UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

TESIS

"REDES GENERATIVAS ANTAGÓNICAS PARA LA SÍNTESIS Y GENERACIÓN DE TEXTO A IMAGEN: UN ANÁLISIS CUANTITATIVO Y CUALITATIVO DE CODIFICADORES DE PROCESAMIENTO DE LENGUAJE NATURAL PARA EL IDIOMA ESPAÑOL"

PARA OBTENER EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS EN CIENCIA DE LA COMPUTACIÓN CON MENCIÓN EN ESPECIALIDAD COMPUTACIÓN CIENTÍFICA

ELABORADO POR

EDUARDO YAURI LOZANO

ASESOR

DR. JOSE MANUEL CASTILLO CARA

LIMA – PERÚ

2023

Dedicatoria

Este trabajo va dedicado a mis queridos padres José y Clorinda por apoyarme incondicionalmente y aconsejarme en todos los momentos de mi vida forjándome como la persona que soy ahora, y a mi hermano Emerson por ser siempre un modelo y apoyo para seguir adelante. En memoria de mis abuelos paternos Anastacio y Eulalia, y abuelos maternos Nicolas y Basilia.

Agradecimientos

A mis Abuelos por todo el cariño brindado y los mejores padres que pude tener. A mis padres y hermano, por apoyarme, aconsejarme y motivarme en todo momento a culminar este trabajo y en general por apoyarme siempre y generar todas las oportunidades de las que disfruto.

Al Dr. Jose Manuel Castillo Cara, por el asesoramiento en la elaboración de la Tesis con bastante esmero y paciencia. Cada uno de sus consejos y recomendaciones fueron valiosas enseñanzas para ir mejorando la presente investigación.

A la Facultad de Ciencias de la Universidad Nacional de Ingeniería por brindarme una media beca para llevar la Maestría en Ciencias en Ciencia de la Computación. La presente Tesis es el resultado de la culminación satisfactoria de dicha Maestría.

Al Laboratorio Redes de Altas Prestaciones (RAAP) del Instituto de Investigación en Informática de Albacete (I^3A) de la Universidad de Castilla-La Mancha (UCLM) por brindarme la infraestructura computacional para realizar las pruebas de la presente investigación.

Resumen

En la actualidad, los avances en las técnicas de Inteligencia Artificial (IA) para la síntesis y generación de imágenes a partir de texto han experimentado un crecimiento y desarrollo constantes. Un caso especifico es la síntesis texto a cara mediante Redes Generativas Antagónicas (GANs), que consiste en generar una imagen facial a partir de una descripción textual de características físicas.

Siguiendo este enfoque, las investigaciones se han centrado en dos áreas principales, en Procesamiento del Lenguaje Natural (PLN) codificadores para la síntesis texto-cara, y en visión por computador, GANs para la generación texto-cara. Sin embargo, la mayoría de los codificadores se han desarrollado para el idioma inglés. En este contexto, este trabajo presenta un primer estudio de tres codificadores texto-cara diferentes, el modelo preentrenado RoBERTa, y los modelos Sent2Vec y RoBERTa, entrenados con un corpus descriptivo en español del conjunto de datos CelebA. Además, se presenta un modelo personalizado de Redes Generativas Antagónicas Convolucionales Profundas condicionadas (cDCGANs) entrenado con el conjunto de datos CelebA para la síntesis texto-cara en español.

Para la validación de los resultados obtenidos, se realiza una evaluación cualitativa con un análisis visual y una evaluación cuantitativa basada en las métricas IS, FID y LPIPS. Nuestra investigación muestra resultados prometedores con respecto a la literatura mejorando las métricas numéricas de FID en un 5 % y LPIPS en un 37 %. Además, esta misma investigación también muestra, a través de una comparación cuantitativa-cualitativa de las épocas de entrenamiento de cDCGAN, que la métrica IS no es una métrica objetiva adecuada para ser considerada en la evaluación de trabajos similares.

El generador implementado puede ser usado como una herramienta valiosa para la elaboración del retrato hablado de una persona y tiene múltiples áreas de aplicación. Siendo una de los principales el ámbito Policial. Dado que para la identificación de criminales es necesario realizar bocetos de dichas personas.

Índice general

Índice de figuras

Índice de tablas

Índice de Acrónimos

GAN	Generative Adversarial Networks
CGAN	Conditional Generative Adversarial Networks
NLP	Natural Language Processing
DCGAN	Deep Convolutional Generative Adversarial Networks
cDCGAN	Conditional Deep Convolutional Generative Adversarial Networks
SBERT	Sentence Bidirectional Encoder Representations from Transformers
FID	Frechet Inception Distance
IS	Inception Score
LPIPS	Learned Perceptual Image Patch Similarity
CWGAN	Conditional Wasserstein Generative Adversarial Network
NLG	Natural Language Generation
NLTK	Natural Language Toolkit
SAGAN	Self Attention Generative Adversarial Network
DFGAN	Deep Fusion Generative Adversarial Networks
BERT	Bidirectional Encoder Representations from
CNN	Convolutional Nueural Network
CNN	Convolutional Nueural Network
MCGAN	Multi Conditional Generative Adversarial Network
SSIM	Structural Similarity Index Measure
PSNR	Peak Signal to Noise Ratio
CBOW	Continuous Bag of Words
RNN	Recurrent Neural Networks
ROBERTA	Robustly Optimized BERT Pretraining Approach

Capítulo 1

Introducción

En el presente capítulo, se presentan las motivaciones para el desarrollo del presente trabajo enfocadas en el ámbito académico y social. Además, se expone el objetivo principal y los objetivos específicos. Finalmente, se describe de forma resumida la estructura de cada uno de los capítulos que componente el presente trabajo de investigación.

1.1. Motivación

En el Perú, año tras año se viene dando un aumento de hechos delictivos en sus diferentes modalidades en el ámbito policial. Esto se ve reflejado en el aumento constante en el número de personas detenidas por haber cometido delitos. Entre los años 2016 y 2021 la tasa aumento en 56,1 % al pasar de 111233 a 173616 detenciones respectivamente, siendo las mayores taza de aumento en delitos contra la vida, el cuerpo y contra el patrimonio [1]. En la mayoría de estos no se conoce a priori la identidad de los perpetradores, por lo cual, es necesario construir un perfil o retrato hablado de la persona como punto de partida para las investigaciones.

El retrato hablado es definido como una disciplina artística y consiste en elaborar el rostro o retrato de una persona tomando como guía los datos fisiológicos aportados por un observador externo. Sin embargo, en la actualidad su forma más común de elaboración es realizando un dibujo a mano. Por

lo tanto, dicho proceso generalmente resulta lento, dificultoso y con poca precisión; ello a su vez, en el ámbito policial traza el accionar y genera malestar en los interesados.

En el área de las tecnologías de la información, en los últimos años los avances en las técnicas de aprendizaje automático para la síntesis de imágenes han tenido un constante crecimiento. En el año 2014, se dió inicio con el desarrollo de una arquitectura denominada Red Generativa Antagónica (GAN, por sus siglas en inglés *Generative Adversarial Networks*) [2]. Este nuevo modelo generó resultados bastante prometedores. Sin embargo, debido a la limitación de no poder especificar características puntuales que se deseará para las imágenes creadas, ese mismo año se implementó un nuevo modelo denominado Red Generativa Antagónica Condicional (cGAN, por sus siglas en inglés *Conditional Generative Adversarial Networks*) [3]. Este modelo introduce entradas adicionales al modelo GAN original. Esta nueva contribución científica permite que el modelo se entrene con información como etiquetas de clase u otras variables condicionantes, así como con las propias muestras al mismo tiempo [4].

Actualmente según [4], las cGAN han sido mejoradas por un lado utilizando Redes Convoluciones Profundas (DCNN, por sus siglas en inglés *Deep Convolutional Neural Networks*) en vez de redes *fully-connected*, dado que son más adecuadas para trabajar con imágenes. Y por otra parte, se han desarrollado numerosas técnicas para generar los vectores de información que ingresan a la red para la generación de imágenes. En este sentido una de las más recientes, robustas y eficientes es haciendo uso de *transformers* cuyo modelo es el estado de arte del Procesamiento del Lenguaje Natural (NLP por sus siglas en inglés *Natural Language Processing*) en la actualidad.

El desafío para la elaboración de imágenes para retrato hablado y el avance de los modelos de aprendizaje automático antes descritos, han motivado la implementación de una arquirectura generadora empleando *Sentence-BERT* (SBERT) [5] como codificador del texto descriptivo de entrada y una Red Generativa Antagónica Convolucional Profunda Condicionada (cDCGAN,

por sus siglas en inglés *Conditional Deep Convolutional Generative Adversarial Networks*) para generar imágenes de rostros. Es importante resaltar que se realizará un entrenamiento personalizado del codificador para que trabaje con texto en idioma Español, una de las contribuciones principales de este trabajo ya que no existe previamente un modelo pre-entrenado para esta tarea.

1.2. Objetivos

El objetivo principal del presente trabajo es el desarrollo una arquitectura cDCGAN que pueda generar imágenes de rostros a partir de entradas de texto en idioma español con información de características físicas que guiarán su creación. La arquitectura planteada hará uso de los modelos SBERT y Sent2vec [6] para codificar el texto de entrada a la red y utilizará una red cDCGAN para generar imágenes.

Adicionalmente, este trabajo tiene los siguientes objetivos específicos:

- Crear un corpus de oraciones descriptivas en idioma español (*captions*) para las imágenes del conjunto de datos CelebA[7, 8].
- Crear un corpus de entrenamiento para el modelo SBERT a partir del corpus antes generado.
- Crear un corpus de entrenamiento para el codificador Sent2vec a partir del corpus antes generado.
- Entrenar el *transformer* a partir del modelo base para idioma español `RoBERTa-large-bne` [9, 10] (de aquí en adelante llamado RoBERTa) utilizando el conjunto de datos generados, de modo que se incremente su rendimiento en la codificación de oraciones.
- Entrenar el codificador Sent2vec con el corpus en español generado anteriormente.
- Diseñar una arquitectura cDCGAN para generar imágenes, tomando como base las referencias estudiadas en el Estado del Arte.

- Implementar las métricas de evaluación de calidad de imágenes: *Frechet Inception Distance* (FID) [11], *Inception Score* (IS) [12] y *Learned Perceptual Image Patch Similarity* (LPIPS) [13].

- Establecer una comparación cuantitativa y cualitativa de calidad de imágenes generadas en base al modelo del codificador de oraciones utilizado: Sent2vec entrenado con el corpus de CelebA (de aquí en adelante llamado Sent2vec+CelebA), el modelo base RoBERTa y nuestro modelo propio entrenando al modelo RoBERTa con el corpus de CelebA (de aquí en adelante llamado RoBERTa+CelebA).

Y los objetivos con respecto a las competencias académicas desplegadas en el trabajo son:

- Establecer el uso de lenguajes y librerías *Open Source* para la implementación y pruebas de la arquitectura desarrollada.

- Examinar los conceptos de NLP y *transformers* enfocados en el campo de *sentence embedding*. Así como también, analizar los tipos de arquitecturas GAN y sus mejoras para la síntesis de imágenes.

- Establecer una metodología para la generación y preprocesamiento de un conjunto de datos de entrenamiento y la implementación y entrenamiento de la arquitectura generadora.

1.3. Contribuciones de la Tesis

Con el fin de lograr los objetivos planteados en la presente tesis, se implementaron una conjunto de recursos los cuales quedarán como aporte académico para futuros trabajos. Las principales contribuciones de la presente tesis son:

- Implementar una cDCGAN para generación de rostros para texto en Español.

- Crear un corpus descriptivo para el conjunto de datos CelebA en idioma Español.
- Diseñar un algoritmo propio para generar un corpus de entrenamiento para SBERT en Español, usando los corpus en Español e inglés de CelebA.
- Realizar un entrenamiento personalizado de los codificadores RoBERTa y Sent2vec con los corpus de CelebA desarrollados en español. Los modelos resultantes pueden usarse para futuros trabajos relacionados a esta área.
- Realizar una evaluación comparativa de los codificadores para determinar la mejor configuración para el cDCGAN.
- Realizar análisis cuantitativo (IS, FID y LPIPS) y cualitativo (visual) de las imágenes sintéticas generadas por la cDCGAN en función del codificador: Sent2vec+CelebA, RoBERTa y RoBERTa+CelebA.
- Demostrar que IS no es una métrica confiable para la generación de texto a cara y debe ser reemplazada por métricas más sólidas como FID y LPIPS

1.4. Estructura de la Tesis

Para brindar al lector una idea global del contenido de este trabajo, a continuación se hace una breve descripción del propósito de cada capítulo en la presente tesis.

- **Capítulo 1: Introducción**

  En este capítulo, se presentan las motivaciones para el desarrollo del presente trabajo enfocadas en el ámbito académico y social. Además, se expone el objetivo principal y los objetivos específicos. Finalmente, se describe de forma resumida la estructura de cada uno de los capítulos que componente el presente trabajo de investigación.

- **Capítulo 2: Estado del arte**

  En esta sección se discutirán los principales conceptos relacionados a las

GAN y sus variantes, así como también, las principales métricas para evaluar la calidad de imágenes generadas. Se describirá la clasificación de modelos para síntesis de imágenes a partir de texto descriptivo. Finalmente, se estudiarán las principales herramientas de NLP enfocadas en *sentence embedding*.

- **Capítulo 3: Marco Teórico**

  En el presente capítulo, se realizará una descripción de trabajos relacionados con la generación de imágenes a partir de descripción textual utilizando redes GAN y, más específicamente, la generación de rostros. Dichos estudios nos servirán como punto de partida y contribuirán al desarrollo del presente trabajo, el cual abarca la solución de problemas en varias áreas de estudio. Por tanto, en esta sección nos enfocaremos en las 3 principales áreas en las que se enmarca esta tesis: (i) Incrustación de oraciones; (ii) Generación de imágenes guiada por texto; y (iii) Generación de rostros guiada por texto.

- **Capítulo 4: Metodología e implementación**

  En este punto se define la metodología usada en la implementación del presente trabajo y se describen sus fases más importantes del desarrollo del mismo. Adicionalmente, se detalla el preprocesamiento aplicado al corpus descriptivo e imágenes del conjunto de datos CelebA. Finalmente, se describe la arquitectura de la red cDCGAN usada para generar las imágenes sintéticas.

- **Capítulo 5: Experimentación y resultados**

  En este capítulo se mostrarán los resultados obtenidos luego del entrenamiento y testeo del modelo generador de imágenes planteado. Además, se mostrará también los resultados de los codificadores utilizados. Finalmente, se realiza una discusión de los resultados obtenidos en contraste con los ya obtenidos en la literatura.

- **Capítulo 6: Conclusiones y trabajo a futuro**

  En la última sección se mencionan las principales conclusiones obtenidas

en el presente trabajo. Adicionalmente, se establecerán los posibles trabajos a futuro que se pueden desarrollar partiendo del modelo implementado.

Capítulo 2

Marco Teórico

En este capítulo se discutirán los principales conceptos relacionados a las GAN y sus variantes, así como también, las principales métricas para evaluar la calidad de imágenes generadas. Se describirá la clasificación de modelos para síntesis de imágenes a partir de texto descriptivo. Finalmente, se estudiarán las principales herramientas de NLP enfocadas en *sentence embedding*.

2.1. Generación de imágenes usando modelos generativos

Los algoritmos de aprendizaje automático son una gran herramienta para reconocer patrones en los datos existentes y usar esa información para tareas como clasificación y regresión. Sin embargo, se ha observado que tienen dificultades para generar nuevos datos [14]. Este problema cambió en el año 2014 cuando se desarrollaron las GAN. Esta técnica ha permitido a las computadoras generar datos realistas utilizando dos redes neuronales separadas: generativa y discriminadora. En la actualidad sus resultados y versatilidad distinguen a este tipo de redes del resto.

2.1.1. Redes generativas antagónicas

Según Jason Brownlee [14], las GAN son una forma inteligente de entrenar un modelo generativo al definir el problema como uno de aprendizaje supervisado con dos submodelos. De esta manera puede usarse para generar nuevos datos que tengan igual calidad que el conjunto de datos original de manera convincente. Se tiene un modelo o red generadora para crear nuevos ejemplos; y otro modelo o red discriminadora que intenta clasificar los ejemplos como reales o falsos. Los dos modelos se entrenan juntos en un juego de suma cero hasta que el modelo discriminador es "engañado" la mitad de las veces, aproximadamente; lo que significa que el modelo generador está creando datos plausibles.

En la Figura 2.1 podemos observar una arquitectura básica de una GAN, la cual está compuesta por una red neuronal generadora G, y una red neuronal discriminadora D. Al comienzo, G crea imágenes a partir de un vector de ruido simple, generalmente una distribución normal. El trabajo de D será etiquetarlas como verdaderas o falsas a partir de la recepción como entradas, tanto a imágenes reales como a las generadas. El entrenamiento de ambas redes se realiza utilizando *backpropagation* de modo que, con este método, G recibe retroalimentación desde D mejorando gradualmente su producción.

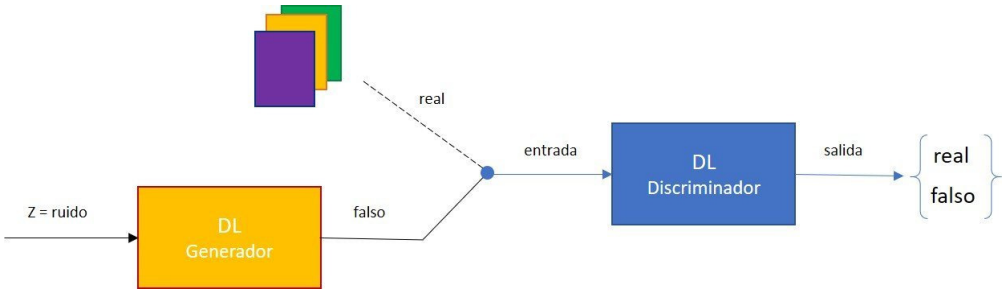


FIGURA 2.1: Arquitectura general de una red GAN. Fuente: IArtifical [15].

Por otro lado, la Ecuación 2.1 muestra la función objetivo general para las redes generativas. Dicha función se define análogamente al juego *min-max* entre las salidas esperadas de los modelos generador y discriminador buscando

el equilibro de **Nash** [1], donde θ_g y θ_d son los parámetros de ejecución del generador y discriminador, respectivamente. La función $D(x)$ representa al discriminador, el cual, para una imagen x la clasifica en un espacio binario (0 para imagen falsa o 1 para imagen genuina); la salida ideal de dicha función debería ser 1. La función $G(z)$, utilizando un vector de ruido z, genera la imagen sintética o falsa. A partir de ello, la salida ideal de $D(G(z))$ debería ser 0 lo cual indica que ha identificado satisfactoriamente a la imagen creada. Durante el entrenamiento el discriminador tiene la intención de diferenciar las imágenes verdaderas de las falsas con la máxima capacidad max_d; mientras que el generador tiene la intención de minimizar la diferencia entre una imagen falsa y una imagen verdadera min_g [4].

$$\min_g \max_d v(\theta_g, \theta_d) = E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p_z}[\log 1 - D(G(z)] \tag{2.1}$$

En este sentido, este tipo de redes neuronales son muy eficientes en la tarea de completar información faltante durante el proceso de generación, por lo que tiene un gran número de aplicaciones. En [16] se definen algunas de las principales aplicaciones:

- **Generación de Imágenes:** Las redes generativas se pueden utilizar para generar imágenes realistas después de haber sido entrenadas con bases de datos de muestra. Una vez finalizada la formación, la red de generadora podrá crear nuevas imágenes diferentes a las imágenes del conjunto de entrenamiento. La generación de imágenes se utiliza en marketing, generación de logotipos, entretenimiento, redes sociales, etc.

- **Síntesis de texto a imagen o voz a imagen:** La generación de imágenes guiadas por descripciones de texto o voz son casos particularmente interesantes. En el área de seguridad, su uso puede ser muy importante a la hora de crear perfiles de rostros de criminales a partir de las descripciones provistas por los denunciantes. También es usado en la

[1] `https://jonathan-hui.medium.com/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b`

industria cinematográfica o de los *comics* para crear nuevo contenido audiovisual.

- **Envejecimiento facial:** Generar versiones más jóvenes o más maduras de una persona es otra de las aplicaciones de este tipo de redes. Esto puede ser muy útil tanto para la industria del entretenimiento como para la vigilancia. Es particularmente útil para la verificación facial porque significa que una empresa no necesita cambiar sus sistemas de seguridad a medida que las personas envejecen.

- **Traducción de imagen a imagen:** Transferir características visuales de imagen a imagen tiene múltiples aplicaciones. Por ejemplo, se usa para convertir imágenes tomadas durante el día en imágenes tomadas por la noche, convertir imágenes aéreas a imágenes satelitales automáticamente. Estos casos de uso son innovadores porque pueden ahorrarnos tiempo de creación.

- **Generación de imágenes en alta resolución:** Otra importante aplicación de las GAN es la generación de imágenes de una mayor resolución sin perder sus detalles esenciales. Mejoramiento de imágenes antiguas, aumento de la resolución de imágenes satelitales o mejoramiento de imágenes clínicas son algunas aplicaciones importantes.

2.1.2. Redes generativas antagónicas condicionales

Este modelo fué introducido en el año 2014 por Mehdi Mirza y Simon Osindero. Las cGAN son un tipo de red antagónica cuyos generador y discriminador se condicionan durante el entrenamiento mediante el uso de información adicional. Estos datos condicionales pueden ser etiquetas de clase, descripción escrita e incluso entradas de voz [17].

Utilizando el modelo original se puede producir ejemplos que van desde simples dígitos escritos a mano hasta imágenes fotorrealistas de rostros humanos. Sin embargo, aunque se puede controlar el dominio de los ejemplos

que la GAN aprendió a emular mediante selección del conjunto de datos de entrenamiento, no es posible especificar ninguna de las características de las muestras de datos que generaría. Es por ello que las cGAN son una de las primeras innovaciones que hizo posible la generación de datos específicos y posiblemente la más influyente [17].

Por tanto, tomando como referencia el diagrama mostrado en la Figura 2.2, tenemos un vector de ruido z y una etiqueta condicional, y. Estas entradas generarán una imagen artificial $x^*|y$ la cual cumple con las condiciones definidas en y. Siguiendo este proceso y gracias a la retroalimentación del discriminador, las imágenes generadas van mejorando considerablemente. Por otra parte, el discriminador recibe una imagen real x o una sintética $x^*|y$ y la etiqueta condicional y. De esta forma aprende a diferenciar imágenes falsas (sintéticas) de las verdaderas y, finalmente, como salida se tiene un booleano indicando si fue capaz de reconocer o no a la imagen falsa.

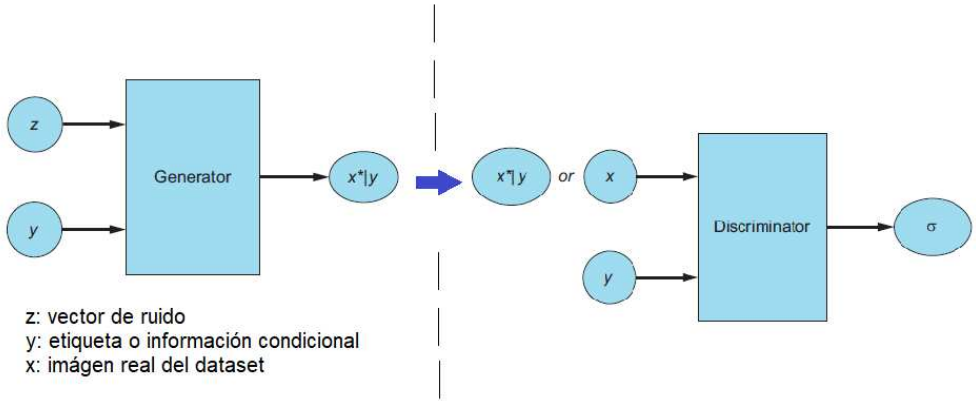


FIGURA 2.2: Arquitectura de las redes Generador y Discriminador de CGAN. Fuente: Elaboración Propia.

En este contexto, como se puede observar en la Ecuación 2.2, la función objetivo es muy similar a la desarrollada por las GAN. Sin embargo, las funciones del discriminador y generador ahora dependen de la etiqueta de clase o información adicional y. El vector condicional no se limita solo a etiquetas de clase; por el contrario, puede provenir de diferentes fuentes y en diferentes tipos de información. Sin embargo, la entrada a la red solo permite ingresar vectores numéricos, así, por tanto, será necesario codificar y.

Actualmente existen numerosas herramientas y técnicas de NLP para este fin, los cuales describiremos en los siguientes apartados [4].

$$\min_{g} \max_{d} v(\theta_g, \theta_d) = E_{x,y\sim p_{data}}[\log D(x,y)] + E_{x\sim p_z, y'\sim p_y}[\log 1 - D(G(z,y'),y'))] \quad (2.2)$$

2.1.3. Redes generativas antagónicas convolucionales profundas

Planteadas por Radford en [18], a diferencia de las redes GAN comunes las cuales utilizan capas densas intermedias tanto en su generador como en su discriminador, estas emplean capas convolucionales y deconvolucionales también conocidas como convolucionales transpuestas. El generador es un conjunto de capas deconvolucionales o de transposición, junto a capas de *Batch normalization* y capas LeakyReLU. El discriminador consiste en un conjunto de capas convolucionales escalonadas, a las cuales también se componen de capas complementarias similares al generador entre ellas *Batch normalization* y ReLU [14].

