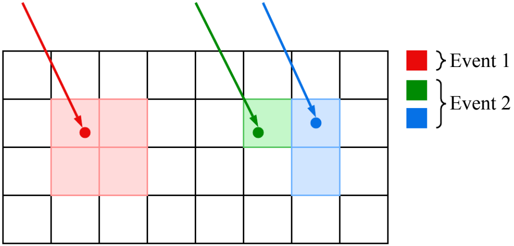
New approach to the clustering problem in pixel detectors, in a framework of LUXE experiment

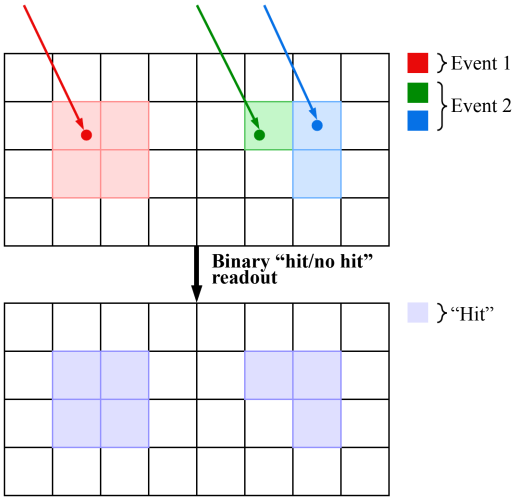
Roman Urmanov

01.05.2023

LUXE experiment: Clustering

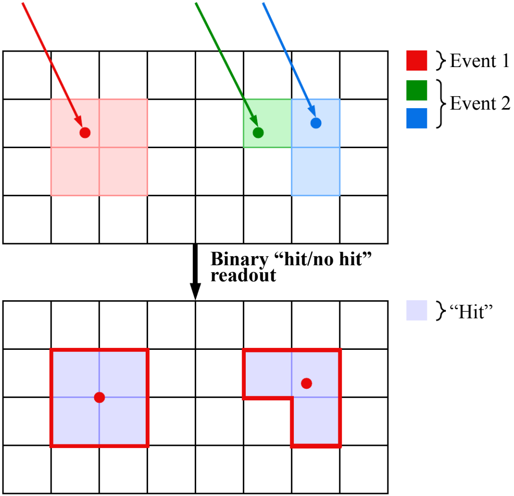


LUXE experiment: Clustering



- Binary output – no information on number of particles or position

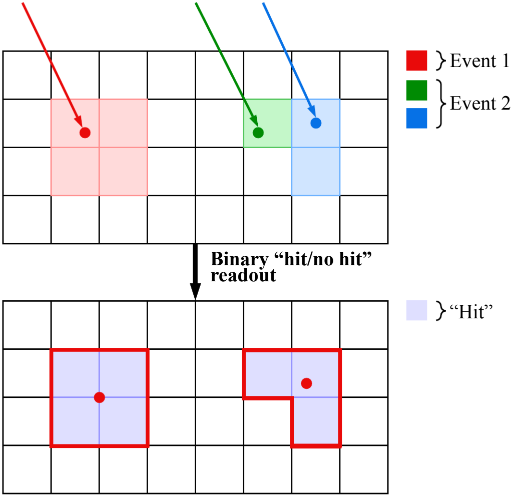
LUXE experiment: Clustering



"Pac-Man": adjacent hits combined into clusters. Cluster's "center of mass" is taken for track inference

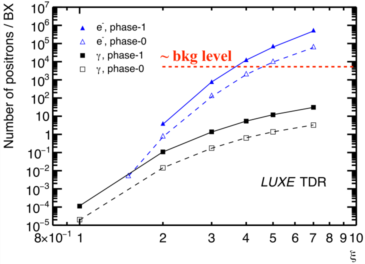
- Binary output – no information on number of particles or position

LUXE experiment: Clustering

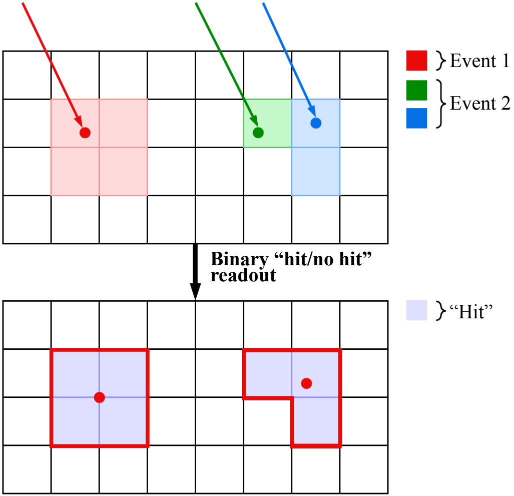


"Pac-Man": adjacent hits combined into clusters. Cluster's "center of mass" is taken for track inference

- Binary output – no information on number of particles or position
- LUXE will work in a wide range of pixel occupancies

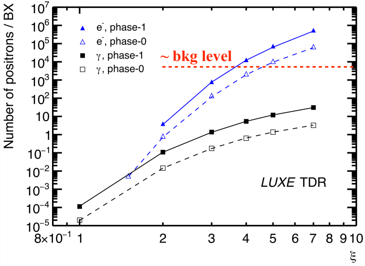


LUXE experiment: Clustering

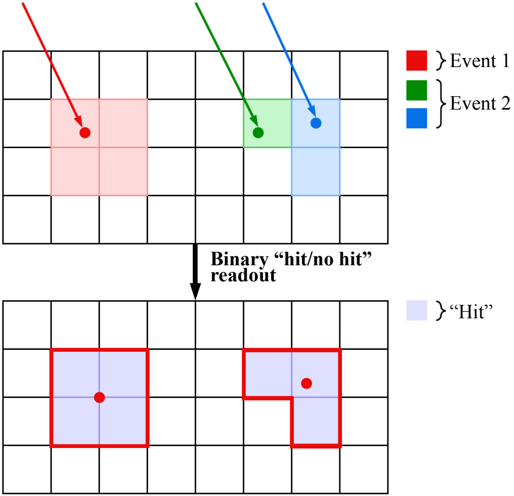


"Pac-Man": adjacent hits combined into clusters. Cluster's "center of mass" is taken for track inference

- Binary output – no information on number of particles or position
- LUXE will work in a wide range of pixel occupancies
- "Pac-Man" – up to $\lesssim 10^4$ e per BX
- Up to 3×10^5 e per BX in later stages



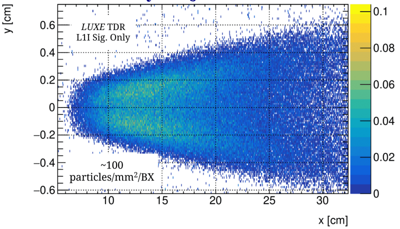
LUXE experiment: Clustering



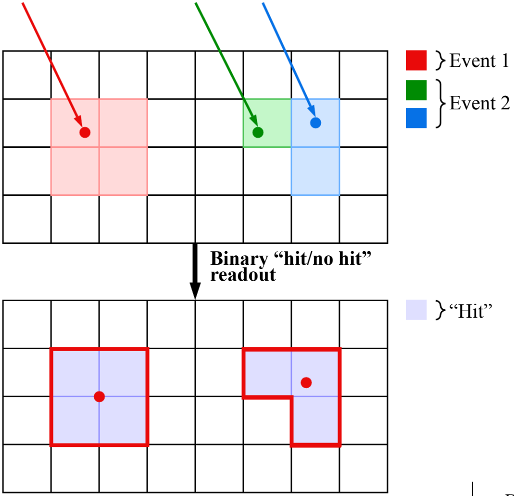
"Pac-Man": adjacent hits combined into clusters. Cluster's "center of mass" is taken for track inference

- Binary output – no information on number of particles or position
- LUXE will work in a wide range of pixel occupancies
- "Pac-Man" – up to $\lesssim 10^4$ e per BX
- Up to 3×10^5 e per BX in later stages

