

# **Implementazione di una Rete Convoluzionale in CUDA**

Michele Valsesia

Nicholas Aspes

Anno accademico 2018/2019

# Introduzione

## Obiettivi

- Descrivere brevemente l'architettura ed il funzionamento di una *Rete Neurale*

# Introduzione

## Obiettivi

- ▶ Descrivere brevemente l'architettura ed il funzionamento di una *Rete Neurale*
- ▶ Motivare le differenti scelte implementative adottate durante lo svolgimento del progetto

# Introduzione

## Obiettivi

- ▶ Descrivere brevemente l'architettura ed il funzionamento di una *Rete Neurale*
- ▶ Motivare le differenti scelte implementative adottate durante lo svolgimento del progetto
- ▶ Valutare l'accuratezza e lo speed-up della rete rispetto ad una sua implementazione sequenziale

# Reti Neurali

# Reti Neurali

## Scopo

- Le *Reti Neurali* vengono principalmente usate per la classificazione di immagini

# Reti Neurali

## Scopo

- ▶ Le *Reti Neurali* vengono principalmente usate per la classificazione di immagini
- ▶ Il processo di classificazione consiste nell'assegnare ad un immagine un'etichetta che identifichi nel miglior modo possibile il suo contenuto semantico

# Reti Neurali

## Scopo

- ▶ Le *Reti Neurali* vengono principalmente usate per la classificazione di immagini
- ▶ Il processo di classificazione consiste nell'assegnare ad un immagine un'etichetta che identifichi nel miglior modo possibile il suo contenuto semantico
- ▶ L'insieme delle immagini che hanno tutte la stessa etichetta costituiscono una *classe*



# Reti Neurali

## Scopo

- ▶ Le *Reti Neurali* vengono principalmente usate per la classificazione di immagini
- ▶ Il processo di classificazione consiste nell'assegnare ad un immagine un'etichetta che identifichi nel miglior modo possibile il suo contenuto semantico
- ▶ L'insieme delle immagini che hanno tutte la stessa etichetta costituiscono una *classe*
- ▶ Le reti neurali ricevono in input un'immagine e forniscono in output la relativa classe

# Reti Neurali

## Funzionamento

- Una rete neurale deve *apprendere* come assegnare correttamente le immagini alle varie classi

# Reti Neurali

## Funzionamento

- ▶ Una rete neurale deve *apprendere* come assegnare correttamente le immagini alle varie classi
- ▶ Un *esempio* è una coppia (immagine, etichetta)

# Reti Neurali

## Funzionamento

- ▶ Una rete neurale deve *apprendere* come assegnare correttamente le immagini alle varie classi
- ▶ Un *esempio* è una coppia (immagine, etichetta)
- ▶ Un team di persone valuta il contenuto semantico di ciascuna immagine e assegna all'esempio l'etichetta corrispondente

# Reti Neurali

## Funzionamento

- ▶ Una rete neurale deve *apprendere* come assegnare correttamente le immagini alle varie classi
- ▶ Un *esempio* è una coppia (immagine, etichetta)
- ▶ Un team di persone valuta il contenuto semantico di ciascuna immagine e assegna all'esempio l'etichetta corrispondente
- ▶ Il *training set* ed il *test set* sono insiemi di esempi

# Reti Neurali

## Funzionamento

- ▶ Una rete neurale deve *apprendere* come assegnare correttamente le immagini alle varie classi
- ▶ Un *esempio* è una coppia (immagine, etichetta)
- ▶ Un team di persone valuta il contenuto semantico di ciascuna immagine e assegna all'esempio l'etichetta corrispondente
- ▶ Il *training set* ed il *test set* sono insiemi di esempi
- ▶ Il training set viene usato per l'addestramento (training) della rete

# Reti Neurali

## Funzionamento

- ▶ Una rete neurale deve *apprendere* come assegnare correttamente le immagini alle varie classi
- ▶ Un *esempio* è una coppia (immagine, etichetta)
- ▶ Un team di persone valuta il contenuto semantico di ciascuna immagine e assegna all'esempio l'etichetta corrispondente
- ▶ Il *training set* ed il *test set* sono insiemi di esempi
- ▶ Il training set viene usato per l'addestramento (training) della rete
- ▶ Il test set serve a controllare che la rete abbia imparato a discriminare correttamente le immagini

# Reti Neurali

## Training

- ▶ Per ognuno degli esempi del training set



# Reti Neurali

## Training

- ▶ Per ognuno degli esempi del training set
  - La rete riceve in input l'immagine relativa all'esempio considerato e l'associa ad una delle classi presenti

# Reti Neurali

## Training

- ▶ Per ognuno degli esempi del training set
  - La rete riceve in input l'immagine relativa all'esempio considerato e l'associa ad una delle classi presenti
  - Se la classe in output è diversa dall'etichetta dell'esempio, la rete corregge i suoi parametri interni e passa all'immagine successiva

# Reti Neurali

## Testing

- L'*accuratezza* della rete è data dal rapporto tra il numero di esempi classificati scorrettamente ed il numero totale di esempi

# Reti Neurali

## Testing

- ▶ L'*accuratezza* della rete è data dal rapporto tra il numero di esempi classificati scorrettamente ed il numero totale di esempi
- ▶ Per ognuno degli esempi del test set

# Reti Neurali

## Testing

- ▶ L'*accuratezza* della rete è data dal rapporto tra il numero di esempi classificati scorrettamente ed il numero totale di esempi
- ▶ Per ognuno degli esempi del test set
  - La rete riceve in input l'immagine dell'esempio considerato e l'associa ad una delle classi presenti

# Reti Neurali

## Testing

- ▶ L'*accuratezza* della rete è data dal rapporto tra il numero di esempi classificati scorrettamente ed il numero totale di esempi
- ▶ Per ognuno degli esempi del test set
  - La rete riceve in input l'immagine dell'esempio considerato e l'associa ad una delle classi presenti
  - Ogni volta che l'output della rete non corrisponde all'etichetta dell'esempio viene incrementato un contatore, necessario per il calcolo dell'accuratezza

# Reti Neurali

## Significato Biologico

- Le *Reti Neurali* nascono con lo scopo di modellare una rete neurale biologica

# Reti Neurali

## Significato Biologico

- ▶ Le *Reti Neurali* nascono con lo scopo di modellare una rete neurale biologica
- ▶ Una rete neurale biologica si compone di unità cellulari di base: i *neuroni*



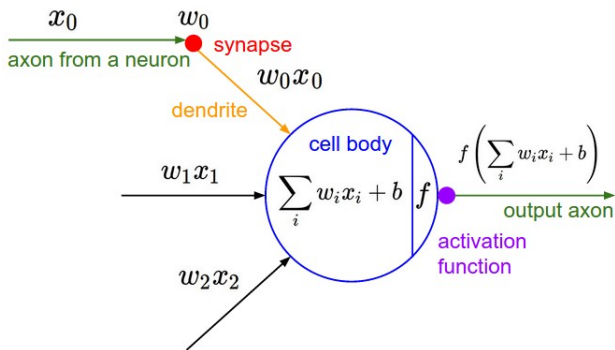
# Reti Neurali

## Significato Biologico

- ▶ Le *Reti Neurali* nascono con lo scopo di modellare una rete neurale biologica
- ▶ Una rete neurale biologica si compone di unità cellulari di base: i *neuroni*
- ▶ I neuroni sono collegati tra loro per mezzo di specifiche giunture chiamate *sinapsi*

# Reti Neurali

## Neurone



*Modello matematico di un neurone*

# Reti Neurali

## Funzionamento Neurone

- Attraverso un meccanismo di eccitazione ed inibizione i pesi sinaptici controllano quanto un neurone sia influenzato dagli altri

# Reti Neurali

## Funzionamento Neurone

- ▶ Attraverso un meccanismo di eccitazione ed inibizione i pesi sinaptici controllano quanto un neurone sia influenzato dagli altri
- ▶ I segnali in ingresso al neurone vengono pesati dalle differenti sinapsi, trasportati dai dendriti all'interno del corpo cellulare e sommati tra loro

