Machine Learning 2: 097209 - HW3
Rademacher Complexity & Generative Models

Daniel Vaanunu 206418642

Omri Lazover 318862323

Question 1- Rademacher Complexity

1. **Calculations for the minimum & maximum values of Rademacher Complexity for a {-1, +1} loss function:**

- The minimum value is obtained when $|H| = 1$:

$$R(F \circ S) = E_\sigma \left[\sup_{f \in F} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * f(z_i) \right]$$

We can remove the 'sup' since there is only 1 hypothesis:

$$R(F \circ S) = E_\sigma \left[\frac{1}{m} \sum_{i=1}^{m} \sigma_i * f(z_i) \right]$$

From the linearity of the expectancy:

$$R(F \circ S) = \frac{1}{m} \sum_{i=1}^{m} f(z_i) * E_\sigma[\sigma_i]$$

$since$: $E_\sigma[\sigma_i] = 0.5 * (+1) + 0.5 * (-1) = 0$, we get that the minimum value is:

$R(F \circ S) = 0$

- The maximum value is obtained when $|H| = 2^m$:

- 2^m is the size of all possible permutations of a binary vector of size m.
- Therefore, necessarily exists a hypothesis $h \in H$ that misclassifies all the data.

$$R(F \circ S) = E_\sigma \left[\sup_{f \in F} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * f(z_i) \right]$$

$$R(F \circ S) = P(\sigma = +1) * \frac{1}{m} \sum_{i=1}^{m} (+1) + P(\sigma = -1) * \frac{1}{m} \sum_{i=1}^{m} (+1)$$

From the linearity of the expectancy:

$$R(F \circ S) = 0.5 * \frac{1}{m} * m + 0.5 * \frac{1}{m} * m$$

$R(F \circ S) = 1$

2. Proof for $R(L \circ S) = \frac{1}{2} * R(H \circ S)$:

$$R(L \circ S) = E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * l(z_i, h) \right] = E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * I[h(x_i) \neq y_i] \right]$$

Using the identity from the tutorial for 0-1 loss: $L_s = \frac{1}{m} \sum_{i=1}^{m} \frac{1 - y_i * h(z_i)}{2}$

$$= E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * \frac{1 - y_i * h(z_i)}{2} \right] = E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \frac{\sigma_i}{2} - \frac{1}{m} \sum_{i=1}^{m} \frac{\sigma_i * y_i * h(z_i)}{2} \right]$$

Since $\sum_{i=1}^{m} \sigma_i$ is not related to the 'sup', we can take it out

$$= E_\sigma \left[\frac{1}{2m} \sum_{i=1}^{m} \sigma_i \right] - E_\sigma \left[\sup_{h \in H} \frac{1}{2m} \sum_{i=1}^{m} \sigma_i * y_i * h(z_i) \right]$$

From the linearity of the expectancy:

$$= \frac{1}{2m} \sum_{i=1}^{m} E_\sigma[\sigma_i] + \frac{1}{2m} * E_\sigma \left[\sup_{h \in H} \sum_{i=1}^{m} (-\sigma_i) * y_i * h(z_i) \right]$$

We know that $E_\sigma[\sigma_i] = 0.5 * (+1) + 0.5 * (-1) = 0 = E_\sigma[-\sigma_i]$:

$$= \frac{1}{2} * \frac{1}{m} * E_\sigma \left[\sup_{h \in H} \sum_{i=1}^{m} \sigma_i * y_i * h(z_i) \right] = \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * y_i * h(z_i) \right]$$

We'll show that for every $y_i \in \{-1, 1\}$, it holds that $\frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * y_i * h(z_i) \right] =$
$= \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * h(x_i) \right] = \frac{1}{2} * R(H \circ S)$:

$y_i = 1$:

$$\frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * y_i * h(z_i) \right] = \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * 1 * h(z_i) \right] =$$
$$= \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * h(x_i) \right] = \frac{1}{2} * R(H \circ S)$$

$y_i = -1$:

$$\frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * y_i * h(z_i) \right] = \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * (-1) * h(z_i) \right] =$$

$$= \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} (-\sigma_i) * h(x_i) \right]$$

As we saw before: $E_\sigma[-\sigma_i] = E_\sigma[\sigma_i]$

$$\frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} (-\sigma_i) * h(x_i) \right] = \frac{1}{2} * E_\sigma \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * h(x_i) \right] = \frac{1}{2} * R(H \circ S)$$

