



# Redes Adversarias para la generación de Pokemons

---

A quién no le va a gustar...

**Presentación y trabajo creados por:**

Arturo Sirvent Fresneda y  
Ángel Guevara Ros

*Master Ciencia de Datos (UV)*

2022

# Índice

1. Objetivo.
2. Metodología / aproximación al problema.
3. Resultados.
4. Conclusiones.
5. Posibles mejoras y resultados similares.



# 01

## Objetivo

---

¿Qué resultado queremos obtener?



## ¿Qué buscamos?

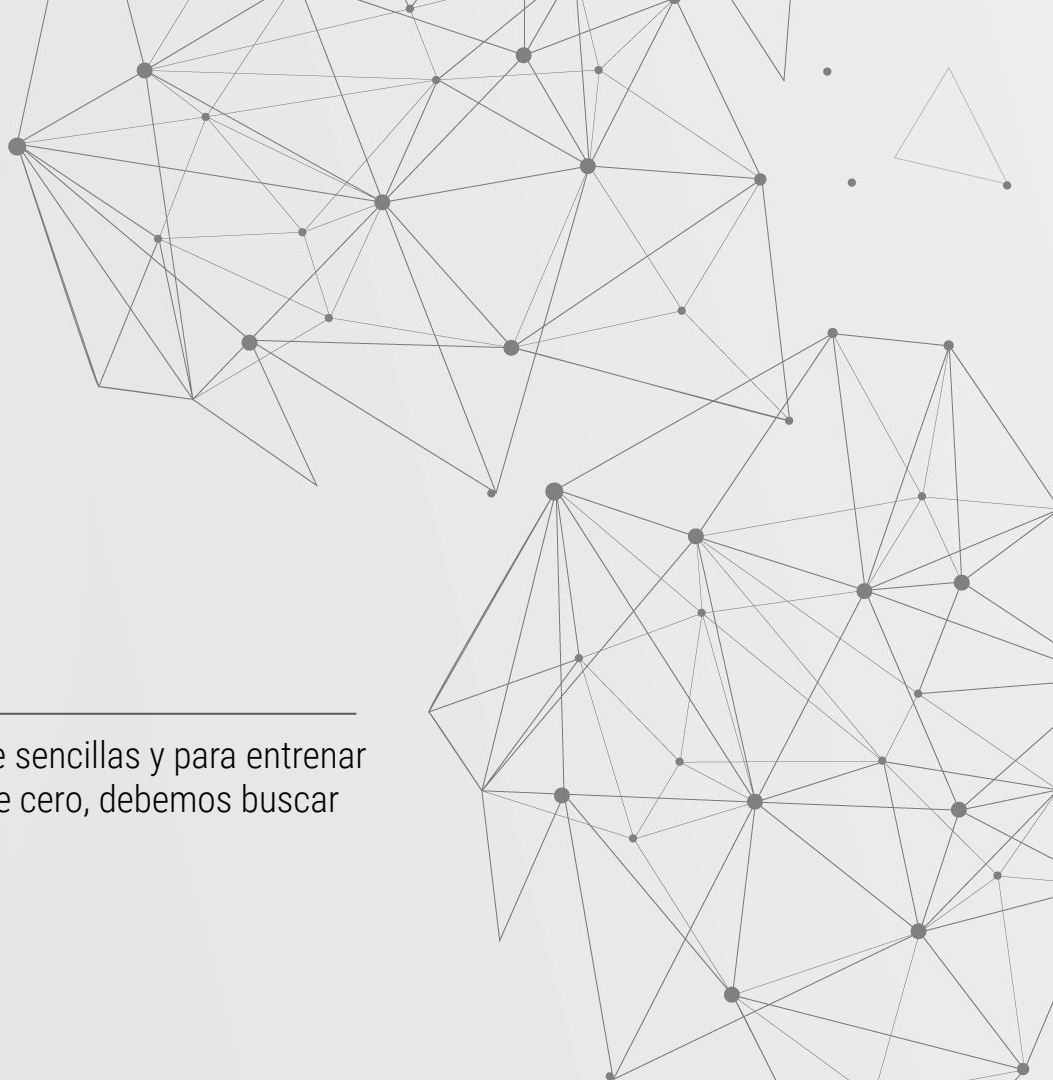
---

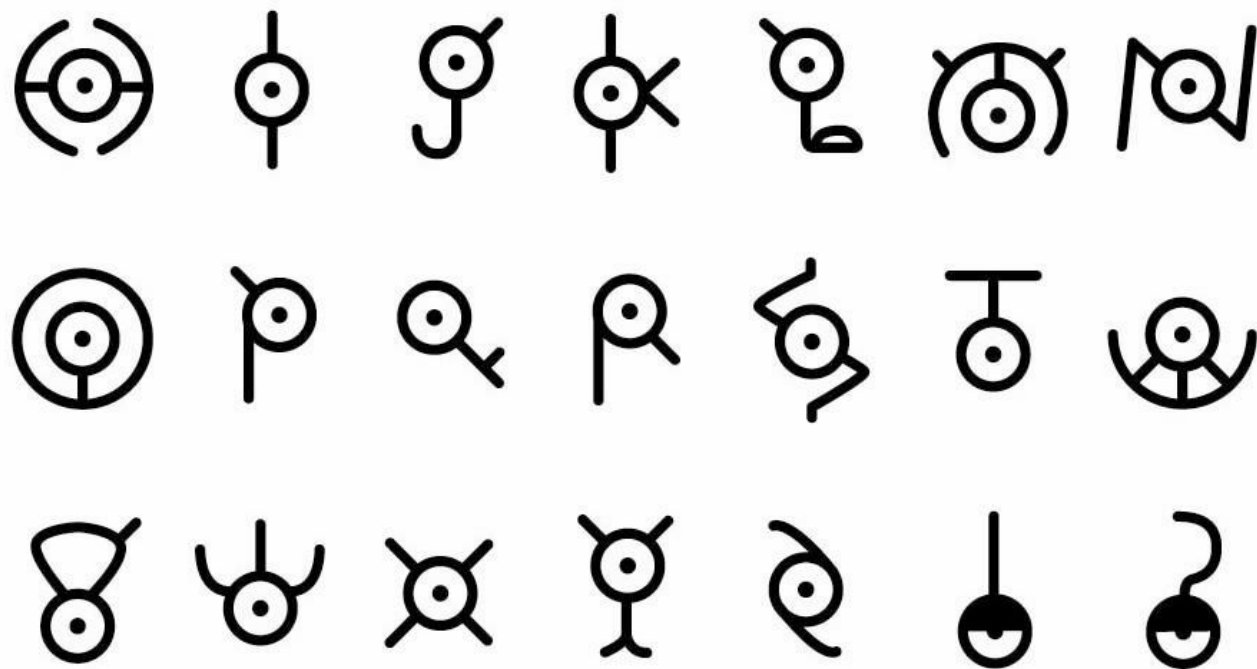
- Queremos crear una red Generativa Adversaria que nos genere más formas como las que le introducimos.
- Buscamos aplicar las técnicas aprendidas en clase, e ir un paso más allá aplicando la estrategia de entrenamiento progresivo.

## ¿Por qué pokemons?

---

- Las formas de estos pokemons son relativamente sencillas y para entrenar nosotros con nuestros medios una red GAN desde cero, debemos buscar problemas no muy ambiciosos.
- Estan chulos.





**Unowns:** Viven en ruinas, en Johto en las Ruinas Alfa, en Isla Séptima en las Ruinas Sete y en Sinnoh en las Ruinas Sosiego. En total **existen 28 formas de Unown**. Cada una representa una letra del abecedario y los signos de exclamación e interrogación.



