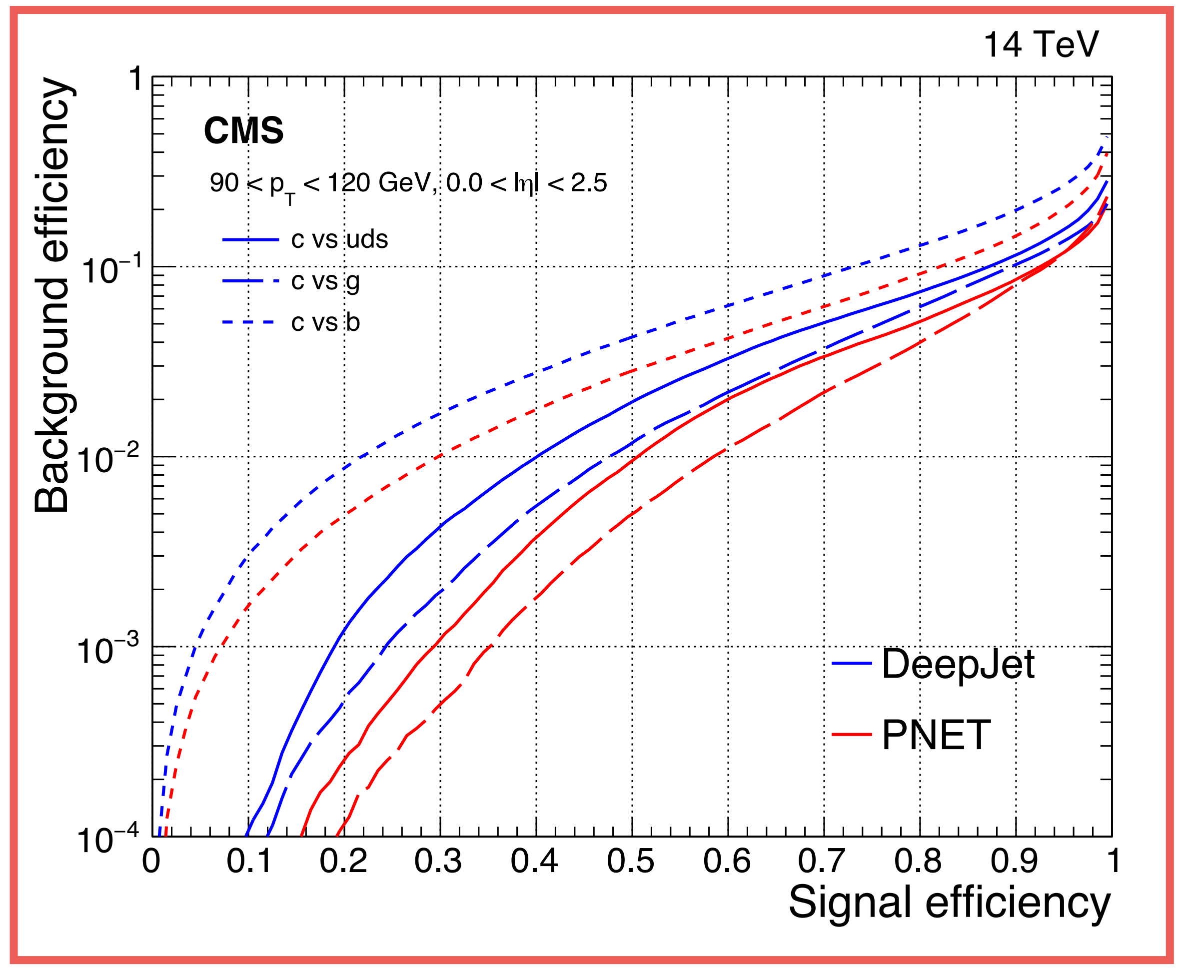
Bkg. efficiency

P(c vs b) = P(c)/(P(c)+P(b))

P(c vs uds) = P(c)/(P(c)+P(uds))

P(c vs g) = P(c)/(P(c)+P(g))





Final results: hadronic τ_h

Disclaimer: *in the plots shown, performance is not necessarily the best to-date → they are meant to provide a glance of the performance*

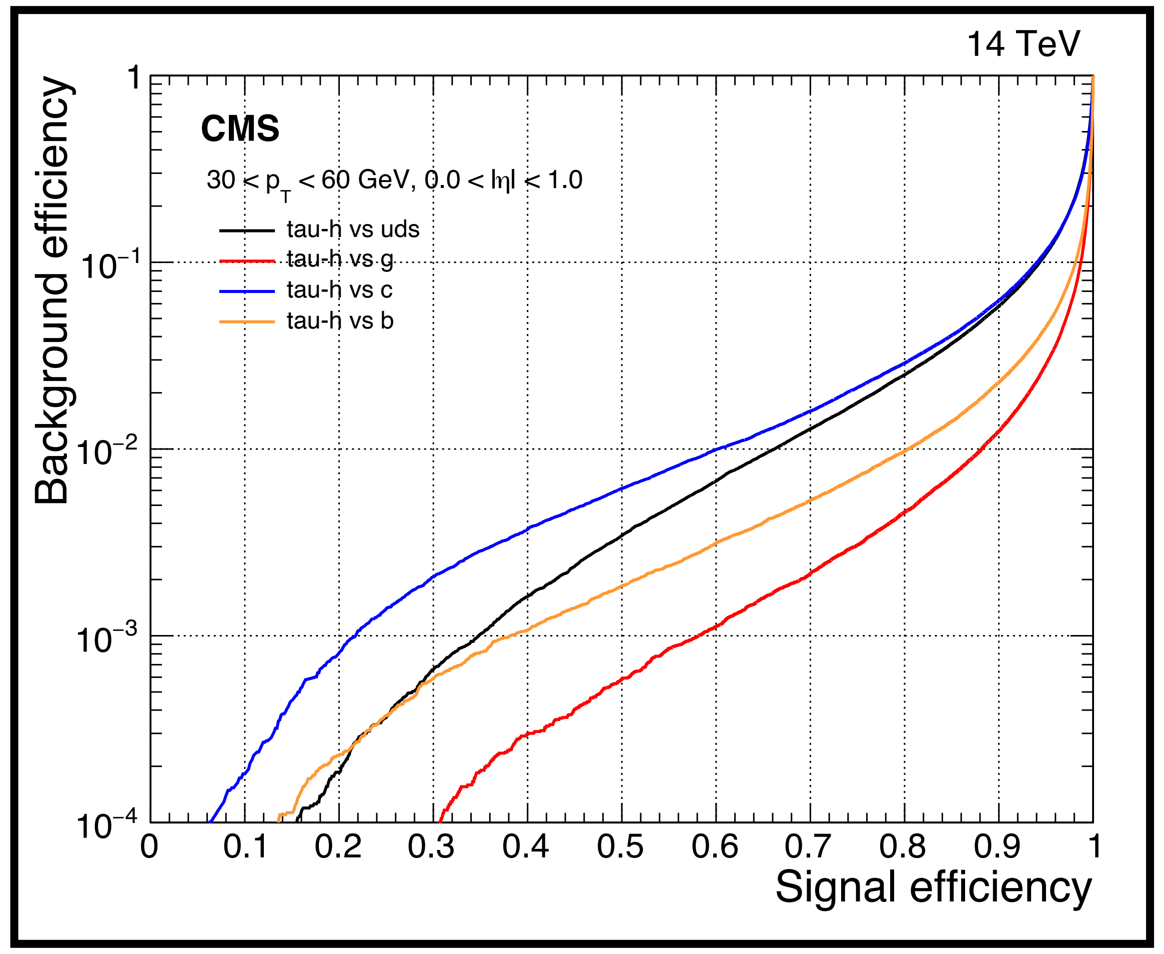
P(τ_h vs c) = P(τ_h)/(P(τ_h)+P(c))

