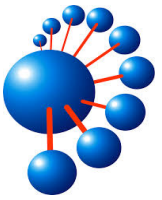

Universidad de Concepción

Facultad de Ingeniería

Proyecto Semestral

Deep Learning

Martín Molina
Iván Astorga
Vicente Lermanda

Julio 2022

Índice

1. Introducción

Una de las mayores capacidades que han demostrado las redes neuronales a través de los años es la de resolver problemas de clasificación con una alta precisión (Krizhevsky et al. 2012) y la generación de nuevos datos.
En el presente informe se trabajará con datos correspondientes a imágenes de famosos con el fin de cumplir dos objetivos.
El primer objetivo aborda la clasificación de atributos de las personas, de los cuales se verificará cual es el sexo de la persona de la imagen, si posee lentes y su gesticulación facial, mediante el uso de redes convolucionales. Por otra parte, el segundo objetivo esta relacionado a la generacion de imágenes nuevas utilizando Generative Adversarial Networks (GANs) (Goodfellow et al. 2014).

2. Estudio Bibliográfico

En el 2010 se entreno una CNN para clasificar 1.2 millones de imágenes de alta resolución, todo esto en el contexto del concurso ImageNet LSVRC-2010 en el cual se pide clasificar dichas imágenes en 1.000 categorías diferentes.
La red Constaba de 60 millones de parámetros y 650.000 neuronas, además, se regularizo con Dropout el cual era un método reciente que funciono bastante bien.
Posteriormente el año 2012 se introdujo una variante de este modelo en la competencia LSVRC-2012 obteniendo el primer lugar superando ampliamente al segundo lugar.

En el año 2014 se utiliza por primera vez el término GAN (Goodfellow et al. 2014) donde se propone un nuevo modelo de redes neuronales generador, basado en un proceso de adversario, donde dos sub-modelos, generador y discriminador, "compiten" en un problema de optimización (o juego de dos jugadores) minimax.
Esta competencia, a grandes rasgos se basa en que el modelo generador recibe ruido de input y lo transforma en datos parecidos a los originales, mientras que el discriminador debe distinguir si estos son reales o creados por el generador.

3. Implementaciones

3.1. Modelo Discriminativo

3.1.1. Pre-Entrenamiento

Antes de realizar el entrenamiento se requiere definir el espacio de trabajo, el cual en este caso es Google Colab, en donde se utilizó Jupyter Notebooks con el lenguaje Python incorporado. El siguiente paso consiste en importar todas las herramientas necesarias para el trabajo, entre ellas se destacan *Numpy*, *Pandas* y *Matplotlib* (para manipular, analizar y visualizar datos), como backend **Tensorflow**, del cual se puede importar **Keras** y desde acá se importa el modelo, capas y herramientas de procesamiento de imágenes.

3.1.2. Datos

La primera tarea consiste en crear un modelo discriminativo que sea capaz de clasificar ciertos atributos presentes en las imágenes. Para esto se propone un modelo de aprendizaje profundo basado en una arquitectura de redes neuronales convolucionales, ya que, dichas redes presentan una gran capacidad para reconocer características y por ende, realizar el trabajo.

Primeramente, el dataset contiene imágenes de rostros de famosos, principalmente de habla inglesa. Cada imagen cuenta con atributos binarios (40 atributos), de estos se escogieron los que se teoriza que pueden dar mejores resultados, los cuales son: Género, si lleva gafas y si está sonriendo.

Figura 1: Muestra de imagenes del dataset.

Antes de poder cargar los datos son divididas las 202.599 imágenes en, 162.770 destinadas al entrenamiento, 19.867 destinadas a validación y 19.962 a test. Se tomaron 3

muestras de cada conjunto de datos (una por cada atributo a clasificar) de manera aleatoria y además, dicha muestra, se balanceó de manera que el 50 % de los datos presenta el atributo y el otro 50 % no lo presenta, además, se normalizaron los datos. Quedando en 10.000 datos de entrenamiento, 2.000 de validación y 1.000 para el test.

