Generating blood cells with ML

José Filipe

Universidade de Coimbra josepedrofilipe1@gmail.com

Abstract. This report explores the use of deep generative models to synthesize realistic medical images using the BloodMNIST dataset. Three model families were implemented and evaluated: a Variational Autoencoder (VAE), a Deep Convolutional GAN (DCGAN) and a Diffusion Model based on a simplified U-Net architecture. The main evaluation metric was the Fréchet Inception Distance (FID), computed over 10,000 generated and real samples across multiple runs. Contrary to expectations, the VAE achieved the best FID score, followed by the DCGAN, while the diffusion model underperformed significantly. These results may be attributed to the simplicity of the architectures and severe computational constraints, particularly in the diffusion model's generation phase, which required extreme memory management strategies. The project highlights the trade-offs between model complexity, training cost and generative quality in a medical imaging context.

Keywords: Generative Models · BloodMNIST · VAE · GAN · Diffusion Model · FID · Medical Imaging

Introduction

The field of generative modeling has seen significant advancements in recent years, becoming a cornerstone of modern machine learning research. These models aim to learn complex data distributions and generate new samples that are similar to the original data, with applications ranging from image and audio synthesis to data augmentation and anomaly detection.

Medical imaging, in particular, stands to benefit greatly from generative approaches. Synthetic data can be used to enrich datasets, improve the robustness of diagnostic models and support scenarios where data collection is limited due to privacy, cost or rarity. In this context, the ability to generate high-quality, realistic medical images becomes both a technical challenge and a valuable tool.

This report focuses on exploring and evaluating different deep generative models applied to medical image data, specifically using the BloodMNIST dataset. The aim is to understand the capabilities and limitations of various generative architectures when applied to real-world biomedical imagery.

Description of the problem

The problem addressed in this project is the generation of synthetic medical images that closely resemble real data in both structure and variability. The

dataset used is BloodMNIST, which contains 17,092 labeled RGB images of microscopic blood cells, spanning 8 distinct classes. These images exhibit subtle yet important differences in shape, color and texture, making the generative task both non-trivial and clinically relevant.

The goal is to train models capable of learning the underlying distribution of the dataset and using it to generate new, plausible samples. This involves capturing complex visual features and patterns in an unsupervised manner, without access to class labels or explicit targets during generation.

This problem poses several challenges:

1. Preserving fine-grained visual features, which are essential for realism in medical imagery;
2. Avoiding mode collapse or oversimplified generations, which often occur when the generative model fails to represent the full diversity of the dataset;
3. Ensuring diversity and consistency across generated samples while maintaining fidelity to the original data distribution.

By solving this problem, generative models can support a range of applications, from dataset augmentation and model pretraining to educational tools and synthetic data generation for privacy-sensitive scenarios.

Approach

To tackle this problem, we implemented and evaluated three distinct families of deep generative models: a Variational Autoencoder (VAE), a Deep Convolutional Generative Adversarial Network (DCGAN), and a Diffusion Model based on a simplified U-Net architecture. These models represent different paradigms in generative modeling, each with their own strengths and challenges. While these three models form the foundation of our approach, the goal is to obtain the best possible results. Ideally, additional strategies and improvements will be adopted throughout the project to enhance generation quality, training stability, and overall performance.

The evaluation part is based on the Fréchet Inception Distance (FID), which quantitatively compares the distributions of real and generated images in a high-level feature space. For each model, 10,000 synthetic samples are generated and compared to 10,000 real samples. This metric captures both visual fidelity and diversity, with lower values indicating better generative performance.

Experimental Setup

All models were implemented in Python using the PyTorch framework and trained in a CUDA-enabled environment with GPU acceleration. The dataset was preprocessed by normalizing all images to the range $[-1, 1]$, matching the output activation functions used in the generative architectures. A uniform batch size of 128 was adopted across all experiments to ensure consistency.

To quantitatively evaluate the generative quality, the *Fréchet Inception Distance* (FID) was used. An initial pool of 10,000 real images was randomly sampled from the dataset and fixed throughout the evaluation. For each model, 10,000 synthetic samples were generated and compared to this real set. To account for variability due to model stochasticity, the process was repeated across five independent runs using different random seeds and the average FID score was reported. Metric computation was performed using the `pytorch-fid` implementation.

