UPC
Universidad Peruana
de Ciencias Aplicadas

Ciencias de la Computación 2022-02

Machine Learning - CC57

TA2

Integrantes :

Daniel Carlos Guillén Rojas - U201920113

Sebastián Alonso Gonzales Sotomayor - U201923816

Antonio Salinas Roca - U201924931

Sección:

CC72

Profesor :

Luis Martín Canaval Sánchez

Octubre, 2022

Índice

1. Introducción
2. Archivos Relevantes
3. Descripción de algoritmos y modelos
4. Descripción de la data
5. Descripción de la configuración
6. Bibliografía

1. Introducción

Las redes adversarias generadoras o GAN (por sus siglas en inglés), son modelos generativos los cuales nos permiten generar datos idénticos y a veces indistinguibles de la data de entrenamiento. Un modelo GAN logra esto implementando dos redes, una red generadora, y una red clasificadora, las cuales son entrenadas en alternancia, con la red clasificadora evaluando el output generado por la red generadora y esta última ajustando su función de pérdida. El resultado de tener estas dos redes en competencia una con otra es un modelo capaz de replicar con alto detalle y precisión la data de entrenamiento, ejemplos de esto son modelos capaces de generar imágenes que aparentan ser fotos de personas reales cuando en realidad son totalmente generadas por el modelo.

Haciendo uso de esta tecnología, en el paper *3D Reconstruction of Incomplete Archaeological Objects Using a Generative Adversarial Network,* Renato Hermosa e Ivan Sipiran (2018) aplican las GAN para crear un modelo capaz de generar fragmentos faltantes de objetos 3D representados como voxels. Bajo este contexto en el presente proyecto busca profundizar el trabajo realizado por Hermoza y Sipirán y aplicarlo junto con tecnología de impresión 3D.

2. Archivos Relevantes

Archivos en el repositorio "3D-ORGAN" relevantes para el trabajo a realizar.

a. reconstruction/model/gan_utils.py

Este archivo contiene la definición de fórmulas y utilidades para implementar el modelo GAN, la penalidad, la formula de perdida de wasserstein y la función para construir el modelo GAN.

b. reconstruction/model/models.py

Este archivo contiene las funciones para construir el modelo Generador y el modelo Discriminador de la GAN.

c. reconstruction/utils/data_prep.py

Contiene la función para simular fracturas en la data

d. datasets/arq_dataset.tar.gz

Este zip contiene el dataset con información de los modelos 3D. De aquí se extraerán los datasets para entrenamiento y testeo.

3. Descripción de algoritmos y modelos

a. GAN

Las GAN son una tecnología reciente dentro de las redes del aprendizaje automático. Estos modelos generan datos nuevos que se asemejan a los datos de entrenamiento. Una GAN posee dos partes:

i. Generador

El generador de una GAN tiene la capacidad de crear datos falsos ayudado con los comentarios del discriminante. Además, logra hacer que el discriminante clasifique el resultado dado como real. La

principal manera de entrenamiento del generador es tomar ruido aleatorio, que se usa para transformar en un resultado significativo.

ii. Discriminador

El discriminante de una GAN se encarga de clasificar e intenta distinguir entre datos reales y los datos entregados por el generador. La principal forma de entrenar el discriminante provienen de dos fuentes: datos reales, como fotos, y datos falsos, generados por el generador.

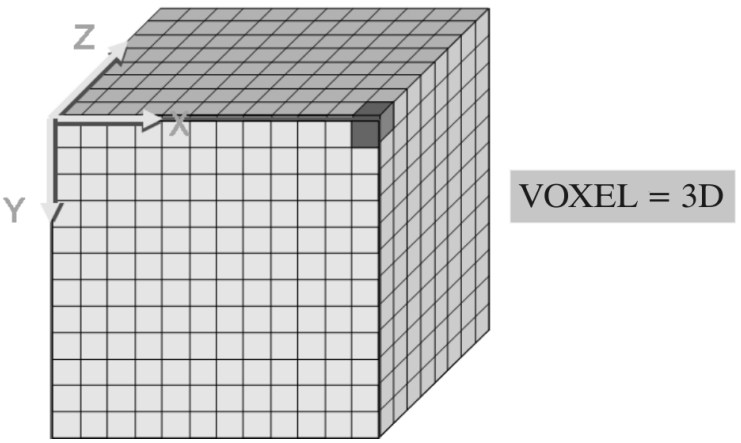
iii. Función de pérdida

Las funciones de pérdida reflejan la distancia entre la distribución de los datos generados por la red y los datos reales. Existen dos funciones comunes de pérdida de GAN, pérdida máxima y pérdida de Wasserstein.

