Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Estado de México

Escuela de ingeniería y ciencias

Inteligencia artificial avanzada para la ciencia de datos II (Gpo 501)

Momento de Retroalimentación: Módulo 2 Implementación de un modelo de deep learning.

Andrea Vianey Díaz Álvarez A01750147

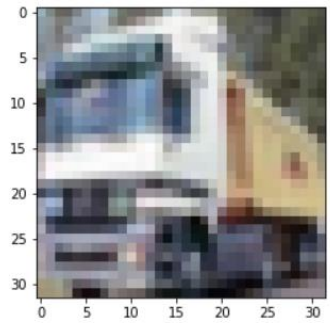
Profesor:
Julio Arriaga

Fecha: 02/11/2022

Análisis del modelo CNN

Descripción de las imágenes:

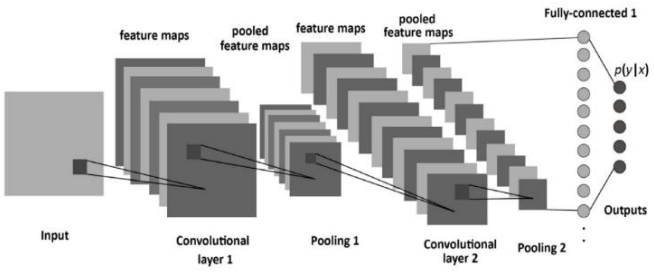
- Cada imagen tiene alguna de las siguientes opciones:
    - Avión
    - Automóvil
    - Pájaro
    - Gato
    - Ciervo
    - Perro
    - Rana
    - Caballo
    - Barco
    - Camión
- 50000 imágenes de entrenamiento y 10000 imágenes de prueba.
- *Ejemplo:*



Tamaño del Vecindario: (32,32)

Profundidad: 3

Arquitectura CNN Típica:



Basándome en la arquitectura típica, he decidió empezar a cambiar algunos parámetros de cada capa donde:

1. Las capas de **convolución** tienen los parámetros:
    a. Número de kernels tridimensionales (k)

   b. Tamaño de los kernels (F x F)
   c. Tamaño del paso del Kernel (stride)
   d. Tamaño del padding
   e. Función de activación
2. Las capas de **pooling** tienen los parámetros:
   a. Tamaño del Pool (F x F)
   b. Tamaño del paso del Pool (stride)
3. Las capas de **dropout** tienen el parámetro:
   a. Rate entre 0 y 1: % de neuronas que se va a desactivar
4. Las capas de **dense o fully connected** tienen el parámetro:
   a. Función de activación

Batch_size = 120
Epochs = 20

Prueba 1

```
model = Sequential([
    Conv2D(100, kernel_size=(2, 2), padding='same', strides = 1, activation="relu", input_shape=(32,32,3)),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(100, kernel_size=(2, 2), activation="relu"),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.1),
    Flatten(),
    Dense(10, activation='softmax')
])
```

Train:

- ```
  Epoch 20/20
  417/417 [==============================] - 4s 10ms/step - loss: 0.6641 - sparse_categorical_accuracy: 0.7702 - mean_absolute_error: 4.4200
  ```

Test:

- ```
  313/313 [==============================]
  loss 0.8185326457023621
  acc 0.7229999899864197
  mae 4.4199981689453125
  ```

Prueba 2

```
model = Sequential([
    Conv2D(200, kernel_size=(3, 3), padding='same', strides = 1, activation="relu", input_shape=(32,32,3)),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(200, kernel_size=(3, 3), activation="relu"),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.1),
    Flatten(),
    Dense(10, activation='softmax')
])
```

Train:

- ```
  Epoch 20/20
  417/417 [==============================] - 8s 19ms/step - loss: 0.4347 - sparse_categorical_accuracy: 0.8497 - mean_absolute_error: 4.4200
  ```

Test:
  ```

```
313/313 [==============================]
loss 0.8103728294372559
acc 0.7350000143051147
mae 4.4199981689453125
```

Prueba 3

```
model = Sequential([
    Conv2D(200, kernel_size=(3, 3), padding='valid', strides = 1, activation="relu", input_shape=(32,32,3)),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(200, kernel_size=(3, 3), activation="relu"),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.1),
    Flatten(),
    Dense(10, activation='softmax')
])
```

Train:

```
Epoch 20/20
417/417 [==============================] - 8s 18ms/step - loss: 0.5092 - sparse_categorical_accuracy: 0.8250 - mean_absolute_error: 4.4200
```

Test:

```
313/313 [==============================]
loss 0.7947880625724792
acc 0.7366999983787537
mae 4.4199981689453125
```

Prueba 4

```
model = Sequential([
    Conv2D(200, kernel_size=(3, 3), padding='same', strides = 2, activation="relu", input_shape=(32,32,3)),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(200, kernel_size=(3, 3), activation="relu"),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.1),
    Flatten(),
    Dense(10, activation='softmax')
])
```

Train:

```
Epoch 20/20
417/417 [==============================] - 3s 7ms/step - loss: 0.6828 - sparse_categorical_accuracy: 0.7666 - mean_absolute_error: 4.4200
```

Test:

```
313/313 [==============================]
loss 0.8164435029029846
acc 0.725600004196167
mae 4.4199981689453125
```

Prueba 5 (Mejor resultado)

Epochs = 50

```
model = Sequential([
    Conv2D(100, kernel_size=(3, 3), padding='same', strides = 1, activation="relu", input_shape=(32,32,3)),
    MaxPooling2D(pool_size=(2, 2), strides = 2),
    Conv2D(100, kernel_size=(3, 3), activation="relu"),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.1),
    Flatten(),
    Dense(10, activation='softmax')
])
```

Train:

- ```
  Epoch 30/30
  417/417 [==============================] - 4s 10ms/step - loss: 0.5208 - sparse_categorical_accuracy: 0.8205 - mean_absolute_error: 4.4200
  ```

Test:

- ```
  313/313 [==============================]
  loss 0.7988223433494568
  acc 0.7398999929428101
  mae 4.4199981689453125
  ```

ANÁLISIS DE RESULTADOS

Empecé hacer mis pruebas tomando como base la arquitectura típica además de esto, los parámetros que escogí fueron pensando en lo que probablemente iba conseguir los mejores resultados en este caso, tomando en cuenta el tamaño de mis imágenes, cantidad y los colores de las imágenes.

Para resolver el problema de los bordes que se puede poner el kernel por fuera de las imágenes decidí intentar con 2 soluciones: “valid” que coloca el kernel únicamente en ventanas válidas y “same” que aplica un relleno con ceros.

Para el desplazamiento del kernel, empecé usando un valor de 1, sin embargo, decidí cambiarlo a 2 para ver si esto de alguna manera lograba mejorar mi modelo. En general el modelo había tenido excelentes resultados con un valor de 1, el resultado fue el esperado, no se logró mejorar los resultados y opté por dejarlo en 1.

Una de las pruebas que hice fue hacer una comparación para ver cómo afecta tener un número distinto de kernels. Primero lo probé con 100 y después con 200. Esto me comprobó que tener un número muy bajo como 10 no trae buenos resultados, pero tampoco hubo un cambio significativo cuando probé con 200, por lo tanto, optaría por dejarlo en 100.

También dando resultados significativos fue el cambiar el tamaño tanto del kernel como del pool. En el mejor caso fue tener el kernel con tamaño 3 x 3 y el pool con tamaño 2 x 2 pero para hacer la comparación intercambié los valores.

Lo último que hice, después de definir los mejores parámetros con respecto a las pruebas fue modificar el número de Epochs, intenté ponerle 150 pero la verdad que no hubo ningún cambio significativo así que opté por dejarlo en 30 que fue hasta donde se lograron realmente resultados significativos ya que a pesar de que el entrenamiento si tuvo casi el 99 de accuracy en la prueba seguía teniendo 73.

CONCLUSIÓN

Ya que decidí partir de una arquitectura que funciona para muchos casos, tuve resultados positivos desde el principio y las pruebas que hice fueron para ver si podría obtener aun mejores resultados y de qué manera podrían afectar los parámetros a mi modelo, en muchos casos es prueba y error hasta obtener los resultados deseados e investigar para poder tener mejor conocimiento de que es lo que mejor funciona para el modelo y los datos que se están utilizando. En el futuro me gustaría hacer más pruebas y cambiar más las variables para aprender a hacer modelos eficientes para cualquier conjunto de imágenes.