ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Informatica - Scienza e Ingegneria

Corso di Laurea in Informatica

Super-Risoluzione di Immagini Satellitari Multispettrali tramite Modelli di Deep Learning

Relatore:
Dott.
Davide Evangelista

Presentata da:
Leonardo Berselli

Correlatrice:
Chiar.ma Prof.
Elena Loli Piccolomini

II Sessione
Anno Accademico 2024/2025

"Il più grande nemico della conoscenza non è l'ignoranza, ma l'illusione della conoscenza."

Stephen Hawking

Abstract

Il numero di frane è in aumento a causa dei cambiamenti climatici e una risposta rapida e mirata a questi eventi è fondamentale per mitigare i danni. Le immagini satellitari offrono una alternativa veloce ed economica alle rilevazioni aeree per l'individuazione degli smottamenti, tuttavia la risoluzione spaziale di queste è spesso insufficiente per una identificazione accurata. In questo lavoro si esplora l'uso di tecniche di super-risoluzione basate su reti neurali per affrontare questo problema. In particolare, questo studio è un lavoro preliminare per un progetto più ampio che mira a sviluppare una pipeline per la rimozione delle nuvole, la super-risoluzione e la segmentazione a partire dai dati forniti dai satelliti della missione Sentinel-2. Vengono confrontati diversi modelli di super-risoluzione, tra cui RCAN, Swin2MoSE e DRCT, per portare le immagini a una risoluzione spaziale di 2 metri per pixel. Inoltre, vengono condotti esperimenti per valutare l'impatto di vari fattori come la durata dell'allenamento, la dimensione del dataset e altri iperparametri. I risultati ottenuti dimostrano l'efficacia nell'utilizzo di queste reti per migliorare la qualità delle immagini satellitari e forniscono indicazioni utili per ulteriori miglioramenti, in preparazione per l'integrazione nella pipeline completa.

Indice

Capitolo 1

Introduzione

Nei prossimi decenni, a causa dei cambiamenti climatici dovuti all'attività antropica, si prevede un aumento degli eventi franosi a livello globale [1]. Per questo motivo, il monitoraggio delle frane è diventato un aspetto cruciale per la gestione del territorio e l'intervento tempestivo in caso di emergenze. Una parte del monitoraggio avviene tramite l'uso di immagini aeree, le quali data l'alta risoluzione spaziale permettono di evidenziare e segmentare le frane in una zona molto ampia. Questo compito viene però svolto principalmente a mano da esperti del settore, il che risulta essere un processo lungo e impraticabile su larga scala [2]. Negli ultimi anni, con lo sviluppo delle tecniche per l'acquisizione di immagini satellitari, la qualità e la disponibilità al pubblico sono aumentate notevolmente. Le immagini satellitari offrono una copertura globale, una frequenza di acquisizione elevata e rappresentano un'alternativa economica alle rilevazioni aeree. Con l'aumento della disponibilità dei dati satellitari, il numero di studi che utilizzano queste immagini per il monitoraggio delle frane è cresciuto rapidamente [3]. Inoltre, dall'avvento delle reti neurali convoluzionali, i metodi automatici per la segmentazione delle frane hanno raggiunto prestazioni eccellenti e il loro utilizzo è in rapida crescita [4]. Si è quindi deciso di studiare un metodo per il monitoraggio delle frane basato su immagini satellitari, con l'obiettivo di sviluppare una pipeline completamente automatizzata che possa essere utilizzata su larga scala. I

dati per lo studio provengono dalla missione *Sentinel-2*, sviluppata dall'Agenzia Spaziale Europea (ESA) nell'ambito del programma Copernicus. La costellazione è composta da due satelliti gemelli, che garantiscono un tempo di rivisitazione di circa 5 giorni alla stessa latitudine. Le immagini raccolte presentano una risoluzione spaziale compresa tra 10 e 60 metri, a seconda della banda spettrale [5, 6]. Per l'analisi delle frane, risultano particolarmente utili le bande nel visibile (B2: blu, B3: verde, B4: rosso) e nel near infrared (B8), da cui è possibile derivare indici di vegetazione, come il *Normalized Difference Vegetation Index* (NDVI) [7] ottenuto da:

$$\text{NDVI} = \frac{NIR - RED}{NIR + RED} \tag{1.1}$$

Questi indici forniscono informazioni sulla copertura vegetale, la cui variazione può essere un indicatore della presenza o dell'evoluzione di fenomeni franosi. L'utilizzo di queste immagini per il monitoraggio delle frane presenta però alcuni problemi quali l'eventuale presenza di nuvole che possono coprire parte dell'area di interesse e la bassa risoluzione spaziale che rende difficile l'individuazione precisa delle frane. Per affrontare questi problemi, la pipeline si basa su tre modelli di reti neurali distinte, dedicate rispettivamente alla rimozione delle nuvole, alla super-risoluzione e alla segmentazione delle frane. Il secondo step, ovvero la super-risoluzione, ha quindi l'obiettivo di portare la risoluzione spaziale dei dati satellitari a un livello paragonabile a quello delle immagini aeree normalmente utilizzate per il monitoraggio delle frane. Il presente lavoro si concentra su questo, utilizzando e confrontando diversi modelli di super-risoluzione, sia basati sulle CNN, come RCAN [8], che sui Transformer, come DRCT [9] e Swin2MoSE [10], al fine di valutare le loro prestazioni in questo specifico ambito.

Capitolo 2

Metodologia e Modelli

2.1 Super-Risoluzione

La super-risoluzione (SR) è una tecnica di elaborazione delle immagini che mira a ricostruire immagini ad alta risoluzione (HR) a partire da immagini a bassa risoluzione (LR). Data un'immagine LR Y, si assume generalmente che essa sia ottenuta da una corrispondente immagine HR X tramite un processo di degradazione, che può includere downsampling, sfocatura e rumore:

$$Y = D(X) + \sigma \tag{2.1}$$

dove D rappresenta il processo di degradazione e σ è il rumore. In scenari reali, i parametri di degradazione sono generalmente sconosciuti, e l'unica informazione disponibile è l'immagine LR osservata. Il problema della SR è intrinsecamente mal posto, poiché molte immagini HR diverse possono generare la stessa immagine LR dopo il processo di degradazione [11].

Il *Remote Sensing Image Super-Resolution* (RSISR) è un ambito specifico della SR applicato alle immagini satellitari. Nonostante i progressi tecnologici nell'acquisizione di immagini dallo spazio, la risoluzione spaziale dei dati satellitari rimane spesso insufficiente per molte applicazioni pratiche. Per questo motivo, il RSISR è stato applicato in contesti come nel monitoraggio ambientale, l'agricoltura di precisione o il monitoraggio urbano [12]. Dal

punto di vista metodologico, il RSISR può essere suddiviso in due approcci principali [13]:

1. **Single-Image Super-Resolution**: si concentra sulla ricostruzione di un'immagine HR a partire da una singola immagine LR.
2. **Multi-Image Super-Resolution**: utilizza più immagini LR della stessa scena per migliorare la qualità della ricostruzione.

Con lo sviluppo delle reti neurali, in particolare le Convolutional Neural Networks (CNN), questo campo ha visto un notevole progresso con l'introduzione di modelli sempre più sofisticati che hanno superato le prestazioni dei metodi tradizionali.

2.2 Interpolazione

Una prima tecnica per la super-risoluzione delle immagini è l'*Interpolazione*, che consiste nel calcolare i valori dei pixel mancanti in un'immagine a bassa risoluzione basandosi sui pixel circostanti. Esistono diversi metodi di interpolazione, tra cui:

- **Nearest Neighbor**: assegna a ogni pixel mancante il valore del pixel più vicino, è molto veloce ma produce immagini a blocchi.
- **Bilinear**: calcola il valore del pixel mancante come media pesata dei 4 pixel più vicini, producendo immagini più uniformi ma sfuocate.
- **Bicubic**: utilizza i 16 pixel più vicini per calcolare il valore del pixel mancante, offrendo una qualità superiore rispetto ai metodi precedenti, ma con un costo computazionale maggiore.

L'interpolazione è semplice e computazionalmente efficiente, tuttavia tende a produrre immagini sfocate e prive di dettagli fini, specialmente per fattori di scala elevati. Per questo motivo questa tecnica è generalmente utilizzata come base per confrontare le prestazioni di tecniche più complesse, come le reti neurali.

2.3 Convolutional Neural Networks

Le *Convolutional Neural Networks* (CNN) sono una classe di reti neurali alla base della maggior parte degli algoritmi di Computer Vision[1]. Le CNN sono in grado di catturare caratteristiche spaziali locali, come bordi, texture e forme, che sono fondamentali per compiti di visione come il riconoscimento di oggetti, la segmentazione delle immagini e la classificazione. Questa categoria di modelli si distingue dalle reti neurali completamente connesse per la loro capacità di sfruttare la struttura spaziale delle immagini. Attraverso l'utilizzo di convoluzioni, le CNN riducono drasticamente il numero di parametri da apprendere, rendendo l'addestramento più efficiente e riducendo il rischio di overfitting. Inoltre, la condivisione dei pesi dei kernel consente alla rete di essere traslazionalmente invariante, cioè di riconoscere una stessa caratteristica indipendentemente dalla sua posizione all'interno dell'immagine. Grazie a queste proprietà, le CNN sono diventate lo standard de facto per la rappresentazione e l'elaborazione di dati visivi.

2.3.1 Kernel

I *kernel* sono matrici di pesi che vengono appresi durante l'allenamento della rete. Ogni kernel k ha una dimensione $m \times n \times d$, dove m e n sono le dimensioni spaziali del kernel e d è la profondità, che corrisponde al numero di canali dell'immagine di input. Questo viene applicato all'immagine di input I di dimensione $H \times W \times D$ tramite la seguente operazione di convoluzione:

$$Y_{i,j} = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} \sum_{z=0}^{d-1} I_{i+x,j+y,z} \cdot k_{x,y,z} \tag{2.2}$$

dove Y rappresenta la feature map ottenuta dalla convoluzione e ogni valore $Y_{i,j}$ è calcolato come la somma pesata dei valori dell'immagine di input moltiplicati per i corrispondenti pesi del kernel centrato sulla posizione (i, j).

[1]La Computer Vision è il ramo dell'intelligenza artificiale che si concentra sull'estrarre informazioni significative da immagini e video.

L'operazione di convoluzione viene eseguita scorrendo il kernel sull'immagine di input con un certo stride, che determina di quanti pixel il kernel si sposta ad ogni passo.

Per evitare che il kernel esca dai bordi dell'immagine, si può applicare un padding, ovvero aggiungere dei pixel intorno all'immagine di input, per mantenere le dimensioni dell'output desiderate. Essendo quindi che i kernel applicano gli stessi pesi su tutta l'immagine, ognuno di essi è in grado di rilevare una specifica caratteristica indipendentemente dalla sua posizione nell'immagine. Ogni kernel rappresenta un neurone del layer convoluzionale e il numero di kernel determina la profondità della feature map in output. In Figura 2.1 è mostrato un esempio di convoluzione con un kernel 3×3 su un'immagine 5×5 per il calcolo di un singolo valore della feature map in output, per comprendere visivamente l'operazione.

