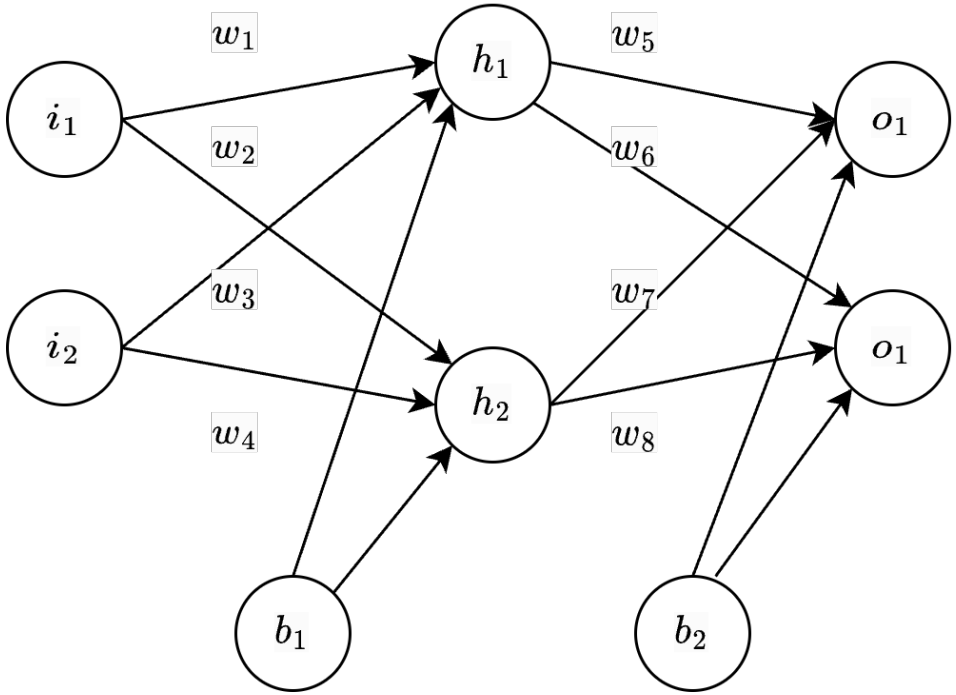
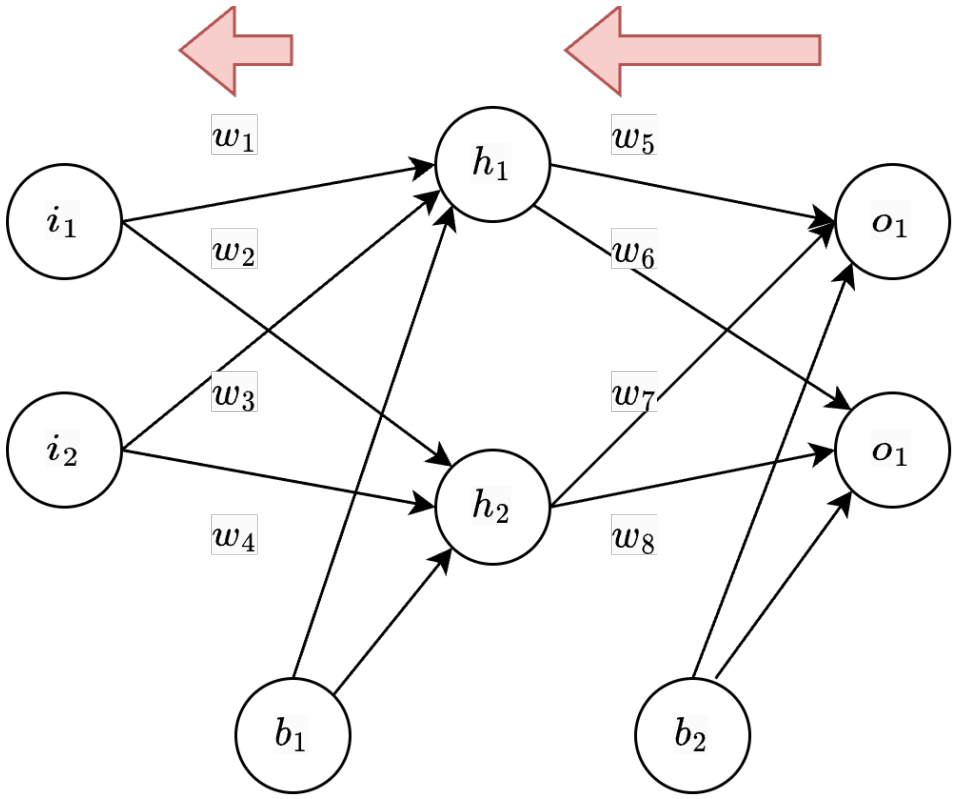
Backpropagation issues in deep networks



