Universidad del Valle de Guatemala
Deep Learning y Sistemas Inteligentes
Sección 10

PROYECTO FINAL

Arquitectura Generative Adversarial Network (GAN) de perros y gatos

José Pablo Kiesling Lange - 21581
Herber Sebastián Silva Muñoz - 21764

Guatemala, 11 de noviembre del 2024

PROBLEMA

En la actualidad, la generación de imágenes a partir de GANs es un área de interés en el campo del aprendizaje profundo y la VC. Sin embargo, uno de los desafíos más significativos es la necesidad de grandes conjuntos de datos para entrenar modelos efectivos. En situaciones donde los datos son limitados, como en el caso de imágenes de perros y gatos disponibles para este proyecto, es esencial implementar técnicas que permitan aumentar la cantidad y diversidad de datos disponibles.

Asimismo, a diferencia de GANs famosas como *thispersondoesnotexist* la GAN generada posee una desventaja, y es que a diferencia de los humanos, tanto los perros como los gatos varían mucho entre razas, dando así una complejidad extra en la generación de estos animales.

Es por ello, que el problema se centra en cómo generar imágenes realistas de perros y gatos utilizando una cantidad limitada de datos. Se busca entrenar modelos capaces de aprender las características fundamentales de cada clase (perros y gatos) y generar nuevas imágenes que sean indistinguibles de las imágenes reales.

ANÁLISIS

Las GANs han demostrado ser efectivas en la generación de imágenes realistas. Como vimos en clase, una GAN consta de dos componentes principales: un generador y un discriminador. El generador intenta producir imágenes que el discriminador no pueda distinguir de las reales, mientras que el discriminador aprende a diferenciar entre imágenes reales y generadas.

Dado el conjunto de datos limitado (12500 imágenes para cada clase), se hizo una técnica que aprendimos en Data Science qué es es fundamental: data augmentation. Esto ayuda a incrementar la diversidad del conjunto de entrenamiento. Estas técnicas incluyen transformaciones como rotaciones, volteos horizontales, cambios de escala y otros que no alteran la esencia de las imágenes pero sí aportan variabilidad.

Además, al tratarse de dos clases distintas (perros y gatos), es conveniente entrenar dos modelos GAN separados para cada clase. Esto permite que cada modelo se especialice en las características particulares de su clase, mejorando la calidad de las imágenes generadas.

PROPUESTA DE SOLUCIÓN

La solución propuesta consiste en desarrollar dos modelos GAN independientes, uno para generar imágenes de perros y otro para imágenes de gatos. Como se mencionó anteriormente, se utilizarán técnicas de aumento de datos para ampliar el conjunto de entrenamiento y mejorar la capacidad de generalización de los modelos. El entrenamiento se realizará utilizando PyTorch y aprovechando la capacidad de procesamiento en GPU mediante CUDA para acelerar el proceso.

Los pasos clave incluyen:

- **Preparación de los datos**: Unificar las imágenes de perros y gatos en un solo directorio, distinguiéndose por el prefijo en el nombre del archivo (cat.x.jpg y dog.x.jpg donde x es el correlativo de la imagen 0 - 12499).
- **Aumento de datos**: Aplicar transformaciones aleatorias a las imágenes para aumentar la diversidad.
- **Desarrollo de modelos GAN**: Definir las arquitecturas del generador y discriminador para cada clase.
- **Entrenamiento de los modelos**: Entrenar cada GAN de forma independiente utilizando los conjuntos de datos aumentados.
- **Evaluación y visualización**: Generar y visualizar nuevas imágenes para evaluar la calidad de los modelos entrenados.

DESCRIPCIÓN DE LA SOLUCIÓN

Preparación de los datos
Se unificaron todas las imágenes en el directorio data_d, utilizando nombres de archivo que indican la clase (cat.x.jpg para gatos y dog.x.jpg para perros). Se creó un Dataset personalizado, FilteredImageDataset, que filtra y carga las imágenes basándose en el prefijo del nombre del archivo.

