Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Department of Computer Engineering

- ***Submitted by:***
- ***Name: Kshitij Narkhede PRN : 22110941***
- ***Roll No: 322047 (TY-B) Batch: 2***

Assignment : 4

Problem Statement: Apply GAN for generating new images

Dataset: Fashion MNIST

Software Used: Google Co-laboratory, Python

Libraries: TensorFlow (open-source ML Library) , keras, matplotlib , PTL, IPython imageio

Theory and Understanding:

Following steps and training followed to build GAN.

1. **Load & Preprocess the Dataset:**
   - This dataset consists of images of clothing items with 70,000 images for training and 10,000 images for testing. Each image is 28x28 pixels and grayscale.
   - Then, it normalizes the pixel values from the range [0, 255] to the range [-1, 1] by subtracting 127.5 and dividing by 127.5.

2. **Building Models – Generator and Discriminator :**
   - This model takes a random noise vector of size 100 as input.
   - It uses several convolutional transpose layers with LeakyReLU activations and batch normalization to up sample the noise vector into a 28x28 grayscale image.
   - It uses several convolutional layers with LeakyReLU activations and dropout to downsample the image and extract features.
   - Finally, it uses a single neuron with a sigmoid activation function to classify the image as real (1) or fake (0).

3. **Loss Functions:**
   - **Discriminator Loss:** This measures how well the discriminator can distinguish between real and fake images. It is calculated as the sum of the binary cross-entropy loss between the discriminator's predictions on real images and fake images.
   - **Generator Loss:** This measures how well the generator can fool the discriminator. It is calculated as the binary cross-entropy loss between the discriminator's predictions on the generated images

4. **Define the Optimizers:**
   - The Adam optimizer is used for both the generator and discriminator with a learning rate of 1e-4.

5. **Training :**
   - The number of epochs (50) and the noise dimension (100).

- In each epoch, it iterates over the training data in batches.

  For each batch:
  - The generator is used to create fake images from the noise.
  - The discriminator is used to classify both real and fake images.
  - The discriminator loss and generator loss are calculated.
  - The gradients of the loss functions are used to update the weights of the generator and discriminator using their respective optimizers.

6. **Model Checkpointing:**
   - Every 15 epochs, the current state of the generator and discriminator (including their weights and optimizers) is saved as a checkpoint. This allows resuming training from a saved point if needed.
   - After training, Discriminator discarded and Generator is well trained . Find Best checkpoint where Generator is best fit .

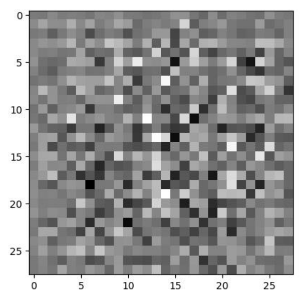
Hyperparameter Tunning:

- In Generator Model we use Leaky Relu because , Leaky ReLU is computationally also it helps the generator learn from a wider range of inputs and avoids getting stuck with inactive neurons.
- This can lead to a more robust and efficient training process for the generator . The noise vector used as input to the generator has a dimension of 100. This size allows for sufficient variability in the generated images without being too computationally expensive.
- A relatively low learning rate is often used in GANs to ensure more stable training and prevent model updates from being too drastic, which can lead to oscillations or divergence.
- It's important to balance the learning of the generator and discriminator. Hyperparameters can be adjusted to prevent one model from overpowering the other, leading to better overall results.

Result and Discussion :

https://colab.research.google.com/drive/1fS7IYVfXJVUbtDSaXUGGyth_o2nG6D4Y?usp=sharing

Image Generated before Training :



After every Epoch , generator tries to generate more likely real image.

Compared to real-world clothing images, Fashion MNIST images are smaller (28x28 pixels) and lack details like textures or intricate patterns. GANs might struggle to capture these complexities when generating new images.

Model collapse can occur where the generator gets stuck in a loop, producing only a limited set of clothing types or variations. This can be particularly problematic for a dataset with diverse categories like Fashion MNIST.

Conclusion:

GANs operate in an unsupervised manner, eliminating the need for manual feature extraction. This makes them efficient and adaptable to various image domains. GANs can produce highly realistic and believable images that closely resemble real-world data. This makes them ideal for applications like creating new product prototypes, generating artistic content, or supplementing image datasets.

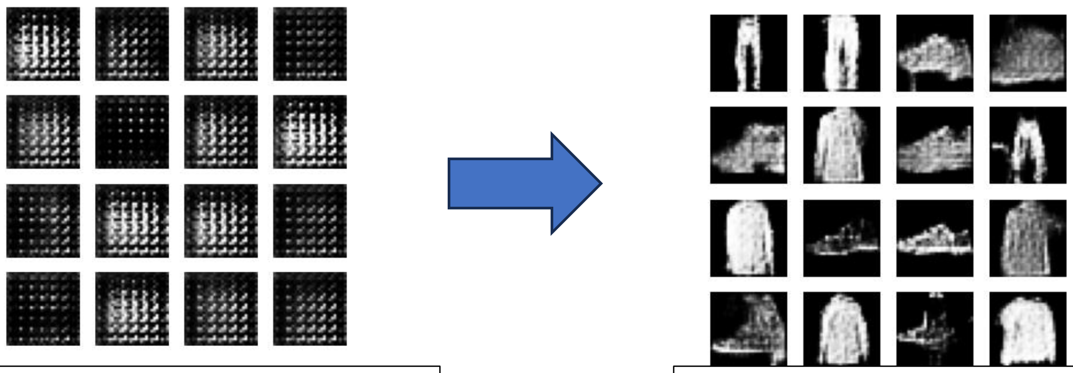


Image generated by generator at 1st Epoch

Image generated by generator at 50th Epoch