Deepfakes: creación de nuevas caras a partir de imágenes de famosos

Maria Teresa Sastre Toral
Máster Universitario en Ciberseguridad y Privacidad (MUCIP)
Análisis de datos

Enric Hernández Jiménez
Cristina Pérez Solà

01 de 06 de 2222

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Deepfakes: creación de nuevas caras a partir de imágenes de famosos*
Nombre del autor:	*Maria Teresa Sastre Toral*
Nombre del consultor/a:	*Enric Hernández Jiménez*
Nombre del PRA:	*Cristina Pérez Solà*
Fecha de entrega (mm/aaaa):	*06/2022*
Titulación:	*Máster Universitario en Ciberseguridad y Privacidad (MUCIP)*
Área del Trabajo Final:	*Análisis de datos*
Idioma del trabajo:	*Español*
Palabras clave:	*deepfakes, GAN, CNN*

Resumen del Trabajo:

Las técnicas deepfakes son usadas para la generación de vídeos, imágenes o audio manipulados. Aunque estas técnicas se basan en el engaño de hacer ver que los resultados parezcan verdaderos, no sólo el fin es la manipulación de las personas, pudiendo ser útiles como, por ejemplo, en la industria del cine.

En este trabajo se van a clasificar los distintos tipos de técnicas de deepfakes que hay, explicando la finalidad de cada una, incluyendo varios modelos que implementen dichas técnicas. Estos modelos se han acotado a los que se basan en las redes GAN para su implementación, explicando estás y las redes CNN en las que se basan.

A continuación, se ha implementado una prueba de concepto sobre la técnica deepfakes de generación de caras nuevas. El modelo elegido ha sido DCGAN porque no usa tantos recursos como otros modelos más exactos. El dataset usado en la prueba de concepto ha sido CelebA por su resolución, pero recortando las imágenes al contorno de la cara y así mejorando su procesamiento. El resultado ha mostrado que, con una red no muy profunda, se dan unas imágenes aceptables. Esto se ha evaluado de forma cualitativa, ya que la función de perdida no asegura la convergencia del modelo, para ello sería necesario usar otro tipo de métricas.

Para finalizar, una posible continuación de este trabajo sería buscar las técnicas de detección para estos modelos de deepfakes y realizar alguna prueba con las imágenes generadas que compruebe si detecta falsificaciones.

Abstract:

Deepfakes techniques are used to generate manipulated videos, images or audio. Although these techniques are based on the deception of pretending that the results seem true, but the purpose is not only the manipulation of people, and they can be useful, for example, in the film industry.
In this work, the different types of deepfake techniques are going to be classified, explaining the purpose of each one, including several models that implement these techniques. These models have been limited to those based on GAN networks for their implementation, explaining these and the CNN networks on which they are based.
Next, a proof of concept has been implemented on the deepfakes technique of generating new faces. The chosen model has been DCGAN because it does not use as many resources as other more exact models. The dataset used in the proof of concept has been CelebA for its resolution but cropping the images to the contour of the face and thus improving its processing. The result has shown that, with a not very deep network, acceptable images are given. This has been evaluated qualitatively, since the loss function does not ensure the convergence of the model, for this it would be necessary to use other types of metrics.
Finally, as a possible continuation of this work, the detection techniques for these deepfake models can be searched and some test can be carried out with generated images to check if they detect fakes.

Índice

Índice de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

En los últimos años hemos visto como cada vez más se pueden manipular imágenes, vídeos y audios que parecen reales o pertenecen a personas que realmente no lo son. Todas estas innovaciones nos llevan a no confiar en lo que podemos observar a simple vista. Los motivos de las modificaciones pueden ser varios, desde hacer creer a una persona o sociedad algo que realmente no es cierto y manipular su pensamiento hasta desacreditar a alguien pasando imágenes suyas haciendo o diciendo algo que realmente no ha realizado.

Todas estas acciones pueden conllevar a que la sociedad dude siempre de lo que está viendo y no confíe en las supuestas personas que aparecen en dichas imágenes.

Si salimos, además, del ámbito de las personas, también se puede hacer pasar por falsas imágenes que no están relacionadas con caras. Por ejemplo, se puede intentar hacer que un ordenador de a bordo no detecte correctamente las señales de carretera aplicándoles alguna red adversaria que haga pasar por buenas imágenes que no lo son [1], se pueden falsificar archivos como los códigos QR que puedan pasar como válidos, etc.

En esta última década, las técnicas de creación de imágenes impostadas se han mejorado bastante, en parte gracias a las redes neuronales que se aplican a este proceso. No obstante, aunque haya formas de detectar cuando estamos ante una deepfake [4], cada vez resulta más complicado descubrir una falsificación.

El trabajo que se va a llevar a cabo tratará de ubicar que tipos de deepfakes existen hoy en día y que técnicas se están utilizando para llevarlas a cabo. De entre ellas, se elegirá una para implementar un caso de uso (prueba de concepto) que demuestre como realmente se aplica una deepfake asociándola a redes neuronales. Para ello, se va a hacer uso de las redes antagónicas que ayudarán al proceso de modificación de imágenes haciéndolas pasar por reales. Este trabajo no ahonda en la necesidad de detección de dichas deepfakes aunque sería un punto de partida para poder intentar detectarlas de forma automatizada.

1.2. Objetivos del Trabajo

El objetivo principal de este trabajo fin de máster es poder generar caras nuevas artificiales a través de imágenes de caras reales. Estos nuevos perfiles pueden ser usados, por ejemplo, en la creación de cuentas fakes de Facebook o Instagram para simulación de followers.

Como objetivos secundarios, se pueden plantear varios:

- El estudio de las diferentes deepfakes que existen actualmente con sus técnicas de solución viendo la que se puede llevar a cabo en la posterior prueba de concepto.
- Estudio de cómo funcionan las redes neuronales convolucionales y las redes antagónica para poder llevar a cabo el ejemplo aplicado.
- El estudio de un dataset que pueda ser proporcionado como entrada a la red neuronal que pueda generar las nuevas caras.
- Tratamiento de las imágenes del dataset elegido para que tengan todas las mismas dimensiones a la hora de cargar el conjunto de entrenamiento (train).
- Estudio de las librerías de redes neuronales para poder llevar a cabo la prueba de concepto, en concreto, se estudiarán las librerías de Python Keras y TensorFlow.
- Creación de la prueba de concepto haciendo uso de los conocimientos previos de estudio y partiendo del dataset elegido. Para ello, se aplicarán uno o varios modelos de redes neuronales sobre el anterior dataset para ver cuál de ellos puede alcanzar mejores resultados.

1.3. Enfoque y método seguido

Como ya se ha comentado, las deepfakes son una forma de generación de imágenes falsas que se usan para modificar aspectos del vídeo, audio o imagen para que parezcan de otra persona que no es la originaria. Existen numerosos tipos de deepfakes y cada uno de ellos puede tener distintas técnicas de actuación para llevar a cabo el engaño sobre los ficheros originales. Para llevar a cabo el proceso de generación de las nuevas imágenes se hace uso de redes neuronales profundas. En esta área, se está avanzando de forma muy rápida en los últimos años, haciendo que las imágenes falsas sean cada vez más difíciles de descubrir. Con este proyecto, no se va a desarrollar algo novedoso con respecto a la generación de imágenes falsas, sino que se va a investigar lo que tenemos hoy en día poniendo en práctica alguna de los métodos de generación de deepfake para poder ahondar en como montar la red convolucional y la antagónica para su elaboración. No sólo se verán las redes usadas mostrando el diseño de red, sino que se explicarán las capas y parámetros que las formas pudiendo hacer ciertas modificaciones que permitan mejorar la optimización de su salida. Esto llevará a tener un mejor entendimiento de una de las técnicas usadas para la generación de deepfakes.

La estrategia a seguir se basa en el estudio de las diversas técnicas de deepfake y como llevarlas a cabo con distintas tecnologías asociadas a las redes profundas. A continuación, se elegirá una de las estrategias para llevar

a cabo una prueba de concepto. El siguiente paso será buscar un dataset que pueda tener un buen conjunto de imágenes para que los resultados puedan ser bastante reales. Se tendrá que evaluar si el dataset elegido necesitara de un tratamiento de las imágenes que lo componen para su carga en el modelo elegido. Para finalizar, se aplicará el dataset sobre el modelo y se analizaran los resultados obtenidos pudiéndose modificar dicho modelo para optimizar la salida del modelo.

Los anteriores pasos se pueden relacionar con la metodología CRISP-DM [2] ya que incluye el entendimiento de lo que hay realizar (comprensión del negocio), estudio y comprensión de los datos, preparación de los datos, modelado de las redes necesarias, comprobación de los resultados de aplicación del modelo y puesta en marcha.

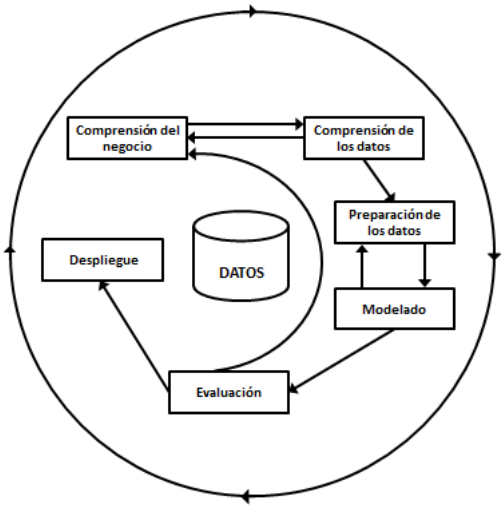


Figura 1: Metodología CRISP-DM

Se puede decir que la metodología CRIP-DM es la más usada dentro del análisis asociado al machine learning, y por tanto al deep learning también. Como se ha visto anteriormente, es la metodología que más se acopla a las fases de este trabajo.

1.4. Planificación del Trabajo

La planificación del proyecto se va a desarrollar entorno a los hitos genéricos definidos en el TFM. Dentro de estos hitos se van a generar tareas específicas para llevar a cabo cada entrega del proyecto como: el estudio y/o procesamiento del dataset elegido para poder entrenar el modelo, el conocimiento de las redes convolucionales y las redes antagónicas, el conocimiento de las librerías que implementan dichos modelos en Python como son: Keras y TensorFlow y el estudio de los modelos propuestos para resolver distintas

técnicas que generaran el deepfake (dentro de estado del arte) entre otras cosas. Con respecto a los recursos necesarios para llevar a cabo este TFM se necesitará de un ordenador con capacidad de procesamiento. Se va a utilizar la herramienta Colab de Google ya que esta tiene RAM de 32 Gb y puede hacer uso de una GPU que ellos proporcionan. Como herramientas de desarrollo se va a necesitar el compilador de Python con librerías como Keras y Tensorflow, necesarias para la programación de las redes neuronales, además de otras que permitan visualizar los resultados. Se hará uso del Jupyter Notebook para ejecutar los scripts programados para llevar a cabo el entrenamiento de los modelos.

Para tener claro que se prioriza en cada fase del proyecto se ha valorado mostrar las etapas del proyecto haciendo uso de un diagrama de Gantt, que muestra de forma sencilla los hitos que hay que seguir, junto con su duración y situación en el tiempo, incluyendo en estos las tareas también con su temporización.

Como se puede observar en la siguiente figura se ha podido generar un diagrama de Gantt con la herramienta libre projectLibre [3]. Se han desarrollo todas las tareas agrupadas por bloques y asociadas a un hito final que da paso a la siguiente fase. En concreto se van a detallar 4 grandes fases que son: la definición y planificación del proyecto, el estudio del estado de la cuestión para ver que se está desarrollando en esta área de trabajo, el estudio del dataset para la carga de datos y el diseño del modelo a implementar, su implementación y tratamiento con pruebas que validen la exactitud del modelo, y para terminal la extracción de los resultados, sacando conclusiones al respecto, todo junto englobado en la redacción de esta memoria. Además, dado que este es un TFM se ha añadido la preparación para la defensa del proyecto con la documentación necesaria para llevarla a cabo.

	Nombre	Duracion	Inicio	Terminado	Pred...	Nombres del Recurso
1	**TFM: Deepfakes: creación de nuevas caras a partir de imágenes de famosos**	**80 days?**	**16/02/22 8:00**	**7/06/22 17:00**		
2	**1. Definición y planificación del proyecto**	**10 days?**	**16/02/22 8:00**	**1/03/22 17:00**		
3	1.1. Contexto y justificación del trabajo	2 days?	16/02/22 8:00	17/02/22 17:00		
4	1.2. Objetivos del proyecto	2 days?	18/02/22 8:00	21/02/22 17:00	3	
5	1.3. Enfoque y método seguido	2 days?	22/02/22 8:00	23/02/22 17:00	4	
6	1.4. Tareas y planificación el trabajo	2 days?	24/02/22 8:00	25/02/22 17:00	5	
7	1.5. Entrega	1 day?	1/03/22 8:00	1/03/22 17:00	6	
8	**2. Estado del arte**	**20 days?**	**2/03/22 8:00**	**29/03/22 17...**		
9	2.1. Busqueda de bibliografía asociada	9,625 da...	2/03/22 11:00	15/03/22 17:00		
10	2.2. Estudio de las distintas técnicas de deepfakes	10 days?	2/03/22 8:00	15/03/22 17:00		
11	2.3. Estudio de los modelos que se aplican en las distintas técnicas de deepfakes	10 days?	2/03/22 8:00	15/03/22 17:00		
12	2.4. Estudio de como funcionan las CNN	3 days?	16/03/22 8:00	18/03/22 17:00	11;...	
13	2.5. Estudio de como funcionan las GAN	3 days?	21/03/22 8:00	23/03/22 17:00	12	
14	2.6. Redacción de la memoria del estado del arte	3 days?	24/03/22 8:00	28/03/22 17:00	13	
15	2.7. Entrega	1 day?	29/03/22 8:00	29/03/22 17:00	14	
16	**3. Realización de una prueba de concepto**	**20 days?**	**30/03/22 7:00**	**26/04/22 17...**		
17	3.1. Elección del deepfake elegido	1 day?	30/03/22 7:00	30/03/22 17:00		
18	3.2. Estudio del dataset para aplicarlo en el modelo del deepfake elegido	2 days?	31/03/22 8:00	1/04/22 17:00	17	
19	3.3. Estudio del modelo del deepfake, su arquitectura tanto generadora como discriminador	2 days?	4/04/22 8:00	5/04/22 17:00	18	
20	3.4. Estudio de las librerías Python: Keras y Tensorflow	3 days?	6/04/22 8:00	8/04/22 17:00	19	
21	3.5. Implantación del modelo	6 days?	11/04/22 8:00	18/04/22 17:00	20	
22	3.6. Análisis y pruebas sobre el modelo	3 days?	19/04/22 8:00	21/04/22 17:00	21	
23	3.7. Posibles cambios del modelo para mejorar los resultados	2 days?	22/04/22 8:00	25/04/22 17:00	22	
24	3.8. Entrega	1 day?	26/04/22 8:00	26/04/22 17:00	23	
25	**4. Documentación**	**25 days?**	**27/04/22 7:00**	**31/05/22 17...**		
26	4.1. Finalización de la memoria	22 days?	27/04/22 7:00	26/05/22 17:00		
27	4.2. Condusiones	2 days?	27/05/22 8:00	30/05/22 17:00	26	
28	4.3. Entrega	1 day?	31/05/22 8:00	31/05/22 17:00	27	
29	**5. Presentación**	**5 days?**	**1/06/22 7:00**	**7/06/22 17:00**		
30	5.1. Diapositivas	4 days?	1/06/22 7:00	6/06/22 17:00		
31	5.2. Video	2 days?	3/06/22 7:00	6/06/22 17:00		
32	5.3. Entrega	1 day?	7/06/22 8:00	7/06/22 17:00	31;30	

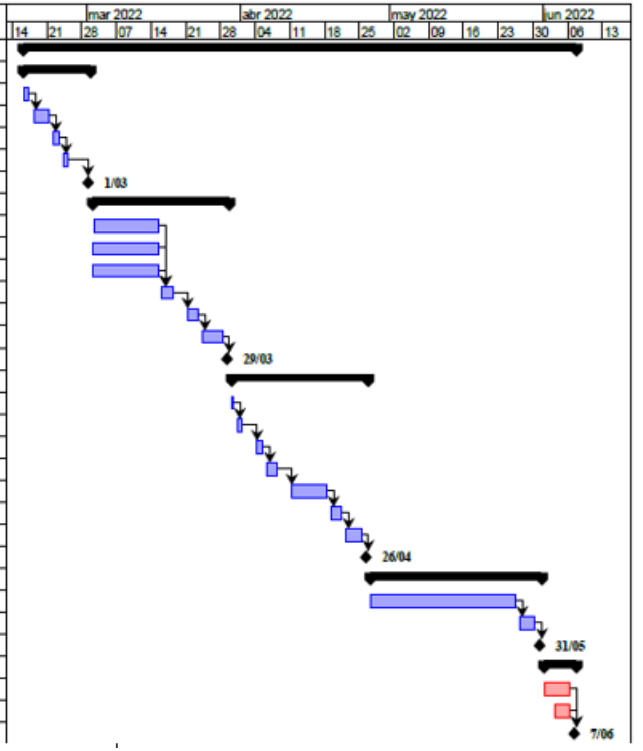


Figura 2: Planificación inicial del TFM mediante un diagrama de Gantt

1.5. Breve sumario de productos obtenidos

El producto final será la prueba de concepto, en concreto será implementar y comprobar el modelo de tipo GAN elegido para realizar la deepfake escogida. Este modelo podrá ser pulido para mejorar la salida de la red y en concreto la imagen falsificada será más difícil de detectar.

1.6. Breve descripción de los otros capítulos de la memoria

En las siguientes partes de esta memoria podemos encontrar:

- El estudio de las distintas técnicas basadas en el aprendizaje profundo para la generación de deepfakes con los modelos que las implementen.
- La profundización en los modelos usados para la generación de una prueba de concepto, en concreto se va a hablar de las de las CNN y de las GAN.
- El desarrollo de la prueba de concepto. Para ello, se elegirá la técnica deepfake a usar, se describirá el dataset elegido y por qué y se explicaran los modelos concretos usados para la resolución de este ejemplo, tanto el modelo generador como el discriminador. Se pasará a realizar pruebas de entrenamiento, a través de la programación del modelo con las librerías Keras y Python, con el dataset elegido y se evaluaran los resultados mostrados. Estos resultados en función de su exactitud pueden ser mejorados o no mediante los parámetros de las redes que forman el modelo. También se pueden mejorar modificando las capas de la GAN.

