
Generación de imágenes con Inteligencia Artificial para la construcción de libros de vida para pacientes con Alzheimer



**Trabajo de Fin de Grado
Curso 2023–2024**

Autor
Sergio Llorente Hernando
Isabella Romano Ramos

Director
Gonzalo Mendez Pozo
Pablo Gérvias Gómez-Navarro

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Generación de imágenes con Inteligencia Artificial para la construcción de libros de vida para pacientes con Alzheimer

Trabajo de Fin de Grado en **Ingeniería Informática**

Autor

Sergio Llorente Hernando
Isabella Romano Ramos

Director

Gonzalo Mendez Pozo
Pablo Gérvias Gómez-Navarro

Convocatoria: *Junio 2024*

Grado en **Ingeniería Informática**

Facultad de Informática
Universidad Complutense de Madrid

14 de mayo de 2024

Dedicatoria

*A todas aquellas personas que han visto como
sus recuerdos o los de un ser querido se
desvanecían.*

Agradecimientos

A todas las personas que han apoyado este proyecto y durante nuestra etapa universitaria.

Resumen

El mundo tal y como lo conocemos está avanzando cada vez más y en todos los aspectos que conocemos, entre estos avances podemos destacar la esperanza de vida y el uso de las nuevas tecnologías. Aunque en principio no veamos mucha relación entre ambos conceptos, están mucho más unidos de lo que pensamos. El aumento de la esperanza de vida se debe a las mejoras de los procesos y condiciones médicas y sanitarias que han sido de revolución en los últimos 100 años, gracias a ello podemos vivir mucho más tiempo y disfrutar de la vida unos años más. Sin embargo, hay consecuencias que sufren las personas que alcanzan los años más avanzados de edad, entre tantas, las más importantes son el deterioro de algunas capacidades sensoriales, como lo pueden ser la vista o el oído; y capacidades cognitivas como la orientación o la memoria. En esta última es en la que se basa la principal enfermedad que afecta a las personas de mayor edad, el Alzheimer, que ataca directamente a la memoria destruyendo lentamente su capacidad hasta llegar a las funciones más básicas. Aquí es cuando entran las últimas innovaciones tecnológicas realizadas, en dónde introducimos el concepto de Inteligencia Artificial para la realización de este trabajo. En concreto, la generación de imágenes a través de esta. Nuestro objetivo es crear recuerdos representados por imágenes generadas automáticamente y al momento, a través de palabras del propio afectado por el Alzheimer, para poder construir un libro de vida, que le ayudará a través de terapias de reminiscencia que favorecen que la enfermedad avance paulatinamente.

Palabras clave

Inteligencia artificial, generación de imágenes, prompt, Alzheimer, Stable Diffusion, redes neuronales.

Abstract

The world as we know it is advancing more and more and in all the aspects that we know, among these advances we can highlight life expectancy and the use of new technologies. Although at first sight we do not see much relationship between both concepts, they are much closer than we think. The increase in life expectancy is due to improvements in medical and health processes and conditions that have been revolutionary in the last 100 years. Thanks to this, we can live much longer and enjoy life for a few more years. However, there are consequences suffered by people who reach the most advanced years of age, among many, the most important are the deterioration of some sensory abilities, such as sight or hearing; and cognitive abilities such as orientation or memory. The latter is the basis of the main disease that affects older people, Alzheimer's, which directly attacks memory, slowly destroying its capacity until it reaches the most basic functions. This is when the latest technological innovations come in, where we introduce the concept of Artificial Intelligence. Specifically, the generation of images through it. Our objective is to create memories represented by images generated automatically and at the moment, through the words of the person affected by Alzheimer's, in order to build a life book, which will help them through reminiscence therapies that encourage the disease to progress gradually.

Keywords

Artificial intelligence, image generation, prompt, Alzheimer's, reminiscence therapies.

Índice

1. Introducción	1
1.1. Motivos	2
1.2. Objetivos	2
1.3. Plan de trabajo	3
2. Estado de la Cuestión	5
2.1. ¿Qué es el Alzheimer?	5
2.2. La terapia de reminiscencia	6
2.3. ¿Qué es la Inteligencia Artificial?	8
2.4. Redes neuronales	9
2.4.1. Redes Neuronales Feedforward (FNN)	11
2.4.2. Redes Neuronales Recurrentes (RNN)	12
2.4.3. Redes Neuronales LSTM	13
2.4.4. Redes Neuronales Convolutivas (CNN)	15
2.4.5. Redes Generativas Adversarias (GAN)	17
2.4.6. Redes Neuronales Transformer	18
2.5. Stable Diffusion	21
2.5.1. Funcionamiento interno de Stable Diffusion	21
3. Generación de imágenes con Inteligencia Artificial	27
3.1. Elección del modelo	27
3.2. Entrenamiento con Stable Diffusion	28

3.2.1.	Versiones y métodos de entrenamiento	28
3.2.2.	Requisitos en las imágenes de entrenamiento	30
3.2.3.	Procedimiento de entrenamiento	30
3.2.4.	Entrenamiento con la técnica de LORA	33
3.3.	Interfaz de Stable Diffusion	36
3.3.1.	Requisitos de instalación	36
3.3.2.	Funcionamiento y detalles de la interfaz	37
3.4.	Resultados de generación de imágenes	38
3.4.1.	Resultados con personas	38
3.4.2.	Resultados con lugares	39
3.4.3.	Resultados con animales	40
3.4.4.	Resultados con múltiples capas de entrenamiento	41
3.4.5.	Logros y fallos en resultados	43
3.4.6.	Soluciones: Técnica Inpainting	45
3.5.	Evolución de los modelos del proyecto	48
4. Aplicación		51
4.1.	Back-end	51
4.2.	Ejecución de la aplicación	54
5. Libro de vida		57
5.1.	Fase de la vida 1: Juventud	58
5.2.	Fase de la vida 2: Adulterz	59
5.3.	Fase de la vida 3: Vejez	61
6. Conclusiones y Trabajo Futuro		65
6.1.	Conclusiones	65
6.2.	Propuestas de mejora	66
Introduction		69
Conclusions and Future Work		71

Contribuciones Personales **73**

Bibliografia **77**

Índice de figuras

2.1.	Obstrucción de neuronas por la vitamina tau	6
2.2.	Terapia de reminiscencia	7
2.3.	Estructura del perceptrón	10
2.4.	Red neuronal feedforward	11
2.5.	Red neuronal recurrente	13
2.6.	Red neuronal Long Short Term Memory	15
2.7.	Red neuronal convolucional	17
2.8.	Red neuronal generativa antagónica	18
2.9.	Red neuronal Transformer	20
2.10.	difusión directa de una imagen de un cachorro de Shiba Inu comiendo	22
2.11.	difusión inversa de una imagen de un cachorro de Shiba Inu comiendo	22
2.12.	Representación gráfica del proceso de conversión al espacio latente .	23
2.13.	Arquitectura de una red U-Net	24
2.14.	Representación gráfica del funcionamiento de Stable Diffusion . . .	25
3.1.	Dataset seleccionado para el entrenamiento de personas con Lora . .	31
3.2.	Procedimiento del entrenamiento mediante Dreambooth	32
3.3.	Dataset seleccionado para el entrenamiento con lugares	33
3.4.	Dataset seleccionado para el entrenamiento con animales	34
3.5.	Parámetros relevantes del entrenamiento con LoRA	35
3.6.	Aplicación SDGUI, utilizada para probar todos los modelos generadores de imágenes	37
3.7.	Resultados de entrenamiento de una persona con Dreambooth . . .	39

3.8.	Resultados de entrenamiento de una persona con diferentes estilos . . .	39
3.9.	Resultados de imágenes del lugar entrenado al estilo Van Gogh y viñeta	40
3.10.	Resultados de imágenes del lugar en invierno y con una persona . . .	40
3.11.	Imagen generada con el modelo Stable Diffusion 1.5, Lora Hachiko . .	41
3.12.	Mismo prompt con 1000, 2000 y 10.000 pasos de entrenamiento . . .	41
3.13.	Dataset del entrenamiento con el salón de Friends (frndslivro) . . .	42
3.14.	Resultados de la generación de imágenes con el modelo frndslivro . .	42
3.15.	Mismo prompt con 20, 40 y 50 steps al mezclar dos elementos: persona y lugar	43
3.16.	Duplicidad de elementos	44
3.17.	Configuración de los parámetros en la técnica de inpainting . . .	46
3.18.	Ventana de la WEBUI implementada para realizar el painting . . .	47
3.19.	Evolución de una imagen con la técnica de inpainting aplicada . .	48
4.1.	Prototipo de la aplicación en Python	52
4.2.	Funcionamiento de modelos de Stable Diffusion	53
4.3.	Aplicación generadora de imágenes	54
4.4.	Ejecución de la app	55
4.5.	Portada de la aplicación	56
5.1.	Resultados obtenidos bajo un mismo prompt en las 3 fases . . .	58
5.2.	Dataset seleccionado para el entrenamiento con Juan de joven . . .	59
5.3.	Dataset seleccionado para el entrenamiento con Juan de adulto . . .	61
5.4.	Dataset seleccionado para el entrenamiento con Juan de anciano . .	63

Índice de tablas

- | | | |
|------|---|----|
| 5.1. | Tabla de resultados obtenidos del Capítulo 1 de juventud del paciente | 60 |
| 5.2. | Tabla de resultados obtenidos del Capítulo 2 de adultez del paciente | 62 |
| 5.3. | Tabla de resultados obtenidos del Capítulo 3 de vejez del paciente | 64 |

Capítulo 1

Introducción

“Todo se hunde en la niebla del olvido pero cuando la niebla se despeja, el olvido está lleno de memoria”

— Mario Benedetti

Nuestra vida se compone desde los primeros instantes en los tomamos nuestro primer aliento hasta que respiramos el último. Sin embargo, a la hora de relatar nuestra vida, esta historia estaría compuesta solamente de recuerdos. Desde nuestro primer recuerdo, que normalmente está comprendido entre los 2 y 4 años, debido a que la formación de nuevas neuronas impide que la corteza almacene recuerdos hasta esa edad, hasta el último, el cual puede darse el caso de ser distorsionado o borrado completamente debido al deterioro de nuestro cerebro y la imposibilidad de nuestras neuronas a realizar las conexiones posibles para ello. Este último caso es una enfermedad degenerativa comúnmente conocida como Alzheimer, que actualmente afecta a más de 55 millones de personas en todo el mundo. Sin embargo, no son sólo estas personas quiénes la sufren, sino todos sus allegados también.

Para ayudar a combatir esta enfermedad existen métodos farmacológicos y no farmacológicos, en este proyecto nos centraremos en el segundo grupo, más concretamente, en la terapia de reminiscencia. En cuanto a los métodos que existen actualmente para realizar esta terapia se ha comprobado que la ayuda de apoyo visual es mucho más efectiva a entrevistas con muchas preguntas seguidas, ya que esta última puede provocar que el paciente se vea agobiado y abrumado. Es por este motivo, entre muchos otros, que nuestro proyecto está orientado a ser una herramienta de apoyo visual a la narración de libros de vida. Para ello haremos uso de Inteligencias artificiales que convierten entradas de texto a imágenes y de esta manera, podremos transformar recuerdos narrados por el paciente y convertirlos en imágenes. Lo interesante es poder dar la posibilidad de hacer estas imágenes más personales de forma que se puedan añadir una serie de imágenes propias, de familiares o de eventos importantes de la vida del paciente, para entrenar al modelo y que el resultado final sea único, especial y de gran ayuda para la persona que sufre de esta enfermedad.

1.1. Motivos

El trabajo nace con base al proyecto CANTOR (Composición Automática de Narrativas personales como apoyo a Terapia Ocupacional basada en Reminiscencia, Plan Nacional de Investigación), una iniciativa planteada por investigadores de un conjunto de universidades españolas, entre ellas la Universidad Complutense de Madrid.

La principal motivación de la creación de este proyecto es brindar la posibilidad a los pacientes de demencia causada por el Alzheimer a construir un libro compuesto por sus recuerdos más felices, brindándole la oportunidad de rememorarlos. Se ha demostrado que cuando los pacientes hablan sobre determinadas épocas de su vida y momentos vividos en el pasado, se genera un impacto positivo en su persona consiguiendo que aumente su confianza y sentido de la vida.

El proyecto CANTOR está pensado para elaborarse en dos etapas: La primera sería la parte técnica en la que se elabora una herramienta que, mediante Inteligencia Artificial, para automatizar la construcción del libro de vida.

La segunda etapa consiste en llevar la herramienta a los terapeutas y personas que asisten a los afectados para comprobar la funcionalidad y efectividad de la misma.

Nosotros nos centraremos en abordar la primera etapa del proyecto, asegurándonos de implementar una herramienta capaz de generar imágenes a través de una inteligencia artificial especializada en ello, de manera que asegure la calidad máxima posible en los resultados y procurando una semejanza prácticamente total a la realidad.

1.2. Objetivos

- Utilizar la inteligencia artificial generativa para crear imágenes personales que ayuden a los pacientes a evocar momentos y experiencias emocionales e integrarlos en el presente.
- Generar fotografías que respalden historias de los pacientes.
- Brindar material de apoyo para la terapia de reminiscencia.
- Elaborar un programa mediante el cual, un ayudante pueda incorporar imágenes propias al modelo.

Por ejemplo: un paciente recuerda cuando vio el mar por primera vez, pero no conoce detalles suficientes como para tener integrada una historia que contar en la mente, ni tomó ninguna fotografía en aquel entonces. El modelo puede generar una

foto del paciente en el mar, y el hecho de evocar ese recuerdo, le provoca bienestar y felicidad.

De esta manera, podemos aportar un material muy valioso para la terapia de reminiscencia, ya que se necesita material visual que permita crear una conexión con la vida del paciente, y este material en ocasiones puede ser muy limitado.

1.3. Plan de trabajo

Una vez definidos los objetivos, se debe establecer un método para tratar de llegar a los resultados esperados. En primer lugar, se debe realizar una amplia investigación acerca de los tipos de inteligencia artificial que existen y cuál de todas es la que mejor se adapta a nuestro objeto de estudio. Pero antes de profundizar en las diferentes técnicas y modelos, debemos ser conscientes del motivo por el que realizamos este trabajo, es decir, quién es el destinatario y qué espera del producto final. Por tanto, necesitamos indagar en el foco del problema y saber cómo la inteligencia artificial puede ayudar a resolverlo, o bien a mitigarlo.

Sabiendo que para llegar a los resultados deseados necesitamos una inteligencia artificial generativa de imágenes a partir de texto, necesitamos conocer cuáles son las mejores que están a nuestra disposición, y si las podemos utilizar y trabajar sobre ellas. Por tanto, una parte importante de nuestro proyecto consiste en realizar pruebas de cada una de las inteligencias artificiales generativas y valorar cuál genera imágenes de mayor calidad y cuál la genera en un tiempo aceptable. Es necesario probarlas todas y cada una de las que estén disponibles, y saber cuál va a ser nuestro entorno de ejecución.

Una vez se haya decidido cuál va a ser el o los modelos que elegimos para desarrollar nuestro proyecto, llegará el momento de saber cómo vamos a desplegar las diferentes tecnologías y qué vamos a añadir para que sea algo útil y completamente novedoso para nuestros destinatarios. La idea es crear un programa, que contenga el modelo de inteligencia artificial elegido y una interfaz sencilla y eficaz para utilizarla en las terapias ocupacionales. Además, estos modelos deberán tener la posibilidad de ser personalizables, de manera que se puedan crear imágenes con los elementos o personas que se deseen. Para ello, se investigará sobre los distintos modos de entrenamiento, y realizando múltiples pruebas y analizando los diferentes resultados, se razonará cuál será el entrenamiento óptimo para nuestro proyecto.

Cuando ya disponemos de un modelo de inteligencia artificial generativa de imágenes que otorgue buenos resultados, simularemos casos de uso que ejemplifiquen cómo los usuarios pueden tener una experiencia plenamente satisfactoria. A raíz de esto, podremos obtener una serie de conclusiones y anotar si se han cumplido las expectativas y si en un futuro los pacientes pueden ver su calidad de vida incrementada gracias a nuestra iniciativa y a nuestro trabajo.

Capítulo 2

Estado de la Cuestión

A continuación, se abordarán diferentes aspectos referentes al proyecto y, que son de suma importancia para la comprensión de éste. Es preciso saber cómo está actualmente el campo de estudio que trata el proyecto para que al final, se pueda sacar en claro conclusiones de lo que hemos aportado y aprendido realizándolo.

En primer lugar, trataremos en líneas generales lo que es el Alzheimer y en específico, la terapia de reminiscencia y cómo ayuda a paliar esta enfermedad. Una vez ya puestos en contexto, se explicará lo que es la Inteligencia Artificial y cómo funciona exactamente, en concreto, la Inteligencia Artificial generativa de imágenes que funciona a través de redes neuronales. Lo que nos llevará al siguiente punto a tratar, qué son las redes neuronales, las diferentes variedades que existen, qué puntos positivos presenta cada una y cuál hemos elegido y por qué para implementar el modelo base que se utilizará en nuestro trabajo.

2.1. ¿Qué es el Alzheimer?

El Alzheimer es una enfermedad que destruye lentamente la memoria y que, además, también va deteriorando los pensamientos y la conducta, hasta que poco a poco se ven afectadas las funciones más básicas. El Alzheimer es la principal causa de la demencia.

El cerebro envía estímulos químicos a través de las neuronas creando conexiones cerebrales, y mediante miles de millones de estas conexiones, se obtienen nuestros recuerdos, sentimientos, pensamientos y capacidades locomotoras. Aunque todavía no se sabe con certeza el motivo que causa esta enfermedad en el cerebro, se ha investigado que hay dos proteínas en el cerebro que con el tiempo se vuelven tóxicas, tau y beta-amiloide, que se acumulan hasta obstruir la conexión entre las neuronas

y provocar que estas mueran, como se puede ver en ?? Con la destrucción de las neuronas, el cerebro se va encogiendo y con él también severamente, el hipocampo, que es una parte clave fundamental en nuestro cerebro a la hora de formar nuevos recuerdos y para el aprendizaje, lo que causa que nuestra memoria, nuestra capacidad para tomar decisiones y el habla, fallen.

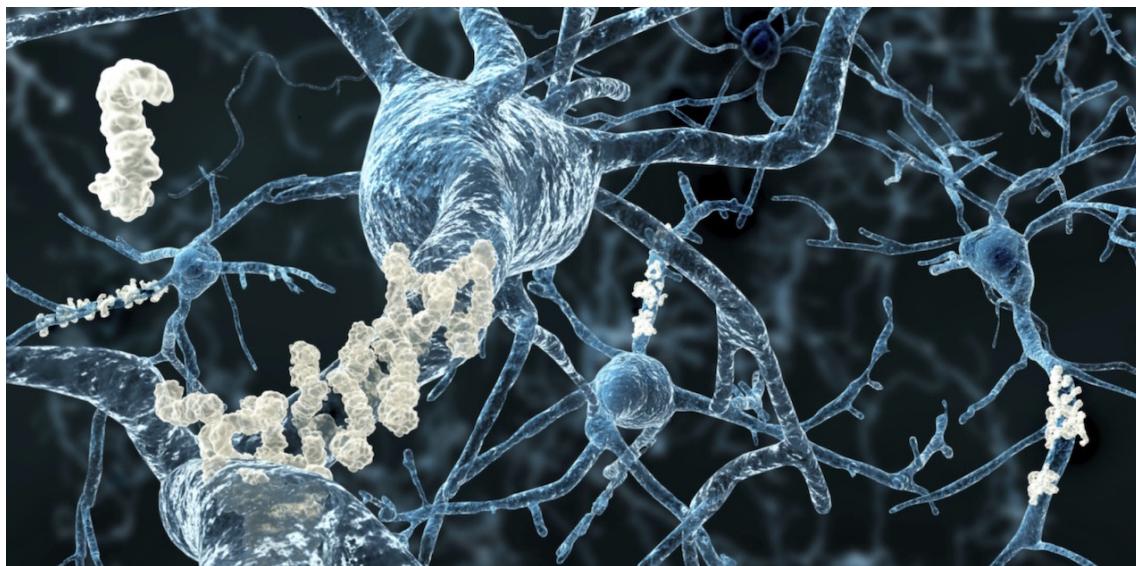


Figura 2.1: Obstrucción de neuronas por la vitamina tau

2.2. La terapia de reminiscencia

Afortunadamente, existen terapias que ayudan a mejorar la calidad de vida de las personas que padecen esta enfermedad. Entre ellas la estimulación cognitiva, orientación a la realidad, ejercicio terapéutico, musicoterapia, y estimulación multisensorial, entre otras.

Nuestro proyecto está ligado a la terapia de reminiscencia, que es un proceso que ayuda a la persona a evocar momentos y experiencias emocionales e integrarlos en el presente, lo cual puede mejorar la autoestima y la calidad de vida.

En concreto, la técnica consiste en mostrar a la persona una herramienta o material visual, musical o incluso, olfativo, vinculado a su propia experiencia o hechos históricos. De esta manera se promoverá una revisión de la vida de la persona, de modo que se consiga una conexión con sus vivencias, con el fin de reconstruir un libro de vida y reforzar su identidad como persona. Esta terapia se puede clasificar como un tipo de ensueño que los lleva a su pasado, lo que les permite entrar en un estado de concentración con el fin de poder reforzar su memoria general, lo cual fortalece al cerebro y desarrolla sus capacidades sociales, estimula recuerdos a través de órganos sensoriales y activa su sentido de la identidad.



Figura 2.2: Terapia de reminiscencia

Algunos de los beneficios de la creación del libro de vida y la terapia ocupacional de reminiscencia, entre muchos otros, son:

Bienestar emocional: Se permite que las personas puedan recordar experiencias positivas y significativas de sus vidas, lo cual puede generar sensaciones satisfactorias, de alegría y felicidad.

Autoconocimiento: Cuando una persona revisa y reflexiona sobre eventos pasados y logros personales, puede obtener una comprensión de sí misma. Mediante la interacción de sus recuerdos, ayudado de la guía de cuidadores y terapeutas, puede evocar aspectos de sus valores, fortalezas y debilidades que de otra manera no serían accesibles.

Sentido y propósito personal: Mediante la rememoración de escenas significativas, las personas pueden encontrar una guía emocional que permite dar sentido y dirección a sus vidas, especialmente en momentos de confusión o desorientación.

Reducción del estrés: Al enfocarse en recuerdos positivos, las personas experimentan una sensación de calma, de tranquilidad y de bienestar emocional, y encuentran un espacio para escapar del estrés y tensiones del presente.

