Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

[Improving Breast Cancer Detection Using DCGANs Generated Synthetic Mammograms]

Trabajo Fin de Máster

Presentado por: García Tíscar, Luis

Director/a: Antolínez García, Alfonso

Ciudad: Orihuela
Fecha: 11/09/2021

Resumen

En el campo del Aprendizaje Automático, existe un compromiso entre la cantidad de datos con los que se entrena un modelo y los resultados del mismo. Uno de los principales problemas de la adquisición de datos son las anomalías, los valores erróneos o, principalmente, faltantes. En aplicaciones médicas, como el análisis de mamografías para la detección de tumores, estas anomalías pueden mermar el rendimiento del modelo y repercutir directamente en la vida de los pacientes, debido a un diagnóstico erróneo. Para subsanar este problema, presentamos un desarrollo de software capaz de generar mamografías sintéticas de alta resolución con presencia de tumores benignos y malignos a través de redes neuronales DCGAN (Deep Convolutional Generative Adversarial Networks). El objetivo es comprobar si estas imágenes sintéticas son clasificadas correctamente como tumores benignos o malignos por los clasificadores de cáncer de mama del estado del arte.

Palabras Clave: Mamografía, DCGAN, Redes Neuronales, Aprendizaje Automático

Abstract

The amount of available training data and a Machine Learning model’s results are strongly related. Missing values are one of the main issues in this field, making models lose accuracy and overall quality in their results. Machine Learning models, especially in healthcare applications, need to be very accurate, because poor results lead to misdiagnosis and can affect patients' health. In order to solve this issue, we present an unsupervised ML model to generate high resolution synthetic mammograms based on DCGAN (Deep Convolutional Generative Adversarial Networks). Our goal is to create synthetic mammograms, indistinguishable from real ones, to feed these synthetic images to state-of-the-art breast cancer detection models and check if they are treated as real mammograms.

Keywords: Machine Learning, Breast Cancer, DCGAN, Mammogram.

Índice de contenidos

Índice de tablas

Índice de figuras

1 Introducción

1.1 Motivación

En los datos recogidos mediante series temporales (*time series* en inglés), suele haber fallos en el registro que dan lugar a valores faltantes (missing values, NaN, None) en los datasets. Para solucionar este problema, se desarrollaron diversas metodologías de *data imputation*, es decir, métodos para "rellenar" estos valores faltantes o erróneos con datos sintéticos similares al resto de datos correctos. En las bases de datos de imágenes médicas, el problema es otro: pocas de estas bases son públicas o tienen suficientes imágenes disponibles.

A veces, pedir acceso a estas bases es un proceso largo y complicado, llegando en muchos casos a encontrarse con negativas de los hospitales o centros que guardan las bases. Creo que el libre acceso a los recursos para la comunidad científica es un aspecto clave del progreso en la era de Internet.

En Aprendizaje Automático y Aprendizaje Profundo, sobre todo, la cantidad de datos disponibles para entrenar un modelo es clave para el buen funcionamiento del mismo. Cuantos más y de mejor calidad sean los datos, mejores serán los resultados.

1.2 Planteamiento del trabajo

Gracias al auge de las **GAN**, ***Generative Adversarial Networks,*** (Goodfellow et al., 2014), como fuente de infinidad de proyectos creativos, llegamos a la conclusión de que debería ser posible usar estas redes neuronales para ofrecer una base de datos de imágenes médicas sintéticas de uso libre y gratuito para la comunidad científica y cualquier usuario que lo requiera.

Proponemos generar una base de datos de imágenes sintéticas de alta resolución mediante el uso de un tipo mejorado de **DCGAN** (Radford et al., 2015), la ***StyleGAN 2 ADA*** (Karras et al., 2020). Entrenaremos dos modelos: uno con bases de datos públicas de mamografías con presencia de tumores, tanto benignos como malignos y el otro, libre de tumores. Decidimos dedicarnos principalmente a las mamografías con presencia de tumores porque creemos que es importante proveer a la comunidad de imágenes con este tipo de imágenes, ya que la mayoría de datasets públicos, aparte de tener pocas muestras, normalmente están desbalanceados debido a que la mayoría de las muestras de mamografías no tienen presencia de tumores.

Una vez hayamos entrenado nuestro modelo personalizado para generar imágenes JPG de resolución 512x512 de mamografías sintéticas, usaremos modelos de aprendizaje automático de clasificación/predicción de cáncer de mama del estado del arte para comprobar si realmente trata las imágenes que hemos generado como reales, y sirven para el entrenamiento de estos modelos.

1.3 Estructura de la memoria

El presente documento se estructura de la siguiente manera:

Capítulo 2. En él se aborda el estado del arte, donde realizamos una revisión de los últimos avances, soluciones y metodologías para el problema planteado, así como los resultados de dichas soluciones.

Capítulo 3. Objetivos y metodología de trabajo, donde marcamos los objetivos finales e intermedios que queremos alcanzar con nuestra solución, y la metodología que llevamos a cabo para conseguirlos.

Capítulo 4. Identificación de requisitos, donde especificamos los requisitos necesarios para el uso del software en cuanto a las imágenes de entrenamiento y generación, los lenguajes de programación necesarios y los recursos a los que accedimos para el desarrollo del software aquí presentado.

Capítulo 5. Descripción de la herramienta software desarrollada, explicamos el software desarrollado en su versión final e indicaciones de cómo usarlo para generar imágenes sintéticas de principio a fin.

Capítulo 6. Evaluación, presentamos los resultados alcanzados, así como las evaluaciones pertinentes para comprobar la calidad de dichos resultados, tanto de forma numérica como gráfica.

Capítulo 7. Conclusiones, discutiremos los resultados obtenidos, compararemos con los resultados de los trabajos similares del estado del arte y presentaremos futuras líneas de investigación que podrían realizarse a partir de los resultados de este proyecto.

2 Contexto y estado del arte

2.1 Arquitecturas GAN

Las GAN son un tipo de sistema de redes neuronales compuestas por dos modelos de red neuronal: un Generador y un Discriminador:

El Generador es una red neuronal que tiene como entrada un vector numérico, z, que sigue una distribución aleatoria normal(0,1). A partir de este vector que llamaremos vector latente, empieza a generar imágenes de mayor resolución mediante el uso de capas convolucionales transpuestas. Estas capas aumentan la resolución de una matriz de entrada según el tamaño del filtro que apliquen. Se va aumentando la resolución hasta un máximo, tal como puede verse en la figura 1, la x̂. Esta matriz se pasa al Discriminador junto con una matriz extraída de imágenes del dataset de imágenes de entrenamiento. En el Discriminador, se usan capas convolucionales que, al contrario que las transpuestas, reducen la dimensión de la matriz progresivamente hasta la capa de salida de una neurona: 0 o 1 (es real o es sintética). Una vez el Discriminador clasifica la matriz x̂ como real o sintética, se utiliza backpropagation para proporcionar feedback al Generador de lo bien o mal que ha generado la matriz, comparada con la matriz de imagen real, para que actualice los pesos de sus capas ocultas. Un modelo GAN se entiende que funciona correctamente cuando la pérdida del Generador disminuye progresivamente y la del Discriminador oscila entre el 0.5. Llegado a este punto, no es capaz de discernir si una imagen es real o sintética, y asigna al azar, como lanzar una moneda al aire.

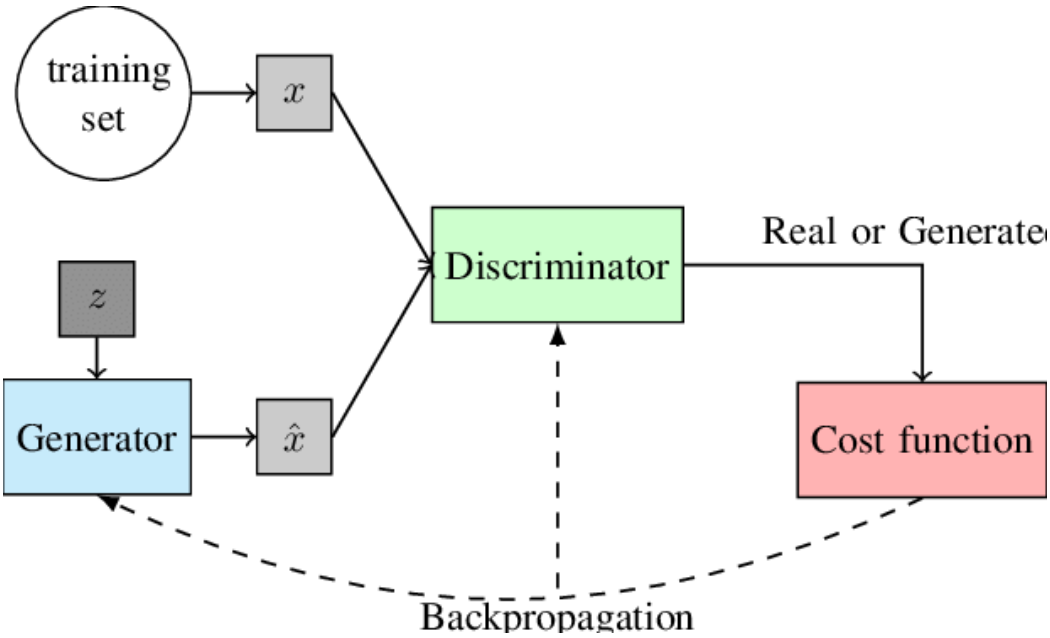


Figura 1: Arquitectura esquematizada del modelo original **GAN**. (Ponti, Sampaio Ferraz Ribeiro, Santana Nazare, & Bui, 2017)

Desde el origen de las GAN su aplicación ha sido llevada a numerosos campos: **style transfer** (Xu et al., 2019), **image to image translation** (Isola et al., 2016), **text to image translation** (Krishna Goti & Ma, 2018) y, la aplicación que más nos interesa: la capacidad de generar contenido totalmente sintético a partir de contenido real. Ejemplos de esta última aplicación son: **generación de imágenes** (Radford et al., 2015), **generación de texto** (Yuan et al., 2020) o la brillante idea de generación y combinación de música multiestilo de **MuseNET** (Payne, 2019).

Uno de los modelos que más llamó nuestra atención es **DCGAN** (Radford et al., 2015), cuya arquitectura puede verse en la siguiente figura (Solo presentaron la del Generador, ya que el discriminador es simétrico y está colocado a continuación del Generador):

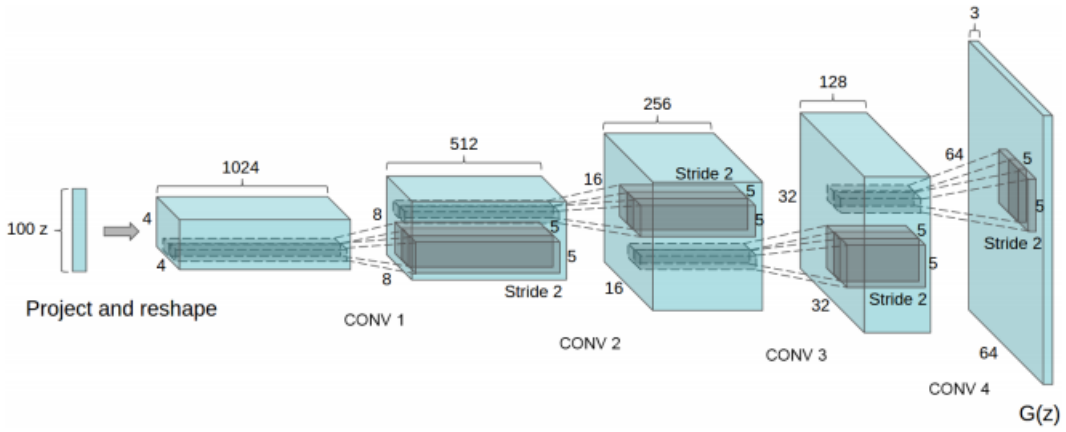


Figura 2: Arquitectura del Generador **DCGAN**. (Radford, Metz, & Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015)

Las GAN originales presentaban resultados irregulares y poco coherentes en muchas de las aplicaciones. Las DCGAN, usando el Discriminador como extractor de características de las muestras de entrenamiento y pasando estas características al Generador para que las aprendiese, conseguía mejorar los resultados. Sin embargo, existía una limitación adicional: la resolución.

Esta limitación fue resuelta con la introducción de **proGAN** (Karras et al., 2018).Hasta entonces, las DCGAN solo eran capaces de generar imágenes de pequeña resolución (hasta 64x64 píxeles, como denota la figura 2). Esta limitación se ve resuelta cuando el equipo de investigadores de nVIDIA contribuyó con la proGAN. La idea era “sencilla”: si hasta entonces, las capas de convolución transpuesta del generador aumentaban la resolución de la matriz latente, sería lógico pensar que, añadiendo más capas ocultas de este tipo, aumentaría la resolución final de la imagen generada. Su solución consistió en diseñar un modelo que añadía capas ocultas convolucionales transpuestas en cada época para aumentar la resolución progresivamente. Así, el generador centraba su atención en aprender a generar resoluciones mayores en una primera fase, y luego cambiaba el foco de atención en mejorar los detalles de la imagen resultante. Una vez que genera imágenes con buen nivel de detalle a una resolución determinada, se pasa a una resolución mayor, y así progresivamente hasta llegar a los resultados finales de la mayor resolución permitida en el modelo.

Generator	Act.	Output shape	Params
Latent vector	–	512 × 1 × 1	–
Conv 4 × 4	LReLU	512 × 4 × 4	4.2M
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Upsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Upsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Upsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Upsample	–	512 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	1.2M
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Upsample	–	256 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	295k
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Upsample	–	128 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	74k
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Upsample	–	64 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	18k
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Upsample	–	32 × 1024 × 1024	–
Conv 3 × 3	LReLU	16 × 1024 × 1024	4.6k
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 1 × 1	linear	3 × 1024 × 1024	51
Total trainable parameters			**23.1M**

Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			**23.1M**

Figura 3: Arquitectura de **proGAN**. (Karras, Aila, Laine, & Lehtinen, 2018)

Esta arquitectura, aunque mejoraba el funcionamiento de las DCGAN, tenía un problema, al generar imágenes de más resolución, era más fácil para el discriminador determinar si era real o sintética.

Para solucionar también este problema, el equipo liderado por Karras desarrolló el modelo **StyleGAN** (Karras et al., 2018). En este artículo, denunciaban la falta de conocimiento sobre cómo el generador y el vector latente influyen en el resultado final de la imagen generada. Por ello, investigaron cómo modificar los datos de entrada del sistema y las capas internas del generador. Entre otras mejoras, la principal característica es que el vector latente ya no es la entrada directa del generador, sino que se pasa por una primera capa de normalización y por una serie de capas densas (Fully Connected, o FC, en inglés) para formar un estado latente intermedio *W*, con el que se alimenta al generador después de cada capa de convolución. Además, también se inyecta el generador con ruido escalado por un factor de aprendizaje por canal. En la siguiente figura se ilustran los cambios aplicados al generador, donde (a) es la arquitectura tradicional de una GAN y (b) la arquitectura propuesta:

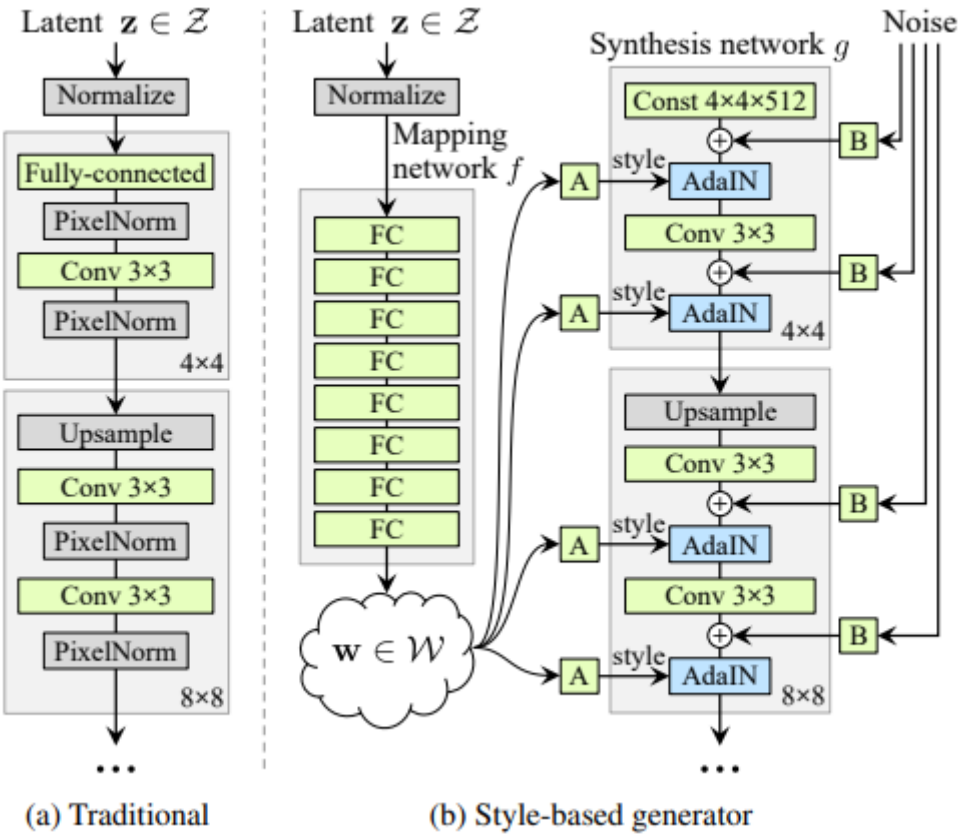


Figura 4: Comparación arquitectura generador tradicional GAN vs **StyleGAN**. (Karras, Laine, & Aila, A Style-Based Generator Architecture for Generative Adversarial Networks, 2018)

Estos cambios, según los autores, fueron inspirados en las arquitecturas de las redes neuronales de transferencia de estilo.

En 2019, estos investigadores consiguieron mejorar aún más el modelo para conseguir más calidad en las imágenes generadas, dando lugar a **StyleGAN 2** (Karras et al., 2019). StyleGAN 1 presentaba ciertos artefactos en las imágenes generadas:

Figure 5: Artefactos en imágenes generadas por **StyleGAN 1**. (Karras, Laine, Aittala, & Hellsten, 2019)

En esta nueva versión, los investigadores cambiaron la forma del estado intermedio latente W, que era el causante de estos artefactos. Lo solucionaron separando la operación de normalización de la de modulación, dentro del bloque ADA-IN de la anterior arquitectura:

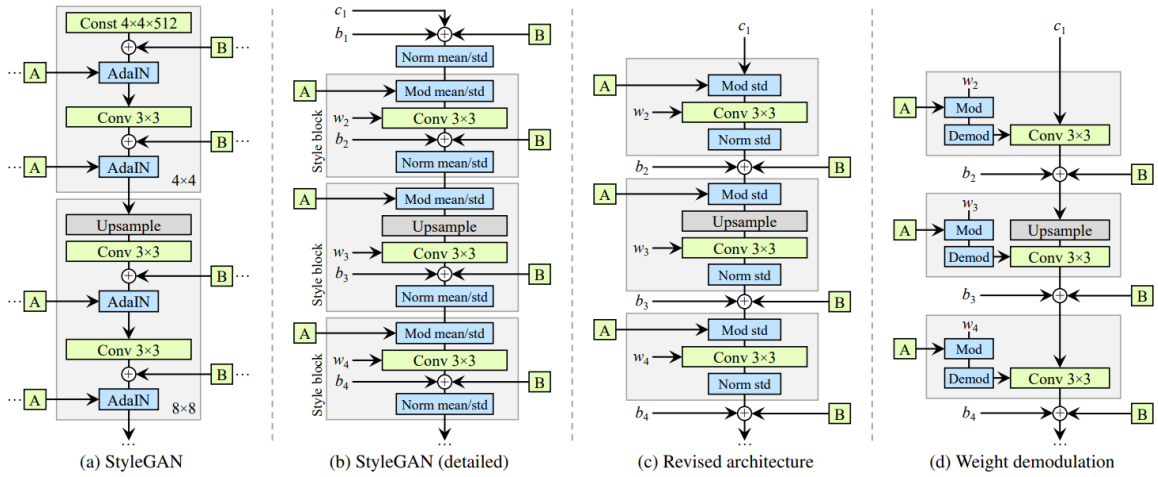


Figura 6: Comparación arquitecturas de generador entre StyleGAN y StyleGAN 2. (Karras, Laine, Aittala, & Hellsten, 2019)

El segundo de los artefactos era que el diseño de una arquitectura progresiva para la GAN provocaba una fuerte dependencia en la localización de detalles importantes como ojos y dientes, provocando que, aunque la imagen de la cara se moviera, StyleGAN dejaba intactos la localización de estas partes de la imagen:

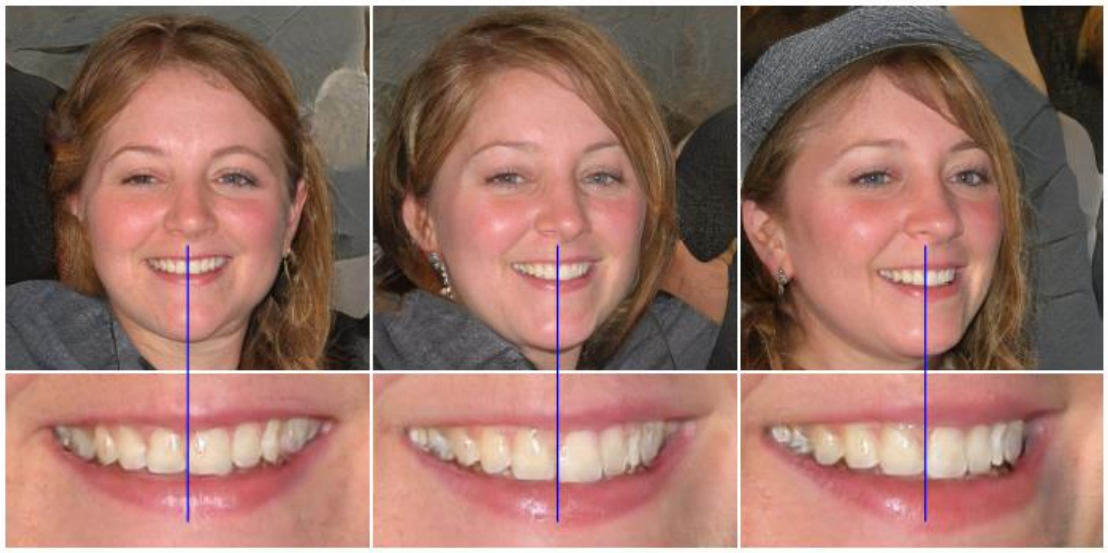


Figura 7 :Errores de “fase” (*Phase errors*, en el paper). (Karras, Laine, Aittala, & Hellsten, 2019)

Solucionaron este artefacto cambiando la estructura de conexión entre capas de la red, probando con conexiones de salto (skip connections), redes residuales y métodos jerárquicos.

Finalmente, el modelo que hemos usado para generar nuestras imágenes sintéticas es el **StyleGAN 2 ADA** (Karras et al., 2020). Dado que, como hemos detallado anteriormente, los datasets de imágenes médicas disponibles cuentan con pocos miles de muestras, esta nueva versión mejora notablemente los resultados de los modelos, ya que aplica técnicas de data augmentation con imágenes, como las de la figura 8:

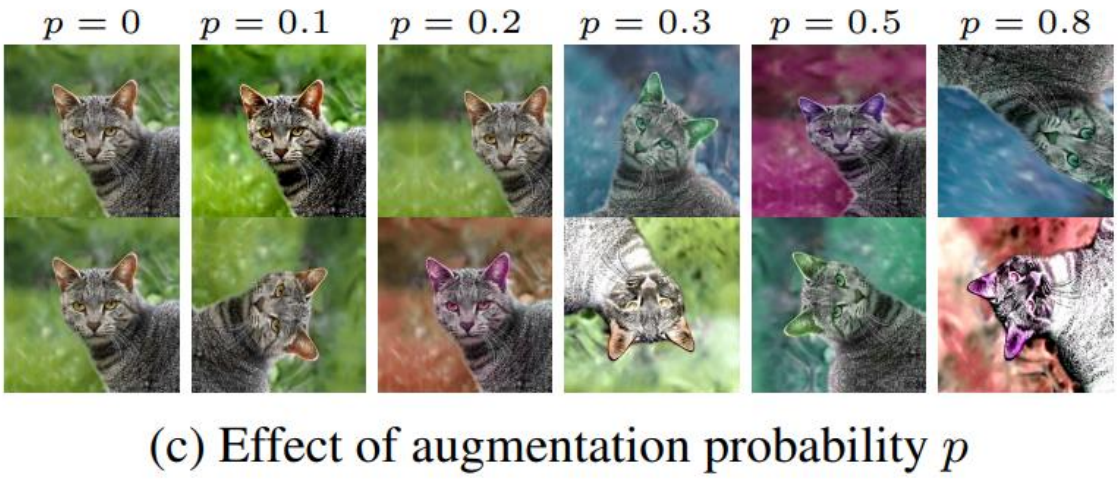


Figura 8: Técnicas de data augmentation de **StyleGAN 2 ADA**. (Karras, y otros, 2020)

Dado que nuestras imágenes son en escala de grises, no harán falta las implementaciones de las técnicas que modifican el color, solamente las de rotaciones de la imagen.

2.2 Trabajos similares

En cuanto a los trabajos similares, uno de los que más se acerca a nuestro objetivo es el de (Alyafi et al., 2019). Los investigadores desarrollaron un modelo de generación de “parches”, imágenes de 128x128 píxeles, estas imágenes corresponden con zonas de tejidos con presencia de tumores, benignos y malignos, en radiografías de pecho:

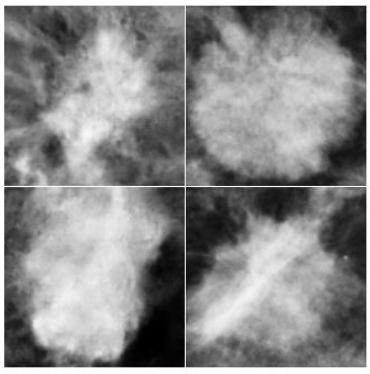


Figura 9: Parches generados (izquierda) comparados con los tumores reales (derecha). (Alyafi, Diaz, & Martí, 2019)

Nuestro objetivo se diferencia con el de ellos en el tamaño y calidad de las imágenes generadas, así como en el contenido, ya que nos centramos en obtener imágenes sintéticas de toda la mamografía al completo:

Figura 10: Ejemplo de mamografía real del dataset de entrenamiento. (Cheddad)

Como detallan en el artículo, usaron una red DCGAN para generar los parches de 128x128, algo razonable ya que solo tuvieron que añadir una capa convolucional extra, y los resultados no eran lo suficientemente grandes en resolución como para desbalancear la relación Generador-Discriminador.

Otros proyectos similares son (Cha et al., 2019) y (Wu et al., 2018). El primero genera imágenes mediante técnicas 3D, que requieren un conocimiento profundo de modelado 3D y el segundo crea parches de tumores como los creados por (Alyafi et al., 2019), y los integra en imágenes de mamografías reales, una solución bastante elegante y efectiva, teniendo en cuenta que los resultados de las redes GAN, por aquel entonces, no superaban los 128x128, excepto StyleGAN:

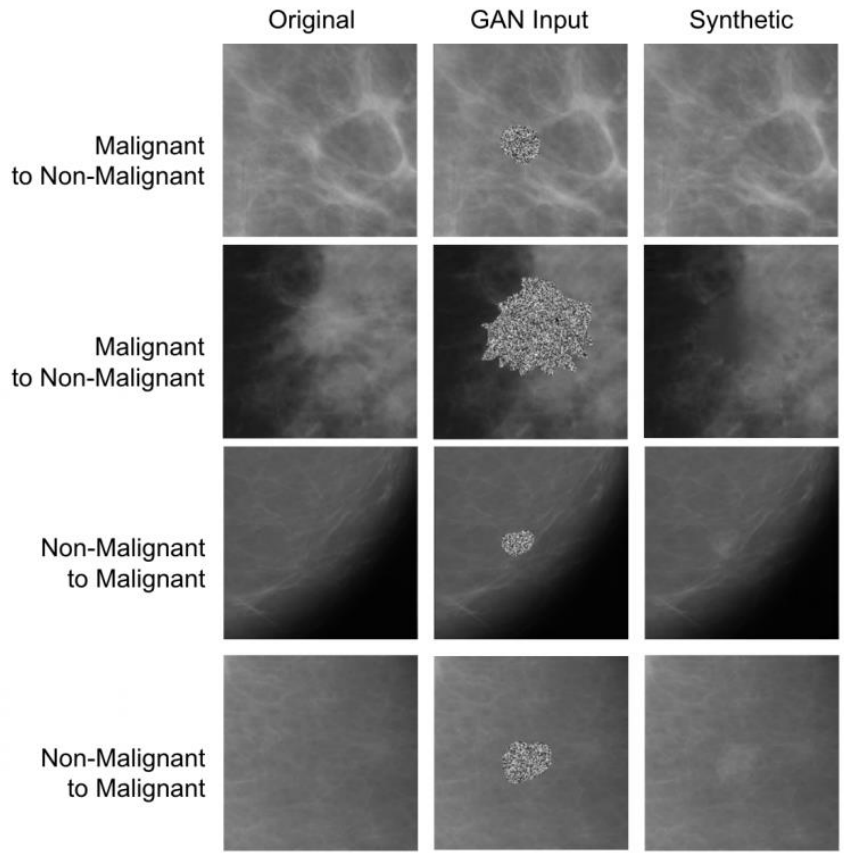


Figura 11: Ejemplos de integración de tumores generados en mamografías reales. (Wu, Wu, Cox, & Lotter, 2018)

Por lo tanto, podemos afirmar que nuestro proyecto proporciona una nueva vía innovadora y sencilla para la generación de imágenes sintéticas de mamografías con presencia de tumores, tanto benignos y malignos, usando un modelo de red neuronal cuya calidad ya ha sido probada en multitud de proyectos.

3 Objetivos y metodología de trabajo

3.1 Objetivo general

El objetivo general es, por tanto, conseguir que un clasificador del estado del arte entrenado con nuestros datasets de imágenes generadas sintéticamente alcance unos resultados más precisos que los entrenados con datasets públicamente disponibles desbalanceados.

Para ello, desarrollaremos un software capaz de usar el modelo de StyleGAN 2 ADA para la generación de una base de datos de mamografías sintéticas, tanto con presencia de tumores como sin ellos. Cuando evaluemos la calidad de las imágenes mediante métricas adecuadas, generaremos varios datasets de imágenes exclusivamente sintéticas siguiendo diferentes distribuciones: partiremos desde la más usual (70% imágenes normales – 30% con presencia de tumores) e iremos aumentando el porcentaje de presencia de tumores y disminuyendo el de normales. Entrenaremos un modelo de clasificación de imágenes del estado del arte con dichos datasets para, seguidamente, ejecutarlo sobre una muestra de un dataset real y comprobar si mejoran los resultados.

Podemos afirmar que se trata de un objetivo viable, ya que el desarrollo de este software es muy asequible. En nuestro caso, hemos usado la plataforma Google Colab en su versión PRO, que cuesta unos 10$ al mes. Es relevante en cuanto a que las demás técnicas de generación de mamografías sintéticas con presencia de tumores son complicadas o presentan poca resolución. Y, además, se puede generar una base de datos en un tiempo razonable, contando el tiempo de entrenamiento del modelo y la generación de dicha base de datos.

3.2 Objetivos específicos

Los objetivos específicos que se desean alcanzar con este proyecto son:

- Analizar los modelos disponibles de generación de imágenes basados en redes GAN.
- Implementar el modelo elegido en un Google Colab Notebook.
- Entrenar el modelo con un dataset público de mamografías.
- Validar el modelo mediante métricas pertinentes y el resultado de un clasificador del estado del arte sobre las imágenes generadas y originales.

3.3 Metodología del trabajo

En cuanto a la metodología, los pasos a seguir son los siguientes:

- **Implementar el modelo elegido en un Google Colab Notebook**: Hemos elegido usar la plataforma de Google porque da acceso a máquinas alojadas con GPUs potentes y memorias RAM de gran tamaño.

Con el fin de reducir los tiempos de cómputo, se ha optado por pagar una licencia PRO para conseguir tiempos de ejecución ininterrumpidos de 24h, lo que facilita las cosas a la hora de entrenar el modelo.

- **Elegir un dataset público y prepararlo adecuadamente:** Elegimos el dataset **DDSM** (Dissanayake Lekamlage, Afzal, Westerberg, & Cheddad, 2020). Para poder integrarlo en nuestro modelo, nos vimos obligados a editar las imágenes una por una mediante un modelo de edición de imágenes de (Schultz, n.d.) en GitHub, dado que las dimensiones de las imágenes originales eran rectangulares y StyleGAN 2 ADA requiere que las imágenes de entrada sean cuadradas.
- **Entrenar el modelo con imágenes del dataset preparado:** Entrenamos el modelo usando Google Colab PRO, aplicando la técnica de Transfer Learning desde el modelo preentrenado FFHQ512, disponible en el repositorio de StyleGAN 2 ADA. Las especificaciones de la máquina alojada del Google Colab PRO notebook son las siguientes:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.19.01    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   37C    P0    25W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figura 12: Especificaciones de la máquina alojada de Google Colab PRO usada para entrenar el modelo. (Imagen propia)

- **Comparar las imágenes generadas con las reales mediante métricas pertinentes.** Igual que los investigadores de nVIDIA, usaremos una métrica fiable para comprobar los resultados de las imágenes generadas: **Frechet inception distance (FID)** (Heusel et al., 2018).
- **Entrenar un clasificador del estado del arte con las imágenes generadas.** Entrenar con las imágenes generadas un clasificador del estado del arte, **VGG16** (Simonyan & Zisserman, 2015). Empezaremos entrenando el modelo solo con imágenes sintéticas generadas como resultado base, para después mejorar el mismo modificando el modelo o mezclando con imágenes reales.
- **Verificar los resultados del clasificador y modificar los parámetros del modelo para mejorar su eficacia.** Según los resultados del clasificador, modificaremos los parámetros del modelo como las capas congeladas o reentrenadas. Además, si los resultados del clasificador entrenado solo con imágenes generadas sintéticas no son buenos, mezclaremos las imágenes generadas con reales en diferentes proporciones. Este paso será recursivo hasta obtener unos resultados aceptables.

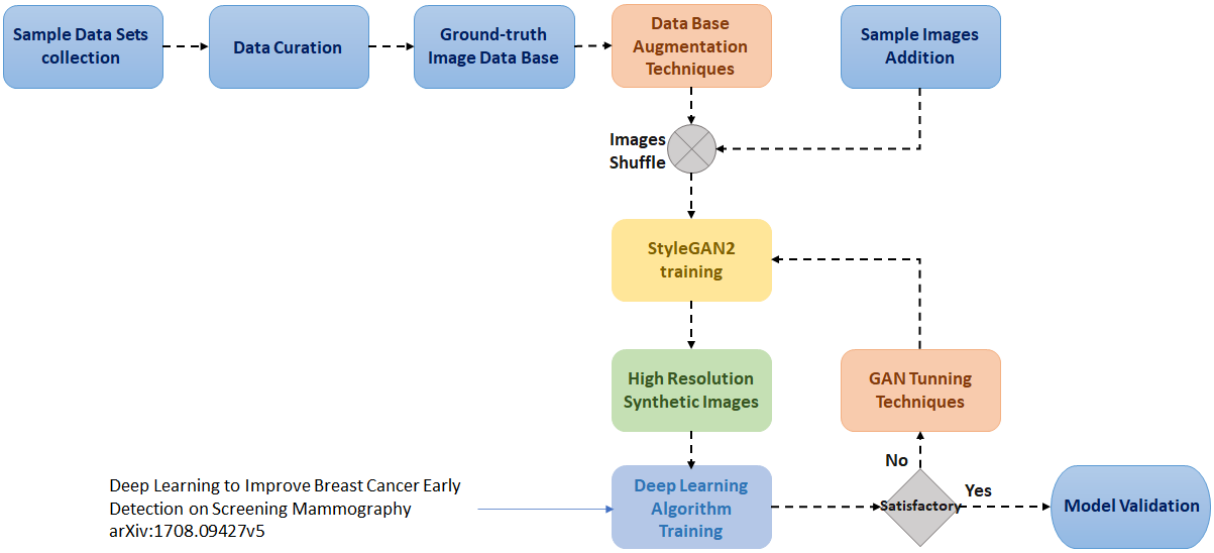


Figura 13. Esquema de la metodología seguida. (Imagen propia)

4 Identificación de requisitos

En este capítulo vamos a describir todo el trabajo previo realizado al desarrollo en sí de la herramienta software. Esta etapa del desarrollo suele ser crítica, ya que una buena recolección y tratamiento de los datos a usar suele producir mejores resultados en los entrenamientos de modelos con redes neuronales.

4.1 Equipo y material

Como hemos comentado anteriormente, una condición de la viabilidad del proyecto es su escasa necesidad de inversión.

Estos últimos años, debido al auge de las criptomonedas y las mafias de minería, conseguir una tarjeta gráfica potente es un asunto complicado, por no decir, que conseguir varias unidades de GPU para un proyecto como este es casi imposible. Aquí entran en juego los entornos de ejecución en remoto que ofrecen plataformas como las de Amazon, Azure de Microsoft o Google Colab. Estos entornos en remoto son una ventaja decisiva, ya que no precisamos de un equipo potente personal, podemos conectarnos a nuestro entorno desde cualquier dispositivo: tablet, pc, portátil, móvil… Por lo tanto, solo requerimos de uno de estos dispositivos para llevar a cabo el proyecto.

De entre todos los servicios de ejecución en remoto, elegimos Google Colab, debido a su precio competitivo. Por ejemplo, Amazon Web Services (AWS), ofrece su instancia más básica a partir de 50 centavos estadounidenses por hora. Teniendo en cuenta que un entrenamiento como el que nos propusimos nos ha llevado meses de entrenamientos con diferentes configuraciones, investigaciones de ensayo/error, etc. Dicho precio se saldría del objetivo de crear una herramienta barata. Google Colab ofrece su cuenta PRO por 9,99$ / mes. A pesar de que las especificaciones de su entorno en remoto son un poco escasas (una sola GPU, posibles desconexiones del entorno a partir de las 24h de uso continuado…), creemos que su uso para este proyecto ha sido suficiente y adecuado.

En resumen, lo único que necesitamos fue un dispositivo para conectarnos a nuestro entorno de Google Colab, una conexión a internet estable para evitar desconexiones al mismo y suscribirnos al servicio PRO de Google Colab, unos 9,99$/mes, durante unos 5 meses.

4.2 Elección de la base de datos

Existen múltiples bases de datos de mamografías disponibles públicamente. En nuestro caso, elegimos la denominada DDSM, unos 47 GB de imágenes repartidas en 3 directorios: Benign, Cancer y Normal, junto con una hoja de cálculo Excel con datos médicos etiquetados, no sensitivos, de cada una de las pacientes que proporcionaron las imágenes.

Son varias las razones que nos llevaron a elegir dicha base de datos, entre ellas:

- Poseen un número aceptable de muestras, 6543 en total. Era de las bases de datos que más se acercaba al número mínimo de imágenes para este tipo de entrenamientos de red neuronal profunda, unas 10000.
- Las imágenes estaban disponibles tanto en formato PNG de 16 bit como en JPEG, a diferencia de otras bases de datos públicas que disponen sus imágenes en formato DICOM. Este formato requiere un preprocesamiento para poder ser visualizado e inspeccionar su calidad, ya sea en PNG o JPEG.
- En la distribución de esta base de datos, el porcentaje de imágenes con cáncer era mayor que las que no presentaban tumores, algo inusual como ya hemos comentado y una de las motivaciones para realizar este proyecto.

4.3 Preprocesamiento de las imágenes

4.3.1 Reescalado y encuadrado de las imágenes

Dado que StyleGAN 2 ADA solo acepta imágenes cuadradas, fue preciso aplicar ciertos cambios a las imágenes originales de la base de datos, que son rectangulares. Este proceso presenta ciertas dificultades, ya que no podíamos aplicarles una transformación directamente para forzar que fueran cuadradas, perdiendo su proporción original. Buscando herramientas en la web, encontramos un repositorio en GitHub(Schultz, n.d.) que ofrecía un archivo en python, implementable en nuestro notebook, capaz de realizar diversas transformaciones a las imágenes. Redujimos la resolución a 512x512, añadiendo dos rectángulos negros a la imagen para que mantuviera su proporción original. En la siguiente figura, podemos comprobar el antes y después de una foto original con su homóloga preprocesada:

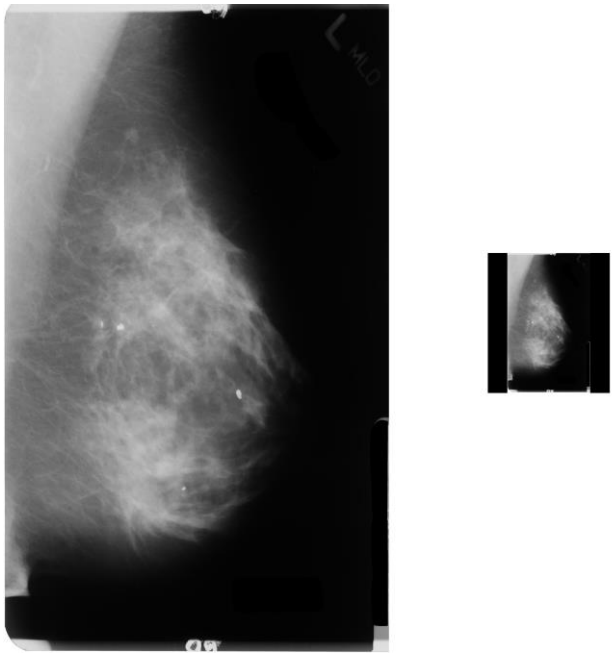


Figura 14. Resultado del preprocesamiento en una imágen del dataset original DDSM. (Imagen propia)

5 Descripción de la herramienta software desarrollada

5.1 Paso previo: configurar Google Colab Notebook

Debido a la elección de nuestro entorno de ejecución en remoto que hemos detallado en el capítulo 4, el desarrollo de nuestra herramienta fue llevado a cabo en un notebook de Google Colab. Estos notebooks son entornos de programación interactivos compuestos de celdas ejecutables:

Figura 15: Celdas ejecutables de un notebook, en el entorno de Google PRO. (Imagen propia)

Lo primero que debe hacerse es cambiar el entorno de ejecución a uno con uso de GPU. Para ello, nos dirigiremos a “Entorno de ejecución” y pincharemos en “Cambiar tipo de entorno de ejecución”, como se ve en la figura 16:

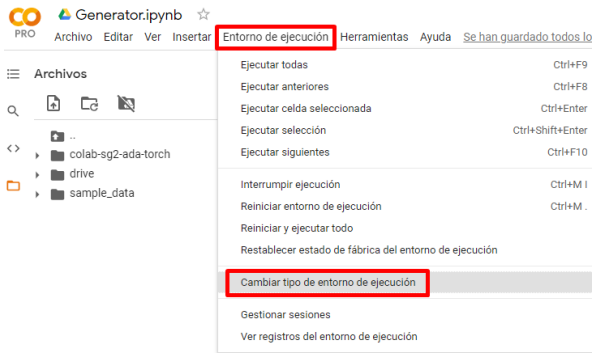


Figura 16: Cómo cambiar a un entorno de ejecución con GPU, parte 1. (Imagen propia)

Después, elegiremos GPU en el desplegable de “Aceleración por hardware”:

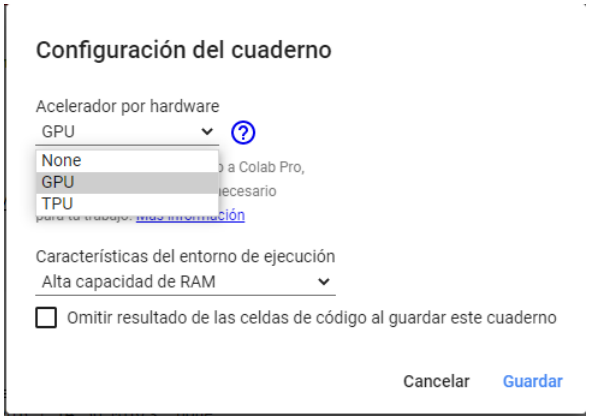


Figura 17: Cómo cambiar a un entorno de ejecución con GPU, parte 2. (Imagen propia)

Una vez completados estos pasos, ya tendremos configurado nuestro entorno de GPU adecuadamente para el uso de la herramienta.

5.2 Descripción y uso de la herramienta

La herramienta en sí es muy intuitiva, las celdas han de ejecutarse en un orden ya establecido e indicado en el propio notebook, para que incluso usuarios poco experimentados en Machine Learning puedan usarla. Se adjuntará el cuaderno en texto como anexo, para una mayor profundización.

Todas las celdas están debidamente comentadas tanto en castellano como en inglés dentro del notebook. Aun así, procedemos a detallar brevemente las partes del mismo.

