Trabajo Final Métodos Generativos

Jorge Carrasco, Iván López e Ismael Perdomo

18 de diciembre de 2023

1 Índice

2 Introducción

El objetivo principal de este proyecto es diseñar un modelo capaz de reconocer la presencia de objetos específicos en una imagen para, posteriormente, eliminarlos. En particular, hemos enfocado nuestra atención en la tarea de reconocer y eliminar los bigotes presentes en fotografías de personas, tomando como referencia el paper *A Novel GAN-Based Network for Unmasking of Masked Face* que en época covid pretendía crear un modelo que eliminase las mascarillas al menos de manera virtual. La adaptación y amoldamiento de dicha estructura no ha sido sencilla ni mucho menos trivial, puesto que hemos cambiado, experimentado y probado distintos tipos de morfologías para cada red con el objetivo de maximizar los resultados. .

En este documento, presentaremos la arquitectura y metodología empleadas para desarrollar nuestro modelo, detallaremos el conjunto de datos utilizado en el proceso de entrenamiento, y analizaremos los resultados obtenidos.

3 Metodología

Para llevar a cabo nuestro objetivo hemos utilizado funciones para procesar imágenes del dataset y también, hemos implementado un autoencoder U-NET y una GAN.

Función *image-processing*

La función *image-processing* transforma una imagen de entrada, en nuestro caso, esta función transforma todas las imágenes del dataset, añadiéndole un bigote a la persona de cada imagen. Los bigotes son extraídos aleatoriamente de otro dataset, exclusivamente de imágenes de bigotes. Cuando finaliza crea dos nuevos datasets, en uno añade las imágenes compuestas de la concatenación de cada imagen original sin bigote y la misma imagen con la adición del bigote. En el otro dataset añade un mapa binario que indica la ubicación del bigote en blanco y el resto en negro, por cada imagen modificada con bigote.

Función *mask-point-detection*

Para colocar el bigote en la posición correcta, se ha utilizado la función *mask-point-detection*, que realiza una serie de operaciones para lograr ubicar la zona exacta donde debería ir el bigote. Primeramente, utiliza un modelo llamado *detector* que importamos de la librería *dlib*, el cual nos detecta caras en las imágenes. Posteriormente, se utiliza el *predictor* para predecir la forma facial en la región de la cara y, obtiene las coordenadas del rectángulo que rodea la cara. Se definen las coordenadas de la región de interés, en este caso la zona entre el labio y la nariz, y se dibujan unos puntos en los puntos faciales detectados y devuelve sus coordenadas.

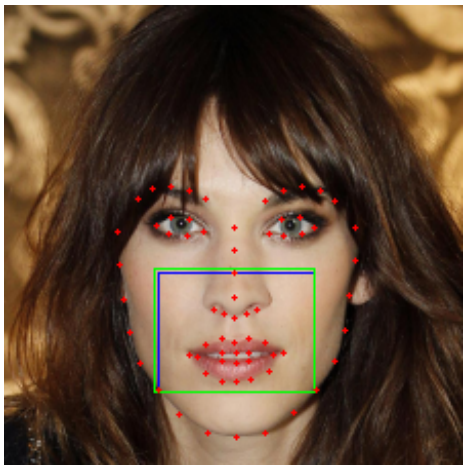


Imagen de un rostro con los puntos del rostro

Imagen de los puntos que se identifican en un rostro.

Función *noise-processing*

La función *noise-processing* recibe, por un lado, recibe la imagen de la persona sin bigote y por otro lado, recibe desde la U-NET la posición en forma de mapa de activación (blanco y negro) del bigote. Con estas dos imágenes, la función se encarga de cubrir la zona del bigote en la imagen modificada con bigote, para que esta zona se cubra de ruido. Cuando termina, le pasa directamente a la GAN la imagen para que genere la imagen sin ruido.

U-NET

La arquitectura U-NET es una red neuronal convolucional (CNN) utilizada para tareas de segmentación de imágenes. Se compone de una fase de downsampling, donde se capturan características a diferentes escalas mediante convoluciones y submuestreo, seguida por una fase de upsampling que utiliza convoluciones transpuestas para aumentar la resolución espacial. La capa de bottleneck conecta estas fases para facilitar la transferencia de información.

