Video processing system for object recognition using convolutional neural networks (CNN)

1st Jorge Luis Corredor Cely
Facultad de ingeniería electrónica
Universidad Santo Tomás
Bogotá D.C, Colombia
Jorgecorredor@usantotomas.edu.co

2nd Juan Camilo Garcia Carrillo
Facultad de ingeniería electrónica
Universidad Santo Tomás
Bogotá D.C, Colombia
Juancgarcia@usantotomas.edu.co

3rd Viviana Alejandra Mora Diaz
Facultad de ingeniería electrónica
Universidad Santo Tomás
Bogotá D.C, Colombia
Vivianamora@usantotomas.edu.co

***Abstract*—The following document describes the procedure performed in the development of video processing for object recognition by means of convolutional neural networks (CNN). For the aforementioned, a Python code was used using libraries such as TensorFlow, Open CV, Numpy, and Keras, allowing to process each of the video frames.**

***Index Terms*—CNN, Python, Video Processing.**

I. Introducción

En su mayoría los métodos de aprendizaje utilizados en Deep Learning, comprenden las arquitecturas de redes neuronales. Estos modelos se entrenan a partir del uso de determinados conjuntos de datos etiquetados con estructuras (Redes Neuronales) que aprenden a partir de dichos datos, sin la necesidad de hacer una extracción manual de estos conjuntos.

Una de las arquitecturas de redes neuronales más conocidas, corresponde a la red neuronal convencional CNN, la cual se encarga de implementar capas convolucionales 2D a partir de las características aprendidas con los datos de entrada. Su principio de funcionamiento se basa en la extracción de características de un conjunto de imágenes, donde aquellas relevantes no se entrenan con anterioridad, sino que se aprenden mientras la red entrena a partir de una colección de imágenes (Dataset). Es por esto que, de esta forma, esta estructura ayuda de gran forma a implementaciones como lo son la detección de objetos. Teniendo en cuenta que estas pueden implementar cientos de capas neuronales ocultas, cada una de ellas asignada a una tarea específica, concluyendo con una estructura bastante precisa.

A partir de esto, se implementa una red neuronal convolucional, que se encargue de la detección de vehículos. Esta será entrenada a partir de recortes de vehículos obtenidos de los frames resultantes de la descomposición de un video en 360° en tránsito vehicular.

II. Objetivos

A. General

Procesar un video con formato 360° con el fin de generar uno nuevo donde se reconozcan los automóviles presentes.

B. Específicos

- Descomponer el video de prueba en frames por medio de un programa en Python, con el fin de procesarlos posteriormente de manera individual.
- Extraer las secciones de cada uno de los frames en donde se localizan los automóviles y en donde no se encuentra presencia de ellos, con el fin de generar una base de datos.
- Entrenar una red neuronal convolucional utilizando la base de datos generada para reconocer los vehículos.
- Procesar cada uno de los frames usando la CNN entrenada e indicar en cada uno las secciones donde se identifican los vehículos.

III. Desarrollo y procedimiento

A. Descomposición en Frames

Debido a que en una primera instancia se realiza un procesamiento de imágenes, más no videos, se necesita realizar la descomposición de este contenido multimedia en una secuencia de imágenes que describan dicha reproducción. Para esto se realizó el siguiente fragmento de código que realizara esta tarea:

```
import cv2

capture = cv2.VideoCapture('Home2School_1.mp4')
cont = 0
path = '.\\Frames\\'

while (capture.isOpened()):
    ret, frame = capture.read()
    if (ret == True):
        cv2.imwrite(path + 'IMG_\%04d.jpg' \% cont,
                    f
                    )
        cont += 1
        if (cv2.waitKey(1) == ord('s')):
            break
    else:
        break

capture.release()
cv2.destroyAllWindows()
```

Una vez ejecutado el código anterior, el video tomado se descompondrá en 5384 imágenes, de las cuales se extraerá los objetos (Colección de imágenes) con los que se realizará el entrenamiento posterior.

B. Recortes de imágenes

A partir de los 5384 frames obtenidos, se procede a realizar la extracción de la base de datos que será usada en el entrenamiento de la CNN. Para esto, se genera una colección de imágenes correspondiente a los carros a identificar y otra colección de objetos que no pertenecen al conjunto anterior.

