GAN Model for Medical Image Generation: DFU Dataset Analysis

Model Architecture

Generator Network

Input: 100-dimensional noise vector (nz=100)

Architecture:

- Transposed convolutional layers with batch normalization and ReLU activation
- Progressive upsampling from 4x4 to 64x64 resolution
- Final layer uses Tanh activation to produce images in range [-1, 1]

Feature map sizes: 512 → 256 → 128 → 64 → 3 channels

Discriminator Network

Input: 64x64 RGB images (nc=3)

Architecture:

- Convolutional layers with batch normalization and LeakyReLU (0.2 slope)
- Progressive downsampling from 64x64 to 4x4 resolution
- Final layer uses Sigmoid activation for binary classification

Feature map sizes: 64 → 128 → 256 → 512 → 1 output

Training Configuration

- **Dataset:** DFU infection images from `/content/drive/MyDrive/dfu/PartB_DFU_Dataset/Infection/Aug-Negative`
- **Transformations:**
    - Resize to 64x64
    - Center crop
    - Normalize to [-1, 1] range
- **Hyperparameters:**
    - Batch size: 64
    - Learning rate: 0.0002 (both networks)
    - Beta coefficients: (0.5, 0.999)
    - Loss function: Binary Cross Entropy (BCELoss)
- **Training Duration:** 65 epochs

Training Metrics Analysis

Loss Progression (First 8 Epochs)

Epoch	Discriminator Loss (D)	Generator Loss (G)	D(x)	D(G(z)) Stage 1	D(G(z)) Stage 2
0	0.0447	8.1997	0.9656	0.0002	0.0003
1	0.0550	8.3065	0.9506	0.0002	0.0003
2	0.9907	4.9008	0.8769	0.4845	0.0103
3	0.6607	2.2563	0.8695	0.3883	0.1205
4	2.5532	5.0203	0.9861	0.8504	0.0097
5	1.6327	3.4366	0.9154	0.7073	0.0429
6	0.4198	2.1837	0.7569	0.0983	0.1312

Key Observations:

- **Initial Phase :** Discriminator dominates with near-perfect real/fake discrimination. This is evident from the very low Discriminator Loss (Dloss) values (0.0447, 0.0550) and high D(x) values (0.9656, 0.9506), indicating it accurately classifies real images as real. The extremely low D(G(z)) values (0.0002, 0.0003) show it's

also highly confident in classifying generated images as fake. The Generator Loss (Gloss) is initially very high, as the generator is struggling to produce convincing fakes.

- **Adversarial Balance :** The Generator begins to improve, leading to fluctuations in the discriminator's performance. The Dloss starts to increase (0.9907, 0.6607, 2.5532), and D(G(z)) values increase significantly (0.4845, 0.3883, 0.8504) as the discriminator is less consistently identifying generated images as fake. The Gloss also shows a decrease (4.9008, 2.2563, 5.0203), indicating the generator is becoming more effective. The jump in Dloss at epoch 4 suggests the generator might have made a breakthrough, making fakes that are harder to distinguish.

- **Instability :** The significant oscillations in both Dloss and Gloss (e.g., Dloss jumps from 2.5532 to 1.6327 then drops to 0.4198; Gloss from 5.0203 to 3.4366 then 2.1837) suggest that the model has not yet found a stable equilibrium. The discriminator and generator are continuously adjusting to each other's improvements, leading to an unstable training dynamic. This could indicate a need for more epochs, adjustments to learning rates, or other hyperparameter tuning to achieve convergence.

You've provided a clear and concise summary of the core hyperparameters used in your GAN model's training. This table effectively lists the critical settings that define your model's capacity and training process.

Let's break down each parameter and its significance:

Core Hyperparameters

Parameter	Value	Description
batch_size	64	Number of samples processed per batch during training.
image_size	64	Spatial dimensions of input/output images (64x64 pixels).

nz (noise dimension)	100	Size of the latent noise vector for generator input.
ngf	64	Number of feature maps in the generator's initial layers.
ndf	64	Number of feature maps in the discriminator's initial layers.
nc	3	Number of channels in images (RGB = 3).
num_epochs	65	Total training epochs.

