JUNIPR:
a framework for unsupervised learning in jet physics

Anders Andreassen
aja@lbl.gov

in collaboration with
Feige, Frye and Schwartz

arXiv: 1804.09720

BERKELEY LAB

Berkeley
UNIVERSITY OF CALIFORNIA

VE RI TAS
HARVARD

JUNIPR Motivation

How can we understand jets better using machine learning?

Idea: (1) Let neural network learn about jets
(2) Look inside to see what it's doing ← challenging!

JUNIPR Motivation

How can we understand jets better using machine learning?

Idea: (1) Let neural network learn about jets
(2) Look inside to see what it's doing ← challenging!

Our strategy:

- use network architecture inspired by QCD shower
- but general enough to fit any non-QCD structure

JUNIPR Motivation

How can we understand jets better using machine learning?

Idea: (1) Let neural network learn about jets
(2) Look inside to see what it's doing ← challenging!

Our strategy:

- use network architecture inspired by QCD shower
- but general enough to fit any non-QCD structure

RNN built around a clustering tree $\Longrightarrow$ interpret output from intermediate layers!

JUNIPR:

Jets using UNsupervised Interpretable PRobabilistic models

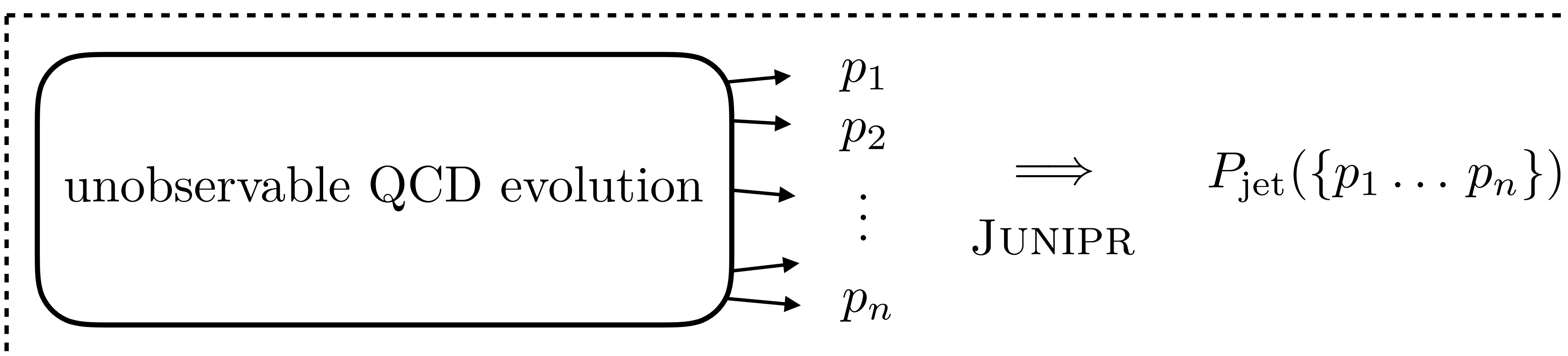
Andreassen, Feige, Frye, Schwartz (2018)

a Probabilistic Model for Jets

- given training data, JUNIPR learns probability distribution over constituent momenta in individual jets

- proportional to totally differential cross section:

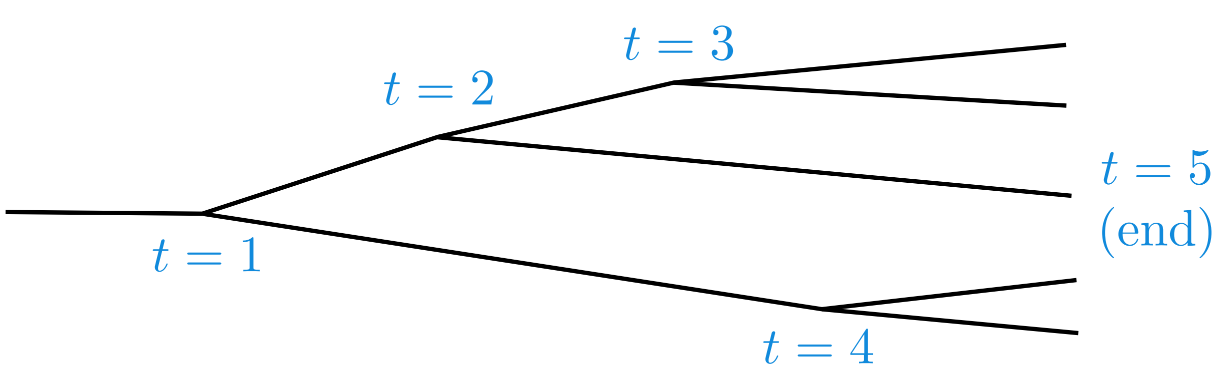
$$P_{\text{jet}}(\{p_1 \ldots p_n\}) \sim \frac{d\sigma}{d^3p_1 \cdots d^3p_n}$$



Reducing Complexity with Clustering Trees

- with ~30 particles in a jet, $P_{\text{jet}}(\{p_1 \ldots p_n\})$ is ~100 dimensional!

- break into product over "time" steps in clustering tree;
  each 4-momentum conserving branching is only 4-dimensional



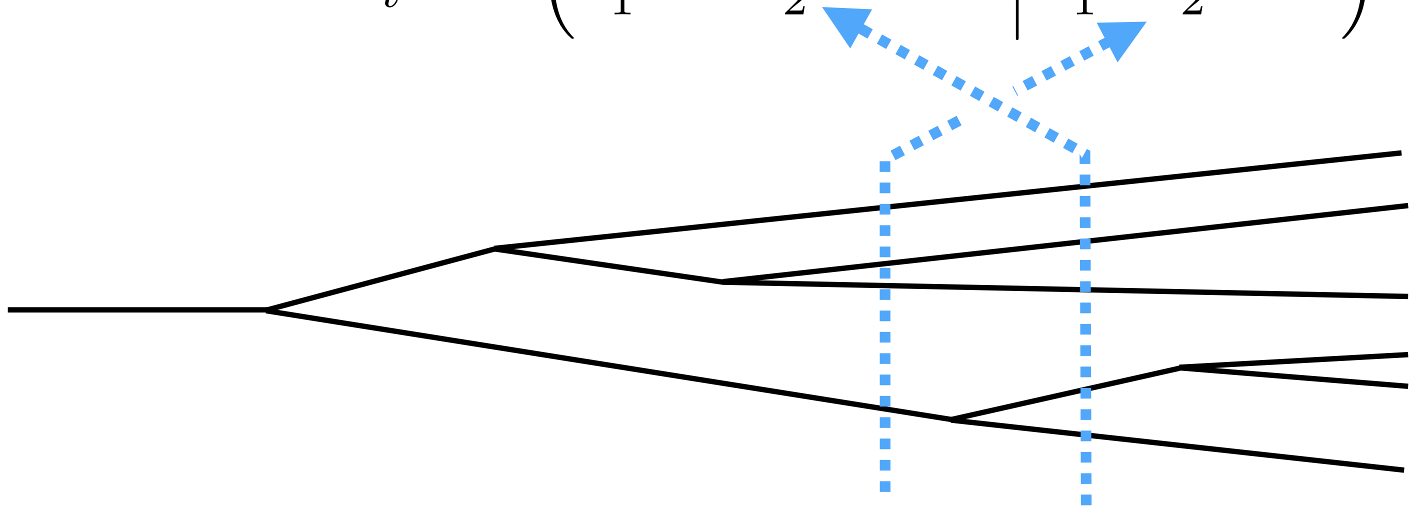
$$\underset{\textsc{Junipr}}{\Longrightarrow} \qquad P_{\text{jet}}(\{p_1 \ldots p_n\}) = P_{t=1} \cdots P_{t=n}$$

- model structured on leading-order description of underlying physics
  - increases efficiency, reduces complexity
  - leads to interpretability!

General Form of Probabilistic Model

- decomposition into time steps $$P_{\text{jet}}(\{p_1 \dots p_n\}) = \prod_{t=1}^{n} P_t$$

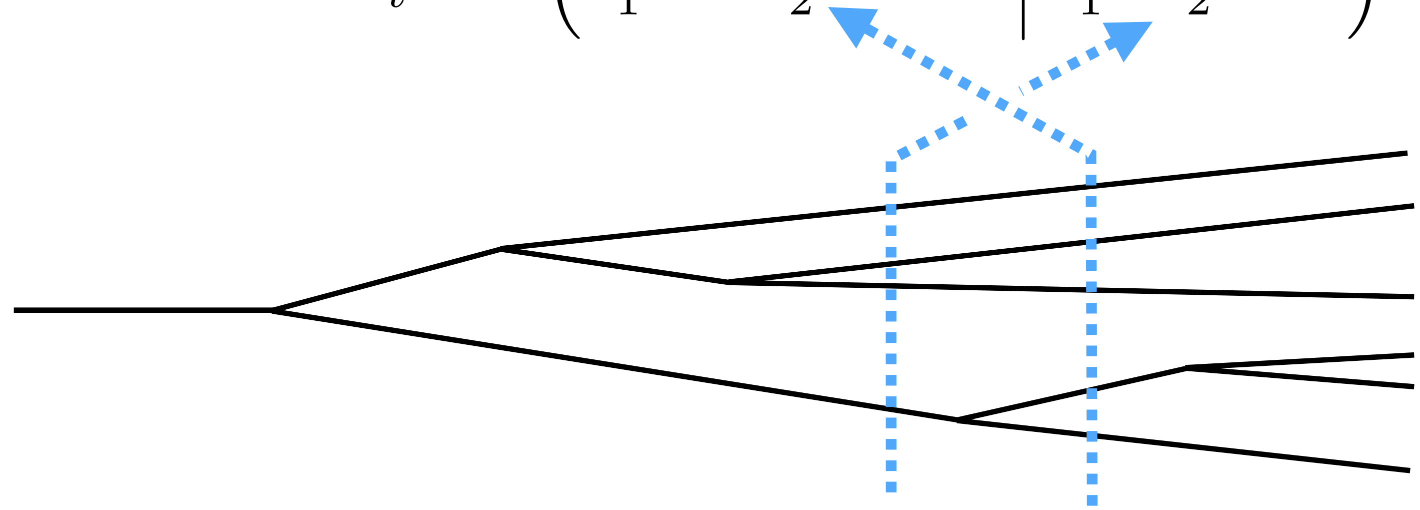
where $$P_t = P\left(k_1^{(t+1)}\, k_2^{(t+1)} \cdots \,\middle|\, k_1^{(t)}\, k_2^{(t)} \cdots\right)$$



General Form of Probabilistic Model

- decomposition into time steps $$P_{\text{jet}}(\{p_1 \dots p_n\}) = \prod_{t=1}^{n} P_t$$

where $$P_t = P\Big(k_1^{(t+1)}\, k_2^{(t+1)} \cdots \Big|\, k_1^{(t)}\, k_2^{(t)} \cdots\Big)$$



- further simplification: only $k_m^{(t)} \to k_{d_1}^{(t+1)}\, k_{d_2}^{(t+1)}$ at each time step

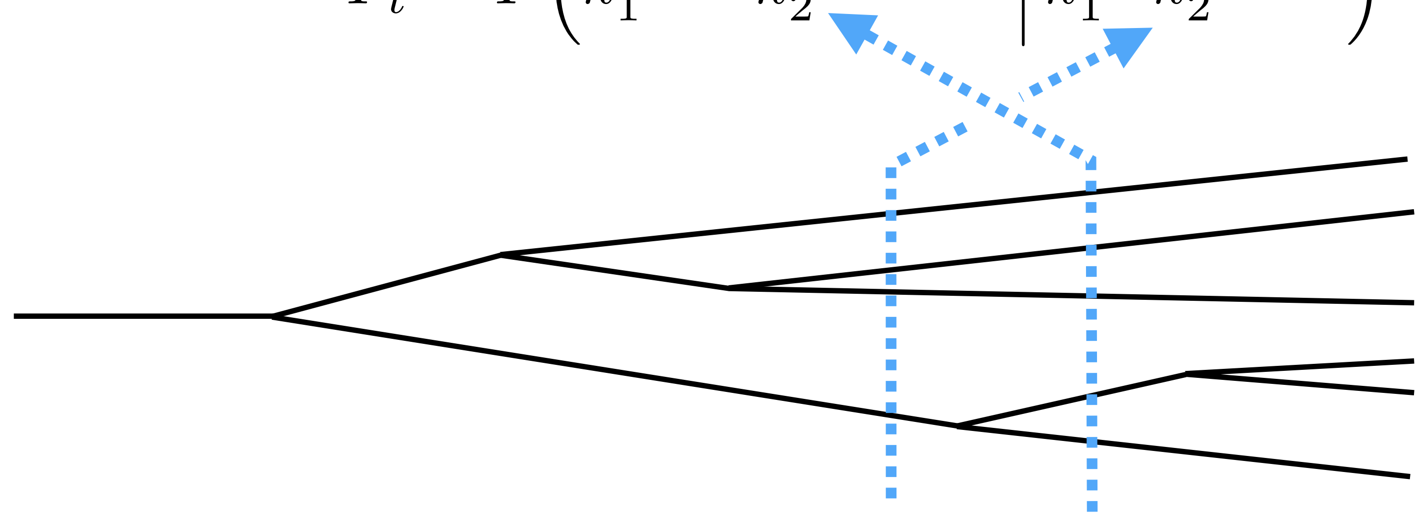
$$P_t = P_{\text{end}}\big(0\big|h^{(t)}\big) \cdot P_{\text{mother}}\big(m^{(t)}\big|h^{(t)}\big) \cdot P_{\text{branch}}\big(k_{d_1}^{(t+1)}\, k_{d_2}^{(t+1)}\big|\, k_m^{(t)}\, h^{(t)}\big)$$

or $$P_t = P_{\text{end}}\big(1\big|h^{(t)}\big)$$

General Form of Probabilistic Model

- decomposition into time steps $$P_{\text{jet}}(\{p_1 \dots p_n\}) = \prod_{t=1}^{n} P_t$$

where $$P_t = P\left(k_1^{(t+1)}\, k_2^{(t+1)} \cdots \,\middle|\, k_1^{(t)}\, k_2^{(t)} \cdots\right)$$



- further simplification: only $k_m^{(t)} \to k_{d_1}^{(t+1)}\, k_{d_2}^{(t+1)}$ at each time step

$$P_t = P_{\text{end}}\left(0\middle|h^{(t)}\right) \cdot P_{\text{mother}}\left(m^{(t)}\middle|h^{(t)}\right) \cdot P_{\text{branch}}\left(k_{d_1}^{(t+1)}\, k_{d_2}^{(t+1)}\middle|\, k_m^{(t)}\, h^{(t)}\right)$$

