UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Redes Generativas Antagónicas para un problema desbalanceado de clasificación de Carcinoma Ductal Invasivo

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Arias Moncho, Jose

Tutor/a: Paredes Palacios, Roberto

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Redes Generativas Antagónicas para un problema desbalanceado de clasificación de Carcinoma Ductal Invasivo

TRABAJO FIN DE MASTER

Master Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Jose Arias Moncho
Director: Roberto Paredes Palacios

Curso 2021-2022

Resum

La classificació d'imatges en àmbits mèdics és una àrea bastant prometedora, a causa de l'interessant que seria utilitzar aquests sistemes per a ajudar a professionals, alleugerint el seu treball a partir de l'automatització d'algunes tasques. No obstant això, un dels inconvenients que existeixen a l'hora d'entrenar un model de classificació en aquesta àrea és el desequilibri existent entre les imatges donades. En altres paraules, normalment es disposa de més imatges de subjectes sans que malalts.

En aquest projecte, particularment ens centrem en la detecció de teixit cancerigen (carcinoma ductal invasiu). D'aquesta manera, utilitzarem Xarxes Generatives Antagòniques (GAN) com un generador d'imatges de teixit cancerigen i sa amb l'objectiu de crear mostres sintètiques per a reentrenar els models disponibles amb aquestes mostres i millorar els seus resultats de classificació. Amb això en ment, primer utilitzarem xarxes neuronals per a obtindre resultats amb el conjunt real de dades de partida, seguidament entrenarem un parell de GANs per a generar imatges de tots dos tipus (sanes i cancerígenes) i, finalment, avaluarem si generant mostres amb aquest enfocament millora els resultats prèviament obtinguts.

Paraules clau: Xarxes Generatives Antagòniques - Classificació de imatges - Classificació de càncer

Resumen

La clasificación de imágenes en ámbitos médicos es un área bastante prometedora, debido a lo interesante que sería utilizar estos sistemas para ayudar a profesionales, aliviando su trabajo a partir de la automatización de algunas tareas. Sin embargo, uno de los inconvenientes que existen a la hora de entrenar un modelo de clasificación en esta área es el desbalance existente entre las imágenes dadas. En otras palabras, normalmente se dispone de más imágenes de sujetos sanos que enfermos.

En este proyecto, particularmente nos centramos en la detección de tejido cancerígeno (carcinoma ductal invasivo). De esta forma, vamos a utilizar Redes Generativas Antagónicas (GAN) como un generador de imágenes de tejido cancerígeno y sano con el objetivo de crear muestras sintéticas para reentrenar los modelos disponibles con estas muestras y mejorar sus resultados de clasificación. Con esto en mente, primero utilizaremos redes neuronales para obtener resultados con el conjunto real de datos de partida, seguidamente entrenaremos un par de GANs para generar imágenes de ambos tipos (sanas y cancerígenas) y, finalmente, evaluaremos si generando muestras con este enfoque mejora los resultados previamente obtenidos.

Palabras clave: Redes Generativas Antagónicas - Clasificación de imágenes - Clasificación de cáncer

Abstract

Image classification in medical fields is a very promising area, due to how interesting it would be to use these systems to help professionals, relieving their work by automating some tasks. However, one of the drawbacks when training a classification model in this area is the imbalance between the given images. In other words, there are usually more images of healthy subjects available than diseased ones.

In this project, we particularly focus on the detection of cancerous tissue (invasive ductal carcinoma). Thus, we are going to use Generative Adversarial Networks (GAN) as an image generator of cancerous and healthy tissue with the goal of creating synthetic samples to retrain the available models with these samples and improve their classification results. With this in mind, we will first use neural networks to obtain results with the real starting data set, then we will train a pair of GANs to generate images of both types (healthy and cancerous) and, finally, we will evaluate whether generating samples with this approach improves the previously obtained results.

Key words: Generative Adversial Networks - Image Classification - Cancer Classification

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

La inteligencia artificial está cada día introduciéndose más y más en todos los ámbitos de nuestra vida. Uno de ellos es la medicina, una ciencia muy antigua para la que hoy en día están surgiendo mejoras y avances de la mano de este campo tan novedoso como es la inteligencia artificial. Actualmente, el desarrollo de fármacos, el análisis de planes de salud y la consulta digital son solo algunos de los campos para los que se está utilizando estas técnicas [1].

Otro de los campos de la medicina en los que la inteligencia artificial está siendo muy relevante es en el diagnóstico de enfermedades. Específicamente, la radiología es el campo que más ha aprovechado estas nuevas técnicas para mejorar en diferentes aspectos. Según [6], las mejoras varían desde las radiografías torácicas para la detección de cáncer de pulmón hasta las radiografías cerebrales. Y es dentro de todas estas aplicaciones donde se encuentra el área de la medicina en la que centraremos este trabajo: el cáncer de mama.

El cáncer de mama es el más común en las mujeres españolas, por delante del cáncer colorrectal, de útero, de pulmón y de ovario. En concreto, las estadísticas indican que 1 de cada 8 mujeres en España desarrollan cáncer de mama y es una de las principales causas de muerte en mujeres en España, haciendo que su detección temprana sea vital para asegurar la supervivencia de las pacientes.

Los casos de cáncer de mama pueden separarse por tipos haciendo una clasificación morfológica, de forma que tenemos dos tipos: el primero se desarrolla a partir de tejido epitelial y son los carcinomas y el segundo se denomina sarcoma. Este primer tipo se origina en el tejido glandular de la mama, en concreto en los lobulillos, gándulas donde se produce la leche, y en los conductos galactóforos, que transportan esta leche hasta el pezón durante la lactancia materna. El segundo tipo es mucho menos frecuente y se desarrolla a partir de células musculares, grasa o tejido conectivo.

Centrándonos en los carcinomas, tenemos los tres tipos más frecuentes: carcinoma ductal invasivo o infiltrante (CDI), carcinoma lobulillar invasivo o infiltrante (CLI) y carcinoma ductal "in situ" o carcinoma intraductal (CDIS). En este trabajo nos centraremos en el CDI, que supone entre un 70 y un 80 % de todos los casos de cáncer de mama, mientras que el segundo más común (CLI) solo supone un 5-7 %.

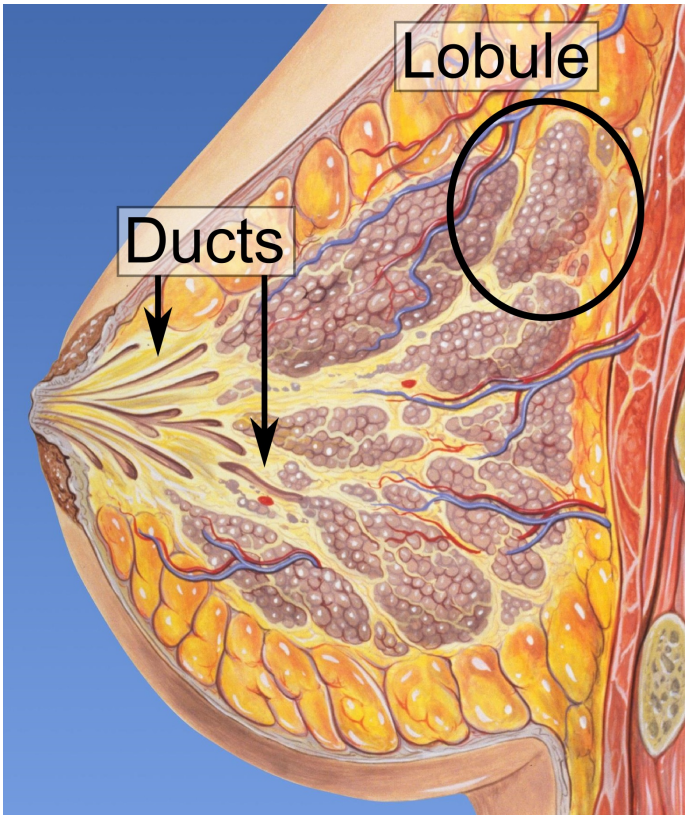


Figura 1.1: Imagen del tejido que podemos encontrar en una mama.

Como podemos ver en la figura 1.1, tenemos una imagen de un pecho sano en el que encontramos las zonas previamente descritas. En concreto, el CDI se desarrollaría en las células de los ductales, específicamente dentro de los conductos galactóforos (los conductos que salen del pezón), y se podría extender e invadir otros tejidos adyacentes de la mama, pudiendo diseminarse hacia los ganglios linfáticos y otras zonas del cuerpo.

En concreto, el procedimiento a seguir por los médicos para detectar este cáncer es utilizar una o varias de las siguientes técnicas: análisis exploratorio de la mama, mamografía y/o biopsia. Una vez diagnosticado este carcinoma con alguno de estos métodos, se debe separar las zonas de tejido sano y zonas donde se encuentren células cancerígenas, para así poder determinar entre otros factores en qué estadio se encuentra el cáncer. Este proceso debe realizarse por un profesional, y será un proceso lento y manual que puede verse acelerado si se utilizan modelos con buenos resultados para poder descartar zonas que claramente se consideren sanas o otras metodologías que permitieran semi automatizar estos procesos.

Estos modelos tendrían que poder realizar segmentaciones en estas imágenes del pecho, o también podrían ser entrenados con estas imágenes segmentadas para que pudieran trabajar con zonas locales de la imagen del pecho, es decir, con parches de la mamografía. Esto supone que para cada paciente se generaran una gran cantidad de parches, los cuales servirán para entrenar a estos modelos a clasificar correctamente estas zonas de forma local, pudiendo segmentar aquellas zonas en las que sí pudiera encontrarse zonas con carcinomas o en las que la confianza del modelo no fuera elevada.

Una de las dificultades que tenemos en este ámbito sería el desbalanceo de los datos, ya que la cantidad de parches de los que se dispone será generalmente mayor de tejido sano que de tejido con carcinomas, lo que dificulta el aprendizaje de la red. En caso de no entrenar correctamente este modelo, podríamos encontrarnos con problemas típicos al tener conjuntos de datos con desbalanceo, como que el modelo se quedara en un mínimo

local en el que prefiere no arriesgarse e indicar que todas las muestras que ve pertenecen a la clase mayoritaria con la que se ha realizado los experimentos.

Para evitar esta problemática existen diferentes enfoques, dentro de los cuales vamos a centrarnos en el aumento de datos o data augmentation, específicamente por medio de técnicas adversiales, dentro de las cuales destacan las Redes Generativas Antagónicas o Adversiales (GAN) [3]. Estas redes, una vez se han entrenado para un dominio concreto de forma exitosa, permiten generar muestras según se hayan entrenado, permitiendo obtener infinitas muestras sintéticas, que pueden tener una calidad bastante elevada y asemejarse de forma muy cercana a las muestras reales con las que se han entrenado.

Una vez hemos introducido el CDI, su presencia en la sociedad actual y la metodología para detectarlo y segmentarlo, además de las dificultades que surgen con conjuntos de datos desbalanceados y el enfoque que le daremos a este trabajo para superar esta problemática con las GAN, vamos a comentar qué ha motivado la realización de este trabajo.

1.1 Motivación

El día en que estábamos en clase de Redes Neuronales Artificiales y dimos el Tema 4 de Reinforcement Learning and Generative Models y escuché el concepto de Generative Adversial Networks, en concreto una CycleGAN, supe que era algo muy interesante y llamativo. Me pareció una idea brillante, la de generar imágenes sintéticas con un modelo gracias al feedback de otro y me interesó mucho. Le comenté a Roberto la posibilidad de realizar un TFM con esta arquitectura y después de comentarlo decidimos centrarnos en el ámbito médico. Tras buscar diferentes conjuntos de datos acabé encontrando este, en el que teníamos una gran cantidad de datos, pero estaba bastante desbalanceado y me pareció interesante intentar utilizar una GAN como método para balancear el conjunto.

Por otro lado, y a nivel académico, ya trabajé con Natural Language Processing en mi TFG y nunca había visto nada de imágenes hasta dar las asignaturas en este máster, y me parecía interesante trabajar con este tipo de datos también para poder tener una visión más general y plena de la Inteligencia Artificial, por lo que este trabajo cumplía otro de los objetivos que quería cumplir con este trabajo.

Por último, estaba realmente interesado en realizar el trabajo con Roberto, ya que a nivel personal he disfrutado mucho de sus asignaturas en este máster y he podido observar la gran cantidad de conocimiento que tiene sobre una gran cantidad de campos dentro de la Inteligencia Artificial.

Todos estos factores consiguieron motivarme de manera inicial en la realización de este trabajo. Posteriormente y una vez estaba realizando el trabajo en sí, comprendí que entrenar una GAN no es una tarea sencilla y en cualquier página que se mencionen, se comenta que no es fácil conseguir entrenarlas y que funcionen. Por ello, cuando me enfrenté a este problema de cara y finalmente conseguí que funcionara, me produjo una gran satisfacción.

Con todo esto y una vez hemos realizado la introducción a este trabajo así como los aspectos que han motivado la realización del mismo, vamos a introducir las GAN, esta arquitectura con la que vamos a trabajar, así como realizar un repaso de cómo han ido avanzando desde su aparición en 2014.

CAPÍTULO 2
Estado del arte

En este apartado me gustaría centrarme en explicar de manera breve las Redes Generativas Antagónicas (GAN) [3] y sus diferentes avances en el estado del arte. Las GAN surgen en 2014, basándose en la idea de los juegos de suma zero de la teoría de juegos. En estos juegos, nos encontramos con dos jugadores o partes contrapuestas que batallan por ganar al otro, de forma que cuando uno hace un movimiento que le genera beneficio, este se obtiene quitándoselo al otro. Así, en las GAN vamos a tener a estas dos partes: el discriminador y el generador. El primero tiene la tarea de juzgar una entrada de entrada y decidir si es real o generada. Por otro lado, el segundo intenta generar datos de entrada e intentar confundir al discriminador. Al principio, la idea consiste en obtener un discriminador que pueda distinguir fácilmente las muestras generadas y ayudar al discriminador a hacer imágenes de mejor calidad hasta que, llegado cierto punto (si se alcanza), el discriminador no sepa distinguir imágenes reales de generadas.

