Trabajo Final

Fundamentos de aprendizaje automático

Ignacio Lembo Ferrari

Introducción a la Física Atómica Molecular y Óptica.

24- 06 - 2024

INTRODUCCIÓN

Objetivos del Proyecto

- Implementar una red Conditional Generative Adversarial Network (cGAN).
- Utilizar datasets: MNIST, CIFAR-10, CIFAR-100.
- Generar imágenes fake basadas en solicitudes del usuario.
- Explorar variantes de cGAN para mejorar calidad y precisión de las imágenes generadas.

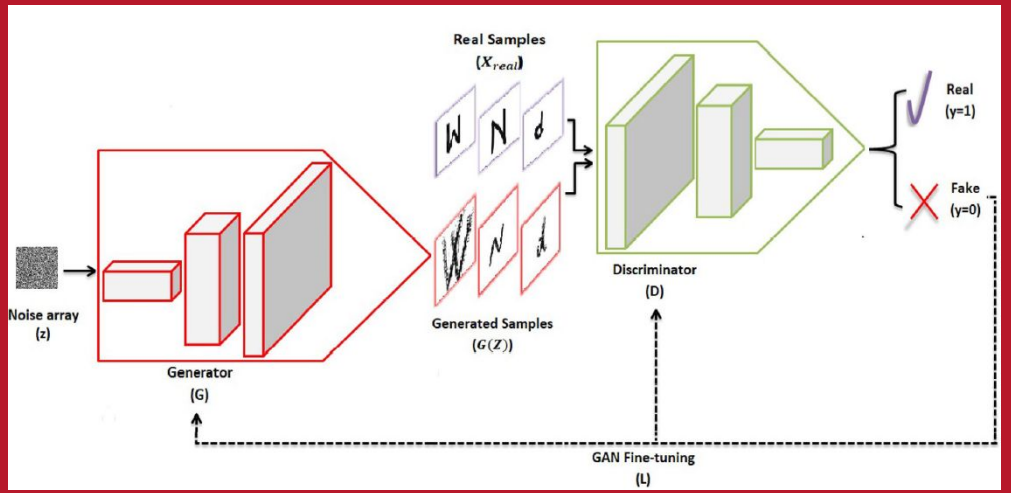
INTRODUCCIÓN

Teoría de cGANs

- **GANs: Dos redes neuronales (Generador y Discriminador) compitiendo.**

- cGANs: Introducen una condición adicional (etiqueta de clase) para generar imágenes específicas de una clase.

El generador crea imágenes falsas a partir de ruido aleatorio, mientras que el discriminador trata de distinguir entre las imágenes reales del conjunto de datos y las imágenes falsas generadas por el generador.



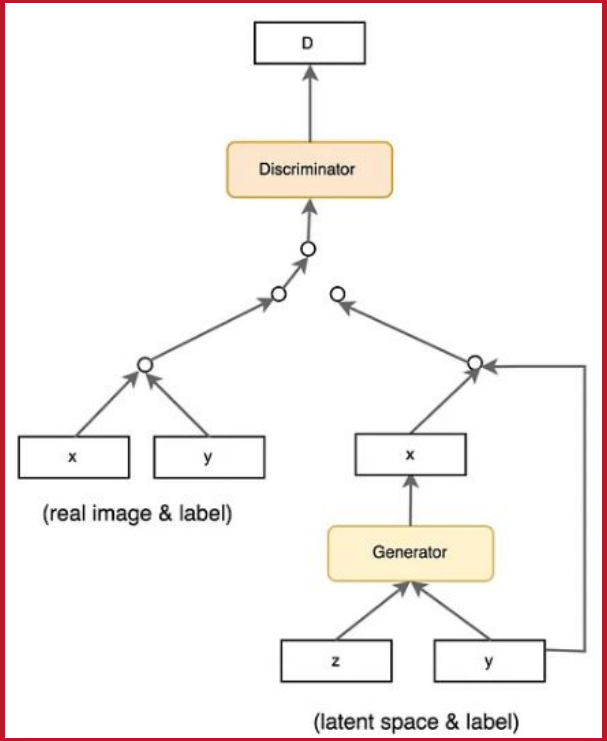
El objetivo del generador es engañar al discriminador para que no pueda diferenciar entre las imágenes reales y las generadas.

INTRODUCCIÓN

Teoría de cGANs

- GANs: Dos redes neuronales (Generador y Discriminador) compitiendo.
- **cGANs: Introducen una condición adicional (etiqueta de clase) para generar imágenes específicas de una clase.**

El generador crea imágenes falsas a partir de ruido aleatorio, mientras que el discriminador trata de distinguir entre las imágenes reales del conjunto de datos y las imágenes falsas generadas por el generador.



El objetivo del generador es engañar al discriminador para que no pueda diferenciar entre las imágenes reales y las generadas.

INTRODUCCIÓN

Teoría de cGANs

Las GANs se formulan como un juego de **minimax**, donde el discriminador intenta maximizar su recompensa V(D, G) y el generador intenta minimizar la recompensa del discriminador, o en otras palabras, maximizar su pérdida. Esto se puede describir matemáticamente con la siguiente fórmula:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)))]$$

El primer término del lado derecha incentiva al discriminador a clasificar correctamente las muestras reales.
El segundo término incentiva al generador a producir muestras que sean clasificadas como reales por el discriminador.

