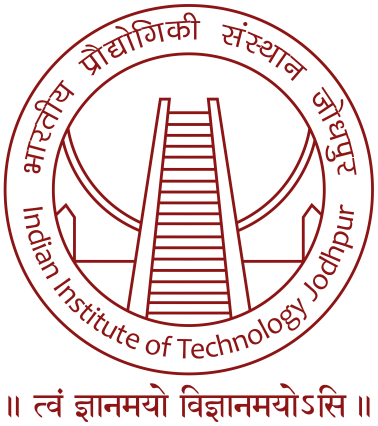
Indian Institute of Technology, Jodhpur



Constructing a Conditional GANs

CSL7590: Deep Learning (Spring, 2024)

Assignment: 4

Due Date: April 14, 2024

Name:- Sahil, Aman Kanshotia and Sougata Moi
Roll No.:- M21MA210, M21MA201 and M23MAC008

M.Tech and M.Sc - M.Tech (Data and Computational Sciences)
Department of Mathematics

Contents

List of Figures

List of Tables

1 About the Dataset

We have the ISIC 2016 Dataset. The dataset contains the following composition:

- **Training Data**: Training images (9015).
- **Training Labels**: Training labels.
- **Training Sketch**: Unpaired and paired training sketches.
- **Test Data**: Test images (1000) .
- **Test Labels**: Test labels.
- **Test Sketch**: Unpaired and paired test sketches.

2 Methodology

To obtain the objective, we used WGAN GP (gradient penalty). This ensures the training stability.

- The gradient_penalty function in the code is used to compute the gradient penalty term for WGAN loss and also to address some of the training challenges faced by traditional GANs.
- This penalty term is introduced to enforce the lipschitz constraint on the discriminator (Critic) model that ensures that the norm of its gradients should be bounded by 1 everywhere which helps in ensuring stable training and convergence of the GAN.
- We faced problem of Gradient Exploiting that lead to unstable training and mode collapse(the generator producing similar samples).Lipschitz constraint on the critic provides us better gradients for the generator, allowing for more stable and efficient training.
- It also ensures that the wasserstein distance used as the objective function is continuous and differentiable almost everywhere.This makes the optimization process more well behaved and lead to Improved convergence of model
- The gradient penalty provides a more effective and principled way to enforce the lipschitz constraint without directly clipping the weights.
- The concept of optimal transport and the KR duality provides a way to approximate the wasserstein distance between the real and generated distributions, which is a more meaningful metric than the JS divergence used in traditional GANs.
- By this term the training process is stabilized, and the Generator receives better gradients, leading to improved convergence and higher-quality generated samples.

3 Experimental Setup

To achieve the objective of generating realistic images from sketches using a conditional Generative Adversarial Network (GAN), the following experimental setup will be implemented:

1. **Dataset Preparation**: Incorporated the given training images, labels and sketches (utilizing both paired and unpaired in our assignment). The results attached are for unpaired sketches.
2. **Model Architecture**: The architecture of the completed is described in the next section.
3. **Dimension Scaling**: Adjusted the dimensions of input sketches and generated images to 64 $\times$ 64
4. **Training Procedure**: Trained the conditional GAN model using the training data and labels. Optimize the model parameters using Adam Optimizer.
5. **Evaluation Metrics**: Evaluated the performance of the generated images using Frechet Inception Distance (FID) and Inception Score (IS).
6. **Testing (Inference) Phase**: Tested the trained model using the test data and labels. Then, generated images from the test sketches and compare them with the ground truth to assess the model's performance.

4 Architecture of the Model

4.1 Architectural Components of the Generator

1. **Input**: The generator accepts two inputs: sketches, which serves the purpose of random noise vectors and corresponding labels. Labels are embedded into a continuous vector space via an Embedding layer, analogous to the discriminator.
2. **Embedding Layer**: The embedding layer transforms labels into vectors of dimension 'embed_size', aligning with the noise vector dimensionality. This embedding facilitates conditional image generation based on the provided labels.
3. **Transposed Convolutional Layers**: The generator network comprises transposed convolutional layers, also referred to as de-convolutional or up sampling layers. Each transposed convolutional layer integrates batch normalization and ReLU activation functions to foster stable and effective learning.
4. **Convolutional Blocks**: This block structure is iteratively employed within the generator, enhancing modularity and scalability.
5. **Final Convolutional Layer**: The terminal convolutional layer transmutes feature maps into the desired output channels, typically mirroring the image's channel count (e.g., 3 for RGB). This layer finalizes the image generation process, yielding the generator's output.

