

# Lossy Compression of HEP data through Normalizing Flows

Presentation for the exam project of the course

Computing Methods for Experimental Physics and Data Analysis

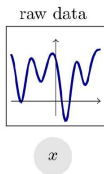
---

Luca Callisti, Marco Carotta, Igor Di Tota

- Idea del progetto
- Elementi di Teoria
  - Normalizing Flows
  - Valutazione del modello
- Implementazione
  - Dataset
  - Preprocessing
  - Modello
  - Iperparametri
  - Training
  - Compressione
- Risultati
- Test
- Sample
- Conclusioni

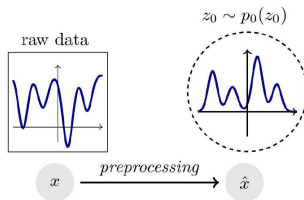
# Idea del progetto

Lo scopo di questo progetto è implementare una lossy compression dei dati, attraverso i *Normalizing Flows*.



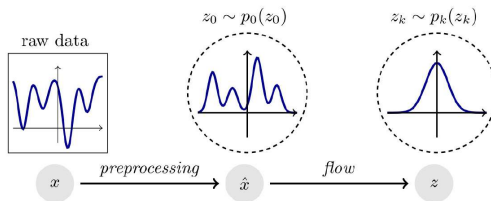
# Idea del progetto

Lo scopo di questo progetto è implementare una lossy compression dei dati, attraverso i *Normalizing Flows*.



# Idea del progetto

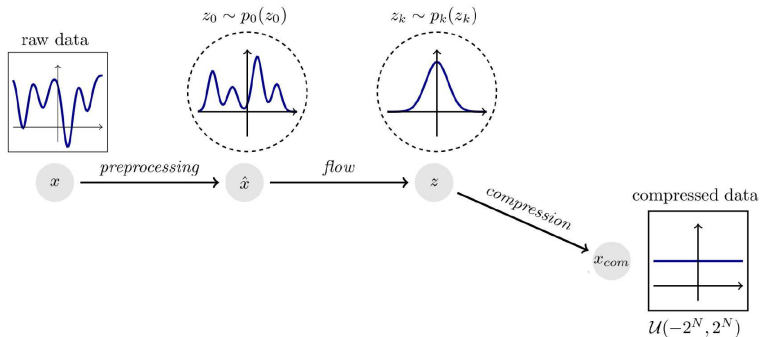
Lo scopo di questo progetto è implementare una lossy compression dei dati, attraverso i *Normalizing Flows*.



# Idea del progetto

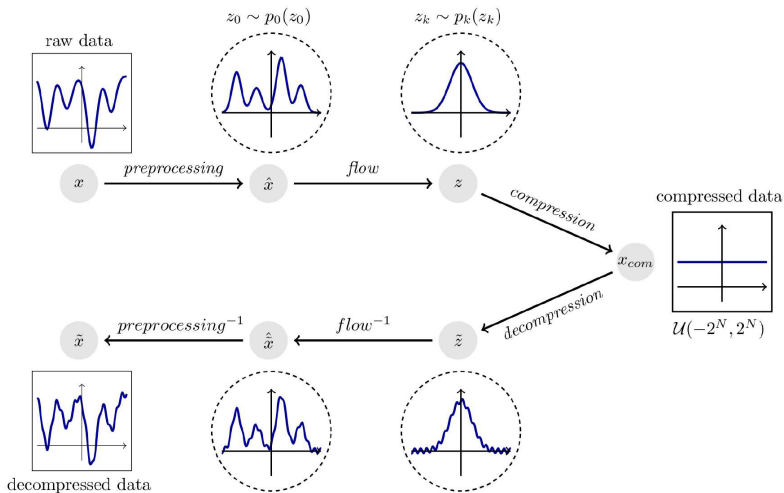
Attraverso la funzione *Erf* portiamo i dati in una distribuzione uniforme.

Per ottenere la compressione riduciamo il numero di bit usati per salvarli.



# Idea del progetto

Otteniamo i dati decompressi invertendo gli step precedenti. Infine, per valutare il modello, confronteremo i dati originali con quelli decompressi.



## Elementi di Teoria - Normalizing Flows

I *Normalizing Flows* sono un modello generativo e forniscono un metodo per costruire distribuzioni di probabilità su variabili aleatorie continue.

L'idea principale è esprimere  $\mathbf{x} \in \mathbb{R}^n$  come immagine, attraverso una trasformazione  $T$ , di  $\mathbf{u} \sim p_u(\mathbf{u})$ :

$$\mathbf{x} = T(\mathbf{u}), \quad \text{where} \quad \mathbf{u} \sim p_u(\mathbf{u})$$

in questo modo abbiamo un'espressione esplicita della distribuzione di  $\mathbf{x}$

$$p_x(\mathbf{x}) = p_u(\mathbf{u}) |\det J_T(\mathbf{u})|^{-1} = p_u(T^{-1}(\mathbf{x})) |\det J_{T^{-1}}(\mathbf{x})|$$

Per usare una rete neurale per approssimare  $T$ , si può usare come loss:

$$\mathcal{L}(\phi) = -\mathbb{E}_{p_x^*(\mathbf{x})} [\log(p_u(T^{-1}(\mathbf{x}; \phi))) + \log(\det J_{T^{-1}}(\mathbf{x}; \phi))]$$

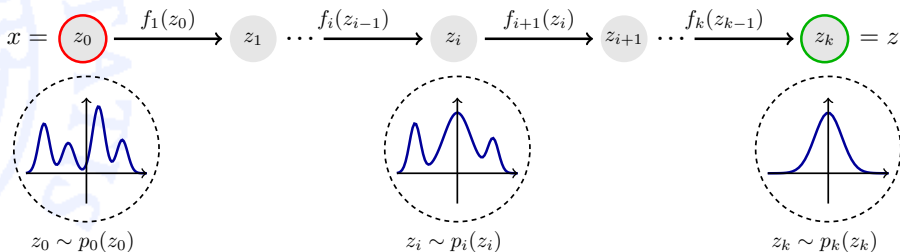
che viene dalla divergenza di Kullback-Leibler.



