UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA DE COMPUTADORES

Escalado y Reconstrucción de Imágenes

Desarrollado por: Daniel Marín Bartolome

Dirigido por: Alberto Díaz Álvarez

Madrid, 22 de mayo de 2024

Escalado y Reconstrucción de Imágenes
Desarrollado por: Daniel Marín Bartolome
Dirigido por: Alberto Díaz Álvarez
Proyecto Fin de Grado, 22 de mayo de 2024

E.T.S. de Ingeniería de Sistemas Informáticos
Campus Sur UPM, Carretera de Valencia (A-3), km. 7
28031, Madrid, España

Si deseas citar este trabajo, la entrada completa en BibTeX es la siguiente:

```
@mastersthesis{citekey,
title = {Escalado y Reconstrucción de Imágenes},
author = {Marín, D. y Díaz, A.},
school = {E.T.S. de Ingeniería de Sistemas Informáticos},
year = {2024},
month = {5},
type = {Proyecto Fin de Grado}
}
```

Agradecimientos

Aquí los agradecimientos que quieras dar. Y si no quieres, borras la entrada `\acknowledgements` de `report.tex` y ya está.

Resumen

El resumen de un Proyecto Fin de Grado o de un Proyecto Fin de Máster condensa en tres o cuatro párrafos el contenido de la memoria. Se debe dar por sentado que el lector podrá tener una idea clara de lo que trata, y suele ser la primera barrera donde decide si continúa leyendo o no el texto.

Condensado no quiere decir incompleto. Debe contener la información más destacable. Lo ideal es que ocupe entre media y una cara de un folio A4. Comenzará por el propósito y principales objetivos de la memoria. Luego hablaremos sobre los aspectos más destacables de la metodología empleada, seguido de los resultados obtenidos. Por último se presentarán las conclusiones de forma condensada.

Debe tener un estilo claro y conciso, sin ambigüedades de ningún tipo. Además, al ser un resumen de todo el contenido, ni que decir tiene que deberá ser lo último que elaboraremos, y deberá mantener una absoluta fidelidad con el contenido de la memoria.

Palabras clave: Cuatro o cinco; Expresiones clave; Que resuman; Nuestro proyecto o; Investigación

Abstract

This section must contain the summary that we have written before in Spanish, but in English, as well as the keywords.

Keywords: Four or five; Key Expressions; Summarising; Our Project or; Research

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Índice de ecuaciones

1. Introducción

La reconstrucción y escalado de imágenes son áreas fundamentales en el procesamiento de imágenes digitales y en visión por computadora. Estas técnicas han sido muy útiles en campos como la medicina, la astronomía y la seguridad entre otros[1]-[4].

Lo que tienen en común estos tres campos es que, ya sean imágenes de objetos alejados a millones de kilómetros, a unos cuantos metros o incluso a escasos centímetros, es probable que las imágenes estén deterioradas. El deterioro de una imagen puede mostrarse de distintas formas como, desenfoque de movimiento, ruido o un simple error en la fuente donde se ha tomado la imagen[5].

Es importante disponer de herramientas que puedan reconstruir este tipo de imágenes, ya que, por ejemplo, en medicina podría ser decisivo para detectar posibles enfermedades. Por ello, he desarrollado este Proyecto Fin de Grado (PFG) para explorar diferentes técnicas de reconstrucción y escalado de imágenes utilizando redes de neuronas.

1.1. Objetivos

El objetivo principal de este PFG es diseñar y desarrollar modelos de redes neuronales para reconstruir y escalar imágenes. Dentro de este objetivo se han establecido los siguientes objetivos específicos:

- Comparar diversas arquitecturas de redes neuronales y evaluar su tiempo de entrenamiento, con el propósito de identificar el modelo más eficiente.
- Investigar cómo distintas arquitecturas de redes neuronales pueden influir en la precisión de la restauración y escalado de imágenes.

1.2. Motivación

El principal motivo que me ha llevado a realizar este PFG son las posibles aplicaciones prácticas que puede tener el uso de redes neuronales para la reconstrucción y escalado de imágenes. En muchas profesiones, la captura de imágenes es esencial, por lo que si estas presentan deterioro, este tipo de tecnología puede ayudar en ámbitos como la medicina, astronomía, seguridad o incluso en entretenimiento[1]-[4], [6].

1.3. Justificación

La principal razón por la que se ha elegido este PFG es para abordar el problema que puede presentar la captura de imágenes, y es el deterioro. En muchas profesiones se trabaja día a día con imágenes y es esencial procesarlas para que sean lo mas fieles a la realidad:

- En el ámbito de la medicina, es necesario la visualización de imágenes por parte del personal médico para la detección o prevención de posibles enfermedades. Las imágenes más comunes son los rayosX, ultrasonidos, resonancias magnéticas, endoscopias, *Tomografía Axial Computarizada* (TAC) o mamografías para la posible detección del cáncer de mama[2], [7]-[9]. Un software muy utilizado es ImageJ2, escrito en java aunque con la capacidad de trabajar con scripts en python, tiene herramientas de escalado y restauración de imágenes[10].
- En el ámbito de la astronomía, la observación y el análisis de imágenes juegan un papel fundamental en la comprensión del universo y sus fenómenos. Los astrónomos utilizan una amplia variedad de técnicas de observación, desde telescopios terrestres hasta observatorios espaciales, para capturar imágenes de objetos celestes en diferentes longitudes de onda del espectro electromagnético. Estas imágenes proporcionan información crucial sobre la composición, la estructura y la evolución de objetos astronómicos, como estrellas, galaxias, nebulosas y planetas[3], [11].

- En el ámbito de la seguridad, las cámaras de seguridad son un elemento muy importante para identificar a posibles delincuentes, monitorear áreas de interés y prevenir incidentes[4].
- En el ámbito del entretenimiento, como pueden ser los videojuegos, se han desarrollado técnicas para mejorar la resolución de los juegos manteniendo la tasa de refresco intacta, tales como el *Deep Learning Super Sampling* (DLSS) por parte de NVIDIA o *FidelityFX Super Resolution* (FSR) de su competidora AMD[6], [12].

1.4. Estructura de la memoria

El documento está estructurado de la siguiente manera:

1. En el apartado 2 se mirará detalladamente el marco teórico y el estado del arte, mostrando tanto el modelo clásico de restauración y escalado como el actual.
2. En el apartado 3 se explicará el preprocesamiento de los datos realizado y las arquitecturas de redes neuronales utilizadas.
3. En el apartado 4 se mostrarán y explicarán los resultados obtenidos, los objetivos logrados y los problemas encontrados durante el desarrollo del PFG.
4. En el apartado 5 se presentarán las conclusiones, el impacto social y medioambiental, y las líneas futuras.

2. Marco teórico y Estado del arte

El problema del procesamiento de imágenes para su restauración y escalado lleva tratándose desde hace décadas. A lo largo de este tiempo, se han desarrollado diferentes técnicas y algoritmos. Se llamará modelo clásico a todos los algoritmos y técnicas que se desarrollaron antes de la popularización de las las redes de neuronas.

2.1. Modelo Clásico: Restauración

El deterioro en las imágenes es un factor que puede afectar su captura. Este deterioro puede manifestarse de distintas formas, como desenfoque de movimiento, presencia de ruido o fallos en el dispositivo de captura[5].

El deterioro se puede representar matemáticamente de la siguiente forma[13]:

$$y = D * X + n \tag{2.1}$$

Donde X es la imagen original, D es la función de desenfoque (o filtro paso bajo) y n es el ruido que se le añade a la imagen. En (2.1), y es la imagen resultante de aplicar las transformaciones. Un ejemplo simple de función en el modelo clásico es el **filtro inverso**. Este método parte de:

$$g = X * b \tag{2.2}$$

Donde g es la imagen borrosa y b es un filtro paso bajo.

Esta aproximación aplica una convolución $*$ de la imagen borrosa g con un filtro de paso alto h para recuperar la imagen original:

$$X = g * h \tag{2.3}$$

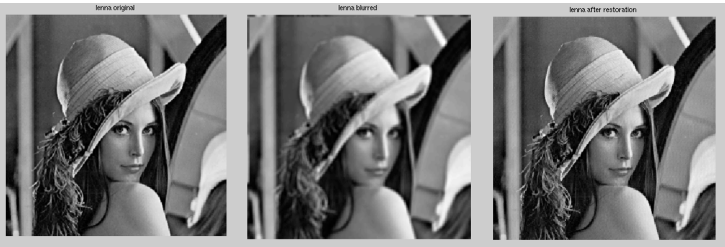


Figura 2.1. Imagen comparativa del uso del filtro inverso en una imagen con desenfoque. Fuente: [14]

Sin embargo, el filtrado inverso responde negativamente ante la presencia de ruido:

Figura 2.2. Imagen comparativa del uso del filtro inverso en una imagen con ruido. Fuente: [14]

Este algoritmo necesita saber cuál es la función de paso bajo. Existen otras aproximaciones llamadas **deconvoluciones a ciegas** que no necesitan esta información, con tan solo la imagen pueden funcionar. Un ejemplo es el método *Power Spectrum Equalization* (PSE) que busca restaurar la distribución original de la energía en el espectro de frecuencias, lo que puede mejorar la claridad y calidad de la imagen. Esto se logra aplicando filtros o técnicas de procesamiento que ajustan la amplitud de las diferentes frecuencias para igualarlas a un espectro de referencia deseado[15].

Su fórmula principal es:

$$G = [\frac{S_{uu}}{|H|^2 S_{uu} + S_{nn}}]^{1/2} \tag{2.4}$$

Donde H es la transformada de Fourier de h (la función de paso bajo). Este es el parámetro que se quiere estimar para hallar una imagen G sin deterioro. S_{uu} es la densidad espectral y S_{nn} es el ruido espectral. Ambos son parámetros que se supone que se conocen o se pueden estimar, aunque los resultados no son los mejores:

Figura 2.3. Imagen comparativa del uso de la deconvolución a ciegas. Fuente: [15]

2.2. Modelo Clásico: Escalado

En el problema del escalado de imágenes, existen varias técnicas de interpolación. La interpolación es una técnica que consiste en rellenar y añadir nuevos píxeles a una imagen únicamente con la información de la misma [16].

Si se tiene una imagen como esta:

Figura 2.4. Imagen de El Fary reescalada a 64x64. Fuente: [17]

Una primera aproximación sería separar los puntos y rellenarlos con los píxe-

les que hay alrededor, a este proceso se le conoce como ***Nearest Neighbor***:

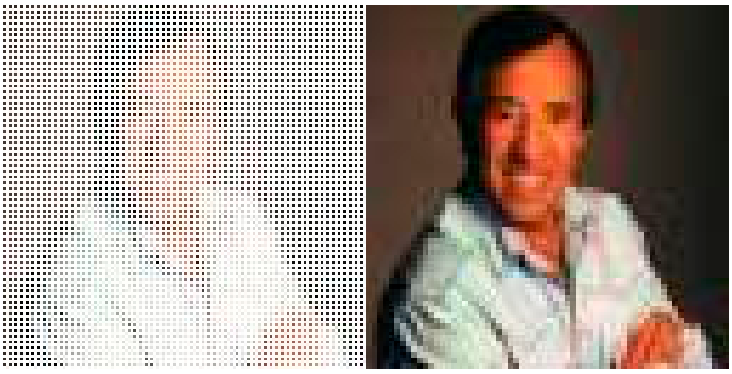


Figura 2.5. Proceso de *Nearest Neighbor*. Fuente: Elaboración propia.

El resultado no es más que una imagen con píxeles más grandes y no consigue verse con mejor resolución. Se puede mejorar rellenando los huecos con la media entre los puntos, esta aproximación la utilizan la interpolación bilinear y la bicúbica:

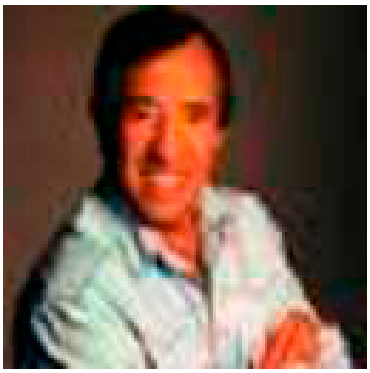


Figura 2.6. Escalado con bilinear. Fuente: Elaboración propia.

2.3. Modelo Actual

Como se ha visto en la sección anterior, los algoritmos clásicos para restaurar imágenes son ineficientes y requieren información adicional que debe ser estimada mediante cálculos muy complejos. En cuanto a los modelos para escalar imágenes, no logran obtener una resolución satisfactoria.

El principal inconveniente de estas aproximaciones de restauración y escala-

do es que no pueden generar datos que no existen en la imagen original. Es decir, no pueden proporcionar información que la imagen en sí misma no contenga. Para abordar este desafío, se recurre a las redes neuronales.

