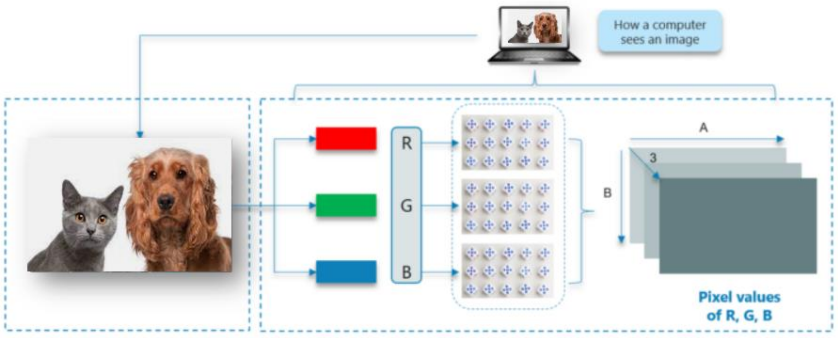
1. Introducción

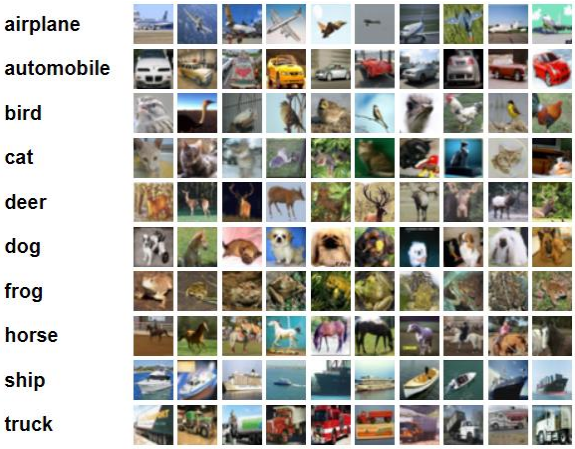
El objetivo de este proyecto consiste en el desarrollo de un modelo en base a un algoritmo de aprendizaje automático, basado en la clasificación y reconocimiento de imágenes.



Las computadoras procesan imágenes dividiendo estas en 3 canales de color que son rojo, verde y azul. Cada uno de estos canales de color se asigna al píxel de la imagen.Luego, la computadora reconoce el valor asociado con cada píxel y determina el tamaño de la imagen.
Sin embargo, para las imágenes en blanco y negro, solo hay un canal y el concepto es el mismo.

2. Dataset para entrenar el modelo de ML

El dataset CIFAR-10 (Instituto Canadiense de Investigación Avanzada, 10 clases) es un subconjunto del conjunto de datos Tiny Images y consta de 60000 imágenes en color de 32x32. Las imágenes están etiquetadas con una de las 10 clases mutuamente excluyentes: avión, automóvil (pero no camioneta), pájaro, gato, venado, perro, rana, caballo, barco y camioneta. Hay 6000 imágenes por clase con 5000 imágenes de entrenamiento y 1000 imágenes de test.



Target: Nuestros datos no son categóricos por naturaleza nuestros objetivos son números enteros en el rango [0,9] que representan las clases.

Procesamiento del dataset: Convertimos la data en el formato float32 , esto para de alguna manera acelerar el training. Luego normalizamos la data , n el rango [−1,1]

```
# Parse numbers as floats
train_x=train_X.astype('float32')
test_X=test_X.astype('float32')
# Normalize data
train_X=train_X/255.0
test_X=test_X/255.0
```

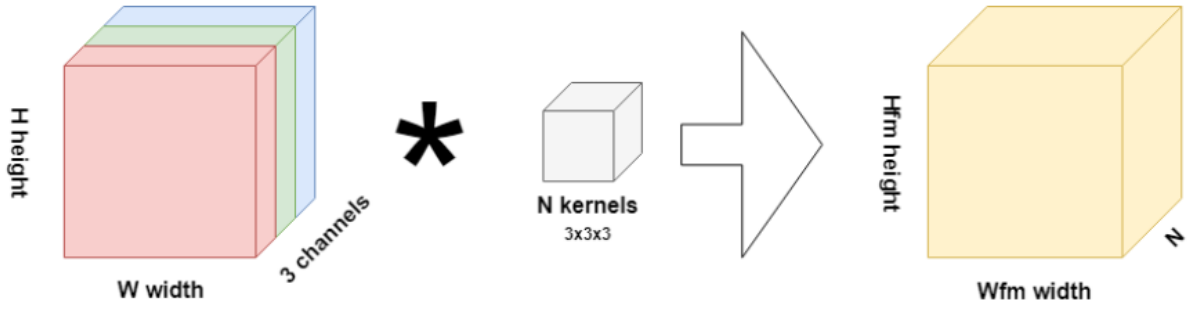
```
train_X = train_X.reshape(50000,32*32*3)
test_X = test_X.reshape(10000,32*32*3)
```

```
(50000, 3072)
(10000, 3072)
```

