

Additional Type of GANs

Youtube Reference : [WGANS: A stable alternative to traditional GANs || Wasserstein GAN](#)

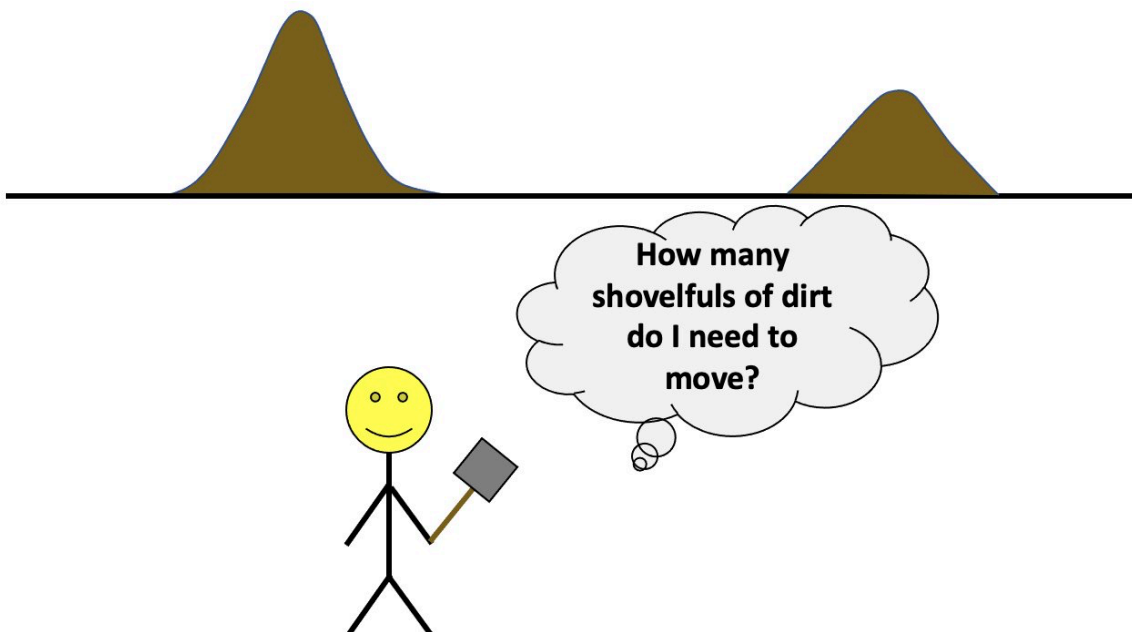
Wasserstein GANs:

Wasserstein distance between two distributions (Earth Mover's Distance)

The Earth Mover's Distance (EMD) is an intuitive way to measure how different two distributions are by imagining them as physical entities. Here's a clearer explanation using the metaphor of earth and holes:

- **Earth and Holes Metaphor:**

- Think of one distribution as a "mass of earth" (a pile of dirt) spread across a certain space.
- Think of the other distribution as "holes" in the same space that need to be filled. These holes represent where the mass of the second distribution is concentrated.



Earth mover's distance is an intuitive metric that denotes the minimum cost flow between two distributions (in other words, minimum work that needs to be done to turn one into the other)

But wasserstein loss for images are simply this:

```
wasserstein_loss = tf.reduce_mean(fake_image) - tf.reduce_mean(real_image)
```

Here the mean of an image represents the average pixel value of the image.

Why Wasserstein Loss is better than cross entropy loss:

Because it gives better gradients to improve the model upon than the cross entropy loss.

What about Exploding and Vanishing Gradient problems?

A gradient penalty term is introduced somewhat like a regularization term to limit the gradient update for prevention of these problems.

$$\text{Disc loss} = D_{\text{mean}}\left(\text{\right) - D_{\text{mean}}\left(\text{\right) + \lambda * \text{penalty}$$

$$\text{penalty} = [(\|\nabla_x D(x)\|_2 - 1)^2]$$

In Code

```
def discriminator_loss(real_output, fake_output, gradient_penalty, lambda_gp=10):
    wasserstein_loss = tf.reduce_mean(fake_output) - tf.reduce_mean(real_output) #
    Difference of means
    total_loss = wasserstein_loss + lambda_gp * gradient_penalty # Add gradient
    penalty
    return total_loss

def generator_loss(fake_output):
    return -tf.reduce_mean(fake_output)
```

wGANs Code : [wGANs.ipynb](#)

Rest of the things remain exactly the same.

The `gradient_penalty` function is used in the **Wasserstein GAN with Gradient Penalty (WGAN-GP)** to enforce the **Lipschitz continuity constraint** on the discriminator (critic). Mathematically, this means ensuring that the gradients of the critic's output with respect to its input have a norm of at most 1. This helps stabilize training and avoids mode collapse. Here's what is happening step by step:

1. Batch Size Consistency

python

[Copy code](#)

```
batch_size = min(real_images.shape[0], fake_images.shape[0])
real_images = real_images[:batch_size]
fake_images = fake_images[:batch_size]
```

- Ensures that `real_images` and `fake_images` have the same batch size, as required for interpolation.

