Tarea 1

Detección de objetos en imágenes basado en el modelo de Hubel y Wiesel

Profesor: Claudio Pérez F.
Auxiliar: Diego Maureira
Ayudantes: Juan Pablo Pérez, Jhon Pilataxi, Jorge Zambrano

1. Objetivo

El objetivo de esta tarea es desarrollar de forma progresiva un sistema simple de detección de objetos en imágenes, basado en la teoría de campos receptivos de Hubel y Wiesel.

2. Descripción

El reconocimiento de objetos es un problema altamente desafiante y que continúa bajo investigación, debido a la alta variabilidad de patrones y las diferentes estrategias para abordar el problema. En la actualidad, las Redes Neuronales Convolucionales (CNNs, por sus siglas en inglés) han presentado un buen desempeño comparadas con otros tipos de sistemas detectores. Las CNNs extraen características importantes de las imágenes por medio de la utilización de filtros convolucionales. Este tipo de filtros se basan en el modelo de Hubel y Wiesel, quienes descubrieron que ciertas neuronas se excitan cuando un patrón simple se sitúa en una zona particular de la retina, y que dichos estímulos decrecen cuando el patrón se aleja.

Esta tarea tiene como finalidad implementar detectores simples de patrones, diseñando filtros capaces de encontrar ciertos objetos dentro de una imagen. La primera aproximación se hará mediante un ejemplo básico, para luego aplicar lo aprendido en imágenes entregadas por el equipo docente, llegando a detectar elementos como una pieza de ajedrez en un tablero o palabras en una sopa de letras.

3. Detección de un patrón simple

En esta sección se desarrollará la detección de un patrón simple en una imagen pequeña. Esta aproximación será útil para interiorizar los conceptos básicos de los detectores basados en filtros convolucionales.

3.1. Implementación de la función de convolución

Implemente una función que reciba como entrada un kernel y una imagen, donde ambos puedan tener cualquier tamaño, y que ejecute la convolución entre ellos. Resuelva el proble-

ma que se genera en los bordes mediante la aplicación de la técnica de *padding*. Sean *fil* y *mat* el kernel y matriz que se quieren convolucionar, respectivamente. Una forma muy simple de hacer *zero padding* sobre la matriz, podría verse así:

```
import numpy as np

m, n = fil.shape
y, x = mat.shape
padded = np.zeros((y+m-1,x+n-1))
padded[m//2:-m//2+1, n//2:-n//2+1] = mat

```

Nota: Puede investigar acerca de otras metodologías de *padding* y utilizar la que quiera.

3.2. Creación de filtro detector de letras L

Dentro de una matriz de ceros de 10×10 pixeles, cambie el valor desde 0 a 1 en los pixeles verticales de índices 3 a 5 con el índice horizontal fijo en 3, y también en los pixeles horizontales de índices 3 a 4 con el índice vertical fijo en 5. Esto generará una letra "L" en la matriz.

Luego, utilizando un kernel de 3×3 balanceado (-1 en los bordes y 8 en el centro), genere un filtro detector para este patrón, siguiendo el esquema de Hubel y Wiesel. Para generar el filtro detector, haga la convolución entre el kernel balanceado y la matriz que contiene la letra "L". Finalmente, elimine las filas y columnas que contengan solo ceros en la matriz resultante.

3.3. Detección de letra L

A través de la función de convolución creada previamente, aplique el filtro generado sobre la matriz que contiene la letra "L". De esta forma, se está haciendo la detección de esta letra.

Presente los valores del filtro detector y del resultado de la detección. Para una mayor comprensión, genere mapas de calor (colorbar) de las matrices y explique los resultados.

4. Detección de patrones complejos

En esta sección, se cargarán las diferentes imágenes entregadas por el equipo docente y se detectarán ciertos objetos en ellas. Para esto, debe trabajar con imágenes binarias, pero presentar los resultados sobre las imágenes originales. **Muestre los resultados intermedios que se solicitan solo para un ejemplo en particular que usted desee**.

4.1. Lectura de imágenes

Lea una de las imágenes entregadas y conviértala a imagen binaria. **Hint**: Verifique si es necesario invertir el patrón binario.

4.2. Definición de patrón

Diseñe un algoritmo que permita recortar un patrón desde una imagen, señalando con un clic la parte superior izquierda (L) y la parte inferior derecha (R) del rectángulo (también llamado *bounding box*) que contiene al objeto, como se muestra en la Figura 1. **Hint**: Puede utilizar la función ginput de la librería Matplotlib para obtener los puntos solicitados.

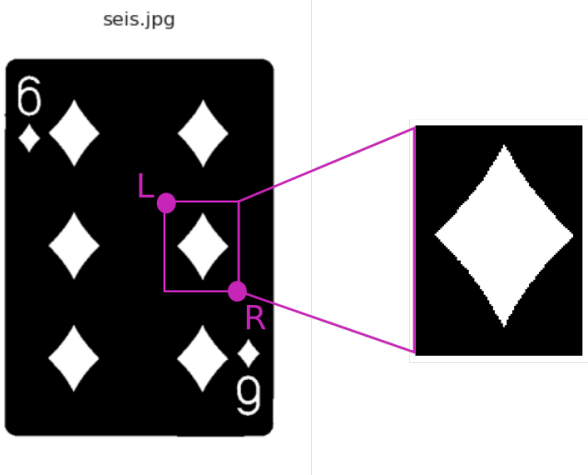

Figura 1: Funcionamiento esperado del algoritmo de recorte.

