# Synthetic Vision: DCGAN vs. VAE in Image Generation on CIFAR-10

**Manikantan Srinivasan and Poornima Jaykumar Dharamdasani**
MS in Artificial Intelligence
Northeastern University
Boston, MA 02115

## Abstract

This project explores Deep Convolutional Generative Adversarial Networks (DCGANs) and Variational Autoencoders (VAEs) by developing and comparing them with the CIFAR-10 dataset. Utilizing the PyTorch library, these models are built from scratch to generate and reconstruct images. The study evaluates their performance using the Inception score (IS) and the Fréchet Inception Distance (FID).

## 1 Introduction

This investigation explores synthetic image creation, significantly enhanced by deep learning advancements, focusing on DCGAN [1] and VAE[2]. DCGANs leverage convolutional layers to handle complex image features effectively, while VAEs employ a probabilistic approach to encode and decode inputs through latent space, facilitating new image generation and smooth data representation. Using the CIFAR-10 dataset and PyTorch, we developed and analyzed DCGAN and VAE from scratch for image generation and reconstruction. The models were assessed using the FID and IS. DCGANs are noted for producing high-quality images, and VAEs for their ease of sampling and interpolating between classes, allowing us to assess the capabilities and constraints of these generative models in producing diverse and realistic images.

## 2 Dataset

The CIFAR-10 dataset, known for its moderate size and diverse image content, is utilized for both the development and evaluation of models. It comprises 60,000 32x32 color images across 10 classes. Of these, 50,000 images form the training set. For testing purposes, 10,000 generated images were utilized for DCGANs, while 10,000 images from the CIFAR-10 test set were used to evaluate VAEs.

## 3 Implementation details

### 3.1 DCGAN architecture and training

A GAN consists of a generator model for generating new data and a discriminator model for classifying whether the generated data is real or fake. A DCGAN is a direct extension of the GAN, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.

Architecture- As shown in Fig. 1, the Discriminator starts with a convolution layer and LeakyReLU activation, followed by layers that increase channel depth through 128, 256, and 512 filters, each also using LeakyReLU and batch normalization for stability. The final linear layer consolidates features into a single output. As shown in Fig. 2, the Generator begins with a transposed convolutional layer, increasing image size while decreasing channel depth from 1024 to 128, incorporating batch

Discriminator Model Summary:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 64, 16, 16] | 4,800 |
| LeakyReLU-2 | [-1, 64, 16, 16] | 0 |
| Conv2d-3 | [-1, 128, 8, 8] | 204,800 |
| LeakyReLU-4 | [-1, 128, 8, 8] | 0 |
| BatchNorm2d-5 | [-1, 128, 8, 8] | 256 |
| Conv2d-6 | [-1, 256, 4, 4] | 819,200 |
| LeakyReLU-7 | [-1, 256, 4, 4] | 0 |
| BatchNorm2d-8 | [-1, 256, 4, 4] | 512 |
| Conv2d-9 | [-1, 512, 2, 2] | 3,276,800 |
| LeakyReLU-10 | [-1, 512, 2, 2] | 0 |
| BatchNorm2d-11 | [-1, 512, 2, 2] | 1,024 |
| Linear-12 | [-1, 1] | 2,049 |

Total params: 4,309,441

Figure 1: Discriminator model architecture

Generator Model Summary:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| ConvTranspose2d-1 | [-1, 1024, 2, 2] | 409,600 |
| BatchNorm2d-2 | [-1, 1024, 2, 2] | 2,048 |
| ReLU-3 | [-1, 1024, 2, 2] | 0 |
| ConvTranspose2d-4 | [-1, 512, 4, 4] | 8,388,608 |
| BatchNorm2d-5 | [-1, 512, 4, 4] | 1,024 |
| ReLU-6 | [-1, 512, 4, 4] | 0 |
| ConvTranspose2d-7 | [-1, 256, 8, 8] | 2,097,152 |
| BatchNorm2d-8 | [-1, 256, 8, 8] | 512 |
| ReLU-9 | [-1, 256, 8, 8] | 0 |
| ConvTranspose2d-10 | [-1, 128, 16, 16] | 524,288 |
| BatchNorm2d-11 | [-1, 128, 16, 16] | 256 |
| ReLU-12 | [-1, 128, 16, 16] | 0 |
| ConvTranspose2d-13 | [-1, 3, 32, 32] | 6,144 |
| Tanh-14 | [-1, 3, 32, 32] | 0 |

Total params: 11,429,632

Figure 2: Generator model architecture

normalization and ReLU activation at each step. Its final layer reduces to 3 channels and applies Tanh activation, producing a 32x32 image. The Binary Cross-Entropy With Logits (BCEWithLogits) loss function, an extension of the standard PyTorch's nn.BCEWithLogitsLoss is used to compute the binary cross-entropy loss between predictions and true values.

