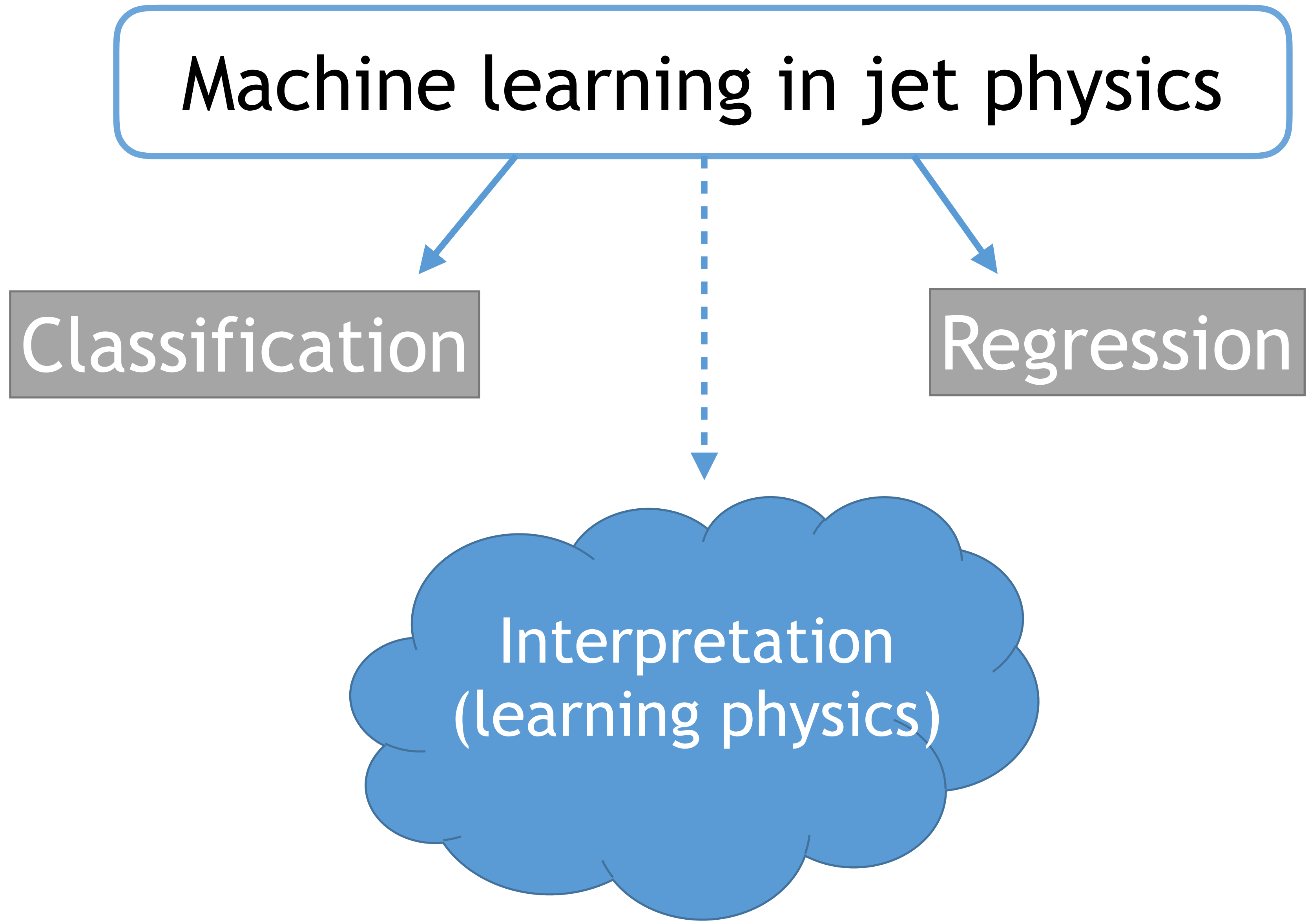
Learning Jet Evolution with a Recurrent Neural Network Based Probabilistic Model

Anders Andreassen & Christopher Frye

in collaboration with

Ilya Feige & Matthew Schwartz



How can we learn about jets using machine learning?

Idea: (1) Use a neural network to reproduce jets
(2) Look inside network to see what it's doing

difficult!

Want to reproduce mapping:
hard parton $\mapsto$ momenta of stable hadrons

Try to learn MC parton shower?

- not repeatable on data

No!

Learn the jet clustering tree!

- works on data

Yes!

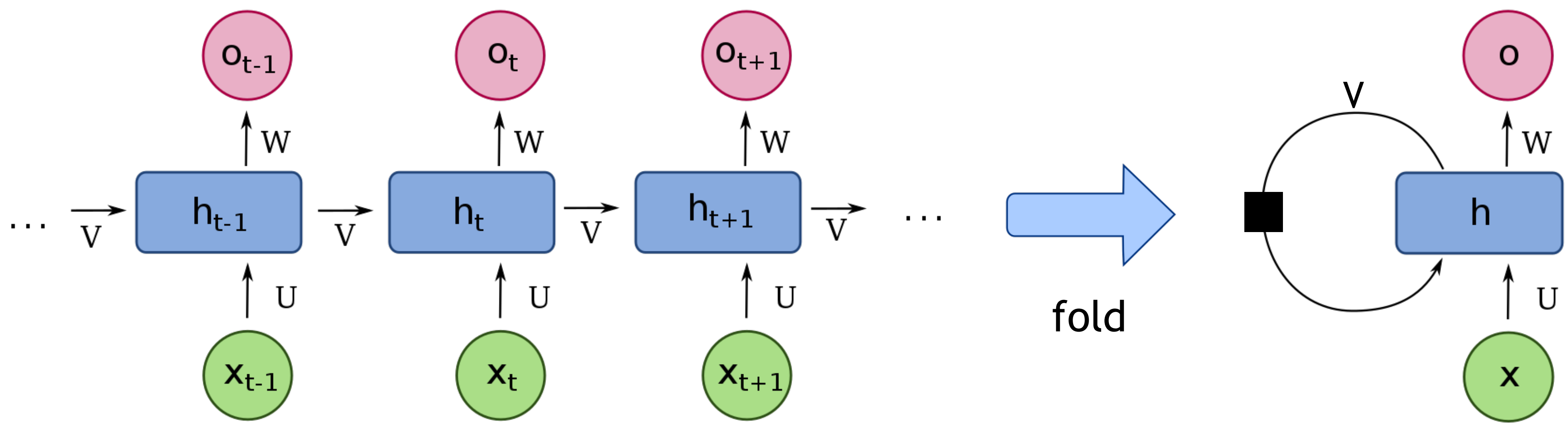
How can we build a machine learning model
sufficiently flexible to fit any data,
and sufficiently transparent to probe what it learned?

Want an architecture inspired by factorization,
but general enough to fit any non-factorizing structure

We have built a probabilistic model with architecture customized for jet evolution

Its transparent structure allows us to interpret output from intermediate layers and probe what the model has learned

Recurrent neural networks naturally model sequential evolution



…and can handle data with indeterminate number of time steps

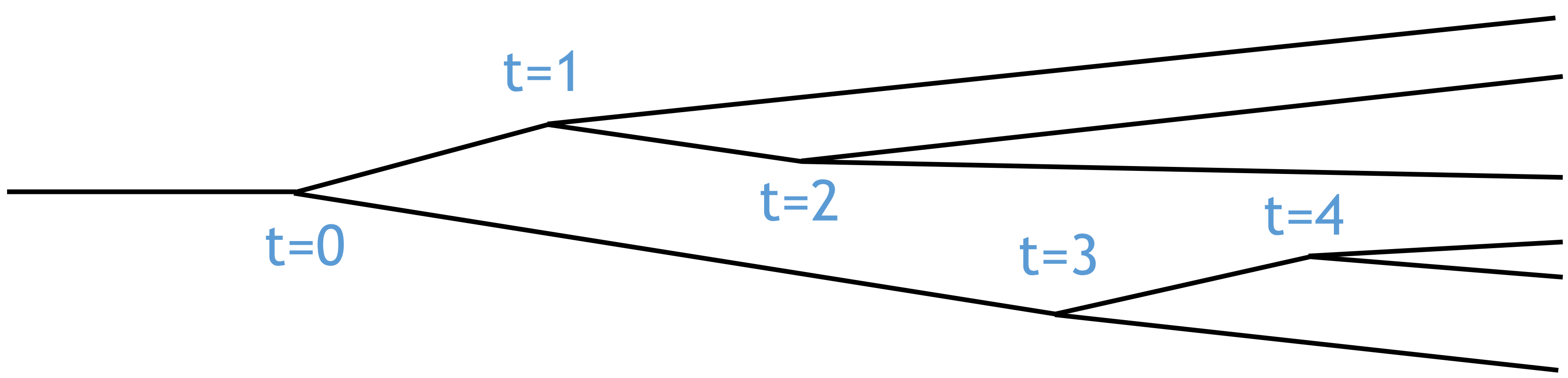
RNNs are perfect for modeling jet evolution!

figure by François Deloche

PART 1: OUR MODEL

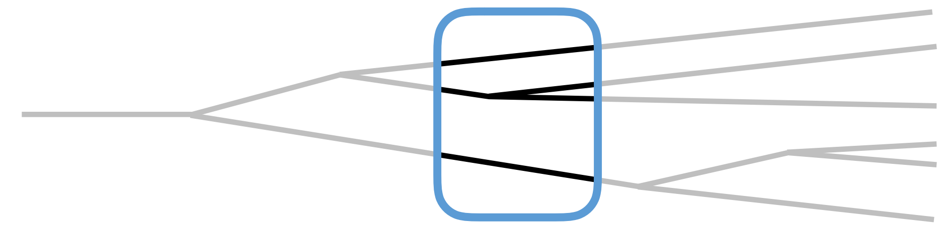
Our model computes the probability of a jet...

...as a product over "time steps" using a clustering tree

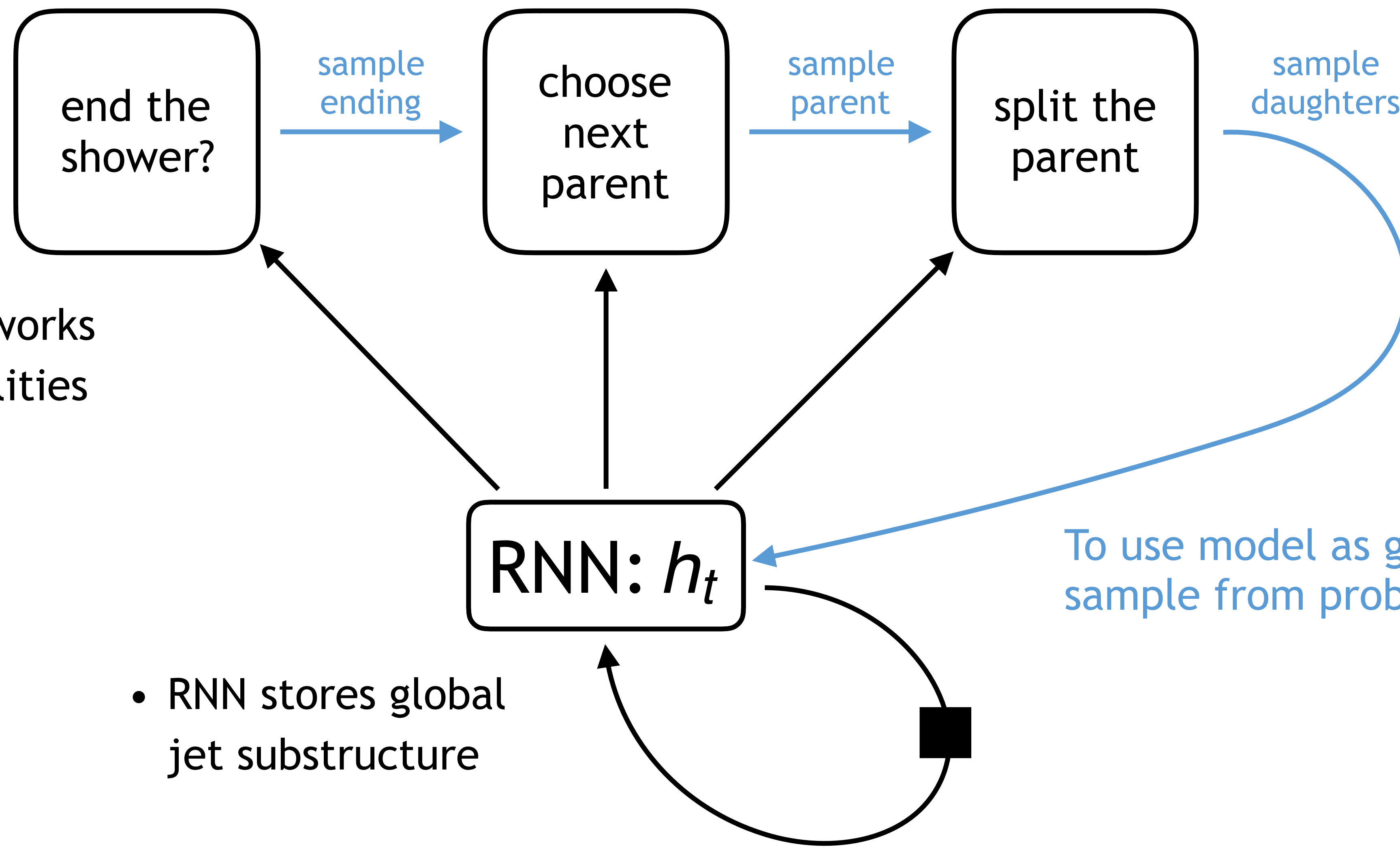


$$P(\text{jet}) = P_{t=0} \cdot P_{t=1} \cdot P_{t=2} \cdots$$

Our model at a single time step



$$P_t = P(\text{not end}) \cdot P(\text{parent} \mid \text{not end}) \cdot P(\text{daughters} \mid \text{parent})$$

