ESCOLA TÉCNICA SUPERIOR D'ENGINYERIA – ETSE-UV

VNIVERSITAT DE VALÈNCIA

Máster de Inteligencia Artificial Avanzada y Aplicada

Trabajo de fin de máster:

Aprendizaje profundo aplicado a técnicas de imagen con luz estructurada y detección con un solo píxel

---

Autor: Carlos Chabert Ull

Tutor: Valero Laparra Pérez-Muelas

Valencia, septiembre de 2023

RESUMEN

La Inteligencia Artificial ha empezado a formar parte de nuestra vida cotidiana, influenciando dispositivos y servicios que utilizamos a diario, desde teléfonos móviles hasta publicidad en línea o cámaras fotográficas. Estos son algunos de los dispositivos que utilizan Inteligencia Artificial para su funcionamiento, no obstante, nuevas aplicaciones van surgiendo día tras día resultado de la evolución de las arquitecturas utilizadas y las unidades de procesamiento donde se ejecutan.

En el presente Trabajo Final de Máster (TFM) se estudia una aplicación novedosa de la Inteligencia Artificial: la técnica de imagen con luz estructurada y detección con un solo píxel, o *Single-Pixel Imaging* (SPI) en conjunto con técnicas de *Compressive Sensing* (CS). El objetivo es emplear redes neuronales, como alternativa a las técnicas convencionales de reconstrucción de este tipo de imágenes, que permitan obtener la misma calidad de imagen con un número menor de medidas.

Dentro del trabajo, se modela numéricamente el funcionamiento de una cámara SPI, lo que permite simular los experimentos ópticos con los que se han generado los conjuntos de datos. Estos conjuntos de datos posteriormente se utilizan para entrenar los modelos de aprendizaje profundo. Se realiza una comparación de distintos modelos con el objetivo de mejorar la calidad de las imágenes reconstruidas.

RESUM

La Intel·ligència Artificial ha començat a formar part de la nostra vida quotidiana, influenciant dispositius i serveis que utilitzem diàriament, des de telèfons mòbils fins a publicitat en línia o càmeres fotogràfiques. Aquests són alguns dels dispositius que utilitzen Intel·ligència Artificial per al seu funcionament, no obstant això, noves aplicacions van sorgint dia rere dia com a resultat de l'evolució de les arquitectures utilitzades i les unitats de processament on s'executen.

En el present Treball Final de Màster (TFM) s'estudia una aplicació innovadora de la Intel·ligència Artificial: la tècnica d'imatge amb llum estructurada i detecció amb un sol píxel, o Single-Pixel Imaging (SPI) juntament amb tècniques de Compressive Sensing (CS). L'objectiu és emprar xarxes neuronals, com a alternativa a les tècniques convencionals de reconstrucció d'este tipus d'imatges, que permeten obtenir la mateixa qualitat d'imatge amb un nombre menor de mesures.

Dins del treball, es modela numèricament el funcionament d'una càmera SPI, la qual cosa permet simular els experiments òptics amb els quals s'han generat els conjunts de dades. Estos conjunts de dades posteriorment s'utilitzen per a entrenar els models d'aprenentatge profund. Es realitza una comparació de diferents models amb l'objectiu de millorar la qualitat de les imatges reconstruïdes.

ABSTRACT

Artificial Intelligence has begun to become a part of our everyday life, influencing devices and services we use daily, from mobile phones to online advertising or cameras. These are some of the devices that use Artificial Intelligence for their operation, however, new applications are emerging day by day as a result of the evolution of the architectures used and the processing units where they run.

In the present Master's Final Project (TFM), a novel application of Artificial Intelligence is studied: the technique of structured light imaging and detection with a single pixel, or Single-Pixel Imaging (SPI) in conjunction with Compressive Sensing (CS) techniques. The goal is to use neural networks as an alternative to conventional image reconstruction techniques, allowing the same image quality to be achieved with fewer measurements.

Within the work, the operation of an SPI camera is numerically modeled, allowing optical experiments to be simulated with which the data sets have been generated. These data sets are subsequently used to train the deep learning models. A comparison of different models is made with the aim of improving the quality of the reconstructed images.

CONTENIDO

1 INTRODUCCIÓN

El campo del procesamiento de imágenes está ganando popularidad dentro del mundo de la inteligencia artificial (IA). El objetivo de esta disciplina es transformar y mejorar imágenes con el propósito de extraer información que pueda ser de utilidad. Anteriormente a la aparición de la I.A., dichas transformaciones se llevaban a cabo mediante la extracción manual de características como, por ejemplo, a través del uso de diversos filtros. No obstante, el procesamiento de imágenes apoyado en I.A. se fundamenta en la habilidad intrínseca de las redes neuronales para reconocer y discernir patrones presentes en las imágenes.

Una aplicación común y destacada en el procesamiento de imágenes es la reducción de ruido en las mismas [1]. Existen variadas topologías de redes neuronales diseñadas para esta tarea; sin embargo, su eficacia puede ser influenciada por múltiples factores, tales como la calidad del conjunto de datos de entrenamiento y las características particulares del ruido.

1.1 Motivación

Una de las áreas donde se ha observado un progreso significativo en el campo de la IA es en la generación y manipulación de imágenes. Por citar ejemplos, Midjourney y Dall-E 2 han supuesto un hito en la generación de imágenes, y ha ampliado la percepción que teníamos sobre las capacidades artísticas de las máquinas.

La I.A. abre un abanico de posibilidades para mejorar o implementar nuevas técnicas de procesamiento de imagen, para aumentar la calidad o como veremos en el caso de la cámara single píxel para aumentar la velocidad de adquisición al reducir el número de patrones capturados necesarios para generar una escena con una calidad aceptable.

El elevado ritmo al cual la I.A. está avanzando se debe a la intensa investigación tanto del sector académico como del privado. La velocidad a la que se aparecen nuevos modelos y técnicas hace que mantenerse actualizado sea difícil. Por esta razón es una motivación de este TFM recopilar el estado actual de las topologías de redes neuronales orientadas a la reducción de ruido en imágenes.

Este TFM constituye una etapa preliminar de un proyecto más ambicioso donde se desea implementar una red neuronal en un sistema embebido basado en tecnología de matriz de puertas lógicas programables (FPGA), para mejorar las prestaciones en las técnicas de análisis de imágenes obtenidas con luz estructurada.

1.2 Objetivos

El objetivo de este TFM es comparar y analizar distintos modelos de aprendizaje profundo para mejorar la calidad de las imágenes reconstruidas con técnicas SPI y CS. Concretamente se busca atenuar el ruido que se añade a las imágenes durante la reconstrucción de las mismas cuando se reduce el número de medidas respecto a un valor óptimo. Asimismo, como segundo objetivo del proyecto se desea explorar la viabilidad de reconstruir la imagen directamente a partir del vector de intensidades de Hadamard.

Para comenzar, se ha llevado a cabo un exhaustivo estudio del estado del arte relacionado con modelos de aprendizaje profundo enfocados en la reducción de ruido. Tras esta revisión, se procedió a validar su eficacia frente a un ruido de tipo Gaussiano.

El siguiente paso ha sido realizar las simulaciones numéricas que emulan los experimentos ópticos. Las simulaciones fueron hechas mediante Python usando la librería Sympy, dado que tiene implementado el algoritmo de Fast Walsh Hadamard y su correspondiente inversa. Con esto se ha conseguido automatizar la generación de los conjuntos de datos, que de haberse realizado de manera experimental hubiera requerido mucho más tiempo.

Posteriormente se ha llevado a cabo el entrenamiento de los modelos, para ello se han entrenado dos topologías de redes neuronales distintas: Autoencoder Convolucional y U-Net. Para cada una de estas topologías se han utilizado diversos conjuntos de datos donde se ha variado diferentes aspectos como la naturaleza de las imágenes, el tipo de ruido y su grado de intensidad.

Para cada uno de estos modelos entrenados se han evaluado los resultados obtenidos variando distintos hiperparámetros y obteniendo métricas de rendimiento.

El flujo de trabajo adoptado en este TFM sigue una estructura bastante común en el ámbito de la ciencia de datos [2], con la particularidad de que el conjunto de datos es generado a partir de simulaciones numéricas. La estructura de éste se muestra en la siguiente figura:

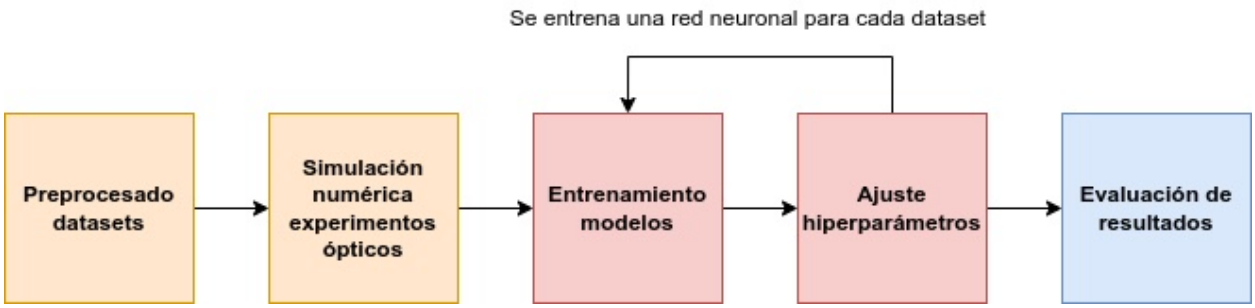


Ilustración 1 - Flujo de trabajo

1.3 Uso y futuro

El uso de la IA se ha democratizado en los últimos años ha permitido que no sean necesarios extensos conocimientos ni recursos para entrenar modelos. Es un campo que ha ganado mucha popularidad pues frecuentemente se liberan modelos capaces resolver problemas complejos, que están revolucionando el mundo tal y como lo conocemos. Podemos diferenciar dos grandes mercados para la IA: los centros de datos y el *big data*, y por otro lado los sistemas embebidos y la ejecución en el borde. Este TFM se enfocará en esta última categoría.

Los sistemas embebidos se caracterizan por tener ciertas restricciones, como limitaciones en cuanto a capacidad de computación y cantidad de memoria disponible. Estos sistemas están destinados en gran parte a dispositivos portátiles, por esta razón el tipo de arquitecturas, la complejidad y el tamaño de los modelos que podemos ejecutar es mucho más limitado comparado con los vastos recursos de los centros de datos. Por ello se suelen utilizar técnicas como la cuantización y topologías de redes más reducidas y que necesitan menos recursos para hacer la inferencia, además se suelen utilizar para resolver problemas específicos en lugar de ser modelos generalistas. Por mencionar algunos ejemplos de aplicaciones de redes neuronales en imagen en sistemas embebidos, podemos destacar los siguientes: segmentación semántica, reconocimiento de objetos, detección de rostros o reducción de ruido.

La aplicación objeto de estudio en este TFM es la reducción de ruido generado por la reconstrucción de imágenes reconstruidas a partir de la técnica de SPI (del inglés *single pixel camera*) combinada con técnicas de CS (del inglés *compressive sensing)*. Las cámaras que utilizan esta tecnología son de interés para el desarrollo de nuevas técnicas de imagen biomédica y cámaras hiperespectrales.

La integración de redes neuronales en sistemas embebidos autónomos presenta numerosos beneficios: reducción de latencia, mayor garantía de privacidad y la eliminación de la dependencia de conectividad a Internet.

En futuras etapas del presente proyecto se desea hacer la inferencia en un dispositivo basado en puertas lógica programables (FPGA), por lo tanto, se optará por un enfoque orientado a los sistemas embebidos en el cual el modelo tendrá un número reducido de coeficientes que pueda ser inferido por este tipo de plataformas hardware.

2 REVISIÓN DEL ESTADO DEL ARTE EN TÉCNICAS DE ATENUACIÓN DE RUIDO EN IMÁGENES OBTENIDAS POR TÉCNICAS DE UN SOLO PÍXEL

Cuando nos referimos en imágenes nos encontramos con distintos tipos de ruido cuyas características depende de su origen. Este ruido puede originarse durante la captura, la transmisión o el procesamiento de la imagen.

Por mencionar algunos ejemplos, los dispositivos fotoelectrónicos generalmente manifiestan una distribución de ruido de tipo gaussiana, mientras que un ruido derivado de fallos durante la transmisión debido a la falta de integridad de señal suele presentarse como un ruido del tipo “sal y pimienta”.

En este trabajo nos centraremos en dos tipos de ruido. El primero de ellos se genera durante la etapa de la adquisición de la imagen, y es introducido por el propio sensor fotoelectrónico. El segundo tipo emerge durante el proceso de reconstrucción de la imagen con técnicas de *compressive sensing*, a raíz de emplear un número de medidas menor al óptimo durante el proceso de adquisición de información, para reducir el tiempo de medida.

Para abordar el análisis del ruido introducido durante la etapa de adquisición, se pretende simular numéricamente el ruido gaussiano generado por los sensores fotoelectrónicos que se utilizan en el laboratorio. Con el objetivo de optimizar la generación del conjunto de datos, el modelo del sensor utilizado es el fotodiodo de avalancha modelo APD440A del fabricante “Thorlabs”. Es importante destacar que modelizar su comportamiento de manera precisa sería complejo, pues su respuesta depende de la temperatura de trabajo, la longitud de onda y la velocidad de adquisición. Por ello, se ha simplificado el ruido generado por un modelo equivalente suponiendo que el fotodiodo genera un ruido aditivo de perfil gaussiano (ver ilustración 2).

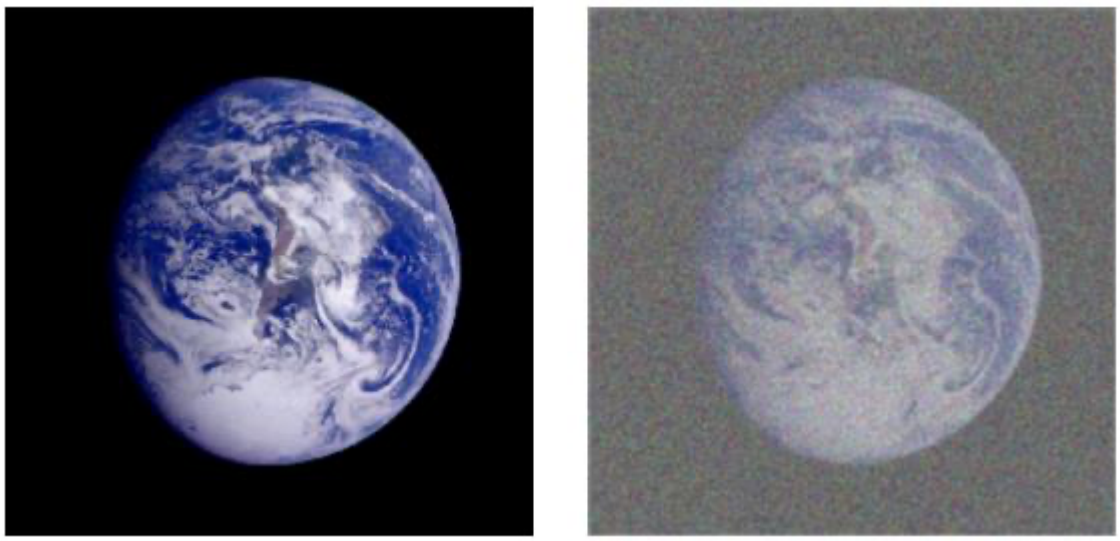


Ilustración 2- Ejemplo imagen con ruido gaussiano. Izquierda, imagen original. Derecha, ruido gaussiano con media en 0 y varianza 800

El segundo tipo de ruido surge debido a la técnica de imagen *single pixel* con *compressive sensing*. Este tipo de ruido es generado por la falta de información cuando se regenera la imagen. Este fenómeno es consecuencia de muestrear menos patrones que píxeles tiene la imagen que se pretende reconstruir. A continuación, presentamos una ilustración de una imagen generada mediante la técnica *single pixel imaging* con *compressive sensing*, basada en una simulación numérica.

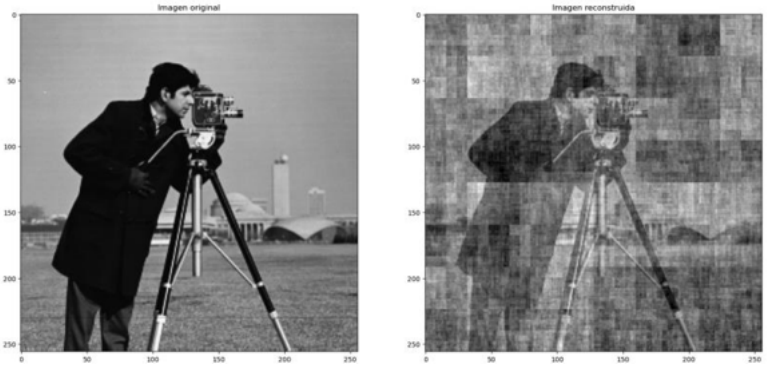


Ilustración 3 – Ejemplo de imagen con ruido debido al proceso de reconstrucción al utilizar SPI y CS. Izquierda, imagen original. Derecha, imagen con ruido.

2.1 Redes neuronales para reducción de ruido en imágenes

La tarea de minimizar el ruido en imágenes ya era un tema que se trataba antes de que se popularizaran los modelos de la IA para este propósito. En aquel entonces, predominaban técnicas más tradicionales basadas en filtros que se aplicaban en el dominio espacial o frecuencial de la imagen. En este apartado me gustaría ilustrar una comparación utilizando un enfoque práctico, poniendo en contraste el

filtrado de un mismo tipo de ruido utilizando métodos convencionales y posteriormente mediante el uso de redes neuronales.

En las siguientes imágenes se ha añadido un ruido del tipo “sal y pimienta”, y se pretende minimizarlo utilizando métodos de procesado de imagen convencionales. Un análisis más detallado puede encontrarse en el siguiente artículo [3].

