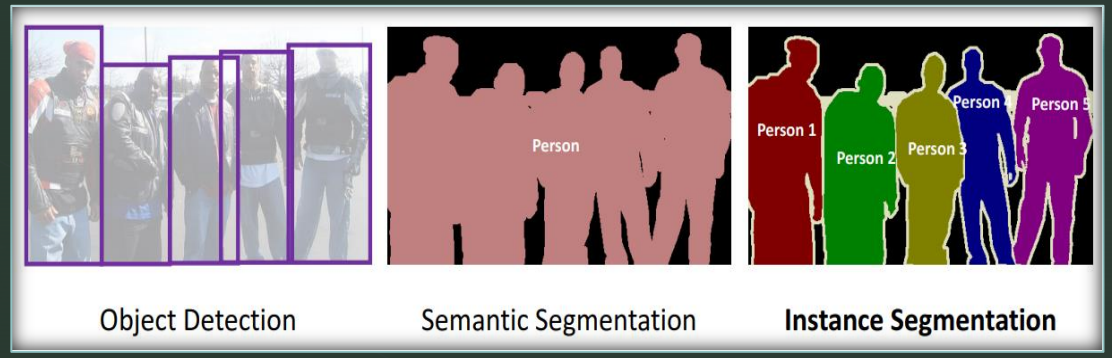
Mask R-CNN

Un’applicazione nel campo della moda

I macro compiti di una rete neurale:
Applicata al computer vision

- **Object detection**: consiste nel riconoscere gli oggetti nell'immagine
- **Semantic segmentation**: estrapola per ogni pixel, la classe a cui appartiene l'oggetto, ma non segmenta le istanze della stessa classe.
- **Instace segmentation**: Analisi dell'immagini più profonda. Distinguendo anche le istanze che appartengono alla stessa classe



Percettrone

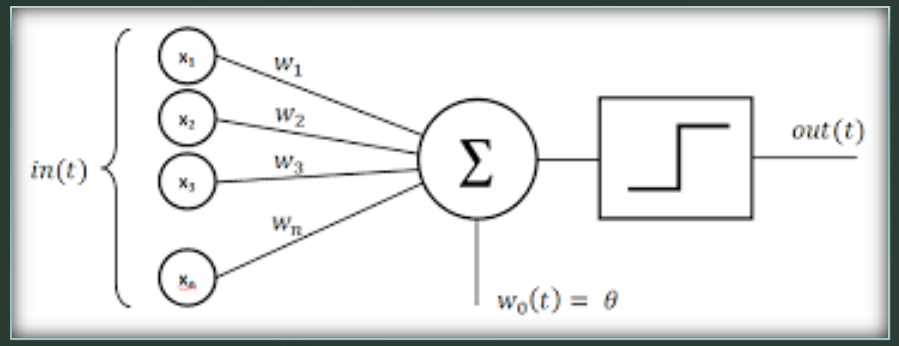
- Rappresenta l'unità funzionale di una rete neurale
- L'architettura è composta da un ingresso, un'uscita ed una regola di apprendimento.
- L'apprendimento avviene tramite esempi etichettati con la soluzione
- E' un classificatore binario:

  $F(x) = \{1\ se\ w\ \cdot x + b > 0 | \ 0\ altrimenti\}$

- L'aggiornamento dei pesi avviene tramite:

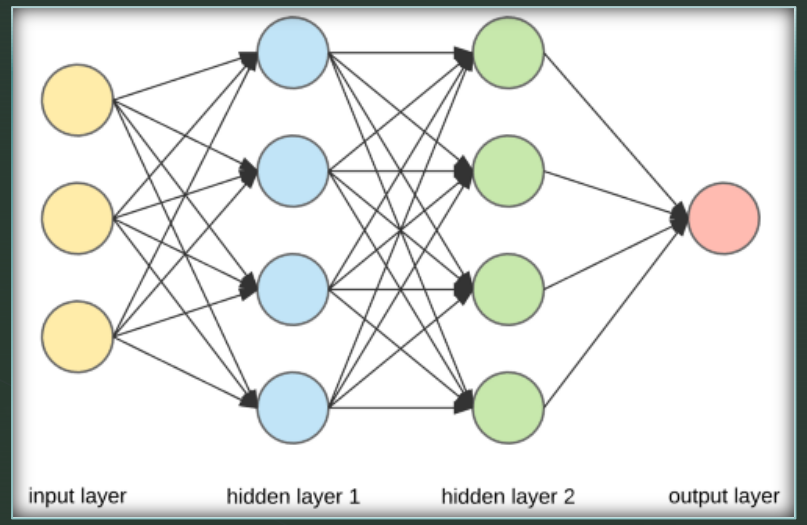
  wi(t + 1) = wi(t) + r · (dj - yj(t)) xj, i

- Se i dati non sono linearmente separabili, l'apprendimento si blocca



Multilayer perceptron

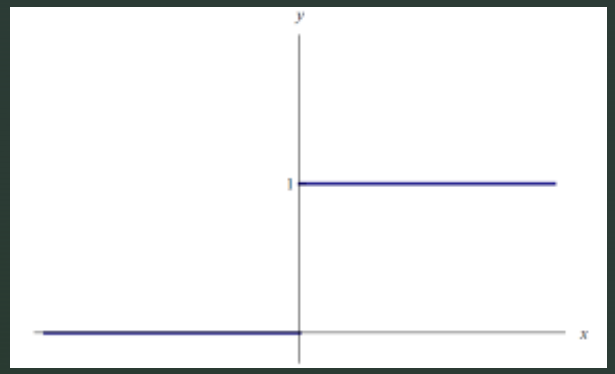
- Sono composte da più percettroni, divisi in livelli e collegati fra loro
- L'input layer riceve informazioni dall'esterno
- L'hidden layer computare le informazioni dal livello precedente
- L'output layer produce il risultato finale
- I collegamenti fra percettroni sono definiti pesi
- Utilizzano più iperpiani per delimitare le soluzioni.