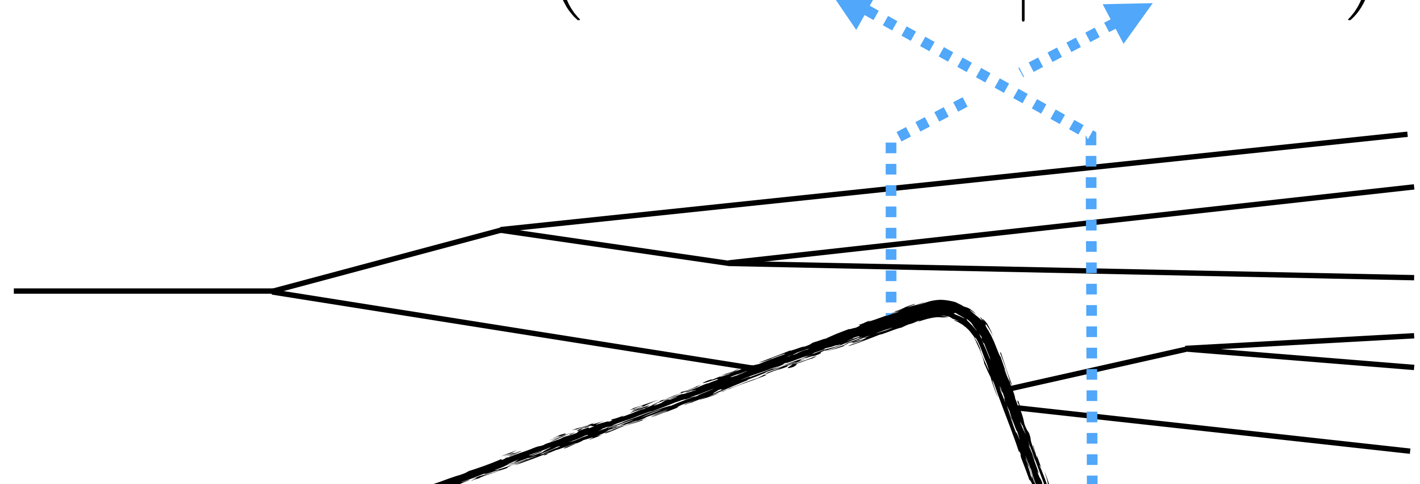
or $$P_t = P_{\text{end}}\left(1\middle|h^{(t)}\right)$$

representation of
“the rest of the jet”
at step t

General Form of Probabilistic Model

- decomposition into time steps $$P_{\text{jet}}(\{p_1 \ldots p_n\}) = \prod_{t=1}^{n} P_t$$

where $$P_t = P\left(k_1^{(t+1)}\, k_2^{(t+1)} \cdots \,\middle|\, k_1^{(t)}\, k_2^{(t)} \cdots\right)$$



- further simplification: onl … step

$P_t =$ … $\text{…ch}\left(k_{d_1}^{(t+1)}\, k_{d_2}^{(t+1)} \,\middle|\, k_m^{(t)}\, h^{(t)}\right)$

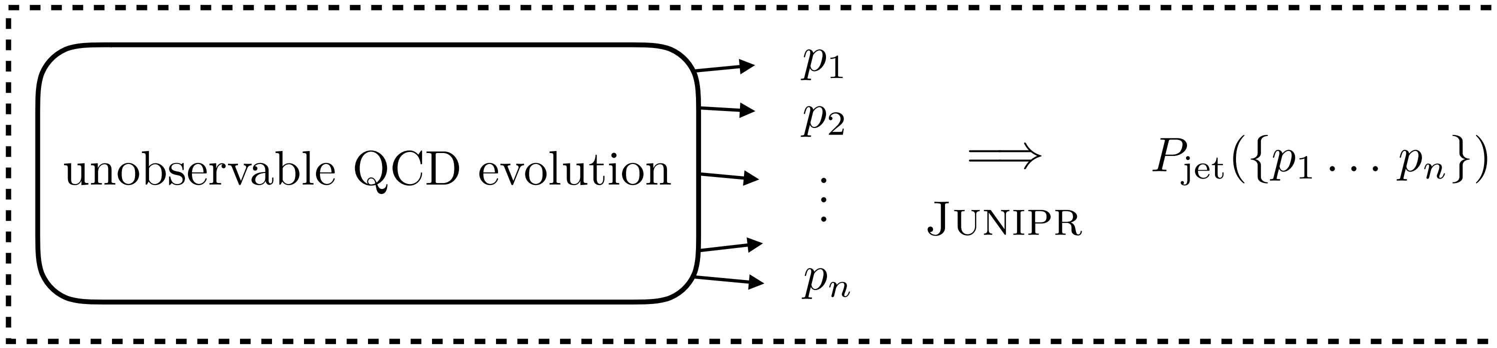
or $P_t =$ …

without loss of generality!

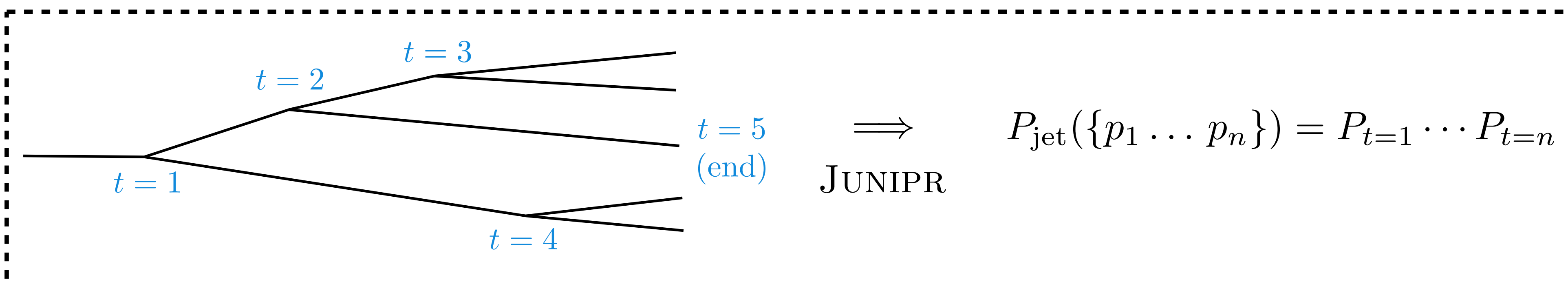
representation of
"the rest of the jet"
at step t

Summary So Far

JUNIPR computes the probability of a jet…



…as a product over time steps in its clustering tree…



…where each time step is decomposed into 3 parts:

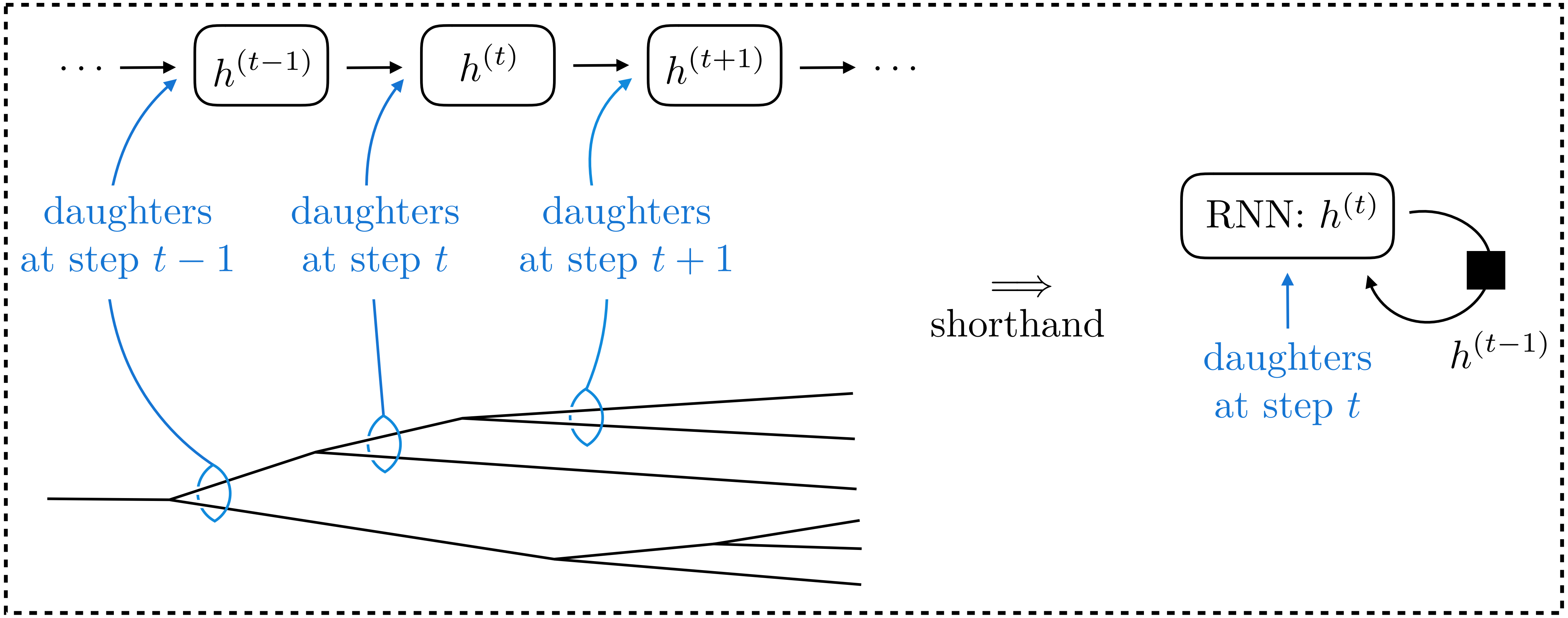
$$P_t = P_{\text{end}} \cdot P_{\text{mother}} \cdot P_{\text{branch}}$$

Implementation with RNN

- STEP 1) encode jet's structure into neural network
- STEP 2) use encoding to compute $P_{\rm end} \cdot P_{\rm mother} \cdot P_{\rm branch}$

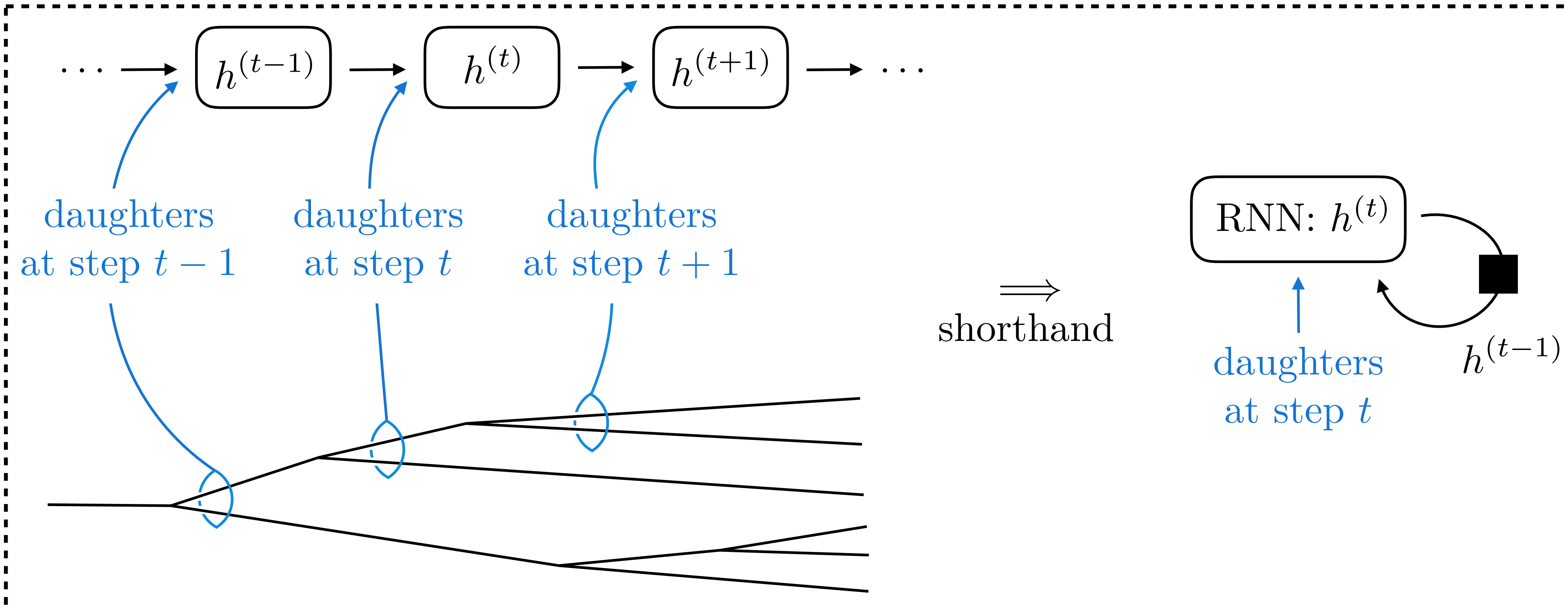
Implementation with RNN

- STEP 1) sequentially encode jet's information in hidden representation $h^{(t)}$
  - only feed latest additions to tree into update rule



Implementation with RNN

- STEP 1) sequentially encode jet's information in hidden representation $h^{(t)}$
  - only feed latest additions to tree into update rule



- to be precise, $h^{(t)} = \tanh\left(W \cdot (k_{d_1}^{(t)}\, k_{d_2}^{(t)}) + V \cdot h^{(t-1)} + b\right)$

