Detección de automóviles a partir de imágenes aéreas

Luis Vasquez
Facultad de Ciencias
Universidad Nacional de Ingeniería
Lima, Perú
luis.vasquez@uni.pe

***Resumen*—Este documento presenta las disposiciones generales y estado del arte del proceso a emplear para que, mediante el uso de alguna arquitectura de redes neuronales convolucionales, se logre detectar autómoviles a partir de fotografías capturadas desde una vista aérea. Se expondrá el problema inicial y se explicará la solución aplicada por los autores de los artículos consultados. Finalmente se definirá el diseño arquitectónico apropiado para llevar a cabo la investigación.**

***Index Terms*—Deep, Learning, CNN, Detection, Computer vision**

I. Introducción

La detección de automóviles presenta aplicaciones comerciales muy diversas, desde autómoviles autómatas hasta control de tráfico urbano, creando herramientas de vigilancia, rastreo y control en muchos contextos. El problema computacional para lograr esta detección radica en la capacidad de implementación de una red neuronal convolucional que no solo clasifique que lo ingresado es un automóvil o no, si no que a demás ubique dicho positivo (de existir) en la imágen. El lenguaje python se integra a esta investigación al usar las herramientas de tensorflow y keras, librerías especializadas en deep learning. El presente curso de ciencia de datos apoya estas herramientas con la presentación de metodologías de preparación y procesamiento de datos, facilitando documentación alusiva a los métodos de regresión y clasificación que se utilizarán durante el entrenamiento de la red. Se tiene como objetivo general el detectar y ubicar los automóviles en cualquier buffer de imágenes ingresado. Se tiene como objetivo específico el poder hacer esto de manera óptima, modificando meta e hiperparámetros para hacer el entrenamiento eficiente. Se tiene como objetivo personal el lograr entender la teoría matemática y computacional detrás de las operaciones comúnes que se emplean dentro de las redes neuronales convolucionales, sin ninguna noción previa.

II. Estado del arte

II-A. Redes Neuronales Convolucionales

II-A1. Convolución: La convolución es un proceso recurrente en este tipo de red neuronal. Se aplica a la señal ingresada para detectar ciertos patrones y reducir su dimensión (o mantenerla constante) a fin de que se maneje la información de manera reducida sin perder características importantes. Se debe entender la convolución aplicada al procesamiento de imágenes como el deslizamiento paramétrico de una ventana N-Dimensional llamada **filtro**, sobre la señal que se ingrese.

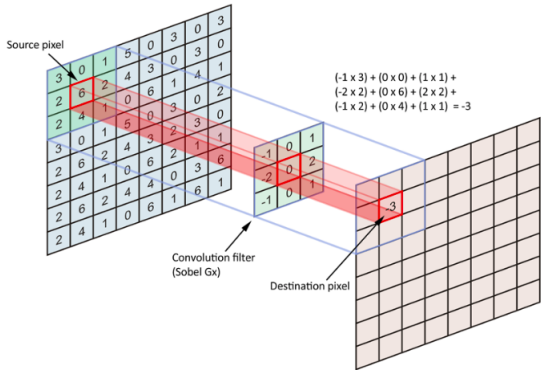

Figura 1. Ejemplo de convolución con una señal bidimension (matriz). Se observa la correspondencia entre cada submatriz de la entrada con el filtro aplicado según la dimensión del mismo.

La convolución es la operación más compleja por la que pasa la señal al ingresarla en la red, dado que posee muchos hiperparámetros que modifican significativamente su comportamiento y producto final:

- **Padding:** Se aplica un margen (borde) a la señal original en todos sus canales (cuyas celdas usualmente presentan ceros) antes de realizar la convolución original; gracias a esto información que no se convolucionaría muchas veces (como la de las esquinas de una matriz similar a la de la Fig. 1) se operarán más veces. Cuando una convolución fija un padding específico para que la dimensión del output sea la misma que la del input, se denomina **Same Convolution**; y si no existe padding alguno se denomina **Valid Convolution.**
- **Striding:** Se varía la longitud del paso del filtro a través de la entrada. Para un stride de 2 unidades, por ejemplo, el filtro recorrerá la señal de dos en dos de izquierda a derecha y de arriba hacia abajo. Si debido a esto el filtro intenta convolucionar una región fuera de la matriz de entrada, dicha operación no se considerará.