INTRODUCCIÓN

Teoría de cGANs

```python
bce_loss = tf.keras.losses.BinaryCrossentropy()

# Discriminator Loss
def discriminator_loss(real, fake):
    real_loss = bce_loss(tf.ones_like(real), real) # Calculo la loss para las imagenes reales
    fake_loss = bce_loss(tf.zeros_like(fake), fake) # Calculo la loss para las imagenes falsas
    total_loss = real_loss + fake_loss
    return total_loss

# Generator Loss
def generator_loss(preds):
    return bce_loss(tf.ones_like(preds), preds) # Calculo la loss para el generador
```

```python
generated_images = g_model([noise, real_labels], training = True) #  Genero imagenes falsas
pred_real = d_model([real_images, real_labels], training = True) # Obtengo las predicciones del discriminador para las imagenes reales
pred_fake = d_model([generated_images, real_labels], training = True) # Obtengo las predicciones del discriminador para las imagenes falsas

d_loss = discriminator_loss(pred_real, pred_fake) # Calculo la loss del discriminador
g_loss = generator_loss(pred_fake) # Calculo la loss del generador
```

Existen dos variantes para realizar este último paso

Análisis EDA y PCA

EDA:

- Distribución y estructura de datasets utilizando Pandas.
- MNIST: 785 features.
- CIFAR-10: 3073 features.

PCA: Análisis de varianza acumulada y reducción dimensional.

- MNIST: ~200 componentes principales.
- CIFAR-10: ~400 componentes principales.
- Gráficos de dispersión muestran complejidad en la distribución de clases.

Modelo 1: MNIST

Referencia: Modificación de la arquitectura en GeeksforGeeks.

Características:

- Generador y discriminador con 2 capas de deconvolución.
- Aproximadamente 1M de parámetros.
- Optimizador: Cambiado de Adam a RMSprop.

Resultados:

- Tiempo de cada época: ~58 segundos.
- Imágenes generadas muy realistas, difícil de distinguir de las reales.
- Problemas con archivos exportados en formato .keras.
- Discriminador alcanzó accuracy de ~0.9.
- Mucho ruido en la función loss.

Conclusión: Muy buen desempeño en generación de imágenes del MNIST.

Modelo 3: MNIST

Referencia: Versión reducida de un paper especializado en letras.

Características:

- Más capas de deconvolución, filtros variables, Dropout para prevenir overfitting y BatchNormalization para mantener la estabilidad del entrenamiento.
- Aproximadamente 5M de parámetros.
- Problemas de overfitting detectados.

Resultados:

- Tiempo de cada época: ~64 segundos.
- Imágenes generadas son de alta calidad.
- Discriminador con loss casi nula y precisión >0.9.
- Problemas con archivos exportados en formato .keras.

Conclusión: Complejidad excesiva para MNIST, problemas de overfitting.

Modelo 3: MNIST

Referencia: Versión reducida de un paper especializado en letras.

Características:

- Más capas de deconvolución, filtros variables, Dropout y BatchNormalization.
- Aproximadamente 5M de parámetros.
- Problemas de overfitting detectados.

Resultados:

- Tiempo de cada época: ~64 segundos.
- Imágenes generadas son de alta calidad.
- Discriminador con loss casi nula y precisión >0.9.
- Problemas con archivos exportados en formato .keras.

Conclusión: Complejidad excesiva para MNIST, problemas de overfitting.

Modelo 3: CIFAR-100

Características:

- Adaptación del modelo 3 para CIFAR-100.
- Problemas de overfitting en el discriminador.

Resultados:

- Tiempo de cada época: ~55 segundos.
- Imágenes generadas muestran características básicas.
- Discriminador con precisión alta en entrenamiento y baja en testeo.

Conclusión: Desempeño limitado, imágenes generadas de muy baja resolución.

Modelo 4: CIFAR-100

Características:

- Variante del modelo 3 con una capa adicional de deconvolución y cambios en la última capa del generador.
- Aproximadamente 200k parámetros más que el modelo 3.

Resultados:

- Mejoras en la calidad de las imágenes generadas.

Conclusión: Incremento en la complejidad mejora la calidad de las imágenes.

Modelo 5: CIFAR-100

Características:

- Versión compleja con 4 capas de deconvolución en el generador y 5 capas de convolución en el discriminador.
- Aproximadamente 55M de parámetros.

Resultados:

- Tiempo de cada época: ~115 segundos.
- Imágenes generadas son más fieles a las esperadas, aunque no siempre representan objetos claros.

Conclusión: Alta complejidad mejora la generación, pero aún no alcanza alta fidelidad.

Modelo 5: CIFAR-10

Características:

- Varias versiones probadas con diferentes configuraciones de filtros, posición de la capa de Dropout, adición de momentum a la capa de Batchnormalization y uso de optimizador ADAM.

Resultados:

- Mejores resultados en CIFAR-10 que en CIFAR-100.
- Imágenes generadas identificables por humanos después de varias iteraciones.

Conclusión: Estrategias de optimización y configuraciones específicas mejoran el desempeño.

CONCLUSIONES

Arquitectura y Complejidad:

- Modelos simples son más efectivos para datasets como MNIST.
- Datasets complejos como CIFAR100 requieren modelos más sofisticados.
- Aumento excesivo en la complejidad puede causar sobreajuste.

Desempeño en Diferentes Datasets:

- MNIST: Imágenes generadas de alta calidad.
- CIFAR10: Mejoras con modelos complejos, pero aún requieren ajustes.
- CIFAR100: Imágenes básicas, pero poco realistas.

Evaluación de la Loss:

- Comportamiento ruidoso en datasets complejos.
- Evaluación a lo largo de las épocas crucial para ajustar el entrenamiento.

Eficiencia y Tiempo de Entrenamiento:

- Tiempo de época depende más del número de imágenes que de la complejidad del modelo.
- Modelos con más parámetros no aumentan demasiado el tiempo de entrenamiento.

MUCHAS GRACIAS

¿PREGUNTAS?