La herramienta consta de **1 paso opcional** y, a continuación, de **3 pasos bien diferenciados**:

1. **(OPCIONAL) Activar Google Drive:** Google Colab cuenta con la posibilidad de conectarse a una cuenta de Google Drive, para así acceder a archivos personales y almacenados a dicha cuenta. De hecho, en el proceso de entrenamiento, usamos Google Drive para almacenar todas las imágenes originales, los resultados del entrenamiento, las imágenes generadas, etc. Es ALTAMENTE RECOMENDABLE usar esta opción, ya que, si se usa el espacio de almacenamiento de la sesión actual del notebook, el usuario corre el riesgo de perder todos los datos si se desconecta la sesión o cierra el notebook.
2. **Clonar y acceder a las carpetas del repositorio:** Clonar un repositorio significa, básicamente, copiar su contenido a otro directorio de nuestra propiedad. Por esta razón, indicamos que es recomendable conectar Google Drive al notebook, para clonar solo una vez el repositorio y mantener siempre actualizada la copia con los resultados de los entrenamientos, imágenes generadas, modelos entrenados, etc. Una vez tengamos el repositorio clonado en nuestro equipo o almacenamiento de Google Drive, procederemos a instalar las librerías de las que depende StyleGAN 2 ADA TORCH para funcionar correctamente, simplemente ejecutando la celda siguiente.
3. **Cargar los modelos ya entrenados:** La parte de generación de imágenes sintéticas de este proyecto ha producido 2 modelos: uno para generar imágenes con cáncer y otro para generar sin presencia de tumores. Estos dos modelos se guardan en archivos de extensión ".pkl", que se deberán cargar en la

herramienta, de modo que sea capaz de generar imágenes. De nuevo, si están cargados en el Google Drive del usuario, este paso no es necesario dado que, en el siguiente punto de generación de imágenes, solo tendrán que actualizar uno de los argumentos del comando de python con la ruta en la que se encuentra cada archivo .pkl.

4. **Generar las imágenes:** Finalmente, llegamos a la parte de generar el dataset. Se pueden observar dos celdas casi idénticas, diferenciándose solamente en la ruta que lleva a cada modelo, uno para cáncer y otro para normales. En estas celdas se ejecuta un comando python que llama al archivo “generate.py” de StyleGAN 2 ADA, comando que recibe 4 argumentos muy importantes. Aunque estos argumentos se hallan perfectamente explicados en la herramienta, creemos conveniente repetir la explicación aquí puesto que es un paso crucial:
    a. **carpeta_salida**: La carpeta donde queremos que se guarden las imágenes. De nuevo, si se quiere usar una carpeta de Google Drive, solo se deberá pegar la ruta a dicho directorio en este argumento.
    b. **trunc**: Si existen muestras con poco peso en el dataset de entrenamiento, puede que algunas de las imágenes generadas presenten artefactos como saturaciones excesivas, deformaciones en las siluetas, color, etc. Para evitar dichos artefactos, StyleGAN implementa este argumento, que "trunca" o fuerza el valor del vector latente introducido para que sea de un valor más cercano al de la media. En resumen, este valor solo ha de modificarse si se requiere más diversidad entre las imágenes a generar, pero el usuario ha de ser consciente de que pueden aparecer artefactos y errores en dichas imágenes. **Por defecto, el valor es de 1 (sin truncamiento) y se puede variar hasta llegar al 0 (máximo truncamiento)**.
    c. **seeds**: Este argumento se usa para especificar el número de imágenes que queremos generar. Debe seguir el formato X-Y, siendo X la primera de dichas "semillas", que el generador convierte en imágenes e Y, la última. **Por defecto, está definido como 0-9999, es decir, generará 10000 imágenes.**
    d. **network**: Este argumento es la ruta al archivo .pkl que contiene el modelo para generar imágenes. Si el usuario ha activado su drive y tiene el archivo allí, solo tiene que copiar y pegar la ruta **SIN CORCHETES, por ejemplo: --network=/content/drive/MyDrive/models/cancer.pkl**. Si no tiene drive activado, asumimos que ha subido el modelo con las celdas anteriores, por lo que se encontrará en los archivos temporales de google, "/content/".
5. **Compresión y descarga del dataset:** Solo hay que introducir la ruta donde están almacenadas las dos carpetas con imágenes generadas de cáncer y normales para comprimirlas en un .zip, llamado “dataset.zip” y que se descargue automáticamente cuando acabe de crearlo.

5.3 Resultado

El resultado de las imágenes generadas se guardará dentro del comprimido, cada tipo de imágenes en la carpeta de salida indicada, en formato PNG, y con el número de semilla que la produjo:

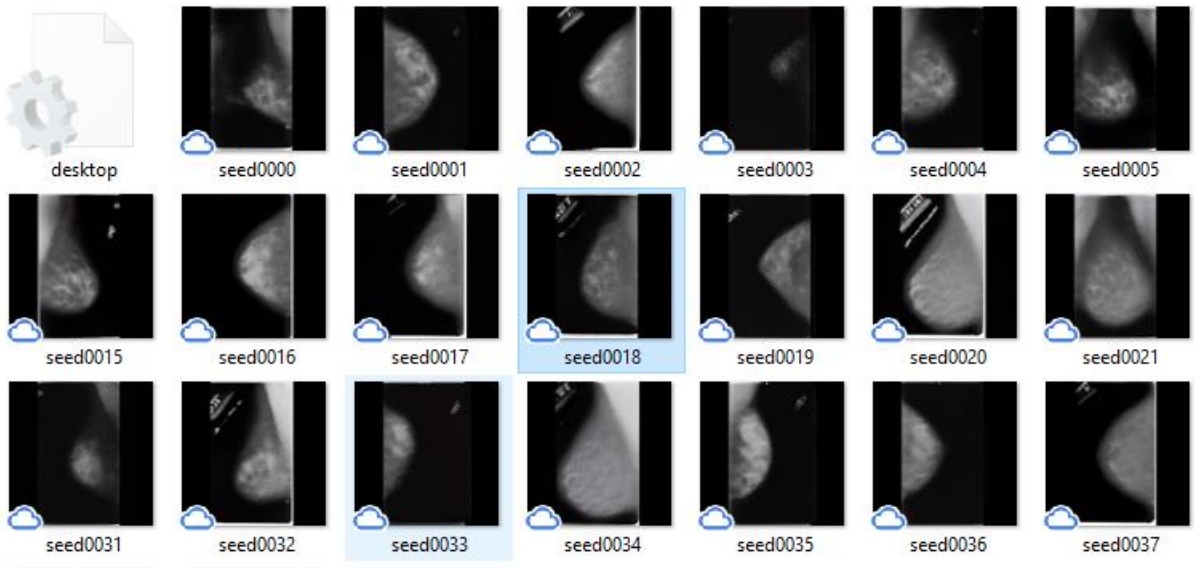


Figura 18: Muestra del resultado de las imágenes generadas. (Imagen propia)

La calidad de las imágenes generadas será evaluada en el siguiente capítulo, junto con los resultados de la clasificación llevada a cabo.

Como hemos demostrado, es una herramienta muy intuitiva, sencilla y directa de usar. Simplemente hay que ejecutar paso por paso las celdas del archivo notebook en un entorno que lo soporte (Google Colab, DeepNote de Microsoft, Jupyter Notebook) y descargar el comprimido con el dataset resultante.

6 Evaluación

En este capítulo vamos a detallar y analizar los resultados producidos, con la intención de comprobar la utilidad de la herramienta desarrollada. La metodología ya se ha explicado en el capítulo 4, pero a modo de resumen, los pasos que hemos realizado para clasificar mamografías y validar los resultados de los datasets generados son los siguientes:

- Generar datasets de mamografías sintéticas siguiendo diferentes configuraciones con nuestra herramienta.
- Implementar un clasificador del estado del arte.
- Aplicar técnicas de Transfer Learning para reentrenar el clasificador con nuestras imágenes sintéticas generadas, siguiendo diferentes configuraciones respecto al porcentaje de imágenes con presencia de cáncer y sin él.
- Ejecutar las diferentes versiones del clasificador frente a un mismo subconjunto de imágenes originales reales del dataset DDSM y comprobar los resultados de métricas usuales en tareas de clasificación de imágenes.

El primer paso en la evaluación de resultados debe ser, sin duda, evaluar la calidad de las imágenes generadas.

6.1 Evaluación de las imágenes generadas

Para este proyecto generamos 3 datasets con 24.000 imágenes, todas sintéticas. El porcentaje de imágenes con cáncer y sin él sigue 3 distribuciones diferentes:

1. 30% imágenes con cáncer y 70 % sin él:

30-70			
24000			
TRAIN		**TEST**	
70%		30%	
16800		7200	
CANCER	**NORMAL**	**CANCER**	**NORMAL**
30%	70%	30%	70%
5040	11760	2160	5040

Tabla 1: Distribución dataset sintético de entrenamiento 30% cáncer – 70% normal.

2. 50% imágenes con cáncer y 50% sin él:

50-50			
24000			
TRAIN		**TEST**	
70%		30%	
16800		7200	
CANCER	**NORMAL**	**CANCER**	**NORMAL**
50%	50%	30%	70%
8400	8400	2160	5040

Tabla 2: Distribución dataset sintético de entrenamiento 50% cáncer – 50% normal

3. 70% imágenes con cáncer y 30% sin él:

70-30			
24000			
TRAIN		**TEST**	
70%		30%	
16800		7200	
CANCER	**NORMAL**	**CANCER**	**NORMAL**
70%	30%	30%	70%
11760	5040	2160	5040

Tabla 3: Distribución dataset sintético de entrenamiento 70% cáncer – 30% normal

El objetivo de generar estos datasets solo con imágenes sintéticas era comprobar si se alcanzaban resultados aceptables en el entrenamiento y posterior clasificación. En caso contrario, como se mencionó en el capítulo de metodología, mezclaremos las imágenes generadas con imágenes reales del dataset original en diferentes proporciones, para que el dataset sea más variado y evitar aún más el sobreajuste. Dichas proporciones serían las siguientes:

1. 30% imágenes con cáncer y 70 % sin él, de las cuales, 95% son sintéticas y 5% reales:

MIXED 30% CANCER - 70% NORMAL (95% SYNT - 5% REAL)							
24000							
TRAIN				**TEST**			
70%				30%			
16800				7200			
CANCER		**NORMAL**		**CANCER**		**NORMAL**	
30%		70%		30%		70%	
5040		11760		2160		5040	
SYNTHETHIC	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**
95%	5%	95%	5%	95%	5%	95%	5%
4788	252	11172	588	2052	108	4788	252

Tabla 4: Distribución dataset de entrenamiento 30% cáncer – 70% normal, con 95% de imágenes sintéticas y 5% imágenes reales.

2. 50% imágenes con cáncer y 50 % sin él, de las cuales, 95% son sintéticas y 5% reales:

MIXED 50% CANCER - 50% NORMAL (95% SYNT - 5% REAL)							
24000							
TRAIN				**TEST**			
70%				30%			
16800				7200			
CANCER		**NORMAL**		**CANCER**		**NORMAL**	
50%		50%		30%		70%	
8400		8400		2160		5040	
SYNTHETHIC	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**
95%	5%	95%	5%	95%	5%	95%	5%
7980	420	7980	420	2052	108	4788	252

Tabla 5: Distribución dataset de entrenamiento 50% cáncer – 50% normal, con 95% de imágenes sintéticas y 5% imágenes reales.

3. 70% imágenes con cáncer y 30 % sin él, de las cuales, 95% son sintéticas y 5% reales:

MIXED 70% CANCER - 30% NORMAL (95% SYNT - 5% REAL)							
24000							
TRAIN				**TEST**			
70%				30%			
16800				7200			
CANCER		**NORMAL**		**CANCER**		**NORMAL**	
70%		30%		30%		70%	
11760		5040		2160		5040	
SYNTHETHIC	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**
95%	5%	95%	5%	95%	5%	95%	5%
11172	588	4788	252	2052	108	4788	252

Tabla 6: Distribución dataset de entrenamiento 70% cáncer – 30% normal, con 95% de imágenes sintéticas y 5% imágenes reales.

4. 30% imágenes con cáncer y 70 % sin él, de las cuales, 90% son sintéticas y 10% reales:

MIXED 50% CANCER - 50% NORMAL (90% SYNT - 10% REAL)							
24000							
TRAIN				**TEST**			
70%				30%			
16800				7200			
CANCER		**NORMAL**		**CANCER**		**NORMAL**	
50%		50%		30%		70%	
8400		8400		2160		5040	
SYNTHETHIC	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**
90%	10%	90%	10%	90%	10%	90%	10%
7560	840	7560	840	1944	216	4536	504

Tabla 7: Distribución dataset de entrenamiento 30% cáncer – 70% normal, con 90% de imágenes sintéticas y 10% imágenes reales.

5. 50% imágenes con cáncer y 50 % sin él, de las cuales, 90% son sintéticas y 10% reales:

MIXED 50% CANCER - 50% NORMAL (90% SYNT - 10% REAL)							
24000							
TRAIN				**TEST**			
70%				30%			
16800				7200			
CANCER		**NORMAL**		**CANCER**		**NORMAL**	
50%		50%		30%		70%	
8400		8400		2160		5040	
SYNTHETHIC	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**
90%	10%	90%	10%	90%	10%	90%	10%
7560	840	7560	840	1944	216	4536	504

Tabla 8: Distribución dataset de entrenamiento 50% cáncer – 50% normal, con 90% de imágenes sintéticas y 10% imágenes reales.

6. 70% imágenes con cáncer y 30 % sin él, de las cuales, 90% son sintéticas y 10% reales:

MIXED 70% CANCER - 30% NORMAL (90% SYNT - 10% REAL)							
24000							
TRAIN				**TEST**			
70%				30%			
16800				7200			
CANCER		**NORMAL**		**CANCER**		**NORMAL**	
70%		30%		30%		70%	
11760		5040		2160		5040	
SYNTHETHIC	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**	**SYNTHETHIC**	**REAL**
90%	10%	90%	10%	90%	10%	90%	10%
10584	1176	4536	504	1944	216	4536	504

Tabla 9: Distribución dataset de entrenamiento 70% cáncer – 30% normal, con 90% de imágenes sintéticas y 10% imágenes reales.

Una vez descritas todas las configuraciones de datasets que llevaremos a cabo, es conveniente detallar qué es por qué y cómo se usa la puntuación FID, una métrica para analizar la calidad de imágenes sintéticas generadas por redes GAN.

6.1.1 Una métrica para evaluar imágenes generadas por una GAN: Frechet Inception Distance (FID)

La calidad de las imágenes sintéticas que hemos generado es primordial, dado que el clasificador se entrenará con ellas y los resultados de dicho entrenamiento dependen en gran medida de la calidad de los datos con los que se entrena. Por ello, como mencionamos en el capítulo 3, usaremos la métrica FID para la evaluación de las mamografías sintéticas. Para explicar con detalle qué es realmente y por qué se usa la puntuación FID hay que detenerse en qué propiedades queremos usar para medir si una imagen sintética es "buena". Nos basaremos en dos propiedades:

1. Fidelidad: Queremos que el modelo genere imágenes de una calidad similar a las originales.
2. Diversidad: Queremos que el modelo genere imágenes que NO están presentes en el dataset, es decir, que genere imágenes nuevas.

Por lo tanto, queremos usar una métrica que evalúe ambas propiedades, pero ¿en qué debe basarse nuestra métrica para comparar imágenes basándose en su fidelidad y diversidad? A priori, son dos términos muy abstractos y difíciles de concretar matemáticamente. Dos de las aproximaciones más usuales en el campo de la visión computacional son:

1. La distancia de píxeles: Es una métrica anticuada y poco eficiente, dado que simplemente sustrae los valores de píxeles entre dos imágenes y el resultado es la "distancia de píxel".

2. La distancia de características: Para esta métrica, se usa un modelo preentrenado de clasificación de imágenes y se usa como una capa intermedia. El vector resultante es una representación a alto nivel de la imagen con el que se puede calcular una métrica de distancia más fiable y estable.

Como decíamos, se usa el modelo **Inception V3 model** (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015), preentrenado con un dataset masivo, el renombrado **Imagenet** (Deng, y otros, 2009). En lugar de comparar píxel a píxel las imágenes, lo que la FID compara es la media y la desviación estándar de una de las capas profundas del modelo Inception V3. El modelo, como decíamos anteriormente, recibe dos distribuciones Gaussianas multidimensionales distintas: la de las características de las imágenes sintéticas generadas por el Generador y la de las características de las imágenes reales usadas para el entrenamiento de la DCGAN. El valor de la FID, cuanto más cercano a 0, mayor similitud entre las imágenes sintéticas y originales, lo que indica mayor calidad de las sintéticas:

$$FID = |\mu - \mu_w|^2 + tr\,(\Sigma + \Sigma_w - 2(\Sigma\Sigma_w)^{1/2}$$

Ecuación 2: Fórmula matemática de la puntuación FID.

Donde μ y $\mu_w$son las medias de las características de imágenes reales y generadas. Tr es la traza, es decir, el resultado de la suma de elementos a lo largo de la diagonal principal de las matrices de covarianza (Σ y Σ_w, matrices de covarianza para los vectores de las imágenes generadas y reales).

6.1.2 Evolución de la FID vs Tiempo de entrenamiento de cada modelo generador

Una vez que comprendemos cómo funciona y lo que compara la FID, podemos visualizar la evolución de dicha métrica frente al tiempo de entrenamiento de cada modelo generador de nuestra herramienta: el generador de mamografías con cáncer y el generador de normales:

6.1.2.1 FID vs Tiempo de entrenamiento para el generador de mamografías con cáncer

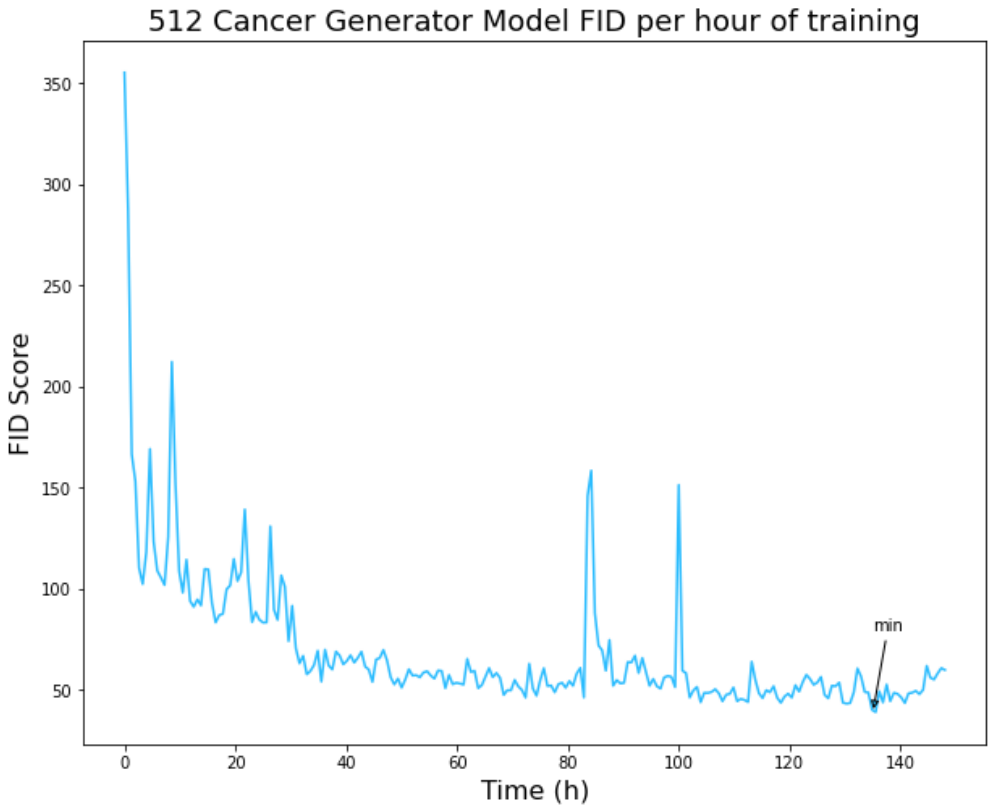


Figura 19: Gráfica de la evolución de la FID vs Tiempo de entrenamiento del generador de cáncer. (Imagen propia)

Lo primero que llama la atención de la Figura 19 son las anomalías o “picos” que aparecen en la función representada de la FID vs tiempo. Estas anomalías se deben a que Google Colab PRO, al ser un servicio tan económico, debe restringir de alguna manera las sesiones de conexión a sus entornos remotos. Dichas sesiones pueden ser, como mucho, de 24h consecutivas, pero después de ese límite desconectan el entorno y detienen la ejecución de cualquier programa que se estuviera ejecutando en el notebook. Un modelo generador como el que proponemos necesita de días para entrenarse, pero por suerte, los investigadores de StyleGAN ya contaban con este inconveniente, y desarrollaron su herramienta para que se pudiera “guardar” el estado del modelo en cada época en un fichero .pkl, y retomar el entrenamiento más adelante a partir de los valores guardados en el fichero. Sin embargo, en la primera época de cada vuelta al entrenamiento después de una ejecución, se producen esos valores tan anómalos de la FID, aunque en la siguiente época vuelva a obtener un valor muy aproximado al que tenía el modelo cuando se paró la ejecución anterior.

Obviando dichas anomalías, el valor mínimo alcanzado es de 49. Teniendo en cuenta que los investigadores de nVIDIA, en un entrenamiento de 21 días, usando una GPU mucho más potente y acceso a 25.000 imágenes para entrenar consiguieron un FID de alrededor de 10, nuestro resultado es lo suficientemente bueno para generar imágenes sintéticas con el modelo.

6.1.2.2 FID vs Tiempo de entrenamiento para el generador de mamografías normales

En cuanto al modelo generador de mamografías sin cáncer, el número de imágenes en el dataset de entrenamiento era menor, 2409 frente a 4049 de cáncer. Era de esperar, por tanto, que a menor cantidad de imágenes para entrenar, peor calidad de las imágenes sintéticas. Atendiendo a la figura 20, en efecto, fue así:

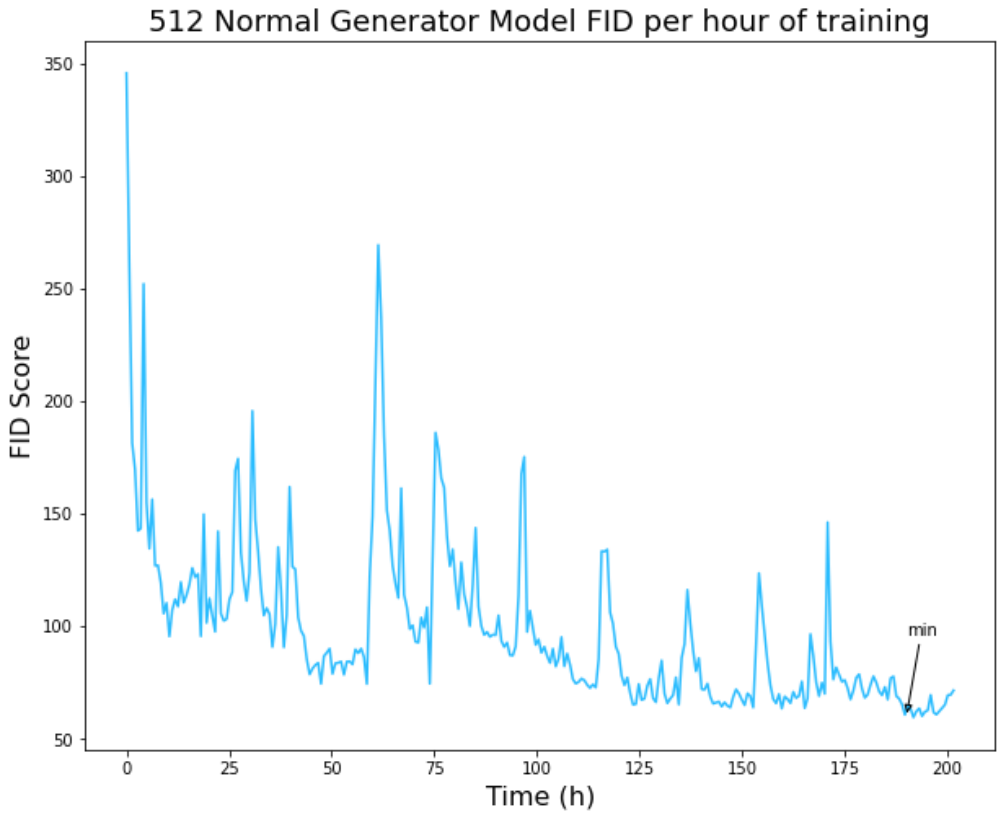


Figura 20: Gráfica de la evolución de la FID vs Tiempo de entrenamiento del generador normal. (Imagen propia)

El valor mínimo de FID es de 60 aproximadamente, y se tardó casi 20 horas más en alcanzarlo. Cabe destacar que ambos entrenamientos se pararon cuando el mecanismo de callback determinó que la mejora de la métrica FID había convergido a su máximo. Este mecanismo alarga el entrenamiento durante un número específico de épocas de margen y comprueba si ha mejorado la métrica pertinente respecto a un mínimo. Si ha mejorado, permite continuar el entrenamiento y si no, lo para.

Por lo tanto, una vez alcanzados los mínimos valores de la puntuación FID que permitían las condiciones de entrenamiento (número de imágenes originales, potencia de la GPU, capacidad de la memoria, etc.), decidimos empezar a generar los datasets sintéticos siguiendo las diferentes configuraciones para entrenar con ellos el clasificador VGG16.

6.2 Modelo clasificador VGG16

El modelo clasificador VGG16 ha sido reconocido como uno de los mejores clasificadores de imágenes por la comunidad de investigación en Aprendizaje Automático y Profundo. Tanto es así, que Keras, la famosa API de alto nivel de Google, ha incluido el modelo directamente en su librería, facilitando su implementación. La arquitectura del modelo se adjunta en el capítulo 9: Anexos.

El modelo se compone de 5 bloques compuestos por 2 capas de convolución seguidas de una capa de MaxPooling, esta es una configuración ya estandarizada para modelos de clasificación de imágenes. Estos bloques analizan pormenorizadamente las imágenes en sus componentes más básicos (formas, líneas, colores, brillo, etc).

Ahora bien, reentrenar este tipo de modelos tan grandes y complejos es una tarea titánica, al alcance casi exclusivamente de sus creadores, empresas como Google, Facebook, Amazon, que cuentan con una gran potencia de cálculo en clusters de GPU's y almacenamiento casi ilimitado. Este problema se resuelve aplicando las llamadas técnicas de Transfer Learning.

6.3 Transfer Learning

Uno de los problemas más frecuentes a la hora de entrenar un modelo de machine learning es el coste, tanto temporal como computacional del propio proceso de entrenamiento. Dicho proceso, sin profundizar en detalles matemáticos y de cálculo numérico, consiste en actualizar iterativamente los parámetros de cada neurona, según el valor de una función de error particular para cada modelo, que también se calcula con cada iteración.

Para los modelos más sencillos, este proceso puede ser bastante liviano y terminar en unas pocas horas. Para modelos como el VGG16 que queremos usar, el proceso de entrenarlo desde cero es totalmente inviable en un entorno remoto como el que disponemos, con pérdidas de conexión, solo una GPU, poca memoria RAM… Para evitar este inconveniente, la clave está en entrenarlo, pero **no desde cero**. Aquí entran en juego las técnicas de Transfer Learning.

Las técnicas de Transfer Learning utilizan los modelos preentrenados que publican los investigadores y desarrolladores originales de dichos modelos. Es decir, los parámetros de las neuronas ya fueron calculados y optimizados en el entrenamiento original, seguramente en entornos con gran potencia de cálculo y datasets enormes como el que se usó para entrenar VGG16, el Imagenet. Una vez se ha descargado el fichero que contiene el modelo preentrenado, se puede cargar en un entorno de ejecución de Python y modificar parámetros como las capas de entrada, profundas y de salida, funciones de activación, función de pérdida, pesos de las neuronas de cada capa, etc.

Existen multitud de técnicas para aplicar Transfer Learning, pero en este proyecto solo nos detendremos a explicar dos de las técnicas más populares y que mejores resultados ofrecen actualmente: ***Feature Extraction*** y ***Fine Tuning***.

6.3.1 Feature Extraction

La técnica de Feature Extraction "congela" las capas convolucionales con sus pesos ya entrenados y "desconecta" o elimina directamente las capas de salida, las capas densas o *Fully Connected*. Entonces, pasa las imágenes del dataset con el que queramos entrenar el modelo por las capas de convolución y, a la salida, tendremos un paquete apilado o *stack*, un vector numérico con todas las características extraídas. Este vector se aplana, *flattening*, para poder ser procesado como la entrada de cualquier otro clasificador de imágenes o para implementar nuevas capas densas de salida configuradas según nuestras necesidades. Visualmente:

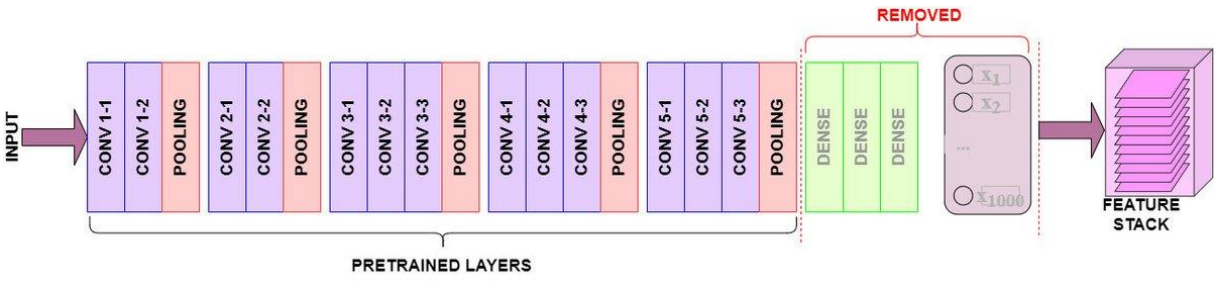


Figura 21: Ilustración de la técnica de Feature Extraction. (McDermott, s.f.)

6.3.2 Fine Tuning

La técnica de Fine Tuning, sin embargo, permite que un número *n* de capas preentrenadas se formateen y se reentrenen los pesos al pasar nuestro dataset por el modelo. Después, reemplazamos las capas densas de salida preentrenadas por nuestras propias capas densas de salida:

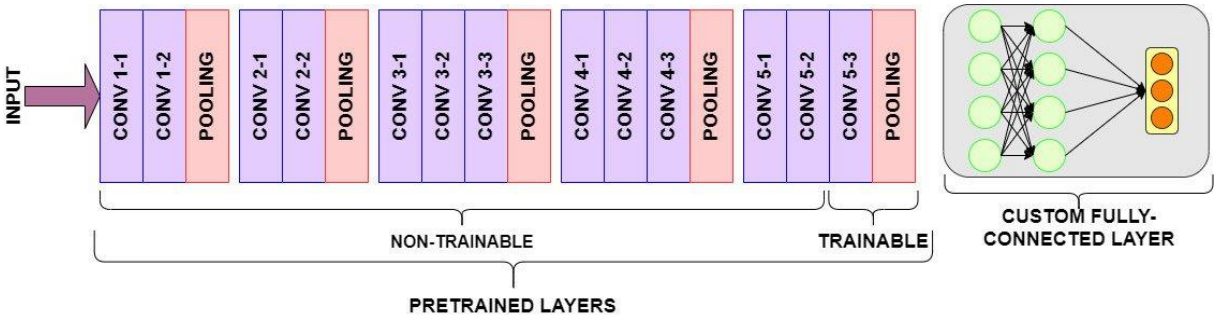


Figura 22: Ilustración de la técnica Fine Tuning. (McDermott, s.f.)

En nuestro caso, elegimos la técnica de Fine Tunning, al permitir más flexibilidad para entrenar las capas convolucionales del modelo.

Antes de analizar el resultado de los entrenamientos, sería conveniente detallar las cuatro métricas que hemos usado, las métricas habituales en tareas de clasificación de imágenes.

Métricas para la clasificación de imágenes

Todas estas métricas se basan en dos valores, los aciertos y los fallos de un clasificador. La forma más sencilla de entender este sistema es visualmente:

CONFUSION MATRIX			
True Class	**Positives**	TP (True Positive)	FN (False Negative)
	Negatives	FP (False Positive)	TN (True Negative)
		Positives	**Negatives**
		Predicted Class	

Tabla 10: Matriz de confusión.

La tabla 13 presenta el esquema que sigue una matriz de confusión. Está formada por dos clases: La *True Class*, que serían las etiquetas reales o verdaderas de las imágenes y la *Predicted Class*, que serían las etiquetas predichas por el clasificador para dichas imágenes. Dentro de cada clase existen dos categorías: *Positives* (Positivos) y *Negatives* (Negativos). Según esta clasificación, pueden ocurrir 4 escenarios:

1. TP (True Positive o Verdadero Positivo): El clasificador ha predicho que la imagen era positiva y ha acertado.
2. FP (False Positive o Falso Positivo): El clasificador ha predicho que era positivo, pero realmente era negativo.
3. FN (False Negative o Falso Negativo): El clasificador ha predicho que era negativo, pero realmente era positivo.
4. TN (True Negative o Verdadero Positivo): El clasificador ha predicho que era positivo y ha acertado.

De todos estos resultados, el más **crítico** a la hora de un diagnóstico médico es el número 3. FN (False Negative o Falso Negativo). Si nos ponemos en el caso de que una paciente padece un cáncer de mama y el clasificador ha clasificado su mamografía como “Normal” o “Negativa”, este resultado podría desencadenar en un deterioro crítico del estado de salud de la paciente a la hora de tratar el cáncer, desencadenando incluso la muerte de la misma. Es por ello por lo que deberemos fijarnos especialmente en esta métrica para analizar los resultados de las diferentes configuraciones.

Basándonos en esta clasificación de los resultados, se suelen usar 5 métricas para el análisis de un clasificador:

1. **Exactitud (Accuracy):** $\frac{(TP+TN)}{(TP+FP+FN+TN)}$ (Ecuación 2). Indica el número de elementos clasificados correctamente en comparación con el número total de elementos.
2. **Sensibilidad (Recall):** $\frac{(TP)}{(TP+FN)}$ (Ecuación 3). Muestra la cantidad de verdaderos positivos que el modelo ha clasificado en función del número total de valores positivos.
3. **Precision (Precision):** $\frac{(TP)}{(TP+FP)}$ (Ecuación 4). Representa el número de verdaderos positivos que son realmente positivos en comparación con el número total de valores positivos predichos.
4. **Puntuación F1 (F1 Score):** $2 * \frac{Precision * Recall}{Precision + Recall}$(Ecuación 5). Esta métrica es una combinación de las métricas de precisión y sensibilidad, además de servir de compromiso entre ellas. La mejor puntuación **F1 es igual a 1 y la peor a 0**. Aunque no la hemos usado en los resultados del entrenamiento de los clasificadores, la usaremos a la hora de analizar el resultado de las clasificaciones frente al dataset real.
5. **Pérdida (Loss):** En nuestro caso, hemos usado la función de pérdida *Binary Cross Entropy* o *Log Loss*. Es una función que primero analiza cuánto de lejos está el valor predicho respecto al valor verdadero y después penaliza según esa distancia. Estas funciones se usan durante el entrenamiento para realizar el update de los pesos de las capas y la técnica de Backpropagation. Su fórmula es la siguiente y, aunque analizarla matemáticamente se aleja del objetivo de esta memoria, en resumidas cuentas, Pi es la probabilidad de que una imagen sea de la clase 1 y 1-Pi, de la clase 0. Cuando la observación pertenece a la clase 1, el segundo miembro del sumando se hace 0 y viceversa:

$$Log\ Loss\ = \frac{1}{N}\Sigma_{i=1}^{N} - (y_i * log(p_i) + (1 - y_i) * log(1 - p_i))$$ (Ecuación 6).

Una vez detalladas las métricas que usaremos para el análisis de los resultados, empezaremos por el primer grupo de análisis: los resultados del modelo entrenado exclusivamente con datasets sintéticos generados.

6.4 Reentrenar VGG16 con datasets formados exclusivamente por imágenes sintéticas generadas.

El siguiente paso, una vez aplicado el Fine Tuning en el modelo original, reentrenamos el modelo siguiendo las diferentes configuraciones de dataset que detallamos en el apartado 6.1.

Como ground truth, escogimos un subconjunto de 3000 imágenes originales del dataset DDSM, siguiendo una distribución de 30% imágenes con cáncer y 70% libres de él, una distribución bastante usual en los datasets disponibles públicos.

6.6 Configuración 30% cáncer 70 % normal

Esta configuración es muy interesante desde el punto de vista comparativo, ya que los datasets reales de mamografías suelen seguir una proporción parecida: pocos casos de cáncer y muchos de imágenes normales. Por lo tanto, los resultados de la clasificación usando el modelo entrenado con esta configuración servirán de background para comparar los resultados posteriores.

6.6.1 Resultados del entrenamiento

Los resultados del entrenamiento son extremadamente buenos. Partiendo de un valor cercano al 65% de accuracy y acabando en un valor cercano al 98%. El resultado de los valores de las métricas / época se adjunta en el capítulo 9: Anexos.

A continuación, las figuras de cada métrica para un análisis más breve y visual, en las que se presentan dos series de datos: los de train (azul) y los de test (rosa).

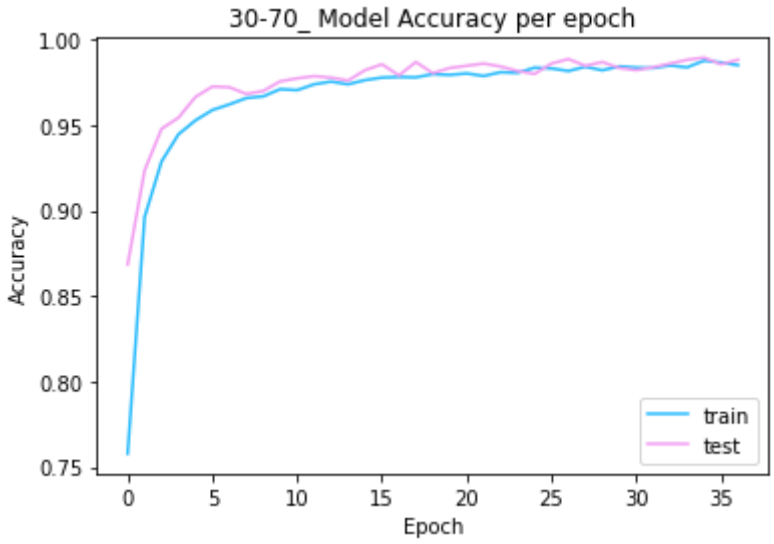


Figura 23: Exactitud del modelo entrenado con el dataset 30%-70%. (Imagen propia)

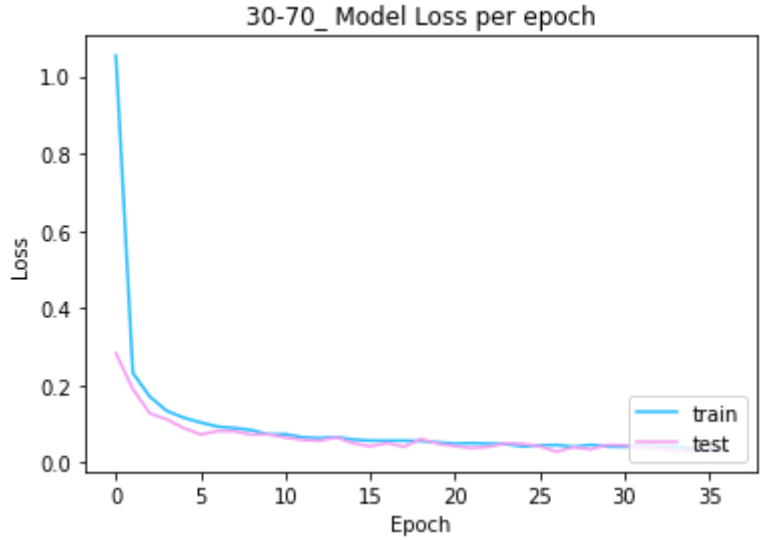


Figura 24: Pérdida del modelo entrenado con el dataset 30%-70%. (Imagen propia)

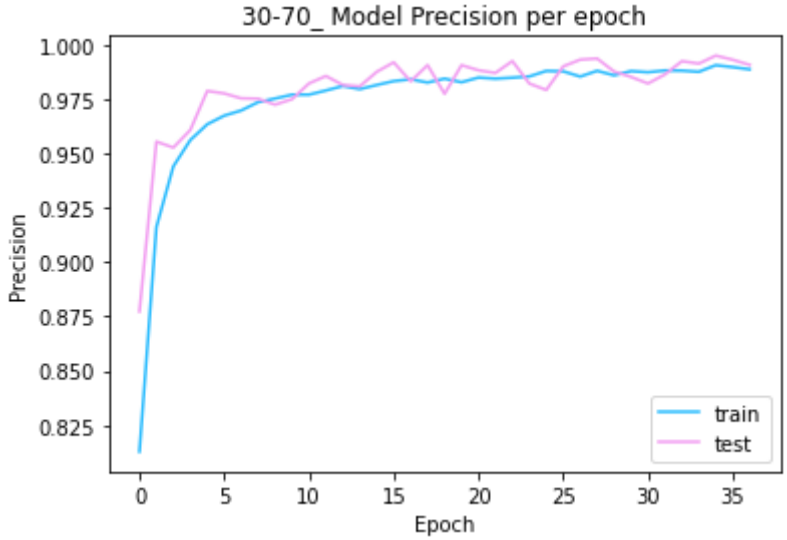


Figura 25: Precisión del modelo entrenado con el dataset 30%-70%. (Imagen propia)

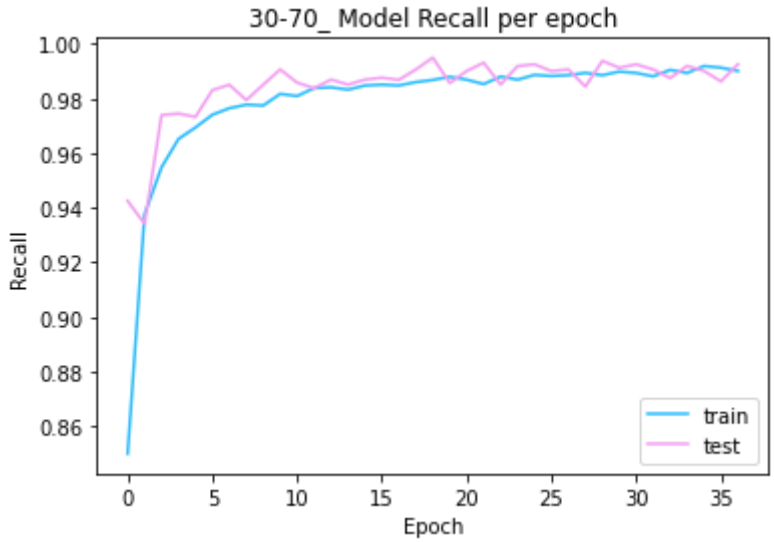


Figura 26: Sensibilidad del modelo entrenado con el dataset 30%-70%. (Imagen propia)

Lo primero que se aprecia del análisis del entrenamiento del modelo con esta configuración es su, aparente, brillante resultado. El valor inicial de casi todas las métricas ya es alto (alrededor del 65-70%) y alcanzan todas un valor muy cercano al 100%, característica que compartirán el resto de configuraciones respecto a los resultados del entrenamiento. Es posible que las mamografías, al ser un tipo de imagen con poca varianza entre muestras, junto con la gran cantidad de muestras que hemos generado (recordemos, 24.000), permiten unos resultados de entrenamiento tan buenos. Sin embargo, como veremos más adelante en el resultado de la clasificación, las configuraciones puramente sintéticas generan que el modelo caiga en sobreajuste (*overfitting*).

Otro aspecto que puede atraer nuestra atención es que la serie de datos de test presenta *mejores* resultados en casi todas las métricas respecto de la de entrenamiento. Si bien es cierto que este fenómeno resulta extraño para los iniciados en la creación de modelos de aprendizaje automático y profundo, es sabido que es bastante usual en el entrenamiento de modelos grandes. Este fenómeno se produce debido a las capas de regularización (L1, L2 y Dropout) del modelo, que evitan el sobreajuste u *overfitting* del mismo, “apagando” un porcentaje de neuronas durante el entrenamiento. De esta manera, se consigue que el modelo cree nuevos “caminos” o conexiones neuronales y no se “acostumbre” a usar las mismas conexiones que probó al principio. Sin embargo, estos “apagones” de neuronas no se producen durante la clasificación con el subconjunto de test, por lo que el modelo, al contar con todas las conexiones que ha entrenado, produce mejores resultados.

6.6.2 Resultados de la clasificación

En este apartado, veremos los resultados mediante una matriz de confusión, parecida a la que hemos detallado antes, pero cortesía de un repositorio en GitHub (Trimarchi, 2019).

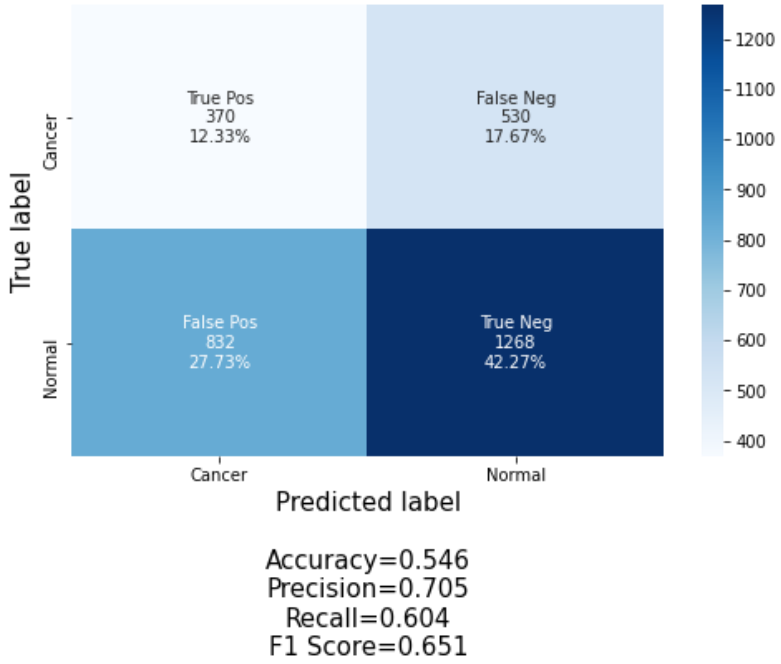


Figura 27: Matriz de confusión de la clasificación con el modelo 30% cáncer - 70% normal. (Imagen propia)

De la matriz de confusión podemos concluir que el modelo generado con la configuración 30%-70% cae en sobreajuste u overfitting, dado que las métricas obtenidas con la muestra del dataset real son mucho peores en comparación. En particular, el porcentaje de Falsos Negativos es preocupante, ya que llega a un 18%, casi una quinta parte del total de predicciones, cantidad inaceptable si el uso del clasificador es el diagnóstico de cáncer.