Una vez introducida esta arquitectura, vamos a proceder a explorar los diferentes avances que se han producido a lo largo de estos años y que han ido mejorando diferentes aspectos de estas redes:

En primer lugar tendríamos el paper original, escrito por Goodfellow y sus compañeros [3] en 2014, en el cual se establece la idea fundamental de la arquitectura y se comenta la fórmula con la que se entrena estas dos redes de forma conjunta. A continuación tenemos un diagrama que intenta ilustrar la idea de esta arquitectura:

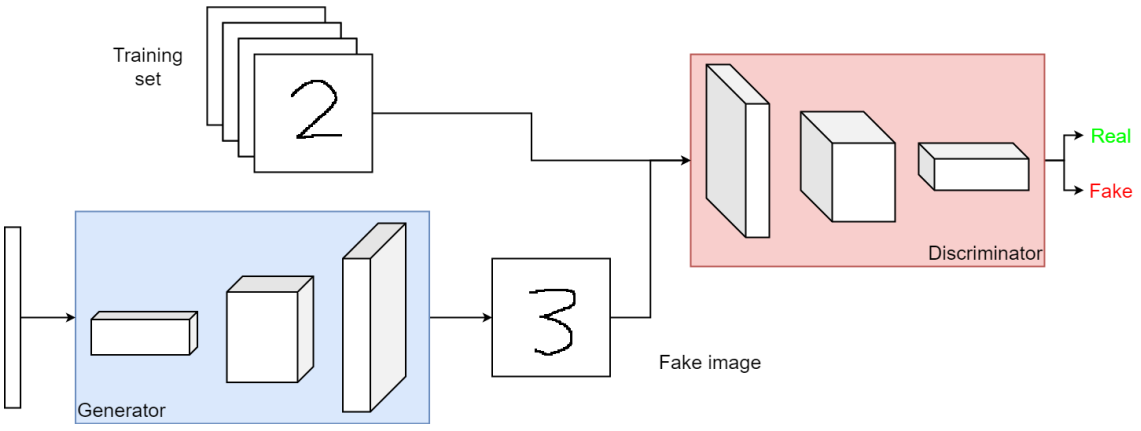


Figura 2.1: Imagen de la arquitectura de una GAN.

Como podemos ver en 2.2, estamos intentando entrenar un par de redes con las que pretendemos poder generar imágenes de un ámbito concreto. Para ello, el discriminador predice si la imagen es verdadera o no y el generador genera imágenes falsas.

Posteriormente, y el mismo año de la aparición de las GAN, apareció el trabajo de Mehdi Mirza y Simon Osindero [9], donde proponen las Conditional GAN, una red muy similar a la GAN, pero le añadimos a la entrada del discriminador y generador información adicional (y). Por ejemplo, esta y podría ser la clase real de la imagen. En ese caso, la red tendría la siguiente estructura:

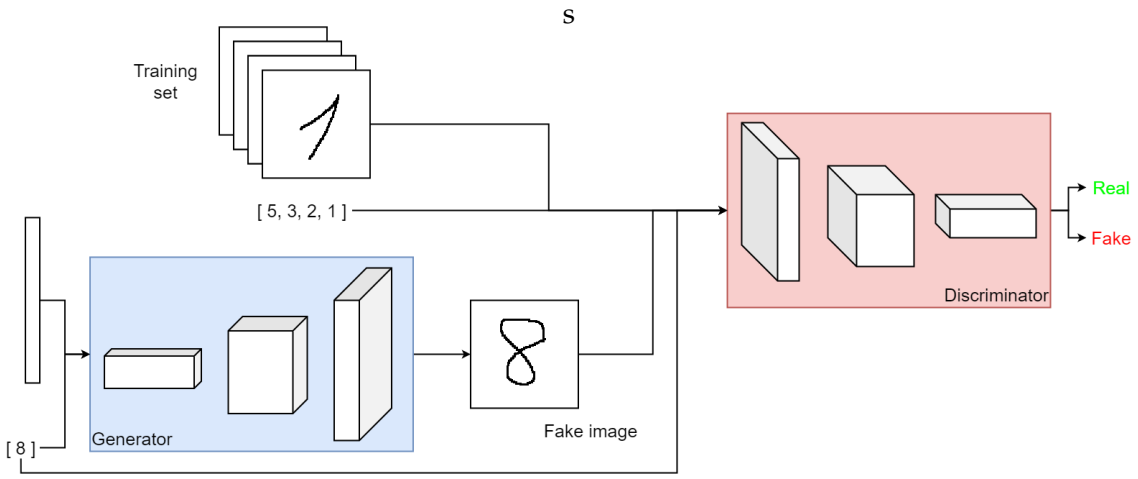


Figura 2.2: Imagen de la arquitectura de una Conditional GAN.

Con esto, podemos facilitar el entrenamiento y simplificar el problema al darle a ambos modelos información adicional que les facilitará la tarea. También se podría separar estos datos y entrenar un modelo por cada clase, pero esto significaría tener muchos más datos por clase y tener que almacenar más modelos.

Un año más tarde (2015), Radford y sus compañeros publicaron su trabajo con las GAN [10], donde presentaron las GAN Profundas Convolucionales (DCGAN), donde establecieron la arquitectura más utilizada durante los próximos años. Específicamente, establecieron los siguientes puntos claves a la hora de crear la arquitectura de ambas redes:

- Cambiar cualquier capa de pooling por convoluciones con stride.
- Utilizar batchnorm en el generador y discriminador.
- Eliminar las capas fully connected (FC) de las arquitecturas.
- Utilizar activaciones ReLU en el generador en todas sus capas menos en la salida, que será una Tanh.
- Utilizar activaciones LeakyReLU en el discriminador en todas sus capas.

Hasta este punto, uno de los principales problemas que tienen las GAN es que su entrenamiento es bastante difícil, y tienes que tener mucho cuidado y prestar mucha atención a cómo realizar correctamente el entrenamiento, ya que debes conseguir un buen balance entre la potencia de ambas redes. El mayor problema suele darse cuando una de las dos redes de la arquitectura domina a la otra, lo que consigue que uno de los dos gradientes sea muy cercano a 0 y que no se aprenda nada más. Esta problemática se ve reflejada en la práctica debido a diferentes causas y la mayoría de ellas tienen un nombre y una solución o procedimiento por el cual intentar solucionarlas. Dicho esto, ahora vamos a introducir dos arquitecturas que surgieron posteriores a las ya indicadas y que se centraron en intentar mejorar este proceso de entrenamiento.

En primer lugar, un año después, en 2016, Mao y sus compañeros publicaron las GAN por mínimos cuadrados (LSGAN) [8], donde proponen que el discriminador trabaje con un problema de cuantificación y no de clasificación binaria en falso o real. Para ello, se establece un valor a,b y c:

- a será el valor de la etiqueta de las muestras falsas.
- b será el valor de la etiqueta de las muestras reales.
- Por último, c es el valor que el generador quiere que el discriminador indique para las muestras falsas.

Según indican en la publicación, esto permite que los gradientes sean mayores cuando el valor indicado es mucho mayor que b o mucho menor que a, como puede verse en la siguiente figura.

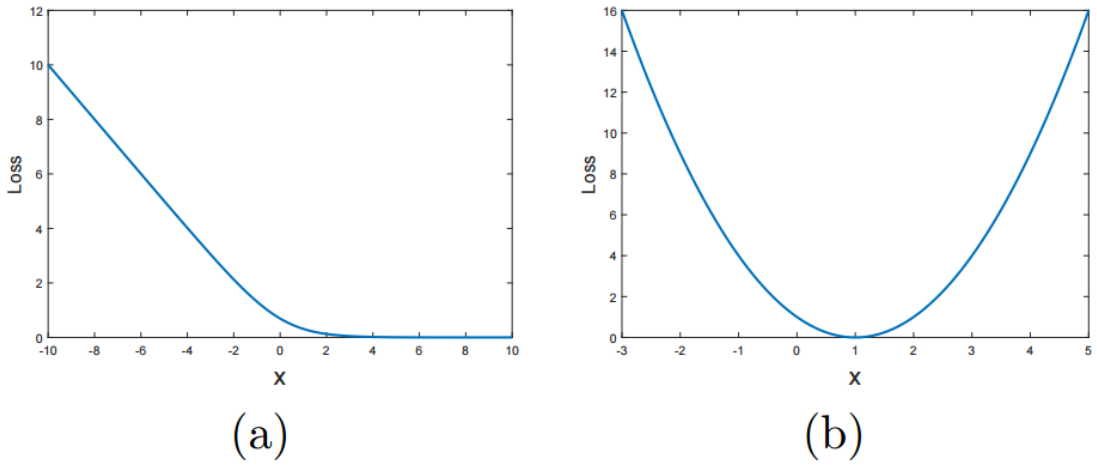


Figura 2.3: Imagen con la función de pérdida para la cross entropía y los mínimos cuadrados. Fuente

En segundo lugar, tenemos las Wasserstein GAN, un trabajo de Martin Arjovsky, Soumith Chintala y Léon Bottou [2] (2017), en el que se propone trabajar con la distancia de Earth Mover (EM) en vez de con distancias de probabilidades. En la publicación demuestran de manera matemática y formal que esta distancia converge en situaciones en las que otras distancias, como la distancia entre dos distribuciones de probabilidad no consiguen converger.

Por otro lado, para poder utilizar esta distancia de manera práctica con una red GAN se basan en diferentes asunciones con las que acaban indicando que se necesita mantener los pesos en un espacio compacto ($\mathcal{W}$). Esto en la práctica ellos lo aproximan a partir de un concepto que denominan weight clipping, que consiste en mantener los pesos de la red entre un rango de valores, de forma que la red no podrá tener pesos con valor superior o inferior a este parámetro de clipping.

Dentro de este trabajo, en la parte inicial en la que se compara estas diferentes distancias entre distribuciones de probabilidad, se muestra de manera gráfica un ejemplo muy similar a la Figura 2.3 de las LSGAN. Esta imagen se puede ver en la figura 2.4, que presentamos a continuación:

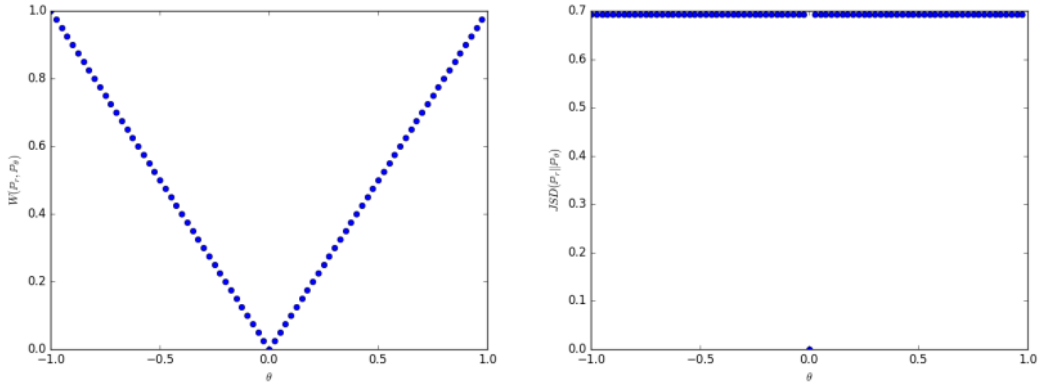


Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

Figura 2.4: Imagen extraída del trabajo original.

Una vez comentada de manera breve algunas publicaciones y avances en estas arquitecturas desde su aparición en 2014, además de después comentar diferentes problemáticas que surgen al tratar de entrenar estas arquitecturas, así como soluciones que surgieron para subsanar estas problemáticas, vamos a continuar con el siguiente capítulo, en el que se establecerá el conjunto de datos con el que trabajaremos.

CAPÍTULO 3
Conjunto de datos

En este apartado posterior al estado del arte vamos a introducir nuestro conjunto de datos inicial. En concreto, vamos a trabajar con un conjunto de datos de Kaggle denominado Breast Histopathology Images, que surge del trabajo de Andrew Janowczyk, un investigador que publicó en su página un conjunto de datasets, dentro de los cuales está el de imágenes de carcinoma ductal invasivo.

En este enlace se describe detalladamente el conjunto de datos, que está formado por un total de 162 imágenes con este tipo de cáncer. Como se explica, este tipo de cáncer ha de ser correctamente separado por un patologista, que tiene que obtener las partes exactas del pecho donde se encuentra este tejido cancerígeno. Para ello, deben examinar todo el tejido del pecho de manera minuciosa para identificar estas partes, de forma que se tiene que hacer una segmentación de este tejido del pecho en zonas sanas y con carcinoma.

Por ello, se ha realizado esta misma tarea para generar este conjunto de datos específicos, separando las enormes imágenes de cada paciente en parches cuadrados y pequeños de 50 píxeles que después han sido cuidadosamente clasificados como zonas de tejido sano o con carcinoma. Este proceso ha producido un conjunto de imágenes de 50x50 con un total de 277.524 muestras, de las cuales 198.738 son tejido sano y las 78.786 restantes son de tejido cancerígeno. Como podemos comprobar, este conjunto está bastante desbalanceado, de forma que el 72 % de las muestras es tejido sano y solo un 28 % son imágenes de tejido cancerígeno.

Así, al obtener este conjunto de datos, tenemos una estructura de carpetas separada por cada uno de los 279 pacientes de los que disponemos, donde cada carpeta hace referencia a un id de paciente específico, dentro de la cual podemos encontrar dos subcarpetas, una con un 0 para las muestras de tejido sano y otra con un 1 para las zonas con tejido con carcinoma.

