

Laboratorio 6

Elisa Samayoa y Julio Avila

Preparación de los datos

- Carga correcta del conjunto de datos:
- Preprocesamiento adecuado:

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(  
    "/kaggle/input/celeba-dataset/img_align_celeba/img_align_celeba/",  
    label_mode = None,  
    image_size = (IM_SHAPE[0], IM_SHAPE[1]),  
    batch_size = BATCH_SIZE  
)
```

Found 202599 files belonging to 1 classes.

```
def preprocess(image):  
    return tf.cast(image, dtype = tf.float32)/127.5 - 1.0
```

Implementación de la GAN

```
BATCH_SIZE = 128  
IM_SHAPE = (64, 64, 3)  
LEARNING_RATE = 0.0000001  
LATENT_DIM = 100  
EPOCHS = 30
```

```
generator = tf.keras.Sequential([  
    Input(shape = (LATENT_DIM,)),  
    Dense(4*4*LATENT_DIM,),  
    Reshape((4, 4, LATENT_DIM)),  
  
    Conv2DTranspose(512, kernel_size = 4, strides = 2, padding = 'same'),  
    BatchNormalization(),  
    LeakyReLU(0.3),  
  
    Conv2DTranspose(256, kernel_size = 4, strides = 2, padding = 'same'),  
    BatchNormalization(),  
    LeakyReLU(0.3),  
  
    Conv2DTranspose(128, kernel_size = 4, strides = 2, padding = 'same'),  
    BatchNormalization(),  
    LeakyReLU(0.3),  
  
    Conv2DTranspose(3, kernel_size = 4, strides = 2, activation = tf.keras.activations.tanh, padding = 'same'),  
, name = 'generator')
```

```

discriminator = tf.keras.Sequential([
    Input(shape = (IM_SHAPE[0],IM_SHAPE[1], IM_SHAPE[2])),

    Conv2D(64, kernel_size = 4, strides = 2, padding = 'same'),
    LeakyReLU(0.3),

    Conv2D(128, kernel_size = 4, strides = 2, padding = 'same'),
    BatchNormalization(),
    LeakyReLU(0.3),

    Conv2D(256, kernel_size = 4, strides = 2, padding = 'same'),
    BatchNormalization(),
    LeakyReLU(0.3),

    Conv2D(1, kernel_size = 4, strides = 2, padding = 'same'),

    Flatten(),
    Dense(1, activation = 'sigmoid')

], name = 'discriminator')

```

```

class GAN(tf.keras.Model):
    def __init__(self, generator, discriminator):
        super(GAN, self).__init__()

        self.generator = generator
        self.discriminator = discriminator

    def compile(self, d_optimizer, g_optimizer, loss_fun):
        super(GAN, self).compile()
        self.d_optimizer = d_optimizer
        self.g_optimizer = g_optimizer
        self.loss = loss_fun
        self.d_loss_mat = tf.keras.metrics.Mean(name = 'd_loss')
        self.g_loss_mat = tf.keras.metrics.Mean(name = 'g_loss')

```

Entrenamiento de la GAN

```

train_data = (
    dataset
    .map(preprocess)
    .unbatch()
    .shuffle(buffer_size = 1024, reshuffle_each_iteration = True)
    .batch(BATCH_SIZE, drop_remainder = True)
    .prefetch(tf.data.AUTOTUNE)
)

```

```

def train_step(self, real_img):
    batch_size = tf.shape(real_img)[0]
    random_noise = tf.random.normal(shape = (batch_size, LATENT_DIM))
    fake_imgs = self.generator(random_noise)
    real_labels = tf.ones((batch_size, 1)) + 0.25*tf.random.uniform((batch_size, 1), minval = -1, maxval = 1)
    fake_labels = tf.zeros((batch_size, 1)) + 0.25*tf.random.uniform((batch_size, 1),)

    with tf.GradientTape() as recorder:
        real_pred = self.discriminator(real_img)
        d_loss_real = self.loss(real_labels, real_pred)

        fake_pred = self.discriminator(fake_imgs)
        d_loss_fake = self.loss(fake_labels, fake_pred)

        d_loss = d_loss_real + d_loss_fake
    partial_derivatives = recorder.gradient(d_loss, self.discriminator.trainable_weights)
    self.d_optimizer.apply_gradients(zip(partial_derivatives, self.discriminator.trainable_weights))

    random_noise = tf.random.normal(shape = (batch_size, LATENT_DIM))
    flipped_fake_labels = tf.ones((batch_size, 1))