En cuanto a la exactitud, un 54,6% es un porcentaje muy bajo, casi como “echar una moneda al aire”. La precisión es buena, un 70%, aunque es un porcentaje engañoso porque recordemos que había un 40% más de imágenes Normales o “Negativas” que de Cáncer o “Positivas”. La sensibilidad es baja también, del 60,04%, igual que la Puntuación F1, cuyo valor óptimo recordemos que es el 1.00 y se encuentra en 0,65.

Este problema, como veremos a continuación, se resuelve mediante el uso de un dataset cuyas clases están equilibradas.

6.7 Configuración 50% cáncer 50% normal

6.7.1 Resultados del entrenamiento

Igual que la configuración anterior, los resultados del entrenamiento de este dataset más equilibrado son muy buenos. Procedemos a compararlos con los de la distribución base, la 30% cáncer - 70% normal

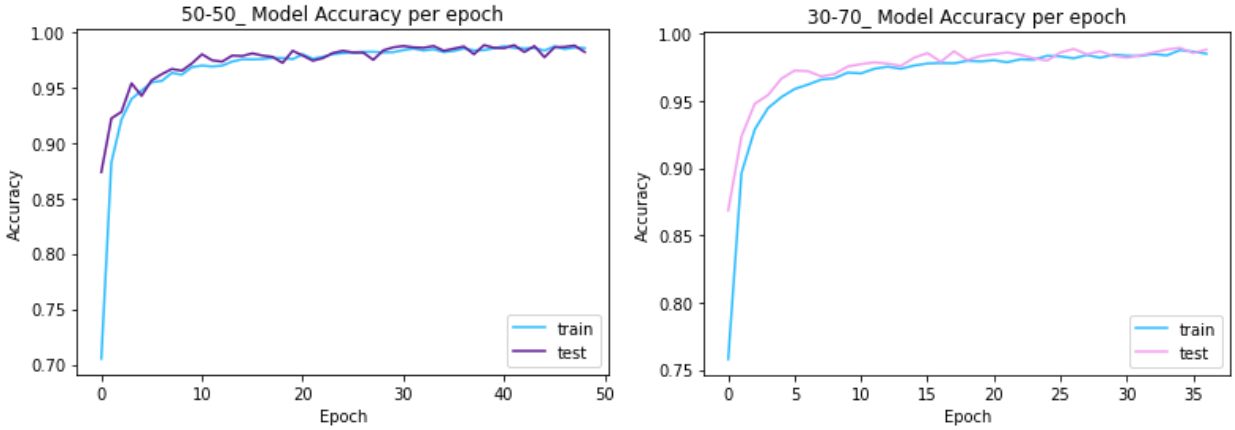


Figura 28: Exactitud del modelo entrenado con el dataset 50%-50% frente al de 30% - 70%. (Imagen propia)

Aunque los resultados se diferencian un poco, podemos observar que en el dataset desequilibrado, el de 30% cáncer y 70% normal, la diferencia entre la exactitud del subset de training y el de test es más grande, indicador de que existe mayor sobreajuste en este último. Este resultado era esperado, dado que el modelo se ha entrenado con muchas más imágenes normales que de cáncer, pudiendo clasificar fácilmente las normales y teniendo dificultades en las de cáncer.

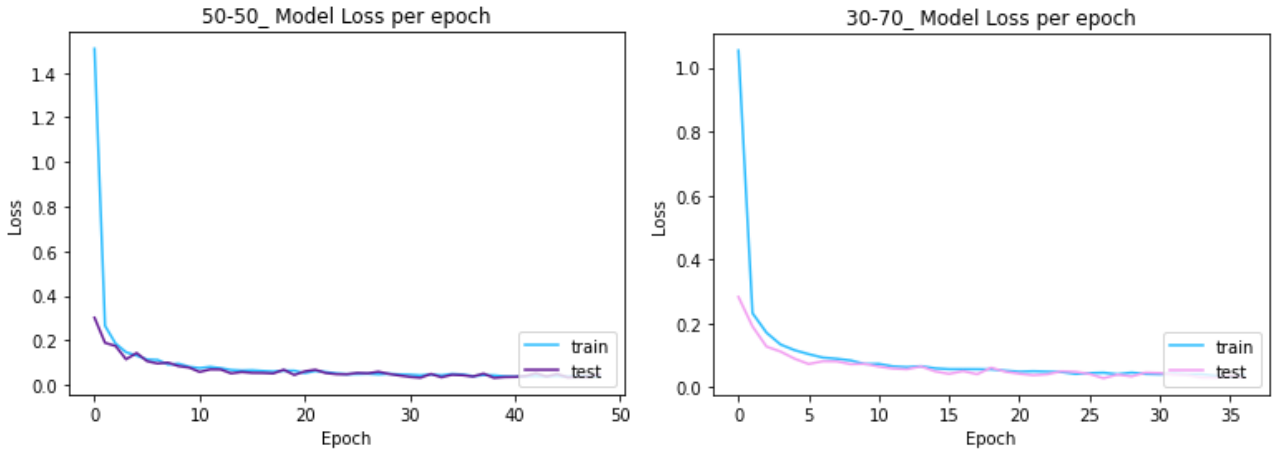


Figura 29: Pérdida del del modelo entrenado con el dataset 50%-50% frente al de 30% - 70%. (Imagen propia)

Podemos llegar a la misma conclusión: menor sobreajuste en el dataset equilibrado.

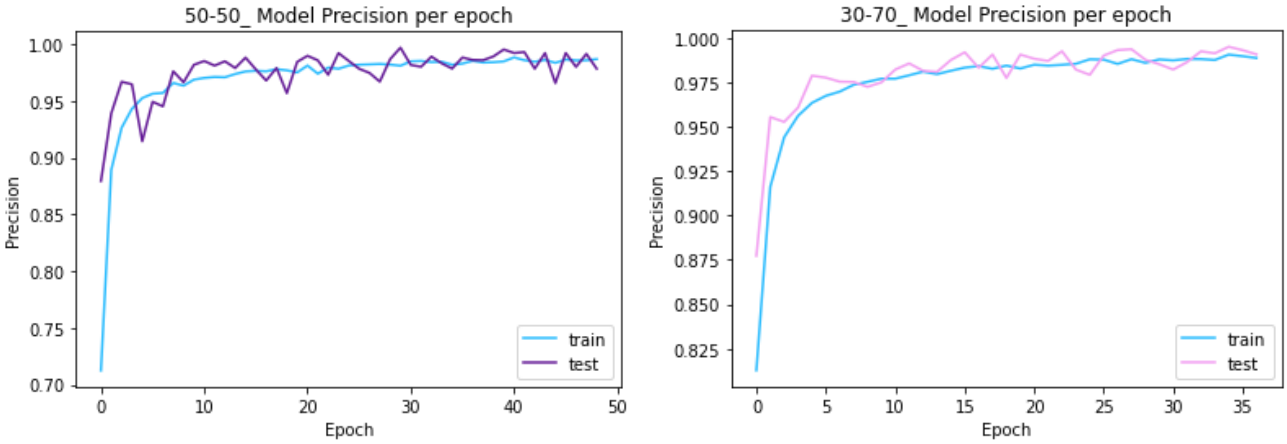


Figura 30: Precisión del del modelo entrenado con el dataset 50%-50% frente al de 30% - 70%. (Imagen propia)

Lo que podemos observar en cuanto a la precisión, es que en el modelo de 50%-50%, los resultados del subset de test varían más abruptamente que en el de 30% - 70%, aunque el resultado final es bueno (cerca del 96-98% en ambos). Esto podría ser debido a que en el modelo entrenado con el dataset equilibrado, al haber el mismo porcentaje de imágenes de ambos tipos, “duda” más de si una imágen presenta cáncer o no durante su entrenamiento.

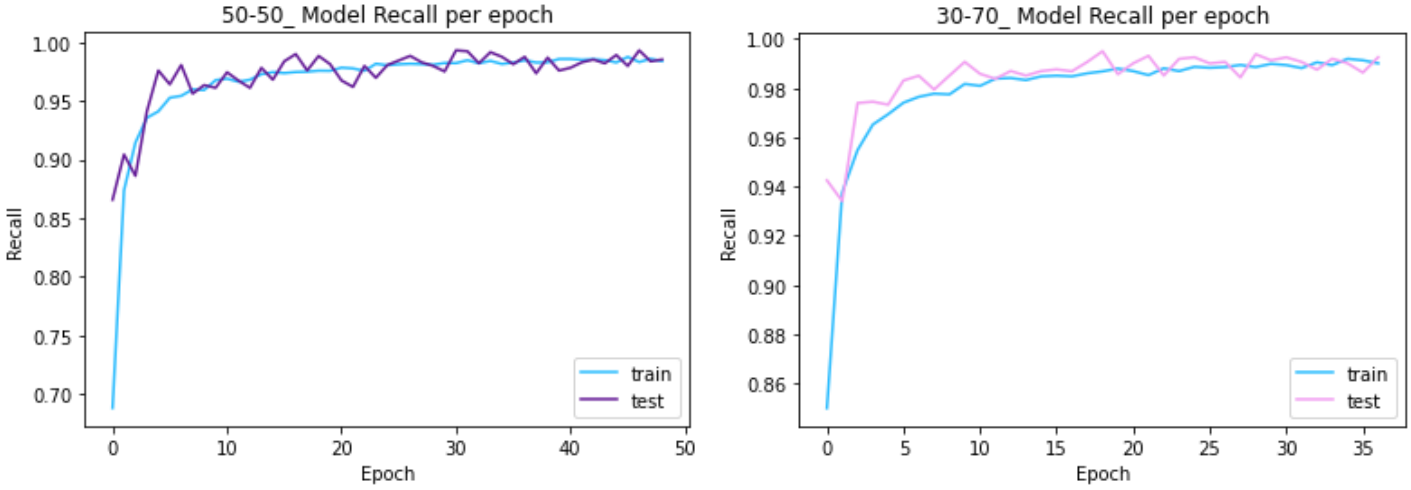


Figura 31: Sensibilidad del del modelo entrenado con el dataset 50%-50% frente al de 30% - 70%. (Imagen propia)

En cuanto a la sensibilidad, las gráficas son bastante similares entre sí a lo largo del entrenamiento y convergen a valores similares, cerca del 97-98%.

6.7.2 Resultados de la clasificación

Aunque en teoría, el haber entrenado el modelo con un dataset equilibrado debería hacer que el resultado mejorase, lo cierto es que empeora ligeramente:

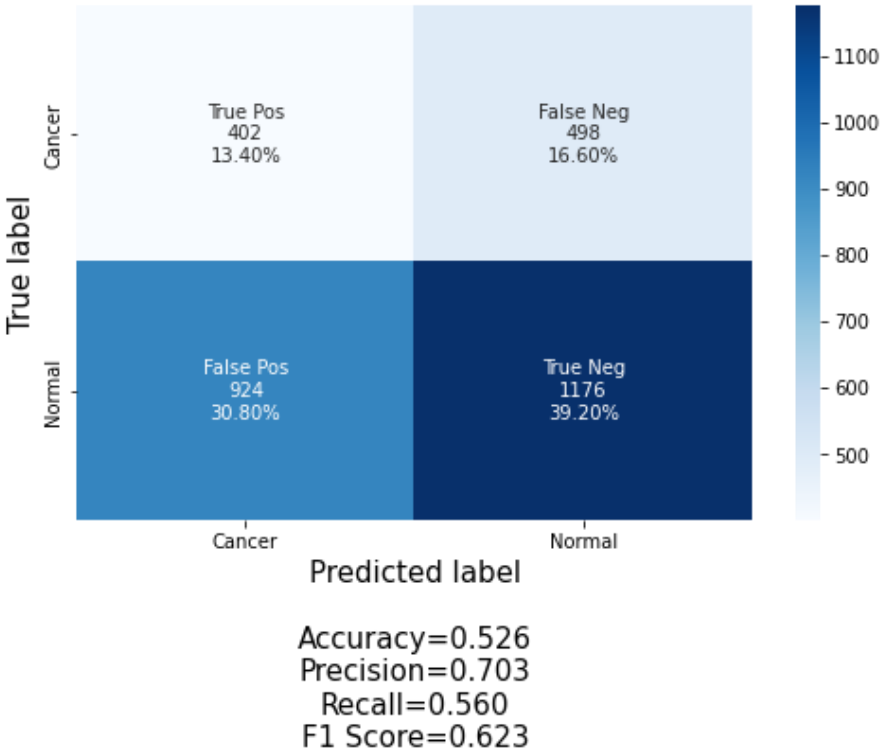


Figura 32: Matriz de confusión de la clasificación con el modelo 50%-50%. (Imagen propia)

Aunque el porcentaje de Falsos Negativos ha disminuido, algo positivo, el resto de las métricas han empeorado ligeramente. Es muy probable que el hecho de que solo se entrene con imágenes sintéticas propicie un sobreajuste demasiado potente como para que el clasificador ofrezca buenos resultados cuando se enfrenta a imágenes reales, con diferentes características.

6.8 Configuración 70% cáncer 30% normal

Finalmente, en cuanto a los resultados de entrenamiento, son muy similares a los otros dos casos anteriores. Las 4 métricas de entrenamiento convergen rápidamente hacia resultados muy positivos. La diferencia entre los resultados del subset de train y test es mínima, indicador de poco sobreajuste inicial en el entrenamiento del modelo. Procedemos a comparar los resultados entre los de 50% cáncer -50% normal y los de 30% cáncer-70% normal.

6.8.1 Resultados del entrenamiento

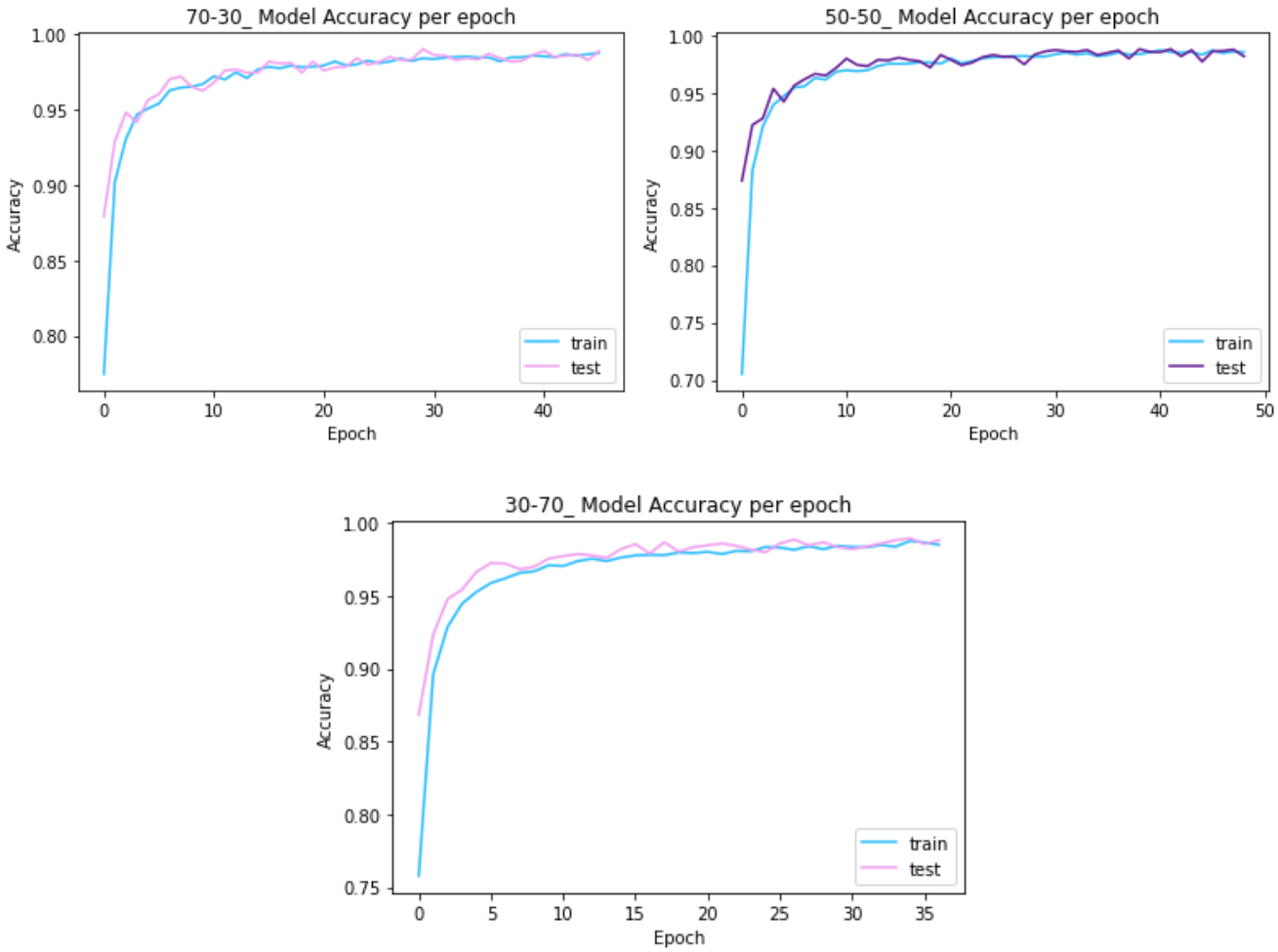


Figura 33: Exactitud del modelo entrenado con el dataset 70%-30% frente a los de 50%-50% y 30% - 70%. (Imagen propia)

Podemos observar que, durante las primeras épocas del entrenamiento, el modelo entrenado con el dataset 70% cáncer - 30% normal es parecido al de 30%-70% en cuanto a la diferencia entre test y train, pero después converge como el de 50%-50%, y se reduce esta distancia y, por lo tanto, aumenta el sobreajuste.

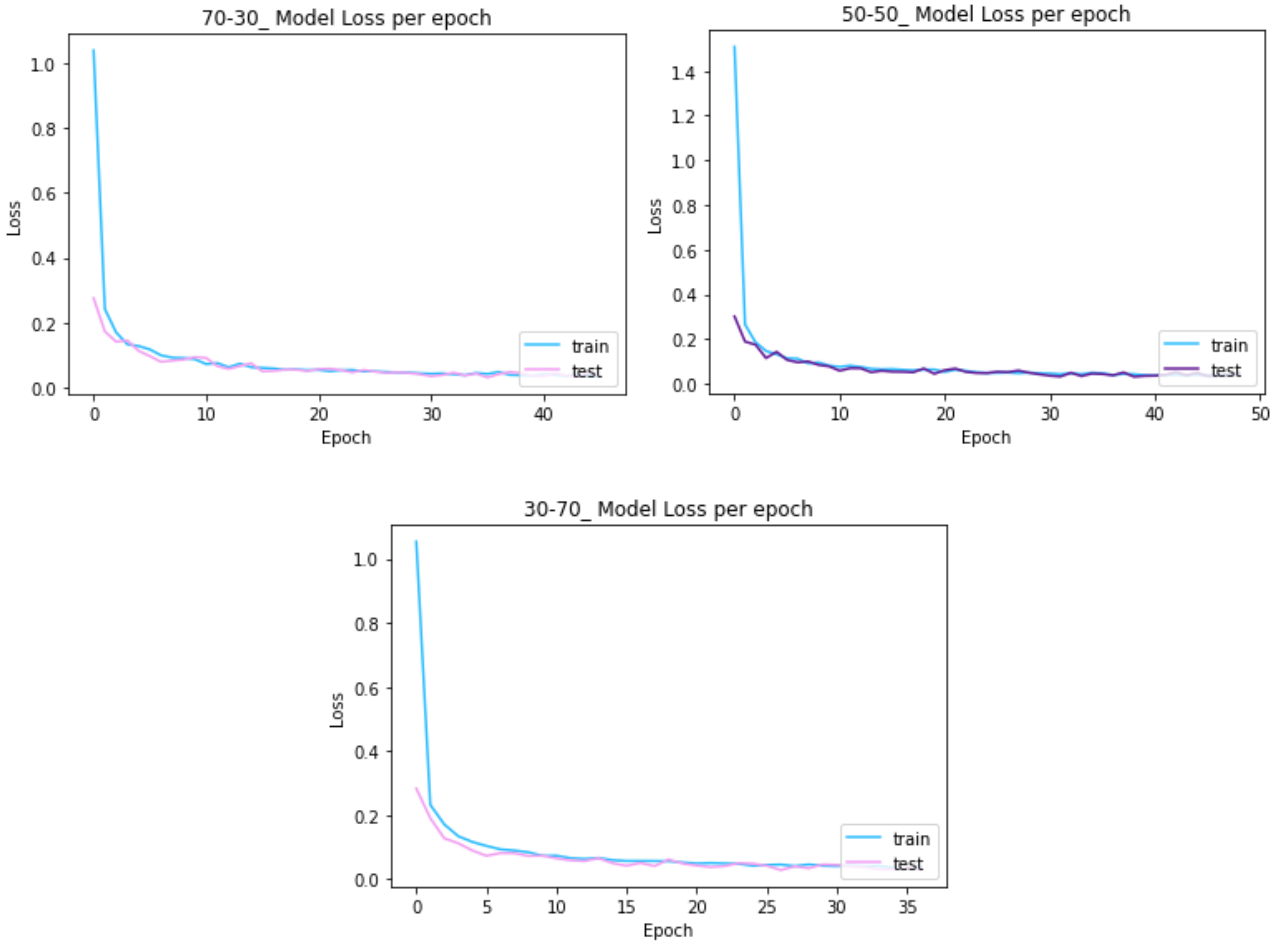


Figura 34: Pérdida del modelo entrenado con el dataset 70%-30% frente a los de 50%-50% y 30% - 70%. (Imagen propia)

Podríamos determinar que la evolución de la pérdida es buena y converge rápidamente durante los 3 entrenamientos, sin grandes diferencias.

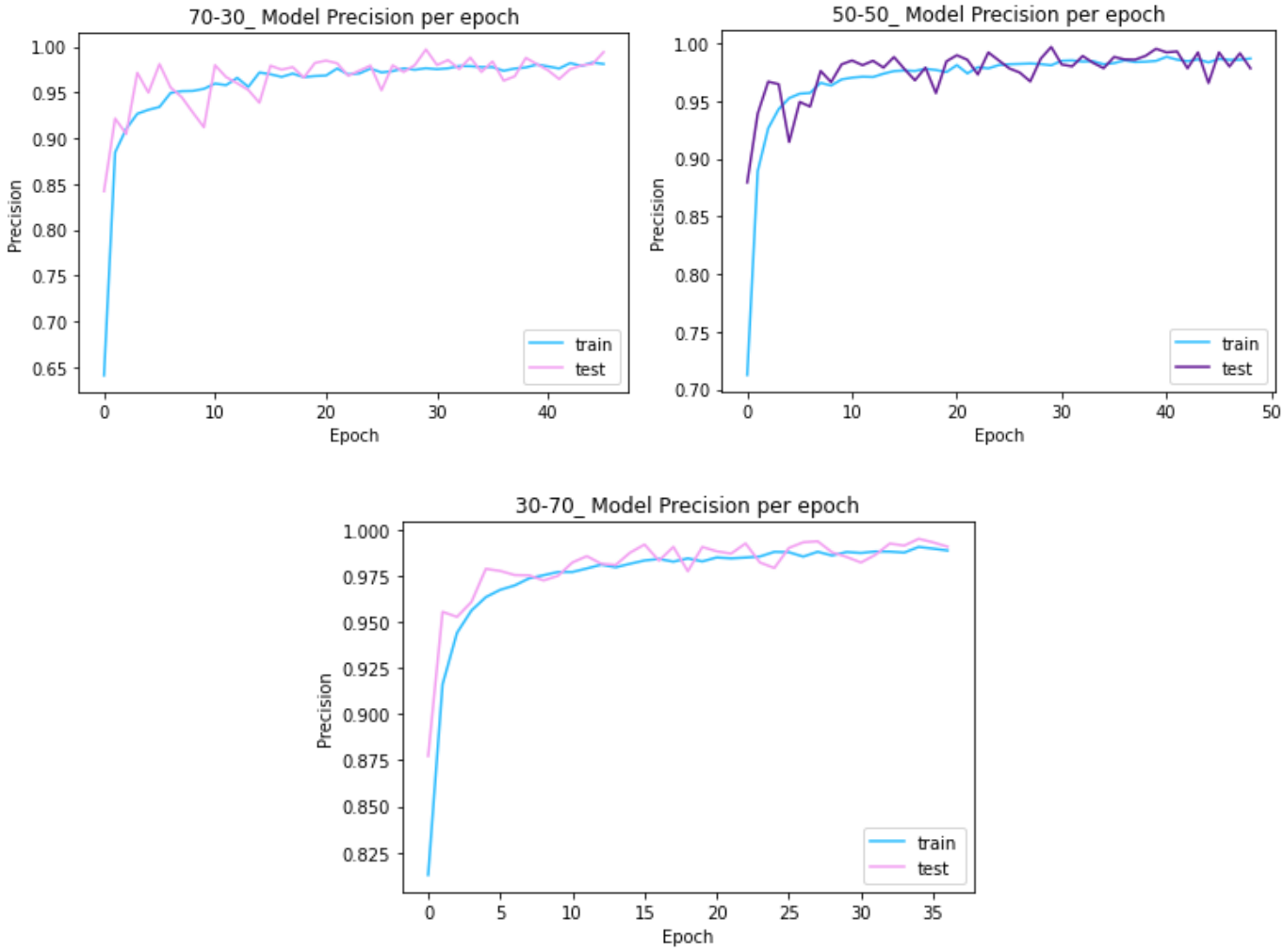


Figura 35: Precisión del modelo entrenado con el dataset 70%-30% frente a los de 50%-50% y 30% - 70%. (Imagen propia)

En cuanto a la precisión, el análisis es parecido al de la exactitud, aunque varía tanto como el del dataset entrenado con 50% cáncer y 50% normal al principio de entrenamiento, aunque después presenta menos diferencia entre test y train que éste, mostrando un menor sobreajuste parecido al del dataset entrenado con 30% cáncer y 70% normal.

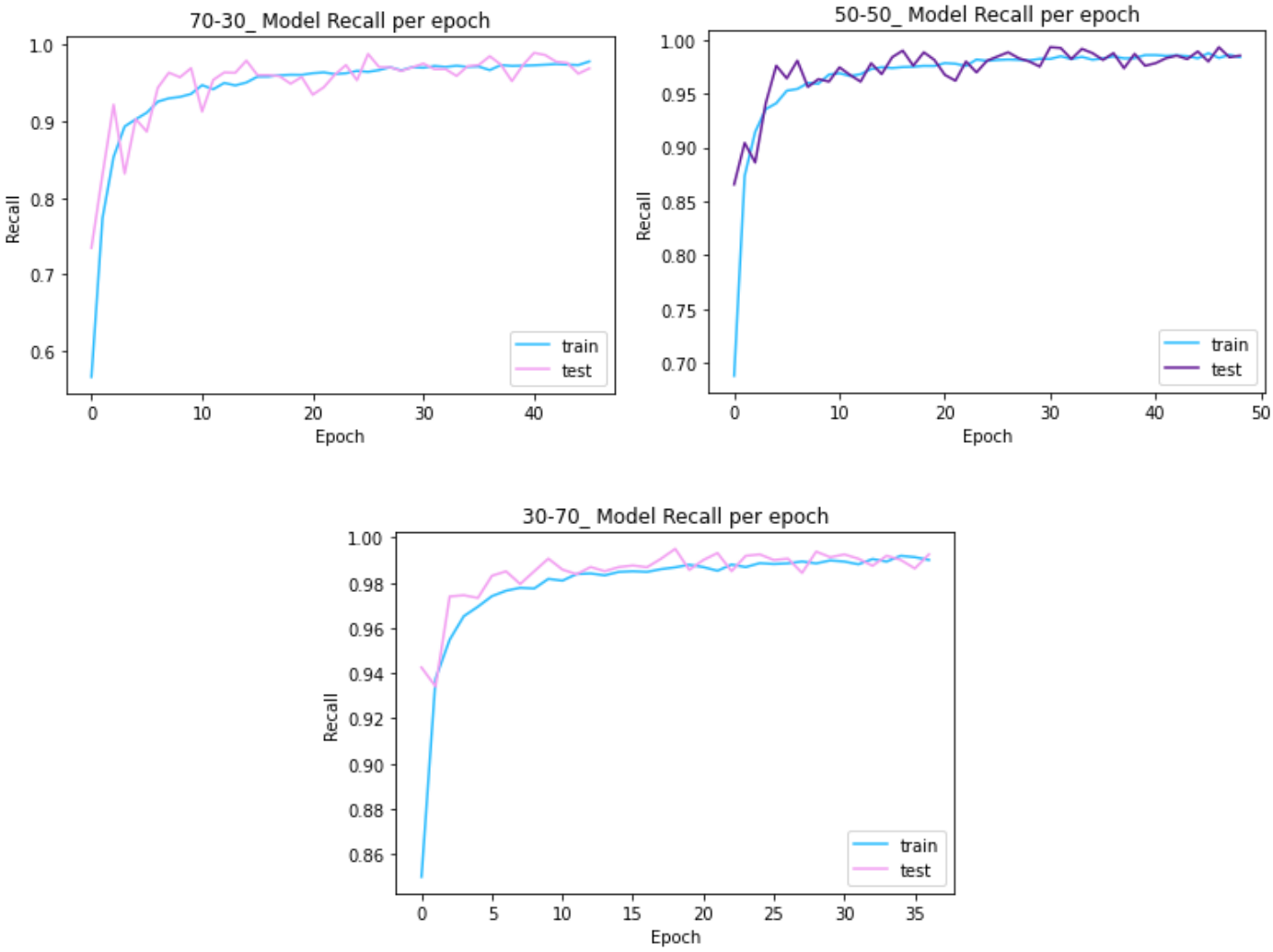


Figura 36: Sensibilidad del modelo entrenado con el dataset 70%-30% frente a los de 50%-50% y 30% - 70%. (Imagen propia)

En cuanto a la sensibilidad, presenta más variabilidad al principio del entrenamiento que los dos conjuntos anteriores, aunque los resultados finales son muy próximos entre sí.

Podríamos concluir que los resultados de los entrenamientos son muy parecidos entre sí.

6.8.2 Resultados de la clasificación:

Finalmente, los resultados de la clasificación del modelo entrenado con este dataset desequilibrado, 70% cáncer y 30% normal, son bastante parecidos a los del modelo entrenado por el dataset equilibrado 50%-50% y el primero, entrenado por 30% cáncer y 70% normal. Lo podemos comprobar con la siguiente matriz de confusión:

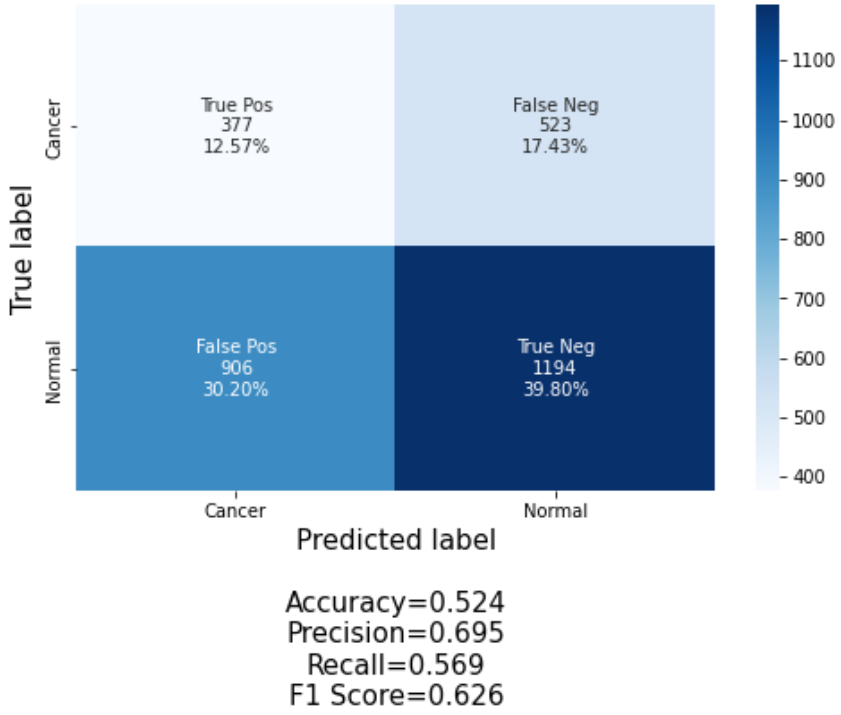


Figura 37: Matriz de confusión de la clasificación con el modelo 70%-30%. (Imagen propia)

Creemos que esta similitud podría estar producida por la falta de variedad en las imágenes con las que se entrenan los 3 modelos. Por ello, el siguiente paso fue formar datasets con imágenes sintéticas generadas mezcladas con imágenes reales del dataset original, en diferentes proporciones.

Reentrenar VGG16 con datasets formados por imágenes sintéticas generadas mezcladas con reales.

Como hemos podido comprobar, los resultados del clasificador entrenado con distintas configuraciones de datasets formados exclusivamente por imágenes sintéticas distan mucho de lo deseable. Por tanto, para evitar que el modelo caiga en sobreajuste, la opción más viable era mezclar las imágenes sintéticas generadas con imágenes reales del dataset original. De esta forma, el modelo tendrá más capacidad de generalización, al aprender de las diferentes características de imágenes reales y sintéticas. Siguiendo lo indicado en el apartado 6.1, generamos los datasets mezclados con diferentes porcentajes de imágenes reales y sintéticas y entrenamos el modelo con ellos, obteniendo los siguientes resultados.

Configuración 30% cáncer – 70% normal, con 95% imágenes sintéticas y 5% reales.

Resultados del entrenamiento

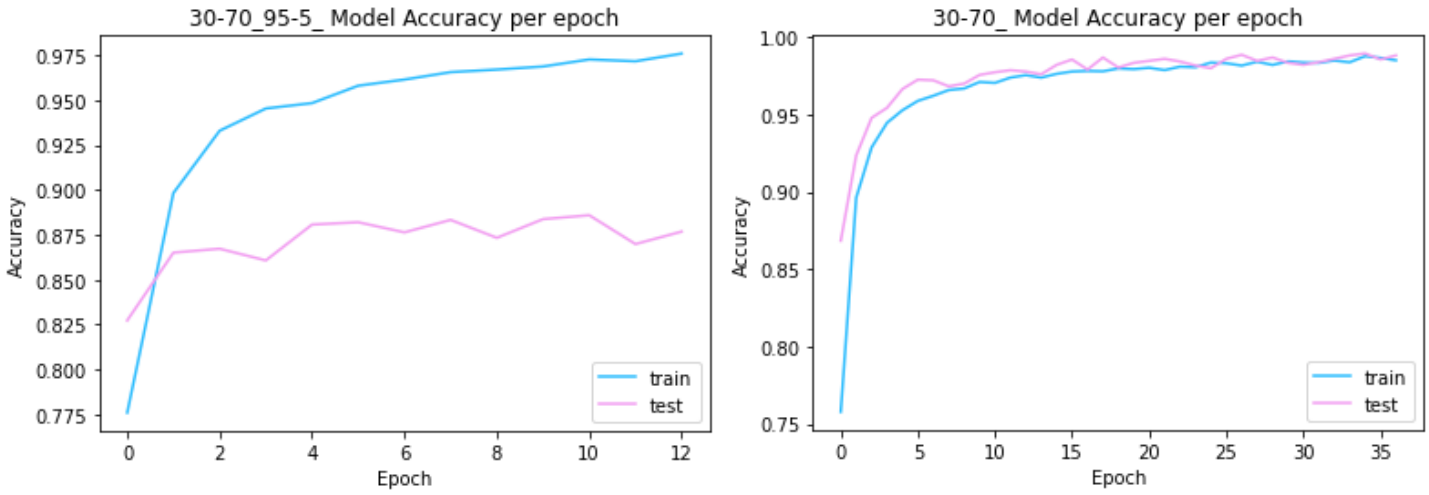


Figura 38: Comparación de la exactitud entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Lo primero que observamos, comparando los resultados, es que la diferencia entre precisión sobre el subconjunto de test se diferencia más en el modelo entrenado con un 5% de imágenes reales. Aunque esta diferencia parezca abrupta, si nos fijamos en la escala del eje vertical, la diferencia es tan solo del orden de décimas. Esta diferencia se observa debido a que el subconjunto de test ahora presenta imágenes reales, poco presentes en el dataset de entrenamiento y tiende a fallar cuando debe clasificar una de las muestras reales. Aunque parezca algo negativo (presenta más sobreajuste que el modelo sintético puro), el modelo tiene que aprender a identificar las imágenes reales y seguramente obtenga mejores resultados en un ejercicio de clasificación real.

Además, si nos fijamos en el eje horizontal, el modelo ha considerado que a partir de la época 12 ya no se mejoraba la clasificación y decide parar el entrenamiento. En cuestión de optimización de tiempo, el nuevo modelo es mucho más rápido de entrenar.

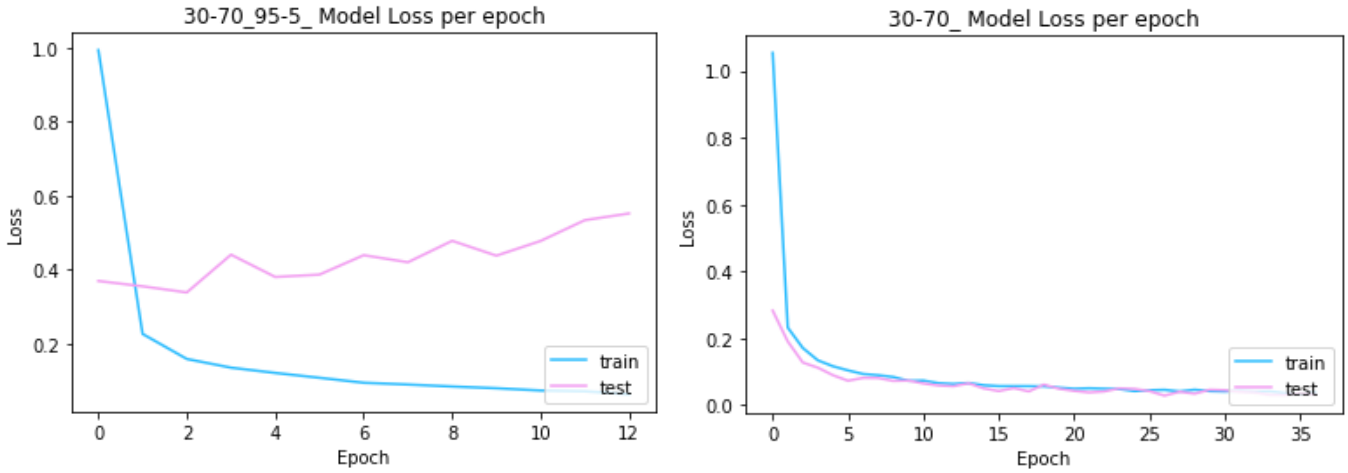


Figura 39: Comparación de la pérdida entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la pérdida, el nuevo modelo (izq) presenta una diferencia en la evolución de la función con el tiempo en cuanto al subconjunto de test. No evoluciona favorablemente, convergiendo a un valor menor como el modelo puramente sintético. Este comportamiento es problemático y tendremos que comprobar si se resuelve introduciendo un mayor porcentaje de imágenes reales con las sintéticas. La pérdida, al ser la métrica con la que la función de callback detiene el entrenamiento si no mejora dentro de un margen temporal, ha provocado que el entrenamiento durase solo 12 épocas, como comentábamos en el resultado anterior.

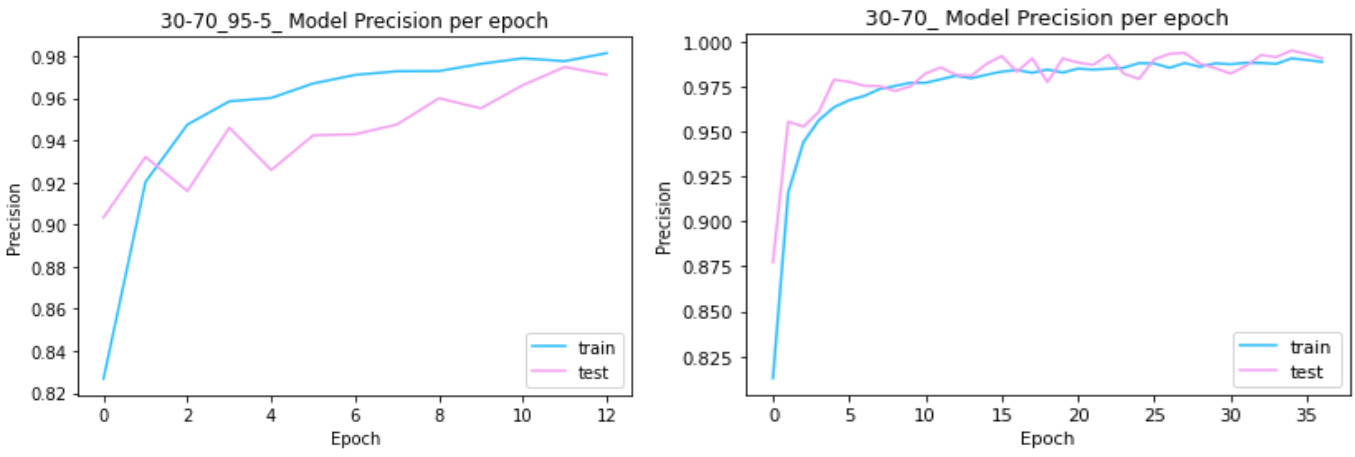


Figura 40: Comparación de la precisión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la precisión, podemos observar que la diferencia entre train y test no es tan grande como en la exactitud y converge favorablemente a resultados cada vez más cerca del 1.

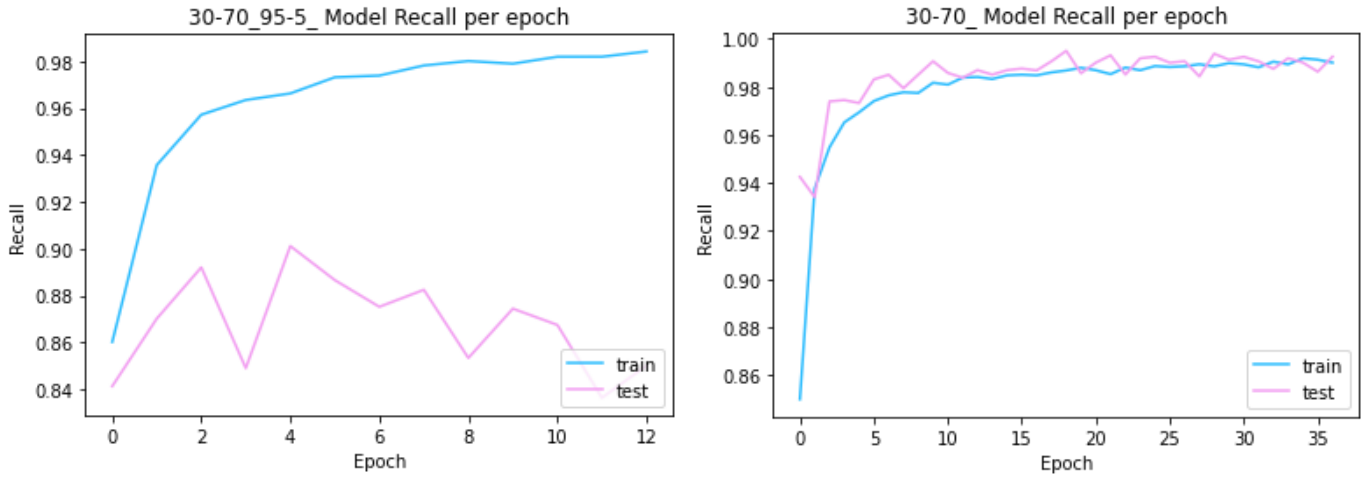


Figura 41: Comparación de la sensibilidad entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Igual que la pérdida, la sensibilidad del modelo se desploma, comparando con el modelo puramente sintético. El ratio de los verdaderamente positivos decrece, por lo que el modelo no es capaz de diferenciar bien cuando es cáncer o normal.

