Computing and Software for Big Science (2025) 9:1
https://doi.org/10.1007/s41781-025-00130-x

RESEARCH

Fast Simulation of the Zero Degree Calorimeter Responses with Generative Neural Networks

Maksymilian Wojnar[1] · Emilia Majerz[1] · Witold Dzwinel[1]

Received: 24 September 2024 / Accepted: 6 January 2025

Abstract
Applying machine learning methods to high-energy physics simulations has recently emerged as a rapidly developing area. A prominent example is the Zero Degree Calorimeter (ZDC) simulation in the ALICE experiment at CERN, where substituting the traditional computationally extensive Monte Carlo methods with generative models radically reduces computation time. Although numerous studies have addressed the fast ZDC simulation, there remains significant potential for innovations. Recent developments in generative neural networks have enabled the creation of models capable of producing high-quality samples indistinguishable from real data. In this paper, we apply the latest advances to the simulation of the ZDC neutron detector and achieve a significant improvement in the Wasserstein metric compared to existing methods with a low generation time of 5 ms per sample. Our focus is on exploring novel architectures and state-of-the-art generative frameworks. We compare their performance against established methods, demonstrating competitive outcomes in speed and efficiency. The source code and hyperparameters of the models can be found at https://github.com/m-wojnar/zdc.

Keywords Generative neural networks · Fast simulations · High-energy physics · Zero degree calorimeter

Introduction

Numerical simulations using high-performance computing have become integral to scientific research, forming the third pillar of science alongside theory and experiments. However, modeling these systems often demands extensive computational resources and memory, posing significant challenges. To mitigate these issues, researchers employ surrogate models—simplified versions of more complex simulations that approximate the behavior of the system [1]. Creating these surrogates has been a standard research practice to balance accuracy and computational efficiency. Recently, the advent of machine learning (ML) and neural networks has revolutionized this approach. These networks, with their computational universality, have demonstrated remarkable success in approximating complex systems, enabling faster and more efficient physical simulations without directly modeling the experiments.

The application of ML in physical simulations can take various forms, such as supervised learning, incorporating a physical loss term, or leveraging differentiable numerical simulations [2]. Generative neural networks have recently shown promise as a viable alternative to traditional methods for fast simulations. These networks are capable of generating high-quality samples that are indistinguishable from real data, significantly reducing computational costs. Using various frameworks, such as autoencoders, generative adversarial networks (GAN), and normalizing flows (NF), these models have also shown their potential in high-energy physics [3–5].

The Zero Degree Calorimeter (ZDC) [6] in the ALICE experiment at CERN plays a crucial role in measuring particle showers to determine the centrality of collisions [7]. Traditionally, the simulation of ZDC responses relies on GEANT [8], a Monte Carlo toolkit designed to model the trajectory of particles. Although this method provides precise results, it comes with a cost of significant time and computational resources needed to perform the simulation. Since the demand for calorimeter simulations grows [9], the

✉ Emilia Majerz
majerz@agh.edu.pl

Maksymilian Wojnar
mwojnar@agh.edu.pl

Witold Dzwinel
dzwinel@agh.edu.pl

1 AGH University of Krakow, Kraków, Poland

motivation to develop more efficient techniques is stronger than ever.

This paper explores the application of state-of-the-art (SOTA) generative neural network architectures to the simulation of the ZDC neutron detector. We examine novel architectures and frameworks, such as the Vision Transformer (ViT) and MLP-Mixer, which have not yet been extensively applied to this domain. We also consider modern generative frameworks, such as vector quantization (VQ), NFs, and diffusion models. Our aim is to assess the performance of these models against established methods, highlight their potential benefits, and identify challenges. Figure 1 schematically shows the relationship between GEANT (simulation model) and generative neural networks (surrogate models). The data generated by the Monte Carlo simulator are used to train a faster model, which could eventually be used interchangeably with the traditional simulator.

Our contribution includes:

- Implementation, evaluation, and confrontation of various SOTA neural network architectures and generative frameworks with comparison to existing methods on the task of fast simulation of the ZDC neutron detector.
- Achieving a competitive to SOTA Wasserstein metric score of 3.15 in the ZDC neutron detector simulation task using a diffusion model, with a low generation time of 5 ms per sample (orders of magnitude faster than Monte Carlo methods [10]). Until now, SDI-GAN [11] achieved the best fast simulation score of 4.5. Alternative approaches have shown results as low as 1.2, but with a generation time of 109 ms per sample [12].
- NF, as the second-best performing model presented in this paper, also achieved a strong score of 4.11, with the potential to improve the time of the computations in the future, and potentially even surpass the diffusion model in the performance vs time trade-off.
- Providing open-source code and detailed hyperparameter settings for reproducibility and further research.

The next section provides a review of the literature on fast ZDC simulations, Sect. "Dataset analysis" focuses on the analysis of the dataset, and Sect. "Background" briefly describes the architectures and generative frameworks we use. Then, in Sects. "Training setup" and "Experimental results", we present the methodology and results of our research, and finally we conclude the paper in Sect. Conclusions.

Related Works

Generative neural networks have been extensively applied to high-speed simulations in various CERN projects since 2017. Early research introduced applications of GANs [5, 13–17], autoencoders [3, 18–20], and NFs [4, 19–21] (Table 1).

Fast supervised ZDC simulation, with its two-dimensional structure and high resource requirements, is the perfect task for generative neural networks. A notable advancement came in 2021 with the introduction of the end-to-end Sinkhorn Autoencoder (e2e SAE) [22]. This model enhances standard autoencoder designs by incorporating a noise generator neural network, allowing distinct classes to be encoded separately in the latent space. The e2e SAE demonstrated competitive performance in capturing critical physical relations, such as collision centers, crucial for realistic data simulation.

A significant challenge in the ZDC simulations is the varying diversity of the calorimeter responses for different particles. In 2023, a study addressed the limited diversity of GAN-generated samples [11]. The proposed SDI-GAN attempts to solve mode collapse in conditional GANs by

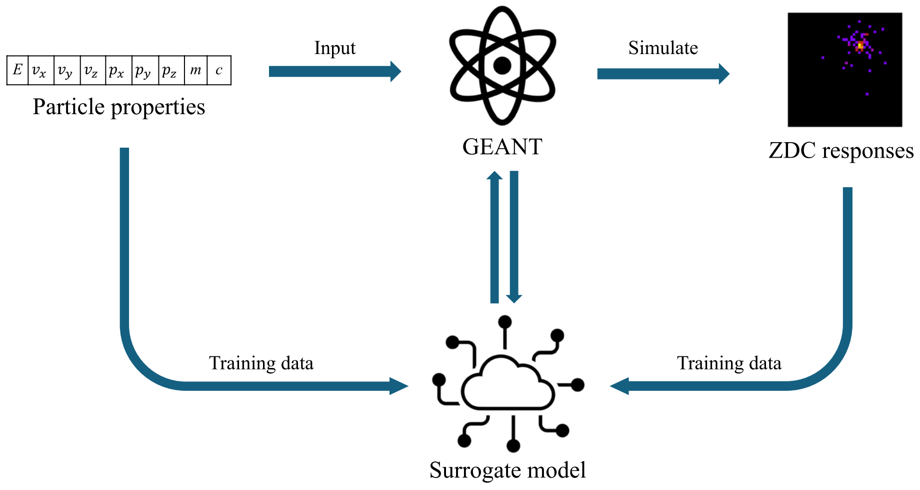


Fig. 1 Interaction between the simulation model, surrogate model, and its training process

Table 1 Chronological overview of papers applying generative neural networks for fast simulations in high-energy physics. Note that VQ models are not used in any papers

Year	Reference	GAN	Autoencoder	NF	Diffusion
2017	[5]	✓			
2018	[17]	✓			
2018	[15]	✓			
2018	[16]	✓			
2019	[14]	✓			
2019	[13]	✓			
2020	[3]		✓		
2021	[4]			✓	
2021	[21]			✓	
2021	[18]		✓		
2021	[22]		✓		
2022	[19]		✓	✓	
2023	[20]		✓	✓	
2023	[11]	✓			
2023	[10]	✓	✓		
2024	[12]				✓
2024	[23]		✓		
2024	[25]	✓			

incorporating a regularization factor into the loss function, encouraging the generator to explore new data modes for particles with diverse responses, while maintaining consistency for others.

The 2023 study evaluated basic variational autoencoders (VAE) and GAN models for the simulation of the ZDC neutron detector [10]. They improved the GAN framework with an auxiliary regularization network, additional postprocessing steps, and proposed a neural network classifier to filter inputs that do not trigger the neutron detector's response.

A different approach proposed in 2024 in [12] leverages a diffusion model, which achieves high-fidelity results and provides control over the simulation quality by adjusting the number of denoising steps. The authors note a high generation time and propose a compromise using a latent diffusion model to accelerate the simulation process.

The 2024 study [23] employs CorrVAE [24] to encode different aspects of an object into two separate hidden variables. This is achieved by having separate encoders for properties and objects. In addition, the correlation between properties is identified and processed by the mask pool layer, which consolidates the relevant information into a bridging latent variable.

The authors of [25], presented in 2024, integrate SDI-GAN with an auxiliary regressor and intensity regularizator to model the responses of the ZDC proton detector. This work is a pioneering application of generative neural networks for fast ZDC proton simulation.