Favorecer las relaciones sociales: El hecho de compartir recuerdos y experiencias con otras personas, puede fortalecer los lazos sociales y fomentar una mayor conexión con cuidadores, familia y amigos, lo cual es altamente satisfactorio para el paciente.

Aumentar el desarrollo del lenguaje y la persona: Al relatar experiencias pasadas, las personas mejoran su capacidad para comunicarse de manera efectiva, así como su expresión verbal, lo cual impulsa un crecimiento personal significativo a nivel emocional e intelectual.

Prevenir la incapacidad: El hecho de potenciar las habilidades lingüísticas y de mantener la mente activa, puede ayudar a preservar la función cognitiva a lo largo del tiempo.

Existen numerosos estudios y tesis que afirman que las capacidades cognitivas se mantienen y se consigue frenar en cierta medida el deterioro, además de reducir la ansiedad y la depresión, por lo que se hace una recomendación y un llamamiento para que se realicen estas actividades de terapia ocupacional.

2.3. ¿Qué es la Inteligencia Artificial?

La Inteligencia Artificial es un campo de la Informática que trata la creación de herramientas, procedimientos, máquinas y computadores que simulen el proceso de inteligencia humana siendo capaces de realizar tareas como el aprendizaje, el razonamiento, la resolución de problemas, el reconocimiento de patrones, la comprensión del lenguaje natural, la percepción visual e incluso la creatividad. Este proceso se lleva a cabo utilizando principalmente el análisis de datos y las estadísticas, la ingeniería de hardware y software, aunque también abarca disciplinas como la lingüística, la neurociencia y hasta la filosofía y la psicología.

Ahora, existen dos tipos principales de inteligencia artificial:
La IA débil que se centra en tareas específicas y está diseñada para realizar funciones específicas sin poseer una inteligencia general. Y, por otro lado, la IA fuerte que busca replicar la inteligencia humana de manera más completa, con capacidad para comprender, aprender y adaptarse a una amplia variedad de tareas, con el fin de percibir su entorno y aplicando los conocimientos y datos almacenados, tener la capacidad de tomar decisiones para lograr la consecución de objetivos proporcionados.

Al hablar sobre inteligencia artificial, es muy importante mencionar los conceptos de aprendizaje automático (machine learning) y aprendizaje profundo (deep learning), que son subconjuntos de la inteligencia artificial que comparten la meta común de permitir a las máquinas aprender y mejorar su rendimiento en tareas específicas sin intervención humana directa.

El aprendizaje automático es un enfoque que abarca diferentes técnicas en las que se crean modelos de entrenamiento a partir de datos para aplicar los conocimientos

adquiridos con el fin de realizar predicciones o tomar decisiones en función de nuevos datos. Es decir, se utiliza para mejorar el rendimiento de la inteligencia artificial en tareas específicas a medida que se expone a mayor cantidad de datos, que en general requiere intervención humana de forma manual.

Algunas categorías dentro del aprendizaje automático son:

- Aprendizaje supervisado: Un algoritmo se entrena con un conjunto de datos etiquetado, donde se le proporcionan ejemplos de entrada y la salida esperada. El modelo aprende a realizar predicciones o tomar decisiones basándose en estos ejemplos.
- Aprendizaje no supervisado: El algoritmo se enfrenta a datos no etiquetados y debe encontrar patrones o estructuras por sí mismo. Esto se utiliza comúnmente para la clasificación o agrupación de datos.
- Aprendizaje por refuerzo: El modelo aprende a través de la interacción con un entorno. Recibe recompensas o castigos según las acciones que realiza, lo que le ayuda a aprender qué comportamientos son más beneficiosos.

El aprendizaje profundo, a lo que respecta, es una disciplina dentro del aprendizaje automático que utiliza redes neuronales profundas con múltiples capas, que contienen datos de entrada y salida, de manera que cada capa puede aprender de la anterior a partir de los datos transformados para realizar predicciones y aprender de manera automatizada, reduciendo gran parte de la intervención humana y permitiendo aumentar el conjunto de los datos a tratar.

2.4. Redes neuronales

Las redes neuronales fueron creadas con el objetivo de simular el comportamiento del cerebro humano para que las máquinas pudieran ser capaces de aprender tareas computacionales de forma similar a las neuronas de nuestro sistema nervioso. La forma más básica de una neurona artificial se puede definir a través del concepto de perceptrón. El concepto de perceptrón fue visto por primera vez en 1956 por Frank Rosenblatt, psicólogo estadounidense. Un perceptrón es una unidad de red neuronal que funciona a través de un algoritmo de clasificación que realiza determinados cálculos en el que se ponderan las entradas, sumándolas y generando así una salida binaria, de manera que permite separar los datos en dos categorías. Para ello, cada entrada tiene asignada un peso que determina la relevancia de esta para su cálculo posterior en la suma ponderada, es decir, cada entrada se multiplica por su peso correspondiente para determinar su importancia relativa. La salida, además, se calcula aplicando una función de activación al resultado de la suma ponderada

previamente calculada para así, dictaminar la salida final.

El perceptrón puede constar de una sola capa, que se compone de entrada y salida, o puede constar de múltiples capas, más conocidos como MLP (Multilayer Perceptrons). A diferencia de los perceptrones simples, los MLP constan de unas capas ocultas adicionales, componiéndose así de múltiples neuronas conectadas que se dividen típicamente en tres capas: capa de entrada, capa oculta y capa de salida. En resumidas cuentas, los perceptrones son la forma más simple de una red neuronal.

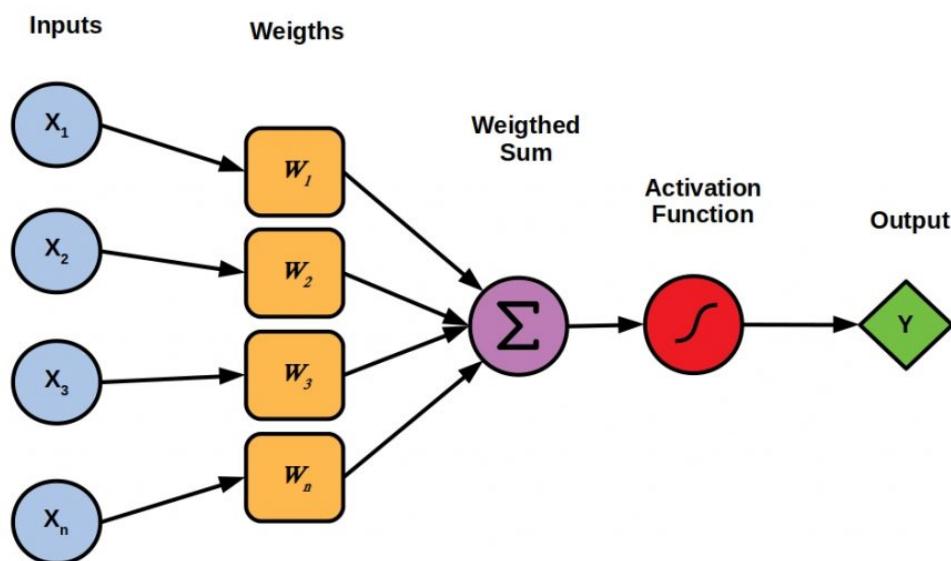


Figura 2.3: Estructura del perceptrón

La primera imagen generada por Inteligencia Artificial fue en 1957, por el propio creador del perceptrón, Frank Rosenblatt, quien entrenó al propio perceptrón con una serie de imágenes de rostros humanos para que éste aprendiera e identificara un patrón y reprodujera una nueva imagen. La imagen fue generada a través de una matriz de puntos de luz que aunque no se asemejara a lo que realmente era una fotografía de una persona real, hizo que este fenómeno marcará un antes y un después en el desarrollo de la Inteligencia artificial generativa.

¿Qué tipos de redes neuronales existen?

La variedad de redes neuronales es considerablemente grande, y cada una de ellas se han ido desarrollando y diseñando para elaborar tareas específicas. De esta

manera, las diferentes redes neuronales han ido adoptando diferentes arquitecturas para tratar diferentes tipos de datos y problemas.

Entre ellas, mencionaremos las más relevantes hoy en día y, analizaremos el funcionamiento de cada una para tener claro cuál es la más adecuada para este modelo, haciendo una profunda comparación entre unas y otras sobre todo, nos centraremos en la red neuronal convolucional, ya que es la implementada para este trabajo.

2.4.1. Redes Neuronales Feedforward (FNN)

Son un tipo de redes multicapa, que como hemos visto, están formadas por conjuntos de neuronas agrupadas en varios niveles o capas, en los que cada neurona está conectada y recibe señales de otras neuronas pertenecientes a la capa anterior, que a su vez, se encargan de transmitir información por señales a las neuronas de la capa posterior, en dirección a la salida de la red. De esta forma, las salidas de cada capa constituyen la entrada a la capa inmediatamente posterior. En todo caso, las conexiones de la red fluyen exclusivamente en una sola y única dirección, de ahí el nombre feedforward, que traducido es hacia delante. Por norma general, la arquitectura típica que sigue una red neuronal multicapa consta de tres capas: capa de entrada, capa oculta y capa de salida.

Sin embargo, cabe la posibilidad de que haya más de una capa de cada tipo, y cuando se da el caso de que la red consta de más de una capa oculta, la red se califica como profunda, traducida al inglés como deep neural network. Concretamente, añadir más de una capa oculta a la red permite crear un modelo interno que reconoce patrones y proporciona un mayor rendimiento en la interpretación y estructuración de diferentes propiedades de objetos. Por ello, estas redes fueron diseñadas específicamente para resolver problemas de clasificación, regresión y por supuesto, como hemos visto, realizar reconocimiento de patrones.

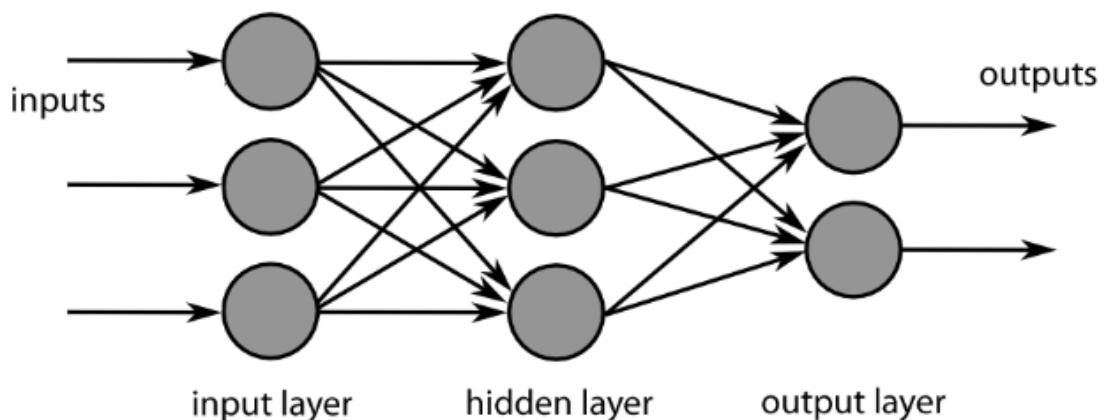


Figura 2.4: Red neuronal feedforward

2.4.2. Redes Neuronales Recurrentes (RNN)

Como hemos visto, las redes neuronales feedforward están habilitadas para que la información fluya en una sola dirección. Sin embargo, cuando se les proporciona una memoria, el resultado que se obtiene son las redes neuronales recurrentes, o Recurrent Neural Networks (RNN) en inglés. El origen de estas redes se popularizó en 1982 por el físico americano John Hopfield, en las que se destacan por su comportamiento dinámico y estable.

¿Y cómo es posible añadir memoria a las propias conexiones? Simplemente generalizando sus conexiones, es decir, alimentar a sus propias entradas o inputs con las salidas o outputs generados previamente por las conexiones anteriores provocando que el modo en el que fluyen las conexiones sea bidireccional. Es decir, incluyendo conexiones hacia atrás con las que se trabaja en una serie de pasos de tiempo, conocidos como timesteps, donde se procesan los elementos de la secuencia uno por uno, manteniendo una memoria de los estados anteriores a medida que avanzan en la secuencia.

El entrenamiento de redes neuronales de estas características se realiza contando con diferentes algoritmos y técnicas, la más básica y conocida en este tipo de red neuronal es el algoritmo de propagación de errores (back-propagation en inglés) formalizado en 1986 por Rumelhart, Hinton y Williams. Este método, en concreto, consiste en aplicar un patrón a la primera capa de la red, el cual se va propagando hacia las capas superiores con el objetivo de generar una salida que se pueda comparar con la salida deseada y así poder calcular el error para cada neurona de la salida obtenida en función de los diferentes parámetros de la red. En otras palabras, cómo varía el error en relación con la variación de los parámetros de la red neuronal. Si a este fenómeno le sumamos la variable del tiempo, se obtiene la técnica de retropropagación a través del tiempo (backpropagation through time, BPTT), en la que después de analizarse la secuencia completa, calcular el error y cambiar los parámetros para minimizar el mismo, se propaga el error a través del tiempo desde la última capa hasta la superior en cada paso de tiempo con el fin de que la red aprenda de las secuencias y mejore su predicción futura. Este método combinado con la técnica del gradiente descendente encargada de la optimización del error buscando los parámetros adecuados para poder reducirlo al mínimo es el modo de aprendizaje supervisado más popular que existe. Aunque, como hemos visto anteriormente, no es el único, ya que coexiste con técnicas de entrenamiento como el aprendizaje no supervisado y el aprendizaje por refuerzo.

Así que, podemos concluir en que las redes neuronales son idóneas para tareas de modelado de datos secuenciales y con dependencias temporales, como lo pueden ser el procesamiento de lenguaje natural, la generación de texto o la traducción automática, entre muchas otras.

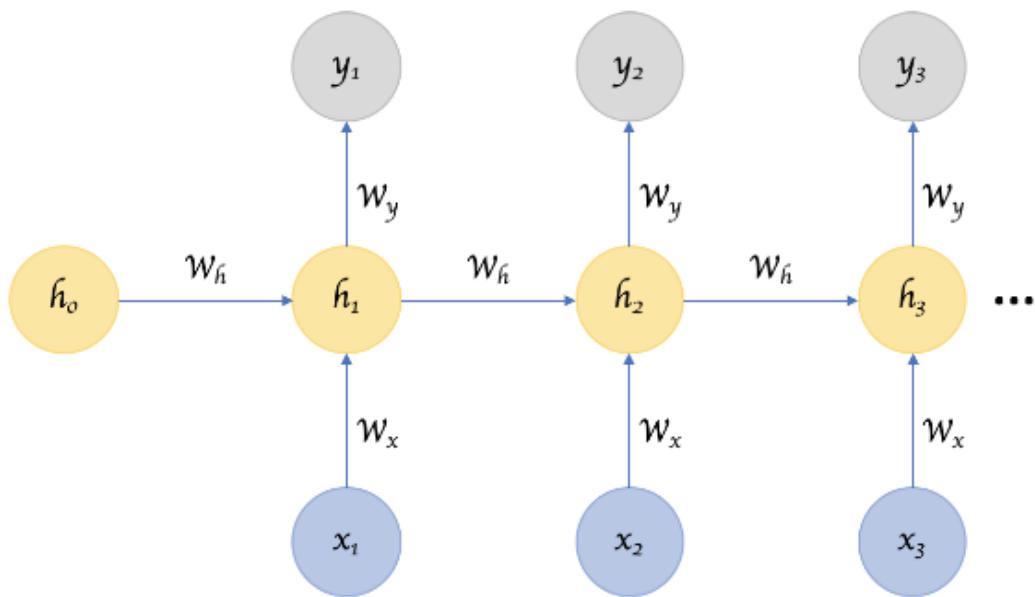


Figura 2.5: Red neuronal recurrente

2.4.3. Redes Neuronales LSTM

Las redes neuronales Long Short-Term Memory surgen a partir de las redes neuronales recurrentes cuyo principal objetivo y propósito es ratificar el problema del desvanecimiento del gradiente.

En primer lugar y para entender en qué consiste este problema, es necesario saber qué es exactamente un gradiente y a qué nos referimos cuando hablamos del desvanecimiento del mismo. Un gradiente es una medida que determina la variación de una función con respecto al cambio que se produce en sus variables, ya sea en términos de optimización, rapidez o maximización de la función objetivo.

Matemáticamente, se puede entender como un vector multivariable cuyas variables son las derivadas parciales de dicha función. Se puede ver en el ejemplo de la figura. Geométricamente hablando, el gradiente indica la dirección en la que la función crece a mayor velocidad y, de esta manera, podemos concluir que el uso de los gradientes se hace con el fin de ajustar los parámetros para reducir al mínimo la diferencia entre los valores obtenidos y los deseados. Un claro ejemplo de ello son los modelos de predicción, en los que se busca reducir dichas predicciones con los valores reales.

Ahora bien, cuando hablamos del desvanecimiento del gradiente nos referimos al fenómeno que ocurre en la propagación hacia atrás, cuando a medida que se va produciendo la propagación en capas cada vez más profundas, el gradiente va dis-

minuyendo hasta llegar a ser tan pequeño que las capas superiores sean incapaces de aprender de manera efectiva. Este tipo de problema es común en redes conformadas por una gran cantidad de capas, dado que se disminuye exponencialmente a medida que va recorriendo cada capa hasta impedir que los pesos se actualicen de manera correcta y que puedan tener un aprendizaje lo suficientemente eficiente para ser capaces de resolver patrones complejos. La solución a este problema se abordó introduciendo nuevos componentes a la red que ayudan y controlan el flujo de información de la red, consiguiendo aumentar la memoria del conjunto de la red al almacenar la información durante períodos más largos de tiempo. Los componentes principales son los siguientes: Memory cells o celdas de memoria: contienen la información relevante y actualizada a lo largo del tiempo.

Input gates o puertas de entrada: son las encargadas de controlar la cantidad de información que entra a las celdas de memoria.

Forget gates o puertas de olvido: seleccionan la información que debe ser eliminada de la celda, al no ser de utilidad.

Output gates o puertas de salida: su tarea radica en seleccionar la información de la celda que va a pasar a la siguiente capa de la red, basándose en el estado oculto actual.

Para entender cómo funcionan y se relacionan estos componentes en la estructura de la red, lo veremos con un ejemplo hipotético en el que la celda de memoria representa una caja fuerte y las diferentes puertas son componentes de una cinta transportadora.

La caja fuerte contiene información valiosa que se transporta en la cinta, el la puerta de entrada actúa como una máquina que trabaja en la cinta que se encarga de introducir nueva información a la caja fuerte y regular la información contenida en esta; la puerta de olvido actúa como una máquina que descarta fragmentos de información contenidas en la caja fuerte; y por último, el componente que representa la puerta de salida, se encarga de elegir si la información contenida en la caja fuerte es apta para salir de la cinta transportadora.

De este modo, se consigue erradicar el problema del desvanecimiento del gradiente y se consigue procesar grandes secuencias de datos durante largos períodos de tiempo. Es especialmente útil en tareas que requieren capturar grandes dependencias de datos temporales, ejemplo de ello son el procesado y/o reconocimiento de lenguaje natural, en la traducción automática y en la generación de texto.

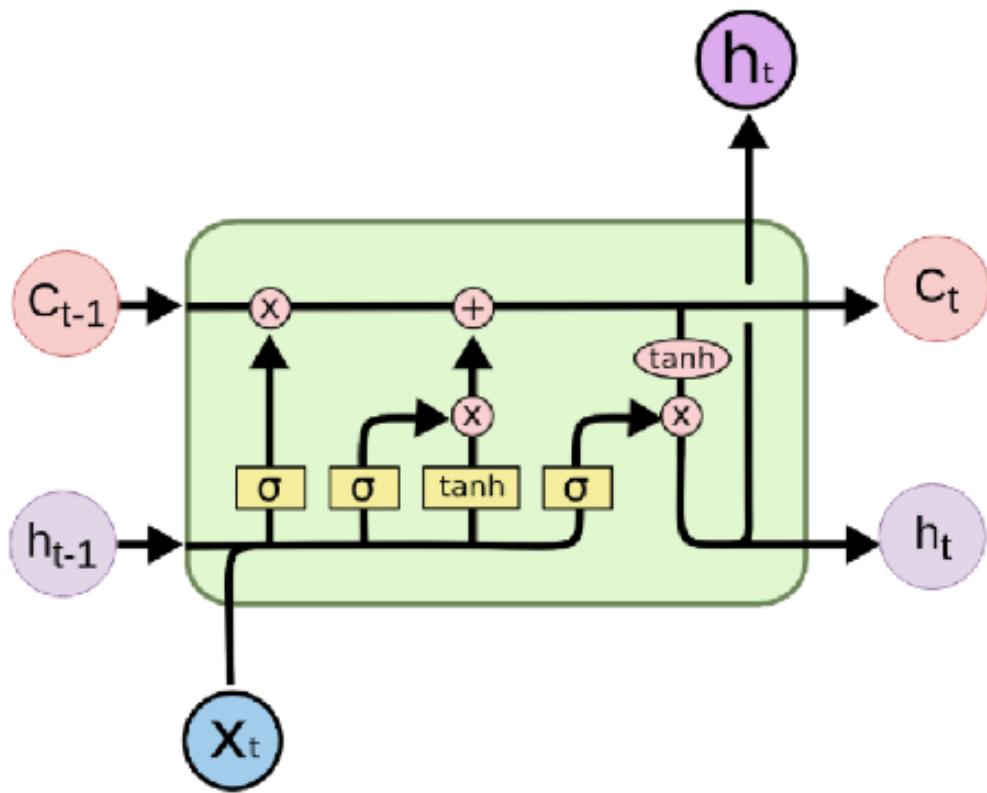


Figura 2.6: Red neuronal Long Short Term Memory

2.4.4. Redes Neuronales Convolutivas (CNN)

Las redes neuronales convolucionales se introdujeron por primera vez en la década de los 50 por David Hubel y Torsten Wiesel cuando experimentaron con las neuronas biológicas, lo que le sirvió de inspiración a Kunihiko Fukushima en la década de los 80 para desarrollar el Neocognitron, una red neuronal que se conoce como la primera CNN. Este concepto fue cobrando forma con el paso de los años y el modelo tal y como lo conocemos hoy en día, fue obra de Yann LeCun en 1998 al introducir el aprendizaje a través de la técnica de backforwarding.

Las redes neuronales convolutivas están formadas por una secuencia de capas que podemos clasificar en tres tipos: capas convolutivas, capas de pooling y capas completamente conectadas.

