

[Open in app ↗](#)

Search



Write



Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



# Clasificación de imágenes con redes profundas



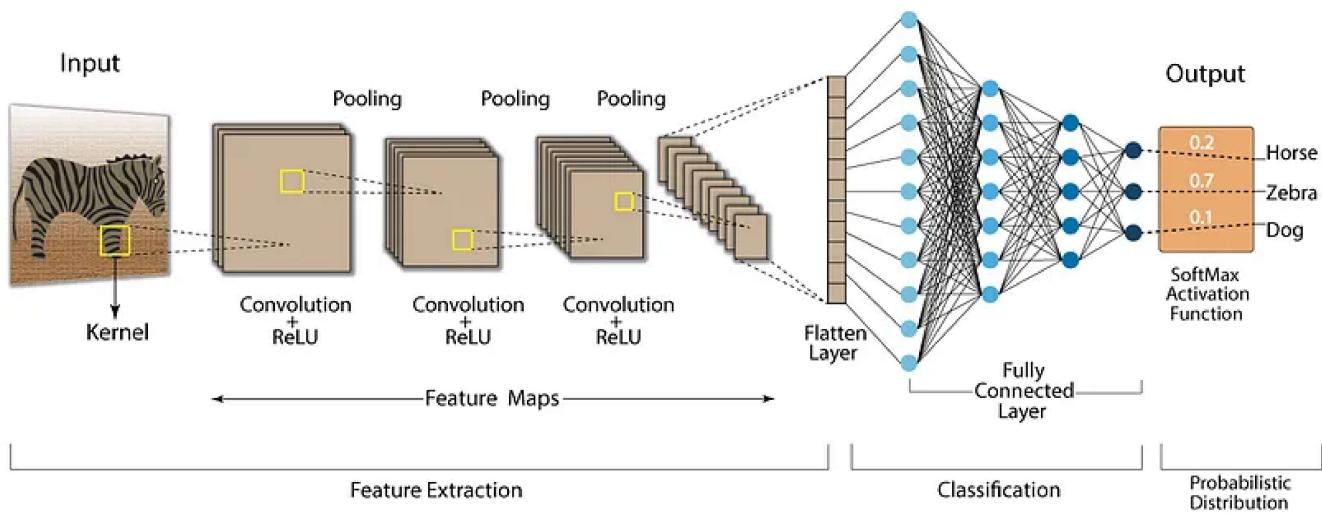
Mayra Sarahí de Luna Castillo

12 min read · 2 days ago



...

**Convolution Neural Network (CNN)**



Ejemplo de CNN

En la era de la revolución digital, el análisis de imágenes se ha convertido en una empresa crucial, impulsada en gran medida por los avances en

inteligencia artificial. En este trabajo se hará uso de las redes neuronales profundas a tres conjuntos de bases de datos (mismas que en los otros dos blogs). El objetivo principal es que a través de las arquitecturas neuronales se encuentre un nivel alto de precisión en tareas de clasificación y reconocimiento.

Como se mencionó anteriormente, estas bases de datos ya fueron usadas en dos blogs para implementar distintos algoritmos (SVM, SVM base radial, Perceptrón Multicapa Y SOM), si te interesa ver el trabajo que se realizó te dejo los links.

