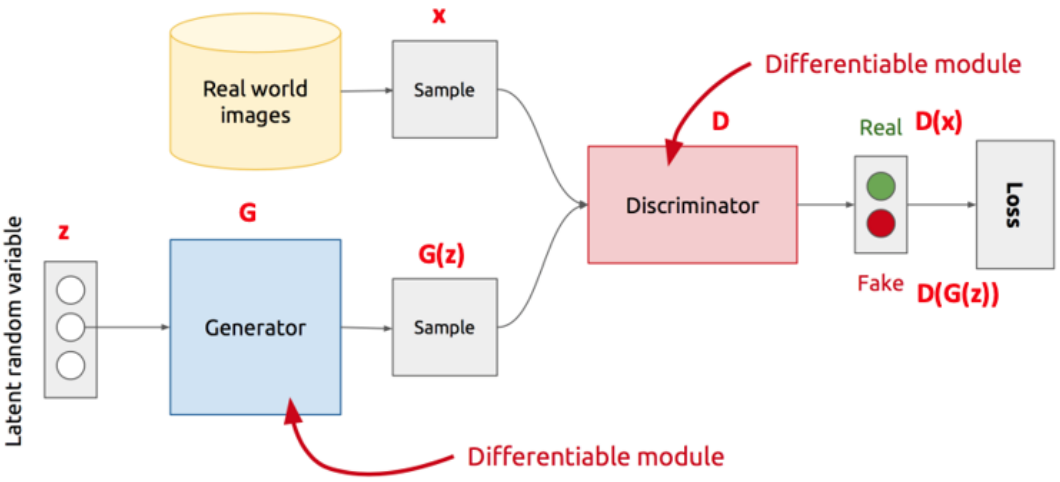
Shahaf Wieder

HW4 - Generative Models

1. Model Architecture

The implemented model follows the **Generative Adversarial Network (GAN)** framework, consisting of a **Generator (G)** and a **Discriminator (D)**. The architecture is as follows:



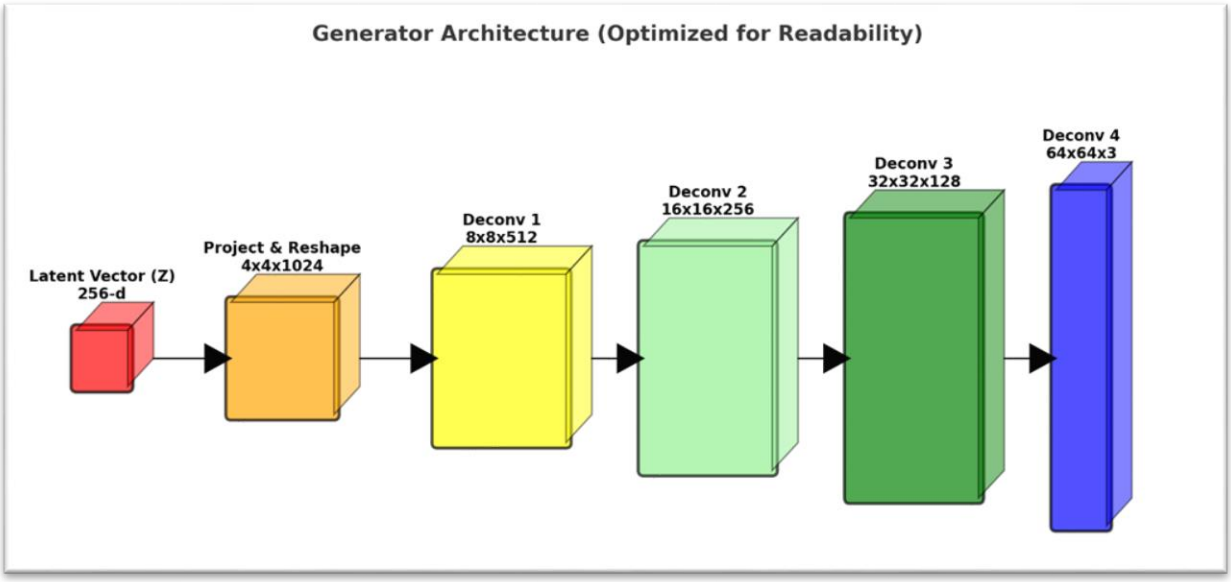
- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Generator Architecture

