UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"SVILUPPO FUNZIONE OBIETTIVO PER LA SEGMENTAZIONE SEMANTICA DI STRADE NEL TELERILEVAMENTO"

Relatore: Prof. Loris Nanni

Laureando: Davide Neffat

ANNO ACCADEMICO 2021 – 2022

Data di laurea 22/09/2022

Abstract

Attualmente è divenuto di fondamentale importanza il riconoscimento automatico delle strade a partire da immagini satellitari, basti pensare alle possibili applicazioni nella guida autonoma. Questo però non è affatto banale, soprattutto quando la vista è ostruita da alberi o altre strutture. La maggior parte delle ricerche esistenti sono concentrate sull'ottimizzazione delle reti di deep-learning disponibili. Tuttavia, l'accuratezza della segmentazione è anche influenzata dalla loss function, ovvero la funzione che valuta la distanza tra il modello e la soluzione ideale. Di conseguenza, la scelta della funzione obiettivo è strettamente legata alle prestazioni degli algoritmi perché questi sono orientati ad ottenere i migliori valori possibili per la funzione, modificando, di conseguenza, i parametri del sistema per avvicinarvisi. Lo studio propone l'analisi di una loss function chiamata GapLoss e la proposta di una nuova soluzione volta a migliorarla.

Indice

Capitolo 1

Introduzione

1.1 Segmentazione Semantica

La visione artificiale (nota anche come computer vision) è l'insieme dei processi che mirano a creare un modello approssimato del mondo reale (3D) partendo da immagini bidimensionali (2D). Lo scopo principale della visione artificiale è quello di riprodurre la vista umana. Vedere è inteso non solo come l'acquisizione di una fotografia bidimensionale di un'area ma soprattutto come l'interpretazione del contenuto di quell'area. L'informazione è intesa in questo caso come qualcosa che implica una decisione automatica.

Alcune delle maggiori tecniche utilizzate in computer vision sono:
- **Classificazione di immagini (image classification)** che ha l'obbiettivo di assegnare una determinata etichetta o classe all'intera immagine.
- **Localizzazione (localization)** che oltre ad assegnare una classe all'oggetto, ne determina anche la posizione attraverso un rettangolo di selezione (bounding box).
- **Rilevazione degli oggetti (object detection)** la quale identifica più oggetti in una stessa immagine e assegna loro una precisa posizione con l'uso di una bounding box.
- **Segmentazione semantica (semantic segmentation)** la quale non fa più uso dei rettangoli di selezione ma etichetta ogni singolo pixel, non distinguendo però diverse istanze di uno stesso oggetto.

- **Segmentazione dell'istanza (instance segmentation)** che riconosce i singoli pixel come appartenenti ad un determinato oggetto e distingue le varie istanze di quell'oggetto presenti nell'immagine.

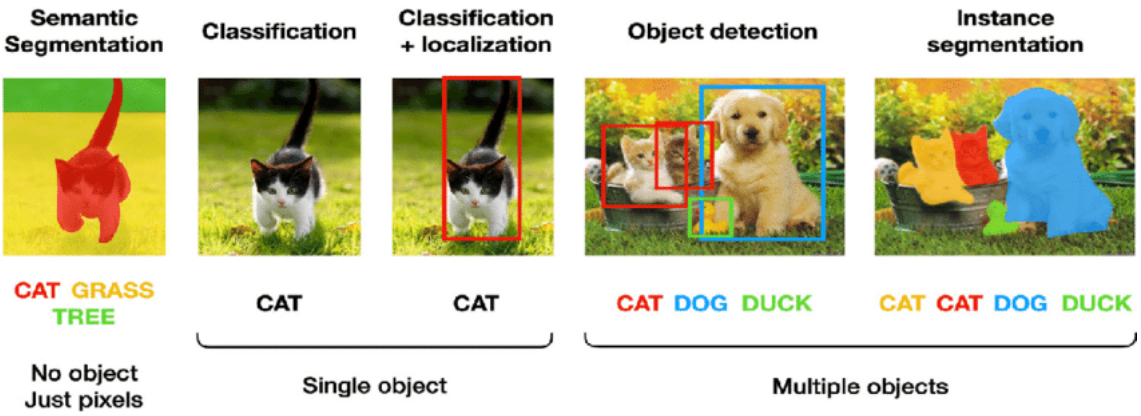

Figura 1: Varie tecniche utilizzate nel campo della computer vision

La segmentazione semantica è un algoritmo di deep learning che associa un'etichetta o una categoria a ogni pixel di un'immagine. Viene utilizzata per riconoscere una serie di pixel che formano categorie distinte. Ad esempio, un veicolo autonomo deve identificare veicoli, pedoni, segnali stradali, marciapiedi e altri elementi della strada.

Un semplice esempio di segmentazione semantica consiste nella separazione delle immagini in due classi. Ad esempio, nella Figura 2 vediamo un'immagine che mostra una persona in spiaggia e, a fianco, una versione che mostra i pixel dell'immagine segmentati in due classi separate: persona e sfondo.

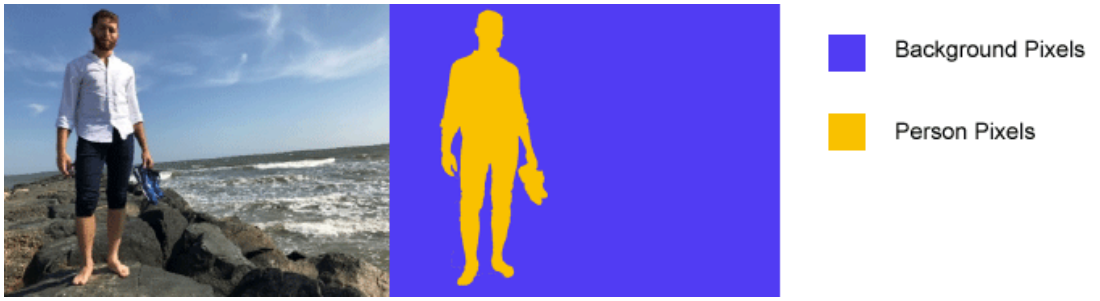

Figura 2: Esempio di segmentazione semantica

La segmentazione semantica non è limitata a due categorie. È possibile modificare il numero di categorie per la classificazione del contenuto dell'immagine. Questa stessa immagine potrebbe essere segmentata in quattro classi: persona, cielo, acqua e sfondo.

