UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"GapLoss: una loss function per segmentazione semantica di strade in immagini satellitari"

Relatore: Prof. Loris Nanni

Laureando: Kristian Marcon

ANNO ACCADEMICO 2021 – 2022

Data di laurea 14 novembre 2022

Sommario

Attualmente l'analisi della continuità stradale nelle immagini satellitari è una sfida complessa dovuta alla difficoltà nell'individuare il vettore direzionale dei tratti stradali, soprattutto quando la vista satellitare delle strade è ostruita da alberi o altre strutture. Al giorno d'oggi, la maggioranza delle ricerche si sofferma sull'ottimizzazione delle reti deep learning già disponibili per questo scopo, tuttavia, l'accuratezza della segmentazione di queste è influenzata specialmente dalla loss function usata nei test.
Attualmente sono state pubblicate poche ricerche su loss function usate nell'ambito della segmentazione stradale. Per risolvere questo problema, lo studio si è concentrato in particolare su una loss function denominata GapLoss, combinabile con qualsiasi rete di segmentazione proposta. Il procedimento di questa loss function è prettamente un'analisi topologica dei pixel ed è basato su cinque fasi, alla fine delle quali si attribuisce un peso maggiore per i pixel problematici che rappresentano interruzioni stradali non reali.
In questo studio verranno esaminati tre modelli di segmentazione semantica per condurre confronti tra i risultati ottenuti utilizzando un dataset. Lo scopo è confrontare i valori ottenuti da questa loss function e dalla versione alternativa, con quelli ottenuti dal modello basato su Dice Loss, fissati come obbiettivo. Inoltre nelle conclusioni va controllato se nell'immagine predetta, la continuità stradale risulta abbastanza continua, soprattutto nelle intersezioni e nelle parti in cui le strade sono nascoste da ostacoli di vario tipo.

Indice

Introduzione

Nell'ambito delle infrastrutture, le strade sono uno dei principali soggetti di studio nell'ambito del tracciamento e della mappatura, dato che influenzano le attività di trasporto, l'uso della navigazione satellitare e la cartografia elettronica; un problema principale che le caratterizza però è l'estrazione manuale della loro struttura topologica dalle immagini satellitari, spesso eseguita per soddisfare lo sviluppo della maggior parte delle applicazioni di navigazione. Questo tracciamento strutturale è un'operazione molto onerosa e complessa nella pratica e per semplificarlo molti ricercatori hanno studiato come automatizzare l'estrazione stradale sfruttando il machine learning e reti neurali molto avanzate.
Nelle fasi iniziali degli studi, i metodi di estrazione automatica utilizzarono principalmente le caratteristiche spettrali delle immagini satellitari e furono supportati da un algoritmo morfologico per individuare la soglia appropriata, al fine di segmentare la strada. Principalmente fu notato che le strade potevano caratterizzarsi per la loro morfologia, ma risultò difficile distinguerle ad esempio da una casa, a causa dell'eccessiva somiglianza tra queste e gli edifici.
Altri ricercatori invece applicarono metodi di apprendimento automatico, come $random\ forest$, SVM, i metodi basati sul $mean-shift$ e gli algoritmi di $clustering\ K-means$, per estrarre le strade dalle immagini satellitari, ma il successo di questi metodi fu molto condizionato dalle caratteristiche di progettazione, le quali rappresentarono un'ulteriore sfida.
Successivamente con l'introduzione del deep learning, i metodi di ricerca tradizionali cambiarono radicalmente. Venne adottato l'uso di reti deep learning come $UNet$, la famiglia $DeepLab$, $SegNet$, $PSPNet$, $UNet++$ e $MUNet$, capaci di estrarre caratteristiche da un'immagine in modo più accurato e affidabile rispetto all'apprendimento automatico tradizionale, per questo diventarono i principali metodi utilizzati per la segmentazione stradale. Data questa innovazione, molti ricercatori applicarono così il deep learning alla segmentazione delle strade utilizzando immagini satellitari. Con il passare degli studi fu notato però che l'accuratezza della segmentazione dipendeva

più dalla loss function applicata, ma poche sperimentazioni produssero prove sufficienti a dimostrare ciò. A distinguersi maggiormente per i risultati, fu la loss function $cross-entropy$, mentre le altre come $focal\ loss$, $Tversky\ loss$, $log-cosh\ dice\ loss$ e $generalized\ dice\ loss$, furono progettate per insiemi di dati non bilanciati, ma non riuscirono a garantire una continuità stradale accurata, in quanto non progettate per il riconoscimento di oggetti lineari. Una lacuna aggiuntiva sta nel fatto che esistono poche loss function progettate per questo ambito, in più molte di queste, essendo integrate nella rete neurale con cui sono state progettate, non possono essere adattatate ad altre reti.
Per cercare di ottenere risultati migliori dalle loss function appena citate, in questa tesi viene proposta una loss function denominata *GapLoss* e una sua versione alternativa, combinabili con qualsiasi rete di segmentazione attualmente disponibile o futura, cosa che permette di testare vari modelli. Essa è progettata specificamente per affrontare i problemi di continuità stradale, per migliorare la connettività dei tratti e la segmentazione stradale contemporaneamente nei risultati finali.
Alla fine di questo studio verranno poi confrontati i risultati di segmentazione ottenuti tra tre modelli, due basati sulle due versioni di GapLoss e uno sulla loss function DiceLoss, tutti testati sul dataset Massachusetts Road Dataset ridotto.

Capitolo 1

Nozioni fondamentali

In questo capitolo introduttivo verranno presentati dei concetti utili per focalizzare al meglio il contesto su cui si è andato a svolgere l'esperimento in esame.
Per prima cosa, nelle sezioni 1.1 e 1.2 saranno visti a grandi linee i concetti di deep learning e rete neurale, per poi toccare l'ambito più specifico di questa tesi, ovvero segmentazione semantica e loss function, rispettivamente nelle sezioni 1.3 e 1.4.