- Backpropagation: propagazione all'**indietro** dei gradienti
- **Necessaria** per l'aggiornamento dei parametri [w1, … , w8] e dei bias [b1, b2]

Backpropagation issues in deep networks



- **Problema**: il contributo dei gradienti è ***numericamente*** più alto nei livelli più vicini all'uscita
- Man mano che ci avviciniamo all'input della rete neurale i **gradienti diminuiscono sempre di più**
- La conseguenza è che l'aggiornamento dei parametri nei livelli più vicini all'input avviene più **lentamente**

Vanishing/Exploding Gradient

Può succedere che i gradienti diventino sempre più piccoli fino a diventare pari a zero:

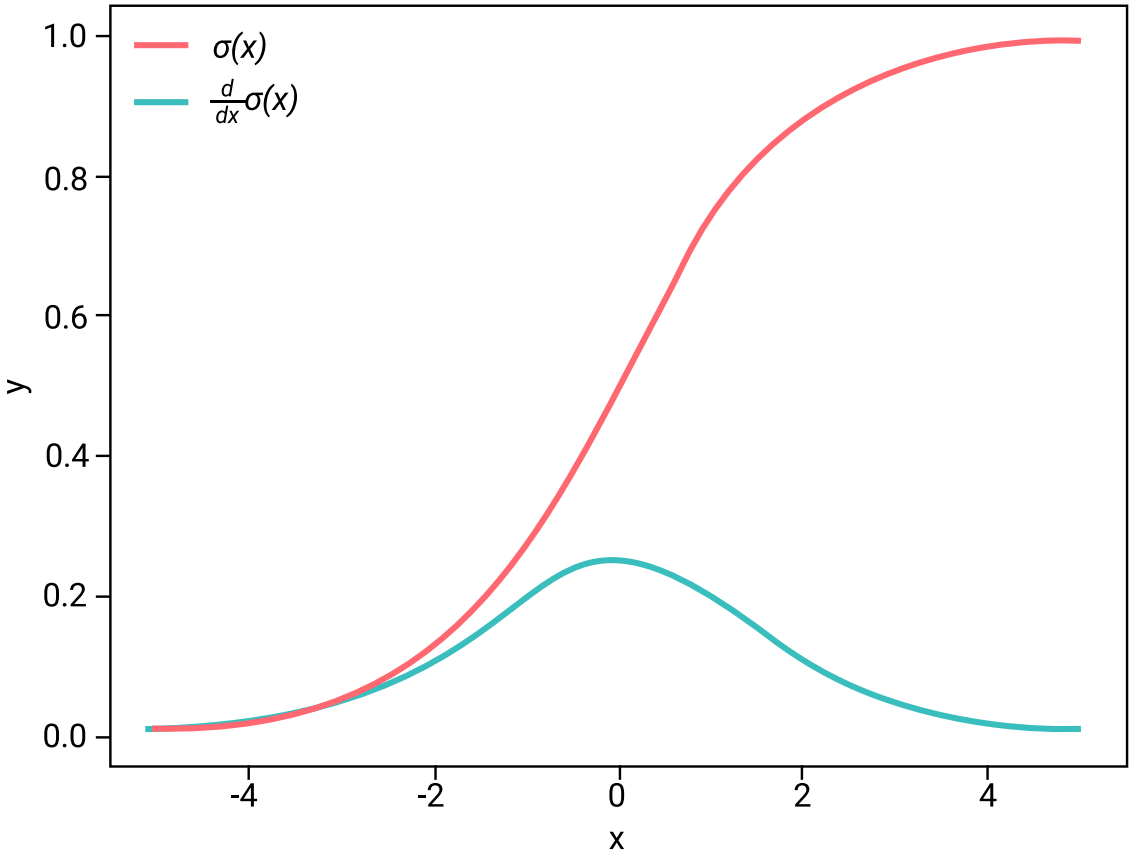
- I pesi non si aggiornano più o lo fanno molto lentamente
- **Vanishing Gradient**