Aumento de datos
Se aplicaron las siguientes transformaciones a las imágenes:

- Redimensionamiento y recorte central: Ajustar las imágenes a un tamaño uniforme de 64x64 píxeles.
- Volteo horizontal aleatorio: Simular imágenes especulares para aumentar la diversidad.
- Rotación aleatoria: Rotar las imágenes dentro de un rango de ±15 grados.
- Normalización: Escalar los valores de los píxeles para que tengan una media de 0.5 y una desviación estándar de 0.5.

Estas transformaciones se implementaron utilizando la clase transforms.Compose de torchvision.

Desarrollo de los modelos GAN
Se definieron las clases Generator y Discriminator utilizando PyTorch. Las arquitecturas se basan en modelos convencionales para GANs:

- Generador: Utiliza capas de convolución transpuesta (nn.ConvTranspose2d), seguidas de capas de Batch Normalization y activaciones ReLU, terminando con una capa Tanh para generar imágenes de salida en el rango [-1, 1].
- Discriminador: Emplea capas de convolución (nn.Conv2d), seguidas de Batch Normalization y activaciones LeakyReLU, finalizando con una capa Sigmoid para producir una probabilidad de que la imagen sea real o falsa.

Se instanciaron modelos separados para perros y gatos, inicializando los parámetros y definiendo los optimizadores correspondientes (optim.Adam).

Entrenamiento de los modelos
El entrenamiento se llevó a cabo en dos fases separadas, una para cada clase:

1. Se iteró sobre el conjunto de datos d, utilizando el discriminador para aprender a distinguir entre imágenes reales (del conjunto de datos) y falsas (generadas por el generador).
2. El generador se entrenó para engañar al discriminador, mejorando iterativamente la calidad de las imágenes generadas.

Se utilizaron funciones de pérdida basadas en entropía cruzada binaria (nn.BCELoss) y se entrenaron los modelos durante 1000 épocas, guardando imágenes generadas y modelos en cada iteración.

HERRAMIENTAS APLICADAS

Aumento de datos (Data Augmentation)
Técnicas aplicadas para incrementar la cantidad y diversidad de datos de entrenamiento. En este proyecto, se usaron transformaciones de imágenes proporcionadas por torchvision.transforms, como:

- RandomHorizontalFlip(): Volteo horizontal aleatorio.
- RandomRotation(15): Rotación aleatoria dentro de un rango de ±15 grados.

Estas técnicas permiten que el modelo sea más robusto y generalice mejor a datos no vistos.

Dataset personalizado (FilteredImageDataset)
Se implementó un Dataset personalizado para filtrar y cargar las imágenes basándose en el prefijo del nombre del archivo. Esto es esencial dado que las imágenes de ambas clases están en el mismo directorio.