2. Estado del Arte

Podemos describir un deepfake como la manipulación de un vídeo, imagen o audio para que parezca real sin serlo, haciendo uso de la inteligencia artificial. Las dos palabras usadas vienen de fake (falso) y deep learning, un área de la inteligencia artificial que usa las redes neuronales para simular el comportamiento de un cerebro humano. El hecho de que los nuevos vídeos, imágenes o audio sean generados a través de redes neuronales, hace más difícil la detección de posibles anomalías que pudieran corroborar su falsedad. Un ejemplo de deepfake fue el vídeo del expresidente de los Estados Unidos, Barack Obama, realizado en 2018 donde se mezclaba las imágenes y audio de un actor, pero con la voz y cara del expresidente. El uso de estas técnicas de engaño puede llevar a la población a creerlo en un principio, pero tras sufrir las consecuencias (posible fraude por creer vídeos de personas con buena reputación que han sido manipulados, chantajes por mostrar vídeos o audios que la persona no ha realizado, desacreditación de la persona, etc.), y con un mayor conocimiento de que existen estas técnicas ya no se sabrá si lo que vemos en cualquier medio es cierto o no, provocando una falta de confianza a un nivel mundial, algo realmente grave para la sociedad.

Dentro de los tipos de deepfakes nos podemos encontrar los que aparecen en la siguiente imagen.

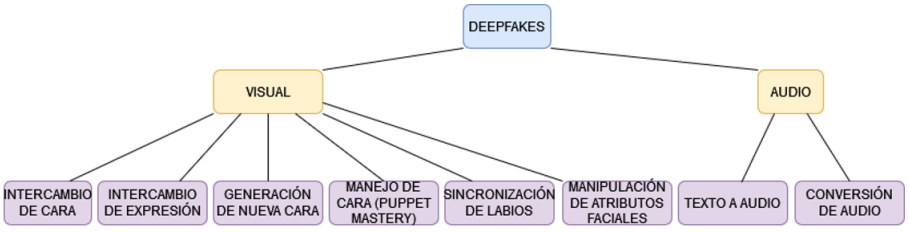


Figura 3: Tipos de deepfakes [4][5]

De los anteriores tipos definidos en el estado del arte descrito por Masood et at. [4] nos podemos encontrar con dos clasificaciones, las asociadas a la parte visual como: el intercambio de caras, la creación de nuevas caras o la manipulación de algún atributo los cuales ya se han descrito en artículos previos [5], y la parte del audio. A continuación, vamos a realizar una explicación de cada tipo (se desarrollarán más detalladamente los tipos que posteriormente se van a llevar a cabo en la prueba de concepto) incluyendo las técnicas para generar dichos resultados.

2.1. Intercambio de caras

Con este tipo de deepfake se usa una foto o vídeo origen que se quiere modificar y un objetivo que sería la cara de otra persona incrustada en ese imagen o vídeo [6]. Para llevar a cabo el intercambio, primero se tiene que detectar la cara de la imagen o vídeo objetivo, se observará su posición y se intercambiara por otra cara que se acople a la original debido a su similitud. La fusión se hará intercambiando los ojos, la nariz y la boca. También se ajustará el color de la cara y se hará uso de la iluminación para mejorar el acoplamiento [7]. Finalmente, se superponen las dos caras realizando un cálculo de sus coincidencia que dará como resultado la imagen superpuesta. Al ser un remplazo se puede perder las expresiones de la cara de origen quedando la imagen reemplazada muy artificial.

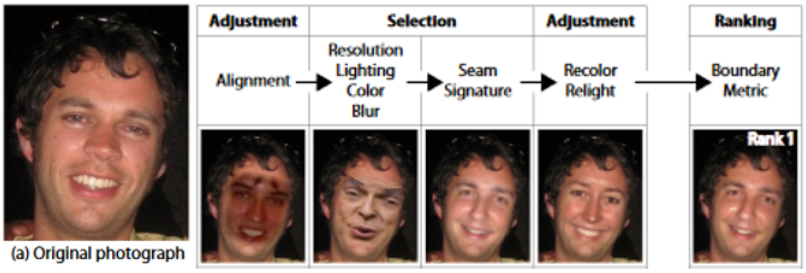


Figura 4: Ajustes de distintas caras para ver la que más se acoplaría a la original [7]

Con la llegada de los algoritmos de deepface se han podido mostrar resultados más realistas como el uso de alguna aplicaciones como: FaceSwap (usa autoencoder) [8], FaceSwap-GAN (añade las redes GAN [12] a los autoencoders) [9], etc. .Estos algoritmos hacen que la expresión facial del vídeo original quede intacta con el reemplazo de la otra cara .

Una de las técnicas que se usa para poder realizar estas acciones es la de autocodificación (codificación-decodificación). El codificador extrae las características del rostro de origen y el decodificador lo que hace es reconstruir el rostro. Para realizar el intercambio de caras es necesario tener dos codificadores y dos decodificadores. Cada codificador se entrena con cada cara origen y destino. Una vez completado el entrenamiento, la imagen real se codifica con el codificador de origen pero se decodifica con la imagen a suplantar para implementar en ella las características de la originaria. Con esta técnica permanecen las expresiones faciales de la imagen de origen.

Con la técnica del autoencoder no se hace un intercambio de rostros tal cual, sino que se tienen a extraer mediante la técnica de CNN las características del rostro para plasmarlas a la imagen que queremos cambiar. Para ayudar a este proceso se ha hecho uso de una función de perdida sobre dichas características extraídas para que el resultado pueda ser más realista

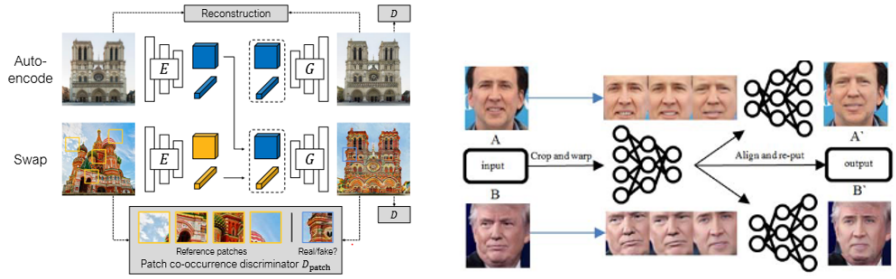


Figura 5: Explicación de la técnica de Autoencoder con una imagen de un edificio y un intercambio de caras [10][11]

[4] aunque requiera de un volumen de entrenamiento mayor para conseguirlo. Como handicap tiene que el modelo entrenado sólo se puede usar para transforma una sola imagen.

Otra técnica que se usa para el intercambio de caras es la que añade GAN [12] para dicho intercambio, como por ejemplo las FaceSwap-GAN [9]. Las características que se aplican son:

- La perdida de percepción con el modelo VGGFace que hace que mejore la dirección de los glóbulos oculares, además de suavizar los elementos del filtro (mascara) de segmentación. Esto hace que haya mayor calidad en la salida.
- Resolución de entrada/salida configurable (64x64,128x128 y 256x256).
- Predicción de una mascara de "atención" que permite mejorar entre otras cosas el tono de piel, manejar la oclusión dental o la eliminación de elementos que hay en el origen pero no el destino (gafas, pendientes,etc).
- Alineaciones del rostro más exactas con MTCNN y eliminación de la fluctuación de la cara intercambiada, además de los bordes de los marcos con el filtro de Kalman.
- Perdida de los bordes del área del ojo y su reconstrucción los hacen más realistas.

El enfoque basado en GAN [12] mejora el autoencoder ya que puede funcionar sin ser entrenado explicitamente para una imagen. Se usan también en la técnica de generación de caras nuevas como se verá en el siguiente apartado. A continuación, se muestra una tabla con las principales técnicas aplicadas para el intercambio de caras.

Modelo	Técnica	Limitaciones
Faceswap[8]	Cod. – Dec.	Requiere muchas imágenes objetivo (target), los resultados puede ser borrosos debido a perdida en la compresión. Fallos en la dirección de la mirada, el peinado, la iluminación, la expresión facial y la postura.
FaceSwapGAN[13]	GAN	Los detalles no se aprecian y los resultados no son muy claros.
DeepFaceLab[14]	Cod. – Dec.	Requiere bastantes datos de entrenamiento. No logra mezclar tonos faciales muy diferentes.
Fast Face-swap [15]	CNN	Funciona para una sola persona dando mejores resultados con la vista frontal de la cara. Fallan en los detalles de la textura de la piel. Las transferencias de los resultados son muy tenues. No se aprecian detalles como las gafas.
Nirkin et al.[16]	FCN-8sVGG	Si las resoluciones son diferentes se generan malos resultados. No se logran mezclar los tonos faciales diferentes.
Chen et al.[17]	VGG16	Los resultados son más realistas pero fallan en la postura y la mirada.
FSGAN[18]	GAN	Mala gestión de la postura (ángulo) degradándose la identidad de la imagen y la calidad de esta. No hay punto de referencia, lo que empeora la captación de las expresiones faciales.
FSNet[19]	GAN	Sensible a la variación del ángulo.
RSGAN[20]	GAN	Sensible a la variación de ángulo, la oclusión dental y la iluminación. La resolución de salida es limitada.
FaceShifter[21]	GAN	No aparecen detalles como gafas o pendientes.

De las anteriores técnicas (aparte de las ya comentadas) caben destacar alguna como:

FSGAN [18]. Esta técnica es independientemente del sujeto. Se puede

aplicar a pares de rostros sin necesidad de entrenamiento previo. Esto se realiza a partir del enfoque de una RNN que hará posible la recreación facial ajustada a la postura y expresión. Se puede aplicar esta técnica tanto a la imagen como al vídeo. Las regiones de caras no visibles son recreadas por una red de terminación de caras. Finalmente, se usas otra red de combinación de caras que conservarán el color de la piel y las condiciones de iluminación.

RSGAN [20]. Esta técnica genera rostros a partir del intercambio de estos usando sus atributos y la síntesis aleatoria de partes del rostro. Se usa una red CNN que detecta las regiones de la cara y el cabello a una gran escala de forma independiente. El intercambio de caras se logra reemplazando las representaciones del espacio latente de las caras y reconstruyendo la imagen completa de la cara con ellas mejorando anteriores técnicas donde el ajuste no es adecuado. La independencia del cabello con la cara permite, además, combinaciones de caras con distinto pelo.

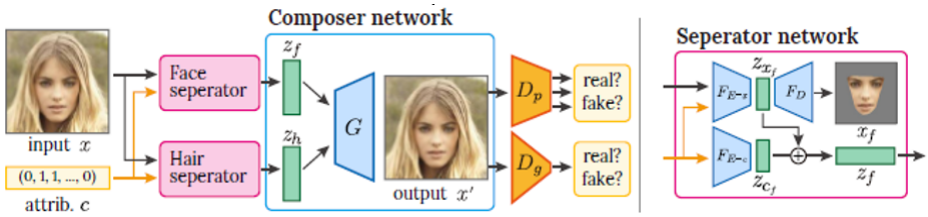


Figura 6: Arquitectura RSGAN [20]

Las arquitectura RSGAN usa tres redes: dos separadoras y una compositora. Las redes separadoras extraen representaciones del espacio latente de las regiones de la cara (Zf) y el pelo (Zh) de forma separada a partir de una imagen X. La red compositora reconstruye la imagen de la cara a partir de las representaciones de espacio latente de cada parte de la imagen. La imagen reconstruida X' y la imagen de entrada X son evaluadas por dos redes discriminadoras. Una de ella distingue si la imagen es real o falsa y la otra distingue si las partes son reales o falsas.

2.2. Manipulación de atributos faciales

A diferencia de la anterior técnica, en esta solo se quieren intercambiar ciertas partes del rostro manteniendo el resto intacto. No solo se usa para modificar un atributo del rostro, sino que se pueden cambiar cosas como: la piel (que sea más oscura o clara), la eliminación de arrugas o cicatrices, poner o quitar gafas, etc. Incluso se podrían modificar características más genéricas como puede ser el sexo o la edad. Aplicaciones como FaceApp [22] se basan en esta técnica. La siguiente tabla muestran distintos enfoques, casi todos basados en GAN y las limitaciones que se encuentran en cada uno:

Técnica	Limitaciones
IcGAN[23]	No conserva la cara de la identidad original
Encoder-decoder[24]	La imagen aparece distorsionada y borrosa no conservando los detalles.
StarGAN[25]	La piel queda con imperfecciones.
AttGAN[26]	Genera resultados de baja calidad y agrega cambios no deseados incluido borrosidad.
STGAN[27]	Malos resultados con múltiples atributos.
SAGAN[28]	No se muestran detalles en las demás regiones.
PA-GAN[29]	Atributos no deseados: calva, boca abierta, etc.

Un resumen de alguna de las técnicas enumeradas anteriormente sería el siguiente:

IcGAN[23] La técnica invertible condicional GAN se usa con la combinación de la técnica cGAN[48] para trabajar cambiando atributos faciales. Como se ve en la siguiente figura hay dos codificadores, el Z que dada una imagen X codificar su representación latente Z y el codificador Y que asigna la imagen X a un vector de información condicional Y.

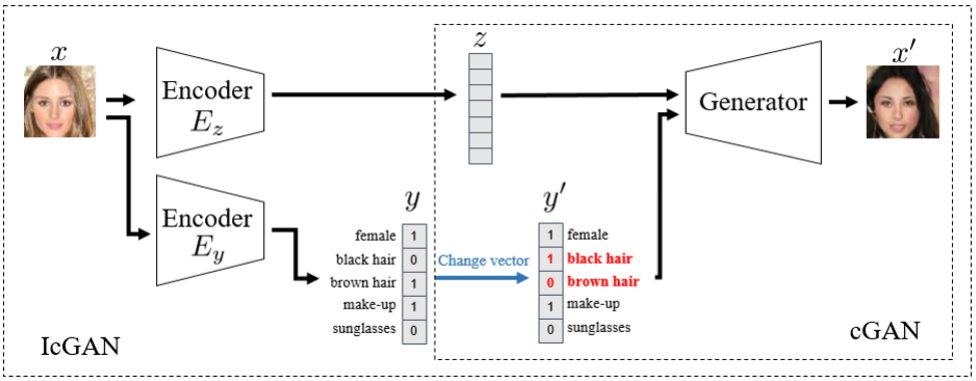


Figura 7: Arquitectura IcGAN [23]

A continuación, el generador cGAN reconstruye la imagen de la cara con nuevos atributos dado el vector de atributos alterado como condición. Esta técnica sufre de pérdida de información y puede alterar la identidad de rostro original. Como desventaja esta técnica se centra en el manejo de traducciones de imagen a imagen entre dos dominios por lo que limitan su uso práctico.

StarGAN[25]. Es una técnica capaz de traducir imágenes de múltiples dominios con un solo generador. La red de transferencia de atributos es entrenada con pérdida de clasificación de atributos y perdida de consistencia en el ciclo. StarGAN mejora a anteriores técnicas en la manipulación de atributos y expresiones, pero generar efectos en la piel. LA StarGAN-2 aumentará la calidad de salida al agregar un vector de ruido gaussiano aleatorio en el

generador.

AttGAN[26]. Esta técnica, en lugar de imponer una restricción independientemente de los atributos sobre la representación latente como pasa con IcGAN, se aplica una restricción de clasificación de atributos a la imagen generada para garantizar el correcto cambio de los atributos elegidos (relación entre codicador-decodicador y representación latente). Esto mejora los resultados en la edición de atributos faciales, pero por contra, estos son generados a baja calidad.

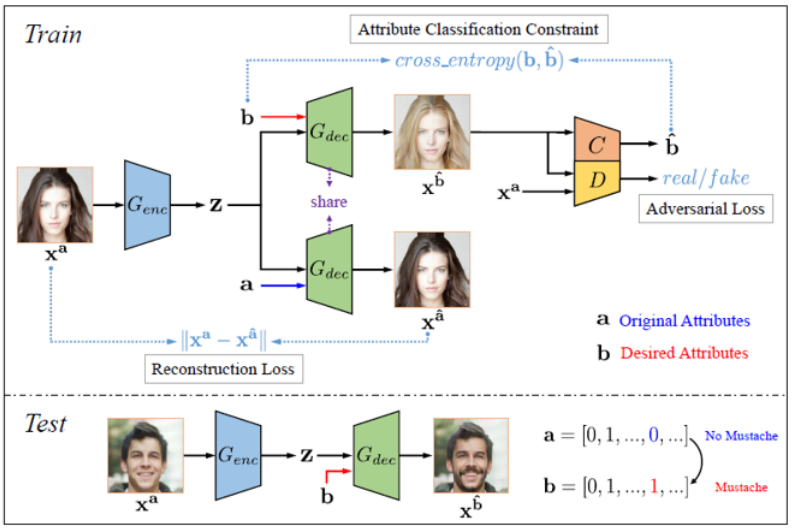


Figura 8: Arquitectura AttGAN [26]

Como se puede ver en la anterior figura el modelo AttGAN está formado por tres componentes: una restricción de clasificación de atributos, un aprendizaje de la reconstrucción y el aprendizaje adversario. El primero garantiza la correcta manipulación de los atributos de la imagen generada, el segundo tiene como objetivo preservar los detalles excluyentes de los atributos y el tercero se encarga de una generación de imágenes visualmente realistas.

STGAN[27]. Esta técnica hace uso de las transferencias selectivas. Teniendo en cuenta que la tarea de edición específica ciertamente solo está relacionada con los atributos que se quieren modificar (en lugar de con todos los atributos de destino), este modelo toma selectivamente la diferencia entre los vectores de atributos origen y destino para su entrada. Las unidades de transferencia selectivas (STU) se incorporarán al codificador-decodificador para seleccionar y modificar de forma adaptativa al codificador y así mejorar la edición de atributos. Esto mejoró en la precisión de la manipulación de los atributos y su percepción en la imagen.

PA-GAN[29]. Esta técnica usa la "atención progresiva" en la edición de los atributos faciales. Con este enfoque, la edición se lleva a cabo progresi-

vamente desde el nivel de característica más alto hasta el más bajo mientras se restringe en un área el atributo haciendo uso de una máscara de atención por cada nivel. De esta manera, se evitan modificaciones no deseadas en las regiones irrelevantes desde el principio. A medida que el nivel de funciones es más bajo (mayor resolución), la máscara de atención y la edición de atributos se vuelven más precisas. De esta manera, se evitan modificaciones no deseadas en las regiones irrelevantes desde el principio.

