



SAPIENZA
UNIVERSITÀ DI ROMA

Change Detection su Immagini Satellitari mediante Reti Neurali

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Ingegneria Informatica e Automatica

Alessio Maiola

Matricola 193744

Relatore

Prof. Ciarfuglia Alessandro

Correlatore

Dr. Motoi Ionut

Anno Accademico 2022/2023

Change Detection su Immagini Satellitari mediante Reti Neurali
Tesi di Laurea Triennale. Sapienza Università di Roma

© 2023 Alessio Maiola. Tutti i diritti riservati

Questa tesi è stata composta con L^AT_EX e la classe Sapthesis.

Email dell'autore: maiola.1933744@studenti.uniroma1.it

*Dedicato a
Luigi Ricci*

Ringraziamenti

Voglio ringraziare tutte le persone che mi hanno supportato e sostenuto in questo lungo e faticoso viaggio.

Prima tra tutti mia madre, che per tre anni mi ha spinto a dedicarmi e dare il massimo sullo studio, confortato nei momenti di fatica e sostenuto in ogni maniera possibile.

Voglio dire grazie a mio padre, per avermi accompagnato in un viaggio di scoperta e creatività che solo gli ingegneri possono cogliere, pur con il dovuto distacco tipico di chi ha effettivamente sostenuto "Scienza delle costruzioni".

Voglio dire grazie a mio fratello, perché, anche se il 90% del tempo mi fa arrabbiare e il restante 9% ci ignoriamo, in quell'1% di cose sensate che dice permette di cogliere perle di saggezza e consigli di vita di elevato spessore.

Voglio dire grazie anche e soprattutto agli amici di sempre, con cui ho iniziato e terminato questo viaggio.

Grazie a Maria Penelope, sorella, complice e cugino in un'unica persona.

Grazie a Ludovica, capace di darmi sempre forza e sostenermi nei momenti più difficili.

Grazie a Francesco, con cui sono cresciuto e maturato moltissimo in questi anni.

Grazie a Ilaria, che è riuscita a non dimenticarsi della mia esistenza pur stando a Milano.

Grazie a QXIII, la mia seconda famiglia, sempre presente nei momenti di difficoltà.

Grazie a Marilena, per essere capace di sopportarmi persino quando sono intrattabile.

Dulcis in fundo voglio fare un ringraziamento speciale a delle persone speciali, che mi hanno sempre riempito d'amore e sono la mia principale fonte d'ispirazione.

Grazie nonno Salvatore, nonna Luciana, nonna Rosa, per avermi reso la persona che sono oggi.

Sommario

In questa tesi sono descritti dettagliatamente gli approcci utilizzati per risolvere il compito assegnatomi dal Prof. Ciarfuglia, che desidero ringraziare. L'obiettivo del progetto consiste nell'implementazione di modelli di apprendimento automatico supervisionato, in grado di rilevare i cambiamenti provocati dall'azione umana su coppie di immagini satellitari provviste delle mappe contenenti i cambiamenti effettivamente avvenuti. Per effettuare i rilevamenti viene prodotta una segmentazione delle immagini, classificando ogni singolo pixel come rappresentativo di un cambiamento rilevante o meno. Per lo svolgimento di questo task ho utilizzato reti neurali convoluzionali, testato diverse loss functions, implementato varie tecniche di data augmentation e raccolto i risultati, utilizzando metriche differenti. I modelli implementati hanno ottenuto prestazioni soddisfacenti e la scelta dell'ottimizzazione di diverse metriche ha prodotto varianti eterogenee, maggiormente robuste o più accurate, utilizzabili per vari scopi.

Indice

1 Introduzione al task	1
1.1 Analisi del dataset	1
1.1.1 Il satellite Sentinel-2	1
1.1.2 Le difficoltà affrontate	2
1.2 Suddivisione e manipolazione del dataset	2
1.2.1 Train Set, Validation Set, Test Set	3
2 Reti Neurali	4
2.1 Introduzione	4
2.1.1 Percettrone	4
2.1.2 Chain Rule	4
2.1.3 Overfitting	6
2.2 Reti Neurali Convoluzionali	7
2.2.1 Criticità dell'applicazione delle reti fully-connected alle immagini	7
2.2.2 Struttura delle reti neurali convoluzionali	7
2.2.3 Autoencoder	9
2.3 Le architetture utilizzate	9
2.3.1 U-Net	10
2.3.2 L'architettura utilizzata	10
2.3.3 Early Fusion e Rete Siamese	13
3 Metodologie Utilizzate	14
3.1 Loss Function	14
3.1.1 Cross-Entropy Loss	14
3.1.2 Lo sbilanciamento del dataset	15
3.1.3 Focal Loss	16
3.2 L'algoritmo di ottimizzazione	17
3.2.1 Gradient Descent	17
3.2.2 Criticità dell'algoritmo di discesa del gradiente	19
3.2.3 Gradient Descent with Momentum, Adagrad, RMSProp	20
3.2.4 Adam	21
3.3 Data Augmentation	22
3.3.1 Rotazione e ribaltamento	23
3.3.2 Salt-and-pepper noise	25
3.3.3 Blurring and sharpening	25

3.3.4	Distorsione Elastica	27
3.3.5	Ulteriori proposte	27
3.4	Ulteriori metodologie per mitigare l'overfitting	28
4	Risultati	29
4.1	Metriche utilizzate	29
4.1.1	Loss e accuratezza	30
4.1.2	Accuratezza per classe, precisione, recupero e F1 score	30
4.2	Risultati Raccolti	31
4.2.1	Minimizzazione della loss	32
4.2.2	Massimizzazione dell'F1 score	33
4.2.3	Differenze tra le predizioni dei modelli	34
4.3	Analisi dei risultati	35
4.3.1	Analisi dei fallimenti	35
4.3.2	L'effetto della Focal Loss	38
4.4	Conclusione	38
	Bibliografia	40

Capitolo 1

Introduzione al task

Il compito assegnatomi per la redazione di questa tesi consiste nella segmentazione di immagini satellitari appartenenti all’Onera Satellite Change Detection dataset¹. Per affrontare questo compito occorre classificare ogni pixel della coppia di immagini contenuta nel dataset, per stabilire se esso rappresenti o meno un cambiamento rilevante.

1.1 Analisi del dataset

L’OSCD dataset è un insieme di dati composto da coppie di immagini satellitari acquisite dal satellite Sentinel-2. Le coppie sono costituite da fotografie della stessa area geografica in momenti differenti. Questo dataset è particolarmente interessante poiché fornisce informazioni sulla presenza di cambiamenti nell’ambiente osservato.

Le dimensioni delle immagini non sono fisse, ma variano per ogni sample², pur rimanendo sempre affini a 600x600 pixel.

Ogni coppia di immagini viene fornita insieme a una mappa di ground truth, che indica per ciascun pixel se si è verificato un cambiamento rilevante o meno, come si può vedere dalla figura 1.1. È importante sottolineare che il focus di questo dataset è sul riconoscimento dei cambiamenti causati da attività umane, come la costruzione di edifici o strade, mentre i cambiamenti naturali, ad esempio le variazioni di colore nei campi coltivati, non sono di interesse per questo task. Per questa ragione, gli ambienti proposti nelle immagini date in input alla rete sono unicamente urbani.

L’obiettivo assegnatomi è quello di addestrare modelli di apprendimento automatico, come reti neurali convoluzionali, per riconoscere automaticamente i cambiamenti nell’ambiente osservato dalle immagini satellitari, sfruttando l’OSCD dataset. Quest’analisi può essere di grande utilità in diversi campi, come l’urbanistica, o la sorveglianza ambientale.

1.1.1 Il satellite Sentinel-2

Il satellite Sentinel-2 rientra in un progetto di osservazione del pianeta, permettendo di acquisire immagini multi-spettrali ad elevata risoluzione spaziale, da 10 a 60

¹Per dataset si intende un insieme di dati raccolti.

²Per sample di un dataset si intende un campione dell’insieme di dati.

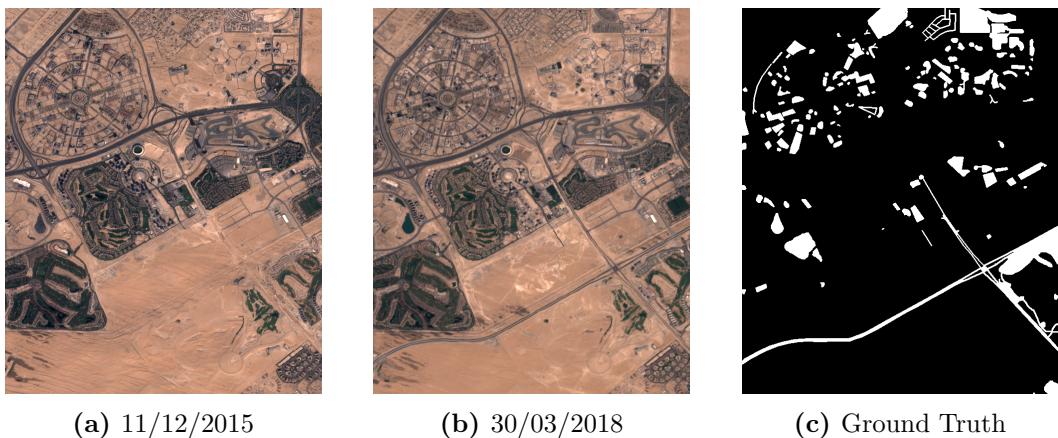


Figura 1.1. Un esempio di sample dell’OSCD Dataset: Dubai

metri, su 13 bande di colore, comprendenti le frequenze della luce visibile e molteplici bande infrarosse dello spettro elettromagnetico.

1.1.2 Le difficoltà affrontate

Nonostante le numerose immagini acquisite, la maggiore difficoltà per questo tipo di compito è data dall’assenza delle necessarie mappe di cambiamento che permettano di utilizzare modelli di apprendimento supervisionato [5].

Un’ulteriore difficoltà è data dal fatto che le immagini presentano 13 bande, a differenza delle immagini tradizionali che hanno solamente 3 bande di colore (bande RGB³). La presenza di 10 bande aggiuntive rende più complesso l’utilizzo degli algoritmi di apprendimento automatico convenzionali, che sono progettati per lavorare con immagini a 3 canali.

Un ulteriore problema riscontrato è la mancanza di precisione nel labeling⁴: in diverse occasioni ampie patches di cambiamento sono state posizionate approssimativamente su variazioni di dimensione inferiore. Ciò ha comportato maggiori difficoltà nella corretta localizzazione dei cambiamenti.

Infine, occorre notare che il lancio del satellite Sentinel-2 è avvenuto a Giugno del 2015, perciò non si dispone di dati antecedenti.

1.2 Suddivisione e manipolazione del dataset

Disponendo, come già precedentemente evidenziato, di un’esigua quantità di dati provvisti di label⁵, ho suddiviso, così come gli autori del paper che affronta questo stesso problema [5], le immagini dell’insieme di addestramento in patches⁶ di

³Per bande RGB si intendono le bande di colore Rosso, Verde e Blu, la cui combinazione permette di ricostruire tutti i colori visibili.

⁴Per labeling si intende la produzione delle mappe di cambiamento, che forniscono le informazioni su quali pixel vadano classificati come cambiati.

⁵Per label si intende la ground truth, cioè la conoscenza a priori di quali pixel identifichino effettivamente un cambiamento rilevante.

⁶Per patch si intende una porzione di immagine di dimensione prestabilita.

dimensione massima 96x96 pixels. Questa tecnica permette di applicare differenti trasformazioni anche a patches appartenenti alla stessa immagine. L'importanza dell'applicazione di varie trasformazioni alle immagini verrà discussa più avanti.

Rispetto al paper appena citato, che è stato preso come riferimento e base per il mio lavoro, si è ritenuto opportuno aggiungere un insieme di validazione, oltre alla suddivisione dei dati proposta.

1.2.1 Train Set, Validation Set, Test Set

Per istruire un modello di apprendimento automatico e valutarne le prestazioni in maniera efficace, occorre suddividere il dataset in tre sottoinsiemi.

Il Train Set è il sottoinsieme di dati sui quali effettivamente la rete apprende, perciò deve essere il più possibile rappresentativo del campione raccolto e grande a sufficienza per permettere alla rete di imparare e generalizzare.

Il Validation Set è un sottoinsieme di dati sui quali la rete non apprende, ma rappresenta dati volutamente accantonati per testare la validità e l'efficacia di modifiche apportate al modello.

Il Test Set è un sottoinsieme di dati non appresi dalla rete, che rappresenta dati effettivamente sconosciuti, sul quale non si può neppure verificare alcun tipo di ipotesi. Questo perché simula i dati assenti nel dataset e serve a valutare le performances della rete su campioni mai visti prima.

Comprendere la necessarietà di questa suddivisione risulta fondamentale per implementare qualsiasi modello di apprendimento automatico. Se si verificasse la correttezza delle proprie ipotesi sui dati di test, si incorrerebbe nell'errore di correlare i miglioramenti in termini di prestazioni del modello ai dati su cui si sta testando la rete, perdendo la totale estraneità del modello predittivo ai dati su cui viene testata.

Gli autori del paper preso come riferimento [5], si limitavano a proporre una suddivisione tra train set e test set. Nel corso delle mie sperimentazioni, per poter ottenere risultati confrontabili con quelli da loro ottenuti, ho mantenuto gli stessi campioni nell'insieme di addestramento, suddividendo la loro proposta di test set in un insieme di validazione e l'insieme che ho effettivamente utilizzato per testare le performances del modello.

Per operare una suddivisione coerente, ho avuto cura che entrambi i sottoinsiemi rimanessero quanto più possibili rappresentativi.

Riporto quindi la suddivisione dei campioni da me utilizzata:

- Train Set: Aguasclaras, Bercy, Bordeaux, Nantes, Paris, Rennes, Saclay East, Abu Dhabi, Cupertino, Pisa, Beihai, Hong Kong, Beirut, Mumbai
- Validation Set: Brasilia, Norcia, Valencia, Las Vegas, Chongqing
- Test Set: Montpellier, Rio, Saclay West, Dubai, Milano

Capitolo 2

Reti Neurali

2.1 Introduzione

Per poter comprendere al meglio le metodologie da me utilizzate, ritengo opportuna una breve introduzione alle reti neurali artificiali.

Le reti neurali artificiali sono uno dei modelli di apprendimento automatico supervisionato (Supervised Learning) e non supervisionato (Unsupervised Learning) maggiormente diffusi e utilizzati. La loro ampia adozione deriva dalla notevole versatilità e capacità di raggiungere ottimi risultati in una vasta gamma di problemi.

L'ispirazione delle reti neurali artificiali è data dalla profonda interconnessione dei neuroni del cervello umano, emulata da parametri numerici che rappresentano i collegamenti tra coppie di neuroni artificiali, detti pesi della rete.

2.1.1 Percettrone

Il modello più semplice di rete neurale, necessario per comprendere tutti gli altri, è il percettrone. Esso permette di approssimare una funzione sulla base di un insieme di dati (Train Set), contenente vari input e l'output desiderato, nel caso di apprendimento supervisionato. Ciò avviene attraverso l'aggiornamento dei pesi tra i livelli che separano quello di input da quello di output, detti livelli nascosti. La rete ottimizza i parametri per minimizzare il gradiente di una funzione che rappresenta l'errore nell'apprendimento (funzione di costo o loss function).

