Red Generativa Adversaria para la creación de rostros de anime

Bergna Aguilar, Diego
Universidad Nacional de Ingeniería
Lima, Perú
diego.bergna.a@uni.pe

Mechan Osorio, Rodrigo
Universidad Nacional de Ingeniería
Lima, Perú
rodrigo.mechan.o@uni.pe

Paima Mijahuanca, Kevin
Universidad Nacional de Ingeniería
Lima, Perú
kpaimam@uni.pe

***Resumen*—Las animaciones japonesas, o anime, han llegado a ser tantos que actualmente uno puede confundir el rostro de algún personaje de anime con alguno de otro anime; esto posiblemente debido a la falta de originalidad y creatividad o esfuerzo que le dedican los ilustradores. Añadamos también que por temporadas llega a existir una tendencia en el estilo de dibujo de los anime, con lo que aumenta la probabilidad de que muchos de estos rostros se parezcan.**

El presente trabajo implementa una red generativa adversaria (GAN - por sus siglas en inglés), la cual es capaz de generar nuevos rostros de personajes de anime al azar al emplear dos redes que compiten entre sí, la primera red es una generadora de rostros de anime mientras que la segunda es una red que clasifica la salida que da la red generadora, logrando así resolver el problema al generar rostro para un nuevo personaje de anime.

Índice de Términos — GAN, red generadora, red discriminadora, rostros de anime, convoluciones, convoluciones transpuestas.

I. Introducción

El anime es una parte de la industria del entretenimiento japonesa el cual genera una gran cantidad de ingresos, la cual ha logrado alcanzar una cantidad de 19,9 mil millones de dólares en el año 2019, sin embargo el ámbito comercial del anime se ha visto envuelta en problemas por derechos de autor debido a la similitud que tienen los personajes de distintas empresas llegando a generar pérdidas debido a juicios entre ellas.

Con ello como problemática, entre las posibles soluciones se encuentra la de idear un modelo de inteligencia artificial que pueda generar nuevos personajes de anime no existentes.

Este proyecto se centra en la realización de un modelo de redes generativas adversarias con el fin de generar rostros de personajes de anime.

I-A. Objetivos

- Construir una red generativa adversaria para la generación de rostros de anime.

I-A1. Objetivos específicos:

- Implementación de la red generativa.
- Implementación de la red discriminadora.

I-B. Resultados esperados

Dado nuestros objetivos, esperamos lograr la implementación de una GAN funcional con una arquitectura personalizada empezando por nuestra red discriminadora al usar como referencia base la arquitectura de la DCGAN [9] modificada acorde a arquitecturas de clasificadores conocidos como VGG y ResNet.

Para la arquitectura de la red generadora se planea su creación como un proceso inverso a la arquitectura del discriminador con algunas variaciones.

I-C. Visión más allá de este artículo

A futuro esperamos que esta GAN no solo genere imágenes de rostros de anime, sino que sea posible personalizar o direccionar las características que uno quiere obtener tal como color de cabello, color de rostro, género, etc. también queremos que sea posible generar imágenes de cuerpo completo de personajes de anime.

Se espera también poder dejar el uso de redes convolucionales convencionales para hacer uso de nuevos enfoques como los Transformers que permiten la generación de imágenes en alta resolución [2] y reemplazar así a la clásica red generadora.

II. GAN's

Las Redes Generativas Adversarias [3], mejor conocidas como GAN's por sus siglas en inglés, fueron introducidas para resolver la problemática de la complejidad de cálculo que tenían los modelos generativos profundos [1]. Las GANs cuentan con dos modelos: el modelo generativo y el modelo clasificador, en el cual el modelo generativo(G) se enfrenta al modelo discriminador(D) donde este último aprende a determinar si la muestra generada por (G) pertenece o no al conjunto de los datos reales con los cuales fue entrenado. Una analogía es que nuestro modelo generativo sea un falsificador de obras de arte mientras que nuestro modelo discriminador es un experto anticuario que intenta detectar si la obra presentada por (G) es real o falsa, aunque es mejor tomarlo como que (D) clasifique la data en base a la probabilidad de