### **Clasificación de imágenes con métodos clásicos y redes neuronales**

Clasificación de imágenes con

SVM, SVM RADIAL Y PERCEPTRÓN MULTICAPA

### **Agrupamiento de imágenes**

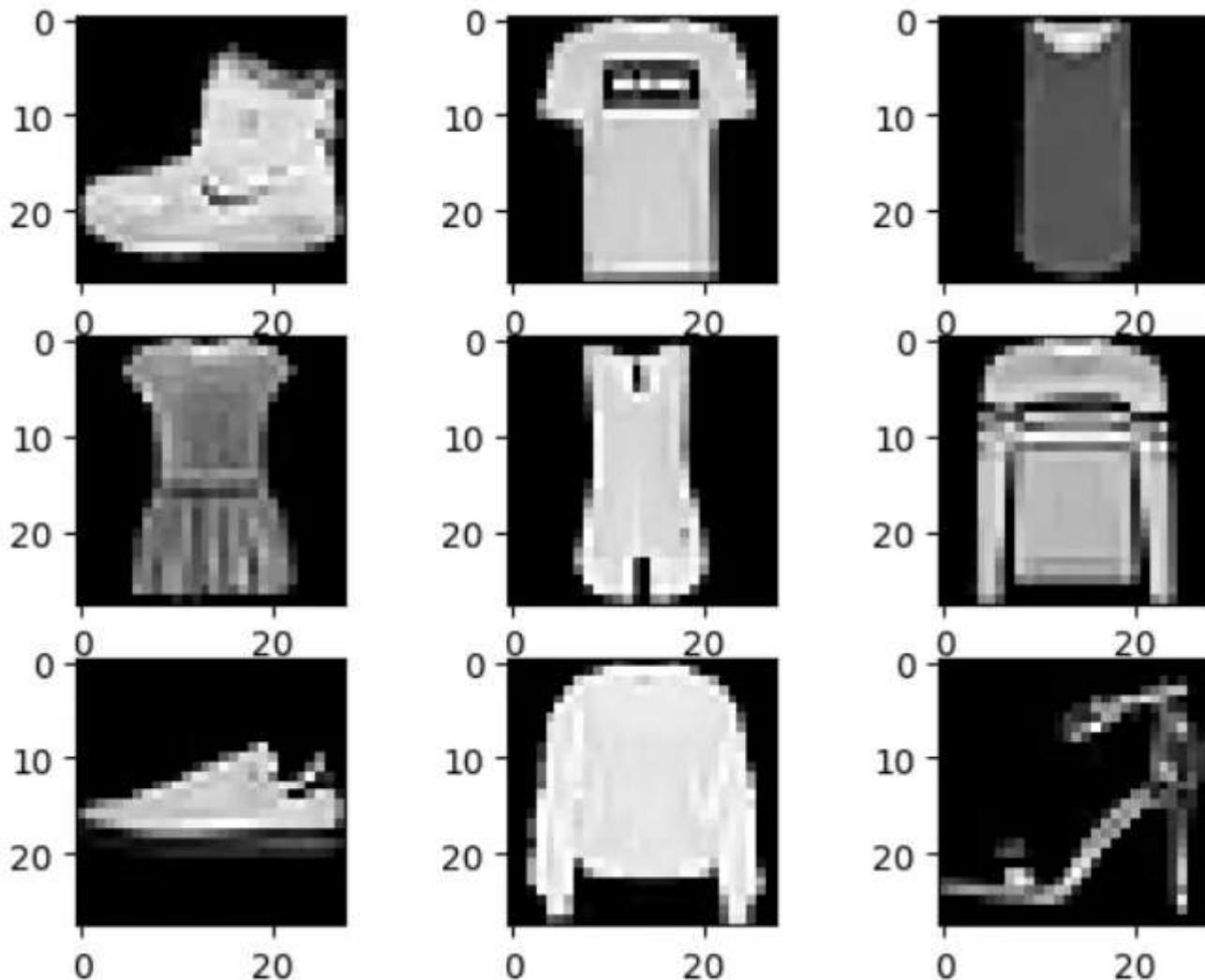
Agrupamiento de imágenes

Usando SOM

- **Descripción de los datos utilizados para esta parte del proyecto.**

Para realizar este proyecto manejamos tres bases de datos. La primera es la Fashion-MNIST con 70000 imágenes escaladas y normalizadas (60000 de entrenamiento y 10000 de prueba) de 10 clases de prendas de vestir (blusa, pantalón, suéter, vestido, abrigo, zapatos, playera, tenis, bolsa, botas) en

escala de grises de tamaño 28x28. Las imágenes están perfectamente centradas y las distintas prendas tienen la misma figura.



Ejemplo de la base de datos Fashion-MNIST

La segunda es una base de 2000 fotografías satelitales de ambientes de México. Las fotografías son de alta calidad y muestran imágenes de Agua, Bosque, Ciudad, Cultivo, Desierto y Montaña.



Ejemplo de la base de datos de fotografías satelitales

Por último, para la tercera base de datos, se utilizó una base de datos de 2,500 imágenes tomadas por nosotros, se tomaron 500 imágenes (con diferentes ángulos e iluminación) de cada una estas imágenes son de un zapato, una gorra, un celular, un lápiz y un libro.



Ejemplo de la base de datos de fotografías propias

En etapas previas, se empleó un enfoque de extracción de características para adquirir los histogramas de color y las matrices de co-ocurrencia de las últimas dos bases de datos. No obstante, al abordar esta situación con una arquitectura de red reconocida por su eficacia en el procesamiento y clasificación de imágenes, prescindiremos de los modelos de extracción previamente implementados.

- **Descripción de la teoría de los modelos entrenados.**

Las redes neuronales constituyen la esencia del aprendizaje profundo y se configuran con capas de nodos, que incluyen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo establece conexiones con

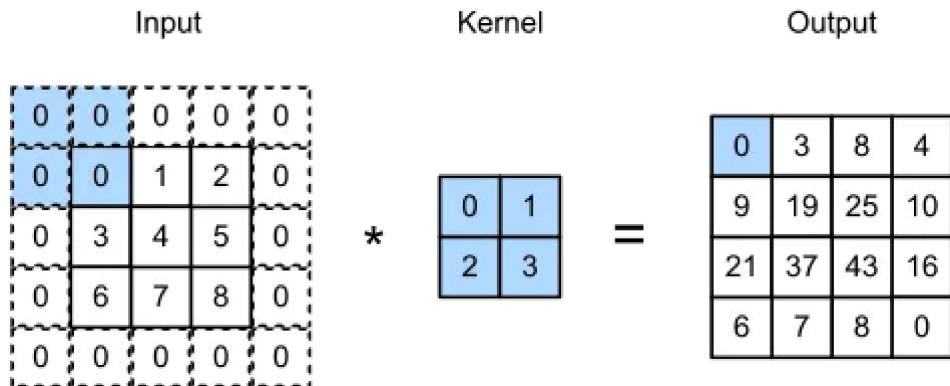
otros nodos y posee pesos y umbrales asociados. Cuando la salida de un nodo supera el umbral especificado, se activa y transmite datos a la siguiente capa de la red; de lo contrario, no se transmiten datos a la capa subsiguiente.

## CNN (Red Neuronal Convolucional)

Las redes neuronales convolucionales son comúnmente empleadas en tareas de clasificación y visión por computadora. Estas ofrecen un enfoque más escalable para clasificar imágenes y realizar reconocimiento de objetos, aprovechando principios de álgebra lineal, en particular la multiplicación de matrices, para detectar patrones en una imagen.

### Capas de convolución

Para llevar a cabo la convolución en una imagen, se utiliza un kernel, una matriz de tamaño fijo que se desplaza sobre la imagen multiplicando entrada por entrada y sumando los resultados. Esto permite resumir la información con un solo número, capturando información clave dentro de una imagen que no es directamente interpretable por humanos (semántica). El resultado de la convolución es un mapa de características, una matriz de menor dimensión derivada de la aplicación del kernel. A medida que se realizan más capas con convoluciones, los mapas de características pueden reconocer formas más complejas. El modelo aprende los valores óptimos para la matriz kernel, y el kernel puede desplazarse una, dos o más posiciones a la vez.



[Ejemplo de kernel para CNN.](https://medium.com/codex/kernels-filters-in-convolutional-neural-network-cnn-lets-talk-about-them-ee4e94f3319) Recuperado de <https://medium.com/codex/kernels-filters-in-convolutional-neural-network-cnn-lets-talk-about-them-ee4e94f3319>