Table 1 summarizes the reviewed works in chronological order, indicating the models used. Despite these advances, the literature does not fully explore recent developments in generative neural networks. From our literature review, we found that the ViT [26] and MLP-Mixer [27] architectures have not yet been applied to this domain, and modern generative frameworks, such as VQ and diffusion, are rarely utilized. Furthermore, many existing solutions do not incorporate physical loss terms.

Dataset Analysis

The ZDC consists of four calorimeters, two for proton detection and two for neutron detection. Two sets of two detectors (proton—ZP and neutron—ZN) are placed on each side of the interaction point IP2. These devices are designed as "spaghetti calorimeters", comprising stacked heavy metal plates interspersed with a matrix of quartz fibers. The Cherenkov light emitted by charged particles passing through these fibers is detected and counted by photomultipliers (PMs). The schematic diagram of the ALICE central barrel including the ZDC can be found in [28], while the image of the ZDC is included in [29].

The output of every second fiber is sent to one of the PMs. The remaining ones are divided into four groups (upper-left, upper-right, lower-left, lower-right) and sent to four separate PMs. Following this scheme, each detector response can be processed to produce five values, denoting the sums of the signal from each channel. These five values are further denoted as *channels*. The dataset, based on GEANT simulations, consists of two parts: 9-dimensional particle properties (which serve as conditional variables) and ZN responses of dimensions 44×44 (which can be interpreted as images in generative models; the task is then to produce images with $44 \times 44 = 1936$ pixels). Figure 2 presents histograms of particle features. Note that momenta and mass formally have units of $\frac{\text{GeV}}{c}$ and $\frac{\text{GeV}}{c^2}$, respectively, where c is the speed of light in a vacuum. However, in this dataset, the values are normalized to GeV for simplicity.

The dataset generated with the GEANT4 Monte Carlo toolkit contains 306,780 samples, with responses having at least 10 photons. Among all the examples, 99,695 have a different sum of photons, and there are only 1805 unique particles, indicating that the dataset might not cover the entire space of possible particles. Furthermore, there is a noticeable imbalance in the dataset—Fig. 3 illustrates each particle's contribution, visualized by the circle's size. Overall, the dataset includes 21 different types of particles. The observed dataset limitations may impact the results; for example, the model may generate low-quality results for the underrepresented particles. This might be mitigated by

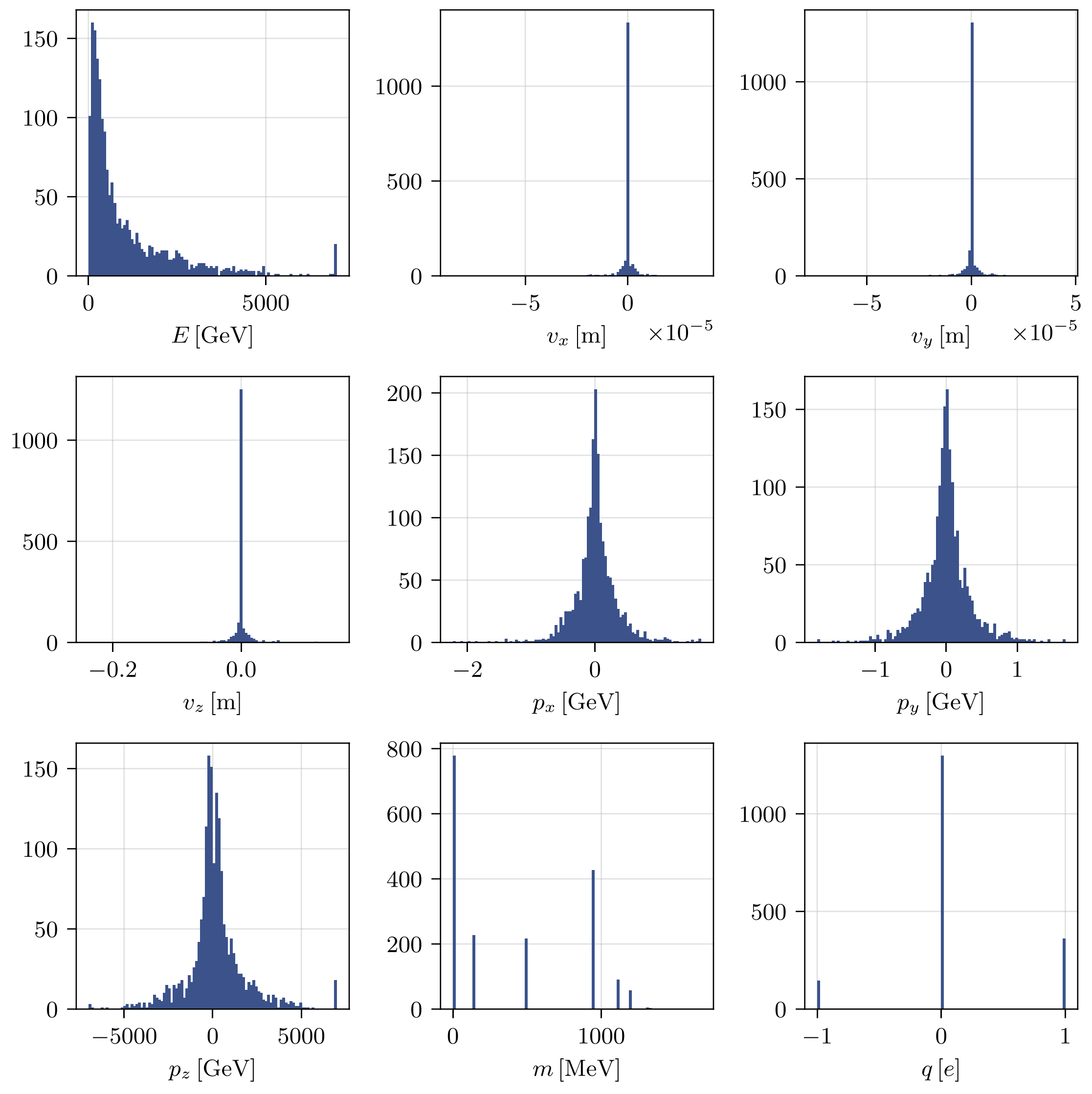


Fig. 2 Histograms of particle features. E stands for energy, v for primary vertex positions, p for momenta, m for mass, and q for charge

generating more data to make the dataset balanced, but since this task requires expert knowledge, it is out of the scope of this paper. Also, traditional methods for dealing with imbalanced datasets do not apply in our case, due to the high level of imbalance.

Figure 4 shows dataset visualizations made using t-distributed stochastic neighbor embedding (t-SNE). Particle features were normalized prior to generating the embeddings and only the unique particles were used for visualization, thus there are 1805 points. The t-SNE embedding separates the particles into four clusters. The cluster on the left contains negatively charged particles, the cluster on the right—positively charged, and the other two are neutral. Note that the most frequently represented γ particles have their own cluster. The figure uses colors to represent the six most common particle types, while all other types are depicted in a uniform color (yellow). Particle proportions may be different than in Fig. 3, as this visualization shows only unique feature vectors. The visualization reveals the structure of the particles, highlighting separation by the charge and mass of the particles.

In an ZDC simulation, the varying diversity of the detector responses plays an important role, i.e., for some particles, the detector gives consistent responses in different independent runs. In contrast, for others they are diverse, as illustrated in the Monte Carlo simulations shown in Fig. 5. The consistency regards the position of the center of the shower, as well as the number of generated photons.

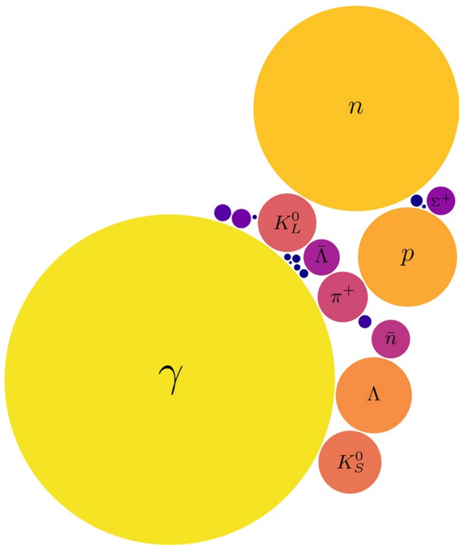


Fig. 3 Diagram depicting the distribution of particles in the dataset

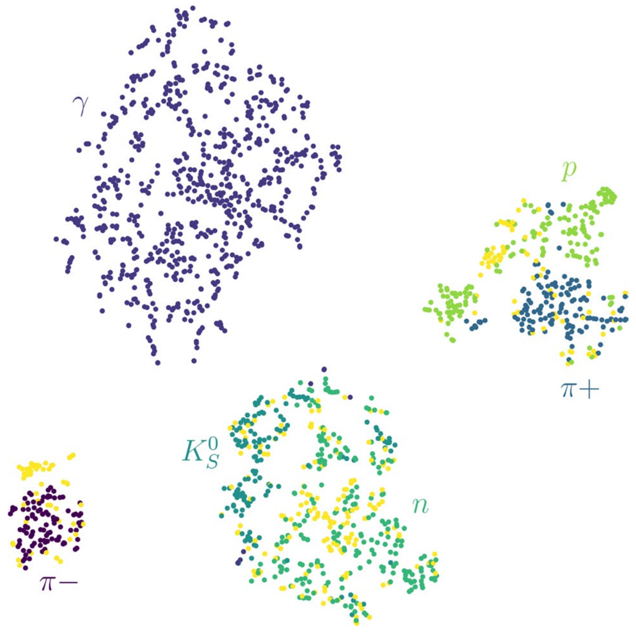


Fig. 4 Unique particle features visualization using t-SNE. The colors denote the types of the particles

Background

This section outlines the theory and key characteristics of the generative neural networks discussed in this paper. Details of the neural network architectures employed in this study can be found in Appendix C.