Resultado de la clasificación

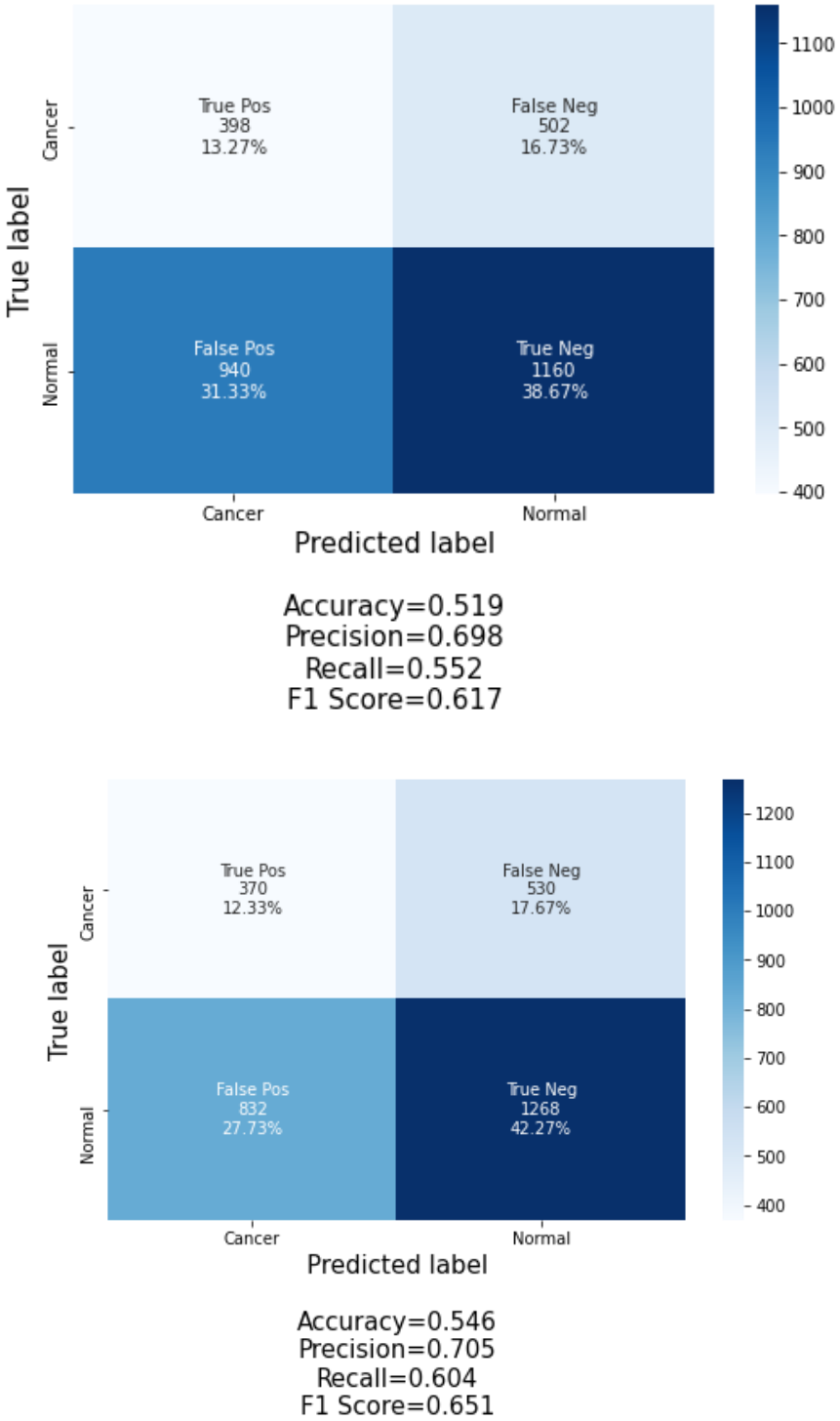


Figura 42: Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (arriba) y el entrenado con un dataset 30%-70% puramente sintético (abajo). (Imagen propia)

Observamos que la calidad en general de los resultados baja respecto a la clasificación que se realizó con el modelo puramente sintético. Creemos que esto se debe a que la introducción de un porcentaje de imágenes reales tan pequeño, el 5%, no influye positivamente en los resultados del modelo. Es por ello que realizamos un segundo entrenamiento y clasificación con un porcentaje un poco más alto, del 10%. Este porcentaje es el máximo al que podemos aspirar, dado que el dataset DDSM no es tan grande como para aumentar el número de imágenes reales introducidas más allá del 10%. Sin embargo, subir dicho número tampoco sería coherente con el objetivo proyecto, dado que buscábamos una herramienta para poblar datasets con un número pequeño de imágenes reales, como es el caso.

Configuración 30% cáncer – 70% normal, con 90% imágenes sintéticas y 10% reales.

Resultados del entrenamiento

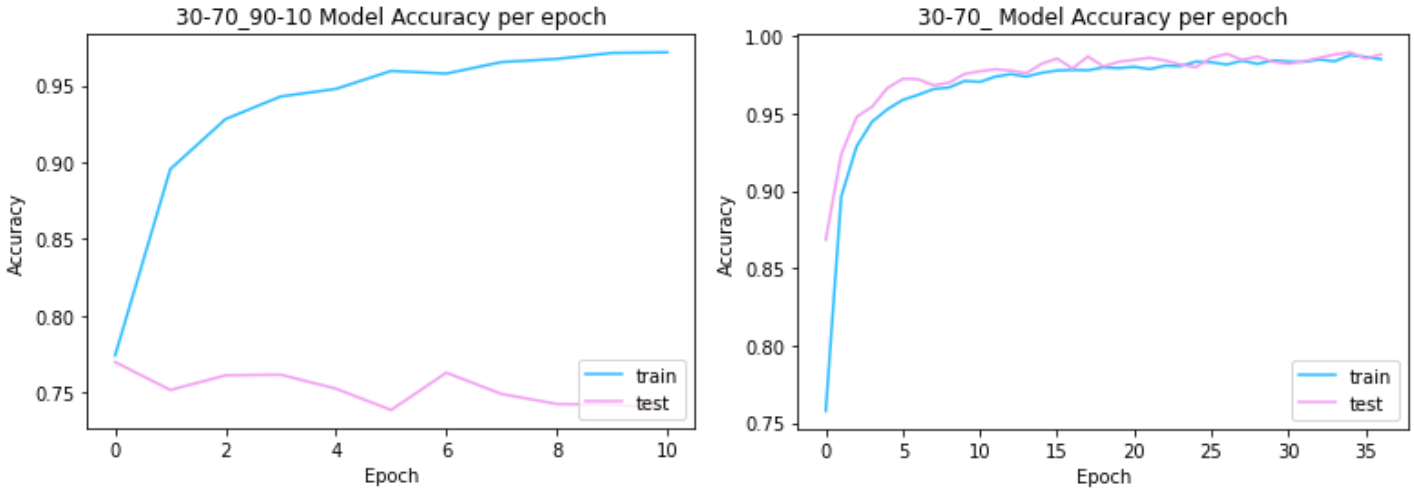


Figura 43: Comparación de la exactitud entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Como en el caso del modelo entrenado por el dataset mezclado 95% sintético - 5% real, se aprecia una diferencia más abrupta entre los resultados de la clasificación sobre el subset de train y test, indicando más sobreajuste del modelo.

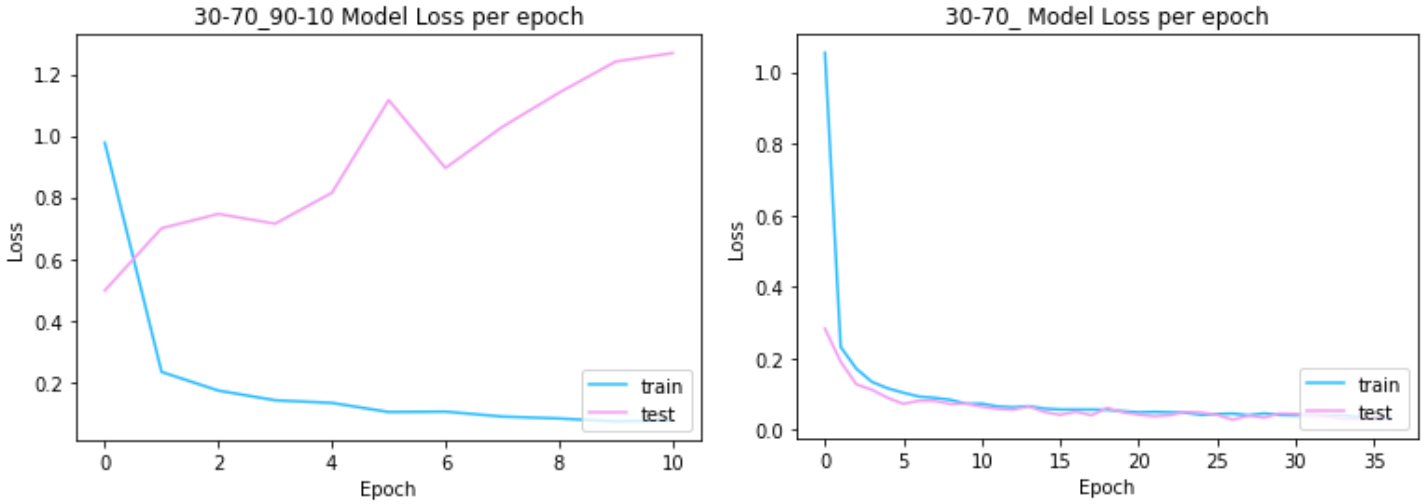


Figura 44: Comparación de la pérdida entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la pérdida, observamos que crece más rápido aún que el caso 95%-5%, dado que ha parado el entrenamiento 2 épocas antes, en la diez. Estos resultados parecen reflejar que, como en la mayoría de las aplicaciones de aprendizaje automático, sería importante buscar un compromiso entre la pérdida y las demás métricas de clasificación, para encontrar un resultado óptimo.

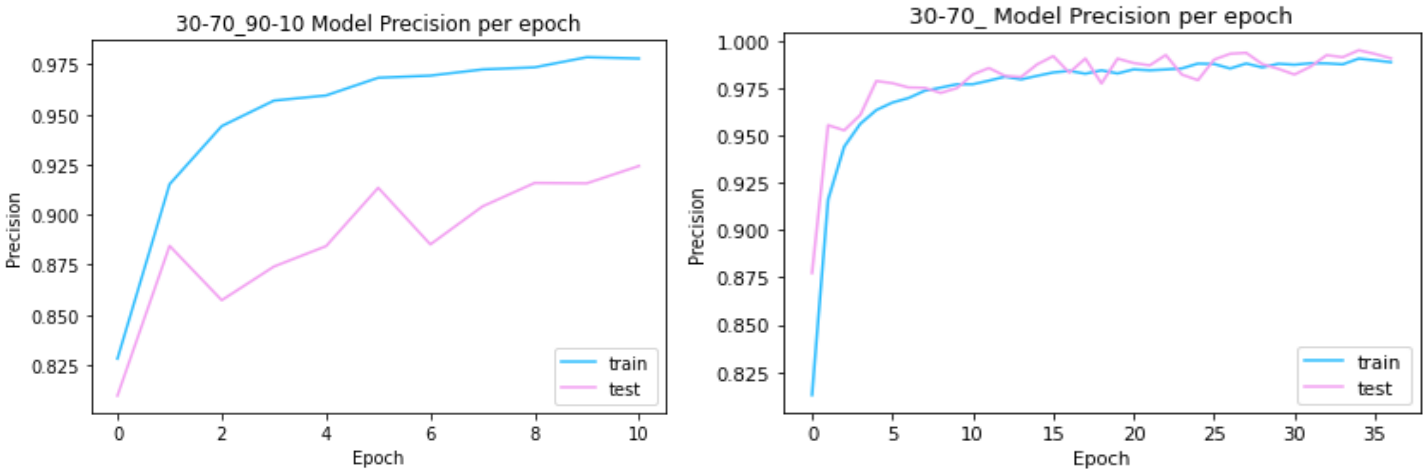


Figura 45: Comparación de la precisión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la precisión, refleja un poco más de sobreajuste del modelo, pero se comporta adecuadamente, convergiendo a valores más altos según pasa el tiempo de entrenamiento.

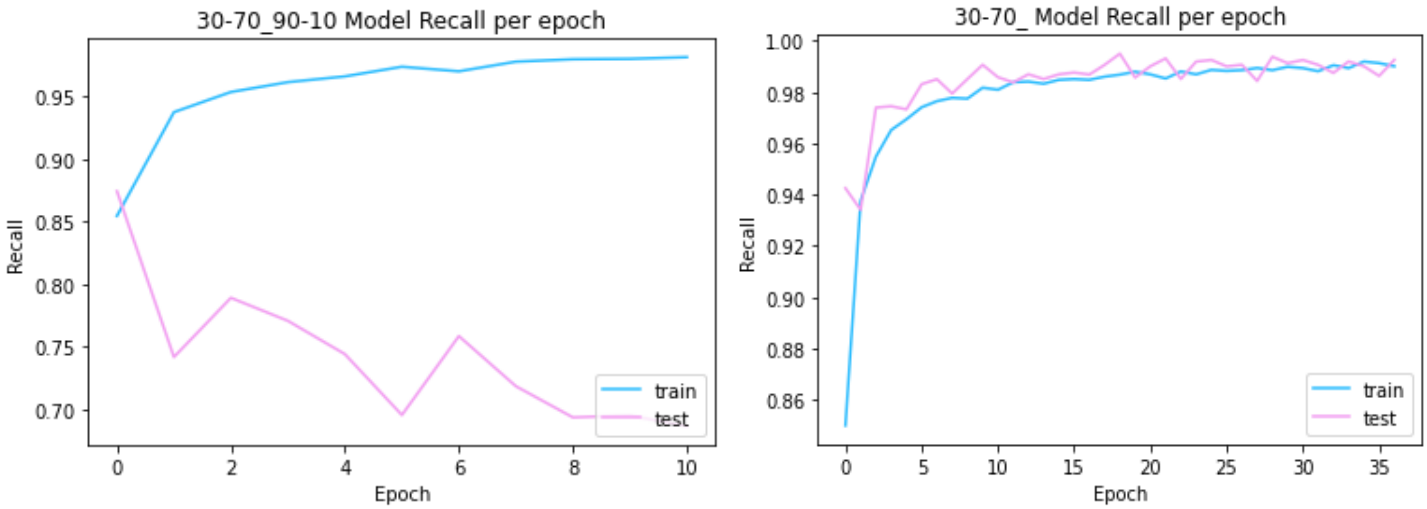
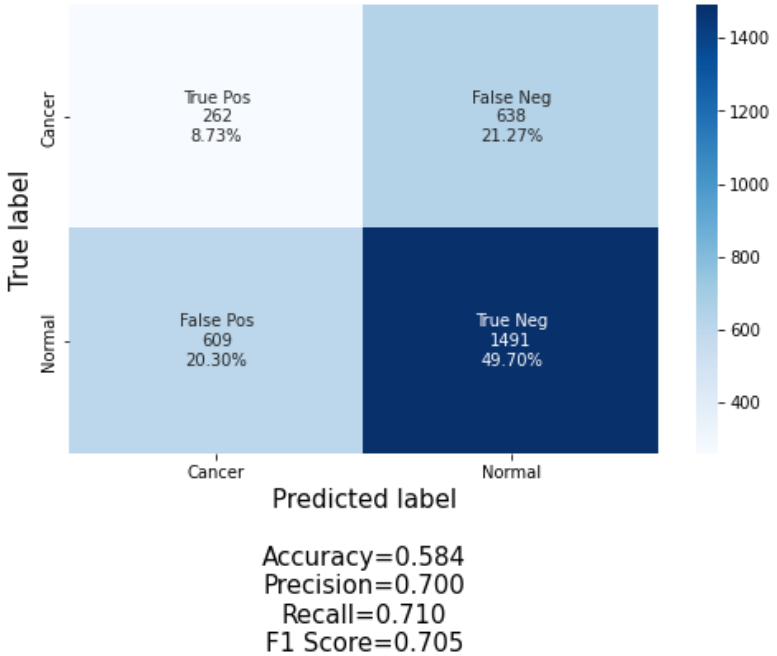


Figura 46: Comparación de la sensibilidad entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

La sensibilidad, como en el caso anterior de 95% - 5%, también decrece y más abruptamente. Como decíamos, la introducción de imágenes reales al dataset genera un poco más de sobreajuste en el proceso de entrenamiento, pero realmente también lo reduce cuando se prueba el modelo en una clasificación real, como veremos a continuación.

Resultados de la clasificación



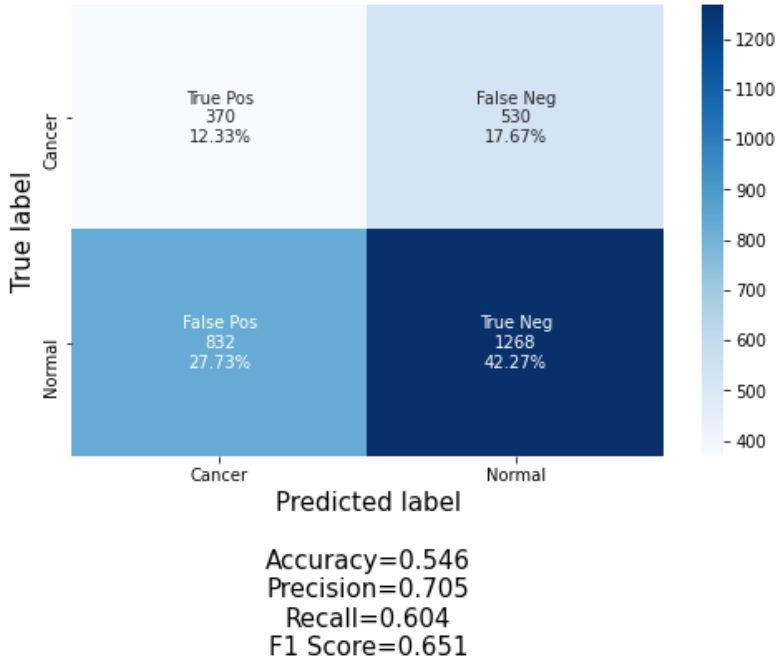


Figura 47: Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (arriba) y el entrenado con un dataset 30%-70% puramente sintético (abajo). (Imagen propia)

Como podemos observar, mejoran todas las métricas salvo la precisión, que cae solamente en un 0.005. parece ser que la introducción de imágenes reales al dataset de entrenamiento ha mejorado las métricas del modelo entrenado solo con imágenes sintéticas. Sin embargo, algo preocupante es que el porcentaje de Falsos Negativos, recordemos que es una métrica crucial en diagnóstico médico, ha subido. Será muy importante comprobar cómo se comporta dicha métrica en los resultados del dataset equilibrado (50% cáncer 50% normales) y mezclado con imágenes reales.

Configuración 50% cáncer – 50% normal, con 95% imágenes sintéticas y 5% reales.

Resultados del entrenamiento

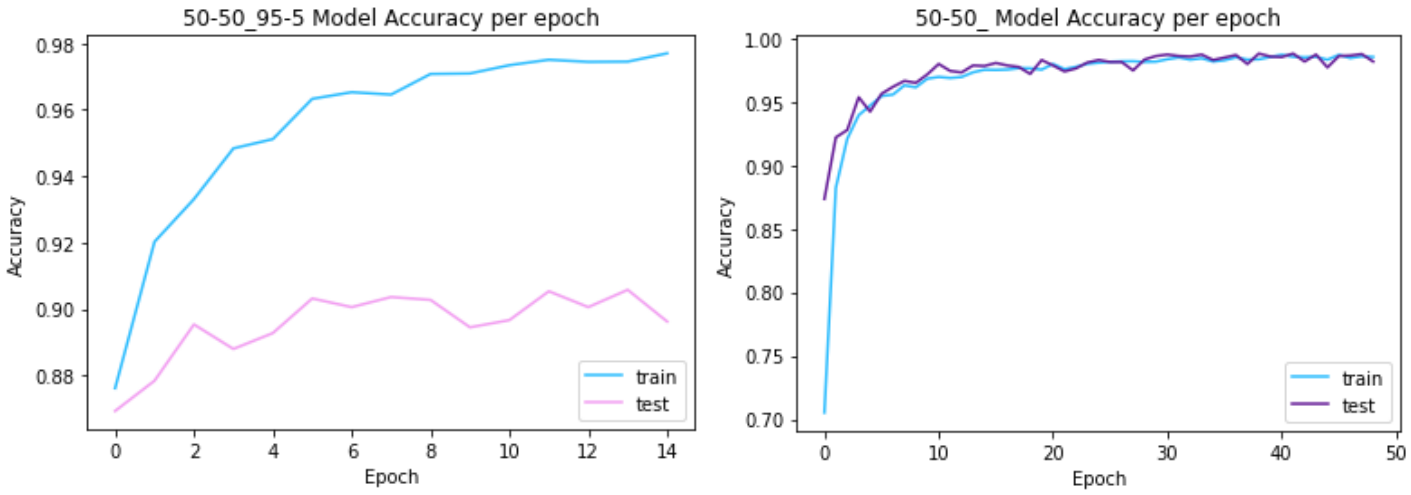


Figura 48: Comparación de la exactitud entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Siguiendo el tono general de los últimos entrenamientos, la distancia de los resultados de la clasificación entre test y train aumenta respecto a la de los resultados del modelo entrenado con un dataset 50% cáncer 50% normal puramente sintético.

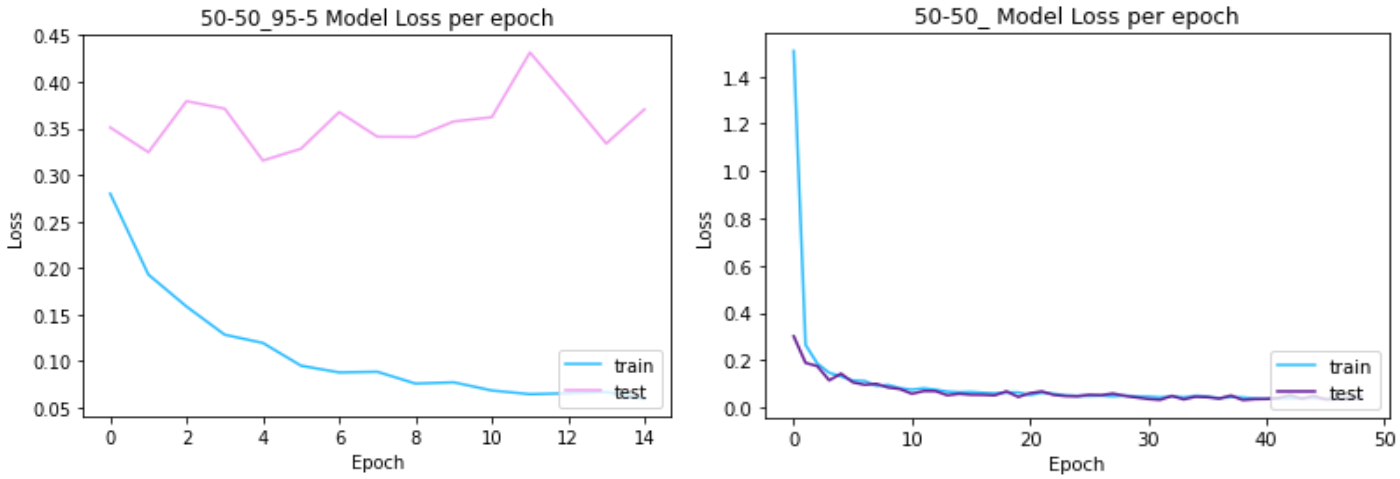


Figura 49: Comparación de la pérdida entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la pérdida, aunque más alta en el subset de test que en el de entrenamiento, en este caso no desciende continuamente, provocando que el mecanismo de callback termine el entrenamiento considerando que no va a mejorar el resultado con el tiempo. En este caso, la pérdida varía alrededor del valor fijo de 0.35 aproximadamente.

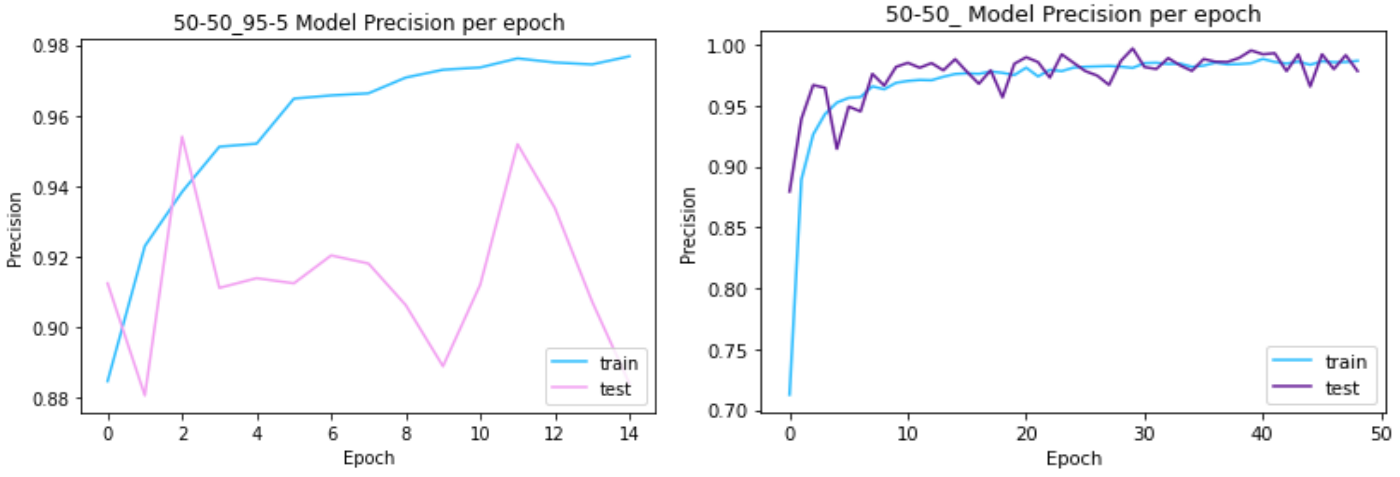


Figura 50: Comparación de la precisión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Si bien la precisión parece que varía abruptamente, en realidad los cambios son del orden de las centésimas. El mínimo alcanzado es del 0.88, un muy buen valor para la precisión.

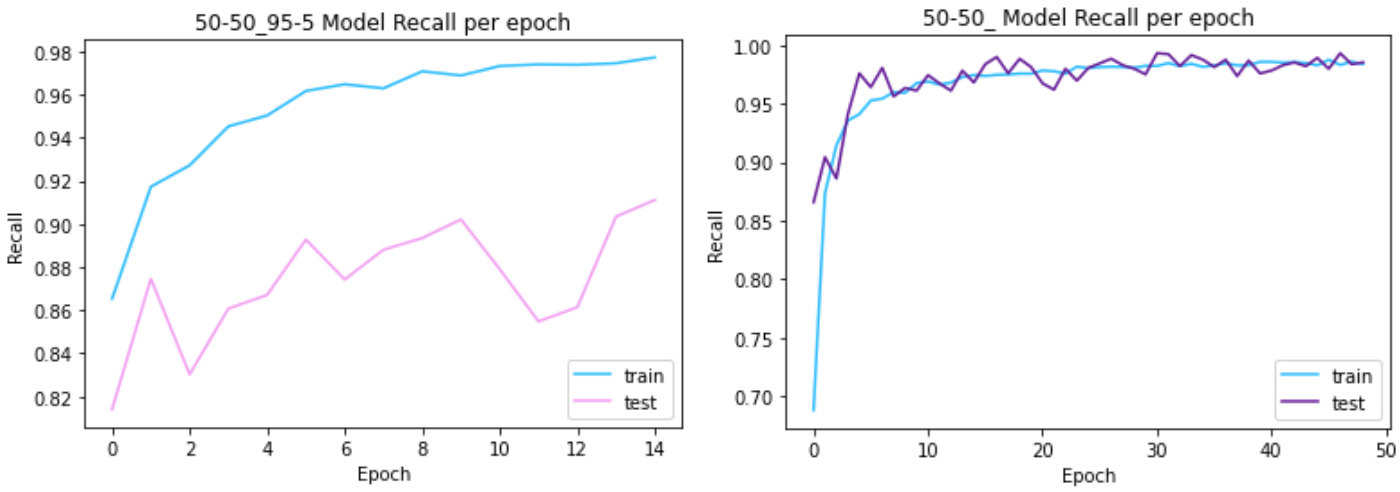
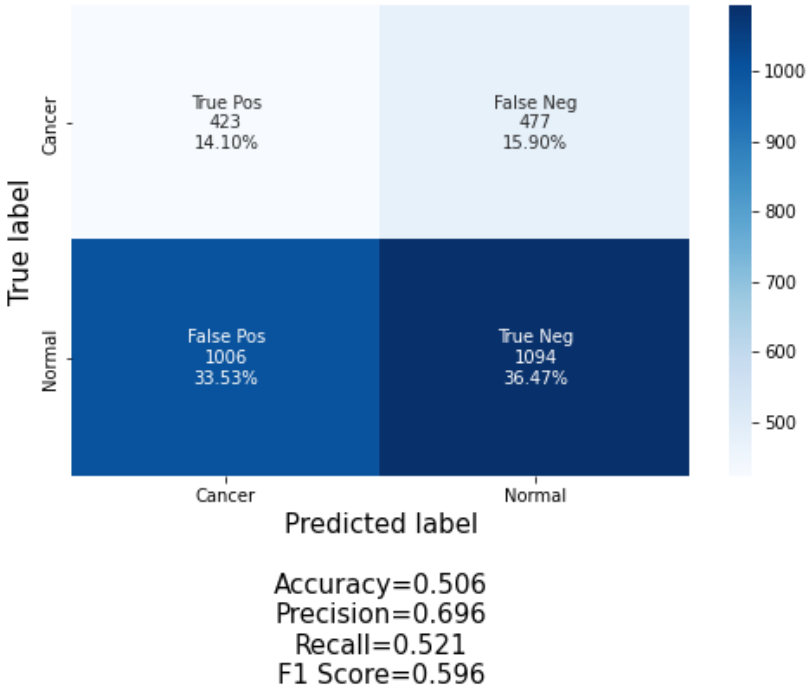


Figura 51: Comparación de la sensibilidad entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Finalmente, la precisión ya no desciende, si no que se mantiene constante, intuyéndose incluso una pequeña subida hacia el final del entrenamiento, cerca del 0.91.

Resultados de la clasificación



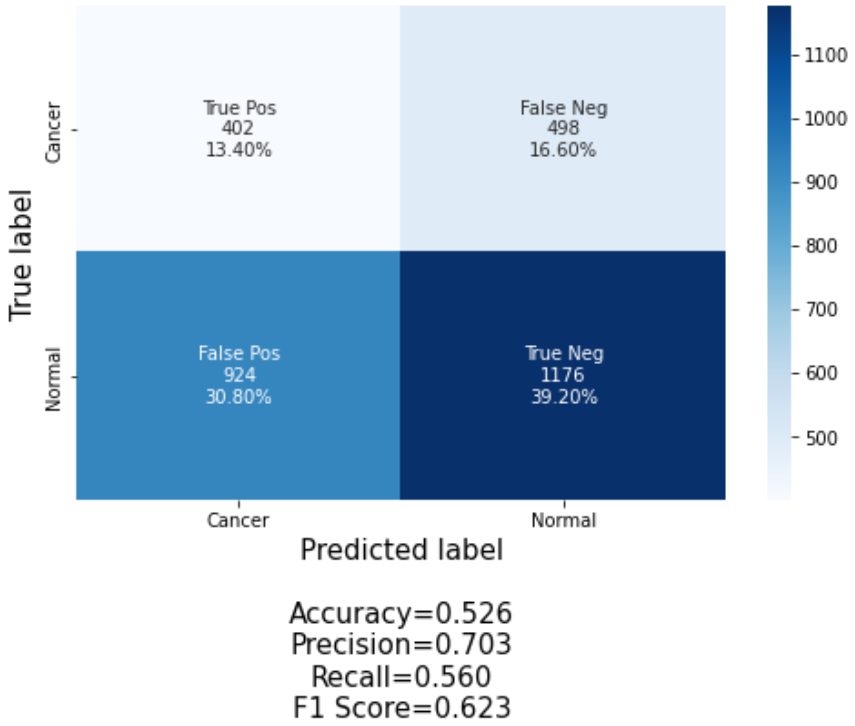


Figura 52: Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (arriba) y el entrenado con un dataset 50% -50% puramente sintético (abajo). (Imagen propia)

En cuanto a la clasificación, los resultados son pobres, incluso peores que los registrados por el modelo entrenado con el dataset puramente sintético. Lo único positivo es que el porcentaje de falsos negativos desciende ligeramente, pero no es una mejora suficientemente buena como para declarar que este modelo funcione mejor. Veremos a continuación si mejoran los resultados con un porcentaje de imágenes sintéticas ligeramente mayor.

Configuración 50% cáncer – 50% normal, con 90% imágenes sintéticas y 10% reales.

Resultados del entrenamiento

Nos encontramos ante el entrenamiento que mayor diferencia presenta entre las funciones de train y test, es decir, mayor sobreajuste del modelo en esta etapa de entrenamiento. Además, es la que peores valores refleja en todas las métricas, debido a la introducción de más imágenes reales en el entrenamiento que el modelo anterior. Si bien estos resultados de entrenamiento no son determinantes a la hora de medir la calidad del modelo clasificador, esperábamos que fueran mejores, dado que es la configuración de dataset que más prometía.

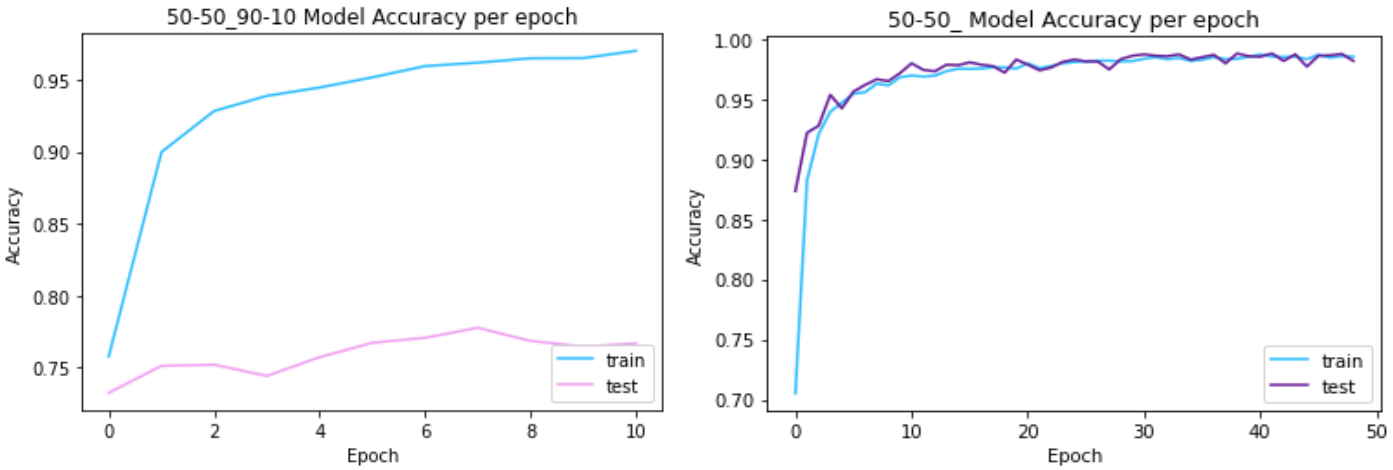


Figura 53: Comparación de la exactitud entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

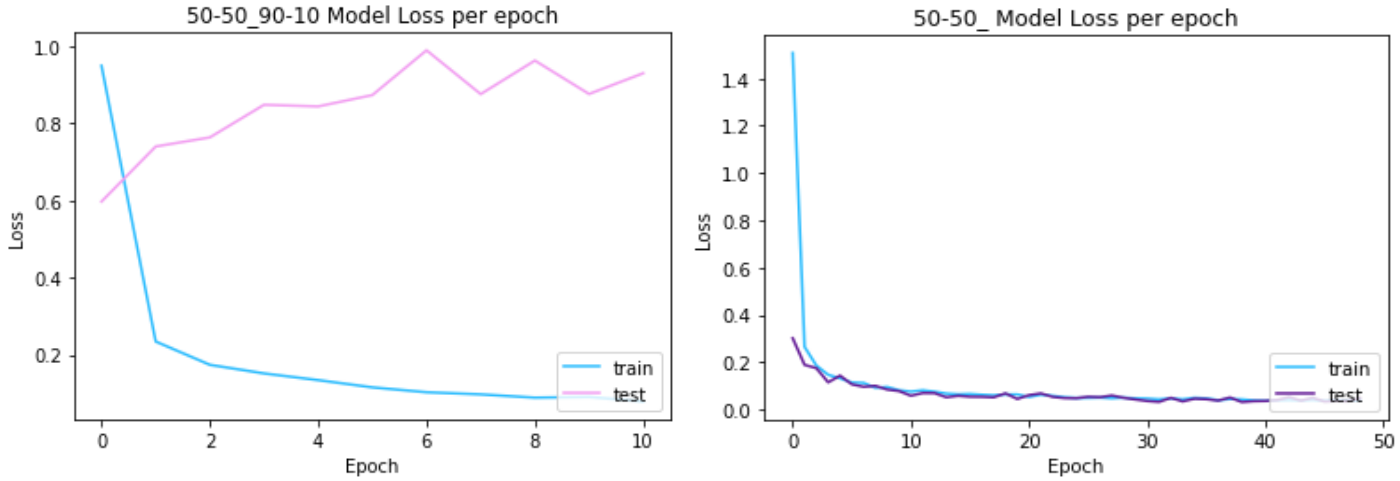


Figura 54: Comparación de la pérdida entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

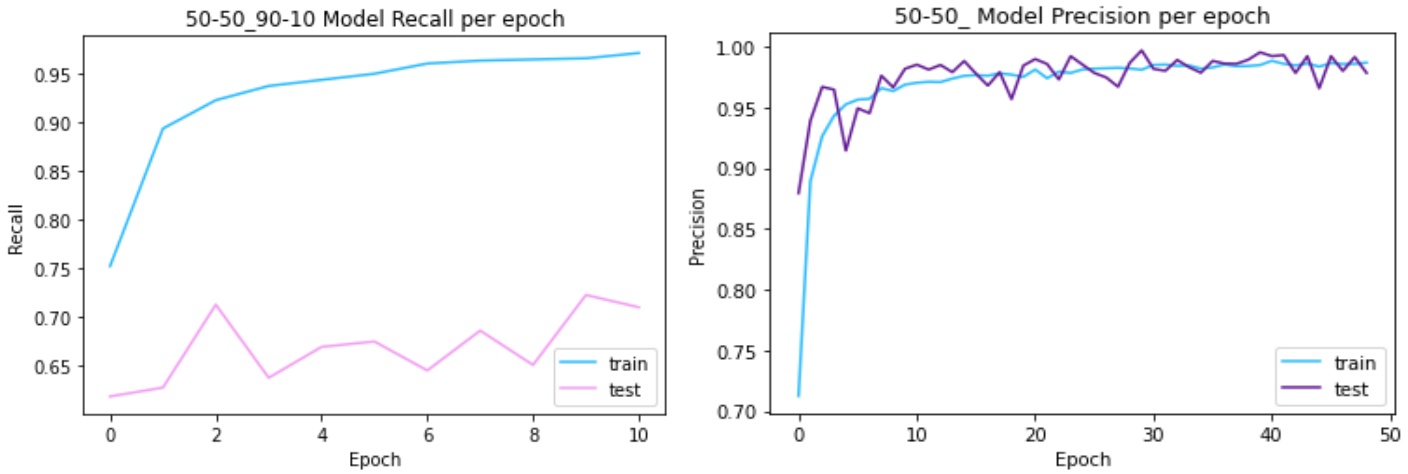


Figura 55: Comparación de la precisión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

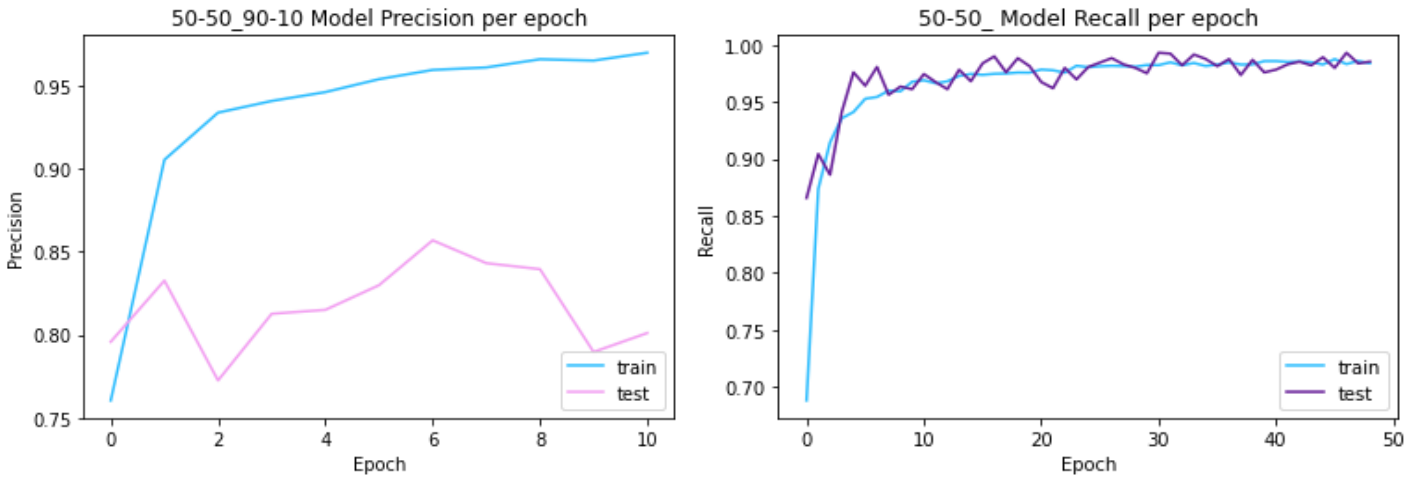
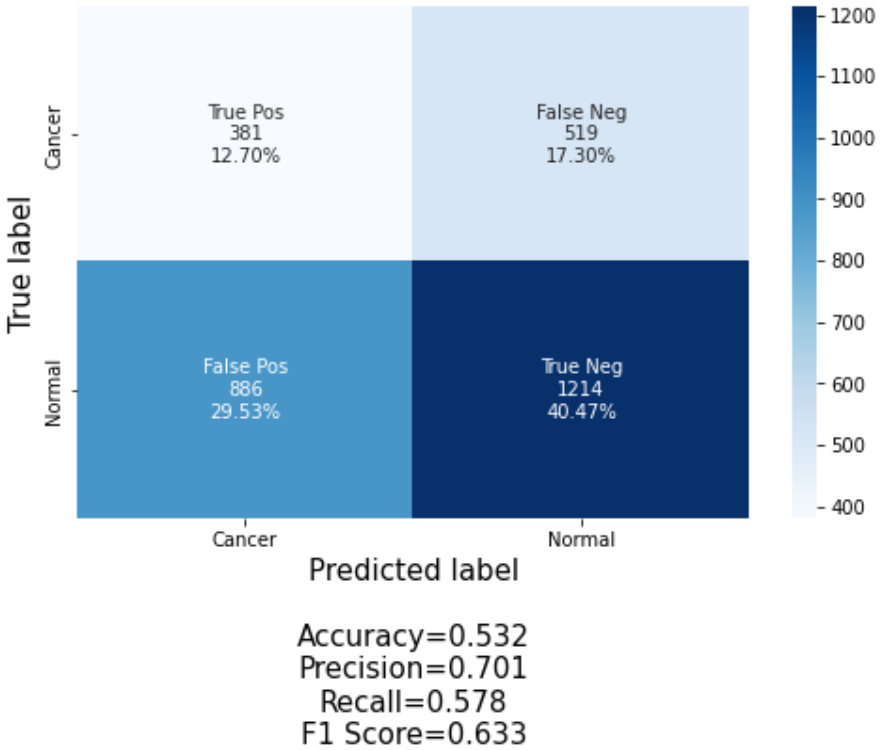


Figura 56: Comparación de la sensibilidad entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Resultados de la clasificación



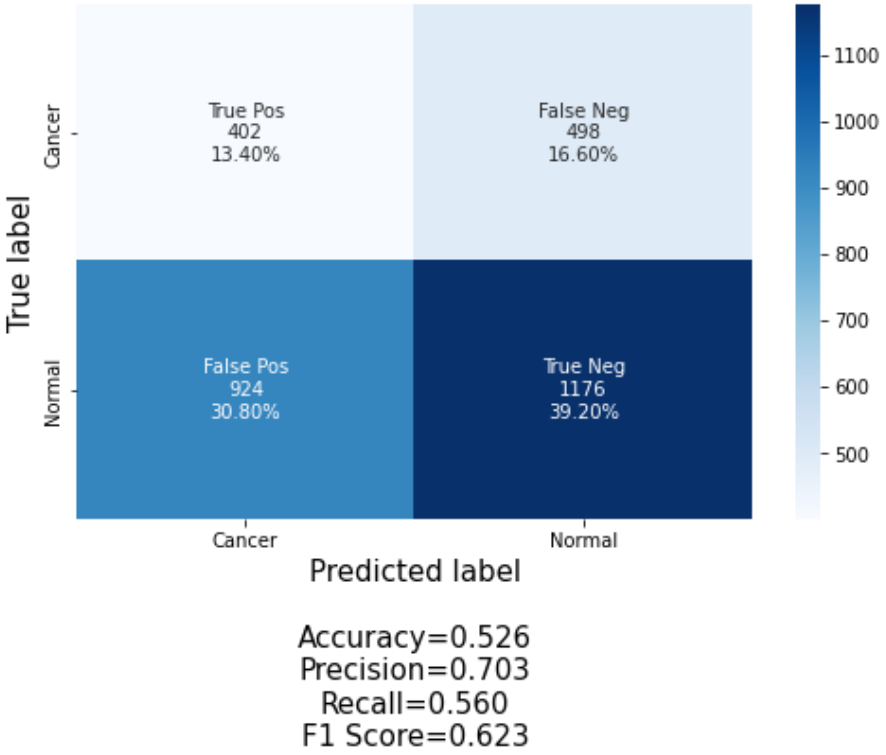


Figura 57: Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (arriba) y el entrenado con un dataset 50% -50% puramente sintético (abajo). (Imagen propia)

Como decíamos, esta configuración se suponía la más adecuada, dado que el dataset con el que se ha entrenado el modelo estaba equilibrado, 50% cáncer y 50% normal, además de contar con más imágenes reales para aportar variedad de características a las imágenes de entrenamiento. Sin embargo, presenta unos resultados poco mejores que los que conseguía el modelo entrenado con una configuración puramente sintética.

Por último, probaremos las configuraciones desequilibradas de 70% cáncer y 30% normales, antes de avanzar hacia las conclusiones del proyecto.

Configuración 70% cáncer – 30% normal, con 95% imágenes sintéticas y 5% reales.

Resultados del entrenamiento

Ésta última configuración presenta relativamente las mismas evoluciones métricas que su homóloga puramente sintética en cuanto a los resultados de entrenamiento. En general, ninguna métrica desciende a lo largo del tiempo de entrenamiento, siendo la única diferencia con el resto de las configuraciones que presentaban una mezcla de sintéticas con reales. Éste hecho junto con una menor diferencia en las funciones sobre el conjunto de test y de train, indican un mejor rendimiento en el entrenamiento que las configuraciones de 30% cáncer – 70% normal y 50% cáncer – 50% normal. Después comprobaremos si este rendimiento se refleja en los resultados de la clasificación sobre el conjunto de test con imágenes reales.