Ogni livello nascosto applica una funzione non lineare per garantire il vantaggio distintivo di una rete neurale profonda. Se la rete si basasse esclusivamente su funzioni lineari, non si otterrebbe alcun beneficio significativo, poiché la combinazione di trasformazioni lineari sarebbe equivalente all'applicazione di una singola trasformazione lineare.

Le reti neurali sono realizzate con l'obiettivo di mappare correttamente degli input che non fanno parte dell'insieme di dati utilizzato per l'apprendimento, detti dati di test (Test Set), massimizzando così la loro capacità di generalizzazione.

2.1.2 Chain Rule

Le formule matematiche utilizzate nelle reti neurali si basano principalmente sull'applicazione della chain rule per le derivate di Newton e Leibniz [10].

Considerando, come mostrato nella figura 2.1:

- X, y : l'input e l'output da predire (ground truth) forniti alla rete,
- n_i, n_j : gli input dei nodi di due livelli contigui della rete, $i \in \{1, \dots, n\}$, $j \in \{n+1, \dots, n+m\}$
- w_{ij} : i parametri che mappano gli input n_i negli output n_j ;
- $\sigma(x) = \frac{1}{1+e^{-x}}$: supponiamo che la funzione di attivazione (non linearità) utilizzata sia la funzione sigmoidea, che si può osservare nella figura 2.8;
- \hat{y} : l'output della rete, supponiamo di usare come funzione di costo (loss function) la Binary Cross Entropy: $BCE(y, \hat{y}) = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$

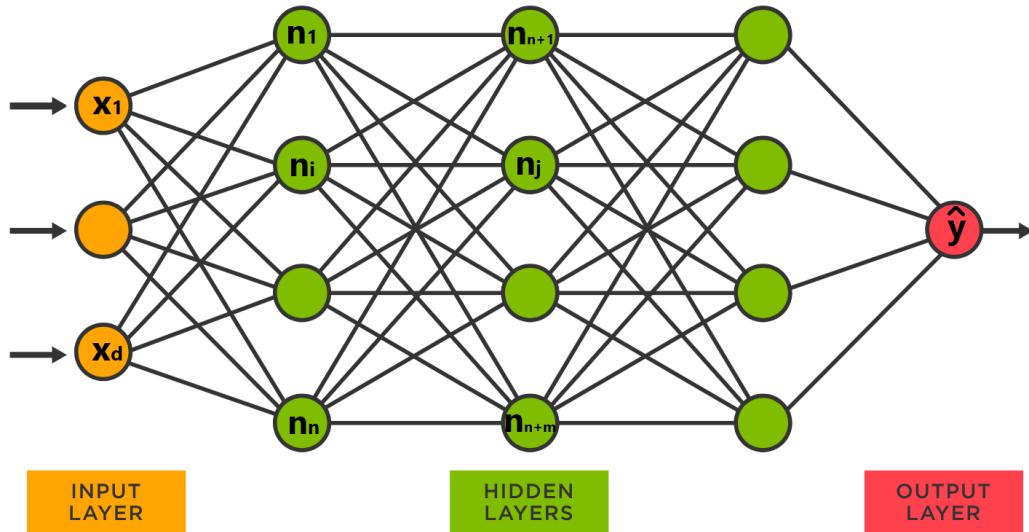


Figura 2.1. Rappresentazione grafica della rete neurale appena descritta

Durante la fase di predizione (Forward Pass), viene attraversata la rete in profondità, calcolando in ogni nodo:

$$\forall j \in \{n+1, \dots, n+m\} : n_j = \sigma \left(\sum_{i=1}^n w_{ij} n_i \right)$$

Fino a raggiungere l'output.

Nel passo di aggiornamento dei parametri per minimizzare l'errore (Backpropagation), si calcola la derivata della funzione di costo sulle predizioni del forward pass, distribuendolo su tutti i nodi mediante la chain rule per le derivate.

Considerando n_k l'input di un nodo della rete:

$$\frac{\partial BCE}{\partial w_{ij}} = \sum_k \frac{\partial BCE}{\partial n_k} \frac{\partial n_k}{\partial w_{ij}}$$

Ricordando che nel passo di forward si calcola $n_j = \sigma(\sum_{j>i} w_{ij} n_i)$, in cui il nodo n_j è l'input di un nodo a profondità maggiore dei nodi con input n_i e definendo:

$$\delta_i = \frac{\partial BCE}{\partial n_i}$$

Si avrà quindi che per i nodi di output:

$$\delta_k = \frac{\partial BCE}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial n_k} = \frac{\hat{y}_k - y_k}{\hat{y}_k(1 - \hat{y}_k)} \sigma'(\hat{y}_k) = \frac{\hat{y}_k - y_k}{\hat{y}_k(1 - \hat{y}_k)} (\hat{y}_k(1 - \hat{y}_k)) = \hat{y}_k - y_k$$

Mentre per i nodi intermedi, se esiste un collegamento dal nodo i -esimo al nodo k -esimo e indicando con \hat{y}_i l'output del nodo i -esimo:

$$\delta_i = \sum_k \frac{\partial BCE}{\partial n_k} \frac{\partial n_k}{\partial n_i} = \sum_k \delta_k \frac{\partial n_k}{\partial n_i} = \sigma'(\hat{y}_i) \sum_k w_{ik} \delta_k$$

Infine, poiché $\frac{\partial n_i}{\partial w_{ij}} = \hat{y}_j$, si avrà:

$$\frac{\partial BCE}{\partial w_{ij}} = \delta_i \hat{y}_j$$

Ora, sapendo calcolare i delta sia per nodi intermedi sia per quelli finali, è possibile propagare il calcolo del gradiente della funzione di costo all'indietro nella rete e utilizzarlo per migliorare i parametri. Esistono molteplici algoritmi di ottimizzazione, che minimizzano la funzione di costo utilizzando il calcolo del gradiente e più avanti descriverò quello che ho scelto e utilizzato.

2.1.3 Overfitting

Il problema più frequente che si riscontra nell'addestramento di una rete neurale è l'overfitting.

Un qualsiasi modello di apprendimento automatico si dice in overfitting quando l'eccessivo adattamento ai dati di apprendimento provoca un peggioramento delle prestazioni nella predizione dei dati di test, come è possibile osservare in figura 2.2.

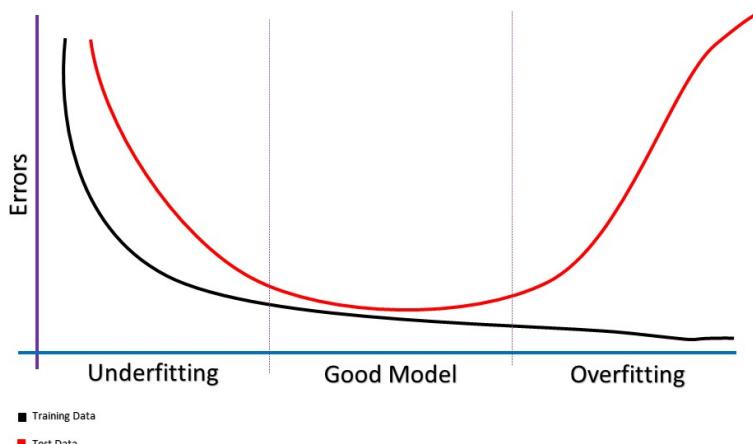


Figura 2.2. Grafico che mostra l'evoluzione di una loss function nel corso delle iterazioni, calcolata sui dati di train e di test, ed evidenzia le situazioni di underfitting e overfitting

Questo fenomeno, comunque indice del corretto funzionamento della rete neurale, può essere provocato dall'eseguità dei dati di apprendimento o dall'eccessiva complessità del modello implementato, ma può essere affrontato e gestito in diversi modi.

Ad esempio, nel caso specifico del task, ho utilizzato tecniche di data augmentation, regolarizzazione ed early stopping, di cui discuterò più avanti.

2.2 Reti Neurali Convoluzionali

2.2.1 Criticità dell'applicazione delle reti fully-connected alle immagini

L'applicazione delle reti fully-connected¹ alle immagini presenta importanti criticità. Le reti neurali descritte fin qui, infatti, presentano come input un vettore di valori che contengono tutte le informazioni (features) per l'apprendimento.

Mediante l'operazione di appiattimento (flattening), che consiste nella trasformazione di un'immagine in un vettore contenente i valori dei suoi pixel, è possibile avere un input valido per questi modelli di apprendimento. Le informazioni sulla località spaziale dei pixel, però, vengono irrimediabilmente perse a causa dell'appiattimento dell'immagine.

Risulta perciò indispensabile introdurre un altro tipo di reti neurali, la cui peculiarità è proprio quella di preserverare anche le informazioni sulla posizione dei pixel all'interno delle immagini: le reti neurali convoluzionali.

2.2.2 Struttura delle reti neurali convoluzionali

Questo tipo di reti neurali sfrutta l'operazione di convoluzione di un'immagine con più filtri per estrarne le features [8]. Per filtro si intende una matrice (solitamente quadrata) sovrapposta ad aree di pixel della stessa dimensione. Ogni pixel viene moltiplicato per il corrispondente elemento della matrice nel filtro e la somma di tutti questi valori costituisce il risultato della convoluzione.

I valori contenuti nei filtri diventano quindi i nuovi parametri della rete neurale, a cui si aggiungono anche diversi iperparametri² strutturali:

- la dimensione del filtro (la dimensione più frequentemente utilizzata è 3x3);
- lo stride, cioè la distanza in pixel di due diverse applicazioni del filtro all'immagine;
- il padding, cioè il numero di pixel da aggiungere all'immagine originale per evitare di decrementarne le dimensioni.

Un esempio, utile a comprendere al meglio questi concetti, è presente nella figura 2.3.

¹Per rete neurale fully-connected si intende una specifica architettura di rete neurale in cui ogni nodo di ogni livello è connesso solamente con tutti i nodi del livello successivo, come nella figura 2.1.

²In un modello di apprendimento automatico gli iperparametri sono valori che condizionano l'apprendimento, anche se non sono direttamente coinvolti nel computo del risultato.

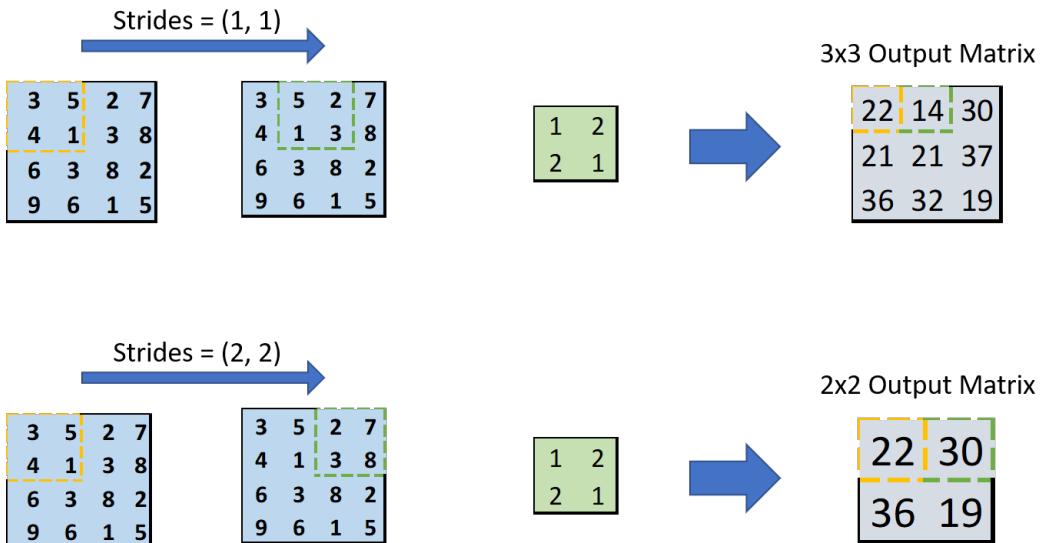


Figura 2.3. Un esempio di convoluzione della stessa immagine con lo stesso filtro, ma con valori di stride differenti e senza l'utilizzo di padding

Inoltre, è comune inserire strati di maxpooling, cioè l'applicazione di filtri che estraggono solo il valore massimo della superficie d'immagine coperta, al fine di permettere alla rete di apprendere solo le features più rilevanti e al contempo di ridurre il rumore. Quest'operazione riduce intuitivamente le dimensioni dell'immagine, poiché per ogni applicazione del filtro viene estratto un solo pixel.

Dopo diversi livelli composti da convoluzioni e maxpooling, è possibile inserire uno strato finale "fully-connected", trasformando in un vettore il risultato delle convoluzioni, altrimenti si avrà una rete neurale "fully-convolutional"³. La sequenza di queste operazioni è descritta anche nella figura 2.4.

Ogni livello nascosto di convoluzioni della rete è descritto da una tripla, costituita dalla dimensione dei filtri e il numero di filtri applicati (anche detto profondità del livello). Ad esempio, un livello descritto dalla tripla 3x3x128 applicherà 128 filtri di dimensione 3x3 all'immagine.

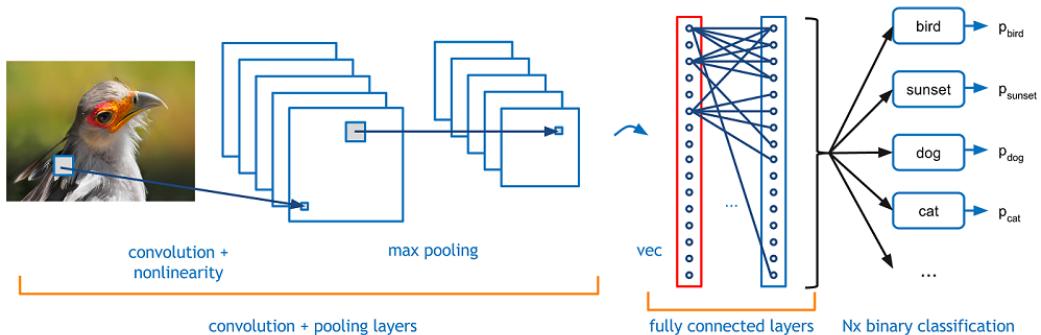


Figura 2.4. La sequenza di operazioni tipica di una rete neurale convoluzionale

³Per rete neurale fully-convolutional si intende una rete neurale che sfrutta solo le convoluzioni, priva quindi di livelli completamente connessi.

2.2.3 Autoencoder

Le reti da me implementate sono reti fully-convolutional e utilizzano come struttura di base quella di un autocodificatore.

L'autoencoder, o autocodificatore, è una rete neurale convoluzionale appartenente ai modelli di apprendimento non supervisionato [4], in quanto ha come obiettivo la ricostruzione dell'input con la minima distorsione possibile. Ciò viene realizzato attraverso una sequenza di strati di convoluzione, non linearità e maxpooling, che riducono le dimensioni dell'immagine. Successivamente, mediante l'utilizzo di convoluzioni trasposte, l'operazione inversa rispetto alla convoluzione e che ingrandisce l'immagine, le dimensioni originali vengono ripristinate e l'immagine viene ricostruita utilizzando le caratteristiche apprese dalla rete. Nella figura è presente una rappresentazione schematica di un autoencoder: 2.5.