Di contro può accadere che i gradienti diventino sempre più grandi fino a generare variazioni dei pesi molto elevate:

- Le prestazioni non convergono più
- **Exploding Gradient**

Vanishing Gradient

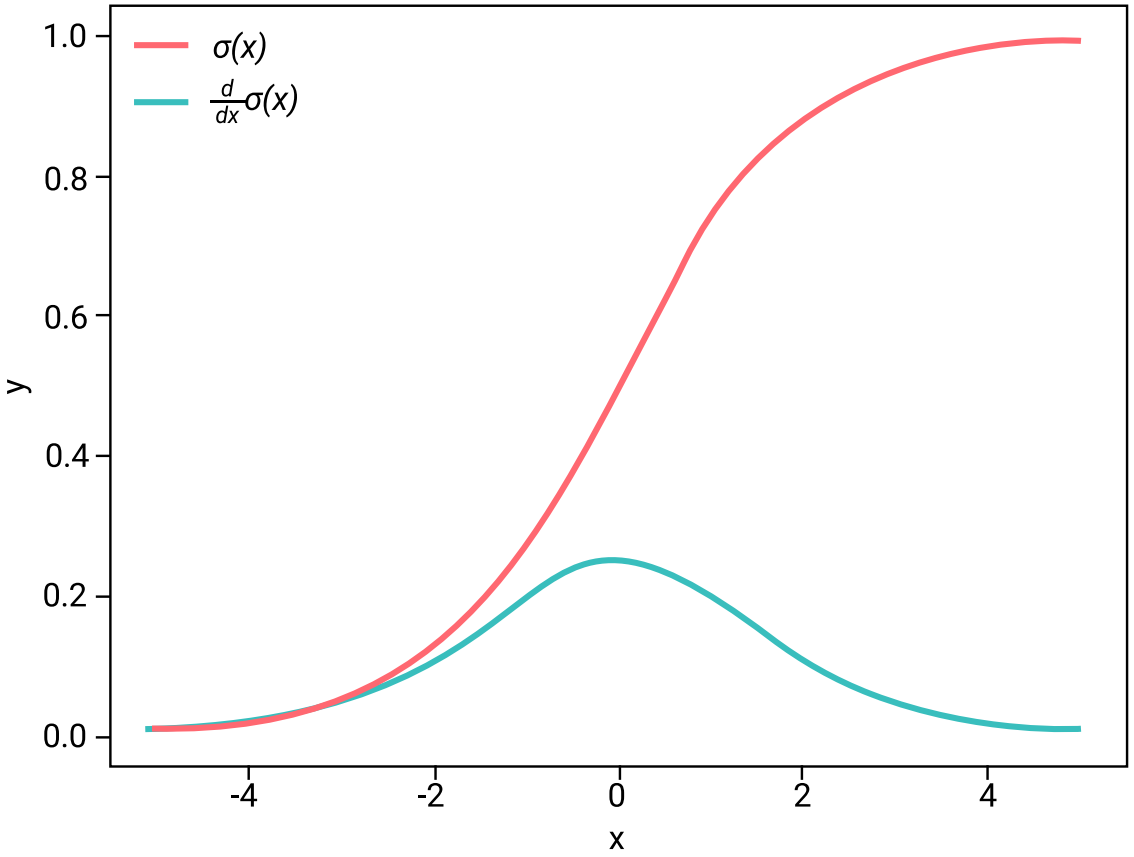
Sigmoide e sua derivata



- Causa principale: la non linearità delle funzioni di attivazione
- Alta varianza fra input e output: mappano uno spazio di input ampio ([-∞, +∞])in uno spazio di output molto ristretto (0, 1):
- Input molto piccoli o molto grandi vengono mappati a 0 o 1 con una derivata che tende a 0

