Fast and Robust Neural Networks for HEP

Abhijith Gandrakota[1], Ryan Liu[2], Jennifer Ngadiuba[1], Aahlad Puli[3], Nhan Tran[1], Lily Zang[3]
et. al

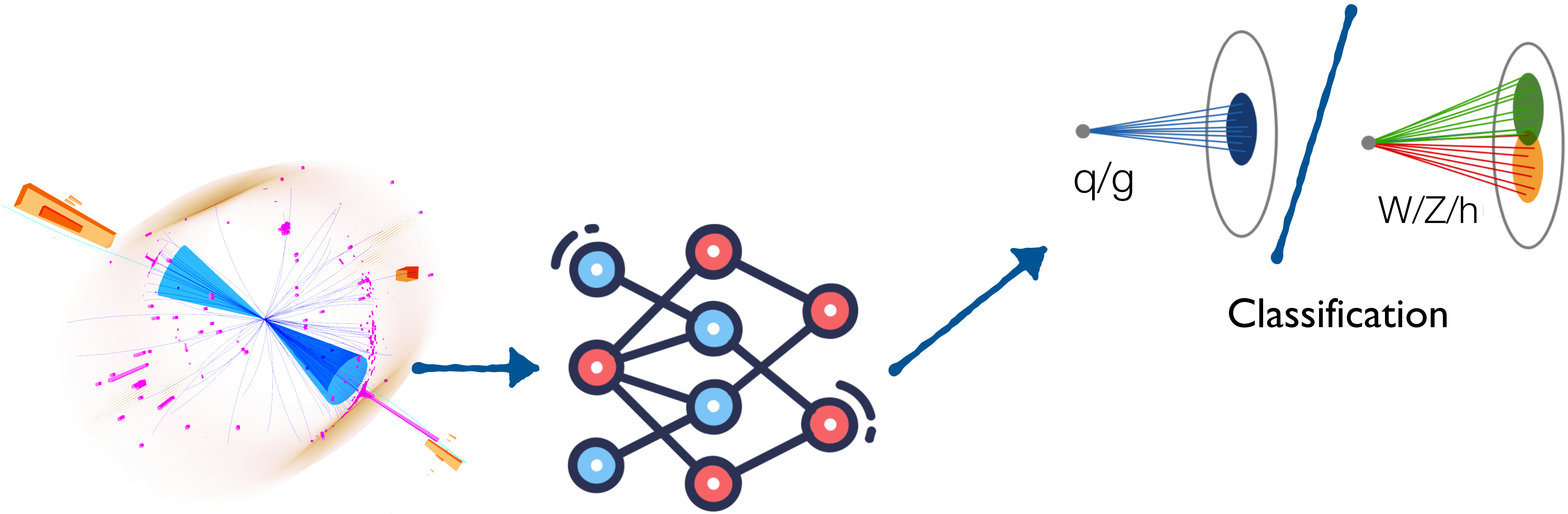
Fermilab [1] Berkeley UNIVERSITY OF CALIFORNIA [2] NYU [3]

ACAT 2024, Stony Brook

See: arxiv:2311.14160, arxiv:2311.17162, arxiv:2401.08777

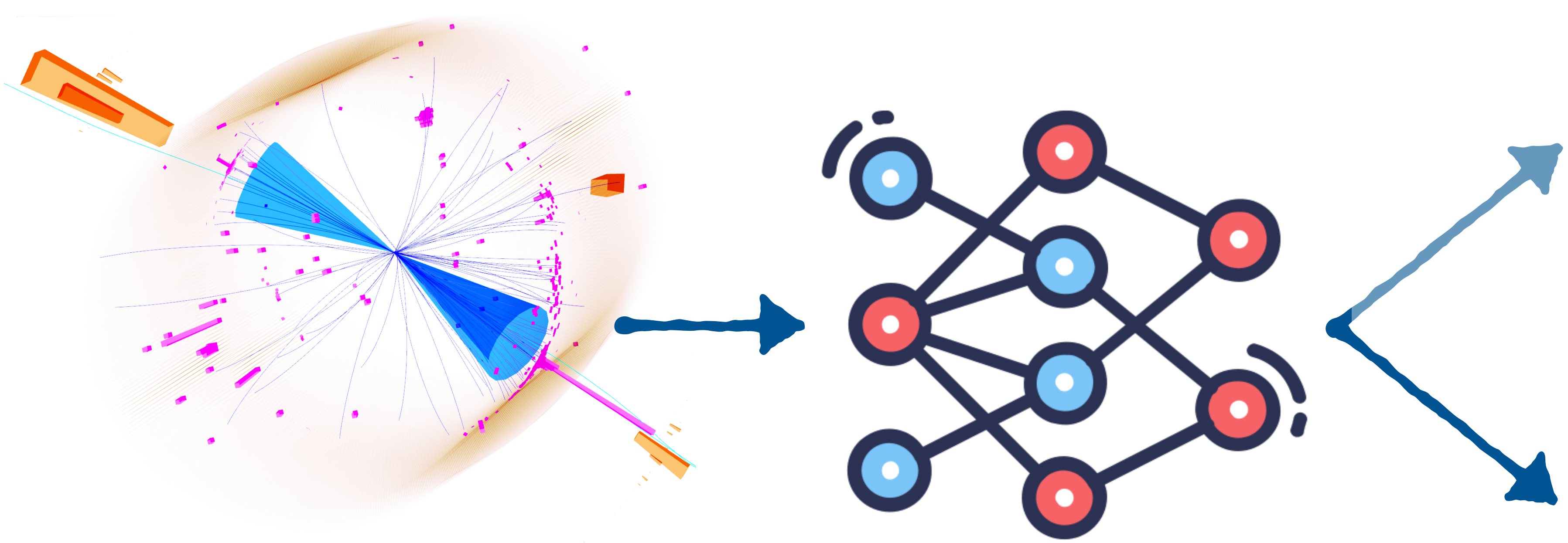
NNs in HEP experiments

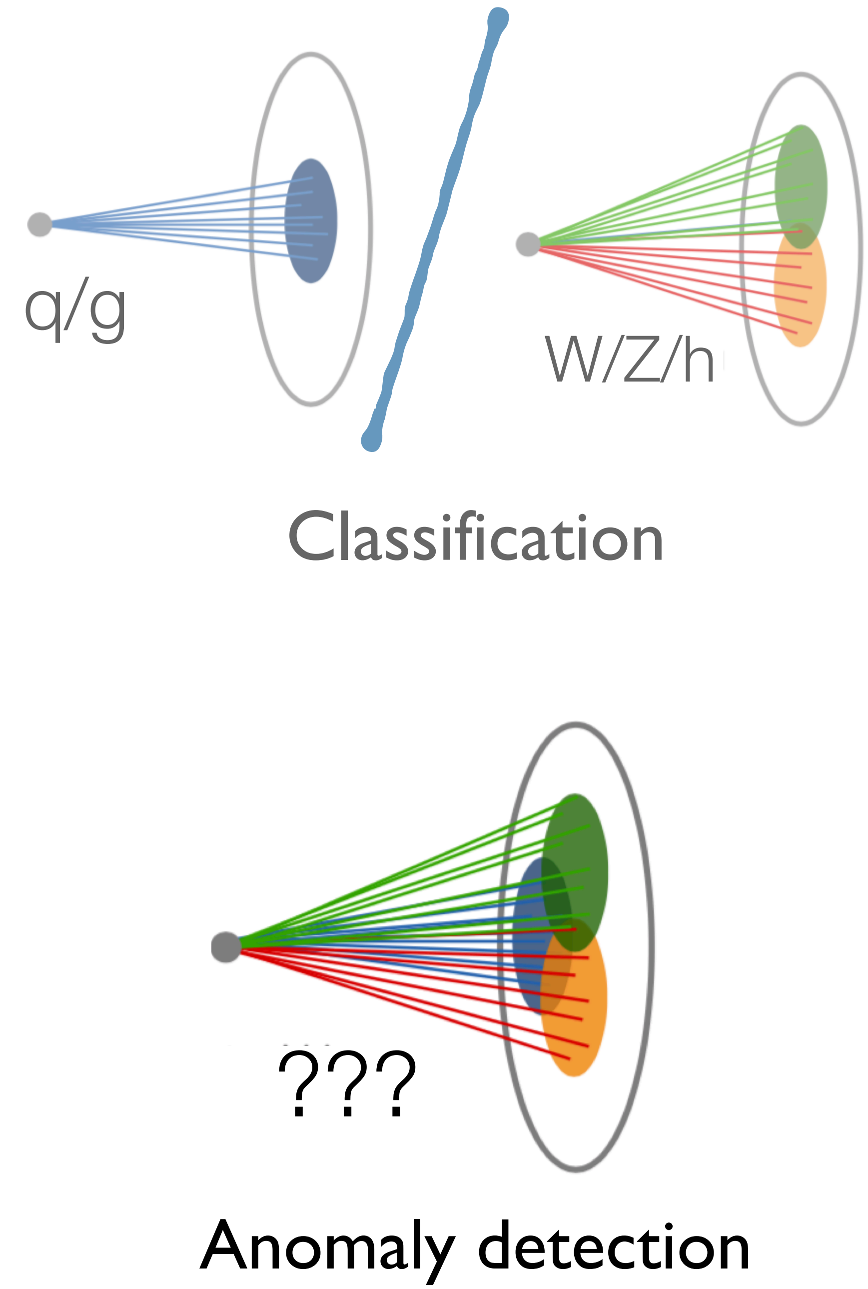
- Neural Networks are a crucial a component in our effort to find BSM physics
    - Identifying different decays from Standard Model



NNs in HEP experiments

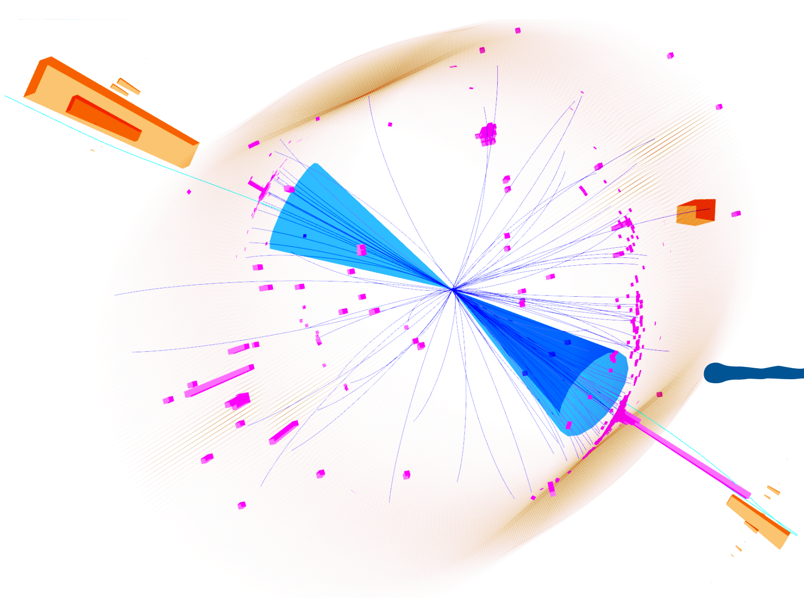
- Neural Networks are a crucial an tool to find physics beyond SM (BSM)
  - Identifying different decays from Standard Model
  - Even finding "anomalous decays" from BSM