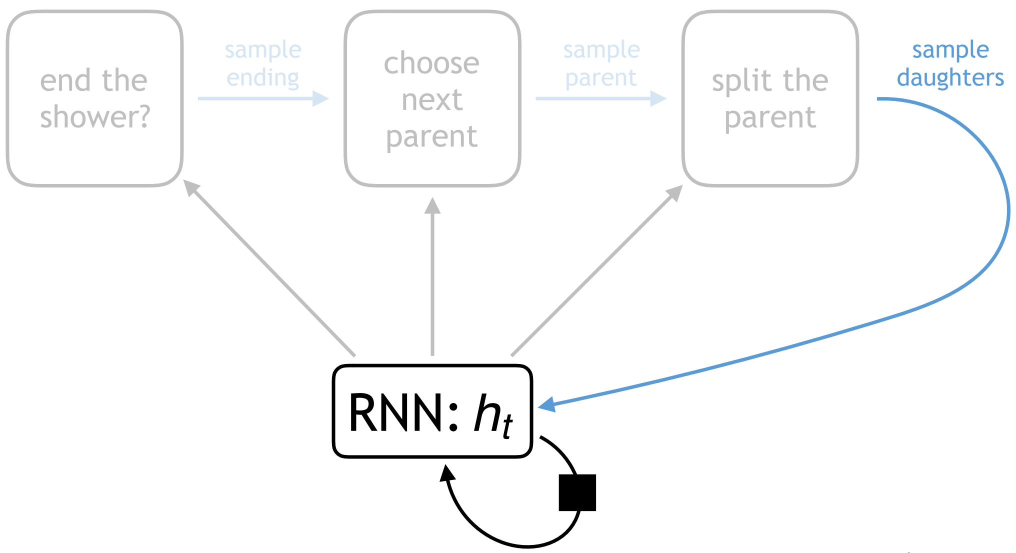


- Task specific networks compute probabilities

- RNN stores global jet substructure

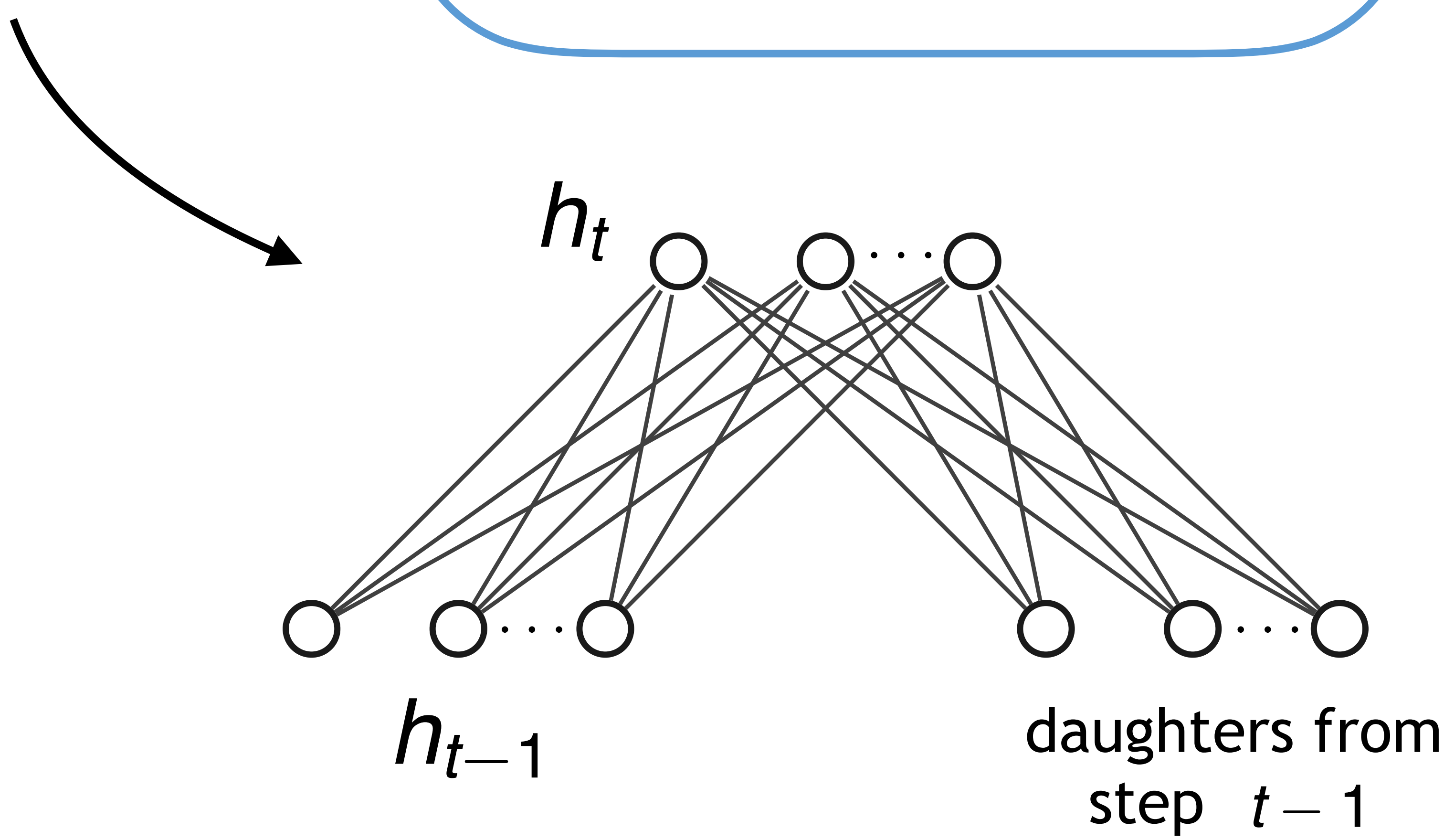
To use model as generator, sample from probabilities

RNN builds representation of global structure



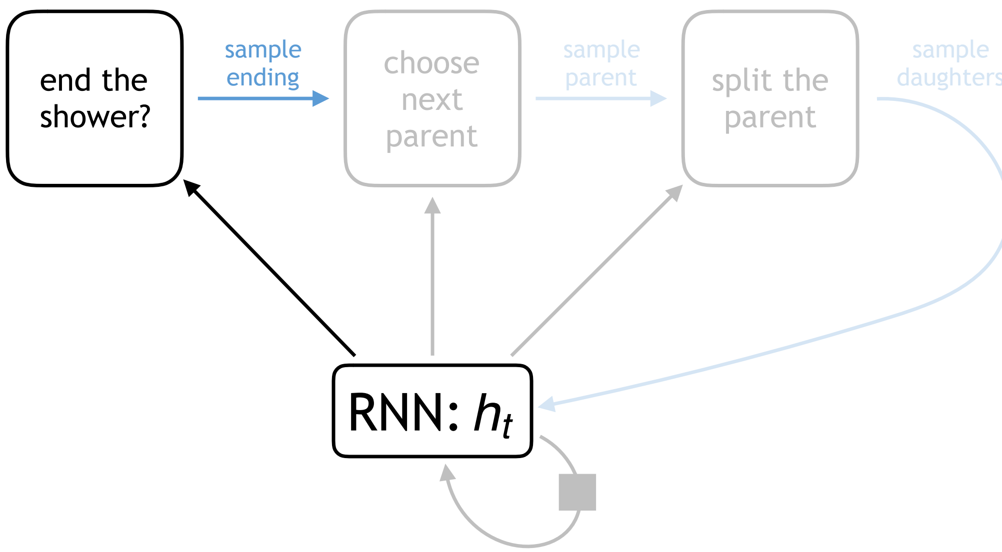
- Only latest daughters and h_{t-1} feed forward to time step t

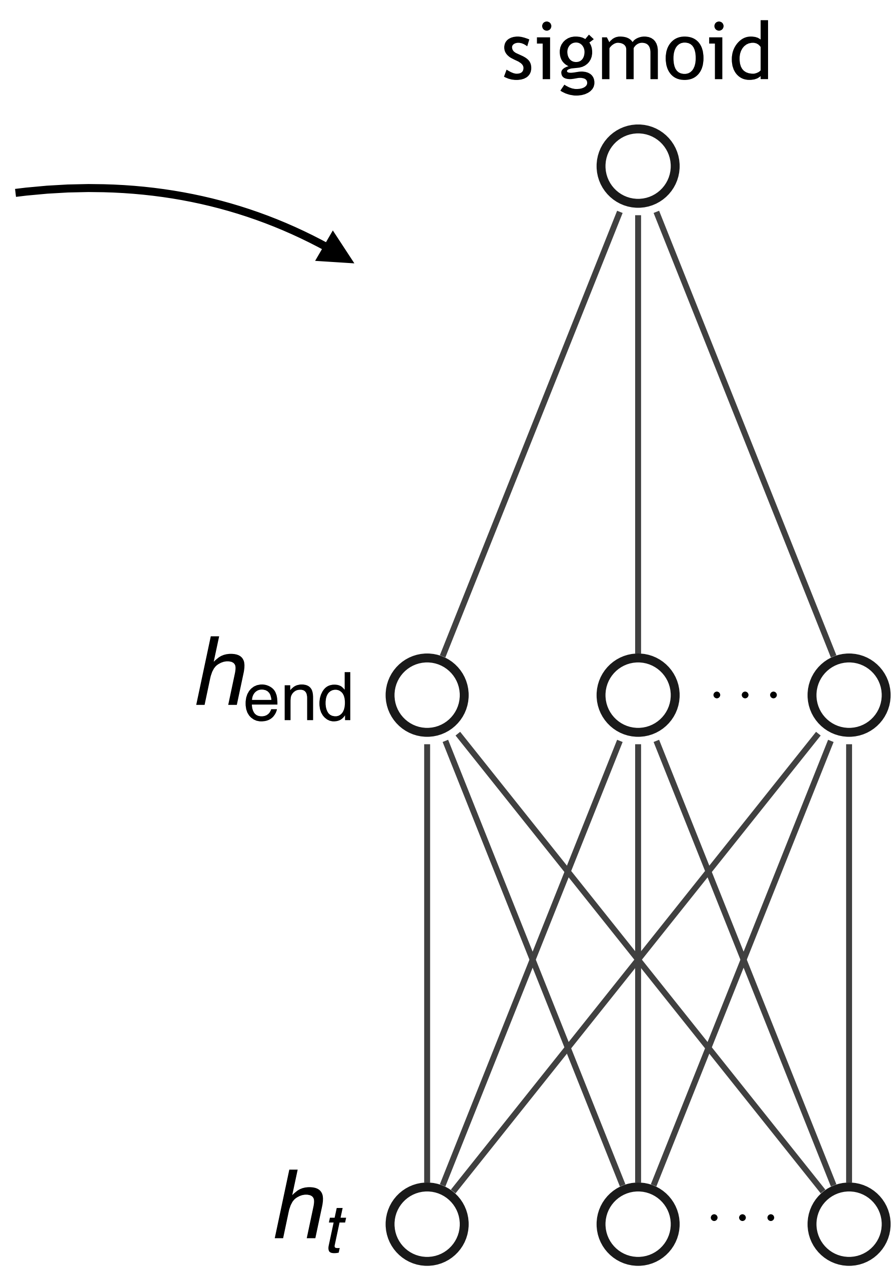
- Rely on recurrent h_t to build representation of global jet substructure



- $\dim(h_t) = 100$

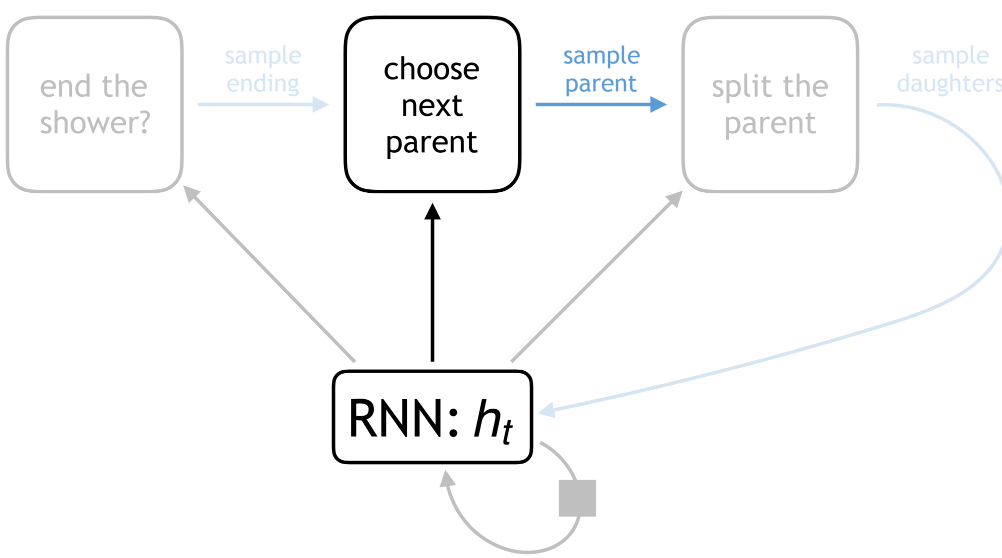
Task specific network #1 $\Longrightarrow$ probability shower will end

