# 8. Reţele neurale convoluţionale

Rețelele convoluționale
Convoluția
Operația de pooling
Rețeaua neurală convoluțională

În ultimele două lucrări de laboratoar, am vorbit despre rețelele neurale simple, care au fiecare unitate din fiecare strat conectată la fiecare unitate din stratul anterior. Această abordare funcționează pentru o dimensiune relativ redusă a datelor de antrenare. Însă, atunci când lucrăm cu date multe și de dimensiuni mari (de exemplu, dacă dorim să antrenăm o rețea neurală pentru clasificarea unor imagini), antrenarea acestor rețele devine foarte costisitoare din punct de vedere al puterii de calcul necesare. De exemplu, dacă imaginile pe care le folosim la antrenare au dimensiunea de $96 \times 96$ pixeli, vom avea aproximativ $10^4$ unități de intrare și dacă vrem să învățăm 100 de caracteristici, vom aveam $10^6$ parametri de învățat. Propagarea înainte și propagarea înapoi vor dura de aproximativ 100 de ori mai mult față de cazul în care imaginile ar avea dimensiunile de $28 \times 28$ de pixeli.

O soluție pentru problema enunțată mai sus este restricționarea conxiunilor dintre unitățile straturilor adiacente, astfel încât acestea să fie conectate doar la o anumită zonă din stratul anterior [1]. De exemplu, pentru problema clasificării imaginilor, o unitate se conectează doar la un anumit grup de pixeli din imaginea de intrare. Această tehnică își are inspirația în modul de lucru al creierului uman, mai exact cortexul vizual, unde neuronii răspund doar la stimuli care provin dintr-o anumită locație.

### 8.1 Convoluția

Imaginile au proprietatea de a fi "staționare", mai exact caracteristicile într-o parte a imaginii sunt la fel în cealaltă parte (spunem că imaginea este invariabilă la translație). Așadar, caracteristicile calculate într-o locație se pot folosi în toate locațiile. Învățând, de exemplu, o serie de caracteristici într-o zonă de $8 \times 8$, le putem aplica în oricare parte a imaginii. Se spune că realizăm convoluția acelei zone de $8 \times 8$ cu imaginea mai mare. Ca un exemplu concret, presupunem că am învățat caracteristici pe "bucăți" de $8 \times 8$ dintr-o imagine de $96 \times 96$ și presupunem că avem un strat ascuns cu 100 de unități. Pentru a obține caracteristicile convoluționale pentru fiecare regiune de $8 \times 8$ din imagine, trebuie să se aplice

funcția de activare a stratului ascuns. Ar rezulta 100 de seturi de $89 \times 89$ de caracteristici convoluționale.

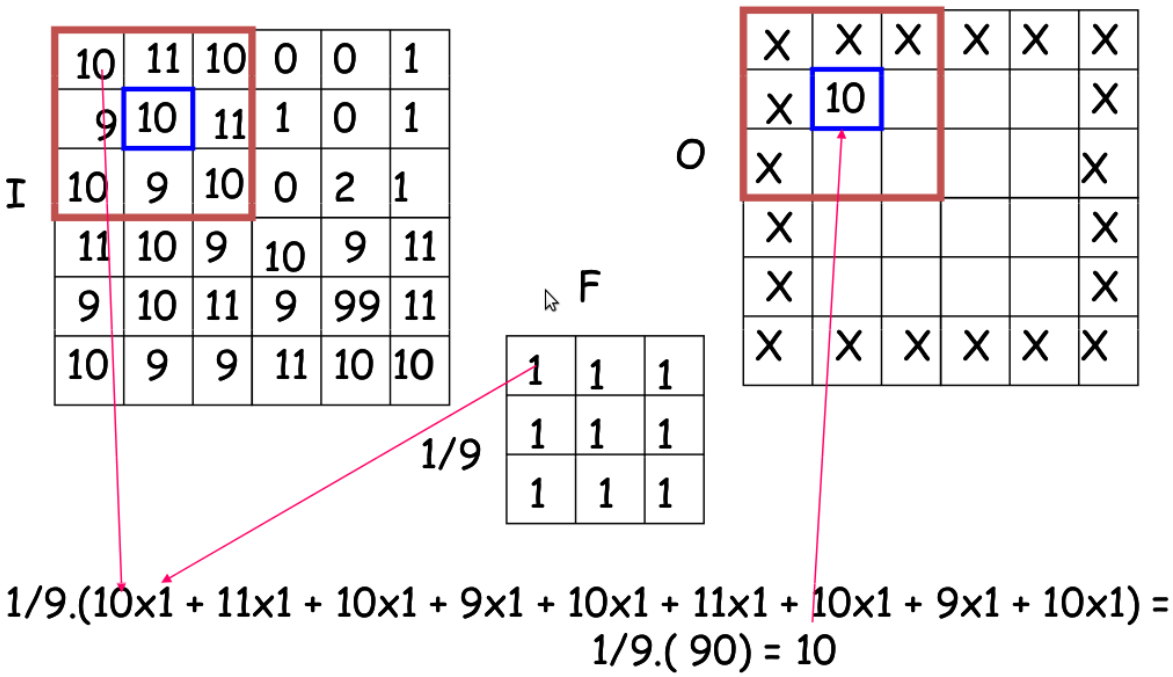


Fig. 8.1 Exemplu al operației de convoluție.

