Resumen de la CNN

Christofer Fabián Chávez Carazas
Universidad Nacional de San Agustín
Inteligencia Artificial

12 de junio de 2017

1. Convolutional Neural Networks

Las redes neuronales covolucionales (CNN) son son una clase mejorada de las redes neuronales convencionales, muy efectivas en la clasificación y reconocimiento de imágenes. Mientras que a las redes de Kohonen, Hopfield, entre otras, se les pasa como entrada el vector característica de una imagen; a una CNN se le pasa la imagen entera, tal como está, como entrada. Una CNN reconoce patrones gracias a las matrices de convolución, y forma vectores característica, que al final son procesados para obtener un vector de salida con las similitudes con todas las clases, y así poder clasificar la imagen.

2. Convolución

Una convolución es un operador matemático que convierte dos funciones en una tercera, siendo esta la representación de la magnitud cuando las dos funciones se superponen. Matemáticamente, la convolución de f y g se denota $f * g$ y se define con la integral:

$$f * g = \int_{-\infty}^{\infty} f(n)g(t-n)dn$$

En una CNN, al estar operando imágenes, no tiene sentido utilizar esta definición. En cambio se utiliza la definición de convolución discreta:

$$f[m] * g[m] = \sum_{n=-\infty}^{\infty} f[n]g[m-n]$$

2.1. Convolución en una CNN

En una CNN existen las llamadas capas de convolución, las cuales aplican la convolución a una matriz o a un conjunto de matrices con diferentes filtros. En esta etapa se extraen las características de una imagen. La entrada de la capa es la imagen original si es que es

la primero, o una matriz o varias matrices si es que no lo es. Cada capa tiene varios filtros representados por una matriz de convolución, que si comparamos con las redes neuronales tradiciones, estas matrices vendrían a ser los pesos de la neuronas. La idea principal es hacer un barrido con la matriz de convolución por toda la matriz de entrada. Si tenemos una matriz de entrada I de tamaño (I_n, I_m) y nuestra matriz de convolución K de tamaño (K_n, K_m) entonces tendremos una matriz de salida IK de tamaño $(I_n - K_n + 1, I_m - K_m + 1)$. Cada valor (i, j) de la matriz IK se halla con la siguiente fórmula:

$$IK_{(i,j)} = \sum_{a=0}^{K_n-1} \sum_{b=0}^{K_m-1} K_{(a,b)} * I_{(i+a,j+b)} \tag{1}$$

Estas operaciones se hacen con cada filtro que tenga la capa. En la ecuación 1 la matriz de entrada I se mantiene igual, el K se cambia por cada matriz de convolución. La salida son q matrices, siendo q el número de matrices de convolución. Cuando la imagen de entrada es en blanco y negro, no hay mucho problema en aplicar la ecuación 1, pero si la imagen es a colores, entonces tendremos tres matrices de entrada; una para cada canal de la imagen: R, G y B. Para este caso la fórmula cambiará [2] de la siguiente forma:

$$IK_{(i,j)} = \sum_{a=0}^{K_n-1} \sum_{b=0}^{K_m-1} \sum_{d=0}^{D-1} K_{(a,b,d)} * I_{(i+a,j+b,d)} \tag{2}$$

Donde D es el número de dimensiones; en el caso de imágenes a color sería 3. Para que la ecuación 2 funcione, los filtros de la capa deben tener el mismo número de dimensiones que la imagen de entrada. Este caso sólo puede suceder en la primera capa de convolución, ya que la salida de esta capa van a ser matrices de una sola dimensión, lo cual significa que las siguientes capas de convolución recibirán como entrada una matriz con una sola dimensión.

3. Función de Activación

En redes neuronales, una función de activación determina cuánta energía es disparada por una neurona, o si se dispara o no dicha energía. En general, la suma de la multiplicación de los pesos con las entradas, o *Neta*, es el valor que se le pasa a la función. Existen varias funciones de activación: Escalón, Sigmoidea, Gaussiana, Simusoidal, entre otras. Pero la función adecuada para una CNN es la llamada *rectified linear unit (ReLU)*.