# Elementi di Teoria - Normalizing Flows

Nel seguito useremo la trasformazione inversa per modellare i dati nella distribuzione gaussiana e considereremo questo come flusso, cioè  $F = T^{-1}$ .

Il flusso può essere visto come composizioni di più trasformazioni che gradualmente modificano la distribuzione mappandola nella *base distribution*.



Abbiamo valutato il nostro modello in tre modi diversi:

- **Rapporto di compressione:** definito come  $R = \frac{\text{size(input file)}}{\text{size(compressed file)}}$
- **Metriche:** confronteremo i dati originali con quelli compressi attraverso:
  - $\text{Difference} = \text{original} - \text{decompressed}$
  - $\text{Response} = \frac{\text{difference}}{\text{original}}$
  - $\text{Errore relativo} = \sqrt{\left(\frac{\text{difference}}{\text{std(original)}}\right)^2}$
- **Test di Normalità:**
  - statistics: definito come  $s^2 + k^2$  dove  $s$  è la skewness e  $k$  è la kurtosis,
  - p-value: è la probabilità di rigettare l'ipotesi nulla fissato un livello di significatività  $\alpha = 0.05$ .

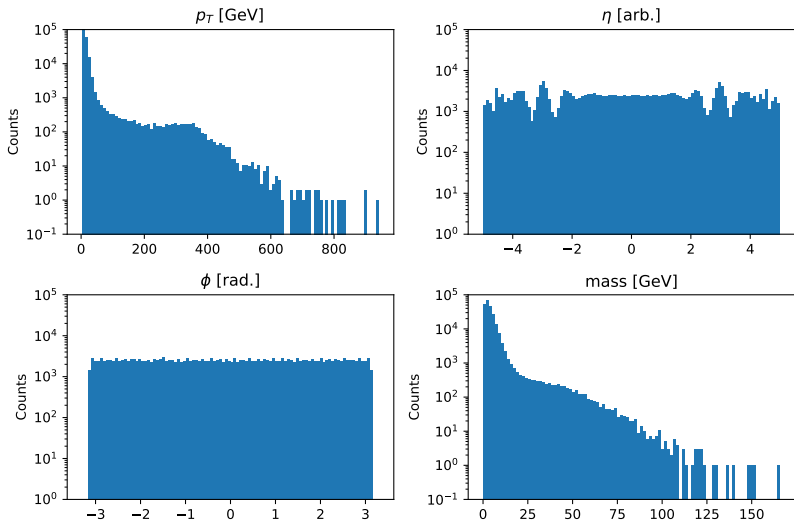
I dati utilizzati sono presi da [JetHT primary dataset in AOD format from Run of 2012](#), scaricati dal CERN Open Data Portal usando la versione CMSSW\_5\_3\_32 della CERN Virtual Machine.

Essi sono costituiti da collisioni protone-protone, in cui sono stati selezionati solo gli eventi contenenti jet. Ogni evento è caratterizzato da 24 variabili ma solo 4 sono state studiate, le componenti del quadrivettore *jet*:

$$j^\mu = (p_T, \eta, \phi, m)$$

Il numero di eventi per il dataset di training è 236413.

Variables distribution



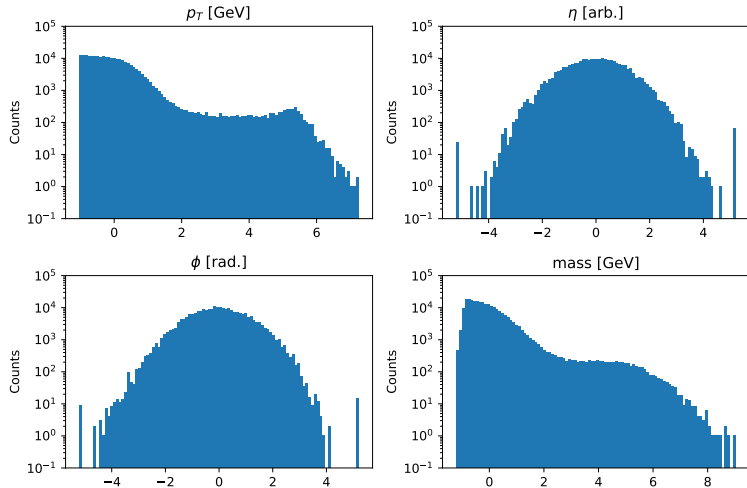
Alle features  $p_T$  e  $mass$  abbiamo applicato le seguenti trasformazioni:

- $x \rightarrow \log(x + 10)$
- StandardScaler

Mentre a  $\phi$  e  $\eta$  si è applicato:

- StandardScaler
- QuantileTrasformer

Preprocessed variables distribution



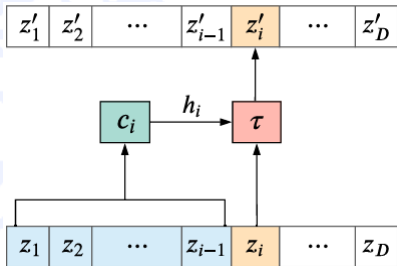
Per la creazione del modello si è usato il pacchetto `nflows`.

Ogni  $f_i$  del nostro flusso è realizzata con una  
`MaskedAffineAutoregressiveTransform`:

# Implementazione - Modello

Per la creazione del modello si è usato il pacchetto nflows.

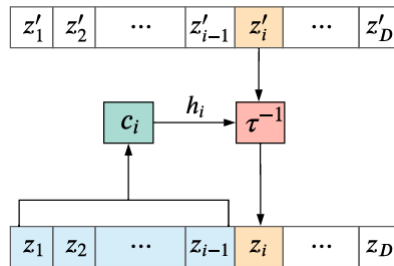
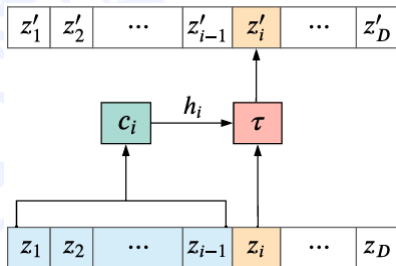
Ogni  $f_i$  del nostro flusso è realizzata con una `MaskedAffineAutoregressiveTransform`:



# Implementazione - Modello

Per la creazione del modello si è usato il pacchetto nflows.

