Informe Aplicación Solucionador De Sudokus

Juan Alejandro Bernal, Orlando Hernandez

February 11, 2020

1 Problema a resolver

1.1 Introducción

El sudoku es un juego matemático inventado en 1970, recibió muy poca atención del publico en sus inicios, pero en la decada de 1984 recibió mucha atención en japón, finalmente en el 2005 tomo importancia internacional ya que empezaron a ser publicados en periódicos.

1.2 Descripción

Un sudoku es una cuadricula de 9 x 9 dividida en subcuadrillas de 3 x 3, en total hay 81 casillas. El objetivo es rellenar las 81 casillas, teniendo en cuenta que algunas casillas ya están rellenadas de ante mano. La solución de un sudoku siempre es un cuadrado latino con la única diferencia que en cada subcuadrillas no hayan nmeros repetidos.

1.3 Dificultad

La dificultad de un sudoku depende del número de soluciones posibles que hayan dependiendo de los valores que están ya dispuestos en las celdas. El matemático Gary McGuire ha demostrado que el mínimo número de cifras para conseguir un sudoku con una única solución es de 17 cifras. Esto se da por que dadas las condiciones del cuadrado latino y que los números no se pueden repetir en subcuadrillas, las posibilidades que existen para poner números en lugares equivocados es muy alta.

2 Solución

Por la alta dificultad que pueden tener algunos sudokus hemos diseñado una aplicación que se encarga de recibir un sudoku y devolver una solucin. Haciendo uso de tcnicas de reconocimiento de patrones y visin artificial. Con la idea de hacerlo más sencillo de usar se creo una aplicación móvil por medio de Flutter.

2.1 Pasos para reproducir la solución

2.1.1 instalacion de Dependencias

2.1.2 Manual

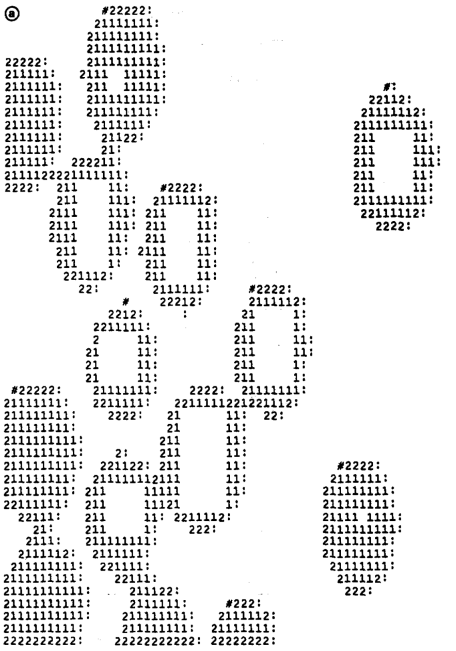
2.2 Explicacin tcnica

En esta sección se describirán las técnicas usadas para construir la aplicación.

2.2.1 Visión por computadora

El algoritmo utilizado para encontrar el contorno de las imágenes sigue el contorno binario haciendo un análisis topologico de la imagen digital, segn [**?**] se sigue un patrón de color o intensidad la cual la computadora sigue como patrón de la siguiente manera:

Figure 1: Ejemplo de imágenes en visión artificial



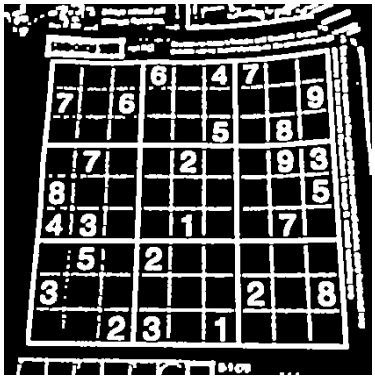


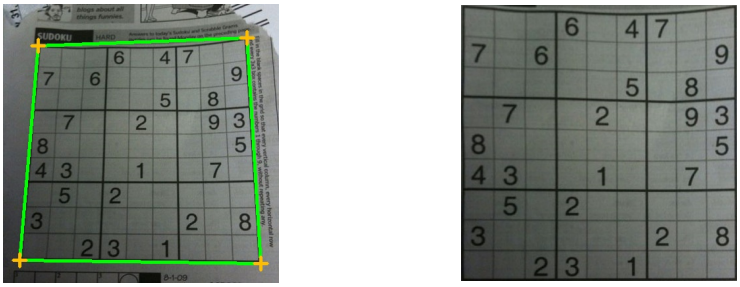
Figure 2: Filtro aplicado a un sudoku

Como se puede observar en la imagen los contornos empiezan en el smbolo # y continúan donde están marcados los :. El uso de filtros hace que este algoritmo reconozca estos patrones de manera más sencilla, en especial el pasar la imagen a un estado binario de forma que quede así:

Haciendo uso de esto podemos conseguir todos los contornos de una imagen de sudoku, al conseguir los contornos los procesamos haciendo uso del algoritmo de Ramer-Douglas Peucker [**?**] la cual (en resumen) aproxima los puntos a lineas, las cuales consideraremos como lados, el cual nos permite encontrar todos los cuadrados del sudoku. Pero como primera instancia debemos de encontrar el cuadrado mas grande, es decir el mas externo.

```
for contour in contours:
        perimeter = cv2.arcLength(contour,True) # largo del
    contour
        approx = cv2.approxPolyDP(contour, 0.02*perimeter , True)
    # funcion Ramer, 0.02 es una constante
        if len(approx) == 4: # si es un cuadrado
            if perimeter > maxPerimeter: # para conseguir el
    contorno mas grande
                maxPerimeter = perimeter
                big_square = approx
    return big_square
```

Una vez encontrado el contorno más grande nos queda un arreglo con 4 tuplas las cuales representan los puntos en las esquinas, los cuales usaremos para poder hacer una transformacin de perspectiva a la imagen, ya que como las fotos pueden ser tomadas en ángulos extraños, se necesita arreglar la perspectiva para que ayude a la detección de números después.

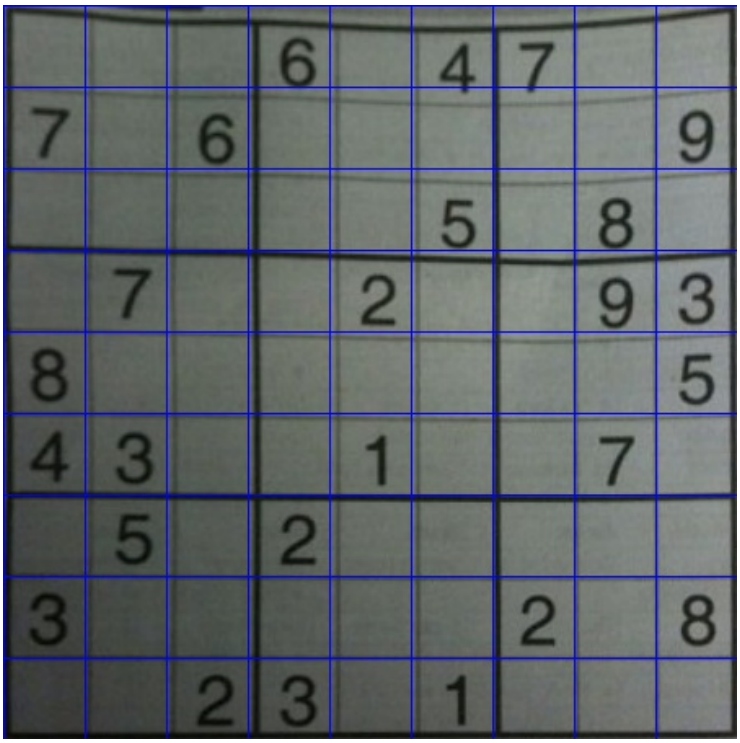


(a) Transformación (b) Esquinas encontradas

Figure 3: Procesamiento de imagen

Finalmente dividimos el sudoku en sus lineas interiores para poder conseguir un arreglo de subcuadros, los cuales usaremos para hacer el reconocimiento de dígitos