Training details- The dataset preprocessing includes random horizontal flipping and normalization of images to a mean and standard deviation of 0.5, enhancing training speed and reducing convergence issues. The Adam optimizer is used for both the discriminator and generator, with a learning rate scheduler that linearly decreases the rate during training. Training begins by generating noise vectors to create fake images. The discriminator evaluates these alongside real images, calculating its loss based on its ability to distinguish between them. The generator's loss is determined by its success in deceiving the discriminator. The entire model is trained for 128 epochs.

### 3.2 VAE architecture and training

A VAE differs from traditional autoencoders by encoding inputs as probability distributions in the latent space, rather than discrete points. Each input is described by distribution parameters like mean and standard deviation, enhancing the VAE's ability to manage data uncertainties and variabilities.

Encoder Summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 128, 16, 16] | 3,584 |
| Conv2d-2 | [-1, 256, 8, 8] | 295,168 |
| Conv2d-3 | [-1, 512, 4, 4] | 1,180,160 |
| Conv2d-4 | [-1, 1024, 2, 2] | 4,719,616 |
| Flatten-5 | [-1, 4096] | 0 |
| Linear-6 | [-1, 3072] | 12,585,984 |
| Linear-7 | [-1, 512] | 1,573,376 |
| Linear-8 | [-1, 512] | 1,573,376 |

Total params: 21,931,264

Figure 3: Encoder model architecture

Decoder Summary

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Linear-1 | [-1, 4096] | 2,101,248 |
| ConvTranspose2d-2 | [-1, 256, 4, 4] | 2,359,552 |
| ConvTranspose2d-3 | [-1, 128, 8, 8] | 295,040 |
| ConvTranspose2d-4 | [-1, 64, 16, 16] | 73,792 |
| ConvTranspose2d-5 | [-1, 3, 32, 32] | 1,731 |
| ConvTranspose2d-6 | [-1, 3, 32, 32] | 84 |

Total params: 4,831,447

Figure 4: Decoder model architecture

Table 1: Evaluation results

| Model | Inception Score (IS) | Fréchet Inception Distance (FID) |
|---|---|---|
| DCGAN | 5.48 | 58.4 |
| VAE | 2.9 | 163.3 |

Architecture- The encoder in the VAE transforms high-dimensional input data into a probabilistic latent space, outputting parameters such as mean and variance for Gaussian distributions, as depicted in Fig. 3. This latent space, characterized by its n-dimensional Gaussian nature, compresses the input data. The decoder then reconstructs the data from this space using sampled parameters to approximate the original inputs, with its architecture shown in Fig. 4. This mechanism allows the VAE to generate new, varied data points, enhancing its generative capabilities. The autoencoder's loss function combines Binary Cross-Entropy Loss (BCE) and KL Divergence (KLD). BCE penalizes discrepancies between original and reconstructed images, ensuring accuracy, while KL Divergence measures deviation from a standard Gaussian distribution, optimizing for generative performance. The total loss is a weighted sum of these components and is given by Eq. 1.

$$L = \frac{1}{N} \sum_{n=1}^{N} \left( L_{\text{recon}}^{(n)} + L_{\text{KL}}^{(n)} \right) \tag{1}$$

where $L_{recon}$ and $L_{KL}$ is the reconstruction loss and KL-divergence respectively.

Training details- The data is first normalized to the range of [0,1]. The training process of a VAE involves optimizing the encoder and decoder through backpropagation and stochastic gradient descent. A critical aspect of training VAEs is the reparameterization trick, which enables gradient-based optimization by allowing gradients to flow through the random sampling process. During training, the loss function—combining BCE for reconstruction fidelity and KLD for distribution alignment—is minimized. This dual-focus loss ensures that the VAE not only accurately reconstructs the input data but also maintains a meaningful and structured latent space with generative capabilities. The entire model is trained for 100 epochs.