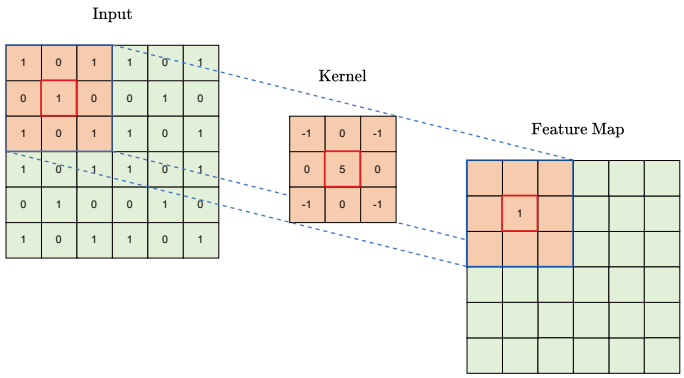

Figura 2.1: Operazione di Convoluzione.

I layer basati su convoluzioni sono detti Convolutional Layers e applicandoli in sequenza la rete è in grado di catturare caratteristiche sempre più complesse e astratte. Inoltre, con l'aumento della profondità della rete, il cosiddetto *receptive field* di ogni neurone aumenta, permettendo di utilizzare le informazioni da un contesto più ampio dell'immagine [14]. Il kernel infatti riceve in input i valori delle feature map adiacenti prodotte dal layer precedente e quindi il valore di ogni neurone dipende da un'area sempre più grande dell'immagine di input.

2.3.2 Funzione di Attivazione

La *funzione di attivazione* è una funzione matematica che permette di introdurre non-linearità in un modello, permettendo a questo di imparare pattern complessi nei dati. Con non-linearità si intende che la relazione tra input e output non cambia proporzionalmente con l'input, come altrimenti farebbe con delle semplici somme pesate e moltiplicazioni. Senza queste il modello indipendentemente dalla profondità risolverebbe una semplice combinazione lineare dei layer. Una delle più classiche funzioni è la Rectified Linear Unit che è definita come:

$$\text{ReLU}(x) = \max(0, x) \tag{2.3}$$

che intuitivamente propaga l'input solamente quando questo supera una certa soglia, funzionando come una sorta di filtro. Nel caso invece di problemi di classificazione viene per esempio spesso utilizzata come funzione di attivazione la softmax che trasforma l'input in probabilità. L'output è infatti un vettore che sommato è 1 e dove ogni valore rappresenta quanto è probabile che l'input appartenga ad una determinata classe.

Non tutte le funzioni di attivazione sono uguali e devono quindi essere scelte in relazione al problema di interesse. La scelta della funzione di attivazione deve essere quindi decisa in relazione al tipo di problema, andando ad influire sull'abilità del modello di apprendere.

2.3.3 Pooling

Le CNN spesso includono layer di pooling, ovvero operazioni che riducono la dimensione spaziale delle feature map. Questo aiuta a ridurre il numero di parametri e il costo computazionale, mantenendo le informazioni più rilevanti e riducendo il rischio di overfitting. I metodi di pooling più comuni sono il max pooling e l'average pooling e consistono nel dividere la feature map in regioni non sovrapposte e calcolare rispettivamente il valore massimo o la media all'interno di ogni zona.

2.3.4 CNN per la Super-Risoluzione

Un ramo specifico della Computer Vision è la super-risoluzione, che, come già accennato, ha conosciuto un notevole progresso a partire dall'introduzione delle CNN. Il primo modello basato su questo tipo di architettura è stato SRCNN [15], seguito da architetture più profonde come VDSR [16] ed EDSR [17], fino a modelli più recenti come RCAN [8]. Questi modelli hanno progressivamente superato le limitazioni dei metodi precedenti grazie a innovazioni architetturali, quali l'uso di blocchi residui, meccanismi di attenzione e reti più profonde, che ne hanno migliorato la capacità di ricostruzione dell'immagine. Negli ultimi anni, lo sviluppo di architetture basate su Transformer in altri ambiti ha portato all'adozione di queste tecniche anche nel campo della Computer Vision e della super-risoluzione, consentendo ulteriori miglioramenti delle prestazioni.

2.4 Transformers

I *Transformers* sono una classe di modelli basati su meccanismi di *self-attention*, introdotti originariamente per il Natural Language Processing (NLP) al fine di gestire sequenze di testo lunghe senza utilizzare strutture ricorrenti. Il principio fondamentale dei Transformers è la capacità di pesare dinamicamente l'importanza relativa di ciascun elemento dell'input rispetto agli altri, consentendo al modello di catturare relazioni globali all'interno dei dati [18].

2.4.1 Struttura dei Transformers

La struttura classica dei Transformers, visibile nella Figura 2.2 è composta da due blocchi principali:

- **Encoder:** L'encoder è composto da uno stack di N layers identici, ognuno composto da due sottolayers. Il primo è un meccani-

smo di *Multi-Head Self-Attention* e il secondo è una rete feed-forward completamente connessa (FNN).

- **Decoder:** Il decoder ha una struttura simile all'encoder utilizzando anch'esso N layers identici, ma con un terzo sottolayer aggiuntivo che effettua l'attenzione sui valori dell'encoder. Inoltre, il decoder utilizza una maschera per impedire l'accesso alle informazioni delle posizioni future, e l'output obiettivo passato in input a questo è spostato di una posizione a destra rispetto all'input, per garantire che la previsione per la posizione i possa dipendere solo dai target noti alle posizioni precedenti $< i$.

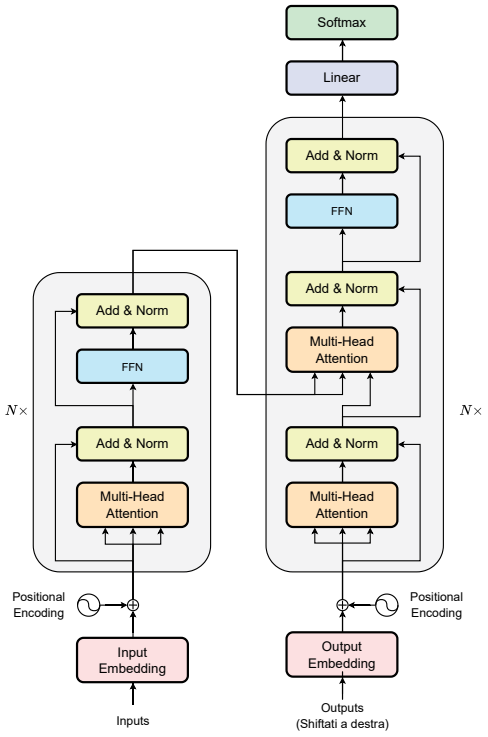

Figura 2.2: Architettura del Transformer.

Oltre alla struttura generale, i Transformers si basano su diversi meccanismi che permettono a questo tipo di modelli di funzionare efficacemente. Questi concetti sono fondamentali per comprendere il funzionamento dei Transformers e includono:

Scaled Dot-Product Attention: Dati tre vettori Q (query), K (key) e V (value), dove i primi due sono di dimensione d_k e l'ultimo di dimensione d_v, l'attenzione è data da:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

Ovvero si calcola il prodotto scalare tra Q e K, si scala il risultato dividendo per $\sqrt{d_k}$ per evitare valori troppo grandi che renderebbero la softmax instabile, si applica la funzione softmax per ottenere i pesi di attenzione, e infine si moltiplica per V per ottenere l'output finale. Questo rappresenta un vettore pesato dei valori V, dove i pesi sono determinati dalla similarità tra Q e K.

Multi-Head Attention: L'attenzione su una singola testa può limitare la capacità del modello di focalizzarsi su diverse parti dell'input. Per questo motivo date le d_{model} dimensioni dell'input, si proiettano Q, K e V in h spazi diversi di dimensione d_k, d_k e d_v rispettivamente, dove $d_k = d_v = d_{model}/h$. La funzione di attenzione è quindi calcolata in parallelo per ogni testa secondo la formula:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{dove head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.5}$$

Le matrici sono di parametri apprendibili dove $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ e $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. Tramite W^O si proietta l'output concatenato delle h teste di attenzione nuovamente nello spazio originale di dimensione d_{model}. Una visualizzazione dell'architettura della Multi-Head Attention è mostrata in Figura 2.3.

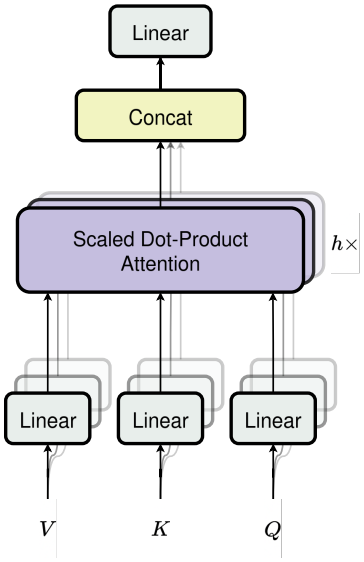

Figura 2.3: Architettura della Multi-Head Attention.

Feed-Forward Networks (FNN): Per quanto riguarda le reti feed-forward, ogni sottolayer FNN all'interno dell'encoder e del decoder è costituito da due layer lineari con una funzione di attivazione ReLU tra di essi. Dato un input x, l'output della FNN è calcolato come:

$$\text{FNN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{2.6}$$

Il layer interno ha una dimensione variabile d_{ff}, che è generalmente più grande di d_{model} per aumentare la capacità del modello.

Embedding: Nei Transformer i token di input e di output vengono convertiti in vettori di dimensione fissa d_{model} tramite un embedding lineare apprendibile. All'uscita dal decoder, le rappresentazioni vengono trasformate mediante una proiezione lineare seguita da una softmax, ottenendo così una distribuzione di probabilità sui possibili token di output. Per ridurre il numero di parametri e favorire una migliore generalizzazione, la matrice dei pesi dell'embedding e quella della proiezione finale vengono condivise.

Positional Encoding: Il modello Transformer, non contenendo né ricorrenze né convoluzioni, non ha di per sé informazioni sulla posizione dei token nella sequenza. Per questo motivo viene aggiunto un *positional encoding* (PE) agli embedding di input, che fornisce informazioni sulla posizione assoluta e facilita l'apprendimento delle relazioni relative. Il PE utilizzato nell'architettura originale è basato su funzioni sinusoidali a diverse frequenze, definite come:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \end{aligned} \tag{2.7}$$

dove *pos* è la posizione del token nella sequenza e i è la dimensione dell'embedding. Viene quindi generato un vettore di dimensione d_{model} per ogni posizione, che viene sommato all'embedding del token corrispondente, dove all'aumento della dimensione i corrisponde una frequenza più bassa. In questo modo ogni posizione della sequenza ha un encoding unico, e la natura periodica delle funzioni sinusoidali permette al modello di apprendere facilmente le relazioni tra posizioni diverse.