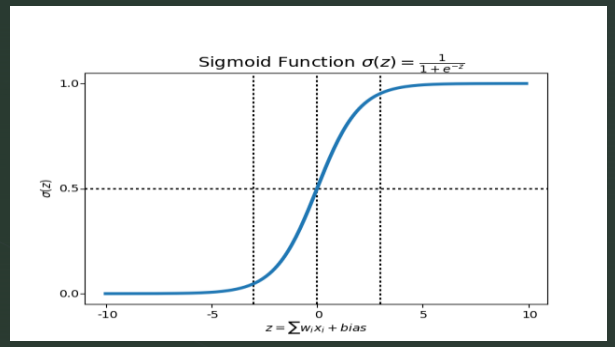


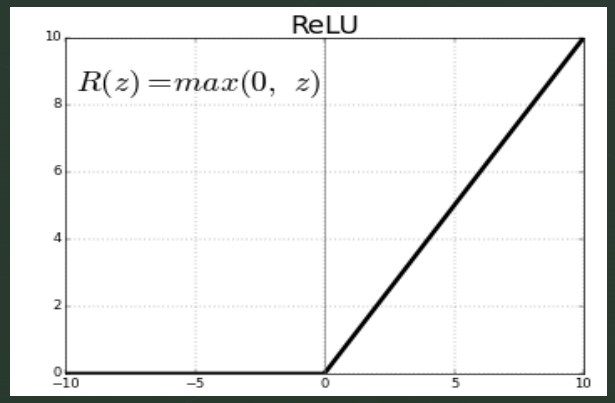
Multilayer perceptron:

Funzione d'attivazione

- descrive la correlazione tra l'input e l'output in modo non lineare
- **Funzione a gradino unitaria:** per valori negativi restituisce zero, mentre per positivi restituisce uno. Non derivabile in tutti i punti
- **Funzione sigmoide**: derivabile in tutto l'intervallo. Soffre della scomparsa del gradiente e dell'esplosione del gradiente
- **Funzione ReLU**: L'output è zero per valori minori e uguali a zero, mentre se maggiori di zero viene restituito il valore stesso. Rischio di morte dei neuroni.

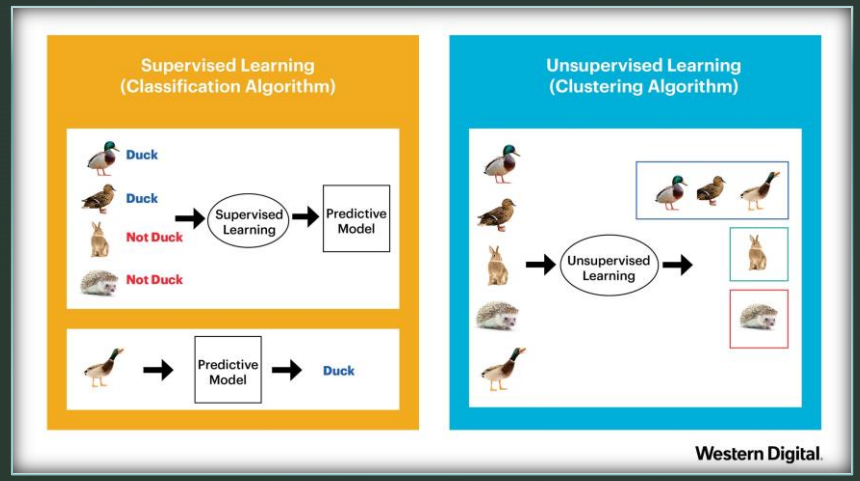


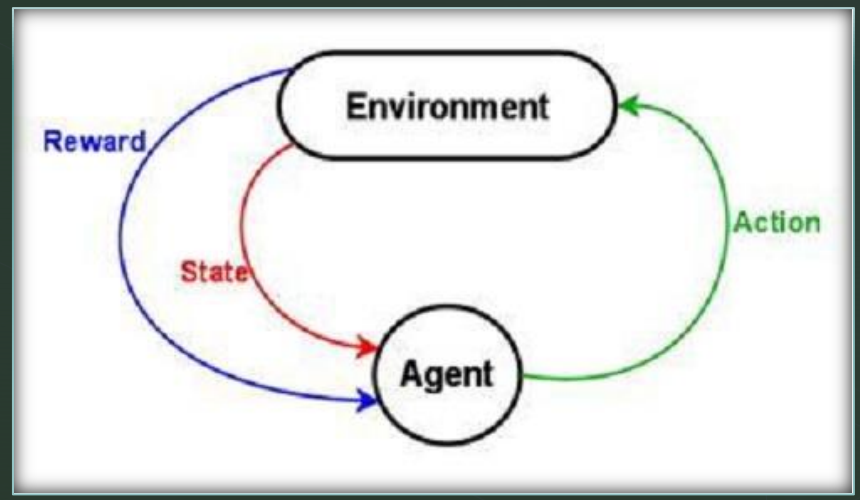




Multilayer perceptron:
Paradigmi di addestramento

- Il training di una rete neurale può essere fatto tramite l'utilizzo di tre paradigmi
- **Supervised Learning**: Ad ogni esempio in input è associato il risultato atteso.
- **Unsupervised learning**: A differenza del paradigma precedente gli esempi in input non hanno la soluzione associata.
- **Reinforcement learning**: Per raggiungere la soluzione l'agente deve determinare la sequenza d'azioni, influenzate dall'ambiente, per raggiungere una «ricompensa»





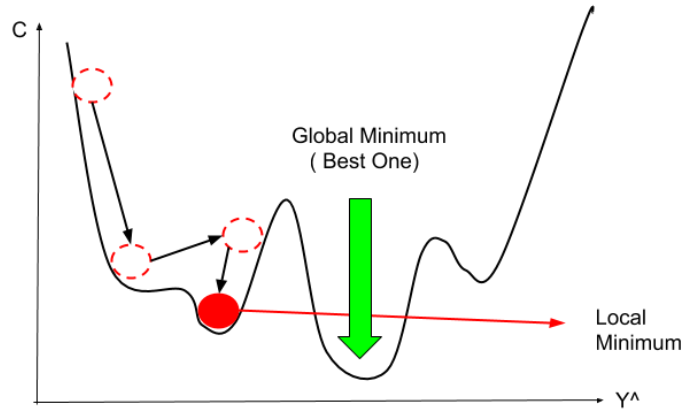
Multilayer perceptron: Discesa del gradiente

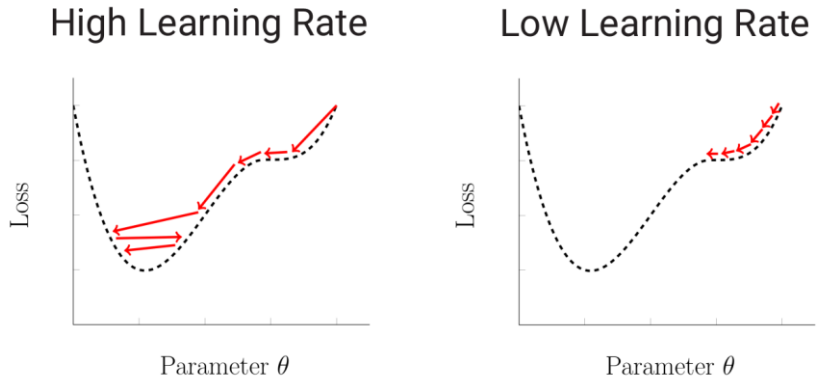
- Tecnica matematica di ricerca di un minimo locale in una funzione.
- In AI utilizzata per la ricerca di un minimo nella loss function.
- La derivata della funzione indica come devono essere modificati i pesi.
- L'aggiornamento dei pesi avviene:

$$m^1 = m^0 - X\left(y_i - (wx_i - b)\right) * a$$

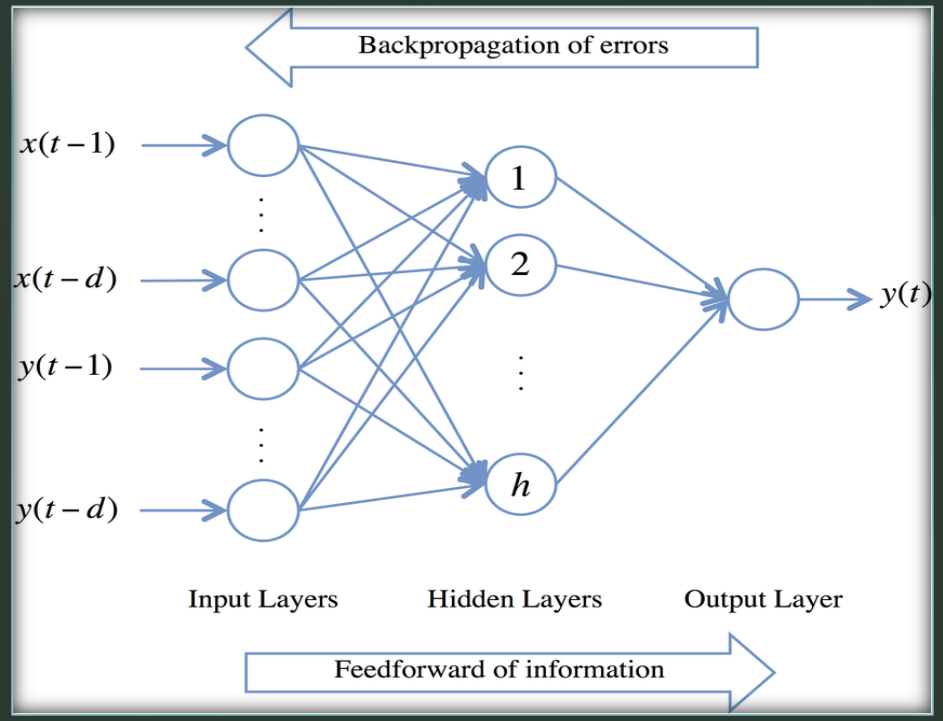
- L'aggiornamento del bias avviene:

$$b^1 = b^0 - \left(y_i - (wx_i - b)\right) * a$$





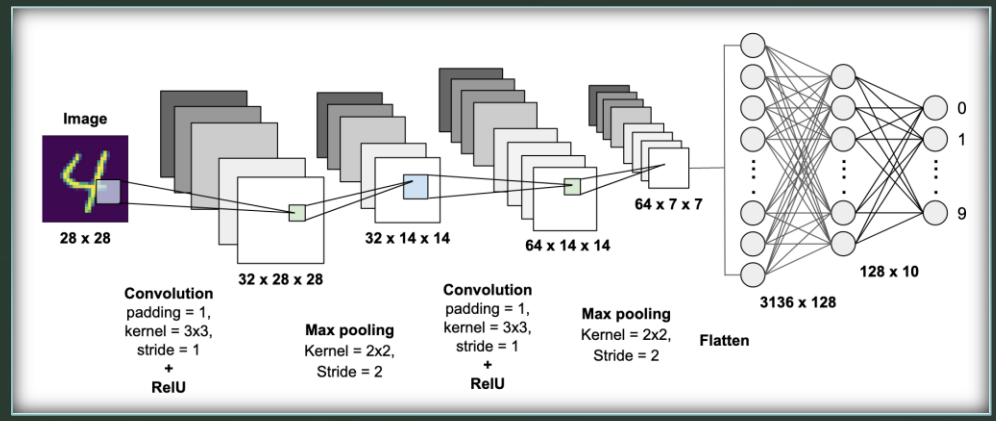
Multilayer perceptron:
Backpropagation



- Tecnica utilizzata per l'aggiornamento dei pesi
- I nuovi pesi fluiscono in direzione opposta rispetto ai dati in input
- I vantaggi sono la semplicità di implementazione e non richiede conoscenze preliminari sulla rete
- Insieme alla discesa del gradiente permettono alla rete di costruire la sua conoscenza

Reti neurali convoluzionali

- Fondamentali nella elaborazione d'immagini
- Composte da meno nodi rispetto Multilayer perceptron
- Si basano su tre caratteristiche:
  1. Local percetive field
  2. Pesi e bias condivisi
  3. Operazioni di pooling
- Le sopracitate rendono la rete invariante a traslazioni e distorsioni dell'immagini originale
- La funzione d'attivazione è di solito di tipo ReLU

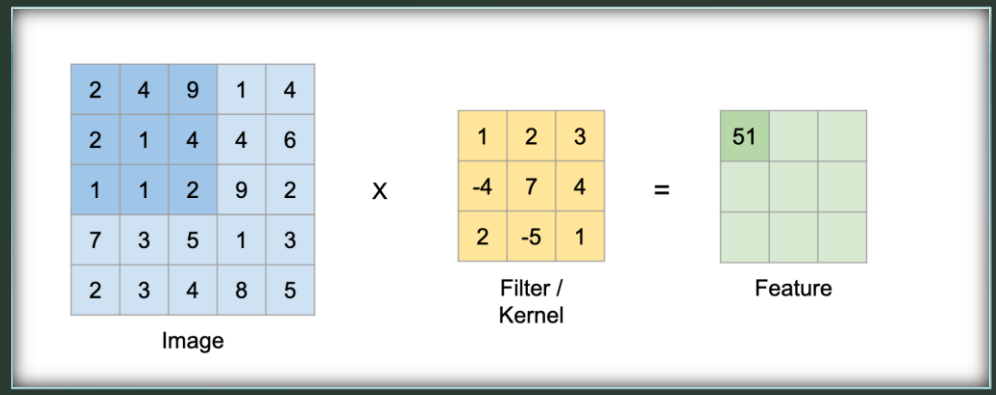


Reti neurali convoluzionali:
Operazione di convoluzione

- In campo matematico si definisce come l'integrale di due funzioni, in cui una viene traslata di un valore definito stride.
- In una ConvNet è definita come:

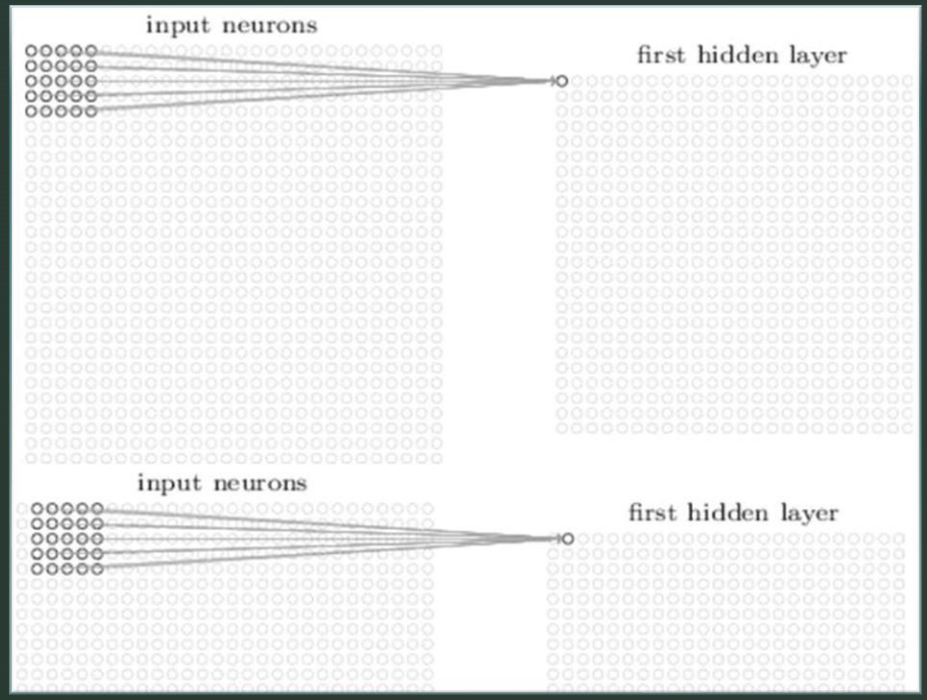
$$A * B(x,y) = \sum_{i=0}^{M} \sum_{j=0}^{N} A(x,y)B(x-i, y-j)$$

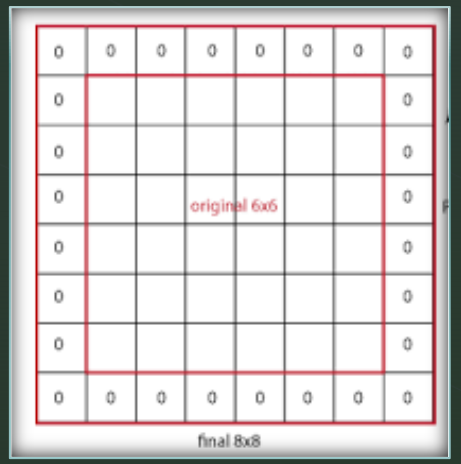
- Dove A è l'immagine in input e B è il filtro o kernel.
- I filtri permettono l'estrazione delle features dall'immagine.

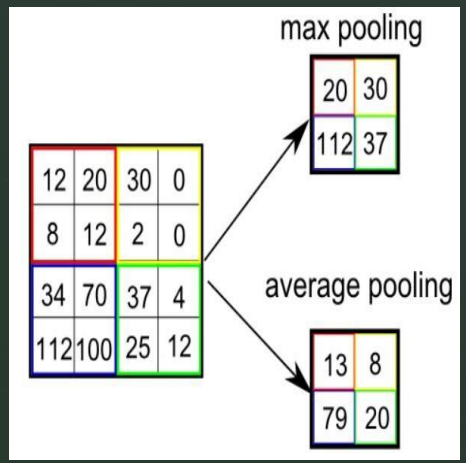


Reti neurali convoluzionali:
Architettura

- Local receptive field: situato tra l'input layer e il primo hidden layer. Riduce la dimensione dell'immagine.
- Convolutional layer: L'obiettivo è l'estrazione di features dall'immagine. Una rete CNN può avere più livelli convoluzionali.
- Pooling layer: riduce drasticamente la dimensione spaziale dell'output del livello precedente. Esistono due tecniche di pooling: max pooling e average pooling.
- Fully-Connected Layer: ha in input una matrice che trasforma in vettore, ed ogni elemento di quest'ultimo viene classificato.

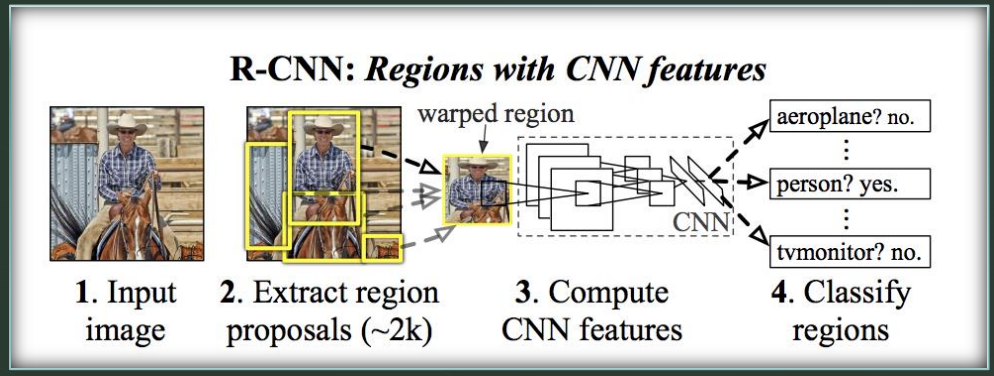






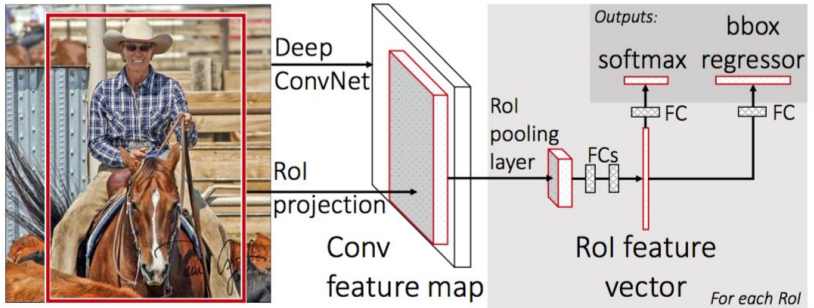
Region Based CNN:
R-CNN

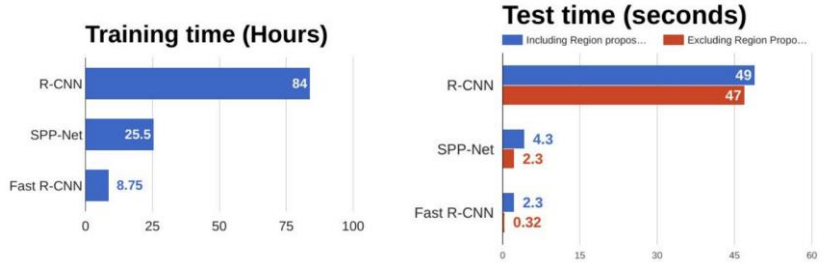
- Le CNN viste hanno dei limiti nel task di instance segmentation, siccome non conoscono a priori il numero di istanze da classificare.
- Un approccio è la ricerca bruta delle zone in cui si possono trovare le istanze.
- Girshick nel 2014 propone le Region-based Convolutional Neural Networks(R-CNN).
- Per trovare le zone d'interesse si utilizza il selective search.
- Mentre la classificazione delle aree avviene tramite il Support Vector Machine.
- Nonostante R-CNN elimini il problema, l'addestramento è costoso e lento.



