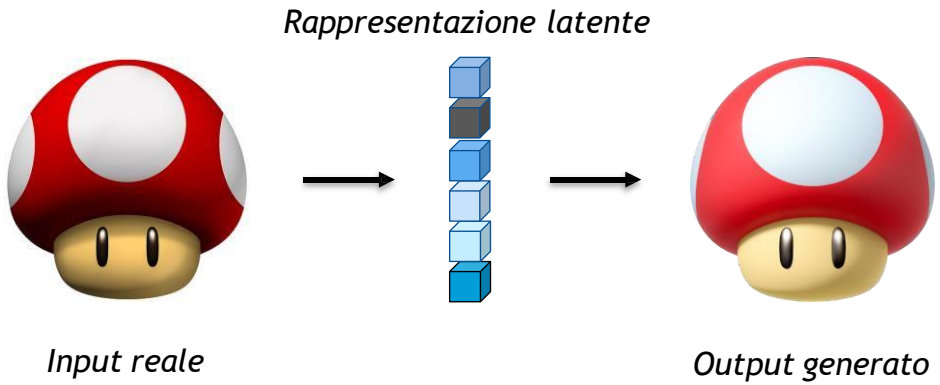
Auto-Encoder

Auto-Encoder

Definizione

Un Auto-Encoder, **AE**, è un tipo di rete neurale che sfrutta il concetto di variabile latente ed è in grado di ridurre la dimensionalità dell'input laddove i metodi tradizionali come la PCA non funzionano a causa della complessità dei dati.

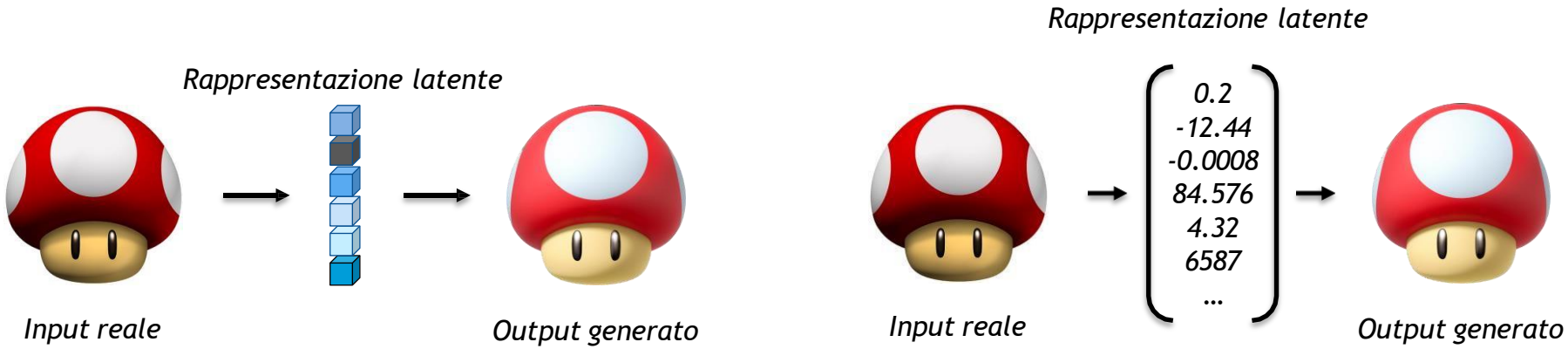
Partendo dai dati di input, li riduce dimensionalmente ottenendo le variabili latenti, che poi sfrutta per effettuare una ricostruzione.



Auto-Encoder

Definizione

La rappresentazione latente, il vettore latente, lo spazio latente…ci sono molteplici modi per descriverlo…non sarà nient'altro che una raccolta di valori.



Questa raccolta di valori sarà significativa per l'**AE**, il quale avrà appreso, durante la fase di addestramento, come 'trasformali' e decodificarli in un dato nuovamente significativo.

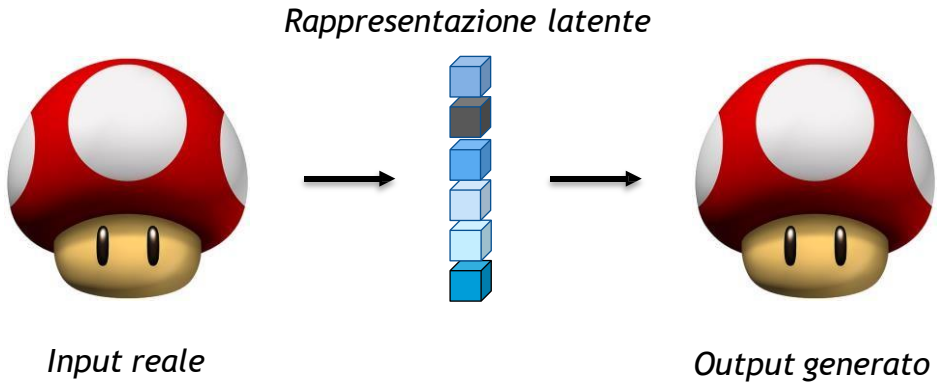
Auto-Encoder

L'ottimo

Un **AE** in grado di ricostruire **al meglio** l'input fornito, *ha correttamente condensato le informazioni necessarie alla ricostruzione in una variabile latente*.