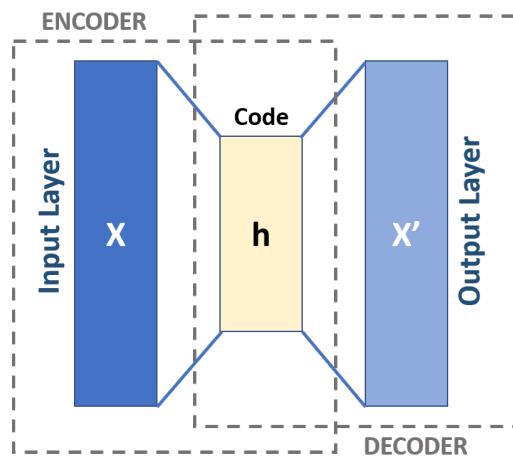


Figura 2.5. Rappresentazione schematica di un autoencoder

Gli autocodificatori hanno diverse applicazioni pratiche, come ad esempio la riduzione del rumore negli input, l'estrazione di features, la generazione sintetica di nuovi dati e sono molto utilizzati nei più disparati ambiti dell'apprendimento automatico.

2.3 Le architetture utilizzate

Dopo una panoramica generale, essenziale a comprendere cos'è una rete neurale e quali sono le reti più efficaci da utilizzare sulle immagini, nonché una breve introduzione agli autoencoder, è il momento di presentare le reti neurali da me implementate e utilizzate per risolvere il compito assegnato.

Occorre tenere a mente che queste reti sono state inizialmente concepite per trattare singole immagini, occorre perciò adattarle a riconoscere e distinguere l'esiguo numero di differenze nelle coppie di immagini.

Bisogna quindi introdurre una specifica variante degli autoencoder: la U-Net, il modello da me scelto e implementato.

2.3.1 U-Net

Una U-Net è una rete neurale che presenta la struttura di base di un autoencoder. La U-Net deve il suo nome alla riconoscibile forma con cui viene rappresentata, come si può vedere dalla figura 2.6. Per migliorare l'apprendimento delle immagini, infatti, la U-Net, perdendo la capacità di ricostruzione dell'input tipica degli autocodificatori, concatena al risultato delle convoluzioni trasposte di upsampling⁴ il risultato delle convoluzioni iniziali.

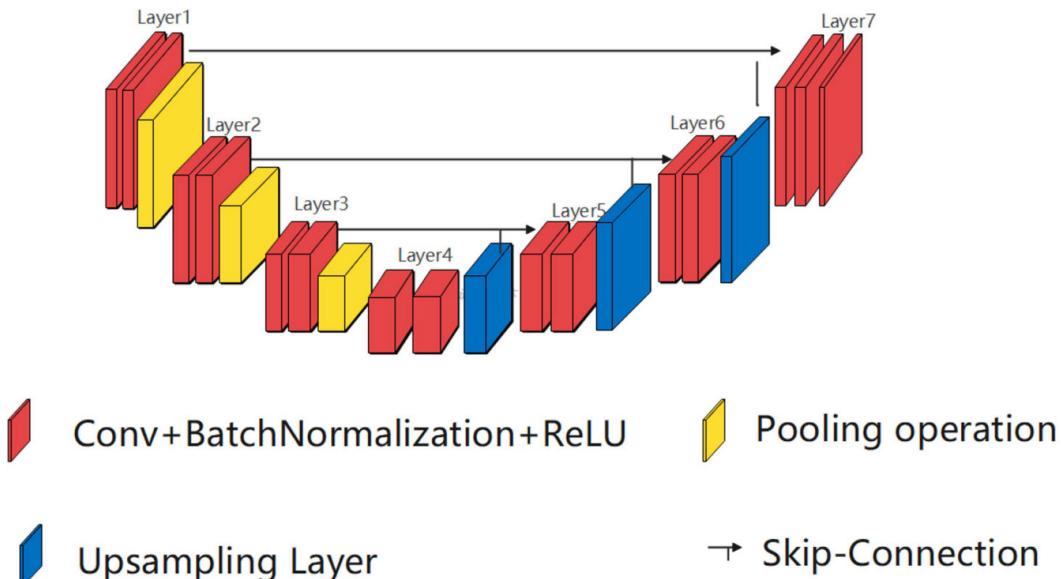


Figura 2.6. Rappresentazione schematica di una U-Net, che utilizza ReLU come funzione di attivazione e la Batch Normalization per migliorare l'apprendimento

In particolare, i risultati delle convoluzioni ai livelli meno profondi vengono concatenati ai risultati delle convoluzioni dei livelli più profondi, così da facilitare l'apprendimento della rete, che non deve concentrarsi sul ricostruire l'intera immagine, ma solo le features non presenti all'interno del risultato delle prime convoluzioni e perciò non ancora apprese.

L'utilizzo di skip-connections⁵, permette alle caratteristiche di basso livello, come quelle locali, di fluire direttamente verso i livelli di decodifica, consentendo una migliore integrazione delle informazioni a diverse scale spaziali e contribuendo a una segmentazione più precisa [9].

2.3.2 L'architettura utilizzata

Per risolvere il task, ho implementato una U-Net fully-convolutional con 9 livelli nascosti. La profondità dei livelli è crescente fino alla fase di upsampling, dove

⁴Per upsampling si intende la fase di decodifica dell'immagine, in cui la dimensione viene aumentata tramite le convoluzioni trasposte.

⁵Per skip-connections si intendono le concatenazioni dei risultati delle convoluzioni meno profonde a quelle delle convoluzioni più profonde. Il termine "skip" permette di intuire il salto di interi livelli provocato da questo collegamento.

diminuisce in maniera simmetrica alla precedente crescita. Dopo ogni convoluzione viene applicata una normalizzazione del batch e una non linearità.

Alle estremità del terzo, del quarto e del quinto strato del modello, per rallentare l'overfitting della rete, ho aggiunto dei livelli di regolarizzazione (Random Dropout Layers [11]). Questo tipo di regolarizzazione consiste nello scartare le informazioni dei canali dei livelli con una probabilità predeterminata.

Il 30% delle informazioni del terzo, del quarto e del quinto livello viene quindi scartato in ogni passo di addestramento del modello. La regolarizzazione tramite random dropout ha come vantaggio di irrobustire la rete ed aumentare la capacità di generalizzazione, permettendo ai livelli nascosti di apprendere features indipendenti attraverso l'aggiunta di rumore stocastico.

Un esempio di applicazione della regolarizzazione che utilizza random dropout si può vedere in figura 2.7.

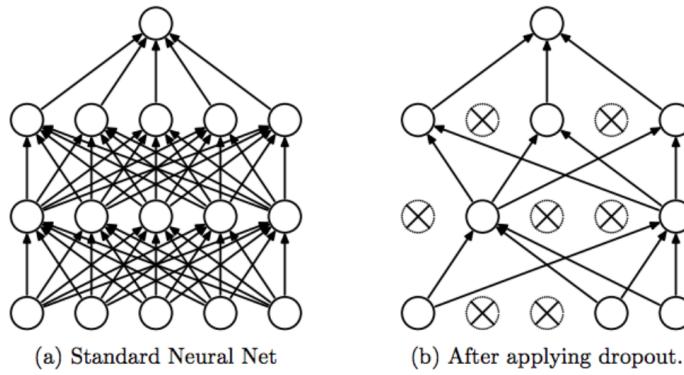


Figura 2.7. L'effetto dell'applicazione della regolarizzazione dropout a una rete fully-connected. Per una rete convoluzionale vengono scartati i risultati di alcune convoluzioni

La non linearità applicata in ogni livello nascosto è il rettificatore o Rectified Linear Unit (ReLU), una scelta molto comune per diversi tipi di reti neurali, che si può osservare in figura 2.8.

$$\text{ReLU}(x) = \max(0, x)$$

Questa funzione di attivazione è molto adoperata poiché gode di interessanti proprietà: è una funzione facile da implementare e computazionalmente molto efficiente, la sua derivata è molto semplice e, essendo costante a tratti, semplifica il passo di backpropagation, limitando inoltre il problema della saturazione del modello di apprendimento automatico.

La saturazione di una rete neurale si può riscontrare utilizzando funzioni di attivazione che presentano valori massimi (o minimi) globali, come la funzione sigmoida, che mappa tutti gli input in valori tra 0 e 1. L'utilizzo di queste funzioni di attivazione può causare il problema noto come "scomparsa del gradiente" (Vanishing Gradient Problem), in cui input molto grandi (o molto piccoli) portano il valore dell'output a uno degli estremi, provocando così l'annullamento del gradiente nella retropropagazione.

Intuitivamente, l'annullamento del gradiente blocca l'apprendimento del modello, l'uso del rettificatore accelera invece la convergenza. Tuttavia, l'utilizzo della

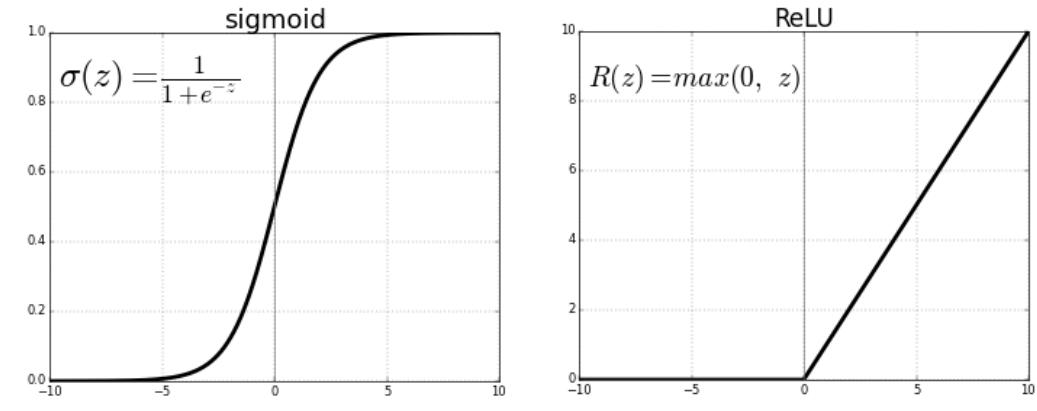


Figura 2.8. Confronto tra la funzione di attivazione sigmoidea, precedentemente citata, e il rettificatore

ReLU non è sempre sufficiente ad addestrare qualsiasi rete neurale, potendo essere soggetta al Dying ReLU Problem: se l'input è negativo per tutti i dati di addestramento, le unità ReLU rimangono inattive, annullandosi e impedendo così l'apprendimento.

Ciò non si verifica nel caso del task in questione. Una possibile soluzione al problema appena citato è quello di utilizzare una funzione di attivazione che non blocchi mai completamente l'apprendimento, come Leaky ReLU o ELU. Una panoramica delle alternative alla semplice ReLU più frequentemente utilizzate è presente in figura 2.9

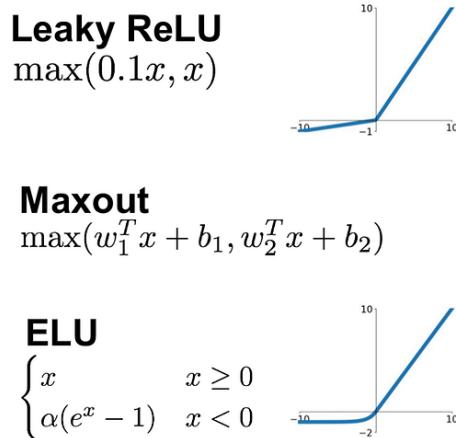


Figura 2.9. Le funzioni di attivazione che mitigano il Dying ReLU Problem

Dopo l'ultima convoluzione, trattandosi di un problema di classificazione binaria, ho testato due possibili soluzioni: l'uso della funzione sigmoidea per le predizioni e l'uso della funzione di attivazione Softmax. Quest'ultima è una funzione di attivazione utilizzabile, al contrario della funzione sigmoidea, non solo per operazioni di classificazione binaria, come nel caso del task, ma con qualsiasi numero di classi.

Mentre per la classificazione con la funzione sigmoidea si utilizza una soglia oltre la quale il campione è considerato appartenente a una delle due classi, la funzione di attivazione Softmax risulta maggiormente versatile. Essa infatti cerca di approssimare la probabilità di appartenenza di ogni campione a ogni classe. Nel passo di predizione, viene selezionata per ogni campione la classe a cui più probabilmente appartiene, cioè il valore massimo predetto della Softmax.

La funzione Softmax prende in input un vettore di dimensione D, pari al numero di classi da predire, e restituisce un vettore di dimensione D, in cui ogni componente rappresenta la probabilità che il sample appartenga alla classe d-esima $\forall d \in \{1, \dots, D\}$.

$$z \in \mathbb{R}^D; \quad \text{Softmax}(z_d) = \frac{e^{z_d}}{\sum_{k=1}^D e^{z_k}}$$

2.3.3 Early Fusion e Rete Siamese

L’architettura appena descritta è stata sfruttata per l’implementazione di due differenti reti neurali. La prima rete neurale che ho implementato utilizza la tecnica dell’Early Fusion delle due immagini in input: le immagini vengono concatenate all’inizio dell’apprendimento, perciò l’input della rete sarà trattato come un’unica immagine contenente 26 canali, anziché una coppia di immagini ognuna caratterizzata da 13 canali. Questa tecnica, molto intuitiva, risulta effettivamente efficace e permette di ottenere risultati soddisfacenti.

La seconda implementazione è una rete siamese che sfrutta la stessa architettura della U-Net precedentemente descritta. I rami di codifica diventano i rami gemelli⁶ della rete siamese, all’interno dei quali le immagini vengono processate separatamente, per ottenerne una rappresentazione latente. Nel passo di upsampling al risultato delle convoluzioni viene concatenata la differenza in valore assoluto dei risultati delle prime convoluzioni sulle due immagini, mediante le skip-connections. Le reti siamesi vengono frequentemente impiegate per rilevare le differenze tra immagini in virtù del particolare utilizzo delle skip-connections. Questi collegamenti integrano le informazioni sulla disparità delle immagini, rendendo frequentemente questo tipo di reti neurali più efficaci delle architetture classiche.

⁶I rami sono detti gemelli, richiamando il paragone che dà il nome alla rete siamese, in quanto condividono gli stessi pesi.

Capitolo 3

Metodologie Utilizzate

Come anticipato nell'introduzione alle reti neurali, nell'implementazione di un modello di apprendimento automatico bisogna curare diversi aspetti:

- la scelta della loss function;
- la scelta dell'algoritmo di ottimizzazione;
- la presenza dell'overfitting.

In questo capitolo, affronterò tutti questi aspetti, presentando e motivando le mie scelte implementative.

3.1 Loss Function

Un aspetto cruciale per l'addestramento di qualsiasi modello di apprendimento automatico è la scelta della funzione di costo (loss function). Questa funzione modella l'errore nelle predizioni della rete e fornisce una misura dell'adeguatezza delle sue prestazioni.

La funzione di costo è progettata per misurare quanto lontane siano le predizioni del modello dai valori di target desiderati. Durante il processo di addestramento l'obiettivo è minimizzare questa discrepanza tra le predizioni e i valori di target reali, forniti in input (ground truth).

La scelta della funzione di costo dipende dal tipo di problema affrontato e dall'obiettivo specifico del modello. Ad esempio, per problemi di regressione in cui l'obiettivo è predire un valore continuo, le funzioni di costo comuni includono l'errore quadratico medio (MSE) e l'errore assoluto medio (MAE).

D'altra parte per problemi di classificazione binaria o multiclasse, in cui l'obiettivo è predire una classe o una categoria, si utilizza spesso l'entropia incrociata (cross-entropy).