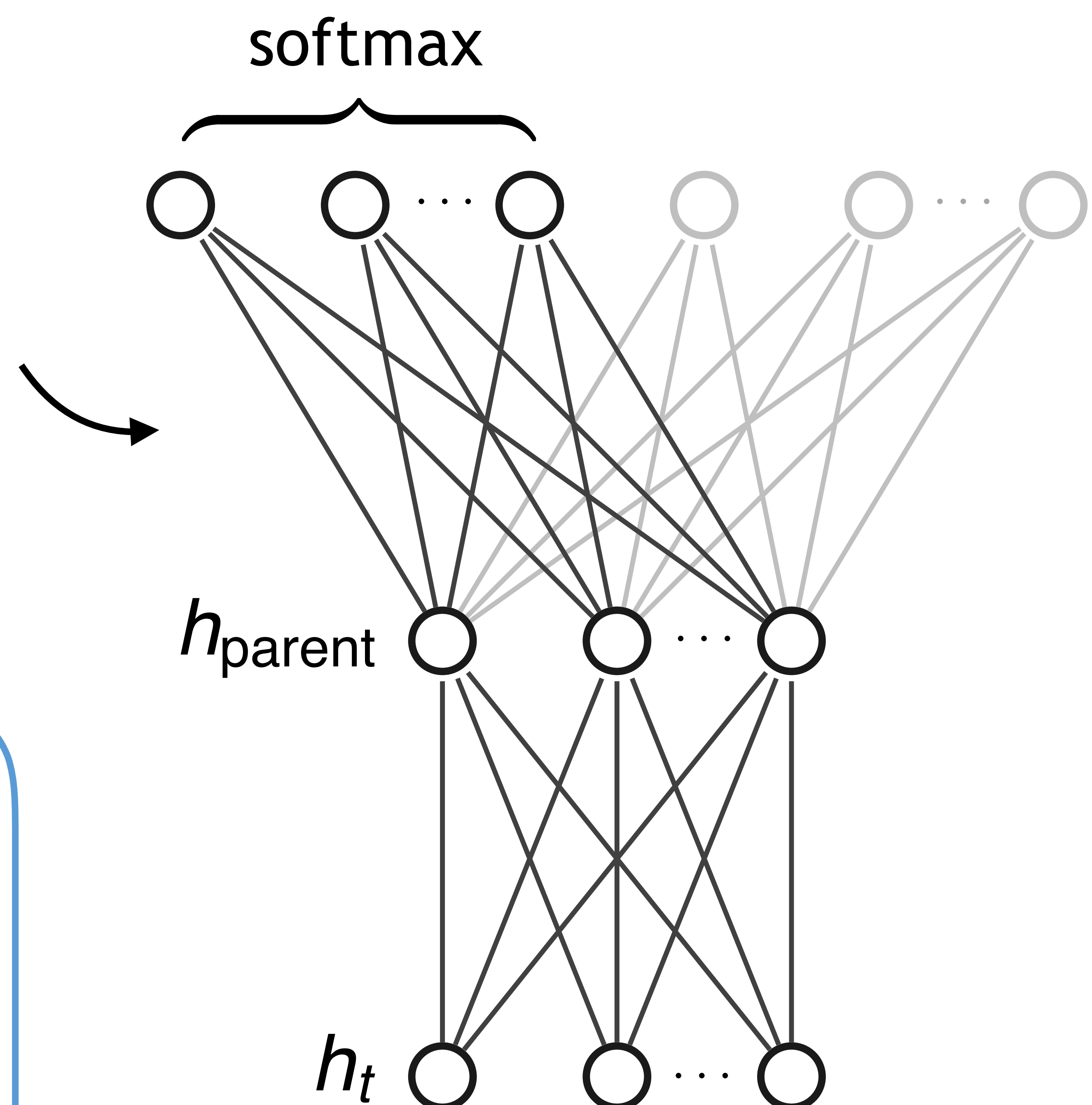




- For jet generation, sample from sigmoid
- $\dim(h_{\text{end}}) = 10$

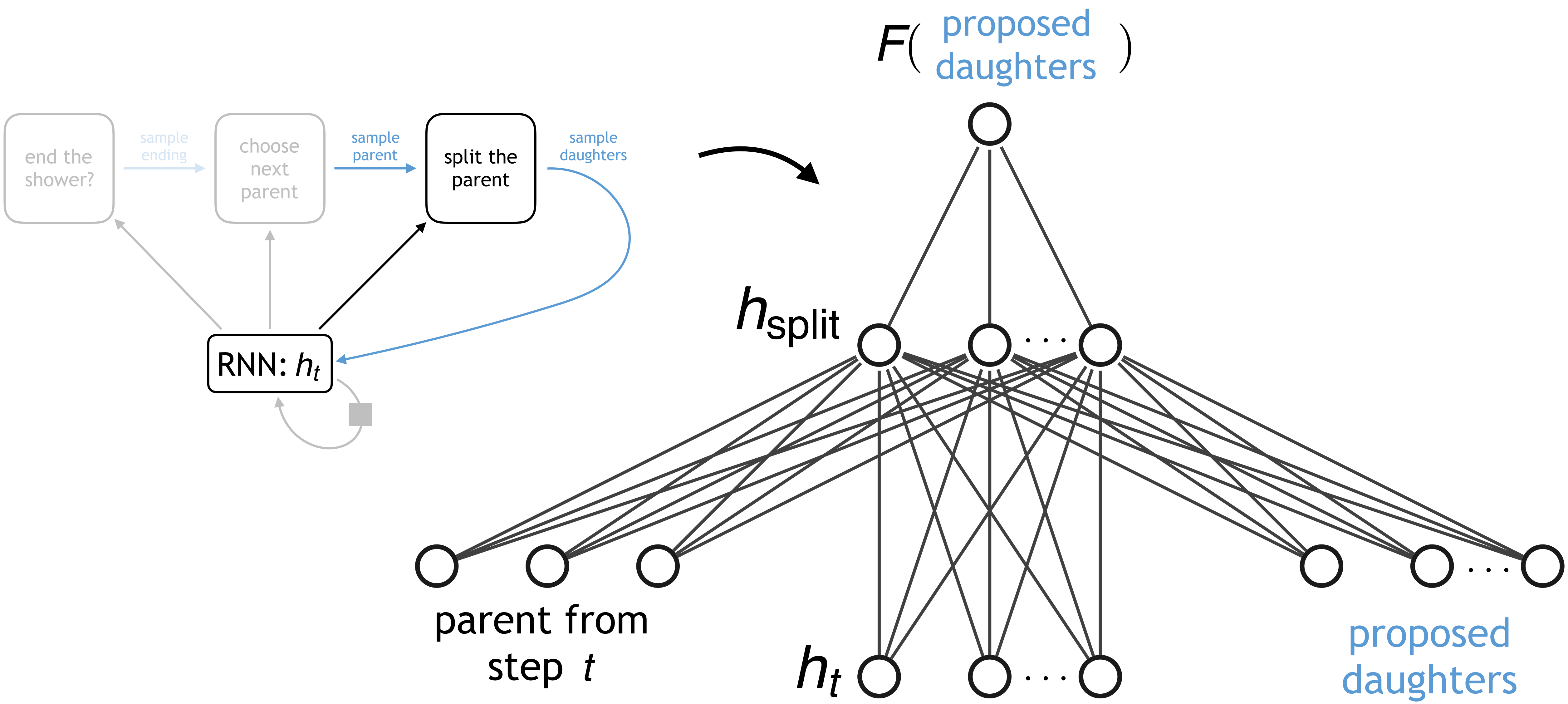
Task specific network #2 $\Longrightarrow$ probability of next parent





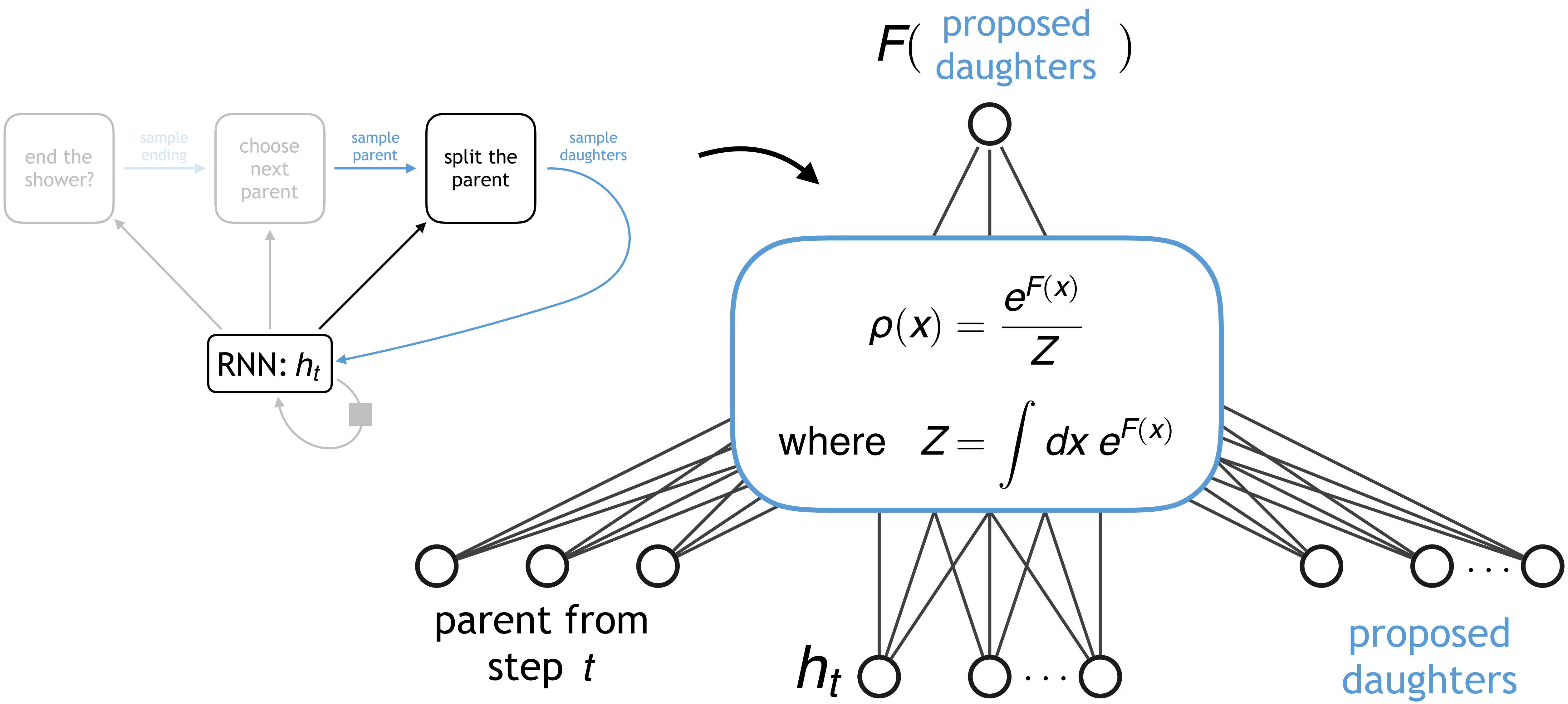
- Nodes in top layer $\Longrightarrow$ particles in energy ordered state
- For jet generation, sample from softmax
- $\dim(h_{\text{parent}}) = 10$

Task specific network #3 $\Longrightarrow$ probability over daughters



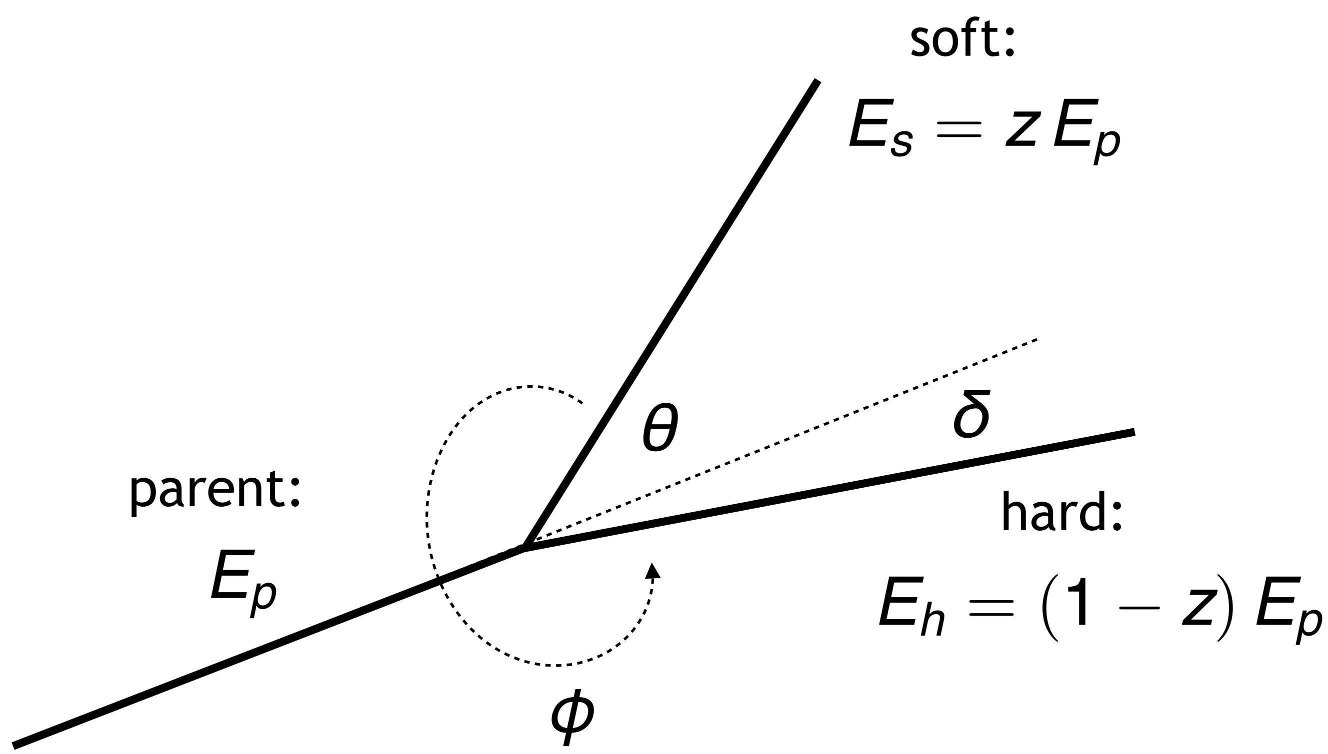
- For jet generation, sample daughters from F
- dim(h_{split}) = 100
- 25,000 parameters in all

Task specific network #3 $\Longrightarrow$ probability over daughters



- For jet generation, sample daughters from F
- dim(h_{split}) = 100
- 25,000 parameters in all

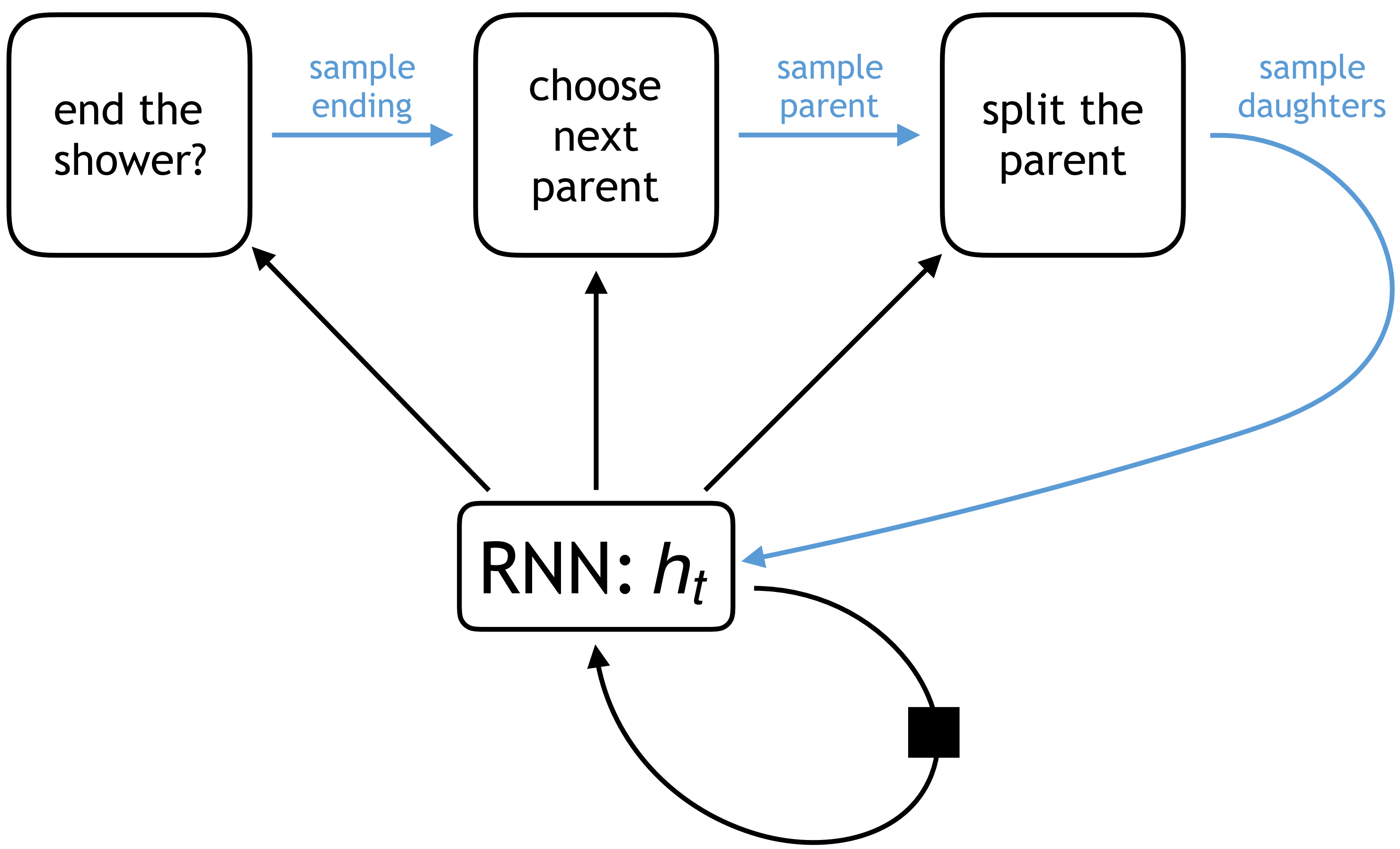
Parametrization of daughter momenta: z, θ, ϕ, δ