Capas convolutivas: suponen la capa principal de la red, y el papel que desempeña en el procesamiento de imágenes, es realizar la gran parte de los cálculos que se necesitan para extraer características de las imágenes. Para ello, hace falta la intervención de tres elementos principales: datos de entrada, un filtro y un mapa de características.

Los datos de entrada son el elemento que se trata de analizar, por ejemplo, en el

caso de una imagen de color que estuviera compuesta por una matriz de píxeles en 3D, las dimensiones serían la altura, la anchura y la profundidad de la misma.

Por otro lado, el filtro o kernel, se trata de un detector de características, que va pasando por cada área de la imagen para identificar diferentes características, este proceso se denomina convolución.

El siguiente paso es representar mediante una matriz bidimensional de pesos, que al aplicarse en cada área calcula un producto escalar a partir de los píxeles de los datos de entrada y del filtro. El producto escalar se utiliza como input en la matriz de salida para que el filtro sea capaz de repetir el proceso por toda la imagen. La finalidad del filtro es ser capaz de distinguir diferentes patrones, como lo pueden ser texturas, bordes o figuras.

Finalmente, la suma de los diferentes productos escalares y el o los filtros utilizados, da como resultado final lo que se conoce como mapa de características.

Después de cada capa de convolución, se aplica una función de activación no lineal al mapa de características, a través de la ReLU (Rectified Linear Unit). El fin de esta función es introducir no linealidad a la red, lo que le permite mejorar la complejidad entre las diferentes características.

Capa de agrupación o pooling: son capas dedicadas a reducir la dimensión del mapa a través de la disminución del número de parámetros de entrada.

Comúnmente, se utilizan dos técnicas conocidas como max pooling y average pooling. Max pooling consiste en seleccionar el pixel con mayor valor a medida que el filtro recorre la imagen para enviar el máximo a la matriz de salida, en cambio, average pooling lo que busca es calcular el valor medio del campo. El inconveniente que presenta esta capa es la gran pérdida de información que existe, sin embargo, presenta una gran ventaja a la hora de reducir la complejidad computacional y concentrarse en las características más importantes, lo que claramente, mejora el rendimiento del modelo y evita el riesgo de que se produzca un sobreajuste.

Capa totalmente conectada: en las últimas capas de la red los nodos están conectados con los de la capa anterior para producir la salida final incorporando las características extraídas y aprendidas de los procesos realizados en las capas anteriores. Las capas totalmente conectadas se preocupan de realizar las funciones de clasificación de la imagen y de regresión para producir el resultado deseado.

Las principales funciones y tareas que abarcan las redes neuronales convolucionales son el reconocimiento de imágenes identificando y detectando objetos, personas o animales; análisis de imágenes para diferentes propósitos, por ejemplo, médicos; reconocimiento facial; segmentación semántica; y por supuesto, generación de imágenes.

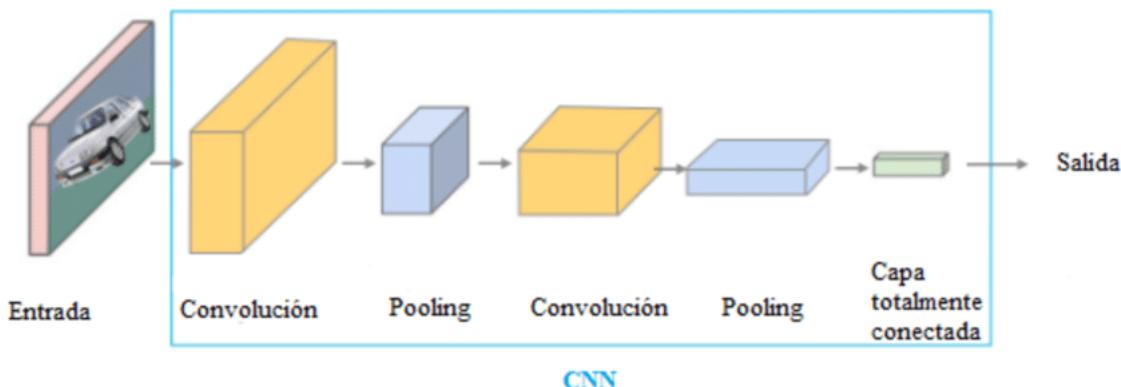


Figura 2.7: Red neuronal convolucional

2.4.5. Redes Generativas Adversarias (GAN)

Las redes GAN, del inglés, Generative Adversarial Networks son un tipo de redes neuronales profundas revolucionarias, que son relativamente novedosas, ya que surgieron en el año 2014 por primera vez por Ian Goodfellow y sus compañeros en la Universidad de Montréal.

El funcionamiento de este tipo de redes involucra a dos redes neuronales diferentes, en las que cada una se encarga de realizar una tarea específica y “competir” contra la otra, para así, cada vez ir mejorando más los resultados obtenidos.

La primera red neuronal de este sistema se conoce como “Generador” y se encarga de producir y crear datos totalmente nuevos basándose en los datos aportados en el entrenamiento de la red. Se le asigna como entrada un vector de ruido y es responsable de crear datos que se asimilen a los datos originales. El entrenamiento del Generador es constante y siempre busca mejorar los resultados a medida que los va produciendo para alcanzar el máximo realismo posible.

La segunda red neuronal de la que se compone este sistema es el “Discriminador”, se dedica a analizar los resultados producidos por el Generador e identificar si son los reales o los creados por la red. A medida que va avanzando su entrenamiento, la capacidad del discriminador en distinguir entre un resultado real o falso dada una cierta entrada va mejorando cada vez más, haciendo que su predicción sea más certera. Un clásico ejemplo llevado a la vida real de esto puede ser el caso de un falsificador de billetes y un detective, en el que el primero tiene como muestra cierta cantidad de billetes e intenta replicarlos para que más adelante el segundo agente intente detectar la copia del original. A medida que pasa el tiempo, cada uno de los dos individuos van mejorando en su tarea llegando a un nivel de equilibrio.

Matemáticamente, el discriminador tiene que generar una salida, expresada como

$D(x)$, basándose en la probabilidad de que la entrada sea sintética o real, suponiendo que en cuanto más cercana a 1 sea, la entrada es original. Y por el contrario y dada una muestra aleatoria z en función de cierta distribución de probabilidad, el generador tiene que producir una muestra, expresada como $G(z)$, que el discriminador tiene que clasificar como cercana a 0, produciendo una salida del tipo $D(G(z))$ en la que el generador tiene que intentar que su probabilidad se aproxime a 1, justo al contrario que el discriminador. Suponiendo que entre todas las muestras del modelo, una mitad son auténticas y la otra mitad son falsas, se tiene que alcanzar el conocido como equilibrio de Nash en el que las muestras del modelo son igual a los datos y en las que la probabilidad del discriminador es $D(x) 0,5$ para todo x .

Por último, cabe destacar que para el entrenamiento de este tipo de red se utilizan los métodos del gradiente descendente y backpropagation, vistos en redes neuronales anteriores.

Este tipo de redes son ideales para tareas que requieren creación de datos realistas y artísticos, como lo pueden ser la generación de imágenes, de música o incluso, síntesis de voz.

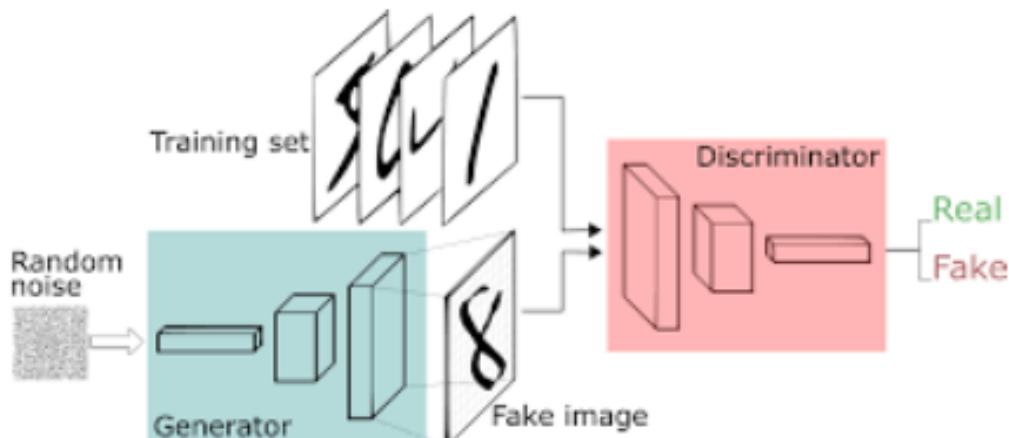


Figura 2.8: Red neuronal generativa antagónica

2.4.6. Redes Neuronales Transformer

Las redes neuronales Transformer son las más actuales tratadas en este trabajo, surgieron en 2017 a través del artículo “Attention is all you need” elaborado por Vaswani et al.. El artículo trata una mejora de las redes recurrentes y convolucionales conocidas introduciendo un mecanismo basado sólidamente en la atención, es decir, realizar paralelismo entre las diferentes tareas que realiza la red. Para ello, se parte de una entrada mediante embedding y se hace uso de dos componentes: codificador

y decodificador.

Embedding: es el primer bloque por el que está compuesto la red neuronal, su función es primordial para el manejo de los datos, ya que transforma el texto de entrada en unos determinados vectores o tokens, los cuales son la representación numérica del valor inicial.

Codificador o encoder: después del embedding de la entrada, el siguiente bloque es el codificador posicional cuya tarea es indicar a la red el orden de los diferentes elementos del vector, es decir, de las palabras en el texto. Esta función es esencial ya que la secuencia se procesa en paralelo, y del contrario, no se podrían concretar las posiciones de cada uno.

A continuación, se encuentran conectados en secuencia los codificadores. Cada codificador se compone de 4 elementos: el bloque residual, una red neuronal, otro bloque residual, y por último, un bloque atencional siendo el más importante de todos al encargarse de determinar la relevancia de cada uno de los tokens para la frase junto a su asociación. Los restantes elementos se encargan de normalizar la entrada y la salida para poder seguir entrenando la red de forma productiva.

Decodificador o decoder: cada codificador está conectado a un decodificador, que cumplen una composición similar, sino igual, a la explicada en los codificadores, es decir, cuatro elementos: dos bloques residuales, una red neuronal y un bloque atencional. A los que se le añaden dos elementos más en la estructura: un tercer bloque residual y un bloque atencional con enmascaramiento.

Sin embargo, el funcionamiento difiere al de la estructura anterior. Se comienza con el bloque atencional con enmascaramiento que codifica las relaciones entre elementos atendiendo únicamente a palabras actuales y pasadas. Por otro lado, el bloque atencional del decodificador se conecta con el del codificador para establecer el orden de prioridad al que se debe prestar atención en la secuencia, sus valores son probabilidades entre 0 y 1 siendo el valor más alto el seleccionado. Por último, el comportamiento de los demás bloques cumplen las mismas funciones que en el codificador, incluyendo entre ellos, los bloques residuales, el bloque de codificación posicional y la salida en función del embedding.

Por lo tanto, podemos afirmar que estas redes neuronales aprenden contexto y significado mediante el seguimiento de relaciones en datos secuenciales. Todo tipo de organizaciones utilizan estos modelos para conversión de secuencias, incluidas las de reconocimiento de voz y la traducción automática. Esta red neuronal procesa secuencias largas con cálculo paralelo, con el objetivo de reducir significativamente el tiempo de entrenamiento y de procesamiento. A raíz de este modelo, surgen técnicas innovadoras como el aprendizaje por transferencia y la generación aumentada

de recuperación (RAG). El objetivo principal es entrenar inicialmente los modelos de conjuntos de datos amplios y después refinarlos de manera precisa, utilizando conjuntos de datos más específicos. De este modo, se maximiza su utilidad y relevancia dentro de cualquier contexto.

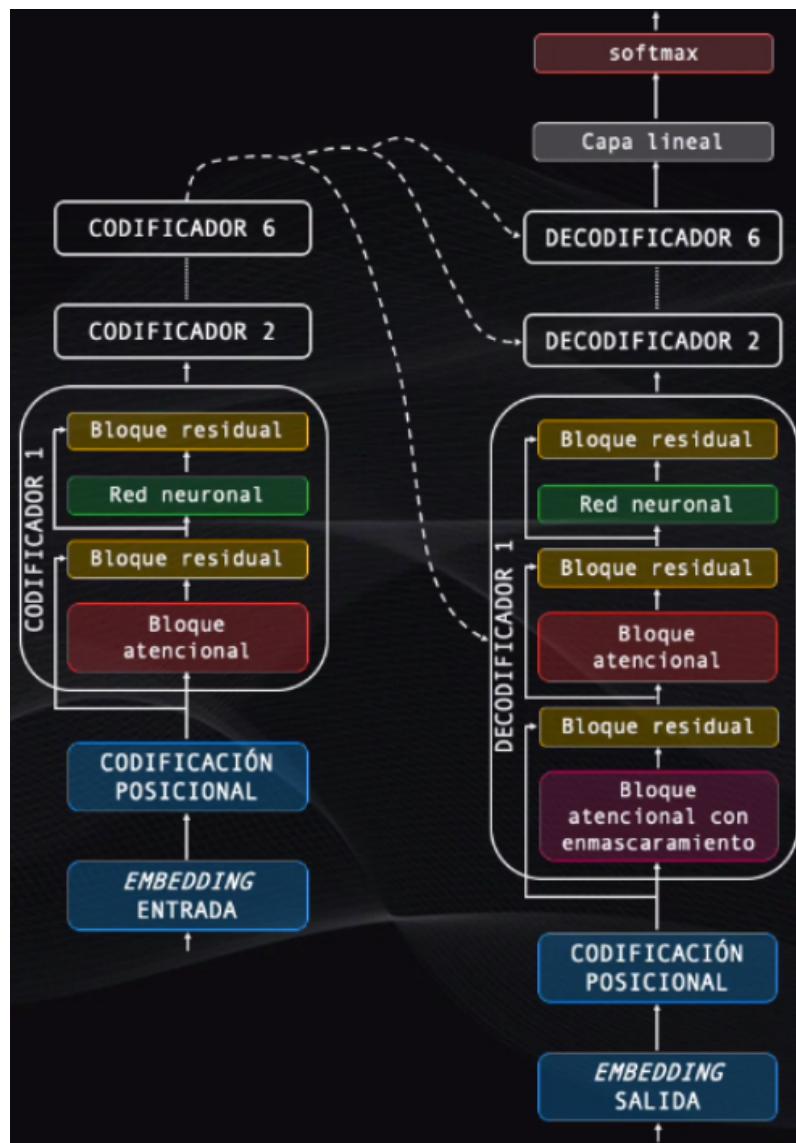


Figura 2.9: Red neuronal Transformer

Estos suponen sólo algunos ejemplos de los tipos de redes neuronales más comunes y ampliamente utilizadas. Cada tipo tiene sus propias características, fortalezas y debilidades, y es importante elegir el tipo adecuado según el problema específico que se esté abordando y el tipo de datos disponibles.

2.5. Stable Diffusion

Como hemos visto, hay diferentes redes neuronales destinadas a la generación tanto de texto como de imágenes. En el caso de Stable Diffusion, el modelo de IA utilizado en nuestro proyecto, funciona con un modelo de difusión latente basado en CNNs y en Transformers. Ya visto el comportamiento y funcionamiento de ambas redes neuronales en apartados anteriores, analizaremos más en profundidad de qué manera se integran y complementan en el modelo que utilizaremos más adelante.

Stable Diffusion es un modelo de Inteligencia Artificial generativa cuya principal función es transformar el texto a imagen, aunque también presenta otras funciones asombrosas como la transformación de imagen a imagen o incluso lo más novedoso hasta el momento, transformaciones de texto a vídeo. Las empresas desarrolladoras hicieron una colaboración conjunta entre CompVis LMU, Runway y Stability AI y el lanzamiento finalmente se produjo a mediados de 2022, es decir, esta poderosa herramienta relativamente nueva y los avances tecnológicos que ha alcanzado hasta la fecha son impresionantes.

2.5.1. Funcionamiento interno de Stable Diffusion

En principio, Stable Diffusion cuenta con un entrenamiento de más de 5 millones de imágenes proporcionado por el dataset Laion-5B que permite contar con una gran variedad de opciones de creación de imágenes, desde objetos, animales, paisajes y lugares, personas e incluso celebridades mundialmente conocidas con una calidad notable.

Comencemos entendiendo el funcionamiento del modelo de difusión latente, y para facilitar su explicación y comprensión, lo disecaremos en dos partes: difusión y el espacio latente. El proceso de difusión que sigue la generación de una imagen es, en primer lugar y partiendo de las millones de imágenes del dataset mencionadas, se procede a añadir ruido gaussiano gradualmente a través de una serie de pasos hasta que las fotografías pierden todo valor y terminan siendo irreconocibles. Se podría decir que se comporta como una cadena de Markov, al ser el ruido gaussiano una variable aleatoria y al depender cada paso exclusivamente de su anterior. Este primer método se conoce como difusión directa hacia delante o forward diffusion.

Así mismo, cuando se procesa una petición dado un prompt, se parte de una imagen únicamente hecha de ruido aleatorio, es decir, un imagen sin nada relevante ni identificable en ella. Partiendo de esta imagen llena de ruido, se intenta revertir lo hecho anteriormente para volver a la imagen original quitando el ruido gradualmente. El ruido es aleatorio, y su aleatoriedad depende de un parámetro llamado seed o semilla que asocia el ruido generado con un número aleatorio. Por lo tanto,

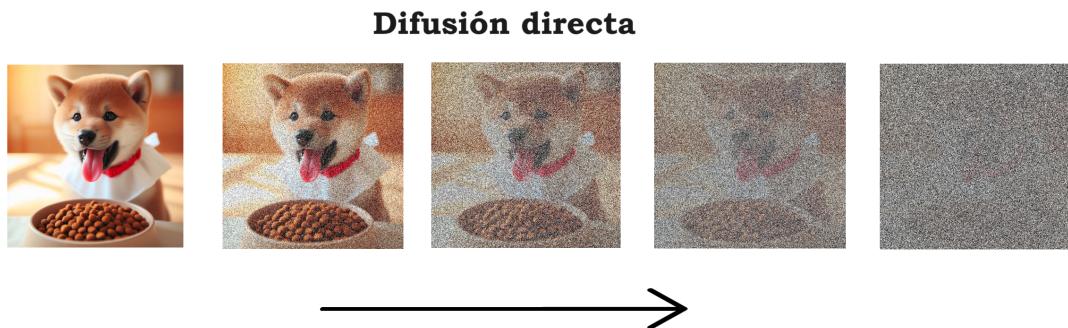


Figura 2.10: difusión directa de una imagen de un cachorro de Shiba Inu comiendo

si se repite la semilla se volverá a generar exactamente el mismo ruido y tendríamos como resultado una réplica de una imagen ya generada con esa misma semilla. Este proceso se conoce como difusión inversa y su objetivo principal es que el modelo aprenda a eliminar el ruido completamente.

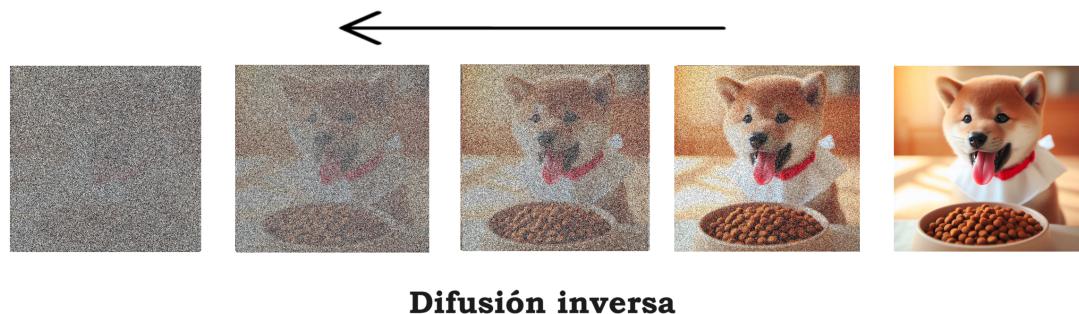


Figura 2.11: difusión inversa de una imagen de un cachorro de Shiba Inu comiendo

Una vez tenemos la imagen llena de ruido y la forma de calcular el ruido que hay en cada imagen para poder eliminarlo, se empieza el proceso al que llamamos sampler o muestreo. Aquí es cuando entra un componente importante llamado noise scheduler, cuya función es determinar la cantidad de ruido que se debe suprimir en cada paso para alcanzar la forma óptima y que se puedan evitar cambios bruscos entre paso y paso, que sea de forma gradual y que los detalles se vayan puliendo conforme la imagen vaya cobrando más forma. El proceso de muestreo se repite la cantidad de steps o pasos especificada por el usuario. Y de esta manera, concluimos con el primer componente del modelo de difusión latente: la difusión.

Pasemos a la segunda pieza del puzzle: el espacio latente. El proceso de difusión, por lo que hemos podido ver, es un proceso algo lento y costoso al tener que trabajar con los píxeles de una imagen. Si tenemos una imagen en color con la escala RGB de 512x512 píxeles, estamos hablando de una multiplicación de 3x512x512, es decir, un espacio de casi 80 mil dimensiones. Para solucionar este problema, se trabaja

con el espacio latente que trata de reducir la imagen a una escala de 64x64 píxeles, asegurando mayor velocidad y una menor carga de trabajo al trabajar, de esta forma, con unas dimensiones más pequeñas de apenas 12 mil.

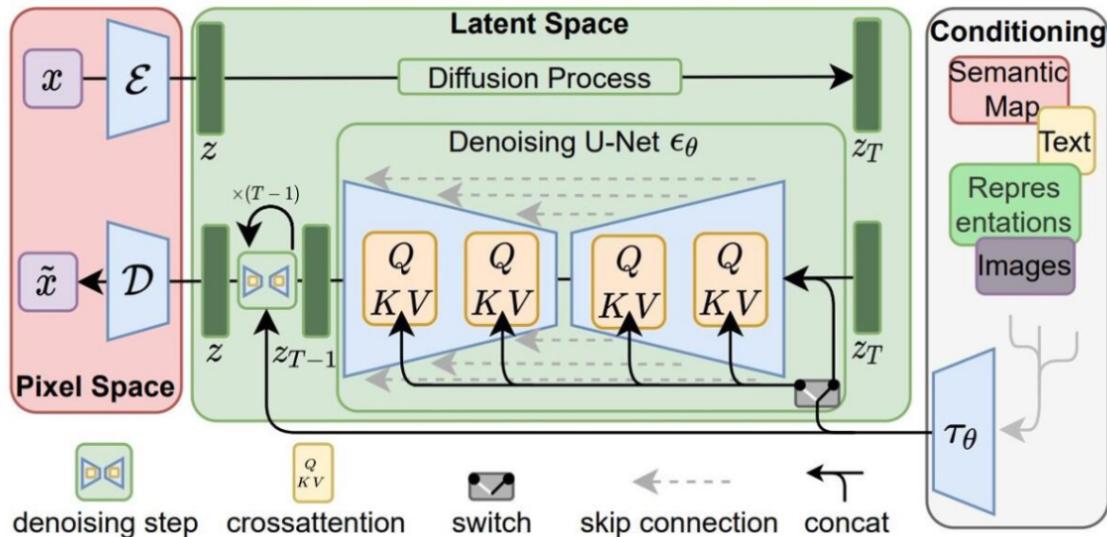


Figura 2.12: Representación gráfica del proceso de conversión al espacio latente

Ahora que entendemos el propósito y el funcionamiento de las dos variables del modelo de difusión latente, profundicemos más en cómo se lleva a cabo cada uno de estos dos procesos y qué mecanismos se utilizan para ello. El modelo se compone principalmente de 3 componentes:

Un codificador de texto basado en redes transformer, una red U-Net compuesta de dos redes ResNet y un autocodificador variacional (VAE).