3.1.3. Modelo

Como se mencionó anteriormente, se utiliza un modelo de redes neuronales convolucionales, como Backend **Tensorflow** y la arquitectura es creada utilizando **Keras**. Se Inicializa un modelo secuencial, descrito por el siguiente diagrama:

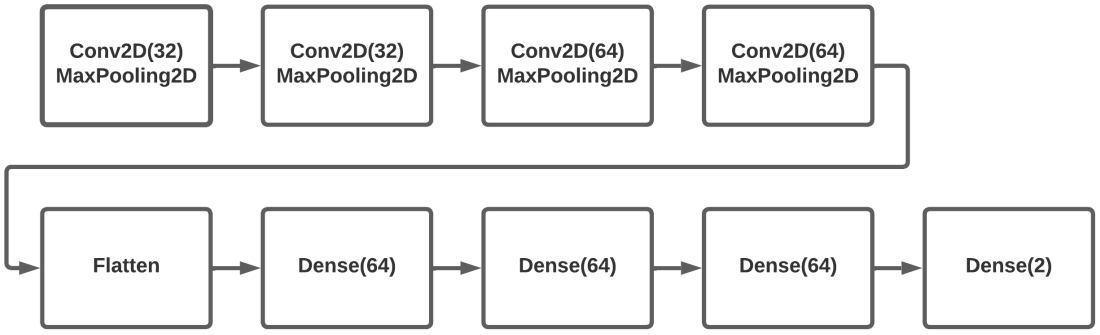

Figura 2: Arquitectura de la Red Clasificadora

La dimensión de entrada a las capas convolucionales es de $(218, 178, 3)$, cada una de ellas tienen un *padding* con *stride* de $(1, 1)$, un *MaxPooling* de $(2, 2)$ y la dimension del kernel de 2, además, se activan con la función **ReLu**. Luego se pasa por una capa **Flatten** para *vectorizar* la salida de las capas convolucionales para luego pasar por 3 capas Densas de 64 neuronas con activación *sigmoide* y una última capa densa con 2 neuronas con activación *softmax*. Con esta arquitectura se entrenaron 3 modelos clasificadores: Modelo 1 (Género), Modelo 2 (Lentes), Modelo 3 (Sonriendo).

3.1.4. Entrenamiento

Para el entrenamiento, se utiliza como función de perdida **Binary Crossentropy**, como optimizador **RMSprop** y como metrica de evaluación **Accuracy**. Luego para el ajuste se usan los datos de entrenamiento y validación con un tamaño del batch de 50 y 15 épocas. Se entrenan en total 623.490 parámetros.

3.2. Modelo Generativo

3.2.1. Pre-Entrenamiento

Como en el modelo discriminativo, se define el mismo espacio de trabajo, que nuevamente será Google Colab. Importándose las mismas librerías que en el caso anterior.

3.2.2. Datos

Para la segunda parte la cual consiste en crear un modelo generativo de imágenes de rostros, se utilizó nuevamente el dataset de **CelebA**. A diferencia del caso anterior, gracias a la incorporación de ColabPro+ se pudo utilizar la totalidad del dataset, es decir, 202.599 imágenes de famosos. Las cuales fueron re-escalonadas a 64×64 pixeles, como también normalizadas.

3.2.3. Modelo

Para el segundo modelo, se utilizó un modelo generativo adversarial de redes neuronales convolucionales (**DCGAN**), como Backend **Tensorflow** y la arquitectura fue creada usando **Keras**. Se crearon dos redes neuronales que se conectan, una de ellas será el generador la que tendrá como misión generar en un inicio ruido aleatorio para que luego la segunda red, a la cual llamaremos discriminadora, clasifique entre imagenes falsas (provenientes del generador) y las imágenes reales (provenientes del dataset). Dicha red se entrena de tal manera que el generador vaya generando imágenes cada vez más realistas para confundir al discriminador y a su vez el discriminador ira aprendiendo de dichas imágenes reales para poder determinar de mejor manera si la imagen que entra es real o falsa. En los siguientes diagramas se presentan las arquitecturas de ambas redes:

