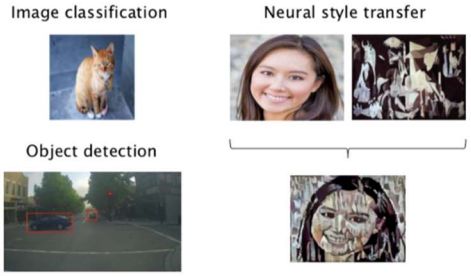
09 - COMPUTER VISION AND CONVOLUTIONAL NETWORKS

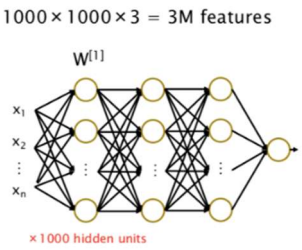
Le reti convolutive sono state principalmente create per supportare computer vision

Computer Vision problems



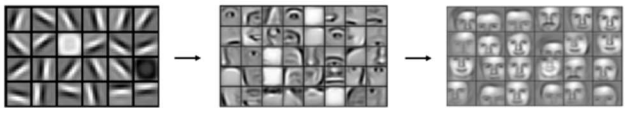
Problems of Computer Vision

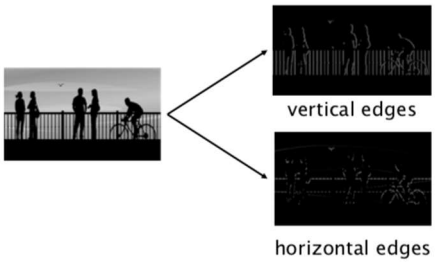
Gli input possono essere davvero grossi



Una soluzione sta nella convoluzione

CONVOLUZIONE

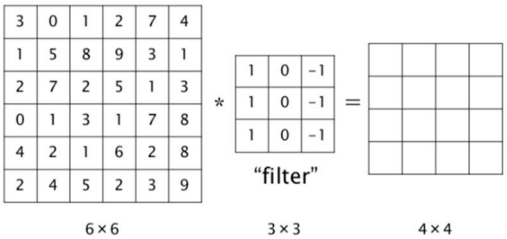




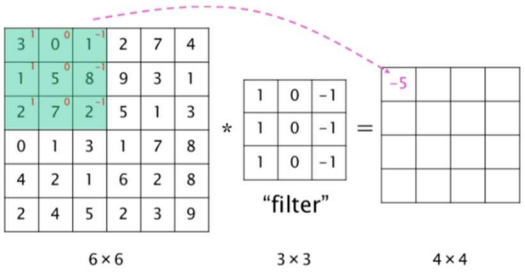
Abbiamo detto che ci sono delle aree che vengono stimolate dagli spigoli che convergono a qualcosa di sempre più complesso fino ad arrivare a risolvere il task che ci siamo posti

Vertical edge detection

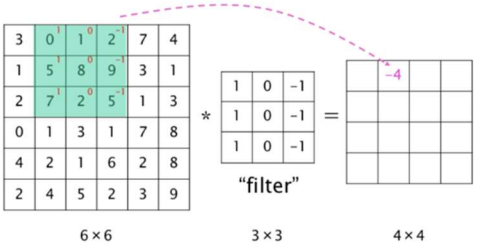
Abbiamo per esempio un'immagine 6x6



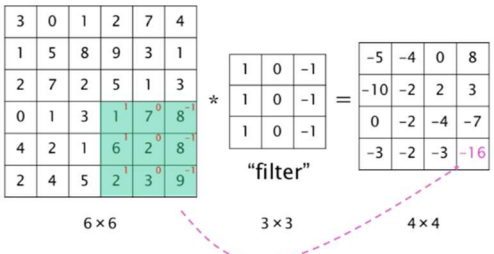
Per fare la edge detection ne facciamo la convoluzione con una matrice 3x3 che è un filtro configurato in maniera intenzionale. Il simbolo non è da confondere con il normale prodotto.
Si ottiene da questa operazione una matrice 4x4. L'operazione è quella di prendere il filtro e metterlo su una regione 3x3 della matrice di partenza moltiplicando gli elementi sovrapposti e facendo infine la somma di tutti i valori della regione.



