

Clasificación de imágenes con redes profundas

Bienvenidos a mi último blog, donde exploraremos la practicidad de las CNNs en la clasificación de imágenes, centrándonos en su aplicación en conjuntos de datos diversos como Fashion-MNIST, fotografías satelitales y un conjunto de imágenes de objetos creado internamente. Además, examinaremos la importancia crucial de TensorFlow en este panorama, destacando cómo esta biblioteca no solo simplifica el desarrollo de modelos complejos, sino que también facilita la escalabilidad para la producción en gran escala.

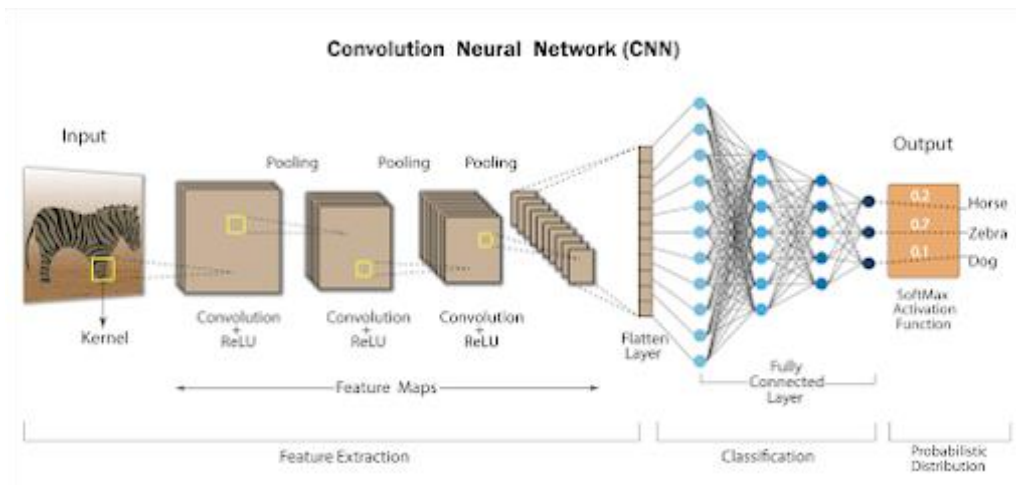


Imagen recuperada

de <https://citt.itsm.edu.mx/ingeniantes/articulos/ingeniantes10no1vol1/10.-%20Comparativa%20de%20algoritmos%20de%20Inteligencia%20artificial%20para%20la%20detecci%C3%B3n%20de%20personas%20a%20trav%C3%A9s%20de%20una%20c%C3%A1mara%20instalada%20en%20el%20interior%20de%20un%20veh%C3%ADculo.pdf>

Las Redes Neuronales Convolucionales (CNNs) han emergido como una herramienta fundamental en el ámbito de la visión por computadora, revolucionando la capacidad de las máquinas para entender y clasificar imágenes de manera efectiva. Este avance ha sido especialmente prominente en campos como reconocimiento facial, diagnóstico médico y análisis de imágenes satelitales, donde la capacidad de interpretar y extraer patrones visuales es crucial.

En el corazón de esta revolución se encuentra la práctica aplicación de CNNs, una clase especializada de redes neuronales diseñada específicamente para procesar datos visuales. Su estructura imita la organización jerárquica del sistema visual biológico, permitiéndoles identificar características desde lo más básico hasta lo más complejo. Esta capacidad inherente de reconocimiento jerárquico ha demostrado ser excepcionalmente eficiente en tareas de clasificación de imágenes.

Uno de los pilares que ha permitido la proliferación y accesibilidad de estas poderosas herramientas es TensorFlow, una biblioteca de código abierto que ha llevado la computación numérica y el aprendizaje automático a nuevas alturas. TensorFlow proporciona una infraestructura robusta y eficiente para la implementación de modelos de deep learning, y se ha convertido en una opción predilecta para el desarrollo y despliegue de CNNs.

