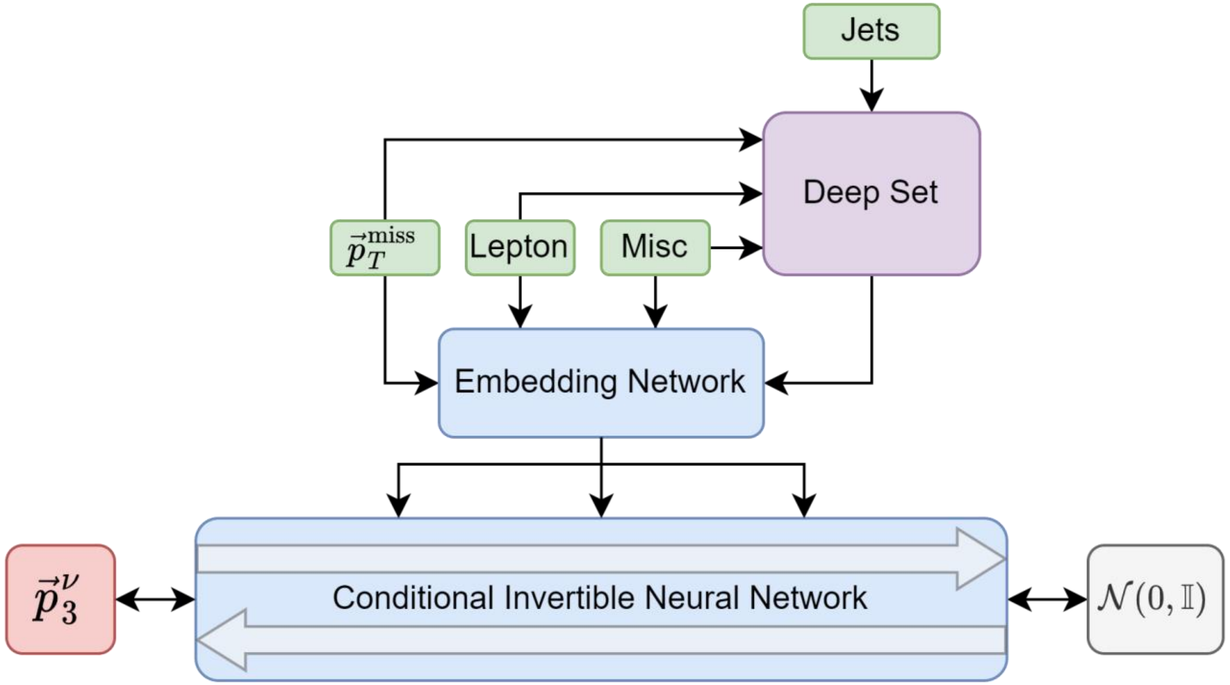
ν-Flows

Conditional Neutrino Regression



Matthew Leigh **Tobias Golling** **Johnny Raine**

ML4Jets 2022

ATLAS EXPERIMENT

UNIVERSITÉ DE GENÈVE

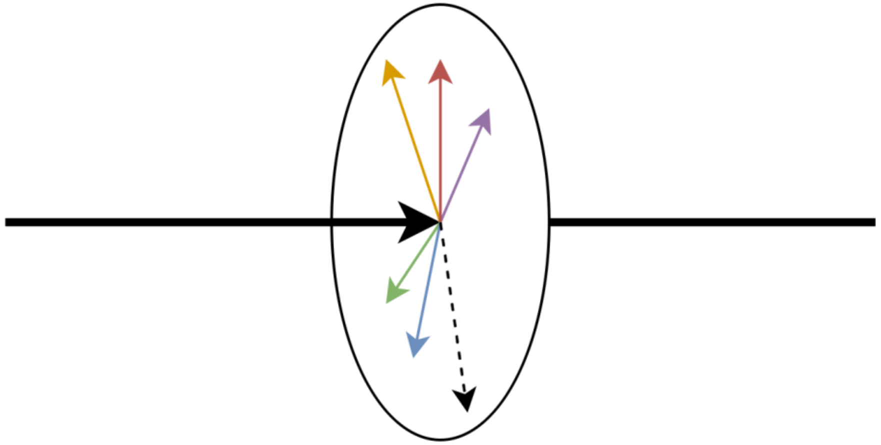
Problem: Neutrino Kinematics

- Neutrinos are not directly observable

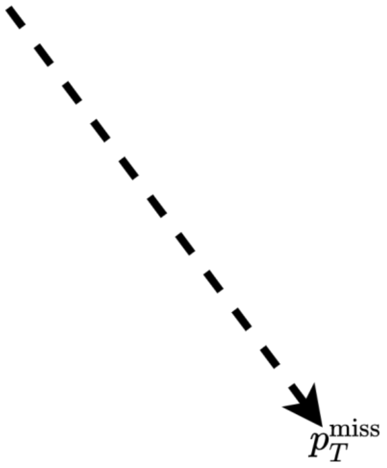
Problem: Neutrino Kinematics

- Neutrinos are not directly observable

- Total neutrino **transverse momenta** is often measured by the experimental proxy $\boldsymbol{p_T^{\mathbf{miss}}}$



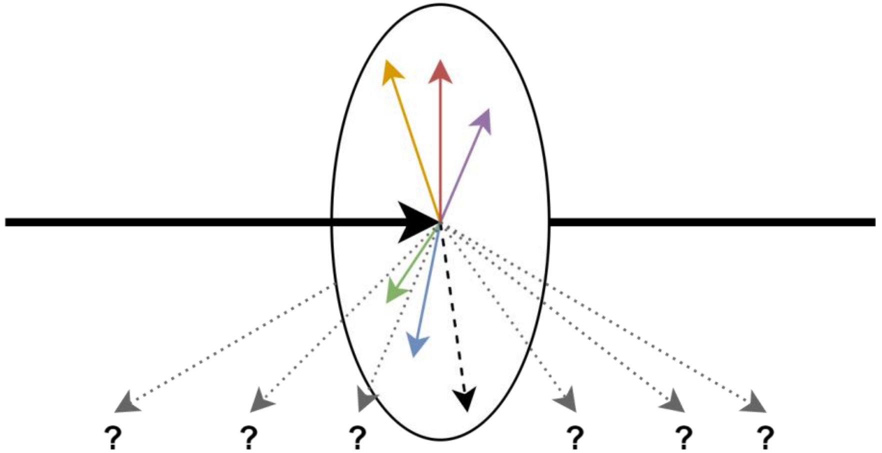
$$\boldsymbol{p_T^{\mathbf{miss}}} = -\sum_{\mathbf{i \in everything}} \boldsymbol{p_T^i}$$



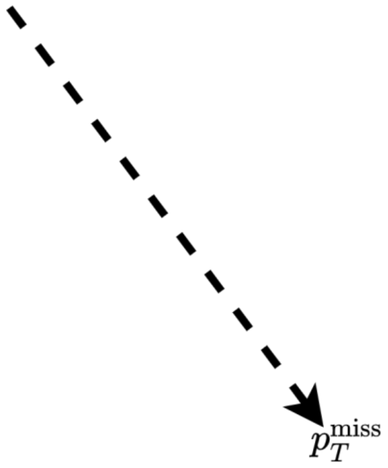
Problem: Neutrino Kinematics

- Neutrinos are not directly observable

- Total neutrino **transverse momenta** is often measured by the experimental proxy $\boldsymbol{p}_T^{\mathbf{miss}}$
  - **No information** about the **longitudinal** momentum

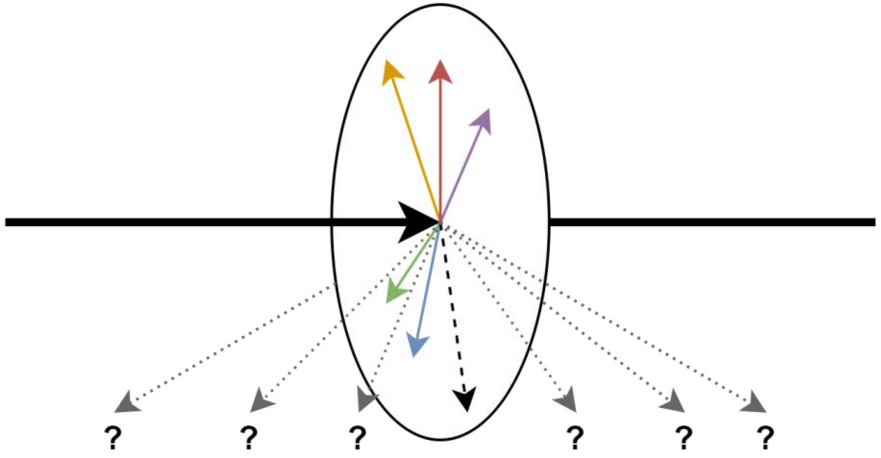


$$\boldsymbol{p}_T^{\mathbf{miss}} = -\sum_{\mathbf{i} \in \mathbf{everything}} \boldsymbol{p}_T^{\boldsymbol{i}}$$

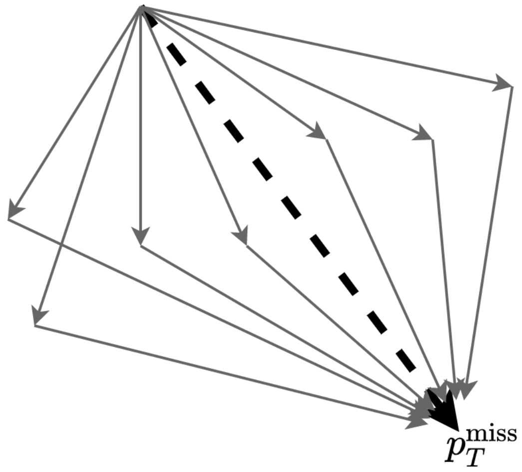


Problem: Neutrino Kinematics

- Neutrinos are not directly observable
- Total neutrino **transverse momenta** is often measured by the experimental proxy $\boldsymbol{p}_{\boldsymbol{T}}^{\textbf{miss}}$
  - **No information** about the **longitudinal** momentum
  - **Transverse** momenta in final states with more than one neutrino are **under-constrained**

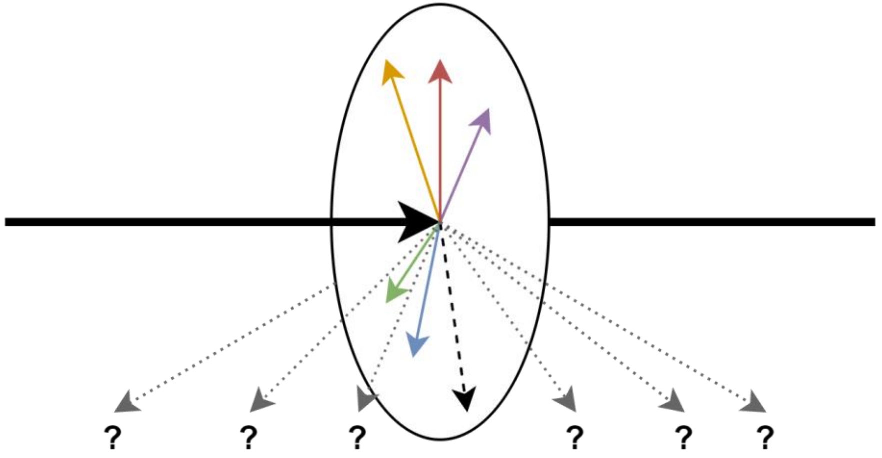


$$\boldsymbol{p}_{\boldsymbol{T}}^{\textbf{miss}} = -\sum\nolimits_{\mathbf{i}\in\textbf{everything}} \boldsymbol{p}_{\boldsymbol{T}}^{\boldsymbol{i}}$$

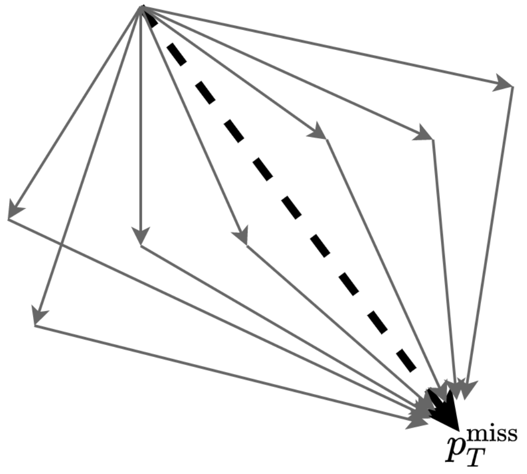


Problem: Neutrino Kinematics

- Neutrinos are not directly observable
- Total neutrino **transverse momenta** is often measured by the experimental proxy $\boldsymbol{p_T^{\text{miss}}}$
  - **No information** about the longitudinal momentum
  - **Transverse** momenta in final states with more than one neutrino are **under-constrained**
- Any indication of the neutrino kinematics could benefit a wide variety of analyses in collider physics

