Universidad Zaragoza
1542

Trabajo Fin de Grado

Marcas de agua imperceptibles usando redes neuronales profundas

Imperceptible Watermarking using Deep Neural Networks

Autor/es

Juan Manuel Sánchez Casabona

Director/es

Javier Civera Sancho

Escuela de Ingeniería y Arquitectura. Universidad de Zaragoza.

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Juan Manuel Sánchez Casabona,

con nº de DNI 73020376E en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Marcas de agua imperceptibles usando redes neuronales profundas

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, a 25 de Agosto de 2018

Fdo: Juan Manuel Sánchez Casabona

Agradecimientos

En primer lugar, me gustaría agradecer su infinita dedicación a las personas que me han ayudado en la realización de este proyecto, José M. Fácil y Javier Civera. Gracias a Chema por atenderme desde Alemania siempre que lo he necesitado y guiarme a lo largo de la realización de todo el trabajo.

Me gustaría también agradecer de una forma especial a todos mis compañeros del departamento de Ingeniería Mecánica, que me enseñaron en el periodo que estuve ahí a cómo afrontar los problemas que surgían y conseguir resolverlos y me permitieron embarcarme en este proyecto para descubrir una nueva disciplina.

Por último, agradecer a mi familia y amigos la constante atención que le han prestado al trabajo animando en los momentos en los que las cosas no salían.

Resumen

El crecimiento del flujo de imágenes a través de la red, con su consecuente trasmisión descentralizada, ha provocado necesidades legales por parte de muchos profesionales para defender su autoría. Uno de los retos que se plantean es conseguir la mayor calidad en las imágenes, pero manteniendo un elemento diferencial que demuestre la autoría. El marcado de agua imperceptible es una de las técnicas en investigación para conseguir la defensa de la autoría de las imágenes con la mínima pérdida de calidad. Por lo tanto, este trabajo se va a centrar en, a través del aprendizaje automático, usar redes neuronales que sean capaces de introducir marcas de agua de forma imperceptible en las imágenes.

La implementación conlleva varios desafíos. Por un lado, va a ser necesario encontrar el método para conseguir realizar el marcado de agua imperceptible. Por otro, va a ser necesario tener un método que sea capaz de recuperar la marca de la imagen marcada. La complejidad del problema radica en acoplar ambos métodos y determinar qué redes neuronales son las que mejor resuelven el problema planteado. Se han usado los conjuntos de datos MNIST e Image-Net. Se han empleado redes neuronales con sucesiones de capas convolucionales, a partir de las cuales, se han conseguido obtener buenos resultados.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Los avances en las nuevas tecnologías han provocado un flujo de información entre usuarios que va en constante crecimiento. Las imágenes representan una gran parte de los contenidos que se comparten a través de la red. Debido a la sencillez de transmisión, todas las personas las comparten a diario para asuntos que van desde el ámbito más estrictamente profesional (medios de comunicación, estudios de fotografía, e-mails a proveedores) hasta el ámbito social (Facebook, Twitter, Instagram). Los avances en las nuevas tecnologías han provocado muchos beneficios para el ser humano, pero a veces se encuentran algunos puntos negativos que pueden llevar a conflicto. La sencillez que ofrece la transmisión de la información de forma instantánea es algo positivo. Sin embargo, esta facilidad puede producir una la falta de control sobre los contenidos y provocar conflictos entre usuarios. Estos conflictos van desde robos de identidad, publicación de imágenes con *Copyright* hasta la falsificación de documentos.

La apropiación indebida se produce cuando un usuario toma posesión de algo que no le pertenece y lo usa como propio. Como conflicto, la apropiación indebida de imágenes se podría considerar como uno de los temas principales. Un ejemplo sería un medio de comunicación que comparte una imagen en las redes, y otro usario toma esa imagen y decide compartirla y lucrarse a través de sus propios medios sin indicar en ningún momento la autoría original de la imagen. Este problema no es un tema nuevo que vaya ligado a las nuevas tecnologías, pero últimamente se ha visto incrementado por el desarrollo de estas.

El marcado de agua, más conocido por el termino inglés *watermarking*, consiste en el uso de un elemento identificativo de un autor con el objetivo de defender la autoría de un trabajo. Se usa en pintura cuando un autor firma su lienzo y en fotografía cuando el propietario de una imagen incrusta su logo en sus fotografías. El marcado de agua en imágenes se puede realizar de múltiples formas (Figura 1.1) Aporta la ventaja de defender la autoría, pero tiene puntos negativos. Uno de los principales es la pérdida de calidad que provoca sobre las imágenes. Las marcas de agua provocan que no se aprecien los detalles y la imagen se vea completamente mermada por la marca de agua. En una sociedad como la actual, en la cual la estética y la calidad son una exigencia cada vez más presente, esta pérdida de calidad en el contenido de la imagen no es deseable.

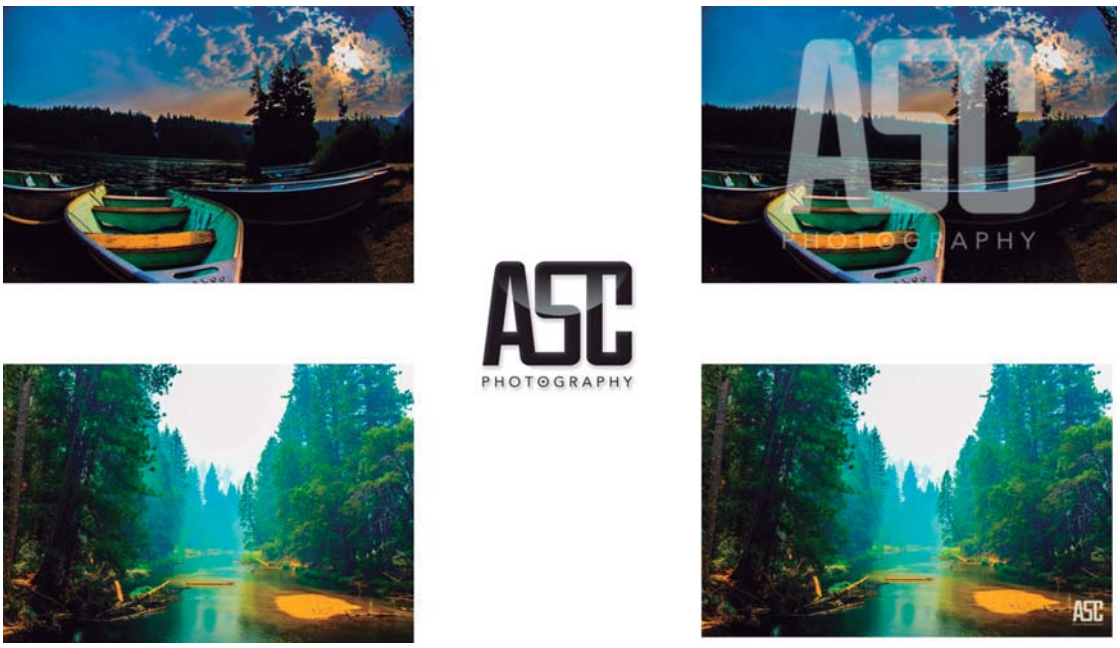


Figura 1.1: Ejemplo de dos tipos distintos de marcado. Se observa la marca de agua del fotógrafo en la parte central. A la izquierda las fotos sin marcado de agua y a la derecha las fotos con el marcado de agua.

Para resolver el problema de mantener la calidad de la imagen nace el marcado de agua imperceptible (Figura 1.2), conocido en inglés como *imperceptible watermarking*. Este consiste en incrustar una marca de agua en una imagen de forma que el usuario no la puede detectar, pero sí puede ser detectada por otros métodos. Las aplicaciones y ventajas que se obtienen respecto al watermarking perceptible son múltiples. Por ejemplo, cualquier medio de comunicación puede tener protegidas todas las imágenes que tengan en su página web sin necesidad de que los usuarios se vean penalizados por la pérdida de calidad. Otro ejemplo sería el de un fotógrafo profesional que quiere compartir sus fotos para su comercialización, pero quiere que la gente valore la calidad. En cualquiera de estos dos casos, ante una situación de apropiación indebida de imágenes, si el autor tuviera las herramientas necesarias, podría reclamar y demostrar la autoría de las imágenes.

Este trabajo se centra en la problemática que presenta realizar marcado de agua imperceptible sobre una imagen haciendo uso de alguna de las técnicas que se encuentran disponibles. El trabajo se enmarca dentro del campo de la visión por computador. La visión por ordenador es una disciplina que estudia el análisis, procesado e interpretación de las imágenes mediante sistemas computacionales. La visión abarca, entre otros, problemas como el reconocimiento de patrones en imágenes, la localización y el mapeado simultáneos o la detección de expresiones faciales. Se aplica en campos muy diversos, que pueden ir desde la identificación de elementos en procesos en procesos industriales hasta el reconocimiento de señales en un vehículo autónomo (Figura 1.3). En la ultima década,

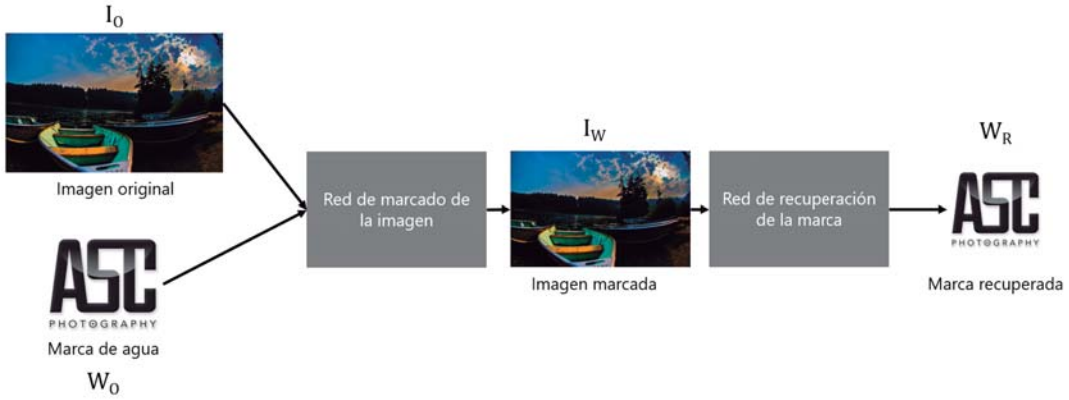


Figura 1.2: Estructura de un algoritmo con *imperceptible watermarking*. La imagen y la marca de agua pasan por un proceso de marcado, obteniendo como salida la imagen con la marca de agua incrustada. A partir de otro algoritmo de recuperación de la marca se obtiene la marca de agua incrustada.

la mayoría de problemas del campo han visto un progreso sensacional debido al gran desarrollo del *deep learning*[1]. Dentro de esta evolución, las redes neuronales son una de las herramientas más usadas para alcanzar el aprendizaje. Son estructuras de capas que tienen distintos parámetros que van actualizándose durante un proceso de entrenamiento hasta estar configuradas para obtener la salida deseada.

Figura 1.3: Dos distintos ejemplos de aplicaciones de la visión por ordenador. A la izquierda una cámara de vision usada como Poka-Yoke en un sistema productivo. A la derecha un ejemplo de reconocimiento de señales en un vehículo.

Dentro del campo de la visión, este trabajo se enmarca en la comprensión y genera-

[1]Palabra inglesa usada para referirse en castellano al aprendizaje profundo. Es un aspecto de la inteligencia artificial que se ocupa de la extracción de patrones a partir de cantidades masivas de datos.

lización de datos. El objetivo es hacer que un marcado de agua que esté presente en una imagen no pueda ser observado por el ser humano, pero si por un algoritmo de visión. La complejidad del problema radica por ello en configurar y diseñar las redes neuronales de forma precisa para que sean capaces de incrustar el marcado haciendo que sea lo más imperceptible posible y de forma posterior, cuando sea necesario, recuperar la marca de agua sin alteraciones en ella. Hasta el momento se han desarrollado técnicas de marcado de agua imperceptible, como en [1], pero hasta donde llega el conocimiento del autor no se ha hecho uso de redes neuronales para afrontar este problema.

Para resolver el problema se van a tener que alcanzar dos objetivos principales. Uno de ellos corresponde a la parte del diseño de la red neuronal. El otro, consiste en evaluar los resultados para obtener una validación del modelo propuesto. Se exponen en detalle a continuación:

- Crear un sistema acoplado, que, por un lado, sea capaz de incrustar una marca de agua sobre una imagen y que la marca no altere la calidad de la imagen y no pueda ser percibida por el ojo humano. Por otro lado, implementar un sistema que sea capaz de recuperar una marca de agua de una imagen que contiene una marca de agua imperceptible, recuperándola con la mayor calidad posible. Para ambos procesos se utilizarán redes neuronales artificiales que se configurarán a través de un proceso de entrenamiento.
- Evaluar las soluciones desarrolladas y comparar las distintas redes y técnicas usadas para realizar el imperceptible watermarking. La evaluación se realizará comparando la calidad de la imagen original con la imagen con el marcado de agua incrustado y, por otro lado, comparando la marca de agua original con la marca de agua recuperada por la red neuronal.