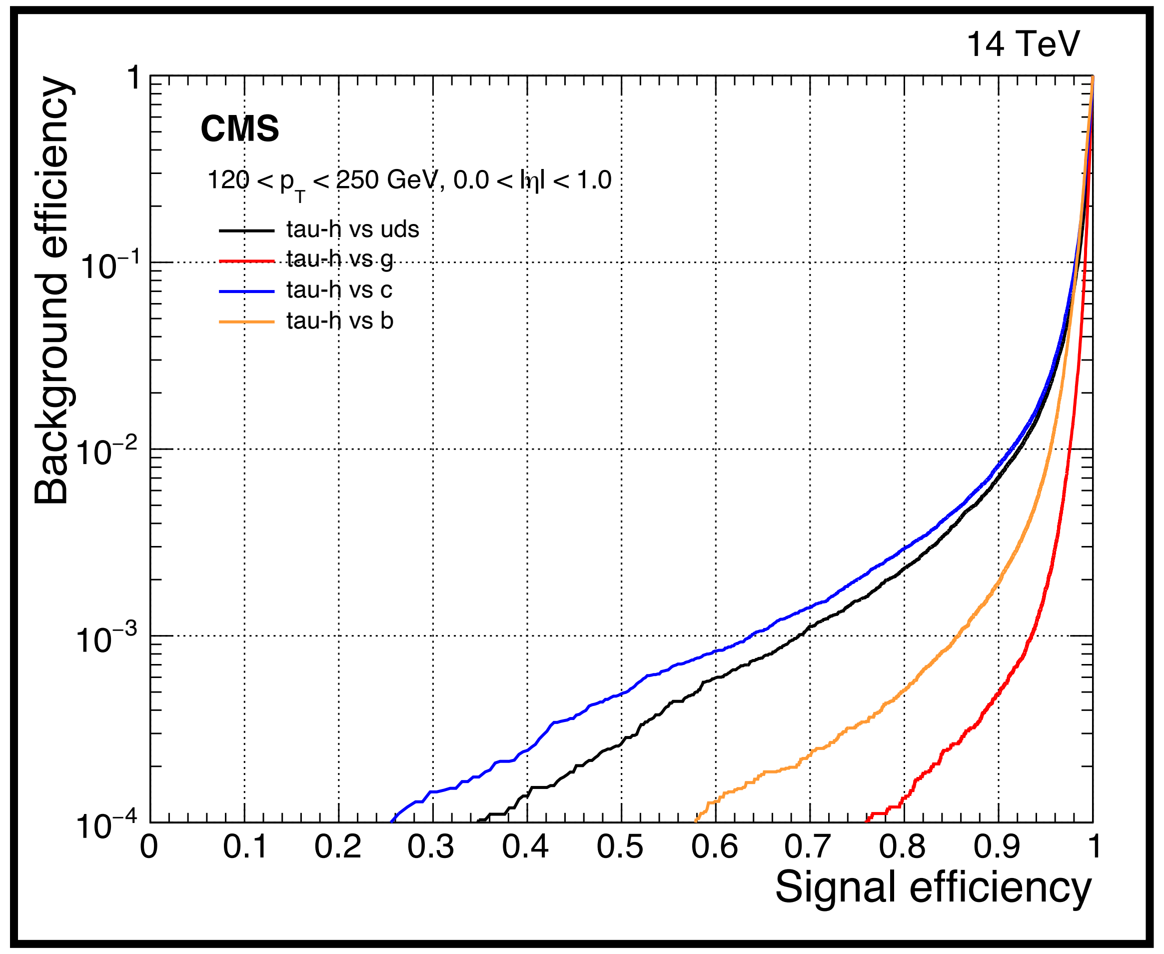
P(τ_h vs uds) = P(τ_h)/(P(τ_h)+P(uds))

P(τ_h vs b) = P(τ_h)/(P(τ_h)+P(b))

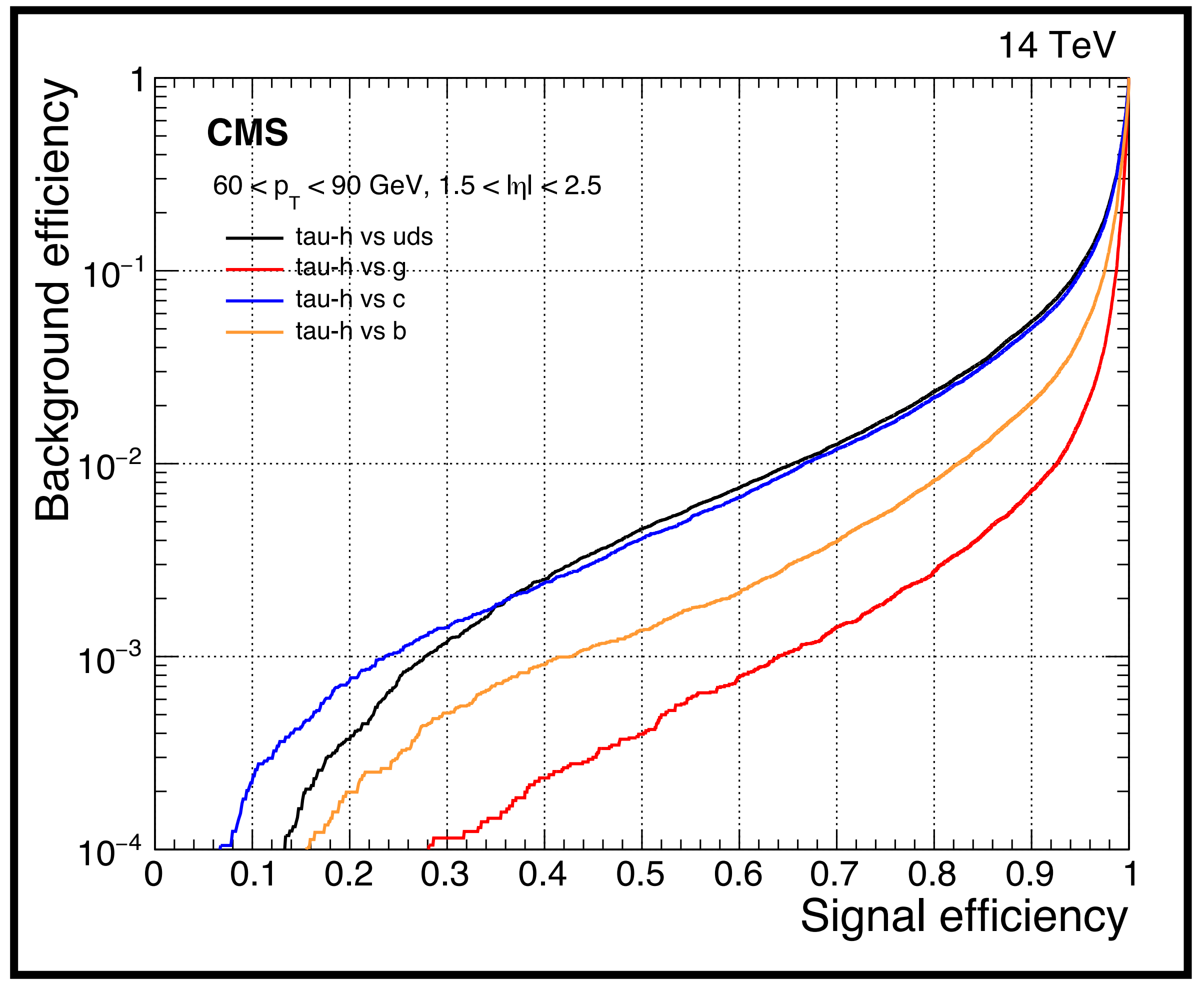
P(τ_h vs g) = P(τ_h)/(P(τ_h)+Pg))