La idea de implementar esta U-NET viene influenciada por los papers, notebooks y artículos que tomamos como referencia [1][2][6][7], y nace como solución al problema que supone que la red tiene que aprender a detectar dónde se encuentra el bigote. La estructura de la U-NET era perfecta para este problema, puesto que ya conocíamos de sus aplicaciones en los campos de segmentación de imágenes y detección de objetos.

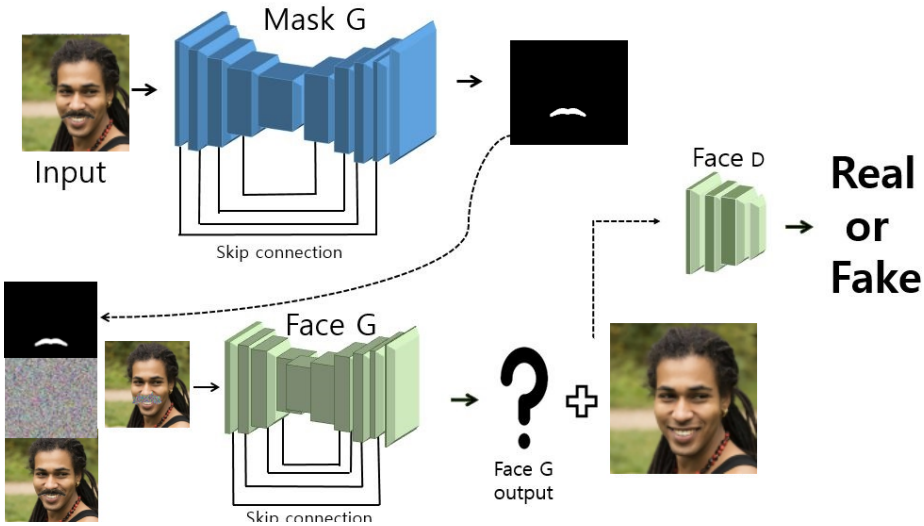
Nos hemos decantado por el uso de capas convolucionales y capas convolucionales traspuestas con strides iguales a 2, puesto que en este caso particular nos ofrecían resultados ligeramente superiores a los entrenamientos realizados con capas de *UpSampling* y *MaxPooling*.

Usamos la función de activación $ReLU$, introduciendo no linealidades en la red, en todas las capas exceptuando la última, que presenta una activación *Sigmoid* por la siguiente razón:

En este modelo, la codificación de imágenes de activación de los bigotes resulta en una imagen totalmente negra y una zona blanca (la del bigote). De usar una función de activación tangencial hiperbólica los cálculos de superposición y mapeo entre imágenes sería algo más complicado al tener un rango de valores de [-1, 1], en lugar de [0, 1].

La comodidad que en este sentido nos ofrecía la activación sigmoidal no se ha visto perjudicada de manera palpable por su menor rango de activación que la *tanh*, y ha sido suficiente para decantar la balanza a su favor, generando una máscara binaria que representa la segmentación.

El modelo se compila con el optimizador Adam y la pérdida de binary-crossentropy, preparándolo de este modo para tareas de segmentación precisa en imágenes.



Ejemplo del modelo completo con imagenes

La implementación de la U-NET en nuestro modelo no fue trabajo fácil, pues hasta dar con la estructura final nos encontramos con una serie de problemas relacionados con los valores que adquirían los píxeles al ser procesados, la definición de la función de pérdida y las dimensiones de los tensores de entrada y salida.

Estos problemas fueron solucionados poco a poco a través del estudio y la comprensión de la morfología de nuestro modelo y nuestros datos, que se vio dificultada por la naturaleza y poca variedad de las distintas referencias sobre este tipo de estructura [1][6].