Ogni  $f_i$  del nostro flusso è realizzata con una `MaskedAffineAutoregressiveTransform`:





Prima del training, abbiamo fatto alcune run con diversi valori degli iperparametri, in particolare abbiamo fatto variare:

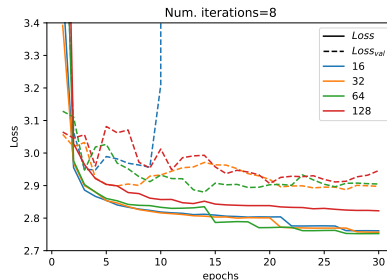
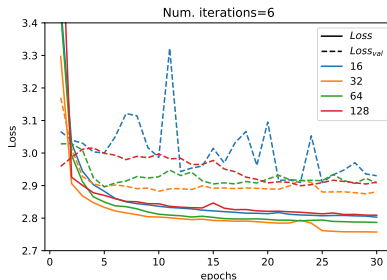
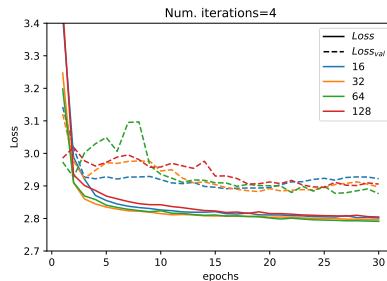
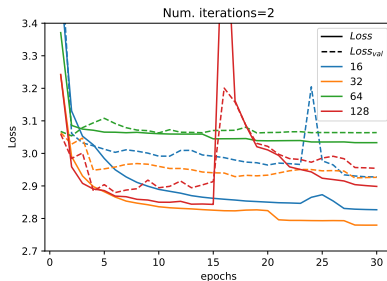
- Numero di funzioni  $f_i$  che compongono il flusso: `num_iterations` = 2,4,6,8
- Numero di hidden features: `range_hidden_features` = 16,32,64,128

Questi modelli sono stati allenati per 30 epoche e con `batch_size` = 1024 per il train e `val_batch_size` = 10000 per la validation.

# Implementazione - Iperparametri

Abbiamo scelto i seguenti modelli per il training:

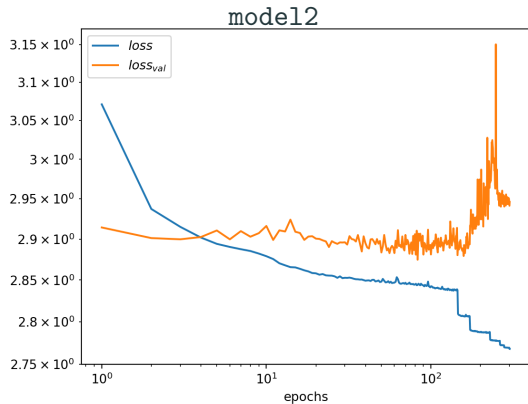
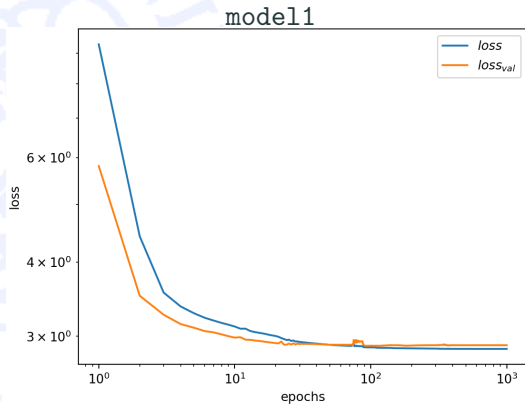
- model1:  
iterations = 6  
hidden  
features = 64
- model2:  
iterations = 4  
hidden  
features = 32



# Implementazione - Training

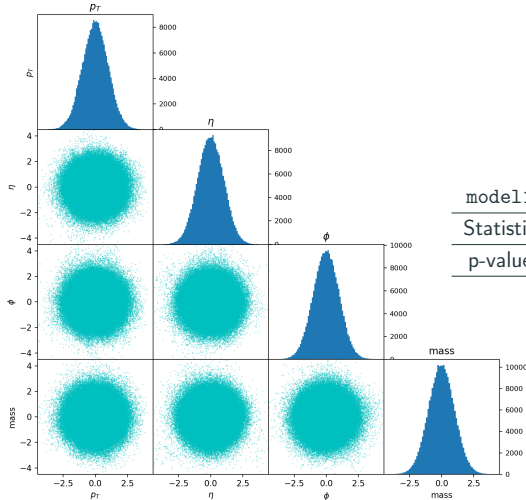
I risultato del train sono:

Name	Epochs	Batch size	Validation batch size	Loss	Validation Loss	Time [min]
model1	1000	10000	10000	2.857	2.899	37
model2	300	32	10000	2.768	2.946	311



# Implementazione - Compressione - Test di Normalità

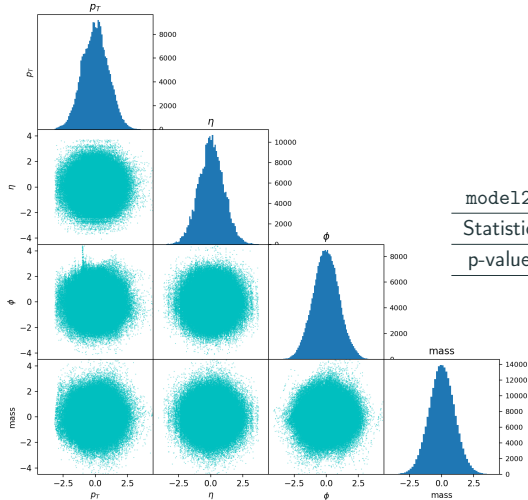
Gaussian distributions



model1	$p_T$	$\eta$	$\phi$	$m$
Statistic	7	37	128	61
p-value	$2.79 \times 10^{-2}$	$1.16 \times 10^{-8}$	$1.41 \times 10^{-28}$	$5.11 \times 10^{-14}$

# Implementazione - Compressione - Test di Normalità

Gaussian distributions



model2	$p_T$	$\eta$	$\phi$	$m$
Statistic	85	383	129	439
p-value	$3.98 \times 10^{-19}$	$5.25 \times 10^{-84}$	$9.28 \times 10^{-29}$	$5.51 \times 10^{-96}$

