Monocular Depth Estimation with ResNet-34 and a Custom Upsampling Decoder

Dr. Lopopolo
Department of Computer and Robotics Engineering
University of Perugia
Email: drlopopolo@unipg.it

***Abstract*—La stima della profondità monoculare (MDE) è un'attività fondamentale nella computer vision, utilizzata in applicazioni che vanno dalla navigazione autonoma alla robotica. Questo lavoro propone un modello MDE supervisionato che combina un encoder ResNet-34 pre-addestrato con un decoder upsampling personalizzato per generare DepthMaps da immagini RGB. I contributi principali includono il congelamento dei primi strati dell'encoder per la regolarizzazione e l'efficienza e l'utilizzo di una funzione di perdita combinata basata sulle metriche RMSE e SSIM. Gli esperimenti, condotti su un set di dati generato con il simulatore grafico Unreal Engine, evidenziano la capacità del modello di stimare le DepthMaps. Lo studio dimostra il potenziale del deep learning nell'affrontare sfide tradizionalmente risolte da tecniche geometriche.**

I. Introduction

Questo progetto è stato sviluppato nell'ambito di una sfida del corso di Deep Learning and Robot Perception della laurea magistrale in Ingegneria Informatica e Robotica dell'Università di Perugia. La sfida si è concentrata sulla stima monoculare della profondità utilizzando tecniche di deep learning supervisionate, con il dataset fornito dal professore del corso.

La stima monoculare della profondità (MDE) cerca di estrarre informazioni sulla profondità da una singola immagine RGB, un compito che pone sfide significative a causa dell'assenza di spunti stereoscopici o temporali. A differenza dei metodi stereoscopici o multivisione, la MDE deve basarsi esclusivamente su modelli spaziali e spunti contestuali all'interno di un singolo fotogramma, il che la rende intrinsecamente ambigua. Nonostante i progressi compiuti con il deep learning, ottenere previsioni accurate della profondità garantendo al contempo l'efficienza computazionale rimane una sfida non banale.

L'obiettivo principale di questo lavoro è stimare la profondità di ogni pixel di un'immagine RGB in condizioni di supervisione, producendo una DepthMap in cui ogni pixel codifica la distanza dall'oggetto corrispondente in metri. Per raggiungere questo obiettivo, proponiamo un approccio supervisionato che integra un codificatore ResNet-34 con un decodificatore upsampling personalizzato, progettato per migliorare la risoluzione spaziale delle DepthMap. Ispirato alla percezione umana della profondità, che si basa sulla comprensione del contesto, delle proporzioni degli oggetti e delle ombre, questo lavoro mira a esplorare il potenziale delle reti neurali profonde per emulare capacità simili.

II. Related Work

I recenti progressi nella stima monoculare della profondità (MDE) possono essere ampiamente classificati in approcci supervisionati e auto-supervisionati. I metodi supervisionati, come quelli introdotti da Eigen et al. [11] e Laina et al. [12], raggiungono un'elevata accuratezza sfruttando la verità di terra delle DepthMaps durante l'addestramento. Questi approcci eccellono in ambienti controllati, ma richiedono ampi set di dati etichettati, che sono costosi e lunghi da produrre. D'altra parte, i metodi auto-supervisionati, come quelli proposti da Godard et al. [**?**] e Zhou et al. [14], affrontano questa limitazione sfruttando la coerenza fotometrica tra fotogrammi video consecutivi. Sebbene questo riduca la dipendenza dai dati etichettati, spesso compromette l'accuratezza, in particolare nelle regioni con oggetti dinamici o texture povere.

Gli approcci geometrici tradizionali alla stima della profondità, come la corrispondenza stereo o la ricostruzione multivista, si basano su più immagini o telecamere per risolvere le ambiguità della profondità. Questi metodi utilizzano le corrispondenze tra le diverse viste per ricostruire le scene 3D. Tuttavia, nelle configurazioni monoculari, la perdita intrinseca di informazioni dimensionali durante la proiezione 2D rende queste tecniche inapplicabili senza ulteriori vincoli.