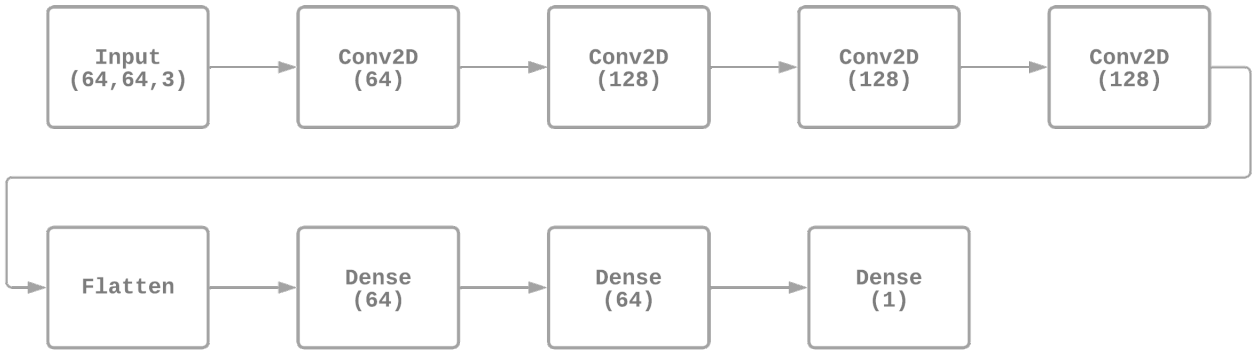

Figura 3: Arquitectura del Discriminador

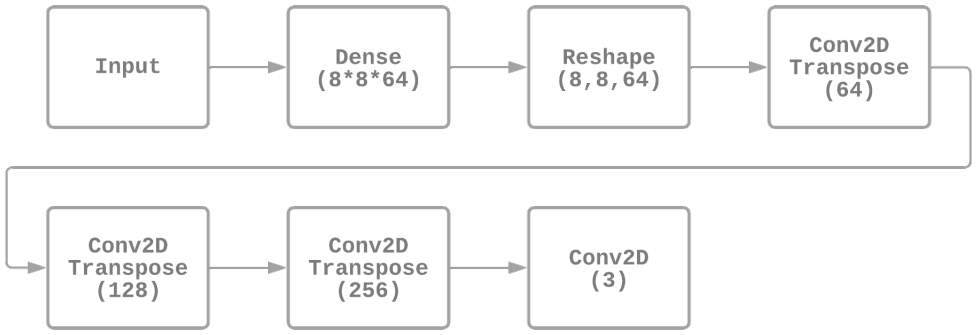

Figura 4: Arquitectura del Generador

Discriminador: Se inicia con una capa de entrada de dimensión $(64, 64, 3)$, para luego pasar por 4 capas convolucionales de 64 y 128 neuronas respectivamente. Cada una de ellas presentan un *padding* con *stride* de $(2, 2)$ y la dimension del kernel de 4, además se activan con la función **LeakyReLu** con $\alpha = 0{,}2$. Luego se pasa por una capa **Flatten** para *vectorizar* la salida de las capas convolucionales, además, se regulariza con un **Dropout** de 20 % para, posteriormente, pasar por 2 capas Densas de 64 neuronas con activación *ReLu* y una última capa densa con 1 neuronas con activación *sigmoide*.

Generador: Se define en 64 la dimensión del espacio latente (Ruido) y se inicia con una capa de entrada de dimensión igual a la del espacio latente, para luego pasar por una capa densa de $8 \times 8 \times 64$ neuronas, después, a una capa de Reshape de $(8, 8, 64)$, siguiendo con 3 capas convolucionales traspuestas de 64, 128 y 256 neuronas respectivamente cada una de ellas presenta un *padding* con *stride* de $(2, 2)$ y la dimensión del kernel de 4, además se activan con la función **LeakyReLu** con $\alpha = 0{,}2$. Finalmente se pasa por una capa convolucional de 3 neuronas, que tiene padding con stride de $(1, 1)$, dimensión del kernel de 5 y con activación *sigmoide*.

3.2.4. Entrenamiento

Para el entrenamiento, se utilizó como función de perdida **Binary Crossentropy**, como optimizador **Adam** con un learning rate de 0.0001 para ambas redes. Luego para el ajuste se utilizaron todos los datos de entrenamiento con un tamaño del batch de 32 y 30 epocas. Se entrenaron en total 1.801.028 parámetros (1.006.787 del generador y 794.241 del discriminador). El tiempo de entrenamiento fueron 6 horas.