# 02

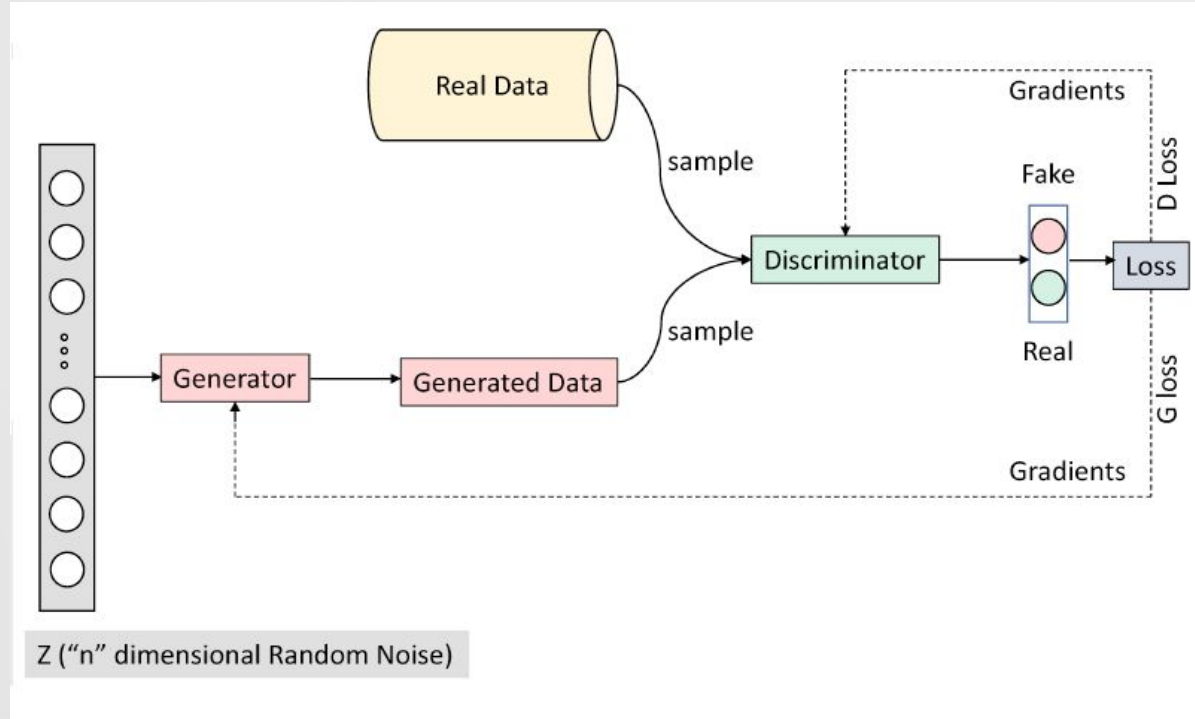
## Metodología

---

¿Cómo lo hacemos? ¿Qué técnicas usamos?

# Teoría GANS

- GAN = Generative Adaptive Networks.
- Técnica de data augmentation.
- El principal objetivo de estas redes es el de generar datos lo más cercanos a la realidad.
- Están compuestas de un Discriminador y de un Generador, que suelen ser redes neuronales.
- Si el Discriminador funciona mal, la GAN no funcionará correctamente.
- En la mayoría de aplicaciones, al final de la convergencia se usa solo el generador.





## Entrenamiento de una GAN

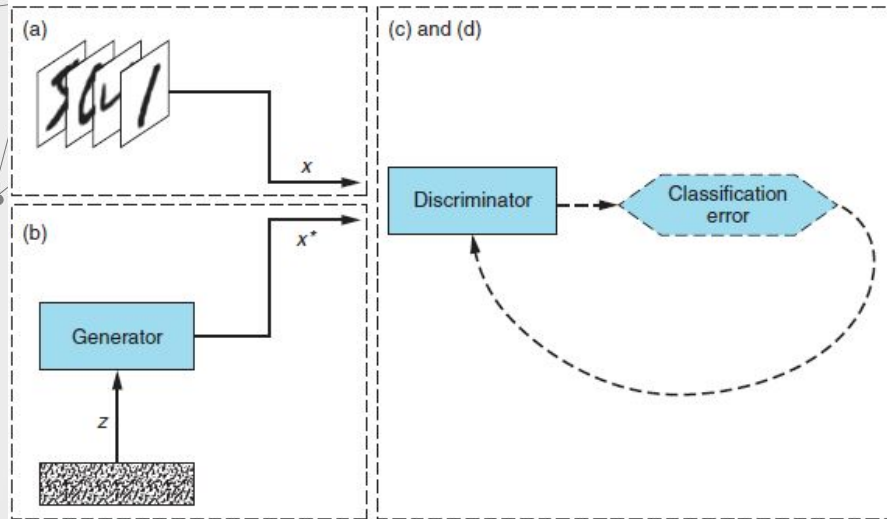
- El entrenamiento se puede expresar como un problema de MinMax.
- Buscamos maximizar los aciertos del Discriminador para que sea exigente, y minimizar el error del Generador para que cada vez produzca datos más parecidos a los reales.

