High Energy Physics
Coursework Report

Iman Faisal (igf23)
Word Count: 2512

Department of Physics, Cambridge

1 Question A2

Equivariance under a transformation group implies that transforming the input before applying a function yields the same result as applying the function first and then transforming the output in a corresponding manner. Mathematically, for a layer f, an input space X, an output space Y, and a group element g acting on these spaces, we require the following diagram to commute:

$$\begin{tikzcd} X \arrow[r, "g\cdot"] \arrow[d, "f"'] & X \arrow[d, "f"] \\ Y \arrow[r, "g\cdot"'] & Y \end{tikzcd}$$

This must hold for all group elements g. In this question, we restrict ourselves to the finite group of 2D rotations by multiples of 90°, $g \in \{R_{0^\circ}, R_{90^\circ}, R_{180^\circ}, R_{270^\circ}\}$, commonly known as the cyclic group C4. We aim to construct a layer that respects this symmetry.

One method for building such equivariant layers is structured weight sharing. By imposing constraints on the layer's learnable parameters we constrain the function f itself. This forces f to respect the desired symmetry. For translation equivariance in standard CNNs, the constraint is simple, the same kernel is applied everywhere. For C4 rotation equivariance, our custom Keras layer `RotationEquivariantConv` employs a different structure based on group convolution principles. The layer learns a single set of 'base' kernels, W_{base}, designed to detect specific local patterns. From this single set, four distinct filter sets, $\{W_k = R_{k\times 90^\circ}(W_{base})\}_{k=0}^{3}$, are deterministically generated by physically rotating the base kernel by 0, 90, 180, and 270 degrees using the `rotate_kernel_90` function. These four filter sets share the same underlying learnable parameters from W_{base} but are explicitly oriented to detect the base pattern at each of the four C4 orientations. The layer performs four parallel convolutions, applying each oriented filter set W_k to the input x: $C_k(x) = \text{conv2d}(x, W_k)$. The resulting feature maps are concatenated along the channel dimension, producing $f(x) = \text{concat}[C_0, C_{90}, C_{180}, C_{270}]$. A compatible tiled bias is added. This construction yields C4 equivariance because the layer is explicitly designed with oriented feature detectors derived from a shared base, and these detectors are applied simultaneously.

The C4 equivariance property of the implemented layer was numerically verified using the `check_equivariance` function. This function calculates the mean absolute difference between the layer's output for a rotated input and the appropriately transformed output for the original input. Over 1000 trials with random inputs, the difference remained below 10^{-7}, confirming the layer's equivariance within floating-point precision.

To evaluate the practical benefit, the standard MNIST dataset [1] was adapted. Each training and test image was randomly rotated by $k\times 90^\circ$ ($k \in \{0, 1, 2, 3\}$) using `numpy.rot90`. Pixel values were normalized to $[0, 1]$. Two models were then trained on this augmented dataset. The *Equivariant Model* utilized two blocks of `RotationEquivariantConv` (8 and 16 base filters) with ReLU and Average Pooling, followed by a dense classification head. The *Baseline Model* was a standard Multi-Layer Perceptron (MLP) with two hidden layers (256 and 64 units), operating on images flattened into vectors. Both models have approximately 200k trainable parameters and used Adam optimization, sparse categorical cross-entropy loss, and were trained for 5 epochs with a batch size of 128 and 10% validation split.

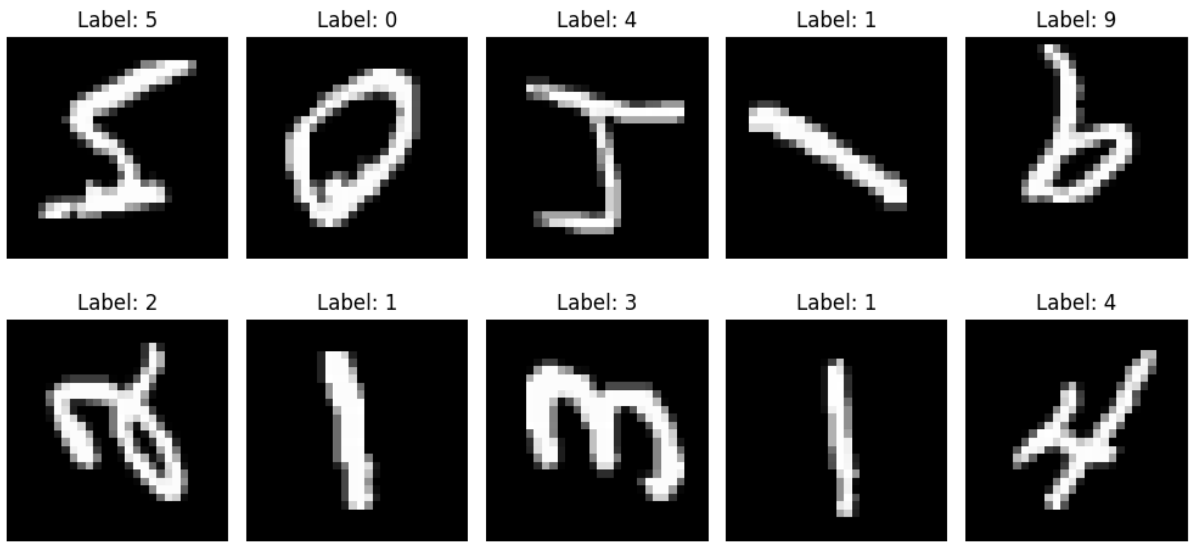

Figure 1: Visualisation of samples from the augmented MNIST dataset.