Codificador de texto basado en Transformers: para producir la imagen que deseamos, previamente es necesario escribir una descripción detallada de la imagen y es por tanto, el primer paso. Este texto que le introducimos al modelo se conoce como prompt y es de lo que se encarga de procesar el codificador de texto. Lo primero que realiza el text-encoder de Stable Diffusion es a través de un modelo llamado CLIP (Contrastive Language-Image Pre-Training) que se encarga de ofrecer una descripción detallada de las imágenes a través de su propio tokenizador. El siguiente paso y como ya hemos visto en apartados anteriores, el transformer se encarga de realizar la fase de embedding en la que se transforman las palabras de texto en tokens que la red neuronal pueda entender y manejar, para que después, a través del método de self-attention, se decida qué palabras son las más relevancia tienen.

Además, Stable Diffusion ha añadido a esto una pequeña variación y mejora que añade a esta última técnica, otra llamada cross-attention (o atención cruzada) con la que se permite crear relaciones entre los diferentes embedding y mejorar la precisión del resultado. Por ejemplo, si el prompt pedido es “unas flores pequeñas sobre una

bicicleta azul”, solamente con la técnica de self-attention podría procesarse una imagen que fuera “una flor azul sobre una bicicleta pequeña”, lo cual es válido para esa arquitectura pero no es lo que el usuario ha pedido. En cambio, con la ayuda de la técnica de atención cruzada se crean relaciones que tienen más a menos distancia, en la que en este ejemplo la relación flores con azul tiene más distancia que la relación de flores con pequeña, y por tanto, es esta última la que se utilizaría al tener más relación y menos distancia.

Red neuronal U-Net: es una red neuronal convolucional entrenada para identificar e intentar predecir la cantidad de ruido contenida en una imagen. Consta de un codificador y un decodificador en los que cada uno de estos componentes son, a su vez, bloques ResNet, que son redes convolucionales profundas compuestas por una gran cantidad de capas. La función del codificador se basa en reducir la calidad de la resolución de la imagen mientras que la función del decodificador es la contraria, generar la imagen en la máxima resolución posible. Entre ambos componentes se añaden conexiones de acceso directo para evitar la pérdida de información importante. Al final, lo que se pretende conseguir es que la red U-Net consiga determinar el ruido para posteriormente, poder conseguir una representación libre de ruido en la imagen final.

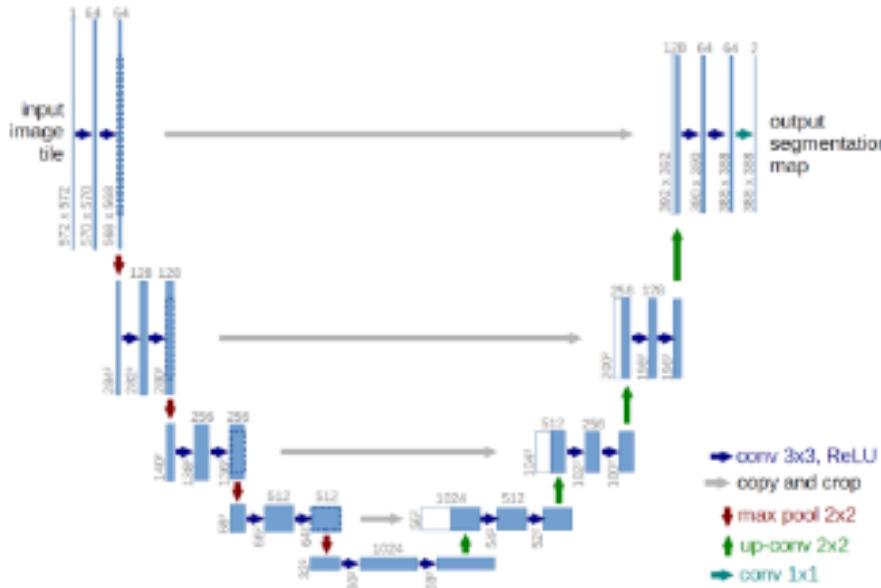


Figura 2.13: Arquitectura de una red U-Net

Autocodificador variacional VAE: es un tipo de red neuronal que al igual que las redes U-Net consta de dos componentes: un codificador y un decodificador, sin embargo sus funciones distan mucho las unas de las otras. El primer componente del autocodificador variacional se encarga de uno de los primeros pasos que realiza Stable Diffusion, que es convertir el espacio de píxeles de la imagen en un tensor dentro del espacio latente de menores dimensiones sustrayendo las características más relevantes de la imagen original y comprimiéndolas en el tensor latente. Y al

final del proceso de difusión, se parte del tensor del espacio latente con el que se ha trabajado, para transformarlo en la imagen final generada en una escala de 512x512 píxeles.

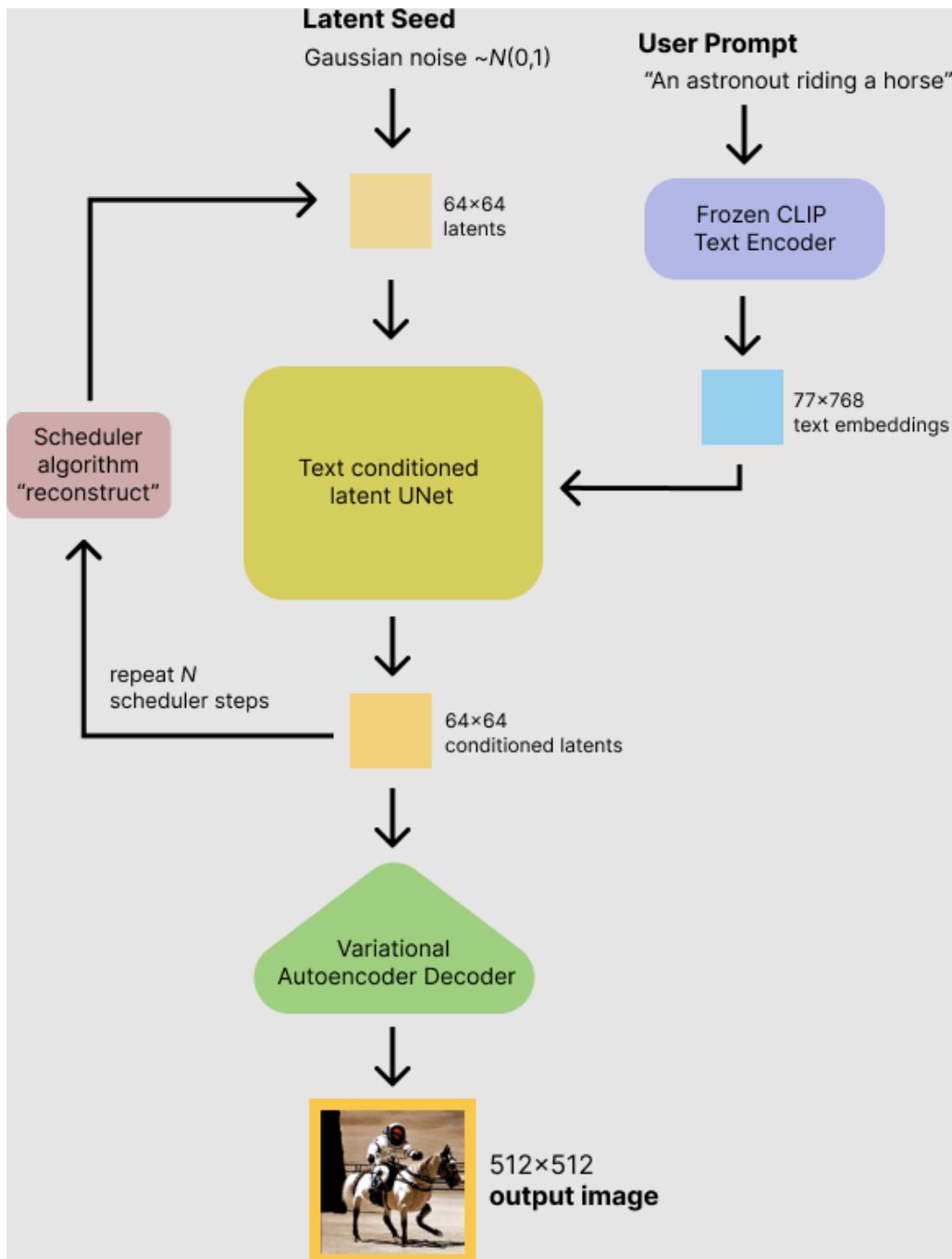


Figura 2.14: Representación gráfica del funcionamiento de Stable Diffusion

Capítulo 3

Generación de imágenes con Inteligencia Artificial

3.1. Elección del modelo

Como hemos podido ver, Stable Diffusion es una herramienta potente y eficaz con una buena estructura que cuenta con una gran base de datos en la que podemos encontrar una inmensa variedad de imágenes. Estas características la convirtieron en nuestra elección final, pero veamos más en profundidad el proceso de elección de esta Inteligencia Artificial generativa en comparación con la amplia variedad de las que hay presentes actualmente en el mercado.

En primer lugar, para entender nuestra elección es necesario poner en contexto el gran abanico de posibilidades de IAs generativas que hay hoy en día, sus prestaciones, características, y sobre todo, su accesibilidad.

Las IAs generadoras de imágenes más potentes del mercado y las que consideramos para desarrollar la base de nuestro trabajo son Midjourney, DALL-E, Leonardo AI y Stable Diffusion, todas ellas producen unos resultados bastante satisfactorios. Sin embargo, descartamos rápidamente las dos primeras: Midjourney producida por un laboratorio independiente y, DALL-E producida por la famosa empresa creadora de ChatGPT, OpenAI. El motivo fue que al ser ambas de pago, no podíamos tener acceso a su modelo de forma tan amplia como las demás, siendo prácticamente imposible acceder a ellas y mucho menos a poder entrenarlas.

Para acceder a los modelos utilizamos Hugging Face, la plataforma que cuenta con una amplia gama de bases de datos de todo tipo y de modelos de IA generativa de texto a imagen, imagen a imagen, imagen a texto y un largo etcétera. Intentamos buscar modelos de ambas en la plataforma sin éxito ya que lo máximo que encontramos eran imitaciones o pequeñas demos que no alcanzaban el nivel de calidad requerido.

Leonardo AI sí que es una opción algo más accesible ya que la plataforma sí que cuenta con un plan gratuito que te deja probar el modelo con una limitación de 150 imágenes a generar al día, lo cual está muy bien. Además, la calidad es bastante buena y permite ajustar gran variedad de parámetros, como por ejemplo, el número de imágenes que deseamos que se generen al mismo tiempo, el estilo, la paleta de colores que deseamos, tamaño e incluso la resolución. El inconveniente era la accesibilidad al modelo y la información sobre este mismo, ya que era escasa y en Hugging Face no había acceso. En comparación con Stable Diffusion, la incertidumbre era mucho más alta y había muchísimas limitaciones a la hora de utilizarlo.

Todo ello nos llevó a elegir Stable Diffusion rápidamente, una herramienta open source, es decir, de código abierto que nos facilita mucho el entrenamiento de personas para obtener imágenes personalizadas de forma rápida y con alta calidad. Además, sus diferentes versiones nos permitían explorar aún más a fondo el modelo y saber cuál era el que encajaba con las características que presentaban las prestaciones de nuestros equipos. Stable Diffusion resultó ser la candidata ideal para que la creación de imágenes destinadas a los libros de vida fuera lo más sencilla, familiar y creativa posible.

Ahora expuestos todos los motivos de elección de Stable Diffusion, tanto en comparación con otras redes neuronales vistas en el Capítulo 2 como en comparación con otros modelos de IA Generativa, expondremos cómo implementamos la herramienta a lo largo de todo el trabajo y las dificultades a las que nos enfrentamos durante el proceso.

3.2. Entrenamiento con Stable Diffusion

3.2.1. Versiones y métodos de entrenamiento

Stable Diffusion cuenta, hasta el momento, con 3 grandes versiones que, paulatinamente, han ido mejorando la calidad en las imágenes generadas. Todas ellas son totalmente gratuitas y de libre acceso, la primera versión que se presentó fue Stable Diffusion 1 (en sus variaciones 1.4 y 1.5), seguida por Stable Diffusion 2 (con sus respectivas variaciones 2.0 y 2.1) y por último, Stable Diffusion XL (que cuenta con su variación XL Turbo).

La diferencia principal entre las dos primeras versiones es el tamaño de resolución de las imágenes ya que las primeras versiones trabajaban en un espacio de 512x512 píxeles y en la versión 2 dicho tamaño aumentó a 768x768. Además, se introdujeron algunas correcciones y mejoras como la técnica de inpainting, que se trata de la restauración de algunas partes de la imagen mejorando la calidad y los detalles de la misma o incluso, reemplazando ese área por lo especificado por el usuario en el prompt.

Con la última versión Stable Diffusion XL, se generan imágenes con una calidad excepcional, lo que supuso una gran mejora en el modelo al contar con un dataset mucho más extenso. El inconveniente con la versión XL, en nuestro caso, era la limitación de que requiere una tarjeta gráfica demasiado potente, con la que, por desgracia, no contamos en nuestro equipo.

Respecto a la versión 2, si bien es verdad que no requiere tanta GPU como la versión XL, sí que requiere más que en la primera versión, al ser las imágenes con mayor resolución y, viendo la comparación en la calidad que presentaban los resultados de ambas, optamos por utilizar la versión 1.5 ya que era la que mejor se adecuaba a nosotros en términos de calidad y tiempo.

Ahora bien, existen varios métodos a través de los cuales se puede utilizar esta herramienta y hemos probado sus funcionalidades de diferentes maneras. En primer lugar, se puede utilizar mediante código escrito en Python a través de Google Colab, ya que la plataforma ofrece cuadernos en los que se trabaja de manera online y que además, proporciona una GPU en la nube a la que Google te da acceso. En concreto, esta GPU es la T4, que es la única opción que nos deja Google entre las que hay (A100 GPU, L4 GPU, V100 GPU) ya que se conoce que las demás son de pago. Otra alternativa es mediante la propia página de Stable Diffusion, que ofrece una demo para utilizar esta avanzada versión. Por último, ejecutar el modelo en local, consiguiéndolo descargar en la página Hugging Face, que incluye multitud de modelos de todo tipo, bases de datos, librerías y licencias para descargar y utilizar, por lo que hemos podido comprobar, presenta muy buenos resultados.

El hecho de probar un modelo de inteligencia artificial en un servidor no es concordante con nuestros objetivos del proyecto, puesto que necesitamos entrenar un modelo e incluirlo en una aplicación, de manera que el usuario pueda interactuar y conseguir imágenes personalizadas en un tiempo aceptable, por ello descartamos la opción de utilizar la demo que se encuentra en la página de Stable Diffusion.

Una vez que tenemos el modelo de generación de imágenes elegido, se debe ejecutar en nuestro ordenador y ver cuál es el rendimiento real. Esto quiere decir que la imagen debe generarse de manera correcta y sin deformaciones, y debe incluir todos los elementos solicitados en la descripción introducida. Además, debe realizar esta generación en un tiempo adecuado.

Para ello, el proceso más óptimo y que finalmente elegimos llevar a cabo tras gran cantidad de pruebas es, en primer lugar, realizando el entrenamiento de imágenes personales a través de la plataforma de Google Colab en internet y en segundo lugar, para la generación de imágenes desde nuestro ordenador optamos por la instalación de una interfaz, llamada NMKD Stable Diffusion GUI. Esta herramienta nos permite ejecutar localmente cualquier modelo de generación de imágenes a partir de texto, e incluso permite aceptar imágenes como input, es decir, generaciones de tipo imagen a imagen.

El principal de los objetivos que establecimos en la realización del proyecto era generar imágenes personalizadas del paciente en cuestión, y para ello es estrictamente necesario entrenar el modelo elegido.

Como se ha dicho anteriormente, el método elegido fue un cuaderno en Google Colab mediante Dreambooth, un modelo de generación de aprendizaje profundo, y que fue desarrollado en 2022 por un grupo de investigadores de Google Research y la Universidad de Boston. Este modelo nos permite añadir capas de entrenamiento a la inteligencia artificial para que reconozca objetos concretos. Esto es muy importante, porque es el mecanismo que consigue mejores resultados y con una velocidad aceptable, que era la utilización que queríamos otorgarle. Por consiguiente, podemos decir que la misión de esta tecnología es la de poder entrenar a modelos de inteligencia artificial para personalizarlo según tus necesidades.

3.2.2. Requisitos en las imágenes de entrenamiento

Para realizar el entrenamiento de una forma correcta, lo primero que tenemos que tener claro es el elemento o token al que queremos dar una identidad. Por ejemplo, si seleccionamos una persona, debemos elegir unas imágenes en las que aparezca, de tal manera que, tras el entrenamiento, la IA pueda identificarla.

Lo ideal es que se elija un número considerable de fotografías, a partir de 10, las cuales tienen que cumplir ciertas características. Deben ser fotografías de buena calidad, bajo diferentes ángulos, escenarios y luces, se recomienda que como mínimo hayan 1 o 2 fotografías en las que la persona aparezca de perfil, mostrando 3 cuartos de la cara, de frente y si es posible que en alguna esté sonriendo (para que la IA pueda reconocer la expresión), de cuerpo entero, cintura para arriba y del rostro de cerca. Además, es importante que la ropa no sea siempre la misma, sino al ser así, el modelo podría interpretar como que la ropa forma parte de la persona y siempre se la generaría con la misma, lo cual no queremos que ocurra. Idealmente, las fotografías deberían alternar la luz y estar hechas tanto en interiores como en exteriores. Por último y esto es fundamental, estas deberán tener un tamaño igual o mayor a 512 x 512 píxeles, y deberán llamarse exactamente de la misma manera, con el identificador del token al que hagamos referencia. Además, es preferible que todas las imágenes tengan la misma extensión, ya sea .jpg o .png.

3.2.3. Procedimiento de entrenamiento

En el ejemplo de la figura 3.1 elegimos como persona de entrenamiento una que no fuese reconocida por nuestro modelo (al contar con un dataset de 5 mil millones de imágenes, ya de por sí reconoce a varias personas famosas sin necesidad de entrenarlas). En este caso, se trata de una actriz coreana llamada Jung Hoyeon, y el token que le otorgamos, como se puede ver, respondía bajo el nombre de "sqgkhoju". Lo ideal, es que la etiqueta no sea nombrada bajo una palabra que el modelo pueda reconocer, es decir, que el token no tenga significado. Si por un casual llamásemos al

token "mujer", seguramente la IA no logre ni identificar ni asociar a la persona que hemos entrenado, y probablemente acabe generando la imagen de una mujer que no existe, que es justamente lo que queremos evitar.

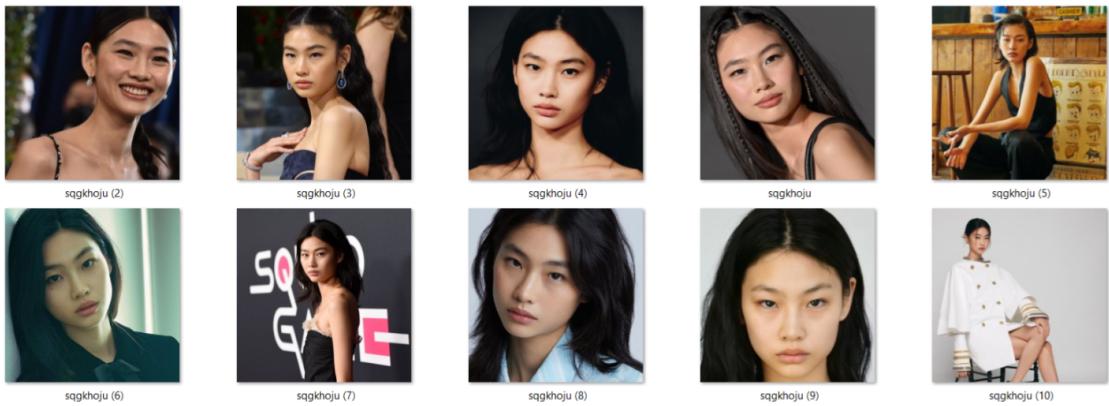


Figura 3.1: Dataset seleccionado para el entrenamiento de personas con Lora

Una vez que las imágenes cumplan con todos los requisitos, debemos utilizar el código abierto en la plataforma Google Colab para realizar el entrenamiento. El proceso que sigue el cuaderno es muy sencillo, en él sólo hace falta seguir una serie de pasos para completarlo. El primero es conectar el cuaderno a una cuenta de Google, para que se pueda guardar en el Drive asociado a esa cuenta una carpeta llamada "Fast-Dreambooth", en la que se guardaran todos los archivos que se generen durante el proceso. El siguiente paso es instalar las dependencias necesarias para ejecutar el código en python, seguido de establecer un nombre a la sesión en la que estamos trabajando, para que en un futuro cuando se hagan entrenamientos diferentes, se puedan distinguir unos de otros. En este paso se crea una carpeta llamada Sessions dentro de la carpeta mencionada recientemente. A su vez, esta carpeta contendrá otra bajo el nombre de la sesión que hayamos especificado en el cuaderno, y es ahí donde se guardaran todos los archivos que se creen en la ejecución.

A continuación, es turno de subir las imágenes previa y cuidadosamente seleccionadas. Ya sea seleccionándolas directamente desde la carpeta en la que las tengamos guardadas en local, o habiéndolas subido previamente a una carpeta de la cuenta de Google Drive, y proporcionar la ruta en la que están en la celda del cuaderno habilitada para ello.