Hit density at ξ=7



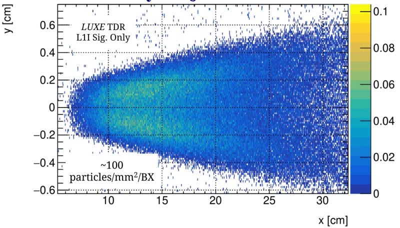
LUXE experiment: Clustering



“Pac-Man”: adjacent hits combined into clusters. Cluster’s “center of mass” is taken for track inference

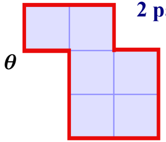
- Binary output – no information on number of particles or position
- LUXE will work in a wide range of pixel occupancies
- “Pac-Man” – up to $\lesssim 10^4$ e per BX
- Up to 3×10^5 e per BX in later stages

Hit density at ξ=7



Detector reference	Hit density [mm^{-2}]		
	MCD	ATLAS ITk	ALICE ITS3
Pixel Layer 0	3.68	0.643	0.85
Pixel Layer 1	0.51	0.022	0.51

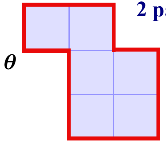
Modeling the hits



2 particles – 4 correlated random variables

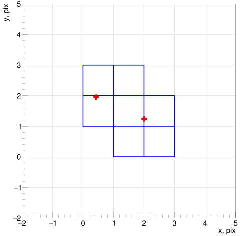
$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

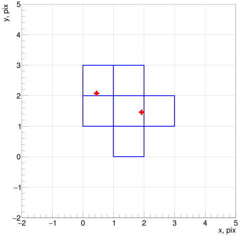
Modeling the hits



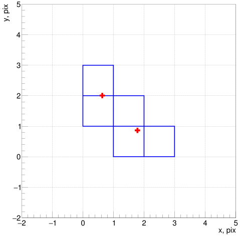
2 particles – 4 correlated random variables

- Strong correlations between the y-components are observed (order of variance, $> 3\sigma$ from zero)

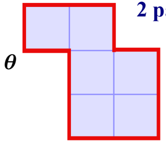




$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

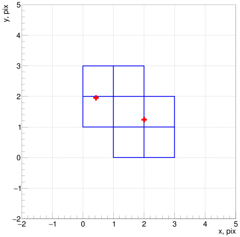


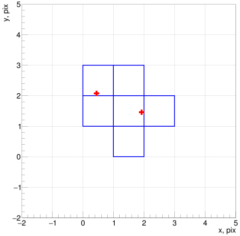
Modeling the hits



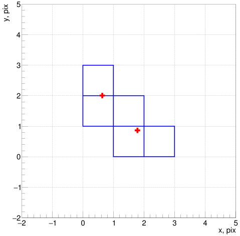
2 particles – 4 correlated random variables

- Strong correlations between the y-components are observed (order of variance, $> 3\sigma$ from zero)
- x correlations of $> 2\sigma$ deviation from zero are also observed

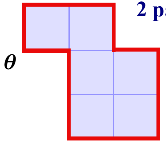




$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$



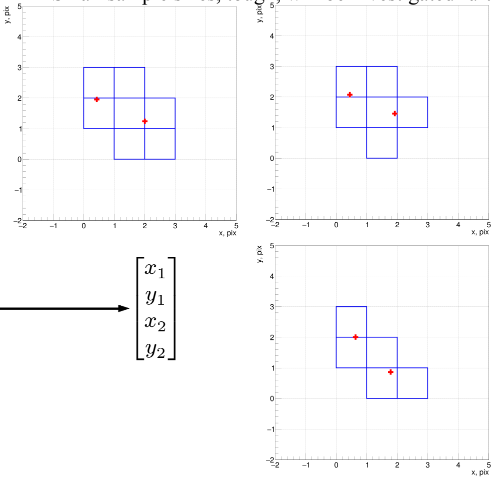
Modeling the hits



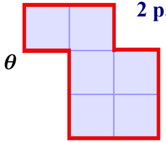
2 particles – 4 correlated random variables

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

- Strong correlations between the y-components are observed (order of variance, $> 3\sigma$ from zero)
- x correlations of $> 2\sigma$ deviation from zero are also observed
- Small sample sizes, tough, will be investigated further



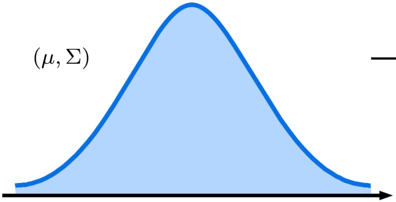
Modeling the hits



2 particles – 4 correlated random variables

- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$

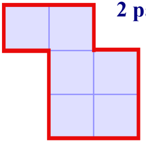
(μ, Σ)



$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

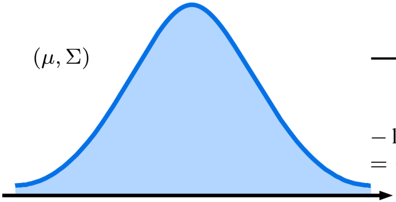
Modeling the hits

$\boldsymbol{\theta}$



2 particles – 4 correlated random variables

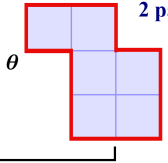
- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$



$$\longrightarrow \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

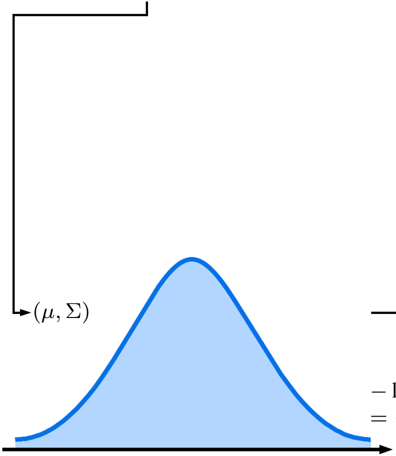
$$-\log \mathcal{L}(\mu, \Sigma) =$$
$$= -\log \mathbb{P}(\mu, \Sigma) \to \min$$

Modeling the hits



2 particles – 4 correlated random variables

- Step 1: Multivariate Gaussian model – $\mathcal{N}\left(\mu, \Sigma\right)$

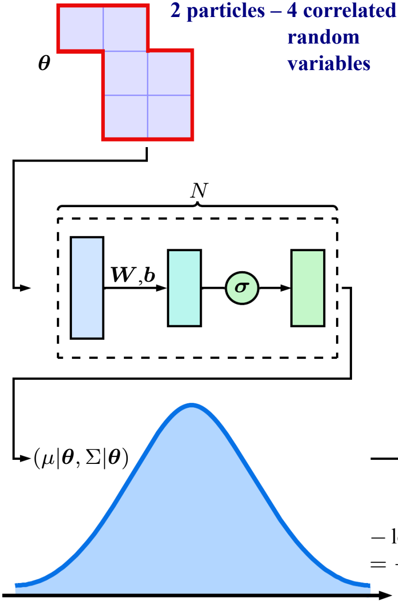


$$\longrightarrow \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

$$-\log \mathcal{L}\left(\mu, \Sigma\right) =$$
$$= -\log \mathbb{P}\left(\mu, \Sigma\right) \rightarrow \min$$

Modeling the hits

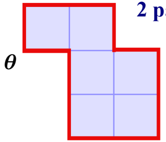
- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network



$$\longrightarrow \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

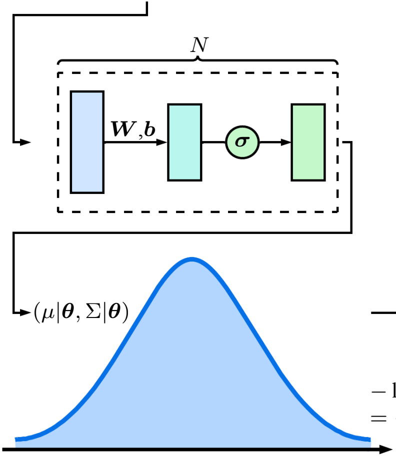
$$-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}) =$$
$$= -\log \mathbb{P}\left(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}\right) \to \min$$