Dentro de estos, una imagen toma el siguiente formato de nombre: u_xX_yY_classC.png. Este formato puede dividirse en los siguientes componentes:

- La u hace referencia al id del paciente específico.
- La X establece la coordenada incial en el eje X.
- La Y indica la coordenada inicial en el eje Y.
- Finalmente, la C indica a que clase pertenece el parche.

Como ejemplo, 10260_idx5_x51_y801_class0.png, donde 10260_idx5 indica el id del paciente, x51 es la posición del parche en el eje X, y801 en el eje Y y finalmente class0 indica que es una muestra de tejido sano.

Tras esto, vamos a realizar un poco de análisis de nuestros datos. Para ello, hemos utilizado diferentes funciones en Python que hemos obtenido gracias a Laura Fink a partir de su notebook sobre este dataset. En concreto, primero vamos a observar la cantidad de parches que tenemos a partir de cada paciente. Esto se muestra en la siguiente gráfica:

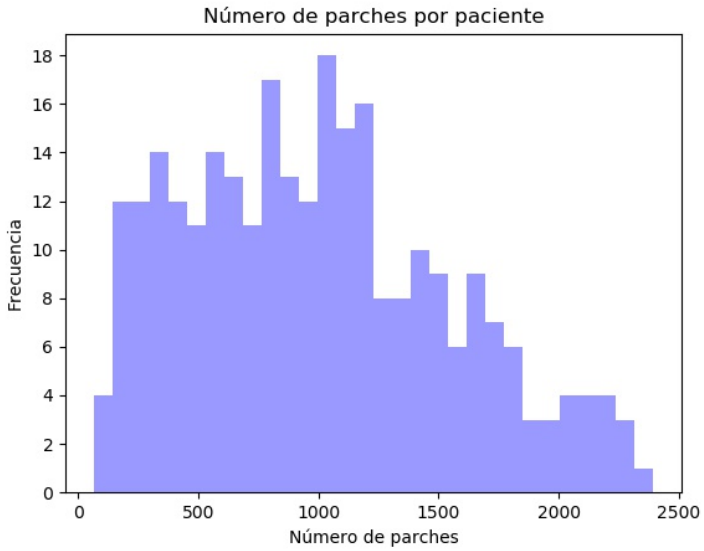


Figura 3.1: Gráfica con la distribución del número de parches por paciente.

Como podemos observar, tenemos algunos pacientes que solo tienen unos cientos de imágenes, mientras que los que más tienen llegan a las 2300 o 2400 aproximadamente. También podemos comprobar que tenemos más pacientes con menos de 1250 imágenes que con más.

Por otro lado, se ha reconstruido también una imagen original a partir de los parches del dataset, de forma que el resultado obtenido sería:

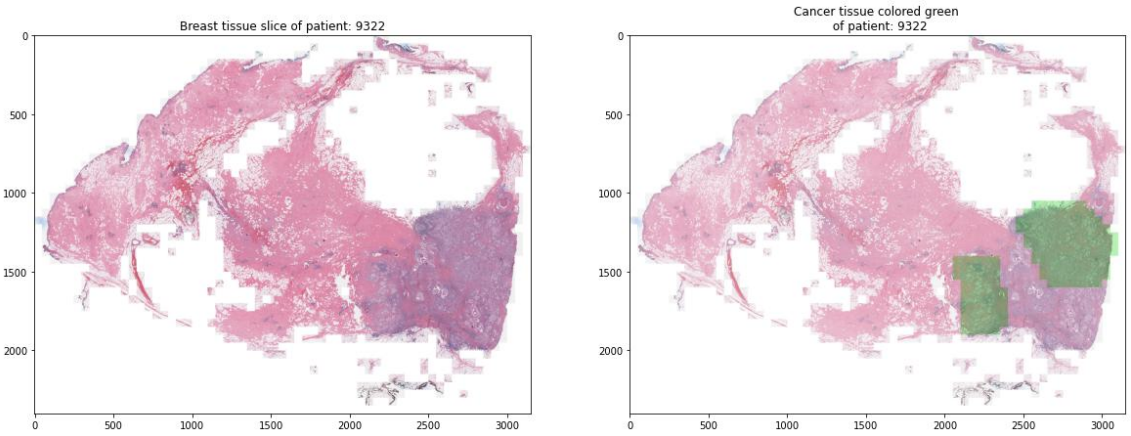


Figura 3.2: Imagen de la reconstrucción del tejido de la paciente 9322.

Como se puede observar, tenemos a la izquierda la imagen del tejido de la paciente 9322, para la cual no se tiene todos los parches de la imagen original, sin tener muy claro el motivo por el que esto es así. Podemos ver que estas imágenes tienen zonas con colores blancos, rosas, alguna línea negra y zonas moradas. Después tenemos la imagen de la derecha, donde podemos ver coloreadas en verde las zonas donde se tiene tejido con carcinoma. Se puede observar que son zonas con colores más morados, pero no son todas las zonas con un color más morado tejido cancerígeno, por lo que tampoco podemos

afirmar esta distinción. Parece claro que la distinción entre este tipo de tejidos es una tarea compleja y para la cual se necesita a un experto.

Una vez introducido el conjunto de datos con detalles específicos de este, pasamos a centrarnos en la metodología desarrollada a lo largo de este trabajo.

CAPÍTULO 4
Metodología

En este capítulo vamos a presentar la forma de proceder que se ha seguido en este trabajo. En primer lugar, explicaremos cómo se harán las pruebas iniciales de clasificación con datos reales. Posteriormente, se comentará de manera general cómo entrenar nuestras redes GAN. Tras esto, tendremos que establecer cómo evaluar la calidad de nuestra GAN y finalmente se especificará cómo comparar los resultados obtenidos en clasificación con las muestras originales en contraposición a cuando nos apoyamos en la generación de muestras con nuestra red.

Pasamos ahora a definir cómo se realizará la primera etapa de clasificación previa:

4.1 Evaluación del dataset original

En esta sección vamos a establecer cómo se realizarán las pruebas iniciales para obtener unos resultados que servirán de baseline para el resto del trabajo. Para ello, vamos a realizar particiones de los datos, obteniendo una cantidad fija de imágenes para tejidos sanos y cancerígenos, separando los conjuntos siempre de forma balanceada, que nos permitirá evitar problemas clásicos de problemas desbalanceados como mínimos locales en los que se predice siempre la clase más poblada. Esto también implica que nuestra métrica de evaluación para esta clasificación será la precisión.

Teniendo en cuenta que nuestra intención es evitar el desbalanceo de clases para este problema, no vamos a trabajar con la partición que se suele hacer en problemas médicos a nivel de pacientes, seleccionando aquellos que pertenecen al conjunto de entrenamiento, validación y test, sino que vamos a obtener siempre las mismas 1000 imágenes para validación y otras 1000 imágenes para test, ambos conjuntos totalmente balanceados. Esta forma de realizar las particiones consigue que tengamos solape en los pacientes para los diferentes conjuntos, pero siendo este solape perteneciente a parches de tejido, consideramos que no es una partición inválida como sí podría serlo en otros problemas dentro del ámbito médico.

Una vez tenemos este conjunto de test y validación, vamos a realizar diferentes conjuntos de entrenamiento para los que buscaremos obtener los mejores resultados posibles. En concreto, realizaremos pruebas con las diferentes particiones:

Nombre del conjunto	Nº imágenes tejido sano	Nº imágenes tejido cancerígeno	Nº imágenes totales
Conjunto1K	500	500	1000
Conjunto2K	1000	1000	2000
Conjunto5K	2500	2500	5000
Conjunto10K	5000	5000	10000
Conjunto25K	12500	12500	25000
Conjunto50K	25000	25000	50000

Tabla 4.1: Tabla con los diferentes conjuntos de datos creados a partir del conjunto de datos original.

Una vez establecidas las particiones y los diferentes conjuntos de entrenamiento, vamos a desarrollar la metodología seguida para entrenar una red GAN para este conjunto de datos.

4.2 Entrenamiento de la red GAN

Esta segunda sección servirá para comentar cómo entrenaremos nuestras dos redes GAN, una para generar imágenes con tejido sano y otra para tejido con carcinoma, aunque realmente la que más nos interesará será esta segunda.

En este caso, vamos a utilizar solo 1000 imágenes originales para realizar el entrenamiento, de forma que entrenaremos por batches, ya que debemos entrenar a nuestro discriminador con un conjunto de imágenes reales y otro conjunto de imágenes generadas por el generador, además de luego entrenar al generador con un conjunto de vectores de ruido (z).

Adentrándonos un poco más en detalles específicos del entrenamiento de estas redes, tenemos que tener en cuenta que tendremos por un lado el discriminador, que se compilará como un modelo independiente, con su función de pérdida y su optimizador, y por otro lado, tendremos la gan, formada por el generador y el discriminador (no entrenable), con su función de pérdida y su optimizador, siendo estos independientes al discriminador.

De este modo, se trabajará a nivel de iteraciones o epochs, en las cuales se irán haciendo entrenamientos por batches, alternando el entrenamiento del discriminador y el generador (la gan previamente mencionada), pudiendo realizar más actualizaciones a una de las redes si así se considerara oportuno.

Por último, un punto clave para que este entrenamiento sea exitoso es guardar de manera periódica imágenes obtenidas con el generador, para así poder observar el avance del entrenamiento, ya que ni las funciones de pérdida ni ninguna otra métrica indicará realmente si los resultados son buenos o no. Así, una vez realizado el entrenamiento podremos comprobar gracias a las funciones de pérdida y las imágenes generadas a lo largo de las epochs si el generador está aprendiendo algo o si por el contrario el entrenamiento está fallando de alguna forma por algún factor.

Una vez establecidos estos puntos específicos del entrenamiento de las GAN, vamos a proceder a comentar algunos trucos y detalles de implementación ampliamente cono-

cidos y establecidos por diferentes fuentes e investigadores para conseguir entrenar una GAN de forma exitosa.

4.2.1. Detalles de implementación para entrenar una GAN

Dentro de esta subsección, tenemos que establecer algunos trucos ampliamente conocidos y obtenidos de aquí, aquí y aquí sobre como entrenar una GAN.

- Un punto clave es el formato de los datos de entrada, es decir, las imágenes reales y generadas. En concreto, al trabajar con redes GAN es común normalizar el valor de los píxeles de las imágenes entre -1 y 1. Esto supone que, como se comentaba en 2 en los puntos establecidos en la DCGAN, la capa de salida del generador tenga una capa de activación tanh, que se encargará de obtener estos valores de los diferentes píxeles normalizados.
- Otro truco se centra en la función de pérdida, para la que en vez de trabajar con la formulación original [3] y buscar el mín$(\log 1 - D)$, en la práctica optimizaremos máx $\log D$. Para ello, cuando entrenemos al generador (la gan) con el discriminador no entrenable, le mentiremos al discriminador diciéndole que las imágenes generadas son imágenes reales, intentando que el discriminador le indique al generador qué debe ir cambiando para que realmente estas imágenes engañen al discriminador.
- También es importante separar las imágenes generadas y reales en diferentes batches al entrenar al discriminador, ya que no queremos aplicar técnicas de normalización como BatchNorm con imágenes de distintas clases. Tampoco querremos que el discriminador vea más batches de una clase que de la otra, ya que la idea es entrenar al discriminador de forma balanceada para ambas clases.
- Además de estos otros trucos, será importante trabajar con label smoothing (soft labels) y label flipping (noisy labels) al entrenar al discriminador. Label smoothing hace referencia a que no trabajaremos con etiquetas con valor 0 para imágenes reales o 1 para imágenes generadas, sino que para las etiquetas reales sumaremos y para las generadas restaremos un valor aleatorio entre un rango fijo. Por otro lado, label flipping hace referencia a añadir ruido a las etiquetas, de forma que en un pequeño porcentaje de muestras se cambie la etiqueta por la contraria, indicando que una imagen real es generada y viceversa. Estos trucos conseguirán que el entrenamiento funcione y no sea un aprendizaje tan estricto que falle rápidamente.

Una vez se han establecido estos trucos, vamos a comentar de manera breve algunos problemas que pueden surgir al entrenar estas redes.

4.2.2. Problemas al entrenar una GAN

En esta segunda subsección simplemente queríamos destacar que las GAN tienen muchas dificultades para entrenarse de forma exitosa y que dentro de este ámbito, existen errores comunes al entrenarlas, que se conocen como GAN failure modes.

El primero se conoce como mode collapse y realmente lo que indica este mode de la GAN es que genera imágenes correctas, pero solo genera una pequeña variedad de imágenes, siendo muchas de ellas parecidas, sino las mismas. Esto se ejemplifica a continuación en la siguiente imagen:

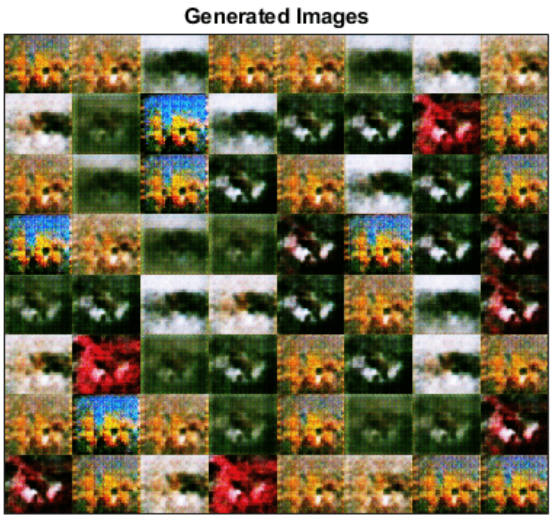


Figura 4.1: Imagen extraída de la fuente.

Como podemos ver, la GAN genera imágenes prácticamente idénticas, y lo único que cambia en ellas es el color que tienen, a pesar de que el vector de ruido de todas ellas era distinto.