3.1.1 Cross-Entropy Loss

La funzione di entropia incrociata è stata inizialmente impiegata nel campo della teoria dell'informazione. Essa sfrutta la definizione di entropia della teoria dell'informazione: l'entropia o autoinformazione di un evento x , emesso da una sorgente

con probabilità $P(x) \in [0, 1]$, è pari a $-\log_2(P(x))$. L'entropia dell'intera distribuzione di probabilità rappresenta la quantità media d'informazione necessaria a descrivere i suoi eventi, cioè la sua incertezza media. Per tale ragione un'elevata entropia corrisponde a una distribuzione di probabilità maggiormente imprevedibile.

$$H(P) = - \sum_{x \in X} P(x) \log_2(P(x))$$

La funzione d'entropia incrociata misura la divergenza tra la distribuzione di probabilità predetta dal modello di apprendimento automatico e quella reale data dalla ground truth. Gli errori commessi dal modello con grande sicurezza, corrispondenti alle predizioni errate con la probabilità più elevata, vengono maggiormente penalizzati. Considerando $t_d \in \{0, 1\}$ la ground truth dell'appartenenza del sample alla classe d-esima e $p_d \in [0, 1]$ la probabilità predetta dal modello di appartenenza del sample alla classe d-esima $\forall d \in \{1, \dots, D\}$:

$$CE\ Loss(p, t) = - \sum_{d=1}^D t_d \log_2 p_d$$

Poiché il compito è di classificazione binaria, verrà utilizzata la versione binaria di questa funzione di costo, detta Binary Cross-Entropy Loss. Considerando t la ground truth (pari a 0 se il pixel non identifica un cambiamento rilevante, altrimenti 1) e p la probabilità che il pixel identifichi un cambiamento rilevante predetto dall'ultimo livello della rete (mediante la funzione Softmax o sigmoidale):

$$BCE\ Loss(p, t) = -(t \log_2 p + (1 - t) \log_2(1 - p))$$

3.1.2 Lo sbilanciamento del dataset

Una delle problematiche affrontate durante lo svolgimento del progetto è lo sbilanciamento del dataset.

Considerando la breve distanza temporale tra le fotografie acquisite e la rilevanza dei soli cambiamenti dovuti all'azione umana, è evidente che la maggior parte dei pixel nelle immagini non rappresenta un cambiamento di interesse, com'è possibile vedere nella figura 3.1.



Figura 3.1. La coppia d'immagini raffigurante la città di Valencia, che mette in risalto l'esiguità dei cambiamenti d'interesse

In particolare, nell'insieme di addestramento meno del 3% dei pixel descrive un cambiamento. Questa significativa disomogeneità dev'essere tenuta in considerazione per la formulazione di una funzione di costo appropriata: se la funzione di costo utilizzata penalizzasse tutti gli errori allo stesso modo, si otterrebbe l'effetto di spingere la rete a predire per ogni pixel la classe maggiormente rappresentata.

Ho quindi calcolato dei pesi da associare agli errori di entrambe le classi, basandomi sulla frequenza di comparsa dei pixel che rappresentano un cambiamento nell'insieme di addestramento. Considerando c il numero di pixel che rappresentano un cambiamento d'interesse nel train set e n il numero totale di pixel contenuti nel train set, la prima funzione di costo da me è utilizzata è la seguente:

$$\text{Balanced BCE Loss}(p, t) = -\left(\frac{n-c}{n} t \log_2 p + \frac{c}{n} (1-t) \log_2(1-p)\right)$$

Questa loss function penalizza gli errori nella predizione dei pixel che indicano un cambiamento, con un contributo proporzionale al numero di pixel che non indicano un cambiamento e viceversa. Così, la loss function si comporterà come se il numero di pixel fosse perfettamente bilanciato, cercando quindi di massimizzare l'accuratezza nelle predizioni di entrambe le classi, trascurando quindi l'esiguità dei cambiamenti effettivi.

3.1.3 Focal Loss

Un'altra loss function che ho utilizzato è la Focal Loss. Questa funzione di costo ha la peculiarità di incrementare l'importanza dei samples difficili da predire, trascurando maggiormente quelli considerati più semplici. L'incertezza della predizione dei samples entra nel calcolo della loss attraverso la probabilità predetta dalla funzione Softmax.

La Focal Loss prevede anche l'introduzione al suo interno di pesi per il bilanciamento del dataset. Considerando, in un problema di classificazione generico, $t_d \in \{0, 1\}$ la ground truth dell'appartenenza del sample alla classe d -esima, $p_d \in [0, 1]$ la probabilità predetta dal modello di appartenenza del sample alla classe d -esima $\forall d \in \{1, \dots, D\}$ e α_d il peso associato alla classe d -esima:

$$FL(p, t) = -\left(\sum_{d=1}^D \alpha_d (1-p_d)^\gamma t_d \log_2 p_d\right)$$

L'iperparametro principale introdotto dalla Focal Loss è il valore γ . Quest'iperparametro definisce il peso da associare all'incertezza nelle predizioni del modello. Una maggiore incertezza incrementa il valore della probabilità che il sample sia stato classificato male e quindi il contributo del sample all'interno della loss function. Viceversa, i sample più semplici, la classificazione dei quali risulta più sicura, presenteranno un contributo inferiore all'interno della loss [7].

L'annullamento del valore di γ corrisponde all'uso della funzione di entropia incrociata bilanciata. Un incremento di γ corrisponde all'incremento del peso associato agli esempi più complessi da predire. Solitamente si utilizzano valori di $\gamma \in [0, 5]$, ma per il task ho empiricamente verificato che la scelta migliore sia

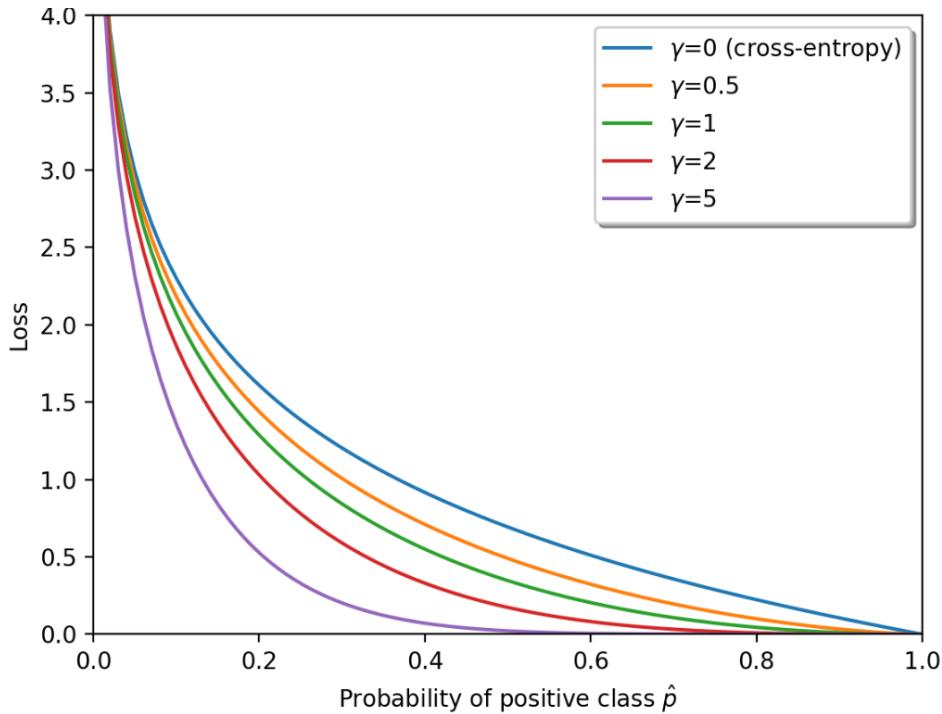


Figura 3.2. La funzione Focal Loss calcolata sulla probabilità della classe positiva (\hat{p}), utilizzando diversi valori per l'iperparametro γ

$\gamma = 2$. I grafici raffiguranti l'evoluzione della Focal Loss al variare del parametro γ sono visibili in figura 3.2.

Considerando quindi nuovamente p la probabilità predetta che il pixel identifichi un cambiamento, t la ground truth, c il numero di pixel che identificano un cambiamento e n il numero di pixel, la Focal Loss da me implementata e utilizzata è la seguente:

$$FL(p, t) = -((1-p)^\gamma \frac{n-c}{n} t \log_2 p + p^\gamma \frac{c}{n} (1-t) \log_2(1-p))$$

3.2 L'algoritmo di ottimizzazione

Esistono diversi algoritmi di ottimizzazione dei parametri di un modello di apprendimento automatico. Per affrontare questo task mi sono affidato, come gli autori del paper di riferimento [5], all'algoritmo di ottimizzazione Adam.

Questa scelta è risultata molto affidabile, poiché Adam combina le caratteristiche positive di tutti gli algoritmi antecedenti [1]. Per poterlo comprendere a fondo occorre quindi analizzare l'evoluzione di una famiglia di algoritmi di apprendimento.

3.2.1 Gradient Descent

Il più semplice algoritmo di apprendimento automatico per l'ottimizzazione dei parametri è l'algoritmo di discesa del gradiente.

L'obiettivo dell'algoritmo è quello di convergere a un minimo globale della funzione di costo, dove il gradiente è nullo, aggiornando iterativamente i parametri del modello di apprendimento automatico nella direzione opposta a quella di massima crescita del gradiente della loss.

Considerando:

- w_t i parametri del modello all'iterazione t -esima, $t \in \{1, \dots, T - 1\}$;
- α il learning rate dell'algoritmo di discesa del gradiente;
- $\nabla Loss(w_t)$ il gradiente della loss rispetto ai parametri del modello w_t .

L'aggiornamento dei parametri viene effettuato applicando la seguente formula:

$$w_{t+1} = w_t - \alpha \nabla Loss(w_t)$$

Il grafico in figura 3.3 illustra il comportamento dell'algoritmo.

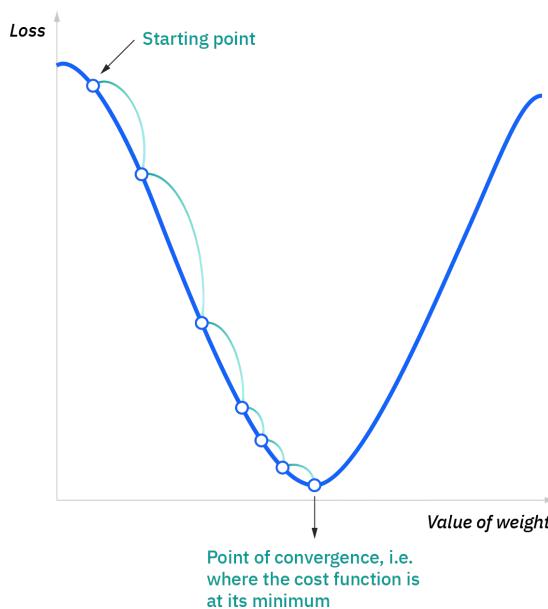


Figura 3.3. Figura che illustra il comportamento dell'algoritmo di discesa del gradiente

L'iperparametro principale dell'algoritmo di discesa del gradiente è il tasso di apprendimento (learning rate, α nella formula). L'aggiornamento dei parametri avviene calcolando il prodotto di questo valore per il gradiente. Per questa ragione il learning rate è l'iperparametro maggiormente interconnesso con l'overfitting e l'underfitting dei modelli di apprendimento automatico.

Un learning rate elevato aumenta la velocità di apprendimento di un modello, ma ne riduce la stabilità, come si può vedere in figura 3.4. Risulta perciò fondamentale non solo scegliere adeguatamente il valore del learning rate, ma anche eventualmente modificarlo nel corso dell'apprendimento, mediante uno scheduler¹.

¹Per scheduler s'intende l'introduzione di un'operazione predeterminata che viene applicata al learning rate in ogni iterazione.

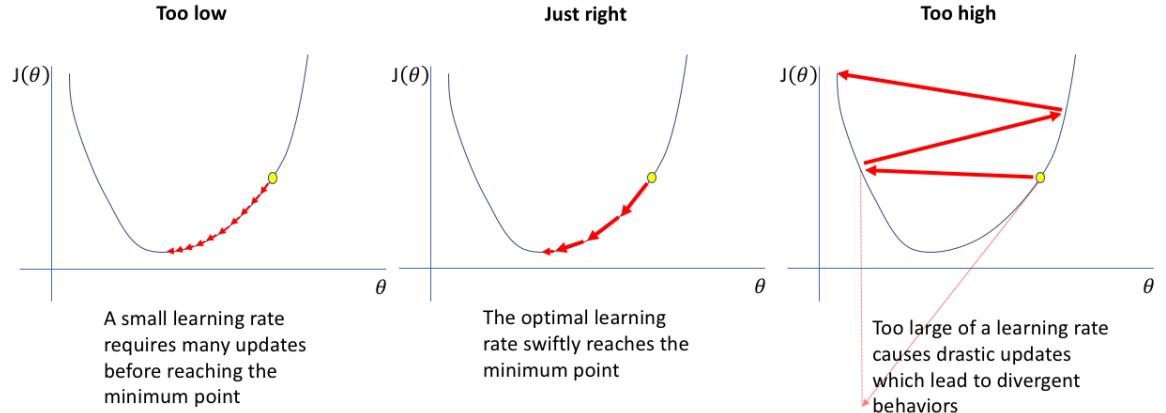


Figura 3.4. Grafico che mostra gli effetti dell'utilizzo di learning rates eccessivamente e non sufficientemente elevati

In particolare ho mantenuto la scelta degli autori del paper di riferimento [5] di utilizzare uno scheduler esponenziale. In ogni epoca di apprendimento della mia rete neurale il learning rate viene ricalcolato, indicando con e l'epoca corrente e con E il numero totale di epoche, come segue:

$$\alpha_{e+1} = 0.95 \cdot \alpha_e, e \in \{1, \dots, E-1\}$$

3.2.2 Criticità dell'algoritmo di discesa del gradiente

Il gradient descent beneficia particolarmente delle funzioni di costo convesse, in quanto presentano un unico minimo globale, come ad esempio la funzione di costo MSE.

Se la funzione di costo utilizzata non è convessa, l'algoritmo di discesa del gradiente potrebbe convergere a minimi locali o punti di sella, presentando anch'essi una derivata nulla. Una funzione di costo ricca di punti di sella o minimi globali risulta perciò molto difficile da ottimizzare con questo algoritmo.

Un esempio di loss function problematica per l'algoritmo di discesa del gradiente è presente nel grafico in figura 3.5.

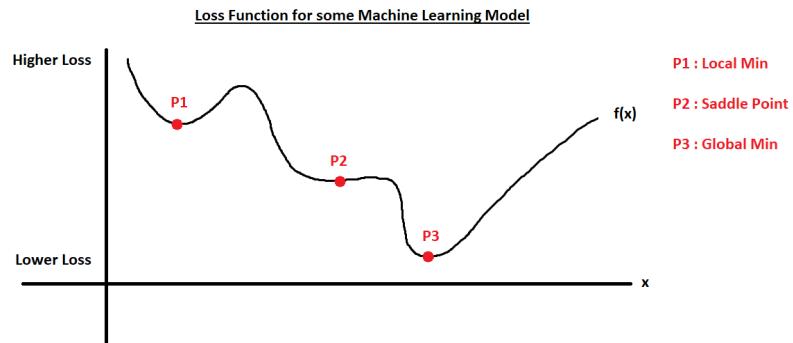


Figura 3.5. Grafico di una loss function che presenta un minimo locale e un punto di sella

3.2.3 Gradient Descent with Momentum, Adagrad, RMSProp

Il gradient descent with momentum è un'estensione dell'algoritmo di discesa del gradiente. L'uso del momento è una tecnica che introduce una componente di "inerzia" nel passo di aggiornamento. Invece di aggiornare i pesi solamente in base al gradiente istantaneo, il momento tiene traccia della media ponderata dei gradienti precedenti, che utilizza per determinare la direzione e la dimensione dell'aggiornamento dei pesi.

