INF-477. Redes Neuronales Artificiales. Apuntes CNNs

Prof. Ricardo Ñanculef & Carlos Valle
jnancu@inf.utfsm.cl & *cvalle@inf.utfsm.cl*

1 Introducción

Las redes neuronales convolucionales (CNNs o ConvNets) representan una familia de modelos inspirados y altamente especializados en problemas de visión computacional (si bien hoy se pueden encontrar aplicaciones en muchas otras tareas donde es posible identificar algún tipo de topología espacial/temporal, como es el análisis de series de tiempo y el procesamiento de lenguaje natural). Una de las tareas de visión artificial en las que las CNN usualmente proveen state-of-the-art accuracy es el reconocimiento de imágenes. Uno de los problemas que aparecen cuando uno intenta aplicar FFN en estos problemas es la alta dimensionalidad de las imágenes de entrada, lo que típicamente resulta en un exagerado número de parámetros libres, sobre todo si se intenta definir modelos formados por más de 1 capa oculta. Esto trae consigo problemas computacionales y un mayor riesgo de overfitting. Para lidiar con este problema, las CNN implementan tres ideas claves

1. **Local/sparse connectivity:** A diferencia de las neuronas de una FNN, las unidades de una CNN estarán conectadas a un subconjunto de las neuronas de la capa anterior, conocido como *el campo receptivo* de la unidad.
2. **Parameter Sharing:** Como en algunos de los auto-encoders que hemos estudiado en capítulos anteriores, en una CNN hay un alto de grado de compartición de parámetros: subconjuntos de unidades con campos receptivos diferentes serán obligadas a compartir los mismos pesos. A una colección de neuronas que comparten pesos se les denomina un *feature map*.
3. **Pooling:** Para reducir dimensionalidad es posible construir operadores que "compriman" un subconjunto de características relacionadas espacialmente.

2 Convolutions

Matemáticamente, la idea de conectividad local y compartición de pesos se puede implementar introduciendo un nuevo tipo de operación a la red: *la convolución*. La convolución es una de los conceptos fundamentales en procesamiento digital de señales ya que provee una forma súper conveniente de describir la respuesta de un sistema frente a un estímulo ó señal.

Dadas dos señales $x(t)$ y $k(t)$ a tiempo continuo, la convolución de $x(t)$ y $k(t)$ se anota $x * k$ y se define como

$$s(t) = (x * k)(t) = \int x(a)\, k(t-a) da \ . \tag{1}$$

La idea detrás de esta operación, como habíamos adelantado, es caracterizar la reacción de un sistema a la señal $x(t)$. Para entender esto supongamos que $x(t)$ es un impulso unitario que se le presenta al sistema en el tiempo t_0, es decir $x(t) = \delta(t - t_0)$. Si calculamos la convolución de $x(t)$ y $k(t)$ para este caso especial, obtenemos

$$s(t) = (x * k)(t) = \int \delta(a - t_0)\, k(t-a) da = k(t - t_0) \ . \tag{2}$$

Si evaluamos el resultado en $t = t_0$, obtenemos $s(t_0) = k(0)$. Si la evaluamos en $t = t_0 + 1$, obtenemos $s(t_0+1) = k(1)$. Si la evaluamos en $t = t_0+2$, obtenemos $s(t_0+2) = k(2)$. Es decir, la función $k(i)$ representa la reacción del sistema i unidades de tiempo después del estímulo. El punto más interesante es, sin embargo, que $k(t)$ nos permite conocer la reacción del sistema a cualquier otra señal discreta $x(t)$, no sólo al impulso unitario. En efecto, una señal discreta arbitraria indexada en los tiempos $t_0, t_1, t_2, \ldots$ se puede escribir como una suma de impulsos unitarios,