Por otro lado, tendríamos el convergence failure, que implica que no se llega a un balance entre ambas partes durante el entrenamiento, y puede darse porque el discriminador domina al generador o al revés. En este caso tenemos dos opciones:

- reducir la potencia de la red dominante reduciendo el número de filtros, añadiendo dropout o, en el caso del discriminador, cambiando las etiquetas de imágenes reales por generadas (one-sided label flipping).
- aumentar la potencia de la red dominada añadiendo filtros y parámetros.

Una vez se ha introducido la forma de entrenar una GAN, algunos trucos y detalles de implementación que serán necesarios y algunos problemas reconocidos que podemos encontrarnos al entrenar estas redes, pasamos a comentar cómo se evalúa una GAN, una vez se tiene esta arquitectura entrenada.

4.3 Evaluación de la calidad de la GAN

En este apartado vamos a comentar algunas formas de evaluar los resultados que se obtienen con una GAN entrenada, pero este es un problema abierto dentro de la investigación y no hay un consenso establecido sobre una métrica que funcione bien para todos los problemas como podría ser la precisión o el F1-Score cuando se trabaja con problemas de clasificación. Por ello, vamos a introducir algunos métodos y métricas a la hora de evaluar estos modelos y en siguientes capítulos se explorará su uso para este problema concreto.

En primer lugar tenemos un método clásico y utilizado para muchos problemas de este tipo que es la evaluación manual, de forma que se selecciona a un conjunto de personas para evaluar las imágenes generadas y si estas son de buena calidad para evaluar el modelo. Esta solución podría servirnos para seleccionar un modelo, pero no nos sirve

para evaluar la calidad, ya que es un método subjetivo, que no puede darnos una puntuación objetiva como obtendríamos con la precisión. Además de esto, y centrándonos en el dominio de este trabajo, necesitaríamos que estas personas que realizan la evaluación fueran expertos patologistas, ya que una persona corriente no sabrá indicar si la calidad de las imágenes generadas es suficiente o no.

A continuación se comenta un conjunto de métodos y técnicas para realizar una evaluación cualitativa, donde se pretende comparar estas imágenes reales y generadas. Dentro de este conjunto, tenemos técnicas que también se basan en evaluaciones manuales, pero en este caso la idea es seleccionar aquellas imágenes que más realistas te parezcan, ya sea viendo la imagen en todo momento (Rating and Preference Judgment) o solo durante un corto periodo de tiempo (Rapid Scene Categorization). Sin embargo, dentro de este grupo, queremos destacar el uso de Nearest Neighbors, una técnica que podría utilizarse para explorar si se puede realizar una separación de forma natural en el espacio vectorial entre ambas clases.

Finalmente, se introduce técnicas y métricas para realizar evaluaciones cuantitativas, con las que poder generar un valor numérico que mida la calidad de estas imágenes generadas. Entre estas métricas, queremos introducir:

- En primer lugar tenemos el Inception Score (IS) [11], que se obtiene a partir de utilizar el Inception v3 preentrenado con Imagenet. La idea es utilizar una gran cantidad de imágenes generadas y clasificarlas con este modelo, obteniendo las probabilidades de la capa de clasificación y juntándolas para generar el IS, que indicará cuánto se parece cada imagen a una clase conocida y cuánta diversidad hay. Cuanto mayor sea este valor, mayor será la calidad de las imágenes generadas por la GAN.
- Después de este score, se propuso como mejora en [5] el Frechet Inception Distance (FID), una métrica que se obtiene también a partir del Inception v3 preentrenado con Imagenet, pero en este caso la idea será utilizar la salida de la última capa previa a la de clasificación para generar características visuales de la imagen de entrada. Estas características se obtienen para imágenes generadas y reales, para posteriormente juntarlas todas como una distribución gaussiana multivariada y obtener posteriormente la distancia de Frechet entre ambas distribuciones. Cuanto menor sea esta distancia, más cercanas serán estas imágenes a las reales en base a sus características.

Una vez introducidas estas métricas y metodologías para evaluar los resultados obtenidos por nuestra red GAN, vamos ahora a comentar en último lugar la evaluación de este dataset generado.

4.4 Evaluación del dataset generado

En esta última sección del capítulo de metodología vamos a explicar cómo evaluaremos este dataset utilizando muestras generadas por nuestra GAN entrenada. En primer lugar, la idea será comparar los resultados obtenidos sobre el conjunto de datos original con resultados utilizando estas muestras generadas para la clasificación de imágenes reales del conjunto de test. Para ello, realizaremos una prueba con un nuevo conjunto de datos denominado **Conjunto10KGen**, que tendrá las siguientes características:

Conjunto10KGen	Clase de la imagen	Cantidad
Imágenes reales	Tejido sano	2500
	Tejido cancerígeno	2500
Imágenes generadas	Tejido sano	2500
	Tejido cancerígeno	2500
Nº imágenes totales		**10000**

Tabla 4.2: Tabla con el **Conjunto10KGen** detallado.

Tras entrenar, evaluaremos y discutiremos los resultados obtenidos con este conjunto y lo compararemos con los conjuntos mencionados en 4.1.

Finalmente, se realizarán dos experimentos más para comprobar el aporte de esta generación de imágenes cuando se intenta explotar todo el conjunto de imágenes disponibles del conjunto de datos original. Así, tendremos la comparativa entre los siguientes experimentos:

1. El primero se realizará con el conjunto de datos **ConjuntoAllReal** y consistirá en entrenar una red con todos los datos disponibles en el conjunto de datos originales de manera balanceada. Estas serán sus estadísticas:

ConjuntoAllReal	Clase de la imagen	Cantidad
Imágenes reales	Tejido sano	77786
	Tejido cancerígeno	77786
Nº imágenes totales		**155572**

Tabla 4.3: Tabla con los detalles del **ConjuntoAllReal**.

2. El segundo y último experimento de clasificación se hará con el conjunto de datos denominado **ConjuntoAllGen**, que como su nombre indica, se basa en una prueba con todas las muestras generadas que sean necesarias para balancear el conjunto original y así aprovechar y utilizar todos los datos reales. En concreto, el conjunto tendrá:

ConjuntoAllGen	Clase de la imagen	Cantidad
Imágenes reales	Tejido sano	197738
	Tejido cancerígeno	77786
Imágenes generadas	Tejido sano	0
	Tejido cancerígeno	119952
Nº imágenes totales		**395476**

Tabla 4.4: Tabla con el **ConjuntoAllGen** detallado.

Como podemos ver, esta comparativa se realizará entre un modelo con 150K imágenes y otro con 400K imágenes. El modelo entrenado será el mismo para ambos casos, es decir, utilizaremos la mejor configuración posible basándonos en los experimentos previos para que la comparativa sea lo más justa posible dentro de que el modelo con imágenes generadas tendrá 2.5 veces más datos de entrenamiento que el primer modelo.

Una vez se ha establecido esta metodología, pasamos al siguiente capítulo, donde estableceremos los diferentes experimentos y pruebas que se han llevado a cabo.

CAPÍTULO 5
Experimentos

En este siguiente capítulo vamos a definir las diferentes pruebas que se realizaran para cada uno de los apartados definidos previamente en 4.

5.1 Evaluación del dataset original

En esta sección vamos a especificar que pruebas se han realizado para los conjuntos de datos que se definieron en 4.1. En concreto, hemos probado diferentes configuraciones de Convolutional Neural Networks (CNN), específicamente ResNets [4]. Para estas pruebas, se ha variado el número de parámetros, cambiando la profundidad de la red y el número de filtros, pero con los mismos bloques residuales siempre. Específicamente, los bloques residuales tienen el siguiente aspecto:

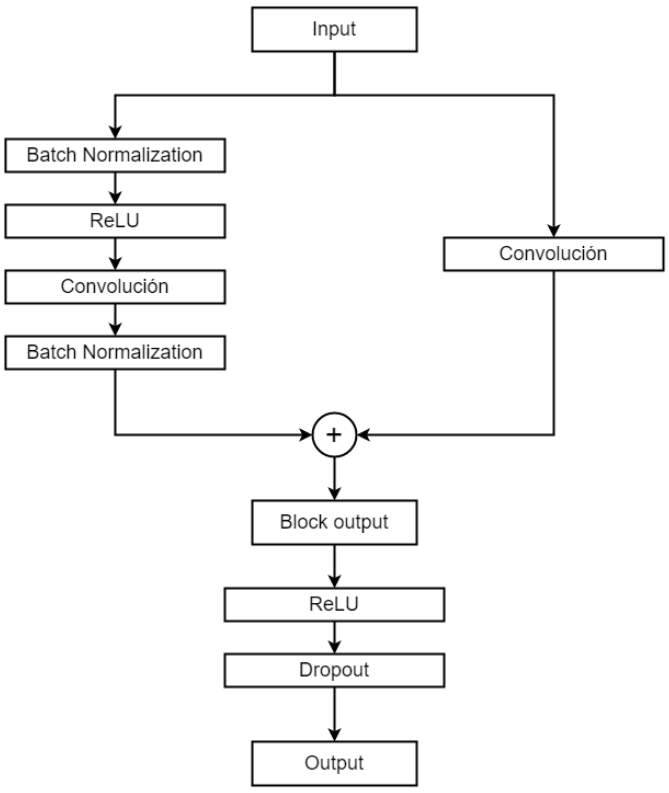


Figura 5.1: Esquema a alto nivel de los bloques residuales empleados.

Como podemos ver en la imagen previa, hemos decidido utilizar el dropout como nuestro método de regularización para evitar el overfitting. Debido a que en nuestro problema el data augmentation que podíamos aplicar a estas imágenes era bastante reducido, en algunos experimentos este dropout ha tenido que ser bastante elevado.

Estableciendo detalles generales de todos los experimentos realizados, el data augmentation aplicado ha sido el siguiente:

Data augmentation	**Valor**
Reescalado	1./255.
Desplazamiento vertical	0.1
Desplazamiento horizontal	0.1
Rango de rotación	30º
Rango de zoom	[0.9, 1.1]

Tabla 5.1: Tabla con los detalles del data augmentation aplicado para las imágenes.

Este aumento de datos es bastante suave, ya que los desplazamientos son bastante bajos, también debido a que los parches son bastante pequeños. La rotación y el zoom tampoco son muy elevados, ya que queremos evitar cambios muy bruscos que puedan generar imágenes inservibles. Este data augmentation podría verse enriquecido y potenciado a partir de un análisis con conocimiento experto de este ámbito. Con estos aumentos, hemos desarrollado un generador que nos proveyese batches de imágenes totalmente balanceadas entre ambas clases, para que el modelo viera en todo momento la misma cantidad de imágenes de ambas clases, incluso dentro del mismo batch.

Finalmente, vamos a centrarnos en estos modelos que hemos empleado y se han mencionado previamente:

Nombre del modelo	**Nº de parámetros**
Model150K*	150850
Model300K	299618
Model5M	4797346
Model6M	6310402
Model11M	11296770

Tabla 5.2: Tabla con los diferentes modelos y su número de parámetros.[1]

Como podemos ver, el modelo **Model150K** no tiene los bloques residuales que se mencionaron antes, sino que es un modelo que hemos seleccionado de otra persona que había abordado este problema de clasificación. En concreto, es una CNN simple con capas convolucionales, pools, dropout, relus y una densa.

Al realizar los entrenamientos con estas redes, se observó que el entrenamiento era muy inestable, dando saltos en la precisión del modelo en validación, además de que no parecía haber mucha transferencia del conjunto de entrenamiento al de validación o test. Se intentó reducir el factor de aprendizaje, variar el dropout aplicado a todos los bloques, llegando a valores de 0.8, y añadir una capa densa al final, pero este problema no se pudo subsanar del todo.

El problema de esta inestabilidad la encontramos al entrenar la GAN, que explicaremos en la siguiente sección y en el siguiente capítulo al comentar los resultados, pero el problema se hallaba en la capa Batch Normalization, una capa que normaliza los datos en función de la media y desviación típica trabajando a nivel de batch. El problema viene

[1] *Este modelo se ha obtenido de este código y se le han realizado pequeñas modificaciones.

de las imágenes con las que estamos trabajando, ya que al ser imágenes con una estructura celular variante, no nos encontramos casos como imágenes de caras donde tenemos en muchas ese patrón facil en la misma zona de la imagen. En este caso, podemos tener una zona morada arriba en una imagen y que en la siguiente sea todo blanco y algunas líneas negras. Esto supone que esta normalización estaba arruinando nuestras imágenes y complicando el entrenamiento del modelo.

Otro problema que surge de estos experimentos es que se realizaron de manera inicial en un cuaderno en Kaggle, pero al entrenar la GAN y realizar la evaluación del dataset generado posteriormente, el orden en que se recorría las imágenes por los directorios cambiaba entre ambas máquinas. Esto implica que el conjunto de validación y test no eran los mismos. Ambos factores sumados conseguían que los resultados no pudieran compararse con ninguno de los trabajos posteriores, por lo que se han realizado todos los experimentos de nuevo, esta vez con las condiciones correctas para realizar la comparativa. De esta forma, los valores de los diferentes parámetros se muestran a continuación:

Nombre del parámetro	**Valor**
Batch size	128
Número de iteraciones	80
Learning Rate	$1e-3$
Dropout	0.3
ReduceLROnPlateau	Sí
Paciencia	6
Learning Rate mínimo	$1e-7$

Tabla 5.3: Tabla con los parámetros aplicados para entrenar a los modelos de clasificación.

Como podemos ver en la tabla superior y dado que hemos corregido este entrenamiento inestable, no hemos tenido que realizar cambios en el dropout para que funcionara y por ello se ha mantenido en 0.3 para todos los experimentos.

5.2 Entrenamiento de la red GAN

En este segundo apartado vamos a explicar el proceso de entrenamiento de la GAN. En primer lugar se realizó todos los detalles de implementación que se han comentado en 4.2.1 para entrenar al modelo para obtener imágenes de tejido con carcinoma. Para ello, nuestra GAN tenía la siguiente configuración de los parámetros globales:

Nombre del parámetro	**Valor**
Dimensión del vector de ruido	100
Altura y anchura de las imágenes generadas	64
Batch size	32
Número de iteraciones	50000
Dropout	No
Batch Normalization	Sí

Tabla 5.4: Tabla con los parámetros aplicados para entrenar a la GAN.