L'aggiunta del momento consente di superare alcuni problemi che possono verificarsi con l'applicazione del gradient descent standard. Ad esempio, se la funzione di costo presenta regioni con gradienti molto ripidi o rumore nel calcolo dei gradienti, il momento aiuta a evitare di rimanere bloccati in minimi locali o di oscillare troppo tra i punti nell'aggiornamento dei pesi, fornendo stabilità e controllo nella ricerca dello spazio di parametri durante l'addestramento di un modello di machine learning.

Qui viene descritto l'aggiornamento dei pesi dell'algoritmo, utilizzando la notazione precedentemente introdotta:

$$\begin{aligned} m_{t+1} &= \rho m_t + \nabla Loss(w_t) \\ w_{t+1} &= w_t - \alpha m_{t+1} \end{aligned}$$

Adagrad è un'ulteriore alternativa alla classica implementazione dell'algoritmo di gradient descent. L'idea principale di Adagrad è di adattare il tasso di apprendimento per ciascun parametro in base ai gradienti passati. Invece di utilizzare un tasso di apprendimento globale e costante per tutti i parametri, Adagrad assegna un tasso di apprendimento specifico per ognuno, accumulando il quadrato dei gradienti passati di ciascuno. Questi accumuli vengono utilizzati per scalare il tasso di apprendimento del parametro corrispondente.

L'effetto ottenuto è la riduzione del tasso di apprendimento dei parametri che ricevono gradienti elevati e l'aumento del tasso di apprendimento per quelli che ricevono gradienti più bassi. Considerando G_t il termine di accumulo per la t-esima iterazione di addestramento:

$$\begin{aligned} G_{t+1} &= G_t + (\nabla Loss(w_t))^2 \\ w_{t+1} &= w_t - \frac{\alpha}{\sqrt{G_{t+1}} + \epsilon} \nabla Loss(w_t) \end{aligned}$$

Poiché la funzione d'accumulo è sempre crescente, per addestramenti molto lunghi c'è il rischio che il tasso di apprendimento venga ridotto eccessivamente. Per mitigare questo problema si può utilizzare una variante di Adagrad, detta RMSProp (Root Mean Squared Error Propagation), che aggiorna il termine di accumulo effettuando una media ponderata grazie all'inserimento di un iperparametro $\beta \in [0, 1]$.

$$\begin{aligned} G_{t+1} &= \beta G_t + (1 - \beta)(\nabla Loss(w_t))^2 \\ w_{t+1} &= w_t - \frac{\alpha}{\sqrt{G_{t+1}} + \epsilon} \nabla Loss(w_t) \end{aligned}$$

La combinazione di questi algoritmi ha portato alla creazione dell'algoritmo Adam, che presenta i vantaggi di tutte le varianti introdotte.

3.2.4 Adam

L'algoritmo di ottimizzazione Adam combina l'utilizzo dei momenti del primo e del secondo ordine dei gradienti per regolare dinamicamente il tasso di apprendimento [6].

Rispetto alla classica discesa del gradiente Adam garantisce una maggiore velocità di convergenza, come si può vedere dalla figura 3.6, e una robustezza più elevata all'inizializzazione dei parametri.

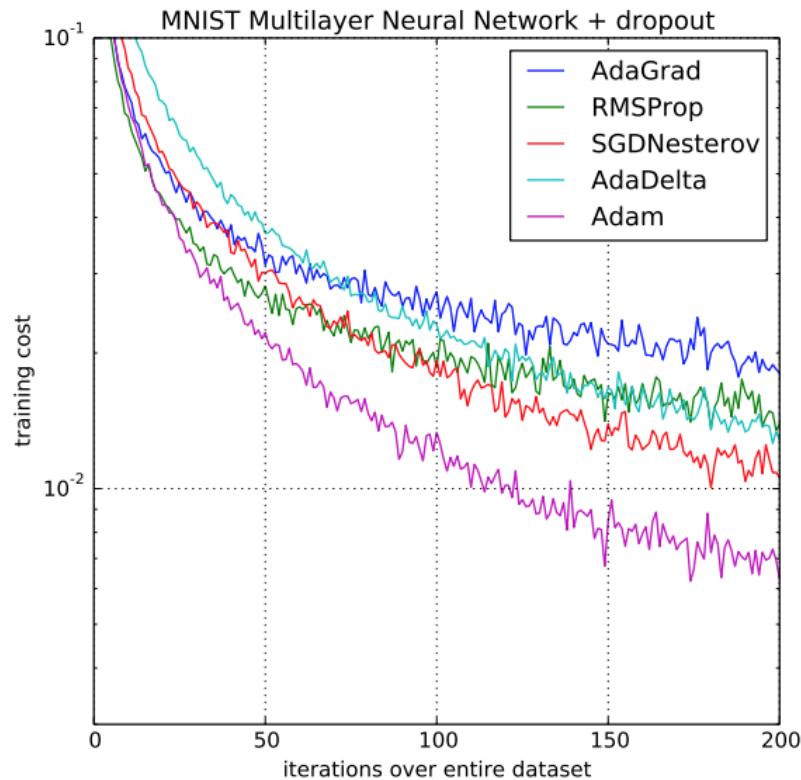


Figura 3.6. Confronto dell'ottimizzazione di una loss function con vari ottimizzatori, per la classificazione del noto dataset MNIST [12]

Inoltre, pur presentando una maggiore quantità di iperparametri, la loro configurazione è di gran lunga meno critica e l'utilizzo dei valori di default risulta spesso efficace.

Ecco quindi la descrizione del funzionamento dell'algoritmo Adam, utilizzando la stessa notazione precedentemente impiegata:

$$\begin{aligned} m_{t+1} &= \rho m_t + (1 - \rho) \nabla Loss(w_t) \\ G_{t+1} &= \beta G_t + (1 - \beta) (\nabla Loss(w_t))^2 \end{aligned}$$

Dopo l'approssimazione dei momenti del primo e del secondo ordine, occorre effettuare l'operazione di unbiasing dei momenti. Le stime iniziali possono infatti essere eccessivamente influenzate dai valori iniziali nulli dei momenti, introducendo

così un bias rilevante. I valori vengono perciò divisi per una correzione basata sul numero di iterazioni eseguite durante l'addestramento.

$$\begin{aligned}\hat{m}_{t+1} &= \frac{m_{t+1}}{1 - \rho^{t+1}} \\ \hat{G}_{t+1} &= \frac{G_{t+1}}{1 - \beta^{t+1}} \\ w_{t+1} &= w_t - \alpha \frac{\hat{m}_{t+1}}{\sqrt{\hat{G}_{t+1}} + \epsilon}\end{aligned}$$

3.3 Data Augmentation

La tecnica principale utilizzata per limitare il problema dell'overfitting è l'aumento dei dati a disposizione. Questa tecnica consiste nell'applicazione di trasformazioni sui dati a disposizione, con il fine di aumentare la varietà dei dati e la robustezza del modello. L'overfitting della rete neurale viene ridotto in quanto una maggiore varietà dei dati di input incrementa la sua capacità di generalizzare e riduce il rischio di bias nell'insieme dei dati di addestramento, poiché le variazioni introdotte tramite le trasformazioni possono aiutare a coprire una gamma più ampia di condizioni e contesti.

Molte trasformazioni utilizzate nella data augmentation sono trasformazioni reali che si possono verificare nelle diverse situazioni di acquisizione delle immagini satellitari. Ad esempio le trasformazioni come la rotazione, la sfocatura, il rumore e il ribaltamento possono simulare le variazioni che si verificano nella cattura delle immagini, a causa del movimento della fotocamera, della sfocatura dell'immagine o della presenza di rumore. Incrementando mediante la data augmentation la varietà dei samples si permette al modello di riconoscere le features d'interesse indipendentemente da questo tipo di variazioni.

Un esempio di applicazione di varie tecniche di data augmentation a un'immagine è visibile in figura 3.7



Figura 3.7. Esempio di applicazione della tecnica di aumento dei dati su un'immagine

Per affrontare il compito assegnatomi ho utilizzato molteplici tecniche di data augmentation. L'overfitting risulta in effetti il problema principale per la risoluzione del task, poiché l'OSCD dataset presenta 24 coppie di immagini, di cui solamente 14 coppie vengono utilizzate per l'addestramento del modello.

In virtù dell'eseguità del dataset ho deciso di suddividere le immagini, le cui dimensioni sono approssimativamente 600x600 pixel, in patches. In particolare, le

patches hanno una dimensione di al più 96x96 pixel e le trasformazioni vengono applicate indipendentemente a ogni coppia di patches ricavata dalle immagini.

Occorre precisare che, lavorando con coppie d'immagini, bisogna decidere volta per volta se la scelta migliore sia l'applicazione delle trasformazioni in maniera indipendente alle due immagini o l'utilizzo della stessa trasformazione sulle patches ricavate da entrambe le acquisizioni.

Inoltre, queste trasformazioni vengono applicate randomicamente alle patches: ciò significa che con una certa probabilità esse non vengono applicate. L'obiettivo è infatti di utilizzare la data augmentation per aumentare la capacità del modello di generalizzare le caratteristiche principali delle immagini indipendentemente dalle variazioni specifiche presenti. L'introduzione di casualità nella scelta delle trasformazioni permette al modello di adattarsi a una vasta gamma di condizioni e combinazioni delle trasformazioni utilizzate, applicate in maniera indipendente l'una dall'altra.

Per presentare le trasformazioni utilizzate, sarà fornita un'illustrazione della loro applicazione al sample della città di Aguasclaras, trascurando per maggiore chiarezza la suddivisione in patches, presentato in figura 3.8.

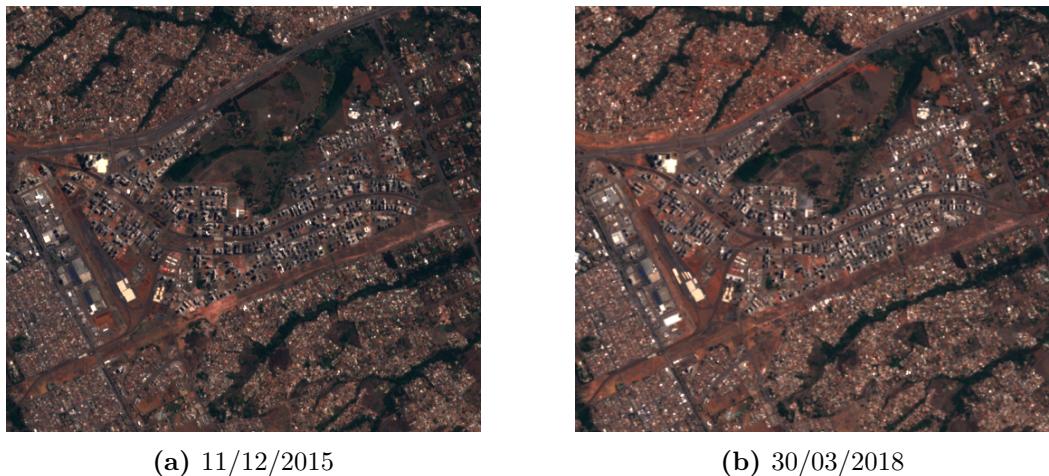
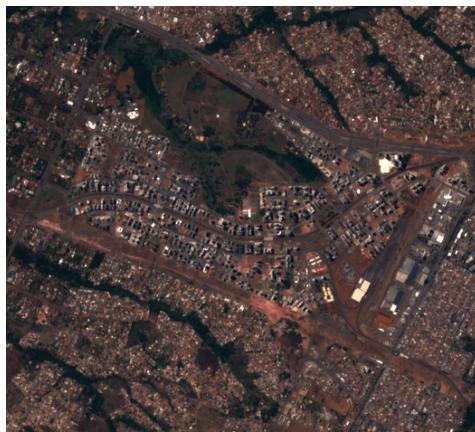


Figura 3.8. La coppia d'immagini del campione raccolto della città di Aguasclaras

3.3.1 Rotazione e ribaltamento

La prima tecnica di data augmentation utilizzata è una scelta molto comune, implementata anche dagli autori del paper di riferimento [5], consiste nel ruotare e ribaltare le patches delle immagini con una certa probabilità.

In particolare, entrambe le patches e la ground truth vengono specchiate con una probabilità del 50%. L'effetto dell'applicazione di questa trasformazione alla coppia di immagini del dataset precedentemente presentata (figura 3.8) è visibile in figura 3.9.



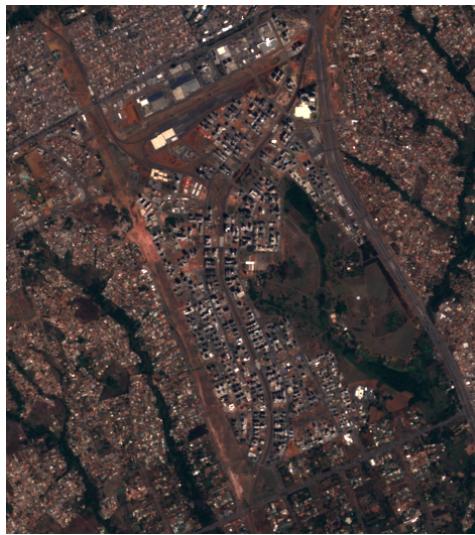
(a) Ribaltamento della figura 3.8a



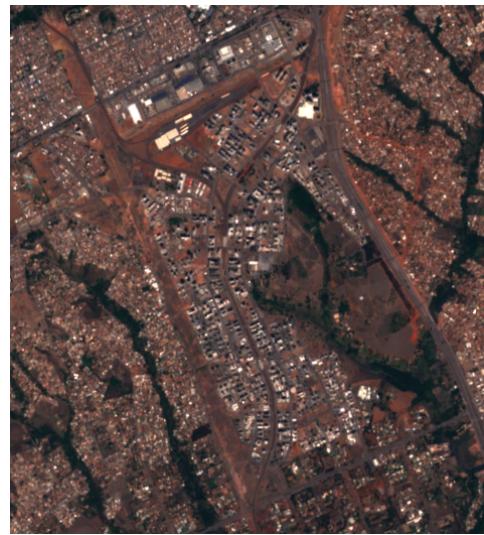
(b) Ribaltamento della figura 3.8b

Figura 3.9. Ribaltamento del sample presentato in figura 3.8

Inoltre, le pateches e la ground truth subiscono una rotazione di un multiplo di 90°, tra 0°e 270°, ciascuno selezionato con una probabilità del 25%. Il risultato dell'applicazione di questa trasformazione al sample in figura 3.8 è presentato in figura 3.10.



(a) Rotazione della figura 3.8a



(b) Rotazione della figura 3.8b

Figura 3.10. Rotazione di 90°del sample presentato in figura 3.8

L'invarianza del modello rispetto alle trasformazioni di rotazione e ribaltamento dell'immagine è fondamentale poiché l'angolazione e il verso di acquisizione dell'immagine dipendono strettamente dalla fotocamera e non dovrebbero influire sulla capacità del modello di rilevare cambiamenti. Ciò consente al modello di essere più flessibile e di adattarsi a una varietà di angolazioni e orientamenti dell'immagine durante l'applicazione pratica, migliorando così la sua capacità di generalizzazione e la sua efficacia nel rilevamento dei cambiamenti rilevanti.