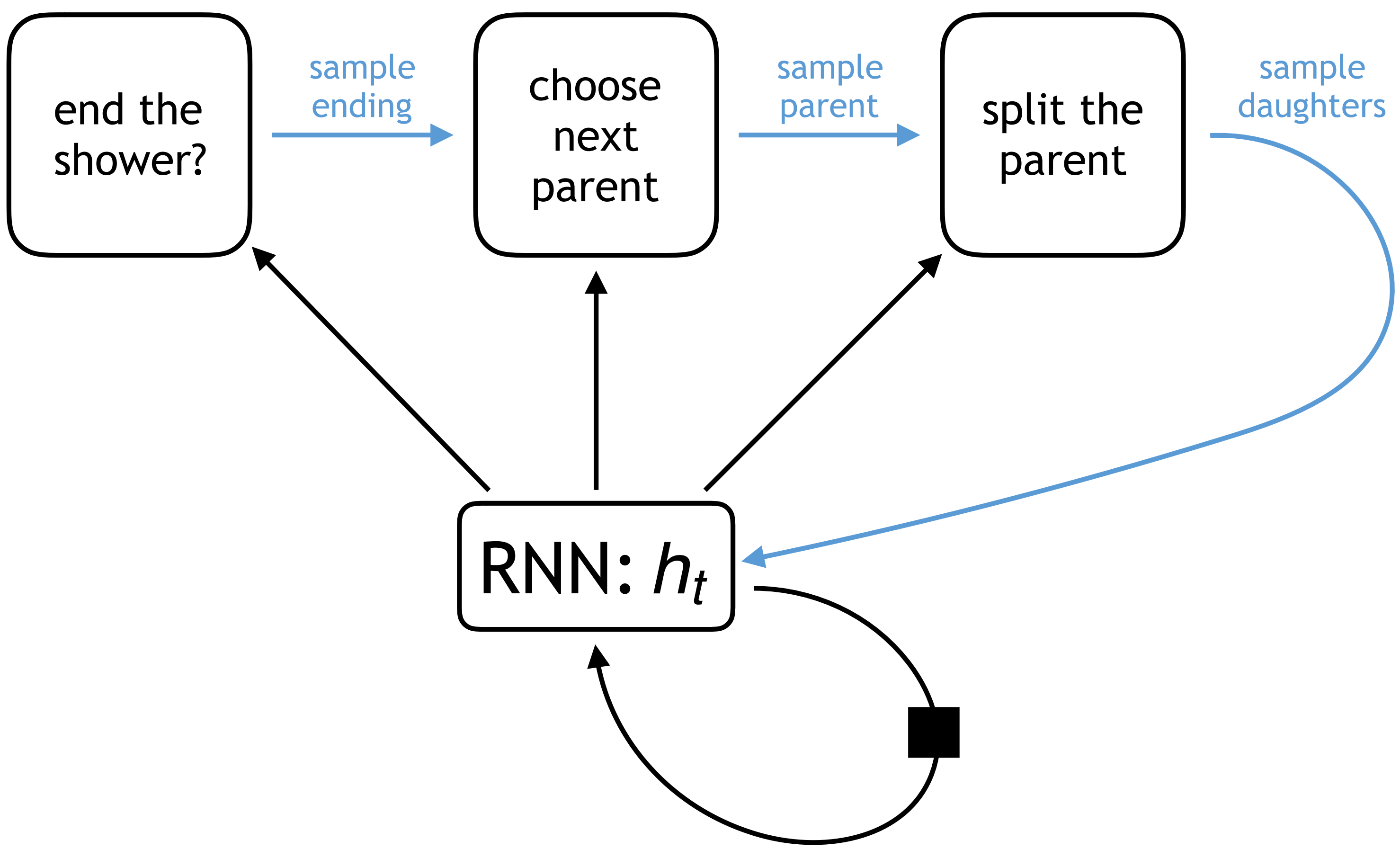
INTERMISSION

PART 2: TRAINING & RESULTS

Reminder: our model at a single time step

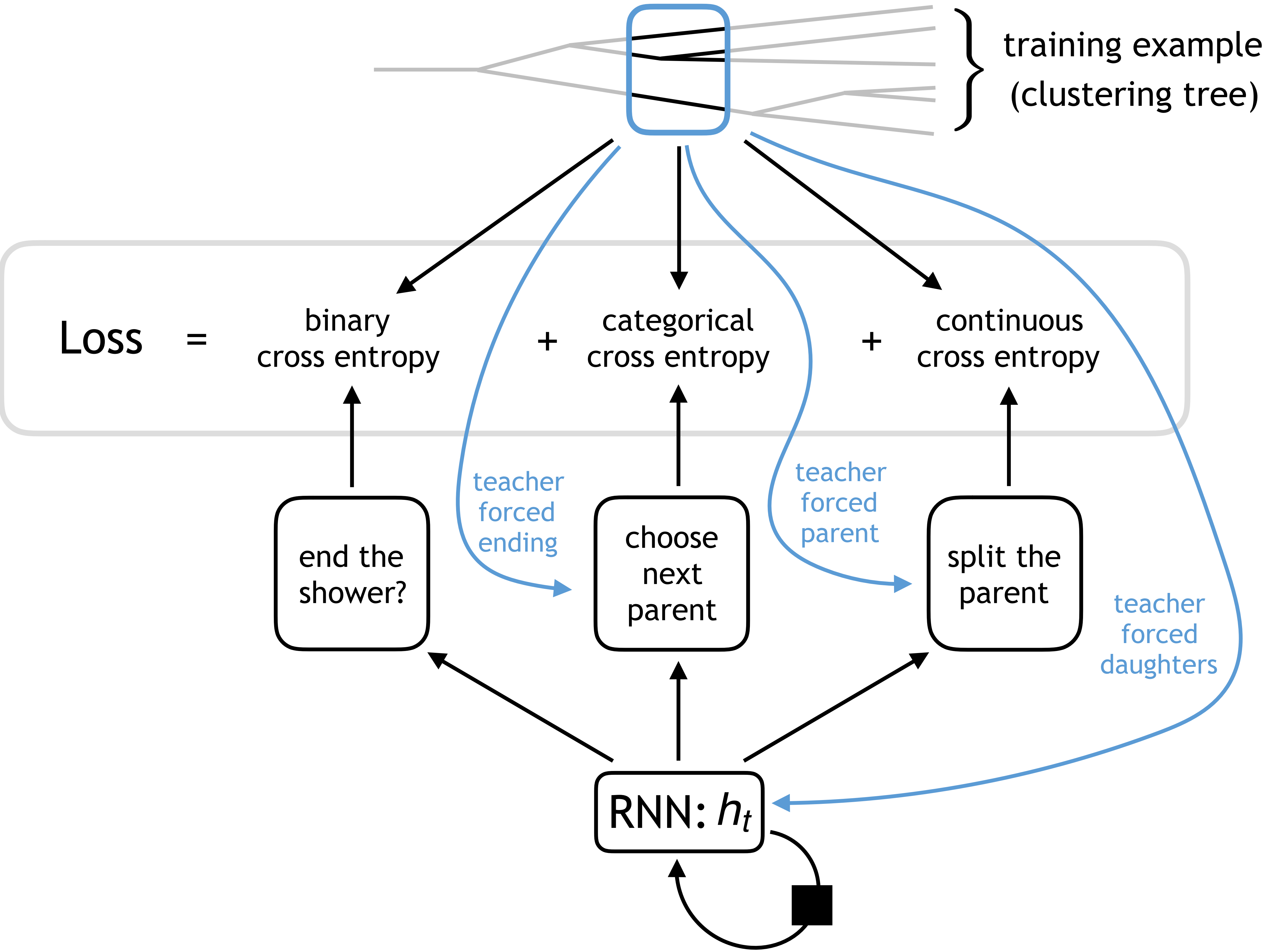


Reminder: our model at a single time step



We do NOT train model by comparing generated jets to training examples

Training our model



Details about our training set

- Training set from Pythia 8 as proof-of-concept:
  500,000 e^+e^- events at $E_{\text{CM}} = 10$ TeV

$$E_{\text{jet}} \approx 5 \text{ TeV} \quad \text{with} \quad R_{\text{jet}} = \pi/2$$

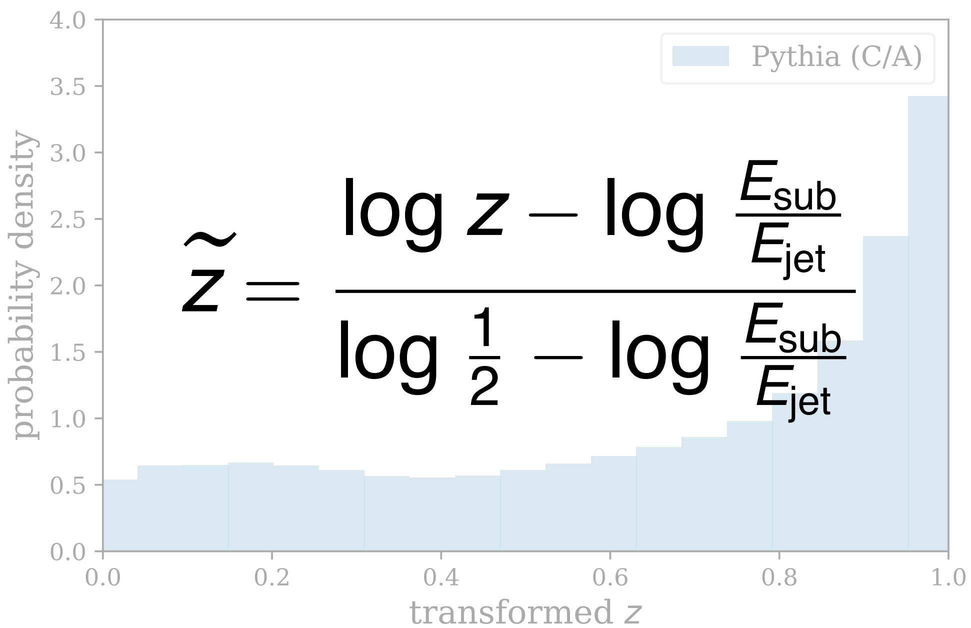
- Jet constituents reclustered with C/A to obtain trees

$$E_{\text{sub}} = 1 \text{ GeV} \quad \text{with} \quad R_{\text{sub}} = 0.1$$

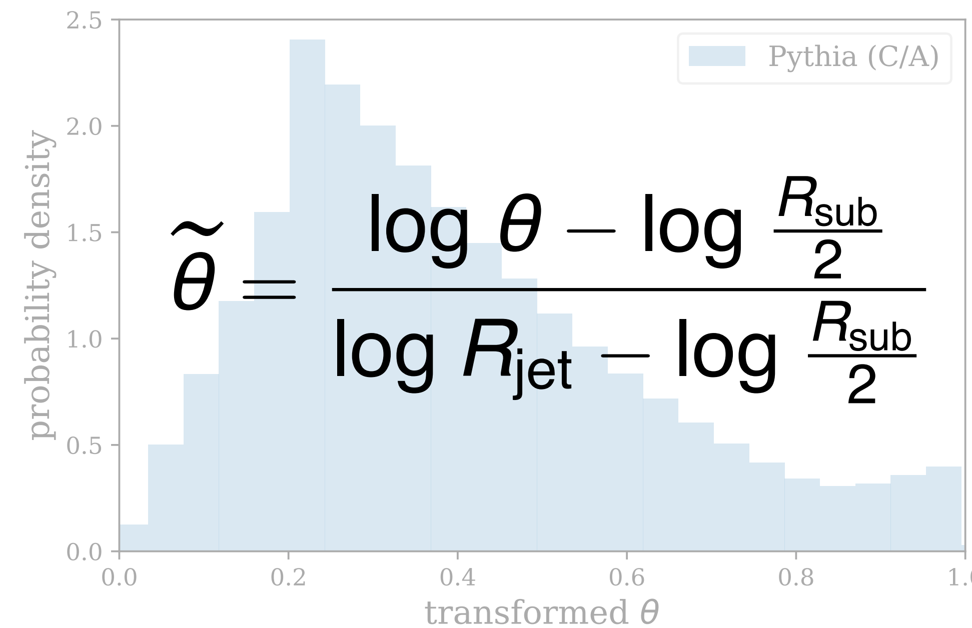
- Model and training implemented in Theano

- Train using final state momenta only,
  all methods repeatable on LHC data

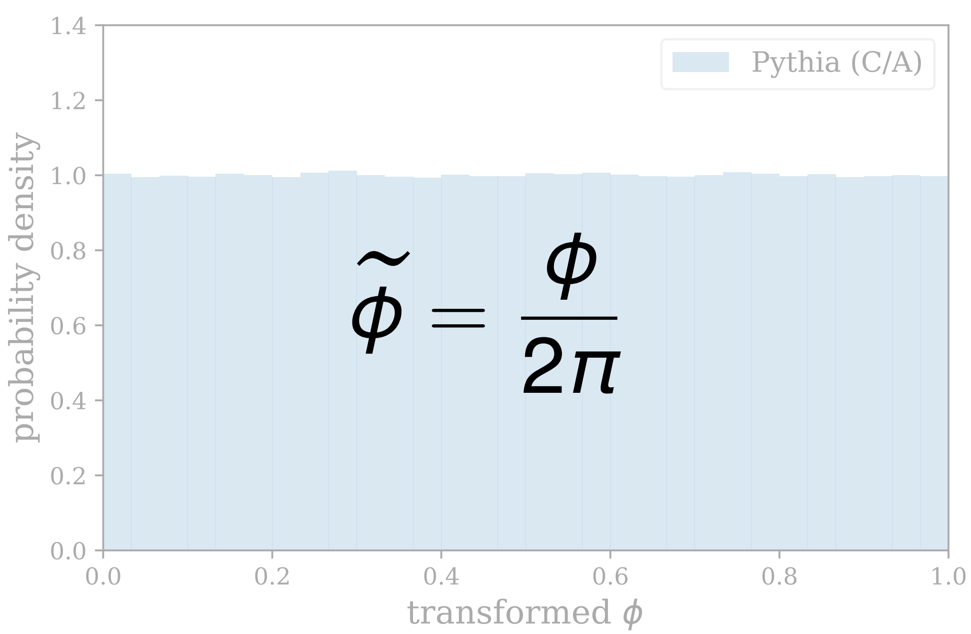
Coordinate transformation $\Longrightarrow$ features in [0,1]



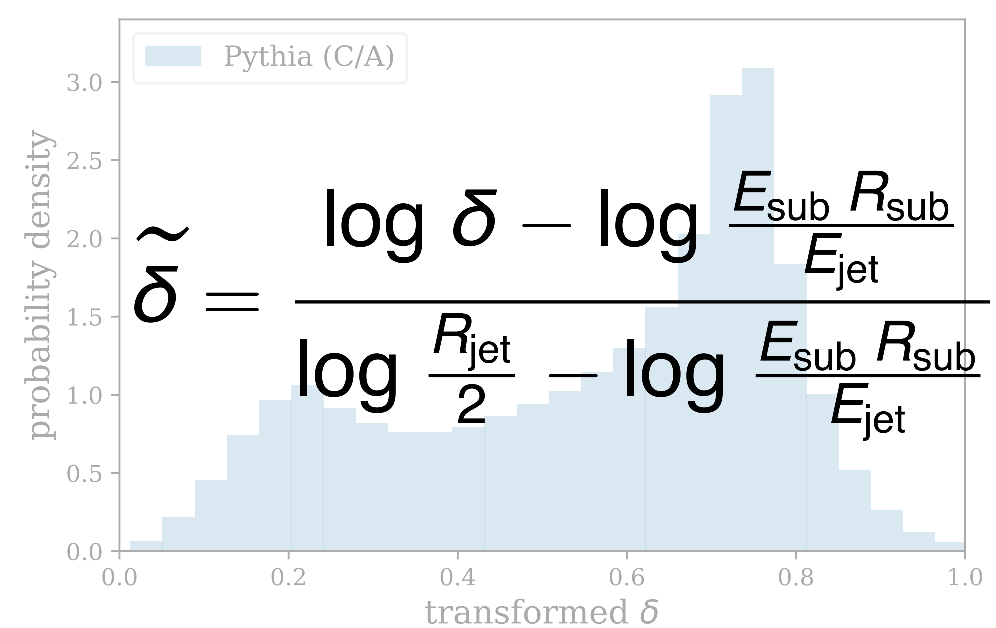
$$\tilde{z} = \frac{\log z - \log \frac{E_{\text{sub}}}{E_{\text{jet}}}}{\log \frac{1}{2} - \log \frac{E_{\text{sub}}}{E_{\text{jet}}}}$$



$$\tilde{\theta} = \frac{\log \theta - \log \frac{R_{\text{sub}}}{2}}{\log R_{\text{jet}} - \log \frac{R_{\text{sub}}}{2}}$$