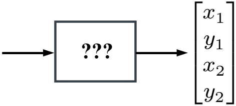
Modeling the hits



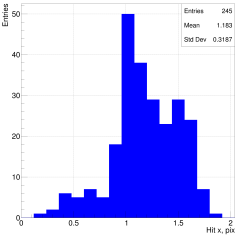
2 particles – 4 correlated random variables

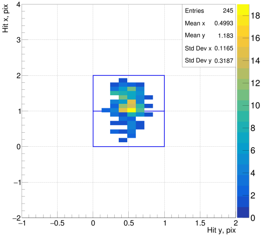
- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network
- Step 3: Look at the hit distribution – need something more expressive

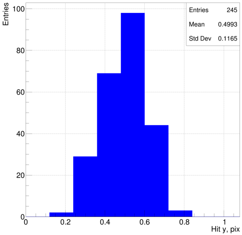




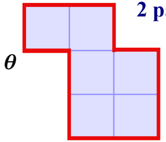
$$-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}) =$$
$$= -\log \mathbb{P}\left(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}\right) \to \min$$





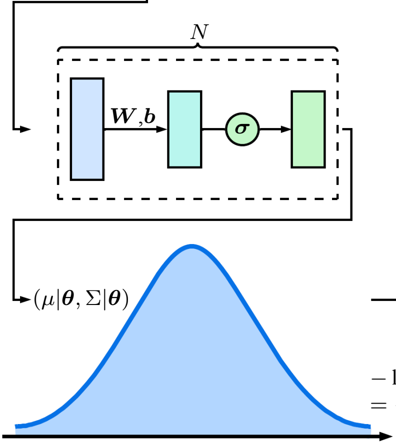


Modeling the hits



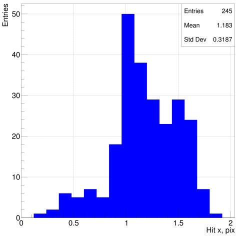
2 particles – 4 correlated random variables

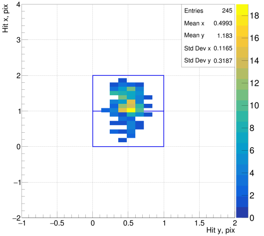
- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network
- Step 3: Look at the hit distribution – need something more expressive (**Normalizing Flow**)

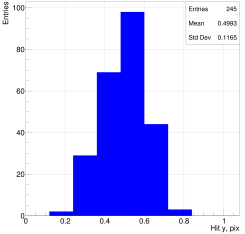


$$\longrightarrow \boxed{\textbf{NF}} \longrightarrow \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

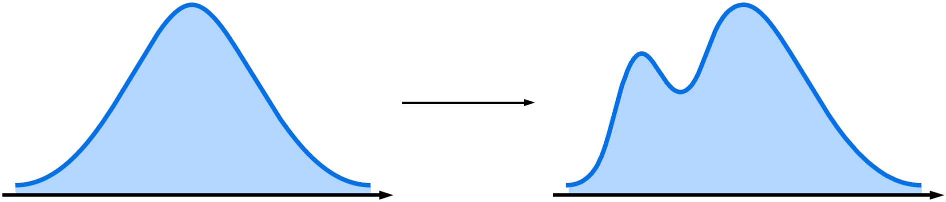
$$-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}) =$$
$$= -\log \mathbb{P}\left(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}\right) \to \min$$





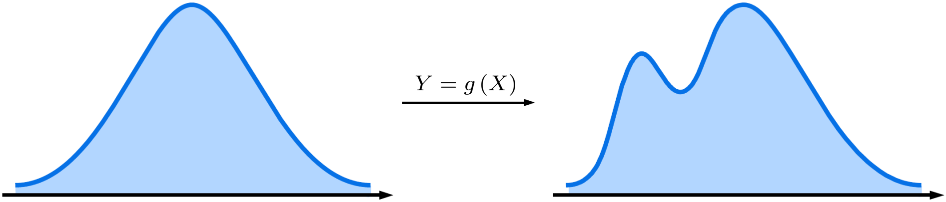


Normalizing Flows detour



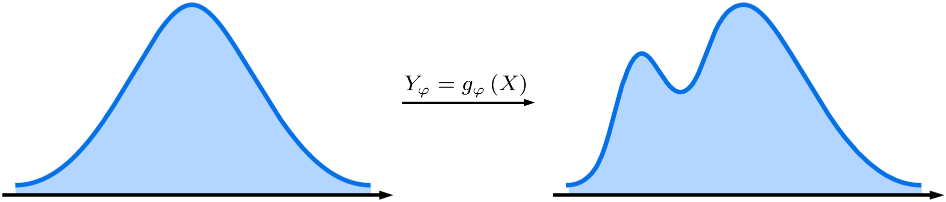
- Goal – transform a simple (tractable, samplable) distribution into a complex one

Normalizing Flows detour



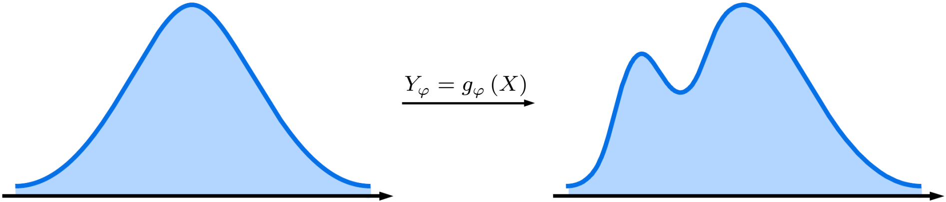
- Goal – transform a simple (tractable, samplable) distribution into a complex one
- Method – apply a sequence of complex transformations

Normalizing Flows detour



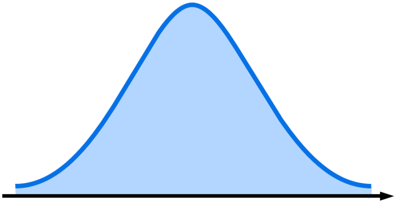
- Goal – transform a simple (tractable, samplable) distribution into a complex one
- Method – apply a sequence of complex transformations
- Final distribution may not have a tractable density – transformation has to be learnable

Normalizing Flows detour

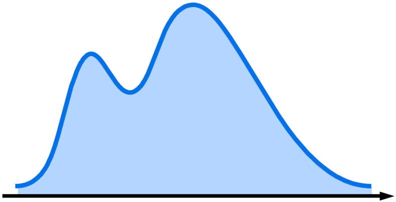


- Goal – transform a simple (tractable, samplable) distribution into a complex one
- Method – apply a sequence of complex transformations
- Final distribution may not have a tractable density – transformation has to be learnable
- We need:
  - To take gradients with respect to φ – g has to be differentiable

Normalizing Flows detour

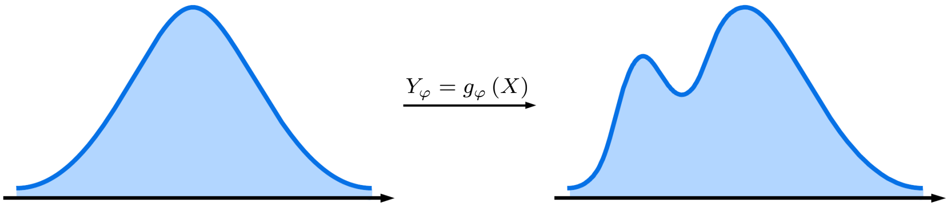


$$Y_\varphi = g_\varphi(X)$$



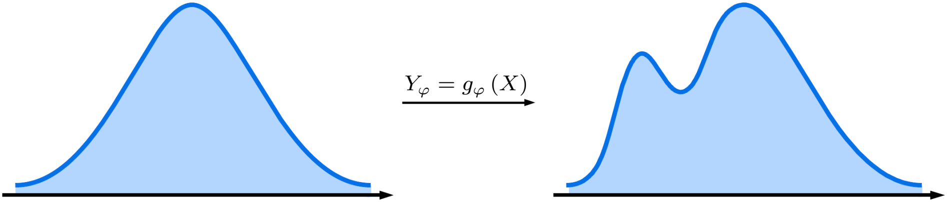
- Goal – transform a simple (tractable, samplable) distribution into a complex one
- Method – apply a sequence of complex transformations
- Final distribution may not have a tractable density – transformation has to be learnable
- We need:
  - To take gradients with respect to φ – g has to be differentiable
  - To evaluate likelihood $\mathbb{P}_Y(y|\varphi) = \mathbb{P}_X\left(g^{-1}(y)\middle|\,\varphi\right)\left|\det Dg^{-1}(y)\right|$ – g has to be invertible