Históricamente, hacer uso de Redes Neuronales Convolucionales (CNN, por sus siglas en inglés *Convolutional Neural Networks*) para la implementación del modelo generador y discriminador de una GAN fue un gran desafío su desarrollo y sujeto a fallos. En [18] se describen una conjunto de modificaciones que se deben realizar a las CNN para obtener arquitecturas DCGAN estables:

- Reemplazar todas las capas de agrupación (*pooling*) con capas Convolucionales con estrías para el discriminador y capas Convolucionales con estrías fraccionadas para el generador.
- Usar la técnica *Batch normalization* tanto en el generador como en el discriminador.
- Eliminar capas ocultas totalmente conectadas para arquitecturas más profundas.

- Usar la función de activación ReLU en el generador para todas las capas excepto para última; en la capa de salida debe usarse Tanh.
- Utilizar la función de activación de LeakyReLU en el discriminador para todas las capas.

En este sentido, en la Figura 2.3 se observa la arquitectura de una DCGAN condicionada. El generador toma como entradas a un vector de ruido z junto a un vector condicional y. A partir de las entradas a través de una serie de capas convolucionales transpuestas, se realiza un muestreo ascendente escalando positivamente las dimensiones de las imágenes. La imagen creada por el generador junto al vector condicional y ingresan al discriminador, el cual, a través de una serie de capas convolucionales, hace el proceso de submuestreo desescalando la imagen a una de menor dimensión. Al final, como salida del discriminador se tiene un valor *booleano* que indicará si la imagen logro engañar o no a dicho modelo.

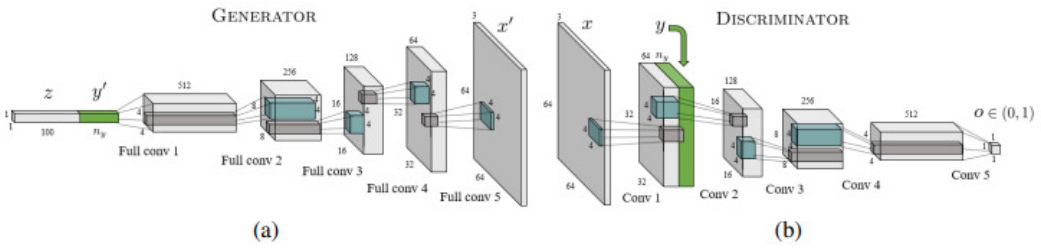


FIGURA 2.3: Arquitectura de un modelo CDCGAN. Fuente: CoRR [19].

2.1.4. Síntesis de texto a imagen utilizando CGAN

Una caso particular y aplicativo de las cGAN estudiadas es la síntesis de texto a imagen. Para este caso, el texto descriptivo será el condicionante para guiar la generación de la imagen. El texto descriptivo puede incluir características físicas de una persona, características de un animal en particular o la descripción de un paisaje, entre otros [4].

Sin embargo, una desventaja sensible de usar cGAN para la síntesis de texto a imagen, es que no puede manejar descripciones textuales complicadas para la generación de imágenes porque los modelos iniciales y comunes usan etiquetas como condiciones para restringir las entradas. Si las entradas de texto tienen varias palabras clave (o descripciones de texto largas), no se pueden utilizar simultáneamente para restringir la entrada. Por este motivo, y en base a estas arquitecturas iniciales, se han creado numerosas arquitecturas de red y *frameworks* GAN para generar imágenes con diferentes diseños y arquitecturas, uso de discriminadores múltiples, uso de discriminadores entrenados progresivamente o uso de discriminadores jerárquicos.

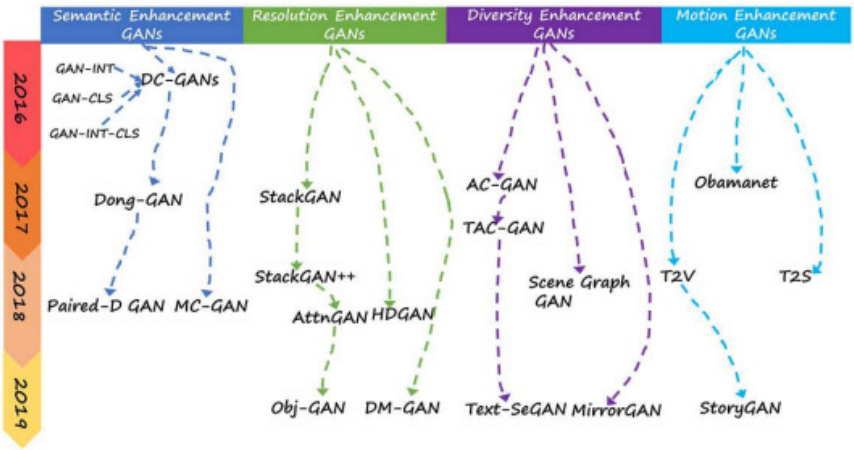


FIGURA 2.4: Taxonomía de arquitecturas GAN para síntesis de texto a imagen. Fuente: Willey [4].

Por tanto, en la Figura 2.4 se puede apreciar un resumen completo por años y por categorías de las diferentes arquitecturas o *frameworks* avanzados para la síntesis de imágenes condicionadas por texto. Así, en la investigación desarrollada en [4], se propone generar una taxonomía para clasificar todas las arquitecturas y *frameworks* para síntesis de texto a imagen usando GAN avanzadas definiéndose las siguientes categorías:

- **GANs de mejora semántica:** Representan trabajos pioneros para la síntesis de texto a imagen, su enfoque principal es garantizar que las

imágenes generadas estén relacionadas semánticamente con los textos de entrada. Este objetivo se logra mediante el uso de una red neuronal para codificar textos como características densas, que luego se alimentan a una segunda red para generar imágenes que coincidan con los textos. En esta categoría destacan los modelos DCGAN y *Multi Conditional Generative Adversarial Network* (MCGAN [20], por sus siglas en inglés).

- **GANs de mejora de resolución:** Las GAN de mejora de resolución se centran en generar imágenes de alta calidad que se emparejan y concuerdan semánticamente con los textos de entrada. Esto se logra a través de un marco de etapas múltiples, donde los resultados de las de etapas anteriores alimentan a las posteriores para generar imágenes de mejor calidad. Destacan las redes *AttnGAN* y *StackGAN* [21].

- **GAN de mejora de la diversidad:** Los modelos para la mejora de la diversidad tienen la intención de diversificar las imágenes de salida, de modo que las imágenes generadas no solo estén relacionadas semánticamente, sino que también tengan diferentes tipos y apariencia visual. Este objetivo se logra mediante un componente adicional para estimar la relevancia semántica entre imágenes y textos generados con el fin de maximizar la diversidad de salida. Entre las arquitecturas que destacan en esta categoría tenemos a ACGAN [22] y *MirrorGAN* [23].

- **GAN de mejora de movimiento:** Las GAN de mejora de movimiento tienen la intención de agregar una dimensión temporal a las imágenes de salida, de modo que puedan formar acciones significativas con respecto a las descripciones del texto. Este objetivo se logró principalmente a través de un proceso de dos pasos que primero genera imágenes que coinciden con las acciones de los textos, seguido de un procedimiento de mapeo o alineación para garantizar que las imágenes sean coherentes en el orden temporal. Se tienen como modelos representativos a *StoryGAN* [24] y *TV2* [25].

2.2. Métricas de evaluación de GANs

Las GAN han llegado a ser notablemente eficaces para generar imágenes sintéticas de alta calidad. Sin embargo, en lugar de entrenarse en solitario como la mayoría de estas redes, los modelos generadores son entrenados junto a un segundo modelo, llamado discriminador, que aprende a diferenciar imágenes reales de imágenes falsas o generadas. Como tal, no existe una función o medida objetivo para el generador. A continuación, se describen algunas de las principales métricas de evaluación que usaremos en el presente trabajo [24].

2.2.1. Inception Score

Es una métrica objetiva adecuada para evaluar la calidad de las imágenes generadas y a su vez tratar de eliminar la evaluación humana subjetiva en las imágenes. Fue planteada por Tim Salimans et al. en el año 2016 [14].

Esta métrica implica el uso de un modelo de red neuronal de aprendizaje profundo previamente entrenado Inception-v3, el cual fue entrenado con la base de datos ImageNET. Una gran cantidad de imágenes generadas se clasificaron utilizando el modelo y, en concreto, se predice la probabilidad de que una imagen pertenezca a cada clase definida en ese conjunto de datos. La clasificación depende de dos criterios:

- **Calidad de imagen:** Se plantea la pregunta, ¿las imágenes se ven como un objeto específico definido?
- **Diversidad de imágenes:** Se plantea, ¿se genera una amplia gama de objetos de diferentes clases?

En este contexto, para un conjunto de imágenes a evaluar, como es el caso representado en la Figura 2.5 de generación de imágenes de animales, se compara la distribución de etiquetas de cada imagen (qué tan bien etiquetada está la imagen) con la distribución marginal (qué tanta variedad de clases se generó) de todo el conjunto de datos. Así, el valor de diferencia entre ambas distribuciones nos proporcionará el puntaje final, IS.

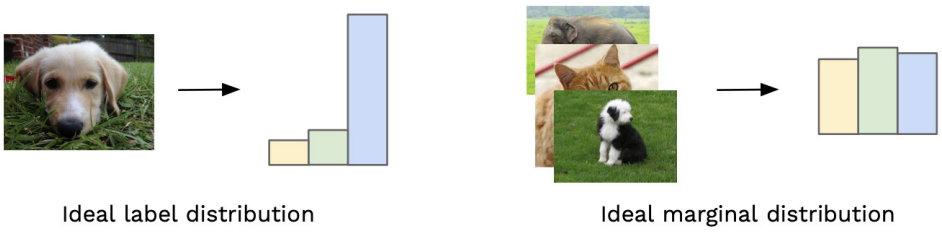


FIGURA 2.5: Distribución de etiquetas y distribución marginal.
Fuente: Medium.com [26].

Para su cálculo se utiliza la formula estadística de divergencia de Kullback-Leibler (KL) mostrada en la Ecuación 2.3, donde $KL(p(y|x) \parallel p(y))$ es la divergencia KL entre la distribución condicional $p(y|x)$ y la distribución marginal $p(y)$. La puntuación más baja posible para IS es cero. Por otro lado, matemáticamente el puntaje más alto posible es infinito aunque, en la práctica, es igual al número de clases que contiene la base de datos donde fue entrenada, cuyo número era 1000 en el año 2014 [27, 12].

$$IS = exp\left(E_x KL(p(y|x) \parallel p(y)\right) \tag{2.3}$$

Es importante resaltar que, si bien es una de las más medidas más utilizadas hasta hace poco, actualmente ha sido reemplazado en gran medida por nuevos y más completos métodos a la hora de evaluar imágenes generadas debido a las siguientes desventajas descritas en [28]:

- La puntuación está limitada a lo que el clasificador de Inception-v3 puede detectar, que a su vez está directamente vinculado a los datos de entrenamiento ILSVRC.
- Puede obtenerse un valor bajo, aún así, las imágenes generadas son de alta calidad si estas etiquetas no se encuentran en la data con la cual fue entrenado el clasificador.
- El clasificador no es capaz de detectar características relevantes para ver la calidad de imágenes según un criterio determinado. Por ejemplo, puede generar personas con 3 ojos y seguir recibiendo alto puntaje.

- Existen varias formas de engañar al clasificador para obtener puntuaciones altas.

2.2.2. Frechet Inception Distance

FID es una métrica que calcula la distancia entre los vectores de características calculados para imágenes reales y generadas. Al igual que IS, la métrica FID también utiliza al modelo *Inception-v3*. Más precisamente, la última capa de codificación antes del clasificador final para obtener el vector de características específicas de la imagen de entrada cuya dimensión es 2048 [14].

Por tanto, se generan los vectores de características para las imágenes originales y las generadas; al final se tienen dos grupos de vectores de dimensión 2048. Utilizando dichas características, se pueden formar distribuciones Gausianas Multivariadas y, para calcular la distancia entre ellas, se utiliza la distancia de Frechet, usando las medias y las matrices de covarianza de los vectores de características según la fórmula mostrada en la Ecuación 2.4. Por tanto, en la ecuación 2.4, se puede observar que la métrica FID depende de μ y μ_w, medias de vectores de características de las imágenes originales y creadas, respectivamente. También participan las matrices de covarianza $\sum$ de los vectores de características de las imágenes reales y $\sum_w$ misma matriz para las imágenes creadas.

$$FID = |\mu - \mu_w|^2 + tr\left(\sum + \sum_w -2\left(\sum \sum_w\right)^{\frac{1}{2}}\right) \tag{2.4}$$

El objetivo de la métrica FID es evaluar la calidad de un conjunto de imágenes sintéticas en función de sus indicadores estadísticos, en comparación con los indicadores estadísticos de un conjunto de imágenes reales del mismo dominio de destino. Un valor pequeño o bajo de FID indica imágenes de mejor calidad; por el contrario, una puntuación alta indica imágenes de menor calidad. La relación de estos valores con la calidad puede ser lineal [14].

En este contexto, en la Figura 2.6 se observa los distintos valores de FID calculados para una variedad de casos de distorsión de un conjunto de imágenes. Se muestran las siguientes distorsiones:

- superior izquierdo, ruido gaussiano;
- superior medio, desenfoque gaussiano;
- superior derecho, rectángulos negros implantados;
- inferior izquierdo, distorsión de remolino;
- inferior medio, ruido de sal y pimienta;
- inferior derecho, mezcla de imágenes de los conjuntos de datos CelebA y *ImageNET*.

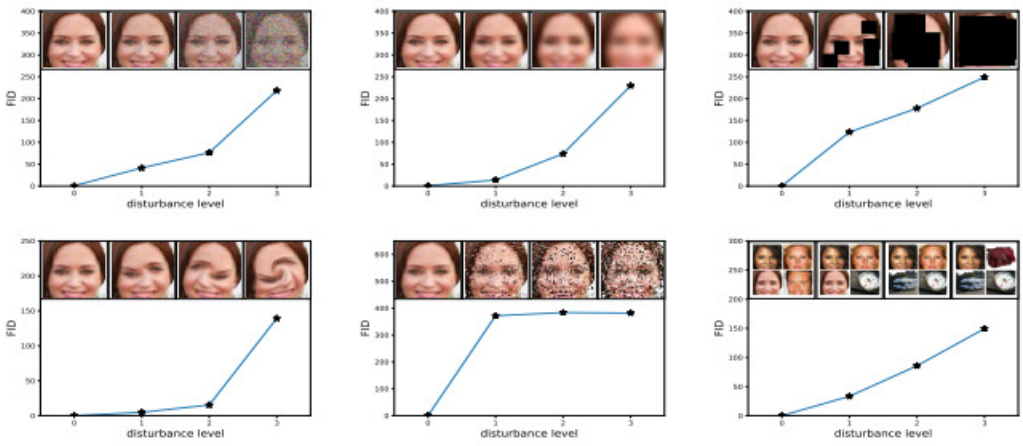


FIGURA 2.6: Comparación de valores FID obtenidso frente a distorsión de imágenes. Fuente: NeurIPS [11].

Por consiguiente, en [11] se menciona que las distorsiones comienzan desde cero y van aumentando de izquierda a derecha. Se observa que los valores de FID van aumentando en la misma dirección con algunas variaciones en proporción de cambio.

2.2.3. Learned Perceptual Image Patch Similarity

Según los autores de [13], las medidas clásicas y más usadas para evaluar la calidad de las imágenes son funciones simples y superficiales que pueden obviar algunos matices de la percepción humana. Métricas como distancia euclidiana (SSIM, por sus siglas en inglés *Structural Similarity Index Measure* o (PSNR, por sus siglas en inglés *Peak Signal-to-Noise Ratio*), asumen independencia en cuanto a píxeles y no realizan la evaluación como un total. Por tanto, se desea implementar una distancia perceptiva que pueda medir que tan similares son dos imágenes y que a su vez, coincida con el juicio humano.

Así, el objetivo de la métrica LPIPS es hacer uso de las características de las redes profundas para medir la similitud entre imágenes. Hace algún tiempo, la comunidad de visión por computadora descubrió que las características de la red VGGNet [29] entrenadas en la clasificación de ImageNET han sido notablemente útiles como una medida de pérdida de entrenamiento para la síntesis de imágenes. A partir de ese hecho, se han realizado pruebas de medición de distancia de características utilizando diferentes arquitecturas entre las que además de la primera destacan SqueezeNet [30] y AlexNet [31] obteniendo resultados similares bastante buenos [13].

Por tanto, dadas dos imágenes x y x_0, se calcula la distancia entre dos porciones de imagen utilizando una red establecida F, según la fórmula mostrada en la Ecuación 2.5.

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left| w_l \odot \left(\hat{y}^l_{hw} - \hat{y}^l_{0hw} \right) \right|_2^2 \tag{2.5}$$

Dónde:

- $\widehat{y}^l$ y $\widehat{y}^l_0$ son las pilas de características extraídas a ambas imágenes en la capa l,
- w_l es un vector de escalado.
- H y W los componentes espaciales.

El cálculo empieza obteniendo las características de las imágenes, luego se normalizan en la dimensión del canal, se escala cada uno de los canales usando el vector w_i, se toma la distancia coseno y, finalmente, se realiza un promedio a través de la dimensión espacial y en todas las capas. Cuanto más pequeño es el valor de la métrica más parecidas son las imágenes [13].

2.3. Técnicas para incrustación de oraciones

Las técnicas de incrustación de oraciones (*sentence embedding*) han cambiado radicalmente el campo del NLP en los últimos años dado que posibilitan codificar fragmentos de texto como vectores de tamaño fijo. Actualmente, han dado un gran impulso a muchos campos de la inteligencia artificial, entre los que destaca la síntesis de imágenes condicionadas debido a que la única forma de ingresar la información a las redes neuronales es mapeándola en vectores de números reales [32].

Las incrustaciones son vectores multidimensionales de números reales de longitud fija que permiten extraer y manipular el significado de los segmentos que representan, por ejemplo, comparando cuán similares son dos oraciones semánticamente entre sí. Actualmente, existe una conjunto de estos modelos también llamados codificadores y entre los más usados se pueden mencionar a Sent2vec y Sentence-BERT [33].

2.3.1. Incrustación de oraciones utilizando Sent2Vec

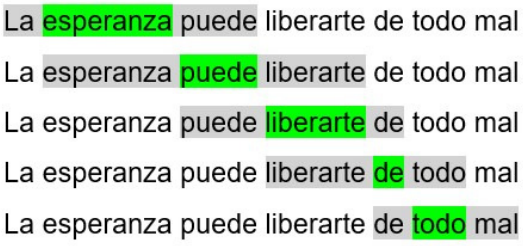
El algoritmo Sent2vec combina las técnicas y fundamentos de un conjunto de modelos y algoritmos previamente diseñados, los cuales describiremos brevemente a continuación.

Word2Vec

En primero lugar tenemos Word2Vec [34] que es un modelo de red neuronal no supervisado el cual trabaja con grandes cantidades de texto. Permite crear

un vocabulario de palabras y un conjunto de vectores de incrustación para cada palabra en un determinado espacio vectorial que representa a dicho vocabulario [35].

Como primer paso de ejecución, se crea un vocabulario y, posteriormente, una representación *one-hot encoding* del mismo como se observa en la Figura 2.7. Además, se define una ventana de palabras con las cuales se entrenará la red neuronal siendo tres para este caso.



Palabra	One hot vector
La	[1,0,0,0,0,0,0]
esperanza	[0,1,0,0,0,0,0]
puede	[0,0,1,0,0,0,0]
liberarte	[0,0,0,1,0,0,0]
de	[0,0,0,0,1,0,0]
todo	[0,0,0,0,0,1,0]
mal	[0,0,0,0,0,0,1]

FIGURA 2.7: Representación *one-hot encoding* y selección de ventana de palabras. Fuente: Elaboración propia.

Este método ofrece dos arquitecturas para crear los vectores de incrustación de palabras, éstas se definen dependiendo del enfoque utilizado para seleccionar que palabras que serán objetivos y adyacentes. Se tienen las siguientes arquitecturas [35]:

- **Continuous Bag of Words (CBOW)**: Dado un conjunto de oraciones, para una determinada palabra existen otras adyacentes o vecinas con las que en conjunto forman parte del contexto de la oración, como se observa en la red derecha de la Figura 2.8. El objetivo de ésta metodología es predecir una palabra objetivo en base a las palabras adyacentes utilizando una red neuronal simple. La red se entrenará de manera continua con cada

tupla de palabras *adyacente-objetivo-adyacente* en su representación *one-hot encoding*.

- **Skip-gram**: Al igual que CBOW este modelo también trabaja con un conjunto de palabras de las cuales se tiene un palabra objetivo y algunas adyacentes a ella. Sin embargo, trabaja de forma opuesta a CBOW dado que utiliza la palabra objetivo para predecir las palabras vecinas utilizando una red neuronal como se observa en la red izquierda de la Figura 2.8.

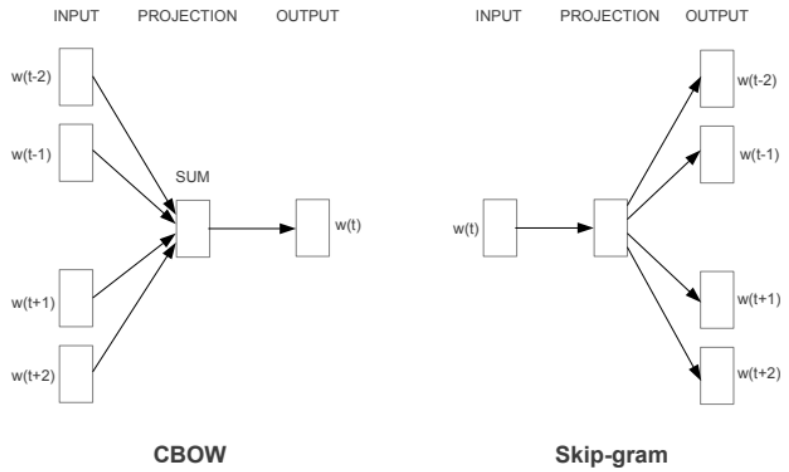


FIGURA 2.8: Representación gráfica de CBOW y Skip-gram.
Fuente: Cornerll University [34].

Finalmente, para obtener los vectores de incrustación para una determinada palabra, se multiplica la matriz de pesos de la red neuronal definida con su respectivo vector *one-hot encoding*.

FasText

Por otro lado, FastText [36] es una metodología desarrollada por Facebook que puede generar incrustaciones para las palabras que no aparecen en un corpus de entrenamiento debido al uso de n-gramas. Si una determinada palabra no se encontrara en la base de datos de entrenamiento, pero si mantiene

una representación vectorial de todos sus n-gramas, entonces podemos calcular su representación vectorial promediando la representación vectorizada de todos sus n-gramas constituyentes.

En este contexto, Sent2vec combina el enfoque CBOW planteado en el algoritmo Word2Vec junto con la técnica de FastText de usar n-gramas constituyentes para construir la incrustación de oraciones. Este algoritmo, amplía la metodología CBOW para definir los vectores de incrustación de oraciones como el promedio de las vectores las palabras de contexto presentes en la oración [6].

Según los autores de [6], la formula mostrada en la ecuación 2.6 se define que dada una oración S se calcula su vector de *embedding* v_S como el promedio de los vectores de *embeddings* de las palabras de contexto v_w de sus palabras constituyentes; donde $R(S)$ es la lista de n-gramas incluyendo *unigramas* presentes en la oración S.

$$v_S := \frac{1}{|R(S)|} Vl_{R(S} = \frac{1}{|R(S)|} \sum_{w \in R(S)} v_w \tag{2.6}$$

En este contexto, el modelo utiliza un método de entrenamiento no supervisado simple pero eficiente para entrenar representaciones distribuidas de oraciones. El algoritmo supera a los modelos no supervisados de última generación en la mayoría de las tareas de referencia e incluso supera a los modelos supervisados, destacando la solidez de las incrustaciones de oraciones producidas y su baja complejidad computacional.

2.3.2. Incrustación de oraciones utilizando Sentence Transformer

Desde la introducción de los modelos *transformer* en el año 2017, se ha dado un gran avance en el campo de NLP dejando atrás las limitaciones generadas por las arquitecturas que incluían Redes Neuronales Recurrentes (RNN, por sus siglas en inglés *Recurrent Neural Networks*). Este cambio se hizo posible gracias a

que los vectores de incrustación de palabras creadas por *transformers* son mucho más ricos en información; incrementando el rendimiento de arquitecturas incluso si sus últimas capas son igual que los modelos clásicos [37].

El autor de [37] menciona que la arquitectura *transformer* basa su funcionamiento en 3 conceptos base: (i) Codificación posicional (*positional encoding*); (ii) autoatención (*self-attention*); y (iii) atención multicabecera (*multi-head attention*). Además, con los modelos de *transformers* es posible usar el mismo núcleo y simplemente intercambiar las últimas capas para diferentes casos de uso (sin volver a entrenar el núcleo). Esta última propiedad ha permitido el surgimiento de modelos preentrenados para NLP. Estos modelos se entrenan con grandes cantidades de datos de entrenamiento como Google u OpenAI. Posteriormente, son liberados para que el público los use de forma gratuita.

En la actualidad, los modelos de red *transformer* (BERT [38], por sus siglas en ingles *Bidirectional Encoder Representations from Transformers*) y (RoBERTa [39], por sus siglas en ingles *Robustly Optimized BERT Pretraining Approach*), se han colocado como el estado de arte del NLP en tareas de clasificación y regresión. Sin embargo, tiene serias limitaciones para tareas de evaluación de similaridad semántica de oraciones debido a su limitación para generar vectores para oraciones. BERT usa una estructura de codificador cruzado para trabajar con oraciones y, a su vez, esto requiere que ambas oraciones ingresen a la arquitectura produciendo una sobrecarga computacional masiva. Por ejemplo, para 10000 oraciones requiere de 50 millones de cálculos de inferencia demorando 65 horas aproximadamente [5].