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

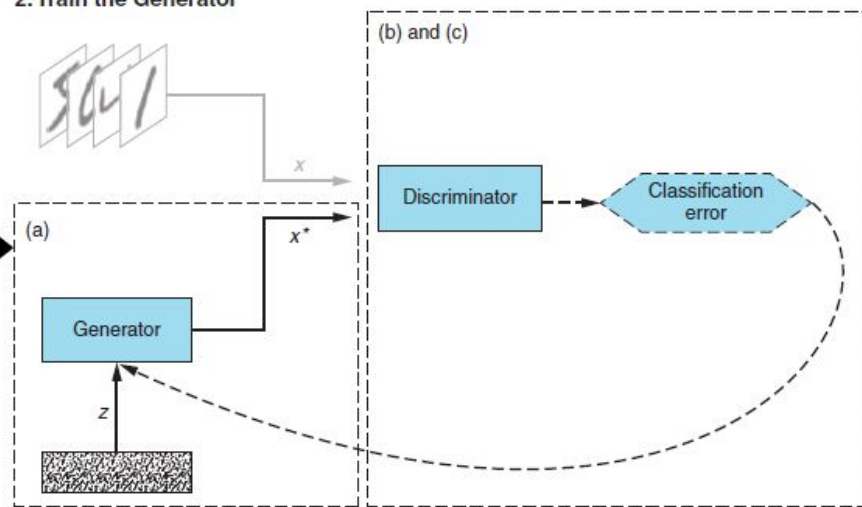


# Entrenamiento de una GAN

## 1. Train the Discriminator



## 2. Train the Generator





## Limitaciones de las GANs

- **No convergencia y colapso de nodos:** los métodos de descenso del gradiente no funcionan bien con los problemas de MinMax. Cuando el Generador es muy bueno, puede colapsar dando lugar a una variedad limitada de muestras.
- **Desaparición del gradiente:** Cuando el Discriminador es muy bueno, éste no proporciona la información suficiente para que el generador progrese y el entrenamiento es para.
- **Desequilibrio :** uno de los dos modelos aprende más rápido que el otro.
- **Inicialización:** es un modelo muy sensible a la selección de los hiperparámetros.