4.3. Creación del filtro detector

Calcule el filtro detector del patrón seleccionado al igual que en la sección anterior, es decir, ejecutando una convolución entre el **kernel balanceado** y el patrón recortado. La matriz resultante de esta convolución corresponde al filtro detector buscado.

4.4. Detección del patrón

Ejecute una convolución entre el filtro detector y la imagen binaria completa. De esta forma, está realizando la detección de todas las instancias del patrón en la imagen.

4.5. Determinación del umbral

Convierta la matriz resultante del punto anterior a un vector fila, y muestre los resultados en un gráfico. Esto le servirá para decidir un umbral de detección, por ejemplo, para el resultado mostrado en la Figura 2, un umbral adecuado sería aproximadamente 300 (poco más de un 60 % del estímulo máximo).

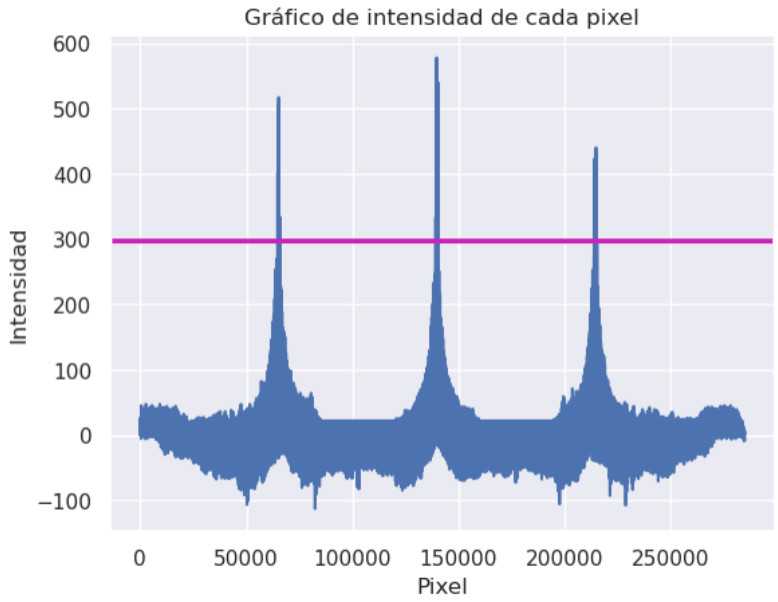

Figura 2: Ejemplo de representación gráfica de los resultados de activación.

4.6. Supresión de no Máximos o *Non-Max Supression* (NMS)

Debe tener en consideración que puede existir más de un punto, asociado a la detección de un mismo objeto, que supere el umbral definido. En estos casos, la idea es conservar solo el punto de máxima activación. Para esto, recupere las posiciones (x, y) de los valores que superan el umbral e implemente y ejecute un algoritmo para evitar múltiples detecciones de un mismo objeto.

4.7. Visualización de la detección

Grafique sobre la imagen original *bounding boxes* alrededor de las posiciones (x, y) que sobreviven al algoritmo de NMS. **Hint**: El tamaño de la *bounding box* debería tener relación con el tamaño del filtro usado.

5. Actividad final

- A continuación se listan las imágenes entregadas y se indican los patrones que deben ser detectados en cada una de ellas, a través de la metodología descrita en la sección 4. Además, debe obtener visualizaciones como las ejemplificadas en la Figura 3, especificando la cantidad de objetos encontrados.

  1. **ChessBoard.jpg**: Caballos - Peones
  2. **SopaLetras.jpg**: La palabra CAT (Vertical) y DOG (Horizontal)
  3. **cartas.jpg** y **seis.jpg**: Diamantes grandes
  4. **mancha.jpg**: Palabra “no"

5. **seis.jpg**: Diamantes nuevamente, pero **usando el filtro obtenido desde la imagen cartas.jpg**

- Comente las ventajas y desventajas de este método. Explique por qué se dan falsos positivos y negativos al momento de la detección.
- Finalmente, proponga (no hace falta implementarlo) cómo se pueden obtener mejores resultados con este algoritmo.

A continuación, se presentan algunos resultados esperados:

Figura 3: Ejemplos de resultados esperados.

6. Entregables

- Presente un reporte del trabajo. Incluya en el informe las detecciones solicitadas en cada imagen.
- El código también debe ser entregado, asegúrese de que pueda ser ejecutado por los revisores para comprobar el algoritmo.

7. Recordatorio

- El reporte es individual.
- Recuerde que, para realizar la convolución, el filtro debe ser reflejado horizontal y verticalmente.

La fecha de entrega de trabajo será el día 10/09/2024 a las 23:59 hrs por medio de U-cursos.