$$x(t) = \sum_{i=-\infty}^{\infty} \alpha_i \delta(t - t_i). \tag{3}$$

Si convulsionamos $x(t)$ y $k(t)$ obtenemos

$$s(t) = (x * k)(t) = \int \left(\sum_{i=-\infty}^{\infty} \alpha_i \delta(a - t_i) \right) k(t - a) da \tag{4}$$

$$= \sum_{i=-\infty}^{\infty} \alpha_i k(t - t_i). \tag{5}$$

Es decir, $k(t)$ es lo único que necesitamos saber para describir la reacción del sistema a un estímulo arbitrario. En procesamiento digital de señales $k(t)$ se denomina *respuesta a impulso* (*impulse response*). En otros contextos se denomina *kernel* o *filtro*. En el caso de que se necesite trabajar con señales bi-dimensionales (las imágenes lo son), se define el operador convolución como sigue:

$$s(t, w) = (x * k)(t, w) = \int \int x(a, b)\, k(t - a, w - b) da\, db\ . \tag{6}$$

En el caso de señales discretas (con las que usualmente trabajaremos), el operador de convolución se define como sigue (caso uni-dimensional y bi-dimensional respectivamente).

$$s(i) = (x * k)(i) = \sum\nolimits_{p=-\infty}^{\infty} x(p) k(i - p)\ , \tag{7}$$

$$s(i, j) = (x * k)(i, j) = \sum\nolimits_{p,q=-\infty}^{\infty} x(p, q) k(i - p, j - q)\ . \tag{8}$$

Una cosa "interesante" de la convolución es el hecho de que cuando se calcula, la señal se "recorre" en una dirección y el kernel en otra. La señal se "recorre" en sentido creciente de sus índices, y el kernel en sentido decreciente. En el caso bi-dimensional este efecto se produce en ambas direcciones. Una forma de eliminar este efecto es considerar un operador muy similar al de convolución denominado *correlación cruzada* (*cross correlation*). Si x y k son dos señales bi-dimensionales discretas, la correlación de x y k se anota $x \star k$ y se define como

$$s(i, j) = (x \star k)(i, j) = \sum\nolimits_{m,n=-\infty}^{\infty} x(m, n) k(i + m, j + n)\ . \tag{9}$$

Una motivación fuerte para preferir la convolución a la correlación es que ésta última tiene propiedades importantes como la conmutatividad y la asociatividad.

$$(x * k) = (x * k) \tag{10}$$

$$x * (k_1 * k_2) = (x * k_1) * k_2 \tag{11}$$

3 Convoluciones sobre Arrays

Desde el punto de vista práctico, las señales que nosotros procesamos son discretas, finitas y se encuentran típicamente representadas como arreglos uni-dimensionales (por ejemplo una serie de tiempo), bi-dimensionales (por ejemplo una imagen en escala de grises), tri-dimensionales (por ejemplo una imagen RGB) o de mayor dimensión. Las definiciones que hemos dado de las operaciones de convolución hasta el minuto resultan un poco abstractas si nos interesa implementarlas rápidamente en un algoritmo. Por este motivo resulta conveniente adaptarlas explícitamente pensando que los inputs del operador serán arrays y queremos obtener como resultado un array válidamente definido.

3.1 Caso 1D

Sea $\mathbf{x} \in \mathbb{R}^I$ un vector de entrada representado en un array uni-dimensional de largo I. Representaremos un kernel uni-dimensional $\boldsymbol{w} \in \mathbb{R}^R$ como un array uni-dimensional de tamaño R. La convolución entre $\mathbf{x}$ y $\boldsymbol{w}$ da como resultado un array uni-dimensional de tamaño $I - R + 1$ definido como

$$\mathbf{x} * \boldsymbol{w}[i] = \sum_{r=0}^{R-1} \mathbf{x}[i+r]\boldsymbol{w}[R-r-1] \;\; \forall i \in \{0, \ldots, I-R\} \tag{12}$$

Notemos que por conveniencia hemos decidido indexar las entradas de un array de tamaño T desde 0 a $T-1$ (en vez de 1 a T como es usual en matemáticas) siendo lo usual en la literatura específica de redes neuronales.