En el año 2019 Nils Reimers e Iryna Gurevychse presentaron el modelo *Sentence-BERT* (SBERT), una modificación de la red preentrenada BERT que utiliza una estructura de red siamesa y triplete para generar vectores de incrustación de oraciones de alta calidad que pueden ser comparados utilizando la similitud de coseno. Este nuevo modelo solucionó el problema inicial de similaridad semántica [5]. A partir de ese estudio inicial se implementó la biblioteca `Sentence Transformers` y se han desarrollado

más modelos de *transformers* para oraciones en diferentes idiomas utilizando conceptos similares a los usados para entrenar el SBERT original [37].

En la Figura 2.9 se puede observar la arquitectura SBERT aplicado a dos oraciones A y B. Inicialmente el modelo BERT genera vectores de *embedding* para los *tokens*. En total se tiene 512 vectores de dimensión 768 y posteriormente utilizando la capa de *pooling* se comprimen en un solo vector de dimensión 768.

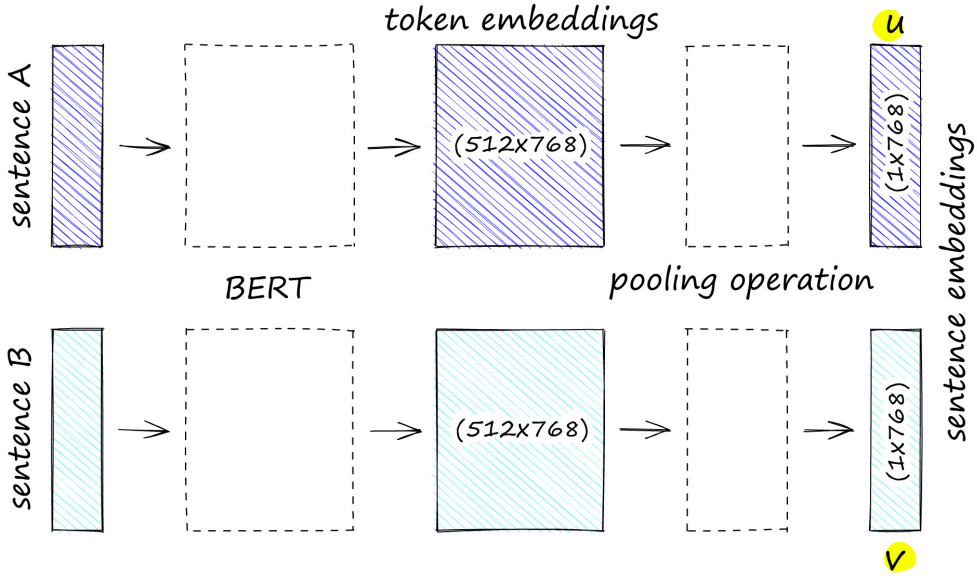


FIGURA 2.9: Modelo SBERT aplicado a un par de oraciones. Fuente: Picone.io [37].

Entrenamiento de Sentence Trasnformers

Utilizando la red siamesa se tiene dos formas de entrenar la arquitectura dependiendo del enfoque y objetivo planteado. La primera forma es implementar una arquitectura con una función objetivo de clasificación. La segunda es implementando una arquitectura para computar puntajes de similaridad. Dado que el entrenamiento del codificador para el presente proyecto se realiza utilizando la segunda forma, ésta se describirá más a detalle.

Durante el entrenamiento la función de pérdida juega un papel fundamental, sin embargo, hay muchas formas de definirla. La función de pérdida adecuada depende de los datos de entrenamiento disponibles y de la tarea objetivo [37]. Para el caso mostrado en la Figura 2.10, se define una

puntuación que indique la similitud entre dos oraciones en una escala de 0 a 1. Para cada par de oraciones, se ingresan las oraciones A y B a través de la red; como resultado se generan los vectores u y v. La similitud de estas incrustaciones se calcula utilizando la medida de similitud de coseno.

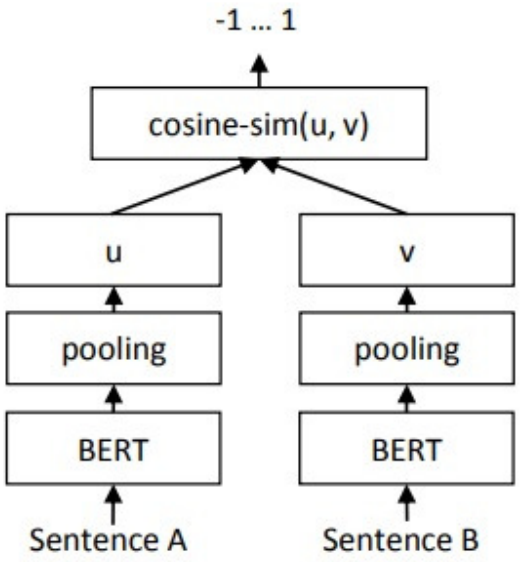


FIGURA 2.10: Arquitectura de entrenamiento siamesa orientada a evaluar similaridad. Fuente: Association for Computational Linguistics [5].

2.3.3. Modelo preentrenado RoBERTa para el idioma español

Como se mencionó en el apartado anterior, una de los mayores beneficios de los *transformers* es la implementación de modelos preentrenados para ser usados como punto de partida para entrenamiento en nuevos proyectos.

El modelo `RoBERTa-large-bne` ha sido preentrenado utilizando el corpus español más grande conocido hasta la fecha, el de la Biblioteca Nacional de España[2], con un total de 570 GB de texto limpio y procesado para este trabajo el cual fue compilado a partir de rastreos web entre los años 2009 y 2019 [10].

[2] `https://www.bne.es/es`

El corpus inicial de 2TB ha sido preprocesado realizando procedimientos de división de oraciones, detección de lenguaje, filtrado de oraciones mal formadas y duplicación de contenidos repetitivos. Durante el proceso se mantienen los límites del documento. Al final, se tiene 570 GB de información compuesta por 201,080,084 documentos hasta la actualidad. El preentrenamiento del modelo se realizó utilizando utilizando el enfoque de `RoBERTa-large` y tuvo una duración total de 96 horas con 32 nodos de cómputo cada uno con 4 GPUs NVIDIA V100 de 16GB VRAM [10].

Capítulo 3

Estado del Arte

En el presente capítulo, se realizará una descripción de trabajos relacionados con la generación de imágenes a partir de descripción textual utilizando redes GAN y, más específicamente, la generación de rostros. Dichos estudios nos servirán como punto de partida y contribuirán al desarrollo del presente trabajo, el cual abarca la solución de problemas en varias áreas de estudio. Por tanto, en esta sección nos enfocaremos en las 3 principales áreas en las que se enmarca esta tesis: (i) Incrustación de oraciones; (ii) Generación de imágenes guiada por texto; y (iii) Generación de rostros guiada por texto.

3.1. Incrustación de oraciones

La incrustación de oraciones es un tema ampliamente estudiado en el área de la inteligencia artificial, producto de ello, en la actualidad existen numerosos algoritmos y/o métodos han contribuido eficientemente en lograr avances muy novedosos. Esa técnica surgió como una ampliación de los métodos de incrustación de palabras en donde destacan *one-hot encoding*, Word2vec o FastText.

En este contexto, en 2015 surge el algoritmo *Skip-Thought* [40] que plantea el entrenamiento de una arquitectura *encoder-decoder* para predecir oraciones circundantes a partir de un vector de *encoder*. Dicha arquitectura comprende tres partes: (i) Red de codificadores, para generar el vector codificado; (ii) Red

de decodificadores anteriores, para formar la oración anterior a partir de un vector de incrustación; y (iii) Red de decodificadores posteriores, para formar la oración siguiente. Las tres redes están formadas por una red Unidad Recurrente Cerrada (GRU, por su siglas en inglés *Gated recurrent unit*) o una red de Memoria de Corto-Largo Plazo (LSTM, por sus siglas en inglés *Long-Short Term Memory*). Como resultado, las representaciones de oraciones semánticamente similares también son similares; sin embargo, esta sujeto a posibles errores de precisión a la hora de predecir las oraciones circundantes.

Así, en el año 2017 se desarrolló el algoritmo **Infersent** [41] que superaba los resultados de la mayoría de métodos no supervisados como *Skip-Thought*. Su objetivo consiste en encontrar una relación direccional entre fragmentos de texto a través de implicaciones textuales bajo etiquetas. La arquitectura esta compuesta por: (i) Un codificador de oraciones que toma vectores de palabras y genera su respectivo vector codificado; y (ii) Un clasificador Inferencia del lenguaje Natural (NLI, por sus siglas en inglés *Natural Language Inference*) que toma los vectores codificados y genera una etiqueta de clase que puede ser implicación, contradicción y neutral. Para el entrenamiento se utilizó el conjunto de datos de *Stanford Natural Language Inference*[1] (SNLI) consistente en 570000 pares de oraciones en inglés generadas por humanos y etiquetadas manualmente con una de las tres categorías antes mencionadas. Resalta el uso de redes LSTM bidireccionales para la implementación del codificador, esta arquitectura genera un conjunto de vectores concatenados los cuales se pasan por una capa *max-pooling* para formar el vector final de longitud fija.

Continuando con el desarrollo de nuevos modelos, en el año 2018 se presentó otro modelo denominado *Universal Sentence Encoder* [42], el cual se entrena una red *transformer* o una *Deep Averaging Network* (DAN) y se aumenta el aprendizaje no supervisado entrenando con la base de datos *SNLI*. Dicho trabajo demostró que la tarea en las que se entrenan a los *encoders* afecta significativamente su calidad. La idea central consiste en implementar un codificador que será empleado para resolver múltiples tareas de NLP como

[1] `https://nlp.stanford.edu/projects/snli/`

clasificación de texto, detección de parafraseo y agrupamiento. Basado en los errores en estas, se actualiza los pesos del codificador. El resultado permite convertir casi cualquier oración en un vector numérico de longitud 512.

El algoritmo Sent2vec desarrollado en el año 2019, combina las técnicas y fundamentos de otros algoritmos como Word2vec y FastText combinando el enfoque CBOW y usando n-gramas constituyentes para generar el vector resultado [6]. Durante el entrenamiento se utiliza un método no supervisado simple, eficiente y, sobre todo, rápido debido a su baja complejidad computacional. Los resultados del entrenamiento muestran que supera los resultados de la mayoría de modelos no supervisados y supervisados, destacando la solidez de los resultados.

En el mismo año 2019, se implementó el codificador Sentence-BERT basado en *transformers* el cual permite generar incrustación de oraciones de alta calidad en comparación de los modelos antes estudiados. Los modelos BERT tradicionales son denominados el estado del arte del NLP, sin embargo, presentan limitaciones para tareas de similaridad semántica debido a su dificultad para crear vectores de codificación de oraciones por sobrecarga computacional durante el entrenamiento. Para solucionar dicho inconveniente los autores Nils Reimers e Iryna Gurevychse [5] desarrollaron Sentence-BERT el cual genera incrustación de oraciones utilizando una red siamesa y triplete durante el entrenamiento. La arquitectura basa su funcionamiento en tres conceptos: (i) Codificación posicional, (ii) Auto-atención y (iii) Atención multicabecera. Además, al igual que cualquier otro *transformer* permite utilizar modelos preentrenados de NLP sobre los cuales se puede aumentar el rendimiento mediante entrenamiento con nuevos corpus de texto. Esta última característica abre paso a múltiples estudios de mejora de rendimiento y aplicación en futuros trabajos.

3.2. Generación de imágenes guiada por texto

Todos los trabajos relacionados a generación de imágenes a partir de texto se enfocan en el desarrollo y perfeccionamiento del codificador de texto y la GAN condicional. Las investigaciones se centran en mejorar la calidad de las imágenes.

GAN

El primer trabajo relacionado a generación de imágenes guiadas por texto se desarrolló en el año 2016 por Scott Reed et al., presentaron el trabajo denominado *Generative Adversarial Text to Image Synthesis* [43]. En este trabajo desarrollan una novedosa arquitectura profunda y una formulación GAN para unir de manera efectiva los avances en el modelado de texto e imágenes, traduciendo conceptos visuales de caracteres a píxeles.

El modelo utiliza como codificador a una red neuronal usando una capa completamente conectada a una dimensión pequeña (comúnmente 128), seguido de Leaky-ReLU; mientras que la GAN está compuesta por un generador de múltiples capas con Leaky-ReLU y un discriminador con múltiples capas con ReLU y *Batch normalization*.

El modelo trabaja con los conjuntos de datos de prueba *Common Objects in Context* (COCO)[2] y *Caltech-UCSD Birds* (CUB)[3].

StackGAN

En el año 2017 Han Zhang et al.[21], presentaron la arquitectura denominada StackGAN la cual es capaz de generar imágenes de dimensión 256×256 condicionadas a oraciones descriptivas pasadas al modelo. La novedad de este modelo es que divide la tarea total en un conjunto de subtareas más fáciles de realizar a través de un proceso de refinamiento de bocetos.

En la denominada etapa I esboza la forma y los colores primitivos del

[2] `https://cocodataset.org/#home`

[3] `https://paperswithcode.com/dataset/cub-200-2011`

objeto en función de la descripción de texto, lo que genera imágenes de baja resolución. En la etapa II, toma los resultados de la etapa anterior y, junto a las descripciones de texto como entradas, genera imágenes de alta resolución con detalles de alta calidad. En esta etapa se puede rectificar defectos en los resultados de la etapa I y agregar detalles convincentes con el proceso de refinamiento.

Al igual que el trabajo anterior, se realizaron las pruebas de eficiencia utilizando los conjuntos de datos de imágenes COCO, CUB y Oxford-102[4].

AttnGAN

El modelo AttnGAN [28] desarrollado ese mismo año por los investigadores Tao Xu et al., representa una mejora en el modelo StackGAN en el refinamiento de múltiples etapas. Este modelo puede sintetizar detalles especificos en una cantidad mucho mayor de diferentes pequeñas subregiones de la imagen, prestando atención a las palabras relevantes en la descripción. Además, se proponen un Modelo de Similitud Multimodal de Atención Profunda (DAMSM, por sus siglas en inglés *Deep Attentional Multimodal Similarity Model*), para calcular una pérdida de coincidencia de imagen-texto de para entrenar al generador.

Este nuevo modelo aprende de dos redes neuronales que asignan subregiones de la imagen y las palabras de la oración a un espacio semántico común. Esta novedosa técnica permite medir la similitud de imagen y texto a nivel de palabra para calcular una pérdida de grano fino para la generación de imágenes. El codificador utilizado, es una red LSTM que extrae vectores semánticos de la descripción del texto. En la red LSTM bidireccional, cada palabra corresponde a dos estados ocultos, uno para cada dirección. De esa forma, se concatenan sus dos estados ocultos para representar el significado semántico de una palabra.

Las pruebas se realizaron utilizando los conjuntos de datos clásicos COCO y CUB, y se observó que brinda mejores resultados que los estudios de su estado

[4] https://www.robots.ox.ac.uk/~vgg/data/flowers/102/

del arte.

MirrorGAN

En el año 2019 los autores Tingting Qiao et al. [23], presentaron el modelo MirrorGAN el cual logra un realismo visual y coherencia semántica mucho más altos que los modelos antes estudiados. La arquitectura planteada aprovecha la idea de aprender la generación de texto a imagen mediante la descripción y consta de tres módulos: (i) Un Módulo de Incrustación de Texto Semántico (STEM, por sus siglas en inglés *Semantic Text Embedding Module*) que utiliza una red RNN; (ii) Un Módulo para la Generación de Imágenes en Cascada (GLAM, por sus siglas en inglés *Global-Local collaborative Attentive Module in Cascaded Image Generators*); y (iii) Un Módulo de Alineación y Regeneración de Texto Semántico (STREAM, por sus siglas en inglés *Semantic Text Regeneration and Alignment Module*).

En este sentido, el módulo STEM genera vectores de incrustaciones a nivel de palabra y oración. Por otro lado, el módulo GLAM tiene una arquitectura para generar imágenes de destino desde las escalas más gruesas a finas. Aprovechando tanto la atención de palabras locales como la atención de oraciones globales para mejorar progresivamente la diversidad y la consistencia semántica de las imágenes generadas, STREAM busca regenerar la descripción del texto a partir de la imagen generada; que se alinea semánticamente con la descripción del texto dada.

El rendimiento de la red se calculó utilizando los conjuntos de datos COCO y CUB. Los autores señalan que supero los valores calculados en su estado del arte.

3.3. Generación de rostros guiada por texto

Un problema derivado de la síntesis de imágenes guiada por texto es la síntesis de rostros. La elaboración de imágenes de personas es mucho más compleja y sujeta a errores que otros tipos de imágenes. Por ello, recientemente

se han desarrollado varios modelos y aplicaciones enfocados en esta tarea, los cuales en esta sección se describirán los que han tenido un mayor impacto.

En el año 2018 Diogo Rodrigues desarrollo el trabajo denominado *Integrating Vision and Language for Automatic Face Descriptions* [44]. Este trabajo consiste en la creación de una aplicación que genera imágenes a partir de su descripción textual y también su proceso inverso (generación de texto descriptivo a partir de una imagen), utilizando para ambos objetivos técnicas de visión por computadora y NLP. El objetivo fue implementar un sistema que pueda utilizarse, por ejemplo, para describir rostros de personas con discapacidad visual o generar rostros a partir de descripciones para investigaciones penales [44].

El desarrollo del trabajo de Diogo Rodrigues fué realizado en dos partes: (i) La primera parte predice atributos de las imágenes faciales a través de un método de CNN, además, como base para el modelo de Generación de Lenguaje Natural (NLG, por sus siglas en inglés *Natural Language Generation*) en el que las descripciones se generan con una metodología basada en reglas; y (ii) La segunda parte del sistema utiliza una técnica de extracción de palabras clave simple para analizar el texto e identificar los atributos en esa descripción. Después de ello, utiliza una GAN condicional para generar una imagen facial con un conjunto específico de atributos deseados. La razón por la que los atributos se utilizan como base en el método es porque son un identificador dominante que puede transmitir de manera eficiente las características de un rostro.

Para el procesamiento de texto de entrada, el trabajo resalta y aporta el desarrollo de un modelo que analiza el contenido lingüístico y reconoce las similitudes semánticas de las posibles palabras clave seleccionadas para el modelo. El modelo en primer lugar, hace una simplificación del texto descripción mediante la eliminación de palabras sin significado (*stopwords*). Posteriormente, se identifican los atributos presentes los cuales no son necesariamente la misma palabra. Seguidamente se hace coincidir la lista de palabras candidatas junto a la lista de palabras similares y el conjunto de

palabras de la lista de atributos. Finalmente, el modelo genera 40 valores binarios. Un valor positivo significa que se identificó positivamente el atributo de la descripción del texto. El resto de atributos se asignan como faltantes y, por tanto, se les atribuyen valores negativos.

Como se observa en la Figura 3.1, el generador recibe, además del ruido aleatorio, información adicional sobre las características de la imagen. La información adicional es un vector con valores binarios que identifica los 40 posibles atributos que se pueden presentar en esa imagen de rostro. Por otra parte, el discriminador arroja dos resultados: (i) Uno relacionado con la decisión que representa si la imagen se considera real o falsa; así como (ii) Un vector correspondiente a la predicción de los 40 atributos que caracterizan la imagen del rostro. Los autores de la investigación afirman que el modelo obtiene buenos resultados para el conjunto de datos CelebA.

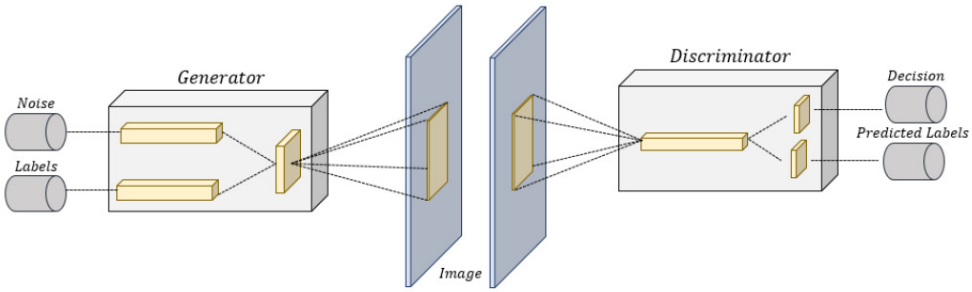


FIGURA 3.1: Esquema general de la arquitectura propuesta.
Fuente: Diogo Rodrigues [44].

Por otro lado, en el año 2019 se presenta el trabajo de J. Zhao et al. [45], denominado *Generating Photographic Faces From the Sketch Guided by Attribute Using GAN*. Este trabajo brinda un enfoque diferente a la forma tradicional de generar rostros condicionados, dado que utilizan bocetos en vez de imágenes completas y, junto a la descripción textual, son las entradas del generador. Los principales aportes que ofrece la investigación se resumen en:

- Se determina que el proceso de completado del boceto para generar la imagen real es similar al proceso de reconstrucción de superresolución de

rostros. Por tanto, para afrontar el problema cambiaron el enfoque y se desarrolló una red neuronal generativa para tal finalidad.

- La fase de extracción de características es diferente a la mayoría de redes de ese tipo. El método común consiste en expandir la dimensión de los atributos y modificar su tamaño. Posteriormente son concatenadas a la salida de las capas de convolución. Para la nueva arquitectura planteada, las características semánticas son extraídas del vector de atributos y se redimensionan al mismo tamaño que el boceto de la imagen ingresada.

- Utilizando el método *Skip-connection* [43], se logró reducir el número de capas de la red generativa sin perder la eficiencia de la misma debido a que actualmente la tendencia es mejorar el rendimiento aumentando el número de capas. Sin embargo, esto implica que aumente la complejidad computacional y, por ende, la aparición de diversos problemas en su ejecución.

- La arquitectura planteada utiliza una estructura de subtramas (A para entrada de imágenes y B para entrada de atributos) en la fase de extracción de características. Por lo general, la mayoría de algoritmos de generación de imágenes solo utilizan la rama A de imagen a imagen o la rama B de texto a imagen. Como resultado de las pruebas hechas se observó que, la combinación de las dos tramas permite que la imagen de la cara generada se comporte mejor en términos de textura y color; además, se tiene una estructura más sutil.

En este contexto, como se observa en la Figura 3.2, la arquitectura planteada está compuesta de dos partes: El generador y el discriminador. La entrada al generador presenta dos ramas, la rama A por la cual se ingresa el boceto dibujado y la rama B por la cual se ingresa un vector de atributos. La imagen del rostro resultante se genera dentro del módulo generador y se define la función de pérdida de reconstrucción como la diferencia con respecto a la cara real, la cual se encuentra en el discriminador [45].

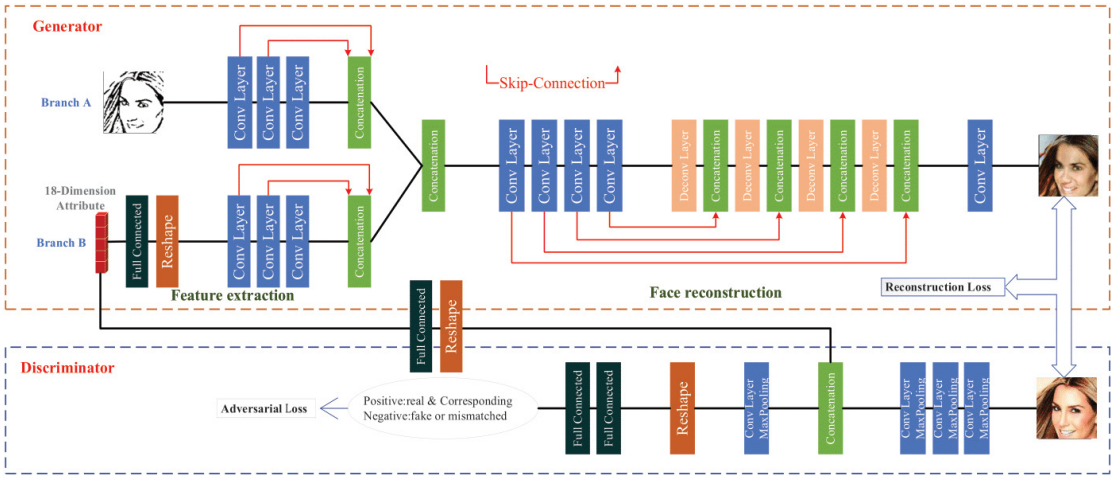


FIGURA 3.2: Esquema general de la arquitectura propuesta.
Fuente: IEEE [45].

Ese mismo año Osaid Rehman Nasir et al. implementaron una arquitectura GAN condicional denominada Text2faceGAN [46, 47] la cual, genera imágenes de rostros a partir de descripción textual de grano fino. Para las pruebas se utilizaron las imágenes de CelebA junto con un conjunto de textos descriptores implementados para cada imagen utilizando un algoritmo propio. Esta investigación es una de las primeras en donde se hizo uso de un modelo/algoritmo codificador para la incrustación de oraciones enteras, para este proyecto se uso Skip-Thought. Los dos trabajos antes mencionados utilizaban descripciones de una sola palabra o algún algoritmo propio. La principal contribución hecha por ese trabajo se centra en la preparación de los datos para el entrenamiento, la arquitectura de red GAN planteada y la métrica de calidad IS definida para evaluar sus resultados. Como parte de su implementación describen los siguientes objetivos logrados:

- Implementación de un corpus en español para las imágenes de CelebA.
- Desarrollo de una arquitectura CDCGAN para generar rostros a partir de descripciones textuales bien detalladas.
- Implementación de IS como métrica de evaluación de calidad y sus limitaciones al usarlo.

