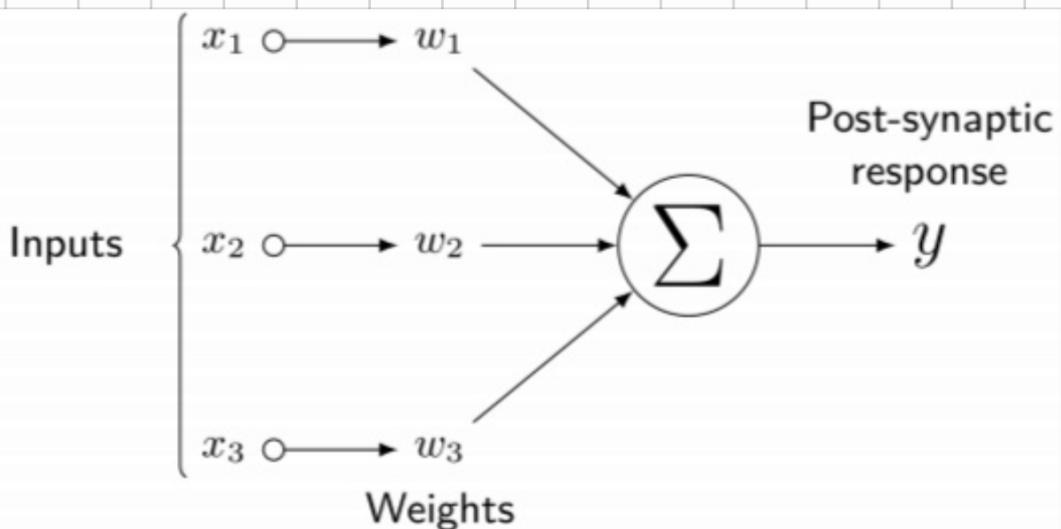


UNA DELLE PRIME IDEE DI APPRENDIMENTO CONCRETE È LA **Regola di Hebb**:

SE UNA CELLULA A CONTRIBUISCE CONSISTENTEMENTE ALL'ATTIVITÀ DELLA CELLULA B, ALLORA LA SINAPSIS DA A VERSO B DEVE ESSERE RINFORZATA.

In altre parole:

NEURONI CHE SPEDISCONO INSIEME DEVONO ESSERE COLLEGATI, NEURONI CHE NON SPEDISCONO IN SINCRONIA NON SPEDISCONO COMMESSI



**PERCEPTRON**

L'ALGORITMO DEL PERCEPTRON È IL SEGUENTE:

INPUT:  $D = \{(x_i, y_i)\}_{i=1}^N$

OUTPUT: 1 PESI IN APPRESI

. Si inizierà in maniera random  $w_0$

$t \leftarrow 1$

. WHILE (non si ha convergenza):

FOR  $(\bar{x}, y) \in D$

$$\hat{y} = f(\bar{w}^T \bar{x})$$

$$\bar{w}_t \leftarrow w_{t-1} - \eta(y - \hat{y}) \bar{x}$$

$$. t = t + 1$$

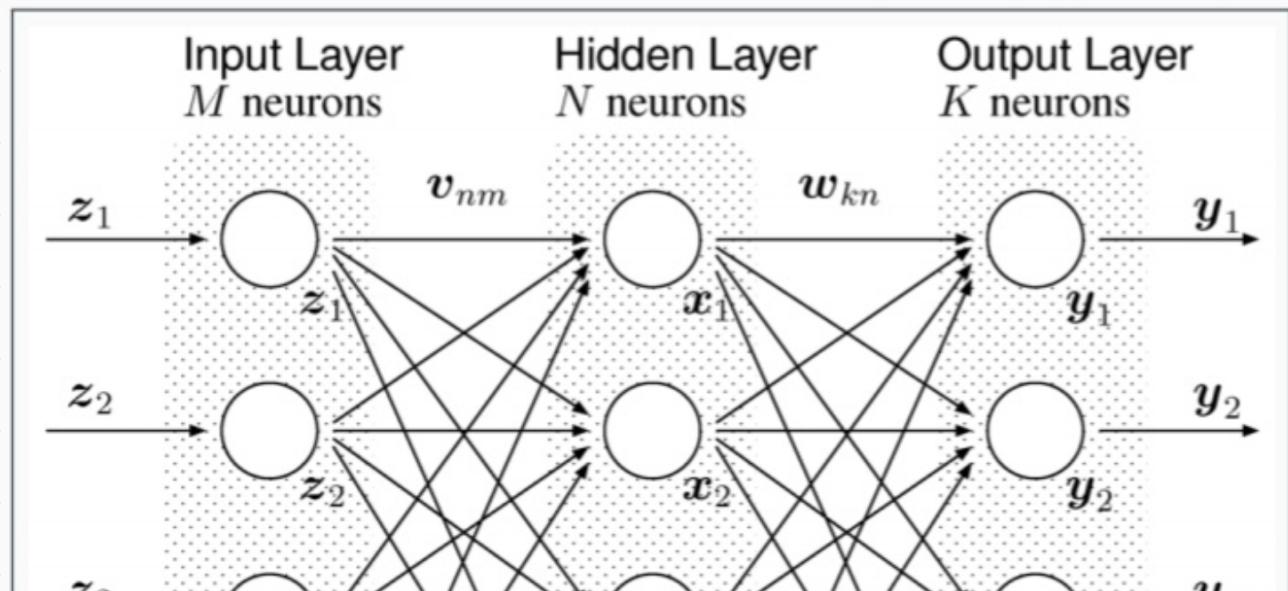
» Funzione di attivazione  $f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & \text{altrimenti} \end{cases}$

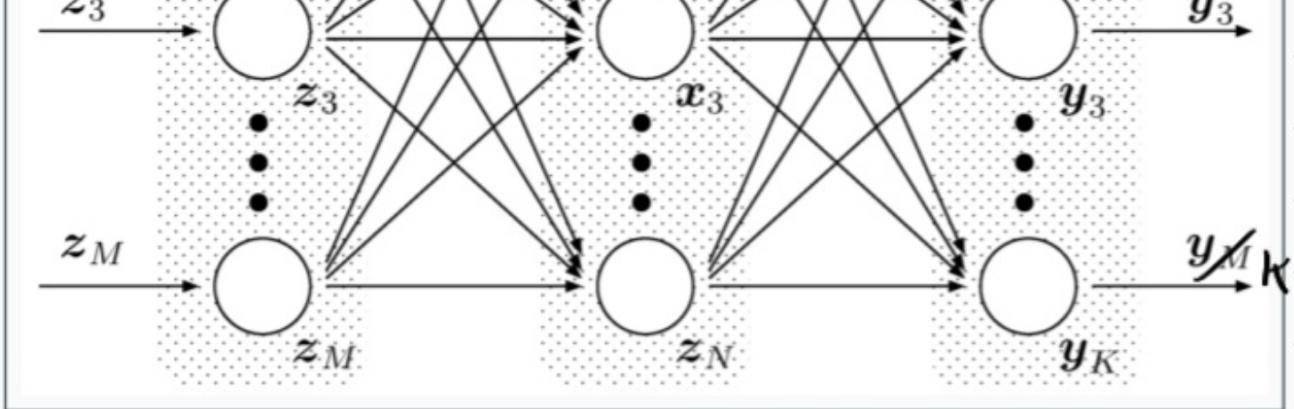
! LEARNING RATE

SINGOLO GRADIENTE  $\in \{-1, 0, 1\}$

NELLA SUA SIMPLICITÀ IL PERCEPTRON È MAI UTILE MA TUTTI NOI NON PUÒ  
RISOLVERE PROBLEMI DI CLASSIFICAZIONE SEPARABILI!

### MULTILAYER PERCEPTRON





Dove l'equazione nel caso di un unico Hidden Layer vale:

$$\hat{y}(x) = \sigma(\bar{w}_2^T \sigma(\bar{w}_1^T x + b_1) + b_2)$$

Dove  $\bar{w}_1$  e  $\bar{w}_2$  sono le matrici dei pesi.

ritroviamo da questo equazione che, tranne che per la funzione di attivazione  $\sigma$ , questo è un sistema lineare

Alcuni esempi di funzioni di attivazione comuni:

- $\sigma(x) = \tanh(x)$
- $\sigma(x) = (1 + e^{-x})^{-1}$
- $\sigma(x) = \frac{\exp(x)}{\sum_i \exp(i)}$  (softmax, used for **outputs**).

Al fine di accrescere il nostro si scrive il seguente approccio:

- Si scrive una **loss function**:

NLL

$$L(y, \hat{y}(x)) = -\frac{1}{C} \sum_i y_i \log(\hat{y}_i)$$

- SI ESEGUONO UNA DISCESA DEL GRADIENTE RISPETTO A TUTTI I PARAMETRI DEL MODELLO:

$$\begin{aligned}\theta_{m+1} &= \theta_m - \epsilon \nabla_{\theta} L(y, \hat{y}(\bar{x})) \\ &= \theta_m - \epsilon \sum_{i=1}^N \frac{1}{N} \nabla_{\theta} L(y_i, \hat{y}(\bar{x}_i))\end{aligned}$$

↳ LEARNING RATE

*nuovi parametri*

L'INCORRITURA STANDARD PER QUESTO È LA BACKPROPAGATION.

L'IDEA È QUELLA DI APPLICARE LA CHAIN-RULE. Dopo quando l'ESEMPLARE DEL MLP SOLO UN UNICO HIDDEN LAYER:

↗ PREDIZIONE DEL MODELLO

$$\hat{y}(\bar{x}) = \sigma(\bar{w}_2^T \sigma(\bar{w}_1^T \bar{x} + b_1) + b_2)$$

con una certa funzione  $f(\bar{x}; \theta)$  che definisce le RETE Parametrizzata da  $\theta$ :

$$\theta = (\bar{w}_1, b_1, \bar{w}_2, b_2)$$

con 6 certe FUNZIONI DI ATTIVAZIONE NON LINEARI.

Dobbiamo pure calcolare:

$$V_{\theta}(y - f(\bar{x}; \theta)) = V_{\theta}\left(y - \delta\left(\bar{W}_2^T \delta\left(\bar{W}_1^T \bar{x} + b_1\right) + b_2\right)\right)^2$$

$$= -2\left(y - \delta\left(\bar{W}_2^T \delta\left(\bar{W}_1^T \bar{x} + b_1\right) + b_2\right)\right) V_{\theta} \delta\left(\bar{W}_2^T \delta\left(\bar{W}_1^T \bar{x} + b_1\right) + b_2\right)$$

ESEMPPIO!

LA TECNICA DEL BACKPROPAGATION, PER QUANTO SI PUÒ, PRESENZA DI UN PROBLEMA DURANTE L'ADDESTRAMENTO

. **SATURATING ACTS:** quando i nodi della rete raggiungono uno stato in cui l'output è vicino agli estremi delle funzioni di attivazione, con il caso del sigmoid. quando l'input di cui i nodi raggiungono valori limiti la funzione di attivazione satura, producendo (nel caso del sigmoid) output vicini a 0 o 1.

Nel caso di saturazione si degrada la funzione di attivazione perché quasi zero, rendendo difficile aggiornare gli pesi  $\Rightarrow$   rallentamento apprendimento.

. **VANISHING GRADIENTS:** quando i gradienti diventano molto piccoli ed è difficile aggiornare i pesi. quando si usano funzioni di attivazione con derivate limitate, quando i gradienti diventano piccoli gli aggiornamenti dei pesi diventano trascurabili.

Funzioni di attivazione con derivate lineare generano gradienti piccoli!

POSSIBILI SOLUZIONI Sono le funzioni di attivazione diverse (ReLU), oppure riguardanti come la normalizzazione.

NEL CASO IN CUI IL NUMERO DI TRAINING SAMPLES È MOLTO LARGO SI RISOLVE

D) MILITARE NUOVO.  $\Rightarrow$  SI APPOROSSIMA IL VERO GRADIENTE

## C) AVVISO STOCHASTIC GRADIENT DESCENT

. SI CREA UN VETTORE INIZIALE DI PESI  $\theta$  E LEARNING RATE  $\eta$ .

. SI RIPETE FINO A CHE NON SI TROVA UN MINIMO APPROSSIMATO:

① SI MISURANO DIVERSI I CASI UNI  $D$

② PER  $(\bar{x}, y) \in D$

$$\theta = \theta - \eta \nabla_{\theta} \mathcal{L}(\{\bar{x}, y\}; \theta)$$



## Prima di continuare su YouTube

Prima di continuare su YouTube Accedi Una società Google Accedi Prima di continua...

[youtube.com](https://youtube.com)

## INPUT DATA

LE INFORMAZIONI IN INPUT DOVONO ESSERE DEI VETTORI / TENSORI. È MOLTO

IMPORTANTE NORMALIZZARE LE INFORMAZIONI IN INPUT: AD ESEMPIO, PER LE IMMAGINI, RESTRIZIONE  
LE RANGHE DI VALORI DA  $[0, 255]$  A  $[-1, 1]$  OPPURE  $[0, 1]$

## LOSS FUNCTIONS

## C) CLASSIFICAZIONE BINARIA:

BCE

IN QUESTA CASA SI USUARIA LA BINARIAL-LOSS-FUNCTION.

$$\text{NLL} = - \sum_{i=0}^{|I|} y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)$$

GROUNDS TRUTH

PREDICTION

DONG NEL CASO IN CUI LA PREDIZIONE È CORRETTA SI HA CHE  $y_i = \hat{y}_i$  E QUINDI IL  
VALORE DELLA LOSS È NULLO!

### CLASSIFICAZIONE MULTICLASSE:

IN QUESTO CASO SI UTILIZZA LA **CROSS-ENTROPY = NEGATIVE-LOG-LIKELIHOOD**

$$- \sum_{i=0}^{|I|} \left\{ \sum_{h=0}^{|H|} y_{i,h} \log \hat{y}_{i,h} \right\}$$

- CLASSIFICAZIONE BINARIA: BCE
- CLASSIFICAZIONE MULTICLASSE: NLL
- REGRESSIONE: MSE

SI UTILIZZA **MEAN-SQUARED-ERROR**:

$$\sum_{i=0}^{|I|} \|y_i - \hat{y}\|^2$$

### BUILDING BLOCKS

UNO STRATO IN CUI TUTTI I NODI SONO CONNESSI A TUTTI (NON SONO STRATI PRECEDENTI E SUCCESSIVI).

DEFINISMO UN **FULLY-CONNECTED-LAYER** CON FUNZIONE LINEARE  $f(x) = w^T x + b$

DEFINISMO IN TERMINI DI PARAMETRI IN INGRESSO (**FAN-IN**) E PARAMETRI DI USCITA (**FAN-OUT**)

$D_{in}, D_{out} = 768, 10$   
`torch.nn.Linear(D_in, D_out)`

$$\text{weights} = \begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \\ w_{2,0} & w_{2,1} \end{bmatrix} \quad \text{bias} = \begin{bmatrix} b_0 & b_1 \end{bmatrix}$$

PER QUANTO RIGUARDA LE VARIIE FUNZIONI DI ATTIVAZIONE DELLA RETE SI HANNO:

. Sigmoid:

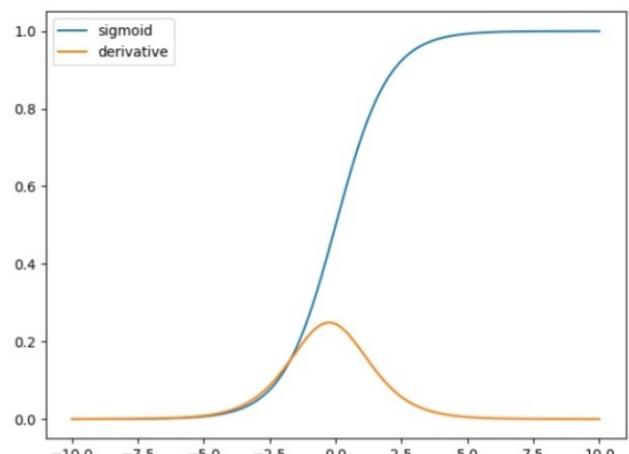
$$\delta(x) = \frac{1}{1+e^{-x}}$$

CHE TUTTISSIMO HA IL PROBLEMA DI SATURAZIONE

IL GRADIENTE È VIGENTE QUINDI UTILIZZARO

SOTTO CONCETTO UNA FUNZIONE DI ATTIVAZIONE

NELLA CLASSIFICAZIONE BINARIA



. ReLU:

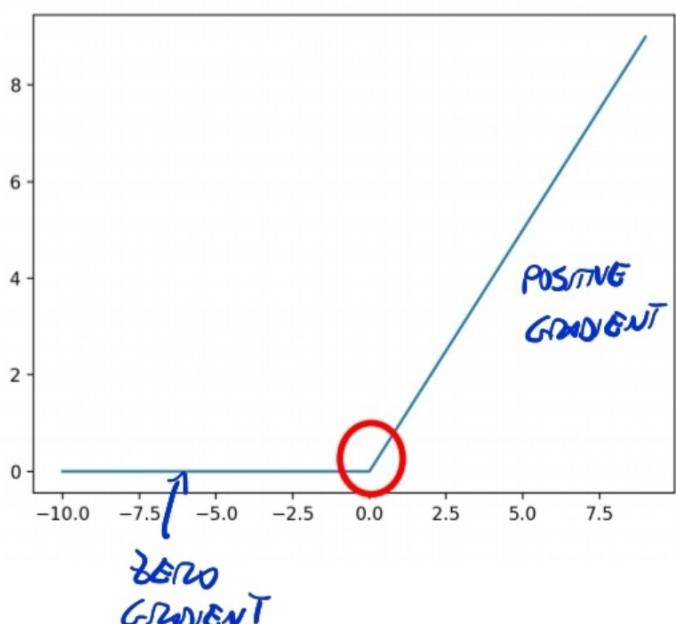
$$g(x) = \max(0, x)$$

PERMETTE DI INTRODURRE UN NON-LINEARITÀ

SENZA ANDARE A SATURAZIONE. TUTTISSIMO NON

È DIFFERENZIABILE IN ZERO! E TUTTI

I VALORI NEGATIVI SONO SCARICI!



## OPTIMIZATION

AL PRIMO DI IMPARARE occorre inizializzare gli starting points di tutti i parametri della rete. Alcune guida suggeriscono di inizializzare in maniera random i pesi dei parametri sulla base della specifica architettura.

In PyTorch l'inizializzazione per layer  $i$  dato da:

$$W_{i,j} \sim U\left(-\frac{1}{\sqrt{m}}; \frac{1}{\sqrt{m}}\right)$$

INPUT SIZE DEL LAYER

QUESTA IDEA consiste quindi nel campionare ciascun peso da un distribuit uniforme in modo da ottenere una varianza del gradiente uniforme su tutta la rete

I due algoritmi di ottimizzazione più noti e utilizzati sono:

- **MINIBATCH SGD** → lo step dipende solo dalla norma del gradiente

---

### Algorithm 8.1 Stochastic gradient descent (SGD) update

---

Require: Learning rate schedule  $\epsilon_1, \epsilon_2, \dots$

Require: Initial parameter  $\theta$

$k \leftarrow 1$

while stopping criterion not met do

    Sample a minibatch of  $m$  examples from the training set  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  with corresponding targets  $\mathbf{y}^{(i)}$ .

    Compute gradient estimate:  $\hat{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

    Apply update:  $\theta \leftarrow \theta - \epsilon_k \hat{g}$

$k \leftarrow k + 1$

end while

---

- **MOMENTUM** → lo step dipende da quanto sono larghi gli allungati i gradienti consecutivi

---

### Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

---

Require: Learning rate  $\epsilon$ , momentum parameter  $\alpha$

Require: Initial parameter  $\theta$ , initial velocity  $v$

while stopping criterion not met do

while stopping criterion not met do

Sample a minibatch of  $m$  examples from the training set  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  with corresponding targets  $\mathbf{y}^{(i)}$ .

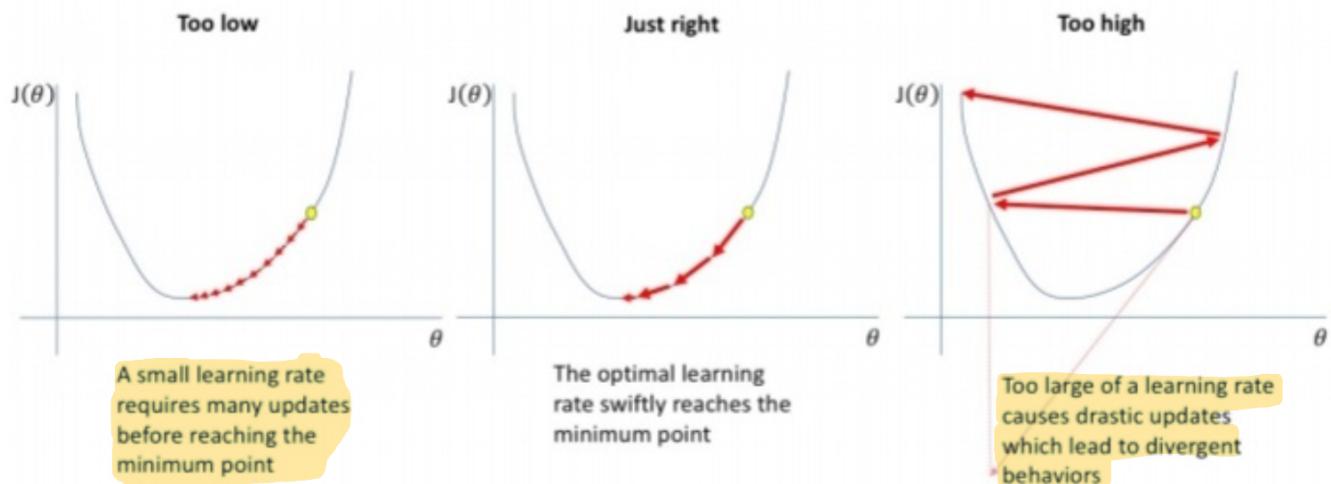
Compute gradient estimate:  $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$ .

Compute velocity update:  $\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \mathbf{g}$ .

Apply update:  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}$ .

end while

PER PENSARE NUOVA IL **LEARNING-RATE**:



## DROPOUT

L'OGNA È PUGLIA DI **INIBIRE** IN MANNERA CASUALE UNA UNITÀ CON CUES CERTA PROBABILITÀ. È PUNO UNA TECNICA DI REGOLARIZZAZIONE CHE SERVE A CONTRASTARE L'OVERFITTING.

PUNO UN NODO VENG INIBITO LA SUA USCITA È IMPOSTATA A ZERO E QUINDI, TEMPORANEAMENTE, IL NODO NON CONTRIBUISCE ALLA PROIEZIONE DEL SEGNALE ATTIVATO SOLO AL RETE.

IL DROPOUT INTRODUCE PUNO UNA **VARIABILITÀ** FORZATA DURANTE L'ADDESTRAMENTO COSTRUENDO LA RETE A NON FARLE ECESSIVO AFFIDAMENTO Solo SU STOPIO NOI

# CNN

Sono reti neurali specializzate nell'elaborazione di dati strutturati a "griglia", ad esempio video e immagini. Sono particolarmente efficienti nel Riconoscimento di pattern ricorrenti.

Si parte da una rappresentazione "nascosta"  $H_{i,j}$  per il pixel  $X_{i,j}$  dell'immagine di input  $X$ .

Se abbiammo un Fully Connected layer capace di comporre  $H_{i,j}$  (senza considerare i bias) otteniamo che:

$$H_{i,j} = \sum_h \sum_l W_{i,j,h,l} X_{h,l}$$

width      ↓      height      ↓  
                |      |      PESI

Puesso esservi quindi un Fully Connected layer dove per ciascun pixel della rappresentazione "nascosta" si valutano tutti i pixel in input.

Eseguiamo un cambio di indice:

$$H = i + a$$

$$l = j + b$$

$$\Rightarrow H_{i,j} = \sum_a \sum_b W_{i,j,i+a,j+b} X_{i+a,j+b}$$

E introduciamo

$$V_{i,j,a,b} = W_{i,j,i+a,j+b}$$

$$\Rightarrow H_{i,j} = \sum_a \sum_b V_{i,j,a,b} X_{i+a,j+b}$$

SHIFT IN INPUT

↑ SHIFT RAPPRESENTANTE

INTRODUCIAMO ADesso IL CONCETTO DI TRANSLATION INVARIANCE: UNA SHIFT NELL'INPUT

X DENG PORTARE AD UNA SHIFT NELL'RAPPRESENTAZIONE H

QUINDI L'UNICO MODO PER AVERE INVARIANZA SULLA BASE DELLA POSIZIONE DEL

PIXEL È PEGGIO DI RIMUOVERE I, J DAL TENSORE DEI PESI V:

↪ PER AVERE INVARIANZA SULLA POSIZIONE DEL PIXEL

$$\Rightarrow H_{i,j} = \sum_a \sum_b V_{a,b} X_{i+a, j+b}$$

PUESA È UNA CONVOLUZIONE!

CONVOLUZIONE DISCRETA

$$h(x) \otimes g(x) = \sum_{s=-N}^N h(s)g(x-s)$$

NON HA BISOGNO DELLA IMMAGINE INTERA MA SOLO NELLA REGIONE CENTRALE NEL PIXELE CONSIDERATO

VOLGONO ADesso IMPORTE LA LOCALITÀ ANDA I PRIMI CICLI NON SONO MAI

BISOGNA DI OSSERVARE L'IMMAGINE INTERA AL FINE DI RIUSCIRE A CREARE UNA CERTA

FEATURE.

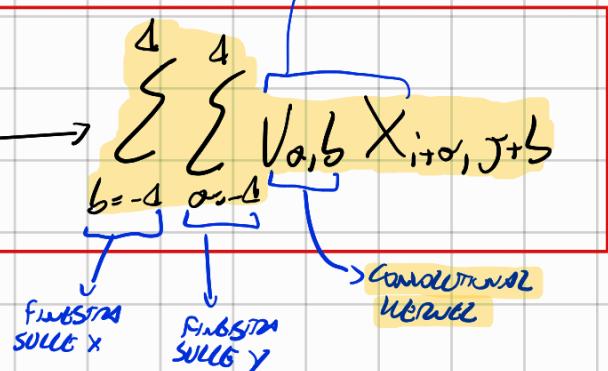
PER IMPORTE QUESTA LOCALITÀ NELLA NOSTRA CONVOLUZIONE DOBBIANO ABBEDIRE

LA RISPOSTA DEL FILTRO AL DI FUORI DI UNA CERTA REGIONE CHE È CENTRATA NEL

PIXEL i, j:

LOCAL INNER PRODOTTO TRA I PESI  
6 IN PIXEL!

$$H_{i,j} = \sum_{b=-\infty}^{\infty} \sum_{a=-\infty}^{\infty} V_{a,b} X_{i+a, j+b}$$



QUINDI, PER LE IMMAGINI, DEFINIAMO IL SEGUENTE CONVOLUZIONE DISCRETA:



input feature  $224 \times 224 \times 3$

Imaging I

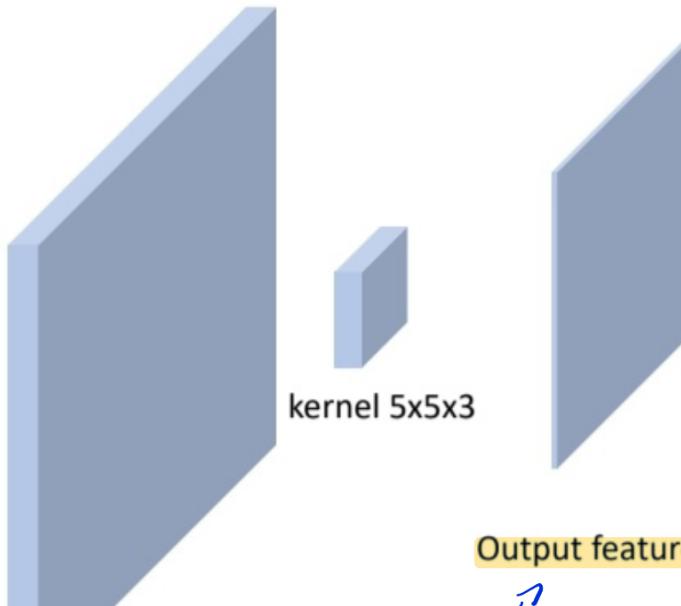


kernel  $5 \times 5 \times 3$

$\hookrightarrow$  convolutional kernel  $H$

$$H(x, y) \odot I(x, y) = \sum_{s_1=-N}^N \sum_{s_2=-N}^N H(s_1, s_2) \cdot I(x-s_1, y-s_2)$$

QUESTO CONVOLUTIVE PRODUCE' PURSO IL SEGUENTE OUTPUT, ANCHE  
UNA FEATURE MAP:



input feature  $224 \times 224 \times 3$

Output feature  $220 \times 220 \times 1$

SI PERDONO 4 PIXEL!

PURSO, DATA UNA CERCA IMMAGING I (TENSORE DEL TERZO ORDINE)

$T \in \mathbb{R}^{W \times H \times 3}$

SE ESTENDIAMO UNA CONVOLUZIONE CON UN SOLO KERNEL

$H \in \mathbb{R}^{n \times n \times 3}$

OTTENIAMO UNA FEATURE MAP CHE E' IL PRODOTTO DELLA CONVOLUZIONE

110/112

ORGANIZZAZIONE DELLE MAPPE DI FEATURA DELL'ARCHITETTURA

CHEG ALI DIMENSIONI

$$[W - (n-1)] \times [H - (n-1)] \times \# \text{ kernels}$$

COME SI CREANO PIÙ MAPPE DI FEATURA?

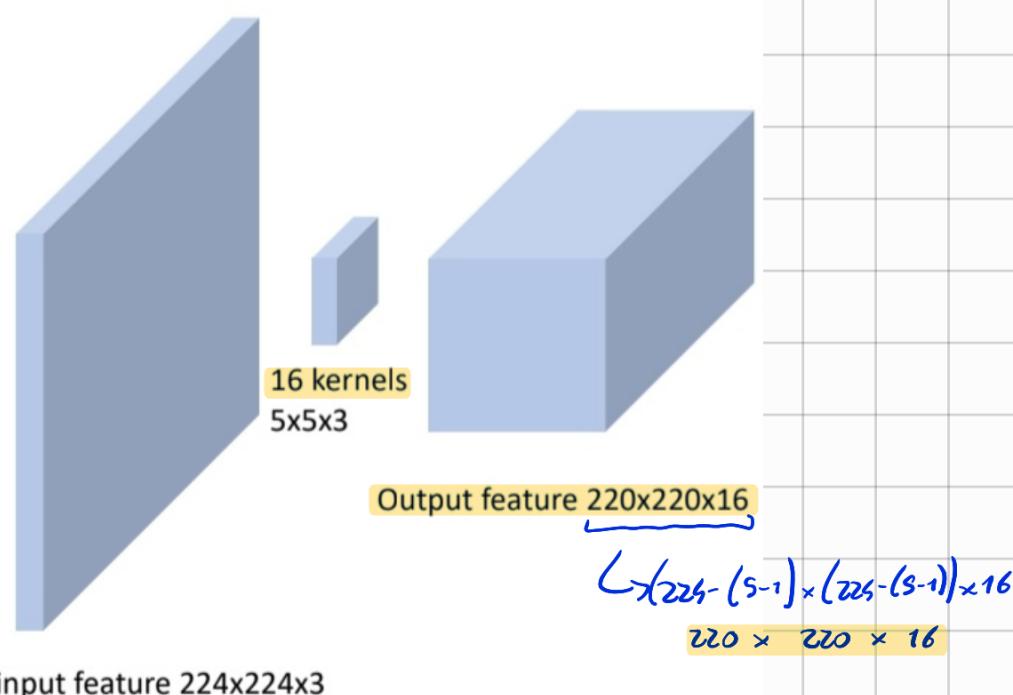
SÌ ESCE CON CONVOLUZIONE MULTIPLA

CON DIFFERENTI KERNELS  $K \in \mathbb{R}^{m \times n \times 3}$

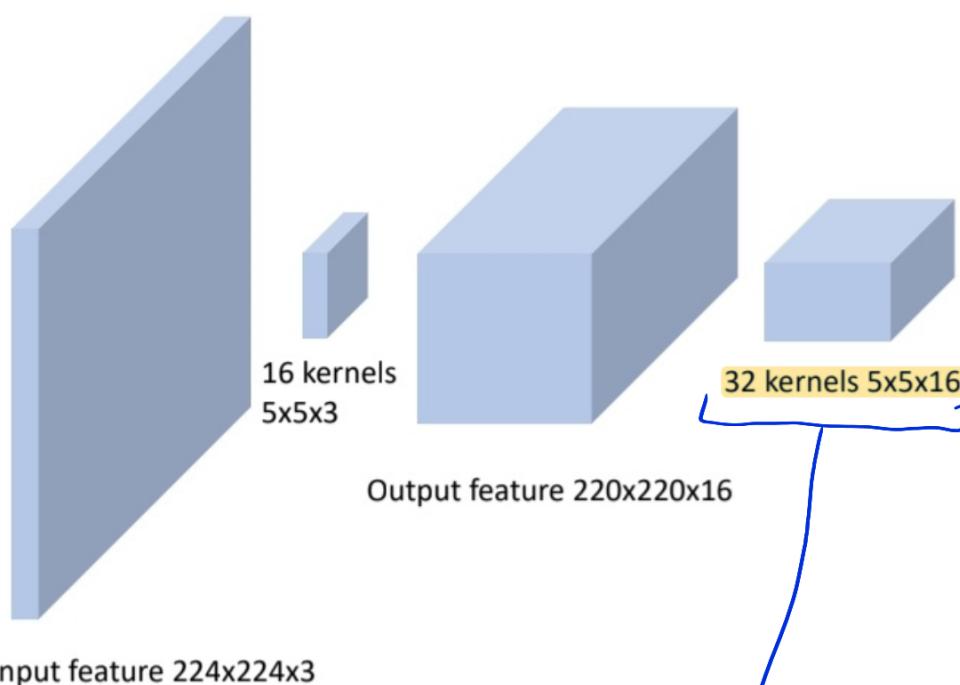
E OTTENGONO QUINDI  $C$  MAPPE DI FEATURA

CON LA STESSA FORMA, POSSIAMO QUINDI ASSEMBLARE QUESTE MAPPE IN UN UNICO

TENSORE CON C CANALI!

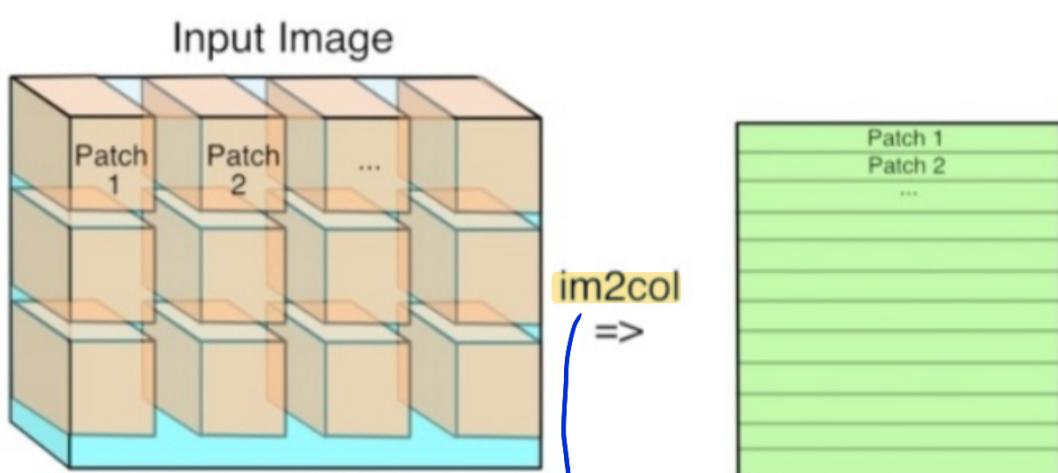


ESEMPPIO:



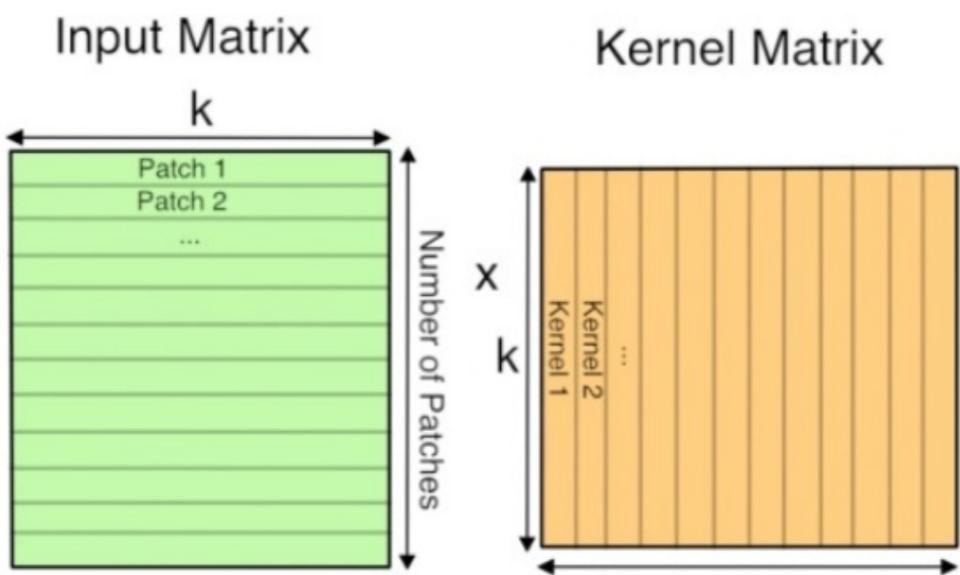
LA MIA FEATURE MAP HA  
DIMENSIONE  $216 \times 216 \times 32$

A LIVELLO IMPLEMENTAZIONE LE CONVOLUZIONI SONO ESEGUITE TRAMITE  
MOLTIPLICAZIONI DI MATRICI:



Transforma il tensor in un vettore  
una dimensione

PUNTO, TRAMITE LE OPPORTUNE TRANSFORMAZIONI, OTTIENIAMO:



Dove la MUTIPICAZIONE tra lo MATRICE DELLE PATCH E LA MATRICE DEI KERNELS RESTITUISCE IL PRODOTTO SCALARE DI UNICO KERNEL PER UNICO PATCH.

## STRIDE

Lo STRIDE rappresenta il passo (oppure l'intervallo) tra le posizioni in cui il filtro viene applicato all'INPUT. Punto, quando si applica un filtro con stride diverso da 1, il filtro si sposta attraverso l'INPUT saltando il numero di pixel specificati. Usare lo stride permette di ridurre la dimensione della feature map e quindi la complessità.

Input	Kernel	Output
$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$	$\begin{matrix} * & & \\ & 0 & 1 \\ & 2 & 3 \end{matrix}$	$\begin{matrix} 0 & 8 \\ 6 & 8 \end{matrix}$

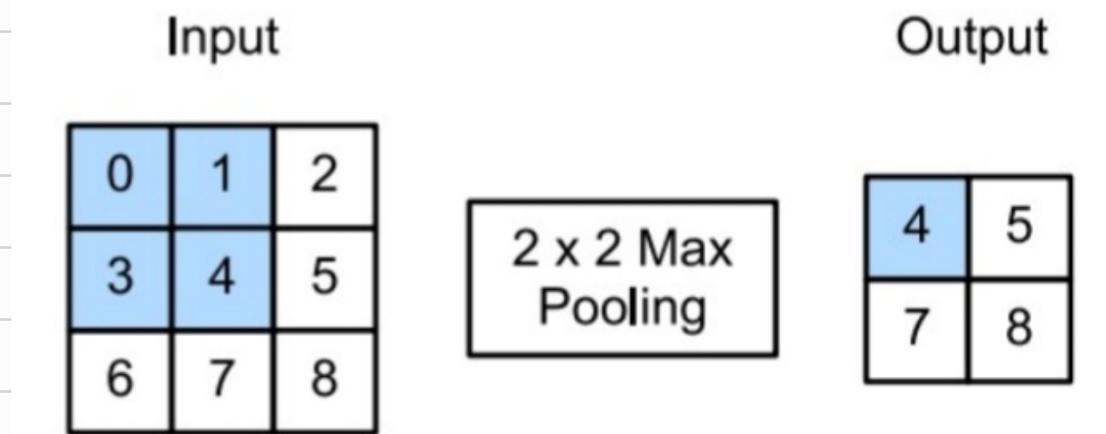
$\hookrightarrow$  stride (2,3)  
 $\Downarrow$

Ci muoviamo su 2 pixel su width  
 e 3 pixel su height

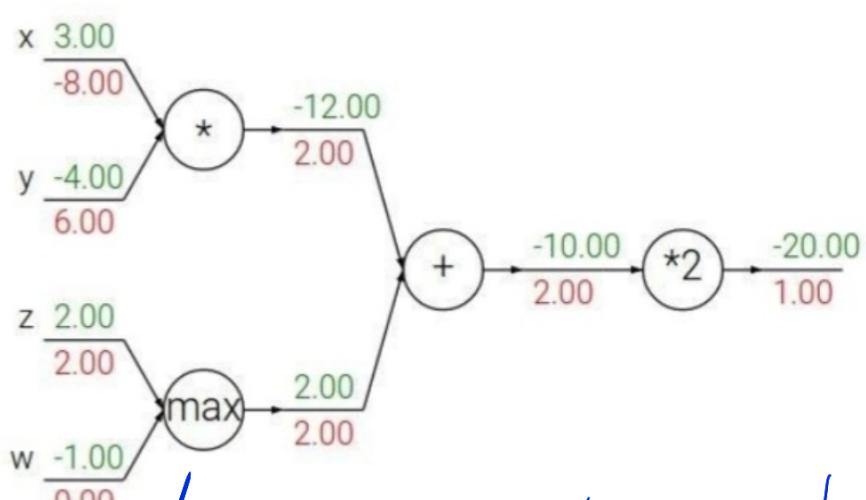
## POOLING

IL POOLING È UNA TECNICA USATA PER RIDURRE LE DIMENSIONI SPAZIALI DELLE FEATURE MAPS E RIDURRE QUINDI LA COMPLESSITÀ COMPUTAZIONALE. IL POOLING PUÒ USARE IL CALCOLO MAX POOLING:

QUESTA DEFINIZIONE LA FINESSA C'È SULLE ATTIVAZIONI DI FEATURE MAP, PER Ogni POSIZIONE DELLA FINESSA VENGONO SELEZIONATI IL VALORE MASSIMO E QUESTO DIVENTA QUINDI IL NUOVO VALORE NELLA FEATURE MAP RIDOTTA. MAX POOLING EVITA QUEGLI CHARACTERISTICS PIÙ RILEVANTI E ALCUNI SONO ROBUSTI E NON VULNERABILI AL NOISE.



INOLTRE, DURANTE IL PROCESSO DI BATCH PROPAGATION, SI HA CHE MAX POOLING VIENE PROPAGANDO IL GRADIENTE SOLONTRO PER GLI INPUT "ATTIVI", OVVERO QUELLI MASSIMI; QUESTO SIGNIFICA CHE IL FUNZIONE  $\max()$  RISCE CON CUIA SORTA DI GRADIENT SWITCHER.



0.00

↳ Proprio sotto l'input massimo!