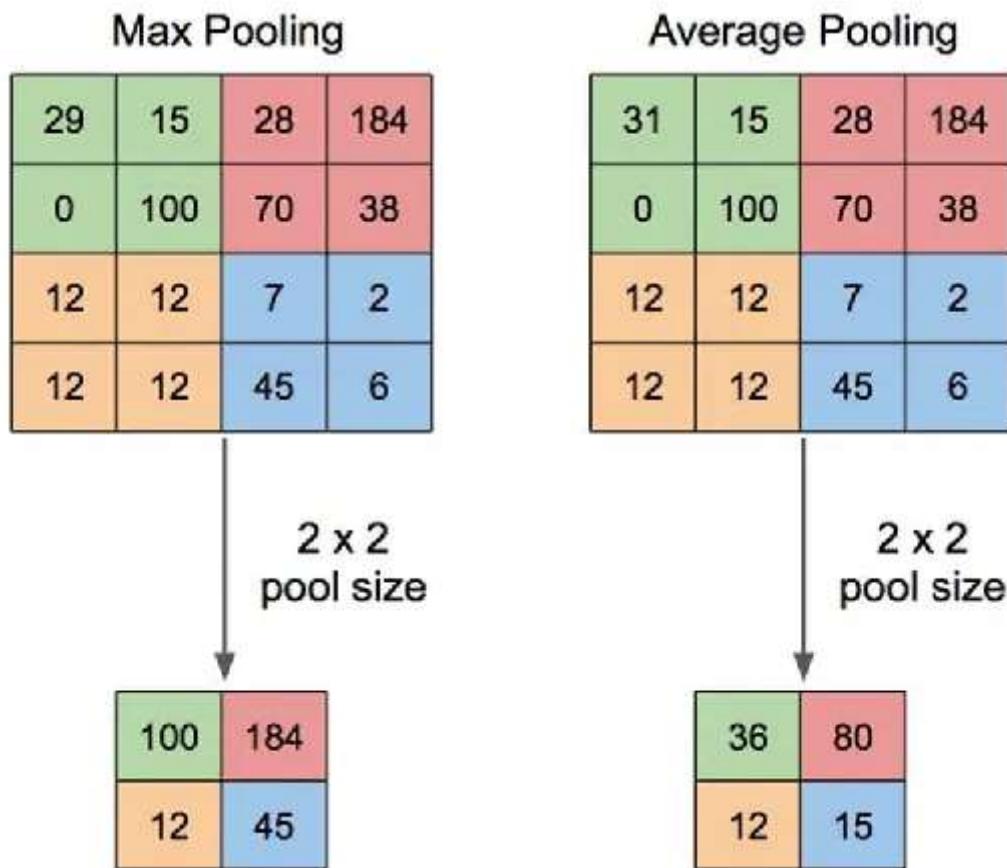
## Capas de agrupación

En ocasiones, aplicar una convolución no reduce la imagen lo suficiente. Las capas de agrupación, también conocidas como capas de reducción de dimensionalidad, reducen la dimensionalidad, disminuyendo el número de parámetros de entrada. Similar a la capa convolucional, la operación de agrupación desliza un filtro a lo largo de toda la entrada, pero a diferencia de la convolución, este filtro no tiene pesos.

Existen dos tipos principales de capas de agrupación:

1. **Max pooling (agrupación máxima):** Durante el desplazamiento del filtro sobre la entrada, se selecciona el píxel con el valor máximo para ser transmitido a la matriz de salida. Este método es ampliamente preferido en comparación con la agrupación promedio.
2. **Average pooling (agrupación promedio):** A medida que el filtro se desplaza sobre la entrada, calcula el valor promedio dentro del campo receptivo y lo envía a la matriz de salida.

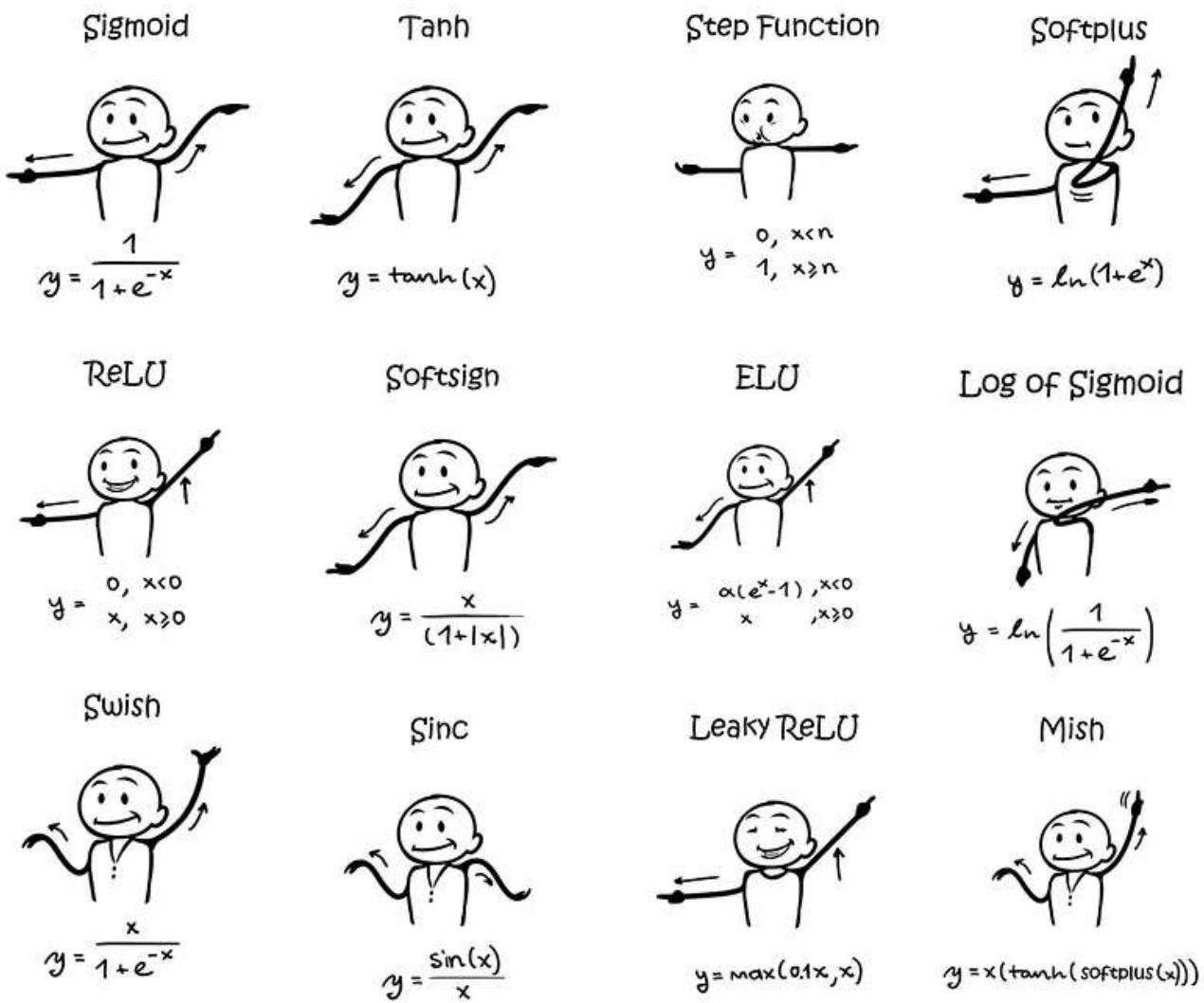
Aunque la capa de agrupación implica la pérdida de información, presenta diversos beneficios para la CNN: Contribuye a la reducción de la complejidad, mejora la eficiencia y mitiga el riesgo de sobreajuste. (Max Pooling and Average Pooling, s.f.)