GAN

Generador:

El generador de la GAN utiliza una arquitectura U-NET modificada con capas de convolución y deconvolución. La arquitectura se compone de capas de *downsampling* y *upsampling*, y utiliza capas de *Dropout* en algunos casos.

En concreto, estas capas de Dropout se encuentran en el *decoder* de la U-NET del generador con un valor de 0.5, que aleatoriamente establecen el valor de activación del 50% de las neuronas a 0, previniendo el *overfitting* y sirviendo como lastre al entrenamiento para propiciar una generalización óptima.

Además, la U-NET del generador cuenta con una estructura similar a la U-NET segmentadora de bigotes, pero que ha sido modificada ligeramente para optimizar el entrenamiento.

El generador tiene un papel crucial en la transformación de las imágenes de entrada con ruido en una imagen de salida realista sin la presencia de bigote.

La fase de codificación comienza con capas de *downsampling* que reducen progresivamente las dimensiones espaciales, capturando características a diversas escalas. La fase de decodificación utiliza capas de *upsampling* que aumentan la resolución, restaurando así los detalles y proporcionando una imagen generada final

Sin embargo, a diferencia de la anterior U-NET, esta cuenta en su última capa con una función de activación tangencial hiperbólica que, en este caso sí, nos permite gozar de sus ventajes en comparación a la sigmoidal, sin dar nada a cambio, pues no tendremos que realizar operaciones complicadas de seguir con las imágenes resultantes, sino que se generará una imagen de la persona que recibe como entrada, sin bigote.

Discriminador:

El discriminador de la GAN actúa como un clasificador binario, evaluando la autenticidad de las imágenes generadas en comparación con las imágenes reales. Compuesto por capas de convolución y *downsampling*, el discriminador busca distinguir entre imágenes auténticas y generadas.

La red comienza con la concatenación de la imagen de entrada y la imagen objetivo, creando un tensor que se somete a capas de *downsampling*. Estas capas reducen gradualmente las dimensiones espaciales y extraen características discriminativas. La introducción de capas de *padding* y técnicas de normalización refuerza la capacidad del discriminador para aprender patrones representativos.

Inmediatamente después de la definición del bloque de *downsample* se introduce una capa de *BatchNormalization* que permite normalizar los valores de activación de las capas ocultas, mejorando la estabilización y eficiencia del entrenamiento, con un valor por defecto del momentum del 0.99, es decir, que las medias y varianzas anteriores no tienen un gran impacto en la estimación actual de activación. A continuación, presenta una capa de activación $LeakyReLU$ para mitigar los desvanecimientos de gradiente, pasándole un alpha no muy elevado, 0.3 por defecto.

El discriminador concluye con una capa de convolución que produce un mapa de activación unidimensional. Este mapa refleja la autenticidad de la imagen, con valores más altos indicando imágenes más auténticas y viceversa.

Esta arquitectura discriminatoria se integra con el generador en el marco del entrenamiento adversarial, donde ambos modelos se optimizan simultáneamente para mejorar la calidad de las imágenes generadas.

4 Conjunto de Datos

Para empezar hemos cargado un dataset de 1071 imágenes de tamaño 512 x 512. Este conjunto contiene fotografías de personas con diversidad de expresiones y características faciales. Estas imágenes han sido modificadas para cambiar su tamaño a 224 x 224 píxeles. Por otro lado, también hay un dataset con 5 imágenes de bigotes de diferentes resolución, pero se les hace un resize a 224x224 píxeles.

Posteriormente se ejecuta la función *image-processing*, a la cual le entra, por cada interacción, una imagen del dataset de imagen y una imagen de bigote de su respectivo dataset para combinarlo. Como salida producirá los dos datasets de imágenes de 224x224 píxeles, indicados anteriormente.

Imagen de salida de la función *image-processing*

Mapeo de salida de la función *image-processing*

Una vez tenemos las imágenes con bigote, las pasamos como input a una U-NET y como etiqueta el mapa del bigote para que la U-NET se entrene con estas imágenes y sepa detectar en qué posición se encuentra el bigote. Una vez ha terminado, coge y le pasa el mapa de activación que ha generado para cada imagen a la GAN.

