



Aprendizaje Automático Profundo

Clase 6 - 2019

Profs: Franco Ronchetti - Facundo Quiroga



IMÁGENES DIGITALES

Imagen digital

Una imagen digital es una representación estructurada y discretizada que modela a una imagen analógica, con posibilidad de ser almacenada y procesada en un sistema informático.

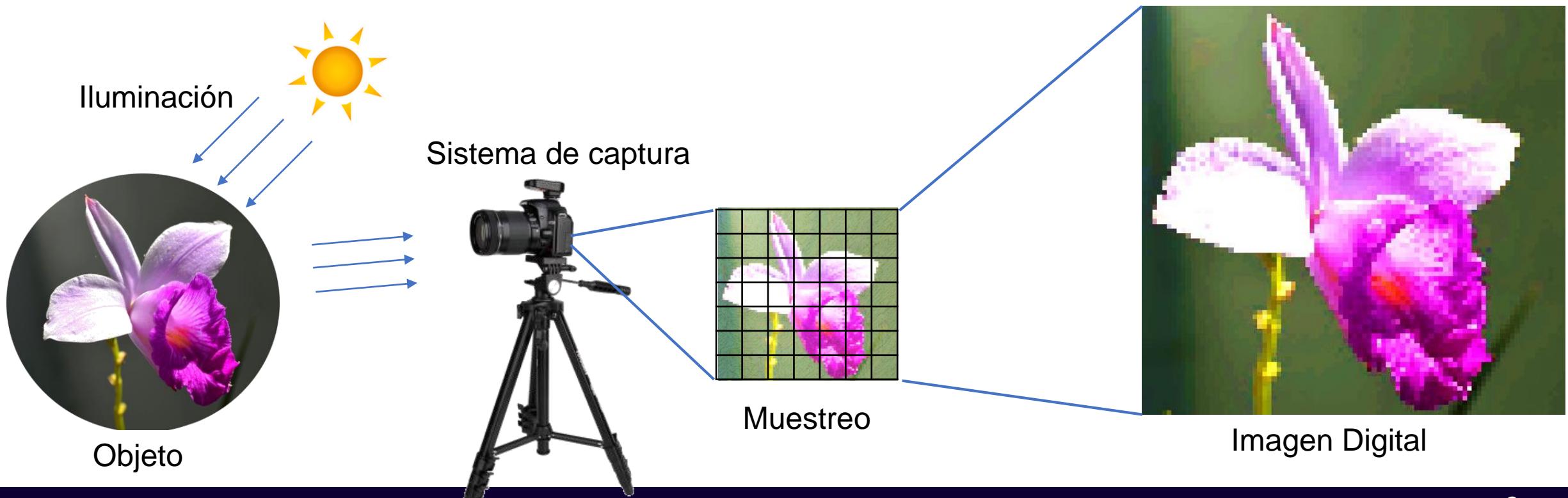


Imagen digital

- La estructura de una imagen es siempre una matriz de $N \times M$ píxeles.
- El pixel, entonces, es la unidad de información más pequeña en una imagen digital
- Cada pixel representa un punto en la imagen, de un color particular.

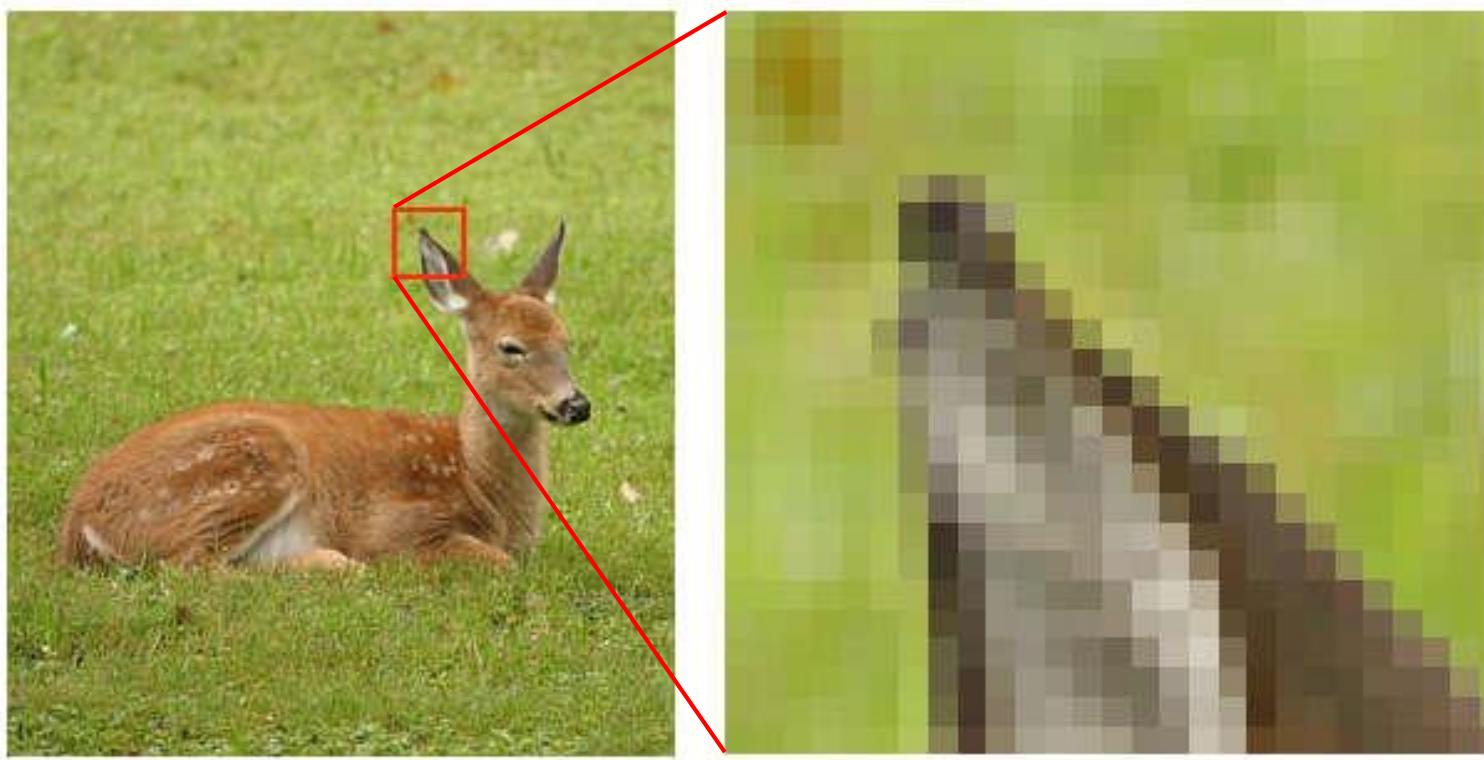
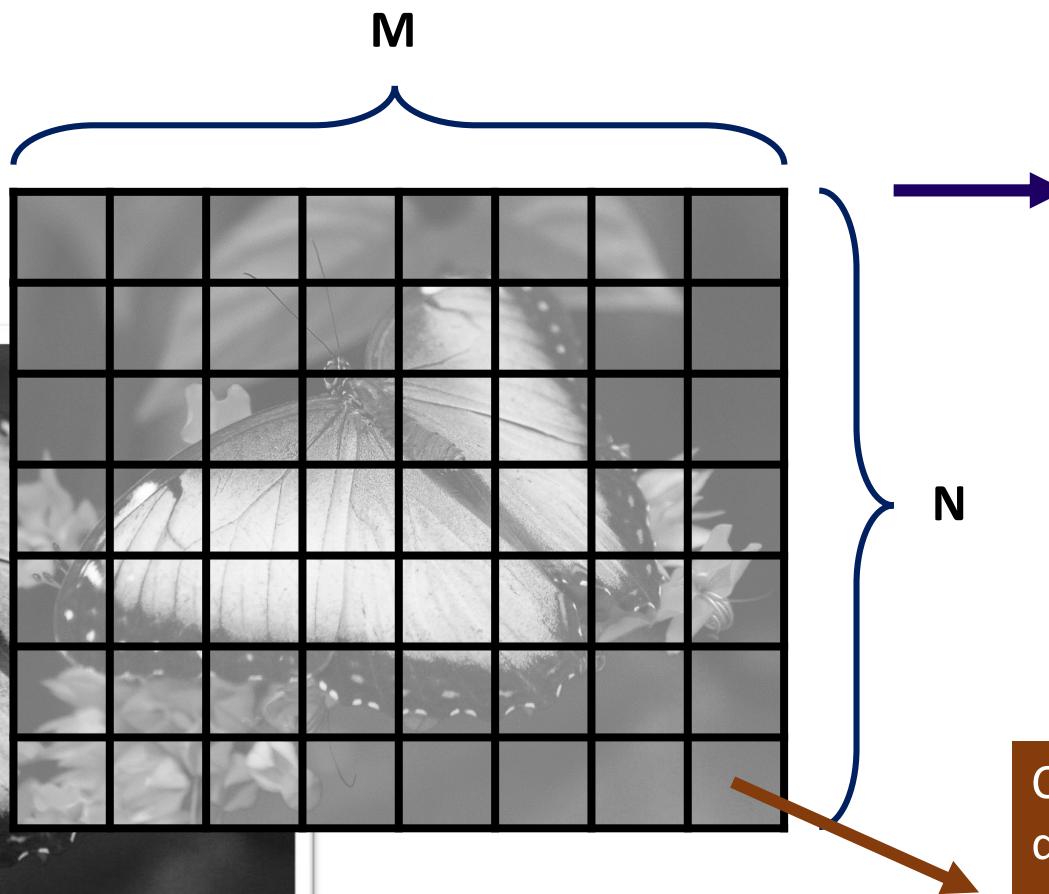
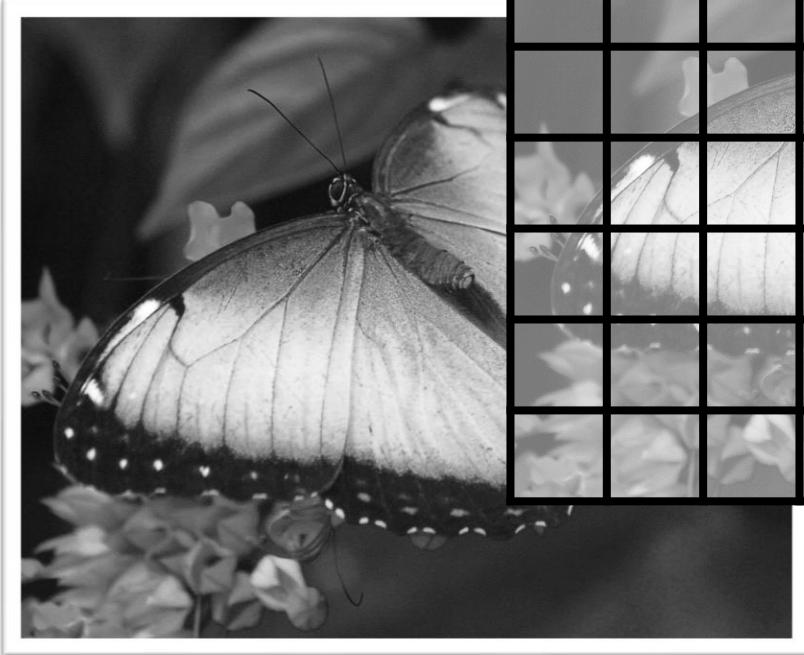


Imagen digital - Representación interna



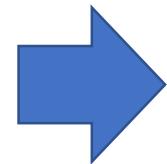
$N \times M$ es la cantidad de píxeles, o tamaño, de la imagen. A este tamaño normalmente se lo conoce como **Resolución** de la imagen.

A mayor resolución, mejor “calidad” de la imagen.
Por ejemplo una imagen de 2.000×2.000 píxeles, tendrá 4 millones de píxeles, lo que equivale a decir 4 Mp.

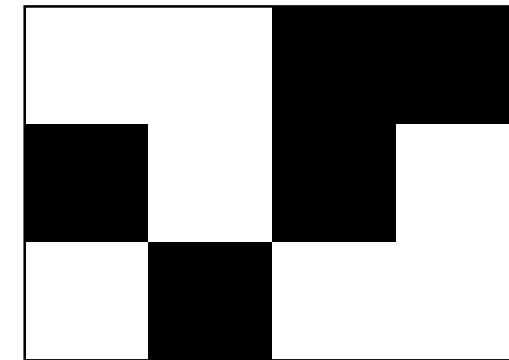
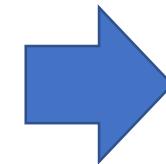
Cada pixel representa la intensidad de luz en ese punto y tiene que ser representado discretamente en una computadora.
¿Qué rango de valores utilizar?

Imagen digital - B&N

1	1	0	0
0	1	0	1
1	0	1	1



Si a cada pixel le damos un valor binario (0/1), podríamos tener una imagen con dos colores: Blanco & Negro



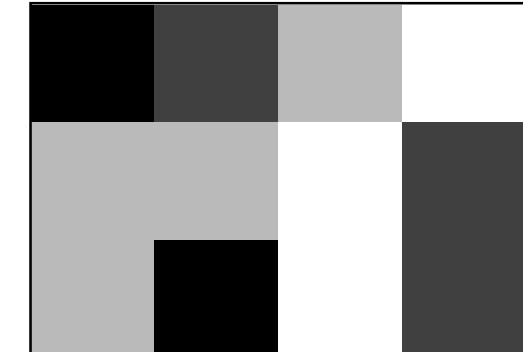
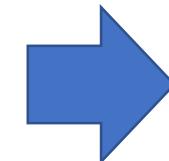
Nuestro Monitor interpreta los “0” como color negro y los “1” como color blanco

Imagen digital – Escala de grises

0	1	2	3
2	2	3	1
2	0	3	1



Ahora bien, si a cada pixel le damos la posibilidad de tener 4 valores distintos, podríamos tener 4 colores.