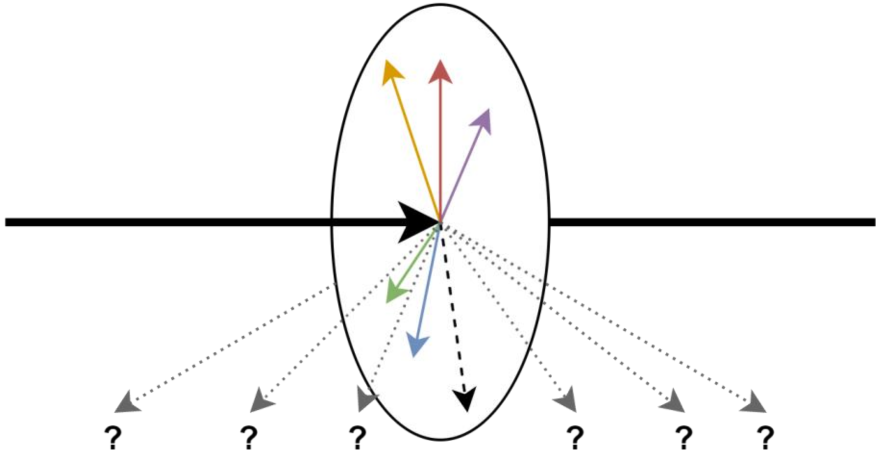


$$\boldsymbol{p_T^{\text{miss}}} = -\sum_{\mathbf{i}\in\textbf{everything}} \boldsymbol{p_T^i}$$

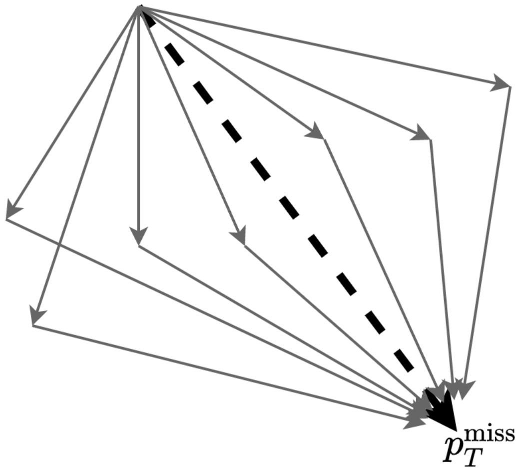


Our Approach

- Many solutions might be **possible** but not equally **likely**
  - **Leverage information** from the event to constrain this likelihood

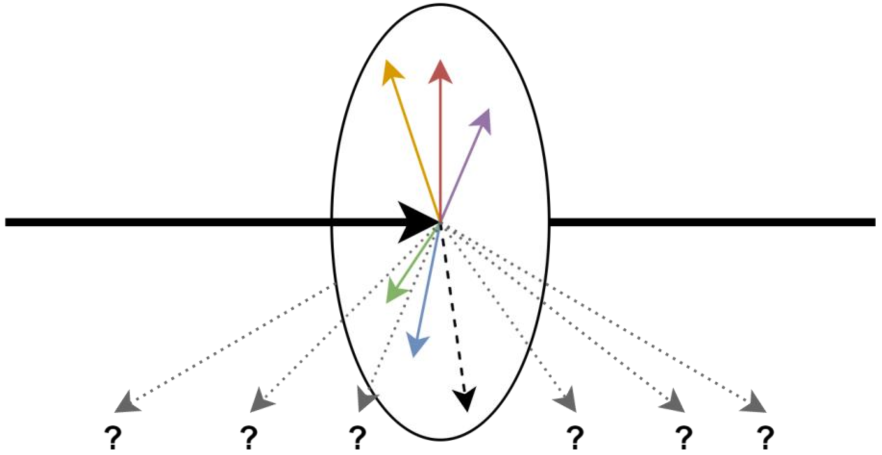


$$p_T^{miss} = -\sum_{i \in everything} p_T^i$$

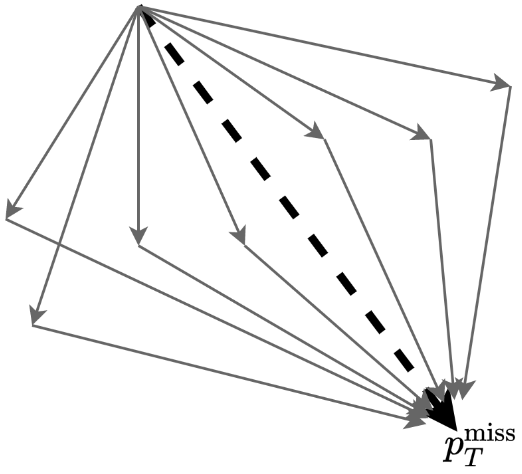


Our Approach

- Many solutions might be **possible** but not equally **likely**
  - **Leverage information** from the event to constrain this likelihood
  - **Leverage an inductive bias**
    - Assume an underlying process



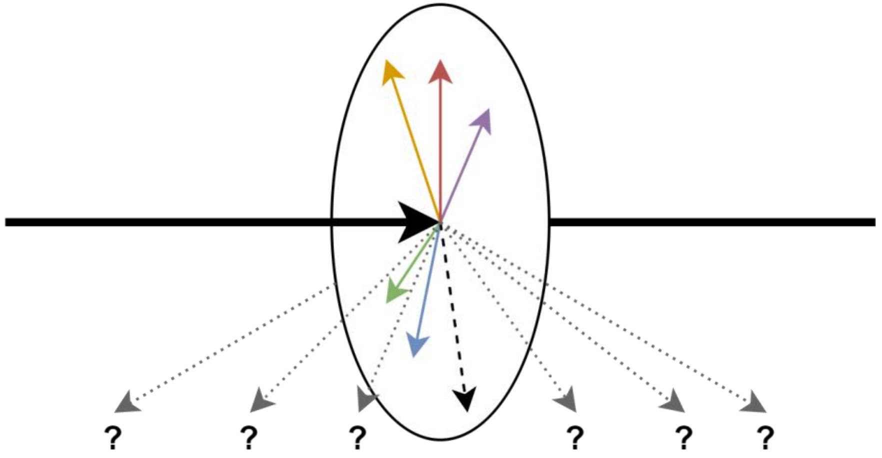
$$\boldsymbol{p}_{\boldsymbol{T}}^{\textbf{miss}} = -\sum_{\textbf{i}\in\textbf{everything}} \boldsymbol{p}_{\boldsymbol{T}}^{\boldsymbol{i}}$$



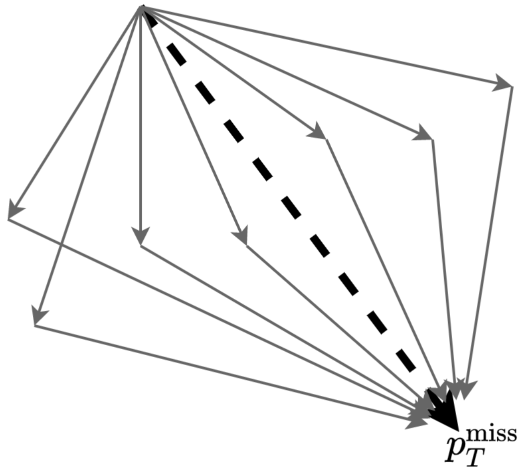
Our Approach

- Many solutions might be **possible** but not equally **likely**
  - **Leverage information** from the event to constrain this likelihood
  - **Leverage an inductive bias**
    - Assume an underlying process

- Trying to recover a **degree of freedom**
  - Standard supervised regression methods might not be applicable

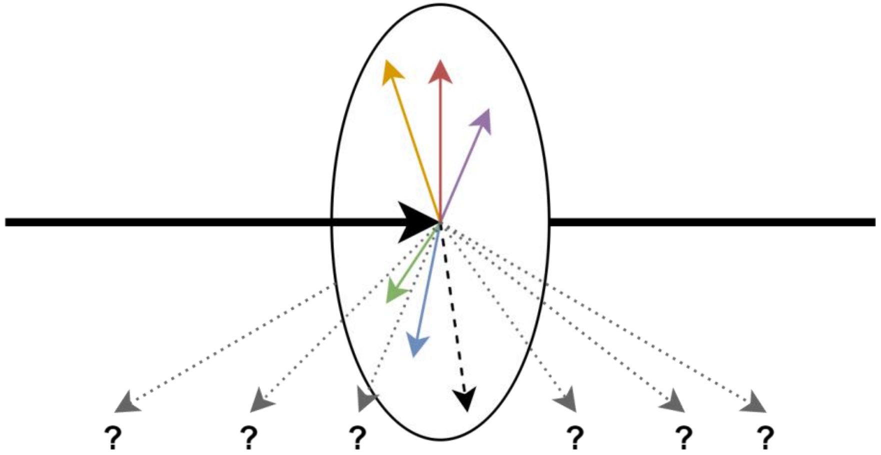


$$\boldsymbol{p_T^{\text{miss}}} = -\sum_{\text{i}\in\text{everything}} \boldsymbol{p_T^i}$$



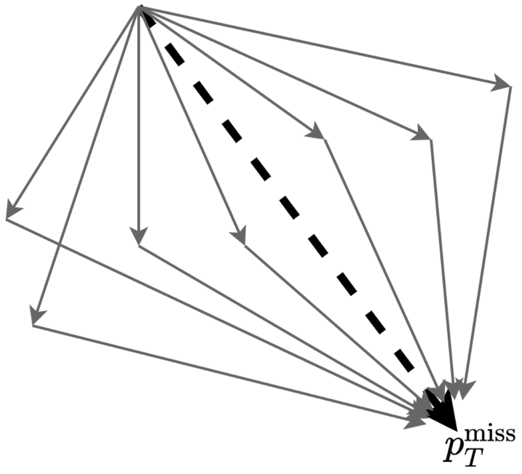
Our Approach

- Many solutions might be **possible** but not equally **likely**
  - **Leverage information** from the event to constrain this likelihood
  - **Leverage an inductive bias**
    - Assume an underlying process



$$p_T^{\mathbf{miss}} = -\sum_{\mathbf{i \in everything}} p_T^{i}$$

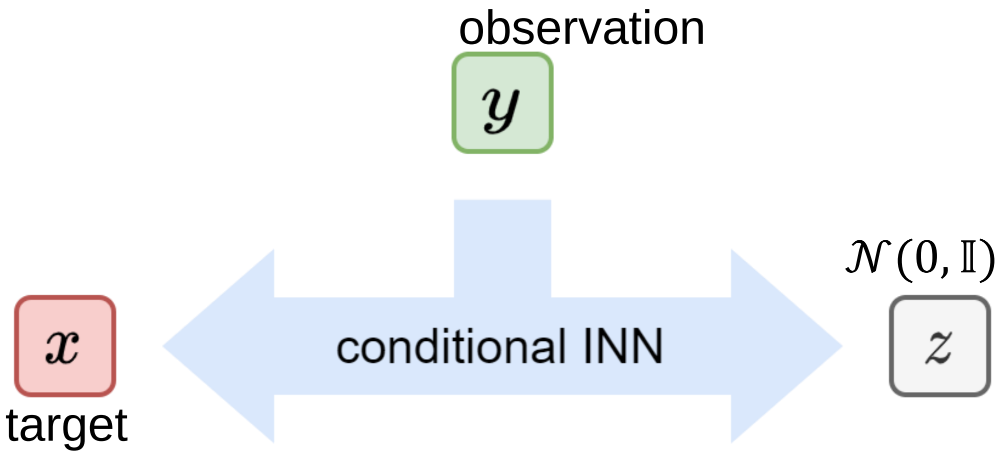
- Trying to recover a **degree of freedom**
  - Standard supervised regression methods might not be applicable



- Propose to a **conditional normalizing flow** to learn a **conditional likelihood** over the neutrino momenta

Conditional Normalising Flows

- **Conditional normalising flows** parameterise an invertible map from $\boldsymbol{x}$ to $\boldsymbol{z}$ given $\boldsymbol{y}$ as **conditioning** inputs
  - $(\boldsymbol{y}, \boldsymbol{x})$ can be seen as **input/target** training pair