Ilustración 4- Imagen original. Se trata de una imagen de una tomografía

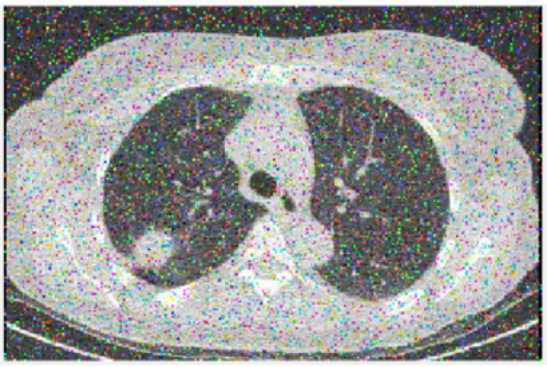


Ilustración 5 - Imagen original con ruido tipo "sal y pimienta" con un ratio del 10%

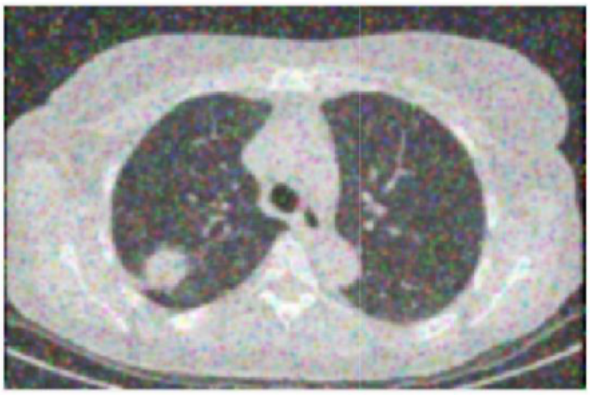


Ilustración 6 - Reducción de ruido utilizando un filtro gaussiano

A continuación, procedemos a explorar cómo se comportan las redes neuronales para solucionar este tipo de problemas. Y cómo, en muchos casos superan los métodos convencionales, como los descritos anteriormente. A modo de ejemplo, en la siguiente ilustración se muestra el caso de reducción de ruido con una red neuronal del tipo *Pulse Coupled Neuroal Network* (PCNN) [4].

Ilustración 7 - Ejemplo reducción de ruido del tipo "sal y pimienta" con redes neuronales. Izquierda, imagen original. Centro, imagen con ruido. Derecha, imagen con reducción de ruido usando redes neuronales.

Al analizar el resultado de manera cualitativa, es evidente que las capacidades de las redes neuronales en la reducción de ruido son muy prometedoras. Sin embargo, es necesario considerar que la efectividad de esta técnica dependerá de diversos factores, como la arquitectura de la red neuronal, el conjunto de datos de entrenamiento y la naturaleza del ruido de la imagen de entrada.

Tras hacer un estudio de las distintas topologías de redes neuronales empleadas en la reducción de ruido en imágenes, una de las tendencias es usar redes del tipo autoencoder convolucional y U-Net para esta aplicación. Aunque el funcionamiento de estas redes no es equivalente, ambas poseen características intrínsecas que las hacen idóneas para esta tarea:

- **Codificación y decodificación**: ambas arquitecturas cuentan con dos etapas principales: una primera de codificación que reduce la dimensión espacial, y una segunda de decodificación donde se restaura la imagen original.
- **Convoluciones**: Este proceso es fundamental cuando se trabaja con imágenes, ya que permite aprender patrones jerárquicos y comprender estructuras espaciales en los datos.
- **Flexibilidad y adaptabilidad**: Tanto los autoencoders como las U-Nets son arquitecturas flexibles. Pueden ser ajustadas y modificadas según las necesidades específicas de una tarea, permitiendo que se adapten a diferentes tipos y niveles de ruido

2.2 Trabajos relevantes en aplicaciones de reducción de ruido en imágenes obtenidas con técnicas de un solo píxel

En este apartado se explican algunos de los trabajos más relevantes en el ámbito del desarrollo de redes neuronales para mejora de imágenes obtenidas a través de técnicas de un solo píxel.

2.2.1 *Deep-learning-based ghost imaging*

En uno de los trabajos [5] se ha implementado la IA para mejorar la reconstrucción de imágenes utilizando técnicas de un solo píxel utilizando una red convolucional con tres capas densas, y el conjunto de datos de entrenamiento consta de imágenes con ruido gaussiano y ruido debido al submuestreo por la técnica de *compressive sensing.*

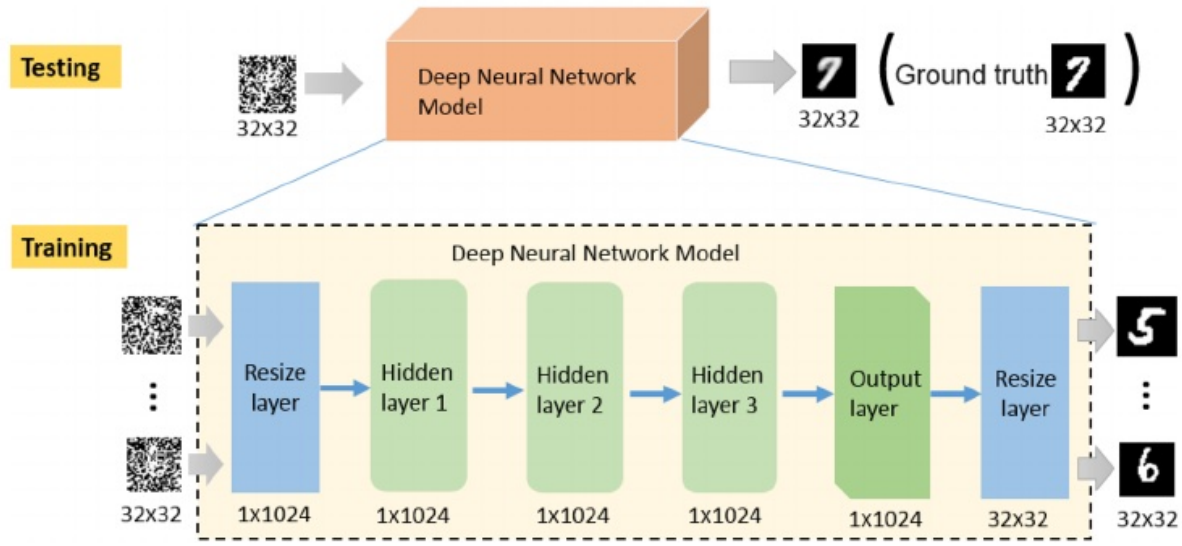


Ilustración 8 - En la parte superior, esquema de test. En la mitad inferior arquitectura de la red neuronal y flujo de entrenamiento

A continuación, se muestran los resultados de la reconstrucción de las imágenes sin utilizar la red neuronal, es decir, con CSGI (Compressive Sensing Ghost Imaging). Siendo β= 1 el caso en que se muestrean todos los patrones (máxima información), y β=0.1 muestreando solo el 10% de los patrones (10% de información de la imagen). Y la "w" indica el nivel de ruido gaussiano añadido, siendo w=1 mínimo, y w=50 máximo.

Posteriormente en la mitad inferior de la ilustración, se muestra el resultado después de procesar con la red neuronal ya entrenada, es decir, GIDL (Ghost Imaging Deep Learning).

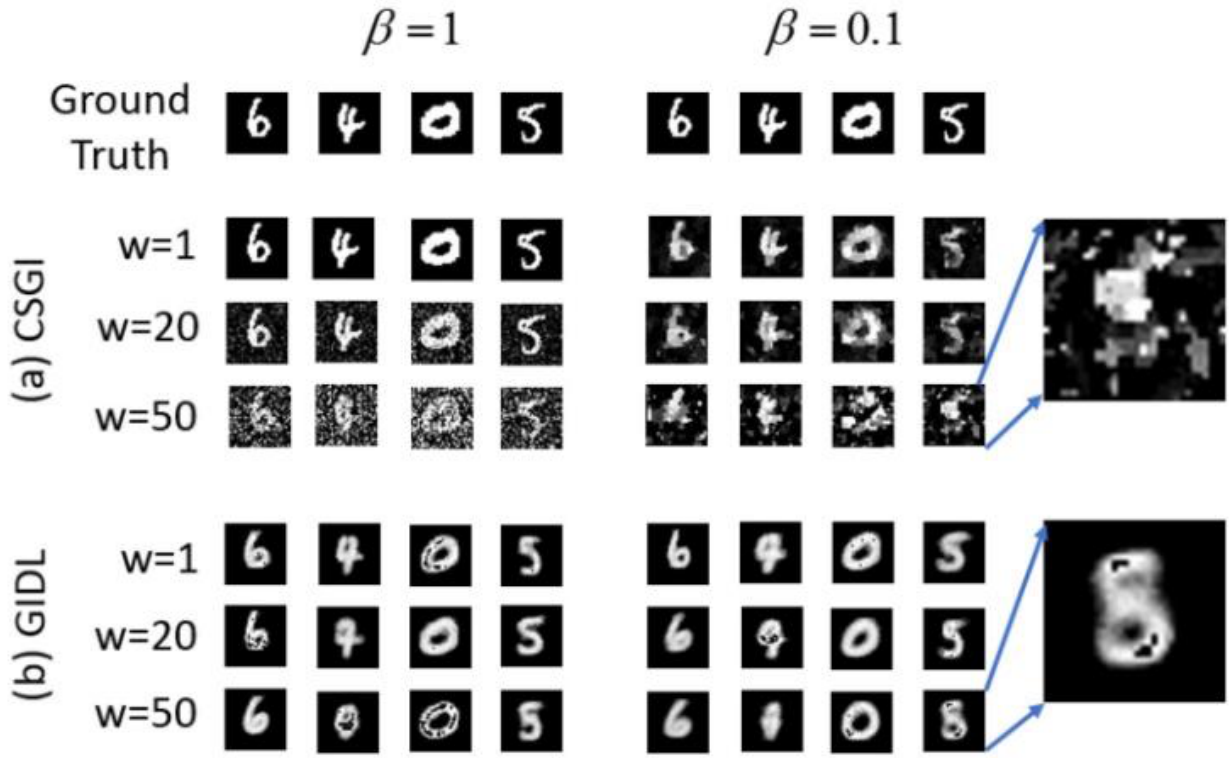


Ilustración 9 - Resultados experimento de reducción compressive sensing. Imágenes reconstruidas usando CSGI y GIDL con distintos niveles de ruido y ratio de muestras

Podemos observar que los resultados para las imágenes procesadas con CSGI pueden tolerar bajos niveles de ruido, pero para niveles más altos de ruido la imagen reconstruida se corrompe.

Al utilizar GIDL para reconstruir la imagen se obtiene mejores resultados en todos los niveles de ruido. Aunque la imagen final tiene unos bordes más suaves y menos definida.

En conclusión, la red neuronal se comporta como se espera. Pero en comparación con otros tipos de redes que veremos más adelante, una convolucional pura no es topología óptima para este tipo de problemas.

2.2.2 *Learning from simulation: An end-to-end deep-learning approach for computational ghost imaging*

En otro artículo [6] se utiliza una red neuronal más elaborada que está basada en ResNet y HoloNet, y es capaz de extraer características a diferentes niveles de detalle gracias a su estructura con varios caminos. Cabe mencionar que los datos de entrada en este caso no son las imágenes con ruido, sino que son los vectores de intensidad normalizados resultantes de la técnica de imagen con un sólo píxel, y la distribución espacial de los píxeles en la entrada no corresponde con la misma en la salida.

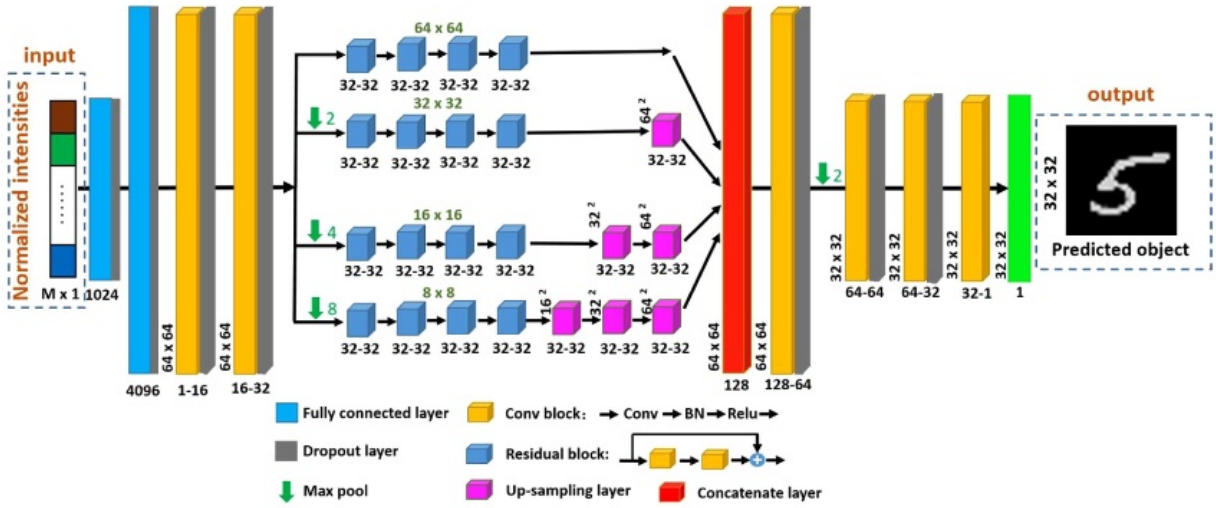


Ilustración 10 - Arquitectura propuesta en el artículo. Los datos de entrada son vectores de intensidades normalizadas y la salida son imágenes

En la siguiente imagen se observa que la imagen resultante para tasas de muestreo bajas (6.25%, 1.56% y 0.39%) con la técnica de GI y CSGI los resultados no son muy buenos. Pero cuando se utiliza la red neuronal entrenada la imagen resultante es muy similar a la de referencia. Aunque para tasas de muestreo muy bajas (0.39%), no hay información suficiente para reconstruir la imagen con una calidad que permita identificar la imagen de referencia.

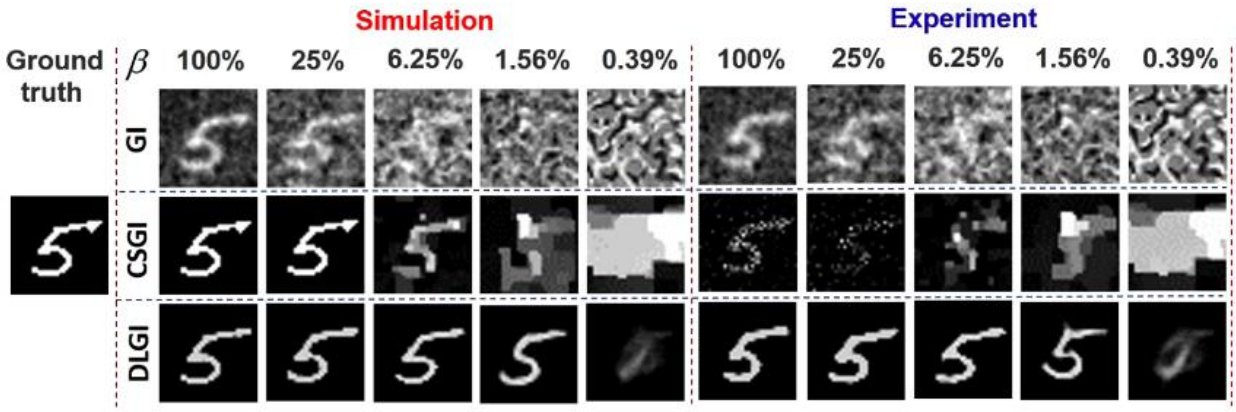


Ilustración 11 - Comparación de simulaciones y experimentos para distintas técnicas de reconstrucción (GI, CSGI, DLGI) y distintas ratios de muestreo.

La última parte de este artículo trata un aspecto muy interesante como es la capacidad de generalización de la red neuronal. Es decir, explora la posibilidad de usar el modelo entrenado con otros conjuntos de datos distintos a los de entrenamiento. Aunque funciona bien con imágenes del conjunto de datos de entrenamiento (*MNIST Handwritten Digits*), muestra limitaciones al tratar con imágenes no vistas previamente, especialmente a tasas de muestreo bajas. En la siguiente tabla se muestran las salidas

de la red neuronal al pasarle como dato de entrada la letra G, la letra I, y posteriormente dos líneas verticales y dos líneas horizontales.

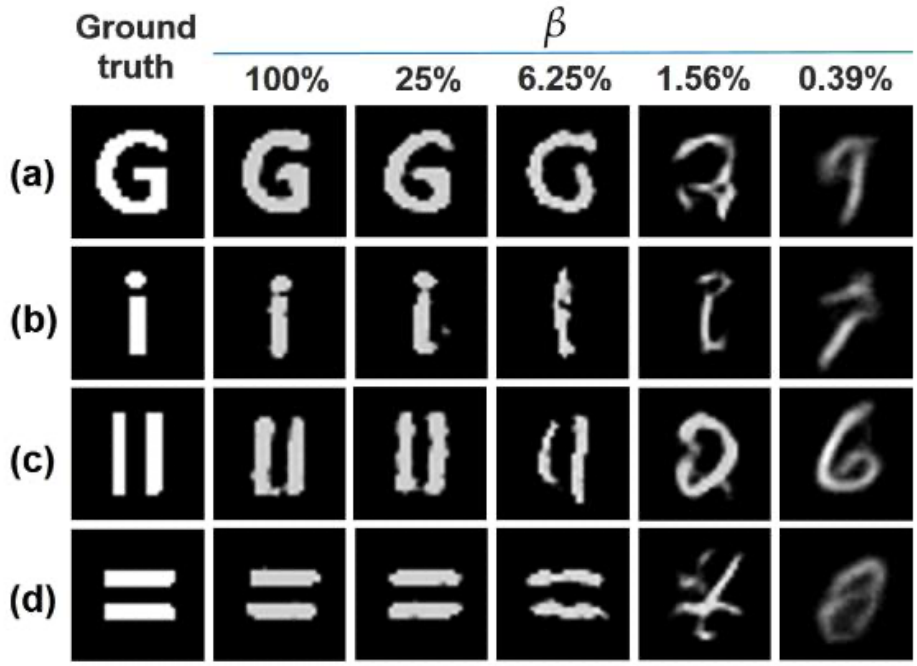


Ilustración 12 - Resultados de la prueba de generalización

Como podemos observar este tipo de red se comporta muy bien con datos similares a aquellos con los que ha sido entrenada. Pero cuando se utilizan datos de entrada distintos, y tasas de muestreo β menores que 6.25%, las imágenes reconstruidas son indistinguibles. Si prestamos atención a los resultados para valores de β=0.39% vemos que debido a la falta de información la red neuronal intenta completar la información con el espacio latente de su conjunto de datos de entrenamiento, por esta razón se pueden distinguir números en lugar de las correspondientes imágenes de entrada.