En este capítulo han quedado acotados tanto el problema como los objetivos que se desean alcanzar. En el capítulo 2 se introducen conceptos básicos sobre redes neuronales con el objetivo de facilitar la comprensión del resto del trabajo. En el capítulo 3 se van a explicar las técnicas usadas para realizar el marcado de agua imperceptible. En el capítulo 4 se van a mostrar los resultados experimentales, explicando la metodología, los distintos experimentos realizados y las comparaciones usadas. En el capítulo 5 se van a extraer las conclusiones generales del trabajo. En el capítulo 6 se van a explicar las herramientas usadas y la distribución temporal del trabajo.

2. Introducción a las redes neuronales

En este capítulo se desarrollan conceptos de redes neuronales que van a facilitar la comprensión del resto del trabajo.

Las redes neuronales se encuentran dentro de lo que se conoce como aprendizaje automático o *machine learning*, la rama de las ciencias de la computación cuyo objetivo es la extracción de información a partir de datos. A pesar de que las redes neuronales se han estudiado a lo largo de todo el S.XX, estas no han mostrado todo su potencial hasta la actualidad. Los avances en computación han permitido desarrollar componentes con memoria y capacidad de procesado suficientes para resolver estos problemas, sobre todo en el ámbito de las tarjetas gráficas (GPUs).

Las redes neuronales basan su funcionamiento en el aprendizaje a partir de unos datos de entrada que le sirven de entrenamiento para fijar sus parámetros. Estos datos de entrada pueden ir desde conjuntos de datos estadísticos para predecir la tendencia de venta de un producto hasta conjuntos de imágenes con el objetivo de encontrar patrones ocultos, como es el caso de este trabajo.

2.1. Neuronas artificiales

La unidad básica de una red neuronal es la neurona artificial (Figura 2.1). Cada neurona emula el comportamiento de una neurona biológica, de forma que procesa unos datos de entrada. Estos se introducen como un vector n-dimensional $X = (x_1, x_2, ..., x_n)$, y produce una salida (y) como resultado del procesamiento de la entrada.

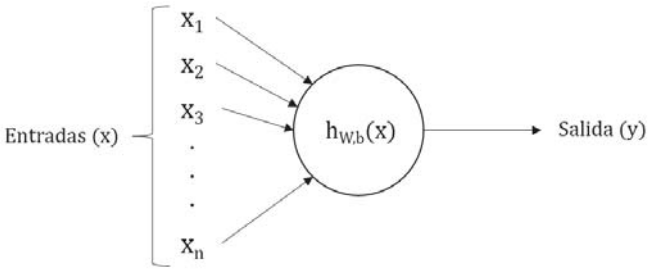


Figura 2.1: Estructura de una neurona artificial. Una serie de entradas son procesadas por la neurona a través de sus pesos y *bias* obteniendo una salida.

La salida de la neurona viene determinada por el producto escalar de los datos de entrada por un vector de pesos $W = (w_1, w_2, ..., w_n)^\intercal$ (tantos como entradas) más un sesgo, denominado en inglés como *bias*. Por último, es necesario aplicar una función de activación no lineal f para poder obtener la salida. La formulación matemática es la siguiente:

$$h_{W,b}(X) = f\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{2.1}$$

Alguna de las funciones de activación más comunes (Figura 2.2) son la tangente hiperbólica, la sigmoide y la ReLU[1].

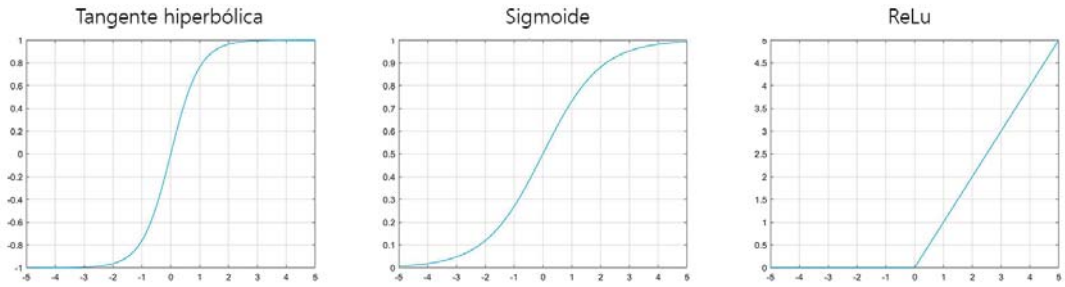


Figura 2.2: A la izquierda la tangente hiperbólica $f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, en el centro la función Sigmoide $f(x) = \frac{1}{1+e^{-x}}$, y a la derecha la función ReLU (*Rectified Linear Unit*) $f(x) = max(0, x)$.

[1]Acrónimo del inglés Rectifier Linear Unit

2.2. Tipos de estructuras

Una capa está constituida por un conjunto de neuronas artificiales en paralelo, y un conjunto de capas en serie forman una red neuronal.

El objetivo de las redes neuronales es aproximar funciones mucho más complejas, que no serían posibles con una sola neurona. Dentro de una capa encontramos un conjunto de neuronas, cada una con un número de pesos propios, que se ven multiplicados por la entrada. De esta forma, cada una de las neuronas de cada una de las capas tienen la misma entrada, que proviene de la salida de la capa anterior, pero una salida distinta al tener cada una de las neuronas distintos pesos.

Dentro de una capa, los pesos de cada neurona vienen determinados por el proceso de aprendizaje de la red en el entrenamiento. El número de neuronas de cada capa, el número de capas de la red y la manera de combinarlas viene fijado por el diseñador de la red. Estos parámetros son los que se conocen como hiperparámetros. Las capas intermedias entre la salida y la entrada son conocidas como capas ocultas. Un ejemplo de red sencilla sería la mostrada en la Figura 2.3.

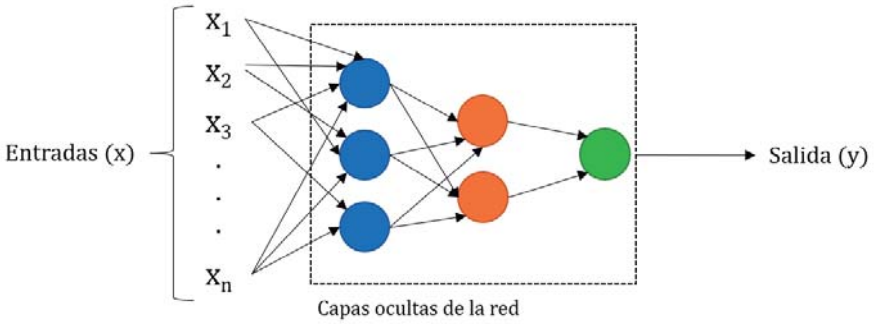


Figura 2.3: Ejemplo de una estructura de una red. Las neuronas se representan con círculos. Tiene dos capas ocultas, con tres y dos neuronas respectivamente, y una neurona de salida.

Los distintos tipos de capas que se pueden encontrar son las siguientes:

- **Capa totalmente conectada ("fully-connected"):** las neuronas de esta capa se conectan a todas las neuronas de la capa anterior (Figura 2.4). Lo que se consigue es un procesamiento global de la capa anterior. La principal desventaja es que considera una capa "cara", es decir, al conectar todas las neuronas se requiere un gran número de operaciones. El número de parámetros a manejar viene determinado de la siguiente manera: (M+1)*N, donde M es el tamaño de la entrada y N el número de neuronas de la capa actual. Ejemplificando, para una capa con de entrada de tamaño de 256x256 = 65.536 neuronas, si se quisiera conectar a una capa

con la mitad de neuronas 32.768, serían necesarios un total de (65.536+1)*32.768= 2.147.516.416 parámetros.

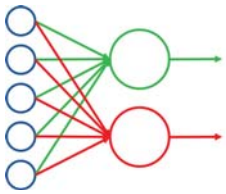


Figura 2.4: Ejemplo de una capa totalmente conectada. Todas las neuronas de una capa están conectadas a todas las neuronas.

- **Capa localmente conectada:** son usadas para identificar relaciones locales, es decir, en el caso de una imagen, patrones que no incluyen información de toda la imagen. Por ejemplo, en una foto de una habitación, una puerta guarda rasgos muy comunes sea cual sea la habitación. En estas capas las neuronas se conectan solamente con unas determinadas neuronas de la capa anterior (Figura 2.5), consiguiendo así que si los datos que hay a la entrada son parecidos conecte solo los datos que guardan similitud, como por ejemplo la puerta en el caso de la habitación. Imaginemos que en el caso de una entrada de tamaño 256x256, el patrón común ocupe únicamente un tamaño de 5x10, por lo tanto, el número de parámetros totales se reducirá a (65.536+1)*50 = 3.276.800. De esta manera el nuevo número de parámetros es un 0,15 % de los parámetros obtenidos en la capa fully connected de este ejemplo.

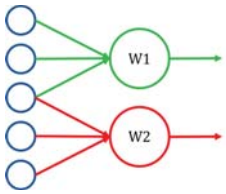


Figura 2.5: Ejemplo de una capa localmente conectada. No todas los elementos anteriores van conectados a todas las neuronas. Cada neurona posee sus propios pesos.

- **Capa convolucional (capa localmente conectada):** es una de las capas localmente conectadas más usadas, tienen la particularidad de que las neuronas de esta capa se conectan al mismo número de neuronas de la capa anterior y comparten el valor de los pesos W entre todas las neuronas de la capa (Figura 2.6). Son un tipo de capas especiales ya que basan su operativa matemática en operaciones de convolución. También poseen la ventaja de tener invarianza espacial, es decir, para el ejemplo de la puerta de la habitación, es capaz de detectarla en cualquier punto de la imagen sin necesidad de estar siempre en el mismo sitio. De vuelta al ejemplo numérico, supongamos que se tiene la imagen de entrada de 256x256 y se crea una capa convolucional con la mitad de neuronas que datos de la entrada, es

decir, 32.768, y cada una de estas neuronas se conectan a una cuarta parte de las entradas $65.536/4 = 16.384$ parámetros serían necesarios únicamente ya que todas las neuronas comparten el valor de todos los pesos.

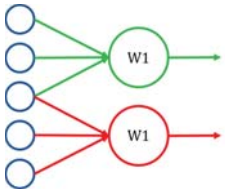


Figura 2.6: Ejemplo de una capa convolucional. Todas las neuronas comparten sus pesos.

- **Capa reductora (*pooling*):** es otro tipo de capa localmente conectada que aplica directamente una función conocida sobre la entrada y no tiene parámetros para aprender. Hay varios tipos, las más conocidas son la del mínimo, la media y el máximo. Por ejemplo, la capa *max-pooling* obtiene a la entrada el máximo del vector de entrada (Figura 2.7). Son capas con muy poco coste computacional. No son de interés en este trabajo debido a la gran pérdida de información que producen.

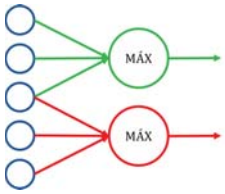


Figura 2.7: Capa reductora máxima. La salida sería el máximo de los valores de los elementos conectados a la neurona.

2.3. Redes neuronales 2D

Trabajar con imágenes implica cambiar a dos dimensiones y ello conlleva cambios en las capas que son explicados a continuación.

Capas convolucionales: al realizar el cambio a dos dimensiones ahora es necesario trabajar con la operación de convolución sobre matrices. La entrada va a ser una imagen y las neuronas van a actuar como un filtro con los pesos de las neuronas, que va a realizar un barrido de la imagen. El tamaño del filtro es un hiperparámetro, por lo tanto, queda definido por el diseñador de la red.

En la figura 2.8 se muestra un proceso de convolución en dos dimensiones, en el que se usa un filtro de tamaño 2x2. El filtro posee los pesos de las neuronas y va recorriendo la imagen multiplicando los valores de los pesos por los valores de la imagen obteniendo así un mapa de salida.

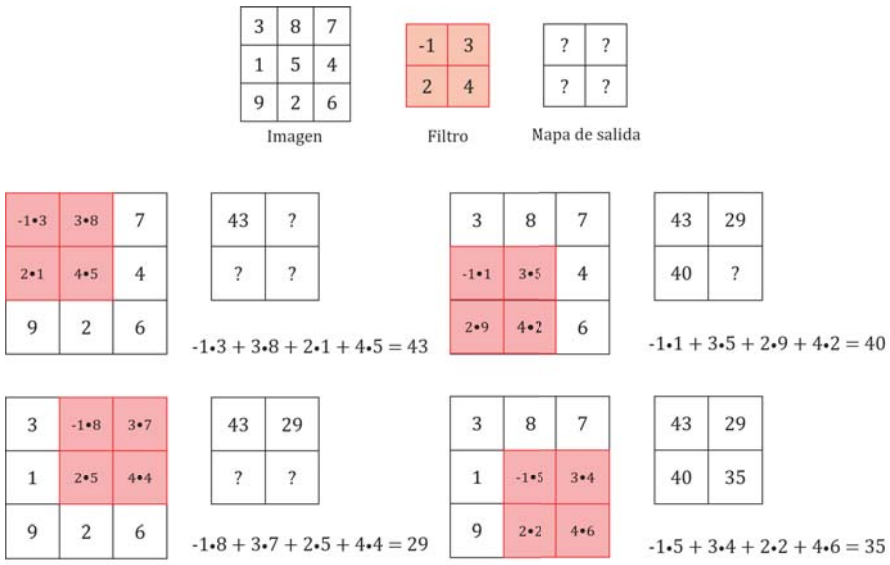


Figura 2.8: Ejemplo de un proceso de convolución sobre una imagen bidimensional. Un filtro de tamaño 2x2 recorre la imagen obteniendo un mapa de salida como resultado.