Aunque las redes de neuronas fueron propuestas en 1943 por McCulloch y Pitts[18], no ha sido hasta ahora, gracias a los avances en la tecnología y en el campo de la inteligencia artificial, que es posible crear modelos complejos de redes capaces de reconocer patrones complicados en los datos de entrada.

Las redes de neuronas artificiales intentan imitar el comportamiento de las neuronas biológicas. Es decir, las neuronas reciben estímulos (cálculos matemáticos) y generan una salida. Estas constan de cinco partes: las entradas x_i, la salida y, los pesos w_i, el *bias* b y la función de activación a[18], [19].

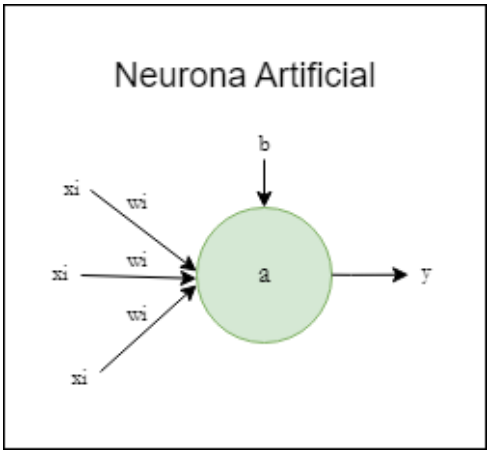


Figura 2.7. Neurona Artificial. Fuente: Elaboración propia.

Si se juntan más neuronas, forman lo que se llama **perceptrón multicapa**. Este modelo tiene la siguiente estructura: una capa de entrada, una o muchas capas ocultas y una capa de salida[19].

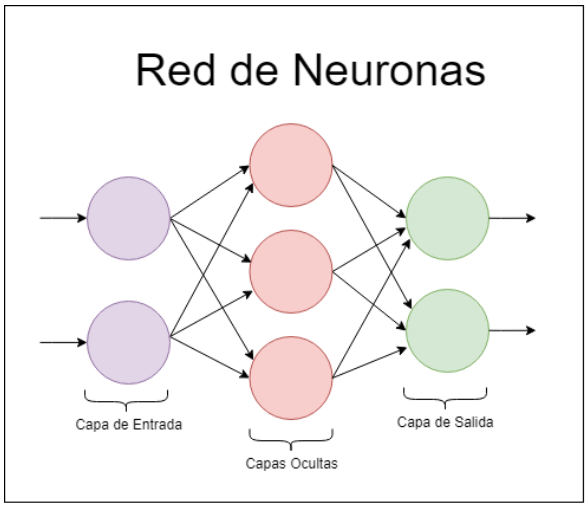


Figura 2.8. Red de neuronas. Fuente: Elaboración propia.

Existen tres fases principales en las redes de neuronas: la inferencia o propagación, el cálculo del error y la retropropagación.

La **inferencia** sirve para generar un resultado a partir de unas entradas. El algoritmo es el siguiente: se multiplica cada entrada x_i por su peso w_i y luego se suma cada producto pasando el resultado a la función de activación a que genera una salida y.

El **cálculo del error** consisten en comparar la salida de la red con el valor objetivo al que se quiere llegar. Existen distintos tipos de errores, los mas comunes son: *Error Absoluto Medio* (MAE) y *Error Cuadrático Medio* (MSE).

En la **retropropagación**, se ajustan los pesos w_i para que la red pueda adaptarse a su tarea[20]. La fórmula es la siguiente:

$$\Delta w_i = w_i - \alpha \cdot \delta_i \tag{2.5}$$

Donde α es el factor de aprendizaje o *learning rate* Indica cómo de rápido tiene que ajustarse el modelo.

2.3.1. Redes Convolucionales

Entendiendo cómo funcionan las redes de neuronas, si se translada al problema de este PFG, es decir, trabajar con imágenes, siguiendo el modelo anterior, se tendría que aplanar la imagen para pasarla a la red de neuronas, cosa que supondría dos principales problemas:

1. Para imágenes grandes, el número de neuronas necesario para cada píxel escala de forma exponencial. Es decir, si se tiene una imagen de 1024x1024, las neuronas de entrada serían 1.048.576.
2. Esta distribución hace que se pierda la información espacial de la imagen. El primer pixel de la imagen está relacionado con el último, cuando estos no tienen ninguna relación.

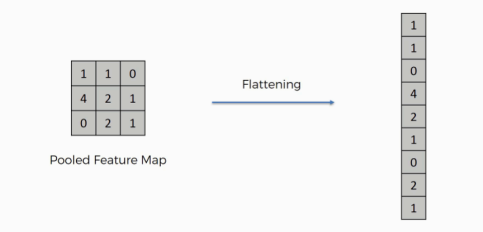


Figura 2.9. Aplanar una imagen para la entrada de una red de neuronas. Fuente: [21]

Teniendo todo esto en cuenta, se hace necesaria la creación de otro tipo de redes orientadas a imágenes con el fin de reducir la dimensionalidad de la red y evitar la pérdida de información espacial de la imagen. Las redes convolucionales surgen para abordar este tipo de problema.

Las ***redes de neuronas convolucionales* (CNN)** tienen sus raíces en el modelo *Neocognitron*, desarrollado por Kunihiko Fukushima en 1980, que sentó las bases para el reconocimiento de patrones a través de estructuras jerárquicas[22]. Posteriormente, en 1998, Yann LeCun y sus colaboradores introdujeron el uso de la retropropagación para el entrenamiento eficaz de estas redes[23].

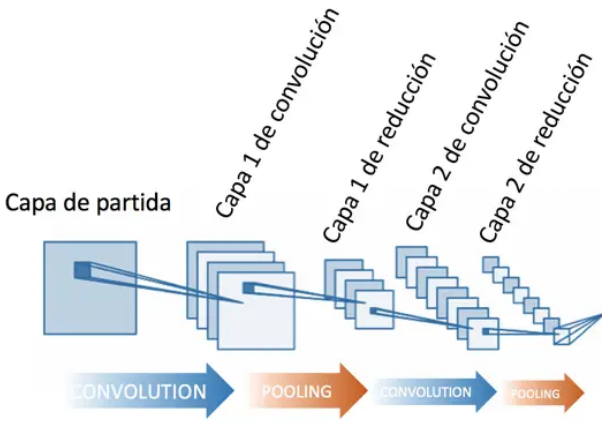


Figura 2.10. Arquitectura de una *redes de neuronas convolucionales* (CNN). Fuente: [24]

Las ***redes de neuronas convolucionales*** extraen características de las imágenes mediante capas compuestas por diferentes filtros del mismo tamaño, cada uno de los cuales contiene pesos w_i. Estas redes neuronales aprenden qué filtros son los más efectivos para extraer las características relevantes de las imágenes.

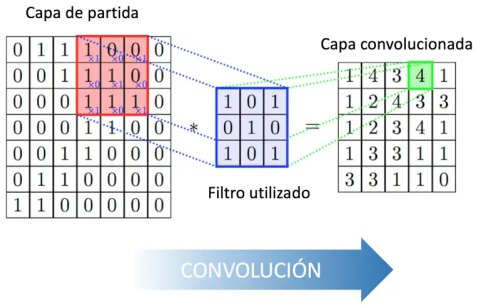


Figura 2.11. Operación de convolución. Fuente: [24]

Para reducir la dimensionalidad de la imagen manteniendo la información importante, se puede utilizar una técnica llamada *pooling*. El *pooling* aplica un operador a pequeñas subregiones de la imagen, llamadas *kernels*, para combinar sus valores y producir una única salida[24]. Existen dos variantes principales de *pooling*:

- ***MaxPooling***: Selecciona el valor más grande dentro de cada subregión del *kernel*, reteniendo la característica más prominente de esa región.

- ***AveragePooling***: Calcula el valor promedio de los elementos en cada subregión del *kernel*, obteniendo una representación suavizada.

El uso del *pooling* ayuda a reducir el tamaño de las representaciones intermedias en la red, disminuyendo la cantidad de parámetros y la carga computacional.

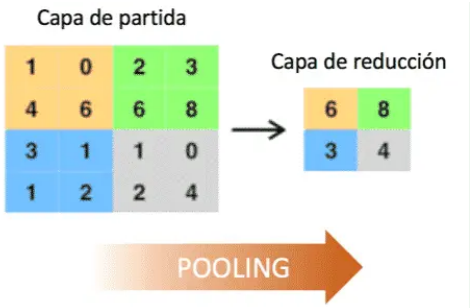


Figura 2.12. Operación de MaxPooling. Fuente: [24]

También existe la operación contraria llamada *transpose convolution*:

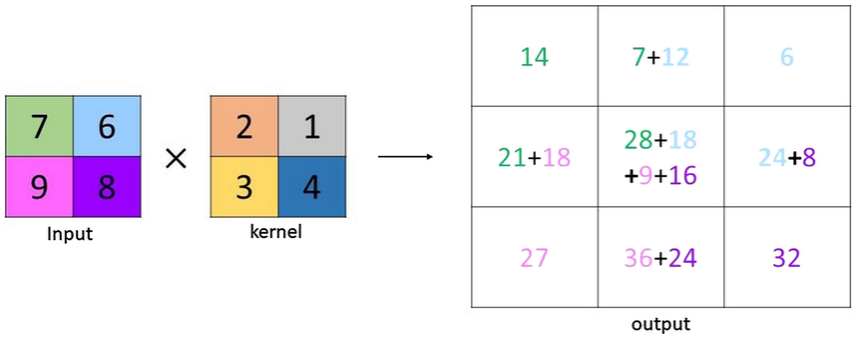


Figura 2.13. Transpose Convolution. Fuente: [25]

Esta operación es útil para **escalar** una imagen.

2.3.2. Arquitecturas de redes de neuronas

Una arquitectura de red de neuronas se refiere a la estructura organizativa de las neuronas artificiales y las conexiones entre ellas[26]. Existen distintos tipos de arquitecturas como:

Autoencoders

Los **autoencoders** fueron propuestos por primera vez por Kramer en 1991 como una forma de redimensionar datos[27], [28]. Los autoencoders constan de tres partes: el codificador, el espacio latente y el decodificador.

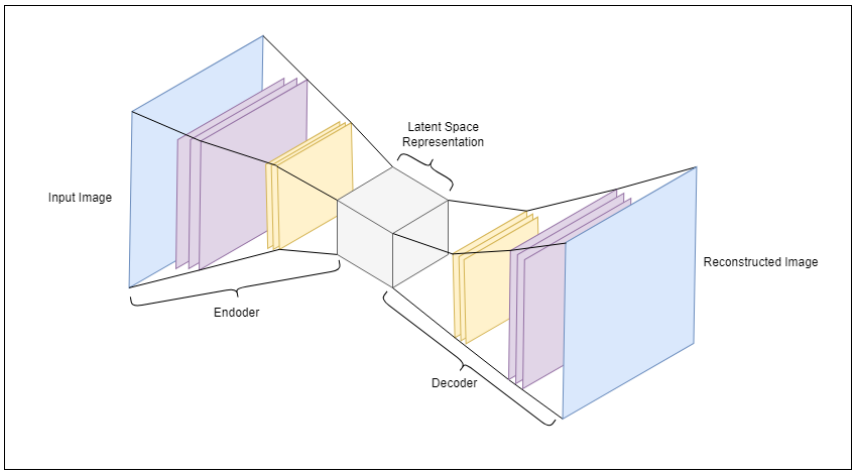


Figura 2.14. Autoencoder. Fuente: Elaboración Propia.

El **codificador** o en ingles *encoder* se puede representar de la siguiente forma[28]:

$$h_i = g(X_i) \quad \text{donde } h_i \in R^q \text{ simboliza el espacio latente.} \tag{2.6}$$

El **decodificador** o en ingles *decoder* se puede representar de la siguiente forma:

$$\tilde{x}_i = f(h_i) = f(g(x_i)) \quad \text{donde } \tilde{x}_i \in R^n \tag{2.7}$$

Para el entrenamiento del autoencoder, hay que buscar las funciones f y g que minimicen la diferencia entre x_i y $\tilde{x}_i$:

$$argmin_{f,g} < [\Delta(x_i, f(g(x_i)))] > \tag{2.8}$$

VAEs

Los ***Variational Autoencoders* (VAEs)** fueron introducidos por primera vez en 2013 por Diederik P. Kingma y Max Welling[29]. Los VAEs usan la capacidad de los autoencoders para aprender representaciones eficientes de datos con principios de inferencia variacional, permitiendo la generación de nuevos datos similares a los de entrenamiento.

En los VAEs, se trabaja con dos tipos de variables: las observables y las latentes. Las variables latentes son variables que están dentro del modelo y capturan las características subyacentes de los datos.