Matricea din stânga din imaginea 8.1 este imaginea de intrare, matricea $F$ este filtrul (în acest exemplu conține doar valori de 1), iar matricea din dreapta este rezultatul convoluției. Din puncte de vedere matematic, convoluția poate fi descrisă de formula următoare:

$$V = \frac{\sum_{i=1}^{q} (\sum_{j=1}^{q} f_{ij} d_{ij})}{F}, \tag{8.1}$$

unde, $f_{ij}$ sunt coeficienții filtrului aplicat, $d_{ij}$ sunt valorile din imaginea de intrare, $q$ este dimensiunea filtrului, iar $F$ este un factor de normalizare.

## 8.2 Operația de pooling

Operația de pooling se foloseşte pentru a reduce dimensiunea caracteristicilor într-o rețea convoluțională, ceea ce comprimă ieșirile unui strat convoluțional. Luând din nou exemplul de mai sus, cu o imagine de $96 \times 96$ pixeli, presupunem că am învățat 400 de caracteristici folosind filtre de dimensiunea $8 \times 8$. Obținem așadar un rezultat de dimensiunea $(96 - 8 + 1) * (96 - 8 + 1) = 7921$ și având 400 de caracteristici, rezultă un vector de dimensiunea $89^2 * 400 = 3168400$ de caracteristici per exemplu. Un clasificator pentru un număr așa mare de caracteristici este costisitor din punct de vedere computațional și predispus la supraantrenare. Profitând de proprietatea imaginilor de a fi staționare, putem agrega aceste

caracteristici la diferite locații. De exemplu, se poate calcula media sau maximul unei caracteristici dintr-o regiune a imaginii și se poate folosi aceasta, în locul tuturor valorilor. Așadar, rezultă niște caracteristici comprimate, acestea reducând dimensiunea datelor și îmbunătățind performanțele clasificatorului. Operația de agregare a caracteristicilor poartă denumirea de *pooling* (*mean pooling* pentru cazul în care se folosește media, respectiv *max pooling* pentru maxim).

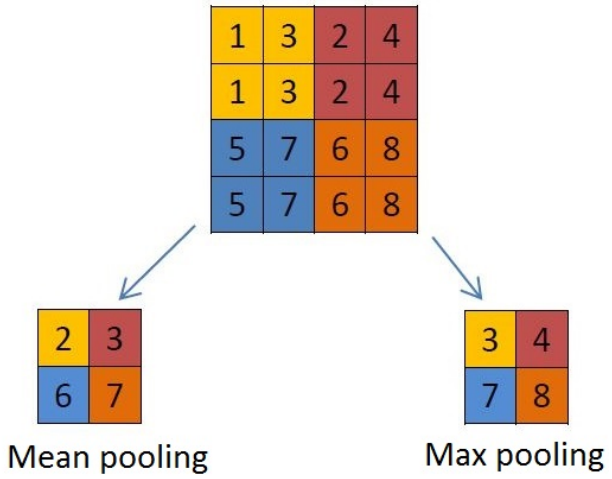


Fig. 8.2 Operația de pooling.

În figura 8.2 avem exemplificarea operațiilor de mean pooling (stânga) și max pooling (dreapta). Fiecare element din straturile de tip pool este obținut din matricea de intrare, din zona cu aceeași culoare. În acest exemplu, din matricea de intrare de dimensiuni $8 \times 8$ se obține rezultatul cu dimensiunea $2 \times 2$.

## 8.3 Rețeaua neurală convoluțională

O rețea neurală convoluțională este alcătuită din unul sau mai multe straturi convoluționale (cu sau fără straturi de pooling) și unul sau mai multe straturi complet conectate. Arhitectura unei rețele neurale convoluționale este proiectată să profite de structura bidimensională a imaginilor folosind conexiuni locale, ponderi share-uite și straturi de tip pooling, acestea producând caracteristici invariante la translație. Un alt avantaj al acestui tip de rețele este acela că sunt mai ușor de antrenat, necesitând mai puțini parametri decât rețelele complet conectate echivalente.

Intrarea unui strat convoluțional este o imagine de dimensiunea $m \times m \times r$, unde $m$ este înălțimea și lățimea imaginii, iar $r$ este numărul de canale (ex. imaginile RGB au 3 canale: roșu, verde și albastru). Stratul convoluțional va avea $k$ filtre de dimensiunea $n \times n \times q$, unde $n$ este mai mic decât dimensiunea imaginii și $q$ poate fi egal sau mai mic decât $r$. Dimensiunea filtrelor ne dă și dimensiunea rezultatului convoluției, denumit *feature map*. Aceste feature maps au dimensiunea $m - n + 1$. Fiecărui feature map i se aplică un strat de tip pooling asupra unei zone continue de dimensiune $p \times p$, în general $p$ având valori cuprinse

între 2 pentru imagini mici și 5 pentru imagini mai mari.

