Universidad ORT Uruguay

Facultad de Ingeniería

MODELOS GENERATIVOS DE DIFUSIÓN PARA LA CARACTERIZACIÓN DE LESIONES DERMATOLÓGICAS EN IMAGENOLOGÍA MÉDICA

Entregado como requisito para la obtención del título de Máster en Big Data

Ignacio Aristimuño - 192584
Daniel Gallino - 163568

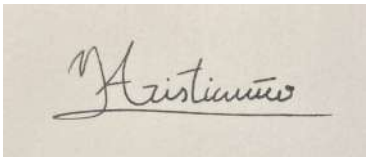
Tutor: Sergio Yovine

2023

DECLARACIÓN DE AUTORÍA

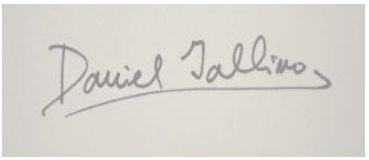
Nosotros, Ignacio Aristimuño y Daniel Gallino, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el trabajo final del Máster en Big Data;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Ignacio Aristimuño

11/09/2023



Daniel Gallino

11/09/2023

AGRADECIMIENTOS

Queremos agradecer a nuestras familias y amigos que han sido un apoyo incondicional a lo largo de este proceso.

También agradecer a nuestro tutor Sergio Yovine por habernos asesorado y guiado en el trascurso de proyecto.

A la comunidad open-source, tanto a sus contribuidores individuales como a empresas que fomentan esta ideología, gracias a la cuales se han disponibilzado una variedad de modelos de uso libre, así como tutoriales y otros recursos que permitieron el desarrollo de este proyecto dentro de un marco temporal razonable.

Por último, agradecer a la universidad ORT y a todos aquellos que colaboraron brindando información útil para el desarrollo de esta tesis.

ABSTRACT

El presente informe aborda las técnicas generativas en inteligencia artificial, centrándose en los modelos de difusión dentro del campo de la imagenología médica, y en particular, en su aplicación para la caracterización de lesiones dermatológicas.

El objetivo principal consiste en la generación de imágenes médicas de calidad comparable con las originales, de manera de poder afirmar si con las técnicas existentes previas a la finalización de este proyecto, se tienen suficientes elementos para lograr la calidad deseada. Si bien existen diversas aplicaciones en las que los modelos de difusión podrían ayudar en este campo de estudio, como la reconstrucción 3D de regiones del cuerpo a partir de un conjunto de imágenes 2D, la predicción de la evolución de posibles tumores o anomalías, y el aumento de resolución de las imágenes obtenidas con scanners industriales; en este proyecto se abordan las limitaciones en la implementación de modelos de aprendizaje profundo en imágenes médicas, que recaen principalmente en la escasez de datos, el desequilibrio de los mismos y la falta de datos etiquetados en poblaciones subrepresentadas.

En el contexto de la imagenología médica, las restricciones en los datos surgen debido a la alta especialización de las imágenes, la privacidad de los pacientes y la falta de representación de ciertas poblaciones. Estos desafíos hacen que las técnicas generativas sean una potencial solución al generar datos sintéticos de alta calidad para superar la escasez y heterogeneidad de los datos reales, de manera de asistir en el desarrollo de modelos de clasificación, segmentación, detección, entre otros.

Hasta aproximadamente el año 2020, las técnicas disponibles para la generación de imágenes no cumplían con los estándares de calidad y realismo requeridos para ser utilizados en la imagenología médica. Las redes generativas antagónicas (GANs), conocidas por su capacidad para generar imágenes realistas en ese entonces, enfrentaban problemas de estabilidad durante el entrenamiento, dificultando la investigación en este campo. Por esta razón, es que se enfoca este estudio en una aplicación práctica de los modelos de difusión de forma de evaluar la calidad de las imágenes generadas y su impacto en el desempeño de modelos de aprendizaje profundo.

Se realiza una investigación del estado del arte con respecto a los modelos de difusión, seleccionando aquellas técnicas y modelos que presenten un mayor potencial para su aplicación dentro de la caracterización de lesiones dermatológicas. Se implementan distintas técnicas de fine-tuning sobre modelos de difusión pre-entrenados como Stable Diffusion para lograr mejorar la calidad de las imágenes generadas, concluyendo que la calidad de las imágenes generadas es muy buena al compararlas con imágenes reales de un dataset público de referencia. Por último, se entrena un modelo de clasificación de lesiones de piel con imágenes reales, y se utilizan imágenes generadas tanto como técnica de data augmentation como también como técnica de upsampling. Se logró una mejora del 5.5% y un 11% del accuracy respectivamente, y un 4.5% y 8.5% del recall promedio por clase con respecto al clasificador entrenado únicamente con imágenes reales, concluyendo que estas técnicas generativas pueden resultar útiles para crear soluciones de aprendizaje profundo cuando hay escasez o desbalance de datos.

PALABRAS CLAVE

Modelos de difusión, modelos generativos, IA generativa, Stable Diffusion, Low-Rank Adaptation (LoRA), Textual Inversion, DreamBooth, imagenología médica, text-to-image, datos sintéticos.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

En la presente sección se proporciona una visión general de los objetivos, desafíos y enfoques centrales abordados en este informe de investigación. Se presenta un contexto fundamental sobre la importancia de las técnicas generativas en el campo de la inteligencia artificial, con un énfasis particular en los modelos de difusión. Además, se introduce la aplicación específica en el campo de la imagenología médica, resaltando los objetivos, limitaciones y las motivaciones que impulsaron este estudio.

1.1 Motivación

En los últimos años, la inteligencia artificial (IA) ha experimentado una evolución notable, extendiendo su influencia en diversos sectores, siendo uno de ellos la medicina. Esta transformación ha llevado a una mayor adopción de la IA en distintas organizaciones, incluyendo hospitales de todo el mundo. De hecho, algunos centros médicos ya han integrado con éxito soluciones basadas en inteligencia artificial en sus prácticas diarias.

Estos avances evidencian cómo la IA ha dejado de ser una mera promesa tecnológica para convertirse en una herramienta tangible y eficaz que está redefiniendo la forma en que los profesionales de la salud abordan el diagnóstico, el tratamiento y la atención al paciente. Un ejemplo claro del impacto de este avance tecnológico, se puede observar al leer el reporte de la utilización de IA en el departamento de radiología de NYU Langone Health [1], donde se comenta que herramientas de IA han incrementado la habilidad de los radiologistas de detectar enfermedades en un 37%, y han reducido en un 27% la necesidad de extraer muestras de tejido o realizar biopsias para confirmar tumores.

Sin embargo, el desarrollo y entrenamiento de estos modelos de aprendizaje automático no es una tarea simple. Requiere de un volumen considerable de datos, que no solo deben ser abundantes, sino también diversos, abarcando múltiples pacientes con distintas características clínicas y contextos. Además, la calidad y precisión son cruciales, lo que exige que los datos estén etiquetados por expertos en el dominio médico para garantizar su validez. Esta necesidad de datos enriquecidos y etiquetados ha resultado ser un desafío significativo en la adopción acelerada de modelos de aprendizaje automático en medicina. La recopilación y curación de conjuntos de datos médicos robustos es un proceso complejo y trabajoso, que a menudo implica la colaboración interdisciplinaria y la superación de obstáculos éticos y de privacidad. Esta realidad, aunque desafiante, subraya la importancia crítica de abordar la escasez y diversidad de datos en la investigación y aplicación de modelos de IA en la medicina, a fin de lograr un impacto positivo en la atención al paciente.

Debido a la presencia de estos obstáculos y desafíos es que se genera un profundo interés en la temática de este proyecto. La posibilidad de utilizar técnicas generativas y modelos de difusión para enfrentar estos retos y avanzar un paso hacia elevar la calidad de la atención médica es lo que motiva la definición de la temática de este proyecto.

1.2 Situación actual

La imagenología médica se encuentra experimentando un proceso de transformación digital hace ya un par de años, debido principalmente a los avances tecnológicos en el campo de la informática, datos y la inteligencia artificial. En particular, la aplicación de técnicas de aprendizaje profundo en el procesamiento y generación de imágenes médicas ha abierto nuevas oportunidades para mejorar la precisión y eficiencia en el diagnóstico y el tratamiento de diversas afecciones, así como una herramienta de apoyo para el personal de diversos centros hospitalarios.

Aún no se encuentran muchas publicaciones sobre aplicaciones prácticas de soluciones que utilicen aprendizaje profundo para el análisis de imágenes médicas, aunque sí se aplican técnicas de procesamiento de imágenes como estudios más específicos. En este sentido, los modelos de difusión se han ido convirtiendo en área de investigación cada vez más estudiada para su posible aplicación en la imagenología médica.

Una aplicación estudiada consiste en la reconstrucción en tres dimensiones de algún área de interés de los pacientes a partir de una cantidad limitada de imágenes en dos dimensiones [2] [3]. También han sido estudiadas aplicaciones para aumentar la resolución de imágenes [4], debido a que en la investigación médica se tienen imágenes de alta resolución, facilitando la interpretación o detección de posibles problemas en algunos tejidos. En cambio, en los centros médicos se tienen imágenes de menor resolución. Al obtener imágenes con mejor resolución podría facilitar su estudio y comparar resultados con los resultados académicos.

Por otro lado, han habido hallazgos en cuanto a la utilización de modelos de difusión para realizar la segmentación de imágenes médicas, condicionando con imágenes de entrada [5] [6]. Por último, las últimas dos semanas anteriores a la entrega del presente informe, se publicaron algunos artículos haciendo referencia a la generación de imágenes médicas como técnica de data augmentation para mejorar la performance de clasificadores [7], uno de los objetivos principales de esta investigación, que se mencionará con mayor detalle en la siguiente subsección.

Algunas publicaciones hacen referencia a la mejora de calidad de imágenes médicas al remover ruido de las mismas a través de la utilización de modelos de difusión [8] [9] [10].

Tomando en consideración que los modelos de difusión se encuentran generando imágenes de alta calidad hace menos de dos años, se espera que la cantidad de artículos y grupos de investigación que realicen experimentos sobre la aplicación práctica de modelos de difusión, y en particular para la imagenología médica, aumenten en el corto plazo.

1.3 Objetivos y alcance

El objetivo central de este proyecto es explorar cómo las técnicas generativas, en particular los modelos de difusión, pueden abordar los desafíos de la escasez y desbalance de datos. Por otro lado, se busca evaluar cómo estas imágenes generadas pueden impactar la calidad y la utilidad clínica, mejorando el desempeño de modelos de aprendizaje automático. A través

de la investigación y la implementación, se aspira avanzar en soluciones para la imagenología médica y su aplicación práctica en un caso de estudio específico.

El presente trabajo busca responder a preguntas fundamentales en la intersección entre la inteligencia artificial y la medicina, como por ejemplo:

- ¿Cómo mitigar el efecto de la falta de datos en la imagenología médica?
- ¿Qué impacto tienen las técnicas generativas para mejorar el rendimiento de los modelos de aprendizaje?
- ¿En qué medida pueden las técnicas generativas mejorar la representación de poblaciones poco frecuentes en los datos médicos?
- ¿Cómo se comparan las imágenes generadas por técnicas generativas con las imágenes médicas reales en términos de calidad y utilidad diagnóstica?
- ¿Cómo superar los desafíos técnicos al implementar datos sintéticos en entornos médicos reales?

1.4 Estructura del informe

En la sección de Marco Teórico (Sección 2), se exponen los conceptos clave para comprender la teoría detrás de las redes generativas, mostrando algunas arquitecturas como *Variational AutoEncoders* (VAEs) y *Generative Adversarial Networks* (GANs), profundizando sobre los modelos de difusión. Se profundiza sobre cómo funcionan los modelos de difusión, cómo se entrenan, y las arquitecturas del estado del arte actual como *Stable Diffusion*. Por último, se presentan alternativas sobre cómo realizar un fine-tuning para enseñarle a los modelos de difusión pre-entrenados a generar imágenes sobre un concepto nuevo.

En la sección de Metodología (Sección 3), se describen los pasos detallados, desde la investigación del estado del arte hasta la selección y aplicación de las técnicas generativas en un estudio de caso específico en el ámbito de la imagenología médica. También se abordan aspectos cruciales como la selección de datos, la implementación práctica, la optimización y la evaluación de resultados.

En la sección de Implementación (Sección 4) se centra en describir los pasos seguidos para la implementación de las técnicas de *fine-tuning* de los modelos de difusión y los resultados obtenidos, siendo implementados en un caso de prueba en una primera instancia, y luego en casos de uso dentro del campo de la imagenología médica, resaltando desafíos y consideraciones relevantes en el proceso.

En la sección de Conclusiones (Sección 5) se resumen los resultados más relevantes, haciendo referencia a las preguntas y objetivos clave que guían el desarrollo del presente proyecto. Se resaltan las contribuciones significativas de la investigación, haciendo énfasis en la relevancia de las técnicas generativas en la mejora de la imagenología médica, discutiendo las limitaciones identificadas y proponiendo líneas para la investigación futura que puedan enriquecer aún más este campo.

2. MARCO TEÓRICO

Dentro del Marco Teórico de este proyecto se establecen las bases teóricas fundamentales para comprender cómo funcionan las redes generativas, mencionando las arquitecturas más relevantes en la actualidad. En particular se profundiza sobre el diseño y entrenamiento a más bajo nivel de los modelos de difusión, incluyendo alternativas para su aplicación dentro del contexto de la imagenología médica.

2.1 Introducción a las redes generativas

Las redes generativas son un conjunto de algoritmos y técnicas dentro del campo del aprendizaje profundo que tienen como objetivo principal la generación de datos realistas, siguiendo una distribución similar a los datos con los que fueron entrenadas.

Son herramientas poderosas para la síntesis de datos, permitiendo la generación de imágenes, textos, audio, y hasta utilizando enfoques multimodales, en donde se puede guiar la generación de imágenes con texto, con otras imágenes, generando audios a partir de una descripción de la salida que se desea generar, entre otros.

Algunos de los modelos y soluciones más conocidas en la actualidad son:

- **ChatGPT**: un chatbot desarrollado a partir del entrenamiento de modelos grandes de lenguaje basados en la arquitectura *decoder-only* de los transformers, al cual se le agrega la capacidad de persistir la memoria del chat utilizando formateo de las prompts. A su vez se optimiza para el lenguaje conversacional con técnicas de aprendizaje por refuerzos como *Reinforcement Learning from Human Feedback* (RLHF) y *Proximal Policy Optimization* (PPO). [11]

- **Audiocraft**: consiste en una familia de tres modelos capaces de producir audio de alta calidad y larga duración consistentes. MusicGen fue entrenado para la generación de música a partir de prompts de texto, AudioGen entrenado para generar efectos de sonido a partir de prompts de texto y EnCodec que es utilizado para realizar un encoding y decoding de la información proveniente de audios para mayor eficiencia de los modelos. [12]

- **Midjourney**: es un laboratorio independiente que se dedica a la investigación de la inteligencia artificial, pero también es el nombre que recibe su software de IA, que sirve para crear imágenes a partir de texto. [13]

A continuación se muestran unos ejemplos de la utilización de ChatGPT y Midjourney.

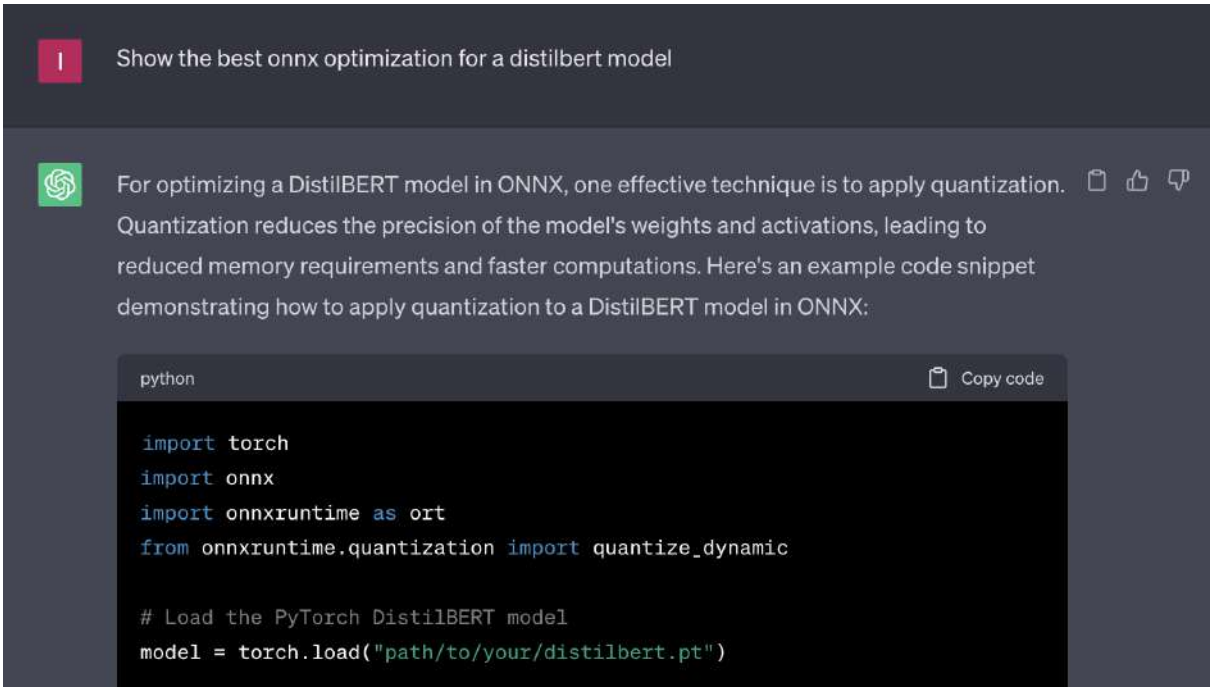


Figura 2.1 - Ejemplo de la utilización de ChatGPT, herramienta generativa de texto. El usuario (ícono cuadrado rosado) envía una prompt de texto para la cual quiere una respuesta del modelo y el modelo (ícono verde, con el logo de ChatGPT) genera una respuesta a partir de esa prompt y de la conversación que lleva teniendo el usuario con el chat. [11]

Figura 2.2 - Ejemplo de la utilización de Midjourney, obteniéndose la imagen a través de la siguiente prompt: "A labrador dog in a spacesuit on the moon, Cubism". [13]

2.2 Aplicaciones en la medicina

En el ámbito de la medicina, la inteligencia artificial generativa tiene hoy en día un impacto positivo en varias áreas, pudiendo tener un potencial de crecimiento a medida que aumente su adopción. Algunos de los casos de uso en los que se están realizando mayores esfuerzos de investigación se centran en:

- **Descubrimiento y desarrollo de medicamentos:** puede facilitar y acelerar el proceso de descubrimiento y desarrollo de medicamentos al identificar candidatos potenciales y evaluar su efectividad mediante simulaciones computacionales antes de las pruebas clínicas en animales y humanos.

- **Medicina personalizada:** se pueden crear planes de tratamiento personalizados al considerar el historial médico, síntomas y otros factores de los pacientes. Aunque todavía no hay casos de estudio reales, sí existen antecedentes de la utilización de *transformers* para predicción de futuros reingresos a salas de emergencias, tanto de la razón de la visita cómo de la fecha de reingreso. Se podrían adaptar tratamientos a cada paciente de manera personalizada considerando su historial.

- **Mejora de imágenes médicas:** la IA generativa podría mejorar la precisión y eficiencia de técnicas de aprendizaje automático en combinación con imágenes médicas como tomografías, resonancias magnéticas, entre otros. Estos modelos de aprendizaje automático pueden identificar automáticamente anomalías en las imágenes y alertar a los médicos sobre posibles problemas.

- **Concientización al público:** existen organizaciones cuyo objetivo es brindarle al público herramientas que permitan conocer el impacto de tener un determinado estilo de vida sobre su salud, a través de diversas herramientas que utilizan IA generativa. Un ejemplo es el caso de ageing, en donde, a través de modelos como GANs y modelos de difusión, se puede mostrar a las personas cómo se vería su rostro y boca luego de fumar 10 años, o una imagen de su cuerpo al llevar su dieta actual y no una más sana por un período de tiempo determinado.

2.3 Autoencoders Variacionales

Los autoencoders variacionales (VAEs, por sus siglas en inglés: *Variational AutoEncoders*) se basan en los autoencoders tradicionales con una alteración que los vuelve variaciones. Los autoencoders tradicionales surgen como modelos para la reducción de la dimensionalidad, aprendiendo a comprimir los datos de entrada a un espacio latente.

Están compuestos por dos elementos: un encoder y un decoder, siendo el encoder el encargado de tomar las imágenes de entrada y reducirlas hasta el tamaño del espacio latente que se desee, también llamado cuello de botella, y luego el decoder se encarga de tratar de recuperar la información perdida al realizar dicha compresión.

2.3.1 Aspecto variacional

El autoencoder tradicional se entrena utilizando imágenes de entrada que se intentan reconstruir luego de pasar por el cuello de botella de la red, y luego comparando la salida generada con la salida esperada, que es justamente la imagen de entrada. Por esta razón es que resultan muy prácticos, ya que no se precisa de etiquetado de datos per se.

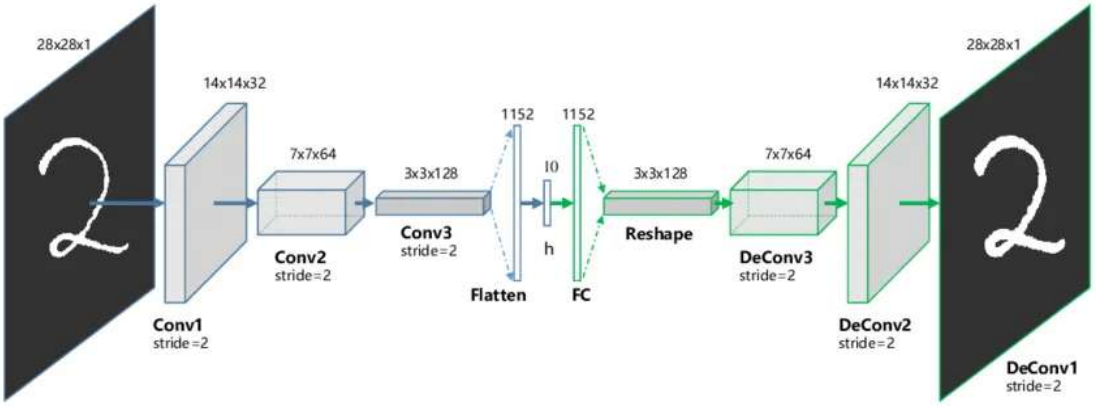


Figura 2.3 - Representación del proceso de entrenamiento y generación de imágenes con VAEs. [14]

Sin embargo, los autoencoders tradicionales no nos habilitan a generar nuevo contenido, sino que su utilidad reside en la compresión de información utilizando el encoder, pudiendo utilizar esta versión comprimida de los datos para hacer retrieval o algunas transformaciones, y luego volver del espacio latente al espacio de las imágenes utilizando el decoder. Por esta razón es que se introduce la idea de utilizar distribuciones en vez de puntos en el espacio latente, expuesto en el artículo "Auto-Encoding Variational Bayes" presentado por D. P. Kingma y M. Welling [15].

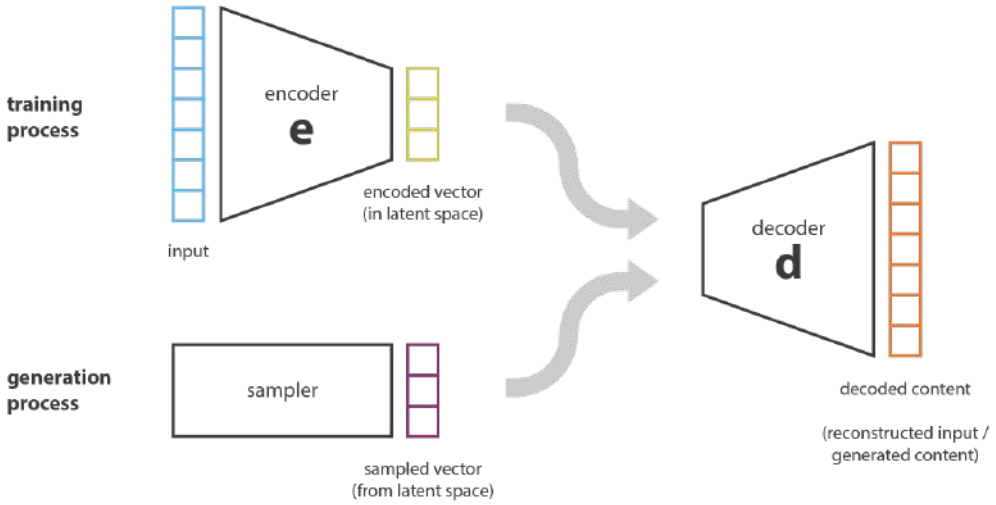


Figura 2.4 - Representación del proceso de entrenamiento y generación de imágenes con VAEs. [16]

A grandes rasgos, en vez de codificar los datos en un vector del espacio latente (punto en el espacio), se comprime para obtener una distribución probabilística, que luego se tomará una muestra de la misma para reconstruir la imagen. Al reconstruir a partir de un muestreo de una distribución, estamos agregando cierta variabilidad a la reconstrucción. Esto induce cierta

regularización, permitiendo que sea posible tomar diversos puntos dentro del espacio latente para generar imágenes, aunque dichos puntos no hayan sido vistos durante el entrenamiento.

Es por esto que, incorporando esta modificación, es posible generar nuevos datos a partir de muestrear de manera aleatoria distribuciones del espacio latente (cuello de botella), y luego utilizando el decoder para pasar del espacio latente al espacio de imágenes.

2.3.2 Principales aspectos

Previo a la utilización de estos tipos de modelos, es necesario conocer las ventajas y desventajas que estos presentan:

- **Baja fidelidad:** la función de pérdida comúnmente surge de la comparación de píxeles entre la imagen de entrada y la generada. Esto puede ser subóptimo, ya que, a modo de ejemplo, en patrones que se intercalan píxeles con valores muy distintos como el cabello de una persona, se puede generar una pérdida grande si la imagen generada se encuentra levemente desplazada con respecto a la de entrada. Por otro lado, dos imágenes que no son muy similares entre sí pueden tener cierta intersección dentro de las distribuciones de sus espacios latentes, presentando problemas al generar imágenes a partir de estos.

- **Gran diversidad:** la función de maximización de la verosimilitud fuerza a cubrir las distintas clases del dataset provisto, evitando problemas como el mode collapse que se presentan en las GANs.

- **Entrenamiento simple:** son sencillas de entrenar y su función de pérdida es rastreable, pudiendo visualizarse cómo se va desarrollando el entrenamiento de la red.

- **Utilidad:** si bien se presentan en el marco de la generación de imágenes, el encoder entrenado puede utilizarse para otras tareas como la generación de embeddings a partir de imágenes de entrada.

2.3.3 Casos de uso

Por otro lado, debemos conocer los casos de uso más comunes de los VAEs, que en general son utilizados para tareas de:

- **Reducción de la dimensionalidad:** si bien los autoencoders son buenos para comprimir la información de entrada, el efecto que se logra de regularización con los VAEs puede llegar a tener un impacto positivo para la obtención de estas representaciones del espacio latente. En particular, se pueden utilizar para distintas tareas como retrieval, ranking y hasta para ser utilizados por otro modelo.

- **Remoción de ruido:** en varios contextos, las imágenes que se tienen de entrada no tienen la calidad deseada y el ruido presente en las mismas puede generar un aprendizaje subóptimo de modelos de aprendizaje automático. Los VAEs pueden ser

utilizados para remover este ruido, usando imágenes sin ruido como entrada, a las que se le agregan ruido (similar al que se querría quitar en la realidad), y luego comparando la salida con las imágenes originales sin ruido.

- **Detección de anomalías y datos atípicos:** es posible entrenar VAEs en base a datos "estándar" o "comunes" dentro de cierto contexto o caso de uso. Luego, al realizar inferencias sobre nuevos datos, se reconstruye siguiendo los patrones aprendidos (ej: reconstrucción de imágenes "comunes") y se calcula una puntuación de anomalía contrastando la imagen reconstruida contra la original, comparando la diferencia que existe entre ambas. El planteo recae en que, una puntuación alta, se debería dar porque patrones u objetos presentes en la imagen de entrada no fueron vistos durante el entrenamiento, por ende, podría tratarse de una anomalía.

- **Super resolución:** consiste en aumentar la resolución de imágenes a través de entrenar un VAE en donde la salida del decoder es de mayor dimensión que la entrada al *encoder*.

2.4 Redes Generativas Antagónicas

Las Redes Generativas Antagónicas o Redes Generativas Adversarias (GANs, por sus siglas en inglés), inventadas en 2014 por Ian Goodfellow junto con su equipo de investigadores de la Universidad de Montreal [17], con el objetivo de generar imágenes más realistas. Hasta ese momento, el enfoque con mejores resultados para la síntesis de imágenes eran las VAEs. Si bien logran generar una gran diversidad de imágenes y el tiempo de generación es rápido, no se obtenían resultados realistas de alta calidad.

Es por esto que se propone un nuevo enfoque en el cual se entrena una red de generación de imágenes a partir de la construcción de dos redes que entrenan de manera simultánea: un generador y un discriminador, buscando no solo tener una red que logre generar imágenes, sino tener otra red que vaya corrigiendo la salida de esta generación para que cada vez sean más similares a imágenes reales. Este enfoque se puede considerar dentro de la rama de teoría de juegos como un juego de suma cero, en donde la ganancia de uno de los agentes, es la pérdida del otro.

2.4.1 Arquitectura

El generador toma como entrada un muestreo de ruido, usualmente Gaussiano, y se va pasando por una serie de capas convolutivas traspuestas; es decir, que en vez de reducir el tamaño del *input* de cada capa, se aumenta. Esto permite pasar de un vector de ruido de tamaño k, a la dimensión de la imagen que se desee generar.

Por otro lado, el discriminador es una red convolucional tradicional que se entrena para un problema de clasificación binaria: detectar si una imagen es real o es generada.