Para pasar las imágenes con bigote a la GAN, primero utilizamos la función *noise-processing* que añade ruido en la zona del bigote de la imagen de la cara. Este proceso se realiza del siguiente modo: la función recibe una imagen de un mapa de activación del bigote(mapeo) y la imagen de la persona con el bigote. Entonces, una vez ha recibido estas imágenes, añade ruido en la imagen de la persona en el punto del mapa de activación. Es decir, se añadirá ruido solo en la zona del bigote. Cuando finaliza este proceso, pasa las imágenes con el ruido en el bigote a la GAN.

Imagen de persona con la sección del bigote con ruido

La GAN coge como entrada las imágenes de una persona con ruido en el bigote y la etiqueta de cada imagen, que es el mapa de activación del bigote. Con estas dos entradas la entrada GAN se entrena para devolver una imagen generada, habiendo eliminado el bigote.

5 Entrenamiento

Entrenamiento de la U-NET

La U-NET se entrena a lo largo de 100 épocas utilizando el algoritmo de optimización *Adam* y la función de pérdida de *binary-crossentropy*. Utilizamos *PlotLearning* para registrar y visualizar en tiempo real las métricas críticas.

Al final de cada época, se selecciona aleatoriamente una imagen de prueba del conjunto de validación para mostrar la imagen original junto con la máscara segmentada generada por la U-NET.

Este enfoque de entrenamiento monitorizado permite que la U-NET aprenda a identificar con precisión la región del bigote en las imágenes de entrada.

Entrenamiento de la GAN

Durante cada una de las 30 épocas durante las que se entrena la GAN, esta sigue un flujo de trabajo que abarca desde la generación de imágenes iniciales hasta la evaluación y mejora de su desempeño. Aquí se destaca el papel clave de dos componentes principales: el generador y el discriminador.

En cada iteración, el generador utiliza la información de entrada en forma de máscaras para predecir imágenes iniciales. Estas imágenes generadas se someten a un procesamiento adicional mediante la función 'noise-processing'. El discriminador, por otro lado, evalúa la autenticidad de las imágenes generadas en comparación con las imágenes reales del conjunto de datos. Este proceso competitivo entre el generador y el discriminador impulsa la mejora continua de ambos componentes.

La visualización de los resultados ocurre cada 100 pasos, ofreciendo una visión rápida del progreso del generador y permitiendo ajustes según sea necesario.

Como parte del monitoreo del progreso, se guardan puntos de control de los modelos cada 200 pasos. Estos puntos de control representan versiones grabadas del generador y el discriminador, lo que facilita la recuperación o la reanudación del entrenamiento en caso de interrupciones.

En resumen, este código recoge el proceso de entrenamiento de una GAN, destacando la dinámica competitiva entre el generador y el discriminador para lograr la generación de imágenes realistas a partir de máscaras de entrada.

6 Resultados

Resultados de la U-NET

La implementación de la arquitectura U-NET ha sido efectiva en la generación de imágenes de mapeo blanco y negro de la zona del bigote. Durante las primeras etapas del entrenamiento, las imágenes resultantes presentaban una calidad inferior, con un mapeo del bigote menos preciso. Sin embargo, a medida que el modelo se sometía a más épocas de entrenamiento, se observó una mejora significativa en la capacidad de la U-NET para identificar con precisión la región del bigote en las imágenes, generando mapas más definidos y detallados.

Ejemplo de resultado de la U-NET en la época nº 6

Resultados de la GAN

En cuanto a la GAN, los resultados obtenidos indican una tendencia similar de mejora progresiva a lo largo del entrenamiento. Durante las primeras épocas, las imágenes generadas muestran completamente el ruido añadido en la zona del bigote, siendo al principio imperceptible el cambio. No obstante, con el tiempo, la GAN demuestra su capacidad para aprender y eliminar la región del bigote con ruido, generando imágenes sin bigote algo más realistas, mostrando una tendencia al *overfitting*.