Como se observa en la Figura 3.3, la arquitectura esta compuesta por una serie de capas convolucionales y convolucionales transpuestas normalizadas con funciones de pérdida ReLU y LeakyReLU, además se utiliza capas densas, de activación y concatenación.

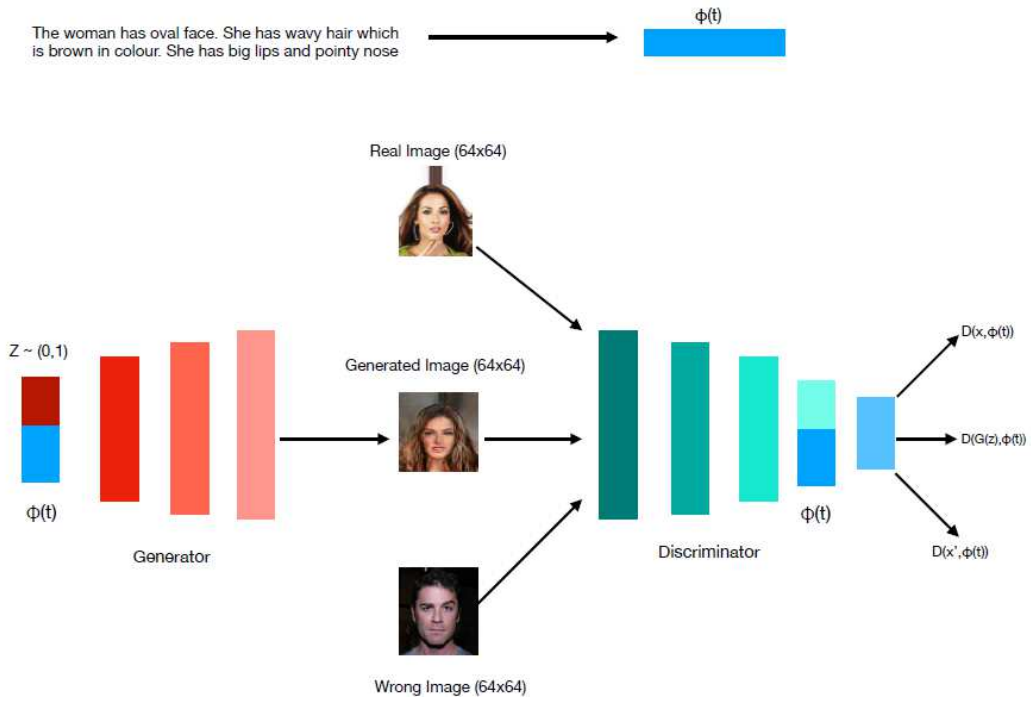


FIGURA 3.3: Arquitectura general de Text2FaceGAN. Fuente: 2019 IEEE [46].

Al año siguiente Ryan Cain et al. [48] crearon una aplicación web por la que los usuarios pueden escribir una descripción de un rostro humano en formato texto plano y, a partir de ello, se generen fotos que se asemejen a su descripción. Para ello, utilizan la arquitectura GAN condicional denominada *Conditional Wasserstein Generative Adversarial Network* (CWGAN). En relación a los resultados generados por el estudio resaltan dos hechos. Lo primero, consideran que es muy dificultoso medir el rendimiento o calidad de la producción debido a que, por ejemplo, una misma descripción podría describir varios rostros y un rostro podría aparecer como resultado de varias descripciones. El segundo hecho a resaltar es que notaron que la calidad de imagen generada depende de la cantidad de imágenes con la misma descripción en el conjunto de entrenamiento; por ejemplo, 11.48 % de imágenes

contenía test oscuro y sonreía por lo que se generaban imágenes de buena calidad con esa descripción.

En la Figura 3.4, se puede observar de manera general el diagrama de actividad o funcionamiento de la aplicación. A través de la interfaz web se ingresan datos característicos deseados como color de cabello, color de piel, forma de los ojos, etc. Como observación del trabajo, se menciona que a raíz de que se utilizó una arquitectura relativamente simple y fue entrenado durante periodos de tiempo cortos, la calidad de las imágenes obtenidas no era del todo aceptables en muchos casos por lo que se recomienda probar con arquitecturas más complejas y durante periodos de tiempo más grandes. Así también, incluir un modelo de texto embebido para codificar los datos pasados al modelo.

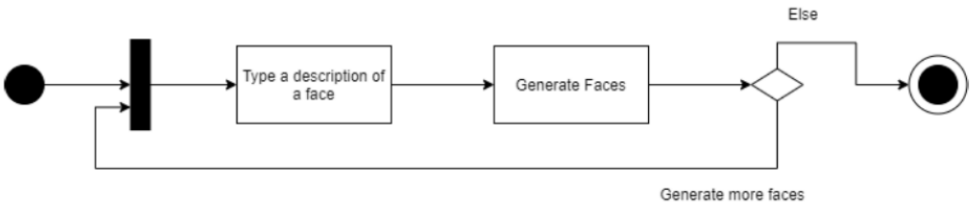


FIGURA 3.4: Diseño de actividad de la aplicación. Fuente: Ryan Cain et al. [48].

Por otro lado, en el año 2021 los autores Xia et al. [49] desarrolaron un modelo denominado *TediGAN: TextGuided Diverse Face Image Generation and Manipulation* en la cual presentaron un novedoso framework para la generación y manipulación de imágenes con descripciones textuales denominado TediGAN, el cual esta compuesto por 3 elementos. Un módulo StyleGAN que se encarga de asignar imágenes reales a un espacio latente. El módulo de similitud visual-linguistica que se encarga de identificar similitudes entre texto e imágen al mapearlos en un espacio de *embbeding* común. Finalmente un módulo de optimización a nivel de instancia para la preservación de la identidad (texto o imágen) durante manipulaciones. El trabajo presenta las siguientes contribuciones:

- Propone un marco completo que permite generar diversas imágenes a

partir de un texto de entrada, o texto con imagen para su manipulación de forma interactiva.

- Desarrolla una técnica de inversión de GAN que puede mapear información multimodal (varios tipos) en un espacio latente común de un StyleGAN preentrenado, donde se puede aprender la alineación de imagen y texto.

- Genera un nuevo conjunto de datos derivado de CelebA denominado CelebA-HQ, el cual consta de imágenes de rostros reales y el correspondiente mapa de segmentación semántica, boceto y descripciones textuales.

Adicionalmente, como parte de su investigación los autores hicieron pruebas comparativas con otros modelos como DFGAN y DMGAN cuyos valores de métricas FID y LPIPS son mostrados en la tabla 3.1.

Ese mismo año, los autores Manan Oza et al. [50] plantean una arquitectura única denominada *Semantic TexttoFace GAN*. Esta arquitectura puede generar caras a partir de entradas de texto genérico y permita modificarlas a partir de otra entrada de texto auxiliar empleando la misma red. La arquitectura planteada está compuesta de 2 partes importantes y complementarias. La primera parte de la arquitectura, genera imágenes de baja resolución a partir de una descripción de texto codificado. La segunda parte recibe como entrada el texto descriptivo y la imagen en baja resolución para generar el resultado final en alta resolución. Una característica a resaltar, es el uso de un modelo preentrenado base de SBERT para realizar la incrustación de oraciones en el idioma inglés.

Así, como se observa en la Figura 3.5, la primera parte de la arquitectura consta de una red *encoder-decoder* compuesta por un bloque ResNet [51] y una estructura de Módulo de combinación afín (ACM, por sus siglas en inglés *Affine Combination Module*) para unir entradas. Dicha red es entrenada utilizando imágenes e incrustaciones de texto generadas por SBERT. El objetivo es entrenar al codificador para mapear la imagen real y obtener el espacio latente de las

imágenes. La representación producida por el codificador se puede recuperar tanto a nivel de píxel como semántico. Posteriormente, el modelo tiene como objetivo generar imágenes de baja resolución utilizando incrustaciones de texto a partir del espacio latente de las imágenes aprendidas.

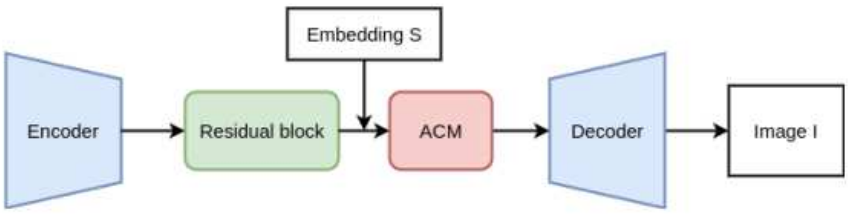


FIGURA 3.5: Primer componente de la arquitectura ST2FG, una red para obtener un espacio latente. Fuente: Manan Oza et al. [50]

Como se puede observar en la Figura 3.6, el segundo componente es una red GAN. El generador codifica la imagen de entrada en baja resolución en un vector de características utilizando la red Inception-v3 [52] y luego las concatena con vectores de características textuales codificadas por SBERT. Los vectores de texto codificados también alimentan una serie de bloques residuales y de muestreo ascendente. Finalmente, se tiene una red generadora de imágenes para producir el resultado modificado con la resolución deseada. Por otro lado, el discriminador recibe dos entradas. La primera, es la unión de los vectores de características de la imagen de entrada en baja resolución y la obtenida por el generador, ambos obtenidos utilizando Inception-v3. La segunda entrada es el vector de texto codificado generado por el *transformer*. El modelo ST2FG fue entrenado utilizando dos conjuntos de datos de imágenes, con la finalidad de tener más variedad de entradas y formatos de texto. Se entreno con Multi-Modal-CelebA-HQ [53, 54] durante 500 épocas y con CelebA durante 350 épocas. Para la evaluación, se genero un conjunto de datos propio con 250 imágenes y 50 descripciones textuales. Utilizando las métricas FID, LPIPS, *Accuracy* y *Photorealism* se compararon los resultados con otros modelos como son TediGAN [49], AttnGAN, ControlGAN [55] y Text2FaceGAN; observándose los mejores resultados en cada métrica. Dichos

resultados son mostrados en la Tabla 3.1.

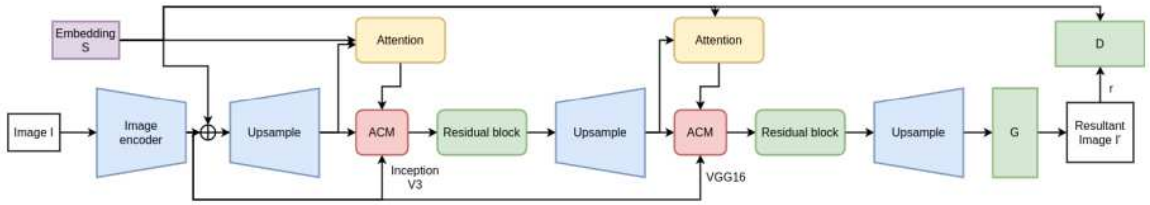


FIGURA 3.6: Segundo componente de la arquitectura ST2FG, una red para mejorar la calidad de las imágenes. Fuente: Manan Oza et al. [50]

Finalmente, en el año 2022 se presentó el trabajo de investigación denominado *FGTD: Face generation from textual description* [56] realizado por Kalpana Deorukhkar et al., en donde se realiza una comparación de tres modelos GAN en su desempeño para la generación de rostros sintéticos utilizando las métricas FID y IS, los resultados son mostrados en la Tabla 3.1. Los algoritmos evaluados son DCGAN, *Self Attention Generative Adversarial Network* (SAGAN) [57] y *Deep Fusion Generative Adversarial Networks* (DFGAN) [58]. Para realizar el entrenamiento de los modelos, se utilizó el conjunto de datos CelebA en conjunto con un corpus de texto descriptivo de las imágenes que lo componen. Al igual que en el trabajo del modelo Text2FaceGAN, los autores crearon su propio corpus de oraciones descriptivas utilizando un algoritmo propio a partir de los atributos los cuales son listados en el conjunto de datos. Finalmente, utilizan un modelo preentrenado de SBERT en el idioma inglés para codificar las oraciones descriptivas de las imágenes del conjunto de datos. En el trabajo se verifica que SBERT brinda mejores resultados en comparación al modelo original BERT y a otros algoritmos como Skip-Thought, a la hora de crear imágenes de mejor calidad.

Como se puede observar en la Figura 3.7, los autores implementaron una metodología de entrenamiento sencilla y bastante ilustrativa en donde se definen claramente los pasos a seguir en el entrenamiento y evaluación de cada uno de los modelos trabajados. Estos pasos se resumen en 4 puntos

importantes: [56]

- El texto descriptivo correspondiente al par texto-imagen es codificado en un vector de características usando SBERT.
- Los vectores semánticos son ingresados junto a su respectiva imagen a cada una las arquitecturas evaluadas.
- Después de cada época, las pérdidas son monitoreadas para evaluar el desempeño del entrenamiento.
- Una vez entrenada la red se procede a evaluar los resultados utilizando las siguientes métricas: FID, IS y Clean FID [59].

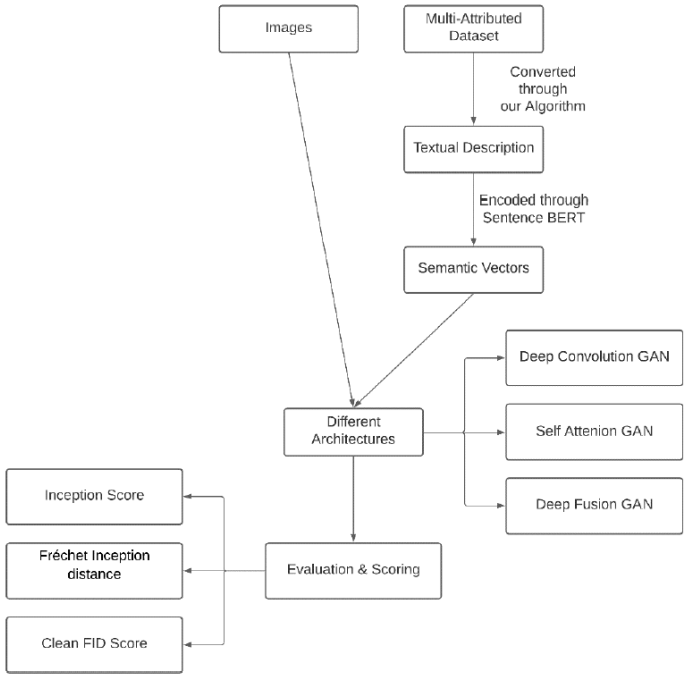


FIGURA 3.7: Diagrama de flujo del la metodología de entrenamiento. Fuente: Springer [56].

3.4. Discusión

Dividir el estudio de los trabajos presentados en estado de arte en tres áreas principales nos ha permitido enfocarnos en la evaluación y mejoras que se han

ido agregando con cada investigación reciente y conocer las diferentes ideas con las que abordaron cada problema y sus mejoras; las cuales pueden aplicarse al presente trabajo.

3.4.1. Incrustación de oraciones

En el área de incrustación de oraciones todos los trabajos estudiados abordan el problema de manera distinta. En este sentido, los autores de [40] utilizan un codificador y dos decodificadores, entrena con el objetivo de predecir oraciones vecinas a una oración base codificada y se recorre todo el corpus de entrenamiento. Los autores de [41] entrena el codificador ingresando el vector generado en una red clasificadora y comparando el resultado con unas etiquetas de clase definidas. En [42] entrena el codificador empleándolo en diferentes tareas de NLP. Por otro lado, los autores de [6] combina y expande los enfoques de Word2vec y FastText, inicialmente diseñado para palabras, para generar vectores de oraciones completas. En este estudio resalta la facilidad y rapidez de entrenamiento en comparación a los demás. Finalmente, los autores de [5], al hacer uso de *transformes*, consigue los mejores resultados hasta la fecha en rendimiento y calidad de incrustación de oraciones. Siguiendo la linea del uso de *transformers*, en el presente trabajo se mejorará el rendimiento del modelo base RoBERTa entrenándolo con el corpus descriptivo del conjunto de datos CelebA en idioma español.

3.4.2. Generación de imágenes guiadas por texto

La generación de imágenes guiadas por texto es una tarea especializada derivada de la tarea general de crear imágenes de diferentes tipos. Esta última área de estudio comenzó en 2016 y se han ido creando diferentes modelos GAN, los cuales tienen diferentes formas de codificar oraciones que han ido mejorando con el paso del tiempo. En [43] se utiliza una red neuronal básica como codificador y la resolución de imágenes es de 64×64. Posteriormente, en [21] y [28] mejoran la codificación del texto de entrada utilizando una red

LSTM; además, la resolución de las imágenes se incrementa a 256×256. En [21] utilizando dos fases de procesamiento y en [28] utilizando un modelo de similitud multimodal de atención enfocada en subpartes o regiones de imágenes prestando atención a las características relevantes. Finalmente, los autores de [23] utilizan una combinación de 3 módulos: una red RNN para codificar texto, una red para generar imágenes y un modulo de regeneración de texto descriptivo a partir de una imagen para que se alineen semánticamente.

3.4.3. Generación de rostros

Los trabajos enfocados a generación de rostros, al igual que en el área de estudio general, se centran en mejorar la forma en la que se codifica el texto descriptivo y el diseño de la GAN en sí. En [44] la información descriptiva se pasa a través de valores binarios sobre 40 posibles atributos, todavía no se utiliza un codificador de oraciones, resalta su arquitectura bidireccional para generar imágenes e identificar atributos. En [45] se plantea ingresar a la red un boceto inicial (*squetch*) adicionalmente al texto descriptor como entradas a la red; se utiliza un algoritmo propio para codificar oraciones. En [46, 47] se emplea un codificador denominado Skip-thought y una DCGAN basada en redes convolucionales, funciones de activación ReLU y LeakyReLU y *Batch normalization* según las recomendaciones básicas de mejoramiento de GAN para síntesis de imágenes. Finalmente, en [48] se implementa un generador simple con una interfaz web. Tras realizar varias pruebas destacan sus resultados que mencionan que es difícil medir el rendimiento de la red por la variedad de posibles rostros que se generan a partir de una sola descripción. Además, la calidad de imágenes depende de la cantidad de imágenes de entrenamiento con una misma característica o descripción específica. Se resalta que todos los trabajos se realizaron tomando al inglés como idioma base.

En los trabajos [50] y [56] se resalta el uso de tecnologías y modelos más recientes y complejos como BERT y SBERT para la codificación de oraciones descriptivas, RestNet para la obtención del espacio latente de imágenes e Inception-v3 para obtener vectores de características. En [50] se plantea una

arquitectura que permita generar imágenes y éstas se puedan modificara en base a otra descripción posterior en un segundo paso. En [56] se hace una variada comparación de eficiencia entre varias arquitecturas GAN conocidas utilizando FID e IS. Se resalta que todos los trabajos se realizaron tomando al inglés como idioma base.

3.4.4. Líneas de actuación

Tomando como referencia estos trabajos estudiados, el proyecto actual busca generar imágenes de rostros humanos a partir de una descripción textual en el idioma español. Por su facilidad de implementación y equilibrio entre rendimiento y complejidad, se tomará como arquitectura base a la implementada en [46, 47]. Por otro lado, para mejorar la eficiencia se utilizará un codificador basado en SBERT, el cual será previamente entrenado con el corpus del conjunto de datos CelebA. Se resalta el hecho de que el 100 % de trabajos revisados en el estado de arte fue desarrollado con el idioma inglés como principal; siendo el propósito de la tesis realizarlo en el idioma español.

Finalmente, la tabla 5.2 describe un resumen de los codificadores y modelos GAN utilizados por los diferentes trabajos estudiados en el estado del arte y, también, en la presente tesis. Se puede observar que los estudios más antiguos no utilizaban codificadores dedicados, por el contrario, empleaban arreglos de atributos directamente como en [44], [45] y [48] o redes neuronales como en [43]. Posteriormente, se emplearon redes más complejas como LSTM y RNN empleadas en [21], [28] y [23] respectivamente. A partir del modelo desarrollado en [46] en 2019 ya se observa el uso de codificadores propiamente dichos (Skip-Thought) y dichos modelos se han ido mejorando hasta llegar al uso de *transformers* como en [50] y [56], en esa línea dos de los modelos planteados en nuestro trabajo hacen uso de *transformers* como codificadores. En relación a los modelos GAN generadores, se observa que cada estudio crea un modelo propio, siendo la arquitectura cDCGAN personalizada la más empleada (hasta en 5 trabajos). Nuestro modelo generador es una variación del planteado en [46].

TABLA 3.1: Comparativa de recursos usados en los trabajos estudiados enfocados en generación de de imágenes y rostros. Las últimas tres filas se encuentran los detalles de los utilizados en esta tesis y señalado como *Este trabajo*. Fuente: Elaboración propia.

Trabajo	Encoder	GAN arq.	IS	FID	LPIPS	Imágenes	Idioma
Reed et al. [43]	NN	CDCGAN	–	–	–	–	Inglés
Zhang et al. [21]	LSTM	StackGAN	–	–	–	–	Inglés
Xu et al. [28]	LSTM	DAMSM	–	–	–	–	Inglés
Qiao et al. [23]	RNN	GLAM & STREAM	–	–	–	–	Inglés
Diogo [44]	40 Atributos	cDCGAN	–	–	–	–	Inglés
Zhan et al. [45]	18 Atributos	cDCGAN	–	–	–	–	Inglés
Cain et al [48]	9 Atributos	cWGAN	–	–	–	–	Inglés
Nasir et al [46, 47]	Skip-Thought	cDCGAN	1.4	–	–	–	Inglés
Xia et al. [49]	LSTM Bi.	DFGAN	–	137.600	0.581	50	Inglés
Xia et al. [49]	LSTM Bi.	DMGAN	–	131.050	0.544	50	Inglés
Oza et al. [50]	Skip-Thought	cDCGAN	–	128.460	0.590	250	Inglés
Oza et al. [50]	LSTM	DAMSM	–	125.980	0.512	250	Inglés
Oza et al. [50]	RNN	AttGAN	–	116.320	0.522	250	Inglés
Oza et al. [50]	Módulo similitud	StyleGAN	–	106.370	0.465	50	Inglés
Oza et al. [50]	BERT	ACM	–	105.730	0.449	250	Inglés
Deorukhkar et al. [56]	SBERT	cDCGAN	2.732	90.268	–	5	Inglés
Deorukhkar et al. [56]	SBERT	SAGAN	2.855	95.052	–	5	Inglés
Deorukhkar et al. [56]	SBERT	DFGAN	3.455	88.748	–	5	Inglés
Este trabajo	Sent2vec+ CelebA	CDCGAN	2.738	105.561	0.304	1000	Español
Este trabajo	RoBERTa	CDCGAN	2.714	89.357	0.293	1000	Español
Este trabajo	RoBERT+ CelebA	CDCGAN	2.789	**84.221**	**0.285**	1000	Español

Capítulo 4

Metodología e implementación

En este capítulo se define la metodología usada en la implementación del presente trabajo y se describen sus fases más importantes del desarrollo del mismo. Adicionalmente, se detalla el preprocesamiento aplicado al corpus descriptivo e imágenes del conjunto de datos CelebA. Finalmente, se describe la arquitectura de la red cDCGAN usada para generar las imágenes sintéticas.

4.1. Esquema general

Con la finalidad de lograr el objetivo principal y los objetivos específicos, se planteó una metodología simple, la cual es representada en el diagrama de flujo mostrado en la Figura 4.1, en ello se muestran cada uno de los pasos agrupados en colores de cuadros (azul, rojo, amarillo y verde) según el objetivo a lograr. La implementación total del trabajo se divide en 3 fases, estas son:

- Preprocesamiento de los datos de entrenamiento (cuadros azules).
- Entrenamiento del codificador (cuadros rojos).
- Entrenamiento de la red GAN (cuadro amarillo) y aplicación de las métricas de calidad (cuadros de color verde).

En base a las fases definidas, el conjunto de pasos de la metodología seguida se dividen en 3 grupos y son descritos a detalle en los siguientes apartados.