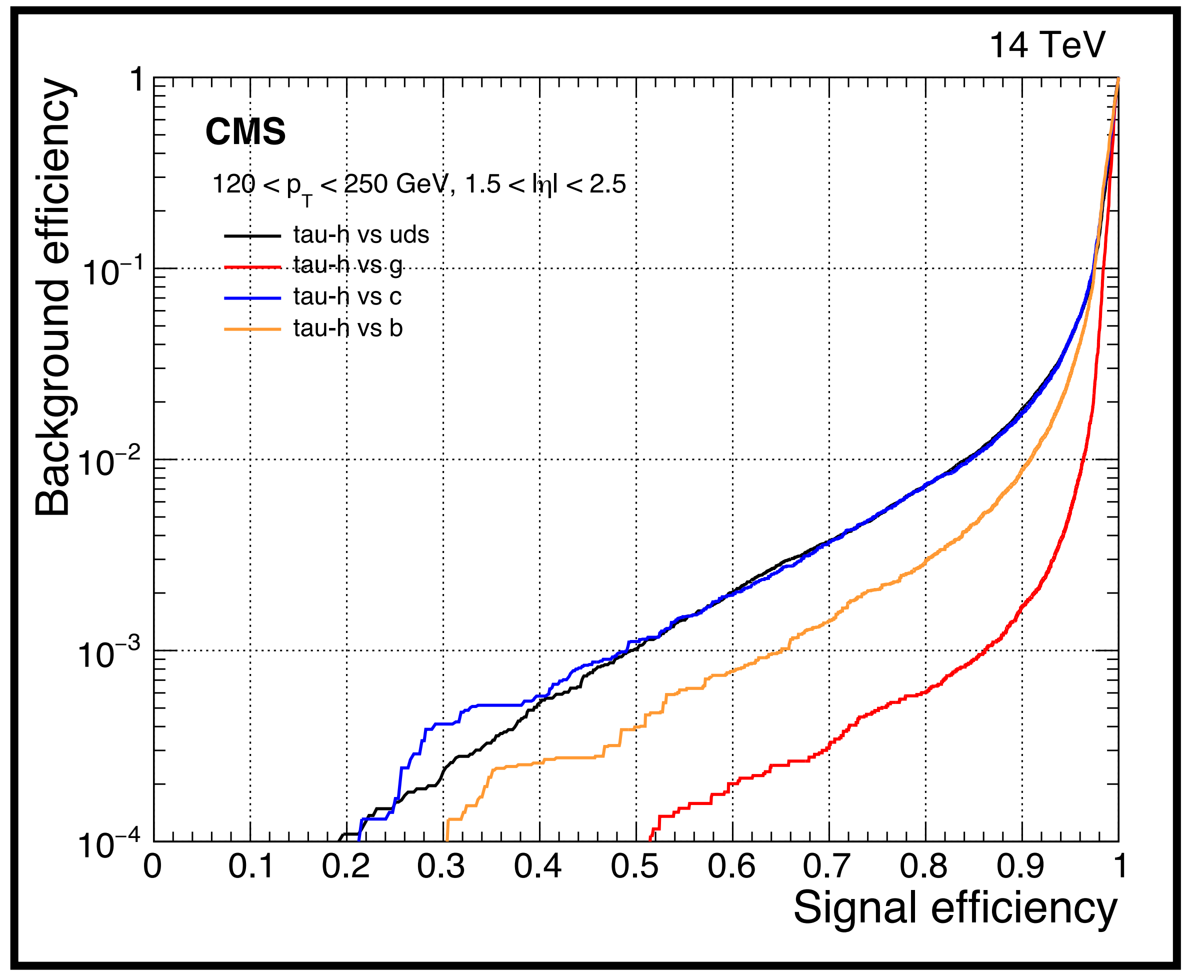
Bkg. efficiency





Tagging performance improves with p_T and better in central barrel





ParT show the capability of being:
- **Better** than DeepTau on **ID**
- **Better** than HPS in **decay mode assignment**
- **Recover inefficiencies** in HPS reconstruction

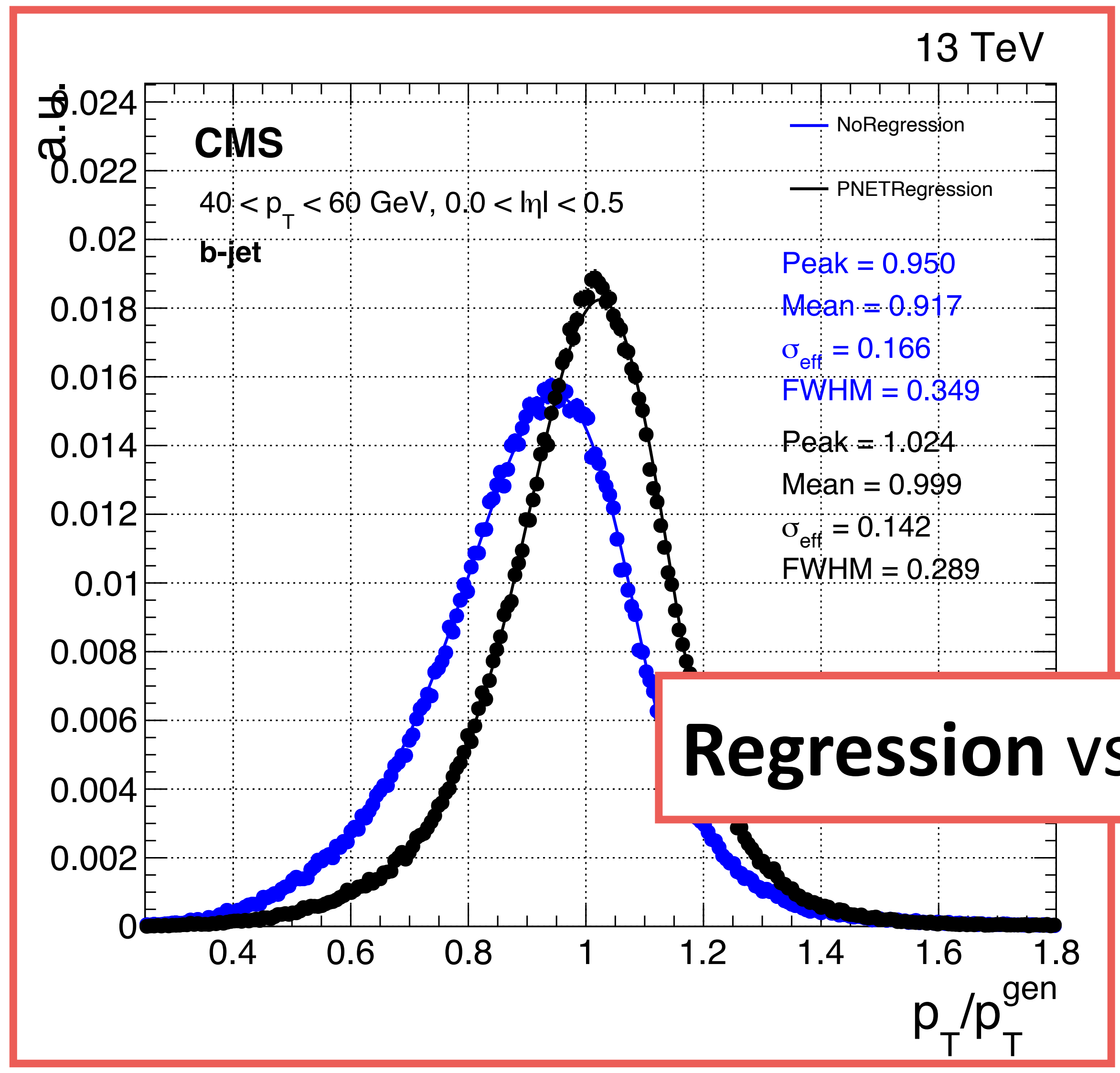
Final results: jet p_T regression and per-jet resolution