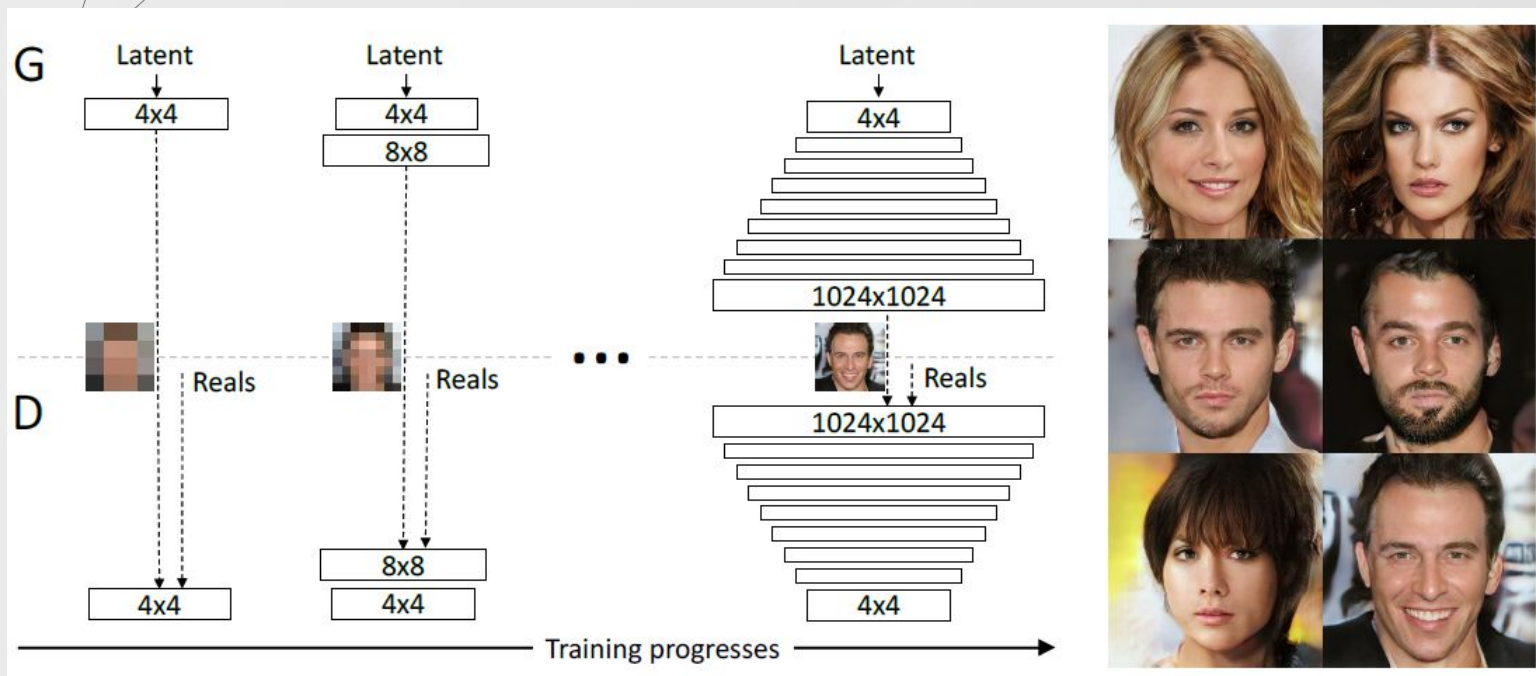


# Entrenamiento PROGRESIVO de una GAN

- La generación de imágenes de alta resolución es complicada y puede llevarnos al problema del gradiente.
- El entrenamiento progresivo nos permite llevar a cabo esa tarea al ir aumentando poco a poco la resolución de las imágenes que le pasamos a la GAN durante el entrenamiento.
- Aprende primero la estructura general de las imágenes y luego va captando los detalles.
- En nuestro caso, nosotros hemos implementado solamente 1 step de entrenamiento progresivo. Hemos empezado con una resolución de 28x28 y después la hemos aumentado a 56x56.

# Entrenamiento **PROGRESIVO** de una GAN

- En la práctica, el hecho de ir aumentando la resolución se traduce en ir aumentando el número de capas ocultas de nuestra red





# 03

## RESULTADOS

---

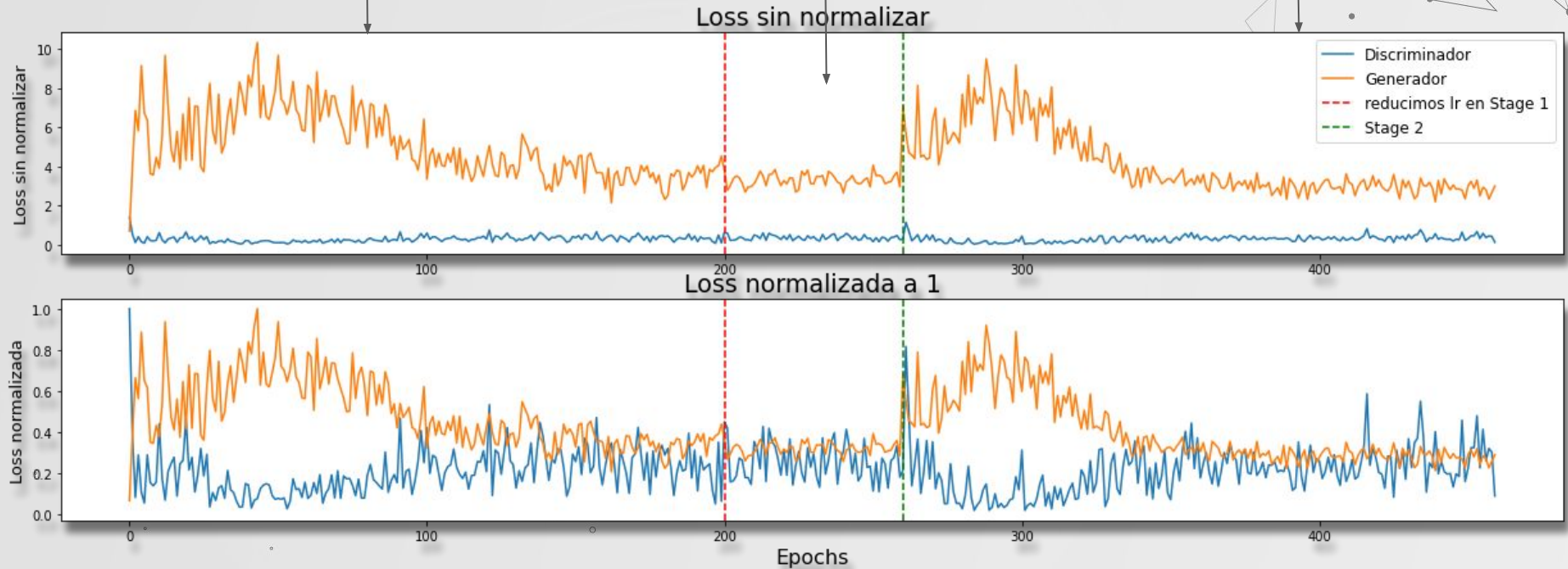
¿Hemos generado Pokemons nuevos?  
¿Cómo ha ido el entrenamiento?