Significance of Each Parameter:

1. batch_size **(64):**
    - **Impact:** Determines how many images are processed at once before the model's weights are updated.
    - **Observations:** A batch size of 64 is common for GANs. Larger batches can provide a more stable estimate of the gradient but require more memory. Smaller batches might introduce more noise but can help the model explore the loss landscape more effectively.
2. image_size **(64):**
    - **Impact:** The resolution of the generated and input images.
    - **Observations:** 64x64 pixels is a standard resolution for DCGANs and is a good starting point for computational efficiency while still allowing for reasonable detail. Higher resolutions (e.g., 128x128, 256x256) would require more complex architectures (like progressively growing GANs or BigGANs) and significantly more computational resources.
3. nz **(noise dimension) (100):**
    - **Impact:** The dimensionality of the random noise vector fed into the generator. This vector is the primary input that the generator transforms into an image.

    - **Observations:** 100 dimensions is a standard choice. A higher `nz` theoretically allows the generator to learn more complex latent representations and potentially generate more diverse images, but it also increases the generator's complexity.
4. `ngf` **(64):**
    - **Impact:** Controls the "width" or capacity of the generator network. It's typically the number of feature maps in the initial transposed convolutional layer that transforms the noise vector. The subsequent layers usually halve this number as they upsample (e.g., `ngf*8`, `ngf*4`, `ngf*2`, `ngf`).
    - **Observations:** 64 is a common starting point. Increasing `ngf` would allow the generator to learn richer features but would also increase its computational cost and memory footprint.
5. `ndf` **(64):**
    - **Impact:** Controls the "width" or capacity of the discriminator network. It's typically the number of feature maps in the initial convolutional layer that processes the input image. Subsequent layers usually double this number as they downsample.
    - **Observations:** Similar to `ngf`, 64 is a standard choice. A larger `ndf` gives the discriminator more capacity to learn to distinguish real from fake images, which can be beneficial but might also lead to the discriminator becoming too powerful too quickly, making the generator's job harder.
6. `nc` **(3):**
    - **Impact:** The number of color channels in the images.
    - **Observations:** 3 indicates RGB (Red, Green, Blue) images, which is standard for natural and medical images unless they are grayscale.
7. `num_epochs` **(65):**
    - **Impact:** The total number of times the entire dataset is passed forward and backward through the neural networks.
    - **Observations:** 65 epochs is a moderate training duration. As seen from your "Instability" observation in the loss progression, 65 epochs might not have been enough for the DCGAN to fully stabilize and converge, especially given the inherent challenges of DCGAN training. Longer training or hyperparameter tuning might be needed to achieve better stability and image quality.

These hyperparameters represent the fundamental configuration choices for your DCGAN. Their values significantly influence the training dynamics, convergence, and the ultimate quality of the generated DFU images.

Quantitative Metrics:

F1 score:- 0.68 (Moderate)

Precision 0.62(Average)

F1 Score ranges from **0 to 1**, where:

- **1** is the best possible F1 Score, indicating perfect precision and recall (no false positives or false negatives).
- **0** is the worst possible F1 Score, meaning the model failed to identify any true positives.

Here are 5 simple points comparing DCGAN and WGAN, keeping in mind the context of a potential research paper:

1. **Core Problem Addressed:** DCGAN provided a foundational convolutional architecture for GANs, while WGAN was developed primarily to solve the training instability and mode collapse issues often seen in DCGANs.
2. **Loss Function Difference:** DCGAN uses a binary cross-entropy (log) loss, which can lead to vanishing gradients. WGAN uses the Wasserstein distance, aiming for smoother gradients and more stable training.
3. **Mode Collapse Mitigation:** WGANs are generally better at avoiding mode collapse (where the generator produces limited varieties of samples) compared to DCGANs, leading to more diverse outputs.
4. **Meaningful Loss Metric:** The "critic" loss in WGAN tends to correlate better with the visual quality of generated samples than the discriminator loss in DCGAN, offering a more reliable training progress indicator.
5. **Architectural Modifications:** While both are convolutional, WGAN introduces key changes to the discriminator (often called a critic), such as removing the final sigmoid layer and using techniques like weight clipping or gradient penalty to enforce constraints.

- Stability: WGAN-GP is generally more stable than DCGAN.
- Mode Collapse: DCGAN is prone to it; WGAN variants mitigate this.
- Medical Applications: WGAN-GP is often preferred for generating high-fidelity medical images.
- Implementation: Switching from DCGAN to WGAN-GP involves:
- Removing Sigmoid in the discriminator (use linear output).
- Adding a gradient penalty term (for WGAN-GP).
- Using Wasserstein loss (critic_loss = D(fake) - D(real)).