Uno de los aspectos más relevantes de esta tabla es el tamaño de las imágenes de salida del generador, ya que nuestros parches son de 50x50, pero nuestro generador obtendrá imágenes de 64x64. Esto se ha realizado porque trabajamos con capas Conv2DTranspose

en el generador, y no puedes especificar el tamaño de salida de la capa. Por ello, esta red trabajara con parches reales y generados de 64x64 y cuando esté entrenado, haremos un reshape a 50x50.

Esta primera configuración de la red denominada **GAN_latentdim100_bn** no funcionó, así que decidimos aumentar la dimensión del vector de ruido, pasando a 512. Además de este cambio, decidimos añadir dropout también, ya que a lo largo de las epochs, íbamos generando periódicamente tres imágenes del generador, que resultaban ser prácticamente idénticas sin serlo por pequeños cambios en píxeles. Este segundo experimento (**GAN_latentdim512_drop_bn**) tampoco consiguió mejorar los resultados. Tras esto, decidimos eliminar el Batch Normalization, ya que en algunos problemas, su uso podía entorpecer el entrenamiento. Este modelo denominado **GAN_latentdim512_drop** fue nuestro último intento sin éxito.

Llegados a este punto, y sin ninguna opción más que probar, decidimos mirar trabajos relacionados con este trabajo y en los que se utilizaran GAN para imágenes histopatológicas. Fue en este punto donde encontramos el trabajo de Tasleem Kausar y sus compañeros [7], donde comentan su arquitectura, para la que utilizan capas denominadas Instance Normalization [12]. Esta capa normaliza la imagen teniendo en cuenta solo sus valores, y no trabajando a nivel de batch. Esta diferencia es clave, ya que como hemos mencionado antes, las imágenes celulares histopatológicas pueden ser muy distintas unas de otras a pesar de pertenecer a la misma clase. Con esto en mente, realizamos una última prueba (**GAN_latentdim512_in**) con esta normalización en vez de Batch Normalization y eliminamos las capas de dropout. Además, extendimos el número de iteraciones al doble (100000).

Con esto conseguimos generar buenos resultados y con esta misma configuración, entrenamos después al modelo **GAN_latentdim512_in_noncancer** para generar imágenes de tejido sano.

5.3 Evaluación de la calidad de la GAN

En esta sección realizaremos un experimento para calcular el FID, para el cual utilizaremos solo las imágenes de tejido cancerígeno, teniendo un total de 78786 imágenes reales y 119952 imágenes.

Cabe destacar que esta técnica de evaluación no funciona para nuestro caso, igual que el IS. Esto se debe a un motivo principalmente: un modelo Inception, por mucho que se entrene con Imagenet, no sabrá qué es una imagen con tejido celular, haciendo imposible que pueda generar embeddings que realmente representen correctamente a estos parches. Otro factor importante también es el hecho de que los parches son de 50x50, imágenes muy pequeñas para un modelo preentrenado para Imagenet, ya que el mínimo tamaño de las imágenes de entrada en este modelo es de 75x75.

Otro de los motivos por los que podemos realizar esta afirmación es porque en los experimentos realizados en 5.1, se probó la clasificación sobre el conjunto de datos original con redes preentrenadas (DenseNet169) sin buenos resultados, quedando claro que estos modelos no podían extrapolarse completamente a este ámbito.

A pesar de todo esto y que no consideremos que vaya a ser una técnica de evaluación adecuada, vamos a calcularla igualmente. Sin embargo, necesitaremos otra manera de evaluar estas imágenes. Para ello, hemos decidido utilizar dos métodos:

- El primero será realizar histogramas del valor de los píxeles diferenciando por colores RGB para todas las imágenes generadas y reales. Esto no será una medida

numérica ni tiene por qué indicar una semejanza en la estructura celular y de las imágenes, pero sí puede establecer una similitud en los píxeles de las imágenes, que es lo que realmente forma estas.

- La evaluación de los nuevos conjuntos de datos con imágenes generadas también serán una forma de evaluar si las imágenes son realmente de calidad o no, ya que si no cumplen con un estándar de calidad, deberían empeorar los resultados del modelo, especialmente con el **ConjuntoAllGen**, para el que tenemos muchas más imágenes generadas que reales, y si no son de buena calidad, el modelo no podrá diferenciar correctamente la clase de imágenes cancerígenas.

Finalmente, y a modo de evaluación personal, también se incluirán GIFs con una pequeña muestra de imágenes generadas y reales, para que se pueda realizar una comparativa subjetiva de la calidad de estas imágenes.

5.4 Evaluación del dataset generado

Por último, tendremos que realizar una evaluación de nuestros modelos de clasificación con estos nuevos datasets que contienen imágenes generadas.

En primer lugar, realizaremos las pruebas del **Conjunto10KGen** con todos nuestros modelos y realizaremos la comparativa con los resultados de los últimos experimentos del apartado 5.1. Para ello, aplicaremos los mismos valores para los parámetros que se especificaron en la Tabla 5.3.

Una vez realizada la comparativa, obtendremos el modelo que obtiene mejores resultados para los experimentos realizados hasta el momento y después utilizaremos los conjuntos **ConjuntoAllReal** y **ConjuntoAllGen** y los compararemos. Esta comparativa tendrá diferentes parámetros, ya que el número de datos aumenta y queremos obtener los mejores resultados posibles. De este modo, estos serán los parámetros escogidos para los experimentos:

Nombre del parámetro	**Valor**
Batch size	256
Número de iteraciones	100
Learning Rate	$1e-3$
Dropout	0.3
ReduceLROnPlateau	Sí
Paciencia	8
Learning Rate mínimo	$1e-7$

Tabla 5.5: Tabla con los parámetros aplicados para entrenar con los conjuntos **ConjuntoAllReal** y **ConjuntoAllGen**.

Una vez establecidos los experimentos que se van a realizar, en el siguiente capítulo se comenta y discute los resultados obtenidos.

CAPÍTULO 6
Resultados

Pasamos ahora a comentar los resultados obtenidos en las diferentes secciones del trabajo que se ha realizado.

6.1 Evaluación del dataset original

Estos son los resultados obtenidos para el conjunto de datos original de manera inicial, con el entrenamiento inestable:

Precisión en validación	**Model150K**	**Model300K**	**Model5M**	**Model6M**	**Model11M**
Conjunto1K	-	78.1 ± 0.03	-	-	73.1 ± 0.03
Conjunto2K	-	80.9 ± 0.02	-	74.4 ± 0.03	65.0 ± 0.03
Conjunto5K	-	81.6 ± 0.02	77.2 ± 0.03	78.1 ± 0.03	63.2 ± 0.03
Conjunto10K	85.8 ± 0.02	83.5 ± 0.02	84.4 ± 0.03	86.8 ± 0.02	87.6 ± 0.02
Conjunto25K	89.0 ± 0.02	89.1 ± 0.02	89.2 ± 0.02	86.5 ± 0.02	87.1 ± 0.02
Conjunto50K	91.7 ± 0.02	92.3 ± 0.02	91.7 ± 0.02	89.6 ± 0.02	86.0 ± 0.02

Tabla 6.1: Tabla con los resultados iniciales en validación.

Como podemos comprobar, los resultados son buenos cuando tenemos muchos datos, pero en los primeros conjuntos, no se llega a ningún valor bueno. Todo esto con un entrenamiento inestable que tenía picos de una precisión de 0.5 que subía una iteración a ese 0.8 y luego volvía a irse a 0.5. De hecho, el entrenamiento era probar parámetros y ver si funcionaban en las primeras 15 iteraciones, porque después de eso sobre entrenaba y ya era una precisión de 0.5 en validación siempre. Debido a que hemos realizado los experimentos de nuevo, por los motivos comentados en 5.1, no vamos a mostrar los resultados en test de estos primeros experimentos y vamos a mostrar los resultados que compararemos con nuestros datasets generados. Los resultados de los últimos experimentos son los siguientes para el conjunto de validación:

Precisión en validación	Model150K	Model300K	Model5M	Model6M	Model11M
Conjunto1K	91.8 ± 0.02	90.3 ± 0.02	89.8 ± 0.02	88.6 ± 0.02	90.7 ± 0.02
Conjunto2K	92.9 ± 0.02	87.3 ± 0.02	88.7 ± 0.02	89.7 ± 0.02	89.6 ± 0.02
Conjunto5K	88.3 ± 0.02	92.1 ± 0.02	91.9 ± 0.02	92.4 ± 0.02	91.4 ± 0.02
Conjunto10K	74.2 ± 0.02	93.4 ± 0.02	91.8 ± 0.02	90.7 ± 0.02	93.0 ± 0.02
Conjunto25K	88.5 ± 0.02	93.3 ± 0.02	93.8 ± 0.02	93.8 ± 0.02	93.9 ± 0.02
Conjunto50K	93.0 ± 0.02	93.5 ± 0.02	94.8 ± 0.02	94.9 ± 0.02	94.7 ± 0.02

Tabla 6.2: Tabla con los resultados del entrenamiento estable en validación.

Centrándonos en esta tabla, podemos comprobar que tiene resultados mucho más estables, en los que el entrenamiento funciona de manera continuada y sin dar picos de rendimiento. Específicamente, podemos observar que con pocas muestras, los mejores resultados se obtienen con el **Model150K**. Sin embargo, conforme aumentamos el tamaño del conjunto de entrenamiento, sus resultados se mantienen estables. Por otro lado, el resto de modelos tiende a mejorar por regla general cuantos más datos le proveemos para entrenar. Los mejores resultados se obtienen con el **Conjunto50K**, como era de esperar, en concreto con los modelos **Model6M**, **Model5M** y **Model11M**.

Dicho esto, adjuntamos a continuación la precisión obtenida en estos experimentos para conjunto de test:

Precisión en test	Model150K	Model300K	Model5M	Model6M	Model11M
Conjunto1K	87.8 ± 0.02	89.8 ± 0.02	88.6 ± 0.02	86.2 ± 0.02	87.1 ± 0.02
Conjunto2K	87.7 ± 0.02	86.8 ± 0.02	86.6 ± 0.02	86.4 ± 0.02	87.2 ± 0.02
Conjunto5K	81.9 ± 0.02	87.3 ± 0.02	87.5 ± 0.03	83.6 ± 0.02	86.5 ± 0.02
Conjunto10K	80.5 ± 0.02	91.7 ± 0.02	90.6 ± 0.02	88.9 ± 0.02	91.0 ± 0.02
Conjunto25K	87.6 ± 0.02	93.0 ± 0.02	93.2 ± 0.02	91.3 ± 0.02	92.6 ± 0.02
Conjunto50K	91.2 ± 0.02	93.6 ± 0.02	94.3 ± 0.02	94.5 ± 0.02	95.1 ± 0.02

Tabla 6.3: Tabla con los resultados del entrenamiento estable en test.

Centrándonos en los resultados, podemos comprobar que en validación los resultados son en general mejores, algo de esperar dado que los modelos se entrenan con los datos de entrenamiento, pero el modelo se escoge en función del desempeño para el conjunto de validación. Sin embargo, en algunos casos los resultados para el conjunto de test superan a los obtenidos en la validación. Este es el caso de nuestro mejor resultado en la tabla y los experimentos, una precisión de 95.1 % con el **Conjunto50K** y el **Model11M**.

Estos son los resultados con los que compararemos nuestros resultados con el dataset con muestras sintéticas.

6.2 Entrenamiento de la red GAN

En este apartado vamos a mostrar los resultados del entrenamiento de cada modelo. En primer lugar tenemos la **GAN_latentdim100_bn**, cuya generación puede apreciarse a continuación:

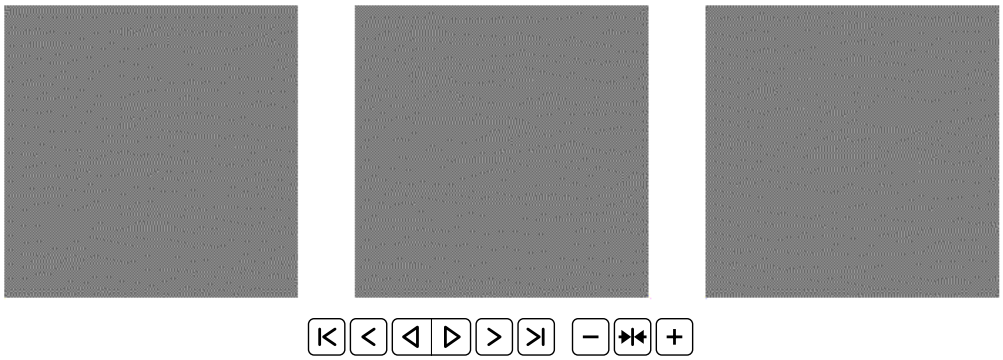


Figura 6.1: GIF con el entrenamiento iterativo del modelo **GAN_latentdim100_bn**.[1]

Como puede apreciarse (en caso de no poder reproducir los GIFs, ver el Apéndice A, en concreto, las Figuras A.1, A.2 y A.3), el resultado es bastante malo y no avanza hacia la generación de ninguna imagen mínimamente cercana a imágenes histopatológicas. Tras esto, se probó el modelo **GAN_latentdim512_drop_bn**, que tampoco obtuvo buenos resultados, como podemos comprobar a continuación o en las Figuras A.4, A.5 y A.6.