Gli step per ottenere la lossy compression sono i seguenti:

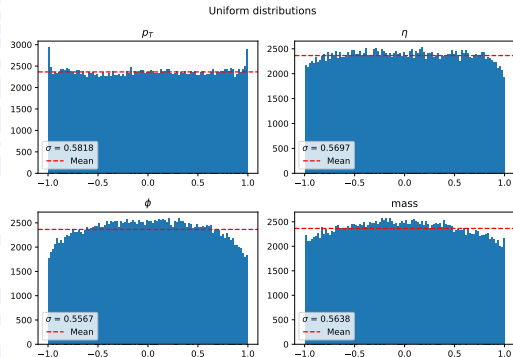
- si usa `MaxAbsScaler` sui dati distribuiti in modo gaussiano per riscalarli tra  $[-3, 3]$ ,
- si applica la funzione *Erf* per ottenere le distribuzioni uniformi,
- si moltiplica per  $2^N$  e si prende la parte intera delle distribuzioni uniformi.

I risultati di questa procedura sono i dati compressi.

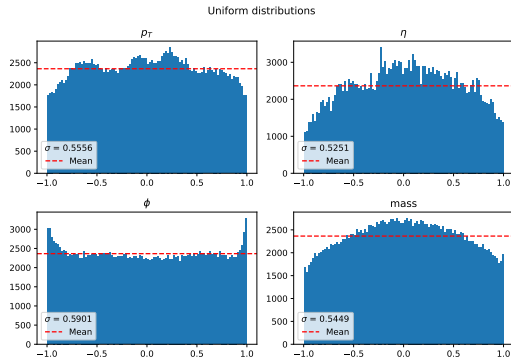
Per ottenere i dati decompressi facciamo le inverse delle trasformazioni precedenti. L'uso di `MaxAbsScaler` è giustificato dalla possibilità di ottenere valori `Inf` come risultati di *ErfInv*.

# Implementazione - Compressione - Distribuzione uniforme

Distribuzioni uniformi ottenute applicando *Erf* alle distribuzioni gaussiane:



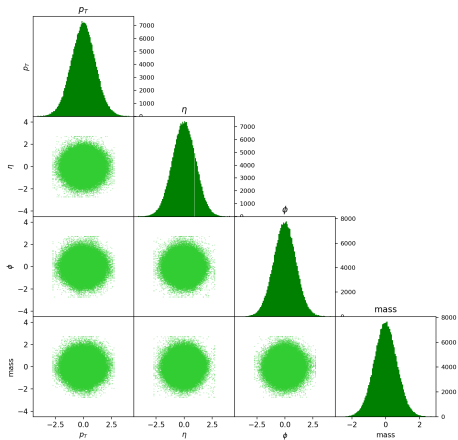
model1



model2

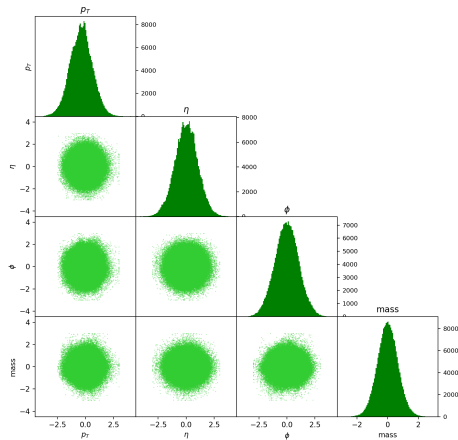
# Implementazione - Compressione

Gaussian distributions after compression



model11,  $N=13$

Gaussian distributions after compression



model12,  $N=20$



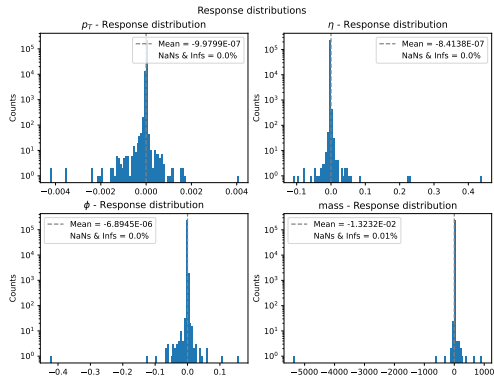
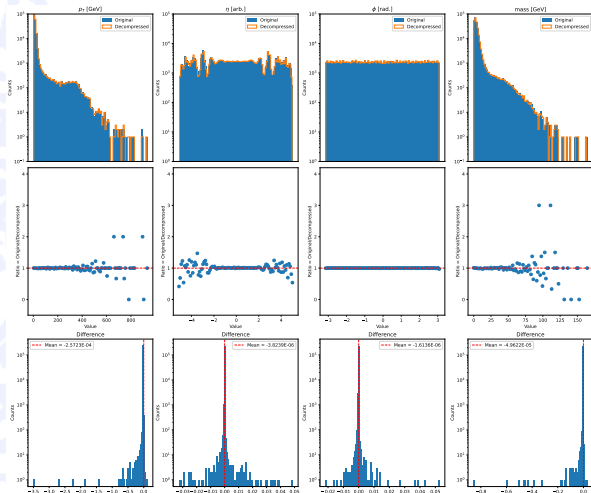
Si è calcolato  $R$  utilizzando due formati diversi: .txt e .npz.

Si notano due proprietà:

- $R$  non dipende dal modello perché le dimensioni dei file non vi dipendono,
- Le dimensioni dei file di output non dipende dal numero di bit di compressione  $N$ .

