

Redes Convolucionales

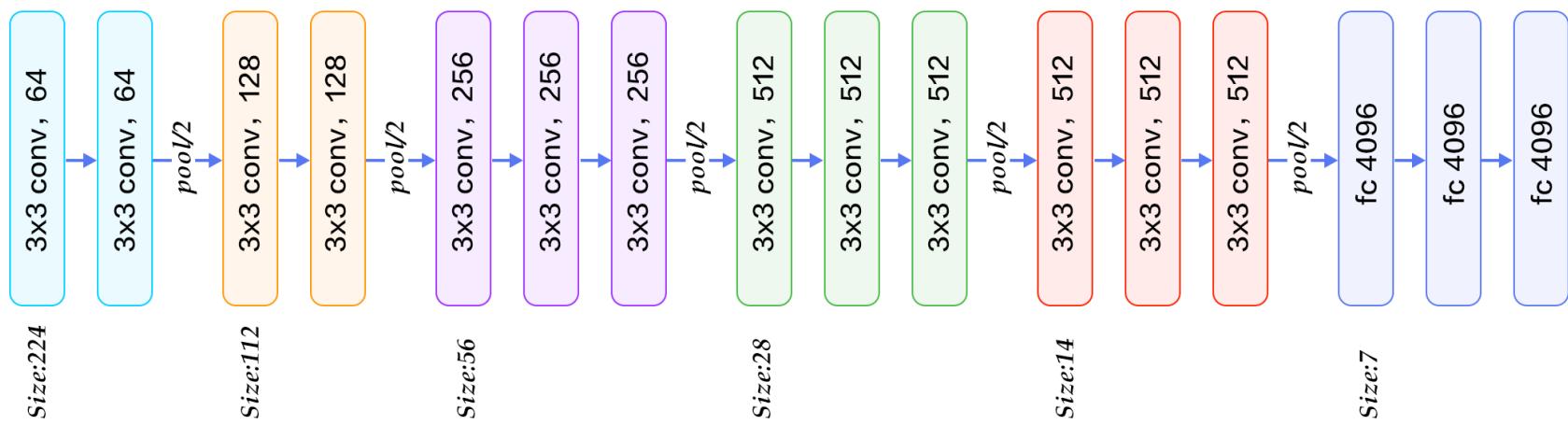
Backpropagation Convolucional



Prof. Ricardo Ñanculef - Departamento de Informática UTFSM 2022

Entrenamiento Redes Convolucionales

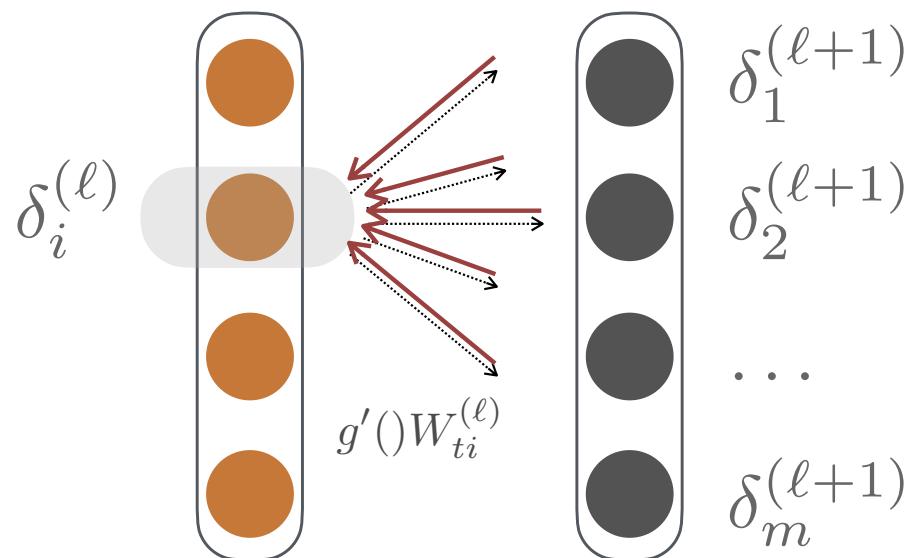
- Una red convolucional es una red que incluye capas convolucionales y capas de pooling que pueden estar organizadas de acuerdo a diferentes patrones de diseño.



