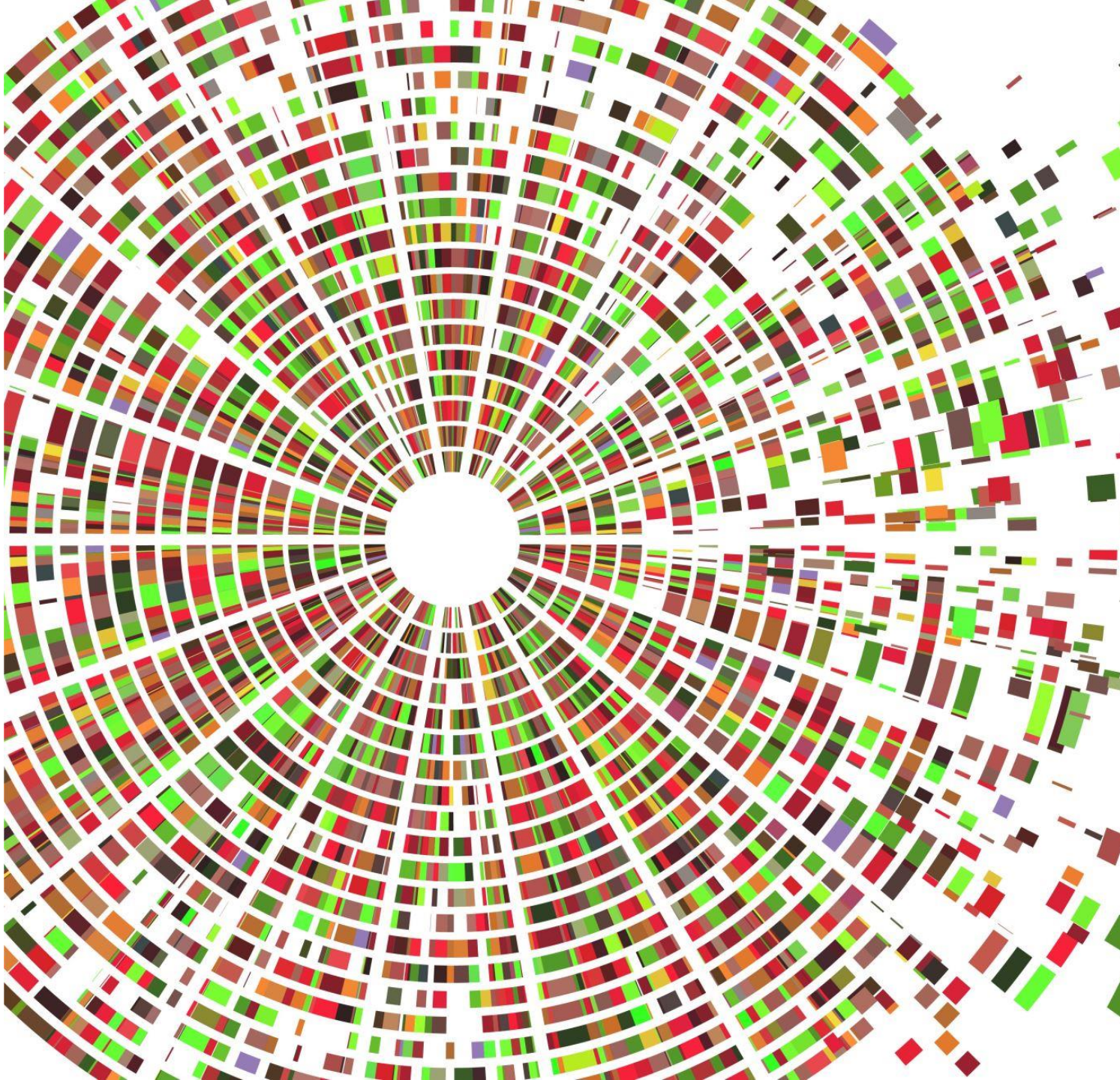
Métodos avanzados de ciencia de datos

Prof. Emily Díaz



Contenido

Principios de procesamiento de imágenes

Casos de uso

Análisis de imagenes clásicos

Redes neuronales aplicadas a imagenes

Principios de procesamiento de imagenes

¿Qué es una imagen?

¿Qué es procesamiento de imágenes?

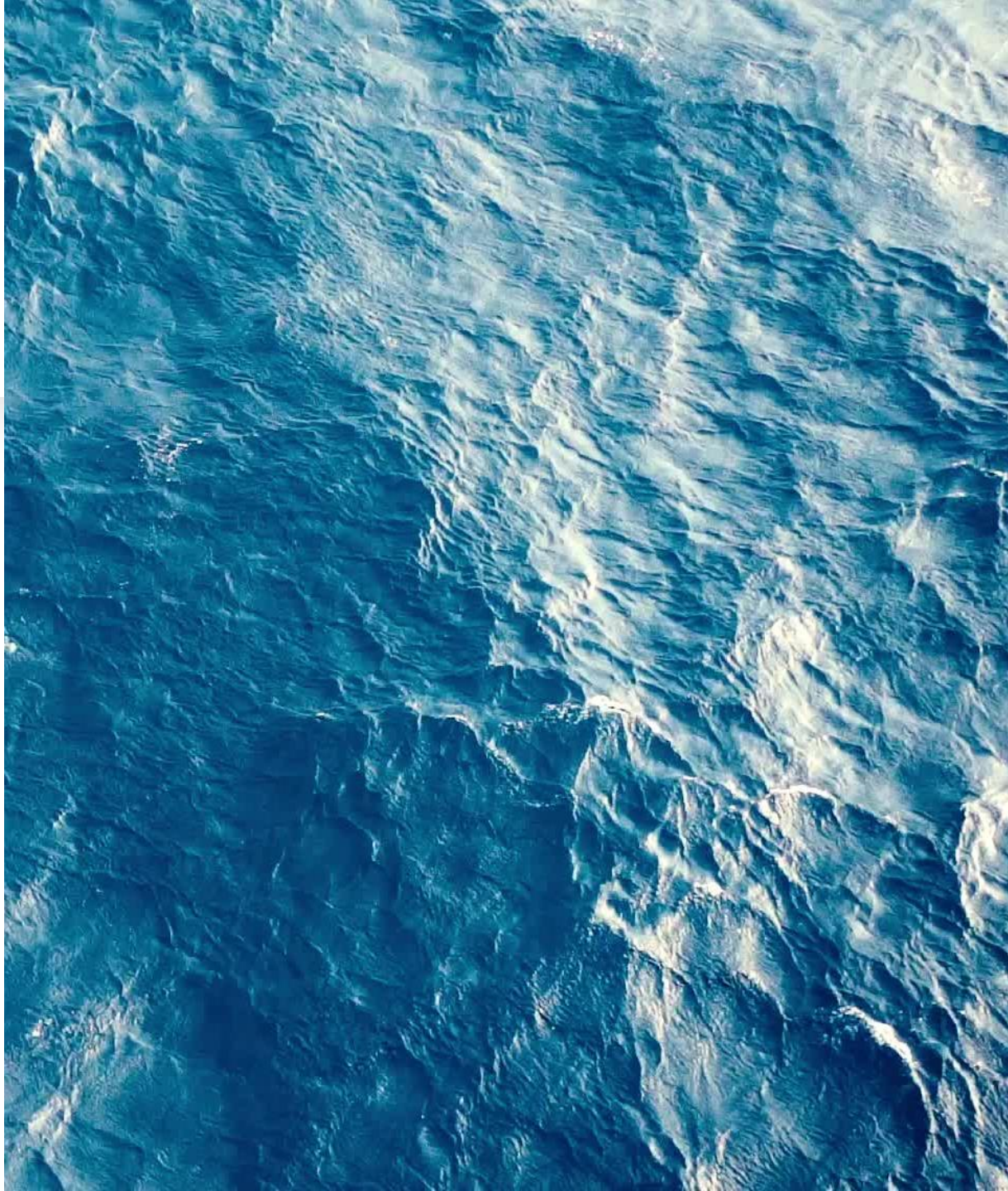
- Proceso de **transformar** una imagen en formato digital y realizar determinadas operaciones para **obtener información útil de ella**.
- Algunos de los tipos de procesamiento de imágenes son:
  - **Visualización:** Encontrat objetos que no son visibles en la imagen
  - **Reconocimiento:** Distinguir o detectar objetos en la imagen
  - **Nitidez y restauración:** Crear una imagen mejorada a partir de la imagen original
  - **Reconocimiento de patrones:** Medir los distintos patrones que rodean los objetos en la imagen
  - **Recuperación:** Explorar y buscar imágenes en una gran base de datos de imágenes digitales que sean similares a la imagen original