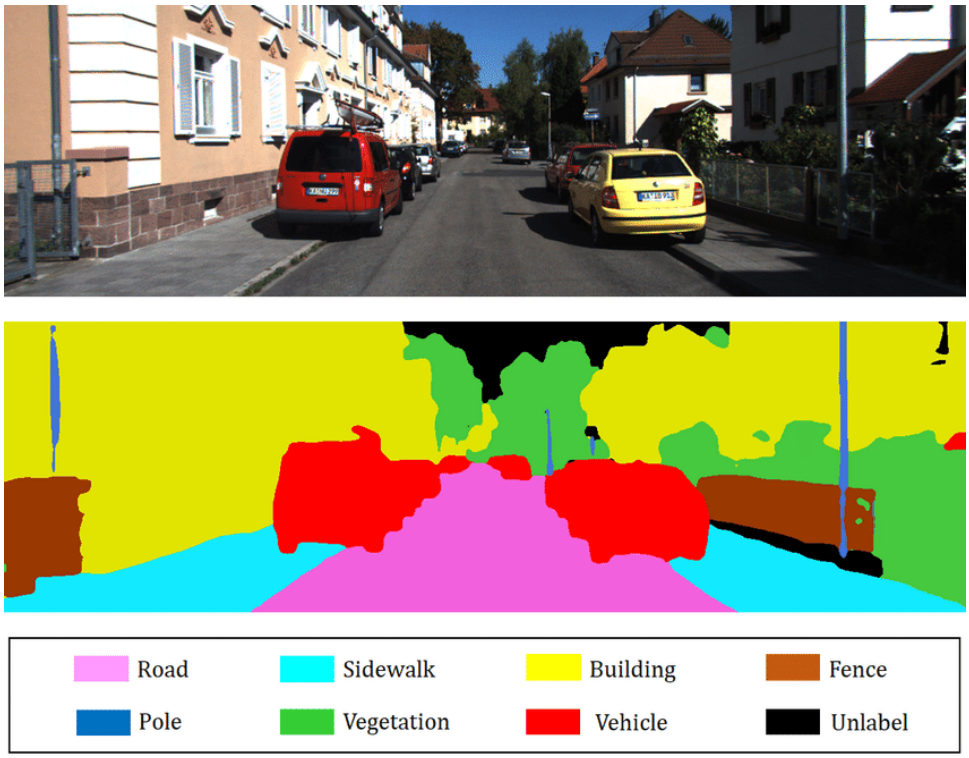

Figura 3: Esempio di segmentazione semantica multiclasse

Poiché la segmentazione semantica etichetta i pixel di un'immagine, è più precisa rispetto ad altre forme di rilevazione di oggetti. Per questo motivo, la segmentazione semantica è utile per applicazioni in una varietà di settori che richiedono mappe-immagine precise, come ad esempio:

- Guida autonoma: per identificare un percorso percorribile dalle auto separando la strada da ostacoli come pedoni, marciapiedi, pali e altre auto
- Ispezione industriale: per rilevare difetti nei materiali o nei prodotti finiti
- Immagini satellitari: per identificare montagne, fiumi, deserti e altri tipi di terreno per studi geologici o per l'edilizia
- Imaging medicale: per analizzare e rilevare anomalie cancerose nelle cellule
- Visione robotica: per identificare e muoversi tra oggetti e terreni

Il processo di addestramento di una rete di segmentazione semantica per classificare le immagini prevede i seguenti passaggi:

- Analizzare una raccolta di immagini con etichette per ogni pixel
- Creare una rete di segmentazione semantica
- Addestrare la rete per classificare le immagini in categorie di pixel
- Valutare la precisione della rete

1.2 Convolutional Neural Network (CNN)

Grazie all'aumentare della potenza di calcolo dell'hardware disponibile e alla creazione di vasti dataset per l'allenamento, nell'ultimo decennio il Deep learning ha preso piede sempre più. Il Deep learning sfrutta reti neurali cosiddette "profonde" (Deep Neural Network) composte da almeno due livelli nascosti (hidden) e ispirate nel funzionamento al sistema visivo degli organismi viventi. Una tipologia molto diffusa di reti "profonde" sono quelle convoluzionali (CNN), introdotte da LeCun nel 1980, varianti delle reti MLP usate prevalentemente per problemi di classificazione o regressione. Uno dei principali motivi della loro diffusione è che, a differenza delle reti MLP, le CNN sfruttano due caratteristiche per ridurre la loro complessità: il processing locale e i pesi condivisi. Nel primo caso i neuroni sono connessi solo localmente ai neuroni del livello precedente, mentre nel secondo i pesi delle connessioni sono condivisi da gruppi di neuroni. Queste peculiarità delle reti neurali convoluzionali permettono di ridurre drasticamente sia il numero delle connessioni sia quello dei pesi, semplificando notevolmente la struttura della rete.