6. **Up-sampling**: The Upsampling layer rescales the output feature maps to achieve the target resolution through bi linear interpolation.

7. **Output Activation**: The generator's output undergoes a **tanh** activation function.

4.2 Architectural Components of the Discriminator

1. **Conditional Discrimination**:

   - The architecture integrates label information into the discriminator network, facilitating conditional discrimination.
   - Conditioning on labels enables the discriminator to differentiate between real and synthesized images tailored to specific classes or categories.

2. **Hierarchical Feature Extraction**: The architecture employs a series of convolutional layers for extracting hierarchical features from input images. Through successive convolutional layers, the network captures progressively abstract representations of the input, enhancing discriminative capabilities.

3. **Instance Normalization**: Instance normalization is applied post each convolutional layer. This technique stabilizes the training by normalizing activations for each instance independently, potentially enhancing convergence and generalization.

4. **Leaky Rectified Linear Unit (ReLU) Activation**: Leaky ReLU activation functions are employed throughout the discriminator's architecture.

5. **Embedding Layer**: An embedding layer is utilized to project label information into a continuous vector space. Embedding labels alongside image data facilitates the learning of meaningful, class-specific representations, thereby aiding the conditional discrimination task.

6. **Final Classification Layer**: The terminal convolutional layer aggregates feature maps into a scalar value. This scalar denotes the discriminator's confidence level regarding the authenticity of the input image, serving as the foundation for adversarial loss computation during training.

5 Training with WGAN GP

5.1 Training Loop

- Specified the number of epochs as:100 and the sketches, images and their labels are loaded.

Hyper parameter	**Value**
λ (Gradient Penalty Weight)	100
Critic Iterations	5
Learning Rate	1×10^{-4}

Table 1: Hyper parameters for WGAN Training

- A batch of fake images is generated by the Generator using random noise as sketches and labels. The Critic's output scores for the real images and fake images are computed.
- The gradient penalty term (gp) is calculated using the gradient penalty function, which enforces the Lipschitz constraint on the Critic.
- The WGAN loss for the Critic is computed as the negative of the Critic's score for real images minus the Critic's score for fake images, plus the gradient penalty term weighted by a hyper parameter.

$$L = \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x})] - \mathbb{E}_{x\sim P_r}[D(x)] + \lambda \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \quad (1)$$

- Here, $D(x)$, $D(\tilde{(x)})$ is real and generated distribution respectively, λ is the hyper parameter, and we used $\lambda = 100$, in our task.
- The Critic's gradients are computed, and the optimizer performs a step to update the Critic's weights.
- After the Critic updates, the code performs an update to the generator model. The Critic's output scores for the previously generated fake images are computed..
- The WGAN loss for the Generator is computed as the negative of the Critic's scores for the fake images.
- The generator's gradients are computed, and the optimizer performs a step to update the generator's weights.