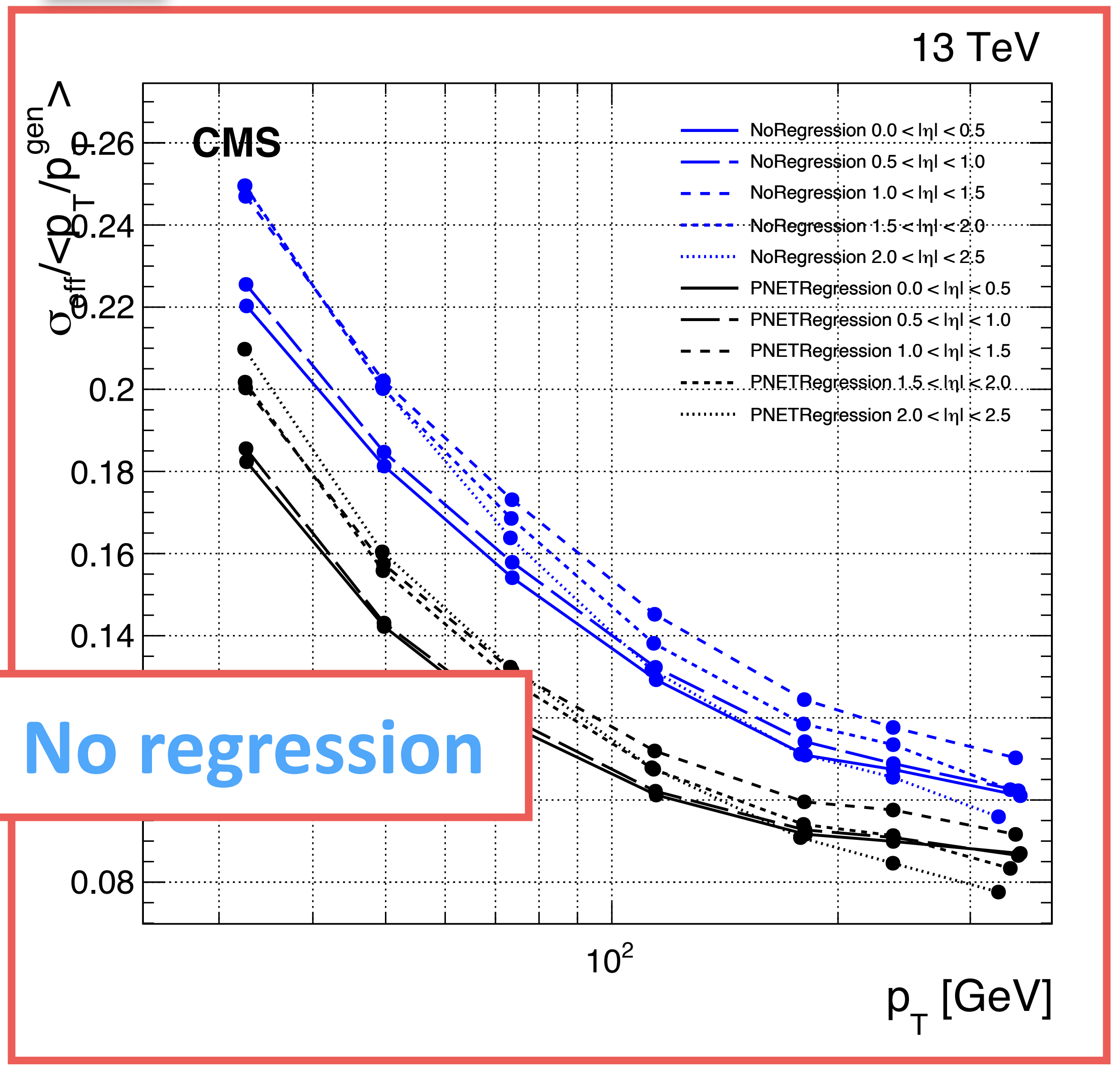
Disclaimer: *this was a rather experimental task I was trying … a kind of promising but not in production .. showing it just for fun*

- Implemented without chaining the network layout but by **adding a regression term** to the **loss-function**

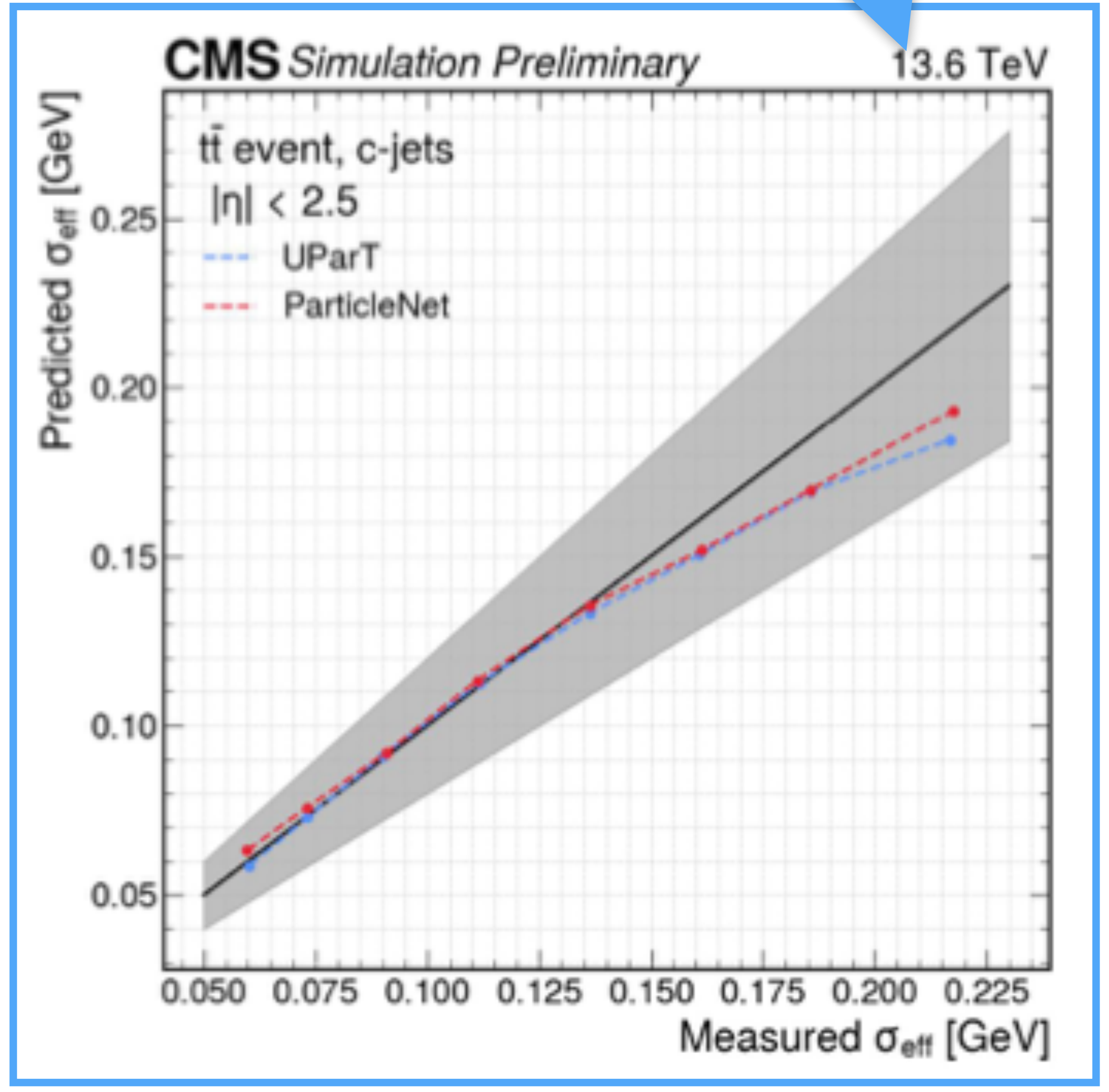
$$L = \boxed{\mathrm{CrossEntropy}(\mathrm{x}, \mathrm{x_{truth}})} + \boxed{\lambda \times \log(\cosh(y - y_{target}))} + \boxed{\gamma \times [\rho_{0.16}(\mathrm{z} - \mathrm{y_{target}}) + \rho_{0.84}(\mathrm{k} - \mathrm{y_{target}})]}$$