Exploding Gradient

Sigmoide e sua derivata

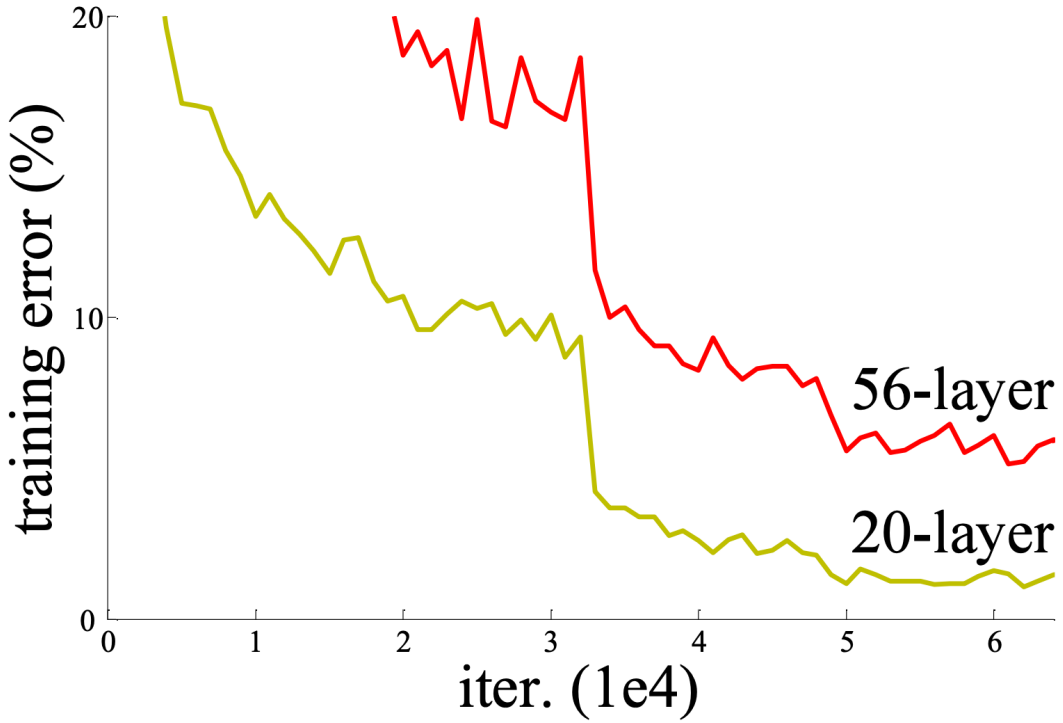


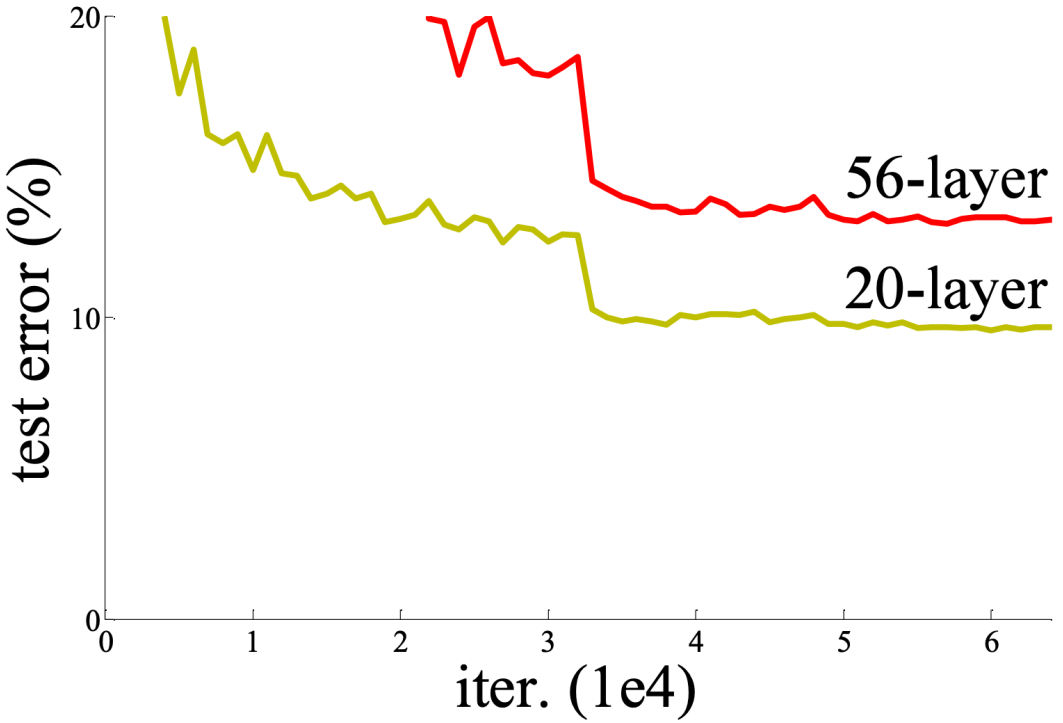
- Problema duale al vanishing gradient
- Se i pesi iniziali generano degli errori grandi durante la fase di backpropagation i gradienti saranno molto grandi
- Avere gradienti molto grandi significa avere variazioni dei pesi molto grandi:
  ciò può rendere **instabile** la rete
- I parametri potrebbero addirittura andare in overflow con valori NaN