The Equivariant Model achieved **96.84%** accuracy, whereas the Baseline MLP reached **93.90%** accuracy. The training history, shown in Figure 2, provides further insight. While both models learn effectively on the training data, with the baseline model's training accuracy even slightly exceeding the equivariant model's by the final epoch. The Equivariant Model's validation accuracy starts higher and remains consistently above the Baseline Model's validation accuracy throughout training, stabilizing near 96.5%-97%. Furthermore, the Equivariant Model's validation loss remains significantly lower and more stable compared to the Baseline Model's validation loss, which fluctuates and begins to overfit. From this, it seems the equivariant layer's inductive bias leads to better generalisation.

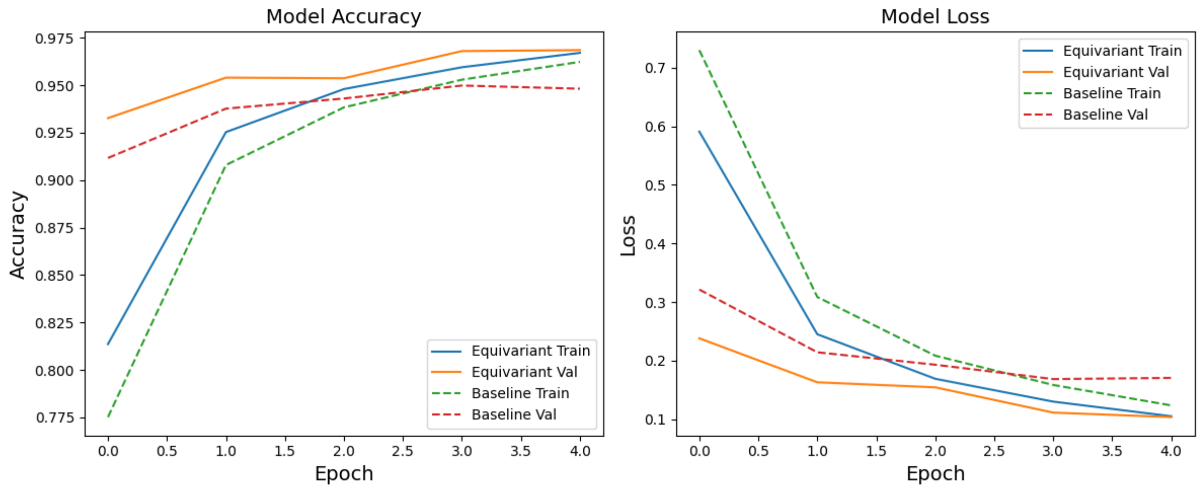

Figure 2: Training curves of both models.

2 Question B1

This analysis revisits the Higgs boson discovery in the diphoton ($H \to \gamma\gamma$) channel using a larger dataset provided by the ATLAS collaboration. The dataset corresponds to an integrated luminosity of 36.1 fb^{-1} from Run 2 proton-proton collisions at $\sqrt{s} = 13$ TeV.

The initial step involved accessing the data files hosted on the CERN Open Data portal using `uproot` and converting the relevant branches into an `awkward` array, subsequently processed using `pandas`. Events were required to contain at least two photons. The two photons with the highest transverse momentum (p_T) in each event were selected, and their four-momenta were constructed using the `vector` library to calculate the diphoton invariant mass, $m_{\gamma\gamma}$.

Several selection cuts were applied to isolate potential Higgs boson candidate events within the mass range $100 \leq m_{\gamma\gamma} \leq 160$ GeV. These cuts, adapted from the lectures and updated for the new data format, include:

- Leading photon $p_T > 50$ GeV and subleading photon $p_T > 35$ GeV.
- Both photons must satisfy the 'photon_isTightID' requirement.
- Both photons must pass the merged isolation criteria indicated by the 'photon_isTightIso' flag, replacing the separate cone-based cuts used previously.
- Both photons must satisfy $p_T/m_{\gamma\gamma} > 0.35$.

The efficiencies of these cuts were monitored during the selection process.

Signal modelling relied on simulated Monte Carlo (MC) samples provided by ATLAS Open Data, encompassing various Higgs production mechanisms (ggH, VBF, WH, ZH, ttH). These MC events, processed using the same kinematic and identification selections, were weighted according to their cross-sections, generated luminosities, and the target data luminosity (36.1 fb^{-1}). As instructed, the total expected signal yield from MC was scaled down by a factor of six before use in the final fit. A Crystal Ball plus Gaussian function was fitted to the weighted MC $m_{\gamma\gamma}$ distribution in the range 110-140 GeV using `iminuit` and `numba_stats` to determine the signal probability density function (PDF) shape; its parameters were fixed in the subsequent fit to data.