Each model was trained independently using the Adam optimizer, with learning rates and training durations adapted to their respective architectures, as detailed below.

Variational Autoencoder (VAE)

The implemented VAE follows the standard encoder–decoder architecture with a reparameterization trick. The encoder consists of two convolutional layers with ReLU activations, which progressively reduce the input resolution from $28\times28\times3$ to $7 \times 7 \times 64$. The resulting feature map is flattened and projected into a latent space of dimension 64 using two fully connected layers, one for the mean and one for the log-variance. Sampling is done using the reparameterization trick to ensure backpropagation through stochastic nodes.

The decoder mirrors the encoder, beginning with a fully connected layer that reshapes the latent vector into a $7 \times 7 \times 64$ tensor. This is followed by two transposed convolutional layers that upsample the data back to $28 \times 28 \times 3$. A `Tanh` activation is applied at the output to normalize pixel values to the range $[-1, 1]$.

The loss function combines mean squared error (MSE) for image reconstruction and the Kullback–Leibler (KL) divergence to regularize the latent distribution toward a standard normal prior. The model was trained using the Adam optimizer for 20 epochs with a learning rate of 1×10^{-3}.

Deep Convolutional GAN (DCGAN)

The DCGAN implementation uses a standard generator–discriminator adversarial setup. The generator takes a 100-dimensional latent vector sampled from a standard Gaussian distribution and first projects it into a dense $7\times7\times512$ feature map using a linear layer, followed by batch normalization and ReLU activation. The image is then progressively upsampled to the original resolution through two transposed convolutional layers with intermediate batch normalization and ReLU activations. A `Tanh` activation is applied at the output.

The discriminator uses a mirrored architecture of the generator with convolutional layers that downsample the input image. LeakyReLU activations and batch normalization are applied between layers. The final output is a single sigmoid neuron, indicating the probability that the input image is real.

Both models were trained with the Adam optimizer using a learning rate of 2×10^{-4} and β parameters $(0.5,\ 0.999)$. The discriminator and generator are

updated alternately during training. Binary cross-entropy loss is used for both networks. The model was trained for 50 epochs.

Diffusion Model

The diffusion model was implemented using a simplified denoising diffusion probabilistic model (DDPM) framework, which progressively adds Gaussian noise to training images over 1000 steps and trains a neural network to reverse this process. The forward diffusion process is controlled by a linear noise schedule where the noise variance β increases from 1×10^{-4} to 0.02 over $T = 1000$ steps.

The denoising model is a lightweight U-Net composed of four convolutional layers and incorporates temporal information through sinusoidal time embeddings. A positional encoding block maps each timestep to a feature vector, which is injected into intermediate layers of the network. The model predicts the added noise at each timestep using an MSE loss.

Training was done using the Adam optimizer with a learning rate of 1×10^{-4} for 10 epochs. To improve stability and allow for recovery in case of interruptions, a checkpointing mechanism was implemented.

Analysis of the results

Before presenting the results, it is useful to consider what performance can be expected from each model in terms of generative quality. In general, diffusion models are known for producing high-fidelity and diverse images, often outperforming other approaches in tasks involving complex data distributions. Therefore, we expect the diffusion model to achieve the lowest FID score.

DCGANs, which rely on adversarial training, are capable of generating sharp and visually convincing images. However, they are also susceptible to training instability and mode collapse, which may affect diversity. As such, we anticipate intermediate FID scores from the DCGAN.

VAEs, while more stable and efficient to train, tend to generate blurrier images due to their reliance on reconstruction loss functions like MSE. Consequently, we expect VAEs to achieve the highest FID scores among the three models, reflecting lower visual fidelity.

The actual results are presented in the table below:

Model	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Average FID
VAE	225.50	224.35	-	-	-	224.92
DCGAN	275.95	277.08	276.49	277.61	276.46	276.7243
Diffusion Model	416.49	416.75	416.50	415.53	415.60	416.17

Table 1: FID scores for each generative model across 5 different seeds.

