Implementación de una GAN para la reconstrucción de imágenes

Alejandro Cortijo Benito br0163
Diego Cruz Cuervo br0125
Álvaro Karim Fernández Rubio br0342

Resumen

La necesidad de mejorar la calidad de las imágenes médicas para conseguir mejores diagnósticos ha incrementado a lo largo de los años. Los avances tecnológicos en el área del Deep Learning han hecho posible la aparición de nuevos métodos capaces de realizar tareas que parecían impensables unos años atrás, y además obteniendo unos grandes resultados.

Con esta aplicación buscamos reconstruir las imágenes tomadas durante las radiografías, mejorando la percepción de las mismas centrándonos en las características más importantes. Muchas organizaciones están empezando a usar herramientas de IA para ayudar a los radiólogos a leer las imágenes con mayor rapidez y precisión, mejorar de la precisión en el posicionamiento del paciente y la reconstrucción de imágenes de TC [10] [9] o eliminar la complejidad de las mediciones por ultrasonidos.

Este enfoque tecnológico promete simplificar procesos, ofrecer resultados más confiables y avanzar hacia una práctica médica más eficiente y precisa.

1. Introducción

La compleja tarea de mejorar la calidad y/o tamaño de, generalmente una imagen, de baja resolución (LR) a una de alta resolución (HR) con un factor de aumento de 2x, 4x, o más manteniendo o mejorando la calidad de la imagen original, es llamada superresolución (SR). En los últimos años, esta tarea ha mostrado utilidad, por ejemplo, en el área médica para mejorar la calidad de las imágenes obtenidas por resonancia magnética con el beneficio de tener que dar al paciente una menor cantidad de radiación o ayudar a los médicos a detectar diagnósticos que de otra manera podrían ser difíciles de ver. Por otra parte, también ha probado ser importante en la toma de imágenes desde satélites en órbita para mejorar la calidad de los mapas o altitud del terreno.

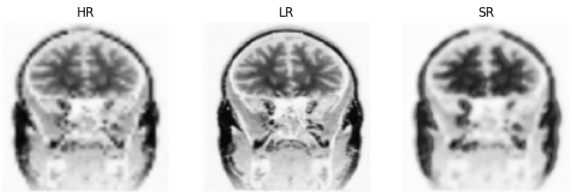

Figura 1: Comparación entre imágenes

Esto ha probado ser un desafío para el campo del Machine Learning, con diferentes aproximaciones y arquitecturas destinadas a conseguir SR sin cambiar la información de la imagen o introducir datos que no sean coherentes. En este trabajo, hemos intentado construir una red neuronal con arquitectura GAN que permita realizar un aumento de 3x con respecto a un dataset de imágenes de MRIs cerebrales y torsales y comparamos los resultados con nuestra arquitectura de referencia, SRGAN, con el SOTA de diferentes arquitecturas GAN, Transformer y modelos basados en difusión.

1.1. Investigaciones relacionadas

1.1.1. Arquitecturas GAN

Las arquitecturas GAN más modernas para superresolución son SRGAN [1], ESRGAN [4] y Real-ESRGAN [5]. Hemos consultado las tres para realizar nuestros experimentos, pero por límites computacionales hemos acabando tomando como referencia ESRGAN, ya que Real-ESRGAN requiere muchos más recursos.

1.1.2. Arquitecturas Transformer

Uno de las arquitecturas más interesantes en los últimos años ha sido basadas en Transformers es SwinIR [3]. Para nuestros experimentos hemos realizado comparaciones, pero nos ha sido imposible entrenar una arquitectura parecida por razones

de tiempo.

1.1.3. Arquitecturas de difusión

Las aproximaciones al problema de resolución desde el punto de vista de arquitecturas de difusión son relativamente recientes, surgiendo en en año anterior una arquitectura prometedora llamada ResShift [6]. Por la misma razón que SwinIR, no hemos podido continuar por esa línea debido a razones de potencia computacional y tiempo, pero la hemos tenido en cuenta para realizar comparaciones con nuestra arquitectura.

1.1.4. Data Augmentation

Data Augmentation ha sido una técnica usada más frecuentemente para clasificación y otros modelos generativos que para SR. A la hora de buscar pipelines y técnicas de Data Augmentation específicas para el dominio de SR nos hemos encontrado con CutBlur [2]. Nos habría gustado probarla pero no hemos sido capaces por problemas de implementación.