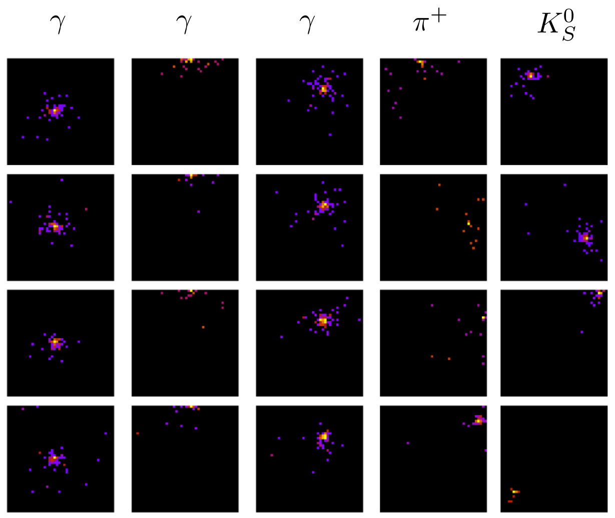


Fig. 5 Example ZDC neutron detector simulations generated with GEANT software. The columns represent responses to different particle types, labeled above each column for clarity. The rows show independent runs for the same particles. This figure stresses the diversity of the detector responses, as the first three are much more consistent than the other two

Autoencoders

The classical autoencoder framework consists of an encoder and a decoder, which, respectively, map the input data to the latent space and reconstruct the data from this space. VAE [30] extends the model with variational inference of latent variables to enable sampling from the model.

Another variation of the autoencoder is the supervised autoencoder [31]. As stated in [32], the supervised autoencoder can be viewed as a way to incorporate the physical loss term into the model [2]. These models transform the input into conditional variables (e.g., particle parameters in the ZDC) and then reconstruct the original data. Their loss function is

$$\mathcal{L}(\theta, \phi; x, c) = ||D_\phi(E_\theta(x)) - x||_2^2 + \beta||E_\theta(x) - c||_2^2, \qquad (1)$$

where E_θ is the encoder with parameters θ, D_ϕ is the decoder with parameters ϕ, c represents the conditional variables, and β is a regularization weight.

VAEs constrain the hidden space to match a specific distribution, which can limit model expressiveness and force conditional variables onto a single manifold in the latent space [22]. In addition, the common Gaussian distribution can impair the linear separability of complex data and does not support sparse latent representations via ReLU activation [33]. To address this, autoencoders with noise generators do not regularize the hidden space but preserve generative capabilities through an additional neural network—the noise generator. This network maps conditional variables to the latent space. e2e SAE combines reconstruction loss $\mathcal{L}_{rec}$, Sinkhorn loss $\mathcal{L}_S$ aligning noise generator and encoder outputs, and a regularization term $\mathcal{L}_{reg}$:

$$\begin{aligned}\mathcal{L}(\theta, \phi, \psi; x, c) = &\mathcal{L}_{rec}(\theta, \phi; x) + \\ &\beta \mathcal{L}_S(\theta, \psi; x, c) + \\ &\gamma \mathcal{L}_{reg}(\theta, \phi, \psi),\end{aligned} \tag{2}$$

where ψ are noise generator parameters, β is the Sinkhorn loss weight, and γ is the regularization weight. The original e2e SAE uses a complex Laplacian pyramid loss combined with $l2$ loss for reconstruction, where $l2$ loss minimizes the mean squared error between predicted and target values (i.e., $||y_{\text{pred}} - y_{\text{target}}||_2^2$). However, for ZDC responses, only $l2$ is used, omitting the regularization term.[1] For comparison, we propose an autoencoder with a noise generator trained with the $l2$ loss rather than the Sinkhorn loss to align the outputs of the noise generator and the encoder. This leads to a simpler formulation that requires fewer computational resources.

As well-established models, autoencoders are favored for ZDC simulations because of their ease of training and robust mathematical basis. On the other hand, classic autoencoders have difficulty reproducing the input faithfully and suffer from blurred images.

Generative Adversarial Networks

GANs [34] consist of two competing neural networks: the generator and the discriminator. The generator aims to produce realistic data, while the discriminator strives to distinguish between real and generated samples. The training objective is

$$\begin{aligned}\min_{G_\phi} \max_{D_\theta} &\mathbb{E}_{x \sim p(x)}[\log D_\theta(x)] + \\ &\mathbb{E}_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))],\end{aligned} \tag{3}$$

where G_ϕ is the generator parameterized by ϕ and D_θ the discriminator parameterized by θ.

SDI-GAN [11] adjusts the diversity of generated samples based on the randomness of the original particle responses. The diversity measure is defined as $f_{div}(c) = \sum_{i,j} \sqrt{\sum_t (x_{tij} - \mu_{ij})^2 / |X_c|}$, where x_{tij} is the pixel value at coordinates i, j for sample t, and μ_{ij} is the mean of these pixel values for the condition c. The regularization term scales the diversity proportionally to the data variance:

$$\mathcal{L}_{\text{SDI}}(\theta, \phi; x, c) = f_{div}(c) \cdot \left(\frac{d_I(G_\phi(c, z_1), G_\phi(c, z_2))}{d_z(z_1, z_2)} \right)^{-1}, \tag{4}$$

where z_1 and z_2 are two different latent codes, d_z is a latent space distance metric, and d_I measures the dissimilarity between two images conditioned on c, using image embeddings generated by the penultimate layer of the discriminator.

The authors of [10] enhance GANs with an auxiliary regressor and postprocessing step to improve the quality of generated ZDC responses. The regressor, pretrained to predict peak photon count coordinates, adds a physical loss component during training. A postprocessing step further refines the output images by scalar multiplication to optimize metric values that distinguish the generated from actual data distributions. An alternative method to ensure the global consistency of GAN-generated outputs (e.g., the position of the photons peak) involves incorporating $l1$ or $l2$ loss into the generator [35], where $l1$ loss minimizes the mean absolute error between predicted and target values (i.e., $|y_{\text{pred}} - y_{\text{target}}|$). This approach is intended to faithfully recreate the global relationships within the image, in contrast to adversarial loss, which focuses on capturing local details.

For a long time, GANs have been the SOTA generative models, recognized for their ability to quickly generate high-quality images [36]. Nevertheless, they are difficult to train, struggle to maintain adequate diversity in their outputs, and often suffer from mode collapse.

Vector Quantization

Vector Quantized Variational Autoencoder (VQ-VAE) [37] introduces discrete latent representations through VQ. Quantized vectors z_q are selected as nearest neighbors of the encoder's output from a codebook $\mathcal{Z}$:

$$z_q(x) = \underset{z \in \mathcal{Z}}{\operatorname{argmin}} ||E_\theta(x) - z||_2, \tag{5}$$

where $||x||_2$ denotes the $l2$ norm of the vector x. The decoder reconstructs input based on the quantized vectors, thus a straight-through gradient estimator is used during the training. The VQ-VAE loss function includes reconstruction loss, codebook alignment loss, and commitment loss:

$$\begin{aligned}\mathcal{L}(\theta, \phi, \mathcal{Z}; x) = &\mathbb{E}[\log p_{\theta,\phi}(x|z_q(x))] + \\ &|| \text{sg}[E_\theta(x)] - z_q(x)||_2^2 + \\ &\beta ||E_\theta(x) - \text{sg}[z_q(x)]||_2^2,\end{aligned} \tag{6}$$

where sg denotes a “stop gradient” operator, preventing updates for selected parts of the network. The second training phase aims to obtain a learnable prior. Since the VQ-VAE encoder generates discrete outputs, autoregressive discrete models, such as the PixelCNN model [38] or transformers, are employed for this task.

[1] https://gitlab.cern.ch/swenzel/zdcfastsim/.

VQ-GAN [39] enhances VQ-VAE by incorporating patch-based adversarial and perceptual [40] terms into the loss function. VQ addresses VAE posterior collapse but introduces codebook collapse, where embeddings are underutilized. Mitigation strategies include the exponential moving averages (EMA) codebook update [37], a linear projection for a comparison of vectors, and $l2$ normalization for cosine similarity lookup [41].

VQ models enable the generation of high-fidelity images with a more interpretable latent space. However, their main drawbacks include the complex training procedure and the potential for codebook collapse, leading to inefficient learning and reduced output quality.

Normalizing Flows

NFs model complex probability distributions by stacking multiple simple, invertible transformations. Each transformation, denoted as f_i, progressively maps a simple base distribution (e.g., Gaussian) to a complex target distribution. The transformation is given by $x = (f_n \circ f_{n-1} \circ \cdots \circ f_1)(z)$, where x is the input data and z the latent representation. The NF framework allows the exact data likelihood computation using the change of variables formula:

$$\begin{aligned} p_x(x) &= p_z(f^{-1}(x)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right| \\ &= p_z(z) \left| \det \frac{\partial f(z)}{\partial z} \right|^{-1}, \end{aligned} \tag{7}$$

where $x = f(z)$, $p_z(z)$ is the base density, $p_x(x)$—the target, and $\frac{\partial f(z)}{\partial z}$ is the Jacobian of f. For NFs, the density estimation in the log domain can be written as

$$\log p_x(x) = \log p_z(z) + \sum_{i=1}^{n} \log \left| \det \frac{\partial f_i}{\partial z_{i-1}} \right|^{-1}, \tag{8}$$

where z_{i-1} is the latent variable at the $(i-1)$th stage.