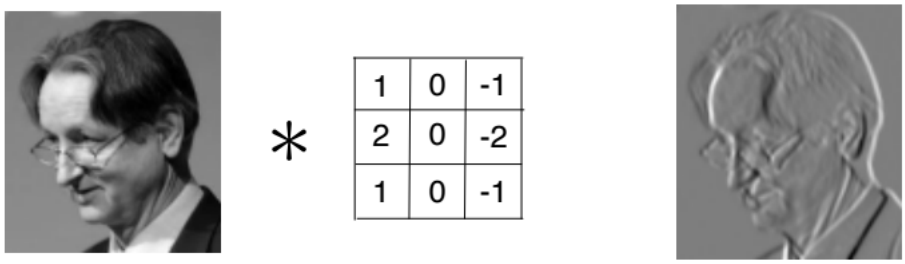
Figure 4: Arreglo de subcuadros



2.2.2 Reconocimiento de dígitos

A continuación, se describirá la solución propuesta para identificar los diferentes números encontrados en el tablero del sudoku. El método para la identificación de los números se logra por medio de un modelo que es entrenado por una Red Neuronal Convolucional utilizando el dataset Chars74k.
Una red neuronal convolucional es una red que es frecuentemente usada para procesar imágenes aprendiendo de relaciones de entrada-salida donde la entrada es una imagen. La operación básica de una CNN (Red Neuronal Convolucional por sus siglas en inglés) es la convolución la cual consiste en filtrar una imagen usando una máscara.

Figure 5: Ejemplo de una convolución

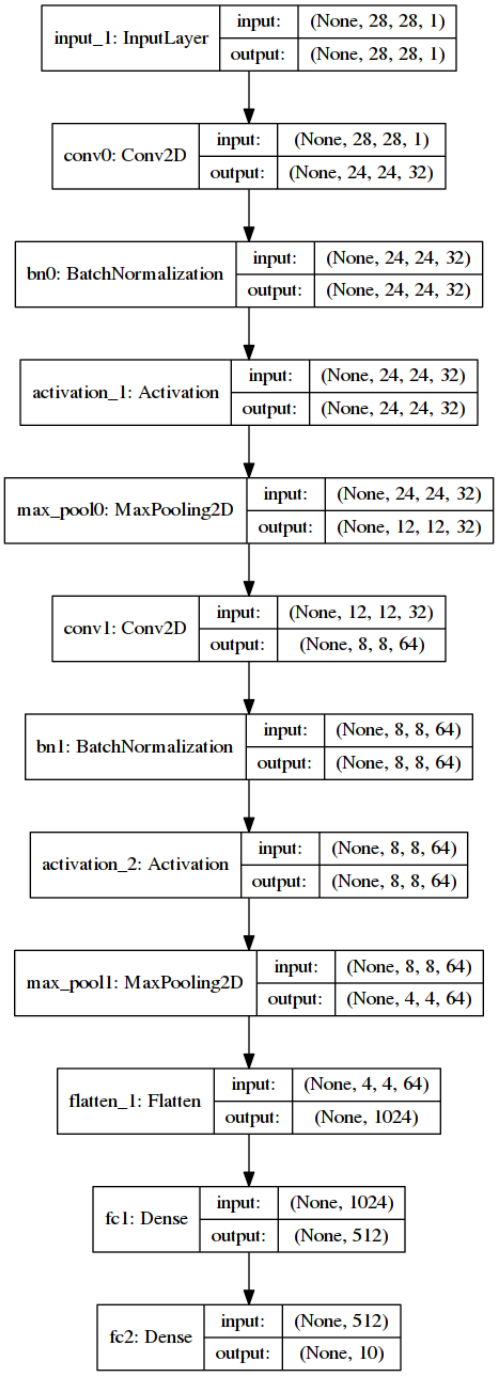


Las 10160 imágenes para entrenar a la red fueron tomadas del dataset Chars74K. Estas imágenes fueron divididas en el 98% para el entrenamiento de la red y el restante para realizar la validación del modelo resultante. Cada una de estas imágenes es de fondo negro y contiene un dígito del 0 al 9 en color blanco; su tamaño es de 28x28 pixeles.