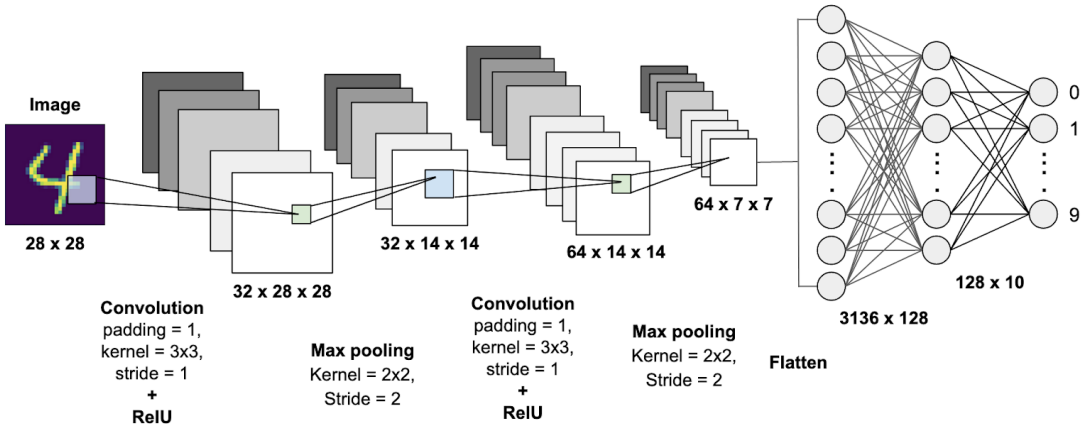

Figura 4: Struttura di una CNN

Le reti convoluzionali si distinguono anche per la costruzione a blocchi, queste infatti sfruttano livelli di convoluzione, di pooling, di attivazione e livelli completamente connessi. I livelli di convoluzione e di pooling in particolare svolgono la funzione di "pre-processing" dei dati, motivo per cui le CNN usano al minimo la pre-elaborazione degli input. Le CNN sono molto efficienti nel riconoscimento di immagini e video, nei sistemi di raccomandazione, nell'elaborazione del linguaggio naturale e ultimamente sono usate nel campo della bioinformatica.

1.3 Loss Function

Le loss function (funzioni obiettivo) sono usate per determinare l'errore tra l'output dell'algoritmo e la soluzione ottimale da raggiungere. Sono quelle funzioni che l'algoritmo tenta di minimizzare durante il training.

Alcune delle principali loss function sono:

1.3.1 **Mean Squared Error (MSE)**

Calcola l'errore quadratico ovvero la differenza al quadrato tra l'output predetto e quello reale per ogni pattern, somma tutti questi valori e ne calcola la media. Si tratta della loss function più famosa e più semplice in assoluto. Elevando le distanze al quadrato penalizza molto gli errori più gravi (outliers), ovvero quelli più distanti dal valore corretto.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$\hat{y}_i$ = output predetto

y_i = output corretto

n = numero di patterns

1.3.2 **Mean Absolute Error (MAE)**

Molto simile a MSE ma invece che calcolare il quadrato delle distanze ne calcola il valore assoluto. È molto più robusta rispetto agli outliers, tuttavia il valore assoluto rende l'equazione matematica più complessa da gestire.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} | x_i - x |$$

x = output predetto

x_i = output corretto

n = numero di patterns

Alcune loss function specifiche per problemi di classificazione sono:

1.3.3 **Binary Cross-Entropy**

La funzione Cross-Entropy è usata per calcolare la differenza tra due distribuzioni di probabilità; non è altro che la funzione logaritmo negata (-log(x))

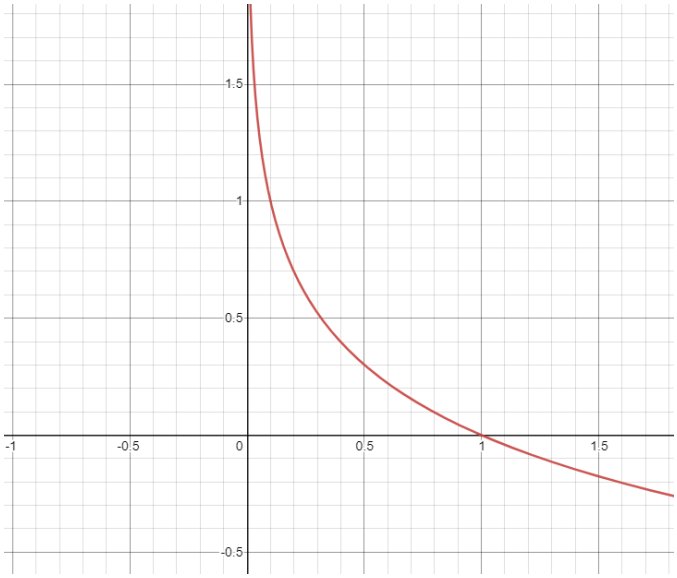

Figura 5: Grafico della funzione -log(x)

In questo modo l'output è inversamente proporzionale all'input. Binary Cross-Entropy viene utilizzata in problemi di classificazione comprendenti due sole classi e le etichette di output possono avere valori 0 o 1, oppure valori compresi tra 0 e 1. Ottiene i migliori risultati quando i pattern sono equamente distribuiti tra le due classi.

$$BCE = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{c}[y_i\, log(p_i\,) + (1 - y_i\,)log(1 - p_i\,)]$$

y – etichetta della classe attuale (0 o 1)

p -probabilità predetta per la classe

c – numero di classi (2 per la versione Binary)

n- numero di pattern

Pixel-Wise Softmax con Cross-Entropy è una variante di Binary Cross-Entropy usata per la segmentazione semantica, dato che usa come pattern i singoli pixel di un'immagine per verificare se sono stati classificati correttamente.

Cross-Entropy viene usata anche per problemi multi-classe, non considerando le probabilità delle classi sbagliate, ma solo la probabilità della classe corretta e semplificando quindi la formula al solo primo termine:

$$CE = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=0}^{c} y_i\, log\, \hat{y_i}$$

i – numero della classe

c – numero di classi

y_i – etichetta attuale

$\hat{y}_i$- etichetta predetta

1.3.4 **Dice Loss**

Un'altra importante funzione è Dice, la quale usa l'omonimo coefficiente per stimare la sovrapposizione dei pixel predetti e quelli reali. Il coefficiente Dice assume valori tra 0 e 1, dove 1 indica la perfetta sovrapposizione delle due immagini.

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|}$$

1.3.5 **Focal Loss**

Focal Loss modifica Pixel-Wise Softmax con Cross-Entropy riducendo il valore di loss nei pattern più facili da classificare. Questo avviene aggiungendo il termine $(1 - p_t)$ dove p_t è la probabilità stimata, elevandolo alla potenza gamma in modo da pesare maggiormente i pattern più difficili da classificare e meno quelli più semplici.

$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

1.3.6 **Tversky**

Tversky Loss è basata su Tversky Index, che misura la differenza tra due immagini segmentate.

Il Tversky Index (TIc) tra un'immagine Y e la corrispondente label T si calcola come

$$\mathrm{TI}_c = \frac{\sum_{m=1}^{M} Y_{cm} T_{cm}}{\sum_{m=1}^{M} Y_{cm} T_{cm} + \alpha \sum_{m=1}^{M} Y_{cm} T_{\bar{c}m} + \beta \sum_{m=1}^{M} Y_{\bar{c}m} T_{cm}}$$

- c indica l'appartenenza alla classe e c negato il contrario.
- M indica il numero di elementi che appartengono alle prime due dimensioni di Y.
- α e β sono dei pesi che servono per controllare i contributi che i falsi negativi e i falsi positivi apportano per ogni classe alla loss.