One type of frequently used transformations in the scope of NFs is the autoregressive models, where the value of x_d in the next autoregressive layer is conditioned on $x_1, x_2, \cdots, x_{d-1}$ values in the previous (d—currently processed dimension). In the case of autoregressive flows, each Jacobian $\frac{\partial f_i}{\partial z_{i-1}}$ is a lower triangular matrix, which means that its determinant can be computed as a product of its diagonal elements. Autoregressive flows face a trade-off between the efficiency of sampling and training. For example, Masked Autoregressive Flow (MAF) [42] enables fast training but is slow during the sampling, as it requires sequential processing. Conversely, Inverse Autoregressive Flow (IAF) [43] allows for fast sampling at the cost of slower training. The efficient implementation of the faster pass is given by MADE [44] blocks, which allow for computing all parameters of the transformation simultaneously, while preserving the autoregressive property.

NFs are elegant models, which offer stable training due to the minimization of the exact negative log-likelihood. However, their training requires a significant amount of resources, as they usually have many parameters, since there is no dimensionality reduction between layers. Also, MAF-based architectures are slow to sample.

Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM) [45] are generative models that generate new samples by incrementally reversing a diffusion process. The technique involves gradually adding pure Gaussian noise ϵ_t to the data x_0 at each step t. Throughout the training, the model parameterized by θ learns to predict noise as $\epsilon_\theta(x_t)$. In every iteration, the network computes an approximation of the data $\hat{x}_0$, and then the noise is applied to obtain x_{t-1}:

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t)}{\sqrt{\bar{\alpha}_t}}, \tag{9}$$

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_{t-1}, \tag{10}$$

where $\bar{\alpha}_t = \Pi_{s=1}^{t} \alpha_s$, $\alpha_t = 1 - \beta_t$, and β_t is a variance schedule. The variance schedule controls the amount of noise added at each step, ensuring a gradual and stable diffusion. The loss function is typically:

$$\mathcal{L}(\theta; x_t, \epsilon_t) = \mathbb{E}\left[\| \epsilon_t - \epsilon_\theta(x_t) \|_2^2 \right]. \tag{11}$$

Denoising Diffusion Implicit Models (DDIM) [46] introduces a non-Markovian diffusion process that allows for fewer sampling steps without significant loss of sample quality. The reverse process can be formulated as:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\eta_t^2}\,\epsilon_\theta(x_t) + \eta_t \epsilon_{t-1}, \tag{12}$$

where η_t controls the amount of noise added during the reverse process allowing a balance between the quality and diversity of samples. Setting $\eta_t = 0$ makes the reverse process deterministic, leading to DDIM. If $\eta_t = \sqrt{(1-\bar{\alpha}_{t-1})/(1-\bar{\alpha}_t)}\sqrt{1-\bar{\alpha}_t/\bar{\alpha}_{t-1}}$ the process becomes Markovian, resulting in DDPM.

Diffusion models are new SOTA generative models, producing high-quality, diverse samples [47]. Recent research has concentrated on improving their mathematical

Table 2 Performance of CNN-based autoencoders across various architectures

Architecture	CNN		
Metric	Wasserstein	MAE	RMSE
VAE	**11.52**	17.76	50.38
Supervised AE	23.71	31.90	72.32
AE + Sinkhorn NG	26.53	29.07	66.16
AE + *l*2 NG	37.56	39.32	92.28

Table 3 Performance of ViT-based autoencoders across various architectures

Architecture	ViT		
Metric	Wasserstein	MAE	RMSE
VAE	11.90	18.05	49.48
Supervised AE	20.43	30.60	74.64
AE + Sinkhorn NG	**11.34**	15.88	44.17
AE + *l*2 NG	**11.19**	15.47	43.49

Table 4 Performance of MLP-Mixer-based autoencoders across various architectures.[a]

Architecture	MLP-Mixer		
Metric	Wasserstein	MAE	RMSE
VAE	12.22	18.00	49.51
Supervised AE	17.08	26.90	104.83
AE + Sinkhorn NG	×	×	×
AE + *l*2 NG	×	×	×

[a] The × sign indicates that the model did not converge during the training

Table 5 VQ-VAE reconstruction performance depending on model size

Model size	Wasserstein	MAE	RMSE
0.25M	11.54	12.96	38.46
1 M	**9.86**	**11.84**	**37.22**
4 M	11.73	13.78	43.54
13 M	11.40	12.87	37.90
52 M	12.12	13.73	39.74

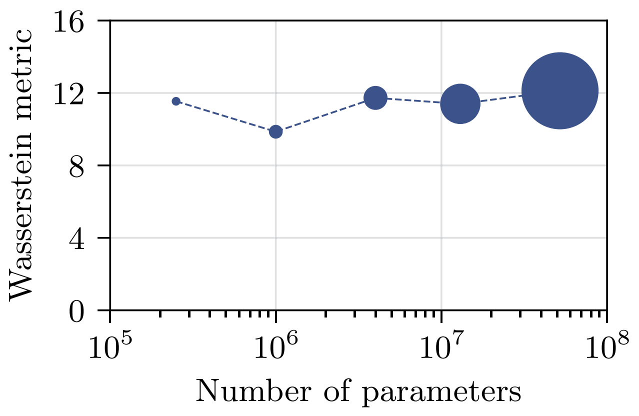


Fig. 6 Reconstruction performance of VQ-VAE as a function of model size. The size of the marker graphically corresponds to the size of the model

foundation and remarkable capabilities [48, 49]. However, they still require significant computational resources.

Training Setup

In this section, we provide an overview of the experimental setup, including dataset preparation, optimizer tuning, and the evaluation metrics used. In addition, we describe the architectures of the models employed in the study.

Dataset Preparation

The dataset was divided into training, validation, and testing sets in proportions of 70%, 10%, and 20%, respectively. The training was carried out on the training set, the parameters were tuned on the validation set, and the final results were calculated on the test set. Conditional variables (i.e., particle parameters) were standardized to a zero mean and a standard deviation of 1, and ZDC responses were logarithmized (except for NFs). Note that the metrics were calculated after inverse-transforming the data to their original scale.

Optimizer Tuning

The AdamW optimizer [50], with hyperparameters fine-tuned through more than 100 trials per model with Optuna software,[2] was used, focusing on the Wasserstein metric. The optimization procedure leveraged the Tree-structured Parzen Estimator (TPE) sampler [51] without pruning and considered the following parameters: learning rate, β_1, β_2, ϵ, use of cosine decay, weight decay, and Nesterov momentum. The repository linked to this paper contains the source code for the models, as well as the hyperparameters for both the models and the optimizers.

Metrics

In the literature on fast ZDC simulations, the Wasserstein distance is the most commonly used metric of model

[2] https://optuna.readthedocs.io/.

performance, defined as the ability of the model to generate samples which closely resemble the input data. This metric is calculated as the average Wasserstein distance of the five detector channels described in Sect. "Dataset analysis". Example histograms for both the original and generated samples are provided in Appendix B. Employing a metric that assesses the global characteristics of the samples is justified as the detector responses often vary between runs, and variations at the individual pixel level should not influence the metric's value. A second metric that also appears in the literature is the mean absolute error (MAE). Unlike the Wasserstein metric, this is a local metric that directly compares the channel values of the original and generated samples. The last metric that we introduce is the pixel-wise root mean squared error (RMSE), which directly compares pixels. This metric indicates the extent to which the model attempts to match the original data. The used metrics can be defined as follows:

$$\text{Wasserstein-1}(w, \hat{w}) = \frac{1}{5}\sum_{i=1}^{5}\int_0^1 \left|F_{w_i}^{-1}(z) - F_{\hat{w}_i}^{-1}(z)\right| dz, \quad (13)$$

$$\text{MAE}(w, \hat{w}) = \frac{1}{n}\sum_{k=0}^{n}\frac{1}{5}\sum_{i=1}^{5}|w_i^k - \hat{w}_i^k|, \quad (14)$$

$$\text{RMSE}(x, \hat{x}) = \sqrt{\frac{1}{n}\sum_{k=0}^{n}\frac{1}{44\cdot 44}\sum_{i=1}^{44}\sum_{j=1}^{44}(x_{ij}^k - \hat{x}_{ij}^k)^2}, \quad (15)$$

where F_q^{-1} is the inverse cumulative distribution function of the distribution q, w_i denotes the distribution of the ith channel, n refers to the number of evaluated examples, w_i^k represents the value of the ith channel of the kth response, x_{ij}^k is the value of the pixel with i and j coordinates of the kth response, and $\hat{w}$ and $\hat{x}$ are the corresponding predicted values.

Note that while lower MAE and RMSE metrics are generally considered better, it does not necessarily imply that the model is superior in the ZDC simulation task. Achieving the lowest possible value is not always desired, as it may suggest that the model frequently produces consistent responses, even when the actual physical process yields diverse results. The optimal values of these metrics should be close to those derived from the original dataset. Due to the high stochasticity of the simulated process, the reported metric values are the average of five runs.