Ejemplo de Max Pooling y Average Pooling. Recuperado de [https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max-fig2\\_333593451](https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max-fig2_333593451)

## Capas de función de activación

Esta capa es muy importante ya que las funciones introducen no linealidades esenciales, permitiendo a la red aprender patrones más complejos y realizar tareas más sofisticadas, como la clasificación y la detección de características en imágenes. La elección de la función de activación depende del problema específico y de las características deseadas en el modelo.



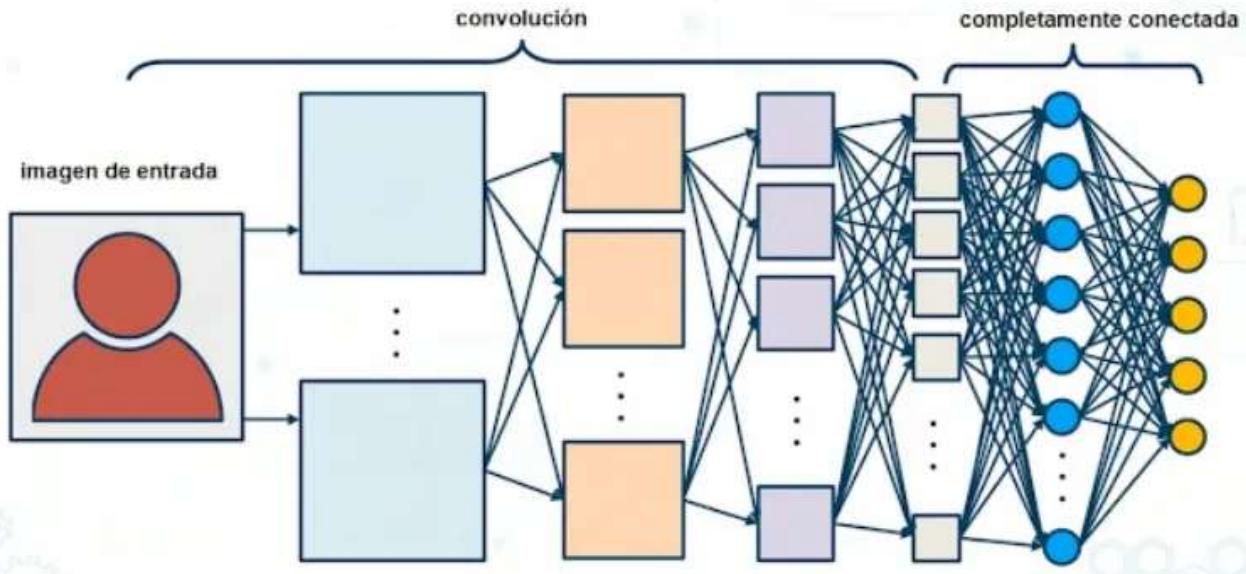
Ejemplos de función de activación. Recuperado de <https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow/>

## Capas Completamente Conectadas

En una capa completamente conectada, cada nodo o neurona está conectado a cada neurona de la capa anterior. Esto significa que cada entrada influye en cada salida, sin importar su posición espacial en las capas anteriores. La función de estas capas es combinar características aprendidas en las capas anteriores para realizar tareas más complejas de clasificación o regresión.

En el contexto de una CNN, las capas completamente conectadas generalmente se utilizan para realizar la clasificación final. Después de

extraer y aprender características importantes mediante capas convolucionales y de agrupación, las capas completamente conectadas toman esas características y las utilizan para clasificar la entrada en las diferentes categorías o clases del problema. (Kalita, 2023).



Ejemplo de capa completamente conectada. Recuperado de  
<https://analisisyprogramacionoop.blogspot.com/2020/05/introduccion-redes-neuronales.html>

## Métricas para el modelo de Red Neuronal Convolucional

Las métricas con las que se va a comparar los resultados de la CNN son las siguientes:

**Precision (Precisión):** Esta métrica nos dice qué tan preciso es nuestro modelo cuando predice algo como positivo. Es la proporción de predicciones correctas (verdaderos positivos) respecto a todas las predicciones positivas (verdaderos positivos más falsos positivos).

**Recall (Recuperación):** El recall indica qué tan bien nuestro modelo captura todos los casos positivos reales. Es la proporción de predicciones correctas

(verdaderos positivos) respecto a todas las instancias positivas reales (verdaderos positivos más falsos negativos).

**F1-score:** Esta métrica es como un promedio ponderado entre precision y recall. Busca un equilibrio entre ambas, brindando una medida más completa del rendimiento del modelo. Es particularmente útil cuando no queremos que una métrica domine sobre la otra.

**Support (Soporte):** El soporte simplemente nos da el número real de instancias de cada clase en nuestro conjunto de datos. Es útil para entender cuántas instancias de cada clase estamos tratando, lo que puede ser crucial para interpretar el rendimiento del modelo en un conjunto de datos desequilibrado. (Ponce, J, s.f.)

## Métricas para cada Época

**Loss (Pérdida):** La pérdida nos indica cuán equivocada está la red mientras se está entrenando. El objetivo es minimizar esta pérdida para que la red aprenda de manera más efectiva y haga predicciones más precisas.

**Accuracy (Precisión):** La precisión es la proporción de predicciones correctas en general. Mide cuántas de las predicciones totales de la red son correctas, ofreciendo una visión general del rendimiento.

**Val\_loss:** La pérdida de validación, o val\_loss, es la medida de cuán bien la red se está desempeñando en datos de validación que no fueron utilizados durante el entrenamiento. Evaluar esta pérdida en datos nuevos ayuda a entender cómo la red generaliza y se comporta ante ejemplos que no ha visto previamente.