- Pérdida máxima
  Según el artículo elaborado por Ian J. Goodfellow et. al., el generador intenta minimizar la siguiente función, mientras que el discriminante intenta maximizarla.
  $$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$
- Pérdida de Wasserstein
  Esta función depende de una modificación dentro del esquema de GAN o también donde el discriminante clasifica instancias. Estas instancias dan como resultado un número, no limitado entre 1 y 0. Por lo cual no se puede usar un número entre esos valores para validar instancias reales o falsas.
  Las funciones de pérdida que posee:
  - ☐ Pérdida de críticos
    El discriminante trata maximizar la diferencia entre su resultado en instancias reales y su resultado de instancias falsas.
    $$D(x) - D(G(z))$$
  - ☐ Pérdida del generador
    El generador trata de maximizar el resultado del discriminante para sus instancias falsas.
    $$D(G(z))$$

b. Vóxel

Un vóxel (del inglés volumetric pixel) es una unidad cúbica que constituye un objeto tridimensional. Esto significa que en lugar de que los píxeles tengan coordenadas x e y, los vóxeles también tienen coordenadas z, lo que proporciona datos de profundidad. Para crear una imagen 3D, necesitamos hacer una transformación de opacidad en los vóxeles. Esta información proporciona diferentes valores de opacidad para cada vóxel. Esto es importante para ver detalles dentro de la imagen que están oscurecidos por la capa exterior más opaca de vóxeles.



4. **Descripción de la data**
   La data es tomada del dataset contenido en el zip *arq_dataset.tar.gz,* "custom_arq_dataset.npy".
   Se observa que el dataset del repositorio está conformado por dos subdatasets: train y test.
   Ambos se encuentran respectivamente separados en
   - labels
     - El nombre/etiqueta correspondiente a los 11 distintos tipos de objetos en el dataset
     - 'arq', 'bathtub', 'bed', 'chair', 'desk', 'dresser', 'monitor', 'night_stand', 'sofa', 'table', 'toilet'
   - data
     - Guarda matrices booleanas de 32x32x32
     - Representa los objetos como voxels en el espacio 3D de 32x32x32
   - errors
     - Columna en blanco, almacena información relacionada a cualquier error relacionado al punto de data
5. **Descripción de la configuración**
   De estos datasets se extrajeron las entradas etiquetadas como 'table' del dataset de 'train' y 'test'. A estos nuevos datasets extraídos se les generó una copia procesada por la función fragmentadora del repositorio. De esta forma nos quedan datos de entrenamiento completos y fragmentados y datos de testeo, completos y fragmentados. Los objetos fragmentados servirán de input para el Generador y los enteros servirán para entrenar el Discriminador.
   Repositorio con el entorno y dataset creado:
   https://github.com/DanielCGR/ObjectsReconstruction

6. Bibliografía

FILExt (s.f.). Todo sobre los archivos OFF. Recuperado de: https://filext.com/es/extension-de-archivo/OFF [Consulta: 27 de septiembre de 2022].

Hermoza, R. & Sipiran, I. (2018). 3D reconstruction of incomplete archaeological objects using a generative adversarial network. Recuperado de: https://dl.acm.org/doi/abs/10.1145/3208159.3208173 [Consulta: 27 de septiembre de 2022].

Goodfellow, I. J. (2014). Generative Adversarial Networks. arXiv.org. Recuperado de https://arxiv.org/abs/1406.2661 [Consulta: 23 de octubre de 2022]

Voxel | IDIS. (s. f.). Recuperado de https://proyectoidis.org/voxel-2/ [Consulta 23 de octubre de 2022]