NNOVIEMBRE 2025

UABC. Facultad Ingeniería Mexicali

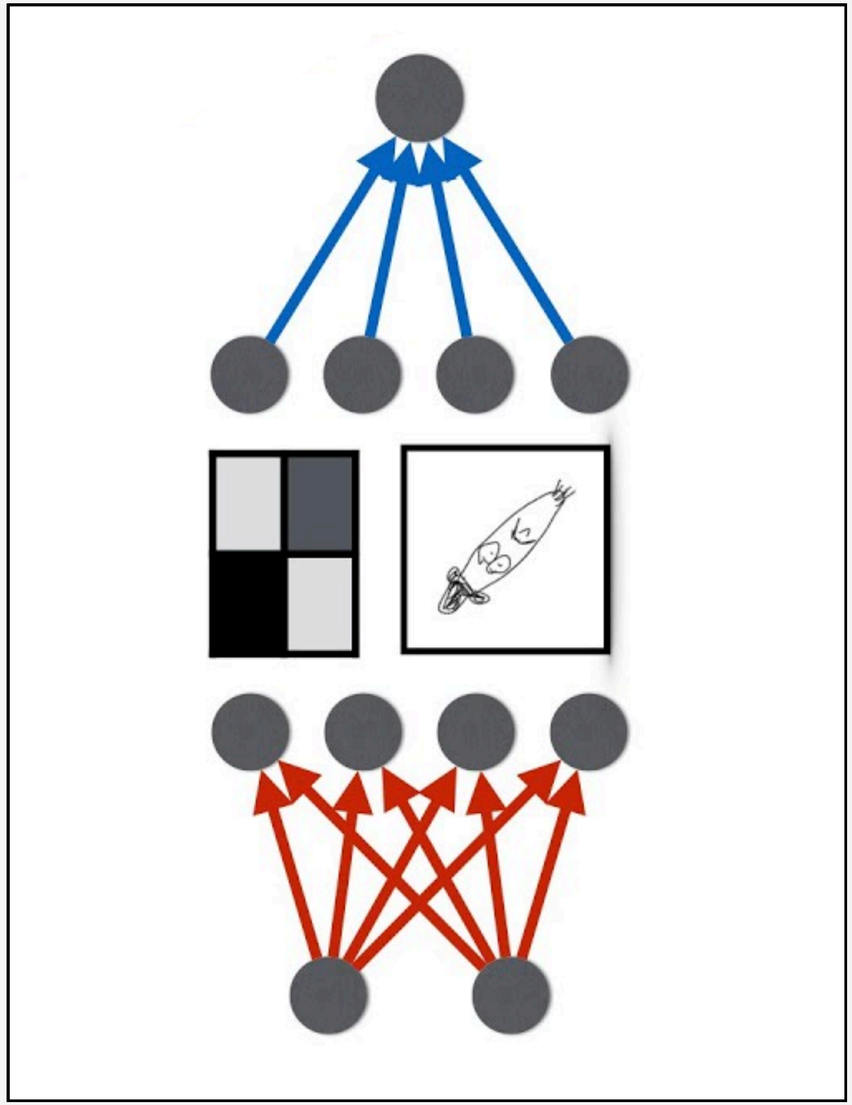
Redes Generativas Adversarias

Alejandro Torres Villalobos

Dinámica

Las Redes Generativas Adversarias (GAN) funcionan mediante la interacción competitiva entre dos redes: el generador y el discriminador.

Esta dinámica es esencial, porque el generador intenta transformar ruido aleatorio en observaciones que parezcan reales, mientras que el discriminador intenta distinguir entre imágenes reales y falsas. Ambos se entrenan simultáneamente: el discriminador se refuerza detectando falsificaciones y el generador mejora a medida que aprende a engañarlo.