**Val\_accuracy:** La precisión de validación, o val\_accuracy, es similar a la precisión, pero se calcula en los datos de validación. Mide qué tan bien la red puede hacer predicciones precisas en datos que no formaron parte del proceso de entrenamiento. Es una métrica crucial para asegurarse de que el modelo pueda generalizar bien a nuevas situaciones.

- **Implementación en código**

Primero se hace un re-shape para poder hacer compatibles las formas de las imágenes en las redes neuronales, además de normalizar los datos para que se encuentren entre 0 y 1. Para cada base de datos el re-shape fue distinto. A continuación se muestra el código para el reshape del Fashion-MNIST.

```
# Reshape los datos (Las CNN esperan una imagen)
x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

Además, llevamos a cabo la codificación one-hot para las etiquetas de clases, una práctica esencial en tareas de clasificación multiclas.

```
y_train_categorical = to_categorical(y_train, num_classes)
```

```
y_test_categorical = to_categorical(y_test, num_classes)
```

Después se realiza la implemetación de la CNN con el siguiente código, para las tres bases de imágenes se usa la misma cantidad de capas.

```
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(150, 150, 3))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Dropout(0.25))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))
```

El modelo hace lo siguiente

- 1. Capa Conv2D (Convolucional):** 32 filtros con un tamaño de kernel de (3, 3) y función de activación ReLU.
- 2. Capa Conv2D (Convolucional):** 64 filtros con un tamaño de kernel de (3, 3) y función de activación ReLU.
- 3. Capa MaxPooling2D (Agrupación Máxima):** Reduce la dimensionalidad utilizando un tamaño de agrupación de (2, 2).
- 4. Capa Dropout:** Desactiva aleatoriamente el 25% de las neuronas para prevenir el sobreajuste.
- 5. Capa Flatten:** Transforma la salida en un vector unidimensional.

6. **Capa Dense (Completamente Conectada):** 128 neuronas con función de activación ReLU.

7. **Capa Dropout:** Desactiva aleatoriamente el 50% de las neuronas.

8. **Capa Dense (Completamente Conectada):** Número de neuronas igual a `num_classes` (clases de salida) con función de activación Softmax.

Se compila el modelo usando optimizador Adam y pérdida de entropía cruzada

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accu
```

Y finalmente se entrena el modelo, cada modelo hace diferentes épocas y batch size, en este caso se hicieron 10 épocas

```
# Entrenar el modelo
history = model.fit(X_train, y_train_categorical, epochs=10, validation_data=(X_
```

- Resultados obtenidos con los diferentes modelos y conjuntos de datos.