- **¿Qué se hizo?**

En este trabajo se utilizarán redes neuronales convolucionales (CNN) para clasificación de imágenes, utilizando tres diferentes conjuntos de datos de imágenes:

1. Fashion-MNIST: Esta es una base de datos de imágenes de la revistad Zalando que consiste en un set de entrenamiento de 60,000 ejemplos y un set de prueba de 10,000 ejemplos. Cada imagen es de 28 pixeles de altura por 28 pixeles de anchura, dando un total de 784 pixeles por imagen. Cada pixel tiene un único valor asociado, indicando la iluminación u oscuridad del pixel, donde un número alto indica un pixel más oscuro.

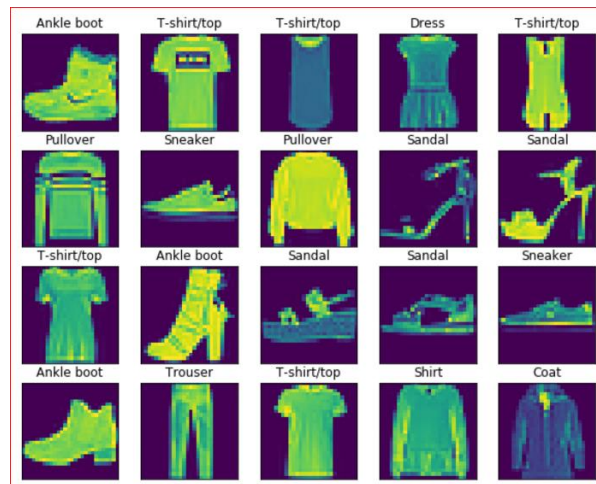


Imagen recuperada
de <https://medium.datadriveninvestor.com/implementing-convolutional-neural-network-using-tensorflow-for-fashion-mnist-caa99e423371>

1. Fotografías satelitales: Esta base de datos tiene alrededor de 2000 fotografías satelitales de ambientes de México. Cada imagen está asociada a una de las siguientes clases: agua, bosque, ciudad, cultivo, desierto y montaña.



2. Imágenes de objetos: Nosotros, como equipo, creamos un conjunto de datos de imágenes de unas llaves, un celular, un audífono, un objeto antiestrés y una cartera. Todas estas fotos son de los mismos objetos con diferentes ángulos y niveles de iluminación para darle variedad computacional.



En el código base que se nos fue proporcionado para lograr obtener las características principales de nuestras imágenes para los 3 diferentes base de datos, primero, cargamos y redimensionamos las imágenes. Luego, se calculan los histogramas de color para los canales rojo, verde y azul, generando un descriptor de histograma de color normalizado. Posteriormente, se convierte la imagen a escala de grises y se calcula la matriz de co-ocurrencia de nivel de gris (GLCM) con ciertos parámetros. A partir de la GLCM, se extraen descriptores de textura, como disimilitud, homogeneidad, energía y correlación. Estos descriptores ofrecen información valiosa sobre las características visuales y texturales de cada una de nuestras imágenes, siendo útiles para nuestro modelo de clasificación de imágenes. El código se muestra a continuación:

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from skimage import io
```

```
from skimage.transform import resize
```

```
from skimage.feature import graycomatrix, graycoprops
```

```
from skimage.color import rgb2gray
```

```
from skimage import img_as_ubyte
```

```
#-----
```

```
# Load image
```

```
#-----
```

```
scale = 8
```

```
img_width = int(1920/scale)
```

```
img_height = int(1080/scale)
```

```
rgb = io.imread('img.jpg')
```

```
rgb_resized = resize(rgb, (img_height, img_width), anti_aliasing=True)
```

```
#-----
```

```
# Color histograms
```

```
#-----
```

```
nbins = 16
```

```
rh = np.histogram(rgb_resized[:, :, 0].flatten(), nbins, density = True)
```

```
gh = np.histogram(rgb_resized[:, :, 1].flatten(), nbins, density = True)
```

```
bh = np.histogram(rgb_resized[:, :, 2].flatten(), nbins, density = True)
```

```
hist_descriptor = np.concatenate((rh[0], gh[0], bh[0]))
```

```
#-----
```

```
# Texture descriptors
```

```
#-----  
  
gray_resized = img_as_ubyte(rgb2gray(rgb_resized))  
  
glcm = graycomatrix(gray_resized, distances=[5], angles=[0, np.pi/4, np.pi/2, 3*np.pi/4])  
  