Region Based CNN:
Fast R-CNN

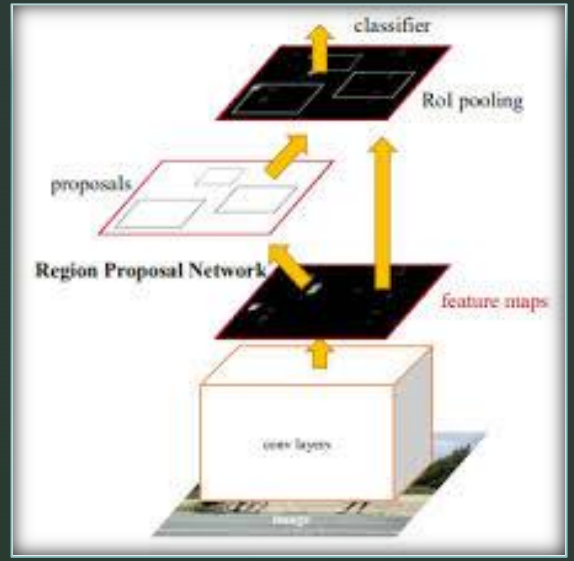
- L’immagine viene passata direttamente alla ConvNet che genera la mappa delle features.
- Dalla mappa si estraggono le RP, tramite selective search, che subiscono una trasformazione in matrice.
- La matrice viene classificata tramite un algoritmo softmax
- ha una complessità temporale minore della precedente, ma l’individuazione delle RP sono il collo di bottiglia.

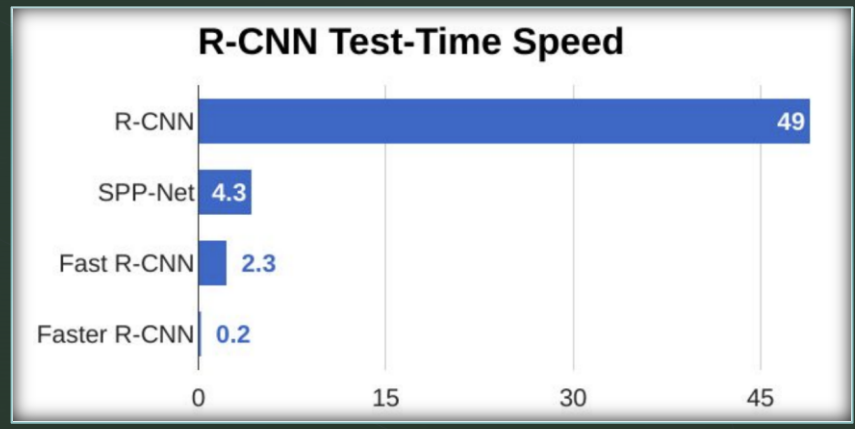




Region Based CNN:
Faster R-CNN

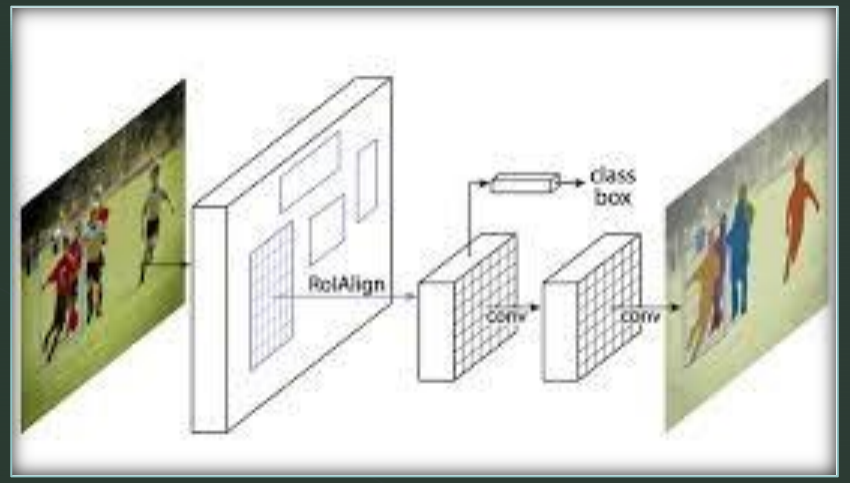
- Nel 2015 Ren propone un algoritmo di object detection in grado di insegnare alla rete come costruire le RP.
- Si utilizza un'altra rete convoluzionale, la Region Proposal Network (RPN) per prevedere le RP
- Alle box in output, vengono associate la probabilità di contenere un'istanza
- L'utilizzo di questo metodo di estrazione, migliora di molto le performance





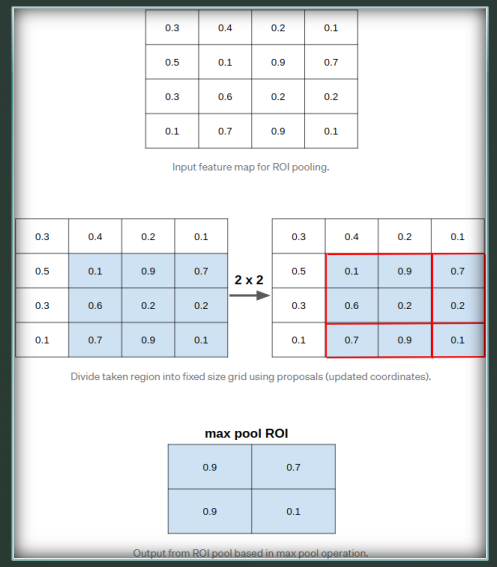
Region Based CNN:
Mask R-CNN

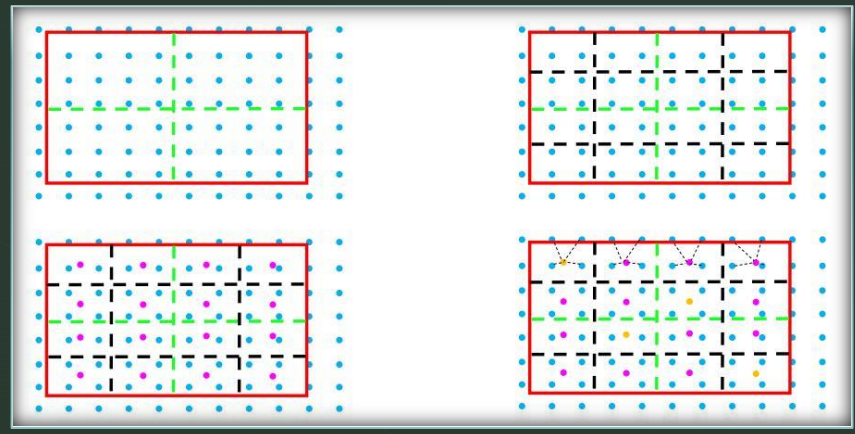
- si differenzia dalla precedente per l'aggiunta del terzo branch, con il compito di generare una maschera dell'istanza
- L'algoritmo Mask-RCNN si divide in due stage:
  1. l'individuazione delle RP
  2. classifica le istanze
- Entrambi gli stage sono collegati al backbone
- Il backbone è una parte di rete, che si occupa dell'estrazione delle features.