Since we got that $\mathrm{R}(\mathrm{L} \circ \mathrm{S}) = \frac{1}{2} * R(H \circ S)$ for every $y_i \in \{-1, 1\}$, we can say that for every $\mathrm{y} = \{ y_1, \dots, y_n \}$:

$$\mathrm{R}(\mathrm{L} \circ \mathrm{S}) = \frac{1}{2} R(H \circ S)$$

3. **Proof for:** $R(\hat{F} \circ S) = R(F \circ S) + R(G \circ S)$ **:**

$$R(\hat{F} \circ S) = E_\sigma \left[\sup_{\hat{f} \in \hat{F}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * \hat{f}(z_i) \right]$$

According to the definition of $\hat{F} = \{f + g : f \in F, g \in G\}$:

$$R(\hat{F} \circ S) = E_\sigma \left[\sup_{\substack{f \in F \\ g \in G}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * (f(z_i) + g(z_i)) \right]$$

$$R(\hat{F} \circ S) = E_\sigma \left[\sup_{\substack{f \in F \\ g \in G}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * f(z_i) + \sigma_i * g(z_i)) \right]$$

$$R(\hat{F} \circ S) = E_\sigma \left[\sup_{f \in F} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * f(z_i) + \sup_{g \in G} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * g(z_i)) \right]$$

$$R(\hat{F} \circ S) = E_\sigma \left[\sup_{f \in F} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * f(z_i)] \right] + E_\sigma \left[\sup_{g \in G} \frac{1}{m} \sum_{i=1}^{m} \sigma_i * g(z_i) \right]$$

$$R(\hat{F} \circ S) = R(F \circ S) + R(G \circ S)$$

Question 2 - Generative models

Discussion:

Question 2 - Generative models: Discussion

Note: all the photos are taken from the Jupiter notebook.

GAN Architecture:

1. **Generator:**
   The Generator class defines a CNN in order to generate new images from a latent vector.
   It takes a latent vector as input (of size 100) and gradually upsamples it through a series of transposed convolutional layers. These layers increase the spatial dimensions while decreasing the depth until the desired output dimension is reached: 64 * 64 * 3. Each transposed convolutional layer is followed by batch normalization to stabilize training and ReLU activation functions to introduce non-linearity. The final layer applies Tanh to ensure the output values are within the range [-1, 1].

2. **Discriminator:**
   The Discriminator class implements another CNN responsible for discriminating between real and fake images. It receives images as input (of size 64 * 64 * 3) and processes them through several convolutional layers, reducing their spatial dimensions while increasing the depth. Each convolutional layer is followed by batch normalization to aid in learning and LeakyReLU activation functions to introduce non-linearity. The final layer produces a single output indicating the probability that the input image is real.

- The weights of both networks are initialized using a specific function to facilitate training.

Model Training

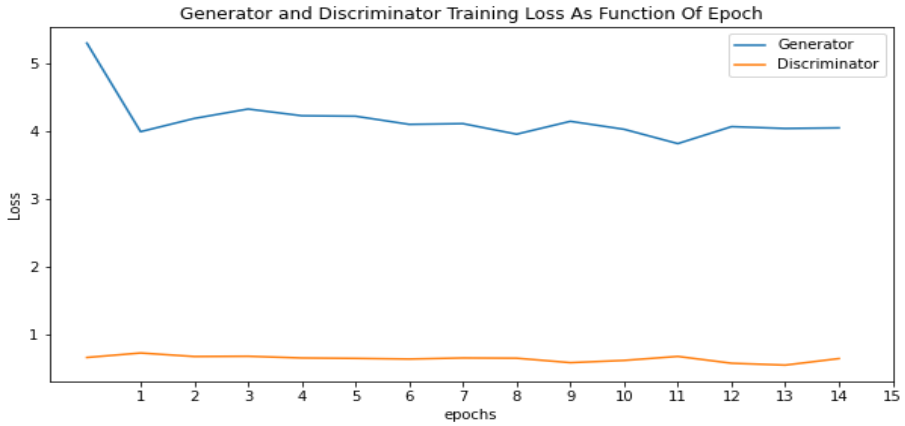
The training approach involves a cyclical process of updating the discriminator and generator models. The discriminator is trained to differentiate between real and fake images, while the generator aims to produce images that can effectively deceive the discriminator into classifying them as real. Both models utilize the Binary Cross-Entropy (BCE) Loss function to quantify their performance during training.

As for the hyperparameters, we used a learning rate of 0.0002 to both the generator and discriminator, in order to maintain stable training progress without excessively rapid adjustments. A batch size of 64 is employed, balancing computational efficiency with training stability.