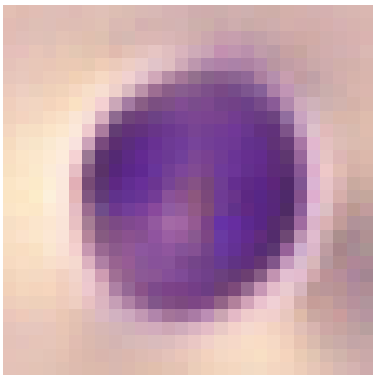
Ironically, the results obtained were the exact opposite of what was initially expected. As we can see in Table 1, the VAE achieved the lowest FID score

(225.51), followed by the DCGAN (276.72), while the diffusion model—typically regarded as the most powerful in terms of image quality—performed the worst with a score of 416.18. This inversion of expectations suggests that the models may not have reached their full potential under the current training conditions.

It is also important to note that, due to computational limitations, only two independent FID evaluations were completed for the VAE. The reported score corresponds to the average of these two runs and should be interpreted as an approximate estimate of the model's true performance. These constraints likely affected the overall results, particularly in the case of the diffusion model, which is especially demanding in terms of memory and training time.

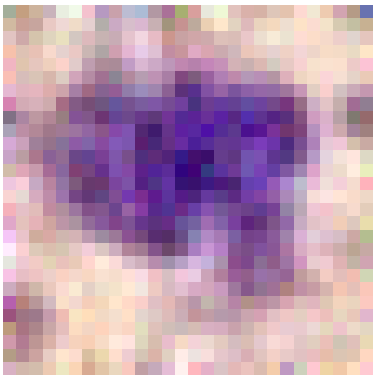
Qualitative Observations

(a) Sample from the real dataset



(b) Image generated by the VAE

(c) Image generated by the DCGAN



(d) Image generated by the Diffusion Model

Fig. 1: Example of one real and three generated images, one per model.

About the generated images, it is important to note that they serve only as isolated examples and should not be used to draw definitive conclusions about model performance. Since only one image per model is shown, they do not reflect the full range of variability, consistency, or typical failure cases observed during training. Still, it is worth noting that the image produced by the DCGAN appears particularly convincing—ironically, perhaps even more realistic than the actual real image displayed. This highlights the limitations of qualitative, single-sample comparisons and reinforces the need for quantitative evaluation on a larger scale.

Conclusion

Despite being initiated with reasonable anticipation, it was not possible to implement additional strategies to improve results due to the computational cost associated with training generative models. In particular, the diffusion model plus the DCGAN posed significant memory challenges, including frequent out-of-memory errors during execution. Although measures such as checkpointing and explicit memory management after each epoch were introduced to mitigate these issues, they only partially alleviated the problem and did not fully meet expectations.

The results obtained in this project were, overall, far from satisfactory. Contrary to expectations, simpler models such as the VAE outperformed the diffusion model in terms of FID score, which calls into question the effectiveness of the current implementations. This outcome may be partially explained by the simplicity of the architectures used. The VAE and DCGAN implementations were relatively minimal, lacking features like residual connections, attention mechanisms, or deeper convolutional hierarchies, which are known to enhance expressiveness and stability. The diffusion model, although conceptually more powerful, was also implemented in a lightweight form, with a shallow U-Net and basic time embedding, limiting its potential to fully capture the complexity of the data.

The initial plan was to start with baseline versions of each model so that the effect of progressively introduced strategies could be more clearly observed and evaluated. However, due to time and resource constraints, it was not possible to proceed with the intended enhancements.

Several improvements could have contributed to better results, especially for the diffusion model. These include using a more expressive U-Net architecture, applying classifier-free guidance, fine-tuning the noise schedule, or extending the number of training epochs. Some of these strategies were partially explored but ultimately discarded, as the image generation phase with diffusion proved to be prohibitively time-consuming and memory-intensive.

In fact, generating 10,000 images per seed using the implemented sampling loop required iterating over 1,000 reverse diffusion steps for each batch, leading to long runtimes even with moderate batch sizes. Memory management became a critical bottleneck, requiring extreme measures such as freeing tensors and manually clearing GPU cache after every batch. While this allowed the genera-

tion process to complete, it significantly slowed down the entire experimentation pipeline and made iterative development impractical.

Overall, the project highlights both the promise and the challenges of using generative models in medical imaging. Future work should focus on more robust and scalable architectures, as well as on optimizing generation pipelines to better balance quality and computational efficiency.

References

1. OpenAI ChatGPT, https://chat.openai.com. Last accessed 1 Jun 2025