Region Based CNN:
Mask R-CNN Roi align | Roi pooling

- Mask R-CNN utilizza del RoI Align in sostituzione del RoI Pooling per l'estrazione delle RP
- Le coordinate output di RPN sono rispetto all'immagine originale, non possono essere usate sulla mappa delle features
- ROI pooling divide ogni coordinata per k e prende solamente la parte intera.
- Con le nuove coordinate ritaglia l'area in cui si è individuato un oggetto.
- ROI Align divide ogni coordinata per k mantenendo la parte decimale.
- Ritaglia la mappa in aree e sceglie 4 punti per ognuna.
- L'output è una matrice i cui valori sono il massimo o la media dei quattro punti di ogni zona.
- ROI Align è più precisa di ROI pooling





Region Based CNN:
Tecniche di valutazione per Mask R-CNN

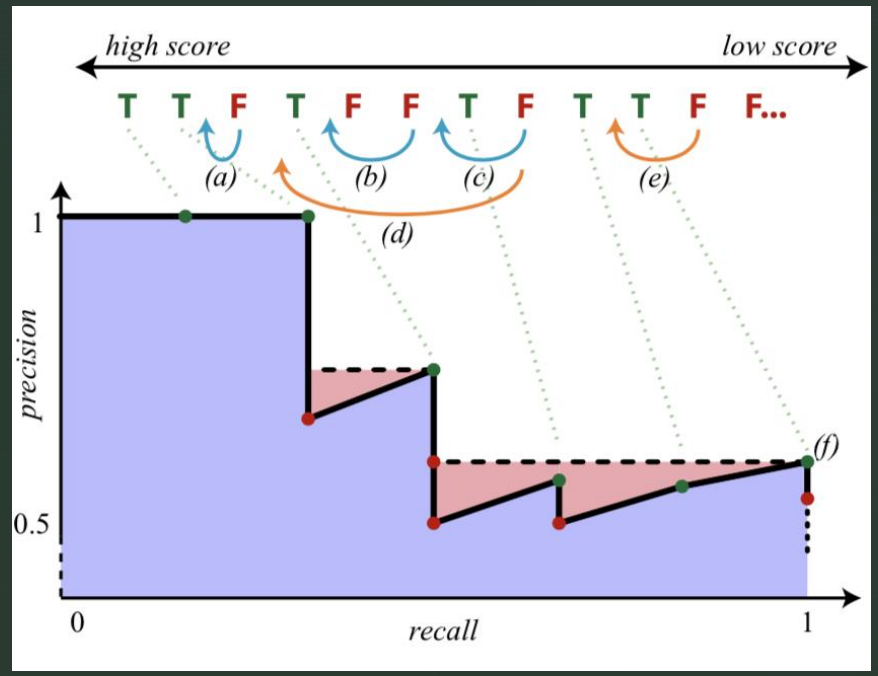
- Precision-Recall valutano la capacità di classificazione delle istanze
- Precision si calcola come:

$$TP/(TP + FP)$$

- Precision definisce la percentuale di positivi classificati correttamente dal modello
- Recall si calcola:

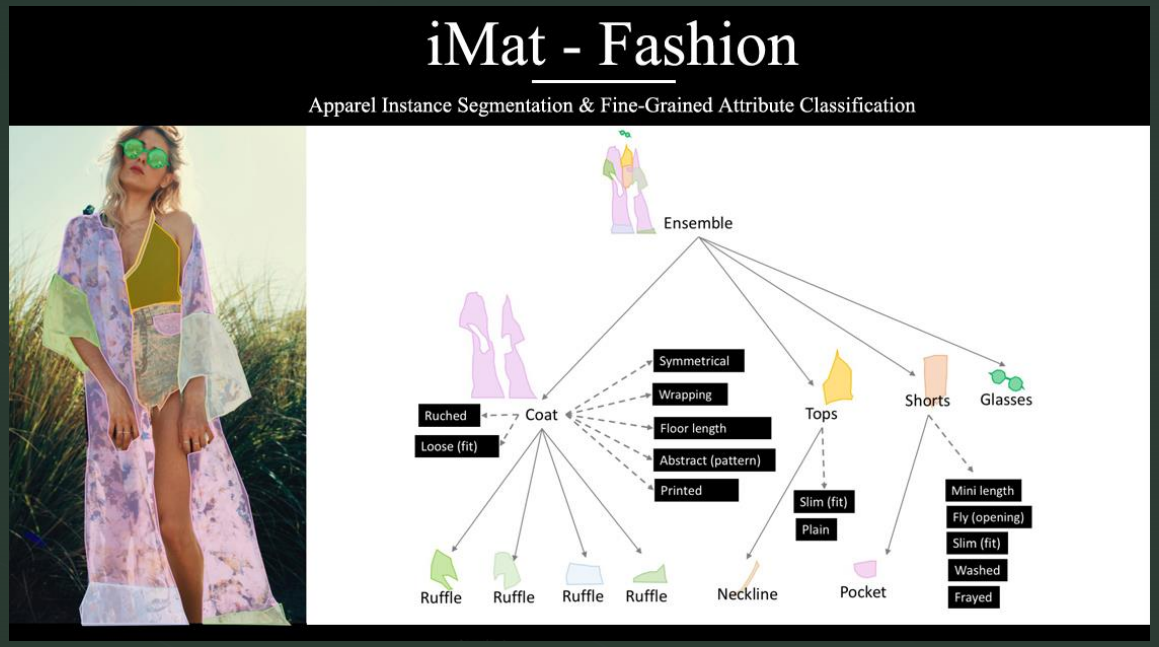
$$TP/(TP + FN)$$

- Recall restituisce, la percentuale di positivi classificati correttamente di una classe .
- Utilizzando recall come argomento della funzione Precision, si ottiene l' average precision(AP) con valore nell'intervallo [0.0, 1.0].
- La media dei valori di average precision per ogni classe è la mean average precision(mAP).