The background contribution in the data distribution after selection was modelled using a Bernstein polynomial (degree 4). A final unbinned maximum likelihood fit was performed on the selected data sample using `iminuit`. The fit model comprised the fixed signal PDF shape and the background Bernstein polynomial. The free parameters were the background shape parameters, the total number of background events N_b, and the signal strength parameter μ, defined relative to the scaled expected Standard Model yield. The value $\mu = 1$ corresponds to the scaled SM prediction.

A background-only fit (μ fixed to 0) was performed first, followed by a signal-plus-background fit allowing μ to float. The significance of the signal was estimated from the difference in the log-likelihood values ($\Delta \ln L$) between the two fits using Wilks' theorem, $Z = \sqrt{-2\Delta \ln L}$. A profile likelihood scan was performed to visualise the likelihood dependence on μ.

The final fit result is shown in Figure 3. A clear peak consistent with the Higgs boson mass is visible over the smoothly falling background shape. The Minuit fit output indicates a successful convergence with the best-fit signal strength $\mu = 0.99 \pm 0.28$. This value is in agreement with the Standard Model expectation. The profile likelihood scan, presented in Figure 4, confirms this, showing a well-defined minimum near $\mu = 1$. The significance calculated from the likelihood ratio between the best signal-plus-background fit and the background-only fit corresponds to $-2\Delta \ln L = 12.67$, yielding $Z = \sqrt{12.67} = 3.56\sigma$. This result provides strong evidence for the Higgs boson signal in this dataset, clearly improved from the $\sim 2\sigma$ indication seen in the smaller lecture sample. However, it is below the 5σ threshold for discovery.

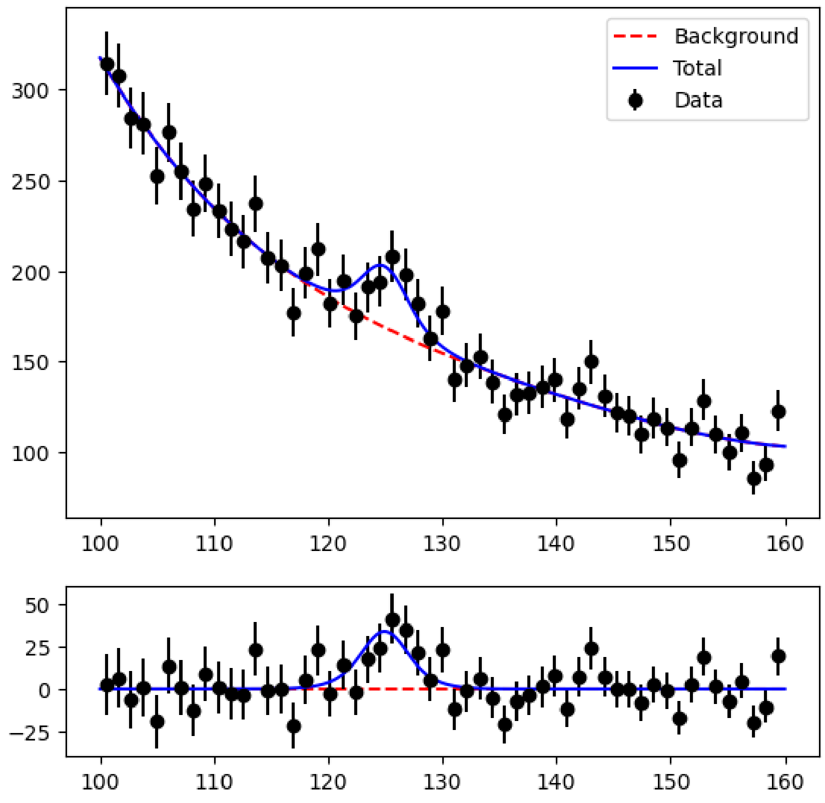

Figure 3: Fit to the diphoton invariant mass distribution after final selection. Data points are shown with statistical errors. The solid blue line represents the total signal-plus-background fit, while the dashed red line shows the background-only component. The lower panel displays the residuals.