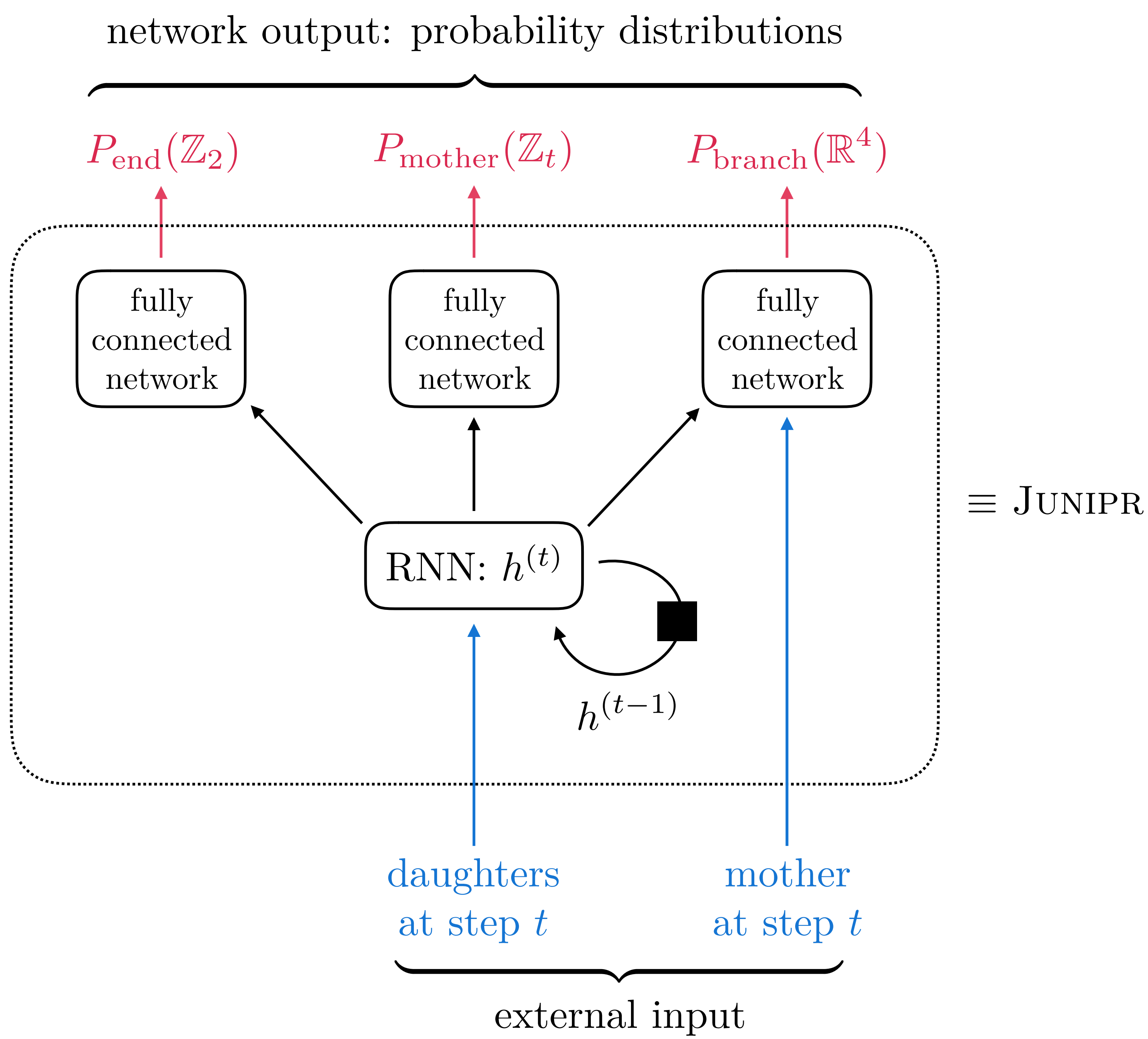
100-dim vector

learned "weights"

learned "bias"

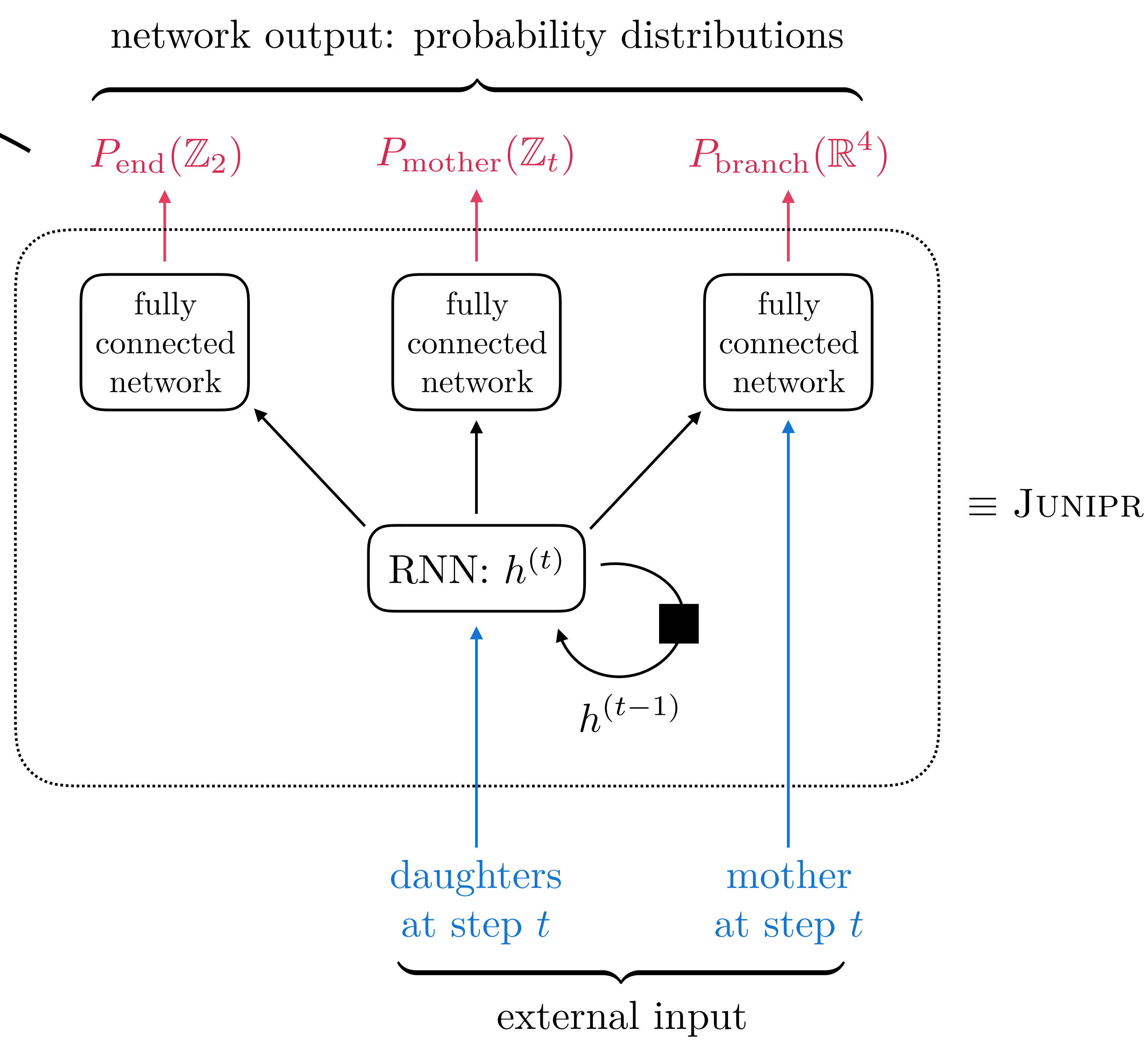
Implementation with RNN

- STEP 2) feed $h^{(t)}$ into neural networks computing probability distributions



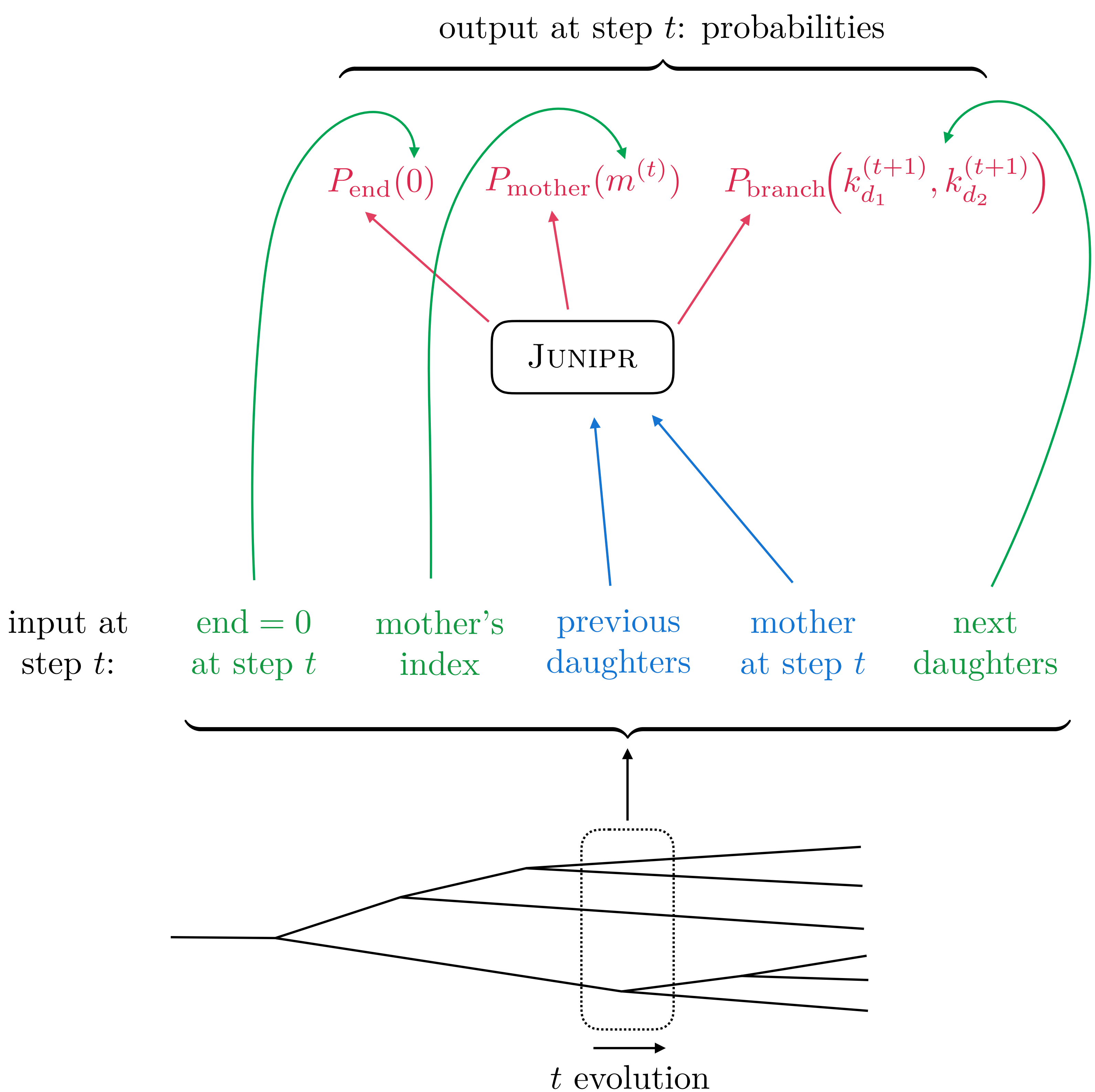
Implementation with RNN

- STEP 2) feed $h^{(t)}$ into neural networks computing probability distributions
  - — each component is neural network
  - — simple building blocks model arbitrarily general functions



Using JUNIPR to compute P_{jet}

- sequentially feed jet's tree into JUNIPR
- evaluate probability distributions at actual outcomes
- P_{jet} is product $P_{\text{end}}\, P_{\text{mother}}\, P_{\text{branch}}$ over all time steps



Unsupervised Learning

- JUNIPR is a model $P_\theta(\text{jet})$ with 10^6 parameters θ
- best parameters are learned from training data:

$$\theta \;=\; \underset{\theta'}{\operatorname{argmax}} \sum_{\substack{\text{jet in}\\ \text{data}}} \log P_{\theta'}(\text{jet})$$

i.e. choose θ to maximize log likelihood

- in practice, use stochastic gradient ascent:

$$\theta_{n+1} \;=\; \theta_n + \alpha \cdot \nabla \sum_{\substack{\text{jet in batch}\\ \text{of data}}} \log P_{\theta_n}(\text{jet})$$

Training Data

- simulated in Pythia as proof-of-concept: 500k jets from $e^+e^- \to q\bar{q}$ events

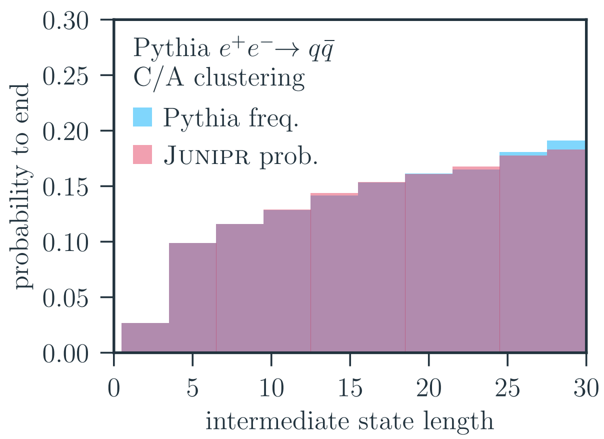
$$E_{\text{jet}} \sim 500 \text{ GeV} \quad \text{with} \quad R_{\text{jet}} \sim \pi/2$$

