Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Tarea 1

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

Integrantes: Ricardo García
Profesor: Claudio Pérez F.
Auxiliar: Jorge Zambrano I.
Ayudante: Juan Pablo Pérez C.
Fecha de entrega: 12 de Octubre de 2021
Santiago de Chile

Índice de Contenidos

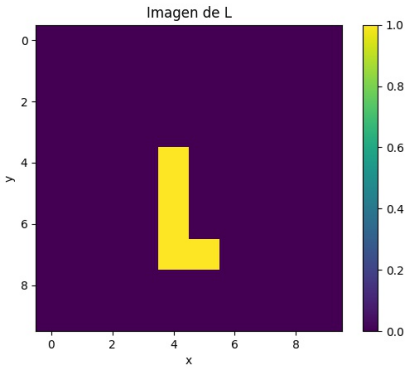
Índice de Figuras

Índice de Tablas

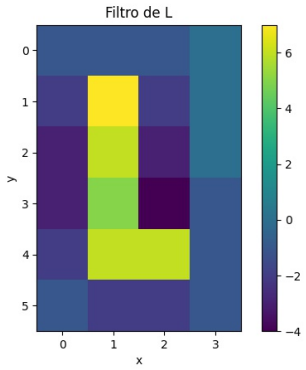
1. Parte a

Primero se implementó una función que realizara la convolución en dos dimensiones de una imágen con un kernel dado. Posteriormente se construyó una matriz de ceros que en el centro tenía una agrupación de 1's con la forma de una L. Finalmente se realizó la convolución de un kernel balanceado de 3x3 con valor 8 en el centro y -1 en los bordes.

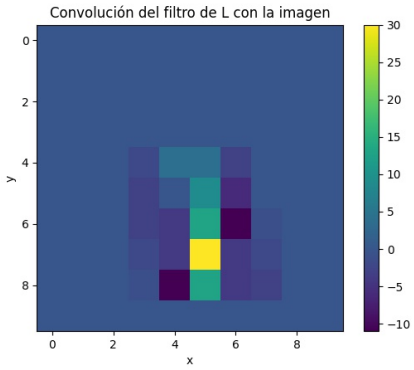
La imagen generada de una L, junto a su filtro detector y la convolución entre ambas se muestra en la figura 1. En ellas, se puede apreciar que la figura (a) es una imagen binaria en donde los pixeles que continen la L tienen valor 1, mientras que el fondo está representado con un 1. Además, en la figura (b) se aprecia que el filtro concentra valores más altos en el centro con forma de L, y en la figura (c), se observa que al convolucionar la imagen con el filtro existe un punto con el máximo valor que representa el lugar de detección del filtro, que tiene un valor aproximado de 30, y este estímulo disminuye a medida de que se aleja del patrón.



(a) Imagen de L generada en matriz de 10x10



(b) Filtro generado para detectar L



(c) Convolución realizada con el filtro detector de L con la imagen

Figura 1: Imagen generada de una L con sus respectivo filtro de detección y la convolución entre ambos

2. Parte b

Primero se implementó un algoritmo que convierta una imagen RGB, es decir que contiene un 3 valores por pixel que representan los colores rojo, verde y azul, a una imagen que contiene sólo un valor por pixel que puede ser 1 para blanco o 0 para negro. Esta transformación se hizo sumando los tres canales de color por cada pixel y si el resultado se compara con un umbral $threshold = \frac{255 \cdot 3}{2} = 382.5$. Este umbral se calculó considerando que cada valor en cada canal está entre 0 y 255, por lo que si la suma está por sobre el máximo de la suma de los tres canales, se considera que el pixel es blanco, de lo contrario negro. Los resultados al convertir la imagen "ChessBoard" a imagen binaria se muestran en la figura 2