Normalizing Flows detour



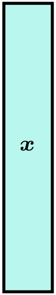
- Goal – transform a simple (tractable, samplable) distribution into a complex one
- Method – apply a sequence of complex transformations
- Final distribution may not have a tractable density – transformation has to be learnable
- We need:
  - To take gradients with respect to φ – g has to be differentiable
  - To evaluate likelihood $\mathbb{P}_Y(y|\varphi) = \mathbb{P}_X\left(g^{-1}(y)\middle|\,\varphi\right)\left|\det Dg^{-1}(y)\right|$ – g has to be invertible
- We want:
  - To model complex spaces – g has to be expressive

Normalizing Flows detour



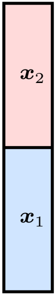
- Goal – transform a simple (tractable, samplable) distribution into a complex one
- Method – apply a sequence of complex transformations
- Final distribution may not have a tractable density – transformation has to be learnable
- We need:
  - To take gradients with respect to φ – g has to be differentiable
  - To evaluate likelihood $\mathbb{P}_Y(y|\varphi) = \mathbb{P}_X\left(g^{-1}(y)\middle|\,\varphi\right)\left|\det Dg^{-1}(y)\right|$ – g has to be invertible
- We want:
  - To model complex spaces – g has to be expressive
  - To be computationally feasible – g has to have a tractable Jacobian

Normalizing Flows detour



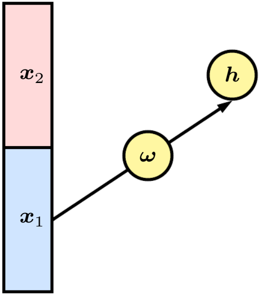
- Coupling layer approach:

Normalizing Flows detour



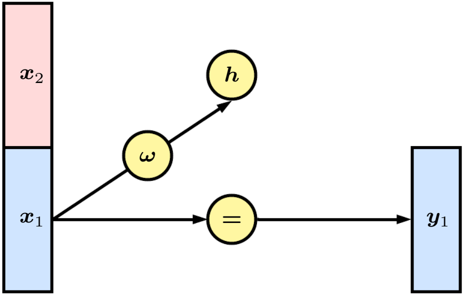
- Coupling layer approach:
  - Divide the input vector into two parts

Normalizing Flows detour



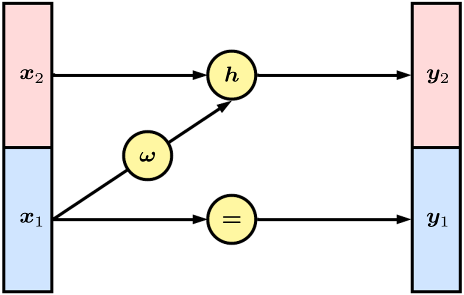
- Coupling layer approach:
  - Divide the input vector into two parts
  - Transform x_1 into a set of parameters $\boldsymbol{\omega}$

Normalizing Flows detour



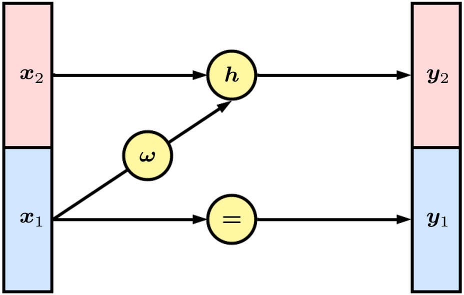
- Coupling layer approach:
  - Divide the input vector into two parts
  - Transform x_1 into a set of parameters $\boldsymbol{\omega}$
  - Pass x_1 unchanged

Normalizing Flows detour



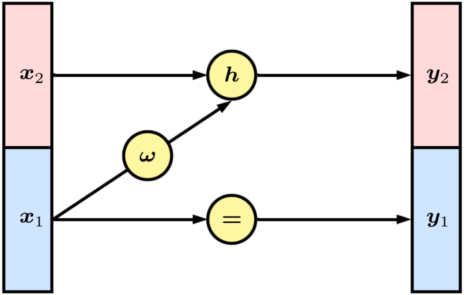
- Coupling layer approach:
  - Divide the input vector into two parts
  - Transform x_1 into a set of parameters $\boldsymbol{\omega}$
  - Pass x_1 unchanged
  - Transform x_2 via $h(.|\boldsymbol{\omega})$

Normalizing Flows detour



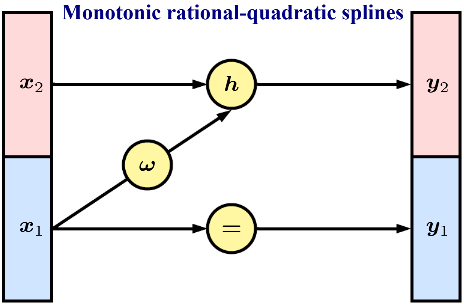
- Coupling layer approach:
  - Divide the input vector into two parts
  - Transform x_1 into a set of parameters $\boldsymbol{\omega}$
  - Pass x_1 unchanged
  - Transform x_2 via $h(.|\boldsymbol{\omega})$
- Pluses:
  - Invertibility depends only on $h(.|\boldsymbol{\omega})$ – can use a NN for $\boldsymbol{\omega}$

Normalizing Flows detour



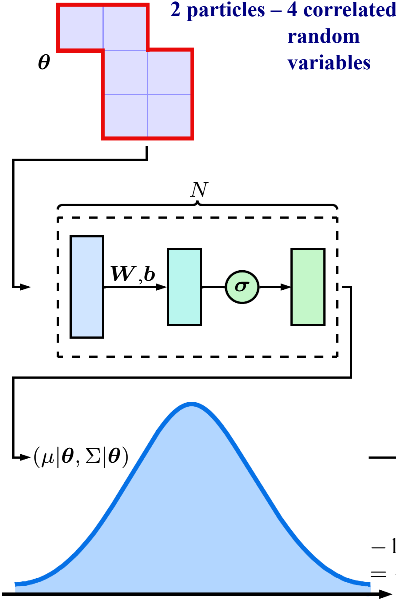
- Coupling layer approach:
  - Divide the input vector into two parts
  - Transform x_1 into a set of parameters $\boldsymbol{\omega}$
  - Pass x_1 unchanged
  - Transform x_2 via $h(.|\boldsymbol{\omega})$
- Pluses:
  - Invertibility depends only on $h(.|\boldsymbol{\omega})$ – can use a NN for $\boldsymbol{\omega}$
  - Jacobian is block-triagonal – determinant is easy to compute

Normalizing Flows detour



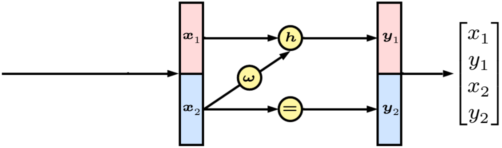
- Coupling layer approach:
  - Divide the input vector into two parts
  - Transform x_1 into a set of parameters $\boldsymbol{\omega}$
  - Pass x_1 unchanged
  - Transform x_2 via $h(.|\boldsymbol{\omega})$
- Pluses:
  - Invertibility depends only on $h(.|\boldsymbol{\omega})$ – can use a NN for $\boldsymbol{\omega}$
  - Jacobian is block-triagonal – determinant is easy to compute