¿Por qué es importante?

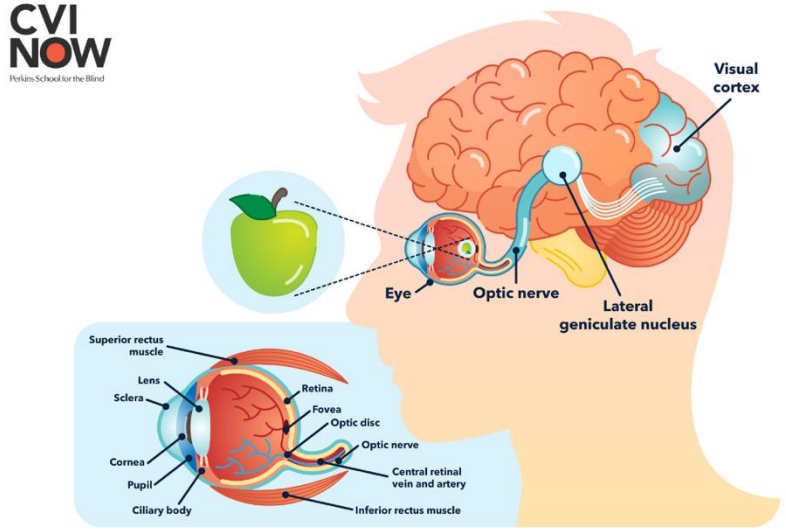
- Visión es uno de los sentidos que más usamos los humanos. Un robot no entiendo naturalmente lo que “ve”
- Cómo enseñarle al robot a reconocer y comprender el contenido de una imagen? La visión artificial **no cambia la imagen**, sino que **intenta darle sentido**, de forma muy similar a cómo nuestro cerebro interpreta lo que ven nuestros ojos.
- El aprendizaje profundo es como darle a una computadora un cerebro muy complejo que aprende a partir de ejemplos. Al alimentarla con miles, o incluso millones, de imágenes, **una computadora aprende a identificar y comprender varios elementos en esas imágenes**.

Procesamiento de imágenes vs Visión artificial – conceptos cercanos pero distintos

- El objetivo principal del procesamiento de imágenes **es mejorar la calidad de la imagen**. Ya sea mejorando el contraste, ajustando los colores o suavizando los bordes, el objetivo es hacer que la imagen sea más atractiva visualmente o adecuada para su uso posterior. **Se trata de transformar la imagen original en una versión refinada de sí misma.**
- La **visión artificial**, por otro lado, busca extraer significado de las imágenes. **El objetivo no es cambiar el aspecto de la imagen, sino comprender lo que representa**. Esto implica identificar objetos, interpretar escenas e incluso reconocer patrones y comportamientos dentro de la imagen.



Cuando vemos algo, nuestros ojos y cerebro trabajan juntos en un proceso complejo para interpretar la información visual



Luz entra al ojo (cornea, pupila y retina)

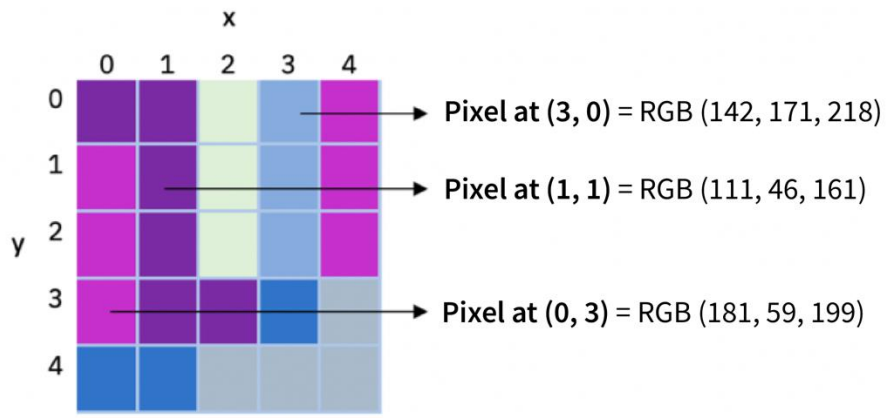
Fotoreceptores en la retina detectan luz (sensor de cámara) y la convierten en señal eléctrica

Señales son enviadas al cerebro mediante el nervio óptico y llegan a la corteza visual

Corteza visual las interpreta. Organiza la imagen, identifica formas e interpreta los colores, la profundidad y el movimiento.

Qué es una imagen y cómo verla de manera matemática

- Una imagen es una **representación bidimensional de datos visuales**, generalmente captada por una cámara o generada por una computadora.
- Puede ser en **escala de grises o en color**, dependiendo de si contiene información sobre la intensidad de la luz o la combinación de diferentes colores.
- Las imágenes se representan como **matrices multidimensionales de valores de píxeles**.
- Los valores de los píxeles varían de **0 a 255**, donde 0 representa el negro y 255 representa el blanco.
- En las imágenes en color, cada píxel está representado por una combinación de valores de los tres colores primarios: **rojo, verde y azul (RGB)**. Los valores de píxel de cada canal de color varían de 0 a 255, por lo que un solo píxel de una imagen RGB está representado por un vector tridimensional de valores (R, G, B).