Con el fin de facilitar la extracción de las imágenes, se realiza el siguiente código en Python que permite recortar múltiples muestras de cada uno de los frames. Siendo especialmente eficaz, cuando los automóviles se encuentran estáticos:

```
import cv2

path = '.\\Frames\\'
path2 = '.\\Carros\\'
path3 = '.\\Nocarros\\'
cont = 0

while(cont<=720):
    image = cv2.imread(path+'IMG_%04d.jpg' % cont)
    imageout = image[450:550+150,200:600+200]
    cv2.imwrite(path2 + 'IMG_%04d.jpg' % cont,...
    ...imageout)
    cont += 1
    image = cv2.imread(path+'IMG_%04d.jpg' % cont)
    imageout = image[550:550+200,1120:1200+280]
    cv2.imwrite(path2 + 'IMG_%04d.jpg' % cont,...
    ...imageout)
    cont += 1
    image = cv2.imread(path+'IMG_%04d.jpg' % cont)
    imageout = image[505:450+110,50:50+50]
    cv2.imwrite(path2 + 'IMG_%04d.jpg' % cont,...
    ...imageout)
    cont += 1
    image = cv2.imread(path+'IMG_%04d.jpg' % cont)
    imageout = image[505:450+110,100:100+50]
    cv2.imwrite(path2 + 'IMG_%04d.jpg' % cont,...
    ...imageout)
    cont += 1
    image = cv2.imread(path+'IMG_%04d.jpg' % cont)
    imageout = image[515:515+70,1800:1700+250]
    cv2.imwrite(path2 + 'IMG_%04d.jpg' % cont,...
    ...imageout)
    cont += 1
```

El código anterior se encarga de generar recortes en diferentes secciones en donde se sabe, se encuentran carros estáticos en una cantidad específica de frames.De igual modo, se genera un código encargado de la extracción de las imágenes que serán agregadas a la colección de "No-carros", con delimitaciones de recorte en donde se sabe, no se presentan automóviles en los frames a procesar.

En el caso de los frames que presentan un desplazamiento significativo de los automóviles, se procede a realizar una extracción manual por medio de recortes de los objetos de interés.

Es importante tener en cuenta que el tamaño de las colecciones de "Carros" y "No-carros" debe ser lo más similar posible, de modo que se tomó una cantidad de 1481 muestras de cada una de las categorías.

C. Red neuronal convolucional

Para la implementación de la red neuronal convolucional se deben definir algunos parámetros de la misma. Para las capas convolucionales se definen en un tamaño de entrada de (128,128,3), siendo los primeros dos parámetros correspondientes a las dimensiones de las imágenes, y el tercero indicando el formato RGB. Como segunda etapa se tiene la capa de Pooling, la cual se tiene un parámetro de Pool Size establecido en (2,2). Como tercera capa, se tiene la capa de Flatten, sin ningún parámetro establecido. Por último, se tiene la capa de Full Connection, la cual tiene cuatro capas, las primeras dos capas ocultas de un tamaño de 512, la siguiente es de 128, y la última tiene una dimensión de 1, esto debido a que es la salida de nuestra red.

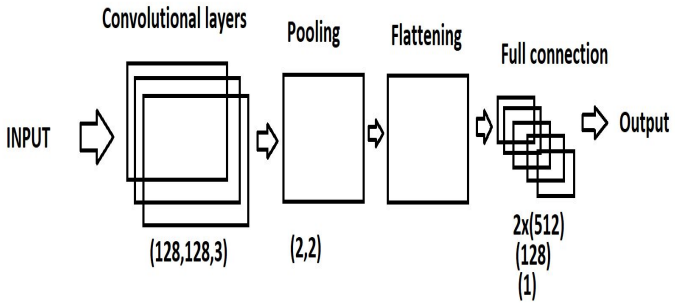

Fig. 1. Modelo de red neuronal convolucional

Con la red neuronal convolucional ya implementada se procede a entrenarla, por lo que se le cargan 2962 imágenes de entrenamiento, en donde, la mitad corresponde a imágenes de carros, y la otra mitad a imágenes de no carros, ambos paquetes extraídos anteriormente del vídeo. Además de esto se cargan 200 imágenes de prueba, que sirven para realizar un test en el entrenamiento.

Fig. 2. Salida del entrenamiento

Para corroborar el clasificador, se pone a prueba con la siguiente imagen.