0	
1	
2	
3	

Interpretación de la imagen (paleta de colores).

Imagen digital – Escala de grises

Profundidad de color: Entonces, a mayor rango de valores posibles, mayor cantidad de colores. Esto se conoce como “profundidad de color”. Generalmente, la profundidad se mide en “bits”.

$$1 \text{ bit} = 2 \text{ colores}$$

$$2 \text{ bits} = 4 \text{ colores}$$

$$8 \text{ bits} = 256 \text{ colores}$$

0	1	2	255
128	2	180	1
30	40	70	1

8 bits

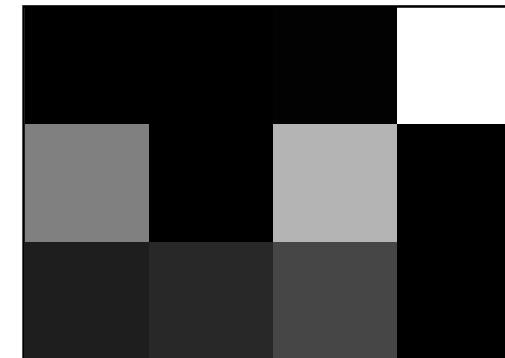
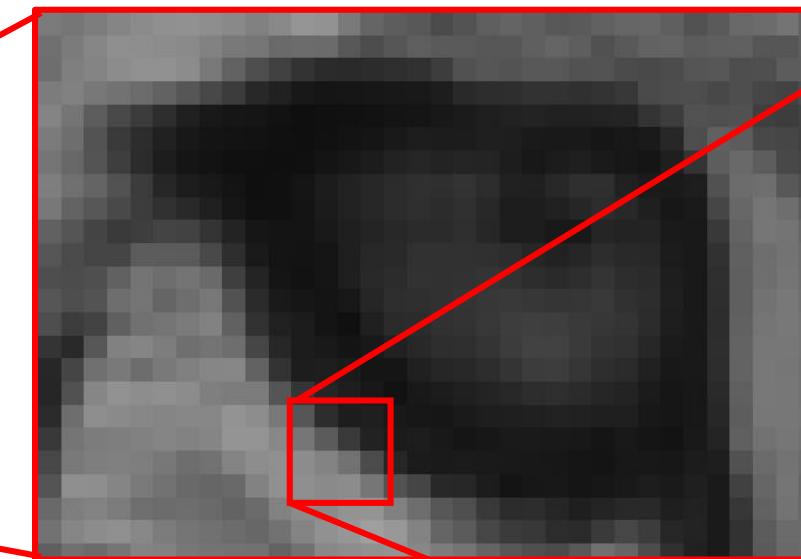
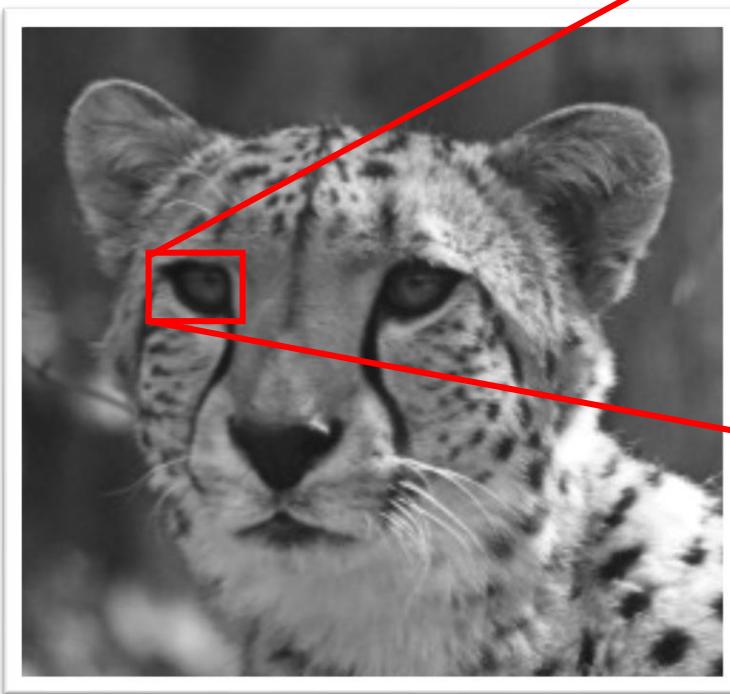


Imagen digital – Escala de grises

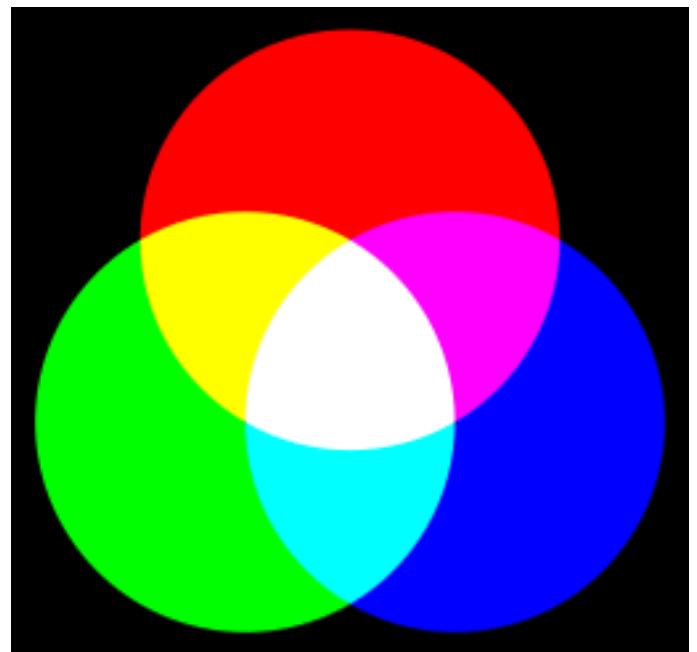


96	55	42	30
120	110	50	32
175	132	115	105
180	182	152	122

Pueden ser **int (0-255)** o **float (0-1)**

Imagen color

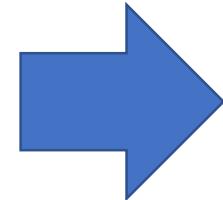
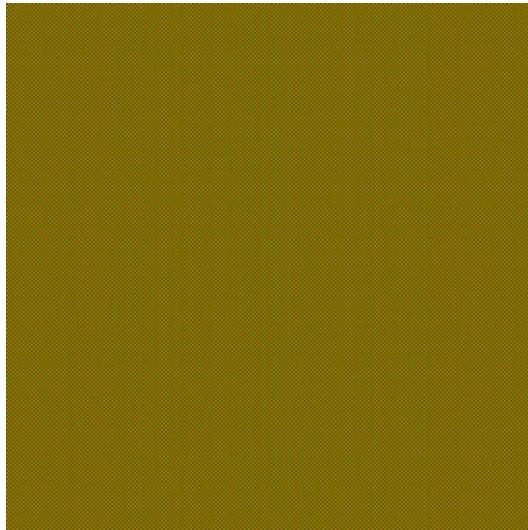
- Para representar imágenes a color, se utiliza un modelo de percepción humana. Es un sistema aditivo de color.
- Este mismo modelo utilizan todas las pantallas (monitores, celulares, etc.).
- Este modelo se basa en tres componentes de color: Rojo, Verde y Azul (*Red, Green, Blue - RGB*).
- Con estos tres colores primarios es posible formar cualquier otro color, visible por un ser humano.



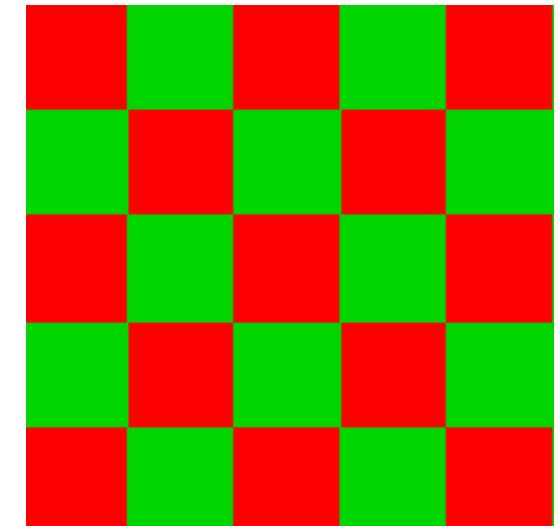
Modelo RGB

Imagen color

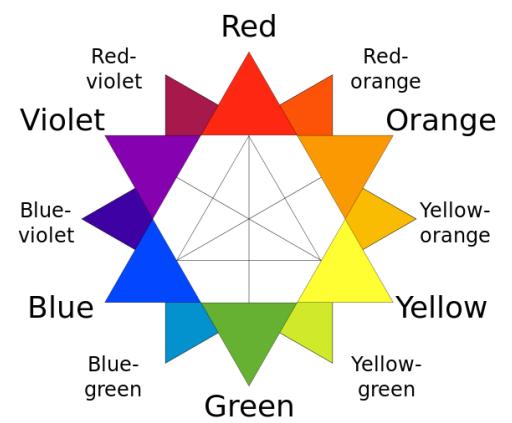
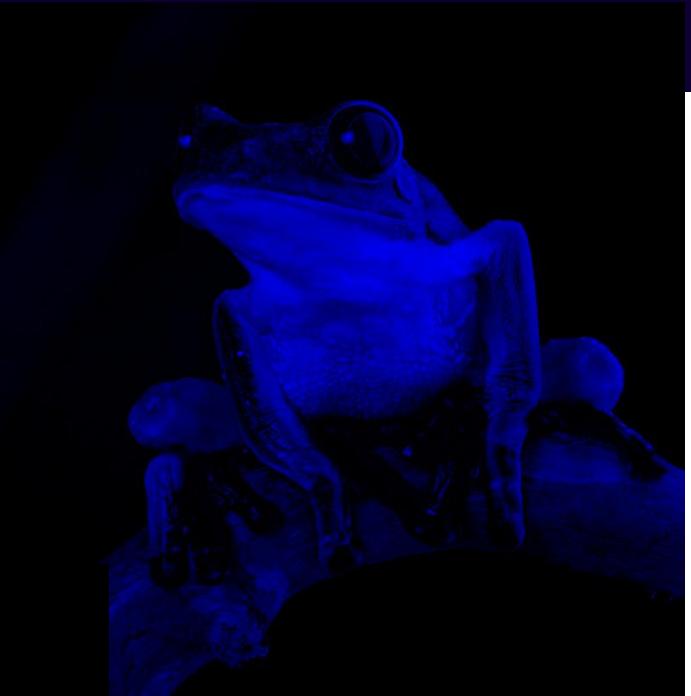
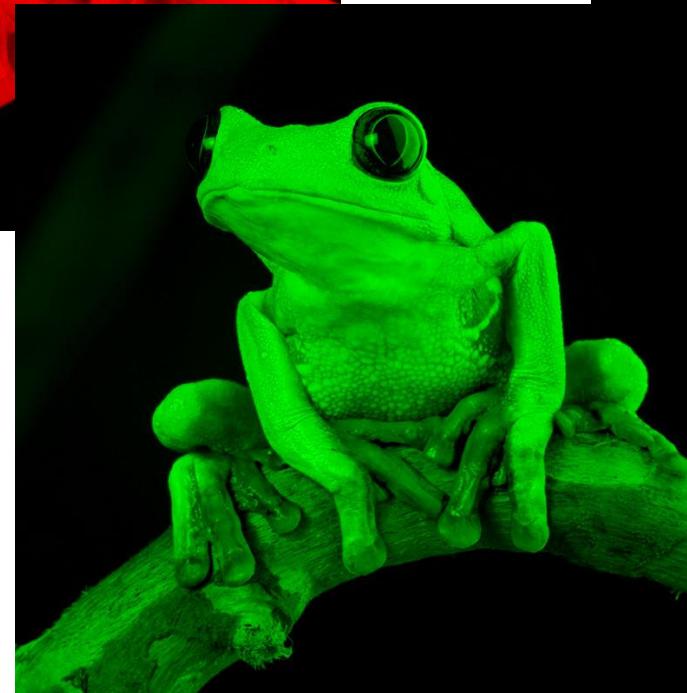
¿De qué color es esta imagen?



El efecto se produce por la percepción del sistema de visión humana



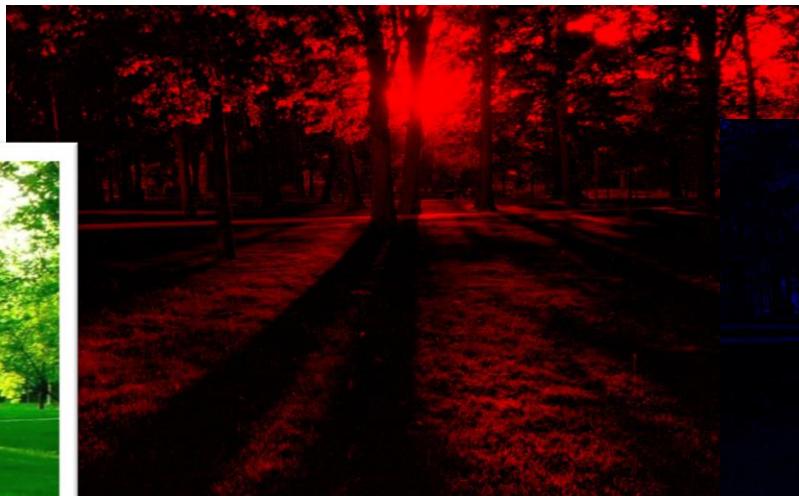
Modelo RGB



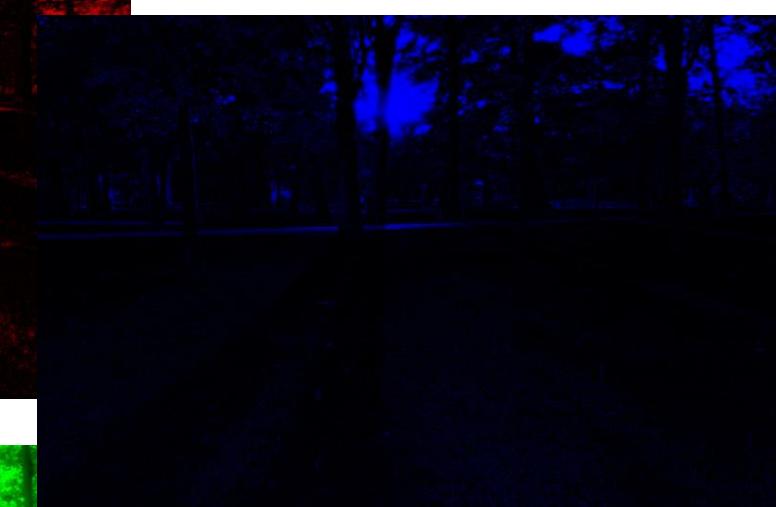
Modelo RGB



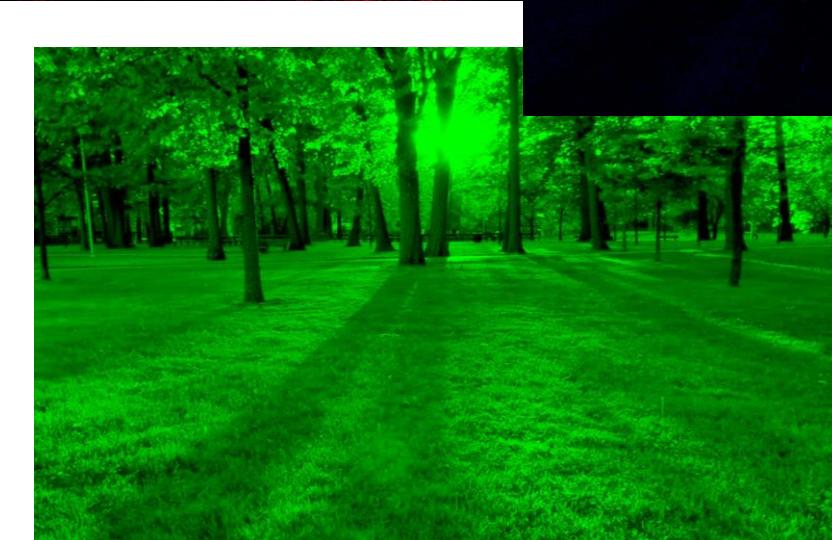
Red



Blue



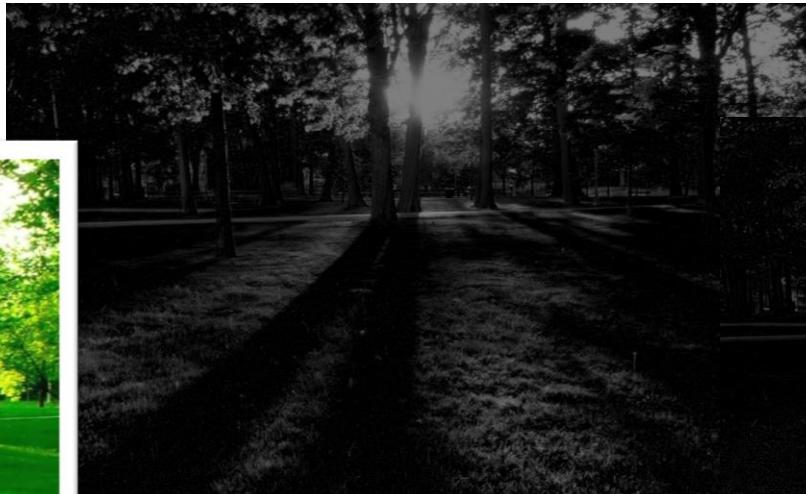
Green



Modelo RGB



Red



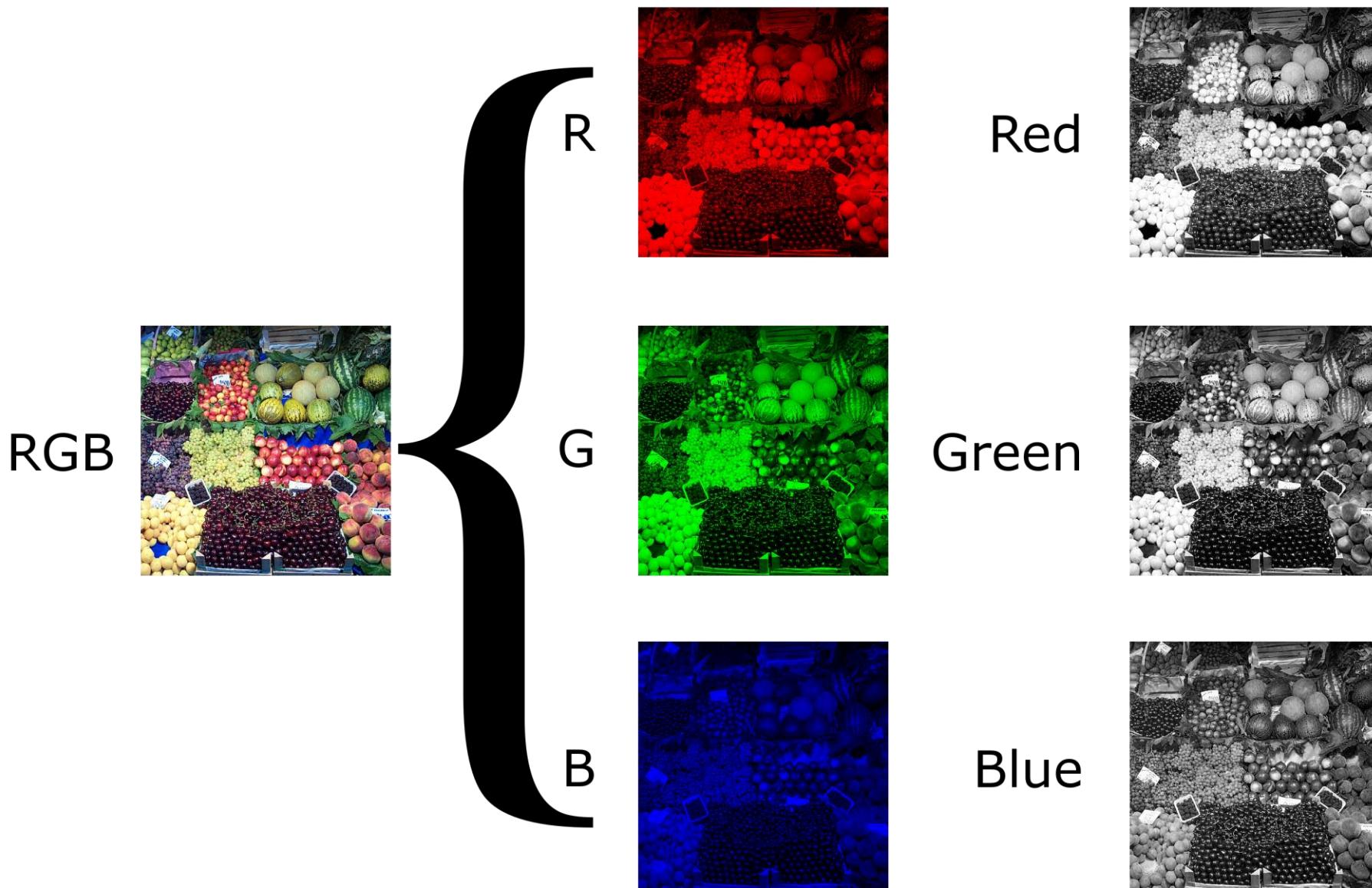
Blue



Green



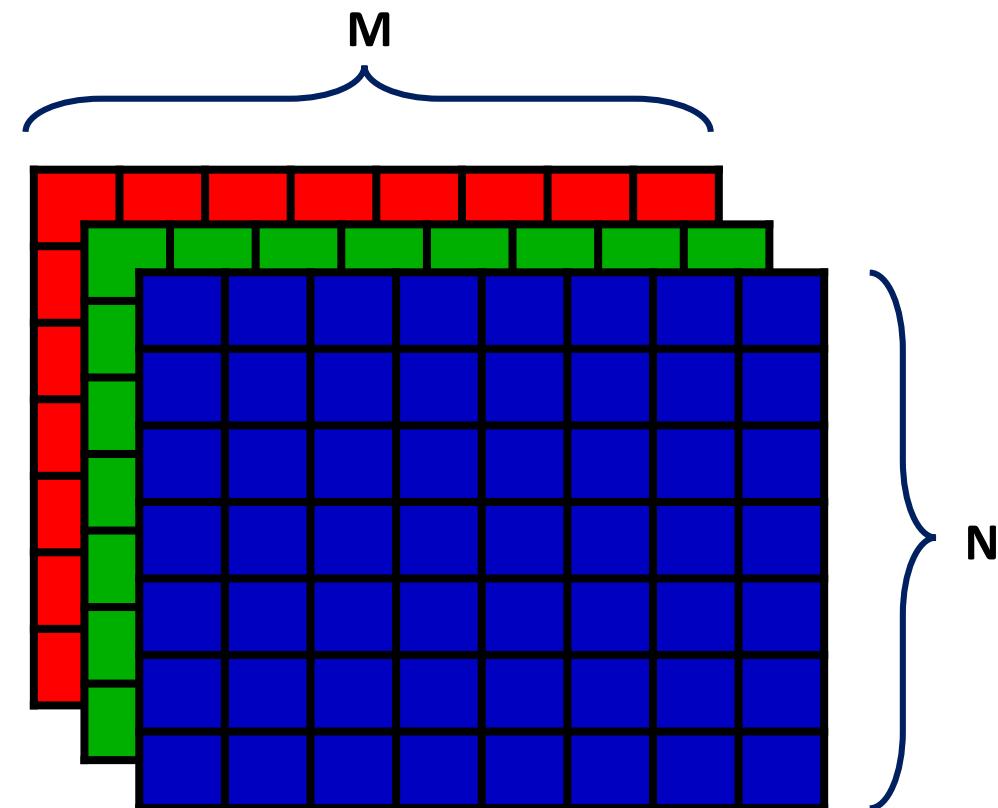
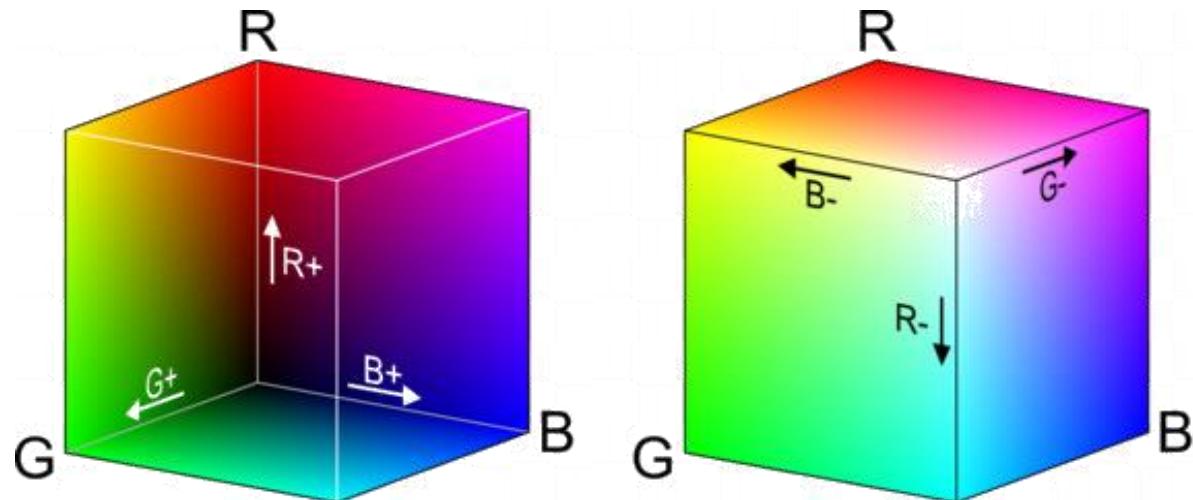
Modelo RGB



Modelo RGB

Entonces, para representar digitalmente una imagen color, se necesitan 3 matrices de $N \times M$. Una para cada canal.

A mayor intensidad de luz en los 3 canales, los colores se aclaran. A menor intensidad, se oscurecen.



Profundidad de color

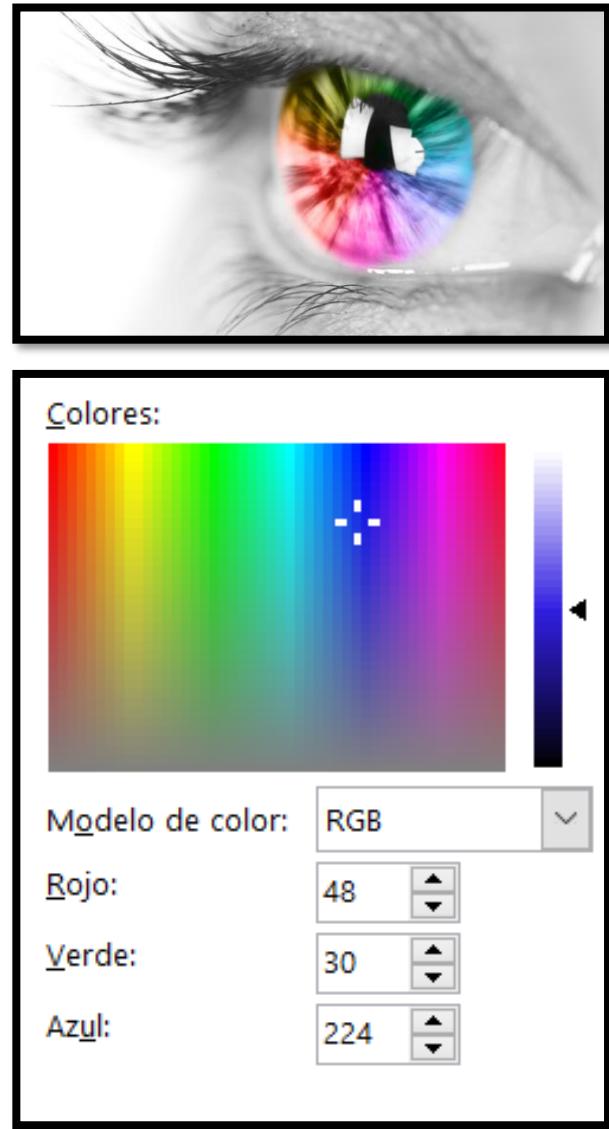
- ¿Cuántos bits son necesarios para modelar el color?
- En un principio se utilizaban modelos de pocos bits. Por ejemplo: **8 bits**.
- Con 8 bits, tenemos: $2^8 = \mathbf{256}$ colores

Generalmente, las diferentes computadoras/consolas usaban paletas de colores para optimizar las pocas combinaciones que tenían



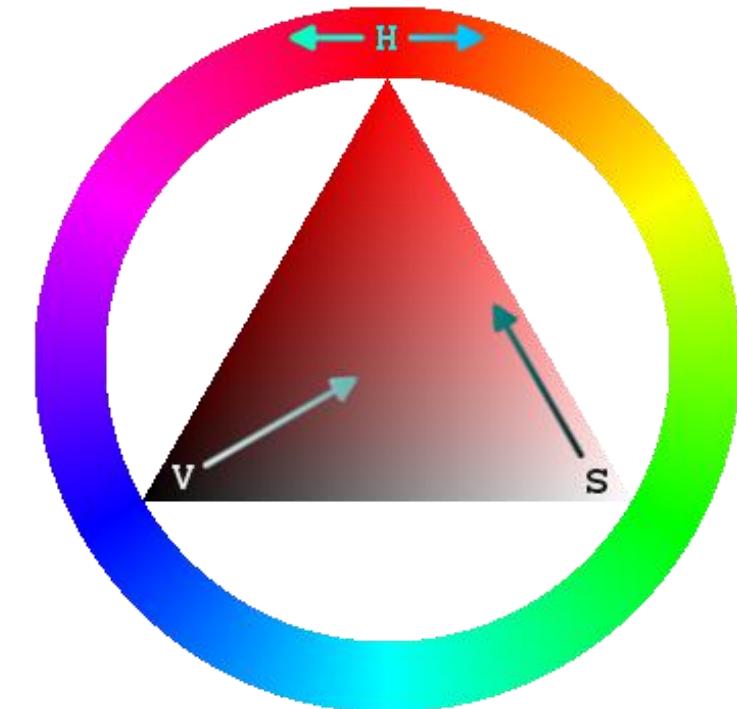
Profundidad de color

- Entonces: ¿Qué cantidad de bits necesitamos para ver una imagen con “buena” calidad?
- Un byte para cada canal. Hasta el momento, el estándar 24 bits (8x3) *TrueColor* sigue siendo uno de los más utilizados.
- $24 \text{ bits} = 2^{24} = 16.777.216 \text{ colores}$
- Si bien no hay un consenso, se supone que esta cantidad supera el alcance de la visión humana.
- Existen sistemas más nuevos que incluyen transparencias (32 bits), etc.

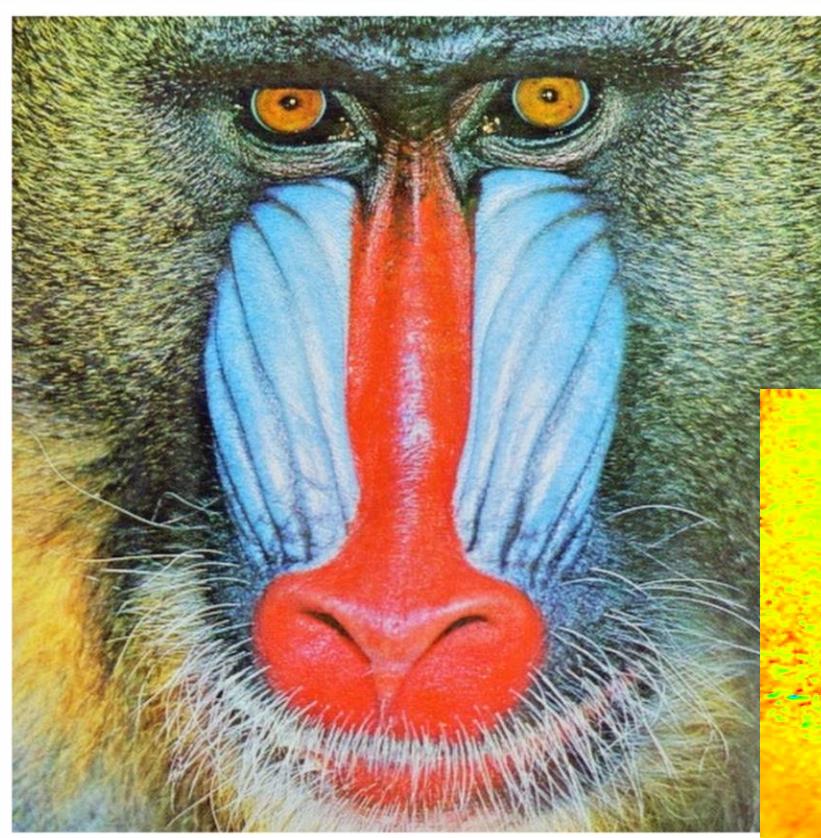


Modelos de color - HSV

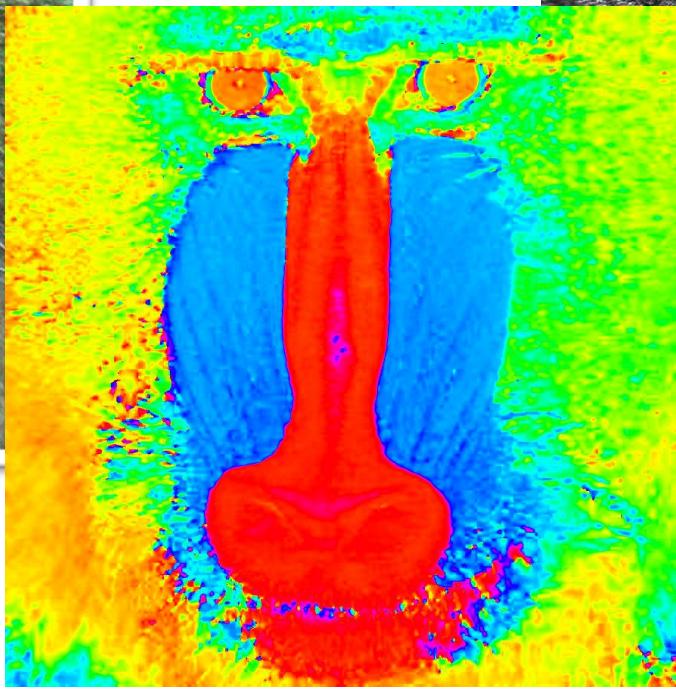
- El modelo RGB no es el único modelo de color. Existen diversos sistemas.
- Uno de los más utilizados para procesamiento de imágenes es el modelo HSV (Hue-Saturation-Value), debido a cómo almacena la información de los colores.
- Descompone la imagen en 3 canales:
 - Tono (Hue): nos muestra la información de color del pixel.
 - Saturación (Saturation): indica la intensidad del color
 - Valor (Value). Brillo. Es un modo de convertir la imagen a escala de grises.



Modelos de color - HSV



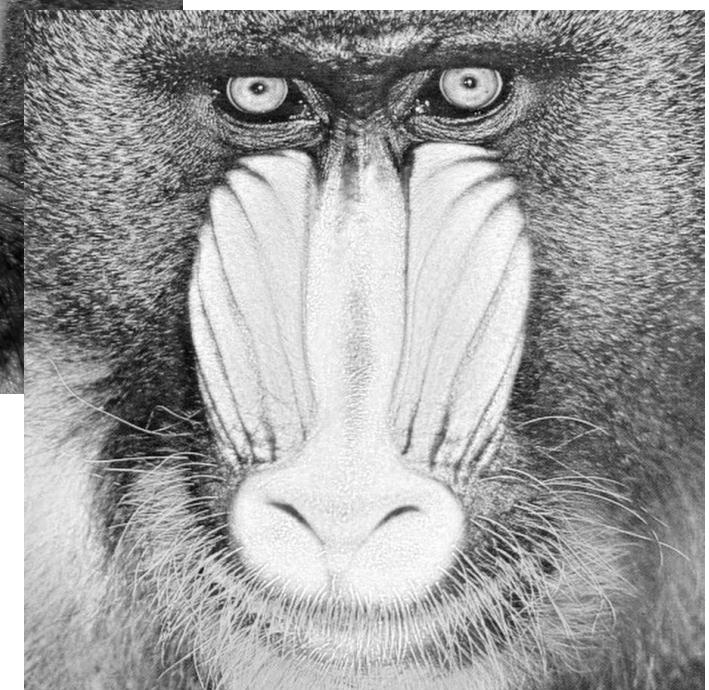
Tono



Saturación



Intensidad



RGB → Grayscale

- Una tarea muy común en procesamiento de imágenes es convertir una imagen color a escala de grises.
- Ya que cada canal RGB presenta información de luminosidad, el modo trivial de hacer esta conversión es calculando el promedio de los tres canales:

$$gray = \frac{R + G + B}{3}$$

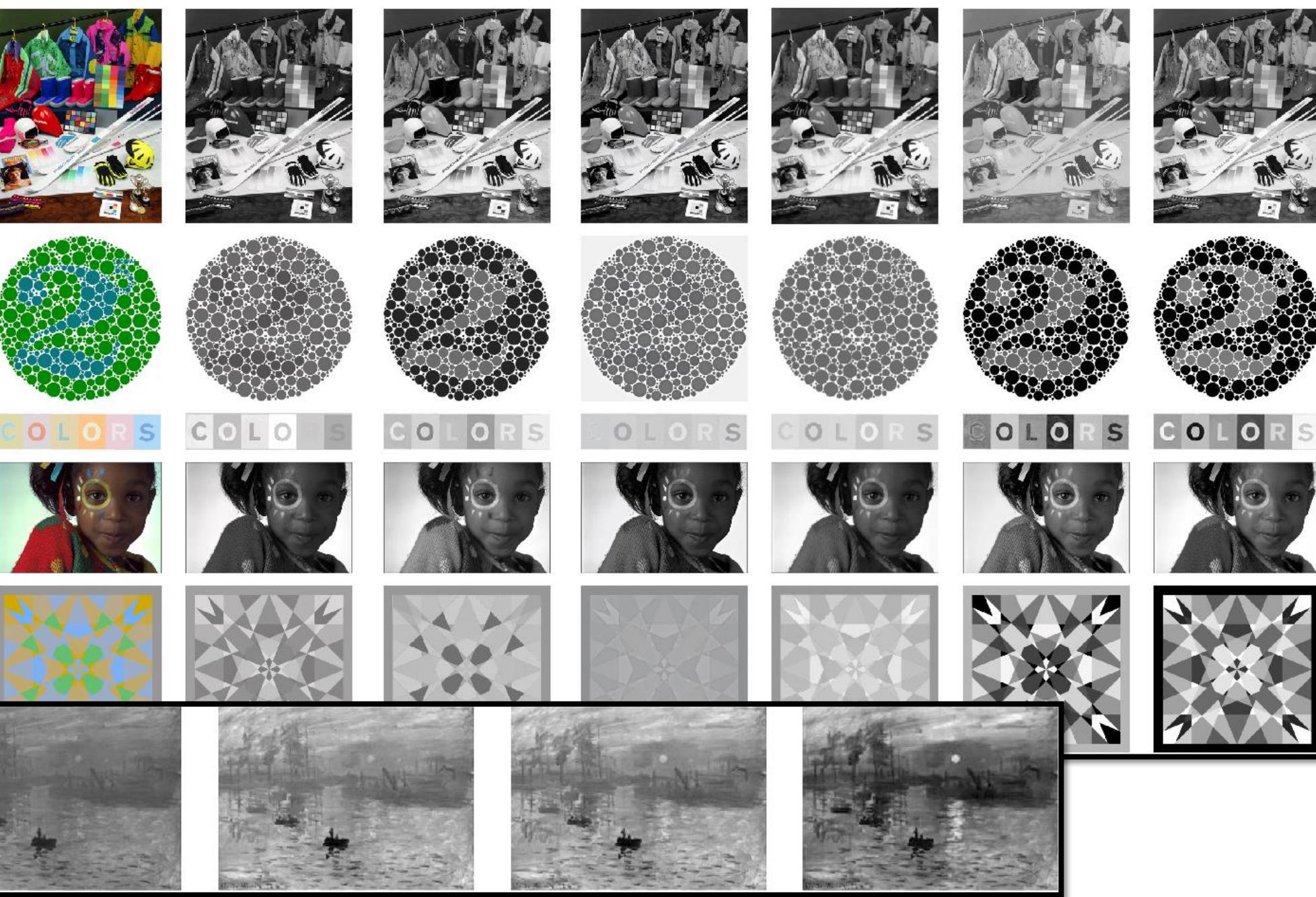
- Sin embargo la sensibilidad del ojo humano no es constante a través del espectro visible. El brillo percibido es mayor para los tonos verdes, menores para los rojos y aún menos para los azules. Entonces, podría aproximarse esta percepción con la siguiente ecuación:

$$gray = \frac{30R + 60G + 10B}{3}$$

RGB → Grayscale



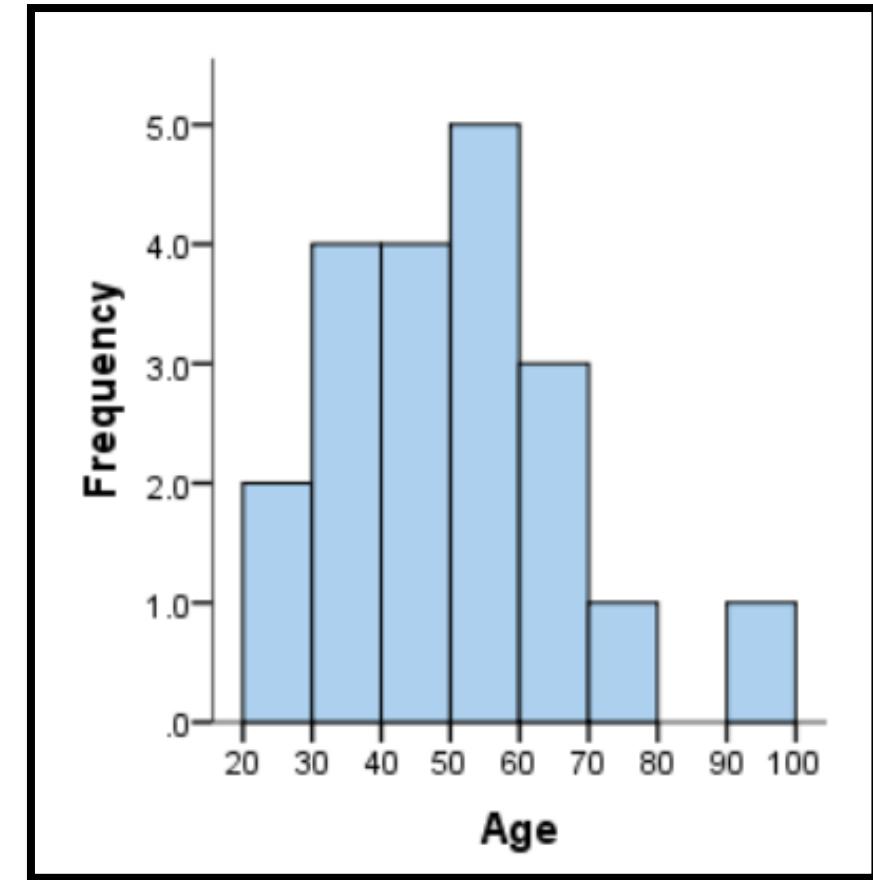
Otras
conversiones
 $RGB \rightarrow gray$



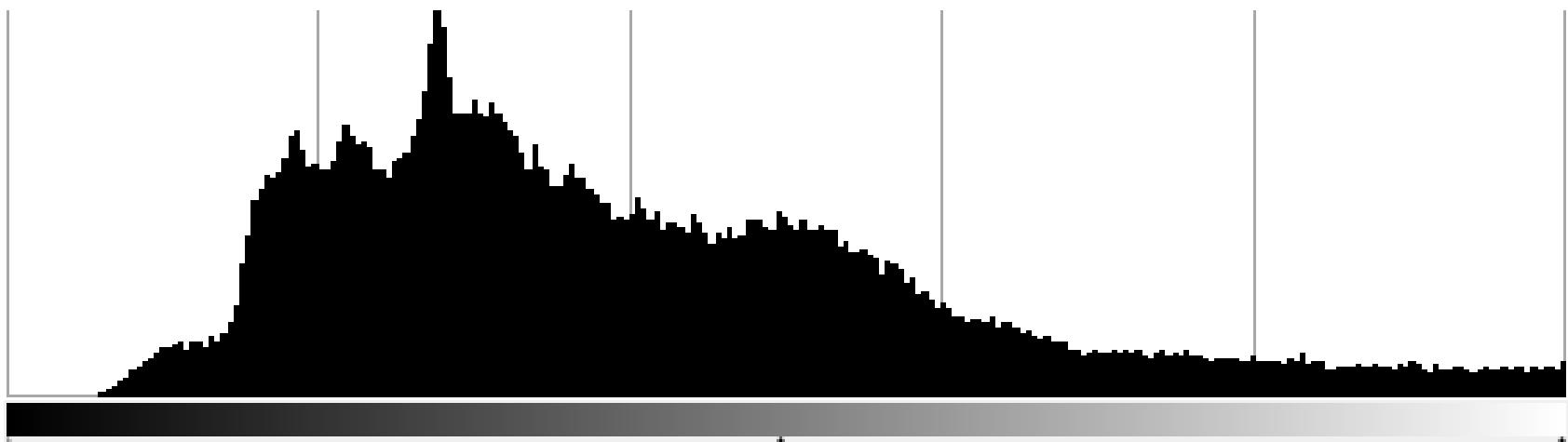
Histograma

36	25	38	46	55	68	72	55	36	38
67	45	22	48	91	46	52	61	58	55

- Un histograma es una representación de la distribución de los datos en una variable.
- Permite ver (estimar) la distribución de probabilidad de la variable.
- El rango de datos es dividido en N partes iguales (*bins*) y se contabiliza cuántos valores entran en cada rango.



Histograma



Histograma



Imagen con alto contraste

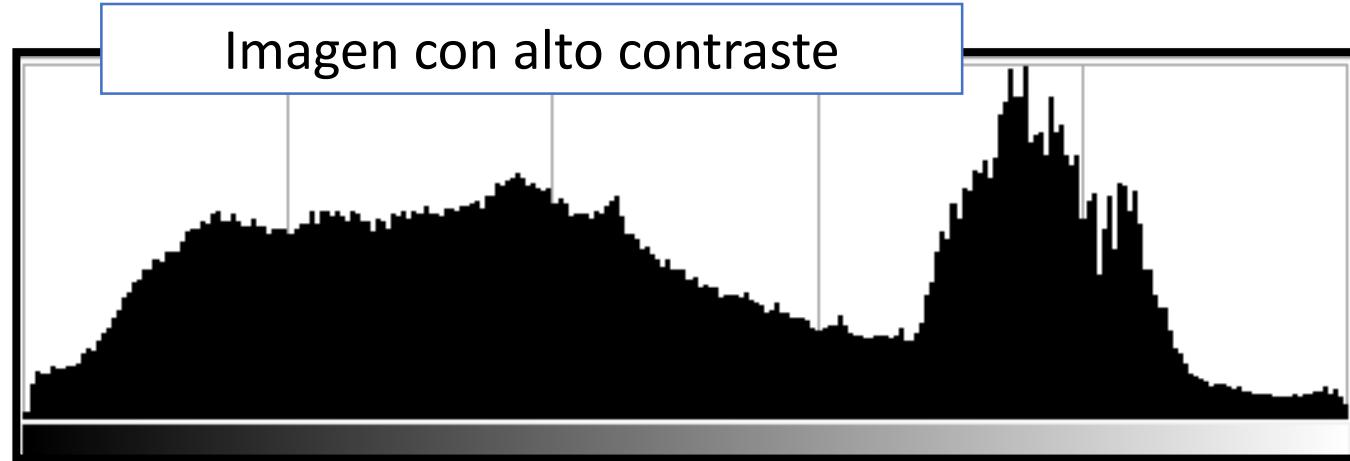
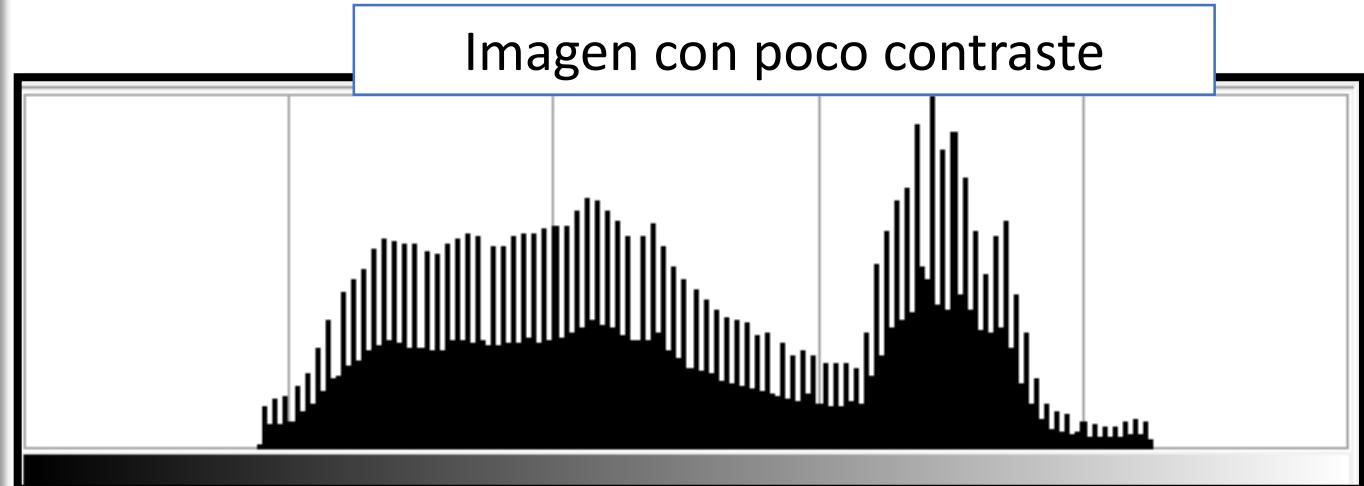
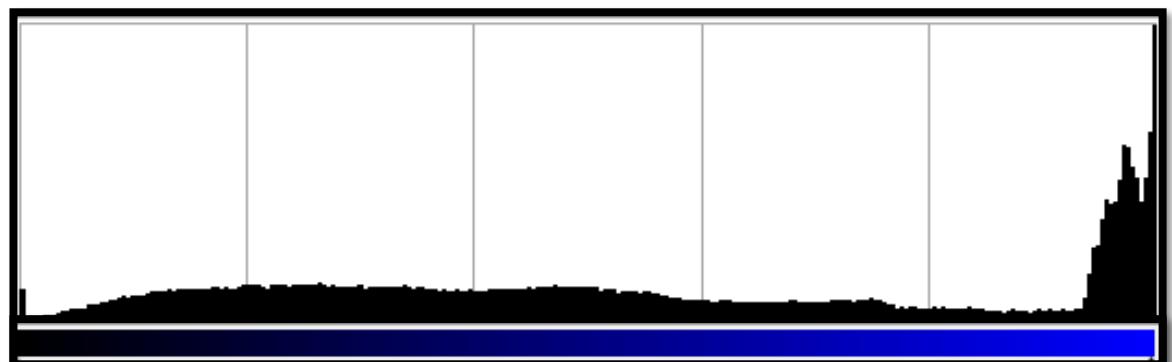
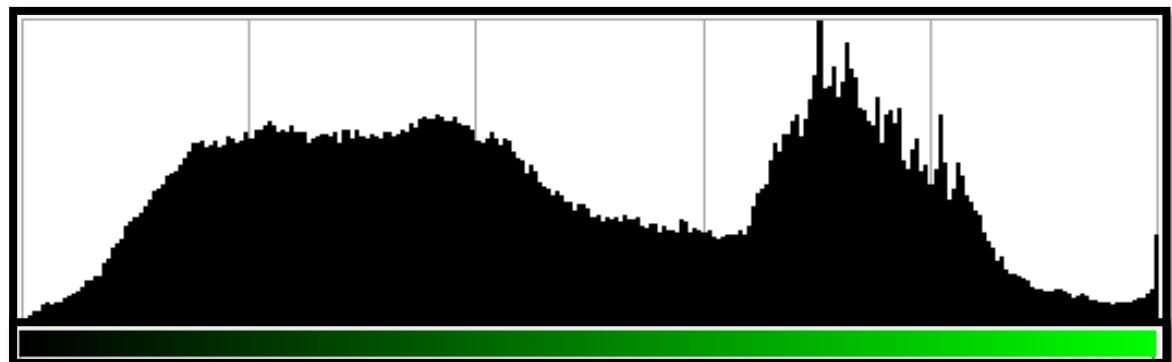
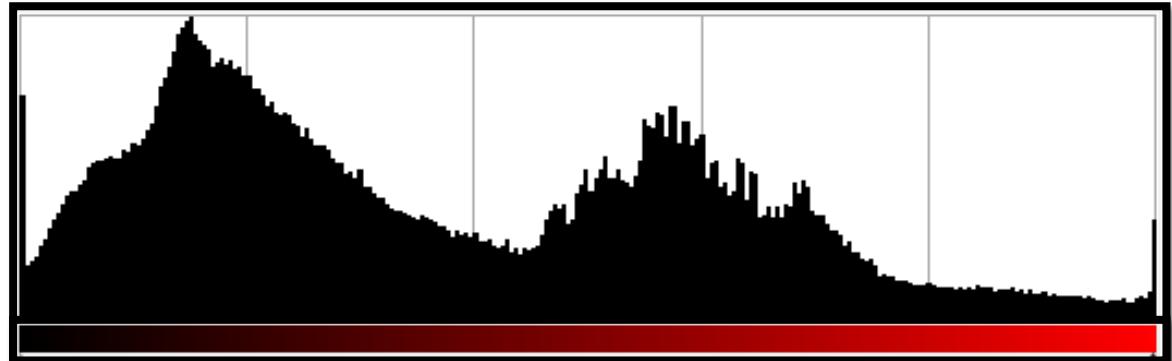


Imagen con poco contraste



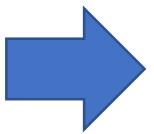
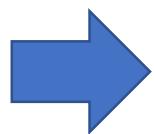
Histograma color



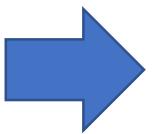
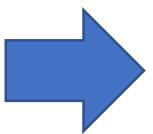
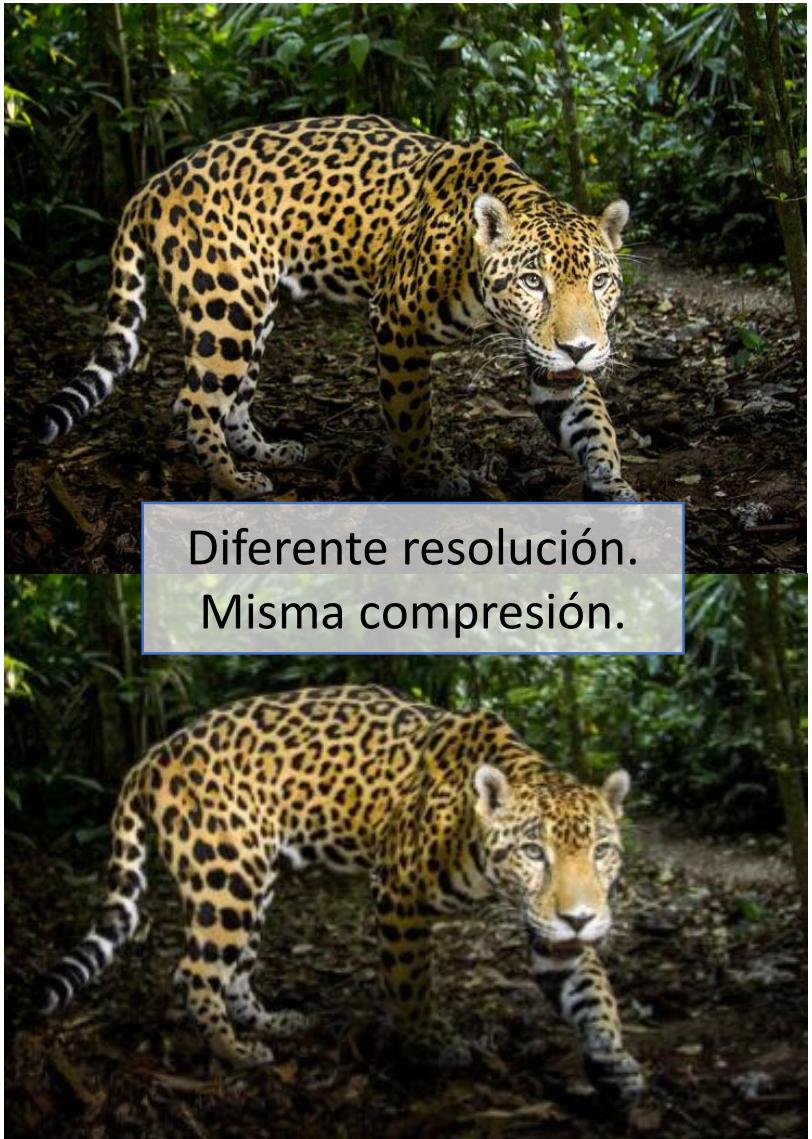
Problemas de codificación



Misma resolución.
Diferente compresión.



Problemas de codificación

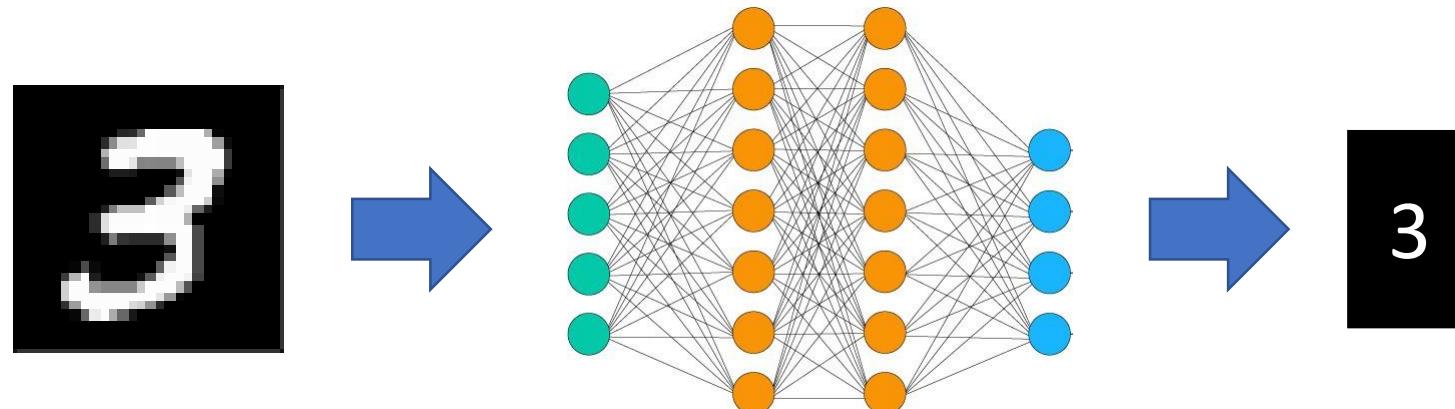


CLASIFICACION DE IMÁGENES

Clasificación de imágenes

Para crear un modelo que permita clasificar imágenes es necesario tener en cuenta algunas consideraciones:

- Las imágenes tienen los pixeles en formato matricial. Un modelo neuronal no acepta esto. Es necesario “aplanar” la imagen.
- Si es una imagen color, el problema se triplica al tener los 3 canales.



Clasificación de imágenes

Imagen original

0	1	2
3	4	5
6	7	8

Aplanar

Imagen aplanada

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

```
# Reshape de una imagen individual  
img = np.reshape(img, (img.shape[0]* img.shape[1]))  
  
# Reshape de todo el dataset  
X = np.reshape(X, (X.shape[0], X.shape[1]*X.shape[2]))
```

```
# En Keras se agrega una capa “flatten” al comienzo para que aplane la imagen.  
model.add(Flatten(input_shape= X_train[0].shape))
```

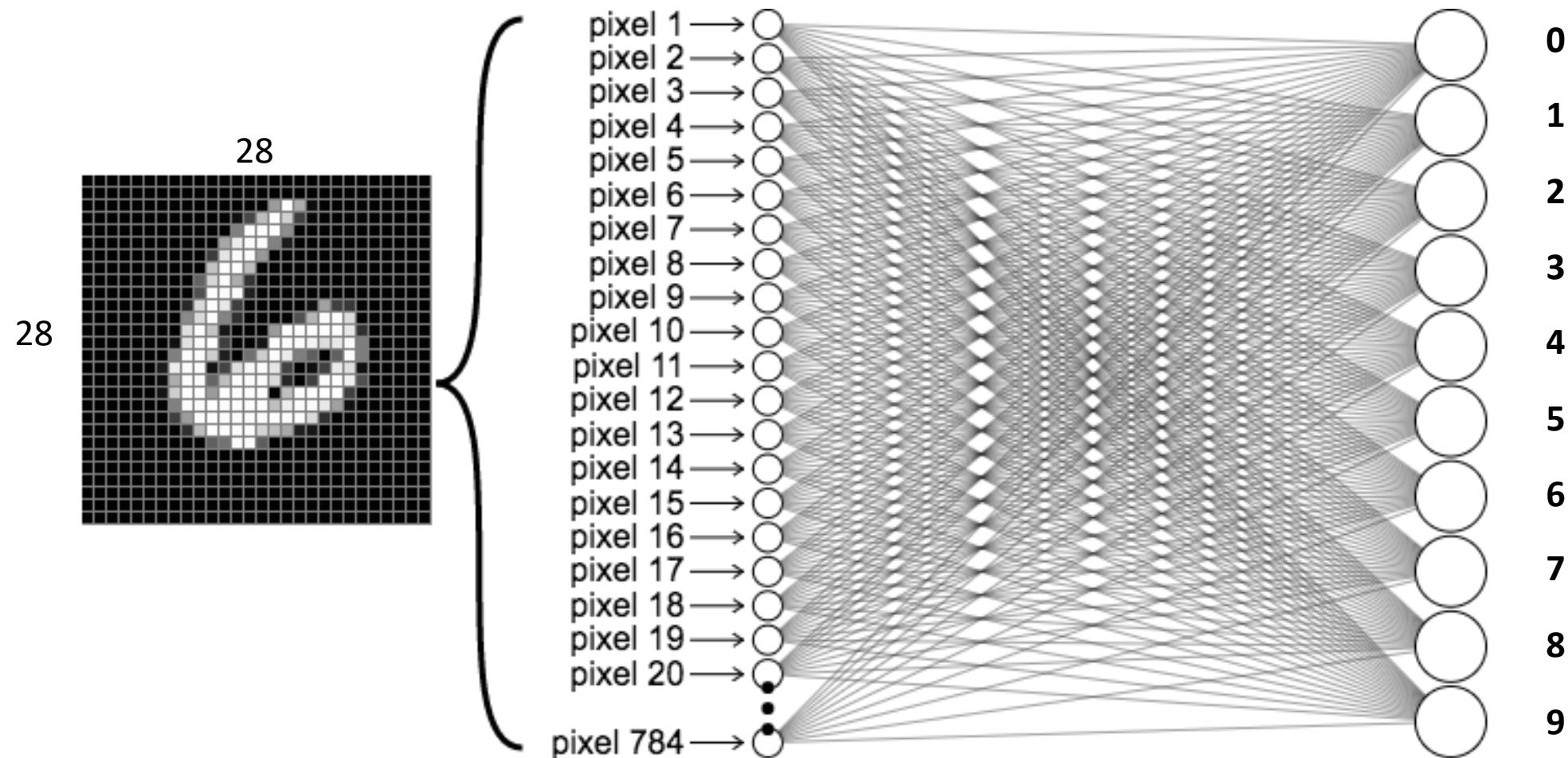
Clasificación de imágenes - MNIST

MNIST es una base de datos con decenas de miles de dígitos escritos a manos. El objetivo es entrenar un clasificador que clasifique correctamente estos números. Cada dígito está codificado en 28x28 pixeles en escala de grises.



Clasificación de imágenes - MNIST

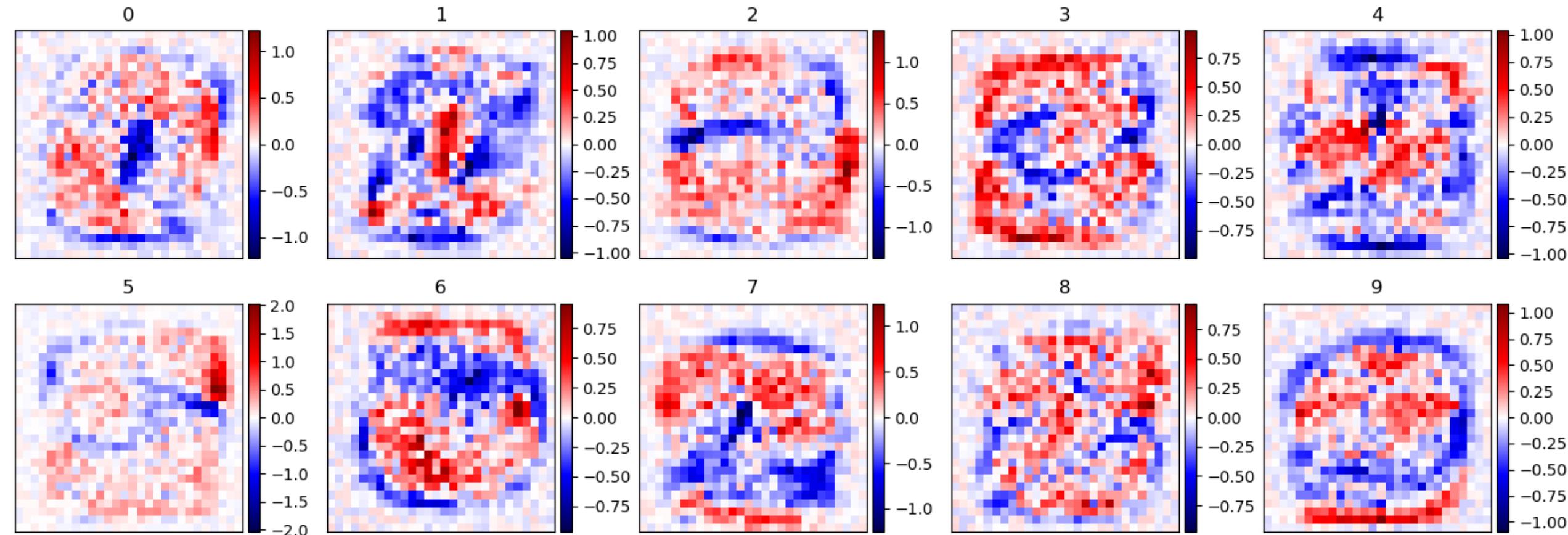
Un clasificador lineal (el más simple que podemos realizar)



Clasificación de imágenes - MNIST

Vector de pesos aprendidos por cada clase (graficados en forma de matriz).

```
w_r= np.reshape(w, (28,28,10))
```



Clasificación de imágenes – CIFAR10

CIFAR10 es una base de datos mucho más realista que contiene decenas de miles de imágenes con diferentes objetos como animales y vehículos.

Cada imagen está codificada en 32x32 pixeles con 3 canales de color.

