EUROPEAN ORGANISATION FOR NUCLEAR RESEARCH (CERN)

Submitted to: CDS

CERN-EP/ADE
12 September 2025

Hadronic Jet Calibration & Particle Flow Networks

Student
Jacob H. Hafjall
jhafjall@gmail.com

Advisors
Tancredi Carli
tancredi.carli@cern.ch

Lukas Pfaffenbichler
lukas.pfaffenbichler@cern.ch

We calibrate trigger jets in 2025 ATLAS data by training a particle flow neural network with a Gaussian ansatz. The network learns to predict calibrated offline jet-p_t's from selected jet features while at the same time improving the jet energy resolution both as pure outputs of the network.

Summer Student Project 2025 @ CERN

Contents

1 Introduction **1**

2 Jet Calibration **1**

3 Statistics and Information Theory **2**
3.1 Frequentist Inference 2
3.2 Information 3
3.2.1 Mutual Information 5
3.2.2 Units of Information Content 6
3.2.3 Kullback-Leibler divergence 6

4 Learning Jet Inference by a Gaussian Ansatz **7**
4.1 MINE Loss 7
4.1.1 Training a PFN Network with MINE Loss 9

5 Results and Discussion **10**
5.1 Training with Floating Event Quantities 12

6 Outlook **15**

7 Appendix **16**
7.1 Functional Optiumum of DV-representation 16

1 Introduction

In the LHC, mainly proton bunches each containing 10^{11} particles, are accelerated to near the speed of light before their trajectories are bend to cross each other allowing protons to collide. The energy is so vast, 13.6 TeV, that the protons penetrate each other and what really collides is the building blocks of the proton, the quarks and gluons (partons). Each such bunch crossing occurs every 25 ns with 60 pp-collisions per bunch crossing. That is 2400 million particle collisions every second! From here, a huge number of final state particles are measured. Extracting data containing useful information as well as interpreting it requires highly sophisticated detector components and analysis techniques.

In this project we will be concerned with calibration of trigger jets. Jets are phenomenological reconstructed objects build by clustering together final state particles. . They represent kinematical properties of the initially colliding partons from the hard scattering process and therefore carry the information of the initially colliding partons. We train a particle flow deep neural network (PFN) on 2025 Atlas data with high $Z+$jet p_t transverse momentum balance. The network assumes a Gaussian ansatz [1] and give directly as outputs, a frequentist inference of the calibrated trigger jet $\hat{p}_t$ and the resolution of this inference $\hat{\sigma}$ both independent of the prior distributions, the log-likelihood function needed for statistical inference and finally an estimate of the mutual information between the feature data and the target data.

In the upcoming section Sec. 2 we introduce jet calibration by studying a simple toy example. In Sec. 3 we discuss the key concepts of information theory needed to understand the reference paper of the project [1]. Ultimately, we arrive at the concept of mutual information, a key quantity in the project. In Sec. 4 we move on to discuss the Gaussian ansatz for the neural network as well as discuss the training algorithm. Finally, in Sec. 5 we discuss the results of the project and conclude in Sec. 6.

2 Jet Calibration

Unmeasurable hadron interactions and energy losses in the detector causes jet energies to mismatch the actual energy dispersed in the initial collision. Correcting for this missing energy is *jet (energy) calibration* JEC. In practice, one can compare with truth data obtainable from some reference, like Monte Carlo simulations of the detectors and then, on average, invert the measured shape accordingly. To quantify jet calibration we introduce *response* $R = p_{t,raw}/p_{t,truth}$ where $p_{t,raw}$ is the measured jet transverse momentum and $p_{t,truth}$ is a truth measure of the same quantity computed from a reference. A simple calibration procedure would be to split the 2d distribution $(p_{t,truth}, R)$ into intervals of size Δ along the $p_{t,truth}$ axis. Then, for each interval compute the mean[1] response $\langle R(a < p_{t,truth} < a + \Delta)\rangle = \text{JEC}^{-1}(a)$. Now scaling all $p_{t,raw}$ by the jet energy correction factor JEC calibrates the jet such that $\langle R_{cali}\rangle(a) = 1 \quad \forall a$. The spread in this data is known as the jet energy *resolution* (JER) denoted σ. Since all we are really doing is a rescaling of the energy there cannot be any resolution improvement from this method. In this work, however, we also aim to improve the resolution by calibration and therefore resolution will be key in identifying the calibration performance. However the resolution is scale dependent since

$$\sigma^2 = \sum_i (R_i - \langle R\rangle)^2 \to k^2\sigma^2 \text{ for } p \to kp. \tag{1}$$

Therefore it is more natural to compute the relative resolution by dividing with the mean response. Below we illustrate this procedure with randomly generated data for visual purposes only.

[1] Or the median to be less sensitive to outliers.

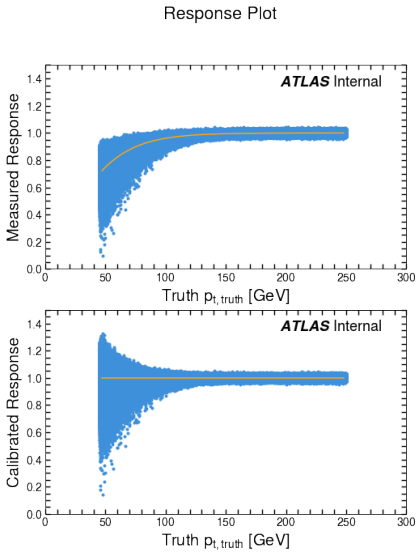

Figure 1: Response plot showing the process of JEC. The orange line is the mean calculated in small bin intervals of size corresponding to 50 bins on the truth $p_{t,truth}$ interval.

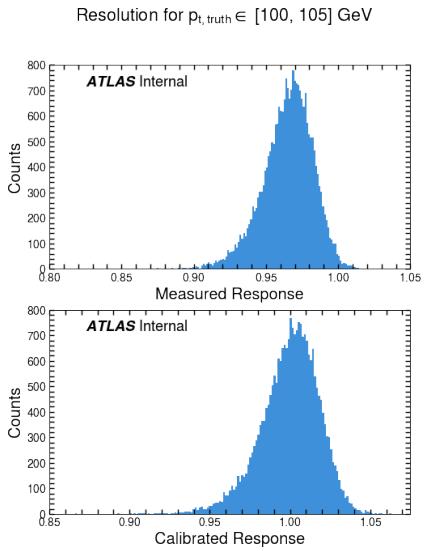

Figure 2: Computed projection of response plot in specified interval. Here the width is the resolution.

3 Statistics and Information Theory

In this section we explore theoretical basics of information theory needed for the paper [2] on which the project is based. Information theory and in general statistics are large subjects on their own and thus we will not attempt to rigorously cover every method used throughout. Instead, we will assume the *Summer Student* 2025 *Lecture Series* on statistics and only highlight key points in order to quickly dive into information theory with the aim of arriving at the idea of *mutual information*. To do so, we must first understand what *information* actually means. We gain an intuitive understanding of *Shannon information* by playing a strategic game. Following, the concept of *entropy* shall naturally emerge and mutual information an unavoidable quantity. Furthermore, we take this as an opportunity to set the notation used throughout the report.