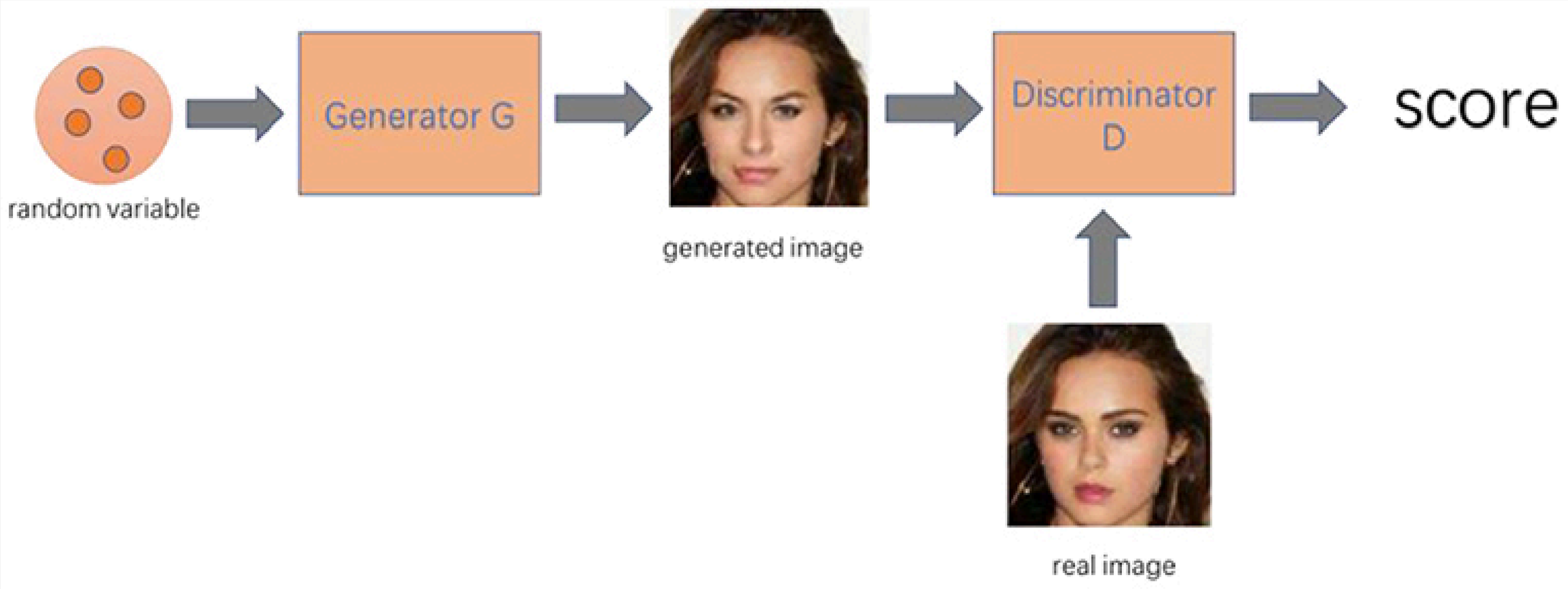
Dinámica

La dinámica es inestable por naturaleza:

- Si el discriminador domina, los gradientes para el generador se vuelven débiles y deja de aprender.
- Si el generador domina, produce un conjunto pequeño y repetitivo de imágenes muy parecidas, engañando fácilmente al discriminador.

Se aplican técnicas como añadir ruido a etiquetas, ajustar la tasa de aprendizaje o modificar la arquitectura para estabilizar esta interacción.

Dinámica


Generator G
Discriminator D
score
random variable
generated image
real image

Discriminante

El discriminador actúa como un clasificador que aprende una frontera entre lo real y lo generado.

Su tarea se entiende como un problema de aprendizaje supervisado:

- Imágenes reales se etiquetan como 1.
- Imágenes generadas se etiquetan como 0.

El discriminador evalúa cada entrada y produce una puntuación que indica si cree que la imagen es auténtica o no.