3.3.2 Salt-and-pepper noise

Per simulare disturbi e imperfezioni presenti nelle immagini reali, causati da interferenze o errori di trasmissione durante l'acquisizione delle immagini reali e dall'influenza degli agenti atmosferici, si è rivelato utile aggiungere del rumore alle immagini.

La tecnica di aggiunta di rumore da me utilizzata è quella del "salt-and-pepper noise" [2], che consiste nell'annerire randomicamente i pixel con una certa probabilità. Il valore utilizzato per la probabilità che un pixel venga annerito è il 10%. Perciò un decimo dell'informazione delle due immagini appare corrotta alla rete neurale, aumentandone la robustezza ai possibili disturbi sopracitati.

Il rumore viene applicato ai pixel delle due patches in maniera indipendente, ma non alla ground truth. L'effetto dell'applicazione del rumore al campione in figura 3.8 è visibile in figura 3.11.

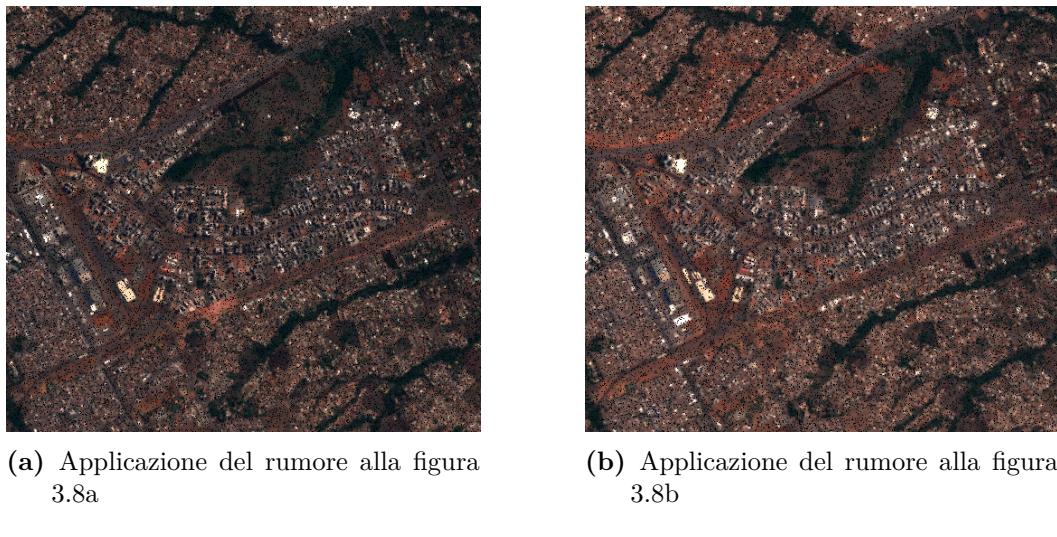


Figura 3.11. Applicazione del rumore al sample presentato in figura 3.8

3.3.3 Blurring and sharpening

L'applicazione di filtri alle immagini è una delle tecniche di data augmentation più diffuse e utilizzate, consentendo di pre-elaborare i dati di input della rete neurale. I due filtri che ho scelto di applicare alle patches delle immagini sono ampiamente utilizzati a causa dell'efficacia dimostrata nell'estrazione di specifiche caratteristiche dalle immagini; la loro utilità è stata documentata in numerosi studi e applicazioni, rendendoli scelte affidabili per l'elaborazione delle immagini, fornendo ai modelli maggiore robustezza al riconoscimento degli oggetti [13].

L'applicazione del filtro di sfocamento gaussiano (Gaussian blur) alle immagini ne riduce i dettagli e i bordi. Permette di diminuire il rumore alle alte frequenze e contribuisce a rendere l'immagine meno suscettibile alle piccole variazioni.

Per mantenere la coerenza visiva i filtri vengono applicati a entrambe le immagini, ma non alla ground truth. L'applicazione del filtro di sfocamento al sample in figura 3.8 è presente in figura 3.12.



(a) Applicazione dello sfocamento alla figura 3.8a

(b) Applicazione dello sfocamento alla figura 3.8b

Figura 3.12. Applicazione dello sfocamento gaussiano al sample presentato in figura 3.8

L'applicazione del filtro di miglioramento (sharpening) ha un effetto opposto rispetto a quello del filtro di sfocamento. Esso enfatizza i bordi e i dettagli delle immagini, rendendoli maggiormente nitidi e permettendo al modello di cogliere variazioni più sottili e le strutture più importanti presenti nelle acquisizioni, agevolando il rilevamento di cambiamenti. Un esempio di applicazione del filtro di sharpening al sample in figura 3.8 è presente in figura 3.13.



(a) Applicazione del miglioramento alla figura 3.8a

(b) Applicazione del miglioramento alla figura 3.8b

Figura 3.13. Applicazione del filtro di miglioramento al sample presentato in figura 3.8

È possibile notare l'overflow di alcuni pixel, ma ciò non si verifica applicando il filtro alle patches di addestramento, poiché le immagini vengono preventivamente normalizzate².

²La normalizzazione dei dati di input facilita l'apprendimento del modello, portando i dati a una distribuzione normale standard, sottraendo la media e dividendo per la deviazione standard, secondo la seguente formula: $Normalized(x) = \frac{x-\mu}{\sigma}$, dove $\mu = mean(x)$ e $\sigma = std(x)$.

Ciascun filtro viene applicato alle patches con una probabilità del 25%.

3.3.4 Distorsione Elastica

La trasformazione più particolare che viene applicata alle patches delle immagini è la distorsione elastica. Questa trasformazione agisce su diversi fattori, permettendo di rallentare di molto l'overfitting. Essa simula distorsioni reali presenti all'interno delle immagini satellitari, come errori di calibrazione o distorsioni ottiche. Inoltre, questo tipo di trasformazione incrementa la robustezza del modello di machine learning alle distorsioni spaziali, migliorando la capacità del modello di riconoscere le variazioni nelle immagini indipendentemente dalla loro posizione precisa nell'immagine e dalla geometria della scena.

L'introduzione di randomicità nell'entità della distorsione e la combinazione con tutte le altre trasformazioni sopracitate incrementa di molto la varietà dei dati e la capacità della rete di generalizzare.

La distorsione elastica viene applicata alle immagini utilizzando una griglia regolare. La deviazione standard di questa griglia rappresenta l'entità della distorsione. Maggiore è la deviazione standard, maggiore sarà l'effetto di distorsione applicato alle immagini. Il valore di questo parametro viene scelto randomicamente nell'intervallo $\{20, \dots, 50\}$, tuttavia essa viene applicata con una probabilità del 50%.

L'introduzione di questa trasformazione si è dimostrata quindi una scelta vincente. Un esempio dell'applicazione della distorsione elastica al sample presentato in figura 3.8 si trova in figura 3.14.



(a) Distorsione elastica della figura 3.8a



(b) Distorsione elastica della figura 3.8b

Figura 3.14. Distorsione elastica del sample presentato in figura 3.8

3.3.5 Ulteriori proposte

Una classe di trasformazioni che non ho avuto modo di introdurre nella data augmentation è quella delle variazioni di colore. Aumentare la robustezza e l'invarianza del modello rispetto a variazioni di luminosità, contrasto e alcune variazioni di colore avrebbe potuto limitare degli errori di predizione che verranno affrontati nell'analisi dei fallimenti.

La presenza di informazioni provenienti da 13 bande rende però estremamente più delicato effettuare questo tipo di operazioni sulle immagini, anche limitandosi solamente ai canali RGB. Modificando i valori di questi canali, si rischia di perdere la coerenza con le informazioni contenute negli altri e con essa l'integrità dei dati di input.

3.4 Ulteriori metodologie per mitigare l'overfitting

Oltre alle già citate tecniche di dropout regularization e data augmentation, occorre introdurre le restanti metodologie utilizzate per affrontare il problema dell'overfitting.

- L2 Regularization: l'uso della regolarizzazione L2 consiste nell'aggiunta alla funzione di costo di un termine proporzionale al quadrato dei parametri utilizzati nel modello, per limitarne la complessità. Considerando w il vettore contenente i parametri del modello e P l'insieme degli indici dei parametri:

$$Cost(w) = Loss(w) + \lambda \sum_{p \in P} w_p^2$$

λ è un iperparametro.

- Early Stopping: questa tecnica consiste nel bloccare l'apprendimento del modello nell'epoca in cui i parametri hanno le migliori performances, calcolate sull'insieme di validazione [3].
- Scelta degli iperparametri: vista la molteplicità degli iperparametri presenti, risultava impossibile testare ogni possibile configurazione. Per questa ragione ho testato diverse inizializzazioni per il learning rate, l'iperparametro più influente sul comportamento di un modello di apprendimento automatico.

Capitolo 4

Risultati

In quest'ultimo capitolo saranno presentati i risultati ottenuti dalle reti neurali descritte, utilizzando le metodologie precedentemente spiegate.

Occorre però precisare come sono stati raccolti i dati e quali metriche sono state utilizzate in fase di testing.

4.1 Metriche utilizzate

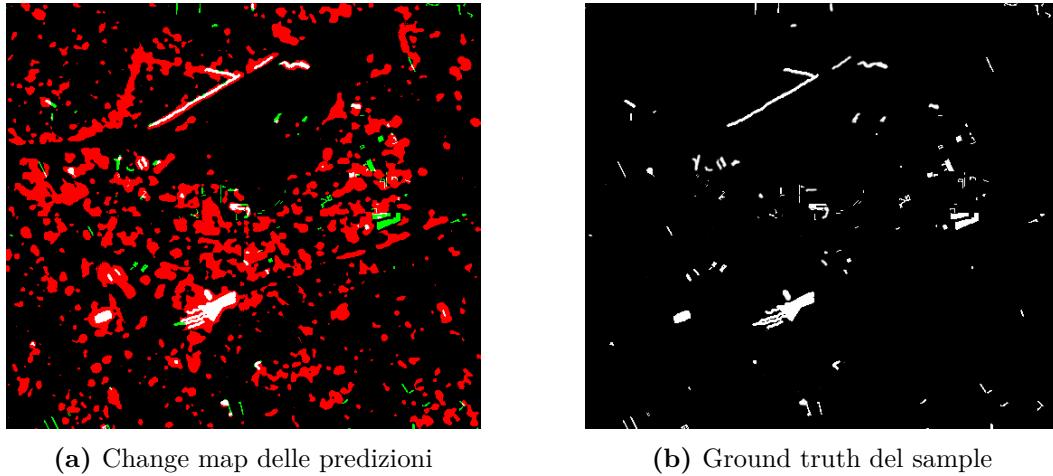
In fase di testing, tutte le immagini vengono divise in quattro patches grandi un quarto dell'immagine. Queste patches vengono date in input alla rete, che classifica ogni pixel, decidendo se questo rappresenta o meno un cambiamento d'interesse per la risoluzione del task.

Per poter valutare le predizioni del modello di apprendimento automatico è necessario suddividere i pixel predetti in quattro sottoinsiemi:

- True positive: questi pixel rappresentano un cambiamento d'interesse e vengono predetti correttamente.
- True negative: questi pixel non rappresentano un cambiamento e vengono predetti correttamente.
- False positive: questi pixel rappresentano un cambiamento, ma la rete predice il contrario.
- False negative: questi pixel non rappresentano un cambiamento, ma la rete predice il contrario.

Per poter visualizzare correttamente questa suddivisione, utilizzando i valori di verità presenti nella ground truth, sono state realizzate delle mappe di cambiamento. In queste mappe, i true positive vengono rappresentati di colore bianco, i true negative di colore nero, i false positive di colore verde, mentre i false negative di colore rosso.

Queste mappe risultano utili per l'analisi dei fallimenti e per il calcolo delle metriche, poiché permettono di visualizzare graficamente le predizioni della rete neurale. Un esempio di mappa di cambiamenti sul sample di Aguasclaras, presentato in figura 3.8, è presente in figura 4.1.



(a) Change map delle predizioni

(b) Ground truth del sample

Figura 4.1. Confronto tra la ground truth e la mappa di cambiamento del sample di Aguasclaras in figura 3.8

4.1.1 Loss e accuratezza

Le prime metriche che occorre introdurre sono la loss function e l'accuratezza. Poiché ogni modello di apprendimento automatico minimizza la propria funzione di costo, che modella gli errori nelle predizioni, i parametri del modello con i quali questa funzione assume il valore minimo saranno i migliori.

Le loss functions utilizzate sono la Balanced BCE Loss con la funzione di attivazione sigmoidea e la Focal Loss con funzione di attivazione Softmax, precedentemente introdotte.

L'accuratezza è una metrica intuitiva per valutare le predizioni di un modello di apprendimento automatico, descrivendo il numero di predizioni corrette sul totale. Questo criterio, tuttavia, può risultare fuorviante in virtù dello sbilanciamento del dataset. La formula per il calcolo dell'accuratezza è qui riportata, indicando con tp i pixel "true positive", con tn i pixel "true negative", con fp i pixel "false positive" e con fn i pixel "false negative":

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

4.1.2 Accuratezza per classe, precisione, recupero e F1 score

Una metrica più rilevante da utilizzare su un dataset contraddistinto da un grande sbilanciamento è l'accuratezza per classe, che valuta il numero di campioni classificati correttamente per ogni classe.

L'accuratezza per classe della classe d-esima in un dataset contenente D classi si calcola secondo la seguente formula:

$$\text{Accuracy per class}(d) = \frac{tp_d}{tp_d + fn_d}$$

Poiché il task è di classificazione binaria, le formule utilizzate per il calcolo dell'accuratezza per classe sono le seguenti:

$$\text{Accuracy change} = \frac{tp}{tp + fn}; \quad \text{Accuracy no change} = \frac{tn}{fp + tn}$$

Inoltre, vengono calcolate anche precisione (precision) e recupero (recall).

La precisione calcola la quantità di predizioni corrette per la classe dei pixel che hanno subito un cambiamento sul totale dei pixel predetti come appartenenti a quella classe.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Mentre il recupero corrisponde esattamente all'accuratezza della classe analizzata. Esso infatti calcola la quantità di predizione corrette per la classe di cambiamento sul totale dei pixel effettivamente appartenenti a quella classe.

$$\text{Recall} = \frac{tp}{tp + fn}$$

Nell'F1 score si combinano queste due metriche, calcolandone una media armonica. Indicando con p la precisione e con r il recupero, l'F1 score si calcola secondo la seguente formula:

$$F1 \text{ score} = \frac{2}{\frac{1}{p} + \frac{1}{r}} = 2 \frac{p \cdot r}{p + r}$$

4.2 Risultati Raccolti

Mediante la tecnica dell'Early Stopping, per ogni modello sono stati salvati i parametri delle epoche in cui erano raggiunti i valori minimi della loss function o massimi dell'F1 score.

Questi risultati, pur derivando dagli stessi modelli, sono molto eterogenei, perciò si è ritenuto opportuno raccoglierli e analizzarli separatamente.

Entrambe le reti neurali presentate sono state testate in quattro configurazioni, dalle quali sono stati ottenuti i risultati più rilevanti, operando ogni volta una scelta differente tra le seguenti:

- scelta del learning rate tra i valori {0.0001, 0.001};
- scelta della funzione di attivazione tra il sigmoide con Balanced BCE Loss o la softmax con Focal Loss.