In questa, saranno presenti le caratteristiche determinanti e fondamentali alla ricostruzione del dato. Non è detto però che vi siano solo quelle.

*Lo spazio latente ottimale si ha quando questo ha la **minima** dimensione che permette di ricostruire correttamente l'input.*



Auto-Encoder

Rappresentazione

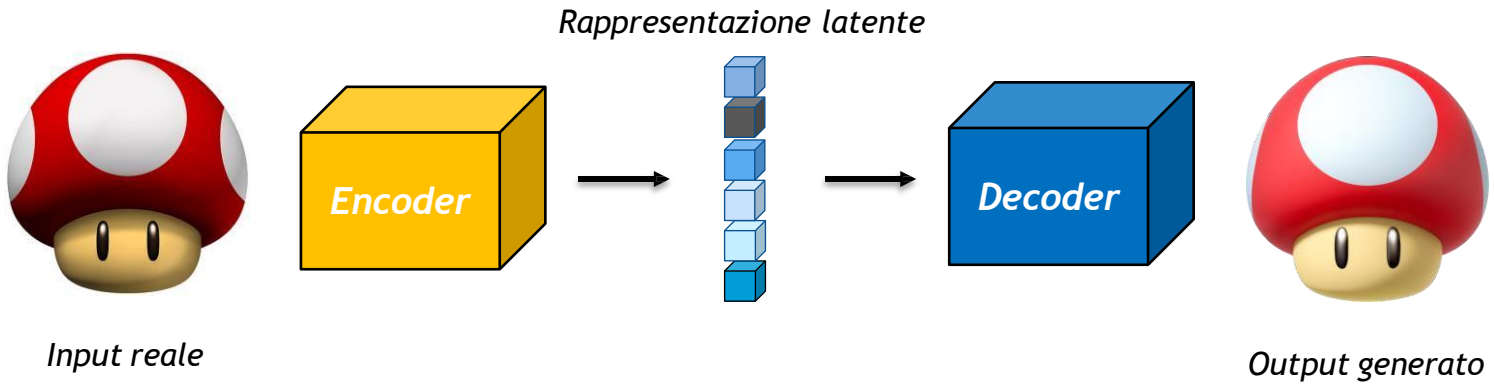
L'Auto-Encoder è costituito da due blocchi:

Encoder

Dato l'input, codifica lo spazio latente e ne riduce la dimensionalità al fine di crearne una rappresentazione condensata.

Decoder

Dallo spazio latente, esegue una decodifica al fine di generare un output dimensionalmente pari all'input fornito inizialmente.



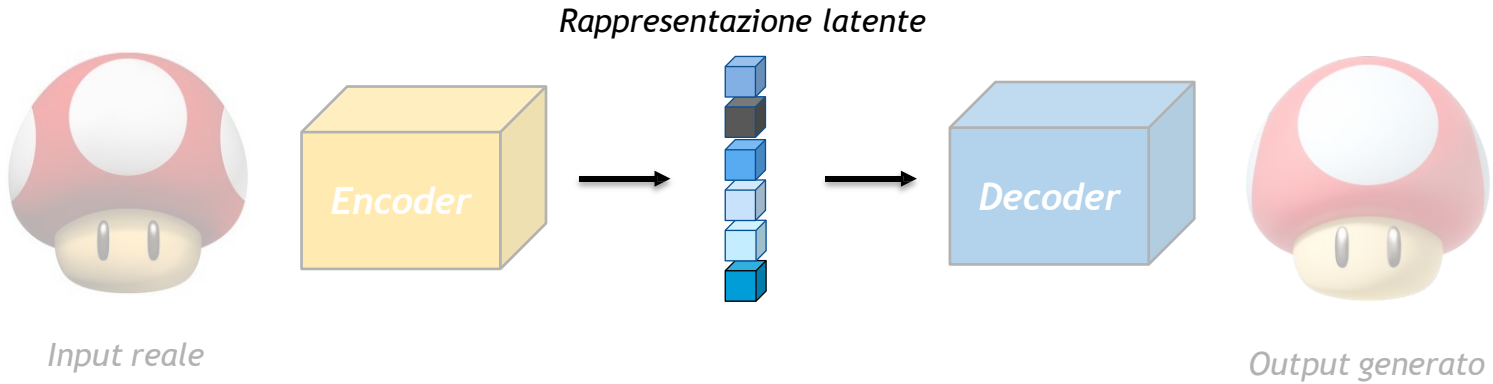
Auto-Encoder

Rappresentazione

A completare la rappresentazione di un **AE** c'è poi il componente centrale che separa ***Encoder*** **e** ***Decoder***, chiamato:

Code *(codice) o* ***Bottleneck*** *(collo di bottiglia)*

Nel **bottleneck** risiede lo spazio latente, ossia lo spazio in cui si potranno trovare quelle raccolte di valori che, assieme, costituiscono i *vettori latenti* che trasformano l'input nell'output.

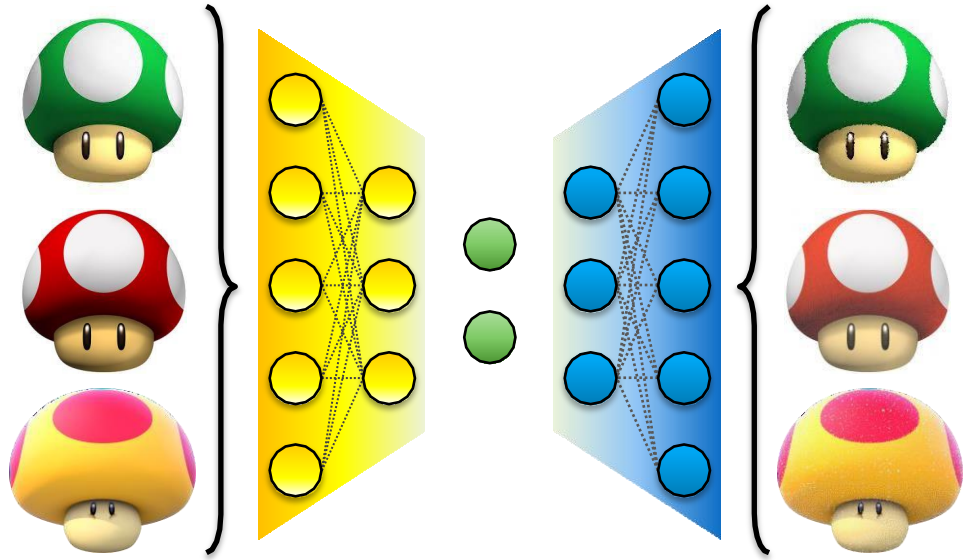


Auto-Encoder

Rappresentazione

Un **AE** può essere idealmente riassunto dalla figura sottostante: una rete neurale costituita di strati di neuroni completamente connessi che:

- *Inizialmente estrae le caratteristiche fondamentali dell'input.*
- *Le codifica in uno spazio latente.*
- *Infine, sfrutta le caratteristiche codificate per ricostruire l'output.*



Auto-Encoder

Rappresentazione

La rappresentazione architetturale di un **AE** non è però vincolata al solo uso di neuroni connessi fra loro in una maniera ‘completamente connessa’.

Sulla base del problema affrontato e della richiesta, si possono trovare:

- ***AE** costituiti di strati completamente connessi.*
- ***AE** costituiti di strati convoluzionali.*
- ***AE** costituiti di celle temporali: LSTM, GRU…*
- ***AE** costituiti di celle transformer.*
- …