Figura 8.3 ilustrează un exemplu de arhitectură pentru o rețea neurală proiectată să clasifice cifre scrise de mână. Aceasta are două straturi convoluționale, fiecare cu câte un strat de tip pooling, un strat ascuns și stratul de ieșire.

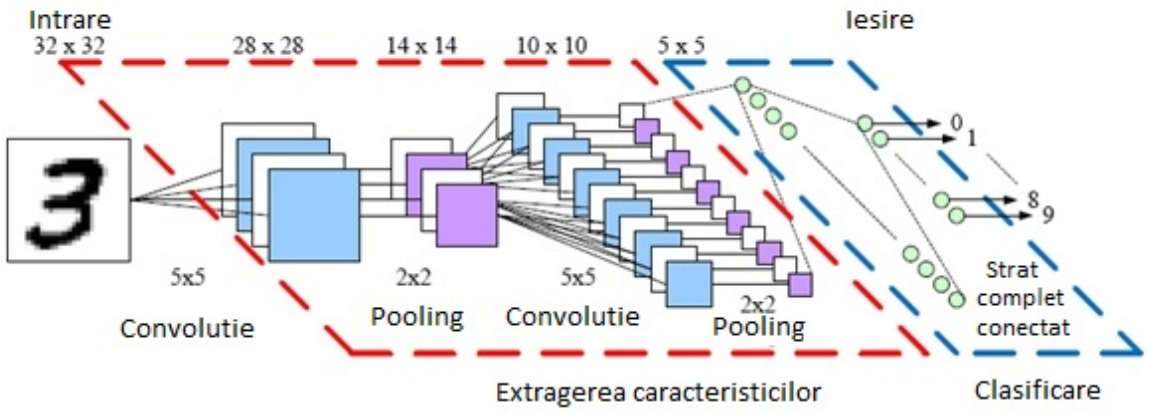


Fig. 8.3 Arhitectura rețelei neurale convoluţională.

### 8.3.1 Propagarea înapoi a erorii în stratul convoluțional

Dacă $\delta^{(l+1)}$ este eroarea pentru stratul $(l+1)$ din rețeaua având funcția de cost $J(W, b; x, y)$, unde $(W, b)$ sunt parametrii, iar $(x, y)$ sunt datele de antrenare și stratul $l$ este complet conectat la stratul $(l+1)$, atunci vom avea eroarea în stratul $l$:

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \bullet f'(z^{(l)}), \tag{8.2}$$

iar gradientul este dat de:

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T, \tag{8.3}$$

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}. \tag{8.4}$$

Dacă, în schimb, stratul $l$ este convoluțional, avem eroarea dată de:

$$\delta_k^{(l)} = \text{upsample}\left((\Theta_k^{(l)})^T \delta_k^{(l+1)}\right) \bullet f'(z_k^{(l)}), \tag{8.5}$$

unde, indicele $k$ se referă la filtrul cu numărul $k$, iar $f'(z_k^{(l)})$ este derivata funcției de activare. Operația upsample are rolul de a propaga eroarea prin stratul de tip pooling, prin calcularea erorii în raport cu fiecare unitate din stratul pooling.

Gradientul în stratul convoluțional este dat de:

$$\nabla_{\Theta_k^{(l)}} J(W, b; x, y) = \sum_{i=1}^{m} (a_i^{(l)}) * \text{rot90}(\delta_k^{(l+1)}, 2), \tag{8.6}$$

$$\nabla_{b_k^{(l)}} J(\Theta, b; x, y) = \sum_{a,b} (\delta_k^{(l+1)})_{a,b}. \tag{8.7}$$

unde, $a^{(l)}$ este intrarea pentru stratul $l$ și $a^{(1)}$ este imaginea de intrare. Operația $(a_i^{(l)}) * \delta_k^{(l+1)}$ este convoluția dintre intrarea $i$ din stratul $l$ și eroarea în funcție de filtrul $k$.

## 8.4 Cerinţe

Folosind Python 3.5 și modulul TensorFlow, vom construi o rețea neurală convoluțională pentru a recunoaște cifre scrise de mână din baza de date MNIST.

### 8.4.1 Instalarea modulului TensorFlow

TensorFlow este o librărie de tip open source dezvoltată de către Google, fiind folosită pentru aplicații cu tablouri multidimensionale de date. Inițial, a fost dezvoltată pentru aplicații ale rețelelor neural, dar poate fi folosită într-o arie extinsă de domenii. TensorFlow poate fi instalată pentru Python folosind utilitarul pip, astfel: `pip install --upgrade tensorflow` . Pentru a funcționa, TensorFlow are nevoie de Python versiunea 3.5.x.

### 8.4.2 Baza de date MNIST

Baza de date MNIST conține 70000 de imagini cu cifre scrise de mână organizate în două seturi: unul de 60000 de imagini de antrenare și unul de 10000 de imagini de test. Aceasta se găsește la adresa: http://yann.lecun.com/exdb/mnist/ . Această bază de date este folosită pentru testarea unor algoritmi de inteligență artificială, pe site aflându-se și o listă cu unii dintre acești algoritmi și performanța obținută.

# Bibliografie

[1] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.