The **generator** transforms a latent vector **z** (sampled from a Gaussian distribution) into a realistic image through a series of **transposed convolutional layers**:

- **Input**: A 256-dimensional latent vector Z
- **Projection & Reshape**: 4×4×1024
- **Deconv 1**: 8×8×512 (ConvTranspose2d + BatchNorm + ReLU)
- **Deconv 2**: 16×16×256 (ConvTranspose2d + BatchNorm + ReLU)
- **Deconv 3**: 32×32×128 (ConvTranspose2d + BatchNorm + ReLU)
- **Deconv 4**: 64×64×3 (ConvTranspose2d + Tanh activation)
- **Output**: A 64×64×3 RGB image

The final activation function is **Tanh**, ensuring pixel values are scaled between -1 and 1.

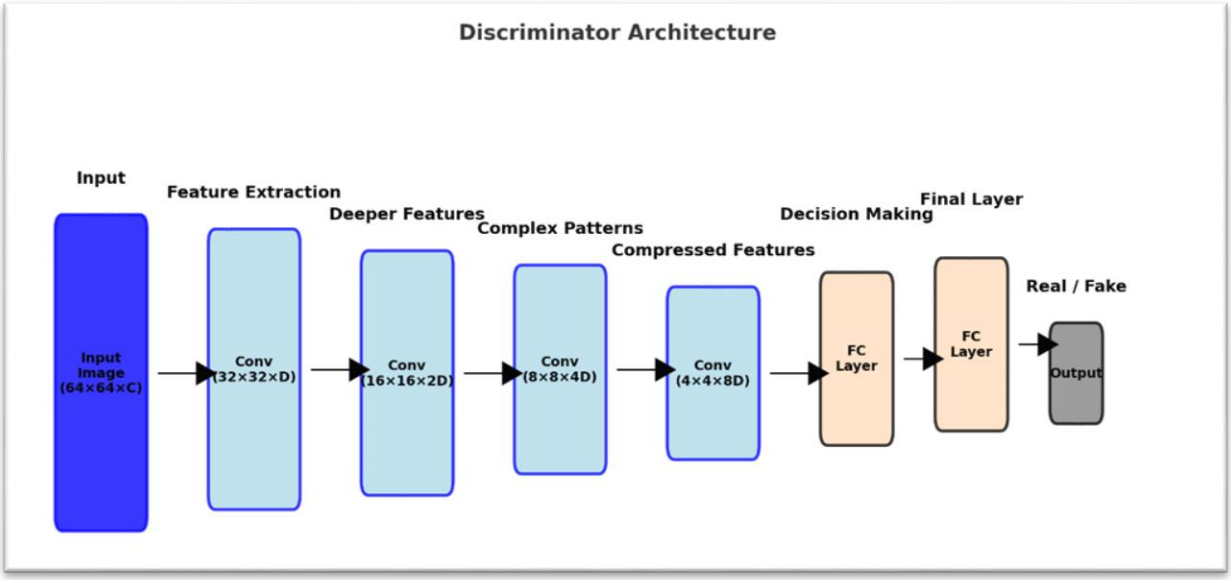


Discriminator Architecture

The **discriminator** is a convolutional neural network (CNN) that classifies whether an input image is **real** or **fake**:

- **Input**: 64×64×3 RGB image
- **Conv 1**: 32×32×64 (Conv2d + LeakyReLU)
- **Conv 2**: 16×16×128 (Conv2d + BatchNorm + LeakyReLU)
- **Conv 3**: 8×8×256 (Conv2d + BatchNorm + LeakyReLU)
- **Conv 4**: 4×4×512 (Conv2d + BatchNorm + LeakyReLU)
- **Conv 5**: 1×1×1 (Conv2d + Sigmoid)
- **Output**: A single value indicating the probability of the input being real or fake.

