Sistema de deteccion de rostros con el método de Redes Convolucionales

February 2021

Resumen

Se presentarán resultados de la implementación de método de Redes Convolucionales para la detección de rostros humanos.

1. Introducción

1.1. Detección y reconocimiento facial

La detección facial es el proceso en el que el software determina, mediante algoritmos, si hay rostros humanos en una foto o vídeo. No determina la identidad de una persona, tan solo determina si hay alguna cara. Por otro lado el sistema de reconocimiento facial es una aplicación que se encarga de identificar automáticamente a una persona, el cual realiza un análisis de las características faciales del usuario adquiridas mediante una imagen y comparándolas con una base de datos.

1.2. Redes Neuronales convoluciones

Siguiendo la documentación presentada en [3], se dice que una red neuronal convolución replica en lo posible el comportamiento de la corteza visual, en particular la corteza visual primaria V1, en dónde se lleva la tarea de detectar ciertas características, por ejemplo algunas células detectan bordes, y mas adelante definiremos el concepto de filtro que permite "dejar pasarçierta información de la imagen.
Existen células simples que tienen regiones exítadoras e inhibidoras, que en conjunto forman patrones para detectar un padrón simple, combinando la recepción de información. Existen células complejas que tienen una orientación sobre las cuales son sensibles, pero en este tipo de células no tienen sensibilidad a la posición.

Con lo anterior nos damos cuenta que las investigaciones realizados en visión humana han ayudado a crear arquitecturas de redes que se adapten a las necesidades de detección, segmentación y clasificación de imágenes.

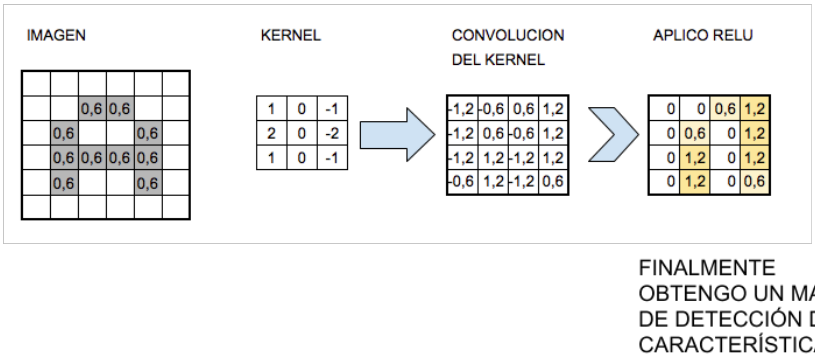

Figura 1: Aplicar una convolución y la función RELU https://www.juanbarrios.com/redes-neurales-convolucionales/

Recordemos que las imágenes es una representación en un arreglo de la información capturada por un sensor de cámara, esa información si se intenta modelar para clasificar la imagen capturada usando una red neuronal convencional, por ejemplo una fully conected layer (FCL), ocurren dos cosas importantes [**?**]:

- Si se reordenan los elementos de la matriz que representa la imagen la precisión obtenida será la misma que originalmente se tenia antes del nuevo ordenamiento.
- No considera simetría, es decir si se tiene un objeto en una esquina y otro en la esquina contraria se tendría pesos distintos para cada objeto

Para solucionar los problemas anteriores surgen las redes convoluciones que permite.

En el proceso de creación de la arquitectura de la red neuronal convolucional se siguen estos pasos:

1. Leer la imagen con una capa de entrada que esperará diferentes canales, por ejemplo una imagen a color tiene 3 canales: rojo , verde y azul.
2. Opcional: Si se desea rescatar la información presente en las orillas de la imagen se agregan ceros en las orillas de la imagen, esto para que al realizar la convolución no se pierda la información de las orillas
3. Sobre esa matriz de matrices, que dependerá del número de canales se procede a aplicar un filtro o kernel, este permite pasar ciertas estructuras presentes en la imagen, se pueden tener de resultado el número de matrices que se quiera, esto es cada matriz dejará pasar la información necesaria, para saber como realizar el desplazamiento sobre la matriz, es decir los

saltos que se hará la hacer la convolución se define un parámetro conocido como stride Figura 1

4. Muestreo sobre la matriz resultante anterior, debido a la complejidad de crear cada vez mas canales resultado de la convolución se realiza un submuestreo, regularmente usando maxpoling que permite reducir el tamaño de la imagen. quitar el componente lineal hasta este momento se usa una función de activación, tomando el resultado de los filtros, por temas de costo computacional se emplea RELU

5. El resultado de lo anterior se le puede aplicar todo desde el paso 1, esto lo que va a ha provocar es extraer ciertos patrones presentes en la imagen.