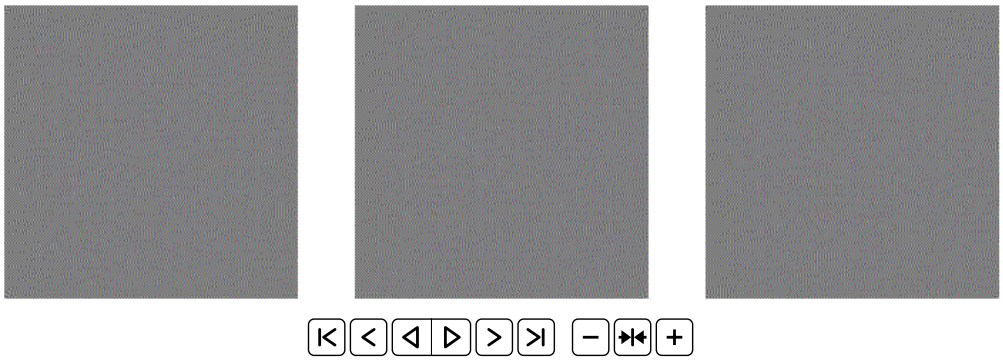


Figura 6.2: GIF con el entrenamiento iterativo del modelo **GAN_latentdim512_drop_bn**.

Nuestra siguiente prueba implicó eliminar las capas de Batch Normalization, y con este nuevo modelo (**GAN_latentdim512_drop**) obtuvimos nuestro último experimento fallido (Figuras A.7, A.8 y A.9).

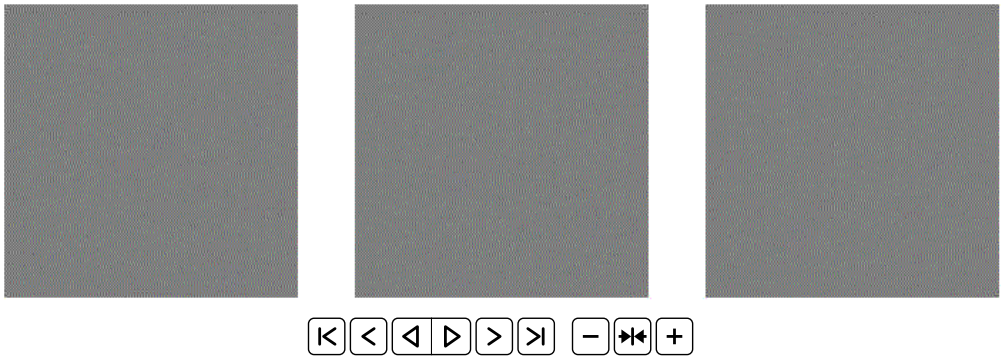


Figura 6.3: GIF con el entrenamiento iterativo del modelo **GAN_latentdim512_drop**.

[1]*Recomendamos utilizar un lector de PDFs que permita visualizar GIFs como Adobe Acrobat o Foxit Reader.

Como se ha comentado en 5.2, al añadir el Instance Normalization realizamos nuestra última prueba, con la que obtuvimos el modelo **GAN_latentdim512_in**, que generó las siguientes imágenes en entrenamiento:

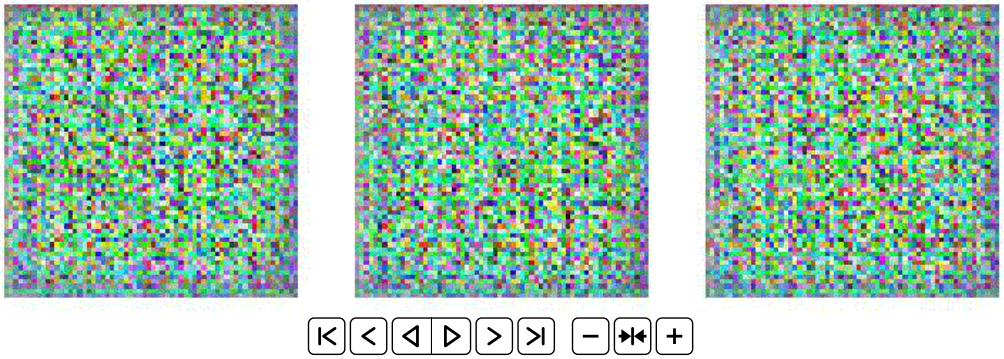


Figura 6.4: GIF con el entrenamiento iterativo del modelo **GAN_latentdim512_in**.

En el Apéndice A, serían las Figuras A.10, A.11 y A.12. Con esta misma configuración, entrenamos a otra arquitectura (**GAN_noncancer_latentdim512_in**) para tejido sano, obteniendo las Figuras A.13, A.14 y A.15 y el siguiente GIF:

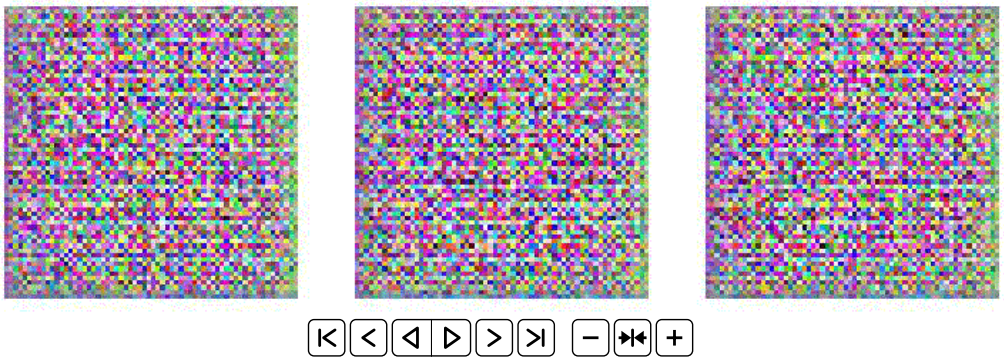


Figura 6.5: GIF con el entrenamiento iterativo del modelo **GAN_noncancer_latentdim512_in**.

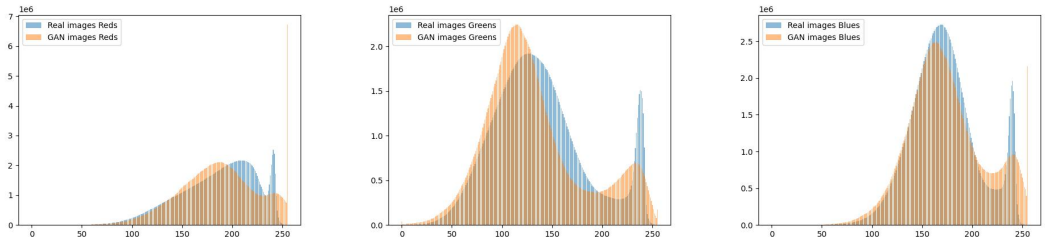
Con la GAN entrenada correctamente, pasamos a comentar los resultados a la hora de evaluar la calidad de estas imágenes generadas.

6.3 Evaluación de la calidad de la GAN

Como hemos comentado en la sección 5.3, el FID no es una métrica que nos sirva para medir la calidad de nuestra GAN. Sin embargo, hemos realizado el cálculo utilizando imágenes reales y generadas y cambiando su tamaño a 75x75. Con esto, se ha obtenido un FID de 728668.74, que es una distancia muy elevada y que no nos indica nada desgraciadamente.

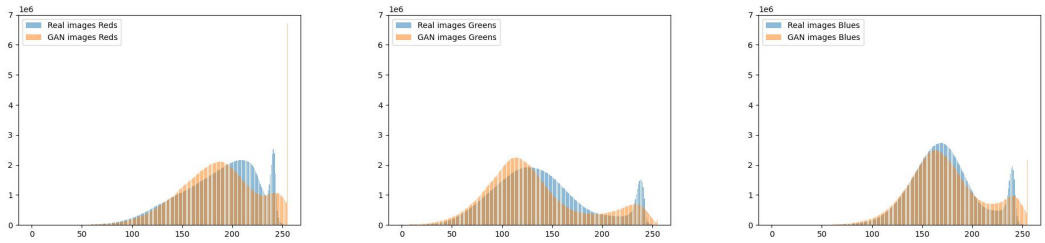
Por otro lado, a continuación se añaden histogramas de la distribución de valores en los diferentes píxeles RGB de las imágenes reales y sintéticas.

Si nos fijamos, estos histogramas no están normalizados en el eje Y, ya que el píxel 255 para el color rojo es el más repetido con diferencia, siendo bastante mayor al resto de valores, ya que en estas imágenes destacan los colores rosas y morados. A continuación añadimos pues estos histogramas normalizados.



(a) Píxeles rojos. **(b)** Píxeles verdes. **(c)** Píxeles azules.

Figura 6.6: Histograma de los píxeles de las imágenes reales y generadas.



(a) Píxeles rojos. **(b)** Píxeles verdes. **(c)** Píxeles azules.

Figura 6.7: Histograma normalizado de los píxeles de las imágenes reales y generadas.

Una vez normalizados, y algo que puede apreciarse en ambos es que las distribuciones de los píxeles son bastante similares, lo que indica que el color de las imágenes sí se parece. Esto se ve potenciado por el hecho de que el número de imágenes empleado en estos histogramas es significativo y las muestras no son pequeñas.

Por otro lado, y realizando la comparativa visual que comentaba en 5.3, tenemos la siguiente figura (o las Figuras A.16, A.17, A.18, A.19 y A.20):

(a) Imágenes generadas. **(b)** Imágenes reales.

Figura 6.8: Comparativa de las imágenes de tejido cancerígeno generadas y reales.

En ella se puede apreciar que los resultados son buenos, pero no perfectos, ya que la calidad y definición de las imágenes no es del todo igual. Ambas se ven pixeladas por ser parches pequeños de 50x50, pero las generadas tienen más píxeles que parecen pixelación cuando realmente serán fallos pequeños en el color del píxel o de la textura en sí.

6.4 Evaluación del dataset generado

En este último apartado vamos a comentar en primer lugar qué resultados se ha obtenido con el conjunto **Conjunto10KGen**, de forma que tenemos:

Como podemos observar, los resultados obtenidos no son buenos y comparándolos con la Tabla 6.2 y 6.3, podemos observar que comparado con el **Conjunto10K** no mejoramos los resultados con ningún modelo. Además, si nos comparamos con el **Conjunto5K**,

Figura 6.9: A figure

Figura 6.10: Another figure

Precisión para el Conjunto10KGen	**Model150K**	**Model300K**	**Model5M**	**Model6M**	**Model11M**
Conjunto de validación	86.2 ± 0.02	87.6 ± 0.02	86.5 ± 0.02	86.9 ± 0.02	88.3 ± 0.02
Conjunto de test	72.4 ± 0.02	85.6 ± 0.02	82.4 ± 0.02	80.6 ± 0.02	86.1 ± 0.02

Tabla 6.4: Tabla con los resultados del **Conjunto10KGen** en validación y test.

que tiene la misma cantidad de datos del conjunto original, tampoco tenemos resultados mejores.

Un detalle curioso de los experimentos con el conjunto original y con estos es que el entrenamiento con conjuntos con pocas muestras como el **Conjunto1K**, el **Conjunto2K** o el **Conjunto5K** producen un overfitting, mientras que con el resto se produce el efecto contrario. Este sería un ejemplo del entrenamiento del **Model5M** con el **Conjunto5K**:

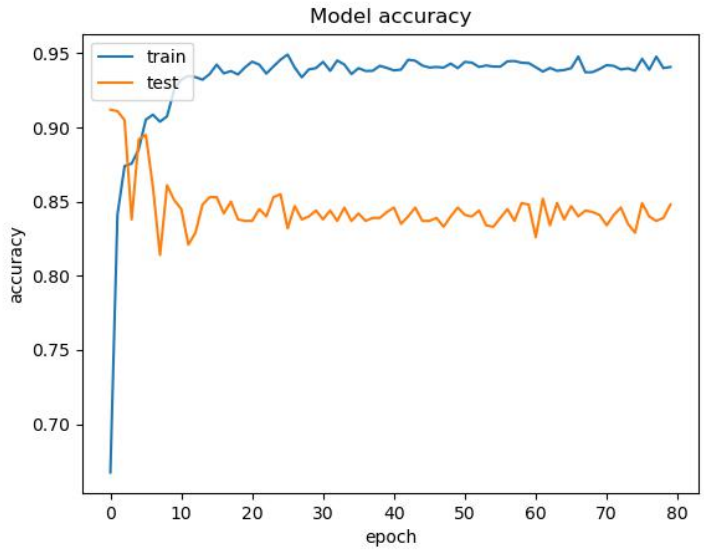


Figura 6.11: Gráfica con la precisión durante el entrenamiento para el **Conjunto5K**.

Como podemos observar, la precisión en train es mayor a partir del epoch 5 y se mantiene por encima a lo largo de todo el entrenamiento. Sin embargo, con el entrenamiento del **Model5M** con el **Conjunto10K** tenemos la siguiente gráfica:

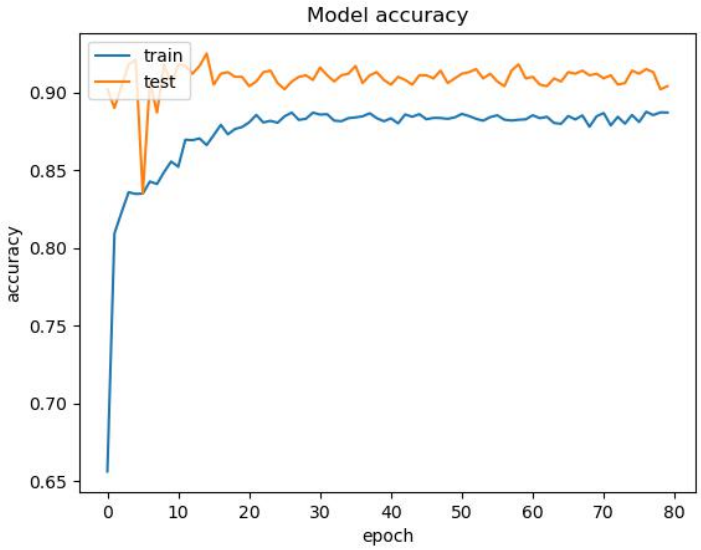


Figura 6.12: Gráfica con la precisión durante el entrenamiento para el **Conjunto10K**.