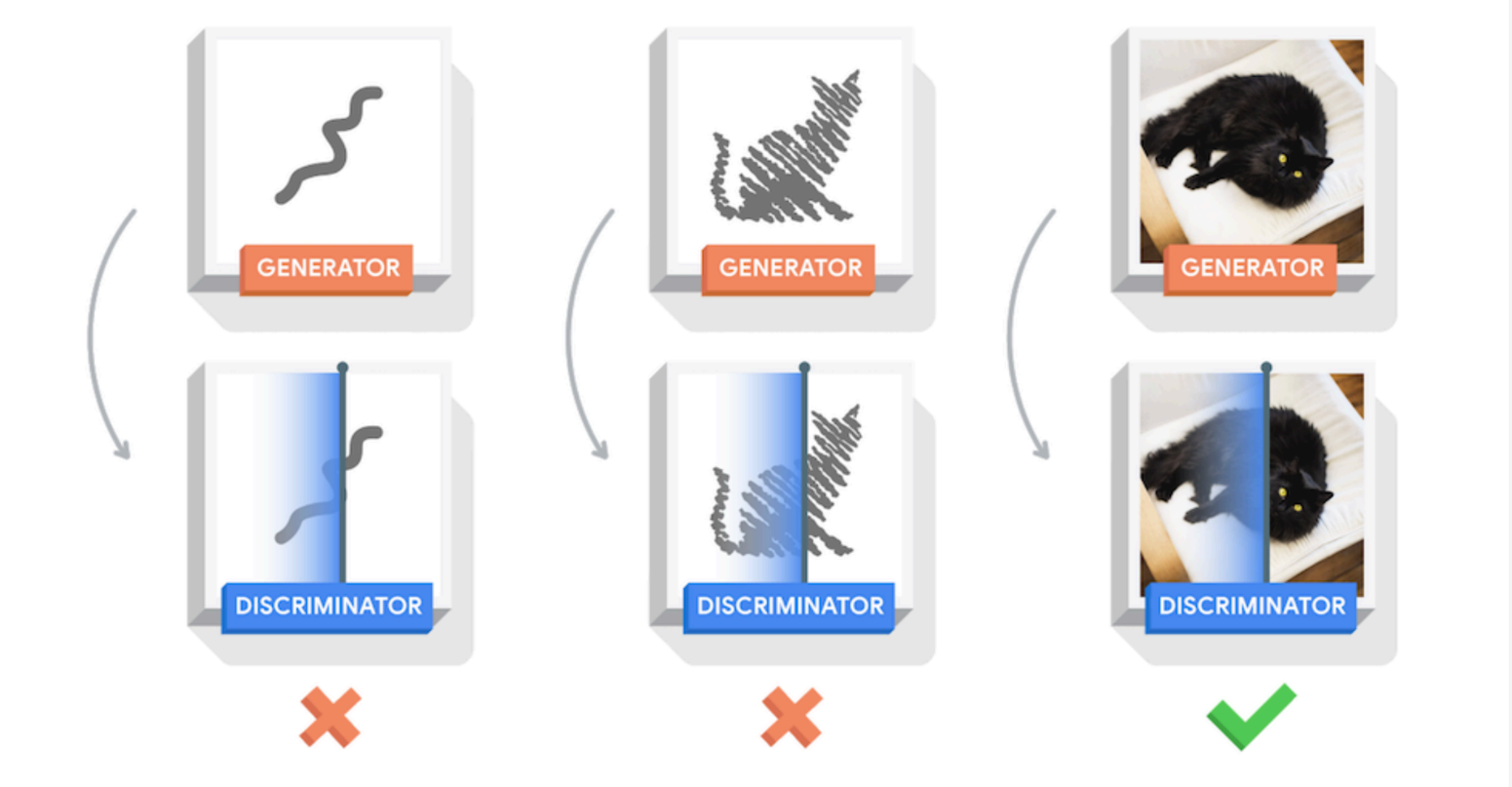
Discriminante

Si el discriminador es demasiado fuerte, la red completa deja de aprender; por ello, se aplican estrategias como:

- Reducir su tasa de aprendizaje.
- Disminuir filtros convolucionales.
- Añadir ruido a etiquetas o intercambiar etiquetas.

En variantes como WGAN, el discriminador pasa a llamarse crítico, y ya no predice probabilidades sino puntuaciones reales, lo que mejora la estabilidad del entrenamiento y la calidad de las imágenes generadas.