En el caso de trabajar con imágenes a color, al tener tres canales (RGB), es necesario considerar una tercera dimensión, pero el proceso de convolución resulta similar. La única variación es que ahora el filtro de convolución va a tener también tres canales.

2.4. Entrenamiento

Para que una red neuronal se adapte al problema para el que se ha diseñado y aprenda a llegar a la solución es necesario el proceso de entrenamiento. Las neuronas de la red actualizan los valores de sus pesos intentando minimizar el error de predicción.

Durante el proceso de entrenamiento también se realiza un ajuste de los hiperparámetros hasta llegar a los valores óptimos para el problema a resolver.

Los elementos que se necesitan para el entrenamiento de una red son los siguientes:

- **"Dataset":** es el conjunto de datos que se va a usar como entrada a la red. Tiene que tener un número elevado de ejemplos similares al problema que se desea resolver.

  Es necesario disponer de dos tipos de datasets, uno para entrenamiento y otro para evaluación. El de entrenamiento se usa en el proceso de entrenamiento de la red. Este configura los valores de los pesos de las neuronas. Con el conjunto de evaluación se testea la red para comprobar su desempeño. Esto es necesario ya

que en ocasiones el entrenamiento puede provocar el fenómeno de sobreajuste [2.5] sobre los datos de entrenamiento, este efecto se describe más adelante. Un dataset puede estar etiquetado o no, entendiendo como etiquetado que cada imagen lleve asociada una etiqueta de texto indicando el contenido de la imagen. Los conjuntos de datos etiquetados son muy prácticos para tareas de reconocimiento de lugares u objetos. En estas tareas se entrena la red con el objetivo de luego introducir a la red un objeto o lugar no etiquetado y que la red realice el etiquetado. En este trabajo, dado que no se trata de un proceso de clasificación, el dataset no va a ser necesario que esté etiquetado.

- **Función de coste:** conocida en inglés como *loss*. Es la función que obtiene el valor del error entre la salida de la red y la salida que se esperaba. Es la función a optimizar durante el proceso de entrenamiento.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Coste}(h_\theta(x^{(i)}), y^{(i)}) \tag{2.2}$$

  La función de coste tiene dos términos, uno de ellos es el de error y el término de regularización. El término de regularización es un hiperparámetro y es el encargado de limitar la evolución de la red, evitando así el sobreajuste [2.5].

  Al obtener la salida de la red durante el entrenamiento se determina el valor de la función de coste y posteriormente se usan técnicas como propagación hacia atrás (*backpropagation* en inglés) [2] para actualizar los parámetros de la red. El proceso se va realizando de forma iterativa hasta alcanzar el valor óptimo a la salida de la red.

  En propagación hacia atrás el término de regularización va multiplicado por un término conocido como *learning rate* que regula la velocidad a la que la red evoluciona. Un *learning rate* bajo puede provocar un entrenamiento muy lento o que la red caiga en un mínimo local, pero sin embargo, uno muy alto, puede provocar que nunca se alcance un buen mínimo local.

- **Número de épocas:** es el número de veces que durante el entrenamiento se recorre el conjunto de datos de entrenamiento completo.

- ***Batch size:*** define el número de ejemplos del conjunto completo de datos que se van a propagar a lo largo de la red. Por ejemplo, si se tiene un conjunto de datos de entrenamiento 1000 ejemplos, se fija un *batch size* de tamaño 100 y una época de entrenamiento, la red va a realizar el proceso de propagación hacia atrás 10 veces hasta recorrer todo el conjunto de entrenamiento. Una de las principales ventajas de usar *batch size* es la menor necesidad de memoria durante el proceso de propagación hacia atrás.

2.5. Sobreajuste

El sobreajuste o más conocido como sobreajuste es el efecto que se produce al sobreentrenar una red neuronal. Cuando se produce, la red encuentra una solución particular al problema a través de los datos del conjunto de datos entrenamiento y no es capaz de generalizar el problema. De esta forma, al introducir los datos de evaluación, la red no responde debidamente.

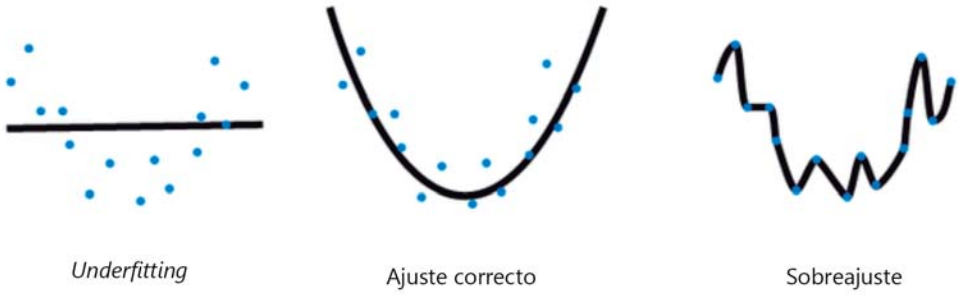


Figura 2.9: A la izquierda un caso de poco entrenamiento, la red no es capaz de comprender la parábola que forman los puntos. En la parte central un caso de ajuste correcto, la red une los puntos formando una parábola. En la parte derecha un caso claro de sobreajuste, la red une todos los puntos realizando una forma compleja.

Uno de los casos más sencillos de comprender de sobreajuste es el de la Figura 2.9. En este ejemplo se disponen unos puntos colocados formando una parábola. La red realiza un sobreajuste adaptando los puntos a un polinomio de elevado grado.

3. Marcado de agua con redes neuronales profundas

Este capítulo se van a mostrar las distintas alternativas usadas para la resolución del problema propuesto.

En primer lugar, se van a introducir los conjuntos de datos utilizados. Luego se van a explicar y determinar los hiperparámetros definidos para resolver el problema. Por último, se van a definir las arquitecturas de red utilizadas.

3.1. Conjuntos de datos utilizados

En prácticamente todos los casos que implican investigación se comienza con la resolución de problemas simplificados. Esto se hace con el objetivo de comprender el funcionamiento de los sistemas para, de forma posterior, llevar el problema a un nivel superior en el que la complejidad aumenta. En el campo del aprendizaje automático, un parámetro que va muy asociado a la complejidad y rapidez de la resolución de los problemas es el conjunto de datos (más conocido por el término inglés *dataset*) utilizado para afrontar el problema.

Por lo expuesto anteriormente, se han usado dos datasets de uso libre, uno más complejo y otro más sencillo, que son los siguientes:

3.1.1. MNIST

Es uno de los conjuntos de datos más usado por los usuarios que comienzan a realizar pruebas con herramientas de aprendizaje automático debido a la facilidad de compresión de los problemas que ofrece su uso.

Está formado por 70.000 imágenes de números del cero al nueve escritos a mano en color blanco sobre fondo negro, de forma que sólo tienen un único canal, a diferencia de las imágenes en color que están formadas por tres.

De las 70.000 imágenes de las que está compuesto, 60.000 imágenes pertenecen a un conjunto de entrenamiento y otras 10.000 pertenecen al conjunto de evaluación. Algunos

ejemplos son los de la Figura 3.1.

Figura 3.1: Ejemplo de alguna de las imágenes del conjunto de datos MNIST.

3.1.2. Image-Net

Es una base de datos de imágenes de uso libre para investigación [3]. Está compuesta de 1.300.000 imágenes, de las cuales aproximadamente 1.250.000 pertenecen al conjunto de datos de entrenamiento, y las otras 50.000 pertenecen al conjunto de datos de validación. Algunos ejemplos son los de la Figura 3.2.

Figura 3.2: Ejemplo de alguna de las imágenes del conjunto de datos Image-Net.

Está disponible para descarga en múltiples tamaños de imagen. La necesidad de recursos de computación ha provocado que para este trabajo se haya optado por la descarga de las imágenes en resolución 32x32.

3.2. Hiperparámetros

Un factor muy importante a la hora de definir una red son los hiperparámetros. Para la decisión de cómo determinar los óptimos para la resolución del problema planteado se han realizado distintos análisis de sensibilidad.

Los hiperparámetros más críticos que se han ajustado, en lo conocido por el término inglés como *fine tuning* (ajuste fino), son los siguientes:

- Función de coste (más conocida en inglés como *loss*)
- Algoritmo de optimización (para encontrar el mínimo de la función de pérdida)

- *Learning rate*
- *Batch size*
- Número de épocas

Una de las mayores complejidades de este problema radica en la función de coste que se ha utilizado, la cual es una ponderación de dos distintas. Por un lado, como primer término, se determina una pérdida entre la imagen original y la imagen marcada. Por otro lado, el segundo término, es la pérdida entre la marca de agua original y la marca de agua recuperada.

$$L = L_{img} + L_{wmk} * \alpha, \tag{3.1}$$

donde L es el valor de la función de coste final, L_{img} es el valor de la función de coste para la imagen marcada, L_{wmk} es el valor de la función de coste para la marca recuperada y α es el valor de la ponderación entre las dos funciones de coste.

La función utilizada para cada una de las pérdidas es la función del error cuadrático medio, más comúnmente conocida como MSE (*mean squared error*), cuya formula es la siguiente:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(h_\theta - y^{(i)})^2, \tag{3.2}$$

siendo N el número total de imágenes usadas en la validación, h_θ la imagen de salida de la predicción de la red e y^i la imagen original.

Como algoritmos de optimización para encontrar el mínimo de la función de pérdidas se han usado el método del descenso del gradiente, fijando manualmente varios *learning rate*, y la función de optimización *AdaDelta*, la cual fue desarrollada por D.Zeiler [4], que utiliza un método de ajuste adaptativo del *learning rate*. Los resultados de estos experimentos se pueden encontrar en la sección 4.3.

Para determinar el *batch size* óptimo se ha procedido de forma similar a lo explicado. Se ha realizado un estudio paramétrico, determinando cuál era el más óptimo para la resolución del problema. Se han hecho pruebas con *batch size* de tamaño 50, 100 y 256. Se concluye en que no es un factor determinante en los resultados para las métricas y que el mejor de los testeados es 100.

El número de épocas se ha establecido realizando una determinación de la convergencia del modelo. Se concluye en que un total de 100 épocas son suficientes para obtener convergencia, siendo más épocas innecesarias frente al coste computacional que hay que hacer frente para obtener mejores resultados.

3.3. Arquitecturas de red

A lo largo del trabajo se ha trabajado con dos planteamientos distintos. Se ha distinguido entre redes que se entrenan usando una única marca de agua y redes que se entrenan usando múltiples marcas de agua. El primer caso se podría ejemplificar con un fotógrafo que usa una única marca de agua para todas sus imágenes. El segundo caso se correspondería con un medio de comunicación en el que cada fotógrafo marca sus imágenes de manera particular.

3.3.1. Marcado de agua con una única marca

El planteamiento general de la red es haciendo uso de dos entradas y obteniendo dos salidas. Como entradas se tienen las imágenes sobre las que se quiere hacer el marcado y la marca de agua única que se va a usar para realizar el marcado. Como salidas se tienen la imagen marcada y la marca recuperada.

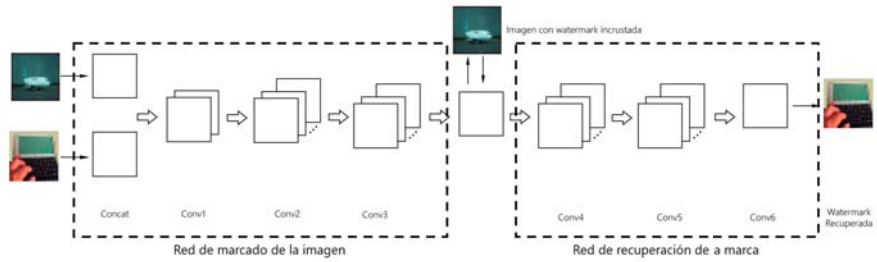
Figura 3.3: Esquema del marcado de agua con una única marca.

Como se puede observar en el esquema (Figura 3.3) las entradas se producen de forma simultánea, pero las salidas se producen en distintos instantes. Es decir, la imagen a marcar y la marca se unen al principio de la red, pero las salidas se obtienen tras dos procesos distintos. La imagen marcada se obtiene tras pasar por una red de marcado y la marca recuperada se obtiene tras pasar la imagen marcada por una red de recuperación de la marca. La primera de las salidas tiene que parecerse lo máximo posible a la imagen original. La segunda se ha de parecer lo máximo a la marca de agua.

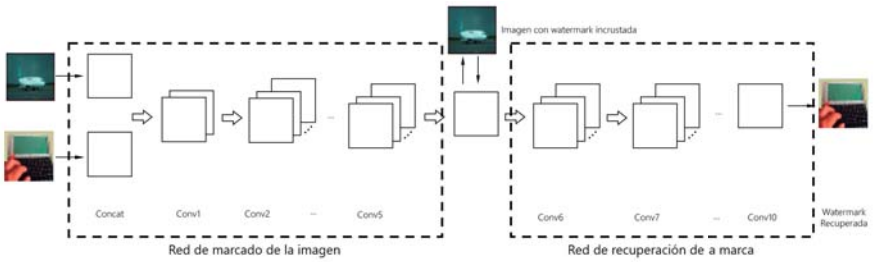
Uno de los mayores problemas a evitar en este tipo de marcado es el sobreajuste (Sección 2.5). El uso de una única marca de agua puede provocar que la red se "aprenda" que la salida de la marca de agua es siempre la misma. De esta manera, a pesar de que se intentase marcar con otra marca de agua, la red siempre devolvería como salida la marca que se ha aprendido.