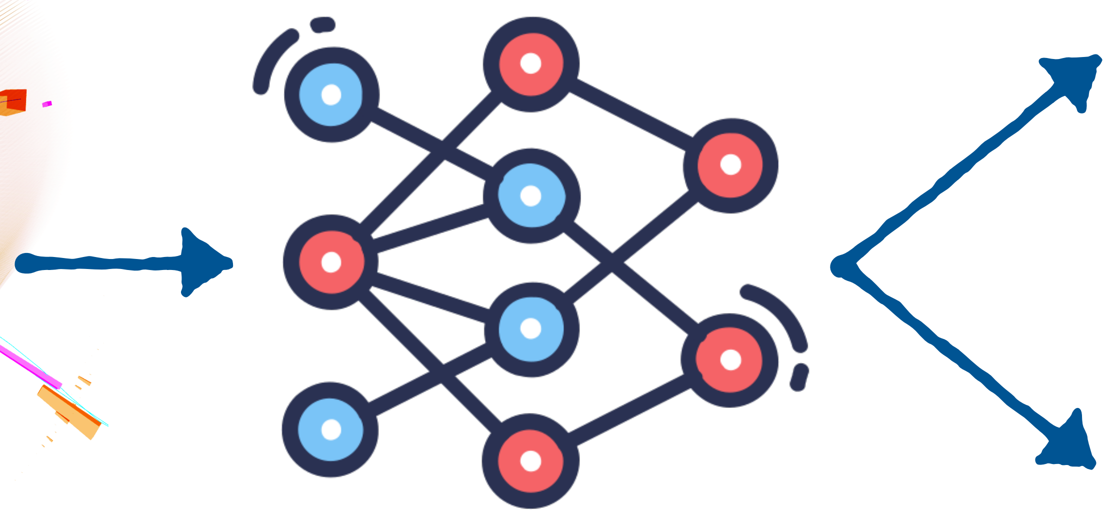


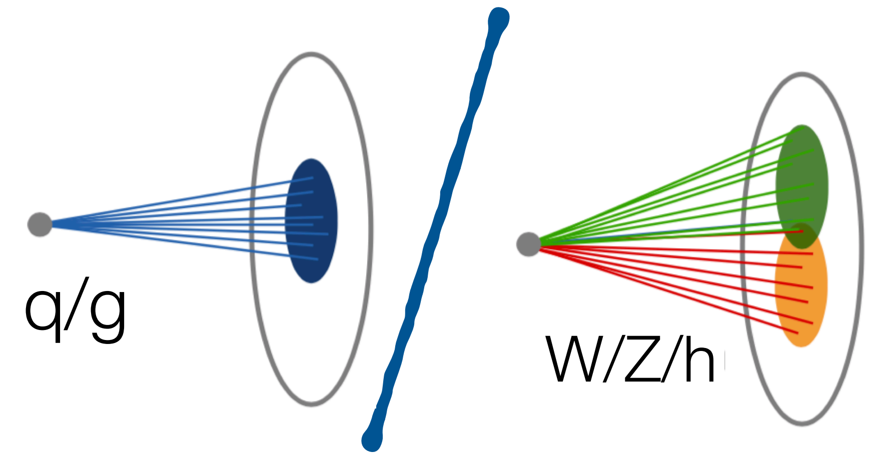


NNs in HEP experiments

- Neural Networks are a crucial an tool to find physics beyond SM (BSM)
    - Identifying different decays from Standard Model
    - Even finding "anomalous decays" from BSM

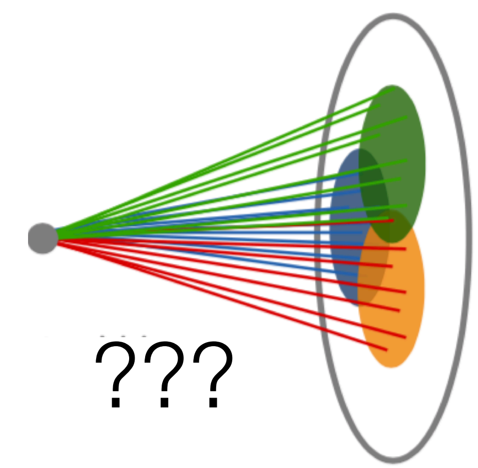






Classification

- Need to be Robust for high sensitivity and avoid false discovery
    - How to adapt it for classification and anomaly detection ?



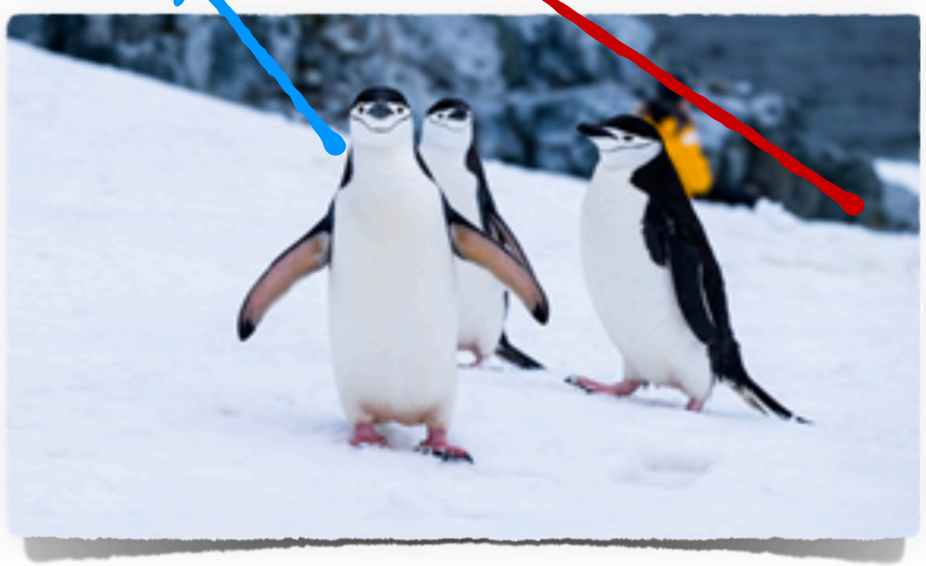
Anomaly detection

What is it ?

- NN trained to classify *cows* vs *penguins*

Prevent learning this !

Needs to learn this !



What is it ?

- Lets say we train a algorithm(NN) to identify *cows* vs *penguins*

Cows typically in grassland backdrop

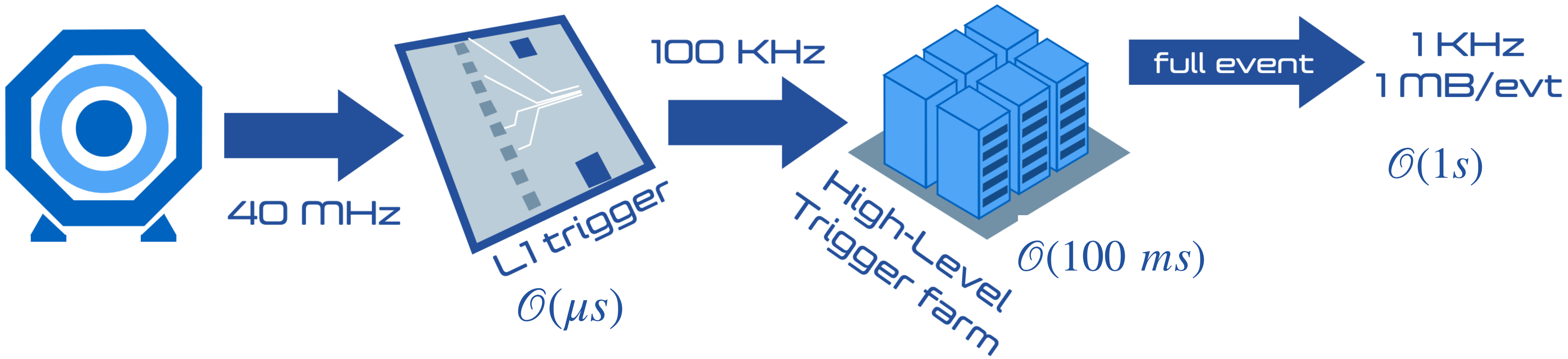
Penguins typically Photographed in snow

- What about pictures of *cows* on snow ?
  - Robust Classifier

- Can it predict if this is neither of them ?
  - Robust Anomaly Detection

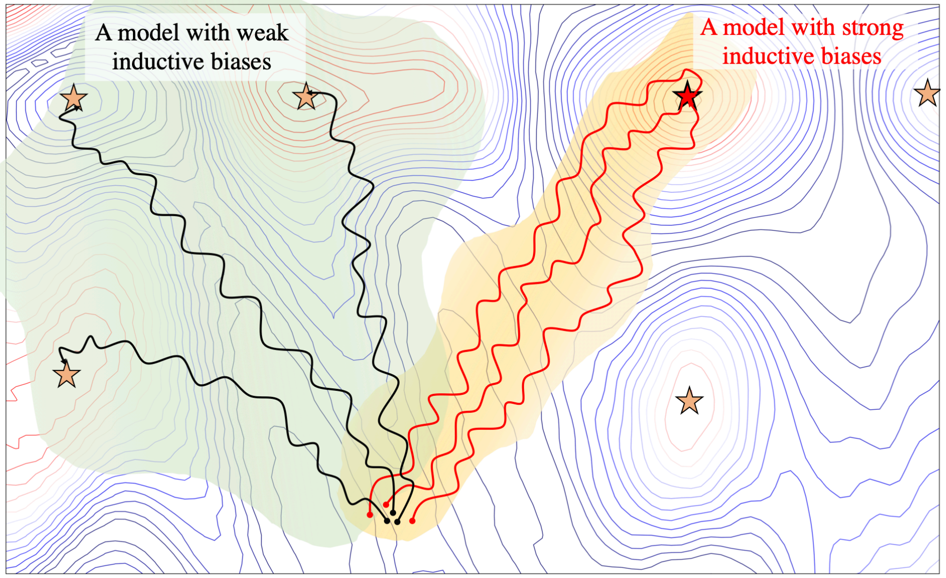
NNs in HEP experiments

- Neural Networks are a crucial an tool to find physics beyond SM (BSM)
  - Identifying different decays from Standard Model
  - Even finding "anomalous decays" from BSM
- Need to be Robust for high sensitivity and avoid false discovery
- NNs are also very important in data acquisition and processing pipelines
  - Strict inference time / latency and resource constraints
  - Need to be fast and efficient



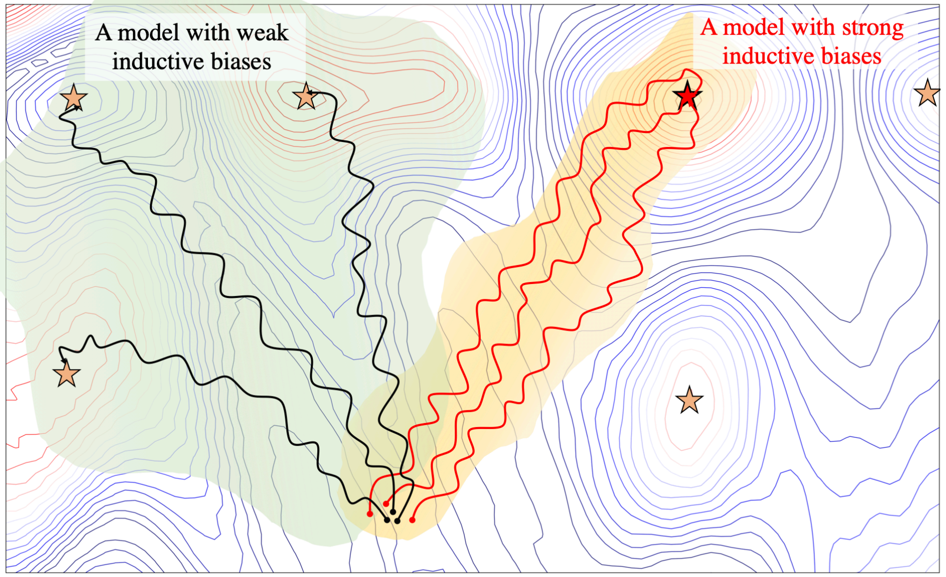
Robustness w/ Inductive bias

- We can make NNs robust with inductive bias
  - Design the model with physics knowledge
  - Explicit Inductive bias to make models robust