II-A2. Canales: Cuando se trabaja con señales que necesiten varias dimensiones para representarse (como imágenes RGB) hablamos de canales de información. Para este tipo

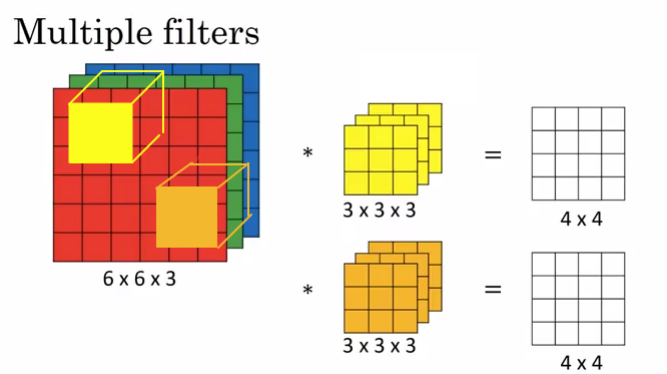

Figura 2. Aplicación de 2 filtros a una matriz tridimensional (imágen RGB). El primer filtro (amarillo) se convoluciona a lo largo de los tres canales, opera y obtiene una matriz de convolucion. Similarmente con el segundo filtro (naranja). Ambas resultantes se convierten en una nueva señal de dos canales.

de señales usamos filtros que tengan la misma profundidad (número de señales) que la data, generando así una convolución bidimensional, teniendo como consecuencia la pérdida de la profundidad. Para recuperar esta profundidad solo bastará con aplicar varios filtros, y cada uno de estos generará un nuevo canal en la salida. (Fig. 2)

En general, luego de un proceso de convolución en una capa l, la dimensión final del resultado será de dimensiones

$$n_H^{[l]} \text{ x } n_W^{[l]}$$

Donde:

- $n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} \right\rfloor + 1$
- $n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} \right\rfloor + 1$
- $f^{[l]}$: Dimensión del filtro de la capa l
- $p^{[l]}$: Factor de pooling de la capa l
- $s^{[l]}$: Longitud de stripping de la capa l

II-A3. Pooling: Si bien la redimensión del resultado era un efecto secundario de la convolución, se requiere controlar esta contracción de la señal. El pooling logra esto recorriendo la matriz con un filtro simple de selección. El pooling más usado es el **Max Pooling** que toma el valor máximo revisado por el filtro (Fig. 3)

II-B. Stochastic Gradient Descent

Cuando se utilizan redes neuronales sencillas como perceptrones (multicapa o no) se suele emplear el método de *gradiente descendente* para actualizar la matriz de pesos durante el aprendizaje, usando:

$$W \leftarrow W - \alpha \nabla_W J(W)$$

La función de costo J era calculada usando todos los datos predecidos y esperados agrupados en batchs; sin embargo

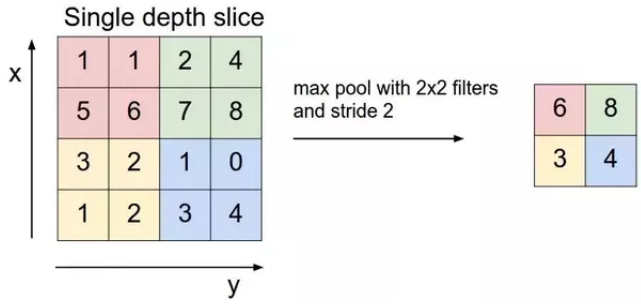

Figura 3. Max pooling con un filtro de $f^{[l]} = 2$

aplicar esto al tipo de datos que se trabajarán en CNN será muy costoso computacionalmente, por lo que se deberá tomar una muestra de cada batch y usar dicho subconjunto de datos (minibatch) para calcular el costo. Esta nueva función de costo depende de los pesos W y de los subconjuntos de datos empleados X e T como subconjunto del total de datos ingresados y sus clases respectivamente. La nueva función de costo estará dada por

$$L(W, X, T) = \sum_{X,T} L(W, x, t)$$

El método de **Stochastic Gradient Descent** entrena los pesos mencionados de la siguiente manera:

$$W \leftarrow W - \alpha \nabla_W L(W, X, T)$$

La transmisión y actualización de estos nuevos pesos se realizarán usando back propagation y forward propagation.