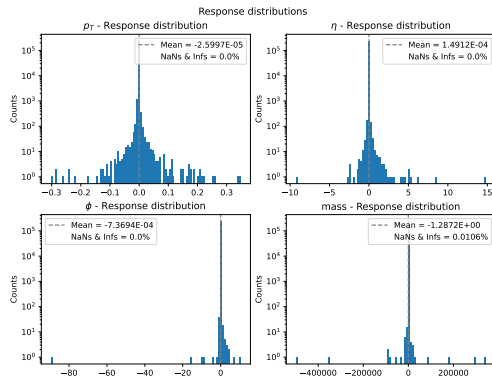
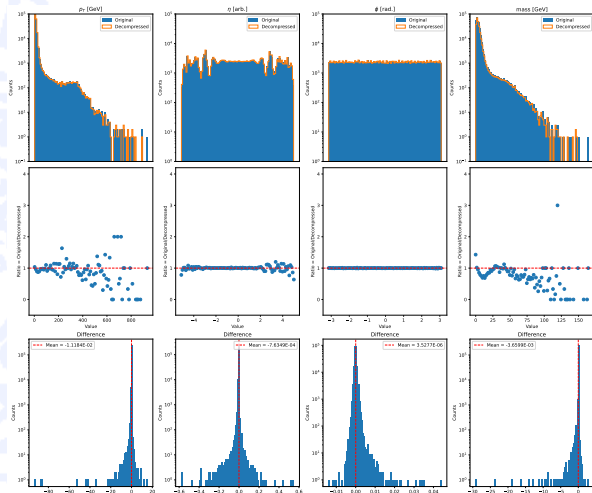
File		size( .txt ) [MB]	size( .npz ) [MB]
input		12.90	5.36
output		16.97	6.69
compressed	$N = 13$	4.83	2.34
	$N = 20$	6.69	3.18
$R$	$N = 13$	2.67	2.29
	$N = 20$	1.93	1.68

# Risultati - model1, $N = 20$



Media del modulo dei residui =  $2.60 \times 10^{-5}$   
 Errore relativo =  $1.60 \times 10^{-5}$

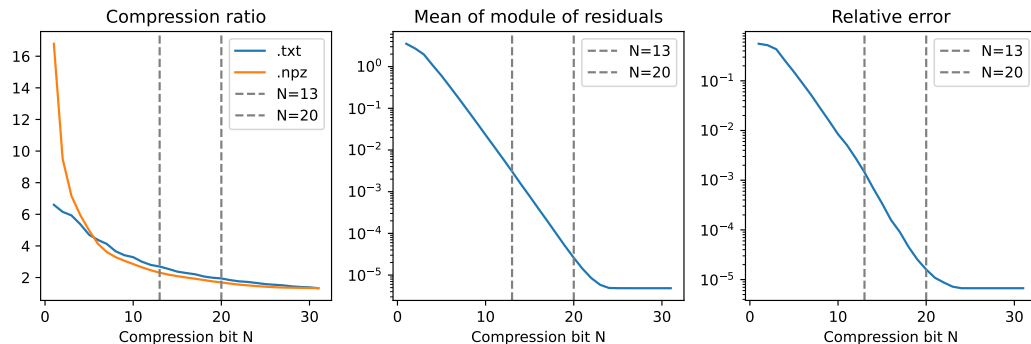
# Risultati - model2, $N = 13$

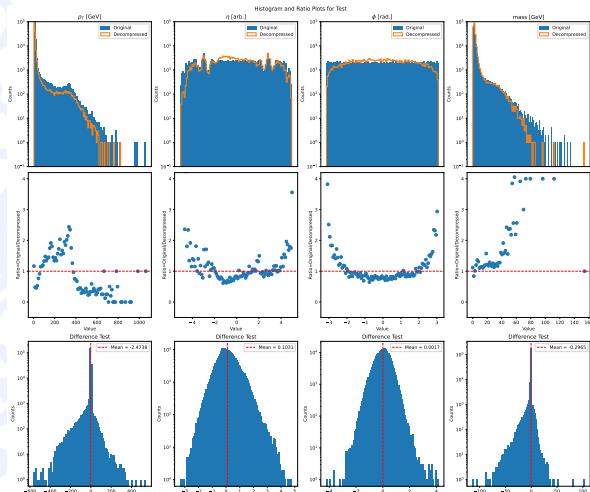


Media del modulo dei residui =  $2.44 \times 10^{-3}$   
 Errore relativo =  $7.80 \times 10^{-4}$

Si sono studiate le seguenti quantità al variare di  $N$ : il rapporto di compressione  $R$ , la media del modulo dei residui e l'errore relativo.

I plot di seguito si riferiscono al `model1`:





Test: model11,  $N = 13$

Si è testato il modello su 5 dataset da 200000 entrate. Si è calcolato il rapporto di compressione  $R$  per ogni dataset con  $N = 13$  fissato.

Riportiamo la media e la deviazione standard per ogni modello:

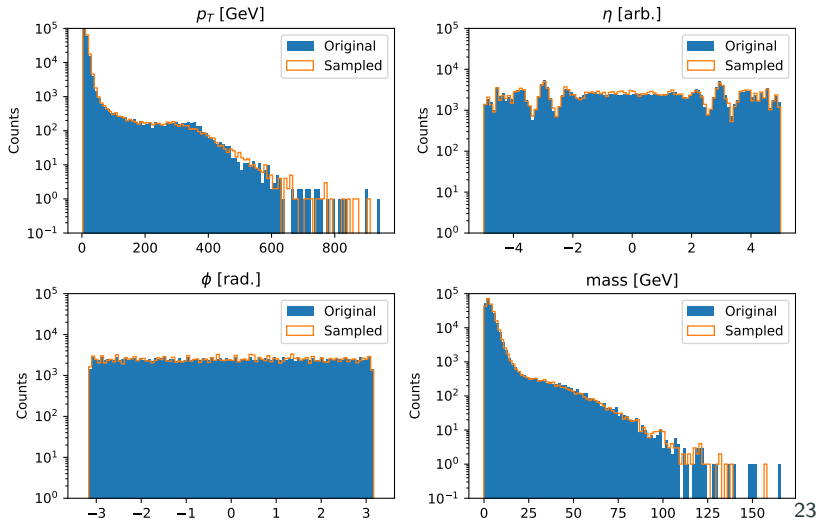
	.txt	.npz
model11	$2.682 \pm 0.011$	$2.309 \pm 0.048$
model12	$2.702 \pm 0.007$	$2.308 \pm 0.004$

# Sample

Come applicazione principale dei *Normalizing Flows* è possibile generare nuovi dati.

Riportiamo i risultati per model1.