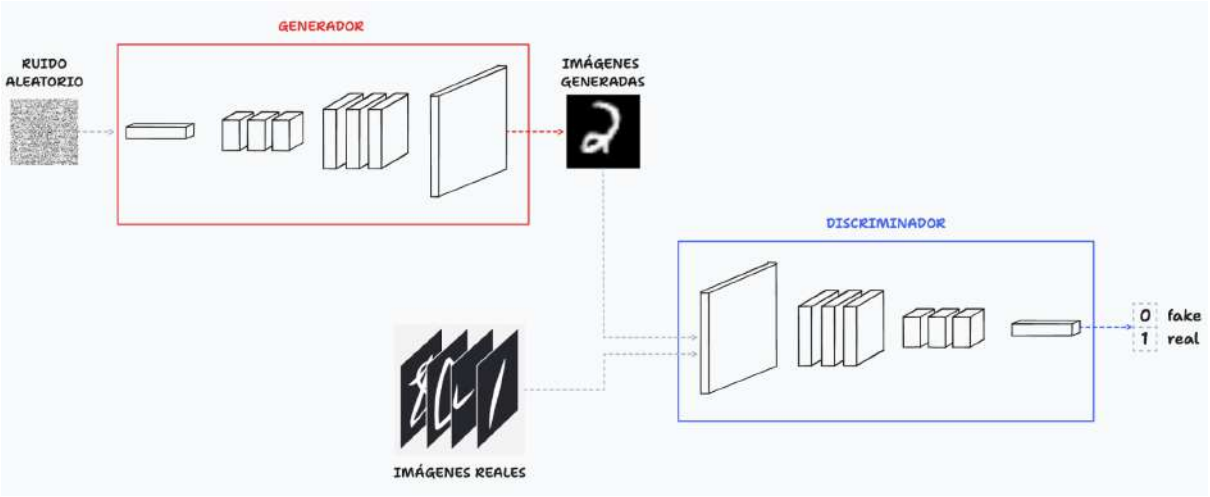


Figura 2.5 - Diagrama representativo del proceso de entrenamiento adversario del generador y discriminador para un caso de imágenes con GANs.

De esta manera es que se realiza el entrenamiento adversario, donde el generador en una etapa genera N imágenes que luego son pasadas al discriminador, junto con N imágenes reales, y esta red debe predecir cuáles de las imágenes son generadas por el generador y cuáles son las imágenes reales de referencia. El entrenamiento, por lo general, se realiza de a etapas. En una primera etapa se actualizan los parámetros del generador, manteniendo los parámetros del discriminador congelados, y luego al revés con el siguiente *batch* de imágenes, se mantienen los parámetros del generador congelados y se actualizan los del discriminador.

Una vez entrenadas las redes, solamente se precisa el generador para producir imágenes nuevas a partir de ruido, tal como se muestra en la siguiente imagen:

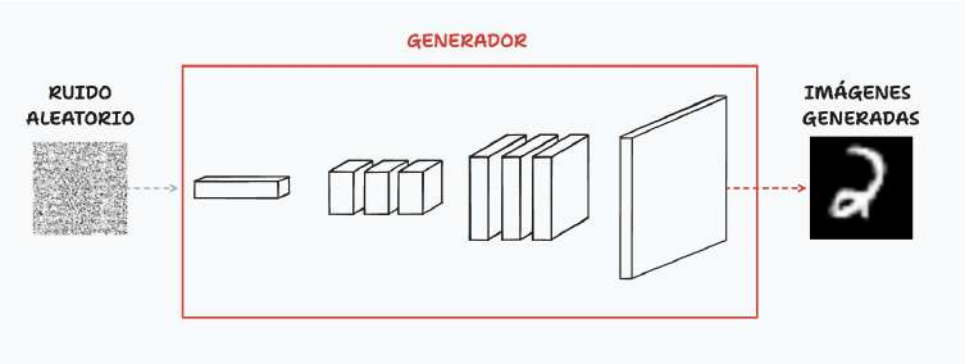


Figura 2.6 - Representación del proceso de generación de imágenes con GANs.

2.4.2 Principales aspectos

Previo a la utilización de estos tipos de modelos, es necesario conocer las ventajas y desventajas que estos presentan:

- **Alta fidelidad:** si se entrena de manera correcta, convergiendo a la mejora de performance de ambas redes de manera simultánea y estable, se llega al punto en que el discriminador no es capaz de distinguir si las imágenes generadas son reales o

no. A nivel cualitativo, se puede ver la diferencia de calidad en distintas comparativas en artículos científicos. Este es el motivo más relevante del por qué lograron ser el estado del arte por varios años, a pesar de sus dificultades de entrenamiento.

- **Baja diversidad:** la función de pérdida original presentada no tiene incentivo para que el generador cree imágenes de todas las clases posibles. Lo que tiende a pasar es que la red aprende a generar algunos casos particulares que logran engañar al discriminador y tiende a generar muchas imágenes de esas clases, fenómeno que se conoce como *mode collapse*. Si bien hay maneras de mitigar este efecto, no se logra la diversidad de resultados que se tiene con, por ejemplo, modelos de difusión.

- **Dificultad de entrenamiento:** puede ser difícil determinar cuándo la red converge durante el entrenamiento. Esto se debe a que en vez de monitorear la pérdida de una sola red, se debe monitorear las funciones de pérdida de ambas redes de manera simultánea que actúan de manera adversaria, influyendo fenómenos difíciles de identificar durante el entrenamiento como el *mode collapse*.

- **Desvanecimiento del gradiente:** durante el entrenamiento, el gradiente que fluye desde el discriminador al generador puede desvanecerse, lo que dificulta la actualización efectiva del generador.

2.4.3 Artículos más conocidos

A continuación, se resaltan algunas de las GANs más influyentes y ampliamente utilizadas en el estado del arte dentro de la IA generativa.

StyleGAN

StyleGAN, presentado en el paper "A Style-Based Generator Architecture for Generative Adversarial Networks" en 2019 por un grupo de investigadores de NVIDIA [18], introdujo avances significativos en la generación de imágenes. Se destacó por su habilidad para generar imágenes de alta resolución con un nivel de detalle que no se había alcanzado hasta el momento, permitiendo a su vez un control preciso del estilo y atributos de las imágenes. A su vez, estos estilos se agrupan en: estilos gruesos (como la pose, el pelo y la forma de la cara), estilos medios (ojos y rasgos faciales) y estilos finos (relacionados con los esquemas de color en las imágenes), pudiéndose controlar cualquiera de estos grupos utilizando otras imágenes de referencia.

Esta arquitectura habilitó las interpolaciones suaves entre diferentes estilos, conduciendo a resultados visuales fascinantes. Además de su impacto en aplicaciones realistas, StyleGAN se destacó en la creación de caras humanas realistas y demostró potencial en la exploración artística y la investigación en visión por computadora.

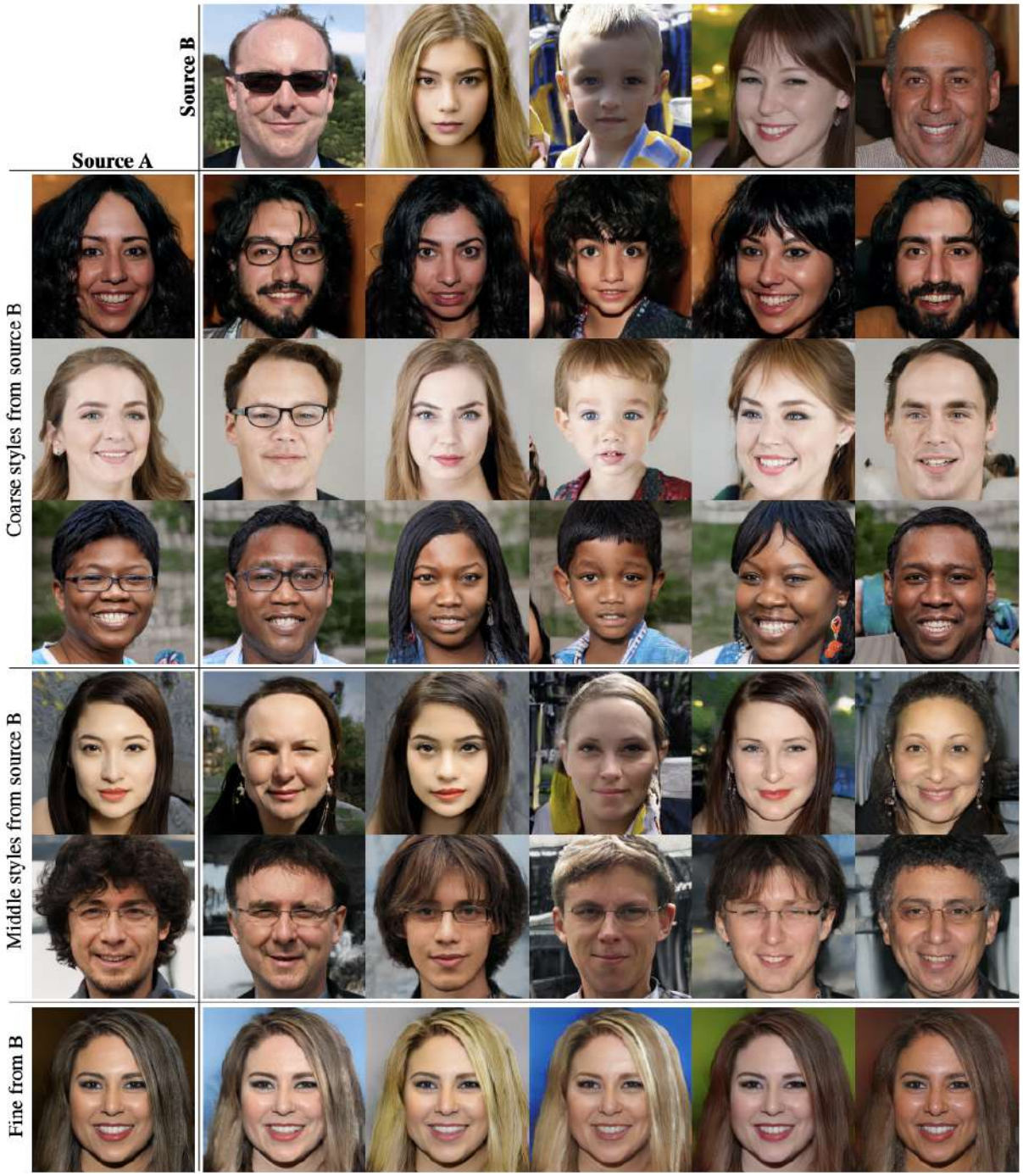


Figura 2.7 - Representación de cómo modificar el estilo de una imagen de entrada (*source A*) utilizando los estilos de otra imagen de referencia (*source B*), para estilos gruesos, medios y finos. [18]

Pix2Pix

Pix2Pix, introducida en el paper "Image-to-Image Translation with Conditional Adversarial Networks" en 2018 por un grupo de investigadores de UC Berkeley [19], presentó innovaciones significativas en la traducción de imágenes condicionales. Esta red permitió la generación de imágenes realistas y detalladas al traducir imágenes de entrada en un estilo o dominio específico. Algunas de las novedades más importantes incluyen la capacidad de realizar mapeo uno a uno entre imágenes de entrada y salida, lo que resultó en aplicaciones

como conversión de estilo artístico, transformación de mapas y segmentación de imágenes, entre otros. Pix2Pix también destacó por su capacidad para mejorar la calidad y coherencia de las traducciones mediante el uso de información condicional en el proceso de entrenamiento.

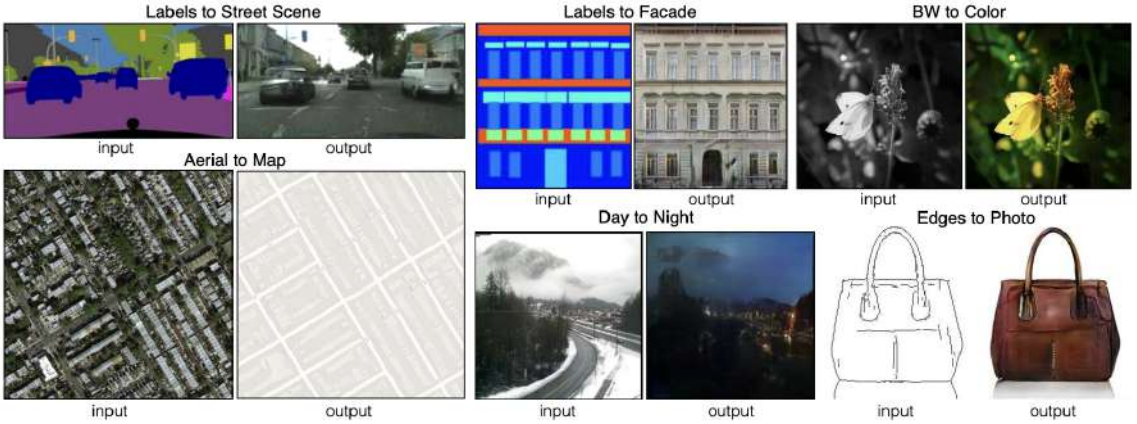


Figura 2.8 - Representación de algunas de las traducciones posibles con Pix2Pix. [19]

CycleGAN

CycleGAN, presentada en el paper "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" en 2017 por un grupo de investigadores de UC Berkeley [20], introdujo avances esenciales en la traducción de imágenes sin correspondencia directa. Una de sus novedades más importantes fue la habilidad de realizar la traducción entre dos dominios de imágenes sin requerir parejas de datos coincidentes en el conjunto de entrenamiento, lo que amplió significativamente las aplicaciones potenciales. CycleGAN destacó por su concepto de "consistencia cíclica", que garantiza que las traducciones de ida y vuelta entre los dominios sean coherentes y mantengan sus características esenciales. Esto resultó en aplicaciones como el cambio de estilo de imágenes, conversión de paisajes y transformación de objetos sin necesidad de datos correspondientes.

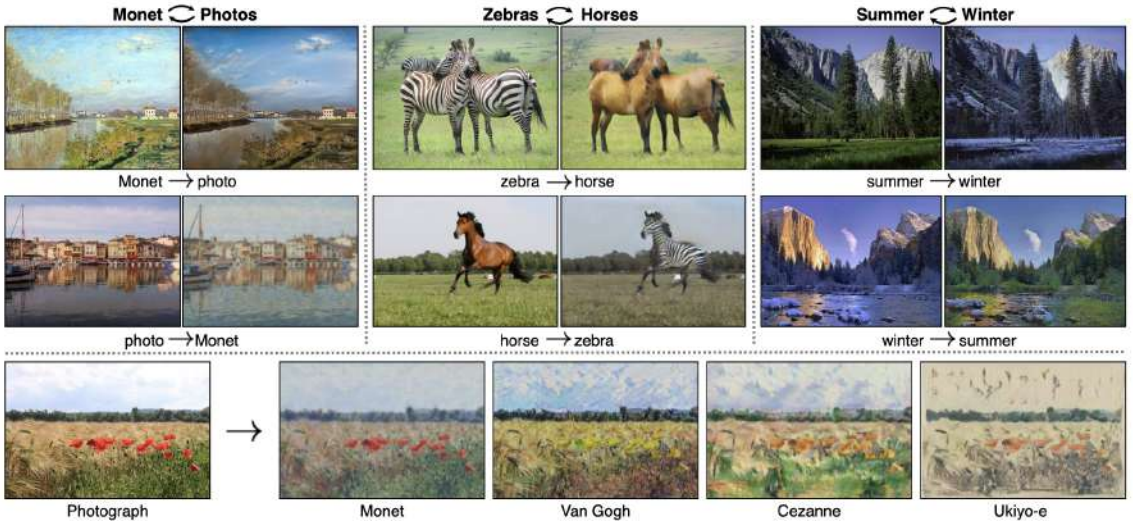


Figura 2.9 - Representación de algunas de las traducciones posibles con CycleGAN. [20]

2.5 Modelos de difusión

Los modelos de difusión son modelos generativos, lo que significa que se utilizan para generar datos similares a los datos con los que fueron entrenados. Se inspiran en la termodinámica no equilibrada y han alcanzado una calidad superando el estado del arte en la generación de diversas formas de datos, incluyendo imágenes de alta calidad e incluso audio.

En pocas palabras, los modelos de difusión funcionan alterando los datos de entrenamiento mediante la adición de ruido gaussiano (proceso denominado *forward diffusion*), y luego aprenden a recuperar la información original revirtiendo este proceso de adición de ruido paso a paso (proceso denominado *reverse diffusion*). Una vez entrenados, estos modelos pueden generar nuevos datos al muestrear ruido gaussiano aleatorio y pasarlo a través del proceso de eliminación de ruido aprendido.

El atractivo de los modelos de difusión va más allá de su capacidad para producir una calidad de imagen de vanguardia. Han ganado gran popularidad al abordar los desafíos conocidos asociados con el entrenamiento adversarial de las GANs. Los modelos de difusión ofrecen ventajas en términos de estabilidad de entrenamiento, eficiencia, escalabilidad y paralelismo.

En esta subsección se profundiza sobre las complejidades de los modelos de difusión, explorando su proceso de difusión hacia adelante, proceso de difusión inversa y el cálculo de la función de pérdida, seguido de una descripción general del proceso de entrenamiento. Al examinar estos componentes, obtendremos una comprensión integral de cómo funcionan los modelos de difusión y cómo logran sus resultados impresionantes.

Luego, en las siguientes subsecciones dentro del marco teórico del informe, se introducirá una de las arquitecturas más destacadas y utilizadas hasta el momento, *Stable Diffusion*. Seguido de una presentación de distintas técnicas para realizar un *fine-tuning* de este modelo para lograr generar imágenes para uno o varios conceptos específicos, resaltando las ventajas y desventajas de cada alternativa.

2.5.1 Comparación con otras técnicas

Más adelante en el informe se mencionarán las diferencias en términos de arquitectura, proceso de entrenamiento y otras consideraciones relevantes con respecto a los VAEs y GANs. Sin embargo, a continuación se presenta un diagrama popular en la comunidad de la IA generativa que destaca las fortalezas y limitantes propias de cada técnica.

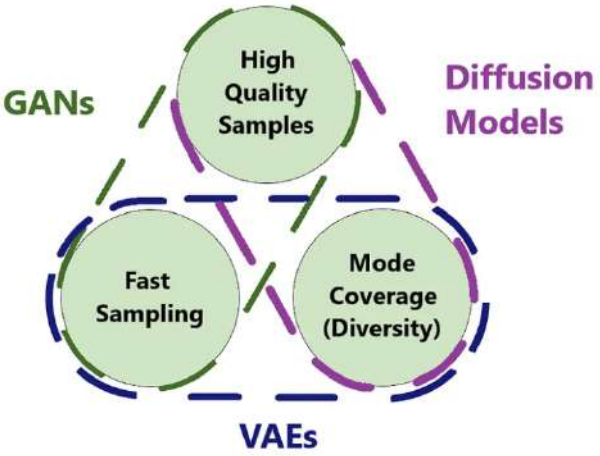


Figura 2.10 - Esquema representativo de las variables más relevantes para la comparación de las distintas técnicas generativas. [21]

Como se puede observar en el diagrama, y como fue mencionado anteriormente, los autoencoders variacionales se destacan por generar imágenes rápidamente y por ser capaces de generar una gran diversidad de imágenes, a pesar de no llegar a una calidad alta comparable con las GANs y modelos de difusión. Por otro lado, las GANs logran generar imágenes de alta calidad de manera rápida, teniendo problemas para mantener la diversidad de las VAEs o modelos de difusión, debido al mode collapse y a que, en muchos casos, generan imágenes realistas pero similares entre sí. Por último, los modelos de difusión logran superar la calidad de las imágenes en comparación con las otras dos técnicas, también logrando una gran diversidad en su generación pero son más lentas en su generación debido a que la eliminación del ruido ocurre iterativamente para mayor estabilidad.

Dependiendo del caso de uso un enfoque puede ser más útil que otro, pero en la actualidad, debido a los grandes esfuerzos de investigadores y trabajadores de la industria, se está trabajando fuertemente para realizar optimizaciones que permiten un entrenamiento e inferencia mucho más veloz sin necesidad de modificar la arquitectura de los modelos. Estas optimizaciones van desde la reducción de la precisión de los parámetros de los modelos, que usualmente no alteran de manera significativa la performance de los modelos si no se reducen drásticamente, y otras optimizaciones enfocadas en la utilización más eficiente de la GPU (*Graphical Processing Unit*) y utilización de múltiples GPUs para aprovechar el paralelismo de las operaciones.

Debido a estos avances, y la muy buena calidad y diversidad de las imágenes generadas por los modelos de difusión, es que actualmente están dominando el área de la generación de imágenes, audio, estructuras 3D, entre otros.

2.5.2 Difusión hacia adelante

El proceso de difusión hacia adelante (*forward diffusion*) es, esencialmente, un proceso en el cual el ruido gaussiano se añade gradualmente a una imagen de entrada paso a paso, a lo largo de un total de T pasos. En el paso 0 tenemos la imagen original, en el paso 1 una

imagen ligeramente modificada que se irá corrompiendo aún más paso a paso hasta que se pierde por completo toda la información de la imagen original.

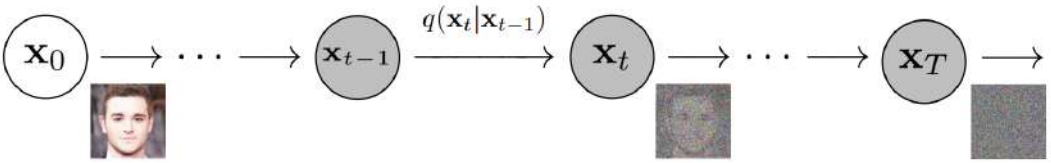


Figura 2.11 - Esquema representativo del proceso de difusión hacia adelante. [22]

Para formalizar este proceso, podemos modelarlo como una cadena de Markov fija con T pasos, donde la imagen en el tiempo t se mapea a su estado siguiente en el tiempo t+1. Como una cadena de Markov, cada paso depende solamente del anterior, lo que nos permite derivar una fórmula cerrada para obtener la imagen corrompida en cualquier tiempo deseado, evitando la necesidad de realizar cálculos iterativos.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Fórmula 2.1 - Formulación matemática del proceso de difusión hacia adelante. [22]

Esta fórmula cerrada permite un muestreo directo de x_t en cualquier tiempo deseado, acelerando significativamente el proceso de difusión hacia adelante.

Además, el ruido añadido en cada paso sigue un patrón deliberado. Se define un *Scheduler* que determina la cantidad de ruido agregado en cada paso. En el paper original de "Denoising Diffusion Probabilistic Models" (DDPM) [22], se propone añadir ruido de manera lineal, variando la cantidad de ruido agregada según un parámetro β_t desde 0.0001 en el tiempo 0 hasta 0.02 en el tiempo T. Sin embargo, métodos alternativos como la adición de ruido con una función coseno fue introducido en el paper "Improved Denoising Diffusion Probabilistic Models" [23], ganando popularidad.

La siguiente imagen muestra visualmente la diferencia entre el agregado de ruido de manera lineal, y su alternativa con la función coseno para el proceso de difusión hacia adelante. La primera fila muestra el método lineal original, mientras que la segunda fila muestra el método con coseno.

Figura 2.12 - Métodos lineal y coseno para la adición de ruido durante el proceso de difusión hacia adelante. [23]

Los autores de "Improved Denoising Diffusion Probabilistic Models" argumentan que al utilizar la función coseno se ofrece un rendimiento superior. El programa lineal puede llevar a una pérdida rápida de información en la imagen de entrada, lo que resulta en un proceso de difusión abrupto. Por otro lado, el programa con coseno ofrece una degradación más suave, permitiendo que los pasos posteriores operen en imágenes que no están completamente corrompidas por el ruido.

2.5.3 Difusión inversa

A diferencia del proceso de difusión hacia adelante, el proceso de difusión inversa plantea un desafío computacional, ya que la formulación de $q(x_{t-1}|x_t)$ que se presenta en la imagen a continuación, se vuelve muy costosa de calcular o directamente incalculable. Para abordar este problema se utilizan modelos de aprendizaje profundo a los efectos de aproximar este proceso de difusión inversa.

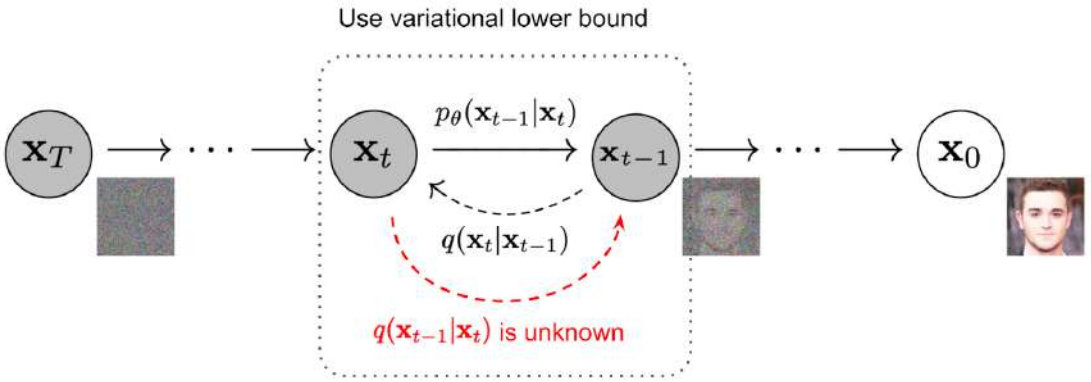


Figura 2.13 - Esquema representativo del proceso de difusión inversa. Adaptado de [22]

Para aproximar el proceso de difusión inversa, se utiliza una red neuronal. Esta red puede operar de dos formas distintas, ambas produciendo resultados válidos. La primera opción consiste en tomar una imagen en cierto tiempo t y generar la imagen en su paso anterior, t-1. Alternativamente, se puede predecir todo el ruido presente en la imagen en el tiempo t, de forma de quitar luego un poco del ruido presente según en qué tiempo se esté. Ambos enfoques contribuyen eficazmente a la reconstrucción de imágenes a partir de sus estados corrompidos, siendo el último enfoque el más habitual.

Algunos artículos científicos y blogs pueden ser confusos en el sentido que, al leerlos, puede dar la impresión de que la red predice el ruido agregado desde el tiempo t-1 hasta el tiempo t. Aunque esto puede aclararse al observar el código proporcionado por sus autores, la red predice todo el ruido presente en el tiempo t. Esto podría dar a entender que ya se tiene la información como para quitar el ruido presente en la imagen, sin la necesidad de quitar iterativamente dicho ruido; sin embargo, se demostró empíricamente que se necesitan pasos más pequeños, y de ahí surge la necesidad de eliminar solo una fracción del ruido presente en la imagen en el tiempo t para obtener la predicción de la imagen en el tiempo t-1.

En términos generales, los modelos de difusión utilizan alguna variante de U-Net para aproximar el proceso de difusión inversa. Esta elección proviene de la popularidad de U-Net

en la comunidad de Computer Visión, siendo además creada originalmente para la segmentación de imágenes médicas. Pero a su vez, el único requisito para el diseño de la arquitectura de los modelos de difusión, salvo casos particulares como difusión para aumentar la resolución de imágenes, es que la entrada y la salida del modelo deben mantener la misma dimensionalidad, comportamiento que presentan las U-Net.

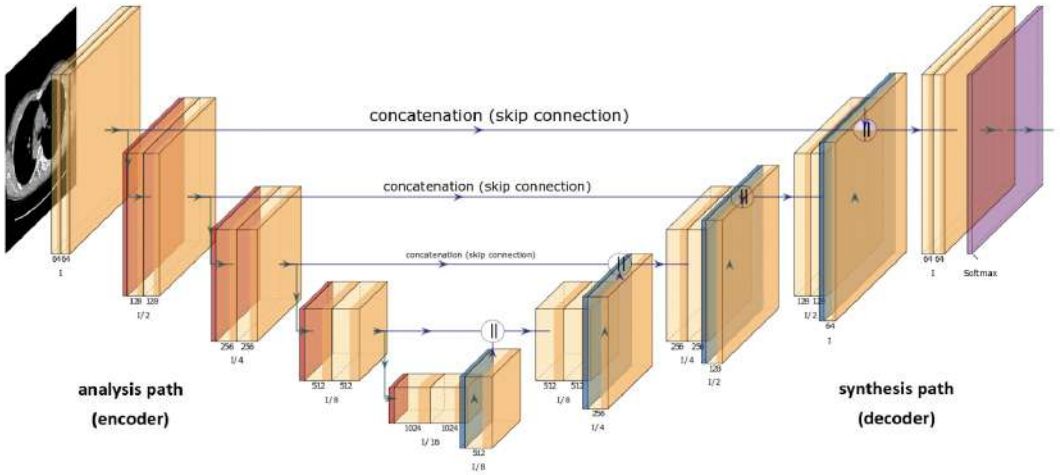


Figura 2.14 - Arquitectura de la U-Net. [24]

En lo que respecta a los detalles de implementación delineados en el paper de DDPM, las elecciones arquitectónicas involucran:

- El *encoder* y el *decoder*, similar al de las VAEs, tienen el mismo número de niveles y dimensiones, incorporando un cuello de botella entre ellos.
- Cada etapa del *encoder* consta de dos bloques residuales seguidos de un pooling, excepto en el último nivel previo al cuello de botella.
- Cada etapa del *decoder* consta de tres bloques residuales, utilizando un nearest neighbor upsampling que realiza un x2 en el tamaño, con convoluciones para restaurar la entrada del nivel anterior.
- Cada conjunto de bloques del *decoder* se encuentra conectado con los conjuntos de bloques del encoder que tienen la misma dimensión mediante skip connections. Esto permite preservar información que puede haberse perdido en el cuello de botella, utilizando *feature maps* con características menos complejas.
- Se utilizan módulos de *self-attention*.
- Adicionalmente, el tiempo se codifica en un *embedding*, similar al *Sinusoidal Positional Encoding* introducido en el paper "Attention Is All You Need" de Vaswani et al. (2017) sobre Transformers [25].

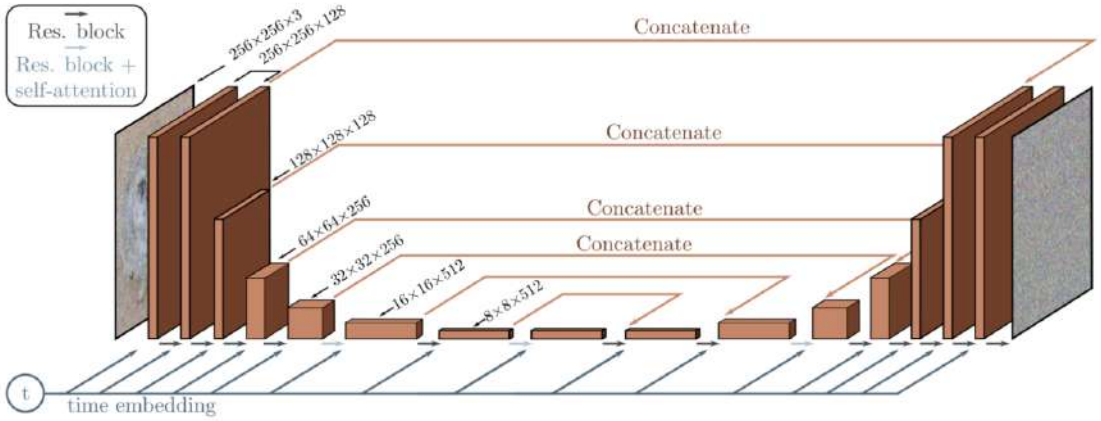


Figura 2.15 - Arquitectura del modelo de difusión basado en la U-Net. [26]

Estos *time embeddings* ayudan a la red neuronal a adquirir información específica sobre en qué estado (paso) se encuentra actualmente la imagen, para saber si se debe quitar más o menos ruido de la imagen.

