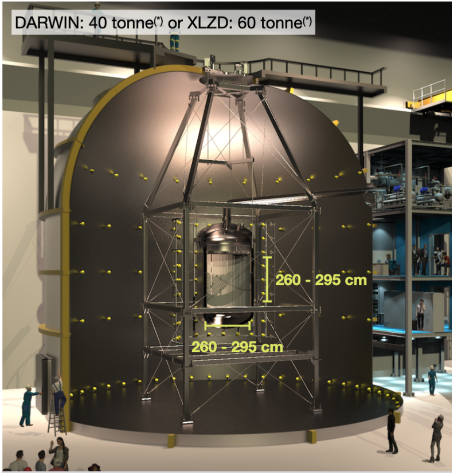
# Anomaly-aware machine learning for model-independent searches of new physics in DARWIN

Andre Scaffidi and Roberto Trotta for the DARWIN collaboration.

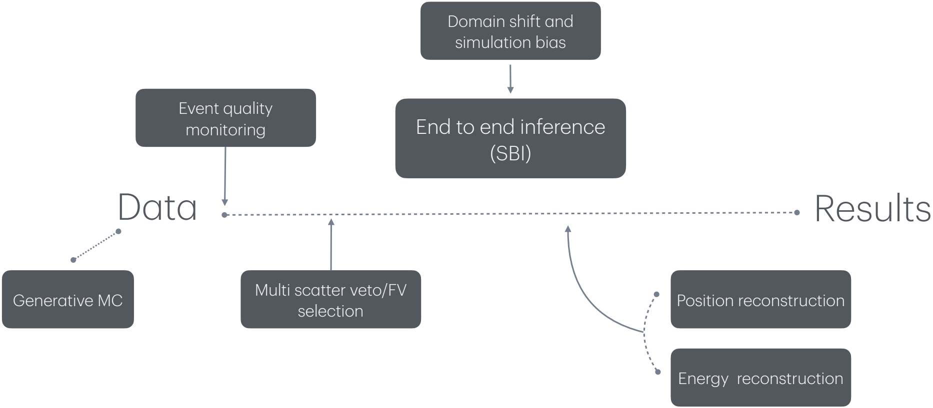
June 26, 2024

# DARWIN collaboration: Proposal

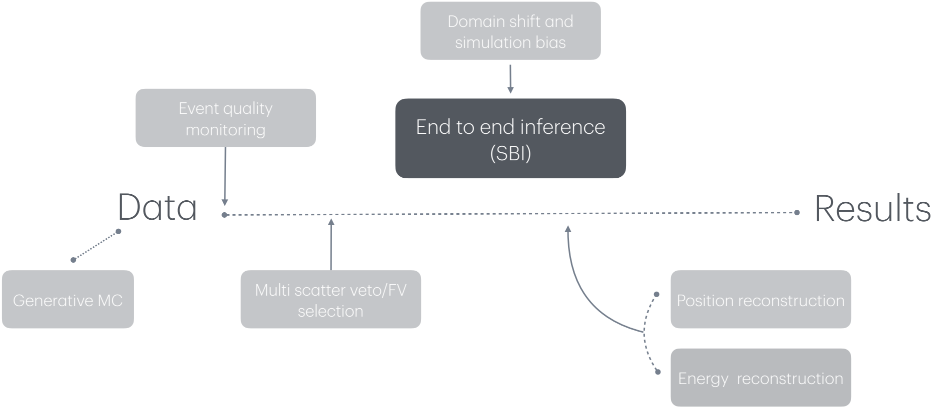
$\sim 200$ members



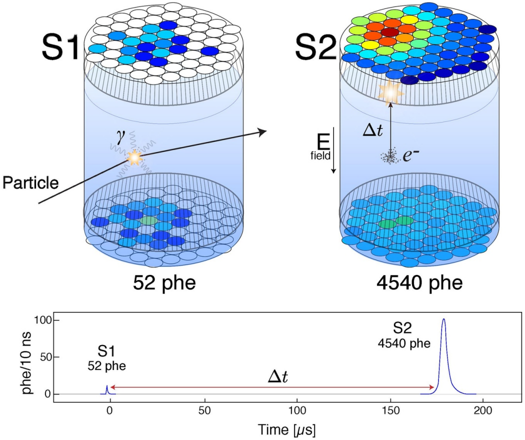
# Current/Future ML scope @ DARWIN



# Current/Future ML scope @ DARWIN



# Underground TPCs: Events

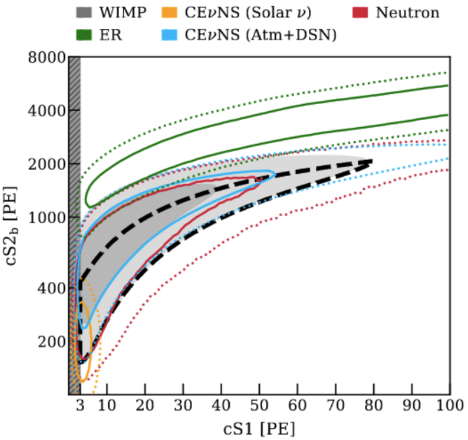



- Traditionally use high level analysis observables: cS1, cS2

# Traditional likelihood-based analysis

$$\log \mathcal{L}(\mathbf{cS1}, \mathbf{cS2} \,|\, \sigma_{\mathrm{SI}}, \boldsymbol{\theta}) = \log \mathcal{L}_{\mathrm{science}}(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}}, \boldsymbol{\theta}) + \log \mathcal{L}_{\mathrm{ancillary}}(\boldsymbol{\theta}) \ ,$$