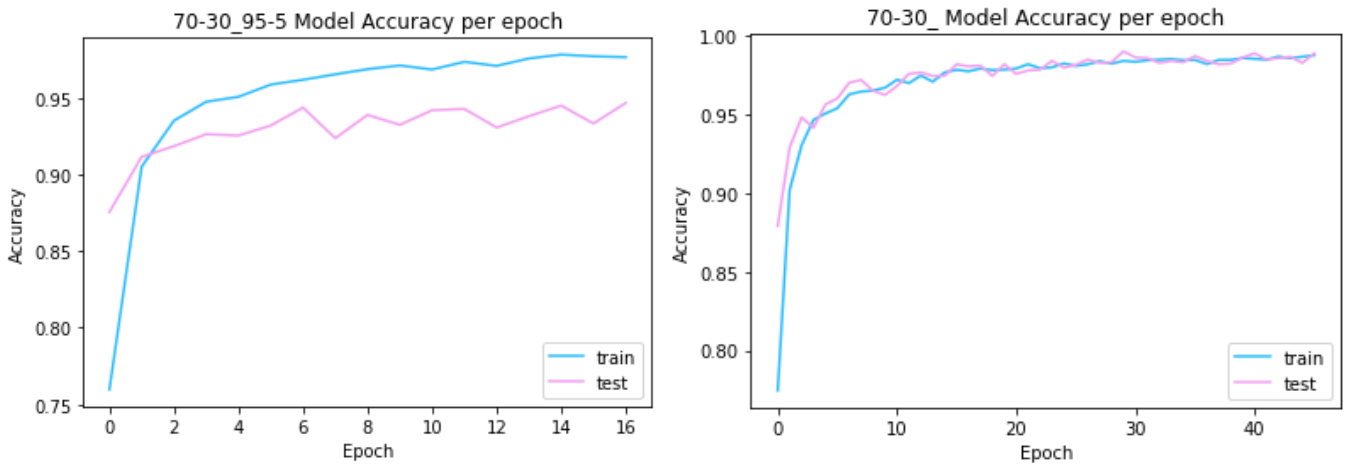


Figura 58: Comparación de la exactitud entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

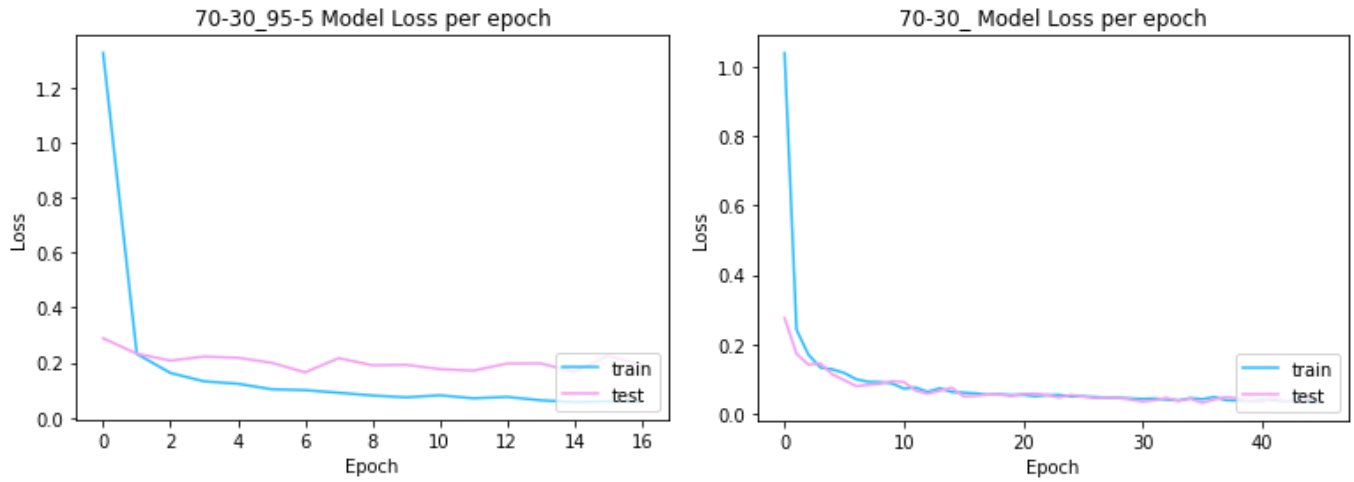


Figura 59: Comparación de la pérdida entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

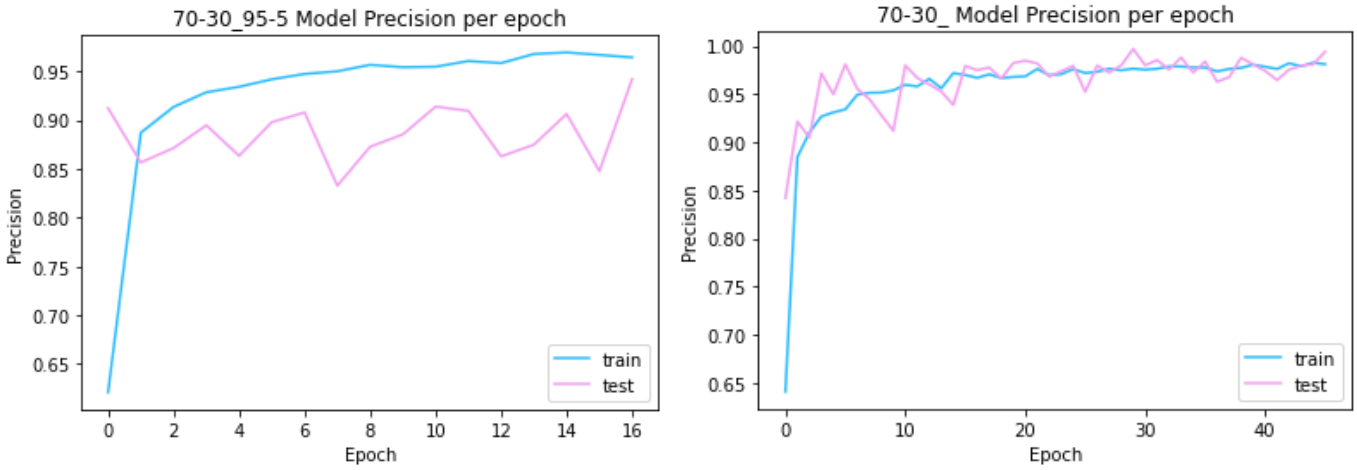


Figura 60: Comparación de la precisión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

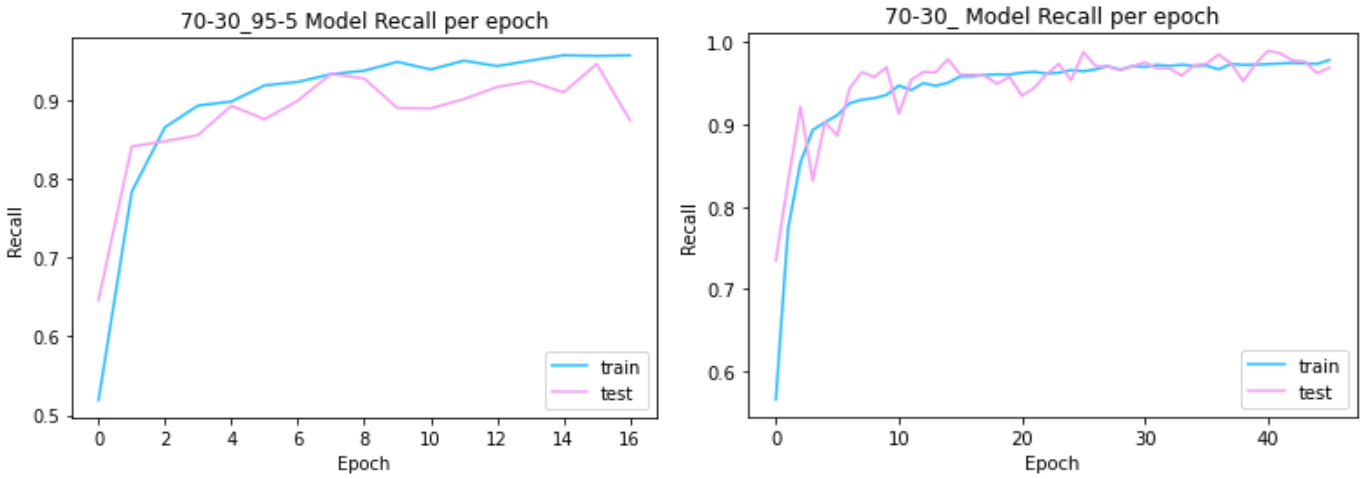
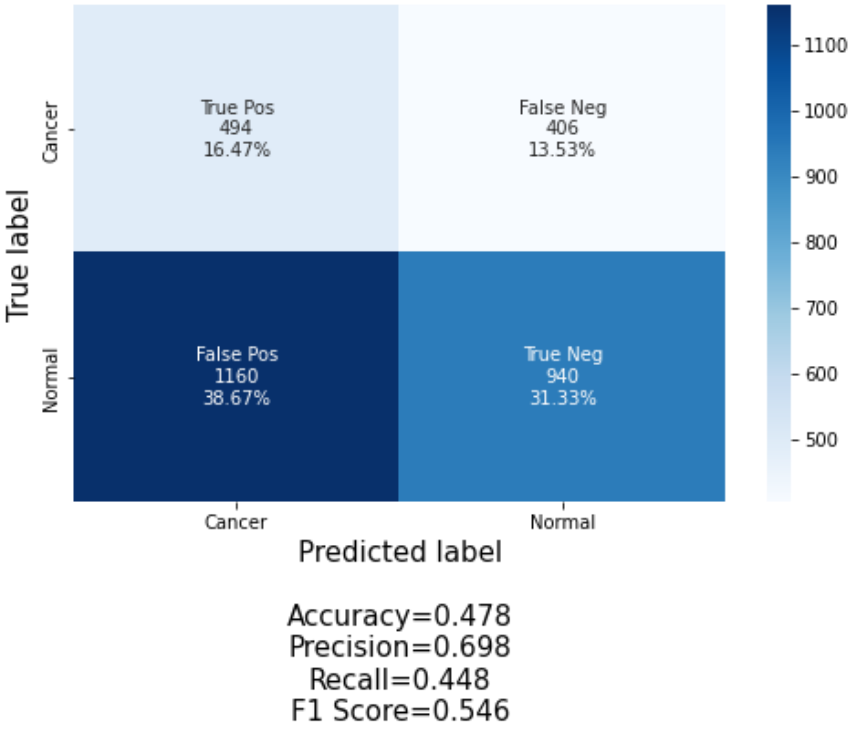


Figura 61: Comparación de la sensibilidad entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Resultados de la clasificación



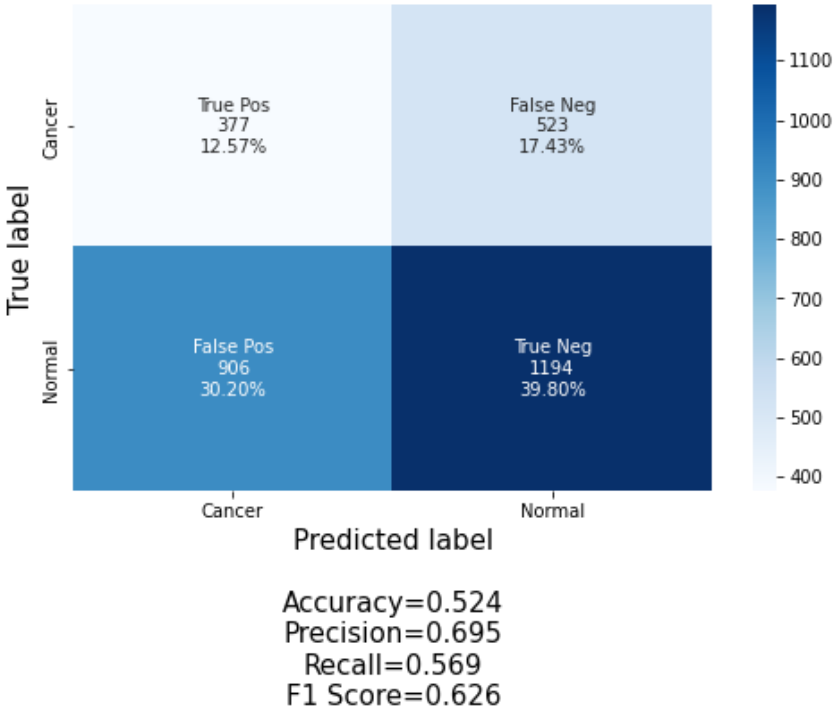


Figura 62: Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (arriba) y el entrenado con un dataset 70% -30% puramente sintético (abajo). (Imagen propia)

Observamos que todas las métricas en general han bajado considerablemente. Los falsos negativos también han bajado pero no es un aspecto suficientemente relevante como para considerar mejor el modelo entrenado con un dataset desequilibrado en cuanto a imágenes con cáncer

Configuración 70% cáncer – 30% normal, con 90% imágenes sintéticas y 10% reales.

Resultado del entrenamiento

Todas las métricas se mantienen constantes durante prácticamente todo el entrenamiento, por lo que una vez agotado el margen de tiempo del proceso de callback, el entrenamiento se detiene en la décima época.

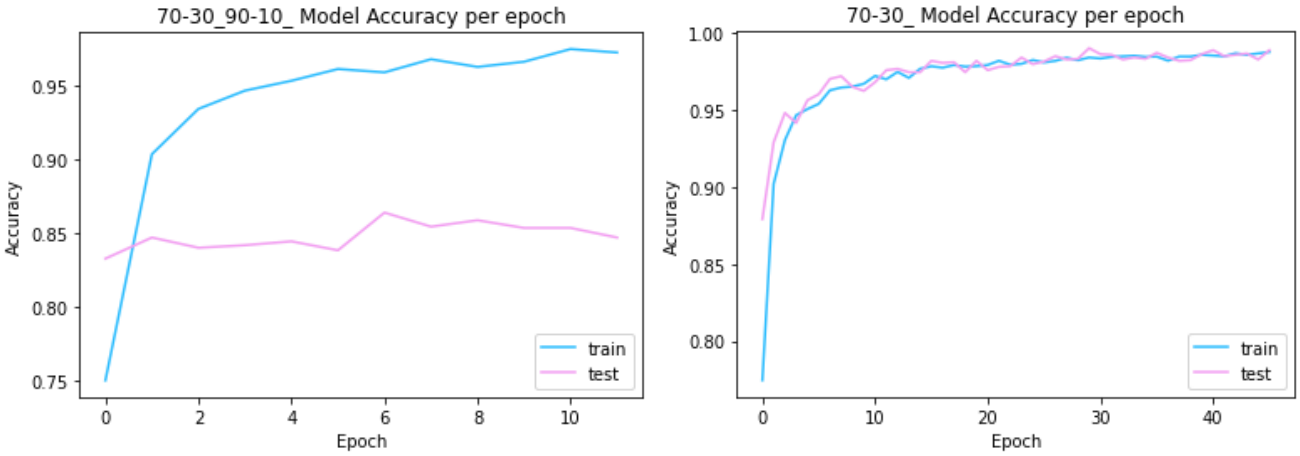


Figura 63: Comparación de la exactitud entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

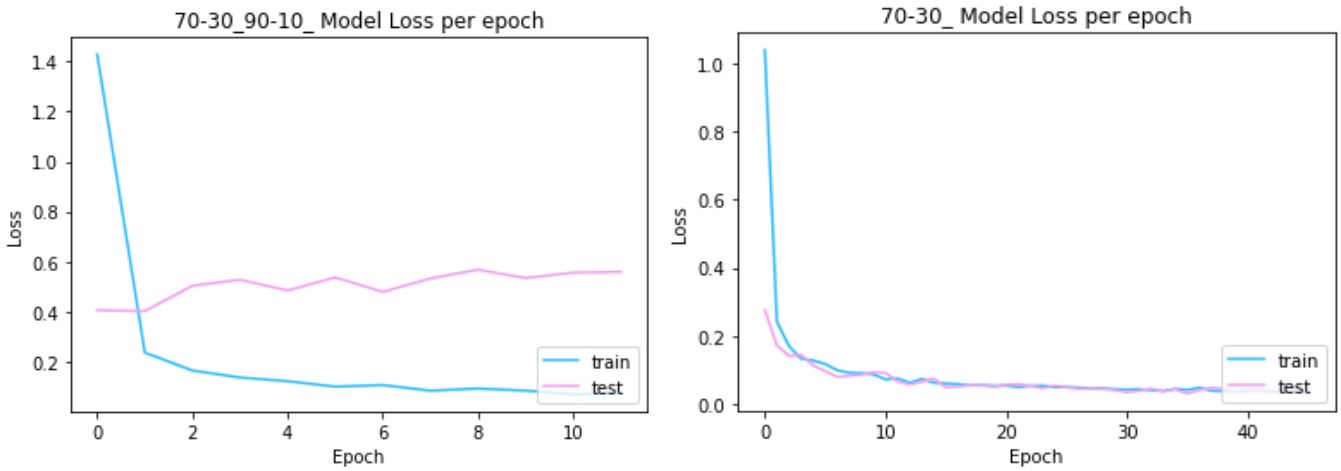


Figura 64: Comparación de la pérdida entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

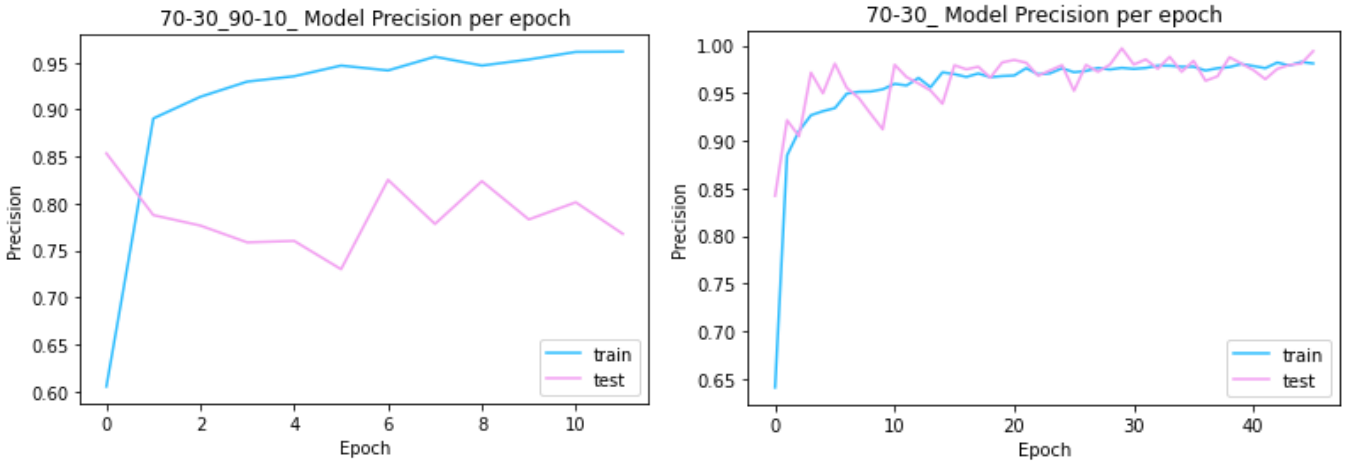


Figura 65: Comparación de la precisión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

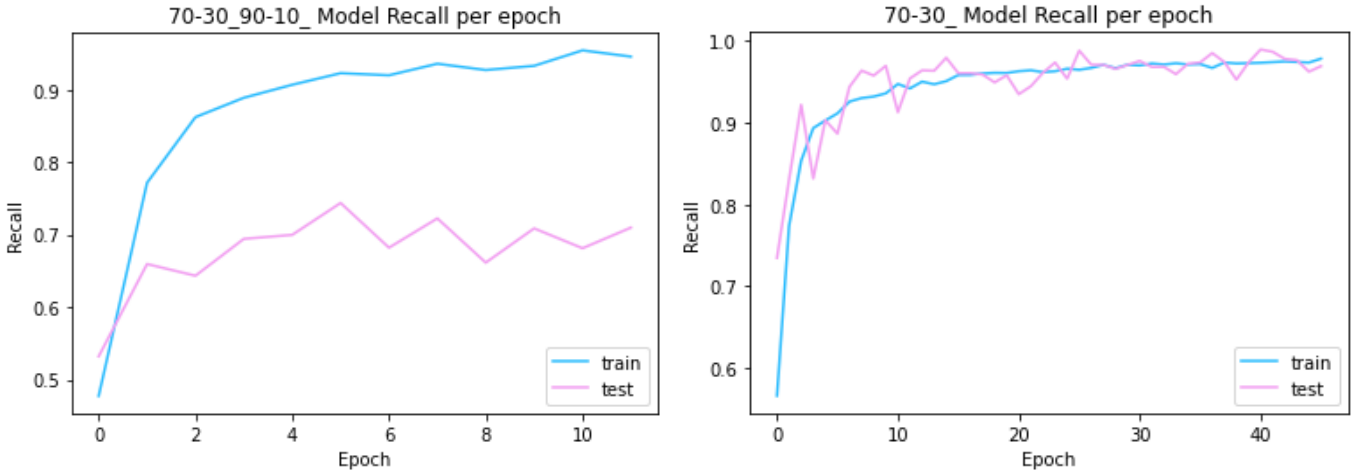


Figura 66: Comparación de la sensibilidad entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Resultado de la clasificación

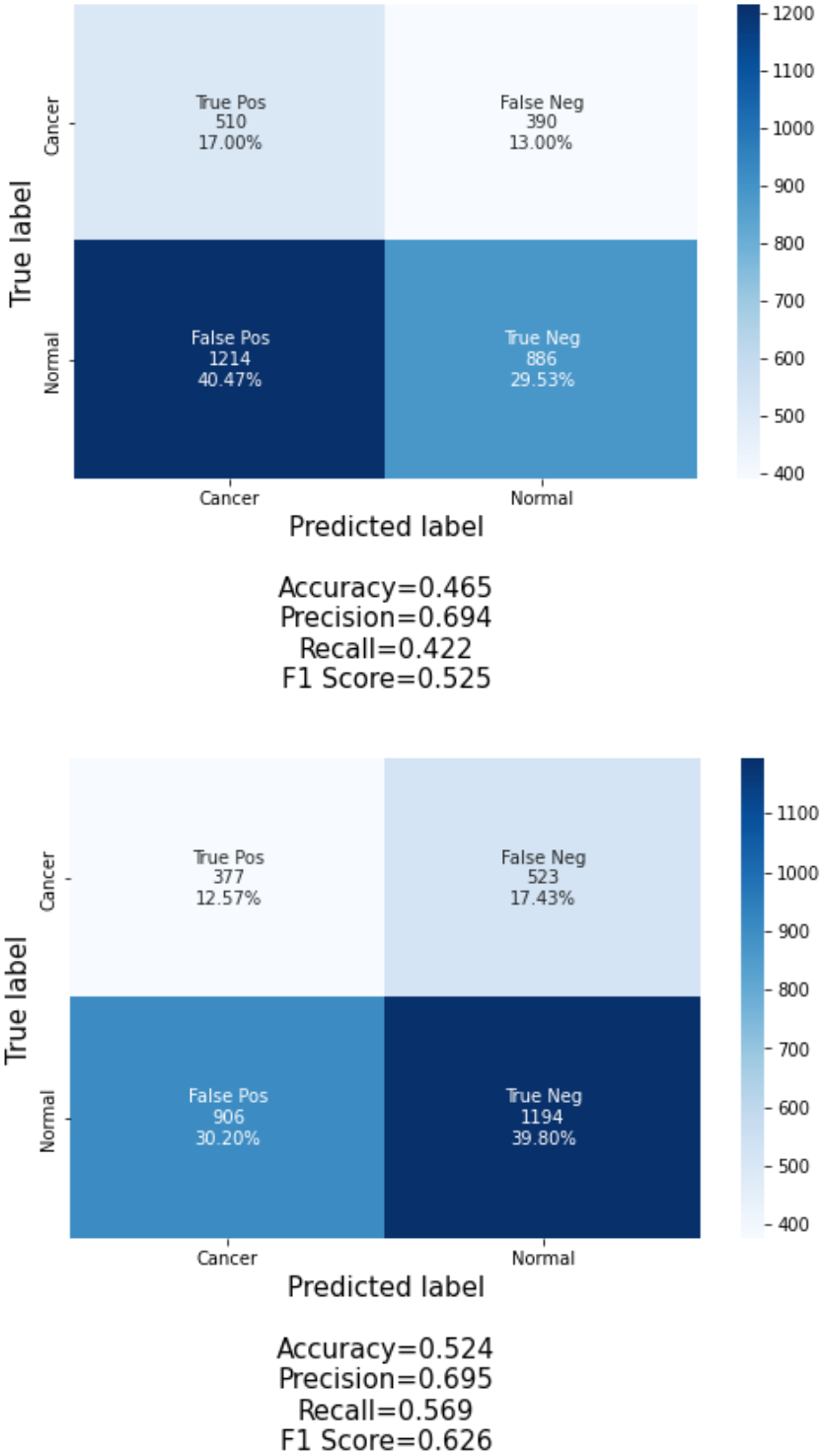


Figura 67: Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (arriba) y el entrenado con un dataset 50% -50% puramente sintético (abajo). (Imagen propia)

Los resultados de este modelo son peores que los que se obtuvieron con el modelo entrenado solamente con imágenes sintéticas. Esta conclusión es lógica, ya que se trata de un modelo entrenado con un dataset desequilibrado, en el que la clase más numerosa (70% cáncer), es a su vez la clase menos numerosa del subconjunto de test (30% cáncer).

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El objetivo de este proyecto era diseñar una herramienta de software barata, eficiente e intuitiva. Esta herramienta debía ser capaz de generar imágenes sintéticas de mamografías de buena calidad, para evitar así el problema de depender de bases de datos públicas, en su mayoría bien custodiadas por las autoridades sanitarias de la sociedad médica internacional, debido al contenido de información sensible de las pacientes.

Creemos firmemente, apoyados en las pruebas aportadas en los capítulos 5 y 6, que nuestra herramienta cumple parte de dichas expectativas: se trata de un software barato de entrenar debido al alojamiento en remoto de Google Colab PRO, es eficiente en cuanto a que consigue generar datasets formados por numerosas mamografías sintéticas en relativo poco tiempo (en el orden minutos, ni siquiera horas), y es intuitiva, el notebook en el que se implementa está comentado línea a línea, tanto en inglés como en español, dos de los idiomas más hablados del mundo.

En cuanto a la calidad de las imágenes, concluimos que no es suficiente para poblar datasets de manera que puedan sustituir a datasets de imágenes reales. La carencia de imágenes en el dataset DDSM elegido para entrenar la herramienta generadora y la falta de potencia de cálculo y memoria en el entorno remoto han supuesto un hándicap demasiado grande para que los resultados presenten una calidad aceptable. Sin embargo, creemos en que una herramienta basada en la que aquí se presenta, entrenada a partir de un dataset más numeroso y de mejor calidad, con herramientas más potentes, podría alcanzar métricas mucho mejores que las conseguidas en nuestro desarrollo.

7.2 Líneas de trabajo futuro

En cuanto a las líneas de futura investigación, como hemos señalado, tuvimos que dejar varios apartados de lado para conseguir mejores resultados con los medios de los que disponíamos. Si bien es verdad que el objetivo era demostrar que con una pequeña base de datos como DDSM con solo alrededor de 6300 imágenes, se podía desarrollar una herramienta lo suficiente potente, usando tecnologías de Aprendizaje Automático y Profundo como las DCGAN, en concreto, la StyleGAN 2 ADA, existen otros datasets (bajo licencia de investigación, obviamente) más numerosos y de mayor calidad. Uno de estos ejemplos es el dataset **DukeDBT** (Mateusz, y otros, 2020), un dataset de tomografías digitales (de mucha más calidad que las mamografías), con aproximadamente 1500 GB de imágenes anotadas. Un dataset como este necesitaría de una potencia de cálculo extremadamente grande para crear una herramienta como la propuesta, por lo que una futura línea de

investigación sería usar datasets más grandes y de mayor calidad para el desarrollo de modelos de generación de mamografías/tomografías sintéticas que arrojen mejores resultados que los aquí propuestos.

Otro de los puntos en los que se podría profundizar es en el uso de otros modelos generadores más actuales, de hecho, los investigadores de nVIDIA que desarrollaron y publicaron StyleGAN acaban de publicar un nuevo modelo de generación de imágenes sintéticas mucho más potente que StyleGAN, **Alias FreeGAN** (Karras, y otros, 2021), que resuelve ciertos artefactos resultado de carencias en el entrenamiento del generador de StyleGAN.

Para concluir, una de las ambiciones que nos planteamos en el inicio del proyecto fue la de generar mamografías sintéticas de gran calidad y, a la vez, mantener una resolución elevada de dichas imágenes. La resolución de las imágenes generadas por este tipo de modelos suele ser relativamente pequeña, en torno a 128x128px o incluso 256x256px. El objetivo inicial era llegar a una resolución de 1024x1024px, incluso llegamos a entrenar el modelo para que generara dichas imágenes, pero no cumplían con la calidad adecuada. En este tipo de redes, existe un compromiso entre la resolución y la calidad de las imágenes generadas, ya que el generador de las DCGAN no puede “atender” a ambas especificaciones. Aun así, conseguimos generar imágenes de 512x512 con calidad puntuada con cerca de un 39 de FID, un hito bastante relevante. Por lo tanto, la línea de investigación que de este punto se propone, es la generación de imágenes de mayor resolución, manteniendo la misma calidad que la conseguida en este proyecto, bien haciendo uso de un entorno con mayor potencia de cálculo y memoria, o bien mediante el uso de los nuevos e innovadores modelos de DCGAN que se publicarán en los próximos años sin ningún atisbo de duda.

8 Bibliografía

Alyafi, B., Diaz, O., & Martí, R. (04 de 09 de 2019). *DCGANs for Realistic Breast Mass Augmentation in X-ray Mammography.* Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1909.02062

Cha, K. H., Petrick, N., Pezeshk, A., Graff, C. G., Sharma, D., Badal, A., & Sahiner, B. (22 de 11 de 2019). *Evaluation of data augmentation via synthetic images for improved breast mass detection on mammograms using deep learning.* Recuperado el 05 de 05 de 2021, de https://www.ncbi.nlm.nih.gov/: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6872953/

Cheddad, A. (s.f.). *The Complete Mini-DDSM.* Obtenido de kaggle.com: https://www.kaggle.com/cheddad/miniddsm2

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition.*

Dissanayake Lekamlage, C., Afzal, F., Westerberg, E., & Cheddad, A. (2020). Mini-DDSM: Mammography-based Automatic Age Estimation. *DMIP '20: 2020 3rd International Conference on Digital Medicine and Image Processing*, (págs. 1-6).

Goodfellow, I. J., Pouget, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Networks. *Cornell University*, 9. Obtenido de https://arxiv.org/abs/1406.2661

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (12 de 01 de 2018). *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/1706.08500

Isola, P., Zhu, J.-Y., Zhu, T., & Efros, A. A. (2016). *Image-to-Image Translation with Conditional Adversarial Networks.* Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1611.07004

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (27 de 10 de 2018). *Progressive Growing of GANs for Improved Quality, Stability, and Variation.* Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1710.10196

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (11 de 06 de 2020). *Training Generative Adversarial Networks with Limited Data.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/2006.06676

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., & Aila, T. (23 de Junio de 2021). *Alias-Free Generative Adversarial Networks*. Obtenido de arXiv: https://arxiv.org/abs/2106.12423

Karras, T., Laine, S., & Aila, T. (12 de 12 de 2018). *A Style-Based Generator Architecture for Generative Adversarial Networks.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/pdf/1812.04948.pdf

Karras, T., Laine, S., Aittala, M., & Hellsten, J. (3 de 12 de 2019). *Analyzing and Improving the Image Quality of StyleGAN.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/pdf/1912.04958.pdf

Krishna Goti, S., & Ma, J. (14 de Agosto de 2018). *Text-to-Image-to-Text Translation using Cycle Consistent Adversarial Networks.* Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1808.04538

Mateusz, B., Saha, A., Walsh, R., Ghate, S., Li, N., Święcicki, A., . . . A. Mazurowski, M. (13 de Noviembre de 2020). *Detection of masses and architectural distortions in digital breast tomosynthesis: a publicly available dataset of 5,060 patients and a deep learning model*. Obtenido de arxiv.org: https://arxiv.org/abs/2011.07995

McDermott, J. (s.f.). *Hands-on Transfer Learning with Keras and the VGG16 Model*. Obtenido de https://www.learndatasci.com: https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/

Payne, C. (25 de 04 de 2019). *MuseNet.* Recuperado el 05 de 05 de 2021, de https://www.openai.com/: https://www.openai.com/blog/musenet/

Ponti, M., Sampaio Ferraz Ribeiro, L., Santana Nazare, T., & Bui, T. (2017). Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask. *ResearchGate*.

Radford, A., Metz, L., & Chintala, S. (19 de 11 de 2015). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* Recuperado el 05 de 05 de 2021, de https://arxiv.org: https://arxiv.org/abs/1511.06434

Radford, A., Metz, L., & Chintala, S. (19 de 11 de 2015). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/1511.06434

Schultz, D. (s.f.). *dataset-tools.* Obtenido de GitHub: https://github.com/dvschultz

Shen, L., Margolies, L. R., Rothstein, J. H., Fluder, E., McBride, R. B., & Sieh, W. (31 de 12 de 2018). *Deep Learning to Improve Breast Cancer Early Detection on Screening Mammography.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/1708.09427

Simonyan, K., & Zisserman, A. (2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. *iCLR 2015.*

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2 de Diciembre de 2015). Rethinking the Inception Architecture for Computer Vision. Obtenido de arXiv.org: https://arxiv.org/abs/1512.00567

Trimarchi, D. (21 de Noviembre de 2019). *confusion_matrix*. Obtenido de GitHub: https://github.com/DTrimarchi10/confusion_matrix

Wu, E., Wu, K., Cox, D., & Lotter, W. (21 de 07 de 2018). *Conditional Infilling GANs for Data Augmentation in Mammogram Classification.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/pdf/1807.08093.pdf

Xu, Z., Wilber, M., Fang, C., Hertzmann, A., & Jin, H. (2019). *Learning from Multi-domain Artistic Images for Arbitrary Style Transfer.* Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/pdf/1805.09987.pdf

Yuan, C., Huang, Y.-C., & Tsai, C.-H. (22 de 06 de 2020). *Efficient text generation of user-defined topic using generative adversarial networks.* Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/2006.12005

9 Anexos

Anexo I. Generator.ipynb

Contenido del cuaderno en el que se implementa la herramienta:

1. (OPCIONAL) Activar DRIVE para acceder a los archivos de su Drive

---

1. (OPTIONAL) Mount Google Drive access

En primer lugar, se puede conectar esta sesión de ejecución remota a su Google Drive personal, para acceder a los archivos pertinentes si se los ha descargado previamente. El usuario solo

tiene que ejecutar la siguiente celda, ir a la ventana emergente, iniciar sesión en su cuenta de Google, copiar el enlace que se ha generado y copiarlo en la celda.

First of all, this remote Google Colab environment can be connected to a personal Google Drive account, granting access to uploaded files if the user downloaded them. If needed, the user would execute the cell below this text, go to the pop window, log in to his Google Drive account, copy the generated link and paste it on the white box in the cell, pressing ENTER afterwards.

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

2.Clonar y acceder a las carpetas del repositorio

---

2. Clone and access repository folders

En segundo lugar, necesitamos acceder a las herramientas del repositorio de StyleGAN 2 ADA, por lo tanto, clonamos todos los archivos y carpetas del repositorio en nuestros archivos

temporales de google colab.

NOTA: Si se quisiera clonar el repositorio en archivos de su Google Drive personal, cambiar las rutas para que se clonen los archivos donde desee.

We need to access the StyleGAN 2 ADA Torch folders, files and tools. In order to do that, we clone the repository in our Google Colab temporary files.

WARNING: If the user would like to clone the repository into his Google Drive account, change the paths to where he/she wants to place the cloned repository"

```
# Clonamos el repositorio / Clone the repository

# Si ya habíamos clonado el repositorio, solamente accedemos a él / If the repository was
import os
if os.path.isdir("/content/colab-sg2-ada-torch"):
    %cd "/content/colab-sg2-ada-torch/stylegan2-ada-pytorch"

# Si es la primera vez que lo clonamos,
# creamos una carpeta en los archivos temporales de Google Colab y lo clonamos allí

# If this is the first time, we make a new directory
# in the temporary files and clone the repository there.

else:
    %cd "/content"
    !mkdir colab-sg2-ada-torch
    %cd colab-sg2-ada-torch
    !git clone https://github.com/NVlabs/stylegan2-ada-pytorch
    %cd stylegan2-ada-pytorch
    !mkdir downloads
    !mkdir datasets

     /content
     /content/colab-sg2-ada-torch
     Cloning into 'stylegan2-ada-pytorch'...
     remote: Enumerating objects: 125, done.
     remote: Total 125 (delta 0), reused 0 (delta 0), pack-reused 125
     Receiving objects: 100% (125/125), 1.12 MiB | 14.16 MiB/s, done.
     Resolving deltas: 100% (55/55), done.
     /content/colab-sg2-ada-torch/stylegan2-ada-pytorch
```

Instalamos las librerías de las que depende la versión del repositorio que hemos clonado. Este es un paso **esencial**, si no se ejecuta la celda de abajo, aparecerán errores en las ejecuciones.

Let's proceed to install all the libraries StyleGAN 2 ADA Torch depends on. This is a **critical** step, if this cell is no executed, error alerts will raise later on.

```
# Instalamos las dependencias / Installing dependencies
!pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 torchaudio==0.7.2 -f https://down
!pip install click requests tqdm pyspng ninja imageio-ffmpeg==0.4.3
```

3. Subimos los modelos ya entrenados

---

3. Upload pre-trained models

En las dos siguientes celdas, vamos a subir DOS ARCHIVOS:

1. El modelo generador de imágenes con cáncer, "cancer_generator.pkl"
2. El modelo generador de imágenes normales, "normal_generator.pkl"

Este paso solo es necesario si el usuario NO tiene los modelos en Google Drive o no quere conectar su cuenta. Tendremos que ejecutar las dos celdas consecutivamente, para que se guarden los dos modelos en los archivos temporales de Google Colab. Esta ejecución puede llegar a tardar entre 10 y 40 minutos.

In the next two cells, we are going to upload TWO FILES: Cancer generator model, "cancer_generator.pkl"

1. Normal generator model, "normal_generator.pkl"

This step is only needed in case the user DO NOT have the model files in their Google Drive or he/she does not want to connect his/her account. We will have to execute the 2 cells, one right after the other, so that the 2 models will be saved in the temporary Colab files. This execution can last anything between 10 and 40 minutes.

```
# Subimos el modelo generador de cáncer / Upload the cancer generator model

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

```
# Subimos el modelo generador de normales / Upload the normal generator model

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

4. Generamos las imágenes

---

4. Image Generation

Las siguientes celdas son casi idénticas, su función es la de generar las imágenes con cáncer y normales para formar un dataset.

Tienen 4 argumentos:

1. **out_dir**: La carpeta donde queremos que se guarden las imágenes. Por defecto, el notebook crea 3 carpetas dentro de los archivos locales de Colab: la primera es generated y, dentro de ella, cancer y normal. Si se quisieran guardar en una carpeta en Drive, se deberá activar drive al principio del notebook y sobreescribir las rutas donde se desean
   guardar las imágenes.

2. **trunc**: Si existen muestras con poco peso en el dataset de entrenamiento, puede que algunas de las imágenes generadas presenten artefactos como saturaciones excesivas, deformaciones en las siluetas, etc. Para evitar dichos artefactos, StyleGAN implementa
   este argumento, que "trunca" o fuerza el valor del vector latente introducido (la seed) para que sea de un valor más cercano al de la media. En resumen, este valor solo ha de modificarse si se requiere más diversidad entre las fotos a generar, pero el usuario ha de ser consciente de que pueden aparecer artefactos y errores en dichas imágenes. **Por defecto, el valor es de 1 (sin truncamiento) y se puede variar hasta llegar al 0 (máximo**

   truncamiento).

3. **seeds**: Este argumento se usa para especificar el número de imágenes que queremos generar. Debe seguir el formato X-Y, siendo X la primera de dichas "semillas", que el

generador convierte en imágenes e Y, la última. **Por defecto, está definido como 0-10000, es decir, generará 10000 imágenes.**

4. **network**: Este argumento es la ruta al archivo .pkl que contiene el modelo para generar imágenes. Si el usuario ha activado su drive y tiene el archivo allí, solo tiene que copiar y pegar la ruta **SIN CORCHETES, por ejemplo: network=/content/drive/MyDrive/models/cancer.pkl**. Si no tiene drive activado, asumimos que ha subido el modelo con las celdas anteriores, por lo que se encontrará en los archivos temporales de google, "/content/".

The next to cells are almost the same, their function is to generate cancer and normal images to build a dataset. They have 4 arguments:

1. **out_dir**: The folder where we want our images to be saved. By default, the notebook makes 3 folders inside Colab temporary files: the first is generated, and inside it, cancer and normal. If the user would like to save the images in a Google Drive's folder, he/she will have to overwrite the path to the desired Drive folder.
2. **trunc**: If there is underrepresented data in the training samples, some generated images may present several artifacts such as excessive saturation, siloutte deformation, incorrect color, etc. To avoid these artifacts, StyleGAN implements uses a “truncation trick” by
truncating the intermediate latent vector w forcing it to be close to average. Summing up, this value should only be modified if the it is required more diversity in the generated images, but the user should be conscient that these artifacts may occur. **By default, the value starts from 1 (full truncation) to 0 (0 truncation)**
3. **seeds**: This argument is used to specify the number of images the user wants to generate. The argument should have a X-Y format, where X is the first of the so called "seeds", which the generator transforms to a latent vector space (w), and further on, to images and Y be
the last of them. By default, it is defined as 0-9999, which means it will generate 10000 images.
4. **network**:It is the path to the .pkl file which contains the model to generate images. if the user has his/her Google Account connected and the file inside it, he/she just has to overwrite the path **WITHOUT QUOTES, as in: network=/content/drive/MyDrive/models/cancer.pkl--**. If Google Drive is not connected, we assume that the file has been uploaded with the previous cells, and it can be found in

4.1 CANCER

```
# Comprueba si existen las carpetas generated y cancer y accede a ellas si existen.
# Si no, las crea y accede a ellas

#Checks if generated and cancer folders exists, and tries to navigates to them. If not,
# it will make them and navigate then.
if os.path.isdir("/content/generated/cancer"):
    %cd "/content/generated/cancer"
else:
    %cd "/content"
    !mkdir generated
    %cd generated
    !mkdir cancer
    %cd cancer

# Volvemos a donde clonamos el repositorio para acceder al archivo generate.py
# Back to where we clonned the repository to access the generate.py file

if os.path.isdir("/content/colab-sg2-ada-torch"):
    %cd "/content/colab-sg2-ada-torch/stylegan2-ada-pytorch"

out_dir='/content/drive/MyDrive/TFM/Generated/30-70/Test/Cancer'

trunc = 1.0

seeds='0-2159'

network='/content/drive/MyDrive/TFM/MODELOS/cancer_generator.pkl' # Sobreescribir aquí la
                                        # Overwrite here the path to the cancer generator

# No modificar esta línea
# Do not modify this line
!python generate.py --outdir={out_dir} --trunc={trunc} --seeds={seeds}\
      --network={network}
```

4.2 NORMALES

```
if os.path.isdir("/content/generated/normal"):
    %cd "/content/generated/normal"
else:
    %cd "/content"
    !mkdir generated
    %cd generated
    !mkdir normal
    %cd normal

if os.path.isdir("/content/colab-sg2-ada-torch"):
```

```
    %cd "/content/colab-sg2-ada-torch/stylegan2-ada-pytorch"

out_dir='/content/drive/MyDrive/TFM/Generated/70-30/Train/Normal'

trunc = 1.0

seeds='0-5039'

network='/content/drive/MyDrive/TFM/MODELOS/normal_generator.pkl'

!python generate.py --outdir={out_dir} --trunc={trunc} --seeds={seeds}\
      --network={network}
```

5. Compresión y descarga del dataset

Finalmente, se comprime en un .zip las dos carpetas y se descargará automáticamente cuando termine.