2. Método

2.1. Arquitectura de referencia

En un principio consideramos tomar como referencia la arquitectura Real-ESRGAN, pero no encontramos documentos ni papers que nos clarificasen la arquitectura que utiliza, e igualmente con las de la ESRGAN. Adicionalmente, en las comparaciones que hacían en papers las tres nos parecía que tenían unos resultados aceptables para los experimentos que queríamos realizar. Por otra parte, la arquitectura SRGAN tenía detallada su arquitectura en el paper y proveían un código más claro que nos permitió implementar las funcionalidades que queríamos.

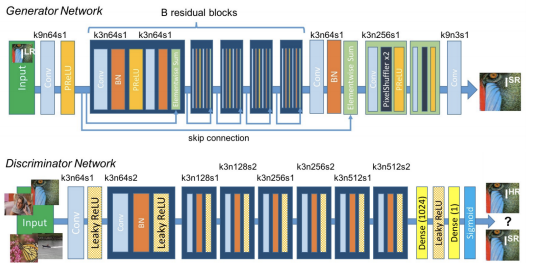

Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

Figura 2: Arquitectura de SRGAN

2.2. Nuestra arquitectura

Inicialmente hicimos un baseline con una arquitectura parecida a la que hicimos en clase con GANs, con la que pudimos reconstruir en gran medida del dataset MNIST y Fashion MNIST, pero se quedaba corta para imágenes con más tamaño y más complejas. Despues de cambiar nuestra arquitectura a la de SRGAN vimos que tuvimos problemas con la complejidad computacional ya que tardaba mucho tiempo en ejecutarse. Para lidiar con este problema, cambiamos tanto el generador como el discriminador, incluyendo Dropouts en el discriminador y reduciendo el número de bloques residuales en el generador. Aparte de esto, también modificamos un número de capas convolucionales y su número de stride o filtros para que se adaptasen a nuestras necesidades. El entrenamiento lo hemos llevado a cabo en GPUs T4, V100 y A100 en google colab con 1957 imágenes.

2.3. Generador

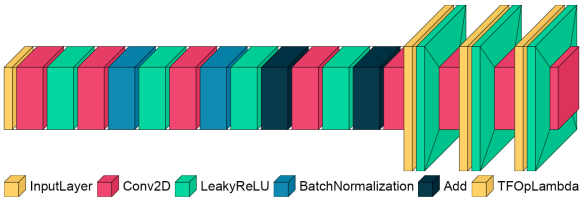

Figura 3: Arquitectura del generador

El generador tiene una capa de input-conv2d-leakyRelu tras lo cual tiene siete bloques conv2d-BN-leakyRelu-conv2d-BN-leakyRelu-Add y conv2d-leakyRelu con una skip connection desde la primera capa. Finalmente tenemos tres bloques conv2d-pixelShuffle x2-leakyRelu y una salida con función de activación tangencial y un solo filtro.

2.4. Discriminador

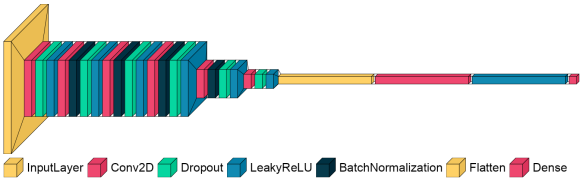

Figura 4: Arquitectura del discriminador

El discriminador empieza con una capa de input seguida por capas conv2d-dropout 0.5-leakyRelu y 3 bloques conv2d-BN-dropout 0.5. Después

de esto tiene tres bloques conv2D-dropout 0.5-leakyRelu donde el primer bloque tiene una capa de BN antes del dropout. Finalmente, tiene un flatten y una capa densa con 1024 neuronas activada con una leakyRelu antes de la salida con función de activación sigmoidal tanto para generar una mayor cantidad de datos como para adherirse a la normalización [-1, 1] que hemos realizado a las imágenes.

2.5. Preprocesamiento de las imágenes

Para preprocesar las imágenes hemos intentado varias aproximaciones como reducirlas por un factor de disminución de tamaño dependiente de su imagen o la introducción de ruido gaussiano. Finalmente no nos han funcionado bien y nos hemos decantado por hacer un crop de MxM donde M=min(W, H) y la reducción de la imagen por un factor común hasta llegar a un tamaño de 50x50, tras lo cual las volvemos a redimensionar a un tamaño fijo de 150x150. Aparte de esto, también hemos normalizado las imágenes en un rango [-1, 1].