Desde el punto de vista técnico, la red se ha planteado como una consecución de capas convolucionales (Sección 2.3). El principal factor a determinar es el número de capas, ya que un número reducido conduce a malos resultados y un número excesivo conduce a un alto coste computacional. La decisión de usar capas convolucionales es debido a su propiedad para encontrar patrones comunes con facilidad, reduciendo a la vez de forma considerable el coste computacional.

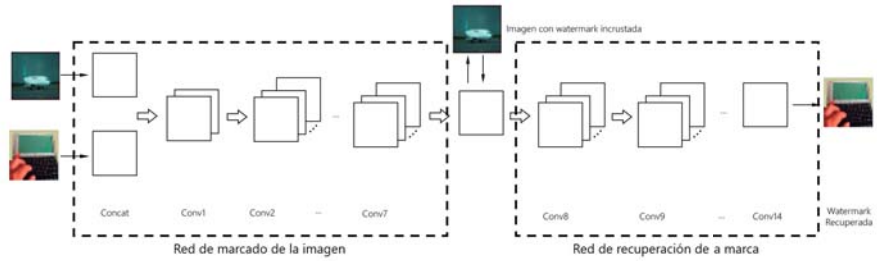
Para resolver este problema, se han hecho test con cuatro arquitecturas distintas (Figura 3.4):



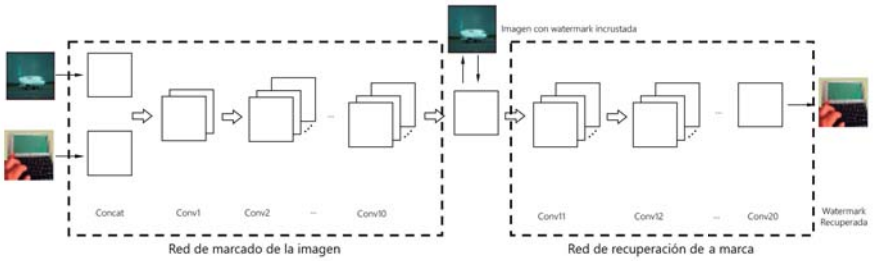
(a) Tres capas por red



(b) Cinco capas por red



(c) Siete capas por red



(d) Diez capas por red

Figura 3.4: Muestra de las distintas arquitecturas usadas para el marcado con una única marca de agua.

Como se observa, se han realizado pruebas con sucesiones de tres, cinco, siete y diez capas convolucionales para cada una de las redes, la de marcado y la de recuperación

de la marca. De todos los modelos, se ha determinado que el que mejor funciona en relación coste computacional/resultados es el uso de siete capas para cada una de las redes. Haciendo uso de tres y cinco no se conseguían resultados aceptables, y haciendo uso de diez no se observaba una mejoría. De esta forma, se ha decidido usar siete capas para la obtención de resultados de este apartado.

Esta arquitectura se ha usado indistintamente para el conjunto de datos MNIST e Image-Net.

3.3.2. Marcado de agua con múltiples marcas

El planteamiento que se hace es similar al del uso de una única marca. Se van a introducir a la red dos entradas de forma simultánea y se van a obtener dos salidas en distintos puntos. La complejidad de este modelo reside en hacer un proceso de entrenamiento que sea robusto, ya que las marcas a introducir pueden ser muy variadas.

Los resultados experimentales previos a obtener resultados definitivos probaron que el conjunto de datos a utilizar en este modelo es muy influyente. De esta manera, ha sido necesario diseñar dos arquitecturas distintas, una para el conjunto de datos MNIST y otra para el conjunto de datos Image-Net.

Para el conjunto MNIST se ha mantenido exactamente la misma arquitectura que en el caso de una única marca de agua. Se ha variado únicamente la entrada de las marcas de agua, que ahora son múltiples. Se puede observar en la Figura 3.5:

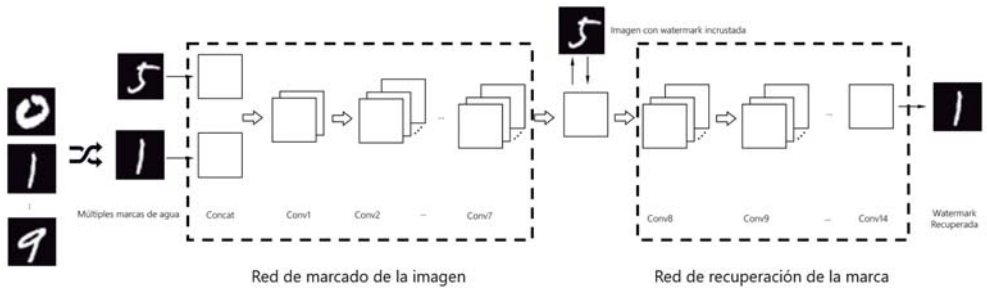


Figura 3.5: Red neuronal planteada para el marcado con múltiples marcas de agua para el conjunto de datos MNIST. Una imagen a marcar (el cinco) y una marca de agua del conjunto global de marcas (el uno) son entradas de la red. En primer lugar, atraviesan el proceso de marcado de la imagen, obteniendo así la imagen con la marca de agua incrustada. Luego, en un proceso acoplado, se obtiene la marca de agua recuperada con una red similar.

Para el conjunto Image-Net se ha diseñado una red con un planteamiento similar de entradas y salidas pero con un número reducido de capas convolucionales, añadiendo una capa de *dropout* y una capa completamente conectada. Las capas de *dropout* son capas

en las que se apagan de forma aleatoria un porcentaje de las neuronas, estipulado por el diseñador de la red. El principal objetivo de conseguir que la red no realice el fenómeno del sobreajuste (Sección 2.5). Con estas capas se consigue incomodar a la red y se obliga a forzar otras neuronas en el proceso de aprendizaje. Otra de las ventajas que tienen es una reducción en el coste computacional.

El diseño de esta arquitectura para el conjunto Image-Net es el que se ve en la Figura 3.6:

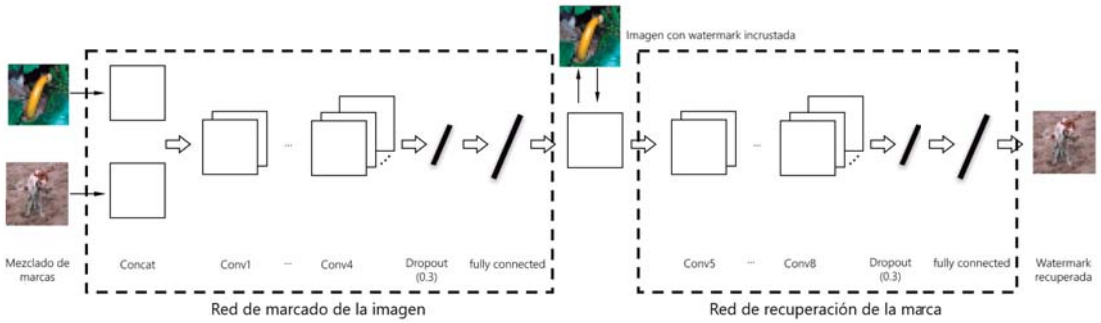


Figura 3.6: Red neuronal planteada para el marcado con múltiples marcas de agua para el conjunto de datos Image-Net.

4. Resultados experimentales

En esta sección se van a exponer los resultados experimentales obtenidos de las distintas arquitecturas previamente explicadas.

Se va a explicar en primer lugar la metodología seguida (Sección 4.1) para la elaboración de los modelos. En segundo lugar, qué son las métricas y cuales han sido las utilizadas para la validación (Sección 4.2). Por último, se van a mostrar los resultados para cada uno de los casos planteados (Sección 4.4).

4.1. Metodología

Para cada uno de los experimentos se ha optado por realizar un entrenamiento progresivo de las redes. Cada red se ha entrenado con un número reducido de épocas, analizando los resultados al final de cada periodo de entrenamiento. Después de cada periodo se decide si continuar con el entrenamiento o deshechar el modelo por falta de resultados coherentes.

En los primeros modelos, también se ha optado por realizar análisis paramétricos exhaustivos. El objetivo de estos es determinar qué valores para los hiperparámetros y qué funciones eran las óptimas a usar.

Finalmente, la evaluación de un modelo como correcto o no, se ha hecho haciendo uso de las métricas.

4.2. Métricas

En ocasiones, la simple inspección visual de los resultados de una imagen puede ser de utilidad para determinar si un modelo ha funcionado de manera correcta o hay que despreciarlo debido a que no se obtienen los resultados deseados.

La inspección visual puede resultar muy útil en la parte de determinar si un modelo no es correcto. Si una imagen no concuerda o distorsiona de forma relevante frente a la

original, la determinación de que el modelo es erróneo se detecta fácilmente. Sin embargo, en el lado opuesto, cuando pensamos que un modelo es correcto, nuestra visión se puede ver alterada. Esta alteración se puede deber, por ejemplo, por la calidad de la pantalla en la que estamos realizando la visualización o simplemente por estar pasando por alto detalles en el color que son de importancia.

Como solución a este problema de validación existe lo que se conocen como métricas. Las métricas son cálculos matemáticos que permiten comparar las imágenes de salida de las redes con las imágenes originales. Se obtiene como resultado un número que determina la calidad de los resultados.

Para este trabajo, basándose en los resultados de las métricas expuestas en [5], se ha decidido usar tres métricas para la comparación de resultados que son las siguientes:

4.2.1. PSNR: Peak Signal Noise Ratio

Determina la relación entre la potencia (valores) máximos de una señal y el ruido que puede alterar la representación de esta señal. Debido al alto rango de valores que se pueden encontrar en una señal esta se suele calcular en medida logarítmica y, por lo tanto, se expresa en decibelios.

En concreto, el PSNR se define como:

$$\text{PSNR} = 10\log_{10}(\frac{R}{MSE}), \tag{4.1}$$

donde R es la máxima fluctuación del tipo de datos de las imágenes, que en este caso R=1 y MSE es el error cuadrático medio, que se calcula de forma similar a la función de coste que se ha utilizado a lo largo de la realización del trabajo.

Es una métrica muy usada para comprobar la calidad de compresión de imágenes a formato JPEG.

Cuanto mayor sea el valor de esta, mejor será el resultado obtenido.

4.2.2. MSE: Mean Squared Error

Cuantifica y mide la calidad entre imágenes haciendo uso del error cuadrático medio de una señal respecto a la otra. Al ser un error cuadrático penaliza de forma considerable las grandes diferencias.

Por ello, para el caso estudiado de imágenes con tamaño pequeño y por lo tanto con pocos elementos, 784 para las imágenes del conjunto MNIST y 3072 para el conjunto Image-Net, un error considerable en cualquiera de los píxeles va a resultar muy significativo y la métrica se va a ver muy penalizada.

Cuanto menor valor mejor es la calidad de la imagen marcada.

4.2.3. SSIM: Structural Similarity Index

Es una de las métricas más recientemente propuestas e intenta combinar y mejorar los resultados de las métricas PSNR y MSE.

Mientras que las métricas comentadas anteriormente miden errores absolutos, a través de esta métrica se incorporan términos más abstractos que comparan otros factores más subjetivos como son la iluminación y el contraste. De esta manera, esta métrica no sólo evalúa las diferencias entre píxeles de una imagen a otra, sino que realizan un análisis global de la imagen.

La formulación de esta métrica es la siguiente:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4.2}$$

donde:

- μ_x es la media de los valores en el eje x de la imagen
- μ_y es la media de los valores en el eje y de la imagen
- σ_x es la varianza de los valores en el eje x de la imagen
- σ_y es la varianza de los valores en el eje x de la imagen
- σ_{xy} es la covarianza de los valores en el eje x e y de la imagen
- C_1 y C_2 son dos constantes

A lo largo de toda la exposición de los resultados se van a comparar seis métricas distintas que son las siguientes:

- PSNR entre la imagen original y la imagen marcada.
- PSNR entre la marca original y la marca recuperada.
- MSE entre la imagen original y la imagen marcada.
- MSE entre la marca original y la marca recuperada.
- SSIM entre la imagen original y la imagen marcada.
- SSIM entre la marca original y la marca recuperada.

4.3. Estudios paramétricos

Con el objetivo de determinar los hiperparámetros óptimos a usar a lo largo del trabajo se han realizado distintos estudios paramétricos, que se encuentran completos en el Anexo.

Para la realización de los estudios paramétricos se ha optado por evaluar el modelo con el conjunto de datos MNIST. Esto es debido a la rapidez en el cálculo y la sencillez para extrapolarlo al resto de modelos.

El **primer estudio** realizado tiene como objetivo determinar cuál es la ponderación óptima entre las funciones de coste. A continuación, se muestran los resultados obtenidos que van a determinar la ponderación de la función de coste que se va a usar durante todo el trabajo.