1.4 Deep learning per la segmentazione semantica (Deeplabv3+)

Per risolvere problemi di segmentazione semantica si possono usare diverse reti neurali e in questo progetto si è utilizzata Deeplabv3plus. Deeplab è una serie di reti neurali classiche per la segmentazione semantica sviluppate da Google. Utilizza un'architettura encoder-decoder in cui l'encoder si occupa di ottenere le feature maps dalle immagini di input, mentre il decoder recupera gradualmente i dettagli dell'oggetto e le dimensioni spaziali, ovvero utilizza l'upsampling per recuperare le informazioni dalle feature maps a basse risoluzioni.

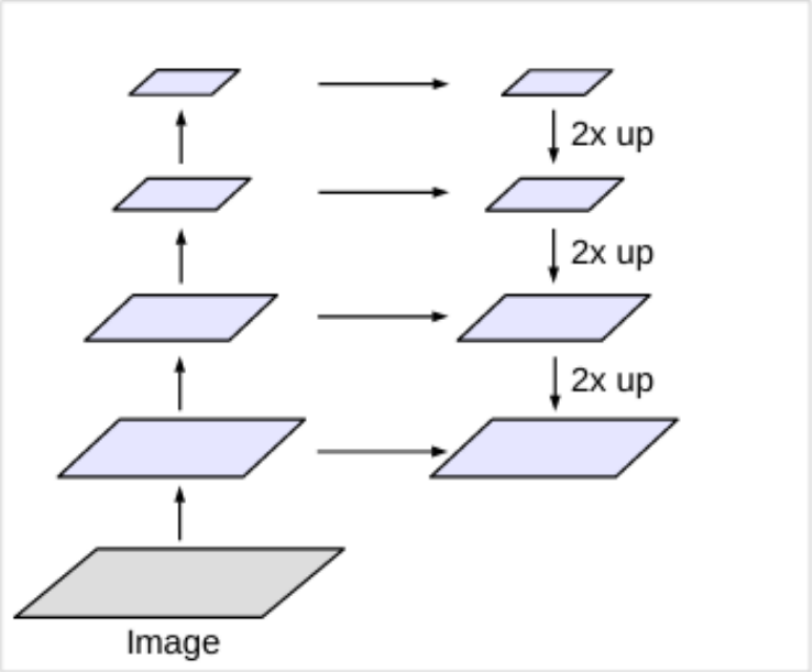

Figura 6: Diagramma dell'architettura encoder-decoder nel modello DeepLabV3

Atrous Convolution

Le reti neurali utilizzano convoluzioni ripetute e max-pooling, che sono efficaci ma portano a mappe spaziali a bassa risoluzione per poi essere recuperate tramite deconvoluzione. Deeplab introduce invece il concetto di Atrous Convolution, convoluzioni che portano a kernel con al loro interno degli spazi/buchi (zeri). Questo permette di aumentare la dimensione del kernel, catturando un'area più ampia ma senza aumentare il numero di parametri e quindi la complessità computazionale.

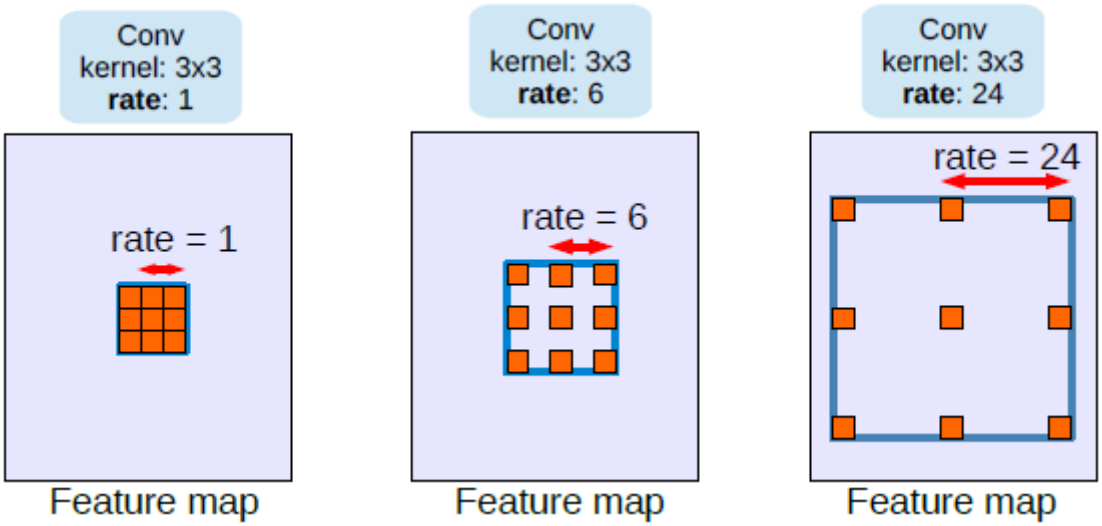

Figura 5: Varie applicazioni di Atrous Convolution variando il parametro rate

Spatial Pyramid Pooling (SPP)

Nel deep learning non tutti gli oggetti in un'immagine saranno della stessa dimensione. Alcuni oggetti saranno piccoli e altri grandi. In questi casi, per trattare oggetti con scale variabili, DeepLab usa Spatial Pyramid Pooling. Spatial Pyramid Pooling utilizza strati di pooling a diverse scale per gestire immagini e oggetti a scale variabili. Questo avviene applicando in modo parallelo quattro Astrous Convolutions con diversi astrous rates ad una feature map, che permette di ridimensionare le feature maps in scale diverse.