Es importante destacar que, aunque los resultados finales de la GAN pueden no alcanzar la perfección debido a limitaciones de tiempo y de recursos, la evolución positiva observada durante el entrenamiento destaca el potencial de esta metodología para la generación de imágenes de personas sin bigote de manera correcta.

Estos resultados sugieren que el uso combinado de la U-NET y la GAN puede ser una buena estrategia para eliminar de forma artificial elementos diversos en imágenes, con mejoras notables a medida que ambos modelos se benefician del entrenamiento continuo, siendo una posible y simpática aplicación para eliminar de forma artificial el bigote.

Ejemplo de un resultado de la GAN

7 Discusión y Conclusiones

El modelo desarrollado nos ha dado unos resultados buenos, aunque las pruebas de Test indiquen *overfitting*. De manera sencilla, a partir del dataset original, añadimos el bigote correctamente a las imágenes. Usando estas imágenes modificadas con bigotes, se ha conseguido añadir ruido en la zona del bigote de las imágenes para después poder ser eliminado del todo, por lo que se obtienen resultados correctos.

Durante el desarrollo del modelo nos hemos encontrado diversos desafíos y problemas, como los que han sido descritos a lo largo del informe. Al principio, por ejemplo, tuvimos algunos pequeños problemas porque no nos colocaba bien los bigotes en las caras, pero se solucionó fácilmente cambiando la configuración de la función que detecta las coordenadas dónde se ubica el bigote, usando otros puntos de identificación en un rostro. Después de esto, nos colocaba bien el bigote incluso en caras que estaban de lado.

Desarrollar este modelo ha sido un reto para los 3 integrantes de nuestro grupo, debido al escaso tiempo disponible para la realización del mismo, viéndose aún más acortado por un cambio de idea propiciado por factores no controlables como la disponibilidad de recursos. Sin embargo, hemos conseguido sacarlo adelante llevándonos una gran experiencia a raíz de la realización del mismo.

8 Referencias

[1] N. Ud Din, K. Javed, S. Bae and J. Yi, "A Novel GAN-Based Network for Unmasking of Masked Face," in IEEE Access, vol. 8, pp. 44276-44287, 2020, doi: 10.1109/ACCESS.2020.2977386.

[2] Google Colab, U-Net

[3] Image segmentation with a U-Net-like architecture. (s/f). Keras.Io.

[4] Shape$_predictor_68_face_landmarks.Datasdasdasdasd/face-$

forgery-detection at ccfc24642e0210d4d885bc7b3dbc9a68ed948ad6. (s/f). Huggingface.co.

[5] Heydarian, A. (2021, octubre 3). U-net for semantic segmentation on unbalanced aerial imagery. Towards Data Science.

[6] deokjin. (s/f). GAN-based-face-mask-removal: This project is to restore the face hidden behind the mask in the image of the person wearing the mask.

[7] Rosebrock, A. (2017, abril 3). Facial landmarks with dlib, OpenCV, and Python. PyImageSearch.

[8] Mantas. (2023). Face Attributes Grouped [Data set].

9 Anexos

Se incluye un modelo que comenzamos a desarrollar pero finalmente no se pudo continuar, puesto que necesitaba demasiados recursos para su procesamiento y no disponíamos de ellos. Este modelo, era un modelo ya creado que nosotros modificamos, cambiando el Fine Tunning para que entrenara como nosotros queríamos, pudiendo crear cuadros como deseábamos a partir de un texto. El problema principal era la falta de espacio en la RAM del sistema dado a que se quedaba sin espacio por eso, intentamos reducir el tamaño del modelo y de los dataset, pero todos los intentos resultaron en vano, por lo que decidimos abortarlo y comenzar con una práctica distinta que pudiese ser desarrollada sin tal necesidad de recursos, para poder ejecutarla sin problemas. En este modelo se realizaba *fine tunning* sobre un modelo de *stable diffusion*, con el que buscábamos generar imágenes de cuadros a partir de texto. Consideramos buena idea aportarlo debido a que nos llevó mucho tiempo, y la implementación estaba prácticamente terminada.

Modelo Stable Diffusion + Fine Tunning