5.2 Advantages of Inner Loop

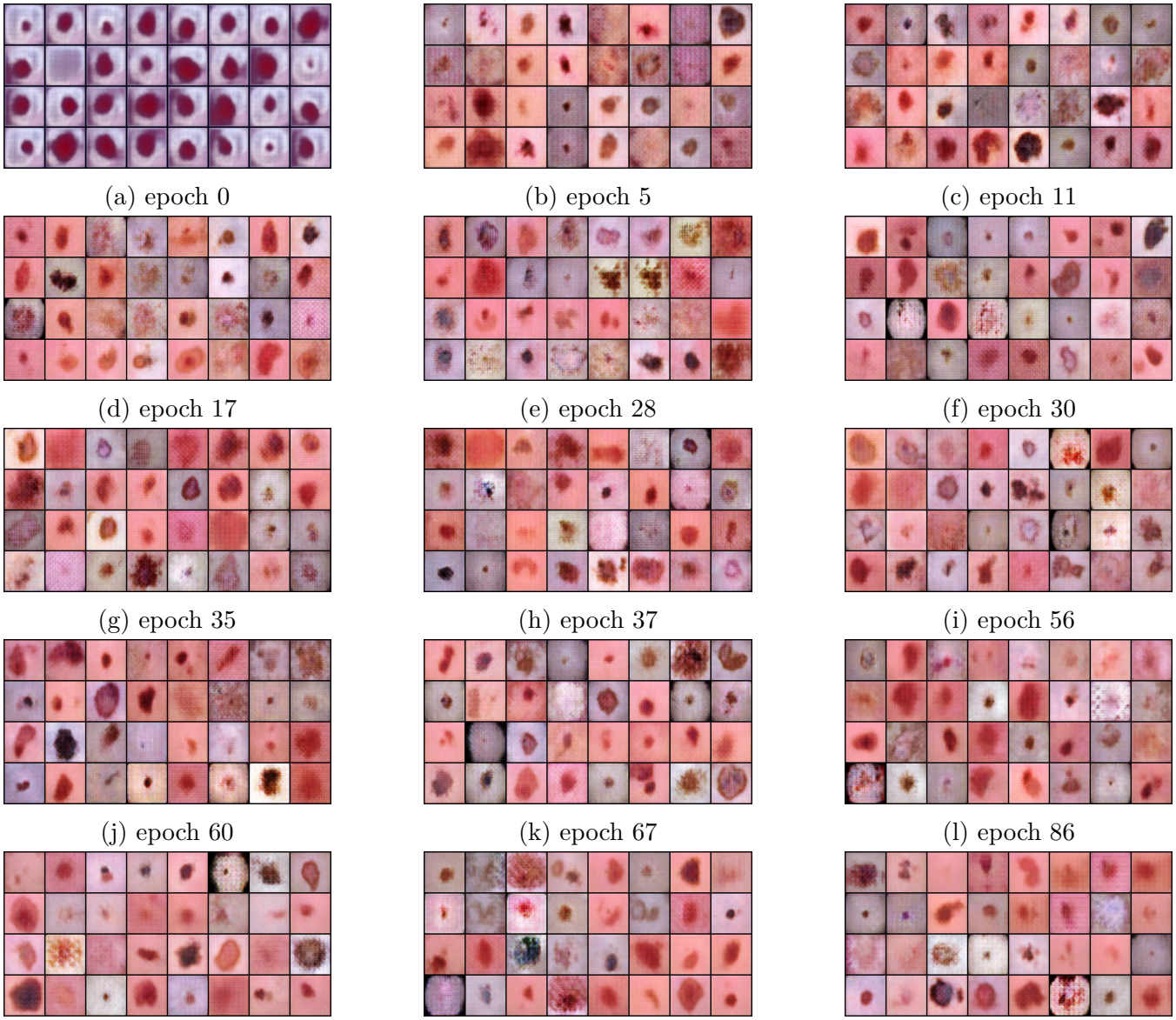
- **Critic Optimization**: Multiple Critic updates facilitate accurate Wasserstein distance estimation, providing meaningful gradients for generator updates.
- **Stabilization**: Increased Critic optimization reduces training instability and mode collapse risks.
- **Lipschitz Constraint Enforcement**: Gradient penalty enforces Lipschitz constraint, promoting stable convergence and high-quality samples.
- **Justification**: The WGAN formulation is based on the concept of optimal transport and the Kantorovich-Rubinstein duality. The inner loop, where the Critic is updated to better approximate the Wasserstein distance, aligns more closely with the theoretical foundations of the WGAN approach, potentially leading to better convergence properties.
- **Computational Efficiency**: While requiring more Critic updates, the inner loop can accelerate convergence and enhance sample quality.

6 Results

6.1 Experiment 1

Here are the results for some generated images by the model.