3.1 Frequentist Inference

Any real world experiment is subject to *variety* and *uncertainty*. Uncertainty arises from practical limits, both experimentally but also theoretically. Experimentally, uncertainty comes from things like our inability to control or measure physical systems with arbitrary precision and theoretically, from unavoidable approximations one has to make in order to write down quantitative predictions of physical observables. Uncertainty itself therefore induce variety. For instance, the number of objects in a jet varies between events. In order to draw conclusions about the real world from real data subject to variability and uncertainty, one needs the mathematical framework provided by statistics. Variety gives a notion of randomness. Randomness is quantified by assigning, to each *possible* outcome of such random events, a random variable X which we wish to observe as well as an associated *probability* to each possible outcome x denoted $P(X = x) = P(x)$. The probability $P(x)$ describes the chance of seeing the outcome x. The set of possible values x might attain is the *sample space* denoted $\mathcal{A}_X$ also known as the alphabet. These three concepts together define an *ensemble*.

Definition 3.1 (Ensemble [3])**.** A (discrete) ensemble is a tuple $(x, \mathcal{A}_X, \mathcal{P}_X)$ where the outcome of the random variable x takes on a set of values $\mathcal{A}_X = \{a_1, a_2, \ldots\}$ with probabilities $\mathcal{P}_X = \{p_1, p_2, \ldots\}$.

We may sometimes refer to an ensemble by simply denoting X only. To summarize, statistics provide a mathematical framework for drawing conclusions about a *population*, a collection of elements which one wish to draw conclusions about, from the sample space. This type of reasoning is known as statistical *inference*.

One way to quantify the probability of an outcome $x = a_i$ is to assign it as the limiting value of the proportion of times the outcome occurs in n independent identically distributed (i.i.d) experiments as n goes to infinity[2]

$$P(x = a_i) = \lim_{n\to\infty} \frac{n_i}{n}. \tag{2}$$

In this way probabilities are interpreted as describing the frequency of an outcome in a random experiment. This interpretation is known as the *frequentist* viewpoint which we shall adapt. When X is continuous the probability of observing an exact value is necessarily zero since there is an infinite number of other possibilities in an arbitrary small vicinity of the fixed outcome. Instead, we can define probabilities in an infinitesimal neighborhood of $x = a$ in terms of a probability density function (pdf) $p(x)$ as

$$P(a < x < a + \mathrm{d}x) = \mathrm{d}P(x = a) = p(x = a)\mathrm{d}x. \tag{3}$$

Statistical quantities like expectation values in either the discrete or continuous case translate into each other by the usual substitution $\sum \leftrightarrow \int$. Hence

$$\langle f \rangle_P = \sum_{i=1}^{n} P(x_i) f(x_i) \qquad \text{Discrete,} \tag{4}$$

$$\langle f \rangle_P = \int \mathrm{d}P(x)\ f(x) = \int \mathrm{d}x\ p(x) f(x) \qquad \text{Continuous.} \tag{5}$$

Given a data sample of size N, both of these quantities can be estimated by the sample mean

$$\overline{f} = \frac{1}{N} \sum_{i=1}^{N} f(x_i). \tag{6}$$

3.2 Information

Definition 3.2 (Information content [3])**.** The information content learned by observing a random variable X attain a value x with probability $P(x)$ is

$$h(X = x) = \log_2 \frac{1}{P(x)}. \tag{7}$$

We will refer to $h(x)$ as *Shannon's information content* [3] and let $\log_2 = \log$.

Information is measured in *bits* meaning that if $P(x) = 1/2$ the Shannon information content of that outcome is 1 bit. Let us try to gain some intuition for why the Shannon information content indeed is a measure of the information learned by observing a random outcome. We do so by playing a game.

Game of sixty-three: Alice shows Bob a set of 64 numbers

$$\mathcal{A}: \quad 0 \quad 1 \quad 2 \quad 3 \quad \ldots 30 \quad 31 \quad 32 \quad 33 \quad \ldots \quad 61 \quad 62 \quad 63$$

and asks him to guess which one she, randomly, thinks on by asking the least number of yes/no questions. One plausible tactic is to divide the space of possibilities in half by asking *is it in this interval* $32 \leq x \leq 64$ etc. The game evolves according to the table below

[2]This is not to be taken as a rigours mathematcal definition as the limit is not well defined.

Is	*Outcome*	P	h
$x \leq 32$?	y	$\frac{1}{2}$	1 bit
$x \bmod 32 \leq 16$?	y	$\frac{1}{2}$	1 bit
$x \bmod 16 \leq 8$?	n	$\frac{1}{2}$	1 bit
$x \bmod 8 \leq 4$?	n	$\frac{1}{2}$	1 bit
$x \bmod 4 \leq 2$?	y	$\frac{1}{2}$	1 bit
$x \bmod 2 = 1$?	n	$\frac{1}{2}$	1 bit

After these six binary questions, Bob is able to identify Alice's number. For each question Bob learns *one* bit and in total six bits of information corresponding to the number of questions asked. We can go for another tactic where the meaning of information is less obvious. Suppose Bob decided to start from 63 and simply ask, *is it 63* and continue in this fashion down the line until he identifies Alice's number. For the moment, suppose Alice thinks of the number 16. Then the game will evolve as

Is x	Outcome	P	h
64?	n	$\frac{63}{64}$	$\log \frac{64}{63} = 0.0227$ bits
63?	n	$\frac{62}{63}$	$\log \frac{63}{62} = 0.0230$ bits
$\vdots$	$\vdots$	$\vdots$	$\vdots$
33?	n	$\frac{32}{33}$	$\log \frac{33}{32}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
17?	n	$\frac{16}{17}$	$\log \frac{17}{16}$
16?	y	$\frac{1}{16}$	$\log 16$

The statement that Shannons information content measures the information learned is rather hidden if one just look at the numerical values of each outcome. However, if we add up the first 32 questions we get

$$\sum_{i=1}^{32} h(x_i) = \log \frac{64 \times 63 \times \ldots \times 33}{63 \times 62 \times \ldots \times 32} = \log \frac{64}{32} = 1 \text{ bit} \tag{8}$$

That is, after covering the first half interval Bob has learned one bit, exactly like in the previous case. In total

$$\sum h(x) = \log \frac{64 \times 63 \times \ldots \times 33 \times \ldots 17 \times 16}{63 \times 62 \times \ldots \times 32 \times \ldots 17 \times 16} = \log 64 = 6 \text{ bits.} \tag{9}$$

Again, Bob has learned 6 bits of information.