Discriminante

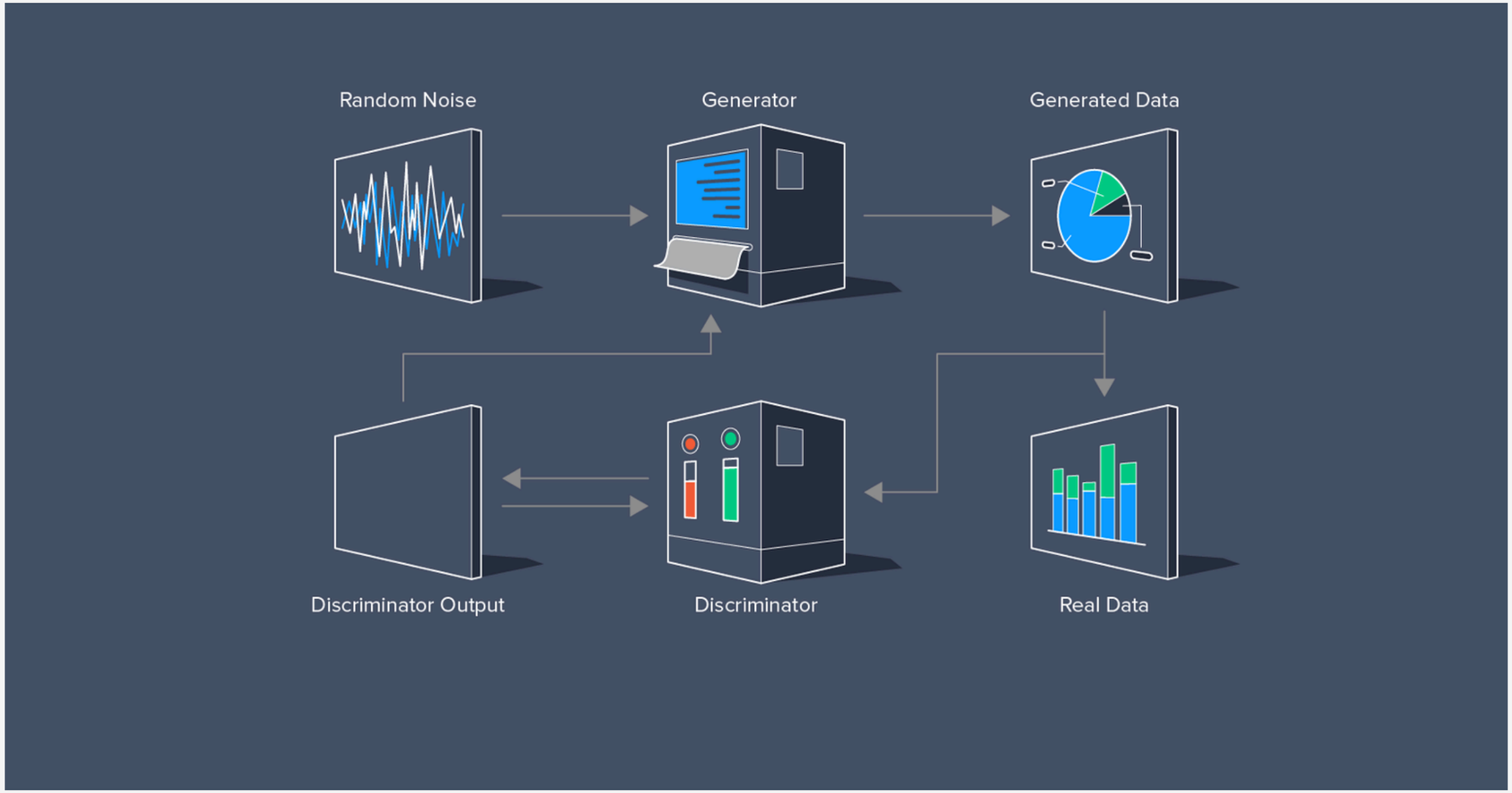


Generador

El generador transforma ruido aleatorio en nuevas observaciones que deberían parecer tomadas del conjunto de datos original.

Su entrenamiento depende completamente del discriminador: la función de pérdida del generador premia las imágenes que el discriminador clasifica como reales.