Il numero di epoche scelto come limite massimo è 40. La lunghezza dell'addestramento potrebbe sembrare insufficiente, tuttavia già dopo un esiguo numero di iterazioni i modelli sono necessariamente soggetti all'overfitting, in virtù dell'eseguità del dataset. In effetti durante la sperimentazione sono stati testati anche addestramenti più lunghi fino a 50 epoche, registrando solo peggioramenti nelle prestazioni durante le iterazioni successive.

L'epoca d'inizio dell'overfitting di una delle reti addestrate è evidenziato dal grafico in figura 4.2, appartenente alla rete U-Net che effettua early fusion, utilizzata con attivazione Softmax e Focal Loss e con un learning rate pari a 0.0001.

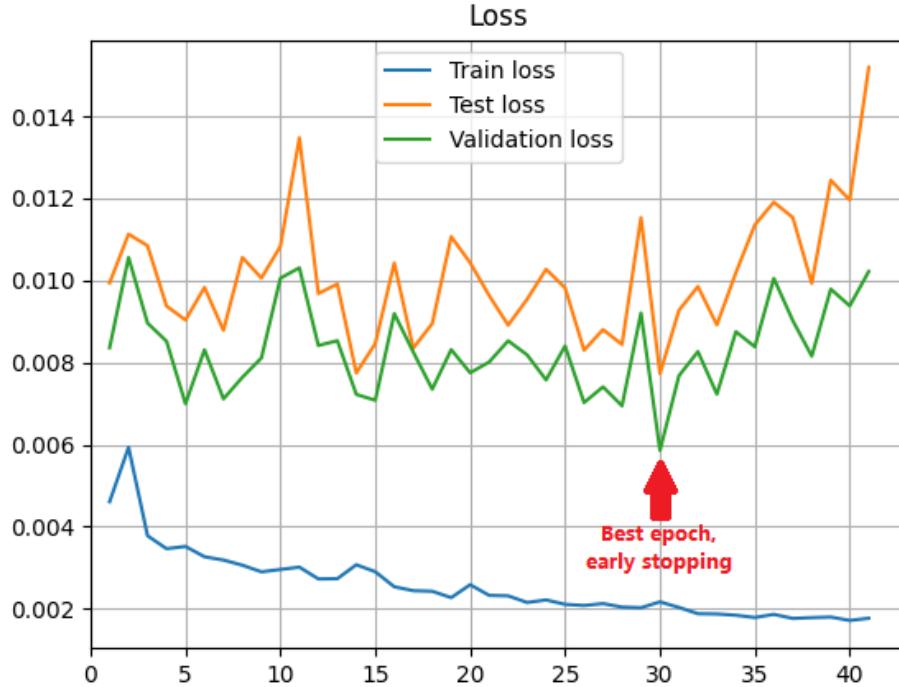


Figura 4.2. Grafico dell'evoluzione della loss function sul numero di epoche della U-Net che utilizza Focal Loss con learning rate pari a 0.0001

4.2.1 Minimizzazione della loss

I risultati riportati nelle tabelle 4.1 e 4.2 appartengono alle iterazioni in cui le reti utilizzati raggiungono il minimo valore della funzione di loss.

Tabella 4.1. Tabella contenente i valori delle metriche di accuratezza sul test set (1.2.1)

	Accuracy	Change accuracy	No change accuracy
U-Net with FL, lr=0.001	81.05%	85.76%	80.80%
U-Net with BBCE, lr=0.001	87.63%	79.95%	88.02%
U-Net with FL, lr=0.0001	83.61%	87.26%	83.42%
U-Net with BBCE, lr=0.0001	80.62%	89.57%	80.16%
Siamese with FL, lr=0.001	84.57%	86.06%	84.50%
Siamese with BBCE, lr=0.001	82.13%	90.02%	81.73%
Siamese with FL, lr=0.0001	86.51%	85.94%	86.54%
Siamese with BBCE, lr=0.0001	83.19%	86.96%	83.00%

I dati proposti sono calcolati sull'insieme di test. Questi dati sono parzialmente comparabili con quelli del paper di riferimento [5], poiché sono calcolati su un sottoinsieme del test set da loro utilizzato. Il sottoinsieme da me selezionato(1.2.1) per testare le performances del modello, però, contiene samples più difficili da pre-

Tabella 4.2. Tabella contenente i valori di precisione, recupero e F1 score sul calcolate sul test set (1.2.1)

	Precision	Recall	F1 score
U-Net with FL, lr=0.001	18.81%	85.76%	30.85%
U-Net with BBCE, lr=0.001	25.71%	79.95%	38.91%
U-Net with FL, lr=0.0001	21.44%	87.26%	34.42%
U-Net with BBCE, lr=0.0001	18.96%	89.57%	31.30%
Siamese with FL, lr=0.001	22.35%	86.06%	35.49%
Siamese with BBCE, lr=0.001	20.35%	90.02%	33.19%
Siamese with FL, lr=0.0001	24.88%	85.94%	38.59%
Siamese with BBCE, lr=0.0001	20.96%	86.96%	33.78%

dire rispetto all’insieme di validazione, sempre derivato dai campioni che gli autori del paper utilizzavano per testare le prestazioni. Perciò questa nuova suddivisione rende maggiormente complesso anche replicare le prestazioni da loro ottenute, che vengono in realtà superate dai miei modelli.

4.2.2 Massimizzazione dell’F1 score

I risultati ottenuti dai modelli massimizzando l’F1 score derivano da iterazioni in cui la rete neurale ha operato scelte maggiormente conservative, predicendo meno cambiamenti, ma aumentando notevolmente la precisione delle predizioni.

In effetti nelle iterazioni che minimizzano la loss il parametro più critico è la precisione, mentre il recupero ottiene valori elevati.

Si riportano quindi i risultati ottenuti nelle tabelle 4.3 e 4.4.

Tabella 4.3. Tabella contenente i valori delle metriche di accuratezza sul test set (1.2.1)

	Accuracy	Change accuracy	No change accuracy
U-Net with FL, lr=0.001	89.84%	69.90%	90.87%
U-Net with BBCE, lr=0.001	91.01%	71.64%	92.01%
U-Net with FL, lr=0.0001	89.64%	79.79%	90.15%
U-Net with BBCE, lr=0.0001	91.30%	69.15%	92.45%
Siamese with FL, lr=0.001	90.71%	69.43%	91.81%
Siamese with BBCE, lr=0.001	91.76%	73.99%	92.68%
Siamese with FL, lr=0.0001	91.12%	72.81%	92.07%
Siamese with BBCE, lr=0.0001	90.61%	72.10%	91.57%

Pur comportando una riduzione dell’accuratezza nelle predizioni dei cambiamenti, è possibile avere un modello più stabile e preciso, comunque utile per evidenziare la maggior parte dei cambiamenti, ma talvolta incapace di segmentarli completamente.

Il primo sottosistema di modelli risulta evidentemente più accurato nel trovare la maggior quantità di cambiamenti possibile, mentre il secondo gode di una maggiore precisione e accuratezza totale.

Tabella 4.4. Tabella contenente i valori di precisione, recupero e F1 score sul calcolate sul test set (1.2.1)

	Precision	Recall	F1 score
U-Net with FL, lr=0.001	28.43%	69.90%	40.41%
U-Net with BBCE, lr=0.001	31.73%	71.64%	43.98%
U-Net with FL, lr=0.0001	29.57%	79.79%	43.15%
U-Net with BBCE, lr=0.0001	32.20%	69.15%	43.94%
Siamese with FL, lr=0.001	30.53%	69.43%	42.41%
Siamese with BBCE, lr=0.001	34.40%	73.99%	46.96%
Siamese with FL, lr=0.0001	32.26%	72.81%	44.71%
Siamese with BBCE, lr=0.0001	30.73%	72.11%	43.09%

4.2.3 Differenze tra le predizioni dei modelli

I modelli ottenuti segmentano quindi le immagini in maniera differente. Quelli che minimizzano la loss function predicono aree di cambiamento maggiormente estese, rendendo però meno precisa la segmentazione e sbagliando la maggior parte delle predizioni (la precisione è inferiore al 50%) per i pixel che presentano una variazione nella coppia d'immagini, ma riuscendo spesso a cogliere tutti i cambiamenti contenuti in esse. I modelli che massimizzano l'F1 score, invece, forniscono una segmentazione più precisa, tuttavia a volte trovano solamente parte delle aree che identificano un cambiamento e tendono a ignorarli maggiormente rispetto ai modelli sopracitati. Anche queste reti sbagliano la maggior parte delle predizioni per pixel di cambiamento.

Un esempio delle predizioni dei modelli sul sample di Aguasclaras, effettuate dalla rete U-Net con early fusion, utilizzando il sigmoide come attivazione finale con la Balanced BCE Loss e con un learning rate pari a 0.001 è presentato in figura 4.3. La ground truth del sample è in figura 4.1b, il sample invece è in figura 3.8.

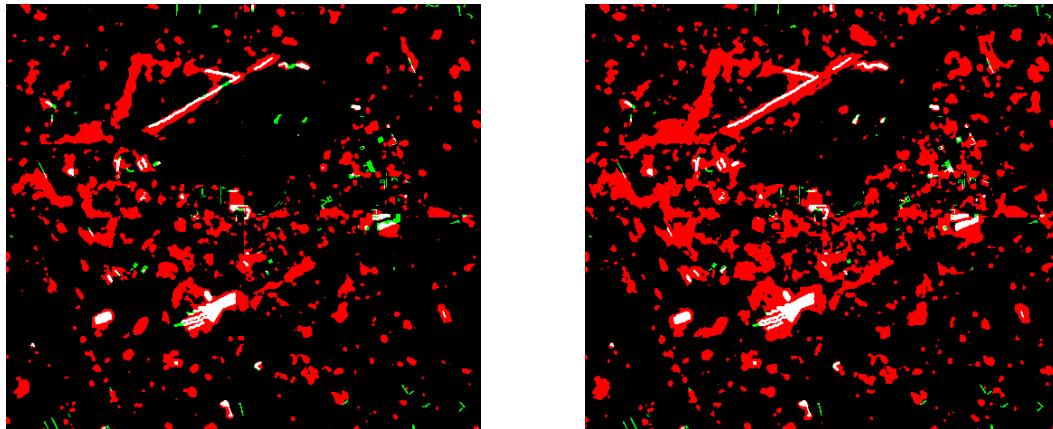


Figura 4.3. Confronto tra le mappe di cambiamento della rete che massimizza l'F1 score e la rete che minimizza la loss

4.3 Analisi dei risultati

I modelli forniscono buone predizioni per i cambiamenti di maggiore estensione e riescono a generalizzare ottimamente coppie di immagini che non presentano grandi scostamenti di colori, come è possibile vedere in figura 4.4.

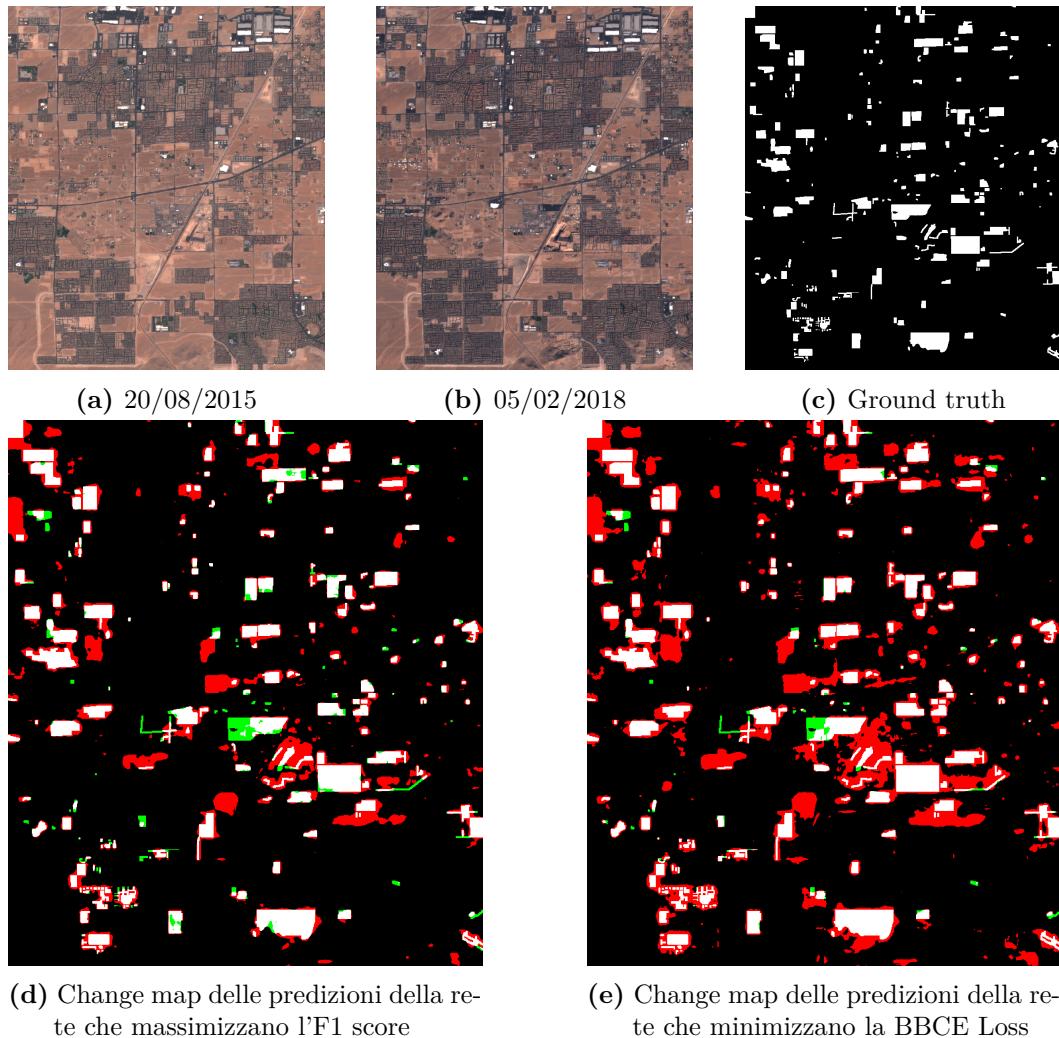


Figura 4.4. Mappe di cambiamento della rete siamese con sigmoide come attivazione finale e learning rate pari a 0.001 sul sample di Las Vegas

4.3.1 Analisi dei fallimenti

Questi modelli faticano particolarmente nel generalizzare i samples che presentano grandi variazioni di luminosità e contrasto nella coppia di immagini o risultano particolarmente bui. Per questa ragione ho suggerito di incrementarne la robustezza con ulteriore data augmentation sulle variazioni di colore (3.3.5).

I samples di Saclay West e Valencia (presentato in figura 3.1) sono un esempio lampante di questo tipo di errori. Il campione di Saclay West con annesse mappe

di cambiamento calcolate dalla rete siamese che utilizza Softmax e Focal Loss con learning rate pari a 0.001 sono presentati in figura 4.5.