2.4.2 Vision Transformers

Vista l'evoluzione delle architetture nel campo del Natural Language Processing con l'introduzione dei Transformers si è cercato di adattare questa architettura anche al campo della Computer Vision. In particolare i *Vision Transformer* (ViT) [19] sono stati introdotti per la classificazione delle immagini, ottenendo risultati competitivi rispetto alle reti convoluzionali limitando quanto più possibile le modifiche alla struttura originale dei Transformers.

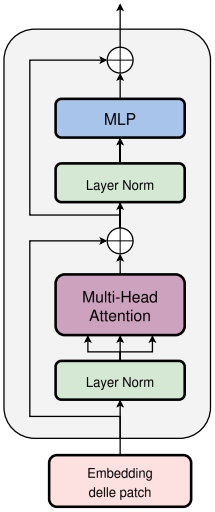

Figura 2.4: Architettura del Vision Transformer.

La struttura dei ViT è mostrata in Figura 2.4 e si basa su un encoder Transformer standard, con alcune modifiche per adattarlo a gestire immagini. Essendo infatti i Transformers progettati per ricevere in input sequenze di token, l'immagine $x \in \mathbb{R}^{H \times W \times C}$ viene appiattita in una sequenza di patch, ovvero sotto-immagini di $n \times n$ pixel. Si ottiene dunque un vettore $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, dove P è la dimensione di ogni patch, $N = HW/P^2$ è il numero totale di patch e C è il numero di canali dell'immagine. Ogni patch viene quindi proiettata in uno spazio di dimensione D tramite una proiezione lineare, ottenendo così una sequenza di embedding delle patch $z_0 \in \mathbb{R}^{N \times D}$. A differenza di questi però viene eseguita una layer normalization prima di ogni sottolayer di attenzione e FNN, e non dopo. Per quanto riguarda il positional encoding, nei ViT viene utilizzato un encoding apprendibile invece di quello sinusoidale.

Nelle CNN l'operazione di convoluzione sfrutta la località spaziale ovvero l'assunzione che i pixel vicini siano più correlati tra loro rispetto a quelli lontani, inoltre la convoluzione è *translation equivariant* ovvero è invariante rispetto alla posizione dell'oggetto da riconoscere essendo i filtri condivisi su tutta l'immagine.

Questo introduce un *bias induttivo*[2] che aiuta il modello a generalizzare meglio con meno dati di addestramento. Nei ViT invece l'attenzione è calcolata globalmente tra tutte le patch, e non vi è condivisione di pesi come nelle CNN, il che obbliga il modello ad imparare queste proprietà dai dati, richiedendo quindi dataset di addestramento molto più grandi per raggiungere prestazioni comparabili.

2.4.3 Swin Transformers

I Vision Transformers hanno mostrato ottime prestazioni per compiti di classificazione delle immagini ma l'architettura non è adatta come backbone per compiti di visione più generali. Inoltre, a causa del costo computazionale elevato associato al calcolo dell'attenzione in modo globale l'applicazione a immagini ad alta risoluzione è limitata. Per calcolare la self-attention in modo efficiente, con *Swin Transformer* l'immagine viene suddivisa in finestre non sovrapposte di dimensione $M \times M$, e l'attenzione viene calcolata solo all'interno di ciascuna finestra. Il limite di questo approccio è però che si perde la correlazione tra le varie finestre dell'immagine.

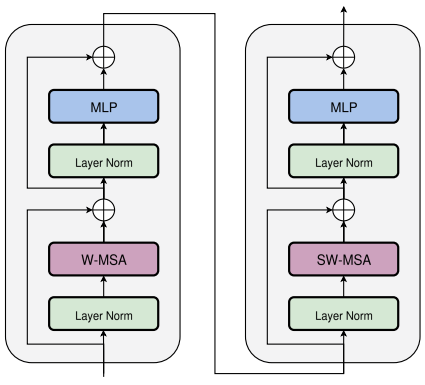

Figura 2.5: Architettura dello Swin Transformer.

[2]Un *bias induttivo* è un insieme di assunzioni integrate in un modello che permettono di ridurre la complessità del problema limitando lo spazio delle soluzioni possibili.

Per questo motivo viene introdotto il meccanismo delle *shifted windows*, in cui si alternano due tipi di layer: uno con finestre regolari (W-MSA) e uno con finestre *shifted* (SW-MSA), come si può vedere in Figura 2.5. Nel layer con finestre shifted, la partizione viene traslata di $(M/2, M/2)$ pixel, in modo che ogni finestra includa parte delle finestre adiacenti del layer precedente.

In pratica, dopo aver suddiviso l'immagine in finestre regolari, il layer successivo effettua una nuova partizione spostata, permettendo all'attenzione di calcolare correlazioni tra finestre contigue. Per gestire i bordi dell'immagine, si applica un *cyclic shift*: i pixel che escono da un lato rientrano dall'altro. Questo consente di mantenere la dimensione fissa delle finestre e di calcolare l'attenzione in maniera efficiente. Poiché alcune finestre contengono pixel che non erano adiacenti nel layer precedente, è necessario applicare una maschera durante il calcolo dell'attenzione per evitare di correlare feature non adiacenti. Dopo il calcolo dell'attenzione, si effettua un *reverse shift* per riportare le finestre alla posizione originale, in modo da poter sommare il residuo e mantenere la coerenza spaziale. Questo processo di shifting è illustrato in Figura 2.6, dove si può vedere come le finestre vengano traslate e poi riportate alla posizione originale.

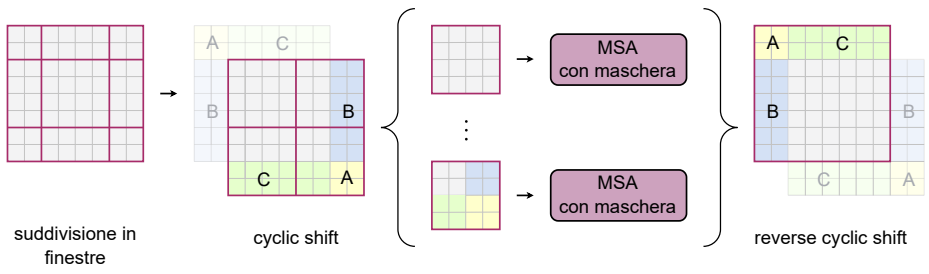

Figura 2.6: Processo di shifting delle finestre nello Swin Transformer.

Relative Position Bias (RPB): Un ulteriore miglioramento introdotto nello Swin Transformer è il *Relative Position Bias* (RPB), che aggiunge un bias relativo alla posizione delle patch all'interno di ogni finestra. A differenza dell'attenzione nei Transformer standard, che segue la formula (2.4), nello

Swin l'attenzione all'interno di ogni testa viene calcolata con la seguente espressione:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B\right)V \tag{2.8}$$

dove $Q, K, V \in \mathbb{R}^{M^2 \times d}$, d è la dimensione di query e key e M^2 è il numero di patch in ogni finestra. Dovendo calcolare una distanza relativa tra ogni coppia di patch, su ciascun asse x e y i possibili offset vanno da $-(M-1)$ a $M-1$, per un totale di $2M-1$ valori distinti. Si definisce quindi una matrice parametrica ridotta $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$, i cui valori vengono usati per costruire la matrice completa dei bias $B \in \mathbb{R}^{M^2 \times M^2}$ impiegata nella self-attention.

2.5 Vision Mixture of Experts

Nel campo del deep learning i modelli di grandi dimensioni hanno dimostrato prestazioni superiori in vari compiti, ma il loro addestramento e utilizzo richiede risorse computazionali significative. Tipicamente però queste reti sono "dense", ovvero tutti i parametri sono attivi per ogni input. Si è quindi andato a sviluppare un approccio differente chiamato Conditional Computation in cui solo una parte dei parametri sono attivati mantenendo quindi il costo computazionale inalterato ma aumentando la capacità del modello.

Dopo che i Mixture of Experts (MoE) e più precisamente i *Sparsely-Gated Mixture of Experts* sono stati introdotti per il NLP [20], si è applicato, come successo per i Transformers, questo concetto anche alla Computer Vision. Sono state dunque introdotti i *Vision Mixture of Experts* (V-MoE) [21], applicati per la prima volta al modello ViT, la cui architettura è mostrata in Figura 2.7.

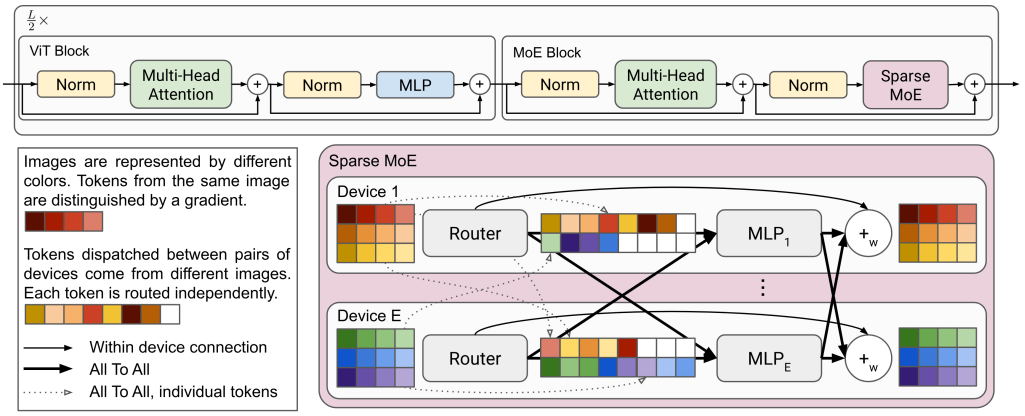

Figura 2.7: Overview dell'architettura V-MoE da Riquelme et al.

Conditional Computation: Come già accennato, l'obiettivo della Conditional Computation è quello di attivare diverse sezioni del modello a seconda dell'input. Con i MoE questo viene ottenuto avendo diversi "esperti" che si specializzano per certi tipi di input. Un layer MoE è composto da E esperti e applica la seguente operazione:

$$\text{MoE}(x) = \sum_{i=1}^{E} g_i(x) e_i(x) \tag{2.9}$$

dove e_i è la funzione calcolata dall'esperto i e g è la funzione di "routing" che assegna un peso $g(x)_i$ a ciascun esperto in base all'input x. In particolare e_i e g sono parametri apprendibili e se g è sparsa allora solo pochi esperti saranno attivati per ogni input poiché sono assegnabili solo $k \ll E$ pesi non nulli limitando di conseguenza il costo computazionale a solamente questi.

Per applicare questo concetto a ViT, una parte dei layer MLP viene sostituita con un layer MoE, ottenendo così il modello V-MoE, in cui ogni esperto è un MLP indipendente che avrà dunque pesi differenti. Possiamo quindi separare la definizione di parametri di un modello in due categorie:

- Sparse Parameters: è il numero totale di parametri.
- Active Parameters: è il numero effettivo di parametri attivi per un dato input.

