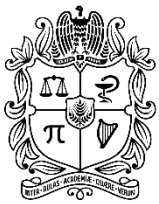


UNIVERSIDAD
NACIONAL
DE COLOMBIA

MÉTODO DE COMPRESIÓN DE ARCHIVOS DE IMAGEN USANDO TÉCNICAS DE DEEP LEARNING

Mario Varas González

Facultad de Minas
Universidad Nacional de Colombia
Sede Medellín
2022

MÉTODO DE COMPRESIÓN DE ARCHIVOS DE IMAGEN USANDO TÉCNICAS DE DEEP LEARNING

Mario Varas González

Trabajo Final de Maestría presentado como requisito parcial para optar al título de:
Magíster en Ingeniería – Ingeniería de Sistemas

Director
John Willian Branch Bedoya, Ph.D.
Departamento de Ciencias de la Computación y de la Decisión

Facultad de Minas
Universidad Nacional de Colombia
Sede Medellín

2022

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

Mario Varas González

Fecha 27/07/2022

Agradecimientos

Al profesor John Willian Branch, por acoger el proyecto, y por su acompañamiento y guía a lo largo de todo el proceso. Gracias por su esmero en fomentar espacios de divulgación y transferencia de conocimiento entre toda la comunidad académica.

A Carlos Salazar por sus comentarios e ideas que inspiraron líneas de trabajo interesantes para el proyecto y para su desarrollo.

A la Universidad Nacional de Colombia por varios años de formación, apoyo y aprendizaje constante, tanto académico como personal.

Y a todas las personas que mostraron su apoyo durante todo el proceso de desarrollo y ejecución de este trabajo.

Índice de Contenidos

Lista de figuras

Lista de tablas

Lista de abreviaturas

Abreviatura	Término
API	Application Programming Interface
CAE	Convolutional Autoencoder
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EDD	Experiment Driven Development
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IA	Inteligencia Artificial
IoT	Internet of Things
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PSNR	Peak Signal-to-noise Ratio
SSIM	Structural Similarity Index

Resumen

En los últimos años, el tráfico en internet ha estado mayormente dominado por aplicaciones relacionadas con archivos de imagen y vídeo, especialmente servicios de *streaming* de contenido y aplicaciones de distribución de video bajo demanda.

Más de tres cuartas partes del tráfico total de internet corresponden a archivos de imagen y vídeo. Que estas tareas sean lo más eficientes posible repercute directamente en la experiencia de uso que tengan los usuarios y en la calidad del servicio prestado. Preservar la calidad de esta experiencia de usuario es el principal objetivo en el desarrollo de estos sistemas de compresión, así como el punto donde estos sistemas más pueden flaquear. Es por ello que minimizar la distorsión o pérdida de información generada en el proceso de compresión de un archivo es algo prioritario y un asunto que ha tratado de abordarse desde diversas perspectivas y métodos a lo largo de la historia.

El presente trabajo se centra en aquellas propuestas de reciente publicación donde el aprendizaje profundo o *Deep Learning* juega un papel principal en este proceso, proponiendo un método basado en redes neuronales para enfrentar el problema de compresión de archivos de imagen, mostrando la investigación llevada a cabo, el desarrollo del método y su puesta a prueba.

Palabras clave: compresión de imágenes, aprendizaje profundo, procesamiento de imágenes

Abstract

In recent years, Internet traffic has been largely dominated by applications related to image and video files, especially content streaming services and video-on-demand distribution applications.

Today, more than three quarters of all Internet traffic is image and video files. Making these tasks as efficient as possible has a direct impact on the user experience and the quality of the service provided. Preserving the quality of this user experience is the main objective in the development of these compression systems, as well as the point where these systems can falter the most. That is why minimizing the distortion or loss of information generated in the file compression process is a priority and an issue that has been addressed from various perspectives and methods throughout history.

This project focuses on those recently published proposals where Deep Learning plays a major role in this process, proposing a method based on neural networks to address the problem of image files compression, showing the research carried out, the development of the method and its testing.

Title: ***Image files compression method using Deep Learning techniques***

Keywords: image compression, Deep Learning, image processing

1. Introducción.

1.1 Motivación

Comprimir archivos es un proceso que consiste en tomar un archivo y reducir su tamaño conservando con la mayor precisión posible su información. Este precepto tan simple puede llevar a mejoras en el manejo y la transmisión de dicho archivo y, consecuentemente, a mejorar el desempeño de un sistema software concreto que haga uso de tareas de manejo y transmisión de archivos, es decir, la mayoría de los sistemas software presentes, por ejemplo, en la web.

Con el paso de los años, los métodos y algoritmos tradicionales de compresión de archivos han ido mejorando y superándose a si mismos precisamente porque mejorar estos sistemas de compresión implica directamente mejorar la velocidad y desempeño de buena parte del software existente presente en la red.

En los últimos años, los sistemas y algoritmos tradicionales de compresión de archivos han ido siendo sustituidos en algunos casos por alternativas venidas del mundo del aprendizaje profundo o *Deep Learning*. Por ejemplo, para el caso particular de compresión de imágenes, se han desarrollado modelos que demuestran conseguir mejores resultados que los más famosos códecs de imagen como JPEG (Wallace, 1992). Todo este esfuerzo por desarrollar mejores compresores por el lado del aprendizaje profundo responde a la necesidad de hacer Internet más rápido y eficiente frente al incipiente incremento de tráfico especialmente en temas de imagen y vídeo, siendo éste el principal tráfico presente en toda la red, constituyendo tres cuartas partes del tráfico total. Por otro lado, también es debido a la reciente demanda de mayores resoluciones y *bitrates* por ejemplo en el caso de aplicaciones de *streaming* o video sobre demanda (VOD), que tan populares vienen siendo entre los usuarios en la última década.

Podemos, partiendo de estas ideas previas, hacer una doble afirmación. En primer lugar, que investigar acerca de nuevos métodos de compresión de imagen, o archivos en general, es sinónimo de contribuir a un Internet más escalable y eficiente y, en segundo lugar, enlazado a la primera afirmación, podemos decir que investigar nuevos métodos de compresión de archivos está relacionado con mejorar cualitativamente la experiencia de los usuarios.

Siendo más absolutos, se puede afirmar que mejores sistemas de compresión de archivos se relacionan con que Internet funcione más rápido. A modo de conclusión para esta introducción, podemos decir que es

precisamente esta doble idea la motivación principal y más fuerte del presente proyecto: tratar de conseguir métodos y líneas de trabajo que lleven a agilizar la compresión de archivos para de esta manera conseguir que un software que emplee dicho método o sistema se comporte más rápido y de manera más eficiente en cuanto a su desempeño, eficiencia y costo, resultando todo esto en ofrecer al usuario final una mejor experiencia de uso.

1.2 Trabajos previos

Los primeros trabajos que abordaban el tema de compresión de imágenes usando Inteligencia Artificial datan de los años 90. El objetivo era buscar alternativas a los algoritmos tradicionales de compresión de datos que funcionaran para el caso de las imágenes y ofrecieran unos resultados superiores, llegando hipotéticamente a reemplazar a estos algoritmos tradicionales. Este es un camino que aún a día de hoy se sigue recorriendo pero, aun así, los resultados que se han ido obteniendo en la última década con algunos métodos propuestos, demuestran que puede llegarse a métodos basados en aprendizaje de máquinas y aprendizaje profundo que superen, en términos de subjetivos y objetivos, a los métodos más tradicionales y establecidos, llevando a algunas de las grandes empresas de servicios tecnológicos en el mundo a dar el paso de adoptarlos.

En el punto 3 del presente documento se realiza una revisión literaria de todos estos estudios y de los previos, haciendo un recorrido cronológico por las diferentes técnicas de compresión de imagen más vinculadas al presente proyecto, destacando sobre la marcha aquellas publicaciones que más interesantes han resultado de cara al desarrollo del mismo.

En el contexto del método propuesto, ha habido un grupo de estudios relacionando con aprendizaje profundo y compresión de imágenes que han sido inspiración y referente a la hora de desarrollar ideas para terminar planteando un método efectivo de compresión de imágenes.

En concreto, se ha hecho una selección de tres, los publicados por Theis et al., Cheng et al. y Akyazi et al., que han propuesto métodos basados en *Deep Learning* que se comparan y superan el rendimiento de JPEG y JPEG2000.

1.3 Descripción del problema

El presente trabajo final de maestría desarrolla un método basado en aprendizaje profundo (*Deep Learning)* para realizar un proceso de compresión a archivos de imagen. Este método ofrece una compresión con pérdida al estilo de JPEG que permite partir de un archivo de imagen A y llegar a obtener un archivo B que representa una versión comprimida y equivalente del primero.

1.4 Objetivos

1.4.1 Objetivo general

Proponer un método que permita realizar un proceso de compresión en imágenes digitales empleando técnicas de aprendizaje profundo.

1.4.2 Objetivos específicos

- Seleccionar un conjunto de datos compuesto por imágenes digitales que garantice unos estándares de calidad apropiados.
- Diseñar un método para la compresión de imágenes digitales basado en técnicas de aprendizaje profundo.
- Validar el desempeño del método de compresión propuesto contrastándolo con otros métodos populares y con trabajos reportados en la literatura.

1.5 Contribución

Este trabajo buscar generar un complemento a los estudios existentes sobre compresión de archivos de imagen con técnicas de *Deep Learning* y proponer un método que aporte resultados convincentes y útiles para ser aplicado en casos de uso reales. Se analiza el problema desde su concepto hasta su desarrollo más avanzado, logrando tratar el problema de compresión de imágenes desde la perspectiva del aprendizaje profundo, y beneficiándose de éste para proponer un método de utilidad real. El modelo propuesto pretende ser una base funcional y efectiva para el problema de compresión de imágenes, así como resultar adaptable y escalable en el tiempo a otros escenarios o casos de uso similares.

1.6 Estructura del documento

En el presente documento se ilustra el proceso de investigación realizado para llegar al método de compresión de imágenes propuesto, así como el contexto sobre el tema central de investigación y una serie de resultados y reflexiones fruto de todo el proceso.

En el capítulo 2, se recoge una explicación teórica acerca del problema de compresión de imágenes y las recientes aportaciones por parte del aprendizaje profundo a dicho problema, argumentando cómo y en qué escenarios resulta especialmente útil e interesante la aplicación de técnicas basadas en aprendizaje profundo a la hora de realizar un proceso de compresión de imágenes. Por otro lado, en el capítulo 3, se realiza una revisión de la literatura existente en la actualidad acerca de este tema, mencionando aquellos estudios e investigaciones publicadas que más relevantes han resultado de cara a la realización del proyecto y que más han inspirado el proceso de investigación y el desarrollo del método propuesto.

La arquitectura y detalles del diseño de dicho método, así como su desarrollo e implementación, su posterior evaluación y puesta a prueba, se indican de forma extensa en los capítulos 3 y 4. Consecuentemente, a continuación, en el capítulo 5, se hace referencia al tercer objetivo específico definido, mostrando un análisis objetivo y subjetivo del método propuesto, contrastándolo principalmente con los estándares de compresión de imagen JPEG y JPEG2000 y mostrando los resultados obtenidos.

Por último, el capítulo 6, recoge una serie de conclusiones obtenidas a lo largo del desarrollo del proyecto, así como reflexiones e ideas de trabajo futuro al respecto de este tema, indicando finalmente una serie de propuestas bien definidas que serán realizadas con posterioridad.

2. Compresión de archivos de imagen.

En ciencias de la computación, comprimir datos es, esencialmente, reducir el volumen de éstos para representar una determinada información empleando una menor cantidad de espacio y conservando con toda o con la mayor fidelidad posible la integridad de estos datos. Al acto de compresión de datos se le denomina compresión, y al proceso antagónico o inverso, descompresión.

El proceso de compresión de datos visto a alto nivel supone simplemente partir de un ítem *A*, aplicarle un método de compresión, y obtener un ítem *B* copia del *A* pero con un tamaño reducido.
Esta reducción del volumen de datos, yendo al nivel de abstracción justamente inferior al anterior, consiste en reducir el número de *bits* necesarios para representar la información. Un *bit* es la unidad mínima de medida de información que se maneja en ciencias de la computación, admitiendo únicamente dos valores, 0 y 1, siendo por ello una unidad binaria. Adicionalmente, tenemos también los *bytes*, que es la unidad de información inmediatamente superior al *bit* y que equivale a ocho *bits*. Estas medidas se utilizan frecuentemente para especificar espacio de almacenamiento, por ejemplo, la cantidad de memoria de un determinado dispositivo, la capacidad de almacenamiento que posee o el espacio que ocupa en memoria un determinado archivo.

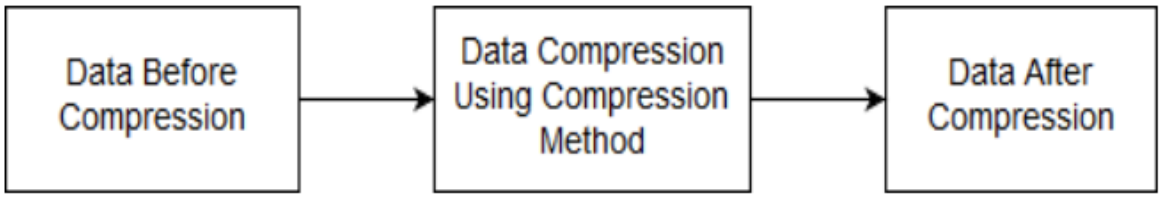


Figura 1 - Esquema básico del proceso de compresión de datos

Comprimir datos ayuda a reducir el espacio que estos ocupan, a agilizar su transferencia y a ahorrar costos de almacenamiento. Para llevar a cabo esto es necesario contar con un método que haga uso de una fórmula o algoritmo de compresión o en su defecto un algoritmo que posibilite el proceso de compresión de los datos.

La utilidad principal de comprimir datos reside principalmente en dos puntos, ya mencionados. Por un lado, a la hora de almacenar esos datos y por otro lado a la hora de transmitirlos o enviarlos a un destino a través de un canal. Comprimir datos puede reducir drásticamente el tamaño de un archivo que los contenga. Por ejemplo, si aplicamos una ratio de compresión de 2:1 a un archivo con un peso de 20 kB (*kilobytes*),

obtendríamos una versión comprimida del mismo con un peso de 10 kB. Con resultado de esta transformación, su espacio requerido de almacenamiento será menor y por lo tanto almacenarlo y mantenerlo en el tiempo guardado costará menos recursos. También, de cara al futuro y para archivos con un tamaño no constante, es decir, que sean modificados en el tiempo aumentando o reduciendo su tamaño, la compresión de datos será algo imprescindible para ahorrar costos y optimizar resultados. Por el lado de la transmisión de los datos, comprimir datos ayudará a minimizar el ancho de banda necesario para llevarlos de un punto *A* a un punto *B*, ahorrando por consiguiente costos asociados al uso del canal de transmisión empleado.

Desde otro punto de vista, también se pueden señalar algunos pequeños contras que tiene realizar compresión a archivos. La principal desventaja de la compresión de datos es el impacto en el rendimiento resultante en cuanto a uso de recursos de CPU y memoria para comprimir los datos y realizar posteriormente la descompresión de los mismos. Es por esto que muchos proveedores han diseñado sus sistemas para tratar de minimizar el impacto computacional que suponen los cálculos intensivos del procesador necesarios para llevar a cabo la compresión. Si la compresión se ejecuta en línea, antes de que los datos se escriban en el disco, el sistema puede descargar la compresión para preservar los recursos del sistema. También, los sistemas de caché son aplicables a estos casos para minimizar los recursos empleados en todo el proceso.

Se puede comprimir prácticamente cualquier tipo de archivo, pero es importante seguir las mejores prácticas al elegir cuáles comprimir. Por ejemplo, es posible que algunos archivos ya vengan comprimidos, por lo que comprimir esos archivos no tendría un impacto significativo.

Existen dos formas de comprimir datos, con pérdida y sin pérdida. La compresión sin pérdida permite la restauración de un archivo a su estado original, sin la pérdida de un solo *bit* durante el proceso. La compresión sin pérdida es el enfoque típico con archivos ejecutables, al igual que con archivos de texto y hojas de cálculo, donde la pérdida de palabras o números perturbaría la información resultante.

Para el caso específico de las imágenes, podemos aplicar una compresión con pérdida o sin pérdida. Los formatos de archivos de imágenes digitales suelen estar diseñados para comprimir información, ya que los archivos tienden a ser grandes. Por ejemplo, JPEG y MPEG son formatos de archivos de imagen que admiten la compresión de imágenes con pérdida, mientras que otros formatos como GIF y PNG utilizan compresión sin pérdidas.

Por otro lado, la idea principal es combinar un proceso de compresión con técnicas basadas en aprendizaje profundo o *Deep Learning*. Diversos estudios publicados en los últimos años demuestran que los resultados de los métodos tradicionales de compresión de imágenes pueden ser superados por métodos nuevos con base en modelos de aprendizaje profundo.

El aprendizaje profundo o *Deep Learning* es un campo del aprendizaje de máquinas que se enfoca en la creación de técnicas y modelos capaces de extraer altos niveles de abstracción de datos a través de métodos no supervisados, con el objetivo de conseguir una nueva representación de dichos datos que facilite la tarea de predicción. Estas técnicas han sido aplicadas con éxito en diversos campos como la visión artificial, el análisis de imágenes, el procesamiento de lenguaje natural y la bioinformática. Conceptualmente, podemos decir que el aprendizaje profundo lleva como principal premisa entrenar a una máquina a comportarse similarmente a cómo lo hacen los seres humanos para ciertos problemas y contextos concretos, siendo el caso particular de este trabajo el problema de compresión de imágenes en un contexto general del mismo.