## 4 Results

The models have been evaluated using two metrics: The IS and FID.The IS evaluates the quality of generated images based on how confidently a pre-trained Inception model classifies each image and the diversity across classes, measured by the entropy of class distributions. It is calculated using Eq.2.

$$\text{IS} = \exp(\mathbb{E}_x[\text{KL}(p(y|x)\|p(y))]) \tag{2}$$

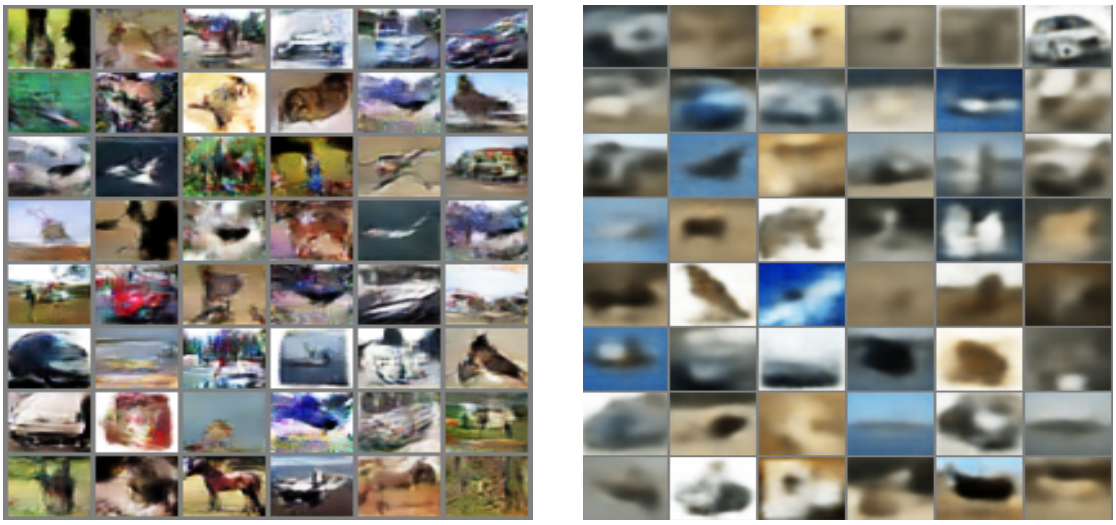
where $p(y|x)$ is the conditional class probability given an image $x$. $p(y)$ is the marginal class probability averaged over all images, and KL represents the Kullback-Leibler divergence. A higher IS indicates images are both clear and varied across classes.The FID measures the similarity between the distributions of real and generated images by comparing features from a pre-trained Inception network. It is calculated using Eq. 3.

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \tag{3}$$

where, $\mu_r, \mu_g$ being the feature-wise means of the real and generated images and $\Sigma_r, \Sigma_g$ their covariance matrices. A lower FID indicates higher image quality.

### 4.1 Comparative Analysis

Each model was evaluated on 10000 generated images, each image having the output dimension of 32x32. The DCGAN generator generates images from a noise vector. The VAE reconstructs images by encoding them into a 512-dimensional latent space and then decoding this representation to generate the output. Figure 5 displays a sample of images generated by the two models. Visual comparisons between Figure 5a (DCGAN) and Figure 5b (VAE) clearly indicate superior image quality from the DCGAN. Furthermore, Table 1 reveals that the DCGAN achieves a higher IS and a lower FID compared to the VAE.



(a) (b)

Figure 5: Images generated using the two models during testing. Fig. 5a and Fig. 5b show the sample of images generated by DCGAN and VAE.

.

VAEs often generate blurrier images compared to DCGANs [3]. This is partly due to the nature of the VAE's encoding-decoding process. During this process, details can be lost because the VAE smooths out the variability that might be critical for sharp image generation. The latent vectors in DCGANs are generated from a random noise distribution and are directly used to generate images without the encoding bottleneck present in VAEs. This allows DCGANs to often produce sharper and more detailed images. A higher IS for DCGAN indicates better discriminability and diversity of the generated images. Since the FID measures the distance between feature vectors of real images and generated images, the lower FID for DCGAN suggests that not only the high-level content (object shapes, primary structures) but potentially also finer details are better matched to the real images compared to those generated by VAEs.

## 5 Conclusion

In this project, it is demonstrated that DCGANs outperform VAEs in terms of image quality, as observed through visual inspection. This superiority is further supported by quantitative metrics, such as the IS and FID. The CIFAR-10 dataset, comprising low-resolution images (32x32), was utilized, which may constrain the generative capabilities of these models. Investigating different datasets that contain more images with higher resolution may improve the generative capabilities of both models, especially the DCGAN. While DCGANs and VAEs may not represent the pinnacle of current generative model technology, their simplicity, ease of interpretation, and hardware efficiency make them foundational in real-world applications and a basis for more complex generative models.

## References


[1] Radford, Alec & Metz, Luke & Chintala, Soumith Unsupervised representation learning with deep convolutional generative adversarial networks *arXiv preprint arXiv:1511.06434* (2015)

[2] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[3] Aleema Parakatta. "VAE v/s GAN — A case study."https://medium.com/@parakatta/vae-v-s-gan-a-case-study-b09c7169ac02 (2023)