Routing: Per ogni layer MoE viene utilizzata la seguente funzione di routing:

$$g(x) = \text{Top}_k(\text{softmax}(Wx + \epsilon)) \tag{2.10}$$

dove Top_k imposta a zero tutti i valori tranne i k più alti e $\epsilon \sim \mathcal{N}(0, 1)$ è un rumore gaussiano aggiunto per favorire l'esplorazione di tutti gli esperti durante l'addestramento. x in questo caso è un singolo token della sequenza di input e che attraverso la funzione di routing viene assegnato agli esperti, i quali generalmente sono 1 o 2.

Expert's Buffer Capacity: Durante l'addestramento i modelli sparsi possono preferire solo un piccolo sottoinsieme di esperti, limitando i benefici della Conditional Computation. Viene quindi definito un parametro chiamato *buffer capacity* B_e che limita il numero di token che ogni esperto può ricevere tramite la seguente formula:

$$B_e = \text{round}\left(\frac{k\,N\,P\,C}{E}\right) \tag{2.11}$$

dove k è il numero di esperti attivati per ogni token, N è il numero di immagini nel batch, P è il numero di token per immagine, E è il numero di esperti e C è ratio a scelta per aumentare o diminuire la capacità. Se però un esperto raggiunge la sua capacità massima, i token in eccesso non vengono elaborati da questo, per quanto l'informazione non viene persa poiché vi è la connessione residua e se $k > 1$ il token viene comunque passato ad un altro esperto.

Batch Prioritized Routing: Con la gestione classica del routing i token vengono assegnati agli esperti in base alla loro posizione nella sequenza senza considerare una priorità. Nel caso in cui però un esperto raggiunga la sua capacità massima, i token in eccesso non vengono elaborati da questo, possibilmente perdendo così informazioni importanti. Per questo motivo viene introdotto il *Batch Prioritized Routing* (BPR) dove si calcola uno punteggio di

priorità per ogni token e si ordinano in base a questo prima dell'assegnazione agli esperti. Il punteggio di priorità viene calcolato come:

$$s(X)_t = \max_i \left(g(X)_{t,i}\right) \tag{2.12}$$

dove X è la sequenza di input e t è l'indice del token e i l'indice dell'esperto e dunque si prende il massimo peso assegnato a quel token tra tutti gli esperti.

2.6 Architetture dei modelli

In questo lavoro sono stati selezionati tre modelli di super-risoluzione. Il primo è *Residual Channel Attention Network* (RCAN) [8], un modello basato su CNN che introduce lunghe skip connection e un meccanismo di attenzione sui canali, consentendo l'addestramento di reti molto profonde. Avendo scelto RCAN come baseline, e considerando la complessità del problema della super-risoluzione, l'attenzione è stata poi rivolta a modelli più recenti e allo stato dell'arte (SOTA). Numerose architetture basate su Transformer sono state proposte per questo compito, molte delle quali utilizzano *SwinIR* come backbone. Tra queste, *Dense Residual Connected Transformer* (DRCT) affronta il problema dell'information bottleneck presente in SwinIR introducendo connessioni dense tra i vari blocchi. Infine, per affrontare in modo più specifico la RSISR, è stato adottato *Swin2MoSE* [10], che integra un meccanismo di Mixture of Experts per incrementare la capacità del modello senza aumentare in maniera eccessiva il numero di parametri e specificatamente testato su dataset di immagini satellitari.

2.6.1 Residual Channel Attention Network

I modelli di super-risoluzione basati su CNN precedenti, come EDSR [17], hanno mostrato che l'aumento della profondità della rete porta a un miglioramento delle prestazioni. Tuttavia, questi approcci trattano tutti i canali delle feature in modo uniforme, senza considerare che alcuni di essi possa-

no risultare più rilevanti di altri nella ricostruzione dell'immagine.[3] Inoltre, l'aumento della profondità comporta il problema del vanishing gradient, che rende complesso l'addestramento di reti molto profonde.

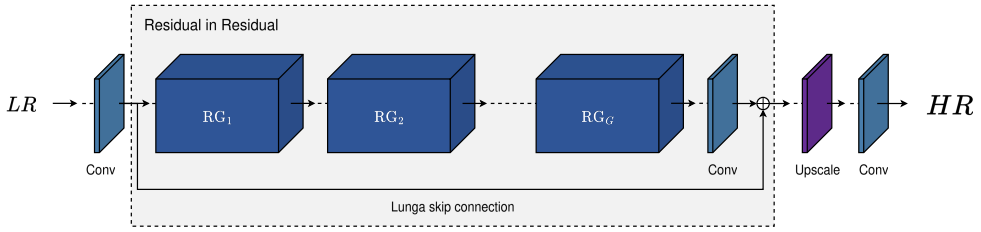

Figura 2.8: Architettura di RCAN.

Per affrontare queste limitazioni RCAN, la cui struttura generale è possibile osservare in Figura 2.8, introduce un blocco principale denominato *Residual in Residual* (RIR), costituito da G *Residual Groups* (RG) e da una connessione residua globale che collega l'input all'output del blocco. La coda del modello è invece formata da un layer di upsample[4] seguito da una convoluzione per generare l'immagine HR finale, in quanto l'applicazione di questo alla fine piuttosto che all'inizio della rete si è dimostrato più efficiente, riducendo il costo computazionale e migliorando le prestazioni. Ogni RG include B *Residual Channel Attention Blocks* (RCAB) e a sua volta è dotato di una connessione residua che collega l'input all'output del gruppo. L'elemento chiave introdotto da RCAN è il meccanismo di *Channel Attention* (CA), che consente di pesare i canali in base alla loro importanza per la ricostruzione.

Dato un input $X = [x_1, x_2, \ldots, x_C]$ con C canali, dove $\forall i \in [1, C]$, $x_i \in \mathbb{R}^{H \times W}$, si calcola dapprima il global pooling, cioè la media spaziale di ciascun canale:

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i, j), \quad c = 1, 2, \ldots, C \tag{2.13}$$

[3] Un'immagine LR contiene principalmente *low-frequency information*, mentre le *high-frequency information*, cruciali per i dettagli, sono più difficili da recuperare e pertanto devono essere prioritarie.

[4] L'upsampling indica il processo di aumento della risoluzione spaziale di un'immagine, ovvero l'incremento del numero di pixel che la compongono.

ottenendo così un vettore compatto che rappresenta l'attivazione media di ciascun canale. Questo vettore viene trasformato in un insieme di pesi attraverso una rete feed-forward composta da due layer lineari: il primo riduce la dimensione di un fattore r, mentre il secondo la riporta a C. Tra i due layer viene applicata una funzione ReLU, mentre in uscita una sigmoid normalizza i pesi tra 0 e 1.[5] Infine, tali pesi vengono applicati ai canali dell'input X, enfatizzando quelli più significativi e attenuando quelli meno rilevanti.

Il blocco base della rete è RCAB, che integra convoluzioni, meccanismo di attenzione sui canali e connessioni residue. In particolare, ogni RCAB è formato da due convoluzioni separate da una ReLU, seguite dal modulo di CA e da una connessione residua che collega l'input all'output del blocco.

2.6.2 Dense Residual Connected Transformer

Con l'introduzione di SwinIR [22], un modello per la super-risoluzione basato su Swin Transformer, questo si è rapidamente affermato come uno dei modelli principali in questo ambito. Tuttavia, osservando le feature map prodotte durante la propagazione, si nota come la loro intensità tenda a ridursi verso la fine della rete. Questo fenomeno evidenzia la presenza di un vero e proprio *information bottleneck*, che limita la capacità del modello di mantenere e propagare informazioni rilevanti.

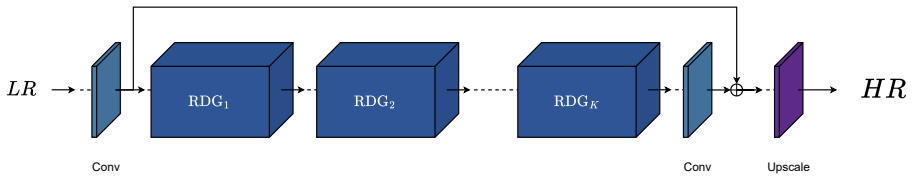

Figura 2.9: Architettura di DRCT.

[5] La sigmoid, a differenza della softmax, permette di attivare più canali contemporaneamente, evitando che l'aumento di un peso implichi la riduzione di un altro.

Per affrontare questo problema, è stato introdotto *DRCT* [9], un'architettura basata su SwinIR, visibile in Figura 2.9 e articolata in tre componenti principali.

- La testa del modello è costituita da una convoluzione 3×3, incaricata di estrarre le feature iniziali dall'immagine LR.
- Il corpo è formato da K *Residual Dense Group* (RDG), seguito poi da una convoluzione e una connessione residua globale.
- Infine, la coda è composta da un modulo di upsampling, che sfrutta le feature estratte per generare l'immagine HR finale.

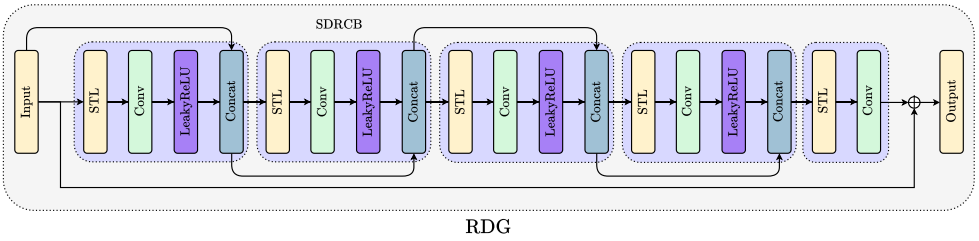

Figura 2.10: Struttura di un RDG.

Ogni RDG contiene 5 blocchi *Swin Dense Residual Connected Block* (SDRCB), cuore dell'architettura e mostrati in Figura 2.10. Questi layer hanno il compito di migliorare la trasmissione delle informazioni aggregando le feature estratte ad ogni livello, limitando la perdita di dettagli del bottleneck. In particolare, ogni RDG può essere definito come:

$$
\begin{aligned}
X_1 &= \mathrm{PE}\big(\mathcal{C}_\sigma\big(\mathrm{PUE}(\mathrm{STL}(X))\big)\big) \\
X_2 &= \mathrm{PE}\big(\mathcal{C}_\sigma\big(\mathrm{PUE}(\mathrm{STL}(\mathrm{Cat}(X, X_1)))\big)\big) \\
X_3 &= \mathrm{PE}\big(\mathcal{C}_\sigma\big(\mathrm{PUE}(\mathrm{STL}(\mathrm{Cat}(X, X_1, X_2)))\big)\big) \\
X_4 &= \mathrm{PE}\big(\mathcal{C}_\sigma\big(\mathrm{PUE}(\mathrm{STL}(\mathrm{Cat}(X, X_1, X_2, X_3)))\big)\big) \\
X_5 &= \mathrm{PE}\big(C_1\big(\mathrm{PUE}(\mathrm{STL}(\mathrm{Cat}(X, X_1, X_2, X_3, X_4)))\big)\big)
\end{aligned}
$$

$$B(X) = \alpha \cdot X_5 + X \tag{2.14}$$

In questa espressione, Cat$(\cdot)$ indica la concatenazione lungo la dimensione dei canali, mentre STL$(\cdot)$ rappresenta un *Swin Transformer Layer*. L'operazione PUE$(\cdot)$ (*Patch Unembedding*) trasforma la sequenza di patch in una mappa bidimensionale di feature, che viene poi passata a $\mathcal{C}_\sigma(\cdot)$, una convoluzione 1×1 seguita da LeakyReLU con slope 0.2. Per l'ultimo passaggio, C_1 è una convoluzione 1×1 senza attivazione, mentre $PE(\cdot)$ (*Patch Embedding*) riporta la mappa di feature in una sequenza di token. Infine, α è impostato a 0.2 per stabilizzare l'addestramento.