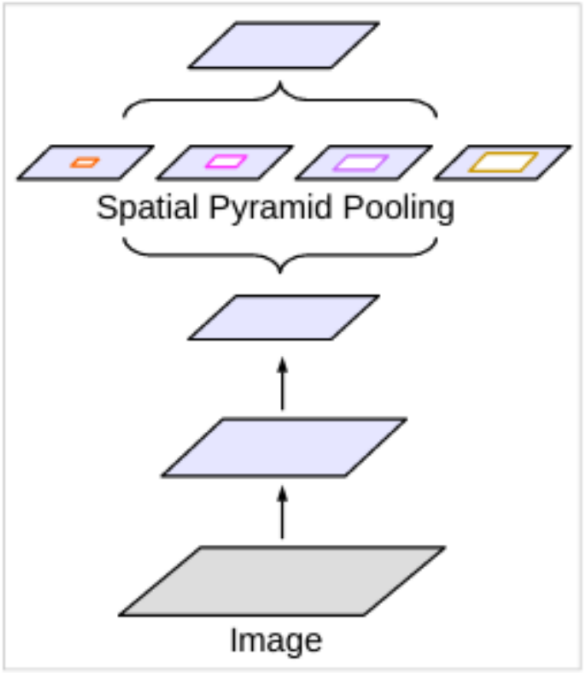

Figura 8: Spatial Pyramid Pooling a diverse scale

Capitolo 2

Materiali e Metodi

2.1 Presentazione del dataset

Massachusetts Roads Dataset è un dataset pubblico ottenuto tramite telerilevamento; contiene immagini satellitari dello stato del Massachusetts, USA, ed è composto da 1108 immagini per il training, 14 immagini per la validation, 49 immagini di test e le corrispettive etichette.
In questo progetto non è compresa la fase di validazione per cui training e validation sets sono stati unificati; le immagini avevano formato TIFF e dimensione 1500x1500 pixel e sono state modificate in formato PNG (immagini) e BMP (maschere) in modo da adattarsi alla rete Deeplabv3plus, e tagliate per ridurre la dimensione a 512x512 pixel così da rendere la fase di training più semplice e rapida. È stato inoltre ridotto il numero degli elementi del dataset a 248 per il training set e 48 per il validation set, a causa dei limiti computazionali della GPU in possesso per l'addestramento. Le maschere erano inizialmente a 8 bit in scala di grigi e utilizzando Matlab sono state convertite in bianco e nero (1 bit).

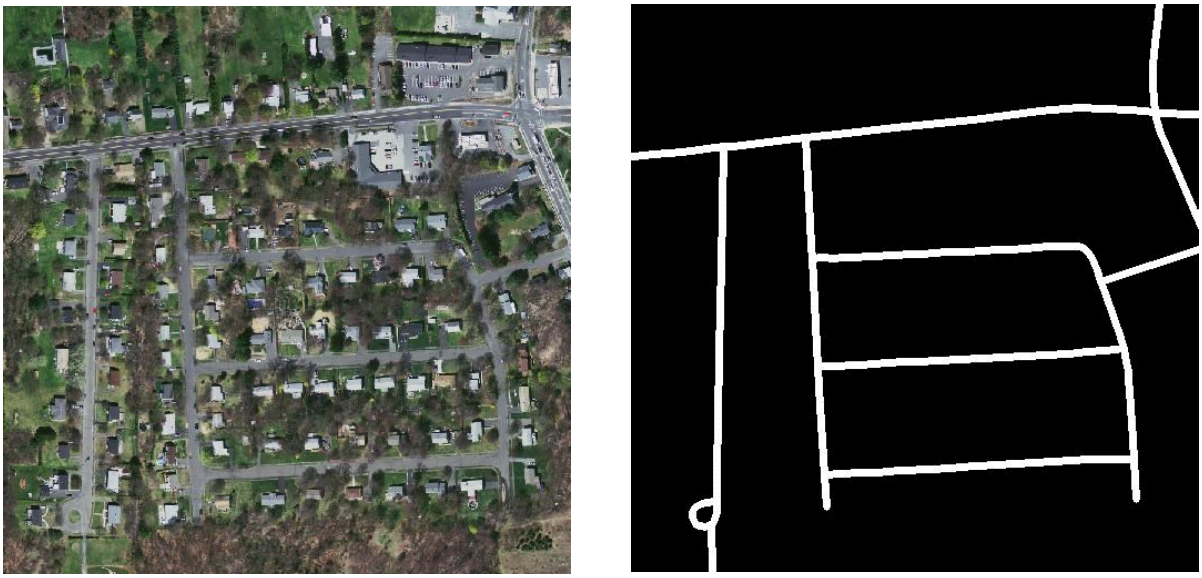

Figura 9: Un'immagine del training set e la rispettiva maschera

2.2 GapLoss

Abbiamo parlato precedentemente delle loss function e della loro utilità; questo progetto mira ad analizzarne un'alternativa a quelle classiche per risolvere un problema specifico: tale funzione è chiamata GapLoss. Nei problemi di segmentazione volti a riconoscere le strade all'interno di immagini satellitari la difficoltà principale consiste nel fatto che le strade possono apparire disconnesse, in particolare in prossimità di incroci o quando coperte da ostacoli come alberi.
L'imprecisione della segmentazione può portare a problemi significativi se utilizzata in applicazioni successive che si basano sulla precisione dei dati.

Come mostrato in Figura 10, Figura 10a rappresenta l'immagine di ingresso, Figura 10b rappresenta l'etichetta corrispondente e Figura 10c è il risultato di segmentazione della rete usando Cross-Entropy come loss function. Nella Figura 10c, si nota come siano presenti tratti disconnessi in strade che dovrebbero essere collegate. A causa degli alberi presenti la rete ha classificato come non strade diverse parti di strade. Inoltre, poiché la maggior parte dei pixel che compongono le strade hanno forme lineari, la rete è stata addestrata a giudicare solo le forme lineari come strade; pertanto, l'intersezione delle strade mostrata nel cerchio rosso della Figura 10c è stata ignorata, in quanto non era linearmente sagomata. Inoltre, non potevamo prevedere direttamente un'immagine più accurata usando algoritmi topologici per eliminare queste lacune poiché, in realtà, le strade a volte sono scollegate, come mostrato nel cerchio verde della Figura 10c.