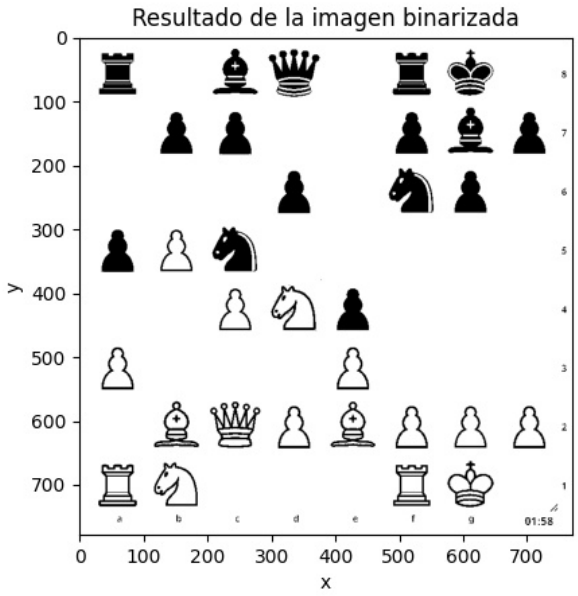


Figura 2: Transformación de la imagen “Chessboard” a imagen binaria

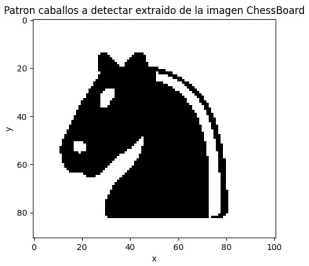
Luego, se utilizó la herramienta *ginput* para poder seleccionar patrones de detección sobre una imagen. Primero se hace un click sobre la esquina superior izquierda y luego en la esquina inferior derecha para obtener los pixeles de la imagen original correspondientes al patrón. En la figura 3 se muestran los patrones recortados para la detección en las imágenes.

Luego, se calcularon los filtros detectores para cada imagen utilizando el mismo kernel balanceado de la sección a. Los resultados de muestran en la figura 4. Posteriormente, se hizo la convolución entre los filtros y las imágenes binarias, en donde las resultantes fueron representadas en un vector fila, y graficadas. Los resultados se muestran en la figura 5. En ellas se puede apreciar que todas presentan peaks en ciertos pixeles los cuales representan la exitación en el filtro y por ende la detección del patrón. Es por esto, que se eligió un umbral para cada imagen, de forma de que se obtengan los pixeles en los cuales hay un peak de estímulo y así la posición de la figura. Los umbrales utilizados se muestran en la tabla 1.

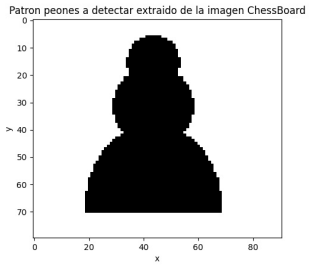
Con la ubicación de los patrones en las imágenes obtenidas anteriormente, se graficó sobre la imagen una caja que indicara dónde se realizó la predicción, junto al número de predicciones totales. El resultado se muestra en la figura 6. En estas, se puede apreciar que a pesar de que en general se hace la detección bien, se predicen más objetos de los que realmente hay. Esto se debe a que al realizarse la convolución, y filtrar por un umbral, los pixeles vecinos al patrón a detectar, también tienen un estímulo similar, como se muestra en la figura 1. Es por esto que se diseñó un algoritmo que filtra los puntos detectados, guardando sólo una ubicación que represente a los puntos que estén en una vecindad cercana. Los resultados de aplicar este filtro se aprecian en la figura 7, en ella se comprueba que el número de detecciones propuestas coinciden con el número de patrones encontrados.

Las ventajas de utilizar este tipo de detección es que es muy simple de implementar y es barato computacional mente hablando. Esto es, porque solo se hacen operaciones matriciales del orden

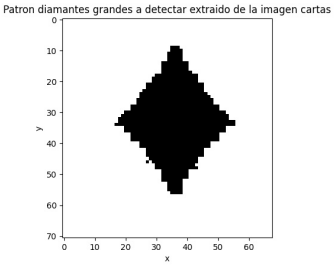
del tamaño de las imágenes. Sin embargo, en la figura 7 (g) se observa que la detección puede ser muy susceptible a patrones similares, ya que se detectan falsos negativos como “he”, “ne” y “mo” que tienen una alta similitud con el patrón “no”. Al tenerse tal semejanza, en la figura 5 (g) se observa que para este caso existen muchos peaks de similar valor que pueden corresponder tanto a patrones detectados correctamente como incorrectamente, por lo que realizar un filtrado por umbral no es suficiente. Es por esto, que se propone realizar una composición de filtros, que se enfoquen en detectar características más simples y al mezclarlos, se pueda detectar con mayor verosimilitud el patrón buscado. Esto es muy similar a como funciona el ojo, ya que las neuronas fotorreceptoras tienen campos receptivos que detectan patrones simples como líneas y es por la composición de estos que se pueden detectar patrones más complejos como letras, palabras, rostros, etcétera.