Generador


Random Noise
Generator
Generated Data
Discriminator Output
Discriminator
Real Data

Generador

En arquitecturas convolucionales profundas el generador:

- Recibe un vector de ruido.
- Usa capas densas y convolucionales traspuestas para ir aumentando la resolución.
- Emplea activaciones ReLU y normalización de lotes para estabilizar el aprendizaje.
- Finaliza con una capa con activación tanh, que produce imágenes en un rango de –1 a 1.

Al inicio, el generador produce imágenes sin sentido, pero a través de retroalimentación constante del discriminador, va aprendiendo características relevantes del conjunto de datos.

Función de costo

En GAN clásicas (entropía cruzada binaria):

- El discriminador aprende comparando imágenes reales (etiqueta 1) y falsas (etiqueta 0).
- El generador aprende intentando que el discriminador etiquete sus imágenes falsas como 1.

Esto genera inestabilidad porque el discriminador puede volverse demasiado preciso y dejar sin gradientes útiles al generador.

Función de costo

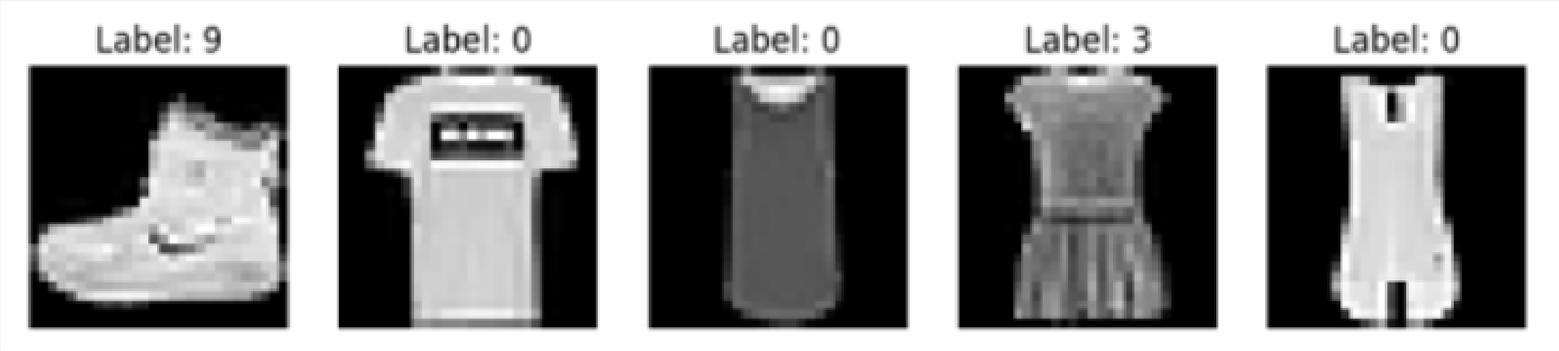
En WGAN y WGAN-GP (pérdida de Wasserstein):

- Se sustituyen las etiquetas 1 y 0 por 1 y –1.
- Se elimina la sigmoide; ahora el discriminador da puntuaciones reales.
- El crítico intenta maximizar la diferencia entre puntuaciones de imágenes reales y falsas.
- El generador intenta producir imágenes con puntuaciones altas.
- Esta función de costo da gradientes estables, evita el colapso y permite entrenar el discriminador hasta la convergencia sin perjudicar al generador.

Aplicación

A continuación se describe cómo se implementa una GAN para generar imágenes de prendas de vestir utilizando el conjunto de datos FashionMNIST.

El conjunto de datos comprende 10 categorías: camiseta/top, pantalón, jersey/suéter, vestido, abrigo, sandalia, camisa, zapatilla deportiva, bolso y bota de tobillo.



Aplicación

Para el procesamiento de los datos, se realiza una normalización de las imágenes, dividiéndolas entre 127.5 y restando 1 para llevar sus valores de píxel al rango de –1 a 1. Esto se hace con el fin de evitar problemas de escala durante el entrenamiento y para asegurar que la red pueda trabajar adecuadamente con valores negativos en las entradas.