2.6.3 Swin2MoSE

A seguito dei miglioramenti ottenuti applicando gli Sparse Mixture of Experts nel campo della computer vision con i V-MoE, sono stati sviluppati diversi modelli specifici per la super-risoluzione. Fra questi vi è *Swin2MoSE* [10], che basandosi sull'architettura di *Swin2SR* [23] sostituisce i MLP all'interno dei blocchi Swin Transformer con layer MoE.

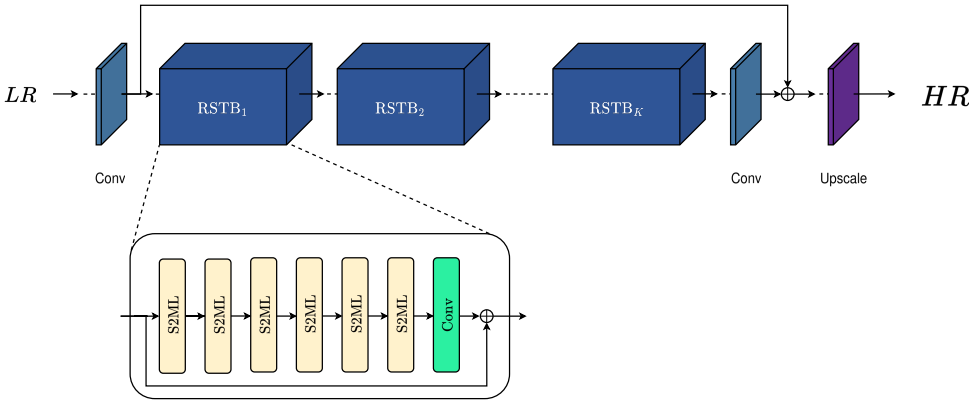

Figura 2.11: Architettura di Swin2MoSE.

Il modello, visibile in Figura 2.11, oltre a una convoluzione iniziale per estrarre le prime feature e il modulo finale di upsampling contiene per l'estrazione delle deep features K *Residual Swin Transformer Block* (RSTB).

Ogni RSTB è composto da *L Swin v2 MoSE Layer* (S2ML), la cui struttura è mostrata in Figura 2.12, e da una lunga skip connection seguita da una convoluzione.

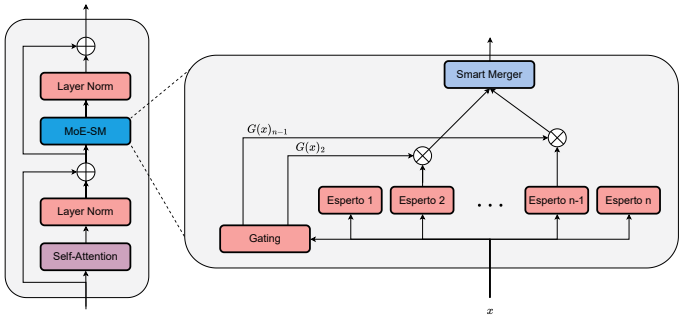

Figura 2.12: Struttura del blocco S2ML.

La novità principale introdotta da Swin2MoSE è un blocco MoE chiamato *MoE Smart Layer* (MoE-SM), che inserisce nella struttura classica di un layer MoE un layer di merging dell'output degli esperti detto Smart Merger (SM). Lo SM è una convoluzione 2D 3×3 che ha come canali in input il numero degli esperti e in output un singolo canale, in modo tale da imparare a combinare le uscite degli esperti in modo più efficace rispetto alla semplice somma pesata. Inoltre in sostituzione al classico meccanismo di routing per-token viene utilizzato un routing per-example dove ogni feature map nel batch viene assegnata a k esperti invece che prendere una decisione per ogni singolo token. Questo permette ad ogni esperto di specializzarsi su un certo tipo di immagine, considerando la variabilità delle immagini satellitari.

2.7 Setup Sperimentale

I dati forniti per il progetto si riferiscono ai seguenti quattro comuni dell'Emilia-Romagna colpiti da frane a seguito delle alluvioni del 2023:

- **Brisighella (RA)**
- **Casola Valsenio (RA)**
- **Modigliana (FC)**
- **Predappio (FC)**

Per ognuno di questi comuni sono stati forniti, in file in formato GeoTIFF, i seguenti dati:

- le immagini Sentinel-2 con risoluzione di 10 metri per pixel, acquisite prima e dopo l'evento franoso.
- immagini aeree con risoluzione di 2 metri per pixel, pre e post evento.
- NDVI calcolato dalle immagini aeree e la differenza pre e post NDVI da satellite.
- carte della pendenza.
- maschere di frana create manualmente da esperti del settore.

In particolare le immagini Sentinel-2 sono state upscalate a 2 metri per pixel tramite interpolazione in modo da allinearle con le immagini aeree. Ogni immagine contiene circa 8000×8000 pixel per coprire l'interezza del comune, corrispondenti a circa 256 km^2.

2.7.1 Dataset

Per la super-risoluzione i dati adatti al compito sono le coppie di immagini Sentinel-2 a bassa risoluzione (LR) e le rispettive immagini aeree ad alta risoluzione (HR), sia pre che post evento. Tuttavia analizzando le varie immagini sono state individuate alcune problematiche che possono influire negativamente sull'allenamento dei modelli. Ogni coppia presenta infatti differenze di luminosità e contrasto, piccole nuvole e zone ombrose dovute alla presenza di queste. Inoltre, le immagini pre evento risultano essere prese in periodi diversi dell'anno e presentano quindi aree non coerenti per la super-risoluzione. Questo fenomeno è facilmente osservabile nei campi, come si può notare dalla Figura 2.13. Considerando tutte queste problematiche, per il dataset finale sono state dunque utilizzate unicamente le coppie LR-HR dei dati post evento.

Figura 2.13: Differenze tra immagini pre evento in LR e HR.

Come dati di validazione sono state usate le immagini del comune di Modigliana, mentre per l'allenamento sono stati usati i dati degli altri tre comuni. È stato inoltre necessario normalizzare le immagini satellitari in modo da avere valori compresi tra 0 e 1 e per farlo è stata applicata la

seguente formula:

$$\text{immagine\_normalizzata} = \frac{\text{immagine}}{10000} \tag{2.15}$$

seguendo le indicazioni fornite dalla documentazione ufficiale di Sentinel-2 [24]. Essendo i comuni ritagliati dai dati Sentinel-2, le aree esterne al comune sono rappresentate da valori NaN e che sono state opportunamente mascherate durante l'allenamento sostituendole da valori posti a 0.

2.7.2 Preprocessing

Dalle immagini Sentinel-2 sono state estratte patch randomiche di dimensione 320×320 o 640×640 pixel, per poi essere downscalate di un fattore 5 tramite interpolazione bicubica per riportare i pixel alla dimensione originale di 10×10 metri. Le patch in input ai modelli di super-risoluzione sono quindi di dimensione 64×64 o 128×128 pixel. Nel caso di zone con valori NaN si è proceduto a scartare le patch che contenevano più del 20% di pixel non validi. Per aumentare la varietà del dataset di allenamento, sono stati applicati i seguenti metodi di data augmentation:

- Flip orizzontali e verticali: il flip di un'immagine lungo l'asse orizzontale o verticale rimane comunque un'immagine valida.
- Rotazioni di 90, 180 e 270 gradi: simile al flip, queste rotazioni mantengono la validità dell'immagine. L'utilizzo di multipli di 90 gradi evita la necessità di interpolazioni che potrebbero introdurre artefatti in quanto la griglia non sarebbe più allineata con l'originale.
- Variazione della luminosità o del contrasto: questo aiuta il modello a generalizzare meglio rispetto a diverse condizioni di illuminazione, in particolare considerando che i dati forniti hanno differenze importanti tra le immagini LR e HR.

Notare che il fattore di upsample di 5 è piuttosto elevato rispetto a quelli comunemente usati in letteratura di 2 e 4 e su cui sono stati testati i mo-

delli usati in questo lavoro, rendendo il problema della super-risoluzione più complesso.

2.7.3 Coda di Upsampling

Mentre la backbone del modello si occupa di estrarre le feature, la coda di upsampling si occupa di ricostruire l'immagine ad alta risoluzione a partire dalle feature estratte. Per questo motivo la scelta della coda di upsampling può influire molto sulla qualità dell'immagine finale. Poiché il compito di un upsample 5× non è trattato comunemente, a favore di fattori multipli di 2, si è cercato di adattare una delle code già esistenti per questi modelli. Il modulo ottenuto e definito *Nearest+Conv* è la base scelta per il testing negli esperimenti svolti in questo lavoro. Questo consiste in tre differenti upsample progressivi 1.25×, 2× e 2× ognuno dei quali è seguito da una convoluzione 3×3. Il primo aumento di risoluzione utilizza l'interpolazione bicubica, mentre i due successivi utilizzano l'interpolazione nearest neighbor, che utilizza il pixel più vicino per calcolare il valore del nuovo pixel.

2.7.4 Metriche di valutazione

Per valutare le prestazioni dei modelli sono state utilizzate tre metriche complementari: *PSNR*, *SSIM* e *LPIPS*. Queste misure forniscono prospettive differenti sulla qualità delle immagini ricostruite: la prima valuta la fedeltà numerica, la seconda la similarità strutturale, e la terza la percezione visiva.

PSNR: Il *Peak Signal-to-Noise Ratio* (PSNR) misura la qualità di ricostruzione sulla base dell'errore quadratico medio (MSE) tra l'immagine originale I e quella ricostruita K. Per immagini di dimensione $m \times n$ con C canali, l'MSE è definito come:

$$\text{MSE} = \frac{1}{m\,n\,C} \sum_{c=1}^{C} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \big(I_c(i,j) - K_c(i,j)\big)^2. \tag{2.16}$$

Il PSNR si ottiene da:

$$\text{PSNR} = 20\log_{10}\left(\frac{MAX_I^2}{\sqrt{\text{MSE}}}\right) = 20\log_{10}(MAX_I) - 10\log_{10}(\text{MSE}), \quad (2.17)$$

dove MAX_I è il valore massimo possibile per un pixel (pari a 1 nelle immagini normalizzate). Valori più alti di PSNR indicano una maggiore fedeltà numerica rispetto all'originale, anche se questa metrica non sempre riflette la percezione visiva umana.