Relleno (Padding). Notemos que, como la hemos definido, el resultado de la convolución "contrae" el patrón de entrada $\mathbf{x}$. A esta forma de definir/computar la convolución la denominaremos *válida* (*valid*). Una forma de evitar este problema y generar un un resultado que mantiene la forma del patrón de entrada es usar lo que se denomina *zero padding*, es decir extender con zeros el patrón de entrada para obtener la forma deseada. La forma más común de hacerlo consiste en agregar zeros de modo simétrico en torno al patrón de entrada, situación que sólo permite preservar el tamaño de éste si el kernel tiene tamaño impar. En este caso se agregan $(R-1)/2$ 0's al inicio y $(R-1)/2$ 0's al final. Si la convolución se hace con zero padding, escribiremos $\bar{\mathbf{x}} * \boldsymbol{w}$,

$$\tilde{\mathbf{x}} * \boldsymbol{w}[i] = \sum_{r=0}^{R-1} \mathbf{x}[i+p]\boldsymbol{w}[R-r-1] \;\; \forall i \in [I]\,, \tag{13}$$

donde $[I] = \{0, \ldots, I-1\}$.

Notemos que la convolución sin zero padding puede obtenerse calculando la convolución con zero padding y luego "recortando" $(R-1)/2$ elementos al inicio y $(R-1)/2$ elementos al final. En este caso escribiremos $\underline{\bar{\mathbf{x}} * \boldsymbol{w}}$.

Full Convolution. Una última forma de definir/computar la convolución entre $\mathbf{x}$ y $\boldsymbol{w}$ se denomina *completa* (*full*),

$$\mathbf{x} * \boldsymbol{w}[i] = \sum_{r+s=i}^{R-1} \mathbf{x}[r]\boldsymbol{w}[s] \;\; \forall i \in [I+R-1]\;. \tag{14}$$

Notemos que con esta definición, la más común en los textos de matemática, el resultado de la convolución es de tamaño $I+R-1$. Notemos además que en esta definición, la suma para calcular el i-ésimo elemento del resultado se ejecuta sobre todos los índices $p \in [I], q \in [R]$ tal que $p+q=i$. De esta forma, el número de sumandos cambia de acuerdo al valor de i. Este inconveniente se compensa con las propiedades que se obtienen. En efecto, la convolución completa es siempre conmutativa y asociativa, lo que puede tener consecuencias importantes en el diseño de algoritmos. Finalmente es útil mencionar que la convolución que hemos llamado *válida*, se puede obtener a partir de una convolución completa recortando $R-1$ elementos al

inicio y $R-1$ elementos al final, para cualquier valor de R. Del mismo modo, la convolución completa puede definirse en modo idéntico a aquella que hemos llamado *válida* si hacemos un rellenado extra de ceros (zero padding): exactamente $R-1$ al inicio y $R-1$ al final de $\mathbf{x}$.

Volteado (Flipping). El efecto que mencionamos en la sección anterior de que durante el cómputo de la convolución, una señal se recorre en un sentido y la otra (típicamente el kernel) en la otra se puede "eliminar" usando una correlación en vez de una convolución, pero "volteando" el kernel antes de proceder, es decir

$$\bar{\mathbf{x}} * \boldsymbol{w}[i] = \mathbf{x} \star \tilde{\boldsymbol{w}}[i] = \sum_{r=0}^{R-1} \mathbf{x}[i+r]\tilde{\boldsymbol{w}}[r] \;\; \forall i \in [I-R+1]\,, \tag{15}$$

donde $\tilde{\boldsymbol{w}}[r] = \boldsymbol{w}[R-1-r]$.