II-C. AlexNet

II-C1. ReLU: Para modelos de Deep Learning la mejor función de activación utilizada es **ReLU** (Rectified Linear Unit). La función devuelve 0 si recibe cualquier entrada negativa, pero para cualquier valor positivo x devuelve ese valor. Por lo tanto, se puede escribir como:

$$f(x) = max(0, x)$$

Gracias a su forma permite que los modelos corrijan valores no lineales de una manera muy simple.

II-C2. Arquitectura: La arquitectura explicada a continuación es una simplificación de la arquitectura original que, para la época en la que salió el artículo original, utilizaba artificios computacionales mas complejos para compensar la falta de poder computacional.

Como se puede observar en la Fig. 4, la arquitectura empleada es bastante directa, y sus capas representan las siguientes operaciones:

- Convolución con $f = 11$ y $s = 4$
- Max pooling con $f = 3$ y $s = 2$
- *Same* Convolution con $f = 5$
- Max pooling con $f = 3$ y $s = 2$
- *Same* Convolution con $f = 5$

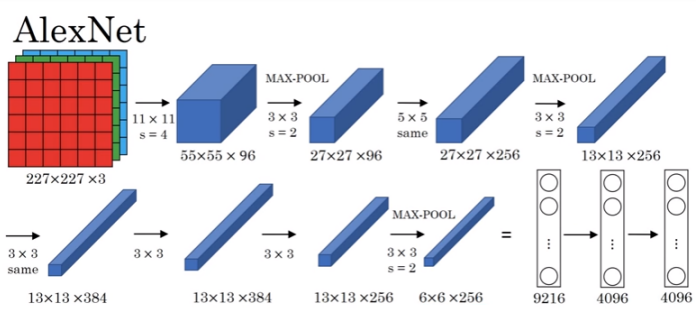

Figura 4. Arquitectura AlexNet para CNN (Simplificación)

- 2 Convoluciones con $f = 3$
- Max pooling con $f = 3$ y $s = 2$
- Reshape a vector (9216 componentes)
- 2 Capas Fully Connected a 4096 componentes

II-D. COWC

El dataset a emplear para este proyecto será *Cars Overhead with Context (COWC)*, compuesto por imágenes aéreas de Columbus, Postdam, Selwyn, Toronto, Utah y Vaihingen. Cada uno de estos sub-datasets contiene la imágen urbana, una imagen de anotación de positivos y otra de negativos indicando dónde están los automóviles.

Figura 5. Ejemplo de sub-imágen seleccionada para entrenamiento. Se separa aproximadamente un 25 % para pruebas posteriores al entrenamiento.

III. Diseño del experimento

La idea de implementación que se tiene para este proyecto comienza con la preparación de la data. Por cada imágen de la base de datos se tomará un volumen del 75 % para entrenamiento y el resto para validación (Fig. 5) (utilizando métodos de OpenCV o alguna librería similar), con esto se mapeará cada sub-imágen a la posición esperada de localización del automóvil. Se ingresará dicha imágen a la red neuronal con la arquitectura explicada y se utilizará la predicción para aplicar el Stochastic Gradient Descent.

Referencias

[1] Mundhenk, T. N., Konjevod, G., Sakla, W. A., & Boakye, K. (2016). "A Large Contextual Dataset for Classification, Detection and Counting of Cars with Deep Learning". Computer Vision – ECCV 2016 Lecture Notes in Computer Science, 785-800. doi:10.1007/978-3-319-46487-9_48

[2] Liu, D., & Wang, Y. (2015). "Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning."

[3] Bautista, C. M., Dy, C. A., Manalac, M. I., Orbe, R. A., & Cordel, M. (2016). "Convolutional neural network for vehicle detection in low resolution traffic videos." 2016 IEEE Region 10 Symposium (TENSYMP). doi:10.1109/tenconspring.2016.7519418

[4] Koga, Y., Miyazaki, H., Shibasaki, R. "A CNN-Based Method of Vehicle Detection from Aerial Images Using Hard Example Mining." Remote Sens. 2018, 10, 124.

[5] Krizhevsky, A., Sutskever, I., Hinton, G. "ImageNet Classification with Deep Convolutional Neural Networks."