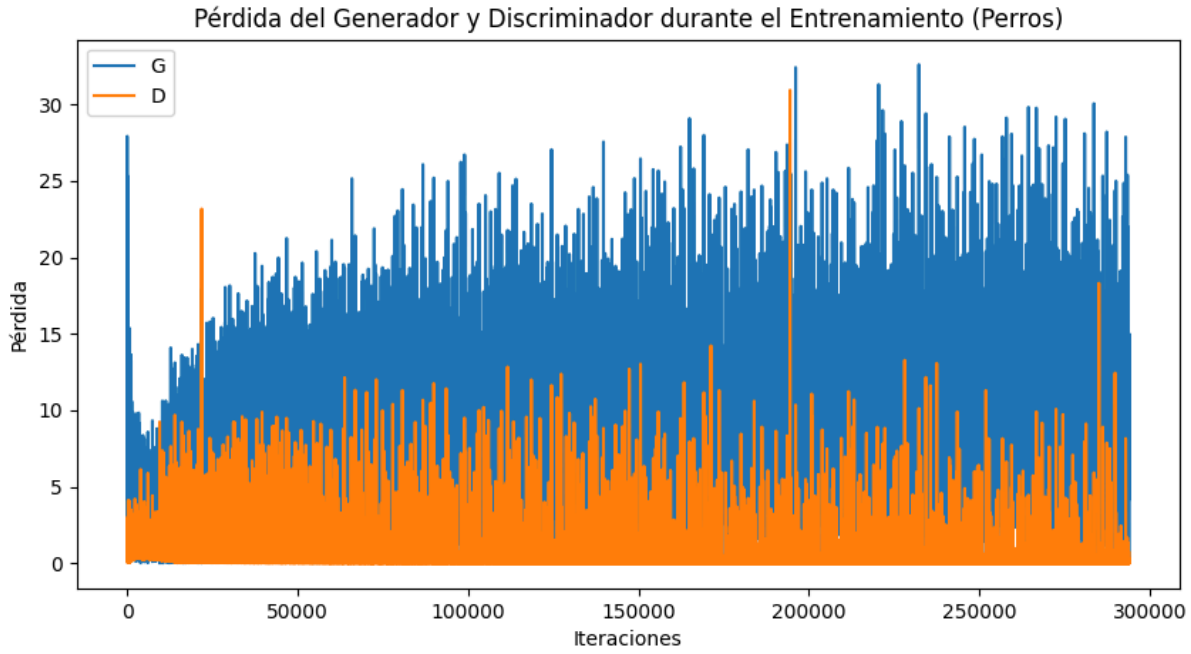
RESULTADOS

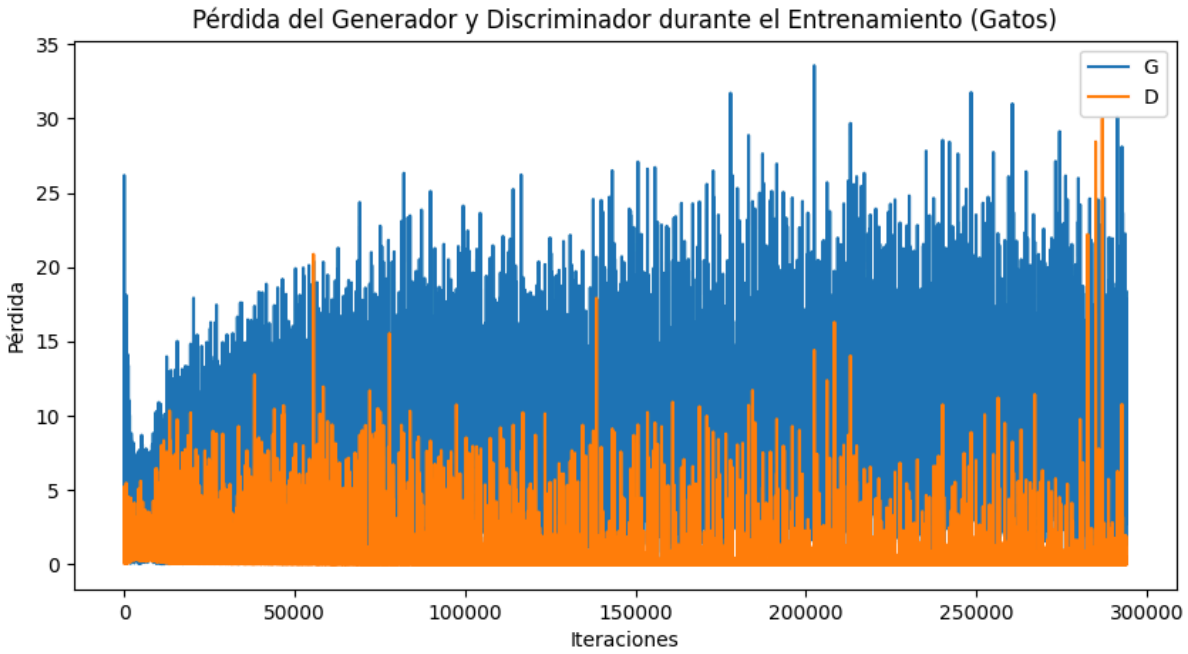
A nivel generativo se puede lograr apreciar ciertas características de cada clase, sin embargo, dado el inconveniente mencionado anteriormente respecto a diferencia entre razas de cada animal, los features de cada clase era más difícil para el generador de generar y que el discriminador los aceptara como válidos. Esto se puede apreciar en ambas generaciones:

Como se puede apreciar, en el caso del perro se logra distinguir características como el ojo, la nariz, las orejas y su forma. En el caso del gato, se puede distinguir la forma y las orejas, siendo diferentes entre clases.