3.2 Caso 2D

Sea $x \in \mathbb{R}^{IJ}$ un patrón de entrada representado en un matriz de tamaño $I \times J$. Definiremos un kernel bi-dimensional $\boldsymbol{w} \in \mathbb{R}^{RS}$ como una matriz de tamaño $R \times S$. La convolución entre $\mathbf{x}$ y $\boldsymbol{w}$ da como resultado una matriz de tamaño $(I-R+1) \times (J-S+1)$ definida como

$$\mathbf{x} * \boldsymbol{w}[i,j] = \sum_{r=0}^{R-1}\sum_{s=0}^{S-1} \mathbf{x}[i+r, j+s]\boldsymbol{w}[R-1-r, S-1-s] \;\; \forall i \in [I-R+1], \forall j \in [J+S-1] \tag{16}$$

Casi siempre se trabaja con kerneles cuadrados, así es que $R = S$.

Relleno 2D(Padding) El padding 2D es análogo al padding 1D, pero se hace en cada una de las dimensiones según sea necesario. Si la convolución se hace con zero padding, escribiremos $\tilde{\mathbf{x}} * k$,

$$\tilde{\mathbf{x}} * \boldsymbol{w}[i,j] = \sum_{r=0}^{R-1}\sum_{s=0}^{S-1} \tilde{\mathbf{x}}[i+r, j+s]\boldsymbol{w}[R-1-p, S-1-q] \;\; \forall i,j \in [I] \times [J] \tag{17}$$

Volteado (Flipping) De nuevo en este caso se puede calcular la convolución usando una correlación, pero "volteando" el kernel antes de proceder, es decir

$$\tilde{\mathbf{x}} * \boldsymbol{w}[i] = \tilde{\mathbf{x}} \star \tilde{\boldsymbol{w}}[i] = \sum_{r=0}^{R-1}\sum_{s=0}^{S-1} \tilde{\mathbf{x}}[i+r, j+s]\tilde{\boldsymbol{w}}[r,s] \;\; \forall i,j \in [I-R+1] \times [J-S+1]\,, \tag{18}$$

donde $\tilde{\boldsymbol{w}}[r,s] = \boldsymbol{w}[R-1-r, S-1-s]$.

4 CNNs

Estamos ahora en condiciones de definir una CNN.

Capa Convolucional Dado un patrón en entrada $\mathbf{x} \in \mathbb{R}^{IJK}$, i.e., un patrón bi-dimensional con K canales $\mathbf{x}_k \in \mathbb{R}^{IJ}$, $k = 1, \ldots, K$, una capa convolucional de N canales, filtros ó mapas característicos, es una transformación

$$\begin{aligned} H : \mathbb{R}^{IJK} &\to \mathbb{R}^{MNO} \\ \mathbf{x} &\to \mathbf{z}\,, \end{aligned} \tag{19}$$

donde

$$\mathbf{z}_o = \sigma\left(\sum_k \mathbf{x}_k \bar{*}\, \boldsymbol{w}_{nk}\right) \in \mathbb{R}^{MN} \;\; \forall o \in [O]\,, \tag{20}$$

donde $\boldsymbol{w}_{ok}$ es el kernel utilizado en la convolución del k-ésimo canal de entrada con el o-ésimo canal de salida y σ es una función, típicamente no lineal, denominada la función de activación de la capa.
Notemos que para producir el o-ésimo canal de salida se consideran (suman los efectos de) todos los canales de entrada. Notemos finalmente que asumimos que la convolución se lleva a cabo con zero padding. Si éste último se elige en modo apropiado, se puede garantizar $M = I$, $N = J$, pero en general serán diferentes.