Clearly there is a strategy to the game of 63. To uncover it, let us generalize the two tactics to include an arbitrary fractional interval x rather than just the $1/2$ or the oné in sixty-four. Then the probability of Alice's number being in x is exactly x and the probability of it not being is $1-x$. The average information learnt in each question is

$$H = x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x}. \tag{10}$$

On average, Bob maximizes the information learn't by maximizing this expression, which precisely occurs at $x = 1/2$ with $H = 1$ bit. See figure 3 below. In other words, you learn the most by observing an event in which you are most ignorant about. The function above is an important function and is known as the *binary entropy function*. It is a special case of the more general quantity known as *entropy*.

Definition 3.3 (Entropy [3])**.** The entropy of an ensemble is its average information content

$$H(X) = \sum_{x \in \mathcal{A}_X} P(x) \frac{1}{\log P(x)}. \tag{11}$$

As described above, the more ignorant one is about the ensemble meaning the closer $P(x)$ is to being uniform across $\mathcal{A}_X$, the more you learn from observing an outcome x. Entropy is therefore a measure of ignorance, hence the uncertainty associated with the ensemble.

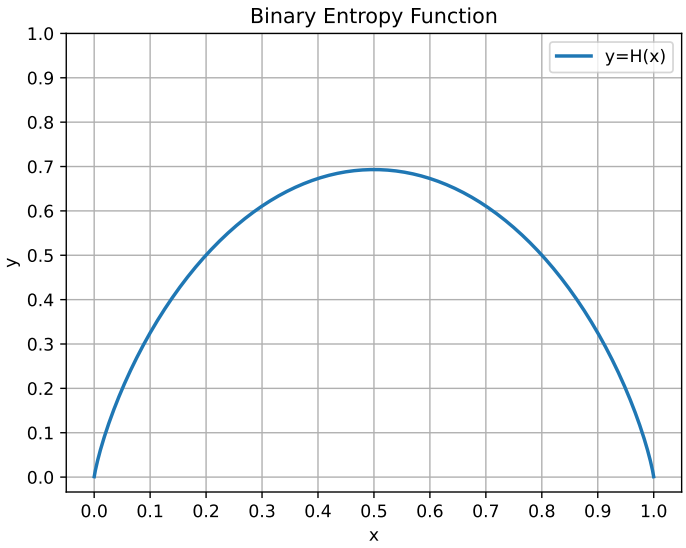

Figure 3: The binary entropy function. Clearly there is a maximum at $x = 1/2$.

3.2.1 Mutual Information

Consider a joint ensemble XY. Here each outcome is an ordered pair (x, y) with $x \in \mathcal{A}_X$ and $y \in \mathcal{A}_Y$. The probability of observing x *and* y is given by the joint probability $P(x, y)$. The entropy of the joint ensemble is defined as

$$H(X,Y) = \sum_{\substack{x\in\mathcal{A}_X \\ y\in\mathcal{A}_y}} P(x,y) \log \frac{1}{P(x,y)}. \tag{12}$$

With joint probabilities we can talk about conditional entropy. The conditional entropy of a *whole* ensemble $\mathcal{A}_X$ given a single observable outcome $y = b_k$ is

$$H(X|y=b_k) = \sum_{x\in\mathcal{A}_X} p(x|y=b_k) \log \frac{1}{p(x|y=b_k)} \tag{13}$$

Now averaging over y yields

$$H(X|Y) = \sum_{Y\in\mathcal{A}_Y} P(y)H(X|y) = \sum_{\substack{x\in\mathcal{A}_X \\ y\in\mathcal{A}_y}} P(x,y) \log \frac{1}{P(x|y)} \tag{14}$$

One can also consider $H(X)$ for joint distributions. Here $P(x)$ simply stands for the marginal probability of the variable X. However, in general it is not true that $H(X,Y) = H(X) + H(Y)$ since for joint probabilities, knowing one variable conveys information about the other as described by the conditional entropy defined above. Instead, we must have

$$H(X,Y) \leq H(X) + H(Y), \tag{15}$$

since the information contained in y has been summed over to compute the marginal $P(x)$, and similarly the other way around. The "overlap" of the right hand side measures the amount each

variable convey about the other. This dependence of X and Y and is called *mutual information* (MI) denoted $I(X;Y)$. We define it by

$$H(X,Y) = H(X) + H(Y) - I(X;Y), \tag{16}$$

with $I(X;Y) \geq 0$. This is rewritten as

$$I(X;Y) = H(X) - H(X|Y) \tag{17}$$
$$= H(Y) - H(Y|X). \tag{18}$$

It can be understood intuitively by the average reduction in uncertainty of either variable given the other. Note that we cannot define $I(X)$ in a meaningful way hence the ; separating X and Y. We may combine the two above terms into a single term

$$I(X;Y) = \left\langle \log \frac{P_{XY}}{P_X \otimes P_Y} \right\rangle_{P_{XY}}. \tag{19}$$

3.2.2 Units of Information Content

So far we have measured information in *bits*. This comes from our choice of the logarithm basis, $\log_2$, as there is otherwise no physical unit associated with our definition of information. To change units one can always multiply by an appropriate scaling factor. For logarithms, the factor is found from the logarithm change of base relation. From now on, we will continue to work in a basis of the natural logarithm ln measured in *nats* and related to $\log_2$ by

$$\ln x = \frac{\log_2 x}{\log_2 e}. \tag{20}$$

Hence to go from *bits* $\rightarrow$ *nats* one simply divides by $\log_2 e$.

3.2.3 Kullback-Leibler divergence

Among the most important quantities in information theory is the Kullback-Leibler (KL-) divergence. Given two distributions P and Q, the KL-divergence of P and Q is a measure of the distance, or similarity, between the two distributions.

Definition 3.4 (Kullback-Leibler Divergence [3])**.** The Kullback-Leibler (KL) divergence of P to Q is

$$D_{KL}(P||Q) = \left\langle \ln \frac{P}{Q} \right\rangle_P. \tag{21}$$

The distance notion follows from the *Gibb's inequality*$D_{KL}(P||Q) \geq 0$, which follows from Jensen's inequality, another important inequality in statistical analysis. Rather than coming directly from a definition of the KL-divergence in terms of probability distributions we can come from a variational point of view instead. Such two seemingly distinct viewpoints of the same quantity, or problem in general, are said to be dual.

Theorem 3.1. The KL-divergence admits a dual functional representation, known as the Donsker-Varadhan representation, given by

$$D_{KL}(P||Q) = \sup_{T \in \mathcal{T}} \langle T \rangle_P - \ln \left\langle e^T \right\rangle_Q, \tag{22}$$

where the supremum is taken over the set of all functions $\mathcal{T}$ that leave the expectations finite.

The functionality comes from the supremum part of the right hand side. It says, consider all functions which map from the space of possibilities to a real number that leaves both expectation finite. For any of the possible T functions, the right hand side is a single number. Put each number into a set. The KL-divergence is then the supremum of that set. The Donser-Varadhan representation of the Kullback-Leibler divergence is essential for the upcoming section. For completeness, let us prove the theorem [2].