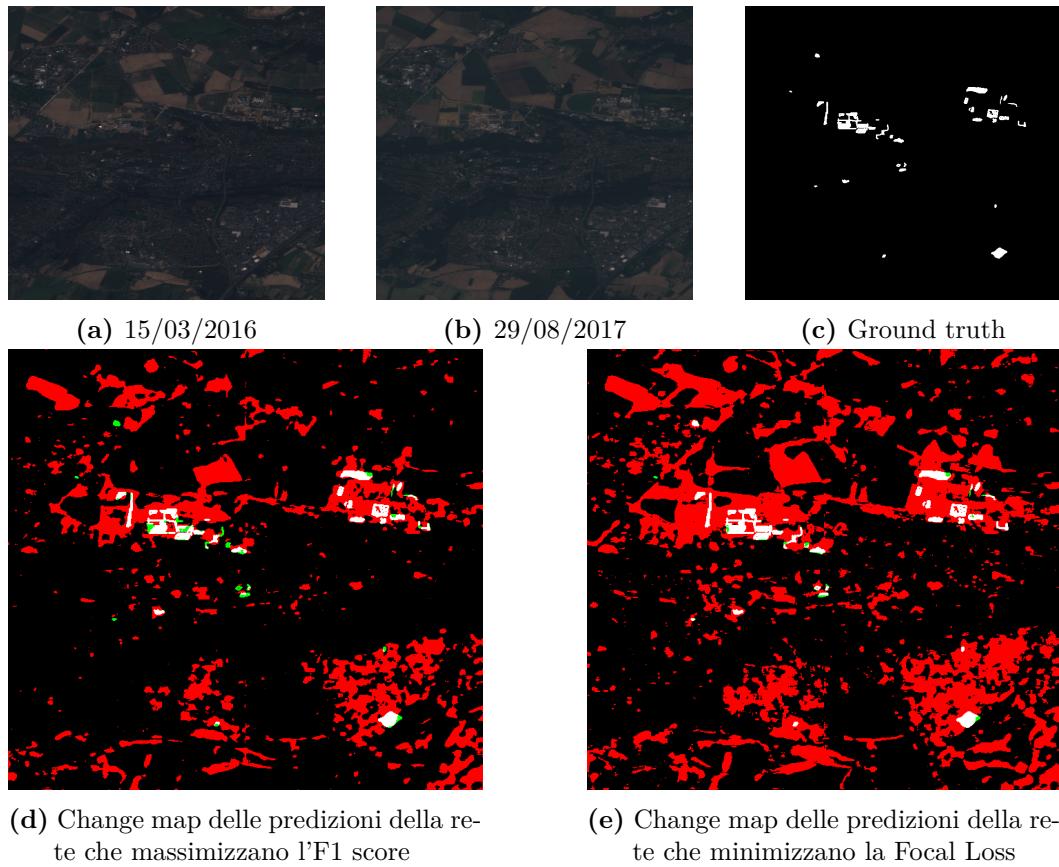


Figura 4.5. Mappe di cambiamento della rete siamese sul sample di Saclay West

Inoltre, la presenza di ostacoli fisici all’acquisizione dell’immagine, come ad esempio le nuvole, viene risolta al più parzialmente dalle reti neurali, com’è possibile vedere nel sample di Parigi, riportato in figura 4.6, insieme alle mappe di cambiamento prodotte dalla rete siamese, utilizzata con Focal Loss e learning rate pari a 0.0001.

Un’altra problematica è la raccolta in un’unica label di cambiamenti eterogenei. Questo rende più complesso il compito di generalizzazione delle reti neurali, che a volte ne predicono solo una parte, com’è possibile vedere dal sample di Rio, presentato in figura 4.7 insieme alle mappe di cambiamento calcolate dalla siamese, utilizzata con attivazione sigmoidea, Balanced BCE Loss e learning rate pari a 0.001.

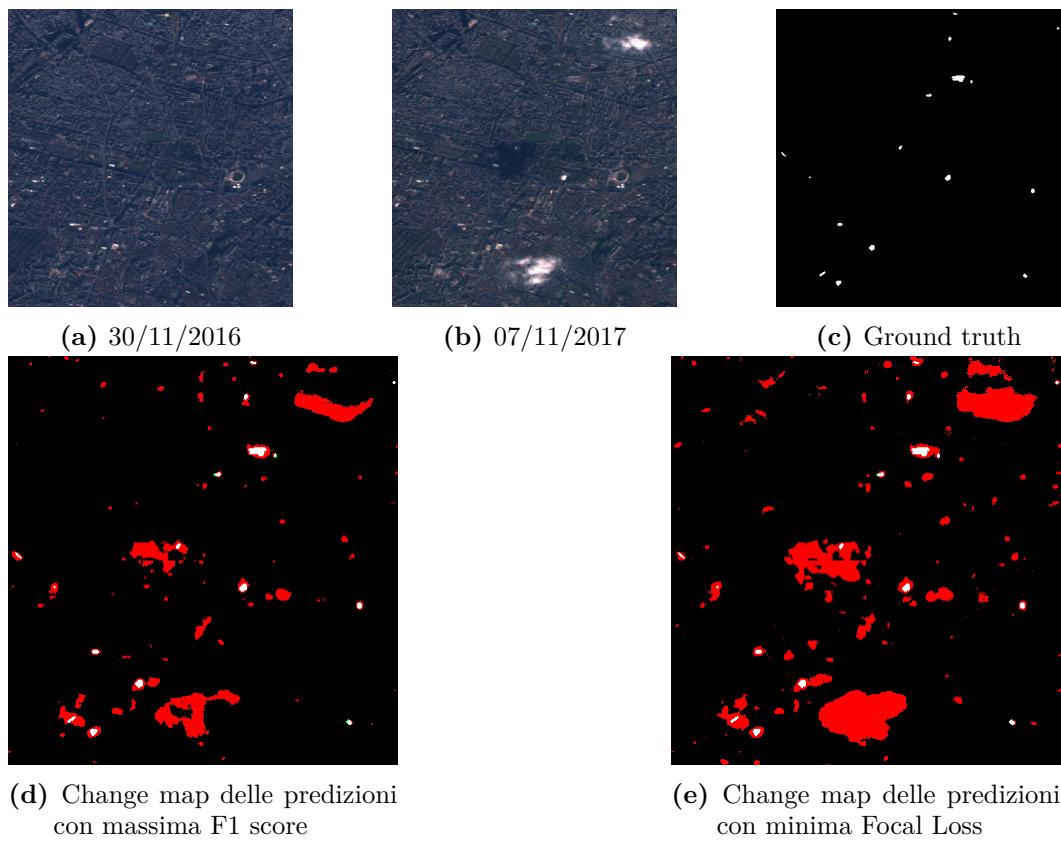


Figura 4.6. Mappe di cambiamento della siamese sul sample di Parigi

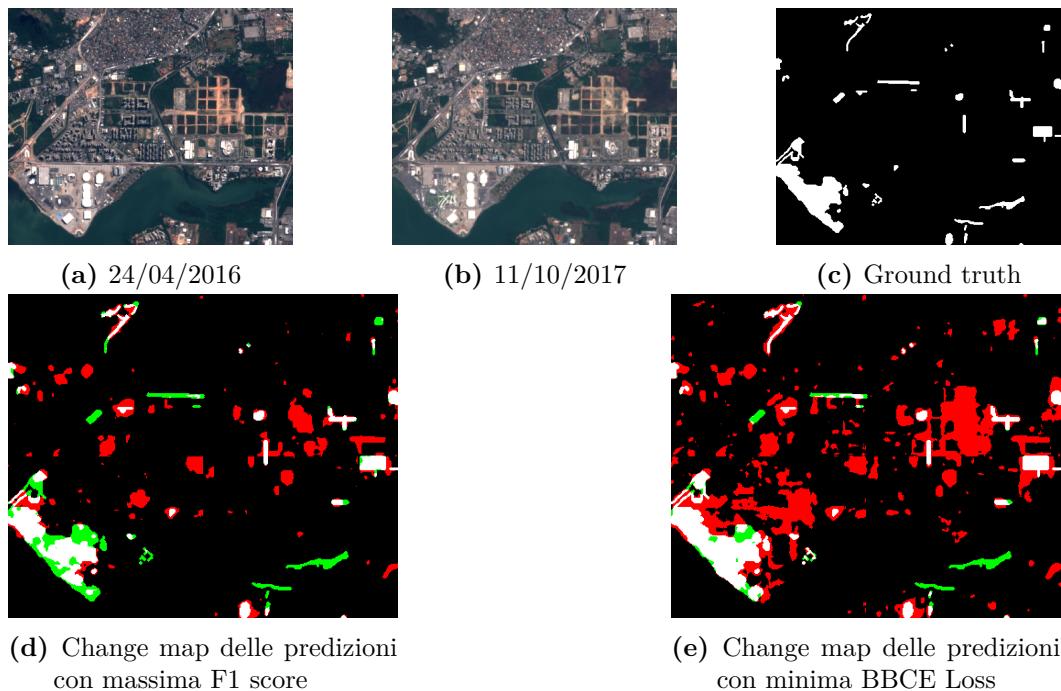


Figura 4.7. Mappe di cambiamento della rete siamese sul sample di Rio

4.3.2 L'effetto della Focal Loss

L'utilizzo della Focal Loss e la sua capacità di predire meglio i campioni più difficili è visualizzabile anche empiricamente.

Un sample in cui la rete che usa Focal Loss riesce a effettuare una segmentazione più accurata è quello di Dubai, già presentato in figura 1.1, le cui change map sono in figura 4.8.

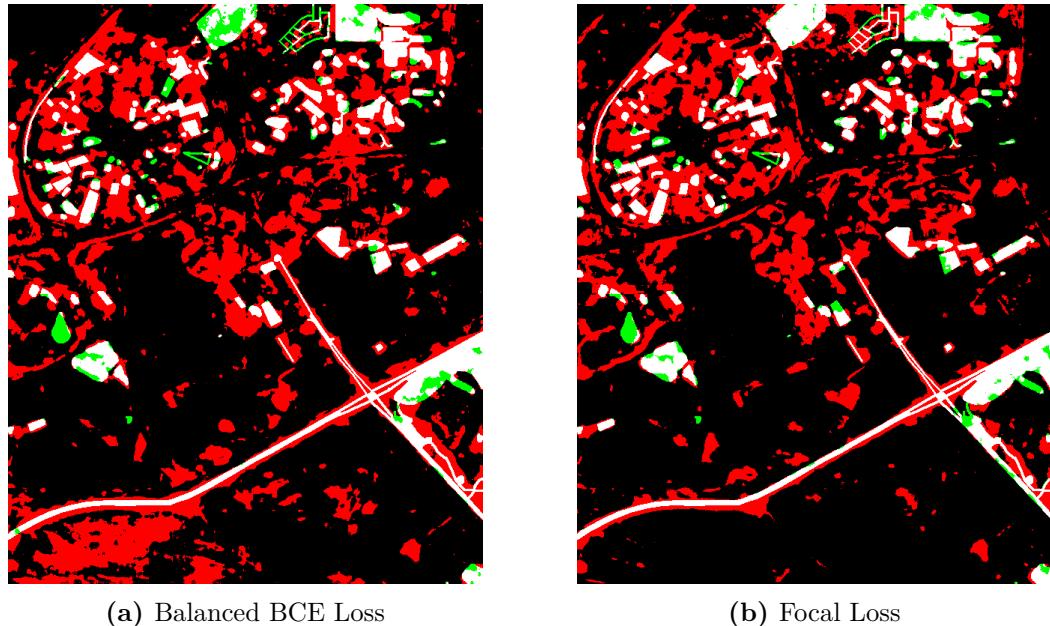


Figura 4.8. Mappe di cambiamento delle reti siamesi

I risultati messi a confronto sono calcolati entrambi utilizzando la rete siamese con learning rate pari a 0.0001, ma il primo è stato calcolato utilizzando il sigmoide con la Balanced BCE Loss, il secondo invece mediante la Softmax e la Focal Loss, nelle iterazioni con loss minima.

Risulta evidente che la segmentazione effettuata con la Focal Loss sia maggiormente precisa e la quantità di cambiamenti segmentati correttamente sia nettamente maggiore, nonostante complessivamente le prestazioni delle due reti neurali siano affini. Ciò significa che i modelli addestrati mediante Focal Loss risultano maggiormente efficienti sui campioni più difficili, come quello presentato o quello della città di Chongqing, tuttavia non risultano globalmente più accurati, generalizzando quindi peggio i samples considerati più semplici. Per questa ragione la scelta della funzione di loss dipende dal compito specifico che si desidera risolvere, risultando entrambe le scelte particolarmente affidabili.

4.4 Conclusione

L'eterogenetità dei modelli ottenuti permette di scegliere quelli maggiormente appropriati per le proprie esigenze.

Se l'obiettivo è quello di massimizzare la quantità di cambiamenti trovati, sicuramente l'utilizzo dei modelli che minimizzano la loss function risulta maggiormente appropriato.

Qualora invece si desideri offrire una segmentazione più precisa, anche se maggiormente incompleta, la scelta migliore è quella di utilizzare uno dei modelli che massimizzano l'F1 score.

In ogni caso, se i campioni presentano molti cambiamenti di diverse forme e dimensioni, la scelta più efficace è l'utilizzo di un modello addestrato mediante la Focal Loss.

Se invece i cambiamenti risultano facilmente riconoscibili e sono di forma e dimensione molto simile tra loro, un modello che utilizza la Balanced Binary Cross-Entropy Loss può risultare maggiormente appropriato.

Infine la rete siamese produce risultati con F1-score più elevata e maggiormente stabili rispetto alla corrispondente U-Net che effettua early-fusion, perciò può essere considerata l'architettura preferibile.

Bibliografia

- [1] Ajagekar Akash. *Adam*. 2021. URL: <https://optimization.cbe.cornell.edu/index.php?title=Adam>.
- [2] Ziad Alqadi. «Salt and Pepper Noise: Effects and Removal». In: *International Journal on Electrical Engineering and Informatics* 2 (lug. 2018).
- [3] Yingbin Bai et al. *Understanding and Improving Early Stopping for Learning with Noisy Labels*. 2021. arXiv: 2106.15853 [cs.LG].
- [4] Dor Bank, Noam Koenigstein e Raja Giryes. *Autoencoders*. 2021. arXiv: 2003.05991 [cs.LG].
- [5] Rodrigo Caye Daudt, Bertr Le Saux e Alexandre Boulch. «Fully Convolutional Siamese Networks for Change Detection». In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 4063–4067. DOI: 10.1109/ICIP.2018.8451652.
- [6] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].
- [7] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].
- [8] Keiron O’Shea e Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].
- [9] Olaf Ronneberger, Philipp Fischer e Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].
- [10] Ritwick Roy. *Neural Networks: Forward pass and Backpropagation*. <https://towardsdatascience.com/neural-networks-forward-pass-and-backpropagation-be3b75a1cfcc>. 2022.
- [11] Sutskever Ilya Srivastava Nitish Hinton Geoffrey Krizhevsky Alex e Salakhutdinov Ruslan. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15 (2014) 1929-1958 (2014). URL: <https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf>.
- [12] Nicholson Chris V. *The MNIST database*. URL: <https://wiki.pathmind.com/mnist>.

- [13] Sou Yoshihara, Taiki Fukuiage e Shin'ya Nishida. «Do training with blurred images make convolutional neural networks closer to humans concerning object recognition performance and internal representations?» In: *bioRxiv* (2022). DOI: 10.1101/2022.06.13.496005. eprint: [https://www.biorxiv.org/content/early/2022/06/16/2022.06.13.496005](https://www.biorxiv.org/content/early/2022/06/16/2022.06.13.496005.full.pdf).