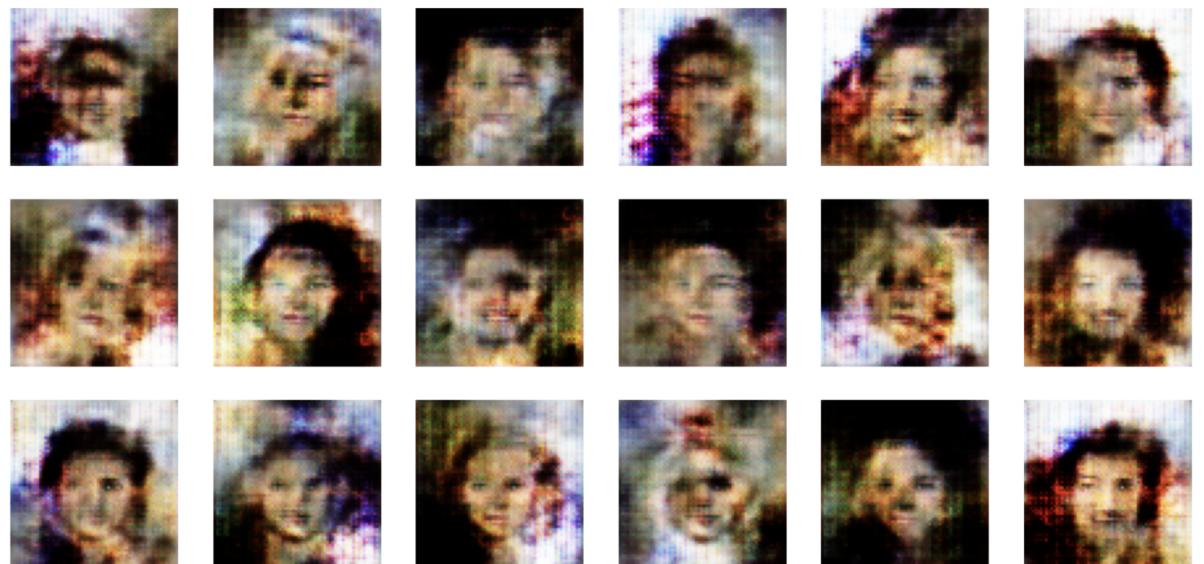
```

```

class ShowGeneratedImages(tf.keras.callbacks.Callback):
    def __init__(self, latent_dim = 100):
        self.latent_dim = latent_dim