1.1 Deep Learning

Come suggerisce il nome, il *deep learning* si basa sul eseguire addestramenti profondi di una macchina intelligente, a scopo di analizzare grandi quantità di dati, completare attività complesse e risolvere problemi molto laboriosi. Esistono varie categorie di questo:

- il *deep learning supervisionato* (come nel caso in esame) nel quale l'algoritmo intelligente, dopo aver determinato delle informazioni, verifica se esse siano vere, controllando delle etichette incluse nei dati analizzati e fornite dall'essere umano (che supervisiona);

- il *deep learning non supervisionato* invece si basa sul training di algoritmi intelligenti con dati non etichettati che successivamente vengono ordinati e classificati autonomamente dal sistema.

I dati in questioni solitamente sono organizzati in *dataset* e sono l'input fondamentale per permettere l'addestramento della macchina intelligente che, nel nostro caso specifico, ha lo scopo di classificare, una volta imparato a

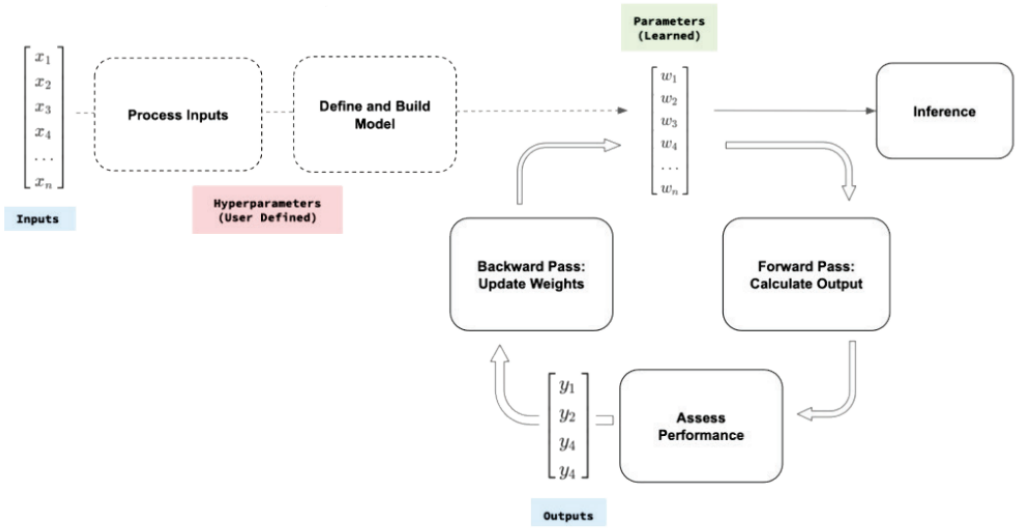

Figura 1.1:
Generalizzazione di un sistema deep learning

farlo mediante il training. Il procedimento di base è un confronto tra previsioni fatte dalla macchina e i risultati noti, per poi aggiornare i parametri specifici, in modo da correggere e affinare il risultato che essa restituisce. L'addestramento si basa su tre fasi fondamentali:

- prima di tutto avviene il *training* dove inizialmente vengono sfruttati dati presenti in un *training set*, come se fossero esempi dati all'algoritmo che deve imparare a interpretare;
- successivamente si passa al *validating* usando un diverso *validation set* con dati inediti; in questa fase la macchina valuta questi al fine di aggiustare i cosiddetti *iperparametri* dell'algoritmo, fondamentali per tarare il suo comportamento;
- infine si ha il *testing*, che tramite un *test set*, con dati mai usati finora e con etichette non visibili al modello, permette di dare delle conclusioni sulla qualità dell'addestramento in base ai risultati e ai valori delle metriche di valutazione.

Una volta completato questo procedimento, sarà possibile utilizzare il modello addestrato per elaborare risultati in base allo scopo e al problema che si vuole affrontare. Genericamente, per ogni modello vanno definiti vari fattori

come iperparametri, il numero e la dimensione dei livelli della rete neurale usata e dalla funzione di attivazione dei neuroni di essa, dettagli su cui non ci soffermeremo, non essendo fulcro centrale di questa tesi.
Nel nostro caso, il modello ha lo scopo di classificare i pixel delle immagini satellitari, in modo da restituire come risultato un'immagine aventi pixel etichettati come strada e altri come "non strada"; in particolare verranno addestrati e confrontati tre modelli in base alla combinazione tra una rete neurale (ResNet18) e tre loss function usate.

1.2 Rete neurale

Nonostante gli studi su esse nacquero già agli inizi del 1990, le reti neurali sono l'elemento del deep learning con più alto sviluppo negli ultimi anni, specialmente grazie alla loro versatilità e adattabilità ai vari ambiti e problemi di intelligenza artificiale esistenti. Con il passare degli anni, il concetto si è poi evoluto in *complex networks*, delle reti più evolute rispetto alle simple networks, in grado di gestire problemi più importanti.

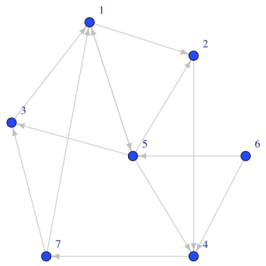

Figura 1.2:
Grafo generico

Il concetto chiave per capire la struttura di una rete neurale è il $grafo$ (esempio in figura 1.2), una struttura dati che rappresenta al meglio la topologia generica delle reti neurali. In un grafo esistono *vertici*, ovvero dei punti o nodi, collegati dai rami detti *archi*. Facendo un paragone per le reti, i vertici corrispondono ai neuroni che formano la rete, mentre gli archi rappresentano le connessioni tra di essi: queste ultime trasmettono le informazioni da neurone in neurone, sempre e solo con una direzione; ciò vuol dire che per una coppia di neuroni connessi, l'output di uno è l'input dell'altro o viceversa. In particolare, ogni neurone ha due gradi riguardanti le connessioni che incidono in esso: il *grado interno* è il numero di connessioni entranti che trasmettono l'input al neurone stesso, mentre il *grado esterno* è il numero di connessioni uscenti che trasmettono il suo output ad altri neuroni. In una rete neurale

solitamente, i nodi sono organizzati in più livelli (come mostrato in figura 1.3) con una struttura matriciale, tra i quali appunto vi sono le connessioni tra neuroni.