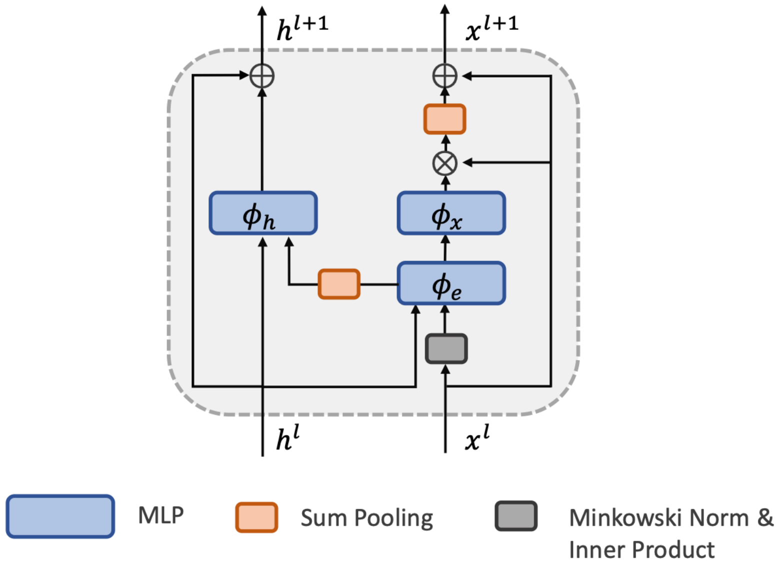


Robustness w/ Inductive bias

- We can make NNs robust with inductive bias
  - Design the model with physics knowledge
  - Explicit Inductive bias to make models robust



- Encoding Lorentz invariance into the NNs
  - Example: Lorrentz Net ([arxiv:2201.08187](https://arxiv.org/abs/2201.08187))
  - Strong invariance, but resource intensive

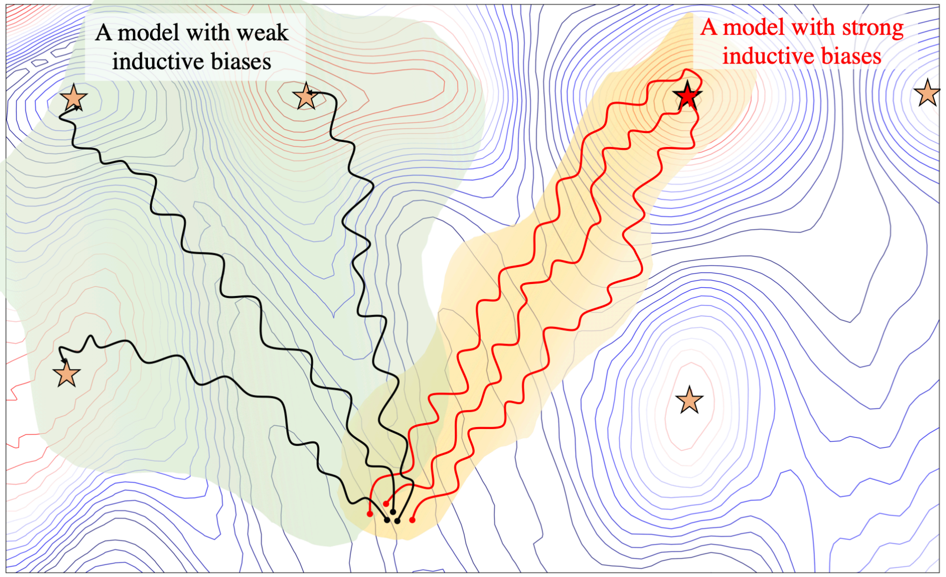


Lorentz Group Equivariant Block (LGEB)

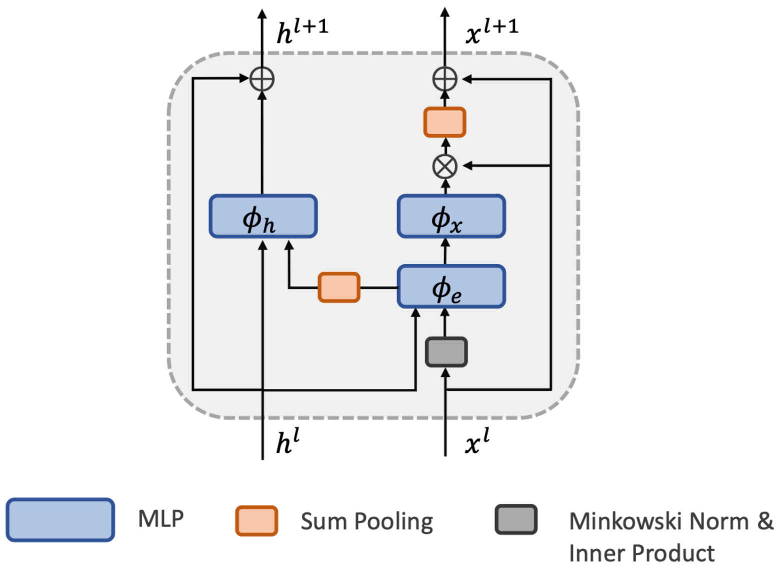
- Problem: How do we make models w/ inductive lighter and faster ?

Robustness w/ Inductive bias

- We can make NNs robust with inductive bias
  - Design the model with physics knowledge
  - Explicit Inductive bias to make models robust



- Encoding Lorentz invariance into the NNs
  - Example: Lorrentz Net (arxiv:2201.08187)
  - Strong invariance, but resource intensive

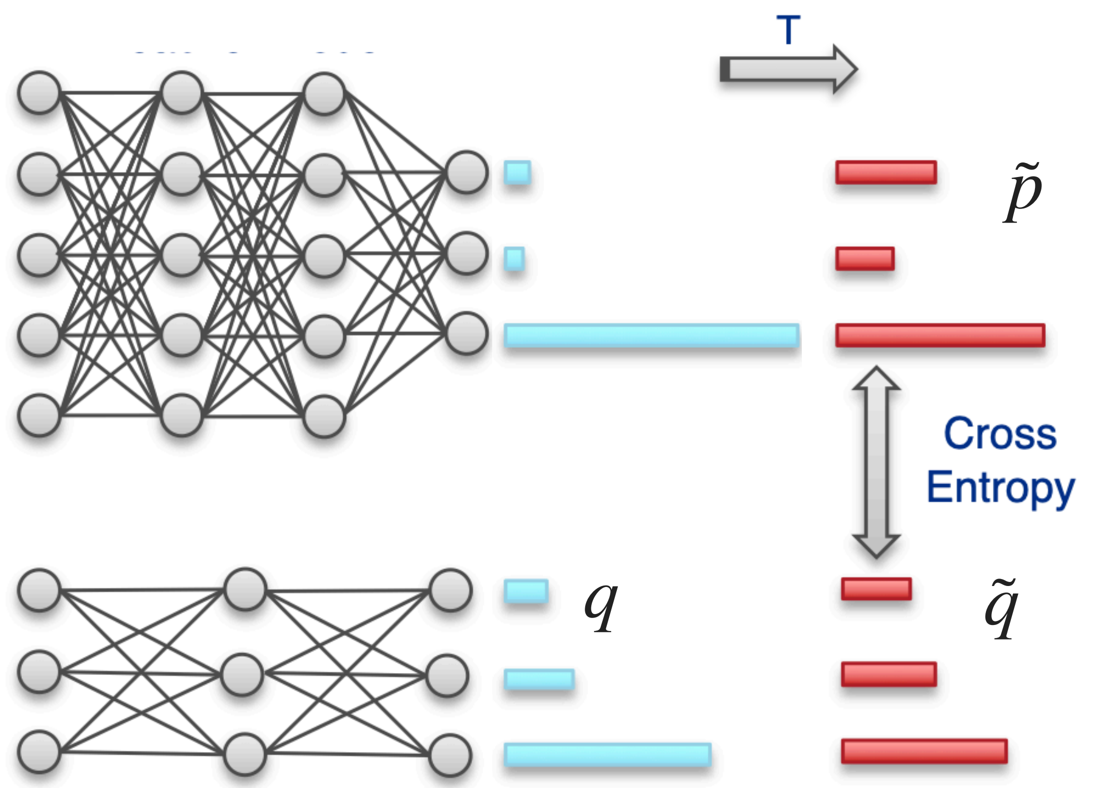


Lorentz Group Equivariant Block (LGEB)

- Problem: How do we make models w/ inductive lighter and faster ?
  - Solution: Transfer the inductive bias to a smaller model w/ Knowledge distillation

Knowledge distillation

- KD: "*Transferring knowledge from a larger complex model to smaller simple model*"
  - Proposed in arXiv:1503.02531 by Hinton et. al
- Uses the soft targets; probability distributions over classes from teacher model
  - Conveys rich information about class relationships aiding in knowledge transfer
  - *Shared insights* from teacher helps in faster convergence
- Loss :
  - $L_{KD}(q; p, y) = (1-\lambda)\mathcal{H}(y, q) + \lambda D_{KL}(\tilde{q}||\tilde{p})$
    - y : Truth labels
    - q: Student predictions
    - $\tilde{p}$: Soften predictions $\left(= \frac{e^{s(x)}/T}{\sum_{x'} e^{s(x')/T}} \right)$



Transferring Inductive Bias

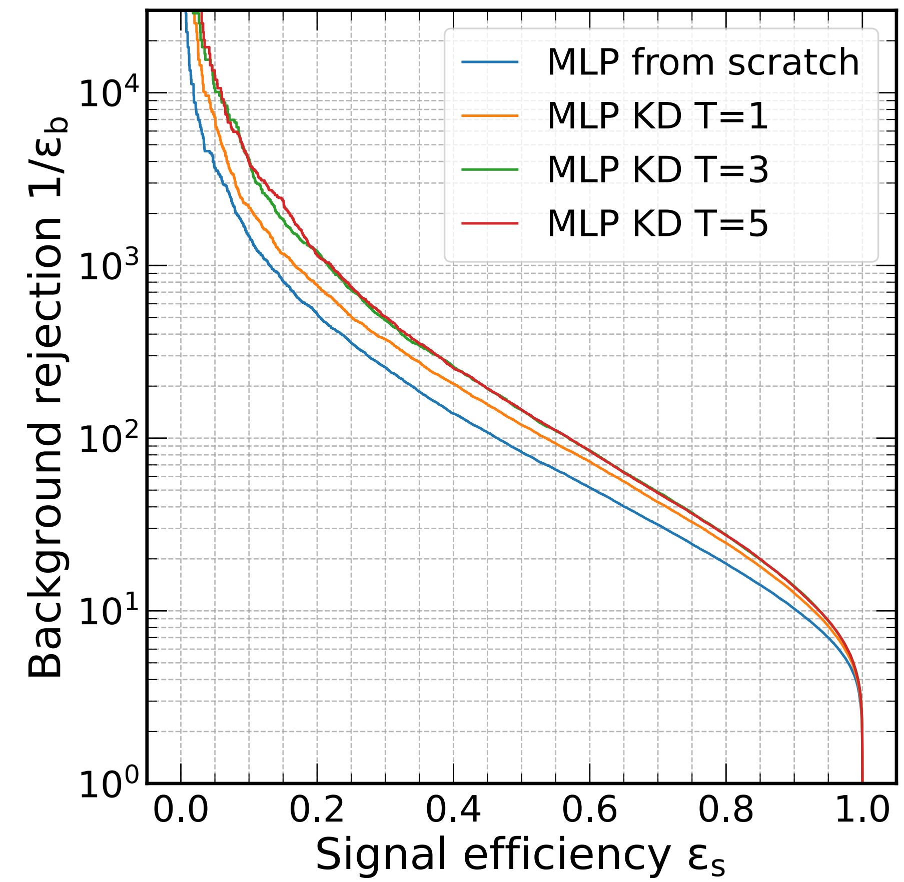
- Teacher: Lorentz Net Group equivariant graph neural network
  - Designed to w/ Lorentz Invariant message passing
- Student Networks:
  - DeepSet: 3 layer X 128 dim. wide FCN for ρ and ϕ networks
  - MLP w/ flat inputs : 3 layer X 512 hidden features
- Trained on top tagging dataset
  - Classifying QCD vs top jets [Study the effect of KD]
  - Augmented training data with boosted jets by β sampled from $[0,\beta_{max}]$ with only KD loss [Study transfer of inductive Bias]