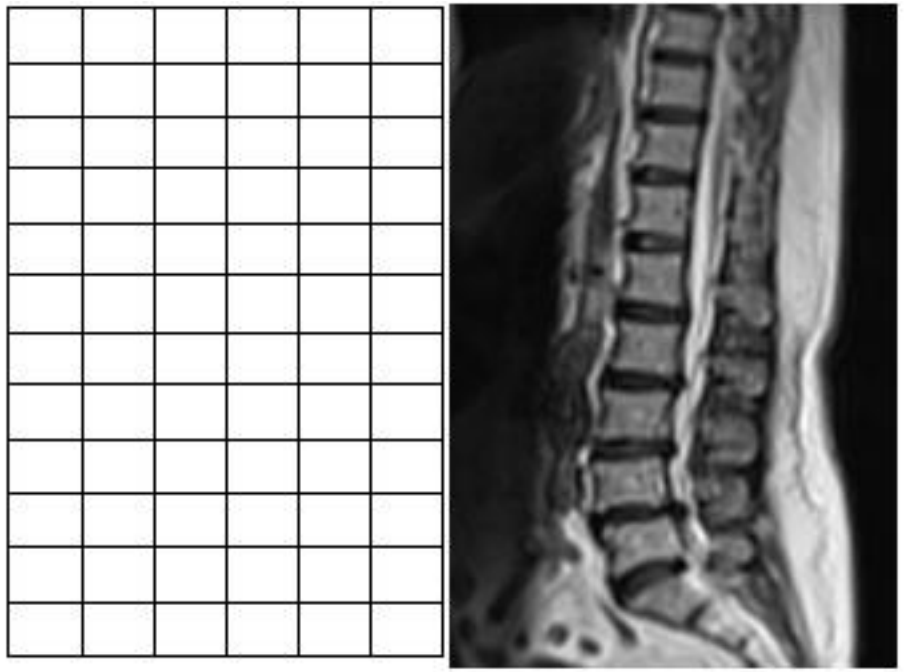


Fuente: https://www.scaler.com/topics/computer-vision-image/

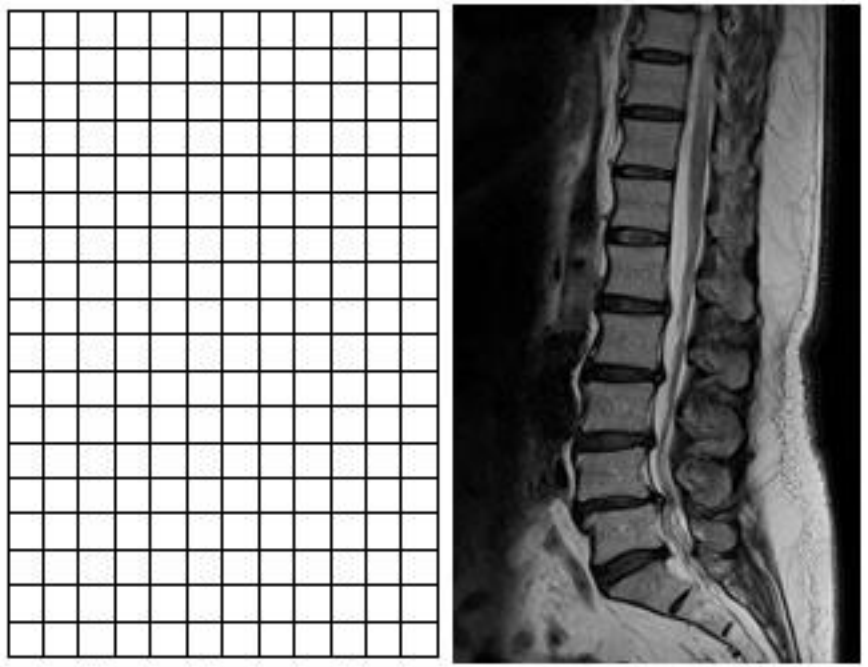
¿Cómo se forman las imágenes?

- Las imágenes tomadas con una cámara se forman **captando la luz reflejada o emitida por los objetos del mundo que nos rodea.**
- Cuando la luz incide en un objeto, se refleja y es captada por una **cámara, que registra la intensidad y la información del color de la luz.** El sensor de la cámara convierte la luz en datos digitales, que se almacenan como un archivo de imagen.
- La **resolución** de la imagen está determinada por la cantidad de píxeles que la componen, y **cada píxel representa una unidad minúscula de la imagen.**
- Cuanto mayor sea la cantidad de píxeles, mayor será el nivel de detalle y la resolución de la imagen.

Ejemplo de resolución: A mayor número de pixeles, mejor resolución

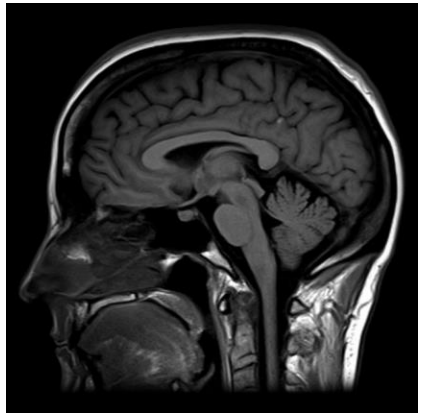


Large pixel size low resolution image



Small pixel size high resolution image

Imágenes en escalas grises solo tienen un canal



Dimensión 1

Dimensión 0

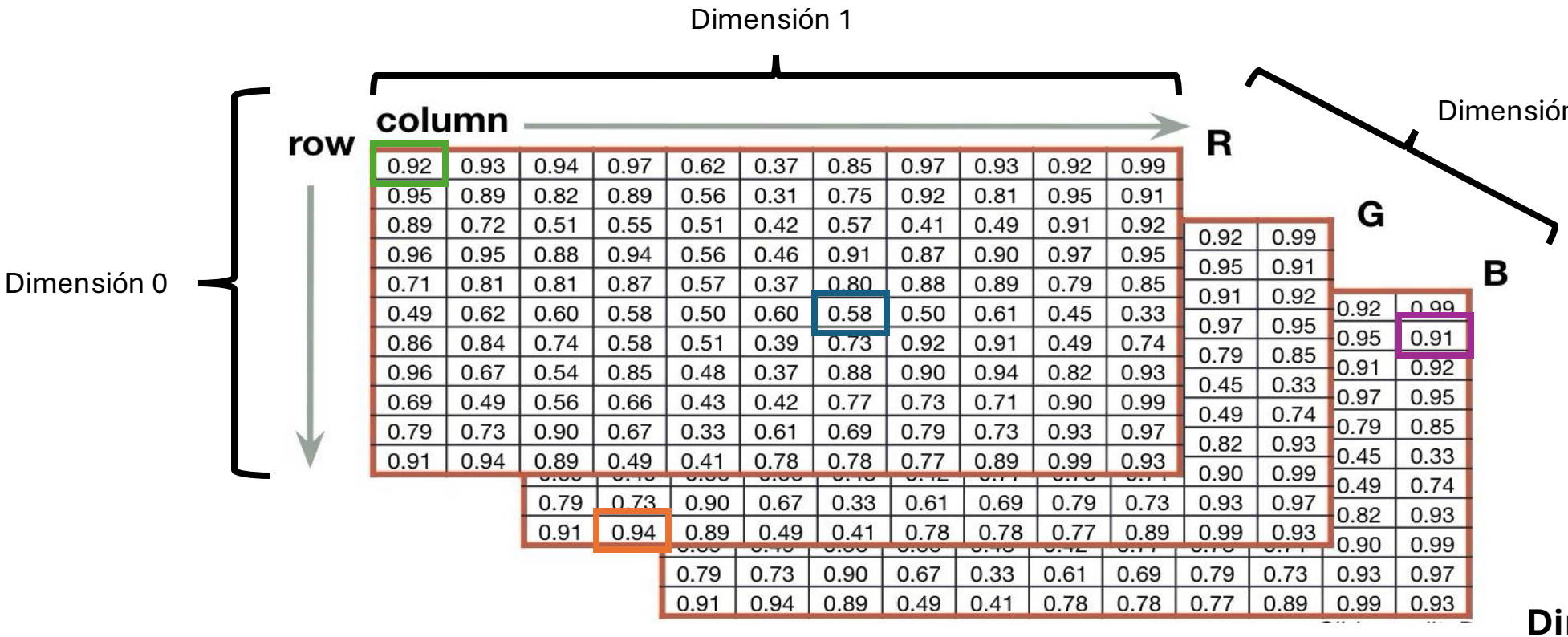
row \ column										
0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