The activation function in the last layer is **Sigmoid**, which outputs a probability score.



2. Training Procedure

The GAN is trained using the **min-max adversarial loss**:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Each training step consists of:

1. **Discriminator Training**:
    - Given real images x, compute D(x) and maximize log(D(x)).
    - Given fake images G(z), compute D(G(z)) and log(1−D(G(z))).
2. **Generator Training**:
    - Generate fake images G(z).
    - Maximize log(D(G(z))), i.e., make the discriminator think generated images are real.

3. Hyperparameters

- **Latent Space Dimension**: z=256
- **Batch Size**: 64
- **Number of Channels**: 3 (RGB images)
- **Generator Feature Maps**: 64
- **Discriminator Feature Maps**: 64
- **Epochs**: 300
- **Learning Rate**: 0.0002
- **Adam Optimizer**:
    - β1=0.5
    - β2=0.999

4. Optimization & Regularization

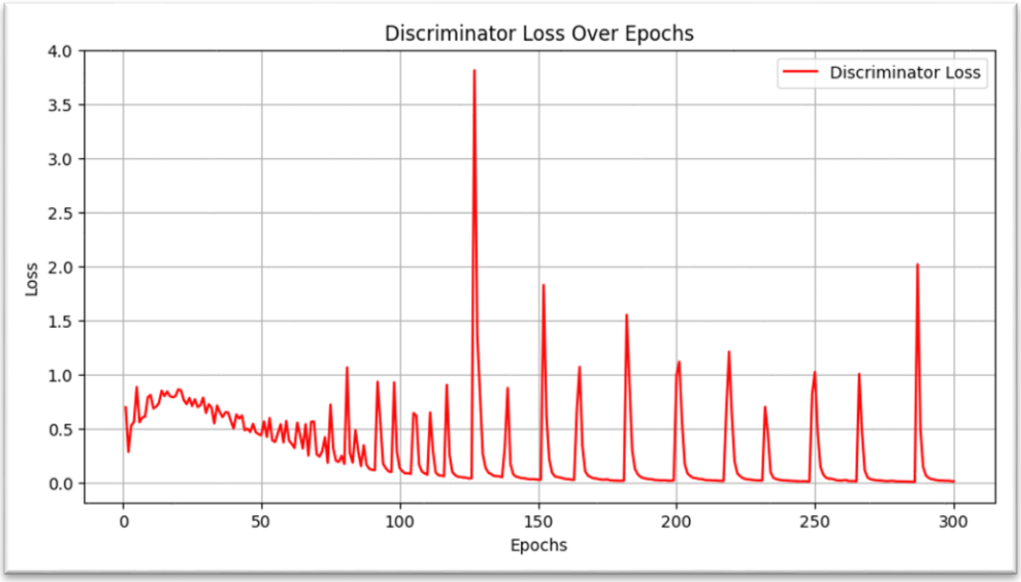
- **Weight Initialization**:
    - Normal distribution with mean 0 and std 0.02 for convolutional layers.
    - BatchNorm layers initialized with mean 1 and std 0.02.
- **Leaky ReLU in Discriminator**: Slope = 0.2
- **Batch Normalization**: Applied to stabilize training.

5. Implementation Details

- **Dataset**: Images were preprocessed using **Resize (64x64), CenterCrop, Normalize ([0.5, 0.5, 0.5])**.
- **Training Setup**: Implemented in **PyTorch**, utilizing CUDA when available.