2.5.1. Data Augmentation

No nos ha funcionado ninguna aproximación usando data augmentation, a pesar de que pudieran parecer intuitivas para conseguir un mejor resultado. Hemos intentado introducir pipelines incluyendo crops, flips introducción de ruido o redimensionamiento con probabilidades diversas. Finalmente hemos decidido no utilizar esta técnica en nuestro entrenamiento porque además ya disponíamos de un número de imágenes que creíamos suficientes para nuestros experimentos.

2.6. Función de pérdida

Para la función de pédida hemos utilizado el Mean Square Error (MSE) en el generador y Binary Crossentropy en el discriminador. Nuestra arquitectura de referencia usaba una función de pérdida basada en MSE y VGG con el objetivo de detectar y reducir la diferencia perceptual entre las imágenes de SR y HR. Nos habría gustado probarla, pero no hemos podido encontrar recursos disponibles que nos dejasen claro como hacerlo de forma correcta.

3. Experimentos

3.1. Baseline

Inicialmente implementamos un modelo basándonos en la ACGAN y DCGAN de la práctica 2 de la asignatura, usando el dataset MNIST como hemos mencionado anteriormente. Aquí estábamos más limitados, debido al tamaño y calidad de las imágenes. Para generar las imágenes de mala calidad, reducimos el tamaño de las mismas para posteriormente aumentarlo y trabajar con imágenes de 28x28, los resultados no fueron malos.

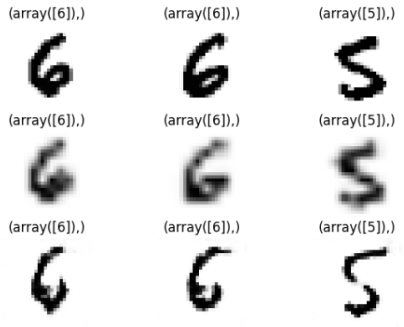

Figura 5: Predicción ACGAN

Las imágenes de alta resolución están en la primera fila, la segunda fila son las imágenes de baja calidad, y la última fila muestra las generadas. La ACGAN ha conseguido mejorar las imágenes de mala calidad, es un buen comienzo.

A diferencia de este caso, tuvimos un problema con el Fashion MNIST, ya que no obtuvimos los resultados esperados, el modelo no fue capaz de generar bien las imágenes. Aquí nos dimos cuenta de que necesitábamos una arquitectura más compleja.

3.2. Pasos previos

En esta versión cambiamosos el conjunto de datos, introducimos imágenes de datasets clínicos [7] [8] procedentes de kaggle, y actualizamos la arquitectura de la GAN en base a la arquitectura SR-GAN de referencia.

Hubo varios problemas para cargar todas las imágenes, tuvimos que reducir el tamaño hasta que el modelo fuera capaz de ejecutarse, además de dejar en escala de grises las imágenes para reducir la dimensionalidad y que sea menos costoso el entrenamiento. Finalmente, adaptando el modelo conseguimos que cada epoch tardara en ejecutarse alrededor del minuto, usando imágenes de 100x100 y estos fueron los mejores resultados.

Al igual que con el baseline la primera fila representa las imágenes de buena calidad, la segunda fila muestra las imágenes de baja resolución y la última las generadas por nuestro modelo.

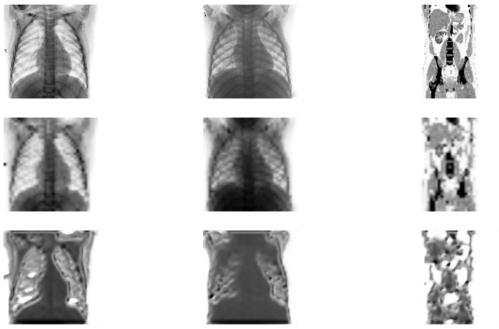

Figura 6: Iteración 1

3.3. Mejoras

En base a estas predicciones modificamos manualmente el número de filtros, capas y estrategias de regulación para tratar de obtener mejores resultados. Nos dimos cuenta de que el discriminador era superior al generador, y esto hacía que las generaciones fueran de peor calidad.

Intentamos igualar las fuerzas en este juego de suma 0 para mejorar el modelo y poco a poco fuimos viendo mejores métricas y generaciones. El dominio del discriminador era menos pronunciado y empezó a bajar su precisión considerablemente.