SSIM: Lo *Structural Similarity Index* (SSIM) [25] è stato introdotto per superare le limitazioni di metriche tradizionali come il PSNR, che si basano unicamente sulla differenza numerica tra i pixel. A differenza di tali approcci, SSIM valuta la qualità di un'immagine in modo simili a quanto farebbe l'occhio umano, concentrandosi su tre componenti fondamentali: luminanza, contrasto, struttura. Combinando queste tre misure, SSIM quantifica quanto due immagini condividano lo stesso contenuto strutturale. Data una coppia di immagini x e y, SSIM è definito come:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (2.18)$$

dove μ_x e μ_y rappresentano la media (luminanza), σ_x^2 e σ_y^2 le varianze (contrasto), mentre σ_{xy} è la covarianza, che misura la similarità strutturale tra le due immagini. Le costanti C_1 e C_2 servono a evitare instabilità numeriche nei casi in cui i denominatori siano prossimi allo zero. Il valore restituito da SSIM è compreso tra 0 e 1, dove 1 indica una corrispondenza perfetta tra le immagini.

LPIPS: Infine, il *Learned Perceptual Image Patch Similarity* (LPIPS) [26] confronta le immagini nello spazio delle feature estratte da reti neurali già addestrate. A differenza di PSNR e SSIM, LPIPS è progettato per riflettere la percezione visiva umana, penalizzando le differenze che risultano percettivamente significative anche se numericamente piccole. Valori bassi di LPIPS indicano una maggiore similarità percettiva.

In sintesi, PSNR valuta la fedeltà numerica, SSIM la similarità strutturale e LPIPS la qualità percepita. La combinazione di queste metriche permette una valutazione più completa delle prestazioni del modello.

2.7.5 Funzioni di loss

Per l'addestramento dei modelli e i relativi esperimenti sono state utilizzate diverse funzioni di loss. La scelta della funzione di loss è cruciale in quanto guida il processo di ottimizzazione e influisce direttamente sulla qualità delle immagini ricostruite.

Charbonnier Loss: La *Charbonnier Loss* [27] è una variante più robusta delle tradizionali L1 e L2 (MSE) loss, ed è definita come:

$$\mathcal{L}_{\text{Charbonnier}}(x, y) = \sqrt{(x - y)^2 + \epsilon^2} \tag{2.19}$$

La L1 loss infatti per differenze molto piccole tra x e y può portare a gradienti nulli, mentre la L2 loss calcolando il quadrato dell'errore può essere troppo sensibile a grandi differenze. La Charbonnier Loss introduce un piccolo termine ϵ come si può vedere dalla formula, che evita problemi di instabilità numerica e garantisce gradienti non nulli anche per piccole differenze. Inoltre calcolando la radice quadrata dell'errore al quadrato, la Charbonnier Loss combina i vantaggi di L1 e L2, risultando quindi più robusta agli outlier rispetto alla L2 e più stabile della L1.

NCC Loss: La *Normalized Cross-Correlation* (NCC) [10] è una misura di similarità tra due immagini che può assumere valori tra -1 e 1, dove 1 indica una correlazione perfetta mentre -1 una anti-correlazione. NCC viene calcolata come:

$$\text{NCC}(x, y) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - \mu_x)(y_{ij} - \mu_y)}{\sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - \mu_x)^2 \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} (y_{ij} - \mu_y)^2}} \tag{2.20}$$

dove x, y sono i pixel dell'immagine ricostruita e quella originale, entrambe di dimensione $M \times N$, e μ_x, μ_y sono le loro medie. L'equazione viene applicata ad

ogni canale dell'immagine per poi fare la media dei risultati. Per usare NCC come funzione di loss e normalizzarla tra 0 e 1, si usa la seguente formula:

$$\mathcal{L}_{\text{NCC}} = 1 - \frac{1}{2}(\text{NCC} + 1) \tag{2.21}$$

Questa loss è particolarmente utile con immagini reali poiché è invariante rispetto a cambiamenti di luminosità e contrasto, in quanto un aumento di luminosità o contrasto non altera la correlazione tra le immagini. Un'immagine non ha un valore minore di NCC rispetto ad un'altra se questa è più luminosa o ha un contrasto maggiore e quindi ha l'obiettivo di preservare la struttura dell'immagine piuttosto che i valori assoluti dei pixel.

SSIM Loss: SSIM, introdotto in precedenza come metrica di valutazione, può essere utilizzato anche come funzione di perdita [25]. Dal momento che SSIM assume valori compresi tra 0 e 1, dove 1 indica immagini perfettamente simili, la formulazione della loss viene definita come:

$$\mathcal{L}_{\text{SSIM}}(x, y) = 1 - \text{SSIM}(x, y). \tag{2.22}$$

In questo modo, minimizzare la loss equivale a massimizzare la similarità strutturale tra immagine ricostruita e immagine originale.

HF Loss: La *High-Frequency Loss* (HF Loss) è stata introdotta per migliorare la qualità delle immagini ricostruite concentrandosi sui dettagli ad alta frequenza [23]. Questa loss viene calcolata applicando la seguente operazione:

$$\mathcal{L}_{\text{HF}}(x, y) = \|\text{HF}(x) - \text{HF}(y)\|_1 = \|(x - (x * b)) - (y - (y * b))\|_1 \tag{2.23}$$

dove b è un filtro di blur gaussiano e $*$ indica una convoluzione 5×5. In pratica, questa loss calcola il blur medio attorno a ogni pixel per poi sottrarlo dall'immagine originale, ottenendo così una mappa che enfatizza i dettagli.

LPIPS Loss: Anche LPIPS, già utilizzata come metrica, è stata anch'essa modificata per essere utilizzata come funzione di loss [28]. A differenza di

PSNR e SSIM che operano direttamente sui valori dei pixel, LPIPS confronta le immagini nello spazio delle feature estratte da reti neurali pre-addestrate, come ad esempio VGG16. In questo modo il modello non si limita a ridurre l'errore pixel per pixel, ma cerca di produrre delle immagini che per quanto numericamente possano differire leggermente dall'originale, risultano comunque più simili all'occhio umano.

Capitolo 3

Esperimenti

3.1 Modelli

Dati i 3 modelli presi in considerazione, si è deciso di testarli seguendo le configurazioni più simili possibili a quelle usate in letteratura ma cercando comunque di mantenere un numero di parametri simile tra questi. L'allenamento e i rispettivi iperparametri sono stati mantenuti costanti tra i modelli, utilizzando come funzione di perdita Charbonnier e AdamW come ottimizzatore. La metrica principale, il PSNR, è stata monitorata durante l'allenamento e calcolata sul test set alla fine dell'allenamento per vedere l'andamento durante questo. In particolare si sono allenati i modelli su 10k patch 64×64 per 100 epoche e un batch size di 8. Come optimizer è stato utilizzato AdamW con un learning rate iniziale di $2e-4$ e come scheduler un MultiStepLR, dove il learning rate viene ridotto di una determinata percentuale al raggiungimento di determinati checkpoint. In questo caso su un totale di 125k steps, i quali vengono calcolati tramite la seguente formula:

$$\text{steps\_totali} = \frac{\text{numero\_patches}}{\text{batch\_size}} \times \text{numero\_epoche} \tag{3.1}$$

il learning rate è stato dimezzato al raggiungimento delle seguenti percentuali di steps totali:

- 40%: 50000 steps.

- 60%: 75000 steps.
- 75%: 93750 steps.
- 80%: 100000 steps.
- 85%: 106250 steps.

La scelta dei checkpoint è basata sull'allenamento per il modello DRCT presentato nell'articolo originale. Come funzione di loss è stata invece usata unicamente la Charbonnier per avere un confronto basato unicamente sul minimizzare l'errore pixel-per-pixel. Per quanto riguarda le architetture, le configurazioni sono le seguenti:

RCAN: Il modello utilizza 10 Residual Groups, ognuno con 20 RCAB, portando quindi il numero di parametri a circa 15 milioni.

DRCT: Il modello utilizza 6 RDG, con un α di 0.2 e una dimensione di embedding di 180. Il growth rate è di 32, ovvero ogni layer SDRCB produce un output con 32 canali che verranno poi concatenati nel layer successivo. L'ultimo SDRCB genera un output di 180 canali dovendo essere sommato al residuo. Il numero di parametri è di circa 14 milioni.

Swin2MoSE: Il modello utilizza la stessa architettura presentata nell'articolo con 6 RSTB e 6 Swin Transformer Block per ogni RSTB. La window size per la self-attention è stata impostata a 16 e per il modulo di MoE sono stati usati 8 esperti con un top-k di 2. Il numero di parametri sparsi è di 46 milioni ma il numero di parametri effettivi è di circa 17 milioni.

3.2 Numero di epoche

Gli iperparametri scelti per il confronto tra i vari modelli sono stati scelti tenendo anche conto delle limitazioni hardware e quindi del tempo di allenamento. L'allenamento di 100 epoche su 10k patch 64×64 per un totale di 125k steps, per quanto possa sembrare sufficiente, è comunque inferiore rispetto a quanto solitamente richiesto dai modelli basati su transformer. Inoltre vista la dimensione ridotta del dataset, si è deciso di testare un allenamento più lungo per valutare se questo possa portare a un miglioramento delle performance. Si è quindi deciso di allenare due modelli, RCAN e DRCT, così da avere un confronto tra un modello CNN e uno transformer-based, per 300 epoche ma con i restanti iperparametri invariati.

3.3 Dimensione dataset

Generalmente i modelli di super-risoluzione basati su transformer richiedono un allenamento più lungo rispetto ai modelli CNN per raggiungere buone performance. Per questo solitamente vengono pre-allenati su dataset grandi come ImageNet che contiene più di un milione di immagini e poi viene fatto un fine tuning sul dataset specifico. In questo caso, dato che il dataset è relativamente piccolo, si è deciso di testare il modello allenandolo su diverse porzioni del dataset per valutare se un eventuale ampliamento del dataset possa portare a un miglioramento delle performance. Considerando che le patch sono estratte randomicamente dalle immagini dei 3 comuni disponibili per l'allenamento, in quanto uno è riservato per la validazione, si è deciso di valutare le prestazioni di un modello allenato su un singolo comune e di un modello allenato su tutti e 3 i comuni.

3.4 Dati sintetici e Fine Tuning

Considerata la differenza nello spettro delle immagini satellitari rispetto alla immagini ad alta risoluzione utilizzate per l'allenamento, si è deciso di

provare a degradare le immagini ad alta risoluzione per avvicinarle alla qualità delle immagini LR, come si può vedere dalla Figura 3.1. Per fare questo si è applicato il semplice downsample bicubico $5\times$, lo stesso che viene applicato anche ai dati satellitari per farli tornare alla risoluzione originale. Il modello RCAN è stato dunque pre-allenato con gli stessi iperparametri usati per il confronto tra i modelli, seguito poi da un fine tuning sui dati reali.