3. Revisión de la literatura.

El proceso de revisión de la literatura ha supuesto una parte esencial de todo el desarrollo del proyecto, por el lado de los conocimientos teóricos y técnicos adquiridos y también, y principalmente, por la inspiración surgida en base a dicha revisión que ha resultado muy útil para desarrollar ideas propias que poder llevar a cabo para satisfacer los objetivos planteados.

De forma esquemática, y con fines de organizar a alto nivel en periodos la literatura publicada en torno al tema de compresión de imágenes, se han identificado tres periodos históricos.

El primero de ellos recoge los primeros algoritmos empleados en procesos de compresión de datos, mayormente basados en métodos matemáticos y estadísticos. Por ejemplo, el algoritmo de Huffman (Dhawale, 2015), que empleaba arboles binarios para representar de forma ordenada los datos y generar su versión codificada, distinguiendo los símbolos más frecuentes de los menos frecuentes, o la codificación de Golomb (S. Golomb, 1966), aplicada en primer lugar en escenarios sencillos como codificar alfabetos de dos símbolos o alfabetos sencillos de símbolos ponderados y posteriormente extendida a casos más complejos. Existen otros, como el algoritmo LZW (Ahlswede et al., 2014), que a diferencia del algoritmo de Huffman emplea un diccionario en vez de un árbol binario para representar y ordenar los datos sobre la marcha.

Más adelante, surgieron otros métodos más complejos y de aplicación más específica, muchos de ellos basados e inspirados en estos anteriores. Aquí merece la pena hablar de JPEG (*Joint Photographic Experts Group*) (Wallace, 1992), un célebre método y estándar de compresión y codificación de imágenes digitales que también da nombre al formato de los archivos de imagen que produce. JPEG lleva siendo un estándar de compresión de imágenes digitales desde el año de su publicación, 1992. Su éxito se debe a ser un método rápido y que da resultados, a pesar de la pérdida, de muy buena calidad. Podemos decir entonces que JPEG es el método y a la vez un formato de archivo. JPEG ofrece compresión con pérdida y es usado y soportado en la actualidad por parte de la mayoría de las cámaras fotográficas digitales existentes, así como en navegadores web, sistemas operativos y programas de edición de imágenes. Cabe destacar su uso en la web, que proporciona un incremento significativo en la estabilidad y desempeño de numerosas aplicaciones web que hacen un uso intensivo de imágenes digitales, permitiendo que éstas carguen y se procesen mejor y más rápido, optimizando en conjunto su funcionamiento.

Es frecuente tender a pensar que JPEG y JPG son formatos diferentes, cuando en realidad aluden al mismo formato, JPEG. JPG es el formato que se utiliza en el sistema operativo Windows y en los PC, que suelen trabajar con extensiones de tres caracteres.

En cuanto al funcionamiento de JPEG, debemos en primer lugar señalar que garantizar una buena compresión con pérdidas, causadas por la tarea de desechar la información de la imagen que carece de mucha relevancia. Aquí el reto es decidir qué información conservar y cuál no. Afrontar este reto comienza por comprender qué partes de una imagen son importantes para la percepción humana y cuáles no, es decir, abordar el tema en primera instancia desde un punto de vista subjetivo y no objetivo. En JPEG, el proceso de compresión con pérdida se basa en dos principios psicovisuales. En primer lugar, el que los cambios en el brillo de la imagen son más importantes que los cambios de color. Esta priorización se basa en el hecho biológico de que la retina humana contiene un aproximado de 120 millones de células sensibles a la luz, pero tan solo 6 millones de células sensibles al color, por lo tanto, es considerablemente más sensible a los cambios de luz que a los cambios de color. El segundo principio es que los cambios de baja frecuencia son más relevantes al ojo humano que los cambios de alta frecuencia. El ojo humano es bueno recibiendo e identificando cambios de luz de baja frecuencia, como por ejemplo los límites de un plano o las esquinas de un objeto, pero es menos preciso con los cambios de luz de alta frecuencia, como texturas o patrones en alguna superficie. En este mismo principio biológico del ojo humano es en lo que se basa también el camuflaje, es decir, en distorsionar las frecuencias bajas empleando más frecuencias altas de luz para confundir al ojo para que no identifique ciertas figuras.
JPEG aplica estas dos ideas simultáneamente. En cada caso, los datos de la imagen se transforman para dar un acceso más sencillo al tipo de información necesaria y deseada, ya sea información de brillo o de frecuencia. A continuación, se decide qué parte de la información puede considerarse de menos relevancia y se procede a descartarla para finalmente preservar la información restante para comprimirla en el menor espacio posible.

De un modo más detallado y desde el punto de vista técnico, los pasos seguidos por JPEG son los siguientes:

1. Separar los canales de color de la imagen usando el espacio de color YCbCr. Este espacio de color, a diferencia de RGB que sólo guarda información de color, incluye la información del brillo en el canal Y, y separadamente almacena la información del color en los otros dos canales, Cb y Cr. Estos dos son, respectivamente, diferencia del azul (relativiza la imagen entre azul y rojo) y diferencia del rojo (relativiza la imagen entre verde y rojo). Ambas señales son conocidas como crominancia o información de color.

2. Empleando los canales Cb y Cr, parte de la información del color es descartada mientras que la información del brillo presente en el canal Y se mantiene.
3. Pasando a la segunda idea en la que se basa JPEG, la referente al tema de las frecuencias de luz, se divide cada uno de los tres canales en bloques de 8x8 píxeles. Estos bloques se transforman de la representación en la que están, el dominio espacial, a un dominio de frecuencias. De esta manera, el valor de más arriba a la izquierda de la imagen, que en el domino espacial representaba el canal Y del píxel correspondiente, va a representar la frecuencia más baja, y así progresivamente hasta llegar al valor con la frecuencia más alta, el de la posición más inferior a la derecha.
4. Estando ya en un dominio de frecuencia, se pasa a desechar una parte de la información de la frecuencia que se tiene. Para realizar este proceso, se hace uso de dos tablas llamadas tablas de cuantización. Se tiene una para el brillo y otra para el color. Aquí es donde entra en juego el nivel de compresión que se desee aplicar. Los valores de estas tablas se definen en función del nivel de compresión elegido y se emplean para dividir los valores correspondientes de la imagen y desechar el resto que queda de esta división.
5. La idea del proceso del paso anterior es que al quedar valores que han sido divididos, la probabilidad de tener varios valores iguales es alta (definida por el nivel de compresión aplicado). Esta igualdad permite juntar los valores iguales llevando a obtener un resultado final comprimido y más pequeño que el original.

Ocho años después de la publicación de JPEG, surgió en el año 2000 una versión mejorada en términos de eficiencia de compresión de éste, JPEG2000 (Skodras et al., 2001), que cuenta con su propia extensión de archivo, .jp2. Está basado en una transformada Wavelet (DWT) (Antoine, 2003) y ofrece unos niveles de compresión mayores a los de JPEG sin caer en las zonas ligeramente borrosas que en algunos tipos de imagen JPEG ofrece. La intención con la publicación de JPEG2000 era sustituir a su antecesor JPEG, pero con los años, JPEG se siguió manteniendo como el estándar, debido principalmente a los tiempos de ejecución de JPEG2000, que son mayores a los de JPEG, a pesar de ofrecer una mejor compresión que éste. Por su validez, uso y longevidad como métodos de compresión de imágenes digitales, ambas versiones de JPEG serán contrastadas con el método propuesto en el presente trabajo y empleadas para valorar los puntos fuertes y débiles del modelo final.

Otros de los formatos populares hoy en día, PNG, también data de esta época, concretamente del año 1996. PNG ofrece una compresión sin pérdida de excelente calidad especialmente para imágenes con grandes áreas de color plano o con pocos cambios de color. PNG se utiliza principalmente para gráficos web, iconos y logotipos, diagramas o ilustraciones, en lugar de fotografías de alta calidad, porque tienen un tamaño

mayor que los archivos JPEG y resultan más difíciles de transferir y almacenar, en comparación. Algo extra que ofrecen los archivos PNG a diferencia de los JPEG es la capacidad de manejar gráficos con fondos transparentes. También tenemos formatos como GIF, muy usado en la web para imágenes y animaciones, derivado del algoritmo de compresión LZW mencionado en el período primero, y publicado en el año 1987. Ofrece una compresión sin pérdida, pero para imágenes de solamente 256 colores, por lo que emplearlo con imágenes de más colores supondría un detrimento de su calidad que llegaría a ser claramente perceptible al ojo humano. Es por esto por lo que su uso es más infrecuente y está mas limitado, siendo sobretodo usado en algunos casos para almacenar animaciones.

Continuando con el análisis de la literatura realizado, y pasando al tercer periodo histórico identificado, en los últimos años han tomado presencia en la escena de investigación alrededor del tema de compresión de imágenes aquellos métodos basados en técnicas de aprendizaje profundo o *Deep Learning,* demostrando capacidades para llegar a igualar y en algunos casos o aspectos a superar el desempeño de los métodos más estandarizados o tradicionales, como por ejemplo los anteriormente mencionados JPEG o JPEG2000.

Desde el punto de vista del aprendizaje profundo, y hablando a alto nivel, podemos identificar tres fases principales en el proceso de compresión de imágenes: preprocesamiento, codificación y posprocesamiento. Desde la perspectiva de la Inteligencia Artificial se ha trabajado en encontrar métodos para aumentar el desempeño y eficiencia de cada una de estas fases, tanto individualmente como de forma conjunta. En el caso del aprendizaje profundo, las técnicas que se han utilizado se centran en el proceso de codificación, tratando de hacer que el *encoder* elimine redundancias y mantenga o incluso mejore la calidad visual junto a una reducción del *bitrate*, sin dejar de perder de vista el contexto de la compresión, es decir, las características concretas al archivo de imagen que se esté tratando. Si este objetivo se cumple exitosamente, tendremos un archivo cuya transmisión y manejo será más sencillo y rápido y, consecuentemente, el usuario de una determinada aplicación o sistema tendrá una experiencia de uso mejor.

Se pasa a continuación a presentar los diferentes métodos, especialmente desde la perspectiva de la red neuronal. El recorrido por los diferentes métodos se ha organizado por orden histórico, incluyendo principalmente modelos como MLP, *Random Neural Networks* y CNN. En la subsección final de este tercer periodo, se menciona el avance reciente de las técnicas de codificación de imágenes utilizando redes generativas antagónicas (GAN).

- **MLP**

Un perceptrón multicapa o MLP (Gardner & Dorling, 1998) es un tipo de red neuronal que se compone de una capa de entrada formada por varias neuronas, una o varias capas ocultas y una capa de salida. Desde un punto de vista teórico, un MLP construido con más de una capa oculta puede aproximar cualquier función computable continua con una precisión arbitraria. De esta manera, en escenarios como la reducción de dimensiones y la compresión de datos MLP puede ofrecer resultados aceptables. Uno de los primeros *papers* sobre el tema data del año 1988 y lleva el título "*A neural network approach to transform image coding*" (Chua & Lin, 1988). Se muestra que los tres pasos convencionales en la tarea de compresión de imágenes, a saber, la transformación unitaria de los datos de imagen del dominio espacial, la cuantificación de los datos del dominio de transformación y la codificación binaria de los datos cuantificados, se pueden unificar en un problema de optimización de un solo paso. Luego, el problema se resuelve mediante una red neuronal relativamente sencilla cuya entrada son los datos de la imagen del dominio espacial y cuya salida son los códigos binarios.

Hay otros estudios publicados que aportan diferentes variaciones de un MLP para conseguir buenos resultados comprimiendo imágenes, como por ejemplo los publicados por Daugman o Abbas et al (Abbas, n.d.; Daugman, 1988).

- **RNN**

La red neuronal aleatoria (RNN) (Gelenbe, 1989) es un modelo de red neuronal recurrente inspirado en el comportamiento basado en picos de las redes neuronales biológicas. Al contrario que la mayoría de los modelos de redes neuronales artificiales, las neuronas en la red neuronal aleatoria interactúan mediante el intercambio probabilístico de señales de picos. El modelo se describe mediante ecuaciones analíticas, tiene un algoritmo de aprendizaje supervisado de baja complejidad y es un aproximador universal para funciones continuas acotadas. Las redes neuronales aleatorias tienen aplicaciones en una variedad de áreas como reconocimiento de patrones y clasificación o procesamiento de imágenes.

Algunos investigadores consideraron la aplicación de redes neuronales aleatorias al tema de compresión de imágenes y presentaron algunos resultados significativos, siendo Gelenbe et al. quién aplicó por primera vez este tipo de redes para comprimir imágenes (Gelenbe & Sungur, 1994). La arquitectura propuesta adopta una red neuronal aleatoria de codificador/descodificador *feedforward* con una capa intermedia. Hay una primera capa encargada de tomar una imagen como entrada y la capa intermedia de producir *bits* comprimidos. Más adelante, Cramer et al. amplia aún más el trabajo anterior mediante el diseño de una red neuronal aleatoria para compresión y descompresión bloque a bloque (Cramer et al., 1996). Hai et al. mejoró aún más el rendimiento de la compresión al integrar la red neuronal aleatoria en el dominio wavelet de las

imágenes (Hai et al., 2001). Estos estudios presentaron resultados buenos pero mejorables, aun dando por demostrado el potencial que este tipo de redes pueden alcanzar para el tema de compresión de imágenes.

- **CNN y Autoencoders**

En el caso de compresión de imágenes, algo de vital importancia, como se ha mencionado previamente, es conservar el contenido de las imágenes con la mayor precisión posible al comprimirlas. Identificar y preservar las partes más representativas del contenido de las imágenes y priorizarlas frente a las menos relevantes es una tarea que puede llevarse a cabo con este tipo de redes, las convolucionales, precisamente por la naturaleza de éstas.

Las redes neuronales convolucionales o CNN son un tipo de red neuronal que organiza sus capas con el objetivo de identificar distintas características en las entradas para que de esta manera pueda identificar objetos y patrones recurrentes y, en última instancia, "ver". Para ello, la CNN contiene varias capas ocultas especializadas siguiendo una jerarquía. Las primeras capas pueden detectar formas a alto nivel como líneas, curvas y patrones recurrentes y a medida que se va profundizando en la estructura de la red van tomando lugar capas más complejas que reconocen formas más específicas como pueden serlo un rostro, siluetas y, en definitiva, más detalles.

Recientemente, los métodos basados en redes neuronales convolucionales superan a los algoritmos tradicionales por un amplio margen en tareas de visión artificial de alto nivel, como por ejemplo la clasificación de imágenes y la detección de objetos (LeCun & Bengio, 2015). Incluso para muchas tareas de visión por computador de bajo nivel y mayor nivel de complejidad, también logran un rendimiento muy impresionante, por ejemplo, en tareas de superresolución de imágenes (Yamanaka et al., 2017). CNN adopta la operación de convolución para identificar la correlación que exista entre píxeles vecinos, identificando de esta manera las zonas con cierta similitud estructural de la imagen. Es gracias a estas capas como se llega a identificar patrones, límites, objetos y demás elementos estructurales. Además, los campos receptivos locales y los pesos compartidos introducidos por las operaciones de convolución también disminuyen los hiperparámetros de las redes neuronales convolucionales, lo que reduce significativamente el riesgo de sobreajuste u *overfitting*. Inspirándose en estas redes y en sus virtudes de cara al tratamiento y procesamiento de imágenes, se han llevado a cabo muchos trabajos para explorar la viabilidad de la compresión de imágenes basada en CNN.

El pionero en introducir el uso de CNN con fines de compresión de imágenes fue Balle, en su *paper* publicado en 2016 "*End-to-end optimized image compression*" (Ballé et al., 2017). Propone un sistema de

compresión con dos módulos, análisis y síntesis, *encoder* y *decoder,* que demuestra igualar y para algunos casos superar el desempeño de JPEG2000 en términos de las métricas más comúnmente empleadas para medir la calidad de imágenes comprimidas, PSNR (*Peak Signal-to-Noise Ratio*) y SSIM (*Structural Similarity Index*), mencionadas con más detalle en el punto 4 del presente documento.

En el módulo de análisis se tienen tres bloques de convolución, donde encontramos una capa convolucional, un *subsampling* o submuestreo de los datos y una normalización. De forma análoga, el módulo de síntesis replica la idea del de análisis, pero invertida. Esta idea de una arquitectura basada en dos módulos relacionados entre sí, el primero de los cuales genera una versión codificada y reducida de la imagen de entrada y el segundo reconstruye la representación entregada por el primero, es lo que lleva a desarrollar métodos de compresión de imágenes basados en *autoencoders*.

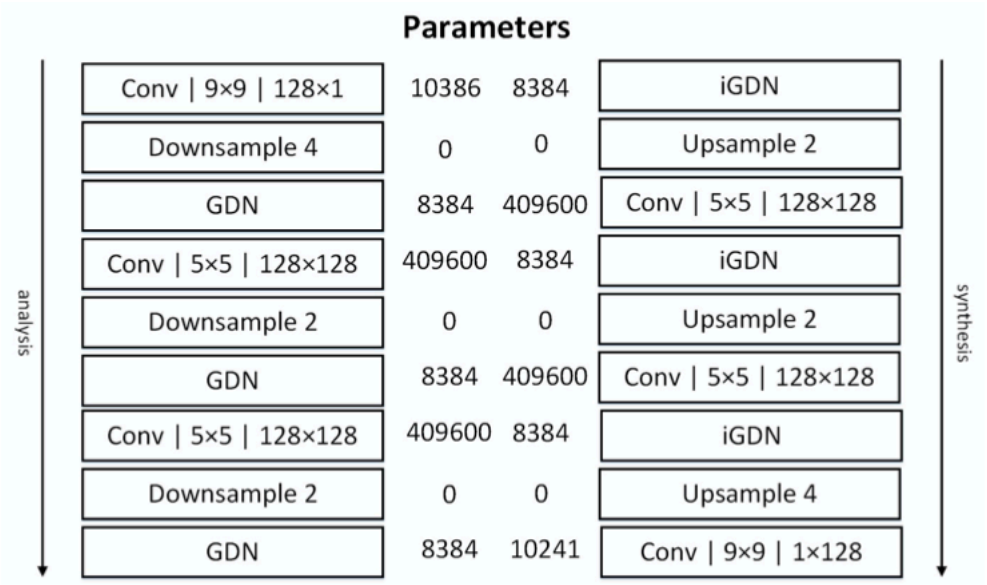


Figura 2 - Arquitectura de CNN propuesta por Balle et al.

Otro estudio relevante que usa una arquitectura basada en redes neuronales convolucionales, y que se menciona con cierta frecuencia en bastantes estudios publicados, es el presentado por Zhou et al., que mejora el desempeño del método de Balle et al. utilizando una estructura de fusión de características piramidales en el *encoder* y un filtro de posprocesamiento basado en CNN en el *decoder* (L. Zhou et al., 2018).

Pasando a hablar de los *autoencoders* anteriormente mencionados, éstos son un tipo de red neuronal, entrenadas de manera no supervisada, que tienen como objetivo partir de unos datos de entrada, generar su representación codificada, y después generar una representación de éstos lo más exacta posible partiendo

en base a dicha versión codificada. Por tanto, y visto desde un punto de vista más simple, la salida del *autoencoder* es su predicción para la entrada dada. Se componen de dos módulos: *encoder* y *decoder*. El *encoder* se encarga de obtener la representación codificada de la entrada y el *decoder* de llegar a partir de ésta a su versión reconstruida. Los *autoencoders* resultan especialmente útiles para resolver problemas de eliminación de ruido en imágenes y en detección de anomalías, dada su facilidad para reconocer e identificar las partes más destacables y relevantes de las imágenes, proceso inherente al proceso de entrenamiento de la red. Esta misma idea y virtud, discriminar el contenido de una imagen en función de la utilidad o relevancia que las diferentes partes del mismo tienen respecto al propósito del problema que se trata de resolver, es precisamente uno de los grandes beneficios que tienen los *autoencoders* para enfrentar el problema de compresión de imágenes digitales. Esta razón, unida a los numerosos estudios publicados usando este tipo de redes convolucionales para resolver este problema y a una motivación e iniciativa personal, fueron los motivos de escoger los *autoencoders*, en concreto los *autoencoders* convolucionales (CAE), como base del método de compresión de archivos de imagen propuesto.

Yendo nuevamente a la literatura publicada, hay varios estudios basados en *autoencoders* que sirvieron de inspiración para el desarrollo del presente proyecto. Estos estudios resultaron bastante interesantes, tanto por el lado técnico y de los resultados que ofrecían como por las explicaciones e ideas que planteaban. Se ha hecho la selección de unos pocos para centrarse en ellos, estudiarlos con profundidad y tomar como referencia.

Por ejemplo, resulta interesante ver como Theis et al. abordan el problema de compresión de imágenes usando una arquitectura sub-píxel (también propuesta en otros trabajos como el publicado por Cai et al. (Cai et al., 2019)) basada en *autoencoders* convolucionales y obtienen unos resultados que compiten con los de JPEG2000 (Theis et al., 2017). En este estudio se consideran imágenes de gran tamaño en lugar de imágenes pequeñas, como es común ver en multitud de trabajos publicados con anterioridad sobre el tema, y es por eso precisamente que resultó útil y enriquecedor revisar y estudiar a conciencia esta publicación, ya que era ese precisamente el enfoque que se deseaba dar al presente proyecto. Theis et al. también demuestra que optimizando un *autoencoder*, junto con un sistema de entrenamiento incremental, es decir, añadiendo datos sobre la marcha, dinámicamente, se pueden tener resultados buenos y competitivos tanto con métricas objetivas como subjetivas, y sin sufrir un coste computacional excesivo, desventaja que presentaban también otros estudios relacionados publicados anteriormente.

Otro trabajo de referencia es el publicado en el año 2018 y lleva el título "*Deep Convolutional AutoEncoder-based Lossy Image Compression*" (Cheng et al., 2018), un año después de la publicación del de Theis et al.

Este estudio presenta una arquitectura simétrica basada en *autoencoders* convolucionales que supera ligeramente el desempeño de JPEG2000, manteniendo su misma complejidad.

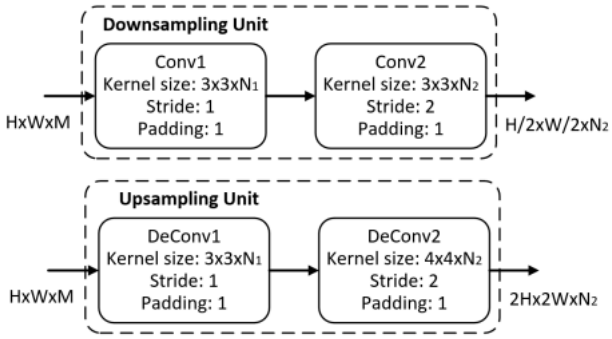


Figura 3 - Bloques convolucionales usados por Cheng et al.

La idea de incluir bloques convolucionales que encapsulen capas convolucionales junto con transformaciones añadidas a éstas se obtuvo de este estudio y su aplicación resultó exitosa al llevarla al modelo propuesto en el presente documento. Estos bloques o unidades forman parte de la arquitectura de la red y el detalle que Cheng et al. muestra en su publicación se puede ver en la Figura 2.

También, Cheng et al. propone añadir un bloque de cuantización de la imagen entre el *encoder* y el *decoder*, para estrechar y comprimir aún más la imagen antes de reconstruirla, junto con un módulo de PCA (*Principal Component Analysis*), técnica estadística basada en una transformación lineal frecuentemente usada para reducir las dimensiones de una imagen o detectar rotaciones en ella.

Estos dos estudios previos son bastante conocidos en el contexto del tema de compresión de imágenes y son frecuentemente citados. Como tercer estudio de referencia, se ha seleccionado el presentado por Akyazi et al., titulado "*Learning-Based Image Compression using Convolutional Autoencoder and Wavelet Decomposition*" y publicado en 2019 (Akyazi & Ebrahimi, 2019). Desarrollan y proponen un método nuevamente basado en *autoencoders*, en este caso más sencillo y convencional que los dos anteriores, que supera ligeramente a JPEG y se compara con otros trabajos similares. Aplica una transformada Wavelet al principio y al final del *autoencoder*, como un bloque de preprocesamiento y posprocesamiento respectivamente. De esta manera, según se explica, las características de frecuencia de la imagen se ajustan mejor y en general el desempeño del modelo aumenta. Hablando a alto nivel, podríamos comparar el estudio publicado por Akyazi et al., en términos de complejidad y en términos de resultados, al método de compresión de imágenes que se propone en este documento.

- **GAN**

Por último, y a modo de curiosidad y con vistas a la evolución a futuro que tendrá este tema de compresión de imágenes, el uso de GANs, redes adversarias generativas o redes generativas antagónicas, (*Generative Adversarial Neural Networks)* para comprimir imágenes, concretamente en la tarea de obtener su representación reducida o codificada ha venido resultando en una serie de publicaciones que demuestran que este tipo de redes pueden mejorar mucho el desempeño de los sistemas de compresión. En la tarea de compresión de imágenes, algunos trabajos publicados se centraron en la calidad perceptiva y el nivel de similaridad estructural de las imágenes decodificadas y utilizaron GANs para mejorar el rendimiento de la codificación. A modo de referencia para explicar el rol que pueden jugar las GANs en compresión de imágenes, se menciona el *paper* publicado en el año 2017, "*Real-time adaptive image compression*" (Rippel & Bourdev, 2017), siendo este uno de los trabajos más representativos. Se trata de un método de compresión de imágenes basado en GANs optimizadas, que no solo logra una mejora destacable en la ratio de compresión resultante, sino que también puede ejecutarse en tiempo real aprovechando la potencia de las GPUs modernas, con varios núcleos de ejecución paralelos. La imagen de entrada se comprime en un espacio de características muy compacto mediante arquitecturas diferentes, y la red generativa se utiliza para reconstruir la imagen en ese punto, pasando de una versión codificada a una decodificada.

Esta forma de incorporar las GANs a un sistema de compresión ofrece unos resultados muy destacables, produciendo archivos comprimidos 2.5 veces más pequeños que los producidos por JPEG y JPEG2000, 2 veces más pequeños que el formato WebP (Si & Shen, 2016) y 1.7 veces más pequeños que BPG en imágenes de dominio general. Sin embargo, el método muestra deficiencias al medir la calidad de la compresión comparando la imagen comprimida y la original con la métrica SSIM, no siendo así empleando la métrica referente a la similaridad estructural SSIM.

El futuro de la compresión de imágenes empleando aprendizaje profundo es incierto pero prometedor. Desde hace unas décadas, diversas arquitecturas e ideas se han ido aplicando y probando poco a poco para comprimir imágenes, o en su defecto para estar involucradas de alguna manera en el proceso de compresión. En los últimos años, el tráfico en internet ha estado mayormente dominado por aplicaciones relacionadas con archivos de imagen y vídeo, especialmente servicios de *streaming* y aplicaciones de distribución de video bajo demanda (VOD). Actualmente, más de tres cuartas partes del tráfico total de internet corresponden al campo audiovisual, lo cual hace tener mucha importancia a la tarea de optimizar al máximo estos servicios para ofrecer una experiencia de uso mejor a los usuarios y, en la medida de lo posible, ahorrar costos y recursos. A esto se le añade también el uso de archivos cada vez con más calidad y peso, en el caso de los vídeos, las resoluciones 4k y 8k (Electronics & Paper, n.d.), por ejemplo. Los resultados tan buenos

alcanzados por algunos de estos métodos han llevado a compañías de la talla de Netflix o Amazon a aplicar sistemas basados en *Deep Learning* para optimizar sus procesos de servicio de vídeo bajo demanda, y poco a poco, otras de menor tamaño lo van incluyendo progresivamente, viendo sus ventajas y potencial presente y futuro. Por ejemplo, Netflix, en el año 2020, anunció la adopción de un nuevo formato de archivos de imagen de creación propia, AVIF1 (*AVIF for Next-Generation Image Coding | by Netflix Technology Blog | Netflix TechBlog*, n.d.). También desarrollaron junto a investigadores de la Universidad de California principalmente, su propia métrica de calidad para vídeos, VMAF, un algoritmo cuyo desarrollo e ideas explican en su blog (*VMAF: The Journey Continues. by Zhi Li, Christos Bampis, Julie… | by Netflix Technology Blog | Netflix TechBlog*, n.d.) y cuyo código está público (*GitHub - Netflix/Vmaf: Perceptual Video Quality Assessment Based on Multi-Method Fusion.*, n.d.), así como librerías para usarlo escritas en C y Python.

Para finalizar la revisión de la literatura, y terminar este tercer periodo, hay otras técnicas novedosas y originales basadas en aprendizaje de máquinas que han ido surgiendo para colaborar y aportar beneficios para el problema de compresión de imágenes, entre las que se ha considerado destacar, por su interés y resultados, el *multitask learning*.

Se trata de un enfoque que busca aprender múltiples tareas de manera simultánea, al momento de entrenar el modelo, optimizando varias funciones de pérdida a la vez. En lugar de entrenar modelos independientes para cada tarea, permitimos que un solo modelo aprenda a completar todas las tareas paralelamente. Esta técnica se emplea en multitud de tareas como por ejemplo procesamiento de lenguaje natural, sistemas de recomendación de contenido o compresión de datos. Para la última, que es la más relevante al contexto del presente documento, tenemos algún *paper* que merece la pena mencionar. Por ejemplo, Du et al. presenta en un *paper* de muy reciente publicación, un método basado en GANs y *multitask learning* para aplicar compresión a imágenes aportadas por dispositivos IoT, ofreciendo unos resultados que se aproximan bastante a los de JPEG2000 (Du et al., 2022).

De esta manera, concluye la revisión de la literatura referente al tema de compresión de imágenes, haciendo un especial énfasis en el papel del aprendizaje profundo en este problema. Se ha recorrido de forma selectiva toda la serie de estudios e investigaciones publicadas sobre el tema, empezando por los estándares más extendidos actualmente pasando por soluciones más ingeniosas y complejas, recalcando algunos conceptos teóricos importantes, y señalando las virtudes de cada una de las técnicas y formas de enfrentar este problema revisadas.

4. Método de compresión de archivos de imagen usando técnicas de *deep learning*.

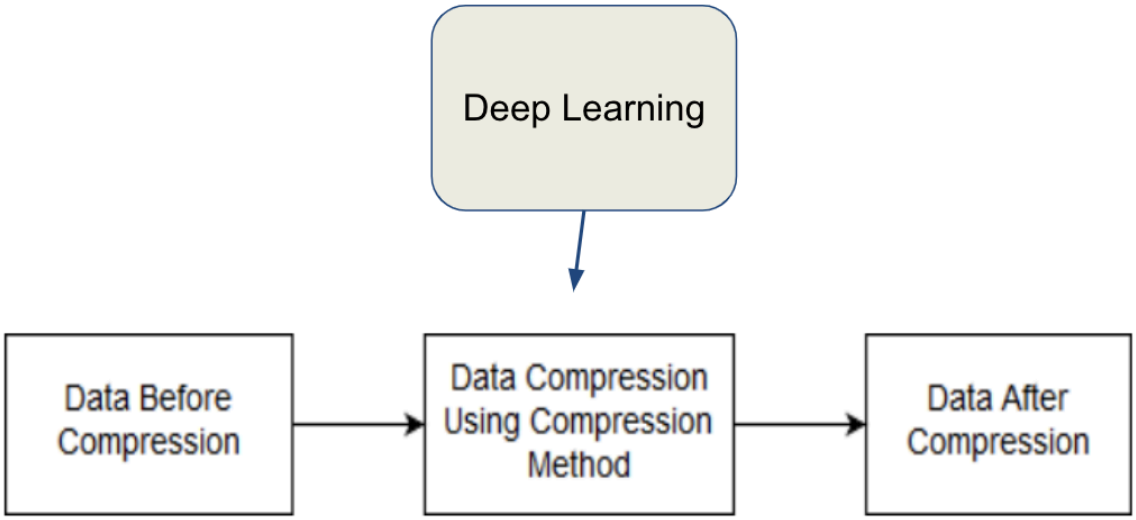


Figura 4 - Compresión de datos y Deep Learning

4.1 Conjunto de datos.

"Everyone wants to do the model work, not the data work" es la frase de cabecera que se presenta en el *paper* "*Data Cascades in High-Stakes AI*" publicado por investigadores y científicos de Google en el año 2021 (Sambasivan et al., 2021), donde se presentan una serie de casos para explicar el concepto de *data cascades*. Se trata de eventos que pueden darse en un proyecto de Inteligencia Artificial, especialmente en las primeras etapas, que perjudican de forma progresiva a medida que se avanza en el desarrollo del mismo a la obtención de buenos resultados y en la correcta toma de decisiones en el proyecto. En este *paper*, argumentan que buena parte de las veces que se produce este problema, es debido al empleo de prácticas de desarrollo de software muy tradicionales y obsoletas y a tratar de afrontar problemas de Inteligencia Artificial como si fueran otro tipo de problemas cualesquiera relacionados con software, especialmente no priorizando la calidad, obtención y procesamiento de los datos. Hacen especial énfasis en el impacto que tiene en este tipo de proyectos la calidad de los datos, y en particular en aquellos casos donde el proyecto tiene algún tipo de impacto social, como los relacionados con temas de educación o salud.

La elección de una buena fuente de datos es primordial en el desarrollo y diseño de un sistema basado en Inteligencia Artificial, siendo éstos la materia prima clave a partir de la cual va a surgir eventualmente esa

inteligencia en el sistema y la parte de todo el proceso de desarrollo más trascendente para conseguir en última instancia unos resultados buenos y útiles. No alcanzar una calidad de datos como la esperada o deseable para el caso de aplicación concreto, va a suponer añadir múltiples dificultades en todo el proceso posterior de desarrollo del modelo, su entrenamiento y su evaluación y puesta a punto en un escenario de uso real. Es decir, podemos afirmar que descuidar la calidad de los datos empleados en un proyecto basado en IA implica de forma directa unos resultados y desempeño muy por debajo de lo esperado.

Para encontrar un conjunto de datos acorde a la naturaleza del problema concreto a tratar, en la compresión de imágenes digitales se optó, como es común en la literatura respectiva, por buscar conjuntos de datos con imágenes de alta calidad y pertenecientes a un dominio general.

En el contexto del caso particular del proyecto expuesto en el presente documento, donde tratamos de construir una imagen a partir de su versión codificada, resulta de vital importancia tener un conjunto de datos de imágenes que presenten características en términos de luminosidad, nitidez, color, variedad de características y definición lo mejores posibles, para de esta manera asegurar que el modelo aprende correctamente a cómo codificar las imágenes con el mayor nivel de detalle posible y sin despreciar información que pudiera ser relevante.

Si para entrenar el modelo hacemos uso de imágenes poco nítidas, borrosas, con mucho ruido añadido o muy pequeñas y pixeladas, llegaremos a experimentar problemas de rendimiento a la hora de usar el modelo con una imagen de un mayor tamaño o calidad, así como queda demostrado en las primeras pruebas que fueron realizadas en las primeras etapas del proceso de experimentación llevado a cabo, descrito de forma detallada en el punto 4.2.6.

Durante dicho proceso de experimentación se probaron varios conjuntos de datos para entrenar el modelo. Se observó durante los experimentos que conjuntos de datos que contenían imágenes de tamaño mediano-pequeño y pertenecientes a un dominio general daban resultados no demasiado buenos a la hora de probar el modelo con imágenes de un tamaño variante. Cabe recalcar, que con tamaño mediano-pequeño se refiere a imágenes de un tamaño aproximado o inferior a 480x360 píxeles. Por otro lado, con dominio del conjunto de datos se refiere a la naturaleza del contenido de las imágenes. Por ejemplo, dominio general son imágenes que pueden contener desde un paisaje hasta un retrato pasando por una imagen obtenida por un microscopio o una imagen satelital.

Fueron llevadas a cabo pruebas, principalmente y entre otros, con tres conjuntos de datos diferentes que ofrecieron resultados cualitativamente buenos, YouTube-8M (Abu-El-Haija et al., 2016), Flickr1024 (Wang et al., 2019) y *places dataset* (B. Zhou et al., 2018). De entre los tres y a pesar de que los resultados fueron aceptables para todos, para el método propuesto, comparativamente, places *dataset* ofreció los mejores resultados, seguido de YouTube-8M.

El conjunto de datos *places dataset* fue creado por científicos e investigadores del MIT (*Massachusetts Institute of Technology*) y está diseñado siguiendo los principios de la cognición visual humana, con el objetivo principal de ofrecer un conjunto de conocimiento visual destinado a usarse en sistemas basados en IA que persigan tareas de comprensión visual como por ejemplo reconocimiento de objetos, predicción de acciones, identificación de patrones etc. Reúne imágenes con diferentes categorías y etiquetas, pero todas ellas aludiendo a lugares de toda índole. Por ejemplo, compila imágenes de calles, habitaciones, paisajes, situaciones cotidianas, detalle de objetos etc. Se empleó un subconjunto de 5300 imágenes con un tamaño por cada una de 768x512 píxeles.

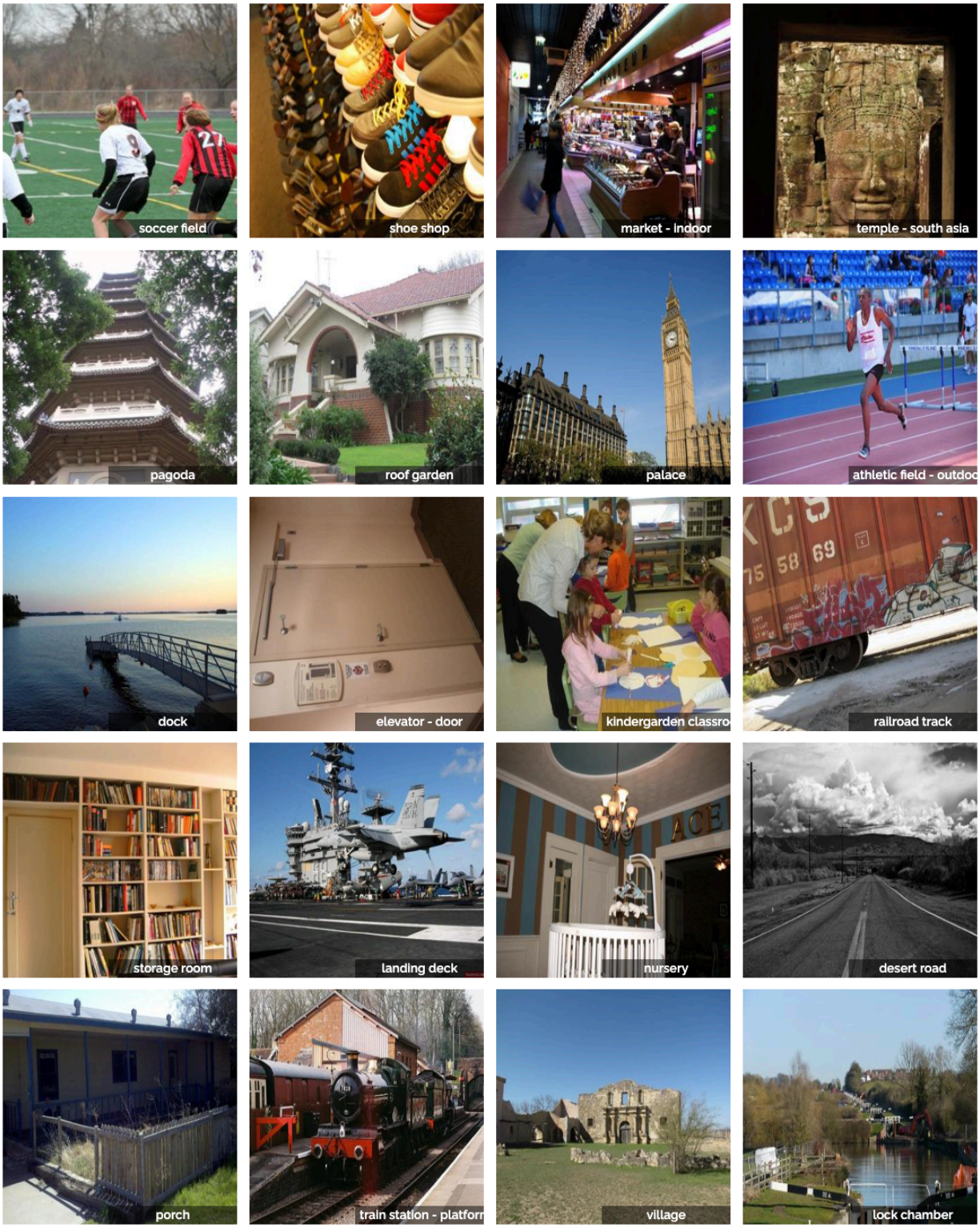


Figura 5 - places dataset, muestra de imágenes

Al comparar los resultados obtenidos usando *places dataset* con los obtenidos usando YouTube-8M, a pesar de ser subjetivamente buenos ambos, resulta en un mejor desempeño del modelo emplear el primero. Al respecto de YouTube-8M, éste es un conjunto de datos creado por investigadores y científicos de Google, concretamente del grupo *Google Research Video Understanding*, que reúne imágenes extraídas de vídeos publicados en *YouTube*, con un tamaño de 1080x720 píxeles y un alto nivel de definición y detalle. También incluye el etiquetado de las imágenes en un total de 1000 categorías diferentes con sus entidades correspondientes, pero para el propósito del presente proyecto no ha sido necesario hacer uso de esta característica adicional. La descripción detallada de cómo construyeron el *dataset*, junto a algunos experimentos y casos de uso reales, puede ser consultada en el *paper* publicado por los autores, "*YouTube-8M: A Large-Scale Video Classification Benchmark*" (Abu-El-Haija et al., 2016).

4.2 Diseño y desarrollo del método.

4.2.1 Arquitectura

Partiendo de las ideas expuestas con anterioridad y de las obtenidas fruto de toda la revisión de la literatura realizada en torno al tema de compresión de imágenes, se llegó a proponer la siguiente arquitectura para el modelo del sistema de compresión.

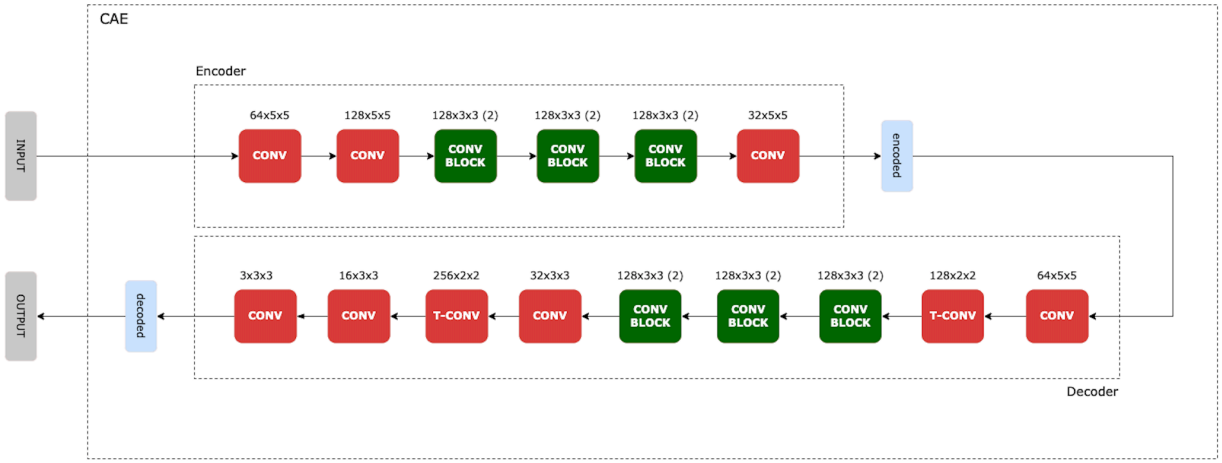


Figura 6 - Arquitectura del Autoencoder Convolucional (CAE) propuesto (elaboración propia)

Tenemos un *autoencoder* convolucional (CAE) con los dos componentes base en estos modelos. Por un lado, un *encoder* que recibe los datos de entrada y por otro un *decoder* que recibe la salida comprimida del *encoder* y devuelve la predicción del *autoencoder*. Este tipo de *autoencoders*, los convolucionales, son las herramientas más avanzadas para el aprendizaje no supervisado de filtros convolucionales. Los filtros

convolucionales son especialmente útiles en este caso de aplicación, como se ha mencionado previamente, porque permiten extraer características representativas del contenido de las imágenes. La principal diferencia que existe entre una red neuronal convolucional (CNN) y un CAE es que la primera está pensada de principio a fin para aprender filtros y combinar funciones con el objetivo principal de clasificar su entrada. Sin embargo, los *autoencoders* convolucionales se centran principalmente en aprender filtros capaces de extraer características que pueden usarse para reconstruir la entrada, teniendo de esta manera una función particular bien específica y concreta que, aplicada al problema de comprimir imágenes, permite llegar a una salida con un nivel de similitud lo más cercano posible a la imagen original.

Analizando en específico la arquitectura propuesta, la mayoría de convoluciones se aplican tras hacer un *downsampling* de la imagen, para de esta manera acelerar el esfuerzo computacional al aplicarlas y reducir tiempos. El proceso de *downsampling* o submuestreo consiste en reducir la resolución espacial de la imagen, básicamente el número de pixeles, sin modificar sus dimensiones (2D).

Cada bloque convolucional (los señalados en verde en la Figura 5) se compone de dos capas convolucionales que siguen a un *padding* de ceros previo. Este *padding* se puede incluir en la definición de la capa convolucional generalmente empleando en la mayoría de *frameworks* el argumento "padding" con el valor "same", pero se decidió separar ambos procesos por un tema meramente estético y visual de la definición de la arquitectura y para tener un esquema más claro de todo el modelo que permitiera hacer modificaciones durante el proceso de manera más sencilla y precisa. La función de activación usada es LeakyRelu, inspirado en el trabajo propuesto por Theis et al. y en contra de escoger Relu, como suele estilarse en la mayoría de los trabajos relacionados, y por otro lado un tamaño de *kernel* de 3x3. En casos prácticos, se puede demostrar que usar LeakyRelu puede llevar a reducir el error obtenido (Xu et al., 2015).

De forma análoga, los ítems convolucionales (señalados en rojo en la Figura 5) se componen de un *padding* previo a la convolución. La diferencia principal con los bloques convolucionales es que se define un tamaño de *kernel* superior, de 5x5 y un valor de desplazamiento o *strides* de 2 en vez de 1 en las capas convolucionales. Aumentar a 2 el valor del desplazamiento o *strides* permite reducir exactamente a la mitad el tamaño de la salida de la convolución. Esto ayuda a reducir las dimensiones de la imagen de entrada en las primeras capas, previo a pasar a los bloques convolucionales. Por otro lado, sirve para reemplazar el hipotético uso de *MaxPooling* que se hace en estos casos, como se verifica en algunos trabajos presentes en la literatura. Algunos optan por este método en vez de hacer un *MaxPooling* y en el caso particular del presente proyecto, tras haber hecho los experimentos correspondientes, ha dado efectivamente unos resultados mejores. Springenber et al. demuestra en su *paper* "*Striving for simplicity: the all convolutional*

net" (Springenberg et al., 2015) que reemplazar capas de subsampleo como las *MaxPooling* por capas convolucionales con un desplazamiento incrementado (empleando un mayor desplazamiento o *strides*) lleva en algunos casos a incrementar la precisión del modelo y por consiguiente a obtener unos mejores resultados.

En cuanto al *decoder*, observando el diagrama del modelo puede apreciarse que presenta una arquitectura análoga con la del *encoder*, pero añadiendo más capas antes y después del trio de bloques convolucionales. En la parte previa a los bloques convolucionales tenemos la primera deconvolución aplicada a la entrada, y en la parte posterior a los bloques convolucionales la segunda, seguida de dos capas convolucionales más para reducir los canales de representación hasta llegar a 3.

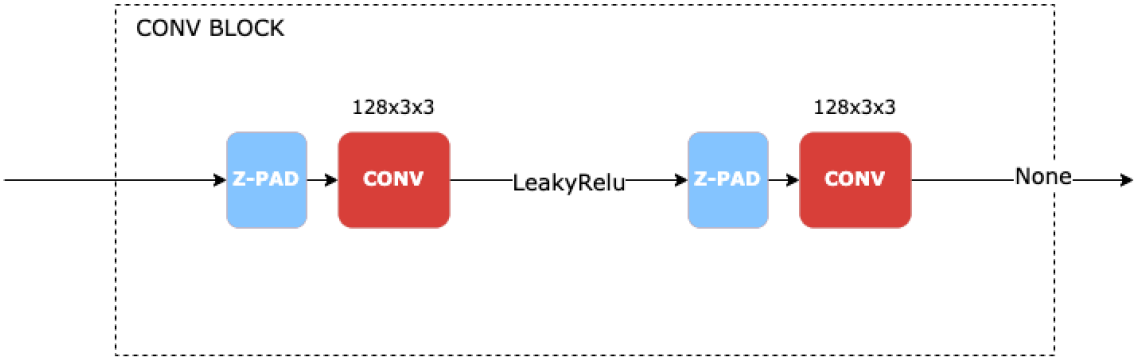


Figura 7 - Detalle de los bloques convolucionales usados en el modelo propuesto (elaboración propia)

Para entrenar el modelo, se empleó la función de error cuadrático medio (MSE) como función de pérdida y el optimizador Adam, empleados ambos en una parte de los *papers* leídos durante la revisión de la literatura realizada.

Para que el proceso de entrenamiento no fuese excesivamente largo y tedioso, se llevó a cabo la idea de dividir cada imagen del conjunto de datos, a medida que iba iterándose de época en época, en una serie de fragmentos de menor tamaño. De esta manera, el modelo irá aprendiendo el contenido de la imagen de una manera mucho más granular y precisa. La explicación de esta idea y de cómo se llevó a cabo está detallada en el punto 4.2.4.

4.2.2 Modelos específicos

Paralelamente a los experimentos realizados con conjuntos de datos con imágenes de un dominio general, es decir, con imágenes de todo tipo en cuanto a su contenido, se fueron también realizando experimentos

con conjuntos de datos de un dominio específico buscando obtener modelos que ofreciesen un mejor desempeño comprimiendo imágenes pertenecientes a un dominio concreto. La idea es plantear que, con modelos entrenados con imágenes de un dominio específico, el desempeño del modelo comprimiendo dichas imágenes sea superior al del modelo entrenado con imágenes de dominio general. De esta manera, llegamos a asociar en ciertos casos el desempeño de un *autoencoder* aplicado al problema de compresión de imágenes con el dominio específico de las imágenes con las que ha sido entrenado y, por otro lado, con las que dicho modelo es usado.

Por ejemplo, una idea que se contempló fue aplicar este tema de la compresión de imágenes al dominio específico de las imágenes médicas. Las imágenes médicas son hoy en día parte fundamental de la exploración diagnóstica de un paciente en muchas situaciones clínicas. Las diferentes modalidades de adquisición de imágenes han permitido mejorar la calidad del diagnóstico y, por tanto, su uso se ha extendido prácticamente a todas las ramas de la actividad médica. Con la integración de las técnicas digitales en la adquisición y el desarrollo de herramientas de análisis y procesamiento de imágenes, el flujo de trabajo del médico ha cambiado también y se ha reestructurado en parte sobre la base de la información visual que se tenga, convirtiéndose ésta en fuente fundamental del conocimiento.

En el campo médico, tan extenso como es, podemos encontrar imágenes de múltiples partes del organismo, desde ojos y oídos hasta encefalogramas o imágenes por rayos X mostrando vasos sanguíneos, órganos y huesos. Es este último el ejemplo de otro de los modelos obtenidos que fue entrenado con una colección de imágenes médicas provenientes de radiografías de tórax, también llamadas placas tórax, de pacientes de diverso género y edad. El conjunto de datos empleado para este caso concreto se compone de 5863 imágenes en total (*Chest X-Ray Images (Pneumonia) | Kaggle*, n.d.). Estas radiografías muestran los pulmones, caja torácica, corazón, vías respiratorias y los huesos de la columna y el tórax del paciente. Las placas tórax presentan la particularidad de estar en blanco y negro y de manejar un espectro amplio de colores dentro de este rango, es decir, presentan una amplia gama de tonos grises. Un ejemplo de este tipo de imágenes se puede ver en la figura siguiente.

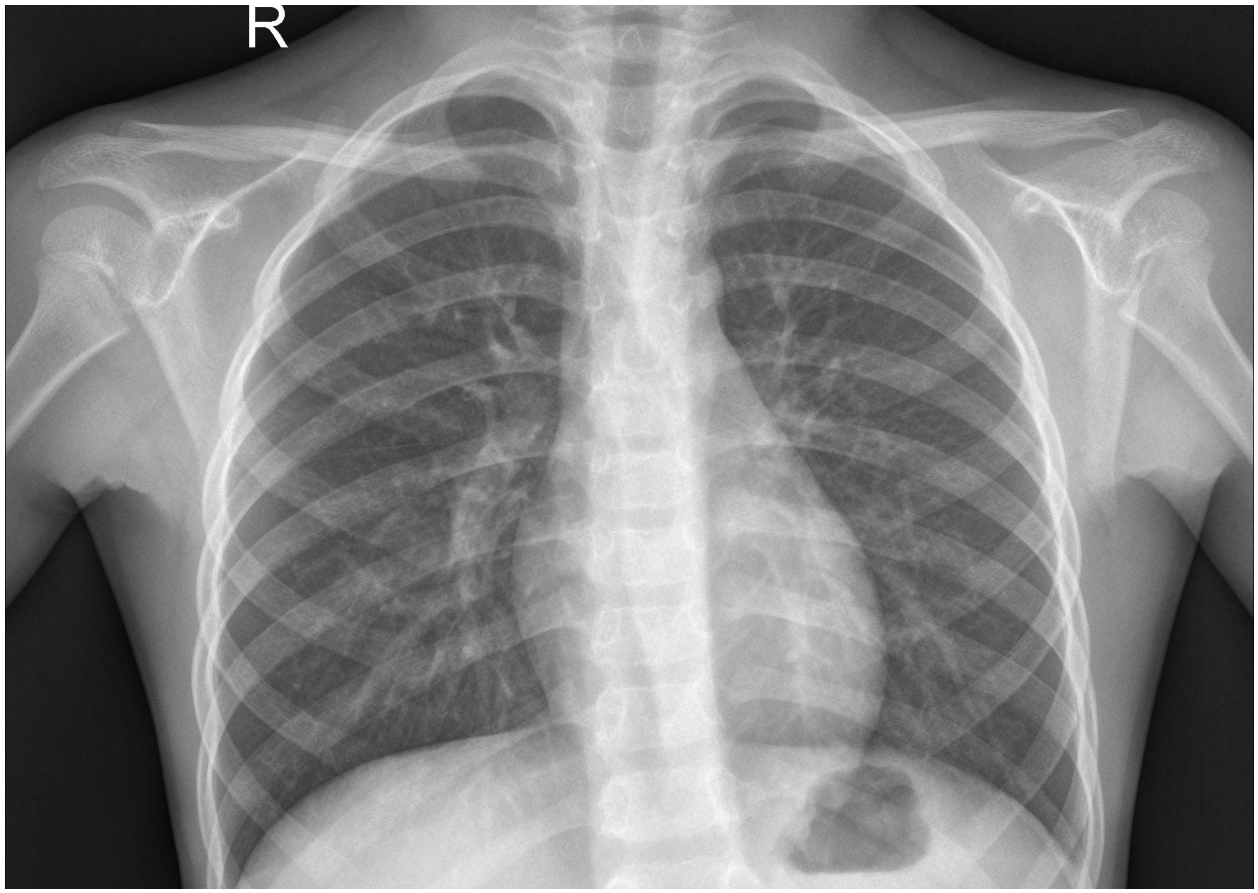


Figura 8 - Imagen perteneciente al dataset Chest X-Ray Images

4.2.3 Métricas

Para evaluar y medir el desempeño del método propuesto se han escogido una serie de métricas descritas a continuación.

- SSIM

SSIM (*Structural Similarity Index*) representa una métrica que hace referencia al grado de similitud estructural que hay entre dos imágenes. Es una métrica que cuantifica la degradación de la calidad de la imagen causada por el procesamiento, como por ejemplo su compresión, o por pérdidas en procesos de transmisión de datos. Es una métrica de referencia completa que requiere de dos imágenes de la misma captura de imagen: una imagen de referencia y una imagen procesada. La imagen procesada normalmente es la versión comprimida de la primera. SSIM es más conocido en la industria del video, pero tiene fuertes aplicaciones para la fotografía fija y sobretodo el análisis de imágenes. La diferencia con otras técnicas como MSE o PSNR es que estos enfoques estiman errores absolutos. Con información estructural nos referimos a la idea de que los píxeles de una imagen tienen fuertes interdependencias, especialmente cuando están espacialmente cerca, es decir, tienen relaciones entre sí que contribuyen de manera directa a la información completa que aporta la imagen. Estas dependencias implican información importante sobre la estructura de los objetos en la escena visual.

- MSE

MSE (*Mean Squared Error*) o error cuadrático medio es una métrica de similitud que se utiliza en compresión de imágenes para medir la calidad de dicha compresión. MSE representa el error cuadrático acumulativo entre la imagen comprimida y la de referencia u original. Calcula la diferencia píxel a píxel por cada canal de color entre ambas imágenes. Cuanto menor sea el valor de MSE, menor será el error y por consiguiente más calidad tendrá la imagen comprimida y más satisfactorio será el proceso de compresión.

La fórmula del error cuadrático medio es la siguiente, donde yi es el i valor observado en la imagen original, $y'i$ el i valor observado en la imagen resultado y n el número de observaciones realizadas.

$$MSE = (1/n) * \Sigma(yi - y'i)2$$

- PSNR

PSNR (*Peak signal-to-noise ratio*) o proporción máxima de señal a ruido es un concepto recurrente en ingeniería para obtener la relación entre la potencia máxima posible de una señal (su valor máximo) y la potencia del ruido que afecta a la fidelidad de su representación. Esta relación se utiliza para el caso de imágenes como una medida de calidad entre la imagen original y la imagen procesada. Cuanto mayor sea el PSNR, mejor será la calidad de la imagen procesada o reconstruida. Debido a que muchas señales tienen un rango dinámico muy amplio, el PSNR generalmente se expresa como una cantidad logarítmica utilizando la escala de decibelios.

La fórmula del PSNR es la siguiente, donde f es la matriz de la imagen original y MSE el error cuadrático medio.

$$PSNR = 20 * \log\left(\frac{MAXf}{\sqrt{MSE}}\right)$$

- CR

El CR (*Compression Ratio*) o relación de compresión se define como la ratio entre el tamaño de la imagen original ($SizeO$) y el de la imagen comprimida ($SizeC$). Determina cuánto se ha comprimido la imagen original en términos puramente de tamaño con respecto a su versión comprimida. La fórmula consiste simplemente en dividir el tamaño de la imagen original entre el tamaño de la imagen comprimida.

$$CR = \frac{SizeO}{SizeC}$$

4.2.4 Implementación

A la hora de pasar a implementar el método propuesto y descrito anteriormente, se enfrentaron una serie de retos técnicos. Un problema importante mencionado brevemente con anterioridad es el tema de la segmentación de las imágenes en el proceso de entrenamiento. Entrenar un *autoencoder* con fines de compresión de imágenes requiere un conjunto de datos con imágenes de alta calidad en cuanto a su definición y sus dimensiones. Con este tipo de imágenes se ralentizaba bastante el proceso de entrenamiento al tener que procesar en cada época una imagen de 768x512 píxeles de manera directa. Viendo esto como un impedimento a poder experimentar e iterar los modelos obtenidos de una manera mínimamente ágil, fue un imperativo encontrar una solución que evitara o mitigara esto y que pudiera adaptarse e implementarse con las menores complicaciones posibles al proceso de entrenamiento ya definido. Es dentro de este contexto donde surge el concepto propiamente llamado de tamaño de segmento. La idea fue evitar procesar una imagen grande directamente pasando a dividir cada imagen en cada época en una serie de subimágenes del mismo tamaño. Este tamaño es determinado por el tamaño de segmento que se defina. De esta manera es posible procesar cada una de estas subimágenes de manera individual en lugar de hacerlo directamente con una de gran tamaño.

Dos caminos pueden seguirse para desarrollar esta idea. El primero de ellos es tomar una serie de subimágenes extraídas de la imagen original cuyas coordenadas dentro de ésta fuesen aleatorias. Resultaría en una serie de subimágenes independientes entre sí y que, dada la aleatoriedad en su elección, podrían compartir porciones de píxeles. El problema con este método para el problema de compresión de imágenes es que se dejaría de tener en cuenta toda la información completa de la imagen, ya que esa aleatoriedad no garantiza preservar la información completa de la imagen original. El otro camino para tomar, y el que se escogió, es dividir la imagen en segmentos de forma secuencial. Dicho de otra manera, plantear una cuadrícula en la imagen basada en el tamaño de segmento escogido e ir entrenando secuencialmente el modelo con cada subimágen obtenida. De esta segunda manera, se garantiza por un lado que toda la información de la imagen original es tenida en cuenta y también que, por cada una de las imágenes del conjunto de datos, se está teniendo exactamente el mismo número de subimágenes por época.

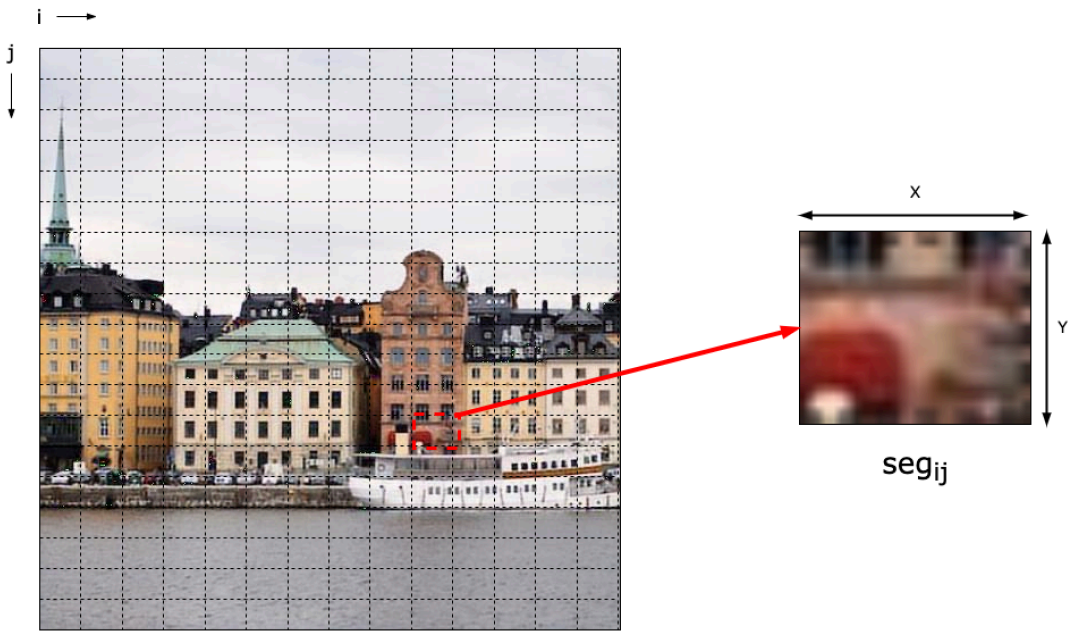


Figura 9 - Conceptos de tamaño de segmento y subimágen (elaboración propia)

Se realizaron pruebas con un tamaño de segmento de 256 y de 128, resultando en subimágenes de 256x256 y 128x128 píxeles respectivamente. A la hora de experimentar con diferentes arquitecturas del modelo, y probando ambos tamaños de segmento, se concluyó que el desempeño tanto por el lado computacional como por el lado del propio modelo era mejor con un tamaño de segmento de 128 píxeles, estableciendo de esta manera este valor para el entrenamiento de nuestro modelo.

Así, y teniendo en cuenta, como se ha señalado en el punto 4.1, que se empleó un conjunto de 5300 imágenes con un tamaño por cada una de 768x512 píxeles, un tamaño de segmento de 128x128 píxeles puede aplicarse de manera exacta para obtener 24 subimágenes por cada imagen, sumando un total de 127200 subimágenes.

Con esta solución, el tiempo de entrenamiento del modelo se reduce considerablemente, así como el esfuerzo computacional del mismo. Es mucho más sencillo para el *autoencoder* tomar como elemento de entrada una imagen de un tamaño *X* que otra de un tamaño 25 o 50 veces mayor, por ejemplo. También, junto a las subimágenes surgen también las subépocas, es decir, que cada época de entrenamiento se compone de tantas iteraciones como subimágenes obtengamos de la imagen base.

4.2.5 Despliegue

Una de las ideas de base que había al comenzar este proyecto era llegar a ofrecer un producto real, o al menos plantear un prototipo funcional que permita hacer uso del método diseñado. Posterior al diseño y evaluación del método propuesto, se decidió construir un prototipo en forma de aplicación web, con el objetivo de llevar el modelo a un entorno de producción y poder interactuar con él a través de una interfaz sencilla. Este prototipo de aplicación web está en su fase inicial. Incluye un apartado para usar el modelo, es decir, subir una imagen y obtener su versión comprimida, junto con una muestra de sus métricas de calidad. La idea es tener un apartado que muestre de una manera gráfica y sencilla la calidad que ha alcanzado la compresión en base a la imagen resultante obtenida, empleando las métricas usadas y otras diferentes. También incluye otra parte donde se presentan una serie de modelos a usar, siendo el principal el modelo de uso general descrito anteriormente.

Tratando de mantener simple tanto la idea como el proceso, se optó por usar Heroku como plataforma en la nube para desplegar el modelo, ya que su simplicidad ayuda a, en pocos pasos, tener una aplicación web desplegada y funcional. De cara al futuro, se prevé dar el paso a usar servicios de Amazon Web Services como *AppSync* o *Batch* para servir el sistema y otros como S3 para versionar el modelo y almacenar los archivos estáticos asociados. Para la aplicación web, siguiendo con la idea de simplificar el desarrollo, se usó *Flask* para construir la API a través de la cual interactuar con el modelo y se diseñó e implementó una interfaz web sencilla usando *React* y *Material UI*.

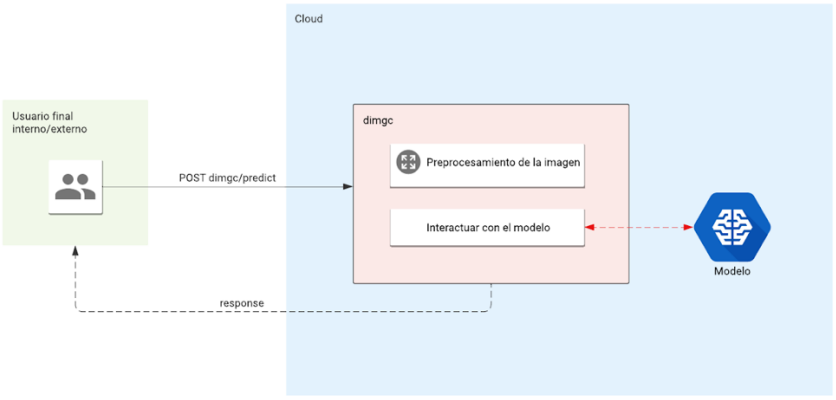


Figura 10 - Diagrama simple planteado para desarrollar la aplicación

La idea principal cuando se planteó el despliegue del modelo era ofrecer el servicio de compresión de imágenes de múltiples maneras. Se pensó como un producto destinado a usuarios generales y también a desarrolladores, y por ello se planteó la idea de distribuir este servicio como una API. Más adelante, y refinando esta idea, cobró también sentido la idea de ofrecerlo como una librería que haga de *wrapper* de dicha API y ofrezca un servicio más variado y abstraído del modelo. Por ejemplo, en una librería de Python que cualquier desarrollador pueda obtener y usar en servicios en la nube donde maneje muchas imágenes en formatos pesados o con grandes dimensiones. De forma provisional, y por la necesidad de darle un nombre, se escogió "dimgc" (*Deep IMaGe Compression*), tanto para el servicio web, como para distribuirlo en forma de librerías para desarrolladores.

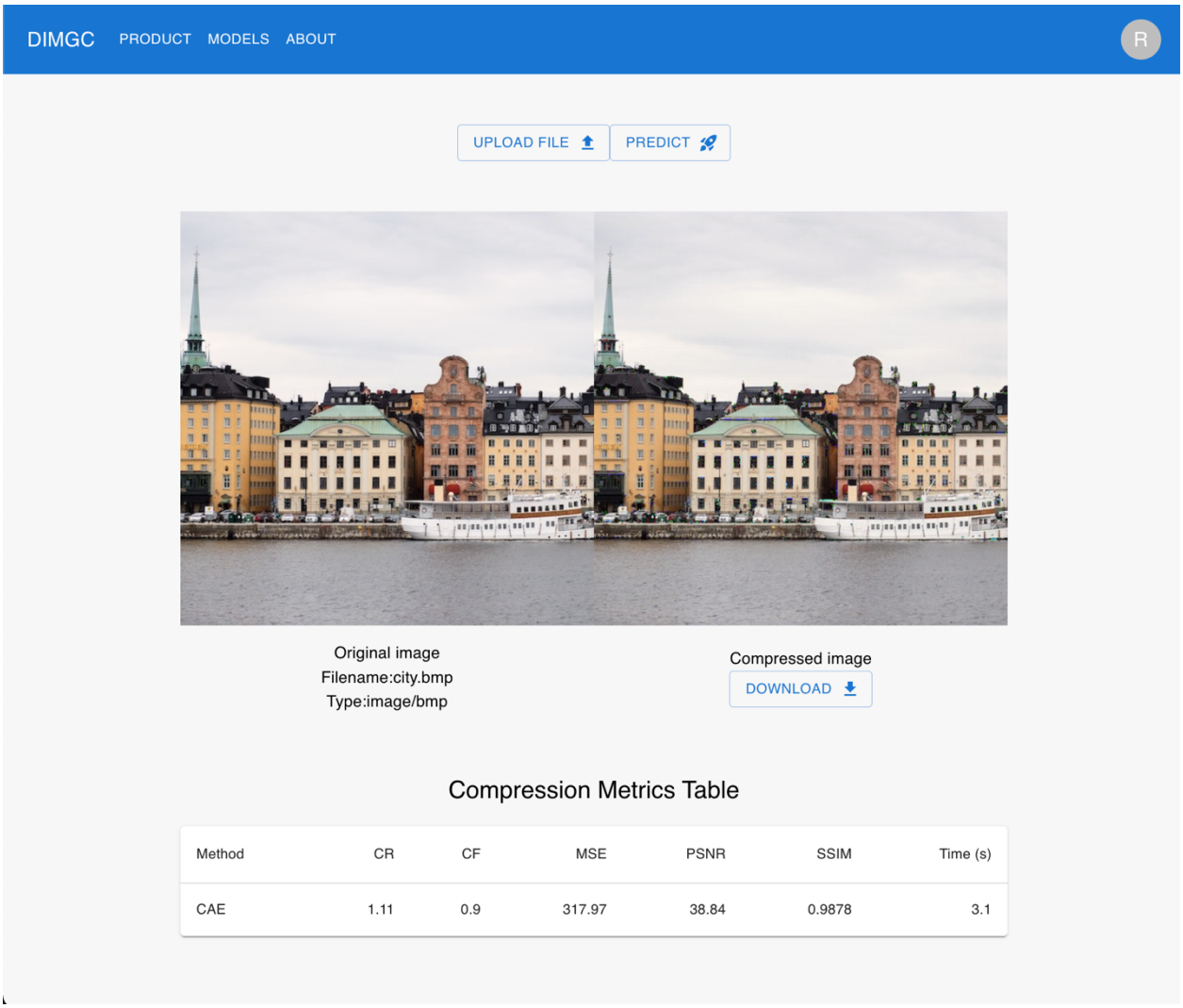


Figura 11 - Muestra del prototipo de aplicación web para el producto "dimgc" (1)

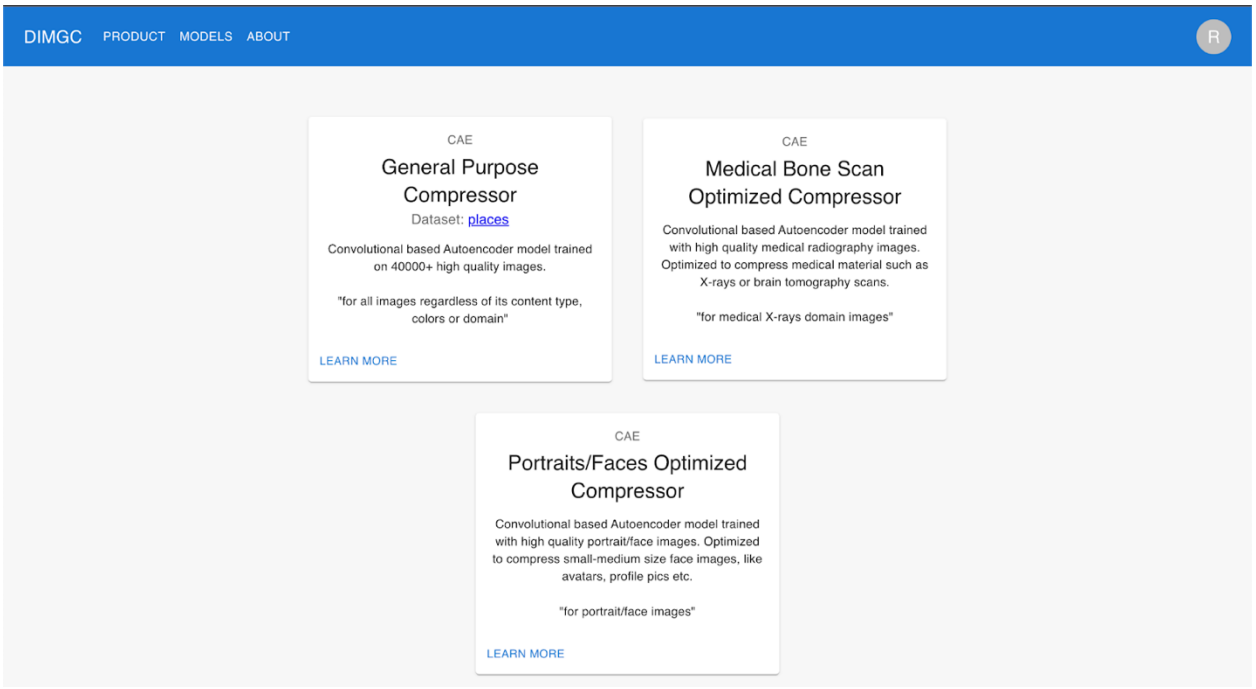


Figura 12 - Muestra del prototipo de aplicación web para el producto "dimgc" (2)

En resumen, estas son ideas secundarias al propósito del presente trabajo pero que se tuvieron desde el inicio del proyecto respecto a qué producto software podría desarrollarse en base al método propuesto, para no dejar todo en la teoría sino para también ofrecer algo que pueda tener espacio en escenarios de uso reales resolviendo problemas reales. Por el momento, el desarrollo se sitúa en la API y la aplicación web que la consume.

4.2.6 Metodología de desarrollo

Como metodología de desarrollo del presente trabajo de investigación se empleó una metodología ágil de desarrollo de software conocida como desarrollo basado en experimentos o *Experiment Driven Development* (EDD) (Melegati et al., 2019). Aplicada de forma metódica y disciplinada, esta metodología consigue garantizar una amplia exploración de posibles resultados, cumplimiento de objetivos y un mayor aprovechamiento del tiempo.

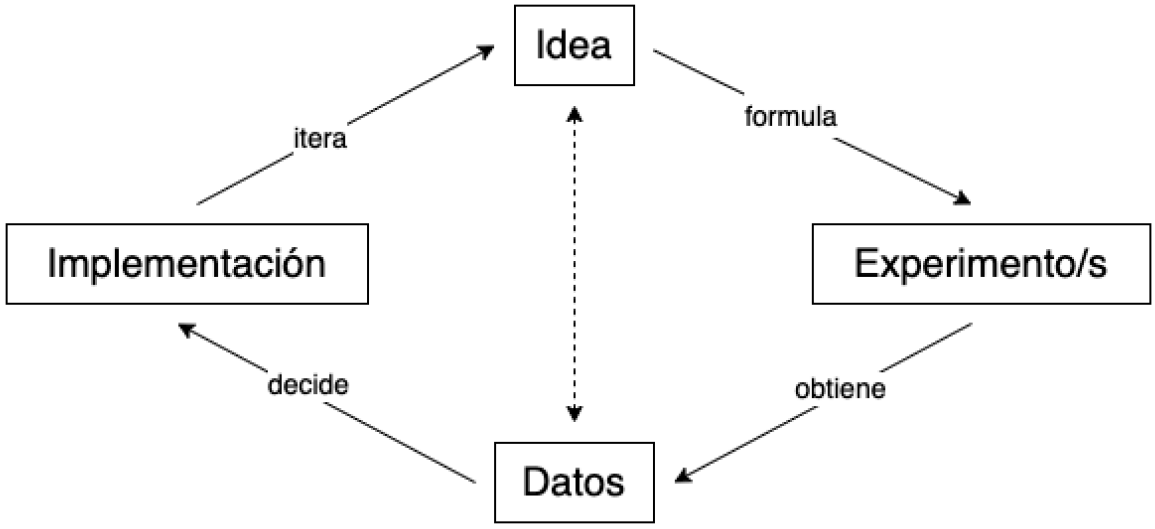


Figura 13 - Diagrama EDD (elaboración propia)

En ingeniería de software, un problema que siempre fue foco de esfuerzos e investigación para paliarlo fue el hecho de identificar el problema adecuado a resolver en el momento adecuado, con el objetivo de minimizar el esfuerzo y el tiempo invertido en resolverlo (*Waste Producing Machine*). Para resolver este reto se necesita priorizar y medir hechos en vez de hipótesis o suposiciones. Justamente es en esto en lo que se basa EDD. Partir de una serie de ideas y asunciones bien meditadas hechas previamente a materializarlas en pequeños ítems entendidos como experimentos. De esta manera, tenemos una serie de hechos medibles y evaluables que nos servirán para conformar una fuente de conocimiento acerca del método a proponer para nuestro problema. Llegar a esta información es la idea principal al aplicar EDD al presente proyecto, desde el lado del desarrollo e implementación del método. Una vez en ese punto, teniendo la información, podemos continuar a su filtrado y posterior análisis cualitativo, o bien pausar y decidir hacer modificaciones para retomar la experimentación.

Aquí es donde EDD es especialmente beneficioso y es éste el principal motivo de haberlo escogido como metodología de desarrollo.

Llevar a cabo un proyecto basado en aprendizaje profundo requiere de mucha experimentación, prototipado y testeo de ideas y soluciones, comparación de resultados, medida de la calidad de la solución obtenida, análisis de resultados etc. Para llegar al método propuesto para el problema de compresión de archivos de imagen, se llevaron a cabo previamente numerosos experimentos. Estos experimentos buscaban probar la validez de ideas y recopilar los resultados, los hechos, y poner a prueba dichos experimentos en casos de uso reales. De esta forma, se iba avanzando en la investigación desechando algunas ideas y adoptando otras nuevas, partiendo en todo momento de una serie de ideas propuestas. El uso de EDD en el caso concreto de este proyecto y para obtener el método propuesto puede dividirse en tres etapas que se realizaron de forma secuencial.

1. **Recopilación de ideas.**

Como primer paso para llegar al objetivo de obtener un método a proponer, y basándose en todo el análisis de la literatura realizado y descrito con anterioridad en este documento, se llevó a cabo un proceso imaginativo y creativo de recopilación de posibles ideas a aplicar para obtener un método a proponer que cumpla con los objetivos propuestos.

La idea de esta primera etapa es reunir, de una manera muy general y de alto nivel, ideas de posibles métodos basados en aprendizaje profundo y arquitecturas que podrían funcionar para el problema a tratar. No solo sobre métodos concretos como tal sino también ideas de cómo testearlos, qué conjuntos de datos emplear, como modificar y variar cómodamente un mismo conjunto de datos para obtener mejores resultados etc. Por ejemplo, se decidió que se probaría cada método obtenido con diferentes conjuntos de datos. Diferente en cuanto al dominio de las imágenes que contenía y también en cuanto al tamaño de éstas, así como su *bitrate* o su tamaño de segmento. También, por el lado de las arquitecturas escogidas, se decidió hacer variaciones en las funciones de activación de las primeras capas de la red, que a menudo son las más influyentes, dado que operan con las primeras etapas de la imagen de entrada, y probar en qué medida afectaban estas variaciones a los resultados finales.

Partiendo de estas ideas y decisiones tomadas en esta etapa, se pasa a la parte de experimentación con un panorama claro y bien definido sobre lo que se iba a hacer, de qué manera y en qué orden.

Podemos decir que esta primera etapa de recopilación de ideas es la base de la cual partir para que en la fase de experimentación maximicemos los buenos resultados, minimicemos el tiempo invertido y establezcamos una serie de reglas a seguir, consiguiendo de esta manera que la fase de experimentación no se convierta en un proceso desestructurado, enredado y sin objetivos claros que pueda llevarnos a extender excesivamente el tiempo dedicado a obtener un método de calidad así como a aumentar la frustración personal.

2. Experimentación y prototipado.

Partiendo de las premisas establecidas en la primera etapa se pasa a esta segunda etapa, que constituye la principal de las tres. Experimentar es probar y examinar la viabilidad en cuanto a virtudes y defectos de una idea particular. El objetivo de experimentar es obtener información de calidad que permita desarrollar nuevos métodos, ideas y procesos, comprender mejor una idea y su implementación y tomar decisiones sobre cómo optimizarlo de manera que mejore su calidad y funcionamiento. Dada la importancia de obtener información de calidad que permita escalar el problema y llevarlo a un mejor desempeño, esta etapa requiere de un tiempo de ejecución y de un esfuerzo de análisis y obtención de conclusiones. Es la parte más densa de todo el ciclo de desarrollo del método propuesto a causa de que se ven involucrados procesos técnicos como por ejemplo la implementación de métodos, entrenamiento de estos, corrección de errores y testeo.

En esta etapa se van recorriendo los métodos elegidos en la etapa anterior y por cada uno de ellos se van haciendo pequeñas variaciones progresivamente para anotar y observar los resultados obtenidos. Cada uno de estos experimentos se materializa en forma de un prototipo, es decir, una implementación sencilla del experimento con el fin de que sea fácil de probar y de modificar y que aporte la información que se necesita. Dicho de otra manera, por cada experimento se acaba teniendo un método basado en aprendizaje profundo diferente. Cada uno de ellos no tiene por qué ser sustancialmente diferente a los demás, sino simplemente, en algunos casos, incluir leves variaciones en cuanto a su arquitectura o en cuanto a su forma de testearlo, como puede ser el conjunto de datos empleado en cuanto a su dominio o características técnicas o el método de entrenamiento y evaluación, la elección de hiperparámetros etc.

Cuando esta etapa de experimentación y prototipado concluye, existirá una pequeña base de datos que recoge toda la información relativa a todos y cada uno de los experimentos que se han realizado, no solo en cuánto a sus resultados sino también en cuanto a sus características particulares. Es esta base de datos la fuente de información que nos permitirá pasar a la siguiente y última etapa y poder llegar a obtener un método de calidad que proponer.

3. Evaluación y comparación de los experimentos.

En esta etapa se busca evaluar y analizar los resultados obtenidos en la etapa anterior con el fin de escoger el mejor de todos para refinarlo posteriormente. En el análisis de los resultados, se comparan entre sí todos los experimentos para tratar de inducir qué rasgos comúnmente caracterizan a los mejores experimentos respecto a los peores. De esta manera se discrimina a los peores experimentos por los resultados que ofrecen y por los rasgos característicos que presentan en relación a los mejores experimentos.

Tras la etapa anterior, se obtuvieron un total de cuarenta y siete experimentos diferentes con sus características particulares y con sus resultados de desempeño. Este número puede ampliarse tanto como el investigador decida en función de la naturaleza del problema que está tratando y de sus limitantes. Desde un principio se consideró obtener alrededor de cincuenta experimentos diferentes para de esta manera tener una base de resultados lo suficientemente amplia como para tener un método a presentar y también, desde el punto de vista de la documentación de la investigación, como para poder sacar conclusiones acerca del problema de compresión de imágenes que luego poder plasmar de forma escrita. Tras evaluar el conjunto de experimentos obtenidos se concluyó que podían identificarse, cualitativamente hablando y a alto nivel, tres grupos diferenciados. Veintiún experimentos entran en el grupo de los no favorables o mediocres, dieciocho en el grupo de aceptables o satisfactorios y ocho en el grupo de buenos o muy satisfactorios.

5. Experimentos y resultados.

Una parte muy importante de la investigación realizada acerca del tema de compresión de imágenes usando aprendizaje profundo, así como en cuanto a la experimentación y el desarrollo del método propuesto, es el análisis de los resultados obtenidos. Este proceso de análisis se plantea en el tercer objetivo específico establecido al comienzo del desarrollo del proyecto. Este análisis de resultados se ha subdividido en dos partes. La primera de ellas es la relativa al análisis de resultados en cuanto al desempeño del método propuesto cualitativamente hablando, y siendo comparado con otros métodos, tanto desarrollados durante la investigación como ajenos. La segunda parte es la relativa al análisis de los resultados de la investigación realizada, revisando el conocimiento adquirido durante todo el proceso, las ideas que dicho conocimiento ha posibilitado alcanzar y la calidad de las propuestas y conclusiones alcanzadas.

Todos los experimentos han sido realizados en un notebook MacBook Pro 2020 corriendo el sistema operativo MacOS Big Sur, con 16Gb de memoria RAM a 3733MHz LPDDR4X y un procesador Intel Core i5 de cuatro núcleos a 2GHz de velocidad.

El método propuesto se ha comparado con los algoritmos JPEG y JPEG2000, siguiendo la práctica más común en los estudios publicados presentes en la literatura, y buscando comparar el método propuesto con el referente más ampliamente usado en la actualidad en cuanto a compresión de imágenes digitales con pérdida. El método propuesto ha alcanzado unos niveles de desempeño y resultados similares a los de JPEG y estando, en términos cualitativos, como era esperado, por debajo de la versión mejorada de JPEG, JPEG2000.

Se ha llevado a cabo un análisis de los resultados desde un punto de vista objetivo y subjetivo, siendo el primero el relativo a métricas de análisis de imágenes comprimidas, basadas en teoría matemática. El punto de vista subjetivo se introduce porque, al fin y al cabo, las imágenes digitales son muestras de la realidad destinadas a ser vistas por humanos, y aportar el punto de vista humano a la hora de valorar los resultados de un proceso de compresión aplicado a imágenes da, por así decirlo, una segunda opinión valiosa y que también ha de considerarse. Si al comprimir una imagen, a simple vista, como humanos, notamos ligeras diferencias de brillo o intensidad de color en ciertas partes, es un hecho significativo y destacable a valorar y a mencionar como parte del análisis de resultados.

La evaluación del modelo se ha hecho haciendo uso del *dataset* de imágenes Kodak PhotoCD. Se trata de un conjunto de datos con 25 imágenes sin comprimir en formato PNG publicadas por la empresa Kodak,

compañía multinacional estadounidense que se dedica al diseño, comercialización y producción de sistemas y equipos fotográficos. Este pequeño conjunto de datos es usado frecuentemente para medir el desempeño de métodos y sistemas relacionados con el tratamiento de imágenes, por ofrecer imágenes sin comprimir con una alta calidad y definición.

De este modo, en primer lugar, se presentan los resultados obtenidos junto con un análisis objetivo de los mismos, siguiendo el uso de las métricas más populares empleadas en la literatura para este tipo de casos, PSNR y SSIM, mencionadas previamente.

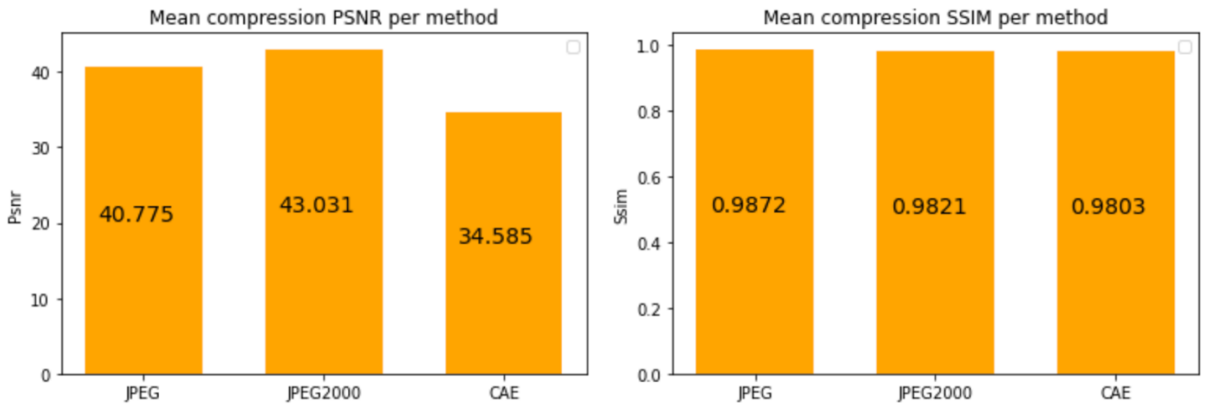


Figura 14 - Comparación del CAE propuesto con JPEG y JPEG2000 en términos de PSNR y SSIM

Tabla 1 - JPEG, JPEG2000 y el CAE propuesto comparados en cuanto a PSNR y SSIM

Método	PSNR	SSIM
JPEG	40.75	0.9872
JPEG2000	43.04	0.9821
CAE	34.58	0.9803

En la Figura 10 tenemos la comparación al respecto de las métricas PSNR y SSIM de los tres métodos considerados. El método propuesto iguala los resultados ofrecidos por JPEG y JPEG2000 en términos de SSIM, sin embargo, en términos de PSNR se sitúa algo por detrás de ambos. Este comportamiento también se ha observado en algunos de los estudios revisados en la literatura. Podemos decir, desde un punto de vista teórico, que PSNR es una métrica que evalúa más por el lado del ruido que la imagen tiene y SSIM está basado en la iluminación, el contraste y las diferencias estructurales, pero a pesar de ello, no hay una

evidencia clara de que una métrica se sitúe por encima de la otra en términos de validez (Horé & Ziou, 2010).

Comparándolo con los referentes de la literatura, el método propuesto supera en términos de PSNR y SSIM al método propuesto por Akyazi et al. Respecto al método propuesto por Theis et al., en términos de SSIM, el CAE propuesto en el presente documento ofrece unos resultados muy similares, pero en cuanto al PSNR se sitúa ligeramente por detrás. El mismo resultado se puede observar comparándolo con los resultados publicados por Cheng et al. Estos tres estudios de referencia han sido comentados con detenimiento en la revisión de la literatura del punto 3 de este documento.

Tabla 2 – Tiempo medio de ejecución de JPEG, JPEG2000 y el CAE para los experimentos realizados

Método	Tiempo (s)
JPEG	0.01
JPEG2000	0.19
CAE	2.40
Cheng et al.	2.29
Balle et al.	7.39

Pasando ahora a los tiempos de ejecución, como era esperado, son superiores a los presentados por JPEG o JPEG2000, dado que la implementación de estos algoritmos es de más bajo nivel (usando C) y por lo tanto son bastante más eficientes computacionalmente hablando. En la Tabla 2 se comparan con los resultados presentados por Cheng et al. y Balle et al., siendo el método propuesto claramente superior al de Balle et al. y comparable al de Cheng et al. Hay que considerar que los tiempos de ejecución son correspondientes a un proceso completo de *encoder* y *decoder* para imágenes a color de 768x512 píxeles, así como Cheng et al. lo plantea, y a diferencia de Balle et al., que presenta esos tiempos de ejecución para imágenes en blanco y negro.

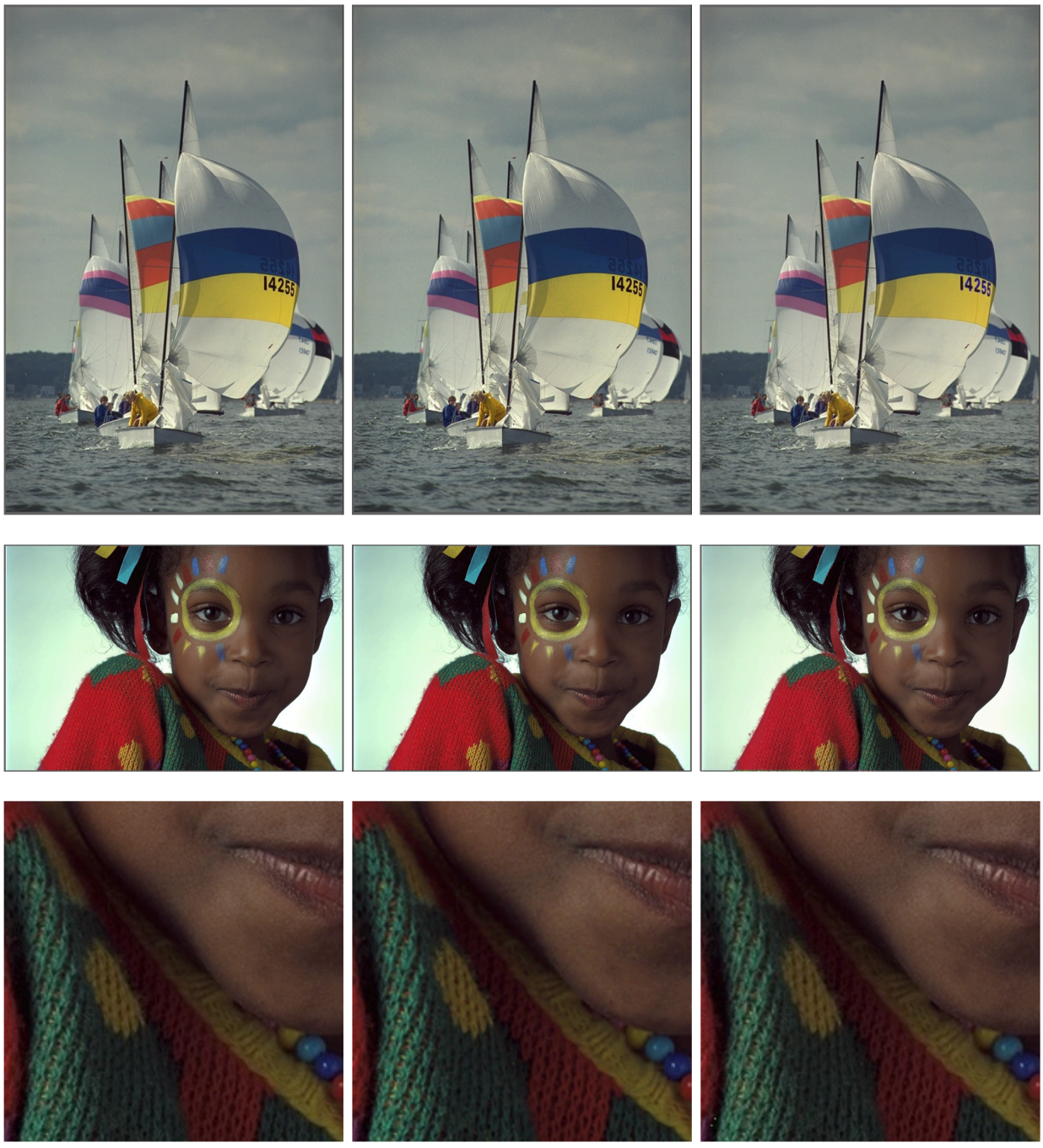


Figura 15 - Muestra del comportamiento del CAE en imágenes de evaluación

Pasando ahora al análisis subjetivo de los resultados, a simple vista podemos percibir ligeras diferencias entre las imágenes de la figura anterior. Observando con mucho detenimiento la Figura 14, donde tenemos la imagen comprimida usando JPEG, JPEG2000 y el CAE propuesto, respectivamente, pueden percibirse colores ligeramente más intensos en las imágenes del centro, las de JPEG2000, y un mayor nivel de iluminación, acorde a la calidad superior de compresión que ofrece este método sobre JPEG. Para la primera

y la tercera, JPEG y el método propuesto respectivamente, las diferencias desde un punto de vista subjetivo son prácticamente imperceptibles. Lo más destacable que se ha observado en las pruebas realizadas son algunos casos concretos donde podemos dar con imágenes donde el método propuesto resalta más los tonos azules comparado con JPEG.

Más ejemplos de imágenes de diferente contenido y dimensiones, comprimidas empleando los tres métodos, a tamaño completo, junto con su original, tanto extraídas del conjunto de datos Kodak PhotoCD como de otras fuentes, se pueden encontrar en el Anexo A del presente documento.

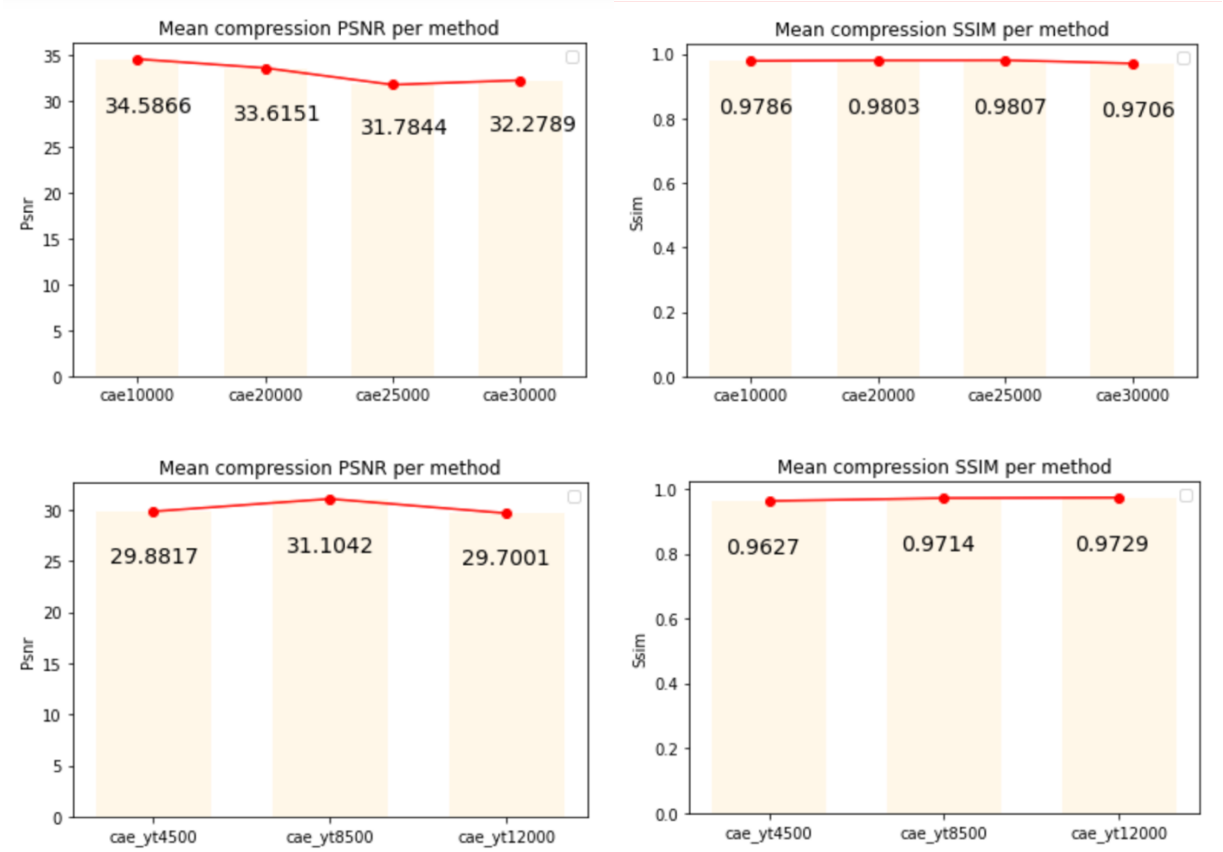


Figura 16 - SSIM y PSNR para diferentes épocas de entrenamiento y para conjuntos de datos dataset places y YouTube-8M

Por otro lado, y en último lugar en esta fase de análisis de los resultados observados, en los gráficos de la Figura 11 se muestra la evolución de los valores de PSNR y SSIM para el modelo propuesto con relación a las épocas de entrenamiento llevadas a cabo, para observar como varía la calidad de los resultados con más o menos tiempo de entrenamiento. Se muestran gráficos para los dos conjuntos de datos que mejores resultados dieron de entre los probados, YouTube-8M (parte inferior de la Figura 11) y *places dataset* (parte superior de la figura), siendo este último el que mejores resultados ofrecía y el usado para entrenar el modelo

final. Usando el conjunto de datos YouTube-8M, los resultados son más lineales y constantes, pero en el caso del *dataset* utilizado, *places dataset*, para el PSNR el valor va decreciendo hasta estabilizarse, no siendo así para el caso del SSIM, que se mantiene más constante con el tiempo. Se concluye que el número de épocas más estable sería entre 20000 y 25000 para este caso.

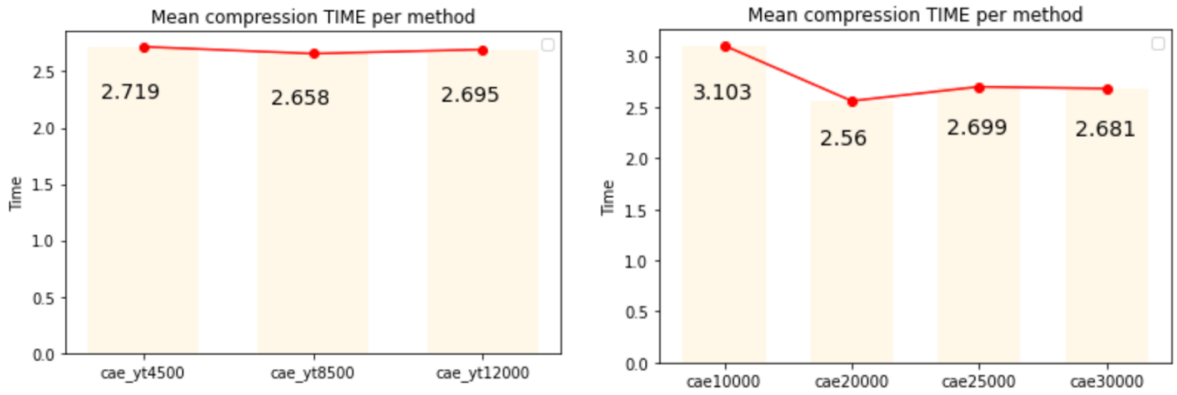


Figura 17 – Tiempo de ejecución para diferentes épocas de entrenamiento y para conjuntos de datos dataset places y YouTube-8M

Para el caso del tiempo de ejecución, también se comparó su evolución con relación al número de épocas de entrenamiento al que se sometió al modelo. Vemos que para el *dataset* utilizado ese tiempo disminuye a partir de las 10000 épocas y para el caso del conjunto de datos YouTube-8M se mantiene en valores superiores a los anteriores pero constantes.