En este segundo caso, los resultados son muy distintos, ya que la precisión en el conjunto de validación es mayor en todo momento. Por desgracia, con el **Conjunto10KGen** tenemos el mismo resultado que en el primer caso, pero exagerado aún más porque los resultados en train son mejores y en validación peores. Esto podría deberse a que el conjunto solo tiene 5000 imágenes reales, pero tampoco podemos afirmarlo con total certeza. La gráfica del entrenamiento se muestra a continuación:

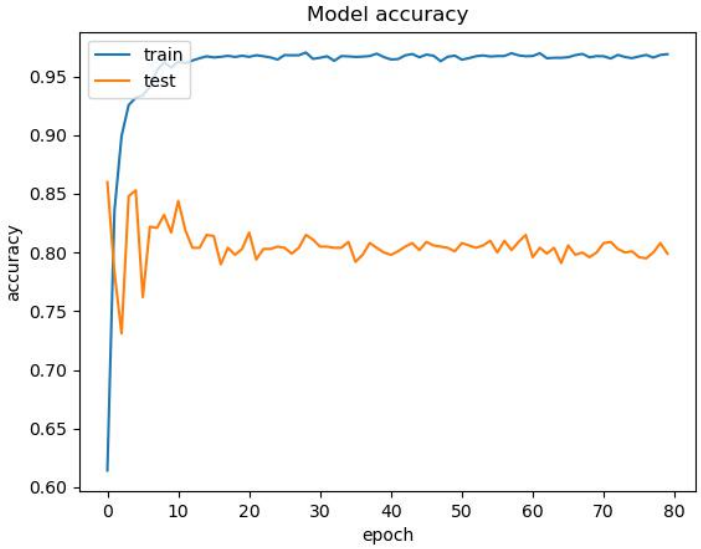


Figura 6.13: Gráfica con la precisión durante el entrenamiento para el **Conjunto10KGen**.

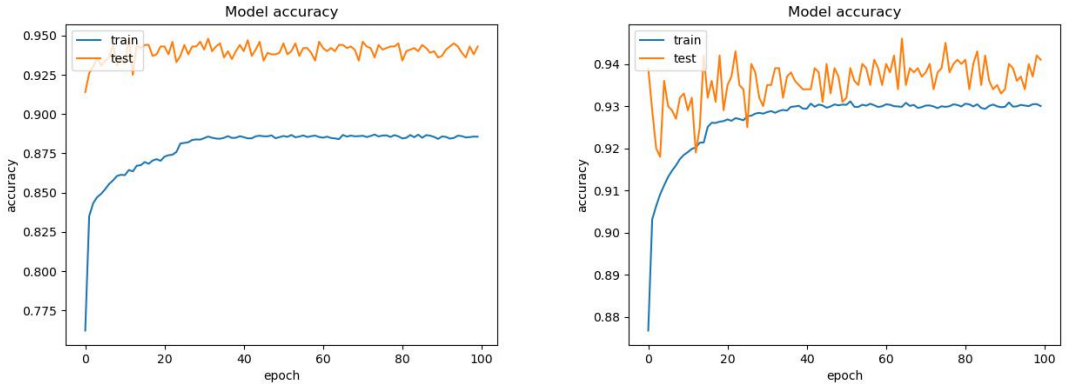
Una vez se ha comentado estos resultados, hemos realizado un entrenamiento con el **ConjuntoAllReal** y **ConjuntoAllGen** con los parámetros que se indicaban en la Tabla 5.5. En concreto, hemos utilizado el **Model11M**, ya que ofrecía muy buenos resultados sobre los conjuntos de datos originales tanto en validación como test, aumentando su rendi-

miento conforme aumentaba el número de datos, que nos interesa bastante con estos dos conjuntos que compararemos a continuación. Además, obtenía también el mejor resultado sobre el **Conjunto10KGen**, algo que nos interesa mucho al realizar experimentos sobre el **ConjuntoAllGen**. Dicho esto, y tras entrenar este modelo con ambos conjuntos, los resultados son los siguientes:

Precisión	**Validación**	**Test**
ConjuntoAllReal	94.0 ± 0.02	95.2 ± 0.02
ConjuntoAllGen	93.9 ± 0.02	94.7 ± 0.02

Tabla 6.5: Tabla con los resultados del **Model11M** en validación y test.

Estos resultados muestran que el aumento de los datos tan grande no ha servido para mejorar los resultados que ya se obtiene con el modelo entrenado con el conjunto original balanceado. De hecho, si observamos las gráficas del entrenamiento para ambos conjuntos que adjuntaremos a continuación, podemos comprobar que el entrenamiento con el **ConjuntoAllReal** es mucho más estable en la precisión en validación que con el **ConjuntoAllGen**, logrando obtener un resultado superior finalmente.



(a) Precisión del **ConjuntoAllOrig**.

(b) Precisión del **ConjuntoAllGen**.

Figura 6.14: Gráficas con la precisión durante el entrenamiento.

CAPÍTULO 7
Conclusiones y trabajo futuro

Finalmente, en este último capítulo se realizarán unas conclusiones del trabajo realizado y se comentarán diferentes aspectos que podrían realizarse como trabajo futuro.

7.1 Conclusiones

Como conclusiones generales, queremos destacar el trabajo realizado:

En primer lugar hemos realizado una exploración de un conjunto de datos de parches con tejido sano y cancerígeno para pacientes con Carcinoma Ductal Invasivo (CDI), para el cual se ha establecido unos resultados iniciales que servían como baseline para el trabajo posterior.

A continuación se ha entrenado una arquitectura GAN para generar imágenes con tejido con carcinoma a partir de 1000 imágenes del conjunto original y tras realizarlo con éxito, se ha entrenado otra arquitectura con los mismos parámetros para obtener una GAN que generara imágenes de tejido sano.

Una vez se han obtenido estas dos arquitecturas, se ha procedido a evaluar la calidad de sus imágenes generadas, que han demostrado ser de una calidad inferior a la que nos habría gustado.

Tras esto, hemos creado un conjunto de datos con muestras generadas (**Conjunto10KGen**) y hemos aplicado los mismos modelos que con los datos originales para compararlos con los resultados de la baseline. Estos resultados han sido inferiores a los obtenidos con el conjunto original y se ha comprobado que se tendía más a un entrenamiento inestable y a un overfitting para el conjunto de entrenamiento.

Finalmente, hemos entrenado con un conjunto de datos balanceado con todas las muestras posibles del conjunto original (**ConjuntoAllReal**) y otro con todas las muestras posibles generando muestras para convertirlo en balanceado (**ConjuntoAllGen**). Esto ha resultado en modelos con la misma precisión prácticamente a pesar de que el modelo con imágenes generadas tenía un total de 2.5 veces más datos para entrenar.

Todo esto nos muestra que hemos obtenido muestras sintéticas que a lo mejor no alcanzaban la calidad necesaria para llegar a aportar algo al entrenamiento de estos modelos. Además de esto, queremos destacar que cuando se trabajaba con conjuntos de datos pequeños, 1000 imágenes de test era un buen conjunto, pero cuando nos situamos en un contexto global, en el que de un solo individuo podemos obtener 2200 parches (como se muestra en la Figura 3.1), el conjunto ya no es tan grande ni representativo.

A pesar de todo esto, consideramos que se ha realizado un buen trabajo al obtener resultados bastante buenos para tratarse de la generación de imágenes histopatológicas, un

campo para el que las GAN no se han explorado excesivamente y siendo unas imágenes celulares con una estructura variable y una paleta de colores específica.

Tras esto, queremos ahora comentar algunas áreas dentro de este trabajo en las que podría profundizarse para poder continuarlo.

7.2 Trabajo Futuro

En concreto, consideramos que podrían realizarse las siguientes exploraciones:

- En primer lugar, podría realizarse más pruebas sobre el conjunto de datos original, ya que no se ha realizado una exploración muy profunda de diferentes modelos para obtener los mejores resultados posibles.
- En segundo lugar, consideramos que podría utilizarse una arquitectura GAN más sofisticada y actual para conseguir imágenes generadas de mayor calidad, que no tengan ese pixelado que tenían nuestras imágenes más allá de ser imágenes pequeñas.
- Por último, nos ha faltado realizar más pruebas sobre los conjuntos con imágenes generadas, ya que no sabemos si cambiando diferentes parámetros o con otras configuraciones habríamos conseguido mejorar los resultados establecidos en la baseline.

Bibliografía

[1] Fnu Amisha, Monika Pathania, and Vyas Rathaur. Overview of artificial intelligence in medicine. *Journal of Family Medicine and Primary Care*, 8:2328, 07 2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

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

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

[5] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[6] Dakai Jin, Adam Harrison, Ling Zhang, Ke Yan, Yirui Wang, Jinzheng Cai, Shun Miao, and Le Lu. *Artificial intelligence in radiology*, pages 265–289. 01 2021.

[7] Tasleem Kausar, Adeeba Kausar, Muhammad Adnan Ashraf, Muhammad Farhan Siddique, Mingjiang Wang, Muhammad Sajid, Muhammad Zeeshan Siddique, Anwar Ul Haq, and Imran Riaz. SA-GAN: Stain acclimation generative adversarial network for histopathology image analysis. *Applied Sciences*, 12(1):288, dec 2021.

[8] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2016.

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[11] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[12] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis, 2017.

APÉNDICE A

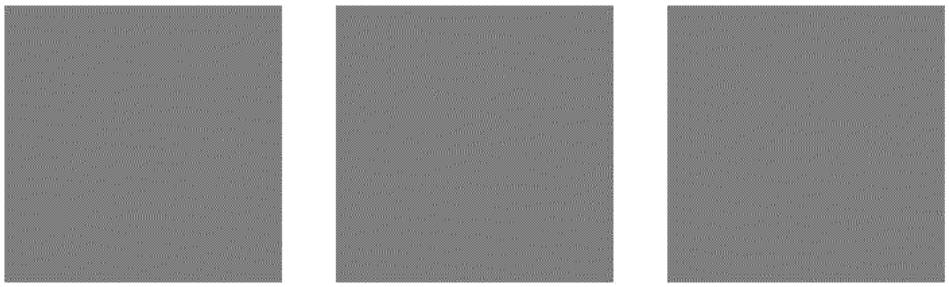


Figura A.1: Imagen de la salida del generador en la primera iteración con el modelo **GAN_latentdim100_bn**.

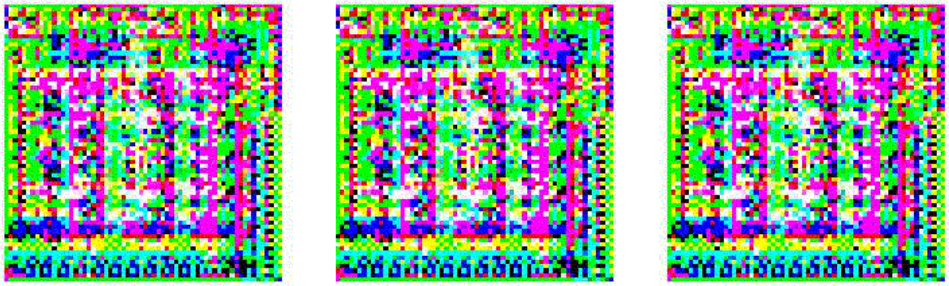


Figura A.2: Imagen de la salida del generador en la iteración de la mitad del entrenamiento con el modelo **GAN_latentdim100_bn**.

Figura A.3: Imagen de la salida del generador en la última iteración con el modelo **GAN_latentdim100_bn**.

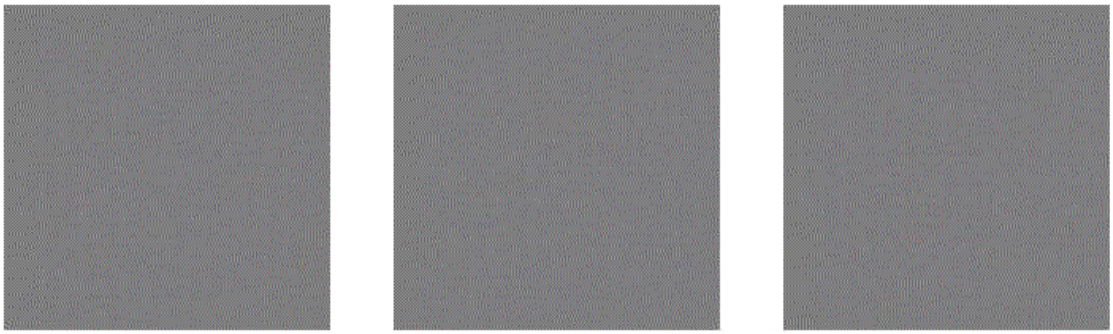


Figura A.4: Imagen de la salida del generador en la primera iteración con el modelo **GAN_latentdim512_drop_bn**.

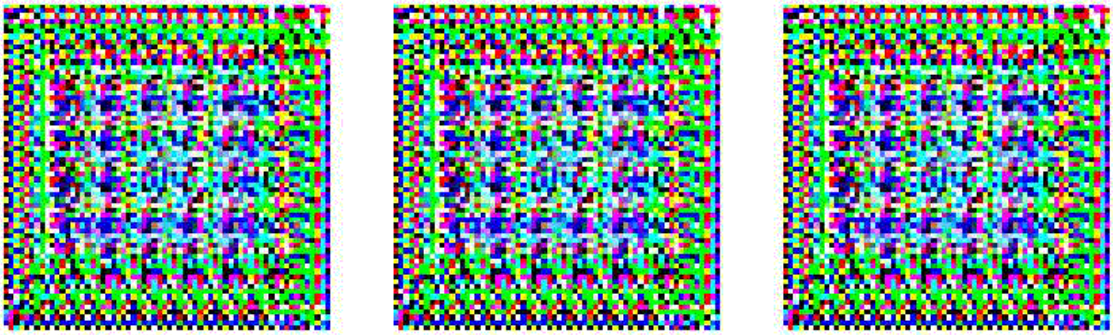


Figura A.5: Imagen de la salida del generador en la iteración de la mitad del entrenamiento con el modelo **GAN_latentdim512_drop_bn**.