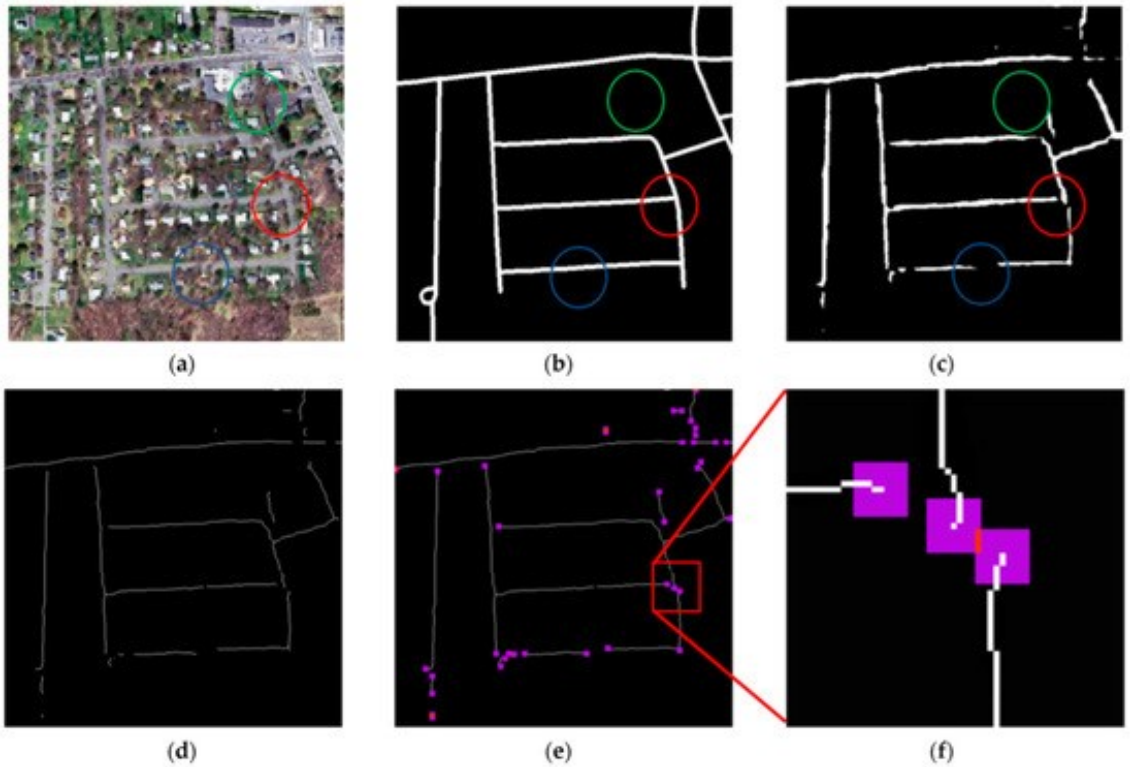

Figura 10: Schema del calcolo di GapLoss. (a) Immagine di input. (b) Maschera corrispondente. (c) Strade predette dalla rete. (d) Funzione skeleton applicata alle strade predette. (e) Riconoscimento degli end-point. (f) Visuale ingrandita di (e)

Il motivo che porta una strada ad essere classificata come disconnessa può essere anche solo l'errata classificazione di pochissimi pixel. 10 pixel, a volte anche meno, possono creare una discontinuità a seconda anche della larghezza della strada e della risoluzione dell'immagine. Poiché i pixel giudicati erroneamente rappresentano una parte insignificante dell'intera immagine, il modello di segmentazione generale e la loss function non riescono a catturare sufficientemente le sue caratteristiche durante l'allenamento. Tuttavia, il peso di questi pixel nella loss function può essere aumentato per far sì che la rete presti maggior attenzione a questi pixel mal classificati. Se questi pixel sono effettivamente degli errori verranno corretti in fase di training così da ridurre il valore della loss function, migliorando così la connettività delle strade e l'accuratezza del modello.

Come primo passo è necessario identificare questi pixel che creano le discontinuità, ma il metodo di calcolo non può essere eccessivamente complesso, in quanto ciò aumenterebbe il tempo di addestramento e richiederebbe più memoria GPU, il che rende difficile, se non impossibile, completare lo studio su computer al di fuori di ambienti di ricerca specifici. Un modo per identificare questi pixel è quello di prendere in considerazione i pixel finali dei vari segmenti di strade. Pertanto, sono state estratte le linee vettoriali delle strade predette, come mostrato in Figura 10d. Successivamente, si sono identificati i pixel finali di ogni linea ed è stato aggiunto un intervallo di buffer in cui la rete dovrebbe prestare maggiore attenzione, come mostrato in Figura 10e. I pixel viola indicano dove il peso sarà aumentato. I pixel rossi indicano dove i due range si sovrappongono, e il loro peso sarà raddoppiato, come mostrato nell'immagine ingrandita in Figura 10f. Questi pixel rossi sono più lontani dalla strada predetta rispetto ai pixel viola e sono più inclini ad essere giudicati erroneamente, quindi abbiamo bisogno di raddoppiare il loro peso per risolvere l'errore.

I passaggi dell'algoritmo di GapLoss sono i seguenti:

1) I valori predetti di tutti i pixel sono processati usando la funzione Softmax per ottenere la Cross-Entropy map L; poi i valori predetti sono resi binari così da ottenere l'immagine A (A = 0 se $y_i < 0.5$, A = 1 se $y_i >= 0.5$, dove y_i è il valore predetto del pixel i dopo aver applicato la funzione Softmax).

2) Lo scheletro dell'immagine (i vettori linee delle strade) è estratto da A, come mostrato in Figura 10c così da ottenere l'immagine B, in Figura 10d.

3) Considerando ogni pixel di B, se c'è un solo pixel con valore 1 negli 8 pixel adiacenti allora quello è considerato un endpoint, il suo valore viene impostato a 1, gli altri pixel vengono settati a 0 e si crea così l'immagine C.

4) Considerando ogni pixel di C, se c'è un endpoint della griglia di 9x9 pixel adiacenti, il peso del pixel viene settato al valore K, se ce ne sono 2 il peso diventa 2K, se ce ne sono 3 3K e così via. Se non ci sono endpoint nella griglia 9x9 il peso del pixel varrà 1. Si crea così la mappa dei pesi W.