El último paso, y uno de los más importantes, consiste en establecer algunos parámetros con los que se va a entrenar al modelo. El más importante y el único que nosotros hemos modificado, entre todos los que hay, es el número de steps. De modo que, en cuanto mayor sean, más tiempo tardará en generarse el archivo. Normalmente, se tardaba unos 20 o 25 minutos en terminar de entrenarse, lo cual hemos considerado que es bastante rápido. En el ejemplo de la figura 3.2 se puede ver la ejecución del progreso de entrenamiento en el que para una cantidad de 2000 pasos lleva 21 minutos y 36 segundos.

Tras la finalización, se creará un archivo de alrededor de 2 giga bytes, que conten-

drá el modelo de Stable Diffusion 1.5, con una capa de entrenamiento más, puesto que incorporará el elemento deseado. Con esto ya tendríamos un elemento de inteligencia artificial personalizado.



The screenshot shows the Dreambooth training interface. At the top, there's a note about saving steps: "• Minimum 200 steps between each save." Below this, the "start_saving_from_the_step:" field is set to 500. Further down, there's another note: "• Start saving intermediary checkpoints from this step." Underneath, the "Disconnect_after_training:" checkbox is unchecked. A third note states: "• Auto-disconnect from google colab after the training to avoid wasting compute units." At the bottom left, there's a "Mostrar código" link. The main area displays the command "Training the UNet..." followed by several lines of code starting with '#'. At the very bottom, a progress bar indicates "92% 1830/2000 [21:36<01:59, 1.42it/s, loss=0.0165, lr=1.7e-7]".

Figura 3.2: Procedimiento del entrenamiento mediante Dreambooth

Este archivo, en formato ckpt, se podrá utilizar en la aplicación NMKD SD GUI más adelante para generar imágenes, y contendrá el elemento entrenado bajo el token seleccionado. Si posteriormente se pretende incluir elementos al modelo ya entrenado, también se puede realizar empezando de nuevo el proceso de entrenamiento y utilizando de base el archivo en extensión ckpt anterior. Cuando se realice este segundo entrenamiento, se podrán generar imágenes acerca de ambos elementos, lo cuál es muy útil para nuestros objetivos, ya que en un mismo modelo enfocado a un paciente, debe haber múltiples elementos. Sin embargo, más adelante veremos que este último aspecto ha supuesto uno de los grandes fallos que experimenta el modelo en cuanto a múltiples capas de entrenamiento.

Un aspecto muy importante a tener en cuenta, es que tras la realización de múltiples pruebas, los resultados óptimos que hemos obtenido ha sido seleccionando un conjunto de datos formado por 10 imágenes, y con 2400 pasos de entrenamiento.

Hemos querido poner a prueba, no sólo las capacidades del modelo de entrenar a personas, ya que hemos podido comprobar sus puntos fuertes y débiles en la generación de seres humanos (los cuales podremos ver más adelante), sino de elementos que consideramos que también son de suma importancia a la hora de representar recuerdos: animales y lugares.

El objetivo de entrenar al modelo con lugares es que estos no se encuentren en la base de datos de imágenes de Stable Diffusion, puesto que de ser así, no tendría sentido realizar el entrenamiento. De este modo, podremos lograr que el paciente pueda rememorar sitios emblemáticos en su memoria y que las imágenes generadas que emulan recuerdos consigan ser aún más personales. Por ejemplo, el parque de su vecindario, la casa de sus padres o incluso, su propio salón.

Las primeras pruebas que realizamos sobre un lugar se trataba de un edificio característico, que por supuesto no estaba incluido en el modelo previamente. Hablamos de la basílica de Colmenar Viejo, Madrid. Las imágenes seleccionadas estaban hechas desde diferentes ángulos, alturas, luces y lejanías. En la figura 3.3 se pueden apreciar las características que reúnen las fotografías en cuestión y el token otorgado.

Además, quisimos comprobar si el número de imágenes y steps establecidos para las personas, producía resultados igual de satisfactorios para lugares. Y efectivamente, corroboramos la hipótesis de manera airosa. Ejemplos de ello, lo podemos ver en el apartado siguiente de Resultados.



Figura 3.3: Dataset seleccionado para el entrenamiento con lugares

3.2.4. Entrenamiento con la técnica de LORA

Para el entrenamiento de animales, seleccionamos 10 fotografías de un perro de la raza Shiba Inu bajo diferentes perspectivas, escenarios y mostrando distintas expresiones para comprobar si la inteligencia artificial permitía entrenar con animales. Sin embargo, a la luz de los resultados vistos en personas y lugares quisimos comprobar la técnica de LORA que presenta Stable Diffusion. Las siglas LORA hacen referencia a *Low-Rank Adaptation of Large Language Models*, del inglés. Esta técnica favorece un equilibrio entre el tamaño del archivo y la eficiencia del propio entrenamiento que ha presentado imágenes de gran calidad en un tiempo excepcional.

En la figura 3.4 se pueden ver más en detalle las imágenes seleccionadas para las pruebas con LORA en animales.

Como se puede apreciar en la figura anterior, se hizo hincapié en la diversidad de las imágenes, para aportar un mayor valor al entrenamiento, de modo que se tuviese una visión completa del elemento a entrenar.

Respecto al modo de ejecución, el hecho de cambiar a LORA no supuso grandes



Figura 3.4: Dataset seleccionado para el entrenamiento con animales

cambios a la hora de entrenar el elemento deseado debido a que la selección de imágenes es exactamente igual. Donde realmente cambia el entrenamiento es en el cuaderno utilizado en Google Colab, este es diferente y por lo tanto, la forma de ejecución y los pasos a seguir también lo son.

Es de vital importancia saber cuáles son los parámetros que se deben ajustar para poder desarrollar el modelo de manera correcta y entender cada uno de ellos, dado que pueden resultar un poco más complejos en comparación con la técnica del cuaderno de Fast-Dreambooth.

El cuaderno de LORA de Google Colab, a diferencia del de Fast-Dreambooth, contiene una única celda de código con varios parámetros a ajustar y que está ordenada en varias secciones. La primera sección es "Setup", lo que se puede traducir como la configuración o preparación, en ella se exige indicar el nombre del proyecto, la estructura de carpetas y por último, se debe especificar el modelo a entrenar entre tres opciones dadas (Anime, AnyLora y Stable Diffusion), y como se ha explicado anteriormente, hemos elegido el Stable Diffusion 1.5, al ser el que sirve de base para todos los entrenamientos escogidos. No obstante, esta técnica de entrenamiento tiene la peculiar característica de que se puede utilizar de base cualquier checkpoint desarrollado previamente, por lo que en caso de realizar un entrenamiento sobre personas, existe la posibilidad de elegir un modelo de base especializado en retratos. Esto garantiza que, seleccionando unas fotografías adecuadas y ajustando de manera correcta cada parámetro, los resultados sean bastante buenos.

La siguiente sección se llama "Processing", el parámetro más destacable en este apartado es la resolución de la imagen, a elegir entre 512, 640, 768, 896 y 1024. La elección dependerá del tamaño mínimo que tenga la imagen más pequeña de nuestro dataset. En particular, y habiendo realizado múltiples entrenamientos con LoRA, cabe destacar que lo más recomendable es dejar la resolución de la imagen en 512 píxeles, es decir, tal y como aparece. Esto es porque aumenta considerablemente el tiempo de entrenamiento cuanto más alta es la resolución. Si se diera el caso de que esto provocase un aumento significativo en la calidad de las imágenes generadas,

podría ser muy beneficioso. No obstante, entre una resolución de 512 y 1024 no se aprecia un gran salto de calidad, pero sí de tiempo, ya que aumenta desde los 20 minutos hasta más de una hora de tiempo de entrenamiento. Los demás parámetros son tags o etiquetas sobre la simetría y calidad de las imágenes.

Terminada esta, empieza la próxima sección "Steps". En cuanto a los steps de entrenamiento, en LoRA se deben indicar de un modo diferente, y aquí sí que presenta una gran ventaja respecto al modelo de Dreambooth. En este otro modelo, se indicaba un número concreto de pasos y como resultado, obteníamos un archivo en formato ckpt, de dos gigas de tamaño. Sin embargo, en LoRA se puede indicar un número de archivos que se deseen obtener. La peculiaridad de esto, es que cada número de pasos que se indique, se genera un resultado. De esta manera, en un mismo entrenamiento, se puede comprobar cuál es el número de pasos que genera la mejor fotografía. Este número de resultados se denomina epochs, y en num repeats, se puede indicar el número de pasos que se entrenarán en cada epoch.

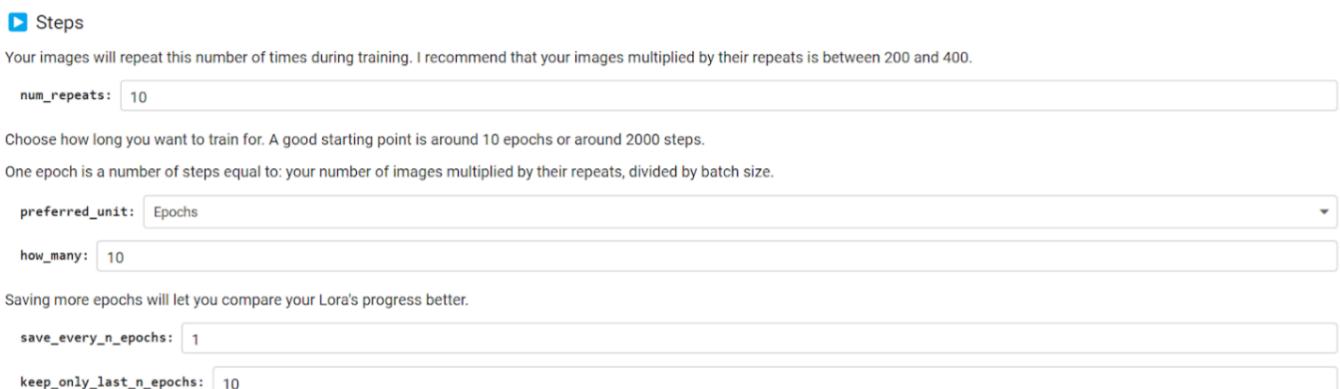


Figura 3.5: Parámetros relevantes del entrenamiento con LoRA

Por ejemplo, si se indica en este valor un 10, se obtendrá como resultado diez archivos en formato safetensor. Estos archivos tienen un tamaño de alrededor de 18 MB, un tamaño mucho menor que un checkpoint, de 2 GB. La diferencia reside en que el checkpoint es un modelo nuevo a partir de uno pre entrenado, y el safetensor, únicamente contiene el elemento que se ha entrenado, y para su ejecución necesita ir acompañado de su modelo de origen, en este caso el Stable Diffusion 1.5, lo que se ve reflejado en la siguiente imagen. De ahí nace la significativa diferencia de tamaño, por lo que, en cuanto a espacio de almacenamiento, es mejor la técnica de entrenamiento de LoRA.

Las secciones siguientes "Learning" de aprendizaje y "Structure" de estructura se refieren a variables que no se deben modificar del entrenamiento a cómo están preestablecidas, y por lo tanto, no hemos incidido en ellas.

Finalmente, la sección Ready nos indica que está ya todo listo para poder empezar el entrenamiento con LORA.

Ahora podemos decir que hemos logrado entrenar un modelo de generación de imágenes incluyendo fotografías propias, y eso es algo que puede ser realmente útil para nuestros siguientes propósitos. Esto es porque podemos lograr que cualquier persona pueda incorporar las imágenes que considere oportunas para servir de apoyo al paciente. Lo cual consideramos un éxito en el desarrollo de nuestro trabajo.

3.3. Interfaz de Stable Diffusion

3.3.1. Requisitos de instalación

Respecto a los requisitos que debe cumplir un equipo para que pueda funcionar la aplicación NMKD Stable Diffusion GUI, es fundamental que tenga un mínimo de 8 GB de memoria RAM, siendo lo más recomendable que sea de 16 GB. También es muy importante el almacenamiento, donde además de tener un mínimo de 10 GB de espacio disponible, es conveniente que haya 5 GB extras, debido a que se van a guardar multitud de archivos temporales a medida que se utiliza la aplicación. Adicionalmente, para nuestro caso, que hemos utilizado la aplicación para comparar diferentes modelos, es necesario conocer que cada uno de ellos tiene un tamaño de alrededor de 5 GB, y los modelos entrenados derivados de Stable Diffusion ocupan 2 GB de almacenamiento cada uno. Por lo tanto, el espacio es algo que hay que tener muy en cuenta previamente a la instalación de este programa, ya que puede comprometer seriamente el funcionamiento del equipo.

La característica principal que debe cumplir un equipo y sin la cual no sería válido para utilizar la aplicación es el hecho de tener una tarjeta gráfica. Además, no sirve cualquier GPU, dado que el mínimo de memoria VRAM que debe tener el equipo es 4 GB, siendo la tarjeta de Nvidia. Más allá de que este sea el mínimo para que la aplicación funcione, a medida de que la tarjeta gráfica sea de mayor calidad y espacio, el rendimiento mejora exponencialmente. Para contextualizar, la tarjeta gráfica de nuestro equipo, una Nvidia GeForce GTX 1050, con una memoria de vídeo dedicada de 3072 MB y una memoria virtual disponible de 8 GB, tarda alrededor de varios minutos en generar una fotografía con cierta calidad. Mientras tanto, tarjetas gráficas como la RTX 4090, genera las imágenes en 1 segundo, lo cual es bastante significativo. No obstante, esta tarjeta tiene un precio en el mercado de alrededor de 2000 euros, por lo que en nuestro nivel, mejorar el rendimiento es algo que se antoja complicado, y que la duración que va a tener nuestro proceso de generación de imágenes será siempre de varios minutos.

3.3.2. Funcionamiento y detalles de la interfaz

Respecto al funcionamiento de SDGUI, esta plataforma tiene una interfaz muy sencilla para el usuario, a pesar de que hay que tener conocimientos previos acerca de ciertos parámetros para poder llevar a cabo la generación de imágenes, además de realizar múltiples pruebas para saber qué función cumple cada elemento. La aplicación se presenta tal y como aparece en la siguiente imagen, y permite generar imágenes con modelos de Stable Diffusion, con modelos obtenidos de Hugging Face, con modelos entrenados y con LoRA. Esto ofrece una gran ventaja para nuestro trabajo, ya que ha sido el programa que nos ha permitido probar qué modelo era el más adecuado para nuestro estudio, y posteriormente testar los entrenamientos realizados. Ha sido fundamental, porque han sido miles de pruebas, ajustando todos los parámetros de múltiples maneras diferentes, y sin esta aplicación, avanzar y obtener resultados habría sido realmente complicado. Respecto a la estructura del programa, cuenta con una parte derecha de la pantalla, donde aparecen las imágenes generadas en un gran recuadro, se puede acceder a un historial de descripciones introducidas y se puede acceder a una carpeta que incluye todas las imágenes que han sido generadas por esta aplicación. Además, en la parte superior, se puede acceder a ajustes, donde es posible instalar nuevas versiones del programa u obtener nuevos modelos que poder utilizar para la generación de imágenes.

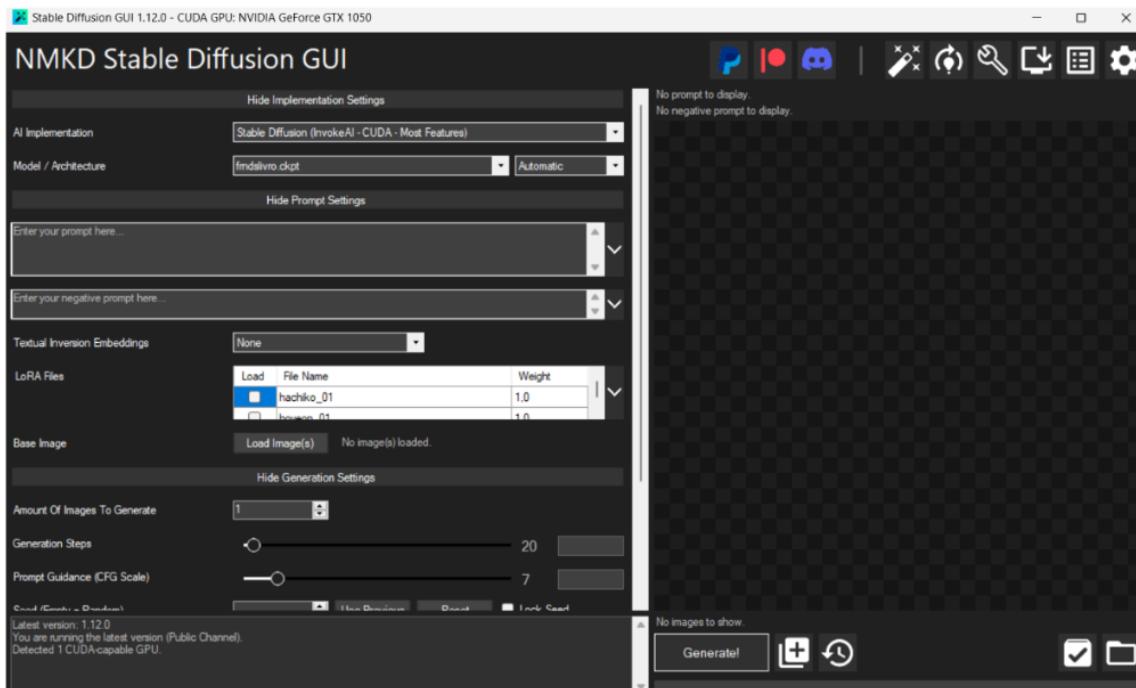


Figura 3.6: Aplicación SDGUI, utilizada para probar todos los modelos generadores de imágenes

En la parte izquierda de la pantalla, podemos encontrar los parámetros realmente importantes que condicionan la manera en la que se creará la siguiente fotografía. En primer lugar, el tipo de inteligencia artificial utilizada. Esto es realmente relevante

vante porque hace una distinción de cuatro tipos de IA generativa: dos son versiones estándar de Stable Diffusion (una para los sistemas que utilicen CUDA o Nvidia, y otra para los que utilicen AMD como tarjeta gráfica), la tercera es la versión XL de Stable Diffusion, que como recordatorio, requiere una tarjeta gráfica muy superior a la que dispone nuestro equipo, y por último, un modelo imagen a imagen. Resulta muy relevante hablar sobre esta IA, dado que a partir de una imagen de archivo, y una descripción, se puede generar una nueva imagen adaptada a esta petición. A pesar de que nuestro objeto de estudio abarca la inteligencia artificial generativa de imágenes a partir de texto, hemos realizado pruebas sobre el tipo de resultado obtenido, pero la calidad de la imagen de este modelo en concreto, resulta inferior a la que nos han otorgado modelos de texto a imagen. Centrándonos en el modelo básico de Stable Diffusion, en el apartado de model / architecture, se debe incluir el modelo, que puede ser uno pre entrenado o uno personalizado en formato checkpoint. En este segundo, cabe destacar que tarda medio minuto más en generar la fotografía debido a la carga, pero no afecta de ninguna manera a la calidad de la imagen.

A continuación, se debe indicar la descripción, que se divide en dos partes en este caso. La primera es el prompt, que guía a la inteligencia artificial para que incluya los elementos que se deseen. Un apunte relevante es que si se quiere hacer referencia a un elemento entrenado, se debe mencionar el nombre exacto del ítem para que se pueda realizar la generación de manera correcta. En segundo lugar, existe un negative prompt, donde se deben indicar aquellos elementos que, a petición del usuario, se desee que no se vean reflejados en la creación de la fotografía. Cuando se desee testar un entrenamiento de LoRA, se debe hacer click en el archivo deseado, e introducir ese nombre en el prompt. Por último, se deben ajustar unos parámetros explicados con anterioridad, como son los steps o pasos de generación, la escala CFG y la seed. Una vez conocidos estos pasos a seguir y cumplidos los requisitos mencionados, cualquier usuario debería tener la capacidad de generar imágenes mediante inteligencia artificial con este programa.

3.4. Resultados de generación de imágenes

Visto el funcionamiento de la aplicación, vamos a analizar los resultados obtenidos en comparación con los prompts especificados y bajo las posibles diferentes técnicas y parámetros que se han contemplado.