# Reti Neurali

## Funzionamento Neurone

- ▶ Attraverso un meccanismo di eccitazione ed inibizione i pesi sinaptici controllano quanto un neurone sia influenzato dagli altri
- ▶ I segnali in ingresso al neurone vengono pesati dalle differenti sinapsi, trasportati dai dendriti all'interno del corpo cellulare e sommati tra loro
- ▶ Quando la somma supera una certa soglia, il neurone *spara* un segnale lungo l'assone

# Reti Neurali

## Funzionamento Neurone

- ▶ Attraverso un meccanismo di eccitazione ed inibizione i pesi sinaptici controllano quanto un neurone sia influenzato dagli altri
- ▶ I segnali in ingresso al neurone vengono pesati dalle differenti sinapsi, trasportati dai dendriti all'interno del corpo cellulare e sommati tra loro
- ▶ Quando la somma supera una certa soglia, il neurone *spara* un segnale lungo l'assone
- ▶ La *frequenza di sparo* del neurone viene modellata con una funzione di attivazione  $f$

# Reti Neurali

## Funzioni di Attivazione

### Definizione

Una *funzione di attivazione* è una funzione matematica non lineare che viene usata per calcolare l'output di un neurone. Il suo input è dato dalla somma pesata dei segnali in ingresso al neurone

# Reti Neurali

## Funzioni di Attivazione

### Definizione

Una *funzione di attivazione* è una funzione matematica non lineare che viene usata per calcolare l'output di un neurone. Il suo input è dato dalla somma pesata dei segnali in ingresso al neurone

- *Rectifier Linear Unit*



# Reti Neurali

## Funzioni di Attivazione

### Definizione

Una *funzione di attivazione* è una funzione matematica non lineare che viene usata per calcolare l'output di un neurone. Il suo input è dato dalla somma pesata dei segnali in ingresso al neurone

- ▶ *Rectifier Linear Unit*
- ▶ *Sigmoide*

# Reti Neurali

## Funzioni di Attivazione

### Definizione

Una *funzione di attivazione* è una funzione matematica non lineare che viene usata per calcolare l'output di un neurone. Il suo input è dato dalla somma pesata dei segnali in ingresso al neurone

- ▶ *Rectifier Linear Unit*
- ▶ *Sigmoide*
- ▶ *Tangente Iperbolica*

# Reti Neurali

## Funzioni di Attivazione

### Definizione

Una *funzione di attivazione* è una funzione matematica non lineare che viene usata per calcolare l'output di un neurone. Il suo input è dato dalla somma pesata dei segnali in ingresso al neurone

- ▶ *Rectifier Linear Unit*
- ▶ *Sigmoide*
- ▶ *Tangente Iperbolica*
- ▶ *Softplus*

# Reti Neurali

## Rectifier Linear Unit

### Definizione

La *Rectifier Linear Unit (ReLU)*  $r : \mathbb{R} \rightarrow [0, +\infty)$  è definita come  $r(x) = \max(0, x)$

# Reti Neurali

## Rectifier Linear Unit

### Definizione

La *Rectifier Linear Unit (ReLU)*  $r : \mathbb{R} \rightarrow [0, +\infty)$  è definita come  $r(x) = \max(0, x)$

- Si differenzia da una funzione di tipo lineare per metà del suo dominio in quanto  $\forall x < 0, \max(0, x) = 0$

# Reti Neurali

## Rectifier Linear Unit

### Definizione

La *Rectifier Linear Unit (ReLU)*  $r : \mathbb{R} \rightarrow [0, +\infty)$  è definita come  $r(x) = \max(0, x)$

- ▶ Si differenzia da una funzione di tipo lineare per metà del suo dominio in quanto  $\forall x < 0, \max(0, x) = 0$
- ▶ Presenta un punto di discontinuità in  $x = 0$

# Reti Neurali

## Rectifier Linear Unit

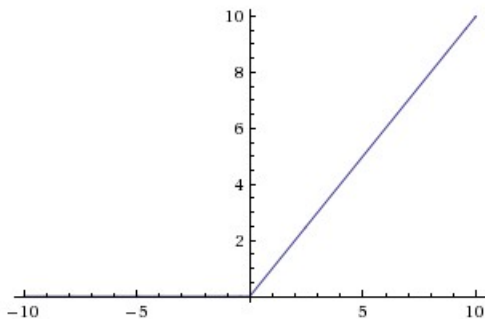
### Definizione

La *Rectifier Linear Unit (ReLU)*  $r : \mathbb{R} \rightarrow [0, +\infty)$  è definita come  $r(x) = \max(0, x)$

- ▶ Si differenzia da una funzione di tipo lineare per metà del suo dominio in quanto  $\forall x < 0, \max(0, x) = 0$
- ▶ Presenta un punto di discontinuità in  $x = 0$
- ▶ La sua derivata è pari a  $\mathbb{1}(x \geq 0)$

# Reti Neurali

## Rectifier Linear Unit



*Rappresentazione grafica ReLU*



# Reti Neurali

## Sigmoide

### Definizione

La *Sigmoide*  $\sigma : \mathbb{R} \rightarrow [0, 1]$  è definita come  $\sigma(x) = \frac{1}{(1+e^{-x})}$

# Reti Neurali

## Sigmoide

### Definizione

La *Sigmoide*  $\sigma : \mathbb{R} \rightarrow [0, 1]$  è definita come  $\sigma(x) = \frac{1}{(1+e^{-x})}$

- Per elevati valori negativi di input la sigmoide restituisce 0: il neurone non spara affatto

# Reti Neurali

## Sigmoide

### Definizione

La *Sigmoide*  $\sigma : \mathbb{R} \rightarrow [0, 1]$  è definita come  $\sigma(x) = \frac{1}{(1+e^{-x})}$

- Per elevati valori negativi di input la sigmoide restituisce 0: il neurone non spara affatto
- Per elevati valori positivi la sigmoide restituisce 1: il neurone satura e spara con frequenza di sparo pari a 1

# Reti Neurali

## Sigmoide

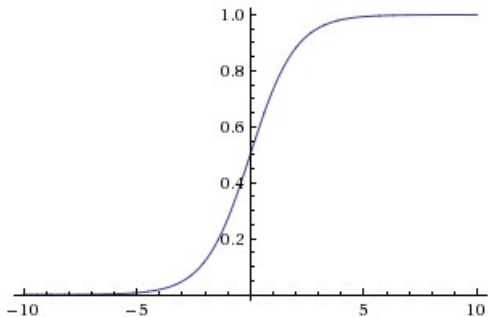
### Definizione

La *Sigmoide*  $\sigma : \mathbb{R} \rightarrow [0, 1]$  è definita come  $\sigma(x) = \frac{1}{(1+e^{-x})}$

- ▶ Per elevati valori negativi di input la sigmoide restituisce 0: il neurone non spara affatto
- ▶ Per elevati valori positivi la sigmoide restituisce 1: il neurone satura e spara con frequenza di sparo pari a 1
- ▶ La sua derivata è uguale a  $\sigma'(x) = \sigma(x)(1 - \sigma(x))$

# Reti Neurali

## Sigmoide



*Rappresentazione grafica Sigmoide*

# Reti Neurali

## Tangente Iperbolica

### Definizione

La *Tangente Iperbolica*  $\tanh : \mathbb{R} \rightarrow [-1, 1]$  è definita come  $\tanh(x) = 2\sigma(2x) - 1$

# Reti Neurali

## Tangente Iperbolica

### Definizione

La *Tangente Iperbolica*  $\tanh : \mathbb{R} \rightarrow [-1, 1]$  è definita come  $\tanh(x) = 2\sigma(2x) - 1$