Table 6 Performance comparison of different GAN models

Model	Postprocessing	Wasserstein	MAE	RMSE
GAN		7.09	25.65	104.60
GAN	✓	**5.70**	24.71	100.98
GAN + $l2$ loss		6.44	27.37	109.24
GAN + $l2$ loss	✓	6.07	26.78	107.07
SDI-GAN		6.57	27.01	107.82
SDI-GAN	✓	6.36	26.58	105.94

Table 7 Performance comparison NF depending on the amount of added noise

Noise range: $[0, x)$	Wasserstein
1.0	12.57
0.75	6.67
0.5	**4.58**
0.1	7.10
0.01	8.10

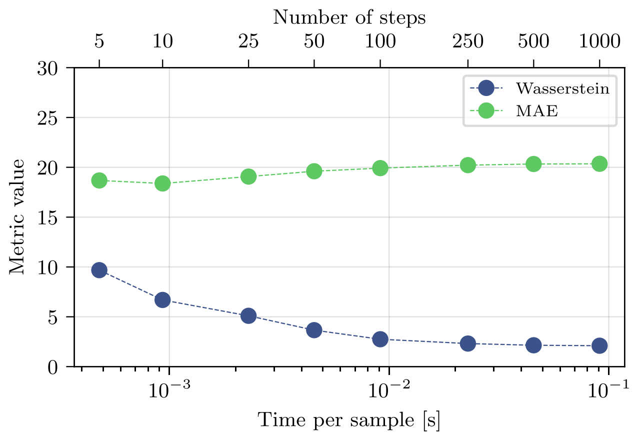


Fig. 7 Performance and time required for a diffusion model to produce samples depending on the number of denoising steps

Models Architecture

Encoder and decoder networks were implemented in the CNN, ViT, and MLP-Mixer architectures. These models have been used as building blocks in autoencoders, GANs, and VQ models, with additional layers specific to each framework. Network designs in various architectures are shown in Appendix C. The diagrams show the scheme of encoder networks, with decoder networks formed by reversing these structures. All models discussed in this paper operate in the raw data space.

Experimental Results

The training was performed on a single NVIDIA A100 GPU with 40 GB of memory on the Athena supercomputer.[3] All models were trained for 100 epochs in batches

[3] https://www.cyfronet.pl/en/19073,artykul,athena.html.

Table 8 Performance comparison of generative frameworks

Model	Wasserstein	MAE	RMSE	Time [ms]
GEANT (original data)	0.53	16.41	59.87	–
Autoencoder	11.19	15.47	43.49	**0.015**
GAN	5.70	24.71	100.98	**0.023**
VQ-VAE	9.61	21.95	65.82	0.091
VQ-GAN	4.58	22.90	85.45	0.091
NF	4.11	19.36	127.22	160.0
Diffusion	**3.15**	20.10	73.58	5.360

of size 256. At the end of each epoch, the weights were stored to allow for reconstruction at any time during the training. The test metrics were computed after the completion of the training. The purpose of the experiments is to compare the performance and speed of different generative models in the task of the ZDC neutron detector simulation. Note, that the most important metric—the Wasserstein distance—for the reference GEANT data equals 0.53 (Table 8), and the goal is to achieve a score as low as possible.

Regarding the autoencoder results, the model with a noise generator in the ViT architecture achieved the highest performance, with a Wasserstein metric score of 11.19, slightly outperforming the VAE based on CNN (Tables 2, 3, and 4). Using Sinkhorn loss improves the results for CNN models but does not provide gains over $l2$ loss in ViT. The VAE consistently produces similar results across various architectures. The results of the experiment indicate that the ViT architecture provides the best and most stable performance and was therefore selected for use in other models in further experiments.

In further experiments, a VQ-VAE model was implemented and the corresponding optimizer was tuned. The encoder and decoder were based on ViT, the codebook size was set to 256, and to ensure good codebook utilization, the gradient update method with $l2$ normalization and projection was employed. To provide more insights about this approach, Table 5 and Fig. 6 show the reconstruction performance of VQ-VAE depending on model size, where the medium-sized model (1 M parameters) achieved the best results with a Wasserstein score of 9.86. Larger models did not show significant improvement, suggesting that 1 M parameters are sufficient for the task. Note that the medium-sized model may have achieved the best results due to the small dataset or codebook size. In this case, because the objective is image reconstruction instead of generation, we aim for the minimum value across all three metrics, which indicates that the reconstructed image is similar to the input. Subsequently, we trained a transformer model acting as a learnable prior with around 4 million parameters. The transformer operated in the next-token prediction regime, which is considered the most efficient approach for small models [52]. Although the size of the model did not significantly affect the results, the selection of suitable sampling parameters ensured optimal generation performance. We experimented with *top-k* and *top-p* sampling methods, but the optimal outcomes were achieved by varying the sampling temperatures, particularly at $\tau = 1.4$. Ultimately, the model achieved a Wasserstein metric value of 9.61, which we report as the final score of VQ-VAE models on the task of generating responses of the ZDC detector.

Following this, VQ-GAN was trained with a combination of $l1$, $l2$, perceptual, and adversarial loss functions. The codebook update method has been changed from gradient update to EMA due to better training stability and codebook utilization. The best results (4.58 in the Wasserstein metric) were achieved by combining $l2$ and adversarial losses with loss weights tuned with Optuna.

Various GANs were implemented with both generator and discriminator optimizers tuned. Additional improvements were made by incorporating the $l2$ loss function, applying the postprocessing step, and training SDI-GAN (where condition c refers to the primary particle 9-D feature vectors). The use of a classical GAN, together with the postprocessing step (scalar multiplication), allowed the best result of 5.70 in the Wasserstein metric (Table 6). Keep in mind that these outcomes might not align with those documented in previous studies due to the utilization of a different network architecture and training methodology.

In addition, we implemented a modified CaloFlow [4] with two separate models: a Bayesian neural network (BNN) for predicting the number of photons based on the other conditional variables, and the MAF for modeling the full ZDC output. Like the authors of [4], we used rational quadratic splines (RQS) as transformations, with their parameters learned by MADE blocks. The flow model was conditioned on the particle properties and the number of photons predicted by the BNN. This way, the second model was only focused on modeling the place and shape of the hit, and not its luminosity. Since the numbers of photons present in the responses are discrete, an additional transformation was needed to help the flow adapt to this distribution. Therefore, a small amount of noise was added to each value, so a particular sum of photons was represented by a range [photon sum; photon sum + max noise) instead of just a photon sum. We investigated how the amount of noise added affects the results. Our experiments revealed that the noise influence is significant; the results for one of the models are shown in Table 7. We treated the noise range as one of the hyperparameters, which allowed for obtaining a final model reaching a Wasserstein score of 4.11.

Finally, a DDPM model with 4 million parameters was trained, utilizing both convolutional and attention layers along with DDIM sampling. The impact of the number of denoising steps on generation time and quality was assessed (Fig. 7), with 50 steps selected as the optimal value due to its balance between generation time and output quality. The time presented in the figure was obtained using one NVIDIA A100 GPU in batches of 2048 images, excluding the compilation time. Furthermore, once the number of steps was determined, the DDIM η parameter was fine-tuned and fixed at 0.7. This adjustment led to an improved model performance, achieving a score of 3.15 in the Wasserstein metric. Note that increasing the number of steps improves the diffusion performance, with 1000 steps resulting in a Wasserstein score of 2.10, although it takes approximately 100 milliseconds to generate a single sample.

Table 8 provides a comprehensive comparison of generative frameworks, presenting performance metrics and generation time. The generation time was measured on the same hardware as before, in batches of 256 images, excluding the compilation time. The diffusion model shows superior performance with the lowest Wasserstein distance of 3.15. The histograms of the values generated by diffusion in the individual channels are very similar to those obtained in Monte Carlo simulations (Appendix B), further confirming the suitability of this model for ZDC simulations. VQ-GAN and NF also perform well with a Wasserstein metric of 4.58 and 4.11, respectively, while GAN shows moderate performance. VQ-VAE and Autoencoder exhibit the poorest performance, indicated by the highest Wasserstein distance. Note that GAN and NF present significantly higher RMSE values compared to the original data, which might imply that while they accurately capture the photon number distribution in the channels, they generate detector responses that are excessively varied from the original ones (Appendix A). Conversely, the autoencoder exhibits a lower RMSE compared to GEANT, probably due to smoothed images and issues with particles that have diverse responses, resulting in the model producing outputs that are close to zero. Example simulation results are shown in Fig. 8. The Wasserstein metric for the original dataset was calculated by randomly dividing the test set in half, with one part acting as detector responses and the other as generated samples. This approach was also applied to the MAE and RMSE metrics, although it required that the compared examples have identical particle features.

Considering the significance of simulation time for surrogate models, we observe that autoencoder and GAN have the shortest generation time, approximately 0.02 milliseconds per sample. VQ-VAE and VQ-GAN have a marginally worse time of 0.1 milliseconds, whereas diffusion and NF, despite their excellent performance, are significantly slower with a generation time of more than 5 and 160 milliseconds, respectively.