Sampled variables distribution



I risultati ottenuti mostrano che c'è una riduzione delle dimensioni dei file, la compressione funziona. Possono essere messe in luce queste osservazioni:

- Il rapporto di compressione  $R$  dipende solo dalla dimensione del file compresso. In particolare non dipende dal modello.
- Le dimensioni del file di output non dipendono dal numero di bit di compressione  $N$ .
- Il p-value delle distribuzioni gaussiane, nonostante sia sempre minore di 0.05, non influisce sulla compressione.
- Il valore statistics appare accettabile in circa la metà delle features.

# Conclusioni

Infine, è possibile un confronto con i risultati ottenuti da Baler - Machine Learning Based Compression of Scientific Data, con  $R = 1.7$  fissato:

$R = 1.7$	$p_T$	$\eta$	$\phi$	$m$
Difference	$-1.44 \times 10^{-2} \pm 1.04 \times 10^{-1}$	$-1.12 \times 10^{-3} \pm 2.67 \times 10^{-3}$	$2.45 \times 10^{-4} \pm 1.80 \times 10^{-3}$	$-8.05 \times 10^{-3} \pm 2.51 \times 10^{-2}$
Response	$-1.07 \times 10^{-3} \pm 1.34 \times 10^{-2}$	$3.75 \times 10^{-4} \pm 6.11 \times 10^{-4}$	$3.44 \times 10^{-4} \pm 8.64 \times 10^{-4}$	$2.39 \times 10^{-1} \pm 7.87$

I nostri risultati sono:

model1		$p_T$	$\eta$	$\phi$	$m$
Difference	$N = 13$	$-2.83 \times 10^{-2}$	$-3.86 \times 10^{-4}$	$-1.75 \times 10^{-4}$	$-4.85 \times 10^{-3}$
	$N = 20$	$-2.57 \times 10^{-4}$	$-3.82 \times 10^{-6}$	$-1.61 \times 10^{-6}$	$-4.96 \times 10^{-5}$
Response	$N = 13$	$-1.22 \times 10^{-4}$	$-1.52 \times 10^{-4}$	$-4.96 \times 10^{-4}$	$-3.67 \times 10^{-1}$
	$N = 20$	$-9.98 \times 10^{-7}$	$-8.41 \times 10^{-7}$	$-6.89 \times 10^{-6}$	$-1.32 \times 10^{-2}$

model2		$p_T$	$\eta$	$\phi$	$m$
Difference	$N = 13$	$-1.12 \times 10^{-2}$	$-7.63 \times 10^{-4}$	$3.53 \times 10^{-6}$	$-3.66 \times 10^{-3}$
	$N = 20$	$-1.16 \times 10^{-4}$	$-9.29 \times 10^{-6}$	$1.02 \times 10^{-7}$	$-3.53 \times 10^{-5}$
Response	$N = 13$	$-2.60 \times 10^{-5}$	$1.49 \times 10^{-4}$	$-7.37 \times 10^{-4}$	$-1.29$
	$N = 20$	$-1.83 \times 10^{-7}$	$-1.16 \times 10^{-6}$	$-2.54 \times 10^{-6}$	$-1.98 \times 10^{-1}$

dove  $N = 13$  corrisponde a  $R = 2.29$ , mentre  $N = 20$  corrisponde a  $R = 1.68$ .



# Conclusioni

Infine, è possibile un confronto con i risultati ottenuti da Baler - Machine Learning Based Compression of Scientific Data, con  $R = 1.7$  fissato:

$R = 1.7$	$p_T$	$\eta$	$\phi$	$m$
Difference	$-1.44 \times 10^{-2} \pm 1.04 \times 10^{-1}$	$-1.12 \times 10^{-3} \pm 2.67 \times 10^{-3}$	$2.45 \times 10^{-4} \pm 1.80 \times 10^{-3}$	$-8.05 \times 10^{-3} \pm 2.51 \times 10^{-2}$
Response	$-1.07 \times 10^{-3} \pm 1.34 \times 10^{-2}$	$3.75 \times 10^{-4} \pm 6.11 \times 10^{-4}$	$3.44 \times 10^{-4} \pm 8.64 \times 10^{-4}$	$2.39 \times 10^{-1} \pm 7.87$

I nostri risultati sono:

model1		$p_T$	$\eta$	$\phi$	$m$
Difference	$N = 13$	$-2.83 \times 10^{-2}$	$-3.86 \times 10^{-4}$	$-1.75 \times 10^{-4}$	$-4.85 \times 10^{-3}$
	$N = 20$	$-2.57 \times 10^{-4}$	$-3.82 \times 10^{-6}$	$-1.61 \times 10^{-6}$	$-4.96 \times 10^{-5}$
Response	$N = 13$	$-1.22 \times 10^{-4}$	$-1.52 \times 10^{-4}$	$-4.96 \times 10^{-4}$	$-3.67 \times 10^{-1}$
	$N = 20$	$-9.98 \times 10^{-7}$	$-8.41 \times 10^{-7}$	$-6.89 \times 10^{-6}$	$-1.32 \times 10^{-2}$

model2		$p_T$	$\eta$	$\phi$	$m$
Difference	$N = 13$	$-1.12 \times 10^{-2}$	$-7.63 \times 10^{-4}$	$3.53 \times 10^{-6}$	$-3.66 \times 10^{-3}$
	$N = 20$	$-1.16 \times 10^{-4}$	$-9.29 \times 10^{-6}$	$1.02 \times 10^{-7}$	$-3.53 \times 10^{-5}$
Response	$N = 13$	$-2.60 \times 10^{-5}$	$1.49 \times 10^{-4}$	$-7.37 \times 10^{-4}$	$-1.29$
	$N = 20$	$-1.83 \times 10^{-7}$	$-1.16 \times 10^{-6}$	$-2.54 \times 10^{-6}$	$-1.98 \times 10^{-1}$

dove  $N = 13$  corrisponde a  $R = 2.29$ , mentre  $N = 20$  corrisponde a  $R = 1.68$ .

## Conclusioni - Confronto con gzip

Si è effettuata una compressione *lossless* usando gzip, il rapporto di compressione ottenuto è  $R = 2.1$ .

Con model1 si ottiene  $R \sim 2.1$  per  $N = 18$ , per cui si ha:

$$\text{Media del modulo dei residui} = 9.68 \times 10^{-5} \quad \text{Errore relativo} = 4.90 \times 10^{-5}$$

Tuttavia non riusciamo a ottenere una decompressione perfetta neanche con un  $N$  sufficientemente alto, da  $N = 24$  l'errore non diminuisce.