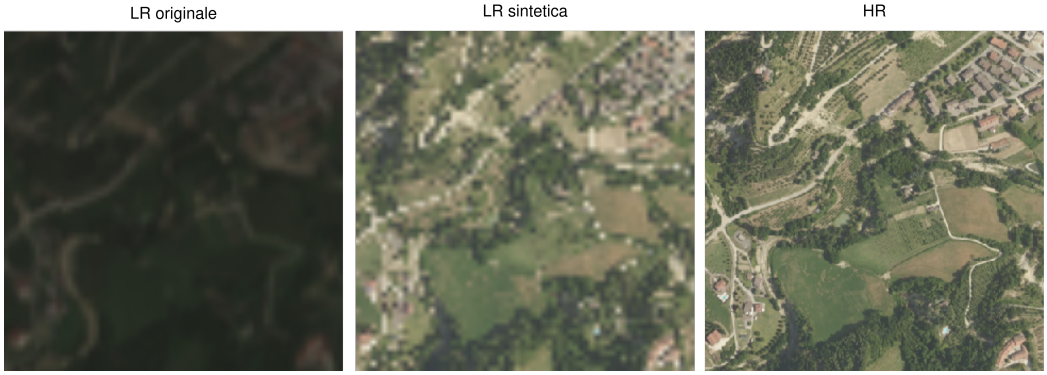

Figura 3.1: Confronto tra immagine satellitare, immagine sintetica e immagine originale.

Il fine tuning è stato fatto caricando i pesi del pre-allenamento e non congelando nessun layer vista la notevole differenza tra i dati sintetici e i dati reali, come si può vedere dalla Figura 3.1. Gli iperparametri sono stati mantenuti gli stessi ad eccezione del learning rate che è stato impostato a $5e-5$ e il numero di epoche che è stato portato a 50.

3.5 Funzioni di Loss

Utilizzare una singola funzione di perdita può non essere sufficiente per catturare tutte le caratteristiche desiderate nell'immagine ricostruita. Per esempio una funzione di perdita basata sulla differenza pixel-per-pixel, come la Charbonnier, può portare a immagini con un alto PSNR medio ma che risultano sfocate o prive di dettagli fini, oppure una funzione di perdita basata su caratteristiche strutturali, come SSIM, può portare a immagini con dettagli

più nitidi ma con un PSNR più basso. Seguendo questa logica, si è deciso di testare diverse combinazioni di funzioni di perdita come già avviene in letteratura per valutare le differenze nella qualità dell'immagine ricostruita. La rete scelta per il confronto è RCAN, con gli stessi iperparametri usati per il confronto tra i modelli. Per le loss principali quali Charbonnier e NCC, il peso è stato impostato a 1, mentre per le loss secondarie quali SSIM, LPIPS e HF, il contributo è stato impostato a 0.1 .

Charbonnier e SSIM: vista la differenza tra le immagini satellitari e le immagini aeree come la luminosità e il contrasto, si è deciso di testare l'aggiunta di SSIM come funzione di loss, con l'obiettivo di migliorare la struttura dell'immagine ricostruita.

Charbonnier e LPIPS: considerato il compito di produrre un upscale a 5×, si è deciso di testare l'aggiunta di LPIPS come funzione di loss, con l'obiettivo di migliorare i bordi e le texture che possono aiutare nella successiva fase di segmentazione delle frane.

Charbonnier e HF: si è voluto testare l'aggiunta di una funzione di perdita basata sulle alte frequenze, con l'obiettivo di migliorare i dettagli fini dell'immagine ricostruita.

NCC e SSIM: questa scelta è stata basata sui risultati ottenuti nell'articolo del modello Swin2MoSE, dove questa combinazione risultava la migliore a discapito della funzione di loss Mean Squared Error (MSE), simile alla Charbonnier.

Capitolo 4

Risultati e Analisi

In questo capitolo vengono presentati i risultati ottenuti dai vari esperimenti descritti nel capitolo precedente. Per ognuno di questi vengono mostrati i risultati basati sulle metriche descritte e viene fornita un'analisi dei dati raccolti.

4.1 Confronto architetture

Una volta allenati i 3 rispettivi modelli come descritto nella sezione esperimenti per il confronto tra questi, i risultati ottenuti sono i seguenti:

Modello	**PSNR**	**PSNR-RGB**	**PSNR-NIR**	**SSIM**	**LPIPS**
RCAN	**21.48**	**22.07**	**20.19**	**0.574**	0.592
DRCT	21.15	21.64	20.03	0.539	0.607
Swin2MoSE	21.11	21.61	20.00	0.538	**0.614**

Tabella 4.1: Risultati ottenuti dai modelli testati sul dataset di validazione.

Come si può vedere dalla Tabella 4.1, il modello che ottiene i risultati migliori in quasi tutte le metriche è RCAN. I modelli SOTA attuali per la super risoluzione, basati su transformer, prediligono un pre-addestramento su dataset molto grandi, seguito da un fine-tune sui dati a propria disposizone. No-

nostante la dimensione ridotta del dataset a disposizione, i risultati tendono a contraddire la letteratura secondo la quale, anche con dataset relativamente piccoli, i modelli transformer-based dovrebbero superare le performance di quelli CNN [22].

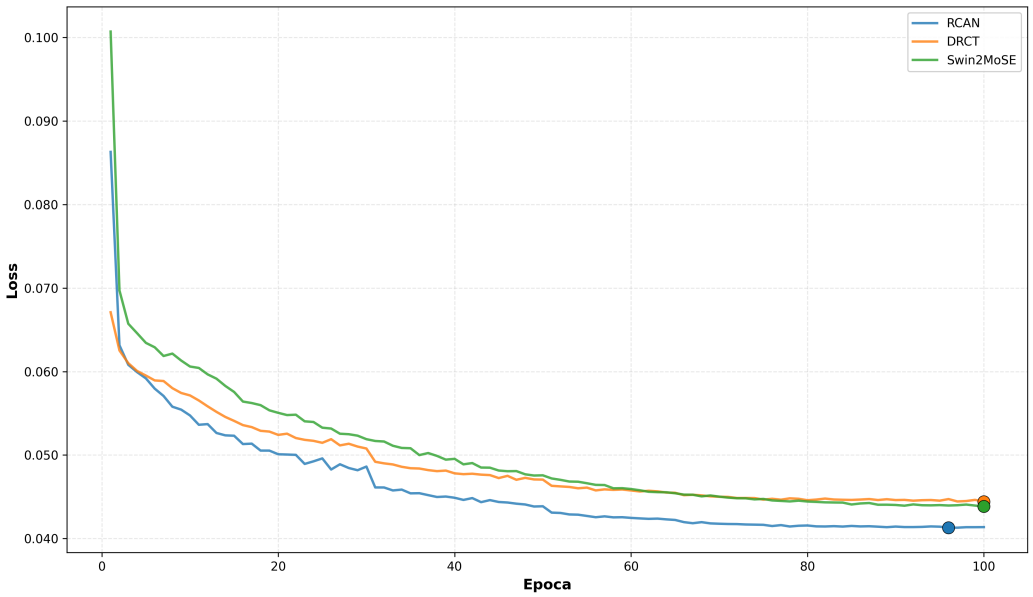

Figura 4.1: Andamento della loss durante l'allenamento.

Tracciando il grafico della loss, visibile nella Figura 4.1, si può notare come, tenendo anche conto della riduzione del learning rate ai vari checkpoint impostati grazie allo scheduler, il modello RCAN tenda a convergere più velocemente rispetto agli altri due modelli. L'allenamento di 100 epoche su 10k patch era stato scelto per bilanciare il tempo di allenamento e la qualità del modello, tuttavia potrebbe essere comunque insufficiente per i modelli basati su transformer, data la loro lenta convergenza.

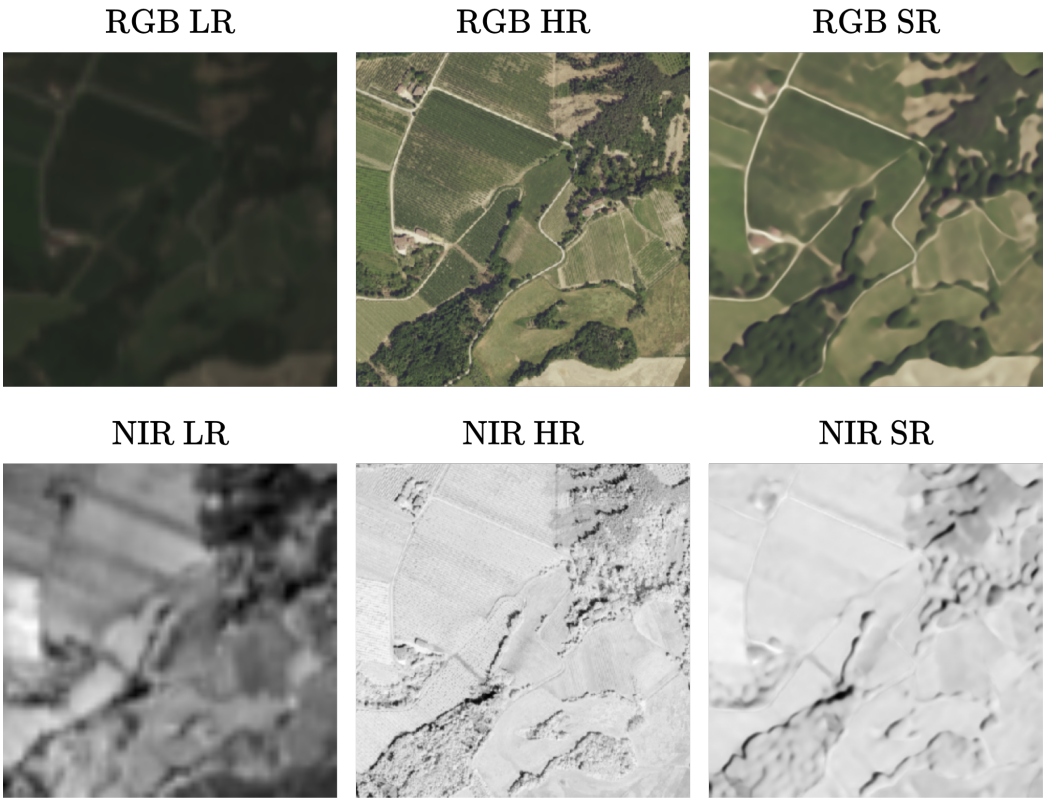

Figura 4.2: Confronto tra LR, HR e ricostruzione sui canali RGB e NIR per il modello RCAN.

Analizzando il PSNR su RGB e NIR si può notare come il secondo abbia valori inferiori rispetto al primo. Questo è probabilmente dovuto al fatto che la differenza qualitativa tra l'immagine LR e HR è più marcata nel canale NIR rispetto ai canali RGB, rendendo più difficile la ricostruzione di quest'ultimo, come osservabile nella Figura 4.2. Andando ad analizzare i risultati assoluti ottenuti si può vedere come le immagini prodotte dai vari modelli abbiano una qualità nettamente più elevata rispetto all'immagine originale, come si può notare dalla Figura 4.3. Da queste infatti, per quanto vi sia una perdita di dettagli come texture o bordi netti rispetto all'immagine ad alta risoluzione, è possibile individuare con maggior precisione le strade, la vegetazione e le zone soggette a frana.