3.4.1. Resultados con personas

buenassssss

que tallll

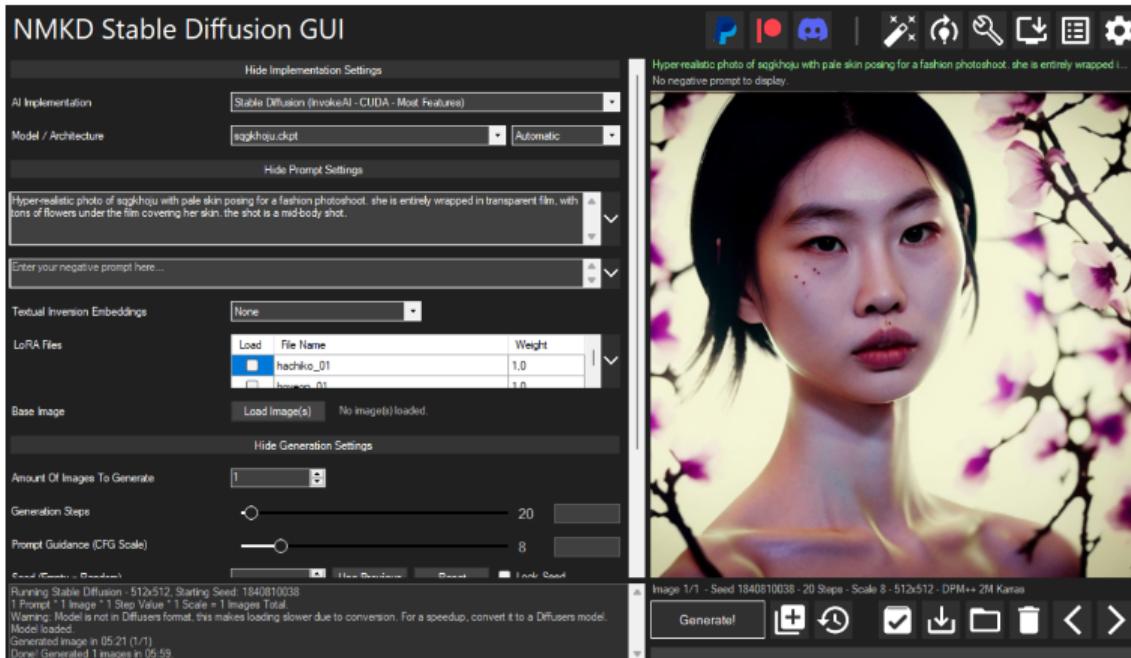


Figura 3.7: Resultados de entrenamiento de una persona con Dreambooth



Figura 3.8: Resultados de entrenamiento de una persona con diferentes estilos

3.4.2. Resultados con lugares

Con diferentes estilos:

Añadiendo otros elementos:



Figura 3.9: Resultados de imágenes del lugar entrenado al estilo Van Gogh y viñeta



Figura 3.10: Resultados de imágenes del lugar en invierno y con una persona

3.4.3. Resultados con animales

hellloouuuuu

heyyyyy

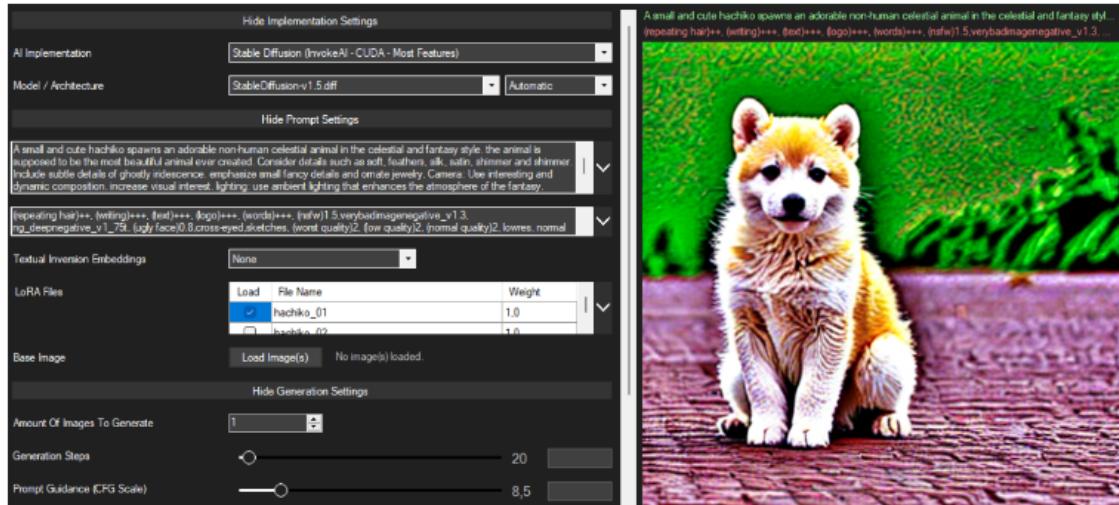


Figura 3.11: Imagen generada con el modelo Stable Diffusion 1.5, Lora Hachiko

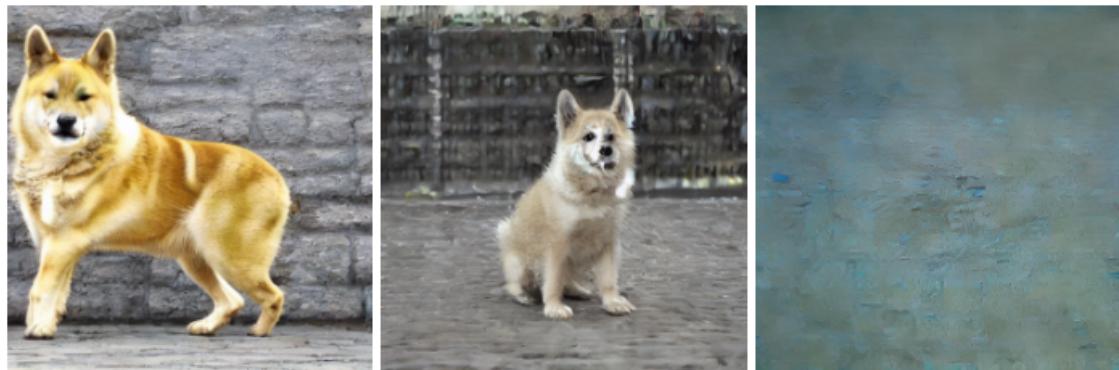


Figura 3.12: Mismo prompt con 1000, 2000 y 10.000 pasos de entrenamiento

3.4.4. Resultados con múltiples capas de entrenamiento

Una parte muy importante de nuestro trabajo es generar imágenes que puedan contener diferentes elementos de manera simultánea, puesto que consideramos que para crear un libro de vida de calidad, se deberían incluir fotografías de una persona en un determinado lugar, o que combinen varias personas diferentes. Durante la mayor parte del tiempo en la que realizamos este proyecto, teníamos la idea de que esto no sería posible, porque siempre que realizábamos pruebas, obteníamos resultados en los que se deformaban los elementos, o simplemente solo incluía uno de los dos elementos, o ninguno de los dos.

A pesar de esto, decidimos resistirnos a aceptar que no podía ser posible una generación de imágenes en la que se combinaseen varios elementos. Como hemos afirmado en apartados anteriores, nuestro modelo de entrenamiento Dreambooth permite realizar un entrenamiento sobre otro previo, por lo que decidimos hacer la prueba y añadir una capa. Para el entrenamiento, utilizamos como base el archivo sqgkhoju.ckpt, el cual entrenó a la actriz Jung Hoyeon, y se entrenó con el salón de

la serie Friends. Como en el resto de pruebas con las que obtuvimos un resultado satisfactorio, incluimos un conjunto de 10 imágenes y realizamos el entrenamiento con 2400 pasos.



Figura 3.13: Dataset del entrenamiento con el salón de Friends (frndslivro)

Una vez realizado el entrenamiento, en primer lugar, se comprobó que se podían generar buenas fotografías que incluyesen únicamente el salón, antes de crear imágenes con ambos elementos.



Figura 3.14: Resultados de la generación de imágenes con el modelo frndslivro

Como se puede apreciar, las imágenes generadas son de calidad, están creadas con 20 pasos de generación, 8 de CFG scale y con diferentes estilos. Viendo que el modelo genera las fotografías de manera exitosa, es el momento de crear imágenes de la persona en el salón. Siguiendo la tendencia de 20 pasos y 8 de CFG, se producían errores en la generación, creándose imágenes en las que aparecía la persona duplicada y no aparecía el lugar. Ateniéndonos a la definición de CFG scale, subimos este

valor, de 8 a 14, para que la inteligencia artificial se guiase en mayor medida por el prompt. Con solo subir este valor no fue suficiente, pues generaba fotografías en las que la persona aparecía entre 3 y 4 veces. Por tanto, también decidimos aumentar los pasos de generación, de manera sucesiva hasta que la generación produjera un resultado satisfactorio. La combinación perfecta fue de 40 pasos y 14,5 de CFG scale. Con un número menor de steps, la imagen aparecía con múltiples personas, y con un número mayor, no generaba ninguna persona. La descripción de la imagen era la siguiente: a sqgkhoju in the frndslivro, hd, high quality.



Figura 3.15: Mismo prompt con 20, 40 y 50 steps al mezclar dos elementos: persona y lugar

3.4.5. Logros y fallos en resultados

A lo largo de la variedad de entrenamientos realizados hemos hecho pruebas exhaustivas sobre la generación de las imágenes en comparación con los prompts especificados y hemos sacado en claro que hay 3 aspectos fundamentales que hay que tener en cuenta para la generación óptima de estas: el número de pasos realizados en el entrenamiento, en el número de pasos para la generación de la foto y la escala CFG.

En los tres aspectos, hemos identificado dos posibles situaciones que se pueden dar y que presentan fallos en cuanto a la calidad de los resultados: exceso y falta de pasos en el entrenamiento.

- Peligros con el sobreentrenamiento: Se produce cuando el modelo no se puede generalizar y se ajusta demasiado al conjunto de datos entrenados. Se debe principalmente a que el tamaño de los datos es demasiado pequeño y no contiene suficientes muestras de datos para poder representar con precisión la totalidad de datos de entrada posibles. Otra razón es que el modelo se entrena durante demasiado tiempo en un solo conjunto de datos de muestra. En nuestro modelo, encontramos sobreentrenamiento cuando elegimos un número de steps muy elevado para un número de fotografías que no es lo suficientemente alto. Una manera de detectar que nuestro modelo está sobreentrenado es cuando no genera bien la cara de la persona, y se

aprecian fallos en determinadas facciones, como en los ojos y la boca, en los cuales se aprecia deformidad.

- Falta de pasos en el entrenamiento: En este caso, se produce cuando el modelo de datos no tiene la capacidad de capturar de forma precisa la relación entre las variables de entrada y de salida, de manera que existe un elevado índice de errores en el conjunto de datos de entrenamiento y en los datos no vistos. Se debe a que el modelo es demasiado simple, porque el tamaño de los datos es demasiado pequeño, o bien porque se necesita más tiempo de entrenamiento. En este caso, cuando generamos las imágenes, se evidencia que el modelo aún no ha aprendido lo suficiente acerca del elemento o token del que se ha realizado el entrenamiento, pues el resultado de la generación refleja una persona que no muestra ningún parecido con la realidad.

En cuanto a mezcla de personas: Para ampliar los horizontes de nuestro entrenamiento, hemos entrenado a una persona, asociando un token a ella para su identificación, sobre un modelo que previamente ya había sido entrenado con una persona y su token asociado, para comprobar si efectivamente se podía generar en una sola imagen una representación de las dos personas entrenadas, una junto a la otra. Es aquí donde se ha detectado una peculiaridad, ya que, a pesar de que un modelo genere buenos resultados de cada una de las dos personas, en el mismo momento que se solicita en un determinado prompt o descripción que se vean reflejados una o varias personas entrenadas en la misma fotografía, ningún modelo de los que se haya probado, ha generado buenos resultados de esta manera. Lo que finalmente se aprecia en la imagen generada, es que aparecen dos personas pero sus rostros son una mezcla de las características faciales y fisiológicas de ambas, produciéndose una deformidad en muchos de los casos y que los rostros aparezcan prácticamente duplicados.



Figura 3.16: Duplicidad de elementos

3.4.6. Soluciones: Técnica Inpainting

Debido a la importancia que consideramos que tiene en el desarrollo de unas adecuadas historias de vida la mezcla de elementos personales con la propia persona, consideramos métodos para arreglar la poca certeza que nos otorga el modelo de Stable Diffusion con múltiples capas de entrenamiento, ya que a la luz de los resultados, el modelo es, por el momento, inválido en este aspecto.

Por ello, decidimos profundizar en las opciones que ofrece la interfaz de Stable Diffusion y dimos con la opción img2img, es decir, imagen a imagen. Esto es, como su nombre indica, un método en el que la entrada, en vez de ser texto como lo venía siendo hasta ahora, es una propia imagen. Sin embargo, el prompt sigue siendo un elemento fundamental que sigue formando parte del modelo, ya que de esta manera, se le especifica a la Inteligencia Artificial lo que se desea en la imagen futura y el modelo se encargará de crearlo, teniendo en cuenta la estructura y objetos ya dispuestos en la imagen original.

Incluso, el propio modelo de Stable Diffusion con el método de img2img tiene sus propias técnicas, de las cuales, una particularmente llamó nuestra atención: la técnica Inpainting.

Entre los problemas que se encontraban en el contenido final de las imágenes pedidas bajo un prompt que integraba dos elementos entrenados, podíamos distinguir 3 casos principales: contenido aleatorio no especificado, falta completa de uno de los dos elementos o un elemento y el concepto del segundo elemento sin llegar a ser el propio.

Por tanto, podemos llegar a afirmar que el último caso es el que más se asemejaba a lo que queríamos llegar a conseguir. Por ejemplo, si en el prompt especificábamos que apareciera el token que se asocia a una persona entrenada junto al token asociado a un animal entrenado, la gran mayoría de los resultados era el perro entrenado a la perfección junto a una persona que no se asemejaba de ninguna manera a la entrenada.

Lo que nos llegó a pensar que realmente la imagen generada de por sí, no era del todo errónea, ya que tan solo necesitábamos cambiar una pequeña porción de la misma: la persona de la imagen por la entrenada por nosotros en el modelo. Y es precisamente lo que logra la técnica de inpainting.

Esta novedosa técnica debe su nombre a que el protagonista es un pincel virtual. Para ejecutarla, necesitamos como entrada una imagen, en nuestro caso, la imagen recién generada por el modelo que presenta una carencia. Con el pincel, dibujamos el área de la imagen que deseamos sustuir por otro elemento. Y a través del prompt, especificamos aquello que solamente queremos que aparezca en el área trazada, en este caso, el token que representa a la persona entrenada.

De esta manera, la Inteligencia artificial genera lo especificado en el prompt solamente en el área indicada sustituyéndolo por lo que había anteriormente. En

este ejemplo, el área que dibujaríamos correspondería a la persona de la foto y el resultado final serían los dos elementos diferentes integrados a la perfección en una sola imagen.

Debido a la novedad de la técnica, los parámetros que estábamos acostumbrados a cambiar, también se han ampliado para poder emplear la técnica con los mejores resultados posibles y de la manera más eficiente.

Entre ellos, los más importantes y que se pueden ver en la figura 3.17 señalados, son 5, con los cuales dos ya estamos familiarizados, que son los teps y cfg scale. Por lo tanto, nos centraremos más en los 3 restantes.

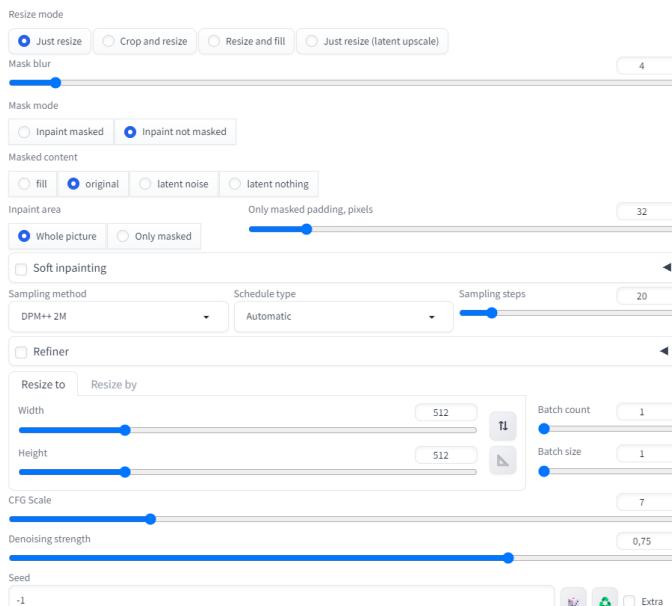


Figura 3.17: Configuración de los parámetros en la técnica de inpainting

El primer parámetro es "Mask Modez" antes de continuar, es importante especificar que en Inpainting, cuando se habla de "mask" se refiere únicamente al área que hemos marcado con el pincel para su posterior modificación. Como podemos ver en la figura 2.8, en este parámetro se encuentran dos opciones: inpaint masked e inpaint not masked. A lo que se refiere es al área que se va a ver sustituida por el prompt, ya que habrá casos en los que se quiera conservar la imagen en su totalidad a excepción de una parte y casos que por el contrario, se desearía modificar la imagen en su totalidad, a excepción de una porción, como lo puede ser en el caso de un cambio de fondo.

El segundo parámetro es "Masked contentz" las opciones que presenta son un poco más extensas que en el parámetro anterior, siendo estas fill, original, latent noise y latent nothing. Lo interesante de estas opciones es el funcionamiento que lleva a cabo Stable Diffusion en la generación de contenido en el área especificada. Como ya hemos podido ver en el Capítulo 2, Stable Diffusion necesita o bien ruido o bien nuevos datos para generar una imagen, y al tener la opción a original, se trabajará a partir del contenido de la foto original, es decir, se coge de referencia la

forma y colores originales para la generación. En general, se recomienda no cambiar estas opciones y siempre mantener la que está por defecto, es decir, original.

Por último, el tercer parámetro "Denoising Strength." es un valor entre 0 y 1 que explica la fuerza en la que cambiará el área seleccionada en comparación la imagen original, siendo el 0 que no cambie nada y el 1 algo completamente diferente.

Cabe destacar que la técnica de inpainting implementada a través del método de imagen a imagen no está incorporada en la aplicación NMKD SD GUI y por ello, se han tenido que buscar alternativas para acceder a ella. En concreto, hablamos de un cuaderno de trabajo hecho en Google Colab en el que se ha importado código de GitHub de Automatic1111, una conocida interfaz gráfica destinada a usar herramientas avanzadas de Stable Diffusion. La principal ventaja que presenta este cuaderno es que permite cargar un modelo propio ya entrenado, como se puede ver en la figura 3.18 que el modelo cargado es el archivo frndslivro.ckpt visto en este mismo capítulo anteriormente. Lo cual es perfecto para el objetivo que queremos conseguir.

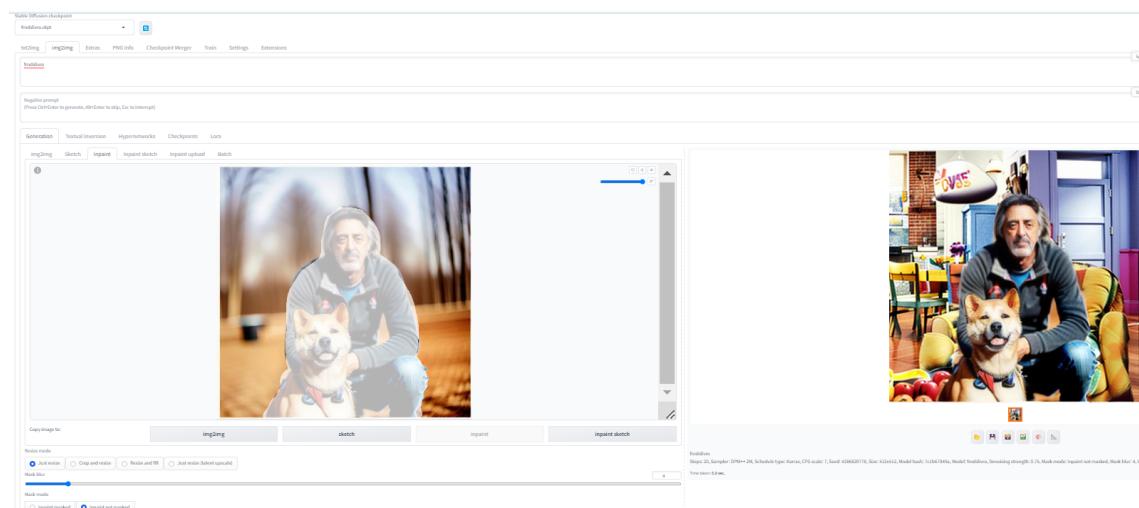


Figura 3.18: Ventana de la WEBUI implementada para realizar el painting

En la figura 3.18 se puede ver a la derecha la imagen ".original", la cual previamente ya había sido pasada por esta técnica para poner la cara de "80alp.en" la persona anteriormente generada. En la misma figura se puede ver el área masked seleccionada con el pincel y el parámetro "Mask mode." establecido a "Inpaint not masked". Finalmente, podemos ver en el prompt que solamente se especifica el token "frdslivro" debido a que lo que se quiere conseguir es que sustituya lo que no está seleccionado por la brocha en la imagen, por el salón entrenado del modelo. El resultado puede verse en la parte derecha de la imagen, el cual podemos afirmar que es todo un éxito, al haber conseguido incorporar 3 tokens entrenados diferentes, la persona, el animal y el lugar.

En el siguiente ejemplo de la figura 3.19 podemos ver la evolución que sigue una misma imagen bajo diferentes peticiones de prompt en la fase de inpainting. En la primera imagen que se ve en la figura, el prompt especificado fue ".a happy 80alp

walking with his dog hachift80alp in the forest next to the mountains, day light, full hd, high qualityz podemos ver como generó todo a la perfección excepto a la persona entrenada. En la segunda foto, se partió de la primera imagen y trazando el área del rostro de la persona, se puso como prompt ".^ close up picture of 80alp face, hd". Finalmente, para que el cuerpo de la persona generada al principio encajara con el de una persona de la edad de la foto, se seleccionó la ropa como área y finalmente el prompt fue .%old man clothesz el resultado final fue excepcional.



Figura 3.19: Evolución de una imagen con la técnica de inpainting aplicada

3.5. Evolución de los modelos del proyecto

Este apartado muestra los pasos que se han llevado a cabo durante la realización del proyecto. Resulta significativo porque el contraste de lo que se podía hacer al principio con lo que se puede hacer ahora es muy elevado, y pone en valor el trabajo realizado.

1. No se podían generar imágenes en local. No estaba instalado CUDA en el equipo, además de otras herramientas necesarias como Diffusers.
2. Generación de imágenes en local por consola en Anaconda mediante el modelo de Stable Diffusion 1.5. Resultado generado en un archivo. Ejecución lenta y de poca calidad.
3. Generación de imágenes en SDGUI. Buena interfaz que permite la ejecución de imágenes con Stable Diffusion 1.5. Mejor tiempo y calidad. Ya sabemos el modelo más óptimo para desarrollar nuestro proyecto.
4. Se descubre que se pueden utilizar más modelos provenientes de Hugging Face en SDGUI. Se utiliza el Lykon Dreamshaper, basado en LCM, que proporciona alta calidad con muchos menos pasos. No obstante, tiene la limitación de que no se puede personalizar y desarrollar.
5. Intento de generar imágenes personalizadas. Se probarán en SDGUI. Resultados aceptables con LORA. Se puede identificar al elemento entrenado, pero se puede mejorar.

6. Nuevas vías de entrenamiento de modelos. Entrenamiento de Stable Diffusion con Dreambooth. Se identifica el elemento pero se aprecia deformidad.

7. A base de prueba y error, se llega a la conclusión de que el mejor rendimiento se consigue con 2400 pasos de entrenamiento y un dataset de 10 imágenes. Buena calidad en las personas entrenadas, sin deformidad.

8. Se consiguen buenos resultados también en lugares y animales, además de en personas. Se llega a la conclusión de que se puede entrenar cualquier elemento.

9. Se quiere crear una aplicación que utilice esta IA generativa. Al principio se llega a una app con una interfaz poco visual, pero que genera buenas fotografías con el modelo Stable Diffusion 1.5.

10. Versión mejorada de la aplicación, con un backend en Python y un frontend en HTML. Buena interfaz y buenos resultados con modelos pre entrenados, pero no con modelos personalizados.

11. La aplicación ya puede utilizar cualquier modelo entrenado por nosotros mismos. Buenos resultados, con calidad y tiempo comparables a SDGUI.

12. Creación de un libro de vida para la aplicación. Sirve de conexión entre la IA generativa de imágenes desarrollada por nosotros y la creación de historias narrativas con apoyo visual para la terapia de reminiscencia.