Conclusions

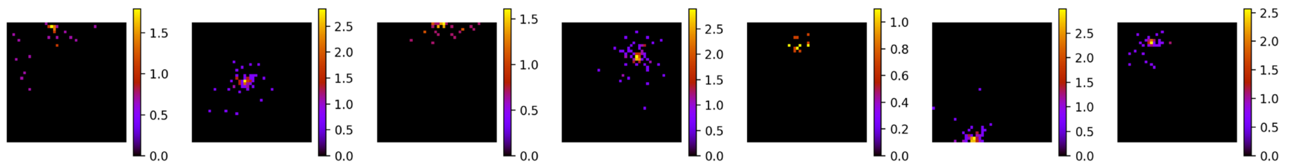
This study presents a comprehensive exploration of SOTA neural network architectures and generative frameworks for fast simulation of the ZDC neutron detector in the ALICE experiment at CERN. Our findings indicate that ViT-based networks are the most effective among the architectures tested. In addition, incorporating VQ methods proves superior to basic VAEs and GANs, offering a good balance between performance (Wasserstein metric of 4.58) and generation speed (0.1 milliseconds per sample). Notably, our implementation of diffusion models and NFs demonstrates excellent performance, with Wasserstein distances of 3.15 and 4.11, respectively. While diffusion models generate samples in a reasonable time frame (5 milliseconds), NFs are not practical due to their longer generation times, despite their strong performance.

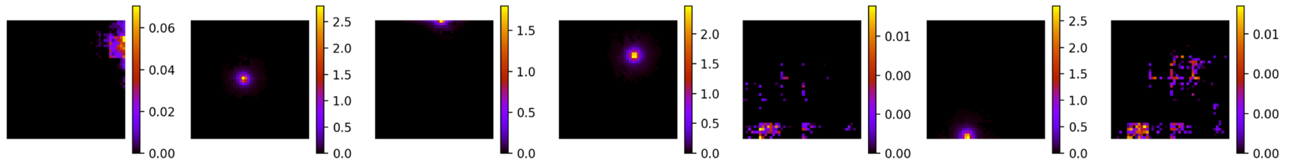
In future work, we primarily want to enhance VQ and diffusion due to their good balance of throughput and performance. We also see potential in the NFs. In specific, we plan to

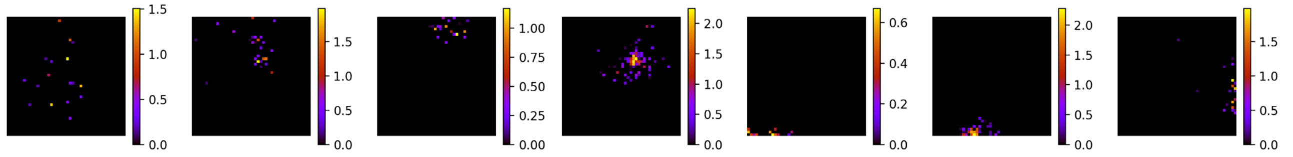
1. Improve the VQ simulation fidelity using the latest ViT advances [53], trying different sampling methods, and testing new codebook update techniques [54].
2. Speed up diffusion the generation time by working in the latent space [55] and reducing the number of denoising steps. To achieve this, exploring knowledge distillation [56] and rectified flows [49] could be beneficial.
3. Further investigate the potential of NFs by testing architectures which allow for a faster generation time at the cost of slower training [43].
4. Create a larger and balanced dataset that accurately covers the space of primary particles. This task is demanding due to the extensive computational resources needed and the complex scientific software involved.

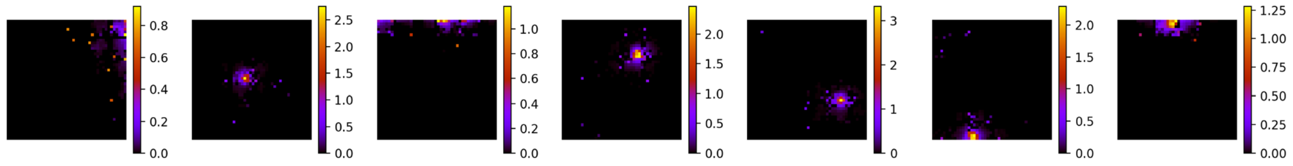
Appendix A: Example simulations

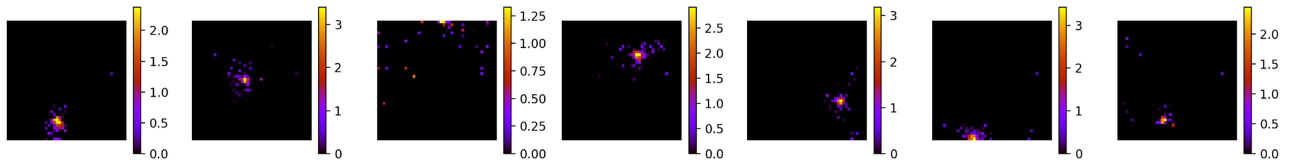
See Fig. 8.

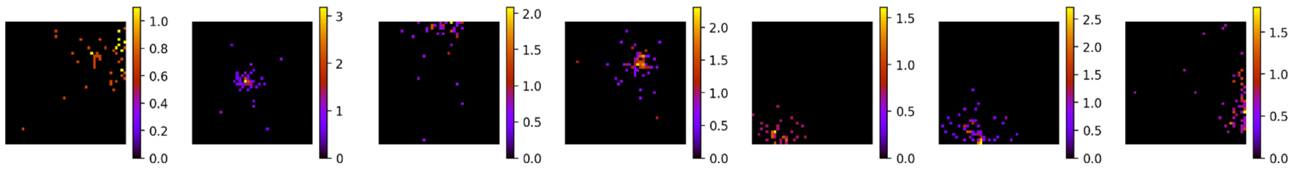

(a) GEANT (original data).


(b) Autoencoder with a noise generator and $l2$ loss.


(c) GAN with a postprocessing step.


(d) VQ-VAE with a transformer as a learnable prior and adjusted sampling temperature.


(e) VQ-GAN with a transformer as a learnable prior.


(f) NF with adjusted training noise.

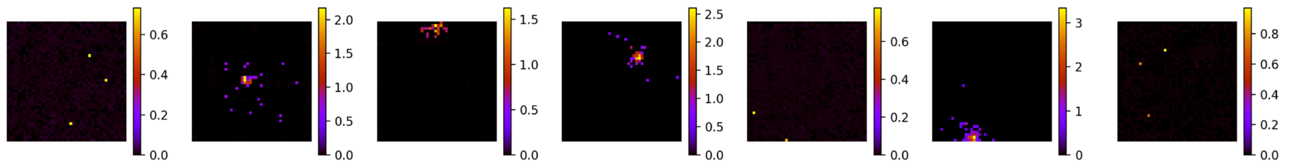

(g) DDIM after 50 denoising steps and adjusted η parameter.

Fig. 8 Example simulations generated by models discussed in this paper. The first column corresponds to the $\pi+$ particle, the middle columns are γ, and the last column is K_S^0

Appendix B: Original and generated histogram

Fig. 9.

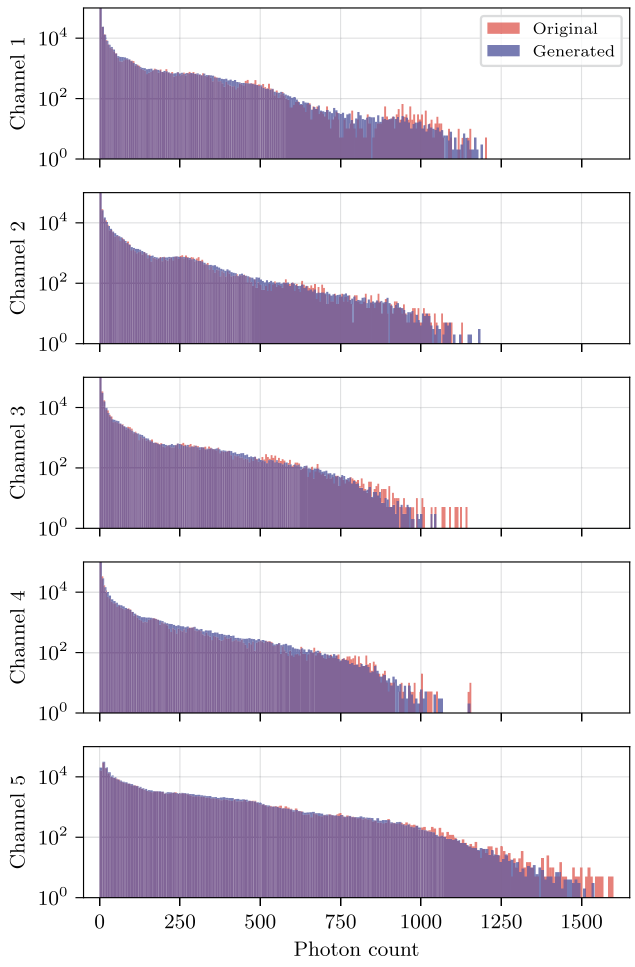


Fig. 9 Histogram of the sum of photons in the ZDC neutron detector channels in responses generated by the GEANT (Original) and the diffusion model (Generated). Note the logarithmic scale on the y-axis

Appendix C: Neural network architectures

Figs. 10, 11 and 12.