Modeling the hits



2 particles – 4 correlated random variables

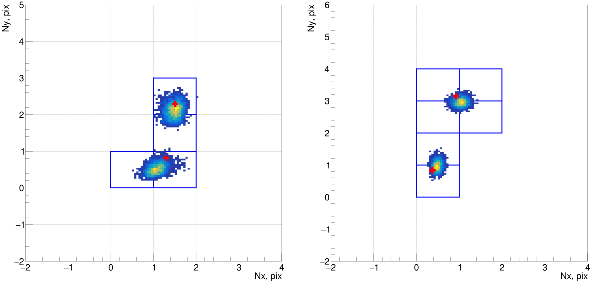
- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network
- Step 3: Look at the hit distribution – need something more expressive (**Normalizing Flow**)



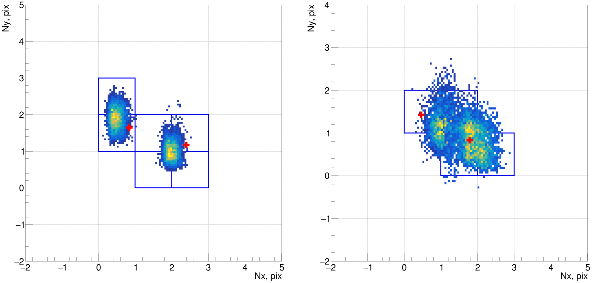
$$-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}) =$$
$$= -\log \mathbb{P}\left(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}\right) \to \min$$

NF network performance

- Observations:
  - Predictions look reasonable

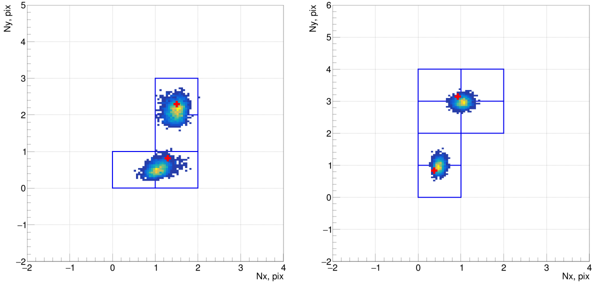


Example clusters

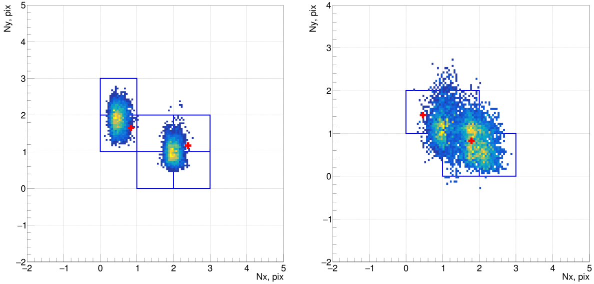


NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian

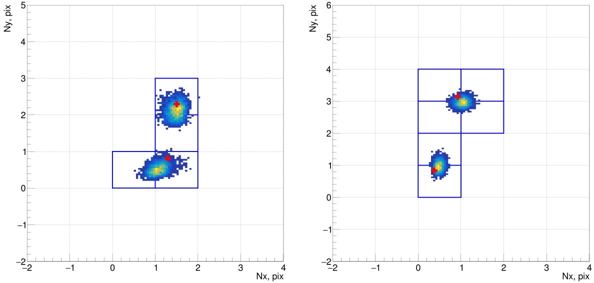


Example clusters

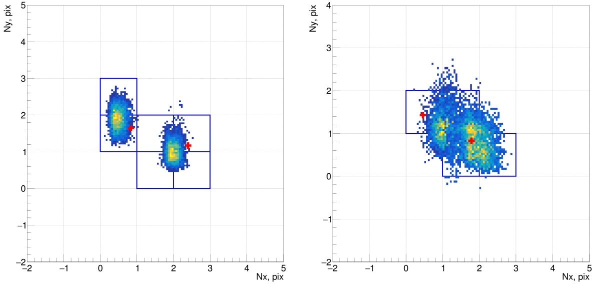


NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters

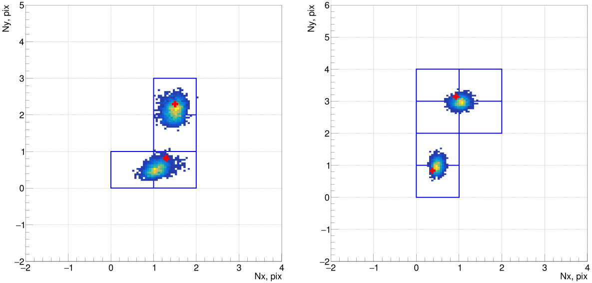


Example clusters

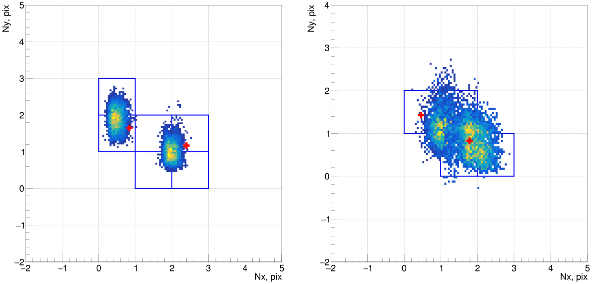


NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters



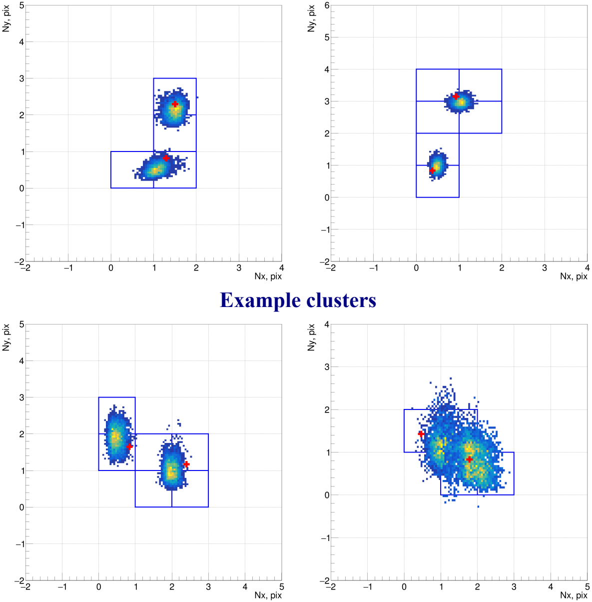
Example clusters



- GoF measures:
  - Residuals: $\boldsymbol{r} = \boldsymbol{x} - \mathbb{E}_{\Omega}\left[\hat{\boldsymbol{x}}\right]$
- Ω – set of all cluster shapes

NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters



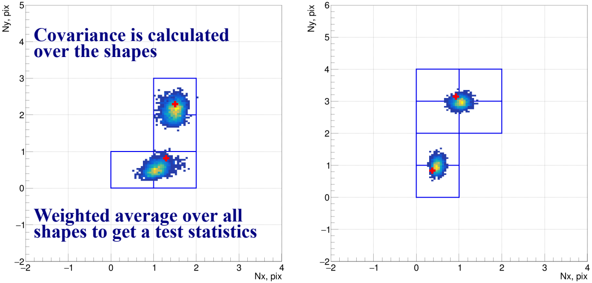
Example clusters

- GoF measures:
  - Residuals: $\boldsymbol{r} = \boldsymbol{x} - \mathbb{E}_{\Omega}\left[\hat{\boldsymbol{x}}\right]$
  - Covariance: $\boldsymbol{r} = \boldsymbol{R}_i - \mathbb{E}_{\Omega_i}\left[\hat{\boldsymbol{R}}_{\boldsymbol{i}}\right]$