6. Con las convoluciones ya realizadas, el resultado después de aplicar el último submuestreo se pasa a la capa de salida que nos permitirá calcular probabilidades de que esa imagen sea de algún clasificación dada

2. Implementación

Para la implementación se opto por utilizar el lenguaje de programación Julia, partiendo del hecho que es un lenguaje de programación rápido y con una sintaxis sencilla. Para eso se uso el data set presente en **??**, que contiene imágenes de 29x29 píxeles separadas en carpeta de entrenamiento y validación, y dentro de cada carpeta se encuentra dos, las cuales contiene imágenes de cara y no cara.

Con el dataset seleccionado se plantea el siguiente problema:
¿Es posible utilizar redes neuronales para detectar cuando hay una cara y cuando no? Para eso se plantea solucionarlo con redes convolucionales. El flujo de la aplicación sigue los siguientes pasos siguiendo el tutorial de [3]

1. Instalar las dependencias necesarias

2. Cargar las dependencias

3. Cargar las imágenes de entrenamiento y test

4. Se crearon dos ejercicio, ya que el primero no consideraba un **reordenamiento** de las imágenes, y al entrenar no mejoraba la exactitud, por eso se agrego un paso para seleccionar de manera aleatoria las imágenes.

5. Crear la arquitectura de la red, la función de perdida que en esta ocasión se usa cross entropy

6. Crear el optimizador, en este ejercicio se uso el de Adam

7. Se entrena la red

8. se calculan las probabilidades y se seleccionan la clase (cara, no cara) con mayor probabilidad

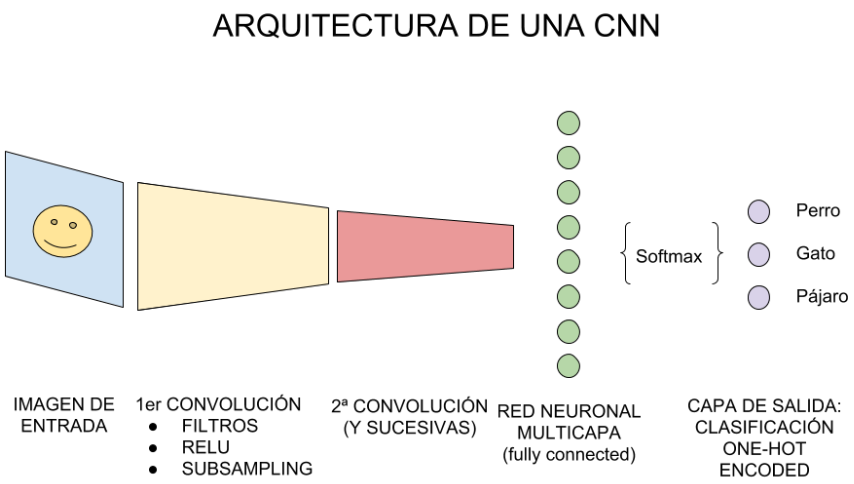

Figura 2: Arquitectura completa de una red convolucional

9. Se calcular las métricas de desempeño del modelo: precisión, re-call, exactitud

La arquitectura implementada en la red convolucional tiene la siguiente arquitectura [1], Figura 2 2:

1. Capa de entrada
2. Agregar zero padding
3. Realizar una convolución con un filtro de 3x3, que crea 16 canales
4. Realizar submuestreo con maxpoling, tomando 4 píxeles y reduciendolo a la mitad
5. Realizar una convolución con un filtro de 3x3, que toma 16 canales y crea 16 canales
6. Realizar submuestreo con maxpoling, tomando 4 píxeles y reduciendolo a la mitad
7. Realizar una capa de salida que toma 800 parámetros y los reduce a 2 (es una cara o no)

3. Resultados

Al implementar este modelo, con el reordenamiento tenemos el siguiente resultado:
La exactitud del modelo fue: 0.97
La precisión del modelo fue: 0.15
El recall del modelo fue: 0.34

Mientras que en una estrategía sin ordenar los resultados fueron:
La exactitud del modelo fue: 0.97
La precisión del modelo fue: 0.0
El recall del modelo fue: NaN

Referencias

[1] Redes Neuronales Convolucionales
`https://www.juanbarrios.com/redes-neurales-convolucionales/`

[2] Redes Neuronales Convolucionales
`https://medium.com/@jayeshbahire/cnn-implementation-using-julia-defe90d86eb2`

[3] Redes Neuronales Convolucionales
`https://towardsdatascience.com/a-primer-on-computer-vision-with-julia-2c7068a35b32`