5) Ogni pixel nella mappa dei pesi W viene moltiplicato per il corrispondente pixel della Cross-Entropy map L e calcolando il valore medio tra tutti questi prodotti si ottiene il valore di GapLoss.

$$GapLoss = mean(W \times L)$$

Dove L è la Cross-Entropy map e W la weight map

2.3 Metriche

Per comprendere se il nostro sistema di classificazione stia avendo successo o meno vengono utilizzate le metriche. Prima di introdurre alcune delle metriche più diffuse è utile dare le seguenti definizioni:

Veri Positivi (VP): il pixel viene correttamente classificato come positivo.
Veri Negativi (VN): il pixel viene correttamente classificato come negativo.
Falsi Positivi (FP): il pixel viene erroneamente classificato come positivo.
Falsi Negativi (FN): il pixel viene erroneamente classificato come negativo.

Accuratezza

L'accuratezza è probabilmente una delle metriche più intuitive, si ottiene attraverso il rapporto tra pixel classificati correttamente e numero totale di pixel:

$$Accuratezza = \frac{VP + VN}{VP + FP + FN + VN}$$

Questa non viene considerata affidabile nei casi in cui una classe è predominante su un'altra, quindi nei problemi con classi fortemente sbilanciate.

Precisione

La precisione viene calcolata con il rapporto tra pixel classificati correttamente come positivi e numero totale di pixel classificati come positivi:

$$Precision = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$$

Recall

La metrica recall indica quanto il sistema è selettivo e viene descritta dalla seguente equazione

$$Recall = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}}$$

F2-score

La metrica F2-score cattura la precisione e il richiamo in una singola equazione:

$$F2 - score = \frac{5 \cdot Precisione \cdot Recall}{4 \cdot Precisione + Recall}$$

Intersection over Union (IoU)

IoU viene definita come:

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP} + \mathrm{FN}}$$

Dove A è la segmentation map prodotta dal nostro modello di classificazione mentre B è la corretta segmentation map.

Dice

Il coefficiente Dice viene calcolato come il doppio dell'area di intersezione tra la segmentation map prodotta dal nostro modello di classificazione e la corretta segmentation map diviso il numero totale di pixel.

$$Dice = \frac{|A \cap B|}{|A| + |B|} = \frac{2 \cdot \mathrm{VP}}{2 \cdot \mathrm{VP} + \mathrm{FP} + \mathrm{FN}}$$

Capitolo 3

Risultati

3.1 Esperimenti

Nel primo esperimento viene usato Deeplabv3+ con resnet18 per addestrare il modello utilizzando Cross-Entropy come loss function. Il learning rate è impostato a 0,001, il numero di epoche è 20 e la dimensione dei mini-batch 5. Si sono ottenuti i seguenti risultati:

Global Accuracy	0,93851
Mean Accuracy	0,71105
Precision	0,4364
Recall	0,7408
F2 score	0,6501
IoU	0,3786
Dice	0,5493

Tabella 1: Risultati del test eseguito con Cross-Entropy

La seguente immagine rappresenta un elemento del validation set, con la rispettiva predizione effettuata dal modello:

Figura 11: Risultati del test eseguito con Cross-Entropy su un'immagine del validation set

Nel secondo esperimento rimangono invariati i parametri ma si utilizza GapLoss come loss function al posto di Cross-Entropy, ottenendo risultati leggermente migliori dei precedenti:

Global Accuracy	0,93928
Mean Accuracy	0,72298
Precision	0,4618
Recall	0,7318
F2 score	0,6552
IoU	0,3950
Dice	0,5623

Tabella 2: Risultati del test eseguito con GapLoss

Il *Figura 12: Risultati del test eseguito con GapLoss*
su un'immagine del validation set

Il terzo esperimento consiste nel trovare un miglioramento alla funzione GapLoss. Notando che GapLoss tende a unire i tratti di strada interrotta, ma solo se a distanza minore di 9 pixel, mentre in molte immagini si arriva anche a tratti di 40/50 pixel si è cercato di risolvere questo problema eliminando il vincolo dei due end-point a distanza di 9 pixel massimo. Per incrementare ulteriormente le performance del modello si è utilizzata un'altra informazione a nostra disposizione: se una strada inizia in una direzione è molto probabile che continui verso quella direzione. Si è utilizzato quindi GapLoss come punto di partenza per trovare i pixel classificabili come end-point dei segmenti predetti dopo aver applicato la funzione Skeleton. A questo punto per ogni end-point si deduce da che direzione veniva la strada (osservando se i precedenti pixel classificati come strada vengono da destra/sinistra/sopra/sotto) e prendono in esame i 3 pixel successivi nella direzione opposta (quella dove a logica dovrebbe continuare la strada ma il modello non ha ancora riconosciuto come continuazione della strada). L'errore della funzione Cross-Entropy di questi pixel viene moltiplicato per una costante il cui valore ottimale rilevato è 10 e calcolando la media tra gli errori di tutti i pixel si ottiene il valore di loss. Così facendo si dà maggiore peso agli errori sui pixel che dovrebbero appartenere al proseguo della strada e i tratti predetti risultano così più continui e privi di interruzioni.