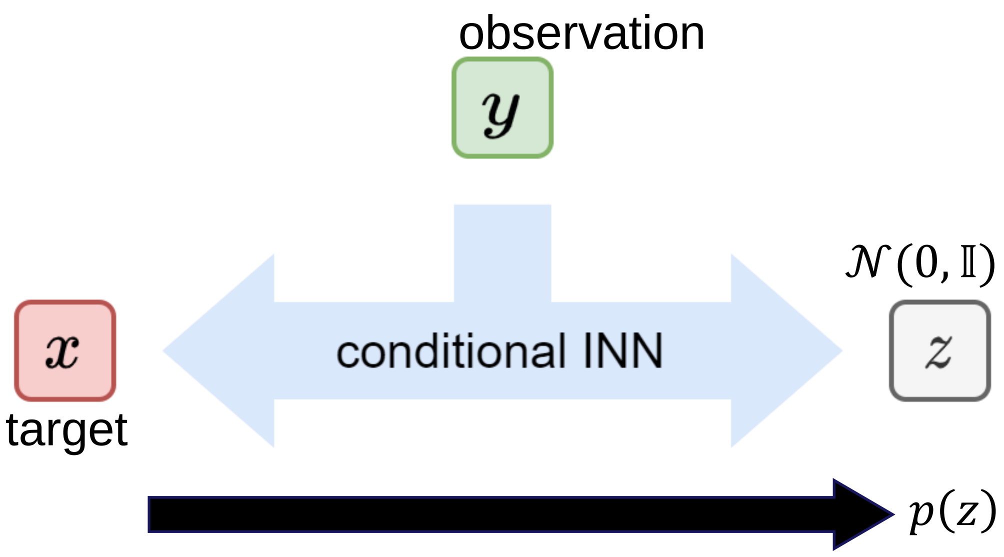


Conditional Normalising Flows

- **Conditional normalising flows** parameterise an invertible map from $\boldsymbol{x}$ to $\boldsymbol{z}$ given $\boldsymbol{y}$ as **conditioning** inputs
  - $(\boldsymbol{y}, \boldsymbol{x})$ can be seen as **input/target** training pair

- **Training:** Model runs forward for **maximum likelihood** objective

$$\begin{aligned} Loss(y, x) &= -\log(p_X(x|y)) \\ &= -\log\left(p_z(f_\theta(x|y))\right) - \log\left|\det(J(x|y))\right| \end{aligned}$$



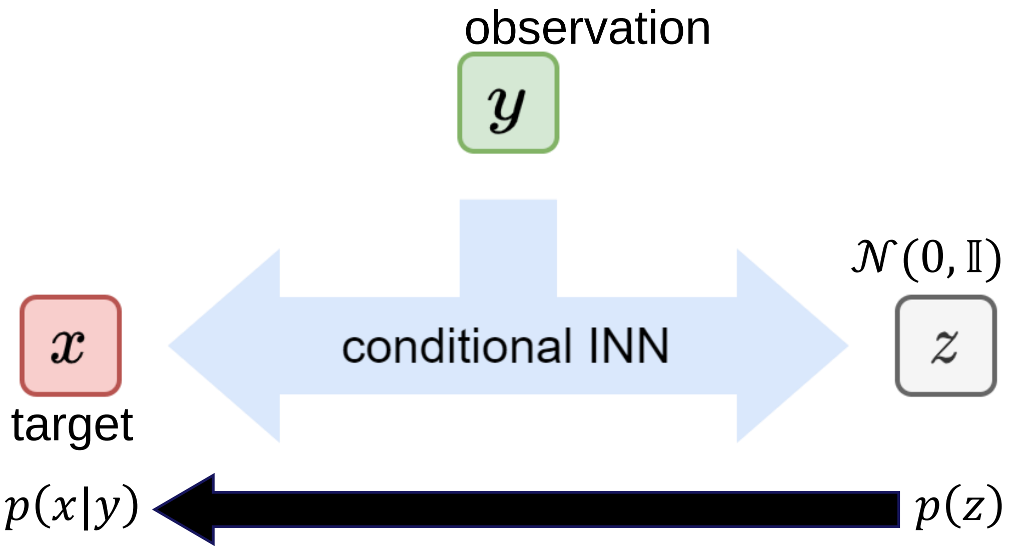
Conditional Normalising Flows

- **Conditional normalising flows** parameterise an invertible map from $\boldsymbol{x}$ to $\boldsymbol{z}$ given $\boldsymbol{y}$ as **conditioning** inputs
  - $(\boldsymbol{y}, \boldsymbol{x})$ can be seen as **input/target** training pair

- **Training:** Model runs forward for **maximum likelihood** objective

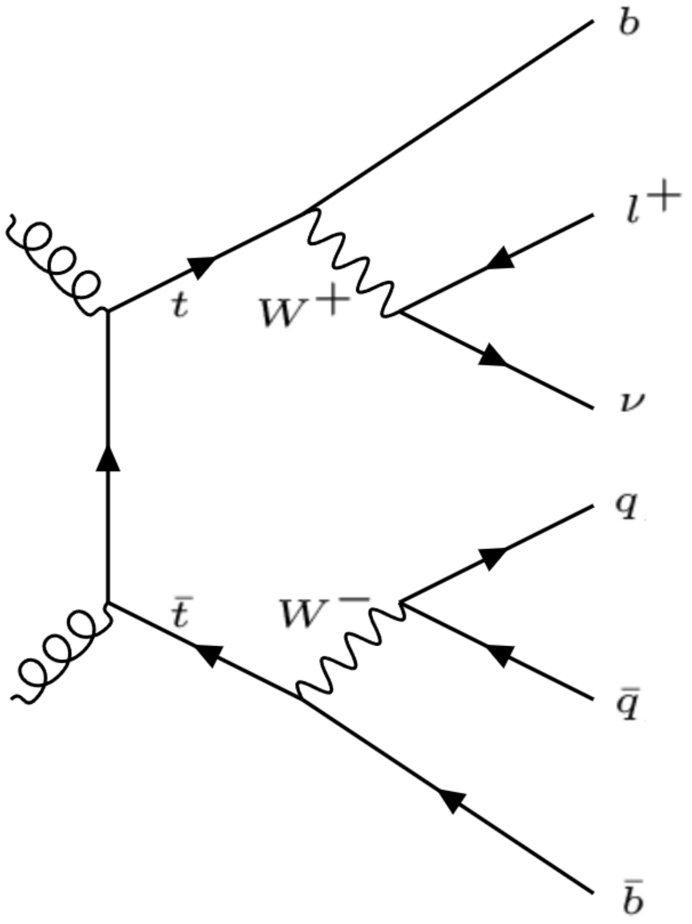
$$
\begin{aligned}
Loss(y, x) &= -\log(p_X(x|y)) \\
&= -\log\left(p_z\left(f_\theta(x|y)\right)\right) - \log\left|\det\left(J(x|y)\right)\right|
\end{aligned}
$$

- **Sampling:** Model runs in reverse giving $p(x|y)$



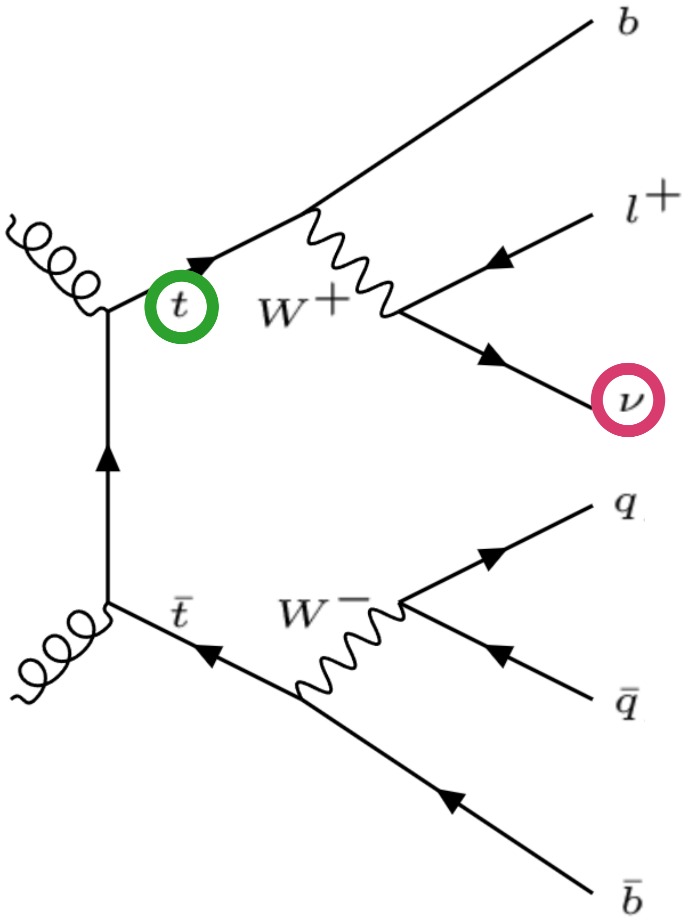
Case Study: Single Leptonic $t\bar{t}$

- Commonly studied process is the **single leptonic $t\bar{t}$** decay



Case Study: Single Leptonic $t\bar{t}$

- Commonly studied process is the **single leptonic $t\bar{t}$** decay
- Full properties of **leptonic top** not directly measurable due to the unknown **longitudinal momentum of the neutrino** in the final state

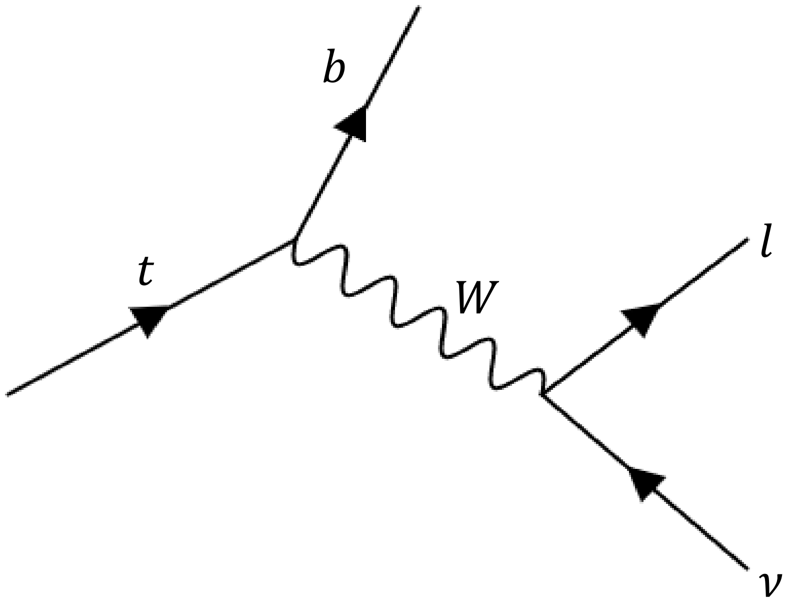


Kinematic Solution: $W \to l\nu$

- If we **assume** that
  - $(p_x^\nu, p_y^\nu) = (p_x^{\mathrm{miss}}, p_y^{\mathrm{miss}})$
  - $m_W = 80.38\ \mathrm{GeV}$

- Can **solve** for neutrino's **longitudinal** momentum:

$$p_z^\nu = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



$$a = (p_z^\ell)^2 - (E^\ell)^2,$$

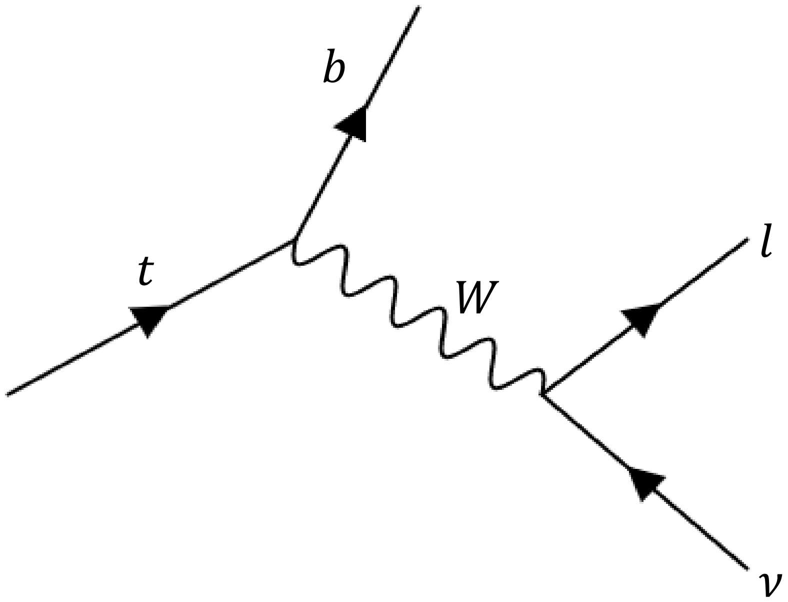
$$b = \alpha p_z^\ell,$$

$$c = \frac{\alpha^2}{4} - (E^\ell)^2 (p_\mathrm{T}^\nu)^2,$$

$$\alpha = m_W^2 - m_\ell^2 + 2(p_x^\ell, p_x^\nu + p_y^\ell p_y^\nu).$$

Kinematic Solution: $W \rightarrow l\nu$

- If we **assume** that
  - $\left(p_x^\nu, p_y^\nu\right) = (p_x^{\text{miss}}, p_y^{\text{miss}})$
  - $m_W = 80.38 \text{ GeV}$