Capítulo 4

Aplicación

4.1. Back-end

Una parte muy importante de nuestro trabajo es brindar una aplicación con la cual, cualquier usuario tenga la capacidad de utilizar nuestros modelos entrenados, es decir que pueda generar imágenes personalizadas de cualquier elemento que desee. El objetivo es aportar un valor añadido a nuestro trabajo, para que los destinatarios puedan, no solo obtener la mejor manera de crear imágenes personalizadas, sino que dispongan de una plataforma donde poder obtener todos los resultados deseados.

Para la creación de la aplicación optamos por realizar el código de diferentes formas, la primera de ellas siendo todo en lenguaje Python. Con esto se logró un buen resultado a nivel de funcionalidad, pues generaba las imágenes deseadas en el mismo tiempo en el que se generan en la aplicación SDGUI, previamente mencionada. No obstante, no logramos que la aplicación tuviese una interfaz atractiva e intuitiva para el usuario.

De este modo, decidimos utilizar una vía mejor para poder centrarnos por un lado en el estilo, y por otro lado en la funcionalidad. Para la lógica o funcionamiento de la generación de imágenes, realizamos un script en Python (app.py), lo que sería el backend del programa. Por otro lado, para el diseño y para solicitar al usuario los datos de entrada, creamos un script en html (index.py). En esta parte del código, se hizo un diseño de la interfaz de la aplicación, es decir, de qué manera queríamos que el usuario genere las imágenes y obtuviese los resultados. Como hemos mencionado con anterioridad, existen una serie de parámetros que son fundamentales para generar una imagen con inteligencia artificial, que son: la descripción (qué queremos que se muestre en la fotografía), el número de pasos o steps (iteraciones en el proceso de creación de la imagen) y el CFG Scale (la medida en la que la ia se debe guiar por la descripción introducida). Estos parámetros, deben ser introducidos por el usuario para la generación. En el método POST del script de Python, obtenemos

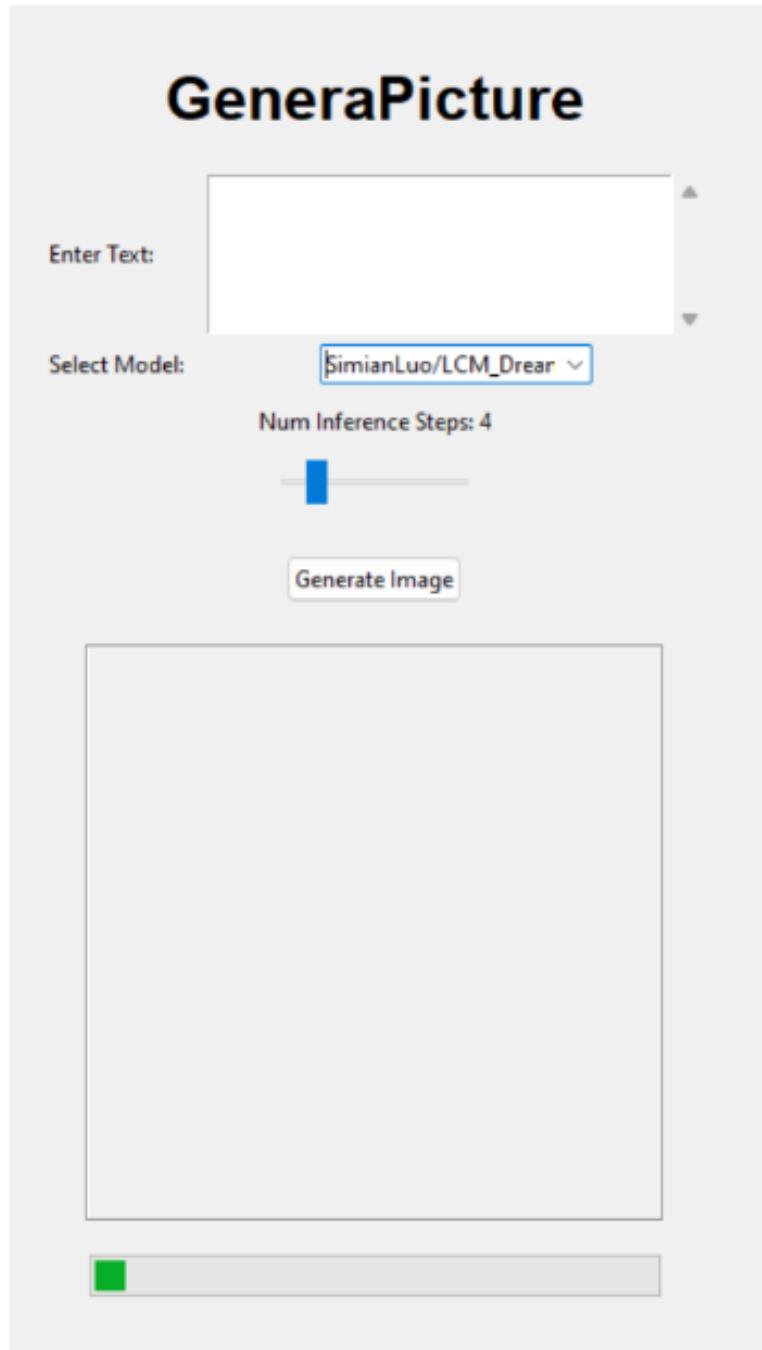


Figura 4.1: Prototipo de la aplicación en Python

las indicaciones introducidas por el usuario y se mandan al método Pipeline, que es el encargado de generar el modelo mediante el modelo cargado.

Esta aplicación ha tenido dos fases diferenciadas. En la primera, el modelo únicamente funcionaba con un modelo pre entrenado, es decir con un modelo instalado no personalizado, como el Stable Diffusion 1.5. Esto generaba buenos resultados, pero consideramos que no era lo suficientemente útil para poder ser utilizada en el

futuro para los libros de vida. Nos encontramos en un punto del proyecto en el que habíamos conseguido entrenar modelos y generar buenas imágenes personalizadas y en el que teníamos un prototipo de aplicación que funcionaba con modelos obtenidos de Hugging Face. Por tanto, el objetivo en esta fase era unir ambas partes que se habían conseguido, y desarrollar un programa con una interfaz atractiva, y con el que se puedan conseguir fotografías adaptadas a cualquier persona.

Este fue uno de nuestros mayores retos, puesto que no existía ninguna referencia de cómo utilizar un modelo entrenado para generar una imagen. Era algo particular y que supuso mucho esfuerzo. Para hacer funcionar un modelo de generación de imágenes en un dispositivo local, se necesitan dos librerías: diffusers y torch. La primera, es la que contiene los modelos de Stable Diffusion y derivados o entrenamientos del mismo. La segunda, es la que se encarga de enviar este modelo a la GPU del computador y hacer que se pueda iniciar la generación de la imagen si se encuentra disponible.

```
from diffusers import StableDiffusionPipeline
import torch

model_id = "runwayml/stable-diffusion-v1-5"
pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16)
pipe = pipe.to("cuda")

prompt = "a photo of an astronaut riding a horse on mars"
image = pipe(prompt).images[0]

image.save("astronaut_rides_horse.png")
```

Figura 4.2: Funcionamiento de modelos de Stable Diffusion

Diffusers contiene un método, llamado from pretrained, a través del cual, partiendo de un modelo de los que llamamos pre entrenados, crea una tubería con la lógica del funcionamiento del propio modelo para después ser ejecutado. Nuestro problema en ese momento, consistía en que este método from pretrained, únicamente aceptaba modelos de la plataforma Hugging Face, pues como parámetro se incluye el PATH de la Web donde se encuentra el modelo. Nosotros persistimos en incluir uno de nuestros modelos propios, pero la carga del archivo en formato .ckpt no era correcta con ninguno de los métodos utilizados, y no se creaba la tubería de manera satisfactoria.

No obstante, dimos con una idea que permitía utilizar cualquier modelo entrenado por nosotros mismos. Después de una exhaustiva búsqueda de diferentes bibliotecas de diffusers, encontramos una llamada StableDiffusionPipeline que permite realizar el proceso de crear la tubería, similar al que realiza el método from pretrained, pero en lugar de partir de un modelo pre entrenado, lo hace del enlace a un archivo concreto, que puede ser en formato .ckpt. Este método se llama from

single file y como indica su propio nombre en inglés, tan solo necesita un único archivo para hacer funcionar el modelo.

Sin embargo, para poder utilizar nuestro propio modelo, se necesitaba un enlace web, no una ruta de archivo. Por tanto, lo que decidimos finalmente fue subir nuestros archivos entrenados en formato .ckpt a Hugging Face, creando una nube personal que incluye todos los entrenamientos realizados. El proceso de subir el archivo es rápido, dependiendo de la conexión a internet, pero si esta es aceptable, en menos de un minuto ya se ha completado con éxito. Al probar la aplicación, la primera vez que se utiliza un modelo entrenado, se debe instalar este en el entorno en el que se está ejecutando. A partir de esto, se generan imágenes con completa normalidad, y de esta manera hemos logrado generar imágenes personalizadas en una aplicación propia.

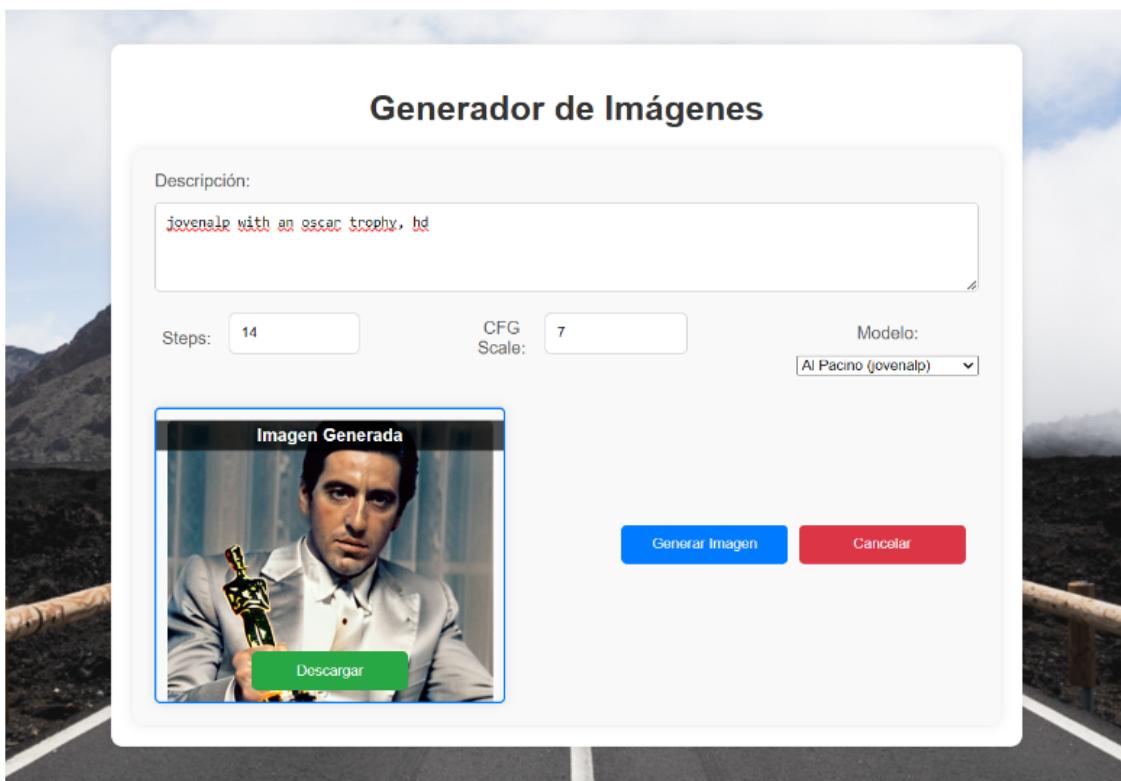


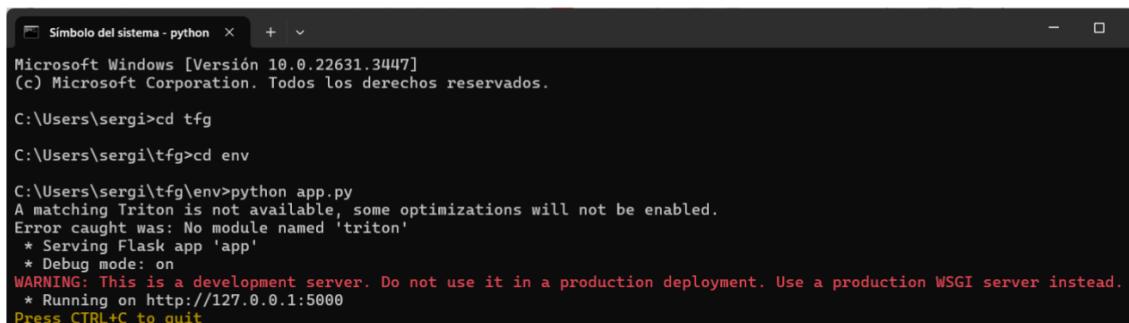
Figura 4.3: Aplicación generadora de imágenes

4.2. Ejecución de la aplicación

La manera de utilizar esta aplicación generadora de imágenes es muy sencilla, pero requiere de una explicación para que el usuario pueda hacer uso de ella. En primer lugar, cabe destacar que existe una serie de requisitos para poder utilizar este programa. El primero y más importante, es que, al igual que con la utilización

de cualquier modelo de Stable Diffusion en local, se requiere una tarjeta gráfica con un mínimo de 3 gigabytes de memoria virtual. Este es un requisito indispensable, sin el cual se podría acceder a la aplicación pero, en ningún caso, se podría generar ninguna fotografía. Para los usuarios que cuenten con la memoria virtual indicada, se debe contar con una serie de requisitos mínimos, que incluyen las versiones de determinados programas y herramientas necesarias para el funcionamiento de la aplicación.