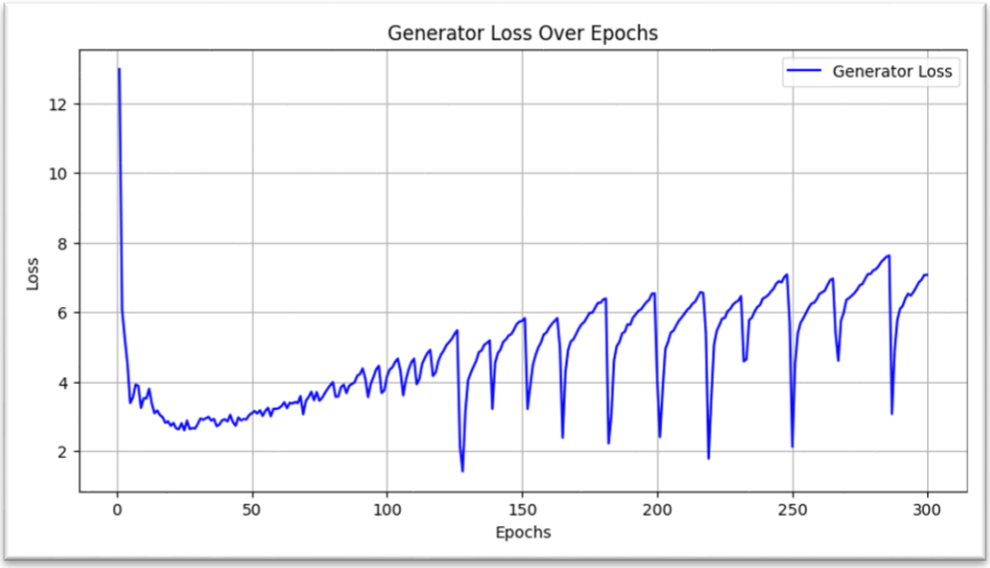
Discriminator Loss Analysis (Red Plot)

- Initially, the **discriminator loss is relatively high**, which is expected because the discriminator is learning to differentiate between real and fake images.
- The loss **gradually decreases**, indicating that the discriminator is getting better at distinguishing real from fake samples.
- However, **sharp spikes** occur periodically, which suggests:
    - The generator temporarily improves and fools the discriminator.
    - The discriminator adapts quickly and corrects itself, leading to sudden loss increases.
- The overall **trend shows oscillations**, which is typical in GAN training, as the generator and discriminator compete.
- The **final loss values are quite low**, indicating that the discriminator has learned to identify fake samples efficiently.



Generator Loss Analysis (Blue Plot)

- At the beginning, the **generator loss is extremely high (~13)**, indicating that the generator is producing very poor images.
- As training progresses, the **loss decreases but later begins to oscillate and increase**.
    - This suggests that the generator is struggling to fool the discriminator.
- Around **epoch 100 onwards**, periodic **sharp dips** appear, indicating:
    - Some iterations where the generator successfully fools the discriminator (lower loss).
    - The discriminator quickly adapts, pushing the generator loss back up.
- The overall **increasing trend after 100 epochs** suggests the **discriminator is overpowering the generator**, preventing it from improving significantly.



Interpretation & Training Stability

- The observed **instability** (loss fluctuations) is a common issue in GAN training due to the adversarial nature of the model.
- The **generator's increasing loss indicates that the discriminator might have become too strong**, making it difficult for the generator to produce realistic images.

Summary of Attempts and Conclusions

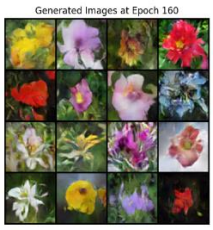
1. Overview

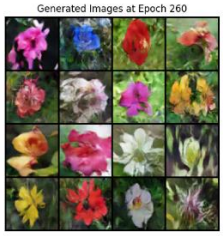
In this HW4, a Generative Adversarial Network (GAN) was trained on a dataset of flower images. The model was evaluated under two different settings of the **latent space dimension**: **100** and **256**. We were interested to assess the effect of latent space size on the quality, diversity, and realism of generated images.