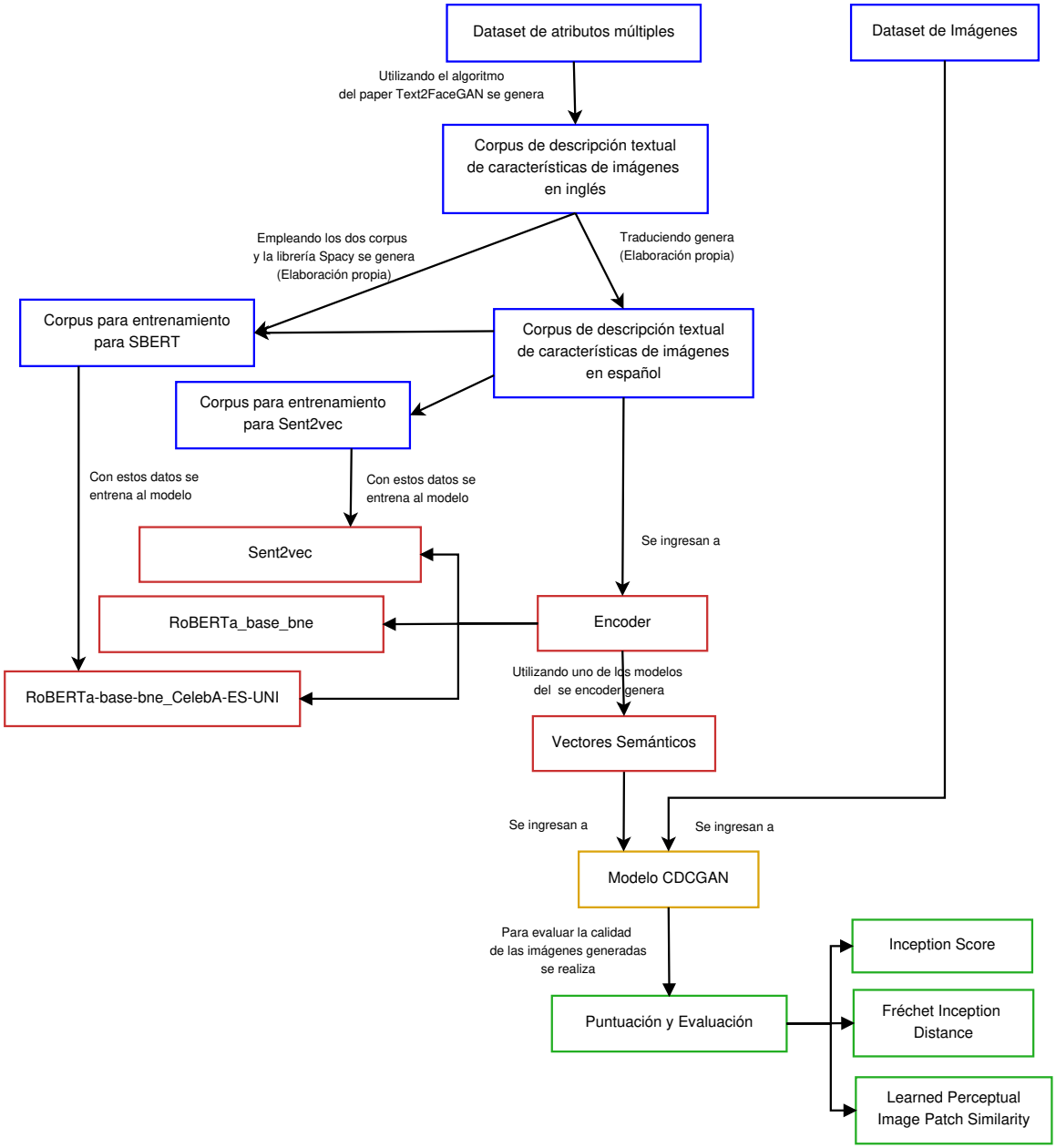


FIGURA 4.1: Diagrama de flujo de la metodología usada para el presente trabajo. Fuente: Elaboración propia.

4.1.1. Preprocesamiento de datos

La primera fase está compuesta por los pasos en cuadros de color azul en la Figura 4.1. A continuación, se hace un resumen de las principales tareas realizadas para preparar los corpus para el entrenamiento de los codificadores y la CDCGAN principal:

- Traducir a español los *captions* del conjunto de datos CelebA generados previamente utilizando un algoritmo propio definido en el trabajo [46] como se observa en la Figura 4.2. Posteriormente, crear un corpus nuevo con la información traducida y con la misma estructura que su versión original.

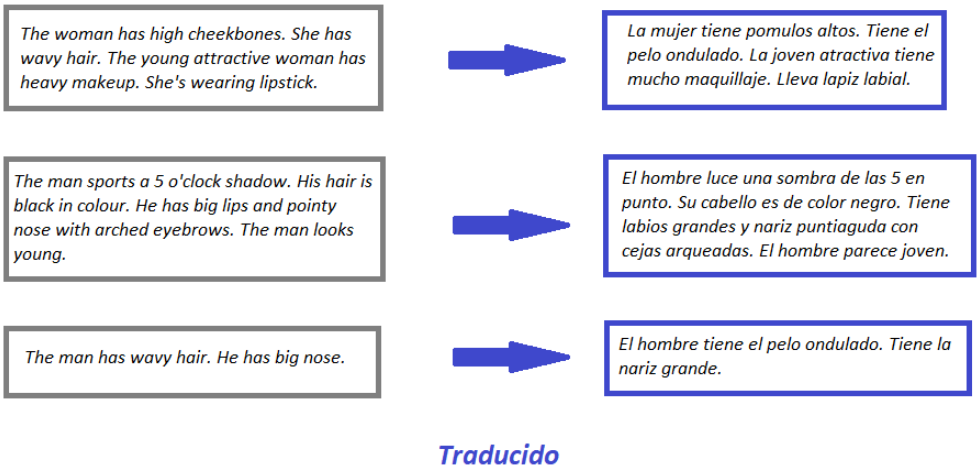


FIGURA 4.2: Traducción de los captions de CelebA. Fuente: Elaboración propia.

- Utilizar los corpus en español, inglés y la librería `Spacy` y, posteriormente, generar un corpus de entrenamiento nuevo para el algoritmo RoBERTa. La estructura del documento define en cada línea (entrada) un par de oraciones en español y su respectivo valor de similaridad entre 0 y 1 calculado por `Spacy` en base a sus respectivas versiones en inglés, tal y como se observa en la Figura 4.3.

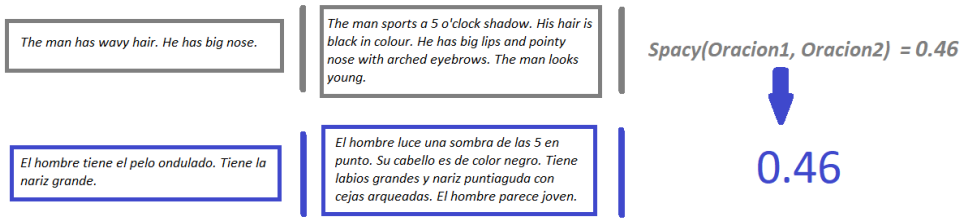


FIGURA 4.3: Cálculo de la similitud en inglés y reutilización para su par en español. Fuente: Elaboración propia.

- Generar un corpus de entrenamiento de texto puro para el algoritmo Sent2vec realizando un proceso de limpieza de información no relevante al conjunto de datos generado en el paso anterior. En concreto se juntan todas las oraciones para generar un corpus más grande, un ejemplo de ello se puede observar en la Figura 4.4.

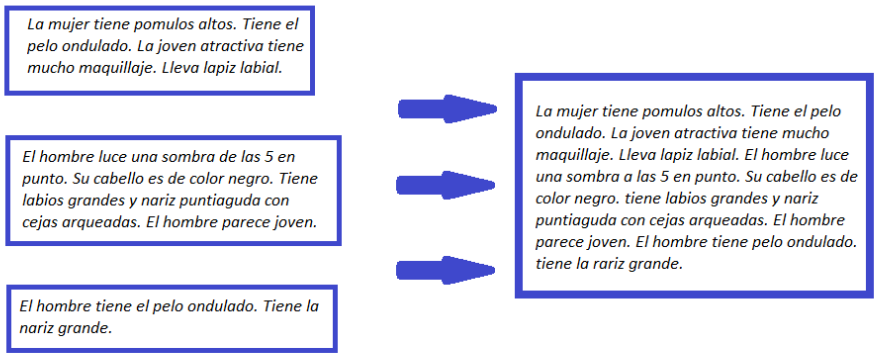


FIGURA 4.4: Formación del corpus de entrenamiento para Sent2vec. Fuente: Elaboración propia.

Finalmente, en la Figura 4.5 se puede observar el esquema general del proceso completo seguido para generar los corpus de entrenamiento para los modelos codificadores, el cual fue descrito a detalle en los puntos anteriores. El corpus de entrenamiento de la GAN se genera traduciendo las oraciones originales del idioma inglés al español. El corpus de entrenamiento de Sent2vec+CelebA se genera uniendo todas las oraciones traducidas. El corpus de RoBERTa+CelebA se crea utilizando las oraciones en inglés y español en conjunto y usando la librería `Spacy`.

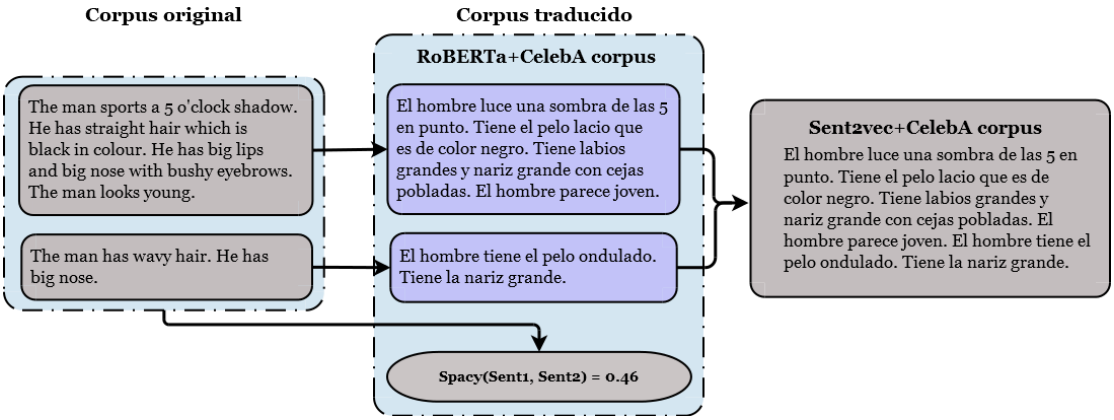


FIGURA 4.5: Esquema completo del preprocesamiento de Datos para la generación de los corpus de entrenamiento. Fuente: Elaboración propia.

4.1.2. Entrenamiento del encoder

Siguiendo la metodología de la Figura 4.1, la fase dos comprende los pasos definidos en los cuadros de color rojo. Los pasos del entrenamiento de los codificadores SBERT y Sent2vec se resumen en las siguientes tareas:

- Definir las librerías para la implementación de los codificadores. Toda la implementación se hará en Python. Para la implementación de SBERT se utilizó la librería `sentence-transformer`.
- Definir un modelo pre-entrenado el cual será la base para el entrenamiento de RoBERTa. El modelo elegido fue `RoBERTa-large-bne`.
- Implementar los *script* de entrenamiento utilizando las librerías definidas en el paso inicial.
- Realizar el entrenamiento de los codificadores evaluando valores óptimos parar sus principales variables de entrenamiento como son: tiempo de ejecución, tasa de aprendizaje, número de épocas de entrenamiento y tamaño de lote, entre otros.

4.1.3. Entrenamiento de la red GAN y aplicación de las métricas de calidad

La fase tres y final de la metodología seguida es descrita por los pasos en los cuadros de color amarillo y verde en la Figura 4.1. Finalmente se procederá a describir los pasos a seguir para el diseño, implementación y evaluación de resultados de la arquitectura generadora planteada. Estos pasos son:

- Recolectar y estudiar información referente a las diversas arquitecturas de red creadas para fines similares, las cuales serán tomadas como base para nuestra implementación.
- Diseño del las redes generadora y discriminante; así como también su conexión con el codificador.
- Implementar las arquitecturas planteadas y las métricas de calidad en Python, Se hace uso de las librerías `Keras`, `Tensorflow` y `Pytorch`. Además, se utilizaron librerías para Ciencia de Datos como `Pandas` y `Numpy`.
- Establecer las métricas de calidad adecuadas para la evaluación de las imágenes generadas. Las métricas IS, FID y LPIPS fueron descritas en el marco teórico.
- Realizar diferentes pruebas de entrenamiento y evaluación con los datos conseguidos. Aplicar las métricas de calidad objetiva sobre las imágenes generadas y realizar el análisis visual complementario para determinar los mejores resultados.

Algunos pasos definidos en la metodología serán descritos más a detalle en la sección 4.3 del presente trabajo. Adicionalmente en la Figura 4.6 se observa el esquema completo de forma sintetizada de la metodología utilizada en el presente trabajo. En ella se observan las fases de preprocesamiento, entrenamiento de los codificadores y entrenamiento de la red cDCGAN y

análisis de la calidad de las imágenes utilizando las métricas antes definidas y también mediante revisión visual.

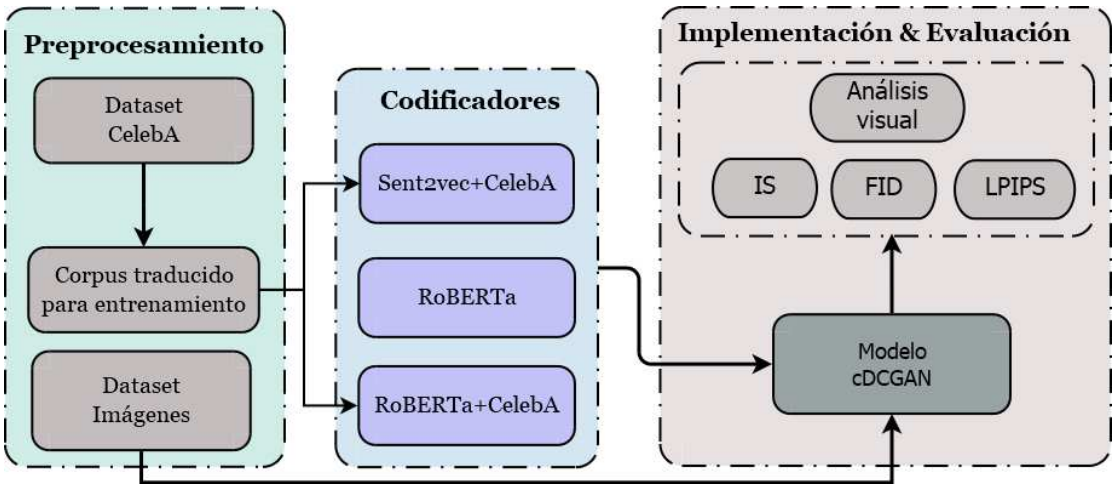


FIGURA 4.6: Esquema completo del preprocesamiento de Datos para la generación de los corpus de entrenamiento. Fuente: Elaboración propia.

4.2. Herramientas utilizadas

Para la presente tesis se utilizó un conjunto de recursos divididos en 2 grupos principales: hardware y software.

4.2.1. Hardware

Para el proceso de entrenamiento del codificador y la red CDCGAN diseñadas, así como también la evaluación de las métricas de calidad, se utilizó el servidor Pekin4 del laboratorio Redes de AltAs Prestaciones (RAAP) del Instituto de Investigación en Informática de Albacete (I^3A) de la Universidad de Castilla-La Mancha (UCLM)[1] el cual cuenta con las siguientes características técnicas:

- Memoria RAM: 220 GB.
- Capacidad de almacenamiento de disco duro: 500 GB.

[1] https://www.uclm.es

- Número de CPUs: 32 *cores*.
- Tarjeta Gráfica: NVIDIA Tesla T4 de 16GB de 2 núcleos.
- Sistema Operativo: Ubuntu Server 20.04 LTS.

4.2.2. Software

Para la implementación de los modelos y las métricas de calidad se utilizaron los siguientes recursos:

- CUDA en su versión 11.5.
- Lenguaje de programación Python en su versión 3.8.10.
- Librería `Tensorflow` en su versión 2.8.0 para GPU.
- Librería `Pytorch` en su versión 1.11.0 para GPU.
- Librería `SpaCy` en su versión 3.0.6.
- Librería `Sentence-transformer` en su versión 2.2.2.
- Librería `Sent2vec` en su versión 1.0.

4.3. Implementación del trabajo

La implementación del presente trabajo según el diagrama mostrado en la Figura 4.1 fue divido en varias tareas enfocadas en la generación de corpus para el entrenamiento de los codificadores y en el diseño de la GAN principal.

4.3.1. Conjunto de datos CelebA

CelebA es una base de datos de imágenes de celebridades a gran escala muy utilizada para trabajos de entrenamiento de GANs para generación de rostros como por ejemplo [46], [50] y [56], trabajos estudiados en el estado del arte. Las imágenes de este conjunto de datos cubren un gran conjunto de posturas, vistas

y ángulos. Además, otra característica importante de este conjunto de datos, es que se puede descargar distintas versiones de las imágenes, por ejemplo, en alta resolución, originales, cortadas y centradas y máscaras. En este trabajo se ha usado las imágenes en su versión “cortadas y centradas". Cuenta con 202,599 imágenes de caras y 10,177 identidades diferentes.

Adicionalmente, cada imagen del conjunto de datos cuenta con un conjunto de 40 atributos físicos los cuales permiten describirla. El archivo de atributos contiene una tabla de dos posibles valores, de los cuales 1 nos indica que un atributo corresponde a la imagen y -1 que no corresponde. Entre las características descriptivas principales se tiene: uso de barba, color de pelo, tipo de cejas, forma del rostro, estilo de cabello, color de ojos y atributos que realzan la apariencia.

4.3.2. Corpus de entrenamiento para la red generadora

El corpus de entrenamiento y validación para la GAN definida en el presente trabajo, se implementó traduciendo al idioma español el corpus generado en la investigación [46, 47]. En dicha investigación, para formar oraciones descriptivas a partir del archivo de atributos inicial, se crearon seis grupos de características en respuesta a seis preguntas que describen progresivamente la cara, desde el contorno hasta las características faciales que mejoran la apariencia. Así, dichos grupos se pueden observar en la Tabla 4.1.

Una vez definidos los grupos faciales, el algoritmo simple para generar descripciones faciales definido en [46, 47] genera oraciones por cada uno de los grupos. Se genera una cola en la que se van agregando cada uno de los miembros de la lista de atributos definido para un grupo en especifico. Adicionalmente, se agregan conectores o palabras intermedias y al inicio de la oración. El corpus generado esta en idioma inglés.

Para el trabajo desarrollado, se procedió a traducir al idioma español cada una las oraciones descriptivas del corpus en inglés. Para dicho proceso se siguieron los siguientes pasos:

TABLA 4.1: Preguntas y respuestas para los grupos de atributos de una imagen. Fuente: Adaptado de Text2face GAN [46, 47].

Preguntas para los grupos faciales	**Atributos faciales de la respuesta**
¿Cuál es la estructura de la cara?	Cara regordeta, papada, cara ovalada, pómulos altos.
¿Qué peinado facial luce la persona?	Sombra de las 5 en punto, perilla, bigote, patillas.
¿Qué peinado luce la persona?	Calvo, cabello lacio, cabello negro, cabello rubio, cabello castaño, cabello gris, flequillo, cabello ondulado, raya en retroceso.
¿Cuál es la descripción de los otros rasgos faciales?	Labios grandes, nariz grande, nariz puntiaguda, ojos estrechos, cejas arqueadas, cejas pobladas, boca ligeramente abierta.
¿Cuáles son los atributos que mejoran la apariencia?	Joven, atractivo, sonriente, piel pálida, mejillas sonrosadas, mucho maquillaje.
¿Cuáles son los accesorios usados por la persona?	Pendientes, sombrero, collar, corbata, lentes, lápiz labial.

- Dividir el archivo original en archivos más pequeños de 30,000 descripciones por cada uno.
- Subir cada una de los archivos al traductor Free Online Translator[2] para su traducción.
- Unión de las descripciones de los archivos un en un archivo final, una vez terminada su traducción.
- Revisión manual del archivo final para corregir errores de traducción y de estructura.

El proceso de traducción y revisión duro aproximadamente 3 días. Así, la tarea más complicada fue la división en partes más pequeñas y posterior unión de las oraciones descriptivas y, también, la corrección manual de la

[2] https://www.onlinedoctranslator.com/es/

información. El corpus final traducido contiene 192,248 oraciones descriptivas de imágenes, en tanto 10,351 imágenes no tienen descripción definida. Es por ello que durante el entrenamiento se les asignó la descripción *"Esta es una persona y nada más"* pero no se guardó la información en el archivo final del corpus.

En la Tabla 4.2 se observa una muestra de las imágenes del conjunto de datos CelebA con su respectiva descripción en idioma español. Se observa que la longitud y el nivel de detalle de las oraciones que componen la descripción de un determinado rostro, no son los mismos en todos los casos.

TABLA 4.2: Muestra de imágenes y su respectiva descripción de características traducida al español. Fuente: Elaboración propia.

Imagen	Descripción
	La mujer tiene el cabello ondulado de color castaño. Tiene labios grandes con cejas arqueadas y la boca ligeramente abierta. La joven atractiva tiene mucho maquillaje. Lleva aretes y lápiz labial.
	El hombre tiene el pelo lacio con tonos negros. El joven atractivo y sonríe.
	El hombre regordete de papada tiene pómulos altos. Su cabello es de color negro. Tiene la nariz grande, ojos estrechos con cejas pobladas y la boca ligeramente abierta. El hombre está sonriendo. Lleva corbata.

4.3.3. Corpus de entrenamiento para el Transformer

El entrenamiento del modelo base RoBERTa se realizó utilizando una red siamesa que durante el proceso se evalúa la similitud de las incrustaciones generadas por la red *transformer* utilizando la métrica de Similitud de Coseno. Por ello, cada entrada de los datos de entrenamiento consiste en un par de oraciones A y B en español y su respectiva similitud en el rango de 0 a 1.

El corpus para el entrenamiento del codificador se genera a partir del corpus en español generado en la Sección 4.3.2 y del corpus original en inglés. Para evaluar la similitud entre 2 oraciones se utilizó la librería `Spacy`. Sin embargo, esta librería solo funciona con entradas de idioma en inglés. Por lo tanto, para evaluar la similitud entre dos oraciones en español se utilizaron sus respectivos pares en inglés como se puede observar en la Figura 4.7. En este contexto, para realizar este proceso se siguieron los siguientes pasos:

- Definir la cantidad de entradas o datos de entrenamiento tendrán el nuevo corpus generado.
- Seleccionar aleatoriamente dos oraciones descriptivas del corpus en español; así como también sus respectivas oraciones del corpus en inglés.
- Calcular la similitud de oraciones utilizando la librería `Spacy` sobre las oraciones en inglés. Dicho valor se tomó como métrica de similitud de su par en español.
- Escribir en un nuevo archivo las dos entradas en español junto a su respectiva medida de similitud.
- Repetir el procedimiento hasta que se haya logrado generar la cantidad de entradas definidas al inicio.

El corpus generado cuenta con 250,000 entradas compuestas por un par de oraciones en español y su respectivo valor de similitud. Posteriormente, se dividió en 2 partes: una para entrenamiento y la otra para validación. Los pasos antes mencionados forman parte de un algoritmo propio implementado para la creación del corpus de entrenamiento del *transformer*. Dicho algoritmo se describe a detalle en el Anexo A.1.

4.3.4. Entrenamiento del encoder SBERT

Con el objetivo de mejorar el rendimiento del codificador SBERT (Modelo base RoBERTa), se entrenó el modelo utilizando el corpus generado en la

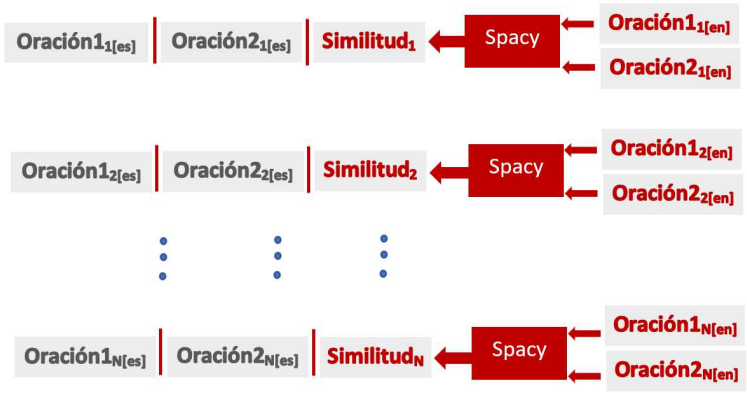


FIGURA 4.7: Estructura del corpus de entrenamiento de SBERT.
Fuente: Elaboración propia

Sección 4.3.3 y siguiendo la estrategia de uso de una red Siamesa junto a la función de pérdida de Similitud de Coseno definidos en [5]. Para ello, se acondicionó el código guía base mostrado en la sección de entrenamiento de la página oficial de la librería SBERT[3] siguiendo los siguientes pasos:

- Definir las librerías `sentence_transformer` y `torch` para el entrenamiento del codificador.
- Dividir el corpus de entrenamiento en dos partes. Para el entrenamiento se utilizan 249,999 oraciones y para la validación se emplean 10,000.
- Cargar los datos de entrenamiento y validación para el modelo. Se generan dos listas para el almacenamiento de la información y en cada una de ellas, las entradas están compuestas por un par de oraciones descriptivas y su valor de similitud. Se utilizan los paquetes `InputExample` y `DataLoader` de las librerías `sentence_transformer` y `torch`, respectivamente.
- Para no iniciar el entrenamiento desde cero, se utiliza un modelo SBERT preentrenado, el cual cuenta con una capa BERT y una capa de agrupación (*pooling*). Este modelo nos da un vector de salida fijo de 768 dimensiones independiente de la longitud del texto de entrada. El modelo elegido fue

[3] `https://www.sbert.net/docs/training/overview.html`

`RoBERTa-large-bne`. Se utiliza las funciones *Transformer* y *Pooling* del paquete `models` de la librería `Sentence_Transformer`.

- El entrenamiento se realiza utilizando una red Siamesa en la cual, para un par de oraciones A y B del corpus de entrenamiento, se evalúa las similitudes de sus vectores de incrustación u y v generados. Para este fin, se implementa la métrica Similitud de Coseno. Para su implementación se utiliza la función `CosineSimilarityLoss` del paquete `losses` de la librería `Sentence_Transformer`.

- Durante el proceso de entrenamiento se definen evaluadores que utilizan el corpus de validación para evaluar la eficacia del modelo entrenado. Estos evaluadores se ejecutan periódicamente. Para su implementación se empleó la función `EmbeddingSimilarityEvaluator` del paquete `evaluations` perteneciente a la librería `Sentence_Transformer`.