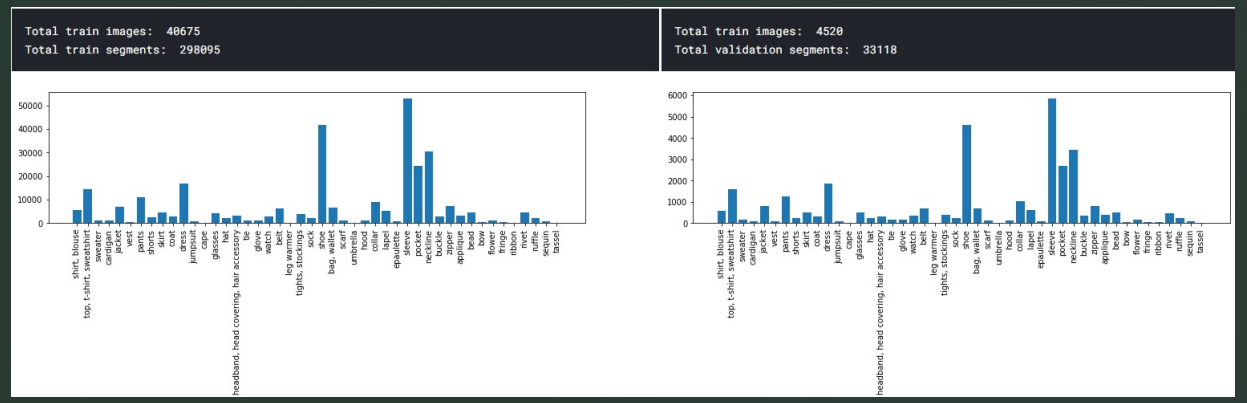
Mask R-CNN:
Un’applicazione nel campo della moda

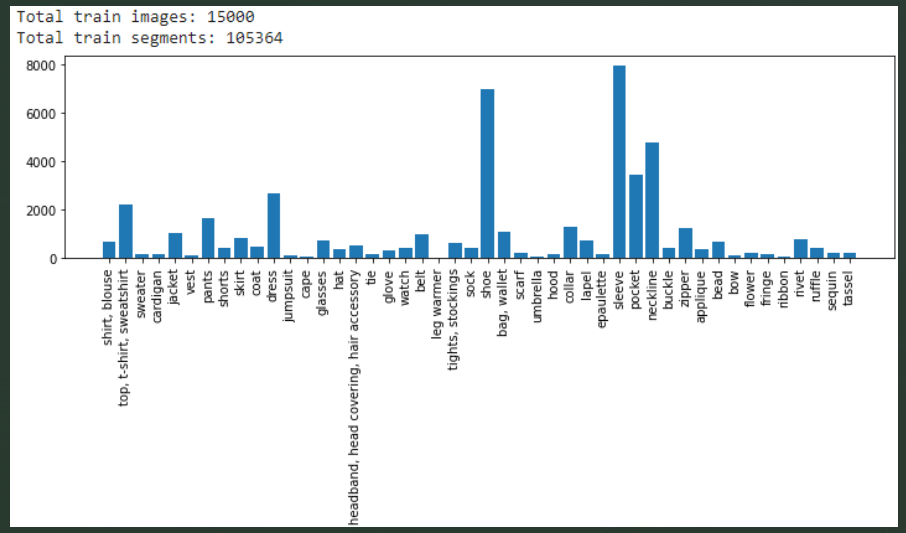
- L’obbiettivo del tirocinio è stato l’implementazione di un modello custom di Mask R-CNN
- Il compito è stato la classificazione di abiti e accessori su modelli e persone comuni
- Il task fa parte di un vecchia competizione di Kaggle
- Per il training è stato utilizzato Google colab, servizio pensato per il machine learning.
- La configurazione di colab comprendeva l’utilizzo di una **accelerazione hardware** tramite GPU, che ha portato un notevole aumento prestazionale
- Il tirocinio è stato diviso in 3 fasi:
  1. Analisi del dataset
  2. Training del modello
  3. Inferenza



Mask R-CNN:
Analisi del dataset

- Il dataset è stato fornito da Kaggle, comprendente di 40675 immagini con labels per l'addestramento e 4520 immagini
- il dataset non è omogeneo, perciò c'è il rischio di perdere alcune classi
- Le immagini contenevano più di una istanza
- Il dataset è stato ridotto a 15000 immagini, mantenendo la stessa distribuzione di classi





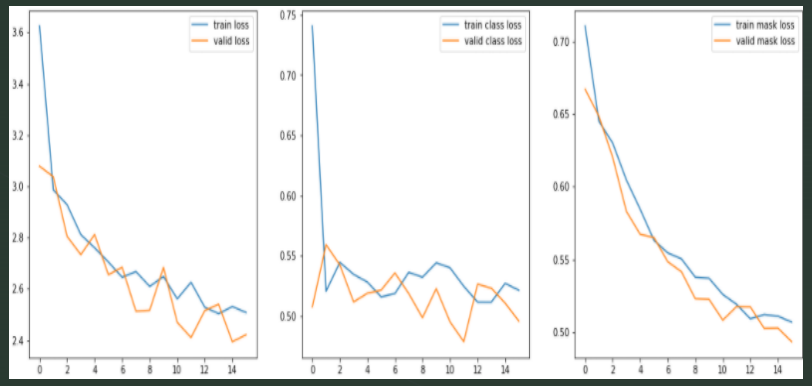
Mask R-CNN:
Training

- Il training è la parte più lunga e delicata
- L'addestramento è stato di tipo supervised
- L'obbiettivo è l'addestramento di un modello che apprenda in modo corretto le istanze
- Bisogna evitare l'overfitting
- I pesi utilizzati inizialmente sono open-source ed ottenuti lavorando sul MS COCO

Mask R-CNN: Training

- Mask R-CNN ha molti parametri, anche una piccola modifica può portare a miglioramenti o peggioramenti
- Sono stati usati due configurazioni diverse
- Nella prima configurazione l'epoca migliore è stata la 15a con una valid loss di 2.394
- La seconda configurazione è stata studiata per velocizzare il training
- Inoltre per migliorare la precision, sono state applicate 3 tecniche:
  1. Variazione del LR
  2. Variazione dei layers coinvolti
  3. L'unione del punto uno e due