Capa de Pooling Dado un patrón de entrada $x \in \mathbb{R}^{IJK}$, una capa de pooling con filtro de tamaño $R \times S$ y función de pooling g es una transformación

$$\begin{aligned} H : \mathbb{R}^{IJK} &\to \mathbb{R}^{MNO} \\ \mathbf{x} &\to \mathbf{z}\,, \end{aligned} \tag{21}$$

donde

$$\mathbf{z}_k[i,j] = g\left(A_x[i,j]\right) \in \mathbb{R}^{MN} \quad \forall k \in [K]\ \forall i,j \in [M] \times [N] \tag{22}$$

$$A_x[i,j] = \{x[p,q] : iR \leq p \leq (i+1)R,\ jS \leq q \leq (j+1)S\} \forall i,j \in [M] \times [N] \tag{23}$$

$$M = \lfloor I/R \rfloor,\ N = \lfloor J/S \rfloor \tag{24}$$

La función más típica de pooling es $g \equiv \max$ pero se han propuesto también otro tipo de operaciones como $g \equiv \text{mean}$.

5 Gradientes

Gradientes para la Capa Convolucional Para adaptar los pesos $\tilde{\boldsymbol{w}}^{\ell}_{ok} \in$ de una determinada capa convolucional $\mathbf{a}^{\ell} = H(\mathbf{a}^{\ell-1})$ que conecta el k-ésimo canal de entrada con el o-ésimo canal de salida, tendremos que calcular $\nabla E^{\ell}_{\tilde{\boldsymbol{w}}_{ok}} = \partial E / \partial \tilde{\boldsymbol{w}}^{\ell}_{ok}$ para nuestra función de costo/error E. Notemos que $\tilde{\boldsymbol{w}}^{\ell}_{ok}$ es una matriz de tamaño $R \times S$ (tamaño del kernel) y por lo tanto $\nabla E^{\ell}_{\tilde{\boldsymbol{w}}_{ok}}$ es también una matriz, del mismo tamaño. Cuando no cause confusión omitiremos los índices o, k, ℓ de los gradientes. Por ejemplo $\nabla E^{\ell}_{\tilde{\boldsymbol{w}}_{ok}}$ será denotado simplemente $\nabla E^{\ell}_{\tilde{\boldsymbol{w}}}$.

Tenemos que calcular

$$\nabla^{\ell}_{\tilde{\boldsymbol{w}}} E[r,s] = \frac{\partial E}{\partial \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s]} \tag{25}$$

Sea $\mathbf{p}^{\ell}_{ok} = \mathbf{a}^{\ell-1}_{k} * \boldsymbol{w}^{\ell}_{ok} = \mathbf{a}^{\ell-1}_{k} \star \tilde{\boldsymbol{w}}^{\ell}_{ok}$, de modo que

$$\mathbf{p}^{\ell}_{ok}[m,n] = \sum_{r',s'} \mathbf{a}^{\ell-1}_{k}[m+r', n+s'] \tilde{\boldsymbol{w}}^{\ell}_{ok}[r',s'] \tag{26}$$

Tenemos (por la compartición de pesos) que

$$\nabla^{\ell}_{\tilde{\boldsymbol{w}}_{ok}} E[r,s] = \frac{\partial E}{\partial \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s]} = \sum_{m,n=0} \frac{\partial E}{\partial \mathbf{p}^{\ell}_{ok}[m,n]} \frac{\mathbf{p}^{\ell}_{ok}[m,n]}{\partial \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s]} \tag{27}$$

$$= \sum_{m,n=0} \frac{\partial E}{\partial \mathbf{p}^{\ell}_{ok}[m,n]} \mathbf{a}^{\ell-1}_{k}[m+r, n+s] \tag{28}$$

$$= \sum_{m,n=0} \mathbf{a}^{\ell-1}_{k}[r+m, s+n] \nabla^{\ell}_{\mathbf{p}} E[m,n] \tag{29}$$