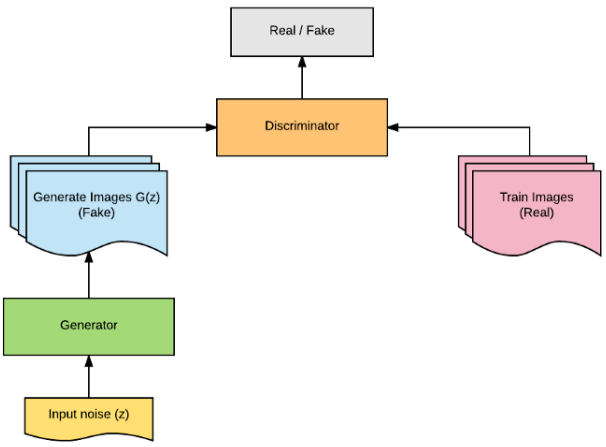

Figura 1. Esquema simple del funcionamiento de una GAN
`Fuente:` https://kknews.cc/code/4v9j3yq.html

que la información presentada por (G) pertenezca a una categoría.

Este proceso se repetirá hasta que nuestro modelo discriminatorio sea incapaz de detectar que la muestra creada por nuestro modelo generativo es falso.

III. Revisión de Literatura

III-A. Cadena de búsqueda

Se ha buscado en la plataforma Arxiv, Medium, IEEEXPLORER y el navegador de Google Scholar en donde se realizaron las siguientes consultas:

- **TITLE-ABS-KEY** (Generative Adversarial Network, types)
- **TITLE-ABS-KEY** (Convolutional Neural Network, GAN)
- **TITLE-ABS-KEY** (Image generator, GAN, tips, use)
- **TITLE-ABS-KEY** (DEEP CNN for image, GAN, anime)
- **TITLE-ABS-KEY** (GAN, anime generator)
- **TITLE-ABS-KEY** (Face generator, cnn, GAN)

III-B. Preguntas de Revisión

Las preguntas que tomamos como referencia para nuestro proyecto fueron:

- ¿Cómo funcionan las GANs?
- ¿En que se aplican las GANs?
- ¿Cómo se genera imágenes empleando GANs?
- ¿Cómo funciona un generador?
- ¿Cómo funciona el discriminador?
- Sugerencias para GAN estables

III-C. Resultados de Revisión

En [3], fue donde encontramos una descripción de como son las GANs y como es su funcionamiento en donde plantearon una idea considerada innovadora para ese tiempo en el que predominaban redes de alta complejidad de cálculo. Como se mencionó en el apartado de GAN (II), las GANs constan de 2 modelos que compiten entre sí, esta competencia se basa en el enfoque MinMax en donde el discriminador y el generador aprenden la distribución de la información, así el discriminador se especializa en reconocer la distribución de la data real mientras que el generador aprende a imitar esta distribución en base a la puntuación obtenida al evaluar su información generada en el discriminador. Con ello, el discriminador busca *maximizar* la diferencia entre su distribución y la distribución que produce el generador mientras que el generador busca *minimizar* la diferencia entre su distribución generada y la distribución que el discriminador posee. En la Figura 2 podemos observar un cuadro con los enfoques de pérdida empleados en algunos modelos de GANs.

Mientras se investigaba qué otros usos se les puede dar a las GANs se encontró este artículo [4] en el cual se muestran algunas aplicaciones como:

- Aumentar la resolución de imágenes
- Síntesis y manipulación de imágenes
- Síntesis de Textura
- Detección de objetos
- Procesamiento de lenguajes naturales

Durante la investigación del funcionamiento básico de una GAN se encontró que la arquitectura de la DCGAN [9] es la GAN de introducción que da la bienvenida a los diferentes enfoques de las GANs, este modelo emplea redes neuronales convolucionales para el aprendizaje no supervisado, brindando al generador y al discriminador arquitecturas opuestas.

Sin embargo al experimentar un modelo de borrador usando DCGAN se encontraron inestabilidades durante el entrenamiento por lo que al investigar sobre pasos para una DCGAN estable:

- Reemplazar las convoluciones simples por convoluciones estriadas (p.e. `stride=2`) para el modelo discriminatorio y convoluciones parcialmente estriadas para el modelo generador (p.e. `stride=(1,2)`) [9].
- Usar normalización batch [6] en el modelo generativo y discriminatorio.
- Eliminar completamente todas las capas convolucionales conectadas; es decir, hacer uso de DropOut.
- Usar la función de activación ReLU [8] en el modelo generador para todas las capas excepto para la salida, el cual usa la función hiperbólica.
- Usar la función de activación LeakyReLU [10] en el modelo discriminante para todas las capas.

GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGANGP}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\nabla D(\alpha x + (1-\alpha\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x)-1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x}-1))^2]$
DRAGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\nabla D(\hat{x})\|_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \mathrm{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$

Figura 2. Distintas funciones de pérdida de las GANs
`Fuente`: Are GANs Created Equal? A Large-Scale Study [7]

IV. Metodología

Procederemos a detallar el planteamiento del desarrollo del proyecto en base a los objetivos establecidos.

Previo a la creación de un modelo, se debe analizar el conjunto de datos con los que se trabajará.

Para propósito de nuestro proyecto optamos por un dataset de rostros de anime. El dataset elegido es: Generating Anime Faces. El cual consta de un conjunto de 63500 archivos aproximadamente con rostros de anime genéricos y conocidos, muchos de ellos cuentan con artículos en el rostro e incluso algunos con rostros de colores fuera de lo común. Durante la exploración del conjunto de imágenes se encontró que la mayoría los rostros pertenecen a personajes femeninos, pero debido a fines prácticos no le daremos importancia por ahora.

IV-A. Implementación de la red generativa y la red discriminadora

IV-A1. Red discriminadora: La red discriminadora puede interpretarse como una red clasificadora que indica la probabilidad de que la imagen recibida pertenezca a una categoría objetivo, en este caso, la probabilidad de que la imagen, o lote de imágenes, tenga las características de un rostro de anime, estas características se obtienen al analizar la distribución de la información.

Se ha tomado a [9] y [5] como referencias para esta red debido a que se busca obtener una red ligera y que sea capaz de obtener características necesarias para la correcta clasificación de las imágenes.

Para el entrenamiento de nuestra red discriminadora, como entrada tendremos un conjunto imágenes PNG con dimensiones 128x128 px las cuales pasarán por una transformación para obtener su información en forma de tensores.

IV-A2. Red generativa: Al haber elegido las redes de referencia para nuestro modelo discriminante, se utilizará el proceso inverso del modelo (D) para la creación de nuestro modelo (G), posterior a ello se realizarán pruebas de estabilidad y se cambiarán las configuraciones de los hiperparámetros de manera que se obtenga los adecuados para una óptima implementación.

Como entrada se tendrá tensores de ruido los cuales se someterán a diversos procesos como convoluciones transpuestas, normalizaciones, dropout y UpSampling para dar lugar a una imagen la cual será evaluada por la red discriminante y esta devolverá una puntuación acorde a su criterio para informar a la red generadora si va por buen camino al crear las imágenes o no.

Pese a que se menciona que se generará una imagen, realmente se están generando un lote de tensores con diversas distribuciones que buscan orientarse con la puntuación obtenida, cada uno de estos tensores representan a imágenes. Este lote es el que se evalúa en el discriminador.

IV-B. Entrenamiento

En las GAN's se deben entrenar a la vez ambos modelos, esto se debe a que si se entrenase por separado la red discriminadora de la generadora, una de ellas terminaría especializándose primero en su campo con lo que no daría oportunidad a la otra red de aprender de sus errores. Imagina que el discriminador sea muy bueno en su trabajo, siempre estaría seguro que la información que recibe por parte del generador es falsa por lo que al devolverle su puntuación a la red generadora, esta última no tendría dirección para empezar su aprendizaje por lo que estaría destinada a nunca aprender.

Sucede lo mismo con la red generadora, si esta se entrenase primero, sería tan buena que la red discriminadora nunca podría detectar si una imagen es tiene rostro de anime o no. Por lo que siempre categorizará lo que sea que envíe la red generadora como rostro de anime, y eso perjudica también a la red generadora pues podría darse el caso que sus imágenes empiecen a mutar y siga creyendo que hace un buen trabajo generando rostros de anime.