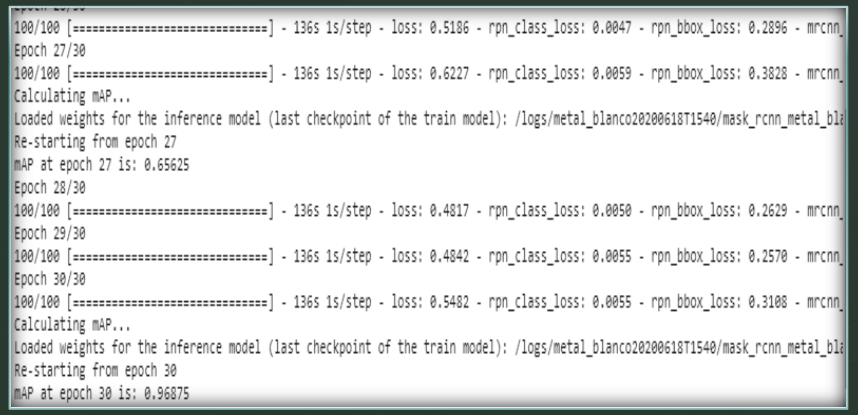
```
class FashionConfig(Config):
    NAME = "fashion"
    NUM_CLASSES = NUM_CATS + 1 # +1 for the background class

    GPU_COUNT = 1
    IMAGES_PER_GPU = 4 # a memory error occurs when IMAGES_PER_GPU is too high

    BACKBONE = 'resnet50'

    IMAGE_MIN_DIM = 1024
    IMAGE_MAX_DIM = 1024
    IMAGE_RESIZE_MODE = 'none'

    RPN_ANCHOR_SCALES = (16, 32, 64, 128, 256)
    #DETECTION_NMS_THRESHOLD = 0.0

    # STEPS_PER_EPOCH should be the number of instances
    # divided by (GPU_COUNT*IMAGES_PER_GPU), and so should VALIDATION_STEPS;
    # however, due to the time limit, I set them so that this kernel can be run in 9 hours
    STEPS_PER_EPOCH = 100
    VALIDATION_STEPS = 50
```




```
class FashionConfig(Config):
    NAME = "fashion"
    NUM_CLASSES = NUM_CATS + 1 # +1 for the background class

    GPU_COUNT = 1
    IMAGES_PER_GPU = 4 # a memory error occurs when IMAGES_PER_GPU is too high

    BACKBONE = 'resnet101'

    IMAGE_MIN_DIM = 512
    IMAGE_MAX_DIM = 512
    IMAGE_RESIZE_MODE = 'none'

    RPN_ANCHOR_SCALES = (16, 32, 64, 128, 256)
    #DETECTION_NMS_THRESHOLD = 0.0

    # STEPS_PER_EPOCH should be the number of instances
    # divided by (GPU_COUNT*IMAGES_PER_GPU), and so should VALIDATION_STEPS;
    # however, due to the time limit, I set them so that this kernel can be run in 9 hours
    STEPS_PER_EPOCH = 100
    VALIDATION_STEPS = 50
```

Mask R-CNN:
Training

- Per evitare l'overfitting e migliorare la flessibilità del modello è stata utilizzata una funzione di data Augmentation
- Le performance dei modelli sono state controllate tramite due tecniche
- K-Fold Cross-Validation
- Mean Average Precision: l'epoca migliore è stata la 30a con valore 0.97 su 1.
- L'utilizzo di queste due metriche, ha fatto in modo che le prestazioni rimanessero invariate anche durante l'inferenza

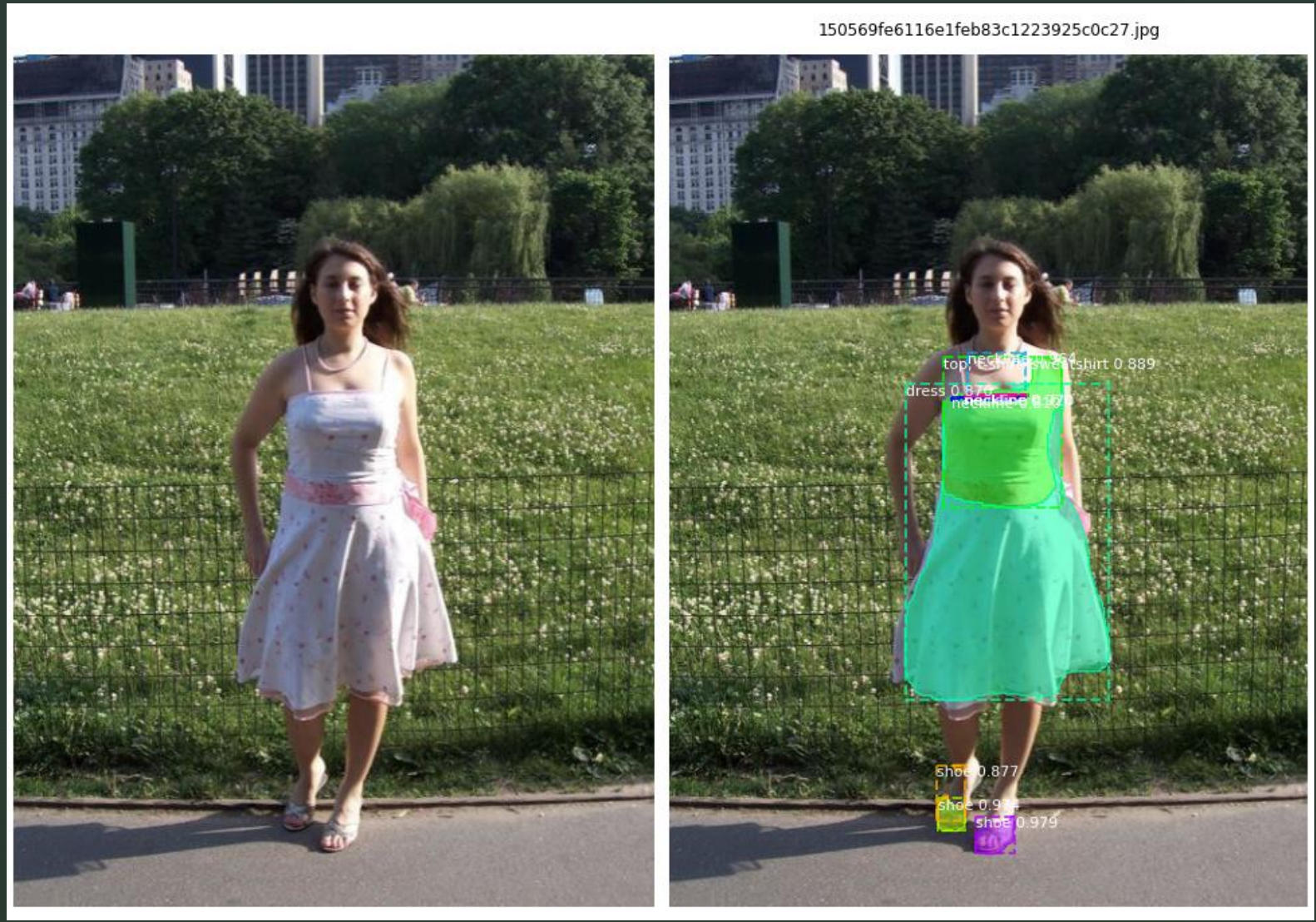


Mask R-CNN:
Inferenza

- L’ultima parte del tirocinio è stato sfruttare il modello per fare previsioni
- In input immagini senza label
- In output maschera, bounding boxes e livello di confidenza per ogni istanza

Mask R-CNN:
Inferenza

Mask R-CNN:
Inferenza



Ringraziamenti

I miei più sentiti ringraziamenti vanno al mio tutor aziendale, Giorgio Geminiani della Deepware S.R.L.

Un tutor paziente e con una grande passione per la materia.

Grazie per l’attenzione!