  
texture_desc = [graycoprops(glcm, 'dissimilarity')[0, 0], graycoprops(glcm, 'homogeneity')[0, 0],  
graycoprops(glcm, 'energy')[0, 0], graycoprops(glcm, 'correlation')[0, 0]]
```

- ¿Dónde se hizo?

JupyterLab es el último entorno de desarrollo interactivo basado en web para cuadernos, código y datos. Su interfaz flexible permite a los usuarios configurar y organizar flujos de trabajo en ciencia de datos, computación científica, periodismo computacional y aprendizaje automático. Un diseño modular invita a extensiones para ampliar y enriquecer la funcionalidad.



Imagen recuperada de <https://jupyter.org/try>

- CNNs

Las redes neuronales son modelos matemáticos que tratan de emular el comportamiento natural de las redes neuronales biológicas.

Se establece una red de unidades lógicas, llamadas neuronas, interconectadas entre sí que procesan la información recibida y emiten un resultado a la siguiente capa determinado por una función de activación que tiene en cuenta el peso de cada entrada, dotando así de mayor importancia a conexiones entrantes concretas.

Durante la fase de entrenamiento de una red neuronal los parámetros de peso y bias son reajustados con el fin de adaptar el modelo a una tarea concreta y mejorar las predicciones. La función e activación será seleccionada de acuerdo al problema que se pretenda resolver.

Las redes neuronales convolucionales (CNNs) son una clase de redes neuronales multicapa feedforward especialmente diseñadas para el reconocimiento y clasificación de imágenes.

Los ordenadores perciben las imágenes como un vector bidimensional con los valores relativos a los píxeles. Contando con un canal para imágenes en escala de grises o tres para el color (RGB).

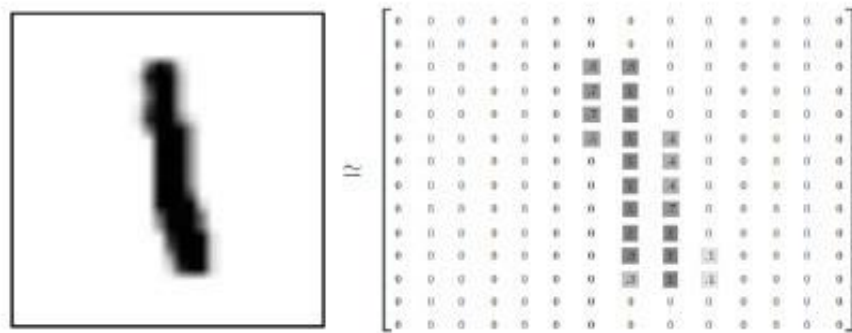
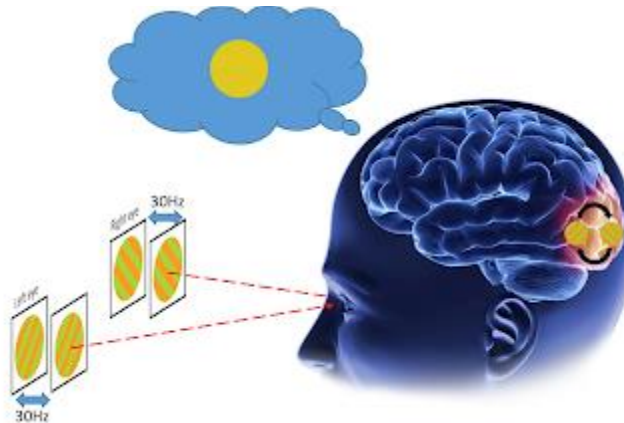


Imagen recuperada de https://oa.upm.es/53050/1/TFG_JAVIER_MARTINEZ_LLAMAS.pdf

La CNN procesa sus capas imitando el córtex visual del cerebro humano para identificar distintas características en las entradas. Para ello, contiene varias capas ocultas especializadas y con una jerarquía: esto significa que las primeras capas detectan propiedades o formas básicas y se van especializando hasta llegar a capas más profundas capaces de reconocer formas complejas como un rostro o una silueta.



La red neuronal por sí misma ha de reconocer una gran cantidad de imágenes para que la pueda captar las características únicas de cada objeto y a su vez poder generalizarlo. Cada imagen se trata de una matriz de píxeles cuyo valor va de 0 a 255 pero se normaliza para la red neuronal de 0 a 1.

La primera capa del esquema de una red convolucional se trata de la entrada a la red. Estas entradas de la red serán imágenes de la base de datos pertinente.

La siguiente capa importante dentro de la CNN es la capa convolucional, en cuya capa se restringe el número de conexiones posibles entre las neuronas de la capa oculta y los elementos de la imagen de la entrada. De esta manera, cada neurona oculta solo estará conectada con un pequeño subconjunto de elementos de la imagen total. Esta idea se inspiró en cómo funciona el sistema visual biológico, debido a que las neuronas que trabajan en el córtex visual poseen una serie de campos receptivos que responden solo ante unos estímulos localizados en una región o área específica.

La convolución es una operación de productos y sumas entre la imagen de entrada y un filtro que genera un mapa de características. La ventaja es que el mismo filtro sirve para extraer el mismo rasgo en cualquier parte de la imagen, atendiendo al carácter estacionario de las imágenes. Esto permite reducir el número de conexiones y el número de parámetros a entrenar en comparación con una red multicapa completamente conectada.

Se puede resumir el comportamiento de la capa convolucional de la siguiente manera:

1. Cuando llega la imagen de entrada a la red, se superponen el filtro y ésta y se calcula la convolución de dos dimensiones entre los respectivos elementos de la imagen y el kernel. Una vez se obtiene el resultado de dicha operación, se almacena en una posición de la matriz de activación.
2. A continuación, se desplaza o desliza el filtro una posición a la derecha sobre la imagen y se vuelve a calcular la convolución, almacenando el resultado en la siguiente posición de la matriz de activación. De esta manera, este proceso se repite a lo largo de toda la imagen desplazándose de izquierda a derecha y bajando una unidad al llegar a un borde. Una vez recorrida toda la imagen se obtiene la matriz de activación completa que contiene las características que se buscan en la imagen para cada filtro.

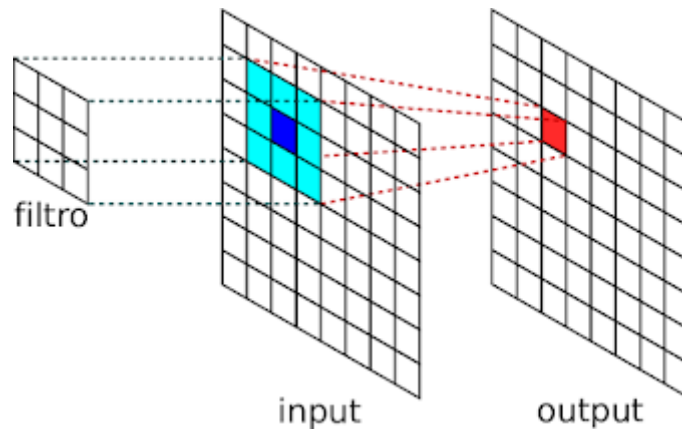


Imagen recuperada de https://www.researchgate.net/figure/Figura-42-Ejemplo-de-un-nucleo-de-convolucion_fig5_336899385

Inmediatamente después de la capa de convolución se encuentra la capa de pooling. Después de haber obtenido las características en la capa de convolución, el siguiente paso es usarlas para la clasificación de las imágenes. Las imágenes tienen la propiedad de ser estacionarias y en la capa convolucional se ha creado un mapa de características. Así, el paso natural es ver en qué zona de la imagen son esos rasgos predominantes. Para realizar dicho paso, es posible calcular la media o buscar el máximo valor de una característica a lo largo de una región de la imagen. La matriz resultante de la operación anterior resulta de unas dimensiones considerablemente menores que la matriz de características, obtenida en la capa de convolución.