Dimensiones: 11x11x1

Posición (0,0)

Posición (5,6)

Ejemplo de estructura en forma de matriz (imagen a color)



Dimensiones: 11x11x3

Posición (0,0,0)

Posición (1,10,2)

Posición (10,1,1)

Posición ?

Los valores de los pixeles puede ser normalizados a una escala de 0 a 1

Librerías en python y formato de las imagenes

- En python hay varias librerías que trabajan con imagenes:

1. **Pillow:** Manipulación básica de imágenes (cambio de tamaño, recorte, conversión de formato), adición de texto o formas y filtrado simple.
2. **Scikit-mage:** Operaciones básicas además de segmentación de imágenes, filtrado, operaciones morfológicas, detección de características y conversiones de espacios de color.
3. **OpenCV:** Preprocesamiento de imágenes, detección de rostros, seguimiento de objetos, transformación de imágenes, extracción de características.
4. **SimpleITK:** Utilizada mucho para análisis de imágenes médicas (TC, RM, rayos X), segmentación de imágenes, registro y transformaciones 3D.

Algunas funciones útiles de las librerías más importantes

	OpenCV	Skimage	PIL	TensorFlow	Pytorch
Leer imagen	cv2.imread()	skimage.io.imread()	Image.open()	tf.keras.utils.load_img()	torchvision.io.read_image()
Tipo de objeto	numpy.ndarray	numpy.ndarray	PIL.JpegImagePlugin.JpegImageFile	PIL.JpegImagePlugin.JpegImageFile	Tensors
Como obtener dimensiones	.shape (height, width, channels)	.shape (height, width, channels)	.size (Width, height)	.size (Width, height)	Torch.size() (Channels, Height, width)
Como mostrar imagen	Cv2.imshow()	skimage.io.imshow()	Object.show()	PIL.Image.open()	torchvision.transforms. ToPILImage()(img) .show()
Convertir en matriz de numpy			np.array(Image. Image.getdata())	tf.keras.utils.img_to_array()	
Guardar imagen	Cv2.imwrite()	skimage.io.imsave()	1. Object.save() 2. Image.fromarray()	tf.keras.utils.save_img()	torchvision.transforms. ToPILImage() .save()

https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2F%40vinod.batra0311%2Fnuances-and-caveats-of-various-image-processing-tools-for-computer-vision-and-neural-networks-251da91d07&psig=AOvVaw337A_EkxqzMYXjy8KXqJaV&ust=1726559774447000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCNCPl8P-xogDFQAAAAAdAAAAABAE

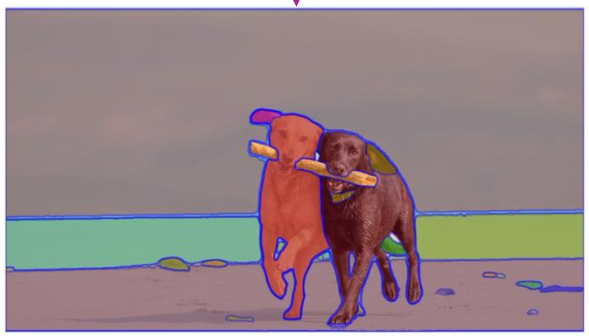
Casos de uso

Ejemplos de análisis de imágenes

- En el área de salud
- En la industrial de automóviles
- Sistemas de seguridad

¿Cuál es uno que utilizamos a diario?

Ejemplos de problemas en que se utiliza



Clasificación
Segmentación de imagenes (semántica y de instancia)
Detección de objetos
Generación de imagenes (GenAI)
Restauración: eliminación de ruido, restauración de color, superresolución
Registro de imagenes (alineamiento)
Subtítulo de imagenes (descripción textual)
Reconstrucción de objetos en 3D

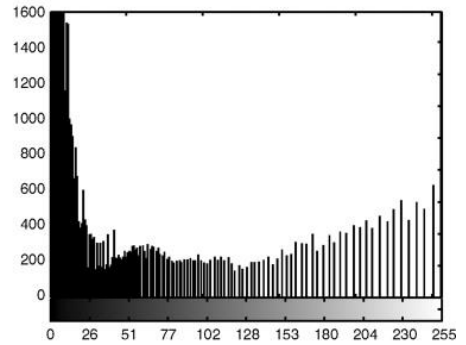
ALGUNAS TECNICAS CLASICAS PARA ANALIZAR IMAGENES

- Histogramas
- Restauración: Mejora de resolución, manejo de ruidos, etc
- Segmentación y bordes

1 Ecualización de histograma

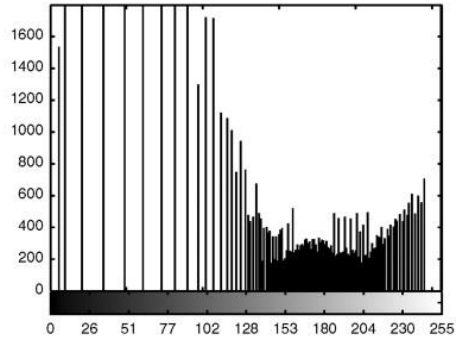
- Es un método de procesamiento de imágenes para **ajustar el contraste** utilizando el **histograma de la imagen**
- Este método suele aumentar el contraste global de muchas imágenes, **especialmente cuando la imagen está representada por un rango estrecho de valores de intensidad.**
- Mediante este ajuste, las intensidades se pueden distribuir mejor en el histograma utilizando el rango completo de intensidades de manera uniforme.
- Esto permite que las áreas de menor contraste local obtengan un mayor contraste. La ecualización del histograma logra esto **al distribuir de manera efectiva los valores de intensidad altamente poblados que se utilizan para degradar el contraste de la imagen.**
- **El método es útil en imágenes con fondos y primeros planos que son a la vez brillantes u oscuros.** En particular, el método puede dar lugar a mejores vistas de la estructura ósea en imágenes de rayos X y a mejores detalles en fotografías que están sobreexpuestas o subexpuestas.