Es decir,

$$\nabla^{\ell}_{\tilde{\boldsymbol{w}}} E = \mathbf{a}^{\ell-1}_{k} \star \nabla^{\ell}_{\mathbf{p}} E = \mathbf{a}^{\ell-1}_{k} * \widetilde{\nabla^{\ell}_{\mathbf{p}} E} \tag{30}$$

$$\nabla^{\ell}_{\mathbf{p}} E[m,n] = \frac{\partial E}{\partial \mathbf{p}^{\ell}_{ok}[m,n]} = \frac{\partial E}{\partial \mathbf{a}^{\ell}_{o}[m,n]} \frac{\partial \mathbf{a}^{\ell}_{o}[m,n]}{\partial \mathbf{p}^{\ell}_{ok}[m,n]} \tag{31}$$

$$= \frac{\partial E}{\partial \mathbf{a}^{\ell}_{o}[m,n]} \frac{\partial \sigma(\sum_{k'} \mathbf{p}^{\ell}_{ok'}[m,n])}{\partial \mathbf{p}^{\ell}_{ok}[m,n]} \tag{32}$$

$$= \frac{\partial E}{\partial \mathbf{a}^{\ell}_{o}[m,n]} \sigma'(\sum\nolimits_{k'} \mathbf{p}^{\ell}_{ok'}[m,n]) \tag{33}$$

es decir,

$$\nabla^{\ell}_{\mathbf{p}} E = \nabla^{\ell}_{\mathbf{a}} E \odot \sigma'(\sum\nolimits_{k'} \mathbf{p}^{\ell}_{ok'}) \tag{34}$$

Asumimos que la señal de error $\nabla^{\ell}_{\mathbf{a}} E = \partial E / \partial \mathbf{a}^{\ell}_{o}$ fue enviada por la capa de más arriba (esa capa sabe como calcular ese gradiente, podría no ser convolucional). Para ser consistentes debemos enviar $\nabla^{\ell-1}_{\mathbf{a}} E = \partial E / \partial \mathbf{a}^{\ell-1}_{k}$. Notemos que este último gradiente es de tamaño $I \times J$. En efecto, sus coordenadas vienen dadas por

$$\nabla^{\ell-1}_{\mathbf{a}} E[i,j] = \frac{\partial E}{\partial \mathbf{a}^{\ell-1}_{k}[i,j]} \tag{35}$$

Para usar la regla de la cadena (multivariada), necesitamos saber en qué elementos de cada canal o de la capa ℓ influye $\mathbf{a}^{\ell-1}_{k}[i,j]$. Recordemos que

$$\mathbf{a}^{\ell}_{o} = \sigma\left(\sum\nolimits_{k'} \mathbf{p}^{\ell}_{ok'}\right) = \sigma\left(\sum\nolimits_{k'} \mathbf{a}^{\ell-1}_{k'} * \boldsymbol{w}^{\ell}_{ok'}\right) = \sigma\left(\sum\nolimits_{k'} \mathbf{a}^{\ell-1}_{k'} \star \tilde{\boldsymbol{w}}^{\ell}_{ok'}\right) \tag{36}$$

con

$$\mathbf{p}^{\ell}_{ok}[m,n] = \sum_{r,s} \mathbf{a}^{\ell-1}_{k}[m+r, n+s] \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s] \tag{37}$$

Por lo tanto, el elemento $\mathbf{a}^{\ell-1}_{k}[i,j]$ influye en los elementos $[m,n]$ de cada canal o tales que para algún $[r,s]$ se da que $m+r=i$ y $n+s=j$. Como r se mueve en $[R]$ y s en $[S]$, tenemos que, para $i \geq r, j \geq s$ (para otros i,j la expresión dependerá del tipo de padding),

$$\nabla^{\ell-1}_{\mathbf{a}} E[i,j] = \frac{\partial E}{\partial \mathbf{a}^{\ell-1}_{k}[i,j]} = \sum_{o} \sum_{r,s} \frac{\partial E}{\partial \mathbf{p}^{\ell}_{ok}[i-r, j-s]} \frac{\partial \mathbf{p}^{\ell}_{ok}[i-r, j-s]}{\partial \mathbf{a}^{\ell-1}_{k}[i,j]} \tag{38}$$