- Se crea el modelo SBERT final tomando como argumentos al modelo preentrenado y la función de agrupamiento. Para su implementación se utiliza el paquete `SentenceTransformer` de la librería `Sentence_Transformer`.

- Finalmente, se procede a entrenar el modelo junto con los argumentos previamente definidos, para ello se emplea la función `fit` y se definen valores para sus principales parámetros de entrenamiento. Los parámetros más importantes son: Objetivos de entrenamiento (`train_objetives`), número de épocas (`epochs`), evaluadores (`evaluator`), nombre del archivo del nuevo modelo (`output_path`) y la habilitación para guardar el mejor valor (`save_best_model`).

Como resultado del entrenamiento del modelo base RoBERTa utilizando el corpus en español generado en la sección 4.3.3 se obtuvo el modelo propio RoBERTa+CelebA el cual es evaluado en el presente trabajo.

4.3.5. Entrenamiento del encoder Sent2vec

Al igual que otros algoritmos de incrustación de oraciones, Sent2vec puede ser utilizado directamente para textos en idioma inglés. Para este fin, únicamente hay que descargar la librería e ingresar la información, dado que la gran mayoría de estos algoritmos fueron entrenados usando el inglés como idioma original. Sin embargo, dado que este trabajo se utiliza con texto en español, ha sido necesario entrenarlo previamente utilizando el corpus generado y descrito en la Sección 4.1.1.

Tomando como guía el manual de entrenamiento descrito por Muhammad Farhat [60], se describe el siguiente proceso de entrenamiento:

- Realizar un preprocesamiento inicial al corpus en español. Para ello, se ha desarrollado un nuevo archivo en donde se guardan cada una de las entradas del corpus original y elimina los demás componentes, como los nombres de imagen al cual describe y símbolos. Se tiene un total de 192,209 oraciones para el entrenamiento.

- Aplicar un segundo preprocesamiento consistente en eliminar los acentos. Por otro lado, los `stopwords` y conectores se conservaron, dado que para este caso son importantes como parte de la estructura de las oraciones durante el entrenamiento. La información procesada es guardada en un archivo de texto.

- Instalar y configurar las librerías `Sent2vec`, `FastText`, `Numpy` y `Cython`; las cuales en conjunto permiten ejecutar el comando de entrenamiento.

Como resultado del entrenamiento de Sent2vec utilizando el corpus en español descrito en la sección 4.1.1, se obtuvo el modelo Sent2vec+CelebA el cual permite codificar oraciones en español. La ejecución del comando de entrenamiento y sus principales argumentos se describen detalladamente en el Anexo A.2.

4.3.6. Arquitectura de la red GAN implementada

La arquitectura usada en el presente trabajo tomó como base el modelo Text2faceGAN implementado en [46, 47]. Posteriormente, esta arquitectura ha sido modificada en base a pruebas de entrenamiento realizadas. La elección de dicho modelo base es debido a que implementa todas las recomendaciones para mejorar modelos GAN mencionadas en el capítulo 5 del libro de Brownlee [33],las cuales son:

- Implementar el submuestreo del discriminante usando capas convolucionales 2D, `conv2d`, en vez de capas `maxpooling` como en redes neuronales profundas comunes.
- Implementar el muestreo incremental en el generador usando capas convolucionales transpuestas, `Conv2DTranspose`.
- Usar capas de activación lineal rectificada con fugas, LeakyReLU, en el generador. De esta manera, la finalidad de que valores de entrada negativos o cero tomen el valor de cero y valores positivos no se alteren.
- Utilizar una función de inicialización de pesos gaussiana y la función de optimización Adam.

Arquitectura del generador

Como se puede observar en la Figura 4.8, la red generadora está compuesta por una serie de capas de convolución transpuesta junto con las capas de activación LeakyRelu y la función de optimización Adam. A continuación se describen a detalle los componentes de la arquitectura de red:

- La red tiene 2 entradas, la primera para el vector de características generadas por el codificador y la segunda para el vector de ruido. Si se trabaja con SBERT la dimensión de la primera entrada será 1,024 y si es con Sent2vec tendrá un valor de 4,800. Para ambos casos, a través de una capa densa se baja su dimensión hasta 256. Las segunda entrada tiene una dimensión de fija de 100.

- Las dos capas definidas en el punto anterior se concatenan y pasan por una capa densa de 512 neuronas antes de ingresar a las capas intermedias.
- Tiene 4 capas de convolución transpuesta, `Conv2DTranspose`, para implementar el *upsampling* de las imágenes. Las cuales tienen 256, 128, 64 y 3 filtros.
- Entre las capas densas y de convolución transpuesta se establecen capas de activación ReLU con sus parámetros por defecto. Este procedimiento permite optimizar los valores de ingreso en cada entrada.
- Al igual que las capas ReLU, también se aplica `BatchNormalization` con `momentum = 0.2`.
- Siguiendo las recomendaciones de mejora, todas las capas `Conv2DTranspose` tiene como inicialización de pesos, `kernel_initializer` a una distribución normal.
- Después de la capa inicial de concatenación, `merge` y de la última capa `Conv2dTranspose` se tiene una capa de activación TanH.
- Se tienen dos capas `lambda`: (i) La primera realiza una operación de división entre 2 del tensor de salida; y (ii) La segunda realiza una adición de 0.5 al tensor.
- La arquitectura utiliza una función de optimización Adam con `learning_rate = 0.0002` y `beta1 = 0.5`.
- Se define como función de perdida *Binary Crossentropy* para la red a la hora de realizar la compilación.
- Al final, la salida genera imágenes de 3 canales y dimensión 64×64, las imágenes generadas cumplen con la descripción textual ingresada a la red. Durante el entrenamiento las imágenes generadas son pasadas a la red discriminante.

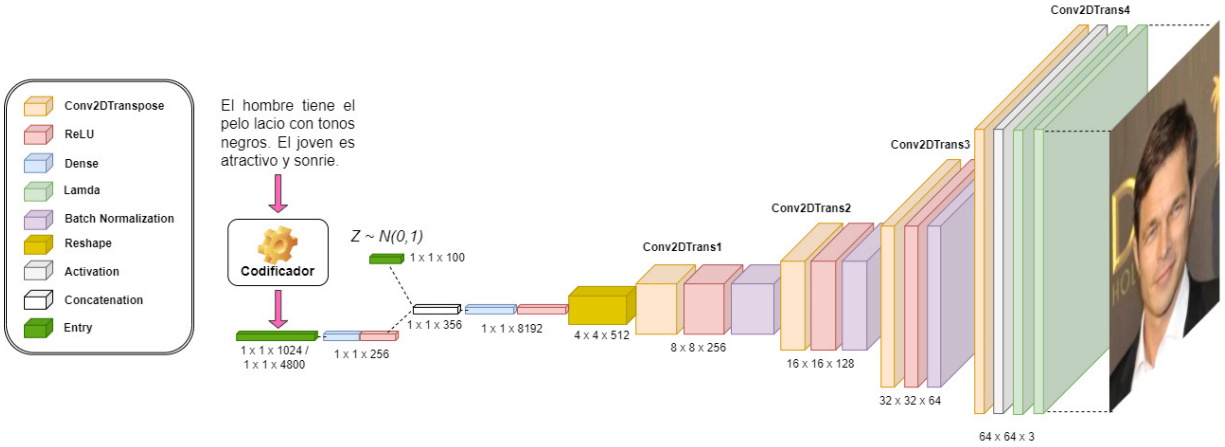


FIGURA 4.8: Arquitectura del Generador. Fuente: Elaboración propia.

Arquitectura del discriminante

El segundo componente de nuestra GAN es la red discriminante. Ese componente interactúa con la red generadora dado que será la encargada de evaluar la calidad de las imágenes generadas. Como se observa en la Figura 4.9, la red está definida con los siguientes componentes:

- Al igual que el modelo generador, este modelo tiene 2 entradas. La primera recibe la imagen creada por el generador, la cual tiene dimensión $64 \times 64 \times 3$. La segunda entrada es el vector de características del texto cuya dimensión será 1,024 si se trabaja con SBERT (modelos RoBERTa y RoBERTa+CelebA) y tendrá un valor de 4,800 si el codificador es Sent2vec (modelo Sent2vec+CelebA). Antes de concatenarse, ambas entradas pasa por un conjunto de capas intermedias.

- En la primera entrada, ingresa la imagen generada por el generador y seguidamente pasa por 3 capas convolucionales normales, `conv2D`, con filtros de 64, 128 y 256 en ese orden para implementar el *downsampling* de la imagen de entrada.

- Entre cada una las capas `Conv2D` se agregan capas de activación LeakyReLU con los valores por defecto de sus parámetros para mejorar las entradas entre las capas.

- Por la otra entrada, el vector de características pasa por 1 capa densa y 3 capas `lambda`. La finalidad de la primera capa fue disminuir su longitud y las 3 siguientes para cambiar su dimensión.

- Finalmente, las entradas se concatenan y antes de salir de la red, pasan por una capa convolucional de 512 filtros y capas LeakyReLU adicionales.

- La información pasa por una capa `Flatten` la cual transforma todo a un solo tensor cuya longitud es la multiplicación de las dimensiones de la entrada. Se adicionan una capa para prevenir el sobreajuste al momento del entrenamiento y una capa densa la cual ofrece como salida una etiqueta simple la cual dice si la imagen es falsa o verdadera.

- De forma similar al generador, esta red utiliza la función de activación `sigmoid` y la función de optimización Adam con `learning_rate = 0.0002` y `beta1 = 0.5`. Adicionalmente, al momento de compilar se utiliza la función de perdida *binary crossentropy*.

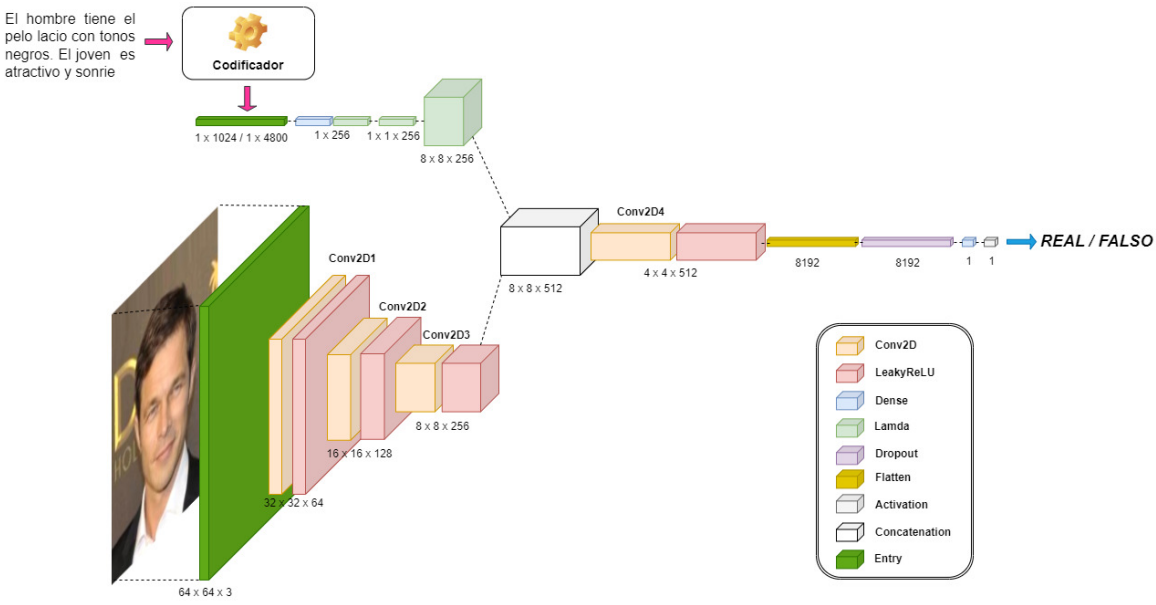


FIGURA 4.9: Arquitectura del Discriminante. Fuente: Elaboración propia.

4.3.7. Implementación de la métrica Inception Score

La implementación de la métrica IS se realizó utilizando la librería `keras.applications.inception_v3`[4], la cual nos brinda los paquetes para procesar las imágenes y definir el modelo de red neuronal. Adicionalmente, se hizo uso de las librerías `numpy`, `PIL`, `tqmd` y `matplotlib` para manejo de imágenes. Se siguieron los siguientes pasos:

- Se implementó un programa en Python y se hizo uso de los paquetes `InceptionV3` y `preprocess_input` de `keras.applications.inception_v3` para definir el modelo de red neuronal a usar y aplicar preprocesamiento a las imágenes, respectivamente.
- Se redimensionaron las imágenes de entrada a un tamaño de $299 \times 299 \times 3$, dicha dimensión es requisito para hacer uso del modelo utilizado por IS.
- Para el cálculo de la métrica se utilizaron dos conjuntos de 1000 imágenes; el primero son las imágenes originales y el segundo son las imágenes generadas por nuestra red. Se preprocesan las imágenes y se calcula su etiqueta utilizando el modelo InceptionV3.
- La función de cálculo toma una matriz de imágenes con el tamaño esperado y los valores de píxeles en el rango de 0 a 255. Calcula los puntajes iniciales de desviación promedio y estándar utilizando el modelo InceptionV3 implementado en Keras.
- Nuestro programa calcula valores de IS para cinco grupos de imágenes diferentes y obtiene el resultado final calculando el promedio de todos.

[4] `https://keras.io/api/applications/inceptionv3/`

4.3.8. Implementación de la métrica Learned Perceptual Image Patch Similarity

Para la implementación de LPIPS se hizo uso de la librería `lpips` [5] de Python especializada en el cálculo de dicha métrica. La implementación se realizó de la siguiente forma:

- LPIPS calcula la similitud entre las activaciones de dos imágenes para una red predefinida. Para esta implementación se uso la red predefinida Alex.
- Cada una de las imágenes utilizadas son transformadas en tensores utilizando el módulo `im2tensor` antes de ingresar a la red Alex para el cálculo. El resultado final se obtiene aplicando la media y la desviación estándar al conjunto de resultados obtenidos de cada uno de los pares de imágenes.
- Al igual que en la métrica anterior, se utilizó dos conjuntos de imágenes para su ejecución: (i) El primero son las imágenes originales; y (i) El segundo con las imágenes generadas por nuestra red.
- El programa hace un total de cinco cálculos para cinco grupos de imágenes diferentes y obtiene el resulta final por medio del promedio de todos los valores calculados.

4.3.9. Implementación de la métrica Frechet Inception Distance

Para la implementación de FID, se utilizó la librería oficial implementada utilizando Pytorch denominada `pytorch-fid` [6]. Entre sus funcionalidades destacan la opción de uso de GPU y determinar la dimensión del vector de características empleado para el cálculo. Se tuvieron en cuenta las siguientes consideraciones:

- Dado que el método de ejecución de la librería es vía terminal de comandos, se implementó un programa en Python el cual utilizando

[5] `https://pypi.org/project/lpips/`

[6] `https://github.com/mseitzer/pytorch-fid`

el comando `subprocess` el cual, llama a `pytorch-fid` junto a los argumentos deseados.

- Al igual que en las métricas anteriores, se utilizó dos conjuntos de imágenes para su ejecución: (i) El primero son las imágenes originales; y (ii) El segundo con las imágenes generadas por nuestra red.
- Los argumentos pasados a la librería son los directorios de imágenes originales y generadas, así como también, la opción de uso de CPU o GPU.
- El programa calcula FID para cinco grupos de imágenes y al final, el resultado de la métrica se define como el promedio de los cinco resultados.

Todos los corpus de entrenamiento, modelos, códigos de entrenamiento y demás recursos descritos en este capitulo y creados para nuestra investigación, fueron publicados en los repositorios HugginFace y Github para que puedan ser descargados y usados por los lectores. Los detalles y enlaces se describen en el Anexo A.3.

Capítulo 5

Experimentación y resultados

En el siguiente capítulo se mostrarán los resultados obtenidos luego del entrenamiento y testeo del modelo generador de imágenes planteado. Además, se mostrará también los resultados de los codificadores utilizados. Finalmente, se realiza una discusión de los resultados obtenidos en contraste con los ya obtenidos en la literatura.

5.1. Fase de entrenamiento del encoder SBERT

Siguiendo los pasos descritos en la Sección 4.3.3 se generó un archivo de 250,000 entradas denominado `caps_bert.txt` con el *corpus* de entrenamiento para el *transformer*. El conjunto de datos del archivo generado se dividió en 2 partes y se guardó en los nuevos archivos `caps_bert_train.txt` de 249,000 entradas y `caps_bert_test.txt` de 1,000 entradas para las tareas de entrenamiento y validación respectivamente.

Por otro lado, se implementó un script Python denominado `SBERT_training.ipynb` siguiendo los pasos establecidos en la Sección 4.3.4. Empleando los *corpus* de entrenamiento y validación, se procedió a entrenar el *transformer* utilizando el hardware descrito en la Sección 4.2.1. Se definió un conjunto de parámetros de entrenamiento los cuales se muestran en Tabla 5.1. Como se puede observar, se establecieron 145 épocas de entrenamiento debido a que en pruebas previas se observó que el tiempo de entrenamiento es

bastante grande para modelos de este tipo con la infraestructura hardware actual. También se activó la opción de guardado del mejor valor obtenido para el indicador de pérdida definido.

TABLA 5.1: Parámetros de entrenamiento del transformer. Fuente: Elaboración propia.

Parámetros de entrenamiento	**Valor**
Número de épocas	145
Archivo de salida	`RoBERTa-large-bne-celebAEs-UNI`
Guardar el mejor modelo	`True`
Mostrar barra de progreso	`True`
Pasos antes de cada guardado	300

El tiempo total de entrenamiento de modelo RoBERTa+CelebA fue de 42 días utilizando todas las GPUs disponibles del servidor, es decir, dejando exclusivamente el servidor para este entrenamiento. Como resultado, se genera un directorio con el nombre definido en los parámetros de entrenamiento `RoBERTa-large-bne-celebAEs-UNI` el cual, contiene información importante del entrenamiento como el archivo CSV de la evaluación de resultados y el archivo binario con los nuevos pesos del modelo entrenado.

La medición de rendimiento del modelo durante el entrenamiento se calculó mediante el coeficiente de correlación de Spearman entre el vector de similitudes reales y el vector de similitudes calculado. Este último se obtuvo aplicando la similitud de coseno de los vectores codificados generados por RoBERTa para cada uno de los pares de oraciones del *corpus* de evaluación `caps_bert_test.txt`.

Como se puede observar en la Figura 5.1, el mayor aumento del coeficiente de correlación se observó durante el inicio del entrenamiento. Además, la oscilación de valores más notable de correlación fue entre 0.998 y 1, por ello se puede decir que rápidamente alcanzo los mejores resultados para el modelo y se mantuvo en esa línea durante todo el entrenamiento.

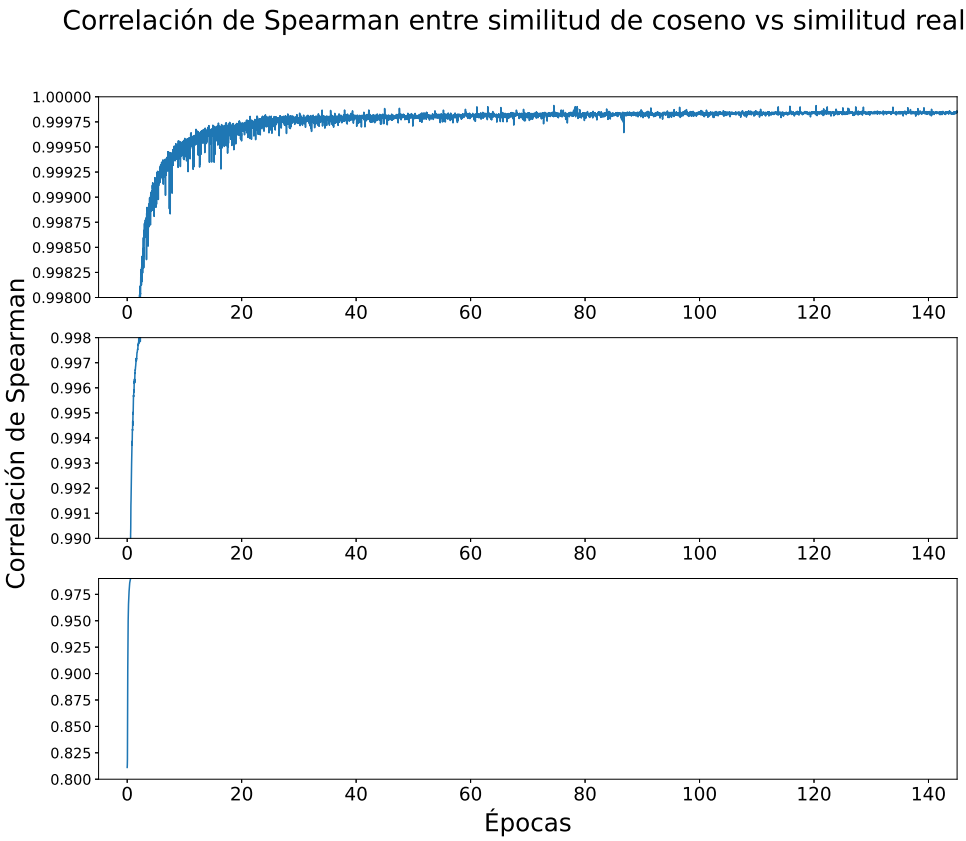


FIGURA 5.1: Variación del coeficiente de correlación de Spearman durante el. entrenamiento de RoBERTa+CelebA. Fuente: Elaboración propia.

Adicionalmente, se ha calculado la correlación de Spearman utilizando el *corpus* de evaluación y el modelo base RoBERTa para generar sus vectores de codificador y se compararon con los resultados obtenidos al evaluar nuestro modelo entrenado. En la Tabla 5.2 se observa que RoBERTa+CelebA obtiene una correlación menor que el RoBERTa al inicio del entrenamiento. Sin embargo, conforme se avanza el entrenamiento se observa que se logra obtener un mejor valor final, por ende, se logra mejorar el rendimiento del codificador.

5.2. Entrenamiento del codificador Sent2vec

Siguiendo los pasos definidos en la Sección 4.3.5 se generó el *corpus* de entrenamiento `s2v_train.txt` y se instalaron los paquetes para

TABLA 5.2: Comparación de resultados del modelo entrenado vs original. Fuente: Elaboración propia.

Modelo	Épocas de entrenamiento	Correlación Spearman
`RoBERTa`	modelo base	0.827176427
`RoBERTa+CelebA`	–	0.811153072
`RoBERTa+CelebA`	74	**0.999913276**

el entrenamiento de Sent2vec. Se definió un conjunto argumentos de entrenamiento de la librería `FastText` que guiaron el entrenamiento del codificador. Los principales argumentos y sus respectivos valores son mostrados en la Tabla 5.3. En ella, resalta la definición de valores altos para el número de épocas de ejecución y la cantidad de hilos usados. Este procedimiento se debe a que en [6] menciona que el algoritmo Sent2vec es mucho más rápido que la mayoría codificadores incluyendo SBERT por lo que se aprovechara dicha característica y las capacidades del hardware disponible.

TABLA 5.3: Parámetros de entrenamiento de Sent2vec+CelebA. Fuente: Elaboración propia.

Argumento de entrenamiento	Valor
Corpus de entrenamiento	`s2v_train.txt`
Nombre del directorio de salida	`sent2vec-celebAEs-UNI`
dimensión del vector de codificador	$4,800$
Número de épocas de entrenamiento	$5,000$
Ratio de aprendizaje	0.05
Función de pérdida	*Skipgram negative sampling*
Número de hilos de ejecución	200

En este contexto, el tiempo de entrenamiento total duró 7 horas ejecutado en el hardware descrito en la Sección 4.2.1 y trabajando con todos los CPUs a máximo rendimiento. Como resultado nos genera un archivo de extensión `bin` el cual es cargado mediante la librería `Sent2vec` desde Python.

Como se puede observar en el siguiente código Python, al cargar el binario del modelo `sent2vec-celebAEs-UNI.bin` entrenado, ya es posible generar el vector de características de una oración en español. La dimensión del vector obtenido es 4,800.

```
import sent2vec
Model_path="sent2vec_celebAEs-UNI.bin"
s2vmodel = sent2vec.Sent2vecModel()
s2vmodel.load_model(Model_path)
caption = """El hombre luce una sombra a las 5 en punto.
Su cabello es de color negro. Tiene una nariz grande
con cejas tupidas. El hombre se ve atractivo"""
vector =  s2vmodel.embed_sentence(caption)
print(vector)
```

5.3. Entrenamiento y evaluación del modelo generador

Cumpliendo con los pasos definidos en el Capítulo 4, se crearon el *corpus* de entrenamiento para la GAN (`caps_es_proc.txt`) con 192,248 entradas de texto, se implementó en Python la red generadora (`T2F_DCGAN-sbert.py`) y las métricas de evaluación IS (`Inception Score.ipynb`), FID (`FID-GAN.ipynb`) y LPIPS (`LPIPS.ipynb`).

Así mismo, como parte del diseño final de la arquitectura cDCGAN, se hicieron múltiples entrenamientos de nuestra arquitectura GAN con distintos valores de variables de entorno. Se pudo observar que el proceso de entrenamiento en general es largo dependiendo de la cantidad de imágenes, el tamaño de lote definido, número de épocas entrenadas y el codificador utilizado.