(A) Original Image



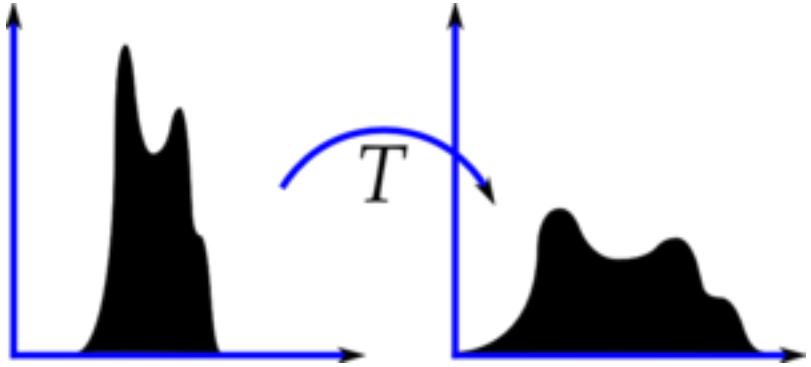
(B) Histogram of the Original Image

(C) Image with Equalized Histogram



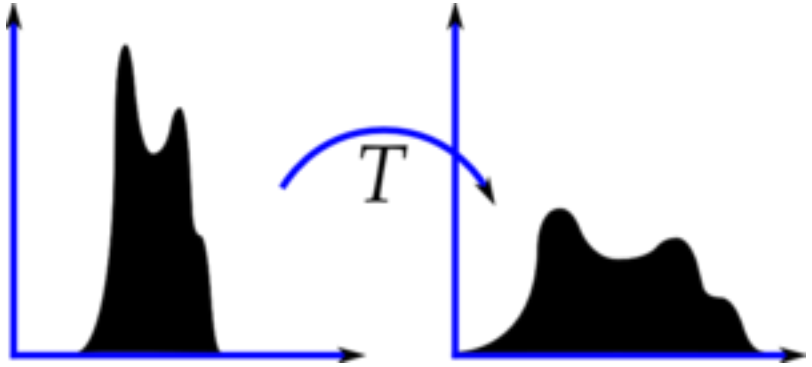
(D) Histogram of the Equalized Image

1 Ecualización de histograma



- Pasos:
  1. **Cálculo del histograma** (una barra para cada valor de intensidad 0-255)
  2. **Cálculo de PDF** (Probability Distribution Function) – obtener p(i) probabilidad de intensidad i $p_i = \frac{n_i}{N}$ donde n_i representa el número de pixeles de esa intensidad y N el total de pixeles
  3. **Cálculo de CDF** (Función de distribución acumulada) = $\sum_{j=0}^{i} p(j)$
  4. **Normalización del CDF**: Mapeo de los valores a un nuevo rango de intensidad $T(i) = \big(CDF(i)\big) * (L-1)$ donde T_i es el nuevo valor de intensidad para los pixeles de intensidad original i. L es 256 si los datos están en escala 0-255 y CDF(min) es el valor mínimo distinto de cero del CDF (para evitar asignar todos los píxeles a cero).

1 Ecualización de histograma



- Pasos:
  1. Cálculo del histograma (una barra para cada valor de intensidad 0-255)
  2. Cálculo de PDF (Probability Distribution Function) – obtener p(i) probabilidad de intensidad i $p_i = \frac{n_i}{N}$ donde n_i representa el número de pixeles de esa intensidad y N el total de pixeles
  3. Cálculo de CDF (Función de distribución acumulada) = $\sum_{j=0}^{i} p(j)$
  4. Normalización del CDF: Mapeo de los valores a un nuevo rango de intensidad $T(i) = \big(CDF(i)\big) * (L-1)$ donde T_i es el nuevo valor de intensidad para los pixeles de intensidad original i. L es número de posibles valores de pixeles 256 si los datos están en escala 0-255 y CDF(min) es el valor mínimo distinto de cero del CDF (para evitar asignar todos los píxeles a cero).

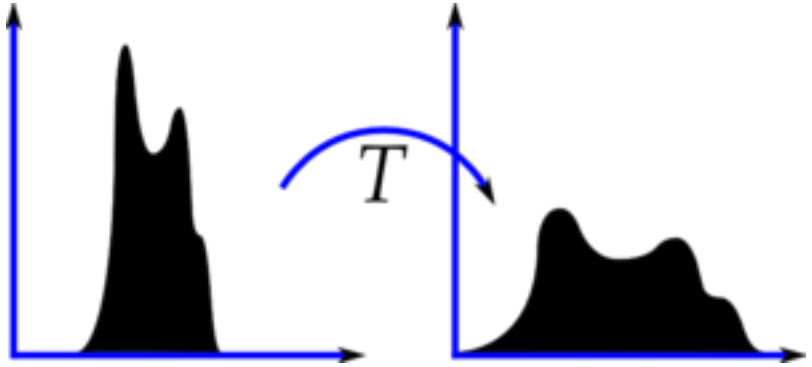
Ejemplo

IMAGEN

0	**0**	**1**	**1**
3	**4**	**3**	**3**
4	**2**	**0**	**0**

Intensidad	**0**	**1**	**2**	**3**	**4**	**Total**
Número de pixeles	4	2	1	3	2	12
p(i)	4/12	2/12	1/12	3/12	2/12	
CDF(i)	4/12	6/12	7/12	10/12	12/12	
T(i)						

1 Ecualización de histograma



- Pasos:
  1. Cálculo del histograma (una barra para cada valor de intensidad 0-255)
  2. Cálculo de PDF (Probability Distribution Function) – obtener p(i) probabilidad de intensidad i $p_i = \frac{n_i}{N}$ donde n_i representa el número de pixeles de esa intensidad y N el total de pixeles
  3. Cálculo de CDF (Función de distribución acumulada) = $\sum_{j=0}^{i} p(j)$
  4. Normalización del CDF: Mapeo de los valores a un nuevo rango de intensidad $T(i) = \left(CDF(i)\right) * (L-1)$ donde T_i es el nuevo valor de intensidad para los pixeles de intensidad original i. L es número de posibles valores de pixeles 256 si los datos están en escala 0-255 y CDF(min) es el valor mínimo distinto de cero del CDF (para evitar asignar todos los píxeles a cero).