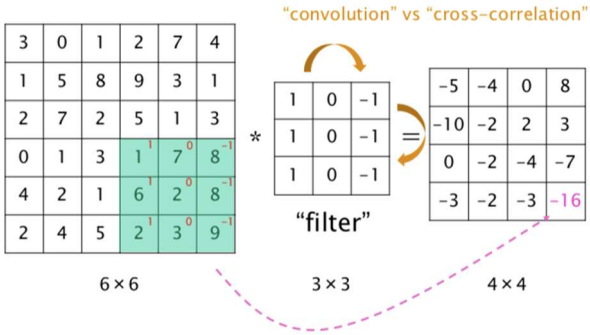
Sposto il filtro di una posizione (lo farò 16 volte)



Per arrivare alla fine

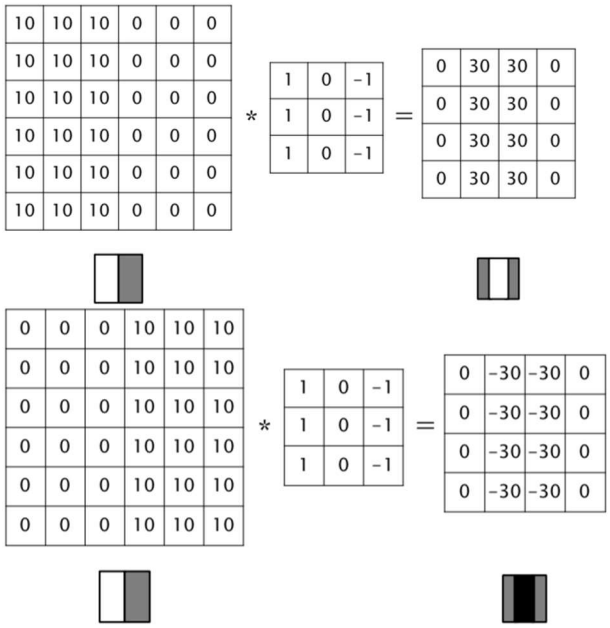


Si ottiene quindi una matrice di dimensioni ridotte rispetto a quella iniziale

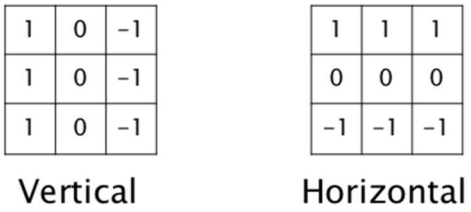


In realtà questa operazione in geometria è chiamata **cross-correlazione**, mentre la convoluzione richiederebbe di andare anche a flippare in orizzontale la matrice. Nel nostro ambito si usa comunque la denominazione di convoluzione.

Quindi se per caso ci troviamo con un edge nell'immagine otteniamo un risultato del genere:



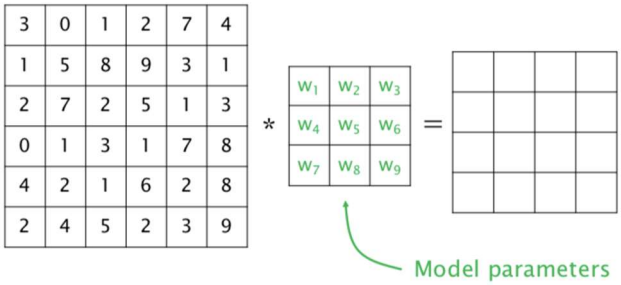
Allo stesso modo esiste il filtro per gli horizontal edges



Nel tempo si è discusso molto sui numeri da utilizzare in questi filtri, ma alla fine ciò che interessa è che sia la rete ad impararli.

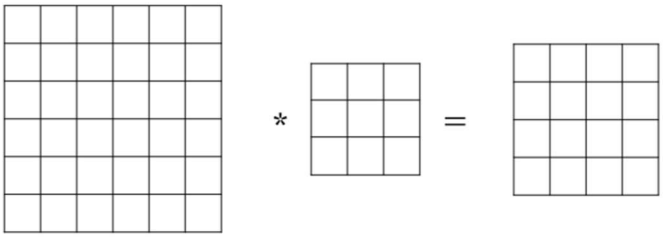
Learning to detect e edges

Quello che si fa è trattare i valori del filtro come **parametri** (weights) per fare in modo che sia la rete ad impararli.