Aplicación

El generador debe tomar valores aleatorios como entrada y producir imágenes 28x28x1. Para ello se define una función de capas que posee la siguiente estructura:

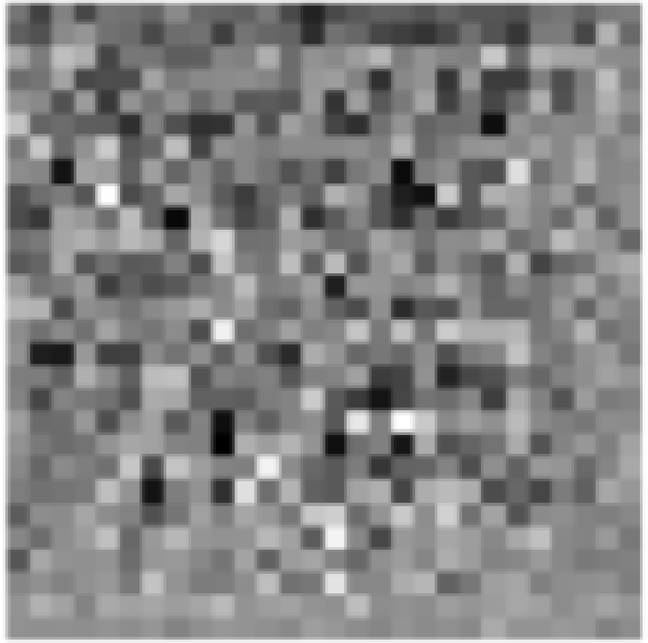
Se destacan dos bloques de capas convolucionales traspuestas (que incluyen activaciones ReLU fugaz 1 y normalización de lotes 2) para ir poco a poco hacia la dimensión 28x28. Después, una vez llegada a la dimensión, añadir dos bloques de capas convolucionales que mantengan la dimensión 28x28 pero pasando 128 filtros convolucionales para aprender más características. Finalmente, la capa de salida con un solo filtro y activación tanh.

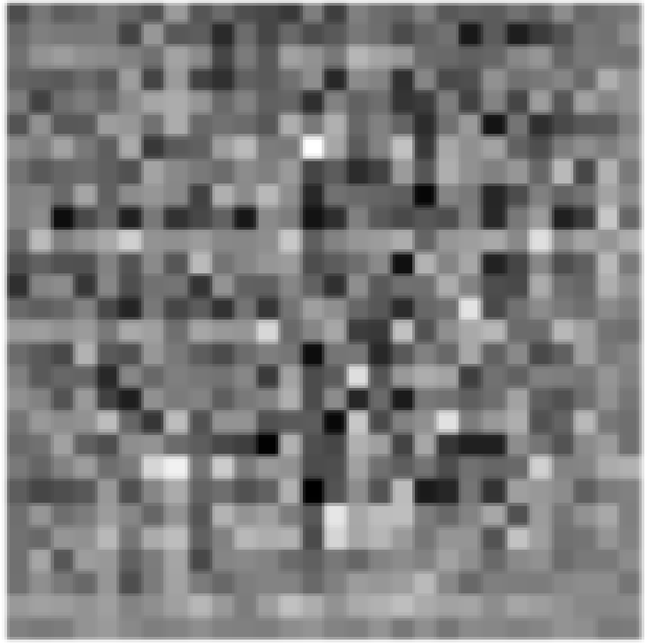
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 6272)	809,088
leaky_re_lu (LeakyReLU)	(None, 6272)	0
reshape (Reshape)	(None, 7, 7, 128)	0
conv2d_transpose (Conv2DTranspose)	(None, 14, 14, 128)	262,272
leaky_re_lu_1 (LeakyReLU)	(None, 14, 14, 128)	0
batch_normalization (BatchNormalization)	(None, 14, 14, 128)	512
conv2d_transpose_1 (Conv2DTranspose)	(None, 28, 28, 1)	2,049
leaky_re_lu_2 (LeakyReLU)	(None, 28, 28, 1)	0
batch_normalization_1 (BatchNormalization)	(None, 28, 28, 1)	4
conv2d (Conv2D)	(None, 28, 28, 128)	2,176
leaky_re_lu_3 (LeakyReLU)	(None, 28, 28, 128)	0
conv2d_1 (Conv2D)	(None, 28, 28, 128)	262,272
leaky_re_lu_4 (LeakyReLU)	(None, 28, 28, 128)	0
conv2d_2 (Conv2D)	(None, 28, 28, 1)	2,049