- jet constituents clustered to obtain 500k angular-ordered trees for training

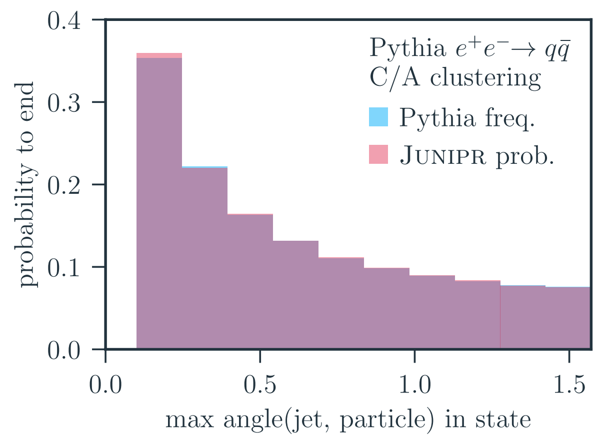
- all methods repeatable on LHC data

Model Validation

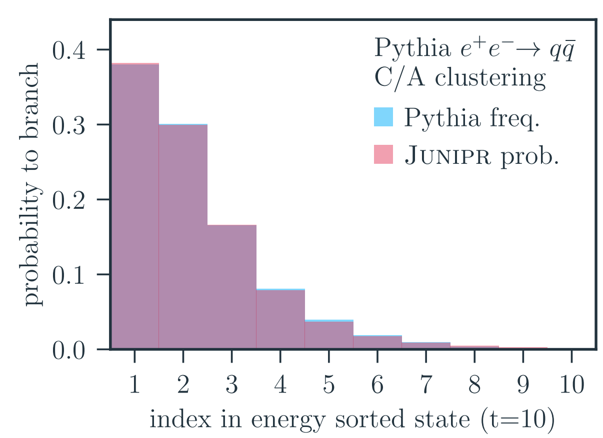
P_{end}
vs
state length



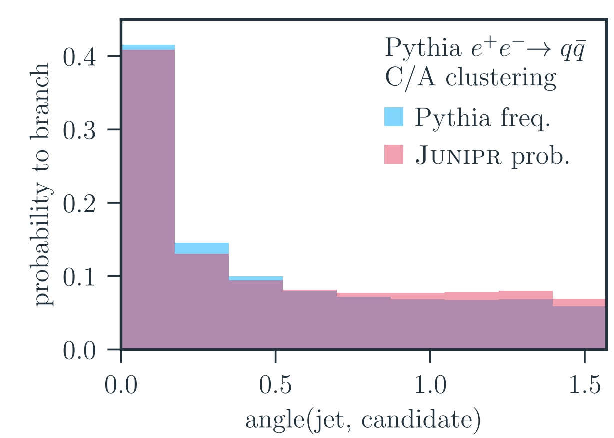
P_{end}
vs
angular distribution



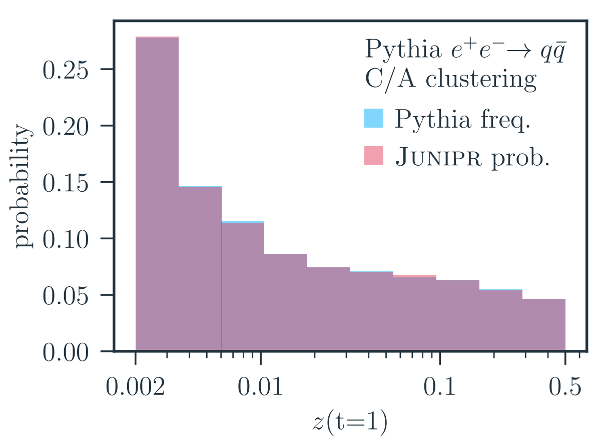
P_{mother}
vs
candidate's energy



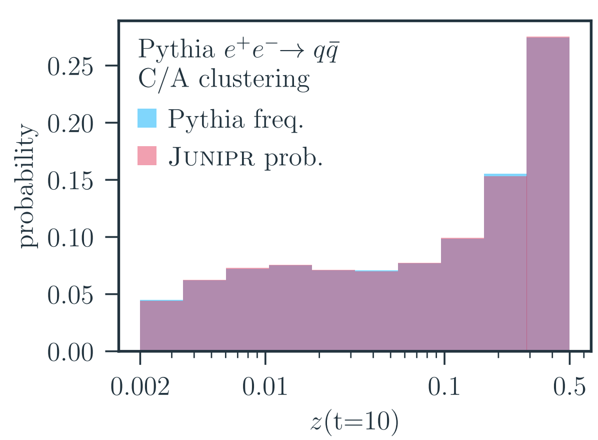
P_{mother}
vs
candidate's angle



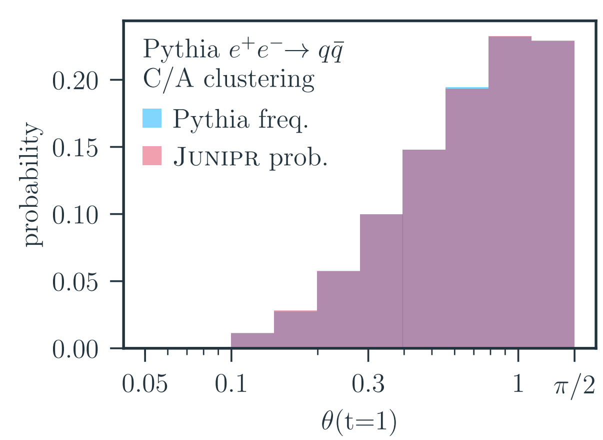
P_{branch}
vs
emission energy
(t = 1)



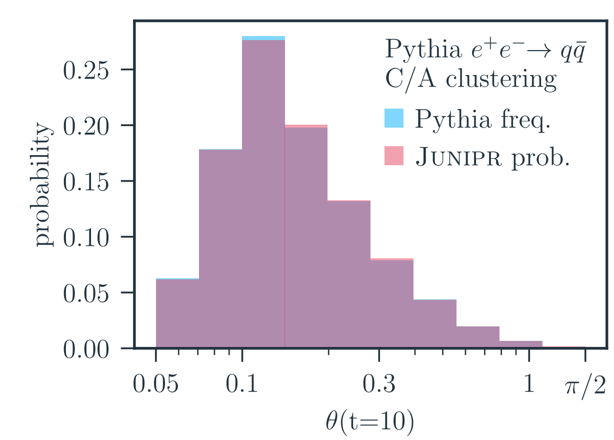
P_{branch}
vs
emission energy
(t = 10)



P_{branch}
vs
emission angle
(t = 1)



P_{branch}
vs
emission angle
(t = 10)

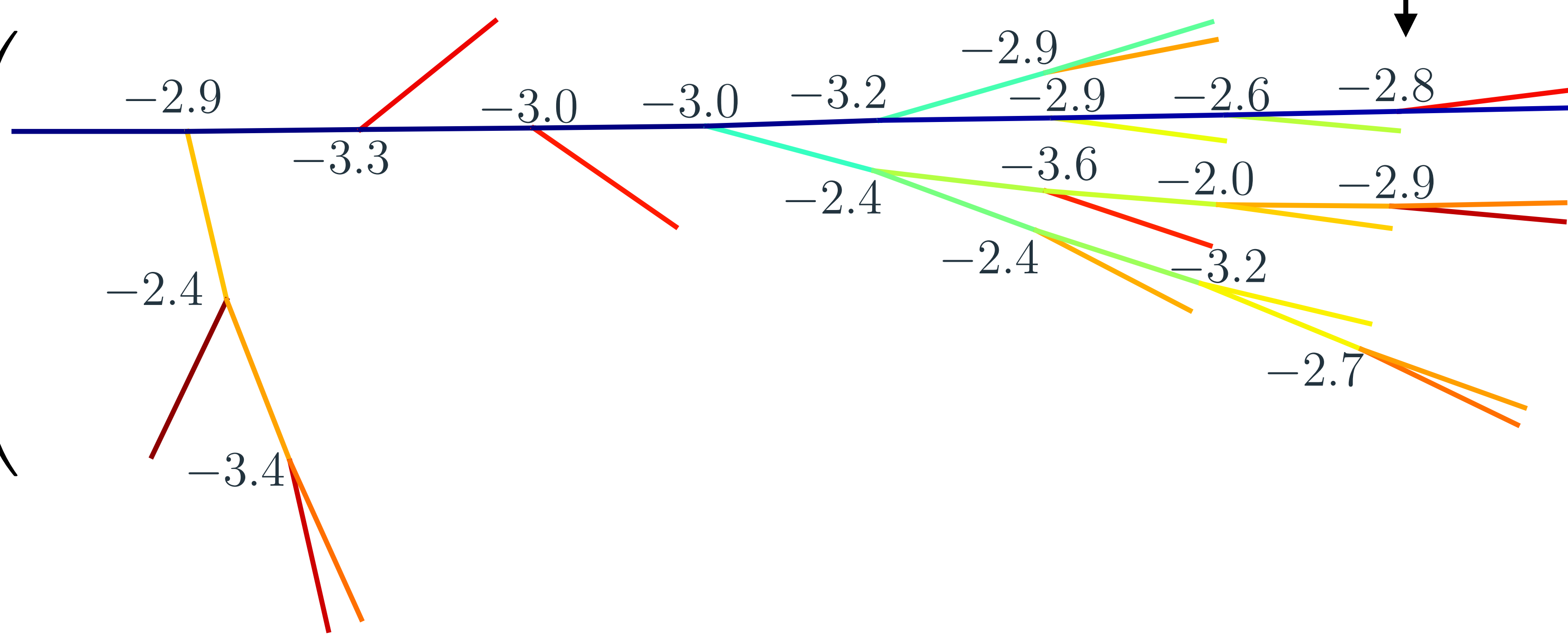


JUNIPR is trained and ready to operate!

- for illustration, JUNIPR can evaluate probability of example Pythia quark jet:

$$P_{t=18} = \overbrace{\left(10^{-0.7}\right)\,\left(10^{-0.1}\right)\,\left(10^{-2.0}\right)}^{P_{\mathrm{end}} \cdot P_{\mathrm{mother}} \cdot P_{\mathrm{branch}}} = 10^{-2.8}$$

P_{jet} (



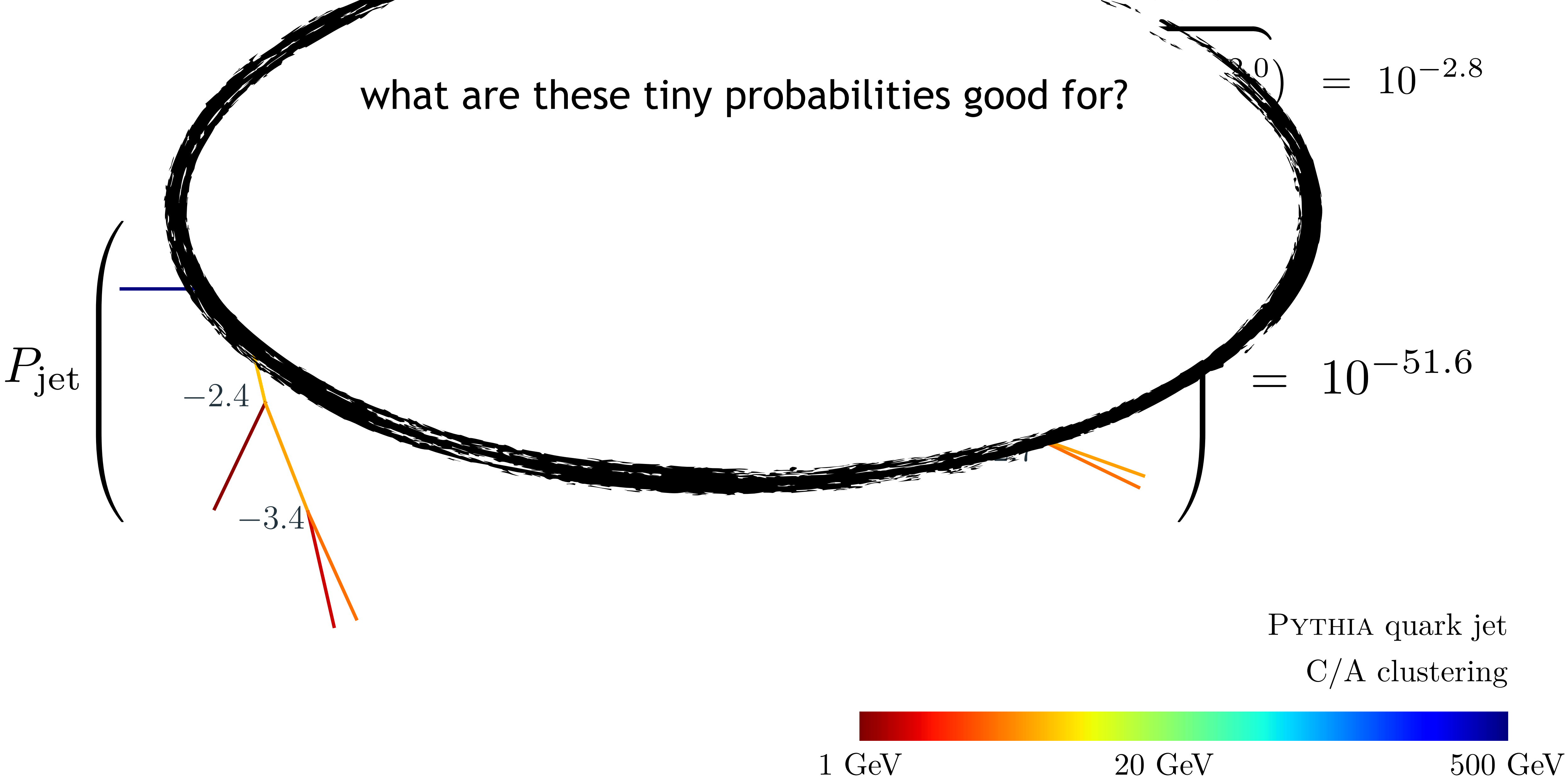
) $= 10^{-51.6}$

PYTHIA quark jet

C/A clustering

JUNIPR is trained and ready to operate!

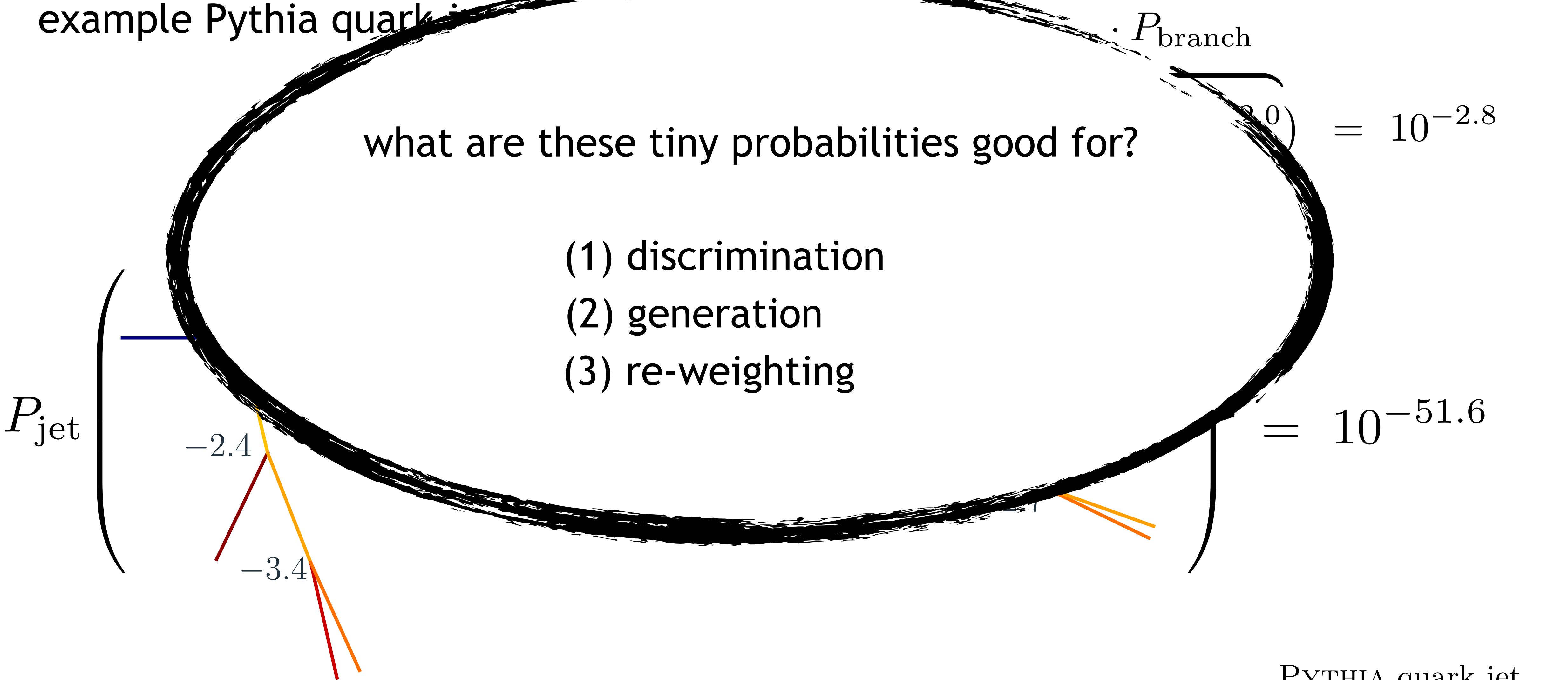
- for illustration, JUNIPR can evaluate probability of example Pythia quark j



what are these tiny probabilities good for?