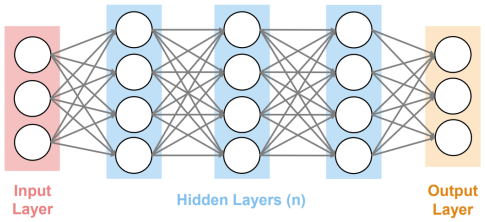

Figura 1.3:
Livello generico

Matematicamente, il neurone elabora delle informazioni in input, ovvero gli ingressi $x_1, x_2, ..., x_n$ con i rispettivi pesi $w_1, w_2, ..., w_n$; questi ingressi pesati vengono sommati tra di loro, dopodichè alla somma può essere aggiunto un *bias* w_0, utile a dare più flessibilità e a tarare il comportamento del neurone; la somma ottenuta viene chiamata *net* e successivamente il suo valore viene sfruttato da una funzione di attivazione che restituisce l'*output* finale del neurone.

$$net = \sum_{i=1}^{n} w_i \cdot x_i + w_0$$
$$output = f(net).$$

Altro aspetto importante è che la connettività tra livelli varia in base a alla profondità della rete: un neurone di livello a può inviare l'output a m neuroni del livello b e un neurone del livello b a sua volta può inviare l'output a n neuroni del livello c, dove m e n possono essere diverse.
Sostanzialmente esiste sempre un primo livello che riceve l'input dato alla rete, che in questo caso è un'immagine satellitare, dove ogni neurone tratta uno o più pixel di una porzione di essa. I neuroni elaborano le informazioni, trasmettendone il risultato al prossimo livello e così via, fino ad arrivare all'ultimo che restituisce l'output finale della rete, che per questo progetto dovrà essere la strada segmentata.
Questo è lo stesso concetto di base su cui si basa la rete ResNet-18 implementata nel Deep Learning Toolbox e usata per i test di questa tesi con Matlab 2022a. ResNet-18 è una rete neurale convoluzionale con 18 livelli di elaborazione; essa è disponibile in versione pre-addestrata su più di un milione di

immagini dal database ImageNet. Questa rete è in grado di classificare le immagini in 1000 categorie di oggetti, come tastiera, mouse, matita e molti animali, di conseguenza, la rete ha appreso rappresentazioni ricche di funzionalità per un'ampia gamma di immagini. Nel nostro caso specifico viene utilizzata per effettuare una segmentazione semantica di immagini satellitari, al fine di evidenziare le strade e ha una dimensione di input adattata in base al test effettuato, date le diverse risoluzioni delle immagini di input dei diversi set.

1.3 Segmentazione semantica

In questa tesi verrà sfruttato un modello deep learning per effettuare una segmentazione semantica di immagini satellitari ad alta risoluzione, che consiste nel assegnare una classe a ciascun pixel via classificazione supervisionata. Tutto ciò sfrutta le *convolutional neural networks* (CNNs) che negli ultimi anni hanno visto uno sviluppo impressionante, diventando tuttora lo strumento più adatto per analizzare le immagini e trarre dei risultati utili ad altre attività.
Questo metodo di deep learning purtroppo non è esente da problemi: per effettuare un addestramento sufficiente ad effettuare una segmentazione semantica, è necessaria una grande quantità di dati; ciò aggrava il collo di bottiglia della classificazione supervisionata nell'ottenere dati di addestramento sufficienti. Questo è per fortuna aggirabile utilizzando dati preesistenti o da mappe di provenienza pubblica che però possono presentare alti livelli di rumore.
La sfida affrontata successivamente consiste nel mettere in pratica la segmentazione semantica e verificare se l'addestramento con etichette disponibili può sostituire una parte parziale dello sforzo di etichettatura manuale, ottenendo comunque risultati sufficienti; ciò si traduce nel ottenere prestazioni soddisfacenti con uno sforzo di annotazione manuale significativamente inferiore. Va detto che tali dati elaborati conterranno inevitabilmente degli errori significativi, ma in compenso ne sono disponibili quantità virtualmente illimitate in gran parte del mondo.
Essendo questa una questione di mappatura stradale, il passo fondamentale è assegnare una classe a ciascun pixel, ossia convertire i dati grezzi dell'immagine in una mappa semantica significativa (che può essere poi ulteriormente elaborata con tecniche di vettorializzazione). Questo studio è così guidato dalle seguenti ipotesi:

- l'enorme volume di dati di addestramento potrebbe compensare la minore accuratezza (se usato con un metodo di apprendimento robusto);

- la grande varietà presente in set di allenamento molto grandi potrebbe potenzialmente migliorare la capacità del classificatore;

- anche se non sono disponibili dati di addestramento di alta qualità, il grande il volume di dati di addestramento addizionali potrebbe potenzialmente migliorare la classificazione;

- se i dati di addestramento su larga scala e di bassa precisione aiutano, allora è consentibile sostituire una grande porzione dei dati di alta qualità annotati manualmente.

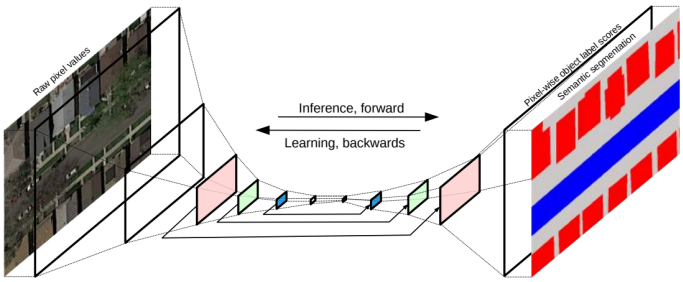

Figura 1.4:
Input e risultato della segmentazione semantica

L'apprendimento automatico mira quindi ad apprendere le regole di classificazione direttamente dai dati. Per l'analisi precisa dei pixel, come evidenza locale, i classificatori convenzionali sono aiutati fornendo loro delle priorità per alcuni pixel, semplici combinazioni aritmetiche come gli indici di vegetazione, diverse statistiche o filtri che descrivono la texture locale dell'immagine. Un'alternativa è quella di precalcolare un insieme ampio e ridondante di caratteristiche locali per l'addestramento e lasciare che un classificatore discriminante selezioni il sottoinsieme ottimale per il compito.
Le architetture di rete all'avanguardia presentano strati di filtri locali e quindi ampi campi recettivi nei livelli profondi della rete neurale, il che rende possibile l'apprendimento di complessi locali direttamente dai dati delle immagini. Inoltre un'importante proprietà delle reti neurali convoluzionali profonde è che sia l'addestramento che l'inferenza sono facilmente parallelizzabili, specialmente sulle GPU, e quindi scalabili a milioni di immagini di addestramento e test.