- Parametrically model dependent
- Derived from 2D templates
- Relies on high-level 'summary statistics' cS1,cS2: $\Rightarrow \mathbf{E} = \mathbf{g}(\mathbf{cS1}, \mathbf{cS2})$

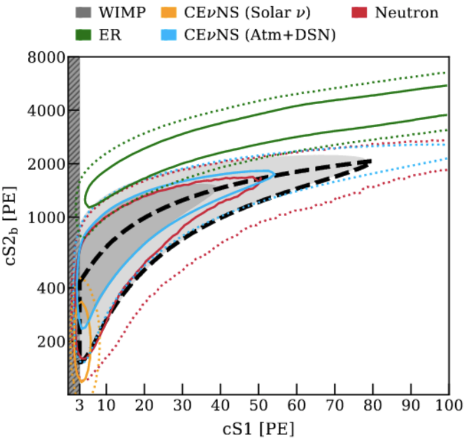



Does this likelihood yield an optimal test statistic?

# Traditional likelihood-based analysis

$$\log \mathcal{L}(\mathbf{cS1}, \mathbf{cS2} \,|\, \sigma_{\mathrm{SI}}, \boldsymbol{\theta}) = \log \mathcal{L}_{\mathrm{science}}(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}}, \boldsymbol{\theta}) + \log \mathcal{L}_{\mathrm{ancillary}}(\boldsymbol{\theta}) \ ,$$

- Parametrically model dependent
- Derived from 2D templates
- Relies on high-level 'summary statistics' cS1,cS2: $\Rightarrow \mathbf{E} = \mathbf{g}(\mathbf{cS1}, \mathbf{cS2})$

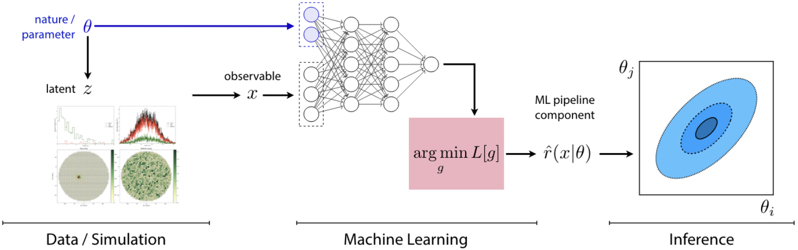



Does this likelihood yield an optimal test statistic?

# Simulation based inference (SBI)

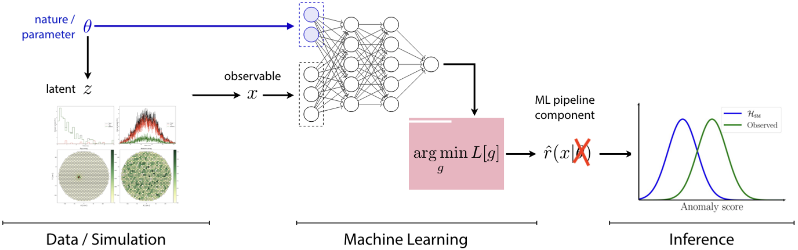
# Simulation-Based Inference (in a nutshell)

Simulation-based inference is a statistical technique that allows us to make inferences about a population or process based on simulated data.



# Simulation-Based Inference (in a nutshell)

Simulation-based inference is a statistical technique that allows us to make inferences about a population or process based on simulated data.



# Benefits of SBI

- Can handle complex models with intractable likelihoods.
- Use deep neural nets to learn underlying features of simulated data/summary stats.
- Once a simulator has been established, possible to include arbitrarily complicated simulations into analysis: prompt readouts $\rightarrow$ high level summary stats.
- Need no special treatment of nuisance parameters.
  - Can in principle simulate/calibrate any detector effects and learn them directly.

# Benefits of SBI

- Can handle complex models with intractable likelihoods.
- Use deep neural nets to learn underlying features of simulated data/summary stats.
- Once a simulator has been established, possible to include arbitrarily complicated simulations into analysis: prompt readouts $\rightarrow$ high level summary stats.
- Need no special treatment of nuisance parameters.
  - Can in principle simulate/calibrate any detector effects and learn them directly.

# Benefits of SBI

- Can handle complex models with intractable likelihoods.
- Use deep neural nets to learn underlying features of simulated data/summary stats.
- Once a simulator has been established, possible to include arbitrarily complicated simulations into analysis: prompt readouts $\rightarrow$ high level summary stats.
- Need no special treatment of nuisance parameters.
    - Can in principle simulate/calibrate any detector effects and learn them directly.

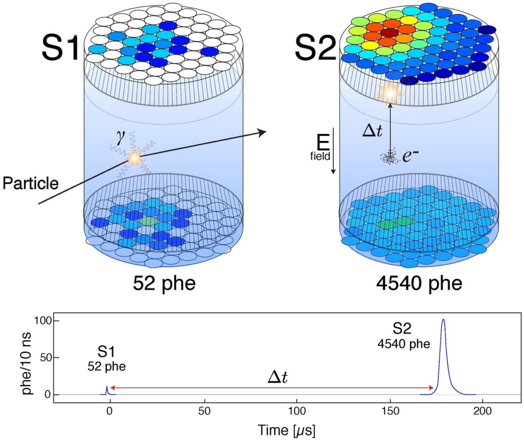
# Benefits of SBI

- Can handle complex models with intractable likelihoods.
- Use deep neural nets to learn underlying features of simulated data/summary stats.
- Once a simulator has been established, possible to include arbitrarily complicated simulations into analysis: prompt readouts $\rightarrow$ high level summary stats.
- Need no special treatment of nuisance parameters.
  - Can in principle simulate/calibrate any detector effects and learn them directly.