JUNIPR is trained and ready to operate!

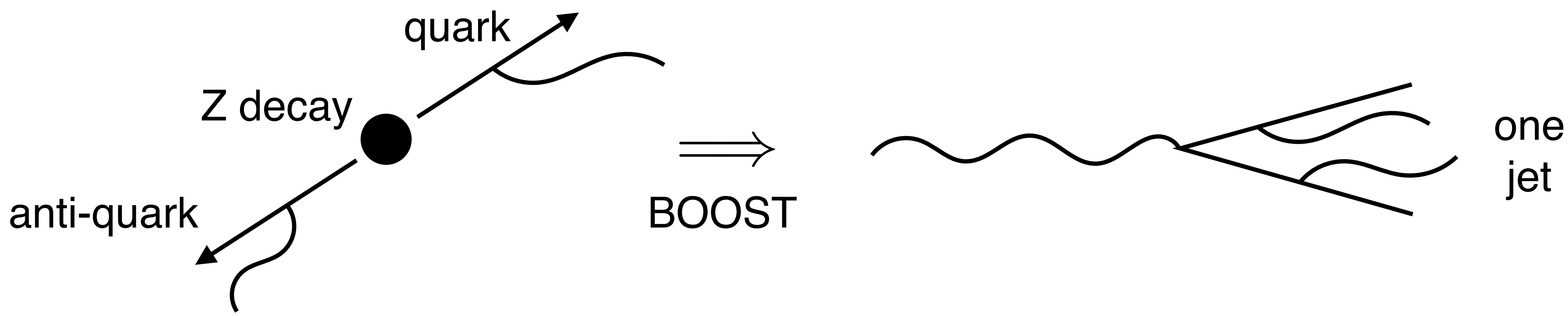
- for illustration, JUNIPR can evaluate probability of example Pythia quark jet



what are these tiny probabilities good for?

(1) discrimination
(2) generation
(3) re-weighting

(1) Discrimination



- boosted Z / quark jet discrimination for proof-of-concept

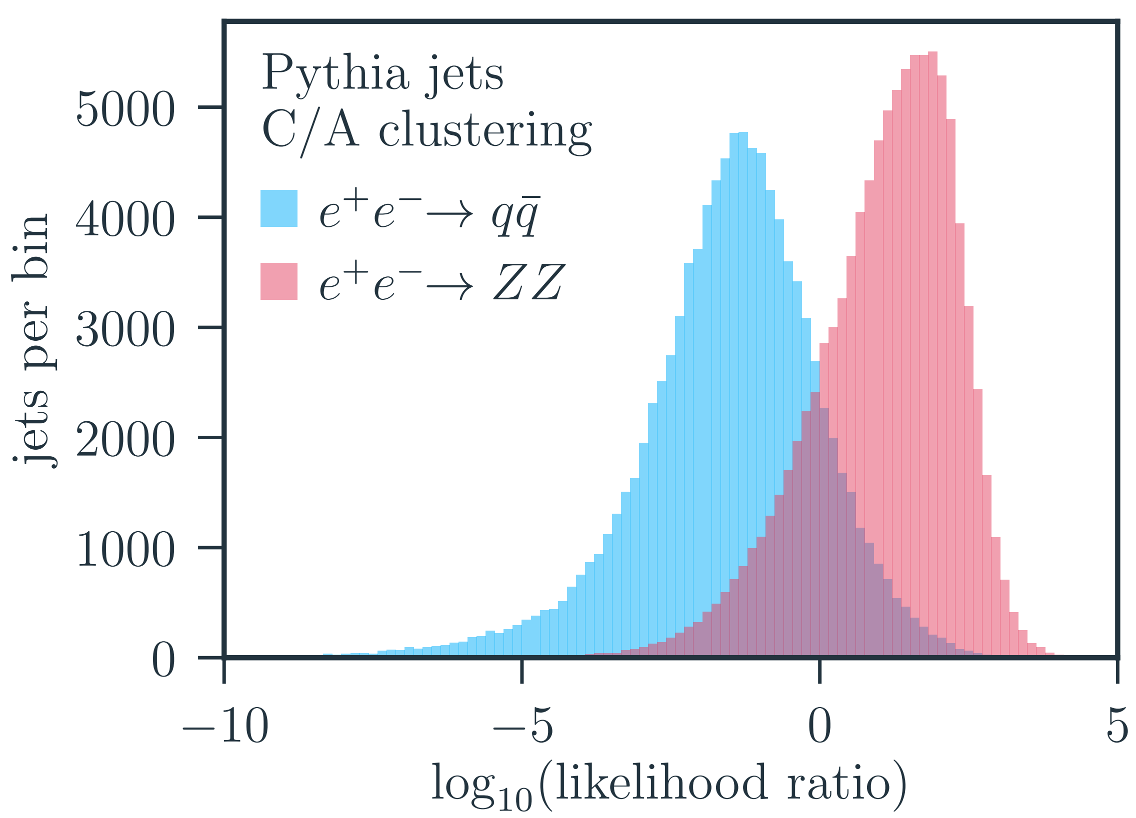
- trained two models: $\left.\begin{array}{r} P_Z(\text{jet}) \\ P_q(\text{jet}) \end{array}\right\}$ mass cut on jets in training data 90.7 — 91.7 GeV

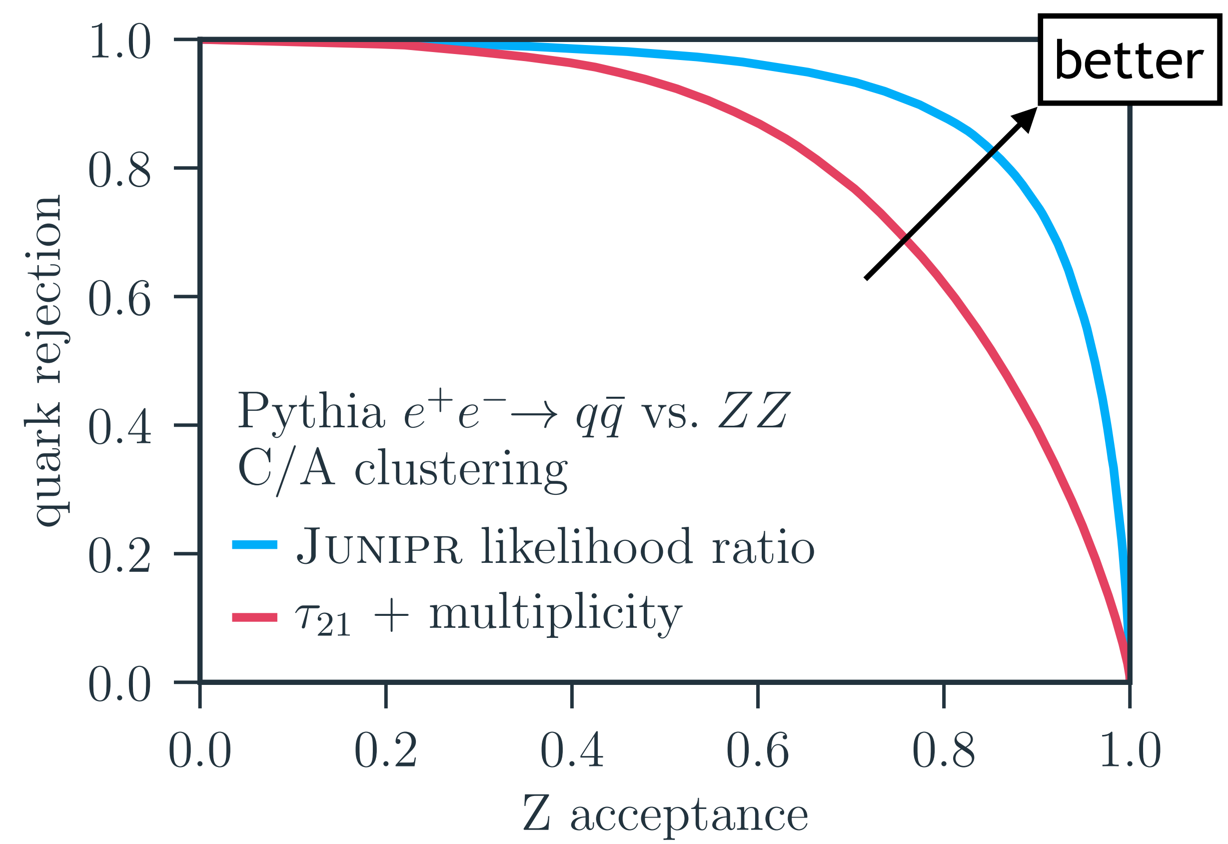
- theoretically most powerful discriminant is likelihood ratio:

$$\frac{P_Z(\text{jet})}{P_q(\text{jet})} > \text{threshold} \quad \Longrightarrow \quad \text{tag jet as boosted } Z$$

(1) Discrimination

$$\boxed{\frac{P_Z(\text{jet})}{P_q(\text{jet})} > \text{threshold} \quad \Longrightarrow \quad \text{tag jet as boosted } Z}$$





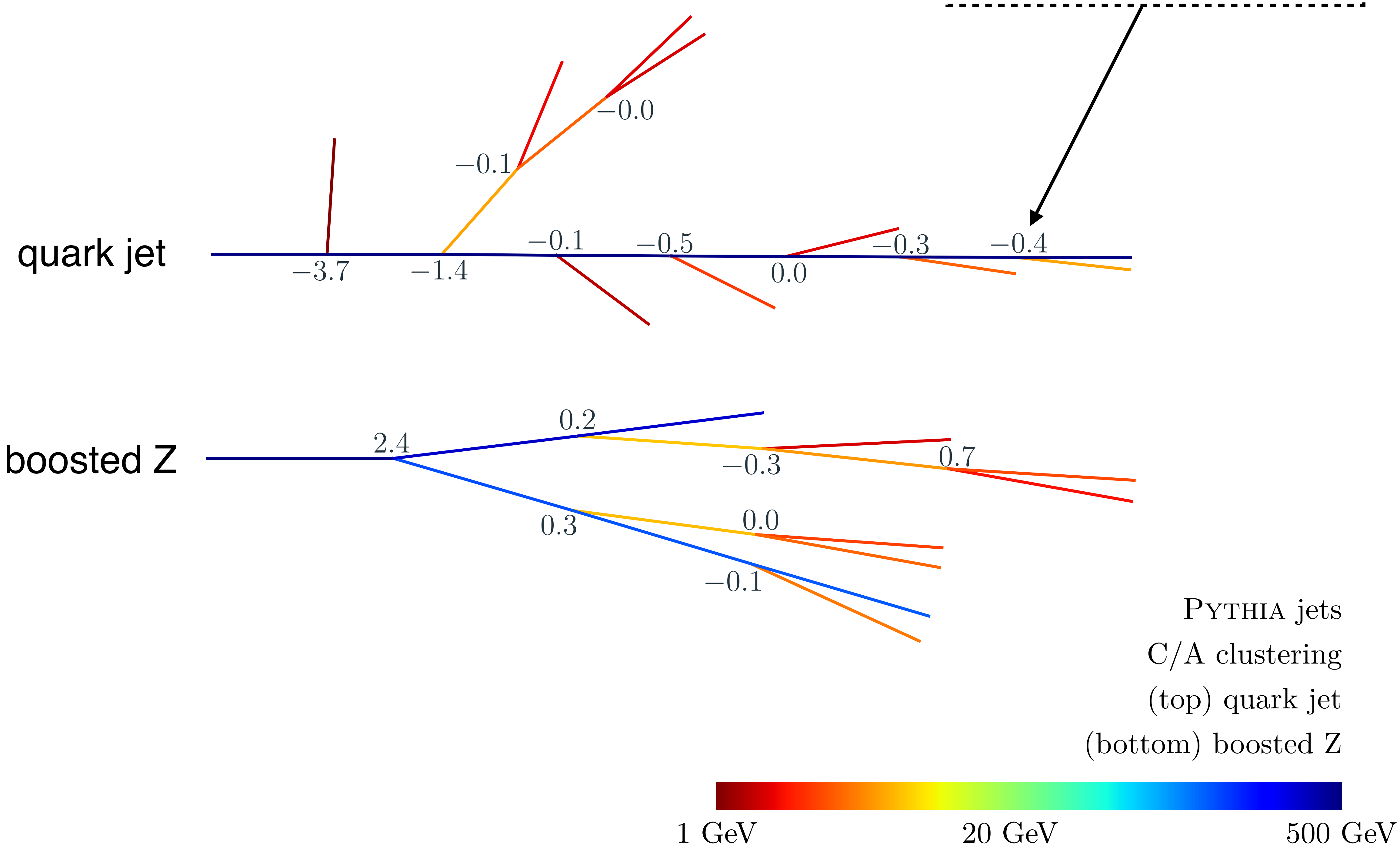
(1) Discrimination

- visualize high-performance discrimination with clustering trees!