È interessante notare che gli esseri umani possono dedurre la profondità da un singolo punto di vista basandosi su indicazioni contestuali, come le proporzioni degli oggetti, le occlusioni e le ombre, nonché sull'esperienza precedente. Questa osservazione motiva lo sviluppo di approcci basati sull'apprendimento profondo che mirano a emulare la percezione della profondità simile a quella umana. Apprendendo da grandi insiemi di dati, le reti neurali profonde possono catturare le intricate relazioni tra le caratteristiche dell'immagine e la profondità, consentendo la stima di DepthMaps direttamente da singole immagini RGB.

Ispirandosi a ciò, l'approccio proposto impiega un codificatore ResNet-34 e un decodificatore upsampling personalizzato per affrontare la MDE. Il metodo cerca di bilanciare l'efficienza computazionale con previsioni accurate della profondità, sfruttando la potenza dell'apprendimento supervisionato per disambiguare le informazioni sulla profondità in una singola immagine RGB.

III. METHODOLOGY

A. Model Architecture

1) Encoder Architecture: In questo lavoro, utilizziamo un approccio di messa a punto parziale per l'encoder ResNet-34, pre-addestrato su ImageNet [1]. ResNet-34 è stato scelto come encoder per la sua capacità di estrarre caratteristiche gerarchiche dell'immagine attraverso la sua architettura a blocchi residui [10]. Pre-allenato su ImageNet, fornisce un robusto estrattore di feature per insiemi di dati di dimensioni moderate, raggiungendo un equilibrio tra complessità e accuratezza. I primi strati convoluzionali, responsabili dell'estrazione di feature generali come bordi e texture, sono congelati per mantenere i loro pesi pre-addestrati. Questa strategia è in linea con i risultati di Yosinski et al. [2], che evidenziano l'utilità di congelare i primi strati per preservare le caratteristiche generalizzabili.

Gli strati fully-connected di ResNet-34 sono stati rimossi, in quanto progettati per compiti di classificazione con 1000 classi in ImageNet. Per la stima della profondità, che richiede output strutturati spazialmente, questi strati sono stati sostituiti da un decoder personalizzato che consente un compito di regressione. Questo approccio è in linea con Razavian et al. citerazavian2014cnn e Laina et al. [12], che sostengono la necessità di modifiche specifiche per adattare i modelli pre-addestrati a compiti non di classificazione. Questo approccio bilancia l'efficienza computazionale e l'adattamento specifico al compito, rendendolo adatto a insiemi di dati di dimensioni moderate, come sostengono Simonyan e Zisserman [4].

2) Decoder Architecture: Il decoder è progettato per aumentare progressivamente il campionamento delle mappe di caratteristiche estratte dall'encoder ResNet-34, ricostruendo una DepthMap densa alla risoluzione di 224×224. Impiega convoluzioni trasposte (`ConvTranspose2d`) per aumentare la risoluzione spaziale, con kernel di dimensione 4×4, stride 2 e padding 1. Questo design garantisce una transizione graduale dall'encoder ResNet-34 alla mappa delle caratteristiche estratte. Questo design garantisce una transizione graduale dalle feature map a bassa risoluzione (7×7) alla DepthMap ad alta risoluzione, che corrisponde alle dimensioni dell'immagine di ingresso.

B. Transposed Convolution Dimensions

The output dimensions of a transposed convolution layer are calculated as:

$$\begin{aligned} H_{\text{out}} &= (H_{\text{in}} - 1) \cdot \text{stride} - 2 \cdot \text{padding} \\ &\quad + \text{kernel\_size} + \text{output\_padding}, \\ W_{\text{out}} &= (W_{\text{in}} - 1) \cdot \text{stride} - 2 \cdot \text{padding} \\ &\quad + \text{kernel\_size} + \text{output\_padding}. \end{aligned}$$

La normalizzazione dei batch è una tecnica introdotta per risolvere il problema dello spostamento interno della covariante durante l'addestramento delle reti neurali profonde. Questa tecnica normalizza le attivazioni di uno strato per mini-batch, migliorando la stabilità e la velocità di convergenza dell'addestramento. Agisce anche come regolarizzatore, eliminando in alcuni casi la necessità di Dropout.