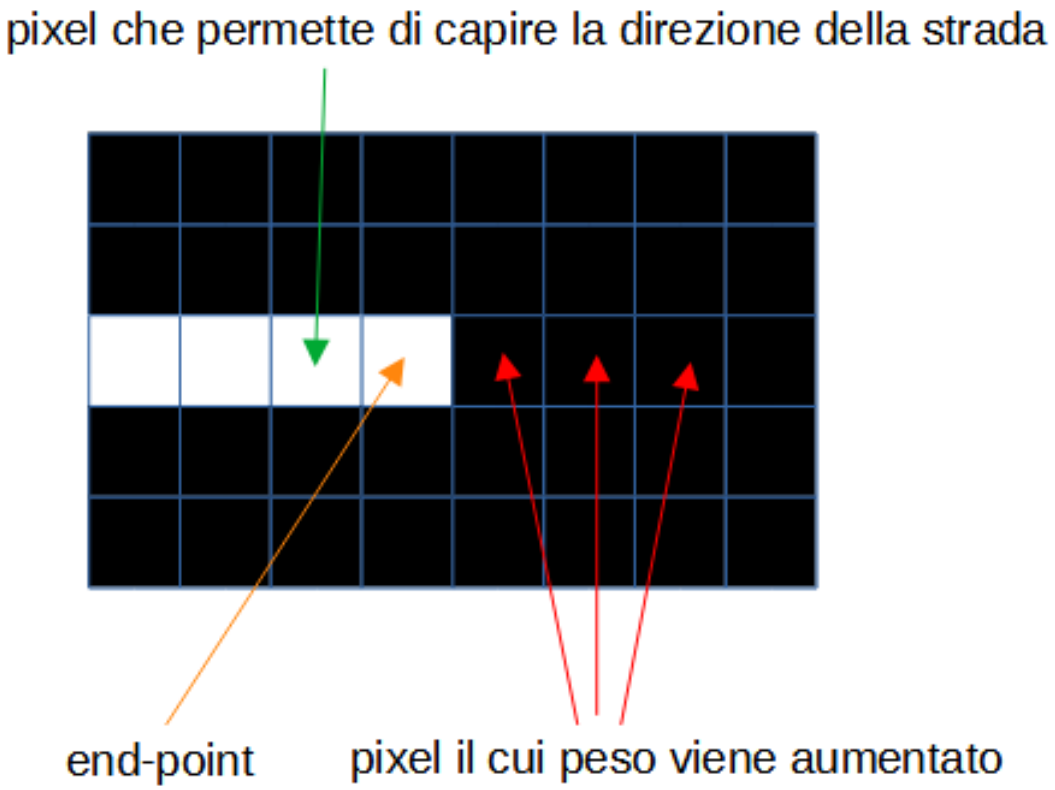

Figura 13: Raffigurazione dell'idea alla base del miglioramento applicato a GapLoss

Utilizzando questa loss function si ottengono i seguenti risultati:

Global Accuracy	0,94071
Mean Accuracy	0,73088
Precision	0,4778
Recall	0,7360
F2 score	0,6642
IoU	0,4079
Dice	0,5795

Tabella 3: Risultati del test eseguito con il miglioramento di GapLoss

Figura 14: Risultati del test eseguito con il miglioramento di GapLoss su un'immagine del validation set

Con questa modifica a GapLoss l'accuratezza e le metriche analizzate risultano migliorate e le strade predette più definite e continue, anche se con le dimensioni del dataset così ridotte appaiono comunque delle interruzioni. Questo problema può però essere facilmente risolto aumentando la risoluzione e il numero di immagini utilizzate in fase di training

Capitolo 4

Conclusioni

I risultati ottenuti danno un'ottima panoramica della bontà delle loss functions utilizzate, ma non devono essere presi come potenziale massimo di questo modello. A causa delle limitate capacità computazionali dell'hardware utilizzato le dimensioni delle immagini sono state più che dimezzate, così come è stato ridotto il numero stesso delle immagini utilizzate per l'addestramento. Queste modifiche hanno inevitabilmente diminuito l'efficacia della rete nella classificazione dei singoli pixel e di conseguenza anche i valori delle metriche calcolate, ma l'obiettivo dell'elaborato era quello di confrontare le performance delle 3 loss functions usate negli altrettanti esperimenti e questo confronto non è stato alterato dalle modifiche effettuate al dataset.
GapLoss si rivela più efficace della semplice Cross-Entropy, riuscendo a rendere i segmenti di strade predetti dal modello più continui e lineari ma tenendo comunque il numero di falsi negativi elevato. Per ovviare questo problema si utilizza la versione migliorata, la quale elimina più velocemente le interruzioni e di conseguenza migliora le metriche prese in esame.
Come ottimizzazione di questo elaborato si potrà in futuro ripetere l'esperimento utilizzando hardware più sofisticato e adatto, potendo così aumentare la dimensione del dataset e delle immagini in modo da raggiungere valori di accuracy attorno al 96 – 97%.

Bibliografia

[1] "Segmentazione Semantica 3 cose da sapere" https://it.mathworks.com/solutions/image-video-processing/semantic-segmentation.html

[2] Tomer Kordonsky. "Loss Functions Loss functions explanations and examples". 14/08/2021 https://medium.com/artificialis/loss-functions-361b2ad439a0

[3] Darth Espressius. "3 Common Loss Functions for Image Segmentation". 30/01/2022 https://dev.to/_aadidev/3-common-loss-functions-for-image-segmentation 545o#:~:text=In%20this%20blog%20post%2C%20I%20will%20focus%20on,Cross-Entropy%20Loss%2C%20Dice%20Loss%20and%20the%20Shape-Aware%20Loss.

[4] Lavanya Gupta. "Focal Loss — What, Why, and How?". 28/01/2021 https://medium.com/swlh/focal-loss-what-why-and-how-df6735f26616#:~:text=Hence%2C%20the%20Focal%20Loss%20function%20is%20a%20dynamically,to%20the%20loss%20based%20on%20the%20classification%20error.

[5] "Tversky index". 19/02/2022

https://en.wikipedia.org/wiki/Tversky_index

[6] “Define Custom Pixel Classification Layer with Tversky Loss” https://it.mathworks.com/help/vision/ug/define-custom-pixel-classification-layer-with-tversky-loss.html#:~:text=The%20Tversky%20loss%20is%20based%20on%20the%20Tversky,1%20M%20Y%20c%20%E2%80%BE%20m%20T%20cm

[7] Salehi, Seyed Sadegh Mohseni, Deniz Erdogmus, and Ali Gholipour. "Tversky loss function for image segmentation using 3D fully convolutional deep networks." *International Workshop on Machine Learning in Medical Imaging*. Springer, Cham, 2017

[8] Sovit Ranjan Rath. “Semantic Segmentation using PyTorch DeepLabV3 ResNet50”. 24/05/2021 https://debuggercafe.com/semantic-segmentation-using-pytorch-deeplabv3-resnet50/

[9] Soumik Rakshit. “Multiclass semantic segmentation using DeepLabV3+”. 31/08/2021 https://keras.io/examples/vision/deeplabv3_plus/

[10] Wei Yuan, Wenbo Xu. “GapLoss: A Loss Function for Semantic Segmentation of Roads in Remote Sensing Images”. 18/05/2022 https://doi.org/10.3390/rs14102422