Sea $x \in X^D$ un vector de variables observables, donde $X \subseteq R$ y sea $z \in R^M$ un vector de variables latentes. Para ver cómo las variables latentes generan las variables observables, se necesita la probabilidad conjunta de x y z.

$$p_\theta(x, z)dz \tag{2.9}$$

Para sacar la distribución marginal con las propiedades de la probabilidad conjunta para datos continuos se tiene[29], [30]:

$$p_\theta(x) = \int p_\theta(x, z)dz \tag{2.10}$$

Para entrenar el modelo es necesario despejar z cosa que computacionalmente es imposible de tratar, así que, se opta por otra solución: intentar maximizar la función *Evidence Lower Bound* (ELBO)[29], [30]:

$$\text{ELBO} => \ln p_\theta(x) \leq E_{z\sim q_\phi(z/x)}[\ln p_\theta(x/z) + \ln p_\theta(z) - \ln q_\phi(z/x)] \tag{2.11}$$

- Donde $q_\phi(z/x)$ es el **codificador**.
- Donde $p_\theta(x/z)$ es el **decodificador**.
- Donde $p_\lambda(z)$ es el prior o la **similitud marginal**.

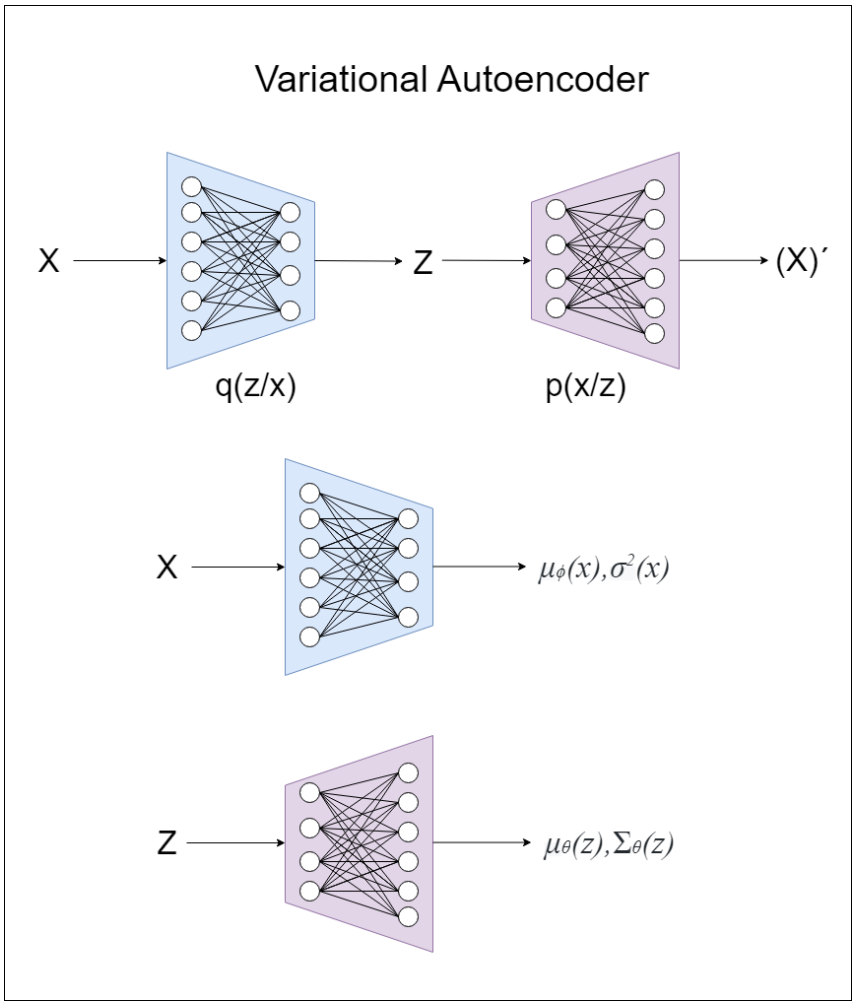


Figura 2.15. Representación de un variational autoencoder. Fuente: Elaboración Propia.

GANs

Las **redes generativas adversarias** o ***Generative Adversarial Networks* (GANs)**, presentada por Ian Goodfellow en 2014[31], han representado un avance significativo en el campo de la Inteligencia Artificial generativa. Esta arquitectura consta de dos redes neuronales que compiten por producir resultados óptimos. La primera red es la **generadora**, que busca crear imágenes realistas, mientras que la segunda es la **discriminadora**, encargada de distinguir entre imágenes reales y generadas.

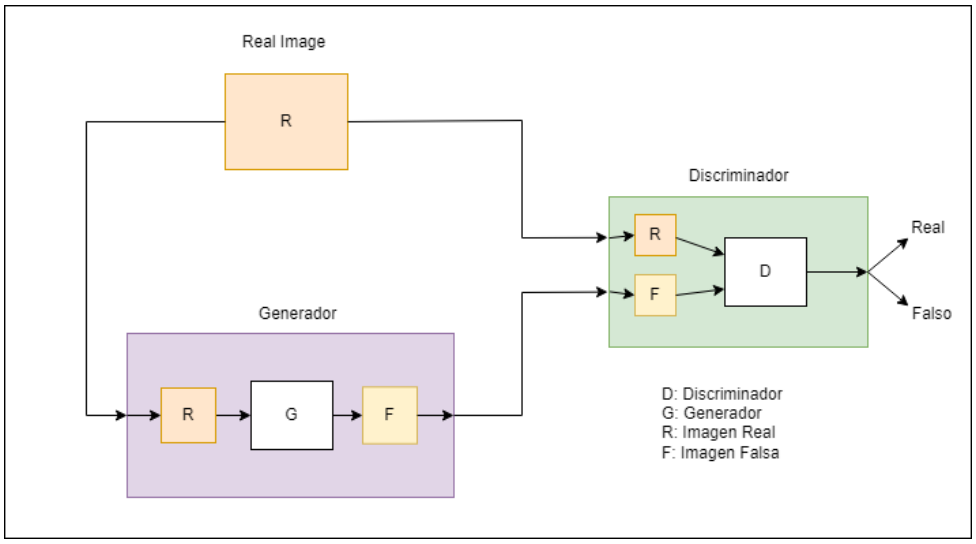


Figura 2.16. Representación de una red *Conditional Generative Adversarial Network* (cGAN). Fuente: Elaboración Propia.

La fórmula general de las redes GAN parte de la fórmula de la entropía cruzada binaria o binary cross entropy:

$$L = -\sum y \ln(\hat{y}) + (1-y)\ln(1-\hat{y}) \tag{2.12}$$

- Si $y = 1$; $\quad \hat{y} = D(x) \Rightarrow L = \ln[D(x)]$
- Si $y = 0$; $\quad \hat{y} = D(G(z)) \Rightarrow L = \ln[1 - D(G(z))]$

Entonces substituyendo sobre 2.12 nos queda:

$$L = \ln[D(x)] + \ln[1 - D(G(z))] \tag{2.13}$$

Si aplicamos la esperanza en toda la igualdad:

$$E(L) = E(ln[D(x)]) + E(ln[1 - D(G(z))]) \tag{2.14}$$

Aplicando las propiedades de la esperanza para valores continuos:

$$E[X] = \int_R X f(X) dx => \int P_{datos}(x)(ln[D(x)])dx + \int P_z(z)(ln[1 - D(G(z))])dz \tag{2.15}$$

Reescribiendo la ecuación 2.15 queda esta expresión en la que ambas funciones compiten, ya que, la función G quiere minimizarla y D maximizarla:

$$min_G max_D V(G,D) = E_{x \sim P_{datos}}[ln(D(x))] + E_{z \sim Pz}[ln(1 - D(G(z)))] \tag{2.16}$$

3. Metodología

En este apartado se empezará explicando los programas que se han utilizado y el proceso de creación del modelo.

3.1. Software utilizado

Los programas que se han utilizado para la realización de este PFG son:

3.1.1. draw.io

Todos los gráficos que se han realizado han sido hechos utilizando la herramienta online draw.io.

Draw.io es una herramienta online capaz de crear diagramas de flujo, organigramas, *wireframes*, diagramas en UML y diagramas de redes. Está hecha en HTML 5 y JavaScript[32].

Figura 3.1. Logotipo de drawio. Fuente: [32]

3.1.2. Python

Python es un lenguaje de programación interpretado de alto nivel. Se distingue por ser multiparadigma, lo que significa que ofrece soporte parcial para la orientación a objetos, la programación imperativa y, en menor medida, la programación funcional[33].

Figura 3.2. Logotipo de Anaconda. Fuente: [33]

Python permite el uso de paquetes los cuales se pueden instalar. Los paquetes que se han utilizado son:

- **Tensorflow con Keras**: Tensorflow es una biblioteca de código abierto creada por Google para aprendizaje automático[34], [35]. Keras también es una biiblioteca de código abierto la cual se puede ejecutar sobre Tensorflow para crear redes de neuronas[36].
- **Numpy**: Numpy es una biblioteca que se utiliza para manejar vectores y matrices[37].
- **Os**: OS es una biblioteca por defecto para llamar operaciones del sistema operativo.
- **Pillow**: Pillow es una biblioteca que sirve para cargar, guardar y mostrar imágenes por pantalla.
- **Matplotlib**: Matplotlib es una biblioteca que se utiliza para mostrar gráficas.
- **Time**: Time se utiliza para llamar conseguir valores como la hora o la fecha del sistema operativo.
- **IPython**: Es un *shell* interactivo de Jupyter Notebooks que añade más funcionalidades.

- **Pandas**: Pandas es una biblioteca que se utiliza para crear, cargar y guardar conjuntos de datos llamados Datasets.

Laversión utilizada de python es la **3.9.18**.

3.1.3. Anaconda

Anaconda es un programa de código abierto que sirve para simplificar la gestión de paquetes en Python y R[38].

Figura 3.3. Logotipo de Anaconda. Fuente: [39]

En este PFG se ha utilizado para facilitar la instalación de tensorflow y cuda para poder entrenar las redes de neuronas en local con una *Graphics Processing Unit* (GPU).

3.1.4. Visual Studio Code

Visual studio code es un editor de código fuente de código abierto creado por *Microsoft*. Este editor permita la instalación de extensiones que permiten manejar diferentes lenguajes de programación.

Figura 3.4. Logotipo de Visual Studio Code. Fuente: [40]

3.1.5. Extensión de visual Studio Code: Jupyter Notebooks

Jupyter es una organización sin ánimo de lucro que alberga una variedad de productos diseñados para la computación interactiva y la colaboración en ciencia de datos y computación científica. Uno de sus productos más destacados es Jupyter Notebook, un entorno interactivo basado en la web que permite la creación de documentos en los que se puede combinar código ejecutable, texto enriquecido, visualizaciones y otros elementos multimedia.

Figura 3.5. Logotipo de Jupyter. Fuente: [41]

La extensión de Jupyter para Visual Studio Code ha sido creada por *Microsoft* y permite la creación de estos notebooks en la propia plataforma.

3.1.6. Microsoft Excel

Microsoft excel es un program desarrollado por Microsoft para visualizar, cargar guardar y editar hojas de cálculo[42].

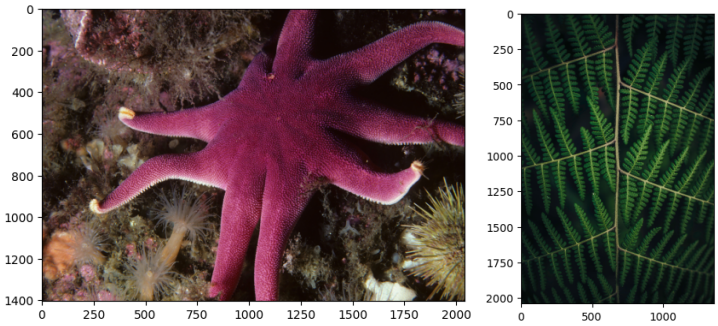
Figura 3.6. Logotipo de Jupyter. Fuente: [43]

3.2. Creación del Dataset

Se ha utilizado el dataset DIV2K. El dataset DIV2K fue creado en 2017 por un equipo de investigadores de la Universidad Técnica de Múnich y de la Universidad de Ciencia y Tecnología de Pohang, Corea del Sur [44], [45].

El dataset cuenta con 1,000 imágenes de alta resolución, de las cuales 800 son para entrenamiento y las 200 restantes para validación y prueba.

Antes de realizar cualquier procesamiento, se examinó la resolución de las imágenes y se decidió reescalarlas a 1024x1024x3. A las imágenes con esta resolución se les denominará de alta resolución. Esta decisión se tomó principalmente porque no todas las imágenes tienen la misma resolución y debido a las limitaciones computacionales.