# Proceso de entrenamiento

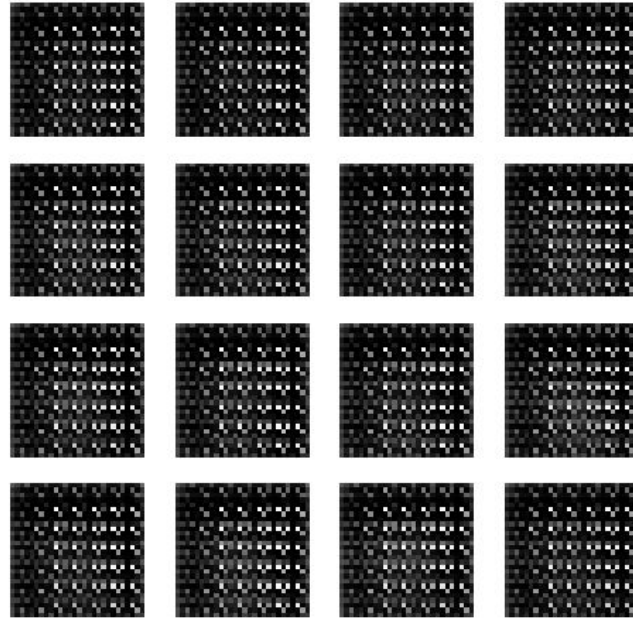
Etapa 1

Fine tuning reduciendo  
el learning rate.