Para la evaluación final se entrenó modelo utilizado los codificadores Sent2vec+CelebA, RoBERTa y RoBERTa+CelebA. Todas las iteraciones del entrenamiento y evaluación se realizaron en el servidor Pekin4 descrito en la Sección 4.2.1. Se procedió a realizar el entrenamiento definiendo los valores de

las variables antes mencionadas:

- **Número de épocas**: Dada la importancia de esta variable en los resultados finales, se procedió a definir valores entre 50 y 600 con intervalos de 50.
- **Número de imágenes de entrenamiento**: El total de imágenes es de 202,599, una cantidad bastante grande la cual, de ser utilizada en su totalidad, dificultaría el entrenamiento en tiempo y recursos. El valor establecido fue 50,000, una cantidad intermedia de información con posibilidades de ser procesada en el hardware actual.
- **Tamaño de lote**: Esta variable es significativa, dado que dependiendo de su valor condiciona si el entrenamiento será exitoso o no. Además el rendimiento del aprendizaje y el tiempo de ejecución son afectados. Su valor fue establecido en 512.

En este contexto, la Tabla 5.4 muestra los tiempos de entrenamiento da la GAN utilizando los codificadores a evaluar. Se pueden mencionar las siguientes observaciones:

- En general el tiempo de entrenamiento de la red cDCGAN es bastante grande, tiempo común en este tipo de redes.
- El tiempo de entrenamiento aumenta en forma directamente proporcional al número de épocas y cantidad de imágenes de entrenamiento.
- La diferencia entre los tiempos de entrenamiento y la proporción de aumento es más notoria para valores de épocas grandes.
- El tiempo de entrenamiento con Sent2vec es más corto en comparación a los *transformers* y, a su vez, no se ve una diferencia considerable de tiempo entre el modelo RoBERTa base y el entrenado por nosotros.

TABLA 5.4: Tiempos de entrenamiento del modelo generador usando los codificadores evaluados. Se muestra el número de épocas (*Épochs*) y los tiempos de entrenamiento de los modelos RoBERTa+CelebA, RoBERTa y Sent2Vec+CelebA. Fuente: Elaboración propia.

Épocas	**RoBERTa+CelebA**	**RoBERTa**	**Sent2vec+CelebA**
50	2h 15min	2h 16 min	2h 11 min
100	4h 42min	4h 40 min	4h 31 min
150	7h 30min	7h 25 min	7h 12 min
200	11h 01min	10h 48min	10h 34min
250	15h 40min	15h 18min	15h 23min
300	21h 44min	21h 13min	20h 59min
350	29h 24min	28h 43min	28h 35min
400	38h 55min	38h 01min	37h 57min
450	50h 29min	49h 21min	49h 18min
500	63h 18min	63h 05min	63h 09min
550	80h 44min	79h 24min	79h 26min
600	99h 31min	98h 02min	98h 11min

Por otro lado, la Figura 5.2 muestra los valores de la función de perdida *Binary Crossentropy* tanto para el generador como para el discriminante entrenados con cada uno de los codificadores utilizados en el presente trabajo. A partir de ello se puede mencionar lo siguiente:

- Se observa que para los 3 casos el generador comienza con un valor de perdida más bajo que el discriminante, sin embargo, con el pasar del tiempo va empeorando superando a los valores del discriminante. Al final del entrenamiento en los 3 casos la pérdida del generador es mayor al discriminante. Este comportamiento oscilante se debe su objetivo de tratar de obtener el equilibrio de Nash durante el entrenamiento.

- Comparando a los resultados obtenidos en determinadas épocas se observa que cuando hay un cambio de tendencia en las perdidas la

calidad de las imágenes comienza a decaer, para los entrenamientos con los codificadores RoBERTa y RoBERTa+CelebA la calidad mejora hasta el final. Sin embargo, para Sent2vec a partir de la época 400 hay un cambio de tendencia de pérdidas y por ende disminución de calidad.

- Tal y como se menciona en las referencias estudiadas como [56], las tendencias de valores de pérdidas tiene muy poca relación con la calidad de las imágenes generadas, es por ello que es mejor hacer uso de lás métricas de calidad como IS, FID y LPIPS.

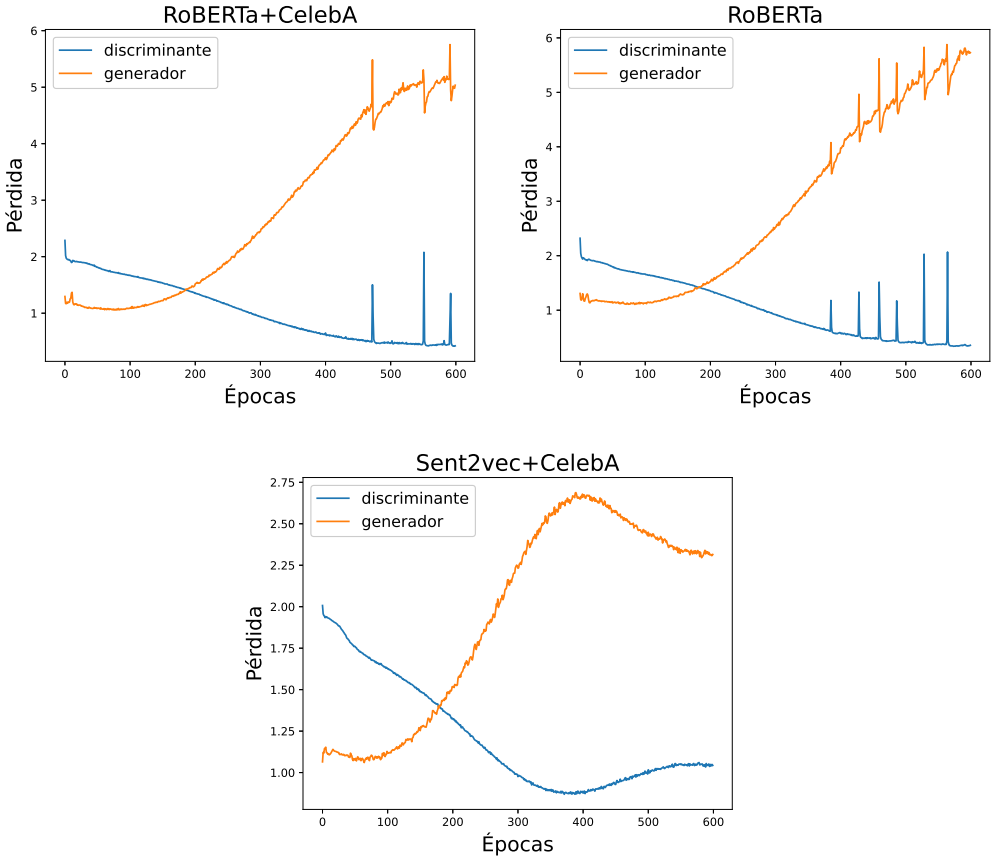


FIGURA 5.2: Gráfica de funciones de pérdida durante el entrenamiento para el modelo generador con cada uno de los encoders. Fuente: Elaboración propia.

La medición de la eficiencia y eficacia del modelo entrenado se realiza mediante las métricas de calidad previamente definidas, así como también, un análisis visual de las imágenes generadas. En los siguientes apartados se muestran los resultados obtenidos para esos dos tipos de evaluación.

5.3.1. Evaluación de las imágenes generadas con las métricas FID, IS y LPIPS

Para implementar dichas evaluaciones, se tomaron 1000 imágenes de manera aleatoria junto con sus respectivas oraciones descriptivas. Utilizando el modelo entrenado y las oraciones descriptivas se generaron 1000 imágenes artificiales. Así, este último paso se realizó 5 veces con la finalidad de obtener 5 grupos de imágenes para las pruebas.

En la Tabla 5.5 se observan los resultados de la métrica FID sobre el conjunto de imágenes de validación. Se puede observar lo siguiente:

- El modelo RoBERTA+CelebA entrenado en el presente trabajo, obtiene el mejor valor con respecto al modelo base de RoBERTa y Sent2vec+CelebA. Especificamente, el mejor valor se logra en la época 550.

- Al inicio los modelos de RoBERTa y RoBERTa+CelebA brindan un mejor valor que Sent2vec+CelebA, sin embargo desde la época 100 hasta la 150, este último muestra un valor mejor que RoBERTA.

- Comparando RoBERTA y nuestro modelo entrenado con CelebA, se puede observar que el primero obtiene mejores resultados hasta en 5 puntos de controla con respecto al número de épocas, un poco menos de la mitad de mediciones.

- Para el caso de Sent2vec, se observa que a partir de la época 150 en adelante, la calidad de las imágenes decae lo cual se ve reflejado en los valores de FID para esos casos.

TABLA 5.5: Valores de FID sobre el conjunto de imágenes de prueba. Se muestra el número de épocas, y los codificadores utilizados: BoBERTa+CelebA, RoBERTa y Sent2Vec+CelebA. En negrita se muestra el mejor resultado. Fuente: Elaboración propia.

Épocas	**Codificadores**		
	RoBERTa+CelebA	**RoBERTa**	**Sent2vec+CelebA**
50	158.4091	151.5272	186.7900
100	102.9159	152.6239	132.7254
150	101.0391	109.2439	105.5604
200	92.9750	104.3698	105.9005
250	97.7691	97.3079	109.5046
300	92.8755	89.4257	114.8679
350	92.8265	89.3566	128.2412
400	85.5626	90.7858	132.0908
450	93.4870	89.5354	125.4737
500	86.6633	89.4690	146.8322
550	**84.2212**	93.5891	151.2761
600	84.4780	89.5146	154.4416

En la Figura 5.3 se puede observar la variación de la métrica FID con el incremento de épocas de entrenamiento, calculada para el conjunto de imágenes de prueba, información mostrada a detalle en la tabla 5.5 y comentada en el párrafo anterior. De manera complementaria se observa lo siguiente:

- Ninguno de los codificadores genera que la métrica tenga un decrecimiento constante durante todo el entrenamiento. Se notan crecimiento durante determinadas épocas de entrenamiento.
- De acuerdo a los valores de la métrica, el de entrenamiento influye en la calidad de imágenes generadas, siendo el caso más notorio en Sent2vec+CelebA cuyos valores de FID se incrementan a partir de la época 200 en adelante produciendo imágenes de peor calidad,

- Se produce una fuerte variación (disminución de valores) de la métrica durante las primeras épocas de entrenamiento. En el caso de RoBERTa+CelebA entre 0 y 100, para RoBERTa entre 0 y 150 y finalmente, para Sent2vec+CelebA entre 100 y 150.

- A partir de la época 250 hasta el final del entrenamiento el decrecimiento y crecimiento de la métrica usando los modelos RoBERTA y RoBERTa+CelebA es pequeño. Además, sus valores toman un comportamiento sinusoidal donde uno es superado por el otro y viceversa en cada época.

- El mejor valor de Sent2vec+CelebA es 105.5046 en la época 200. En el caso de RoBERTa es 89.3566 en la época 350, Finalmente, para nuestro modelo entrenado RoBERTa+CelebA su mejor valor es 84.2212 en la época 550.

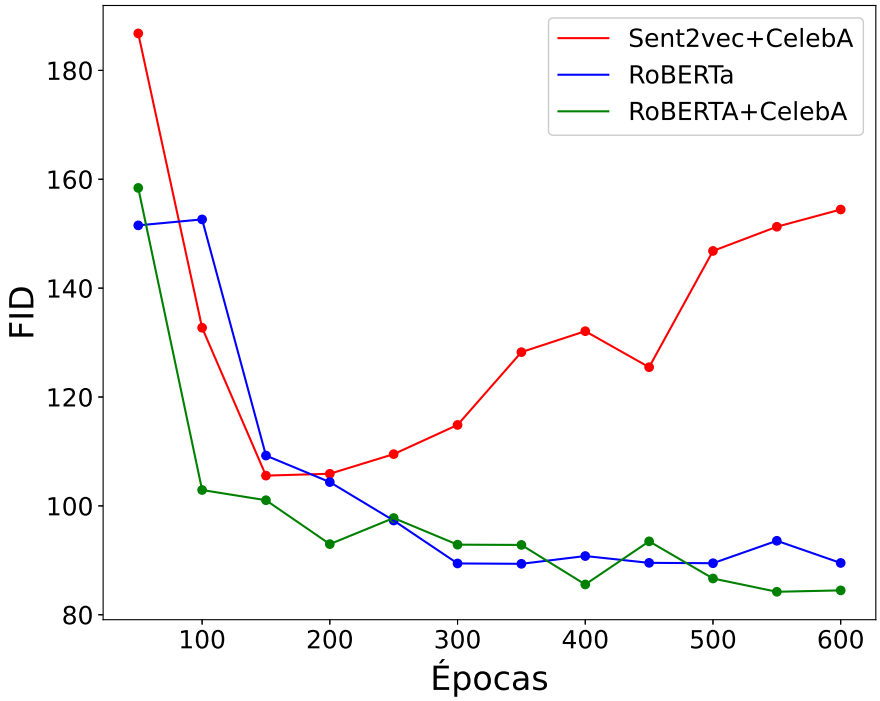


FIGURA 5.3: Variación de la métrica FID con respecto a las épocas de entrenamiento para el modelo usando cada uno de los codificadores estudiados. Fuente: Elaboración propia.

Para la métrica IS, en la Tabla 5.6 se observan los resultados de la métrica sobre el conjunto de imágenes de validación. Se puede observar lo siguiente:

- A partir de la época 250, el modelo Sent2Vec+CelebA obtiene mejores resultados que los otros dos modelos, pero con una diferencia de valores bastante pequeña.
- Comparando los dos modelos RoBERTa y RoBERT+CelebA, este último brinda mejores resultados en 7 de las 12 épocas de entrenamiento.
- El mejor valor de la métrica lo obtiene nuestro modelo RoBERTa+CelebA durante los primeros tiempos de entrenamiento, más específicamente en la época 50. Comparando este resultado y otros en diferentes épocas con sus respectivas evaluaciones visuales de la calidad de imagen se observa que dicha métrica brinda resultados contrarios. A menor calidad de visual de la imagen mayor valor de IS.
- El mejor valor de IS para nuestro modelo entrenado RoBERTa+CelebA es 2.7891 ± 0.1659, para el caso de RoBERTa es 2.71 ± 0.1769 y finalmente, para Sent2vec+CelebA su mejor obtenido es 2.7380 ± 0.1979, siendo los 3 en la época 50.
- La comparación de estos resultados numéricos con el análisis visual, confirma que IS no es un métrica precisa, como se discutió en las secciones anteriores, ya que no existe una relación directa entre una puntuación alta de IS con la calidad de las imágenes. Se tiene en cuenta que una puntuación IS más alta indica una mejor calidad de las imágenes generadas.

TABLA 5.6: Valores de IS sobre el conjunto de imágenes de prueba. Se muestra el número de épocas, y los codificadores utilizados: BoBERTa+CelebA, RoBERTa y Sent2Vec+CelebA. En negrita se muestra el mejor resultado. Fuente: Elaboración propia.

Épocas	**Codificadores**		
	RoBERTa+CelebA	**RoBERTa**	**Sent2vec+CelebA**
50	**2.7891 ± 0.1659**	2.7143 ± 0.1769	2.7380 ± 0.1979
100	2.4372 ± 0.1608	2.6485 ± 0.1837	2.5080 ± 0.1540
150	2.4252 ± 0.1396	2.5958 ± 0.1762	2.3820 ± 0.1525
200	2.4116 ± 0.1399	2.5253 ± 0.1737	2.4915 ± 0.1708
250	2.3389 ± 0.1398	2.4382 ± 0.1448	2.6077 ± 0.1311
300	2.3522 ± 0.1371	2.3418 ± 0.1185	2.3709 ± 0.1472
350	2.4065 ± 0.1495	2.4263 ± 0.1342	2.5316 ± 0.1629
400	2.4458 ± 0.1517	2.4022 ± 0.1635	2.5310 ± 0.1790
450	2.4426 ± 0.0774	2.2848 ± 0.1186	2.5350 ± 0.0949
500	2.4231 ± 0.1236	2.3820 ± 0.1388	2.6081 ± 0.1448
550	2.4057 ± 0.1401	2.2523 ± 0.1260	2.62110 ± 0.1043
600	2.3370 ± 0.1304	2.2744 ± 0.1095	2.6327 ± 0.2290

En la Figura 5.4, se muestran los valores de la métrica IS calculada para la GAN usando cada uno de los codificadores estudiados y el conjunto de imágenes de prueba. Dicha información fue mostrada en la Tabla 5.4 y comentada en el párrafo anterior. Adicionalmente se puede observar:

- Se observa que en ninguno de los casos existe un patrón de crecimiento o disminución continua, siendo el codificador Sent2Vec el que produce mayores saltos bruscos entre valores calculados.
- Comparando los codificadores RoBERTa y RoBERTa+CelebA, se puede ver que el modelo base es mejor hasta la época 350, después de esa época el codificador entrenado genera valores más altos.
- Durante el tiempo inicial de entrenamiento, antes de la época 300,

RoBERTa presento una disminución de valores bastante uniforme.

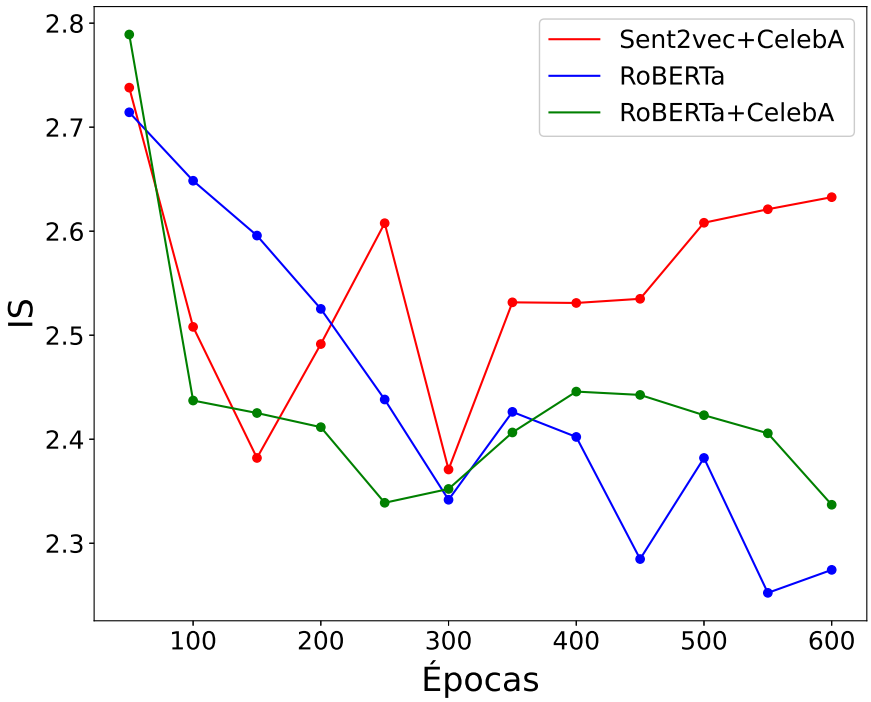


FIGURA 5.4: Variación de la métrica IS con respecto a las épocas de entrenamiento para el modelo usando cada uno de los codificadores estudiados. Fuente: Elaboración propia.

En la Tabla 5.7 se presentan los valores de la métrica LPIPS para el conjunto de imágenes de prueba definido. Se puede observar lo siguiente:

- El codificador RoBERTa+CelebA genera los mejores resultados en comparación a los otros dos modelos estudiados durante todo el entrenamiento. El mejor valor se da en la época 550.
- Comparando el modelo base RoBERTa y Sent2vec+CelebA se observa que el primero brinda mejores resultado en 9 de las 12 épocas estudiadas.
- Comparando LPIPS con la evaluación visual de las imágenes generadas, se comprueba que guardan una estrecha relación y concordancia al igual que la métrica FID y contrario a IS.

- El mejor valor para Sent2vec es 0.3036 $\pm$ 0.0843 en la época 200. Para el caso de RoBERTa su mejor valor es 0.2932 $\pm$ 0.0776. En el caso de RoBERTa+CelebA como ya se había mencionado su mejor valor es 0.2843 $\pm$ 0.0774.

TABLA 5.7: Valores de LPIPS sobre el conjunto de imágenes de prueba. Se muestra el número de épocas, y los codificadores utilizados: BoBERTa+CelebA, RoBERTa y Sent2Vec+CelebA. En negrita se muestra el mejor resultado. Fuente: Elaboración propia.

Épocas	**Codificadores**		
	RoBERTa+CelebA	**RoBERTa**	**Sent2vec+CelebA**
50	0.3326 $\pm$ 0.08640	0.3326 $\pm$ 0.0851	0.3493 $\pm$ 0.0847
100	0.3021 $\pm$ 0.0761	0.3319 $\pm$ 0.0847	0.3172 $\pm$ 0.0776
150	0.2873 $\pm$ 0.0794	0.3241 $\pm$ 0.0805	0.3098 $\pm$ 0.0746
200	0.2916 $\pm$ 0.0781	0.3135 $\pm$ 0.0843	0.3036 $\pm$ 0.0758
250	0.2873 $\pm$ 0.0777	0.3082 $\pm$ 0.0794	0.3195 $\pm$ 0.7777
300	0.2903 $\pm$ 0.0772	0.3009 $\pm$ 0.0776	0.3087 $\pm$ 0.0788
350	0.2915 $\pm$ 0.0792	0.3060 $\pm$ 0.0803	0.3157 $\pm$ 0.0745
400	0.2961 $\pm$ 0.0774	0.3047 $\pm$ 0.0792	0.3178 $\pm$ 0.0756
450	0.2921 $\pm$ 0.0797	0.2932 $\pm$ 0.0776	0.3158 $\pm$ 0.0752
500	0.2851 $\pm$ 0.0769	0.2938 $\pm$ 0.0783	0.3204 $\pm$ 0.0762
550	**0.2843 $\pm$ 0.0774**	0.2971 $\pm$ 0.0785	0.3204 $\pm$ 0.0764
600	0.2892 $\pm$ 0.0774	0.2988 $\pm$ 0.0792	0.3240 $\pm$ 0.0755

En la Figura 5.4, se muestran los valores de la métrica LPIPS calculada para el modelo generador empleando los tres codificadores estudiados. Esta información ya fue presentada en la Tabla 5.5 y comentada en el párrafo anterior. De forma complementaria se puede observar:

- Se observa que a partir de la época 200 Sent2vec+CelebA tiende a empeorar dado que se va incrementando hasta el final del entrenamiento.

- Los valores de la métrica RoBERTa+CelebA disminuyen más rápidamente y en todo momento se mantiene por debajo del modelo base. Los modelos entrenados usando los *transformers* tienden a mantener el patrón de disminución.

- Entre las épocas 100 y 150, Sent2vec+CelebA obtiene mejores resultados que nuestro modelo base RoBERTa.

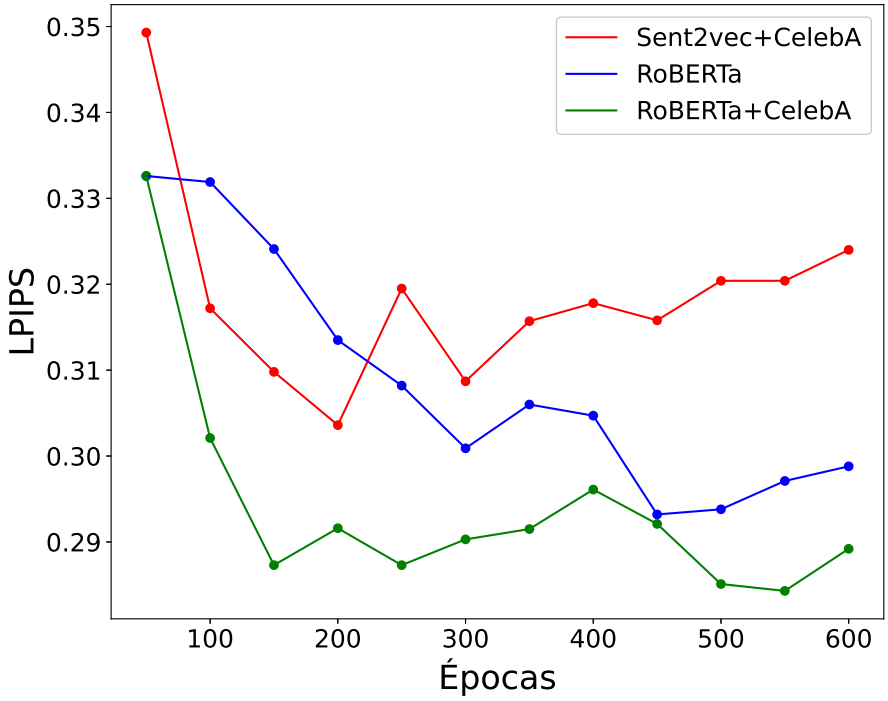


FIGURA 5.5: Variación de la métrica LPIPS con respecto a las épocas de entrenamiento para el modelo usando cada uno de los encoders estudiados. Fuente: Elaboración propia.

En la Tabla 5.8 se presenten los mejores resultados obtenidos en cada una de las métricas para la cDCGAN entrenado con cada uno de los codificadores implementados. Se observa que el modelo RoBERTa+CelebA obtiene los mejores resultados para las 3 métricas aplicadas. Por otro lado, se observa que las métricas FID y LPIPS alcanzan su mejor valor después de un tiempo de entrenamiento prolongado (durante las últimas épocas de entrenamiento)

y mayor a la mitad del tiempo total de ejecución, lo cual concuerda con los resultados visuales generados. Por el contrario, la métrica IS obtiene los valores más altos (y por tanto mejores) durante las primeras épocas. Sin embargo, en dicho tiempo los resultados visuales de las imágenes generadas son bastante deficientes, por lo que se puede afirmar que su efectividad en la evaluación de la calidad de las imágenes bastante inferior a las otras dos métricas.

TABLA 5.8: Mejores valores de las métricas de calidad usando los tres modelos estudiados. Fuente: Elaboración propia.