Ejemplo

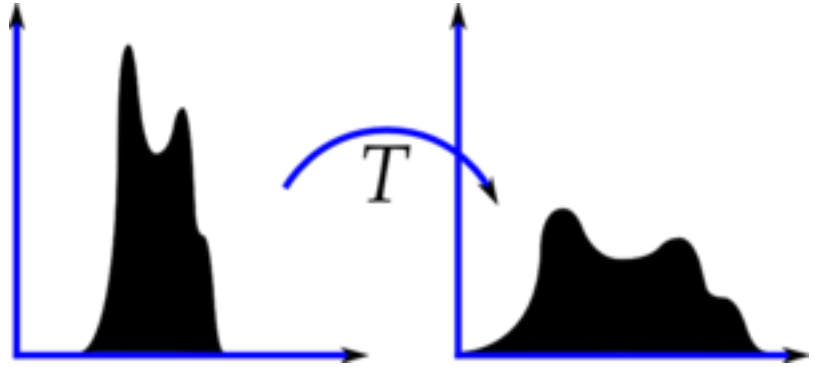
IMAGEN

0	**0**	**1**	**1**
3	**4**	**3**	**3**
4	**2**	**0**	**0**

Intensidad	0	1	2	3	4	Total
Número de pixeles	4	2	1	3	2	12
p(i)	4/12	2/12	1/12	3/12	2/12	
CDF(i)	4/12	6/12	7/12	10/12	12/12	
T(i)	(4/12) *4 = 1.33 = **1**	(6/12)*4 = **2**	(7/12) *4 = 2.33 = **2**	(10/12) *4 = 3.33 = **3**	(12/12) *4 = **4**	

CDF(min) = 4/12
L = 5
N = 12
(L-1) = 4

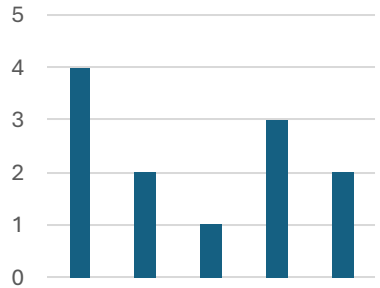
1 Ecualización de histograma



Ejemplo

IMAGEN

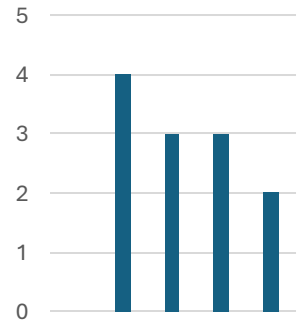
0	**0**	**1**	**1**
3	**4**	**3**	**3**
4	**2**	**0**	**0**



Intensidad	0	1	2	3	4	Total
Número de pixeles	4	2	1	3	2	12
p(i)	4/12	2/12	1/12	3/12	2/12	
CDF(i)	4/12	6/12	7/12	10/12	12/12	
T(i)	(4/12) *4 = 1.33 = **1**	(6/12)*4 = **2**	(7/12) *4 = 2.33 = **2**	(10/12) *4 = 3.33 = **3**	(12/12) *4 = **4**	

NUEVA IMAGEN

1	**1**	**2**	**2**
3	**4**	**3**	**3**
4	**2**	**1**	**1**



2 Filtros: Media y desenfoque gausiano

Uso de filtros para cálcular valores con respect al vecindario

A. **Filtro de la media:** Método simple e intuitivo para la **reducción de ruido**. Funciona **promediando los valores de los píxeles en un área local** alrededor de cada píxel. (convolución con un kernel uniforme)

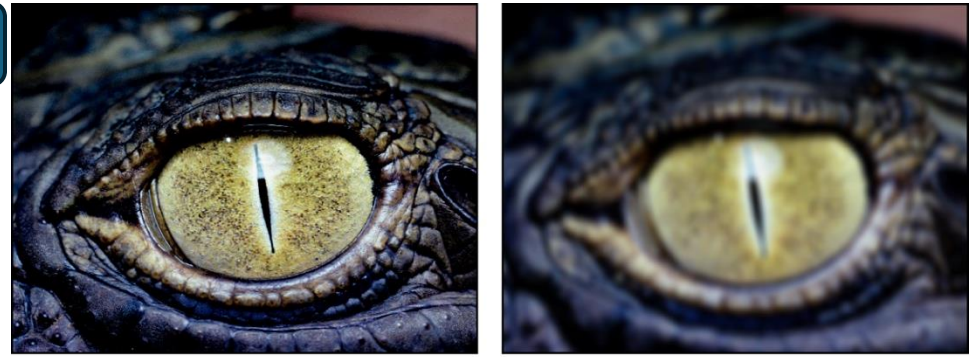
B. **Desenfoque gaussiano:** Se utiliza **para suavizar las imágenes** y reducir el ruido (convolución de la imagen con un kernel gaussiano)

A

Noisy image

Filtered by Mean Filter

B

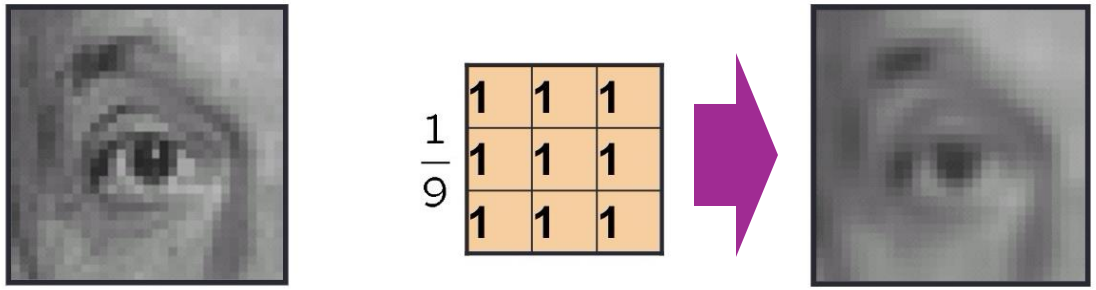


Original image

Gaussian Blur filter applied

2 Filtros: Media y desenfoque gausiano

Ejemplo de filtro de media de tamaño 3x3

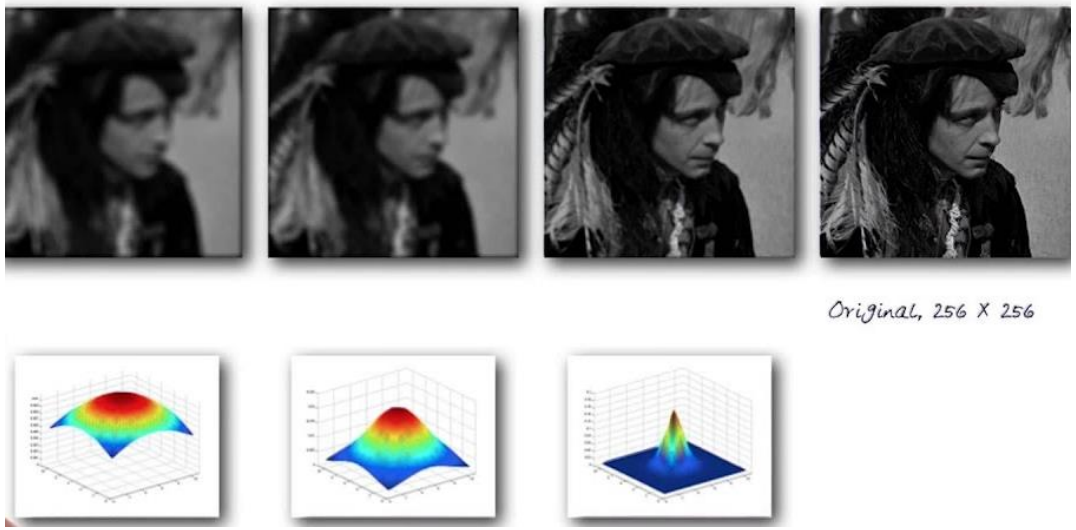


Pasos clave en el cálculo:

1. Aplicar la ventana 3x3 (o deseada) a lo largo de la imagen.
2. Calcular la media de los valores en la ventana.
3. Utilizar el valor de la media en “nueva imagen” por esa posición (en clases siguientes veremos cómo se hace para que quede del mismo tamaño)

2 Filtros: Media y desenfoque gausiano

Ejemplo de filtro gausiano con distintos sigmas



Ejemplo de filtro 3x3 gausiano continuo y discreto

0.075	0.124	0.075
0.124	0.204	0.124
0.075	0.124	0.075

1/16

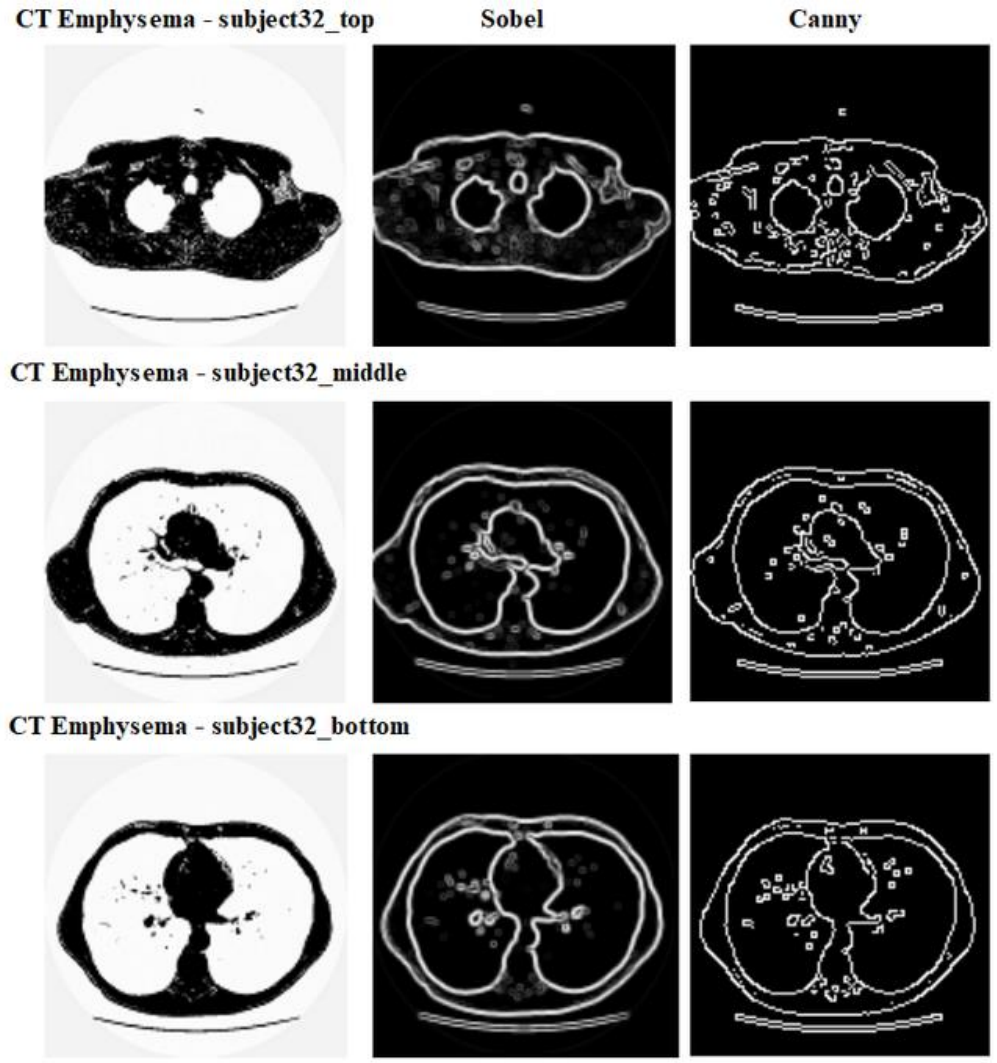
1	2	1
2	4	2
1	2	1

Pasos clave en el cálculo:

1. Decidir tamaño del filtro – cuánto del vecindario considera.
2. Decidir sigma: atenuación o desenfoque que se quiere (entre más alto el sigma, mayor desenfoque
3. Aplicar el filtro y obtener nueva imagen