## Benefits of SBI

- Can handle complex models with intractable likelihoods.
- Use deep neural nets to learn underlying features of simulated data/summary stats.
- Once a simulator has been established, possible to include arbitrarily complicated simulations into analysis: prompt readouts $\rightarrow$ high level summary stats.
- Need no special treatment of nuisance parameters.
    - Can in principle simulate/calibrate any detector effects and learn them directly.

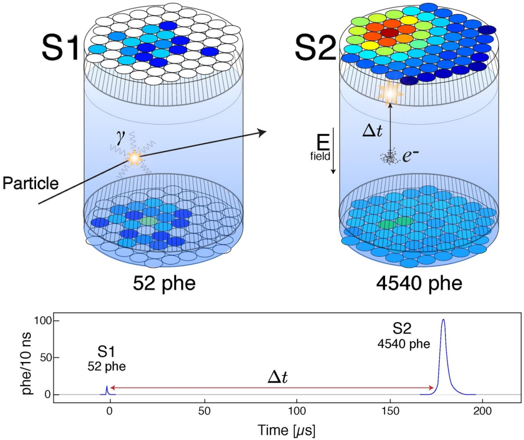
# Underground TPCs: Two types of events




- Nuclear Recoil (NR) → WIMPs
- (Dominant) Background → Electron Recoil (ER).

- Distance and ratio between S1/S2 peaks → NR vs. ER.

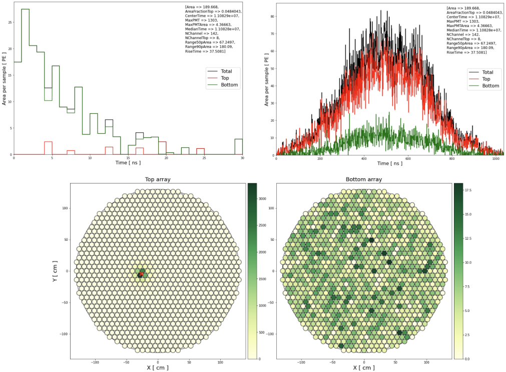
# Underground TPCs: Two types of events




- Nuclear Recoil (NR) → WIMPs
- (Dominant) Background → Electron Recoil (ER).
- Distance and ratio between S1/S2 peaks → NR vs. ER.

# Training data: Simulations

Event event output S1, S2 pulses and PMT deposits (4-fold coincidence, 200 ns, 200 V/cm):

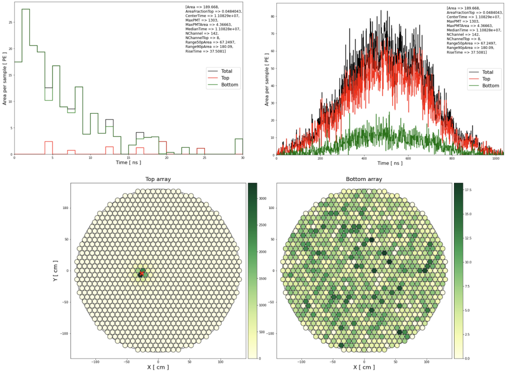



$\Longrightarrow$ $\mathbf{x} = [\texttt{S1WaveformTotal}, \texttt{S2WaveformTotal}, \texttt{S2Pattern}\ ]$

ER/NR. Generate data representatively $\in [1-100]$ keV.

# Training data: Simulations

Event event output S1, S2 pulses and PMT deposits (4-fold coincidence, 200 ns, 200 V/cm):

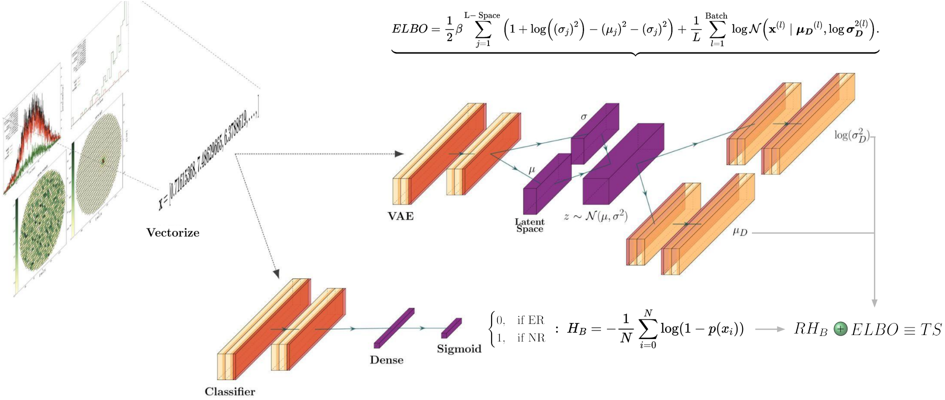



$$\Longrightarrow \mathbf{x} = [\texttt{S1WaveformTotal}, \texttt{S2WaveformTotal}, \texttt{S2Pattern}\ ]$$

ER/NR. Generate data representatively $\in [1 - 100]$ keV.