The Adam optimizer is utilized for both models, with betas set to (0.5, 0.999) to ensure momentum is maintained and gradients are handled effectively for stable convergence. Adam adjusts learning rates based on the model's current state, which is particularly beneficial for the dynamic training dynamics of GANs

The latent space dimension is set to 100, offering enough complexity to encode diverse features for the generator while remaining computationally feasible.

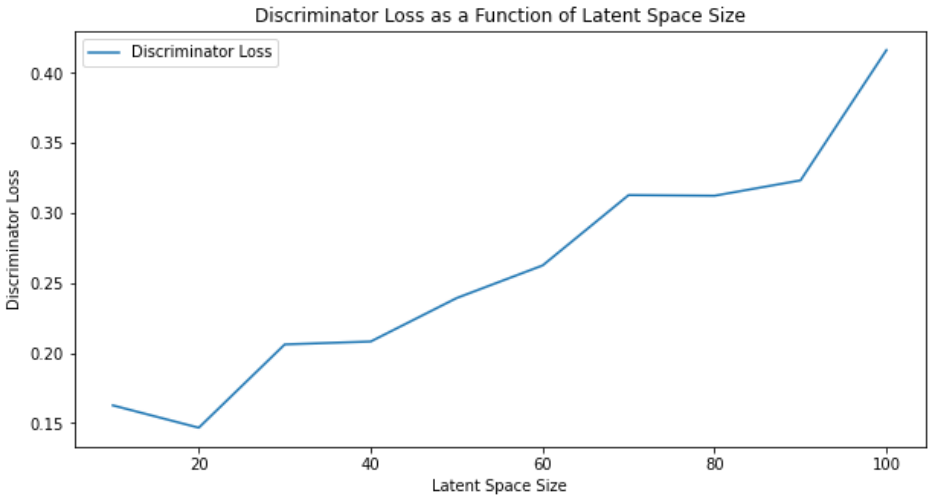
Here we can see the loss function of the generator and the discriminator as a function of epochs:



Summary & Conclusions

Throughout the 15 epochs of training, the discriminator consistently demonstrated strong performance, with marginal changes observed in its ability to differentiate between real and fake images. Despite variations in training epochs, the discriminator consistently assigned higher probabilities to real images (around 0.8) and lower probabilities to fake images (around 0.2). This stability in discriminator performance suggests that while the model may have learned to better distinguish between real and fake images, significant improvements were not evident in the results. Nevertheless, the generator's ability to produce realistic images improved gradually over epochs, as evidenced by the subtle enhancements in image quality and coherence (pictures in the next pages).

In addition, we saw that the loss of the discriminator increases (getting closer to 50%) as a function of the latent space – as the latent size getting bigger and closer to 100 as the models were training on, the discriminator's ability to recognize generated photos decreases:



As can be seen in our results (next pages):

- The first-epoches fake images generated:

```
Epoch [1/15] Loss_D: 0.6517, Loss_G: 5.3014, D(x): 0.8814, D(G(z)): 0.2441
```

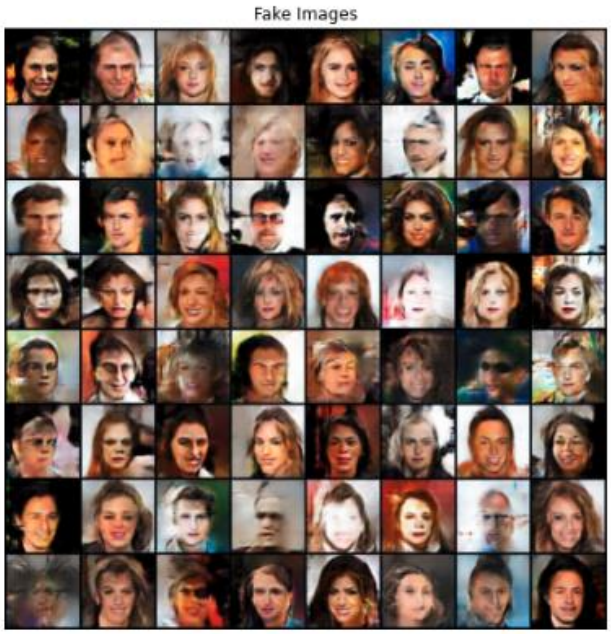
Fake Images

```
Epoch [2/15] Loss_D: 0.7179, Loss_G: 3.9908, D(x): 0.8597, D(G(z)): 0.2840
```

Fake Images

- In comparison to the last-epoches fake images generated:

```
Epoch [14/15] Loss_D: 0.5386, Loss_G: 4.0383, D(x): 0.8940, D(G(z)): 0.2083
```




```
Epoch [15/15] Loss_D: 0.6353, Loss_G: 4.0472, D(x): 0.8816, D(G(z)): 0.2378
```