(a)
Caballo

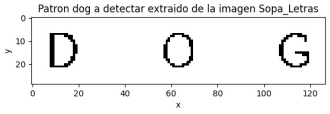


(b)
Peón

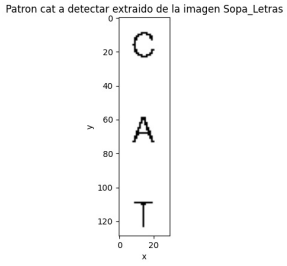


(c) Diamantes en "cartas"

(d)
Diamantes en "seis"

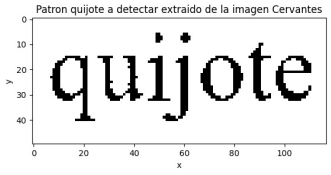


(e) Dog



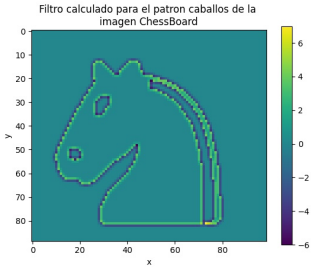
(f) Cat

(g) No

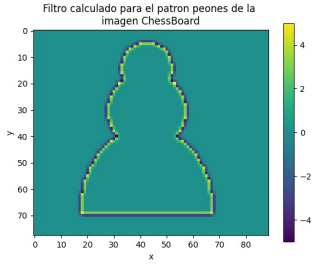


(h)
Quijote

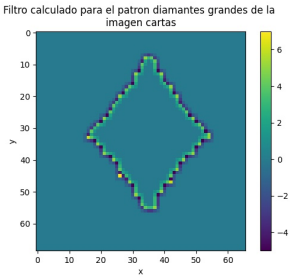
Figura 3: Patrones extraídos de las imágenes



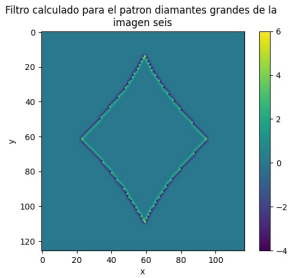
(a) Caballo



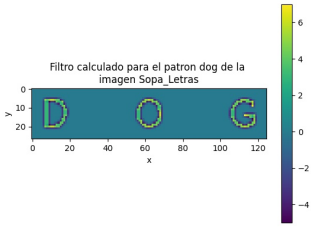
(b) Peón



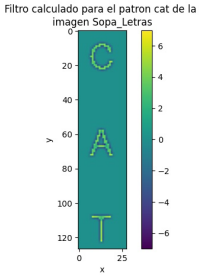
(c) Diamantes en "cartas"



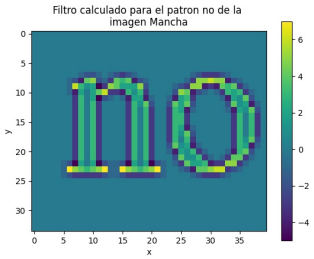
(d) Diamantes en "seis"