- Can **solve** for neutrino's **longitudinal** momentum:

$$p_z^\nu = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- **Gives two solutions** with no preference
  - Take the estimate **closer to zero**
  - Consider **both possibilities** in any downstream tasks

$$a = (p_z^\ell)^2 - (E^\ell)^2,$$

$$b = \alpha p_z^\ell,$$

$$c = \frac{\alpha^2}{4} - (E^\ell)^2 (p_{\mathrm{T}}^\nu)^2,$$

$$\alpha = m_W^2 - m_\ell^2 + 2(p_x^\ell, p_x^\nu + p_y^\ell p_y^\nu).$$

Kinematic Solution: $W \rightarrow l\nu$

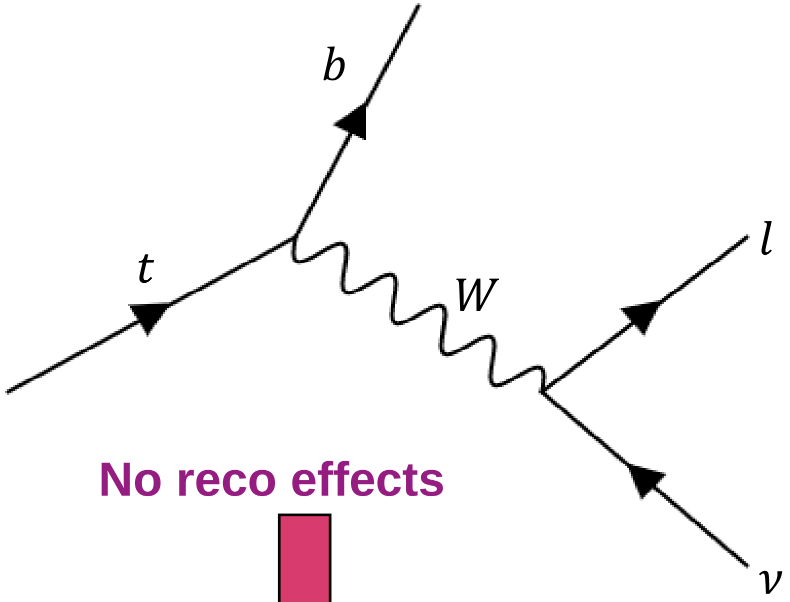
- If we **assume** that
  - $(p_x^{\nu}, p_y^{\nu}) = (p_x^{\text{miss}}, p_y^{\text{miss}})$
  - $m_W = 80.38\ \text{GeV}$

No off-shell
No reco effects

- Can **solve** for neutrino's **longitudinal** momentum:

$$p_z^{\nu} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- **Gives two solutions** with no preference
  - Take the estimate **closer to zero**
  - Consider **both possibilities** in any downstream tasks



No reco effects

$$a = (p_z^{\ell})^2 - (E^{\ell})^2,$$

$$b = \alpha p_z^{\ell},$$

$$c = \frac{\alpha^2}{4} - (E^{\ell})^2 (p_{\mathrm{T}}^{\nu})^2,$$

$$\alpha = m_W^2 - m_{\ell}^2 + 2(p_x^{\ell}, p_x^{\nu} + p_y^{\ell} p_y^{\nu}).$$

Kinematic Solution: $W \to l\nu$

- If we **assume** that
  - $(p_x^\nu, p_y^\nu) = (p_x^{\text{miss}}, p_y^{\text{miss}})$
  - $m_W = 80.38\ \text{GeV}$

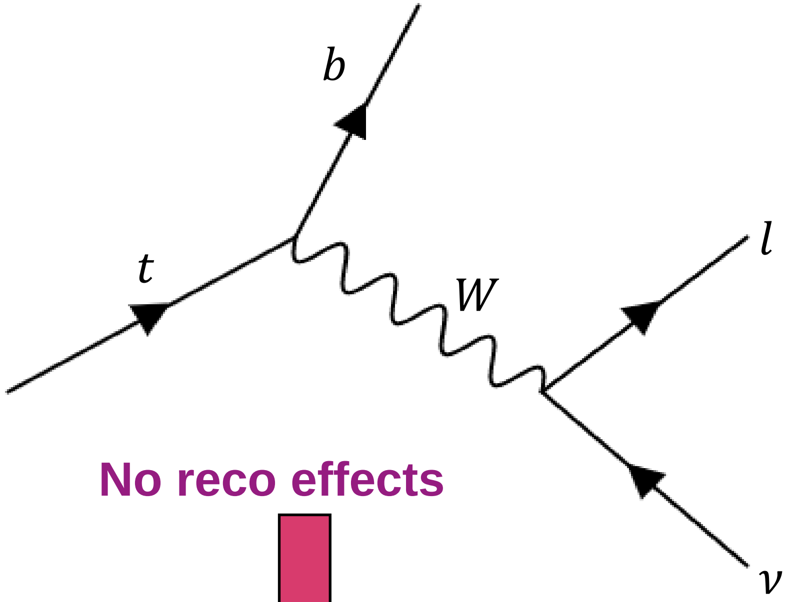
No off-shell
No reco effects

- Can **solve** for neutrino's **longitudinal** momentum:

$$p_z^\nu = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Sometimes lead to no real solutions!

- **Gives two solutions** with no preference
  - Take the estimate **closer to zero**
  - Consider **both possibilities** in any downstream tasks



No reco effects

$$a = (p_z^\ell)^2 - (E^\ell)^2,$$

$$b = \alpha p_z^\ell,$$

$$c = \frac{\alpha^2}{4} - (E^\ell)^2 (p_{\mathrm{T}}^\nu)^2,$$

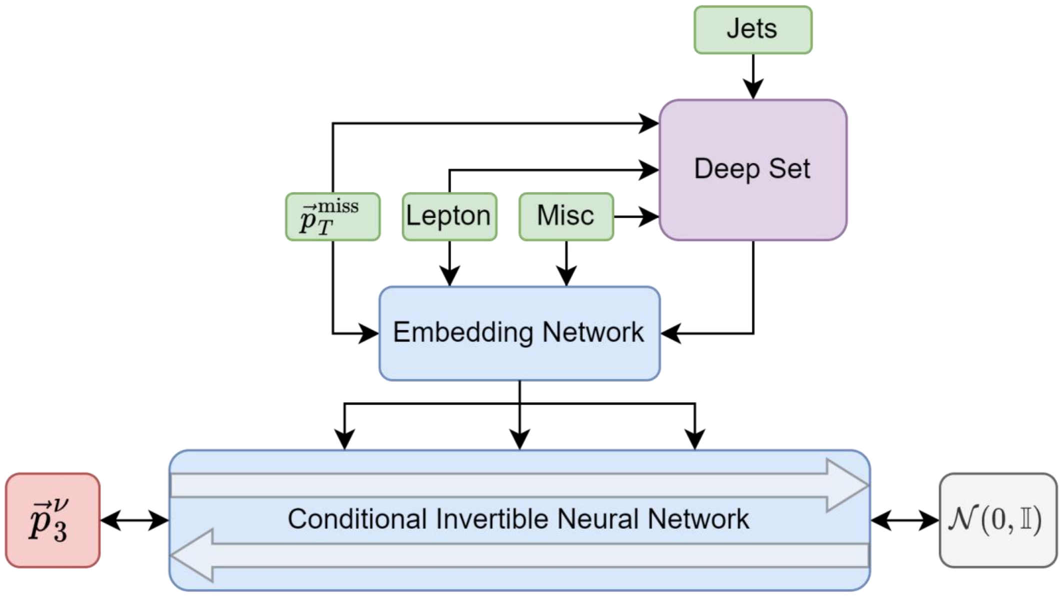
$$\alpha = m_W^2 - m_\ell^2 + 2(p_x^\ell, p_x^\nu + p_y^\ell p_y^\nu).$$

ν-Flow Overview

- Trained using simulated single leptonic $t\bar{t}$

  https://doi.org/10.5281/zenodo.6782987

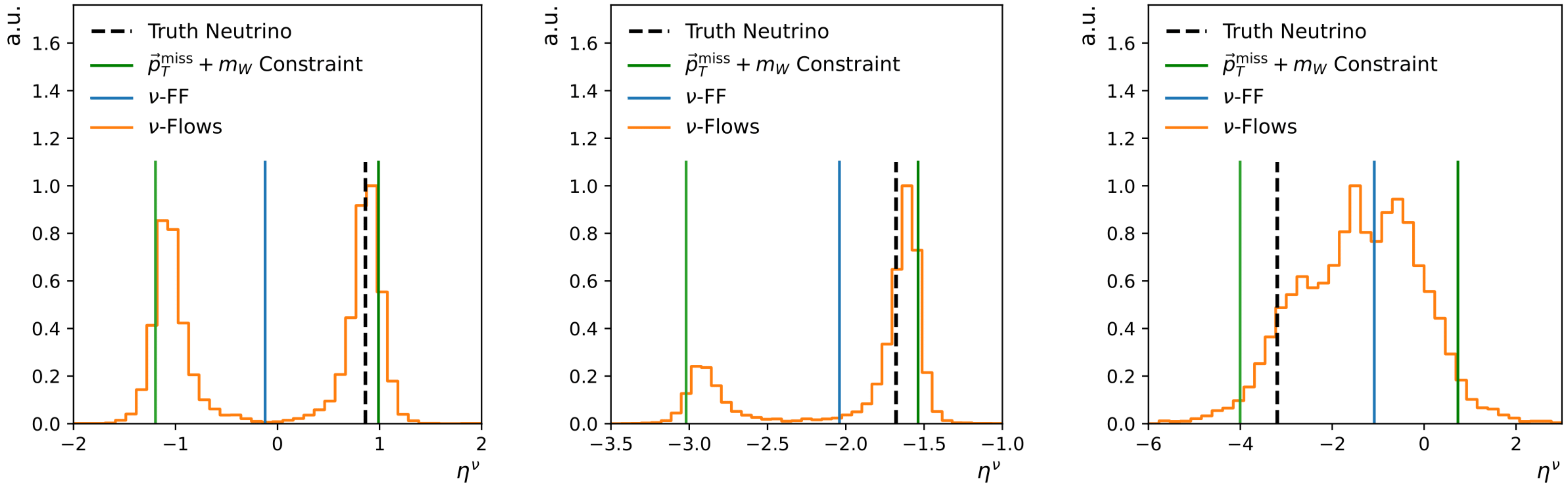
Category	Variables	Description
$\vec{p}_{\mathrm{T}}^{\mathrm{miss}}$	$p_x^{\mathrm{miss}}, p_y^{\mathrm{miss}}$	Missing transverse momentum 2-vector
Lepton	$p_x^\ell, p_y^\ell, \eta^\ell, \log E^\ell$	Lepton momentum 4-vector
	ℓ^{flav}	Whether lepton is an electron or muon
Jets	$p_x^j, p_y^j, \eta^j, \log E^j$	Jet momentum 4-vector
	isB	If jet met b-tagging criteria
Misc	$N_{\mathrm{jets}}, N_{\mathrm{bjets}}$	Jet and b-jet multiplicities in the event



- Combines observations + assumptions in fully probabilistic way
- Can scale to multiple neutrinos