- Images generated on training



(a) epoch 0 (b) epoch 5 (c) epoch 11

(d) epoch 17 (e) epoch 28 (f) epoch 30

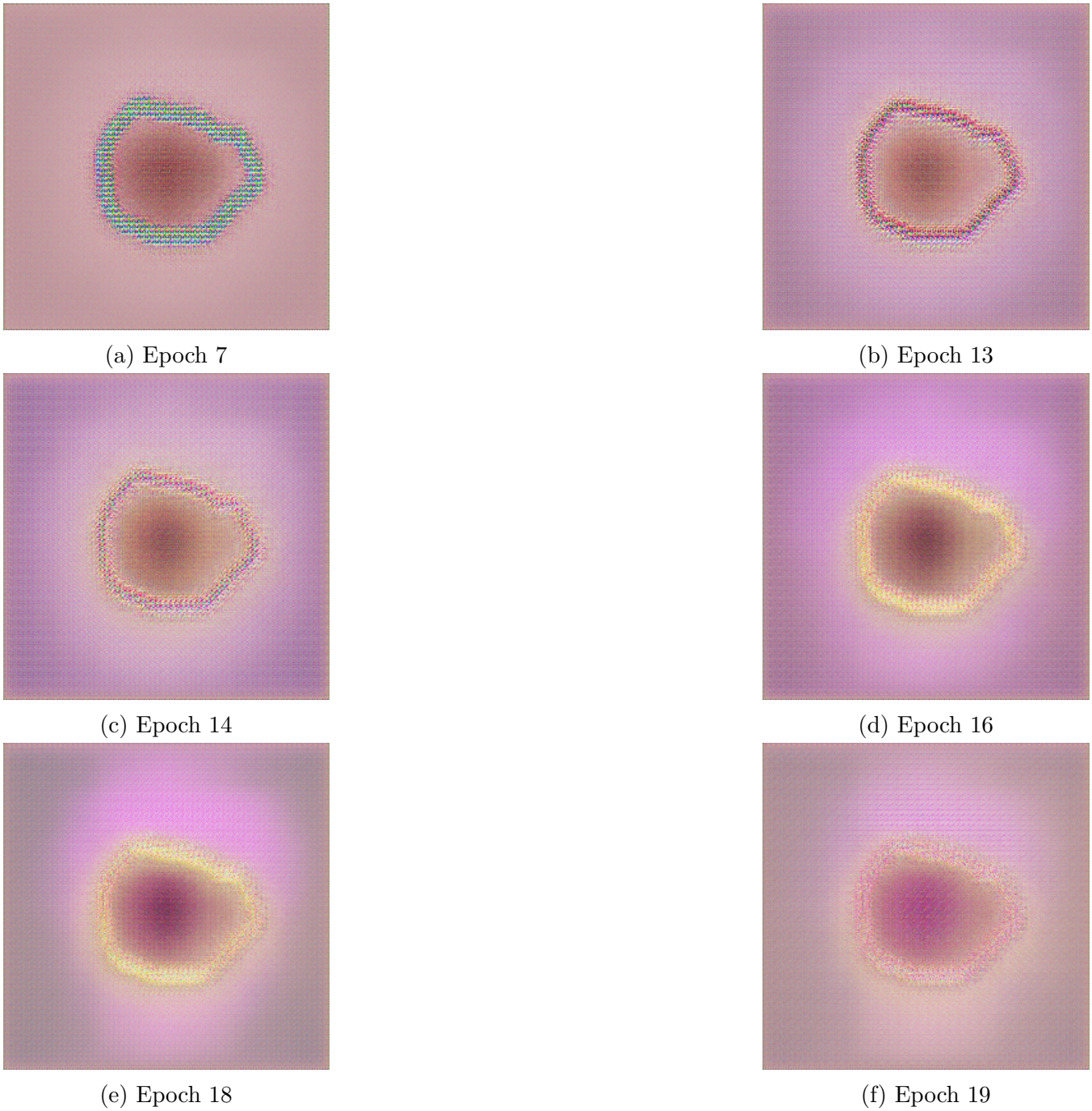
(g) epoch 35 (h) epoch 37 (i) epoch 56

(j) epoch 60 (k) epoch 67 (l) epoch 86

(m) epoch 91 (n) epoch 95 (o) epoch 99

Figure 1: Generated Images at Different Epochs with unpaired sketches

- Images generated on training with paired sketches



(a) Epoch 7 (b) Epoch 13

(c) Epoch 14 (d) Epoch 16

(e) Epoch 18 (f) Epoch 19

Figure 2: Generated Images for train dataset with paired sketches

- The loss curves for both generator and discriminator (for unpaired sketches) is as follows:

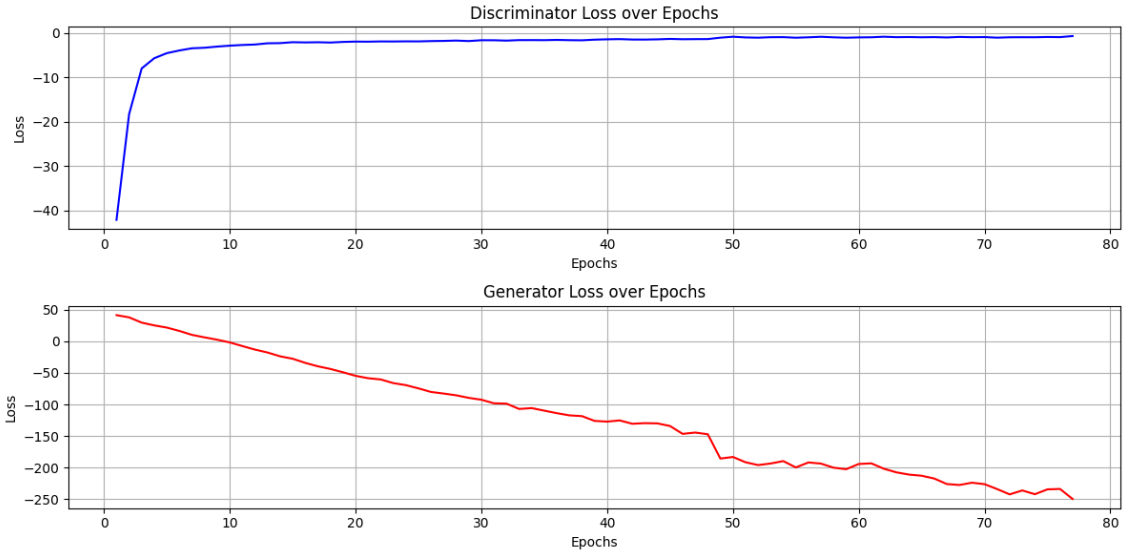


Figure 3: Loss curves

- The wandb plots for the training process with unpaired sketches is as follows:

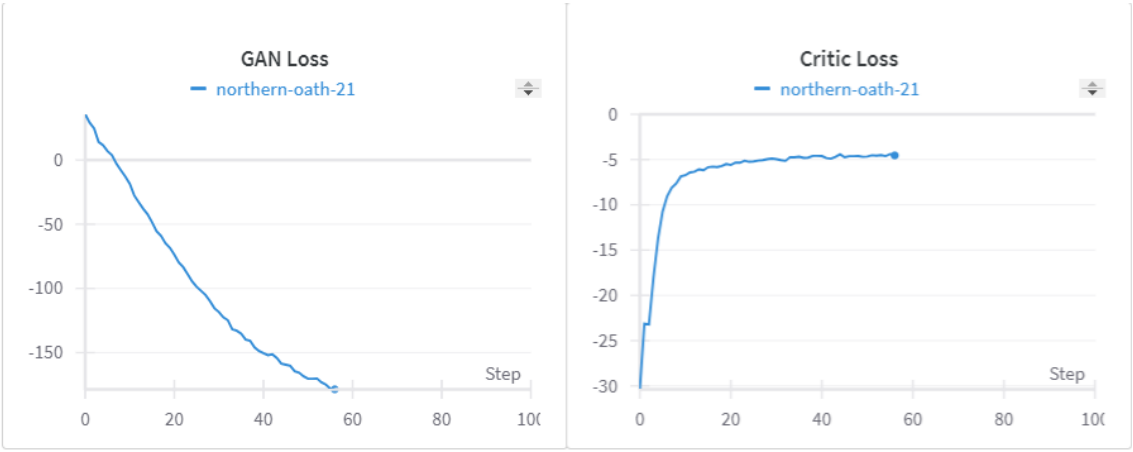


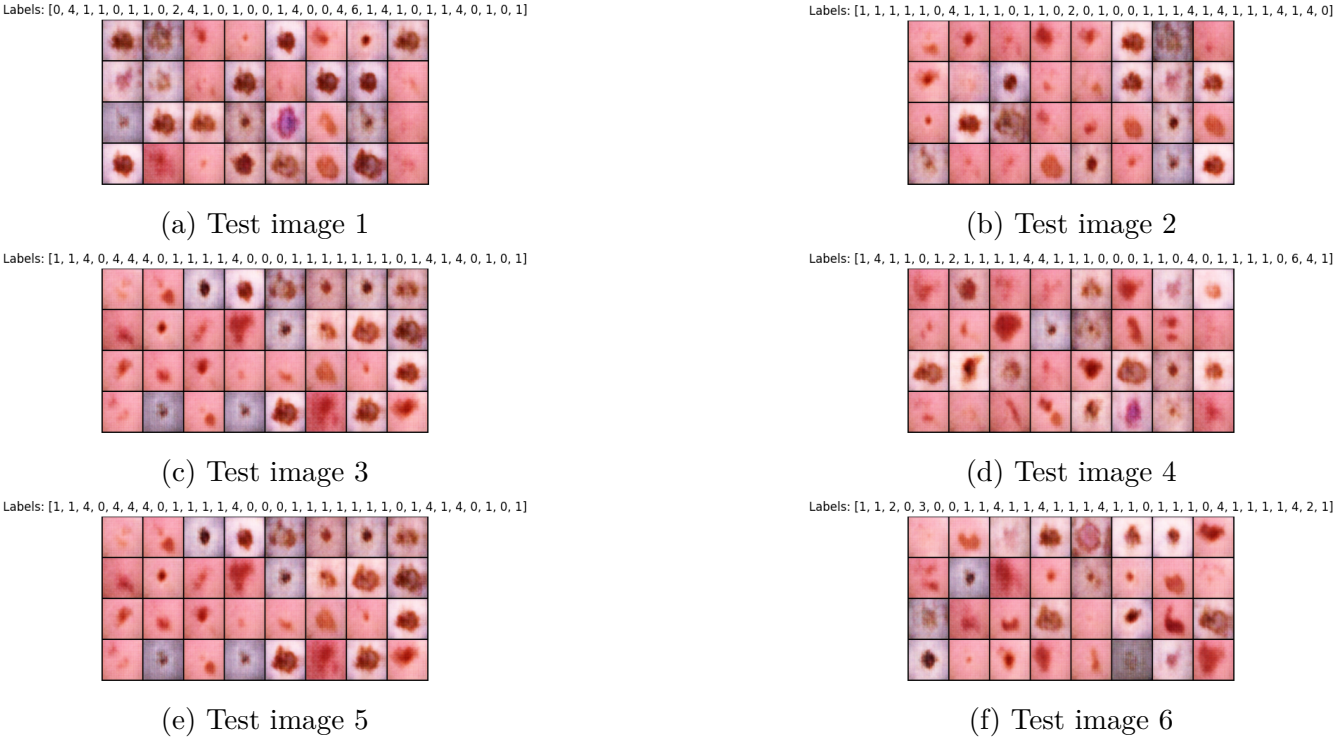
Figure 4: Wandb Plots

- The FID and Inception scores are as follows:

Score	Dataset	**Value**
Inception Score (Without fine tuning)	Test dataset	0.0023
Inception Score (Without fine tuning)	Generated dataset	0.0023
Inception Score (With fine tuning)	Test dataset	1.6
Inception Score (With fine tuning)	Generated dataset	1.43
FID Score	—	69.9046

Table 2: Test Scores

- Images generated on testing dataset



(a) Test image 1

(b) Test image 2

(c) Test image 3

(d) Test image 4

(e) Test image 5

(f) Test image 6

Figure 5: Generated Images for test dataset

6.2 Experiment 2

We used Efficient net and fine tuned it as per our task, and named our classifier as **"Fine tuned efficient net"**. This classifier is trained for 15 epochs.