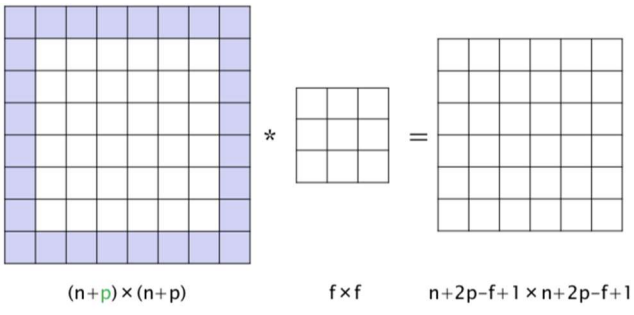


Padding

Le dimensioni dell'output con la convoluzione si riducono secondo la seguente relazione



Gli elementi più periferici della griglia non contribuiscono allo stesso modo per l'uscita. Posso aggiungere uno spazio esterno alla mia griglia di partenza e lo inizializzo a 0. Quando applico la prima volta, il filtro → produco un'immagine di uscita che non è ridotta.



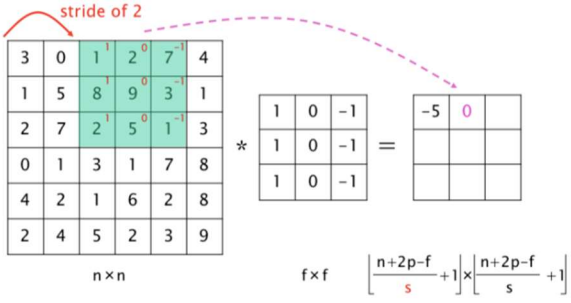
(nella prima immagine dovrebbe essere (n+2p)x(n+2p)

Quanto padding?

- No padding -> **valid** convolution (output si riduce)
- P=(f-1)/2 -> **same** convolution ovvero che l'output è grande quanto l'immagine di partenza (f= dim filtro; P= padding su un lato)
- *Valori tipici di padding dipendono dall'operazione di convoluzione e dalle dimensioni del filtro.*

Strided convolutions

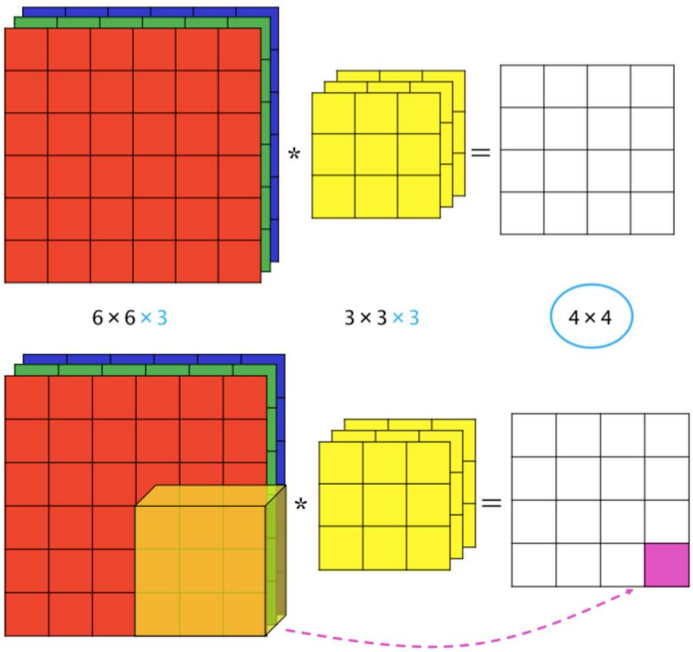
Normalmente quando applichiamo il filtro spostiamo la sovrapposizione del filtro di una unità.
Si può applicare un passo diverso, esempio 2 → le griglie delle matrici di uscita saranno ancora più piccole.



In questo caso se non c'è il padding, l'ultima colonna non viene calcolata nella convoluzione (in quanto con passo di 2, uscirei dalla griglia).

Convoluzione sulle immagini RGB

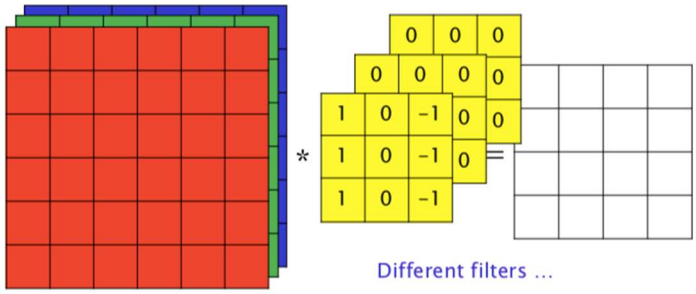
Con le immagini RGB (3 canali), si aggiunge la terza dimensione anche al filtro