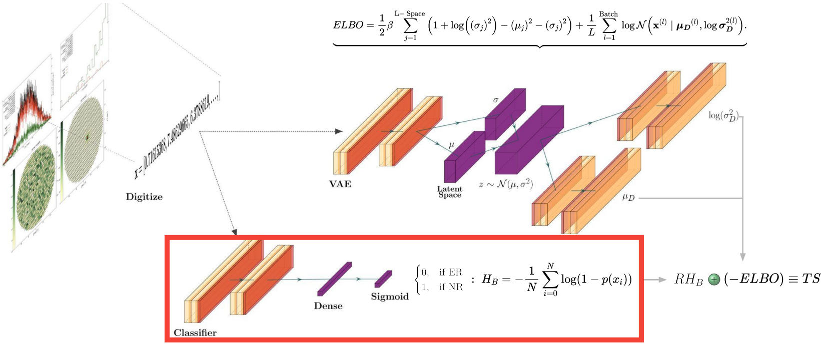
# Proposed analysis pipeline: The neural anomaly detector

# Semi-unsupervised anomaly detection: Full pipeline




- (Top) Variational auto-encoder: Train on ER only
- (Bottom) Fully connected MLP classifier: ER vs NR

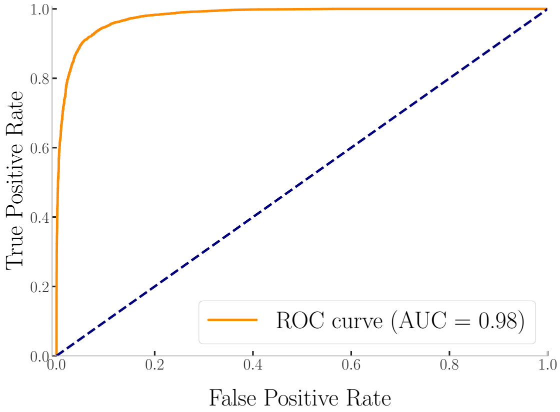
# Semi-unsupervised anomaly detection: Classifier




- (Bottom) Fully connected NN classifier: ER vs NR
- Classifies two interaction types: ER/NR
  - New insights from Lopez-Fogliani et.al 2406.10372: BDT's MLP and transformers all basically just as good...!

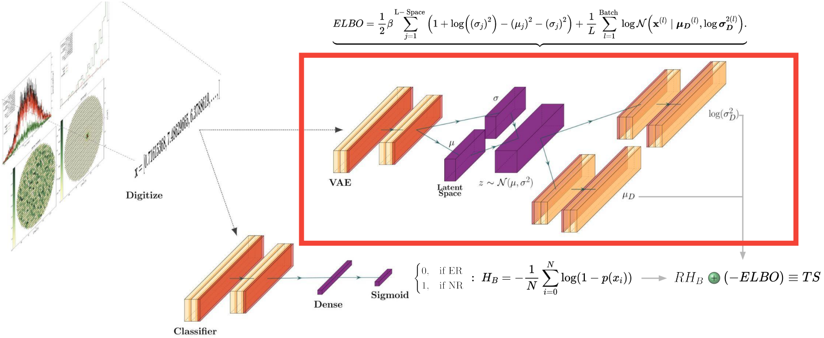
# Classification: ER vs. NR Results

- Train on $\sim$ 40000 events. Take testing sub-sample of $\sim$ 40%
- Check performance $\rightarrow$ ROC:



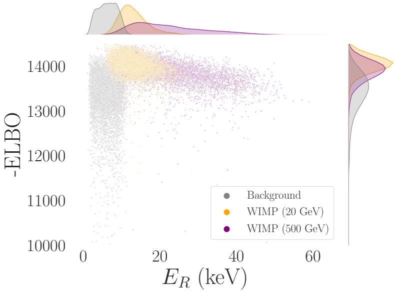

- Takeaway $\Rightarrow$ **98.03% accuracy**. (Recall = 98.07%, Precision = 96.39%)

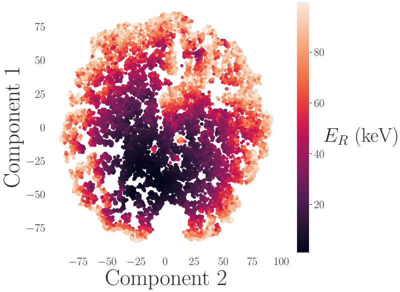
# Semi-unsupervised anomaly detection: VAE




- (Top) Variational auto-encoder: Train on ER only
- Learns low dimensional representation of events ⇒ **energy**.

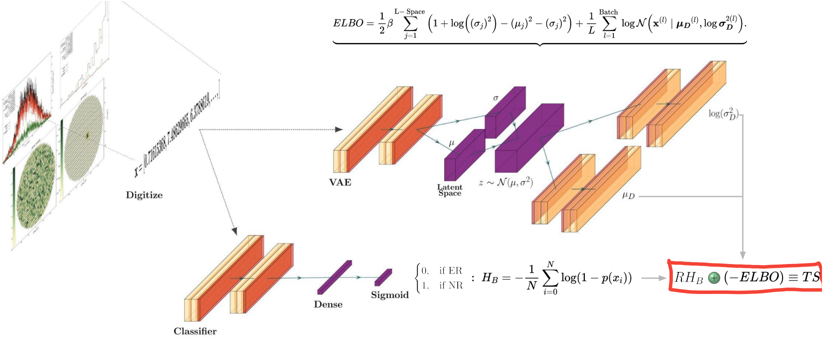
# Spectral information is encoded in the VAE




- Even though trained on just ER: Auto-encoder can learn underlying spectral information of all events!
- Energy reconstruction using neural posterior estimation (Preliminary results promising - backup slides.)

# Semi-unsupervised anomaly detection: Anomaly function




- **Anomaly function** TS constructed as sum of outputs

## Semi-unsupervised anomaly detection: New distance metric

- New 'anomaly function' that utilizes pre-trained supervised NN classifier:

$$TS = -ELBO + R\,H_B\ ,$$

where

- $H_B = -\frac{1}{N}\sum_{i=0}^{N}\log\left(1-p\left(x_i\right)\right)$ (Binary cross-sentropy.)
- $R$ scales the contribution of the cross-entropy term $\rightarrow$ makes it more/less supervised.

Deriving anomaly scores is a game...