2. Random Interpolation Between Real and Fake Images

python

[Copy code](#)

```
alpha = tf.random.uniform([batch_size, 1, 1, 1], 0.0, 1.0)
interpolated = alpha * real_images + (1 - alpha) * fake_images
```

- **Mathematical operation:**


$$\text{interpolated} = \alpha \cdot \text{real_images} + (1 - \alpha) \cdot \text{fake_images}$$

Here:

- α is a random scalar sampled uniformly between 0 and 1, for each image in the batch.
- The interpolation creates a new set of images that are a mix of real and fake images. These interpolated images lie on the line segments between real and fake samples in the input space.

3. Gradient Computation

python

 Copy code

```
with tf.GradientTape() as tape:
    tape.watch(interpolated)
    predictions = discriminator(interpolated, training=True)
    gradients = tape.gradient(predictions, [interpolated])[0]
```


- The **critic's predictions** ($D(\text{interpolated})$) are computed for the interpolated images.
- The **gradient** of the critic's output with respect to the interpolated images is calculated:

$$\nabla_{\text{interpolated}} D(\text{interpolated})$$

This represents how sensitive the critic's output is to small changes in the interpolated images.

4. Gradient Norm

python

 Copy code

```
gradients_norm = tf.sqrt(tf.reduce_sum(tf.square(gradients), axis=[1, 2, 3]))
```


- The **norm** of the gradient is calculated for each image in the batch:

$$\|\nabla_{\text{interpolated}} D(\text{interpolated})\| = \sqrt{\sum_{i,j,k} \left(\frac{\partial D}{\partial x_{i,j,k}} \right)^2}$$

where i, j, k index the spatial dimensions (e.g., height, width, channels).

5. Penalty Calculation

python

 Copy code

```
penalty = tf.reduce_mean((gradients_norm - 1.0) ** 2)
```

- The penalty is computed by measuring how much the gradient norm deviates from 1:

$$\text{penalty} = \mathbb{E} [(\|\nabla_{\text{interpolated}} D(\text{interpolated})\| - 1)^2]$$

- If the gradient norm is exactly 1, the penalty is 0.
- If the norm deviates, the penalty increases quadratically.

Purpose and Mathematical Rationale

1. Enforcing Lipschitz Continuity:

- In WGAN-GP, the critic is required to satisfy the Lipschitz constraint: $\|\nabla_x D(x)\| \leq 1$. This ensures the Wasserstein distance is a valid distance metric.

2. Encouraging Stability:

- The penalty regularizes the critic, preventing gradients from becoming too large or too small, which stabilizes training.

3. Mathematical Loss:

- The gradient penalty is added to the critic's loss:

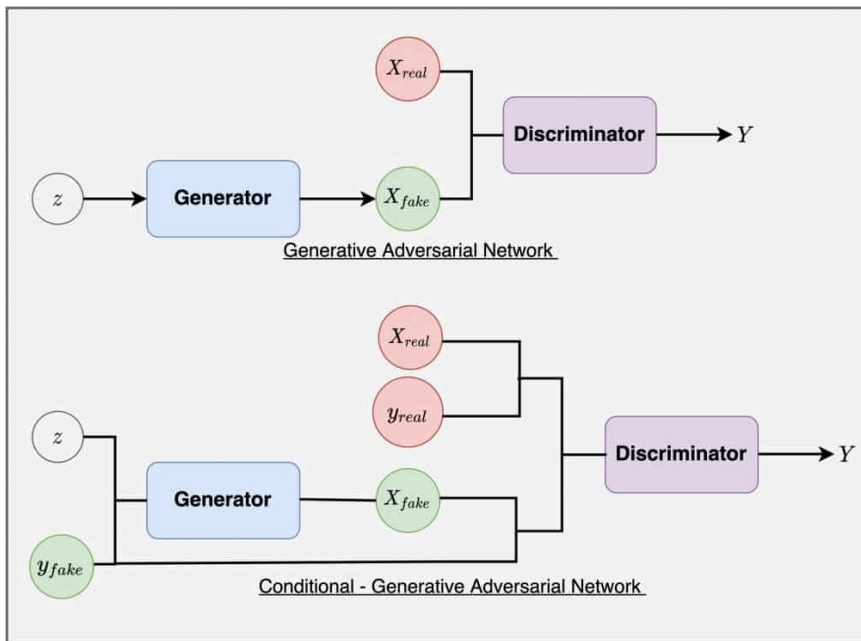
$$L_{\text{critic}} = \mathbb{E}[D(\text{fake})] - \mathbb{E}[D(\text{real})] + \lambda \cdot \text{penalty}$$

where λ is a weighting factor controlling the strength of the penalty.

Conditional GANs:

Why?

To control the output of GANs. We don't want random images, we want to give some conditions on which the model gives output.



We give labels in y_{fake} such as 0,1,2,3 as a condition to generate and discriminate.

cGANs Code : [cGANs.ipynb](#)

Text-to-Image GANs are similar to cGANs as in the labels we are giving are just embeddings as a condition for the models, so we can also give generator and discriminator the embeddings of text and its corresponding image to generate this Text-to-Image GANs.

Stacked GANs

Text-to-Image GANs -

A white bird -> Bad resolution Image white bird -> SRGANs -> HD Images

Cycles GANs

Face reconstruction