## Fashion-MNIST

Para la base de datos Fashion-MNIST primero se muestra la actualización de la CNN junto con sus métricas para cada época.

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Epoch 1/10
469/469 [=====] - 151s 318ms/step - loss: 0.5348 - accuracy: 0.8123 - val_loss: 0.3530 - val_accuracy: 0.8697
Epoch 2/10
469/469 [=====] - 151s 322ms/step - loss: 0.3469 - accuracy: 0.8771 - val_loss: 0.2887 - val_accuracy: 0.8943
Epoch 3/10
469/469 [=====] - 147s 313ms/step - loss: 0.2997 - accuracy: 0.8925 - val_loss: 0.2763 - val_accuracy: 0.9011
Epoch 4/10
469/469 [=====] - 147s 314ms/step - loss: 0.2667 - accuracy: 0.9039 - val_loss: 0.2533 - val_accuracy: 0.9091
Epoch 5/10
469/469 [=====] - 147s 314ms/step - loss: 0.2417 - accuracy: 0.9119 - val_loss: 0.2429 - val_accuracy: 0.9144
Epoch 6/10
469/469 [=====] - 146s 311ms/step - loss: 0.2209 - accuracy: 0.9190 - val_loss: 0.2326 - val_accuracy: 0.9159
Epoch 7/10
469/469 [=====] - 146s 312ms/step - loss: 0.2045 - accuracy: 0.9237 - val_loss: 0.2341 - val_accuracy: 0.9178
Epoch 8/10
469/469 [=====] - 147s 313ms/step - loss: 0.1934 - accuracy: 0.9287 - val_loss: 0.2222 - val_accuracy: 0.9234
Epoch 9/10
```

### Actualización CNN

Se muestra el model summary. Este resumen proporciona información detallada sobre la arquitectura de la CNN, incluyendo el tipo de capa, el tamaño de salida de cada capa, el número de parámetros entrenables y no entrenables, etc.

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_12 (Conv2D)	(None, 26, 26, 32)	320
conv2d_13 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_6 (Dropout)	(None, 12, 12, 64)	0
flatten_5 (Flatten)	(None, 9216)	0
dense_10 (Dense)	(None, 128)	1179776
dropout_7 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 10)	1290
<hr/>		
Total params: 1199882 (4.58 MB)		
Trainable params: 1199882 (4.58 MB)		
Non-trainable params: 0 (0.00 Byte)		

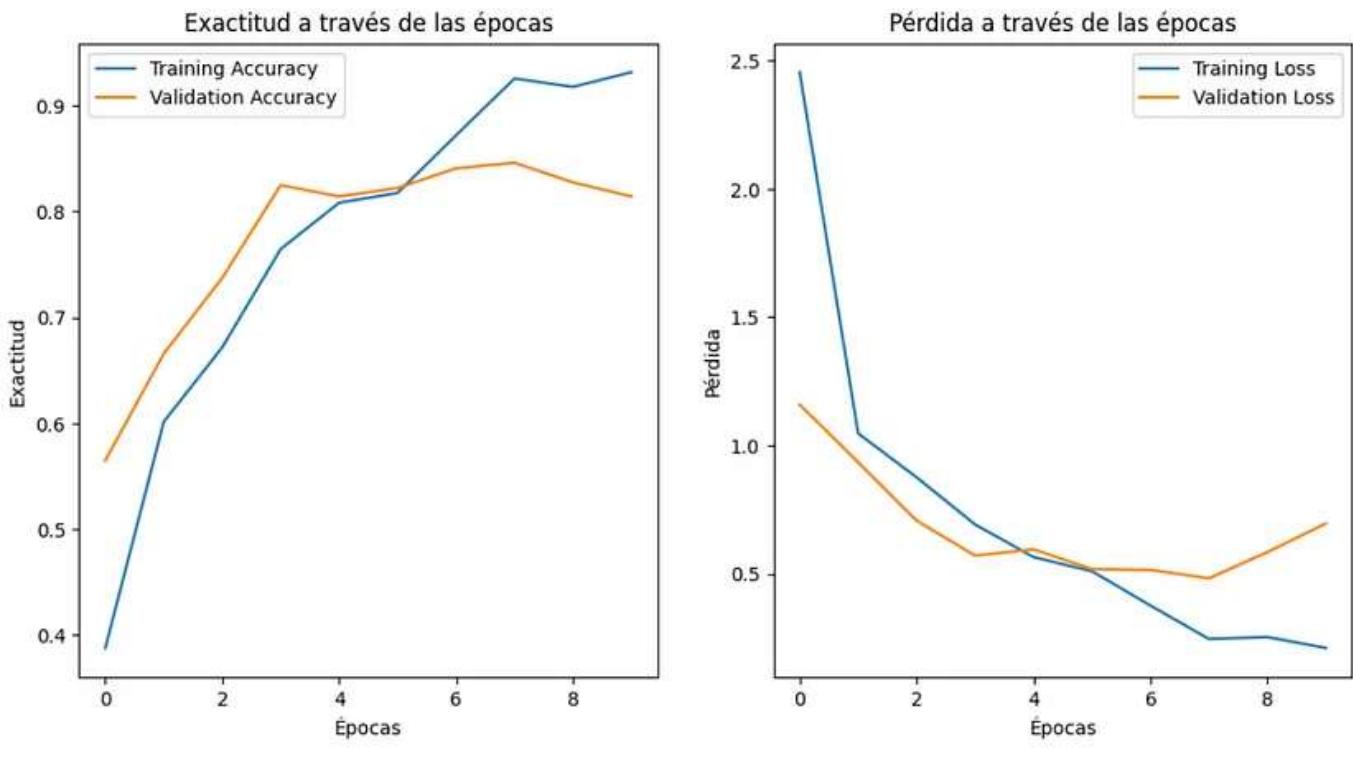
### Model Summary Fashion-MNIST

Al finalizar las épocas, nos muestra las métricas de precisión, recall, f1-score y support para cada clase, y en general.

	precision	recall	f1-score	support
T-shirt/top	0.84	0.92	0.88	1000
Trouser	0.99	0.98	0.99	1000
Pullover	0.90	0.86	0.88	1000
Dress	0.93	0.93	0.93	1000
Coat	0.84	0.93	0.89	1000
Sandal	0.99	0.99	0.99	1000
Shirt	0.84	0.70	0.76	1000
Sneaker	0.96	0.98	0.97	1000
Bag	0.98	0.99	0.99	1000
Ankle boot	0.98	0.96	0.97	1000
accuracy			0.93	10000
macro avg	0.93	0.93	0.92	10000
weighted avg	0.93	0.93	0.92	10000

Resultados CNN Fashion-MNIST

Finalmente se despliegan dos gráficas, una de exactitud y una de pérdida que nos indica como va evolucionando el modelo a través de las épocas



Gráficas Fashion-MNIST

## Fotografías satelitales

Las mismas gráficas se muestran para las 3 bases de datos. Primero observamos el avance de las métricas a través de las épocas