4. Resultados y Análisis

A continuación se muestran los resultados de los modelos clasificadores y del modelo generativo junto a un análisis de estos.

4.1. Modelo clasificador

4.1.1. Modelo atributo 1: sexo

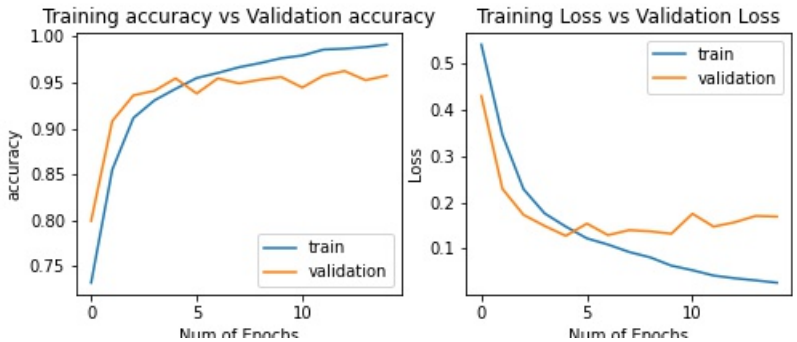

Figura 5: Gráfico de precisión y perdidad a travez de las epocas del Modelo 1

Notamos que el modelo se comporta de manera uniforme a travez de las epocas, sin mucha distancia entre el conjunto de validación y entrenamiento, por lo que se puede inferir que el modelo es óptimo.

Al evaluar el modelo en el conjunto de test se obtuvo lo siguiente:

Metrica	Resultado
Accuracy	0.946
Precision	0.946
Recall	0.946
F1 score	0.945

Se puede observar que dados los scores de las métricas, el modelo acierta en su predicción = 94,6 % de las veces (accuracy), clasifica correctamente como objeto real = 94,6 % del total de objetos reales (recall), y se equivoca = 5,4 % de las veces (precision).

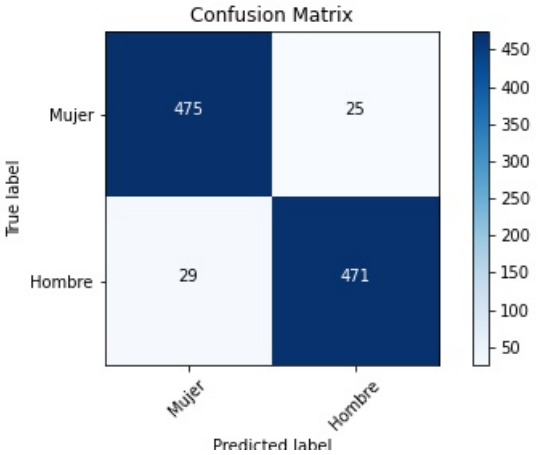

Figura 6: Matriz de confusión del Modelo 1

Se puede notar que la cantidad de falsos positivos y negativos es considerablemente menor a la de verdaderos positivos y negativos, lo cual es deseable para esta tarea de clasificación.

4.1.2. Modelo atributo 2: gafas

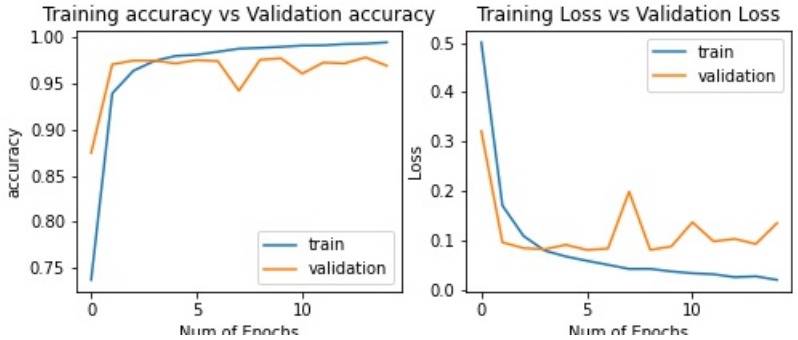

Figura 7: Gráfico de precisión y perdidad a travez de las epocas del Modelo 2

Se puede notar que el modelo es óptimo en los gráficos, y que para este atributo en particular, la pérdida para el set de entrenamiento disminuye más rápido, lo que puede indicar que hay parámetros del dataset que el modelo puede identificar de manera más fácil y rápida.

Al evaluar el modelo en el conjunto de test se obtuvo lo siguiente:

Se puede observar que dados los scores de las métricas, el modelo acierta en su predicción = 98,0 % de las veces (accuracy), clasifica correctamente como objeto real = 98,0 % del total de objetos reales (recall), y se equivoca = 2,0 % de las veces (precision).

Metrica	Resultado
Accuracy	0.98
Precision	0.98
Recall	0.98
F1 score	0.98

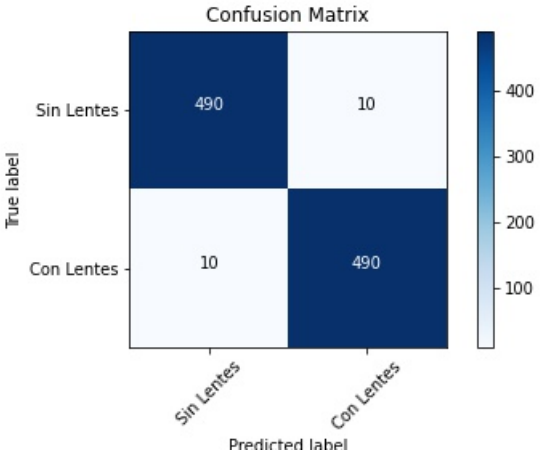

Figura 8: Matriz de confusión del Modelo 2

Se puede notar que la cantidad de falsos positivos y negativos es considerablemente menor a la de verdaderos positivos y negativos, lo cual es deseable para esta tarea de clasificación.

4.1.3. Modelo atributo 3: sonrisa

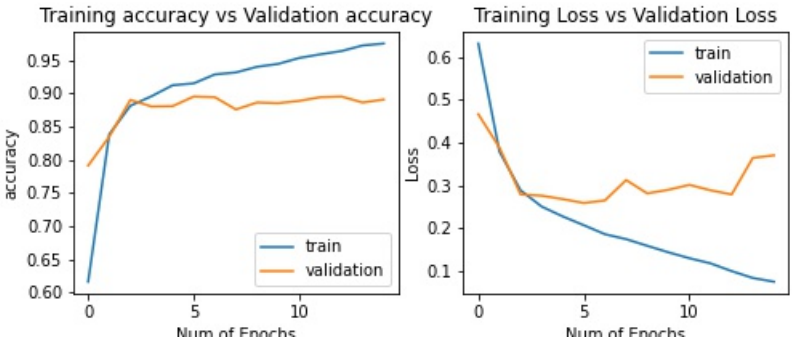

Figura 9: Gráfico de precisión y perdidad a travez de las epocas del Modelo 3

Se puede notar, al igual que en los modelos pasados que el modelo aprende de forma óptima, sin sobre ajustarse.

Al evaluar el modelo en el conjunto de test se obtiene lo siguiente:

Metrica	Resultado
Accuracy	0.883
Precision	0.884
Recall	0.883
F1 score	0.882

Se puede observar que dados los scores de las métricas, el modelo acierta en su predicción = 88,3 % de las veces (accuracy), clasifica correctamente como objeto real = 88,3 % del total de objetos reales (recall), y se equivoca = 11,6 % de las veces (precision).
Además se pueden notar peores scores en comparación a los modelos pasados, lo que puede indicar que este atributo, de ver si la persona sonríe o no, es más complejo de clasificar.

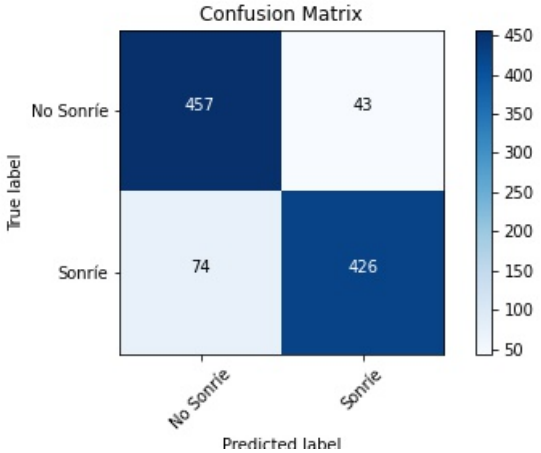

Figura 10: Matriz de confusión del Modelo 3

Se puede notar que la cantidad de falsos positivos y negativos es considerablemente menor a la de verdaderos positivos y negativos, lo cual es deseable para esta tarea de clasificación.

4.2. Modelo generativo

Se visualiza el grafico de la función de perdida a travez de las epocas

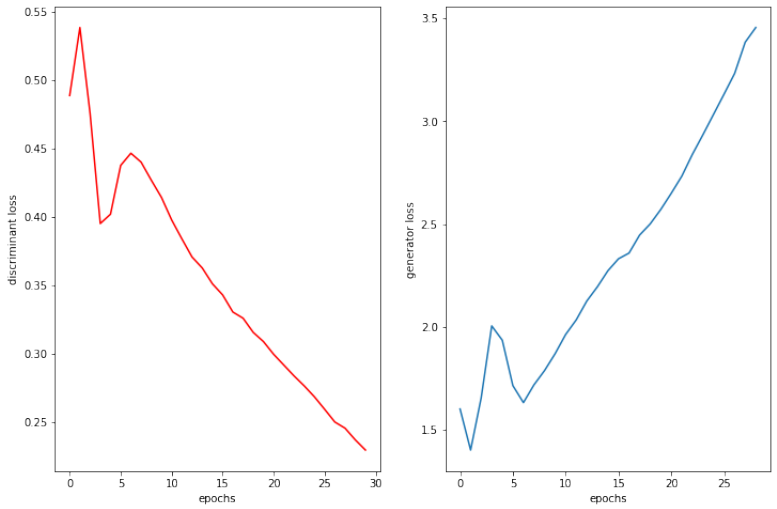

Figura 11: Pérdida del generador y discriminador

Donde podemos notar que se cumple al menos la premisa del problema de minimización-maximización de la pérdida, es decir se cumple que la perdida en el generador aumenta y en el discriminador disminuye.

Adicional a eso, durante el entrenamiento se creo un monitor, el cual iba generando imagenes a partir de ruido aleatorio en cada época. Se realiza enfasis en la primera y la última:

Figura 12: Imagenes generadas en la época 1

Figura 13: Imagenes generadas en la época 30

en las cuales se puede observar una mejora sustancial entre la epoca 1 y 30, en esta última ya se generan rostros realistas de seres humanos. Ya con el modelo calibrado, pasandole ruido aleatorio al predictor este muestra las siguientes imagenes:

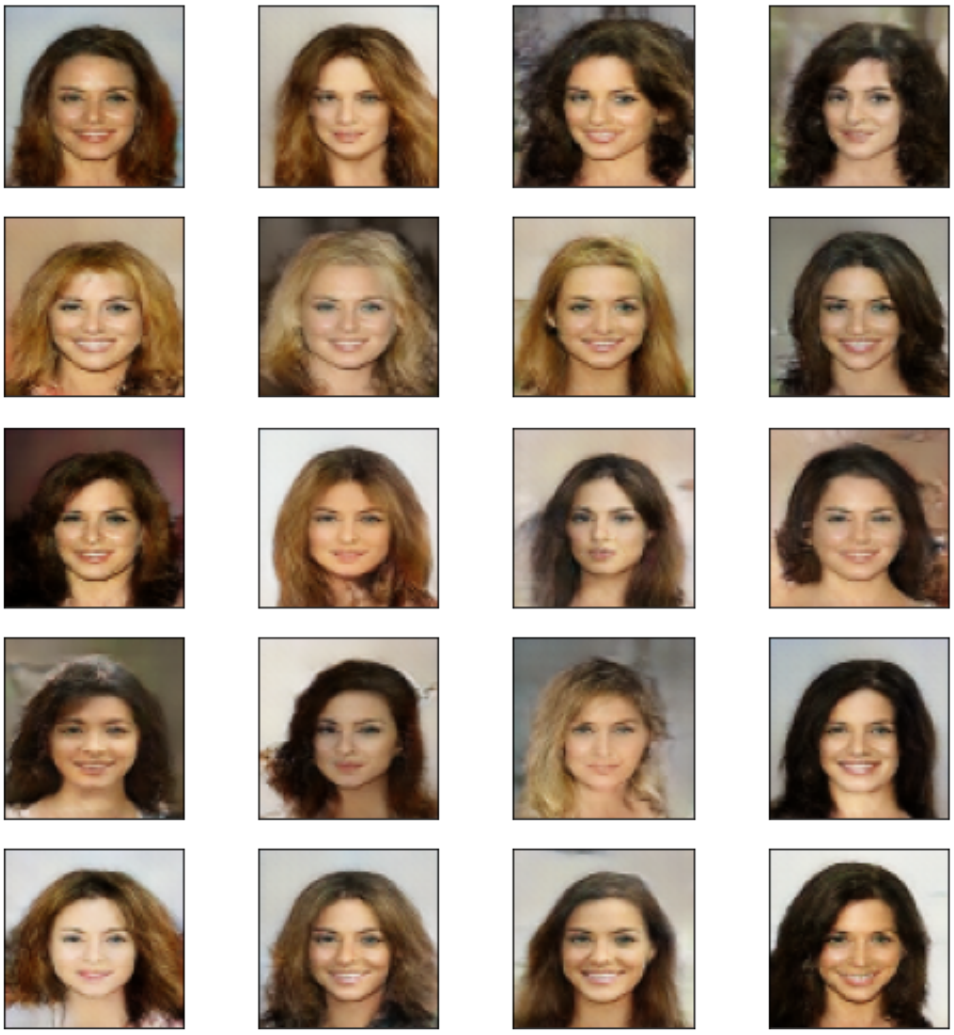

Figura 14: Imagenes generadas por el modelo calibrado

Donde se puede ver que se generan imagenes realistas pero se evidencia un colapso en la moda, ya que el modelo solo genera mujeres de pelo rubio o moreno.

4.3. Dificultades

Las dificultades cursadas varian entre los modelos trabajados. Para el primer modelo clasificador al trabajar con menor memoria y capacidad de procesamiento se limitó en gran medida la cantidad de datos utilizados y muchas veces al cargar estos datos la máquina se quedaba sin memoria RAM.
Para el modelo generativo se tuvo las problemáticas típicas de este tipo de arquitecturas, dificultades para entrenar debido a la gran cantidad de parámetros que se tienen. Dado esto, se tuvo que entrenar el modelo con mejor hardware, por lo que se utilizó la versión pro + de Google Colab.
Además, se presentó el problema del colapso de la moda, para lo cual se propone a futuro balancear el dataset entre las características de hombre o mujer, colores de cabello, entre otros.

5. Conclusiones

Se logra el objetivo de clasificar imágenes y de generar nuevos datos a través del uso de modelos los cuales han ganado popuralidad como lo son las CNNs y las GANs, y a través de la revisión bibliográfica se puede notar el impacto que estas han tenido en los campos del Machine Learning y de las Redes Neuronales.

Como trabajo futuro se puede recalcar que tanto el modelo clasificador cmomo el generador tienen bastante margen de mejora, ya sea, realizar una búsqueda de Hiper parámetros, corregir el desbalance del dataset para generar más variedad de rostros de famosos y por ende eliminar el colapso de la moda, filtrar características particulares para, mas adelante, generar rostros con dichas características aseguradas, entre otras.

6. Referencias

1. Google, Tensorflow Core. “Image classification” disponible Aquí, 2022.

2. Marcos Alvarado, Kaggle. “Image Recognition - Gender Detection” disponible Aquí, 2018.

3. Keras, DCGAN to generate face images, disponible Aquí, 2019.

4. Nagesh Singh Chauhan , Kaggle, “Generate Realistic Human Face using GAN” disponible en Aquí, 2020.