- La tangente iperbolica è una sigmoide scalata

# Reti Neurali

## Tangente Iperbolica

### Definizione

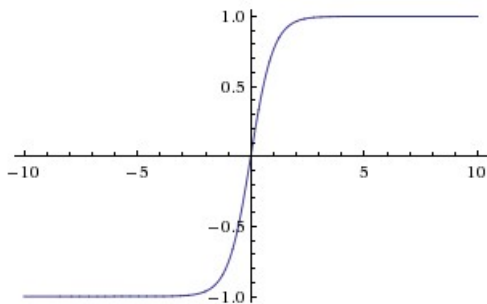
La *Tangente Iperbolica*  $\tanh : \mathbb{R} \rightarrow [-1, 1]$  è definita come  $\tanh(x) = 2\sigma(2x) - 1$

- ▶ La tangente iperbolica è una sigmoide scalata
- ▶ La sua derivata è uguale a  $\tanh'(x) = 1 - \tanh^2(x)$



# Reti Neurali

## Tangente Iperbolica



*Rappresentazione grafica Tangente Iperbolica*

# Reti Neurali

## Softplus

### Definizione

La *Softplus*  $s : \mathbb{R} \rightarrow (0, +\infty)$  è definita come  $s(x) = \log(1 + e^x)$

# Reti Neurali

## Softplus

### Definizione

La *Softplus*  $s : \mathbb{R} \rightarrow (0, +\infty)$  è definita come  $s(x) = \log(1 + e^x)$

- La softplus è una buona approssimazione della ReLU

# Reti Neurali

## Softplus

### Definizione

La *Softplus*  $s : \mathbb{R} \rightarrow (0, +\infty)$  è definita come  $s(x) = \log(1 + e^x)$

- ▶ La softplus è una buona approssimazione della ReLU
- ▶ Viene solitamente usata per sostituire la ReLU perché non presenta punti di discontinuità

# Reti Neurali

## Softplus

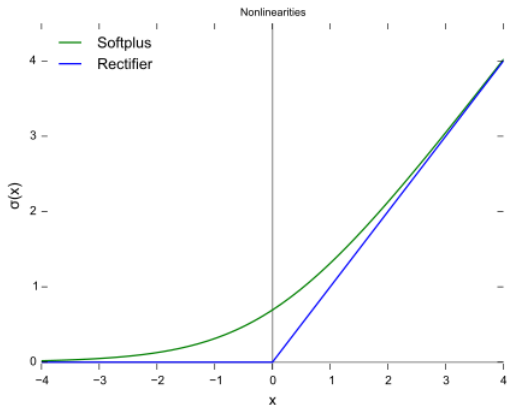
### Definizione

La *Softplus*  $s : \mathbb{R} \rightarrow (0, +\infty)$  è definita come  $s(x) = \log(1 + e^x)$

- ▶ La softplus è una buona approssimazione della ReLU
- ▶ Viene solitamente usata per sostituire la ReLU perché non presenta punti di discontinuità
- ▶ La sua derivata è uguale a  $s'(x) = \frac{1}{(1+e^{-x})}$

# Reti Neurali

## Softplus



*Confronto grafico tra ReLU e Softplus*

# Reti Neurali

## Rete Neurale

### Definizione

Una *Rete Neurale* è composta da un insieme di neuroni connessi tra loro in un grafo aciclico

# Reti Neurali

## Rete Neurale

### Definizione

Una *Rete Neurale* è composta da un insieme di neuroni connessi tra loro in un grafo aciclico

- I neuroni sono organizzati in insiemi distinti chiamati *livelli* o *layer*



# Reti Neurali

## Rete Neurale

### Definizione

Una *Rete Neurale* è composta da un insieme di neuroni connessi tra loro in un grafo aciclico

- ▶ I neuroni sono organizzati in insiemi distinti chiamati *livelli* o *layer*
- ▶ I livelli sono posti uno di seguito all'altro in modo da formare una sequenza

# Reti Neurali

## Rete Neurale

### Definizione

Una *Rete Neurale* è composta da un insieme di neuroni connessi tra loro in un grafo aciclico

- ▶ I neuroni sono organizzati in insiemi distinti chiamati *livelli* o *layer*
- ▶ I livelli sono posti uno di seguito all'altro in modo da formare una sequenza
- ▶ I livelli intermedi prendono il nome di *hidden*

# Reti Neurali

## Rete Neurale

### Definizione

Una *Rete Neurale* è composta da un insieme di neuroni connessi tra loro in un grafo aciclico

- ▶ I neuroni sono organizzati in insiemi distinti chiamati *livelli* o *layer*
- ▶ I livelli sono posti uno di seguito all'altro in modo da formare una sequenza
- ▶ I livelli intermedi prendono il nome di *hidden*
- ▶ L'output dei neuroni di un livello diventano l'input dei neuroni del livello successivo

# Reti Neurali

## Rete Neurale

- ▶ Quando si effettua il conteggio dei livelli di una rete non si considera il livello di input

# Reti Neurali

## Rete Neurale

- ▶ Quando si effettua il conteggio dei livelli di una rete non si considera il livello di input
- ▶ Una rete a *singolo livello* non presenta livelli hidden

# Reti Neurali

## Rete Neurale

- ▶ Quando si effettua il conteggio dei livelli di una rete non si considera il livello di input
- ▶ Una rete a *singolo livello* non presenta livelli hidden
- ▶ Per determinare la grandezza di una rete ci si concentra sul numero di neuroni e sui relativi pesi ad essi associati

# Reti Neurali

## Livello Fully-Connected

### Definizione

Un livello è di tipo *Fully-Connected* quando i neuroni appartenenti a due livelli adiacenti sono completamente connessi tra loro mentre i neuroni associati ad un singolo livello non condividono nessuna connessione

# Reti Neurali

## Livello Fully-Connected

### Definizione

Un livello è di tipo *Fully-Connected* quando i neuroni appartenenti a due livelli adiacenti sono completamente connessi tra loro mentre i neuroni associati ad un singolo livello non condividono nessuna connessione

- I pesi dei neuroni di ciascun livello sono salvati all'interno di matrici



# Reti Neurali

## Livello Fully-Connected

### Definizione

Un livello è di tipo *Fully-Connected* quando i neuroni appartenenti a due livelli adiacenti sono completamente connessi tra loro mentre i neuroni associati ad un singolo livello non condividono nessuna connessione

- ▶ I pesi dei neuroni di ciascun livello sono salvati all'interno di matrici
- ▶ Le righe di una matrice identificano i neuroni del livello mentre le colonne contengono i pesi di ciascun neurone

# Reti Neurali

## Livello Fully-Connected

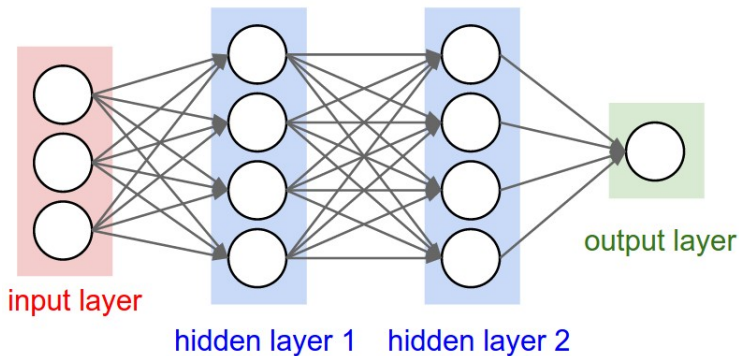
### Definizione

Un livello è di tipo *Fully-Connected* quando i neuroni appartenenti a due livelli adiacenti sono completamente connessi tra loro mentre i neuroni associati ad un singolo livello non condividono nessuna connessione

- ▶ I pesi dei neuroni di ciascun livello sono salvati all'interno di matrici
- ▶ Le righe di una matrice identificano i neuroni del livello mentre le colonne contengono i pesi di ciascun neurone
- ▶ La struttura a livelli di una rete neurale permette di sfruttare le potenzialità del calcolo matriciale

# Reti Neurali

## Livello Fully-Connected



*Una rete neurale a 3 livelli*

# Reti Neurali

## Funzionamento

Il processo di apprendimento di una rete neurale è suddiviso in quattro fasi distinte

# Reti Neurali

## Funzionamento

Il processo di apprendimento di una rete neurale è suddiviso in quattro fasi distinte