2.5.4 Entrenamiento

En cada batch del proceso de entrenamiento:

1. Se muestrea un tiempo t al azar, convirtiéndolos en embeddings para cada uno de los datos de entrenamiento del *batch*, pudiendo ser distintos entre sí.
2. Se aplica ruido gaussiano a cada imagen acorde a su tiempo t, utilizando la fórmula cerrada del proceso de difusión hacia adelante.
3. Luego, se utilizan los *embeddings* de los tiempos y la imagen ruidosa para alimentar la U-Net (u otra arquitectura escogida), tratando de predecir el ruido que fue añadido.

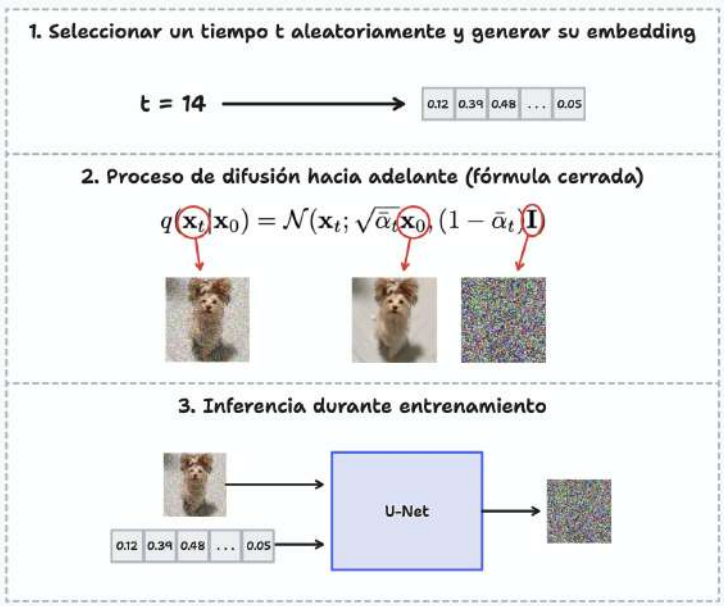


Figura 2.16 - Diagrama representativo de los principales eventos durante el entrenamiento de una única imagen.

Este proceso se repite en cada época, donde las mismas imágenes ingresan al loop de entrenamiento, normalmente variando los tiempos para cada imagen en distintas épocas. Esto permitirá que el modelo aprenda a revertir el proceso de difusión en cualquier paso temporal, tomando las mismas imágenes en cada época pero con diferentes niveles de ruido añadido.

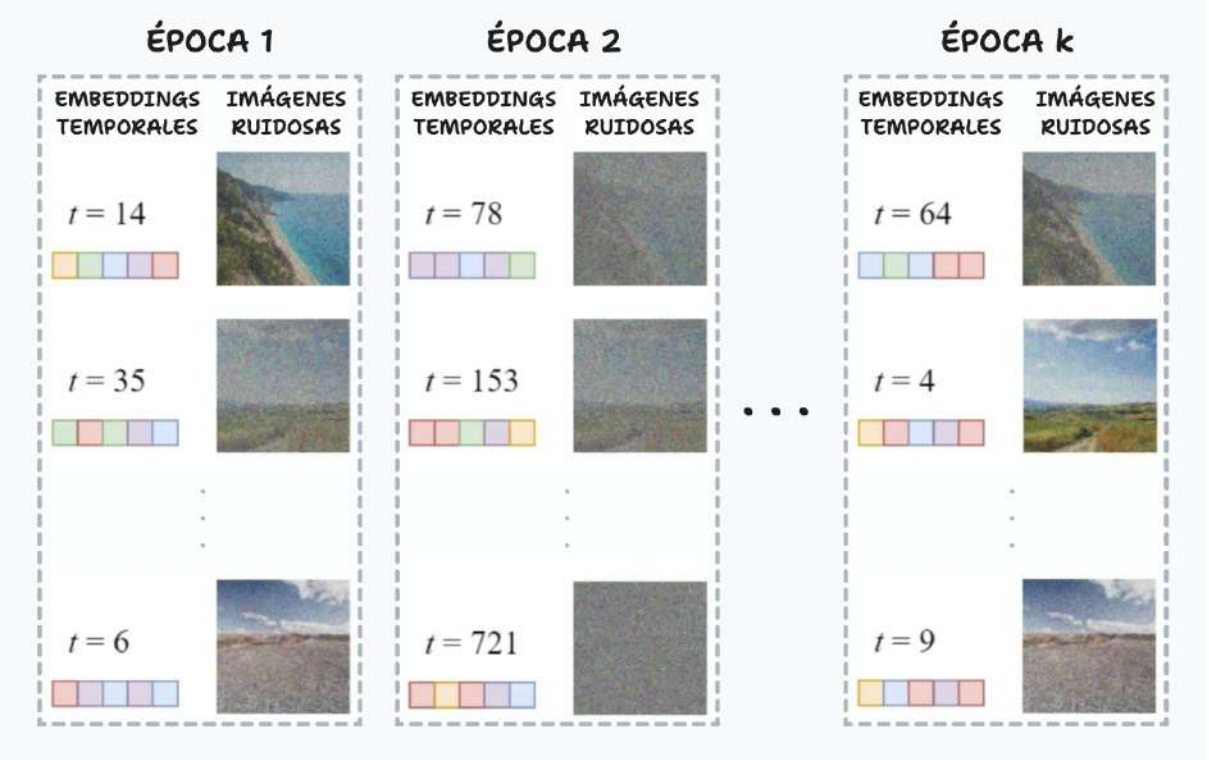


Figura 2.17 - Diagrama representativo de la corrupción de imágenes al añadir ruido gaussiano para un batch fijo de datos a lo largo de distintas épocas en el entrenamiento. Adaptado de [26]

En cuanto a la función de pérdida, se compara la predicción del ruido presente en la imagen contra el ruido real que fue agregado a través del cálculo de la divergencia Kullback-Leibler (KL). Dicha divergencia mide la asimetría entre distribuciones de probabilidad, utilizando medidas de distancia estadística y cuantificando cuánto difiere una distribución P de una distribución de referencia Q.

La derivación de está fórmula involucra un desarrollo matemático extenso, perdiendo un poco el foco de este informe; pero de todas formas se dará una noción de dónde proviene. Idealmente se busca revertir las transiciones de la cadena de Markov durante el proceso de la difusión hacia adelante, maximizando el estimador de máxima verosimilitud. Sin embargo, dicho cálculo en x_0 depende de los estados x_1, x_2, ..., x_T, volviéndose incomputable. Es por esto que se realiza una aproximación, en donde en vez de maximizar la verosimilitud -que equivale a minimizar el logaritmo del opuesto de la verosimilitud- se aproxima esta última con otra función que es siempre menor o igual a ésta, denominada *Variational Lower Bound* (L_{vlb}).

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L_{vlb}$$

Fórmula 2.2 - Desigualdad utilizada para aproximar la función de pérdida. [27]

Donde se puede obtener la expresión final desarrollando el término de L_{vlb} para cada tiempo t:

$$L_{vlb} = L_0 + L_1 + ... + L_{T-1} + L_T$$
$$L_0 = -\log p_\theta(x_0|x_1)$$
$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$
$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T))$$

Fórmula 2.3 - Desarrollo de los términos del Variational Lower Bound para cada paso de tiempo t. [27]

Siendo DKL la función que compara la distribución de las dos probabilidades p y q mencionadas anteriormente:

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

Fórmula 2.4 - Formulación matemática de la divergencia KL para distribuciones continuas. [27]

A pesar de no querer entrar en demasiado detalle matemático sobre la formulación de la función de pérdida de los modelos de difusión, se espera que este pequeño desarrollo proporcione una noción sobre cómo estos modelos calculan la pérdida, actualizando los parámetros en el modelo. En esencia, el modelo predice el ruido agregado a la imagen en un paso de tiempo t, donde se calcula la media del ruido predicho, tomando una desviación estándar fija, y se compara mediante la divergencia KL esta distribución con la distribución con la media y la desviación estándar reales del proceso de difusión hacia adelante. Este cálculo se realiza en todas las imágenes de un batch, lo que permite el posterior paso de backpropagation de la red para actualizar los pesos de la misma.

2.6 Stable Diffusion

El proceso de difusión inversa en los modelos de difusión tradicionales implica pasar iterativamente una imagen de tamaño completo a través de una U-Net hasta obtener una imagen sin ruido. Sin embargo, esta naturaleza iterativa presenta desafíos en términos de eficiencia computacional, especialmente al tratar con tamaños de imagen grandes y un alto número de pasos de difusión T. El tiempo necesario para la eliminación total de ruido de una imagen a partir de puro ruido gaussiano durante el muestreo, puede volverse excesivamente largo. Para abordar este problema, y para incluir algunas otras mejoras que se describirán más adelante, es que se desarrolló un enfoque novedoso llamado *Stable Diffusion*, originalmente conocido como Modelos de Difusión Latente (LDM, por sus siglas en inglés), propuesto en 2022 en el artículo "High-Resolution Image Synthesis with Latent Diffusion Models" [28].

En esta sección se detallarán los principales avances propuestos en este artículo que introdujo a *Stable Diffusion*: trabajar con imágenes en el espacio latente y el condicionamiento de la generación a través de un enfoque novedoso.

2.6.1 Modelos latentes de difusión

Stable Diffusion introduce una modificación clave al realizar el proceso de difusión en el espacio latente. Esto se logra utilizando un *encoder* E entrenado para convertir una imagen de tamaño completo en una representación de menor dimensión (espacio latente), luego llevando a cabo tanto el proceso de difusión hacia adelante como el proceso de difusión inversa en el espacio latente. Con la ayuda de un *decoder* D entrenado, podemos decodificar la imagen a partir de su representación latente. Normalmente, tanto el encoder y el decoder son entrenados utilizando VAEs.

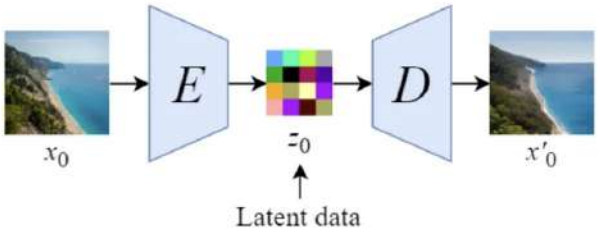


Figura 2.18 - Ilustración del autoencoder variacional propuesto en el paper de Stable Diffusion. [29]

Dentro del espacio latente mostrado en la figura anterior, es donde suceden los procesos de difusión expuestos anteriormente.

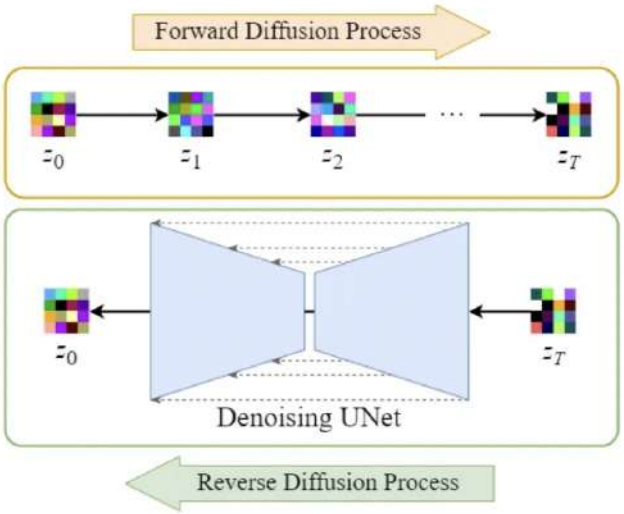


Figura 2.19 - Ilustración de los procesos de difusión hacia adelante e inverso dentro del espacio latente en los modelos latentes de difusión. [29]

Al trasladar las operaciones de difusión al espacio latente, *Stable Diffusion* mejora significativamente la velocidad computacional en comparación con los modelos de difusión tradicionales que trabajan en el espacio de imágenes. Este avance permite procesos de eliminación de ruido y muestreo más rápidos, convirtiendo a los modelos latentes de difusión

en una solución atractiva para generar imágenes de alta calidad de manera eficiente. Por otro lado, la capacidad de realizar la difusión en el espacio latente no solo mejora la velocidad, sino que también contribuye a la estabilidad y robustez general del modelo.

2.6.2 Condicionamiento

Una de las características destacadas de *Stable Diffusion* es su capacidad para generar imágenes basadas en instrucciones de texto específicas u otros datos de condicionamiento. Esto se logra mediante la introducción de mecanismos de condicionamiento en el modelo de difusión interno, también conocidos en la literatura como *Classifier-Free Guidance* (CFG) [30]. Este enfoque se propone como una alternativa al *Classifier Guidance* ya existente, con el cual podemos indicar al modelo qué clase de las vistas durante el entrenamiento queremos generar, teniendo que identificar clases y no pudiendo utilizar texto libre.

Para habilitar el condicionamiento, a la U-Net se le inyecta la información de la introducción de texto (u otro tipo de condicionamiento) con la utilización de mecanismos de atención cruzada (*cross-attention*). Esta modificación permite al modelo incorporar eficazmente información de condicionamiento durante el proceso de generación de imágenes.

Las entradas de condicionamiento pueden adoptar diversas formas según la salida deseada:

- Para las entradas de texto, se transforman primero estas instrucciones en embeddings, utilizando un modelo de lenguaje como BERT o CLIP. Luego, estos embeddings se inyectan a la U-Net mediante capas de *Multi-Head Attention*, denotadas como Q, K y V en el siguiente diagrama.
- Para otros tipos de entradas de condicionamiento, como mapas semánticos, imágenes u otras representaciones, se puede usar la concatenación para integrar la información de condicionamiento en el modelo.

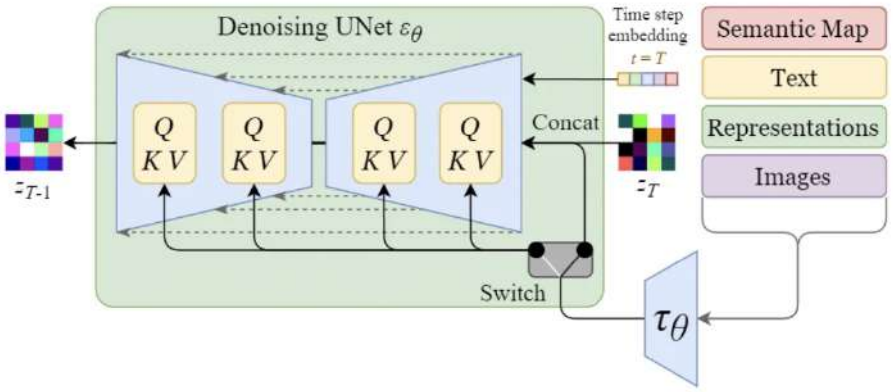


Figura 2.20 - Condicionamiento en la generación de imágenes propuesto por Stable Diffusion. [29]

Al incorporar mecanismos de condicionamiento, *Stable Diffusion* amplía sus capacidades para generar imágenes basadas en instrucciones de texto específicas o incorporar información adicional, lo que permite una síntesis de imágenes más versátil y controlada.

2.6.3 Arquitectura

A continuación se muestra cómo se vería el proceso entero propuesto por *Stable Diffusion*:

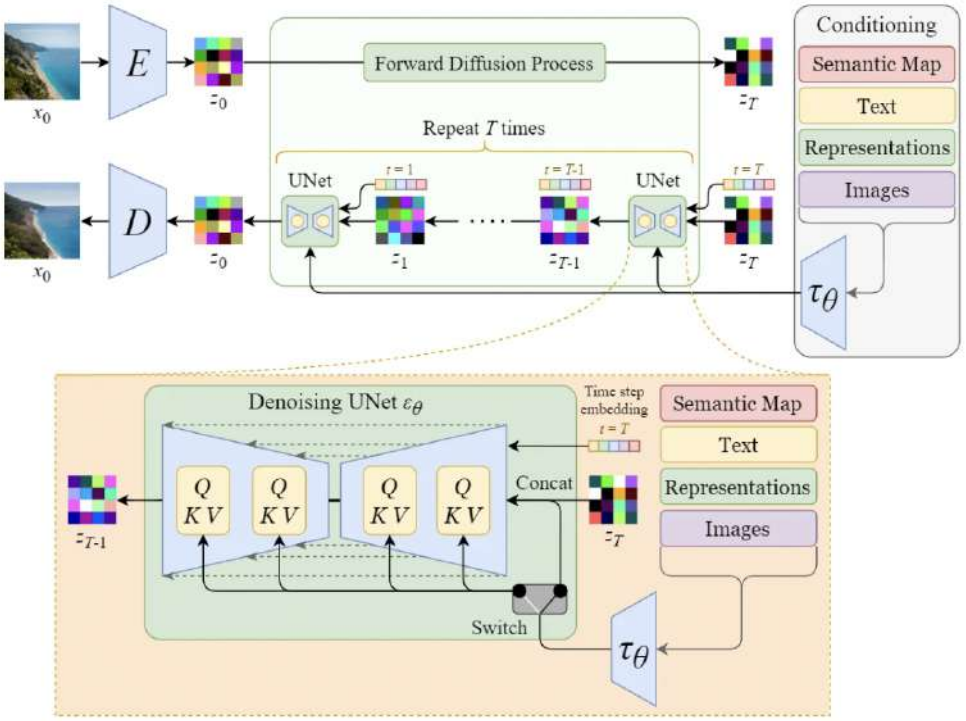


Figura 2.21 - Arquitectura de Stable Diffusion. [29]

A modo de resumen, las imágenes x_0 se codifican en el *encoder* E, que generalmente surge de ser el encoder al entrenar un Autoencoder Variacional (VAE), llegando a la representación latente z_0 de la imagen de entrada x_0. Luego, se agrega ruido gaussiano a la imagen a través del proceso de difusión hacia adelante, terminando con la representación latente de la imagen ruidosa z_T. Después, la imagen pasa iterativamente por la U-Net durante el número especificado de pasos en el proceso de difusión inversa, utilizando tanto los *embeddings* del paso temporal t como los *embeddings* provenientes de condicionamiento (instrucciones de texto, mapas de profundidad, etc.), hasta que alcanzamos la predicción de la representación del espacio latente de la imagen sin ruido $\hat{z}_0$, representada como z_0 en la imagen justo antes del *decoder*. Por último, esta predicción se pasa a través del *decoder* D del VAE para obtener la predicción de la imagen original $\hat{X}_0$, representada como x_0 a la salida del *decoder*.

2.7 Stable Diffusion XL

A fines de junio del 2023, StabilityAI lanzó *Stable Diffusion XL* (SDXL) como una mejora en cuanto a la calidad de imágenes generadas con respecto a *Stable Diffusion*. Las principales diferencias con respecto a *Stable Diffusion* son:

- La utilización de una U-Net tres veces más grande, incrementando sobre todo los parámetros que refieren a los bloques de atención. La U-Net de SDXL utiliza 2.6 mil millones de parámetros, en contraste con los 860 millones de SD 1.5 y los 865 millones de SD 2.1.

- Se tiene un mayor contexto de atención al texto de entrada debido a la utilización de un segundo encoder de texto.
- Se introduce un nuevo modelo llamado el refinador que busca, a partir de las imágenes generadas por el modelo de difusión, corregir detalles de las imágenes para aumentar la fidelidad y realismo de las mismas.

A grandes rasgos, la arquitectura de la solución completa tiene la siguiente forma:

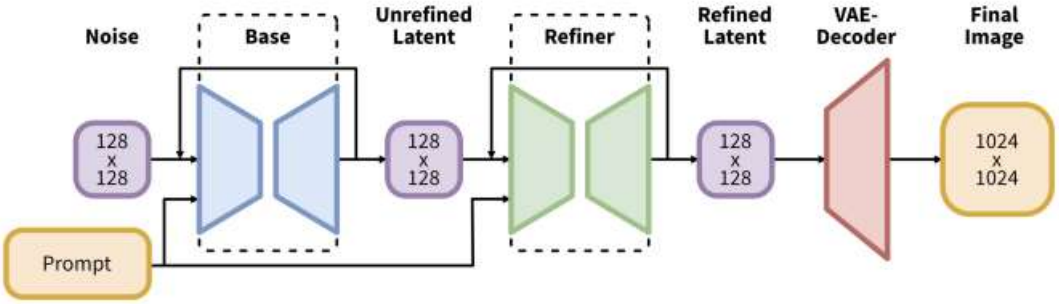


Figura 2.22 - Arquitectura de Stable Diffusion XL. [31]

En la figura anterior se puede observar que el modelo de difusión, denominado base, trabaja dentro del espacio latente de dimensión 128x128 px. Luego, la salida de este modelo que es la imagen generada en el espacio latente, se pasa por un refinador, que utiliza tanto la imagen generada como la *prompt* de texto dada para refinar detalles de la generación. Por último, esta imagen refinada dentro del espacio latente se pasa por el decoder del VAE, generando la imagen final en tamaño 1024x1024 px.

Originalmente para los encoders de texto se utilizaba CLIP ViT-L en las versiones 1.x de Stable Diffusion. Posteriormente, en las versiones 2.x, se comenzó a utilizar OpenCLIP ViT-H. En SDXL dos encoders de texto, que resultan de variantes de los ya mencionados, que son: CLIP ViT-L y OpenCLIP ViT-bigG. La cantidad total de parámetros utilizados por estos encoders es de 817 millones.

A modo de ejemplificar el impacto que tiene el modelo de refinamiento, se presenta a continuación una imagen de la generación de la cara de una persona con y sin el refinamiento.

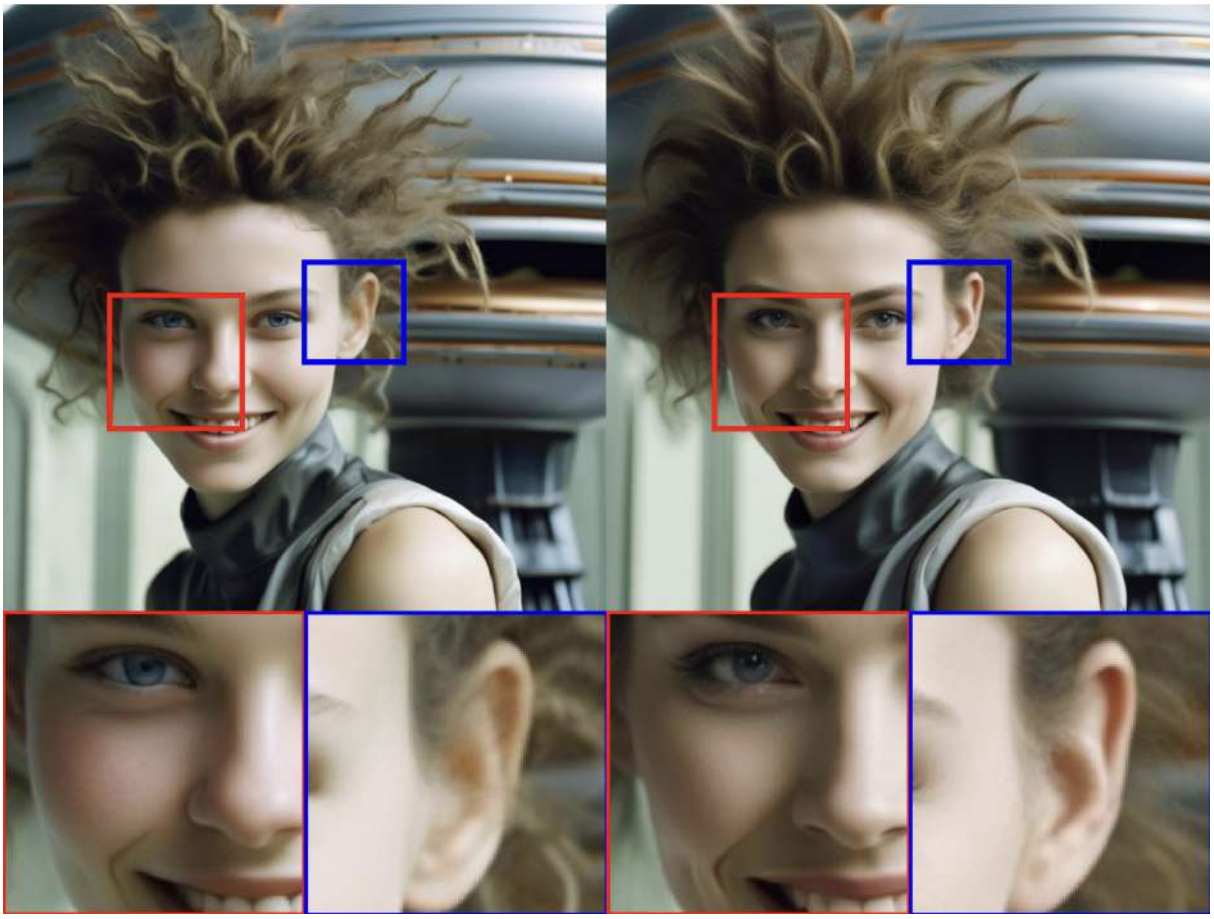


Figura 2.23 - Generación de la cara de una persona sin utilizar el modelo de refinamiento (izquierda) y utilizando el modelo de refinamiento (derecha). Abajo se pueden apreciar los detalles que se resaltan sobre las diferencias de los detalles finos de la imagen. [31]

2.8 Fine-tuning de modelos de difusión

Cuando se trata de enseñarle a un modelo de la familia de Stable Diffusion a comprender un concepto específico nuevo que no conoce tan bien en las versiones disponibles al público, como un objeto o un estilo particular, tenemos a nuestra disposición cuatro métodos principales:

- DreamBooth
- Textual Inversion
- Low-Rank Adaptation (LoRA)
- HyperNetworks

Existen algunas otras técnicas como *aesthetic embeddings* pero que hoy en día no presentan una performance comparable con las técnicas mencionadas en el punteo anterior, por lo que se dejaron fuera de la investigación.

Dentro del marco teórico se describen estas cuatro técnicas principales, de las cuales se analiza su funcionamiento interno, sus fortalezas y debilidades, para luego mencionar cuáles técnicas se deciden implementar para el caso de uso de la imagenología médica.

A modo de simplificación, consideremos un caso de uso en el que nuestro objetivo es generar nuevas imágenes de un concepto específico sólo a través de instrucciones de texto, sin depender de otras condicionantes.

2.8.1 DreamBooth

DreamBooth es un método propuesto en el artículo "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation" en 2023 por Ruiz et al. [32]. Funciona alterando los parámetros del propio modelo de *Stable Diffusion* realizando *fine-tuning* tradicional, hasta que la red termina comprendiendo el nuevo concepto.

Para enseñarle al modelo a comprender este nuevo concepto, por ejemplo, generar imágenes de un perro Corgi específico, se deben tener algunas imágenes que sirvan como referencia para mostrarle a la red este concepto y se debe proporcionar una instrucción de texto utilizando un identificador único. Este identificador es un token que no se ha visto durante el entrenamiento y que no tiene ningún significado para el modelo de *Stable Diffusion*. La idea es usar este nuevo identificador, comúnmente denominado token SKS, para que el modelo pueda comprender la asociación entre este nuevo token con el concepto visual que se ve cada vez que aparece este nuevo token en el texto.

En cuanto a la parte de la generación del *embedding* de texto para alimentar el modelo, DreamBooth no altera cómo se realiza el encoding del embedding para el nuevo token, sino que asocia el mismo al nuevo concepto en las imágenes.

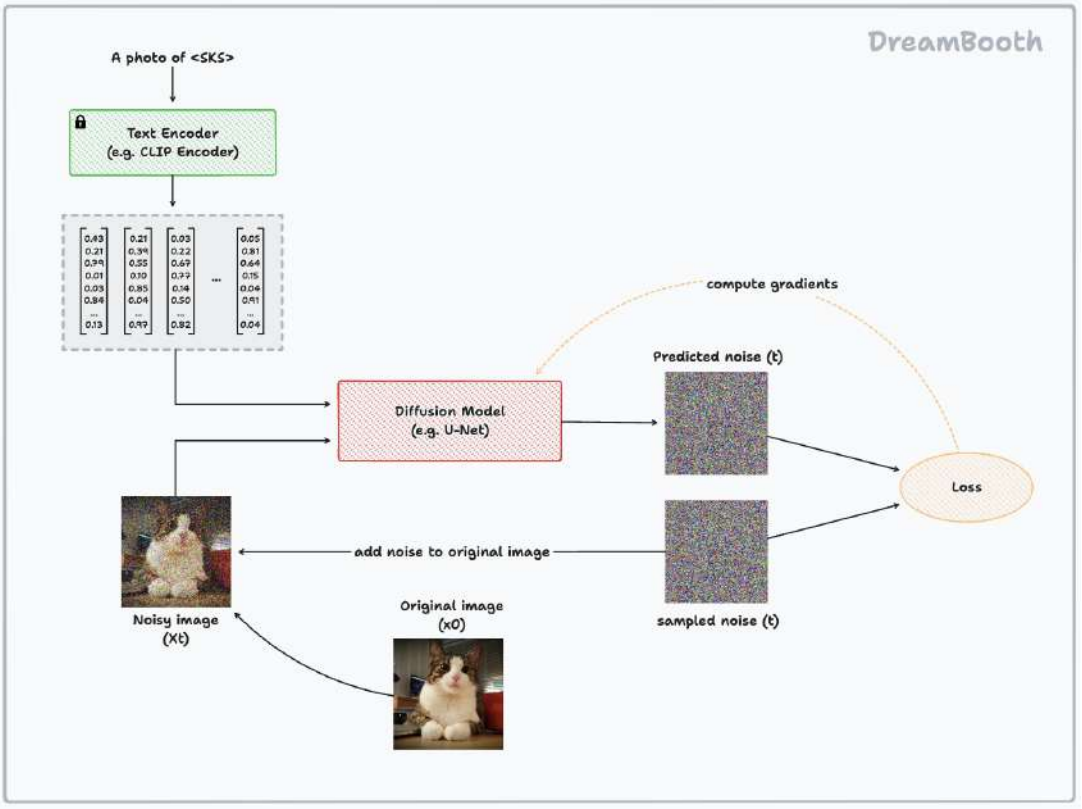


Figura 2.24 - Diagrama representativo del proceso de fine-tuning para una sola imagen utilizando DreamBooth.