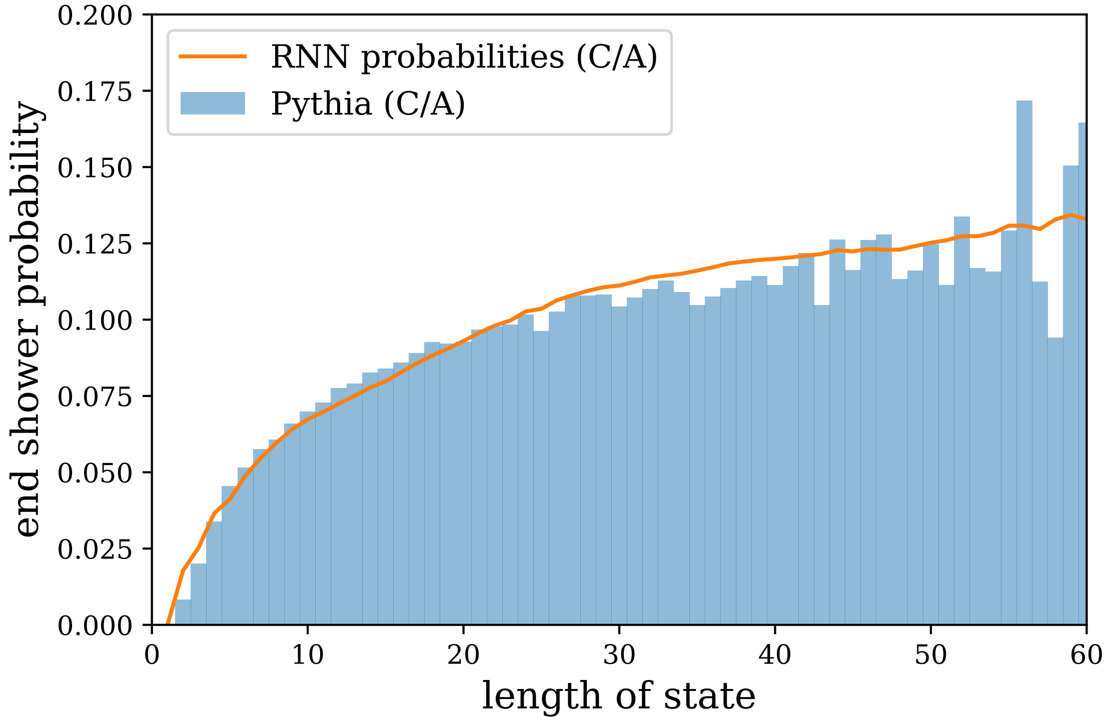


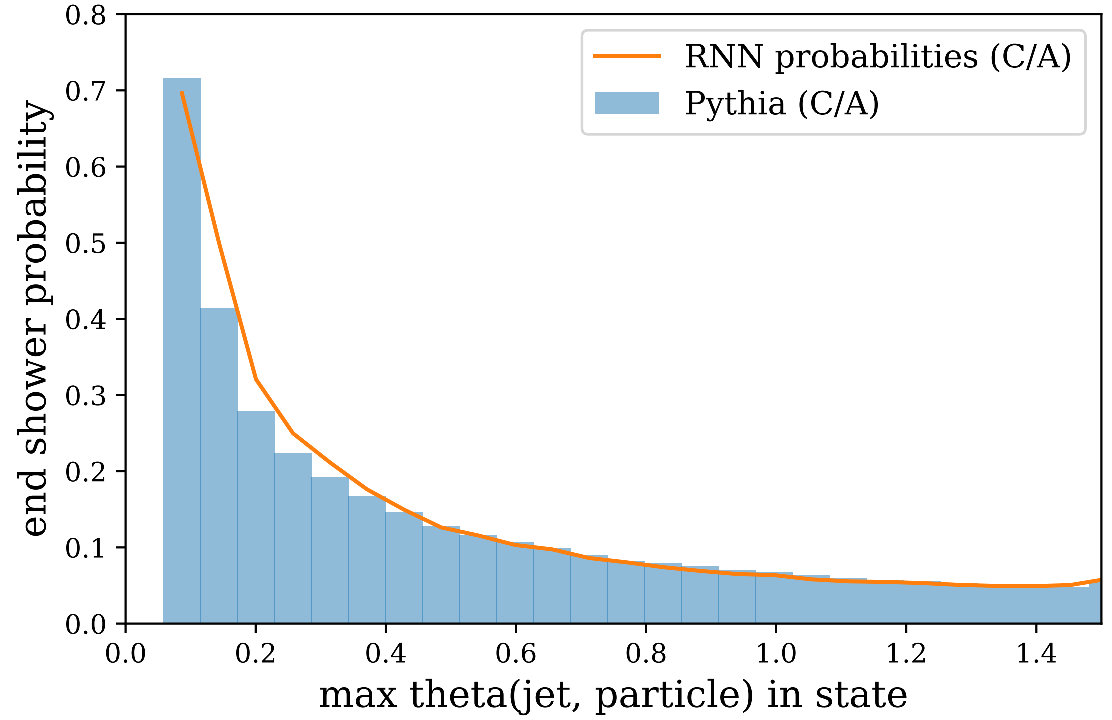
$$\tilde{\phi} = \frac{\phi}{2\pi}$$



$$\tilde{\delta} = \frac{\log \delta - \log \frac{E_{\text{sub}}\, R_{\text{sub}}}{E_{\text{jet}}}}{\log \frac{R_{\text{jet}}}{2} - \log \frac{E_{\text{sub}}\, R_{\text{sub}}}{E_{\text{jet}}}}$$

Looking inside our model: ending the shower

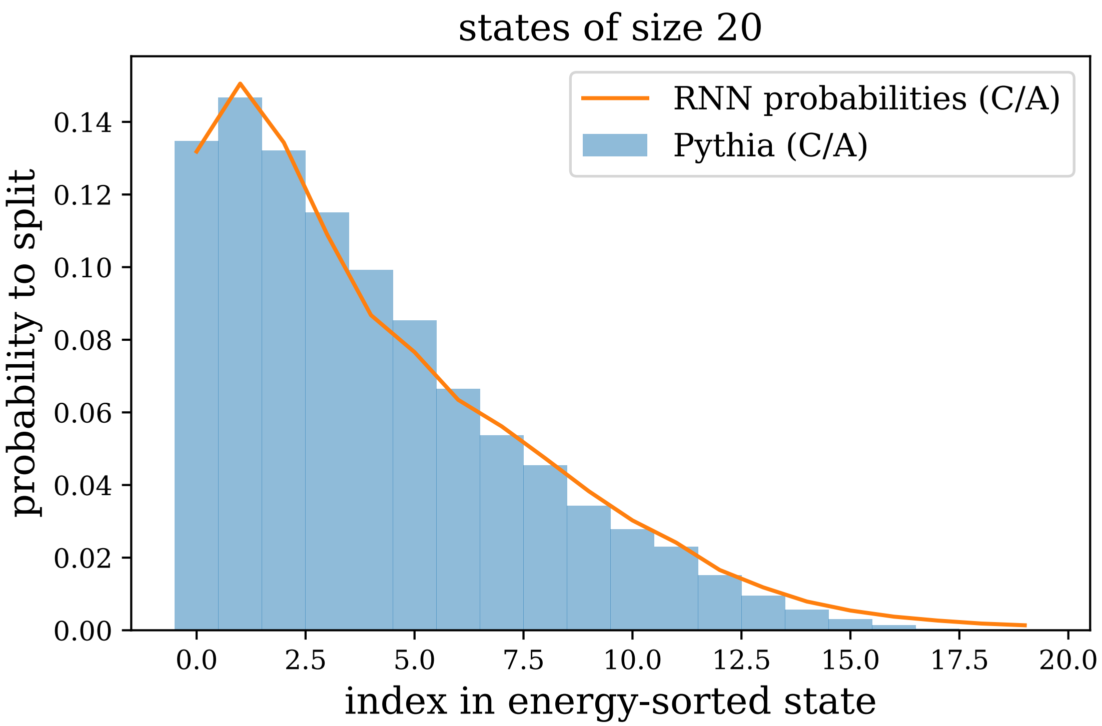


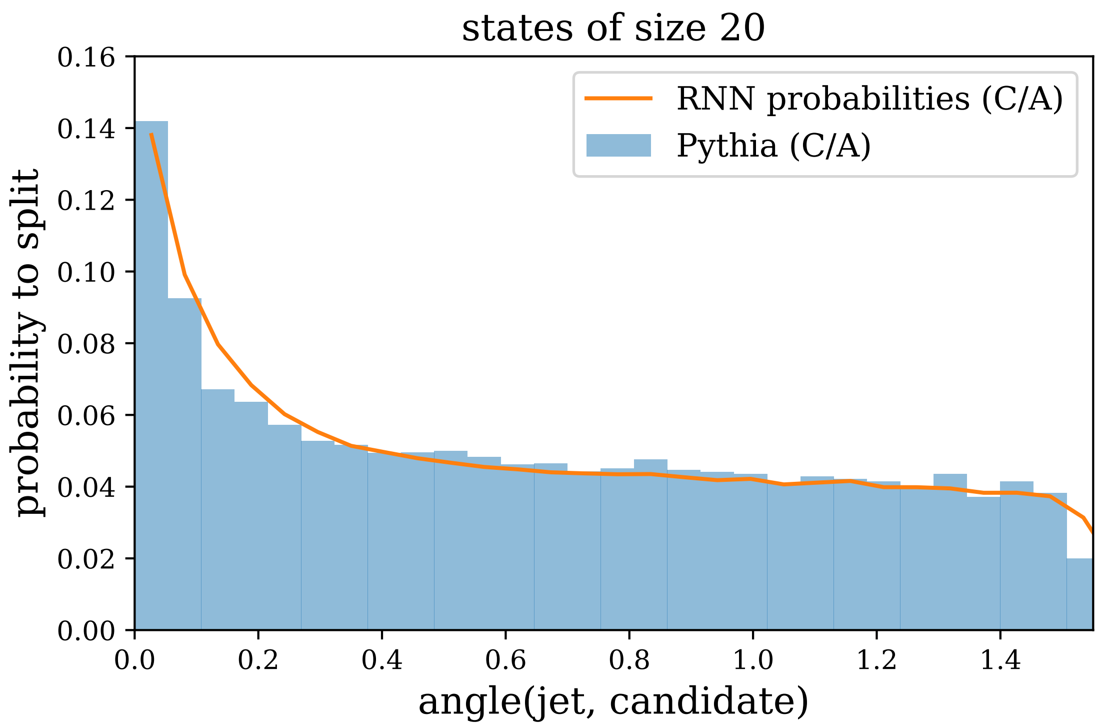


- Model probabilities computed on training examples, then compared to histogram of training data

- Curves averaged over all time steps and all jets

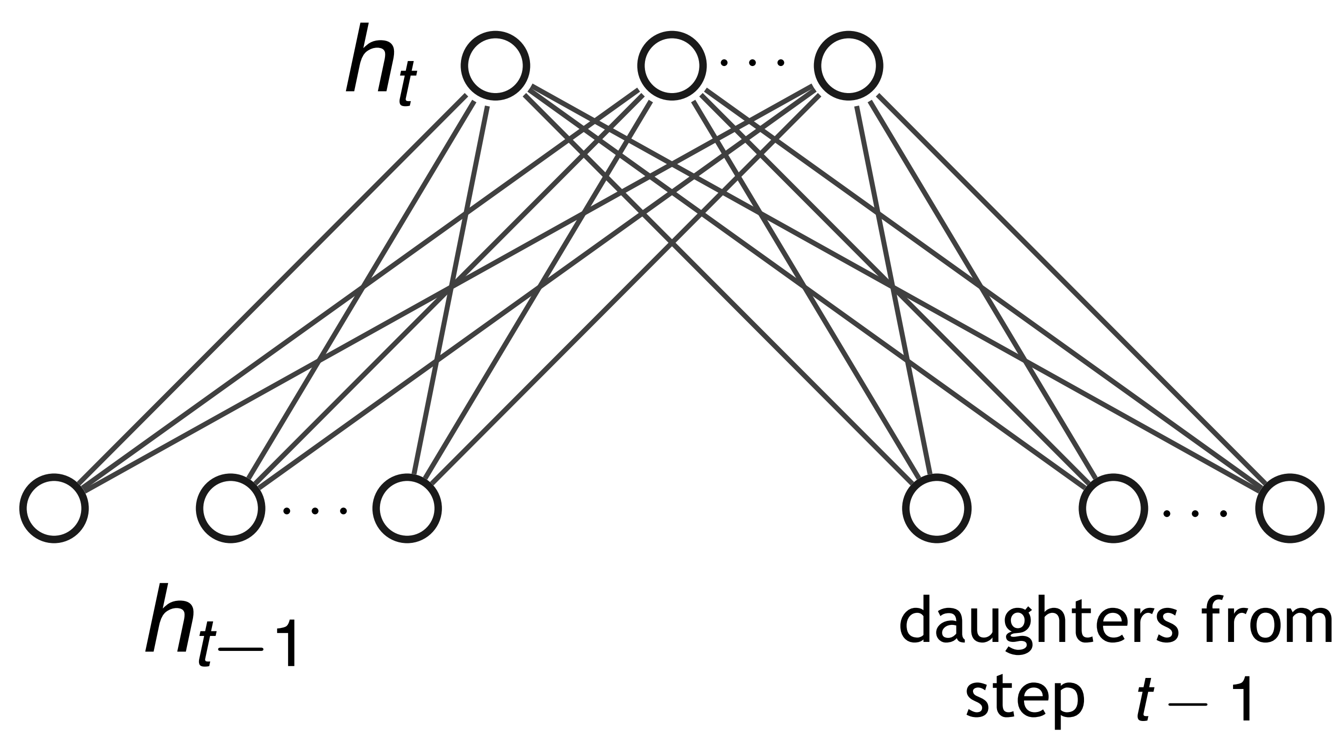
Looking inside our model: choosing the next parent