- *Inizializzazione dei pesi*

# Reti Neurali

## Funzionamento

Il processo di apprendimento di una rete neurale è suddiviso in quattro fasi distinte

- ▶ *Inizializzazione dei pesi*
- ▶ *Forward Propagation*

# Reti Neurali

## Funzionamento

Il processo di apprendimento di una rete neurale è suddiviso in quattro fasi distinte

- ▶ *Inizializzazione dei pesi*
- ▶ *Forward Propagation*
- ▶ *Calcolo della Funzione di Perdita*

# Reti Neurali

## Funzionamento

Il processo di apprendimento di una rete neurale è suddiviso in quattro fasi distinte

- ▶ *Inizializzazione dei pesi*
- ▶ *Forward Propagation*
- ▶ *Calcolo della Funzione di Perdita*
- ▶ *Back Propagation*



# Reti Neurali

## Inizializzazione dei pesi

- Al momento della nascita gli esseri umani non sono in grado di discriminare nessun tipo di oggetto a causa del mancato addestramento della loro rete neurale biologica

# Reti Neurali

## Inizializzazione dei pesi

- ▶ Al momento della nascita gli esseri umani non sono in grado di discriminare nessun tipo di oggetto a causa del mancato addestramento della loro rete neurale biologica
- ▶ Per riprodurre questo comportamento, all'inizio della fase di training, i pesi sinaptici  $w_i$  di ciascun livello vengono inizializzati in maniera casuale

# Reti Neurali

## Forward Propagation

### Definizione

La *Forward Propagation* è il meccanismo utilizzato da una rete neurale per associare un'immagine ad una determinata classe

# Reti Neurali

## Forward Propagation

### Definizione

La *Forward Propagation* è il meccanismo utilizzato da una rete neurale per associare un'immagine ad una determinata classe

- L'output dei neuroni del livello  $i$  viene moltiplicato per la matrice dei pesi del livello  $i + 1$  ottenendo il vettore  $v$

# Reti Neurali

## Forward Propagation

### Definizione

La *Forward Propagation* è il meccanismo utilizzato da una rete neurale per associare un'immagine ad una determinata classe

- ▶ L'output dei neuroni del livello  $i$  viene moltiplicato per la matrice dei pesi del livello  $i + 1$  ottenendo il vettore  $v$
- ▶ Al vettore  $v$  viene aggiunto il vettore dei bias del livello  $i + 1$

# Reti Neurali

## Forward Propagation

### Definizione

La *Forward Propagation* è il meccanismo utilizzato da una rete neurale per associare un'immagine ad una determinata classe

- ▶ L'output dei neuroni del livello  $i$  viene moltiplicato per la matrice dei pesi del livello  $i + 1$  ottenendo il vettore  $v$
- ▶ Al vettore  $v$  viene aggiunto il vettore dei bias del livello  $i + 1$
- ▶ L'output del livello  $i + 1$  si ottiene applicando la funzione di attivazione  $f$  ad ogni entry del vettore  $v$

# Reti Neurali

## Forward Propagation

### Definizione

La *Forward Propagation* è il meccanismo utilizzato da una rete neurale per associare un'immagine ad una determinata classe

- ▶ L'output dei neuroni del livello  $i$  viene moltiplicato per la matrice dei pesi del livello  $i + 1$  ottenendo il vettore  $v$
- ▶ Al vettore  $v$  viene aggiunto il vettore dei bias del livello  $i + 1$
- ▶ L'output del livello  $i + 1$  si ottiene applicando la funzione di attivazione  $f$  ad ogni entry del vettore  $v$
- ▶ Le operazioni precedenti sono svolte per tutti i livelli ad eccezione dell'ultimo

# Reti Neurali

## Calcolo della funzione di perdita

### Definizione

Una *funzione di perdita*  $L$  viene utilizzata per determinare l'errore di classificazione di una rete neurale



# Reti Neurali

## Calcolo della funzione di perdita

### Definizione

Una *funzione di perdita*  $L$  viene utilizzata per determinare l'errore di classificazione di una rete neurale

- La funzione di perdita più usata è la *Mean Squared Error (MSE)*  
$$L = \frac{1}{2} \sum (y - o)^2$$

# Reti Neurali

## Calcolo della funzione di perdita

### Definizione

Una *funzione di perdita*  $L$  viene utilizzata per determinare l'errore di classificazione di una rete neurale

- ▶ La funzione di perdita più usata è la *Mean Squared Error (MSE)*  
$$L = \frac{1}{2} \sum (y - o)^2$$
- ▶  $y$  identifica l'output della rete mentre  $o$  l'etichetta dell'esempio considerato

# Reti Neurali

## Calcolo della funzione di perdita

### Definizione

Una *funzione di perdita*  $L$  viene utilizzata per determinare l'errore di classificazione di una rete neurale

- ▶ La funzione di perdita più usata è la *Mean Squared Error (MSE)*  
$$L = \frac{1}{2} \sum (y - o)^2$$
- ▶  $y$  identifica l'output della rete mentre  $o$  l'etichetta dell'esempio considerato
- ▶ Minimizzando la funzione di perdita  $L$  si riduce l'errore di una rete neurale

# Reti Neurali

## Calcolo della funzione di perdita

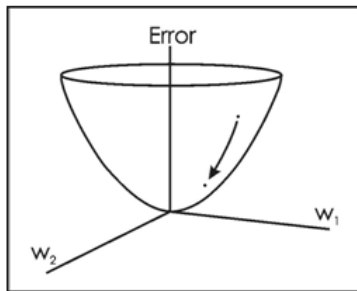
### Definizione

Una *funzione di perdita*  $L$  viene utilizzata per determinare l'errore di classificazione di una rete neurale

- ▶ La funzione di perdita più usata è la *Mean Squared Error (MSE)*  
$$L = \frac{1}{2} \sum (y - o)^2$$
- ▶  $y$  identifica l'output della rete mentre  $o$  l'etichetta dell'esempio considerato
- ▶ Minimizzando la funzione di perdita  $L$  si riduce l'errore di una rete neurale
- ▶ Calcolando la derivata di  $L$  in funzione dei pesi  $w_i$  si individua il minimo globale della funzione di perdita

# Reti Neurali

## Funzione di perdita



*Mean Squared Error (MSE). I pesi  $w_1$  e  $w_2$  sono le variabili indipendenti. La funzione di perdita  $L$  è la variabile dipendente*

# Reti Neurali

## Back Propagation

### Definizione

La *Back Propagation* è il meccanismo utilizzato da una rete neurale per correggere gli errori di classificazione. Vengono individuati i pesi  $w_i$  che hanno influenzato maggiormente l'errore commesso e viene aggiornato il loro valore in modo da ridurre la funzione di perdita

# Reti Neurali

## Back Propagation

### Definizione

La *Back Propagation* è il meccanismo utilizzato da una rete neurale per correggere gli errori di classificazione. Vengono individuati i pesi  $w_i$  che hanno influenzato maggiormente l'errore commesso e viene aggiornato il loro valore in modo da ridurre la funzione di perdita

- Per calcolare la derivata della funzione  $L$  in funzione dei pesi  $w_i$  viene usata la *regola della catena* (*chain rule*)

# Reti Neurali

## Back Propagation

### Definizione

La *Back Propagation* è il meccanismo utilizzato da una rete neurale per correggere gli errori di classificazione. Vengono individuati i pesi  $w_i$  che hanno influenzato maggiormente l'errore commesso e viene aggiornato il loro valore in modo da ridurre la funzione di perdita

- ▶ Per calcolare la derivata della funzione  $L$  in funzione dei pesi  $w_i$  viene usata la *regola della catena* (*chain rule*)
- ▶ Questa regola è usata per trovare la derivata di una funzione composta



# Reti Neurali

## Aggiornamento dei Pesi e Learning Rate

- Il nuovo valore del peso  $w_i$  è dato dalla regola di aggiornamento  
$$w_i = w_i - \eta \frac{\partial L}{\partial w_i} = w_i + \Delta w_i$$