A modo de simplificación, consideremos un *fine-tuning* que consiste en una sola imagen, aunque comúnmente se utilizan de entre 5 y 20 imágenes.

Se toma una imagen que contiene el nuevo concepto, luego se aplican t pasos de ruido para crear una imagen de entrada ruidosa para el modelo, y se pasa esta imagen ruidosa a través del modelo de difusión para predecir el ruido presente en la misma. El ruido predicho luego se compara con el ruido real agregado a la imagen original para computar la función de pérdida, calcular los gradientes y actualizar los parámetros del modelo de difusión.

La entrada al modelo de difusión consta de la imagen ruidosa, el *embedding* del paso temporal t tomado de manera aleatoria, y la instrucción de texto que contiene el token con el nuevo concepto; por ejemplo: "Una imagen de un gato sks".

Dado que el embedding del token SKS puede no hacer referencia a un concepto ya conocido, es probable que el modelo de difusión no haga una buena predicción las primeras veces. Sin embargo, esto eventualmente conducirá a un modelo que comprenda el concepto que estamos tratando de enseñar.

Otro aspecto interesante es que, si bien los contextos, estilos y objetos presentes en nuestras imágenes de referencia pueden no ser del todo diversos, dado el pre-entrenamiento que ya poseen los modelos abiertos al público, se pueden generar variaciones del concepto aprendido en distintos contextos, con distintos estilos y agregando artefactos a la imagen variando la instrucción de texto. A continuación se presenta un ejemplo claro de este fenómeno:

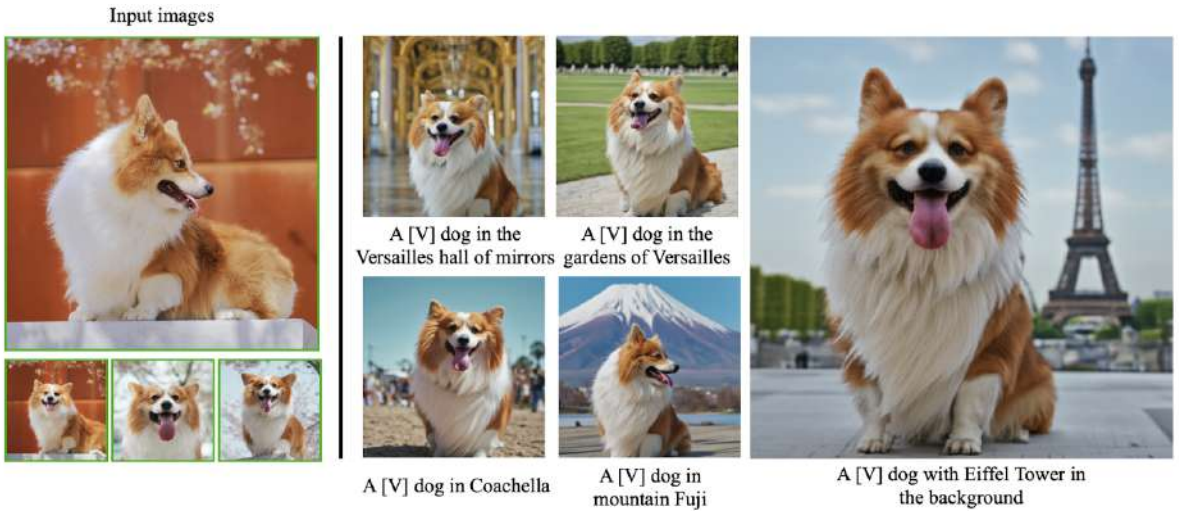


Figura 2.25 - Ejemplo de las imágenes de entrada para DreamBooth y salidas generadas a partir de distintas instrucciones de texto. [32]

Como se puede observar, se presentaron cuatro imágenes del mismo Corgi, dos con fondo naranja y dos con fondo blanco. Esto permite al modelo desambiguar qué parte de la imagen es el objeto y qué es contexto, logrando luego generar únicamente al objeto, el Corgi, en paisajes como la Torre Eiffel.

Este es probablemente el método más efectivo de enseñarle un nuevo concepto a un modelo pre-entrenado de difusión, ya que toda la red se ajusta a su caso de uso específico. Sin

embargo, esto tiene un costo, ya que la red neuronal podría volverse ligeramente (o mucho) peor en la generación de imágenes más generales.

Además, otro inconveniente es que puede que no sea una buena técnica para enseñar varios conceptos a la vez. Aunque es factible, por lo general conduce a resultados insatisfactorios. Se podría realizar *fine-tuning* de diferentes modelos de *Stable Diffusion*, uno para cada nuevo concepto, pero esto también conlleva a un mayor consumo de memoria en disco y otros recursos computacionales como la memoria de la GPU y la RAM si quisiéramos crear una solución con todos estos conceptos funcionando de manera simultánea.

2.8.2 Textual inversion

Textual Inversion es una técnica propuesta en 2022 en el artículo "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion" por Gal et al. [33].

El conjunto de componentes que interaccionan entre sí en Textual Inversion es básicamente idéntico al de DreamBooth, pero con algunas modificaciones. En lugar de realizar la actualización de gradientes sobre los parámetros de la U-Net, dicha actualización fluye hacia el embedding SKS en su lugar, dejando el modelo de difusión congelado. El propósito detrás de esto es crear el *embedding* adecuado para el token SKS, de manera de que logre identificar el fenómeno visual (concepto) que se está tratando de enseñar al modelo, en lugar de actualizar el propio modelo para generar imágenes de un cierto concepto con un *embedding* fijo de un token que no tiene ningún significado en absoluto para el encoder de texto.

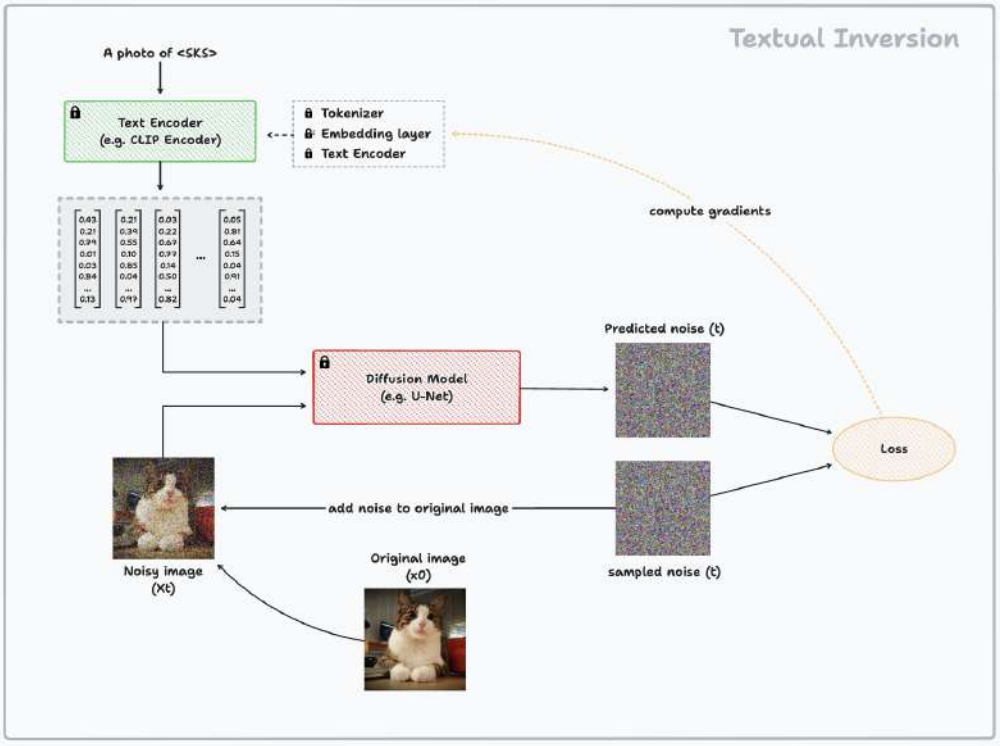


Figura 2.26 - Diagrama representativo del proceso de fine-tuning para una sola imagen utilizando Textual Inversion.

En el diagrama anterior, se puede observar, esencialmente, el mismo diagrama que con DreamBooth pero con la diferencia de que el modelo de difusión está congelado (no entrenable, representado por el icono del candado) y el cálculo de los gradientes se dirige al encoder de texto.

El encoder de texto presentado en el diagrama es en realidad una versión simplificada de lo que está sucediendo en el fondo. Este se compone por dos elementos principales: el tokenizador y el propio encoder. El tokenizador funciona básicamente como un diccionario responsable de asignar a cada token conocido un identificador numérico. Esto es necesario ya que los modelos de aprendizaje profundo sólo pueden aprender a partir de representaciones numéricas de los datos.

El encoder generalmente se compone a su vez por una capa de *embedding* y un serie de bloques que realizan el encoding, como BERT. El primero convierte cada identificador de cada palabra proporcionado por el tokenizador en un *embedding* que representa vagamente cada palabra en un cierto espacio dimensional, mientras que el segundo toma este conjunto de *embeddings* generados por la capa de *embedding* y las pasa a través de un *encoder*, obteniendo como salida una representación vectorial de toda la entrada de texto.

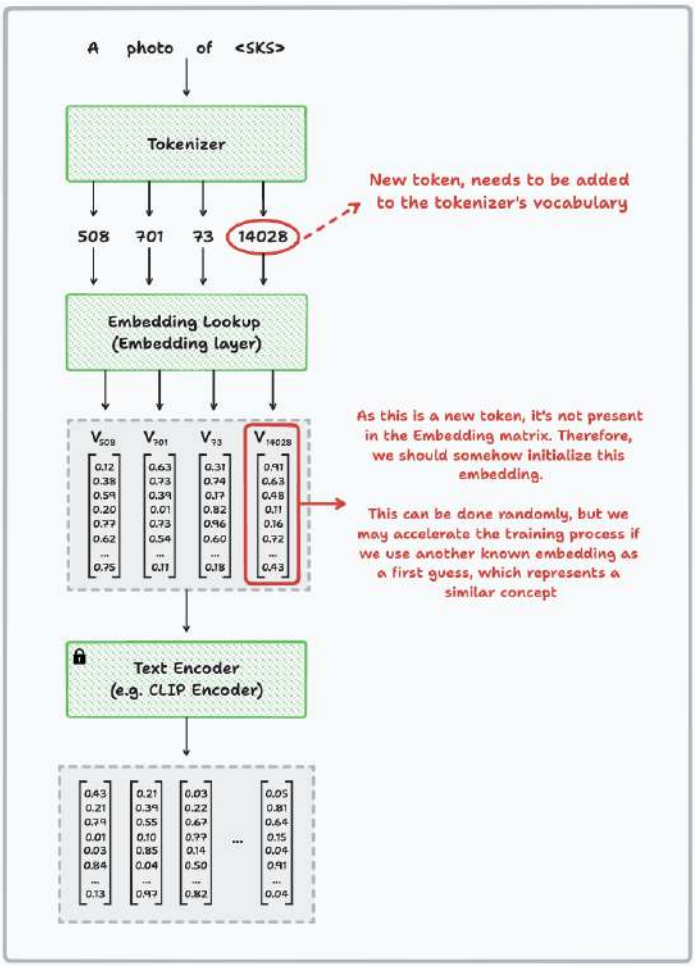


Figura 2.27 - Diagrama representativo de los componentes dentro del encoder de texto utilizado para el proceso de fine-tuning con Textual Inversion.

Para lograr realizar el *fine-tuning* de un modelo de difusión utilizando Textual Inversion, se necesita agregar el nuevo token al vocabulario del tokenizador, ya que nunca se ha visto antes, e inicializar el *embedding* para el nuevo token en la capa de embedding. Esta etapa se puede realizar utilizando una inicialización aleatoria; sin embargo, podríamos tomar un enfoque un poco más inteligente inicializando el nuevo concepto con una palabra que represente a grandes rasgos algo similar al nuevo concepto, aunque no sean exactamente lo mismo. Para entender la intuición detrás de esto, se plantea la siguiente pregunta: ¿entenderías cómo se ve un Corgi específico sin ningún conocimiento previo de cómo son los perros o incluso los animales, o sería más fácil ya sabiendo lo que es un Corgi y lo que es un perro en general?

Puede parecer un poco contra intuitivo pensar que podemos enseñarle al modelo a comprender un cierto concepto simplemente encontrando el *embedding* de texto adecuado para el SKS en lugar de hacer fine-tuning del propio modelo de difusión, pero funciona muy bien en la práctica. Esto significa que los modelos de la familia de *Stable Diffusion* tienen un entendimiento tan grande de los fenómenos visuales que, simplemente al usar el vector perfecto, podemos crear fenómenos visuales arbitrarios que tienen sentido para los humanos. Los autores afirman que "el espacio de embeddings es lo suficientemente expresivo como para capturar la semántica básica de las imágenes". Otra forma de interpretar esto, es que podríamos representar cualquier concepto simplemente escribiendo la prompt de texto correcta. Dado que existen varios conceptos que son extremadamente difíciles de expresar con palabras, de ahí surge la necesidad de una manera automatizada de aprender esta representación del embedding de un concepto determinado.

Uno de los mayores beneficios de utilizar Textual Inversion es que el artefacto que se guarda es más pequeño que con DreamBooth. Mientras que en DreamBooth necesitamos guardar el modelo completo que al que se le hizo *fine-tuning*, con Textual Inversion podemos simplemente guardar el *embedding* del token SKS, y esto a su vez puede escalar a varios nuevos conceptos, ya que consisten en embeddings de texto distintos que son utilizados por el mismo modelo.

2.8.3 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) es una técnica propuesta en 2021 originalmente para modelos grandes de lenguaje, en el artículo "LoRA: Low-Rank Adaptation of Large Language Models" por Hu et al. [34].

Se propuso originalmente para resolver problemas vinculados al de DreamBooth en el que no podemos enseñar varios conceptos específicos al modelo sin tener varias réplicas del modelo, lo que sería ineficiente e inviable. Esto ocurre en un contexto en donde los modelos grandes de lenguaje (LLM, por sus siglas en inglés) venían creciendo muy rápidamente en cuanto a su tamaño, teniendo por ejemplo 175 mil millones de parámetros GPT-3 lanzado en el año 2020.

Para los modelos de difusión, comúnmente usamos un modelo cuya dimensión de entrada y de salida es la misma, como un U-Net tradicional. La idea detrás de LoRA es insertar nuevas

capas de entrenamiento pequeñas y ajustables en el modelo, llamadas capas de LoRA, de tal manera que estas no afecten los parámetros del modelo en absoluto, manteniendo el modelo original congelado.

Durante los primeros pasos del *fine-tuning*, estas capas de LoRA generarán una salida idéntica a su entrada, replicando el comportamiento del modelo de difusión pre-entrenado que está congelado. Sin embargo, eventualmente los parámetros de dichas capas se van modificando por el cálculo de gradientes, ajustando sus pesos de tal manera que estas ligeras modificaciones alteren la entrada a las capas intermedias dentro del modelo de difusión congelado, guiando la generación hacia el nuevo concepto deseado.

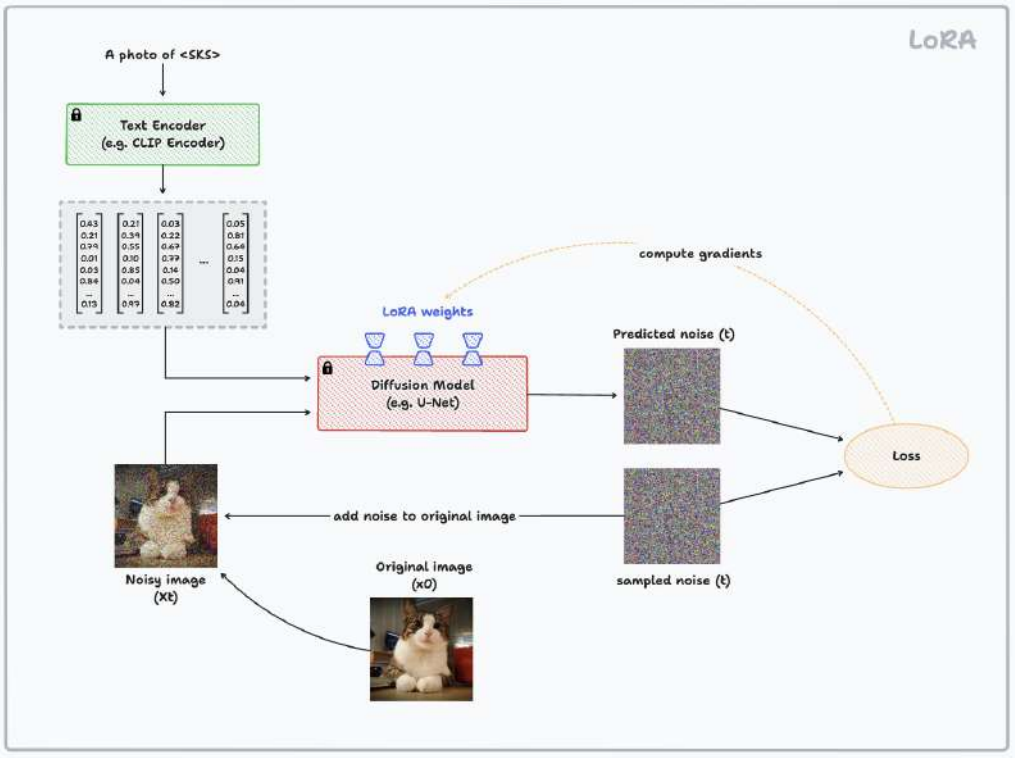


Figura 2.28 - Diagrama representativo del proceso de fine-tuning para una sola imagen utilizando LoRA.

Como se puede observar en el diagrama anterior, el modelo de difusión está congelado, permaneciendo no entrenable durante todo el proceso. Los pesos de LoRA son los que se actualizan mediante el cálculo de gradientes.

Pero, ¿cómo funcionan las capas de LoRA? Estas capas se suelen agregar en los bloques de atención que se encuentran en la U-Net, tomando la misma entrada, realizando una multiplicación de matrices adicional y sumando el resultado a la salida de la capa de atención del modelo de difusión.

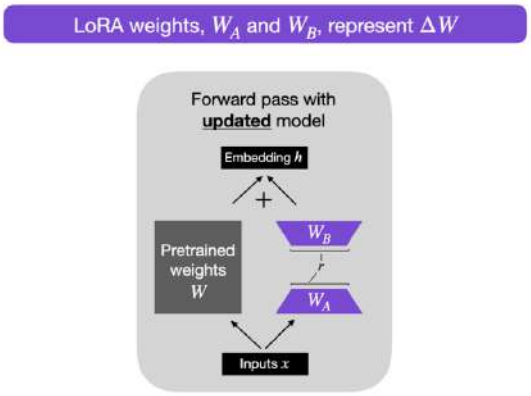


Figura 2.29 - Diagrama representativo de cómo interactúan los pesos de LoRA con respecto a los pesos del modelo congelado de difusión. [35]

Otra pregunta relevante es, ¿por qué necesitamos estas nuevas capas en lugar de simplemente actualizar los pesos de atención congelados? Uno de los principales beneficios de las capas LoRA es la reducción de la cantidad de parámetros que estos utilizan. El término "Low-Rank" (rango bajo) se refiere a esto. En lugar de tener una matriz de dimensiones m por n, configuramos una multiplicación de dos matrices W_A y W_B de dimensiones m por k y k por n, respectivamente. Si contamos la cantidad de parámetros necesarios con cada enfoque, con el enfoque anterior almacenamos 2 millones de valores, mientras que con el nuevo enfoque (LoRA) almacenamos 6000 valores (W_A almacena 2000 valores y W_B almacena 4000 valores).

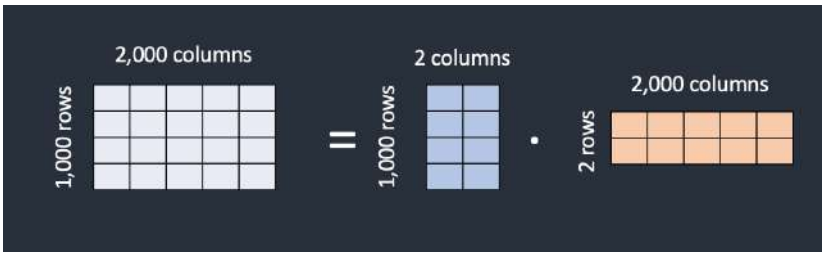


Figura 2.30 - Representación matricial de la descomposición de matrices al utilizar LoRA. [36]

El entrenamiento con LoRA es mucho más rápido que con DreamBooth y requiere menos memoria. Además, las capas de LoRA son considerablemente más pequeñas que el propio modelo de *Stable Diffusion*, por lo que son mucho más fáciles de almacenar y compartir que el modelo completo, aunque son más pesadas y con más parámetros que los embeddings de Textual Inversion.

2.8.4 HyperNetworks

Las HyperNetworks funcionan de manera similar a LoRA, pero en lugar de agregar estas capas auxiliares que modifican levemente los valores en las capas del modelo de Stable Diffusion, se tiene una red neuronal complementaria llamada HyperNetwork. Esta red se entrena de forma de ajustar ligeramente los valores de los mapas de características dentro del modelo de difusión, al igual que las capas de LoRA, de modo de que el modelo de difusión aprenda a generar el concepto deseado.

No hay un artículo oficial que se considere la base para este tipo de redes. Sin embargo, hay varios artículos que introducen esta forma de realizar fine-tuning de los modelos de difusión, donde probablemente uno de los más populares es ControlNet, propuesto en 2023 en el artículo "Adding Conditional Control to Text-to-Image Diffusion Models", de Zhang y Agrawala [37]. Si bien ControlNet introduce también otras ideas innovadoras, que serán presentadas más adelante, se considera dentro de la categoría de HyperNetworks.

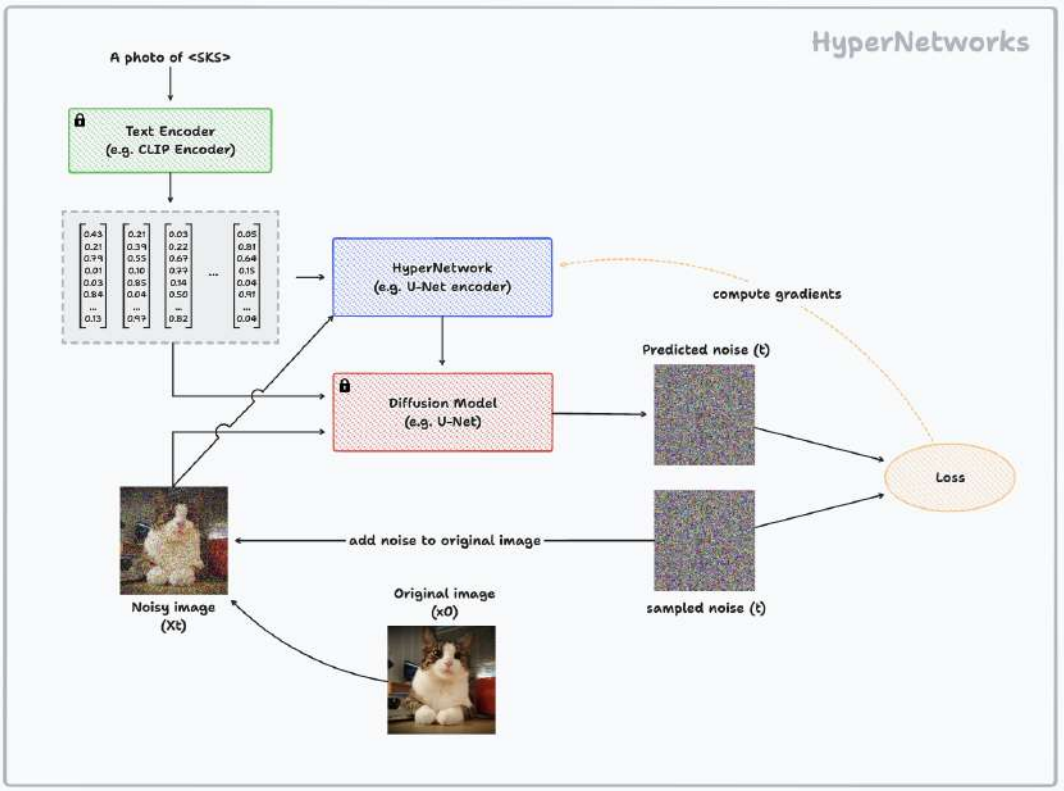


Figura 2.31 - Diagrama representativo del proceso de fine-tuning para una sola imagen con HyperNetworks.

Una forma sencilla e intuitiva de diseñar la arquitectura de la HyperNetwork es crear una copia de los bloques de *encoder* del U-Net, incluido su *bottleneck*, y luego conectar estos bloques a los bloques de *decoder* equivalentes del modelo de difusión. En términos generales, actúa como si tuviéramos dos encoders distintos de U-Net y sólo un decoder, donde uno de los *encoders* y el *decoder* tienen parámetros congelados (el modelo de difusión ya entrenado), y el nuevo encoder actúa al alterar ligeramente la entrada y agregarla a las capas del *decoder* para guiar la generación hacia el nuevo concepto.

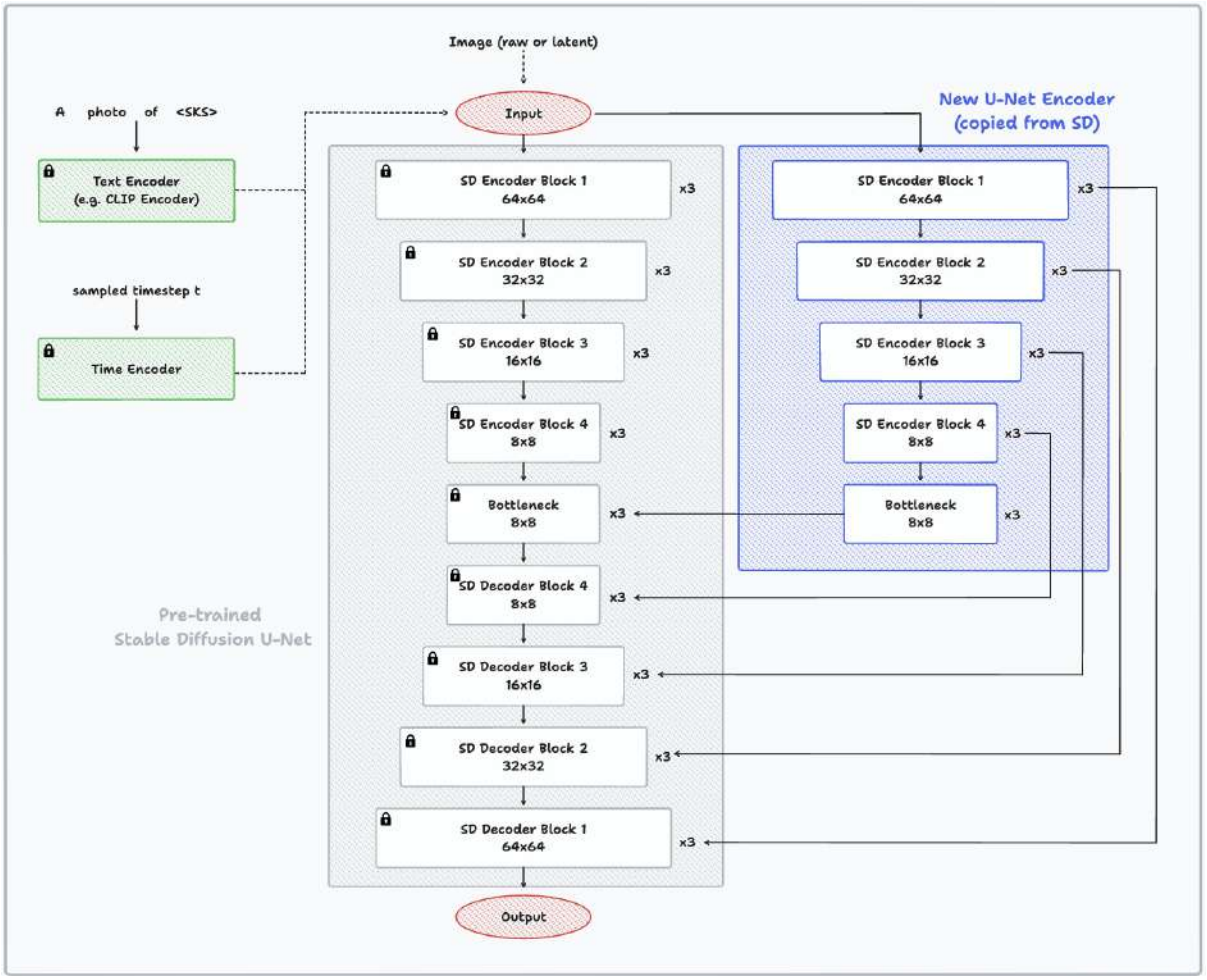


Figura 2.32 - Arquitectura de ejemplo de una HyperNetwork y su conexión con el modelo de difusión. En el centro, en gris, se puede observar la U-Net pre-entrenada de un modelo de Stable Diffusion con parámetros congelados. Del lado derecho, en azul, se puede observar el nuevo encoder (HyperNetwork) con las mismas capas copiadas de la U-Net, que se conectan con las capas del decoder del modelo de difusión pre-entrenado.

Aunque este enfoque aún no ha producido resultados empíricos comparables a las técnicas mencionadas anteriormente, las HyperNetworks han ganado mucha atención con el lanzamiento de ControlNet, arquitectura de la que hablaremos a continuación. Estas redes tienen más parámetros entrenables en comparación con la Textual Inversion y LoRA, lo que teóricamente podría significar que podrían aprender patrones más complejos, pero menos parámetros que realizar *fine-tuning* sobre el modelo de Stable Diffusion como ocurre con DreamBooth.

2.8.5 ControlNet

ControlNet se propuso como una arquitectura de red neuronal capaz de controlar grandes modelos de difusión de imágenes (como Stable Diffusion) para aprender condiciones de entrada específicas para tareas de forma de guiar la generación del modelo con éstas.