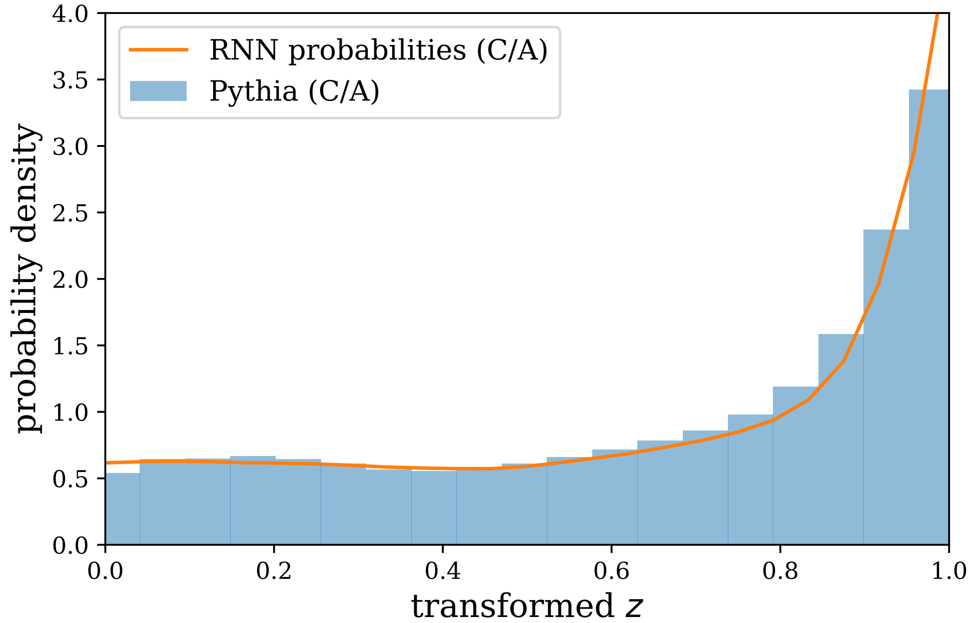


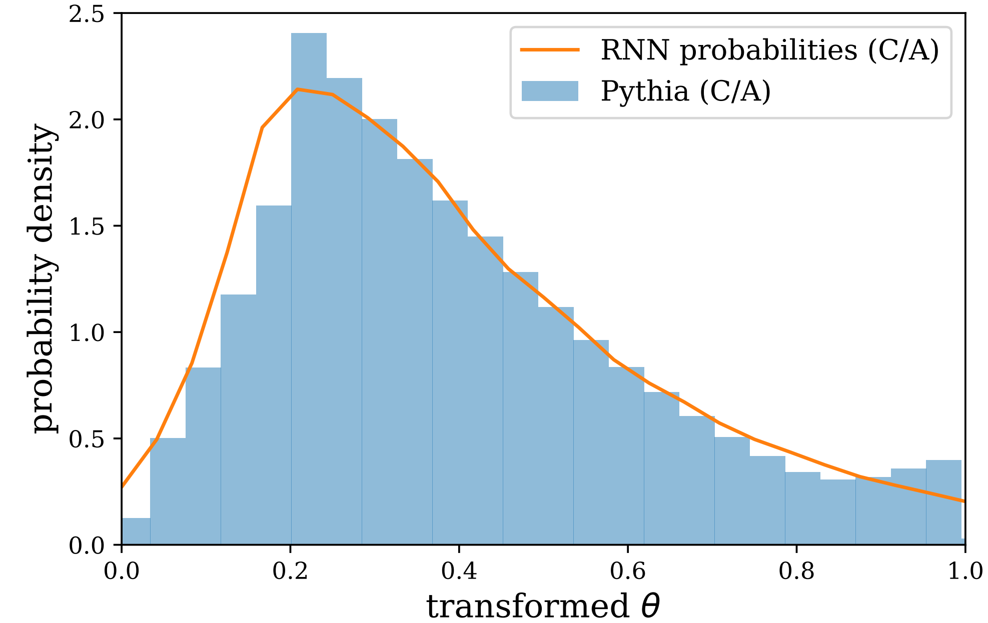
- Model probabilities computed on training examples, then compared to histogram of training data
- Curves averaged over time step t=20 in all jets

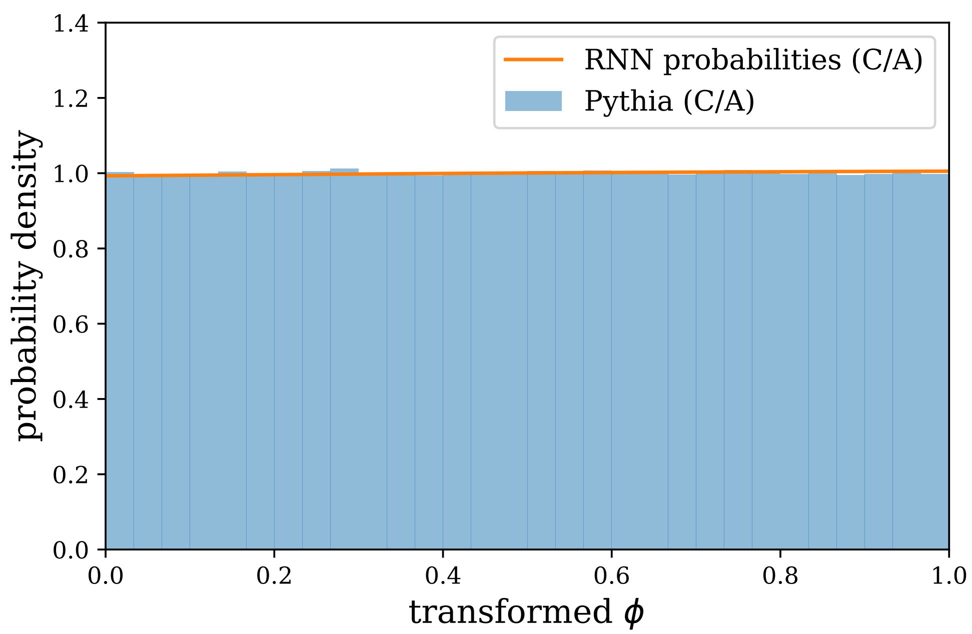
RNN really stores global structure!

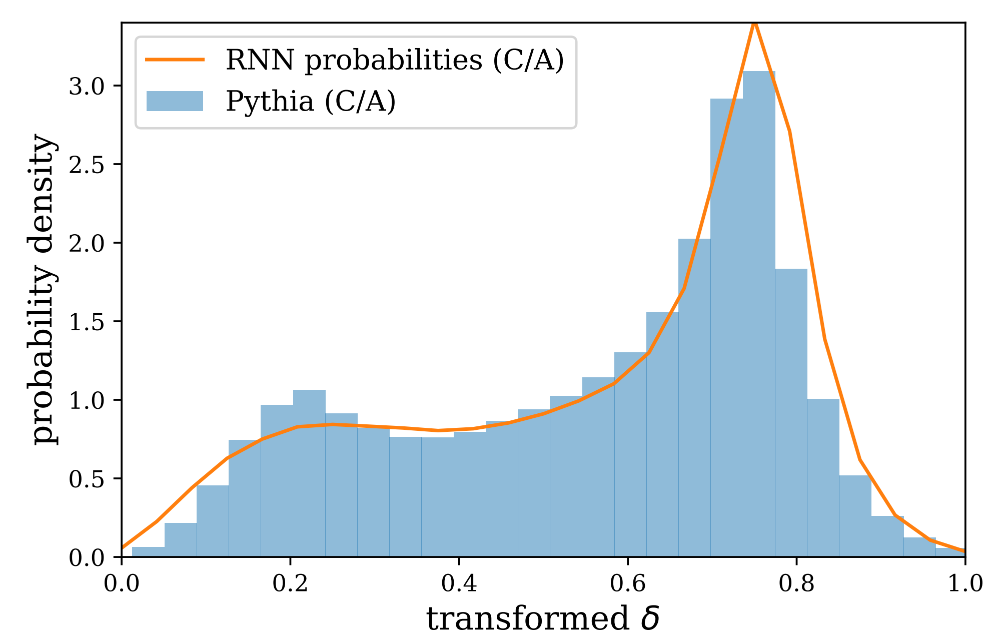


Looking inside our model: splitting into daughters





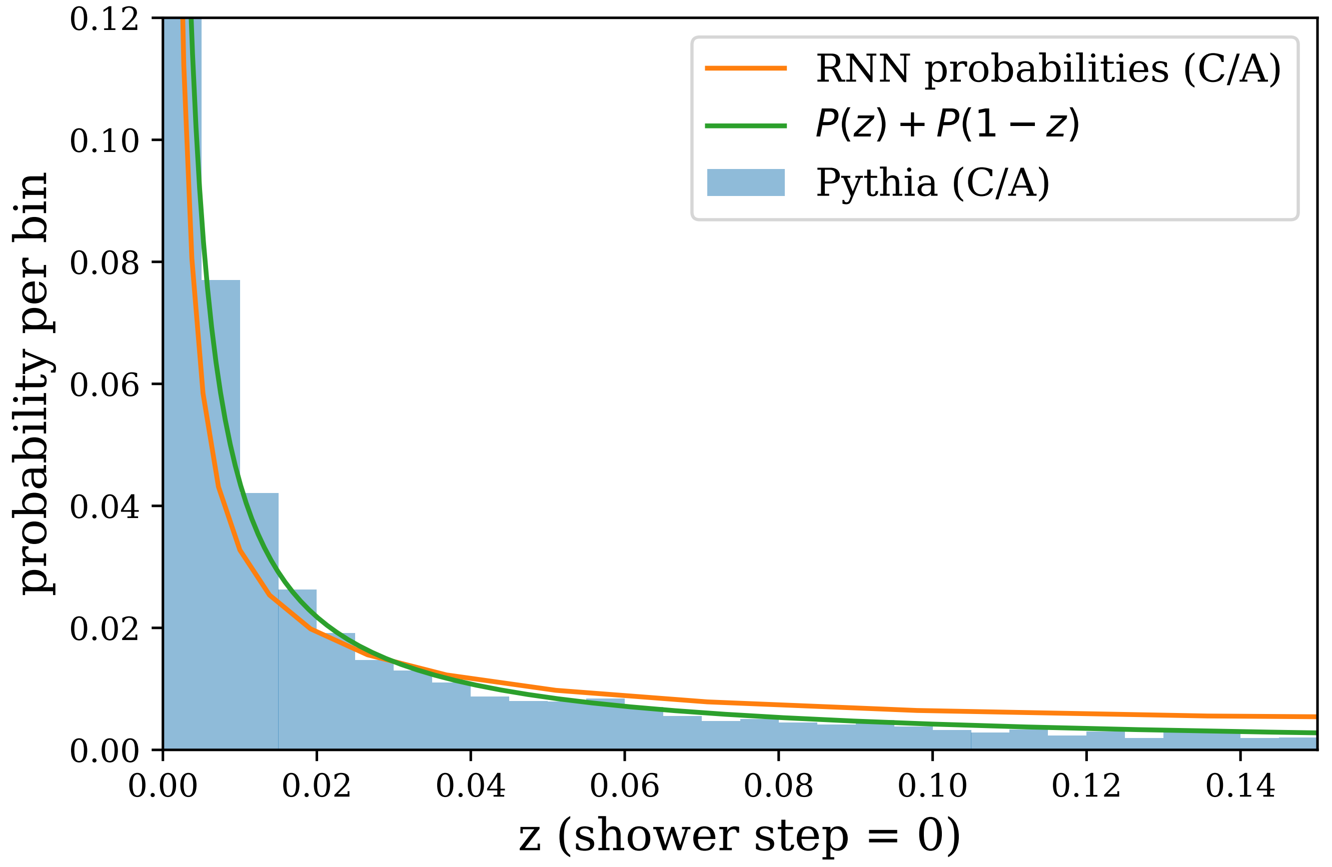




Our model's effective splitting function

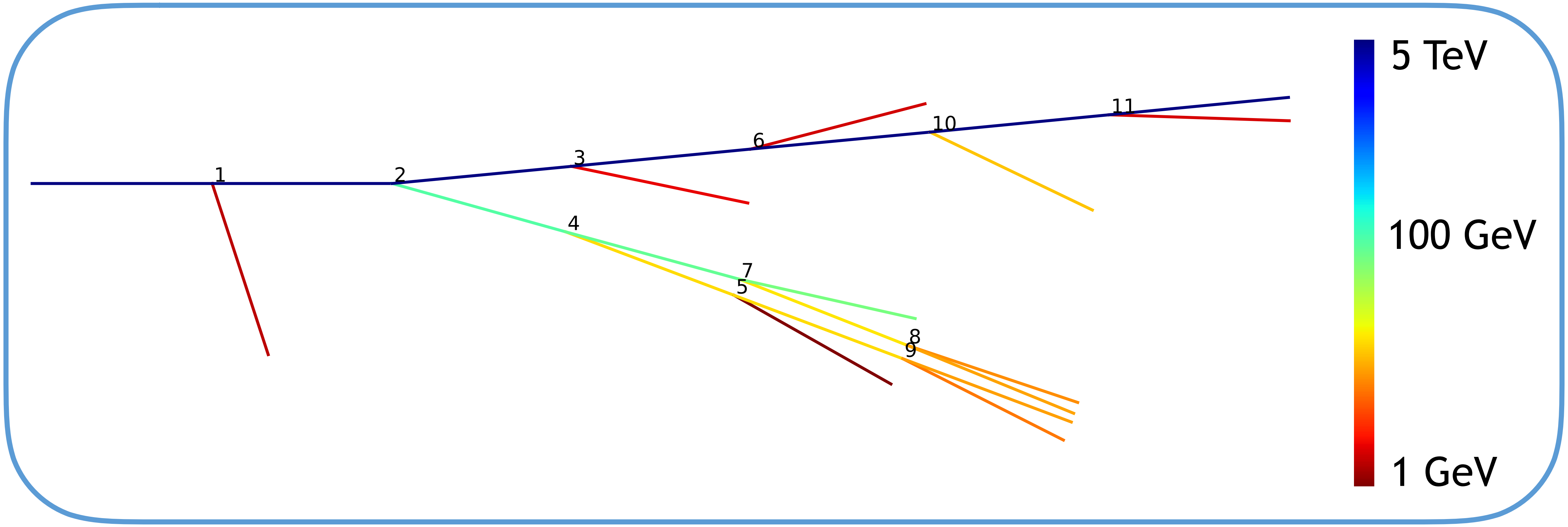
- Distribution of z's at each splitting of C/A tree are closely related to QCD splitting function:

$$P(z) = \frac{1 + (1 - z)^2}{z}$$



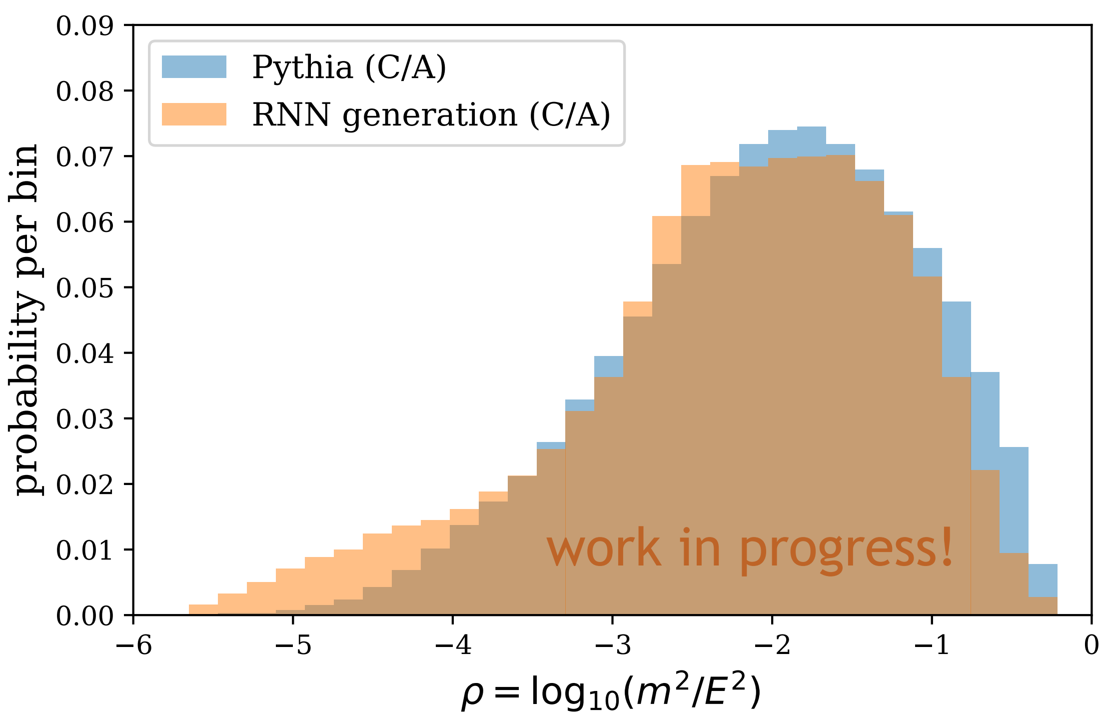
Our model as a shower generator

- Given seed momentum (energy & direction) one can sample final states from our model

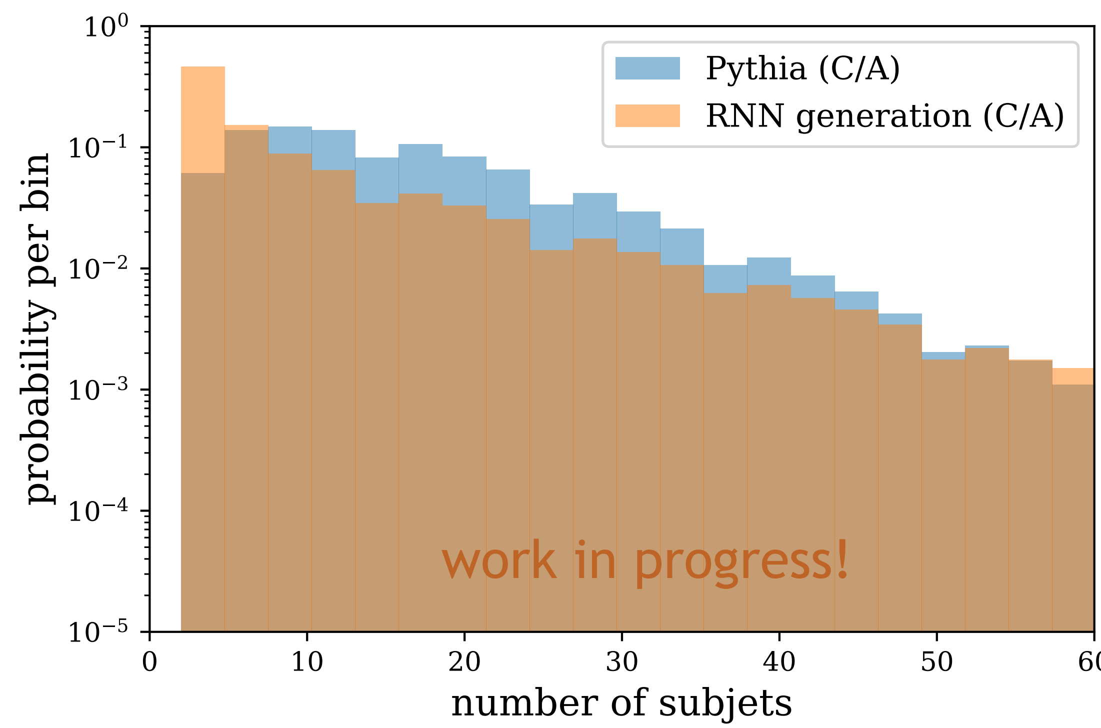


Generated distributions of final-state observables

jet mass

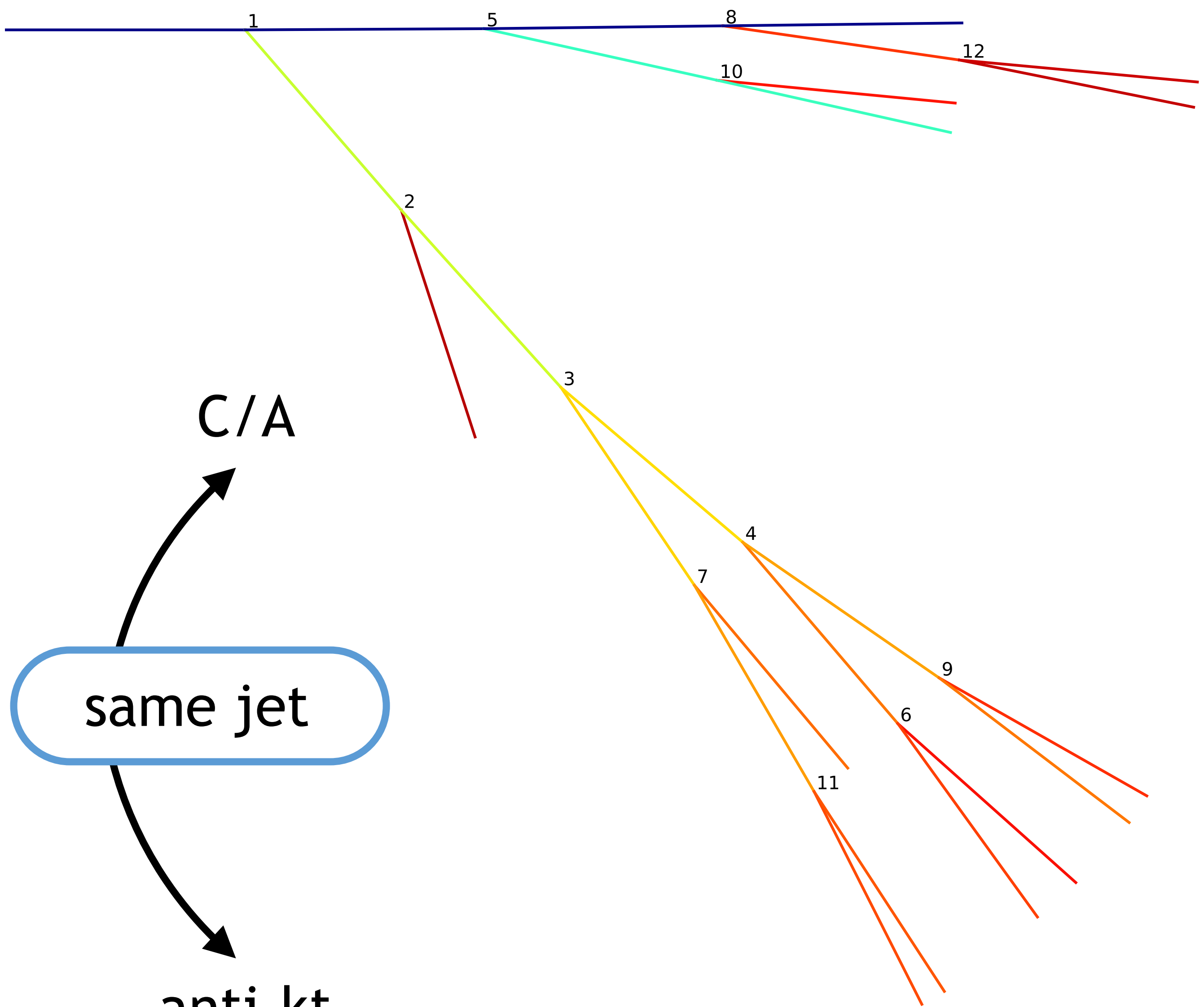


subjet multiplicity



- Jet generation is a rigorous test of all model components: small errors at early time steps can evolve into large mistakes later

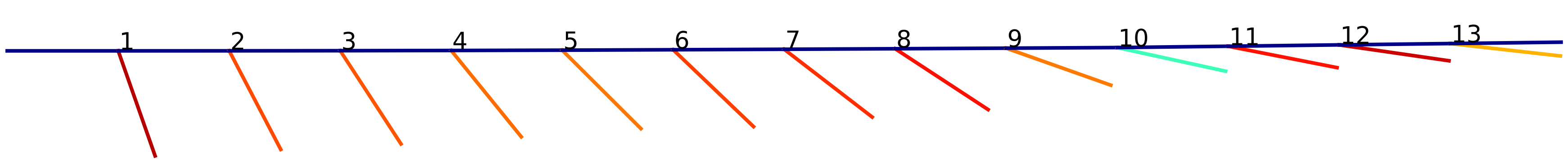
Can we build an anti-kt shower?



C/A

same jet

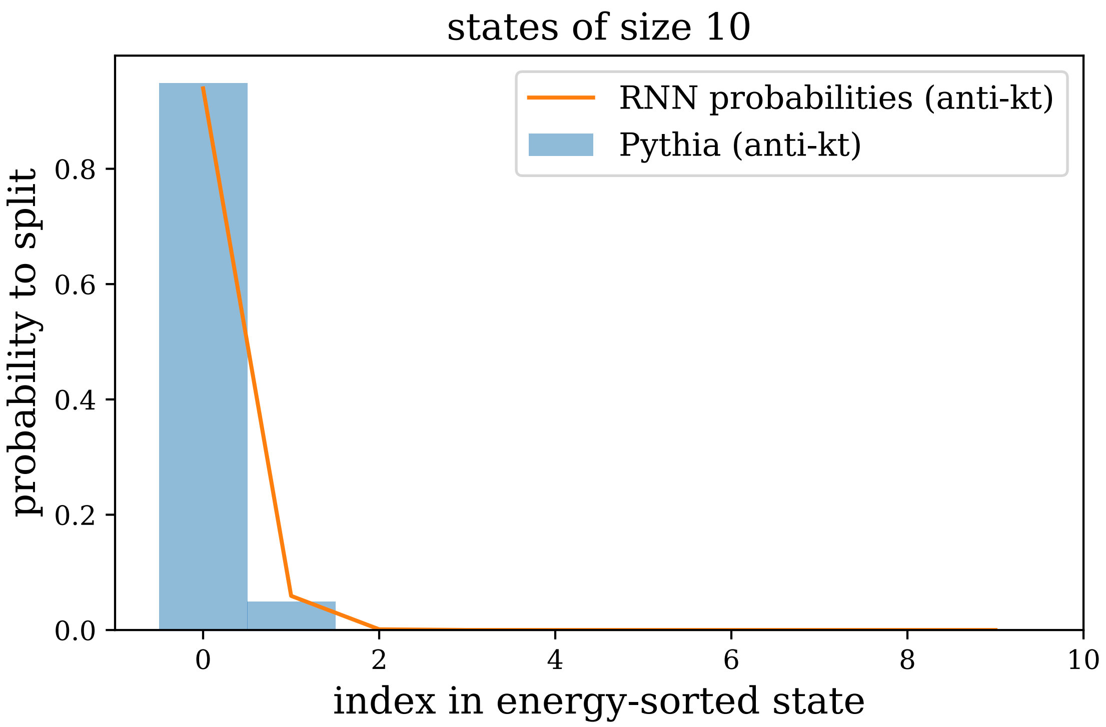
anti-kt

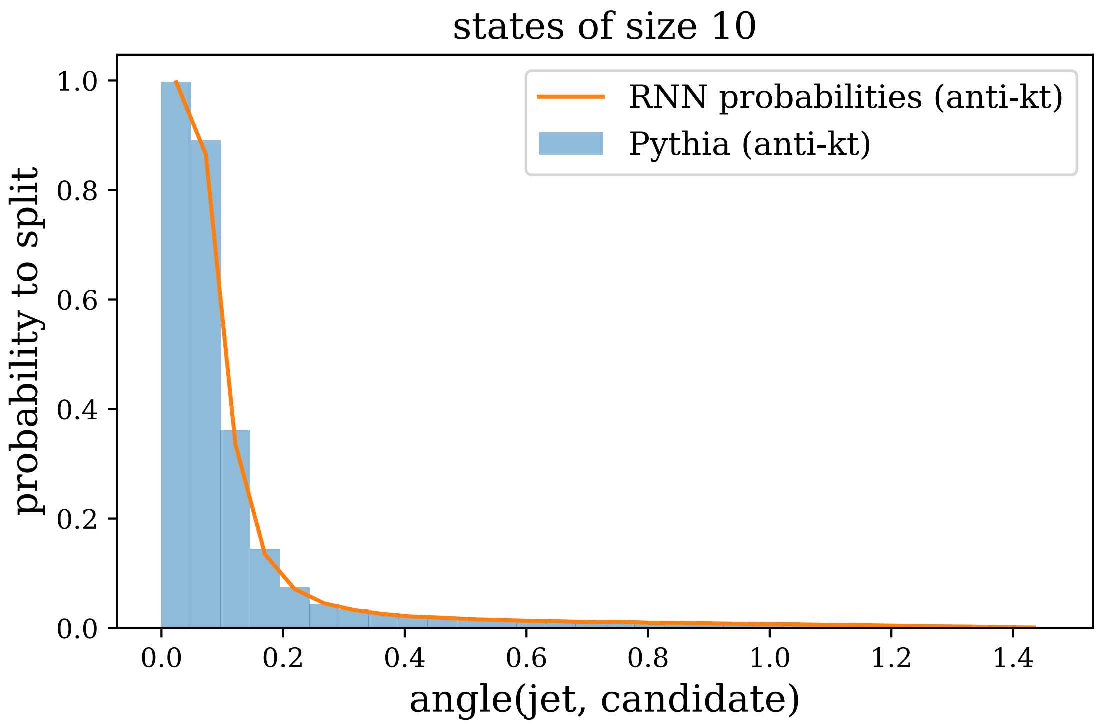


- Roughly, anti-kt clusters hardest particle with nearest, usually soft particle
- Traditionally, anti-kt shower impossible:
- Local $1 \to 2$ splittings ordered in anti-kt cannot reproduce collinear structure

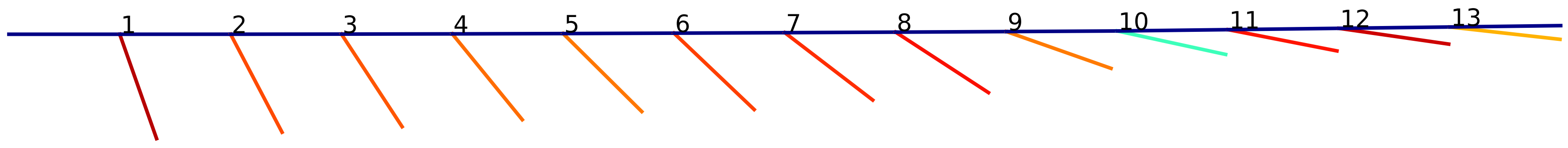
but our model has global structure!