1.4 Loss function

Focalizzandoci ora sull'argomento principale di questa tesi, va detto che una loss function generica, in qualsiasi modello essa venga usata, è di fondamentale importanza. Essa viene sfruttata in particolare per ottimizzare il modello e affinare i risultati, nelle modalità che dipendono da scopo a scopo e per i quali esistono diverse tipologie di loss function. Tecnicamente, una loss function è una funzione matematica che mappa un evento o valori di una o più variabili che rappresentano concettualmente un costo; viene così usata per valutare la qualità con cui l'algoritmo modella i set dei dati. Essa alla fine dei calcoli esprime un valore che deve essere decrescente durante il training, cosa che indica che il modello usato è buono, altrimenti significa che l'apprendimento non sta dando i risultati voluti.
Nel nostro caso specifico, avendo un problema di classificazione binaria, si va ad utilizzare GapLoss, una loss function che analizza una matrice di pixel derivata dall'immagine processata dalla rete neurale, per poi individuare determinati elementi critici e dare ad essi più importanza nella mappa dei pesi. I dettagli di questo procedimento verranno trattati prossimamente nella sezione 2.2.

Capitolo 2

Dataset utilizzato e metodi

In questo capitolo verrà illustrato il dataset utilizzato per effettuare i test finali. Nella sezione 2.2 inoltre sarà illustrata la loss function GapLoss in versione standard, argomento centrale di questa tesi, i passi fondamentali spiegati con i rispettivi ragionamenti e lo pseudo codice. Nella sezione 2.3 vale lo stesso per la versione alternativa e proposta dal sottoscritto. Infine nella sezione 2.4 sono mostrate tutte le metriche utilizzate per valutare i risultati finali dei test.

2.1 Massachusetts Roads Dataset

Il Massachusetts Roads Dataset è un set di foto satellitari ad accesso libero, utilizzato come set sperimentale. Esso comprende l'intero stato di Massachusetts negli USA ed è suddiviso in:

- training set composto da 1108 immagini;
- validation set composto da 14 immagini;
- test set composto da 49 immagini.

In ogni set sono ovviamente comprese anche le immagini maschere per ogni rispettiva immagine satellitare. Ogni immagine è in formato TIFF a 3 canali RGB, risoluzione di 1500 $\times$ 1500 pixel, risoluzione spaziale di 1 m e 24 bit di profondità. Le immagini maschere relative sono invece in formato TIF con 96 dpi e profondità 8 bit.
A causa dei limiti della capacità di elaborazione dell'hardware a disposizione, i test preliminari con GapLoss, svolti per la scrittura e aggiustamento del

codice, sono stati effettuati con un dataset ridotto estratto da questo: precisamente, le immagini originali del training set con risoluzione 1500x1500 sono state partizionate in 9 parti da 512x512 (con un leggero overlap tra loro); poi tra le 9972 immagini ottenute, ne sono state selezionate 248 come training set ridotto; successivamente è stata eseguita l'operazione analoga anche per validation set e test set originali, ottenendo da questi un totale di 567 immagini, di cui 55 scelte per il test set ridotto finale, ignorando la creazione di un validation set dati i numero ridotti. Per mantenere la coerenza, le immagini sono state integrate con valore 255 e infine salvate con formato PNG a 24 bit, mentre le maschere, sempre dopo essere partizionate e estratte in egual numero come detto precedentemente, sono state binarizzate, in modo che ognuna abbia un bit e formato BMP.
Per comodità di calcolo di GapLoss, ed essendo quello trattato un problema di classificazione binaria, si è deciso che il valore con cui viene etichettato un pixel è 1 se viene riconosciuto come "strada" (positivo), altrimenti l'etichetta è 0, se non risulta riconosciuto come "strada" (negativo). Da questa decisione, deriva il formato delle immagini maschere, in cui le strade sono caratterizzate da pixel bianchi (1), al contrario dello sfondo in cui sono neri (0).
Rispetto ad altri set di dati satellitari, come il set di dati ISPRS Vaihingen, la quantità di dati del Massachusetts è significativamente maggiore. Inoltre, gli stessi training set e test set potrebbero escludere le differenze nelle metriche di valutazione causate dal training set; questo fa sì che ricercatori diversi possano confrontare i risultati delle loro ricerche.
Sulla base di questi fattori e per i limiti pratici e temporali, la presente tesi ha sfruttato per i test finali il dataset ridotto per l'addestramento dei modelli, come illustrato nella sezione 3.1.

2.2 GapLoss standard e metodologia

Quando una generica rete di segmentazione con una loss function viene usata allo scopo di estrarre strade, si ha una difficoltà che dipende da: l'addestramento della rete, cosa la loss function considera da valutare e come le strade possono apparire sconnesse, in particolare nelle intersezioni o sezioni di esse nascoste da ostacoli. La scarsa accuratezza della definizione di esse può dare problemi significativi qualora venissero usate in applicazioni successive condizionate dall'accuratezza dei dati. Vediamo quindi un'analisi dei problemi comuni.
Come mostrato dalle figure sottostanti:

- l'immagine 2.1 rappresenta la fotografia data come input da elaborare;
- l'immagine 2.2 rappresenta la rispettiva maschera, ovvero il risultato che si vuole ottenere;
- l'immagine 2.3 è invece il risultato della segmentazione effettuata della rete generica di segmentazione e della loss function.