(a) (b)

Figura 3.7. Comparación de imágenes en el dataset DIV2K Fuente:[44], [45]

Se han creado 3 datasets diferentes dependiendo del escalado de la imagen:

- **Dataset escala x1 (De 1024x1024 a 1024x1024)**: Este dataset contiene pares de imágenes de alta resolución con la misma resolución, es decir,dos imágenes de 1024x1024x3.
- **Dataset escala x2 (De 512x512 a 1024x1024)**: Este dataset contiene pares de imágenes con distinta resolución, una con resolución media, es decir, 512x512x3, y la otra con resolución alta.
- **Dataset escala x4 (De 256x256 a 1024x1024)**: Este dataset contiene también pares de imágenes con distinta resolución, una con baja resolución,

es decir, 256x256x3, y la otra con alta resolución.

En cada dataset, se han cogido las imágenes de alta, media y baja resolución y se les a aplicado una transformación siguiendo el criterio de [5], [13] con la fórmula 2.1 mencionada en el apartado 2. Es decir, se les ha aplicado una función de **desenfoque** a la imagen y ese resultado se le a añadido **ruido**.

La función de desenfoque, obtenida de [46],es la siguiente:

```
def gaussian_blur(img, kernel_size=2, sigma=50):
  def gauss_kernel(channels, kernel_size, sigma):
    ax = tf.range(-kernel_size//2+1.0,kernel_size//2+1.0)
    xx, yy = tf.meshgrid(ax, ax)
    kernel = tf.exp(-(xx ** 2 + yy ** 2)/(2.0 * sigma ** 2))
    kernel = kernel / tf.reduce_sum(kernel)
    kernel = tf.tile(kernel[...,tf.newaxis],[1, 1,channels])
    return kernel
gaussian_kernel = gauss_kernel(tf.shape(img)[-1],kernel_size,
   sigma)
gaussian_kernel = gaussian_kernel[..., tf.newaxis]
img = tf.expand_dims(img, axis=0)
img_blurred = tf.nn.depthwise_conv2d(img, gaussian_kernel,
          [1, 1, 1, 1], padding='SAME', data_format='NHWC')
img_blurred = tf.squeeze(img_blurred, axis=0)
return img_blurred
```

La función gaussian blur aplica un desenfoque gaussiano a una imagen utilizando TensorFlow. Primero, crea un *kernel* gaussiano en función del tamaño del *kernel* y sigma. Luego, expande la dimensión de la imagen para realizar una convolución 2D con este *kernel*, aplicando el desenfoque. Finalmente, devuelve la imagen desenfocada eliminando la dimensión añadida temporalmente.

La función de ruido es la siguiente:

```
def corrupt_part_of_image(image, noise_level, corruption_level
```

```
 =0.04):
mask = tf.random.uniform(tf.shape(image)[:2]) <
   corruption_level
noise = tf.random.normal(tf.shape(image), stddev=noise_level)
noise = (noise + 1.0) / 2.0
corrupted_image = tf.where(mask[..., tf.newaxis],
      tf.clip_by_value(image + noise, 0.0, 1.0), image)
return corrupted_image
```

Esta función realiza las siguientes operaciones: primero, crea una máscara aleatoria para determinar qué partes de la imagen serán corrompidas. Luego, genera ruido con un nivel específico y lo añade a las partes seleccionadas, asegurándose de que los valores de los píxeles permanezcan en el rango válido [0, 1]. Finalmente, devuelve la imagen con las partes afectadas por el ruido.

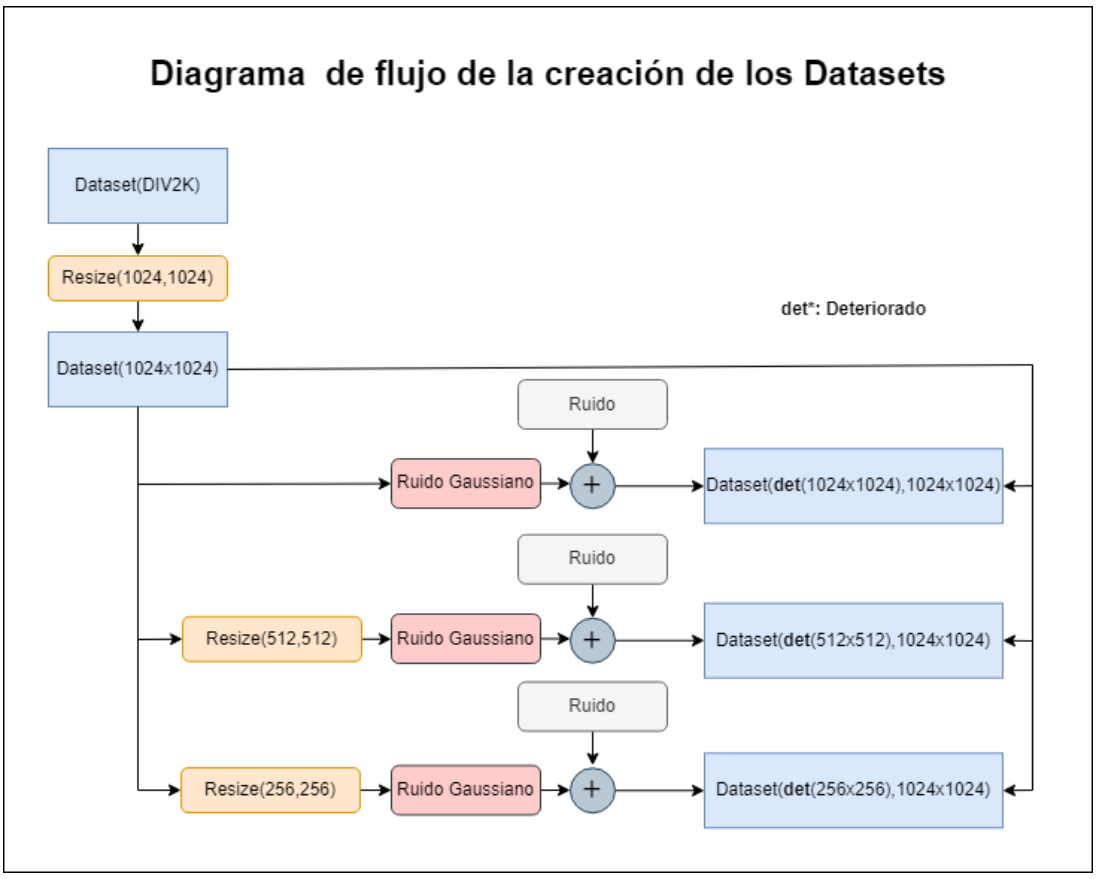


Figura 3.8. Diagrama de flujo de la creación de los Datasets. Fuente: Elaboración propia

Cada dataset tiene un tamaño de lotes (o *batch size*) de 1.

Con los datasets ya listos se pasa a la siguiente fase, la creación de los modelos y su entrenamiento.

3.3. Creación de Modelos y Entrenamiento

Lo siguiente que se ha realizado es la creación y el entrenamiento de los modelos.

Para dar más contexto del hardware utilizado en el entrenamiento de cada red de neuronas, cada red se ha entrenado de forma local en un ordenador de sobremesa con las siguientes especificaciones:

Para cada arquitectura se han creado tres modelos diferentes para cada dataset, a excepción del VAE que únicamente se ha reslizado un modelo.

- CPU: Intel Core i5 de 13ª generación.
- RAM: 16G de memoria RAM.
- GPU: NVIDIA GTX 3080 de 10G de memoria.

3.3.1. Autoencoder

Creación de los modelos

Se han hecho tres arquitecturas de redes de neuronas adaptado para cada dataset. El inicio de los tres autoencoders es la siguiente: se han puesto tres capas de convolución seguidas cada una de ellas con una capa de MaxPooling, luego, se han puesto otras tres capas de convolución seguidas cada una de ellas con una capa de UpSampling. A partir de aquí cada modelo es diferente:

- Para el modelo de alta resolución se ha puesto una capa final de convolución.

- Para el modelo de media resolución se ha puesto una capa extra de convolución seguida de una capa de UpSampling, y por último, una capa de convolución.
- Para el modelo de baja resolución se han puesto dos capas extra de convolución seguidas cada una de ellas con una capa de UpSampling, y por último, una capa de convolución.

Todas las capas intermedias tienen la función de activación *relu* y la capa final tiene la función de activación *sigmoidea*.

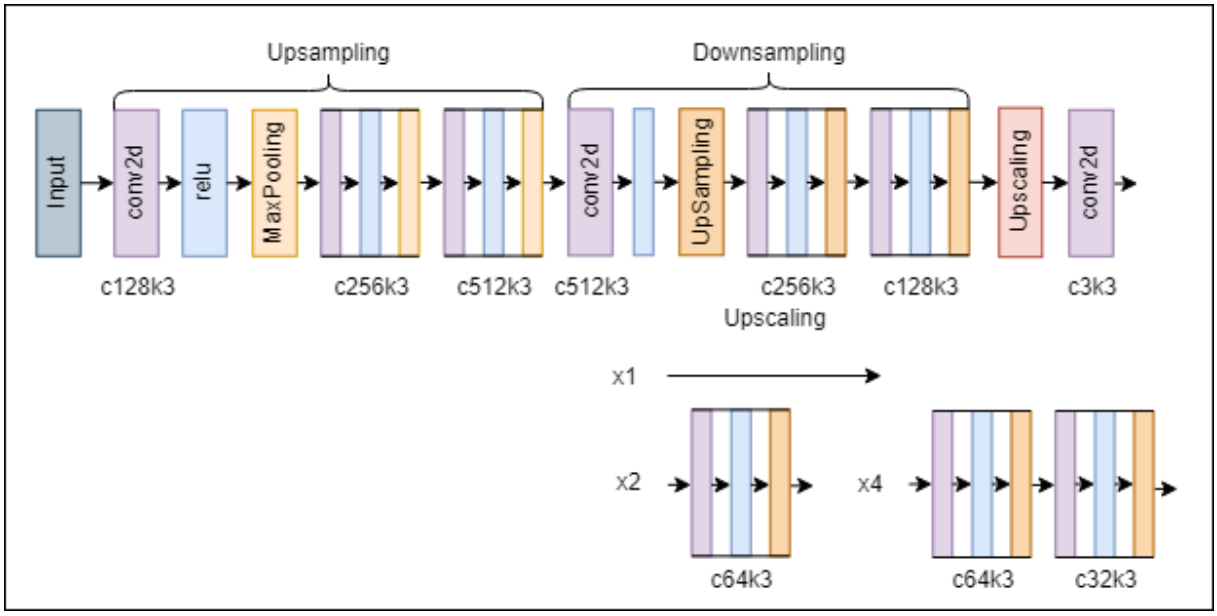


Figura 3.9. Composición del autoencoder. Los valores debajo de la convolución significan: c el número de filtros y k el tamño del kernel. Fuente: Elaboración propia

Entrenamiento de los modelos

Para el entrenamiento de este modelo se ha utilizado como función de pérdida MAE con el optimizador adam.

Para estimar la cantidad de epochs necesarios para el entrenamiento, se ha puesto un valor alto y se ha visto el epoch donde converge la función de pérdida.

3.4. VAE

Creación del modelo

En el caso del VAE, debido a su arquitectura, no se ha realizado ningún escalado, solo se ha hecho un modelo en relación al escalado. La cantidad de parámetros es tan inmensa, que además, se ha decidido crear un nuevo dataset únicamente para este modelo. Este dataset contiene parches de tamaño 64x64x3, con un tamaño de lotes de 30, del dataset DIV2K. En cuanto a la creación del modelo, es un VAE convolucional.

la arquitectura del VAE se divide en dos partes: la parte del codificador y la parte del decodificador.

El codificador comienza con cuatro filtros de convolución, teniendo el tercero un tamaño de stride de 2, con lo que divide a la mitad el alto y ancho de los filtros. La capa final del codificador termina dividiéndose en dos capas densas, la capa z_mean y la capa z_log_var, lo que representa la media y la logvarianza del espacio latente. Estos dos vectores se utilizan para lo que se llama *reparametrization trick*. El *reparametrization trick* se utiliza para poder diferenciar las variables aleatorias continuas respecto de sus parámetros. En lugar de muestrear directamente esta distribución gaussiana (μ, σ), se puede muestrear una distribución gaussiana estándar $N(0, 1)$, y luego transformar este muestreo mediante la siguiente fórmula:

$$z = \mu + \sigma \odot \epsilon \tag{3.1}$$

Donde $\odot$ denota la multiplicación de cada elemento por ϵ, que es una distribución gaussiana estandar (ruido).

En el caso del decodificador, tiene cinco capas de convolución transpuesta, teniendo la segunda capa un tamaño de stride de 2.