```

Epoch 1/10
48/48 [=====] - 287s 6s/step - loss: 2.5821 - accuracy: 0.4456 - val_loss: 0.8684 - val_accuracy: 0.7056
Epoch 2/10
48/48 [=====] - 294s 6s/step - loss: 0.9562 - accuracy: 0.6333 - val_loss: 0.8792 - val_accuracy: 0.7135
Epoch 3/10
48/48 [=====] - 278s 6s/step - loss: 0.7311 - accuracy: 0.7407 - val_loss: 0.7822 - val_accuracy: 0.6976
Epoch 4/10
48/48 [=====] - 276s 6s/step - loss: 0.6030 - accuracy: 0.7779 - val_loss: 0.6807 - val_accuracy: 0.7798
Epoch 5/10
48/48 [=====] - 279s 6s/step - loss: 0.4389 - accuracy: 0.8462 - val_loss: 0.6393 - val_accuracy: 0.7931
Epoch 6/10
48/48 [=====] - 283s 6s/step - loss: 0.3151 - accuracy: 0.9065 - val_loss: 0.5258 - val_accuracy: 0.8170
Epoch 7/10
48/48 [=====] - 276s 6s/step - loss: 0.2191 - accuracy: 0.9198 - val_loss: 0.5338 - val_accuracy: 0.8143
Epoch 8/10
48/48 [=====] - 276s 6s/step - loss: 0.2186 - accuracy: 0.9290 - val_loss: 0.5426 - val_accuracy: 0.8302
Epoch 9/10
48/48 [=====] - 279s 6s/step - loss: 0.1732 - accuracy: 0.9529 - val_loss: 0.5709 - val_accuracy: 0.8541
Epoch 10/10
48/48 [=====] - 276s 6s/step - loss: 0.1581 - accuracy: 0.9456 - val_loss: 0.6497 - val_accuracy: 0.8276

```

Actualización CNN

Posterior, podemos observar el model summary en donde nos indican los tipos de capas existentes en la CNN y los parámetros que entra a esa layer.

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_2 (Conv2D)	(None, 133, 238, 32)	896
conv2d_3 (Conv2D)	(None, 131, 236, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 65, 118, 64)	0
dropout_2 (Dropout)	(None, 65, 118, 64)	0
flatten_1 (Flatten)	(None, 490880)	0
dense_2 (Dense)	(None, 128)	62832768
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 6)	774
<hr/>		
Total params: 62852934 (239.76 MB)		
Trainable params: 62852934 (239.76 MB)		
Non-trainable params: 0 (0.00 Byte)		

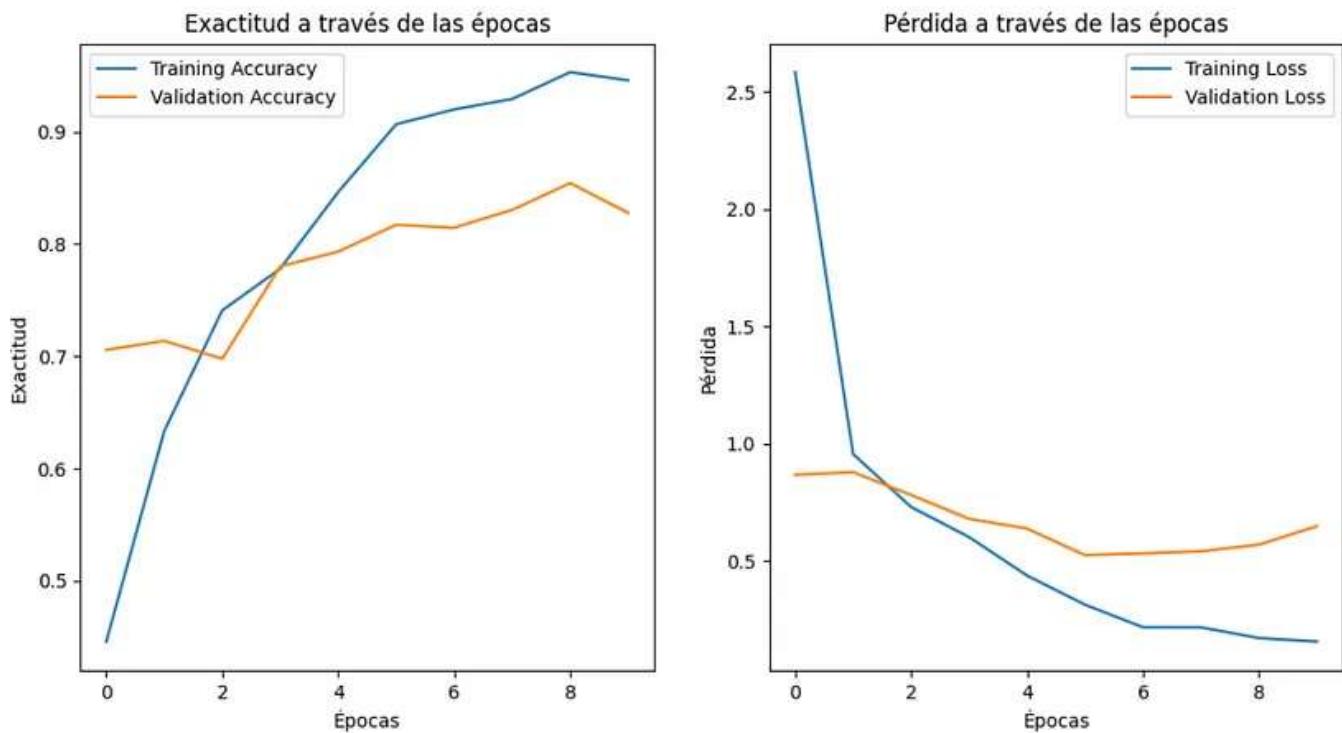
Model Summary Fotografías Satelitales

Obtenemos las métricas de precision, recall, f1-score y support para cada clase, y en general, observamos que en comparación a la base de datos anterior, los valores de las métricas fueron menores

	<b>precision</b>	<b>recall</b>	<b>f1-score</b>	<b>support</b>
Agua	0.90	0.79	0.84	66
Bosque	0.82	0.85	0.83	62
Ciudad	0.94	0.82	0.88	62
Cultivo	0.62	0.72	0.67	39
Desierto	0.81	1.00	0.90	74
Montaña	0.84	0.73	0.78	74
<b>accuracy</b>			0.83	377
<b>macro avg</b>	0.82	0.82	0.82	377
<b>weighted avg</b>	0.84	0.83	0.83	377

Resultados CNN Fotografías Satelitales

Se despliegan las gráficas de Exactitud vs Épocas y Pérdida vs Épocas



Gráficas Fotografías Satelitales

## Fotografías objetos propios

Se muestra la actualización de la CNN a través de las épocas, en este caso únicamente usamos 3 épocas ya que con estas llegamos a un accuracy de

0.99.