2.3. Generación de nuevas caras

Otra de las falsificaciones muy usadas actualmente es la de creación de imágenes nuevas bastante realistas que no corresponden a una persona verdadera. Esta generación de nuevas imágenes puede servir para crear perfiles falsos pudiéndose estos asociar a redes sociales. Por ejemplo, se ha visto no hace tanto tiempo que hay personas que compran followers de redes sociales, siendo estos perfiles falsos. Un aspecto positivo podría ser usar estas caras para la industria del videojuego o como avatar de interacción de aplicaciones de empresas con humanos. La imagen mostrada pudiera ser generada con este tipo de técnica. Las GAN (que hace uso de un generador y discriminador) [12] o las VAE [30] son usadas para generar estas imágenes. Con la técnica DCGAN [31] que se creo a continuación de las técnicas anteriores, se obtuvo mayor precisión en los resultados.

A continuación, se explicarán de forma resumida las técnicas que se han ido usando durante estos años para llevar a cabo este tipo de imágenes[39]:

GAN[12]. Nos encontramos con la primer red antagónica que se uso para la solución de creación de caras nuevas. GAN fue usada en el 2014 pero las imágenes que se obtenían no tenían una gran calidad. Se usaban tanto un modelo generativo como otro discriminante. El primero genera imágenes no reales y el discriminante detectaría cual es real o no mejorando la técnica del modelo generativo hasta llegar a un momento en que la imagen generada no sea detectada por la discriminante, pudiendo realizar caras con cierta precisión.

DCGAN[31]. Como ya se dijo anteriormente, mejora las anteriores técnicas, ya que ha sido una de las primeras que ha introducido una capa de stride en lugar de una padding, añadiendo batchNormalization tanto al generador como al discriminado y eliminando las capas totalmente conectadas por un número mayor de capas convoluciones en el generador. Además, usa el descenso estocástico del gradiente con minilotes de 128 para su entrenamiento con el optimizador Adam tanto para el generador como para el discriminador. Esto hizo que se pudiera obtener mayor precisión en los resultados que con las simples GAN.

CoGAN (Basada en VAE)[32]. Red antagónica generativa acoplada. Con este técnica se usa un par de redes generador-discriminador en vez una para que se entrene un dominio.

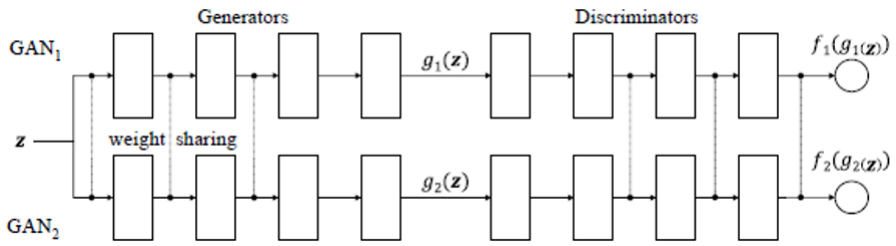


Figura 9: gráfica que muestra las dos redes del modelo CoGAN[32]

CoGAN consta de un par de GAN: GAN1 y GAN2. Cada una tiene un modelo generativo para sintetizar imágenes realistas en un dominio y un modelo discriminativo para clasificar si una imagen es real o generada. Los pesos en las primeras etapas (decodificadoras a alto nivel) de las redes generadoras y los pesos de la últimas (codificadoras a alto nivel) de las discriminadoras se comparten. Esto permite generar una distribución de imágenes conjunta sin supervisar su relación. Estas redes entrenaran pares de imágenes que comparten la misma abstracción a alto nivel pero que se serán diferentes a bajo nivel. Se aplica CoGAN a varias tareas de aprendizaje de distribución conjunta, incluido el aprendizaje de una distribución conjunta de imágenes con respecto al color y la profundidad, o con diferentes atributos faciales. En otras técnicas se usarían tuplas de imágenes correspondientes para alcanzar el existo en la distribución conjunta, sin embargo, con CoGAN esto no sería necesario.

Los resultados son imágenes bastante convincentes en caras nuevas. También es interesante en la transformación de imágenes, por ejemplo, ante la imagen de un paisaje en una determinada estación de año y que este se pueda cambiar a otra estación (verano-invierno).

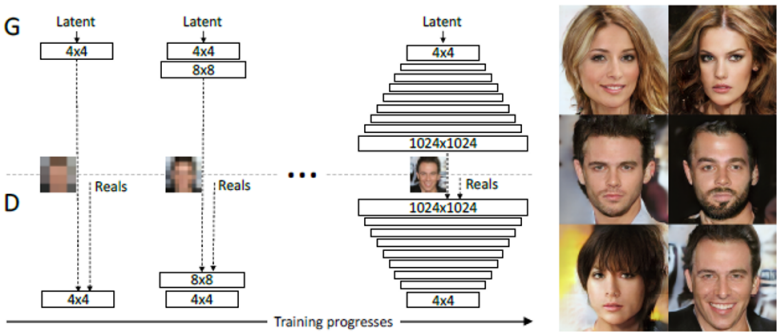


Figura 10: Explicación gráfica del modelo ProGAN[33]

ProGAN[33] mejoró progresivamente la resolución (dependiendo de la resolución de la salida actual) a través del empleo de un tamaño de mini lotes

adaptativo y agregando capas a la red durante el proceso de entrenamiento. La idea es hacer crecer tanto al generador como al discriminador. A partir de una resolución baja, se agregarían nuevas capas que modelen detalles cada vez más finos a medida que avanza el entrenamiento, en lugar de tener que aprender todas las escalas de detalle simultáneamente. Con esto la resolución mejoraría y por tanto la calidad de sus imágenes. Además, aumentará la variación de las imágenes generadas. Sin embargo, la capacidad del modelo ProGAN se limita cuando se tiene que controlar características especificas debido a que las funciones que las generan no se han entrenado por separado.

StyleGAN[34]. Con esta técnica se mejora las imágenes obtenidas a través de sus características, pudiéndose generar mayores detalles en ellas. Estos van, desde los más genéricos (cuya separación no está supervisada) como pueden ser la posición de la cara o el color del pelo hasta los más sutiles, creados a partir de variaciones estocásticas, como pueden ser el color de ojos, forma de la nariz, pecas, lunares, arrugas, etc.

Las imágenes se generarían debido a la mezcla de escalas y la interpolación de operaciones.Todo esto se realiza con un entrenamiento progresivo para modificar cada nivel de detalle por separado. En un principio se pueden regular las características más genéricas (ej. color de pelo) y una vez obtenidas se pueden matizar otras como las comentadas anteriormente (ej. color de ojos). La ventaja de este modelo es que la generación de cada característica no se ve influenciada por las otras (como si pasaría en el modelo ProGAN) pudiéndose reducir el tiempo de entrenamiento.

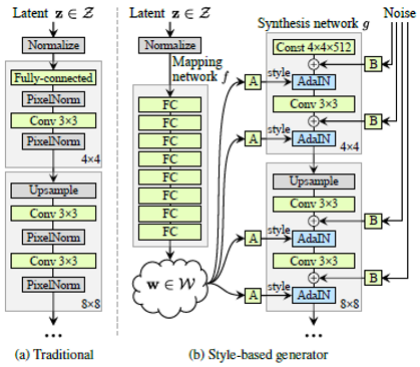


Figura 11: Arquitectura del modelo SyteGAN[34]

El StyleGAN, a diferencia de la red tradicional, tiene una primera "red de mapeo" f que codifica la entrada en un vector W de características que se usará para cargarlo como estilo a través de la normalización de la instancia adaptativa (AdaIN). Esto se realizará después de pasar la capa de convolución y haber añadido ruido gaussiano. Al usar vectores de características separados, el modelo puede combinar múltiples características tanto más genéricas como detallistas pudiendo controlar el efecto que produce el cam-

bio en alguna de ellas.

TP-GAN[35]. La red antagónica de dos vías se pensó no para crear caras nuevas sino para que dadas fotografías de perfil se pudieran generar fotos de frente. Para solventar la creación de esa parte inventada (la otra parte frontal de la cara) se va a generar tanto información general de la cara como local sintetizándolas en una sola imagen. La mejora de la solución se hará con variaciones de posición e iluminación. Con la red discriminadora se podrá ir depurando los resultados obtenidos para dar cada vez más precisión.

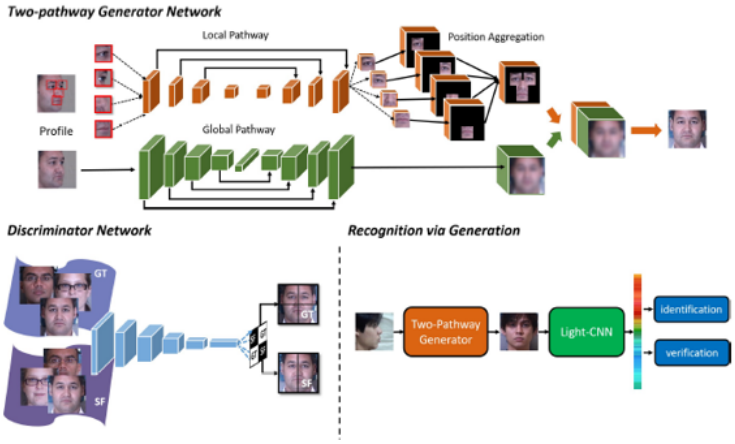


Figura 12: Explicación del modelo TP-GAN[35]

SAGAN[36]. Esta red se ha usado para manejar dependencias globales (las capas convolucionales trabajan con píxeles cercanos y por tanto extraen características locales mientas que las capas de de auto atención trabajan con bloques distantes) y asegurar que el discriminador pueda determinar con precisión las características relacionadas en regiones distantes de la imagen. El resultado de este modelo mejoro la calidad semántica de la imagen generada.

BigGAN[37]. Este tipo de red se usa para trabajar con imágenes de mejor resolución aumentando la escalabilidad (modificando la arquitectura de la red) y comprobando la inestabilidad que produce. Aplicando regularización ortogonal al generador lo hace susceptible al truncamiento. Esto permite un mayor control sobre el equilibro entre la fidelidad y la variedad de muestras al reducir la varianza de entrada del generador. Los resultados dan un mayor rendimiento. El mejor escalado es debido a que la red entrena con dos o cuatro veces más parámetros y 8 veces el tamaño de lote en comparación con anteriores técnicas. En BigGAN, la distribución está incrustada en múltiples capas del generador para influir en las características de diferentes niveles de jerarquía y resoluciones en vez de agregarse a la capa inicial. Esto hace que las fotos generadas sean mas realista y parecidas a las verdaderas.

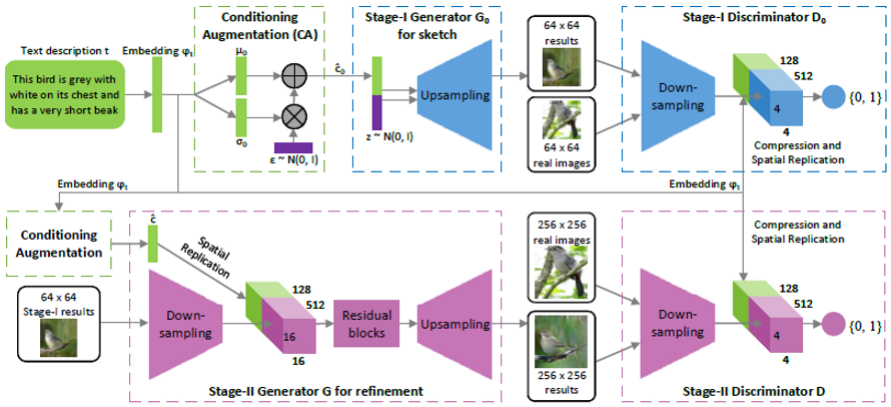


Figura 13: Descripción gráfica del modelo StackGAN[38]

StackGAN[38]. Esta red propone un modelo de GAN apilado donde se generan imágenes de alta resolución gracias a añadir descripciones textuales para su generación. Sintetizar imágenes de alta calidad a partir de descripciones de texto es un problema desafiante en la visión artificial y tiene muchas aplicaciones prácticas. Las muestras generadas por los ejemplos existentes de texto a imagen, pueden reflejar más o menos el significado de las descripciones dadas, pero no contienen detalles minuciosos ni tampoco objetos en movimiento. Para generar imágenes realistas de un cierto tamaño (256x256) se descompone el problema en subproblemas más manejables a través de un refinamiento de bocetos.

Como se puede ver en la figura anterior Stage-I GAN esboza una forma y los colores primitivos del objeto en función de la descripción de texto dado generando imágenes de baja resolución. Stage-II GAN toma los resultados de la red anterior y las descripciones de texto como entrada y genera imágenes de alta resolución. Esta red es capaz de rectificar los defectos de la anterior y agregar detalles en el proceso de refinamiento. Además, para mejorar la diversidad de las imágenes sintetizadas y para estabilizar el entrenamiento de la GAN, se añade la técnica de aumento de condicionamiento que fomenta la suavidad en la variedad del condicionamiento latente.Tras la realización de múltiples ejemplos con este tipo de GAN se muestran mejoras en las imágenes generadas.

2.4. Intercambio de expresión

Con esta técnica de deepfake se pretende modificar la expresión facial de una persona. Dos de las técnicas más populares son Face2Face [40] y [41] NeuralTextures. Estas técnicas pueden hacer cambiar las expresiones faciales de una persona por otra. Esta técnica puede llevar a engaño ya que muestra expresiones en el rostro de personas que no las han realizado.

Con la técnica Face2Face[40] se transfiere la expresión de una vídeo fuente a un vídeo destino manteniendo la identidad de la persona. Para llevar a cabo esta acción, se seleccionan fotogramas clave de un vídeo de forma manual, en concreto, se eligen los primeros fotogramas para obtener una identidad facial temporal y realizar un seguimiento de la expresión sobre los fotogramas restantes. Finalmente se transfiere los parámetros de la expresión de origen de cada fotograma al vídeo destino.

La técnica NeuralTexture[41] aborda contenido de vídeo en 3D incompleto o con ruido y a través del renderizado puede generar unas salidas bastante realistas. Para ello, se va a hacer uso del paradigma de renderizado neuronal diferido que combina la canalización de gráficos de forma tradicional con otros componentes de aprendizaje. Las texturas neuronales aprenden del mapa de características entrenado como parte del proceso de captura de escenas. Estos mapas tienen más información que los mapas de texturas tradicionales siendo estos interpretados por el renderizado neuronal diferido. Tanto las texturas neuronales como el renderizador neuronal diferido se entrenan de principio a fin por lo que trabajan con contenido 3D original imperfecto. Como mejora a las 2D se destaca el control explicito sobre la salida generada y la posibilidad de trabajar con una amplia gama de dominios de aplicación. Esta técnica se puede usar para la recreación facial, la edición de escena, etc.

Además de las anteriores técnicas, existen otras[42] que permiten cambiar la expresión facial usando un vídeo como fuente para modificar una imagen fija y así cambiar su expresión. Esta técnica hace uso del modelo StyleGAN[34] para extraer distintos atributos de la cara en un vector. Con la combinación de los vectores de dos imágenes dadas se podrá generar una nueva cara que combina la expresión del origen con la imagen destino. Esta técnica se puede aplicar a cualquier tipo de imágenes faciales sin necesidad de volver a entrenar.

Otros modelos GAN usados para estas deepfakes son: StarGAN[25] que permite cambiar la imagen de entrada con diferentes expresiones (feliz, triste, sorprendido, temeroso, etc.), AttGAN[26] o STGAN[27].

2.5. Manejo de cara

Esta técnica está asociada a vídeos ya que lo que se va a realizar con ella son los movimientos faciales, de los ojos, de la cabeza o de los gestos usando un actor fuente. La recreación facial se basa sobre todo en la expresión de la cara y el movimiento de la boca. Su uso puede aplicarse, por ejemplo, en la realización de videoconferencias multilingües, doblar actores o recrearlos (cuando estos ya no estén vivos o quieran que aparezcan más jóvenes como paso con la película Roge One, donde aparecía la princesa Leia joven o Mandalorian cuando aparece Luke) o realizar vídeo juegos basados en actores reales.

Figura 14: mejora sobre el fragmento de la serie Mandalorian donde aparece Luke Skiwalker joven[43]

El youtuber "stryder HD" incrusto la imagen de Keanu Reaves en el protagonista Jim Sakai del vídeo juego Ghost of Tsushima. Como se puede ver en la siguiente imagen todo lo que haría el protagonista del vídeo juego ahora aparece con las facciones del actor.

Figura 15: DeepFake de Keanu Reaves sobre imagen de Jim Sakai protagonista de Ghost of Tsushima[44]

Para llegar a realizar todo lo comentado anteriormente se basaron en enfoques de modelado facial 3D. Con ellos se podría capturar con precisión la geometría y el movimiento de las caras[45] pudiéndose usar sensores RGB-D. Los movimientos de la cara de origen se fusionan con la de destino creando una apariencia facial de esta última con los gestos de la primera. Estas técnicas se han llegado a aplicar en tiempo real a través de una cámara web estándar[46].

Las GAN también se han aplicado aquí para la recreación facial. A continuación, se comentan algunas de las técnicas usadas para este tipo de deepfakes:

Una de las técnicas usa **CGAN**[47] para poder generar vídeos.

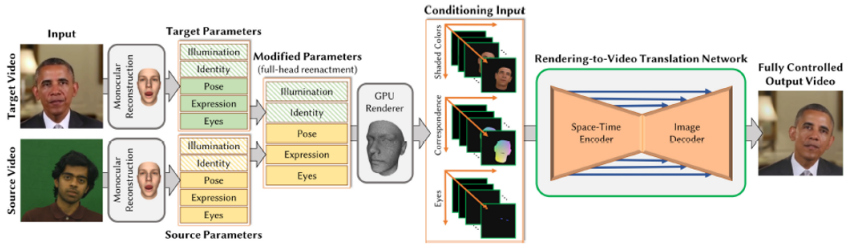


Figura 16: Arquitectura basada en CGAN [48]

Como se ve en la anterior imagen teniendo un vídeo origen y uno destino se parametrizan toda todo lo que se representa en el vídeo tanto la parte de no movimiento como la de movimiento. Esto se hace a través de la reconstrucción facial monocular. Cuando se han parametrizado, se intercambian las partes de la imagen origen como son los ojos, expresión y postura a la imagen destino. A continuación, se procesan las imágenes condicionadas y las convertimos en un vídeo bastante fotorrealista y con una coherencia temporal. Como limitación los vídeos son muy cortos (de uno a tres minutos) y es sensible a la oclusión facial.

ReenactGAN[49]. Con esta técnica se mapea la cara del origen a unos rasgos latentes delimitados. A continuación, se utiliza un transformador para adaptar la imagen latente delimitada del origen con la imagen de la cara latente delimitada del destino. Para terminar, se utiliza un decodificador especifico que genere la cara del objetivo recreada con la imagen origen.