Un problema común que sucede durante el entrenamiento de GANs es que el generador (G) de con un mínimo local, en términos simples significa que el generador (G) ha aprendido un patrón (es decir una distribución) en la que obtiene una buena puntuación por parte del discriminante (D) por lo que tendrá tendencia a repetir el proceso y llegar a resultados semejantes. Cuando sucede lo anterior, el generador deja de aprender a generalizar. A este problema se lo conoce como mode collapse.

Se realizarán pruebas para distintos intervalos de entrenamiento como 10, 20, 40 y 50 épocas de entrenamiento y observar cómo mejoran ambas redes al generar y distinguir entre lo que podría ser un rostro de anime y una imagen cualquiera.

Si se detecta que en algún momento no hay mejoras en el entrenamiento, se decidirá modificar los hiperparámetros o la arquitectura misma a fin de encontrar un mejor desempeño para ambas redes y obtener así mejores resultados.

IV-C. Complementos

Dataloader. Nuestro dataloader primero realiza una confirmación del tamaño de nuestra imagen y en caso sea diferente de 128 x 128px lo redimensiona a ese tamaño; luego las imágenes se pasan a tensores y las normalizamos. Estos se toman en pequeños batch para ingresar al entrenamiento.

Carga de datos. Para poder realizar la carga e iteración con los datos se usa nuestro Dataloader.

Noise Generator. Generará un vector aleatorio el cual sera del tamaño del batch que toma el dataloader en una época el cual tendrá una profundidad de 100.

V. Experimentación y resultados

En esta sección se mostrarán resultados obtenidos para experimentaciones realizadas.

V-A. Limpieza de datos

Lo primero en realizarse fue un análisis del dataset, de las 63500 imágenes muchas de ellas tuvieron dimensiones diferentes, por lo que fueron redimensionadas a imágenes en formato PNG de 128px *x* 128px.

Posterior a ello se realizó una búsqueda de duplicados y se encontraron alrededor de 21000 imágenes repetidas las cuales fueron eliminadas dejando solo un ejemplar de las mismas para obtener finalmente 41,500 imágenes para el entrenamiento aproximadamente.

V-B. Versión 1

Analizando las arquitecturas de DCGAN, VGG y de ResNET, se optó inicialmente por un modelo alterado de DCGAN, añadiendo una capa más de convolución y variando la función de activación ReLU por LeackyReLU con un escape de 0.01.

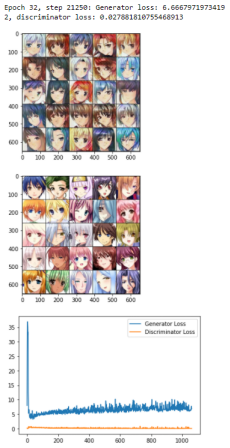

Figura 3. Resultados de la GAN v1
Fuente: autoría propia

V-B1. Definición del modelo discriminante: Este modelo contaba con una arquitectura de 6 bloques: 5 convoluciones internas, BatchNorm2d posterior a las convoluciones seguidas de ReLU y una convolución final seguida de la función de activación Sigmoid, dando como resultado un tensor unidimensional.

V-B2. Definición del modelo generador: Este modelo contaba con una arquitectura inversa del modelo discriminante, tomada de entrada un tensor de ruido $(100, 1, 1)$ y como salida un tensor $(3, 128, 128)$ que representa a las distribuciones de la imagen en los canales RGB.

V-B3. Resultados: En la Figura 3 se puede observar los resultados obtenidos en esta versión de la red durante la época 32.

V-C. Versión 2, 3, 4 y 5

Posterior a la implementación de la versión 1, teníamos nuestras dudas de la arquitectura planteada dado que era demasiado simple por lo que se decidió hacer un replanteo de la misma y tomar como base de partida la combinación de las arquitecturas VGG16 y ResNET18.

V-C1. Definición del modelo discriminante: Para nuestro modelo discriminador, se desarrolló como un modelo simple de clasificación binaria el cual indica si la información brindada por la red generadora coincide con la distribución de las imágenes que el modelo considera real.

El modelo consta de 7 capas, en donde la primera capa realiza un *Conv2d*,un *BatchNorm2d*,un *LeakyReLU* y un *MaxPool2d*.

Las otras capas realiza un intercambio entre:

- Aplicar *Conv2d*, *BatchNorm2d* y *LeakyReLU* además de mantener el tamaño de la dimensión y de la imagen.