Etapa 2

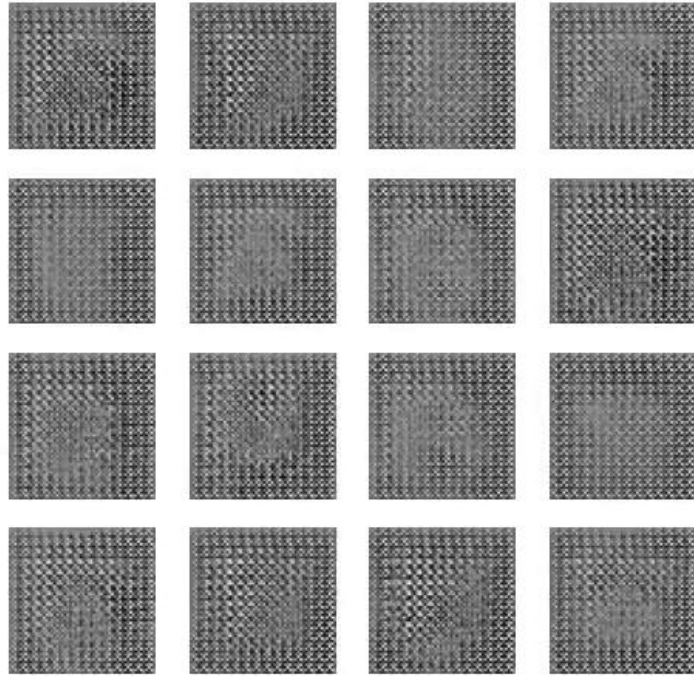


# Imágenes generadas en la Etapa 1



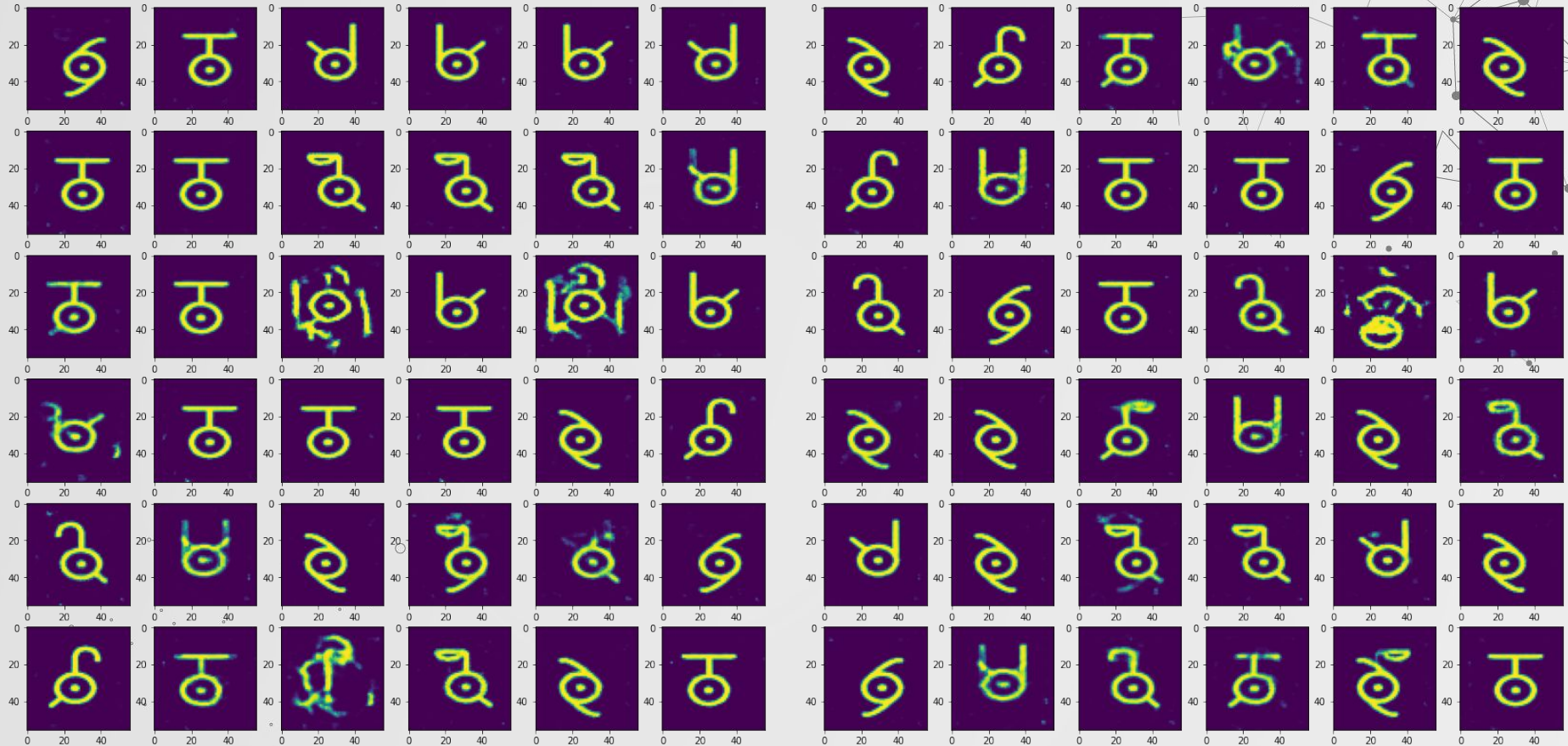


# Imágenes generadas en la Etapa 2

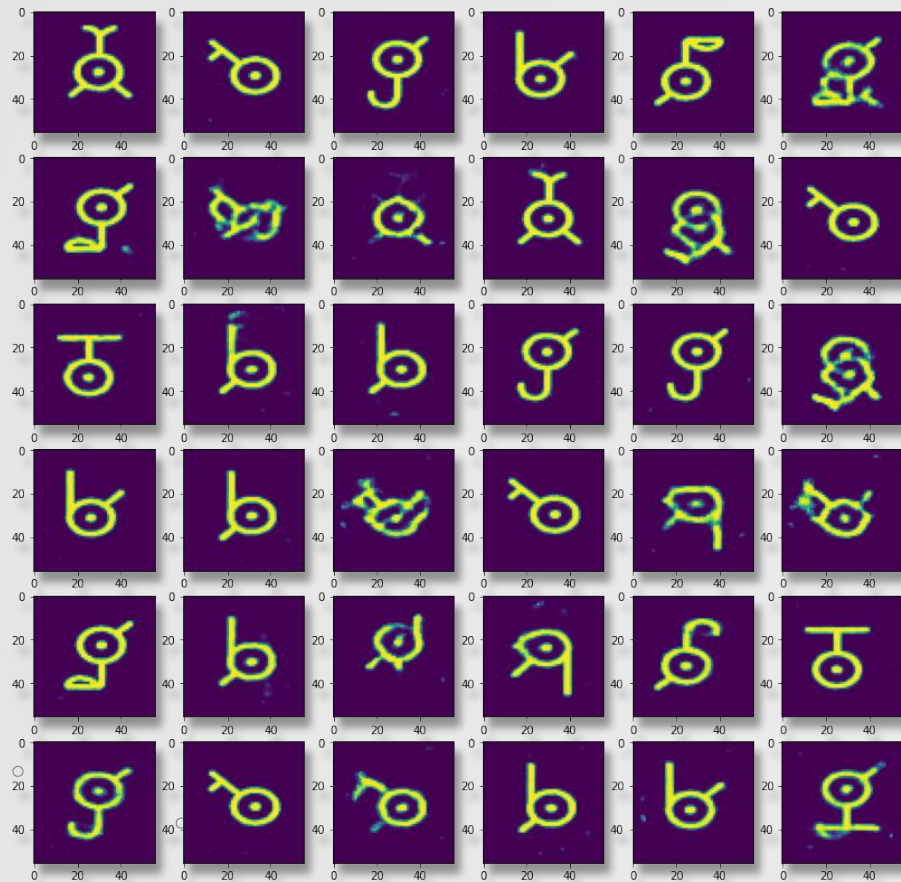




# Imágenes generadas en la **Etapa 2**



# Más imágenes generadas en la **Etapa 2**



# 04

## Conclusiones

---

¿Qué podemos aprender de los resultados obtenidos?

¿Es como esperábamos?



# Beneficios del entrenamiento progresivo



Los principales beneficios del **entrenamiento progresivo** son:

- Menor tiempo de entrenamiento.
- Mayor estabilidad durante el entrenamiento, asegurando la convergencia y evitando el colapso.

En nuestro caso, hemos observado ambos efectos.

- Se acelera el proceso de entrenamiento de la red completa (más alta resolución) al partir de capas preentrenadas.
- En nuestro caso, una imagen de 56x56 no era posible con nuestros recursos (hardware), en un solo entrenamiento, pero al segmentarlo en dos etapas, sí se logró la convergencia.