ControlNet clona los pesos de un gran modelo de difusión en una copia entrenable y una copia bloqueada. La copia bloqueada conserva la capacidad de la red aprendida a partir de

miles de millones de imágenes proporcionadas en el pre-entrenamiento, mientras que la copia entrenable se entrena en conjuntos de datos específicos de tareas para aprender el control condicional, directamente vinculado a la arquitectura presentada en las HyperNetworks, pero con algunas modificaciones.

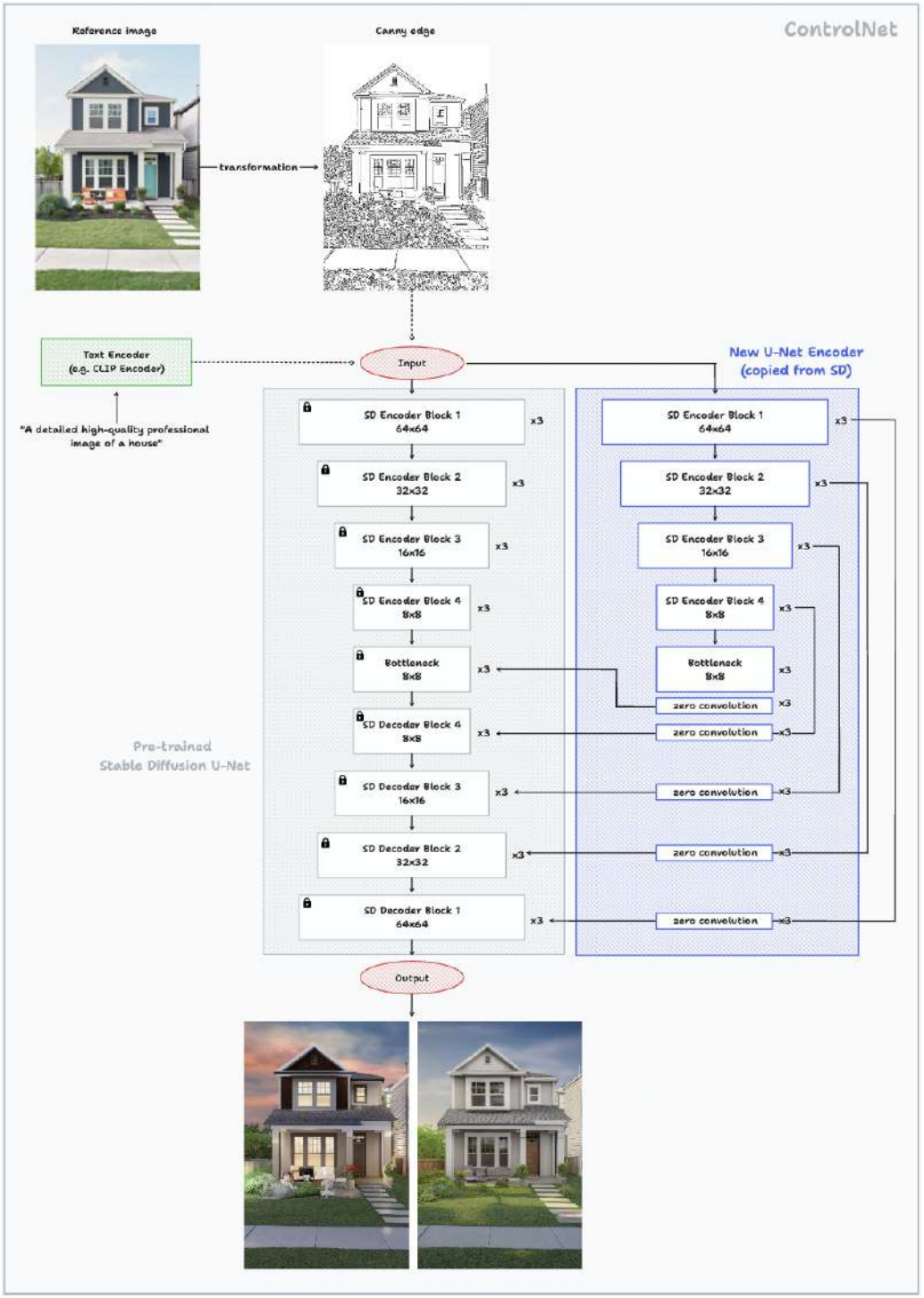


Figura 2.33 - Arquitectura de ControlNet al realizar una inferencia sobre una única imagen con Canny Edge.

Los cambios más notables en la arquitectura en comparación con la mostrada en el diagrama de HyperNetworks son: la adición de capas de convolución cero después de los bloques del nuevo encoder y antes de entrar al decoder de Stable Diffusion, y la capacidad de guiar la generación de imágenes con una nueva entrada como la que se muestra en el ejemplo de Canny Edge.

Los bloques de la red entrenables y la congelada se encuentran conectados mediante una capa denominada "convolución cero", donde los pesos se inicializan como cero de forma de no afectar inicialmente a la salida de *Stable Diffusion*. Estos parámetros van siendo optimizados durante la etapa de *fine-tuning* para generar la salida esperada. Dado que los pesos listos para producción se conservan, el entrenamiento es robusto en conjuntos de datos de diferentes escalas. Como la convolución cero no agrega nuevo ruido a las características profundas, el entrenamiento es tan rápido como ajustar finamente un modelo de difusión, en comparación con entrenar nuevas capas desde cero.

Por otro lado, los autores del artículo resaltan los bajos requerimientos de cómputo en comparación con otras alternativas que realizan generación de imagen a imagen. Dentro del artículo se presentan ejemplos de algunas imágenes a las cuales se les realizó su mapa de profundidad y se compara la calidad en la generación de ControlNet con un *fine-tuning* de Stable Diffusion 2. ControlNet, siendo entrenado en una sola computadora con una GPU Nvidia RTX 3090TI con 200.000 muestras, requirió menos de una semana de entrenamiento, mientras que el *fine-tuning* requirió 12 millones de muestras con grandes *clusters* de Nvidia A100, por más de 2000 horas de GPU. A continuación se muestran las imágenes de entrada y los ejemplos de salidas con ambas alternativas.

Figura 2.34 - Ejemplos de imágenes de entrada utilizadas para la comparación de ControlNet vs fine-tuning de Stable Diffusion 2 para depth-to-image. [37]

Figura 2.35 - Ejemplos de generación con ControlNet. [37]

Figura 2.36 - Ejemplos de generación con fine tuning de Stable Diffusion 2 para depth-to-image. [37]

Algunas de las condiciones más comunes utilizadas en ControlNet son, pero no se limitan a:

- **Canny Edge:** detección de bordes al suavizar la imagen con filtros gaussianos, luego eliminar el ruido usando un kernel gaussiano discreto y, a continuación, identificar las áreas de la imagen con los gradientes de intensidad más fuertes.
- **Depth map:** se refiere a una imagen o canal que contiene información sobre la distancia de las superficies de los objetos presentes en una escena desde un cierto punto de vista.
- **HED:** intenta abordar las limitaciones del detector de bordes Canny a través de una red neural profunda de extremo a extremo.
- **Normal mapping:** es una técnica de mapeo de texturas utilizada para simular la iluminación de protuberancias y abolladuras, mejorando la apariencia y los detalles de un modelo de baja poligonización mediante la generación de un mapa de normales a partir de un modelo de alta poligonización o un mapa de alturas.
- **Line segment detection:** similar a Canny Edge pero para detectar líneas en lugar de segmentos.
- **Scribbles:** se utilizan para controlar la generación con garabatos o bocetos.

3. METODOLOGÍA

A continuación se presentan los puntos más relevantes en cuanto a la metodología utilizada para el desarrollo del proyecto, presentando de manera temporal las etapas que enmarcan el trabajo del equipo.

3.1 Investigación del estado del arte

Durante la etapa de investigación del estado del arte se abordaron dos elementos esenciales para el desarrollo de esta tesis. En primer lugar, se exploraron diversas técnicas aplicadas a modelos de difusión y estrategias para el *fine-tuning* de dichos modelos, tal como se detalló en la sección del marco teórico. Esta revisión proporcionó una base sólida para comprender las metodologías y técnicas para su posterior implementación, así como para comprender de mejor manera los artículos académicos que utilizan distintas variantes de estas técnicas.

Por otro lado, se llevó a cabo una investigación exhaustiva sobre las aplicaciones existentes de los modelos de difusión en el contexto de la imagenología médica, identificando casos de uso posibles y las técnicas utilizadas en tales aplicaciones, tal como se detalló en la sección de la situación actual. Estas aplicaciones incluyen la reconstrucción tridimensional a partir de datos bidimensionales limitados, la mejora de la resolución de imágenes médicas, la segmentación de imágenes médicas y la generación de imágenes médicas como herramienta de aumento de datos para clasificadores, entre otras.

Este proceso de investigación proporcionó una comprensión sobre las posibilidades y desafíos que implica la implementación de modelos de difusión en la imagenología médica. A partir de esta base de conocimiento, se procedió a la etapa de selección de datos, de manera de poder realizar una aplicación práctica en algún sector dentro de la imagenología médica.

3.2 Selección de datos

Para abordar esta investigación, se llevó a cabo una búsqueda exhaustiva de conjuntos de datos públicos de diversas fuentes. Se exploraron entidades reconocidas y páginas web especializadas en la recopilación y distribución de datasets. En particular, se investigaron plataformas como Kaggle, que ofrece una amplia variedad de conjuntos de datos en diferentes dominios, y el UCI Machine Learning Repository [38], que proporciona recursos para la comunidad de machine learning. Además, se examinaron portales de datos abiertos gubernamentales que brindan acceso a conjuntos de datos en áreas de salud y medicina.

Al enfocarnos en imágenes médicas, se identificaron otras entidades clave que ofrecen datasets específicos para este dominio. *The Cancer Imaging Archive* (TCIA) [39] emergió como una fuente valiosa de conjuntos de datos de imágenes médicas en el ámbito oncológico. También se consideró el *Medical Segmentation Decathlon* (MSD) [40], que presenta desafíos de segmentación para imágenes médicas en diversas áreas, como cerebro, pulmón y corazón. Asimismo, se exploraron recursos como Radiopaedia, que proporciona imágenes y casos clínicos en el campo de la radiología, y la iniciativa ImageCLEF [41], que organiza desafíos de recuperación de información en imágenes médicas.

Es importante señalar que las imágenes médicas varían en formatos y calidad, ya que incluyen tomografías, radiografías, rayos X y otros tipos de modalidades. La disponibilidad de datos etiquetados por expertos en el dominio médico es limitada, lo que a menudo requiere el uso de enfoques basados en reglas y procesamiento de lenguaje natural para etiquetar las imágenes. Además, la cantidad de datos en los datasets puede ser escasa, lo que presenta desafíos al entrenar modelos de aprendizaje profundo.

En este contexto, se seleccionó para esta investigación el Desafío ISIC 2018 (organizado por la *International Skin Imaging Collaboration*, ISIC) sobre lesiones de piel [42]. En particular, se hace foco sobre el dataset de lesiones de piel, basado en su relevancia para los objetivos de la investigación y su disponibilidad para abordar los problemas de generación y clasificación de imágenes médicas.

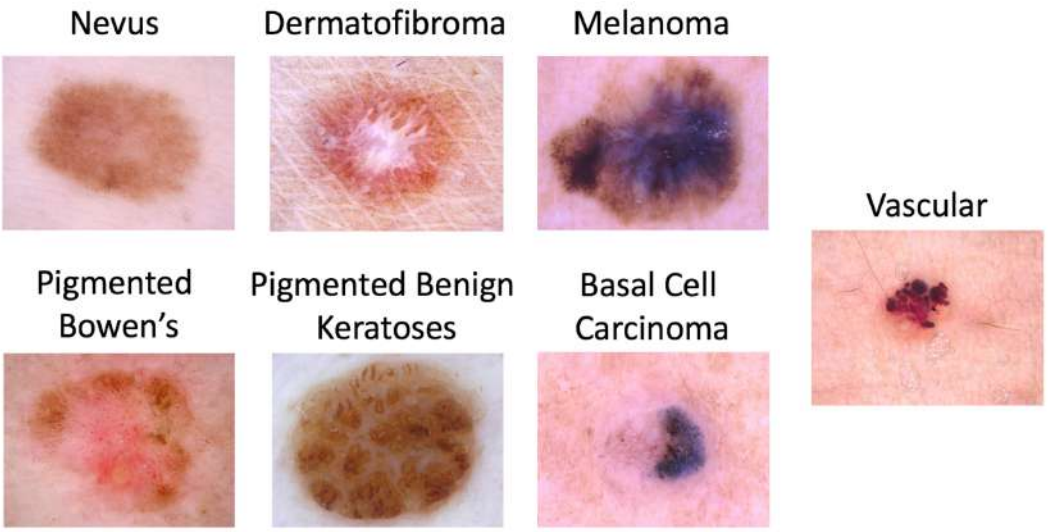


Figura 3.1 - Imágenes de referencia de las siete lesiones de piel disponibles en el dataset seleccionado. [42]

Este proceso de selección permitió obtener una visión general de la diversidad, cantidad y calidad de los datos disponibles en el ámbito de las imágenes médicas, así como de las dificultades asociadas con la falta de datos etiquetados y la variedad de formatos presentes.

3.3 Baseline

Con el objetivo de establecer una base de referencia para evaluar el desempeño de los modelos de difusión, se llevan a cabo pruebas preliminares utilizando distintos modelos de difusión pre-entrenados. Estos modelos se utilizan sin ningún tipo de *fine-tuning*, generando imágenes para evaluar su calidad de manera cualitativa. Para optimizar estos resultados iniciales, se aplica *prompt engineering* para ajustar tanto lo que se busca generar como aquellos artefactos o estilos que se quieren evitar, utilizando *negative prompts*. Se modifican otros parámetros como la cantidad de pasos de inferencia para encontrar valores que generen imágenes de mayor calidad.

Este baseline se utiliza como punto de comparación para evaluar las mejoras resultantes de los métodos de *fine-tuning*, considerando aspectos como la calidad de imagen, la fidelidad a

los patrones esperados y la presencia de artefactos visuales. Estos procedimientos proporcionan un marco sólido para las comparaciones posteriores de este proyecto.

3.4 Selección e implementación de técnicas

Para la selección de las técnicas a implementar, se optó por centrarse en un enfoque de *text-to-image* para mantener la simplicidad, descartando la opción de ControlNet, que involucra entradas adicionales. Por otro lado, se concluyó que los resultados de las HyperNetworks no están a la par de otras técnicas como Textual Inversion y LoRA para la generación de imágenes a partir de texto en la actualidad. Por último, dado que se anticipa la necesidad de varios *fine-tunings* para enseñar diferentes conceptos a la red, como distintos tipos de lesión de piel por separado, se descartó DreamBooth en favor de las técnicas restantes: Textual Inversion y LoRA.

En términos de implementación técnica, se consideraron varias opciones de librerías o *frameworks* para realizar la definición de componentes y realizar el fine-tuning. Se busca un equilibrio entre la facilidad de implementación y el poder visualizar cómo interactúan los distintos componentes en soluciones de difusión. Se exploraron alternativas como PyTorch, Diffusers de Hugging Face [43] y la web-ui de Stable Diffusion de Automatic1111 [44]. Realizar una implementación en PyTorch puede llevar mucho trabajo debido a la necesidad de definir, construir y validar el funcionamiento de algunos componentes como la U-Net y el VAE. Por otro lado, la web-ui de *Stable Diffusion* es de muy alto nivel, no se requiere mucho conocimiento sobre los modelos de difusión para realizar inferencias y hasta hacer un *fine-tuning* con imágenes propias, pero el entrenamiento es más de "caja negra" en comparación con las otras dos alternativas. Por esta razón se decidió utilizar Diffusers, que parece ser un término medio entre lograr resultados dentro de un marco temporal razonable, pero también realizar implementaciones y generar un entendimiento de cómo se construyen estas soluciones.

Se siguió un enfoque gradual en la implementación, comenzando con casos de prueba más simples antes de abordar el problema más complejo de imágenes médicas. Inicialmente, se desarrolló código funcional para las técnicas de LoRA y Textual Inversion utilizando ejemplos de prueba simples. Estos casos de prueba más sencillos permitieron confirmar que las redes estaban aprendiendo correctamente los conceptos. Posteriormente, las técnicas se aplicaron a un subconjunto curado de los datos disponibles de lesiones de piel, realizando ajustes según fuera necesario para adaptarlas de forma de obtener resultados satisfactorios.

3.5 Optimización y mejoras

En la sección de selección e implementación, se desarrollan actividades que permiten lograr resultados razonables para el caso de uso de la imagenología médica. De todas formas, se presenta una etapa posterior de optimización y mejoras, con el objetivo de potenciar la calidad de los resultados obtenidos emprendiendo un proceso de experimentación, tal que resulten útiles en relación a los objetivos de este proyecto.

Reconociendo la importancia de las prompts utilizadas, se realizó un proceso de prompt engineering explorando distintas modificaciones sobre las mismas, con el propósito de mejorar la calidad y especificidad de las imágenes. En particular, se busca que presenten las lesiones adecuadas, texturas deseadas y otros elementos habituales que las imágenes reales también contienen; como fluidos, bellos, entre otros. Por otro lado, también se realizó prompt engineering sobre las *prompts negativas*, de forma de eliminar ciertas texturas, colores y artefactos no deseados en las imágenes generadas, buscando que la imagen creada parezca más realista.

Asimismo, se realizó una experimentación para la optimización de los hiperparámetros más críticos, tanto del proceso de entrenamiento como de inferencia. A modo de ejemplo, durante el entrenamiento se modifican hiperparámetros como la tasa de aprendizaje, la cantidad máxima de pasos de entrenamiento, resoluciones de salida e imágenes utilizadas de entrada. Por otro lado, también se experimenta sobre los parámetros más relevantes en la inferencia, como la cantidad de pasos de inferencia y lo mencionado anteriormente de prompt engineering.

En este proyecto se busca como objetivo principal explorar qué técnicas proveen mejores resultados para lograr generación de imágenes médicas de calidad. Sin embargo, también se plantean objetivos relacionados a aumentar la cantidad de datos de poblaciones subrepresentadas dentro de los *datasets*, y hasta explorar la generación sobre poblaciones no representadas en los mismos. Debido a este punto, es que se realizan otros experimentos buscando mejorar los resultados para estos casos específicos.

3.6 Utilidad en problemas de clasificación

La evaluación de la utilidad de las imágenes generadas en el ámbito de problemas de clasificación constituye un elemento esencial en nuestra investigación. Con el propósito de comprender en qué medida las imágenes generadas desempeñan un papel significativo en la mejora de *performance* de modelos de clasificación, se realizan distintos experimentos enfocados a generar *insights* sobre dicho impacto y bajo qué condiciones se da.

En primer lugar, se construye un modelo de clasificación utilizando únicamente imágenes reales del dataset seleccionado sobre lesiones de piel. Sobre este modelo se trabajará realizando *fine-tuning* y optimización de hiperparámetros hasta alcanzar un nivel de razonable en comparación con los benchmarks más destacados para este *dataset*. Esta base servirá como punto de comparación para evaluar el impacto de las imágenes generadas en el proceso de entrenamiento.

El siguiente paso consiste en utilizar este modelo base como punto de partida y aplicar distintas estrategias de entrenamiento, incorporando imágenes generadas en el proceso. Se exploran distintos enfoques como la utilización de imágenes generadas para un primer *fine-tuning* de un modelo pre-entrenado al utilizar *transfer learning*, para luego realizar una segunda etapa de f*ine-tuning* solamente con imágenes reales. Otra alternativa consiste en realizar el *fine-tuning* del modelo pre-entrenado con una combinación de imágenes reales y generadas, evaluando su rendimiento únicamente en imágenes reales.

El análisis se centrará en la comparación de la *performance* de los modelos resultantes, de aquel modelo entrenado exclusivamente con imágenes reales en comparación con aquellos enriquecidos con las imágenes generadas. Mediante estas comparaciones, se busca comprobar si es cierta la hipótesis de que las imágenes generadas son útiles para mejorar la precisión y robustez de los modelos de clasificación en el ámbito de la imagenología médica.

A modo de generar un análisis más rico en cuanto a los hallazgos de esta investigación, se lleva a cabo una serie de experimentos buscando determinar la influencia de distintos factores sobre los resultados. En particular, se busca responder cómo varía el desempeño según la proporción de imágenes reales y generadas utilizadas, así como determinar el impacto de la utilización de imágenes generadas para aumentar la cantidad de datos en las clases menos representadas.

Se busca generar información valiosa que pueda ser utilizada de referencia a la hora de construir soluciones de *Machine Learning* para problemas de imagenología médica.

4. IMPLEMENTACIÓN

En esta sección se detalla el proceso de implementación de las técnicas seleccionadas en el contexto de modelos de difusión para imágenes médicas. Se describen los frameworks utilizados y los recursos necesarios para la ejecución de los experimentos. Se comentan los aspectos destacados de la implementación de las técnicas seleccionadas: Textual Inversion, LoRA y LoRA aplicado a Stable Diffusion XL como un caso especial. Finalmente, se culminará esta sección realizando comentarios generales sobre el proceso de implementación y las lecciones aprendidas en el camino.

4.1 Frameworks y hardware utilizados

Como fue mencionado anteriormente, se decidió utilizar principalmente la librería Diffusers de Hugging Face por sobre la UI de Stable Diffusion de Automatic1111 debido a que es de muy alto nivel y no se visualiza la lógica del proceso de difusión descrita en la sección del marco teórico. Por otro lado, implementar todo de cero en PyTorch puede resultar costoso en tiempo y esfuerzo, y podría dificultar la integración con modelos pre-entrenados. De todas formas, se utiliza esta librería de base que define los componentes, funciones, *pipelines* y modelos pre-entrenados principales para implementar soluciones que utilizan modelos de difusión, pero se utiliza PyTorch para las implementaciones de las técnicas descritas como Textual Inversion y LoRA.

Los tres principales elementos que provee la librería Diffusers son:

- **Diffusion pipelines:** clases que envuelven los distintos componentes y su interacción, como de los modelos de difusión, como los VAEs, text encoders, U-Net, scheduler, etc.
- **Schedulers:** distintos métodos de schedulers para introducir ruido durante el proceso de forward diffusion.
- **Modelos pre-entrenados:** modelos pre-entrenados, tanto su arquitectura como sus pesos, para utilizarlos dentro de soluciones que involucren modelos de difusión.

En cuanto al hardware utilizado, debido al tiempo que puede requerir realizar el *fine-tuning* e inferencia de los modelos de difusión en CPU, se decidió utilizar las GPUs que provee Google Colab. Se tenía una computadora con GPU RTX 1650 que posee 4 GB de VRAM en la GPU, pero estos modelos requieren de al menos 6 GB, por lo que no fue posible utilizarla.

4.2 Baseline

Como fue mencionado en la sección de metodología, se realizó una primera experimentación utilizado un modelo pre-entrenado de Stable Diffusion para generar imágenes de distintos tipos de lesiones y poder evaluar cuál es la calidad del modelo pre-entrenado para generar las mismas sin ningún tipo de fine-tuning.

Se realizó una experimentación modificando distintos parámetros en la inferencia, como los pasos de inferencia, la *prompt* utilizada (se realizó *prompt engineering*) y utilizando algunos

elementos en la *negative prompt* para evitar ciertos artefactos o elementos en las imágenes generadas. A continuación se muestran algunas imágenes de los resultados.

Figura 4.1 - Imágenes de referencia obtenidas al generar distintos tipos de lesión con el modelo 1.5 de Stable Diffusion pre-entrenado sin ningún tipo de fine-tuning y luego de un proceso de prompt engineering.

Como se puede observar en las imágenes anteriores, si bien el modelo entiende que se busca la presencia de ciertos elementos en la piel de personas, las lesiones generadas no son realistas. A su vez, presentan las imágenes desde una distancia mayor a las presentes en el dataset seleccionado, por más de que se intentó de realizar una imagen más cercana a la piel por medio de *prompt engineering*.

4.3 Textual inversion

La implementación de Textual Inversion se realizó con los modelos estándar de Stable Diffusion, sin utilizar Stable Diffusion XL, tomando como referencia la versión 1.4 disponible en Hugging Face cuyo identificador es CompVis/stable-diffusion-v1-4 [45].

Para esto, como fue mencionado en la sección del marco teórico, se debe cargar el *encoder* de texto, el tokenizer, el VAE, la U-Net, el scheduler para manejar el proceso de *forward diffusion*. Por otro lado, se debe agregar el nuevo token utilizado para el concepto que se le quiere enseñar a la red dentro del corpus del *tokenizer* e inicializar el *embedding* de la capa de embedding dentro del *encoder* de texto con un token conocido de referencia. A continuación se muestra un extracto de código dónde se realizan estas cargas e inicializaciones.

```
from diffusers import AutoencoderKL DDPMScheduler UNet2DConditionModel
from transformers import CLIPTextModel, CLIPTokenizer


class TextualInversionTrainer:
    """
    Class for handling the finetuning of the Stable Diffusion model using
    Textual Inversion for teaching the model a specific concept.
    """

    def __init__(
```

```
        self, concept_name: str, placeholder_token: str,
        initializer_token: str, model_id: str
    ) -> None:

        ...
        self.model_id = model_id  # CompVis/stable-diffusion-v1-4
        self.concept_name = concept_name
        self.initializer_token = initializer_token
        self.placeholder_token = placeholder_token

        self._initialize_tokenizer()
        self._get_special_tokens(initializer_token)
        self._load_diffusion_model()
        self._freeze_models()

    def _initialize_tokenizer(self) -> None:
        # Load prefitted tokenizer
        self.tokenizer = CLIPTokenizer.from_pretrained(
            self.model_id,
            subfolder="tokenizer",
        )

        # Add the placeholder token in tokenizer
        num_added_tokens = self.tokenizer.add_tokens(
            self.placeholder_token
        )

    def _get_special_tokens(self, initializer_token: str) -> None:
        # Get the token id for the initializer token
        token_ids = self.tokenizer.encode(
            initializer_token, add_special_tokens=False
        )
        if len(token_ids) > 1:
            raise ValueError(
                "The initializer token must be a single token."
            )

        # Set the token ids for the initializer and placeholder tokens
        self.initializer_token_id = token_ids[0]
        self.placeholder_token_id = self.tokenizer.convert_tokens_to_ids(
            self.placeholder_token
        )

    def _load_diffusion_model(self) -> None:
        # Load the text encoder, VAE, U-Net and Scheduler
        self.text_encoder = CLIPTextModel.from_pretrained(
            self.model_id, subfolder="text_encoder"
        )
        self.vae = AutoencoderKL.from_pretrained(
            self.model_id, subfolder="vae"
        )
```

```
        self.unet = UNet2DConditionModel.from_pretrained(
            self.model_id, subfolder="unet"
        )
        self.noise_scheduler = DDPMScheduler.from_config(
            self.model_id, subfolder="scheduler"
        )

        # Initialize the newly added placeholder token with the
        # embeddings of the initializer token
        self.text_encoder.resize_token_embeddings(len(self.tokenizer))
        token_embeds = self.text_encoder.get_input_embeddings() \
            .weight.data
        token_embeds[self.placeholder_token_id] = token_embeds[
            self.initializer_token_id
        ]
```

Recorte de código 4.1 - Definición de la clase de fine-tuning de Stable Diffusion con Textual Inversion.

Si bien la adición del nuevo token al *tokenizer* y la inicialización del *embedding* de dicho token con el de referencia, son lógicas particulares de Textual Inversion y no de LoRA, el código anterior también refleja de manera genérica cómo se implementó la carga de los distintos componentes que fueron utilizados en cada enfoque.

En el caso de Textual Inversion, se mantienen todos los parámetros de los distintos modelos congelados, dejando únicamente entrenable el *embedding* del *text encoder*. A continuación se muestra el método que realiza esta definición.

```
    @staticmethod
    def _freeze_params(params) -> None:
        for param in params:
            param.requires_grad = False

    def _freeze_models(self) -> None:
        # Freeze VAE and U-Net
        self.freeze_params(self.vae.parameters())
        self.freeze_params(self.unet.parameters())

        # Freeze all parameters except for the token embeddings in
        # the text encoder
        params_to_freeze = itertools.chain(
            self.text_encoder.text_model.encoder.parameters(),
            self.text_encoder.text_model.final_layer_norm.parameters(),
            self.text_encoder.text_model.embeddings \
                .position_embedding.parameters(),
        )
        self.freeze_params(params_to_freeze)
```

Recorte de código 4.2 - Definición de los parámetros entrenables para el fine-tuning de Stable Diffusion con Textual Inversion.

Se realizó una primera implementación buscando lograr que comience el loop de entrenamiento del modelo, cuyo código completo se puede visualizar en el repositorio de GitHub del proyecto. Una vez que éste parecía correr sin inconvenientes, se migra el código correspondiente a Google Colab para correr el entrenamiento utilizando sesión con una GPU T4 en un caso de uso de prueba, donde el aprendizaje de la generación fuera algo más sencillo que una lesión en la piel. Una vez que se lograron obtener resultados buenos para el caso de prueba, se procedió a correr para el caso de lesiones de piel.

Un primer inconveniente que encontró el equipo al obtener los primeros resultados con Textual Inversion, se debían a la utilización de *prompts* que se tenían como referencia para un *fine-tuning* más genérico, pero que no aplicaban al caso de uso. Esto se pudo solventar cambiando dichas prompts por unas más específicas.

Por otro lado, si bien se había definido qué parámetros de los distintos modelos debían congelarse en el método mostrado anteriormente, esto no se estaba aplicando en un principio, por lo que quedaban todos los parámetros de todos los modelos en estado entrenable, lo cual derivó en resultados malos, similares a imágenes galácticas y no a lesiones de piel. Una vez corregidos estos errores, se lograron mejorar un poco los resultados aunque seguían sin ser buenos.