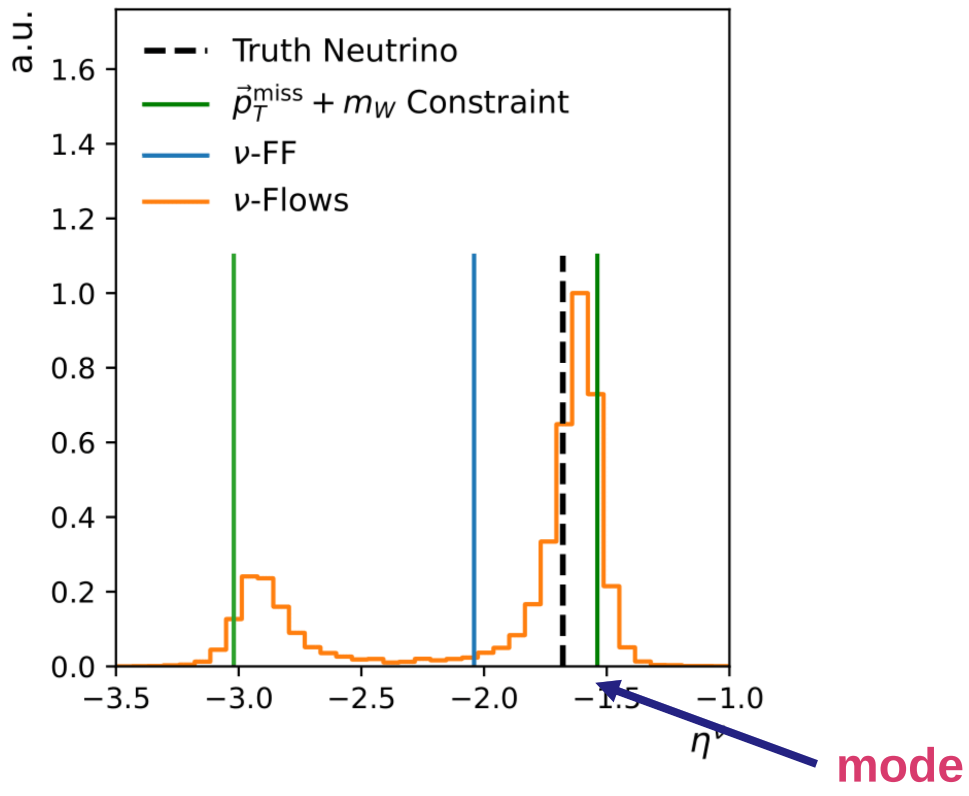
Inference on Individual Events

Cherry picked representative examples



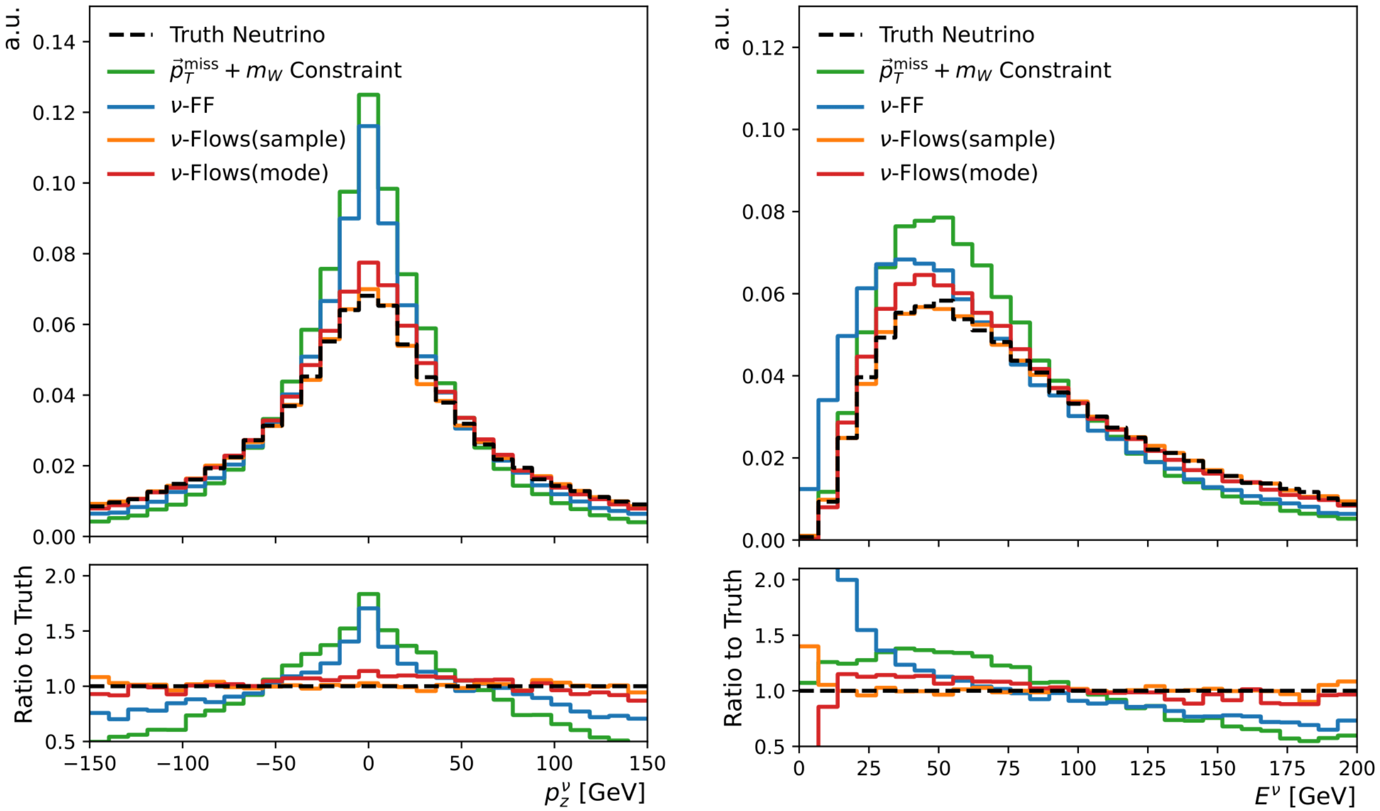
Results: Neutrino Kinematics

- Neutrino kinematic distributions for different methods of generation
  - **ν-Flows(sample)**: Take **one sample** from $p(x|y)$
  - **ν-Flows(mode)**: Take **1024** samples keep one with **highest likelihood under flow**



Results: Neutrino Kinematics

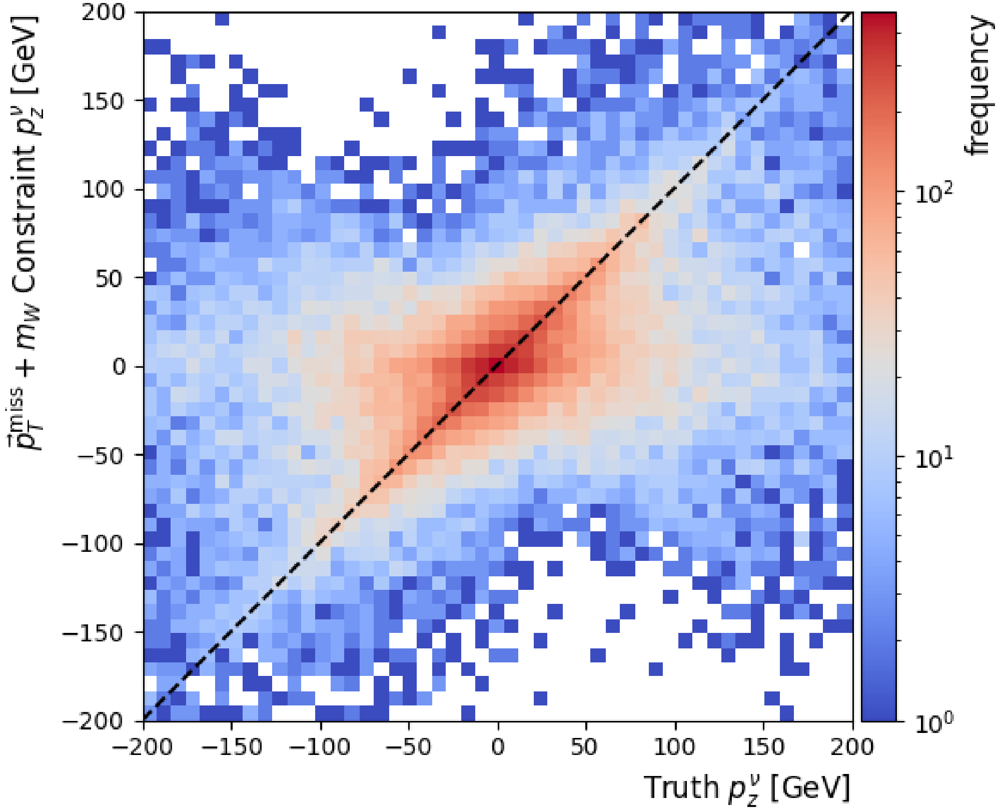
- Neutrino kinematic distributions for different methods of generation
  - **ν-Flows(sample)**: Take **one sample** from $p(x|y)$
  - **ν-Flows(mode)**: Take **256** samples keep one with **highest likelihood under flow**



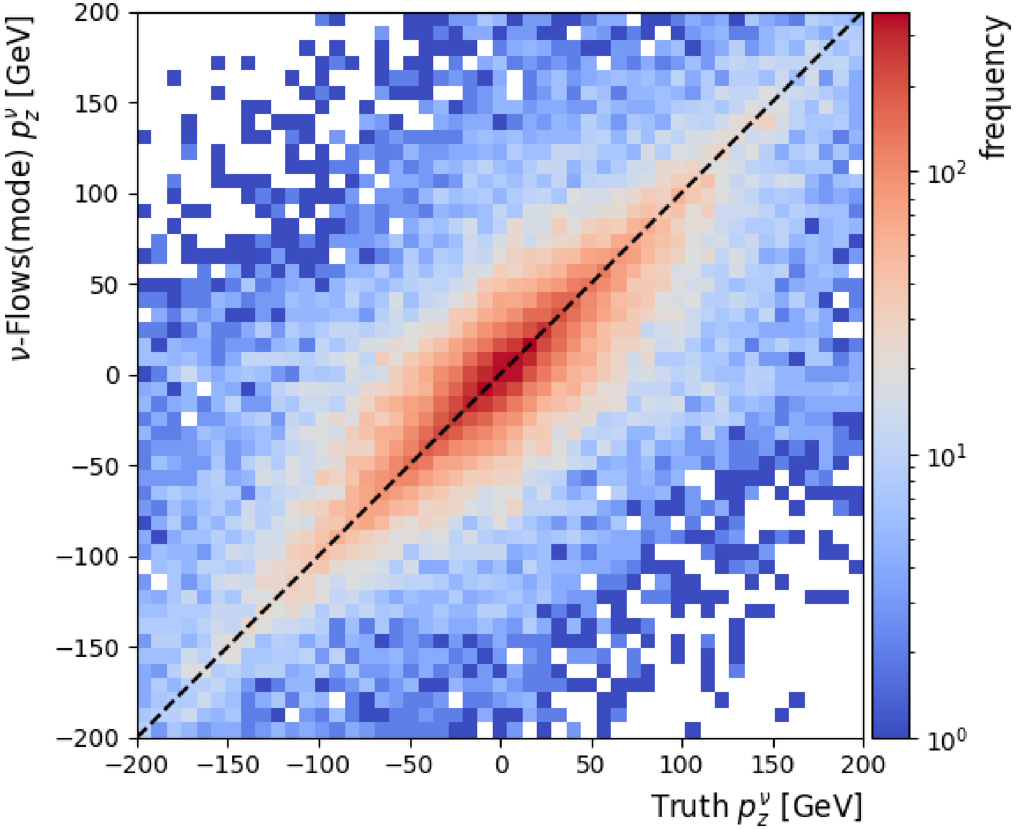
Results: Neutrino Kinematics

- **Reconstructed p_z** vs **Truth p_z**

$p_T^{miss} + \ m_W$ **Constraint**

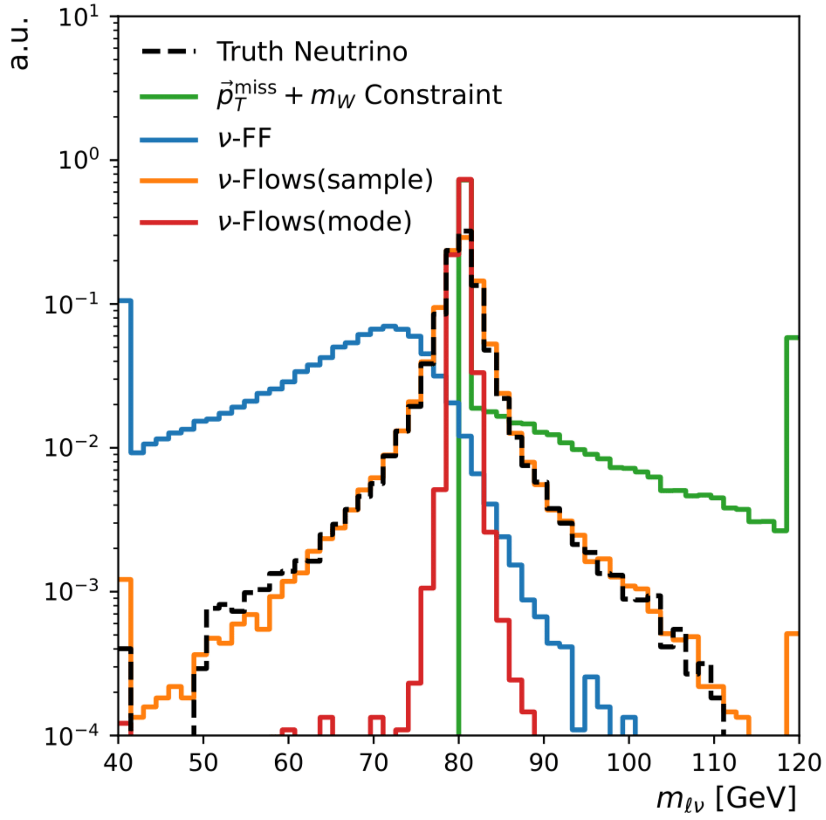


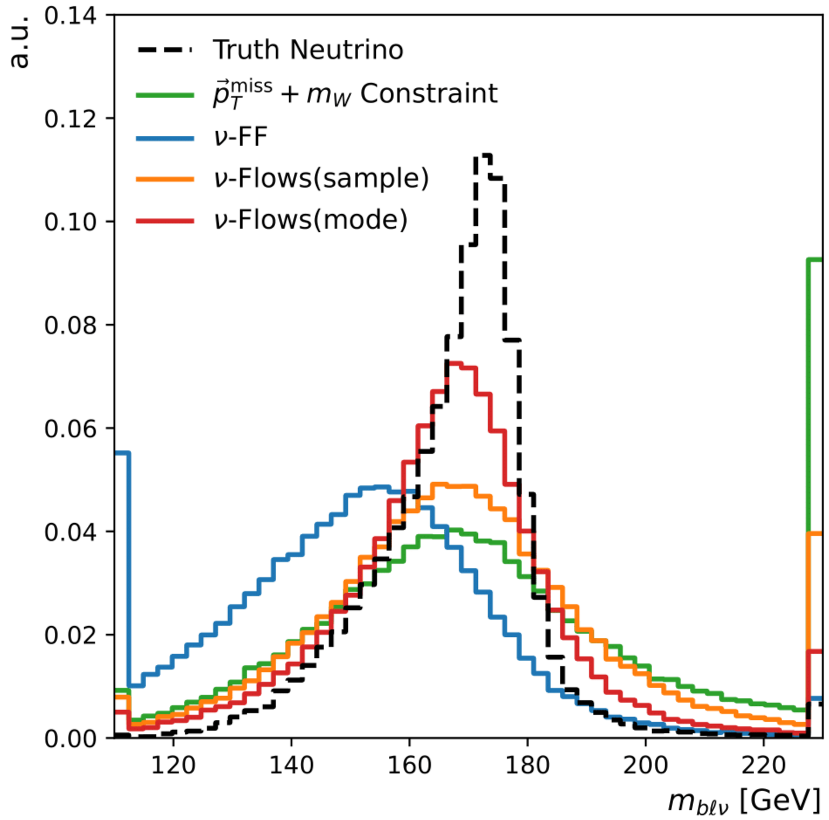
ν**-Flows(mode)**



Results: Invariant Mass Reconstruction

- **Invariant mass** reconstruction of the **leptonic W** and **leptonic t**
  - t is reconstructed **using the correct b-jet** (not guaranteed in data)