La funzione di attivazione utilizzata è ReLU [7] per stabilizzare l'addestramento e introdurre delle non linearità, che migliorano la capacità del modello di apprendere relazioni spaziali complesse.

Le convoluzioni intermedie (1 volte 1) agiscono come trasformazioni di canale, combinando le informazioni tra le feature map senza alterare la risoluzione spaziale, in linea con le strategie utilizzate nelle moderne architetture [12].

Il decoder riduce progressivamente il numero di canali (512 $\rightarrow$ 256 $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 1) aumentando al contempo la risoluzione spaziale, ottenendo un efficiente compromesso tra calcolo e accuratezza. Questa architettura è stata scelta rispetto a progetti più complessi (ad esempio, connessioni saltate di tipo U-Net) per privilegiare l'efficienza computazionale e la semplicità, pur mantenendo una forte capacità di ricostruzione.

TABLE I
EVOLUTION OF FEATURE MAP DIMENSIONS THROUGH THE DECODER

Layer	Feature Map Dimensions	Number of Channels
Input	7×7	512
Transposed Conv 1	14×14	256
Transposed Conv 2	28×28	128
Transposed Conv 3	56×56	64
Transposed Conv 4	112×112	32
Output	224×224	1

1) Comparison with Alternative Choices: Mentre ResNet-50 o ResNet-101 avrebbero potuto essere utilizzati come encoder, la loro maggiore complessità e il rischio più elevato di overfitting hanno reso ResNet-34 una scelta più adatta per un dataset di dimensioni moderate come DepthEstimationUnreal. Inoltre, le architetture DenseNet o EfficientNet, note per la loro efficienza, potrebbero essere esplorate in futuro per migliorare le prestazioni del modello.

Per quanto riguarda il decoder, l'inclusione di connessioni di salto, come visto in U-Net [5], potrebbe migliorare ulteriormente la conservazione dei dettagli fini. Tuttavia, il costo computazionale aggiuntivo associato a tali architetture è stato evitato in questa implementazione per mantenere l'efficienza.

C. Dataset and Preprocessing

Il dataset DepthEstimationUnreal, fornito dal professore del corso, è stato generato utilizzando Unreal Engine, un simulatore grafico in grado di produrre DepthMaps di verità a terra da immagini RGB. Il set di dati comprende migliaia di immagini RGB e mappe di profondità accoppiate, suddivise in sottoinsiemi di allenamento e di convalida. Il test viene utilizzato dal professore del corso per verificare il modello. L'input è costituito da immagini RGB, memorizzate in formato .jpeg e normalizzate all'intervallo [0, 1]. Mentre le DepthMaps, che sono il target associato, sono memorizzate in formato .npy e sono ritagliate nell'intervallo [0.0, 20.0]. In questa codifica, i valori di intensità più bassi (nero, blu, viola) indicano gli oggetti più vicini, mentre i valori di intensità più

alti (giallo, arancione) rappresentano gli oggetti più lontani dalla telecamera.

Poiché ResNet34 è stata sviluppata per ricevere immagini in ingresso con una risoluzione di 224 volte 224, è stato necessario preprocessare sia i dataset di addestramento che quelli di validazione (nonché il dataset di test utilizzato successivamente dal professore del corso). Sono state applicate due trasformazioni per convertire la risoluzione dell'input RGB e dell'obiettivo DepthMap da 144×256 a 224×224:

1) **Resize**$(224, 224)$: This transformation scales the image to the specified dimensions, in this case 224×224.
2) **CenterCrop**$(224, 224)$: This transformation crops a central area of the image to the specified size, ensuring that the central part of the resized image remains meaningful.

Table III lists the dataset preprocessing steps.

TABLE II
PREPROCESSING PARAMETERS

Parameter	Value
Depth Clipping	[0.0, 20.0]
Normalization	RGB: [0, 1]
Transformations	Resize, CenterCrop
Image Resize	224x224

TABLE III
PREPROCESSING PARAMETERS

Parameter	Value
Image Resize	224x224
Depth Clipping	[0.0, 20.0]
Normalization	RGB: [0, 1]
Transformations	Resize, CenterCrop

D. Loss Function

La funzione di perdita utilizzata in questo lavoro combina due componenti complementari: RMSE (Root Mean Squared Error) e SSIM (Structural Similarity Index). Questa combinazione bilancia l'accuratezza globale con la coerenza strutturale delle DepthMaps previste.