Going deeper

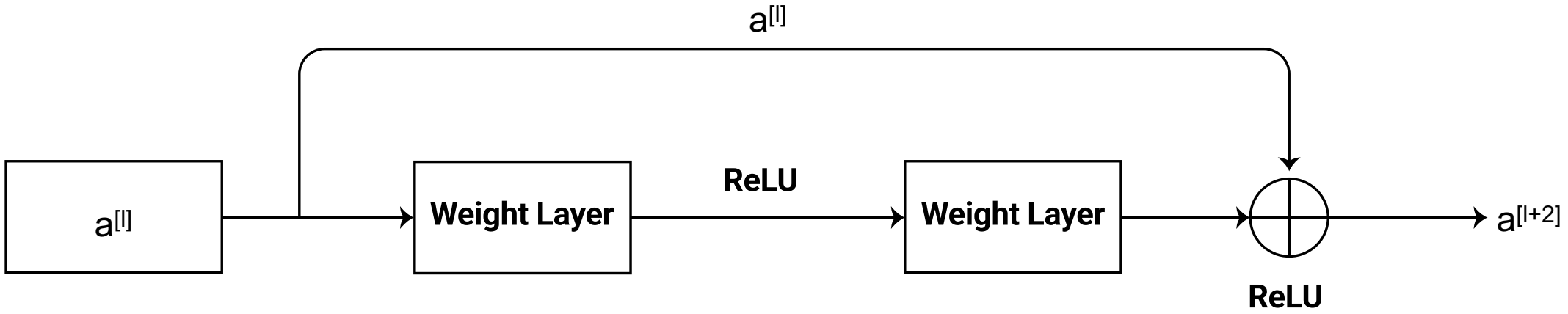
- Aggiungere livelli, dal punto di vista concettuale, dovrebbe rendere migliori le prestazioni di una CNN (in particolare dovrebbe minimizzare l'errore)
- Dal punto di vista pratico/empirico, a causa del vanishing o exploding gradient, dopo una fase di discesa, l'accuracy tenderà a **scendere** man mano che aggiungiamo livelli alla nostra rete
- La performance della rete **degrada** rapidamente man mano che aumentiamo il numero di livelli
- I ricercatori chiamano questo problema «**Degradation problem**»

Degradation Problem





Residual Block



- **Note:** usiamo ReLU come funzione di attivazione
- Facciamo una scorciatoia per cui un livello viene aggiunto direttamente all'output di un intero blocco
- L'idea è quella di "passare" ai livelli più interni l'informazione dei livelli meno interni

$$a^{[l+2]} = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

Residual Block

- Mapping diretto

$$x \implies H(x)$$

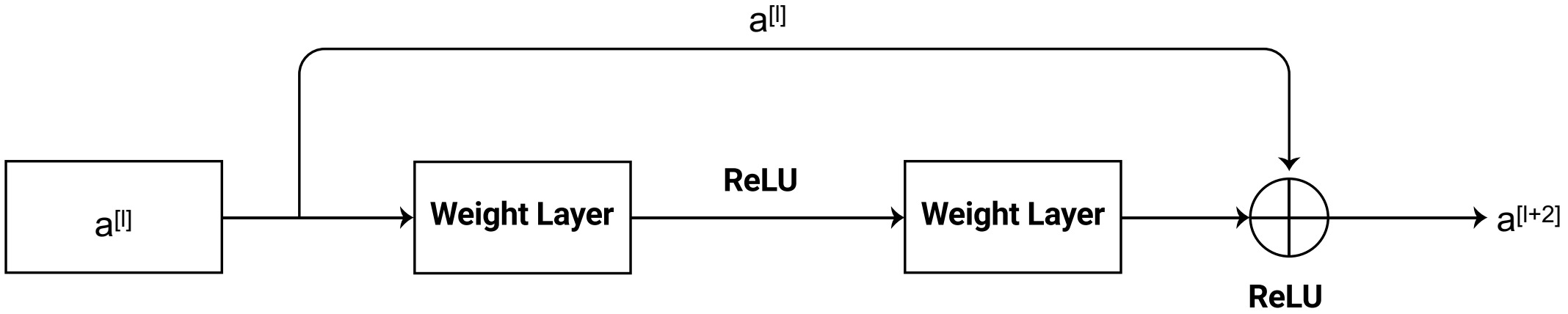
- Mapping **residuale**

$$F(x) = H(x) - x$$

- Successivamente l'output è calcolato come:

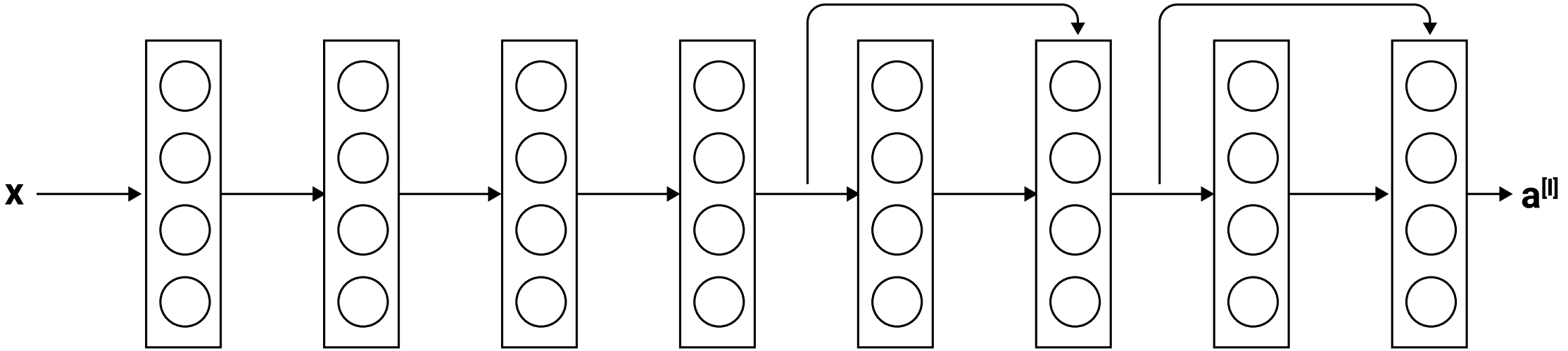
$$H(x) = F(x) + x$$

Residual Block



- Il concetto è quello di aggiungere un **residuo** all'informazione proveniente dal livello l
- I gradienti vengono propagati all'indietro in maniera efficiente grazie alle skip connections
  - Soluzione al problema del vanishing gradient