Finally, the user will compress in a .zip file the cancer and normal folders, and, as soon as it finishes compressing, the file will be available to download.

```
from google.colab import files

 ruta_generated = '/content/generated/cancer' # Pegar entre corchetes la ruta de la carpet
                                              # Overwrite the path to folders Cancer and N

!zip -r /content/dataset.zip {ruta_generated}

files.download("/content/dataset.zip")
```

```
     updating: content/generated/cancer/ (stored 0%)
```

Anexo II. Arquitectura del modelo original preentrenado de VGG16:

Model: "vgg16"

_____

Layer (type) Output Shape Param #

==========

input_1 (InputLayer) [(None, 224, 224, 3)] 0

_____

block1_conv1 (Conv2D) (None, 224, 224, 64) 1792

_____

block1_conv2 (Conv2D) (None, 224, 224, 64) 36928

_____

block1_pool (MaxPooling2D) (None, 112, 112, 64) 0

_____

block2_conv1 (Conv2D) (None, 112, 112, 128) 73856

_____

block2_conv2 (Conv2D) (None, 112, 112, 128) 147584

_____

block2_pool (MaxPooling2D) (None, 56, 56, 128) 0

_____

block3_conv1 (Conv2D) (None, 56, 56, 256) 295168

---

block3_conv2 (Conv2D) (None, 56, 56, 256) 590080

---

block3_conv3 (Conv2D) (None, 56, 56, 256) 590080

---

block3_pool (MaxPooling2D) (None, 28, 28, 256) 0

---

block4_conv1 (Conv2D) (None, 28, 28, 512) 1180160

---

block4_conv2 (Conv2D) (None, 28, 28, 512) 2359808

---

block4_conv3 (Conv2D) (None, 28, 28, 512) 2359808

---

block4_pool (MaxPooling2D) (None, 14, 14, 512) 0

---

block5_conv1 (Conv2D) (None, 14, 14, 512) 2359808

---

block5_conv2 (Conv2D) (None, 14, 14, 512) 2359808

---

block5_conv3 (Conv2D) (None, 14, 14, 512) 2359808

---

block5_pool (MaxPooling2D) (None, 7, 7, 512) 0

flatten (Flatten) (None, 25088) 0

fc1 (Dense) (None, 4096) 102764544

fc2 (Dense) (None, 4096) 16781312

predictions (Dense) (None, 1000) 4097000

Total params: 138,357,544

Trainable params: 138,357,544

Non-trainable params: 0

Anexo III. Resultado del entrenamiento para la configuración 30% Cáncer - 70% Normal

Column1	Column2	Column3	Column4	Column5
EPOCHS	**loss**	**accuracy**	**precision**	**recall**
0	1.2028107643127441	0.7442241907119751	0.8040995597839355	0.8390710949897766
1	0.25259488821029663	0.8872646689414978	0.9065309762954712	0.9352701306343079
2	0.17085912823677063	0.9273388385772705	0.9436889290809631	0.9531949758529663
3	0.13720354437828064	0.9454506635665894	0.9583459496498108	0.9640968441963196
4	0.13003167510032654	0.946805477142334	0.9578161239624023	0.9665953516960144
5	0.10675837099552155	0.9582144618034363	0.9685342907905579	0.9718883633613586
6	0.10043293237686157	0.9598545432090759	0.9691714644432068	0.973614513874054
7	0.08968164026737213	0.965416431427002	0.9715122580528259	0.979327917098999
8	0.08594799041748047	0.9659869074821472	0.9731509685516357	0.9784048199653625
9	0.08157157152891159	0.9679121375083923	0.97639399766922	0.9777868390083313
10	0.06909247487783432	0.9734740257263184	0.9784390926361084	0.9838168025016785
11	0.066069096326828	0.9735453724861145	0.9800610542297363	0.9821592569351196
12	0.06967391818761826	0.9744723439216614	0.9795131683349609	0.984104335308075

13	0.06672228872776031	0.9738305807113647	0.9789068102836609	0.9837953448295593
14	0.05760721117258072	0.9774671792984009	0.9826290011405945	0.9852312207221985
15	0.06283784657716751	0.9753280282020569	0.9807263016700745	0.9841205477714539
16	0.05518813803792	0.9778950214385986	0.9834282398223877	0.9850305318832397
17	0.055789515376091	0.9778237342834473	0.9829164147377014	0.9854215383529663
18	0.04949035868048668	0.9819595217704773	0.985480785369873	0.9887937903404236
19	0.054120659828186035	0.978251576423645	0.9821609854698181	0.9868621826171875
20	0.051009777933359146	0.979962944984436	0.9844385385513306	0.9869480729103088
21	0.045445580035448074	0.9818168878555298	0.9862581491470337	0.9877663254737854
22	0.050282109528779984	0.981389045715332	0.9855543971061707	0.9878658056259155
23	0.0484011285007	0.9808185696601868	0.9848654270172119	0.9877750873565674
24	0.0496407188475132	0.9808185696601868	0.9854453206062317	0.9871533513069153
25	0.04609597101807594	0.9825283885002136	0.9861518144607544	0.9889508485794067
26	0.04711073264479637	0.9827438592910767	0.9859926700592041	0.9894071817398071
27	0.042427562177181244	0.9830290675163269	0.9869733452796936	0.988784670829773

Column1	Column6	Column7	Column8	Column9
EPOCHS	**val_loss**	**val_accuracy**	**val_precision**	**val_recall**
0	0.3167076110839844	0.84375	0.837411642074585	0.9618988037109375
1	0.17900824546813965	0.9318576455116272	0.9448866844177246	0.9583851099014282
2	0.14473220705986023	0.9383680820465088	0.9415310621261597	0.9719974994659424
3	0.1124815121293068	0.9565972089767456	0.9637592434883118	0.9745341539382935
4	0.10575844347476959	0.9596354365348816	0.9550629258155823	0.9888337254524231
5	0.10280372947454453	0.9565972089767456	0.9498820900917053	0.990774929523468
6	0.0840398371219635	0.9635416865348816	0.9791929125785828	0.9682044982910156
7	0.08940721303224564	0.9665798544883728	0.9842569231987	0.9678018689155579
8	0.07226690649986267	0.97265625	0.9782743453979492	0.9825436472892761
9	0.08349280804395676	0.9683159589767456	0.966585636138916	0.9888129234313965
10	0.06965694576501846	0.9735243320465088	0.9832713603973389	0.9790253043174744
11	0.05973927304148674	0.9752604365348816	0.9785670638084412	0.9864197373390198
12	0.07052011787891388	0.9748263955116272	0.9832713603973389	0.9808405637741089
13	0.048245880752801895	0.9813368320465088	0.9839704036712646	0.9894606471061707
14	0.059412069618701935	0.9813368320465088	0.9936668872833252	0.9794007539749146
15	0.07021404802799225	0.9691840410232544	0.9849246144294739	0.9708978533744812
16	0.05421740561723709	0.9782986044883728	0.9881250262260437	0.9807692170143127
17	0.037136469036340714	0.9830729365348816	0.9839802980422974	0.9919254779815674
18	0.053677573800086975	0.9782986044883728	0.9815270900726318	0.9876084327697754
19	0.04166172072291374	0.9865451455116272	0.9912554621696472	0.9894015192985535
20	0.05127456411719322	0.9787326455116272	0.9839605093002319	0.9857849478721619
21	0.04405360668897629	0.984375	0.9858895540237427	0.9919753074645996
22	0.049776799976825714	0.9830729365348816	0.9901356101036072	0.9858809113502502
23	0.06097492203116417	0.9769965410232544	0.9790897965431213	0.9882060885429382
24	0.0391353964805603	0.9856770634651184	0.9919554591178894	0.9876771569252014
25	0.04004605859518051	0.9835069179534912	0.9851393103599548	0.991277277469635
26	0.05327922850847244	0.9809027910232544	0.9924717545509338	0.9801734685897827
27	0.04952773451805115	0.9787326455116272	0.9767156839370728	0.9931464195251465

Anexo IV. Resultado del entrenamiento para la configuración 50% Cáncer - 50% Normal

Column1	Column2	Column3	Column4	Column5
EPOCHS	loss	accuracy	precision	recall
0	1.5082064867019653	0.705198347568512	0.712565541267395	0.6880043745040894
1	0.2650442123413086	0.8821477293968201	0.8892132043838501	0.8735491037368774
2	0.18562547862529755	0.9207934141159058	0.9262647032737732	0.9143521785736084
3	0.14661526679992676	0.9393981099128723	0.9429594874382019	0.9354838728904724
4	0.13166172802448273	0.9469220042228699	0.9524270296096802	0.940975546836853
5	0.11401920020580292	0.9544459581375122	0.9562474489212036	0.9525891542434692
6	0.11220522224903107	0.9556087255477905	0.9569272994995117	0.9541786313056946
7	0.09141235053539276	0.9628590941429138	0.965702474117279	0.9597535729408264
8	0.09521288424730301	0.9612858891487122	0.9633241891860962	0.959108293056488
9	0.08122596144676208	0.9681258797645569	0.9686344265937805	0.9675742387771606
10	0.07553156465291977	0.9694938659667969	0.970161497592926	0.9688354134559631
11	0.08108275383710861	0.968673050403595	0.9709805846214294	0.9661967754364014
12	0.07526513189077377	0.9694254398345947	0.9707256555557251	0.9679320454597473
13	0.06692898273468018	0.9731190204620361	0.9735941886901855	0.972662627696991
14	0.06427162885665894	0.9751709699630737	0.9758672714233398	0.974397599697113
15	0.06553623825311661	0.9750341773033142	0.9764221906661987	0.9736194610595703
16	0.062380701303482056	0.975307822227478	0.9759813547134399	0.9745100736618042
17	0.05989055335521698	0.9764021635055542	0.9778663516044617	0.9747841358184814
18	0.06364139914512634	0.9762653708457947	0.976827085018158	0.9756231307983398
19	0.0636594370007515	0.9752393960952759	0.974972665309906	0.9755063056945801
20	0.05288764834403992	0.979685366153717	0.9810595512390137	0.9782400727272034
21	0.062382061034440994	0.9756497740745544	0.9738562107086182	0.9775834083557129
22	0.05768873915076256	0.9774281978607178	0.9792553782463074	0.9755029678344727
23	0.05143975839018822	0.9798221588134766	0.9782164692878723	0.9815573692321777
24	0.04840845242142677	0.9807113409042358	0.9811010956764221	0.9802955389022827
25	0.04888655245304108	0.9814637303352356	0.9817907810211182	0.9811191558837891
26	0.0500117726624012	0.9818057417869568	0.9820989370346069	0.9815624356269836
27	0.04728206619620323	0.9818741679191589	0.9824561476707458	0.9812456965446472
28	0.04982874542474747	0.9813269376754761	0.9819103479385376	0.9807007908821106
29	0.04704497754573822	0.9815321564674377	0.9808716773986816	0.9822137355804443
30	0.045898791402578354	0.9834473133087158	0.9847862124443054	0.9820940494537354
31	0.04270761460065842	0.9848152995109558	0.9851011633872986	0.9845628142356873
32	0.047136127948760986	0.9830369353294373	0.9840899705886841	0.9819351434707642
33	0.04287680983543396	0.9841997027397156	0.9844006299972534	0.9839966893196106
34	0.04925345629453659	0.9816005229949951	0.9817858338356018	0.9813826084136963
35	0.046856336295604706	0.9826265573501587	0.9827609658241272	0.9824921488761902
36	0.039322882890701294	0.984883725643158	0.9851729869842529	0.9844971895217896
37	0.043869782239198685	0.9831737279891968	0.9836964011192322	0.9826194047927856
38	0.0425768680870533	0.9833789467811584	0.983974814414978	0.9827633500099182
39	0.038159530609846115	0.985157310962677	0.9846952557563782	0.9856380820274353

40	0.03590542823076248	0.9869356751441956	0.9882224202156067	0.9856576919555664
41	0.038678254932165146	0.9853625297546387	0.9857534170150757	0.9849438667297363
42	0.03948182985186577	0.984883725643158	0.9844411015510559	0.9853824973106384
43	0.03803811967372894	0.9853625297546387	0.9860312342643738	0.9846827387809753
44	0.0401129387319088	0.9830369353294373	0.9834269285202026	0.9826194047927856
45	0.036550622433423996	0.9868673086166382	0.9863406419754028	0.9874196648597717
46	0.03965109586715698	0.9843365550041199	0.9855769276618958	0.9830113649368286
47	0.036584991961717606	0.9857045412063599	0.9856479167938232	0.9857826232910156
48	0.03842833265662193	0.9852257370948792	0.9867104887962341	0.9837453961372375
EPOCHS	**val_loss**	**val_accuracy**	**val_precision**	**val_recall**
0	0.30130478739738464	0.8734375238418579	0.879365086555481	0.8656250238418579
1	0.18861886858940125	0.921875	0.9390048384666443	0.9041731357574463
2	0.17464764416217804	0.927734375	0.9667519330978394	0.885937511920929
3	0.11467721313238144	0.9535156488418579	0.9645447134971619	0.9410377144813538
4	0.14310070872306824	0.942187488079071	0.9143484830856323	0.975781261920929
5	0.10631169378757477	0.956250011920929	0.9491525292396545	0.964006245136261
6	0.0959630087018013	0.961718738079071	0.9451127648353577	0.9804992079734802
7	0.0997471958398819	0.9664062261581421	0.9760000109672546	0.9561128616333008
8	0.08410537987947464	0.96484375	0.9663273096084595	0.9633099436759949
9	0.0777372494339943	0.971484363079071	0.9816879034042358	0.9610288143157959
10	0.05801273137331009	0.979687511920929	0.9850393533706665	0.9742990732192993
11	0.06943349540233612	0.9742187261581421	0.9809825420379639	0.9671875238418579
12	0.06907907873392105	0.9730468988418579	0.9848121404647827	0.9609984159469604
13	0.051436811685562134	0.978515625	0.9788401126861572	0.9780735969543457
14	0.05786743760108948	0.9781249761581421	0.9880383014678955	0.967968761920929
15	0.053844206035137177	0.98046875	0.9775888919830322	0.9836702942848206
16	0.05337515473365784	0.978515625	0.9678407311439514	0.9898198843002319
17	0.05121346563100815	0.977343738079071	0.9788732528686523	0.9758190512657166
18	0.06801334023475647	0.971875011920929	0.9567198157310486	0.9882352948188782
19	0.044385284185409546	0.9828125238418579	0.9842891097068787	0.981205940246582
20	0.06065789982676506	0.978515625	0.9896414279937744	0.9672897458076477
21	0.0678732693195343	0.973828136920929	0.9856345057487488	0.9618380069732666
22	0.053086988627910614	0.9761718511581421	0.9728682041168213	0.9797033667564392
23	0.048105232417583466	0.9808593988418579	0.9920191764831543	0.9695788025856018
24	0.04757658392190933	0.9828125238418579	0.985051155090332	0.980422854423523
25	0.05360528081655502	0.981249988079071	0.9782100915908813	0.9843382835388184
26	0.05180699750781059	0.981249988079071	0.9745566844940186	0.9882720708847046
27	0.05907747894525528	0.974609375	0.9668720960617065	0.9827721118927002
28	0.04924872890114784	0.983203113079071	0.9866561889648438	0.9797350168228149
29	0.04210422560572624	0.9859374761581421	0.996802568435669	0.9749804735183716
30	0.035702697932720184	0.987109363079071	0.9815526604652405	0.9930015802383423
31	0.032002076506614685	0.9859374761581421	0.9799692034721375	0.9921996593475342
32	0.04901937395334244	0.985546886920929	0.9889415502548218	0.9819607734680176
33	0.033982791006565094	0.987109363079071	0.9829853177070618	0.9914196729660034
34	0.04550335928797722	0.9828125238418579	0.9783783555030823	0.9875292181968689
35	0.0433981791138649	0.9847656488418579	0.9881516695022583	0.981176495552063

36	0.03740561753511429	0.9867187738418579	0.9859374761581421	0.9874804615974426
37	0.05049677938222885	0.979687511920929	0.9857482314109802	0.9734167456626892
38	0.031220104545354843	0.9878906011581421	0.9890795350074768	0.9867704510688782
39	0.03503170236945152	0.985546886920929	0.995207667350769	0.9757243394851685
40	0.03598872944712639	0.985156238079071	0.9920823574066162	0.9781420826911926
41	0.03962651640176773	0.9878906011581421	0.9929078221321106	0.9828392863273621
42	0.05224153399467468	0.981640636920929	0.9782270789146423	0.9851213693618774
43	0.03734110668301582	0.987109363079071	0.992057204246521	0.981918215751648
44	0.04927246272563934	0.9769531488418579	0.9655700325965881	0.9890282154083252
45	0.0334303192794323	0.9859374761581421	0.9920823574066162	0.9796715974807739
46	0.03599612042307854	0.986328125	0.9798761606216431	0.9929412007331848
47	0.034987278282642365	0.987500011920929	0.9913249015808105	0.9835680723190308
48	0.04726126417517662	0.981640636920929	0.9782945513725281	0.9851678609848022

Anexo V. Resultado del entrenamiento para la configuración 70% Cáncer - 30% Normal

Column1	Column2	Column3	Column4	Column5
EPOCH	**loss**	**accuracy**	**precision**	**recall**
0	1.0413426160812378	0.7746006846427917	0.6410118341445923	0.5659301280975342
1	0.24153955280780792	0.9022390246391296	0.8846153616905212	0.7738010287284851
2	0.16925746202468872	0.9306902289390564	0.9102175235748291	0.8536527752876282
3	0.1322411298751831	0.9468767642974854	0.9268112182617188	0.8935614228248596
4	0.1268775910139084	0.9508699178695679	0.9310429692268372	0.9029033780097961
5	0.11653023213148117	0.9542213082313538	0.9342265725135803	0.911575973033905
6	0.09815572202205658	0.9630632996559143	0.9495121836662292	0.9260228276252747
7	0.09114188700914383	0.9648459553718567	0.9513736963272095	0.9303376078605652
8	0.08993343263864517	0.9654877185821533	0.9516755938529968	0.9322074055671692
9	0.08623101562261581	0.9672703742980957	0.9539951682090759	0.9360893368721008
10	0.07116859406232834	0.972404420375824	0.9597104787826538	0.9475941061973572
11	0.07415373623371124	0.9701939821243286	0.9580318331718445	0.9421252608299255
12	0.06161923706531525	0.9750427603721619	0.9660241007804871	0.9504504799842834
13	0.07234658300876617	0.9711209535598755	0.9561151266098022	0.9472559094429016
14	0.0615524984896183	0.9770393371582031	0.9718173146247864	0.9510223269462585
15	0.05939893051981926	0.9786794185638428	0.9699085354804993	0.9586009979248047
16	0.05753365159034729	0.9777524471282959	0.9670014381408691	0.9587482213973999
17	0.053360965102910995	0.9794637560844421	0.9706165790557861	0.9604384899139404
18	0.05445979908108711	0.9784654974937439	0.9667384624481201	0.9612181782722473
19	0.05272112414240837	0.9787507057189941	0.9679272174835205	0.9610266089439392
20	0.05306673049926758	0.9795351028442383	0.9685039520263672	0.9632178544998169
21	0.048854291439056396	0.9822447299957275	0.9761732816696167	0.964336633682251
22	0.05227511748671532	0.979677677154541	0.9697188138961792	0.9620887041091919
23	0.05313153564929962	0.9802481532096863	0.9707714319229126	0.9631566405296326
24	0.048508938401937485	0.9827438592910767	0.9760192036628723	0.9662867784500122

25	0.04886620491743088	0.9811751246452332	0.9719424247741699	0.9649999737739563
26	0.04703705385327339	0.9821734428405762	0.9732569456100464	0.9672520160675049
27	0.045297469943761826	0.9842413067817688	0.9763553738594055	0.9710213541984558
28	0.04489907994866371	0.9826012253761292	0.9748080372810364	0.9669204950332642
29	0.042648930102586746	0.9843839406967163	0.9765157103538513	0.9711630344390869
30	0.041232965886592865	0.9837421774864197	0.9753941893577576	0.9702946543693542
31	0.042532436549663544	0.9847404360771179	0.9764229655265808	0.9727165102958679
32	0.03961828351020813	0.9851682782173157	0.9788512587547302	0.971380889415741
33	0.037029437720775604	0.9855248332023621	0.978763997554779	0.9729601740837097
34	0.043644823133945465	0.9848117232322693	0.977729856967926	0.9714489579200745
35	0.04060222581028938	0.9849543571472168	0.9777191877365112	0.971898078918457
36	0.046884480863809586	0.9823160171508789	0.9737093448638916	0.9672364592552185
37	0.03777530416846275	0.9850256443023682	0.9761620759963989	0.9738406538963318
38	0.0365433469414711	0.9850256443023682	0.9773432016372681	0.9727035164833069
39	0.03549408167600632	0.9861665964126587	0.9805475473403931	0.9730696082115173
40	0.03896011412143707	0.9856674075126648	0.9784688949584961	0.9735777378082275
41	0.0377609059214592	0.9851682782173157	0.9762074947357178	0.9743528962135315
42	0.03457755967974663	0.9871648550033569	0.9821215867996216	0.9751479029655457
43	0.0381472073495388	0.9860952496528625	0.9789875745773315	0.9745662212371826
44	0.034958578646183014	0.9870222210884094	0.9827172160148621	0.9738344550132751
45	0.031855396926403046	0.9879492521286011	0.9811365604400635	0.9785663485527039

Column1	Column6	Column7	Column8	Column9
EPOCHS	**val_loss**	**val_accuracy**	**val_precision**	**val_recall**
0	0.27502110600471497	0.8793402910232544	0.8421927094459534	0.7347826361656189
1	0.17204029858112335	0.9292534589767456	0.9215686321258545	0.830633282661438
2	0.14024868607521057	0.9483506679534912	0.9047619104385376	0.9220588207244873
3	0.14402556419372559	0.9418402910232544	0.9714764952659607	0.8318965435028076
4	0.1122051328420639	0.9565972089767456	0.9498479962348938	0.9031791687011719
5	0.09567069262266159	0.9605034589767456	0.9809825420379639	0.8868194818496704
6	0.0782787948846817	0.9704861044883728	0.9554234743118286	0.9441996812820435
7	0.08187402784824371	0.9722222089767456	0.944915235042572	0.9639769196510315
8	0.0850730910897255	0.9652777910232544	0.9281690120697021	0.957848846912384
9	0.09270628541707993	0.9626736044883728	0.9120432734489441	0.9697842001914978
10	0.09033545851707458	0.9683159589767456	0.9797822833061218	0.9130434989929199
11	0.06477411836385727	0.9761284589767456	0.9667630195617676	0.9543509483337402
12	0.05658493936061859	0.9769965410232544	0.9601139426231384	0.9642346501350403
13	0.06468281149864197	0.9748263955116272	0.9527220726013184	0.9637681245803833
14	0.07378321141004562	0.9748263955116272	0.9388039112091064	0.9796807169914246
15	0.04855166748166084	0.9822048544883728	0.9792592525482178	0.9607558250427246
16	0.05004125088453293	0.9809027910232544	0.9749631881713867	0.9608127474784851
17	0.05376531556248665	0.9813368320465088	0.9777117371559143	0.9591836929321289
18	0.05391952022910118	0.9748263955116272	0.9662261605262756	0.9494949579238892
19	0.0495980903506279	0.9822048544883728	0.9822747707366943	0.9582132697105408
20	0.055719826370477676	0.9761284589767456	0.9848024249076843	0.9350649118423462
21	0.056908369064331055	0.9782986044883728	0.9819004535675049	0.944847583770752

22	0.05289880931377411	0.9787326455116272	0.9682539701461792	0.9613180756568909
23	0.045358624309301376	0.984375	0.9738371968269348	0.9738371968269348
24	0.05299250781536102	0.9800347089767456	0.9793510437011719	0.954023003578186
25	0.04782656207680702	0.9817708134651184	0.9524475336074829	0.9883889555931091
26	0.04387468844652176	0.9852430820465088	0.9796215295791626	0.9711399674415588
27	0.045825738459825516	0.9830729365348816	0.9723837375640869	0.9709724187850952
28	0.04352383688092232	0.9835069179534912	0.9798850417137146	0.9660056829452515
29	0.039775747805833817	0.9904513955116272	0.9970149397850037	0.9709302186965942
30	0.03359300643205643	0.9865451455116272	0.9799714088439941	0.9757834672927856
31	0.037368565797805786	0.9861111044883728	0.9854227304458618	0.9684813618659973
32	0.044565871357917786	0.9830729365348816	0.9755395650863647	0.9685714244842529
33	0.035788580775260925	0.984375	0.9881129264831543	0.9595959782600403
34	0.04192376881837845	0.9835069179534912	0.972423791885376	0.972423791885376
35	0.02988714724779129	0.9874131679534912	0.9838945865631104	0.9739130139350891
36	0.039683811366558075	0.984375	0.9629101157188416	0.985401451587677
37	0.04677266627550125	0.9822048544883728	0.9676056504249573	0.9744681119918823
38	0.04325813800096512	0.9826388955116272	0.9877862334251404	0.9528718590736389
39	0.03501326963305473	0.9865451455116272	0.9809384346008301	0.9737991094589233
40	0.03370961174368858	0.9891493320465088	0.9742120504379272	0.9898107647895813
41	0.0422794334590435	0.9852430820465088	0.9646393060684204	0.986975371837616
42	0.03599558398127556	0.9861111044883728	0.9754335284233093	0.97826087474823
43	0.032046034932136536	0.9869791865348816	0.9795620441436768	0.976710319519043
44	0.04576149210333824	0.9830729365348816	0.980966329574585	0.9626436829566956
45	0.03325698524713516	0.9891493320465088	0.9940298795700073	0.9694322943687439

Anexo VI. Resultado del entrenamiento para la configuración 30% Cáncer - 70% Normal, 95% sintética, 5% real

Column1	Column2	Column3	Column4	Column5
EPOCH	**loss**	**accuracy**	**precision**	**recall**
0	0.9942446351051331	0.775884211063385	0.8268271088600159	0.8602402210235596
1	0.22633251547813416	0.8982458710670471	0.9202644228935242	0.9357302784919739
2	0.1585666686296463	0.9329007267951965	0.9474798440933228	0.9572257995605469
3	0.13502421975135803	0.9453080296516418	0.95853191614151	0.9635140895843506
4	0.12077875435352325	0.9483029246330261	0.9601456522941589	0.9663035869598389
5	0.10765866190195084	0.9580005407333374	0.9670007228851318	0.9732069969177246
6	0.09421267360448837	0.9614232778549194	0.9710630774497986	0.9739307761192322
7	0.08940800279378891	0.9655590653419495	0.9727925658226013	0.9782343506813049
8	0.0840262696146965	0.9669851660728455	0.9728964567184448	0.9801324605941772
9	0.0793703943490982	0.9686965346336365	0.9763115048408508	0.9789988994598389
10	0.07318174093961716	0.9726183414459229	0.9789634346961975	0.981957197189331
11	0.07112454622983932	0.9716200828552246	0.9775975942611694	0.9819774031639099
12	0.06301765143871307	0.9758984446525574	0.981411874294281	0.9842110872268677

Column1	Column6	Column7	Column8	Column9
EPOCH	val_loss	val_accuracy	val_precision	val_recall
0	0.36955904960632324	0.8272569179534912	0.9034205079078674	0.841349184513092
1	0.3552321195602417	0.8650173544883728	0.9321357011795044	0.8701863288879395
2	0.338529109954834	0.8671875	0.9159235954284668	0.8920595645904541
3	0.4407704174518585	0.8606770634651184	0.946020781993866	0.8490683436393738
4	0.380412220954895	0.8806423544883728	0.9259259104728699	0.9011808633804321
5	0.38688868284225464	0.8819444179534912	0.9423706531524658	0.8866297006607056
6	0.4393898546695709	0.8763020634651184	0.9428763389587402	0.8752339482307434
7	0.4202960133552551	0.8832465410232544	0.9475431442260742	0.8824984431266785
8	0.47824642062187195	0.8732638955116272	0.9599999785423279	0.8533998727798462
9	0.438091516494751	0.8836805820465088	0.9551934599876404	0.8744561672210693
10	0.4779146909713745	0.8858506679534912	0.9662069082260132	0.8674922585487366
11	0.5341655015945435	0.8697916865348816	0.9748201370239258	0.8364197611808777
12	0.5522856116294861	0.8767361044883728	0.9711064100265503	0.8500925302505493

Anexo VII. Resultado del entrenamiento para la configuración 30% Cáncer - 70% Normal, 90% sintética, 10% real

Column1	Column2	Column3	Column4	Column5
EPOCH	loss	accuracy	precision	recall
0	0.977238655090332	0.7742441296577454	0.8281651139259338	0.8546677827835083
1	0.23330318927764893	0.8956075310707092	0.9152002930641174	0.9377610087394714
2	0.17284254729747772	0.928194522857666	0.9442038536071777	0.9539071917533875
3	0.1414128690958023	0.942954957485199	0.9569053053855896	0.9617815017700195
4	0.1328486204147339	0.9478750824928284	0.9594785571098328	0.9664121866226196
5	0.1032804399728775	0.9595693349838257	0.9683037996292114	0.9741238951683044
6	0.10459408909082413	0.9578579664230347	0.9693763256072998	0.970462441444397
7	0.08850903064012527	0.9653451442718506	0.9724584817886353	0.978203296661377
8	0.08228196203708649	0.9674005508422852	0.9735092520713806	0.9801236987113953
9	0.07283110171556473	0.9712635278701782	0.9785569310188293	0.9804500341415405
10	0.07606220990419388	0.971691370010376	0.9778926968574524	0.9817755818367004

Column1	Column6	Column7	Column8	Column9
EPOCH	val_loss	val_accuracy	val_precision	val_recall
0	0.4976810812950134	0.7699652910232544	0.8097165822982788	0.8744534850120544
1	0.699528157711029	0.7517361044883728	0.8844444155693054	0.7416148781776428
2	0.7460072040557861	0.7612847089767456	0.8573360443115234	0.78904789686203
3	0.7143900394439697	0.76171875	0.8741209506988525	0.7706137895584106
4	0.8147019147872925	0.7526041865348816	0.884303629398346	0.7439553737640381
5	1.1155327558517456	0.7387152910232544	0.913430392742157	0.6951970458030701
6	0.8948642015457153	0.7630208134651184	0.8851743936538696	0.7584059834480286
7	1.0283823013305664	0.7491319179534912	0.9041309356689453	0.7182662487030029
8	1.1394071578979492	0.7426215410232544	0.9158415794372559	0.6933166980743408

9	1.2408452033996582	0.7421875	0.9156429171562195	0.6948415040969849
10	1.2680747509002686	0.7400173544883728	0.9242928624153137	0.6862260699272156

Anexo VIII. Resultado del entrenamiento para la configuración 50% Cáncer - 50% Normal, 95% sintética, 5% real

Column1	Column2	Column3	Column4	Column5
EPOCH	loss	accuracy	precision	recall
0	0.27982431650161743	0.8762121796607971	0.8847442865371704	0.8653270602226257
1	0.19295169413089752	0.9202795028686523	0.9230107665061951	0.9172247052192688
2	0.15864135324954987	0.9331146478652954	0.938374936580658	0.9271353483200073
3	0.12839914858341217	0.9483742117881775	0.9511634707450867	0.9451898336410522
4	0.11956709623336792	0.9511551856994629	0.9520205855369568	0.9502565860748291
5	0.0951257273554802	0.9632772207260132	0.9647766351699829	0.9616098403930664
6	0.08789975941181183	0.9652737975120544	0.965679943561554	0.9647142887115479
7	0.08860664069652557	0.9645607471466064	0.9662712812423706	0.9628310799598694
8	0.07592073082923889	0.9707643985748291	0.9707351922988892	0.9707351922988892
9	0.07730884850025177	0.9709070324897766	0.9729458689689636	0.9687856435775757
10	0.068502277135849	0.973402738571167	0.973594069480896	0.9731773138046265
11	0.06455837190151215	0.9750427603721619	0.9761632680892944	0.973939061164856
12	0.06553748995065689	0.9744009971618652	0.9750071167945862	0.9737555384635925
13	0.0669601559638977	0.9744723439216614	0.9744541049003601	0.9744541049003601
14	0.059614092111587524	0.9769680500030518	0.9767475128173828	0.977165699005127

Column1	Column6	Column7	Column8	Column9
EPOCH	val_loss	val_accuracy	val_precision	val_recall
0	0.3510048985481262	0.8693576455116272	0.9124876856803894	0.8140350580215454
1	0.32436105608940125	0.8784722089767456	0.8807017803192139	0.8745644688606262
2	0.3792420029640198	0.8953993320465088	0.9539999961853027	0.8302872180938721
3	0.3711824417114258	0.8880208134651184	0.9111721515655518	0.8607266545295715
4	0.31547051668167114	0.8927951455116272	0.9139193892478943	0.8670721054077148
5	0.3281508684158325	0.9032118320465088	0.912466824054718	0.8927335739135742
6	0.36754000186920166	0.9006076455116272	0.9203703999519348	0.8742304444313049
7	0.34105491638183594	0.9036458134651184	0.9180765748023987	0.888027548789978
8	0.340793251991272	0.9027777910232544	0.9063063263893127	0.8934280872344971
9	0.35741356015205383	0.89453125	0.8889837861061096	0.902079701423645
10	0.3620287775993347	0.8967013955116272	0.9121076464653015	0.8789973855018616
11	0.4315212070941925	0.9053819179534912	0.9518768191337585	0.8547968864440918
12	0.3833180367946625	0.9006076455116272	0.9338374137878418	0.861377477645874
13	0.333709716796875	0.9058159589767456	0.9073426723480225	0.9033942818641663
14	0.37041160464286804	0.8962673544883728	0.884094774723053	0.9110723733901978

Anexo IX. Resultado del entrenamiento para la configuración 50% Cáncer - 50% Normal, 90% sintética, 10% real

Column1	Column2	Column3	Column4	Column5
EPOCH	loss	accuracy	precision	recall
0	0.9491801261901855	0.7579150199890137	0.7603986263275146	0.7520354390144348
1	0.23412716388702393	0.8998146057128906	0.9053194522857666	0.8930557370185852
2	0.17368870973587036	0.9283371567726135	0.9337183833122253	0.9221335053443909
3	0.15139037370681763	0.938819169998169	0.9407386183738708	0.9367160797119141
4	0.13401612639427185	0.9445949792861938	0.9460812211036682	0.9429793357849121
5	0.11518934369087219	0.951654314994812	0.9538748264312744	0.9492515921592712
6	0.1025492325425148	0.9594979882240295	0.9594363570213318	0.9597095847129822
7	0.09703627973794937	0.9618511199951172	0.9609463810920715	0.9627302289009094
8	0.08861243724822998	0.9649173021316528	0.9658669233322144	0.9639395475387573
9	0.0903744325041771	0.9650598764419556	0.9650748372077942	0.9650748372077942
10	0.07844807952642441	0.9701226353645325	0.9698048830032349	0.9704959988594055

EPOCH	Column6	Column7	Column8	Column9
0	val_loss	val_accuracy	val_precision	val_recall
1	0.5970231294631958	0.7326388955116272	0.7957110404968262	0.6184210777282715
2	0.7398263216018677	0.7513020634651184	0.8325635194778442	0.6275022029876709
3	0.7633742690086365	0.7521701455116272	0.7724268436431885	0.7125435471534729
4	0.8479965925216675	0.7443576455116272	0.8125689029693604	0.637543261051178
5	0.8436968922615051	0.7573784589767456	0.8149316310882568	0.6692573428153992
6	0.8731358051300049	0.7673611044883728	0.8297872543334961	0.6747404932975769
7	0.9892734885215759	0.7708333134651184	0.8568102717399597	0.645048201084137
8	0.8756531476974487	0.7777777910232544	0.8430540561676025	0.6859361529350281
9	0.9626398086547852	0.7686631679534912	0.8394495248794556	0.6506666541099548
10	0.875976026058197	0.7647569179534912	0.7895734310150146	0.7224631309509277
	0.9297723770141602	0.7669270634651184	0.8009803891181946	0.7098175287246704

Anexo X. Resultado del entrenamiento para la configuración 70% Cáncer - 30% Normal, 95% sintética, 5% real

Column1	Column2	Column3	Column4	Column5
EPOCH	loss	accuracy	precision	recall
0	1.326989769935608	0.759697675704956	0.6210705041885376	0.5190532803535461
1	0.232170969247818	0.9050199389457703	0.8869495391845703	0.783815860748291
2	0.16244910657405853	0.9350399374961853	0.9133917689323425	0.8659231066703796
3	0.1320403814315796	0.9473759531974792	0.9283950328826904	0.8935361504554749
4	0.12332459539175034	0.9505847096443176	0.9339762330055237	0.8986438512802124
5	0.10337093472480774	0.9585710167884827	0.941662609577179	0.919098436832428

6	0.10006396472454071	0.9617084860801697	0.9471753239631653	0.9234620928764343
7	0.09056594967842102	0.9652737975120544	0.9498546719551086	0.9333491921424866
8	0.08053471893072128	0.9686251878738403	0.9564794301986694	0.9377830624580383
9	0.0740257054567337	0.9710496068000793	0.9540559649467468	0.9490597248077393
10	0.08129255473613739	0.9684112668037415	0.9545674324035645	0.9393579363822937
11	0.07035012543201447	0.973402738571167	0.9603555798530579	0.9505350589752197
12	0.07541337609291077	0.9707643985748291	0.9582428336143494	0.9436653256416321
13	0.0630948469042778	0.9756132364273071	0.9675937294960022	0.9505820870399475
14	0.05664442479610443	0.9781089425086975	0.9691789150238037	0.9574214816093445
15	0.05930314585566521	0.9770393371582031	0.9666426777839661	0.9565423727035522
16	0.05805124342441559	0.9764689207077026	0.9641405940055847	0.9572750926017761

Column1	Column6	Column7	Column8	Column9
EPOCH	**val_loss**	**val_accuracy**	**val_precision**	**val_recall**
0	0.2885317802429199	0.8754340410232544	0.9120654463768005	0.6463767886161804
1	0.23269023001194	0.9114583134651184	0.8565022349357605	0.8414096832275391
2	0.2075013518333435	0.9184027910232544	0.8712121248245239	0.8480826020240784
3	0.22223615646362305	0.9262152910232544	0.8945783376693726	0.8559077978134155
4	0.21788182854652405	0.9253472089767456	0.8633193969726562	0.893217921257019
5	0.19981087744235992	0.9318576455116272	0.8978102207183838	0.8760683536529541
6	0.16479362547397614	0.9435763955116272	0.9077380895614624	0.8997049927711487
7	0.2164766788482666	0.9236111044883728	0.8326898217201233	0.9336219429969788
8	0.19057074189186096	0.9388020634651184	0.8725761771202087	0.9278350472450256
9	0.19233348965644836	0.9322916865348816	0.8852223753929138	0.8903318643569946
10	0.1764671504497528	0.9418402910232544	0.913690447807312	0.8898550868034363
11	0.17100098729133606	0.9427083134651184	0.9093525409698486	0.9015691876411438
12	0.1968229115009308	0.9305555820465088	0.8627187013626099	0.9170243144035339
13	0.19721108675003052	0.9379340410232544	0.8746556639671326	0.9243085980415344
14	0.1665809005498886	0.9448784589767456	0.906204879283905	0.9101449251174927
15	0.2251087874174118	0.9331597089767456	0.84765625	0.9462209343910217
16	0.19190113246440887	0.9466145634651184	0.9420062899589539	0.8748180270195007

Anexo XI. Resultado del entrenamiento para la configuración 70% Cáncer - 30% Normal, 90% sintética, 10% real

Column1	Column2	Column3	Column4	Column5
EPOCH	**loss**	**accuracy**	**precision**	**recall**
0	1.4287718534469604	0.7497860789299011	0.6051838397979736	0.4772997498512268
1	0.23580367863178253	0.9033799171447754	0.890663743019104	0.7714694738388062
2	0.16379666328430176	0.9341129660606384	0.9136672616004944	0.8620280027389526
3	0.1360693722963333	0.9465202689170837	0.929868221282959	0.8887568116188049
4	0.1212775707244873	0.9532230496406555	0.9355707764625549	0.9066951274871826
5	0.09926304221153259	0.9612093567848206	0.9469457268714905	0.9226938486099243
6	0.10577448457479477	0.958927571773529	0.9417641162872314	0.9198001027107239

7	0.0832996591925621	0.967840850353241	0.9563636183738708	0.9357210397720337
8	0.09173199534416199	0.9626354575157166	0.9469327926635742	0.9270734190940857
9	0.08343974500894547	0.96612948179245	0.9534206986427307	0.9328269362449646
10	0.06860277056694031	0.9747575521469116	0.9614832401275635	0.954178512096405
11	0.07376779615879059	0.972404420375824	0.9617988467216492	0.9455668926239014

Column1	Column6	Column7	Column8	Column9
EPOCH	**val_loss**	**val_accuracy**	**val_precision**	**val_recall**
0	0.4051375687122345	0.8324652910232544	0.8534883856773376	0.5318840742111206
1	0.40158092975616455	0.8467881679534912	0.7877193093299866	0.6593245267868042
2	0.503253698348999	0.83984375	0.7765957713127136	0.6431717872619629
3	0.5273653268814087	0.8415798544883728	0.7586750984191895	0.6940836906433105
4	0.4841455817222595	0.8441840410232544	0.7602523565292358	0.6995645761489868
5	0.5364735722541809	0.8381076455116272	0.7300699353218079	0.7435897588729858
6	0.47887128591537476	0.8637152910232544	0.8253119587898254	0.6818851232528687
7	0.5324860215187073	0.8541666865348816	0.778294563293457	0.7223021388053894
8	0.5672143697738647	0.8585069179534912	0.8238531947135925	0.661266565322876
9	0.5340259075164795	0.8532986044883728	0.7830941081047058	0.7085137367248535
10	0.5554533004760742	0.8532986044883728	0.8013582229614258	0.6810966730117798
11	0.5588380694389343	0.8467881679534912	0.7678018808364868	0.7095851302146912

Anexo XII. Artículo de investigación

Improving Breast Cancer Detection Using DCGANs Generated Synthetic Mammograms

Luis García Tíscar

Universidad Internacional de la Rioja, Logroño (España)

Fecha 11/09/2021

RESUMEN

En el campo del Aprendizaje Automático, existe un compromiso entre la cantidad de datos con los que se entrena un modelo y los resultados del mismo. Uno de los principales problemas de la adquisición de datos son las anomalías, los valores erróneos o, principalmente, faltantes. En aplicaciones médicas, como el análisis de mamografías para la detección de tumores, estas anomalías pueden mermar el rendimiento del modelo y repercutir directamente en la vida de los pacientes, debido a un diagnóstico erróneo. Para subsanar este problema, presentamos un desarrollo de software capaz de generar mamografías sintéticas de alta resolución con presencia de tumores benignos y malignos a través de redes neuronales DCGAN (Radford et al., 2015)[1]. El objetivo es comprobar si estas imágenes sintéticas son clasificadas correctamente como tumores benignos o malignos por los clasificadores de cáncer de mama del estado del arte.

PALABRAS CLAVE

Mamografía, DCGAN, Redes Neuronales, Aprendizaje Automático

I. INTRODUCCIÓN

Las bases de datos de imágenes médicas presentan un problema: pocas son públicas y, las que lo son, no tienen suficientes imágenes disponibles.

Gracias al auge de las GAN, Generative Adversarial Networks, (Goodfellow et al., 2014) [2], como fuente de infinidad de proyectos creativos, llegamos a la conclusión de que debería ser posible usar estas redes neuronales para ofrecer una base de datos de imágenes médicas sintéticas de uso libre y gratuito para la comunidad científica y cualquier usuario que lo requiera.

II. ESTADO DEL ARTE

A día de hoy, existen numerosas aplicaciones de las redes GAN, como :style transfer (Xu et al., 2019) [3], image to image translation (Isola et al., 2016) [4], text to image translation (Krishna Goti & Ma, 2018)[5] y, la aplicación que más nos interesa: la capacidad de generar contenido totalmente sintético a partir de contenido real. Ejemplos de esta última aplicación son: generación de imágenes (Radford et al., 2015) [6], generación de texto (Yuan et al., 2020)[7] o la brillante idea de generación y combinación de música multiestilo de MuseNET (Payne, 2019) [8].

En cuanto a proyectos de uso de GANs relacionados con análisis de mamografías, existen varios proyectos, como el de (Alyafi et al., 2019) [9]. El equipo de Alyafi desarrolló un generador de "parches" de cáncer, a partir de mamografías con presencia de tumores. Otros proyectos similares son (Cha et al., 2019)[10] y (Wu et al., 2018)[11]. El primero genera imágenes mediante técnicas 3D, que requieren un conocimiento profundo de modelado 3D y el segundo crea parches de tumores como los creados por el equipo de Alyafi, y los integra en imágenes de mamografías reales. Es una solución elegante para el problema de la falta de mamografías públicas con presencia de tumores, pero creemos que nuestro proyecto es más intuitivo y directo, al ser una herramienta que genera directamente una mamografía con presencia de tumores o libre de ellos según se requiera.

III. OBJETIVOS Y METODOLOGÍA

El objetivo general es, por tanto, conseguir que un clasificador del estado del arte entrenado con nuestros datasets de imágenes generadas sintéticamente alcance unos resultados más precisos que los entrenados con datasets públicamente disponibles desbalanceados.

Para ello, desarrollaremos un software capaz de usar el modelo de StyleGAN 2 ADA para la generación de una base de datos de mamografías sintéticas, tanto con presencia de tumores como sin ellos. Cuando evaluemos la calidad de las imágenes mediante métricas adecuadas, generaremos varios datasets siguiendo diferentes distribuciones: partiremos desde la más usual (70% imágenes normales – 30% con presencia de tumores) e iremos aumentando el porcentaje de presencia de tumores y disminuyendo el de normales. Entrenaremos un modelo de clasificación de imágenes del estado del arte con dichos datasets para, seguidamente, ejecutarlo sobre una muestra de un dataset real y comprobar si mejoran los resultados.

En cuanto a la **metodología**, los pasos a seguir son los

siguientes:

Implementar el modelo elegido en un Google Colab Notebook: Hemos elegido usar la plataforma de Google porque da acceso a máquinas alojadas con GPUs potentes y memorias RAM de gran tamaño. Con el fin de reducir los tiempos de cómputo, se ha optado por pagar una licencia PRO para conseguir tiempos de ejecución ininterrumpidos de 24h, lo que facilita las cosas a la hora de entrenar el modelo.

Elegir un dataset público y prepararlo adecuadamente: Elegimos el dataset DDSM (Dissanayake Lekamlage, Afzal, Westerberg, & Cheddad, 2020) [12]. Para poder integrarlo en nuestro modelo, nos vimos obligados a editar las imágenes una por una mediante un modelo de edición de imágenes de (Schultz, n.d.) [13] en GitHub, dado que las dimensiones de las imágenes originales eran rectangulares y StyleGAN 2 ADA requiere que las imágenes de entrada sean cuadradas.

Entrenar el modelo con imágenes del dataset preparado: Entrenamos el modelo usando Google Colab PRO, aplicando la técnica de Transfer Learning desde el modelo preentrenado FFHQ512, disponible en el repositorio de StyleGAN 2 ADA. Las especificaciones de la máquina alojada del Google Colab PRO notebook son las siguientes:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.19.01    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   37C    P0    25W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Fig. 1. Especificaciones de la máquina alojada de Google Colab PRO usada para entrenar el modelo. (Imagen propia)

Comparar las imágenes generadas con las reales mediante métricas pertinentes. Igual que los investigadores de nVIDIA, usaremos una métrica fiable para comprobar los resultados de las imágenes generadas: Frechet Inception Distance (FID) (Heusel et al., 2018)[14].