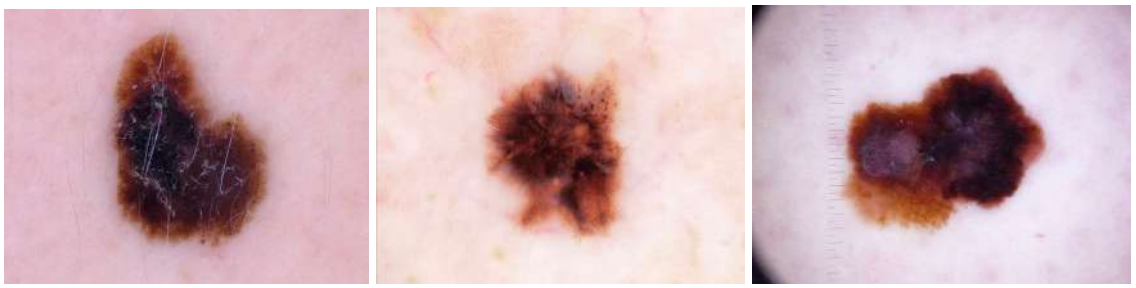


Figura 4.2 - Imágenes reales de referencia utilizadas para realizar fine-tuning con Textual Inversion

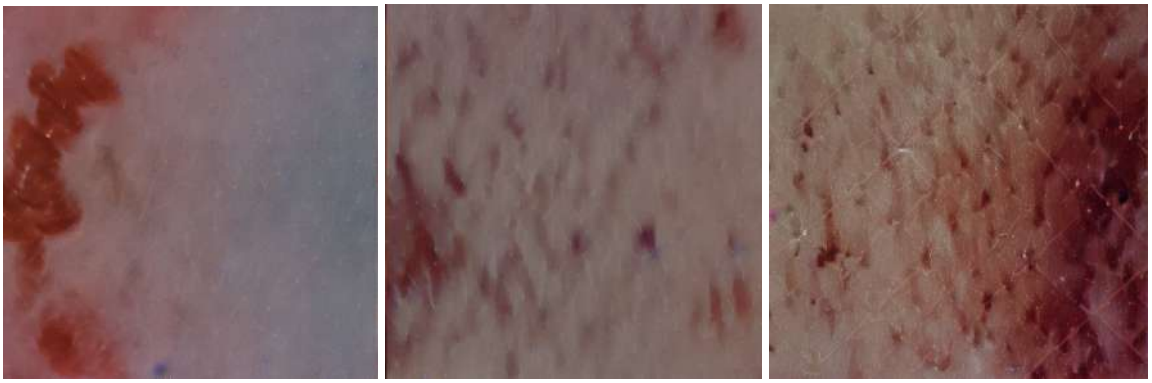


Figura 4.3 - Imágenes de melanomas generadas con Textual Inversion

Como se puede observar en la figura anterior, las imágenes generadas no se asemejan a los patrones que se observan en las imágenes reales. Se realizaron varias pruebas con distintas cantidades de pasos de entrenamiento y modificando algunos de los hiperparámetros más importantes pero siguió sin brindar los resultados buscados. Se realizó una prueba con una

notebook que proveen los autores de la librería Diffusers, de forma de validar si había algún error en el código desarrollado, pero se obtuvieron resultados similares.

Para este *fine tuning* se utilizaron 7 imágenes de melanomas similares para poder observar si era posible obtener la calidad esperada. El entrenamiento transcurrió por 53 minutos utilizando una GPU T4 que provee Google Colab de manera gratuita. El peso del archivo resultante del *embedding* entrenado tiene un tamaño en disco de 4 KB.

4.4 LoRA

La implementación de LoRA en la presente sección también se realizó tomando como referencia el mismo modelo utilizado para Textual Inversion con el mismo conjunto de imágenes para el *fine-tuning*.

De la misma manera que con Textual Inversion, se debe cargar el *encoder* de texto, el *tokenizer*, el VAE, la U-Net, el *scheduler* para manejar el proceso de *forward diffusion*. Sin embargo, en este caso se congelan todos los parámetros de todos los modelos pre-entrenados cargados, incluyendo la capa de embeddings del *encoder* de texto. Los parámetros entrenables son las capas de LoRA que se agregan para modificar la salida de las capas de atención dentro de la U-Net, a pesar de que las capas de atención en sí permanecen no entrenables.

A continuación se muestra el método dónde se implementaron las capas de LoRA, que se instancian a partir de las capas de atención existentes en la red. Se definieron dichas capas de forma dinámica, de manera de que el código sea adaptable a distintos modelos de U-Net para distintos modelos pre-entrenados de difusión.

```
from diffusers.models.attention_processor import LoRAAttnProcessor

class LoRATrainer:
    "Class for training the LoRA weights based on the Diffusers library"
    ...

    def _set_lora_layers(self):
        """Method for setting the LoRA layers within the U-Net"""

        lora_attn_procs = {}
        for name in self.unet.attn_processors.keys():
            cross_attention_dim = (
                None
                if name.endswith("attn1.processor")
                else self.unet.config.cross_attention_dim
            )
            if name.startswith("mid_block"):
                hidden_size = self.unet.config.block_out_channels[-1]
            elif name.startswith("up_blocks"):
                block_id = int(name[len("up_blocks.")])
                hidden_size = list(reversed(
```

```
                    self.unet.config.block_out_channels
                ))[block_id]
            elif name.startswith("down_blocks"):
                block_id = int(name[len("down_blocks.")])
                hidden_size = self.unet.config.block_out_channels[
                   block_id
                ]

            lora_attn_procs[name] = LoRAAttnProcessor(
                hidden_size=hidden_size,
                cross_attention_dim=cross_attention_dim,
            )

        self.unet.set_attn_processor(lora_attn_procs)
```

Recorte de código 4.3 - Definición de las capas de LoRA dentro de las capas de atención de la U-Net para el fine-tuning de Stable Diffusion con LoRA.

En el anterior recorte de código se muestra que, a partir de las propiedades de las capas de atención presentes en la U-Net, es que se van instanciando las nuevas capas de LoRA en la red. Estas nuevas capas serán los parámetros entrenables durante el fine-tuning de la red.

De manera análoga a Textual Inversion, en una primera instancia se desarrolló el código para el *fine-tuning* de Stable Diffusion con LoRA, buscando llegar hasta un código que corra el loop de entrenamiento. Luego, se pasó el código a Google Colab con GPU, y se entrenó un primer modelo para un concepto más sencillo que el de las lesiones de piel.

Una vez que se validó que las capas de LoRA estaban siendo entrenadas de forma de aprender el nuevo concepto, es que se procedió al entrenamiento de los mismos melanomas que con Textual Inversion.

Algunos aspectos importantes que se aprendieron durante la experimentación y validación con este tipo de lesiones de piel, es que se identificó el impacto que puede tener entrenar por más o menos épocas, durante la etapa de entrenamiento, y la diferencia en la calidad de los resultados que se puede obtener al variar el parámetro de la cantidad de pasos de inferencia, durante la etapa de inferencia. A continuación se muestran imágenes de lesiones del tipo vascular para distintos valores de máxima cantidad de pasos de entrenamiento.

Figura 4.4 - Impacto de la variación de los steps de entrenamiento (100 steps, 300 steps y 500 steps, respectivamente) para un mismo número de steps de inferencia en el entrenamiento con LoRA

Figura 4.5 - Imágenes de melanomas generadas con LoRA

Esto resultó en una extensa etapa de experimentación en donde, para cada tipo de lesión, se guardaron *checkpoints* del modelo para distintos steps de entrenamiento y se compararon sus resultados, de forma de poder identificar qué cantidad de steps genera mejores resultados para cada tipo de lesión.

Una vez tomada esta decisión, se procedió a una nueva experimentación, en donde para cada modelo escogido de cada tipo de lesión, se probó la generación de imágenes con distintos pasos de inferencia, de manera de poder comparar sus resultados. Se observó que, para un número de pasos de inferencia menor al óptimo, no se logran texturas definidas y se pueden observar ciertas regiones de la imagen con algunos píxeles que parecen similares a ruido gaussiano. Al aumentar la cantidad de pasos de inferencia, se logran mejores texturas y dichos píxeles desaparecen en su mayor parte. Sin embargo, al definir una cantidad de pasos de inferencia mayor a la óptima, se logran imágenes con pieles y lesiones sobre texturizadas, generando imágenes irrealistas. A continuación se puede observar una imagen representativa de dicha experimentación sobre el caso de prueba de un juguete con forma de gato.

Figura 4.6 - Imágenes reales de referencia utilizadas para el caso de prueba.

Figura 4.7 - Impacto de la variación de los steps de inferencia en el caso de prueba, para un mismo modelo en el entrenamiento con LoRA. Las tres imágenes superiores representan la generación del concepto "cat-toy" con 10, 30 y 50 steps de inferencia respectivamente. Las tres imágenes inferiores fueron generadas con 100, 200 y 300 steps de inferencia respectivamente.

En la figura anterior se puede observar no sólo la mejora en la remoción de ruido conforme aumenta la cantidad de pasos de inferencia, sino que entre las imágenes con 30 y 50 pasos de inferencia, comienza la aparición de la cola del juguete. Estas seis muestras generadas parten de la misma semilla aleatoria, de manera de poder observar estas diferencias.

Un comentario adicional, es que aumentar la cantidad de pasos de inferencia no siempre mejora la calidad y realismo de las imágenes generadas. Normalmente este aumento tiende a resaltar la nitidez y definición más notoria de bordes, que en casos como las pieles de personas puede resultar irrealista. A continuación se presenta un ejemplo de este fenómeno para el caso de dermatofibroma.

Figura 4.8 - Impacto de la utilización de pasos de inferencia superior al óptimo en dermatofibromas. A la izquierda se presenta la generación de un dermatofibroma con 50 pasos de inferencia, mientras que a la derecha, un dermatofibroma generado con el mismo modelo con 400 pasos de inferencia

El entrenamiento de LoRA transcurrió por 1 hora y 15 minutos en la instancia de Google Colab con una T4. El peso del archivo resultante de las capas de LoRA tiene un tamaño en disco de 3.3 MB.

4.5 Stable Diffusion XL

Con los resultados anteriores, se obtuvo una buena aproximación a la calidad de las imágenes buscadas; sin embargo, aún no se habían alcanzado los resultados esperados. Es por esta razón que se tomó provecho del lanzamiento reciente de Stable Diffusion XL (SDXL), un modelo lanzado a fines de junio del 2023 que surgió como una mejora a Stable Diffusion de forma de crear imágenes más realistas, mejorando detalles finos como caras y dedos de las personas que en Stable Diffusion no se lograba aún, añadiendo la capacidad de introducir texto dentro de las imágenes y mayor control de la generación. Los modelos pre-entrenados de SDXL utilizados fueron descargados de Hugging Face, que tienen el identificador stabilityai/stable-diffusion-xl-base-1.0 [46]. Por otro lado, el VAE utilizado tiene el siguiente identificador: madebyollin/sdxl-vae-fp16-fix [47].

En cuanto a su implementación, al utilizar LoRA, gran parte del código realizado para el modelo de Stable Diffusion pudo reutilizarse. Sin embargo, debido a las incorporaciones de más modelos y diferencias en tamaños de la cantidad de parámetros, se tuvieron que considerar estos nuevos modelos en el flujo de entrenamiento y realizar algunas técnicas para reducir la memoria en GPU, de forma tal de que pudiera entrar en una sola GPU gratuita en Google Colab, y otras técnicas para estabilizar el entrenamiento.

En particular, se utiliza la precisión fp16 tanto para el entrenamiento como para inferencia. Esto hace referencia a que, en vez de utilizar la precisión habitual fp32, que utiliza 4 bytes para representar los parámetros de las redes, se utilizan 2 bytes, teniendo una menor precisión, lo cual permite que los parámetros de los modelos ocupen la mitad de su tamaño en memoria, reduciendo la cantidad de memoria RAM en GPU que ocupan los mismos. En

caso de no bajar la precisión de estos parámetros, el entrenamiento no sería posible en la T4 gratuita de Google Colab.

Por otro lado, durante el entrenamiento es común utilizar alguna variante del optimizador Adam para entrenar y hacer *fine-tuning* de estos modelos. Sin embargo, Adam almacena un histórico del estado de todos los parámetros en los últimos pasos, lo que implica una mayor cantidad de parámetros también en memoria. Por esta razón, es que se utiliza una implementación del optimizador AdamW en 8 bits (1 byte) que provee la librería bitsandbytes. Al igual que con la precisión de los modelos, en caso de no utilizar este optimizador con esta precisión, no sería posible realizar el entrenamiento en la T4 de Google Colab debido a falta de memoria VRAM en la GPU.

En cuanto al *batch size* utilizado, no fue posible mantener las 8 imágenes por *batch* utilizadas para entrenar el modelo de Stable Diffusion original con LoRA, debido a que, a mayor tamaño de *batch*, mayor es el consumo de memoria en GPU. Se tuvo que utilizar un *batch* de 2 imágenes. Al utilizar un *batch* tan pequeño, puede llegar a causar cierta inestabilidad adicional en el entrenamiento y se aumenta el tiempo de entrenamiento. En la literatura, y en particular en recomendaciones de blogs de Hugging Face sobre la utilización de Stable Diffusion, autores de la librería utilizada y de donde se descargan los modelos pre-entrenados aconsejan la utilización de la técnica de acumulación de gradientes. Dicha técnica refiere a realizar inferencias sobre n *batches* de manera secuencial, calculando sus gradientes, pero sin realizar *backpropagation* de la red luego de cada uno de ellos. Una vez que se realiza el paso hacia adelante de los n batches y se tienen los gradientes de cada batch, se realiza un promedio de estos para actualizar los pesos de la red.

Si bien los componentes son similares a los que se tienen con Stable Diffusion, al tener dos modelos de difusión, el generador y el refinador, se requiere de dos *encoders* de texto y dos *tokenizers*, además de la U-Net y el VAE.

```
from diffusers import AutoencoderKL, DDPMScheduler, UNet2DConditionModel
import torch
from transformers import (
    AutoTokenizer,
    CLIPTextModel,
    CLIPTextModelWithProjection,
    PretrainedConfig,
)


class StableDiffusionXLLoRATrainer:
    """Class for training the LoRA weights for Stable Diffusion XL"""

    def __init__(self,
        concept_name: str, model_id: str, vae_model_id: str
    ) -> None:

        self.concept_name = concept_name
        self.model_id = model_id
```

```
        self.vae_model_id = vae_model_id

        device = "cuda" if torch.cuda.is_available() else "cpu"
        self.device = torch.device(device)
        self.load_models()

    def load_models(self):
        # Noise scheduler
        self.noise_scheduler = DDPMScheduler.from_pretrained(
            self.model_id, subfolder="scheduler"
        )

        # Load tokenizers
        self.tokenizer_one = AutoTokenizer.from_pretrained(
            self.model_id, subfolder="tokenizer", use_fast=False
        )
        self.tokenizer_two = AutoTokenizer.from_pretrained(
            self.model_id, subfolder="tokenizer_2", use_fast=False
        )

        # Text encoders
        self.text_encoder_cls_one = self._get_text_encoder(
            subfolder="text_encoder"
        )
        self.text_encoder_cls_two = self._get_text_encoder(
            subfolder="text_encoder_2"
        )
        self.text_encoder_one = self.text_encoder_cls_one.from_pretrained(
            self.model_id, subfolder="text_encoder"
        )
        self.text_encoder_two = self.text_encoder_cls_two.from_pretrained(
            self.model_id, subfolder="text_encoder_2"
        )

        # Models (VAE + U-Net)
        self.vae = AutoencoderKL.from_pretrained(
            self.vae_model_id, subfolder="vae"
        )
        self.unet = UNet2DConditionModel.from_pretrained(
            self.model_id, subfolder="unet"
        )

        # Freeze models parameters
        self.unet.requires_grad_(False)
        self.vae.requires_grad_(False)
        self.text_encoder_one.requires_grad_(False)
        self.text_encoder_two.requires_grad_(False)

    def _get_text_encoder(self, subfolder: str):
        """Get the correct text encoder class from the model id"""
```

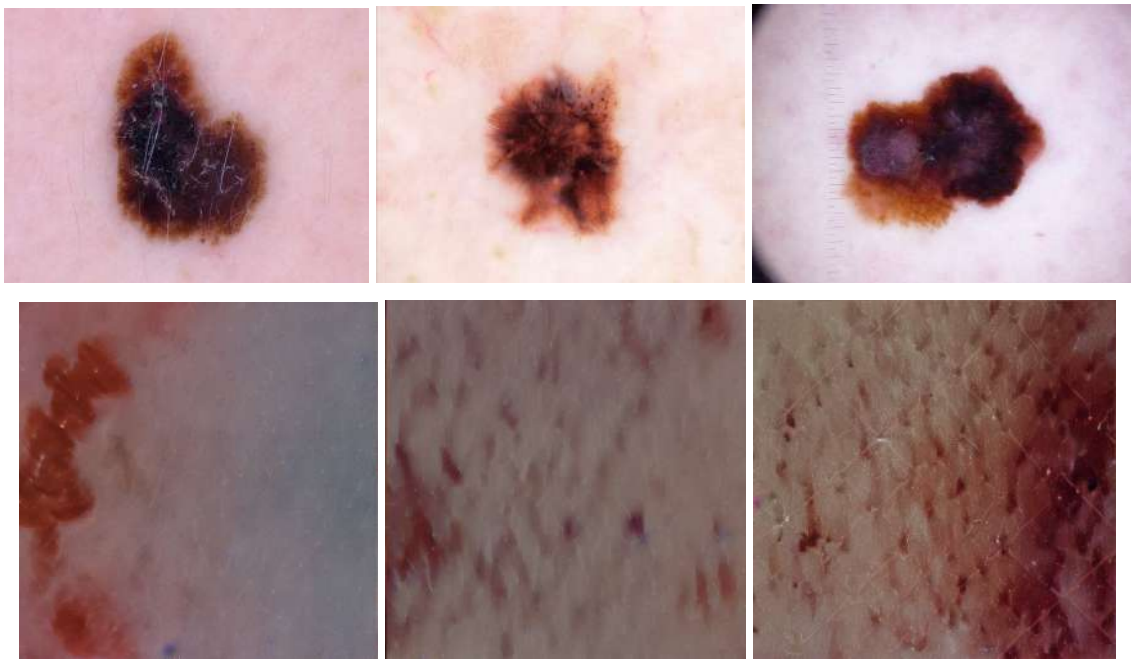
```
        text_encoder_config = PretrainedConfig.from_pretrained(
            self.hyperparameters["model_id"], subfolder=subfolder
        )
        model_class = text_encoder_config.architectures[0]

        if model_class == "CLIPTextModel":
            return CLIPTextModel
        elif model_class == "CLIPTextModelWithProjection":
            return CLIPTextModelWithProjection
        else:
            raise ValueError(f"{model_class} is not supported.")
```

Recorte de código 4.4 - Definición de los componentes principales para el fine-tuning de Stable Diffusion XL con LoRA.

Nuevamente se utilizaron las mismas imágenes de melanomas para realizar un primer *fine-tuning* y evaluar los resultados, que en este caso fueron muy buenos. El entrenamiento transcurrió durante 2 horas y media en una T4 dentro de Google Colab.

A continuación se presenta una comparativa de resultados entre los modelos de Stable Diffusion aplicando Textual Inversion y LoRA, y los resultados de este nuevo modelo de Stable Diffusion XL aplicando LoRA. A su vez se dejan imágenes reales como referencia.



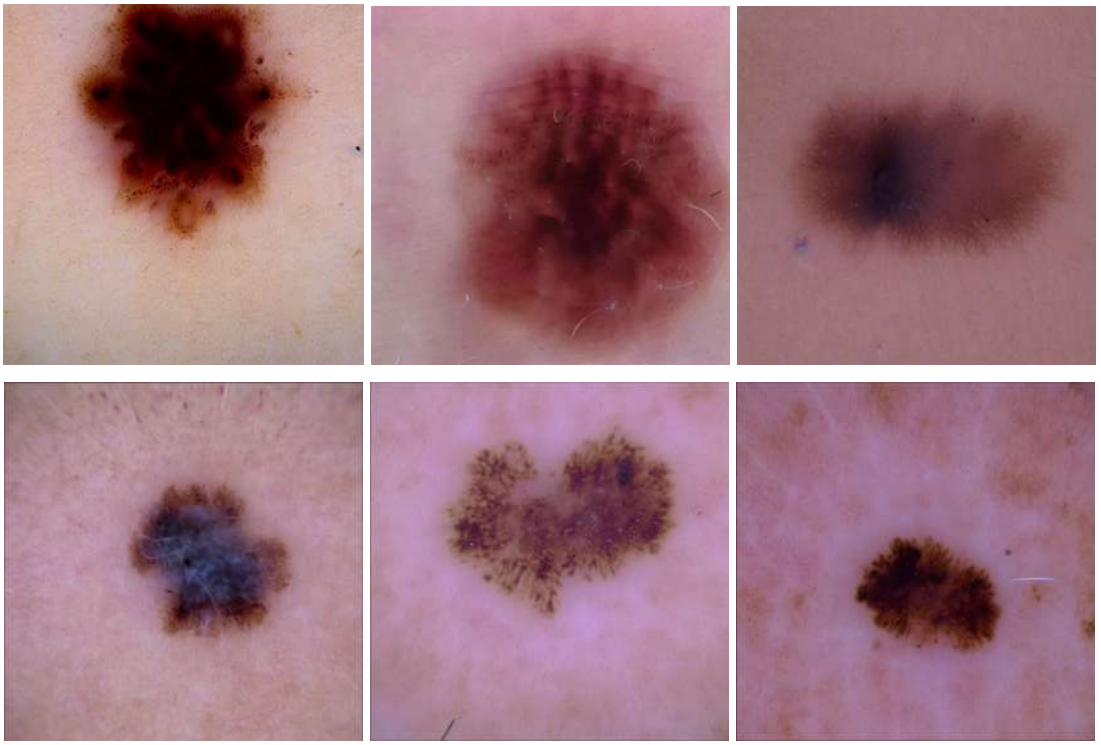


Figura 4.9 - Muestra de los resultados obtenidos para el caso de melanomas. La primera fila de imágenes corresponde a tres de las imágenes reales utilizadas para entrenar los modelos. Las últimas tres filas corresponden a los resultados obtenidos al realizar fine-tuning de: Stable Diffusion con Textual Inversion, Stable Diffusion con LoRA y SDXL con LoRA, respectivamente.

El enfoque de SDXL con LoRA fue el que brindó mejores resultados, acercándose mucho a la calidad de las imágenes reales. Si bien esta valoración es subjetiva y cualitativa, al analizar las imágenes obtenidas con mayor detenimiento y resolución que las presentadas en este informe, se hace más notoria diferencia. De todas formas, en la figura anterior se puede apreciar la diferencia entre las tres técnicas implementadas.

4.6 Generación de clases no representadas

Durante el comienzo del proyecto, se habló de que muchos datasets abiertos se encuentran sesgados, en el sentido de que, por ejemplo, no se encuentran imágenes de personas con distintos tonos de piel.

Por esta razón es que se realizó una breve experimentación en la que, con un modelo entrenado de SDXL, se buscó modificar la prompt de entrada de manera de alterar aspectos físicos de la persona donde se genera la lesión. Sin embargo, no se obtuvieron buenos resultados.

El equipo de trabajo intuye que se puede deber al hecho de que todas las imágenes utilizadas para el entrenamiento tienen tanto la lesión como la piel. De esta forma, quizás la red neuronal no tenga suficientes elementos como para discernir qué es lo que se busca generar, tomando como que el concepto para el cual se realiza el *fine-tuning* es el conjunto lesión de piel / piel. Al no ser el foco principal de este proyecto, se dejó de lado esta experimentación al obtener estos resultados, pero se consideró relevante mencionar este punto para que en futuros trabajos se tenga en cuenta que puede ser necesario tratar de

separar el concepto de su contexto para que la red pueda interpretar cuál es el concepto que debe aprender.

4.7 Clasificación de lesiones

En cuanto a la utilización de imágenes generadas como forma de data augmentation para mejorar las métricas obtenidas al entrenar modelos de clasificación de imágenes médicas, se realiza el código de entrenamiento de una red convolucional multiclase para clasificar distintos tipos de lesiones de piel.

En una primera instancia, se busca atacar el problema de clasificación utilizando únicamente los datos reales del *dataset*, de forma de tener un resultado que pueda ser contrastado con entrenar una red neuronal que utilice imágenes generadas como parte de su entrenamiento. A estos efectos, se realizó la experimentación en tres etapas:

1. Generar un procesamiento de las imágenes para crear la serie de carpetas deseadas con las imágenes correspondientes de manera que puedan ser tomadas luego en el entrenamiento.
2. Realizar código funcional que permita efectuar el entrenamiento de la red a partir de dichas carpetas de imágenes, utilizando métricas de validación para el monitoreo.
3. Realizar modificaciones sobre la arquitectura e hiperparámetros para lograr un buen desempeño de la red convolucional utilizando solamente las imágenes reales.

Luego, en una segunda instancia, se utiliza de base la misma arquitectura e hiperparámetros, pudiendo modificar levemente algunos elementos para lograr el mejor desempeño posible del modelo utilizando tanto imágenes reales como generadas.

Estos dos resultados son luego comparados para evidenciar el impacto que tiene la utilización de imágenes generadas, junto con algunos hallazgos sobre experimentos que se consideran de interés, como los mencionados sobre el impacto de la proporción de imágenes generadas utilizadas y el impacto del tamaño del *dataset* con imágenes reales.

4.5.1 Estructura de carpetas de imágenes

Debido a una mayor facilidad de implementación y prototipado, se decide utilizar TensorFlow como *framework* para el problema de clasificación. Por este motivo, se utiliza la clase ImageDataGenerator para obtener las imágenes desde las carpetas correspondientes y aplicar técnicas de image *data augmentation*. Al ser un problema de clasificación multiclase, se requiere que las imágenes almacenadas sigan determinada estructura, por lo que se almacenan utilizando la siguiente nomenclatura:

`data/classification/{nombre_experimento}/{nombre_del_set}/{clase}/{id_imagen}.jpg`

Donde:

- El nombre del experimento refiere a qué experimento se está realizando. Por ejemplo: *baseline* es el utilizado para sacar métricas sobre los datos reales, sin datos generados.
- El nombre del set se refiere a si son imágenes de entrenamiento, validación o test.
- Clase refiere al nombre del tipo de lesión; por ejemplo: melanoma o vascular.
- El id de la imagen es el identificador que tiene cada una de éstas en el dataset original, o el nombre asignado en caso de ser imágenes generadas.

En todos los experimentos se utilizan imágenes reales tanto para validación como para test. A modo de ejemplo se muestra la estructura para el experimento baseline.

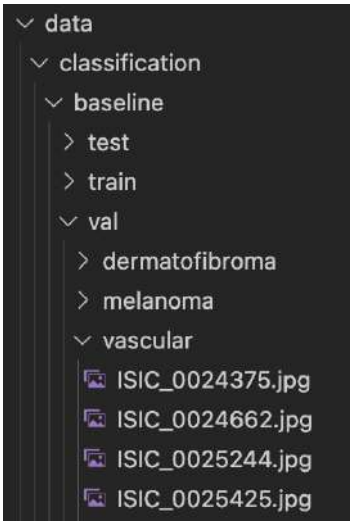


Figura 4.10 - Estructura de carpetas de las imágenes para el entrenamiento del modelo de clasificación.

4.5.2 Arquitectura del modelo

Para la red convolucional (CNN, por sus siglas en inglés), se define realizar *Transfer Learning* sobre algún modelo pre-entrenado, quitando las capas densas para la clasificación y agregando nuevas capas densas para definir la salida necesaria para el problema de la clasificación de lesiones de piel.

Se construyó de manera de poder probar distintos modelos pre-entrenados, donde se terminó utilizando MobileNet como base debido a que otros modelos más complejos tienden a sobreajustar sobre los datos de entrenamiento cuando se tienen pocos datos.

```
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (
    BatchNormalization,
    Dense,
    Dropout,
    GlobalAveragePooling2D,
    Input,
)
```

```
class SkinLesionCNN:
    ...

    def _get_model(self) -> Model:
        # Backbone
        backbone = MobileNet(
            weights="imagenet",
            input_shape=tuple(self.images_shape),
            include_top=False
        )
        backbone.trainable = True

        # Network definition
        inputs = Input(shape=tuple(self.images_shape))
        x = backbone(inputs)
        x = GlobalAveragePooling2D()(x)
        x = Dropout(0.3)(x)

        x = Dense(256, activation="leaky_relu")(x)
        x = BatchNormalization()(x)
        x = Dropout(0.2)(x)

        x = Dense(128, activation="leaky_relu")(x)
        x = BatchNormalization()(x)
        x = Dropout(0.15)(x)

        out = Dense(5)(x)
        model = Model(inputs=inputs, outputs=out)
        return model
```

Recorte de código 4.5 - Definición de la arquitectura de la CNN de clasificación.

En la última neurona no se especifica *softmax* como función de activación porque se utiliza como función de pérdida la función *categorical cross entropy* con el parámetro *from_logits* como verdadero, de forma que no es necesario especificar dicha activación.

4.5.3 Técnicas para evitar el sobreajuste

Debido a que la generación de las imágenes con SDXL se realizó inicialmente con cinco clases (basal cell carcinoma, dermatofibroma, melanoma, pigmented benign keratoses y vascular), también se realizó un primer baseline con estas cinco clases para el modelo de clasificación.

El principal inconveniente encontrado en esta etapa recae en que dos de estas clases mencionadas tienen menos de 150 imágenes reales etiquetadas. Por esta razón, durante el proceso de ETL se agregó una restricción para realizar un *downsampling* de las imágenes de las clases mayoritarias, de forma de que queden medianamente balanceados los datos de entrenamiento. Podría haberse implementado alguna técnica complementaria como *class weights* para utilizar todas las imágenes, y ponderar en la función de pérdida cada clase

tomando en cuenta cuántas imágenes hay de cada clase, pero se obtuvieron resultados levemente menores con esta técnica en comparación con *downsampling*.

Por otro lado, también se implementó *data augmentation* para generar rotaciones, inversiones y otras alteraciones en las imágenes para evitar el sobreajuste.

A su vez, como se puede observar en la sección anterior de la arquitectura del modelo, se implementaron algunas capas de *dropout* y *batch normalization* para prevenir el sobreajuste y mejorar la generalización del modelo.

En un principio, se probaron arquitecturas como ResNet y EfficientNet como base para el problema de clasificación, pero ambas arquitecturas terminaban rápidamente en sobreajuste, logrando métricas más bajas. Por esta razón es que se decidió continuar con *transfer learning* pero utilizando un modelo pre-entrenado menos complejo, teniendo una menor cantidad de parámetros. Esto es lo que decantó en la utilización de MobileNet.

4.5.4 Experimentos

En cuanto a la experimentación con imágenes generadas, se decidió probar con dos alternativas: utilizar únicamente imágenes generadas para un primer *fine-tuning* de la red y luego hacer un segundo *fine-tuning* con una combinación de imágenes reales, o utilizar una combinación de imágenes generadas para hacer un único *fine-tuning*.