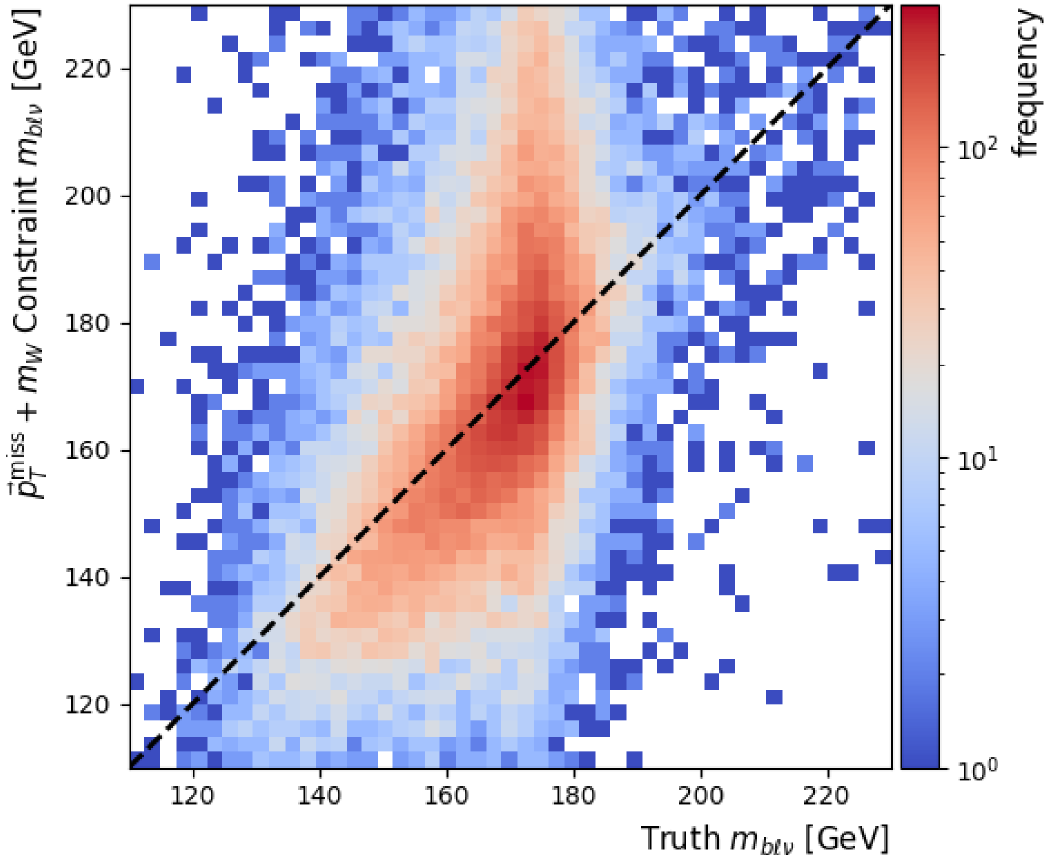




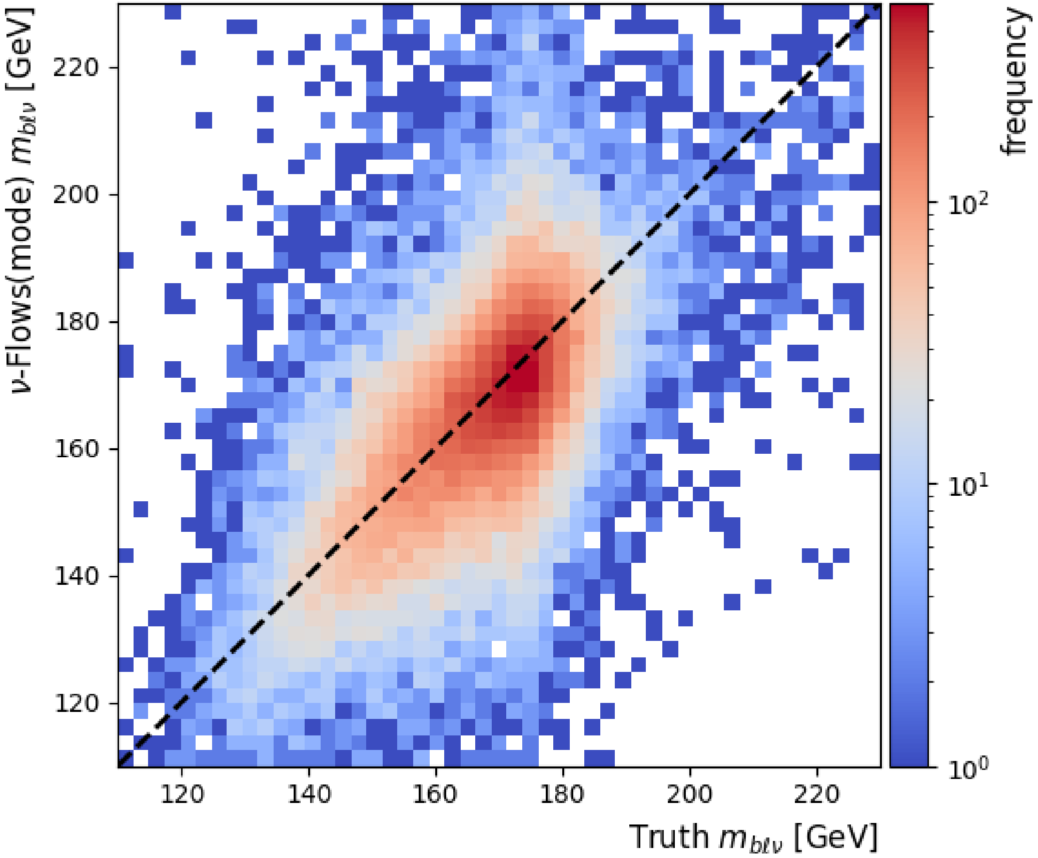
Results: Invariant Mass Reconstruction

- **Reconstructed m_t** vs **Truth m_t**

$p_T^{miss} + \ m_W$ **Constraint**



ν-Flows(mode)



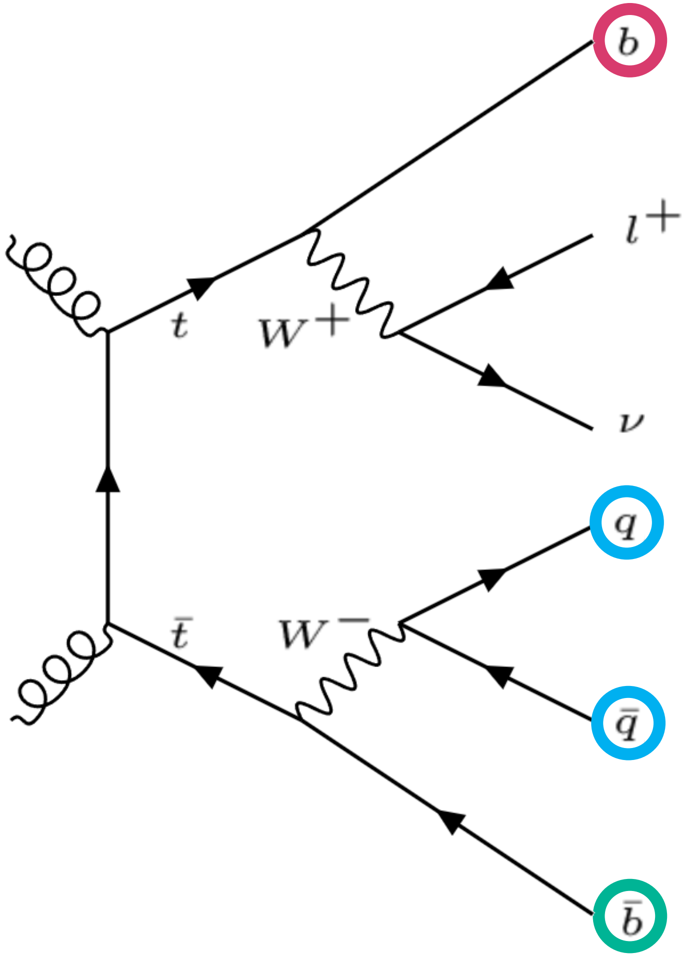
χ^2: Downstream Task Example

- Don't know which of our reconstructed jets correspond to the b_{lep}, b_{had}, q_1, or q_2
  - Up to 9 reconstructed jets

- Test **all possible** jet **permutations**
- Take permutation with **smallest χ^2**

$$\chi^2 = \frac{(m_W - m_{\ell\nu})^2}{\sigma_{\ell\nu}} + \frac{(m_W - m_{qq})^2}{\sigma_{qq}} + \frac{(m_t - m_{b\ell\nu})^2}{\sigma_{b\ell\nu}} + \frac{(m_t - m_{bqq})^2}{\sigma_{bqq}}$$

- Example of a method used in many combinatoric solving approaches (χ^2, KLFitter, etc)



χ^2: Association Results

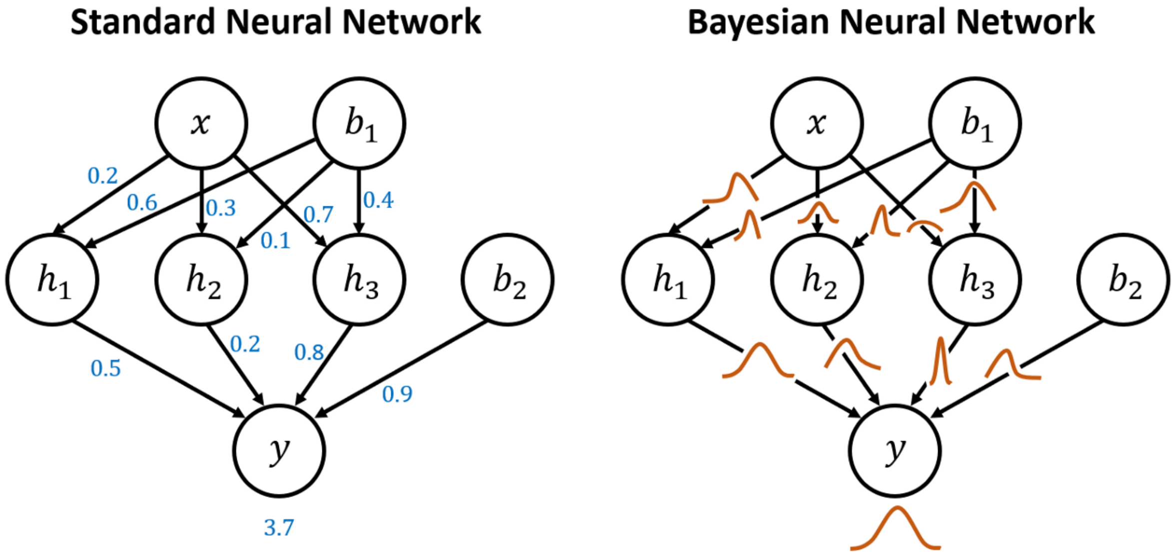
- **Association accuracy** of the b_{lep} verses the **number of jets**
  - Parton with highest dependance on neutrino in χ^2 fit
- Events where the **4 signal jets** were reconstructed