Per  $N = 24$  si ha:

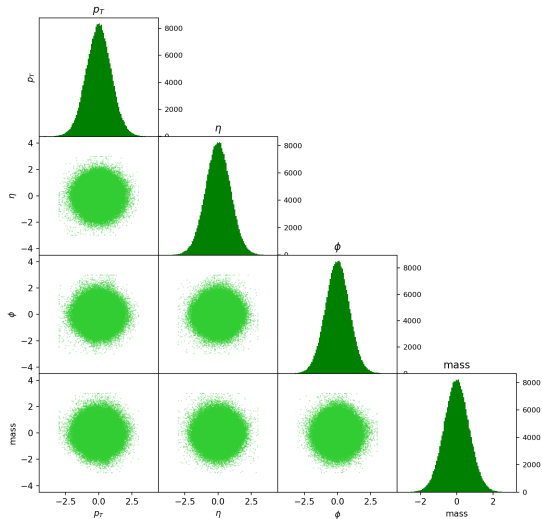
$$\text{Media del modulo dei residui} = 2.9 \times 10^{-6} \quad \text{Errore relativo} = 2.6 \times 10^{-6}$$

## Reference

- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). *Normalizing flows for probabilistic modeling and inference*. Journal of Machine Learning Research, 22(57), 1-64.
- Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. (2020). *nflows: normalizing flows in PyTorch* (v0.14). Zenodo.  
<https://doi.org/10.5281/zenodo.4296287>
- Bengtsson, F., Doglioni, C., Ekman, P. A., Gallén, A., Jawahar, P., Orucevic-Alagic, A., ... and Woolland, O. (2023). *Baler–Machine Learning Based Compression of Scientific Data*. arXiv preprint arXiv:2305.02283.

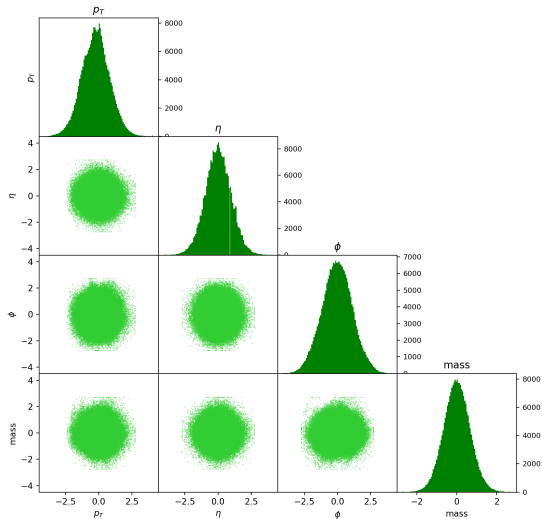
# Backup - Implementazione - Compressione: model1, $N=20$

Gaussian distributions after compression

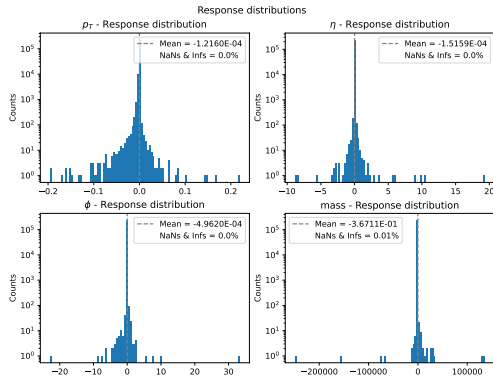
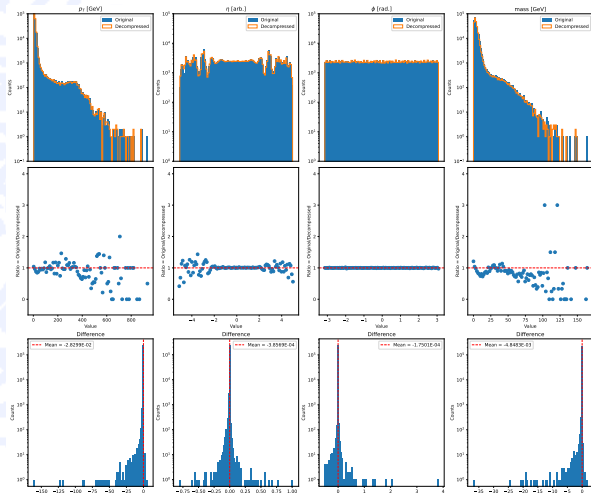


# Backup - Implementazione - Compressione: model2, $N=13$

Gaussian distributions after compression

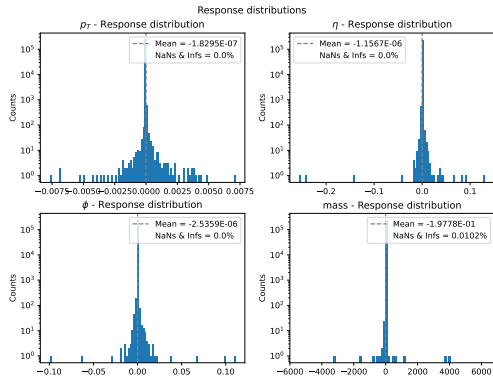
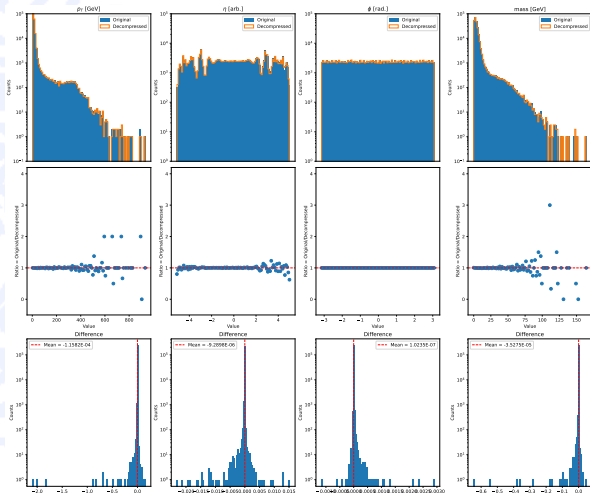