# Semi-unsupervised anomaly detection: New distance metric

- New 'anomaly function' that utilizes pre-trained supervised NN classifier:

$$TS = -ELBO + R\,H_B\ ,$$

where

- $H_B = -\frac{1}{N}\sum_{i=0}^{N} \log\left(1 - p\left(x_i\right)\right)$ (Binary cross-sentropy.)
- $R$ scales the contribution of the cross-entropy term $\rightarrow$ makes it more/less supervised.

Deriving anomaly scores is a game...

# Semi-unsupervised anomaly detection: New distance metric

- New 'anomaly function' that utilizes pre-trained supervised NN classifier:

$$TS = -ELBO + R\,H_B\ ,$$

where

- $H_B = -\frac{1}{N}\sum_{i=0}^{N}\log\left(1 - p\left(x_i\right)\right)$ (Binary cross-sentropy.)
- $R$ scales the contribution of the cross-entropy term $\rightarrow$ makes it more/less supervised.

Deriving anomaly scores is a game...

# Semi-unsupervised anomaly detection: New distance metric

- New ‘anomaly function’ that utilizes pre-trained supervised NN classifier:

$$TS = -ELBO + R\,H_B\ ,$$

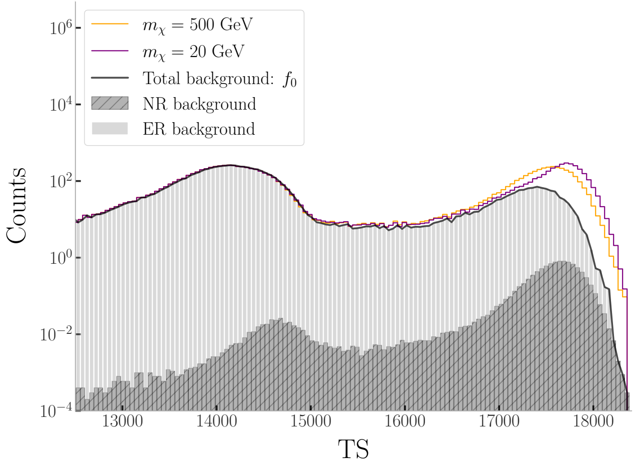
where

- $H_B = -\frac{1}{N}\sum_{i=0}^{N} \log\left(1 - p\left(x_i\right)\right)$ (Binary cross-sentropy.)
- $R$ scales the contribution of the cross-entropy term $\rightarrow$ makes it more/less supervised.

Deriving anomaly scores is a game...

# Pseudo-data sets

- **Anomaly Detection**: Once trained, run data the network has never seen before through trained network. $\Rightarrow$ Extract $f_0$ (null pdf).



# Dimensionally reduced two sample hypothesis test

# Dimensionally reduced analysis

- $\Rightarrow$ 1D analysis in $TS$ space: Accept/reject $\mathcal{H}_0 : X \sim \mathcal{P}(x \mid \text{No signal})$.

$$\mathcal{L}(\mathbf{TS}|\mathcal{H}_0) \propto e^{-B} \prod_{i=1}^{N} \left(B f_0\left(TS_i\right)\right)$$

- Unbinned.
- Parametrically independent on WIMP model.
- No auxiliary terms required assuming simulations have suitably descriptive coverage.
- Can be augmented with more fundamental data representation or calibration. (Current work!)

# Compare with likelihood approach

$$\log \mathcal{L}(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}}, \boldsymbol{\theta}) = \overbrace{\log \mathcal{L}_{\mathrm{science}}(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}}, \boldsymbol{\theta})}^{\text{Background and signal pdfs}/\mu_i} + \overbrace{\log \mathcal{L}_{\mathrm{ancillary}}(\boldsymbol{\theta})}^{\text{Nuisance params.}}$$

- Exact anomaly detection analogue (model indep.):

$$\mathcal{L}\left(\mathbf{cS1}, \mathbf{cS2} \mid \mathcal{H}_0\right) \equiv \mathcal{L}\left(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}} = 0\right)$$

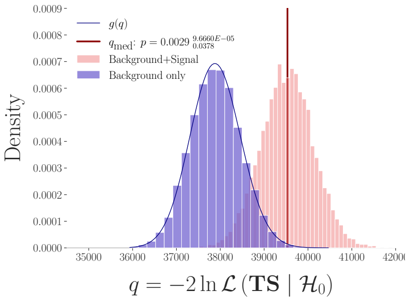
# Compare with likelihood approach

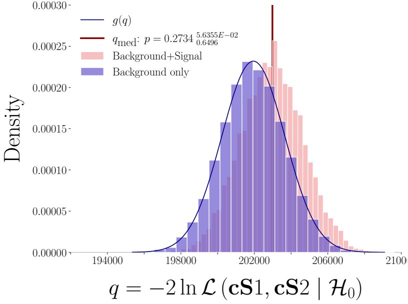
$$\log \mathcal{L}(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}}, \boldsymbol{\theta}) = \overbrace{\log \mathcal{L}_{\mathrm{science}}(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}}, \boldsymbol{\theta})}^{\text{Background and signal pdfs}/\mu_i} + \overbrace{\log \mathcal{L}_{\mathrm{ancillary}}(\boldsymbol{\theta})}^{\text{Nuisance params.}}$$

- Exact anomaly detection analogue (model indep.):