# Reti Neurali

## Aggiornamento dei Pesi e Learning Rate

- ▶ Il nuovo valore del peso  $w_i$  è dato dalla regola di aggiornamento
$$w_i = w_i - \eta \frac{\partial L}{\partial w_i} = w_i + \Delta w_i$$
- ▶ Il *learning rate*  $\eta$  è un parametro usato per controllare la velocità di aggiornamento dei pesi

# Reti Neurali

## Aggiornamento dei Pesi e Learning Rate

- ▶ Il nuovo valore del peso  $w_i$  è dato dalla regola di aggiornamento  
$$w_i = w_i - \eta \frac{\partial L}{\partial w_i} = w_i + \Delta w_i$$
- ▶ Il *learning rate*  $\eta$  è un parametro usato per controllare la velocità di aggiornamento dei pesi
- ▶ Un learning rate alto comporta aggiornamenti rapidi, un tempo di esecuzione più basso, ma una maggiore probabilità di terminare in un minimo locale

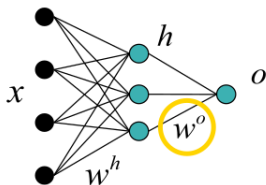
# Reti Neurali

## Aggiornamento dei Pesi e Learning Rate

- ▶ Il nuovo valore del peso  $w_i$  è dato dalla regola di aggiornamento  
$$w_i = w_i - \eta \frac{\partial L}{\partial w_i} = w_i + \Delta w_i$$
- ▶ Il *learning rate*  $\eta$  è un parametro usato per controllare la velocità di aggiornamento dei pesi
- ▶ Un learning rate alto comporta aggiornamenti rapidi, un tempo di esecuzione più basso, ma una maggiore probabilità di terminare in un minimo locale
- ▶ Al contrario, un basso learning rate riduce la probabilità di terminare in un minimo locale, ma i tempi di esecuzione si allungano notevolmente

# Reti Neurali

## Esempio Back Propagation



$$x \in \mathbb{R}^{n,1} \quad w^h \in \mathbb{R}^{n,m}$$

$$h \in \mathbb{R}^{m,1} \quad w^o \in \mathbb{R}^{1,m}$$

$$z_j^h = \sum_{i=0}^n w_{ij}^h x_i$$

$$h_j = f(z_j^h)$$

$$z^o = \sum_{j=0}^m w_j^o h_j$$

$$o = f(z^o)$$

# Reti Neurali

## Esempio Back Propagation

- Derivata della funzione  $L$  in funzione del peso  $w_j^o$

$$\frac{\partial L}{\partial w_j^o} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial w_j}$$

# Reti Neurali

## Esempio Back Propagation

- Derivata della funzione  $L$  in funzione del peso  $w_j^o$

$$\frac{\partial L}{\partial w_j^o} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial w_j}$$

- $\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$

# Reti Neurali

## Esempio Back Propagation

- Derivata della funzione  $L$  in funzione del peso  $w_j^o$

$$\frac{\partial L}{\partial w_j^o} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial w_j}$$

- $\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$

- $\frac{\partial o}{\partial z^o} = f'(z^o)$



# Reti Neurali

## Esempio Back Propagation

- Derivata della funzione  $L$  in funzione del peso  $w_j^o$

$$\frac{\partial L}{\partial w_j^o} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial w_j}$$

- $\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$

- $\frac{\partial o}{\partial z^o} = f'(z^o)$

- $\frac{\partial z^o}{\partial w_j} = h_j$

# Reti Neurali

## Esempio Back Propagation

- Risultato della derivata della funzione  $L$  in funzione del peso  $w_j^o$

$$\frac{\partial L}{\partial w_j^o} = -(y - o) \cdot f'(z^o) \cdot h_j = -\delta_j^o h_j$$

# Reti Neurali

## Esempio Back Propagation

- Risultato della derivata della funzione  $L$  in funzione del peso  $w_j^o$

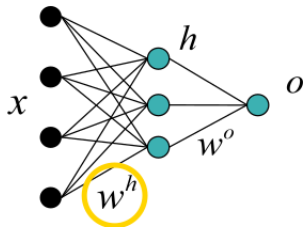
$$\frac{\partial L}{\partial w_j^o} = -(y - o) \cdot f'(z^o) \cdot h_j = -\delta_j^o h_j$$

- Aggiornamento del peso  $w_j^o$

$$\Delta w_j^o = \eta \delta_j^o h_j$$

# Reti Neurali

## Esempio Back Propagation



$$\frac{\partial L}{\partial w_{ij}^h} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial z^o} \cdot \frac{\partial z^o}{\partial h_j} \cdot \frac{\partial h_j}{\partial z_j^h} \cdot \frac{\partial z_j^h}{\partial w_{ij}^h}$$

# Reti Neurali

## Esempio Back Propagation

$$\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$$

# Reti Neurali

## Esempio Back Propagation

$$\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$$

$$\frac{\partial o}{\partial z^o} = f'(z^o)$$

# Reti Neurali

## Esempio Back Propagation

$$\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$$

$$\frac{\partial o}{\partial z^o} = f'(z^o)$$

$$\frac{\partial z^o}{\partial h_j} = w_j^o$$

# Reti Neurali

## Esempio Back Propagation

$$\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$$

$$\frac{\partial o}{\partial z^o} = f'(z^o)$$

$$\frac{\partial z^o}{\partial h_j} = w_j^o$$

$$\frac{\partial h_j}{\partial z_j^h} = f'(z_j^h)$$



# Reti Neurali

## Esempio Back Propagation

$$\frac{\partial L}{\partial o} = \frac{\partial}{\partial o} \left[ \frac{1}{2} (y - o)^2 \right] = -(y - o)$$

$$\frac{\partial o}{\partial z^o} = f'(z^o)$$

$$\frac{\partial z^o}{\partial h_j} = w_j^o$$

$$\frac{\partial h_j}{\partial z_j^h} = f'(z_j^h)$$

$$\frac{\partial z_j^h}{\partial w_{ij}^h} = x_i$$

# Reti Neurali

## Esempio Back Propagation

- Risultato della derivata della funzione  $L$  in funzione del peso  $w_{ij}^h$

$$\frac{\partial L}{\partial w_{ij}^h} = -(y - o) \cdot f'(z^o) \cdot w_j^o \cdot f'(z_j^h) \cdot x_i = -\delta_j^h x_i$$

# Reti Neurali

## Esempio Back Propagation

- Risultato della derivata della funzione  $L$  in funzione del peso  $w_{ij}^h$

$$\frac{\partial L}{\partial w_{ij}^h} = -(y - o) \cdot f'(z^o) \cdot w_j^o \cdot f'(z_j^h) \cdot x_i = -\delta_j^h x_i$$

- Aggiornamento del peso  $w_{ij}^h$

$$\Delta w_{ij}^h = \eta \delta_j^h x_i$$

# Reti Neurali

## Rete Neurale Convoluzionale

### Definizione

Una *Rete Neurale Convoluzionale* è una variante di una rete neurale classica. Permette la condivisione dei pesi sinaptici tra i neuroni di un livello e consente di discriminare le varie feature che compongono un'immagine

# Reti Neurali

## Rete Neurale Convoluzionale

### Definizione

Una *Rete Neurale Convoluzionale* è una variante di una rete neurale classica. Permette la condivisione dei pesi sinaptici tra i neuroni di un livello e consente di discriminare le varie feature che compongono un'immagine

- ▶ Viene definito un nuovo tipo di livello: il *Livello Convoluzionale*

# Reti Neurali

## Rete Neurale Convoluzionale

### Definizione

Una *Rete Neurale Convoluzionale* è una variante di una rete neurale classica. Permette la condivisione dei pesi sinaptici tra i neuroni di un livello e consente di discriminare le varie feature che compongono un'immagine

- ▶ Viene definito un nuovo tipo di livello: il *Livello Convoluzionale*
- ▶ Un livello convoluzionale è formato da diversi *filtri*

# Reti Neurali

## Rete Neurale Convoluzionale

### Definizione

Una *Rete Neurale Convoluzionale* è una variante di una rete neurale classica. Permette la condivisione dei pesi sinaptici tra i neuroni di un livello e consente di discriminare le varie feature che compongono un'immagine

- ▶ Viene definito un nuovo tipo di livello: il *Livello Convoluzionale*
- ▶ Un livello convoluzionale è formato da diversi *filtri*
- ▶ La *profondità (depth)* di un livello convoluzionale è data dal numero di filtri che lo compongono

# Reti Neurali

## Filtri e Livelli Convoluzionali

- ▶ I filtri sono le matrici contenenti i pesi sinaptici del livello convoluzionale



# Reti Neurali

## Filtri e Livelli Convoluzionali

- ▶ I filtri sono le matrici contenenti i pesi sinaptici del livello convoluzionale
- ▶ Ogni filtro ricerca all'interno delle immagini della rete una o più *feature*: linee, curve, pattern

# Reti Neurali

## Filtri e Livelli Convoluzionali

- ▶ I filtri sono le matrici contenenti i pesi sinaptici del livello convoluzionale
- ▶ Ogni filtro ricerca all'interno delle immagini della rete una o più *feature*: linee, curve, pattern
- ▶ Per apprendere nel miglior modo possibile il contenuto semantico di un'immagine, la rete deve saper ricercare feature sempre più complesse

# Reti Neurali

## Filtri e Livelli Convoluzionali

- ▶ I filtri sono le matrici contenenti i pesi sinaptici del livello convoluzionale
- ▶ Ogni filtro ricerca all'interno delle immagini della rete una o più *feature*: linee, curve, pattern
- ▶ Per apprendere nel miglior modo possibile il contenuto semantico di un'immagine, la rete deve saper ricercare feature sempre più complesse
- ▶ Mettendo in sequenza più livelli convoluzionali si possono ottenere feature complesse

# Reti Neurali

## Filtri e Livelli Convoluzionali

- ▶ I filtri sono le matrici contenenti i pesi sinaptici del livello convoluzionale
- ▶ Ogni filtro ricerca all'interno delle immagini della rete una o più *feature*: linee, curve, pattern
- ▶ Per apprendere nel miglior modo possibile il contenuto semantico di un'immagine, la rete deve saper ricercare feature sempre più complesse
- ▶ Mettendo in sequenza più livelli convoluzionali si possono ottenere feature complesse
- ▶ L'output di un generico livello convoluzionale  $i$  diventa l'input del successivo livello  $i + 1$ . Le feature prodotte da  $i$  sono meno complesse di quelle ottenute da  $i + 1$

# Reti Neurali

## Funzionamento

- I pesi dei filtri di un livello convoluzionale sono inizializzati in maniera casuale

# Reti Neurali

## Funzionamento

- ▶ I pesi dei filtri di un livello convoluzionale sono inizializzati in maniera casuale
- ▶ Vengono utilizzate le stesse funzioni di attivazione e le stesse funzioni di perdita dei livelli fully-connected

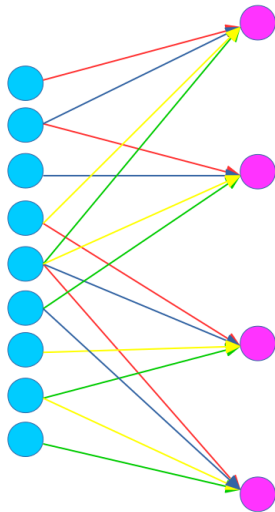
# Reti Neurali

## Funzionamento

- ▶ I pesi dei filtri di un livello convoluzionale sono inizializzati in maniera casuale
- ▶ Vengono utilizzate le stesse funzioni di attivazione e le stesse funzioni di perdita dei livelli fully-connected
- ▶ La forward e la back propagation sono le uniche fasi definite diversamente

# Reti Neurali

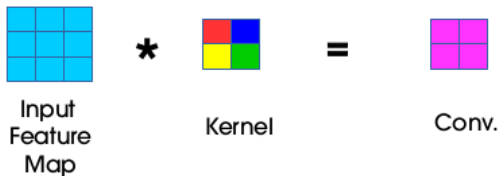
## Forward Propagation





# Reti Neurali

## Forward Propagation



- Le matrici di input e di output di un livello convoluzionale prendono il nome di *feature map*

# Reti Neurali

## Forward Propagation



- Le matrici di input e di output di un livello convoluzionale prendono il nome di *feature map*
- I filtri sono meglio conosciuti con il nome di *kernel*

# Reti Neurali

## Forward Propagation

- All'inizio della forward propagation, il kernel viene sovrapposto alla parte superiore sinistra della feature map di input

# Reti Neurali

## Forward Propagation

- ▶ All'inizio della forward propagation, il kernel viene sovrapposto alla parte superiore sinistra della feature map di input
- ▶ Viene eseguita la *convoluzione* tra le due sottomatrici ed il risultato ottenuto viene salvato nella feature map di output

# Reti Neurali

## Forward Propagation

- ▶ All'inizio della forward propagation, il kernel viene sovrapposto alla parte superiore sinistra della feature map di input
- ▶ Viene eseguita la *convoluzione* tra le due sottomatrici ed il risultato ottenuto viene salvato nella feature map di output
- ▶ Il kernel viene spostato di una posizione verso destra e viene rieseguita nuovamente la convoluzione

# Reti Neurali

## Forward Propagation

- ▶ All'inizio della forward propagation, il kernel viene sovrapposto alla parte superiore sinistra della feature map di input
- ▶ Viene eseguita la *convoluzione* tra le due sottomatrici ed il risultato ottenuto viene salvato nella feature map di output
- ▶ Il kernel viene spostato di una posizione verso destra e viene rieseguita nuovamente la convoluzione
- ▶ Terminata la riga, il kernel viene posizionato nuovamente nella parte sinistra della feature map di input, ma una riga più in basso

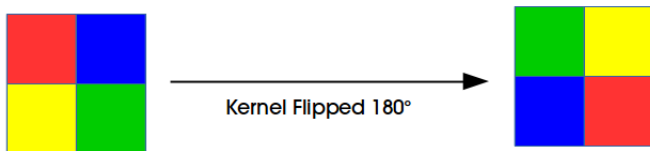
# Reti Neurali

## Forward Propagation

- ▶ All'inizio della forward propagation, il kernel viene sovrapposto alla parte superiore sinistra della feature map di input
- ▶ Viene eseguita la *convoluzione* tra le due sottomatrici ed il risultato ottenuto viene salvato nella feature map di output
- ▶ Il kernel viene spostato di una posizione verso destra e viene rieseguita nuovamente la convoluzione
- ▶ Terminata la riga, il kernel viene posizionato nuovamente nella parte sinistra della feature map di input, ma una riga più in basso
- ▶ Gli ultimi due passaggi vengono ripetuti fino a quando non è stata riempita completamente tutta la feature map di output

# Reti Neurali

## Forward Propagation

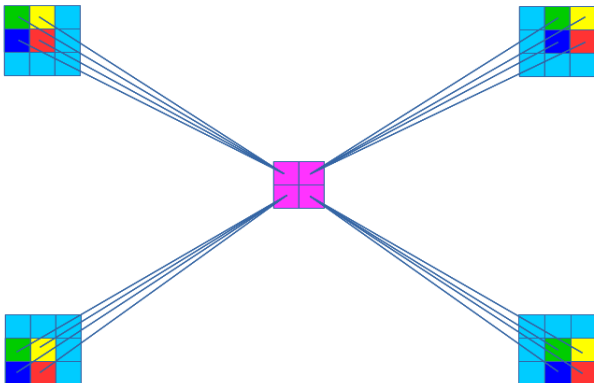


*Il kernel deve essere ruotato di 180° per poter eseguire la convoluzione*



# Reti Neurali

## Forward Propagation



*Forward Propagation di un livello convoluzionale*

# Reti Neurali

## Considerazioni Forward Propagation

- Terminata la forward propagation, viene applicata la funzione di attivazione  $f$  ad ogni entry della feature map di output