Todas las capas tienen la función de activación relu, menos la última que tiene una función de activación sigmoidea.

Entrenamiento del modelo

En cuanto al entrenamiento, se ha proporcionado un tamaño de espacio latente de 200, y se ha utilizado una función de pérdida un tanto diferente.

Como ya se vio en 2.11, existe una forma más útil de escribir esta fórmula y es agrupar la parte del *prior* y del *decodificador* convirtiéndolo en una divergencia *KL(Kullback-Leibler)*

$$\ln p_\theta(x) \leq E_{z\sim q_\phi(z|x)}[\ln p_\theta(x|z)+\ln p_\theta(z)-\ln q_\phi(z|x)] = -KL(q_\phi(z|x)||p(z))+E_{z\sim q_\phi(z|x)}[\ln p_\theta(x|z)] \tag{3.2}$$

La divergencia *KL (Kullback-Leibler)* es una medida de la diferencia entre dos distribuciones de probabilidad y se escribe así:

$$D_{kl}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{3.3}$$

para valores continuos y discretos.

Esto es útil cuando el valor de KL es fácil de analizar siendo:

$$-KL(q_\phi(z|x)||p(z)) = \frac{1}{2}(1 + \ln \sigma_\phi^2(x) - \sigma_\phi^2(x) - \mu_\phi(x)^2) \tag{3.4}$$

Siendo este valor el valor de pérdida del modelo al que se le va a añadir el error L2 con un factor de balance B para que sea más fiel la salida. Entonces, el error del modelo será:

$$loss = B \cdot l2(x, x') + \left(\frac{1}{2}(1 + \ln \sigma_\phi^2(x) - \sigma_\phi^2(x) - \mu_\phi(x)^2)\right) \tag{3.5}$$

3.5. cGAN: Primera aproximación

Creación de los modelos

Como ya se ha visto, en el apartado 2, las redes GAN se dividen en dos modelos: el generador y el discriminador.

En cuanto al generador, es un tanto distinto a los generadores de las redes GAN, en este caso, se ha decidido usar una red *Conditional Generative Adversarial Network* (cGAN), las redes cGAN se distinguen de las redes GAN principalmente por su generador, el cual es condicionado por una imagen que se llamará *input*, en vez de provenir de una distribución normal Gaussiana.

Esta red comienza con un *downsampling* de cuatro capas con la siguiente estructura: primero una capa de convolución, con el valor de stride a 2 y función de activación relu, y una capa de normalización de lotes. Lo siguiente es la capa de upsampling, la cual tiene la misma estructura que la de downsampling pero en vez de cuatro capas de convolución, la substituye la convolución transpuesta. Al final una capa de *upscaling* la cual dependiendo de la resolución cambia:

- x1: No tiene ninguna capa.
- x2: Tiene una capa de convolución transpuesta, con el valor de stride a 2 y función de activación relu, seguida de una capa de normalización de lotes.
- x4: Tiene dos capas de convolución transpuesta, con el valor de stride a 2 y función de activación relu, seguido de cada una de ellas la normalización de lotes.

Por último hay una capa de convolución transpuesta con el valor de strides a 2 y función de activación sigmoidea.

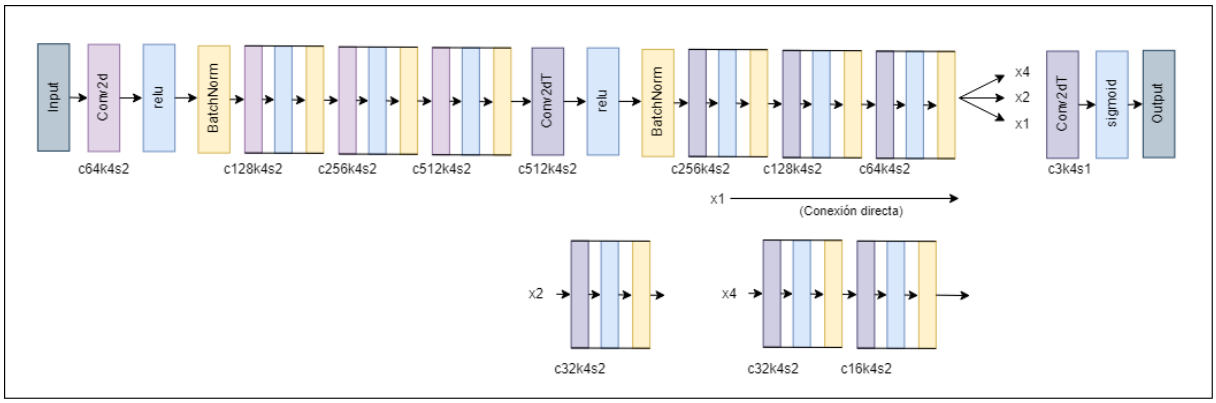


Figura 3.10. Composición del generador. Los valores debajo de cada capa de convolución significan: c el número de filtros, k el tamaño del kernel y s el tamaño del stride. Fuente: Elaboración propia

El discriminador es un discriminador por parches que sigue una estructura específica para todas las redes GAN en adelante. Comienza concatenando dos entradas del mismo tamaño. A continuación, incluye una capa de downsampling que consta de una capa de convolución seguida de una capa de *LeakyReLU*.

Después, se añaden tres capas de downsampling, cada una con la misma estructura: una capa de convolución seguida de una normalización de lotes y, finalmente, una capa de *LeakyReLU*. Posteriormente, se incorpora un padding de ceros y se añade otra capa de downsampling similar a las anteriores pero con el valor de stride a uno. Finalmente, se incluye otra capa de padding con más ceros, seguida de una capa de convolución con un filtro.

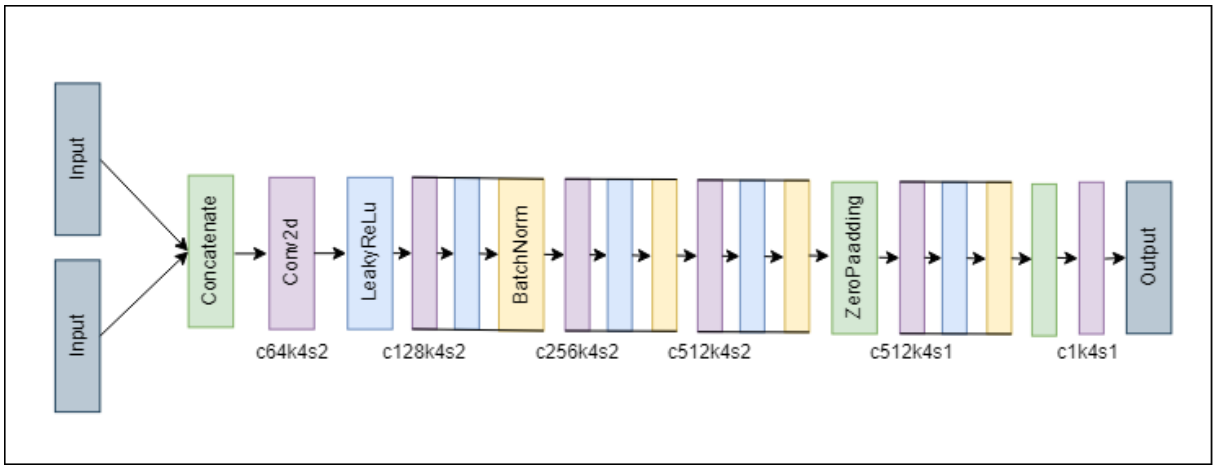


Figura 3.11. Composición del discriminador. Los valores debajo de cada capa de convolución significan: c el número de filtros, k el tamaño del kernel y s el tamaño del stride. Fuente: Elaboración propia

Entrenamiento de los modelos

Cada red tiene una función de pérdida diferente,

3.6. cGAN: (pix2pix)

3.7. EDSR cGAN

4. Configuración de la memoria

La estructura de esta plantilla está dividida en varios ficheros. Aunque puede parecer complicado, en realidad no es tanto:

- `./appendices/`: Los fuentes de los capítulos de apéndices, vamos, los que van al final y que se usan como adenda a la memoria como tal.
- `./chapters/`: Los fuentes de los capítulos que forman parte del cuerpo de la memoria.
- `./figures/`: Las figuras (imágenes, diagramas) que se usarán a lo largo de todo el documento.
- `./frontmatter/`: Los fuentes de todo aquello que se incluye antes del cuerpo de la memoria. Por ahora, todas las entradas del glosario.
- `./sources`: Ficheros con fuentes que se incluyen dentro de listados de fuentes del documento.
- `./references.bib`: Los fuentes en BibTeX de la bibliografía que se referencia en la memoria en la memoria.
- `./report.tex`: El fichero principal a partir del cual se compila todo el proyecto.
- `./upm-report`: El directorio que tiene todo el contenido que hace que la memoria se vea así. Si tocas aquí, que sea con mimo y cariño, porque es muy fácil cargárselo todo.

4.1. ¿Cómo empiezo a escribir la memoria?

Por el principio, es decir, con el fichero `report.tex`. Veamos la primera línea del fichero (listado 4.1).

Listado 4.1. Primeras líneas del fichero `report.tex`

```
\documentclass[%
    school=etsisi,%
    type=pfg,%
    degree=61CI,%
]{upm-report}

\addbibresource{references.bib}
```

En este punto es donde se configura gran parte de la plantilla. Los parámetros y sus opciones son las siguientes:

- `school`: La escuela a la que pertenece el estudiante. La idea de la plantilla es que se use a lo largo de todas las escuelas de la UPM, y que cada una de ellas tenga su propia configuración. La escuela determinará, entre otras cosas, direcciones y colores principales. Las opciones se describen en el Apéndice A.
- `type`: El tipo de memoria. Modifica algunos textos, incluida la portada. Puede tomar los valores `pfg` (Proyecto Fin de Grado) y `pfm` (Proyecto Fin de Máster).
- `degree`: El grado al que aspira el estudiante. Las escuelas y los grados soportados se encuentran en el Apéndice A.

Tras esta configuración, se incluye el fichero de referencias bibliográficas (listado 4.2):

Listado 4.2. Inclusión del fichero de referencias bibliográficas `references.bib`

```
\title{Escalado y Reconstrucción de Imágenes}
```

El tema de las referencias bibliográficas se explica en la Sección 5.9. En principio no habría que tocar nada (bueno sí, añadir las fuentes y referenciarlas), pero si las referencias se tienen en otro fichero, bastaría con cambiar el nombre al de dicho fichero.

Los tres comandos siguientes (listado 4.3) indican el título del proyecto, el nombre del autor y su entrada en la bibliografía, y el nombre del director y su entrada en la bibliografía.

Listado 4.3. Configurando autor, título del proyecto y director

```
\bibauthor{Marín, D.}
\director{Alberto Díaz Álvarez}
\bibdirector{Díaz, A.}
```

Eso de las entradas bibliográficas para el autor y el director no son más que los nombres que aparecen en «cómo citar el proyecto». El día que aprenda cómo hacerlo automáticamente, estas líneas desaparecerán[1].

Tras ello, empieza el primer contenido de verdad: **resumen** y **abstract**, cada uno con sus palabras clave asociadas. Ambos dos son obligatorios y se añaden con la macro `\abstract`, donde se especificarán el idioma (`spanish` o `english`) y el contenido. De la misma manera, las palabras clave se añaden con la macro `\keywords`. Son obligatorios, así que no te los saltes.

Existe la opción de añadir agradecimientos con `\acknowledgements`. Si no se pone no se renderiza en el documento final, pero es algo bonito y a las abuelas les encanta aparecer ahí. Y las abuelas son de lo más bonito que existe en este mundo, así que cuidadlas.

Y ahora sí, empezamos con el documento. Tras el glosario (Sección 5.6), se comenzarán a incluir uno tras otro todos los capítulos de los que se compone nuestra memoria, tal y como se muestra en el listado 4.4.

Listado 4.4. Cómo incluir capítulos y apéndices

```
\input{chapters/introduccion}

\input{chapters/estado-del-arte}
```

[1]Bueno, y si tú, queridísimo lector o lectora sabes cómo hacerlo, haz un *pull request* al repositorio de la plantilla: https://github.com/blazaid/UPM-Report-Template.

La macro `\appendix` del medio indica a partir de qué punto se añaden los apéndices. No son obligatorios, ni mucho menos, pero en algunos PFGs y Proyectos Fin de Máster (PFMs) se suelen incluir para dar información adicional de contexto que no es el objetivo de la memoria, pero sí interesante para complementar. Por ejemplo, en un PFM para el estudio del comportamiento de conductores al volante, uno de los apéndices podría ser cada uno de los formularios que se le ofrecieron para rellenar a cada uno de los conductores de dicho estudio.