3.1. ReLU

Esta función se define de la siguiente manera:

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

siendo x la *Neta* de una neurona. La función reemplaza todos los valores negativos por 0, e introduce la no linealidad a la red neuronal. En una CNN, la función *ReLU* se aplica a cada valor de la matriz de salida, para así obtener una matriz libre de negativos, la cual es la entrada de la siguiente capa.

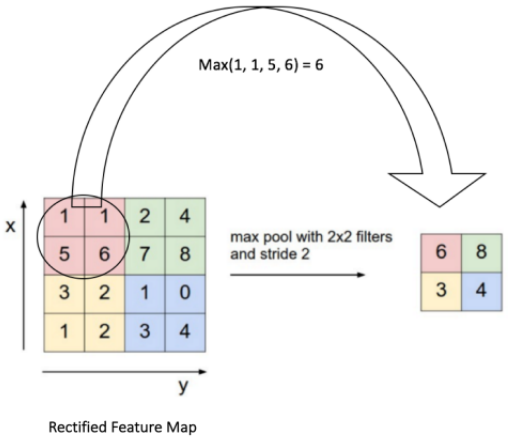

Figura 1: Ejemplo de Max Polling [1]

4. Spatial Pooling

El *Spatial Pooling* reduce la dimensionalidad de una matriz. Una función *pooling* con un tamaño de ventana kxk es aplicada a una matriz NxN de manera distribuida, siendo el resultado una matriz de tamaño $\frac{N}{k}x\frac{N}{k}$. En la Figura 1 se muestra la función *max pooling* con una ventana de $2x2$ aplicada a una función de tamaño $4x4$. La función divide la matriz en 4 ventanas, de las cuales se saca el número más alto y se forma la matriz resultante.

En una CNN existen las capas de submuestero, que lo único que hacen es aplicar una función *pooling* a la matriz o matrices de entrada. La salida es una matriz "resumen" que representa las características más importantes de la matriz de entrada. Las funciones *pooling* más usadas son: Máximo, Mínimo, Promedio y Suma. La función Máximo es la recomendada a usar en una CNN.

5. Fully Connected

La última capa de una CNN es una tradicional red neuronal multicapa totalmente conectada. La finalidad de esta capa es clasificar las entradas que le llegan, generando valores de similitud para cada clase de la red. Si la CNN tiene n clases entonces la salida de la *Fully Conected* tiene n valores. Hay una forma simple de ver esta capa matemáticamente [2]. Tenemos las entradas I con tamaño (I_d, I_n, I_m) siendo I_d el número de matrices de entrada, los pesos W, y las salidas z. Cada valor i de z se calcula con la siguiente ecuación:

$$z_i = \sum_{j=0}^{I_d-1} \sum_{r=0}^{I_n-1} \sum_{s=1}^{I_m} w_{(i,j,r,s)} * I_{(j,r,s)} \tag{3}$$

Luego cada valor z_i se pasa por una función de activación para que los valores de salida estén entre cero y uno, para luego poder ver las cercanías con cada clase.

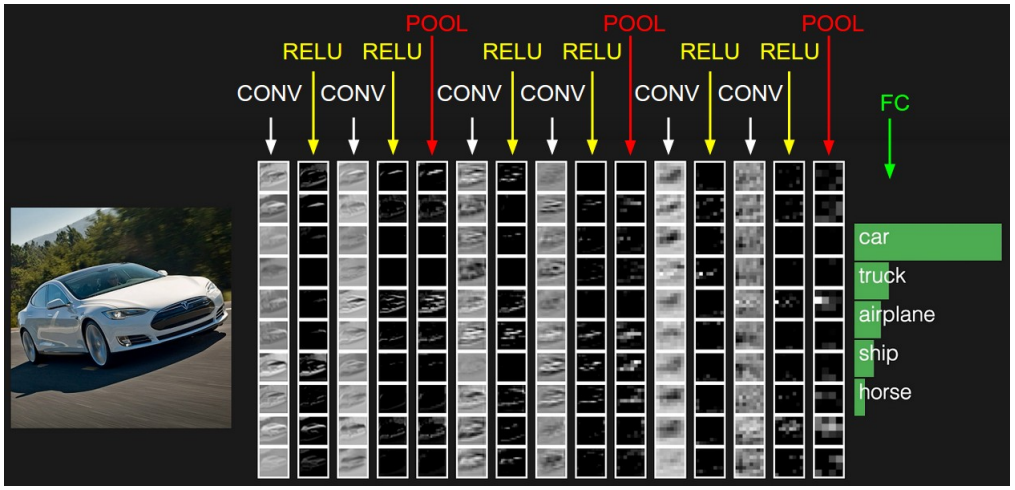

Figura 2: Ejemplo de la arquitectura de una CNN [1]

6. Arquitectura

Una CNN puede contener varias capas de las ya mencionadas anteriormente. El orden de las capas debe seguir las siguientes reglas:

- Una CNN siempre debe comenzar con una capa convolucional.
- Una CNN siempre debe terminar con una capa *Fully Connected* y sólo debe haber una.
- Siempre después de una capa convolucional debe ir una de *ReLu*.
- Las capas de *pooling* siempre deben ir después de las capas de *ReLu*, pero se pueden omitir.

En la Figura 2 se muestra la arquitectura de una CNN.

7. Backpropagation

Para calcular el error de toda la red utilizamos el error cuadrático:

$$E = \sum_{p=1}^{P} (\frac{1}{2}(t_p - a_p)^2) \tag{4}$$

donde P es en número de valores de la salida, t es la salida obtenida y a es la salida deseada. Nosotros queremos minimizar el error de toda la red. Para conseguir esto utilizamos la derivada del error respecto a los pesos. Para hallar esta derivada tenemos que desmenuzar las derivadas. Tenemos que hallar cómo cambia el error respecto a la operación de convolución y cómo cambia la operación de convolución respecto a los pesos:

$$\frac{\partial E}{\partial w_{x,y}} = \sum_{x'} \sum_{y'} \frac{\partial E}{\partial o_{x',y'}} \frac{\partial o_{x',y'}}{\partial w_{x',y'}} \tag{5}$$

Donde o es la operación de convolución. Primero veremos la derivada de la operación de convolución respecto a los pesos:

$$\frac{\partial o_{x',y'}}{\partial w_{x',y'}} = \frac{\partial}{\partial w_{x',y'}}(\sum_{x''}\sum_{y''} w_{x'',y''} * a_{x'+x'',y'+y''}) \tag{6}$$

$$\frac{\partial o_{x',y'}}{\partial w_{x',y'}} = \frac{\partial}{\partial w_{x',y'}}(w_{0,0} * a_{x'-0,y'-0} + \cdots + w_{x,y} * a_{x'-x,y'-y}) \tag{7}$$

$$\frac{\partial o_{x',y'}}{\partial w_{x',y'}} = \frac{\partial}{\partial w_{x',y'}}(w_{x,y} * a_{x'-x,y'-y}) \tag{8}$$

$$\frac{\partial o_{x',y'}}{\partial w_{x',y'}} = a_{x'-x,y'-y} \tag{9}$$

En la ecuación 6 se muestra la operación de convolución. En la ecuación 7 se desmenuza las sumatorias. En la ecuación 8 se eliminan las constantes. Y en la ecuación 9 se saca la derivada quedando sólo el valor de la entrada que multiplica al peso que se le saca la derivada.

Ahora hallamos la derivada del error respecto a la convolución. Para lograrlo desmenuzamos la derivada. Tenemos que hallar cómo cambia el error respecto a la salida, y cómo cambia la salida respecto a la operación de convolución:

$$\frac{\partial E}{\partial o_{x',y'}} = \frac{\partial E}{\partial a_{x',y'}}\frac{\partial a_{x',y'}}{\partial o_{x',y'}} \tag{10}$$

De la ecuación 10 hallamos la derivada de la salida respecto a la operación de convolución:

$$\frac{\partial a_{x',y'}}{\partial o_{x',y'}} = f^{'}(o_{x',y'}) \tag{11}$$

El resultado de esta derivada simplemente es la derivada de la función de activación, en nuestro caso sería la derivada de la función $ReLU$

Referencias

[1] CS231n Convolutional Neural Networks for Visual Recognition, Stanford

[2] Stutz, David. "Understanding convolutional neural networks." In Seminar Report, Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII Computer Vision. 2014.

[3] Jianxin Wu, "Introduction to Convolutional Neural Networks", 2017

[4] Jefkine, "Backpropagation In Convolutional Neural Networks", 2016