Proof. Consider any function $T \in \mathcal{T}$ and two probability measures P and Q of a random variable X. Define the Gibbs distribution $\mathrm{d}G = 1/Ze^T \, \mathrm{d}Q$. Here Z is simply a constant normalization factor given by $Z = \langle e^T \rangle_Q$. Defining now the *gap*:

$$\Delta \equiv D_{KL}(P||Q) - \left(\langle T \rangle_P - \ln \langle e^T \rangle_Q \right), \tag{23}$$

we use the Gibbs distribution to write

$$\Delta = D_{KL}(P||G). \tag{24}$$

Since the gap is a KL-divergence, we must have by the Gibbs inequality $\Delta \geq 0$. Thus

$$D_{KL}(P||Q) \geq \langle T \rangle_P - \ln \langle e^T \rangle_Q \equiv \mathcal{J}[T], \ \forall \ T. \tag{25}$$

Being true for any T implies that the supremum over the set of admissible functions preserves the inequality

$$D_{KL}(P||Q) \geq \sup_{T:\ \mathcal{A}_X \to \mathbb{R}} \mathcal{J}[T]. \tag{26}$$

The largest value in the set is given by the function which extremize $\mathcal{J}[T]$ Sec. 7.1, hence

$$\frac{\delta \mathcal{J}[T]}{\delta T} = 0 \Leftrightarrow T^* = \ln \frac{P}{Q} + \text{const.} \tag{27}$$

We can examine the type of extremum by functionally expanding $\mathcal{J}[T]$ around T^* to find Sec. 7.1

$$\delta \mathcal{J}[T] = \frac{1}{2} \int \mathrm{d}x\mathrm{d}y \ \left(\frac{\delta^2 \mathcal{J}[T]}{\delta T(x) \delta T(y)} \right)_{T=T^*} \delta T(x) \delta T(y) \tag{28}$$

$$= -\frac{1}{2} \left[\langle (\delta T)^2 \rangle_P - \langle \delta T \rangle_P^2 \right] \leq 0. \tag{29}$$

Thus $\mathcal{J}[T]$ is bounded from above by an optimum function T^* and $\mathcal{J}[T^*] = D_{KL}(P||Q)$. Therefore we have $D_{KL}(P||Q) \leq \sup_{T \in \mathcal{T}} J[T]$. Combining this with the previous inequality yields the desired equality. ■

From Eq. (19) we see that MI is a special case of the KL-divergence

$$I(X;Y) = D_{KL}(P_{XY}||P_X \otimes P_Y) = \sup_{T \in \mathcal{T}} \langle T \rangle_{P_{XY}} - \ln \langle e^T \rangle_{P_X \otimes P_Y}, \tag{30}$$

which is a key identification for the upcoming section.

4 Learning Jet Inference by a Gaussian Ansatz

4.1 MINE Loss

In this section we discuss the setup for doing jet calibration using neural networks ($\mathbb{NN}$) following [1]. The starting point is the DV representation of the mutual information Eq. (30). Since $\mathbb{NN} \subseteq \mathcal{T}$ we must have

$$I(X;Y) \geq \sup_{T \in \mathbb{NN}} \langle T \rangle_{P_{XY}} - \ln \langle e^T \rangle_{P_X \otimes P_Y}. \tag{31}$$

Hence we can use any NN to place a lower bound on MI. As first done in Ref. [2], the MINE network, one can use a gradient ascent learning algorithm to maximize the right hand side and thereby give an estimate for the MI. Instead, as in Ref. [1], we put a minus in front and treat it as a loss functional which we now minimize by a gradient descent learning algorithm. Hence

$$\mathcal{L}[T] = -\left(\langle T\rangle_{P_{XY}} - \ln\langle e^T\rangle_{P_X\otimes P_Y}\right). \tag{32}$$

Then the above ineqaulity becomes

$$I(X;Y) \geq -\inf_{T\in\mathbb{NN}} \mathcal{L}[T]. \tag{33}$$

With Eq. (27), we see that minimizing the loss functional results in

$$T^* = \ln\frac{p(x,y)}{p(x)p(y)} + \text{const.} = \ln\frac{p(x|y)}{p(x)} + \text{const.} \tag{34}$$

That is the log-likelihood function describing our inference problem! Hence we can immediately write a (frequentist) maximum likelihood estimate for y

$$\hat{y}(x) = \text{argmax}_{y\in\mathcal{A}_Y}\ T(x,y). \tag{35}$$

The resolution of the estimate is characterized by the width of the log-likelihood function in the vicinity $\hat{y}$. We can characterize the width of the curve by its curvature. Necessarily

$$\text{Small width} \Leftrightarrow \text{Small uncertainty} \Leftrightarrow \text{Large curvature}, \tag{36}$$

$$\text{Large width} \Leftrightarrow \text{Large uncertainty} \Leftrightarrow \text{Small curvature}, \tag{37}$$

Hence the uncertainty of the estimate is directly related to the inverse of the second derivative on the estimate. Since a function is concave around a maximum, we put a minus and find as a measure of the resolution

$$\hat{\sigma}^2(x) = -\left(\frac{\partial^2 T}{\partial y^2}\right)^{-1}_{y=\hat{y}}. \tag{38}$$

Really what happens behind this reasoning is that one approximates the likelihood function $p(x|y)$ by a Gaussian around the estimated $\hat{y}$. Hence

$$p(x|y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\hat{y})^2}{2\sigma^2}}. \tag{39}$$

By taking the logarithm we see that we can compute σ^2 by taking two derivatives with respect to y to find Eq. (38). Had we worked with the likelihood rather than the log-likelihood we see that we would not have been able to solve for the spread σ^2.

The Gaussian ansatz is

$$T(x,y) = A(x) + (y - B(x))D(x) + \frac{1}{2}(y - B(x))^2 C(x,y), \tag{40}$$

where each of the functions $A : \mathbb{R}^M \to \mathbb{R}$, $B : \mathbb{R}^M \to \mathbb{R}$, $C : \mathbb{R}^M \times \mathbb{R} \to \mathbb{R}$ and $D : \mathbb{R}^M \to \mathbb{R}$ are deep neural networks. As training goes on, the Gaussian ansatz becomes a quadratic Taylor approximation of the log-likelihood function evaluated on $\hat{y}$ and therefore we should expect D to vanish. In the training, this is leveraged by initializing a non-zero D and then force it toward zero by an increasing L^1 regularization λ_D. After training, we now immediately find

$$\hat{y} = B(x), \tag{41}$$

$$\hat{\sigma}^2 = -C^{-1}(x,\hat{y}), \tag{42}$$

with T itself yielding an approximation for the log-likelihood function and the loss function a lower bound on mutual information. Finally, to run this algorithm the expectations in Eq. (32) are approximated by sample averages

$$\mathcal{L}[T_{GA}] = -\left(\langle T\rangle_{P^{(n)}_{XY}} - \ln\langle e^T\rangle_{P^{(n)}_X\otimes P^{(n)}_Y}\right), \tag{43}$$

which after training yields an estimate for the mutual information. Here $P^{(n)}$ is the underlying distribution of a n-sized i.i.d. data sample and $\overline{Y}$ is a permutation of the original Y data array. The function T is then evaluated on approximate uncorrelated inputs and the sample average becomes an approximation for the expectation under the product of the marginal distributions. To summarize, the Gaussian ansatz yields from an ensemble XY, a maximum likelihood estimate of y given x and its local resolution. Both quantities are computed as outputs of neural networks. Finally, it yields an estimate of the mutual information between X and Y, and T is the log-likelihood function.