Entrenamiento Redes Convolucionales

- Afortunadamente, el método de entrenamiento que hemos estudiado para redes feed-forward es extremadamente modular: cada capa puede determinar el ajuste que aplicar a sus pesos cuando conoce las señales de error correspondientes a la capa que le sigue:

$$\delta_i^{(\ell)} = \sum_t \delta_t^{(\ell+1)} g'() W_{ti}^{(\ell)}$$



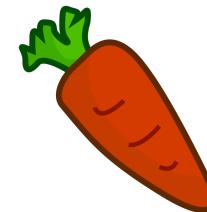
Ecuaciones de BP

$$\delta_i^{(\ell)} = \sum_t \delta_t^{(\ell+1)} g'(p^{(\ell+1)})_t W_{ti}^{(\ell+1)}$$

$$\frac{\partial L(f(x), y)}{\partial W_{ij}^{(\ell)}} = \frac{\partial L(f(x), y)}{\partial a_i^{(\ell)}} \frac{\partial a_i^{(\ell)}}{\partial W_{ij}^{(\ell)}}$$

||

$$g'(p_i^{(\ell)}) a_j^{(\ell-1)}$$

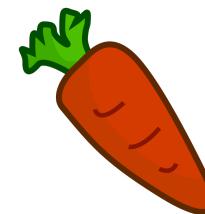


Ecuaciones de BP

$$\delta_j^{(\ell-1)} = \sum_i \delta_i^{(\ell)} g'(p^{(\ell)})_i W_{ij}^{(\ell)}$$

||

$$\frac{\partial L(f(x), y)}{\partial W_{ij}^{(\ell)}} = \frac{\partial L(f(x), y)}{\partial a_i^{(\ell)}} \frac{\partial a_i^{(\ell)}}{\partial W_{ij}^{(\ell)}}$$



||

$$g'(p_i^{(\ell)}) a_j^{(\ell-1)}$$

$$p^{(\ell)} = W^{(\ell)} a^{(\ell-1)} - b^{(\ell)}$$

Entrenamiento Redes Convolucionales

- Para adaptar este algoritmo al caso convolucional sólo necesitamos saber como calcular $\delta_i^{(\ell)}$ en una capa convolucional (de modo de continuar la recursión hacia abajo) y cómo se relaciona $\delta_i^{(\ell)}$ con el ajuste de los pesos convolucionales.

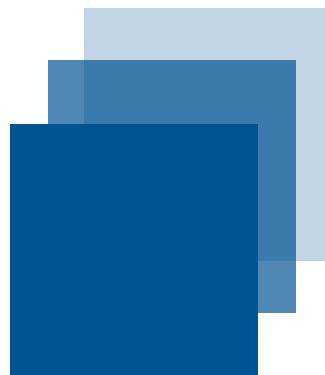
$$\frac{\partial L(f(x), y)}{\partial W_{ij}^{(\ell)}} = \frac{\partial L(f(x), y)}{\partial a_i^{(\ell)}} \frac{\partial a_i^{(\ell)}}{\partial W_{ij}^{(\ell)}}$$

The diagram illustrates the backpropagation formula for training convolutional neural networks. It shows a question mark box at the top, connected by a double vertical bar to a fraction. The numerator of the fraction is another question mark box. The denominator is split into two parts: the first part is a box with a question mark, and the second part is a box with a red border containing the term $\frac{\partial a_i^{(\ell)}}{\partial W_{ij}^{(\ell)}}$. This visualizes how the gradient of the loss function with respect to weights is calculated by propagating errors from the output layer back through the network, taking into account the specific structure of convolutional layers.

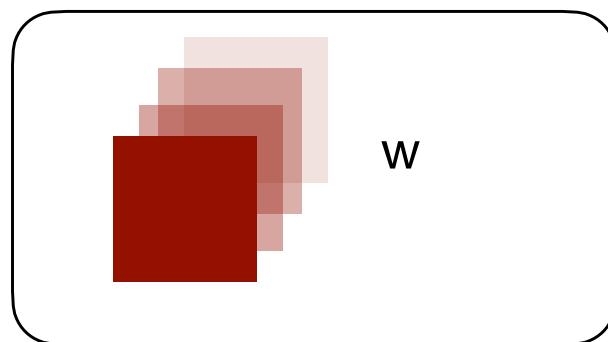
Ecuaciones de la Capa

- Matemáticamente, la ecuación que define la acción de la capa (2D) es

$$y_o = g \left(\sum_k w_{ok} * x_k - b_{ok} \right)$$



output y



W



input x

Ecuaciones de la Capa

- Matemáticamente, la ecuación que define la acción de la capa (2D) es

$$y_o = g \left(\sum_k w_{ok} * x_k - b_{ok} \right)$$

$$a_o^{(\ell)} = g \left(\sum_k w_{ok}^{(\ell)} * a_k^{(\ell-1)} - b_{ok}^{(\ell)} \right)$$