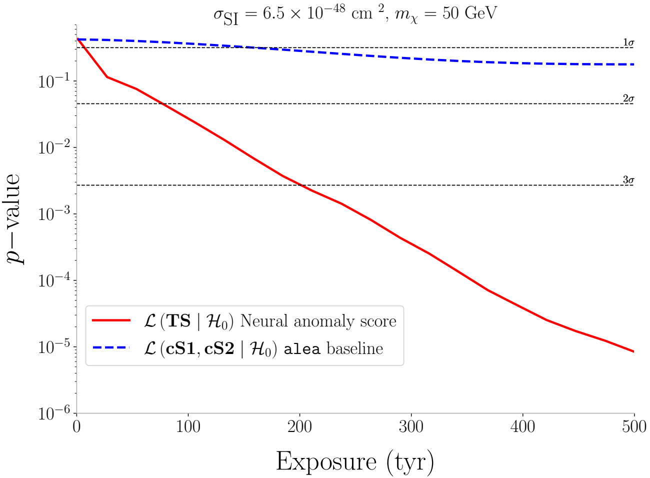
$$\mathcal{L}\left(\mathbf{cS1}, \mathbf{cS2} \mid \mathcal{H}_0\right) \equiv \mathcal{L}\left(\mathbf{cS1}, \mathbf{cS2} \mid \sigma_{\mathrm{SI}} = 0\right)$$

# DM median sensitivity from toys

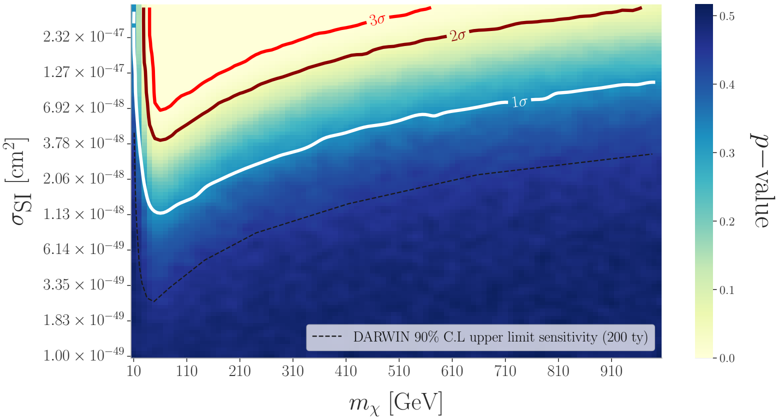


# $\mathcal{H}_0$ rejection in presence of WIMP signal injection



# $\mathcal{H}_0$ rejection in presence of WIMP signal injection



# Current and future work

**Calibration/simulation mismatch**:

- Adversarial DA
- Useful conversations with Omar Alterkait (equivariant NN talk)

**More fundamental data**:

- Time domain in PMT channels: Transformers? Other?
  Extremely high $D$

Implementation and inclusion of energy/position reconstruction and neutron veto into full pipeline.

# Thank you!

# Backup Slides

# Interplay of unsupervised and supervised components

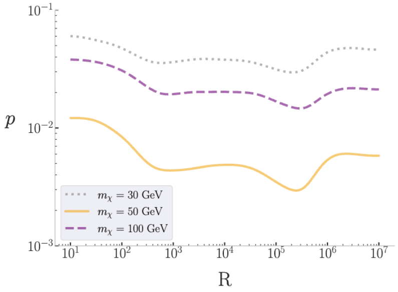
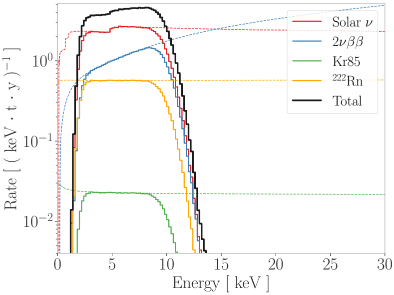
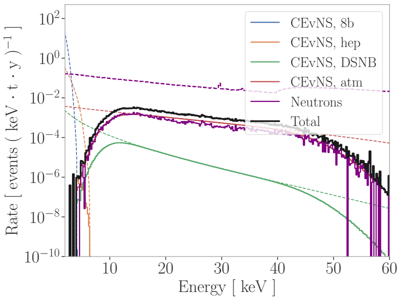



Figure 8: Optimisation of the hyperparameter $R$ that controls the contribution of the supervised classifier in the determination of the anomaly function $TS$ as shown in Eqn. 5. The $p$ =value to reject $\mathcal{H}_0$ is given as a function of $R$ for three benchmark WIMP sensitivity studies at fixed exposure of 200 ty and cross section $\sigma_{\mathrm{SI}} = 6.5 \times 10^{-48}$ cm$^2$ for $m_\chi = 30, 50$ and 100 GeV. We have checked that the scattering cross-section rescales the median sensitivity probability but does not affect the shape of the above curves, and therefore the choice of $R$ and cut value are insensitive to it. An optimal combination of $R$ value is obtained when the probability to accept $\mathcal{H}_0$ is smallest (most sensitivity). For this study we adopt an $R$ value of $2.5 \times 10^5$.

# Backgrounds

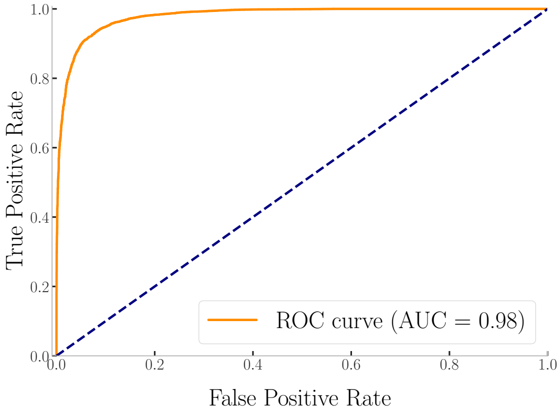



- FV (baseline) 31.5t
- Trigger N4T200
- Single scatter selection (Neutrons)
- CES 2-10 keV: For now cheating a bit...
- Todo: Accidentals