# Backup - Risultati - model1, $N = 13$



Media del modulo dei residui =  $3.06 \times 10^{-3}$   
Errore relativo =  $1.44 \times 10^{-3}$

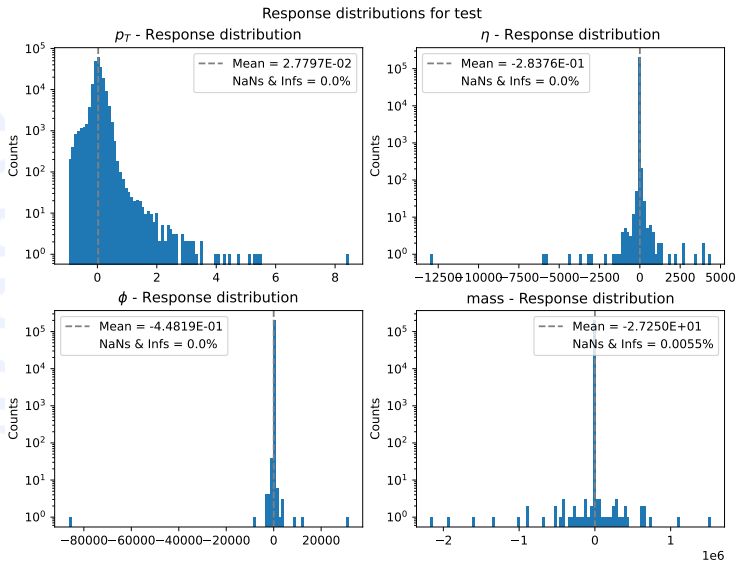
# Backup - Risultati - model2, $N = 20$



Media del modulo dei residui =  $2.09 \times 10^{-5}$

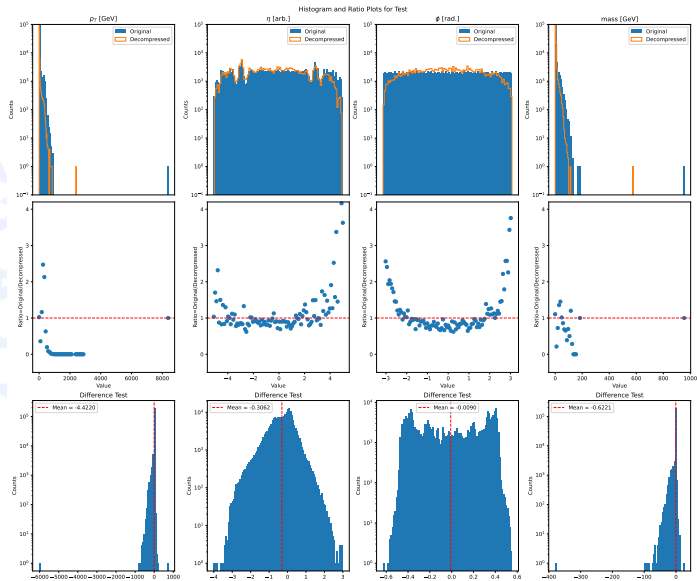
Errore relativo =  $1.13 \times 10^{-5}$

## Backup - Test: model1, $N = 13$ , response

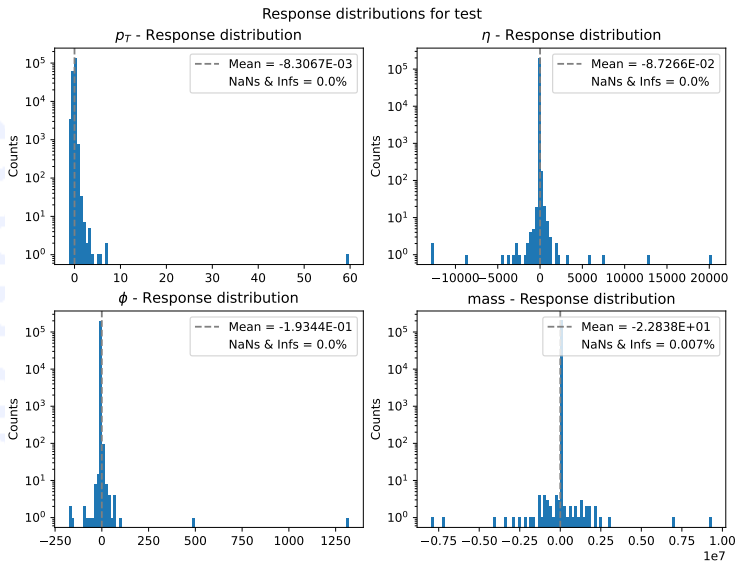




# Backup - Test: model12, $N = 13$ , difference

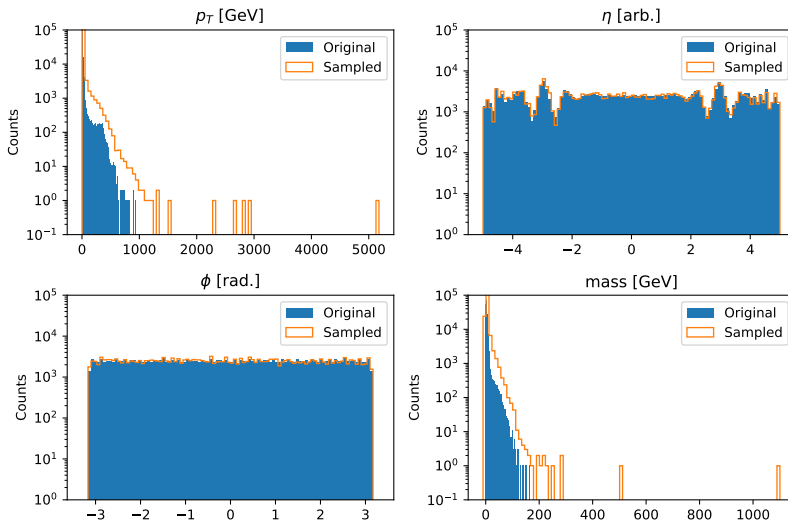


## Backup - Test: model12, $N = 13$ , response



# Backup - Sample model2

Sampled variables distribution



# Backup - Risultati - model1, $N = 30$