$$a_o^{(\ell)} = g \left(\sum_k p_{ok}^{(\ell)} \right)$$

$$p_{ok}^{(\ell)} = w_{ok}^{(\ell)} * a_k^{(\ell-1)} - b_{ok}^{(\ell)}$$

Ecuaciones de la Capa

- Asumiremos que las capas calculan una convolución válida:

$$a_o^{(\ell)} = g \left(\sum_k p_{ok}^{(\ell)} \right)$$
$$p_{ok}^{(\ell)} = w_{ok}^{(\ell)} * a_k^{(\ell-1)} - b_{ok}^{(\ell)}$$

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m+r, n+s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

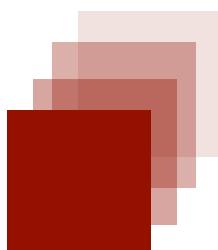
Derivación de BP Convolucional

- ¿Qué necesitamos?

$$a_o^{(\ell)} = g \left(\sum_k p_{ok}^{(\ell)} \right)$$
$$p_{ok}^{(\ell)} = w_{ok}^{(\ell)} * a_k^{(\ell-1)} - b_{ok}^{(\ell)}$$

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m+r, n+s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

Queremos



$$\frac{\partial L}{\partial w_{ok}^{(\ell)}} = \widetilde{\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}}}$$

Esto es una matriz! de RxS (tamaño del campo receptivo). Calcularemos cada elemento de la matriz

$$\frac{\partial L}{\partial w_{ok}^{(\ell)}}[r, s] = \frac{\partial L}{\partial w_{ok}^{(\ell)}[r, s]}$$

Derivación de BP Convolucional

- ¿Qué necesitamos?

$$a_o^{(\ell)}[m, n] = g \left(\sum_k p_{ok}^{(\ell)}[m, n] \right)$$

activación
elemento a elemento

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m+r, n+s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

pre-activación

Queremos

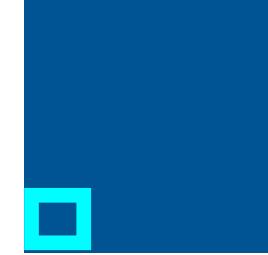
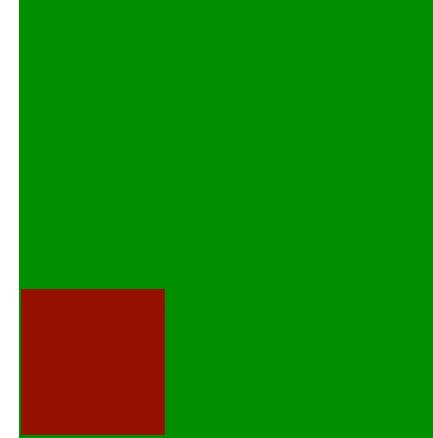
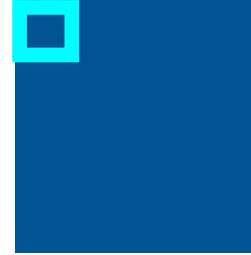
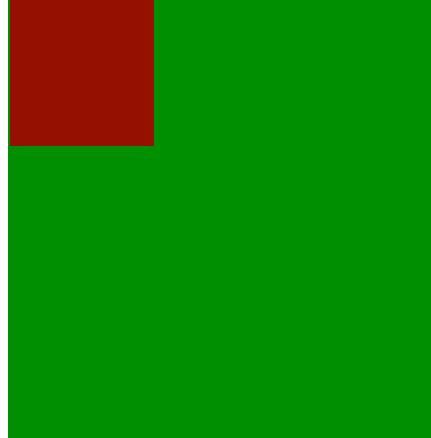
$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}[r, s]}$$

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}}[r, s] = \sum_{m,n} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \frac{\partial p_{ok}^{(\ell)}[m, n]}{\partial \tilde{w}_{ok}^{(\ell)}[r, s]}$$

Debido a la convolución, los pesos (compartidos) w influyen en todos las activaciones de salida.