Y ya estaría todo. Resumiendo, hay que configurar la plantilla, poner el autor, título y director del proyecto e incluir los capítulos y apéndices que queramos.

4.2. ¿Cómo estructurar la memoria?

La respuesta rápida es «como buenamente quieras/puedas». En realidad la estructura de la memoria va a depender del tipo de trabajo desarrollado, pero con carácter general, los trabajos suelen seguir ciertas estructuras.

Un PFG es un trabajo cuyo propósito es demostrar que se han llegado a adquirir las competencias asociadas con la titulación cursada. Con esto queremos decir que, a diferencia de otros tipos de trabajo académico, en éste no es necesario realizar aportaciones originales al estado de la cuestión.

Una estructura típica es la siguiente:

1. Resumen
2. Introducción
3. Estado de la cuestión
4. Metodología
5. Resultados y Discusión
6. Conclusiones
7. Apéndices

8. Referencias bibliográficas
9. Glosario

Un PFM, a diferencia de un PFG trata de profundizar más en un campo concreto de una disciplina, por lo que tiene a ser más extenso y mucho más específico.

En términos generales, la estructura es similar. Sin embargo es de esperar que el nivel de exigencia sea mayor, ya que el estudiante que lo realiza debe demostrar que es un titulado superior. Esto se nota más en la fase de documentación, ya que al tratar de profundizar en un tema más específico, el trabajo de contextualizar y argumentar es más tedioso.

Se pueden identificar dos tipos de proyectos diferentes, aquellos que podríamos catalogar de *profesionales*, con enfoque a la innovación o mejora en un área profesional concreta, y aquellos *de investigación*, más enfocados a la búsqueda de nuevo conocimiento en el área, y que suelen ser el comienzo de la carrera investigadora.

5. Componentes de la plantilla

En este capítulo hablaremos de los componentes principales con los que trabajaremos en nuestra memoria.

5.1. Columnas

TBD

5.2. Ecuaciones

La facilidad de composición de ecuaciones es una de las cosas que más atrae de LaTeX a muchos autores. LaTeXmantiene dos renderizadores diferentes, uno para el texto y otro para las ecuaciones, denominados modo párrafo y modo matemático[1]. El modo párrafo es el modo por defecto y no se le llama explícitamente. Al modo matemático, sin embargo, se le invoca de varias maneras diferentes.

5.2.1. Modo en párrafo

La forma más común es la forma “en línea”, donde el texto para el modo matemático se encierra entre dos signos $. Por ejemplo, veamos la frase del listado 5.1.

Listado 5.1. Ejemplo de inserción de fórmulas en linea

[1] Existe un tercer modo, denominado *LR mode* o *left-to-right mode*, raramente utilizado y que no trataremos aquí

```
El pequeño teorema de Fermat dice que si $p$ es un número primo
   , entonces, para cada número natural $a$, con $a>0$, $a^p \
   equiv a (\mod p)$
```

La frase quedaría como sigue:

El pequeño teorema de Fermat dice que si p es un número primo, entonces, para cada número natural a, con $a > 0$, $a^p \equiv a(mod p)$

5.2.2. Ecuaciones en bloque

Cuando en lugar de poner una ecuación dentro de un párrafo existente la queremos insertar en su propio espacio independiente hacemos uso de los entorno `equation` o `align`, dependiendo de si queremos una o más ecuaciones en el bloque, respectivamente. Por ejemplo, en el caso de una única ecuación, sería similar al ejemplo siguiente:

```
\begin{equation}
  \int_a^b f'(x) \, dx=f(b)-f(a
     )
\end{equation}
```

$$\int_a^b f'(x)\,dx = f(b) - f(a) \tag{5.1}$$

Sin embargo, en el caso de que quisiésemos más de una ecuación en el mismo bloque haríamos uso del carácter & para indicar en qué punto se alinean las ecuaciones; por ejemplo:

```
\begin{align}
    u  &= \arctan x \\
    du &= \frac{1}{1 + x^2}dx
\end{align}
```

$$u = \arctan x \tag{5.2}$$

$$du = \frac{1}{1+x^2}dx \tag{5.3}$$

Por último, si no tenemos por qué referenciarlas en el texto, podemos hacer uso de los entornos `equation*` y `align*`.

```
\begin{equation*}
    \int_a^b f'(x) \, dx=f(b)-f
      (a)
\end{equation*}
```

$$\int_a^b f'(x)\,dx = f(b) - f(a)$$

En caso de querer generar un índice de ecuaciones del documento, de la misma manera que los listados de Figuras, Tablas y Listados. Para ello, debemos añadir a la lista de ecuaciones la ecuación definida con el comando `myequations`. Es importante destacar que para generar un listado de ecuaciones estas deben estar numeradas, es decir.

```
\begin{equation}
     \frac{x^2}{a^2} - \frac{y
       ^2}{b^2} = 1
\end{equation}
\myequations{Ecuación canónica
   de la hipérbola}
```

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \tag{5.4}$$

5.3. Elementos flotantes

Vamos a hablar un poco de los elementos denominados «flotantes» o *floats*. Éstos son bloques de contenido que «flotan» por la página hasta que LaTeX los coloca donde considera a través de ciertos algoritmos.

Lo general es **declarar el elemento flotante inmediatamente después del párrafo donde se ha referenciado** (las referencias cruzadas las vemos en la Sección 5.8 de este capítulo), y después dejar que LaTeX elija el mejor sitio. Y si después de haber escrito todo el documento hay algo que no cuadre, pues ahí modificarlo.

«¿Y si no hago referencia a un elemento flotante, dónde lo pongo?» Bueno, pues en general no se pone. Si algo no es nombrado, suele ser porque no aporta información, y si no la aporta, pues para qué lo vamos a meter. Ojo, que lo mismo existe algún caso donde sí tiene sentido, pero es muy raro.

Tabla 5.1. Opciones para los elementos flotantes de LaTeX

Especificador	Acción
H	Colocar exactamente en el sitio indicado
h	Colocar aproximadamente en el sitio indicado
t	Colocar al comienzo de la página
b	Colocar al final de la página
p	Colocar en una página exclusiva para elementos “flotantes”
!	Forzar las opciones obviando los mecanismos internos de LaTeX

Aunque más adelante veremos los diferentes tipos de *floats*, en caso de que queramos modificar su comportamiento todos tienen los especificadores de posición indicados en la tabla 5.1:

5.3.1. Código fuente

Para la gestión de los listados de código fuente se utiliza el paquete `listings`. El estilo usado es una modificación[2] de Solarized desarrollado por Ethan Schoonover.

Existen muchas formas diferentes de incluir listados de código en una memoria. Aquí introducimos los más comunes.

Código dentro del propio párrafo

TBD

Bloques de código

Insertar código en párrafos no es tan común como insertar bloques enteros. Para ello haremos uso del entorno `lstlisting`. Por ejemplo:

[2] En realidad la modificación es cambiar el fondo de crema a blanco

```
\begin{lstlisting}
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría el siguiente resultado:

```
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

Una cosa que hay que tener en cuenta es que dentro de un entorno `lstlisting` se ignoran todos los comandos de LaTeX [3] y el texto se imprime tal y como se ha introducido. Esto incluye los tabuladores y espacios de principio de línea.

Al igual que en el código de párrafo, también podemos especificar en qué lenguaje está escrito el código para que se resalten en éste las palabras reservadas. Por ejemplo:

```
\begin{lstlisting}[language=python]
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría como resultado el siguiente bloque de código:

```
from collections import Counter
```

[3] En realidad no todos, y si no mira en estos fuentes cómo hemos metido el fin de bloque `lstlisting` dentro del propio bloque.

```
def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

Código directamente desde fichero

Esta forma es muy común, ya que se usa tanto para hacer referencia al código fuente de la aplicación directamente, como a código separado en ficheros para mantener el tamaño de la memoria manejable.

Suponiendo que tenemos el fichero `sources/snippets.py`, para incluirlo entero basta con usar el comando `lstinputlisting`:

```
\lstinputlisting[language=python]{sources/snippets.py}
```

Con este comando conseguiríamos el siguiente resultado:

```
def all_unique(l):
    return len(l) == len(set(l))

def is_palindrome(l):
    return l == l[::-1]
```

En caso de que se desease importar sólo parte del fichero, se pueden indicar las filas que delimitan el trozo de código. Por ejemplo:

```
\lstinputlisting[
    language=python,
    firstline=4,
    lastline=5
]{sources/snippets.py}
```

Esto haría que sólo se imprimiesen las filas 4 y 5, correspondientes a la segunda función:

```
def is_palindrome(l):
    return l == l[::-1]
```

Ambas opciones son opcionales, y los valores por defecto de `firstline` y `lastline` serán el principio y el final del fichero respctivamente.

Etiquetando bloques de código

Al igual que con el resto de bloques *float* (e.g. figuras o tablas), se pueden[4] (**deben**) añadir pies de texto a los bloques de código, lo cual hace más legible su cometido.

Para ello basta con añadir el argumento `caption` a las opciones del bloque. Por ejemplo:

```
\begin{lstlisting}[language=python, caption=Función para
   determinar cuando una palabra \textt{w1} es anagrama de otra
    palabra \textt{w2}]
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría como resultado el siguiente bloque de código:

Listado 5.2. Función para determinar cuando una palabra `w1` es anagrama de otra palabra `w2`

```
from collections import Counter

def is_anagram(w1, w2):
```

[4] Pongo *pueden* porque es opcional, pero en realidad se **deben** poner, porque si no los listados de fuentes quedan horribles, como el de esta plantilla por ejemplo (échale un vistazo si no lo has hecho antes).

```
    return Counter(w1) == Counter(w2)
```

5.3.2. Figuras

El código siguiente (listado 5.3) renderizará una imagen.

Listado 5.3. Inserción de una figura

```
\begin{figure}[H]
    \centering
    \includegraphics[width=0.25\textwidth]{figures/vault-boy.
       png}
    \caption{\label{fig:img-vault-boy} Vault Boy approves that}
\end{figure}
```

La sintaxis es bastante autoexplicativa. El entorno `figure` es el que delimita el contenido de la figura. El comando `centering` determina que se tiene que centrar . Luego, el comando `caption` determina el pie de imagen, el cual además incluye una etiqueta (comando `label`) que sirve para referenciar. Por último, se incluye la imagen con el comando `includegraphics`.

En definitiva, la imagen (figura 5.1) se mostrará con un ancho igual al 25 % del ancho que ocupa el texto, centrada, con un pie de foto y una etiqueta para referenciar.

Figura 5.1. Vault Boy approves that

El comando `includegraphics` puede importar los formatos típicos de ima-

gen, como jpeg, png o pdf. También admite una serie de opciones como rotación, alto, ancho (éste le hemos especificado con `width`), etcétera. ¡Ojo! Siempre que se pueda, hay que intentar insertar imágenes vectoriales. De esta manera, se mantiene la calidad de la imagen. Si no, puede ocurrir que se pixele y no quede nada bien.

Subfiguras

¿Y qué pasa cuando queremos incluir múltiples imágenes dentro de una figura? Bueno, pues aquí hay que usar el entorno `subfigure`. En el listado 5.4 vemos un ejemplo de cómo se manejan.

Listado 5.4. Inserción de varias subfiguras