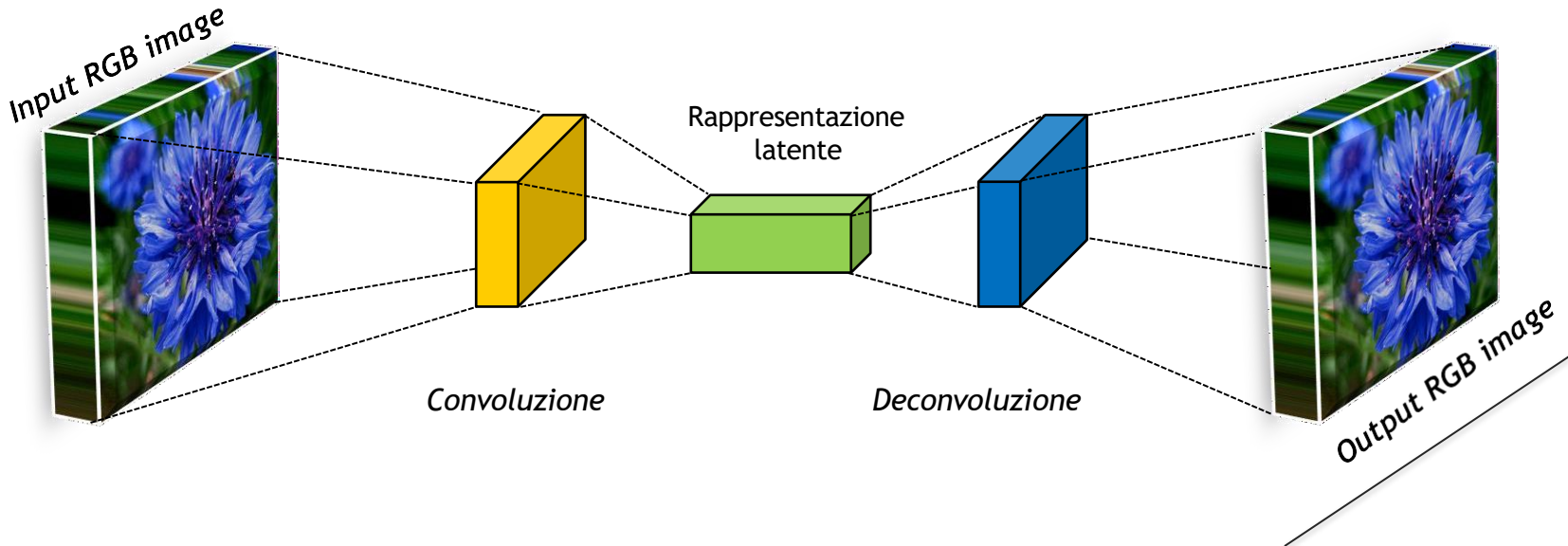
Auto-Encoder

Convolutional AE

Quando si tratta la ricostruzione di immagini, si parla di **Convolutional AE**.

Più nello specifico, l'***encoder*** e il ***decoder*** sono due modelli di reti neurali convoluzionali, due **CNN**, che lavorano sequenzialmente:

- L'encoder sfrutta l'operazione di **convoluzione** per ridurre la dimensione dell'input.
- Il decoder sfrutta l'operazione di **deconvoluzione** per ritornare alle dimensioni originali dell'input.



Auto-Encoder

Convolutional AE

Nell'ambito pratico, i vari framework di **ML** e **DL** permettono di accedere alle funzionalità e alla costruzione di Auto-Encoder convoluzionali fornendo:

- *L'accesso ai layer di convoluzione per la parte di codifica.*
- *L'accesso ai layer di de-convoluzione per la parte di decodifica.*

In **PyTorch**, come per ogni altro layer, il modulo di riferimento è:

torch.nn

Di seguito, alcuni riferimenti specifici:

- *Conv2d*
- *ConvTranspose2d*

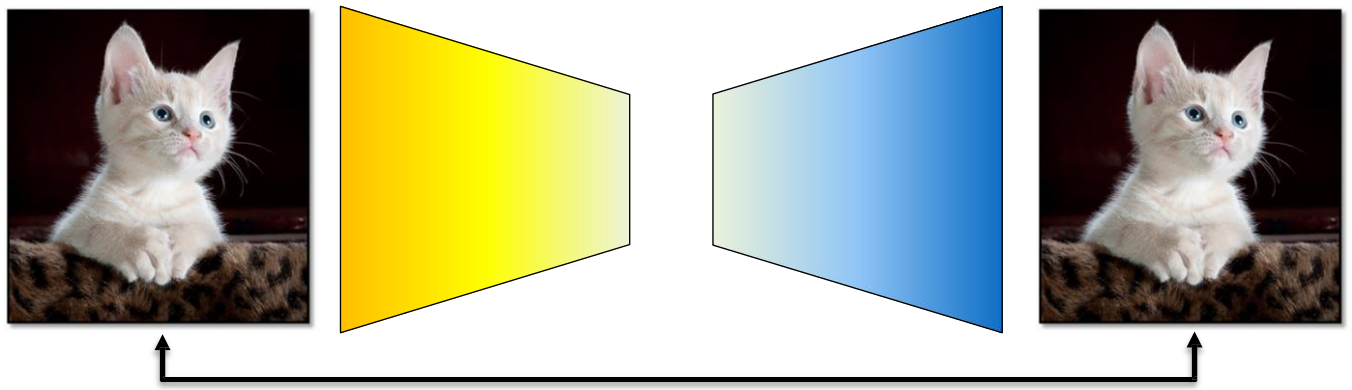
Auto-Encoder

Punti di interesse: supervisione

Uno fra i punti fondamentali da cogliere riguardo gli **AE** è che sono architetture nelle quali, generalmente, l'etichettatura è bypassata e ad avere massima importanza è il dato stesso.