- Sin embargo no hemos podido evitar algo de colapso, debido probablemente al reducido dataset.

# 05

## Redes similares y alternativas posibles

---

¿Qué más se ha hecho sobre el tema?

¿Cómo podría eso implementarse a nuestro modelo?



# Redes GAN famosas

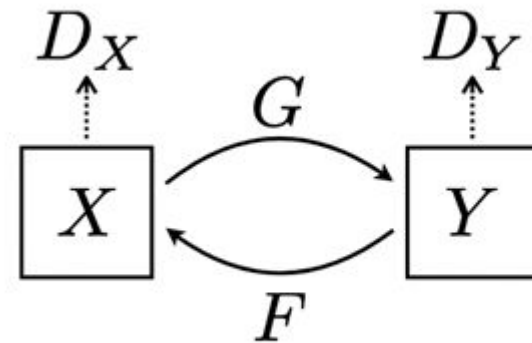
Las principales diferencias entre una GAN y otra son:

- La función de coste a minimizar y maximizar
- La arquitectura de las redes neuronales que se utilizan para el generador y el discriminador.

Algunas de las variantes más conocidas son:

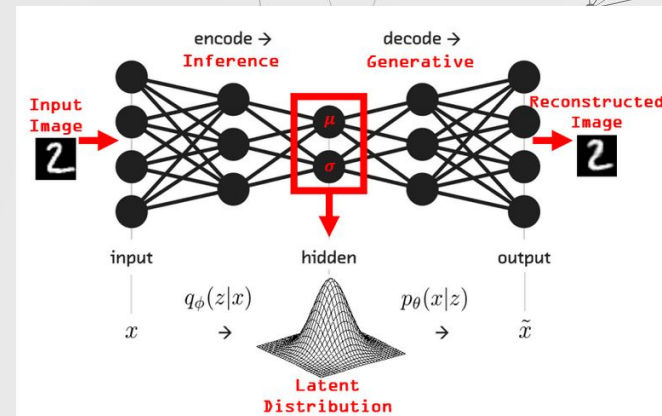
- **DCGAN** (Deep Convolutional GAN)
- **LSGAN** (Least Square GAN)
- **WGAN** (Wasserstein GAN)
- **CYCLEGAN**

**CYCLEGAN**



# Técnicas alternativas

- Autoencoders variacionales.
- Redes de Difusión.





# Gracias

¿Preguntas?

## Recursos y bibliografía usada:

- **Progressive training:**  
<https://arxiv.org/abs/1710.10196>
- **Teoría y notebooks de clase (asignatura de Deep learning MCD-UV)**

*Presentación hecha para como parte de la  
evaluación del Máster en Ciencia de Datos (UV)*