2. Comparison of Results

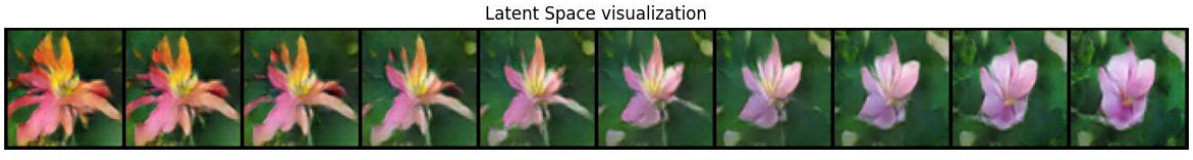
2.1 Latent Dimension = 100

- **Epoch 20**: The generated images were blurry, lacking clear structures and well-defined colors.
- **Epoch 80**: Image quality improved significantly, with distinguishable floral patterns, although still containing some artifacts.
- **Epoch 100-160**: The generated flowers became sharper, with more vibrant colors and diversity.
- **Epoch 180-300**: The model captured fine textures, but some images still exhibited noticeable distortions. However, the overall quality and diversity of the flowers were satisfactory.



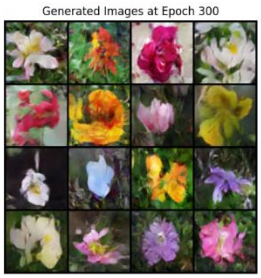


Latent Space Interpolation: The transition between different latent points was smooth, but some intermediate images displayed unnatural blending artifacts.



2.2 Latent Dimension = 256

- **Epoch 20**: Images were slightly more detailed than those from the latent dimension 100 experiment, but still blurry.
- **Epoch 80**: The generated flowers displayed more realistic textures, sharper edges, and better color blending.
- **Epoch 100-160**: Fine details, such as petal edges and vein structures, became visible. The colors appeared more natural and balanced.
- **Epoch 180-300**: The generated flowers showed significant improvement, with distinct species characteristics, higher color fidelity, and better spatial consistency.



Latent Space Interpolation: The transition between different points in latent space was smoother, with fewer blending artifacts, indicating better representation of the learned features.

3. Observations and Analysis

1. **Higher Latent Space Dimension Improves Image Quality**
    - The images generated with **latent dim = 256** were more realistic, with richer textures and sharper details than those produced with **latent dim = 100**.
    - A larger latent space provides more expressive power for the generator, allowing it to capture fine-grained details better.
2. **Training Stability and Convergence**
    - The discriminator loss exhibited oscillations in both cases, indicating the adversarial nature of GAN training.
    - The generator loss, particularly in the **latent dim = 100** case, showed higher fluctuations, suggesting that it struggled more to generate realistic images.
    - The model with **latent dim = 256** converged more smoothly and exhibited better training stability.
3. **Mode Collapse and Diversity**
    - In both cases, **mode collapse was not evident**, meaning the generator was able to produce diverse floral images rather than a limited set of repeating patterns.
    - The diversity in generated images was higher with **latent dim = 256**, as it allowed for more variation in color and structure.
4. **Smoothness of Latent Space Interpolation**
    - The **latent dim = 256** model demonstrated a **smoother and more natural transition** between different points in latent space.
    - The **latent dim = 100** model had more abrupt changes and blending artifacts between latent points, suggesting that the representation capacity was more constrained.

4. Challenges and Improvements

Challenges Encountered:

- Some generated images still had visible artifacts, especially in early epochs.
- The discriminator learned faster than the generator at times, leading to suboptimal training dynamics.

Potential Improvements:

- **Use Progressive Growing of GANs (PGGAN):** This can improve fine-grained texture learning.
- **Spectral Normalization in Discriminator:** This helps stabilize training and prevents discriminator overpowering.
- **Adjust Learning Rate Scheduling:** Dynamic adjustment of learning rate based on convergence metrics could help stabilize training.
- **Experiment with Different Latent Space Sizes:** Trying values between 100 and 256 (e.g., 128, 192) may provide a good balance of expressiveness and training stability.

5. Conclusion

- **Larger latent space (256) led to more detailed and realistic images**, demonstrating its advantage in capturing fine features.
- **Training stability was slightly better with larger latent dimensions**, as it allowed for smoother generator updates.
- **Both models successfully captured floral patterns**, but the **256-dimensional latent space** provided a significant improvement in **sharpness, color richness, and diversity**.