Presentazione DeepLearning

Giuseppe Prudente

January 2025

Indice

1 Introduzione

1.1 Descrizione del problema

La stima della profondità da immagini monoculari è un problema fondamentale nella computer vision, con applicazioni in settori come la robotica, la guida autonoma e la percezione artificiale in generale.

La stima monoculare è un compito estrememente complesso, che non può sfruttare sovrapposizioni di più immagini prese da angolazioni e punti nello spazio differenti per effettuare una stima della profondità.

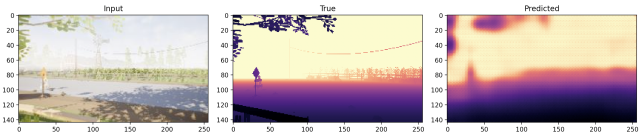
Visto che una singola immagine non può fornire informazioni dirette sulla distanza degli oggetti, è necessario sfruttare atre informazioni presenti come la prospettiva, l'occlusione, la sfocatura le ombre e la conoscenza a priori della scena proposta.

Una soluzione, che di fatto è quella che poi ci viene chiesto di implementare nel nostro elaborato è quella di utilizzare reti neurali profonde, in grado di apprendere rappresentazioni gerarchiche e astratte delle immagini, catturando strutture complesse e relazioni spaziali che un algoritmo specifico non sarebbe in grado di trovare.

1.2 Obiettivo della Challenge

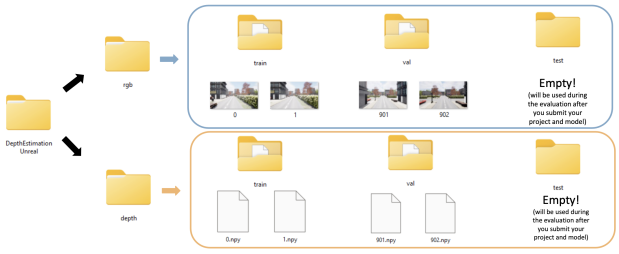
L'obiettivo di questa challenge è implementare e addestrare una rete neurale profonda capace di stimare la profondità di un'immagine a partire da un singolo input RGB. Il problema viene affrontato in un contesto supervisionato, utilizzando un dataset in cui ogni immagine a colori è associata a una mappa di profondità (ground truth), che fornisce informazioni sulla distanza relativa di ogni pixel rispetto alla fotocamera.

Di seguito un esempio diquello che ci si aspetta di ottenere.



1.3 Descrizione del Dataset

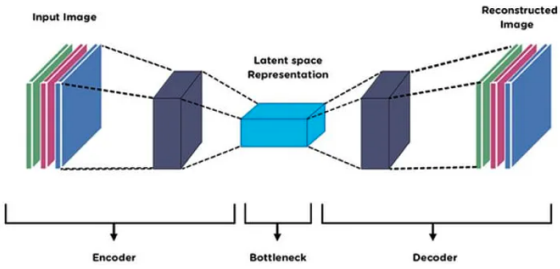
Il dataset di training è costituito da 3468 immagini a colori di dimensione 144x256 pixels associate alle rispettive mappe di profondità. Insieme al dataset di training viene fornito anche un dataset di Validation con le stesse caratteristiche.



2 Rete

2.1 Rete Utilizzata

Il modello utilizzato per la risoluzione del task sfrutta un'architettura encoder-decoder, in cui la prima parte della rete neurale esegue un sottocampionamento delle immagini in ingresso, mentre la parte di decoder le ricostruisce. Il principale vantaggio di questa architettura è una gestione più efficiente della memoria del calcolatore.

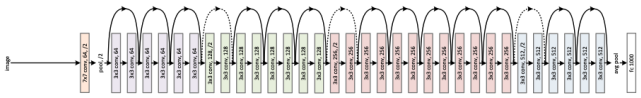


Nella fase di encoding viene utilizzata l'architettura neurale ResNet50, il cui compito principale è sottocampionare l'immagine in ingresso ed estrarre, a partire dalle feature map generate, le caratteristiche più rilevanti per la risoluzione del task. ResNet50 è stata impiegata congelando i suoi parametri, utilizzando quelli pre-addestrati sul dataset ImageNet, relativo alla challenge del 2012. Questa scelta è motivata da due ragioni principali:

1) Evitare l'addestramento da zero, riducendo significativamente il tempo necessario per l'ottimizzazione dei parametri, che altrimenti partirebbero da valori casuali.

2) Limitare il rischio di overfitting, che potrebbe verificarsi a causa del numero relativamente basso di campioni disponibili e della loro scarsa variabilità.

La backbone Resnet50 è stata privata degli ultimi 2 strati Fully-connected, i quali non servivano per il task richiesto.
In questo modo la rete neurale prende in entrata un ensamble di immagini della dimensione (batch-size , 3, 144, 256) e restituisce una insieme di feature-maps della dimensione (batch-size, 2048, 8, 5).



La parte di decoding prende in ingresso questo insieme di tensori e mediante layer convoluzionali ricostruisce la matrice delle stime di profondità, quest'ultlima avrà la dimensione (batch-size, 1, 144, 256).

L'upsampling viene effettuato tramite cinque layer di convoluzione trasposta, in cui, a ogni passaggio, il numero di feature map viene progressivamente dimezzato, fino a raggiungere l'ultima convoluzione trasposta, dove le 64 feature map collassano in una sola. Subito dopo l'ultima convoluzione trasposta, è stato necessario inserire un layer di MaxPooling per riportare le dimensioni dell'immagine alla risoluzione originale. Ogni layer di convoluzione trasposta è seguito da un layer convoluzionale che mantiene invariata sia la dimensione delle feature map sia il numero di canali. Questo accorgimento consente alla rete di consolidare le informazioni apprese durante la fase di upsampling, migliorando così la robustezza dell decoder.

Si noti, infine, che ogni layer ad eccezione dell'ultimo presenta una batch normalzation prima dell'applicazione della funzione di attivazione Relu.

2.1.1 Hyperparameters

-Learning Rate: 0.001
-Batch size: 50
-Epochs: 25
-Funzione di Attivazione: Relu
-Ottimizzatore: Adam

3 Risultati

3.1 Metriche

Per l'addestramento del modello, è stata utilizzata una funzione di perdita composta da una combinazione di Mean Squared Error (MSE) e Structural Similarity Index Measure (SSIM). Questa scelta consente di bilanciare due aspetti fondamentali della stima della profondità: la precisione numerica e la qualità strutturale.

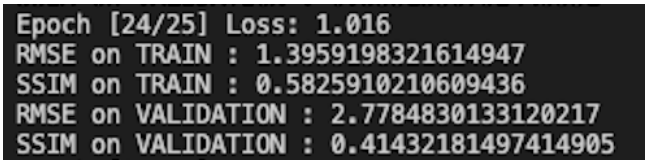
- l'MSE misura la differneza quadratica media tra la mappa di profondità stimata e la ground truth. La pecca di questa metrica è il fatto che implementa un confornto pixel a pixel, tralasciando invece tutto quello che riguarda le relazioni spaziali tra pixel e ogni forma di visione di contesto.

- SSIM permette di valutare la similarità tra immagini, concentrandosi maggiormente su aspetti di contesto ed evitando un confronto pixel per pixel.

Combinare queste due metriche nella funzione di perdita permette di sfruttare i vantaggi di entrambe: il MSE assicura una precisione numerica elevata, mentre l'SSIM preserva le informazioni strutturali della scena.

3.2 Risultati

La figura mostra i risultati ottenuti durante l'addestramento, vengono iìriportati i valori di RMSE e SSIM sia sul dataset di training che di validation.



In fase di test il modello ha ottenuto le seguenti prestazioni.

- RMSE TEST: 3.064665138721466
- SSIM TEST: 0.46462246775627136

3.3 conclusioni

Basandoci sui risultati di SSIM ottenuti in fase di Training e Test, emerge la seguente considerazione:

La presenza di un evidente overfitting del modello: infatti la performance sul set di test si discosta di 19 punti percentuali rispetto a quella sul set di training. Questo indica che il modello apprende i dati di training, ma fatica a generalizzare, ottenedo performaces peggiori su dati mai visti prima.

per ottemperare a questo overfitting si potrebbero adottare diverse strategie. Innanzitutto, ridurre la batch size (ad esempio a 16) potrebbe favorire una migliore generalizzazione introducendo più rumore nel training. Parallelamente, sarebbe utile adattare il learning rate in base alla batch size, seguendo una strategia di scaling lineare per mantenere la stabilità della convergenza. Inoltre, tecniche di regularizzazione, come l'uso weight decay, potrebbero limitare la capacità del modello di memorizzare i dati di training, migliorando così la sua capacità di generalizzazione. Infine, l'utilizzo di Data Augmentation potrebbe aumentare la variabilità dei dati di input, rendendo il modello più robusto. Future iterazioni del progetto potrebbero includere test su diverse combinazioni di queste tecniche per identificare la configurazione ottimale.