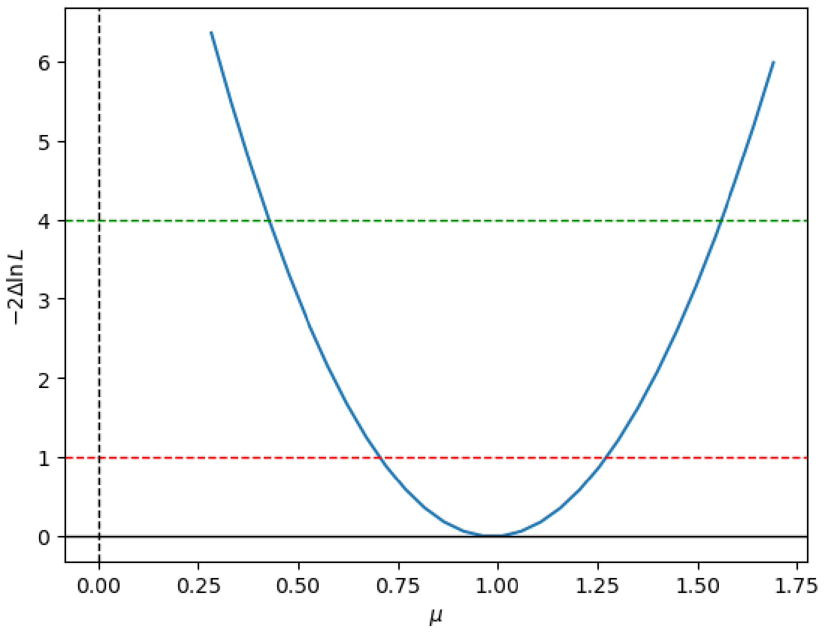

Figure 4: Profile likelihood scan for the signal strength parameter μ. The minimum corresponds to the best-fit value. Dashed lines indicate 1σ $(-2\Delta \ln L = 1)$ and 2σ $(-2\Delta \ln L = 4)$ uncertainty intervals.

3 Question C1

Generating samples from complex, high-dimensional probability distributions is a fundamental challenge in many scientific domains. Normalising Flows (NFs) represent a class of deep generative models designed specifically for this task, offering both efficient sampling and exact likelihood evaluation.

The core principle of a Normalising Flow is to learn an invertible and differentiable transformation $f : \mathcal{Z} \to \mathcal{X}$ that maps samples $\mathbf{z}$ from a simple base distribution $p_Z(\mathbf{z})$) to samples $\mathbf{x}$ from a complex target distribution $p_X(\mathbf{x})$. The transformation $\mathbf{x} = f(\mathbf{z})$ must be invertible, allowing for the reverse mapping $\mathbf{z} = f^{-1}(\mathbf{x})$. In practice, the Jacobian determinant of the transformation must be efficiently computable. This latter requirement is crucial for evaluating the probability density of a sample $\mathbf{x}$ under the target distribution using the change of variables formula:

$$p_X(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x})) \left| \det \left(\frac{\partial f^{-1}}{\partial \mathbf{x}} \right) \right| = p_Z(\mathbf{z}) \left| \det \left(\frac{\partial f}{\partial \mathbf{z}} \right) \right|^{-1}$$

The Box-Muller transformation provides a direct mapping from two independent uniform variables u_1, u_2 to two independent standard normal variables z_1, z_2 via the following equations:

$$z_1 = \sqrt{-2 \ln u_1} \cos(2\pi u_2) \tag{1}$$

$$z_2 = \sqrt{-2 \ln u_1} \sin(2\pi u_2) \tag{2}$$

where u_1, u_2 are drawn independently from $\mathcal{U}(0, 1)$ [2]. This was implemented as a Python function, `box_muller`, using NumPy for numerical operations. A set of $10,000$ samples was generated using this implementation. The resulting two-dimensional distribution was visualized using a hexagonal binning histogram, shown in Figure 5. To quantitatively verify that the marginal distributions of the generated

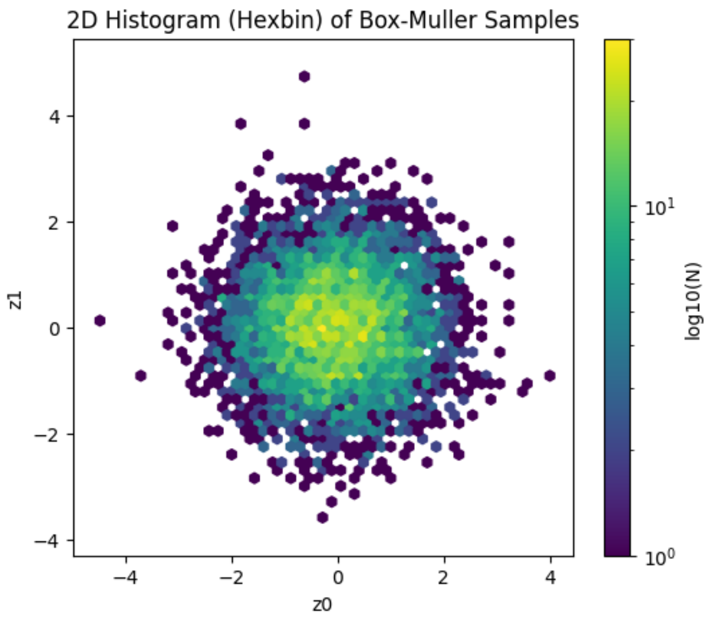

Figure 5: Hexagonal binning histogram of 10,000 samples generated using the Box-Muller implementation. The colour intensity corresponds to the logarithm of the number of samples per bin, showing peak density at the origin.

samples z_0 and z_1 follow the standard normal distribution $\mathcal{N}(0, 1)$, the two-sample Kolmogorov-Smirnov (KS) test was employed using `scipy.stats.kstest`. This test compares the empirical cumulative distribution function (ECDF) of the generated samples against the theoretical CDF of the standard normal distribution. The null hypothesis (H_0) is that the samples are drawn from the standard normal distribution.