Results

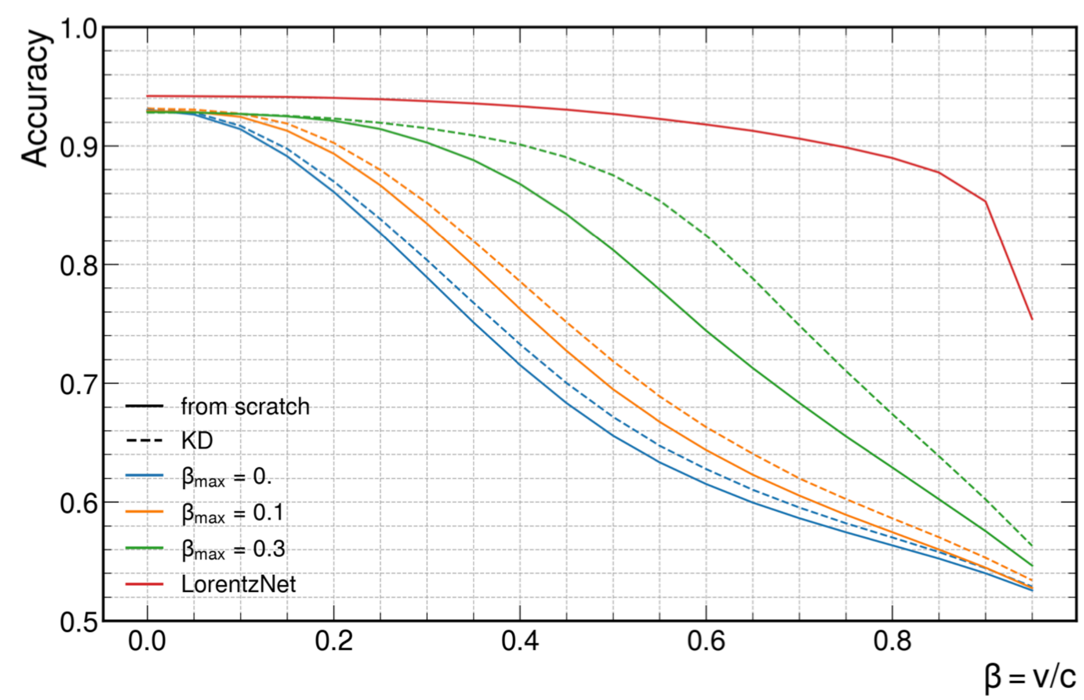
- Training with knowledge distillation leads to better accuracy and performance

- In the case of MLP, we see 1.75x improvement in BKG rejection, compared to training from scratch
  - While reducing FLOPs by 640x !!

- We observe that KD can transfer the inductive bias to the student !



	#params	FLOPs	Accuracy	AUC	$\mathrm{Rej}_{30\%}$	$\mathrm{Rej}_{50\%}$
DeepSet from scratch	68.2K	1.67M	0.930	0.9808	747	219
DeepSet KD $T=1$			**0.932**	0.9818	926	241
DeepSet KD $T=3$			**0.932**	**0.9819**	**970**	**255**
DeepSet KD $T=5$			**0.932**	**0.9819**	**970**	248
MLP from scratch	527K	529K	0.904	0.9663	256	82
MLP KD $T=1$			0.914	0.9726	375	119
MLP KD $T=3$			0.918	**0.9751**	483	144
MLP KD $T=5$			**0.919**	0.9750	**503**	**146**
LorentzNet (teacher)	224K	339M	0.942	0.9868	2195	498



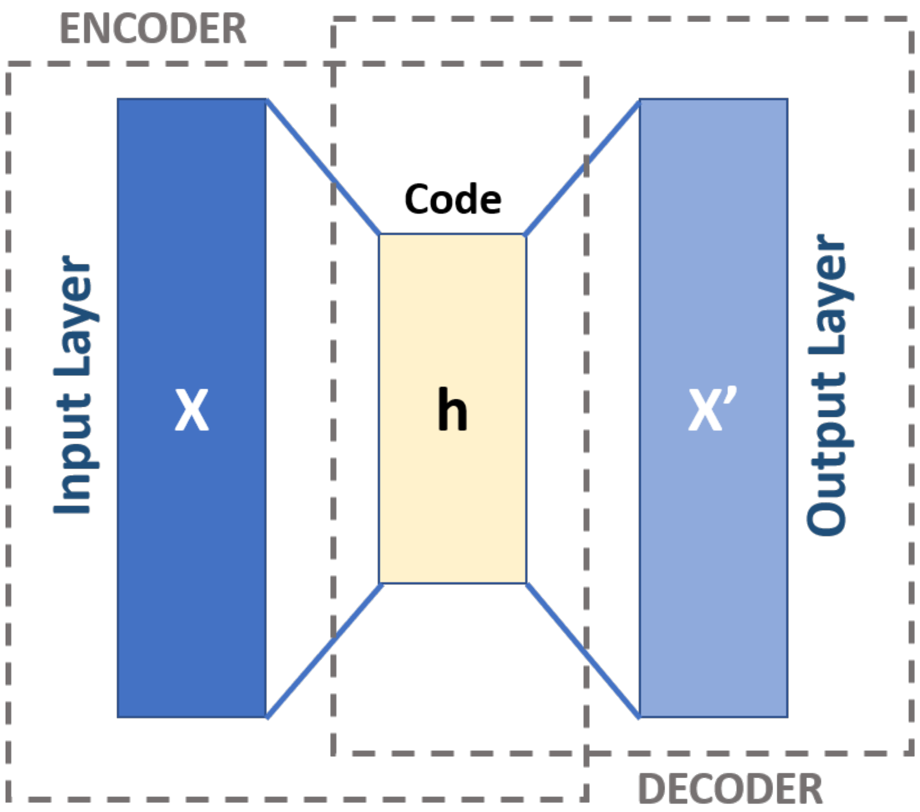
Ryan. L, AG, Jennifer. N, et al [Neurips 2023, 2311.14160]

How about anomaly detection ?

How to make it Fast and Robust ?

How about anomaly detection ?

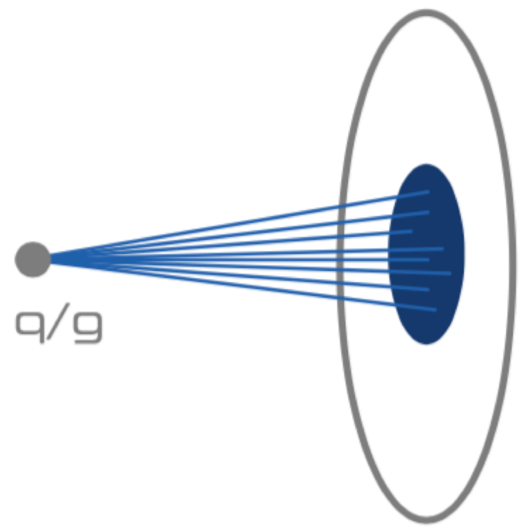
- Predominantly anomaly detection in HEP uses density estimation; e.g Autoencoders
  - Encode input into a latent space; examine reconstruction errors post decoding



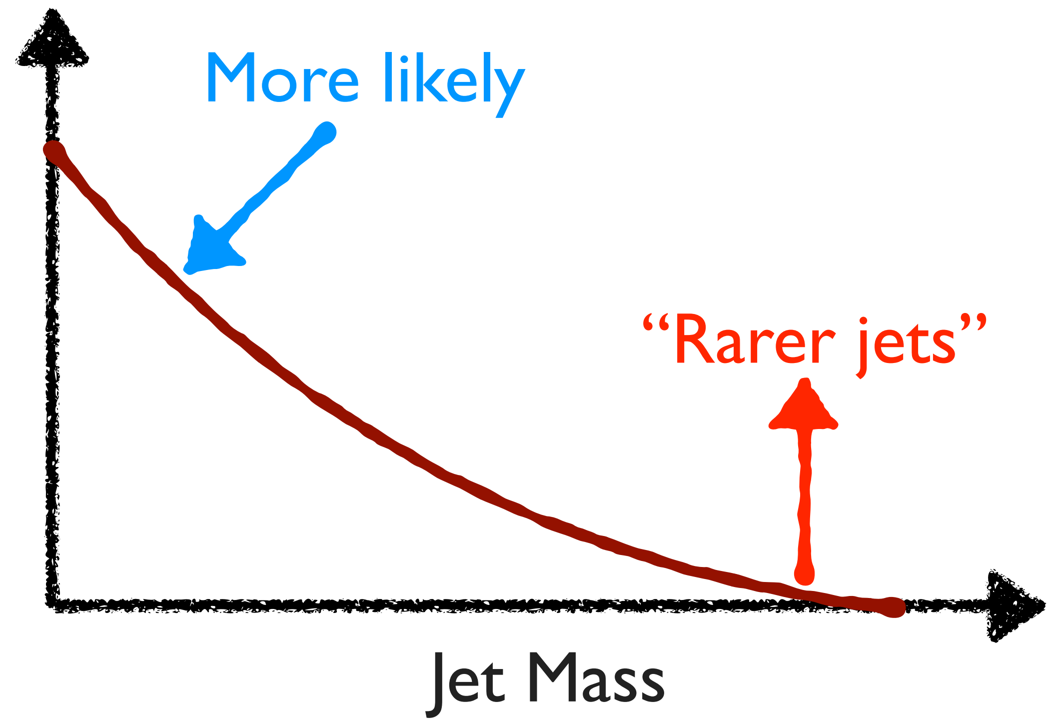
- Typically need both encoder and decoder parts of network to get anomaly metric
  - How do we make it fast and robust ?