Figure 6: Imagen del dataset Chars74K

En 1 se puede observar la arquitectura de la red neuronal propuesta para solucionar el problema.

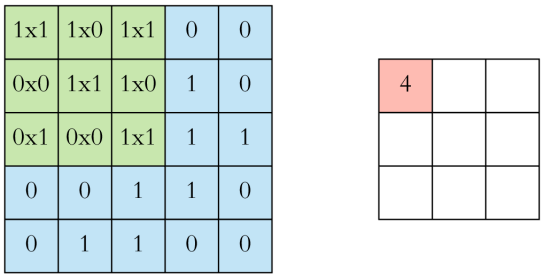
Figure 7: Arquitectura de la red neuronal construida



La red neuronal esta compuesta de diferentes capas cada una con un objetivo específico.

Capa convolucional La red construida se compone de dos de éstas capas, las cuales por medio de filtros se encargan de recorrer las imágenes en busca de características generando un Feature Map.

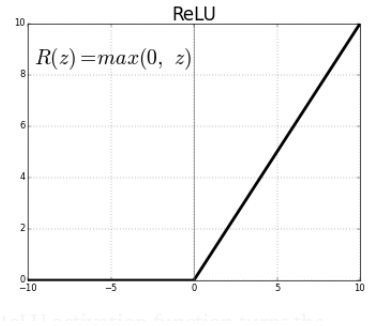
Figure 8: Feature map



Capa de normalización Para esta capa se usa específicamente la implementación de BatchNormalization. Esta capa mantiene los valores que pasan por medio de la red.

Capa de activación La red posee dos capas de este tipo cada una con la función ReLU.

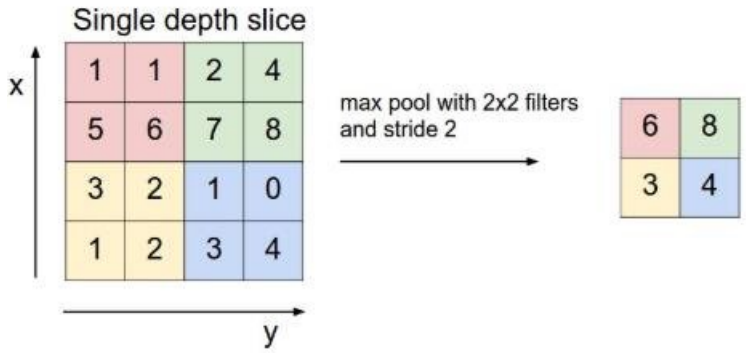
Figure 9: Función ReLU



Capa de pooling Este tipo de capa se encarga de resumir las catarecterísticas obtenidos en los Feature Map generados en la convolución. La configuración de las capas de pooling recibe como parámetros la dimensión de los filtros y el

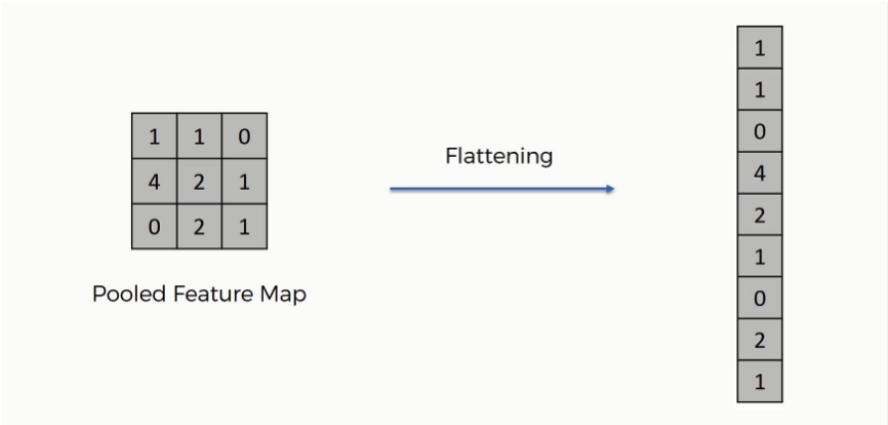
stride que define como va a ser recorrido, en este caso, las dimensiones fueron 2x2 y el paso o zancada fue de 1.