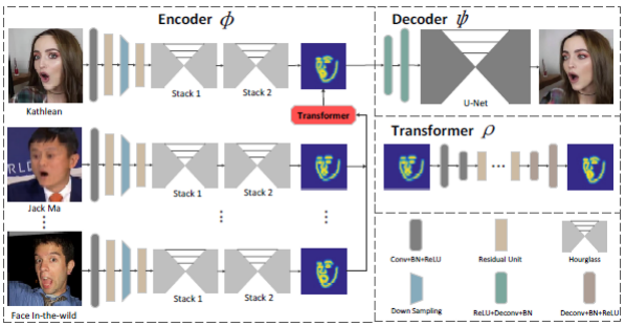


Figura 17: Arquitectura endocer-transformer-decoder del modelo ReenactGAN[49]

Esta técnica uso como proceso de recreación el feed-forward por lo que los resultados se pueden observar en tiempo real. Como limitación vemos que la mirada no queda muy adaptada y los vídeos son solo de 30 minutos.

GANimation[50]. Con esta técnica se generar imágenes mediante un esquema de condicionamiento GAN basado en anotaciones de unidades de

acción (AU) que describen una variedad continua de movimientos faciales que definen expresiones humanas. Se pretende controlar la magnitud de activación de cada AU y combinar varias de ellas. El generador basado en AU usa un mapa de atención para interpolar entre las imágenes originales y recreadas. También se integra un modelo de atención dentro de la red, el cual permite enfocar solo aquellas regiones de la imagen relevantes para cada expresión, permitiendo así procesar imágenes, aunque haya fondos o artefactos que puedan distraer. La estrategia que sigue esta técnica para entrenar el modelo es no supervisada, requiriendo solo de imágenes anotadas con sus AU activadas y usando mecanismos de atención, como ya se mencionó, que permiten que la red sea robusta antes cambios de fondo o iluminación.

GANnotatiation[51]. Se usan puntos de referencia faciales junto con un mecanismo de autodetección para la recreación facial. Esta técnica trabaja con una triple perdida de consistencia que tiene como objetivo minimizar la distancia entre las salidas generadas por la red para diferentes rutas hacia el objetivo, independientemente de cualquier paso intermedio. Las imágenes sintetizadas deben tener una vista facial frontal.

Otras técnicas usadas aquí son FSGAN[18] (descrita también en otros tipos de deepakes), FaR-GAN[52], MarioNETte[73] o pix2pixHD GAN[74] entre otras.

2.6. Sincronización de labios

Esta deepfake sintetiza un vídeo objetivo fijándose en la región de la boca para que esta esté sincronizada y consistente con la entrada de un audio. Para que el vídeo tenga un aspecto realista se ha de fijar en el movimiento y posición de la parte inferior de la boca y su zona de alrededor. Los movimientos de labios tienen que ser naturales y adecuados a lo que se habla junto también con las expresiones. Esta técnica puede tener su aplicación en el doblaje de películas a otros idiomas. También puede ser útil para personas con discapacidad auditiva ya que mediante la lectura de labios podrán entender lo que se habla.

Dentro de las técnicas que se usas para esta deepfakes nos podemos encontrar la LSTM basada en redes RNN que realizan un modelado audio/visual para que una cara hable[53]. Otro tipo de técnica propone un mapeo entre las características del audio y la forma de la boca por cada fotograma, usando la selección de varios fotogramas para completar la forma alrededor de la boca basándose en ciertos puntos de referencia de las partes inferiores de la cara: boca, mentón, nariz y mejillas. Estas partes se procesan para suavizar las líneas de expresión, también para sincronizar las pausas vocales o el movimiento de la cabeza y así hacer vídeos más realistas[54]. Otro modelo Speech2Vid[55] basado en el modelo codificador-descodificador de una CNN que usa una incrustación conjunta de la cara y el audio para

generar cuadros de vídeos sin etiquetar mediante autosupervisión intermodal (combinada). Este método se ejecuta en tiempo real y no se usan vídeos y audios usados en la fase de entrenamiento. Como limitación este modelo no es capaz de sintetizar cierta variedad de emociones en el rostro.

En cuanto a las técnicas basadas en GAN podemos destacar las siguientes:

Temporal GAN [56]. Este tipo de red se ha usado tanto para realizar un vídeo a partir de una imagen fija y la voz. Con esta red se puede sincronizar los labios, los movimiento y expresiones faciales y el parpadeo. Con la GAN Temporal con dos discriminadores se es capaz de capturar diferentes aspectos del vídeo (el primero corrobora si el vídeo generado tiene una enmarque correcto y el segundo si el vídeo tiene una sincronización adecuada).

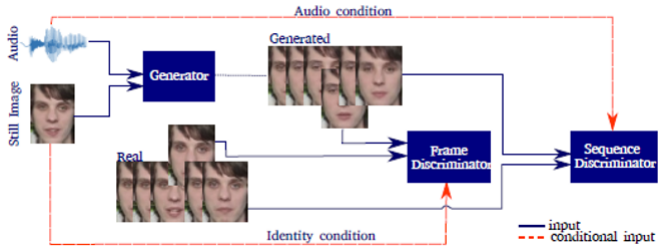


Figura 18: Arquitectura Temporal GAN[56]

LipGAN[57]. Es una red que ha servido para solucionar el problema de intentar hacer que un vídeo de alguien que hable en un idioma A pueda hacerlo en un idioma B sincronizando los labios para que quede natural.

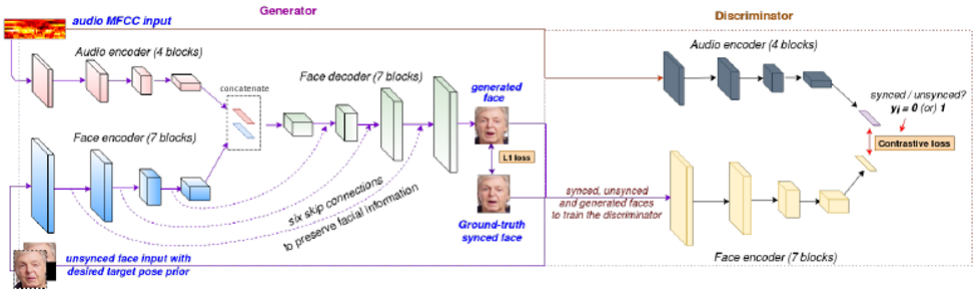


Figura 19: Arquitectura Temporal GAN[56]

Para ello se construye un sistema de traducción de voz a voz. A continuación, a través del audio traducido se podrán generar los movimientos de caras de forma bastante realista como se ve en la anterior figura. El generador generará imágenes faciales condicionadas a la entrada de audio. El discriminador verificará si el fotograma generado y el audio de entrada están sincronizados. Hay que tener en cuenta que mientras se entrena

al discriminador, se alimenta este tanto de ejemplos sincronizados como no sincronizados y así aprenderá a validar no sólo la calidad de la imagen sino también la sincronización de los labios.

Wav2Lip[58] Se puede hacer que en un vídeo de alguien aleatorio se sincronicen los labios con un segmento de voz destino. Para realizar esto se emplea un discriminador de labios ya entrenado cuyo entrenamiento aumenta con la introducción de vídeos con ruido ante la ausencia de un generador. Este modelo utiliza varios fotogramas consecutivos en lugar de uno solo en el discriminador empleando pérdida de calidad visual y de contraste, así incrementará la calidad visual por considerar la correlación temporal.

2.7. Conversión de audio

La conversión de audio manipula la voz de origen para que suene como la voz objetivo a la que quiere que se parezca mientras que el contenido textual no se modifica. Hay muchas aplicaciones al respecto, desde la industria cinematográfica donde se puede ahora dejar la voz de un actor que ya falleció hasta la suplantación de voz para hacerse pasar por otra persona al teléfono. Las características que habría que tener en cuenta son: el timbre, el tono, la amplitud, la acentuación, la duración, etc.

Se puede definir que hay dos técnicas de entrenamiento: paralelas y no paralelas. Las no paralelas se centraron en la conversión de audio multilingüe, donde el origen del que habla es convertido para que suene como el habla del destino utilizando diferentes idiomas y entrenamientos no paralelos. Las técnicas de conversión de voz paralelas se basan en el mapeo del espectro utilizando datos de entrenamiento emparejados, donde las muestras de voz tanto del origen como del destino tienen que pronunciar el mismo contenido lingüístico necesario para la conversión[4].

Existen diversas técnicas para afrontar la conversión de audio entre ellas están las basadas en: redes neuronales, GAN y VAE. De las anteriores vamos a comentar alguna relacionada con las técnicas GAN, que no usan paralelismo como, son las siguientes:

CycleGAN-vc2[59]. Esta técnica es comparable a otros métodos de conversión de voz sin depender de uso del paralelismo ni tampoco módulos o procedimientos que se alineen en el tiempo. Con esta técnica se usa un objetivo mejorado con pérdidas adversarias de dos pasos, un generador mejorado (2-1-2D CNN) y un discriminador también mejorado (PatchGAN). Estas mejoras suponen para cada par de voces mayor naturalidad y similitud entre ellas incluidas voces de distinto género. Como contrapartida tiene la limitación en coste de su generación y que sólo sirve para convertir una voz origen a un destino no a varios.

VAW-GAN[60]. Con esta técnica (GAN de codificación automática de Wasserstein) se usan modelos generativos para el habla. Estos se enfocan en explicar las observaciones con variables latentes en lugar de aprender una

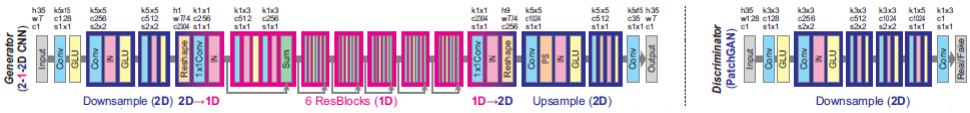


Figura 20: Arquitectura CycleGAN-vc2[59]

función de transformación entre pares evitando el requisito de alineación de tramas de voz.

STARGAN-VC[61]. Esta técnica no requiere de paralelismos, transcripciones o tiempo de procesamiento en la alineación para el tratamiento del generador de voz. Aprende, de forma simultánea, asignaciones de voces de muchos a muchos a través de diferentes dominios de atributos con una sola red generadora. Es capaz de generar señales de voz convertidas lo suficientemente rápido como para permitir implementarlas en tiempo real y solo requiere de varios minutos de ejemplos de entrenamiento para generar una salida de voz realista. Como limitación no genera tan buenas salidas cuando hay intercambio de género y tampoco cuando son voces no entrenadas previamente[4].

2.8. Texto a audio

El TTS (text to speech o texto a audio) sintetiza a partir de un texto de entrada la conversión a una voz con un sonido lo más natural posible. Esta técnica lleva décadas de uso. Sus ventajas son la posibilidad de ayuda a persona invidentes ya que mejora la interacción entre la tecnología y el humano.

Esta técnica funcionaba a través de la grabación de la voz separándola en pequeños fragmentos pudiéndose concatenar varios de estos formando un nuevo discurso. Se puede grabar desde palabras hasta frases que mejorarían la naturalidad del discurso. También se podría almacenar distintos tonos para poder poner entonación al discurso. Esta forma de generar discursos se ha vuelto un tanto obsoleta. Existe otra técnica basada en la estimación de parámetros del discurso. Los más destacados se asignarán a un texto y se convertirán en una señal de audio utilizando los codificadores de voz.

En los siguientes años se pasó a otras técnicas basadas en redes neuronales profundas. De entre ellas destacan los codificadores de voz neuronales, los autocodificadores, los modelos autorregresivos (WaveNet) y las GAN (GAN-TTS[62]) entre otras [4].

Dentro de las GAN para audio podemos hacer uso de **GAN-TTS**[62]. Esta red se compone de un generador feed-forward condicional que produce voz sin procesar y un conjunto de discriminadores que operan con una ventana aleatoria de diferentes tamaños. Los discriminadores analizan el audio en términos de realismo y también en como de bien el audio se corresponde

a una correcta pronunciación. A diferencia de los modelos autorregresivos, que dependen en gran medida de la generación secuencial de una muestra de audio durante un tiempo, las GAN-TTS son altamente paralelizables gracias al generador hacia delante (feed-forward).

3. Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales son un tipo de red neuronal utilizada en gran medida para el tratamiento de imágenes, ya que son capaces de detectar estructuras simples como puede ser líneas, bordes, curvas, y de ahí ir extrayendo formas más complejas que es lo que se pretende buscar. El hecho de realizar interacciones dispersas[63] permite que trabajemos con menos parámetros reduciendo el uso de la memoria y el tiempo de entrenamiento.

3.1. Arquitectura y tipos de redes

3.1.1. Arquitectura

La arquitectura de una red convolucional está formada sobre todo por capas de convolución, capas de agrupación, capas totalmente conectadas y su capa de salida.

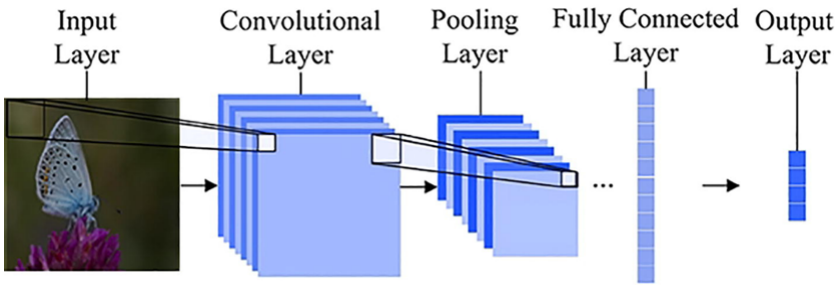


Figura 21: Arquitectura básica de una red neuronal convolucional[64]

Dentro de esta estructura básica se pueden añadir otro tipo de capas como la de activación que suelen usarse después de las de convolución.

3.1.2. Tipos de capas

Capa de convolución

En las capas de convolución se aplican a la entrada de estas una serie de filtros (kernels) definidos por unos pesos que generar un mapa de activación (feature map) por cada filtro. Los kernel se definen de un cierto tamaño que hará que la salida del mapa de activación sea más grande o pequeña. En función de las capas convolucionales que se implementen en un modelo de red convolucional se irán visualizando en sus mapas de activación desde zonas más abstractas a más concretas en función de la profundidad de la red.

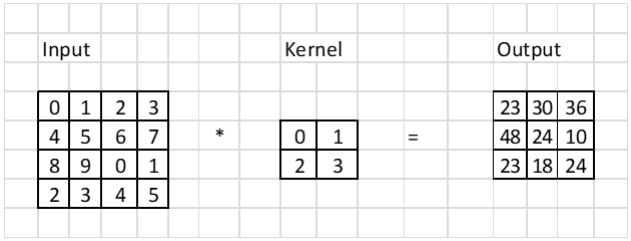


Figura 22: Ejemplo de uso de kernel con la entrada

Alguna de las técnicas para reducir o no la dimensión de la entrada de la capa convolucional son (también se puede denominar hiperparámetros ya que se podrían parametrizar sus datos):

- **padding**: Es el que haga que la salida disminuya o no en función de la entrada. En función del tamaño del padding se van a poner alrededor de la entrada celdas con valor a 0.

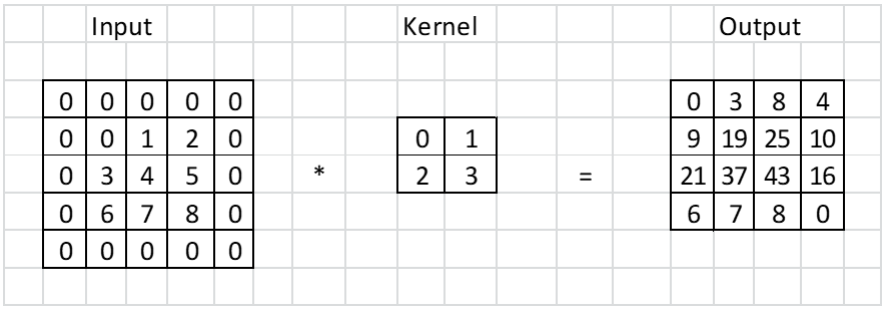


Figura 23: Ejemplo de uso de padding

- **Stride**: Indica el número de saltos que se da en la ventana para aplicar los filtros. Cuanto más grande sea el stride menor es la salida de la capa.

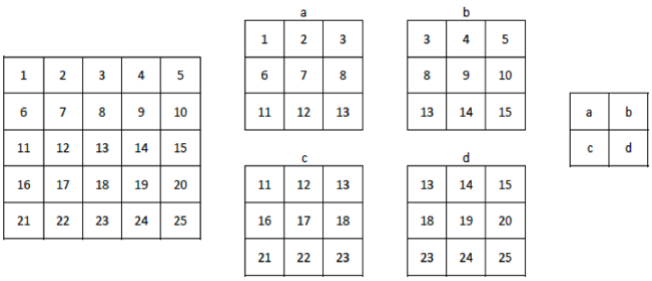


Figura 24: Ejemplo de uso de Stride[65]

Capa de convolución transpuesta

Una capa de convolución transpuesta se usa para transformar la red en sentido opuesto a una red convolucional normal, es decir, transforman algo que tiene la forma de la salida a algo que tiene la forma de entrada mientras mantiene un patrón de conectividad, por lo que el mapa de características de salida tendrá una dimensión mayor que el de la entrada.

Capa densa (Dense)

Las capas densas se utilizan cuando pueden existir una asociación entre cualquier entidad con cualquier otra entidad en ciertos puntos de datos. Se da cuando hay conexiones n1*n2 entre dos capes de tamaño n1 y n2.

Capa de activación

Esta capa está asociada a las funciones de activación que permiten que se propague los datos de un nodo a otro pasando por la función que hace que se propague de determinadas formas.