# Reti Neurali

## Considerazioni Forward Propagation

- ▶ Terminata la forward propagation, viene applicata la funzione di attivazione  $f$  ad ogni entry della feature map di output
- ▶ La dimensione della feature map di output è data da  $O = (W - K) + 1$

# Reti Neurali

## Considerazioni Forward Propagation

- ▶ Terminata la forward propagation, viene applicata la funzione di attivazione  $f$  ad ogni entry della feature map di output
- ▶ La dimensione della feature map di output è data da  $O = (W - K) + 1$
- ▶  $O$  rappresenta sia l'altezza che la larghezza della feature map di output,  $W$  quella della feature map di input e  $K$  la dimensione del kernel

# Reti Neurali

## Back Propagation

### Definizione

La Back Propagation di una rete neurale convoluzionale ha come obiettivo l'aggiornamento dei pesi contenuti nei kernel di un livello. Per ciascuno dei pesi di un kernel viene calcolata la derivata parziale  $\frac{\partial L}{\partial w_{m',n'}^l}$  che rappresenta l'influenza del peso  $w_{m',n'}^l$  sulla funzione di perdita  $L$

# Reti Neurali

## Back Propagation

### Definizione

La Back Propagation di una rete neurale convoluzionale ha come obiettivo l'aggiornamento dei pesi contenuti nei kernel di un livello. Per ciascuno dei pesi di un kernel viene calcolata la derivata parziale  $\frac{\partial L}{\partial w_{m',n'}^l}$  che rappresenta l'influenza del peso  $w_{m',n'}^l$  sulla funzione di perdita  $L$

- La Back Propagation viene suddivisa in due fasi distinte

# Reti Neurali

## Back Propagation

### Definizione

La Back Propagation di una rete neurale convoluzionale ha come obiettivo l'aggiornamento dei pesi contenuti nei kernel di un livello. Per ciascuno dei pesi di un kernel viene calcolata la derivata parziale  $\frac{\partial L}{\partial w_{m',n'}^l}$  che rappresenta l'influenza del peso  $w_{m',n'}^l$  sulla funzione di perdita  $L$

- La Back Propagation viene suddivisa in due fasi distinte
  - Il calcolo della matrice degli errori  $\delta$

# Reti Neurali

## Back Propagation

### Definizione

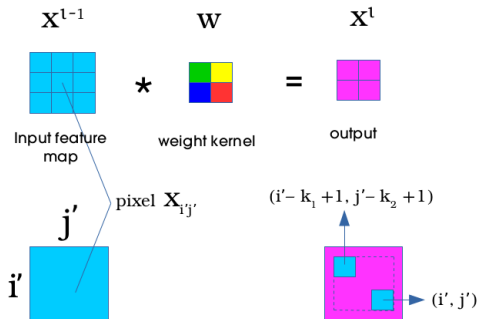
La Back Propagation di una rete neurale convoluzionale ha come obiettivo l'aggiornamento dei pesi contenuti nei kernel di un livello. Per ciascuno dei pesi di un kernel viene calcolata la derivata parziale  $\frac{\partial L}{\partial w_{m',n'}^l}$  che rappresenta l'influenza del peso  $w_{m',n'}^l$  sulla funzione di perdita  $L$

- La Back Propagation viene suddivisa in due fasi distinte
  - Il calcolo della matrice degli errori  $\delta$
  - L'aggiornamento dei pesi del kernel



# Reti Neurali

## Calcolo matrice dei $\delta$



*Le linee tratteggiate presenti nella feature map di output individuano la regione dei pixel influenzati dal pixel  $x_{i',j'}$ .  $k_1$  e  $k_2$  definiscono la grandezza della regione considerata*

# Reti Neurali

## Calcolo matrice dei $\delta$

- L'influenza del pixel  $x_{i',j'}$  sulla funzione di perdita  $L$  è data da

$$\delta_{i',j'}^l = \frac{\partial L}{\partial x_{i',j'}^l}$$

# Reti Neurali

## Calcolo matrice dei $\delta$

- L'influenza del pixel  $x_{i',j'}$  sulla funzione di perdita  $L$  è data da

$$\delta_{i',j'}^l = \frac{\partial L}{\partial x_{i',j'}^l}$$

- Applicando la regola della catena si ottiene

$$\begin{aligned}\frac{\partial L}{\partial x_{i',j'}^l} &= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \frac{\partial L}{\partial x_{i'-m,j'-n}^{l+1}} \frac{\partial x_{i'-m,j'-n}^{l+1}}{\partial x_{i',j'}^l} \\ &= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i'-m,j'-n}^{l+1} \frac{\partial x_{i'-m,j'-n}^{l+1}}{\partial x_{i',j'}^l}\end{aligned}$$

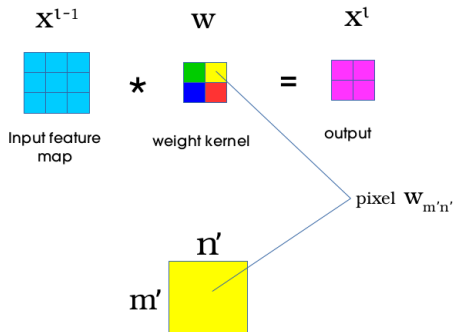
# Reti Neurali

## Calcolo matrice dei $\delta$

$$\begin{aligned}\frac{\partial L}{\partial x_{i',j'}^l} &= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i'-m,j'-n}^{l+1} w_{m,n}^{l+1} f' \left( x_{i',j'}^l \right) \\ &= \text{rot}_{180^\circ} \left\{ \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i'+m,j'+n}^{l+1} w_{m,n}^{l+1} \right\} f' \left( x_{i',j'}^l \right) \\ &= \delta_{i',j'}^{l+1} * \text{rot}_{180^\circ} \left\{ w_{m,n}^{l+1} \right\} f' \left( x_{i',j'}^l \right)\end{aligned}$$

# Reti Neurali

## Aggiornamento dei pesi



*Durante la fase di forward propagation, il peso  $w_{m',n'}$  ha contribuito a calcolare tutti i valori che costituiscono la feature map di output*

# Reti Neurali

## Aggiornamento dei pesi

- Il calcolo di  $\frac{\partial L}{\partial w_{m',n'}^l}$  usando la regola della catena è dato da

$$\begin{aligned}\frac{\partial L}{\partial w_{m',n'}^l} &= \sum_{i=0}^{W-K} \sum_{j=0}^{W-K} \frac{\partial L}{\partial x_{i,j}^l} \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} \\ &= \sum_{i=0}^{W-K} \sum_{j=0}^{W-K} \delta_{i,j}^l \cdot o_{i+m',j+n'}^{l-1} \\ &= \text{rot}_{180^\circ} \{ \delta_{i,j}^l \} * o_{m',n'}^{l-1}\end{aligned}$$

# Reti Neurali

## Aggiornamento dei pesi

- Il calcolo di  $\frac{\partial L}{\partial w_{m',n'}^l}$  usando la regola della catena è dato da

$$\begin{aligned}\frac{\partial L}{\partial w_{m',n'}^l} &= \sum_{i=0}^{W-K} \sum_{j=0}^{W-K} \frac{\partial L}{\partial x_{i,j}^l} \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} \\ &= \sum_{i=0}^{W-K} \sum_{j=0}^{W-K} \delta_{i,j}^l \cdot o_{i+m',j+n'}^{l-1} \\ &= \text{rot}_{180^\circ} \{ \delta_{i,j}^l \} * o_{m',n'}^{l-1}\end{aligned}$$

- $o_{m',n'}^{l-1} = f(x_{i',j'}^{l-1})$

# Reti Neurali

## Aggiornamento dei pesi

- Il calcolo di  $\frac{\partial L}{\partial w_{m',n'}^l}$  usando la regola della catena è dato da

$$\begin{aligned}\frac{\partial L}{\partial w_{m',n'}^l} &= \sum_{i=0}^{W-K} \sum_{j=0}^{W-K} \frac{\partial L}{\partial x_{i,j}^l} \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} \\ &= \sum_{i=0}^{W-K} \sum_{j=0}^{W-K} \delta_{i,j}^l \cdot o_{i+m',j+n'}^{l-1} \\ &= \text{rot}_{180^\circ} \{ \delta_{i,j}^l \} * o_{m',n'}^{l-1}\end{aligned}$$

- $o_{m',n'}^{l-1} = f(x_{i',j'}^{l-1})$
- La feature map di output ha una dimensione pari a  $(W - K + 1)$  sia in altezza che in larghezza



# Reti Neurali

## Aggiornamento dei pesi

- Il risultato della convoluzione tra  $\delta_{i,j}^l$  e  $o_{m',n'}^{l-1}$  individua il nuovo valore del peso  $w_{m',n'}^l$

# Reti Neurali

## Aggiornamento dei pesi

- ▶ Il risultato della convoluzione tra  $\delta_{i,j}^l$  e  $o_{m',n'}^{l-1}$  individua il nuovo valore del peso  $w_{m',n'}^l$
- ▶ La convoluzione è svolta per ciascuno dei pesi che costituiscono un kernel

# Reti Neurali

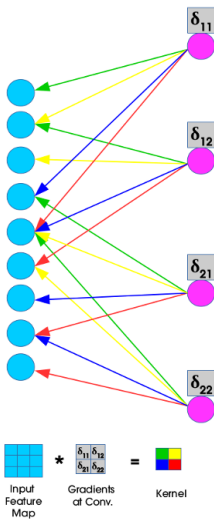
## Back Propagation



*La matrice degli errori  $\delta$  deve essere ruotata di 180° per poter eseguire la convoluzione*

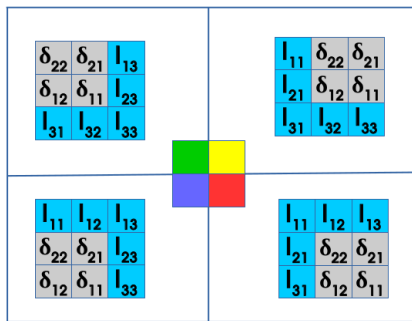
# Reti Neurali

## Esempio Back Propagation



# Reti Neurali

## Esempio Back Propagation



*Il kernel aggiornato viene ricavato dalla convoluzione tra la matrice degli errori  $\delta$  e la feature map di input*

# Implementazione della Rete

# Implementazione della Rete

## Obiettivo

- Si vuole costruire una rete neurale convoluzionale che permetta il riconoscimento di cifre numeriche scritte a mano

# Implementazione della Rete

## Obiettivo

- ▶ Si vuole costruire una rete neurale convoluzionale che permetta il riconoscimento di cifre numeriche scritte a mano
- ▶ Le cifre da riconoscere sono salvate come immagini in scala di grigio a 8 bit. Un pixel può assumere solo i valori che sono compresi nell'intervallo  $[0, 255]$



# Implementazione della Rete

## Obiettivo

- ▶ Si vuole costruire una rete neurale convoluzionale che permetta il riconoscimento di cifre numeriche scritte a mano
- ▶ Le cifre da riconoscere sono salvate come immagini in scala di grigio a 8 bit. Un pixel può assumere solo i valori che sono compresi nell'intervallo  $[0, 255]$
- ▶ L'output della rete è dato dalle 10 cifre numeriche che si vogliono riconoscere

# Implementazione della Rete

## Obiettivo

- ▶ Si vuole costruire una rete neurale convoluzionale che permetta il riconoscimento di cifre numeriche scritte a mano
- ▶ Le cifre da riconoscere sono salvate come immagini in scala di grigio a 8 bit. Un pixel può assumere solo i valori che sono compresi nell'intervallo  $[0, 255]$
- ▶ L'output della rete è dato dalle 10 cifre numeriche che si vogliono riconoscere
- ▶ La rete riceve in input un'immagine e le associa la cifra numerica corrispondente

# Implementazione della Rete

## Dati

- Le immagini che identificano gli esempi del training e del test set hanno una dimensione di  $28 \times 28$  mentre le etichette rappresentano le cifre corrispondenti alle immagini

# Implementazione della Rete

## Dati

- ▶ Le immagini che identificano gli esempi del training e del test set hanno una dimensione di  $28 \times 28$  mentre le etichette rappresentano le cifre corrispondenti alle immagini
- ▶ Il training ed il test set provengono dal database *MNIST* e contengono rispettivamente 60000 esempi di train e 10000 di test

# Implementazione della Rete

## Struttura

- La rete neurale convoluzionale sviluppata si compone di 3 livelli

# Implementazione della Rete

## Struttura

- ▶ La rete neurale convoluzionale sviluppata si compone di 3 livelli
- ▶ Due livelli hidden di tipo convoluzionale ed un livello di output di tipo fully-connected

# Implementazione della Rete

## Struttura

- ▶ La rete neurale convoluzionale sviluppata si compone di 3 livelli
- ▶ Due livelli hidden di tipo convoluzionale ed un livello di output di tipo fully-connected
- ▶ La struttura si basa su una rete neurale convoluzionale chiamata *Dnn*

# Implementazione della Rete

## Struttura

- ▶ La rete neurale convoluzionale sviluppata si compone di 3 livelli
- ▶ Due livelli hidden di tipo convoluzionale ed un livello di output di tipo fully-connected
- ▶ La struttura si basa su una rete neurale convoluzionale chiamata *Dnn*
- ▶ La Dnn è scritta in linguaggio C e adotta un approccio di tipo sequenziale



# Implementazione della Rete

## Struttura

	Input	Hidden 1	Hidden 2	Output
<i>Dimensione</i>	$28 \times 28$	$24 \times 24$	$20 \times 20$	$10 \times 1$
<i>Numero di Nodi</i>	784	2880	2000	10
<i>Profondità</i>	1	1	1	1
<i>Dimensione filtro</i>		5	5	

**Table:** Struttura Rete Neurale

	Input	Hidden 1	Hidden 2	Output
<i>Sigmoide</i>		✓	✓	✓
<i>Tanh</i>		✓	✓	✓
<i>Softplus</i>		✓	✓	✓

**Table:** Funzioni di attivazione per livello

# Implementazione della Rete

## Considerazioni

- ▶ I calcoli interni alla rete sono svolti usando il formato di dato *double* in modo da non perdere precisione numerica nei vari passaggi

# Implementazione della Rete

## Considerazioni

- ▶ I calcoli interni alla rete sono svolti usando il formato di dato *double* in modo da non perdere precisione numerica nei vari passaggi
- ▶ All'inizio della fase di training i pixel delle immagini vengono riscaldati nell'intervallo  $[0, 1]$  per poter essere compatibili con il formato di dato usato dalla rete

# Implementazione della Rete

## Considerazioni

- ▶ I calcoli interni alla rete sono svolti usando il formato di dato *double* in modo da non perdere precisione numerica nei vari passaggi
- ▶ All'inizio della fase di training i pixel delle immagini vengono riscaldati nell'intervallo  $[0, 1]$  per poter essere compatibili con il formato di dato usato dalla rete
- ▶ Tutti i dati e le strutture dati necessarie al funzionamento della rete vengono allocate all'inizio dell'esecuzione e deallocate al suo termine

# Implementazione della Rete

## Considerazioni

- ▶ I calcoli interni alla rete sono svolti usando il formato di dato *double* in modo da non perdere precisione numerica nei vari passaggi
- ▶ All'inizio della fase di training i pixel delle immagini vengono riscaldati nell'intervallo  $[0, 1]$  per poter essere compatibili con il formato di dato usato dalla rete
- ▶ Tutti i dati e le strutture dati necessarie al funzionamento della rete vengono allocate all'inizio dell'esecuzione e deallocate al suo termine
- ▶ In modo da poter confrontare tra loro i risultati ottenuti le funzioni di attivazione utilizzate sono le stesse della rete sequenziale

# Analisi dei Risultati