Figure 10: Comportamiento capa de pooling



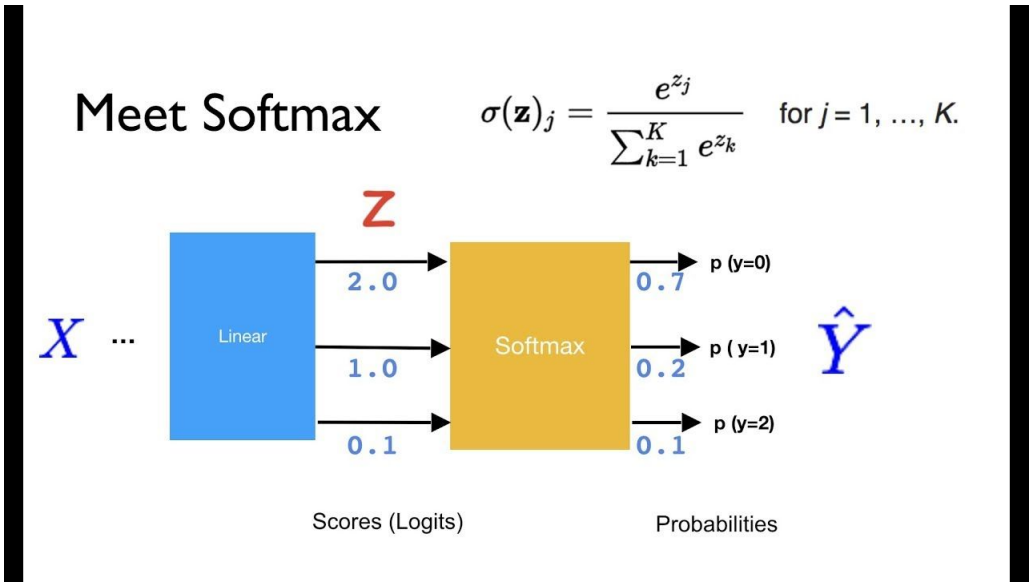
Capa de aplanamiento La capa Flatten se encarga de coger cada una de las matrices y convertirlas en vectores por esto es conocida como capa de aplanamiento.

Figure 11: Comportamiento capa de aplanamiento



Capa completamente conectada (Dense) Este tipo de capa es tal cual la de una red neuronal básica. En la arquitectura de la red neuronal propuesta se encuentran dos capas de este tipo: Una con 1024 neuronas y función de activación ReLU y la otra con 10 salidas y función de activación softmax.

Figure 12: Función Softmax



Los pesos serán optimizados usando el algoritmo ADAM el cual es un método para optimización estocástica. La función de perdida de entropía cruzada (categorical cross entropy) y las métrica será accuracy (o taza de acierto). El entrenamiento fue por medio de lotes, es decir, se actualizaban los parámetros internos del modelo cada 83 muestras. La red neuronal fue entrenada por 10 épocas.

La red neuronal fue construida usando la libreria Keras la cual trabaja sobre Tensorflow y una vez obtenido el modelo, se puso a disposición por medio de una API REST programada en el lenguaje python bajo el framework Falcon.

2.2.3 Backtracking

Una vez identificados los números en el tablero para hallar una solución al sudoku se uso Backtracking el cual es una técnica de programación para hacer una búsqueda a través de todas las configuraciones posibles dentro de un espacio de búsqueda. El algoritmo fue programado en el lenguaje de programación Dart.

3 Resultados

Finalmente, se logra la construcción de una aplicación móvil la cual puede reconocer la disposición de números en el tablero del sudoku. Como trabajo a futuro se podría mejorar si se reconocen las celdas donde no hay un dígito, ya que, actualmente, esta identificación se debe hacer de forma manual.