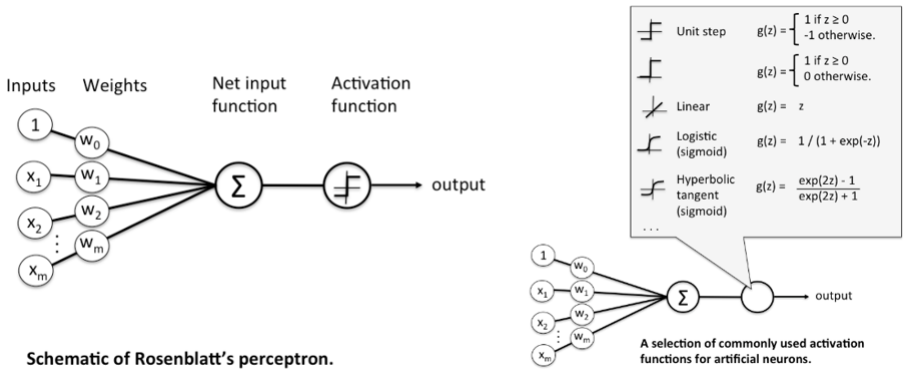


Figura 25: Esquema del perceptrón y funciones de activación[66]

Las más comunes son [67]:

- **Lineal**. Es la más básica. Se puede usar cómo capa final en soluciones de redes neuronales con regresión. Cuando se pasa por esta función la entrada se incrementa sin límite. No se puede normalizar y su convergencia a la larga es inestable.
- **Tangencial hiperbólica**. Es una función centrada en cero cuyo rango se encuentra entre [-1,1]. Proporciona un mejor rendimiento que la sigmoide en el entrenamiento de redes neuronales multicapa. Sin embargo, la función tanh no pudo resolver el problema de desaparición del gradiente sufrido también por las funciones sigmoideas. La principal ventaja que proporciona la función es que produce una salida centrada en cero que ayuda al proceso de retropropagación. Se considera la precursora de la función ReLU, ya que está última mejora

al problema de las neuronas muertas. Se utiliza sobre todo en redes neuronales recurrentes para el tratamiento del lenguaje natural y el reconocimiento del habla.

- **Sigmoide**. Es una función logística. Su salida es un valor continuo comprendido entre [0,1]. A diferencia de la lineal, la salida va a ser más suave debido a las derivadas positivas. Se puede usar en problemas de clasificación (que no sean mutuamente excluyentes) y también en regresión logística. Son fáciles de entender y su uso se observa sobre todo en redes neuronales poco profundas. Como inconveniente destaca la desaparición del gradiente, su lenta convergencia y la salida no centrada a cero, que hace que las actualizaciones del gradiente se propaguen en diferentes direcciones.

- **ReLU**. Es la función más usada en redes neuronales profundas. Ofrece un mejor rendimiento y generación en comparación con las funciones sigmoide y tangencial. Representa una función casi lineal conservando propiedades de los modelos lineales que eran fáciles de optimizar. Funcionan mejor con tasas de aprendizaje pequeñas para que no haya neuronas inutilizadas durante la fase de entrenamiento. Como desventaja es que se realizan sobreajustes más que en otras funciones como la sigmoide. Esto se suele arreglar con una capa Dropout.

- **Softmax**. se utiliza en las redes neuronales para calcular la distribución de probabilidad de un vector de números reales. Los valores que se dan a su salida tienen una probabilidad de [0,1] de forma excluyente, de forma que todos los elementos del vector suman 1. Se usa en modelos de varias clases donde se devuelve la probabilidad de cada clase siendo la clase objetivo con mayor probabilidad la elegida. Como mejora a la función Sigmoid, que puede usarse como clasificación binaria, es que puede usarse con múltiples clases.

Un resumen a modo de cuadro de las principales funciones de activación lo podemos ver en la siguiente figura.

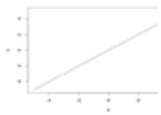
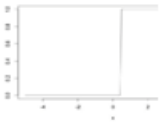
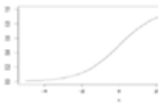
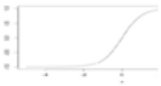
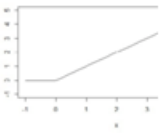
Activation Function	Description	Equation	Implementation
Linear	• The most fundamental activation function is the linear one because it does not change the neuron output at all	$\emptyset(x) = x$	
Step Or threshold	• It is simple. • It return 1 (true) for values that are above the specified threshold •	$\emptyset(x) = \begin{cases} 1, if\ x \geq 0.5 \\ 0, otherwise \end{cases}$	
Sigmoid or Logistic	• It is a popular option for feed-forward NNs that must output only positive numbers. • It ensure that values stay within a relatively small range	$\emptyset(x) = \frac{1}{1+e^{-x}}$	
Hyperbolic Tangent	• It is also a popular activation function for NNs which must output values in the range between -1 and 1. •	$\emptyset(x) = \tanh(x)$	
Rectified Linear Unit (ReLU	• It is a linear, non-saturating function. • ReLU does not saturate to -1, 0, or 1. • A saturating activation function moves towards and eventually attains a value	$\emptyset(x) = \max(0, x)$	

Figura 26: Resumen de funciones de activación[68]

Otras capas asociadas a la capa convolución

Podemos observar otras capas relacionadas con la convolución como son las de activación, vistas anteriormente, las de pooling, las de normalización y las de regulación (veremos Dropout).

Capa de pooling. Esta capa reduce el tamaño de las imágenes, dividendo la entrada en tamaños de la dimensión propuesta por en el pooling. Al final la salida dará una cifra acorde según el tipo elegido. Hay dos tipos básicos que son:

- **Max pooling**: Se utiliza el valor máximo de la dimensión seleccionada. Con esta técnica nos quedamos con las características más relevantes. Es la más usada y la que aplicaremos en los modelos expuestos posteriormente.
- **Average pooling**: Se utiliza el valor medio de la dimensión seleccionada.

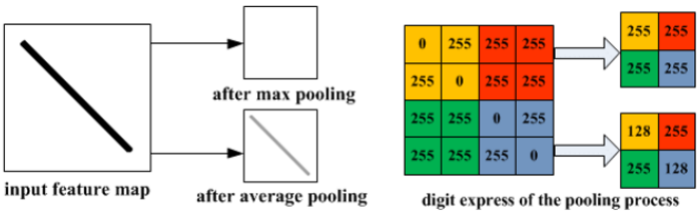


Figura 27: 26 Ejemplo de maxpooling y average pooling[69]

Capa de normalización. Esta capa escala el mapa de activación (features map) para que se puedan alcanzar mejores resultados y más rápidos. Vamos a comentar dos tipos de normalización que se han usado dentro de los modelos propuestos:

- **BatchNormalization**: Hace que cada capa aprenda por si misma de forma independiente. Se pueden usar tasas de aprendizaje alta ya que al normalizarse no hay valores extremos y convergen más rápidamente. Además, reduce el sobreajuste porque tiene un cierto efecto en la regulación. Funciona restando la media del lote y dividiendo por la desviación estándar de este[70].

- **LRN**: Está técnica viene de la neurobiología y se basa en la capacidad de una neurona de sobresalir frente a las vecinas. Se generan máximos por área. Esta técnica se suele usar con funciones de activación ReLU[71] ya que estás tienen activaciones ilimitadas y con la LRN lo que se hace es normalizarlas. Esto hará que se amortigüen las respuestas locales que son grandes. Si todos los valores son grandes la normalización los disminuirá teniéndose en cuenta sólo los que pasen de un umbral.

Capa de Dropout. Esta se considera una de las técnicas de regulación que mejora el sobreajuste de las redes.

Capa totalmente conectada (FC)

Esta capa tiene como entrada los mapas de activación (características). Mediante una o varias capas de conexión sopesa cuales son las características más importantes para llegar a una solución. Al trabajar con vectores se necesita una capa intermedia Flatten o GlobalMaxPooling que aplanen los mapas de activación para que puedan pasar a través de la red neuronal.

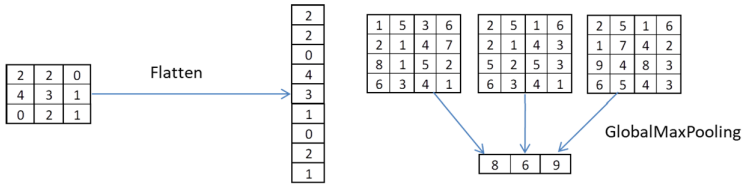


Figura 28: Ejemplo aplicando flatten y globalMaxPooling

Capa de salida

La capa de salida suele ser una capa totalmente conectada que va a tener como número de neuronas las salidas necesarias para resolver el problema. Si es una clasificación de varios elementos se pueden poner tantas neuronas como clases. Si la clasificación es binaria se puede dejar una sola neurona si es de regresión con una única neurona es suficiente. La capa de salida dependiendo del problema suele tener una función de activación distinta. Para clasificar se suele usar la Softmax o Sigmoide sino son excluyentes, mientras que para regresión se podría usar una lineal o una ReLU.

3.2. Hiperparámetros comunes de la fase de entrenamiento

Los hiperparámetros pueden referirse tanto a valores o capas dentro de la estructura del modelo de la red como los parámetros que hay en el nivel de aprendizaje. Dentro de estos últimos se puede definir los más comunes como:

- **Epochs.** Nos indica el número de veces que se pasan los datos de entrenamiento por la red. Esto provoca que el modelo vaya ajustando sus resultados.
- **Batch size.** Es el número de lotes que son necesarios para completar una época (epoch). Si una época es demasiado grande para cargarla en ordenador se divide esta en varios lotes (batch).
- **Learning rate.** Es un parámetro que indica cómo se ajustan los pesos de una red con respecto a la función de pérdida (función de costo). Cuanto más bajo este parámetro más lento iremos en el descenso del gradiente y más tardará en converger si, además, se aplana el modelo. Por lo contrario, con valores alto convergerá más rápido pero no llegaremos al mínimo ya que nos podríamos pasar, en otras palabras, la función de pérdida o costo no alcanzarán su mínimo.
- **Optimizador.** Mejora la velocidad de entrenamiento y aumenta su rendimiento. Algunos de los más usados se encuentran: SGD, Adagrad Adadelta, Adam, RMSProp. Dentro de la implementación de la prueba

de concepto de los modelos elegidos se definirán las características del optimizador usado.

3.3. Modelos de redes neuronales convolucionales

Desde los inicios de la creación de las primeras redes convolucionales como la LeNet5, pasando por la red AlexNet ha habido muchos tipos de redes cada una con mejor o peor precisión en sus resultados. Cabe destacar, que hoy en día no sólo se busca la mayor exactitud, sino que se valoran también el consumo de la energía, el tiempo en el resultado, el recuento de operaciones o el número e parámetros entre otras métricas[72]. En la siguiente figura se muestran dos gráficas, donde, en función de varios tipos de red conocidos, se observa dentro de los parámetros comentados anteriormente cuales cumplen mejor con dichas métricas.

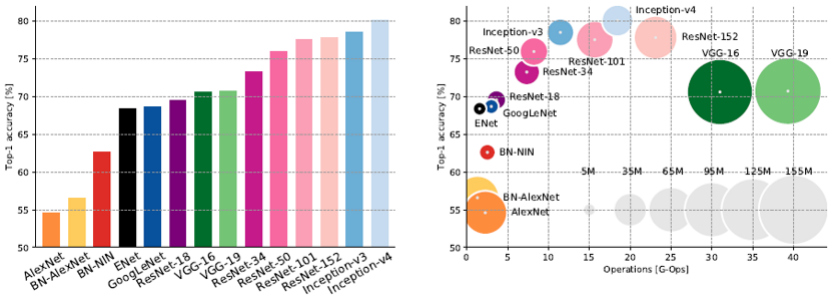


Figura 29: Exactitud con diferentes tipos de redes, operaciones y número de parámetros[72]

Nota: Este apartado 3, en el que se explican las redes neuronales convolucionales, está basado en otro TFM [92] de mi propia autoría.

4. Redes Generativas adversarias (GAN)

Como ya se introdujo en el estado del arte, una GAN[12] es un tipo de red neuronal que se utiliza en el aprendizaje no supervisado para, en este caso, generar imágenes, vídeos o audios falsos.

La GAN trabaja como se puede ver en la siguiente figura con dos partes: una red neuronal generadora que se encarga de falsificar esas imágenes, vídeos o audios y una red neuronal discriminadora que se encarga de detectar si la salida del generador es verdadera o falsa teniendo en cuenta que también se le introducirán otras entradas reales (aparte de las generadas por la red generativa).

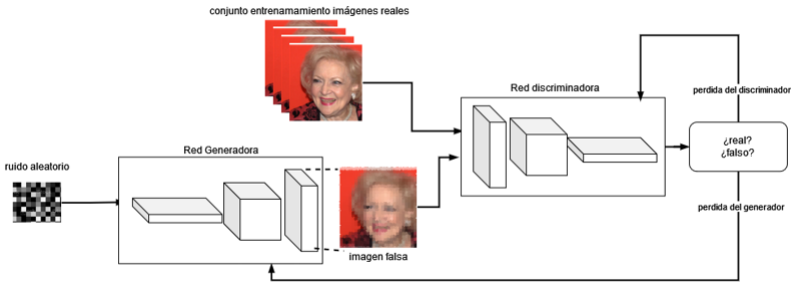


Figura 30: Arquitectura de una red GAN

Como se puede ver al principio la red neuronal generativa mandará imágenes que no se parecen a lo que se pide ya que se usará una especie de vector de ruido aleatorio y la red discriminadora lo que hará es comprobar si esa imagen producida por la red generativa es verdadera o falsa teniendo en cuenta dicha red se entrena también con imágenes reales de un conjunto de entrenamiento, es decir ambas redes se entrenan mutuamente. Para llevar esto a cabo se basan en la función de perdida tanto para el generador como para el discriminador:

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

El generador intenta minimizar la siguiente ecuación que deriva de la Binary Cross Entropy mientras que el discriminador intenta maximizarla usando el juego de sumar 0, por ejemplo, si la función de pérdida del generador bajara subiría la del discriminador y al revés. El generador utiliza los gradientes de pesos con respecto a la función de pérdida del modelo GAN, lo que conlleva a que cuantas más iteraciones se hagan el discriminador tendrá mayor probabilidad de que clasifique las imágenes generadas como verdaderas.

La GAN solo funciona si las dos redes generadora y discriminadora son más o menos igual de fuertes. Si, por ejemplo, la red generadora fuera más

potente la discriminadora no podría distinguir las imágenes falsificadas. Por el contrario, si la red discriminadora comprobara que todas las imágenes generadas son falsas no podría la generadora mejorar en la creación de nuevas imágenes. Esto ocurre cuando la función de perdida de alguna de ellas tenga el valor 0 y se produzca un estancamiento.

5. Prueba de concepto

5.1. Herramientas de ejecución de los notebooks

Dentro de las herramientas web que pueden ejecutar cuadernos con Jupyter notebook sin que lleven coste podemos encontrar varias: Google Colab[75], Kaggle[76] o Gradient[77]. Siguiendo una comparativa [80] de sus características se puede observar lo siguiente:

Plataforma			
	Colab	Kaggle	Gradient
GPU MEMORIA	12 GB	16 GB	8 GB
Modelo GPU	Tesla P100	NVIDIA TESLA P100	NVIDIA M4000
CPU RAM	13 GB	17,2 GB	30 GB
Modelo CPU	Intel Xeon	Intel Xeon	Intel Xeon
Velocidad CPU	2,0 GHz	2,3 GHz	2,6 GHz
Tiempo de uso GPU	2 hour/sesión	30 horas/semana [1]	6 horas
Max. Ejecución por sesión	12 h	9 horas por notebook	6 horas
Tiempo inactivo	30 minuto	1 hora	6 horas
Uso de la plataforma	Sin límites	30 horas/ semana	Bajo disponibilidad
Notebooks privados	Si	Si	No
Ejecución en Background	No	Si	No
Datos privados	78.19 GB	107.37 GB	5 GB

Aunque parece que tanto Kaggle como Gradient pueden dar mejores prestaciones de forma gratuita, Colab es más sencillo y tiene tiempo ilimitado en uso, de manejar por lo que la ejecución de los modelos seleccionados se ha hecho en esta plataforma. Como ya se explicará más tarde, dado el uso que tienen todas las plataformas de horario de uso de GPU que se limita en unas horas en el tiempo y a veces no permite el segundo plano, no se han podido seleccionar modelos más exactos para su ejecución, debido al tiempo en cómputo y recursos que habría que alquilar por horas, lo que conllevaría a un costo del proyecto importante.

Otras formas de ejecutar los notebooks de forma gratuita las encontramos en Azure[78] y Databricks Community Edition[79]. Sólo se ha intentado hacer pruebas con la segunda ya que la primera sólo dispone de 100 dólares que se gastan en seguida. El resultado de ejecutar uno de los modelos de GAN para crear caras no ha sido satisfactorio ya que el cluster que soporta

el cuaderno, aunque tiene 15 Gb de RAM, usa el concepto de DBU (unidad normalizada de potencia de procesamiento para Databrick) y no parece que tenga mucha velocidad de procesamiento. No se ha encontrado la equivalencia en GPU/TPU por lo que no se realizaron más pruebas con esta solución.

5.2. Dataset elegido

Los datasets, que se han usado para llevar a cabo esta prueba de concepto, son los siguientes: MNIST[81], CIFAR-10[82], IMDB-WIKI[83] y CelebA[84].

Los dos primeros dataset se han usado porque son más pequeños. En el caso del MNIST[81] son números escritos a mano de cero a nueve, cuyo tamaño se ha normalizado, siendo este de 28 pixeles cuadrados en una escala de grises (un color), por tanto, el valor del formato quedará de esta forma: 28x28x1. Para cargar estos datos en el notebook programado en Python se puede realizar a través de la librería Keras. En concreto con la sentencia:

```
from keras.datasets import mnist
```

El dasetset CIFAR-10[82] también es de tamaño pequeño, cuenta con 60000 imágenes en color clasificadas por clases (tipos) de unas 6000 imágenes por tipo. Entre las clases se pueden encontrar estas:

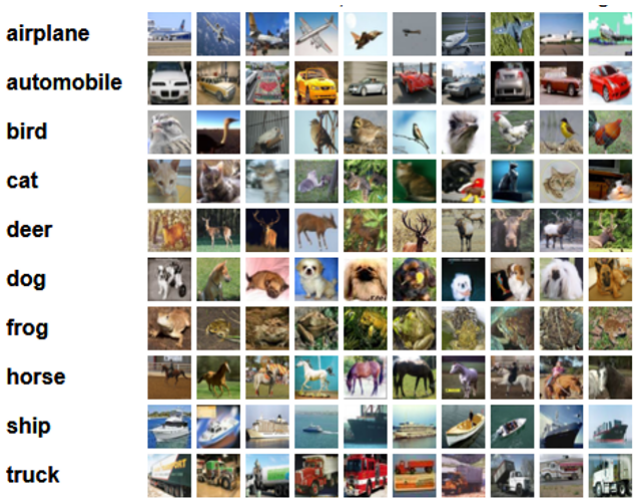


Figura 31: Tipos de clases que tiene el dataset cifar-10

En este caso los pixeles con los que se forman las imágenes de este dataset son de 32 pixeles cuadrados quedando el formato como: 32x32x3. Para trabajar con este dataset desde Python se haría igual que con el dataset MNIST, importándolos de una librería de Keras:

```
from keras.datasets import cifar10
```

Una vez probados estos datasets, se han comprobado dos datasets de caras de famosos para poder llevar a cabo el tipo de deep fake de generación de caras nuevas. En un principio se usado el dataset IMDB-WIKI[83] (el artículo donde se han sacado las imágenes es del año 2015). Para poder trabajar de forma correcta con este conjunto de imágenes, se han filtrado las que no valen, se han redimensionado a 128x128 pixeles y el fichero de los nombres de las imágenes que está en formato .mat se ha reconvertido a .csv para poder ser cargado en el notebook de Python. Todo esto se ha realizado con un script proporcionado en el siguiente github[85]. Los resultados generados con este dataset no se mostrarán en este caso de uso, ya que las imágenes no tienen buena resolución y por tanto los resultados han sido pobres.

