Clasificador de Imágenes con Red Neuronal Convolucional

Abraham Alejandro Salazar Hernandez

B210745

Matemáticas para la ciencia de la computación

Centro de Investigación en Computación

MCIC

November 1, 2021

Contents

1 Capítulo 1

1.1 Objetivo General

• Aprender a usar las herramientas de redes neuronales convolucionales que proporciona Python para un programa de clasificación de imágenes.

1.2 Objetivos Específicos

• Aprender los conceptos básicos de redes neuronales convolucionales.
• Aprender el uso de las librerías empleadas.
• Buscar un banco de datos de imágenes a clasificar.
• Crear un programa en Python para clasificación.

1.3 Justificación

Aprender a usar las herramientas de machine learning que incorpora Python (Keras, Tenserflow y Scikit-learn) para redes neuronales convolucionales, así como los conceptos requeridos para crear un programa de clasificación de imágenes

2 Capítulo 2

2.1 Estado del arte

Autores	Título	Año	Revista
Andrew G. Howard	Improvements on Deep Convolutional Neural Network Based Image Classification.	2013	IEEE
F. Sultana, A. Sufian and P. Dutta.	Advancements in Image Classification using Convolutional Neural Network.	2018	IEEE
N. Jmour, S. Zayen and A. Abdelkrim.	Convolutional neural networks for image classification.	2018	IEEE
Neha Sharma, Vibhor Jain and Anju Mishra.	An Analysis Of Convolutional Neural Networks For Image Classification.	2018	Procedia Computer Science
D. P. Sudharshan and S. Raj	Object recognition in images using convolutional neural network.	2018	IEEE

2.2 Fundamentos

Se creó un programa de clasificación de imágenes para aprender los conceptos básicos de redes neuronales convolucionales.

Dentro de los conceptos principales son Capas y Neuronas, todo programa de inteligencia artificial debe contar con por lo menos una capa de entrada y un de salida, pero puede tener varias capas con varias neuronas, entre capas pueden existir funciones de activación, que esencialmente es una función que se aplica a la salida de los datos de una capa y se manda a los datos de entrada de la siguiente capa.

La red neuronal convolucional tiene una función convolucional entre las capas de neuronas, la función convolucional tiene la propiedad que extrae características de la imagen, además requiere de una capa extra de agrupación antes de pasar a la siguiente capa neuronal.

La capa de agrupación maximiza los rasgos y disminuye la dimensión de la imagen original, permitiendo hacer más convoluciones.

Tanto la capa convolucional como la capa de agrupación consta de una matriz que al ir pasando de pixel en pixel hace una operación con los pixeles circundantes generando una nueva imagen a partir de

los resultados de la convolución y de la agrupación.

2.3 Solución al problema

Se realizo un programa en Python para hacer un clasificador de imágenes, con dos técnicas distintas, para hacer una comparación entre un Machine Learning normal y otro ocupando redes convolucionales.

Para ambos casos se ocupó un banco de imágenes, proporcionado por tensorflow dataset, que es de ropa de una tienda departamental, el banco de imágenes es de 70000 imágenes previamente clasificadas, dentro de ese banco de imágenes, se dividió en 60000 para entrenar y solo 10000 para corroborar, las imágenes tienen un tamaño de 28x28 pixeles en escala de grises.

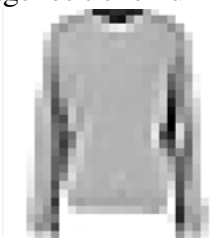


Imagen de 28x28 pixeles de prenda de ropa

Para el entrenamiento se pidió que se realizaran 32 convoluciones con una matriz de 3x3 pixeles, para una entrada de imágenes de 28x28 pixeles de tamaño con una dimensión, debido a que están en escala de grises, si fueran a color, se ocuparían 3 dimensiones (rojo, azul, amarillo), la capa de adición fue de 2x2 pixeles, la segunda convolución fue muy similar, se pidió que se realizaran 64 convoluciones igual de 3x3 pixeles y la misma capa de agrupación.

```
tf.keras.layers.Conv2D(32, (3,3), input_shape=(28,28,1), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
```

Fragmento de código

Una vez exaltado las características principales, y comprimido la imagen esta debe pasar por la red neuronal, para eso la imagen debe ser aplanada, en otras palabras, deben enlistarse todos los pixeles en un arreglo de 1 x n, por poner un ejemplo, una imagen de 28x28 pixeles debe ser reacomodada en un arreglo de 1x784, por lo tanto, debe haber 784 neuronas de entrada.