3. Modelos ML para entrenamiento

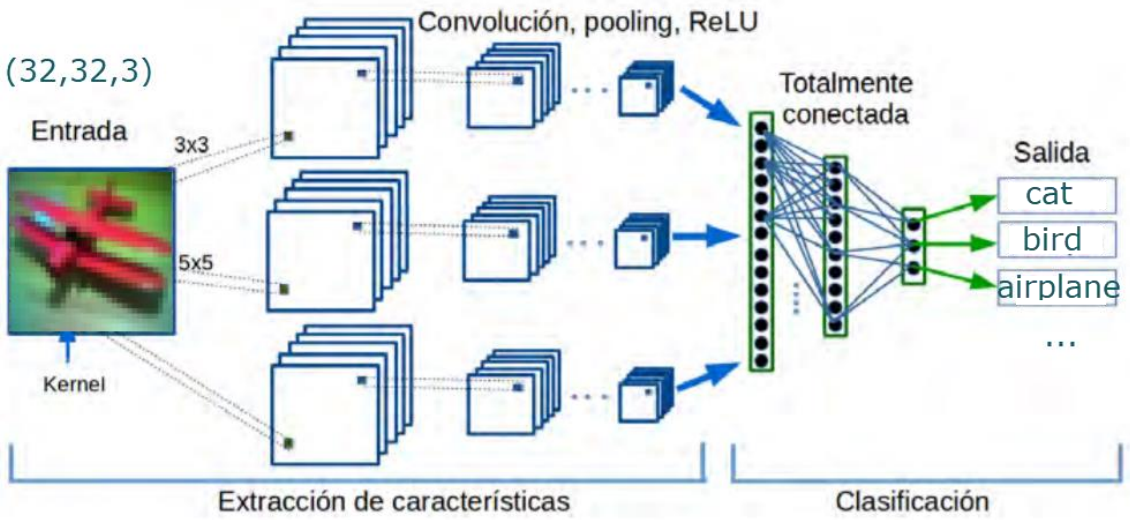
3.1. Redes Neuronales Convolucionales

Las redes neuronales convolucionales son una ampliación de las redes neuronales estándares.



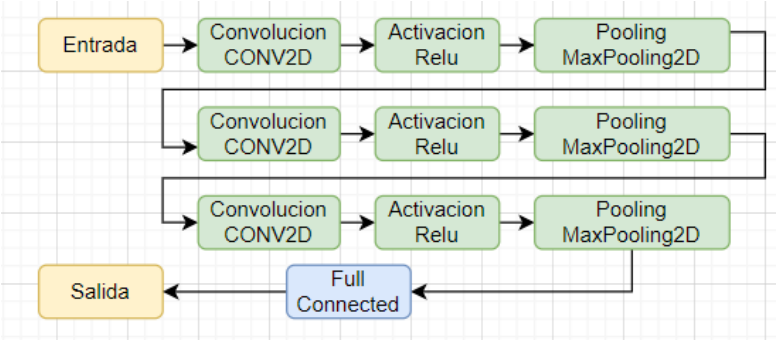
La ampliación consiste en añadir más capas al principio de la red, que se encargarán de manipular los píxeles de las imágenes. En dicha ampliación, existen dos tipos de capas:

- capas de convolución (convolutional layers).
- capas de pooling (pooling layers).



La capa de convolución tiene como objetivo extraer los rasgos característicos de una imagen. Dichos rasgos se extraen aplicando filtros (kernels) para cada capa convolucional.