Finalmente, el dataset elegido para llevar a cabo la prueba de concepto ha sido CelebA[84]. Este dataset también de famosos, tiene mejor resolución que el probado anteriormente. Sus más de 200.000 imágenes tiene muchas variaciones, como se puede ver en la siguiente figura.

Figura 32: Ejemplos de imágenes del dataset CelebA[84]

Para poder hacer un uso óptimo de los modelos que se han llevado a cabo en la prueba se ha realizado un recorte de la cara de dichas imágenes para encuadrarlas y que la imagen sea más pequeña. Así, el procesamiento, al aplicar uno de los modelos elegidos sobre este dataset, durará menos tiempo. El uso de todas las imágenes no es compatible con los requisitos del entorno en el que se va a procesar, por lo que se usarán unas 20.000 imágenes en lugar de conjunto total.

5.3. Librerías utilizadas

Las librerías de Python utilizadas en estas pruebas de concepto para el desarrollo de los modelos GAN junto con su entrenamiento son TensorFlow y Keras. Además, se ha hecho uso de otras librerías como: Matplotlib para poder mostrar las imágenes generadas y numPy para trabajar con arrays.

5.4. Deepfake elegido

Dentro de las arquitecturas explicadas en el estado del arte para generar caras nuevas algunas de las que tienen mejores resultados son: ProGAN[33], StyleGAN[34] y BigGAN[37] entre otras. El problema de dichas arquitecturas es que para ejecutarse necesitan un soporte hardware bastante potente, lo que implica, el alquiler de máquinas que permitan la ejecución de dichos modelos. Como se vio en los entornos de ejecución, Gradient[77] parece que permite las características necesarias de ejecución, pero el coste por hora implicaría un desembolso importante para realizar todas las pruebas, por lo que se han desechado como prueba de concepto.

En el caso del modelo ProGAN[33] usado con el dataset CelebA[84] y una tarjeta de vídeo NVIDIA Tesla V100 durará un mes su procesamiento. Si se optimiza la configuración de la tarjeta GPU tardará la mitad. Si se aplica a esa configuración 2 GPUs tardará una semana, si son 4 GPUs tardara 3 días y si son 8 GPUs tardará 2 días[85].

El modelo StyleGAN[34] recomienda para su procesamiento una GPU NVDIA DGX-1 Tesla V100. Para el dataset FFHQ de imágenes 1024x1024 se necesitarían 8 GPUs. La siguiente tabla muestra por tamaño de imagen y número de GPU lo que se tardaría en procesar el dataset con esta arquitectura[86].

GPUS	1024X1024	512X512	256X256
1	41 días 4 horas	24 días 21 horas	14 días 22 horas
2	21 días 22 horas	13 días 7 horas	9 días 5 horas
4	11 días 8 horas	7 días 0 horas	4 días 21 horas
8	6 días 14 horas	4 días 10 horas	3 días 8 horas

El modelo BigGAN[37] usa una V100 de 8 GPUs y 16 GB VRAM cada una, con precisión total. En este caso el procesamiento con esta arquitectura duraría 15 días con 150.000 iteraciones[87].

Como se puede comprobar todos estos modelos no son operativos al intentar ejecutarlos en un entorno Colab gratuito en un tiempo limitado, por lo que los módulos que se han probado son los siguientes: CoGAN[32] y DCGAN[31].

5.5. CoGAN

Con este modelo[88] se ha entrenado dos dataset. El primero IMDB-WIKI[83] con un resultado muy pobre y el segundo CelebA[84] tanto para tamaños de imágenes de 64x64 píxeles como de 128x128 píxeles. El resultado es parecido.

Arquitectura

La red propuesta [88], no sigue ninguna de las alternativas que nos dan en el artículo[32], pero dicho artículo vincula esta red propuesta como posible solución de esta técnica.

La ***red generadora*** se va a forma con las siguientes capas:

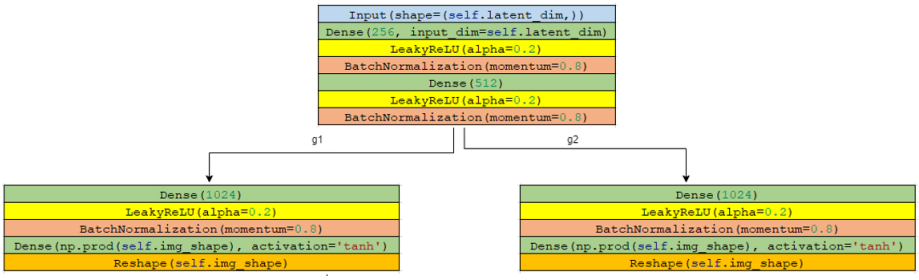


Figura 33: Red generadora CoGAN

Como se puede ver las primeras capas de la red están conectadas tal como se describió en el estado del arte ya que, aunque haya dos redes GAN, usan la primera parte de la red de forma común(**Figura 6**).

Dentro de estas capas podemos declarar que se usas el espacio latente como entrada con una dimensión de 100. Las capas se encuentran son:

- Dense de 256, 512, 1024, la primera capa carga el vector de ruido.
- Las capas de activación serán dos: LeakyRelu, que mejora el ReLu al evitar la desaparición del degradado, y la tanh, está última solo en la fase de salida de la red.
- Capa de normalización BatchNormalization.
- Reshape cambia un vector de una dimensión en tres.

Como se puede ver está red no tiene capas convolucionales sino fuertemente conectadas por las capas Dense.

La ***red discriminadora*** sigue el siguiente modelo:

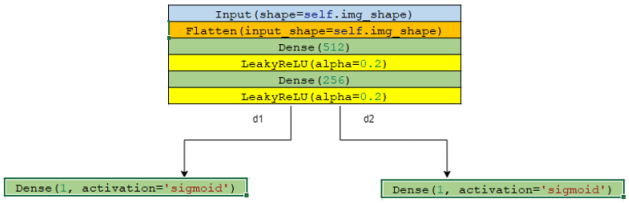


Figura 34: Red discriminadora CoGAN

Esta red recogerá el resultado de la red generadora para comprobar su validez. Aquí lo que se introducen son imágenes, en este caso con la forma 128x128x3 que se aplanarán mediante la capa *Flatten*. Estas imágenes aplanadas se cargan en las capas totalmente conectadas Dense que, junto con las de activación *LeakyRelu*, se encargaran de ir comprobando que dichas imágenes introducidas sean reales o falsas y en qué tanto por ciento. Esto es debido, a que la red no va a generar una imagen sino una clasificación por lo que las capas irán disminuyendo su salida hasta dar como resultado un número decimal, por eso se usa en la última capa la función de activación *Sigmoid*, quedando un valor decimal entre 0 y 1.

Parámetros para el entrenamiento del modelo

Para llevar a cabo el entrenamiento se han usado los siguientes hiperparámetros:

- Optimizador Adams (trabaja bien para un gran conjunto de datos) con un learning rate de 0.002. Suele dar buenos resultados en un corto periodo de tiempo.
- Tamaño del lote (batch_size) de 128.
- Épocas de ejecución 10.000 (aunque en el artículo comentan que se han realizado 25.000).

Debido a que la red discriminadora es una clasificación se usará, además de los siguientes hiperparámetros, la métrica *accuracy*, que detectará el porcentaje total de los elementos clasificados correctamente, y la función objetivo *binary_crossentropy*, ya que se realiza una clasificación.

Resultados

Como se puede ver en las siguientes imágenes ficticias generadas por el modelo CoGAN, donde se muestra la salida de cada imagen asociándolas a 500 epochs (iteración), a partir de la epoch 1000 no se mejoran demasiado la nitidez de las imágenes.

Figura 35: Imágenes generadas por el modelo CoGAN usando 9500 epoch

En las siguientes gráficas se ven la exactitud que tienen las imágenes generadas que es lo que devuelven los discriminadores de las dos GANs y en la otra se ven las pérdidas que muestran los discriminadores junto con una combinación de estas.

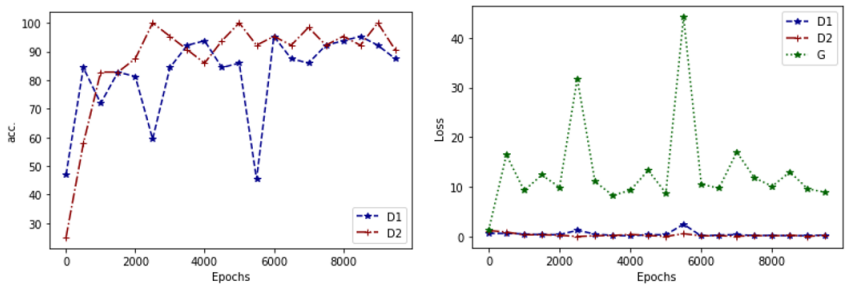


Figura 36: gráficas de exactitud y pérdida (discriminadores modelo CoGAN)

En la gráfica de la derecha se visualiza la exactitud de las dos redes discriminadoras. Parece que dichas redes tienen unos valores bastantes altos, por lo que son capaces de discriminar las imágenes falsas de las verdaderas. Esto se corrobora con la imagen de la derecha donde se ve que los valores de la función de perdida de las redes discriminadoras son muy bajos con

respecto a la G. Esto define que la red generadora no produce imágenes que engañen al discriminador. En las imágenes que se muestran a continuación, las caras no se terminan de formar y su distorsión es fuerte. El modelo parece que no mejora con la ejecución de más iteraciones. El porqué de estos resultados no tan buenos creo que es debido a que la programación de este modelo no ha seguido los patrones que se explican en el artículo, teniendo menos capas tanto convolucionales como densas que permitan mejorar la generación de imágenes.

Table 12: CoGAN for face generation

Generative models			
Layer	Domain 1	Domain 2	Shared?
1	FCONV-(N1024,K4x4,S1), BN, PReLU	FCONV-(N1024,K4x4,S1), BN, PReLU	Yes
2	FCONV-(N512,K4x4,S2), BN, PReLU	FCONV-(N512,K4x4,S2), BN, PReLU	Yes
3	FCONV-(N256,K4x4,S2), BN, PReLU	FCONV-(N256,K4x4,S2), BN, PReLU	Yes
4	FCONV-(N128,K4x4,S2), BN, PReLU	FCONV-(N128,K4x4,S2), BN, PReLU	Yes
5	FCONV-(N64,K4x4,S2), BN, PReLU	FCONV-(N64,K4x4,S2), BN, PReLU	Yes
6	FCONV-(N32,K4x4,S2), BN, PReLU	FCONV-(N32,K4x4,S2), BN, PReLU	No
7	FCONV-(N3,K3x3,S1), TanH	FCONV-(N3,K3x3,S1), TanH	No
Discriminative models			
Layer	Domain 1	Domain 2	Shared?
1	CONV-(N32,K5x5,S2), BN, PReLU	CONV-(N32,K5x5,S2), BN, PReLU	No
2	CONV-(N64,K5x5,S2), BN, PReLU	CONV-(N64,K5x5,S2), BN, PReLU	No
3	CONV-(N128,K5x5,S2), BN, PReLU	CONV-(N128,K5x5,S2), BN, PReLU	Yes
4	CONV-(N256,K3x3,S2), BN, PReLU	CONV-(N256,K3x3,S2), BN, PReLU	Yes
5	CONV-(N512,K3x3,S2), BN, PReLU	CONV-(N512,K3x3,S2), BN, PReLU	Yes
6	CONV-(N1024,K3x3,S2), BN, PReLU	CONV-(N1024,K3x3,S2), BN, PReLU	Yes
7	FC-(N2048), BN, PReLU	FC-(N2048), BN, PReLU	Yes
8	FC-(N1), Sigmoid	FC-(N1), Sigmoid	Yes

Figura 37: Red CoGAN que se propone en el artículo [32]

5.6. DCGAN

Este modelo[89] basado en la arquitectura DCGAN se ha entrenado con el dataset CelebA[84]. Se han usado tamaños de imágenes de 64x64 píxeles para que el procesamiento pueda ser más rápido. Para optimizar más aún la generación de imágenes previamente se ha realizado un recorte de estas para que solo aparezca la cara.

Arquitectura

La red propuesta consta de los siguientes modelos generador y discriminador.

La ***red generadora*** se va a forma con las siguientes capas:

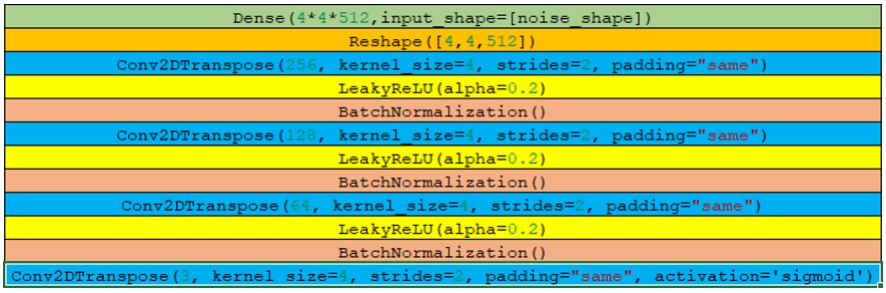


Figura 38: Red generadora DCGAN

Dentro de estas capas podemos declarar que se usas el espacio latente como entrada con una dimensión de 100. Las capas que se encuentran son:

- *Dense* de 4*4*512 que es la capa que carga el vector de ruido.
- *Reshape* cambia un vector de una dimensión en tres.
- 4 capas *convolucionales transpuestas* que van generando la imagen de salida. Se apoyan para que salga una forma parecida a la imagen real en el stride y el kernel.
- Las capas de activación serán dos: *LeakyRelu*, que mejora el ReLu al evitar la desaparición del degradado, y la *sigmoid*, está última solo en la fase de salida de la red.
- Capa de normalización *BatchNormalization*.

Otros ejemplos de DCGAN usan, en vez de *capas convolucionales 2D transpuestas* y activación de *sigmoide* en la capa final, *capas convolucionales 2D* y activación *tanh* para la capa final.

La ***red discriminadora*** sigue el siguiente modelo:

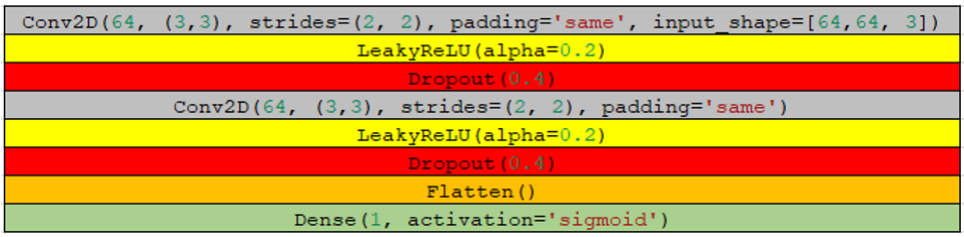


Figura 39: Red discriminadora DCGAN

Esta red recogerá el resultado de la red generadora para comprobar su validez. Aquí lo que se introducen son imágenes, en este caso con la forma 64x64x3. La red trabajará con *capas convolucionales 2D* junto con la capa de activación *LeakyRelu* para poder clasificar dichas imágenes introducidas. Las

capas disminuirán el mapa de características hasta que este esté aplanado. El resultado se cargará en la capa de tipo *Dense* que devolverá, haciendo uso de la función de activación *Sigmoid*, un valor entre 0 y 1 que deducirá el tanto por cierto de acierto en la veracidad de la imagen.

Parámetros para el entrenamiento del modelo

Para llevar a cabo el entrenamiento se han usado los siguientes hiperparámetros:

- Optimizador Adams.
- Tamaño del lote (batch_size) de 128.
- Épocas de ejecución 300.

Para este modelo se ha usado la función objetivo *binary_crossentropy* ya que se realiza una clasificación.

Resultados

Las imágenes finales después de haber entrenado con 300 epochs son las de la siguiente figura. No son muy nítidas, pero hay una evolución entre las 100 primeras iteraciones y las siguientes hasta llegar a 300 como se puede ver en la **Figura 41**.

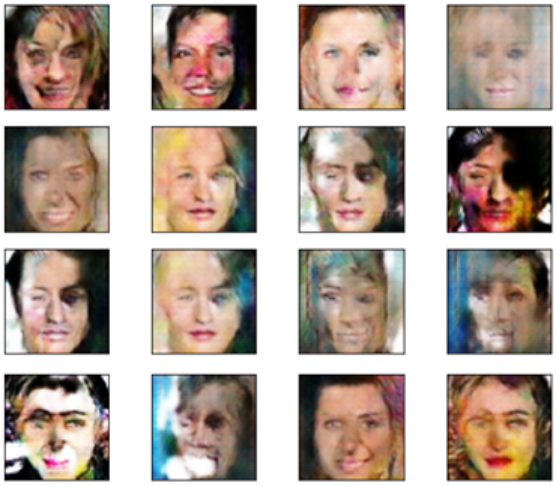


Figura 40: Imágenes generadas por el modelo DCGAN después de 300 epoch

Como se puede ver en la siguiente figura se verán los resultados de las imágenes entrenadas según la iteración que se ha llevado a cabo.