Robust anomaly detection at LHC

- Learning from SM QCD jets to identify any BSM decays
  - More likely that these jets have lower mass



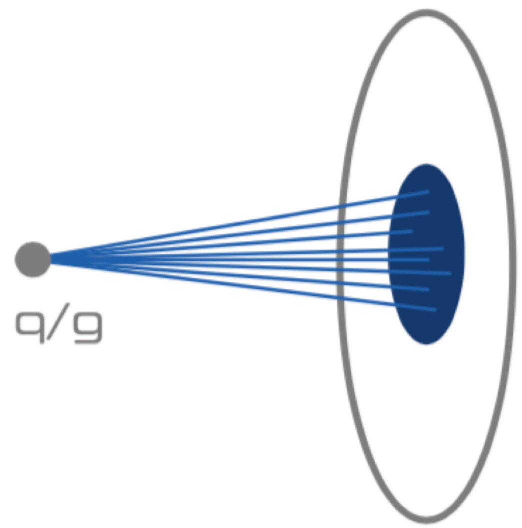
Quark or Gluon Jet



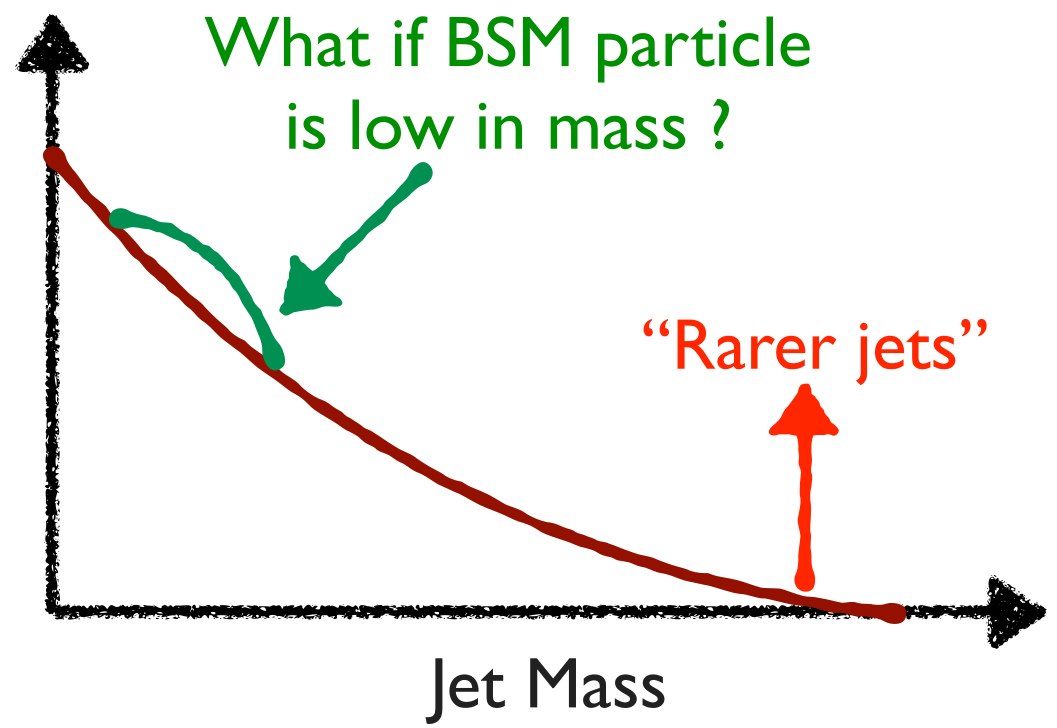
- If ML algorithm learns jet mass, it could just label ***high mass jets as anomalous***

Robust anomaly detection at LHC

- Learning from SM particle jets to identify any BSM decays
    - More likely that these jets have lower mass



Quark or Gluon Jet

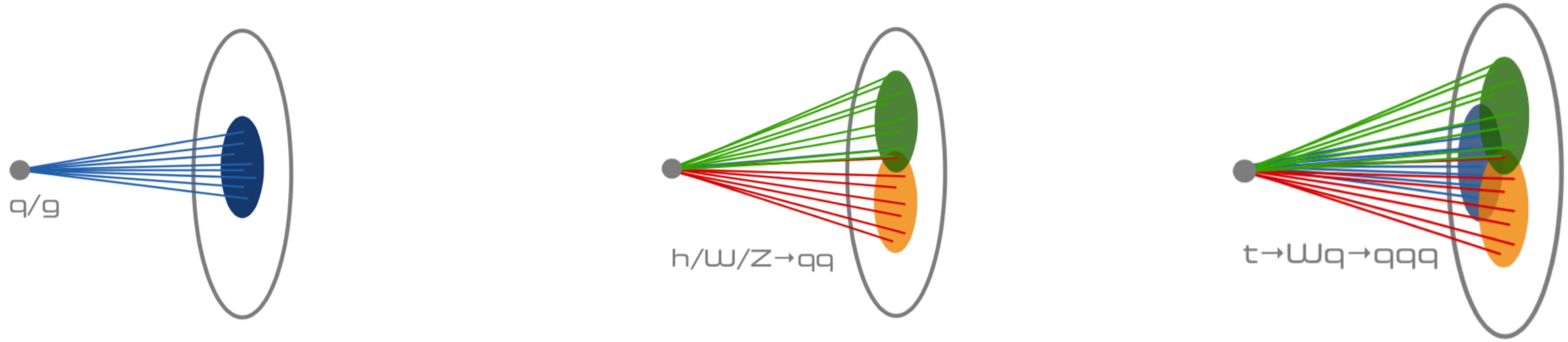


- If ML algorithm learns jet mass, it could just label ***high mass jets as anomalous***
    - We could make wrong calls, may also create signs of artificial resonances

- Need to teach network what is important and what to not to focus on

Robust representations

- Learning from QCD and W/Z jets, can detect top quark decays as outliers ?



- Idea: Use different decay examples to capture underlying physics

  - Train a classifier on MC (labeled data) $\Longrightarrow$ obtain representations
    - Avenue to learn what's important [~ minimal hand holding]
    - Build representations to have maximum information with the labels
  - Ensure representations do not vary w/ nuisances (Zhang et al. 2022, Puli et al. 2022).
    - This way, we can maximize only the relevant physics information

Nuisance Randomized Distillation

- For out dataset we have input features (X), labels for decays (Y), and Nuisance (Z)

- **Nu**isance **R**andomized **D**istillation:

  - I : Avoid learning nuisance: break the dependence b/n label and nuisance.
    - Use importance weights w to break dependence.

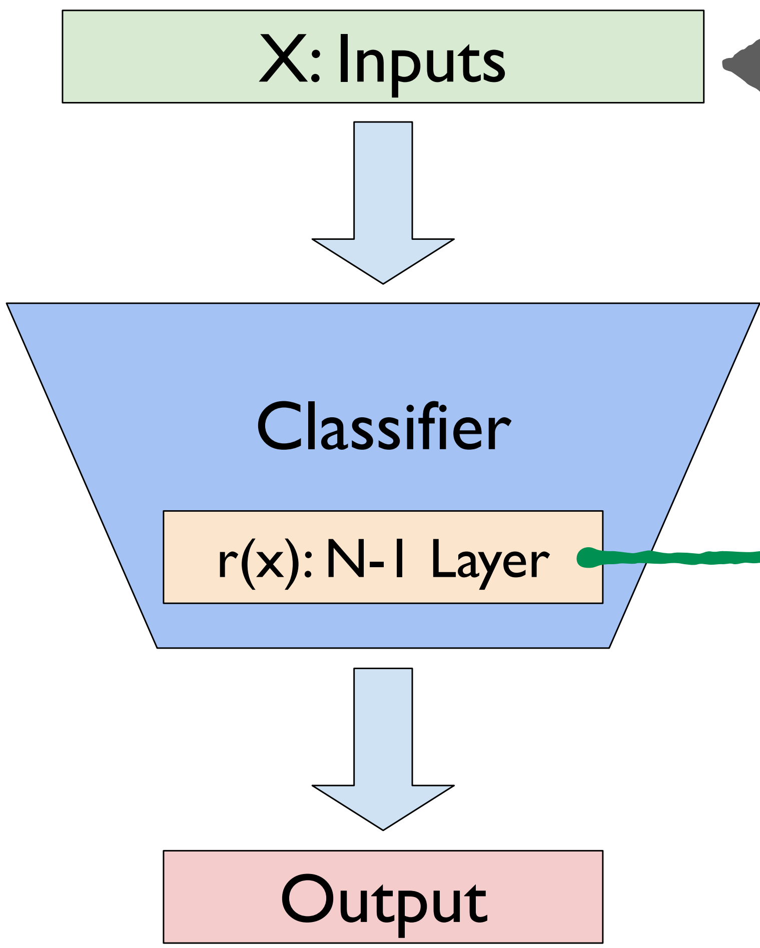
  - II : Build representations that do not vary with the nuisance
    - Intuitively, it shouldn't be possible to distinguish b/n [Joint independence]
      - (r_X, Y, Z) vs (r_X, Y, randomized nuisance($\hat{Z}$))
      - Enforce joint independence

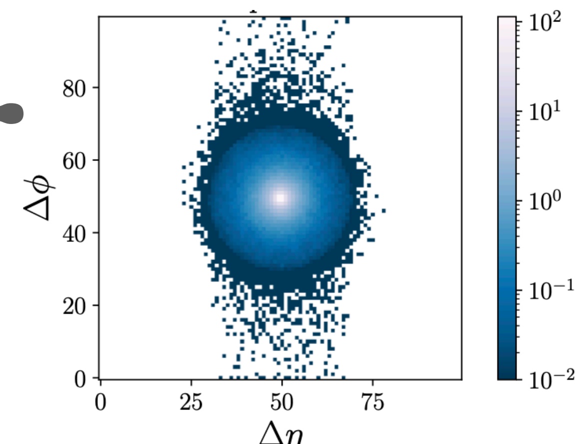
- Use the representations to detect anomalies.

[1] Puli et al. 2022

Nuisance Randomized Distillation

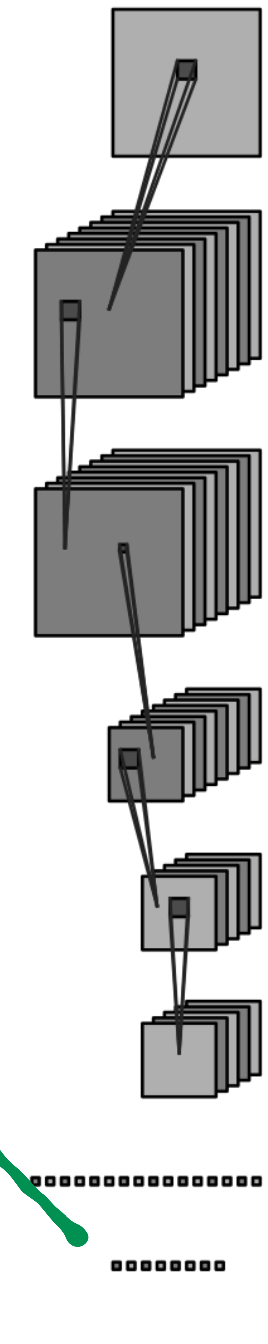
- Building out representation:
  - Start with a simple classifier b/n different particle decays
  - CNNs w/ final dense layers output to logits / softmax probabilities