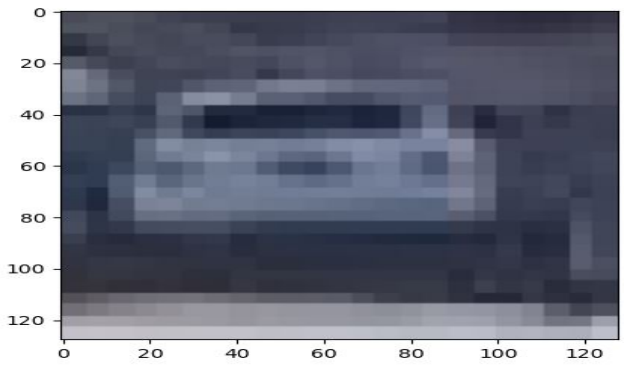

Fig. 3. Imagen de prueba

De la anterior imagen se obtiene la siguiente salida:

```
[[0.01040477]]
{'Carros': 0, 'Nocarros': 1}
Carro
```

Fig. 4. Resultado clasificador

Como se puede observar en la anterior figura en sistema clasifica de manera correcta la imagen.

D. Clasificación de vehículos en las imágenes

Para esta sección, se debe realizar un recorrido por la imagen en la cual se van a clasificar los vehículos. Lo primero que se realiza es la selección del área donde se encuentran los vehículos.

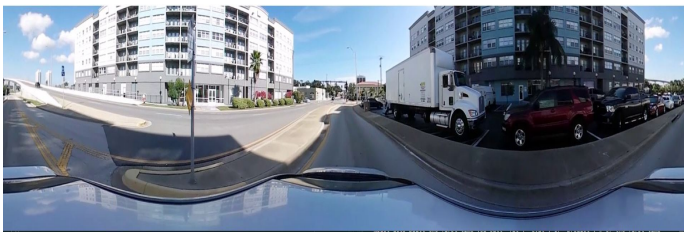

Fig. 5. Área donde se encuentran los vehículos

A partir de la anterior área se realizan recortes de la imagen para luego clasificarlas y así poder encontrar las posiciones donde se encuentran los vehículos. Estos recortes tienen unas dimensiones que corresponden al tamaño del vehículo más grande encontrado, un medio de este tamaño, un cuarto de este tamaño, y un octavo de este tamaño.

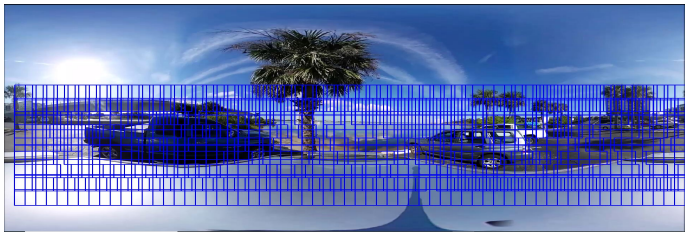

Fig. 6. Imagen con todos los recortes que se realizan

El recorrido anterior se le realiza a todos los Frames extraídos previamente del video, y solo se dibujan los rectángulos en los lugares donde la red detecte que hay un vehículo.

A continuación se muestran algunas imágenes de salida.

Fig. 7. Imagen de salida 1673

Fig. 8. Imagen de salida 2571

Fig. 9. Imagen de salida 5051

E. Reconstrucción del video

Por último se unen todas las imágenes de salida en un arreglo y se reconstruye en un video. Esto se logra a partir de la ejecución del siguiente código.

```
import cv2
path = '.\\Img_video\\'

img_array = []
x=0
while(x<=5383):
    path = path+'IMG_%04d.jpg' % x
    img = cv2.imread(path)
    img_array.append(img)

height, width  = img.shape[:2]

video = cv2.VideoWriter('Videofinal.mp4',
cv2.VideoWriter_fourcc(*'mp4v'), 5, (width,height))

for i in range(len(img_array)):
    video.write(img_array[i])

video.release()
```

IV. Conclusiones

- Para que el entrenamiento se realice como corresponde, las imágenes de entrenamiento, test y de clasificación, deben tener las mismas dimensiones a la primera capa de convolución.
- Como se pudo observar, la clasificación de imágenes no se realizó correctamente, por lo que se puede prever su falla en el hecho de que no se puede tener 200 entradas en un modelo que solo cuenta con 8 de ellas.