# Classification: ER vs. NR Results

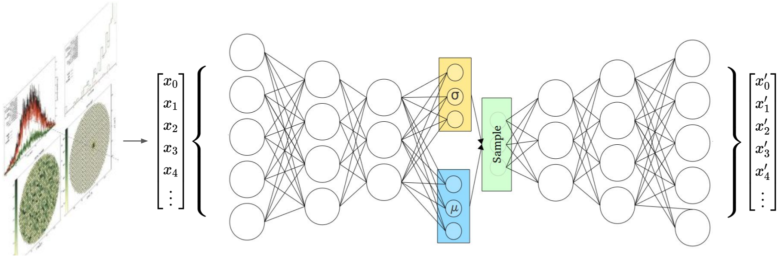
- Train on $\sim$ 40000 events. Take testing sub-sample of $\sim$ 40%
- Check performance $\rightarrow$ ROC:




- Takeaway $\Rightarrow$ **98.03% accuracy**. (Recall = 98.07%, Precision = 96.39%)

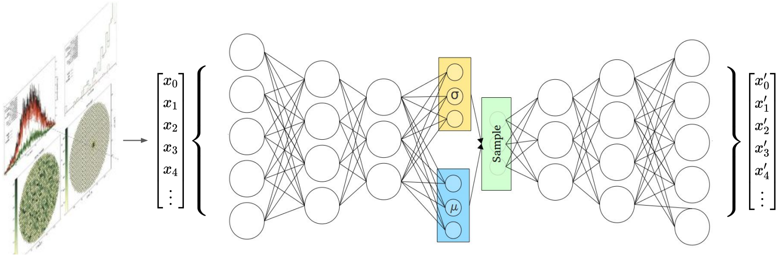
# Generative Deep Learning: The Variational Auto-Encoder

- Variety of studies in HEP use these for anomaly detection tasks.
- Goal: Learn low dimensional representation (encoding) of data via dimensional reduction.
- Latent space (bottleneck) layer is a bunch of normal distributions parameterized by some $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.
- Our goal: Learn the latent representation of the background (ER) events. $\Rightarrow$ Spectral information ($E$).



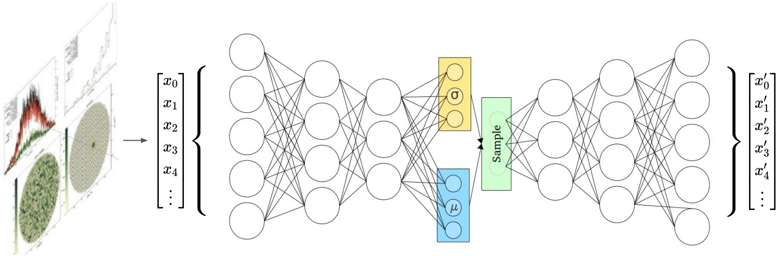
# Generative Deep Learning: The Variational Auto-Encoder

- Variety of studies in HEP use these for anomaly detection tasks.
- Goal: Learn low dimensional representation (encoding) of data via dimensional reduction.
- Latent space (bottleneck) layer is a bunch of normal distributions parameterized by some $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.
- Our goal: Learn the latent representation of the background (ER) events. $\Rightarrow$ Spectral information ($E$).



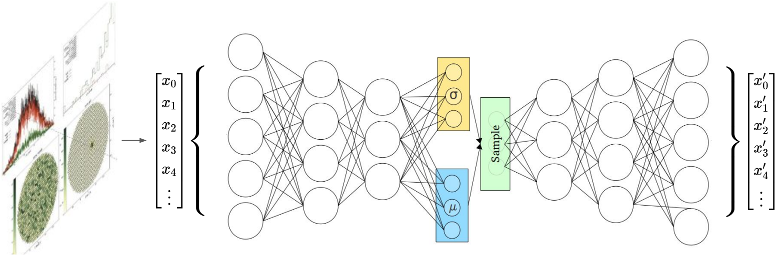
# Generative Deep Learning: The Variational Auto-Encoder

- Variety of studies in HEP use these for anomaly detection tasks.
- Goal: Learn low dimensional representation (encoding) of data via dimensional reduction.
- Latent space (bottleneck) layer is a bunch of normal distributions parameterized by some $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.
- Our goal: Learn the latent representation of the background (ER) events. $\Rightarrow$ Spectral information ($E$).



# Generative Deep Learning: The Variational Auto-Encoder

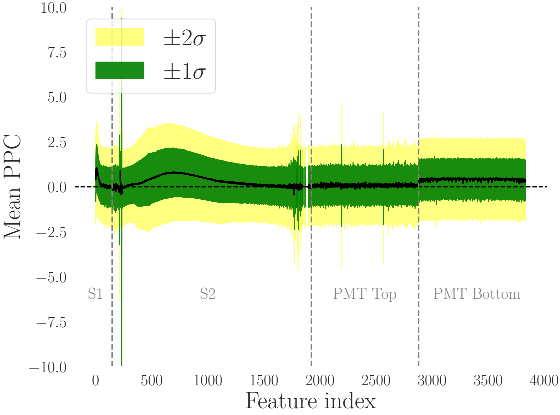
- Variety of studies in HEP use these for anomaly detection tasks.
- Goal: Learn low dimensional representation (encoding) of data via dimensional reduction.
- Latent space (bottleneck) layer is a bunch of normal distributions parameterized by some $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.
- Our goal: Learn the latent representation of the background (ER) events. $\Rightarrow$ Spectral information ($E$).