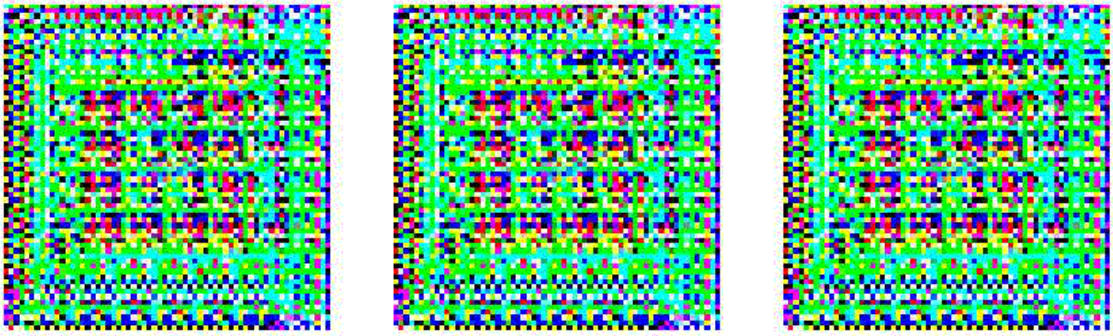


Figura A.6: Imagen de la salida del generador en la última iteración con el modelo **GAN_latentdim512_drop_bn**.

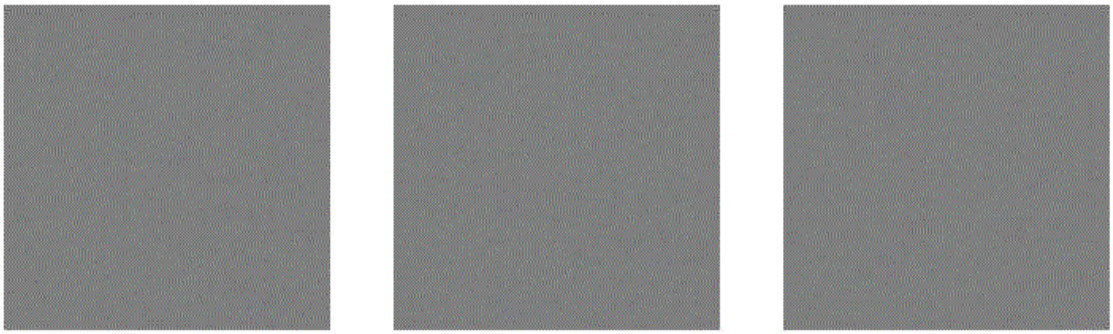


Figura A.7: Imagen de la salida del generador en la primera iteración con el modelo **GAN_latentdim512_drop**.

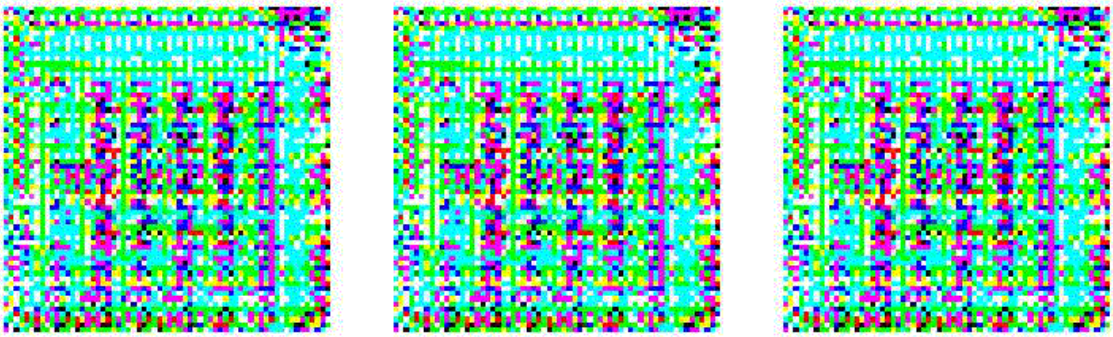


Figura A.8: Imagen de la salida del generador en la iteración de la mitad del entrenamiento con el modelo **GAN_latentdim512_drop**.

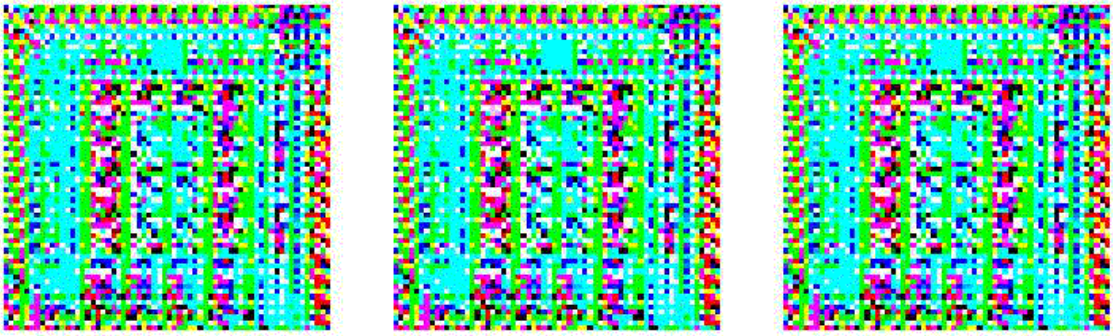


Figura A.9: Imagen de la salida del generador en la última iteración con el modelo **GAN_latentdim512_drop**.

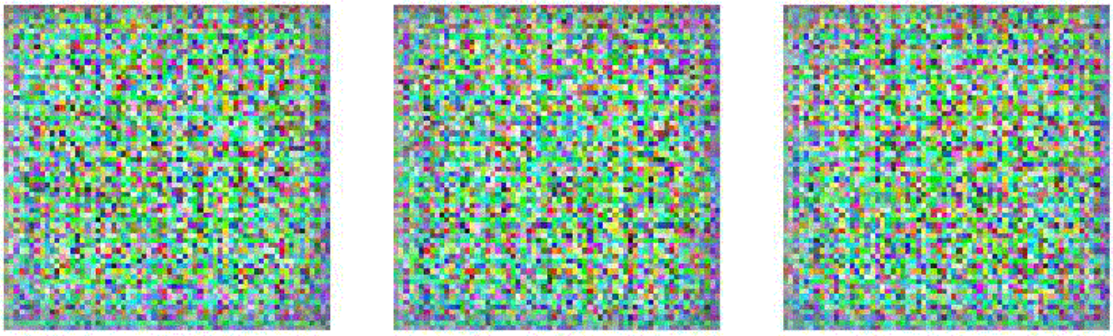


Figura A.10: Imagen de la salida del generador en la primera iteración con el modelo **GAN_latentdim512_in**.

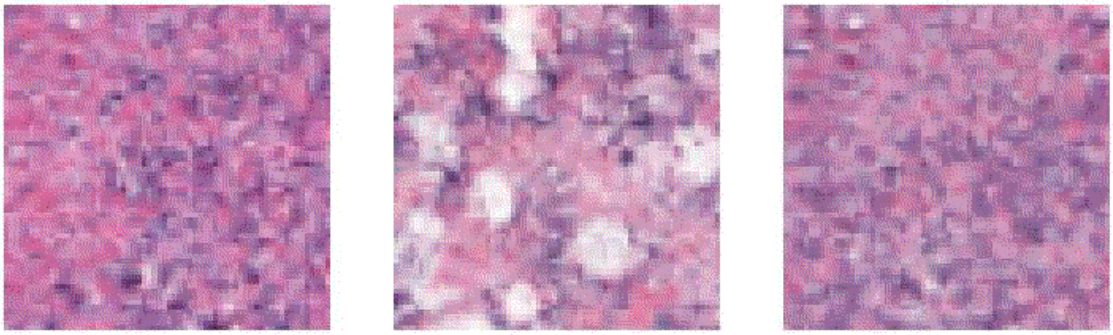


Figura A.11: Imagen de la salida del generador en la iteración de la mitad del entrenamiento con el modelo **GAN_latentdim512_in**.

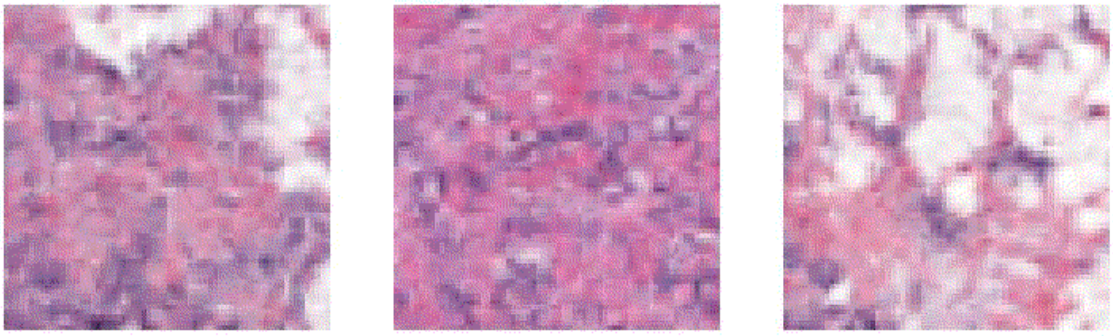


Figura A.12: Imagen de la salida del generador en la última iteración con el modelo **GAN_latentdim512_in**.

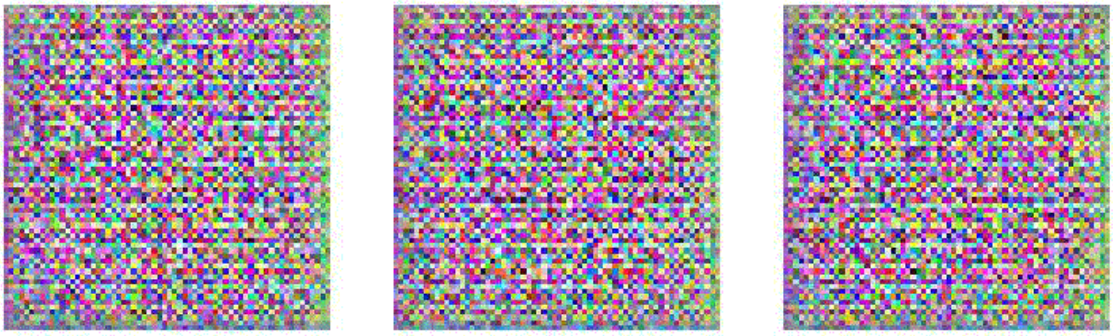


Figura A.13: Imagen de la salida del generador en la primera iteración con el modelo **GAN_noncancer_latentdim512_in**.

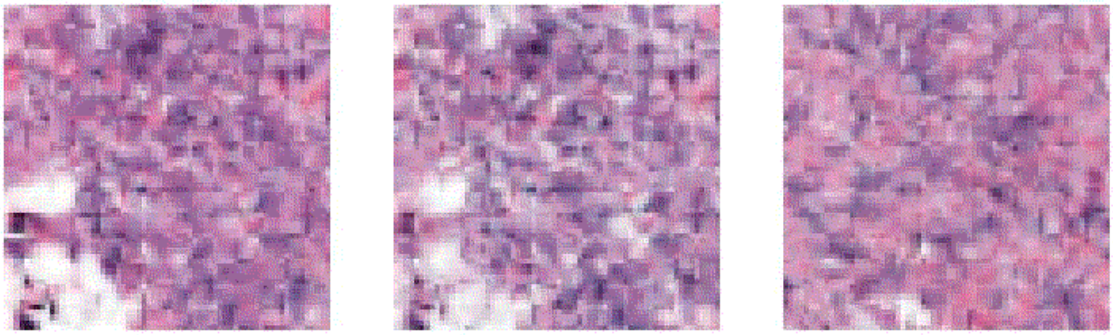


Figura A.14: Imagen de la salida del generador en la iteración de la mitad del entrenamiento con el modelo **GAN_noncancer_latentdim512_in**.

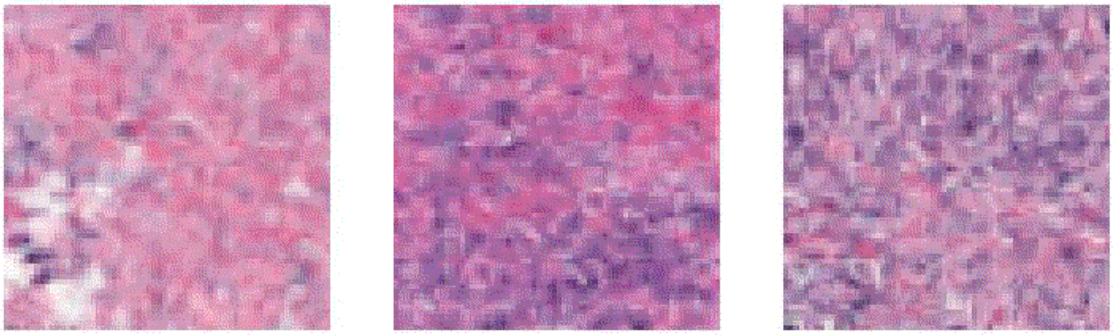


Figura A.15: Imagen de la salida del generador en la última iteración con el modelo **GAN_noncancer_latentdim512_in**.

Figura A.16: Comparativa 1 de dos imágenes de tejido cancerígeno generada y real respectivamente.

Figura A.17: Comparativa 2 de dos imágenes de tejido cancerígeno generada y real respectivamente.

Figura A.18: Comparativa 3 de dos imágenes de tejido cancerígeno generada y real respectivamente.

Figura A.19: Comparativa 4 de dos imágenes de tejido cancerígeno generada y real respectivamente.

Figura A.20: Comparativa 5 de dos imágenes de tejido cancerígeno generada y real respectivamente.