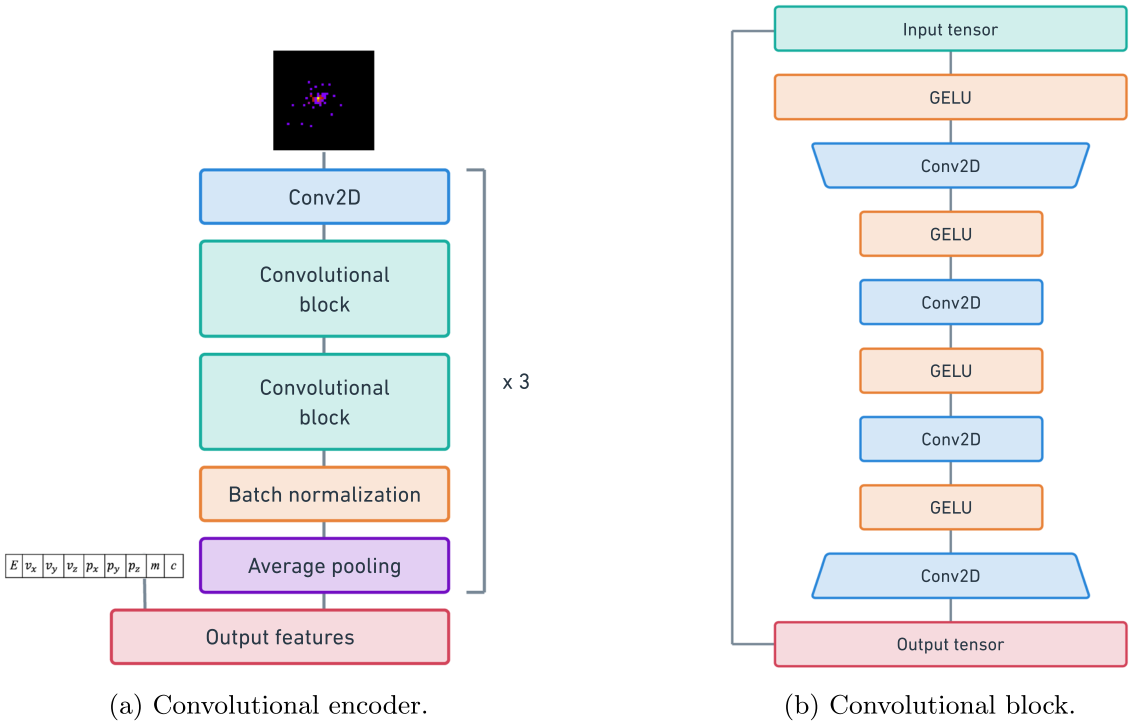
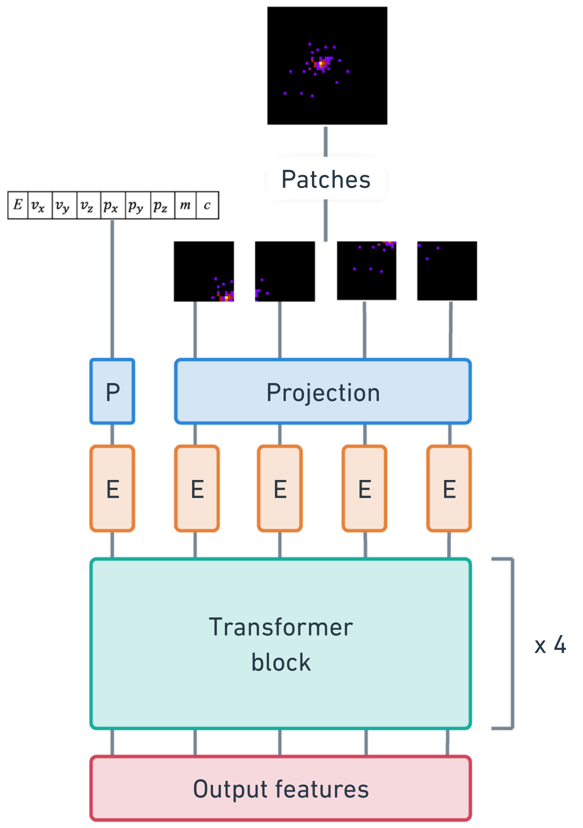
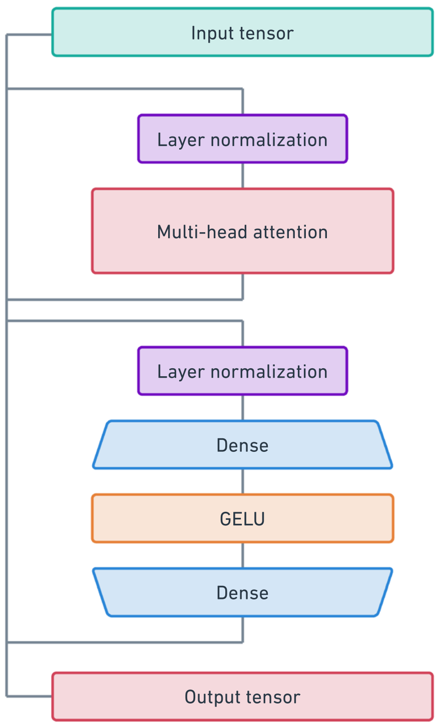


Fig. 10 Convolutional encoder architecture

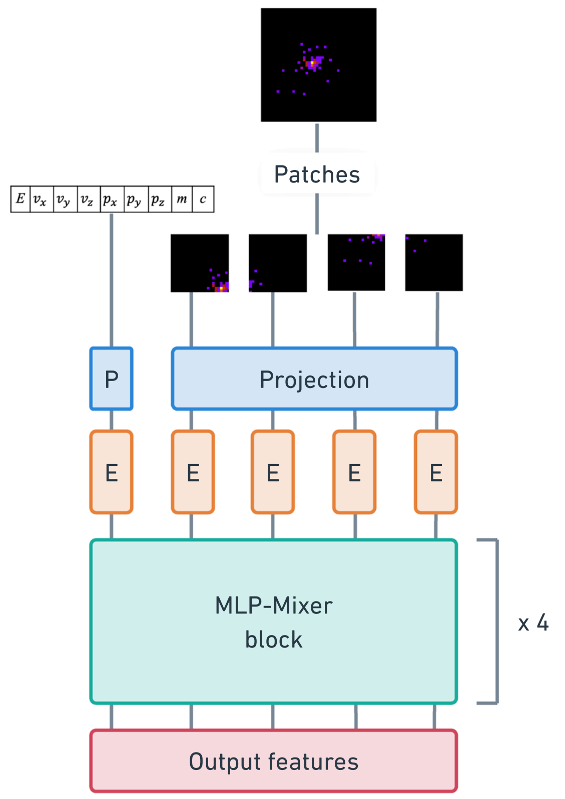


(a) ViT encoder. P is an abbreviation for projection, while E represents embedded vectors.

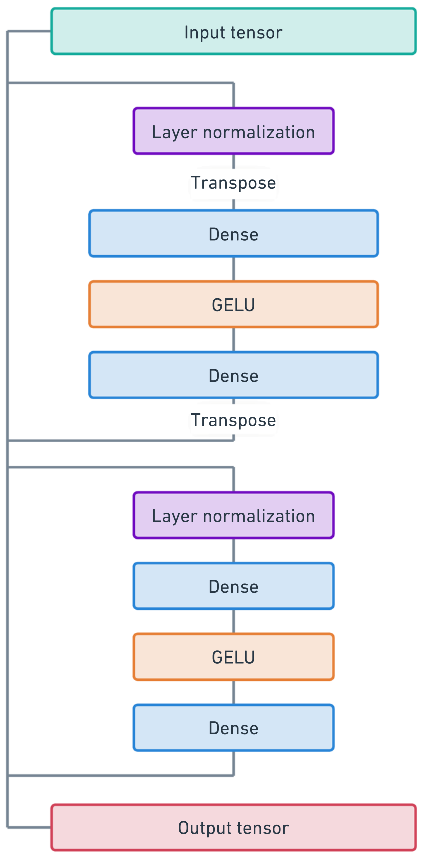


(b) Transformer block.

Fig. 11 ViT encoder architecture



(a) MLP-Mixer encoder. P is an abbreviation for projection, while E represents embedded vectors.



(b) MLP-Mixer block.

Fig. 12 MLP-Mixer encoder architecture

Acknowledgements We would like to thank Professors Jacek Kitowski and Marcin Kurdziel from AGH University of Krakow, Professor Jacek Otwinowski from the Institute of Nuclear Physics PAN in Krakow, and Sandro Wenzel, PhD from CERN for their support in this work. We would also like to thank the Warsaw University of Technology's group: Jan Dubiński, Kamil Deja, PhD, and Professor Tomasz Trzciński for the discussions during our meetings. This work was partly supported by the "Excellence Initiative—Research University" (IDUB) program for the AGH University of Krakow. This work is co-financed and in part supported by the Ministry of Science and Higher Education (Agreement Nr 2023/WK/07) by the program entitled "PMW" and by the Ministry funds assigned to AGH University in Krakow. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH) for providing computer facilities and support within computational grants no. PLG/2023/016410 and PLG/2024/017264.

Author contributions All authors contributed to the research conception and methodology, discussed the results, reviewed and edited the manuscript. M.W. analysed the dataset, prepared and tested models (VAE, GAN, VQ, Diffusion), prepared the original draft. E.M. analysed the dataset, prepared and tested models (NF), prepared the original draft. W.D. supervised the work.

Data availability The data used in this work is owned by the ALICE, CERN.

Declarations

Competing interests The authors declare no competing interests.