4.1.1 Training a PFN Network with MINE Loss

We work with the Particle Flow Network (PFN) as [4] and build our code using the git repository in Ref. [1] as a reference. The PFN is a fully connected deep neural network and has the form

$$PFN: \quad \{z\} \to F\left(\sum_i \Phi(z_i)\right), \tag{44}$$

where $\{z\} = \{z_i \,|i = 1, 2, \ldots, \#\text{ objects}\}$ is a collection of particle objects. For instance, it may be simple trigger objects where $\{z\} = \{(p_t, \phi, \eta)\}$. An example would be training with features whos structure is purely floating event quantities. If vector event quantities are present in the features, like tower energy depositions E_t, then

$$\{z\} = \{(p_t, \phi, \eta, E_{t,1}), (p_t, \phi, \eta, E_{t,2}), \ldots, (p_t, \phi, \eta, E_{t,\#\text{ objects}})\}.$$

In this case $z_i = (p_t, \phi, \eta, E_{t,i})$. Now, the way to understand the PFN network is as follows. The Φ function maps *each* of the particle objects to a point in a latent space. This latent space contain the particle objects in a structure where adding them gives a useful quantity describing the particle objects all together. The function F now maps this quantity to the output of the network.

Each training begins with a pre-training with the mean squared error loss

$$\mathcal{L}_{\text{pre}}[B, C] = \left\langle (B(x) - y)^2 + (C(x, y) + \text{cov}(X, Y))^2 \right\rangle_{P_{XY}}. \tag{45}$$

This allows the main training of the PFN network with MINE loss function to start with good weights. From here, the main training proceeds as described below [2, 1].

Algorithm 1 Training with MINE loss.

1: **repeat**

2: Draw joint and marginal data batches:

$$(x_1, y_1), \ldots, (x_b, y_b) \sim P_{XY},$$
$$(x_1, \overline{y}_1), \ldots, (x_b, \overline{y}_b) \sim P_X \otimes P_Y$$

3: Compute Gaussian Ansatz loss:

$$\tfrac{1}{b}\sum_i T(\mathbf{x}_i, y_i) - \ln\left(\tfrac{1}{b}\sum_i e^{T(\mathbf{x}_i, y_i)}\right) \to \mathcal{L}(\mathbf{w})$$

4: Gradient (batch) descent step:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha \sum_{i=1}^{b} \frac{\partial \mathcal{L}(\mathbf{x}_i; \mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}=\mathbf{w}_i}$$

5: Update network parameters:

$$\alpha \to \tfrac{\alpha}{5}, \qquad \lambda_D \to \lambda_D \times 10$$

6: **until** convergence

If an infinity or NaN is encountered during training or inference the training has failed. To aid the convergence of the training we find it important to transform each input variable to lie in a normalized interval $[0, 1]$. Additionally, we find it convenient to train with respect to log-p_t's and only after taking the logarithm normalize it. Hence

$$p_t \to \ln p_t \;\to\; \frac{\ln p_t - \min \ln p_t}{\max \ln p_t - \min \ln p_t}, \tag{46}$$

$$x \to \frac{x - \min x}{\max x - \min x}, \quad x \neq p_t. \tag{47}$$

Below we will consider two types of inputs. One with purely floating event quantities and another with vector event quantities.

5 Results and Discussion

We train the model on 2025 data with high $Z+$jet balance. To select $Z+$jet events we introduce cuts. First we filter $Z \to \mu\mu$ events by requiring $85 < M_{\mu\mu} < 100\ GeV$ to ensure the Z-boson is present. We wish to calibrate the leading trigger jet, reconstructed with the fast anti-k_t clustering algorithm [5], matched to the corresponding offline jet ($\Delta R < 0.3$). The offline jet itself is reconstructed with the ATLAS particle flow algorithm [6]. We treat the calibrated offline jet, calibrated with the standard ATLAS procedure [7], as our target. Low p_t jets are generally more difficult to detect and reconstruct. Therefore to ensure that the leading jet is energetic enough to be reliably reconstructed we introduce a lower bound on the on the Z-$p_t \geq 45\ GeV$. At leading order the Z and the jet are back to back. This motivates filtering on the angle between the Z and the jet, $\Delta\phi > 2.5$. In addition, we restrict our analysis of the jet to its central region $|\eta| < 1$. Here, the ATLAS detector has the best performance and the calorimeter reconstructs the response homogeneously. Since we will calibrate the leading trigger jet, its transverse momentum will appear as a feature to the network. Here, one might argue that the jet η and ϕ should be included as well as they constitute together with the jet-p_t the kinematics of the jet Lorentz vector. However, the ATLAS detector is symmetric and there should therefore not be any useful information in the jet azimuth ϕ. Additionally since the ATLAS detector is homogeneous in the central region $|\eta| < 1$, the rapidity does also not contain useful information for the jet calibration. It is worth noting that the ATLAS gFex calorimeter trigger system has a granularity of $\eta \times \phi = 0.2 \times 0.2$ [8]. This

causes the jet reconstructed η to appear in integer like bins. We have indeed found that η and ϕ do not improve the jet calibration if included. Instead we give the pile-up energy density ρ which is derived from the online measurement of the luminosity as introduced in [9] and the mean number of collisions per bunch crossing μ.

In figures 4-9 below we show on the left distributions including the target p_t and on the right the distributions of the test and training data of the three features $\{p_t, \rho, \mu\}$.

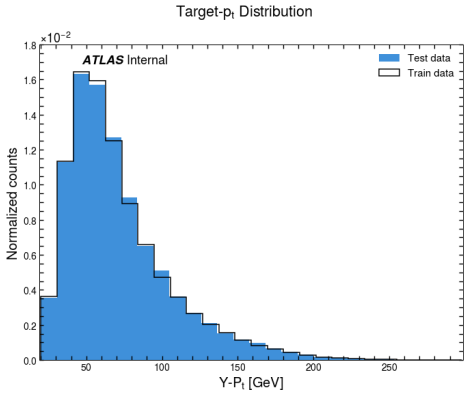

Figure 4: Target distribution.

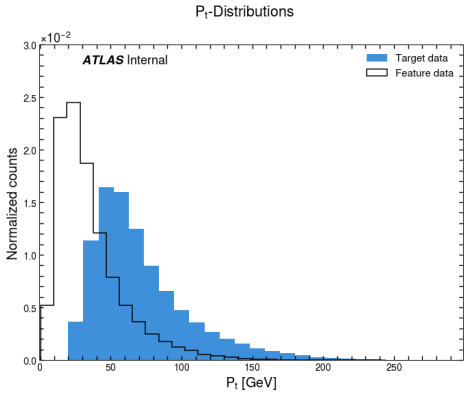

Figure 5: Uncalibrated trigger jet p_t and target distribution.