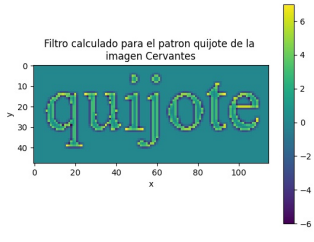
(e) Dog



(f) Cat

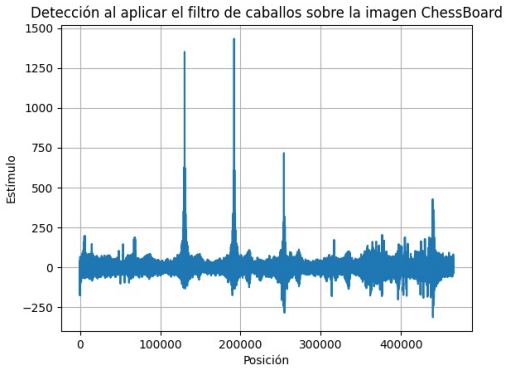


(g) No

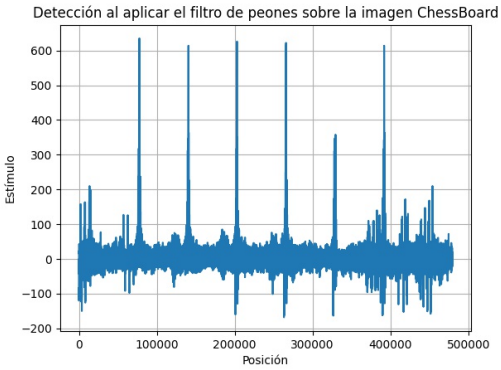


(h) Quijote

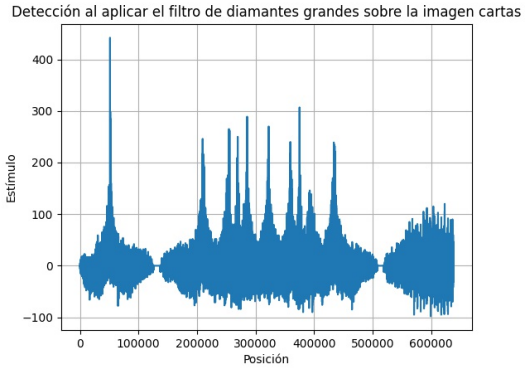
Figura 4: Filtros obtenidos para los patrones



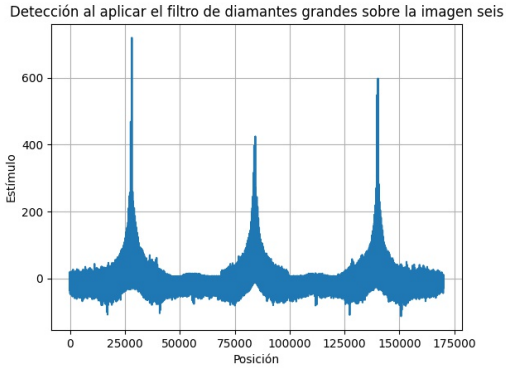
(a) Caballo



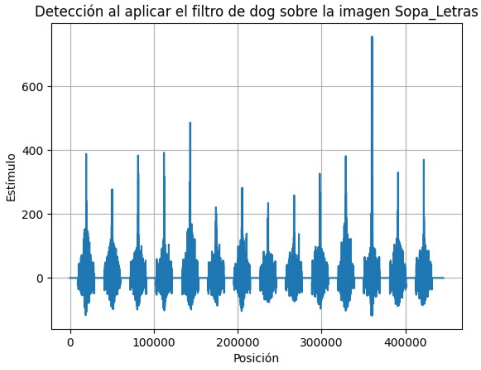
(b) Peón



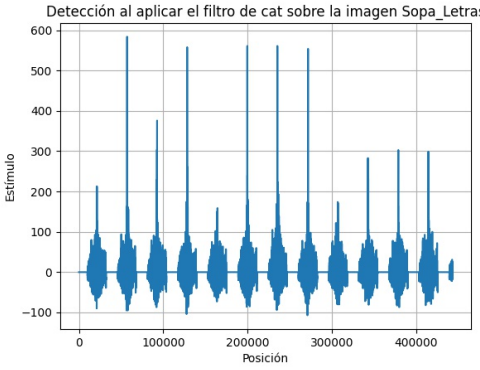
(c) Diamantes en "cartas"



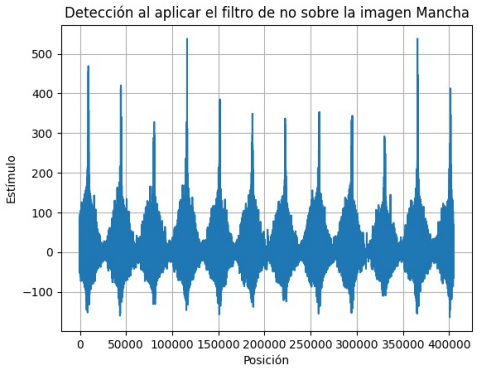
(d) Diamantes en "seis"