Entonces, el objetivo de la capa de pooling es disminuir aún más la carga computacional del sistema y, al mismo tiempo, ayudar con la caracterización de la imagen obteniendo y localizando los rasgos predominantes en ella.

Existen dos tipos de pooling:

- Average-Pooling: Se toman todos los elementos de la submatriz, se calcula la su media y el resultado se guarda en la primera posición de la matriz de salida. Después, se selecciona la siguiente submatriz y se realiza la misma operación. Una vez se llega al borde derecho de la imagen, se vuelve a la izquierda de la imagen, se da un salto hacia abajo y se vuelve a recorrer la imagen de izquierda a derecha.
- Max-Pooling: Se busca el elemento de mayor valor que se encuentre en la submatriz y se guarda en la primera posición de la matriz de salida. Después, se selecciona la siguiente submatriz y se realiza la misma operación. Una vez se llega al borde derecho de la imagen, se vuelve a la izquierda de la imagen, se da un salto hacia abajo y se vuelve a recorrer la imagen de izquierda a derecha.

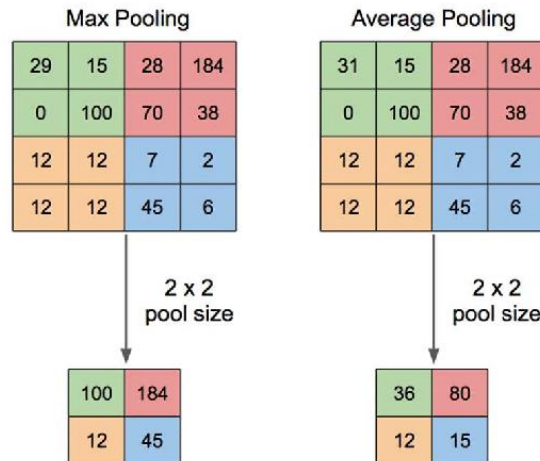


Imagen recuperada de https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451

La última capa del esquema de las CNN se trata de un clasificador que determina a que clase pertenece la imagen de entrada, es decir, indica que categoría "cree" la red que se le ha proporcionado a la entrada.

La capa totalmente conectada se encuentra justo a continuación de la última capa de pooling y esta compuesta por un número de neuronas igual al número de clases. El esquema que sigue esta capa es que cada una de las neuronas está a su vez conectada con todos y cada uno de los elementos de las matrices de la capa inmediatamente anterior.

A la salida de esta capa se encontrará un vector de número de componentes igual al número de clases. Cada uno de estos componentes representa la probabilidad que tiene la imagen de entrada de pertenecer a una determinada clase. Por último, la red determinará cuál de estos elementos del vector es mayor y lo indica.

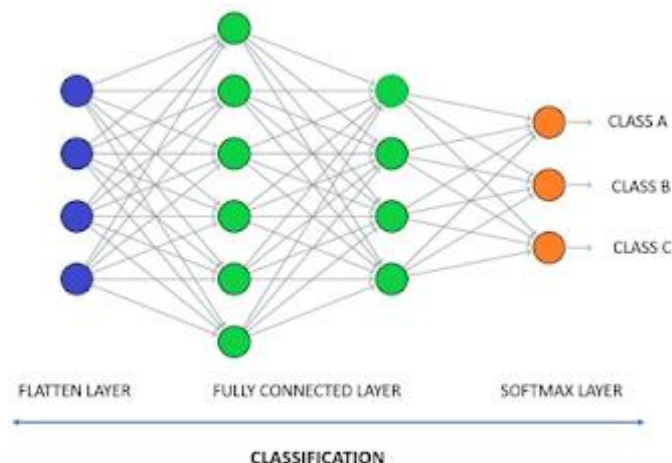


Imagen recuperada de <https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/>

- **¿Cómo se aplicó la CNN?**

TensorFlow es una biblioteca de código abierto para la computación numérica y Machine Learning a gran escala. TensorFlow reúne una serie de modelos y algoritmos de Machine Learning y Deep Learning y los hace útiles mediante una metáfora común.

Utiliza Python para proporcionar una práctica API para crear aplicaciones con el marco de trabajo, a la vez que ejecuta esas aplicaciones en C++ de alto rendimiento.