3.4. Prototipo final

Conseguimos maximizar la ejecución de nuestras imágenes, llegando a procesar imágenes de 150x150 con epoch inferiores a las 2 minutos.

Finalmente conseguimos que generador diera mejores resultados, no tan buenos como para generar imágenes de mejor calidad a las de alta calidad pero si para competir con las de baja calidad siendo capaz de mejorlas en varias ocasiones.

4. Discusión y trabajo futuro

En este trabajo hemos explorado múltiples arquitecturas tanto GAN, como Transformers y modelos de Difusión. Finalmente nos hemos decidido, como hemos comentario anteriormente, por usar un modelo GAN dadas las limitaciones que teníamos y la satisfacción con respecto a los resultados de modelos GAN SOTA con una arqutiectura relativamente replicable. Mirando a los resultados de la **Figura 7** podemos llegar a un número de conclusiones con respecto al resultado de nuestro modelo. Por una parte, podemos llegar a la conclusión de que los resultados de la superresolución con el conjunto de datos que hemos usado para entrenar el modelo, tanto en nuestro como en otros modelos modernos, es mala. Esto es probablemente dado a que las imágenes con procedencia de MRIs tienen detalles que precisan de un entrenamiento exhaustivo con datos muy específicos, y no otros datos que hagan que el modelo pueda generalizar bien en este dominio en particular. Por otra parte, podemos ver que los resultados son más oscuros de lo que esperabamos. Uno de los problemas que hemos tenido es que en múltiples ocasiones, las imágenes cambiaban completamente a negro tras 20-50 epochs. A pesar de que hemos solucionado esto parcialmente, se puede ver que el modelo no ha sido capaz de replicar con la suficiente capacidad la distribución de datos del dataset original, y probablemente habría que entrenarlo durante miles de epochs (en nuestro paper principal de referencia, por ejemplo, lo entrenaban durante 1500 epochs), mientras que nosotros solo hemos podido entrenarlo durante 50-100 epochs. Otras cosas que nos gustaría probar sería funciones de pérdida más sofisticadas que sean capaces de detectar más eficientemente la similaridad relativa de las imágenes generadas con las imágenes con alta resolución.

5. Conclusiones

Uno de los grandes obstáculos que hemos tenido ha sido conseguir acceso a más potencia de cómputo para realizar entrenamientos con imágenes más grandes y/o con más detalle. Hemos estado limitados a usar Nvidia V100 y T4 la mayoría del tiempo, así que hemos entrenado nuestro modelo en escala de grises y con imágenes de 150x150, que no es demasiado, y es posible que hayamos perdido información importante del dataset original para replicar su distribución. En general y a pesar de los obstáculos que hemos tenido, pensamos que el resultado obtenido, aunque mejorable, no ha sido malo considerando nuestra situación.

Referencias

[1] Christian Ledig et al. «Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network». En: (2017). URL: https://arxiv.org/abs/1609.04802.

[2] Jaejun Yoo et al. «Rethinking Data Augmentation for Image Super-resolution: A Comprehensive Analysis and a New Stra-

tegy». En: (2020). URL: https://arxiv.org/abs/2004.00448.

[3] Jingyun Liang et al. «SwinIR: Image Restoration Using Swin Transformer». En: (2021). URL: https://arxiv.org/abs/2108.10257.

[4] Xintao Wang et al. «ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks». En: (2018). URL: https://arxiv.org/abs/1809.00219.

[5] Xintao Wang et al. «Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data». En: (2021). URL: https://arxiv.org/abs/2107.10833.

[6] Zongsheng Yue et al. «ResShift: Efficient Diffusion Model for Image Super-resolution by Residual Shifting». En: (2023). URL: https://arxiv.org/abs/2307.12348.

[7] Paul Mooney. «Chest X-Ray Images (Pneumonia)». En: (2017). URL: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia.

[8] Masoud Nickparvar. «Brain Tumor MRI Dataset». En: (2021). URL: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset.

[9] María Pérez Martínez. «Diseño e implementación de una red neuronal artificial para la mejora de la resolución de imágenes de RMN cerebral». En: (2023). URL: https://riunet.upv.es/bitstream/handle/10251/196259/Perez%20-%20Diseno%20e%20implementacion%20de%20una%20red%20neuronal%20artificial%20para%20la%20mejora%20de%20la%20resolucion%20de...pdf?sequence=1&isAllowed=y.