Dataset	**Accuracy (%)**
Training	85.07
Validation	74.89
Test Data (Real)	63
Test Data (Generated Image)	39.41

Table 3: Classification Results Using Fine-Tuned Efficient Net

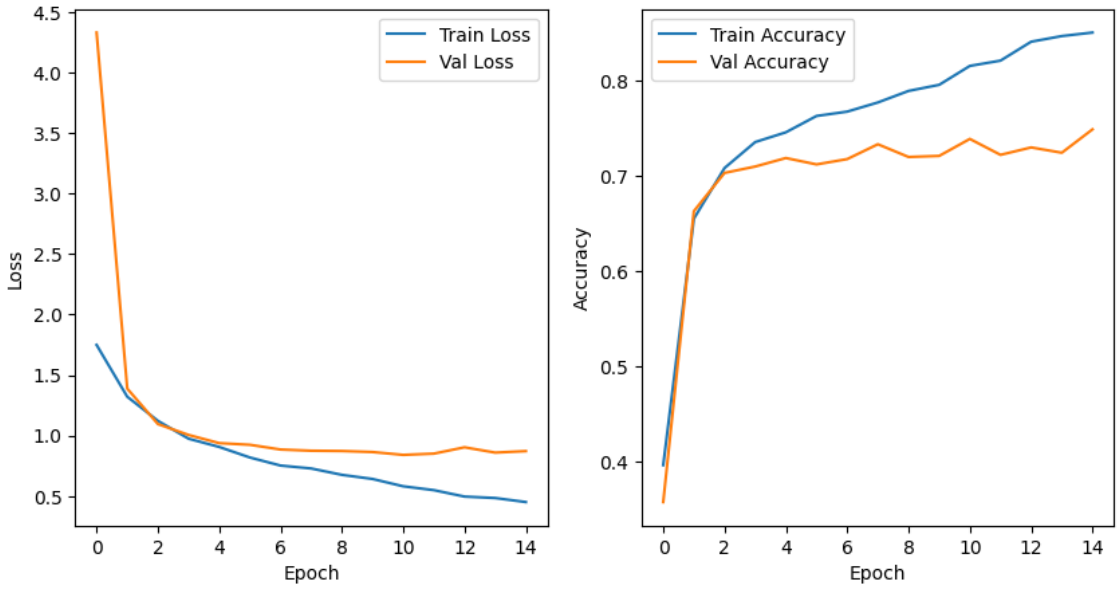


Figure 6: Classification task

7 Analysis

- We have analysed that with unpaired sketch, we have used WGANs GP model and the trend of losses was as expected. The discriminant loss is increasing and generator loss is decreasing.
- The images are also shown for both the phases, training and testing.
- The values of inception score is low which is due to that the inception models are trained on Imagenet dataset and we have different dataset and we had not fine tuned for our task, but after fine tuning, the scores were improved.
- While training the model for paired sketches, it can be seen that the model will converge for very high number of epochs as seen from the following :
- The complete results are also available with us for paired sketches

8 Conclusion

- Simply GANs are not controlled, but using CGANs we can generate a particular type of images.
- The GANs suffers problem of non convergence and mode collapse.
- Thus WGAN GP is used to address such issues upto an extent.
- More modifications can also be done like training model for more number of epochs to get more better results.

- The results with paired sketches seems to be more accurate
- Among both type of sketches, paired sketches will outperform the generation accuracy when trained for very large number of epochs.
- Other values of hyper parameters can also be employed for better modifications.
- Thus we can employ CGAN or WGAN as per our task.

9 Resources Used

- https://sh-tsang.medium.com/brief-review-wgan-gp-improved-training-of-wasserstein-gans-ae3e2acb25b
- https://arxiv.org/abs/1704.00028
- https://arxiv.org/abs/1701.07875
- Lecture Slides provided by the instructor

********** THE END **********