PSNR (Figura 4.1):

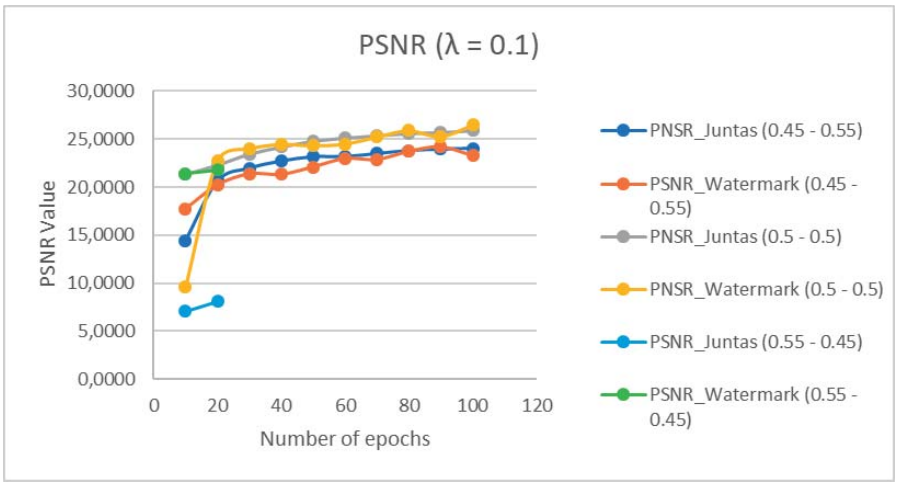


Figura 4.1: PSNR para distintos valores de la ponderación.

Se observa que la ponderación que mejor funciona es en la que se ponderan ambas funciones de coste de manera similar (0.5 – 0.5), seguido del modelo en el que se pondera más la perdida correspondiente a la marca de agua (0.45 – 0.55) y por último, y con peores resultados, el modelo que pondera más la perdida correspondiente a la imagen original (0.55 – 0.45).

MSE (Figura 4.2):

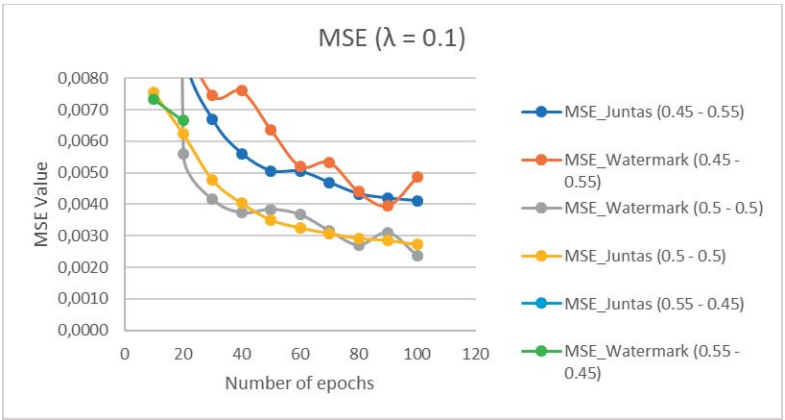


Figura 4.2: MSE para distintos valores de la ponderación.

Se observa que en la métrica MSE también se obtienen los mejores resultados en el modelo que se ponderan ambas funciones de coste de la misma manera. Los modelos de ponderación 0.55 – 0.45 se desprecian desde el principio dado que no ofrecen buenos resultados.

SSIM (Figura 4.3):

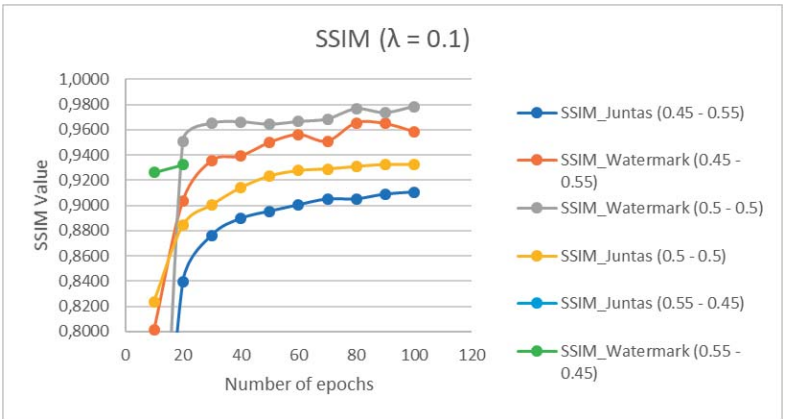


Figura 4.3: SSIM para distintos valores de la ponderación.

Se vuelve a comprobar que el modelo que mejor funciona es el que mantiene la misma ponderación para ambas funciones de coste, muy seguido para esta métrica por el modelo de ponderación 0.45 – 0.55.

Dados los resultados obtenidos en este estudio se decide trabajar con un modelo de ponderación similar para ambas funciones de coste.

El segundo estudio, partiendo de la ponderación determinada en el estudio previo, tiene el objetivo de determinar cuál de los optimizadores de la función de coste es mejor. Se va a elegir entre el método del descenso del gradiente y el optimizador Ada Delta. En el caso de usar el método del descenso del gradiente, hay que determinar que *learning rate* es el más adecuado, variando entre 0.1 y 0.01, que son valores habituales. Recordar que el optimizador AdaDelta usa un *learning rate* que se va adaptando de forma automática.

Los resultados traducidos en métricas de los experimentos, en función del número de épocas, son los siguientes:

PSNR (Figura 4.4):

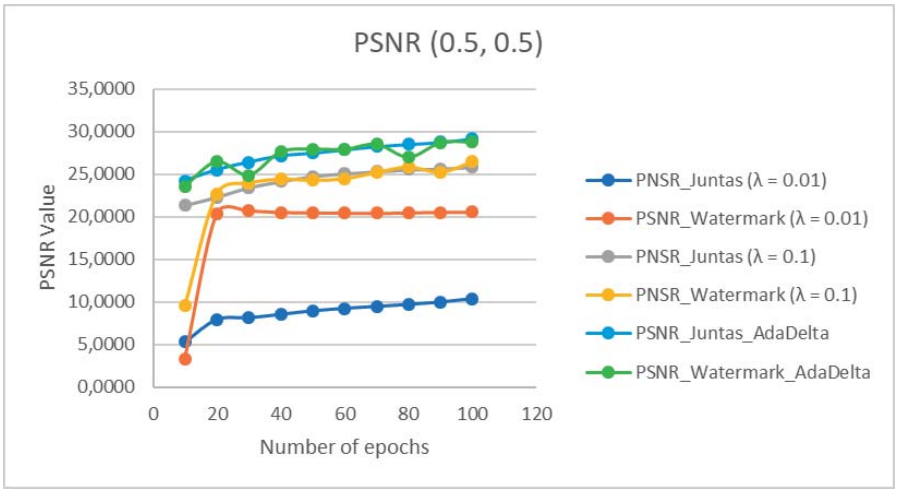
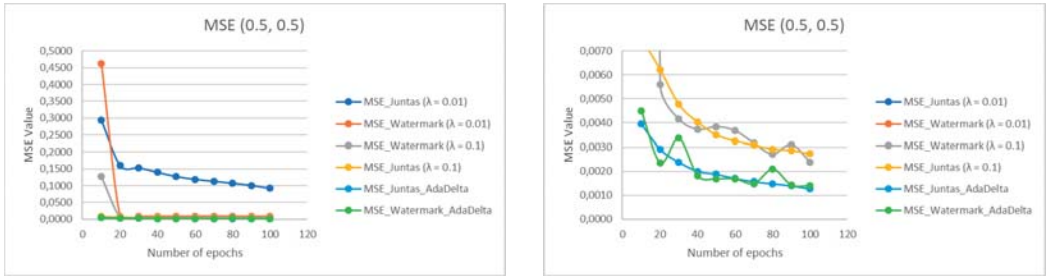


Figura 4.4: PSNR para distintos optimizadores.

Se puede observar en la Figura 4.4 que tanto para el caso del optimizador del descenso del gradiente, el *learning rate* más apropiado es el de 0.1. Sin embargo, también se puede observar que el optimizador Ada Delta obtiene mejores resultados y funciona de manera más eficiente, ya que consigue los mejores resultados de forma mucho más rápida.

MSE (Figura 4.5):



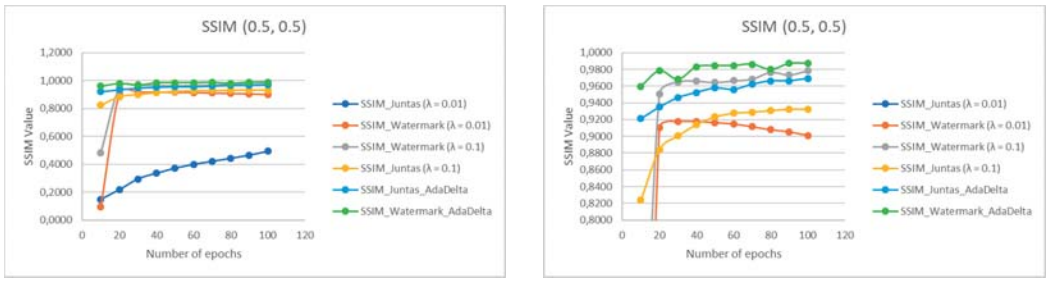
(a) Vista global (b) Vista focalizada en los límites del 0

Figura 4.5: MSE para distintos optimizadores.

Se observa que a simple vista es una métrica a partir de la cual podemos obtener pocas conclusiones. Todas las combinaciones, a excepción del caso con método del descenso del gradiente con *learning rate* de 0.01, convergen rápidamente hacia cero.

Sin embargo, si nos acercamos a los límites del cero para ver cuál de los optimizadores funciona mejor, se vuelve a comprobar que es el optimizado *AdaDelta* el que vuelve a obtener los mejores resultados para esta métrica.

SSIM (Figura 4.6):



(a) Vista global (b) Vista focalizada en los límites del 1

Figura 4.6: SSIM para distintos optimizadores.

Se vuelven a obtener conclusiones similares a las obtenidas para la métrica MSE. La red para un *learning rate* de 0.01 avanza de forma lenta y es necesario acercarse a los límites del 1 para ver cuál es el optimizador que mejor funciona.

Vistos los resultados, se decide que el mejor optimizador es el *AdaDelta* y se va a usar a lo largo de todo el trabajo.

Los hiperparámetros que se van a usar a lo largo (Tabla 4.1) del trabajo son los

siguientes:

Parámetro	**Elección**
Ponderación de función de coste	0.5 - 0.5
Función de coste	MSE
Optimizador	Ada Delta
Batch Size	100

Tabla 4.1: Hiperparámetros a usar a lo largo del trabajo

4.4. Resultados de los distintos modelos

4.4.1. MNIST – Única marca de agua combinada con múltiples imágenes

Para este modelo se ha tomado como marca de agua única el número dos. Se ha combinado con todos los números del cero al nueve haciendo uso de la red neuronal propuesta.

A continuación (Figura 4.7), se muestra un ejemplo de algunos de los resultados obtenidos haciendo uso de este modelo:

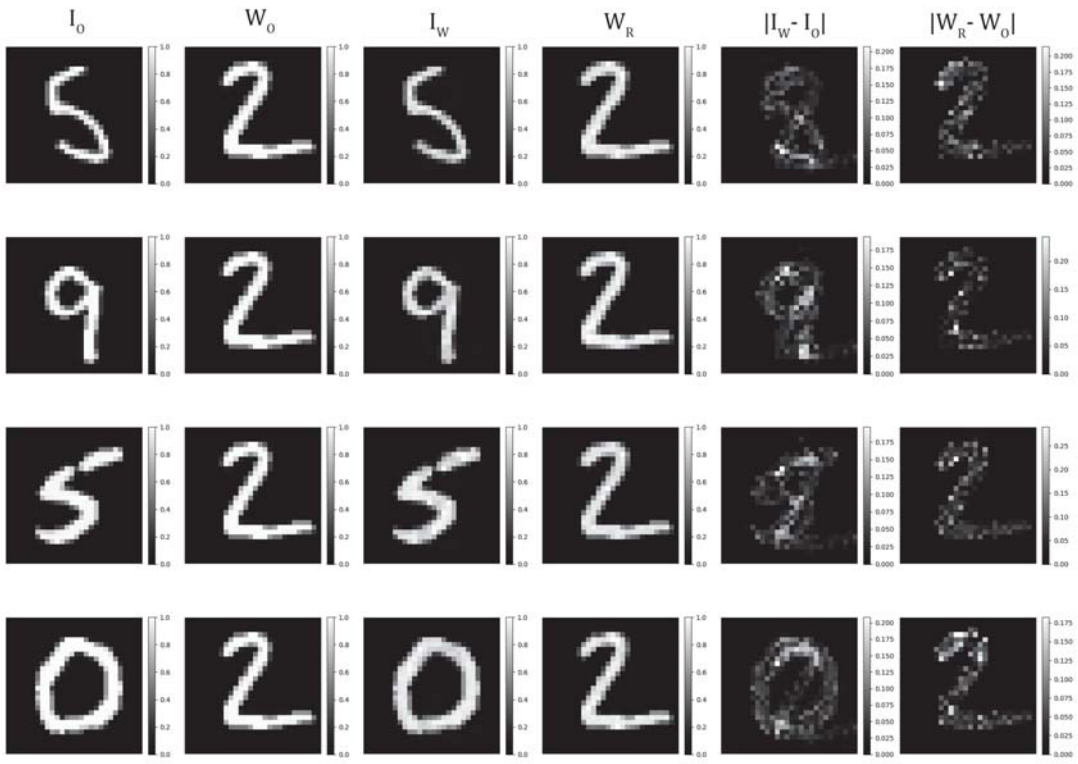


Figura 4.7: Resultados con una única marca de agua para el conjunto MNIST. De izquierda a derecha: en primer lugar, la imagen sobre la que realizar el marcado. En segundo lugar, la marca de agua a usar. En tercer lugar, la imagen con la marca de agua incrustada. En cuarto lugar, la marca recuperada de la imagen marcada. En quinto lugar, la diferencia entre la imagen original y la imagen marcada. En último lugar la diferencia entre la marca original y la marca recuperada.