[10] Computed Tomography Philips. «AI for significantly lower dose and improved image quality». En: (2021). URL: https://www.philips.com/c-dam/b2bhc/master/resource-catalog/landing/precise-suite/incisive_precise_image.pdf?_ga=2.217607062.1889539369.1702748130-582513495.1702748130.

Anexo. Comparativa de resultados

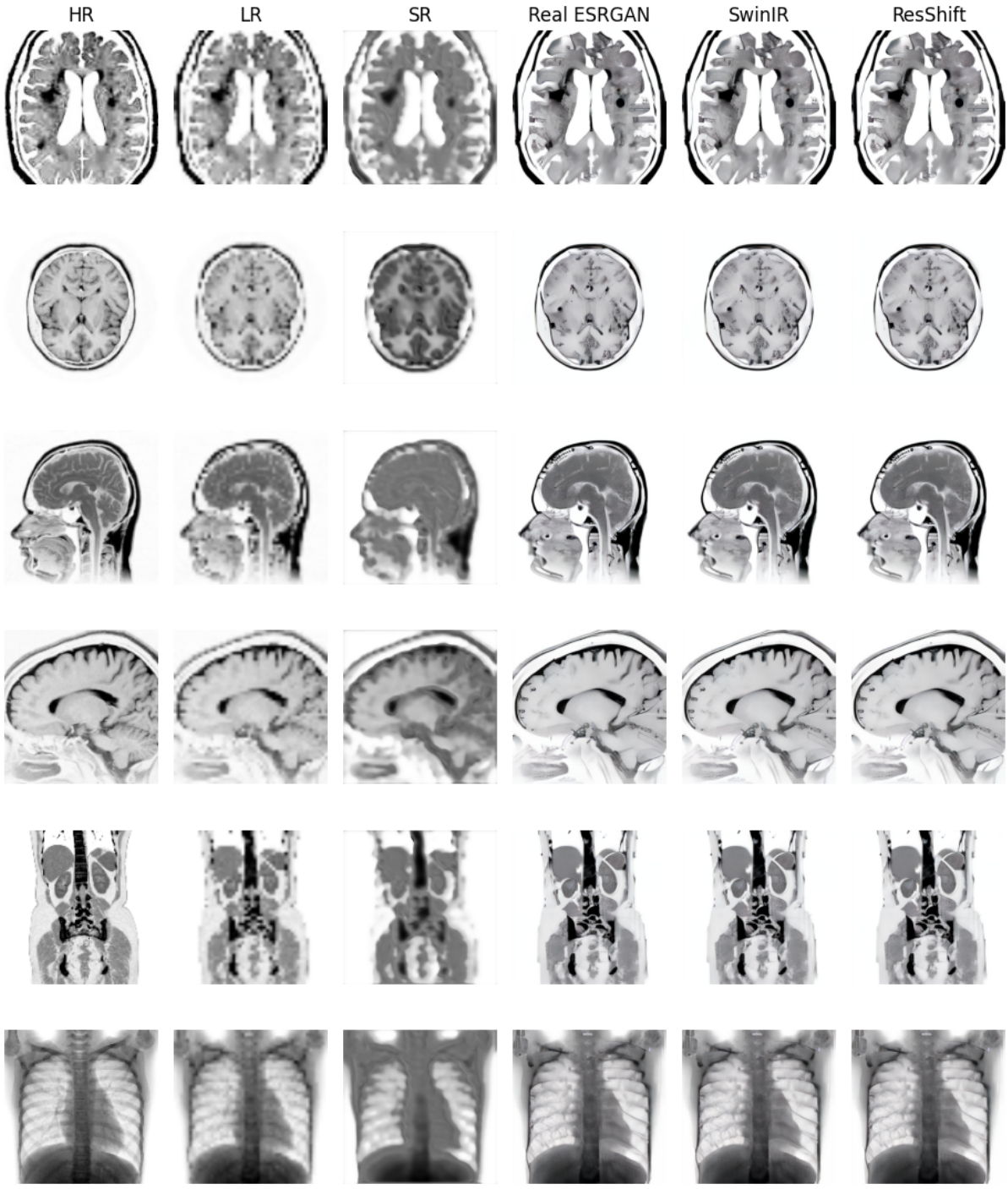

Figura 7: Comparación con otros modelos