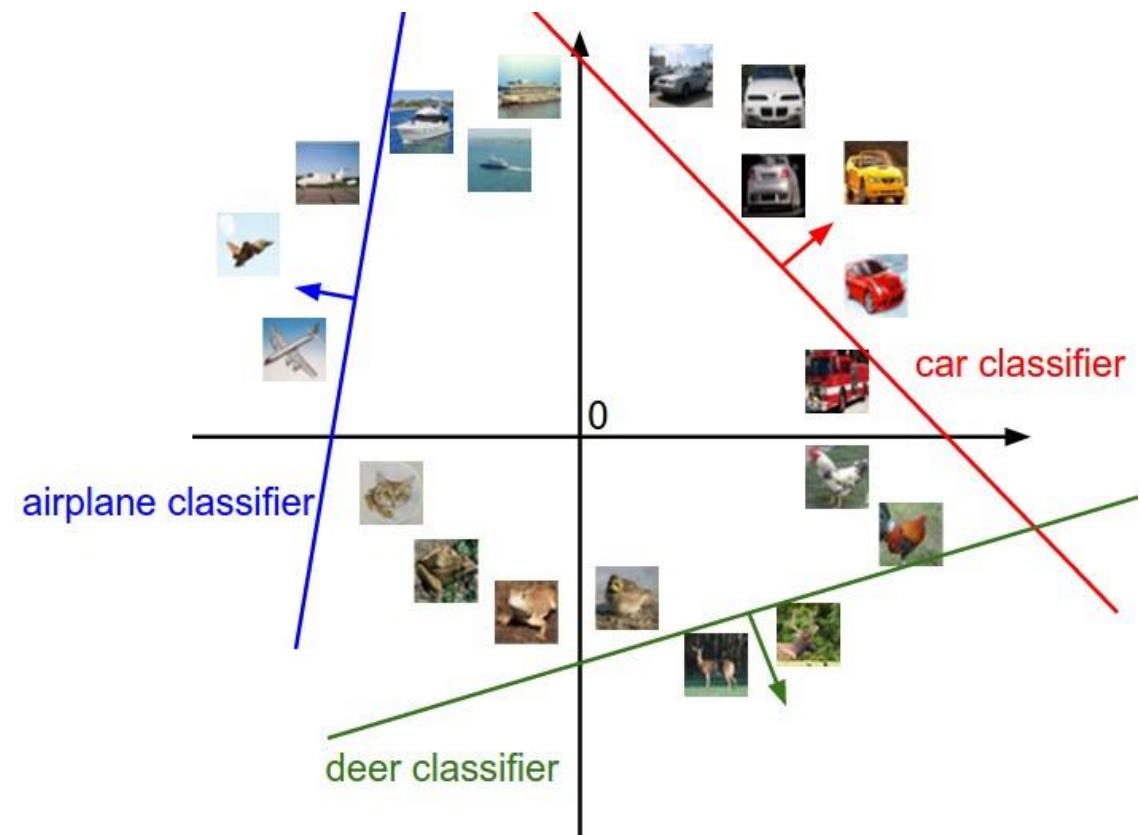
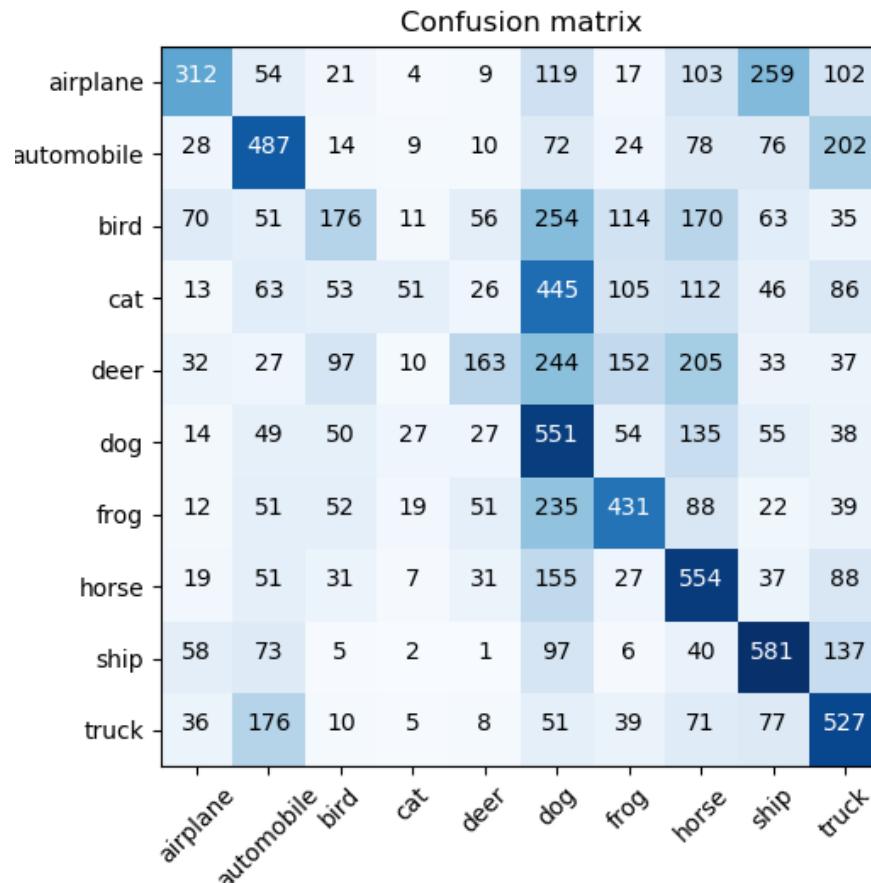


Clasificación de imágenes – CIFAR10

38% de Acc es mejor que $1/10 = 10\%$ (que sería tirar una moneda).

Estos resultados son esperables ya que no es un problema simple. Las imágenes son muy distintas y con mucho ruido.

Train	Accuracy: 0.42	soporte: 50000
Test	Accuracy: 0.38	soporte: 10000

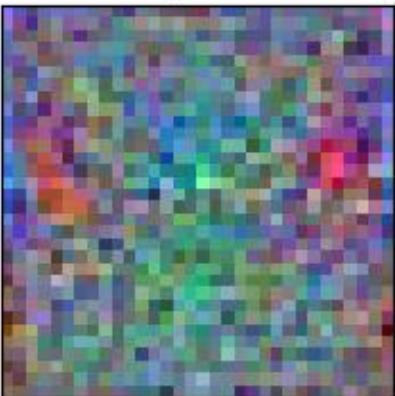


Clasificación de imágenes – CIFAR10

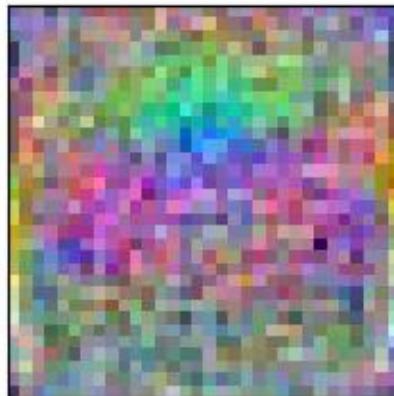
Vector de pesos aprendidos por cada clase (graficados en forma de matriz).

```
w_r= np.reshape(w, (32,32,3,10))
```

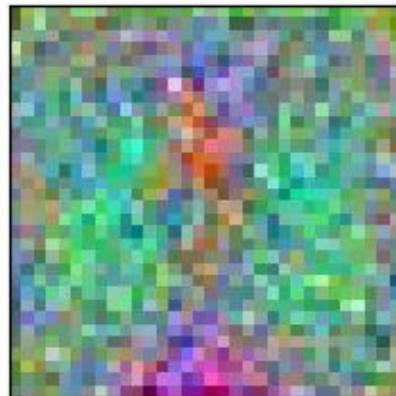
airplane



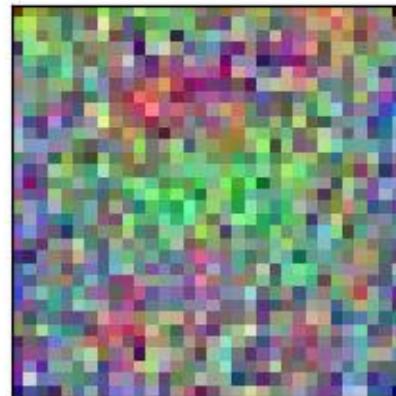
automobile



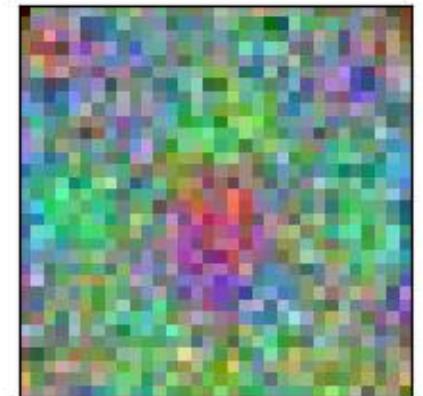
bird



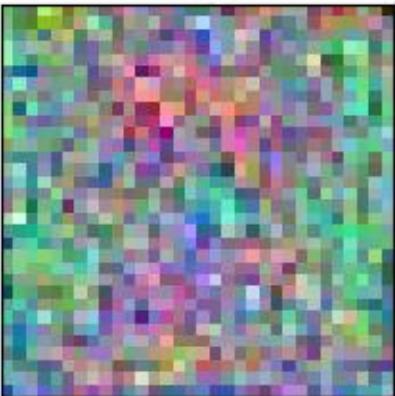
cat



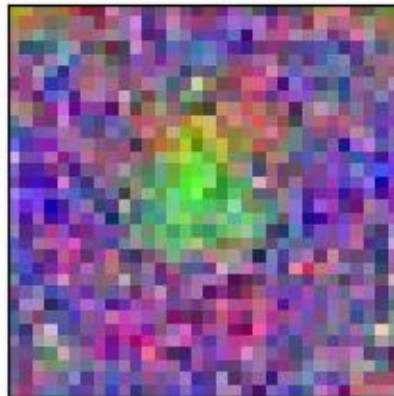
deer



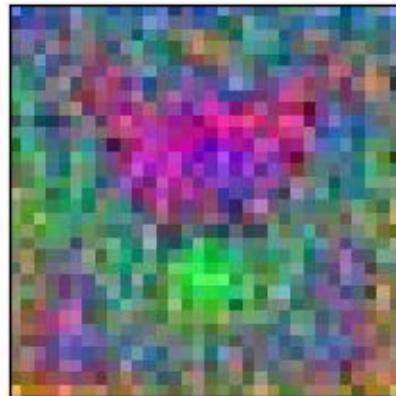
dog



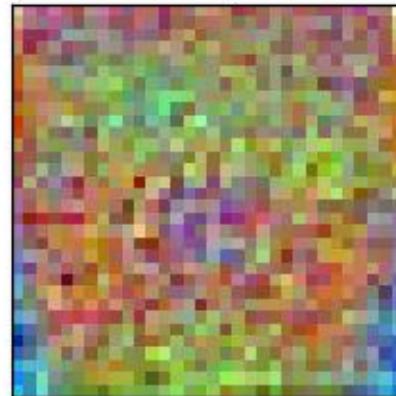
frog



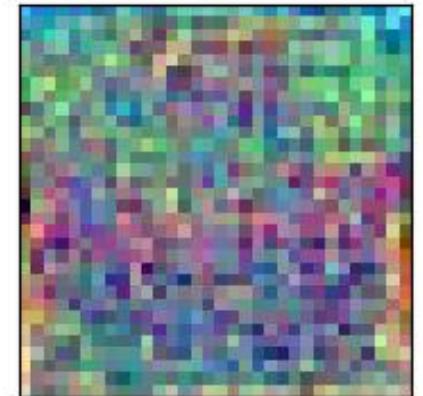
horse



ship

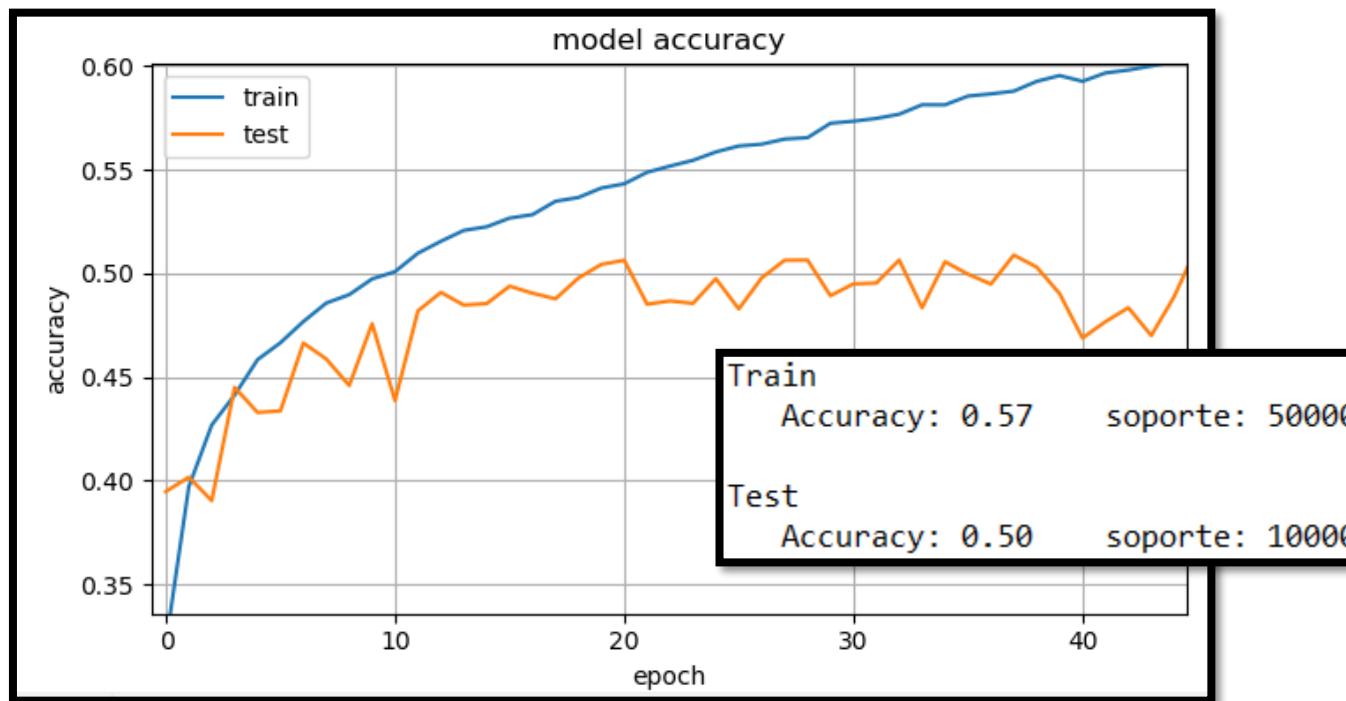


truck



Clasificación de imágenes – CIFAR10

Al agregar una capa oculta a la red elevamos el Acc a 50% (para el testing set). Cuantas más capas ocultas, corremos riesgo de caer en Overfitting.