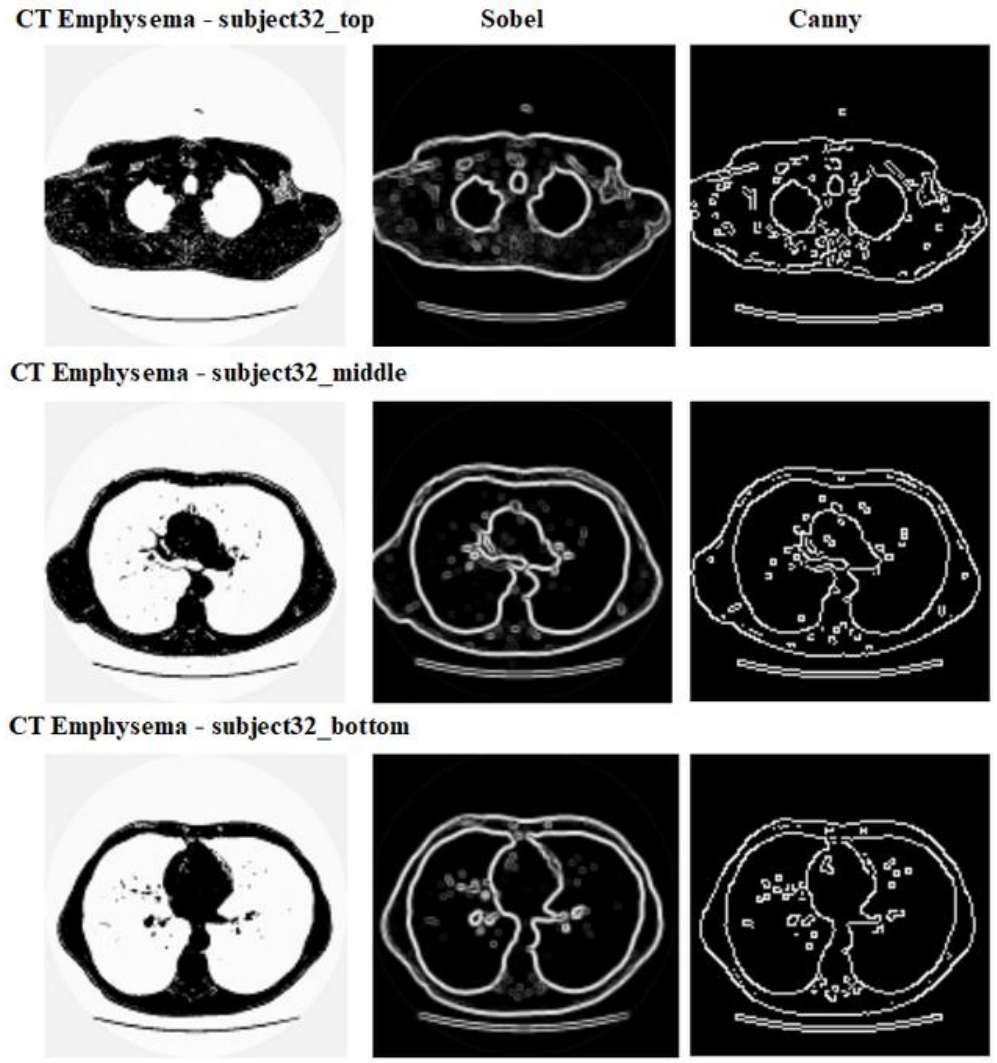
3 Detección de bordes: Sobel y canny

- El detector de bordes **Sobel** es un método sencillo y ampliamente utilizado para detectar bordes en una imagen. Funciona **calculando el gradiente de intensidad** de la imagen en cada píxel, lo que **resalta las regiones de alta frecuencia** espacial que corresponden a los bordes. Los puede calcular de manera vertical, horizontal y la combinación
- Es simple de entender y fácil de calcular
- Puede ser afectado por ruido en la imagen y detector incorrectamente bordes.



3 Detección de bordes: Sobel y canny

- El detector de bordes **Canny** es un método de detección de bordes más avanzado que tiene como objetivo encontrar los bordes más significativos en una imagen.
- Además del cálculo de gradiene, **incluye otros pasos para mejorar la precisión** de la detección de bordes y reducir el ruido como uso **de puntos de corte** para dejar los bordes más relevantes
- Captura menos ruido que Sobel
- Es más complejo y lento de calcular



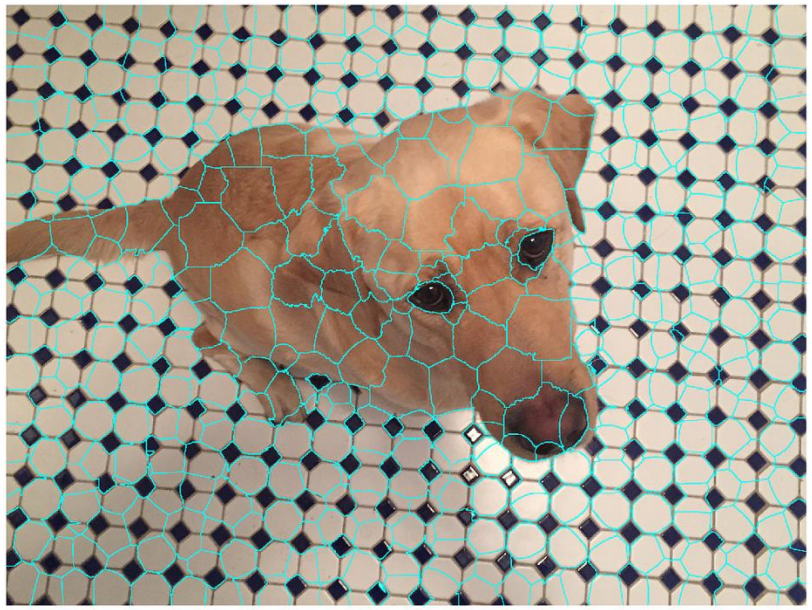
4 Segmentación: k-medias y SLIC

- La agrupación en **k-medias** es una técnica que divide una imagen en k grupos según los **valores de intensidad de los pixels**.
- Pasos:
  - **Inicialización:** Se elijen k centroides iniciales de clúster de forma aleatoria.
  - **Asignación:** Se asigna cada píxel al centroide más cercano, en función de una métrica de distancia.
  - **Actualización:** calcule los nuevos centroides como la media de todos los píxeles asignados a cada clúster.
  - **Iteración:** Se repiten los pasos de asignación y actualización hasta la convergencia (cuando las asignaciones de clúster ya no cambien).

Figure 1: (a) is the original image; (b) and (c) are the segmentation results.

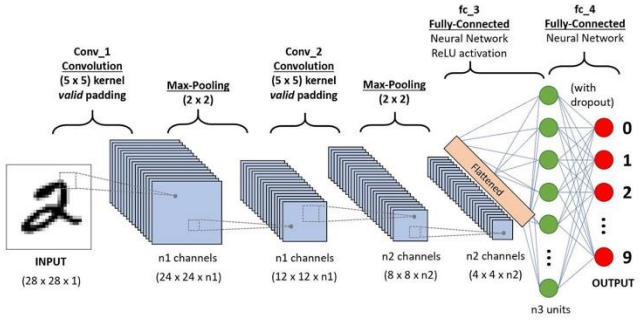
4 Segmentación: k-medias y SLIC

- SLIC (Simple Linear Iterative Clustering) es un método de segmentación de superpíxeles que **divide una imagen en grupos de píxeles, o "superpíxeles",** que son **coherentes espacialmente**.
- Está diseñado para producir segmentos que **respeten los límites de la imagen** y que sean más uniformes y significativos en comparación con los métodos de agrupamiento simples.
- Pasos:
  - **Inicialización:** Se define una cuadrícula de puntos de semilla iniciales en toda la imagen.
  - **Asignación de grupos:** cada píxel se asígna al punto de semilla más cercano en función de la similitud de color y la distancia espacial.
  - **Actualización:** se vuelve a calcular los puntos de semilla como el centro de los grupos formados en el paso anterior.
  - **Iteración:** repite los pasos de asignación y actualización hasta la convergencia.



Redes neuronales aplicadas a imagenes

Cómo aplicamos redes neuronales para solucionar algunos de los casos de uso vistos?



- **Pueden ser utilizadas para la mayoría** de los que hemos comentado clasificación, detección, segmentación, restauración, etc
- Con las redes neuronales MLP que hemos visto en laboratorios, podemos utilizar **todos los pixeles como variables de entrada** del modelo.
  - Cambiamos de **2D a 1D los datos**: "aplanar la imagen" (ejemplo una imagen de dimensiones 3x3 y escalas grises podemos convertirla a un vector de 9 valores, si en cambio es una imagen de 9x9 con 3 canales de colores entonces son 9x9x3 = 243 valores en un solo vector)
- Uno de los problemas de esto es que la **dimensionalidad crece mucho** y se vuelve computacionalmente complejo para imágenes grandes
- Además, al pasarlo a un vector, perdemos el concepto de **vecindario y correlación y estructura espacial**
- Solución? ***Redes Neuronales Convolucionales (CNN)***

Preguntas?