Entrenar un clasificador del estado del arte con las imágenes generadas. Entrenar con las imágenes generadas un clasificador del estado del arte, VGG16 (Simonyan & Zisserman, 2015)[15] para comprobar si mejora sus resultados, medidos sobre una muestra de un dataset público con imágenes reales.

Verificar los resultados del clasificador y modificar los parámetros del modelo para mejorar su eficacia. Según los resultados del clasificador, modificaremos los parámetros del modelo mediante ensayo y error, para mejorar los resultados. Este paso será recursivo hasta obtener unos resultados aceptables.

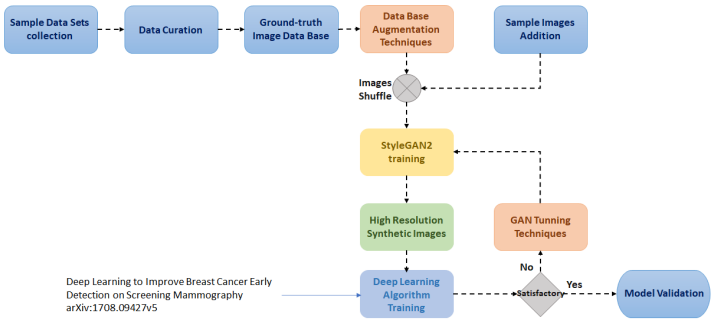


Fig. 2. Esquema de la metodología seguida. (Imagen propia)

IV. Contribución

La herramienta en sí es muy intuitiva, las celdas han de ejecutarse en un orden ya establecido e indicado en el propio notebook, para que incluso usuarios poco experimentados en Machine Learning puedan usarla. Se adjuntará el cuaderno en texto como anexo, para una mayor profundización.

Todas las celdas están debidamente comentadas tanto en castellano como en inglés dentro del notebook. Aun así, procedemos a detallar brevemente las partes del mismo.

La herramienta consta de 1 paso opcional y, a continuación, de 3 pasos bien diferenciados:

1. (OPCIONAL) Activar Google Drive: Google Colab cuenta con la posibilidad de conectarse a una cuenta de Google Drive, para así acceder a archivos personales y almacenados a dicha cuenta. De hecho, en el proceso de entrenamiento, usamos Google Drive para almacenar todas las imágenes originales, los resultados del entrenamiento, las imágenes generadas, etc. Es ALTAMENTE RECOMENDABLE usar esta opción, ya que, si se usa el espacio de almacenamiento de la sesión actual del notebook, el usuario corre el riesgo de perder todos los datos si se desconecta la sesión o cierra el notebook.

2. Clonar y acceder a las carpetas del repositorio: Clonar un repositorio significa, básicamente, copiar su contenido a otro directorio de nuestra propiedad. Por esta razón, indicamos que es recomendable conectar Google Drive al notebook, para clonar solo una vez el repositorio y mantener siempre actualizada la copia con los resultados de los entrenamientos, imágenes generadas, modelos entrenados, etc. Una vez tengamos el repositorio clonado en nuestro equipo o almacenamiento de Google Drive, procederemos a instalar las librerías de las que depende StyleGAN 2 ADA TORCH para funcionar correctamente, simplemente ejecutando la celda siguiente.

3. Cargar los modelos ya entrenados: La parte de generación de imágenes sintéticas de este proyecto ha producido 2 modelos: uno para generar imágenes con cáncer y otro para generar sin presencia de tumores. Estos dos modelos se guardan en archivos de extensión ".pkl", que se deberán cargar en la herramienta, de modo que sea capaz de generar imágenes. De nuevo, si están cargados en el Google Drive del usuario, este paso no es necesario dado que, en el siguiente punto de generación de imágenes, solo tendrán que actualizar uno de los argumentos del comando de python con la ruta en la que se encuentra cada archivo .pkl.

4. Generar las imágenes: Finalmente, llegamos a la parte de generar el dataset. Se pueden observar dos celdas casi idénticas, diferenciándose solamente en la ruta que lleva a cada modelo, uno para cáncer y otro para normales. En estas celdas se ejecuta un comando python que llama al archivo "generate.py" de StyleGAN 2 ADA, comando que recibe 4 argumentos muy

importantes. Aunque estos argumentos se hallan perfectamente explicados en la herramienta, creemos conveniente repetir la explicación aquí puesto que es un paso crucial:

a. carpeta_salida: La carpeta donde queremos que se guarden las imágenes. De nuevo, si se quiere usar una carpeta de Google Drive, solo se deberá pegar la ruta a dicho directorio en este argumento.

b. trunc: Si existen muestras con poco peso en el dataset de entrenamiento, puede que algunas de las imágenes generadas presenten artefactos como saturaciones excesivas, deformaciones en las siluetas, color, etc. Para evitar dichos artefactos, StyleGAN implementa este argumento, que "trunca" o fuerza el valor del vector latente introducido para que sea de un valor más cercano al de la media. En resumen, este valor solo ha de modificarse si se requiere más diversidad entre las imágenes a generar, pero el usuario ha de ser consciente de que pueden aparecer artefactos y errores en dichas imágenes. Por defecto, el valor es de 1 (sin truncamiento) y se puede variar hasta llegar al 0 (máximo truncamiento).

c. seeds: Este argumento se usa para especificar el número de imágenes que queremos generar. Debe seguir el formato X-Y, siendo X la primera de dichas "semillas", que el generador convierte en imágenes e Y, la última. Por defecto, está definido como 0-9999, es decir, generará 10000 imágenes.

d. network: Este argumento es la ruta al archivo .pkl que contiene el modelo para generar imágenes. Si el usuario ha activado su drive y tiene el archivo allí, solo tiene que copiar y pegar la ruta SIN CORCHETES, por ejemplo: --network=/content/drive/MyDrive/models/cancer.pkl. Si no tiene drive activado, asumimos que ha subido el modelo con las celdas anteriores, por lo que se encontrará en los archivos temporales de google, "/content/".

5. Compresión y descarga del dataset: Solo hay que introducir la ruta donde están almacenadas las dos carpetas con imágenes generadas de cáncer y normales para comprimirlas en un .zip, llamado "dataset.zip" y que se descargue automáticamente cuando acabe de crearlo.

El resultado de las imágenes generadas se guardará dentro del comprimido, cada tipo de imágenes en la carpeta de salida indicada, en formato PNG, y con el número de semilla que la produjo:

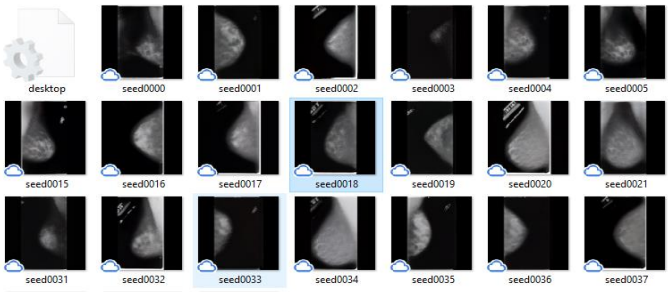


Fig. 3. Muestra del resultado de las imágenes generadas. (Imagen propia)

V. Evaluación y Discusión de resultados

Generar datasets de mamografías sintéticas siguiendo diferentes configuraciones con nuestra herramienta.

El primer paso en la evaluación de resultados debe ser, sin duda, evaluar la calidad de las imágenes generadas.

Evaluación de las imágenes generadas

Para este proyecto generamos 3 datasets con 24.000 imágenes, todas sintéticas. El porcentaje de imágenes con cáncer y sin él sigue 3 distribuciones diferentes:

1. 30% imágenes con cáncer y 70 % sin él:

30-70			
24000			
TRAIN		**TEST**	
70%		30%	
16800		7200	
CANCER	**NORMAL**	**CANCER**	**NORMAL**
30%	70%	30%	70%
5040	11760	2160	5040

Tabla 1: Distribución dataset sintético de entrenamiento 30% cáncer – 70% normal.

2. 50% imágenes con cáncer y 50% sin él:

50-50			
24000			
TRAIN		**TEST**	
70%		30%	
16800		7200	
CANCER	**NORMAL**	**CANCER**	**NORMAL**
50%	50%	30%	70%
8400	8400	2160	5040

Tabla 2: Distribución dataset sintético de entrenamiento 50% cáncer – 50% normal

3. 70% imágenes con cáncer y 30% sin él:

70-30			
24000			
TRAIN		**TEST**	
70%		30%	
16800		7200	
CANCER	**NORMAL**	**CANCER**	**NORMAL**
70%	30%	30%	70%
11760	5040	2160	5040

Tabla 3: Distribución dataset sintético de entrenamiento 70% cáncer – 30% normal

El objetivo de generar estos datasets solo con imágenes sintéticas era comprobar si se alcanzaban resultados aceptables en el entrenamiento y posterior clasificación. En caso contrario, como se mencionó en el capítulo de metodología, mezclaremos las imágenes generadas con imágenes reales del dataset original en diferentes proporciones, para que el dataset sea más variado y evitar aún más el sobreajuste. Dichas proporciones serían las siguientes:

4. 30% imágenes con cáncer y 70 % sin él, de las cuales, 95% son sintéticas y 5% reales:

MIXED 30% CANCER - 70% NORMAL (95% SYNT - 5% REAL)	
24000	
TRAIN	**TEST**
70%	30%
16800	7200

CANCER		NORMAL		CANCER		NORMAL	
30%		70%		30%		70%	
5040		11760		2160		5040	
SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL
95%	5%	95%	5%	95%	5%	95%	5%
4788	252	11172	588	2052	108	4788	252

Tabla 4: Distribución dataset de entrenamiento 70% cáncer – 30% normal, con 95% de imágenes sintéticas y 5% imágenes reales.

5. 50% imágenes con cáncer y 50 % sin él, de las cuales, 95% son sintéticas y 5% reales:

MIXED 50% CANCER - 50% NORMAL (95% SYNT - 5% REAL)							
24000							
TRAIN				TEST			
70%				30%			
16800				7200			
CANCER		NORMAL		CANCER		NORMAL	
50%		50%		30%		70%	
8400		8400		2160		5040	
SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL
95%	5%	95%	5%	95%	5%	95%	5%
7980	420	7980	420	2052	108	4788	252

Tabla 5: Distribución dataset de entrenamiento 50% cáncer – 50% normal, con 95% de imágenes sintéticas y 5% imágenes reales.

6. 70% imágenes con cáncer y 30 % sin él, de las cuales, 95% son sintéticas y 5% reales:

MIXED 70% CANCER - 30% NORMAL (95% SYNT - 5% REAL)							
24000							
TRAIN				TEST			
70%				30%			
16800				7200			
CANCER		NORMAL		CANCER		NORMAL	
70%		30%		30%		70%	
11760		5040		2160		5040	
SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL
95%	5%	95%	5%	95%	5%	95%	5%
11172	588	4788	252	2052	108	4788	252

Tabla 6: Distribución dataset de entrenamiento 70% cáncer – 30% normal, con 95% de imágenes sintéticas y 5% imágenes reales.

7. 30% imágenes con cáncer y 70 % sin él, de las cuales, 90% son sintéticas y 10% reales:

MIXED 50% CANCER - 50% NORMAL (90% SYNT - 10% REAL)							
24000							
TRAIN				TEST			
70%				30%			
16800				7200			
CANCER		NORMAL		CANCER		NORMAL	
50%		50%		30%		70%	
8400		8400		2160		5040	
SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL
90%	10%	90%	10%	90%	10%	90%	10%
7560	840	7560	840	1944	216	4536	504

Tabla 7: Distribución dataset de entrenamiento 30% cáncer – 70% normal, con 90% de imágenes sintéticas y 10% imágenes reales.

8. 50% imágenes con cáncer y 50 % sin él, de las cuales, 90% son sintéticas y 10% reales:

MIXED 50% CANCER - 50% NORMAL (90% SYNT - 10% REAL)							
24000							
TRAIN				TEST			
70%				30%			
16800				7200			
CANCER		NORMAL		CANCER		NORMAL	
50%		50%		30%		70%	
8400		8400		2160		5040	
SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL
90%	10%	90%	10%	90%	10%	90%	10%
7560	840	7560	840	1944	216	4536	504

Tabla 8: Distribución dataset de entrenamiento 50% cáncer – 50% normal, con 90% de imágenes sintéticas y 10% imágenes reales.

9. 70% imágenes con cáncer y 30 % sin él, de las cuales, 90% son sintéticas y 10% reales:

MIXED 70% CANCER - 30% NORMAL (90% SYNT - 10% REAL)							
24000							
TRAIN				TEST			
70%				30%			
16800				7200			
CANCER		NORMAL		CANCER		NORMAL	
70%		30%		30%		70%	
11760		5040		2160		5040	
SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL	SYNTHETHIC	REAL
90%	10%	90%	10%	90%	10%	90%	10%
10584	1176	4536	504	1944	216	4536	504

Tabla 9: Distribución dataset de entrenamiento 70% cáncer – 30% normal, con 90% de imágenes sintéticas y 10% imágenes reales.

Una vez descritas todas las configuraciones de datasets que llevaremos a cabo, es conveniente detallar qué es por qué y cómo se usa la puntuación FID, una métrica para analizar la calidad de imágenes sintéticas generadas por redes GAN.

Una métrica para evaluar imágenes generadas por una GAN: Frechet Inception Distance (FID)

La calidad de las imágenes sintéticas que hemos generado es primordial, dado que el clasificador se entrenará con ellas y los resultados de dicho entrenamiento dependen en gran medida de la calidad de los datos de origen. Por ello, como mencionamos en el capítulo 3, usaremos la métrica FID para la evaluación de las mamografías sintéticas. Para explicar detalladamente por qué se usa esta métrica, hay que detenerse en qué propiedades queremos usar para medir si una imagen sintética es "buena". Nos basaremos en dos propiedades:

1. Fidelidad: Queremos que el modelo genere imágenes de una calidad similar a las originales.
2. Diversidad: Queremos que el modelo genere imágenes que NO están presentes en el dataset, es decir, que genere imágenes nuevas.

Por lo tanto, queremos usar una métrica que evalúe ambas propiedades, pero ¿en qué debe basarse nuestra métrica para comparar imágenes basándose en su fidelidad y diversidad? A priori, son dos términos muy abstractos y difíciles de concretar matemáticamente. Dos de las aproximaciones más usuales en el campo de la visión computacional son:

1. La distancia de píxeles: Es una métrica anticuada y poco eficiente, dado que simplemente sustrae los valores de píxeles entre dos imágenes y el resultado es la "distancia de píxel".

2. La distancia de características: Para esta métrica, se usa un modelo preentrenado de clasificación de imágenes y se usa como una capa intermedia. El vector resultante es una representación a alto nivel de la imagen con el que se puede calcular una métrica de distancia más fiable y estable.

Como decíamos, se usa el modelo Inception V3 model (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015) [16], preentrenado con un dataset masivo, el renombrado Imagenet (Deng, y otros, 2009)[17]. En lugar de comparar píxel a píxel las imágenes, lo que la FID compara es la media y la desviación estándar de una de las capas profundas del modelo Inception V3. El modelo, como decíamos anteriormente, recibe dos distribuciones Gaussianas multidimensionales distintas: la de las características de las imágenes sintéticas generadas por el Generador y la de las características de las imágenes reales usadas para el entrenamiento de la DCGAN. El valor de la FID, cuanto más cercano a 0, mayor similitud entre las imágenes sintéticas y originales, lo que indica mayor calidad de las sintéticas.

Evolución de la FID vs Tiempo de entrenamiento de cada modelo generador

Una vez que comprendemos cómo funciona y lo que compara la FID, podemos visualizar la evolución de dicha métrica frente al tiempo de entrenamiento de cada modelo generador de nuestra herramienta: el generador de mamografías con cáncer y el generador de normales:

FID vs Tiempo de entrenamiento para el generador de mamografías con cáncer

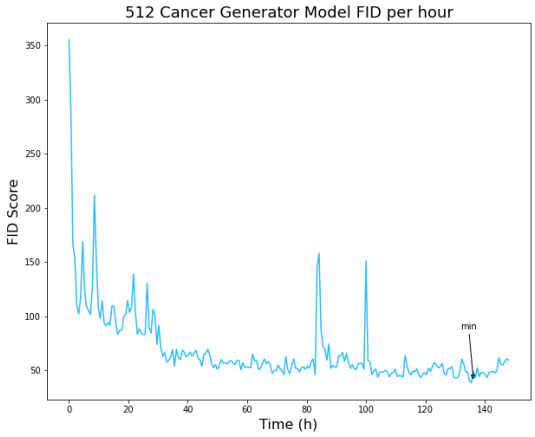


Fig. 4. Gráfica de la evolución de la FID vs Tiempo de entrenamiento del generador de cáncer. (Imagen propia)

Lo primero que llama la atención de la Figura 19 son las anomalías o "picos" que aparecen en la función representada de la FID vs tiempo. Estas anomalías se deben a que Google Colab PRO, al ser un servicio tan económico, debe restringir de alguna manera las sesiones de conexión a sus entornos remotos. Dichas sesiones pueden ser, como mucho, de 24h consecutivas, pero después de ese límite desconectan el entorno y paran la ejecución de cualquier programa que estuvieras ejecutando en el notebook. Un modelo generador como el que proponemos necesita de días para entrenarse, pero por suerte, los investigadores de StyleGAN ya contaban con este inconveniente, y desarrollaron su herramienta para que se pudiera "guardar" el estado del modelo en cada época en un fichero .pkl, y retomar el entrenamiento más adelante a partir de los valores guardados en el fichero. Sin embargo, en la primera época de cada vuelta al entrenamiento después de una ejecución, se producen esos valores tan anómalos de la FID, aunque en la siguiente época vuelva a obtener un valor muy aproximado al que tenía el modelo cuando se paró la ejecución anterior.

Obviando dichas anomalías, el valor mínimo alcanzado es de 49. Teniendo en cuenta que los investigadores de nVIDIA, en un entrenamiento de 21 días, usando una GPU mucho más potente y acceso a 25.000 imágenes para entrenar consiguieron un FID de alrededor de 10, nuestro resultado es lo suficientemente bueno para generar imágenes sintéticas con el modelo.

FID vs Tiempo de entrenamiento para el generador de mamografías normales

En cuanto al modelo generador de mamografías sin cáncer, el número de imágenes en el dataset de entrenamiento era menor, 2409 frente a 4049 de cáncer. Era de esperar, por tanto, que a menor cantidad de imágenes para entrenar, peor calidad de las imágenes sintéticas. Atendiendo a la figura 5, en efecto, fue así:

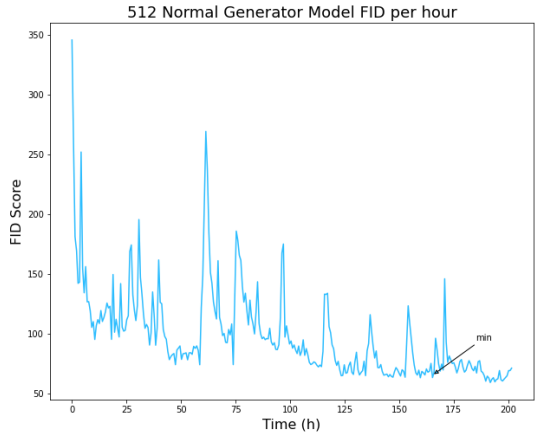


Fig.5 . Gráfica de la evolución de la FID vs Tiempo de entrenamiento del generador normal. (Imagen propia)

El valor mínimo de FID es de 60 aproximadamente, y se tardó casi 20 horas más en alcanzarlo. Cabe destacar que ambos entrenamientos se pararon cuando el mecanismo de callback determinó que la mejora de la métrica FID había convergido a su máximo. Este mecanismo alarga el entrenamiento durante un número específico de épocas de margen y comprueba si ha mejorado la métrica pertinente respecto a un mínimo. Si ha mejorado, permite continuar el entrenamiento y si no, lo para.

Por lo tanto, una vez alcanzados los mínimos valores de la puntuación FID que permitían las condiciones de entrenamiento (número de imágenes originales, potencia de la GPU, capacidad de la memoria, etc.), decidimos empezar a generar los datasets sintéticos siguiendo las diferentes configuraciones para entrenar con ellos el clasificador VGG16.

Como hemos comentado anteriormente, hemos entrenado el clasificador con 3 configuraciones diferentes:

Configuración 30% cáncer 70 % normal:

Esta configuración es muy interesante desde el punto de vista comparativo, ya que los datasets reales de mamografías suelen seguir una proporción parecida: pocos casos de cáncer y muchos de imágenes normales. Por lo tanto, los resultados de la clasificación usando el modelo entrenado con esta configuración servirán de background para comparar los resultados posteriores.

Resultados del entrenamiento:

Los resultados del entrenamiento son extremadamente buenos. Partiendo de un valor cercano al 65% de accuracy y acabando en un valor cercano al 98%. El resultado de los valores de las métricas / época se adjunta en el capítulo 9: Anexos.

A continuación, en las figuras 4,5,6 y 7 presentan los resultados de cada métrica para un análisis más breve y visual, en las que se presentan dos series de datos: los de train (azul) y los de test (rosa):

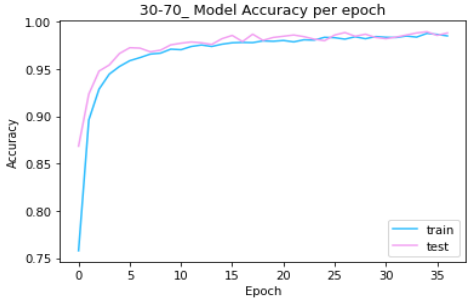


Fig. 6. Exactitud del modelo 30%-70%. (Imagen propia)

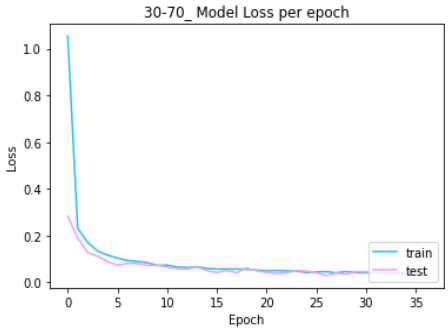


Fig. 7. Pérdida del modelo 30%-70%. (Imagen propia)

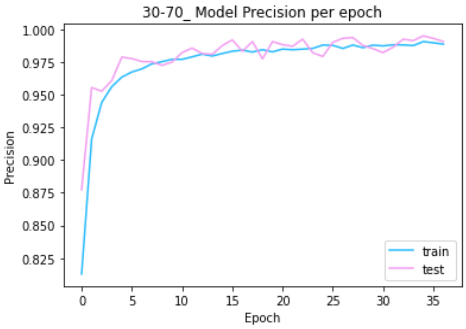


Fig. 8. Pérdida del modelo 30%-70%. (Imagen propia)

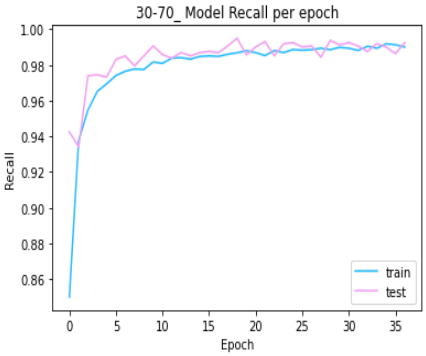


Fig. 9. Sensibilidad del modelo 30%-70%. (Imagen propia)

Resultados de la clasificación

En este apartado, veremos los resultados mediante una matriz de confusión, parecida a la que hemos detallado antes, pero cortesía de un repositorio en GitHub (Trimarchi, 2019)[18].

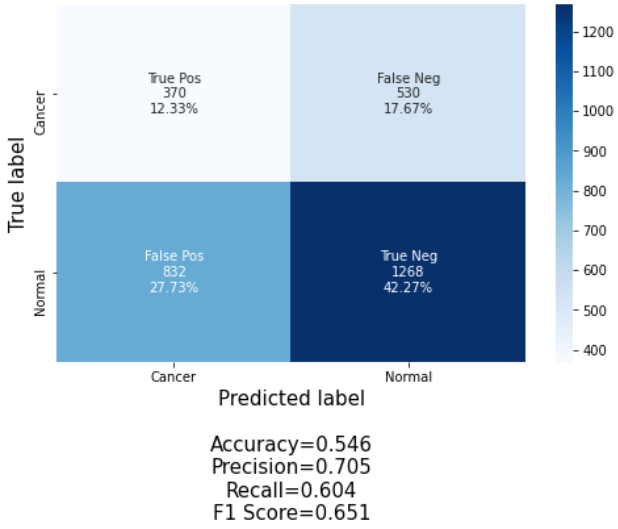


Fig. 10.: Matriz de confusión de la clasificación con el modelo 30%-70%. (Imagen propia)

Configuración 50% cáncer 50% normal:

Resultados del entrenamiento

Igual que la configuración anterior, los resultados del entrenamiento de este dataset más equilibrado son muy buenos:

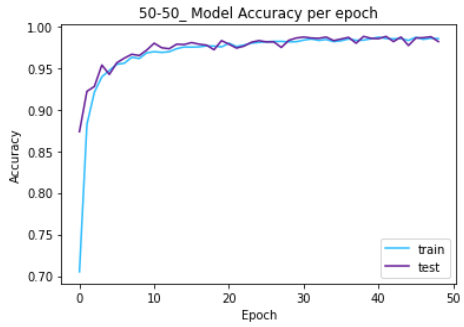


Fig. 11. Exactitud del modelo 50%-50%. (Imagen propia)

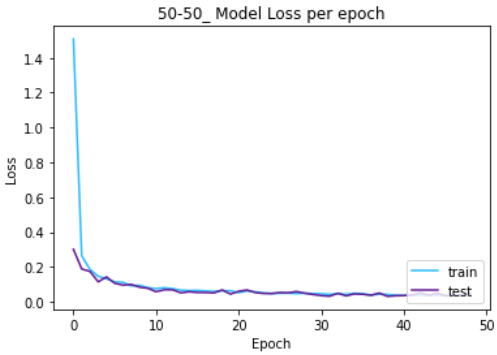


Fig. 12. Pérdida del modelo 50%-50%. (Imagen propia)

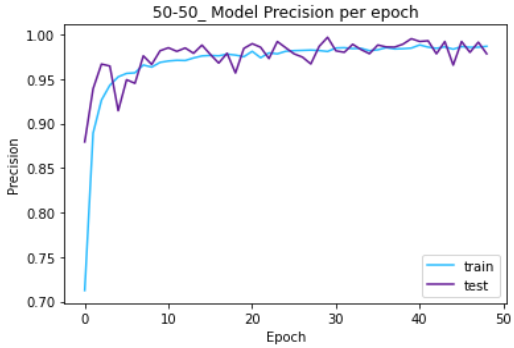


Fig. 13. Precisión del modelo 50%-50%. (Imagen propia)

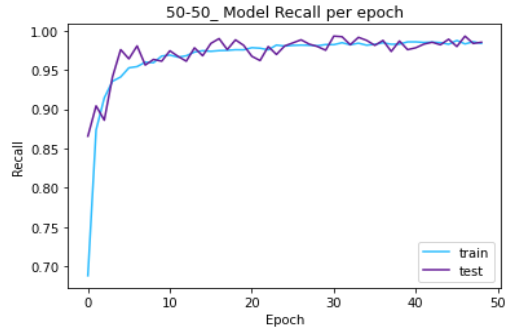


Fig. 14. Sensibilidad del modelo 50%-50%. (Imagen propia)

En este caso, aunque también se aprecia el fenómeno de obtención de mejores resultados en el de test que en train, la diferencia es mínima; indicador de un menor sobreajuste en el modelo.

Resultados de la clasificación

Aunque el porcentaje de Falsos Negativos ha disminuido, algo positivo, el resto de las métricas han empeorado ligeramente. Es muy probable que el hecho de que solo se entrene con imágenes sintéticas propicie un sobreajuste demasiado potente como para que el clasificador ofrezca buenos resultados cuando se enfrenta a imágenes reales, con diferentes características.

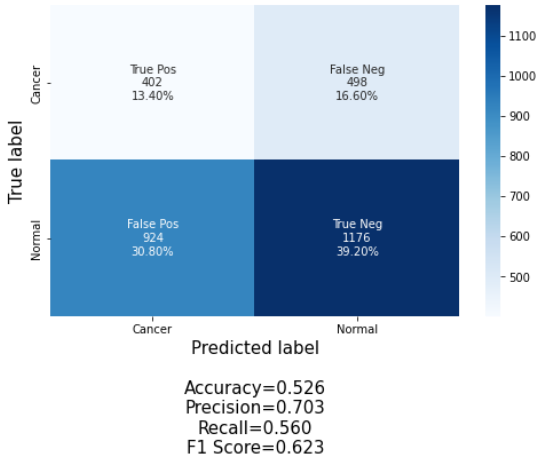


Fig. 15. Matriz de confusión de la clasificación con el modelo 50%-50%. (Imagen propia)

Configuración 70% cancer 30% normal

Resultados del entrenamiento

Finalmente, en cuanto a los resultados de entrenamiento, son muy similares a los otros dos casos anteriores. Las 4 métricas de entrenamiento convergen rápidamente hacia resultados muy positivos. La diferencia entre los resultados del subset de train y test es mínima, indicador de poco sobreajuste inicial en el entrenamiento del modelo. Procedemos a comparar los resultados entre los de 50% cáncer -50% normal y los de 30% cáncer-70% normal.

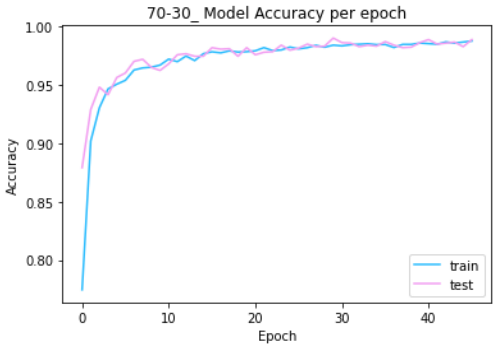


Fig. 16. Exactitud del modelo 70%-30%. (Imagen propia)

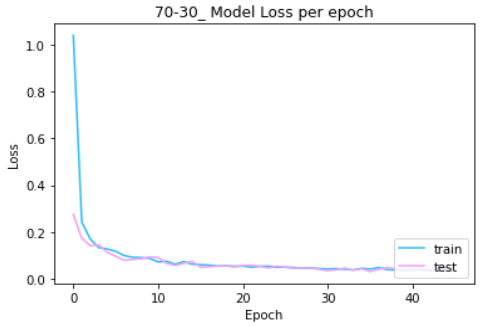


Fig. 17. Pérdida del modelo 70%-30%. (Imagen propia)

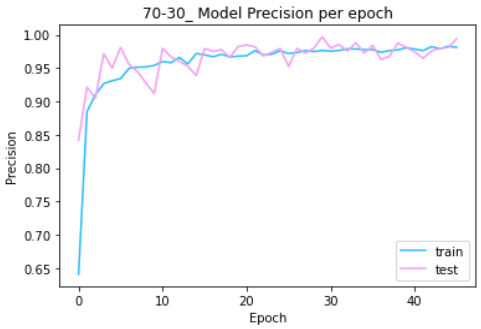


Fig. 18. Precisión del modelo 70%-30%. (Imagen propia)

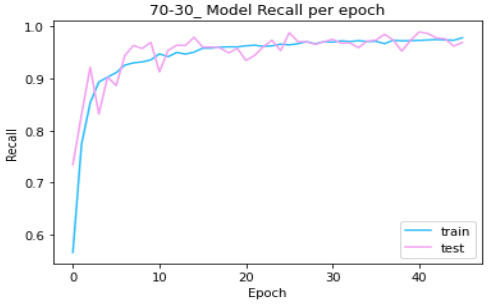


Fig. 19. Sensibilidad del modelo 70%-30%. (Imagen propia)

Podríamos concluir que los resultados de los entrenamientos son muy parecidos entre sí.

Resultados de la clasificación:

Finalmente, los resultados de la clasificación del modelo entrenado con este dataset desequilibrado, 70% cáncer y 30% normal, son bastante parecidos a los del modelo entrenado por el dataset equilibrado 50%-50% y el primero, entrenado por 30% cáncer y 70% normal. Lo podemos comprobar con la siguiente matriz de confusión:

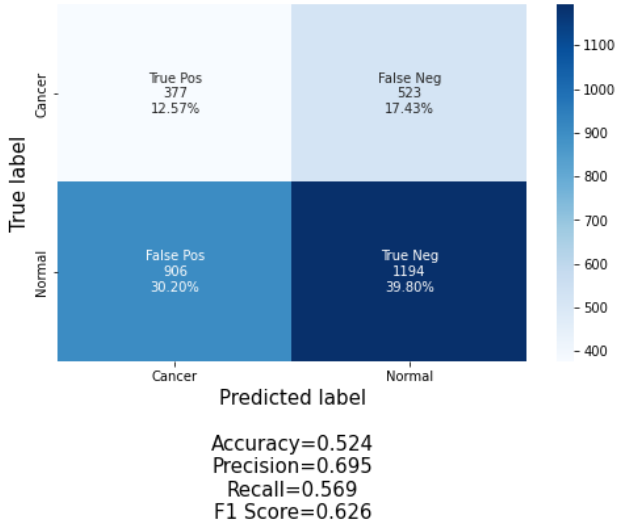


Fig. 20. Matriz de confusión de la clasificación con el modelo 70%-30%. (Imagen propia)

Creemos que esta similitud podría estar producida por la falta de variedad en las imágenes con las que se entrenan los 3 modelos. Por ello, el siguiente paso fue formar datasets con imágenes sintéticas generadas mezcladas con imágenes reales del dataset original, en diferentes proporciones.

Reentrenar VGG16 con datasets formados por imágenes sintéticas generadas mezcladas con reales.

Como hemos podido comprobar, los resultados del clasificador entrenado con distintas configuraciones de datasets formados exclusivamente por imágenes sintéticas distan mucho de lo deseable. Por tanto, para evitar que el modelo caiga en sobreajuste, la opción más viable era mezclar las imágenes sintéticas generadas con imágenes reales del dataset original. De esta forma, el modelo tendrá más capacidad de generalización, al aprender de las diferentes características de imágenes reales y sintéticas. Siguiendo lo indicado en el apartado 6.1, generamos los datasets mezclados con diferentes porcentajes de imágenes reales y sintéticas y entrenamos el modelo con ellos, obteniendo los siguientes resultados.

Configuración 30% cáncer – 70% normal, con 95% imágenes sintéticas y 5% reales.

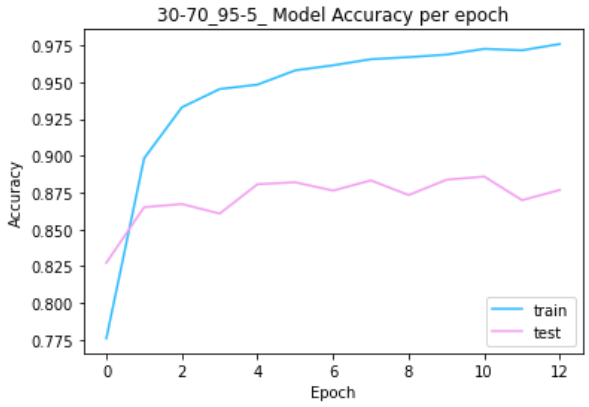


Fig. 21. Comparación de la exactitud entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Lo primero que observamos, comparando los resultados, es que la diferencia entre precisión sobre el subconjunto de test se diferencia más en el modelo entrenado con un 5% de imágenes reales. Aunque esta diferencia parezca abrupta, si nos fijamos en la escala del eje vertical, la diferencia es tan solo del orden de décimas. Esta diferencia se observa debido a que el subconjunto de test ahora presenta imágenes reales, poco presentes en el dataset de entrenamiento y tiende a fallar cuando debe clasificar una de las muestras reales. Aunque parezca algo negativo (presenta más sobreajuste que el modelo sintético puro), el modelo tiene que aprender a identificar las imágenes reales y seguramente obtenga mejores resultados en un ejercicio de clasificación real.

Además, si nos fijamos en el eje horizontal, el modelo ha considerado que a partir de la época 12 ya no se mejoraba la clasificación y decide parar el entrenamiento. En cuestión de optimización de tiempo, el nuevo modelo es mucho más rápido de entrenar.

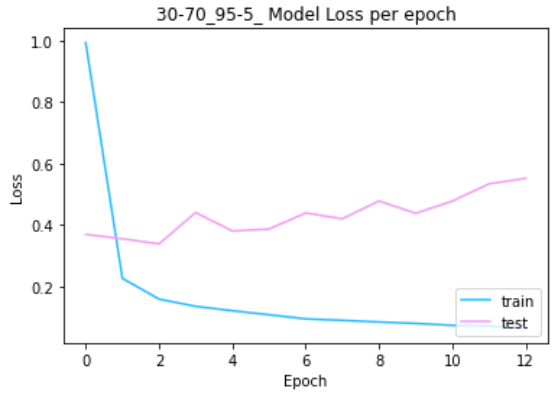


Fig. 22. Comparación de la pérdida entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la pérdida, el nuevo modelo (izq) presenta una diferencia en la evolución de la función con el tiempo en cuanto al subconjunto de test. No evoluciona favorablemente, convergiendo a un valor menor como el modelo puramente sintético. Este comportamiento es problemático y tendremos que comprobar si se resuelve introduciendo un mayor porcentaje de imágenes reales con las sintéticas. La pérdida, al ser la métrica con la que la función de callback detiene el entrenamiento si no mejora dentro de un margen temporal, ha provocado que el entrenamiento durase solo 12 épocas, como comentábamos en el resultado anterior.

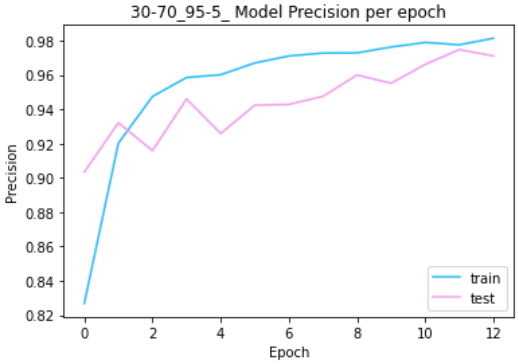


Fig.23. Comparación de la precisión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la precisión, podemos observar que la diferencia entre train y test no es tan grande como en la exactitud y converge favorablemente a resultados cada vez más cerca del 1.

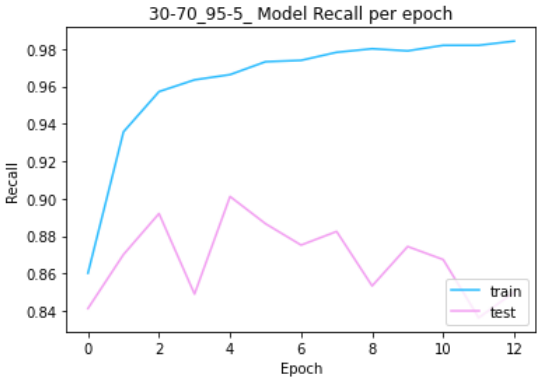


Fig. 24. Comparación de la sensibilidad entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Igual que la pérdida, la sensibilidad del modelo se desploma, comparando con el modelo puramente sintético. El ratio de los verdaderamente positivos decrece, por lo que el modelo no es capaz de diferenciar bien cuando es cáncer o normal.

Resultado de la clasificación

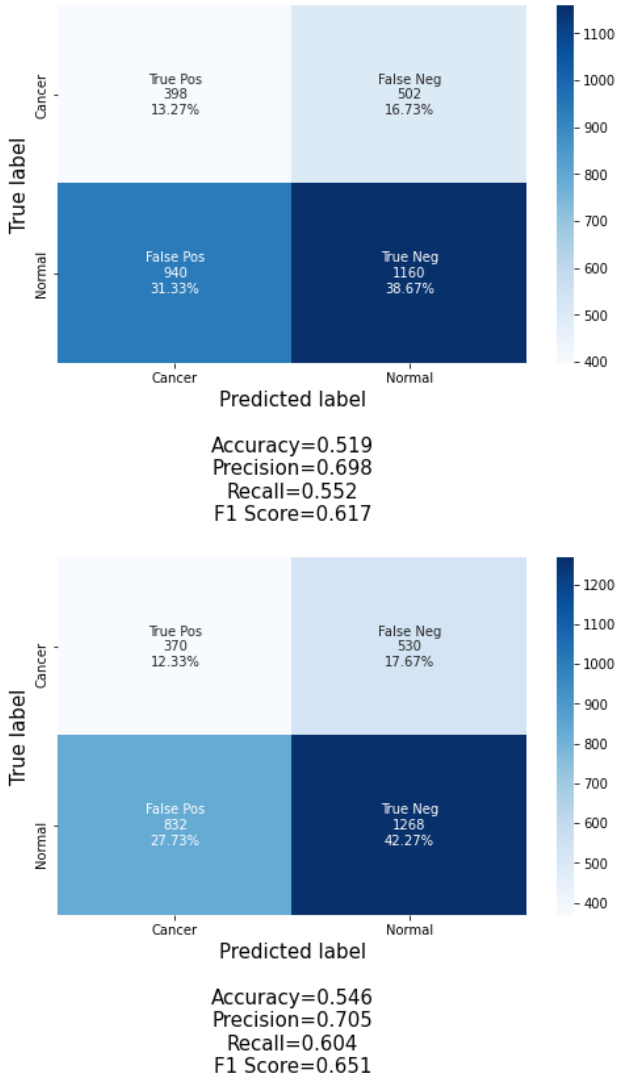


Fig. 25. Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 95% sintéticas – 5% reales (arriba) y el entrenado con un dataset 30%-70% puramente sintético (abajo). (Imagen propia)

Observamos que la calidad en general de los resultados baja respecto a la clasificación que se realizó con el modelo puramente sintético. Creemos que esto se debe a que la introducción de un porcentaje de imágenes reales tan pequeño, el 5%, no influye positivamente en los resultados del modelo. Es por ello que realizamos un segundo entrenamiento y clasificación con un porcentaje un poco más alto, del 10%. Este porcentaje es el máximo al que podemos aspirar, dado que el dataset DDSM no es tan grande como para aumentar el número de imágenes reales introducidas más allá del 10%. Sin embargo, subir dicho número tampoco sería coherente con el objetivo proyecto, dado que buscábamos una herramienta para poblar datasets con un número pequeño de imágenes reales, como es el caso.

Configuración 30% cáncer – 70% normal, con 90% imágenes sintéticas y 10% reales.

Resultados del entrenamiento

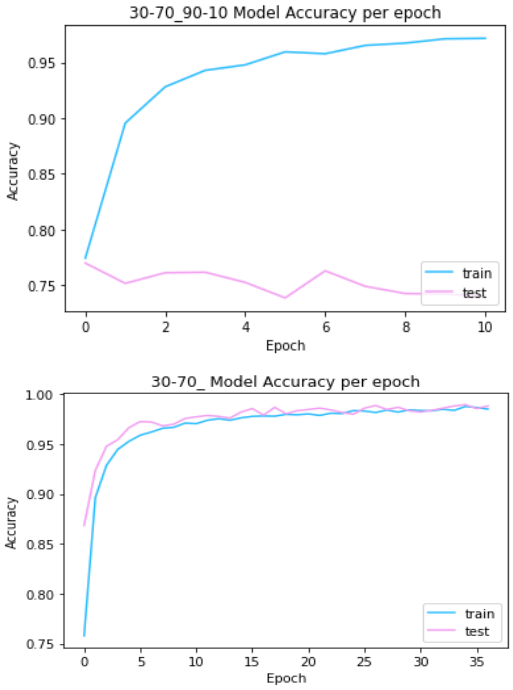


Fig. 26. Comparación de la exactitud entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Como en el caso del modelo entrenado por el dataset mezclado 95% sintético - 5% real, se aprecia una diferencia más abrupta entre los resultados de la clasificación sobre el subset de train y test, indicando más sobreajuste del modelo.

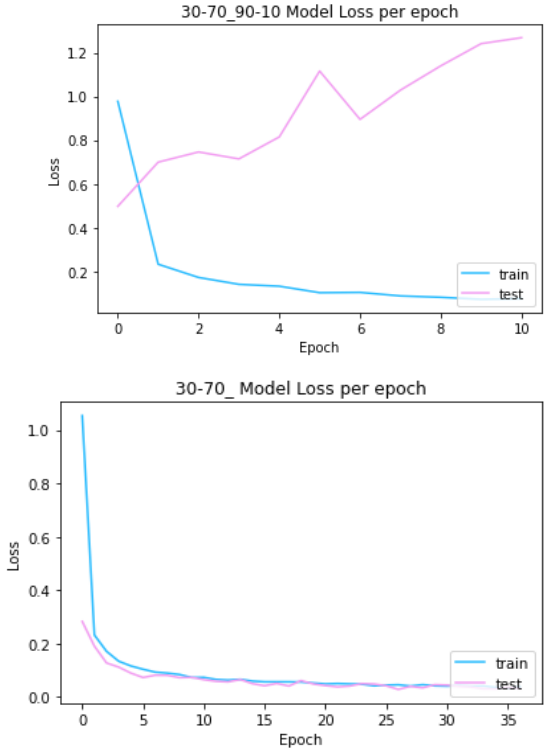


Fig. 27. Comparación de la pérdida entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la pérdida, observamos que crece más rápido aún

que el caso 95%-5%, dado que ha parado el entrenamiento 2 épocas antes, en la diez. Estos resultados parecen reflejar que, como en la mayoría de las aplicaciones de aprendizaje automático, sería importante buscar un compromiso entre la pérdida y las demás métricas de clasificación, para encontrar un resultado óptimo.