Confusion matrix											
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
airplane	497	7	156	21	15	23	27	59	155	40	
automobile	45	400	37	33	10	34	26	38	122	255	
bird	50	8	509	65	77	78	105	79	14	15	
cat	19	1	157	288	53	198	135	89	25	35	
deer	36	1	221	48	353	45	148	112	25	11	
dog	10	2	139	197	40	391	95	95	22	9	
frog	3	4	83	62	86	44	652	39	13	14	
horse	27	5	101	43	51	75	31	629	14	24	
ship	84	19	48	22	24	39	12	24	664	64	
truck	39	67	40	40	12	26	29	88	91	568	

Arquitectura

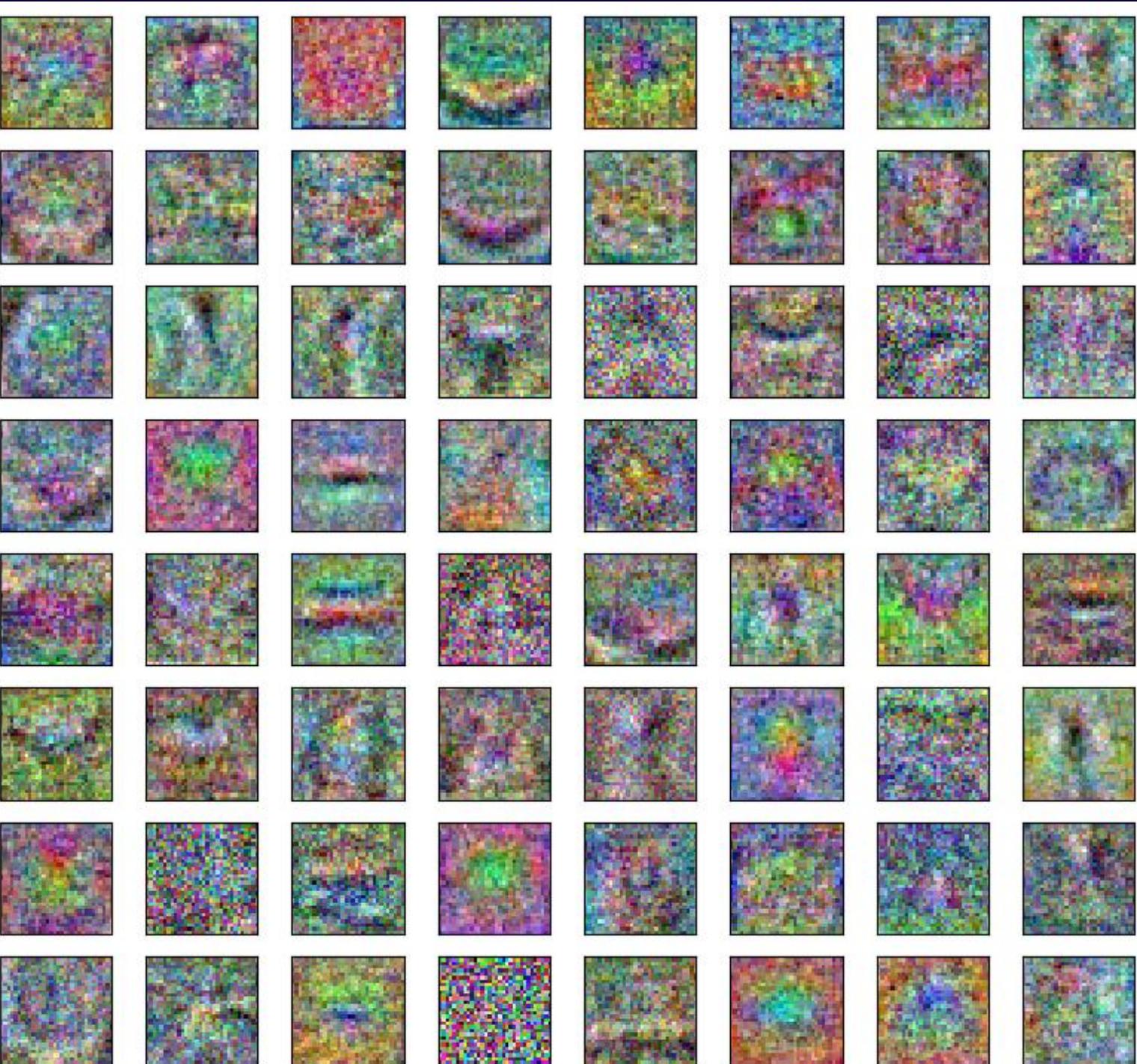
Flatten()

Dense(64, 'Relu')

Dense(10, 'softmax')

Pesos de las 64 neuronas de la capa oculta.

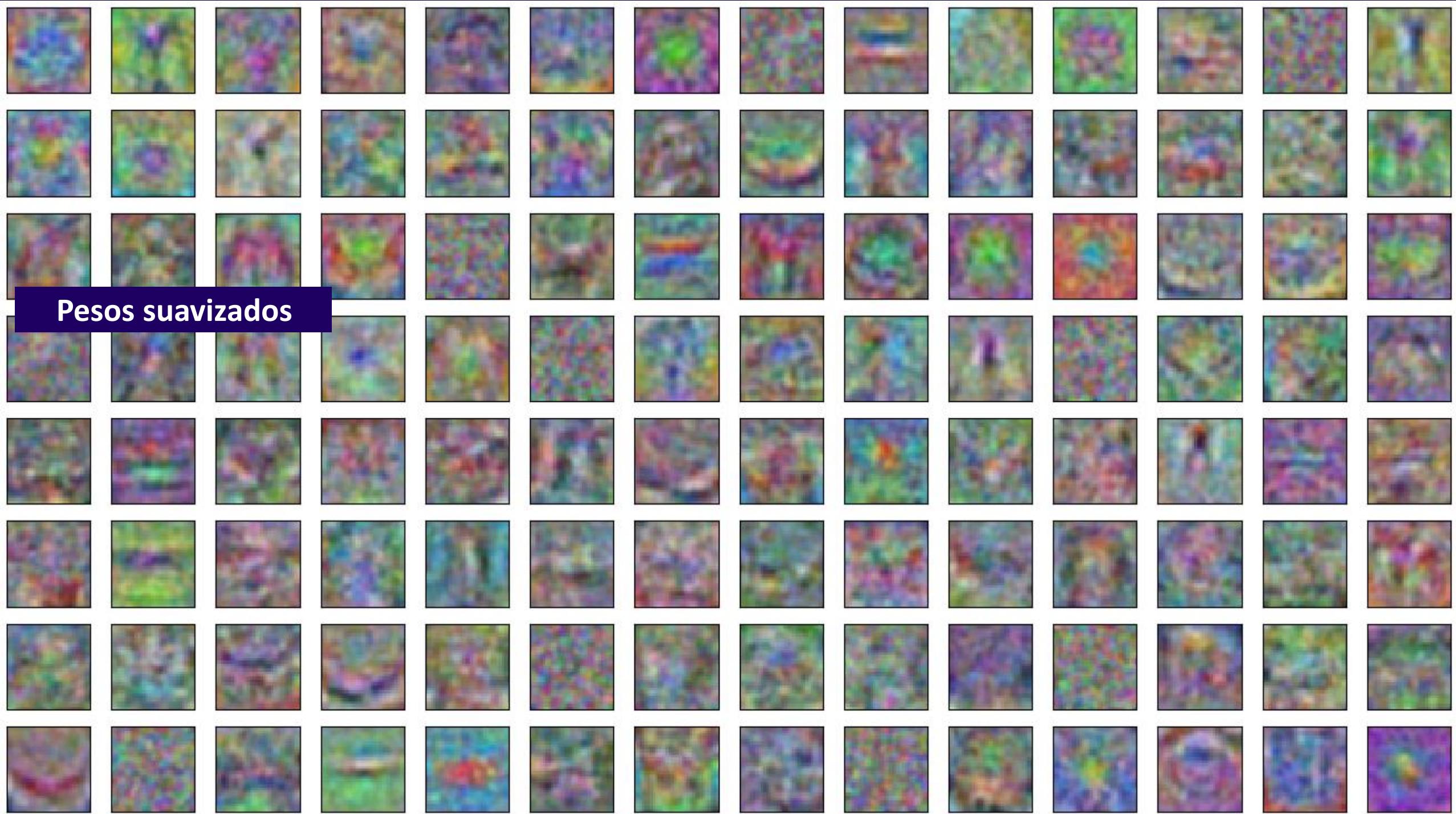
Notar que pareciera que algunas neuronas "encuentran" ciertos patrones en las imágenes.



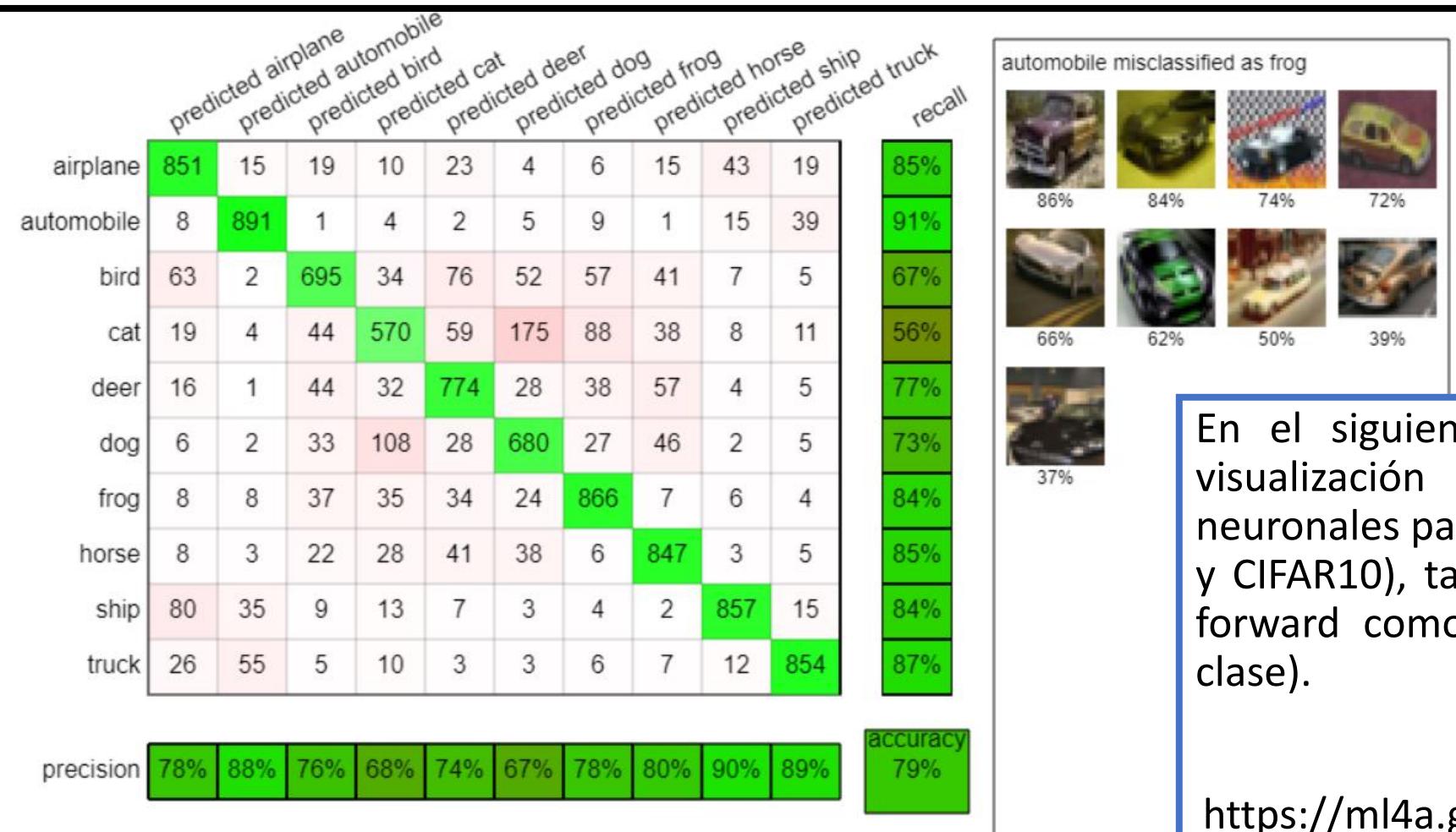


112 neuronas ocultas

Pesos suavizados



Visualización



En el siguiente link podrán encontrar una visualización del entrenamiento de redes neuronales para las dos bases de datos (MNIST y CIFAR10), tanto con redes neuronales feed-forward como con convolucionales (próxima clase).

[https://ml4a.github.io/ml4a/looking inside neural nets/](https://ml4a.github.io/ml4a/looking_inside_neural_nets/)