The KS test yielded the following results for the marginal distributions:

- For z_0: KS statistic = 0.0143, p-value = 0.2559

- For z_1: KS statistic = 0.0181, p-value = 0.0750

Conventionally, a p-value greater than a significance level 0.05 indicates insufficient evidence to reject the null hypothesis. Since both p-values are greater than 0.05, we do not reject H_0 for either marginal distribution. This provides statistical support that the implemented Box-Muller function correctly generates samples whose marginal distributions are consistent with the standard normal distribution.

3.1 Uniform to Normal Flow

A Normalising Flow model was developed using TensorFlow and TensorFlow Probability to learn the mapping from the uniform base to the Gaussian target distribution. The architecture was built by composing several transformation layers, known as bijectors. A common and effective type of bijector employed here, is the coupling layer, specifically the RealNVP (Real-valued Non-Volume Preserving) formulation [3].

The theoretical basis of a coupling layer involves partitioning the input vector $\mathbf{z}$ into two sub-vectors, $\mathbf{z}_A$ and $\mathbf{z}_B$. The transformation acts differently on each part: one part (e.g., $\mathbf{z}_A$) is left unchanged (identity transformation), while the other part ($\mathbf{z}_B$) undergoes a transformation whose parameters depend only on the unchanged part $\mathbf{z}_A$. For RealNVP, this is an affine transformation:

$$\begin{aligned}\mathbf{z}'_A &= \mathbf{z}_A \\ \mathbf{z}'_B &= \mathbf{z}_B \odot \exp(s(\mathbf{z}_A)) + t(\mathbf{z}_A)\end{aligned}$$

Here, $s(\cdot)$ and $t(\cdot)$ are functions, implemented as neural networks, that compute element-wise scale and shift parameters based solely on $\mathbf{z}_A$. This structure is advantageous for Normalising Flows because it is easily invertible:

$$\begin{aligned}\mathbf{z}_A &= \mathbf{z}'_A \\ \mathbf{z}_B &= (\mathbf{z}'_B - t(\mathbf{z}'_A)) \odot \exp(-s(\mathbf{z}'_A))\end{aligned}$$

Furthermore, the Jacobian matrix of this transformation is triangular, meaning its determinant is simply the product of the diagonal elements. For the forward transformation shown, the diagonal elements are ones for the $\mathbf{z}_A$ part and $\exp(s(\mathbf{z}_A))$ for the $\mathbf{z}_B$ part. Thus, the logarithm of the absolute determinant (log-det-Jacobian) is simply the sum of the scale parameters: $\sum s(\mathbf{z}_A)$. This tractability of both inversion and Jacobian determinant calculation is essential for NF training and density evaluation.

In our implementation, the conditioner networks computing s and t were small MLPs with two hidden layers and ReLU activations. To ensure that all dimensions are eventually transformed over the full flow, coupling layers that transform complementary parts of the input are interleaved, often facilitated by permuting the dimensions between layers. The complete flow was constructed as a sequence of these RealNVP coupling layers and permutations.

The model was trained by minimizing the negative log-likelihood, which is equivalent to minimizing the Kullback-Leibler (KL) divergence between the distribution produced by mapping the base distribution through the flow, $f(p_Z)$, and the target distribution p_X. The loss function used was:

$$\mathcal{L} = -\mathbb{E}_{\mathbf{u} \sim p_Z(\mathbf{u})} \left[\log p_X(f(\mathbf{u})) + \log \left| \det \left(\frac{\partial f}{\partial \mathbf{u}} \right) \right| \right]$$

Hyperparameter optimization for the flow architecture (number of coupling layers, conditioner network size) and training parameters (learning rate, batch size) was conducted using Optuna [4]. The best hyperparameters found involved 6 coupling layers and conditioner hidden layer sizes of [512, 512]. A final model with these parameters was trained for 5000 steps using the Adam optimizer.

The trained Normalising Flow model was evaluated by sampling from the uniform base distribution and applying the learned transformation f. Figure 6 compares the input uniform samples and the corresponding output samples from the flow.

The output distribution qualitatively looks Gaussian.