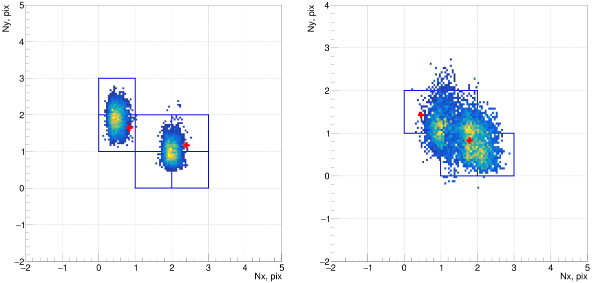
- Ω – set of all cluster shapes
- Ω_i – set of specific cluster shapes

NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters



Example clusters

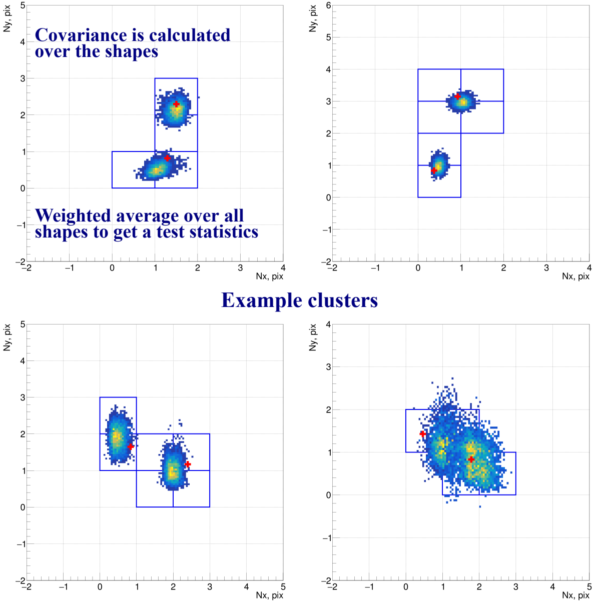


- GoF measures:
  - Residuals: $\boldsymbol{r} = \boldsymbol{x} - \mathbb{E}_{\Omega}\left[\hat{\boldsymbol{x}}\right]$
  - Covariance: $\boldsymbol{r} = \boldsymbol{R}_i - \mathbb{E}_{\Omega_i}\left[\hat{\boldsymbol{R}}_{\boldsymbol{i}}\right]$

- Ω – set of all cluster shapes
- Ω_i – set of specific cluster shapes

NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters



Example clusters

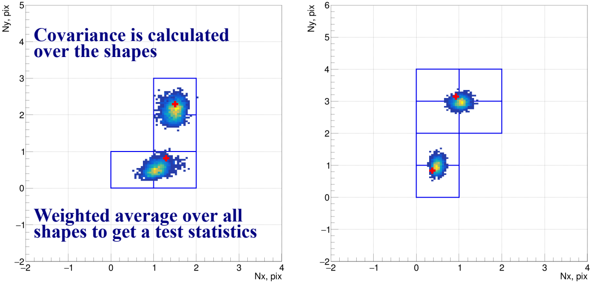
- GoF measures:
  - Residuals: $\boldsymbol{r} = \boldsymbol{x} - \mathbb{E}_{\Omega}\left[\hat{\boldsymbol{x}}\right]$
  - Covariance: $\boldsymbol{r} = \boldsymbol{R}_i - \mathbb{E}_{\Omega_i}\left[\hat{\boldsymbol{R}}_{\boldsymbol{i}}\right]$
  - GoF statisics: Nearest Neighbour mixing

- Ω – set of all cluster shapes
- Ω_i – set of specific cluster shapes

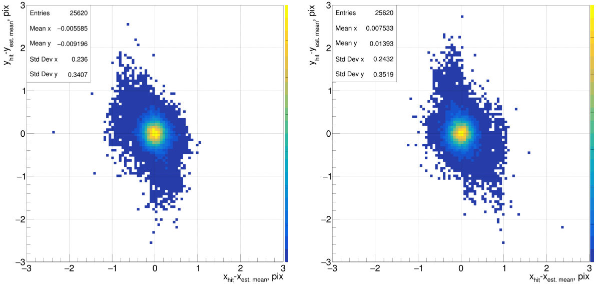
NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters
  - $\boldsymbol{r}$ width is due to the hit spread

- GoF measures:
  - Residuals: approximately 0
  - Covariance:
    On average: 42.01% error
    Diag. elements: 27.57% error
    Off-Diag. elements: 71.98% error

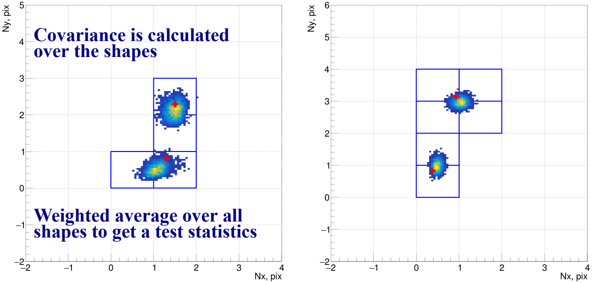


Example clusters

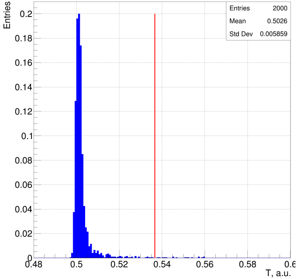


Hit residuals

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters
  - $\boldsymbol{r}$ width is due to the hit spread
  - Distribution does not describe the data (looks like)
- GoF measures:
  - Residuals: approximately 0
  - Covariance:
    On average: 42.01% error
    Diag. elements: 27.57% error
    Off-Diag. elements: 71.98% error
  - NN mixing test: $p \approx 0$

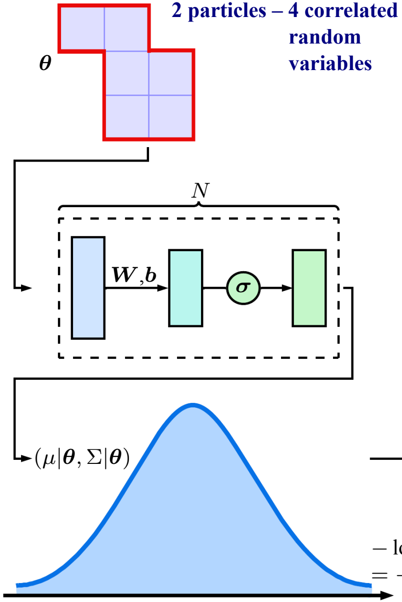


Example clusters

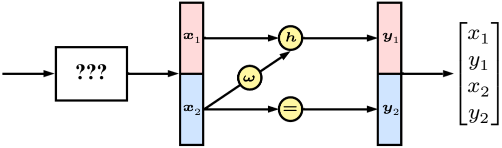


Nearest Neighbour mixing GoF results

Modeling the hits



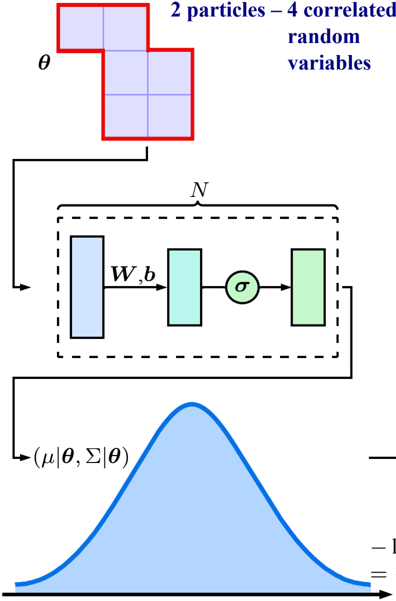
- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network
- Step 3: Look at the hit distribution – need something more expressive (**Normalizing Flow**)
- Step 4: Hit distribution is defined on a finite support – transform the base distribution