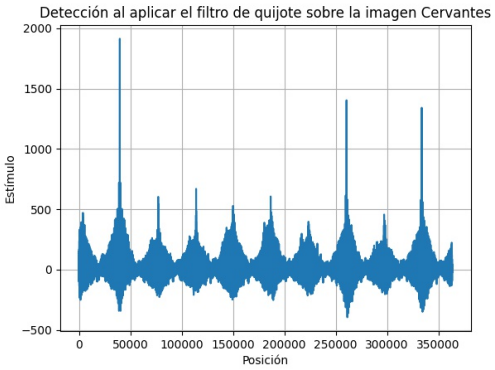
(e) Dog



(f) Cat



(g) No

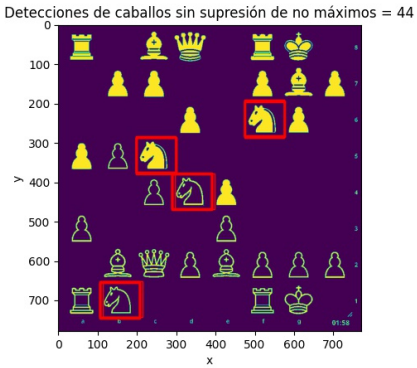


(h) Quijote

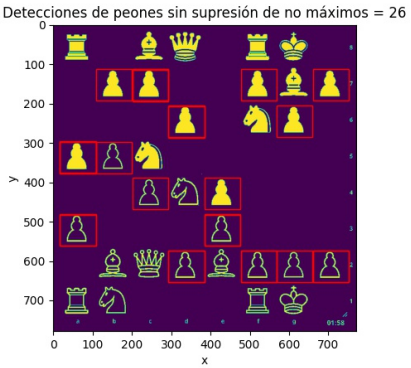
Figura 5: Patrones extraídos de las imágenes

Tabla 1: Valores de umbrales para la detección de los patrones por imagen

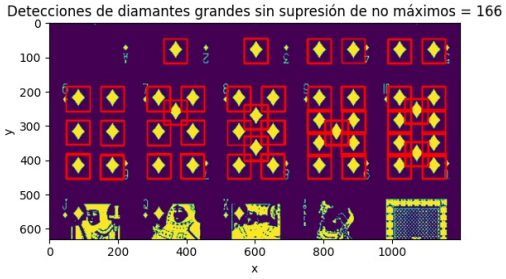
Patrón	Porcentaje del máximo (%)	Valor Threshold (aproximado)
Caballo	0.145	217
Peón	0.5	315
Diamantes en çartas"	0.3	141
Diamantes en "seis"	0.5	390
Dog	0.8	632
Cat	0.8	472
No	0.68	381
Quijote	0.6	1140



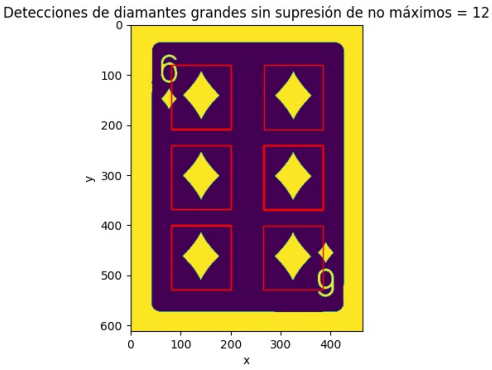
(a) Caballo



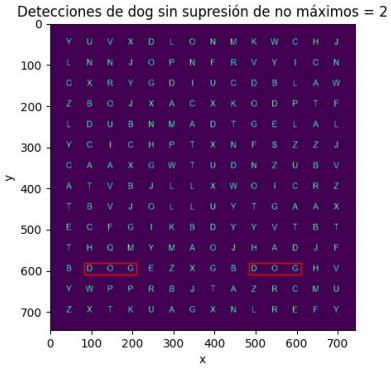
(b) Peón



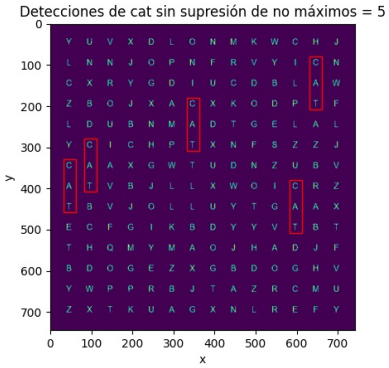
(c) Diamantes en "cartas"



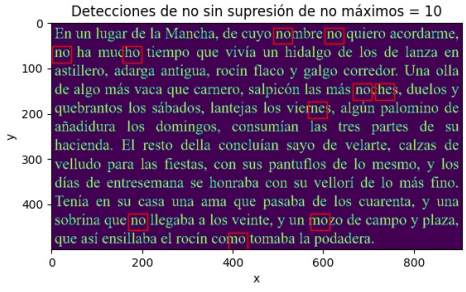
(d) Diamantes en "seis"