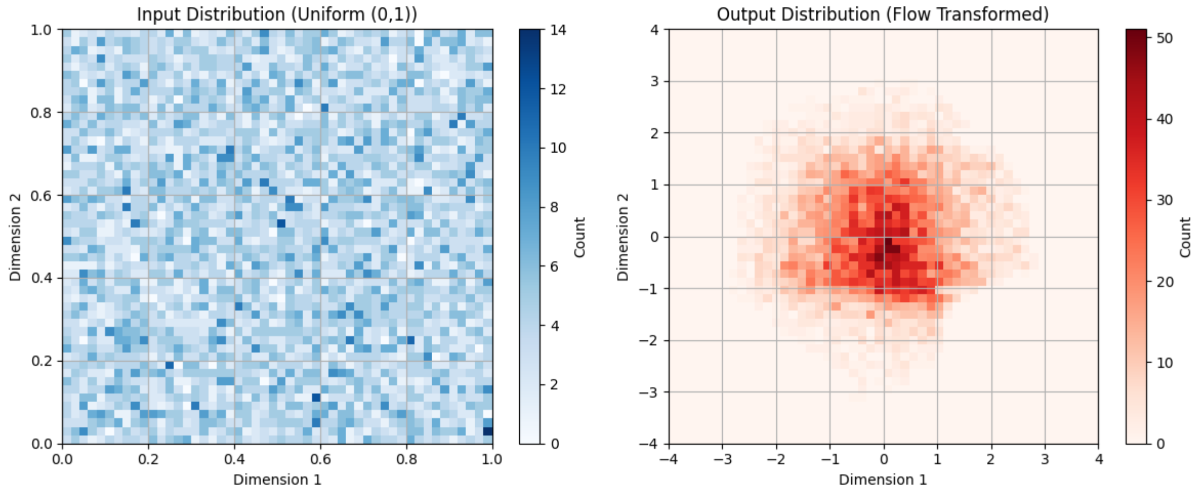

Figure 6: Comparison of distributions. Left: Input uniform base samples. Right: Output samples after transformation by the trained Normalising Flow.

3.2 Normal to Uniform Flow

The task was then inverted: learning the mapping from the Gaussian distribution $p_X(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ back to the uniform distribution $p_Z(\mathbf{u}) = \mathcal{U}([0,1]^2)$ using the same trained flow architecture but optimizing the reverse loss:

$$\mathcal{L}_{rev} = -\mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x})}\left[\log p_Z(f^{-1}(\mathbf{x})) + \log\left|\det\left(\frac{\partial f^{-1}}{\partial \mathbf{x}}\right)\right|\right]$$

This task presented significant challenges due to the mismatch in the support of the distributions. The target distribution p_Z is defined only on the finite interval $[0,1]^2$, while the base distribution p_X has infinite support on $\mathbb{R}^2$. Directly applying the flow's inverse f^{-1} to Gaussian samples $\mathbf{x}$ often results in values outside the required $[0,1]^2$ range, yielding undefined log-probabilities $\log p_Z(f^{-1}(\mathbf{x}))$ and causing training instability (NaN losses).

To address this domain mismatch, the standard approach is to augment the bijector chain. A sigmoid function, acting element-wise, maps $\mathbb{R}^2$ to $(0,1)^2$. Its inverse, the logit function, maps $(0,1)^2$ to $\mathbb{R}^2$. Incorporating these allows the core flow (e.g., RealNVP chain g) to operate on the unbounded space $\mathbb{R}^2$, while ensuring the final output lies within the required domain of the uniform distribution. Specifically, for the reverse direction ($p_X \to p_Z$), the full bijector f^{-1} effectively becomes $f^{-1} = \text{sigmoid} \circ g^{-1}$. The loss calculation must use the corresponding base distribution (Gaussian) and the inverse log-det-Jacobian of the full chain including the sigmoid.

For the purpose of this exercise, we attempted to train the same flow parameters f using the reverse loss objective. This required transforming the base distribution (p_X = Gaussian) through the inverse flow f^{-1} to match the target (p_Z = Uniform). Due to the support mismatch, the log-probability term $\log p_Z(f^{-1}(\mathbf{x}))$ becomes problematic. Standard NFs typically map a bounded base to an unbounded target by incorporating the inverse of the sigmoid (logit) at the start of the forward flow: $f = g \circ \text{logit}$. Attempting to train the reverse direction without such initial transformation from the unbounded Gaussian input encountered difficulties.

3.3 Rejection Sampling

Rejection sampling provides a method to obtain samples that are guaranteed to follow the target distribution, $p(x)$, using proposals generated by an approximate distribution, $q(x)$, such as that implicitly defined by the trained normalizing flow. The method relies on finding a constant M such that $p(x) \leq Mq(x)$ for all x. A proposal sample x generated from $q(x)$ is accepted with probability $\alpha = p(x)/(Mq(x))$. While the generated samples are exact, the efficiency depends on the acceptance rate, which reflects how well $q(x)$ approximates $p(x)$. A higher acceptance rate implies a better approximation by the flow.

Functions `rejection_sample_forward` and `rejection_sample_backward` were implemented using the trained flow model f. For the forward direction ($\mathcal{U} \to \mathcal{N}$), the base uniform distribution $p_Z(\mathbf{u})$ generates proposals $\mathbf{u}$, which are mapped to $\mathbf{x} = f(\mathbf{u})$. The proposal density is $q(\mathbf{x}) = p_Z(\mathbf{u})|\det J_f(\mathbf{u})|^{-1}$. The target density is $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$. For the backward direction ($\mathcal{N} \to \mathcal{U}$), proposals $\mathbf{z}$ are drawn from the target normal distribution $p_X(\mathbf{z})$ and mapped to $\mathbf{u} = f^{-1}(\mathbf{z})$. The proposal density is $q(\mathbf{u}) = p_X(\mathbf{z})|\det J_{f^{-1}}(\mathbf{z})|^{-1}$. The target density is $p(\mathbf{u}) = \mathcal{U}(\mathbf{u}|[0,1]^2)$.

The acceptance rates were compared for the untrained model and the model after training. Figure 7 displays the distributions of accepted samples after training. For the forward direction, the acceptance rate improved significantly from a low value 54.76 % to 65.15% after training, indicating the flow learned to approximate the Gaussian somewhat. The backward direction, however, achieved a high acceptance rate of 100.0% even for the trained model. This is due to the presence of the sigmoid, all values will be in the domain [0,1].