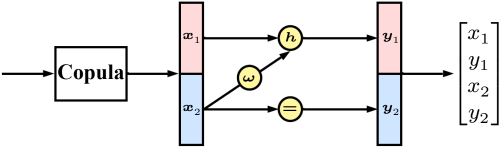
$$-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}) =$$
$$= -\log \mathbb{P}\left(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}\right) \to \min$$

Modeling the hits

2 particles – 4 correlated random variables

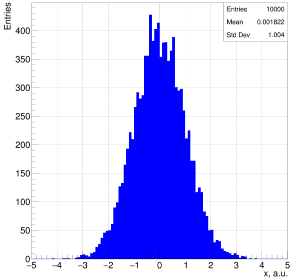


- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network
- Step 3: Look at the hit distribution – need something more expressive (**Normalizing Flow**)
- Step 4: Hit distribution is defined on a finite support – transform the base distribution (**Copulas**)

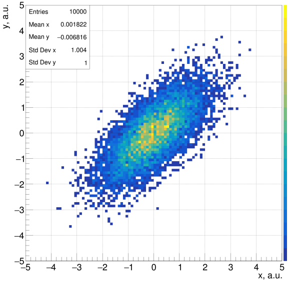


$$-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}) =$$
$$= -\log \mathbb{P}(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}) \to \min$$

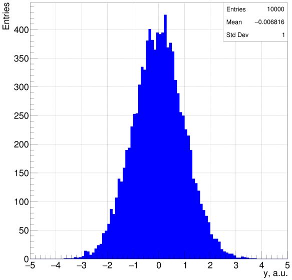
Coupulas detour



Marginal X



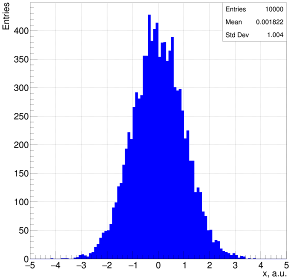
Joint



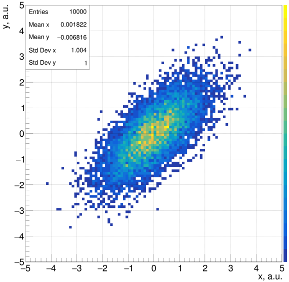
Marginal Y

- Goal – transform a Normal distribution into a finite support one

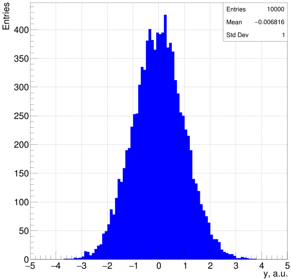
Coupulas detour



Marginal X



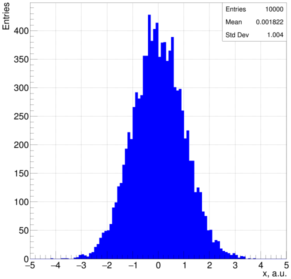
Joint



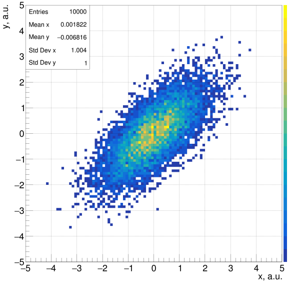
Marginal Y

- Goal – transform a Normal distribution into a finite support one
- We want to leave variables correlated

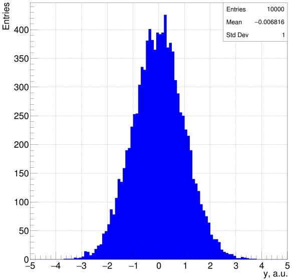
Coupulas detour



Marginal X



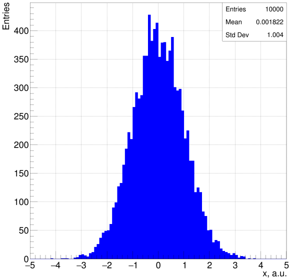
Joint



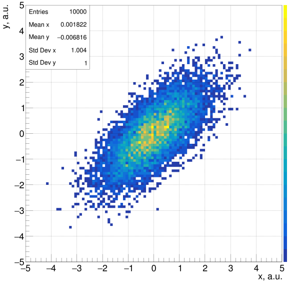
Marginal Y

- Goal – transform a Normal distribution into a finite support one
- We want to leave variables correlated
- Copula – multivariate distribution with uniform marginals

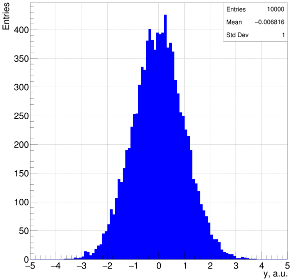
Coupulas detour



Marginal X



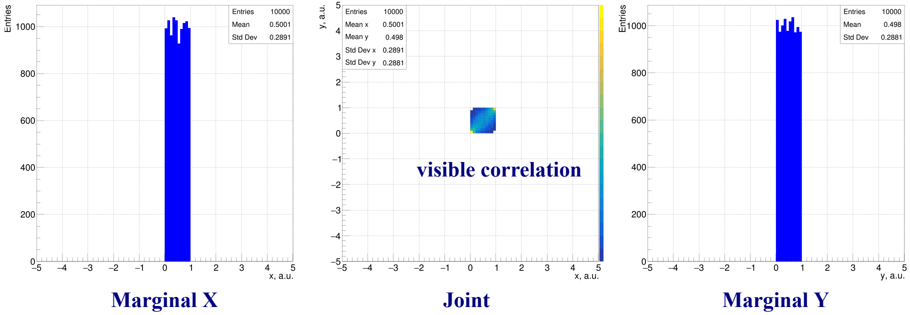
Joint



Marginal Y

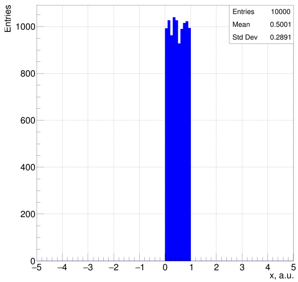
- Goal – transform a Normal distribution into a finite support one
- We want to leave variables correlated
- Copula – multivariate distribution with uniform marginals
- Step 1: Apply a CDF transformation to the normal marginals

Coupulas detour

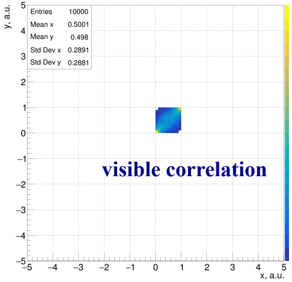


- Goal – transform a Normal distribution into a finite support one
- We want to leave variables correlated
- Copula – multivariate distribution with uniform marginals
- Step 1: Apply a CDF transformation to the normal marginals
  Result: Distribution with uniform marginals and non-trivial correlations

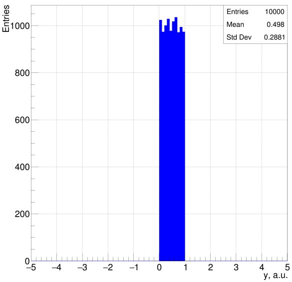
Coupulas detour



Marginal X



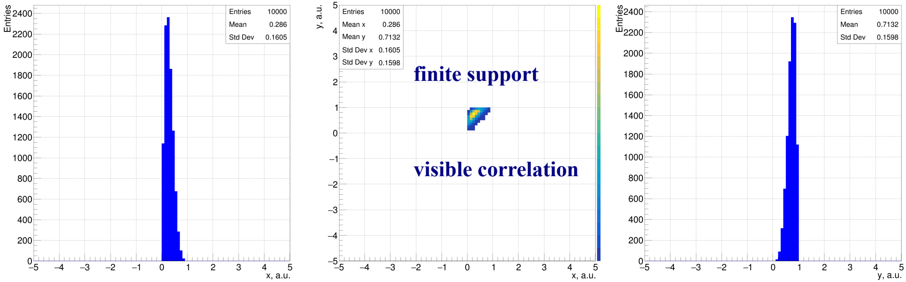
Joint



Marginal Y

- Goal – transform a Normal distribution into a finite support one
- We want to leave variables correlated
- Copula – multivariate distribution with uniform marginals
- Step 1: Apply a CDF transformation to the normal marginals
  Result: Distribution with uniform marginals and non-trivial correlations
- Step 2: Apply a ICDF transformation to the uniform marginals