Sono architetture ad **apprendimento non supervisionato** poiché permettono di sfruttare il dato stesso come input e output della pipeline. In parole povere:

L'input è esso stesso l'etichetta.



Auto-Encoder

Punti di interesse: apprendimento

La struttura di un **AE** è direttamente correlata al modo in cui è intesa la sua fase di apprendimento. In termini generali, l'obbiettivo è uno:

*Ridurre la **reconstruction loss**.*

Per assicurare questo, però, è importante porre attenzione a tre aspetti:

1. Dimensione del bottleneck.
2. Attivazioni dei neuroni.
3. Profondità della rete.

Se non si imponessero vincoli, una soluzione ottimale della rete per ottenere una ricostruzione perfetta dell'output sarebbe quella di imparare a copiare esattamente l'input.

Auto-Encoder

Punti di interesse: apprendimento

1. **Dimensione del bottleneck**.

Imporre una struttura a collo di bottiglia, per la quale si hanno, in generale, meno neuroni nello strato intermedio che in quello di input:

- Forza la struttura dell'*Encoder* ad apprendere cosa realmente è importante dell'input per ottenerne una caratterizzazione.
- Forza la struttura dell'*Encoder* ad apprendere cosa trascurare dell'input perché non di interesse al fine della ricostruzione.

Auto-Encoder

Punti di interesse: apprendimento

2. **Attivazioni dei neuroni.**

Senza l'applicazione di attivazioni non-lineari negli strati di ***Encoder*** e ***Decoder*** si incorre nella possibilità di risolvere (**non è garantito**) il problema di ricostruzione con metodi deterministi e lineari: PCA ad esempio…

Auto-Encoder

Punti di interesse: apprendimento

3. **Profondità della rete.**

Apprendere le caratteristiche dell'input è un lavoro tanto complesso quanto complesso è l'input stesso.

Per poter dare modo ad un encoder di estrarre relazioni fra i dati, combinarle fra loro e trovare pattern complessi, è necessario imporre una struttura profonda.
Ne sono un esempio i *feature extractor* di una generica **CNN**.

Auto-Encoder

Tipologie

Nonostante tutto, di **AE** ne esistono diverse varianti. Ognuna di queste con il suo particolare scopo ed architettura.

È possibile quindi trovare **AE** con diverse tipologie di ***loss*** o non del tutto non-supervisionati.

Di seguito i principali tipi:

- *Undercomplete*.
- *Sparse*
- *Contractive*.
- *Denoising*.
- *Variational*.

Auto-Encoder

Tipologie

Undercomplete

È la versione standard di **AE** finora descritta:

- *Ha un input n-dimensionale.*
- *Produce un output n-dimensionale.*
- *Ha un bottleneck di dimensione minore per ottenere **riduzione di dimensionalità**.*
- *Scopo finale è quello di apprendere come comprimere al meglio l'input.*

La loss di queste architettura è chiamata *reconstruction loss* ed è in genere legata all'input stesso: per le immagini può essere sufficiente una semplice differenza fra l'input e la ricostruzione.

Non possiede meccanismi di regolarizzazione 'automatici'.

Auto-Encoder

Tipologie

Sparse

Aggiunge un sistema di regolarizzazione all'**AE** standard:

- *Ha un input n-dimensionale e produce un output n-dimensionale.*
- *Il bottleneck ha lo scopo di far apprendere come comprimere al meglio l'input.*

La *reconstruction loss* contiene un termine ulteriore che regolarizza la rete 'modificando' il numero di neuroni attivi nel layer (simil *dropout*).

Si ottiene questo effetto facendo sì di **penalizzare l'attivazione dei neuroni stessi**.

In generale, i termini di penalizzazione possono essere:

- ***L1***: considera la somma dei valori assoluti delle attivazioni dei neuroni.
- ***KL-Divergence***: considera una raccolta di neuroni e ne guarda le attivazioni. Per la penalizzazione viene presa in considerazione l'attivazione media nella raccolta.

Auto-Encoder

Tipologie

Contractive

Aggiunge un sistema di regolarizzazione all'**AE** standard**:**

- *Ha un input n-dimensionale e produce un output n-dimensionale.*
- *Il bottleneck ha lo scopo di far apprendere come comprimere al meglio l'input.*

Regolarizza in modo da non apprendere trasformazioni 'banali' quali l'identità e vincola la rete ad avvicinare chi è simile e distanziare chi non lo è.

In termini pratici:

Se l'input cambia di poco, la rappresentazione latente deve cambiare poco.

La loss da ottimizzare sarà data dalla *reconstruction* che contribuirà a cogliere differenze fra diversi input e dal regolarizzatore che contribuirà a ignorare le piccole variazioni.