Figura 2.1:
Immagine input

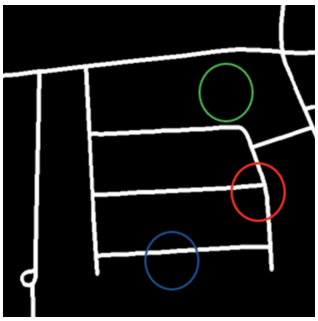

Figura 2.2:
Output obbiettivo

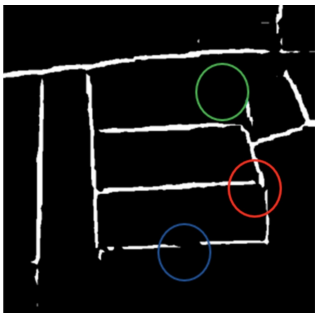

Figura 2.3:
Immagine output

Nell'immagine 2.3, si possono notare come esistono molte interruzioni che non dovrebbero comparire e spesso causate da ostacoli. Per interferenza spesso degli alberi, la rete di segmentazione classifica i pixel, rappresentanti la strada, come se non lo fossero, ad esempio come indicato nel cerchio blu in figura 2.3. Poiché la maggior parte dei pixel che compongono le strade creano forme lineari, la rete di segmentazione è stata addestrata per valutare solo tratti di pixel rettilinei; per questo motivo, le intersezioni tra strade rette, come quella indicata nel cerchio rosso in figura 2.3 vengono ignorate, non essendo delle forme lineari, causando delle interruzioni non volute.
Inoltre non è possibile predire direttamente un'immagine più accurata usando un algoritmo topologico per eliminare queste interruzioni, dato che nella realtà le strade possono essere anche disconnesse a volte, come si nota nel cerchio verde in figura 2.2.
La ragione per cui una strada lineare risulta sconnessa dipende da alcuni pixel valutati erroneamente e che indicano una rottura della continuità stradale, cosa che si vuole evitare. 10 pixel, o alcune volte anche meno, possono indicare un'interruzione dipendente dalla larghezza della strada e dalla risoluzione dell'immagine. Se i pixel valutati erroneamente riguardano una porzione insignificante dell'intera immagine, il modello della rete generica di segmentazione e la loss function, non riesce sufficientemente a catturarne le caratteristiche durante la fase di addestramento. Per rendere la rete più

sensibile a questi pixel valutati erroneamente che evidenziano un'interruzione non reale, il loro peso viene incrementato nella loss function. Se invece questi pixel sono errori effettivi, vengono successivamente corretti durante l'addestramento della rete deep learning e il loro valore viene decrementato nella loss function, al fine di migliorare la connettività e l'accuratezza della predizione stradale.
Come mostrato nelle figure sottostanti:

- l'immagine 2.4 rappresenta lo schema binario della segmentazione risultante;
- l'immagine 2.5 evidenzia i punti finali di ogni tratto stradale;
- l'immagine 2.6 si focalizza nei punti finali e sulla sovrapposizione tra i loro intorni.

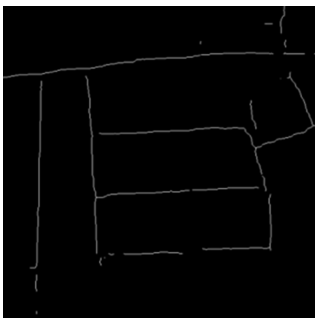

Figura 2.4:
Schema binario

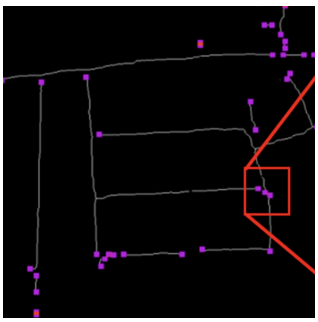

Figura 2.5:
Punti finali evidenziati

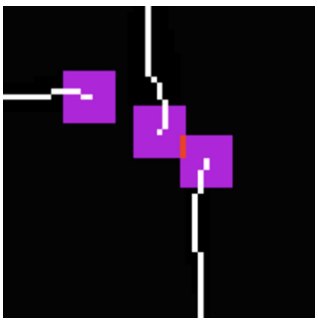

Figura 2.6:
Interruzioni

Al fine di migliorare la predizione di questi pixel mal giudicati, è necessario prima identificarli; il metodo di calcolo però 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, l'addestramento della rete su computer con hardware più vecchi o poco prestanti. Un modo per identificare i pixel errati che causano la discontinuità stradale, consiste nel focalizzarsi sui punti finali dei segmenti stradali. A questo fine, è stata estratta la linea direzionale di ogni strada, come mostrato nella figura 2.4. Successivamente, è stato determinato il pixel finale di ciascuna linea e aggiunto attorno ad esso un buffer per evidenziare l'intorno in cui la rete dovrebbe prestare maggiore attenzione, come mostrato in figura 2.5. I pixel fucsia indicano dove deve essere aumentato il peso, mentre quelli rossi indicano dove due zone di buffer, dei rispettivi due punti finali, si sovrappongono per vicinanza; il loro peso viene così raddoppiato con queste sovrapposizioni,

come mostrato nell'immagine ingrandita 2.6. Questi pixel rossi, essendo più lontani dalla strada prevista rispetto ai pixel fucsia, sono più inclini a essere giudicati erroneamente, per questo è necessario che abbiano un peso raddoppiato per superare l'errore di classificazione.