Codificador	**FID**	**IS**	**LPIPS**
Sent2vec+CelebA	105.5604	2.7380 ± 0.1979	0.3036 ± 0.0758
RoBERTa	89.3566	2.7143 ± 0.1769	0.2932 ± 0.0776
RoBERTa+CelebA	**84.2212**	$\mathbf{2.7891 \pm 0.1659}$	$\mathbf{0.2851 \pm 0.0769}$

5.3.2. Evaluación cualitativa de las imágenes generadas

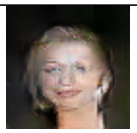
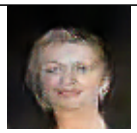
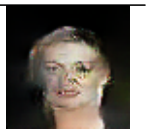
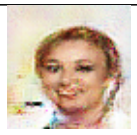
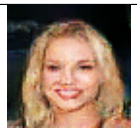
De forma adicional a la evaluación de calidad utilizando las métricas ya mencionadas, se realizó un análisis visual cualitativo de las imágenes generadas por el modelo. Esto permitió complementar los resultados de las métricas y, además, comparar si guardan relación con la apariencia visual de las imágenes generadas. Para la evaluación se utilizan 7 oraciones descriptivas seleccionadas al azar del corpus de validación. Se realizó una comparación de resultados producidos por el modelo utilizando los 3 codificadores a lo largo de las épocas de entrenamiento para una descripción puntual. Además, se evaluaron las imágenes generadas para varias descripciones textuales utilizando los mejores pesos de entrenamiento basándonos en la métrica FID.

En la Tabla 5.9 se puede observar las diferentes imágenes generadas por un mismo texto descriptivo de entrada a largo de las épocas de entrenamiento utilizando los 3 modelos como codificadores. Se puede observar lo siguiente:

- Visualmente los mejores resultados son obtenidos usando los *transformers* como codificadores, siendo por casos una mejor que la otra y viceversa.

- El tiempo de entrenamiento influye en la calidad de las imágenes generadas; más aun en el modelo Sent2vec que a partir de la época 200 ya empieza a bajar en calidad gradualmente.
- En referencia a formación de la imagen cumpliendo con las características descritas en el texto, se puede observar que el codificador RoBERTa+CelebA genera imágenes más detalladas según las características. Se observa que la calidad visual de las imágenes guarda relación directa con las métricas FID y LPIPS e inversa con IS.

TABLA 5.9: Mejores valores de las métricas de calidad usando los tres modelos estudiados. Fuente: Elaboración propia.

Entrada	La mujer tiene rostro ovalado y pomulos altos. Su cabello es de color rubio. Tiene labios grandes con cejas arqueadas y la boca ligeramente abierta. La joven y atractiva mujer sonriente tiene mucho maquillaje. Lleva aretes y lapiz labial.				
Épocas					
100	**200**	**300**	**400**	**500**	**600**
Sent2vec+CelebA					
					
RoBERTa					
					
RoBERTa+CelebA					
					

Por otro lado la Tabla 5.10 muestra las distintas salidas de rostros generados a partir de una descripción textual. Se tomaron algunos textos de pruebas del corpus de evaluación y, además, se utilizaron los pesos de la red neuronal para el generador que dan los mejores valores para FID y LPIPS. Se observa lo siguiente:

- La resolución y calidad de la imagen generadas mejoran en proporción directa a la cantidad de oraciones que tiene la descripción tal y como se observa en los casos 1, 3, 5 y 6. Por el contrario, en los casos 2 y 4 al tener solo 3 oraciones descriptivas como mínimo, las imágenes tienen menos detalles, siendo Sent2vec+CelebA el que más distorsión genera cuando tiene poca información y RoBERTa+CelebA el que genera menos.

- En algunos casos donde no se definen todas las características, el generador asigna características aleatorias como en el caso de el caso 4 donde las características de la cara son asumidas por la GAN.

- Existe la dificultad para generar imágenes con características descriptivas que no se consideraron durante el entrenamiento. Las imágenes presentan distorsiones o características aproximadas. En el caso 6, la característica *"tiene pelo corto que es de color rubio"* provoca que se genere una imagen con distorsión usando Sent2vec. Por otro lado, los *transformers* generaron imágenes más nítidas con características similares a la solicitada.

- Lo más complicado de generar son las descripciones de accesorios como corbata, lápiz labial o sombrero. Se observa que RoBERTa+CelebA genera las imágenes más completas de estos accesorios.

- El modelo RoBERTa+CelebA genera imágenes mejor elaboradas que el modelo base y Sent2vec para la mayoría de los casos.

TABLA 5.10: Generación de rostros usando los tres modelos estudiados: Sent2Vec+CelebA, RoBERTa y RoBERTa+CelebA. Fuente: Elaboración propia.

Id	Descripcion	Sent2vec+CelebA	RoBERTa	RoBERTa+CelebA
1	La mujer tiene rostro ovalado y pómulos altos. Su cabello es de color rubio. Tiene labios grandes con cejas arqueadas y la boca ligeramente abierta. La joven y atractiva mujer sonriente tiene mucho maquillaje. Lleva aretes y lapiz labial.			
2	El hombre tiene pómulos altos. Luce una sombra de las 5 en punto. Su cabello es de color negro. Tiene labios grandes y nariz grande. El joven atractivo sonríe.			
3	La mujer tiene pómulos altos. Tiene cabello lacio de color negro. Tiene labios grandes con cejas arqueadas y la boca ligeramente abierta. La joven atractiva y sonriente tiene mucho maquillaje. Lleva lápiz labial.			
4	El hombre tiene el pelo liso de color negro. Tiene la nariz puntiaguda. El hombre parece joven. Lleva corbata. .			
5	El cabello de la mujer es de color negro. Tiene labios grandes y una nariz grande y puntiaguda con cejas arqueadas. La joven atractiva tiene mucho maquillaje. Lleva aretes, collar y lápiz labial.			
6	La mujer tiene pómulos bajos. Ella tiene el pelo corto que es de color rubio. Tiene las cejas arqueadas. La mujer atractiva joven y sonriente tiene mucho maquillaje. Lleva lápiz labial.			

5.3.3. Discusión de resultados

Una vez implementada la cDCGAN junto a los 3 codificadores a estudiar, se procedió a entrenar el modelo y evaluar la calidad de las imágenes generadas utilizando las métricas objetivas IS y FID y LPIPS, acompañadas de

observaciones visuales para evaluación cualitativa de las imágenes. En todo este proceso se observaron datos y resultados interesantes los cuales se discute a continuación:

- El tiempo de entrenamiento de nuestra red cDCGAN ha sido bastante costoso para todos los casos. Esto es debido a que, en general, el entrenamiento del generador y discriminador en busca del equilibrio es complejo y susceptible a fallos. Además, en vez de capas normales se utilizaron capas convolucionales especializadas en imágenes las cuales por teoría se sabe que son relativamente lentas.

- El tiempo de entrenamiento es directamente proporcional a la cantidad de información y número de épocas definidas, viéndose en todos los casos que en el último tramo se incrementa. Una causa probable de ello es que al pasar el tiempo las imágenes generadas son más elaboradas, lo que toma más tiempo generarlas y analizarlas en el discriminador.

- Como se observa en la Tabla 5.2 el modelo para codificador RoBERTa+CelebA entrenado como parte del presente trabajo genera mejores resultados que el modelo base aplicando la correlación de Spearman para un conjunto de oraciones de prueba. Adicionalmente, se vuelve a comprobar su efectividad utilizándose en la cDCGAN al aplicar las las métricas FID y LPIPS mostradas en las Tablas 5.5 y 5.7. Ello verifica el éxito del entrenamiento del modelo RoBERTa-large-bne con el conjunto de datos CelebA en español para aumentar su rendimiento, lo cuál es uno de los objetivos específicos de la presente tesis.

- Al aplicar la métrica IS a las imágenes generadas con el generador y los 3 codificadores se obtuvieron un conjunto de valores que están por encima de 2. Comparando otros resultados como el el trabajo [17] (valor de IS 1.4 $\pm$ 0.7) del cual se tomó la arquitectura base para nuestra cDCGAN, se ve una mejora considerable aunque, hay que considerar que los idiomas del texto descriptivo son diferentes.

- Se observa que los valores de IS aumentan o disminuyen en proporción contraria a las otras dos métricas y a la calidad visual e imágenes generadas. Esto se debe a que como se menciona en [61], las IS están limitadas por el conjunto de datos utilizados durante el entrenamiento del clasificador, es decir, si no se entrenó con imágenes similares da una puntuación baja, y la capacidad del clasificador de detectar características visuales para definir el concepto de calidad de imagen. Es decir, imágenes de mala calidad pueden obtener altos puntajes.

- Las métricas FID y LPIPS si muestran resultados que van de la mano con la apariencia visual de las imágenes generadas, por lo que se puede comprobar que son métricas mucho más precisas y adecuadas para evaluación de imágenes.

- En la Tabla 5.10 se observó un caso en el cual, el texto ingresado al modelo no había formado parte del *corpus* de entrenamiento. Así, puede observarse que los *transformers* generan imágenes mucho más nítidas y/o con características similares a lo descrito. Esto se debe a que Sent2vec depende íntegramente de *corpus* usado para entrenamiento. Por el contrario, SBERT usa modelos base preentrenados y ricos en información que permiten superar la falta de características descriptivas reemplazándolas con valores similares incluidos en su preentrenamiento.

Capítulo 6

Conclusiones y Trabajo a futuro

Después de finalizar la implementación del modelo generador de imágenes a partir de texto en el idioma español, en el presente capítulo se mencionan las principales conclusiones obtenidas en el presente trabajo. Adicionalmente, se establecerán los posibles trabajos a futuro que se pueden desarrollar partiendo del modelo implementado.

6.1. Conclusiones

En relación al objetivo general, a los objetivos específicos y a los resultados obtenidos se puede concluir que:

- Se logró implementar una red cDCGAN que genera rostros guiado por texto descriptivo. Se probaron 3 diferentes codificadores: Sent2vec+CelebA, RoBERTa y el implementado por nosotros RoBERTa+CelebA, determinándose que este último genera mejores resultados.

- Se logró generar un corpus descriptivo del conjunto de datos CelebA en idioma Español. Este corpus servirá para futuros trabajos en relación al tema tratado.

- Se logro crear un corpus de entrenamiento de RoBERTa en idioma español, mediante la implementación de un algoritmo propio y a partir

de los corpus en español y en inglés de CelebA. Este corpus servirá para futuros trabajos en relación a entrenamiento del *transformer*.

- Utilizando el corpus mencionado en el punto anterior se logró entrenar el codificador tomando como base el modelo RoBERTa. El nuevo modelo se denominó RoBERTa+CelebA y mediante la evaluación de correlación de Spearman y calidad de imágenes generadas comparándolo a los otros dos codificadores estudiados, se determinó que genera mejores resultados.

- Se generó un corpus de entrenamiento para Sent2vec uniendo el conjunto de oraciones traducidas a español del corpus descriptivo original de CelebA. Este corpus servirá para futuros trabajos de entrenamiento de dicha librería.

- Se implementó la arquitectura cDCGAN tomando como base el modelo desarrollado en [46, 47] con algunas modificaciones; entre ellas el cambio de codificador por Sent2ve+CelebA, RoBERTa y RoBERTa+CelebA, eliminación del Batchnormalization en el discriminante y el cambio de LeakyReLU por ReLU en el generador.

- Se logró implementar las métricas de calidad FID, IS y LPIPS.

- Al aplicar las métricas de calidad sobre un conjunto de 1000 imágenes de prueba se determinó que el modelo RoBERTa+CelebA genera los mejores resultados para FID y LPIPS.

- La evaluación visual cualitativa ratifica los valores calculados de la métrica FID y LPIPS. En ambos casos las imágenes de mejor calidad se obtuvieron en 600 épocas para el entrenamiento y usando como codificador nuestro modelo entrenado RoBERTa+CelebA.

- Se mostró que IS no es una métrica objetiva a tener en cuenta en este tipo de experimentos debido a un desempeño totalmente aleatorio en la relación de evaluación cuantitativo-cualitativo, por lo que se recomienda utilizar FID y LPIPS que son más confiables y con rendimiento estable

y robusto durante todo el entrenamiento de acuerdo con las imágenes generadas por nuestro modelo.

- Nuestra investigación muestran resultados prometedores con respecto a la literatura al mejorar las métricas numéricas de FID en un 5 % y LPIPS en un 37 %.

- Una comparación ideal entre las métricas implementadas amerita que la implementación de las referencias también sea en español, dado que la facilidad de recursos facilita la obtención de mejores resultados para las métricas en idioma inglés.

- Nuestra implementación uno de los primeros trabajos de redes DCGAN condicionadas a texto en el idioma español. Ello servirá como base a otros trabajos relacionados al tema.

6.2. Trabajo a futuro

A partir del trabajo desarrollado y tomando en consideración los problemas y limitaciones e ideas de mejora pensados en el camino se pueden plantear los siguientes trabajos a futuro:

- Entrenar el modelo GAN con una mayor cantidad de imágenes y sus respectivos *captions* y durante más tiempo. Se espera a que obtendrán mejores resultados para las métricas estudiadas.

- Utilizar un corpus más grande para el entrenamiento de Sent2vec y SBERT como por ejemplo *Spanish Unannotated Corpora* [1]

- Hacer un estudio comparando la eficiencia de Sentence-BERT con otras arquitecturas, como las que están en el estado del arte.

[1] `https://github.com/josecannete/spanish-corpora`

Bibliografía

[1] Personas detenidas por comisión de delitos (flagrancia), según tipo de delito. https://m.inei.gob.pe/estadisticas/indice-tematico/seguridad-ciudadana/. [Online: Último acceso 04-oct-2022].

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, oct 2020.

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

[4] Jorge Agnese, Jonathan Herrera, Haicheng Tao, and Xingquan Zhu. A survey and taxonomy of adversarial neural networks for text-to-image synthesis. *WIREs Data Mining and Knowledge Discovery*, 10(4):e1345, 2020.

[5] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[6] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

Volume 1 (Long Papers), pages 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. `https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`. [Online: Último acceso 18-mar-2023].

[9] Asier Gutiérrez Fandiño, Jordi Armengol Estapé, Marc Pàmies, Joan Llop Palao, Joaquin Silveira Ocampo, Casimiro Pio Carrino, Carme Armentano Oller, Carlos Rodriguez Penagos, Aitor Gonzalez Agirre, and Marta Villegas. Maria: Spanish language models. *Procesamiento del Lenguaje Natural*, 68, 2022.

[10] Asier Gutiérrez Fandiño, Jordi Armengol Estapé, Marc Pàmies, Joan Llop Palao, Joaquin Silveira Ocampo, Casimiro Pio Carrino, Carme Armentano Oller, Carlos Rodriguez Penagos, Aitor Gonzalez Agirre, and Marta Villegas. Roberta base trained with data from national library of spain (bne). `https://huggingface.co/PlanTL-GOB-ES/roberta-base-bne`. [Online: Último acceso 18-mar-2023].

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[12] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[13] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual

metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] Generative adversarial networks with python - deep learning. `https://machinelearningmastery.com/`. [Online: Último acceso 22-dic-2022].

[15] Redes neuronales generativas adversarias (gans). `https://www.iartificial.net/redes-neuronales-generativas-adversarias-gans`. [Online: Último acceso 22-dic-2022].

[16] Kailash Ahirwar. *Generative Adversarial Network Project book*, chapter 1. Packt, 2019.

[17] Vladimir Bok Jakub Langr. *GANs in action - Deep Learning with Generative Adversarial Networks book*, chapter 1,2. Manning, 2019.

[18] Yang Yu, Zhiqiang Gong, Ping Zhong, and Jiaxin Shan. Unsupervised representation learning with deep convolutional neural network for remote sensing images. In *Image and Graphics: 9th International Conference, ICIG 2017, Shanghai, China, September 13-15, 2017, Revised Selected Papers, Part II 9*, pages 97–108. Springer, 2017.

[19] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible conditional gans for image editing, 2016.

[20] Hyojin Park, YoungJoon Yoo, and Nojun Kwak. Mc-gan: Multi-conditional generative adversarial network for image synthesis, 2018.

[21] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[22] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[23] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1505–1514, 2019.

[24] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. Storygan: A sequential conditional gan for story visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6329–6338, 2019.

[25] Yitong Li, Martin Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[26] A simple explanation of the inception score. `https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a`. [Online: Último acceso 04-ene-2023].

[27] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.

[28] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[30] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5mb model size, 2016.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[32] What are sentence embeddings and their applications? `https://blog.taus.net/what-are-sentence-embeddings-and-their-applications`. [Online: Último acceso 20-oct-2022].

[33] Mayak Rasu Aman Kedia. *Hands-On Python Natural. Language Processing Explore*, chapter 14. Packt, 2020.

[34] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation, 2013.

[35] Implementing deep learning methods and feature engineering for text data: The continuous bag of words (cbow). `https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-cbow.html`. [Online: Último acceso 24-oct-2022].

[36] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[37] Sentence transformers: Meanings in disguis. `https://www.pinecone.io/learn/sentence-embeddings`. [Online: Último acceso 24-oct-2022].

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[39] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[40] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. *Advances in neural information processing systems*, 28, 2015.

[41] A Conneau, D Kiela, H Schwenk, L Barrault, and A Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics, 2017.

[42] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.

[43] Zeynep y Yan Xinchen y Logeswaran Lajanugen y Schiele Bernt Reed, Scott y Akata and Honglak Lee. Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR, 2016.

[44] Integrating vision and language for automatic face descriptions. Master's thesis, Universidad de Coimbra, 2018.

[45] Jian Zhao, Xie Xie, Lin Wang, Meng Cao, and Miao Zhang. Generating photographic faces from the sketch guided by attribute using gan. *IEEE Access*, 7:23844–23851, 2019.

[46] Osaid Rehman Nasir, Shailesh Kumar Jha, Manraj Singh Grover, Yi Yu, Ajit Kumar, and Rajiv Ratn Shah. Text2facegan: Face generation from fine grained textual descriptions. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 58–67. IEEE, IEEE Computer Society, 2019.

[47] O. R. Nasir, S. K. Jha, M. S. Grover, Y. Yu, A. Kumar, and R. R. Shah. Text2facegan: Face generation from fine grained textual descriptions.

https://github.com/midas-research/text2facegan. [Online: Último acceso 18-mar-2023].

[48] Ryan Cain, Gabriel Kralik, and Campbell Munson. Photo-realistic image synthesis from text descriptions. 2020.

[49] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[50] Manan Oza, Sukalpa Chanda, and David Doermann. Semantic Text-to-Face GAN-ST∧2FG, 2021.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[53] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[54] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Multi-modal-celeba-hq. https://github.com/IIGROUP/MM-CelebA-HQ-Dataset. [Online: Último acceso 18-mar-2023].

[55] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip Torr. Controllable text-to-image generation. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] Kalpana Deorukhkar, Kevlyn Kadamala, and Elita Menezes. Fgtd: Face generation from textual description. In *Inventive Communication and*

Computational Technologies: Proceedings of ICICCT 2021, pages 547–562. Springer, 2022.

[57] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[58] Ming Tao, Hao Tang, Fei Wu, Xiao-Yuan Jing, Bing-Kun Bao, and Changsheng Xu. Df-gan: A simple and effective baseline for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16515–16525, 2022.

[59] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11410–11420, 2022.

[60] How to train sent2vec model. `https://ilmoirfan.com/how-to-train-sent2vec-model`. [Online: Último acceso 11-dic-2022].

[61] Shane Barratt and Rishi Sharma. A note on the inception score. 2018.

Anexo A

Recursos para el entrenamiento de los codificadores del modelo cDCGAN

En esta sección se describirá a detalle los pasos del algoritmo propio implementado para generar un corpus de entrenamiento para SBERT a partir de los corpus en inglés y español del conjunto de datos CelebA. Además, se describe el comando de entrenamiento del codificador Sent2vec usando el corpus de entrenamiento en español. Finalmente, se muestran los enlaces de los repositorios donde se publicaron los recursos utilizados en la presente tesis.

A.1. Algoritmo para generar el corpus de entrenamiento para SBERT

Para el entrenamiento de SBERT es necesario contar con un corpus de entrenamiento el cual debe esta compuesto por dos oraciones en español y su respectivo valor de similaridad. Sin embargo, actualmente no existen librerías confiables que evalúen la similaridad entre oraciones en un idioma diferente al inglés. Es por ello que para el presente trabajo se creo un algoritmo simple para crear un corpus utilizando tanto el corpus en inglés como en español. Como se

puede observar en el Algoritmo 1 el proceso consta de los siguientes pasos:

- Se recibe como entrada a los corpus en inglés y español. Como salida se tendrá un corpus en el cual cada entrada estará compuesto por dos oraciones en español y su respectivo valor de similaridad calculado a partir de sus pares en inglés.
- Guardar en listas cada una de las oraciones de los dos corpus en español e inglés.
- Fijar la cantidad de entradas que tendrá el nuevo archivo generado, así como también el índice máximo de búsqueda de oraciones. Para el presente trabajo se fijo el valor de entradas a 250000 y el índice de búsqueda a 192050, el cual es la cantidad de oraciones que tienen los archivos.
- En cada iteración selecciona aleatoriamente dos índices del rango establecido y se recupera las oraciones en español e inglés de dichos índices.
- Utilizando el par de oraciones en inglés se calcula su similaridad usando alguna librería o método confiable. Para el presente trabajo se utilizó la librería `Spicy`.
- Se guarda una entrada en el nuevo archivo. Dicha entrada esta compuesta por el par de oraciones en español antes seleccionados junto al valor de similaridad calculado.

Algoritmo 1: Generación de corpus para SBERT en español

Entradas: *Corpus de oraciones descriptivas en español e inglés*;

Salida: *Corpus de entrenamiento para el transformer*;

$ListaEntradaEs \leftarrow \{Oes_1, Oes_2, Oes_3, ...\}$ ▷ Corpus en español;

$ListaEntradaEn \leftarrow \{Oen_1, Oen_2, Oen_3, ...\}$ ▷ Corpus en inglés;

$Nmax \leftarrow 250000$ ▷ Cantidad de entradas de entrenamiento deseado;

$Emax \leftarrow 192050$ ▷ Cantidad de oraciones de los archivos de entrada;

$n \leftarrow 1$;

while $n \leq Nmax$ **do**

    $x \leftarrow aleatorio(0, Emax)$;

    $y \leftarrow aleatorio(0, Emax)$;

    $oracion1 \leftarrow ListaEntradaEn[x]$;

    $oracion2 \leftarrow ListaEntradaEn[y]$;

    $similaridad \leftarrow evaluarSimilaridad(oracion1, oracion2)$;

    $Escribir:$

     $ListaEntradaEs[x] + "|" + ListaEntradaEs[y] + "|" + similaridad$;

    $n \leftarrow n + 1$;

A.2. Entrenamiento de Sent2vec con el corpus en español

Sent2vec al igual que otros algoritmos como Skip-Thought, a priori fueron entrenados para ser usados en el idioma inglés. Es por ello que para el presente trabajo es necesario entrenarlo previamente utilizando un corpus en español. El comando de entrenamiento se ejecuta de la siguiente forma:

```
./fasttext sent2vec -input s2v_train.txt -output
sent2vec_text2facev2 -minCount 8  -dim 4800 -epoch 5000
-lr 0.05 -wordNgrams 2 -loss ns -neg 10 -thread 200
-t 0.000005 -dropoutK 4 -minCountLabel 20 -bucket 4000
-numCheckPoints 10
```

A continuación, se describe a los argumentos del comando de entrenamiento de la libreria `sent2vec`:

- Nombre del archivo de entrada con el corpus fue *caps_es_train_prec.txt* contiene el conjunto de párrafos pre-procesados en idioma español.
- Nombre del archivo de salida se definió como *sent2vec_text2facev2* para este ejemplo, se generó un archivo de extensión *bin* que almacena el modelo entrenado y se pasará como parámetro cuando se llame a la librería desde Python.
- Dimensión del vector de características se define a 4800, dicho valor fue elegido tomando como referencia el tamaño del vector de características.
- Número de épocas de entrenamiento se define a 5000, un número grande para mejorar el entrenamiento.
- Número de hilos se define en 200, un número grande para aumentar la rapidez del procesamiento.
- Longitud máxima de palabras a seleccionar para los n-gramas se define a 2, el cual es su valor por defecto.

- El ratio de aprendizaje medio se define a 0.05, un valor bastante común para este tipo de entrenamiento.

A.3. Repositorios de datos y modelos utilizados

Durante el desarrollo de la presente tesis se han generado varios recursos (corpus de entrenamiento, modelos y códigos implementados), los cuales pueden ser utilizados en futuros trabajos de investigación y/o pruebas en general. Esta información fue publicada en repositorios públicos para que los lectores puedan revisarlos y descargarlos. Cada repositorio contiene una descripción de la información publicada e incluye un manual de uso. A continuación, se listan los enlaces donde se podrá acceder a dichos materiales:

- Corpus de entrenamiento en español para el codificador Sent2vec: `https://huggingface.co/datasets/oeg/CelebA_Sent2Vect_Sp`
- Corpus de entrenamiento en español para el codificador RoBERTa+CelebA: `https://huggingface.co/datasets/oeg/CelebA_RoBERTa_Sp`
- Modelo del codificador Sent2vec+CelebA entrenado con el corpus descriptivo de CelebA en español: `https://huggingface.co/oeg/Sent2vec_CelebA_Sp`
- Modelo del codificador RoBERTa+CelebA entrenado con el corpus descriptivo de CelebA en español: `https://huggingface.co/oeg/RoBERTa-CelebA-Sp`