	Number of Jets					
Neutrino Type	4	5	6	7	8	9
Truth Neutrino	0.864	0.753	0.686	0.641	0.611	0.587
$\vec{p}_{\mathrm{T}}^{\mathrm{miss}}$ and m_W Constraint	0.790	0.576	0.476	0.398	0.366	0.286
ν-FF	0.754	0.533	0.410	0.353	0.300	0.302
ν-Flows(sample)	0.803	0.624	0.515	0.457	0.391	0.357
ν-Flows(mode)	**0.813**	**0.664**	**0.575**	**0.508**	**0.481**	**0.405**

***ν*-Flow (mode) improves** upon kinematic solution by factor of **1.03 to 1.41**

Bayesian Normalising Flows

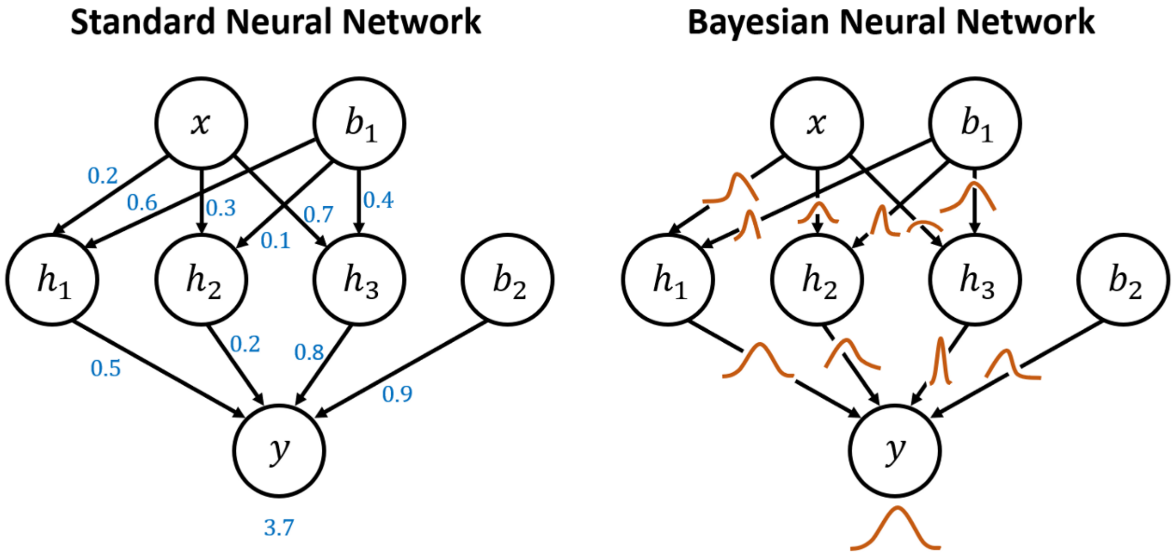
- **Bayesian** networks account for uncertainty in the network's parameters θ



https://towardsdatascience.com/why-you-should-use-bayesian-neural-network-aaf76732c150

Bayesian Normalising Flows

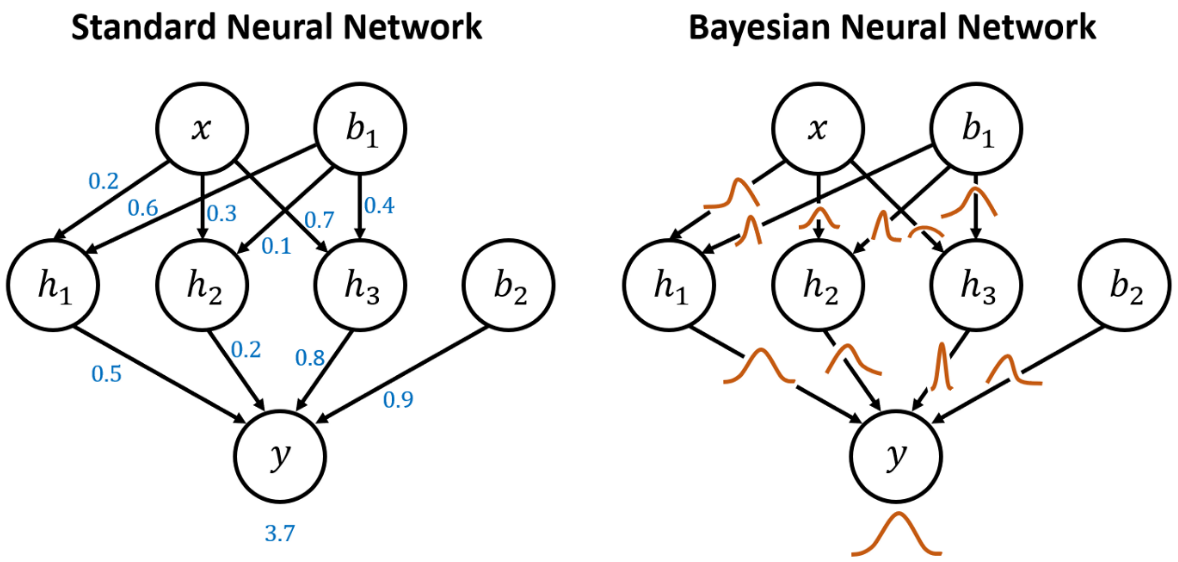
- **Bayesian** networks account for uncertainty in the network's parameters θ
- Switch from a deterministic mapping f to probabilistic transformation



https://towardsdatascience.com/why-you-should-use-bayesian-neural-network-aaf76732c150

Bayesian Normalising Flows

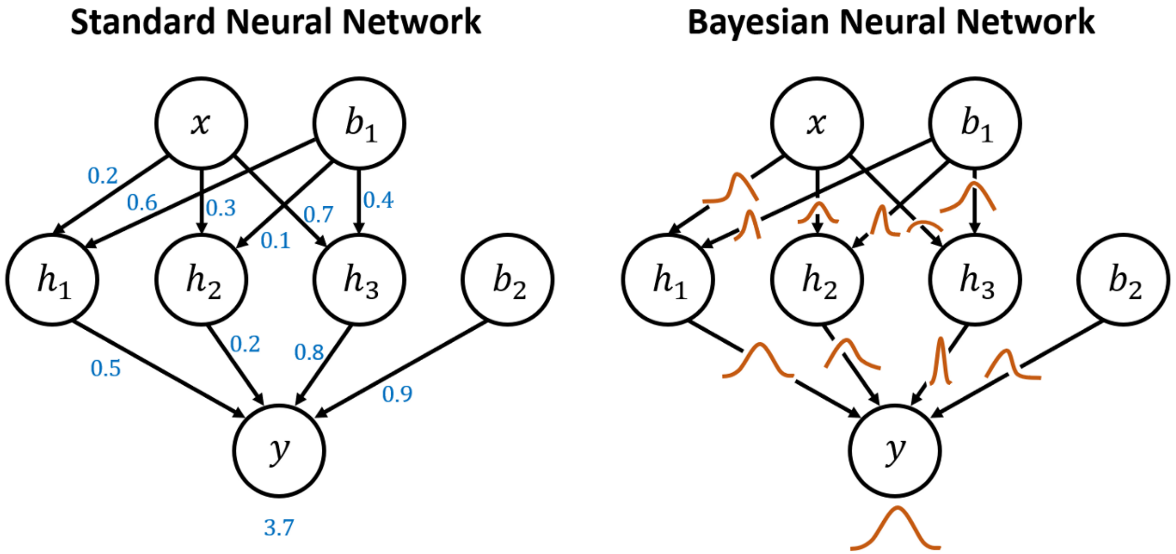
- **Bayesian** networks account for uncertainty in the network's parameters θ
- Switch from a deterministic mapping f to probabilistic transformation
- Parametrise the weights as **mean field gaussians** $\boldsymbol{q_{\phi}(\theta)}$ and use **VI** to ensure that they do not stray too far from **prior** $\boldsymbol{p(\theta)}$



https://towardsdatascience.com/why-you-should-use-bayesian-neural-network-aaf76732c150

Bayesian Normalising Flows

- **Bayesian** networks account for uncertainty in the network's parameters θ
- Switch from a deterministic mapping f to probabilistic transformation
- Parametrise the weights as **mean field gaussians** $\boldsymbol{q_{\phi}(\theta)}$ and use **VI** to ensure that they do not stray too far from **prior** $\boldsymbol{p(\theta)}$

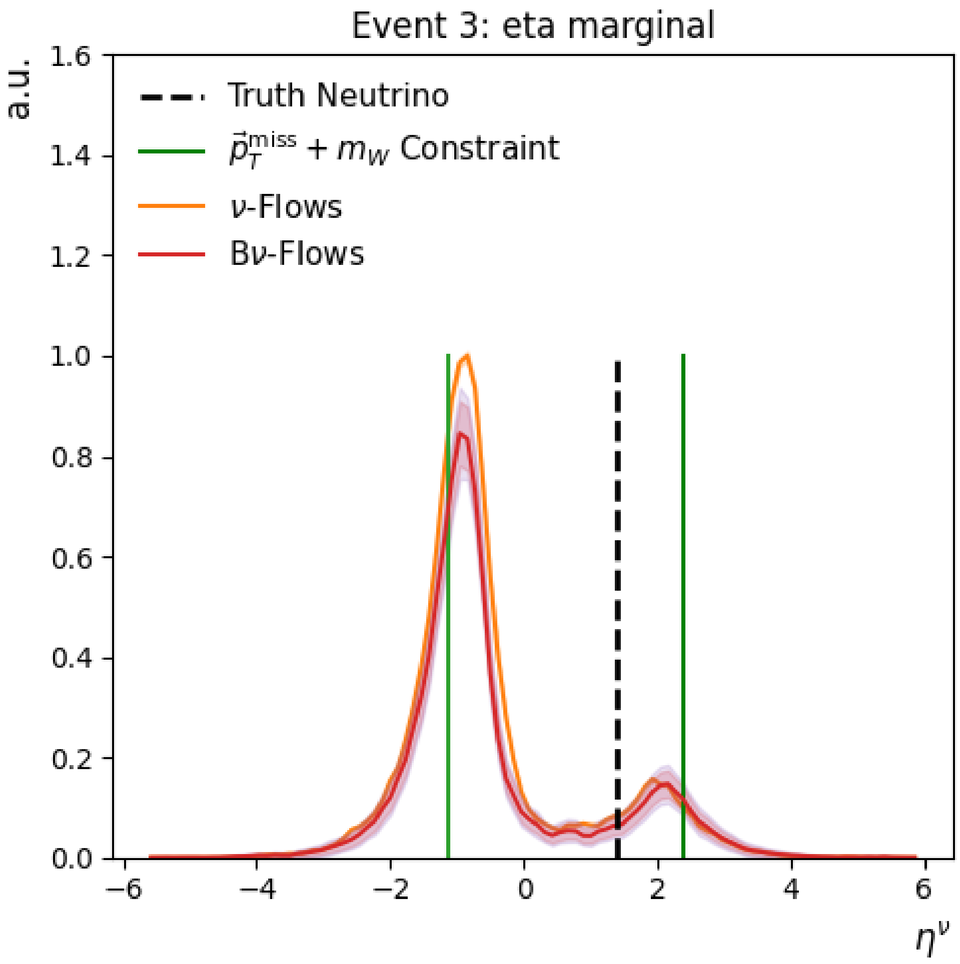


https://towardsdatascience.com/why-you-should-use-bayesian-neural-network-aaf76732c150

$$Loss(y,x) = -\log\left(p_z\left(f_\theta(x|y)\right)\right) - \log\left|\det\left(J(x|y)\right)\right| + \mathrm{KL}\left(q_\phi(\theta)\middle|p(\theta)\right)$$

Bayesian Flow Inference

- Allows us to sample under the base distribution **AND** under the network parameters
- Gives variance of the network's predictions
- Uncertainty from missing data
- Can signify the stability of the flow

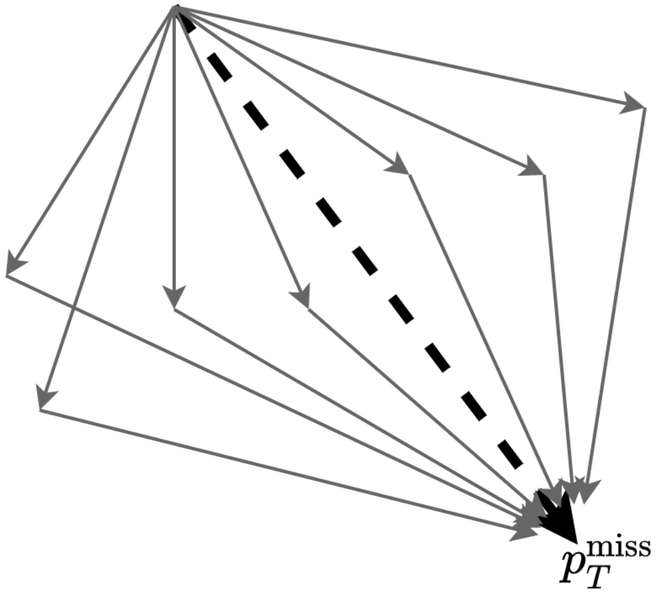


Conclusions

- Demonstrated an approach of using normalizing flow for neutrino momentum regression
- Reconstruction with the flow yields better distributions for p_z^{ν}, $m_{l\nu}$, and $m_{bl\nu}$
- Demonstrated benefits in example downstream task of jet association
- ArXiv pre-print available:
  - https://arxiv.org/abs/2207.00664v2