nodes labeled by

$$\log_{10} \frac{P_Z(t)}{P_q(t)}$$

negative $\Longrightarrow$ quark like
positive $\Longrightarrow$ Z like



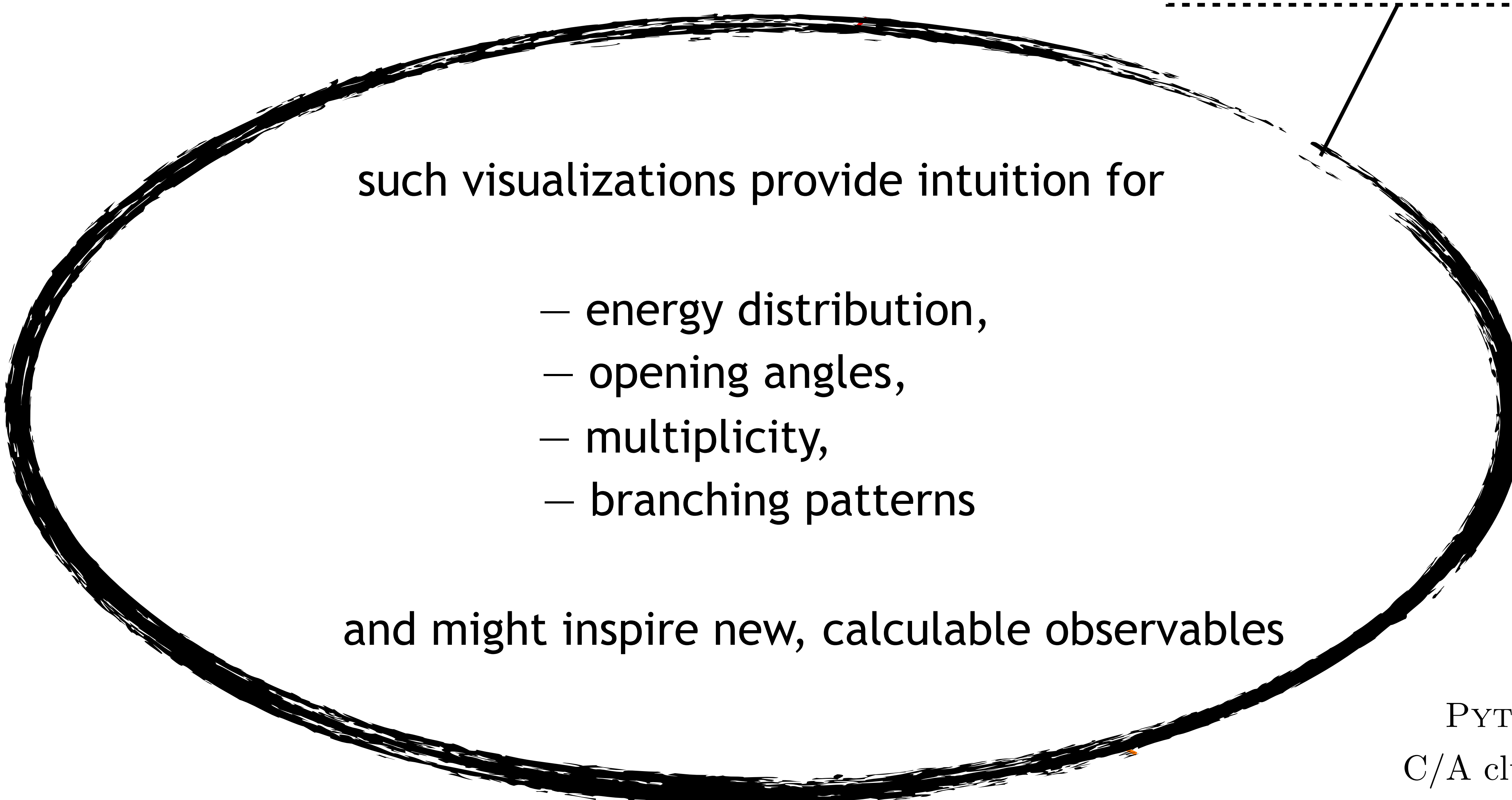
(1) Discrimination

- visualize high-performance discrimination with clustering trees!

nodes labeled by

$$\log_{10} \frac{P_Z(t)}{P_q(t)}$$

negative $\Longrightarrow$ quark like
positive $\Longrightarrow$ Z like



such visualizations provide intuition for

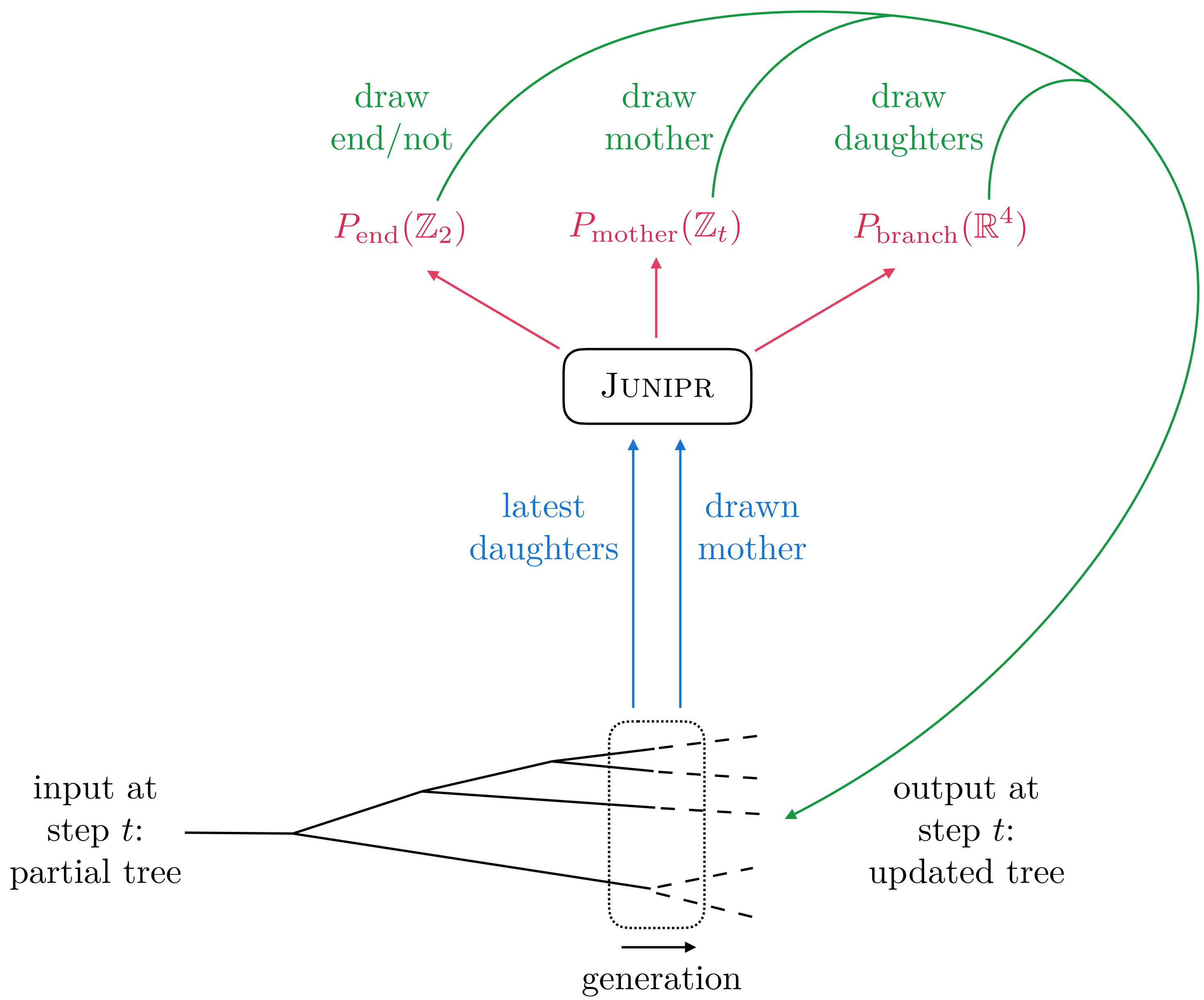
— energy distribution,
— opening angles,
— multiplicity,
— branching patterns

and might inspire new, calculable observables

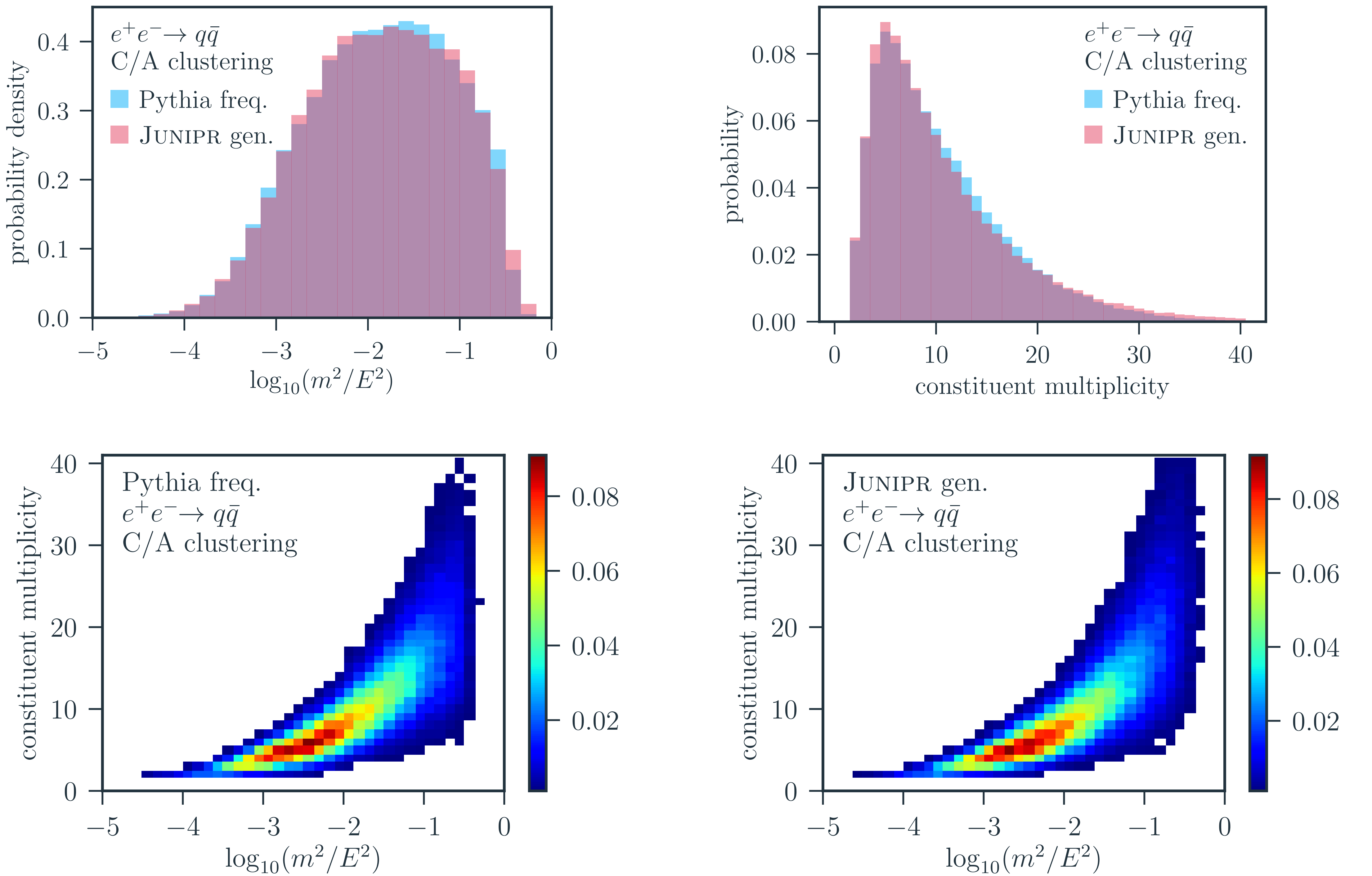
PYTHIA jets
C/A clustering
(top) quark jet
(bottom) boosted Z

(2) Generation

- sample from learned probabilistic model $P_{\text{jet}}(\{p_1 \ldots p_n\})$ to generate jets in agreement with training data



(2) Generation



(3) Re-Weighting

- imagine training 2 versions of JUNIPR
  - one on real LHC data
  - one on imperfect simulated data

$$\Longrightarrow \begin{cases} P_{\mathrm{LHC}}(\mathrm{jet}) \\ P_{\mathrm{sim}}(\mathrm{jet}) \end{cases}$$

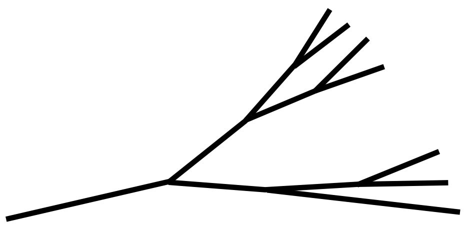
- re-weighting <u>simulated</u> jets by $\dfrac{P_{\mathrm{LHC}}(\mathrm{jet})}{P_{\mathrm{sim}}(\mathrm{jet})}$ should lead to agreement in observable distributions

- proof-of-concept:
  trained JUNIPR on 2 different simulations
  - Pythia with $\alpha_s(m_Z) = 0.1365$
  - Pythia with $\alpha_s(m_Z) = 0.11$

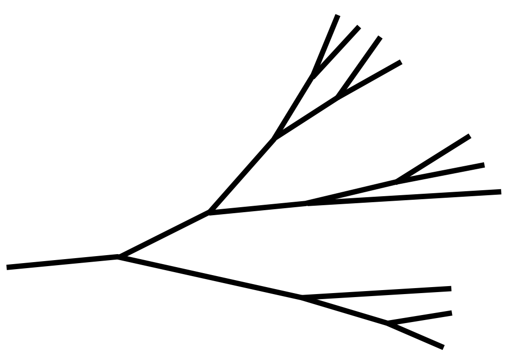
(3) Re-Weighting

two different simulations

events re-weighted to agree!

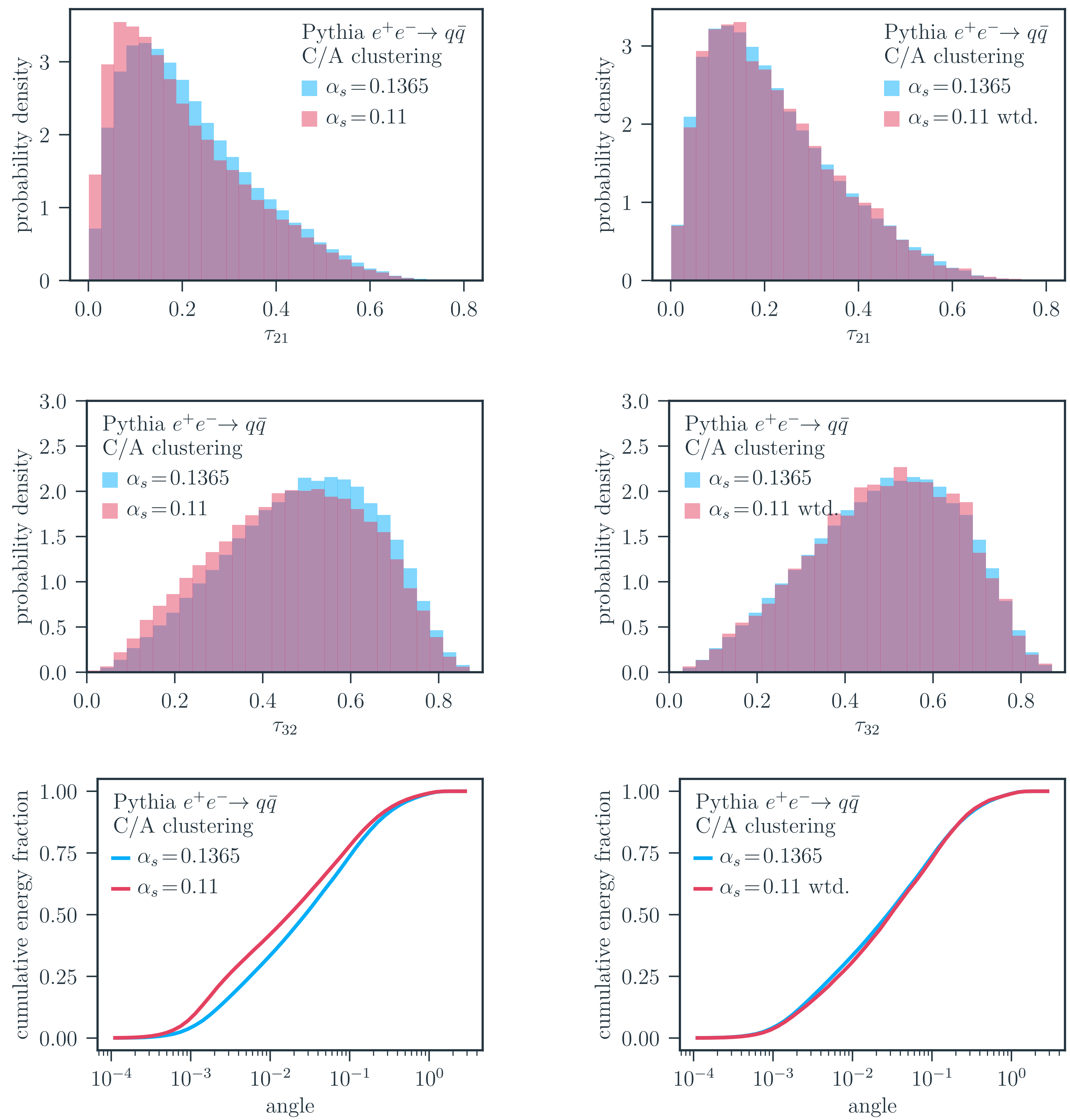


2-pronged substructure



3-pronged substructure

energy distribution vs angle



Quark vs. Gluon Discrimination

Quark vs. Gluon Discrimination

- Naively applying JUNIPR does not give state-of-the-art performance
- JUNIPR is trained to learn each distribution
- Not optimized for discrimination