Auto-Encoder

Tipologie

Denoising

Diversamente dai classici **AE** finora descritti**:**

- *Ha un input n-dimensionale e produce un output n-dimensionale.*
- *L'input è 'rumoroso', l'output viene ripulito del rumore.*
- *Si codifica una rappresentazione che separi rumore ed input in modo da separarli.*

Il tipo di *reconstruction loss* utilizzata è generalmente della tipologia L1 o L2: differenza fra valori assoluti o differenza al quadrato fra ricostruzione e ground-truth (l'immagine non rumorosa)

Come si può notare, non è un apprendimento completamente non supervisionato in quanto si dovrà possedere sia la versione 'pulita' che la versione 'rumorosa' dell'input.

Auto-Encoder

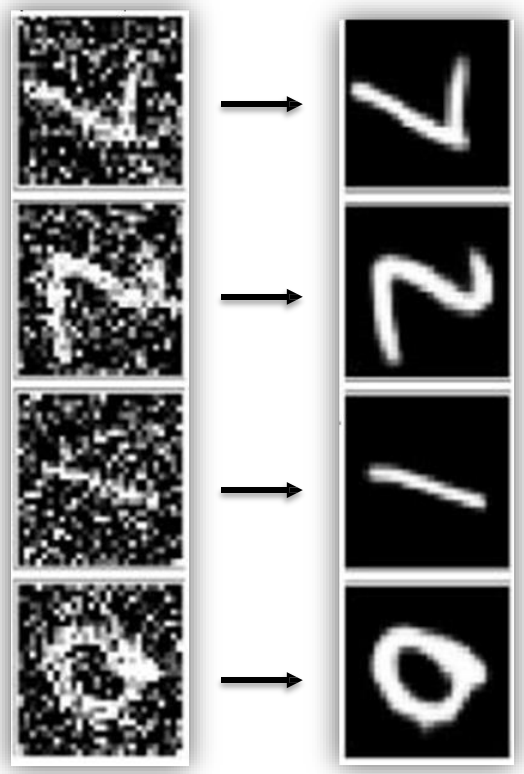
Utilizzi

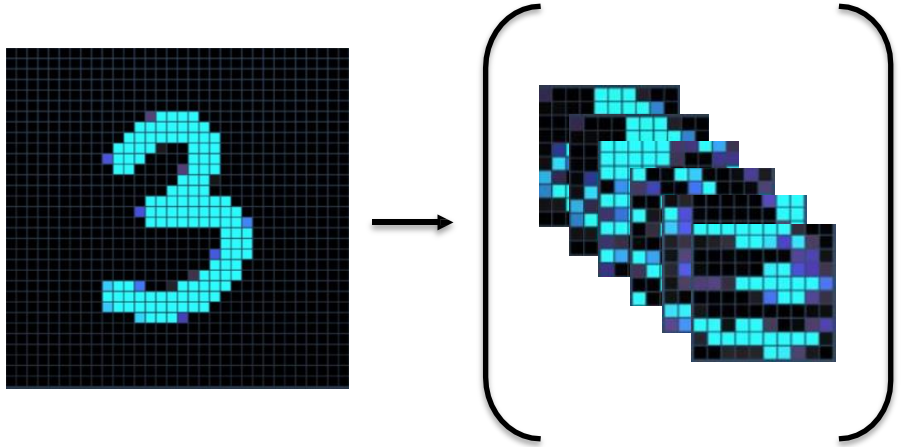
Gli **AE** sono un tipo di struttura estremamente utile e il loro utilizzo spazia in molteplici campi e casi d'uso:

- *Riduzione della dimensionalità.*
- *Estrazione delle features.*
- *Rimozione del rumore, denoising.*
- *Anomaly detection.*
- *Generazione dati sintetici.*
- *Pre-training dei modelli di deep learning.*