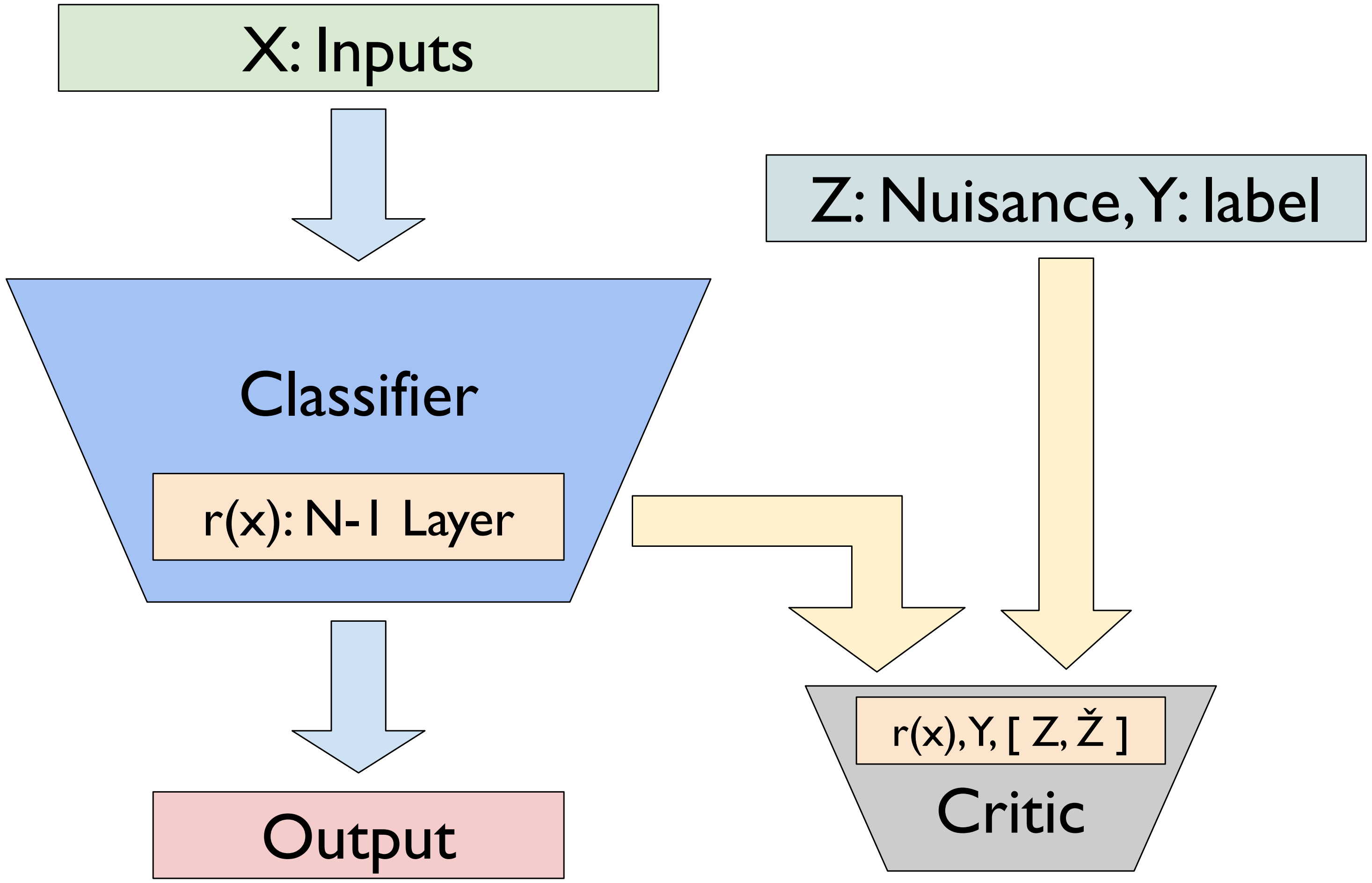
Detector images

Representation (r_X)



Nuisance Randomized Distillation

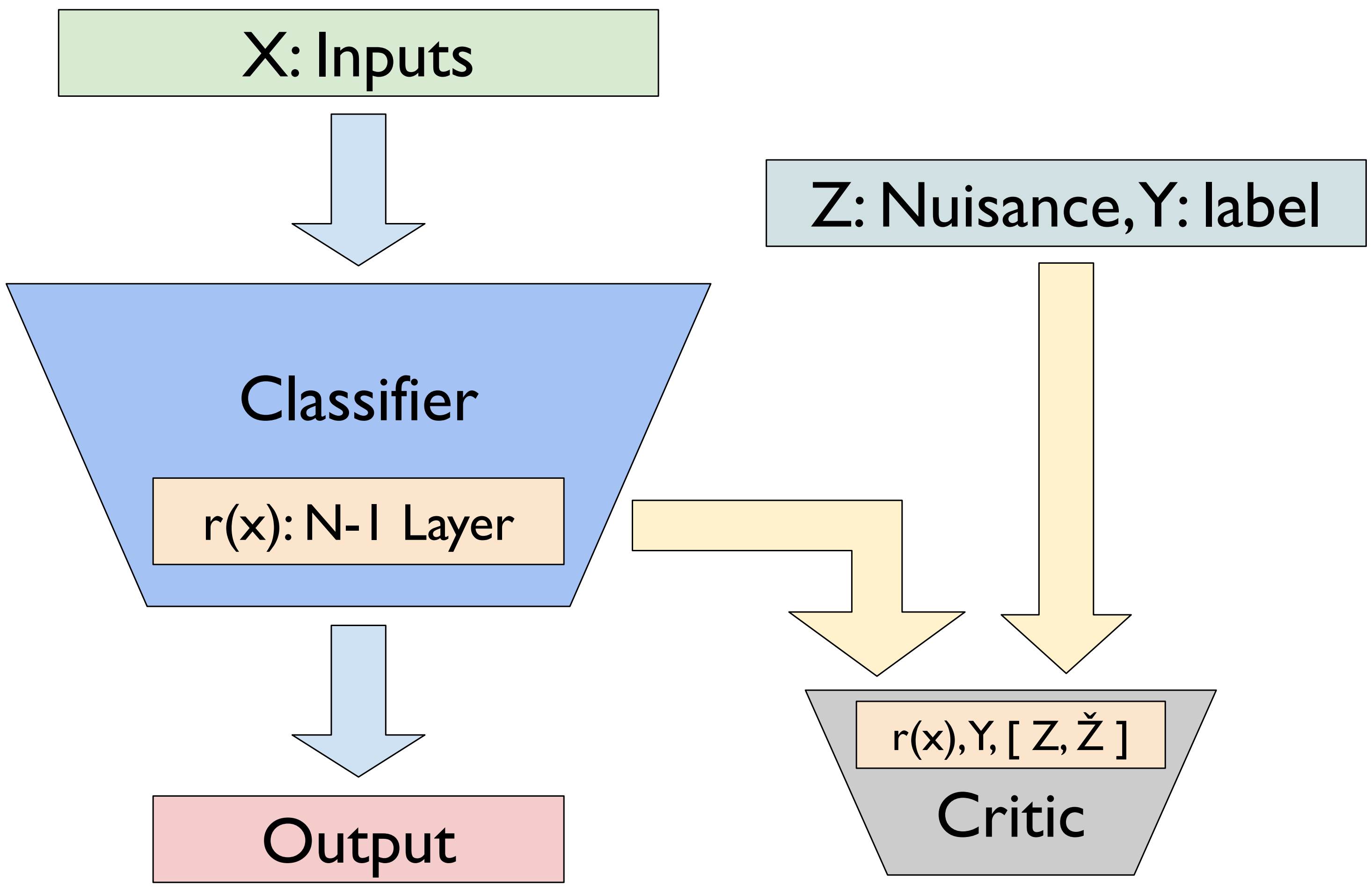
- Penalize mutual information
  - Input $(r_X, Y, [Z, \hat{Z}])$ to critic model (ϕ), a simple MLP
    - Approximates the mutual information, use this to penalize the loss



- Critic is trained to differentiate (r_X, Y, Z) vs $(r_X, Y, \hat{Z})$
  - Critic model is updated for every batch of the classifier training
  - It is proxy as the likelihood approximator

Nuisance Randomized Distillation

- Training
  - Train and update critic model for every batch of classifier training