```

Epoch 1/3
63/63 [=====] - 363s 6s/step - loss: 2.4212 - accuracy: 0.6500 - val_loss: 0.0999 - val_accuracy: 0.9960
Epoch 2/3
63/63 [=====] - 355s 6s/step - loss: 0.1067 - accuracy: 0.9745 - val_loss: 0.0124 - val_accuracy: 0.9980
Epoch 3/3
63/63 [=====] - 357s 6s/step - loss: 0.0391 - accuracy: 0.9920 - val_loss: 0.0155 - val_accuracy: 0.9980

```

Actualización CNN

Se despliega el model summary donde observamos las capas de la red neuronal y los parámetros de entrada en cada una

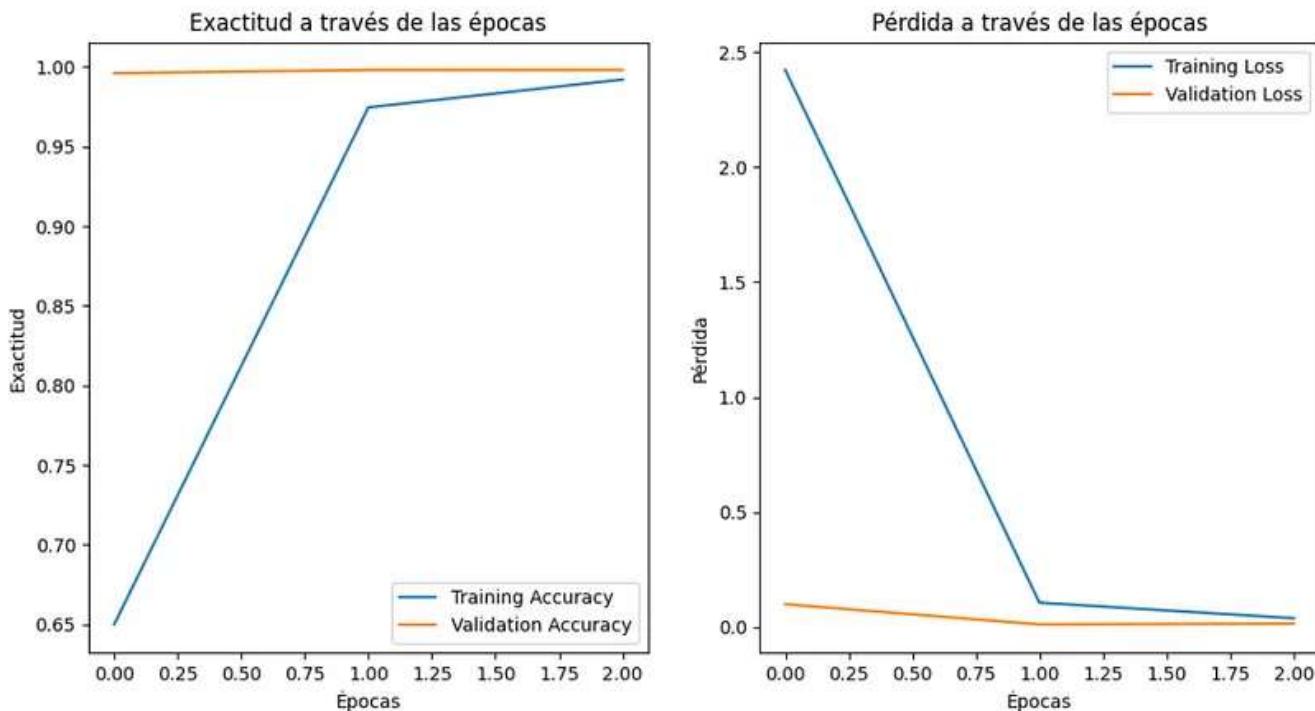
Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 133, 238, 32)	896
conv2d_1 (Conv2D)	(None, 131, 236, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 65, 118, 64)	0
dropout (Dropout)	(None, 65, 118, 64)	0
flatten (Flatten)	(None, 490880)	0
dense (Dense)	(None, 128)	62832768
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 5)	645
<hr/>		
Total params:	62852805 (239.76 MB)	
Trainable params:	62852805 (239.76 MB)	
Non-trainable params:	0 (0.00 Byte)	

Model Summary Fotografías Objetos

Observamos que las métricas finales de cada clase son bastante buenas, casi perfectas, y la general también

	precision	recall	f1-score	support
Celular	1.00	0.99	0.99	100
Cachuchas	1.00	1.00	1.00	88
Lapiz	1.00	1.00	1.00	108
Libro	1.00	1.00	1.00	111
Tenis	0.99	1.00	0.99	93
accuracy			1.00	500
macro avg	1.00	1.00	1.00	500
weighted avg	1.00	1.00	1.00	500

Resultados CNN Fotografías Propias



Gráficas Fotografías Propias

- Discusión sobre la eficacia de los modelos para predecir con los datos utilizados.

En general, la CNN que adaptamos a nuestras 3 bases de datos funcionaron de manera adecuada, para unas bases el rendimiento fue mejor, a continuación se explican los resultados obtenidos en cada una.

## Fashion MNIST

El rendimiento de la CNN en Fashion-MNIST fue destacado, con un accuracy general superior al 0.93 en el conjunto de entrenamiento y superior al 0.92 en el conjunto de prueba. Hablando de las métricas por clase, algunas como playeras y camisas, presentaron una precisión ligeramente, esto puede deberse a las similitudes visuales, sin embargo, la mayoría de las clases obtuvieron una precisión superior al 0.98. Con un accuracy global del 0.93, queda claro que esta arquitectura de red es ideal para clasificar imágenes de prendas en este conjunto de datos extenso, con más de 70,000 fotografías de baja calidad en el conjunto de prueba.

## Fotografías Satelitales

Para las fotografías satelitales obtuvimos un buen resultado, no tan bueno como en las dos otras bases, pero observamos que en la época 10 se tenía un accuracy de 0.94 en el set de entrenamiento y 0.82 en el set de validación. Vemos un accuracy global de 0.83, lo que significa que no es el mejor, pero tampoco es malo, caería en el rango de un modelo aceptable, esto puede ser por la combinación de poca cantidad de fotografías e imágenes complejas, también es muy probable que con una mayor cantidad de épocas las métricas aumenten. Analizando las clases individuales, vemos que ciudad y agua son las categorías que se distinguen con mayor facilidad en el proceso de clasificación, lo cual se refleja en una precisión superior al 0.9 para estas clases. En contraste, la categoría de cultivo presenta una precisión más baja, ya que puede ser confundida con imágenes de desierto o bosque.

## Fotografías Objetos Propios

Finalmente tenemos la base de datos que muestra resultados casi perfectos, solo con tres épocas logró tener una precisión de 1 en 4/5 clases y en la última una precisión de 0.99, también la precisión general fue de 1. Esto tiene sentido ya que las fotografías son menor complejas, las diferencias entre los objetos es notable, se encuentran en un fondo blanco y se localizan bien centradas, esto hace que al utilizar una CNN en este tipo de base de datos, nos arroje resultados increíbles.

- **Conclusiones**

La aplicación de redes neuronales convolucionales (CNN) en la clasificación de imágenes provenientes de tres conjuntos diversos ha revelado su excepcional capacidad para adaptarse a patrones complejos y aprender representaciones jerárquicas. Este trabajo resalta la versatilidad de las CNN al demostrar un rendimiento sobresaliente en la clasificación de imágenes con características y áreas diversas. Esta eficacia subraya su importancia como herramientas fundamentales en la visión por computadora, abriendo posibilidades para aplicaciones innovadoras en la clasificación de imágenes en diversos contextos.

En comparación con modelos clásicos y otras técnicas implementadas, las CNN destacan como los clasificadores ideales, especialmente cuando se trabaja con bases de datos más grandes y complejas. Su continua innovación y desarrollo las posicionan como tecnología clave en el futuro de la Inteligencia Artificial, sugiriendo que seguirán siendo un tema relevante y digno de explorar en la medida en que la tecnología avance.

- **Referencias**

Kalita, D. (2023). Basics of CNN in Deep Learning. Analytics Vidhya.  
Recuperado de <https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/>

Kadam, R. (2021). Kernel Filters in convolutional neural network CNN.  
Recuperado de <https://medium.com/codex/kernels-filters-in-convolutional-neural-network-cnn-lets-talk-about-them-ee4e94f3319>

Max Pooling and Average Pooling. (s.f.). Recuperado de  
[https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max\\_fig2\\_333593451](https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451)

Ponce, J. (s.f.). Jahaziel Ponce. Recuperado de  
<https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow/>

Introducción a las redes neuronales. (2020). BlogSpot. Recuperado de  
<https://analisisyprogramacionoop.blogspot.com/2020/05/introduccion-redes-neuronales.html>

Machine Learning