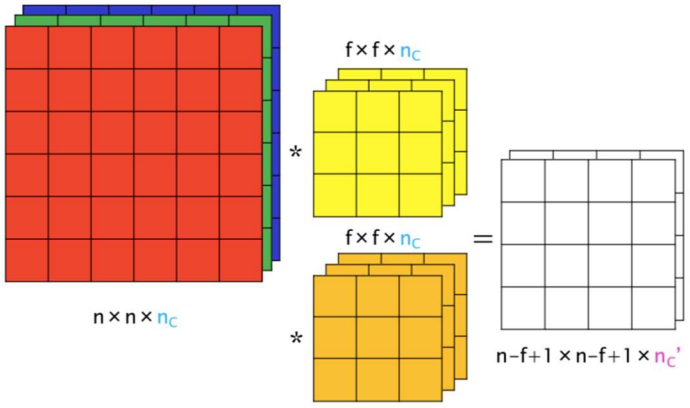
→ ottengo una griglia 4x4

Si prende il volume dove si somma tutto (combinazione lineare di 27 valori)

Per ogni foglio di filtro **posso avere un filtro diverso**, quindi diverso per ogni canale RGB.
Il numero di parametri per il filtro ora non sarà più fxf ma **fxfxn$_c$** (dimensioni filtro per numero di canali).



Si possono applicare anche più filtri ottenendo più immagini in uscita tante quante il numero di filtri

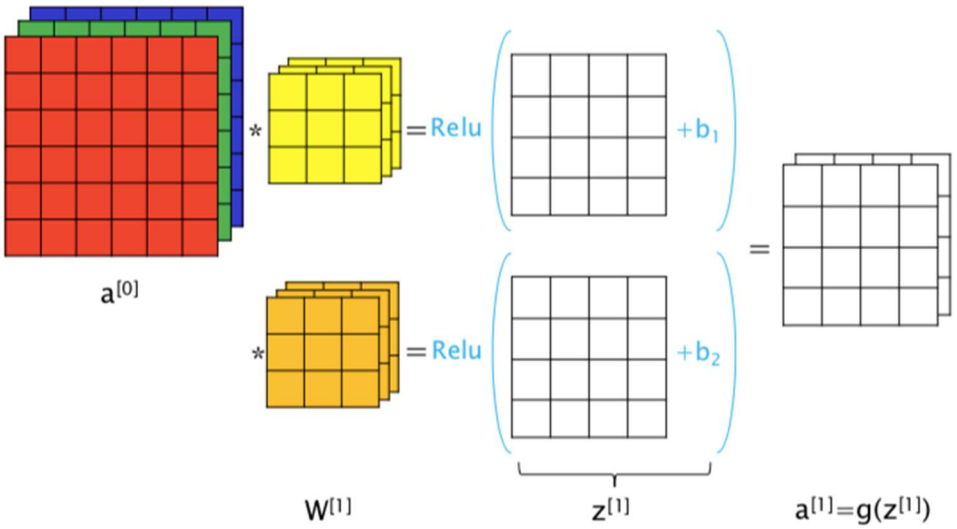


→ ognuno di questi filtri produce un'immagine 4x4 quindi queste due immagini vengono combinate tra di loro → 4x4x2

n_c' è il nuovo numero di canali = # di filtri

Layer convoluzionale

Immaginiamo di avere la nostra immagine RGB con un certo numero di filtri e di voler costruire un layer della rete convoluzionale.



→blocco alto: calcola una determinata feature/caratteristica dell'immagine

→ blocco basso: calcola un'altra feature

→ in uscita : attivazioni della rete neurale

Questo può rappresentare il livello di una rete convoluzionale durante una forward propagation dove:

- l'immagine rappresenta l'attivazione al livello precedente $a^{[l-1]}$
- i filtri con i pesi $W^{[l]}$ che impara la rete
- il risultato della convoluzione $W^{[l]}a^{[l-1]}$ a cui si somma il bias $b^{[l]}$ ottenendo $z^{[l]}$
- a tale risultato si applica una funzione di attivazione tipo la ReLU ottenendo l**a nuove attivazioni $a^{[l]}=g(z^{[l]})$**

Se ad esempio si hanno 10 filtri 3x3x3 si avranno 280 parametri = (3x3x3+1)x10 (il +1 è di bias) che non è un alto numero pensando al numero che si avrebbe senza convoluzioni.