I passi dell'algoritmo della loss function GapLoss sono i seguenti:

1. Come prima cosa, i valori predetti di tutti i pixel vengono elaborati usando una funzione softmax per acquisire la mappa L della cross-entropy; i valori predetti vengono così convertiti in sistema binario per ottenere l'immagine A, in cui ogni pixel y_i è il valore predetto del pixel i dopo la funzione softmax che viene convertito in binario con soglia 0.5;

2. Dall'immagine binaria A viene poi estratto lo schema binario dell'immagine B, come mostrato in figura 2.4;

3. Considerando ogni pixel dell'immagine B, se esiste solo un pixel con valore 1 negli otto vicini adiacenti, allora il pixel in questione viene considerato come punto finale, quindi il rispettivo valore rimane 1, mentre quello dei pixel adiacenti diventa 0, ottenendo così l'immagine C;

4. Considerando ogni pixel dell'immagine C, se esiste un punto finale in una porzione di 9x9 pixel, il peso di tale pixel viene settato come K (si è deciso K=3 per tutti i test, dati i migliori risultati e per evitare ripetitive esecuzioni); se ci sono 2 punti finali, il pesò è invece 2K, o 3K nel caso di 3 punti finali, o così via per gli altri casi. Se invece nella porzione 9x9 non viene rilevato alcun punto finale, il peso del pixel viene settato come 1. Alla fine di questo procedimento si ottiene una matrice W di tutti i pesi;

5. Ogni pixel della mappa dei pesi W viene poi moltiplicato per il corrispondente valore della mappa cross-entropy L. Dopodiche viene calcolata la media per acquisire il valore GapLoss=f(WxL), dove L è il valore della mappa cross-entropy iniziale, f è la media e W la mappa dei pesi sopra ottenuta.

Lo pseudo codice è il seguente:

Algorithm 1 GapLoss classica

Input: predizione dell'immagine ottenuta dalla rete
Output: valore della loss function

Elaborazione input softmax per acquisire mappa L cross-entropy
Binarizzazione dell'input per ottenere A
Estrazione dello schema di A per ottenere B
Generazione matrice di zeri C con stessa dimensione di B
for each pixel in B **do**
    if esiste un solo pixel=1 negli 8 adiacenti → pixel è punto finale **then**
        il corrispondente pixel in C viene settato a 1
    end if
end for
generazione matrice W di soli 1 con stessa dimensione di B
for each pixel in C **do**
    calcola N come somma della porzione 9x9 (N=punti finali del buffer)
    if N!=0 **then**
        setta come KxN il corrispondente pixel in W (K iperparametro)
    end if
end for
return media di WxL

2.3 GapLoss alternativa e metodologia

In questa tesi, oltre alla versione GapLoss classica, viene proposta una versione alternativa che mira a migliorare il risultato. La versione standard ha l'obiettivo di unire i tratti di strada interrotta la cui distanza è inferiore o uguale a 9 pixel (vedi riga 12 dello pseudocodice). Il problema nasce da qui: per molte immagini, in alcuni tratti le interruzioni possono risultare più grandi, nell'ordine di decine di pixel o dimensione che varia da caso a caso. In questa versione viene per questo ignorata la vicinanza tra due end point (estremi dell'interruzione) di massimo 9 pixel.
Successivamente, per ragionamento logico è stata sfruttata un'informazione topologica della rete stradale: per quanto approssimativo sia affermare ciò, solitamente i percorsi tendono ad avere di base una direzione verticale, orizzontale, diagonale crescente o decrescente (e non solo, per questo viene fatta questa semplificazione).

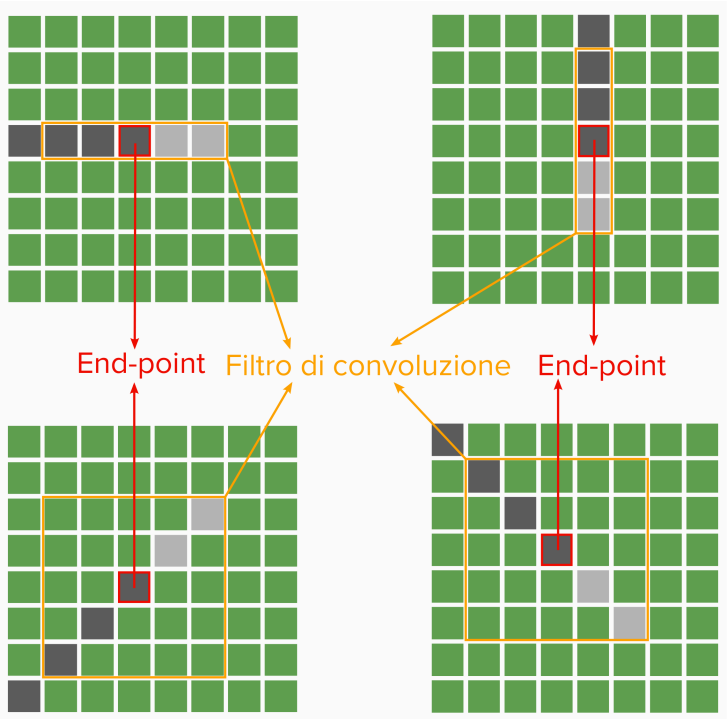

Figura 2.7:
4 direzioni più probabili per i tratti stradali

In più, essendo i tratti stradali spesso dritti e lunghi, è molto probabile che la direzione di questi venga mantenuta per continuità. Sfruttando ciò e utilizzando il procedimento iniziale di GapLoss standard fino all'ottenimento

dello schema binario, si è cercato di identificare i pixel classificabili come finali partendo dai segmenti rettilinei. Per ognuno di questi poi si è cercato di dedurre la direzione della strada precedente, osservando se i precedenti pixel, classificati come strada (in grigio scuro in figura 2.7), vengono da una delle quattro direzioni approssimate dette sopra. Dopodiche, come evidenziato in grigio chiaro in figura 2.7, si sono considerati i pixel successivi, i quali logicamente dovrebbero essere una continuità della strada in quella precisa direzione.
Alla fine, l'errore della funzione Cross-Entropy di questi viene moltiplicato per una costante prefissata a 10 (valore che ha prodotto risultati migliori) e calcolando la media tra gli errori di tutti i pixel, si ottiene il valore loss finale. Con questa modalità, si dà così maggior peso ai pixel che dovrebbero essere una continuità della direzione stradale, producendo tratti più continui e con interruzioni meno presenti.