Derivación de BP Convolucional



Queremos

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}[r, s]}$$

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}[r, s]} = \sum_{m,n} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \frac{\partial p_{ok}^{(\ell)}[m, n]}{\partial \tilde{w}_{ok}^{(\ell)}[r, s]}$$

Debido a la convolución, los pesos (compartidos) w influyen en todas las activaciones de salida.

Derivación de BP Convolucional

- Revisando la ecuación de la **pre-activación** ...

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m + r, n + s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}}[r, s] = \sum_{m,n} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \frac{\partial p_{ok}^{(\ell)}[m, n]}{\partial \tilde{w}_{ok}^{(\ell)}[r, s]}$$

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}}[r, s] = \sum_{m,n} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} a_k^{(\ell-1)}[m + r, n + s]$$

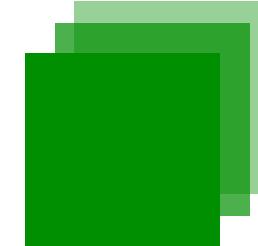
$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}} = \frac{\partial L}{\partial p_{ok}^{(\ell)}} * a_k^{(\ell-1)}$$

Esto es una convolución!
(válida)

Derivación de BP Convolucional

- Revisando la ecuación de la **activación** ...

$$a_o^{(\ell)}[m, n] = g \left(\sum_k p_{ok}^{(\ell)}[m, n] \right)$$

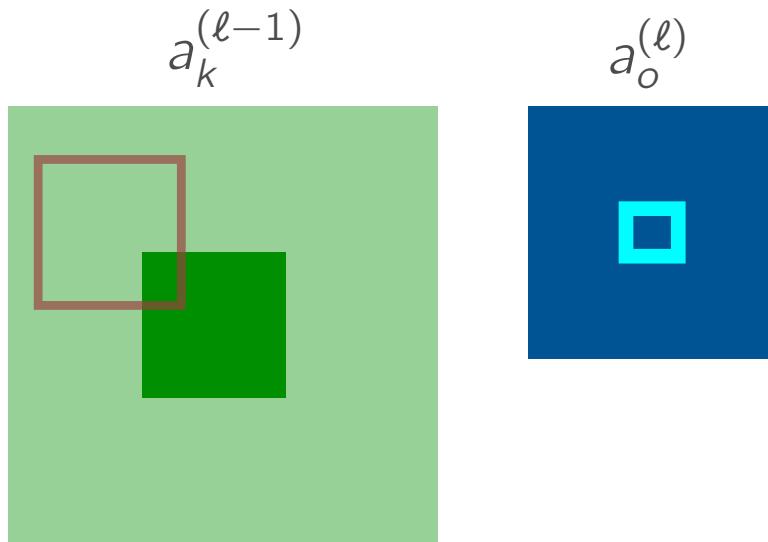


$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}} = \frac{\partial L}{\partial p_{ok}^{(\ell)}} * a_k^{(\ell-1)} \quad \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} = g' \left(\sum_k p_{ok}^{(\ell)}[m, n] \right) \frac{\partial L}{\partial a_o^{(\ell)}[m, n]}$$

Recusión de BP. Asumiremos que este gradiente ha sido calculado por la capa de más arriba (podría no ser convolucional) y enviaremos lo necesario para continuar la recursión

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[m, n]}$$

Derivación de BP Convolucional



Los datos que entran a la capa convolucional no influyen sobre todos los datos de salida, sino sólo sobre las neuronas del FC que tienen campos receptivos direcccionados hacia allí (en cada canal)

Queremos

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[m, n]}$$

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m+r, n+s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

Derivación de BP Convolucional

Queremos

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[m, n]}$$

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m + r, n + s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[i, j]} = \sum_o \sum_{\substack{m: \exists r, m+r=i \\ n: \exists s, n+s=j}} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \frac{\partial p_{ok}^{(\ell)}[m, n]}{\partial a_o^{(\ell-1)}[i, j]}$$



Derivación de BP Convolucional

Queremos

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[m, n]}$$

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m + r, n + s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[i, j]} = \sum_o \sum_{\substack{m: \exists r, m+r=i \\ n: \exists s, n+s=j}} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \frac{\partial p_{ok}^{(\ell)}[m, n]}{\partial a_o^{(\ell-1)}[i, j]}$$