La funzione di perdita combinata è definita come:

$$\mathcal{L} = \alpha \cdot \text{RMSE} + \beta \cdot (1 - \text{SSIM}), \tag{1}$$

dove:

- e β sono fattori di ponderazione, entrambi impostati a 1,0 in questo lavoro per dare la stessa importanza a entrambi i termini, voce RMSE misura la differenza numerica complessiva tra le profondità previste e quelle della verità a terra,
- $1 -$ SSIM misura la dissimilarità strutturale tra le DepthMaps previste e quelle reali.

La perdita RMSE è calcolata come:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{2}$$

dove N è il numero di pixel, $\hat{y}_i$ è il valore di profondità previsto per il i-esimo pixel e y_i è il valore di verità a terra. L'RMSE si concentra sulla minimizzazione degli errori globali nell'intera DepthMap.

La perdita basata su SSIM è calcolata come:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{3}$$

dove μ_x, μ_y sono le medie, σ_x^2, σ_y^2 sono le varianze e σ_{xy} è la covarianza tra la DepthMap prevista (x) e la verità a terra (y). Le costanti C_1 e C_2 stabilizzano il calcolo. Il termine di perdita è calcolato come:

$$\text{SSIM Loss} = 1 - \text{SSIM}(x, y),$$

penalizzando le dissomiglianze strutturali tra le DepthMaps previste e quelle reali.

La perdita combinata è implementata come: $\text{ssim}_loss = 1 - ssim(output, depth) rmse_loss = torch.sqrt(F.mse_loss(output, depth)) loss = alpha * rmse_loss + beta * ssim_loss$

dove `output` è la DepthMap prevista e `depth` è la DepthMap di verità.

La perdita combinata sfrutta l'RMSE per minimizzare gli errori assoluti di profondità, garantendo l'accuratezza numerica, mentre la componente SSIM si concentra sulla conservazione delle somiglianze percettive e strutturali. Questo approccio è particolarmente efficace in compiti come la stima della profondità, in cui sia la coerenza globale che la struttura locale sono fondamentali [13].

E. Optimizer and Learning Rate Scheduler

Per addestrare il modello proposto è stato utilizzato l'ottimizzatore Adam [8]. Adam combina i vantaggi di entrambi gli algoritmi AdaGrad e RMSProp, regolando il tasso di apprendimento per ogni parametro in modo dinamico sulla base delle stime dei primi e secondi momenti dei gradienti. Questo lo rende particolarmente adatto a problemi con gradienti radi o obiettivi rumorosi, come la stima della profondità. L'ottimizzatore è configurato con i seguenti parametri:

- **Learning Rate:** $\texttt{lr} = 0.001$
- **Weight Decay:** 1×10^{-5}

a) Learning Rate Scheduler: Per migliorare la convergenza e la generalizzazione, è stato utilizzato uno scheduler ciclico del tasso di apprendimento (CyclicLR) [9]. CyclicLR varia periodicamente il tasso di apprendimento tra un valore minimo ($\texttt{base\_lr} = 0,001$) e massimo ($\texttt{max\_lr} = 0,005$) nel corso dell'allenamento. Il programma segue una politica triangolare2 , in cui il tasso di apprendimento oscilla in un modello triangolare con un'ampiezza decrescente dopo ogni ciclo. Questo aiuta il modello a sfuggire ai minimi locali e migliora la stabilità dell'addestramento.

Lo scheduler è stato configurato con i seguenti parametri:

- **Base Learning Rate (`base_lr`):** 0.001
- **Maximum Learning Rate (`max_lr`):** 0.005
- **Step Size Up:** 1000 iterations

- **Step Size Down:** 2000 iterations
- **Mode:** triangular2

L'ottimizzatore Adam è stato scelto per la sua capacità di gestire gradienti rumorosi e aggiustamenti dinamici del tasso di apprendimento, garantendo una formazione robusta anche in presenza di obiettivi non stazionari.