Auto-Encoder

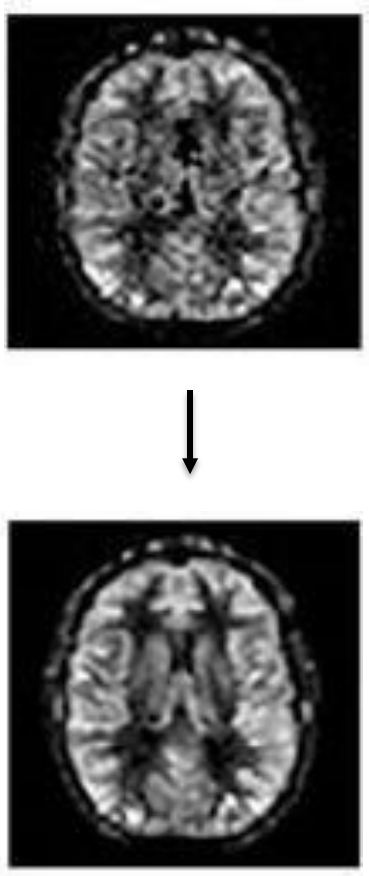
Utilizzi

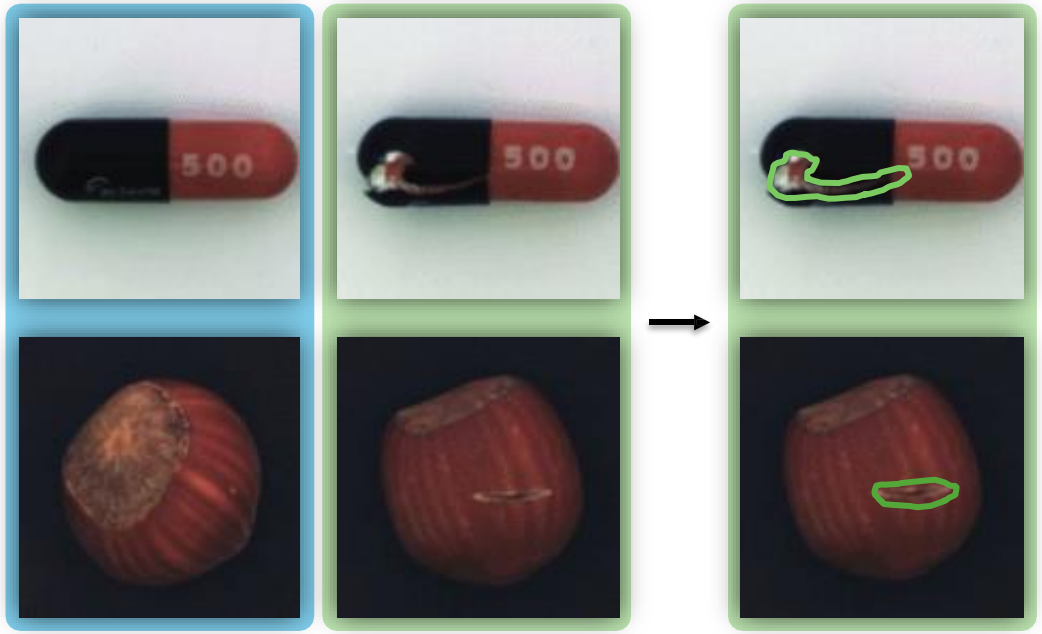




Auto-Encoder

Utilizzi





Auto-Encoder

Utilizzi

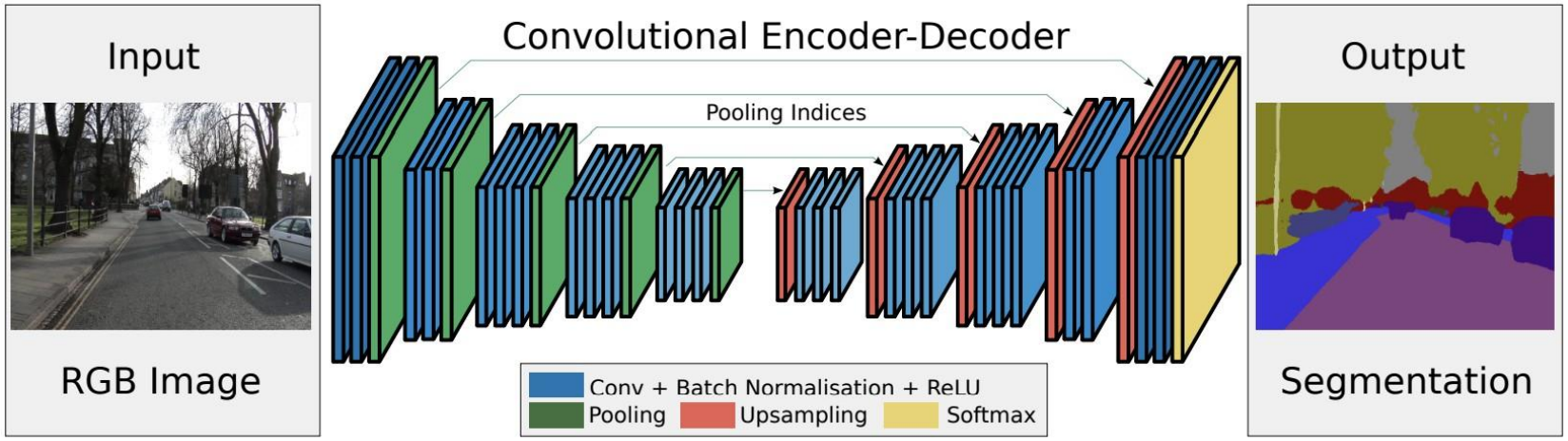
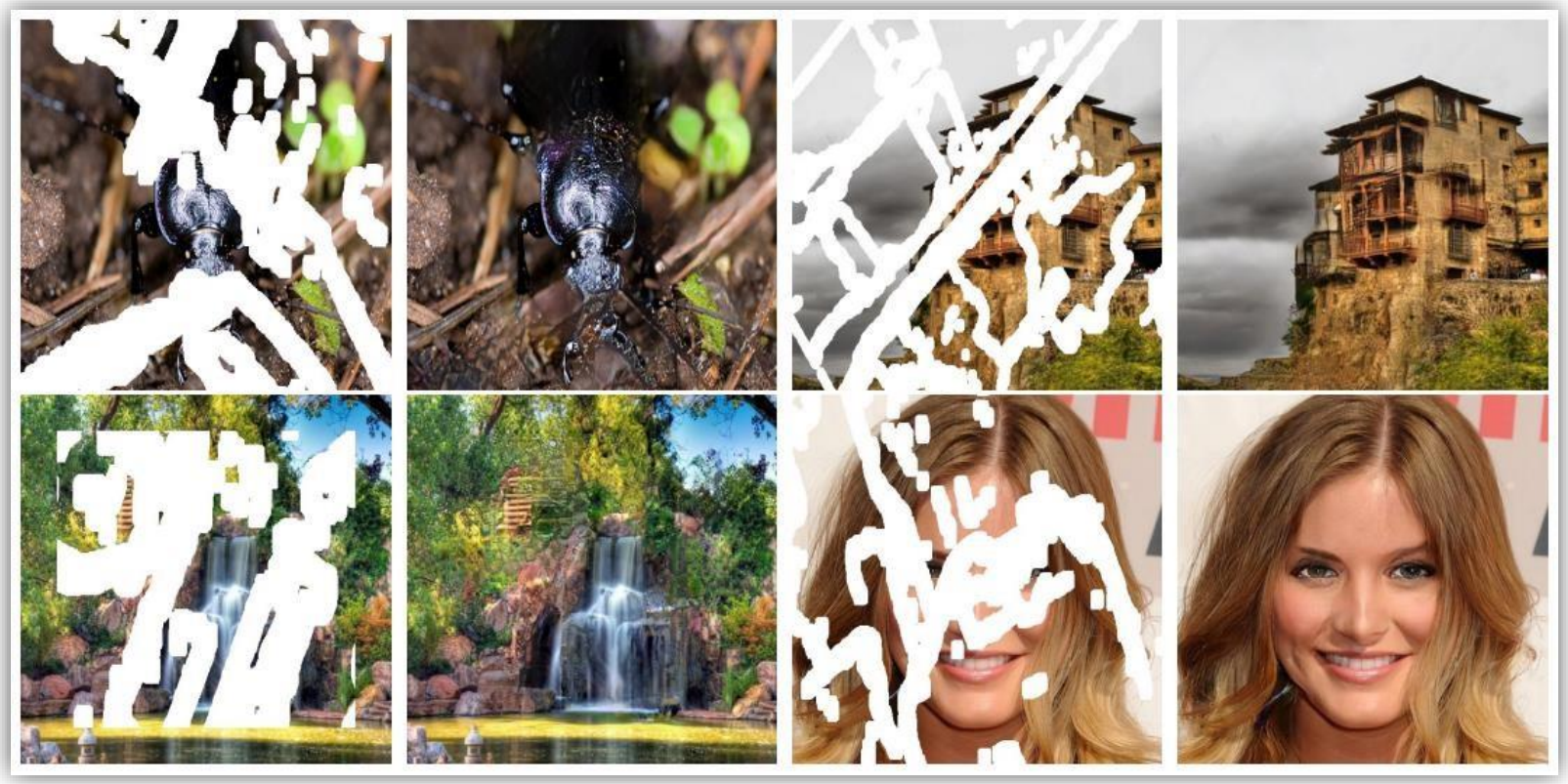


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Auto-Encoder

Utilizzi



Rif: (ArXiv)

Auto-Encoder

Riferimenti

Di seguito un riferimento utile per l'accesso ad esercitazioni e soluzioni sviluppate con il framework **PyTorch**:

GitHub - udacity: Projects and exercises

Di seguito un riferimento specifico ad esempi di *convolutional auto-encoder*:

GitHub - udacity: Convolutional Auto-Encoder

GitHub - udacity: Denoising Auto-Encoder

GitHub - udacity: Linear Auto-Encoder

Proviamo?