Da qui derivano i passi dell'algoritmo di GapLoss alternativa (di cui i punti 1), 2) e 8) ereditati dalla versione standard):

1. I valori predetti di tutti i pixel vengono elaborati usando una funzione softmax per acquisire la mappa L della cross-entropy; i valori predetti vengono così convertiti in sistema binario per ottenere l'immagine A, in cui ogni pixel y_i è il valore predetto del pixel i dopo la funzione softmax che viene convertito in binario con soglia 0.5;

2. Dall'immagine binaria A viene poi estratto lo schema binario ottenendo B;

3. Considerando ogni pixel dell'immagine B, si considerano i 4 pixel adiacenti ad esso in senso orizzontale. Di questi si ha una convoluzione con un filtro riga unario di 5 elementi. Dalla matrice risultante ottenuta dalla convoluzione di B e il filtro, vengono considerati solo gli elementi di valore 2, per poi impostarne il valore a 10, come per i 4 pixel adiacenti orizzontalmente. Si ottiene così D, una matrice dei pesi temporanea;

4. Sempre considerando ogni pixel dell'immagine B, si considerano i 4 pixel adiacenti ad esso in senso verticale. Di questi si ha una convoluzione con un filtro colonna unario di 5 elementi. Dalla matrice risultante ottenuta dalla convoluzione di B e il filtro, vengono considerati solo gli elementi di valore 2, per poi impostarne il valore a 10, come per i 4 pixel adiacenti verticalmente. Si ottiene così E, una matrice dei pesi temporanea;

5. Sempre considerando ogni pixel dell'immagine B, si considerano i 4 pixel adiacenti ad esso in senso diagonale crescente. Di questi si ha una convoluzione con un filtro F diagonale unario di 5 elementi. Dalla matrice risultante ottenuta dalla convoluzione di B e il filtro, vengono considerati solo gli elementi di valore 2, per poi impostarne il valore a 10, come per i 4 pixel adiacenti diagonalmente. Si ottiene così F, una matrice dei pesi temporanea;

6. Sempre considerando ogni pixel dell'immagine B, si considerano i 4 pixel adiacenti ad esso in senso diagonale decrescente. Di questi si ha una convoluzione con un filtro F diagonale unario di 5 elementi. Dalla matrice risultante ottenuta dalla convoluzione di B e il filtro, vengono considerati solo gli elementi di valore 2, per poi impostarne il valore a 10, come per i 4 pixel adiacenti diagonalmente. Si ottiene così G, l'ultima matrice dei pesi temporanea;

7. Successivamente si fondono le matrici D, E, F e G ottenendo W la mappa finale dei pesi;

8. Ogni pixel della mappa dei pesi W viene poi moltiplicato per il corrispondente valore della mappa cross-entropy L e la media viene calcolata per acquisire GapLoss=f(WxL), dove L è il valore della mappa cross-entropy iniziale, f è la media e W la mappa dei pesi sopra ottenuta.

Lo pseudo codice è il seguente:

Algorithm 2 GapLoss alternativa

Input: predizione dell'immagine ottenuta dalla rete
Output: valore della loss function

```
Elaborazione input softmax per acquisire mappa L cross-entropy
Binarizzazione dell'input per ottenere A
Estrazione dello schema di A per ottenere B
for each pixel in B do
    convoluzione con filtro orizzontale di 5 elementi
    if pixel==2 then
        impostazione di esso e 4 pixel orizzontalmente adiacenti a 10
    end if
    matrice risultante D
end for
for each pixel in B do
    convoluzione con filtro verticale di 5 elementi
    if pixel==2 then
        impostazione di esso e 4 pixel verticalmente adiacenti a 10
    end if
    matrice risultante E
end for
for each pixel in B do
    convoluzione con filtro diagonale (crescente) di 5 elementi
    if pixel==2 then
        impostazione di esso e 4 pixel diagonalmente adiacenti a 10
    end if
    matrice risultante F
end for
for each pixel in B do
    convoluzione con filtro diagonale (decrescente) di 5 elementi
    if pixel==2 then
        impostazione di esso e 4 pixel diagonalmente adiacenti a 10
    end if
    matrice risultante G
end for
fusione di D, E, F e G in W
return media di WxL
```

2.4 Metriche di valutazione

Essendo questo un problema di classificazione, la rete neurale ha lo scopo di etichettare i pixel come "strada" (positivo) o "non strada" (negativo). Da qui sorgono tutte le possibili valutazioni dato che un pixel può essere valutato come:

- vero positivo (TP) se viene etichettato come "strada" ed è effettivamente un pixel "strada"
- vero negativo (TN) se viene etichettato come "non strada" ed è effettivamente un pixel "non strada"
- falso positivo (FP) se viene etichettato come "strada" quando invece risulta essere un pixel "non strada"
- falso negativo (FN) se viene etichettato come "non strada" quando invece risulta essere un pixel "strada"

Questi sono stati identificati per ogni immagine di test, al fine di calcolare ogni metrica. Per verificare oggettivamente il metodo, per il confronto e l'analisi sono state utilizzate le seguenti metriche di valutazione, in particolare il punteggio medio ottenuto tra tutte le immagini:

- IoU (intersection over union): rappresenta il grado di somiglianza tra output elaborato dalla rete e la verità di base (ground truth). Questa metrica di valutazione prende in considerazione la regione in comune (intersection) che si ha sovrapponendo verità di base con output previsto, per poi calcolare la percentuale di somiglianza effettiva;

$$IoU = \frac{TP}{TP + FN + FP}$$

- Precision: indica quanto è preciso il modello nell'indicare tra i pixel valutati come positivi, quanti di loro siano effettivamente positivi. Questa metrica ha un valore crescente al diminuire dei falsi positivi, per questo è più utile nello scenario in cui un falso positivo è considerato un errore più indesiderato di un falso negativo;

$$Precision = \frac{TP}{TP + FP}$$

- Recall: rappresenta la selettività del modello e indica tra i pixel realmente positivi, quanti di loro vengono valutati come positivi. Questa

metrica ha un valore crescente al diminuire dei falsi negativi, per questo è più utile nello scenario in cui un falso negativo è considerato un errore più indesiderato di un falso positivo;

$$Recall = \frac{TP}{TP + FN}$$

- F score: viene utilizzato quando falsi negativi e falsi positivi sono indesiderabili allo stesso modo. Questa metrica di valutazione combina precision e recall e aumenta all'aumentare di essi. Inoltre, più ha un valore tendente a 1, più indica che il modello è migliore. La formula generica ha una costante b che nel nostro caso sarà b=1 per F1score e b=2 per F2score;

$$Fbscore = (1 + b^2)\frac{precision \cdot recall}{(b^2 \cdot precision) + recall}$$