Ahora, de la ecuación (37) vemos que $\partial \mathbf{p}^{\ell}_{ok}[m,n] / \partial \mathbf{a}^{\ell-1}_{k}[i,j]$ será no nula sólo cuando $m+r=i$ y $n+s=j$, es decir, cuando $r=i-m$ y $s=j-n$. Por lo tanto,

$$\frac{\partial \mathbf{p}^{\ell}_{ok}[m,n]}{\partial \mathbf{a}^{\ell-1}_{k}[i,j]} = \tilde{\boldsymbol{w}}^{\ell}_{ok}[i-m, j-n] \tag{39}$$

Un simple cambio de variable permite obtener

$$\frac{\partial \mathbf{p}^{\ell}_{ok}[i-r, j-s]}{\partial \mathbf{a}^{\ell-1}_{k}[i,j]} = \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s] \tag{40}$$

Obtenemos entonces una forma recursiva de calcular la derivada requerida

$$\nabla^{\ell-1}_{\mathbf{a}} E[i,j] = \frac{\partial E}{\partial \mathbf{a}^{\ell-1}_{k}[i,j]} = \sum_{o} \sum_{r,s} \frac{\partial E}{\partial \mathbf{p}^{\ell}_{k}[i-r, j-s]} \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s] \tag{41}$$

$$= \sum_{o} \sum_{r,s} \nabla^{\ell}_{\mathbf{p}} E[i-r, j-s] \tilde{\boldsymbol{w}}^{\ell}_{ok}[r,s] \tag{42}$$

Lo anterior no es exactamente una convolución. Además, la fórmula vale para $i \geq r, j \geq s$. La extensión para cualquier i, j es simple: cuando $\nabla^\ell_{\mathbf{p}} E[i-r, j-s]$ corresponda a índices inválidos ($i < r$, $j < s$) asumimos que se trata de un 0. Como el valor máximo de r es $R-1$ y el de s es $S-1$, esto es equivalente a rellenar $\nabla^\ell_{\mathbf{p}} E$ con $R-1$ zeros a ambos lados de la primera dimensión y con $S-1$ zeros a ambos lados de la segunda dimensión. Este truco permite además obtener una fórmula que es una convolución correctamente definida. En efecto, si denotamos por $\overline{\nabla^\ell_{\mathbf{p}} E}$ el gradiente con este tipo de padding, tenemos que $\nabla^\ell_{\mathbf{p}} E[i,j] = \overline{\nabla^\ell_{\mathbf{p}} E}[R-1+i][S-1+j]$

$$\nabla^{\ell-1}_{\mathbf{a}} E[i,j] = \frac{\partial E}{\partial \mathbf{a}^{\ell-1}_k[i,j]} = \sum_o \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} \overline{\nabla^\ell_{\mathbf{p}} E}[R-1+i-r, S-1+j-s] \tilde{\boldsymbol{w}}^\ell_{ok}[r,s] \tag{43}$$

$$= \sum_o \sum_{r'=0}^{R-1} \sum_{s'=0}^{S-1} \overline{\nabla^\ell_{\mathbf{p}} E}[i+r', j+s'] \tilde{\boldsymbol{w}}^\ell_{ok}[R-1-r', S-1-s'] \tag{44}$$

$$= \sum_o \sum_{r',s'} \overline{\nabla^\ell_{\mathbf{p}} E}[i+r', j+s'] \boldsymbol{w}^\ell_{ok}[r', s'] \tag{45}$$

$$\nabla^{\ell-1}_{\mathbf{a}} E = \sum_o \overline{\nabla^\ell_{\mathbf{p}} E} * \tilde{\boldsymbol{w}}^\ell_{ok} = \sum_o \overline{\nabla^\ell_{\mathbf{p}} E} \star \boldsymbol{w}^\ell_{ok} \tag{46}$$