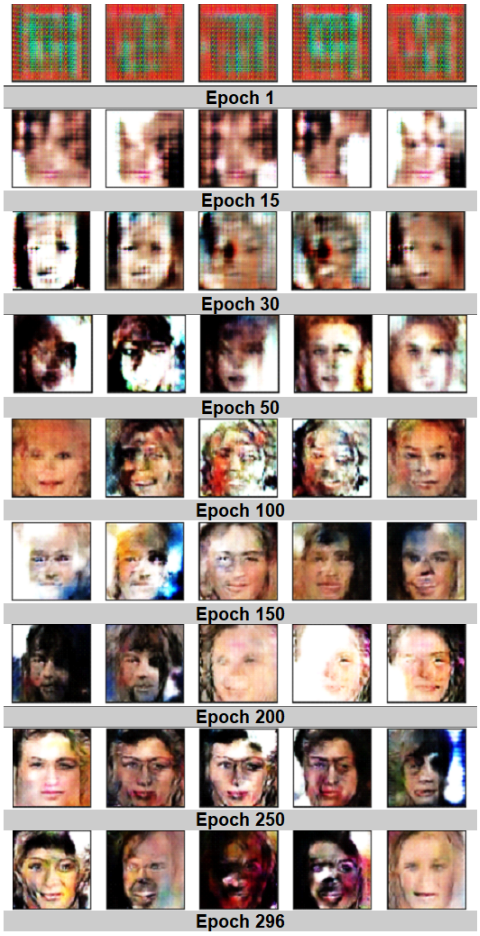


Figura 41: Imágenes generadas por el modelo DCGAN en varias un número determinado de epoch

En este caso se puede observar la evolución que se produce de las imágenes, aunque como sólo se han ejecutado 300 epochs no se terminan de visualizar muy nítidas. Esto puede ser debido a que se necesiten más epochs para ir depurando las imágenes generadas, pero también puede ser debido a que se necesite alguna capa más que mejore el proceso de generación y discriminación de estas caras.

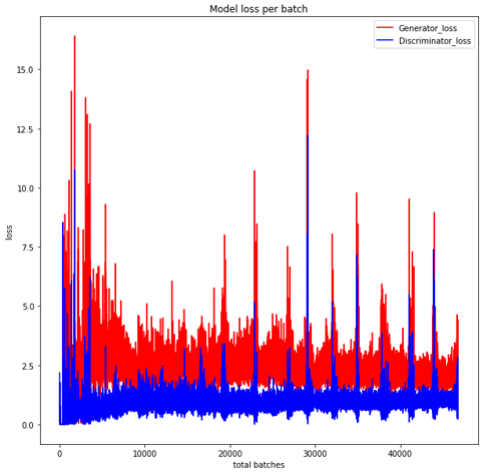


Figura 42: función de perdida en generador y discriminador para todos los lotes de las iteraciones en el modelo DCGAN ejecutado

En la anterior gráfica se puede ver que en un principio el discriminador está más elevado, pero al procesarse más iteraciones, y en consecuencia más lotes de ejecución, queda determinada que la perdida de la red generadora está siempre por encima de la discriminadora y se estabiliza en un rango. Con esto se puede deducir que se mejoran las imágenes en los primeros lotes (batch) y/o iteraciones (epochs) dejando de mejorar pasados unos 3000 batch.

También se quiere destacar que la fluctuación en la pérdida del generador se debe al vector de ruido aleatorio necesario para generar las nuevas imágenes.

5.7. Variantes de DCGAN

La última prueba realizada para la generación de caras inventadas se ha basado en este modelo[90]. Se han elegido dos datasets distintos. El primero que se ha probado es el de CIFAR-10[82] ya que es el implementado en el modelo que se ha seguido y el segundo ha sido el CelebA[84] teniendo que modificar las capas de la red generadora y la discriminadora. Aquí no ha sido necesario recortar las imágenes.

Arquitectura para el ejemplo CIFAR-10

La red propuesta consta de los siguientes modelos generador y discriminador.

La ***red generadora*** se va a forma con las siguientes capas:

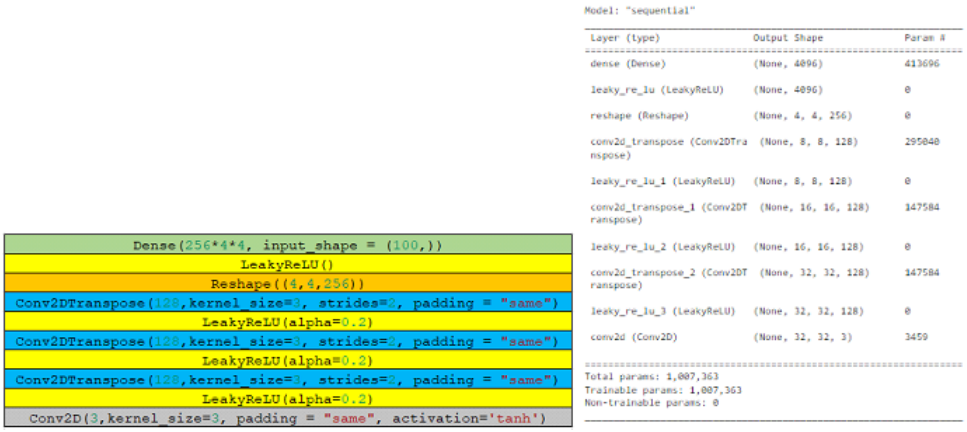


Figura 43: Red generadora GAN

Dentro de estas capas podemos declarar que se usas el espacio latente como entrada con una dimensión de 100. Las capas que se encuentran son:

- *Dense* de 4*4*512 que es la capa que carga el vector de ruido.
- Las capas de activación serán dos: *LeakyRelu*, que mejora el ReLu al evitar la desaparición del degradado, y la *tahn*, está última solo en la fase de salida de la red.
- *Reshape* cambia un vector de una dimensión en tres.
- *4 capas convolucionales transpuestas* que van generando la imagen de salida. Se apoyan para que salga una forma parecida a la imagen real en el stride y el kernel.
- No se ha utilizado la capa *BatchNormalization* debido a que no se comprobado mejorías en los resultados por ponerla.

La red devolverá imágenes con dimensión 32x32x3.

La ***red discriminadora*** sigue el siguiente modelo:

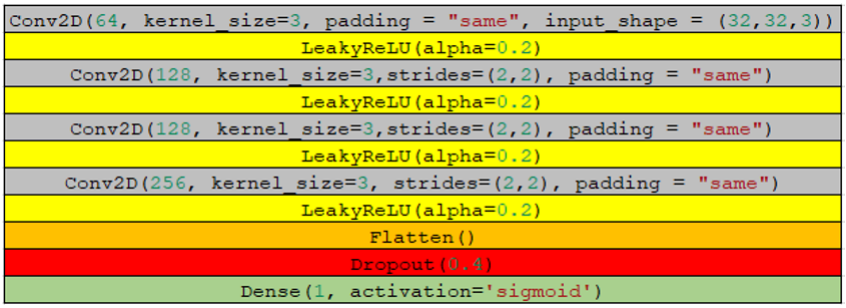


Figura 44: Red discriminadora GAN

La red discriminadora es una red convolucional normal con *4 capas de redes convolucionales* con su capa de activación *LeakyReLU* qe aplanan el mapa de características para pasarlo a la última capa Dense que mostrará como salida una clasificación que será activada por la función *Sigmoid*. Aunque es bueno usar capas de regularización, en este caso *Dropout*, para evitar el sobreajuste, en este modelo solo se ha usado en la penúltima capa, ya que así se produce mayor optimización.

Parámetros para el entrenamiento del modelo

Para llevar a cabo el entrenamiento se han usado los siguientes hiperparámetros:

- Optimizador Adams con un learning rate de 0,002 y un beta_1 0,5 recomendado en las redes DCGAN[31].
- Tamaño del lote (batch_size) de 128.
- Épocas de ejecución 300.

Además, se va a usar la función objetivo *binary_crossentropy* ya que se realiza una clasificación.

Resultados de CIFAR-10

Las imágenes finales después de haber entrenado con 300 epochs son las de la siguiente figura. Se puede observar tipos de aviones, aunque la precisión de la imagen a pesar de que es de tamaño 32x32x3 sigue siendo un poco difusa como en la red anteriormente probada.

Figura 45: Imágenes resultantes de la ejecución de otro modelo DCGAN con imágenes CIFAR-10

Arquitectura para el ejemplo CelebA

La red propuesta para este dataset dista un poco de la de CIFAR-10. Se han añadido alguna capa tanto en la parte generadora como discriminadora que ha hecho mejorar los resultados con el mismo número de iteraciones (epochs).

La ***red generadora***, en este caso, se va a forma con las siguientes capas:

```
Dense(256*4*4, input_shape = (100,))
LeakyReLU()
Reshape((4,4,256))
Conv2DTranspose(128,kernel_size=3, strides=2, padding = "same")
LeakyReLU(alpha=0.2)
Conv2DTranspose(128,kernel_size=3, strides=2, padding = "same")
LeakyReLU(alpha=0.2)
Conv2DTranspose(128,kernel_size=3, strides=2, padding = "same")
LeakyReLU(alpha=0.2)
Conv2DTranspose(128,kernel_size=3, strides=2, padding = "same")
LeakyReLU(alpha=0.2)
Conv2D(3,kernel_size=3, padding = "same", activation='tanh')
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 4096)              413696
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 4096)              0
_________________________________________________________________
reshape (Reshape)            (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_transpose (Conv2DTra  (None, 8, 8, 128)         295040
nspose)
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_transpose_1 (Conv2DT  (None, 16, 16, 128)       147584
ranspose)
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DT  (None, 32, 32, 128)       147584
ranspose)
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_transpose_3 (Conv2DT  (None, 64, 64, 128)       147584
ranspose)
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 64, 64, 3)         6147
=================================================================
Total params: 1,157,635
Trainable params: 1,157,635
Non-trainable params: 0
_________________________________________________________________
```

Figura 46: modelo DGGAN modificado para CelebA e imágenes 64x64x3

La diferencia con la anterior red es que se ha añadido una capa convolucional transpuesta más y una capa de activación para que pueda generar imágenes de mayor tamaño acordes a las que hemos pasado de 64x64x3.

En cuanto a la ***red discriminadora*** el modelo tendría la siguiente modificación del anterior expuesto:

```
Conv2D(64, kernel_size=3, padding = "same", input_shape = (64,64,3))
LeakyReLU(alpha=0.2)
Conv2D(128, kernel_size=3,strides=(2,2), padding = "same")
LeakyReLU(alpha=0.2)
Conv2D(256, kernel_size=3,strides=(2,2), padding = "same")
LeakyReLU(alpha=0.2)
Conv2D(256, kernel_size=3,strides=(2,2), padding = "same")
LeakyReLU(alpha=0.2)
Conv2D(512, kernel_size=3,strides=(2,2), padding = "same")
LeakyReLU(alpha=0.2)
Flatten()
Dropout(0.4)
Dense(1, activation='sigmoid')
```

Figura 47: Red discriminadora con una capa más de convolución DCGAN para CelebA

Esta red es idéntica a la anterior, añadiendo una capa de convolución con su correspondiente activación para que pueda clasificar en mejor medida las imágenes cargadas en el modelo. Como se ven en los resultados, esta capa los ha mejorado gracias a poder clasificar con más exactitud las imágenes cargadas en el modelo.

Parámetros para el entrenamiento del modelo

En cuanto a los parámetros propuestos para esta red se han dejado los mismo que para el anterior modelo DCGAN entrenado para el dataset CIFAR-10.

Resultados de CelebA

En la siguiente imagen se muestran los resultados obtenidos tras ejecutar 200 iteraciones. Aquí se paró antes por el problema de uso de la GPU en Colab, ya que al ser redes más profundas el tiempo ejecutar una iteración es más lento. Como se puede observar las imágenes obtenidas mejoran considerablemente a los anteriores casos. En el caso de dataset CIFAR-10, a pesar de que este modelo se basa en él, se ven peores las imágenes de salida, por lo que las capas añadidas han mejorado el resultado. En cuanto a los resultados de los otros modelos DCGAN y CoGAN también se puede ver que con este modelo son mejores.

Aunque en los anteriores modelos hemos mostrado las gráficas de perdida y exactitud para dar una respuesta cuantitativa a los resultados, no son muy gráficas confiables para ver si un modelo converge a un tipo de solución. Existen otras técnicas para visualizar un resultado más fiable, estas son[91]: inception score, Frechet Inception Distance (FID score) y medidas de similitud percentual (LPIPS) que se ajustaría mejor a la posibilidad de estudiar si un modelo está funcionando bien o no. Por tanto, en este apartado las interpretaciones se basan en lo que se puede observar a simple vista (respuesta cualitativa).

Figura 48: Imágenes resultado de ejecutar el modelo DCGAN mejorado para CelebA

5.8. Resumen comparativo de los resultados

Como se ha venido explicando en este apartado probar cualquier técnica de tipo GAN que solucione un tipo de deep fake (en este caso la creación de caras inventadas) requiere de unos recursos que se salen de un ordenador convencional. Dentro de los modelos que consumen menos recursos se han considerados los modelos CoGAN y DCGAN. En concreto, esté último usando un dataset de imágenes con buena resolución, como es el CelebA, y modificando algo las redes generadora y discriminadora, se han conseguido en un tiempo razonables una buena respuesta, creando caras inventadas más o menos reales.

6. Conclusiones

Las técnicas deepfakes, tanto para vídeo, imagen y audio, están evolucionando de forma muy rápida llegando a unos grados de exactitud sorprendentes, no detectando si el resultado obtenido es fake o no. Dicho éxito se debe en gran medida al uso de las redes GAN para la generación de estos resultados.

En este trabajo se ha podido investigar que tipo de técnicas deepfake hay y que modelos se usan en ellas para obtener buenos resultados. Siendo este un trabajo bibliográfico orientado a la recopilación de técnicas, y dado el volumen de modelos que se pueden encontrar, se han resumido de cada técnica deepfake unas cuantas, centrándose en las que han usado las redes GAN para elaborar un resultado. Cabe destacar que no se han evaluado de todos ellos las ventajas y desventajas que se pueden encontrar, ya que el trabajo sería demasiado extenso, por lo que se han resumido algunas de las técnicas, siendo un poco más exhaustivo en las que van a ser luego llevadas a la práctica en la prueba de concepto (la generación de caras nuevas). El orden expuesto en la explicación de cada técnica implica una cronología en su desarrollo. El modelo asociado a cada técnica suele tener ventajas en cuanto a la exactitud de los resultados que mejoran los modelos anteriormente expuestos.

Como complemento a la explicación de los modelos anteriores, y debido a que estos se basan en redes GAN (que suelen estar basadas en redes profundas y convolucionales), se ha explicado la arquitectura de una red convolucional, diseccionando sus capas y definiendo los hiperparámetros necesarios para su entrenamiento. A su vez, se ha definido como trabajarían las redes GAN, tanto las generadoras como las discriminadoras.

De las técnicas explicadas se ha llevado a una prueba de concepto las asociadas a la técnica deepfake de generación de nuevas caras. Para ello se ha hecho un estudio, primero de las que dieron mejores resultados, y luego de las que han sido viables para llevarlas a cabo dentro de un entorno de ejecución gratuito. Como ya se ha visto en este trabajo, técnicas como ProGAN[33], StyleGAN[34] y BigGAN[37] dan muy buenos resultados e imágenes muy realistas, pero el coste computacional es tan grande que no se ha podido llevar a la práctica. Las técnicas usadas finalmente han sido las de CoGAN[32] y DCGAN[31] siendo está última la que ha generado imágenes más reales con menos iteraciones haciendo uso del dataset CelebA[84]. Dicho esto, aunque se ha usado un modelo basado en esta técnica DCGAN[90] se ha tenido que realizar alguna modificación en la red generadora y la discriminadora para que se vieran mejores resultados. De hecho, el modelo no sigue todas las características de la DCGAN puesto que no hay se aplica la capa de normalización, pero sí de regularización con la capa Dropout para que no se produzca sobreentrenamiento, en este caso, en la capa discriminadora. Con este caso de uso se ha podido ver que con unas redes no muy profundas y

unas 200 iteraciones los resultados son más veraces que con otras técnicas que compartan pesos como puede ser la CoGAN.

Otro punto por destacar es que la función de perdida que se muestra en el transcurso del entrenamiento de los distintos modelos no da una seguridad en la convergencia de la solución, no siendo una buena medida para poder ver cuando parar el entrenamiento de un modelo, siendo necesario basarse en el resultado cualitativo. Se ha podido comprobar que existen otras técnicas más exactas a la hora de ver cuando un modelo ya no mejora los resultados de sus salidas como son: inception score, Frechet Inception Distance (FID score) y medidas de similitud percentual (LPIPS)[91]. Sería interesante, de cara a seguir con este trabajo, realizar pruebas con las anteriores técnicas que puedan ver si los modelos convergen hacia una mejora o se quedan estancados.

Con este trabajo se han podido ahondar en el conocimiento de cómo funcionan la generación de deepakes, algo que tiene usos positivos y negativos. Relacionándolo con la Ciberseguridad, implica que las mejoras producidas en las técnicas descritas en el estado del arte hacen que sea complicado la detección a simple vista de la veracidad de los elementos multimedia (imagen, vídeo y audio) generados con ellas. Esto nos llevará a tener que hacer uso de otras técnicas que comprueben la veracidad de las imágenes generadas [4], siendo también interesante poder hacer un resumen de estas técnicas y probar alguna como una línea futura del proyecto. Todo esto, no sólo supone un avance tecnológico, sino que, si no disponemos de esas técnicas, supondrá el descrédito de la información que recibimos, algo muy peligroso en la sociedad actual.

Como resumen, hay que decir que el objetivo del trabajo ha sido explicar diversas técnicas deepfakes, los posibles modelos que las implementan y como han ido evolucionando, sobre todo en estos últimos años con la llegada de las GAN. Se ha incluido también en este objetivo, llevar a la práctica un ejemplo de modelo GAN que demuestre como trabaja por dentro para poder generar, en este caso imágenes ficticias. Por ello, en líneas generales el trabajo ha cumplido los hitos marcados en su planificación.

7. Glosario

Definición de los términos y acrónimos utilizados:

CNN - Convolutional Neural Network.

DBU – es una unidad normalizada de potencia de procesamiento en la plataforma Databricks Lakehouse que se utiliza con fines de medición y fijación de precios. La cantidad de DBU que consume una carga de trabajo depende de las métricas de procesamiento, que pueden incluir los recursos informáticos utilizados y la cantidad de datos procesados. Consulte los detalles de precios de DBU para AWS, Azure y Google Cloud.

DCGAN - Deep Convolutional Generative Adversarial Network.

Desaparición del gradiente Desaparición del gradiente – Cuando se actualizan los pesos de la red, a través de la derivada parcial de la función de error con respecto al peso actual en cada iteración del entrenamiento, puede ser que estos valores se hagan tan pequeños que el gradiente llegue a desaparecer, y debido a que no se pueda cambiar el valor de la red el entrenamiento deja de continuar.