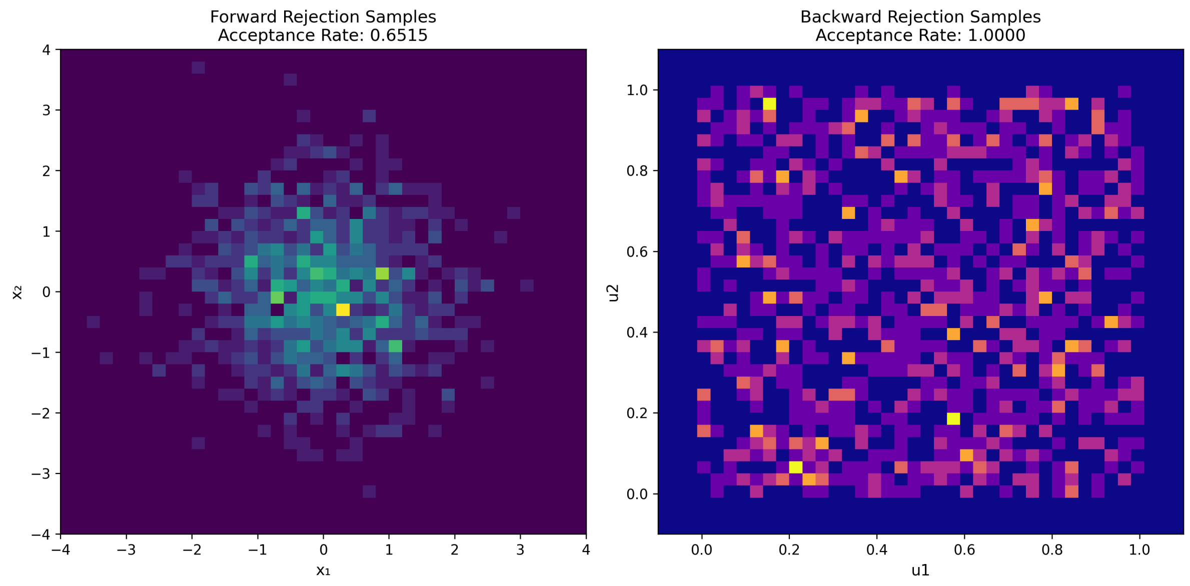

Figure 7: Distributions of 1000 samples obtained via rejection sampling using the trained Normalising Flow model. Left: Forward direction ($\mathcal{U} \to \mathcal{N}$) samples visually approximating a Gaussian. Right: Backward direction ($\mathcal{N} \to \mathcal{U}$) samples visually approximating a uniform distribution.

3.4 Lattice Field Theory Action Evaluation

The final task involved evaluating the action for a two-dimensional scalar field theory on a discrete lattice. The Euclidean action for a ϕ^4 theory is given by:

$$S[\phi] = \int d^2x \left[\frac{1}{2}(\partial_\mu\phi)(\partial^\mu\phi) + \frac{1}{2}m^2\phi^2 + \frac{\lambda}{4!}\phi^4\right]$$

On a discrete $L \times L$ lattice (spacing $a = 1$), using a finite difference approximation for the kinetic term, the action is a sum over sites (i, j):

$$S[\phi] \approx \sum_{i,j} \left[\frac{1}{2}\sum_{\mu=1,2}(\phi_{i+\hat{\mu},j} - \phi_{i,j})^2 + \frac{1}{2}m^2\phi_{i,j}^2 + \frac{\lambda}{4!}\phi_{i,j}^4\right]$$

A Python function, `create_phi4_potential`, calculated this discrete action $S[\phi]$ for a given lattice configuration ϕ (using parameters $m^2 = -1.0, \lambda = 1.0$). The probability of a configuration ϕ is proportional to the Boltzmann factor $e^{-S[\phi]}$. The function was tested by evaluating $e^{-S[\phi]}$ for several random field configurations.

As shown in Figure 8, four random configurations were generated on a 10×10 lattice with increasing overall magnitude scales. The calculated action values clearly increase with the field magnitude and apparent fluctuations, ranging from $S \approx 84$ for the lowest magnitude field to $S \approx 1400$ for the highest. This trend is shown in Figure 9, where the total action increases non-linearly with the field magnitude scale. This behaviour is consistent with the definition of the action, where both kinetic (gradient-squared)

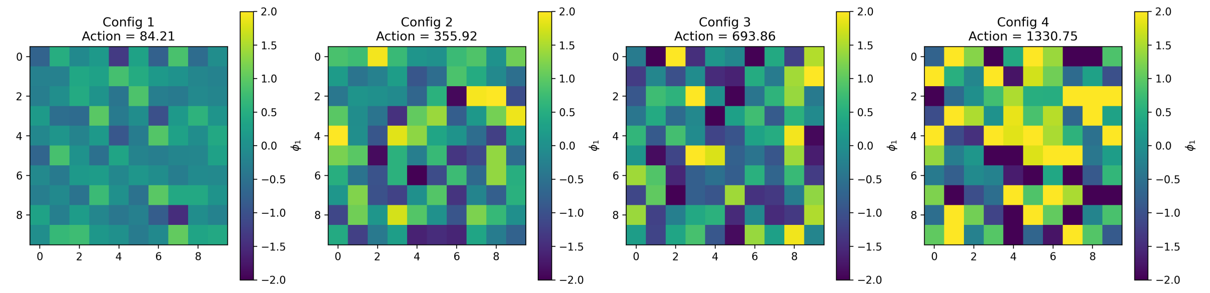

Figure 8: Visualization of four random scalar field configurations on a 10x10 lattice (displaying one component ϕ_1) and their corresponding computed ϕ^4 action values ($m^2 = -1.0, \lambda = 1.0$). Configurations generated with increasing variance show correspondingly larger action values.