Después para mayos precisión se ocupo una capa oculta con 100 neuronas, también se probó con dos capas de 50 neuronas y con tres capas de 75 neuronas, el mejor resultado se observó con una capa con 100 neuronas y para garantizar que todas las neuronas trabajarán se ocupó una función de

desactivación de neuronas (dropout), de esta manera todas las neuronas trabajan por igual, finalmente la capa de salida debe ser igual al número de clasificación que tenemos para las imágenes, para este caso solo se pusieron 10 neuronas de salida.

```
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(units=50, activation='relu'),
tf.keras.layers.Dense(units=50, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
```

Fragmento de código

Cabe resaltar que las neuronas estaban densamente conectadas (todas las neuronas de un capa, están conectadas a todas las neuronas de la capa siguiente), la función de activación a ocupar en las capas intermedias fue de ReLu, existen otras funciones como leaky Relu, Exponencial Lu, Sigmoide y Escalón pero ReLu es más sencilla de ocupar, la capa de salida tiene la función de softmax para normalizar esa última capa y entregue valores entre cero y uno.

Después se tiene una función de optimización que se encarga de buscar los mínimos locales, debido a que el entrenamiento se hace en tiempo real y no se conocen todos los valores que se ocupan en las capas, es imposible sacar una grafica y ver visualmente donde hay un mínimo local o mínimo global, es por eso la importancia de alguna función que se dedique a buscar el menor error posible, para este programa se ocupó una función llamada ADAM, que es la mezcla de dos herramientas que se encargan del descenso del gradiente (MOMENTUM y RMSProp) ya que gracias a la fusión de estas dos herramientas, ADAM converge mar rápido al mínimo local.

El error calcula la diferencia entre la salida del sistema y la salida deseada, el parámetro accuracy calcula el porcentaje de aciertos.

```
optimizer="adam",
loss=tf.keras.losses.SparseCategoricalCrossentropy(),
metrics=["accuracy"]
```

Fragmento de código

3 Capítulo 3

3.1 Resultados

Al inicio del entrenamiento, se observa que la función de perdida es alta y el porcentaje de aciertos es bastante bajo, ya que el entrenamiento aún no comienza, por eso aparecen esos valores.

```
Epoch 1/5
  8/1875 [..............................] - ETA: 1:48 - loss: 2.2377 - accuracy: 0.1562
```

Captura de pantalla

Con forme avanza el entrenamiento, al final de las 5 épocas, observamos que el porcentaje de aciertos está en el 90 por ciento con una capa oculta de 100 neuronas.

```
Epoch 1/5
1875/1875 [==============================] - 133s 68ms/step - loss: 0.5058 - accuracy: 0.8136
Epoch 2/5
1875/1875 [==============================] - 119s 63ms/step - loss: 0.3612 - accuracy: 0.8664
Epoch 3/5
1875/1875 [==============================] - 177s 95ms/step - loss: 0.3140 - accuracy: 0.8846
Epoch 4/5
1875/1875 [==============================] - 114s 61ms/step - loss: 0.2877 - accuracy: 0.8933
Epoch 5/5
1875/1875 [==============================] - 121s 65ms/step - loss: 0.2686 - accuracy: 0.9002
```

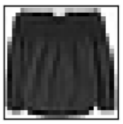
Captura de pantalla

Mayor número de neuronas o más capas, no representa un cambio significativo positivo al aprendizaje, de igual forma el tener un número de neuronas mayor a 87 y menor de 100 tampoco representa una deficiencia drástica ya que el porcentaje de aciertos es muy próximo al 90 por ciento. El entrenamiento de esta red neuronal tomó aproximadamente 10 minutos y los resultados fueron favorables en la mayoría de los casos, incluso con cambios en el numero de neuronas y capas ocultas. A continuación, se muestran algunos valores ejemplos con el set de datos de verificación.


Shirt 59% (Shirt)


Sneaker 93% (Sneaker)


Dress 89% (Dress)


Bag 42% (Shirt)


Dress 98% (Dress)

En verde las clasificadas correctamente, del lado izquierdo del texto el porcentaje de acierto, entre paréntesis, la clasificación correcta.

3.2 Conclusiones

El diseño de redes neuronales es más un arte que un razonamiento puramente lógico, no encontré practicas detalladas de cuando ocupar un método frente a otro o cual es mejor que otros, si bien hay funciones dedicas al tipo de aprendizaje a realizar, entre esas técnicas el saber cuál elegir es más cuestión de conocimiento empírico que otra cosa, se requiere de mucha experiencia para poder elegir bien que técnicas se van a ocupar para que tenga un funcionamiento óptimo por lo que es muy importante tener pleno conocimiento de los conceptos y técnicas que existen, y entre mejor y más información se tengan de las funciones empleadas en machine learning, mejor y más precisos serán los programas que realicemos.