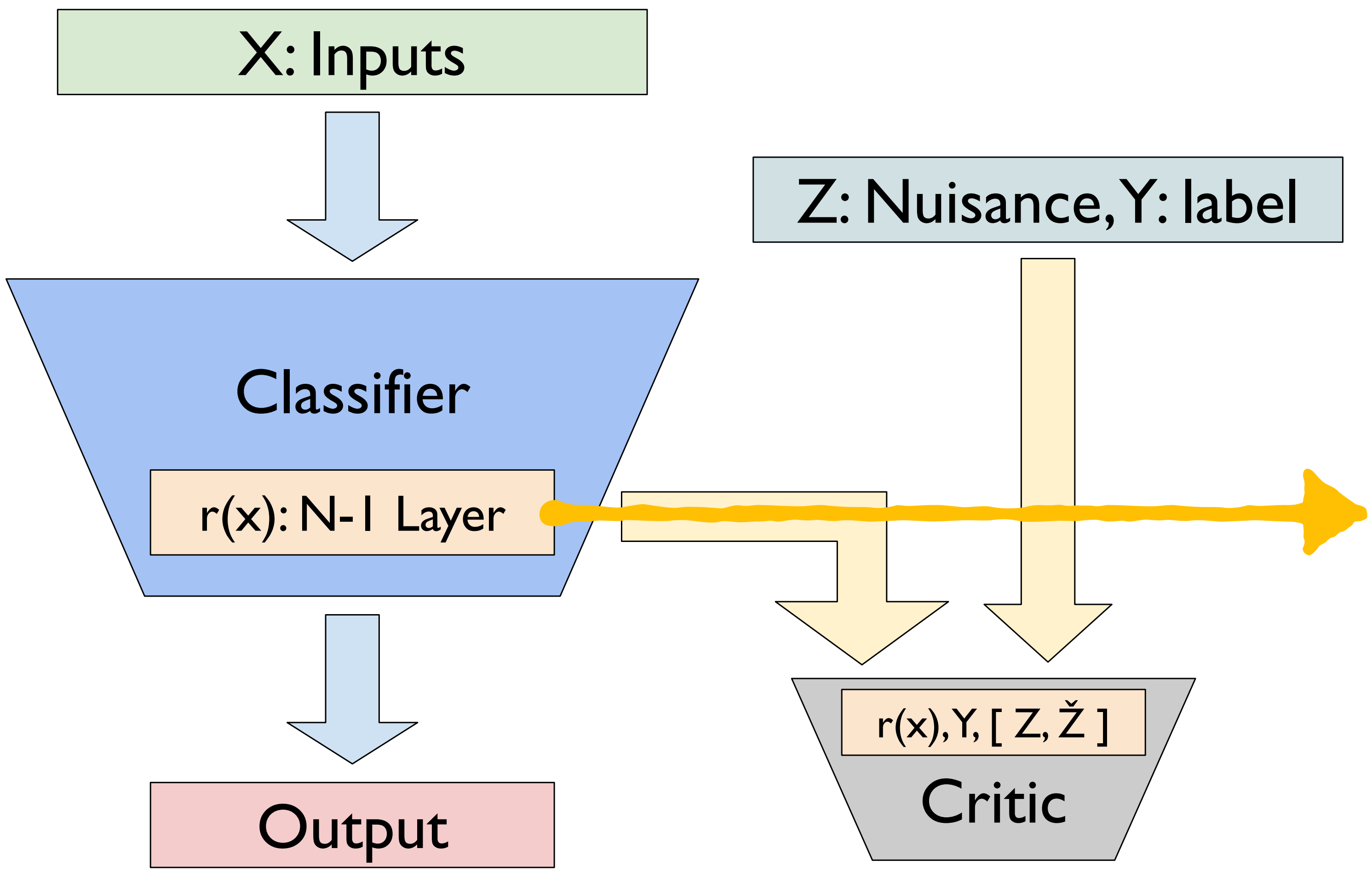


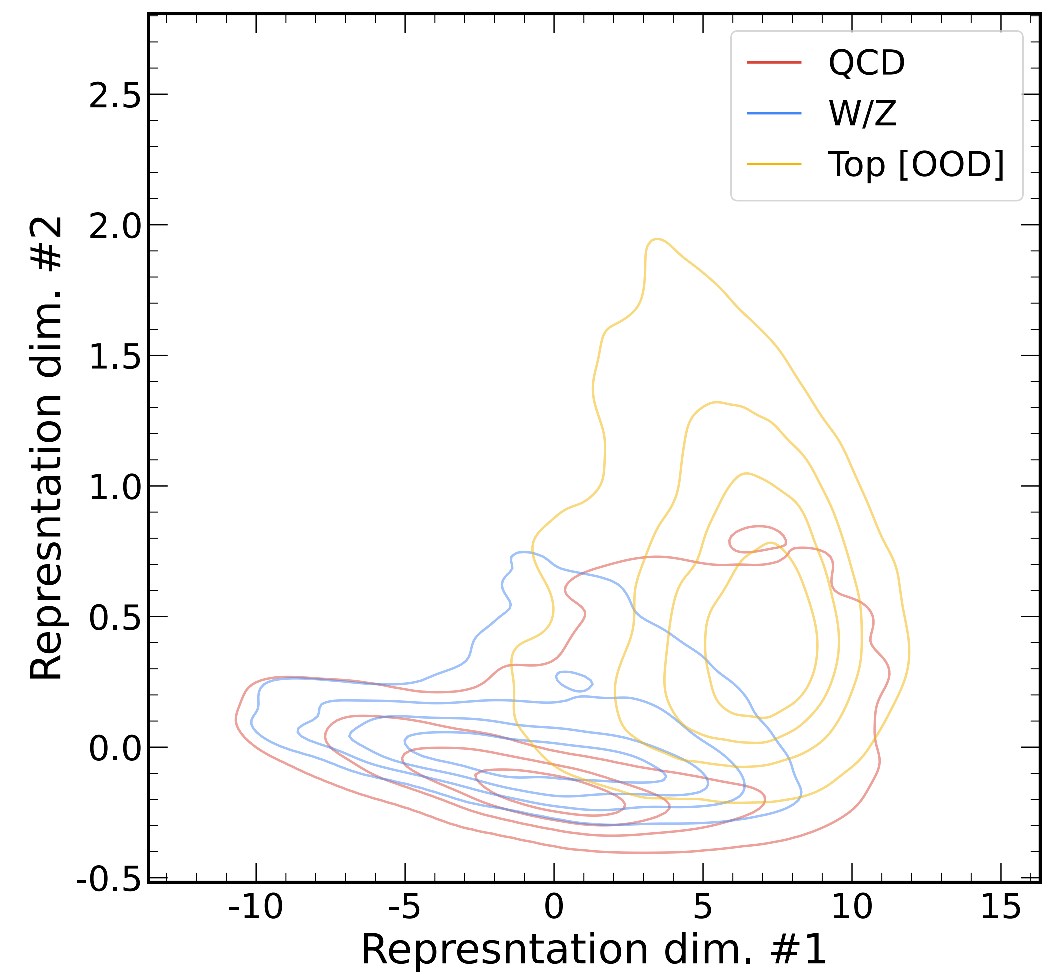
- Critic is trained to differentiate $(r_X,$ Y, Z) vs $(r_X,$ Y, $\hat{Z})$
  - Critic model is updated for every batch of the classifier training
  - It is proxy as the likelihood approximator

$$\mathscr{L} = w\left(CE(Y_{pred}, Y_{true}) - \lambda \log \frac{p_\phi(r_X, Y, [Z, \hat{Z}])}{1 - p_\phi} \right)$$

Nuisance Randomized Distillation

- Training
  - Train and update critic model for every batch of classifier training

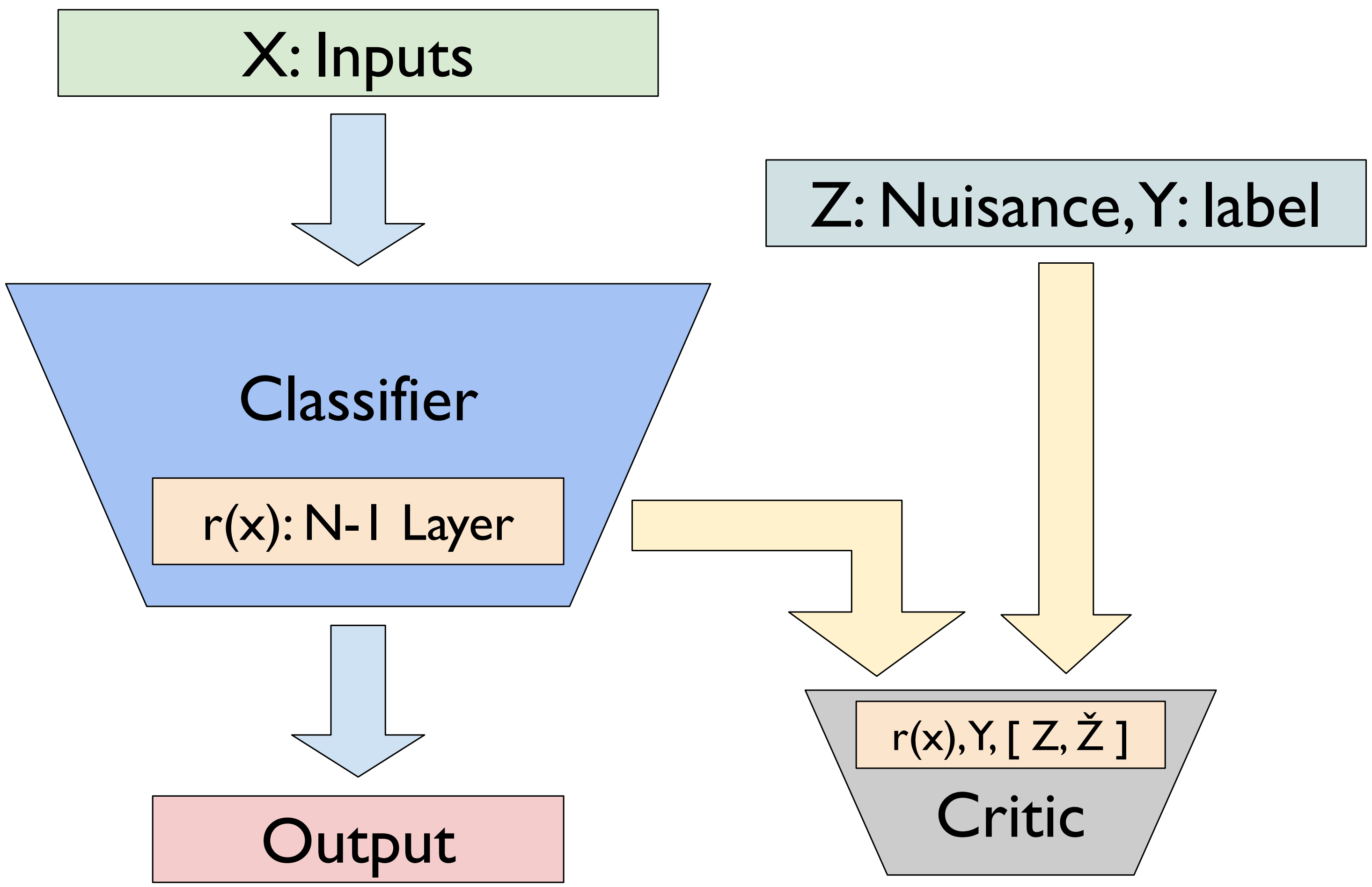




$$\mathscr{L} = w\left(CE(Y_{pred}, Y_{true}) - \lambda \log \frac{p_\phi(r_X, Y, [Z, \hat{Z}])}{1 - p_\phi} \right)$$

Nuisance Randomized Distillation

- OOD Detection:
  - Outlier Dataset: Top quarks jets
  - Use representations to build anomaly metrics



$$\mathcal{L} = w\left(CE(Y_{pred}, Y_{true}) - \lambda \log \frac{p_\phi(r_X, Y, [Z, \hat{Z}])}{1 - p_\phi} \right)$$

- Metrics:
  - Calculate the distance from samples in representation space

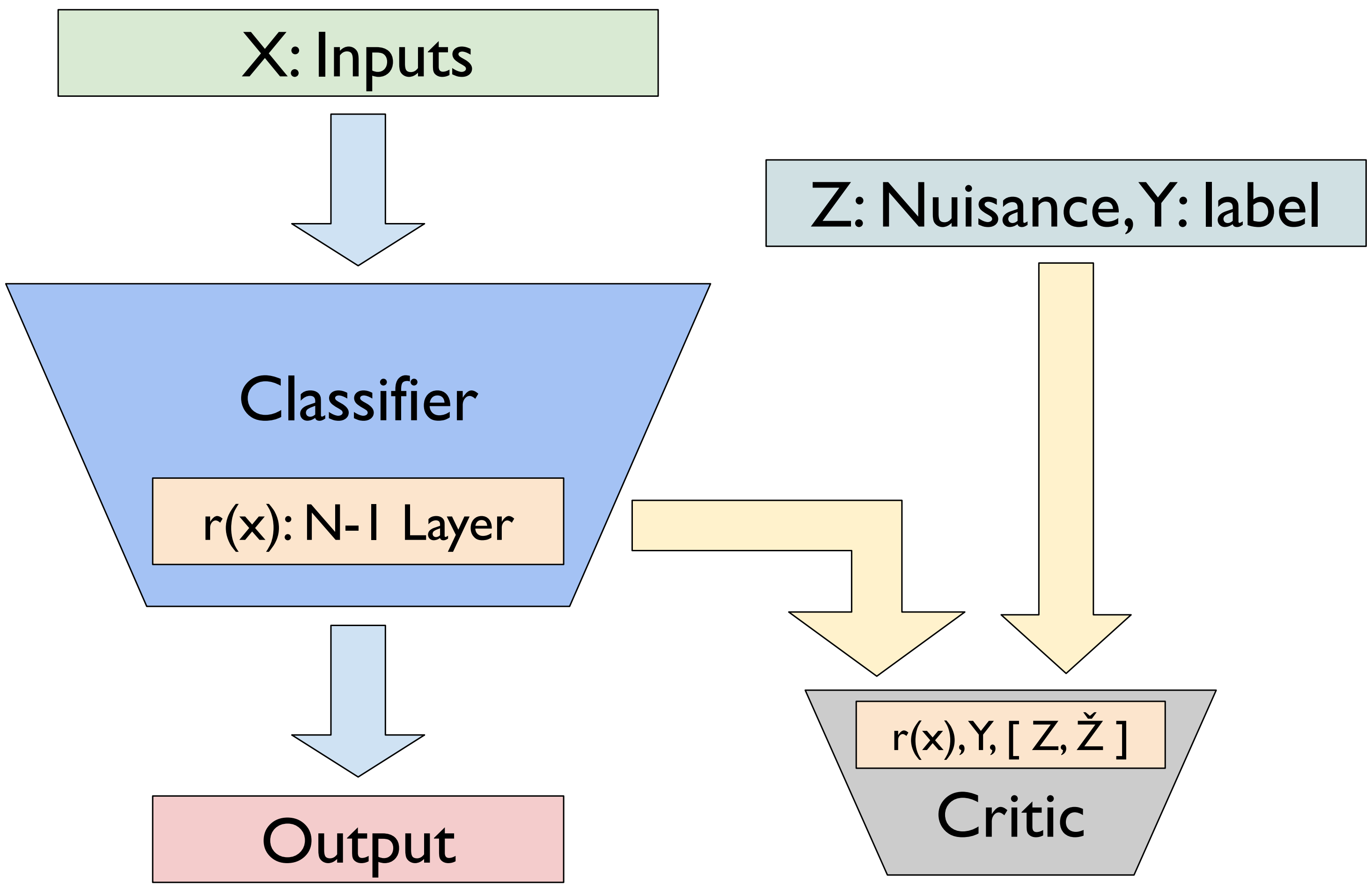
$$d_A = (r_X - \mu_A)\ \Sigma_A^{-1}\ (r_X - \mu_A)^T$$

(dist. from BKG A)

  - Obtain distance from all BKG samples
    - Here: $[d_{QCD}, d_{WZ}]$
  - Use this to find anomalies

Nuisance Randomized Distillation

- OOD Detection:
  - Outlier Dataset: Top quarks jets
  - Use representations to build anomaly metrics