Notazione

Consideriamo il livello convoluzionale "l", devo scegliere:

- dimensione filtro
- dimensione padding
- dimensione stride

$f^{[l]}$ = filter size
$p^{[l]}$ = padding
$s^{[l]}$ = stride

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$

Output: $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

$$n_{H/W}^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$n_C^{[l]}$ = #filters

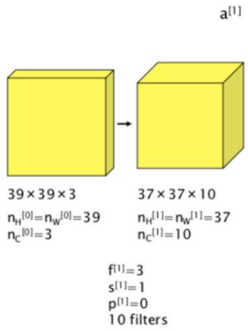
Each filter: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$

Activations: $A^{[l]} = m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

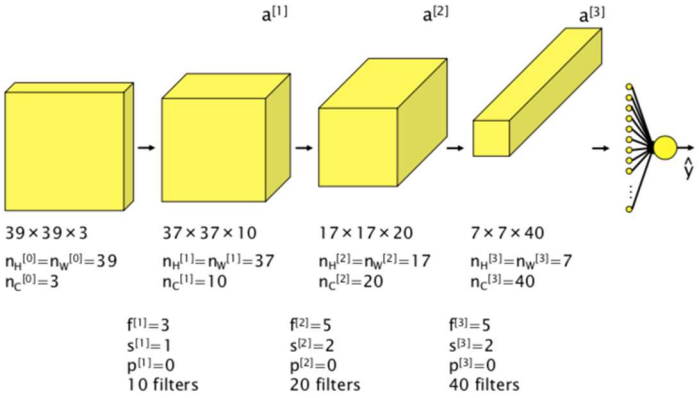
Weights: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$ Bias: $n_C^{[l]}$

Esempio di rete convoluzionale (ConvNet)

Ho questa immagine 39x39x3 a cui applico 10 filtri 3x3 e ottengo le attivazioni del primo livello, ovvero una matrice 37x37x10.



Continuando, applico 20 filtri 5x5



Alla fine, si ottengono 1960 parametri. Facendo poi l'unrolling, si possono mandare ad un nodo per fare ad esempio la logistic regression.

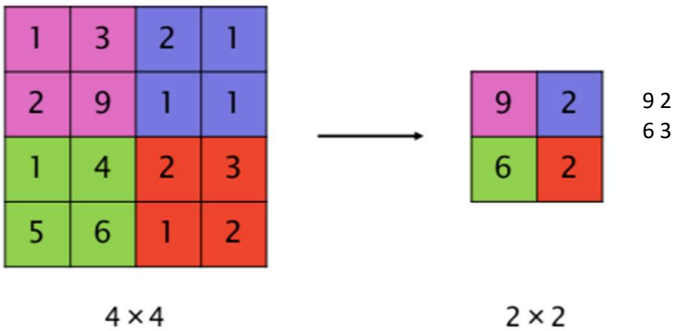
È comune veder crescere la profondità e decrescere le dimensioni delle matrici.
La scelta degli **hyper-parametri** come il numero di livelli, la dimensione dei filtri, lo stride, il padding, il numero dei filtri che si sommano ad altri già visti come il learning rate, i parametri di regolarizzazione, etc… ha le stesse regole.

Tipi di livelli nelle ConvNet

- Convoluzionale (CONV)
- Pooling (POOL)
- Fully connected (FC)

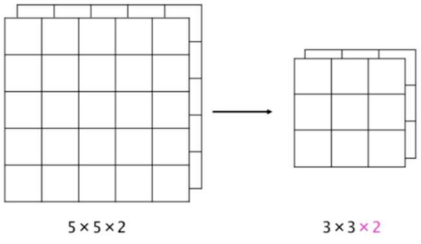
Livello di pooling: Max Pooling

Divide la matrice in sottoregioni. Se cerco il max pooling mette nell'output il massimo di ogni regione.



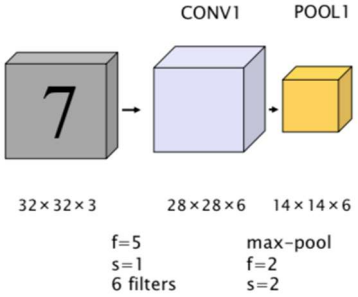
- È come se andassi ad applicare un filtro con f=2 e s=2
- In questo caso gli hyper-parametri sono f e s, ma non ci sono parametri da scegliere o da apprendere.

Serve a capire se in una regione dell'immagine c'è una certa informazione e a mantenerla.
Una particolarità del pooling è che la dimensione dei canali rimane invariata rispetto alla convoluzione perché processati in maniera separata.