El filtro de Kalman - Produce estimaciones de variables no observables basadas en mediciones inexactas e inciertas de variables observables. Además, el filtro de Kalman proporciona una predicción del estado futuro del sistema, basado en estimaciones pasadas.

Espacio latente – Es simplemente una representación de los datos comprimidos, donde los puntos de datos similares están más juntos en el espacio. Con la GAN se obtiene un espacio/distribución latente que se aproxime al espacio/distribución latente real de los datos observados.

FFHQ – Flickr-Faces-HQ Dataset.

FID score - Fréchet inception distance. Es una métrica utilizada para evaluar la calidad de las imágenes creadas por un modelo generativo, como una red adversarial generativa (GAN). FID compara la distribución de imágenes generadas con la distribución de imágenes reales que se utilizaron para entrenar al generador.

GAN - Generative Adversarial Network.

GPU – Graphics processing unit.

IS - Inception Score. Esta métrica utiliza dos criterios para medir el rendimiento de GAN: la calidad de las imágenes generadas y su diversidad basada en la entropía de la distribución de datos sintéticos.

LPIPS - Learned Perceptual Image Patch Similarity.

LSTM – Long Short term memory. Esta red es capaz de recordar un dato relevante y almacenarlo desde un corto espacio de tiempo hasta un largo espacio.

MTCNN - Multi-task Cascaded Convolutional Networks.

RGB-D – Los sensores son un tipo específico de dispositivos de detección de profundidad que funcionan en asociación con una cámara con sensor

RGB (color rojo, verde y azul).

RNN– Red neuronal recurrente. Se suelen usar para tratar la dimensión del tiempo, pero sólo a corto plazo.

TFM - Trabajo Fin de Máster.

TPU – Tensor processing unit.

Variaciones estocásticas – cambia las imágenes al agregar ruido en cada punto del modelo generador. Dicho ruido se agrega al mapa de características completo y permite que el modelo interprete el estilo de manera detallada y por píxel.

VGGFACE - Visual Geometry Group Face, modelo para reconocimiento facial.

8. Bibliografía

Referencias

[1] S. Tietz and K. Nassiri Nazif, "Attacking autonomous driving machine learning algorithms with adversarial examples," Standford University, 2019. [Online]. Available: http://cs230.stanford.edu/projects_spring_2019/reports/18681219.pdf

[2] Chapman P, Clinton J, Kerber R, Khabaza T, Reinart T, Shearer C, Wirth R. CRISP-DM 1.0. Step-by-step data mining guide [Internet]. 1999-2000. Available from: https://www.the-modeling-agency.com/crisp-dm.pdf

[3] O'Brien M, Chrettieneau,L. ProjectLibre [Internet]. [EEUU, France 2012]. Disponible en: https://www.projectlibre.com/

[4] Masood, Momina & Nawaz, Marriam & Malik, Khalid & Javed, Ali & Irtaza, Aun. Deepfakes Generation and Detection: State-of-the-art, open challenges, countermeasures, and way forward. 2021.

[5] Tolosana, Ruben & Vera-Rodriguez, Ruben & Fierrez, Julian & Morales, Aythami & Ortega-Garcia, Javier. DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. 2020.

[6] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, .ºn face segmentation, face swapping, and face perception,ïn 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), 2018, pp. 98-105: IEEE.

[7] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar, "Face swapping: automatically replacing faces in photographs,ïn ACM Transactions on Graphics (TOG), 2008, vol. 27, no. 3, p. 39: ACM.

[8] Faceswap: Deepfakes software for all, Available at: https://github.com/deepfakes/faceswap Accessed: September 08, 2020.

[9] Faceswap-GAN Available at: https://github.com/shaoanlu/faceswap-GAN Accessed: September 18, 2020.

[10] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, Richard Zhang: Swapping Autoencoder for Deep Image Manipulation. CoRR abs/2007.00653 (2020)

[11] Yanzong Guo, Wangpeng He, Juanjuan Zhu, and Cheng Li. 2018. A Light Autoencoder Networks for Face Swapping. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial

Intelligence (CSAI '18). Association for Computing Machinery, New York, NY, USA, 459–462. DOI:https://doi.org/10.1145/3297156.3297210

[12] I. Goodfellow et al., "Generative adversarial nets,ïn Advances in neural information processing systems, 2014, pp. 2672-2680.

[13] Faceswap-GAN Available at: https://github.com/shaoanlu/faceswap-GAN. Accessed: September 18, 2020.

[14] I. Petrov et al., "DeepFaceLab: A simple, flexible and extensible face swapping framework,.ªrXiv preprint arXiv:2005.05535, 2020.

[15] I. Korshunova, W. Shi, J. Dambre, and L. Theis, "Fast face-swap using convolutional neural networks,ïn Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3677-3685.

[16] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, .ºn face segmentation, face swapping, and face perception,ïn 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), 2018, pp. 98-105: IEEE.

[17] D. Chen, Q. Chen, J. Wu, X. Yu, and T. Jia, "Face Swapping: Realistic Image Synthesis Based on Facial Landmarks Alignment,"Mathematical Problems in Engineering, vol. 2019, 2019.

[18] Y. Nirkin, Y. Keller, and T. Hassner, "FSGAN: Subject Agnostic Face Swapping and Reenactment,ïn Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 7184-7193.

[19] R. Natsume, T. Yatagawa, and S. Morishima, "Fsnet: An identity-aware generative model for image-based face swapping,ïn Asian Conference on Computer Vision, 2018, pp. 117-132: Springer.

[20] R. Natsume, T. Yatagawa, and S. Morishima, Rsgan: face swapping and editing using face and hair representation in latent spaces,.ªrXiv preprint arXiv:1804.03447, 2018.

[21] L. Li, J. Bao, H. Yang, D. Chen, and F. Wen, "Faceshifter: Towards high fidelity and occlusion aware face swapping,.ªrXiv preprint arXiv:1912.13457, 2019.

[22] FaceApp, Available at: https://www.faceapp.com/ Accessed: September 17, 2020.

[23] G. Perarnau, J. Van De Weijer, B. Raducanu, and J. M. Álvarez, Ïnvertible conditional gans for image editing,.ªrXiv preprint arXiv:1611.06355, 2016.

[24] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. A. Ranzato, "Fader networks: Manipulating images by sliding attributes,ïn Advances in neural information processing systems, 2017, pp. 5967-5976.

[25] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation,ïn Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8789-8797.

[26] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, .Attgan: Facial attribute editing by only changing what you want,ÏEEE Transactions on Image Processing, vol. 28, no. 11, pp. 5464-5478, 2019.

[27] M. Liu et al., "Stgan: A unified selective transfer network for arbitrary image attribute editing,ïn Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 3673-3682.

[28] G. Zhang, M. Kan, S. Shan, and X. Chen, "Generative adversarial network with spatial attention for face attribute editing,ïn Proceedings of the European conference on computer vision (ECCV), 2018, pp. 417-432.

[29] Z. He, M. Kan, J. Zhang, and S. Shan, "PA-GAN: Progressive Attention Generative Adversarial Network for Facial Attribute Editing,.arXiv preprint arXiv:2007.05892, 2020.

[30] D. P. Kingma and M. Welling, .Auto-encoding variational bayes,.arXiv preprint arXiv:1312.6114, 2013.

[31] A. Radford, L. Metz, and S. Chintala, Ünsupervised representation learning with deep convolutional generative adversarial networks,.arXiv preprint arXiv:1511.06434, 2015.

[32] M.-Y. Liu and O. Tuzel, Çoupled generative adversarial networks,ïn Advances in neural information processing systems, 2016, pp. 469-477.

[33] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation,.arXiv preprint arXiv:1710.10196, 2017.

[34] T. Karras, S. Laine, and T. Aila, .A style-based generator architecture for generative adversarial networks,ïn Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 4401-4410.

[35] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis,ïn Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2439-2448.

[36] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks,ïn International Conference on Machine Learning, 2019, pp. 7354-7363: PMLR.

[37] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis,.ªrXiv preprint arXiv:1809.11096, 2018.

[38] H. Zhang et al., "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,ïn Proceedings of the IEEE international conference on computer vision, 2017, pp. 5907-5915.

[39] Ng Jen Neng, Chandra Reka Ramachandiran, Mandava Rajeswari. "Representation learning based deepfake detection with facial region features" Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 5, no. 1, 2021.

[40] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-Time Face Capture and Reenactment of RGB Videos," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016.

[41] J. Thies, M. Zollhofer, and M. Nießner, "Deferred Neural Rendering: Image Synthesis using Neural Textures," ACM Transactions on Graphics, vol. 38, no. 66, pp. 1–12, 2019.

[42] H. Averbuch-Elor, D. Cohen-Or, J. Kopf and M.F. Cohen, "Bringing Portraits to Life," ACM Transactions on Graphics, vol. 36, no. 6, p. 196, 2017

[43] Shamook https://www.youtube.com/watch?v=wrHXA2cSpNU (10/03/2022)

[44] stryder HD https://www.youtube.com/watch?v=pot-JOEG4fo (10/03/2022)

[45] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, Real-time expression transfer for facial reenactment,.ªCM Trans. Graph., vol. 34, no. 6, pp. 183:1-183:14, 2015.

[46] J. Thies, M. Zollhöfer, and M. Nießner, ÏMU2Face: Real-time Gesture-driven Facial Reenactment,.ªrXiv preprint arXiv:1801.01446, 2017.

[47] Mehdi Mirza, Simon Osindero, Çonditional Generative Adversarial Nets.ªrXiv preprint arXiv:1411.1784, 2014.

[48] H. Kim et al., "Deep video portraits,.ªCM Transactions on Graphics (TOG), vol. 37, no. 4, p. 163, 2018.

[49] W. Wu, Y. Zhang, C. Li, C. Qian, and C. Change Loy, Reenactgan: Learning to reenact faces via boundary transfer,ïn Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 603-619.

[50] A. Pumarola, A. Agudo, A. M. Martínez, A. Sanfeliu, and F. Moreno-Noguer, "GANimation: Anatomically-Aware Facial Animation from a Single Image,ïn Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 818-833.

[51] E. Sanchez and M. Valstar, "Triple consistency loss for pairing distributions in GAN-based face synthesis,.ªrXiv preprint arXiv:1811.03492, 2018.

[52] H. Hao, S. Baireddy, A. R. Reibman, and E. J. Delp, "FaR-GAN for One-Shot Face Reenactment,.ªrXiv preprint arXiv:2005.06402, 2020.

[53] B. Fan, L. Wang, F. K. Soong, and L. Xie, "Photo-real talking head with deep bidirectional LSTM,ïn 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 4884-4888: IEEE.

[54] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing Obama: learning lip sync from audio,.ªCM Trans. Graph., vol. 36, no. 4, pp. 95:1-95:13, 2017.

[55] A. Jamaludin, J. S. Chung, and A. Zisserman, "You said that?: Synthesising talking faces from audio,Ïnternational Journal of Computer Vision, pp. 1-13, 2019

[56] K. Vougioukas, S. Petridis, and M. Pantic, .ºEnd-to-End Speech-Driven Realistic Facial Animation with Temporal GANs,ïn Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 37-40.

[57] P. KR, R. Mukhopadhyay, J. Philip, A. Jha, V. Namboodiri, and C. Jawahar, "Towards automatic face-to-face translation,ïn Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1428-1436.

[58] K. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar, .ª Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild,ïn Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 484-492.

[59] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, Çyclegan-vc2: Improved cyclegan-based non-parallel voice conversion,ïn ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 6820-6824: IEEE.

[60] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks,.ªrXiv preprint arXiv:.00849, 2017.

[61] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "Stargan-vc: Non-parallel many-to-many voice conversion using star generative adversarial networks,ïn 2018 IEEE Spoken Language Technology Workshop (SLT), 2018, pp. 266-273: IEEE.

[62] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, Karen Simonyan "High Fidelity Speech Syntesis with Adversarial Networks.ªrXiv preprint arXiv:1909.11646, 2019

[63] Bosch Anna, Casas Jordi, Lozano Toni. Deep Learning. Principios y fundamentos. Editorial UOC. 2020

[64] Ayad Saad Almryad, Hakan Kutucu. Automatic identification for field butterflies by convolutional neural networks, Engineering Science and Technology, an International Journal, Volume 23, Issue 1, 2020, Pages 189-195, ISSN 2215-0986, (http://www.sciencedirect.com/science/article/pii/S2215098619326011)

[65] Torres Jordi. Deep Learning. Introducción práctica con Keras. Primera parte. Watch this space. 2018.

[66] Sebastian Raschka. Single-Layer Neurnal Networks and Gradient Descent. 2015.

[67] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. 2018 (https://www.researchgate.net/publication/328826136_Activation_Functions_Comparison_of_trends_in_Practice_and_Research_for_Deep_Learning)

[68] Ahmed Salman, Hanaa. (2018). Effect of Successive Convolution Layers to Detect Gender. 59. 1717-1732. 10.24996/ijs.2018.59.3C.17.

[69] Yu, Dingjun & Wang, Hanli & Chen, Peiqiu & Wei, Zhihua. Mixed Pooling for Convolutional Neural Networks. 2014 .364-375.

[70] Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, in PMLR. 2015

[71] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 2012.

[72] Canziani, Alfredo & Paszke, Adam & Culurciello, Eugenio. An Analysis of Deep Neural Network Models for Practical Applications. 2016

[73] S. Ha, M. Kersner, B. Kim, S. Seo, and D. Kim, "Marionette: Few-shot face reenactment preserving identity of unseen targets,ïn Proceedings of the AAAI Conference on Artificial Intelligence, 2020, vol. 34, no. 07, pp. 10893-10900.

[74] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans,ïn Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8798-8807.

[75] https://colab.research.google.com/(05/04/2022)

[76] https://www.kaggle.com/(10/04/2022)

[77] https://gradient.run/(12/04/2022)

[78] https://azure.microsoft.com/es-es/free/students/(14/04/2022)

[79] https://community.cloud.databricks.com/login.html(14/04/2022)

[80] Jeff Holmes. Comparison of Basic Deep Learning Cloud Platforms. https://medium.com/geekculture/comparison-of-basic-deep-learning-cloud-platforms-337657edb710. (06/01/2022)

[81] http://yann.lecun.com/exdb/mnist/ (05/04/2022)

[82] https://www.cs.toronto.edu/~kriz/cifar.html (05/04/2022)

[83] Rasmus Rothe, Radu Timofte, Luc Van Gool IMDB-WIKI – 500k+ face images with age and gender labels. https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

[84] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, 'Deep learning face attributes in the wild", in Proceeding of International Conference on Computer Vision (ICCV).2015.

[85] https://github.com/tkarras/progressive_growing_of_gans (05/04/2022).

[86] https://github.com/NVlabs/stylega (05/04/2022).

[87] https://github.com/ajbrock/BigGAN-PyTorch (06/04/2022).

[88] https://github.com/eriklindernoren/Keras-GAN/tree/master/cogan (11/04/2022).

[89] https://www.kaggle.com/code/sayakdasgupta/fake-faces-with-dcgans/notebook (15/04/2022)

[90] https://anderfernandez.com/en/blog/how-to-code-gan-in-python (16/04/2022)

[91] Ali Borji, "Pros and Cons of GAN Evaluation Measures". CoRR abs/1802.03446. 2018.

[92] Sastre-Toral, M. Teresa "Predicción de edad y género a partir de la imagen de una persona". Master Universitario en Ciencia de Datos. UOC. 2020

9. Anexos

A continuación, mostramos ciertas partes de la programación de la red de la variante DCGAN que se basa en el ejemplo [90].

El siguiente código implementa la red generadora.

```
import keras
from keras.layers import Dense, Conv2DTranspose, LeakyReLU, Reshape,
BatchNormalization, Activation, Conv2D
from keras.models import Model, Sequential
def generador_de_imagenes():
 generador = Sequential()
 generador.add(Dense(256*4*4, input_shape = (100,)))
 generador.add(LeakyReLU())
 generador.add(Reshape((4,4,256)))
 generador.add(Conv2DTranspose(128,kernel_size=3, strides=2, padding
= "same"))
 generador.add(LeakyReLU(alpha=0.2))
 generador.add(Conv2DTranspose(128,kernel_size=3, strides=2, padding
= "same"))
 generador.add(LeakyReLU(alpha=0.2))
 generador.add(Conv2DTranspose(128,kernel_size=3, strides=2, padding
= "same"))
 generador.add(LeakyReLU(alpha=0.2))
 generador.add(Conv2DTranspose(128,kernel_size=3, strides=2, padding
= "same"))
 generador.add(LeakyReLU(alpha=0.2))
  generador.add(Conv2D(3,kernel_size=4, padding = "same", activa-
tion='tanh'))
 return(generador)
modelo_generador = generador_de_imagenes()
modelo_generador.summary()
```

El siguiente código implementa la red discriminadora.

```
from keras.layers import Conv2D, Flatten, Dropout
def discriminador_de_imagenes():
 discriminador = Sequential()
 discriminador.add(Conv2D(64, kernel_size=3, padding = "same", input_shape = (64,64,3)))
 discriminador.add(LeakyReLU(alpha=0.2))
 discriminador.add(Dropout(0.2))
 discriminador.add(Conv2D(128, kernel_size=3,strides=(2,2), padding = "same"))
 discriminador.add(LeakyReLU(alpha=0.2))
 discriminador.add(Conv2D(256, kernel_size=3, strides=(2,2), padding = "same"))
 discriminador.add(LeakyReLU(alpha=0.2))
 discriminador.add(Conv2D(256, kernel_size=3, strides=(2,2), padding = "same"))
 discriminador.add(LeakyReLU(alpha=0.2))
 discriminador.add(Conv2D(512, kernel_size=3, strides=(2,2), padding = "same"))
 discriminador.add(LeakyReLU(alpha=0.2))
 discriminador.add(Flatten())
 discriminador.add(Dropout(0.4))
 discriminador.add(Dense(1, activation='sigmoid'))
 opt = tf.keras.optimizers.Adam(learning_rate=0.0002 ,beta_1=0.5)
 discriminador.compile(loss='binary_crossentropy', optimizer= opt , metrics = ['accuracy'])
 return(discriminador)
modelo_discriminador = discriminador_de_imagenes()
modelo_discriminador.summary()
```

El siguiente código relaciona la red generadora y la discriminadora.

```
def crear_gan(discriminador, generador):
 discriminador.trainable=False
 gan = Sequential()
 gan.add(generador)
 gan.add(discriminador)
 opt = tf.keras.optimizers.Adam(learning_rate=0.0002,beta_1=0.5)
 gan.compile(loss = "binary_crossentropy", optimizer = opt)
 return gan
gan = crear_gan(modelo_discriminador,modelo_generador)
gan.summary()
```