    def on_epoch_end(self, epoch, logs = None):
        n = 6
        k = 0
        out = self.model.generator(tf.random.normal(shape = (64, self.latent_dim)))
        plt.figure(figsize = (16, 16))
        for i in range(n):
            for j in range(n):
                ax = plt.subplot(n, n, k+1)
                plt.imshow((out[k]+1)/2)
                plt.axis('off')
                k+=1
        plt.savefig("gen_images_epoch_{}.png".format(epoch))

```



1. ¿Qué conceptos de la teoría encontraron más desafiantes y por qué?

Aunque no era parte de la teoría, una de las partes más desafiantes del laboratorio fue cargar los datos, ya que eran demasiadas imágenes y con la memoria que contábamos no fue suficiente para trabajarla, por lo que se tuvo que recurrir a los recursos de Kaggle y trabajar allí directamente el notebook.

Además de ello, fue complicado entender cómo funcionaban y afectaban los hiperparámetros a los resultados, como lo fue la tasa de aprendizaje, porque nunca pudimos encontrar el número que mejor se adaptara para los resultados que nosotros esperamos encontrar. Tuvimos que ir cambiando entre tasas cada vez más pequeñas, con diferentes cantidades de épocas, pero nuevamente, sin acabarnos la memoria que se proporcionaba en Kaggle.

Diseñar la arquitectura de la GAN fue la tercera complicación, ya que se tuvo problemas con las capas y con las dimensiones de las imágenes. Al entrenarla, tuvimos muchos errores porque no se adaptaba el shape de las imágenes del generador, con las del discriminador, por lo que se tuvo que iniciar desde 0 varias veces para modificar eso y hacer que funcionara. Otro de los errores fue relacionado con el gradiente, por no haber usado las funciones de activación correctas, pero finalmente, nos pasamos a las ReLu que esas funcionaron mejor.

2. ¿Cómo les ayudó el laboratorio a consolidar o entender mejor estos conceptos?

Por sí solo era difícil entender todos los conceptos, así que gran parte de la ayuda fue ver lo que otros ya había logrado en Kaggle y los ejemplos de internet o como el que vimos en clase.

Pero cuando ya se logró comprender mejor lo que se había hecho, al tener que hacer gran parte de la GAN de acuerdo a lo que mejor se adaptara. Al intentar tantas veces para ver qué hiperparámetros quedaban mejor y no daban error con lo demás, fuimos viendo que era mejor bajarle a la tasa de entrenamiento, por ejemplo, y nunca superar las épocas de 50 porque nos daba error por memoria.

Este laboratorio requería de mucha “prueba y error” y aunque los resultados finales no quedaron como se imaginó, al ponerlo en práctica pudo terminar de entenderse la arquitectura empleada.

3. ¿Qué aplicaciones potenciales ven para las GANs en la industria o en la investigación?

- Podrían utilizarse en el ámbito de la medicina para mejorar la calidad de imágenes o la reconstrucción de las mismas, sobre todo, en casos que aún no cuentan con suficiente información, con enfermedades nuevas, para buscar patrones con otras, como en modelos diagnósticos.
- También podrían usarse en investigaciones geofísicas para la generación de modelos climáticos y simulaciones para predecir y estudiar el clima y el comportamiento de la Tierra.
- En cuanto a la industria, podrían ayudar a la generación y rediseño de productos, creando prototipos virtuales realistas hasta antes de la creación.

4. ¿Qué limitaciones o preocupaciones éticas pueden identificar en el uso de GANs?

- El robo de la propiedad intelectual podría ser uno de ellos. Últimamente, ha entrado en cuestionamiento si al trabajar las imágenes con una IA, los créditos pertenecen a la IA o a la persona de la que basaron el arte. La utilización de las GAN podría desencadenar el mismo problema, porque al tomar fotos o pinturas de inspiración para entrenar el modelo, si bien, las termina generando por “si solas”, no es desde 0.
- Se pueden falsificar identidades, si con el photoshop ya se había llegado lejos, con esta herramienta va a ser posible falsificar con mayor detalle, sin que se puedan dar cuenta. Por lo que va a empezar un cuestionamiento más recurrente de que si lo que se está viendo es real o algo formado a través de una GAN, así como sucedió con el caso del Papa hace unos meses, que todo mundo pensó que era real, hasta que tuvieron que salir a desmentirlo.

5. ¿Cómo se sienten con respecto a la implementación y entrenamiento de GANs después de la experiencia práctica?

Inicialmente, creíamos que era un proceso más fácil para lograr buenos resultados, pero al final, es algo que requiere no solo de una buena cantidad de imágenes para lograr buenos resultados, sino también, de adecuar correctamente los parámetros para observar algo razonable. No es algo que con poca información y tiempo pueda emplearse bien, pero una vez que ya lo hemos hecho, con los suficientes recursos, es algo que podríamos implementarlo fácilmente.