Para saber cuáles eran estos requisitos, creamos un entorno virtual en el que incluimos todo lo relativo a la aplicación. Dentro de este entorno, se creó un archivo llamado dependencies.txt, con el objetivo de que los usuarios descarguen todo lo imprescindible ejecutando el siguiente comando en memoria: pip install -r dependencies.txt. Una vez instalado todo lo necesario, los usuarios deben seguir el proceso que se muestra en la siguiente imagen. En la consola, se debe acceder en primer lugar a la carpeta de nuestro trabajo, llamada tfg, y después al entorno virtual mencionado. Para arrancar el servidor, se debe ejecutar el comando “python app.py”, que como hemos explicado, es el script que recibe los parámetros introducidos por el usuario y los envía a la GPU para generar la imagen y posteriormente mostrarla. Cuando se ha iniciado el servidor, se notifica que está funcionando en el puerto localhost, al cual se puede acceder mediante el navegador, indicando “http://127.0.0.1:5000/”, según aparece indicado.



```
Símbolo del sistema - python + ▾
Microsoft Windows [Versión 10.0.22631.3447]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\sergi>cd tfg
C:\Users\sergi\tfg>cd env
C:\Users\sergi\tfg\env>python app.py
A matching Triton is not available, some optimizations will not be enabled.
Error caught was: No module named 'triton'
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Figura 4.4: Ejecución de la app

A continuación como se muestra en la figura 4.5 se abrirá nuestra aplicación, donde se ven reflejados dos botones. El primero, para acceder a un libro de vida, donde un hipotético usuario registrado, podrá consultar un libro formado por imágenes generadas por nuestra aplicación, que relatan de manera gráfica momentos únicos, personales y emocionales de la vida del paciente. Es una manera de mostrar los resultados de nuestro proyecto, y lo que cualquier usuario podría conseguir si utiliza nuestra aplicación.

El segundo botón accede a nuestra plataforma generadora de imágenes, es decir, lo que se muestra en la imagen anterior. En este caso, el usuario ya está listo para generar su imagen personalizada. Solo debe introducir la descripción, los pasos y la

escala CFG, además del modelo entrenado, que incluirá el o los elementos correspondientes de los que se quiere generar la fotografía. Si es la primera vez que se utiliza un determinado modelo, al pulsar en generar imagen se descarga en el entorno, lo cual supone un tiempo de menos de 1 minuto con una conexión a internet aceptable. A continuación, se genera la imagen con una duración dependiente del número de pasos introducidos y de la tarjeta gráfica de cada usuario. En este caso, en nuestro equipo, la generación tiene una duración de entre 10 y 16 segundos por paso introducido, lo que supone un tiempo de unos 3 minutos en una imagen de 15 pasos, que nos aporta una calidad aceptable. Una vez generada la imagen aparecerá reflejada en la aplicación, y el usuario tendrá la posibilidad de descargarla con el nombre que desee.



Figura 4.5: Portada de la aplicación

Capítulo 5

Libro de vida

El arduo trabajo llevado a cabo en los capítulos anteriores nos ha servido para saber los recursos con los que contamos, pero también con sus limitaciones. Con la experiencia en el entrenamiento de imágenes, decidimos realizar un entrenamiento intensivo sobre una persona para memorar su paso de los años, y de esta manera poder construir el libro de vida personalizado y compuesto exclusivamente con sus recuerdos. Para ello, tomamos como referencia a una persona que tuviera una amplitud de fotografías desde una edad temprana hasta una más avanzada. La persona en cuestión es el actor Alfredo James Pacino, un actor estadounidense de 84 años.

En este punto, vamos a realizar una comprobación de cómo funciona el modelo entrenado de Stable Diffusion con personas en distintas fases de su vida. Lo que queremos testar es que responda bien en todas y cada una de las circunstancias, debido a que para la creación de historias de vida, necesitamos que la persona pueda incorporar imágenes de hechos significativos, que puedan evocar momentos especiales. Para eso se requiere que el modelo funcione bien en todas las etapas, para que la experiencia del usuario pueda ser lo más completa posible.

Vamos a crear 3 entrenamientos diferentes: uno de la persona en su juventud, otro de la persona en la edad adulta y otro de la persona en su edad anciana. Resaltamos que el hecho de que la persona esté incorporada en Stable Diffusion es indiferente, porque para realizar la prueba se va a nombrar de una manera distinta. Es un nombre que asignaremos que será único, con el fin de que el modelo identifique al elemento entrenado y pueda generar las imágenes personalizadas deseadas. Claro está que en las tres fases se han otorgado diferentes identificadores para que el modelo sepa a cuál de ellas nos estamos refiriendo. Una vez tenemos las 3 capas del modelo entrenadas, decidimos generar 3 imágenes bajo el mismo prompt y parámetros para comprobar la exactitud del modelo. En la figura 5.1 podemos apreciar que los resultados son precisos y con una calidad excepcional.

El prompt es cuestión fue "Close up studio photo of jovenalp/adultalp/80alp, detail, studio lighting."

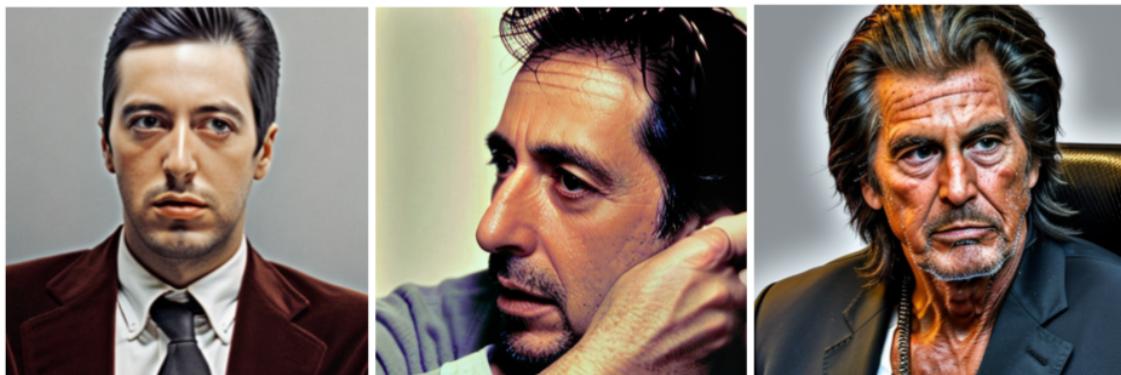


Figura 5.1: Resultados obtenidos bajo un mismo prompt en las 3 fases

El objetivo principal de esta tarea es simular una generación de imágenes para un libro de vida de un paciente. Por lo tanto, no solo necesitamos generar fotografías de la propia persona, si no con diversos elementos para cada etapa, para comprobar cómo puede funcionar en la terapia de reminiscencia, cuando se incluyan imágenes personales del paciente, de lugares, personas y otros elementos importantes de su vida.

Para la elaboración del libro de vida se necesita una cantidad considerable de recuerdos y momentos entrañables que conforman la vida de una persona en diferentes etapas y épocas. Nosotros quisimos elaborar un ejemplo de lo que podría ser el libro de vida de un paciente, sin embargo, al no poder contar de primera mano con una persona que padeciera esta enfermedad que nos pudiera relatar estos recuerdos, decidimos construir un relato de una persona ficticia a modo de exemplificar el resultado final de la generación de un libro de vida completo generado a partir de Inteligencia Artificial generativa, de modo que representara el trabajo que hemos realizado durante todo el proyecto.

En cuestión, la persona del ejemplo se trata de Juan Pacheco, un hombre nacido en Madrid en 1944 y que actualmente tiene 80 años. Para contar su historia, hemos decidido proporcionar detalles que explican su personalidad, entorno y el rumbo que tomó su vida con el paso de los años, como sus intereses, estudios, trabajo y lugar de residencia. Para asegurar una continuidad temporal y poder ubicar los acontecimientos vividos por Juan en una línea de tiempo, hemos diferenciado 3 categorías, como ya se ha mencionado antes.

5.1. Fase de la vida 1: Juventud

Juan nació en Madrid, en 1944 y sus padres residían en una localidad situada al norte de la comunidad madrileña, llamada Colmenar Viejo. Juan fue hijo único y sus padres eran Ana, una profesora de Química en la universidad Complutense de Madrid y Alfredo, un agente inmobiliario. Juan creció en Colmenar y de pequeño presentó un gran interés por el arte, más en concreto por la pintura. Era un joven

alegre que le gustaba pasar el rato con sus amigos y jugar con ellos en el punto céntrico del pueblo donde se encontraba la Basílica de Colmenar Viejo, su lugar favorito. Además, Juan siempre fue un amante del deporte y era partidario de llevar hábitos saludables, fue entonces cuando empezó a correr y su ambición le llevó a apuntarse a un equipo de atletismo, lo cual disfrutaba bastante practicando. Finalmente, su madre hizo que se interesara en su campo de estudio y siguiendo sus pasos, Juan decidió especializarse en ello estudiando la carrera de Química, al igual que ella, en la Universidad Complutense de Madrid.

Cabe destacar que el token otorgado para la primera etapa responde bajo el nombre de "jovenalp" es el que por tanto, usamos en todos los prompts de esta primera fase. En la figura, se muestra el dataset seleccionado para representar la etapa inicial de la vida de Juan, quien es representado en la vida real por el actor de cine anteriormente mencionado.



Figura 5.2: Dataset seleccionado para el entrenamiento con Juan de joven

Tal y como lo hemos dispuesto, el capítulo 1 del libro de vida está compuesto por una serie de recuerdos que representan los acontecimientos más célebres para Juan en sus primeros años. En la tabla 5.1, especificamos los prompts seleccionados a modo de descripción de sus recuerdos con sus respectivas imágenes como resultado.

5.2. Fase de la vida 2: Aduldez

Continuando con la vida de Juan en su etapa adulta, consiguió un trabajo como químico en un laboratorio de investigación en la capital. Por ello, decidió mudarse a la capital ya que siempre le gustó la vida en la ciudad. Finalmente, terminó comprándose un piso en el Paseo de la Castellana, concretamente, a 7 minutos andando de las Torres Kio, un elemento de la arquitectura madrileña que le gustaba especialmente. A pesar de que sus años en el equipo de atletismo llegaron a su fin, Juan sigue disfrutando del deporte, aunque de una manera menos intensa y aprovecha sus tiempos libres y de ocio para hacer senderismo por la montaña y poder estar

Recuerdo	Prompt	Resultado final
Recuerdo de Juan de niño y su madre por primera vez en su laboratorio personal	Jovenalp as a child with his mother in a chemical laboratory with a white coat, full detail, high quality, hd	
Autorretrato que se hizo Juan de sí mismo	A dumb and sad jovenalp, with sad face. posing 7/11, wearing a leather trench. greyscale style, realistic, pencil sketch	
La basílica del pueblo donde creció Juan de niño, Colmenar Viejo en navidad	a picture of cvbasil with snow and christmas decoration, high quality, hd	
Juan en el jardín de su casa vestido de traje para su primera entrevista de trabajo	jovenalp gentleman in light brown pinstripe double breasted suit, standing in a park	

Tabla 5.1: Tabla de resultados obtenidos del Capítulo 1 de juventud del paciente

en contacto con la naturaleza. Además, su afición a la pintura nunca la perdió e incluso, fue mejorando su técnica.

En esta etapa, y para que el libro de vida cobrará más sentido, decimos añadir elementos nuevos para representar los cambios que ocurren en la vida de una persona al cabo de los años. En concreto, hemos entrenado el modelo con las torres Kio de Madrid.

En específico, el token que se le otorgó a esta etapa se llama `.adultalp` el dataset seleccionado son 10 fotos comprendidas entre los 40 y 55 años del actor. En la figura 2.8 se pueden apreciar la variedad de las mismas.



Figura 5.3: Dataset seleccionado para el entrenamiento con Juan de adulto

Por otro lado, en la tabla 5.1 están recogidos los resultados de los recuerdos que representamos bajo este nombre y en esta etapa en concreto.

5.3. Fase de la vida 3: Vejez

Tras 36 años largos de profesión, Juan decide jubilarse. Debido a su soledad y el deseo que siempre le acompañó de adoptar un fiel compañero, decidió adoptar un perro para hacer sus días más entretenidos y que le pudiese acompañar siempre. Debido a su tiempo libre, Juan se aficionó a la serie televisiva Friends y sus familiares le ayudaron a redecorar su casa con la temática que sigue el salón que aparecía en la serie. Entre otras aficiones, Juan tomó la iniciativa de realizar viajes con el imerso y poder descubrir lugares que no conocía. En sus últimos años, Juan fue optando por actividades algo más sedentarias como leer y jugar al ajedrez hasta que el Alzheimer estuvo tan avanzado que necesitaba estar en un centro atendido por profesionales. En el centro empezó a realizar terapias de reminiscencia junto a su familia y cuidadores y de esta manera, terminó cobrando forma este libro de vida.

El último conjunto de datos que escogimos comprendía las edades entre los 70 y 84 del actor, más comúnmente conocido como Al Pacino. En la figura 2.7 se puede

Recuerdo	Prompt	Resultado final
Recuerdo de Juan de niño y su madre por primera vez en su laboratorio personal	Jovenalp as a child with his mother in a chemical laboratory with a white coat, full detail, high quality, hd	
Autoretrato que se hizo Juan de sí mismo en su edad adulta		
Juan muy contento haciendo senderismo en la montaña		
Juan en el laboratorio de su trabajo	serious adultalp in a chemistry laboratory, with a white coat, detailed face, hd	

Tabla 5.2: Tabla de resultados obtenidos del Capítulo 2 de adulteza del paciente

apreciar más detalladamente las características de las imágenes elegidas. Finalmente, el token asignado fue "80alp".



Figura 5.4: Dataset seleccionado para el entrenamiento con Juan de anciano

En la tabla 5.3 podemos apreciar los resultados obtenidos a partir de los últimos recuerdos de Juan, disfrutando de las cosas cotidianas que hace en su día a día a su vez que realizando ejercicios que le fortalecen mentalmente, y por supuesto, en los que se incluyen momentos significativos para él esta etapa.

Esto muestra cómo responde nuestro modelo a entrenar a una misma persona durante diferentes fases de su vida. Resulta muy útil dados nuestros objetivos iniciales del proyecto, dado que son imágenes que se pueden generar con los elementos que se desee para un libro de vida. Al igual que para el resto de pruebas realizadas, únicamente son necesarias 10 fotografías y se puede apreciar que los resultados son muy buenos y detallados, y lo más importante, se demuestra que el modelo es capaz de generar imágenes completamente diferentes a las que ya existía previamente sobre la persona. Esto es vital, porque de no ser así, el modelo no sería necesario, porque no aportaría ningún valor añadido.

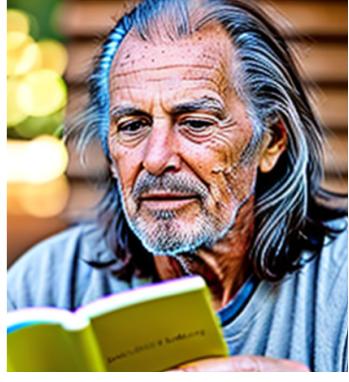
Recuerdo	Prompt	Resultado final
Juan disfrutando de una partida de ajedrez	serious 80alp playing chess, detailed face, long hair, old, hd	
Juan leyendo su libro favorito en la residencia	serious 80alp reading a book, detailed face, long hair, old, hd	
		Imagenes/Vectorial/.png
		Imagenes/Vectorial/.png

Tabla 5.3: Tabla de resultados obtenidos del Capítulo 3 de vejez del paciente

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

En primer lugar, podemos afirmar que, mediante la ayuda de la inteligencia artificial, se puede brindar un material fotográfico personalizado que sirva para colaborar con terapias ocupacionales, de manera que un paciente tenga la posibilidad de evocar momentos significativos, que de otra manera no tendría la capacidad de visualizar.

Con respecto a la generación de imágenes, se ha conseguido el objetivo de entrenar un modelo de inteligencia artificial de convertir texto a imagen, incluyendo a la persona que se considere. Este hecho significa que, en una terapia de reminiscencia, si se nos otorga un número considerable de imágenes en las que aparezca una persona concreta (a partir de diez), se puede trabajar en base a un modelo entrenado que reconozca a esa persona, y generar a partir de él las imágenes deseadas.

Adicionalmente, el modelo que se ha obtenido, puede servir de base para realizar un nuevo entrenamiento, lo cuál encaja a la perfección con nuestros objetivos, puesto que para un determinado paciente, podemos desarrollar un gran modelo que incluya el número de personas que se desee. No obstante, Stable Diffusion y sus modelos derivados, tienen una limitación, dado que una misma imagen es difícil de generar con múltiples elementos. Hemos logrado resultados aceptables en algunas ocasiones con un número considerable de pasos, pero requiere mucha paciencia y no podemos garantizar el éxito absoluto, como sí podemos hacerlo en una generación con un elemento principal.

Otro aspecto importante que debemos tener en cuenta y al que hemos llegado a la conclusión después de toda la investigación, es que para hacer funcionar nuestro modelo, al igual que cualquier otro de inteligencia artificial generativa de imágenes,

se necesita una gran capacidad de memoria gráfica. Con la GPU de nuestro equipo, la generación siempre va a abarcar unos minutos, y cuanta más calidad se desee, más tiempo se incrementará. Esto es un hecho que todas las personas que se presten al servicio de generar las fotografías personalizadas deben conocer.

Respecto al entrenamiento de los modelos, podemos concluir que el número de pasos con el que se realiza correctamente es 2400 para un entrenamiento de 10 imágenes, puesto que se han realizado múltiples pruebas con más pasos, en los que se detecta sobreentrenamiento, y con menos pasos, en los que se detecta falta de entrenamiento.

En base a los resultados obtenidos, podemos afirmar que el entrenamiento que maximiza la calidad de las imágenes es el de Dreambooth, con una calidad superior a la conseguida con LoRA. Con este último modelo, era mucho más difícil obtener imágenes personalizadas a partir del elemento entrenado, puesto que dadas sus características, se centraba en mostrar únicamente el elemento, ignorando los complementos que se pedían en el prompt. A través de Dreambooth, las imágenes mostraban una calidad superior y sí que se veía reflejado todo lo que se pedía en la descripción de la fotografía. Con ambos modelos de entrenamiento, podemos concluir que se puede entrenar todo tipo de elementos, ya sean personas, animales o lugares, ya que con todos ellos se han obtenido resultados correctos.

Con respecto a la aplicación, hemos conseguido realizar un programa, mediante el cual, se puedan generar fotografías personalizadas a partir de nuestros modelos entrenados. Esto produce muy buenos resultados y, además, permite la visualización de un libro de vida basado en una historia creada por nosotros. La idea es que un usuario pueda contar con un programa con el que pueda generar imágenes personalizadas y desarrollar un libro de vida muy atractivo de manera sencilla.

Podemos concluir con que hemos cumplido con los objetivos que teníamos previamente al comienzo este proyecto, puesto que hemos construido un programa mediante el cual un usuario podría obtener imágenes personalizadas con una calidad razonable en un tiempo adecuado. Ha habido una evolución constante en los últimos meses en los que se pasó de no poder generar imágenes localmente, a poder generarlas con modelos entrenados, para después poder mostrar esas fotografías en una aplicación desarrollada por nosotros.

6.2. Propuestas de mejora

Tras la realización de este proyecto, nos gustaría que personas encargadas de asistir a pacientes con pérdida de memoria, pudiesen realizar una investigación partiendo

de nuestro modelo, con el objetivo de estudiar psicológicamente en qué medida se está favoreciendo la reducción del estrés, se está potenciando la capacidad cognitiva del paciente y se está logrando la satisfacción de las personas. Este es el mayor objetivo que tenemos con esta inteligencia artificial, lograr el bienestar personal, y el alcance de este trabajo no nos permite comprobar si realmente hemos obtenido grandes resultados en el aspecto social.

Adicionalmente, sería conveniente que si este proyecto fuese utilizado por los terapeutas, deberían tener un equipo con una gran tarjeta gráfica, para que se invirtiera así el menor tiempo posible en la generación de las imágenes. Tras haber utilizado este mismo modelo en la nube, donde utilizan servidores con GPUs de gran potencia, la obtención de resultados era instantánea, por lo que destacamos un gran margen de mejora que en caso de reducirse, agilizará en gran medida la terapia. Además, cabe resaltar que en nuestro modelo, hemos empleado la versión 1.5 de Stable Diffusion porque es la única que podía funcionar en local con la tarjeta gráfica de nuestro equipo. Con una mejora en este aspecto, se podría trabajar y entrenar la versión XL, que aporta una gran calidad a las imágenes, y se lograrían unos resultados aún mejores minimizando el tiempo de espera. Además, Stable Diffusion recientemente ha lanzado una nueva versión 3 que asegura rapidez, eficiencia y calidad en las imágenes generadas sin necesidad de utilizar tanto espacio como en la versión XL, por lo que sería una opción muy atractiva con la que trabajar. Incluso, podría llevarse el trabajo de los libros de vida un paso más allá

En base a haber realizado una aplicación que ponga a prueba los modelos entrenados y que muestre un libro de vida con imágenes personalizadas, nos gustaría implementar un servidor de manera que el rendimiento dependa exclusivamente de él y de la conexión a internet, y no del equipo de cada usuario y su tarjeta gráfica. De esta manera, se eliminarían los requisitos fuertes de utilización. Además, sería conveniente seguir desarrollando este programa, permitiendo que los usuarios se registren y sean ellos quienes administren el libro de vida de los pacientes de la terapia de reminiscencia. Para llevar a cabo esta propuesta, se tendría que conectar el proyecto a una base de datos, que contenga el registro de usuarios, y dentro de cada uno de ellos, una tabla con el libro de vida del paciente, que debería tener como atributos, principalmente, una imagen, un título y una descripción. Creemos que puede ser un formato atractivo para la terapia, donde los usuarios pueden personalizar de manera muy sencilla un libro de vida, con imágenes reales, o bien creadas por inteligencia artificial, en el caso de que no existan documentos gráficos de un hecho relevante, significativo y emocional para la vida del paciente.

Introduction

Introduction to the subject area. This chapter contains the translation of Chapter 1.

Conclusions and Future Work

Conclusions and future lines of work. This chapter contains the translation of Chapter 6.

Contribuciones Personales

Estudiante 1: Sergio Llorente Hernando

Lo primero de todo fue reflexionar sobre los objetivos del proyecto, de manera que realicé una investigación sobre la terapia ocupacional, los libros de vida y las distintas inteligencias artificiales, con el objetivo de saber qué tecnología debemos utilizar para satisfacer las necesidades de los usuarios.

Desde un primer momento hice una labor de búsqueda y prueba de diferentes modelos de inteligencia artificial generativa de imágenes. En primer lugar, la prueba era con servidores especializados y más adelante, hice una descarga de los modelos que generaban imágenes de calidad, con el objetivo de comprobar cuáles de los modelos podrían funcionar mejor y otorgar unos resultados satisfactorios.

El siguiente paso en el que me centré fue en conseguir hacer funcionar la inteligencia artificial generativa en local. Para ello, en primer lugar opté por probar los modelos en anaconda, lo cual no era una interfaz cómoda para el usuario y la generación de imágenes era lenta. A continuación empecé a trabajar con SDGUI, una aplicación con la que podía testar cualquier tipo de modelo, ya sea entrenado o no, y además de eso, ajustar todos los parámetros necesarios. Con esta aplicación sí que se generaban bien las imágenes. Esto sirvió para diseñar nuestro propio programa que incluyera el modelo generativo.

Para crear la aplicación, primero opté por crear un script de Python. Esto dio buenos resultados, porque mostraba la imagen al igual que en el programa de referencia, pero la interfaz no era buena para el usuario, y además haciendo pruebas en cualquier otro ordenador no obtuve resultados. Este hecho hizo que creara un entorno virtual para saber las dependencias que son necesarias para que funcione el modelo, y por otro lado, dividir el back y el front, con el objetivo de hacer más atractiva la aplicación mediante html, que es un lenguaje con el que ya estaba más familiarizado. Esto dio un mejor resultado, y se consiguió generar imágenes en una aplicación con una buena interfaz y un buen diseño.

La siguiente fase del trabajo, fue la del entrenamiento de los modelos, con el objetivo de tener la capacidad de personalizar la inteligencia artificial a petición de cualquier usuario. Fue un trabajo muy complejo en el que hice muchísimas pruebas con diferentes vías, con el objetivo de obtener los mejores resultados posibles. Fueron múltiples pruebas debido a que, para ofrecer al usuario la experiencia óptima, necesitamos conocer cuántas imágenes y cuántos pasos de aprendizaje se requieren. Tras la realización de estas pruebas se concluyó que la estrategia más óptima era utilizar un conjunto de 10 imágenes y 2400 pasos de entrenamiento. No obstante, hicieron falta múltiples entrenamientos erróneos o parcialmente correctos para conocer este hecho.

La siguiente fase fue realizar más entrenamientos con diferentes elementos y analizar los resultados. Además, aquí opté por examinar si un resultado de un entrenamiento podía servir de base para un próximo. La respuesta fue afirmativa y eso me llevó a añadir múltiples capas a un mismo modelo, lo cual podría ayudar a un usuario a tener un solo modelo personalizado que incluya todos los elementos que desee.

Por último, traté de lograr que la aplicación generase imágenes mediante modelos entrenados por nosotros mismos. Fue una fase complicada, pero finalmente se logró un resultado satisfactorio. Además, al entorno de la aplicación le añadí una página que representa un ejemplo de libro de vida desarrollado por nosotros, con unas imágenes generadas por nuestra aplicación. De esta manera, tenemos una aplicación con dos páginas. Una que muestra el tipo de libros de vida que podríamos desarrollar, y una que es un entorno de generación de imágenes con múltiples modelos y de diferentes maneras.

Estudiante 2: Isabella Romano Ramos

Lo primero que realicé fue la introducción del tema de estudio sobre el trabajo después de la primera sesión de reunión con el tutor en la que quedaron claros los aspectos generales que se iban a abordar, los objetivos principales y las tecnologías que se iban a utilizar.

Una vez definido esto, empezó la fase de investigación sobre los temas principales que abordaban el proyecto: el Alzheimer y la Inteligencia Artificial. Me embarqué en un proceso de recopilación de información para entender lo que ocurre en el cerebro que hace que se deteriore la memoria y que cause esta enfermedad que afecta a millones de personas, y así poder reflejarlo en la memoria. En cuanto a la IA, era un concepto totalmente nuevo para mí ya que no había visto nada sobre este campo en ninguna asignatura de la carrera. Por lo que opté empezar a entenderla por el principio, desde un concepto tan simple como el perceptrón, hasta examinar

las redes neuronales, pasando por todos los tipos de modelos fundamentales qué existen y saber diferenciar la función de cada uno. Por ende, me encargué de buscar la bibliografía que hablaba de la IA, más en concreto sobre el Deep Learning y las redes neuronales. De esta manera, lo pude redactar de manera que se entendiera lo mejor posible en el Capítulo 2 Estado de la cuestión.

Como se ha podido ver a lo largo de todo el proyecto, nos hemos centrado en el campo concreto de la Inteligencia Artificial generativa de imágenes y lo que hicé fue investigar las diferentes alternativas de modelos ya implementados que pudiéramos utilizar para el entrenamiento de personas y la integración en nuestra futura aplicación. Tras examinar profundamente las prestaciones y disponibilidad de cada uno, entre los tres candidatos finales, que se trataban de Midjourney, Dall-e y Stable Diffusion, se eligió usar el último por las limitaciones que nos mostraban los otros dos modelos. Una vez elegido, hice un análisis profundo de cómo está implementado el modelo de difusión estable que utiliza la herramienta y lo redacté extensamente en el Capítulo 2 Estado de la cuestión. Además, durante el proceso surgieron conceptos interesantes en relación con Stable Diffusion, como LORA y Dreambooth e incluso, consideré importante añadir la importancia de la ingeniería del prompt para la creación de imágenes a partir de texto.

Cuando pude comprender los conceptos teóricos que explicaban el funcionamiento de Stable Diffusion, se pasó a la parte práctica, es decir, a pensar en la estructura y usos de la aplicación que se iba a desarrollar como apoyo al modelo de IA generativa seleccionado. Finalmente, caímos en la cuenta de que las prestaciones que ofrecen mis dos equipos, tanto mi portátil como el ordenador de sobremesa, no cumplen los requisitos mínimos para poder ejecutar la aplicación en local ni realizar las pruebas necesarias sobre la generación de imágenes, al no contar con GPU en el portátil y no ser lo suficientemente potente la GPU en el ordenador de sobremesa. Es decir, era inviable que se pudieran generar imágenes en mis equipos.

Por ello, solo me podía limitar a entrenar el modelo que seleccionamos a través de la herramienta de Google Colab. Fue entonces cuando busqué en Hugging Face el modelo de Stable Diffusion más adecuado, que resultó ser el 1.5, y fui probando el número de pasos y de imágenes que producían los mejores resultados.

Para solucionar los problemas presentados en el entrenamiento multicapa, investigué la manera de solventar los errores de generación respecto a una cantidad de tokens mayor a 1 y di con la técnica de inpainting. Pude profundizar en esta técnica y aplicarla a todas las imágenes que presentaban fallos para poder perfeccionarlas y que pudieran ser añadidas en el libro de vida, dándole más sentido y personalidad al mismo.

Una vez teníamos las imágenes generadas mediante el modelo y la parte de la aplicación generadora de imágenes integrada, ideé una manera en la que se pudiera

ver de una forma visual y continua, todos los recuerdos representados de una persona para que conformara lo que efectivamente es un libro de vida. Por ello, incluí una estructura de carrusel de imágenes que pudiera entenderse como los capítulos y las páginas de un libro. Además, me encargué de describir el ejemplo hipotético de la vida de un paciente con Alzheimer y adaptarlo al Capítulo 5 Libro de Vida.

Además, me encargué de organizar y estructurar el contenido de la memoria en la herramienta de Latex, utilizada para generar este mismo documento.

Bibliografía

*Y así, del mucho leer y del poco dormir, se
le secó el celebro de manera que vino a
perder el juicio.
(modificar en Cascaras\bibliografia.tex)*

Miguel de Cervantes Saavedra

BAUTISTA, T., OETIKER, T., PARTL, H., HYNA, I. y SCHLEGL, E. *Una Descripción de LATEX 2_E*. Versión electrónica, 1998.

KNUTH, D. E. *The TeX book*. Addison-Wesley Professional., 1986.

KRISHNAN, E., editor. *LATEX Tutorials. A primer*. Indian TeX Users Group, 2003.

LAMPORT, L. *LATEX: A Document Preparation System, 2nd Edition*. Addison-Wesley Professional, 1994.

MITTELBACH, F., GOOSSENS, M., BRAAMS, J., CARLISLE, D. y ROWLEY, C. *The LATEX Companion*. Addison-Wesley Professional, segunda edición, 2004.

OETIKER, T., PARTL, H., HYNA, I. y SCHLEGL, E. *The Not So Short Introduction to LATEX 2_E*. Versión electrónica, 1996.

Este texto se puede encontrar en el fichero Cascaras/fin.tex. Si deseas eliminarlo, basta con comentar la línea correspondiente al final del fichero TFGTeXiS.tex.

*-¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.*

*Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

*-Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.*

*Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes*