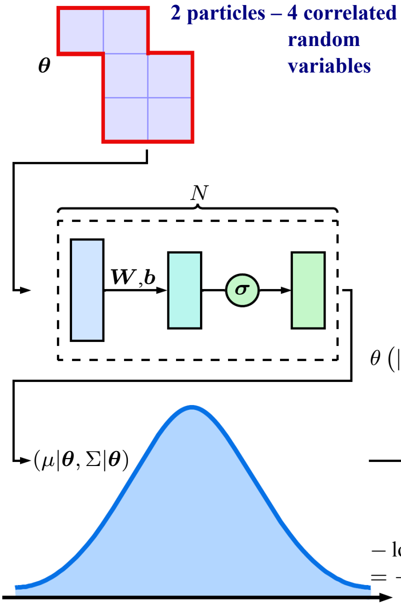
Coupulas detour



Marginal X   Joint   Marginal Y

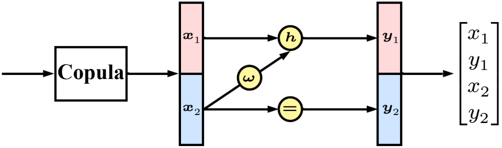
- Goal – transform a Normal distribution into a finite support one
- We want to leave variables correlated
- Copula – multivariate distribution with uniform marginals
- Step 1: Apply a CDF transformation to the normal marginals
  Result: Distribution with uniform marginals and non-trivial correlations
- Step 2: Apply a ICDF transformation to the uniform marginals
  Result: Distribution with new marginals and non-trivial correlations

Modeling the hits



- Step 1: Multivariate Gaussian model – $\mathcal{N}(\mu, \Sigma)$
- Step 2: Condition μ, Σ on $\boldsymbol{\theta}$ – Neural Network
- Step 3: Look at the hit distribution – need something more expressive (**Normalizing Flow**)
- Step 4: Hit distribution is defined on a finite support – transform the base distribution (**Copulas**)

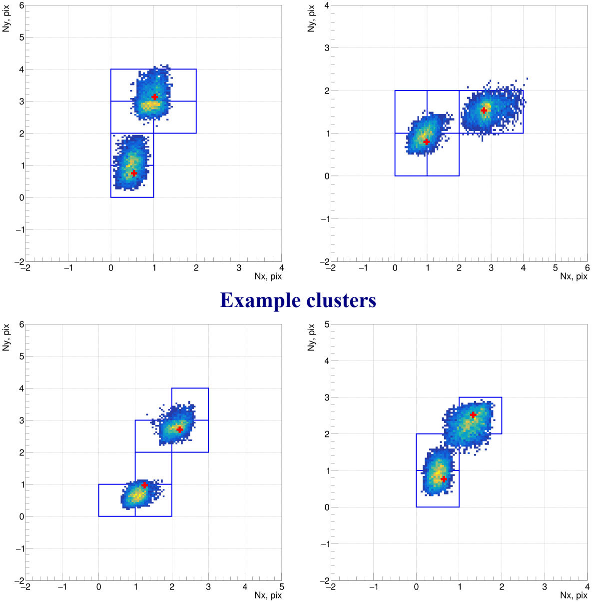
$\theta\left(|x| - \frac{\pi}{2}\right)\cos^2(x)$



$$
\begin{aligned}
&-\log \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}) = \\
&= -\log \mathbb{P}\left(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\omega}\right) \to \min
\end{aligned}
$$

Copula-NF network performance

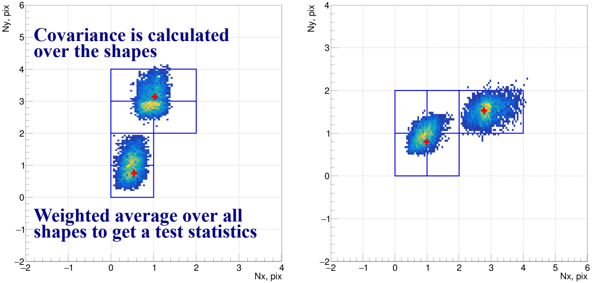
- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters
  - r width is due to the hit spread



Example clusters

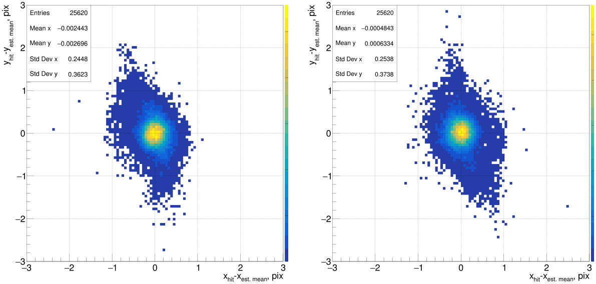
Copula-NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters
  - r width is due to the hit spread



Example clusters

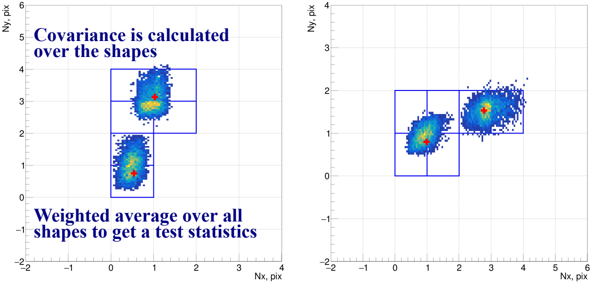
- GoF measures:
  - Residuals: even smaller
  - Covariance:
    On average: 34.05% error
    Diag. elements: 15.29% error
    Off-Diag. elements: 62.47% error



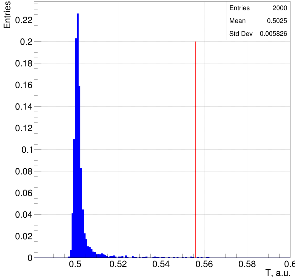
Hit residuals

Copula-NF network performance

- Observations:
  - Predictions look reasonable
  - The shapes are non-gaussian
  - Distributions exist outside the clusters
  - $\boldsymbol{r}$ width is due to the hit spread
  - Distribution does not describe the data (looks like)
- GoF measures:
  - Residuals: even smaller
  - Covariance:
    On average: 34.05% error
    Diag. elements: 15.29% error
    Off-Diag. elements: 62.47% error
  - NN mixing test: $p \approx 0$



Example clusters



Nearest Neighbour mixing GoF results

Network performance comparison

One particle case

Network	GoF parameters			
	$L_2^{\mu_1}$, pix	Err. Cov., %	Err. Cov., diag, %	Err. Cov., off-diag, %
ATLAS, MDN	0.0057	18.23	18.23	**100**
LUXE NF	0.0075	18.89	16.88	8272
LUXE, CNF	**0.0044**	**12.37**	**11.44**	2839

Two particle case

Network	GoF parameters				
	$L_2^{\mu_1}$, pix	$L_2^{\mu_1}$, pix	Err. Cov., %	Err. Cov., diag, %	Err. Cov., off-diag, %
ATLAS, MDN	0.0555	0.8125	90.46	89.43	100
LUXE NF	0.0137	0.0207	42.01	27.57	71.98
LUXE, CNF	**0.0030**	**0.0072**	**34.05**	**15.29**	**62.47**

- The results are shown for the cluster shapes with >5k events
- The bold numbers are the results that are the most compatible with the statistical error
- In one-particle case the hit coordinates are uncorrelated
- In two-particle there's evidence of inter-hit correlations
- Correlations aside, the CNF network is, generally, the top performer