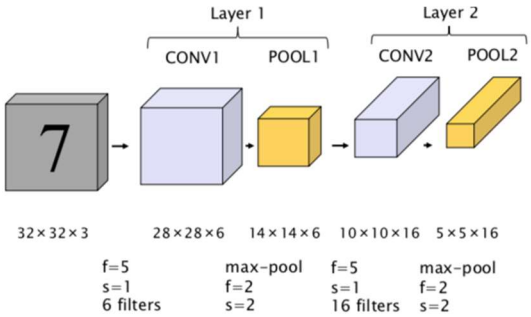
Si può anche avere l'average pooling

Digit recognition (LeNet-5) – esempio di max pooling

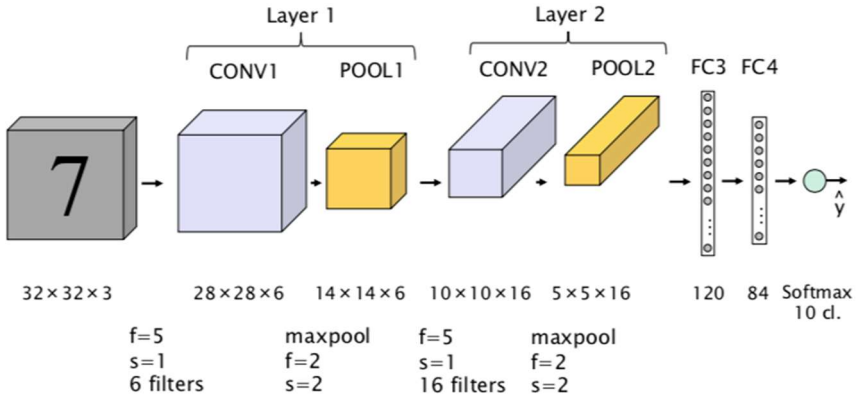


Nota: L'obbiettivo del poooling è quello di compattare la rappresentazione

Applico una convoluzione e poi applico un max-pooling per ridurre le dimensioni. Entrambi sono il **layer 1 perché il pooling non ha parametri**.



Applichiamo nuovamente una convoluzione seguita da un pooling ottenendo il layer 2
Da qui otteniamo 400 output che mandiamo dentro 120 unit e poi dentro a 84 unit e infine tutto dentro ad un classificatore softmax.



La cosa interessante è che rispetto all'utilizzo originale si usano 3 canali anziché uno e si applicano diversi livelli con filtri facendo poi riduzione con i pooling.

Questa tabella sintetizza i calcoli:
Per il conv1 si prende la dimensione del filtro f=5, il numero dei canali del livello precedente n_c=3 moltiplicandoli, si somma il bias=1, e infine si moltiplica tutto per il numero di filtri n_c'=6 → (5x5x3+1)x6
La rete ha circa 60000 parametri

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	456
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	2,416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,120
FC4	(84,1)	84	10,164
Softmax	(10,1)	10	850

(5*5*3+1)*6=456 → 6 filtri (5x5 da 3 volumi + 1 bias);
pool che divide per 4

livello fully connected
livello fully connected

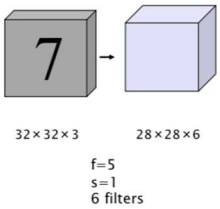
Le convoluzuoni hanno quindi permesso una diminuzione dei parametri

Perché le convoluzioni?

Il primo motivo è il **parameter sharing**: pensiamo al edge recognition dove il nostro filtro può essere applicato in varie parti dell'immagine; quindi, lo stesso filtro e quindi gli stessi parametri possono essere usati in più punti dell'immagine. Questo li rende condivisi.

La seconda caratteristica è la **sparsity of connections**: se vediamo un'uscita dell'applicazione di un filtro abbiamo che un valore dipende solo dalla sottoregione di input e quindi questo è qualcosa che va a ridurre le dimensioni in gioco.
Ogni livello dipende solo da un sotto-insieme dellìinput

Se prendiamo l'esempio precedente



"Standard" network
3,072 × 4,704 parameters

→tutte le combinazioni tra le due

ConvNet
(5 × 5 × 3+1) × 6=456 parameters

Less parameters, less prone to overfitting

È ben chiara la differenza di parametri dall'applicazione classica a quella **convolutiva: meno parametri → reti meno predisposte ad overfittare.**