Next steps

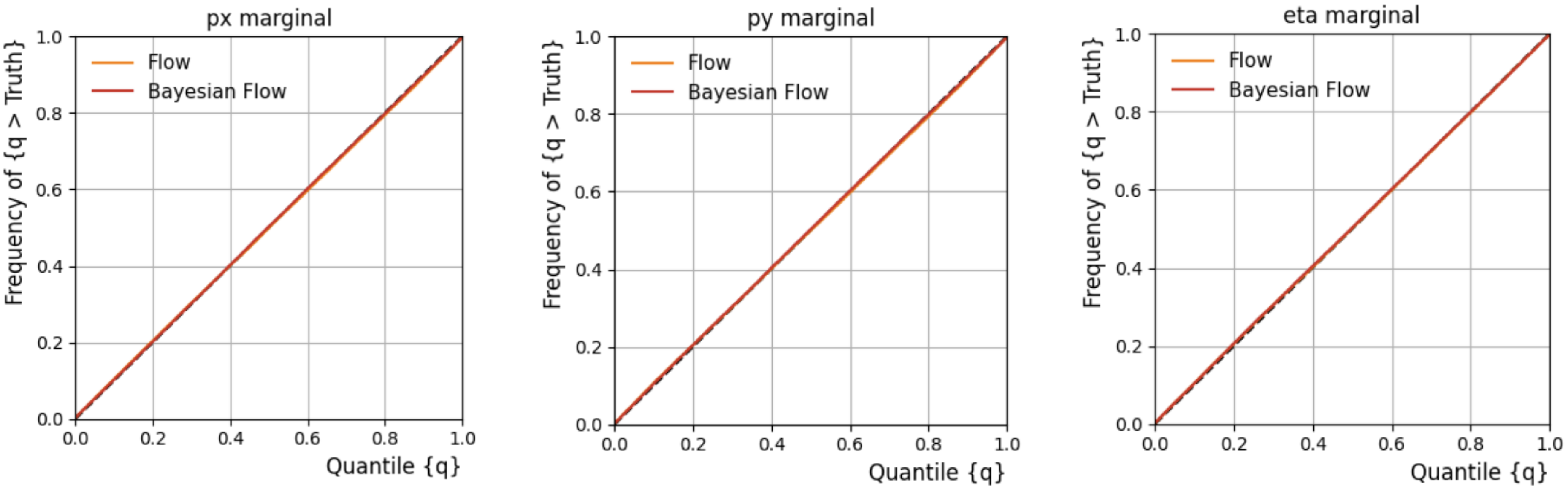
- Move to two neutrino case
- Use the flow / Bayesian flow as an event filter



Thank You

Interpretable Uncertainty

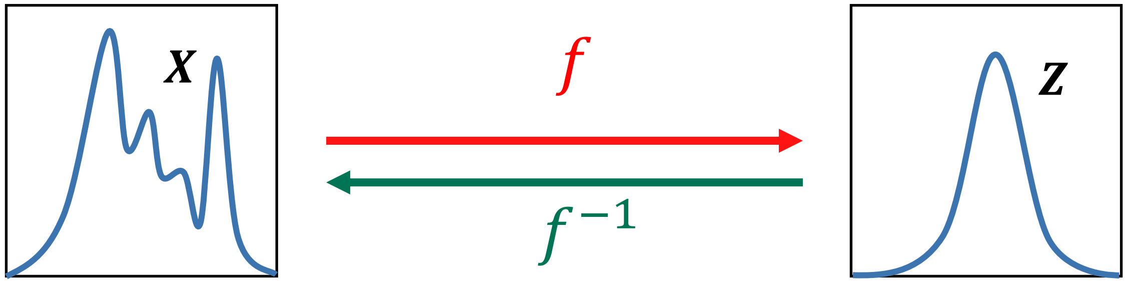
- The distribution of the flow does seem to correspond to real world accuracy!
- **Observed Accuracy** verses **Predicted Confidence**



Normalizing Flows

- A normalising flow is a transformation that typically maps a **complex distribution** $\boldsymbol{p_X(x)}$ into a **simple distribution** $\boldsymbol{p_Z(z)}$
  - $\boldsymbol{Z} = f(\boldsymbol{X})$ with an **invertible** and **differentiable** $f(x)$

"diffeomorphism"



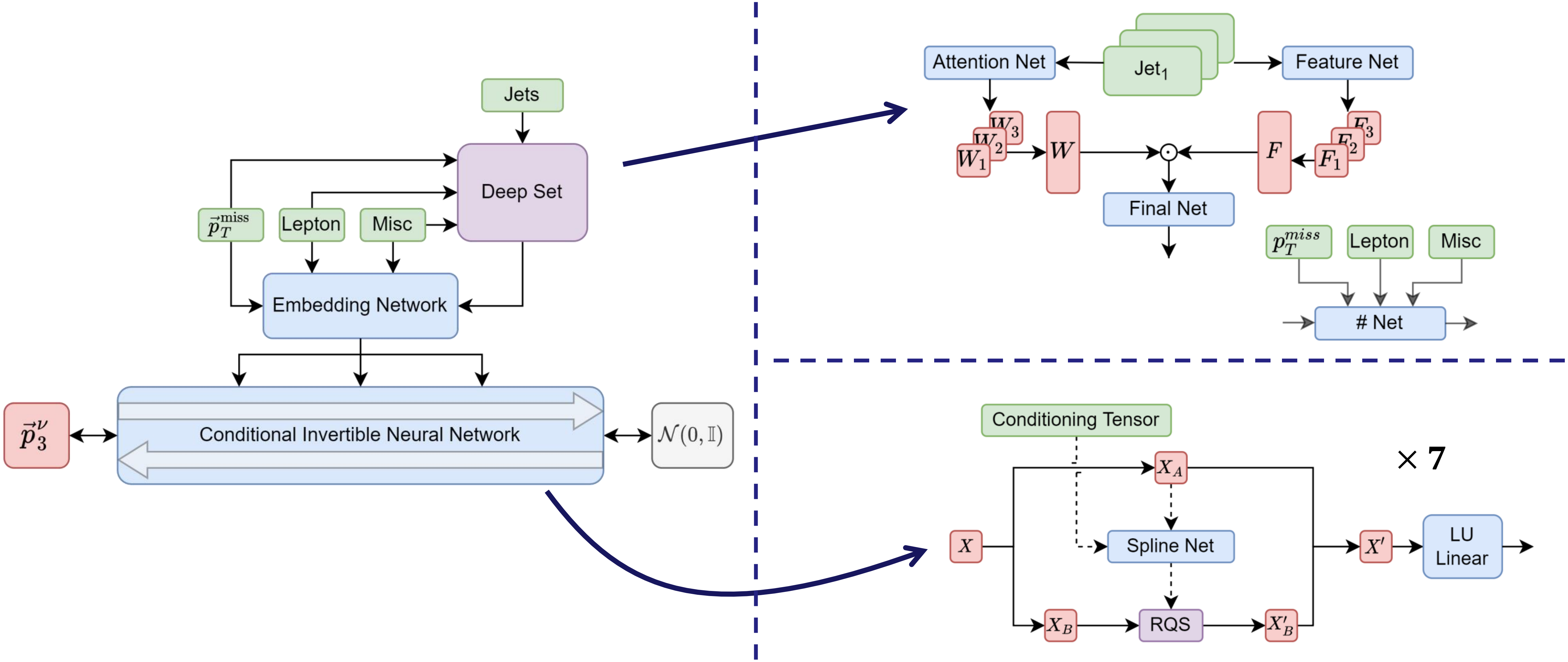
- Taking $\boldsymbol{p_X(x)}$ to be the complex distribution over **our data**
  - Can perform exact density estimation: $p_X(x) = p_Z\big(f(x)\big)|\det(J(x))|$
  - Can generate new data by sampling p_X: Sample $p_Z(\cdot)$, compute $f^{-1}(z)$

INN layers need to be invertible and have a Jacobian that is easy to calculate

- In practice we use an **invertible neural network (INN)** to parameterise $\boldsymbol{f_\theta}$
- Usually train with INNs for generation using **maximum likelihood objective** for observed data:

$$Loss(x) = -\log(p_X(x)) = -\log(p_Z\big(f_\theta(x)\big)) - \log(|\det\big(J(x)\big))$$

ν-Flow Structure



Network Hyperparamers

DeepSet:Feature Network

- 5(11)->Linear->LeakyReLU->LayerNorm->64
- 64->Linear->LeakyReLU->LayerNorm->64
- 64->Linear->8

DeepSet: Attention Network

- 5(11)->Linear->LeakyReLU->LayerNorm->32
- 32->Linear->LeakyReLU->LayerNorm->32
- 32->Linear->1

DeepSet: Final Network

- 8(11)->Linear->LeakyReLU->LayerNorm->64
- 64->Linear->LeakyReLU->LayerNorm->64
- 64->Linear->8

Each row is a layer showing:
Inputs(conditional inputs) -> operations -> outputs(residual)

Embedding Network

- 19->Linear->LeakyReLU->LayerNorm->64
- 64->Linear->LeakyReLU->LayerNorm->64(add)
- 64->Linear->32

Spline Network

- 2(32)->Linear->LeakyReLU->LayerNorm->64
- 64(32)->Linear->LeakyReLU->LayerNorm->64(add)
- 64->Linear->LeakyReLU->LayerNorm->64(add)
- 64->Linear->29

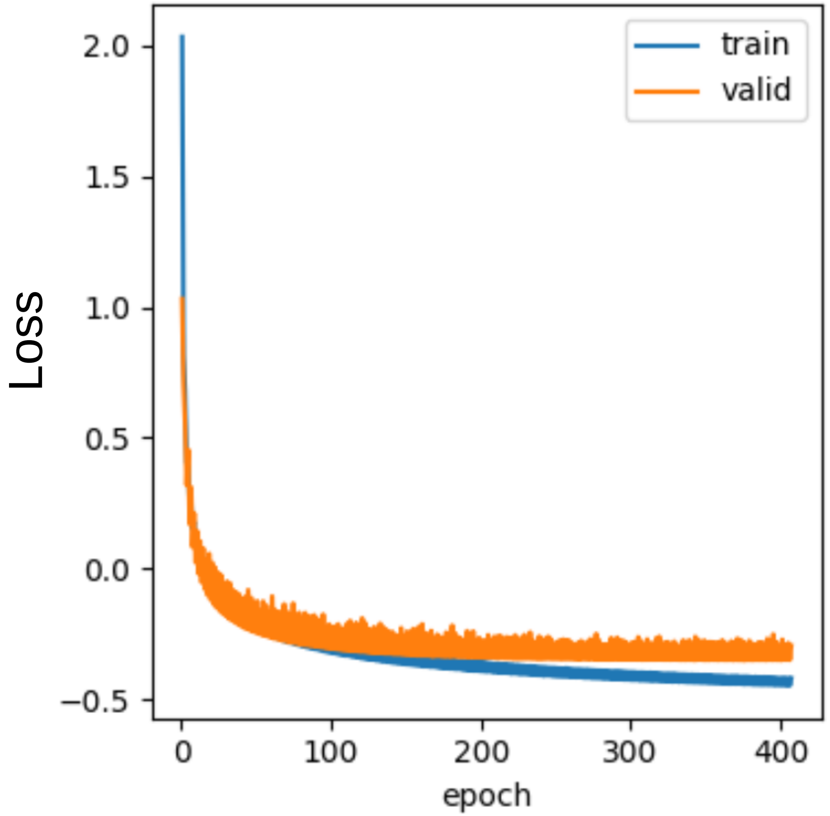
Network Training

Training was done using the **negative log likelihood** as the loss function using the **Adam optimizer** with **early stopping** performed on a **10% holdout** validation set.

- Training set size: 528921
- Validation set size: 58768

Other training parameters:

- Batch size = 256
- Gradient norm clipping = 5
- Early stopping patience = 30



The learning rate followed a **cyclic asymmetrical cosine schedule** with a period of two epochs. Each cycle the learning rate would be ramped up from 0 to 5e-4 and then back down to 0. The fraction of the cycle used for warmup(cooldown) was set to 0.3(0.7).

X^2: Masses

- Looking at the invariant mass of $bl\nu$ using the $\boldsymbol{b}$ **selected by the** $\boldsymbol{X^2}$ **fit**
  - **Idealised** refers to the truth neutrino and the correct b_{lep}
  - Shaded regions are subset of events where X^2 yielded correct b_{lep}