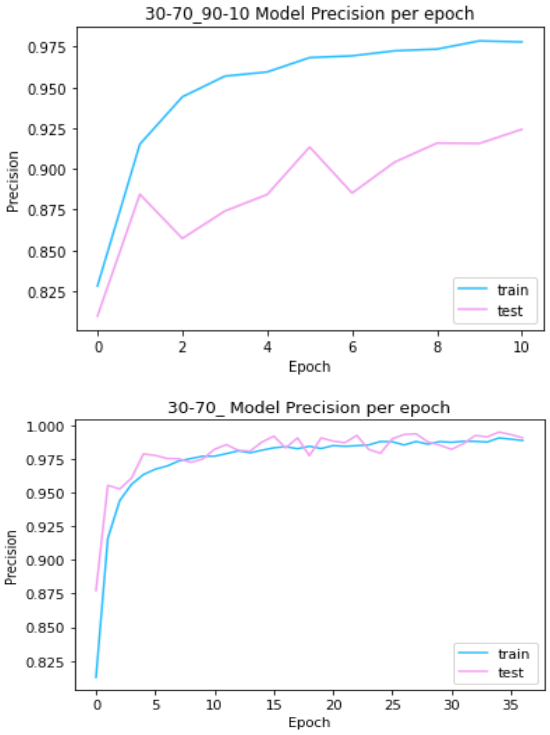
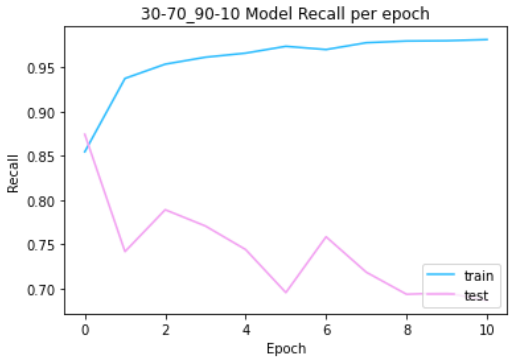


Fig. 28. Comparación de la precisión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la precisión, refleja un poco más de sobreajuste del modelo, pero se comporta adecuadamente, convergiendo a valores más altos según pasa el tiempo de entrenamiento.



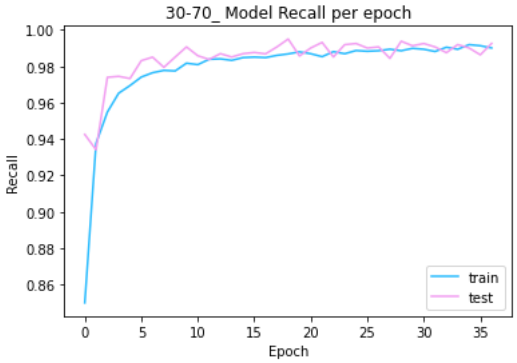


Fig. 29. Comparación de la sensibilidad entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

La sensibilidad, como en el caso anterior de 95% - 5%, también decrece y más abruptamente. Como decíamos, la introducción de imágenes reales al dataset genera un poco más de sobreajuste en el proceso de entrenamiento, pero realmente también lo reduce cuando se prueba el modelo en una clasificación real, como veremos a continuación.

Resultados de la clasificación

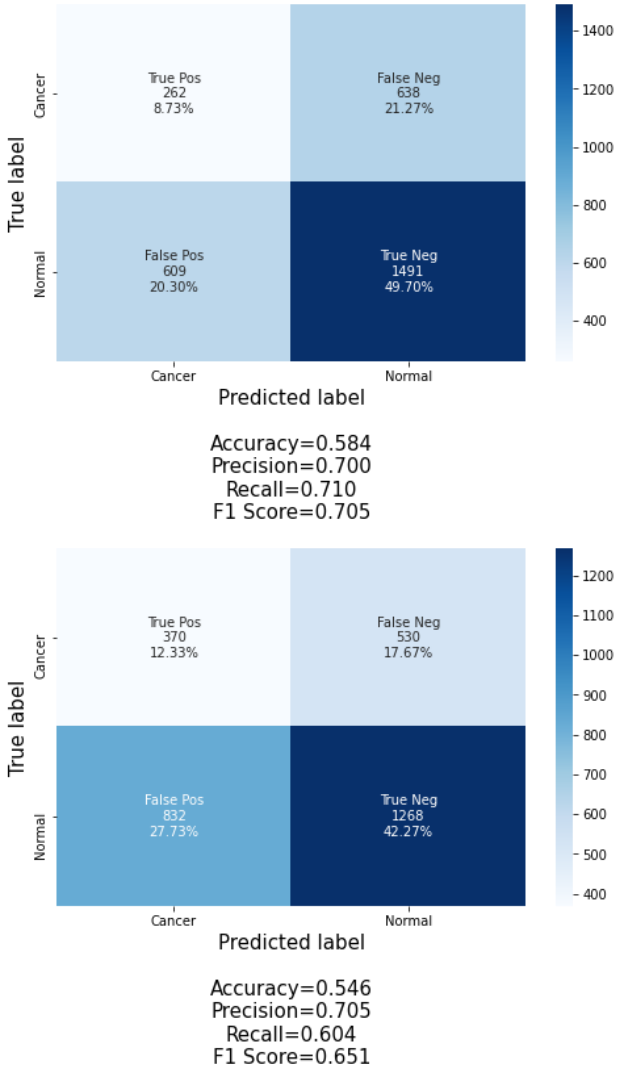


Fig. 30. Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 30% cáncer y 70% normales mezclando 90% sintéticas – 10% reales (arriba) y el entrenado con un dataset 30%-70% puramente sintético (abajo). (Imagen propia)

Como podemos observar, mejoran todas las métricas salvo la precisión, que cae solamente en un 0.005. parece ser que la introducción de imágenes reales al dataset de entrenamiento ha mejorado las métricas del modelo entrenado solo con imágenes sintéticas. Sin embargo, algo preocupante es que el porcentaje de Falsos Negativos, recordemos que es una métrica crucial en diagnóstico médico, ha subido. Será muy importante comprobar cómo se comporta dicha métrica en los resultados del dataset equilibrado (50% cáncer 50% normales) y mezclado con imágenes reales.

Configuración 50% cáncer – 50% normal, con 95% imágenes sintéticas y 5% reales.

Resultados del entrenamiento

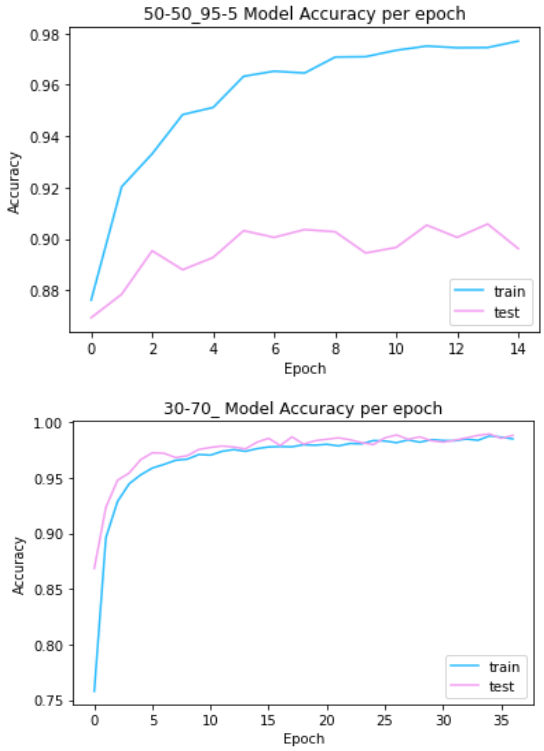
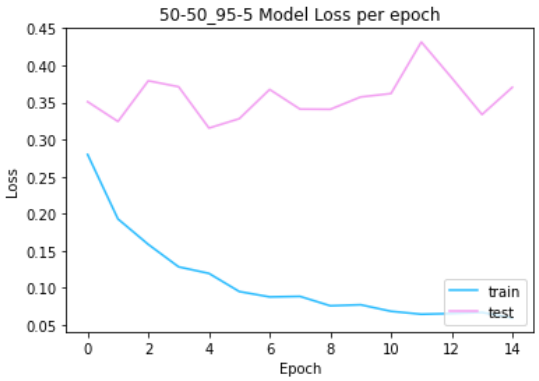


Fig. 31. Comparación de la exactitud entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Siguiendo el tono general de los últimos entrenamientos, la distancia de los resultados de la clasificación entre test y train aumenta respecto a la de los resultados del modelo entrenado con un dataset 50% cáncer 50% normal puramente sintético.



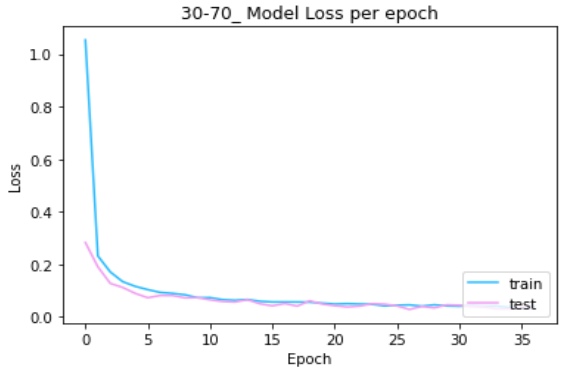


Fig. 32. Comparación de la pérdida entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

En cuanto a la pérdida, aunque más alta en el subset de test que en el de entrenamiento, en este caso no desciende continuamente, provocando que el mecanismo de callback termine el entrenamiento considerando que no va a mejorar el resultado con el tiempo. En este caso, la pérdida varía alrededor del valor fijo de 0.35 aproximadamente.

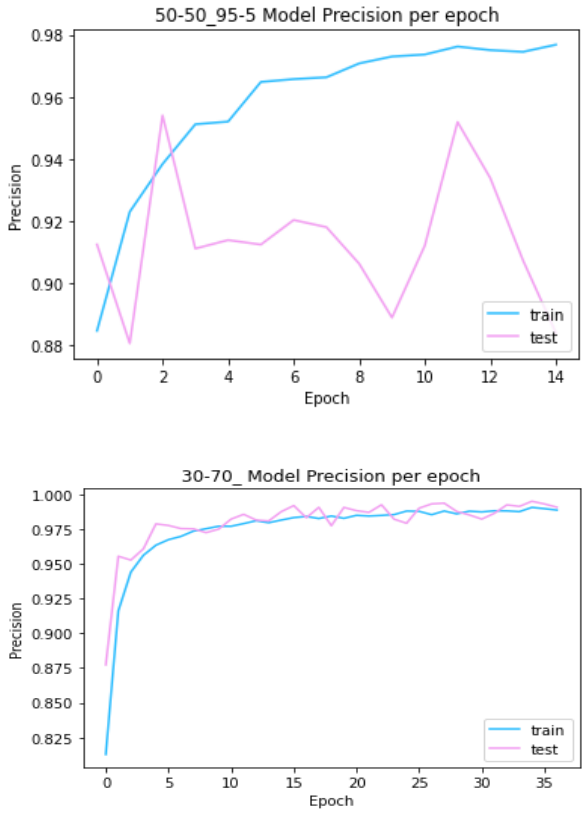


Fig. 33. Comparación de la precisión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Si bien la precisión parece que varía abruptamente, en realidad los cambios son del orden de las centésimas. El mínimo alcanzado es del 0.88, un muy buen valor para la precisión.

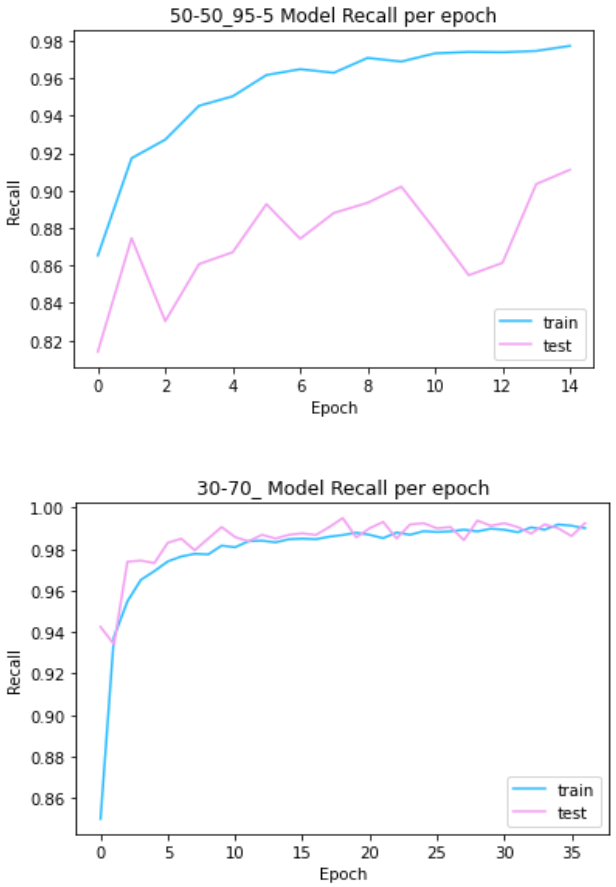
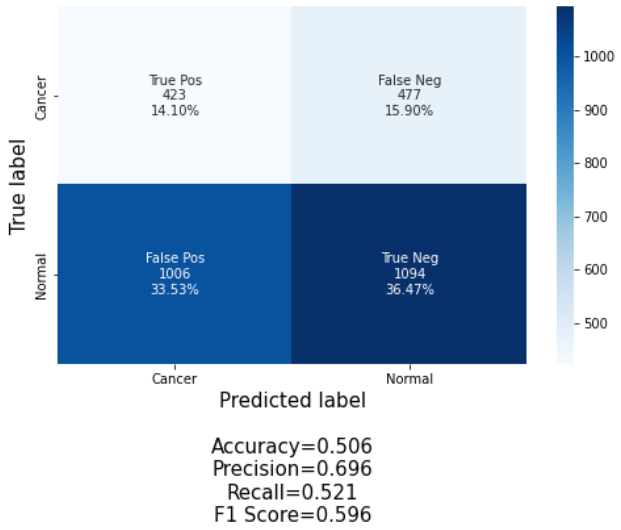


Fig. 34. Comparación de la sensibilidad entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Finalmente, la precisión ya no desciende, sino que se mantiene constante, intuyéndose incluso una pequeña subida hacia el final del entrenamiento, cerca del 0.91.

Resultados de la clasificación



Accuracy=0.506
Precision=0.696
Recall=0.521
F1 Score=0.596

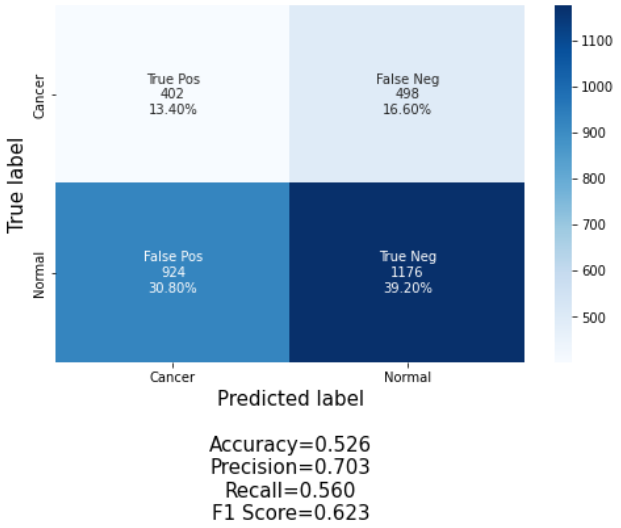


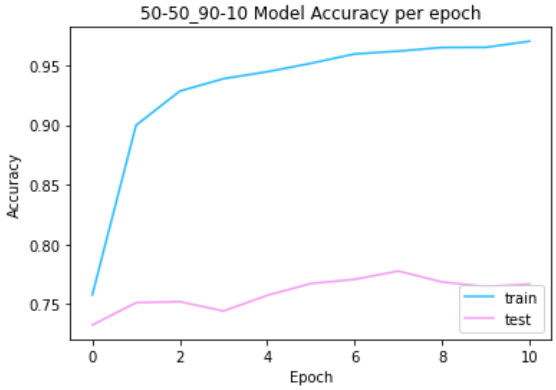
Fig. 35. Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 95% sintéticas – 5% reales (arriba) y el entrenado con un dataset 50% - 50% puramente sintético (abajo). (Imagen propia)

En cuanto a la clasificación, los resultados son pobres, incluso peores que los registrados por el modelo entrenado con el dataset puramente sintético. Lo único positivo es que el porcentaje de falsos negativos desciende ligeramente, pero no es una mejora suficientemente buena como para declarar que este modelo funcione mejor. Veremos a continuación si mejoran los resultados con un porcentaje de imágenes sintéticas ligeramente mayor.

Configuración 50% cáncer – 50% normal, con 90% imágenes sintéticas y 10% reales.

Resultados del entrenamiento

Nos encontramos ante el entrenamiento que mayor diferencia presenta entre las funciones de train y test, es decir, mayor sobreajuste del modelo en esta etapa de entrenamiento. Además, es la que peores valores refleja en todas las métricas, debido a la introducción de más imágenes reales en el entrenamiento que el modelo anterior. Si bien estos resultados de entrenamiento no son determinantes a la hora de medir la calidad del modelo clasificador, esperábamos que fueran mejores, dado que es la configuración de dataset que más prometía.



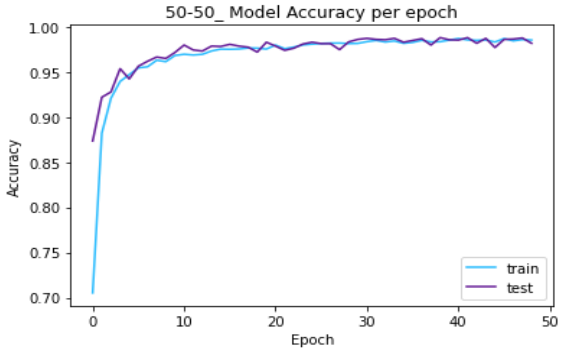
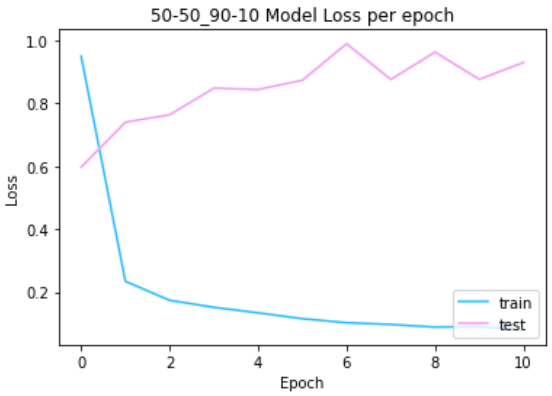


Fig. 36. Comparación de la exactitud entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)



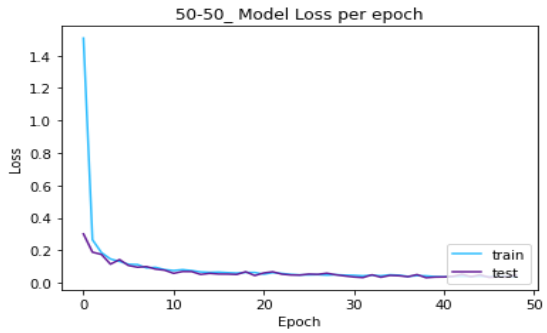


Fig. 37. Comparación de la pérdida entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

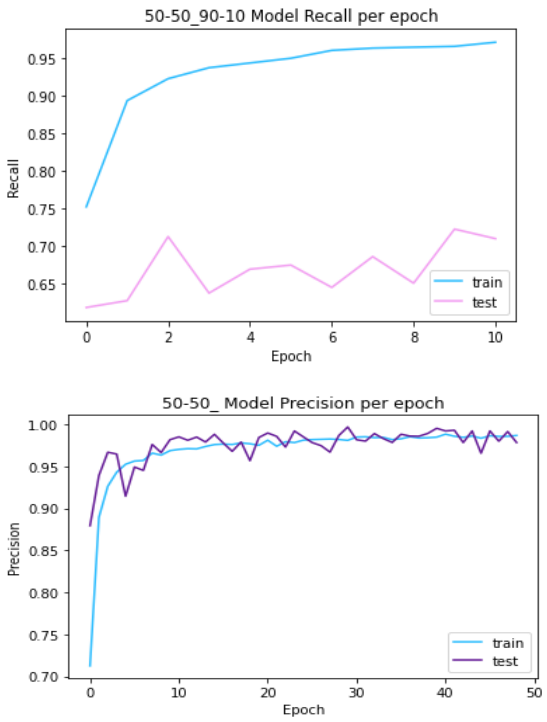


Fig. 38. Comparación de la precisión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

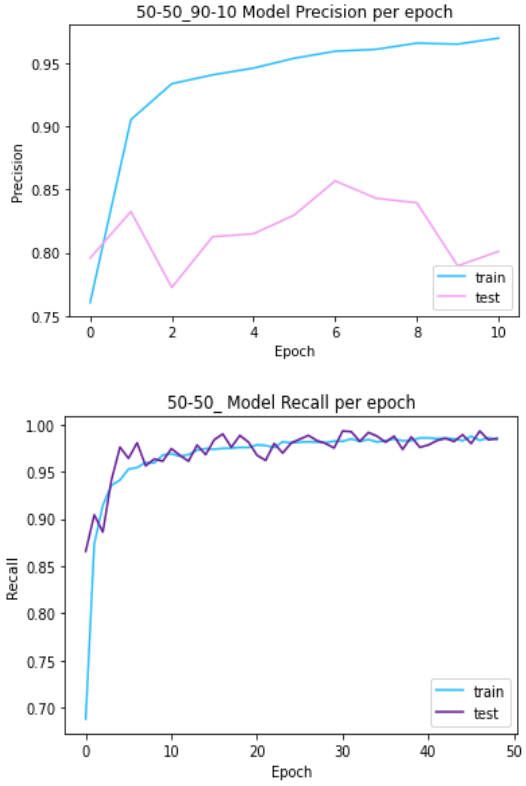


Fig. 39. Comparación de la sensibilidad entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Resultados de la clasificación

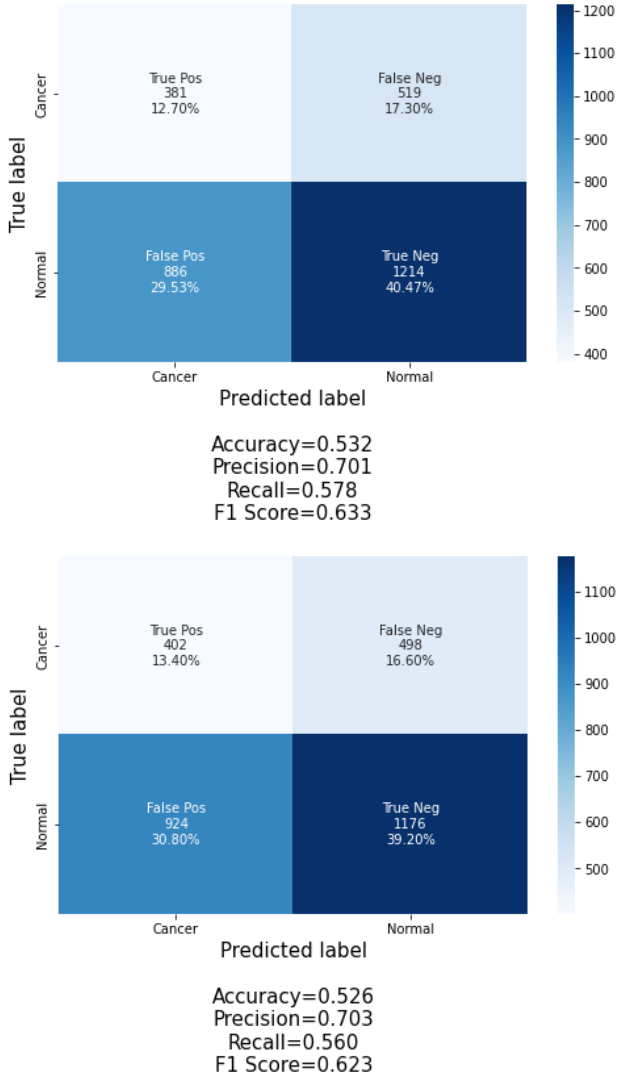


Fig. 40. Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 50% cáncer y 50% normales mezclando 90% sintéticas – 10% reales (arriba) y el entrenado con un dataset 50% - 50% puramente sintético (abajo). (Imagen propia)

Como decíamos, esta configuración se suponía la más adecuada, dado que el dataset con el que se ha entrenado el modelo estaba equilibrado, 50% cáncer y 50% normal, además de contar con más imágenes reales para aportar variedad de características a las imágenes de entrenamiento. Sin embargo, presenta unos resultados poco mejores que los que conseguía el modelo entrenado con una configuración puramente sintética.

Por último, probaremos las configuraciones desequilibradas de 70% cáncer y 30% normales, antes de avanzar hacia las conclusiones del proyecto.

Configuración 70% cáncer – 30% normal, con 95% imágenes sintéticas y 5% reales.

Resultados del entrenamiento

Ésta última configuración presenta relativamente las mismas evoluciones métricas que su homóloga puramente sintética en

cuanto a los resultados de entrenamiento. En general, ninguna métrica desciende a lo largo del tiempo de entrenamiento, siendo la única diferencia con el resto de las configuraciones que presentaban una mezcla de sintéticas con reales. Éste hecho junto con una menor diferencia en las funciones sobre el conjunto de test y de train, indican un mejor rendimiento en el entrenamiento que las configuraciones de 30% cáncer – 70% normal y 50% cáncer – 50% normal. Después comprobaremos si este rendimiento se refleja en los resultados de la clasificación sobre el conjunto de test con imágenes reales.

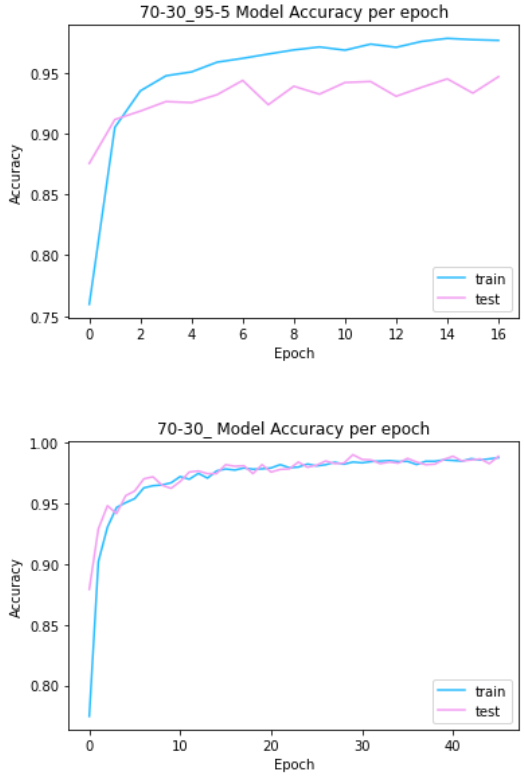
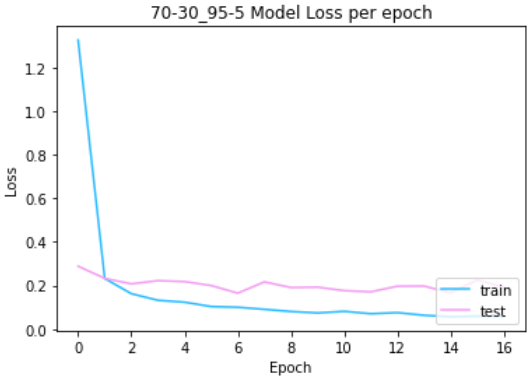


Fig. 41. Comparación de la exactitud entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)



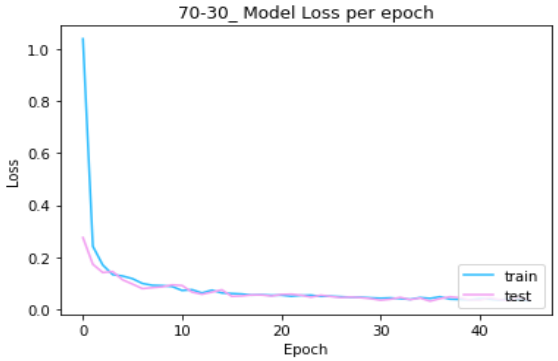


Fig. 42. Comparación de la pérdida entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

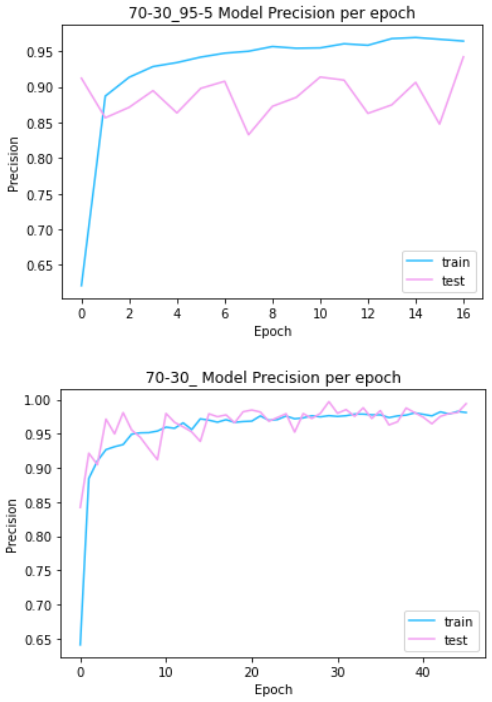


Fig. 43. Comparación de la precisión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

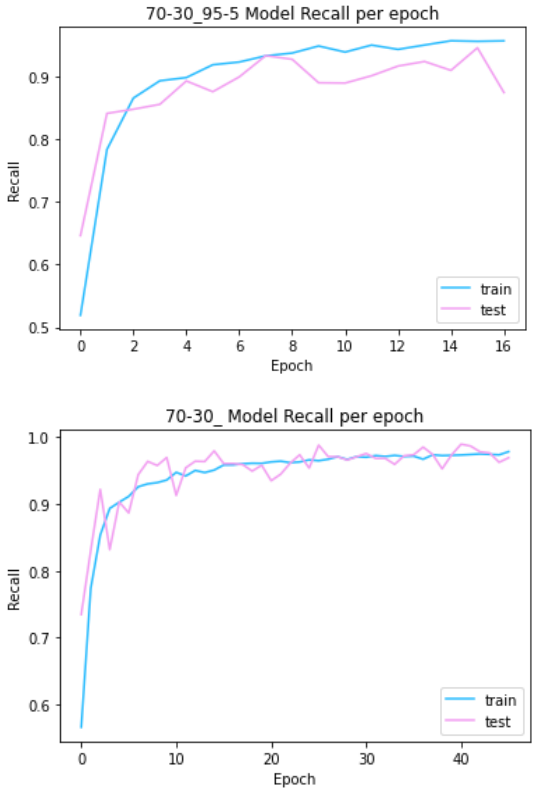
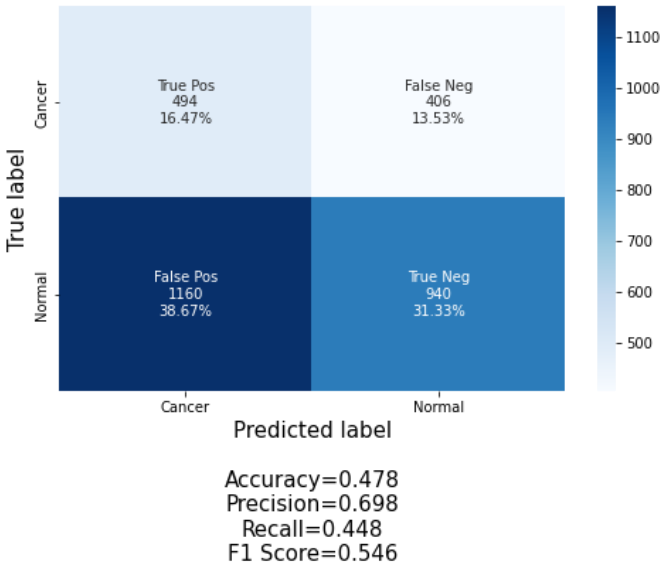


Fig. 44. Comparación de la sensibilidad entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Resultados de la clasificación



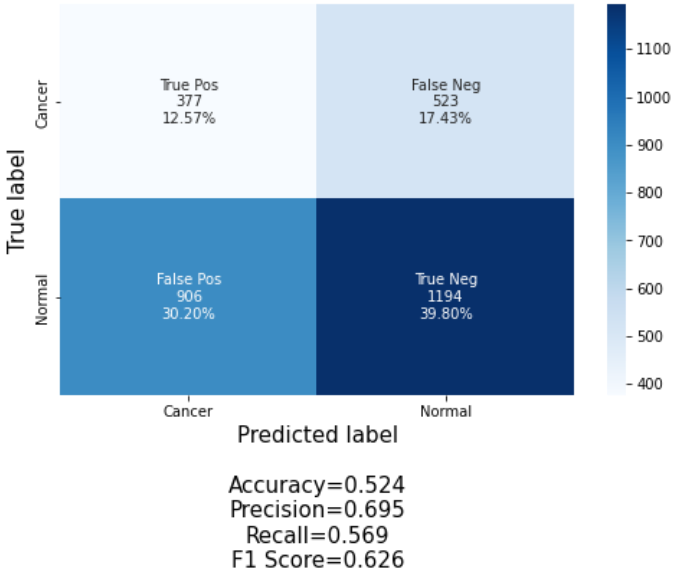


Fig. 45. Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 95% sintéticas – 5% reales (arriba) y el entrenado con un dataset 70% - 30% puramente sintético (abajo). (Imagen propia)

Observamos que todas las métricas en general han bajado considerablemente. Los falsos negativos también han bajado pero no es un aspecto suficientemente relevante como para considerar mejor el modelo entrenado con un dataset desequilibrado en cuanto a imágenes con cáncer.

Configuración 70% cáncer – 30% normal, con 90% imágenes sintéticas y 10% reales.

Resultado del entrenamiento

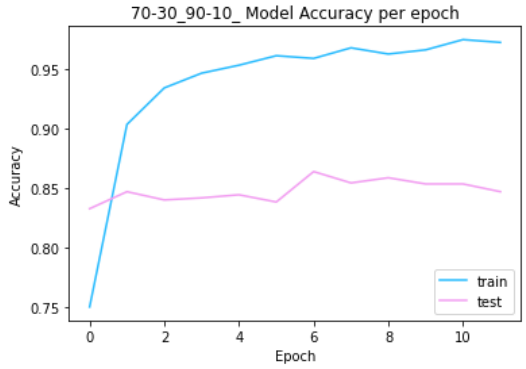


Fig. 45. Comparación de la exactitud entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

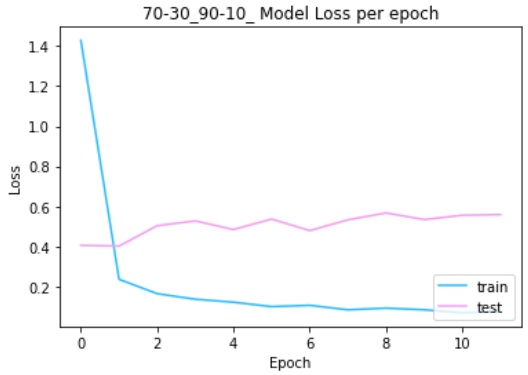


Fig. 46. Comparación de la pérdida entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

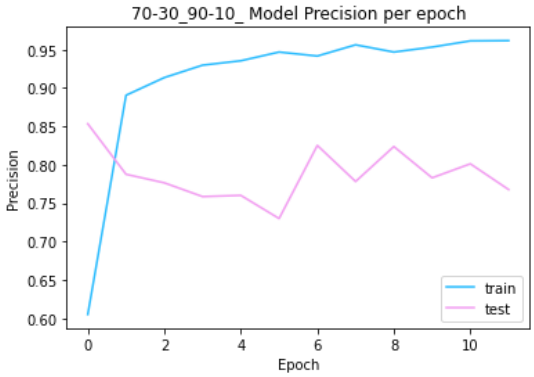


Fig. 47. Comparación de la precisión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

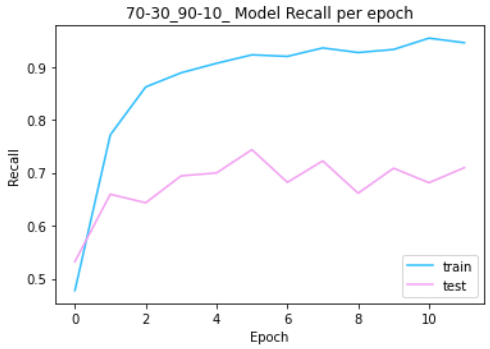


Fig. 48. Comparación de la sensibilidad entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (izq) y el entrenado con un dataset puramente sintético (dcha). (Imagen propia)

Resultado de la clasificación

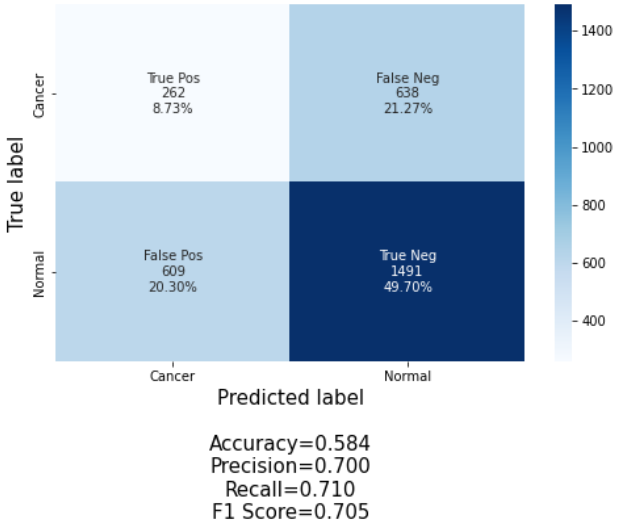


Fig. 49. Comparación de la matriz de confusión entre el modelo entrenado con un dataset de 70% cáncer y 30% normales mezclando 90% sintéticas – 10% reales (arriba) y el entrenado con un dataset 50% - 50% puramente sintético (abajo). (Imagen propia)

VI. Conclusiones

El objetivo de este proyecto era diseñar una herramienta de software barata, eficiente e intuitiva. Esta herramienta debía ser capaz de generar imágenes sintéticas de mamografías de buena calidad, para evitar así el problema de depender de bases de datos públicas, en su mayoría bien custodiadas por las autoridades sanitarias de la sociedad médica internacional, debido al contenido de información sensible de las pacientes.

Creemos firmemente, apoyados en las pruebas aportadas en los capítulos 5 y 6, que nuestra herramienta cumple parte de dichas expectativas: se trata de un software barato de entrenar debido al alojamiento en remoto de Google Colab PRO, es eficiente en cuanto a que consigue generar datasets formados por numerosas mamografías sintéticas en relativo poco tiempo (en el orden minutos, ni siquiera horas), y es intuitiva, el notebook en el que se implementa está comentado línea a línea, tanto en inglés como en español, dos de los idiomas más hablados del mundo.

En cuanto a la calidad de las imágenes, concluimos que no es suficiente para poblar datasets de manera que puedan sustituir a datasets de imágenes reales. La carencia de imágenes en el dataset DDSM elegido para entrenar la herramienta generadora y la falta de potencia de cálculo y memoria en el entorno remoto han supuesto un hándicap demasiado grande para que los resultados presenten una calidad aceptable. Sin embargo, creemos en que una herramienta basada en la que aquí se presenta, entrenada a partir de un dataset más numeroso y de mejor calidad, con herramientas más potentes, podría alcanzar métricas mucho mejores que las conseguidas en nuestro desarrollo.

Líneas de trabajo futuro

En cuanto a las líneas de futura investigación, como hemos señalado, tuvimos que dejar varios apartados de lado para conseguir mejores resultados con los medios de los que disponíamos. Si bien es verdad que el objetivo era demostrar que con una pequeña base de datos como DDSM con solo alrededor de 6300 imágenes, se podía desarrollar una herramienta lo suficiente potente, usando tecnologías de Aprendizaje

Automático y Profundo como las DCGAN, en concreto, la StyleGAN 2 ADA, existen otros datasets (bajo licencia de investigación, obviamente) más numerosos y de mayor calidad. Uno de estos ejemplos es el dataset DukeDBT (Mateusz, y otros, 2020), un dataset de tomografías digitales (de mucha más calidad que las mamografías), con aproximadamente 1500 GB de imágenes anotadas. Un dataset como este necesitaría de una potencia de cálculo extremadamente grande para crear una herramienta como la propuesta, por lo que una futura línea de investigación sería usar datasets más grandes y de mayor calidad para el desarrollo de modelos de generación de mamografías/tomografías sintéticas que arrojen mejores resultados que los aquí propuestos.

Otro de los puntos en los que se podría profundizar es en el uso de otros modelos generadores más actuales, de hecho, los investigadores de nVIDIA que desarrollaron y publicaron StyleGAN acaban de publicar un nuevo modelo de generación de imágenes sintéticas mucho más potente que StyleGAN, Alias FreeGAN (Karras, y otros, 2021), que resuelve ciertos artefactos resultado de carencias en el entrenamiento del generador de StyleGAN.

Para concluir, una de las ambiciones que nos planteamos en el inicio del proyecto fue la de generar mamografías sintéticas de gran calidad y, a la vez, mantener una resolución elevada de dichas imágenes. La resolución de las imágenes generadas por este tipo de modelos suele ser relativamente pequeña, en torno a 128x128px o incluso 256x256px. El objetivo inicial era llegar a una resolución de 1024x1024px, incluso llegamos a entrenar el modelo para que generara dichas imágenes, pero no cumplían con la calidad adecuada. En este tipo de redes, existe un compromiso entre la resolución y la calidad de las imágenes generadas, ya que el generador de las DCGAN no puede "atender" a ambas especificaciones. Aun así, conseguimos generar imágenes de 512x512 con calidad puntuada con cerca de un 39 de FID, un hito bastante relevante. Por lo tanto, la línea de investigación que de este punto se propone, es la generación de imágenes de mayor resolución, manteniendo la misma calidad que la conseguida en este proyecto, bien haciendo uso de un entorno con mayor potencia de cálculo y memoria, o bien mediante el uso de los nuevos e innovadores modelos de DCGAN que se publicarán en los próximos años sin ningún atisbo de duda.

APÉNDICES

Apéndice I: Arquitectura del modelo original preentrenado de VGG16:

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

_____

flatten (Flatten) (None, 25088) 0

_____

_____

fc1 (Dense) (None, 4096) 102764544

_____

_____

fc2 (Dense) (None, 4096) 16781312

_____

_____

predictions (Dense) (None, 1000) 4097000

==========

==========

Total params: 138,357,544

Trainable params: 138,357,544

Non-trainable params: 0

REFERENCIAS

[1] Radford, A., Metz, L., & Chintala, S. (19 de 11 de 2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Recuperado el 05 de 05 de 2021, de https://arxiv.org: https://arxiv.org/abs/1511.06434

[2] Goodfellow, I. J., Pouget, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Networks. Cornell University, 9. Obtenido de https://arxiv.org/abs/1406.2661

[3] Xu, Z., Wilber, M., Fang, C., Hertzmann, A., & Jin, H. (2019). Learning from Multi-domain Artistic Images for Arbitrary Style Transfer. Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/pdf/1805.09987.pdf

[4] Isola, P., Zhu, J.-Y., Zhu, T., & Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1611.07004

[5] Krishna Goti, S., & Ma, J. (14 de Agosto de 2018). Text-to-Image-to-Text Translation using Cycle Consistent Adversarial Networks. Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1808.04538

[6] Radford, A., Metz, L., & Chintala, S. (19 de 11 de 2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Recuperado el 05 de 05 de 2021, de https://arxiv.org: https://arxiv.org/abs/1511.06434

[7] Yuan, C., Huang, Y.-C., & Tsai, C.-H. (22 de 06 de 2020). Efficient text generation of user-defined topic using generative adversarial networks. Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/2006.12005

[8] Payne, C. (25 de 04 de 2019). MuseNet. Recuperado el 05 de 05 de 2021, de https://www.openai.com/blog/musenet/

[9] Alyafi, B., Diaz, O., & Martí, R. (04 de 09 de 2019). DCGANs for Realistic Breast Mass Augmentation in X-ray Mammography. Recuperado el 05 de 05 de 2021, de https://arxiv.org/: https://arxiv.org/abs/1909.02062

[10] Cha, K. H., Petrick, N., Pezeshk, A., Graff, C. G., Sharma, D., Badal, A., & Sahiner, B. (22 de 11 de 2019). Evaluation of data augmentation via synthetic images for improved breast mass detection on mammograms using deep learning. Recuperado el 05 de 05 de 2021, de https://www.ncbi.nlm.nih.gov/: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6872953/

[11] Wu, E., Wu, K., Cox, D., & Lotter, W. (21 de 07 de 2018). Conditional Infilling GANs for Data Augmentation in Mammogram Classification. Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/pdf/1807.08093.pdf

[12] Dissanayake Lekamlage, C., Afzal, F., Westerberg, E., & Cheddad, A. (2020). Mini-DDSM: Mammography-based Automatic Age Estimation. DMIP '20: 2020 3rd International Conference on Digital Medicine and Image Processing, (págs. 1-6).

[13] Schultz, D. (s.f.). dataset-tools. Obtenido de GitHub: https://github.com/dvschultz

[14] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (12 de 01 de 2018). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. Recuperado el 05 de 05 de 2021, de arxiv.org: https://arxiv.org/abs/1706.08500

[15] Simonyan, K., & Zisserman, A. (2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. iCLR 2015.

[16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2 de Diciembre de 2015). Rethinking the Inception Architecture for Computer Vision. Obtenido de arXiv.org: https://arxiv.org/abs/1512.00567

[17] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-fei, L. (2009). ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition.

[18] Trimarchi, D. (21 de Noviembre de 2019). confusion_matrix. Obtenido de GitHub: https://github.com/DTrimarchi10/confusion_matrix