```
\begin{figure}[H]
  \centering
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\linewidth]{figures/vault-boy.png}
    \caption{\label{fig:subfigure-1}Vault Boy 1}
  \end{subfigure}%
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\textwidth]{figures/vault-boy.png}
    \caption{Vault Boy 2}
  \end{subfigure}%
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\textwidth]{figures/vault-boy.png}
    \caption{Vault Boy 3}
  \end{subfigure}
  \caption{\label{fig:subfigures}Todos los Vault Boy}
\end{figure}
```

En realidad cada subfigura se trata como una figura normal, pero en relación con el *float* contenedor. Cuando a una subfigura se le especifica un ancho, se le está diciendo al compilador de qué ancho es esa subfigura en concreto (en nuestro caso 0.3 veces el ancho de la línea). Sin embargo, a la imagen se le da un ancho total de `linewidth`, porque al estar dentro de su espacio

de subfigura, el ancho ha cambiado. El resultado es el que se observa en la figura 5.2. Por cierto, también podemos referenciar a los pies de las subfiguras (e.g. Así: 5.2a).

(a) Vault Boy 1 (b) Vault Boy 2 (c) Vault Boy 3

Figura 5.2. Todos los Vault Boy

5.3.3. Tablas

Las tablas (en realidad «cuadros») son una forma muy eficaz de presentar información. En los resultados de casi cualquier trabajo existen cuadros de algún tipo para que los datos se comprendan de un único vistazo (o para que al menos sea más fácil identificarlos.

¿Por qué "cuadro" en lugar de "tabla"?
Tal y como se indica en el FAQ de CervanTex, *table* (inglés) y *tabla* (español) son falsos amigos; el inglés *table* tiene un sentido más general que el español *tabla*, cuyo uso es únicamente para aquellos cuadros dedicados a la disposición de números (e.g. tabla de multiplicar o tabla de logaritmos).

Sin embargo, las tablas suelen ser bastante complicadas en LaTeX, por lo menos para la gente que empieza. Para no escribir demasiado, la respuesta para casi toda maquetación de tabla está en https://www.tablesgenerator.com/. En serio, no perdáis el tiempo si no es estrictamente necesario. La maquetáis visualmente, la generáis (con estilo *booktabs*) y a correr.

5.4. Enlaces de hipertexto

TBD

5.5. Fórmulas matemáticas

TBD

5.6. Glosario

El glosario de una memoria es el lugar donde se encuentran los términos que se usan a lo largo del documento y que se considera que requieren una aclaración. En esta plantilla, en el momento que generemos un término, se creará un capítulo al final de la memoria con el listado de todos aquellos términos definidos.

Para gestionar el glosario se hace uso del paquete `glossaries` el cual es relativamente complejo de configurar. También su documentación es muy extensa[5], así que en esta sección hablaremos únicamente de lo esencial.

5.6.1. ¿Cuándo y cómo especificar términos?

La regla general del «cuándo» es una vez terminada la memoria. En ese punto, seremos conscientes de qué términos son los más interesantes para incluir en el glosario. En ese punto deberemos ir término por termino sustituyéndolo por la entrada del glosario para que el proceso automático se encargue de la indexación y numeración de páginas.

El «cómo» se refiere a de qué manera escribirlos. La regla general en el cas-

[5] Pero aún así es el sitio donde ir a buscar información. El paquete, junto con su documentación está disponible en la dirección https://www.ctan.org/pkg/glossaries.

tellano (y hasta donde el autor de la plantilla sabe, en cualquier idioma) es de la manera en la que aparecería en medio del texto. Es decir, si la palabra se escribe generalmente en minúscula (e.g. *El jugador blandía un hacha de batalla*) se deberá incluir dentro del glosario en minúscula, mientras que si se escribe generalmente en mayúscula (e.g. *Encontró el Arco de la Perdición*) irá en mayúscula.

5.6.2. Definiendo los términos del glosario

Las entradas se escribirán dentro del fichero `frontmatter/glossary.tex`. La forma estándar de definir un término es la que se muestra en el listado 5.5.

Listado 5.5. Código para crear una entrada en el glosario

```
\newglossaryentry{hacha-batalla}{
    name={hacha de batalla},
    description={Herramienta antigua utilizada en combate}
}
```

Luego, dentro del texto, podremos hacer referencia a dichas entradas con los comandos que se muestran en la tabla 5.2.

Tabla 5.2. Comandos para incluir términos del glosario en el texto de la memoria

Comando	Ejemplo con la clave **`hacha-batalla`**
`\gls`	hacha de batalla
`\Gls`	Hacha de batalla
`\glspl`	hacha de batallas
`\Glslp`	Hacha de batallas

Como los plurales los gestiona automáticamente, puede ser que queramos, como en este caso, modificar el plural de nuestro término. Para ello debemos añadir la opción `plural` a la entrada para especificar cómo es el plural de la entrada, como se muestra en el listado 5.6.

Listado 5.6. Especificando el plural para un término del glosario

```
\newglossaryentry{python}{
    name={Python},
    plural={Pythonacos},
    description={El mejor lenguaje de programación}
}
```

Así, el plural de la clave `python` descrita quedaría como Pythonacos, en lugar del valor por defecto que sería *Pythons*.

Un caso particular de términos del glosario son las siglas y los acrónimos. No vamos a entrar en detalle aquí[6] sino que vamos a introducir las siglas como caso especial de entrada de glosario. Cuando tengamos una sigla, la crearemos en el glosario como se muestra en el listado 5.7.

Listado 5.7. Entrada genérica de una sigla o acrónimo en el glosario

```
\newacronym[
    description={Proyecto Fin de Grado. Proyecto a realizar al
       final de una titulación de Grado},
    longplural={Proyectos Fin de Grado}
    ]{pfg}{PFG}{Proyecto Fin de Grado}
```

En el ejemplo se puede ver que hay dos entradas, `longplural` y `description` que son opcionales. La primera es la equivalente a `plural` de `newglossaryentry`, y no necesita más explicación.

La segunda, `description` suele utilizarse para acrónimos, cuando necesitamos describir la entrada. Cuidado en este caso porque si hace referencia a varias palabras estas se deberían incluir dentro de la descripción (como en el ejemplo, «Proyecto Fin de Grado»).

La regla general de los acrónimos y las siglas es que la primera vez que aparecen en el texto, deben aparecer con el nombre completo mientras que el

[6] Pero recomendamos visitar https://www.fundeu.es/recomendacion/siglas-y-acronimos-claves-de-redaccion/ y darle una leída porque es interesante.

resto de veces pueden aparecer indistintamente como sigla o forma larga. De esto se encarga automáticamente el comando `gls`. Es decir, si tenemos la sigla `special`, la primera vez que incluyamos la sigla con `\gls{special}` saldrá *Strenght, Perception, Endurance, Charisma, Intelligence, Agility & Luck* (SPECIAL) mientras que el resto de veces que la incluyamos se verá simplemente SPECIAL.

Con los acrónimos se incluyen comandos adicionales para controlar su presentación. Estos son los mostrados en la tabla 5.3

Tabla 5.3. Comandos específicos para controlar la presentación de acrónimos

Comando	Ejemplo con la clave `rpg`
`\acrshort`	RPG
`\acrshortpl`	RPG
`\acrlong`	*Role-Playing Game*
`\Acrlong`	*Role-Playing Game*
`\acrlongpl`	*Role-Playing Games*
`\Acrlongpl`	*Role-Playing Games*
`\acrfull`	*Role-Playing Game* (RPG)
`\Acrfull`	*Role-Playing Game* (RPG)
`\acrfullpl`	*Role-Playing Games* (RPG)
`\Acrfullpl`	*Role-Playing Games* (RPG)

Por cierto, en castellano las siglas **no incluyen la «s» al final**, así que no deberíamos usar los comandos que terminan en `pl`. Por eso la definición que se ha hecho de la sigla `rpg` es la mostrada en la figura 5.8.

Listado 5.8. Entrada de `rpg` en `glossaries.tex`

```
\newacronym[
    description={Role-Playing Game. Juego de rol},
    shortplural={RPG}
    ]{rpg}{RPG}{\textit{Role-Playing Game}}
```

5.7. Notas

TBD

5.8. Referencias cruzadas

Las etiquetas (*label*) son una herramienta muy útil en el proceso de composición tipográfica. Se puede pensar en ellas como punteros a zonas de interés del documento, de tal manera que se les pueda referenciar sin necesidad de conocer su posición final en la composición.

Por ejemplo, lo normal es que en un libro, a la hora de referenciar una figura, aparezca una frase del estilo "[...] como muestra la Figura 3 [...]". Lo que es bastante raro son las frases del estilo "[...] como muestra la Figura de los moñecos amarillos [...]" o "[...] como muestra la siguiente Figura [...]"[7].

Una de las propiedades más útiles y, en ocasiones, infravaloradas de LaTeXes la facilidad y potencia de su sistema de etiquetado. Este sistema permite referenciar tablas, listados de código fuente, ecuaciones, capítulos, secciones, etc., con facilidad y flexibilidad. Además, LaTeXlas numera y referencia automáticamente, cambiando la numeración en función de las adiciones y supresiones sin que el autor tenga que hacer nada.

Para referenciar un elemento, lo primero que hay que crear es una etiqueta **después** del elemento a referenciar. Esto ya lo hemos visto anteriormente, por ejemplo en el listado 5.3. Si nos fijamos, se declara una etiqueta justo después de la etiqueta `caption` con el nombre `fig:img-vault-boy`. De esta manera, podemos referenciar varios indicadores de la figura, como se muestra en el listado 5.9

Listado 5.9. Referenciando una figura y su página

[7] Sí, bueno, quizá la segunda frase no es tan rara, pero siempre es preferible referenciar directamente a dar posiciones relativas.

```
Mira la Figura~\ref{fig:img-vault-boy} en la página~\nameref{
   fig:img-vault-boy}.
```

Dicho listado daría el siguiente resultado:

«Mira la Figura 5.1 titulada Vault Boy approves that en la página 44.»

¿Por qué a mí me aparece el símbolo ?? en lugar de una referencia? Pues lo más seguro es que sea un error a la hora de escribir la etiqueta. Menos común, pero también puede pasar, es que el documento no se haya compilado bien. Hay que tener en cuenta que LaTeX fue creado en una época donde las máquinas tenían poca (¡poquísima!) RAM, y para funcionar lo que se hacían eran varias compilaciones sobre el documento, almacenando los valores temporales en ficheros. Y como nadie quiere perder tiempo en cambiar y *debuggear* algo que funciona estupendamente bien, no se reimplementa. De todas formas, si te animas, ahí tienes un buen proyecto que si lo sacas adelante te va a hacer muy famoso.

5.9. Referencias bibliográficas

Hay muchas formas diferentes de gestionar las referencias bibliográficas, así que aquí hemos decidido elegir una de ellas por considerarla la más cómoda y simple, que es mediante el paquete *biblatex*.

El fichero de referencias, `references.bib`, incluirá una entrada por cada una de las referencias que se citan durante la memoria. Luego, en el cuerpo del texto, se podrán hacer referencias a dichas entradas y será LaTeX después quien se encargue de indexar correctamente, crear los hipervínculos y maquetar automáticamente.

El fichero `references.bib` puede tener muchas más de las referencias que se citan en el cuerpo del texto. Sin embargo, sólo aparecerán las referencias que se citen en el texto.

No has dicho en ningún momento *bibliografía* Sí. Las referencias bibliográficas, también conocidas como lista de referencias o simplemente referencias, son todas aquellas fuentes bibliográficas que han sido citadas a lo largo del documento. La bibliografía, también conocida como referencias externas, es simplemente una lista de recursos utilizados, citados o no. Como generalmente los no referenciados no se usan para dar soporte a un texto científico se suelen descartar.

5.9.1. ¿Cómo creamos nuevas referencias?

El fichero `references.bib` contará cero o más entradas con la estructura mostrada en el listado 5.10.

Listado 5.10. Estructura general de una referencia