References

1. Alizadeh R, Allen JK, Mistree F (2020) Managing computational complexity using surrogate models: a critical review. Res Eng Design 31(3):275–298. https://doi.org/10.1007/s00163-020-00336-7
2. Thuerey N, Holl P, Mueller M, et al (2022) Physics-based deep learning. arxiv:2109.05237
3. Dohi K (2020) Variational autoencoders for jet simulation. arxiv:2009.04842
4. Krause C, Shih D (2021) CaloFlow: Fast and accurate simulations of calorimeter showers with normalizing flows. Phys Rev D 107(11).
5. de Oliveira L, Paganini M, Nachman B (2017) Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. Comput Softw Big Sci 1(1). 10.1007/s41781-017-0004-6
6. Gallio M, et al (1999) ALICE zero-degree calorimeter (ZDC): technical design report. Technical design report. ALICE, CERN, Geneva, https://cds.cern.ch/record/381433
7. Oppedisan C (2001) Centrality measurement in the ALICE experiment with the zero degree calorimeters. https://cds.cern.ch/record/1067488
8. Agostinelli S et al (2003) Geant4–a simulation toolkit. Nucl Instrum Methods Phys Res Sect A 506(3):250–303. https://doi.org/10.1016/S0168-9002(03)01368-8
9. (2022) Fast calorimeter simulation challenge. https://indico.cern.ch/event/1140563/. Accessed 06 Nov 2024
10. Dubiński J, Deja K, Wenzel S et al (2024) Machine learning methods for simulating particle response in the zero degree calorimeter at the ALICE experiment. CERN 3061(1):040001. https://doi.org/10.1063/5.0203567
11. Dubiński J, Deja K, Wenzel S, et al (2023) Selectively increasing the diversity of gan-generated samples. In: Neural Information Processing: 29th International Conference, ICONIP 2022, Virtual Event, November 22-26, 2022, Proceedings, Part I. Springer-Verlag, Berlin, Heidelberg, p 260-270.
12. Kita M, Dubiński J, Rokita P, et al (2024) Generative diffusion models for fast simulations of particle collisions at cern. arxiv:2406.03233
13. Butter A, Plehn T, Winterhalder R (2019) How to GAN LHC events. SciPost Phys 7:075
14. Deja K, Trzciński T, Graczykowski Ł (2019) Generative models for fast cluster simulations in the TPC for the ALICE experiment. EPJ Web Conf 214:06003. https://doi.org/10.1051/epjconf/201921406003
15. Khattak Gr, Vallecorsa S, Carminati F (2018) Three dimensional energy parametrized generative adversarial networks for electromagnetic shower simulation. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp 3913–3917.
16. Musella P, Pandolfi F (2018) Fast and accurate simulation of particle detectors using generative adversarial networks. Comput Softw Big Sci 2(1):8. https://doi.org/10.1007/s41781-018-0015-y
17. Paganini M, de Oliveira L, Nachman B (2018) CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. Phys Rev D 97:014021. https://doi.org/10.1103/PhysRevD.97.014021
18. Buhmann E et al (2021) Getting high: high fidelity simulation of high granularity calorimeters with high speed. Comput Softw Big Sci 5(1):13. https://doi.org/10.1007/s41781-021-00056-0
19. Cresswell JC, Ross BL, Loaiza-Ganem G, et al (2022) Caloman: fast generation of calorimeter showers with density estimation on learned manifolds. arxiv:2211.15380
20. Orzari B et al (2023) LHC hadronic jet generation using convolutional variational autoencoders with normalizing flows. Mach Learn Sci Technol 4(4):045023. https://doi.org/10.1088/2632-2153/ad04ea
21. Krause C, Shih D (2023) Caloflow ii: Even faster and still accurate generation of calorimeter showers with normalizing flows. arxiv:2110.11377
22. Deja K et al (2021) End-to-end sinkhorn autoencoder with noise generator. IEEE Access 9:7211–7219. https://doi.org/10.1109/ACCESS.2020.3048622
23. Rogoziński K, Dubiński J, Rokita P, et al (2024) Particle physics dl-simulation with control over generated data properties. arxiv:2405.14049
24. Wang S, et al (2022) Multi-objective deep data generation with correlated property control. In: Advances in Neural Information Processing Systems, vol 35. Curran Associates, Inc., pp 28889–28901, https://proceedings.neurips.cc/paper_files/paper/2022/file/b9c2e8a0bbed5fcfaf62856a3a719ada-Paper-Conference.pdf
25. Będkowski P, Dubiński J, Deja K, et al (2024) Deep generative models for proton zero degree calorimeter simulations in alice, cern. arxiv:2406.03263
26. Dosovitskiy A, et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. CoRR abs/2010.11929. arxiv:2010.11929
27. Tolstikhin I, Houlsby N, Kolesnikov A, et al (2021) Mlp-mixer: an all-mlp architecture for vision. arxiv:2105.01601
28. Schicker R (2017) Overview of ALICE results in pp, pA and AA collisions. EPJ Web Conf 138:01021. https://doi.org/10.1051/epjconf/201713801021
29. (2003) ALICE Zero Degree Calorimeter (ZDC), General Pictures. https://cds.cern.ch/record/630193. ALICE Collection. Accessed 06 Nov 2024
30. Kingma DP, Welling M (2022) Auto-encoding variational bayes. arxiv:1312.6114
31. Le L, Patterson A, White M (2018) Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In: Bengio S, Wallach H, Larochelle H, et al (eds) Advances in Neural Information Processing Systems, vol 31. Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2018/file/2a38a4a9316c49e5a833517c45d31070-Paper.pdf
32. Hu X, Hu H, Verma S et al (2021) Physics-guided deep neural networks for power flow analysis. IEEE Trans Power Syst 36(3):2082–2092. https://doi.org/10.1109/TPWRS.2020.3029557
33. Tonolini F, Jensen BS, Murray-Smith R (2020) Variational sparse coding. In: Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, Proceedings of Machine Learning Research, vol 115. PMLR, pp 690–700, https://proceedings.mlr.press/v115/tonolini20a.html
34. Goodfellow I, Pouget-Abadie J, Mirza M et al (2020) Generative adversarial networks. Commun ACM 63(11):139–144. https://doi.org/10.1145/3422622
35. Isola P, Zhu JY, Zhou T, et al (2017) Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 5967–5976.
36. Brock A, Donahue J, Simonyan K (2018) Large Scale GAN Training for High Fidelity Natural Image Synthesis. CoRR abs/1809.11096. arxiv:1809.11096
37. van den Oord A, Vinyals O, Kavukcuoglu K (2018) Neural discrete representation learning. arxiv:1711.00937
38. van den Oord A, Kalchbrenner N, Kavukcuoglu K (2016) Pixel recurrent neural networks. arxiv:1601.06759
39. Esser P, Rombach R, Ommer B (2021) Taming Transformers for High-Resolution Image Synthesis. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, pp 12868–12878
40. Zhang R, Isola P, Efros AA, et al (2018) The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE/

CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, pp 586–595.

41. Yu J, Li X, Koh JY, et al (2021) Vector-quantized Image Modeling with Improved VQGAN. CoRR abs/2110.04627. arxiv:2110.04627
42. Papamakarios G, Pavlakou T, Murray I (2017) Masked Autoregressive Flow for Density Estimation. In: Advances in Neural Information Processing Systems, vol 30. Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2017/file/6c1da886822c67822bcf3679d04369fa-Paper.pdf
43. Kingma DP, et al (2016) Improved variational inference with inverse autoregressive flow. In: Advances in Neural Information Processing Systems, vol 29. Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf
44. Germain M, Gregor K, Murray I, et al (2015) Made: Masked autoencoder for distribution estimation. arxiv:1502.03509
45. Ho J, Jain A, Abbeel P (2020) Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol 33. Curran Associates, Inc., pp 6840–6851, https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf
46. Song J, Meng C, Ermon S (2022) Denoising diffusion implicit models. arxiv:2010.02502
47. Dhariwal P, Nichol A (2021) Diffusion models beat GANs on image synthesis. CoRR abs/2105.05233. arxiv:2105.05233
48. Esser P, Kulal S, Blattmann A, et al (2024) Scaling rectified flow transformers for high-resolution image synthesis. arxiv:2403.03206
49. Lipman Y, Chen RTQ, Ben-Hamu H, et al (2023) Flow matching for generative modeling. arxiv:2210.02747
50. Loshchilov I, Hutter F (2019) Decoupled Weight Decay Regularization. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, https://openreview.net/forum?id=Bkg6RiCqY7
51. Bergstra J, Bardenet R, Bengio Y, et al (2011) Algorithms for Hyper-Parameter Optimization. In: Advances in Neural Information Processing Systems, vol 24. Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
52. Kilian M, Jampani V, Zettlemoyer L (2024) Computational tradeoffs in image synthesis: Diffusion, masked-token, and next-token prediction. arxiv:2405.13218
53. Alabdulmohsin I, Zhai X, Kolesnikov A, et al (2023) Getting vit in shape: Scaling laws for compute-optimal model design. In: Thirty-seventh Conference on Neural Information Processing Systems, https://openreview.net/forum?id=en4LGxpd9E
54. Yu L, Lezama J, Gundavarapu NB, et al (2024) Language model beats diffusion – tokenizer is key to visual generation. arxiv:2310.05737
55. Rombach R, Blattmann A, Lorenz D, et al (2022) High-resolution image synthesis with latent diffusion models. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, pp 10674–10685, 10.1109/CVPR52688.2022.01042
56. Sauer A, Lorenz D, Blattmann A, et al (2023) Adversarial diffusion distillation. ArXiv abs/2311.17042. https://api.semanticscholar.org/CorpusID:265466173