- Aplicar *Conv2d*, *BatchNorm2d* y *LeakyReLU* además de aumentar al doble el tamaño de la dimensión.

Para la capa final, Se Realiza una *AvgPool2d*, *Conv2d* y finalmente aplicamos la función de activación *Sigmoid*() como función de activación final.

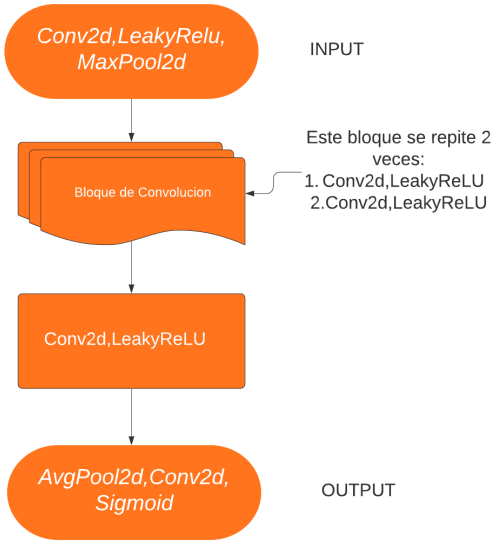

Figura 4. Arquitectura de red discriminadora
Fuente: autoría propia

V-D. Definición del modelo de la red generadora

Se optó por tomar como referencia base la inversa de la arquitectura de la red discriminadora añadiendo dos capas extras. Este modelo toma un vector de ruido, y mediante un conjunto de convoluciones transpuestas, normalización por lote, funciones de activación intermedias ReLU, y una función de activación final Tanh.

El modelo consta de 9 capas, en donde la primera capa incluye *ConvTranspose2d*, *BatchNorm2d*, *DropOut* y *UpSampling*.

Las otras capas usan un intercambio entre:

- Aplicar *ConvTranspose2d*, *BatchNorm2d* y *ReLU* además de mantener el tamaño de la dimensión y de la imagen.
- Aplicar *ConvTranspose2d*, *BatchNorm2d*, *DropOut* y ReLU además de reducir a la mitad el tamaño de la dimensión.

Para la capa final, realizamos una *ConvTranspose2d*, *BatchNorm2d* y finalmente aplicamos la función *Tanh*() como función de activación final.

A diferencia de la red DCGAN se ha buscado experimentar tanto con los valores del *kernel*, *stride* y *padding* de cada capa de convolución sin buscar afectar el tamaño de salida de la misma, también se buscó valores para el *learning rate* de los optimizadores de los modelos.

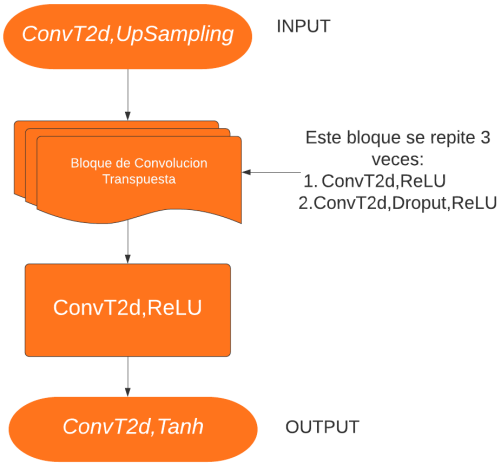

Figura 5. Arquitectura de red generativa
Fuente: autoría propia

V-D1. Resultados:

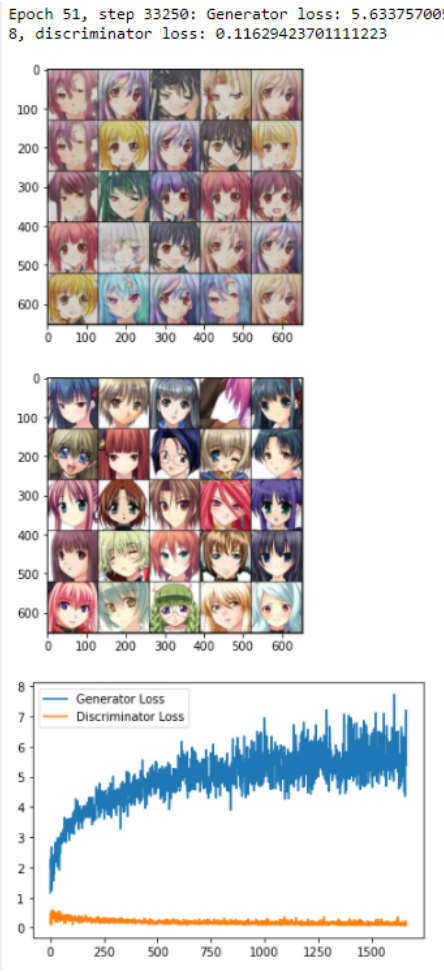

Figura 6. Resultados de la GAN v2
Fuente: autoría propia

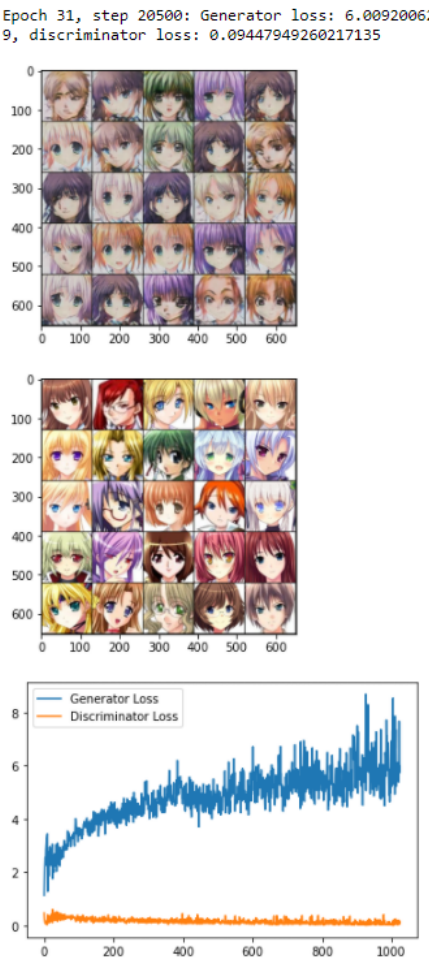

Figura 7. Resultados de la GAN v3
Fuente: autoría propia

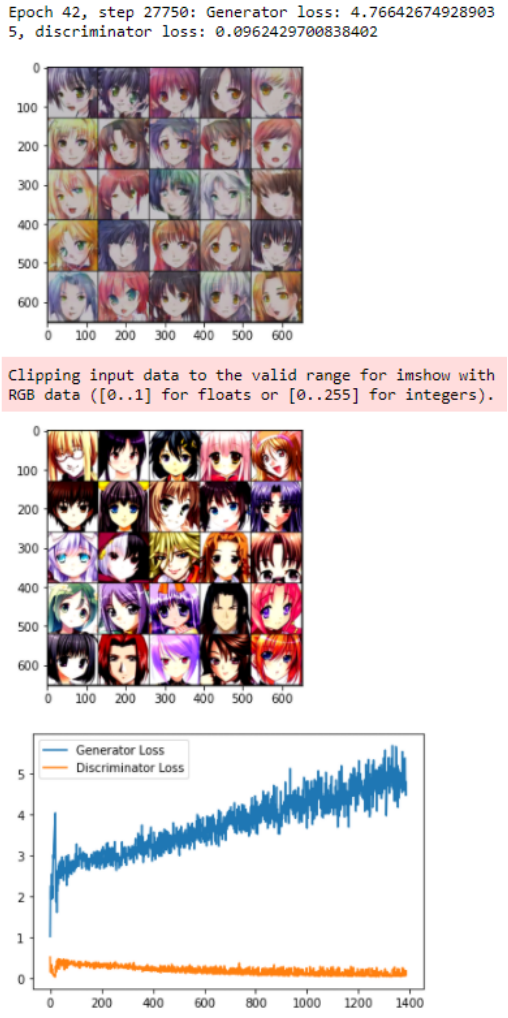

Figura 8. Resultados de la GAN v4
Fuente: autoría propia

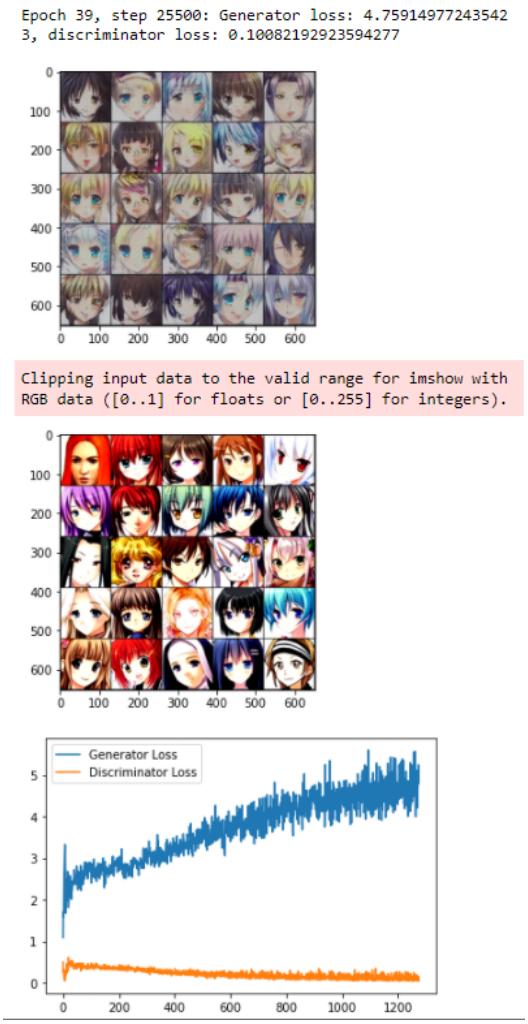

Figura 9. Resultados de la GAN v5
Fuente: autoría propia

VI. Discusión de resultados

En esta sección se hablará de los resultados y presentando argumentos que respalden los mismos.

Los resultados obtenidos al variar la arquitectura y los hiperparámetros fueron aceptables, se puede notar algunas imágenes generadas que carecen de forma definida en todos las versiones de nuestra GAN, sin embargo ya en todas las versiones se ha obtenido los resultados esperados siendo la Figura 10 el resultado final de nuestra GAN.

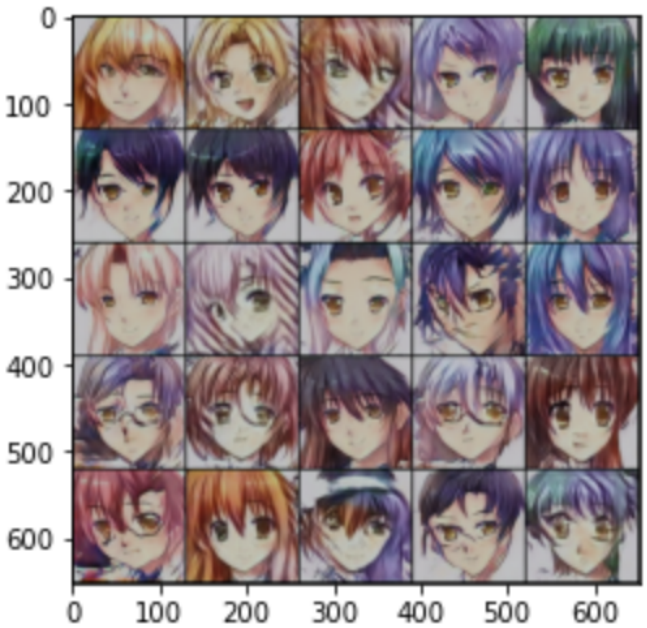

Figura 10. Resultados final
Fuente: autoría propia

VII. Conclusiones

Se logró la implementación correcta de ambas redes de la GAN, siendo la generadora una variante de DCGAN en conjunto con VGG y ResNET la cual dió buenos resultados.

Así mismo la red Discriminadora logró su propósito al dar valores adecuados a la red generadora para que esta aprenda correctamente completando así nuestra GAN funcional.

Referencias

[1] Ludovic Arnold and Yann Ollivier. Layer-wise learning of deep generative models, 2013.

[2] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[4] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *CoRR*, abs/2001.06937, 2020.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[7] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study, 2018.

[8] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[9] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[10] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.