Se puede ver que el objetivo se ha conseguido ya que la marca de agua se encuentra

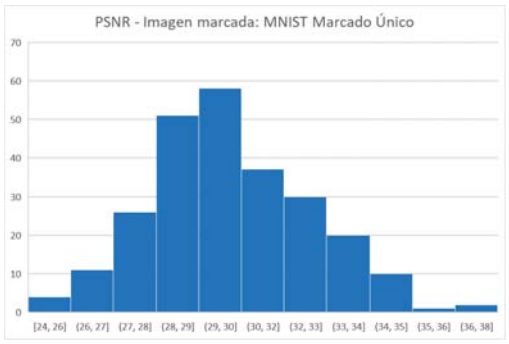
oculta. Sin embargo, al observar la diferencia entre ambas imágenes se puede ver que la marca de agua sí se encuentra presente.

Los valores de las métricas obtenidos (Tabla 4.2) para este modelo son los siguientes:

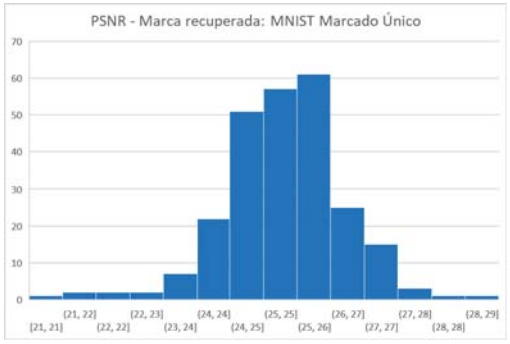
	I_W	W_R
PSNR	29.19 dB	28.81 dB
MSE	0.0013	0.0014
SSIM	0.9691	0.9874

Tabla 4.2: Métricas para una única marca de agua con el conjunto de datos MNIST

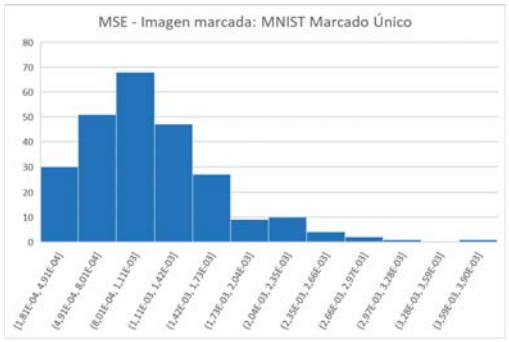
Si realizamos un análisis de cada métrica a través de un histograma podemos ver su distribución para cada una de las métricas (Figura 4.8):



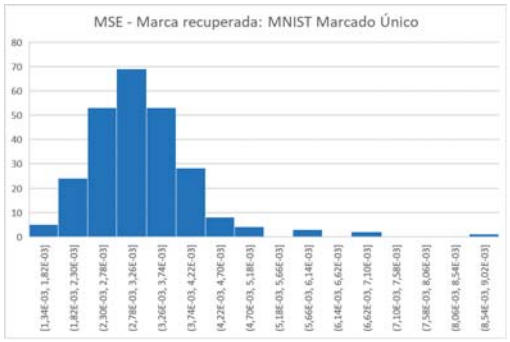
(a) PSNR - Imagen marcada



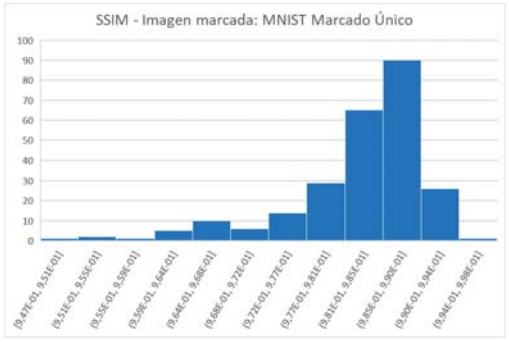
(b) PSNR - Marca recuperada



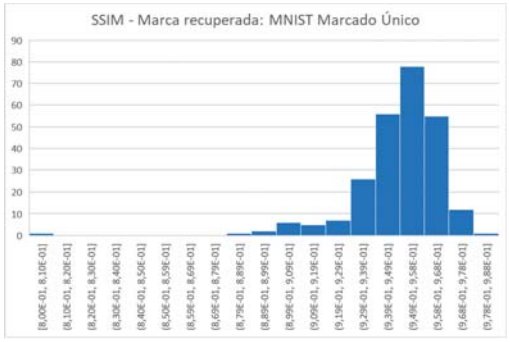
(c) MSE - Imagen marcada



(d) MSE - Marca recuperada



(e) SSIM - Imagen marcada



(f) SSIM - Marca recuperada

Figura 4.8: Histogramas para las distintas métricas para la imagen marcada (izquierda) y la marca recuperada (derecha).

Para compresión de imagen en formato JPEG, la métrica PSNR se encuentra en valores que oscilan entre 30 y 45 dB. Para el modelo propuesto se puede observar en la Figura 4.8 que tanto para la imagen marcada como para la marca recuperada podemos

encontrar valores dentro de este rango. Se toma como referencia la compresión JPEG ya que es un estándar muy usado y que degrada poco la calidad visual de las imágenes. El valor alcanzado indica que la degradación es pequeña, lo cual es bastante meritorio si se tiene en cuenta que se está incluyendo información dentro de la imagen.

4.4.2. MNIST – Múltiples marcas de agua combinadas con múltiples imágenes

En este modelo se mezcla cualquier marca de agua con cualquier imagen, haciendo uso de la red neuronal propuesta en apartados anteriores.

Algunos ejemplos de los resultados obtenidos con este modelo son los siguientes (Figura 4.9):

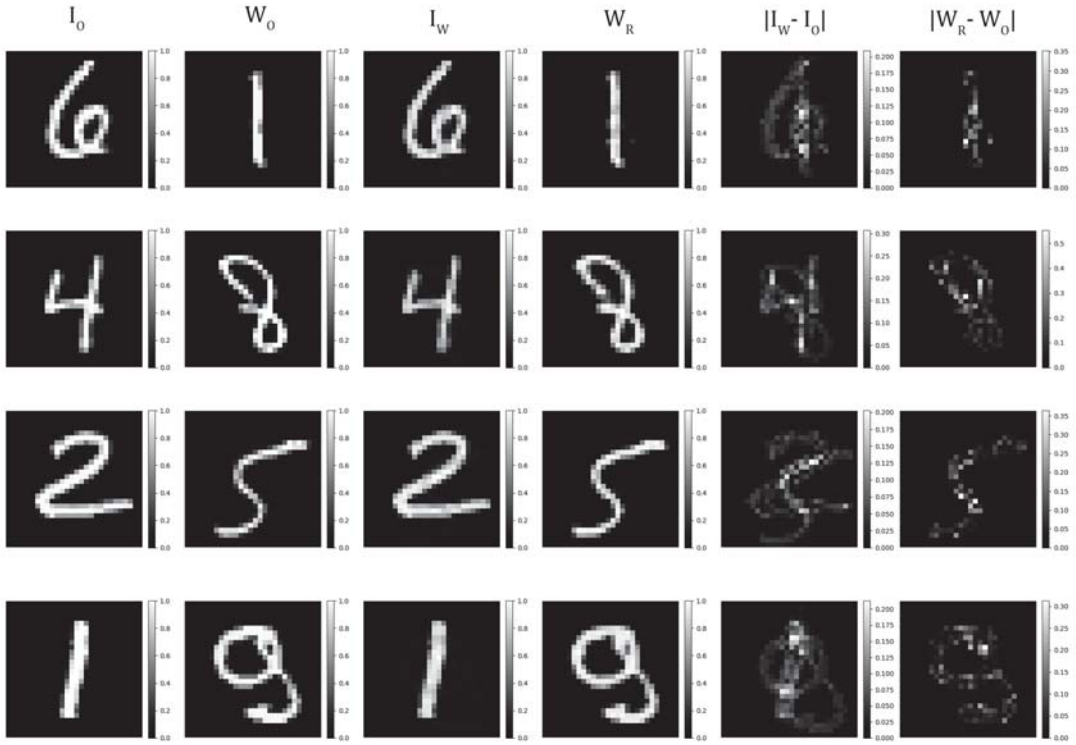


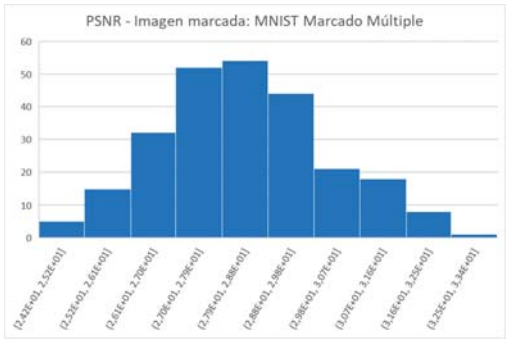
Figura 4.9: Resultados con múltiples marcas de agua para el conjunto MNIST. De izquierda a derecha: en primer lugar, la imagen sobre la que realizar el marcado. En segundo lugar, la marca de agua a usar. En tercer lugar, la imagen con la marca de agua incrustada. En cuarto lugar, la marca recuperada de la imagen marcada. En quinto lugar, la diferencia entre la imagen original y la imagen marcada. En último lugar la diferencia entre la marca original y la marca recuperada.

Las métricas obtenidas (Tabla 4.3) para este modelo son las siguientes:

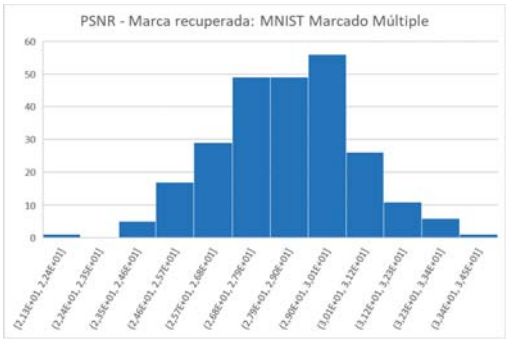
	I_W	W_R
PSNR	29.39 dB	28.53 dB
MSE	0.0016	0.0016
SSIM	0.9554	0.9832

Tabla 4.3: Métricas para múltiples marcas de agua con el conjunto de datos MNIST

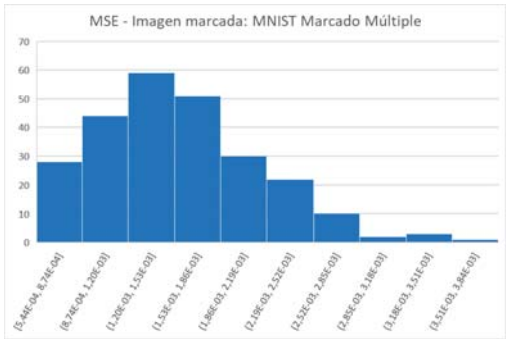
Si realizamos un análisis de cada métrica a través de un histograma podemos ver su distribución para cada una de las métricas (Figura 4.10):



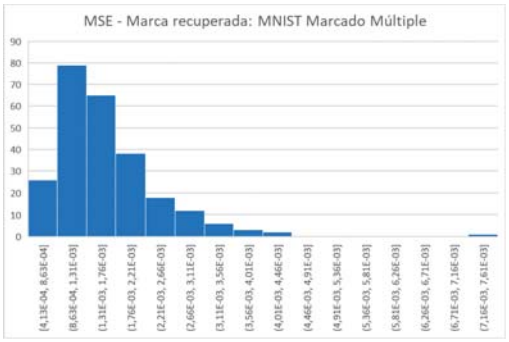
(a) PSNR - Imagen marcada



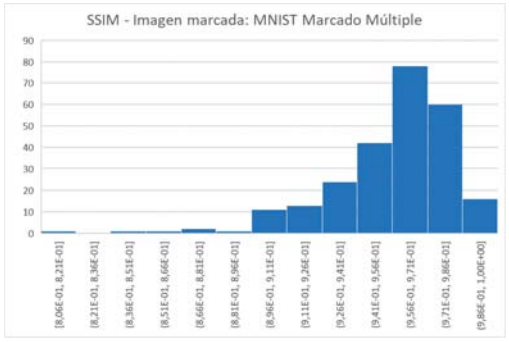
(b) PSNR - Marca recuperada



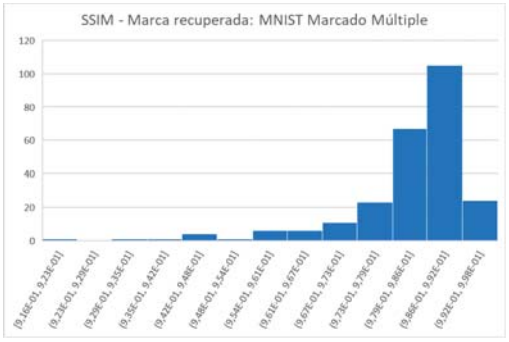
(c) MSE - Imagen marcada



(d) MSE - Marca recuperada



(e) SSIM - Imagen marcada



(f) SSIM - Marca recuperada

Figura 4.10: Histogramas para las distintas métricas para la imagen marcada (izquierda) y la marca recuperada (derecha).

Se puede observar (Figura 4.10) que se continúan teniendo valores dentro del rango de compresión JPEG, con una moda que oscila entre 27 y 30 dB.

A continuación, se muestran dos ejemplos en los que la red neuronal no se ha desempeñado de la manera esperada (Figura 4.11):

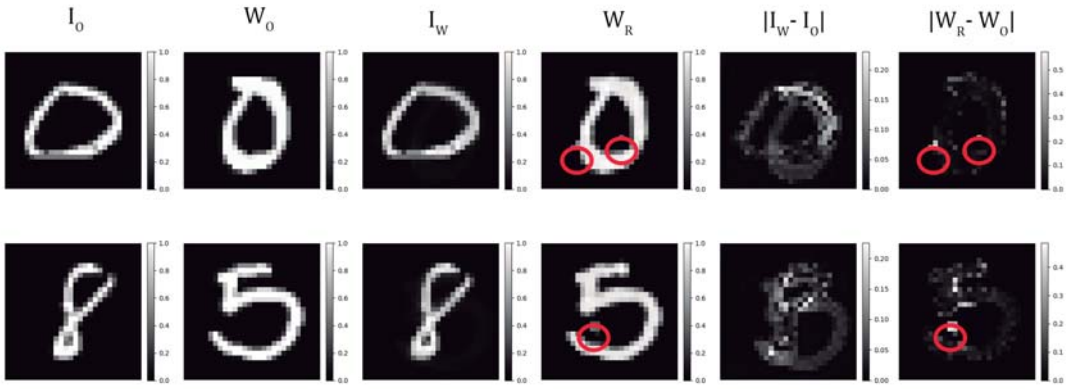


Figura 4.11: Resultados con error con múltiples marcas de agua para el conjunto MNIST. Se puede observar que la red comete errores insertando elementos no presentes en las marcas originales.

4.4.3. Image Net – Única marca de agua combinada con múltiples imágenes

Las redes neuronales no son algoritmos sencillos de adaptar, y en ocasiones, pueden generar problemas. En este caso se ha observado de forma clara el fenómeno del sobreajuste (Sección 2.5). Al entrenar este modelo de red se observado cómo la red no se ha desempeñado de forma esperada, sino que se ha adaptado a mostrar siempre la misma salida independientemente de la entrada dada.

Si se observa la Figura 4.12, la calculadora, que actúa como marca de agua, se recupera de forma perfecta:

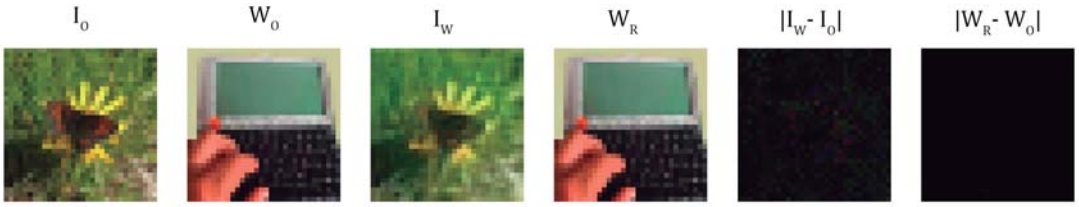


Figura 4.12: Ejemplo de marca de agua recuperada con sobreajuste.

Y fruto de ello, se obtienen unos valores en métricas (Tabla 4.4) muy elevados para la imagen de la marca de agua recuperada:

	I_W	W_R
PSNR	25.72 dB	49.44 dB
MSE	0.0095	3.44e-5
SSIM	0.9472	0.9999

Tabla 4.4: Métricas en caso de sobreajuste (Sección 2.5) con la marca de agua con la que se ha entrenado la red.

Cuando alteramos la red e intentamos que esconda otra marca que no es la de calculadora, que es con la que se ha entrenado la red, la red continua utilizando la calculadora como salida de marca de agua recuperada (Figura 4.13):

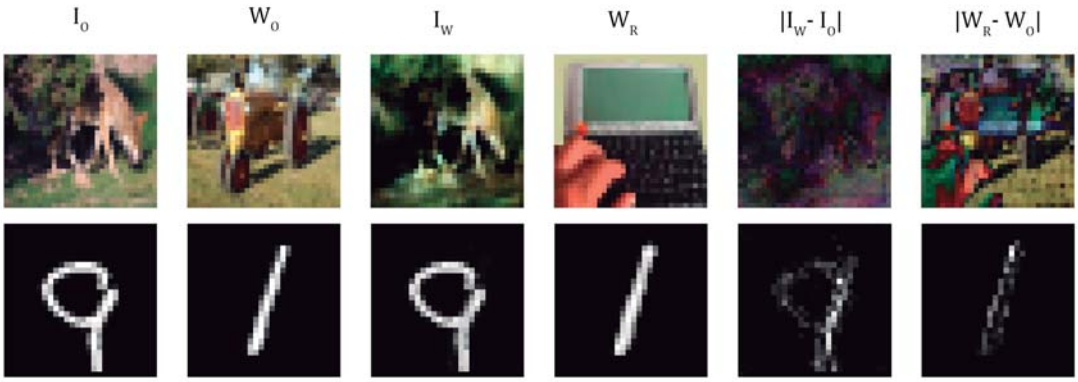


Figura 4.13: En la parte superior, marca de agua mal recuperada debido al sobreajuste (Sección 2.5) en el conjunto Image-Net. En la parte inferior, respuesta de la red de una única marca usando a una marca distinta a la usada en el entrenamiento.

Obteniendo así, resultados en métricas (Tabla 4.5) de bajo valor para la marca recuperada:

	I_W	W_R
PSNR	14.42 dB	9.81 dB
MSE	0.1227	0.3129
SSIM	0.6827	0.0169

Tabla 4.5: Métricas en caso de sobreajuste (Sección 2.5) con una marca de agua distinta a la del entrenamiento.

4.4.4. Image Net – Múltiples marcas de agua combinadas con múltiples imágenes

Por último, se muestran los resultados (Figura 4.14) para el modelo de combinar todas las marcas de agua con todas las imágenes haciendo uso de la red neuronal planteada para este caso.

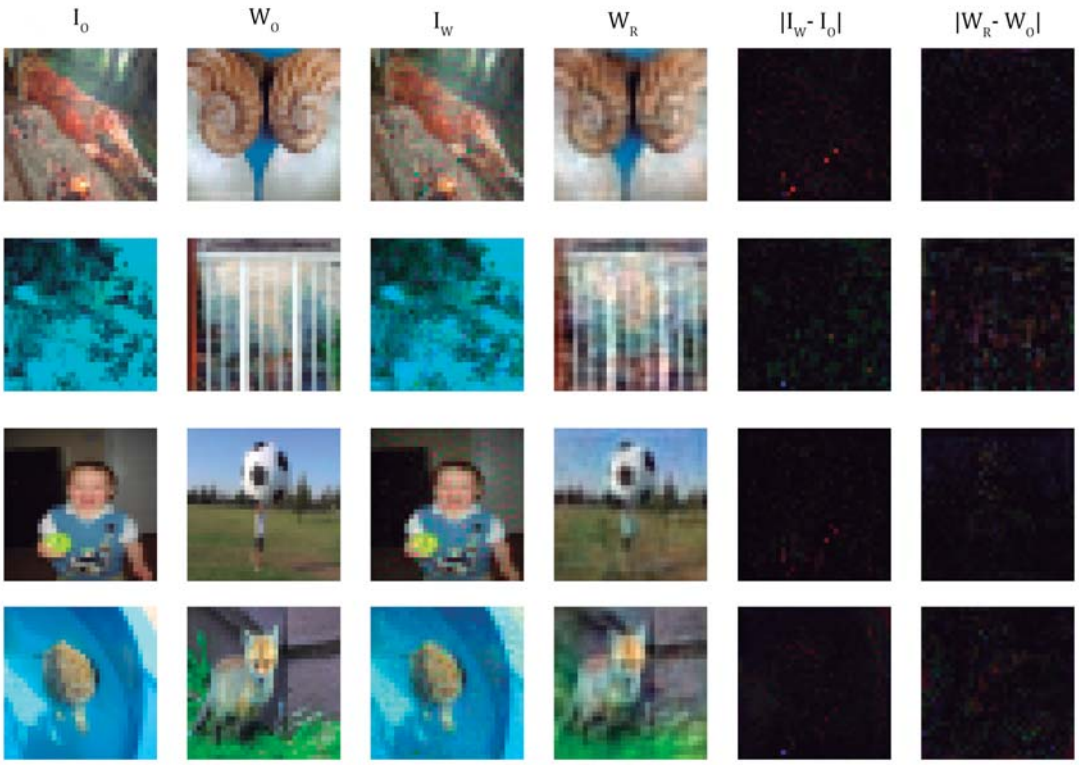


Figura 4.14: Múltiples marcas de agua combinadas con múltiples imágenes con el conjunto de datos Image-Net. De izquierda a derecha: en primer lugar, la imagen sobre la que realizar el marcado. En segundo lugar, la marca de agua a usar. En tercer lugar, la imagen con la marca de agua incrustada. En cuarto lugar, la marca recuperada de la imagen marcada. En quinto lugar, la diferencia entre la imagen original y la imagen marcada. En último lugar la diferencia entre la marca original y la marca recuperada.

Del que se obtienen las métricas de la Tabla 4.6.

Si realizamos un análisis de cada métrica a través de un histograma podemos ver su distribución para cada una de las métricas (Figura 4.15).

Para la métrica PSNR, la cual es la más concluyente, se puede observar que la mayoría de los valores se encuentran entre 26 y 28. Sin embargo, hay un sesgo alto, ya que

	I_W	W_R
PSNR	24.70 dB	23.78 dB
MSE	0.0192	0.0148
SSIM	0.9203	0.9026

Tabla 4.6: Métricas en el caso de múltiples marcas de agua combinadas con múltiples imágenes con el conjunto de datos Image-Net

considerando que es escala logarítmica, podemos encontrar casos con valores de 10 hasta casos con valor 32.

También hay en ocasiones en que la red neuronal no se desempeña de manera apropiada, algunos ejemplos son los de la Figura 4.16.

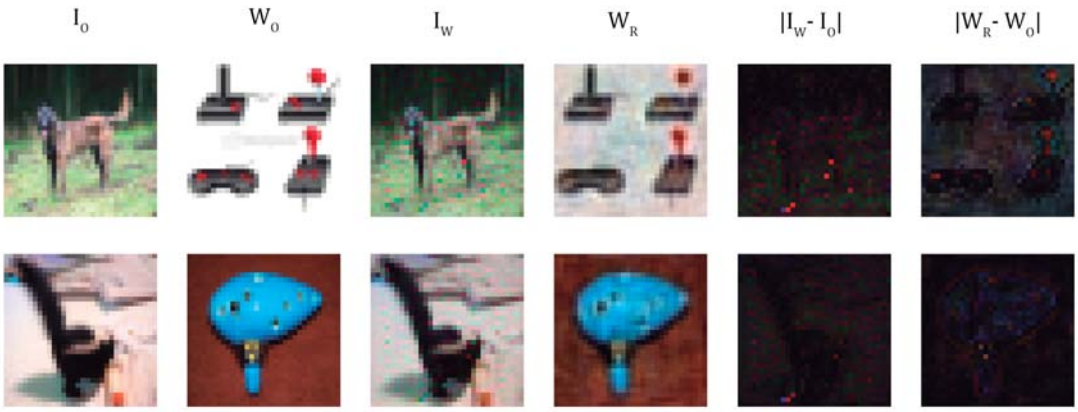
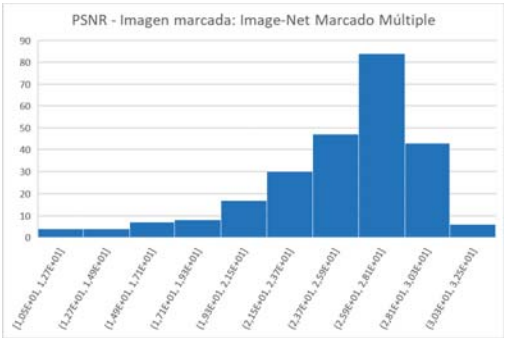
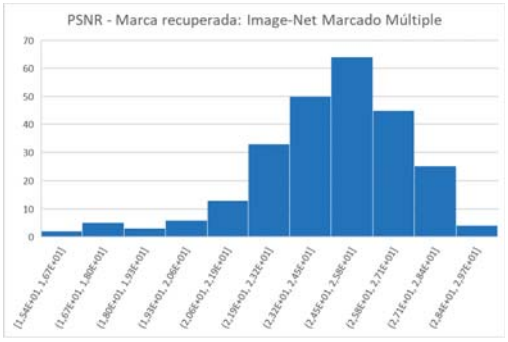


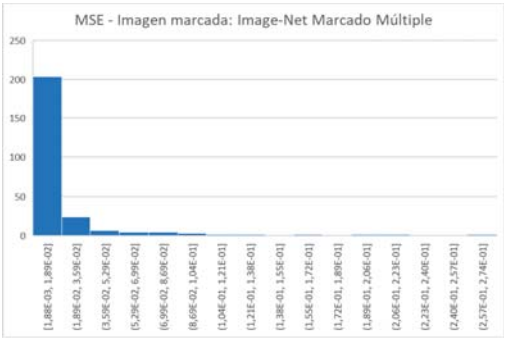
Figura 4.16: Casos de desempeño de la red con pérdida de calidad del fondo de la imagen en la parte superior y pérdida de color en la parte inferior. De izquierda a derecha: en primer lugar, la imagen sobre la que realizar el marcado. En segundo lugar, la marca de agua a usar. En tercer lugar, la imagen con la marca de agua incrustada. En cuarto lugar, la marca recuperada de la imagen marcada. En quinto lugar, la diferencia entre la imagen original y la imagen marcada. En último lugar la diferencia entre la marca original y la marca recuperada.



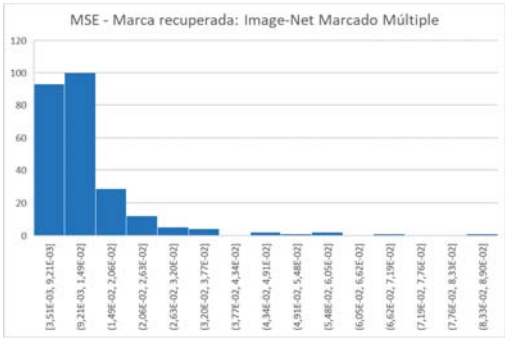
(a) PSNR - Imagen marcada



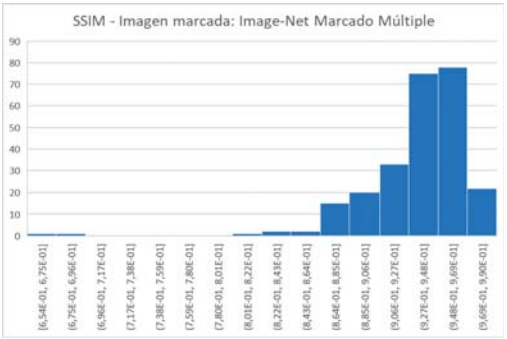
(b) PSNR - Marca recuperada



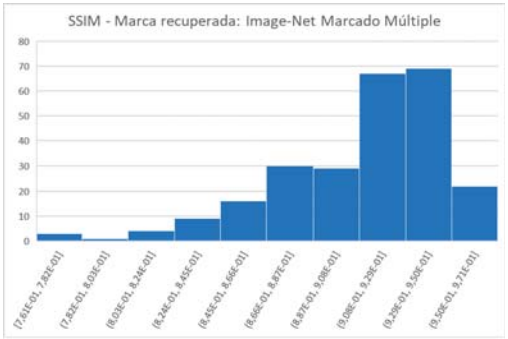
(c) MSE - Imagen marcada



(d) MSE - Marca recuperada



(e) SSIM - Imagen marcada



(f) SSIM - Marca recuperada

Figura 4.15: Histogramas para las distintas métricas para la imagen marcada (izquierda) y la marca recuperada (derecha).

4.5. Comparativa

Se van a comparar los resultados para todas las métricas estudiadas para los distintos modelos estudiados.

Para la métrica PSNR (Figura 4.17) se puede observar que para el MNIST con ambos modelos se obtienen resultados similares. Sin embargo, para el conjunto de datos Image-Net se obtienen resultados que están por debajo.

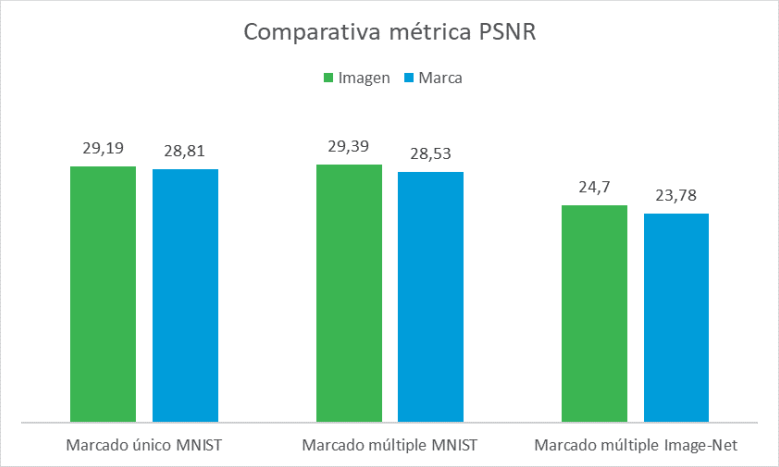


Figura 4.17: Comparativa de los distintos modelos para la métrica PSNR. En verde los valores para la imagen marcada. En azul los valores para la marca recuperada.

Conclusiones similares se obtienen para las métricas MSE (Figura 4.18) y SSIM (Figura 4.19).

Los resultados demuestran que el modelo con el conjunto de datos MNIST funciona de forma robusta, tanto para el marcado único como el múltiple. Sin embargo, debido a la complejidad del dataset como factor principal, se obtiene resultados con calidad inferior para el conjunto de datos Image-Net.

Si comparamos los resultados del modelo propuesto con los resultados obtenidos en [1], con unas condiciones del problema muy similares, se han alcanzado resultados equivalentes. Al tratarse de un estudio preliminar, dentro del alcance del proyecto, se puede considerar que los resultados obtenidos alcanzan niveles de calidad elevados.

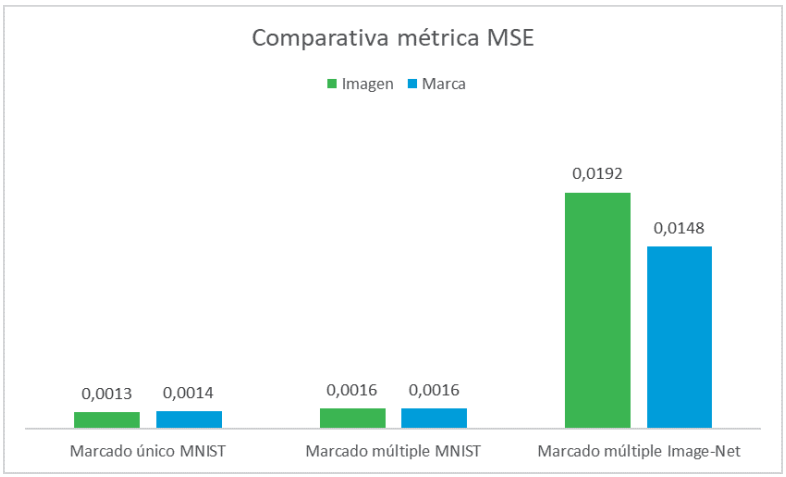


Figura 4.18: Comparativa de los distintos modelos para la métrica MSE. En verde los valores para la imagen marcada. En azul los valores para la marca recuperada.

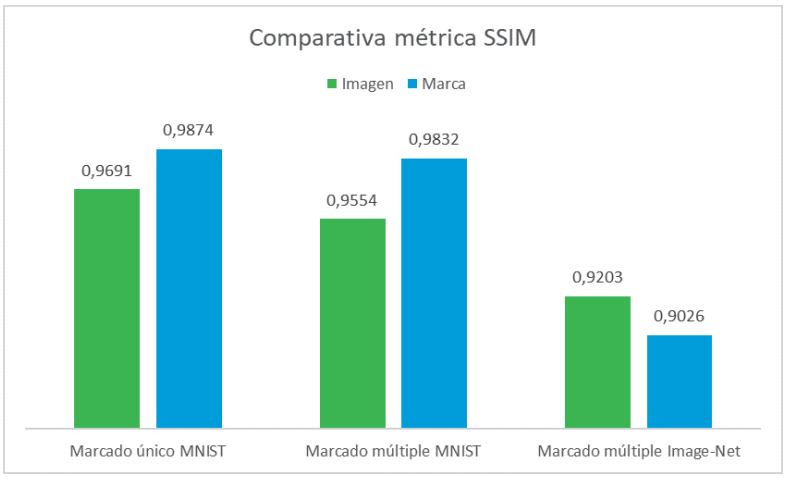


Figura 4.19: Comparativa de los distintos modelos para la métrica SSIM. En verde los valores para la imagen marcada. En azul los valores para la marca recuperada.

5. Conclusiones

En este trabajo se planteaba la creación de un método que permitiera realizar el marcado de agua imperceptible de imágenes de forma eficaz. La forma de realizarlo ha sido a través del uso de redes neuronales que fueran capaces tanto de incrustar la marca como de recuperarla cuando fuese necesario.

Para realizar la implementación se ha comenzado utilizando conjuntos de datos sencillos, como los del MNIST. Una vez resuelto y estudiado el problema sencillo, se ha extrapolado a un conjunto de datos de mayor complejidad, con imágenes a color. La principal complejidad es la de estructurar las redes, que con un coste computacional asumible, sean capaces de resolver el problema planteado.

Los distintos experimentos han conseguido demostrar que, en la mayoría de los casos, es posible realizar marcado de agua imperceptible sobre las imágenes. Para conseguir los resultados es necesario hacer énfasis que lo principal es tener un sistema acoplado de redes. Se ha de entrenar de forma simultánea tanto la red de marcado como la red de recuperación de la marca, aunque luego funcionen de manera desacoplada. Se ha demostrado también que es un problema que puede hacer caer a la red en el fenómeno del sobreajuste, siendo un factor muy importante a controlar.

Se ha hecho uso de distintas métricas con el objetivo de comprobar la validez de los resultados obtenidos. Los resultados obtenidos han indicado que en la mayoría de los casos es posible realizar marcado de agua imperceptible haciendo uso de redes neuronales. Se encuentra algunas excepciones en que, por factores particulares de las imágenes, el marcado de agua no se realiza de forma completamente satisfactoria, apareciendo elementos no deseados en las imágenes.

6. Herramientas y cronograma

6.1. Herramientas

Para el trabajo con redes neuronales se ha trabajado con la librería de *Tensorflow* [6], debido a la gran cantidad de información disponible para conocer su funcionamiento y su gran versatilidad. Como lenguaje de programación se ha utilizado *Python*.

Como librerías de *Python* principalmente se han utilizado *MatPlotLib* [7], *Scipy* [8] y *Numpy* [9].

6.2. Cronograma

El cronograma seguido en la realización del trabajo es el que se muestra en la figura 6.1

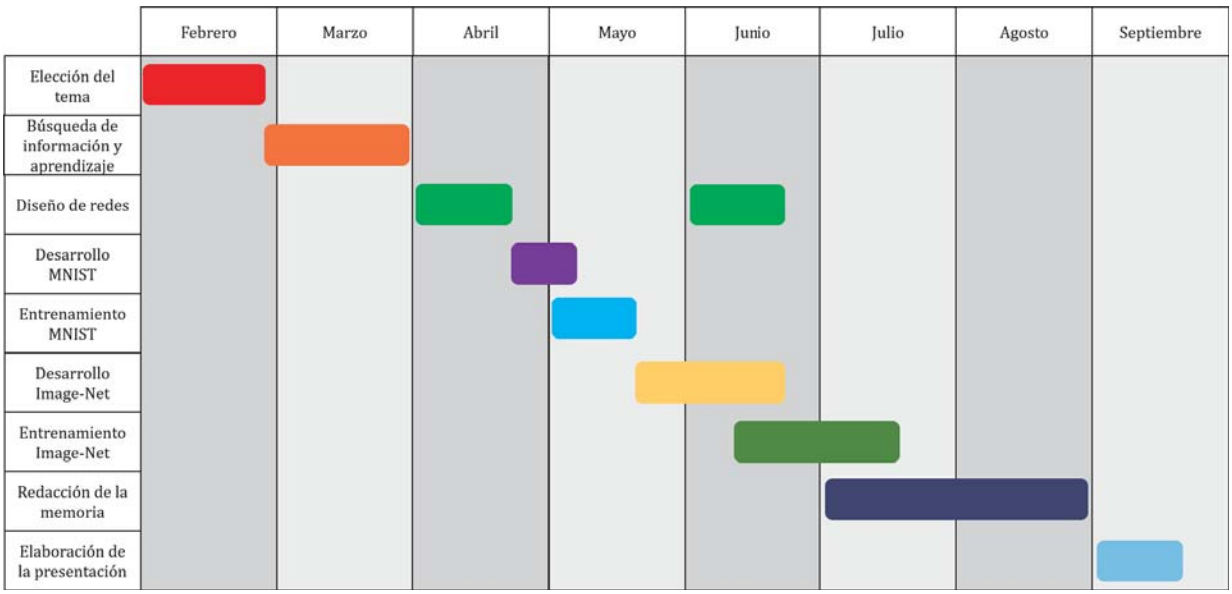


Figura 6.1: Cronograma de la elaboración del trabajo.

Bibliografía

[1] A. K. Singh, "Improved hybrid algorithm for robust and imperceptible multiple watermarking using digital images.," *Springer Science+Business Media*, 2016.

[2] L. B. Yan Lecunn, "Efficiente backprop," *Neural Networks: tricks of the trade*, pp. 2–41, 1998.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[4] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[5] K. P. Eric A. Silva, "Quantifying image similarity using measure of enhancement by entropy," *SPIE Proceedings Vol. 6579: Mobile Multimedia/Image Processing for Military and Security Applications 2007*, pp. 4–11, 2007.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[7] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science Engineering, vol. 9, no. 3*, pp. 90–95, 2007.

[8] P. P. E. Jones, T. Oliphant, "Scipy: Open source scientic tools for python,"

[9] S. C. C. S. v. d. Walt and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science Engineering, vol. 13, no. 2,*, pp. 22–30, 2011.