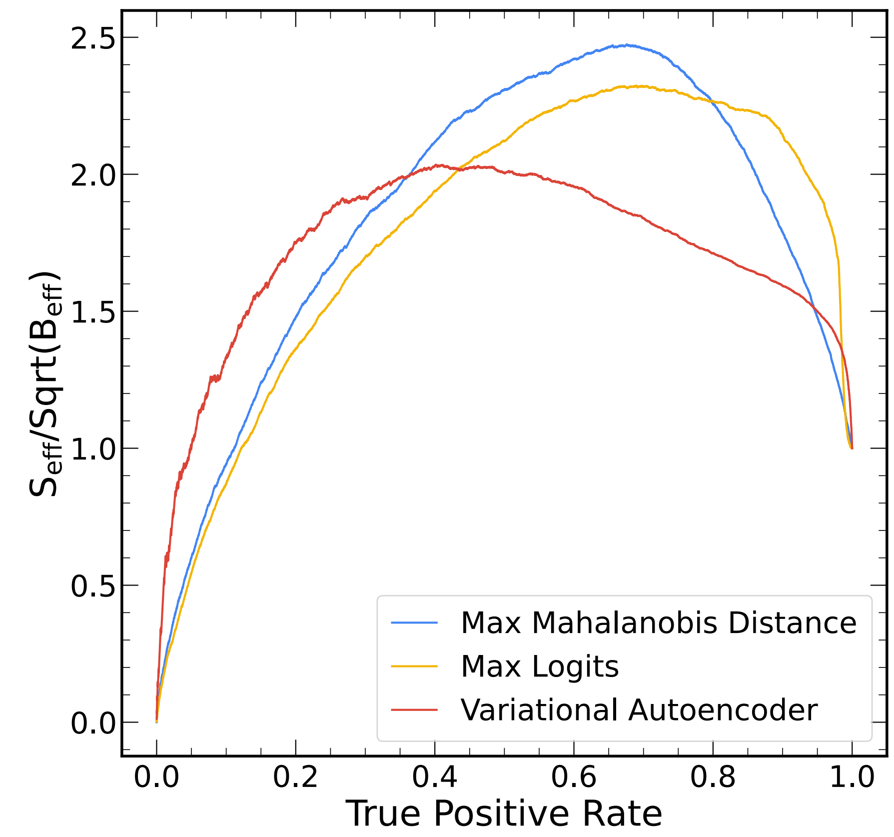


$$\mathscr{L} = w\left(CE(Y_{pred}, Y_{true}) - \lambda \log \frac{p_\phi(r_X, Y, [Z, \hat{Z}])}{1 - p_\phi} \right)$$

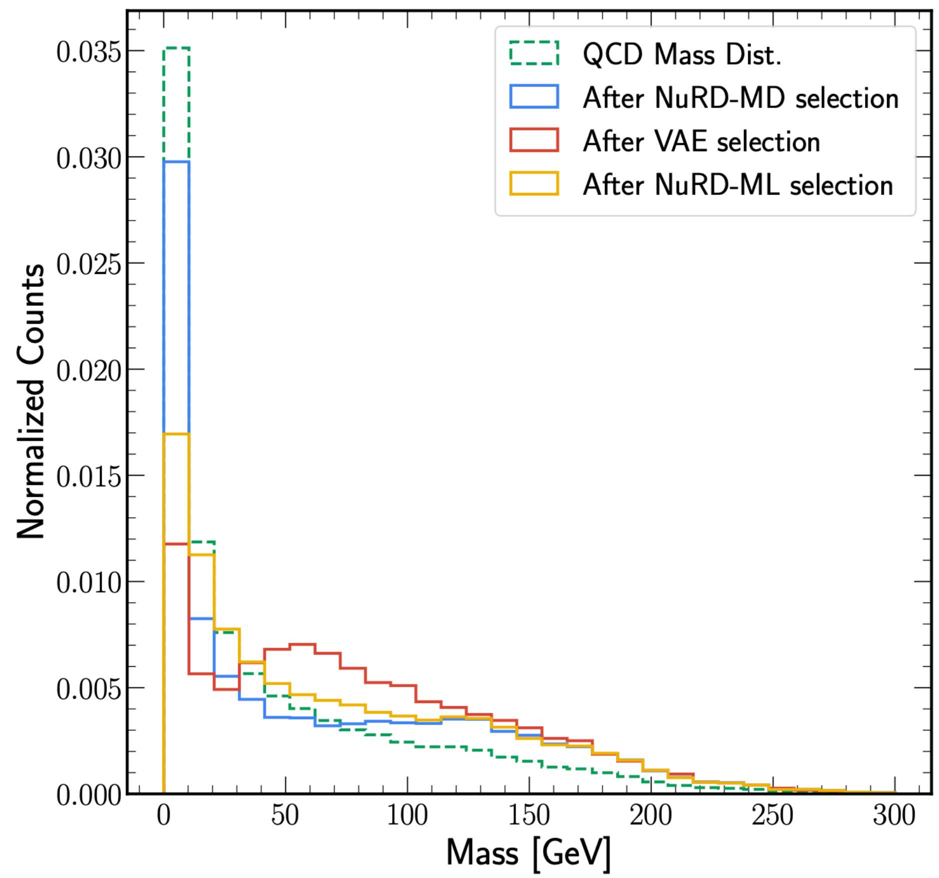
- Metrics:
  - Obtain distance d_A from all BKG samples
    - Here: $[d_{QCD}, d_{WZ}]$
- Alternative Metrics:
  - Max(Logits) also serves as a OOD Score
  - Max $_{\text{Logits}}$ (OOD) < Max $_{\text{Logits}}$ (BKG)

Results

- Obtained representations denotes the diversity of what is *typical*
  - While keeping relevant info for anomaly detection
  - Achieves this while staying decorrelated with kinematics of the jet



Method	AUC ↑	JSD ↓	Sig. Imp. ↑
VAE	0.88	0.065	2.03
NuRD-MD	0.90	**0.013**	**2.47**
NuRD-ML	**0.91**	0.027	2.32



- Can be applied on various use cases, e.g: Domain adaptation
- Easy way to teach NNs physics

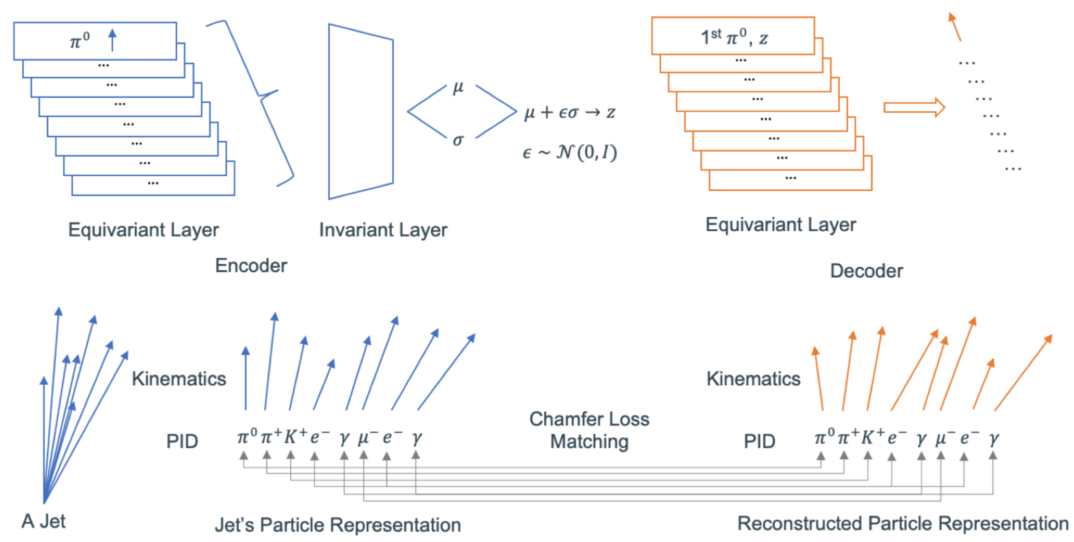
Summary

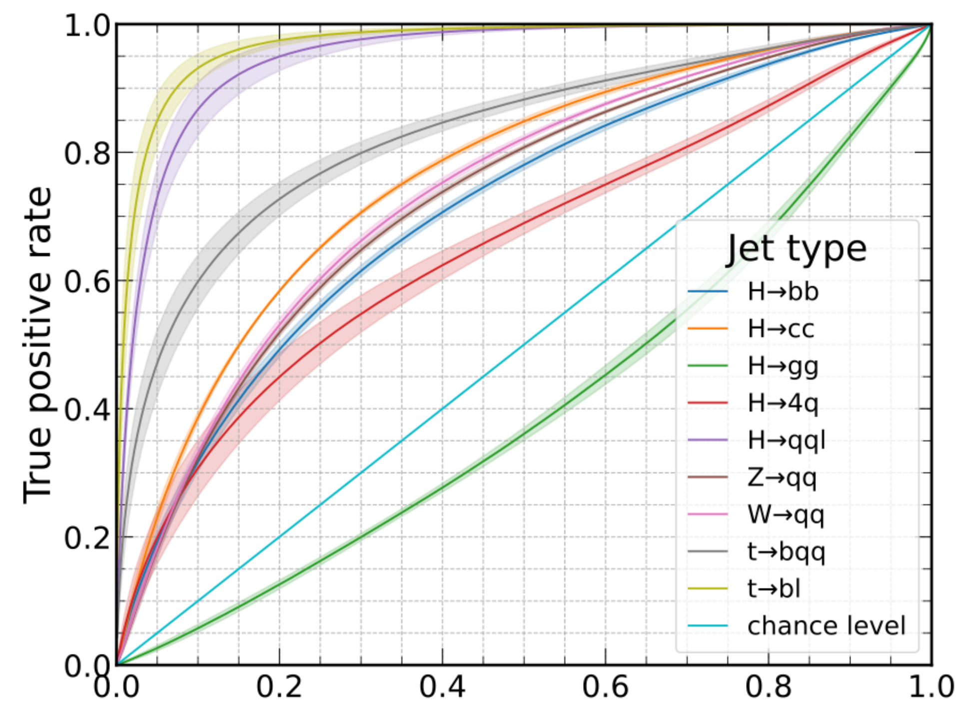
- Fast, Efficient and Robust NNs are crucial for uncovering BSM physics
  - Friendly for use in various computing architectures
- Knowledge Distillation can simplifies model architectures for various use cases
  - Enables transfer of inductive bias from a capable teacher
- New take on building a representation space to detect anomalies
  - NuRD, w/ joint independence, maximize performance while decorrelating nuisances
  - Results in smaller and robust models
- Check out for further details: arxiv:2311.14160, arxiv:2311.17162, arxiv:2401.08777

Thank you !

Efficient DeepSet auto encoder

- Permutation invariant architecture
  - DeepSet / Transformer encoder
  - Chamfer loss as reconstruction objective
  - KLD and / or Reco loss as AD score
- CLIP-VAE:
  - Avoid over-regularization for the poorly reconstructed samples
    - Prevent back-propagation of KLD term





	Model Profile		Signal efficiency (%) at Rej = 100								
	#params	FLOPs	$H \to 4q$	$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to qql$	$W \to qq$	$Z \to qq$	$t \to bl$	$t \to bqq$
DeepSet w/ PID	**103K**	**6.95M**	5.8 ± 2.1	$\mathbf{5.1 \pm 1.2}$	5.2 ± 1.1	0.4 ± 0.1	35 ± 3	3.5 ± 0.6	3.3 ± 0.6	53 ± 8	$\mathbf{22 \pm 5}$
DeepSet w/o PID			1.0 ± 0.2	2.2 ± 0.2	$\mathbf{6.3 \pm 0.5}$	0.2 ± 0.1	19 ± 1	$\mathbf{6.0 \pm 0.6}$	$\mathbf{5.2 \pm 0.5}$	49 ± 2	4 ± 1
Transformer w/ PID	952K	78.9M	$\mathbf{6.5 \pm 0.8}$	4.0 ± 0.9	4.9 ± 0.7	$\mathbf{0.5 \pm 0.1}$	$\mathbf{43 \pm 4}$	3.8 ± 0.3	3.3 ± 0.3	$\mathbf{58 \pm 5}$	19 ± 1
Transformer w/o PID			3.1 ± 0.8	2.2 ± 0.3	5.7 ± 0.6	0.3 ± 0.1	23 ± 3	5.6 ± 0.9	5.0 ± 0.6	41 ± 3	11 ± 1
N-subjettiness	N/A	N/A	0.6	1.9	5.0	0.2	19	4.1	3.5	31	8.8

Ryan. L, AG, Jennifer. N, et al [Neurips 2023, 2311.14160]