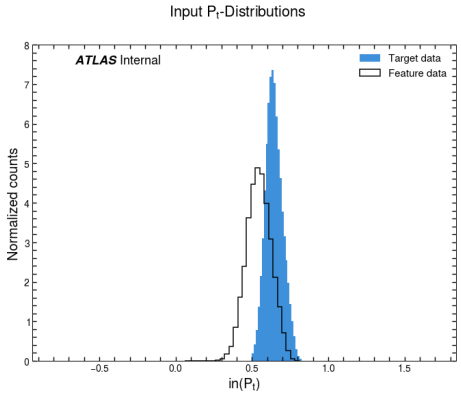

Figure 6: Same as above fig 5 but transformed according to Eq. (46).

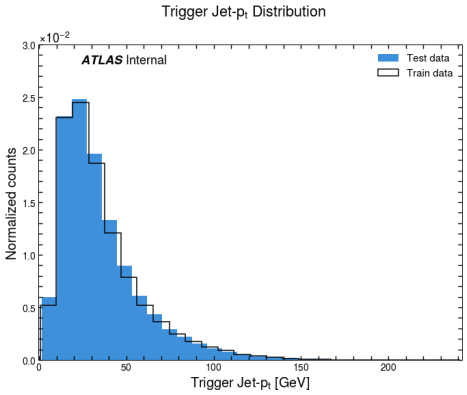

Figure 7: Raw input trigger jet p_t distribution.

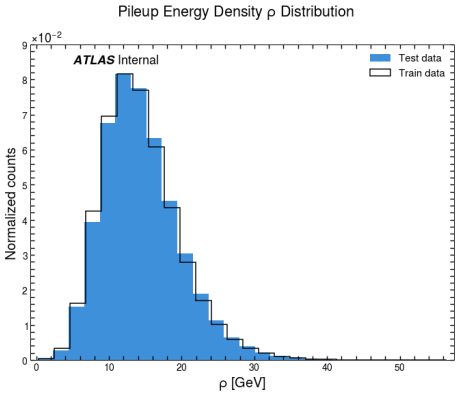

Figure 8: Raw input ρ distribution.

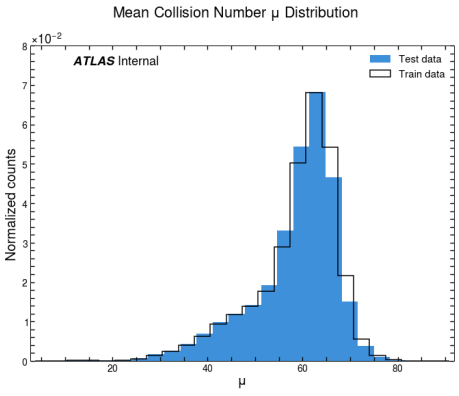

Figure 9: Raw input μ distribution.

Figure 4 shows the target distribution and figure 7 the trigger jet-p_t. On figure 5 the two distributions are shown next to each other. Here it is clear that calibration is needed. In figure 6 we show the result from applying the input transformation Eq. (46). Finally figures 8 and 9 show the distributions of the pile-up energy density ρ and the mean number of collisions per event μ. In figures 4 and figures 7-9 we display both the training and testing data. After filtering, the data is split into train/test-ratio of 75/25 %. We see that the splits are clearly identical as they should be.

5.1 Training with Floating Event Quantities

We train the model with a learning rate $\alpha = 10^{-5}$, L_1 regularization of the D network $\lambda_D = 10^{-3}$ and overall L_2 regularization $\lambda_2 = 10^{-10}$ for 50 epochs. The network parameters α and λ_D are updated four times which we refer to as four retrainings. Thus, in total the network trains for 200 epochs. Finally, the Φ network will have three layers with nodes $(200, 200, 256)$ and the F network $(100, 100, 100)$. This is summarized by table **??** below

α	λ_2	λ_D	Φ	F	Epochs	Retrain
10^{-5}	10^{-10}	10^{-3}	$(200, 200, 256)$	$(100, 100, 100)$	50	4

Table 1: Table displays network parameters.

In Sec. 1 we introduced the response R and the jet energy resolution as performance measures. For good performance we will in general look for a mean inferred response $\hat{R} = 1$ and a numerical smallest value of the inferred resolution $\hat{\sigma}$. Additionally, we will look for an improved relative resolution found from computing the standard deviation of the inferred response divided by the mean response. We will refer to the uncalibrated relative resolution as the Raw-std. and the calibrated relative resolution as Cali.std. These we will compute in three target p_t regions. One for all p_t, one for low $p_t \in [50, 55]$ GeV and one for high target $p_t \in [150, 155]$ GeV. In this sense, we interestingly find that a high mutual information not necessarily guarantees a good performance. It often happens that a training giving slightly less mutual information has better performance than a training resulting in the highest mutual information we encountered. In table 5.1 we show the performance after training. The numbers $\hat{R}$ and $\hat{\sigma}$ are averages of their corresponding distributions. The inferred response $\hat{R}$ is computed by dividing the inferred trigger jet-$\hat{p}_t$ with the target-p_t's.

Quantity	MI	$\hat{\sigma}$ [GeV]	$\hat{R}$ All p_t	$\hat{R}$, $p_t \in [50, 55]$ GeV	$\hat{R}$, $p_t \in [150, 155]$ GeV
Value	0.6840	0.4661	1.023	1.001	0.9998
Raw-std			0.3032	0.3077	0.2131
Cali.-std			0.2207	0.2236	0.1769
Improvement %			27.2	27.3	17.0

Table 2: The table shows the model performance with parameters as specified in table 5.1.

From the table 5.1 we see that the raw relative resolution improves significantly in all three regions. For low p_t the improvement is best by 10 % relative to the high p_t region. This is to be expected, for instance due to pileup. Pileup is extra energy depositions from particles not present in the primary collision but in more soft collisions in each bunch crossing. This simply adds energy and jets with low p_t will be harder to reconstruct causing the jet energy resolution to be more smeared out. This the network correctly learns and thus improves the resolution significantly. The response also averages to one. It is however slightly larger for all target p_t than the average of the two low and high target-p_t regions. Below we discuss these results graphically. First, the convergence of the model is determined by the convergence of mutual information. We find that 150 epochs of training is sufficient for convergence Fig. 10.