3 MARCO TEÓRICO

En este capítulo se tratarán los puntos más relevantes para proporcionar una base sólida para entender y contextualizar los experimentos y resultados presentados en las secciones posteriores de este trabajo.

3.1 Conceptos básicos de redes neuronales

La IA ha surgido como un campo de la informática que busca emular el razonamiento y las capacidades cognitivas humanas mediante sistemas computacionales. A medida que la IA se desarrollaba, nacieron subdisciplinas especializadas. Una de las más prominentes es el *machine learning* (aprendizaje

automático), que se centra en capacitar a las máquinas para aprender de los datos, permitiéndoles realizar predicciones o clasificaciones sin ser explícitamente programadas para ello.

Las redes neuronales son una pieza central en el dominio del machine learning. Inspiradas en la estructura y funcionalidad del cerebro humano, estas estructuras computacionales están diseñadas para reconocer y aprender patrones. Están compuestas por unidades llamadas "neuronas" o "perceptrones".

El perceptrón fue concebido en la década de 1950, representa la forma más elemental de una red neuronal. Es una neurona artificial diseñada para llevar a cabo clasificaciones binarias. Acepta múltiples entradas, las combina y, si la suma excede un determinado umbral, produce una salida. A pesar de su innovación, el perceptrón tiene sus limitaciones, particularmente su incapacidad para resolver problemas no lineales.

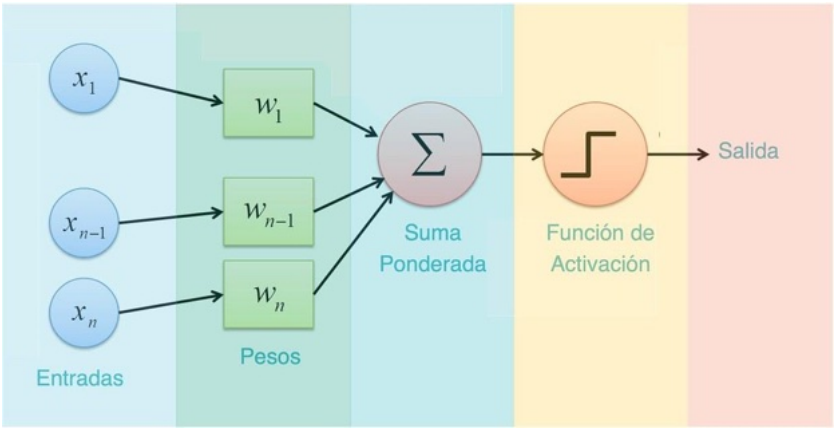


Ilustración 13 - Perceptrón simple y sus distintos componentes

Las redes neuronales multicapa surgieron como una solución a estas restricciones. Están formadas por múltiples capas de perceptrones: capas de entrada, una o más capas ocultas y una capa de salida. Esta estructura permite a la red aprender y modelar funciones más complejas.

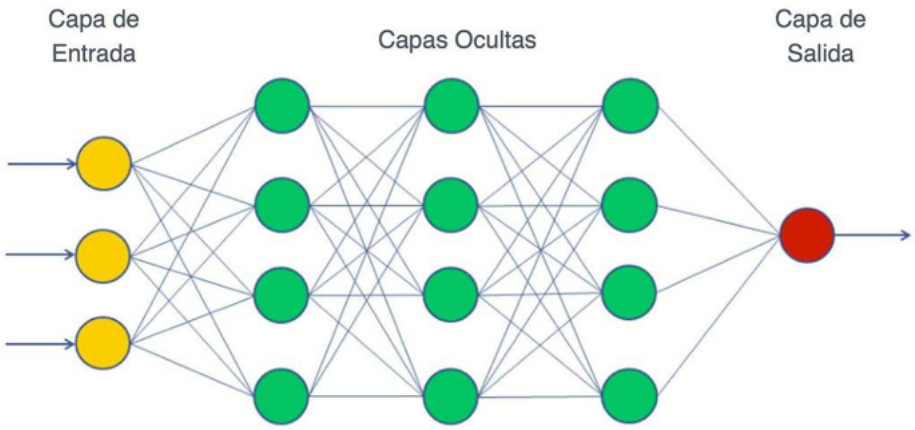


Ilustración 14 - Modelo de perceptrón multicapa

Concluyendo, las redes neuronales suelen estar formadas por distintas capas, y estas se pueden configurar de múltiples maneras para dar soluciones a problemas con técnicas de aprendizaje automático.

3.2 Extracción de características en imágenes: Redes convolucionales y correlación cruzada

El procesamiento de imágenes es un campo donde buscamos que las máquinas entiendan y procesen imágenes de manera eficiente. Una herramienta clave para esto son las redes neuronales convolucionales, conocidas como CNNs (del inglés, *convolutional neural networks*). Las CNNs están diseñadas para detectar patrones en imágenes. Gracias a sus capas convolucionales, pueden identificar características específicas en cualquier imagen, desde detalles minúsculos hasta patrones más amplios. Estas redes son ampliamente utilizadas en tareas como el reconocimiento de objetos y el reconocimiento facial.

Por otro lado, cuando queremos comparar imágenes o buscar similitudes entre ellas, utilizamos una técnica llamada correlación cruzada, o del inglés *cross-correlation*. La correlación cruzada es una medida de similitud de dos series en función del desplazamiento de una con respecto a la otra. También se conoce como producto punto deslizante. Se suele utilizar para buscar en una señal larga otra más corta conocida.

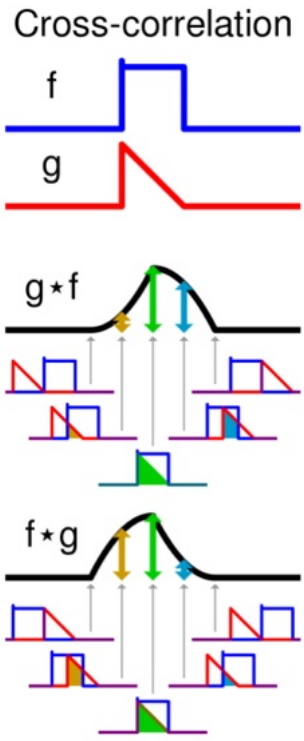


Ilustración 15 - Ejemplo visual de correlación cruzada de dos funciones "f" y "g" unidimensionales

La versión 2D de la técnica *cross-correlation* es especialmente útil con imágenes [7], ya que trabaja directamente sobre su estructura bidimensional. Esta técnica se asemeja a superponer dos imágenes y evaluar dónde y cuánto se parecen.

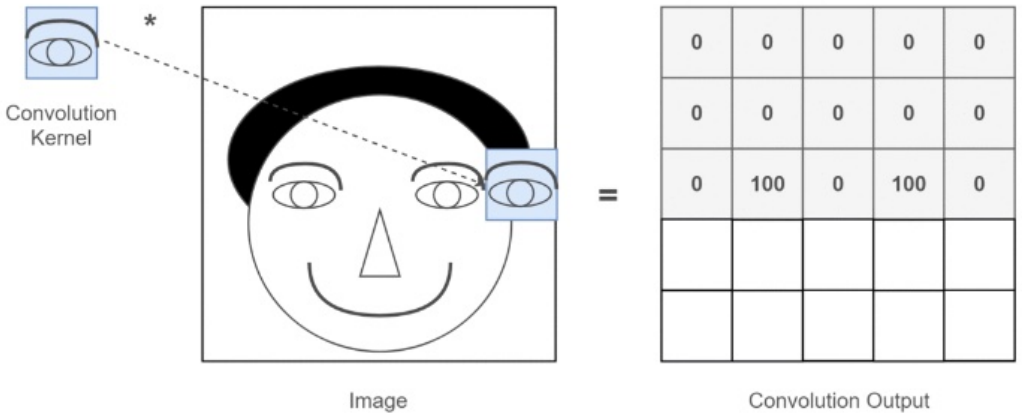


Ilustración 16 - Ejemplo de un kernel con la información de un ojo. Compara su información con la de la imagen, vemos dos casos donde la coincidencia es máxima, que corresponde con los dos ojos de la cara

La combinación de CNNs con la correlación cruzada 2D es particularmente potente. Mientras que las CNNs son expertas en aprender y reconocer patrones en imágenes, la correlación cruzada 2D nos permite confirmar y localizar la presencia de esos patrones en otras imágenes. Por ejemplo, si una CNN está entrenada para detectar ciertas características en imágenes médicas, la correlación cruzada 2D puede ser utilizada posteriormente para comparar nuevas imágenes con las ya conocidas y determinar similitudes o diferencias.

En resumen, la integración de CNNs con técnicas de correlación cruzada 2D brinda una herramienta eficaz para el análisis y comparación de imágenes, permitiendo una interpretación más precisa y detallada del contenido visual. Estas herramientas, juntas, son esenciales para avanzar en áreas como la visión por computadora y el diagnóstico médico basado en imágenes.

3.3 Imágenes de un solo píxel con luz estructurada

Las técnicas basadas en generación de imágenes de un solo píxel (SPI, del inglés *single-pixel imaging)* son una alternativa a la imagen convencional, donde se muestrea una escena con una secuencia de patrones de luz estructurada codificada en un modulador de luz espacial. La luz transmitida o reflejada por la escena será medida y cuantificada por un sensor sin estructura espacial, y posteriormente utilizada para recuperar la imagen de la escena aplicando operaciones matemáticas adecuadas.

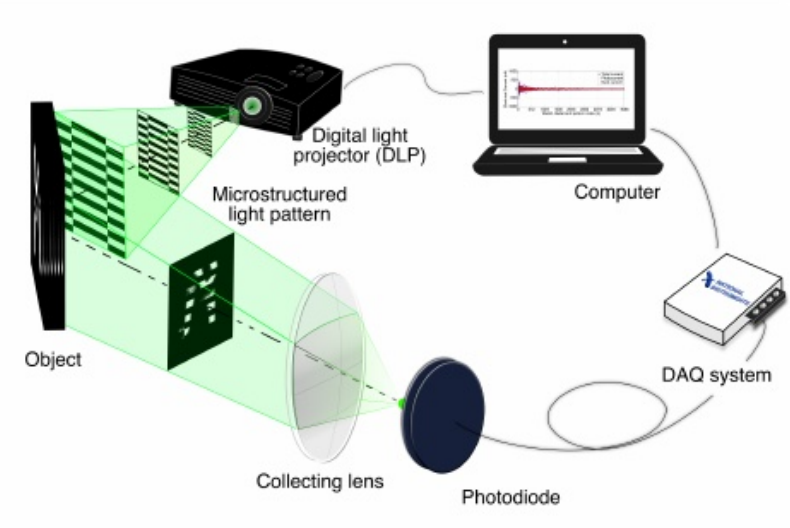


Ilustración 17 - Representación de una cámara de un solo píxel trabajando por reflexión

Normalmente cuando muestreamos una imagen con técnicas SPI, utilizamos tantos patrones como píxeles hay en la imagen para obtener una cantidad de información óptima. Hay distintas bases de patrones que se pueden utilizar para muestrear una imagen, en este trabajo se utilizará la base de funciones de Hadamard.

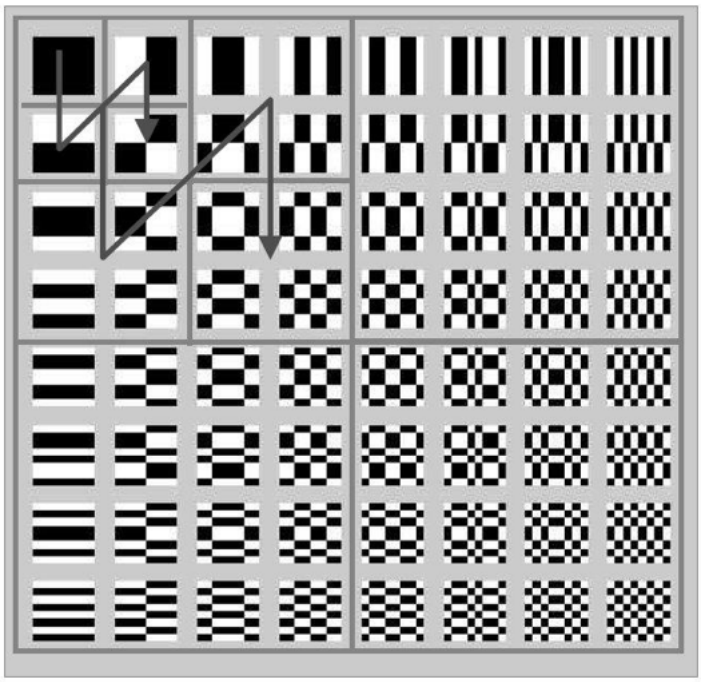


Figure 1: Patrones en base Hadamard para vectores de 8x8

Para capturar la intensidad reflejada por la escena se utiliza un fotosensor, comúnmente un fotodiodo. Éstos son dispositivos semiconductores que convierten las señales ópticas en señales eléctricas que posteriormente son digitalizadas y tratadas para la reconstrucción de la imagen.

Esta técnica de imagen es de especial interés para aplicaciones donde se requiera reducir la complejidad y el coste del sistema de detección. Por ejemplo, cuando se trabaja en rangos espectrales fuera del visible como el infrarrojo o terahercios, ya que en ese caso las cámaras convencionales con muchos píxeles tienen un coste muy elevado o son difíciles de construir. También son muy utilizadas para aplicaciones donde las cámaras convencionales pueden tener dificultades, como en condiciones de baja iluminación o en presencia de medios dispersivos. La resolución de la imagen resultante no está limitada por el número de píxeles del detector, sino por la resolución espacial de los patrones de modulación utilizados y la calidad del proceso de reconstrucción.

3.4 Muestreo comprimido

Los métodos de compresión de imágenes permiten reducir la cantidad de patrones utilizados para reconstruir una escena. Al utilizar sólo una parte del total de patrones se acelera el proceso de adquisición y se reduce la cantidad de datos que hay que almacenar.

La idea principal detrás de la técnica de *compressive sensing* (CS) es que muchas imágenes, aunque pueden tener alta resolución, a menudo cuentan con una estructura subyacente simplificada, o redundante. Por ello, con este tipo de técnicas se busca no utilizar durante el proceso de medida los patrones que no aporten información relevante para la reconstrucción de la imagen, de manera que se minimice la posible pérdida de calidad. Para una reconstrucción exitosa al utilizar CS es necesario tener algún conocimiento sobre la naturaleza de la imagen o emplear algoritmos iterativos de reconstrucción con regularizadores [8].

Esta técnica tiene aplicaciones en diversos campos, como el desarrollo de nuevas técnicas de imagen biomédica y cámaras hiperespectrales [9].

4 DISEÑO Y METODOLOGÍA

En esta sección del proyecto, se hablará sobre la parte más experimental del proyecto. A primera vista, puede parecer que el campo de las técnicas de imagen con un sólo píxel se trate de una aplicación muy nicho, específico y acotado. Sin embargo, hay muchas variables que hacen que el problema pueda cambiar significativamente, por ejemplo, dependiendo de la aplicación, del tipo de escenas que se quieran

reconstruir, del tipo de sensor, de la relación de compresión o del tipo de técnica utilizada para la reconstrucción de la imagen.

Por esta razón se ha diseñado un experimento poniendo énfasis en el testeo de las prestaciones de las redes neuronales para la reducción de ruido. El interés no se centra tanto en la naturaleza de los datos de entrada sino en cómo mejorar la calidad de los datos de la salida.

La metodología seguida para este experimento ha sido bastante similar a la utilizada durante el máster, y podríamos decir que se trata de un procedimiento bastante estándar. La estructura general del experimento es la siguiente:

1. Preprocesado del conjunto de datos
2. Simulación numérica de los experimentos ópticos
3. Entrenamiento modelos
4. Ajuste de hiperparámetros
5. Evaluación de resultados

4.1 Software y hardware utilizados durante el desarrollo

Antes de empezar con el grueso de la explicación del proyecto, es fundamente seleccionar las herramientas de software y hardware apropiadas. Es necesario elegir unas herramientas que nos permitan trabajar de manera eficiente, rápida y cómoda.

4.1.1 Software

Para realizar este proyecto se ha utilizado el siguiente conjunto de herramientas software. Elegimos *Python* como lenguaje de programación no sólo porque ya lo hemos usado en el máster, sino porque es un lenguaje de programación versátil y ampliamente usado en el ámbito de la IA. Su capacidad para ser ejecutado en diversas plataformas es un plus.

En cuanto al *framework* para IA se ha optado por *Keras*. Este *framework* es una API de alto nivel de *Tensorflow* que permite diseñar y entrenar modelos de aprendizaje automático fácilmente y con pocas líneas de código. Aunque no está optimizado para implementaciones de alto rendimiento, su uso para la etapa de prototipado es muy conveniente.

En lo que respecta al entorno de desarrollo se ha utilizado *Google Colab* pues es muy similar a *Jupyter Notebook*, y además tiene la capacidad de mostrar imágenes en la hoja de trabajo, y hace que sea una herramienta perfecta para documentar el trabajo realizado. Asimismo, también permite exportar la hoja de trabajo en formato *Jupyter Notebook* para hacer pruebas con el hardware local.