Al realizar el primer enfoque, se lograron métricas irrealistas entrenando y evaluando únicamente con imágenes generadas, llegando casi al 100% de accuracy. Este experimento sirvió para identificar que, si bien la calidad de las imágenes era muy buena en comparación con las imágenes reales, y hasta indistinguibles en ciertos casos, no presentaban la misma diversidad de lesiones dentro de cada clase. Esto se debe a que para entrenar con LoRA el modelo de SDXL, se decidió inicialmente tomar imágenes de lesiones similares para cada clase.

Figura 4.11 - Subconjunto de las imágenes reales seleccionadas para entrenar las capas de LoRA con SDXL para el caso de lesiones del tipo vascular.

Figura 4.12 - Imágenes generadas de muestra, resultado de entrenar las capas de LoRA con SDXL para el caso de lesiones del tipo vascular.

Como se puede observar a partir de las dos figuras anteriores, la calidad de las imágenes generadas es muy buena en relación a las imágenes reales. A su vez, se logra generar una diversidad de formas de lesiones y algunas leves variaciones de los colores de piel y texturas presentes en las imágenes generadas.

Por otro lado, luego se identificaron de mejor forma cuáles son otros patrones presentes en las imágenes de lesiones de tipo vascular, como se puede apreciar en la siguiente figura.

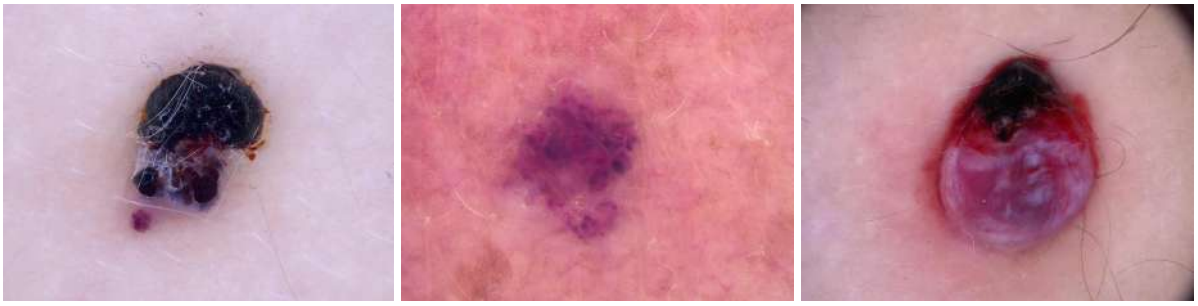


Figura 4.13 - Subconjunto de imágenes reales con patrones distintos a las seleccionadas para entrenar las capas de LoRA con SDXL para el caso de lesiones del tipo vascular.

Es por esto que, luego de haber detectado dicho problema, se procedió nuevamente con el fine-tuning de SDXL con LoRA pero utilizando una mayor diversidad de las imágenes utilizadas para entrenar, buscando lograr generar imágenes diversas que contemplen la mayor parte de los casos. A continuación se presentan casos de imágenes generadas considerando este nuevo enfoque.

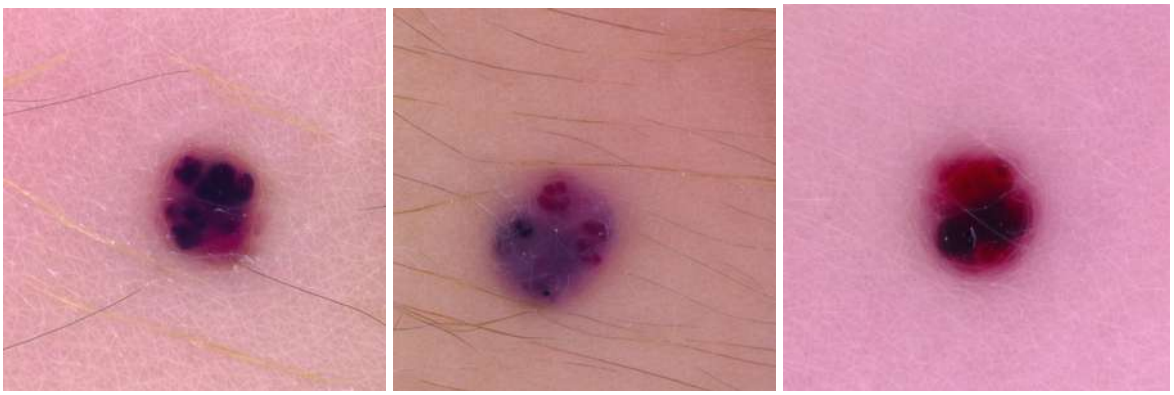


Figura 4.14 - Subconjunto de imágenes generadas con patrones distintos al entrenar las capas de LoRA con SDXL para el caso de lesiones del tipo vascular con imágenes más variadas.

De todas formas, gran parte de las imágenes generaban lesiones menos realistas en comparación con los resultados obtenidos anteriormente, lo cual se vio potenciado aún más en clases donde las lesiones pueden tomar aún mayor variedad de formas y colores, como es el caso del melanoma. Un aspecto a destacar en esta línea es que cuando los datos utilizados para entrenar un modelo eran muy diversos, el modelo era proclive a generar imágenes que tendían a la media de las imágenes de entrenamiento.

A modo de ejemplo, en el caso de lesiones vasculares, se tienen en su mayoría lesiones rojas y negras, con la presencia de alguna lesión violeta. Sin embargo, al entrenar un modelo utilizando estas lesiones, el modelo generaba principalmente lesiones rosadas y violetas que no se encontraban en los datos de entrenamiento. El equipo de trabajo concluyó que es posible que el modelo tienda a ir quitando ruido iterativamente convergiendo a la media de los tonos de colores y formas con los que fue entrenado.

Por esta razón, se decidió utilizar un tercer enfoque, en el que se identifican los patrones más comunes de cada clase de lesión y se generan pesos de LoRA específicos para cada uno de ellos, de manera de generar una mayor variedad de lesiones, luego mezclando imágenes generadas de manera balanceada para poder entrenar la red de clasificación. En el Anexo A se dejan imágenes representativas de cada patrón identificado de cada una de las cinco clases consideradas.

Por último, también se identificó la importancia de utilizar imágenes reales tanto para la validación como para test en todos los casos; dejando estos fijos de manera de poder tener experimentos comparables entre sí.

Desbalance de datos

Debido a que una de las principales motivaciones de este proyecto consiste en solventar el desbalance de datos y la poca representatividad de algunas clases en datasets de imagenología médica, se muestra la cantidad de datos presentes en el dataset de lesiones de piel que fue utilizado durante el proyecto.

El *dataset* consiste de siete clases, que se pueden observar en la siguiente figura:

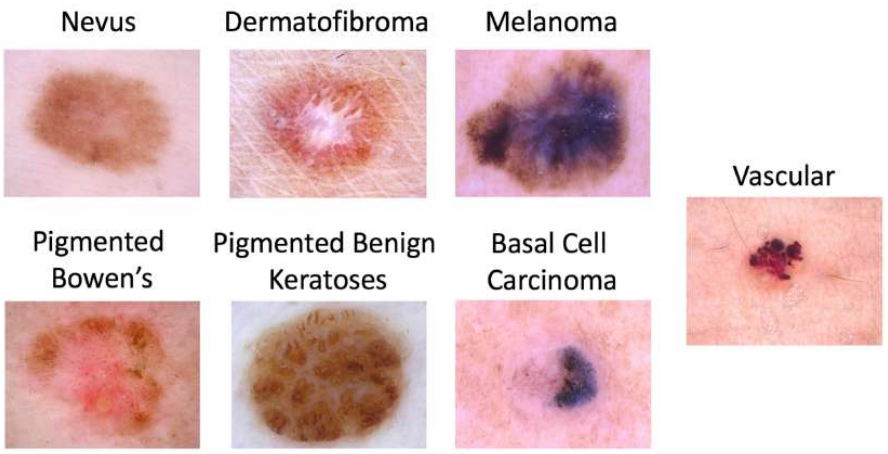


Figura 4.15 - Imágenes de referencia de las siete lesiones de piel disponibles en el dataset seleccionado. [17]

Debido a los extensos tiempos de entrenamiento pero principalmente de generación de las imágenes sintéticas, se decidió enfocar el estudio en las cinco clases que contenían una menor cantidad de datos. Estas son: dermatofibroma, melanoma, pigmented benign keratosis, basal cell carcinoma y vascular.

La cantidad de imágenes por clase dentro del dataset es la siguiente:

lesion_type	# images
basal_cell_carcinoma	514
dermatofibroma	115
melanoma	1113
pigmented_benign_keratosis	1099
vascular	142

Tabla 4.1 - Cantidad de imágenes por tipo de lesión presentes en el dataset

4.5.5 Resultados

Como fue mencionado anteriormente, se realizó un baseline solamente con imágenes reales para luego comparar la *performance* de la misma red al incluir imágenes generadas en el entrenamiento, como técnica de *data augmentation*.

Dado que este modelo fue la base para entrenar el resto de los modelos de los experimentos, se realizaron pruebas para identificar qué arquitectura brindaba mejores resultados. Esto incluyó: la definición sobre qué modelo se toma para transfer learning, las capas a agregar y las técnicas de regularización a utilizar. Luego, se efectuó una optimización de hiperparámetros, con foco en detectar qué transformaciones realizar de *data augmentation* como rotaciones y flips, definir la cantidad de épocas, el tamaño de *batch size*, la tasa de aprendizaje y el optimizador a utilizar.

A continuación se presentan las métricas de *categorical accuracy* y la función de pérdida sobre el dataset balanceado para el baseline:

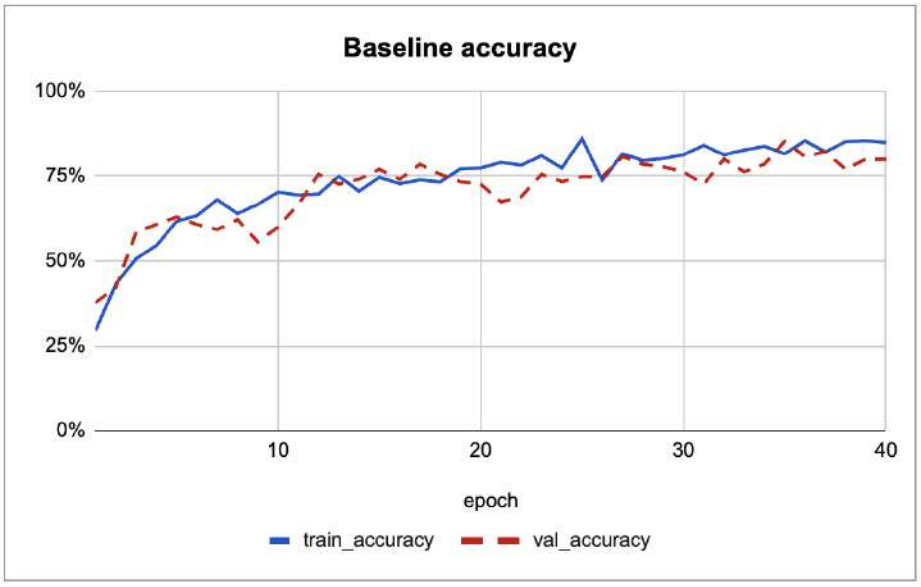


Figura 4.16 - Accuracy obtenido para el baseline compuesto únicamente por imágenes reales.

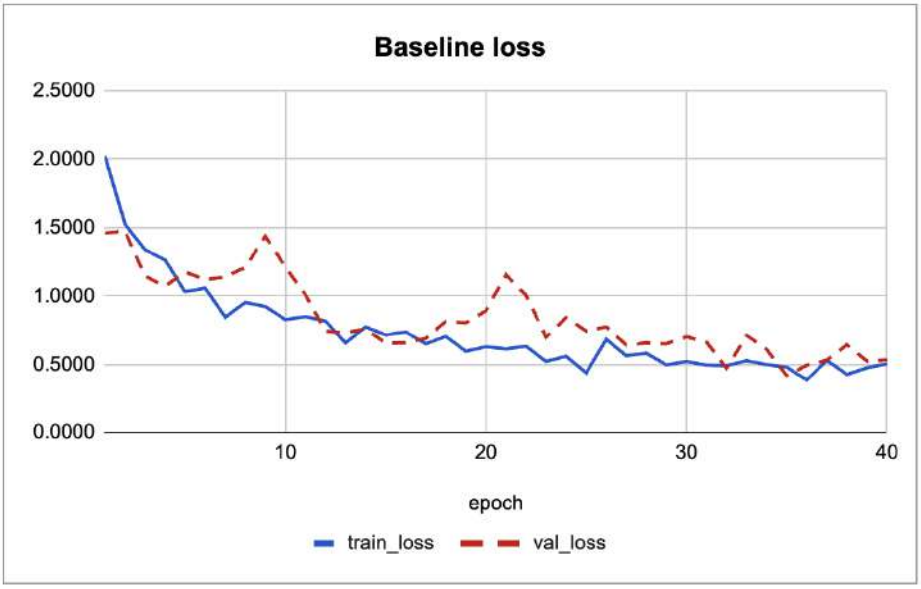


Figura 4.17 - Función de pérdida del baseline compuesto únicamente por imágenes reales.

Si bien puede parecer que se podría entrenar la red por un par de épocas más, terminaba sobreajustando a los datos rápidamente. La dificultad principal durante este proceso fue la prevención del sobreajuste de los datos, debido a que no se tienen muchos datos para entrenar y el modelo utilizado, por el hecho de utilizar *transfer learning*, tiene una cantidad de parámetros considerable.

Tomando este modelo de base, se mantuvieron los mismos sets de validación y test compuestos únicamente por imágenes reales para el resto de los experimentos, de manera de que fuera una comparación justa. En cuanto al set de entrenamiento, se mantienen los mismos datos y se completan con datos generados de manera de llegar a las proporciones deseadas. Resulta de interés conocer en esta investigación cómo afecta a la *performance* utilizar distintas cantidades de imágenes generadas dentro del set de entrenamiento. En

particular, se realizan experimentos probando 10% de imágenes generadas con 90% de imágenes originales, 20% de imágenes generadas con 80% de imágenes originales, y así sucesivamente hasta utilizar solamente imágenes generadas.

Estos experimentos se realizaron para tres distintos *splits* de datos aleatorios, de manera de obtener resultados que fueran estadísticamente más representativos. A continuación se presenta un gráfico que muestra la performance promedio en entrenamiento, validación y test para distintos porcentajes de datos generados.

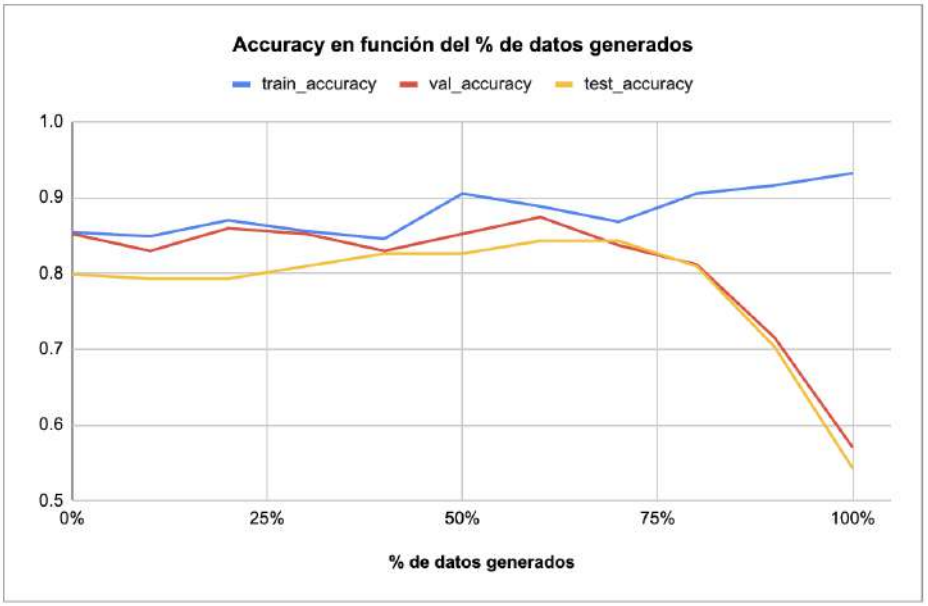


Figura 4.18 - Accuracy obtenido para los sets de entrenamiento, validación y test para distintas composiciones del set de entrenamiento según el porcentaje de imágenes generadas contenido en éste.

A continuación se muestra la misma figura centrando en la región de interés para poder visualizar mejor estas diferencias.

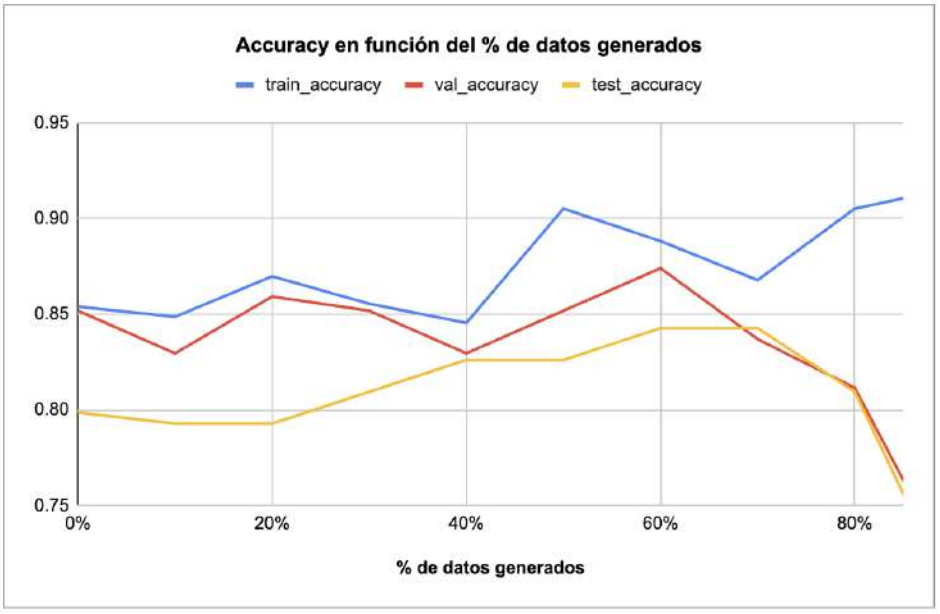


Figura 4.19 - Accuracy obtenido para los sets de entrenamiento, validación y test para distintas composiciones del set de entrenamiento según el porcentaje de imágenes generadas contenido en éste, centrado en la región de interés.

Como se puede observar en las dos figuras anteriores, el desempeño en los tres *sets* -pero en particular en el de test- mejora al comenzar a agregar imágenes generadas en el set de entrenamiento, aunque llega cierto punto en que resulta contraproducente seguir agregando imágenes generadas.

% de datos generados	test_accuracy	% vs baseline
0%	79.87%	0.0%
10%	79.29%	-0.7%
20%	79.29%	-0.7%
30%	80.95%	1.3%
40%	82.61%	3.4%
50%	82.61%	3.4%
60%	**84.28%**	**5.5%**
70%	**84.28%**	**5.5%**
80%	80.95%	1.3%
90%	70.28%	-12.0%
100%	54.28%	-32.0%

Tabla 4.2 - Accuracy de test obtenido para distintas distintas composiciones del set de entrenamiento según el porcentaje de imágenes generadas contenido en éste.

Se destaca que en las composiciones de entre 60% y 70% de imágenes generadas se obtuvo el mejor desempeño en el set de test. Con menos del 30% de imágenes generadas, no se tiene una diferencia notoria con respecto a la utilización únicamente de imágenes reales. Si bien baja levemente el accuracy, podría deberse a algo estocástico. Cuando ya se empieza a tener más del 80% de imágenes generadas en el set de entrenamiento, la performance cae rápidamente.

Aunque no se tiene una explicación fundada, se cree que la distribución de las imágenes de entrenamiento sigue sin ser totalmente representada por las imágenes generadas. A pesar de haber realizado esfuerzos por mejorar esta representación, siguen existiendo otras variaciones de las lesiones que no son tomadas en consideración en las imágenes generadas.

Por otro lado, se efectuó un experimento adicional: en vez de tomar proporciones de la cantidad de imágenes generadas en los datos de entrenamiento, se realizó un nuevo experimento que buscó determinar el impacto en la performance si solamente utilizáramos imágenes generadas como método de *upsampling* para las clases minoritarias en el dataset. Para esto, se mantuvieron los mismos sets de validación y test que en los experimentos anteriores, pero solamente se incorporaron imágenes generadas en las clases de lesiones vasculares y dermatofibromas, ya que las otras clases tienen alta representación en los datos. Se entrenó el mismo modelo con la misma cantidad de hiperparámetros y los resultados fueron los siguientes.

experimento	train_accuracy	val_accuracy	test_accuracy	% vs baseline
baseline	85.40%	85.19%	79.87%	0.0%
upsampling	**86.70%**	**91.11%**	**88.68%**	**+11.02%**

Tabla 4.3 - Accuracy de train, validation y test obtenidos para el baseline y el experimento de utilizar imágenes generadas como herramienta de upsampling de las clases minoritarias.

Como se puede observar en la tabla anterior, al usar imágenes generadas solamente en aquellas clases que se encontraban menos representadas, se logró mejorar en un 11% el accuracy obtenido con el modelo, utilizando la misma arquitectura e hiperparámetros. Aunque parecería que la red podría seguir entrenando debido a que las métricas de validación son superiores a las de entrenamiento, se utilizó el *checkpoint* del modelo que obtuvo mayor accuracy en validación que es el que figura en la tabla. Luego, las métricas de validación se mantenían o bajaban.

Por último, se presenta el *recall* por clase como una métrica adicional que es común en problemas de clasificación dentro del área de la imagenología médica. Esto se debe a que el recall se enfoca en la detección correcta de casos positivos, lo cual es crítico para evitar diagnósticos erróneos, en particular falsos negativos. A continuación se puede ver un gráfico que muestra el *recall* por clase obtenido para cada uno de los tres principales modelos: el *baseline* - modelo entrenado únicamente con imágenes reales -, proportions que hace referencia al modelo entrenado con 60% de imágenes generadas y 40% de imágenes originales por clase, y upsampling que representa este último experimento en donde se utilizaron las imágenes generadas como técnica de *upsampling* para las lesiones con menor cantidad de imágenes reales.

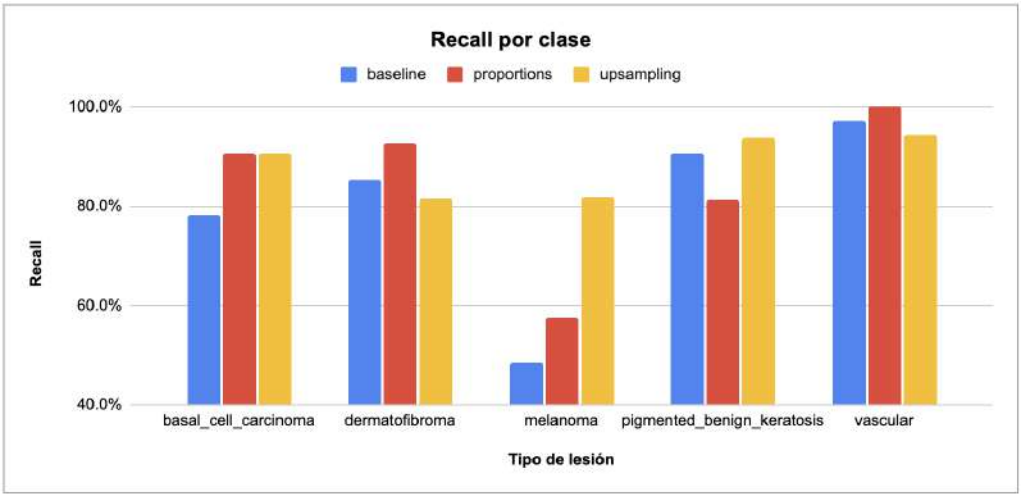


Figura 4.20 - Recall por clase obtenido para los tres experimentos realizados.

A su vez, se presenta la siguiente tabla que concentra los valores de recall obtenidos para dichos experimentos.

lesion_type	baseline	proportions	upsampling
basal_cell_carcinoma	78.1%	**90.6%**	**90.6%**
dermatofibroma	85.2%	**92.6%**	81.5%
melanoma	48.5%	57.6%	**81.8%**
pigmented_benign_keratosis	90.6%	81.3%	**93.8%**
vascular	97.1%	**100.0%**	94.3%

Tabla 4.4 - Métricas de recall obtenidas desglosadas por tipo de lesión para cada uno de los tres experimentos principales: baseline, experimento con proporciones de imágenes generadas y reales y la utilización de imágenes generadas como técnica de upsampling.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este informe, se exploraron las técnicas generativas en el ámbito de la inteligencia artificial, con un enfoque particular en los modelos de difusión aplicados a la imagenología médica, específicamente en la caracterización de lesiones dermatológicas. Se relevó el estado del arte actual, destacando las principales ventajas y desventajas de la utilización de cada una de estas técnicas.

Se implementaron diversas técnicas de *fine-tuning* en modelos de difusión pre-entrenados, como Stable Diffusion. Los resultados de esta implementación demostraron que la calidad de las imágenes generadas es alta, incluso al compararlas con imágenes reales dentro del ámbito de la imagenología médica. En particular, con Textual Inversion no se lograron los resultados esperados, mientras que con LoRA se tuvo una mejora considerable en la calidad de estos. El salto de performance que logró llegar a la calidad esperada se dio cuando se pasó de utilizar el modelo estándar de Stable Diffusion, por Stable Diffusion XL (SDXL), lanzado en junio del presente año.

Durante dicha implementación, se abordaron diversos desafíos que surgen al realizar una implementación propia y no utilizar alternativas de alto nivel, como la web-ui de Stable Diffusion. Debido a esto se pudo adquirir un mayor entendimiento sobre cómo funcionan las soluciones basadas en modelos de difusión, pudiendo ser de utilidad en casos donde se necesita cierta customización en las arquitecturas y componentes de las mismas.

Algunos de los desafíos principales durante la ejecución del proyecto recaen en la falta de documentación en cuanto a la implementación práctica de estas soluciones. Existen varios blogs y tutoriales teóricos sobre su funcionamiento, pero pocos de estos enfocados al aspecto práctico. Se tuvieron que utilizar técnicas de reducción de precisión de los pesos de los modelos, así como una implementación de menor precisión del optimizador para lograr que SDXL pueda ser entrenado en una única GPU de acceso gratuito, como lo es la T4 de Google Colab.

En cuanto a la generación de clases no representadas en el dataset, se concluye que al no poder separar las lesiones de piel de su contexto (piel) en donde se encuentran, resulta difícil lograr alterar dicho contexto sin perjudicar la calidad de la lesión generada.

Otro aspecto a destacar es que el modelo utilizado no produjo buenos resultados cuando se utilizaron imágenes diversas dentro de la misma clase para su entrenamiento. Al menos en este caso en particular, se concluye que los modelos de difusión al realizar *fine-tuning* con LoRA o Textual Inversion, tienden a producir colores y formas cercanos a la media de la distribución de los datos que se utilizaron para dicho *fine-tuning*, pudiendo resultar en imágenes que no son realistas.

Finalmente, se entrenó un modelo de clasificación de lesiones de piel utilizando imágenes reales de un conjunto de datos público, donde se buscó combatir la escasez de datos y desbalance de estos mediante la utilización de las imágenes generadas como técnica de *data augmentation*. Al utilizar las imágenes generadas únicamente en las clases donde había subrepresentación de los datos, se obtuvo una mejora significativa del 11% de *accuracy* en

test con respecto al *baseline*, alcanzando el 88.68% de accuracy. A su vez, se mejoró el *recall* promedio por clase de 79.9% a 88.4%. Por otro lado, al realizar el experimento de qué proporción de imágenes generadas brindaba mejores resultados para el entrenamiento, se encontró que entre el 60% y 70% el accuracy en test con respecto al baseline mejoró un 5.5%, alcanzando un 84.28%. En cuanto al *recall* promedio por clase, también se mejoró con respecto al *baseline*, pasando de un 79.9% a 84.4%.

Estos experimentos resaltan la utilidad de las técnicas de generación, y en particular los modelos de difusión, para la creación de datos sintéticos que permitan potenciar soluciones de aprendizaje profundo cuando existe escasez o desequilibrio en los datos disponibles.

6. REFERENCIAS

[1] K. J. Geras y Linda Moy (NYU Langone), "Artificial Intelligence Tool Improves Accuracy of Breast Cancer Imaging",, 24 de septiembre de 2021. [En línea]. Disponible en: https://nyulangone.org/news/artificial-intelligence-tool-improves-accuracy-breast-cancer-imaging/. [Último acceso: 14 de mayo de 203].

[2] N. Jayakumar, T. Hossain y M. Zhang, "SADIR: Shape-Aware Diffusion Models for 3D Image Reconstruction", ArXiv, 6 de septiembre de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2309.03335. [Último acceso: 6 de setiembre de 2023].

[3] Z. Dorjsembe et al., "Conditional Diffusion Models for Semantic 3D Medical Image Synthesis", IOP Science, 29 de mayo de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2305.18453v3. [Último acceso: 28 de julio de 2023].

[4] J. Wang et al., "InverseSR: 3D Brain MRI Super-Resolution Using a Latent Diffusion Model", ArXiv, 23 de agosto de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2308.12465. [Último acceso: 14 de mayo de 2023].

[5] J. Wu et al., "MedSegDiff: Medical Image Segmentation with Diffusion Probabilistic Model", Open Review, 4 de abril de 2023. [En línea]. Disponible en: https://openreview.net/forum?id=Jdw-cm2jG9. [Último acceso: 14 de mayo de 2023].

[6] Y. Fu et al., "A Recycling Training Strategy for Medical Image Segmentation with Diffusion Denoising Models", ArXiv, 30 de agosto de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2308.16355. [Último acceso: 6 de setiembre de 2023].

[7] L. W. Sagers et al., "Augmenting medical image classifiers with synthetic data from latent diffusion models", ArXiv, 23 de agosto de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2308.12453. [Último acceso: 6 de setiembre de 2023].

[8] F. Bieder et al., "Memory-Efficient 3D Denoising Diffusion Models for Medical Image Processing", Open Review, 4 de abril de 2023. [En línea]. Disponible en: https://openreview.net/forum?id=neXqIGpO-tn. [Último acceso: 14 de mayo de 2023].