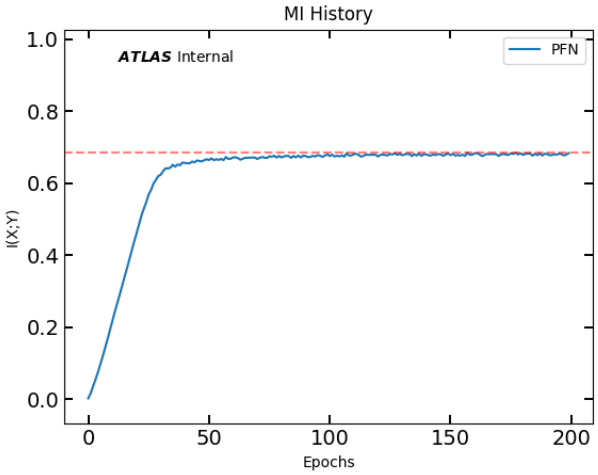

Figure 10: Mutual information as the training algorithm proceeds. After 150 epochs the training has reached equilibrium.

The distribution of the inferred jet energy resolution $\hat{\sigma}$ is seen below

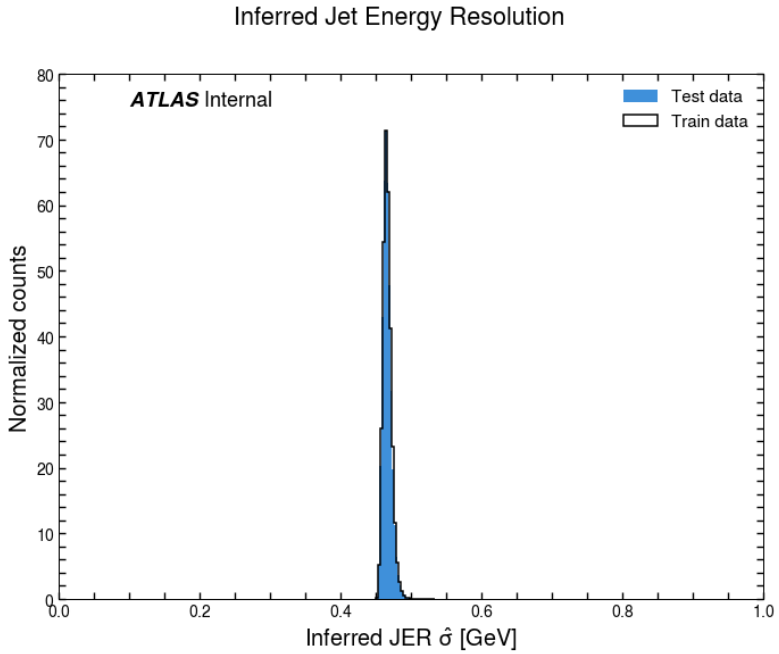

Figure 11: Distribution of the inferred resolution. It is found from computing $\hat{\sigma} = -C^{-1}(x, B(x))$ for each event Eq. (42).

First note that the model inference is equally well on training and testing data. For all trainings we find a small tail to the right. However, with increasing data statistics we have seen it shrink. The response and relative resolutions in the low and high target-p_t regions are seen in figure 12 and figures 13 and 14 respectively. On the response plot Fig. 12 we show on the top figure the raw response and on the bottom the calibrated response. In addition we display the mean and the median computed in small bin intervals. We see that they lie on top of each other which indicates that no significant outliers are present.

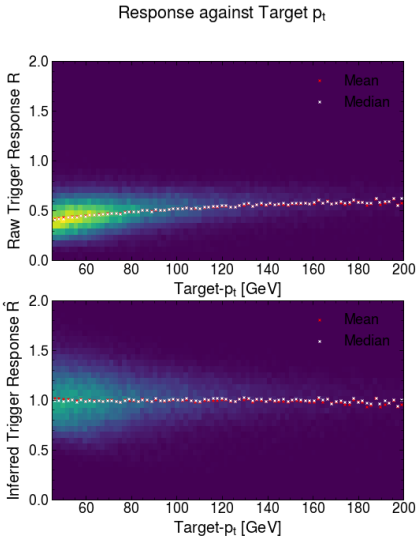

Figure 12: Response plots against target value. Top figure is the raw response, i.e. before training and bottom figure is the calibrated response. The red and white dotted lines are the mean and median computed in small bin intervals respectively.

On figure Fig. 12 it is clear that the jet energy is calibrated to have $\hat{R} = 1$ on average.

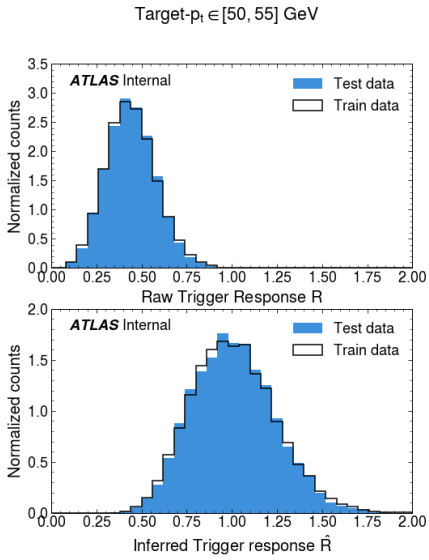

Figure 13: Low p_t projection.

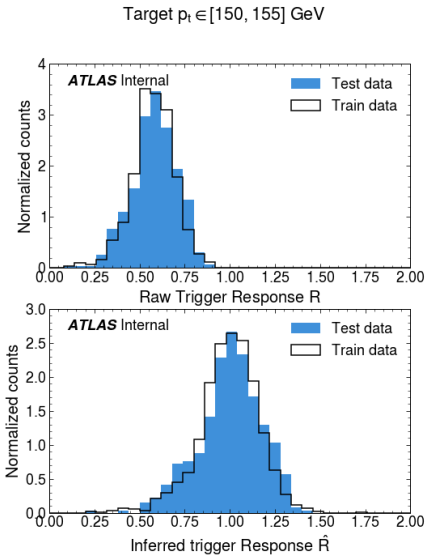

Figure 14: High p_t projection.

A final interesting plot is the T function which after training should approximate the log-likelihood function. Below we plot T against the inferred response $\hat{R}$.

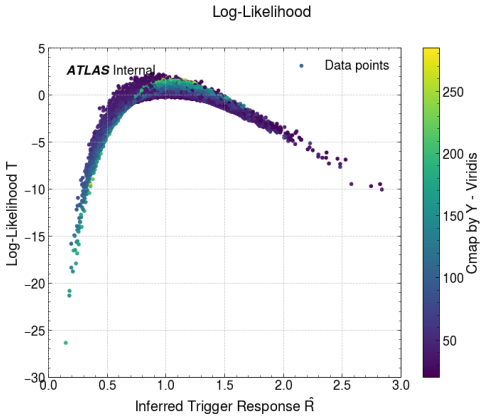

Figure 15: Log-likelihood as a function of the raw response. Color coding is normalized to the target p_t.

We see this is well approximated by a Gaussian function which reflects the well performance of the model. Secondly, the maximum occurs at $R = 1$ as expected.

We have conducted several single parameter variations starting with the initialized weights. This is done by running the training script multiple times for a fixed data loading seed and non-fixed tf-keras seed. The ladder allow weights to be sampled randomly according to tf-keras' build in weight initializer. We find that the training is not very sensitive to initial weights however some initializations do cause the training to completely fail. Furthermore we find the model to be quite robust as no significant performance changes are found by varying the network parameters in the vicinity of those specified in the parameter table 5.1.