TensorFlow puede entrenar y ejecutar redes neuronales profundas para la clasificación de dígitos escritos a mano, el reconocimiento de imágenes, la incrustación de palabras, las redes neuronales recurrentes, los modelos secuencia a secuencia para la traducción automática, el procesamiento del lenguaje natural y las simulaciones basadas en ecuaciones diferenciales parciales. Lo mejor de todo es que TensorFlow admite predicción de producción a escala, con los mismos modelos utilizados para el entrenamiento.



Imagen recuperada de <https://es.wikipedia.org/wiki/TensorFlow>

Keras es una API de redes neuronales escrita en lenguaje Python. Se trata de una biblioteca de código abierto que se ejecuta sobre frameworks como Theano y TensorFlow. Está diseñada para ser intuitiva, modular, fácil de ampliar y para trabajar con Python.

Es posible combinar módulos de capas neuronales, optimizadores, esquemas de inicialización, funciones de activación o esquemas de regularización para crear nuevos módulos.

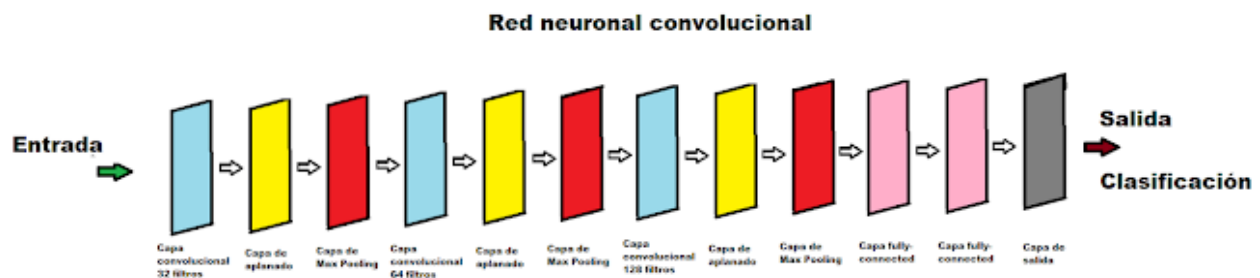


Imagen recuperada de <https://keras.io/>

- **Modelo de CNN utilizado**

Descripción de la CNN:

1. Capa convolucional de 32 filtros de 3x3 con activación ReLu
2. Capa de aplanado
3. Capa de MaxPooling con una ventana de 2x2
4. Segunda capa convolucional con 64 filtros de 3x3 con activación ReLu
5. Capa de aplanado
6. Capa de MaxPooling con ventana de 2x2
7. Tercera capa convolucional de 128 filtros de 3x3 con activación ReLu
8. Capa de aplanado
9. Capa de MaxPooling con ventana de 2x2
10. Capa de aplanado
11. Capa completamente conectada con 128 neuronas y activación ReLu
12. Capa completamente conectada con número de neuronas igual a la cantidad de etiquetas en los datos y activación SoftMax
13. Capa de salida con activación



- Función de pérdida: Sparse_Categorical_Crossentropy

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Imagen recuperada de <https://fmorenovr.medium.com/sparse-categorical-cross-entropy-vs-categorical-cross-entropy-ea01d0392d28>

- Métrica de evaluación: Accuracy
- Épocas de entrenamiento: 10

- **¿Cómo se pueden evaluar los resultados?**

Para evaluar el rendimiento de modelos de clasificación, se utilizan diversas métricas que proporcionan información sobre la calidad de las predicciones. Para este caso se utilizaron las siguientes métricas:

- **Exactitud (Accuracy):**

- Es la proporción de predicciones correctas respecto al total de predicciones.

$$Accuracy = \frac{\# \text{ of correct predictions}}{\text{total \# of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Imagen recuperada de <https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python>

- **Recall:**

- Mide la proporción de instancias positivas correctamente clasificadas respecto al total de instancias clasificadas como positivas.

$$Recall = \frac{TP}{TP + FN}$$

Imagen recuperada de <https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python>

- **Cross Validation (Validación Cruzada):**