Ahora en cuanto a resultados de los modelos, se obtuvo lo siguiente:

Pérdida





Como se puede apreciar en ambos casos, a excepción de casos atípicos, el discriminador se volvía más “crítico” ya que su pérdida iba disminuyendo, por consiguiente, era más difícil para el generador lograr burlar al discriminador.

Inception Score

Esta métrica evalúa dos aspectos clave de las imágenes generadas:

- **Calidad**: Las imágenes generadas deben ser fácilmente clasificables en una clase específica.
- **Diversidad**: Las imágenes deben ser diversas y cubrir varias clases.

Dado que se está generando imágenes de una sola clase (perros o gatos), el IS puede ser adaptado para evaluar la calidad y diversidad dentro de esa clase.

Inception Score para perros: 1.1104 ± 0.0478

Inception Score para gatos: 1.0981 ± 0.0577

Los resultados obtenidos indican que los modelos GAN entrenados no están generando imágenes de perros y gatos con la calidad y diversidad deseadas. Estos valores están muy cerca de 1, lo que sugiere que el modelo Inception no está clasificando las imágenes generadas en clases específicas con alta confianza. Un IS cercano a 1 generalmente indica que las imágenes generadas son de baja calidad o carecen de diversidad, ya que el modelo predice distribuciones de clase casi uniformes.

Fréchet Inception Distance

El Fréchet Inception Distance mide la distancia entre las distribuciones de características extraídas de imágenes reales y generadas utilizando un modelo pre-entrenado (Inception v3). Es más robusto que el IS y puede detectar diferencias sutiles en calidad y diversidad.

Esta métrica es de utilidad ya que el FID compara las estadísticas de las imágenes generadas con las reales, proporcionando una medida más objetiva. Además, puede detectar problemas como la falta de diversidad o artefactos en las imágenes generadas.

FID para perros: 462.3222

FID para gatos: 463.7950

Un FID tan elevado sugiere una diferencia significativa entre las imágenes reales y las generadas, lo que implica que los modelos no están capturando adecuadamente las características esenciales de las clases. Estos resultados pueden deberse a limitaciones en el conjunto de datos, insuficientes épocas de entrenamiento, o arquitecturas de modelo que no son lo suficientemente profundas para representar la complejidad de las imágenes de perros y gatos.

CONCLUSIONES

El proyecto demostró que es posible generar imágenes realistas de perros y gatos utilizando GANs, incluso con un conjunto de datos limitado, mediante el uso de técnicas de aumento de datos. La implementación de modelos separados para cada clase permitió que los generadores capturaran las características distintivas de perros y gatos.

El uso de PyTorch y CUDA facilitó el desarrollo y aceleró el entrenamiento de los modelos. Además, la creación de un Dataset personalizado y la aplicación de transformaciones avanzadas fueron clave para manejar la estructura de datos específica y mejorar el rendimiento de los modelos.

Aunque los resultados no son los esperados, siempre hay espacio para mejoras. Incrementar el número de épocas, ajustar hiperparámetros o utilizar arquitecturas más avanzadas podrían mejorar aún más la calidad de las imágenes generadas.

BIBLIOGRAFÍA

- **Goodfellow, I., et al. (2014).** *Generative Adversarial Nets*. Advances in Neural Information Processing Systems, 27.
- **Radford, A., Metz, L., & Chintala, S. (2015).** *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv preprint arXiv:1511.06434.

- **Torchvision Transforms Documentation.** *Torchvision 0.9.1 documentation*. Recuperado de https://pytorch.org/vision/stable/transforms.html
- **Kaggle Dogs vs. Cats Dataset.** Recuperado de https://www.kaggle.com/c/dogs-vs-cats/data