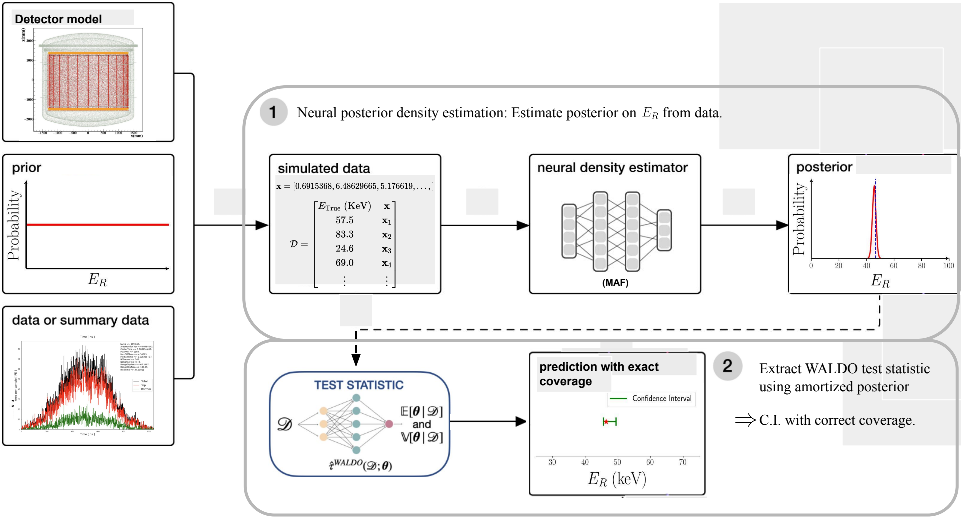
# Generative validation

- Posterior predictive check

$$\text{Mean PPC}_{\,j} = \frac{1}{\sigma^j_{\text{test}}} \frac{1}{N} \sum_{i=1}^{N} \left( x^j_{i_{\text{sample}}} - x^j_{i_{\text{test}}} \right) \; ,$$



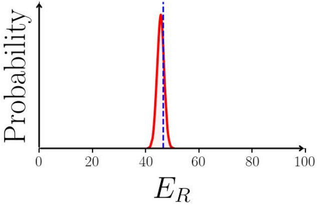
# Energy reconstruction SBI with masked autoregressive flows



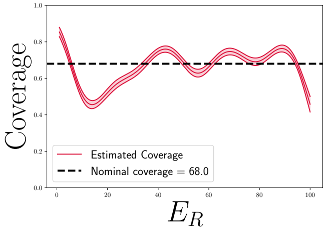

$$\tau^{\text{WALDO}}(\mathcal{D}; \boldsymbol{\theta}_0) = \left(\mathbb{E}[\boldsymbol{\theta} \mid \mathcal{D}] - \boldsymbol{\theta}_0\right)^T \mathbb{V}[\boldsymbol{\theta} \mid \mathcal{D}]^{-1} \left(\mathbb{E}[\boldsymbol{\theta} \mid \mathcal{D}] - \boldsymbol{\theta}_0\right)$$

# Follow up work: $E$ reconstruction

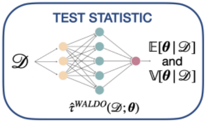
Neural posterior density estimation (Masked auto-regressive flows)



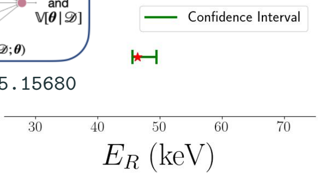

→



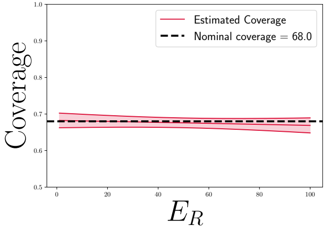

Neural posterior density estimation + WALDO




arXiv:2205.15680




→



# Variational-Auto-Encoder: Training

- Use same data as with supervised classification.
- Train VAE on just* ER data.
- Train by maximising evidence lower bound (ELBO):

$$\log p(x) \geq \text{ELBO} = \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z \mid x)}\right]$$

$$= E[\log p(x|z)] - \beta D_{KL}(q(z|x)||p(z))$$

$$x = \text{Input}$$

$$z = \text{Latent vector}$$

$$\beta = \text{Regularization parameter}$$

- **Loss** $\equiv -\text{ELBO}$

# Variational-Auto-Encoder: Training

- Use same data as with supervised classification.
- Train VAE on just* ER data.
- Train by maximising evidence lower bound (ELBO):

$$\log p(x) \geq \text{ELBO} = \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z \mid x)}\right]$$

$$= E[\log p(x|z)] - \beta D_{KL}(q(z|x)||p(z))$$

$$x = \text{Input}$$

$$z = \text{Latent vector}$$

$$\beta = \text{Regularization parameter}$$

- **Loss** $\equiv -\text{ELBO}$

# Variational-Auto-Encoder: Training

- Use same data as with supervised classification.
- Train VAE on just* ER data.
- Train by maximising evidence lower bound (ELBO):

$$\log p(x) \geq \text{ELBO} = \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z \mid x)}\right]$$

$$= E[\log p(x|z)] - \beta D_{KL}(q(z|x)||p(z))$$

$$x = \text{Input}$$

$$z = \text{Latent vector}$$

$$\beta = \text{Regularization parameter}$$

- **Loss** $\equiv -\text{ELBO}$

# Variational-Auto-Encoder: Training

- Use same data as with supervised classification.
- Train VAE on just* ER data.
- Train by maximising evidence lower bound (ELBO):

$$\begin{aligned}
\log p(x) \geq \text{ELBO} &= \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z \mid x)}\right] \\
&= E[\log p(x|z)] - \beta D_{KL}(q(z|x)||p(z)) \\
x &= \text{Input} \\
z &= \text{Latent vector} \\
\beta &= \text{Regularization parameter}
\end{aligned}$$

- **Loss** $\equiv -\text{ELBO}$

# tSNE of latent space