A modo de conclusión de este análisis de resultados, se ha conseguido llegar a un modelo que cumple los objetivos propuestos y proporciona resultados satisfactorios, tanto desde el punto de vista objetivo como subjetivo. Iguala, y en casos particulares supera ligeramente, el desempeño de JPEG y se queda unos escalones por detrás de su versión mejorada, JPEG2000. Al respecto de algunos de los estudios presentes en la literatura, para la métrica SSIM especialmente, el CAE propuesto está por encima, así como en tiempos de ejecución para el caso de Balle et al. Por último, este análisis de resultados es muy útil al servir como base para plantear las posibles mejoras que el modelo debe tener de cara una versión mejorada del mismo en un futuro, para seguir escalando su funcionamiento e ir refinando progresivamente su calidad.

6. Conclusiones.

En el presente trabajo se abordó el problema de compresión de imágenes empleando técnicas de aprendizaje profundo de manera ordenada y progresiva. Se comenzó con una aproximación al tema leyendo algunas publicaciones introductorias y pasando progresivamente a revisar estudios más específicos y complejos. Se poco a poco fue elaborando, en base a ideas obtenidas revisando algunos *papers* en específico, el modo en el que podían lograrse los objetivos propuestos de forma satisfactoria.

Desde el lado del desarrollo y diseño del método, empleando una metodología de experimentación, se pudo pasar de pequeños prototipos que sirvieron para comprender a alto nivel el problema a resolver y sus posibles implicaciones y dificultades, a pruebas de concepto más complejas y completas funcionalmente hasta llegar a un resultado final que pudo ser evaluado y comparado con otros métodos y estudios.

Desde un punto de vista personal, profundizar en el tema de compresión de datos desde el lado de las imágenes ha resultado enriquecedor y muy interesante. Los objetivos planteados en el inicio del proyecto han sido satisfactoriamente cumplidos y se han añadido funcionalidades e ideas adicionales que han surgido posteriormente, como por ejemplo una idea de proyecto que eventualmente puede llegar a funcionar y a ocupar un espacio valioso en algunos escenarios de uso reales. Especialmente, el proceso de experimentación y las primeras aproximaciones prácticas al problema tratado resultaron la parte más interesante y amena de todo el proceso, junto también a investigación y revisión de la literatura que amplió el abanico de métodos e ideas a aplicar en la propia.

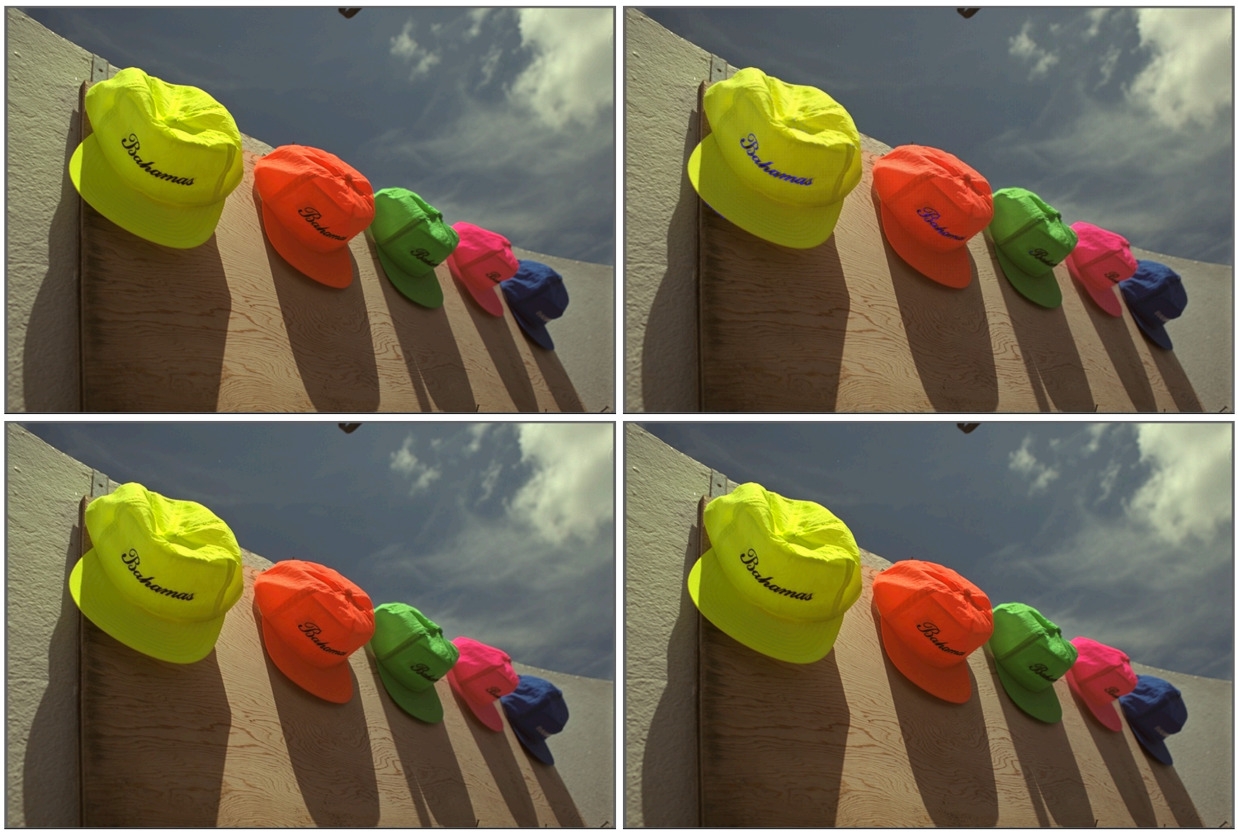
7. Trabajo futuro.

De cara al trabajo adicional y complementario a realizar eventualmente en el futuro, bien en los meses posteriores o a más largo plazo, se han planteado una serie de líneas de trabajo que se pueden seguir:

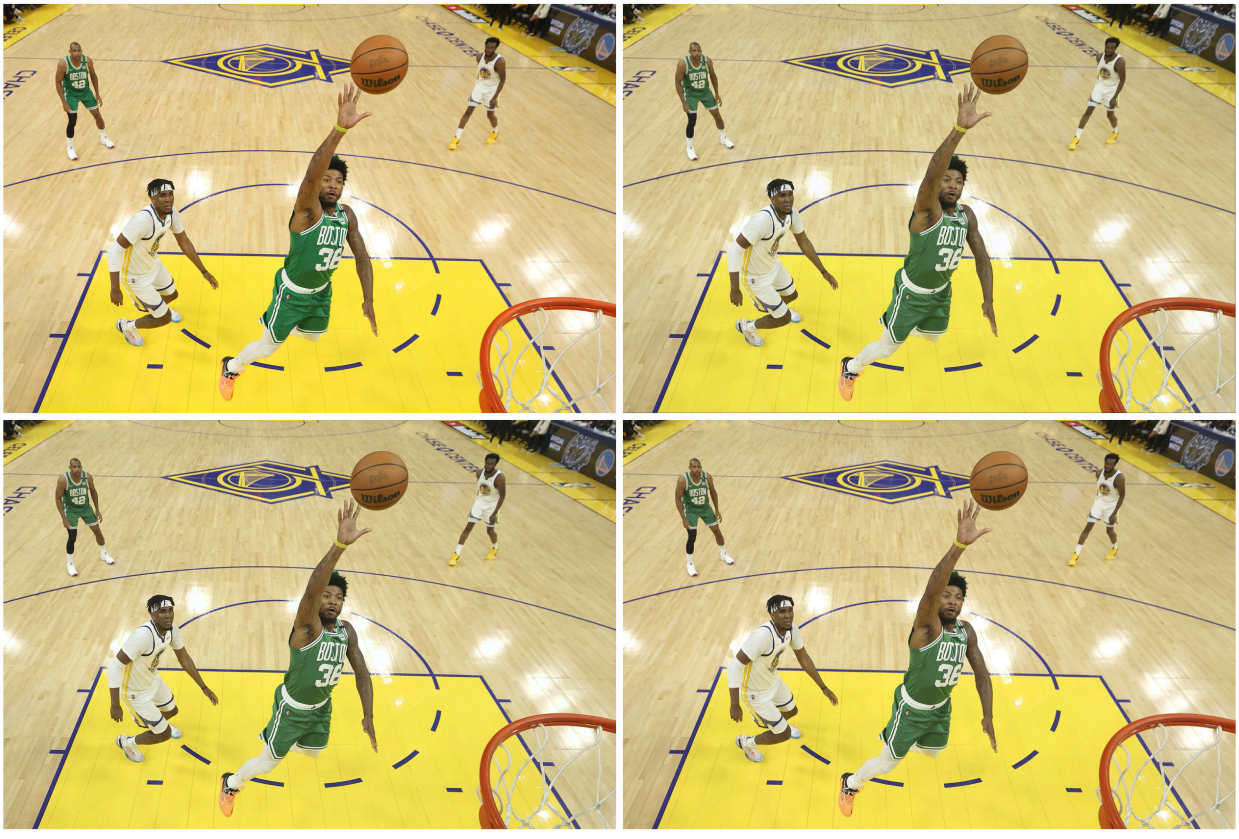
- Retomar la fase de experimentación para tratar de llevar al modelo propuesto al siguiente nivel, buscando resultados cada vez más satisfactorios. En definitiva, seguir experimentando con el modelo en búsqueda de mejoras.
- Desarrollar más modelos específicos personalizados a diferentes usos del método de compresión, como por ejemplo con imágenes de sensores, imágenes satelitales, imágenes de microscopios o imágenes lineales, entre otros.
- Plantearse la incorporación de un módulo de posprocesamiento posterior al CAE, para evaluar si la calidad de la imagen resultante puede mejorarse con él.
- Expandir el trabajo realizado con el servicio propuesto en el punto 4.2.5, *dimgc*, llevándolo a más campos de aplicación. Complejizar la API, añadiendo más funcionalidades, como por ejemplo recibir métricas para evaluar el proceso de compresión, seleccionar el nivel de compresión aplicado o poder comunicarse con servicios en la nube.
- Desplegar el servicio *dimgc* en un entorno de Amazon Web Services, interconectándolo con otros servicios en la nube.
- Integrar *dimgc* con otros lenguajes de programación populares en la red, como por ejemplo Javascript.

Anexo: Muestra de imágenes completas.

A continuación, se aporta una muestra de diferentes imágenes junto a sus versiones comprimidas empleando los tres métodos contemplados en el punto 5, análisis de resultados, del presente documento. En la esquina superior izquierda se muestra la imagen original, seguida de la imagen comprimida usando el método propuesto (CAE) a su derecha, y las imágenes comprimidas con JPEG y JPEG2000, respectivamente, en las esquinas inferior izquierda y derecha.




DEEP FEAR
DEEP FEAR
DEEP FEAR
DEEP FEAR



8. Referencias.

Abbas, H. (n.d.). *Neural model for Karhunen-Loéve transform with application to adaptive image compression*.

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., & Vijayanarasimhan, S. (2016). *YouTube-8M: A Large-Scale Video Classification Benchmark*. http://arxiv.org/abs/1609.08675

Ahlswede, R., Ahlswede, A., Althöfer, I., Deppe, C., & Tamm, U. (2014). LZW Data compression. *Foundations in Signal Processing, Communications and Networking*, *10*(02), 9–38. https://doi.org/10.1007/978-3-319-05479-7_2

Akyazi, P., & Ebrahimi, T. (2019). Learning-Based Image Compression using Convolutional Autoencoder and Wavelet Decomposition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2019-June*, 1–5.

Antoine, J.-P. (2003). Wavelet Transforms and Their ApplicationsWavelet Transforms and Their Applications , Lokenath Debnath , Birkhäuser, Boston, 2002. $79.95 (565 pp.). ISBN 0-8176-4204-8 . *Physics Today*, *56*(4), 68–68. https://doi.org/10.1063/1.1580056

AVIF for Next-Generation Image Coding | by Netflix Technology Blog | Netflix TechBlog. (n.d.). Retrieved June 18, 2022, from https://netflixtechblog.com/avif-for-next-generation-image-coding-b1d75675fe4

Ballé, J., Laparra, V., & Simoncelli, E. P. (2017). End-to-end optimized image compression. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Cai, C., Lu, G., Hu, Q., Chen, L., & Gao, Z. (2019). Efficient learning based sub-pixel image compression. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2019-June*.

Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2018). Deep Convolutional AutoEncoder-based Lossy Image Compression. *2018 Picture Coding Symposium, PCS 2018 - Proceedings*, 253–257. https://doi.org/10.1109/PCS.2018.8456308

Chest X-Ray Images (Pneumonia) | Kaggle. (n.d.). Retrieved June 19, 2022, from https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia?resource=download

Chua, L., & Lin, T. (1988). *A neural network approach to transform image coding*.

Cramer, C., Gelenbe, E., & Bakircioglu, H. (1996). Video compression with random neural networks. *Proceedings of International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing, NICROSP*, 476–484. https://doi.org/10.1109/nicrsp.1996.542792

Daugman, J. G. (1988). Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *36*(7), 1169–1179. https://doi.org/10.1109/29.1644

Dhawale, N. (2015). Implementation of Huffman algorithm and study for optimization. *Proceedings - 2014 IEEE International Conference on Advances in Communication and Computing Technologies, ICACACT 2014*. https://doi.org/10.1109/EIC.2015.7230711

Du, B., Yuang, D., & Zhang, H. (2022). *Collaborative image compression and classification with multi-task learning for visual Internet of Things*.

Electronics, A. K., & Paper, W. (n.d.). *8k: the next level in imaging*. 1–15.

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, *32*(14–15), 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Gelenbe, E. (1989). Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Computation*, *1*(4), 502–510. https://doi.org/10.1162/neco.1989.1.4.502

Gelenbe, E., & Sungur, M. (1994). Random network learning and image compression. *IEEE International Conference on Neural Networks - Conference Proceedings*, *6*, 3996–3999. https://doi.org/10.1109/icnn.1994.374852

GitHub - Netflix/vmaf: Perceptual video quality assessment based on multi-method fusion. (n.d.). Retrieved June 18, 2022, from https://github.com/Netflix/vmaf

Hai, F., Hussain, K. F., & Gelenbe, E. (2001). *Video compression with wavelets and random neural network approximations*.

Horé, A., & Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM. *Proceedings - International Conference on Pattern Recognition*, *August*, 2366–2369. https://doi.org/10.1109/ICPR.2010.579

LeCun, Y., & Bengio, Y. (2015). *Deep Learning, Nature, vol. 521, no. 7553, p. 436*.

Melegati, J., Wang, X., & Abrahams, P. (2019). *Hypotheses Engineering: First Essential Steps of Experiment-Driven Software Development*.

Rippel, O., & Bourdev, L. (2017). *Real-time adaptive image compression,*.

S. Golomb. (1966). *Run-length encodings (Corresp.),*. 6–8.

Sambasivan, N., Kapania, S., & Highfll, H. (2021). Everyone wants to do the model work, not the data work: Data cascades in high-stakes ai. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3411764.3445518

Si, Z., & Shen, K. (2016). Research on the WebP image format. *Lecture Notes in Electrical Engineering*, *369*, 271–277. https://doi.org/10.1007/978-981-10-0072-0_35

Skodras, A., Christopoulos, C., & Ebrahimi, T. (2001). The JPEG 2000 still image compression standard.

IEEE Signal Processing Magazine, *18*(5), 36–58. https://doi.org/10.1109/79.952804

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 1–14.

Theis, L., Shi, W., Cunningham, A., & Huszár, F. (2017). Lossy image compression with compressive autoencoders. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 1–19.

VMAF: The Journey Continues. by Zhi Li, Christos Bampis, Julie… | by Netflix Technology Blog | Netflix TechBlog. (n.d.). Retrieved June 18, 2022, from https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12

Wallace, G. K. (1992). The JPEG still Picture Compression Standard. *Architecture*, *38*(1).

Wang, Y., Wang, L., Yang, J., An, W., & Guo, Y. (2019). Flickr1024: A large-scale dataset for stereo image super-resolution. *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, 3852–3857. https://doi.org/10.1109/ICCVW.2019.00478

Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical Evaluation of Rectified Activations in Convolutional Network*. http://arxiv.org/abs/1505.00853

Yamanaka, J., Kuwashima, S., & Kurita, T. (2017). Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10635 LNCS*, 217–225. https://doi.org/10.1007/978-3-319-70096-0_23

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2018). Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(6), 1452–1464. https://doi.org/10.1109/TPAMI.2017.2723009

Zhou, L., Cai, C., Gao, Y., Su, S., & Wu, J. (2018). Variational autoencoder for low bit-rate image compression. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2018-Janua*, 2617–2620.