Lo scheduler CyclicLR completa Adam introducendo variazioni periodiche nel tasso di apprendimento, che incoraggiano il modello a esplorare uno spazio di soluzioni più ampio, evitando potenzialmente l'overfitting e migliorando la generalizzazione [9].

F. Execution Pipeline

Lo script 'main.py' definisce i parametri chiave (Tabella IV) e orchestra i processi di addestramento e valutazione.

TABLE IV
HYPERPARAMETERS

Parameter	Value
Learning Rate	0.001
Batch Size	32
Max Epochs	100
Eval Frequency	Every 2 epochs
Visualization	Every 100 steps

IV. EXPERIMENTS

A. Metrics

Le prestazioni vengono valutate utilizzando RMSE e SSIM. L'analisi qualitativa comprende la visualizzazione delle DepthMaps con la funzione 'visualize_img' (Fig. 1).

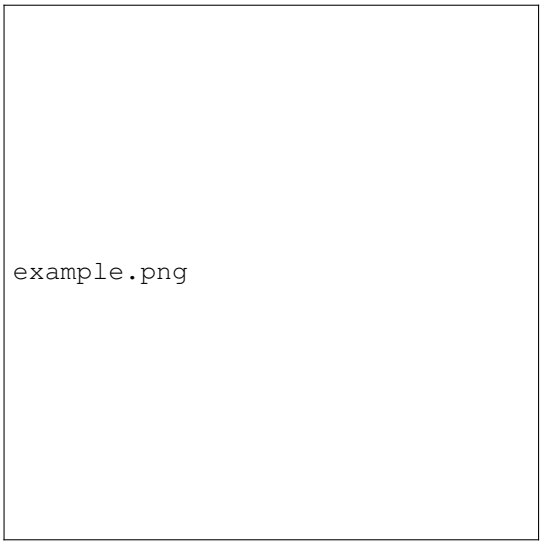

Fig. 1. Visualization of input RGB image, ground truth, and predicted DepthMap.

B. Results

Il modello ha ottenuto un RMSE di 0,5795 e un SSIM di 0,8563 sul set di addestramento, ma le prestazioni di validazione sono scese a RMSE=2,8105 e SSIM=0,5213. Ciò indica problemi di generalizzazione.

V. CONCLUSION

Questo lavoro presenta un approccio MDE supervisionato che combina un codificatore ResNet-34 e un decodificatore upsampling personalizzato. Mentre il modello ha ottenuto risultati promettenti sul set di addestramento, le prestazioni di validazione evidenziano la necessità di una migliore regolarizzazione e di un aumento dei dati.

La capacità umana di percepire la profondità con un occhio solo si basa sulla comprensione del contesto, sulle dimensioni degli oggetti e sul gioco di luci e ombre. Questo lavoro dimostra che le reti neurali profonde possono emulare tali capacità imparando dall'esperienza, evidenziando il loro potenziale nell'affrontare compiti tradizionalmente risolti da tecniche geometriche. Il lavoro futuro esplorerà codificatori avanzati, connessioni saltate e adattamento al dominio per migliorare la generalizzazione.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database, 2009, pp. 248–255.

[2] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable Are Features in Deep Neural Networks?", vol. 27, 2014, pp. 3320–3328.

[3] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014, pp. 512–519.

[4] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", in International Conference on Learning Representations (ICLR), 2015. Available: https://arxiv.org/abs/1409.1556.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI, 2015.

[6] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 448–456. Available: https://arxiv.org/abs/1502.03167.

[7] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. Available: https://arxiv.org/abs/1412.6980.

[9] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464–472. Available: https://arxiv.org/abs/1506.01186.

[10] K. He et al., "Deep Residual Learning for Image Recognition," CVPR, 2016.

[11] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," NeurIPS, 2014.

[12] I. Laina et al., "Deeper Depth Prediction with Fully Convolutional Residual Networks," 3DV, 2016.

[13] Z. Wang et al., "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE TIP, 2004.

[14] T. Zhou et al., "Unsupervised Learning of Depth and Ego-Motion from Video," CVPR, 2017.