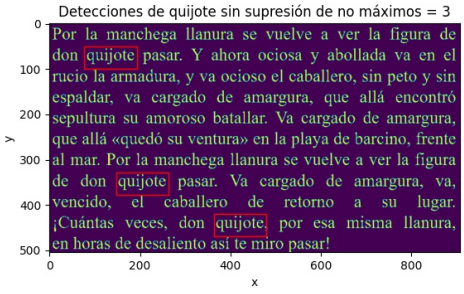
(e) Dog



(f) Cat

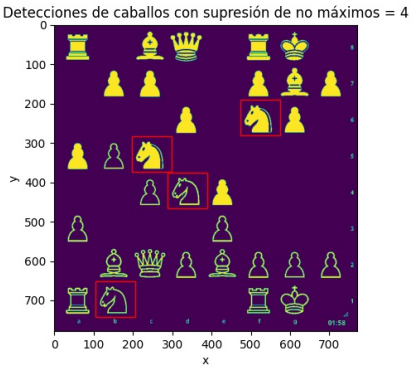


(g) No

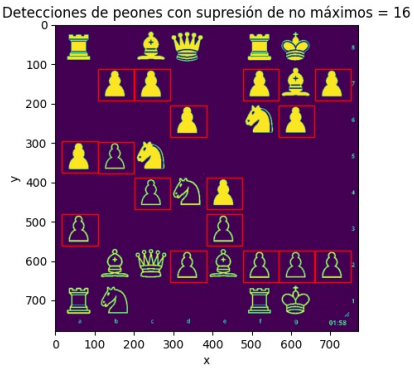


(h) Quijote

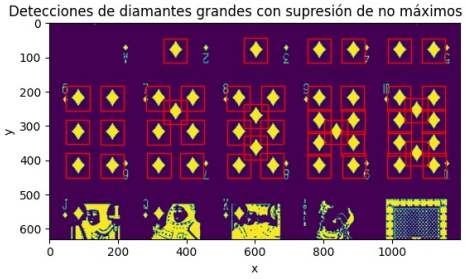
Figura 6: Patrones extraídos de las imágenes



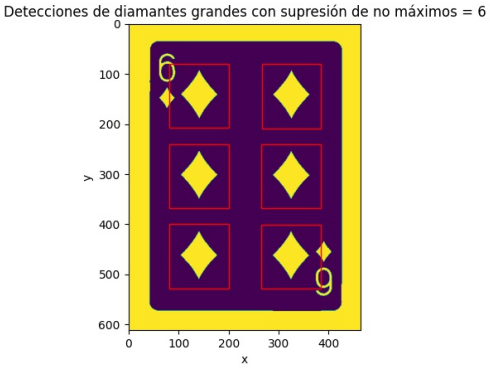
(a) Caballo



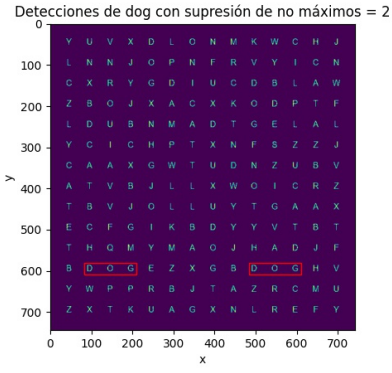
(b) Peón



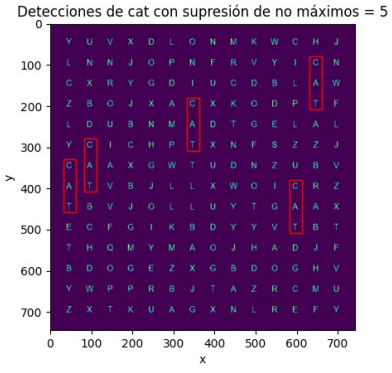
(c) Diamantes en "cartas"



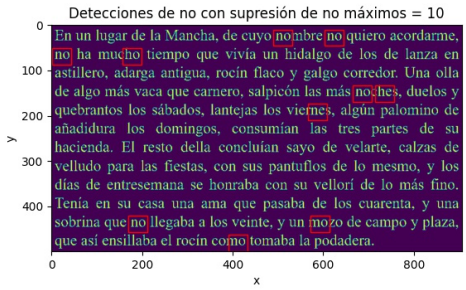
(d) Diamantes en "seis"



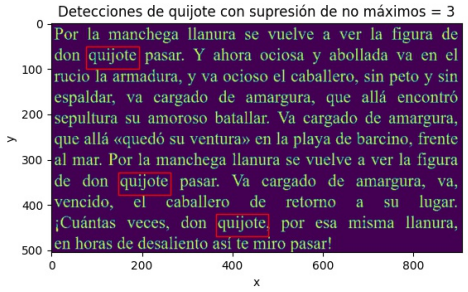
(e) Dog



(f) Cat



(g) No



(h) Quijote

Figura 7: Patrones extraídos de las imágenes