```
@tipo{id,
    author = "Autor",
    title = "Título de la referencia (libro, artículo, enlace,
       ...)",
    campo1 = "valor",
    campo2 = "valor",
    \ldots
}
```

En esta entrada, `@tipo` indica el tipo de elemento (p. ej. `@article` para artículos o `@book` para libros) e `id` es un identificador **único en todo el documento** para el elemento. El resto de campos dependerán del tipo de la referencia, aunque generalmente casi todos los tipos comparten los campos de `author`, `title` o `year`.

5.9.2. ¿De qué manera puedo citar las referencias?

TBD **mcculloch1943logical**

5.10. Referencias cruzadas

TBD

5.11. Referencias a recursos externos

TBD

6. Licencia

Cuando se publica la obra en el archivo digital, por defecto lo hace con la licencia de *Creative Commons* Reconocimiento - Sin obra derivada - No comercial. Aunque usar esta licencia es correcto, no es una licencia libre y a algunos nos parece algo malo.

Considero que todo el conocimiento generado en una universidad pública ha de ser público y libre. Por ello esta obra se publica con la licencia *Creative Commons* Reconocimiento - Sin obra derivada - No comercial - Compartir igual, de tal manera que se garantiza que la obra se comparte con las mismas libertades y así, todo el mundo puede hacer de esta obra el uso que quiera.

A. Escuelas y títulos

A continuación se describen todas las opciones de grados y títulos disponibles en la memoria.

A.1. Escuelas

Las escuelas disponibles se describen en el cuadro A.1.

Clave	Valor
`etsiaab`	E.T.S. de Ingeniería Agronómica, Alimentaria y de Biosistemas
`etsidi`	E.T.S. de Ingeniería y Diseño Industrial
`etsisi`	E.T.S. de Ingeniería de Sistemas Informáticos

Tabla A.1. Relación entre el código de la plantilla y la escuela a la que se refiere

De momento no están todas, así que si te apetece añadir la tuya puedes, o bien contactar con los autores, o bien modificarlo (mira el apéndice B) y también contactar con los autores, así lo podemos hacer público con la mayor cantidad de usuarios posible.

A.2. Titulaciones

Cada una de las escuelas poseen ciertas titulaciones que se han de añadir a la configuración.

A.2.1. E.T.S. de Ingeniería Agronómica, Alimentaria y de Biosistemas

La ETSIAAB tiene configurados como colores principal y de link el RGB $(99, 178, 76)$. El logo es el mostrado la figura A.1.

Figura A.1. Logo de la ETSIAAB utilizado en la cubierta trasera de la memoria

Las titulaciones que existen la última vez que se actualizó este documento son las siguientes:

- `20BT`: Grado en Biotecnología,
- `20BI`: Grado en Ciencias Agrarias y Bioeconomía,
- `20IG`: Grado en Ingeniería Agrícola,
- `02IA`: Grado en Ingeniería Agroambiental,
- `20IA`: Grado en Ingeniería Alimentaria.

A.2.2. E.T.S. de Ingeniería y Diseño Industrial

La ETSIDI tiene configurados como colores principal y de link el RGB $(223, 30, 64)$. El logo es el mostrado la figura A.2.

Las titulaciones que existen la última vez que se actualizó este documento son las siguientes:

- `56IE`: Grado en Ingeniería Eléctrica,
- `56IA`: Grado en Ingeniería Electrónica Industrial y Automática,

Figura A.2. Logo de la ETSIDI utilizado en la cubierta trasera de la memoria

- `56IM`: Grado en Ingeniería Mecánica,
- `56IQ`: Grado en Grado en Ingeniería Química,
- `56DD`: Grado en Ingeniería en Diseño Industrial y Desarrollo de Producto.

A.2.3. E.T.S. de Ingeniería de Sistemas Informáticos

La ETSISI tiene configurados como colores principal y de link el RGB $(0, 177, 230)$. El logo es el mostrado la figura A.3.

Figura A.3. Logo de la ETSISI utilizado en la cubierta trasera de la memoria

Las titulaciones que existen la última vez que se actualizó este documento son las siguientes:

- `61CD`: Grado en Ciencia de Datos e Inteligencia Artificial,
- `61CI`: Grado en Ingeniería de Computadores,
- `61IW`: Grado en Ingeniería del Software,
- `61SI`: Grado en Sistemas de Información,
- `61TI`: Grado en Tecnologías para la Sociedad de la Información.

B. ¿Cómo ampliar la plantilla?

TBD

C. Lista de paquetes incluidos

TBD

Bibliografía

[1] H.-M. Zhang y B. Dong, «A Review on Deep Learning in Medical Image Reconstruction,» *Journal of the Operations Research Society of China*, vol. 8, n.º 2, págs. 311-340, ene. de 2020, ISSN: 2194-6698. DOI: 10.1007/s40305-019-00287-4. dirección: http://dx.doi.org/10.1007/s40305-019-00287-4.

[2] A. V M y J. Mathew, «A Review on Image Restoration in Medical Images,» *COMPUSOFT: An International Journal of Advanced Computer Technology*, vol. 5, n.º 4, págs. 1588-1591, 2024.

[3] R. Molina, J. Nunez, F. Cortijo y J. Mateos, «Image restoration in astronomy: a Bayesian perspective,» *IEEE Signal Processing Magazine*, vol. 18, n.º 2, págs. 11-29, 2001. DOI: 10.1109/79.916318.

[4] Y. Fujii, T. Ito, N. Ohta, S. Saitoh, T. Matsuura y T. Yamamoto, «Importance of Developing Image Restoration Techniques for Security Cameras under Severe Conditions,» en *2006 SICE-ICASE International Joint Conference*, 2006, págs. 2560-2563. DOI: 10.1109/SICE.2006.314757.

[5] Rice University. «Introduction to Digital Image Processing.» Accessed: 2024-05-15. (1999), dirección: https://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/intro.html.

[6] NVIDIA. «DLSS: Superescalado de IA para juegos.» Accessed: 2024-05-15. (), dirección: https://www.nvidia.com/es-es/geforce/technologies/dlss/.

[7] S. Boudreau. «5 Types of Medical Imaging.» (2023), dirección: https://www.visiblebody.com/blog/5-types-of-medical-imaging.

[8] W. Ahmad, H. Ali, Z. Shah y S. Azmat, «A new generative adversarial network for medical images super resolution,» *Scientific Reports*, vol. 12, n.º 1, jun. de 2022, ISSN: 2045-2322. DOI: 10.1038/s41598-022-13658-4. dirección: http://dx.doi.org/10.1038/s41598-022-13658-4.

[9] Sony. «4K Upscaling Upgrades Surgical Imaging.» Accessed: 2024-05-15. (), dirección: https://pro.sony/ue_US/healthcare/imaging-innovations-insights/4k-upscaling-upgrades-surgical-imaging.

[10] ImageJ. «ImageJ2.» Accessed: 2024-05-16. (), dirección: https://imagej.net/software/imagej2/.

[11] Z. Li, Q. Peng, B. Bhanu, Q. Zhang y H. He, «Super resolution for astronomical observations,» *Astrophysics and Space Science*, vol. 363, n.º 5, abr. de 2018, ISSN: 1572-946X. DOI: 10.1007/s10509-018-3315-0. dirección: http://dx.doi.org/10.1007/s10509-018-3315-0.

[12] AMD. «FidelityFX Super Resolution.» Accessed: 2024-05-15. (), dirección: https://www.amd.com/es/products/graphics/technologies/fidelityfx/super-resolution.html.

[13] M. Banham y A. Katsaggelos, «Spatially adaptive wavelet-based multiscale image restoration,» *IEEE Transactions on Image Processing*, vol. 5, n.º 4, págs. 619-634, 1996. DOI: 10.1109/83.491338.

[14] Rice University. «Introduction to Digital Image Processing.» Accessed: 2024-05-15. (1999), dirección: https://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/inverse.html.

[15] Rice University. «Introduction to Digital Image Processing.» Accessed: 2024-05-15. (1999), dirección: https://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/blind/bd.html.

[16] MathWorks. «Interpolation.» Accessed: 2024-05-17. (), dirección: https://es.mathworks.com/help/matlab/interpolation.html.

[17] Biografías y Vidas. «Fary.» Accessed: 2024-05-17. (), dirección: https://www.biografiasyvidas.com/biografia/f/fary.htm.

[18] M. Anthony, *Discrete Mathematics of Neural Networks: Selected Topics*. Philadelphia, PA: Society for Industrial y Applied Mathematics (SIAM), 2001, ISBN: 978-0-89871-480-7.

[19] IBM, *Multilayer Perceptron Architecture*, Accessed: 2024-05-19, 2024. dirección: https://www.ibm.com/docs/es/spss-statistics/saas?topic=perceptron-architecture-multilayer.

[20] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. dirección: https://www.deeplearningbook.org/contents/mlp.html.

[21] SuperDataScience. «Convolutional Neural Networks (CNN) Step 3: Flattening.» Accessed: 2024-05-18. (2023), dirección: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening.

[22] K. Fukushima, «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,» *Biological Cybernetics*, vol. 36, n.º 4, págs. 193-202, 1980. dirección: https://www.cs.princeton.edu/courses/archive/spr08/cos598B/Readings/Fukushima1980.pdf.

[23] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, «Gradient-Based Learning Applied to Document Recognition,» *Proceedings of the IEEE*, vol. 86, n.º 11, págs. 2278-2324, 1998. dirección: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf.

[24] D. Calvo. «Red Neuronal Convolucional CNN.» Accessed: 2024-05-18. (2017), dirección: https://www.diegocalvo.es/red-neuronal-convolucional/.

[25] M. Deb, A. Garai, A. Das y K. G. Dhal, «LS-Net: a convolutional neural network for leaf segmentation of rosette plants,» *Neural Computing and Applications*, vol. 34, n.º 21, págs. 18 511-18 524, jun. de 2022, ISSN: 1433-3058. DOI: 10.1007/s00521-022-07479-9. dirección: http://dx.doi.org/10.1007/s00521-022-07479-9.

[26] Gamco. «Arquitectura de Red Neuronal.» Fecha de acceso: 2024-05-18. (), dirección: %5Curl%7Bhttps://gamco.es/glosario/arquitectura-de-red-neuronal/#:~:text=La%20arquitectura%20de%20red%20neuronal,y%20funcionamiento%20del%20cerebro%20humano.%7D.

[27] M. A. Kramer, «Nonlinear Principal Component Analysis Using Autoassociative Neural Networks,» *Laboratory for Intelligent Systems in Process Engineering, Dept. of Chemical Engineering*, 1991.

[28] U. Michelucci, «An Introduction to Autoencoders,» *arXiv preprint arXiv:2201.03898*, 2022, Available at https://arxiv.org/abs/2201.03898.

[29] D. P. Kingma y M. Welling, *Auto-Encoding Variational Bayes*, 2022. arXiv: 1312.6114 [stat.ML].

[30] D. P. Kingma y M. Welling, «An Introduction to Variational Autoencoders,» *Foundations and Trends® in Machine Learning*, vol. 12, n.º 4, págs. 307-392, 2019, ISSN: 1935-8245. DOI: 10.1561/2200000056. dirección: http://dx.doi.org/10.1561/2200000056.

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., *Generative Adversarial Networks*, 2014. arXiv: 1406.2661 [stat.ML].

[32] W. contributors, *Diagrams.net – Wikipedia, The Free Encyclopedia*, Accessed: 2024-05-19, 2024. dirección: https://en.wikipedia.org/wiki/Diagrams.net.

[33] W. contributors, *Python*, Accessed: 2024-05-19, 2024. dirección: https://es.wikipedia.org/wiki/Python.

[34] W. contributors. «TensorFlow.» Accessed: 2024-05-19. (), dirección: https://es.wikipedia.org/wiki/TensorFlow.

[35] «TensorFlow.» Accessed: 2024-05-19. (), dirección: https://www.tensorflow.org/?hl=es-419.

[36] Keras Contributors. «Keras.» Accessed: 2024-05-19. (), dirección: https://keras.io/.

[37] NumPy Contributors, *NumPy*, Accessed: 2024-05-19. dirección: https://numpy.org/.

[38] W. contributors, *Anaconda (distribución de Python) – Wikipedia, La Enciclopedia Libre*, Accessed: 2024-05-19, 2024. dirección: https://es.wikipedia.org/wiki/Anaconda_(distribuci%C3%B3n_de_Python).

[39] E. de Data Scientist en Python, *Entorno de desarrollo Anaconda*, Accessed: 2024-05-19, 2024. dirección: https://entrenamiento-data-scientist-python.readthedocs.io/leccion1/anaconda.html.

[40] A. Potti. «How to instal VSC?» Accessed: 2024-05-19. (), dirección: https://community.dynamics.com/blogs/post/?postid=ca1db7e7-50d3-4ea3-9b8a-4dfa8e7a943e.

[41] Wikipedia contributors, *Proyecto Jupyter*, Accessed: 2024-05-19. dirección: https://es.wikipedia.org/wiki/Proyecto_Jupyter.

[42] Microsoft Corporation. «Microsoft Excel.» Accessed: 2024-05-19. (), dirección: https://www.microsoft.com/es-es/microsoft-365/excel.

[43] Wikipedia contributors, *Microsoft Excel*, Accessed: 2024-05-19. dirección: https://es.wikipedia.org/wiki/Microsoft_Excel.

[44] E. Agustsson y R. Timofte, «NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study,» en *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, jul. de 2017. dirección: https://data.vision.ee.ethz.ch/cvl/DIV2K/.

[45] E. Agustsson y R. Timofte, «NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study,» en *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, págs. 1122-1131. DOI: 10.1109/CVPRW.2017.150.

[46] blzq, *TensorFlow (Python) implementation of Gaussian blur of image with variable input kernel size and sigma*, Accessed: 2024-05-19, 2023. dirección: https://gist.github.com/blzq/c87d42f45a8c5a53f5b393e27b1f5

Índice de términos

Glosario

Siglas

MAE Error Absoluto Medio o error l1. 9

MSE Error Cuadrático Medio o error l2. 9

PFG Proyecto a realizar al final de una titulación de Grado. 1–4, 10, 17, 19, 21, 22, 35

PFM Proyecto a realizar al final de una titulación de Máster. 4, 19, 21, 22

PSE PSE (Equalización del Espectro de Potencia o Power Spectrum Equalization en inglés) es una técnica usada para restaurar señales deterioradas. 5

RPG Role-Playing Game. Juego de rol. 36

SPECIAL S.P.E.C.I.A.L es la sigla usada para los atributos de Fuerza (**S**trenght), Percepción (**P**erception), Resistencia (**E**ndurance), Carisma, (**C**harisma), Inteligencia, (**I**ntelligence), Agilidad, (**A**gility), y Suerte (**L**uck). 36

TAC TAC (Tomografía Axial Computarizada) utiliza rayosX para obtener imágenes detalladas del cuerpo humano. 2

VAE Los codificadores automáticos variables son un tipo de red de neuronas. 14

Universidad Politécnica de Madrid

ETSI **SISTEMAS INFORMÁTICOS**