Quark vs. Gluon Discrimination

- Naively applying JUNIPR does not give state-of-the-art performance
- JUNIPR is trained to learn each distribution
- Not optimized for discrimination

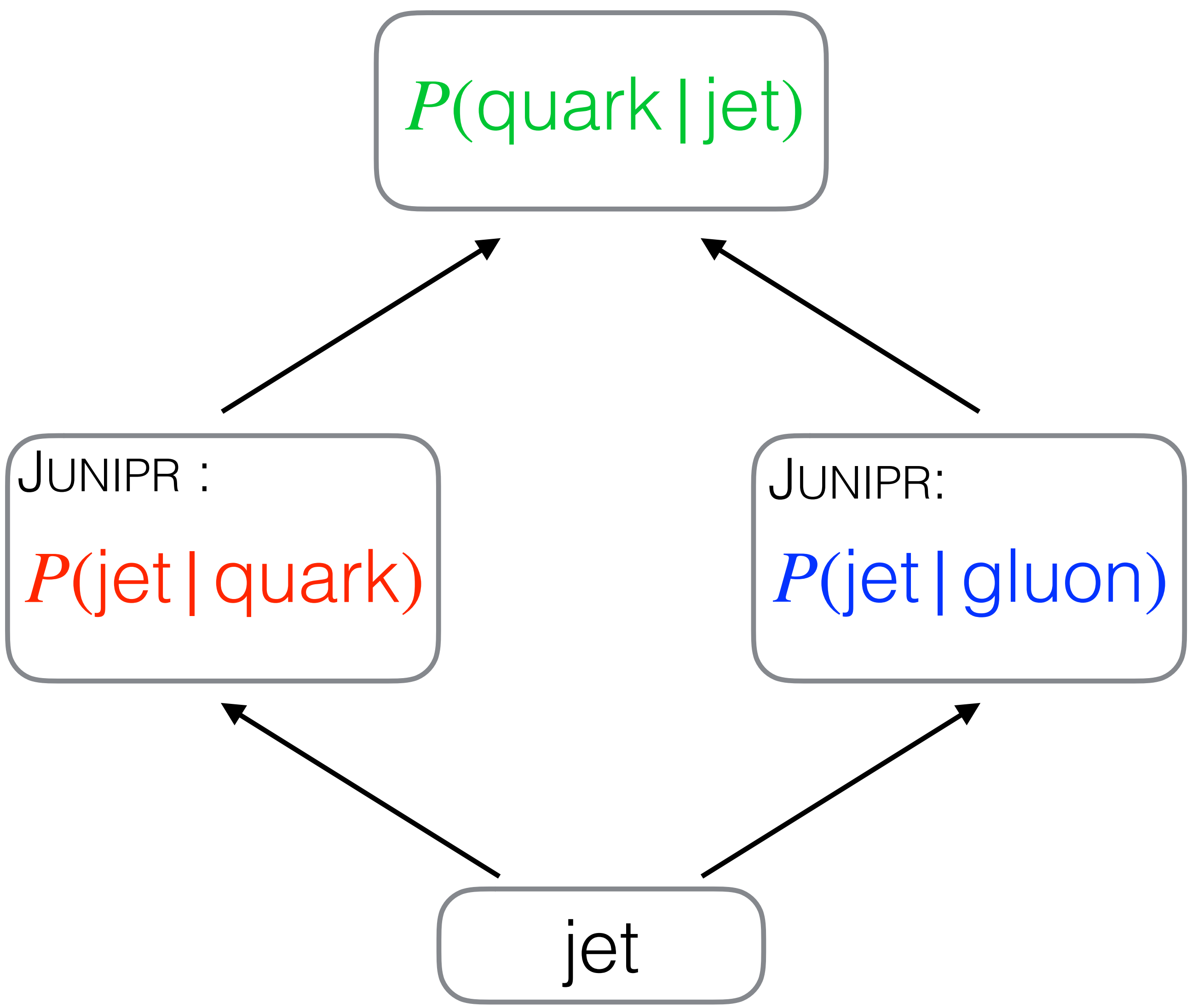
Optimize for discrimination with Bayes' theorem:

- $P_q(\text{jet}) = P(\text{jet}\,|\,\text{quark})$
- $P_g(\text{jet}) = P(\text{jet}\,|\,\text{gluon})$

$$P(\text{quark}\,|\,\text{jet}) = \frac{P(\text{jet}\,|\,\text{quark}) \cdot P(\text{quark})}{P(\text{jet}\,|\,\text{gluon}) \cdot P(\text{gluon}) + P(\text{jet}\,|\,\text{quark}) \cdot P(\text{quark})}$$

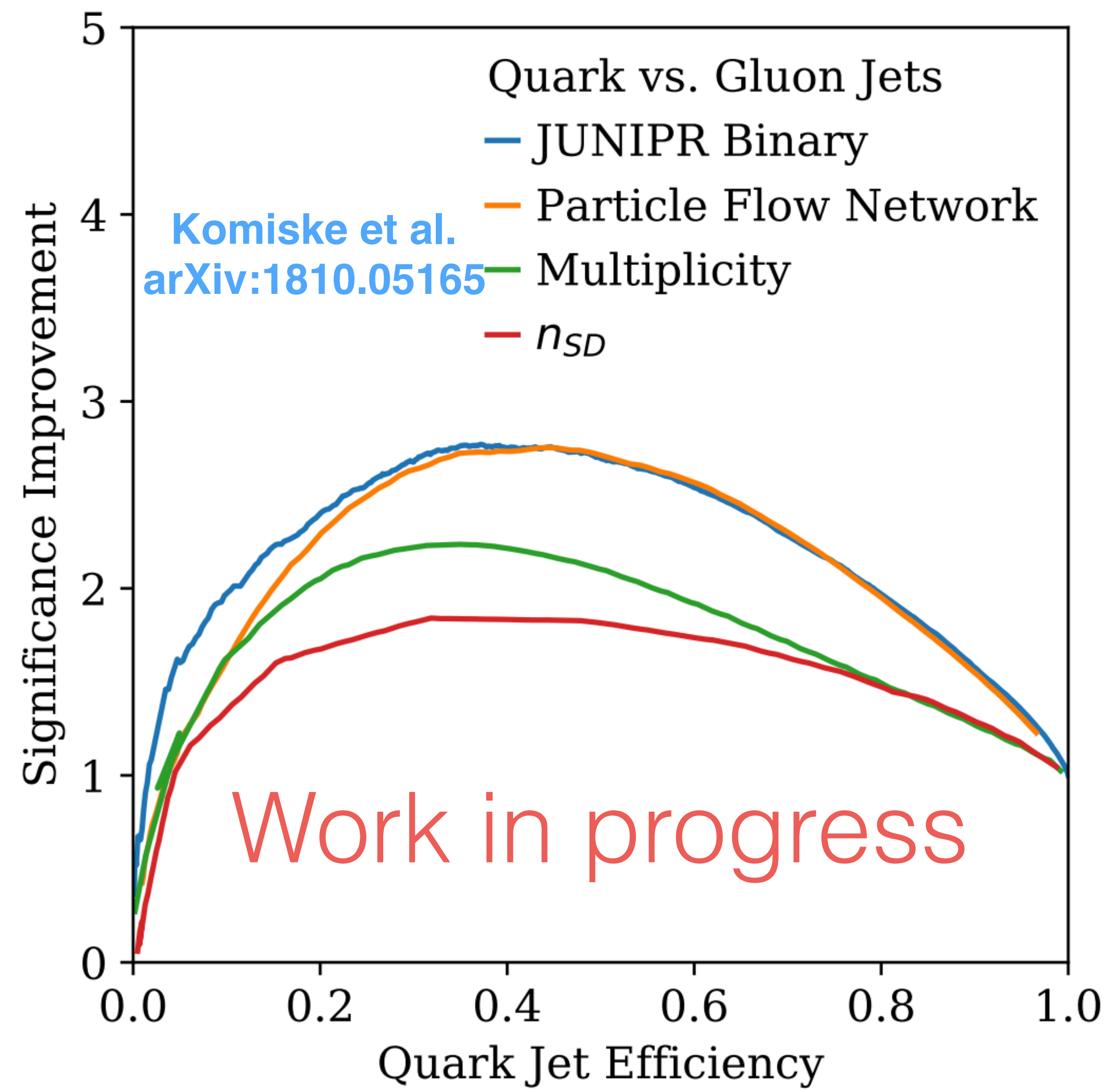
JUNIPR Binary — Optimized for Discrimination

$$P(\text{quark}\,|\,\text{jet}) = \frac{P(\text{jet}\,|\,\text{quark}) \cdot P(\text{quark})}{P(\text{jet}\,|\,\text{gluon}) \cdot P(\text{gluon}) + P(\text{jet}\,|\,\text{quark}) \cdot P(\text{quark})}$$



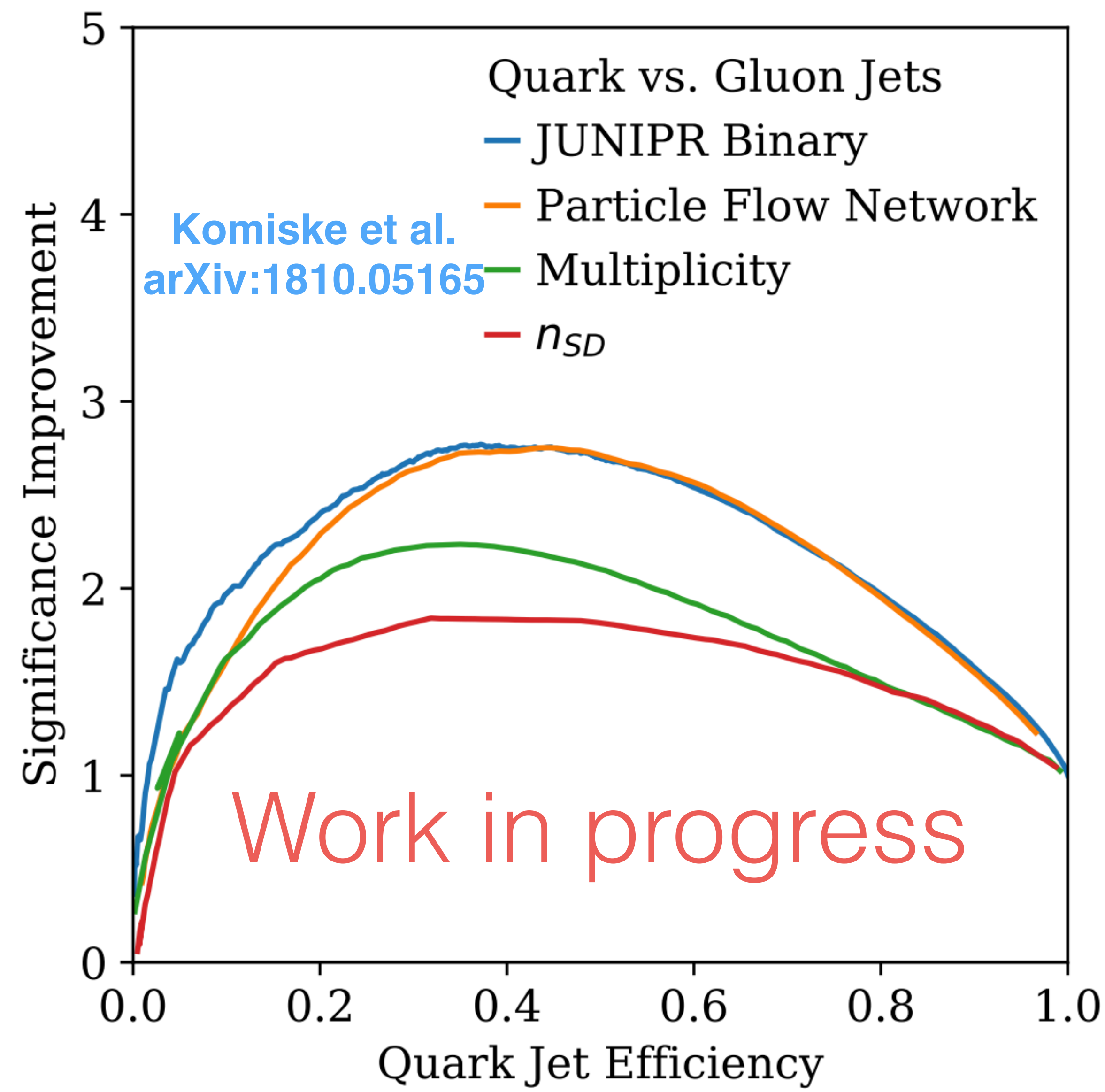
Quark vs. Gluon Discrimination

- JUNIPR Binary trained using EnergyFlow Dataset [Komiske, Metodiev, Thaler (2018)]
- Promising performance — still work in progress
- PFN better with Particle IDs. JUNIPR w/quantum numbers in progress [AA, Culp, Schwartz]



Quark vs. Gluon Discrimination

- JUNIPR Binary trained using EnergyFlow Dataset [Komiske, Metodiev, Thaler (2018)]
- Promising performance — still work in progress
- PFN better with Particle IDs. JUNIPR w/quantum numbers in progress [AA, Culp, Schwartz]



WANT TO LOOK INSIDE AND SEE WHAT JUNIPR HAS LEARNED

Conclusions & Outlook

- unsupervised probabilistic model for jets
  - neural network architecture scaffolded on leading order description of underlying physics
  - can be trained directly on LHC data
    $\Longrightarrow$ exciting future applications...

