Universidad Nacional Autónoma de México

Facultad de Ciencias

Redes Neuronales

Ejercicio 08 - Redes Convolucionales

Carlos Emilio Castañon Maldonado

1 Asocia la descripción con el tipo de acolchamiento

Se agregan píxeles tratando de adivinar como continuaría la imagen ➢ Extrapolación
Al salir por un borde, los valores se repiten pero ahora en orden inverso ➢ Espejo o Simetría
Al salir por un borde, el siguiente valor es el inicial del otro borde ➢ Circular
Se colocan ceros al rededor de la imagen hasta alcanzar el tamaño deseado ➢ Relleno con Ceros (Zero Padding)

2 ¿Que operaciones se utilizan para realizar la convolucion de una imagen con un filtro y obtener como resultado otra imagen del mismo tamaño?
Seleccione una o mas con ★

a) Desenfoque (Blur)
b) Recorte (Crop) ★
c) Softmax
d) Acolchamiento (Padding) ★
e) Union (Pooling)

3 ¿Que dimensiones tiene el área con resultados distintos de cero al calcular la convolucion de las dos funciones discretas siguientes?

$$f = \begin{bmatrix} -1 & 2 & 1 \\ 0 & 1 & -1 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad g = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Recordando que: $N + M - 1$

Tenemos:

$4 + 2 - 1 = 5$
Ren: 5

$3 + 2 - 1 = 4$
Col: 4

4 **¿Cuanto vale la convolucion de las dos funciones siguientes?**

$$f = \begin{bmatrix} -1 & 2 & 1 \\ 0 & 1 & -1 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad g = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ejecutando el siguiente script de python:

```python
import numpy as np
def convolucion(funcion1, funcion2):
    # Obtenemos las dimensiones de las funciones
    m, n = funcion1.shape
    p, q = funcion2.shape

    # Calculamos el tamaño del resultado de la convolucion
    resultado_m = m + p - 1
    resultado_n = n + q - 1

    # Creamos una matriz de ceros para almacenar el resultado
    resultado = np.zeros((resultado_m, resultado_n))

    # Realizamos la convolucion
    for i in range(resultado_m):
        for j in range(resultado_n):
            for k in range(p):
                for l in range(q):
                    if i - k >= 0 and i - k < m and j - l >= 0 and j - l < n:
                        resultado[i, j] += funcion1[i - k, j - l] * funcion2[k, l]

    return resultado


if __name__ == "__main__":
    # Definimos las matrices de las funciones
    funcion1 = np.array([[-1, 2, 1],
                         [0 , 1,-1],
                         [1 , 2, 0],
                         [ 1, 0, 1]])

    funcion2 = np.array([[-1, 0],
                         [ 0, 1]])

    # Calculamos la convolucion
    resultado_conv = convolucion(funcion1, funcion2)

    # Imprimimos el resultado
    print("Funcion 1:")
    print(funcion1)
    print("\nFuncion 2:")
    print(funcion2)
    print("\nConvolucion:")
    print(resultado_conv)
```

Tenemos como resultado:

$$f = \begin{bmatrix} -1 & 2 & 1 \\ 0 & 1 & -1 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad g = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
batman@brother-eye:  /Ejercicio08$ python script.py
Funcion 1:
[[-1  2  1]
 [ 0  1 -1]
 [ 1  2  0]
 [ 1  0  1]]

Funcion 2:
[[-1  0]
 [ 0  1]]

Convolucion:
[[ 1. -2. -1.  0.]
 [ 0. -2.  3.  1.]
 [-1. -2.  1. -1.]
 [-1.  1.  1.  0.]
 [ 0.  1.  0.  1.]]
```

$$f * g = \begin{bmatrix} 1 & -2 & -1 & 0 \\ 0 & -2 & 3 & 1 \\ -1 & -2 & 1 & -1 \\ -1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$