Ilustración 18 - Logotipos de software utilizado

4.1.2 Hardware

En el hardware utilizado se puede diferenciar dos etapas del proyecto bien diferenciadas donde se requerían distintos tipos de equipos. Una primera parte de diseño y desarrollo de los modelos y preprocesamiento de los datos. Y una segunda parte de entrenamiento de los modelos donde se ha requerido un equipo de mayores prestaciones. Las configuraciones de hardware utilizadas han sido las siguientes:

- **Portátil personal**: Este es el equipo que se utilizó durante las etapas iniciales
  - **CPU** : AMD Ryzen 7 4800H with Radeon Graphic.
  - **GPU**: Nvidia TU117M [GeForce GTX 1650 Mobile].
  - **RAM**: 16GB.
- **Google Colab:** A medida que el proyecto avanzaba y se empezaba a utilizar el conjunto de datos completo, la GPU del equipo portátil no fue suficiente y fue necesario migrar a las plataformas de computación que ofrece *Google Colab*. Inicialmente se utilizó la GPU gratuita proporcionada por *Google*. Sin embargo, debido a la cantidad de experimentos realizados y la duración de estos, pronto migramos a *Google Colab Pro*, puesto que brinda acceso a un hardware más potente, y durante más tiempo:
  - **GPU**: Nvidia Tesla P100.
  - **RAM**: 32GB.
  - **vCPU**: 2.

Esta combinación de hardware y la posibilidad de aumentar rápidamente la potencia de computación fue clave para un desarrollo eficiente.

4.2 Selección de la arquitectura de la red neuronal

La reducción de ruido es una aplicación muy estudiada en el campo de la inteligencia artificial, por sus numerosas aplicaciones en imágenes y señales. En el caso particular de nuestra aplicación se han buscado arquitecturas que funcionen especialmente bien para imágenes, que sean eficientes en cuanto a recursos utilizados ya que en el futuro la red se ejecutará en un dispositivo embebido con recursos limitados, probablemente basado en FPGA (*Field Programmable Gate Array*). No todas las arquitecturas

de redes neuronales se adaptan de manera óptima a este tipo de hardware. Y por último que tenga una baja latencia, ya que se desea hacer una cámara capaz de funcionar en tiempo real sin grandes esperas de procesamiento.

En primer lugar, se ha hecho un pequeño estudio cualitativo de los distintos tipos de redes más populares que se utilizan para la reducción de ruido en imágenes.

1. **Autoencoders Convolucionales**: Son modelos que buscan aprender representaciones eficientes de datos y su posterior reconstrucción. Su naturaleza crea un espacio latente que usa para la reconstrucción de la imagen, esto puede ser ventajoso para tareas de atenuación de ruido [10].

2. **DnCNN**: Como se destaca en este artículo [11], DnCNN es una red diseñada específicamente para disminuir la cantidad de ruido, demostrando ser efectiva en diversas condiciones de ruido.

3. **ResNets**: Las redes residuales o ResNets, aunque comúnmente asociadas con tareas de clasificación, han mostrado capacidad en la restauración de imágenes, incluida la reducción de ruido [12].

4. **U-Net**: La referencia [13] presenta U-Net, una arquitectura optimizada para la segmentación pero que ha demostrado ser versátil y efectiva también en tareas de reducción de ruido.

5. **GAN (Redes Generativas Antagónicas)**: Las redes GANs, particularmente las adaptadas para atenuación de ruido, tienen la capacidad de generar imágenes nítidas y realistas, aunque pueden ser más complejas en términos de entrenamiento. Véase esta referencia sobre un ejemplo de uso [14].

Dada la necesidad de equilibrar la eficiencia computacional con el rendimiento en el denoising, y considerando la futura implementación en una FPGA, hemos seleccionado dos arquitecturas que creemos que ofrecen el mejor compromiso: **Autoencoders Convolucionales** y **U-Net**. En los siguientes subcapítulos, profundizaremos en las razones detrás de esta elección y describiremos en detalle estas arquitecturas.

4.2.1 Autoencoder Convolucional

Los autoencoders constan de dos partes: el codificador y el decodificador. Durante el proceso de entrenamiento la etapa de codificación comprime las dimensiones de la imagen aprendiendo el conjunto de características, y va generando un espacio latente a partir de los datos de la entrada. Posteriormente el decodificador se entrena para reconstruir los datos de entrada en función de estas características.

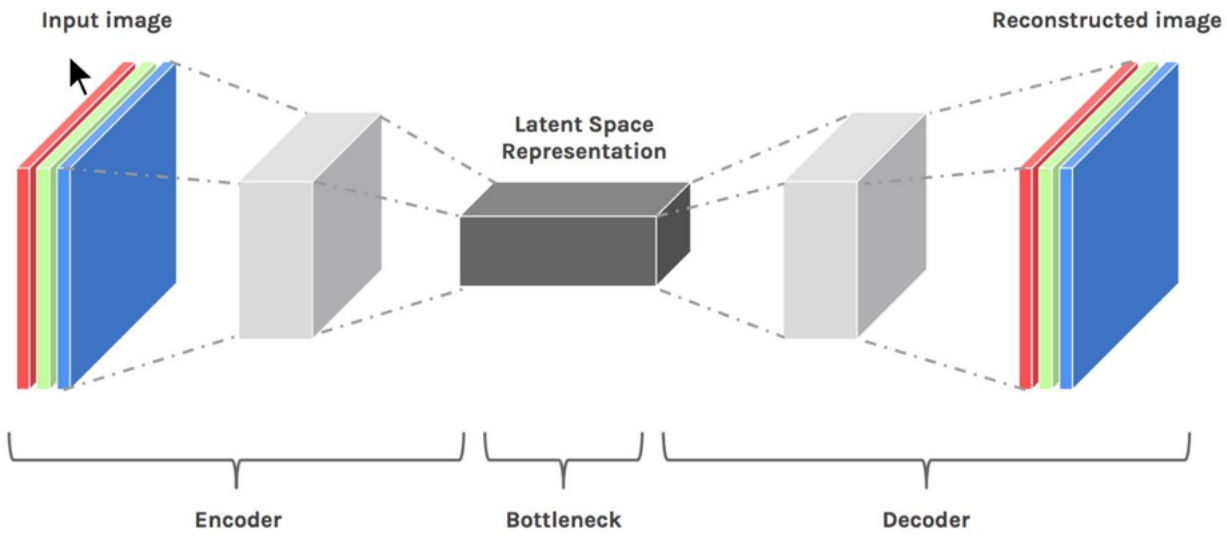


Ilustración 19 - Arquitectura de un Autoencoder. Está formado por un codificador y un decodificador

Una ventaja de los autoencoders es que no requieren de datos etiquetados para el entrenamiento, ya que estos no son supervisados. Existen distintos tipos de autoencoders y se utilizan para distintas tareas, por ejemplo:

• **Autoencoder convolucional:** este tipo es el utilizado en este proyecto. En este tipo, la salida del decodificador intenta copiar la entrada del codificador, para esto utiliza la información almacenada en el espacio latente. Este tipo de red es útil para eliminar ruido.

• **Autocodificadores variacionales:** este tipo crea un modelo generativo en el cual el espacio latente es modelado como una distribución probabilística. A través de este enfoque, pueden generar nuevas muestras que se asemejan a las entradas originales. Este tipo de red es especialmente útil para la detección de anomalías, ya que puede reconocer entradas que no se ajustan a la distribución aprendida.

• **Autocodificadores LSTM:** estos son utilizados especialmente para secuencias o series temporales. En este tipo, tanto el codificador como el decodificador utilizan celdas LSTM, las cuales son excelentes para aprender y generar patrones a lo largo del tiempo. Este tipo de red es ideal para aplicaciones de series temporales, permitiendo tanto la compresión de datos temporales como la generación de nuevas secuencias basadas en las aprendidas.

4.2.2 U-Net

Las arquitecturas U-Net se desarrollaron originalmente para segmentación de imágenes, aunque también se utiliza para otras aplicaciones como reducción de ruido. La estructura general de una U-Net se asemeja a la letra 'U', de ahí su nombre. Y esta consta de dos partes: una de contracción (codificador) y una de expansión (decodificador).

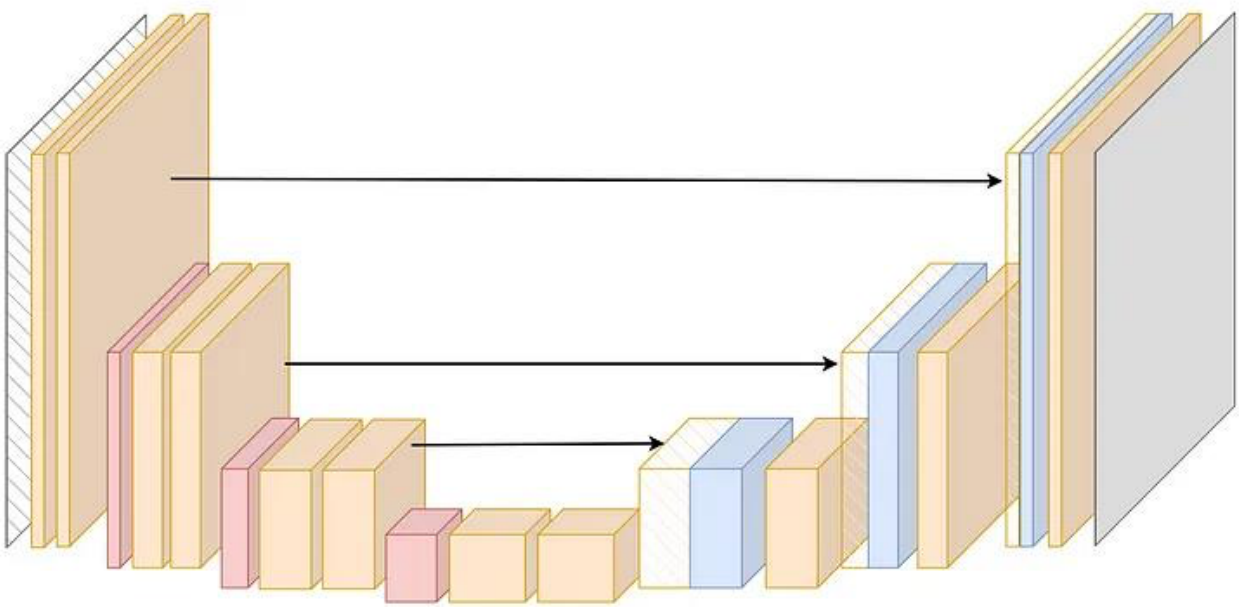


Ilustración 20 - Arquitectura de una red de tipo U-Net. Se caracteriza por etapa de contracción y expansión, así como de sus capas de conexión de salto

En la etapa de codificación, la información se condensa mediante una serie de capas convolucionales y operaciones de *pooling*, reduciendo así la dimensionalidad y capturando características esenciales de la imagen. Una vez que el codificador ha procesado la imagen, comienza la fase de decodificación. Aquí, la imagen codificada se expande de nuevo a su tamaño original. Sin embargo, lo que distingue a U-Net de otras arquitecturas es la inclusión de conexiones de salto entre las capas del codificador y el decodificador, permitiendo que la información de alta resolución del codificador fluya directamente hacia el decodificador.

Las ventajas de U-Net incluyen:

1. **Eficiencia en el Aprendizaje:** Gracias a las conexiones de salto, U-Net puede aprender con un menor número de imágenes en comparación con otras arquitecturas. Estas conexiones permiten que la red conserve información detallada de la imagen, aunque estas capas de salto requieren que haya una coherencia en la información espacial de la imagen a la salida respecto a la entrada.

2. **Especificidad en la Aplicación:** La capacidad de la red para conservar detalles sobre la distribución espacial de la información la hace útil tanto para aplicaciones de segmentación como para reducción de ruido en imágenes.
3. **Adaptabilidad:** Aunque el diseño original de U-Net se basa en capas convolucionales 2D, puede adaptarse para trabajar con 3D.

4.3 Selección de conjunto de datos

Dado que en este proyecto se desea estudiar las prestaciones de redes neuronales para la aplicación de reducción de ruido, los datos de entrenamiento serán dos conjuntos de datos distintos a los cuales se les ha añadido distintos niveles de ruido. Estos dos conjuntos de datos son bastantes diferentes entre sí dado que las imágenes que los componen son de distinta naturaleza. Esto es así porque se ha querido experimentar con el comportamiento de la red neuronal para diversos tipos de imágenes.

Los datasets de base utilizados son: MNIST Handwritten digits y CIFAR-10. A continuación, se realiza un resumen detallado de cada uno.

4.3.1 MNIST Handwritten digits

Este conjunto de datos está formado por imágenes de las cifras del 0 al 9 escritas a mano. Este sea posiblemente uno de los conjuntos de datos más utilizados cuando se empieza con tratamiento de imágenes. Este hecho resulta de ayuda pues es de frecuente uso en gran parte de la literatura que trata temas de reducción de ruido en imágenes.

Ilustración 21 - Extracto de muestra del conjunto de datos. Formado por 10 posibles dígitos escritos a mano

Las principales características de este conjunto de datos son las siguientes:

1. **Cantidad imágenes**: 60.000 entrenamiento y 10.000 test. Una cantidad suficiente de imágenes, que no requerirá que usemos técnicas de aumentación de datos.

2. **Dimensión**: 28x28 píxeles. Un tamaño reducido que agilizará el entrenamiento.

3. **Formato**: Escala de grises 8bits. Para el experimento de reducción de ruido minimizaremos la cantidad de información por eso es muy conveniente que sean imágenes monocromo.

4. **Imágenes etiquetadas para hacer clasificación**: Para nuestra aplicación nos centraremos en la imagen y omitiremos la información de la etiqueta.

5. **Distribución**: Clases bastante equilibradas. Aproximadamente, se dispone de la misma cantidad de elementos en cada clase. Esto es interesante para asegurar que hay suficientes datos para entrenar todas las clases.

6. **Sesgos muy marcados**: Sólo hay 10 posibles números y aunque tienen ciertas diferencias por estar escritos a mano, su estructura es bastante similar.

4.3.2 CIFAR-10

Este conjunto de datos está formado por imágenes naturales que pertenecen a 10 clases distintas. Estas clases son: avión, automóvil, pájaro, gato, reno, perro, rana, caballo, barco y camión. Este conjunto de datos también es muy popular, este hecho resulta de ayuda pues es de frecuente uso en publicaciones que tratan temas de reducción de ruido en imágenes.

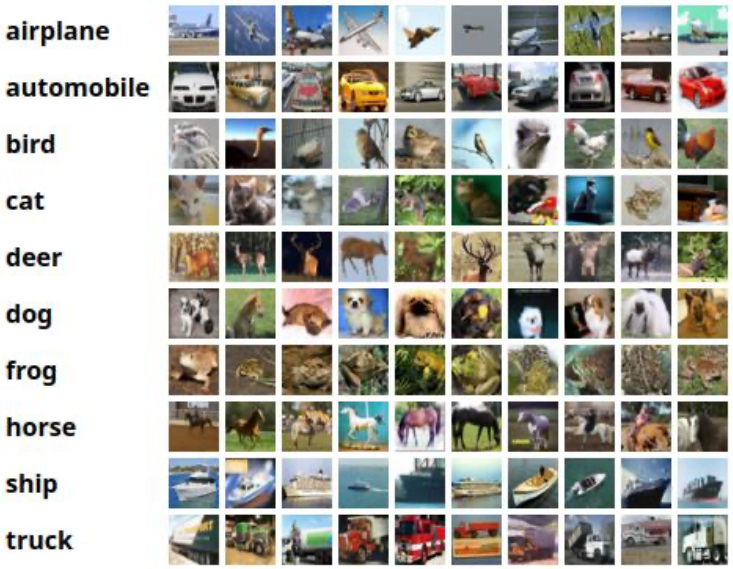


Ilustración 22 - Extracto de muestra del conjunto de datos. Formado por las 10 clases de escenas.

Las principales características de este conjunto de datos son las siguientes:

1. **Cantidad imágenes**: 50.000 entrenamiento y 10.000 test. Hay muchas imágenes por lo que no se requerirá el uso de técnicas de aumentación de datos.

2. **Dimensión**: 32x32. Es potencia de dos y bastante similar a las dimensiones de los números MNIST. Esto será conveniente porque la transformada de Fast Walsh-Hadamard necesita que las dimensiones sean potencia de 2.

3. **Formato**: Originalmente está en RGB, pero la pasaremos a escala de grises para comparar resultados con MNIST Handwritten digits, y minimizar la cantidad de información a procesar.

4. **imágenes etiquetadas para hacer clasificación**: Para nuestra aplicación nos centraremos en la imagen y omitiremos esta información.

5. **Distribución**: Clases equilibradas. Hay 10 clases y 10% de imágenes de cada clase.

6. **Sesgos menos marcados**: Al tratarse de imágenes naturales los sesgos están mucho menos marcados que con los números.

La elección de estos dos conjuntos de datos nos permite explorar cómo se comportan las redes neuronales en escenarios con diferentes niveles de complejidad, desde dígitos manuscritos simples hasta imágenes naturales más complejas.

4.4 Preprocesado

El preprocesado es crucial para asegurar un correcto entrenamiento de la red neuronal. En esta fase se han realizado las modificaciones necesarias para adaptar los datos de entrenamiento y salida que nos permitan entrenar la red para el propósito deseado.

El flujo de las distintas etapas del preprocesado aplicado son las siguientes:

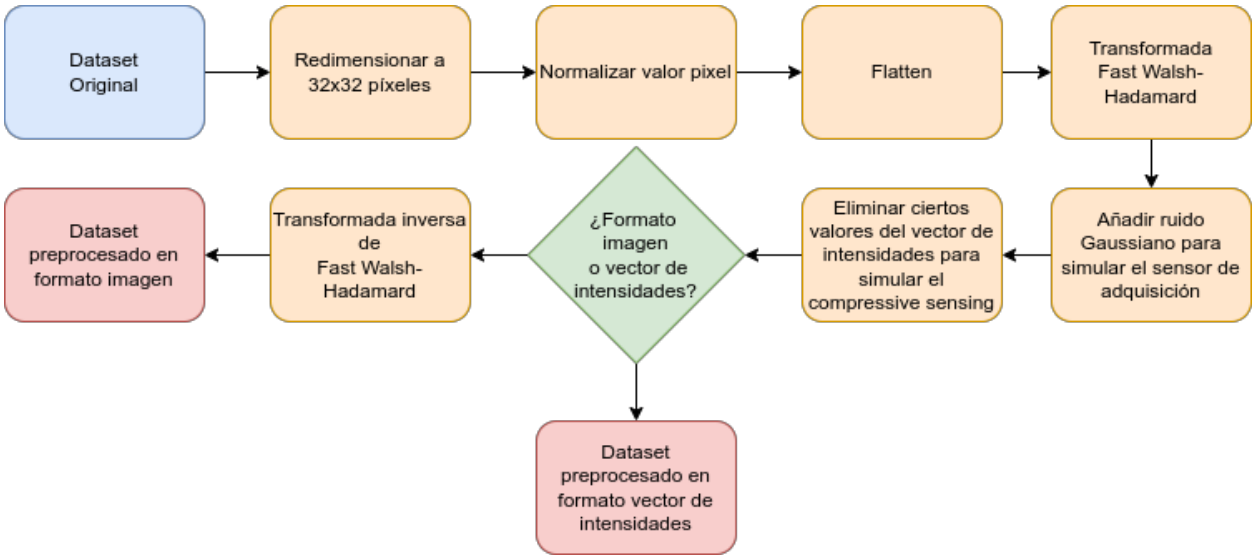


Ilustración 23 - Flujo de las etapas del preprocesado de los conjuntos de datos

El número total de conjuntos de datos que se han preparado han sido cuatro, que se corresponden con cada uno de los experimentos:

- Conjunto de datos MNIST Handwritten digits con ruido por *compressive sensing* en formato imagen
- Conjunto de datos MNIST Handwritten digits con ruido por *compressive sensing* en formato vector de intensidades
- Conjunto de datos CIFAR-10 con ruido por *compressive sensing* en formato imagen
- Conjunto de datos CIFAR-10 con ruido por *compressive sensing* en formato vector de intensidades

4.4.1 Técnica de *compressive sensing* en la simulación numérica

Como parte de la simulación numérica de la técnica de un sólo píxel usando la técnica de *compressive sensing*, algunos de los valores del vector de intensidades se descartarán, sin realizar ningún tipo de optimización adicional en el proceso de selección de los patrones o en el proceso de reconstrucción de la imagen típicos de las técnicas de CS. Cabe recordar que uno de los objetivos de este proyecto es evaluar la capacidad de la red neuronal para filtrar este tipo de ruido.

El vector de intensidades es una lista de valores, y tras hacer la transformada de Fast Walsh-Hadamard hay tantos coeficientes como píxeles tiene la imagen. A continuación, vamos a ver cualitativamente cual es la diferencia entre distintos casos: descartando los coeficientes de índices más altos del vector de intensidades, los de índices más bajos o aleatoriamente. El índice del vector está relacionado con la frecuencia del patrón de Hadamard.

A continuación, se muestra un ejemplo práctico con la imagen *camera man* para mostrar cualitativamente los resultados de eliminar los coeficientes del vector de intensidades de una imagen, y las consecuencias en la reconstrucción de la imagen.

En el primer caso descartamos los valores que corresponden a los índices más altos:

Ilustración 24 - Relación de muestreo 100%

Ilustración 25 - Relación muestreo 99%

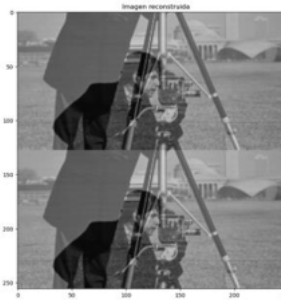


Ilustración 26 - Relación de muestreo 80%

Ilustración 27 - Relación de muestreo 50%

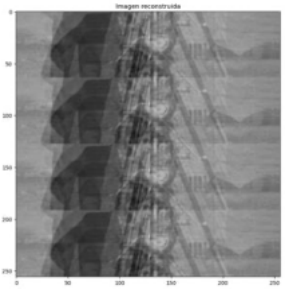


Ilustración 28 - Relación de muestreo 20%

En el segundo caso se han descartado los valores que corresponden a los índices más bajos:


Ilustración 29 - Relación de muestreo 100%


Ilustración 30 - Relación de muestreo 99%

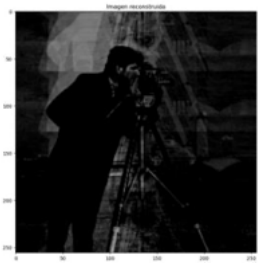



Ilustración 31 - Relación de muestreo 80%

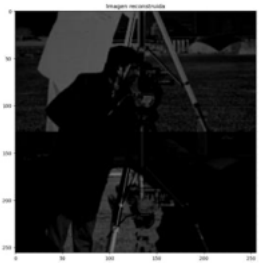



Ilustración 32 - Relación de muestreo 50%

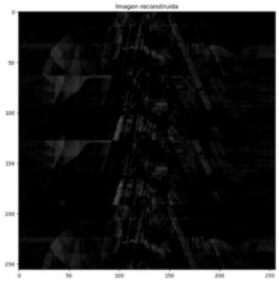



Ilustración 33 - Relación de muestreo 20%

En el tercer caso descartamos los valores del vector de intensidades de manera aleatoria:

Ilustración 34 - Relación de muestreo 100%

Ilustración 35 - Relación de muestreo 99%

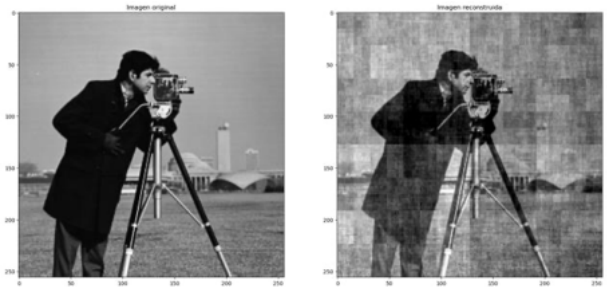


Ilustración 36 - Relación de muestreo 80%

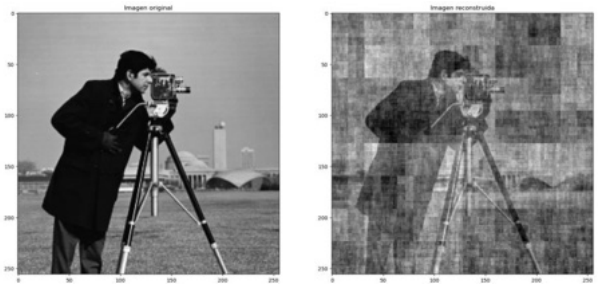


Ilustración 37 - Relación de muestreo 50%

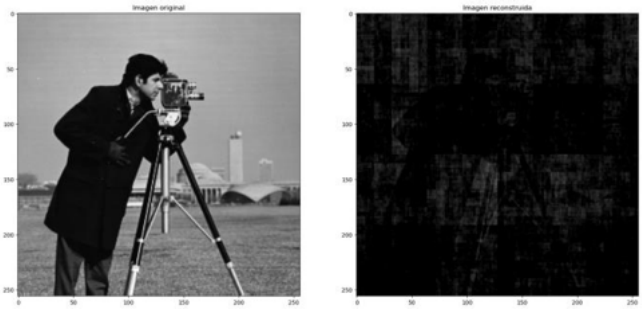


Ilustración 38 - Relación de muestreo 20%

Al final se ha elegido un ruido generado por el descarte aleatorio de patrones de Hadamard, de modo que el ruido sea más impredecible.

Como se puede apreciar el ruido generado por este tipo de técnicas es bastante diferente al ruido gaussiano o al sal y pimienta. Aunque hablamos de ruido debido a la técnica *compressive sensing*, gran parte de ese "ruido" es en realidad una pérdida que ocurre durante la reconstrucción de la imagen a partir del vector de intensidades incompleto.

Al entrenar la red con estas imágenes, se espera que sea capaz de recuperar la información perdida y proporcionar reconstrucciones de calidad, ya que sin la técnica de reducción de ruido consideraríamos esta información corrompida.

4.5 Descripción de las redes neuronales utilizadas

El paso posterior a la preparación del conjunto de datos es decidir los diseños de las redes neuronales con los que se realizarán los entrenamientos. Se pretende usar arquitecturas de tipo Autoencoder Convolucional y U-Net. Para cada una de estas arquitecturas se han preparado cuatro modelos con distintos hiperparámetros y valores de normalización. A continuación, se muestran las cuatro redes utilizadas para el caso del autoencoder convolucional, en el caso de los modelos basados en U-Net se ha procedido de manera análoga y se han planteado cuatro iteraciones de menos a más complejidad. La descripción detallada de cada una de las redes neuronales utilizadas se encuentra en el capítulo de anexos.

- **Iteración 1: Autoencoder reducido, sin normalización**. Esta red es la de menor capacidad y carece de capas de normalización. Se utilizará para comprobar cómo se desempeña una red sin características adicionales.

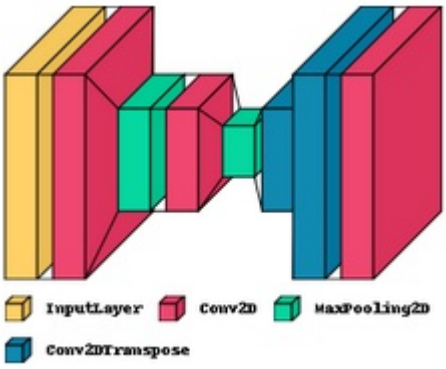


Ilustración 39 - Representación de la red neuronal. Iteración 1

- **Iteración 2: Autoencoder de capas profundas, sin normalización.** Esta red tiene más capas y, por lo tanto, una mayor capacidad para aprender características complejas. Sin embargo, sigue sin tener normalización, lo que puede llevar a problemas como el sobreajuste.

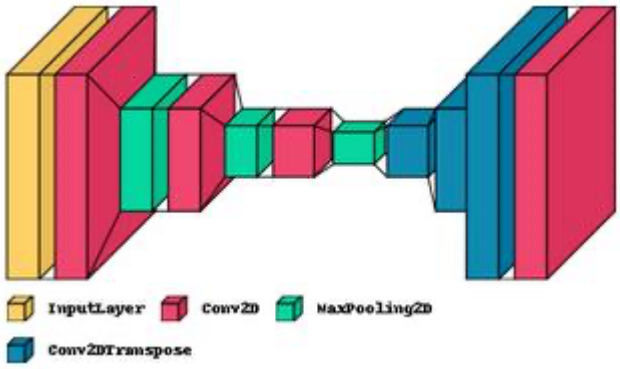


Ilustración 40 - Representación de la red neuronal. Iteración 2

- **Iteración 3: Autoencoder de capas profundas, con normalización (***dropout* **0.3 y *batch normalization*).** Además de estar constituida por muchas capas densas como la anterior, incorpora técnicas de normalización como *dropout* y *batch normalization* para prevenir el sobreajuste. El *dropout* de 0.3 indica que el 30% de las neuronas se "apagan" durante el entrenamiento en cada paso.

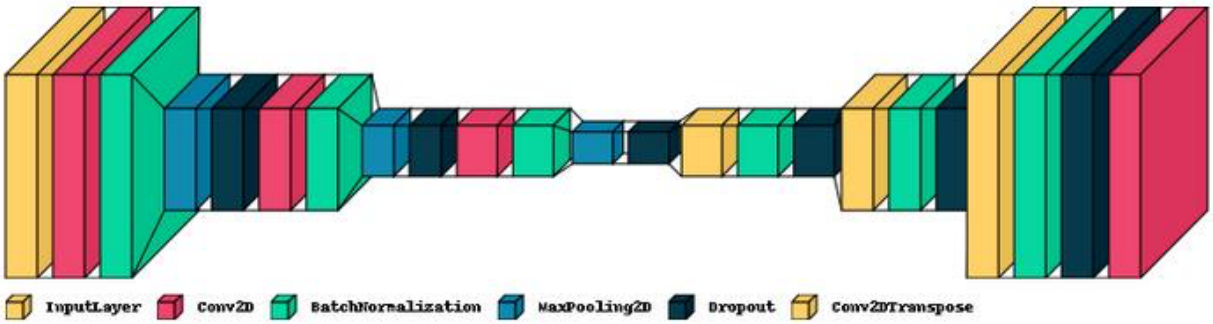


Ilustración 41 - Representación de la red neuronal. Iteración 3

- **Iteración 4: Autoencoder de capas profundas, con normalización (*dropout* 0.1 y *batch normalization*).** Similar a la red anterior, pero con la diferencia de que tiene un *dropout* más bajo, de 0.1. Esto permite más flujo de información a través de la red.

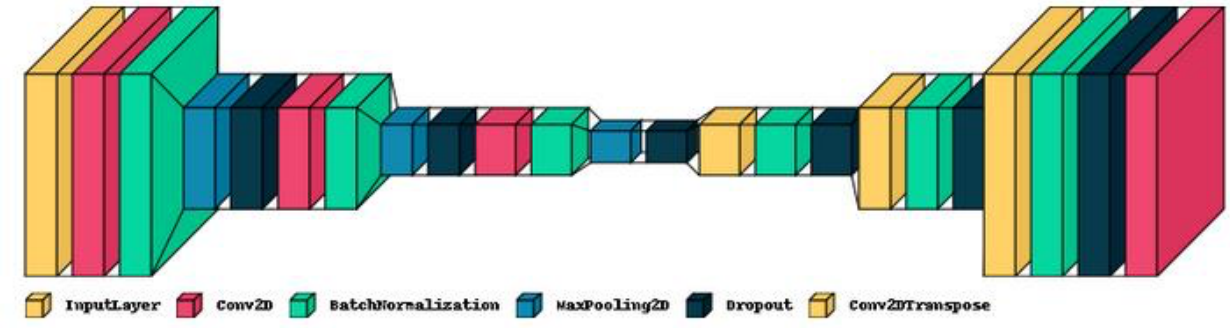


Ilustración 42 - Representación de la red neuronal. Iteración 4

4.6 Descripción del entrenamiento

Encontrar el modelo que dé una solución óptima puede requerir evaluar distintos parámetros en las redes neuronales, como por ejemplo el conjunto de datos de entrenamiento, los hiperparámetros o la arquitectura utilizada. Por esta razón, se han entrenado distintas configuraciones para encontrar la mejor solución para la tarea de reducción de ruido.

Se ha experimentado con un total de 8 combinaciones diferentes por conjunto de datos, lo que proporciona un análisis bastante amplio y permite observar cómo los distintos parámetros afectan al rendimiento. Por cada conjunto de datos se han evaluado las dos arquitecturas objeto de estudio: autoencoder convolucionales y U-Net. Asimismo, también se han probado varios hiperparámetros.

Sin embargo, para cada una de estas combinaciones la metodología de entrenamiento ha sido la misma para cada uno los cuatro conjuntos de datos. Ésta ha consistido en dar como dato de entrada las imágenes o los vectores de intensidades con ruido, y a la salida poner la imagen o el vector de intensidades sin ruido. De esta manera se fuerza a la red neuronal a reconstruir la imagen sin ruido.

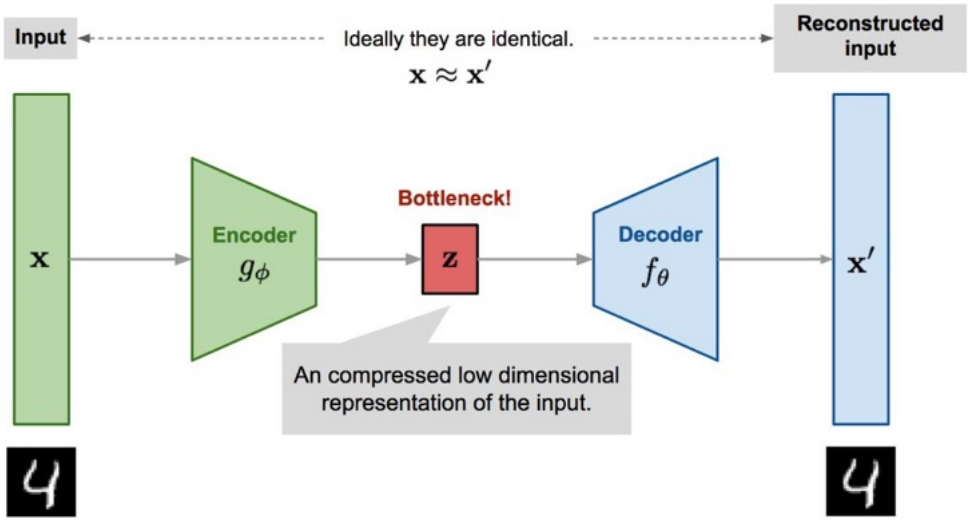


Ilustración 43 - Arquitectura Autoencoder en la fase de entrenamiento la entrada y la salida deben ser idealmente iguales

4.7 Métricas en la evaluación de resultados

Para evaluar cuantitativamente los resultados se ha optado por dos métricas que se suelen utilizar en aplicaciones donde se compara una imagen con la referencia, estas métricas son: PSNR y el SSIM.

La PSNR (del inglés, peak signal-to-noise ratio) es una métrica que se refiere a la relación entre el valor máximo posible de una señal(potencia) y la potencia del ruido que afecta a la calidad de su representación. Esta métrica se mide en una escala logarítmica de decibelios.

La PSNR se puede calcular como:

$$PSNR = 20 \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right)$$

Donde la MSE se puede calcular de la siguiente manera:

$$MSE = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} \|f(i,j) - g(i,j)\|^2$$

El otro indicador utilizado para evaluar la calidad de las imágenes predichas por la red neuronal es el SSIM: Structural Similarity Index. Es un método utilizado habitualmente para medir la similitud entre dos imágenes. En este indicador se mide la calidad de la imagen utilizando otra imagen sin distorsión como referencia.

Se puede calcular como sigue:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

5 RESULTADOS

En este apartado se muestran y se analizan los resultados obtenidos para cada uno de los experimentos realizados. Se han diferenciado dos enfoques para la valoración de los resultados, un primer enfoque donde se evalúan cualitativamente de los resultados observando los resultados de salida y comparando con la imagen esperada, y posteriormente una evaluación cuantitativa comparando los indicadores de PSNR y SSIM obtenidos en cada una de las iteraciones de las redes neuronales.