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[i, j]} = \sum_o \sum_{\substack{m: \exists r, m+r=i \\ n: \exists s, n+s=j}} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \tilde{w}_{ok}^{(\ell)}[r, s]$$



Derivación de BP Convolucional

Queremos

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[m, n]}$$

$$p_{ok}^{(\ell)}[m, n] = \sum_{r,s} a_k^{(\ell-1)}[m + r, n + s] \tilde{w}_{ok}^{(\ell)}[r, s] - b_{ok}^{(\ell)}$$

$$\frac{\partial L}{\partial a_k^{(\ell-1)}[i, j]} = \sum_o \sum_{\substack{m: \exists r, m+r=i \\ n: \exists s, n+s=j}} \frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} \tilde{w}_{ok}^{(\ell)}[r, s]$$

$$\frac{\partial L}{\partial a_k^{(\ell-1)}} = \sum_o \frac{\partial L}{\partial p_{ok}^{(\ell)}} * \tilde{w}_{ok}^{(\ell)}$$

**Esto es una convolución!
(completa)**



Entrenamiento Redes Convolucionales

- En resumen, durante el entrenamiento, una capa convolucional continua la recursión de BP calculando

$$\frac{\partial L}{\partial a_k^{(\ell-1)}} = \sum_o \frac{\partial L}{\partial p_{ok}^{(\ell)}} * \tilde{w}_{ok}^{(\ell)}$$

- La capa pide adaptar los pesos de sus filtros calculando

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}} = \frac{\partial L}{\partial p_{ok}^{(\ell)}} * a_k^{(\ell-1)}$$

$$\frac{\partial L}{\partial p_{ok}^{(\ell)}[m, n]} = g'(\sum_k p_{ok}^{(\ell)}[m, n]) \frac{\partial L}{\partial a_o^{(\ell)}[m, n]}$$

$$\frac{\partial L}{\partial p_{ok}^{(\ell)}} = g'(\sum_k p_{ok}^{(\ell)}) \odot \frac{\partial L}{\partial a_o^{(\ell)}}$$



Entrenamiento Redes Convolucionales

- En resumen, durante el entrenamiento, una capa convolucional continua la recursión de BP calculando

$$\delta_k^{(\ell-1)} = \sum_o g' \left(\sum_k p_{ok}^{(\ell)} \right) \odot \left(\delta_o^{(\ell)} * \tilde{w}_{ok}^{(\ell)} \right)$$

$$\delta_k^{(\ell-1)} = \sum_o g' \left(p_o^{(\ell)} \right) \odot \left(\delta_o^{(\ell)} * \tilde{w}_{ok}^{(\ell)} \right)$$

- La capa pide adaptar los pesos de sus filtros calculando

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}} = g' \left(\sum_k p_{ok}^{(\ell)} \right) \odot \left(\delta_o^{(\ell)} * a_k^{(\ell-1)} \right)$$

Entrenamiento Redes Convolucionales

CONVOLUCIONAL

$$\delta_k^{(\ell-1)} = \sum_o g' \left(p_o^{(\ell)} \right) \odot \left(\delta_o^{(\ell)} * \tilde{w}_{ok}^{(\ell)} \right)$$

DENSO

$$\delta_j^{(\ell-1)} = \sum_i \delta_i^{(\ell)} g'(p^{(\ell)})_i W_{ij}^{(\ell)}$$

$$\frac{\partial L}{\partial \tilde{w}_{ok}^{(\ell)}} = g' \left(p_o^{(\ell)} \right) \odot \left(\delta_o^{(\ell)} * a_k^{(\ell-1)} \right)$$

$$\frac{\partial L}{\partial W_{ij}^{(\ell)}} = g'(p_i^{(\ell)}) a_j^{(\ell-1)}$$



Entonces ...

- Para entrenar una red convolucional usaremos esencialmente el mismo algoritmo que usábamos para entrenar redes densas.
- Gracias a la modularidad de este método, sólo necesitamos (i) adaptar el paso de las señales de error por capas convolucionales (+pooling) para restaurar la recursión y (ii) determinar cómo usar esa señal para adaptar los pesos de cada filtro de la capa.
- La recursión que se obtiene tiene una forma análoga a la del caso denso con los productos cambiamos por una convolución y un producto elemento elemtentoo.
- El gradiente para ajustar los pesos queda también con una forma análoga a la del caso denso y se calcula también como una convolución de la señal de error recursiva con el volumen que haya entrado a la capa.