and potential (ϕ^2, ϕ^4) terms contribute positively to S (for $\lambda > 0$) and grow with field amplitude. Consequently, the Boltzmann suppression factor $e^{-S[\phi]}$ rapidly decreases for configurations with larger field values or fluctuations, indicating lower probability.

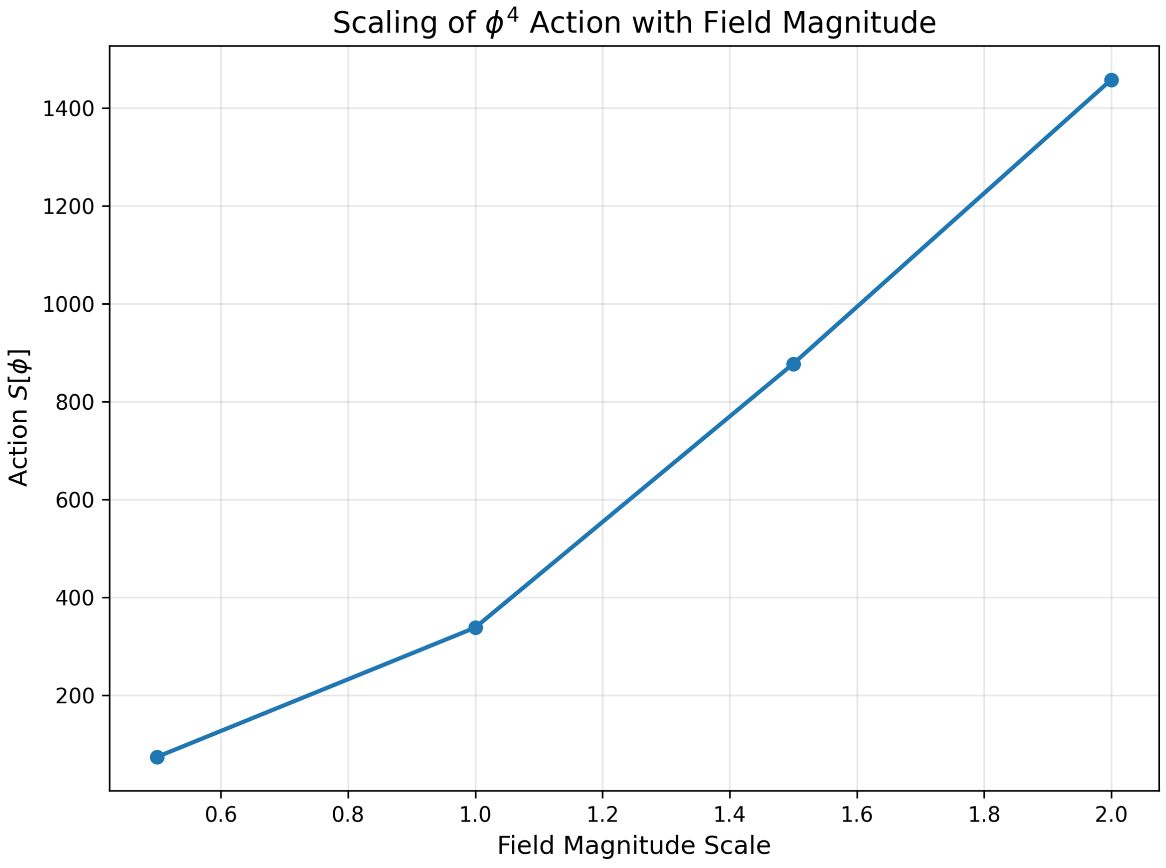

Figure 9: Total action $S[\phi]$ increases with the standard deviation (magnitude scale) used for generating random field configurations.

References

[1] Deng L. The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine. 2012;29(6):141-2.

[2] Freiden A, Herzog TN. Generating normal random deviates in APL. SIGAPL APL Quote Quad. 1979 Mar;9(3):49–51. Available from: https://doi.org/10.1145/586058.586065.

[3] Dinh L, Sohl-Dickstein J, Bengio S. Density estimation using Real NVP; 2017. Available from: https://arxiv.org/abs/1605.08803.

[4] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework; 2019. Available from: https://arxiv.org/abs/1907.10902.

AI Declaration

All code was created in VSCode with GitHub copilot enabled. ChatGPT and ClaudeAI was used to help debug and create the Normalising Flows in C1. ChatGPT was also used to proofread this report.