5.1 Evaluación cualitativa

Para la evaluación cualitativa de los resultados se han elegido diez muestras al azar para cada uno de los experimentos, y estas están organizadas en tres filas de imágenes. Las filas que se muestran en las imágenes corresponden a:

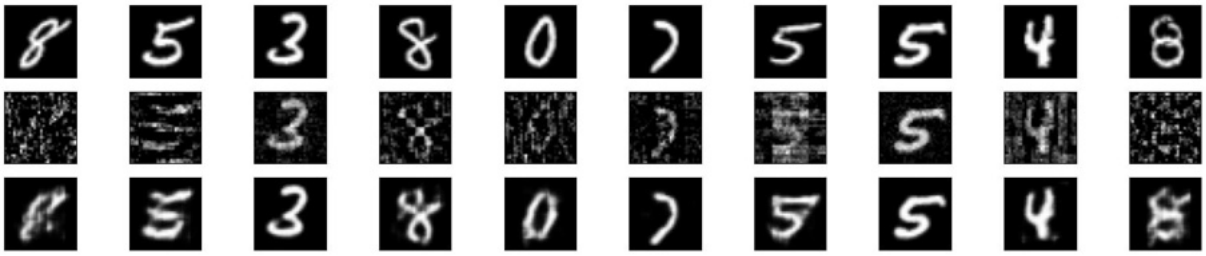
- Fila superior: Es la imagen de referencia sin ruido.
- Fila central: Son los datos de entrada a la red neuronal. Estos datos tienen un ruido añadido, y pueden estar o bien en formato imagen o bien en formato vector de intensidades.
- Fila inferior: Es la imagen de salida de la red neuronal, es decir, la predicción de la imagen. Si se eliminara el ruido de entrada por completo, la imagen de la fila inferior y la imagen de la fila superior deberían ser idénticas.

Los resultados se muestran en dos grupos según el tipo de arquitectura de la red neuronal. En primer lugar, se muestran los resultados obtenidos con el autoencoder convolucional, y posteriormente los resultados con la red U-Net. Adicionalmente, para cada tipo de arquitectura se evalúan los distintos conjuntos de datos y variaciones de hiperparámetros de las redes entrenadas.

5.1.1 Autoencoder convolucional

5.1.1.1 MNIST handwritten digits – Formato imagen

- Iteración 1: Autoencoder reducido, sin normalización

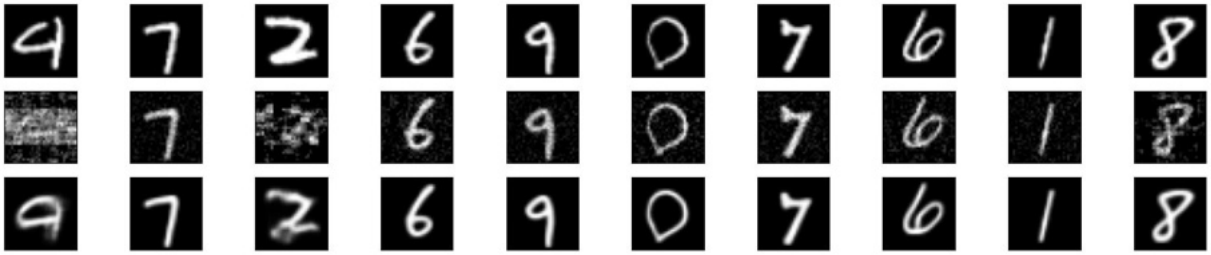


Para imágenes con bajos niveles de ruido la reconstrucción es satisfactoria, pero para imágenes con niveles altos de ruido, como por ejemplo la primera columna, la reconstrucción es deficiente.

- Iteración 2: Autoencoder de capas profundas, sin normalización

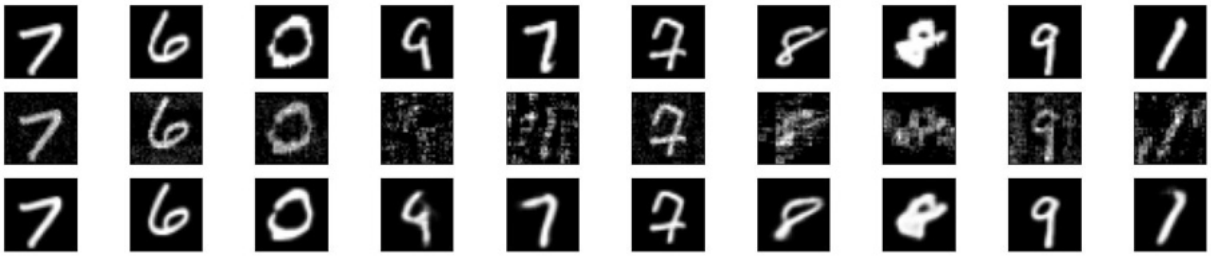
La reconstrucción ha mejorado bastante en general al aumentar la cantidad de capas densas. No obstante, los detalles en imágenes con poca información los bordes no están muy definidos.

- Iteración 3: Autoencoder de capas profundas, con normalización (dropout 0.3 y batch normalization)



La reconstrucción ha mejorado ligeramente respecto a la iteración anterior.

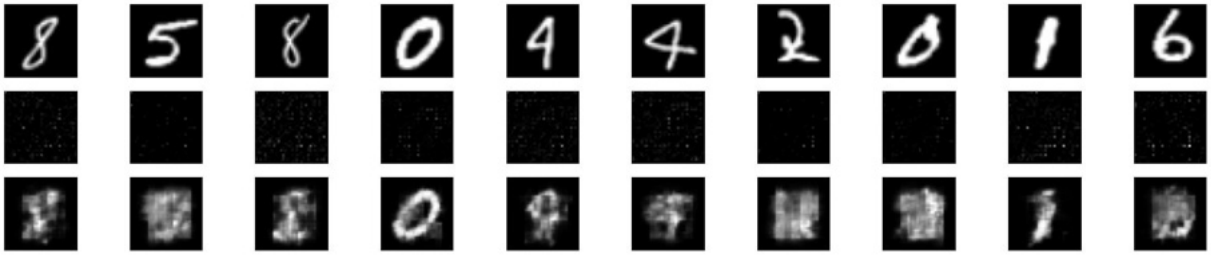
- Iteración 4: Autoencoder de capas profundas, con normalización (dropout 0.1 y batch normalization)



Visualmente los resultados son similares a la iteración 3. En general podemos decir que la reconstrucción de las muestras elegidas es aceptable en todos los casos.

5.1.1.2 MNIST handwritten digits – Formato vector de intensidades

- Iteración 1: Autoencoder reducido, sin normalización



La red neuronal no es capaz de hacer una reconstrucción correcta de los dígitos, aunque sí que está aprendiendo a identificar las zonas oscuras de los dígitos.

- Iteración 2: Autoencoder de capas profundas, sin normalización

Comparado con la iteración anterior, aumentar el tamaño de las capas densas ha mejorado la calidad de la reconstrucción. Aunque todavía tiene problemas al reconstruir algunos dígitos.

- Iteración 3: Autoencoder de capas profundas, con normalización (dropout 0.3 y batch normalization)

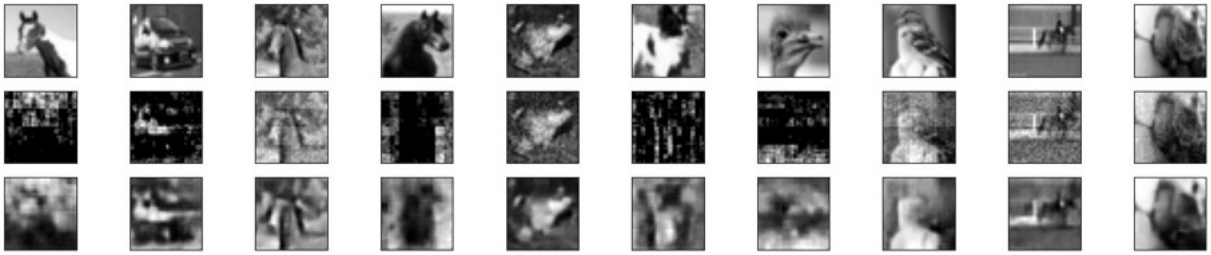
La reconstrucción está más definida que en la iteración anterior, pero todavía queda margen de mejora.

- Iteración 4: Autoencoder de capas profundas, con normalización (dropout 0.1 y batch normalization)

Cualitativamente los resultados son bastante similares a la iteración anterior.

5.1.1.3 CIFAR 10– Formato imagen

- Iteración 1: Autoencoder reducido, sin normalización



Con imágenes naturales, la tarea de reconstrucción se vuelve considerablemente más compleja. En general, la red sólo logra una reconstrucción mínima en presencia de niveles bajos de ruido.

- Iteración 2: Autoencoder de capas profundas, sin normalización

Al incorporar más capas densas, la red muestra un desempeño superior al de la iteración previa. Sin embargo, aún enfrenta dificultados en la reconstrucción con niveles medios o altos de ruido.

- Iteración 3: Autoencoder de capas profundas, con normalización (dropout 0.3 y batch normalization)

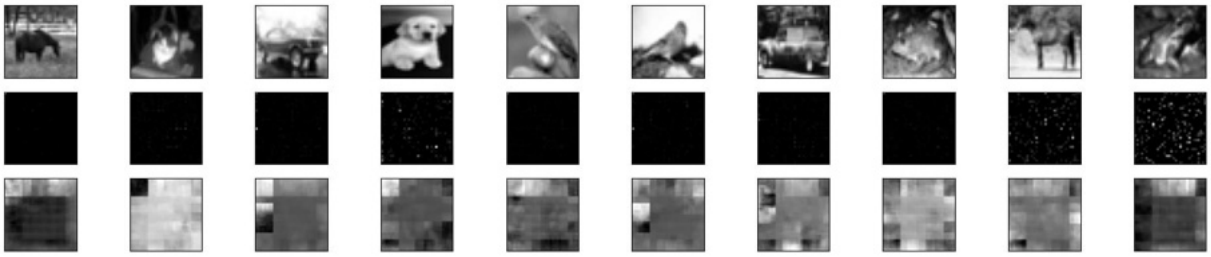
La calidad de reconstrucción mejora significativamente con la adición de capas de normalización. No obstante, la red todavía enfrenta desafíos al definir con precisión los bordes de las escenas.

- Iteración 4: Autoencoder de capas profundas, con normalización (dropout 0.1 y batch normalization)

Se observa que presenta un rendimiento ligeramente superior que el de la iteración anterior. Sin embargo, esto podría deberse a que las imágenes seleccionadas al azar cuentan con niveles de ruido más bajos.

5.1.1.4 CIFAR 10 – Formato vector de intensidades

- Iteración 1: Autoencoder reducido, sin normalización

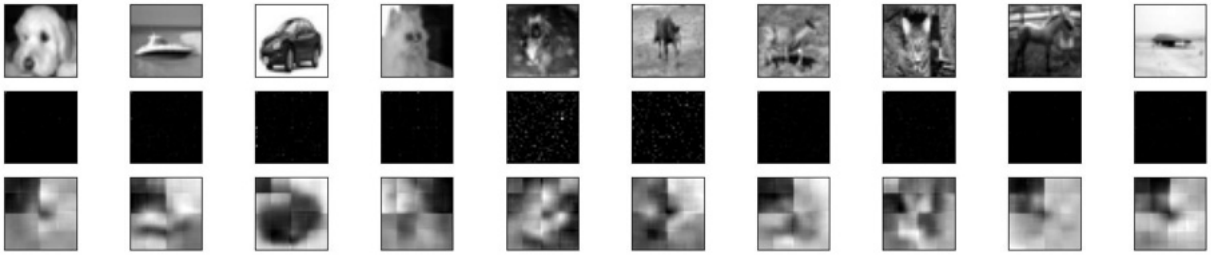


Debido a su tamaño reducido y dada la complejidad de las imágenes, la red no puede realizar reconstrucciones adecuadas a partir del vector de intensidades.

- Iteración 2: Autoencoder de capas profundas, sin normalización

Aunque muestra un rendimiento ligeramente mejor que la iteración anterior, todavía no logra realizar reconstrucciones de calidad aceptable.

- Iteración 3: Autoencoder de capas profundas, con normalización (dropout 0.3 y batch normalization)



El desempeño parece ser menos óptimo en comparación con la iteración anterior. Las capas de dropout parecen haber obstaculizado un aprendizaje adecuado.

- Iteración 4: Autoencoder de capas profundas, con normalización (dropout 0.1 y batch normalization)

Con una tasa de dropout reducida, presenta un rendimiento ligeramente mejor que en la iteración 3.

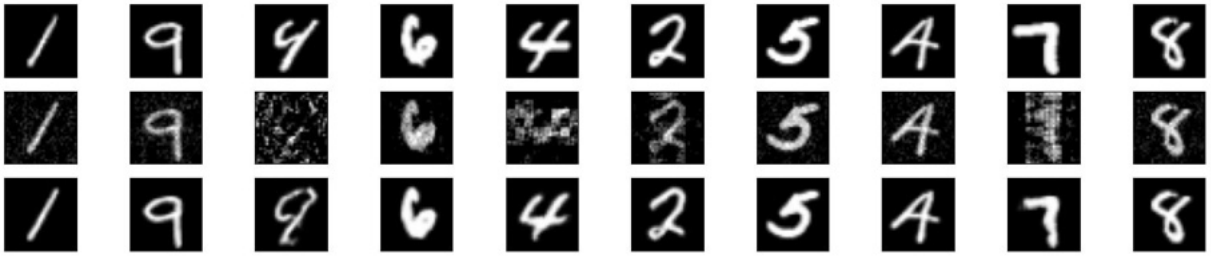
5.1.2 U-Net

5.1.2.1 MNIST handwritten digits – Formato imagen

- Iteración 1: U-Net reducida, sin normalización

En general, la U-Net de tamaño reducido logra una reconstrucción razonablemente buena en la mayoría de los casos, salvo en algunas imágenes con niveles de ruido más elevados.

- Iteración 2: U-Net con capas profunda, sin normalización



Aumentar el tamaño de la red ha posibilitado realizar reconstrucciones notablemente mejores. Sin embargo, en imágenes con información limitada, como la tercera muestra, ha interpretado incorrectamente los datos, confundiendo un 4 con un 9.

- Iteración 3: U-Net con capas profunda, con normalización (dropout 0.3 y batch normalization)

La inclusión de capas de normalización ha beneficiado el proceso de reconstrucción. Los bordes aparecen bien definidos y la reconstrucción es precisa.

- Iteración 4: U-Net con capas profunda, con normalización (dropout 0.1 y batch normalization)

El resultado parece muy similar a la anterior iteración. Puede considerarse óptimo.

5.1.2.2 MNIST handwritten digits – Formato vector de intensidades

- Iteración 1: U-Net reducida, sin normalización

La red, debido a su tamaño compacto, es incapaz de reconstruir la imagen a partir del vector de intensidades.

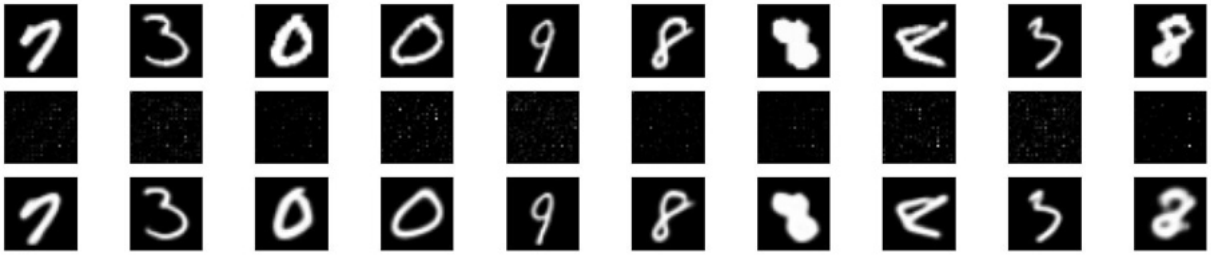
- Iteración 2: U-Net con capas profunda, sin normalización

Aumentar el número de capas profundas ha contribuido a la mejora en la calidad de reconstrucción. Sin embargo, algunas muestras, como la cuarta, no son muy precisas.

- Iteración 3: U-Net con capas profunda, con normalización (dropout 0.3 y batch normalization)

Después de incorporar capas de normalización, la red muestra un rendimiento notablemente mejor en comparación con las iteraciones previas. Los números son nítidos, los detalles se han representado con precisión y sin fallos.

- Iteración 4: U-Net con capas profunda, con normalización(dropout 0.1 y batch normalization)



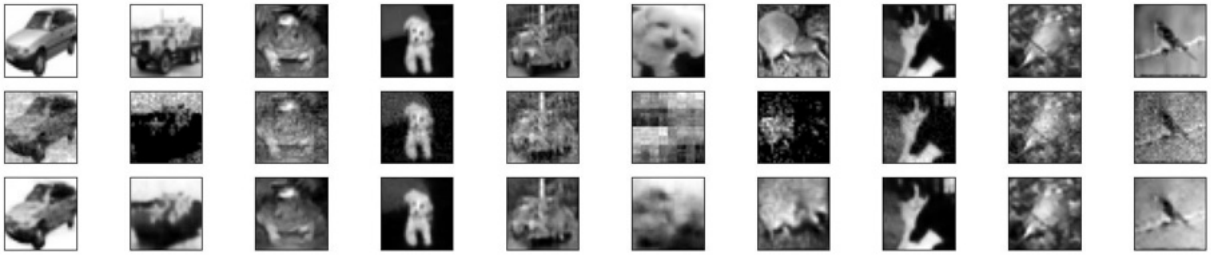
Esta red ha dado unos buenos resultados, similares a la iteración anterior.

5.1.2.3 CIFAR 10– Formato imagen

- Iteración 1: U-Net reducida, sin normalización

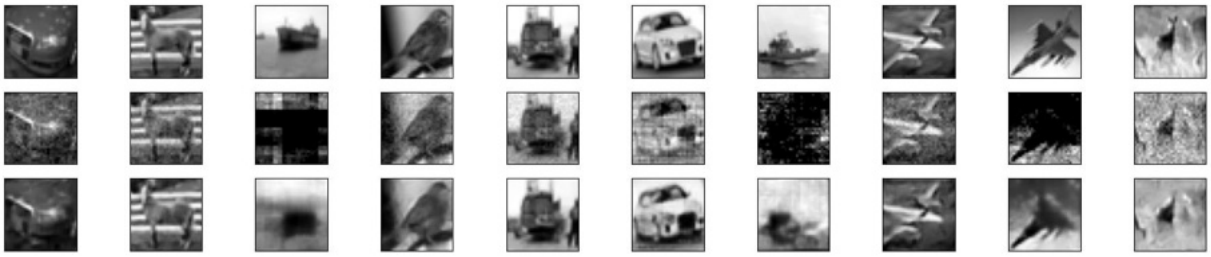
La red de dimensiones compactas ha intentado reconstruir minimizando el ruido en la imagen. Sin embargo, aún resulta complicado identificar ciertas escenas.

- Iteración 2: U-Net con capas profunda, sin normalización



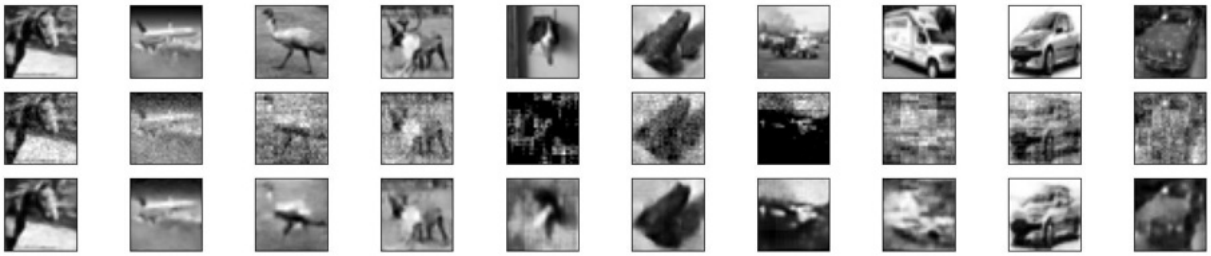
Los resultados obtenidos son bastante precisos. Las capas de salto de la U-Net funcionan de manera eficiente en conjunto con el incremento de las capas densas. Esto se complementa con el hecho de que la imagen de entrada tiene una distribución espacial similar a la de la salida, a diferencia del caso con el vector de intensidades donde los datos de entrada no guardan la misma distribución espacial que a la salida.

- Iteración 3: U-Net con capas profunda, con normalización (dropout 0.3 y batch normalization)



Los resultados son ligeramente mejores que en la iteración anterior.

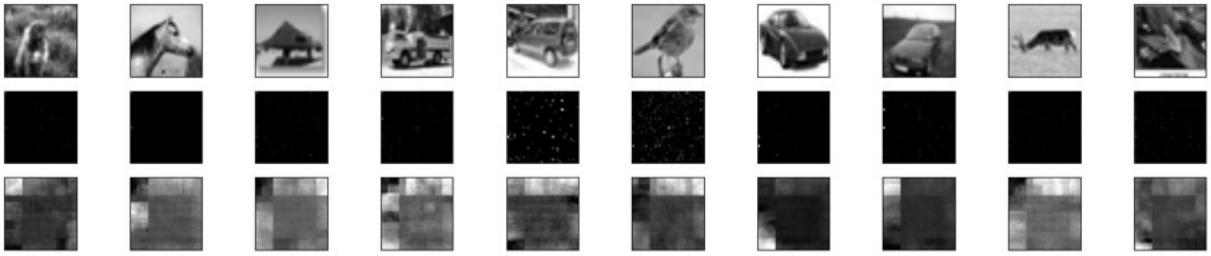
- Iteración 4: U-Net con capas profunda, con normalización (dropout 0.1 y batch normalization)



Los resultados son bastante similares a la iteración anterior.

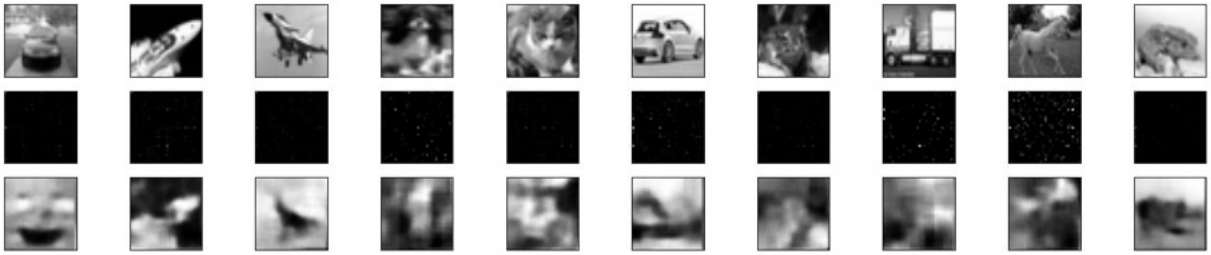
5.1.2.4 CIFAR 10 – Formato vector de intensidades

- Iteración 1: U-Net reducida, sin normalización



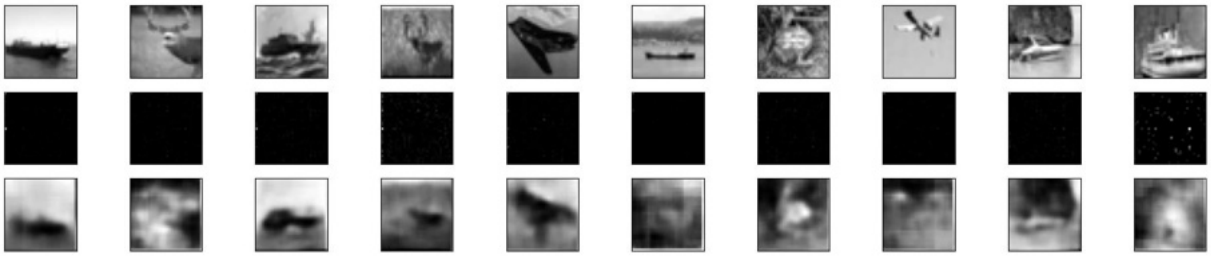
La red, debido a su tamaño compacto, no tiene las suficientes capas densas para procesar adecuadamente el vector de intensidades. Las reconstrucciones resultan inexactas en todas las muestras.

- Iteración 2: U-Net con capas profunda, sin normalización



La reconstrucción es sensiblemente mejor que la iteración anterior, pero las escenas continúan sin ser reproducidas.

- Iteración 3: U-Net con capas profunda, con normalización (dropout 0.3 y batch normalization)



Parece que la reconstrucción es ligeramente peor a la iteración anterior, las capas de normalización no hay sido de ayuda, si no todo lo contrario han dificultado el aprendizaje de la red.

- Iteración 4: U-Net con capas profunda, con normalización (dropout 0.1 y batch normalization)

La reducción en la cantidad de normalización en comparación con la iteración 3 ha posibilitado n aprendizaje ligeramente mejor por parte de la red. Sin embargo, sigue sin proporcionar resultados satisfactorios.

5.2 Evaluación cuantitativa

Para la evaluación cuantitativa, se han utilizado las métricas PSNR y SSIM para cada uno de los conjuntos de datos. Estas se han compilado en una tabla para su comparación, tanto antes como después de la reducción de ruido con las redes neuronales. Y siempre comparado con las imágenes de referencia.

Para cada uno de los valores, se ha calculado la PSNR y la SSIM para todo el conjunto de datos y se ha hecho una media aritmética.

MNIST Handwritten digits - formato imagen

Tipo de Modelo	PSNR antes de limpiar	PSNR después de limpiar	SSIM antes de limpiar	SSIM después de limpiar
Autoencoder 1	71.4797	56.7712	0.5319	0.9647
Autoencoder 2	71.4797	56.8166	0.5319	0.9740
Autoencoder 3	71.4797	56.8095	0.5319	0.9721
Autoencoder 4	71.4797	56.7801	0.5319	0.9759
U-Net 1	71.4797	56.8440	0.5319	0.9754
U-Net 2	71.4797	56.7472	0.5319	0.9839
U-Net 3	71.4797	56.7397	0.5319	0.9837
U-Net 4	71.4797	56.8096	0.5319	0.9851

MNIST Handwritten digits - formato vector de intensidades

Tipo de Modelo	PSNR antes de limpiar	PSNR después de limpiar	SSIM antes de limpiar	SSIM después de limpiar
Autoencoder 1	31.2058	57.9864	0.0188	0.5292
Autoencoder 2	31.2058	56.9334	0.0188	0.8946
Autoencoder 3	31.2058	57.0854	0.0188	0.8896
Autoencoder 4	31.2058	56.9477	0.0188	0.9127
U-Net 1	31.2058	58.0566	0.0188	0.5267
U-Net 2	31.2058	56.8849	0.0188	0.9482
U-Net 3	31.2058	56.9662	0.0188	0.9543
U-Net 4	31.2058	56.9012	0.0188	0.9571

CIFAR10 - formato imagen

Tipo de Modelo	PSNR antes de limpiar	PSNR después de limpiar	SSIM antes de limpiar	SSIM después de limpiar
Autoencoder 1	75.1803	60.0196	0.8647	0.8631
Autoencoder 2	75.1803	60.1237	0.8647	0.8480
Autoencoder 3	75.1803	60.5282	0.8647	0.8384
Autoencoder 4	75.1803	60.2813	0.8647	0.8603
U-Net 1	75.1803	60.0578	0.8647	0.8978
U-Net 2	75.1803	60.0188	0.8647	0.9336
U-Net 3	75.1803	60.1594	0.8647	0.9367
U-Net 4	75.1803	60.0930	0.8647	0.9376

CIFAR10 - vector de intensidades

Tipo de Modelo	PSNR antes de limpiar	PSNR después de limpiar	SSIM antes de limpiar	SSIM después de limpiar
Autoencoder 1	28.8809	61.3657	0.0015	0.3720
Autoencoder 2	28.8809	59.7310	0.0015	0.3973
Autoencoder 3	28.8809	60.7129	0.0015	0.4815
Autoencoder 4	28.8809	60.4255	0.0015	0.4665
U-Net 1	28.8809	61.3888	0.0015	0.3771
U-Net 2	28.8809	59.8401	0.0015	0.6074
U-Net 3	28.8809	59.7720	0.0015	0.6381
U-Net 4	28.8809	59.9918	0.0015	0.6530

A la vista de los resultados obtenidos a partir de las métricas podemos afirmar que la arquitectura U-Net con los hiperparámetros de la iteración 4 ha sido la que ha tenido mejores resultados en los cuatro conjuntos de datos.

No obstante, la congruencia entre las métricas PSNR no es consistente al determinar cuál red brinda los mejores resultados. Aunque ambas métricas (PSNR y SSIM) se emplean para evaluar la calidad de las imágenes reconstruidas, operan bajo principios distintos.

El PSNR evalúa el error cuadrático medio entre dos imágenes, donde un valor más alto denota una mayor calidad. Sin embargo, no considera la percepción humana, lo que puede hacer que no se alinee con nuestro juicio sobre la calidad de una imagen.

Por otro lado, la SSIM analiza la similitud estructural entre dos imágenes y se alinea más estrechamente con la percepción humana de calidad. Dado estos matices, no es raro que estas métricas no concuerden siempre. A pesar de ello, es beneficioso utilizar ambas para lograr una apreciación más integral de la calidad de una imagen.

6 CONCLUSIONES

En el desarrollo de este Trabajo Final de Máster, se ha abordado evaluar distintas arquitecturas neuronales para determinar cual resulta óptima para la reducción de ruido en aplicaciones de SPI con CS. De las observaciones derivadas, se subraya la influencia que las capas densas tienen en la reconstrucción acertada de las imágenes.

Tanto la arquitectura Autoencoder Convolucional como la U-Net han mostrado su eficacia frente a conjuntos de datos más elementales, como el MNIST Handwritten Digits. No obstante, al utilizarse las mismas redes con conjuntos de mayor complejidad, como el CIFAR-10, los retos de reconstrucción aumentaron.

Entre las arquitecturas evaluadas, la U-Net sobresalió particularmente cuando se trataba de conjuntos de datos en formato de imagen. Esto puede atribuirse, en gran medida, a las capas de salto, las cuales transmiten información directamente desde las capas de codificación hacia las capas de decodificación, favoreciendo así la conservación de la información espacial, siendo capaz de reconstruir la escena sin necesidad de mucha información.

A pesar de ello, los conjuntos basados en vectores de intensidades han presentado desafíos sustanciales. La complejidad intrínseca de las imágenes naturales en el conjunto CIFAR-10, sumada a la naturaleza del vector de intensidades, ha dificultado una reconstrucción de calidad con las redes propuestas en este TFM.

Para concluir, este estudio ha logrado no solo cumplir con sus objetivos propuestos, sino que también ha aportado una comprensión profunda de la actuación de las redes Autoencoder y U-Net en el exigente

contexto de reducción de ruido para aplicaciones SPI con CS. Este TFM establece un punto de partida robusto para futuras investigaciones y aplicaciones prácticas en el ámbito de la reconstrucción de imágenes dentro de aplicaciones de SPI y CS.

7 BIBLIOGRAFÍA

[0] Wen-Cheng Li, 2020, "Deep-learnin-based single-photon-counting compressive imagin via jouintly trained subpixel convolution sampling", Optica – Applied Optics
Disponible en: https://opg.optica.org/ao/fulltext.cfm?uri=ao-59-23-6828

[1] Apuntes de "Introducción a la Visión Artificial", 2015, Universidad de Córdoba.
Disponible en: http://www.uco.es/users/ma1fegan/2015-2016/vision/Temas/ruido.pdf

[2] Jose Oriol Moreno Usó, 2022, "Construcción de un pipeline de conversión de autoencoders convolucionales para inferencia en Edge TPU", TFG Universidad Valencia.
Disponible en: https://www.uv.es/lapeva/Thesis/TFG_2022_Jose_Oriol.pdf

[3] Rini Mayasari, Nono Heryana, 2019, "Reduce Noise in Computed Tomography Image using Adaptive Gaussian Filter", International Journal of Computer Techniques.
Disponible en: http://www.ijctjournal.org/Volume6/Issue1/IJCT-V6I1P4.pdf

[4] Serban-Vasile Carta, 2018, "Salt and Pepper Noise Removal by Combining Genetic Algorithms – Neural Networks and Statistical Methods", 2018 International Conference on Communications (COMM)
Disponible en: https://ieeexplore.ieee.org/abstract/document/8484807

[5] Meng Lyu, Wei Wang, 2017, "Deep-learning-based ghost imaging", Nature.com
Disponible en: https://www.nature.com/articles/s41598-017-18171-7

[6] Fei Wang, Hao Wang, 2019, "Learning from simulation: An end-to-end deep-learning approach for computational ghost imaging", Optics Express
Disponible en: https://opg.optica.org/oe/fulltext.cfm?uri=oe-27-18-25560&id=417106

[7] Jose Cuartas, 2021, "El concepto de la convolución en gráficos, para comprender las Convolutional Neural Networks (CNN) o redes convolucionadas", blog personal.

Disponible en: https://josecuartas.medium.com/el-concepto-de-la-convoluci%C3%B3n-en-gr%C3%A1ficos-para-comprender-las-convolutional-neural-networks-cnn-519d2eee009c

[8] Yessenia Jáuregui Sánchez, 2019, “The nature of noise in single-pixel cameras and their application in imaging and their application in imaging through scattering media”, Tesis doctoral UJI
Disponible en: https://www.tdx.cat/handle/10803/667330

[9] Marco F. Duarte, Mark A. Davenport, 2008, “Single-pixel imaging via compressive sampling”, IEEE Signal Processing Magazine
Disponible en : https://ieeexplore.ieee.org/document/4472247

[10] Juhwan Kim, 2017, “Denoising Auto-Encoder Based Image Enhancement For High Resolution Sonar Image”, 2017 IEEE Underwater Technology (UT)
Disponible en: https://ieeexplore.ieee.org/abstract/document/7890316

[11] Kai Zhang, 2017, “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising”, IEEE Transaction on Image Processing
Disponible en: https://arxiv.org/abs/1608.03981

[12] Haoyu Ren, 2018, “DN-ResNet: Efficient Deep Residual Network for Image Denoising”, Asian Conference of Computer Vision 2018
Disponible en: https://arxiv.org/abs/1810.06766

[13] Fan Jia, 2021, “DDUNet: Dense Dense U-Net with Applications in Image Denoising”, 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)
Disponible en: https://ieeexplore.ieee.org/document/9607593

[14] Min-Long Zhu, 2022, “Image Denoising Based on GAN with Optimization Algorithm”, MDPI electronics
Disponible en: https://www.mdpi.com/2079-9292/11/15/2445

8 ANEXOS

8.1 Descripción modelos autoencoders convolucionales utilizados

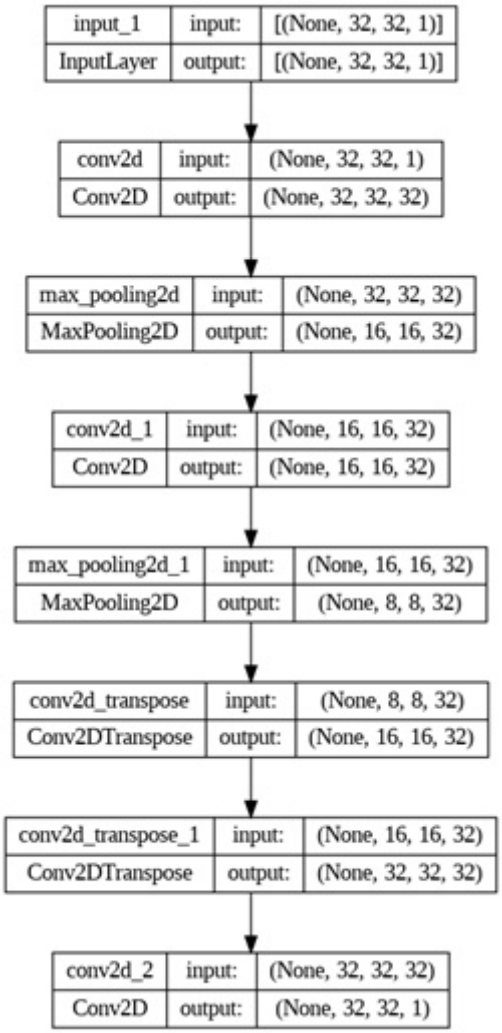


Ilustración 44 - Modelo Autoencoder Convolucional iteración 1

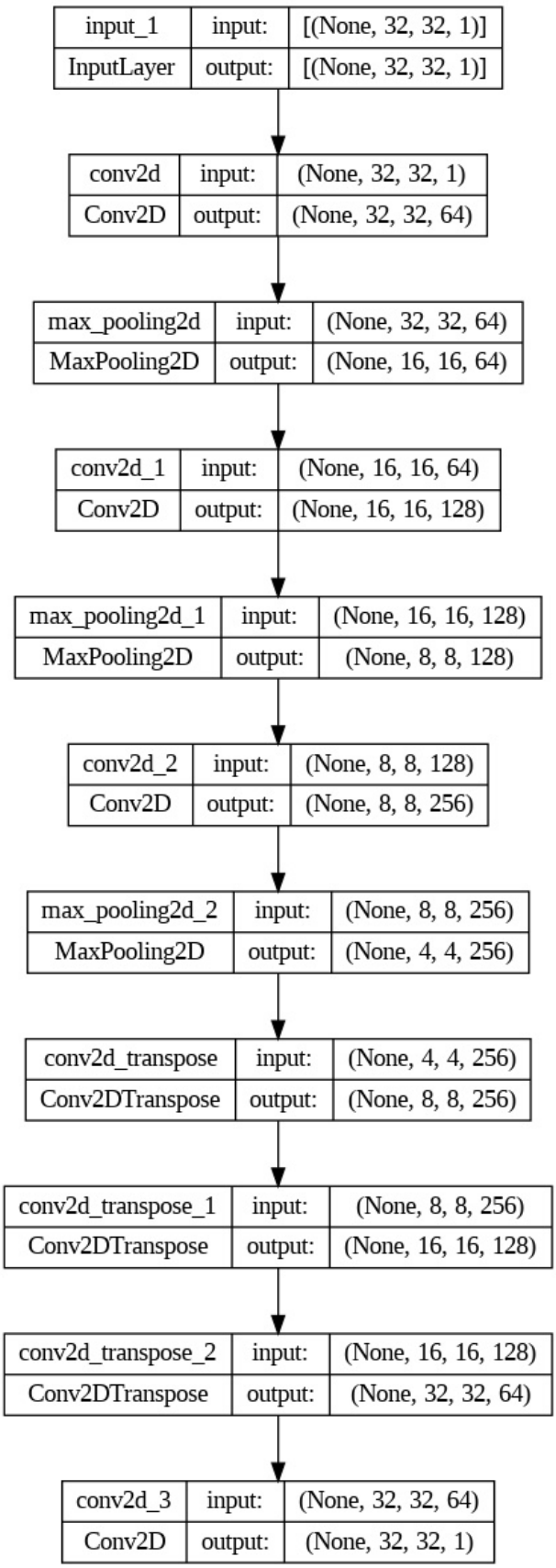


Ilustración 45 - Modelo Autoencoder Convolucional iteración 2

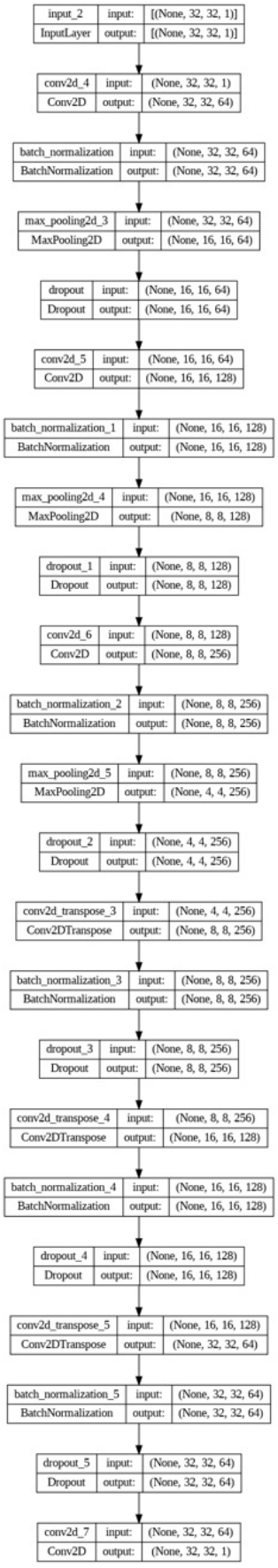


Ilustración 46 - Modelo Autoencoder Convolucional iteración 3

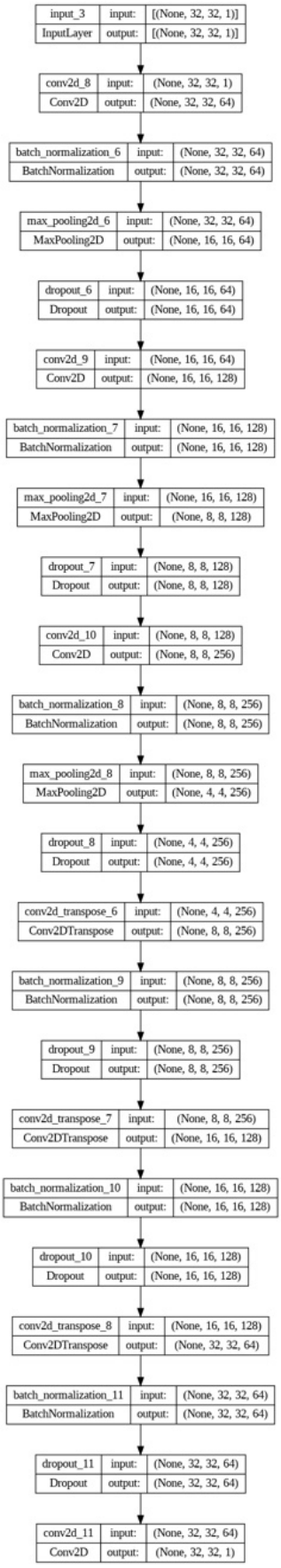


Ilustración 47 - Modelo Autoencoder Convolucional iteración 4

8.2 Descripción modelos U-Net utilizados

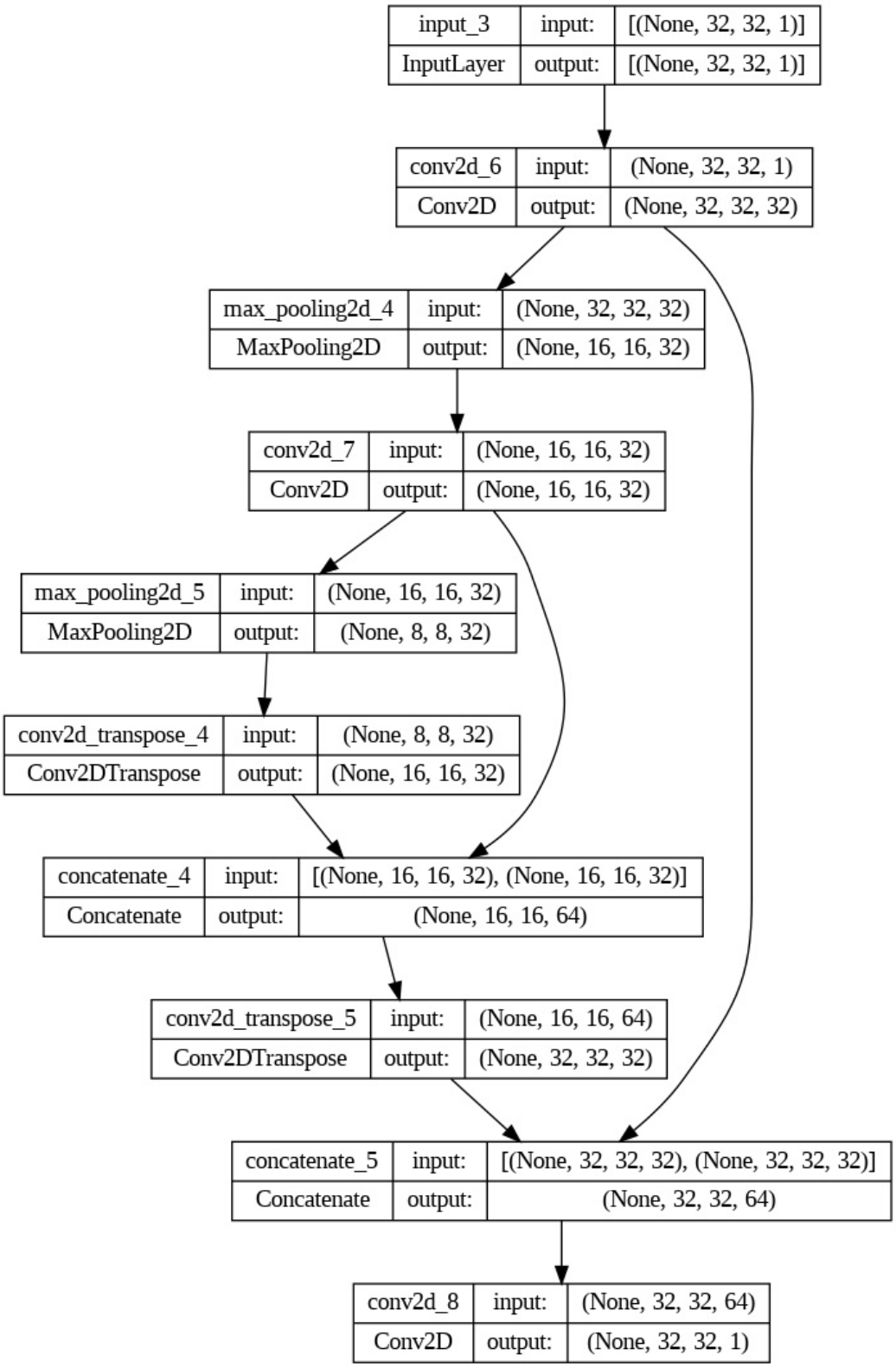


Ilustración 48 - Modelo U-Net iteración 1

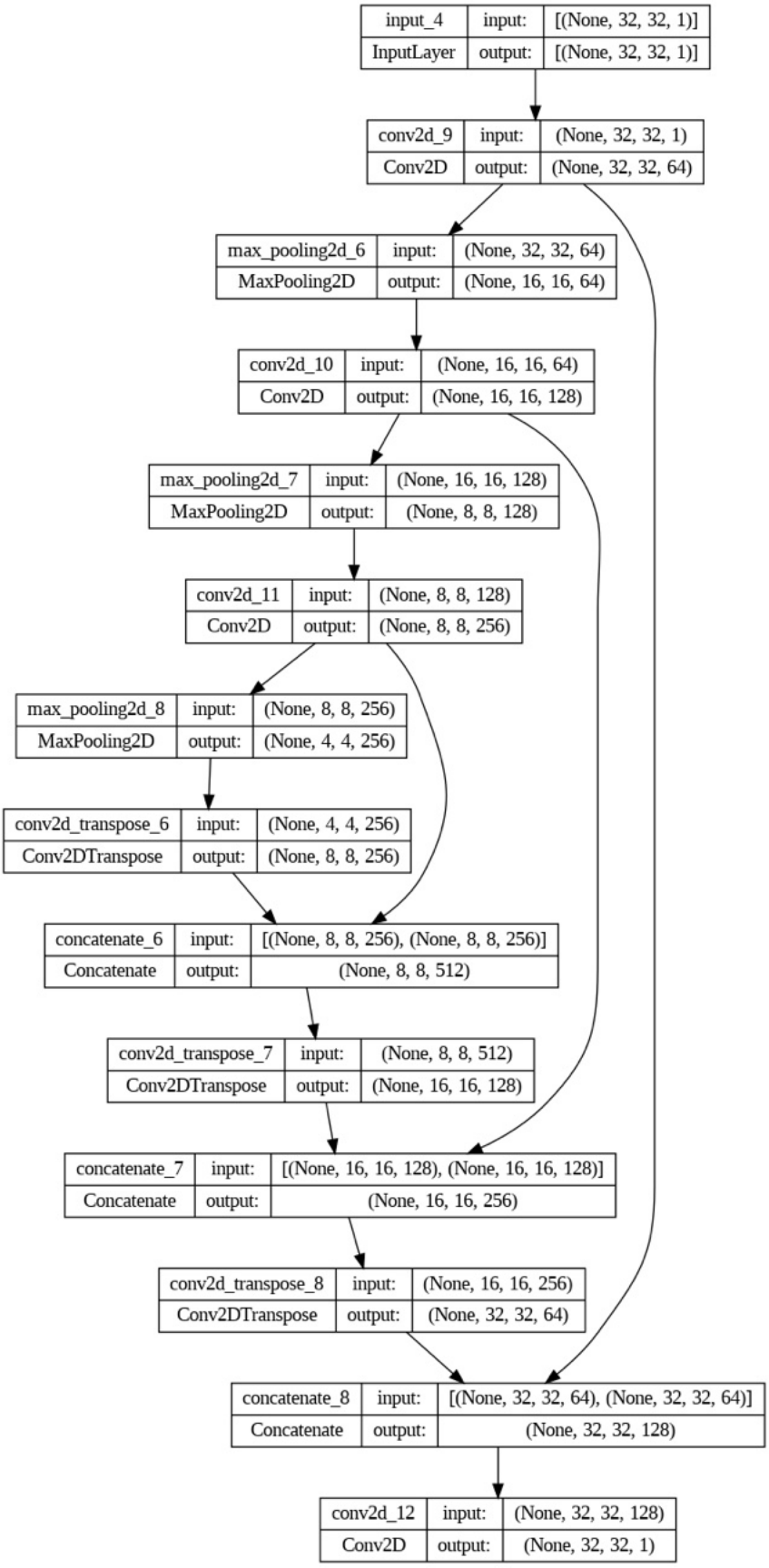


Ilustración 49 - Modelo U-Net iteración 2

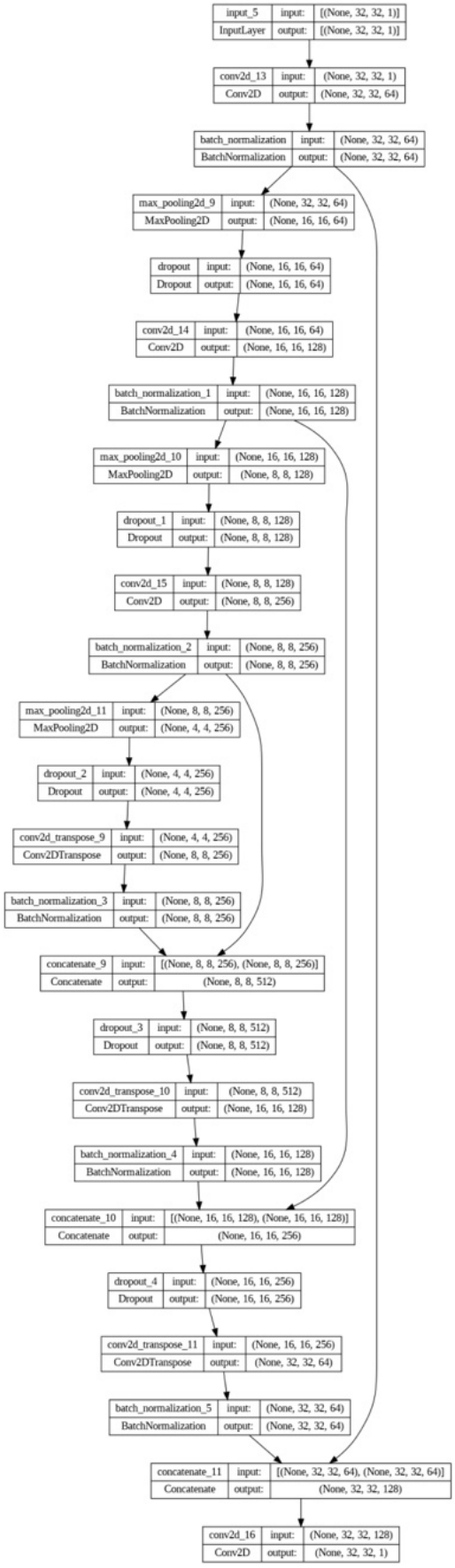


Ilustración 50 - Modelo U-Net iteración 3

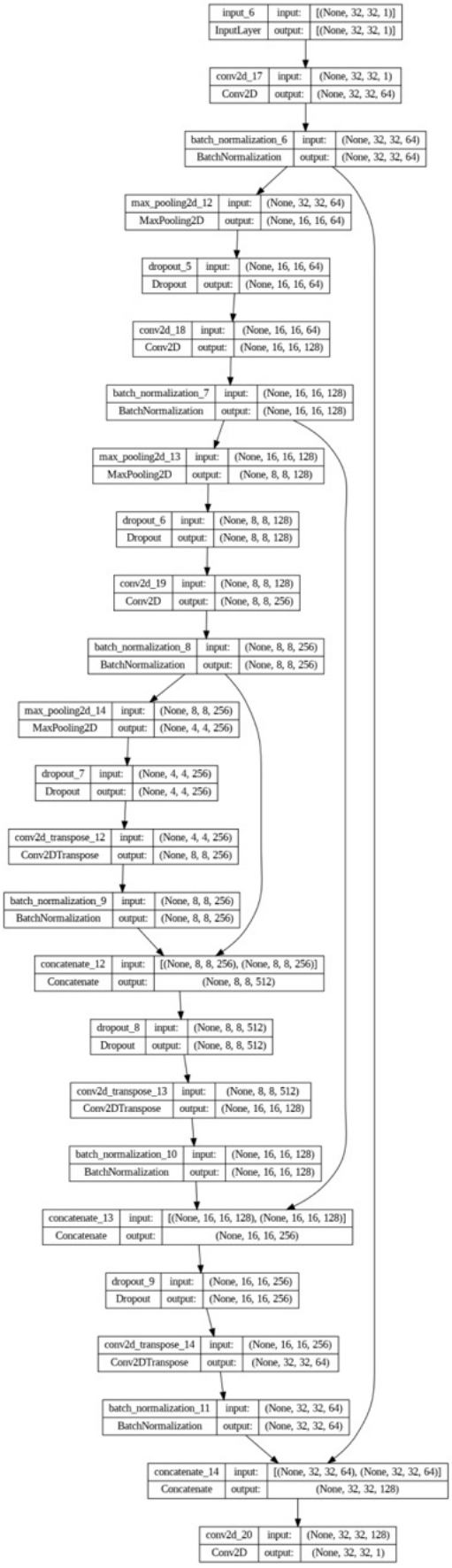


Ilustración 51 - Modelo U-Net iteración 4