3.1.1. Arquitectura Usada



Usé la API secuencial de Keras, donde solo tiene que agregar una capa a la vez, comenzando desde la entrada,se definieron 3 bloques de capas, cada bloque consta de:

- La primera es la capa convolucional (Conv2D). Es como un conjunto de filtros que se pueden aprender. Elegí establecer 32 filtros en el primer bloque, 64 en el segundo y 128 en el tercero Cada filtro transforma una parte de la imagen (definida por el tamaño del núcleo) usando el filtro del núcleo.Como la imagen es de 32x32 use kernel de 3x3 que según la bobliografia es lo recomendado par el tamaño de la imagen. La matriz de filtro de kernel se aplica a toda la imagen. Los filtros pueden verse como una transformación de la imagen.

  La CNN aisla características que son útiles en todas partes de estas imágenes transformadas (mapas de características).

- Relu es el rectificador (función de activación max (0, x). La función de activación del rectificador se utiliza para agregar no linealidad a la red.
- La segunda capa importante en CNN es la capa de agrupación (MaxPool2D). Esta capa simplemente actúa como un filtro de reducción de resolución. Mira los 2 píxeles vecinos y elige el valor máximo. Estos se utilizan para reducir el costo computacional y, hasta cierto punto, también reducen el sobreajuste. Tenemos que elegir el tamaño de la agrupación (es decir, el tamaño del área agrupada cada vez) cuanto más alta es la dimensión de la agrupación, más importante es la reducción de resolución.

Combinando capas convolucionales y agrupadas, CNN puede combinar características locales y aprender más características globales de la imagen.

La capa Flatten/Aplanar se utiliza para convertir los mapas de características finales en un solo vector 1D. Este paso de aplanamiento es necesario para que pueda utilizar capas completamente conectadas después de algunas capas convolucionales / maxpool. Combina todas las características locales encontradas de las capas convolucionales anteriores.

Al final, utilicé las características en tres capas completamente conectadas (densas), que es solo un clasificador artificial de redes neuronales (ANN). En la última capa (Dense (10, activación = "softmax")) la distribución neta de las salidas de probabilidad de cada clase. Don 10 es el número de clases.

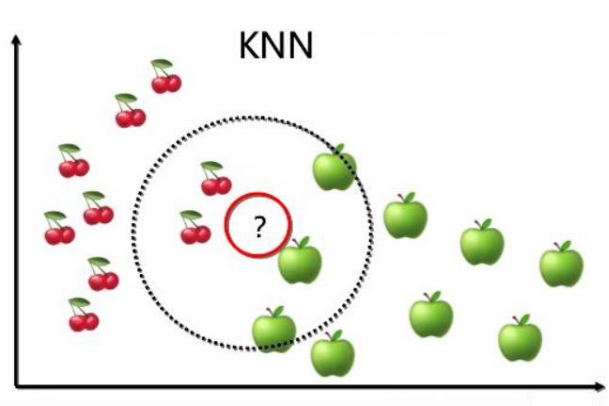
3.2. **KNN – Vecinos más cercanos**

El proceso de clasificación consiste en comparar directamente cada imagen del conjunto de prueba con todas las imágenes del conjunto de entrenamiento y calcular la distancia.

Cuanto mayor sea la distancia, menor será la similitud entre las imágenes; cuanto más cercana sea la distancia, más similares serán las imágenes.

EL objetivo es encontrar las imágenes K más cercanas a la imagen de prueba, cuente sus clasificaciones y use la clasificación más numerosa como clasificación de imagen de prueba.

- Calcular la distancia entre el item a clasificar y el resto de items del dataset de entrenamiento.
- Seleccionar los “k” elementos más cercanos (con menor distancia, según la función que se use)
- Realizar una “votación de mayoría” entre los k puntos: los de una clase/etiqueta que dominen decidirán su clasificación final.



Para decidir la clase de un punto es muy importante el valor de k, pues este terminará casi por definir a qué grupo pertenecerán los puntos, sobre todo en las fronteras entre grupos. Es recomendado elegir valores impares de k para desempatar. No será lo mismo tomar para decidir 3 valores que 13. Esto no quiere decir que necesariamente tomar más puntos implique mejorar la precisión. Cuantos más “puntos k”, más tardará nuestro algoritmo en procesar y darnos respuesta.

Las formas más populares de “medir la cercanía” entre puntos son la distancia o la Cosine Similarity (mide el ángulo de los vectores, cuanto menores, serán similares).