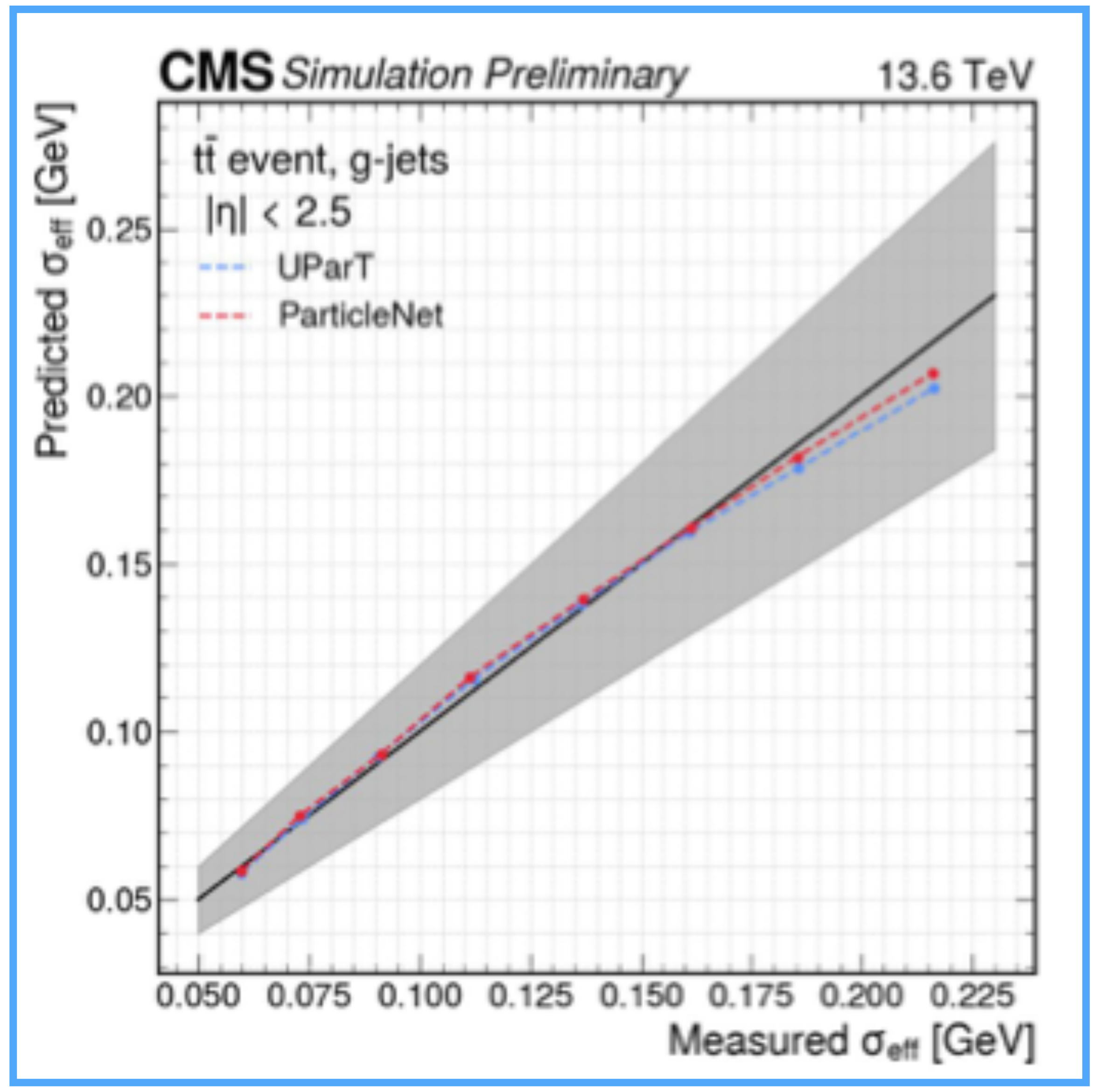
jet-tagging *jet p_T regression* *jet p_T resolution estimator*





Regression vs **No regression**

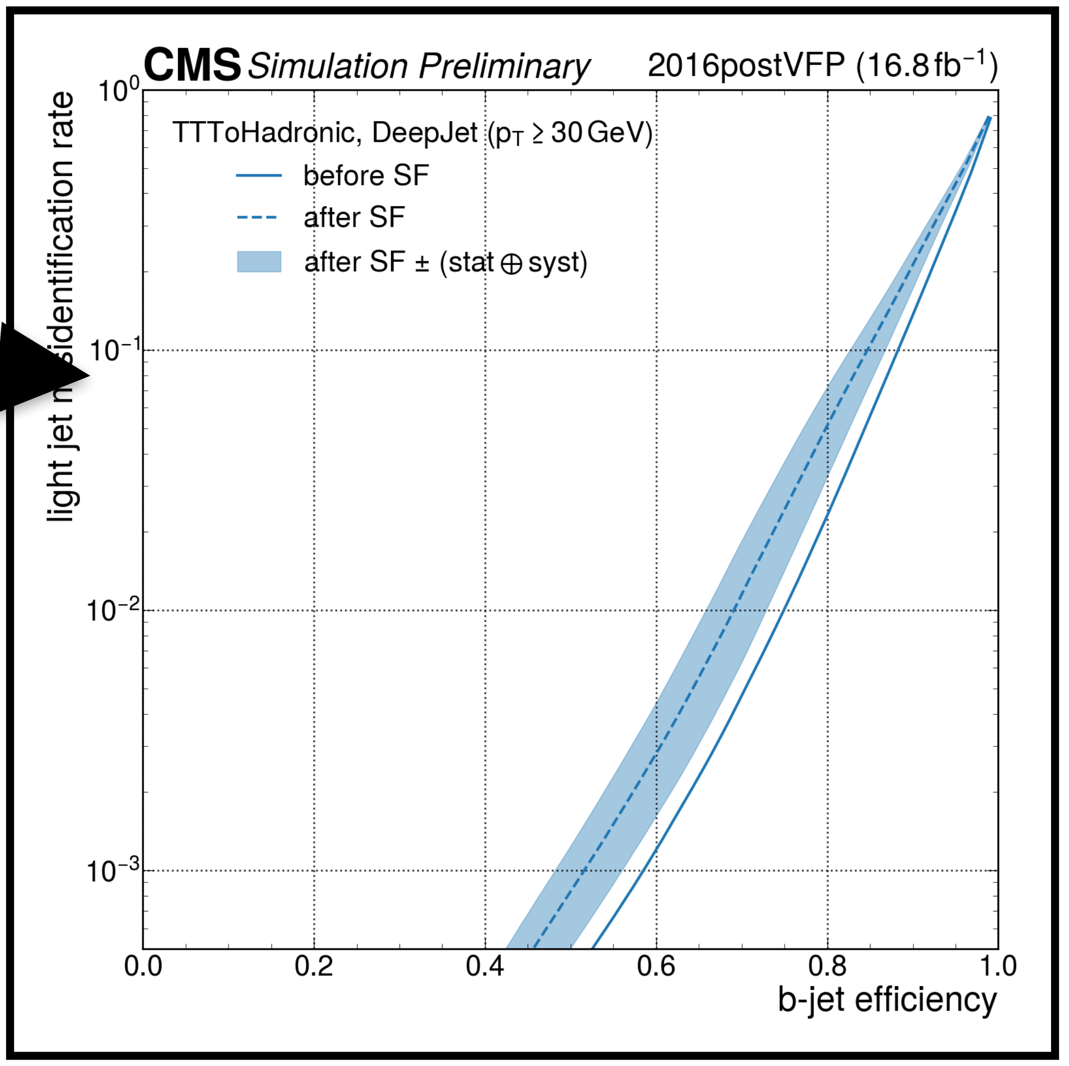




Final results: improve description with data-domain

Disclaimer: *this was a rather experimental task I was trying … a kind of promising but not in production .. showing it just for fun*

- **Jet tagging outputs** are **calibrated in-situ** with data using **data control regions (CRs)**
- Known as **b/c-tagging** efficiency **scale factors** → measured via **iterative simultaneous fit**
- **SFs** can be significantly different from unity → **impact on analysis performance**



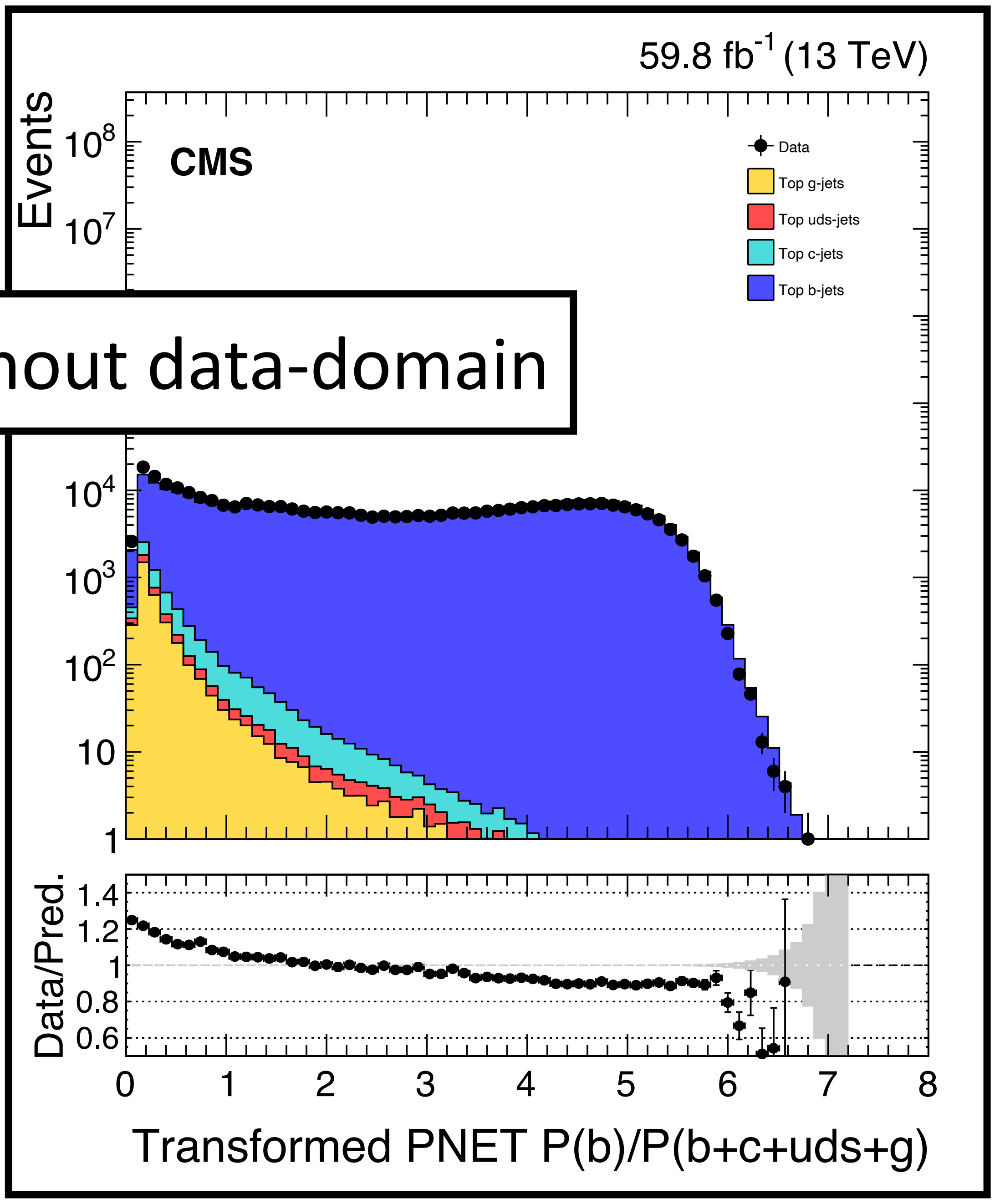
Final results: improve description with data-domain

Disclaimer: *this was a rather experimental task I was trying … a kind of promising but not in production .. showing it just for fun*

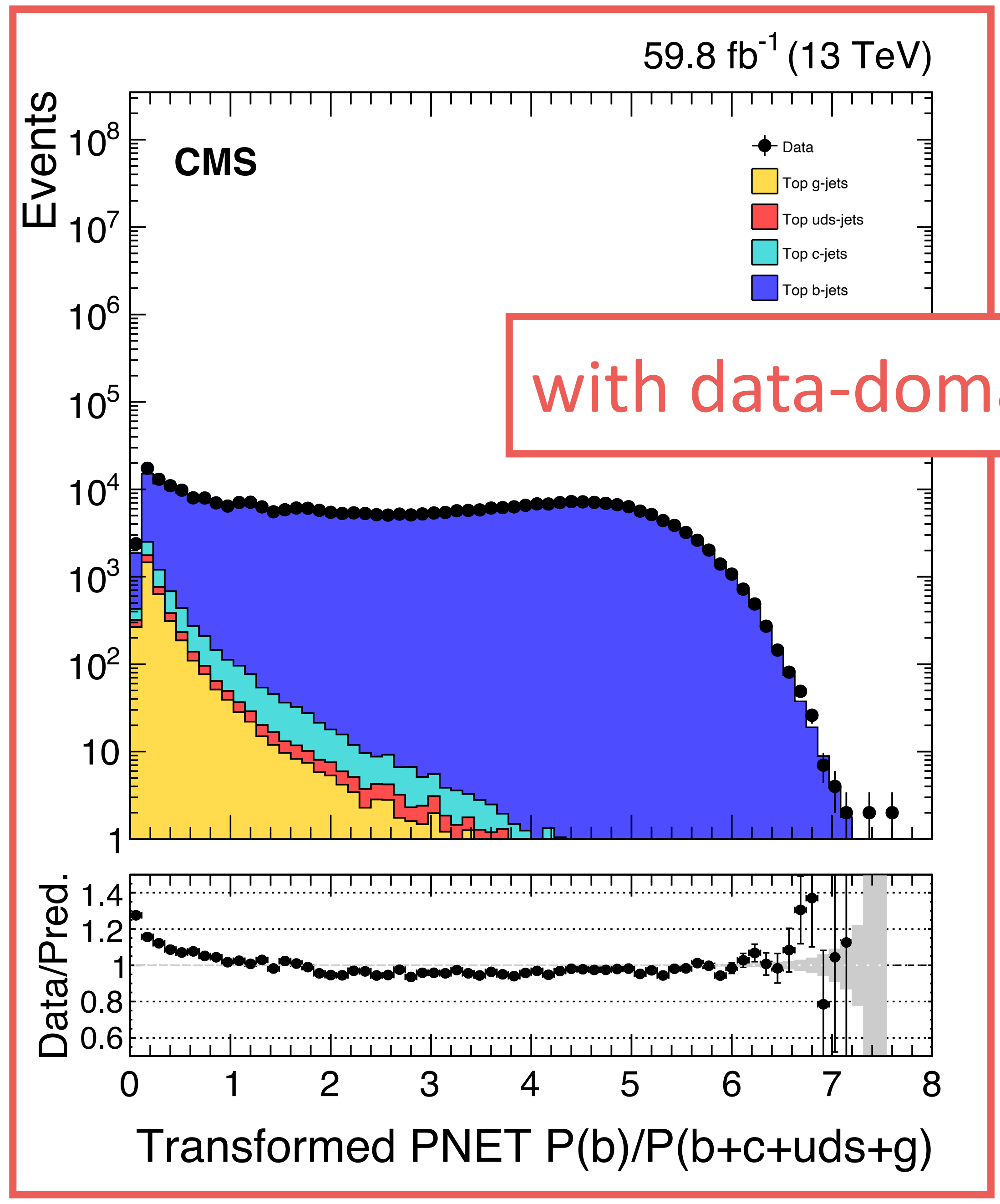
- **Jet tagging outputs** are **calibrated in-situ** with data using **data control regions (CRs)**
- Known as **b/c-tagging** efficiency **scale factors** → measured via **iterative simultaneous fit**
- **SFs** can be significantly different from unity → **impact on analysis performance**

Data-domain adaptation

- Train a first model solely on simulation
- Select data and simulation events from CRs that are "pure" in a certain class of jet
- Train the network again together with a binary classifier in each CR that distinguishes data from simulation
- Invert gradient to penalize the loss function when data are distinguished from MC → domain adaptation
- Example CRs: ttbar(eμ), DY($\mu\tau_h$), Z(μμ), dijet



without data-domain



with data-domain

$$L = L_{class} + \lambda \cdot L_{reg} + \eta \cdot L_{qnt} + k \cdot (L_{\mu\mu} + L_{e\mu} + L_{\mu\tau} + L_{dijet} + L_{tt+c})$$

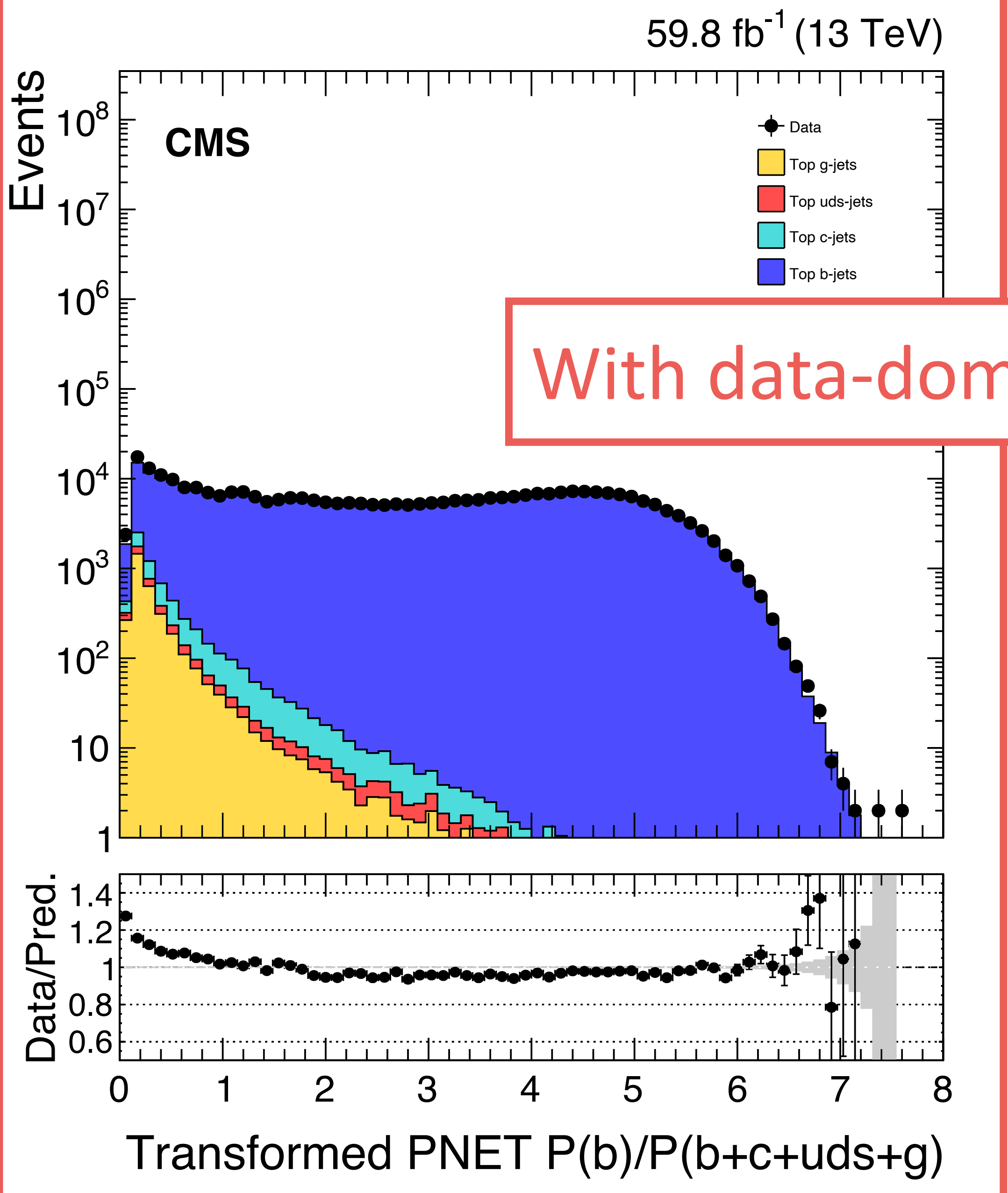
Final results: improve description with data-domain

Disclaimer: *this was a rather experimental task I was trying … a kind of promising but not in production .. showing it just for fun*

- **Jet tagging outputs** are **calibrated in-situ** with data using **data control regions (CRs)**
- Known as **b/c-tagging** efficiency **scale factors** → measured via **iterative simultaneous fit**
- **SFs** can be significantly different from unity → **impact on analysis performance**

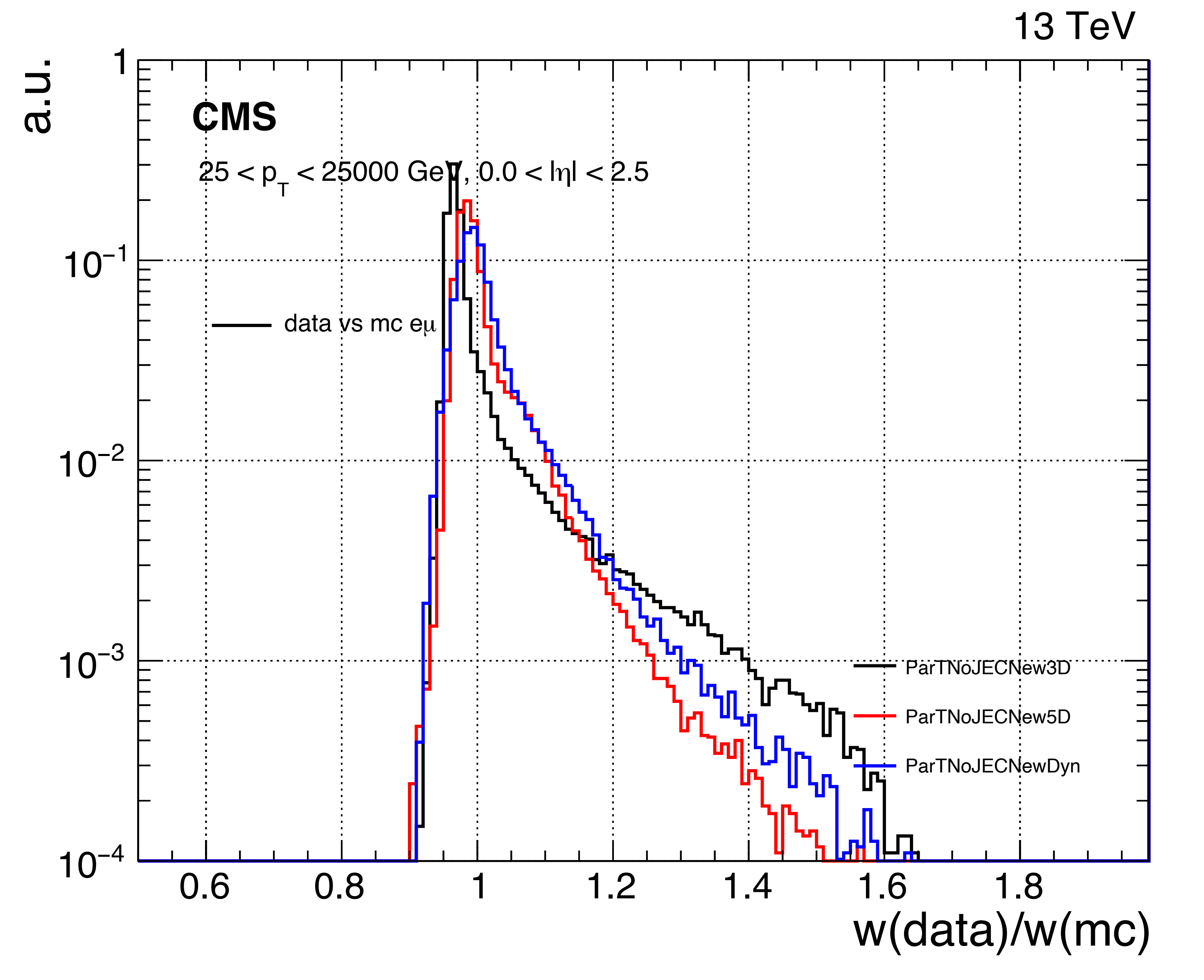
Data-domain adaptation

- Train a first model solely on simulation
- Select data and simulation events from CRs that are "pure" in a certain class of jet
- Train the network again together with a binary classifier in each CR that distinguishes data from simulation
- Invert gradient to penalize the loss function when data are distinguished from MC → domain adaptation
- Example CRs: ttbar(eμ), DY(μτ_h), Z(μμ), dijet



With data-domain

In inference for every MC event P(data)/P(MC) can be computed giving an unbinned SF proxy



Conclusions and outlooks

- Availability of high performance analysis facilities will be crucial for the development of our field towards HL-LHC and FCC
- **Today I focused on a single use case** about data-preparation, ML training, and inference for **jet-tagging**
- Execution of the task from A to Z without R&D requires:

  - O(5) days for dataset preparation starting from MiniAOD
  - O(1) week for training + testing on NRP
  - O(5) days for validation in data from MiniAOD

- With improved workflows for high rate data preparation and a validation, as well as improved pipelines and hardware for ML, ***a large speedup and a more efficiency use of resources is certainly possible***
- ***This will immensely help CMS POGs that are short in person-power, dedicated analysis facilities and connected resources***
- *If some of you is interest in helping to test and automatise a procedure to run such kind of task with INFN resource contact me!*

backup slides

Ingredients prior to jet-tagging

Jet defintion aka jet clustering

- **Clustering** is an **unsupervised task** as we don't know what the result will be before running it
- **Clustering** aggregates **particles** via a logic based on a **distance (metric)** between particles or their clusters
- **Clustering** is **an-iterative procedure** that produces recombinations **until a condition in satisfied**
- **Jets** serve two purposes: **(a) observables** that one can predict and measure, (b) **tools** to extract properties of a final state → constraints are **infrared** and **ultra-violet safety**
- The **anti-k_T jet clustering** takes just one parameter called R distance that defines nearest neighbours of each particle

$$d_{ij} = min(\frac{1}{p^2_{T,i}}, \frac{1}{p^2_{T,j}}) \times \Delta R^2/R^2$$

Ingredients prior to jet-tagging

Jet defintion aka jet clustering

- **Clustering** is an **unsupervised task** as we don't know what the result will be before running it
- **Clustering** aggregates **particles** via a logic based on a **distance (metric)** between particles or their clusters
- **Clustering** is **an-iterative procedure** tha… recombinations **until a conditio**…
- **Jets** serve two pu… **…ervables** that one can predict a… , **tools** to extract properties of a fi… …traints are **infrared** and **ultra-violet safety**
- The **anti-k_T jet clustering** takes just one parameter called R distance that defines nearest neighbours of each particle

$$d_{ij} = min(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2}) \times \Delta R^2/R^2$$

But we need input particles to cluster

Ingredients prior to jet-tagging

Jet defintion aka jet clustering

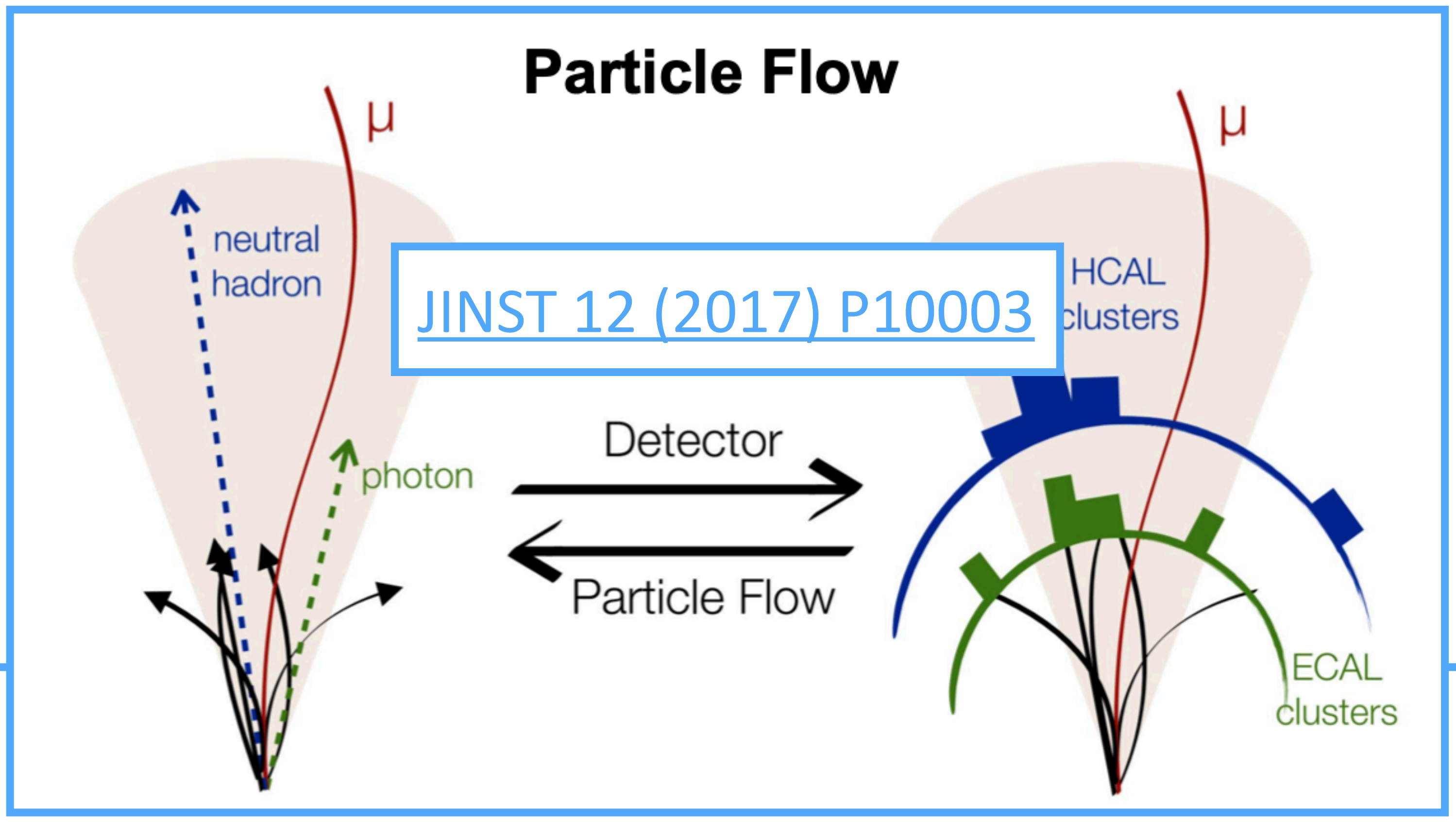
- **Clustering** is an **unsupervised task** as we don't know what the result will be before running it
- **Clustering** aggregates **particles** via a logic based on a **distance (metric)** between particles or their clusters
- **Clustering** is **an-iterative procedure** tha… recombinations **until a conditio…**
- **Jets** serve two pu… …**ervables** that one can predict an… …**tools** to extract properties of a fi… …traints are **infrared** and **ultra-violet safety**
- The **anti-k_T jet clustering** takes just one parameter called R distance that defines nearest neighbours of each particle

$$d_{ij} = min(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2}) \times \Delta R^2/R^2$$

But we need input particles to cluster

Particle-flow reconstruction

- Particle-flow (PF) provides an improved event description by correlating single detector measurements
- PF-candidates in offline reconstruction:
  - Leptons (muons and electrons) and photons
  - Charged hadrons: everything interpreted as pion ($\pi^{\pm}$)
  - Neutral hadrons (neutrons, K_L^0)
- Useful info outside PF-candidate collection:
  - Tracks not linked to any PF-object → lost tracks
  - V_0 candidates: K_S and lambdas



JINST 12 (2017) P10003

Ingredients prior to jet-tagging

Jet defintion aka jet clustering

- **Clustering** is an **unsupervised task** as we don't know what the result will be before running it
- **Clustering** aggregates **particles** via a logic based on a **distance (metric)** between particles or their clusters
- **Clustering** is **an-iterative procedure** tha… recombinations **until a conditio…**
- **Jets** serve two pu… …ervables that one can predict an… …, **tools** to extract properties of a fi… …straints are **infrared** and **ultra-violet safety**
- The **anti-k$_T$ jet clustering** takes just one parameter called R distance that defines nearest neighbours of each particle

$$d_{ij} = min(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2}) \times \Delta R^2/R^2$$

But we need input particles to cluster

Particle-flow reconstruction

- Particle-flow (PF) provides an improved event description by correlating single detector measurements
- PF-candidates in offline reconstruction:
  - Leptons (muons and electrons) and …
  - Charged hadrons: everythi…
  - Neutral hadrons (…
- Useful in… …llection:
  - … any PF-object → lost tracks
  - …idates: K$_S$ and lambdas

Please forget about NanoAODs … you need lower level inputs in MiniAODs