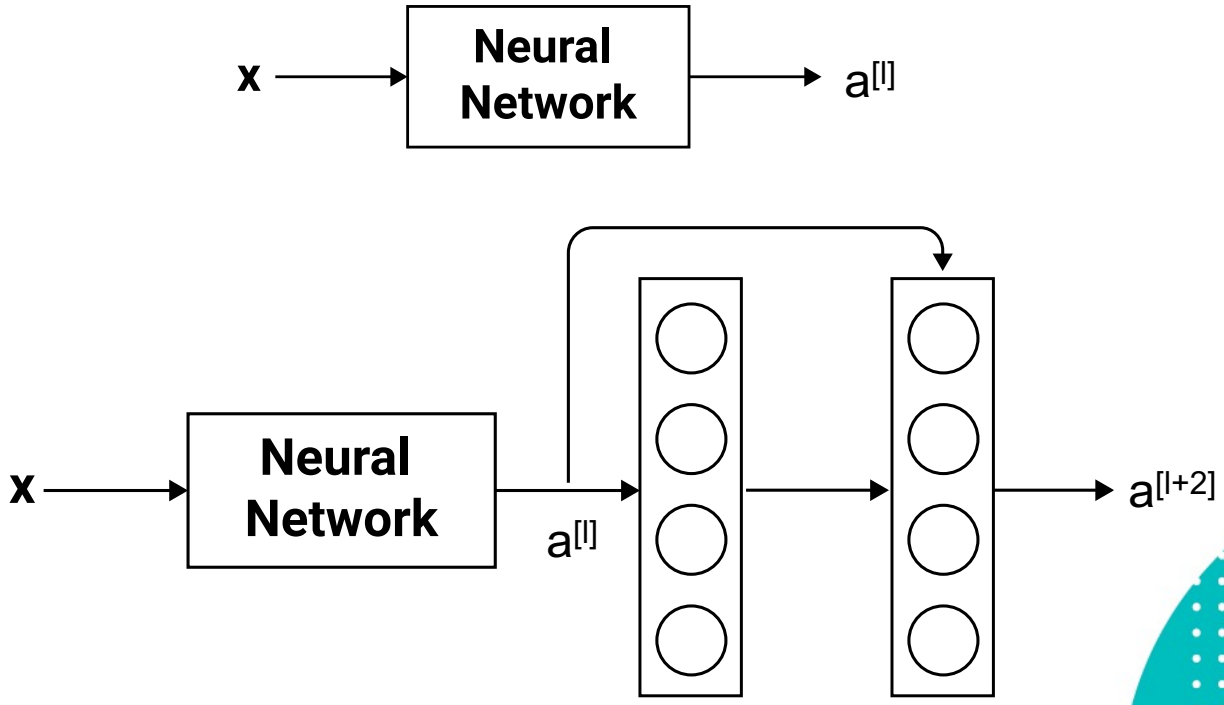
Residual Network



- È una rete formata da molti blocchi residuali
- La presenza dei blocchi residuali trasporta informazione nei livelli più deep
- Ai livelli più deep l'errore convergerà grazie alla presenza dei blocchi residuali

Residual Network

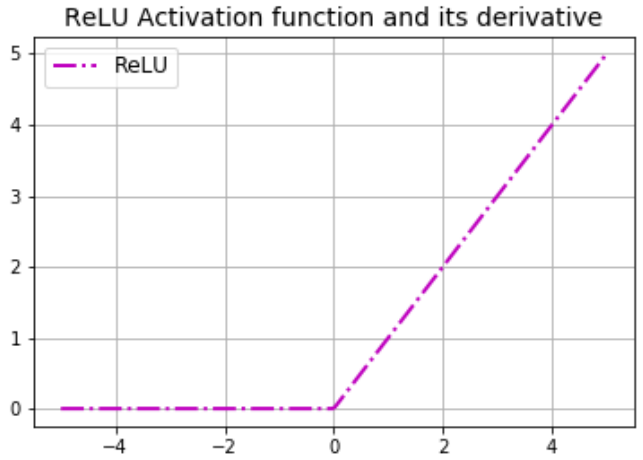
- Supponiamo di avere una rete neurale "plain" e una che aggiunge alla plain un blocco residuale
- Ci aspettiamo che le prestazioni della residuale siano **almeno pari alla rete plain**
- Se i pesi dei livelli del path interno sono pari a zero la rete avrà imparato **l'identity mapping**

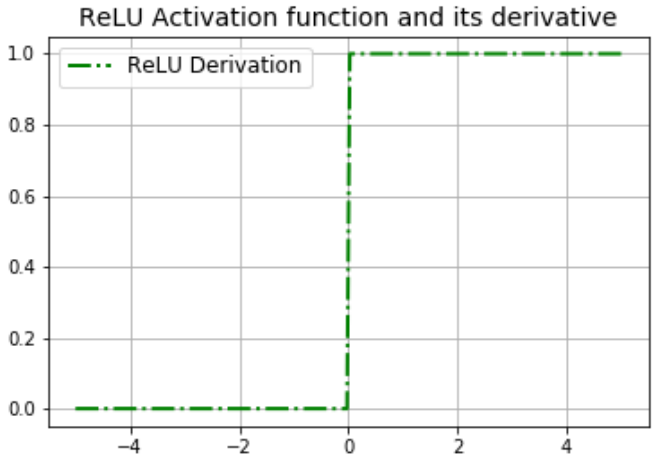


Residual Network

- Funzione di attivazione: ReLU
- Questo significa che tutte le attivazioni saranno maggiori o uguali a zero (con l'eccezione dell'input X)

$$a^{[l+2]} = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$





Residual Network

Nel caso peggiore:

- I livelli più deep avranno **almeno** il livello L da cui partire evitando il degrado dell'errore
- In questo caso si dice che i livelli più deep "imparano" la **funzione identità**

Nel caso migliore:

- I livelli più deep potranno **migliorare le performance** del livello L andando a modificare i propri parametri