Finally, we have examined giving the network weighted resampled data. That is, we have changed the training data as to put more emphasize on the high p_t events by re-selecting event quantities according to a probability distribution inversely proportional to the number of events for a given target p_t. We find no significant differences in the performance.

6 Outlook

We have studied calibrating jets using 2025 ATLAS data as inputs to a particle flow network [4]. When training the network by minimizing the MINE loss functional, see Eq. (43), the learned function T becomes an expression for the log-likelihood function $\ln p(x|y)$ describing the calibration task. This allows us to make a maximum likelihood inference of the calibration task Eq. (41) and infer a resolution Eq. (38). The Gaussian ansatz Eq. (40) allows us in practice to compute the statistical inference as outputs of the networks $B(x)$ and $C(x, y)$.

Moving forward will involve taking full advantage of the ATLAS detector by giving even more information to the network. This will require including trigger tower objects as inputs since they hold information useful for even more resolution improvement which in return will give an even better jet calibration. An example of including particle flow objects is shown for offline jets in the CMS detector in Ref. [1]. Having done this, it is ultimately worth thinking about hardcoding the trained B network into the trigger hardware which in return will give good online trigger jet-p_t estimates while taking full advantage of the ATLAS detector.

7 Appendix

7.1 Functional Optiumum of DV-representation

A function $f : \mathbb{R} \to \mathbb{R}$ that is infinitely continously differentiable has a Taylor series

$$f(x) = f(x^*) + \frac{\mathrm{d}f}{\mathrm{d}x}|_{x=x^*}\Delta x + \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}|_{x=x^*}\Delta x^2 + \dots. \tag{48}$$

Similarly, for a multivariable function $f : \mathbb{R}^n \to \mathbb{R}$ we can write

$$f(\vec{x}) = f(\vec{x}^*) + \sum_{i=1}^{n}\Delta x_i \frac{\partial f}{\partial x}|_{\vec{x}=\vec{x}^*} + \frac{1}{2}\sum_{i,j=1}^{n}\Delta x_i \Delta x_j \frac{\partial^2 f}{\partial x_i \partial x_j}|_{\vec{x}=\vec{x}^*} + \dots, \tag{49}$$

which is seen by expanding f in the most general power series where each coefficient is fixed by the derivatives. Importantly, in doing so one makes use of the simple identities

$$\frac{\partial x_i}{\partial x_j} = \delta_{ij}, \quad \frac{\partial}{\partial x_i}\sum_j k_j x_j = k_i. \tag{50}$$

If we now consider a continuously differentiable functional, the ladder naturally generalizes to [10]

$$\frac{\delta T(y)}{\delta T(x)} = \delta(y-x), \quad \frac{\delta}{\delta T(x)}\int \mathrm{d}y\; T(y)p(y) = p(x), \tag{51}$$

with a functional Taylor series

$$\mathcal{J}[T] = \mathcal{J}[T^*] + \int \mathrm{d}x\; \frac{\delta \mathcal{J}[T]}{\delta T(x)}|_{T=T^*}\delta T(x) + \frac{1}{2}\int \mathrm{d}x\mathrm{d}y \frac{\delta^2 \mathcal{J}[T]}{\delta T(x)\delta T(y)}|_{T=T^*}\delta T(x)\delta T(y) + \dots. \tag{52}$$

We now follow the ordinary derivative rules (defining $\mathcal{J}[T]$ according to Eq. (25))

$$\frac{\delta \mathcal{J}[T]}{\delta T(x)} = \frac{\delta}{\delta T(x)}\int \mathrm{d}z\; p(z)T(z) - \frac{\delta}{\delta T(x)}\ln \int \mathrm{d}z\; q(z)e^{T(z)} \tag{53}$$

$$= p(x) - \frac{1}{\langle e^T \rangle_Q} e^{T(x)} q(x). \tag{54}$$

Equating this to zero and solving for T yields Eq. (28). For the second derivative we get

$$\frac{\delta^2 \mathcal{J}[T]}{\delta T(x)\delta T(y)} = -\frac{e^{T(x)}\delta(x-y)q(x)\langle e^T \rangle_Q - e^{T(x)}q(x)e^{T(y)}q(y)}{\langle e^T \rangle_Q^2} \tag{55}$$

Evaluating on $T = T^* = \ln p(x)/q(x) + \text{const.}$ gives

$$\frac{\delta^2 \mathcal{J}[T]}{\delta T(x)\delta T(y)}|_{T=T^*} = -\left[p(x)\delta(x-y) - p(x)p(y)\right]. \tag{56}$$

Finally evaluating the Taylor expansion to quadratic order yields Eq. (29).

References

[1] Rikab Gambhir, Benjamin Nachman, and Jesse Thaler. "Learning Uncertainties the Frequentist Way: Calibration and Correlation in High Energy Physics". In: *Physical Review Letters* 129.8 (Aug. 2022). ISSN: 1079-7114. DOI: 10.1103/physrevlett.129.082001. URL: http://dx.doi.org/10.1103/PhysRevLett.129.082001.

[2] Mohamed Ishmael Belghazi et al. *MINE: Mutual Information Neural Estimation*. 2021. arXiv: 1801.04062 [cs.LG]. URL: https://arxiv.org/abs/1801.04062.

[3] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[4] EnergyFlow. *EnergyFlow: A Python Package for Particle Physics Tools*. https://energyflow.network/. Accessed: 2025-09-10. 2025.

[5] Matteo Cacciari and Gavin P. Salam. "Dispelling the N^3 myth for the k_t jet-finder". In: *Phys. Lett. B* 641 (2006), pp. 57–61. DOI: 10.1016/j.physletb.2006.08.037. arXiv: hep-ph/0512210.

[6] ATLAS Collaboration. "Jet reconstruction and performance using particle flow with the ATLAS Detector". In: *Eur. Phys. J. C* 77 (2017), p. 466. DOI: 10.1140/epjc/s10052-017-5031-2. arXiv: 1703.10485 [hep-ex].

[7] ATLAS Collaboration. "Jet energy scale and resolution measured in proton–proton collisions at $\sqrt{s} = 13\,\text{TeV}$ with the ATLAS detector". In: *Eur. Phys. J. C* 81 (2021), p. 689. DOI: 10.1140/epjc/s10052-021-09402-3. arXiv: 2007.02645 [hep-ex].

[8] ATLAS Collaboration. *Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System*. Tech. rep. CERN-LHCC-2013-018, ATLAS-TDR-023. CERN, 2013. URL: https://cds.cern.ch/record/1602235.

[9] Matteo Cacciari and Gavin P. Salam. "Pileup subtraction using jet areas". In: *Phys. Lett. B* 659 (2008), pp. 119–126. DOI: 10.1016/j.physletb.2007.09.077. arXiv: 0707.1378 [hep-ph].

[10] Michael E Peskin. *An Introduction to quantum field theory*. CRC press, 2018.