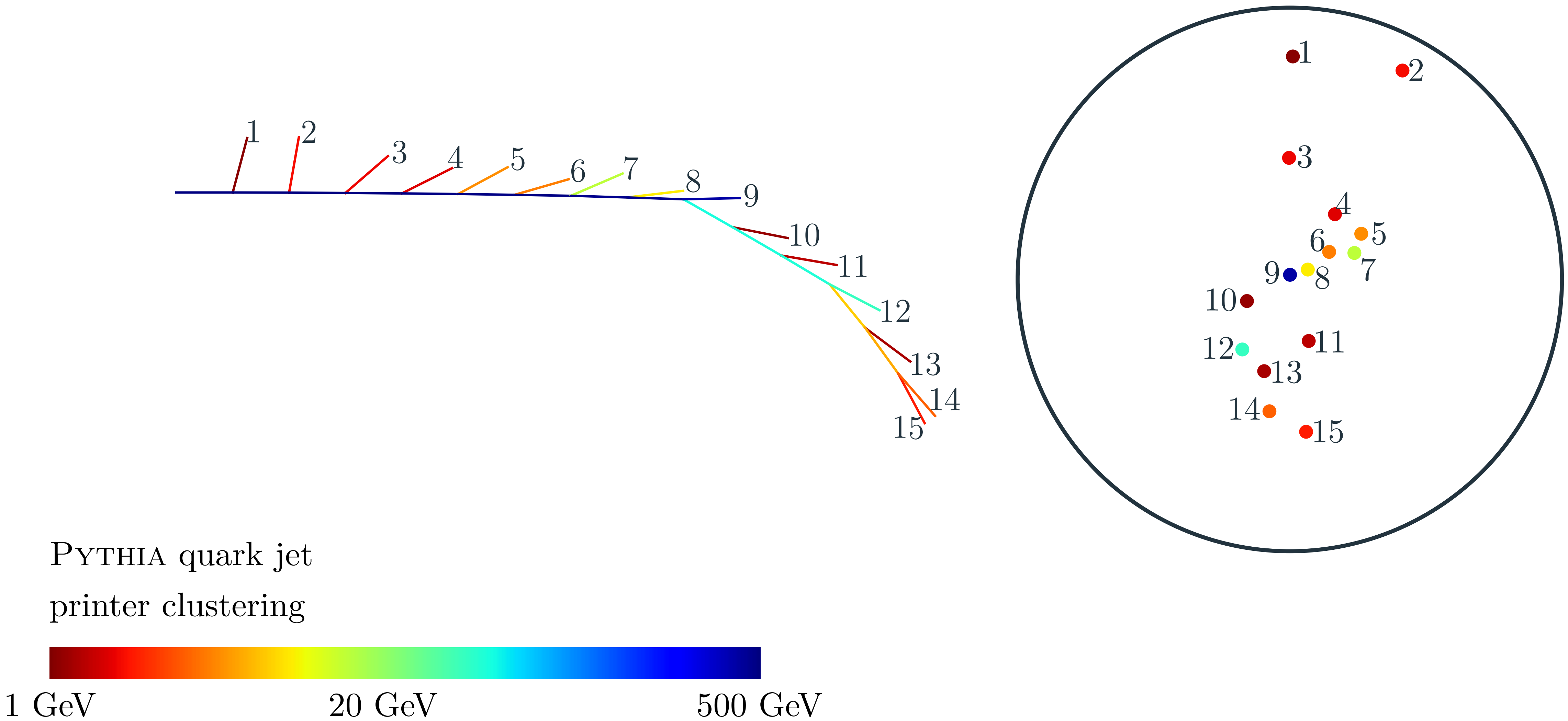
- quark/gluon (top/QCD) discrimination + interpretation?
- re-weighting could improve Monte Carlo event generators
- quantum numbers, full events, heavy ion collisions...

THANKS!

BACKUP SLIDES

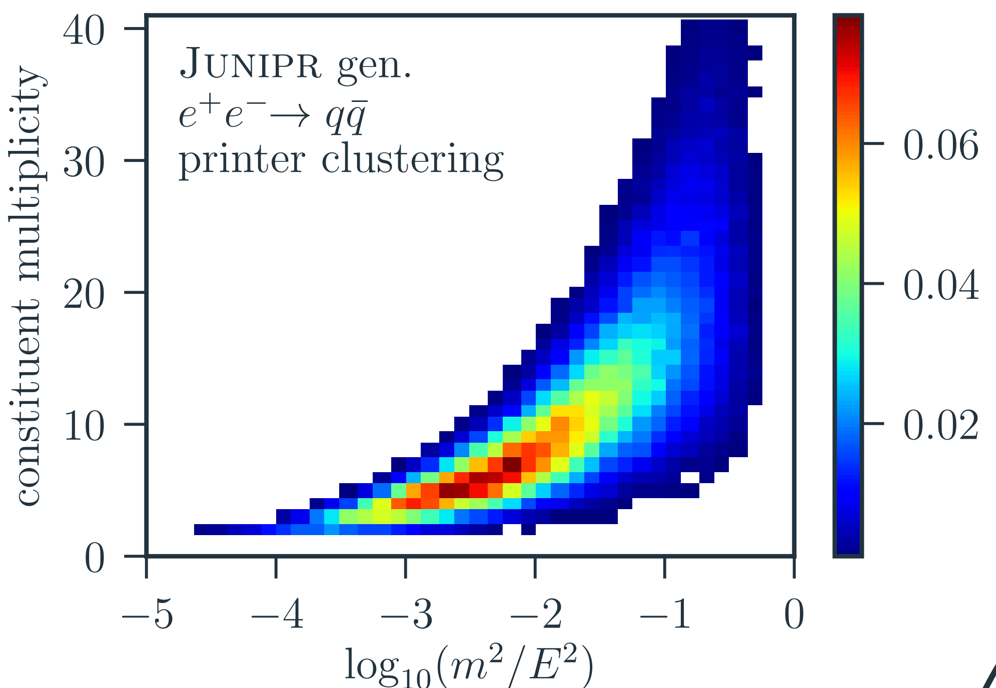
Could JUNIPR learn unexpected physics?

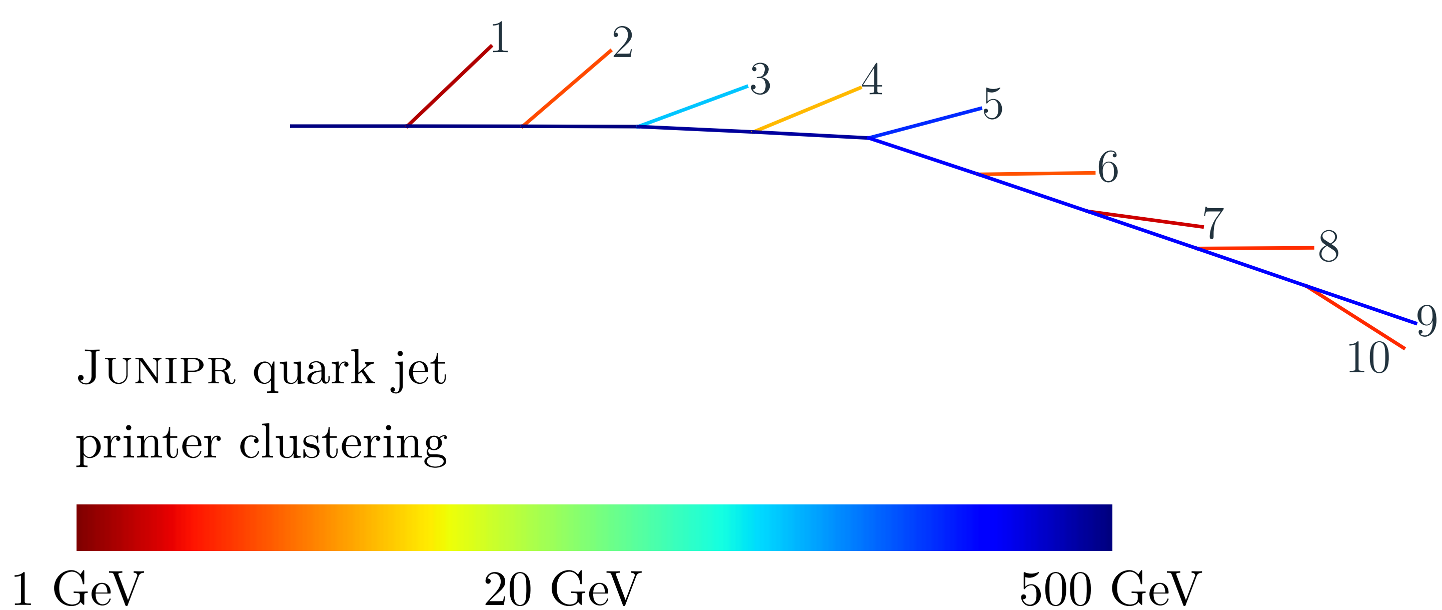
- to check: can JUNIPR learn QCD through an absurd clustering algorithm?
- “2D printer” algorithm processes emissions left-to-right and top-to-bottom

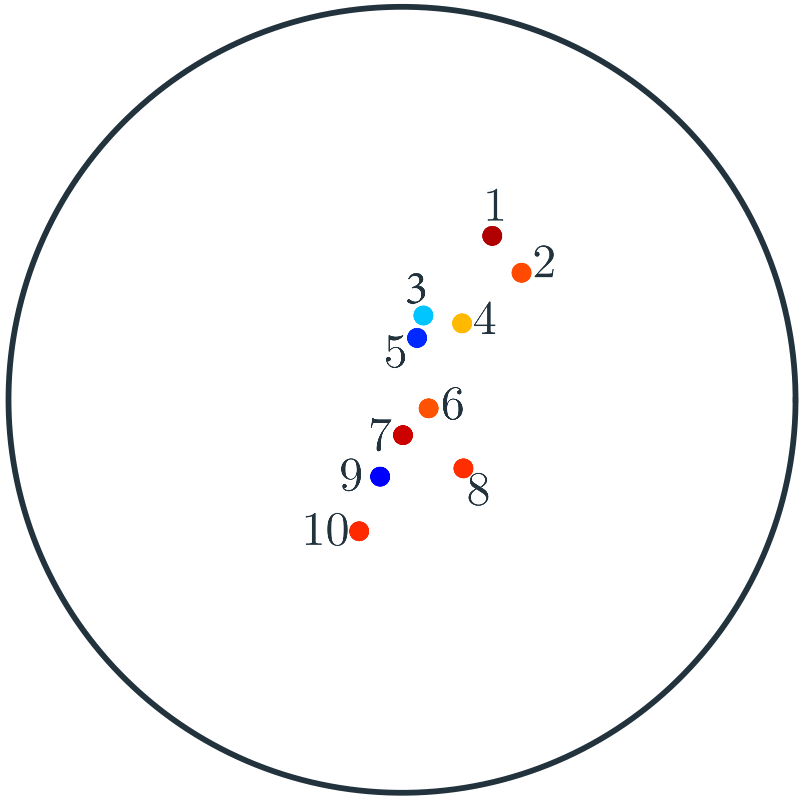


Could JUNIPR learn unexpected physics?

- answer: yes, though it is optimized for QCD

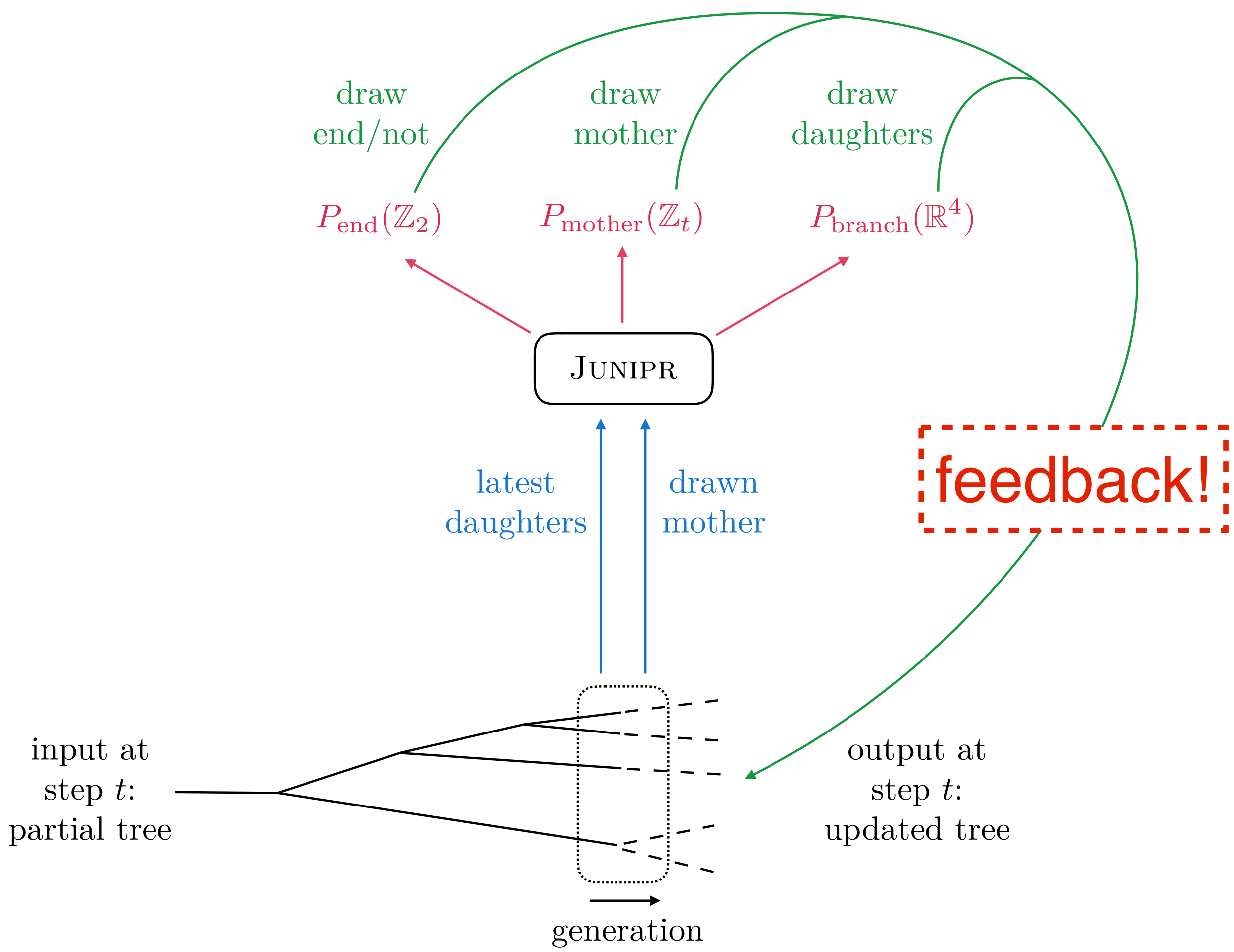






Problem with Generation

- generation is not JUNIPR's intended purpose
- small errors at early time steps feedback to become significant discrepancies



- feedback causes certain observables to be unreliable