Certain aspects of anti-kt are easy to learn…



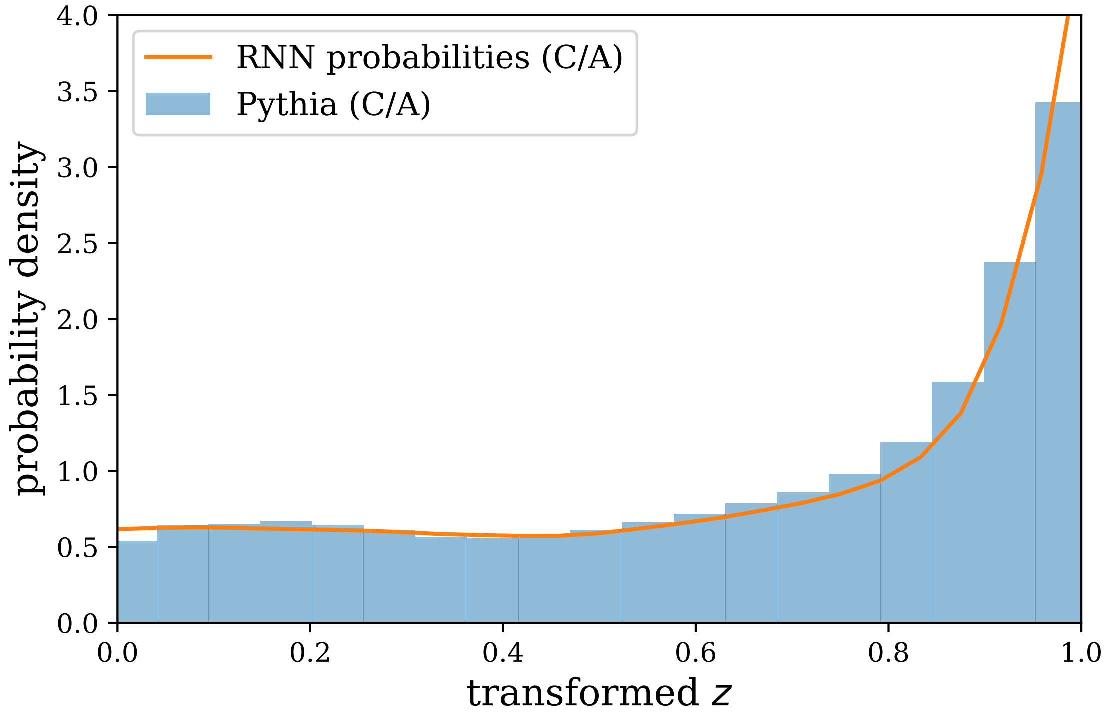


- Hardest candidate is by far most likely to split next:



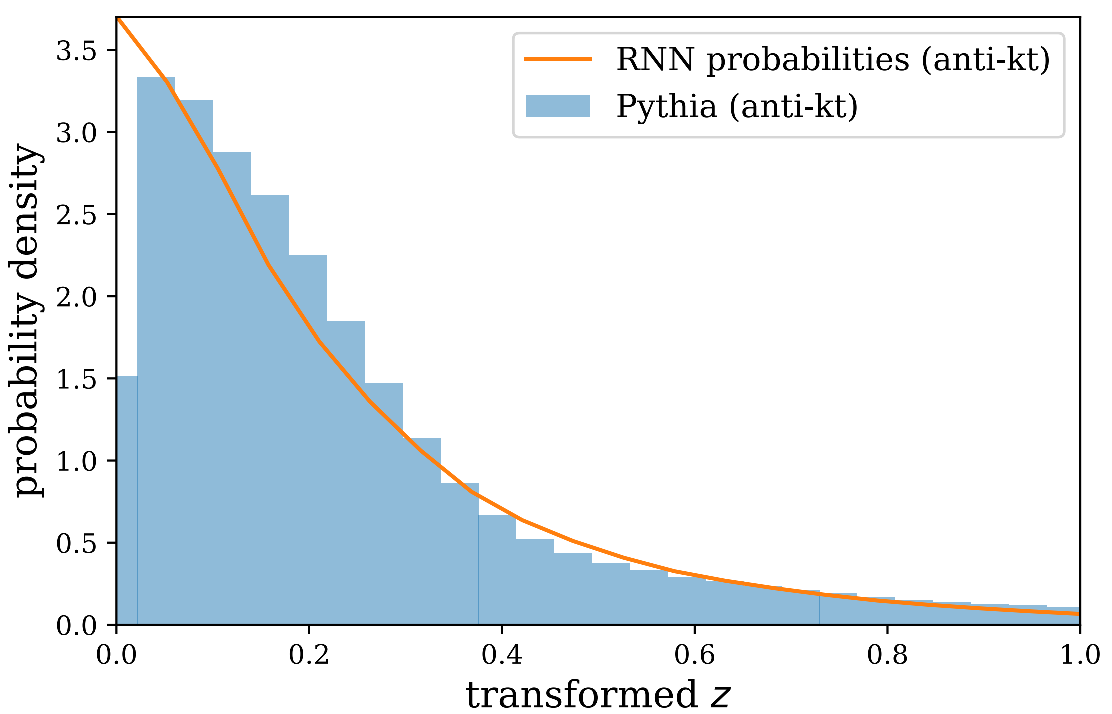
…but the physics gets hidden

C/A shower



- Distribution of z's in C/A trees $\Longrightarrow$ splitting function P(z)

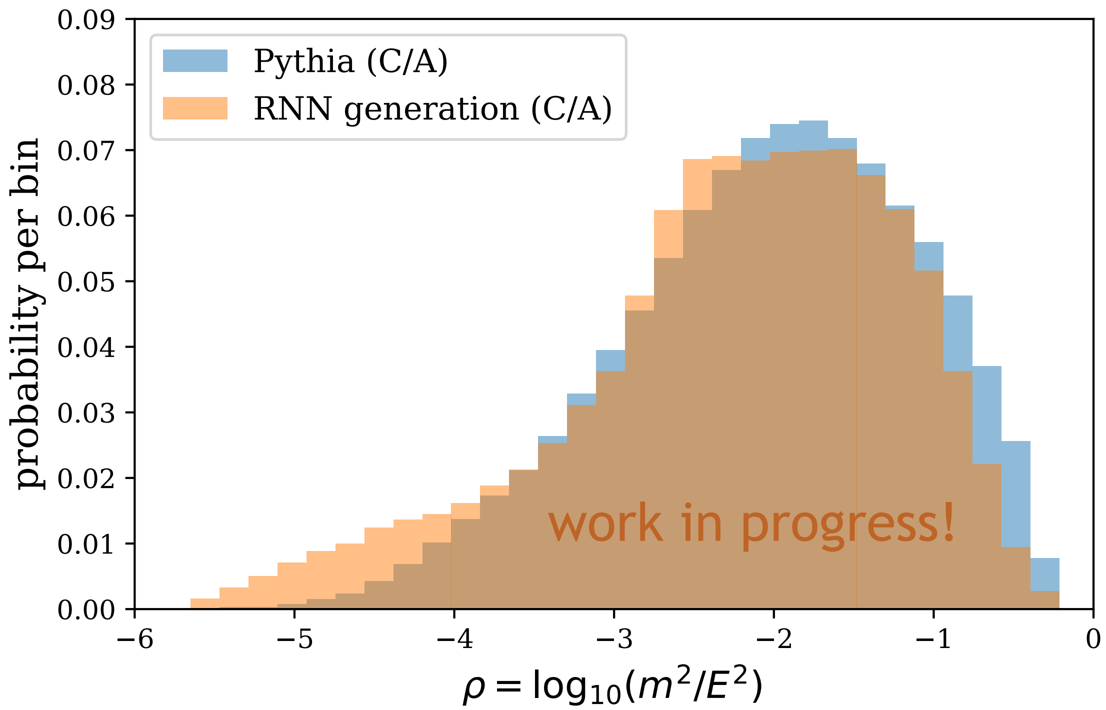
anti-kt shower



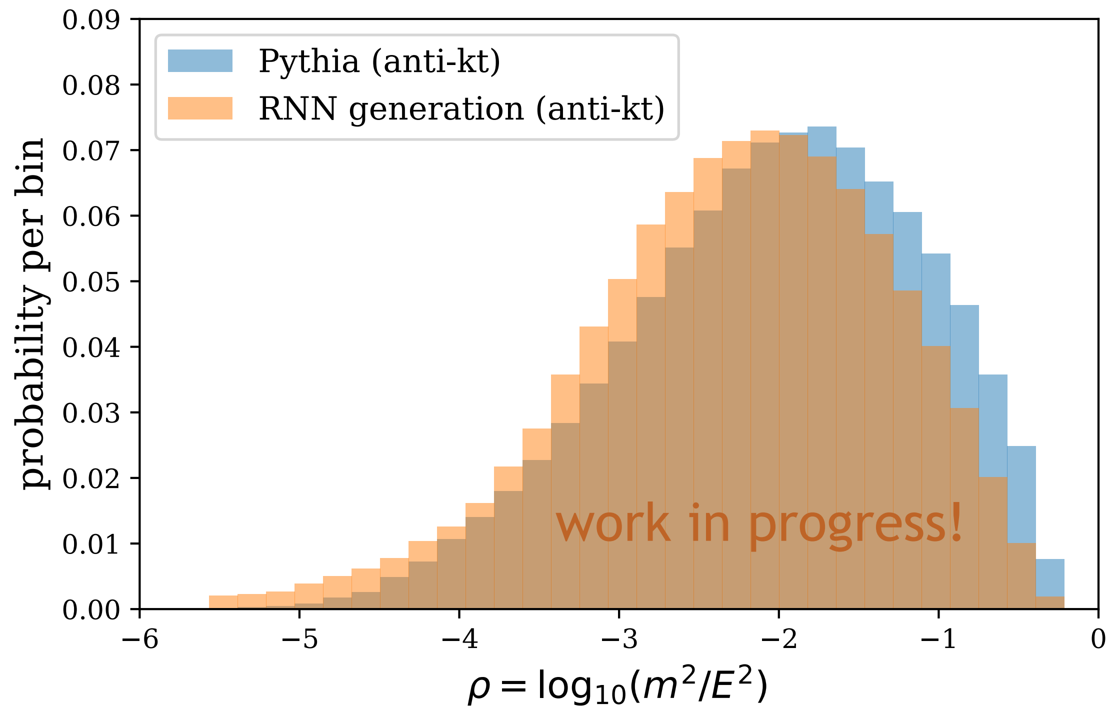
- No transparent correspondence for anti-kt trees

Clustering algorithm independence

C/A shower



anti-kt shower



- Regardless of clustering algorithm used, model and data agree on final state distributions!

Summary

- Probabilistic model with recurrent architecture customized for jet evolution
- Transparent setup lets us probe what jet substructure the network has learned, e.g. QCD splitting functions
- Can be trained directly on data, any clustering algorithm

What next?

- Clustering algorithm independent training
  - sum over clustering trees
  - discover factorization?

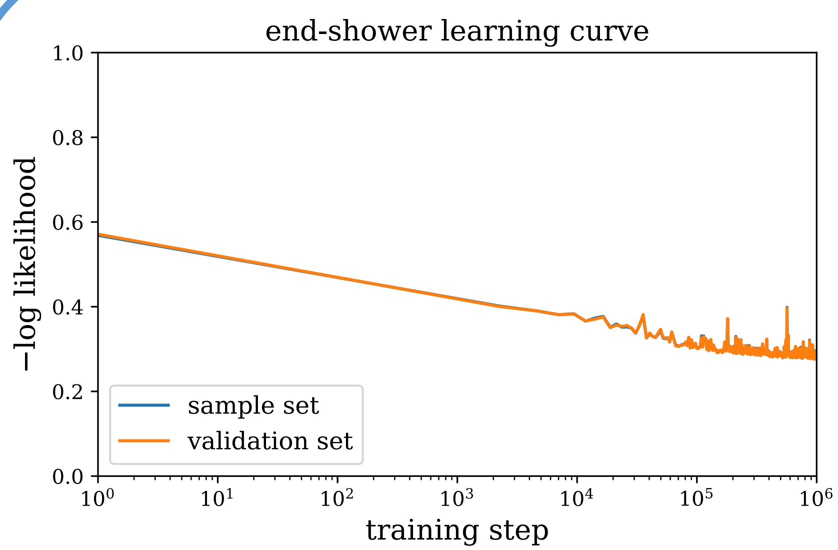
- Including quantum numbers, showering full events

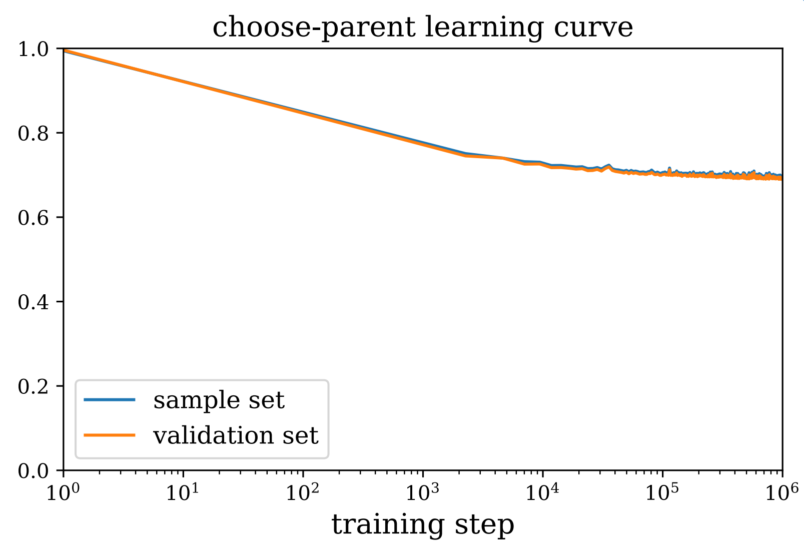
- Training on real data
  - CMS Open Data as a start
  - heavy ion collisions?

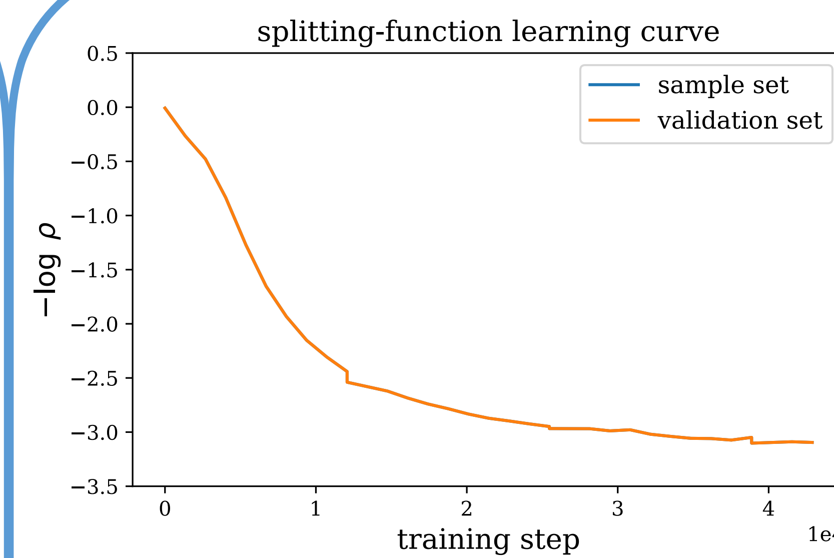
- Unbiased generator for jets

BACK UP SLIDES

Training summary







- Useful to pre-train RNN: $\text{Loss} = L_{\text{end}} + L_{\text{parent}}$

  $\Longrightarrow$ RNN learns representation of global structure

- learning rate = 0.1
  batch size = 10 jets
  10^6 gradient steps
  several hours on 16 cores

- Then co-train all components

  $\text{Loss} = L_{\text{end}} + L_{\text{parent}} + L_{\text{split}}$

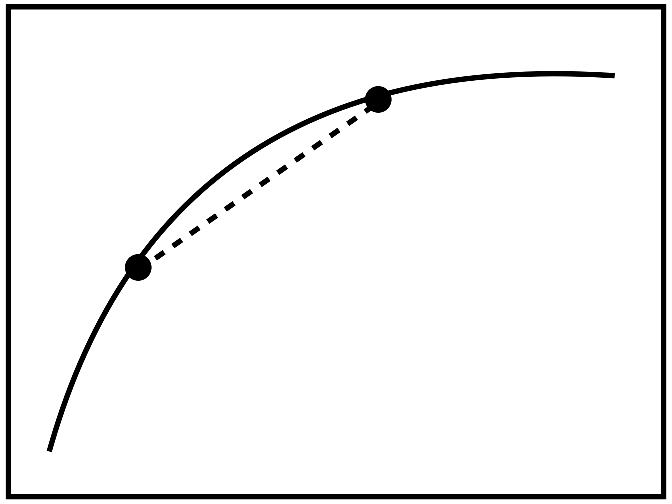
- learning rate = 0.01
  batch size = 10 jets
  10^5 gradient steps
  couple days on 16 cores

Biased estimations

- Loss $= -\log \rho(x) = -E(x) + \log Z$
- Uniform sampling to estimate $Z = \int dx\, e^{E(x)}$

$\implies$ unbiased estimator: $\langle \hat{Z} \rangle = Z$

- Biased estimator of loss: $\langle \log \hat{Z} \rangle < \log \langle \hat{Z} \rangle$



$$\Delta = \frac{\sigma(\hat{Z})}{\hat{Z}}$$

- Monitor standard error of $\hat{Z}$
- Double sample size if $\Delta > \Delta_{\max}$