- Es una técnica utilizada en el campo del aprendizaje automático para evaluar el rendimiento de un modelo y reducir el riesgo de sobreajuste. Su propósito es obtener una estimación más confiable del rendimiento del modelo al entrenar y evaluar el modelo con múltiples subconjuntos de datos. Este método es especialmente útil cuando el tamaño del conjunto de datos es limitado y se quiere maximizar el uso de los datos disponibles para el entrenamiento y la evaluación.

- **F1-Score:**

- El puntaje F1 es una medida de la precisión de una prueba, es la media armónica de precisión y recuperación. Puede tener una puntuación máxima de 1 (precisión y recuerdo perfectos) y una

mínima de 0. En general, es una medida de la precisión y robustez de su modelo.

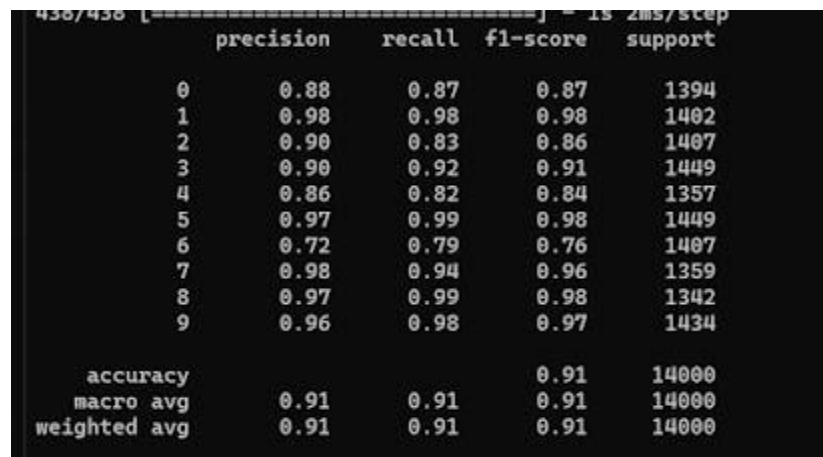
$$F1\ score = \frac{2 * (precision * recall)}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

Imagen recuperada de <https://www.datasource.ai/es/data-science-articles/compreension-de-la-matriz-de-confusion-y-como-implementarla-en-python>

- **¿Qué resultados se obtuvieron?**

Los resultados obtenidos con la CNN para clasificación son los siguientes:

Con la base de datos Fashion-MNIST:



A terminal window showing the output of a classification task. The output is a table with 5 columns: class index, precision, recall, f1-score, and support. The rows represent classes 0 through 9, followed by summary statistics: accuracy, macro avg, and weighted avg. The precision, recall, and f1-score values are all high, indicating good performance. The support values represent the number of samples in each class.

	precision	recall	f1-score	support
0	0.88	0.87	0.87	1394
1	0.98	0.98	0.98	1402
2	0.90	0.83	0.86	1407
3	0.90	0.92	0.91	1449
4	0.86	0.82	0.84	1357
5	0.97	0.99	0.98	1449
6	0.72	0.79	0.76	1407
7	0.98	0.94	0.96	1359
8	0.97	0.99	0.98	1342
9	0.96	0.98	0.97	1434
accuracy			0.91	14000
macro avg	0.91	0.91	0.91	14000
weighted avg	0.91	0.91	0.91	14000

Como podemos observar, la red neuronal logró clasificar con un 91% de exactitud el conjunto de prueba de la base de datos Fashion-MNIST. Esta exactitud nos indica que la elección de parámetros de la red favorecieron el correcto reconocimiento de factores distintivos en las imágenes para la CNN.

El recall por clase nos lanza información adicional sobre cuales son las clases con mayor número de imágenes correctamente clasificadas, de donde podemos notar que la clase más reconocida fue la número 8 con un 99% de aciertos.

Esta correcta y casi perfecta clasificación de las diferentes clases de la base de datos puede deberse a que las clases tienen características muy únicas entre sí, por lo que es fácilmente identificable para la red neuronal el encontrar los factores decisivos para la clasificación de las imágenes de Fashion-MNIST.

Con la base de datos de fotografías satelitales:

```

13/13 [=====] - 1s 59ms/step

```

	precision	recall	f1-score	support
Agua	0.43	0.18	0.26	87
Bosque	0.43	0.87	0.57	69
Ciudad	0.54	0.54	0.54	56
Cultivo	0.43	0.32	0.36	57
Desierto	0.91	0.83	0.87	72
Montaña	0.35	0.35	0.35	63
accuracy			0.51	404
macro avg	0.51	0.51	0.49	404
weighted avg	0.52	0.51	0.49	404

```

Press any key to continue . . . |

```

En esta base de datos la CNN tuvo dificultades para llevar a cabo buenas clasificaciones y alcanzó una exactitud no tan buena del 51%, es decir, solo 1 de 2 imágenes es correctamente clasificada.

Observando el recall, el único bioma que la CNN pudo reconocer y clasificar un porcentaje de decente de sus imágenes fue el bioma de desierto con un el 83% de las imágenes de desierto clasificadas correctamente. Y la peor distinción de características para la CNN se lo lleva el bioma de agua, con un pésimo recall del 18% de las imágenes del bioma correctamente clasificadas.

Este bajo rendimiento de la CNN con la base de datos de biomas podría atribuirse a que las imágenes satelitales cuentan con una resolución y escala variada, pueden llegar a tener incluso interferencia atmosférica, así como diversas características hidrológicas que harían confundir a nuestra pobre, y no tan especializada, red neuronal convolucional.

Con la base de datos de objetos:

```

Epoch 9/10
63/63 [=====] - 182s 2s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 10/10
63/63 [=====] - 111s 2s/step - loss: 9.0047e-04 - accuracy: 1.0000 - val_loss: 0.0012 - val_accuracy: 1.00
16/16 [=====] - 8s 487ms/step - loss: 0.0012 - accuracy: 1.0000
Accuracy en el conjunto de prueba: 1.0
16/16 [=====] - 8s 490ms/step

```

	precision	recall	f1-score	support
...				
accuracy			1.00	500
macro avg	1.00	1.00	1.00	500
weighted avg	1.00	1.00	1.00	500

En este último resultado de la misma CNN con nuestra propia base datos, observamos que el rendimiento fue perfecto, pues contamos con una exactitud del 100% de clasificaciones correctas con el conjunto de prueba.

Este comportamiento super fitness se debe, posiblemente, a las características de las imágenes de la base de datos, tales como la el que los objetos tengan formas muy diferentes entre sí, y que cada uno tiene un color distintivo bastante fácil de notar.

Estos resultados resaltan la importancia de considerar la idoneidad del modelo CNN para el tipo específico de datos con los que se trabaja. Mientras que en algunos casos puede lograr altas tasas de precisión, en otros puede enfrentar dificultades debido a la complejidad inherente de las imágenes. La selección cuidadosa de parámetros y la comprensión profunda de la naturaleza de los datos son fundamentales para maximizar la eficacia de una CNN en tareas de clasificación.

- **Conclusión**

Este análisis subraya la relevancia de evaluar la idoneidad del modelo CNN para cada conjunto de datos específico. Aunque estas redes pueden alcanzar altas tasas de precisión en ciertos escenarios, su rendimiento puede variar significativamente según la complejidad y características únicas de las imágenes. La comprensión profunda de los datos y la selección precisa de parámetros son esenciales para maximizar la eficacia de una CNN en tareas de clasificación.

En última instancia, este viaje a través de la aplicación de CNNs respalda la idea de que, si bien estas redes neuronales han demostrado ser herramientas poderosas, su éxito depende en gran medida de la adaptabilidad y comprensión del contexto específico de cada conjunto de datos. Con TensorFlow como aliado, la exploración y mejora continua de estos modelos continúa siendo una prometedora frontera en la evolución de la visión por computadora y el aprendizaje profundo.

Referencias:

1. **Fashion MNIST.** (2017, 7 diciembre). Kaggle. <https://www.kaggle.com/datasets/zalando-research/fashionmnist>
2. **Artola Moreno, Á.** (2019). Clasificación de imágenes usando redes neuronales convolucionales en Python.
3. **Durán Suárez, J.** (2017). Redes neuronales convolucionales en R: Reconocimiento de caracteres escritos a mano.
4. **Martínez Llamas, J.** (2018). Reconocimiento de imágenes mediante redes neuronales convolucionales.
5. **González, L.** (2022, 7 septiembre). ¿Qué es TensorFlow? ¿Cómo funciona? 🤖 Aprende IA. <https://aprendeia.com/que-es-tensorflow-como-funciona/>
6. **Alexandre.** (2023, 30 octubre). Keras: Todo sobre la API de deep learning. Formation Data Science | DataScientest.com. <https://datascientest.com/es/keras-la-api-de-deep-learning>

7. **Validación cruzada - Amazon Machine Learning. (s. f.).**
https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/cross-validation.html
8. **Project Jupyter. (s. f.). Home.** <https://jupyter.org/>
9. **Comprensión de la matriz de confusión y cómo implementarla en Python. (2023, 1 septiembre). DataSource.ai.** <https://www.datasource.ai/es/data-science-articles/compression-de-la-matriz-de-confusion-y-como-implementarla-en-python>