Total params: 1,340,422 (5.11 MB)
Trainable params: 1,340,164 (5.11 MB)
Non-trainable params: 258 (1.01 KB)

Aplicación

Tras construirlo, se realizan pruebas para ver qué salidas produce y se obtiene lo siguiente:





Aplicación

Claramente, por la figura anterior, el generador necesita la retroalimentación del discriminador para comenzar a producir algo con sentido. El discriminador, por su parte, tomará como entrada imágenes del generador y del conjunto de datos y tendrá que producir un número, 1 o 0 que indicará si la imagen es falsa o verdadera.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 24, 24, 32)	832
leaky_re_lu_5 (LeakyReLU)	(None, 24, 24, 32)	0
dropout (Dropout)	(None, 24, 24, 32)	0
conv2d_4 (Conv2D)	(None, 20, 20, 64)	51,264
leaky_re_lu_6 (LeakyReLU)	(None, 20, 20, 64)	0
dropout_1 (Dropout)	(None, 20, 20, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 128)	204,928
leaky_re_lu_7 (LeakyReLU)	(None, 16, 16, 128)	0
dropout_2 (Dropout)	(None, 16, 16, 128)	0
conv2d_6 (Conv2D)	(None, 12, 12, 256)	819,456
leaky_re_lu_8 (LeakyReLU)	(None, 12, 12, 256)	0
dropout_3 (Dropout)	(None, 12, 12, 256)	0
flatten (Flatten)	(None, 36864)	0
dropout_4 (Dropout)	(None, 36864)	0
dense_1 (Dense)	(None, 1)	36,865

Total params: 1,113,345 (4.25 MB)
Trainable params: 1,113,345 (4.25 MB)
Non-trainable params: 0 (0.00 B)

Aplicación

Consta de 4 bloques convolucionales con número de filtros creciente para aprender características (incluye activación ReLU con fuga y capa de abandono 3) y un último bloque de aplanamiento con tanh para que discrimine si la imagen es real o falsa. Las dos imágenes anteriormente generadas se pasan al discriminador para que determine qué cree que son, obteniendo un resultado de 0.49. Es decir, como no está entrenado y son las primeras imágenes que le llegan, no sabe si son verdaderas o falsas.

Aplicación

Se crea un modelo subclase para llevar a cabo un entrenamiento personalizado, dado que se busca ajustar dos redes de manera simultánea. Se instancia el modelo definiendo al generador y al discriminador como atributos principales. Posteriormente, se define el método de compilación utilizando el optimizador Adam, asignando una tasa de aprendizaje de 0.0001 para el generador y de 0.00001 para el discriminador. Esto se realiza con el propósito de evitar que el discriminador aprenda demasiado rápido y termine dominando el proceso de entrenamiento.

Aplicación

Discriminador:

1. Se genera un lote de imágenes falsas utilizando el generador
2. Se utiliza un lote de imágenes verdaderas del conjunto de datos.
3. Ambas imágenes, falsas y verdaderas, se pasan al discriminador.
4. Se asignan etiquetas: 1 para las imágenes verdaderas y 0 para las imágenes falsas.
5. Se calcula la pérdida del discriminador (d_loss) utilizando la entropía binaria cruzada entre las etiquetas predichas por el discriminador y las etiquetas verdaderas.
6. Se aplica retropropagación para actualizar los pesos del discriminador, minimizando esta pérdida.

Aplicación

Generador:

1. Se generan imágenes a partir de ruido aleatorio utilizando el generador, que se pasan al discriminador para obtener sus predicciones.
2. Se calcula la pérdida del generador (g_loss) comparando las etiquetas predichas por el discriminador con un vector de etiquetas donde todas son 1. Esto premia al generador cuando el discriminador etiqueta las imágenes falsas como 1 (es decir, como verdaderas).
3. Se aplica retropropagación para actualizar los pesos del generador, minimizando esta pérdida.