- Accuracy: misura la frequenza con cui i pixel vengono classificati correttamente, infatti come da formula, si ha un rapporto tra i pixel classificati correttamente (numeratore) e quelli classificati sia erroneamente che correttamente (denominatore). Ovviamente, veri positivi e veri negativi sono i pixel classificati correttamente, mentre falsi positivi e falsi negativi sono i pixel classificati erroneamente.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Capitolo 3

Risultati e confronto

In questo capitolo verranno confrontati i risultati dei test effettuati per i tre modelli, tutti basati sulla rete neurale Resnet18, ma con diversa loss function, ovvero DiceLoss, GapLoss classica e alternativa, rispettivamente per ognuno. Il primo modello con DiceLoss è considerato come obiettivo da raggiungere, essendo questa loss function una delle migliori nell'ambito della segmentazione semantica.
Va tenuto in considerazione che per equità i test sono stati tutti effettuati con learning rate di 0.001, 5 elementi per minibatch e 20 epoche da 49 iterazioni ciascuna, per un totale di 980 iterazioni.

3.1 Test Massachusetts Roads Dataset

Usando per tutti i test il Massachusetts Roads Datset ridotto, si nota come i valori rilevati del secondo e terzo modello, seppur inferiori, siano in linea con quelli obbiettivo del primo. Iou e F1 score calano usando le due versione GapLoss, ma va notato come la versione alternativa proposta migliori i risultati rispetto alla classica; come fattore più importante, l'accuratezza rilevata con GapLoss classica è pari a quella risultata con DiceLoss, la quale viene addirittura superata con la versione alternativa, seppure di qualche decimale.

Modello	IoU	F1 score	Accuracy
Resnet18 - DiceLoss	46.89	62.97	96.41
Resnet18 - GapLoss classica	41.27	57.16	96.41
Resnet18 - GapLoss alternativa	43.37	59.32	96.45

Nelle immagini sottostanti, si ha un esempio di segmentazione della rete stradale in figura 3.1, secondo i tre modelli in ordine nelle figure 3.2, 3.3 e 3.4. Solo osservando attentamente si notano minime differenze. Il risultato di DiceLoss mostra un'esagerazione, ovvero gruppi di pixel falsi positivi dove le strade non sono presenti, come nei perimetri residenziali. Con le due versioni GapLoss invece si nota una maggiore linearità della strada e meno falsi positivi sparsi rispetto al risultato con DiceLoss. In linea di massima i risultati sono pressochè uguali e solo con piccole differenze di dettaglio.

Figura 3.1: Immagine

Figura 3.2: DiceLoss

Figura 3.3: GapLoss classica

Figura 3.4: GapLoss alternativa

Capitolo 4

Conclusioni

Come visto precedentemente dai risultati, anche se essi sono leggermente inferiori a quelli del modello usato come riferimento, si nota come i modelli con le due versioni GapLoss offrano dei risultati abbastanza buoni; inoltre va notato che questi test sono stati effettuati con un dataset troppo piccolo, cosa che non permette di avere risultati ottimali, che si potrebbero invece avere con dei training basati su dataset di più grandi dimensioni, come il Massachusetts Roads Dataset originale o altri, che tra loro avrebbero reso risultati anche diversi a causa della morfologia e della geografia stradale differente da territorio a territorio.
Inizialmente i test finali dovevano basarsi anche sul dataset intero sopracitato e anche altri dataset (Aerial Image Segmentation Dataset e DeepGlobe Roads Dataset), ma in sede finale non sono stati sfruttati per motivi tecnici e pratici.
Infine va precisato che GapLoss può essere usata in qualsiasi modello indipendentemente dalla rete di segmentazione usata, quindi la scelta migliore va decisa in base al compito o allo scopo da raggiungere. Così come proposta, essa ha il potenziale per svolgere un ruolo significativo nella segmentazione semantica stradale nelle immagini satellitari.

Bibliografia

[1] Ajit Jaokar Dan Howarth. *Deep Learning and Computer Vision with CNNs*. Data Science Central, 2019. URL: https://www.myecole.it/biblio/wp-content/uploads/2020/11/6_ML_Deep-Learning.pdf.

[2] Luciano da F. Costa. "What is a Complex Network? (CDT-2)". In: (apr. 2018). DOI: 10.13140/RG.2.2.10450.04804/1.

[3] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[4] Nathalie Japkowicz. "Why question machine learning evaluation methods". In: *AAAI workshop on evaluation methods for machine learning*. 2006, pp. 6–11.

[5] Pascal Kaiser et al. "Learning aerial image segmentation from online maps". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.11 (2017), pp. 6054–6068.

[6] Giuseppe Ridinò. "Deep Learning with MATLAB". In: (2019). URL: https://indico.ict.inaf.it/event/815/contributions/5172/attachments/2405/4818/Deep_LearningINAF_with_MATLAB.pdf.

[7] Shai Ben-David Shai Shalev-Shwartz. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press., 2014.

[8] Wei Yuan e Wenbo Xu. "GapLoss: A Loss Function for Semantic Segmentation of Roads in Remote Sensing Images". In: *Remote Sensing* 14.10 (2022), p. 2422.

Tutti i libri e paper qui citati [8], [6], [1], [3], [2], [5], [7] e [4] sono stati consultati per estendere l'elaborato, mentre le immagini sono state create dal sottoscritto o estratte da [8], [6], [1], [2] e [5].