[9] F. Khader et al., "Denoising diffusion probabilistic models for 3D medical image generation", Nature, 5 de mayo de 2023. [En línea]. Disponible en: https://www.nature.com/articles/s41598-023-34341-2. [Último acceso: 14 de mayo de 2023].

[10] S. Pan et al., "2D medical image synthesis using transformer-based denoising diffusion probabilistic model", IOP Science, 5 de mayo de 2023. [En línea]. Disponible en: https://iopscience.iop.org/article/10.1088/1361-6560/acca5c. [Último acceso: 14 de mayo de 2023].

[11] OpenAI, "Introducing ChatGPT", 30 de noviembre de 2022. [En línea]. Disponible en: https://openai.com/blog/chatgpt. [Último acceso: 5 de mayo de 2023].

[12] Meta, "Introducing AudioCraft: A Generative AI Tool For Audio and Music", 2 de agosto de 2023. [En línea]. Disponible en: https://about.fb.com/news/2023/08/audiocraft-generative-ai-for-music-and-audio/. [Último acceso: 6 de setiembre de 2023].

[13] Midjourney, "Community Showcase", 12 de julio de 2022. [En línea]. Disponible en: https://www.midjourney.com/showcase/recent/. [Último acceso: 12 de mayo de 2023].

[14] M. Jadhav, "Auto Encoders: De-Noising Text Documents", Medium, 31 de enero de 2020. [En línea]. Disponible en: https://medium.com/analytics-vidhya/auto-encoders-de-noising-text-documents-c58d6950dfad. [Último acceso: 31 de abril de 2023].

[15] D. P. Kingma y M. Welling, "Auto-Encoding Variational Bayes", ArXiv, 20 de diciembre de 2013. [En línea]. Disponible en: https://arxiv.org/abs/1312.6114. [Último acceso: 31 de abril de 2023].

[16] J. Rocca, "Understanding Variational Autoencoders (VAEs)", Towards Data Science, 23 de septiembre de 2019. [En línea]. Disponible en: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. [Último acceso: 31 de abril de 2023].

[17] I. J. Goodfellow et al., "Generative Adversarial Networks", ArXiv, 10 de junio de 2014. [En línea]. Disponible en: https://arxiv.org/abs/1406.2661. [Último acceso: 31 de abril de 2023].

[18] T. Karras, S. Laine y T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks", ArXiv, 12 de diciembre de 2018. [En línea]. Disponible en: https://arxiv.org/abs/1812.04948v3. [Último acceso: 3 de junio de 2023].

[19] P. Isola et al., "Image-to-Image Translation with Conditional Adversarial Networks", ArXiv, 21 de noviembre de 2016. [En línea]. Disponible en: https://arxiv.org/abs/1611.07004. [Último acceso: 14 de mayo de 2023].

[20] Zhu et al., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", Arxiv, 30 de Marzo de 2017. [En línea]. Disponible en: https://arxiv.org/abs/1703.10593. [Último acceso: 14 de mayo de 2023].

[21] A. Gainetdinov, "Diffusion Models vs. GANs vs. VAEs: Comparison of Deep Generative Models", Towards AI, 11 de mayo de 2023. [En línea]. Disponible en: https://pub.towardsai.net/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models-67ab93e0d9ae. [Último acceso: 27 de junio de 2023].

[22] J. Ho, A. Jain y P. Abbeel, "Denoising Diffusion Probabilistic Models", ArXiv, 16 de diciembre de 2020. [En línea]. Disponible en: https://arxiv.org/abs/2006.11239. [Último acceso: 31 de abril de 2023].

[23] A. Nichol y P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models", ArXiv, 18 de febrero de 2021. [En línea]. Disponible en: https://arxiv.org/abs/2102.09672. [Último acceso: 4 de abril de 2023].

[24] P. Radiuk, "Applying 3D U-Net Architecture to the Task of Multi-Organ Segmentation in Computed Tomography", Sciendo, mayo de 2020. [En línea]. Disponible en: https://sciendo.com/article/10.2478/acss-2020-0005. [Último acceso: 14 de mayo de 2023].

[25] A. Vaswani et al., "Attention Is All You Need", Arxiv, 12 de junio de 2017. [En línea]. Disponible en: https://arxiv.org/abs/1706.03762. [Último acceso: 31 de abril de 2023].

[26] V. Singh, "An In-Depth Guide to Denoising Diffusion Probabilistic Models – From Theory to Implementation", Learn Open CV, 6 de marzo de 2023. [En línea]. Disponible en: https://learnopencv.com/denoising-diffusion-probabilistic-models/. [Último acceso: 11 de julio de 2023].

[27] R. O'Connor (Assembly AI), "Introduction to Diffusion Models for Machine Learning,", 12 de mayo de 2022. [En línea]. Disponible en: https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction. [Último acceso: 19 de abril de 2023.

[28] R. Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models", ArXiv, 13 de abril de 2022. [En línea]. Disponible en: https://arxiv.org/abs/2112.10752. [Último acceso: 17 de abri de 2023].

[29] Steins, "Stable Diffusion Clearly Explained!", Medium, 2 de enero de 2023. [En línea]. Disponible en: https://medium.com/@steinsfu/stable-diffusion-clearly-explained-ed008044e07e. [Último acceso: 6 de setiembre de 2023].

[30] J. Ho y T. Salimans, "Classifier-Free Diffusion Guidance", ArXiv, 26 de julio de 2022. [En línea]. Disponible en: https://arxiv.org/abs/2207.12598. [Último acceso: 31 de abril de 2023].

[31] D. Podell et al., "SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis", ArXiv, 14 de julio de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2307.01952. [Último acceso: 28 de agosto de 2023].

[32] N. Ruiz et al., "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation," ArXiv, 25 de agosto de 2022. [En línea]. Disponible en: https://arxiv.org/abs/2208.12242. [Último acceso: 17 de junio de 2023].

[33] R. Gal et al., "An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion", ArXiv, 2 de agosto de 2022. [En línea]. Disponible en: https://arxiv.org/abs/2208.01618. [Último acceso: 31 de abril de 2023].

[34] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models", ArXiv, 16 de octubre de 2021. [En línea]. Disponible en: https://arxiv.org/abs/2106.09685. [Último acceso: 31 de abril de 2021].

[35] S. Raschka (Lightning AI), "Parameter-Efficient LLM Finetuning With Low-Rank Adaptation (LoRA)", 26 de abril de 2023. [En línea]. Disponible en: https://lightning.ai/pages/community/article/lora-llm/. [Último acceso: 31 de abril de 2023].

[36] Chris (iWay), "What are LoRA model in Stable Diffusion?", 29 de marzo de 2023. [En línea]. Disponible en: https://iway.org/what-are-lora-model-in-stable-diffusion/. [Último acceso: 7 de julio de 2023].

[37] L. Zhang, A. Rao y M. Agrawala, "Adding Conditional Control to Text-to-Image Diffusion Models", ArXiv, 10 de febrero de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2302.05543. [Último acceso: 27 de junio de 2023].

[38] UC Irvine, "UC Irvine Machine Learning Repository", 19 de septiembre de 2022. [En línea]. Disponible en: https://archive.ics.uci.edu/. [Último acceso: 31 de marzo de 2023].

[39] National Cancer Institute, "The Cancer Imaging Archive (TCIA)", 4 de octubre de 2020. [En línea]. Disponible en: https://imaging.cancer.gov/informatics/cancer_imaging_archive.htm. [Último acceso: 31 de abril de 2023].

[40] M. Antonelli et al., "Medical Segmentation Decathlon", 15 de julio de 2022. [En línea]. Disponible en: http://medicaldecathlon.com/. [Último acceso: 31 de abril de 2023].

[41] Image CLEF, "The CLEF Cross Language Image Retrieval Track", 5 de enero de 2002. [En línea]. Disponible en: https://www.imageclef.org/. [Último acceso: 26 de marzo de 2023].

[42] International Skin Imaging Collaboration (ISIC), "ISIC 2018 Challenge - Task 3: Lesion Diagnosis", 17 de junio de 2018. [En línea]. Disponible en: https://challenge.isic-archive.com/landing/2018/47/. [Último acceso: 18 de agosto de 2023].

[43] Hugging Face, "Diffusers", 21 de julio de 2022. [En línea]. Disponible en: https://github.com/huggingface/diffusers. [Último acceso: 6 de setiembre de 2023].

[44] Automatic1111, "stable-diffusion-webui", 1 de mayo de 2023. [En línea]. Disponible en: https://github.com/AUTOMATIC1111/stable-diffusion-webui. [Último acceso: 13 de mayo de 2023].

[45] CompVis, "Stable Diffusion v1-4 Model Card", junio de 2022. [En línea]. Disponible en: https://huggingface.co/CompVis/stable-diffusion-v1-4. [Último acceso: 15 de agosto de 2023].

[46] Stability AI, "SD-XL 1.0-base Model Card", 26 de julio de 2023. [En línea]. Disponible en: https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0. [Último acceso: 6 de setiembre de 2023].

[47] Madebyollin, "SDXL-VAE-FP16-Fix Model Card", 11 de julio de 2023. [En línea]. Disponible en: https://huggingface.co/madebyollin/sdxl-vae-fp16-fix. [Último acceso: 19 de agosto de 2023].

[48] A. Kazerouni et al., "Diffusion models in medical imaging: A comprehensive survey", Medical Image Analysis, agosto de 2023. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S1361841523001068. [Último acceso: 6 de setiembre de 2023].

[49] C. Dilmegani (AIMultiple), "Generative AI Healthcare Industry: Benefits, Challenges, Potentials", 10 de agosto de 2023. [En línea]. Disponible en: https://research.aimultiple.com/generative-ai-healthcare/. [Último acceso: 6 de setiembre de 2023].

[50] C. Zhang et al., "A Survey on Audio Diffusion Models: Text To Speech Synthesis and Enhancement in Generative AI", ArXiv, 23 de marzo de 2023. [En línea]. Disponible en: https://arxiv.org/abs/2303.13336. [Último acceso: 5 de agosto de 2023].

[51] Keras, "Keras Applications", 12 de abril de 2023. [En línea]. Disponible en: https://keras.io/api/applications/. [Último acceso: 11 de agosto de 2023].

[52] M. Isaksson, "Exploring Generative Adversarial Networks (GANs)", Towards Data Science, 24 de junio de 2020. [En línea]. Disponible en: https://towardsdatascience.com/exploring-generative-adversarial-networks-gans-488e1d901d4a. [Último acceso: 20 de abril de 2023].

[53] O. Ronneberger, P. Fischer y T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", ArXiv, 18 de mayo de 2015. [En línea]. Disponible en: https://arxiv.org/abs/1505.04597. [Último acceso: 2 de julio de 2023].

7. REPOSITORIOS DE TRABAJO

En el siguiente enlace se encuentra el repositorio de trabajo con la implementación de las distintas técnicas utilizadas para fine tuning de los modelos de difusión: repositorio. En caso de no tener acceso, solicitar el mismo a los siguientes correos: `Ignacio Aristimuño` y `Daniel.Gallino@gmail.com`.

Por otro lado, también se encuentra el código del ETL para creación de los splits de datos reales, el código correspondiente al entrenamiento del modelo de clasificación utilizado, y se deja habilitada una demo en Gradio para poder probar la generación de imágenes seleccionando la clase deseada.

Por otro lado, los pesos de los modelos utilizados se encuentran en la siguiente carpeta de Google Drive: pesos de modelos entrenados.

Por último, los datos de las lesiones de piel fueron descargados del sitio web oficial de la competencia de ISIC: datos.

8. ANEXOS

8.1 Anexo A: Selección de grupos de imágenes

Al observar que las imágenes similares tomadas para cada tipo de lesión representaban únicamente un subconjunto de las imágenes reales presentes en el dataset, se decidió seleccionar los grupos de imágenes más identificativos dentro de cada tipo de lesión y se entrenaron modelos individuales para cada uno de ellos. Con estos modelos, luego se generaron imágenes que fueron utilizadas para realizar los experimentos, mezclando imágenes de cada grupo de imágenes del tipo de lesión.

A continuación se presentan imágenes representativas de los grupos de imágenes seleccionados.

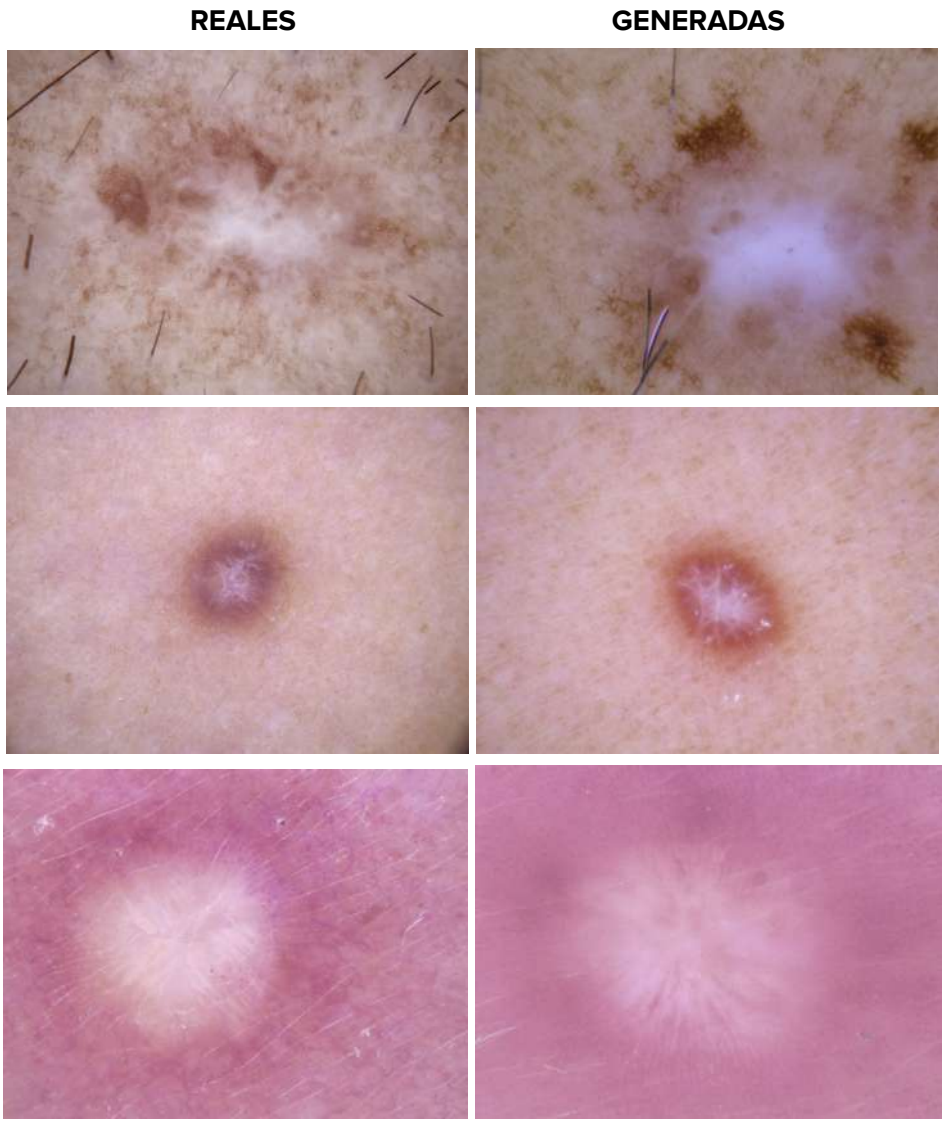


Figura 8.1 - Comparativa de imágenes reales (izquierda) e imágenes generadas (derecha) para cada grupo de imágenes identificado en dermatofibroma.

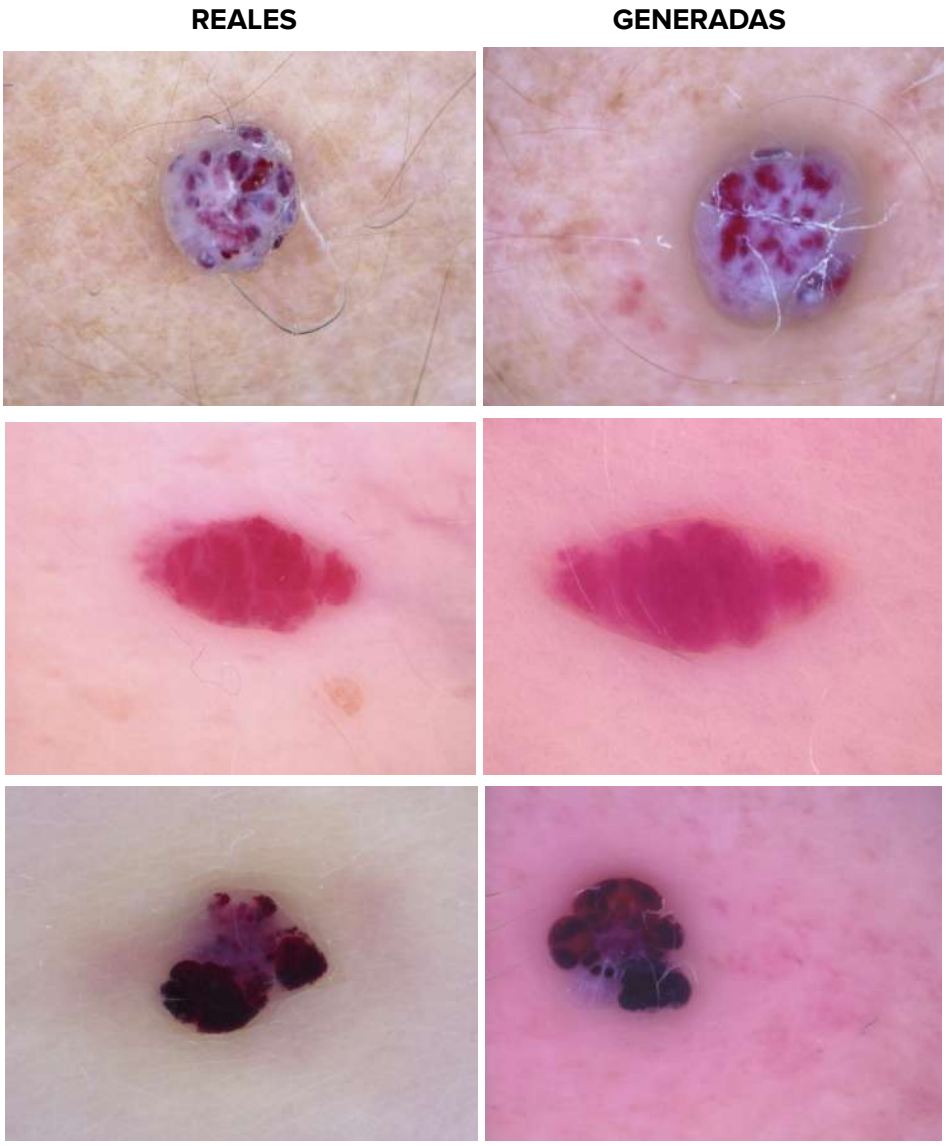


Figura 8.2 - Comparativa de imágenes reales (izquierda) e imágenes generadas (derecha) para cada grupo de imágenes identificado en vascular.

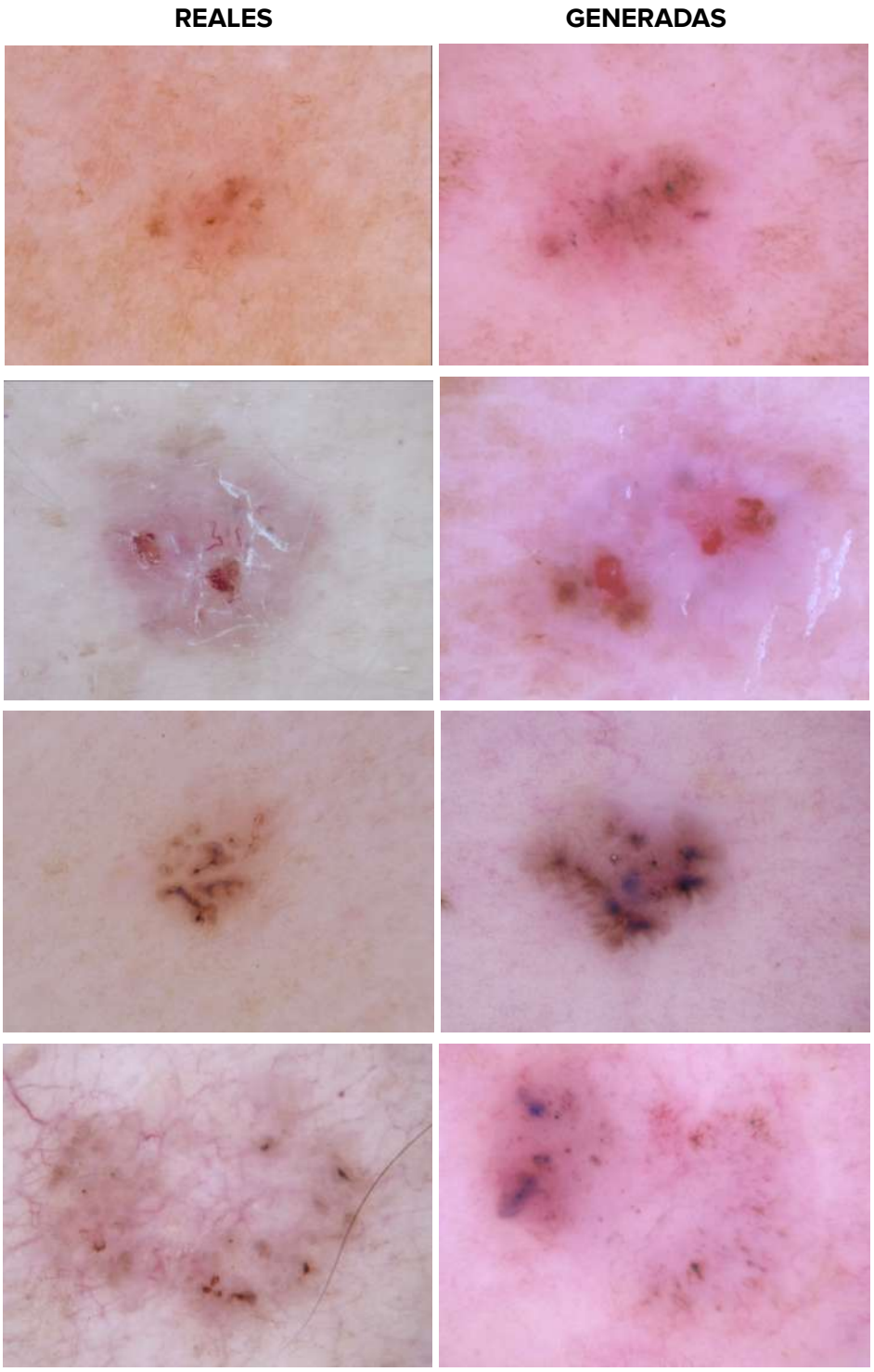


Figura 8.3 - Comparativa de imágenes reales (izquierda) e imágenes generadas (derecha) para cada grupo de imágenes identificado en basal cell carcinoma.

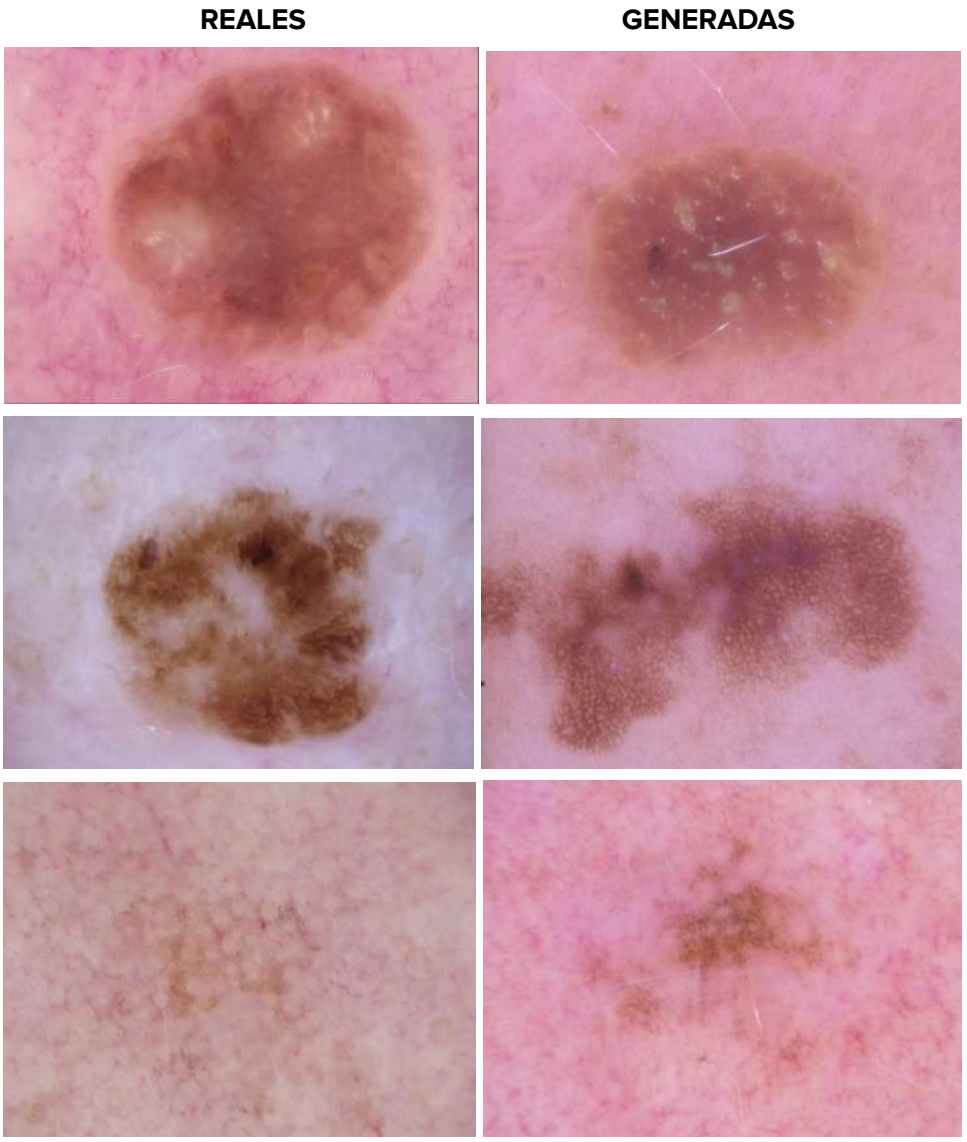


Figura 8.4 - Comparativa de imágenes reales (izquierda) e imágenes generadas (derecha) para cada grupo de imágenes identificado en pigmented benign keratoses.

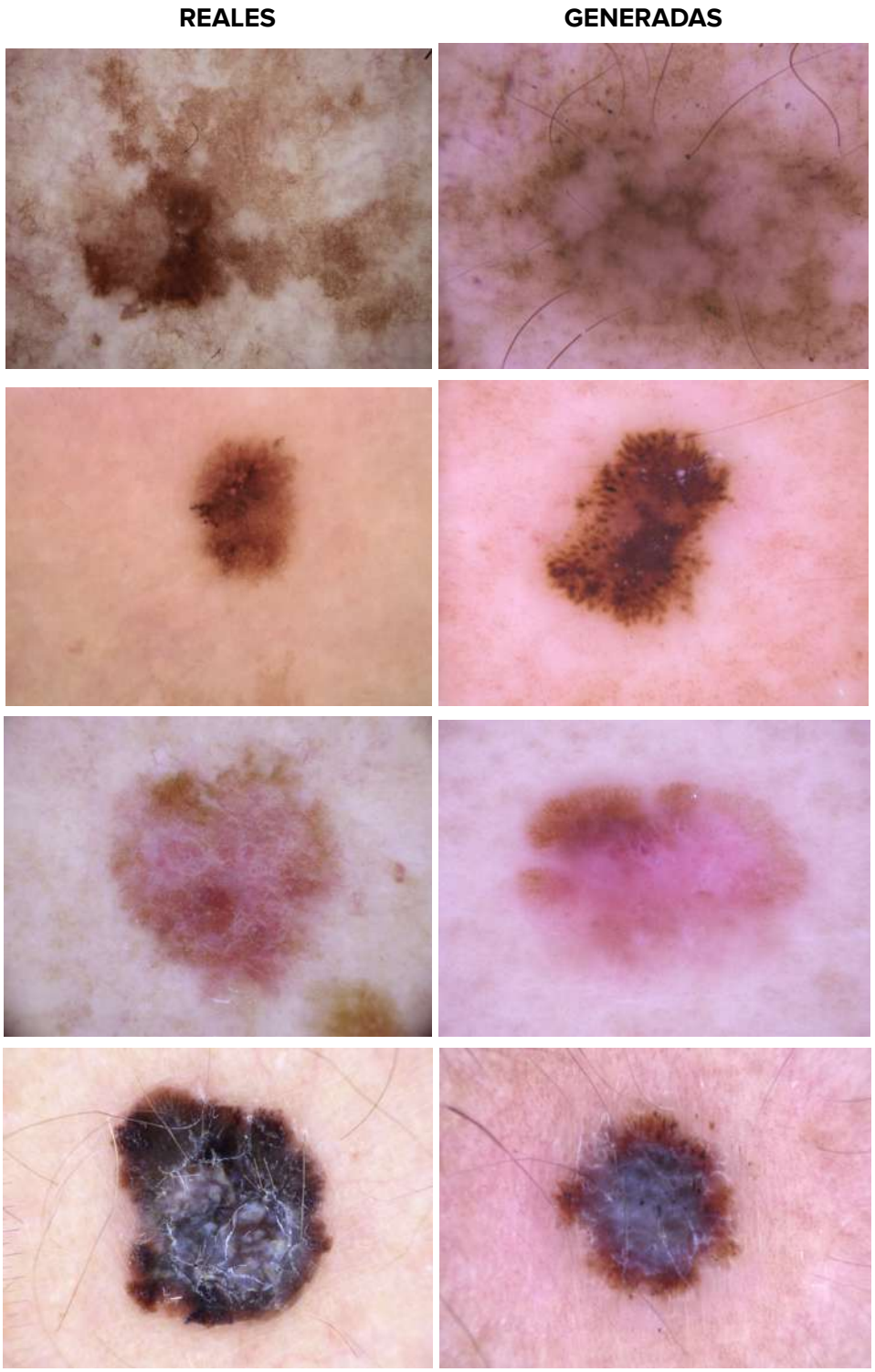


Figura 8.5 - Comparativa de imágenes reales (izquierda) e imágenes generadas (derecha) para cada grupo de imágenes identificado en melanoma.