4. Evaluación de los modelos ML

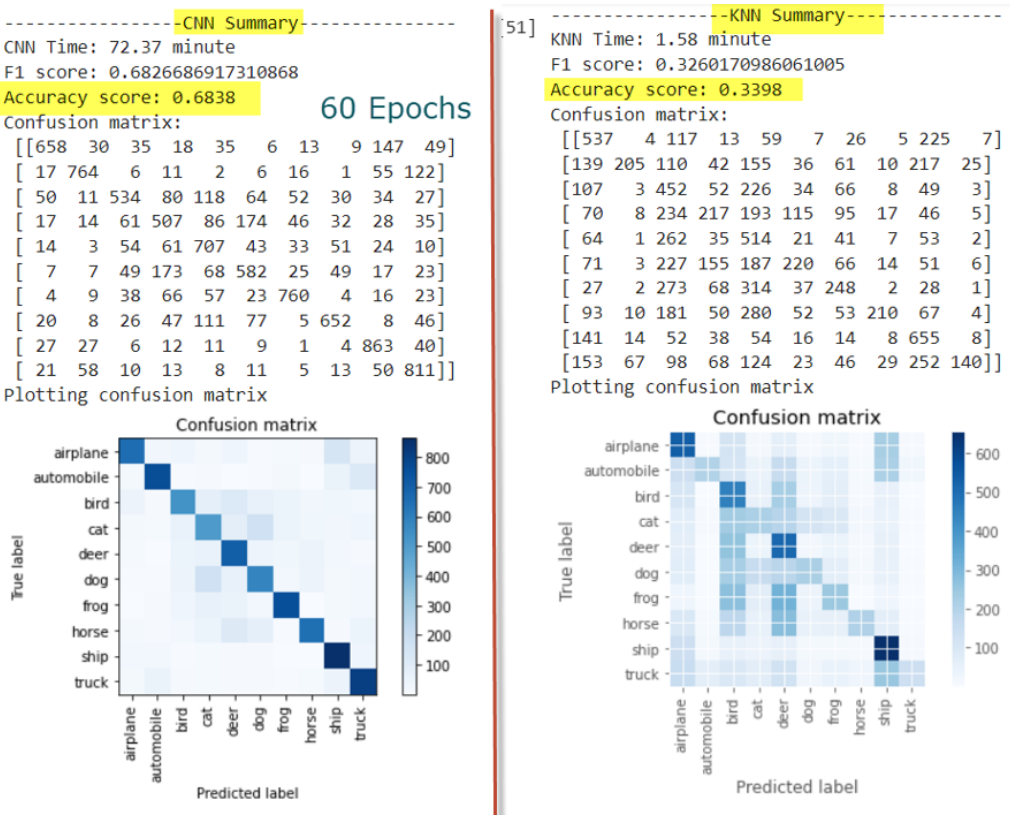
.

Para el presente proyecto se ha utilizado la librería metrics de sklearn de esta forma comparar todos los modelos bajo las mismas métricas:

Accuracy score : La métrica accuracy representa el porcentaje total de valores correctamente clasificados, tanto positivos como negativos.

Confusion Matrix: Es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado.

En la imagen siguiente podemos ver los resultados obtenidos:



5. Conclusiones

La red convolucional (CNN) tiene un mejor performance (casi 70%) ante el KNN que solo llega a 33%, mencionar que inicialmente se hizo un análisis con Kmeans pero el score obtenido ni siquiera llegaba a 1 por lo que se descartó este modelo inicial.

6. Recomendaciones

Como recomendación investigar como es el procesamiento de imágenes, en cuanto a tamaños, colores, calidad, etc etc.
Adicionalmente , se ha visto que las CNN aun tienen mas puntos donde explotar como mejorar la arquitectura de capas o las iteraciones de epoch, estas modificaciones puedes hacer una mejora significativa en el score obtenido

7. Enlaces externos y referencias

Repositorio github:
https://github.com/carminiaeguivar/imageClassifier_ML1_2021.git

Repositorio Dataset cifar-10:
https://www.cs.toronto.edu/~kriz/cifar.html

Bibliografía:
https://www.aprendemachinelearning.com/clasificacion-de-imagenes-en-python/
https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico