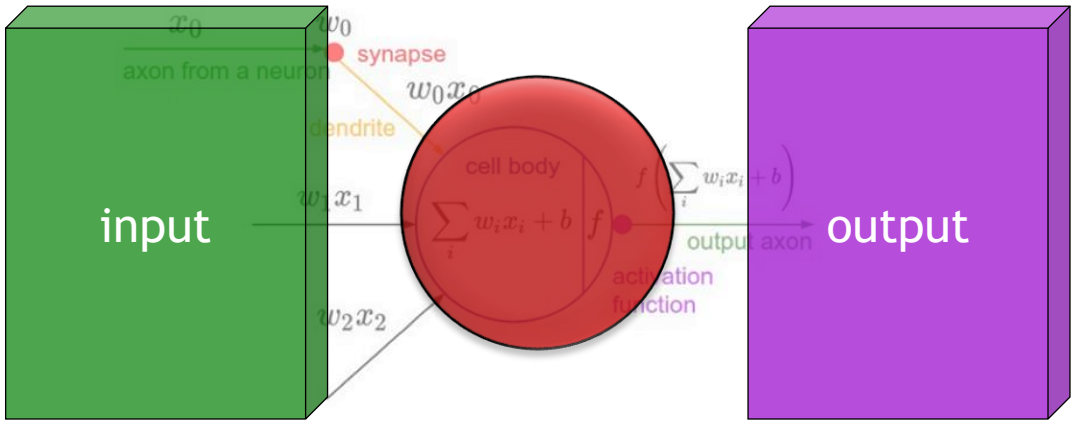
Neuroni e layers

Neuroni e layers

Premessa: neurone

A descrivere il neurone artificiale ci sono semplici caratteristiche:

- Nasce per replicare quello biologico.
- Singolarmente ha una capacità di apprendimento limitata.
- In gruppo, forma una rete interconnessa chiamata **layer**. Riceve uno o più **input**.
- Manipolano l'**input** sulla base del comportamento che vogliono descrivere.
- Producono un **output**.



Neuroni e layers

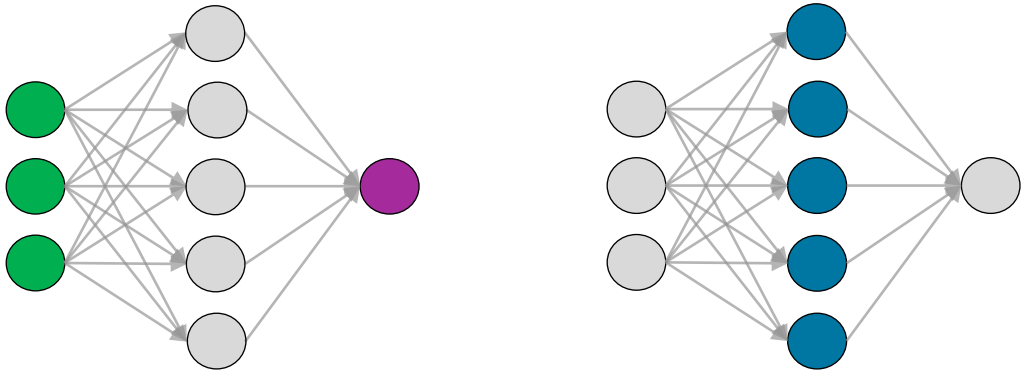
Premessa: layers

Ogni layer è costituito da neuroni dello stesso tipo: ogni neurone all'interno del layer manipolerà gli input ricevuti secondo gli stessi criteri.

Lo strato di input, *input layer*, raggruppa concettualmente l'insieme dei dati di input che entreranno nella rete.

Gli strati intermedi, hidden layers, tra input e output, sono quelli che ci interessano, e che eseguono le principali manipolazioni sui dati.

Lo strato di output, *output layer*, raggruppa concettualmente quello che viene restituito dalla rete.



Neuroni e layers

Tipi di layer

Generalmente, le operazioni eseguite o il modo in cui i dati vengono manipolati dai neuroni, va a dare una rappresentazione ed un nome al layer che li comprende.

I principali layer:

- Convolution.
- Pooling.
- Linear.
- Flatten.
- Dropout.
- Batch-Normalization.

Neuroni e layers

Convolution: un riferimento

Di seguito, un ottimo riferimento per poter comprendere e visualizzare al meglio il comportamento dell'operazione di convoluzione.

In particolare:

YouTube : Animated AI

Github : Animated AI

Neuroni e layers

Convolution: premessa

Come per la convoluzione descritta dalla teoria dell'analisi immagini, il layer convoluzionale applica una similare operazione all'input che gli viene fornito:

- *Definisce un filtro di analisi, il **kernel**.*
- *Si muove lungo l'input.*
- *Produce un nuovo output per ogni applicazione.*

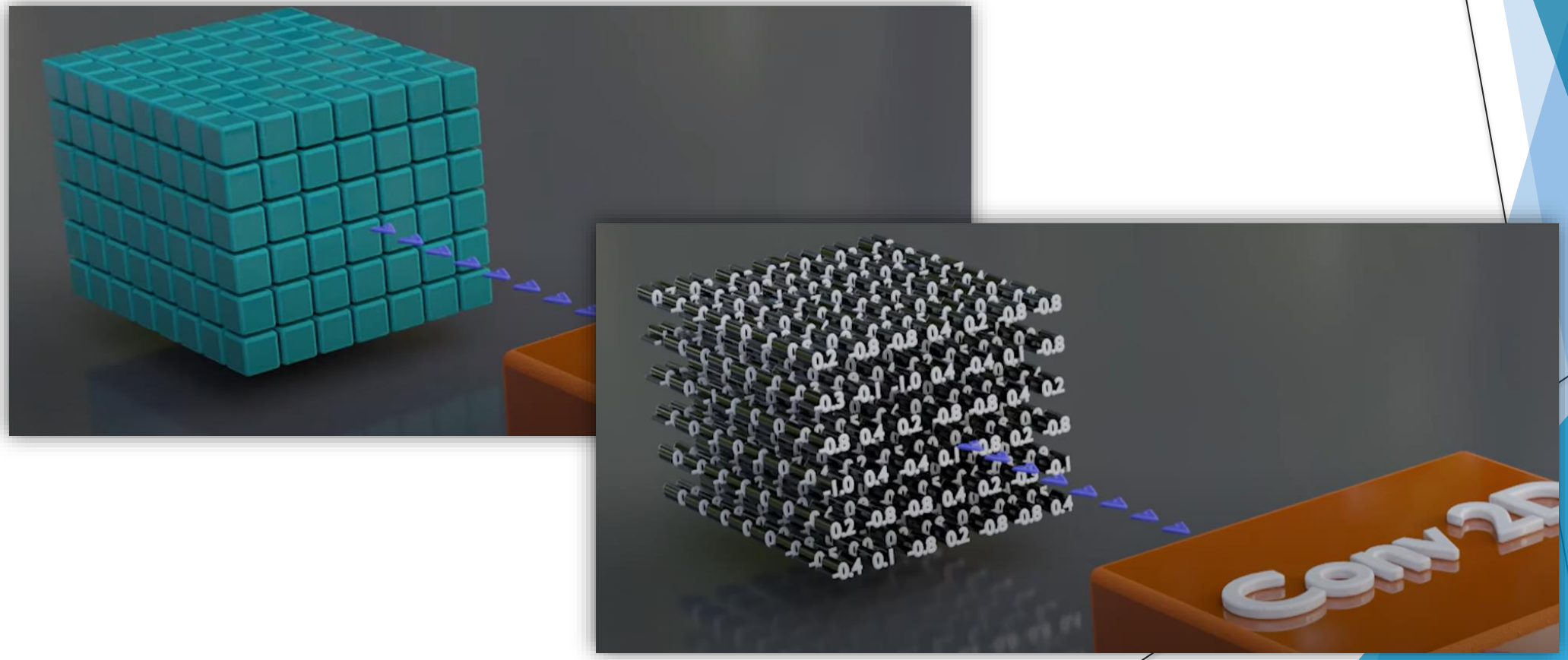
Nello specifico dei layer:

- Si cerca efficienza nella manipolazione di input strutturati «a griglia», come le **immagini**.
- Ogni kernel sarà costituito di **parametri addestrabili** dalla rete, i **pesi**.

Neuroni e layers

Convolution: input

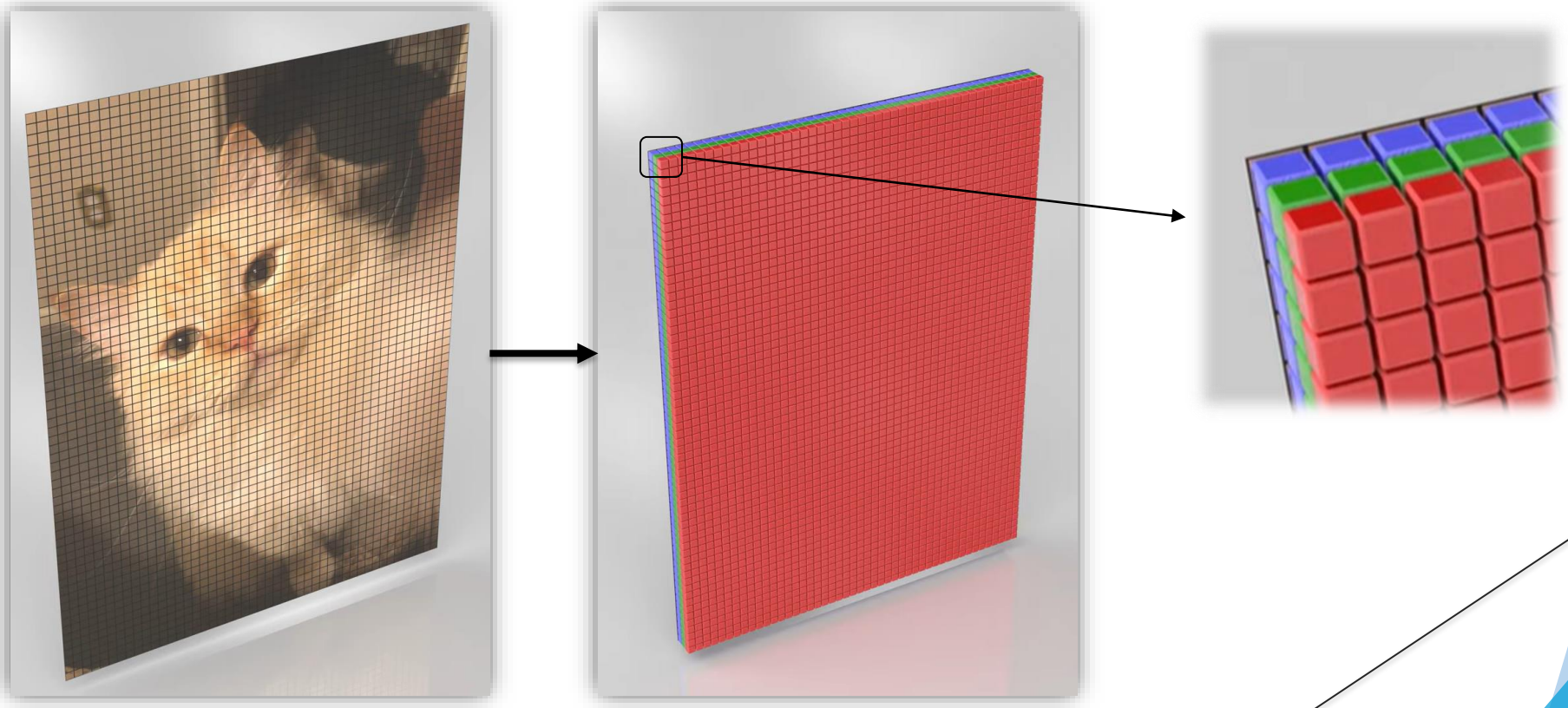
La convoluzione agisce su un input, l'input è un tensore e il tensore è una raccolta di valori.



Neuroni e layers

Convolution: input

L’input può essere qualsiasi tipo di raccolta numerica. Un’immagine a 3 canali è, per questo, un input valido.



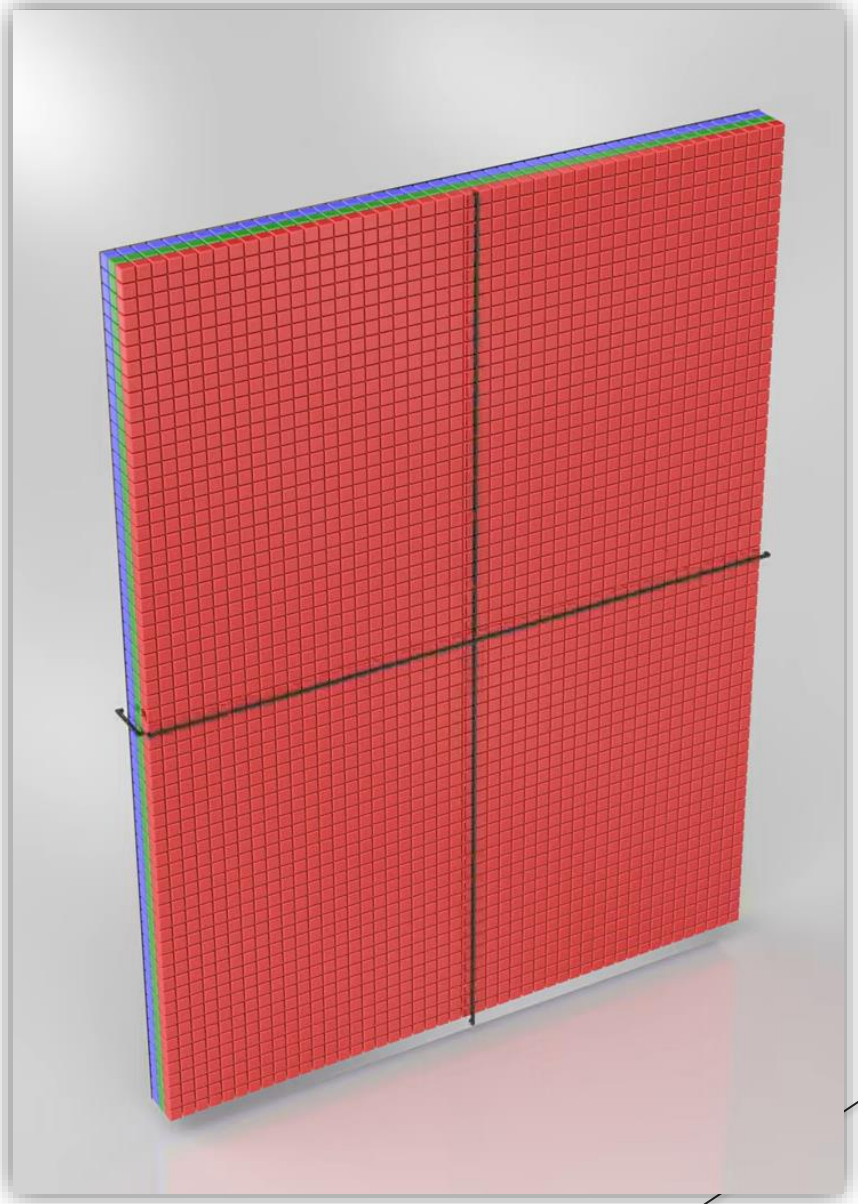
Neuroni e layers

Convolution: input

Dell'input si considerano le tre dimensioni:

- Larghezza
- Altezza
- Profondità: numero di canali o numero di features.

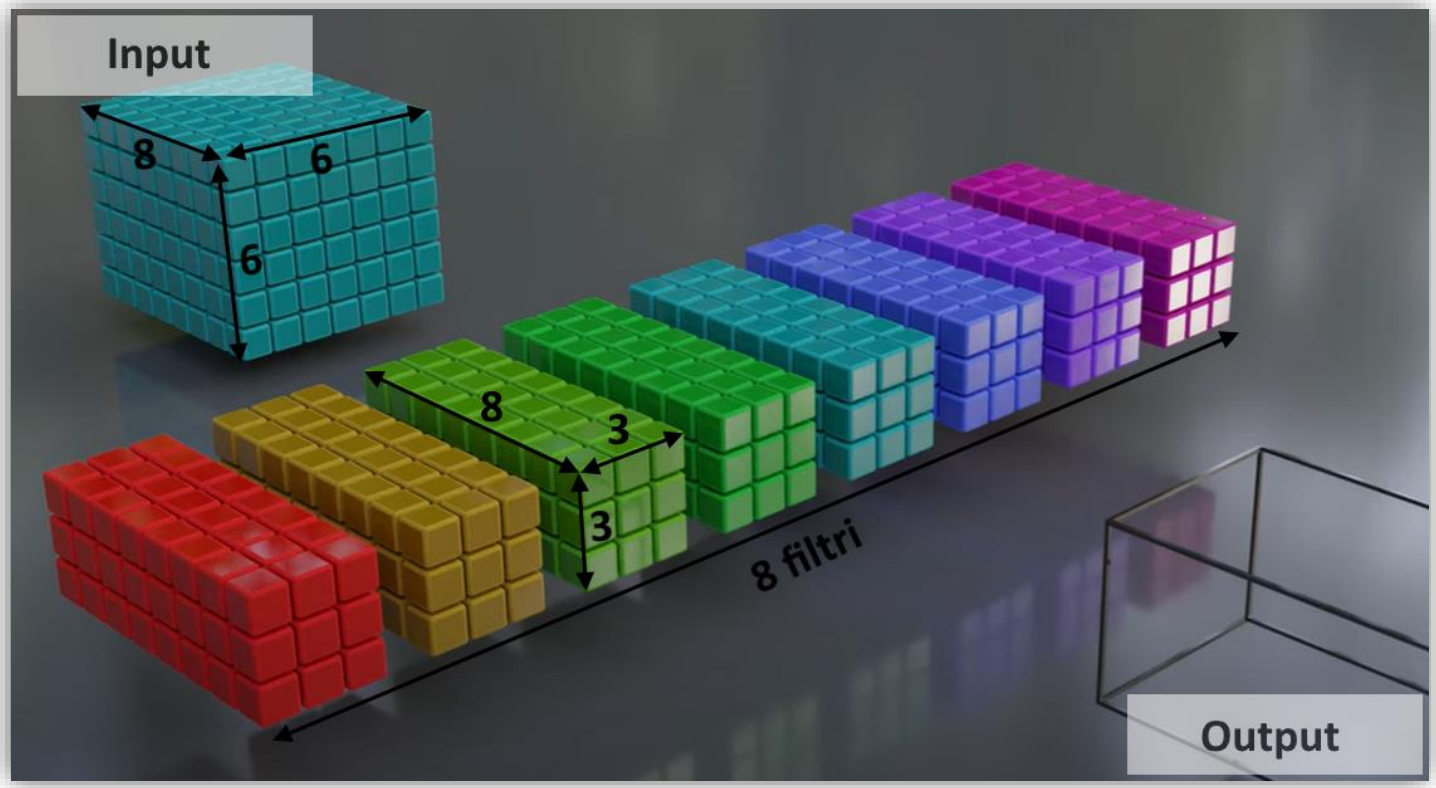
Tensori di questo tipo, sono anche detti ***features map***.



Neuroni e layers

Convolution: kernel

All'input sono applicati *N* filtri, kernel, di convoluzione. Ogni filtro ha una profondità pari a quella dell'input al quale si deve applicare e vi si definisce poi la larghezza e l'altezza.



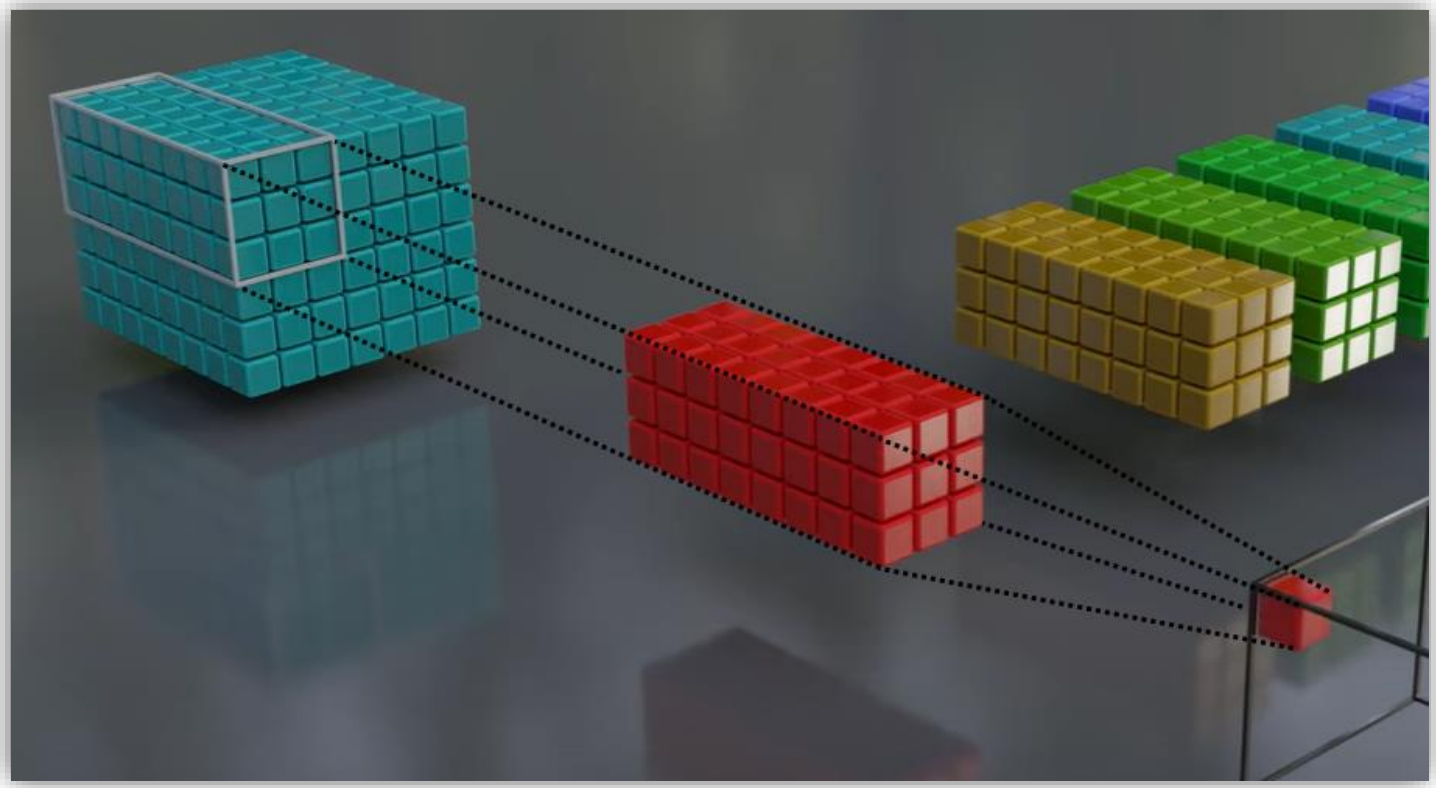
Questi filtri, sono poi trattati, computazionalmente, come un tensore 4 dimensionale:

[8, 3, 3, 8]

Neuroni e layers

Convolution: kernel

Il filtro guarda ad un *patch* dell'input di dimensione pari alla propria. I *pesi* di ogni filtro sono applicati alla patch di input alla quale si aggancia e se ne produce un singolo valore.

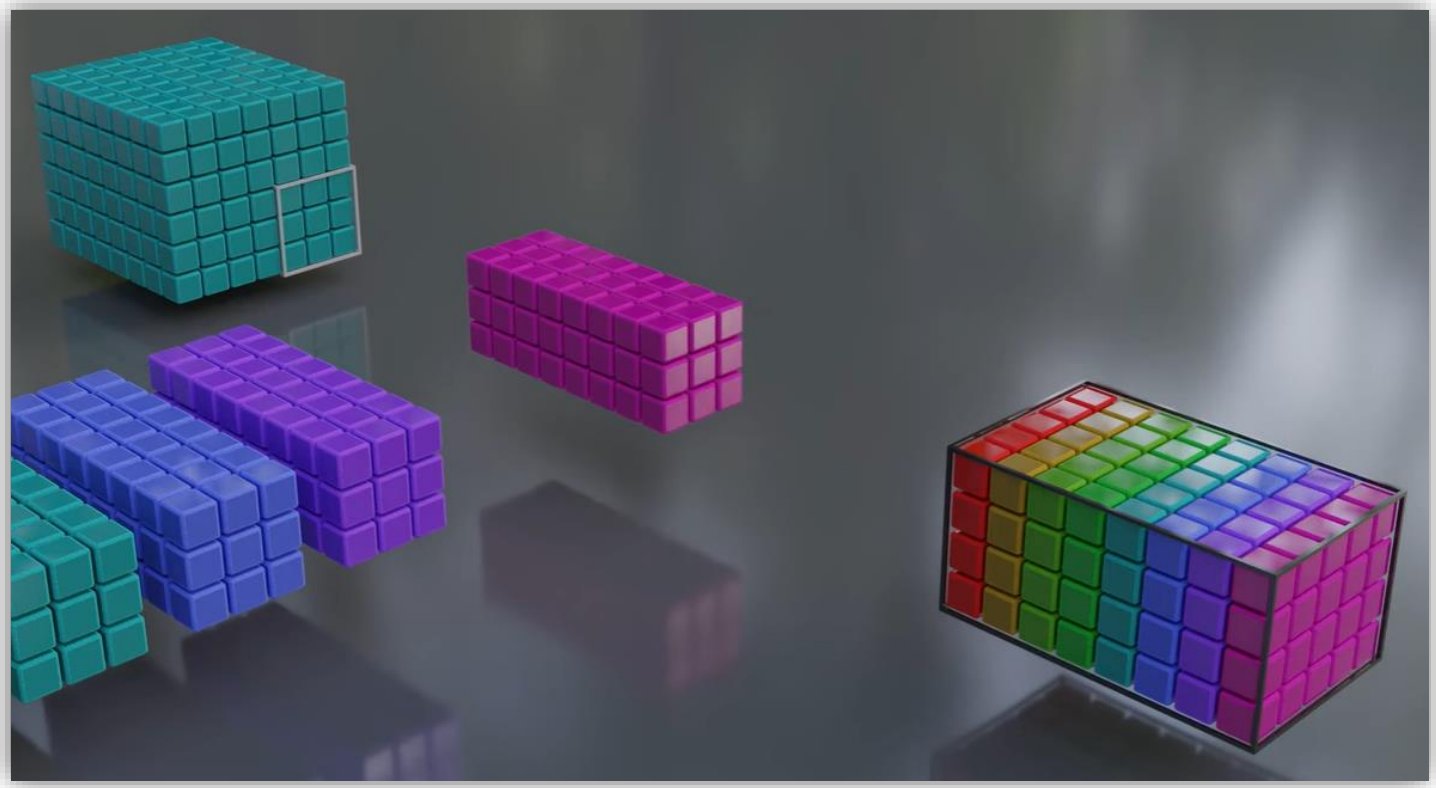


L'operazione poi si ripete muovendosi di patch in patch.

Neuroni e layers

Convolution: kernel

Dalla applicazione di ogni singolo filtro in ogni patch dell'input, si ottiene infine l'output.



Rif:
(*Operazione completa*)

Neuroni e layers

Convolution: proprietà

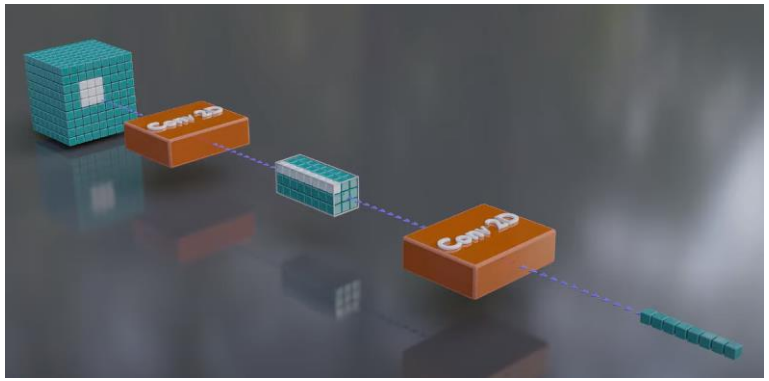
Dei layer convoluzionali, ed in particolare dell'operazione che li definisce, si identificano tre importanti proprietà:

Località spaziale

Dati **vicini** fra loro sono probabilmente correlati.

Applicare il kernel ad una sotto-regione sfrutta questo concetto e cerca di scoprirne le correlazioni più significative.

Concatenare più operazioni di convoluzione, permette quindi di mettere in relazione zone di spazio sempre più grandi.



Rif: (Youtube)

Neuroni e layers

Convolution: proprietà

Dei layer convoluzionali, ed in particolare dell'operazione che li definisce, si identificano tre importanti proprietà:

Condivisione dei pesi

I filtri sono applicati localmente ma rimangono gli stessi quando ci si sposta in un'altra patch dell'input.

Addestrare gli stessi filtri su diverse zone, permette di estrarne caratteristiche di alto livello: colori, forme, orientamenti…

Rende robusto l'apprendimento e riduce il numero di parametri addestrabili.

Rif: (Youtube)

Neuroni e layers

Convolution: proprietà

Dei layer convoluzionali, ed in particolare dell'operazione che li definisce, si identificano tre importanti proprietà:

Gerarchia spaziale

Connettendo un layer convoluzionale all'output di un altro è possibile combinare le informazioni apprese in precedenza.

In questo modo si estraggono tanto più complessi quanto più profondi si va nella rete.

Rif: (Youtube)

Neuroni e layers

Convolution: parametri

L'operazione di convoluzione può modificare il proprio comportamento sulla base di tre parametri principali:

Kernel

Una raccolta di valori, addestrabili durante la fase di addestramento della rete.

Strutturato generalmente come un tensore di valori a virgola mobile.

Stride

È un valore intero che stabilisce il passo di spostamento del kernel, in orizzontale e/o in verticale.

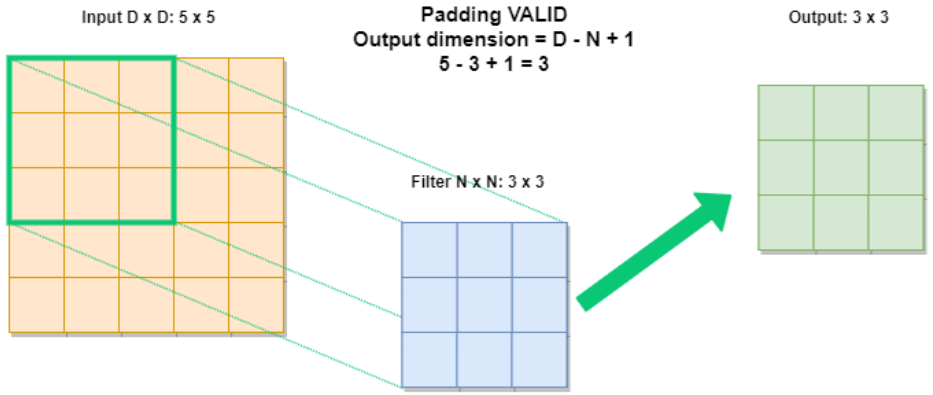
Uno stride di 2 indica che la «prossima» patch di input analizzata si aggancerà due campioni più lontana dalla patch corrente.

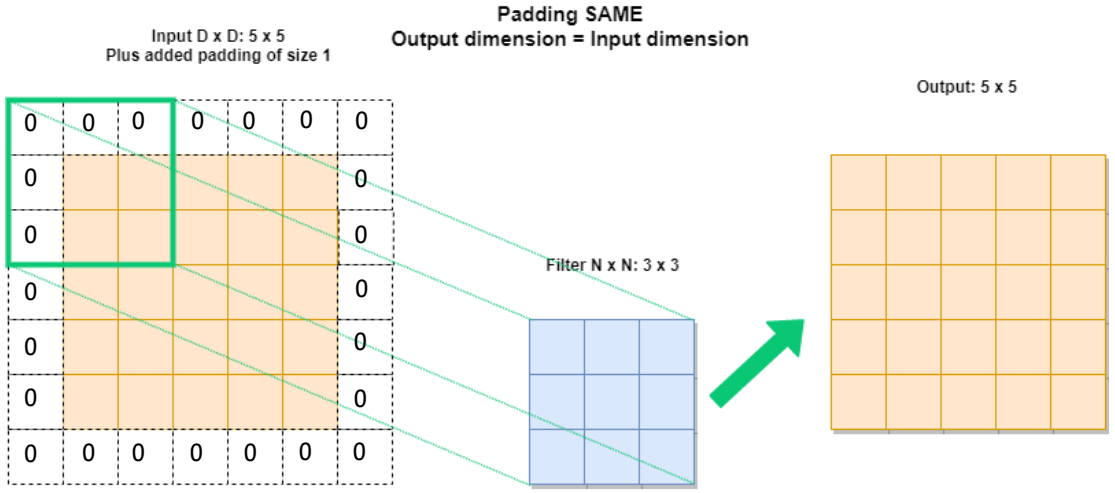
Padding

Indica come la convoluzione dovrà comportarsi nelle situazioni di «bordo» dei dati analizzati.

Neuroni e layers

Convolution: padding





Valid

Esegue la convoluzione solo nei dati di input per i quali il kernel non «sfora».

La dimensione dell'output sarà inferiore a quella dell'output.

Same

Aggiunge dati di bordo all'input per far sì che l'output della convoluzione abbia le stesse dimensioni dell'input.

Neuroni e layers

Convolution: calcolo delle dimensioni

La convoluzione permette di ridurre le dimensionalmente l'input.

La riduzione dimensionale avverrà, in genere, in tutte le dimensioni diverse da quella che rappresenta la profondità.

Input : (W_{in}, H_{in}, C_{in})
Output : $(W_{out}, H_{out}, C_{out})$
Kernel : (W_k, H_k)

$$W_{out} = \frac{W_{in} - W_k + 2 * padding}{stride} + 1$$

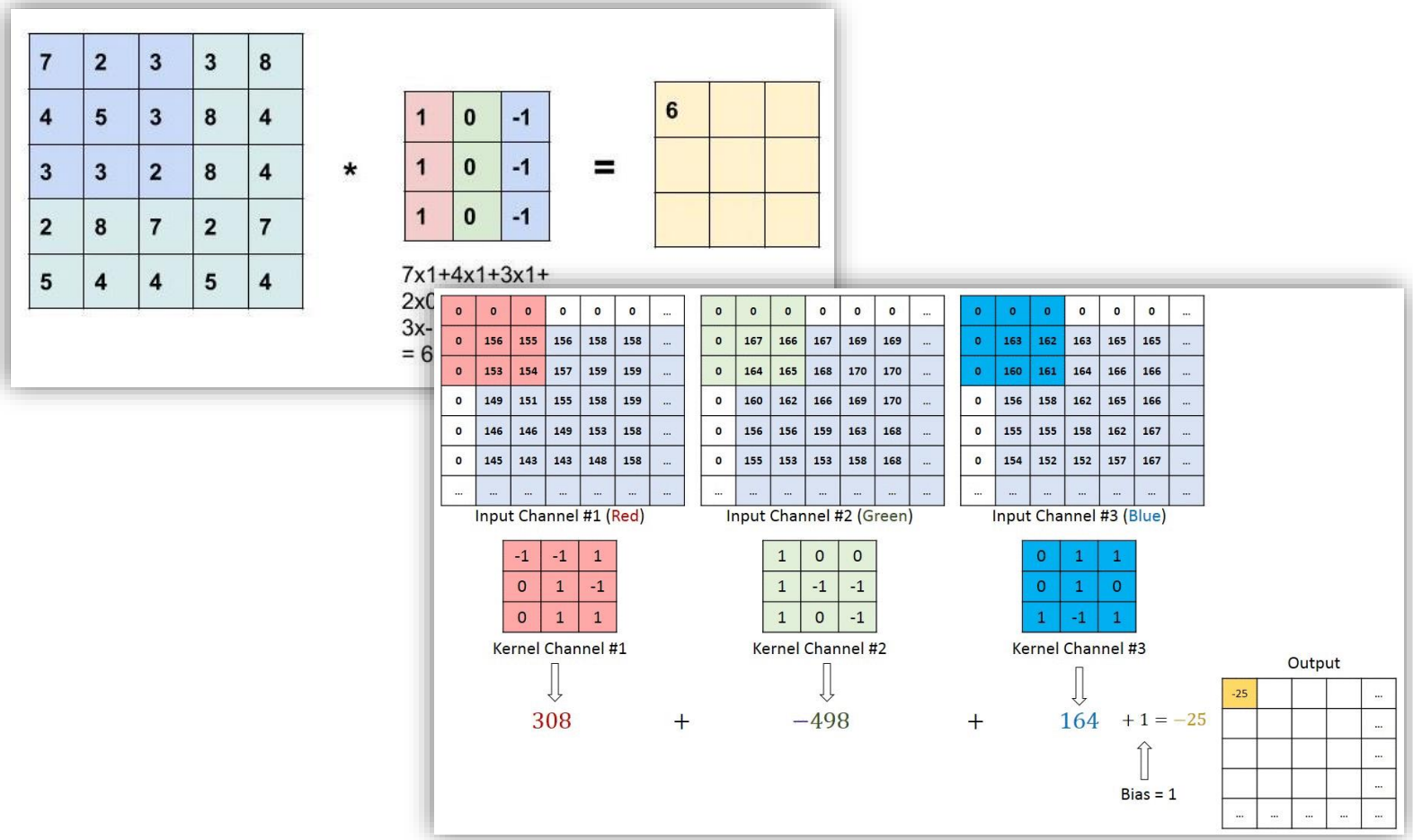
$$H_{out} = \frac{H_{in} - H_k + 2 * padding}{stride} + 1$$

Di seguito, alcuni riferimenti utilizzabili come aiuto per il calcolo delle dimensioni di output della convoluzione:

- Convnet Calculator
- Convolution Shape Calculator

Neuroni e layers

Convolution: l'operazione



Rif: (CNNs)

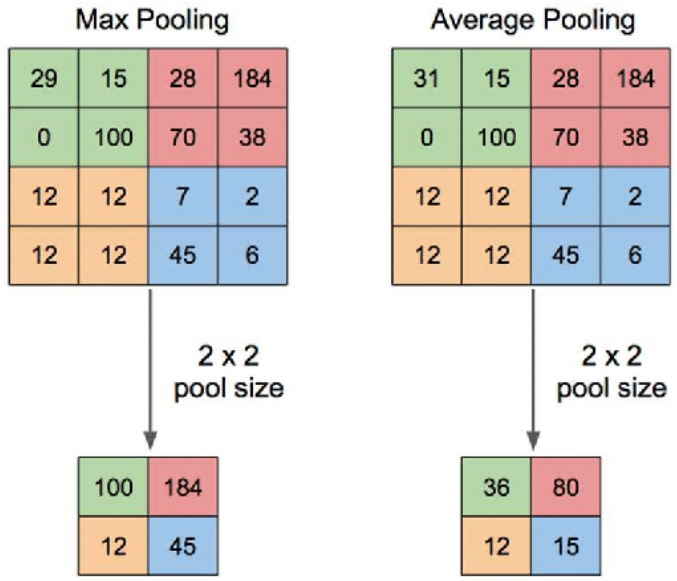
Neuroni e layers

Pooling

Permette di ridurre la dimensione del tensore di input.

Opera come la convoluzione:

- *Definisce un kernel.*
- *Definisce stride e padding.*
- *Filtra l'input sulla base di una condizione: il **pooling**.*



I principali tipi di pooling sono:

- ***Max pooling*** : restituisce il valore massimo fra i valori.
- ***Average pooling*** : restituisce la media dei valori.

Neuroni e layers

Linear

Crea una rete di neuroni completamente connessi fra loro che applicano una trasformazione lineare ai dati in input:

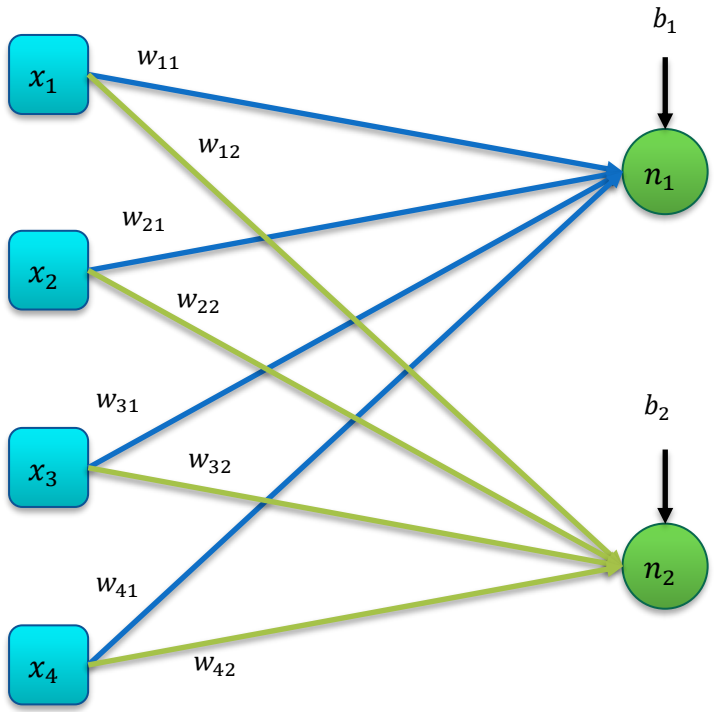
$$y = xW^T + b$$

$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$ $W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix}$ $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

Dati di input

Matrice dei pesi W, inizializzata randomicamente

Matrice dei bias, inizializzata randomicamente



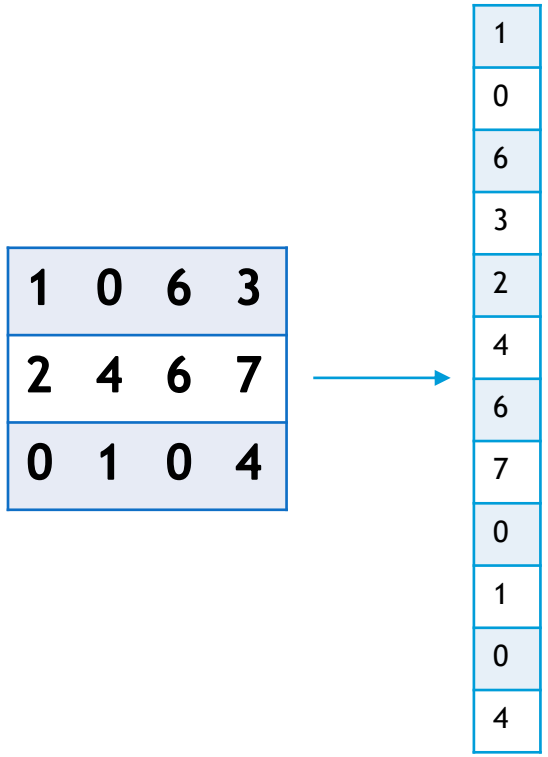
Rif:(Analytics Vidhya)

Neuroni e layers

Flatten

Ridimensiona i dati di input riducendoli ad tensore 1-dimenisonale, un vettore di valori.

Utilizzata nel momento in cui si passa dall'estrazione delle features eseguita dagli strati convoluzionali, all'effettiva classificazione o regressione, eseguita di solito da strati lineari.



Neuroni e layers

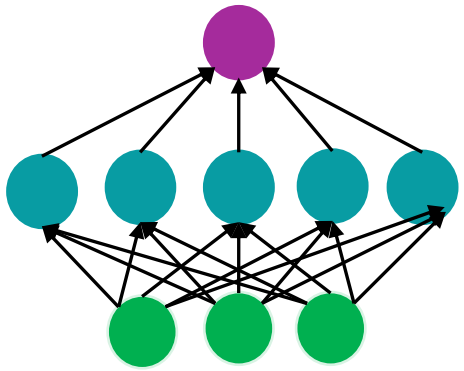
Dropout

Gli strati, in generale, sono completamente connessi: ***fully-connected***:

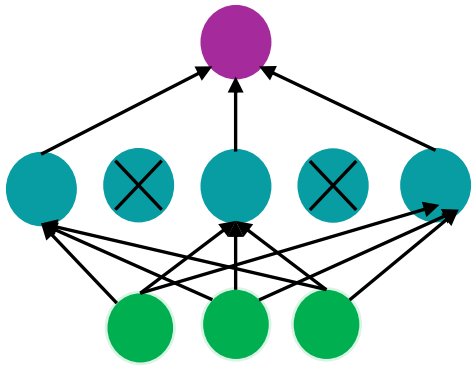
Ogni neurone di uno strato è collegato a quelli dello strato successivo.

Il dropout, nella sua forma più semplice, permette di tagliare alcune di queste connessioni per forzare la rete ad apprendere pattern e creare associazioni altrimenti «sottovalutate» evitandone altre, «predilette».

La sua principale funzione è prevenire l'overfitting.



Senza dropout



Con dropout

Neuroni e layers

Batch-Normalization

Nella creazione di una rete e nel suo apprendimento si incontrano diverse tipologie di problemi:

- *Addestramento lento.*
- *Addestramento instabile.*
- *Overfitting.*

Batch-normalization mira a mitigare e risolve questi ed altri problemi.

Recap:

Normalizzazione: dati in [0,1]. *Standardizzazione*: dati con μ:0 σ:1.

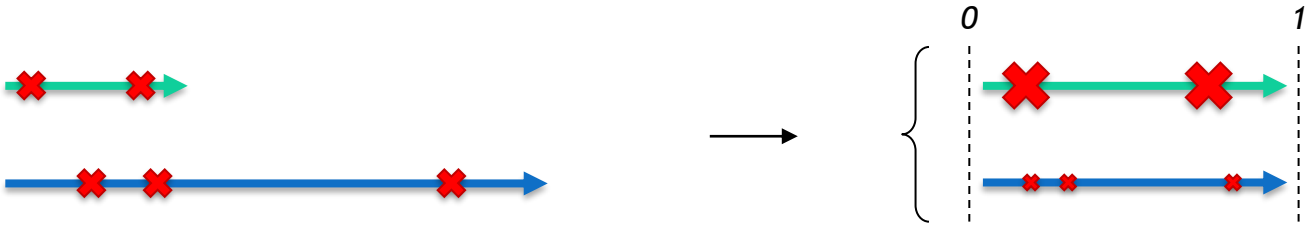
Ref(ArXiv)

Neuroni e layers

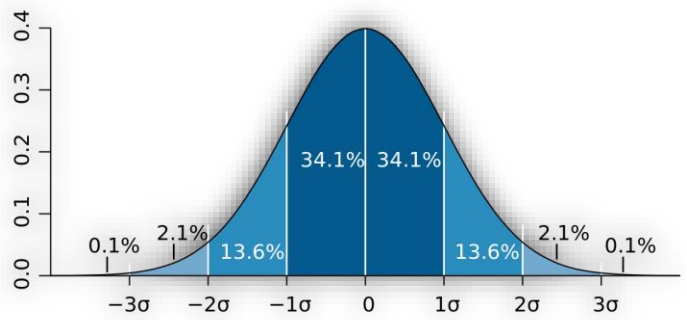
Batch-Normalization: recap

Con la normalizzazione, input diversi (in termini di *range di valori*) vengono allineati fra loro:

- *Le proporzionalità fra i dati sono mantenute.*
- *Si evita che input con valori maggiori sbilancino l'apprendimento a discapito di input in un range inferiore.*



Con la standardizzazione si fa in modo che i dati centrino la propria distribuzione in 0, mitigando effetti di offset, e se ne abbia un controllo sulla dispersione e i valori.



Neuroni e layers

Batch-Normalization: l'operazione

Quello che la Batch-Normalization fa ai dati ricevuti in ingresso è:

- Standardizzare i valori: media zero e varianza unitaria.
- Trasformarli: moltiplicandoli per un fattore e spostandoli di un offset.

Fattore ed offset sono a tutti gli effetti parametri appresi durante la fase di addestramento e automaticamente gestiti dal layer.

Il loro valore sarà strettamente correlato ai dati ricevuti dal layer.

L'applicazione della BN è in genere fatta sull'input e su tutti gli output dei vari layer al fine di mantenerne le proprietà per l'intero attraversamento dei dati.

***Nota**: si parla di normalizzazione anche se in questo caso il primo passo è una standardizzazione.*

Neuroni e layers

Batch-Normalization

L'operazione di BN, in termini pratici, viene applicata subito dopo la funzione di attivazione, non-lineare, dei layer.

*Durante la fase di **training** calcola media e deviazione standard sulla batch di campioni in ingresso.*

*In fase di **validazione**, media e deviazione standard vengono calcolate utilizzando quelle raccolte per ogni batch durante la fase di addestramento.*

Anche se, a primo impatto, potrebbe sembrare che l'applicazione di queste operazioni nei vari layer influenzi negativamente le durate di ogni epoca, è stato sperimentalmente verificato come nella maggior parte dei casi:

- Si velocizzi la convergenza dell'apprendimento.
- Aumenti la stabilità dell'apprendimento.
- Riduce la necessità di applicare altri metodi di regolarizzazione.

Neuroni e layers

Layers in PyTorch

PyTorch definisce i layer all'interno del modulo *torch.nn*.

Ne vengono messi a disposizione molteplici, fra cui quelli visti finora:

- *Convolution* : ConvolutionLayers
- *Pooling* : PoolingLayers
- *Linear* : LinearLayers
- *Flatten* : Utilities
- *Dropout* : DropoutLayers
- *Batch-Normalization* : NormalizationLayers

Neuroni e layers

Layers in PyTorch

Nello specifico:

- *Lo strato più comunemente usato per la convoluzione* : *Conv2D*
- *Le operazioni di pooling più frequenti* : *MaxPool2d*, *AvgPool2d*
- *Per la generazione di strati fully-connected* : *Linear*
- *Per eseguire un'operazione di appiattimento* : *Flatten*
- *Per implementare tecniche di dropout* : *Dropout*
- *Per applicare una batch-normalization* : *BatchNorm2d*

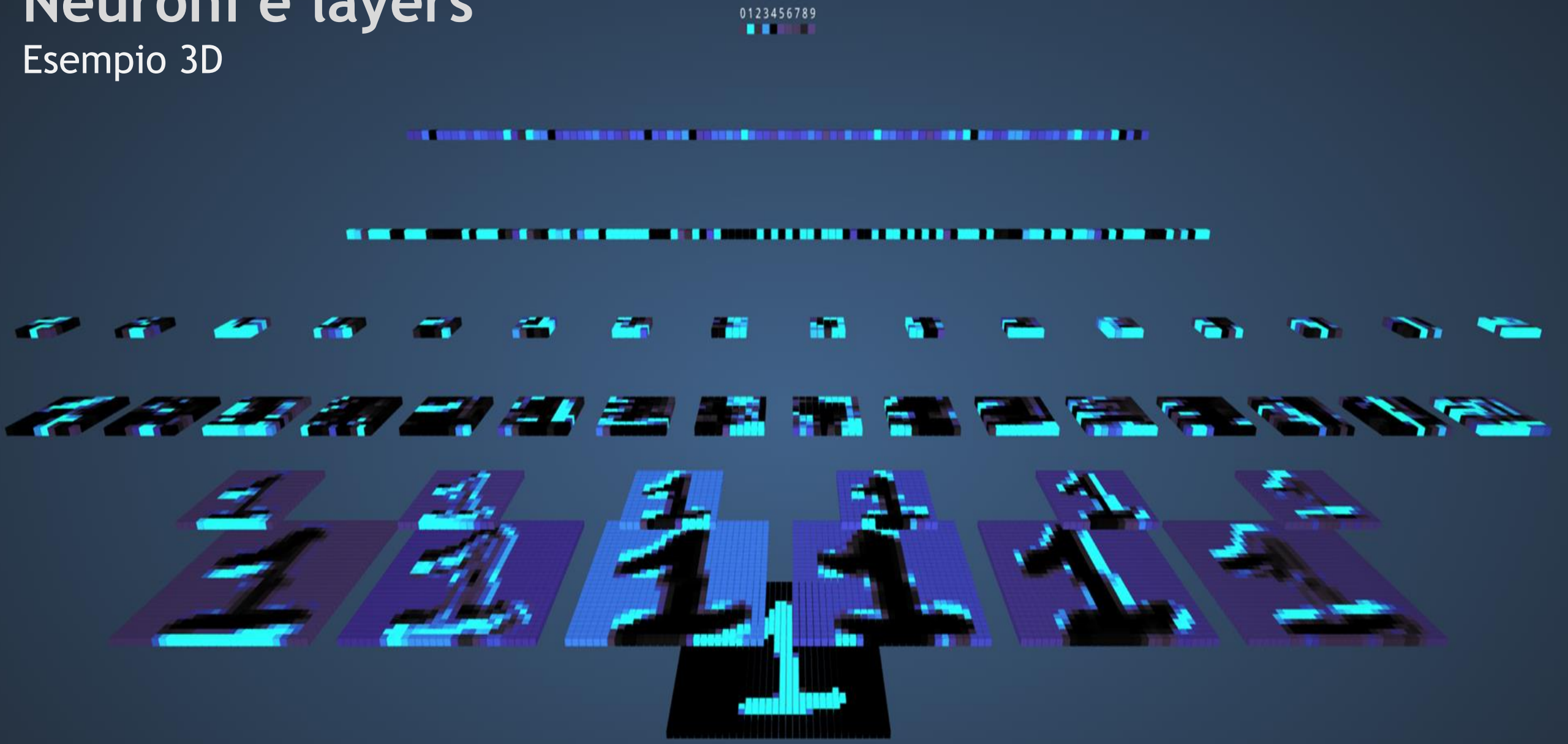
Neuroni e layers

Esempio 2D

Rif: (2D CNN)

Neuroni e layers

Esempio 3D



Rif: (3D CNN)

Proviamo?