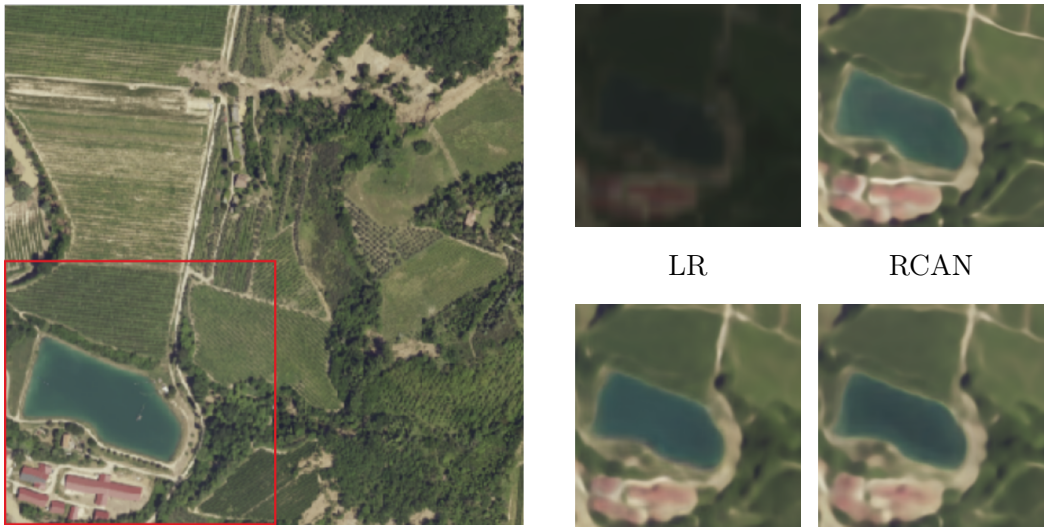

Figura 4.3: Confronto visivo: immagine completa e crop dei risultati ottenuti dai diversi modelli.

4.2 Impatto della durata dell'addestramento

A seguito dei risultati ottenuti dall'allenamento dei vari modelli su 100 epoche, i modelli transformer come detto hanno performato peggio rispetto alla rete basata su CNN. Per questo motivo è ancora più importante, confrontare quanto la durata dell'allenamento influisca della qualità dell'immagine ricostruita, in quanto questi in letteratura sono noti per richiedere un allenamento più lungo. Il modello Swin2MoSE non è stato incluso in questo confronto poiché ha performato in modo molto simile a DRCT, considerato anche che entrambi i modelli sono basati su transformer, e quindi per limiti di tempo e di risorse computazionali si è deciso di non includerlo. I valori ottenuti dai due allenamenti sono i seguenti:

Modello	Epoche	PSNR	PSNR-RGB	PSNR-NIR	SSIM	LPIPS
RCAN	100	21.48	22.07	20.19	0.574	0.592
RCAN	300	**21.62**	**22.14**	**20.45**	**0.586**	0.586
DRCT	100	21.15	21.64	20.03	0.539	**0.607**
DRCT	300	21.19	21.72	20.00	0.563	0.591

Tabella 4.2: Risultati ottenuti dai modelli testati sul dataset di validazione.

Sia RCAN che DRCT hanno beneficiato di un allenamento prolungato, come osservabile nella Tabella 4.2, per quanto il miglioramento non sia stato così marcato come ci si poteva aspettare. Inoltre l'andamento della loss, visibile nella Figura 4.4, mostra come entrambi i modelli tendano a raggiungere gli stessi valori di loss, nonostante la differenza di prestazioni misurate dalle metriche e la convergenza più rapida di RCAN in entrambi i casi. Considerando anche l'optimizer utilizzato, da queste informazioni si può ipotizzare che:

- RCAN, seppur l'addestramento più lungo abbia portato a miglioramenti, probabilmente richiede un addestramento più corto con un learning rate che decresce più rapidamente.

- DRCT, al contrario, a livello di loss beneficia molto di un allenamento più lungo, tuttavia questo non si traduce in un netto miglioramento delle metriche, probabilmente a causa di un overfitting sui dati.

La quantità ridotta di dati a disposizione, unita alla complessità del modello, potrebbe essere la causa di overfitting. Con un dataset più grande, un allenamento più lungo probabilmente porterebbe a miglioramenti più marcati.

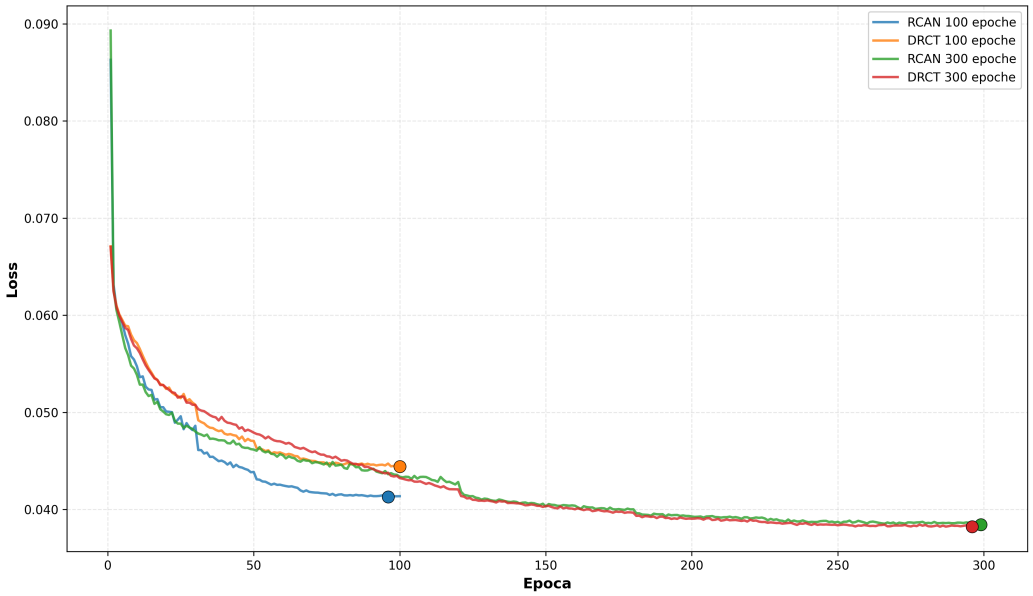

Figura 4.4: Andamento della loss con 100 e 300 epoche.

4.3 Variazione del dataset

Basandosi sulle metriche ottenute dai precedenti modelli allenati sull'intero dataset, si è confrontato come le prestazioni di RCAN e DRCT variano utilizzando un solo comune. I risultati ottenuti sono i seguenti:

Modello	Comuni	PSNR	PSNR-RGB	PSNR-NIR	SSIM	LPIPS
RCAN	1	21.38	21.83	**20.35**	**0.576**	0.589
RCAN	3	**21.48**	**22.07**	20.19	0.574	0.592
DRCT	1	21.05	21.52	19.98	0.548	0.600
DRCT	3	21.15	21.64	20.03	0.539	**0.607**

Tabella 4.3: Risultati ottenuti dai modelli testati sul dataset di validazione.

Le performance ottenute, visibili nella Tabella 4.3, come prevedibile sono peggiorate riducendo la quantità di dati a disposizione. Considerando anche la Figura 4.5 della discesa della loss, si può notare come il modello RCAN

effettivamente riduce la loss coerentemente con il numero di comuni usati per l'allenamento. Le CNN infatti grazie al bias induttivo intrinseco alla loro architettura, tendono a performare meglio su dataset di dimensioni ridotte. Interessante invece come il modello DRCT, riducendo i dati forniti per l'allenamento, abbia un minimo di loss più basso ma con metriche peggiori, probabilmente in quanto essendo la rete più complessa, tende a overfittare più facilmente su un dataset più piccolo. Con una eventuale espansione progressiva del dataset a disposizione, ci si aspetta dopo una prima fase di incremento di prestazioni di RCAN, che i modelli transformer-based possano superarlo vista la maggiore capacità mostrata.

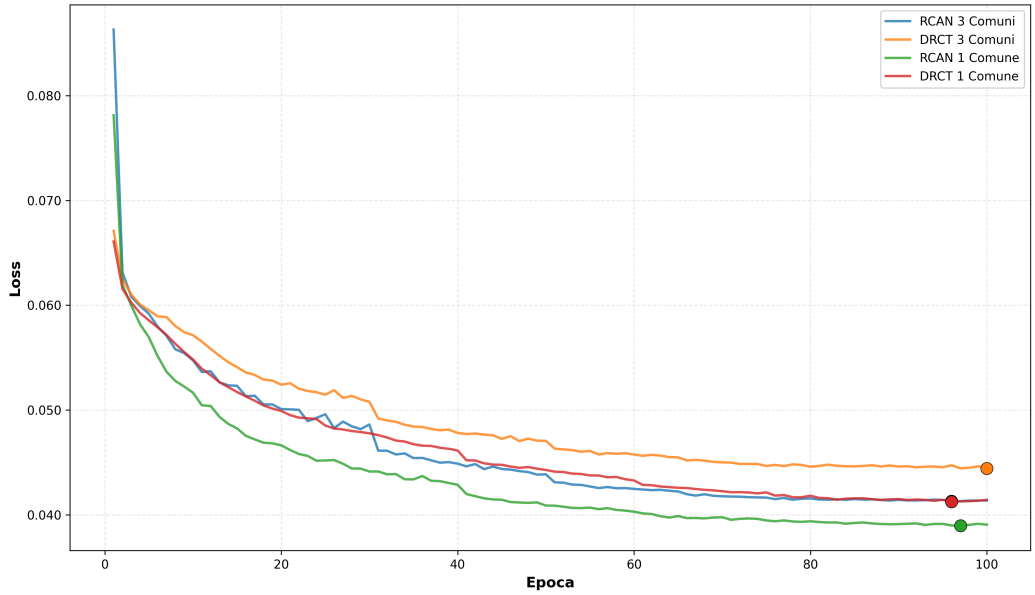

Figura 4.5: Andamento della loss allenando su 1 e 3 comuni.

4.4 Prestazioni con dati sintetici e Fine Tuning

Dal fine tuning sui dati reali dopo un primo allenamento con i dati sintetici, i risultati ottenuti sono i seguenti:

Allenamento	PSNR	PSNR-RGB	PSNR-NIR	SSIM	LPIPS
Dati Sintetici	24.25	24.79	22.98	0.657	0.503
Fine Tuning	21.36	21.90	20.18	0.547	**0.619**
Standard	**21.48**	**22.07**	**20.19**	**0.574**	0.592

Tabella 4.4: Risultati dell'esperimento con dati sintetici e fine tuning.

Come si può intuire leggendo la Tabella 4.4, un allenamento su dati sintetici, per quanto dovrebbe permettere al modello di apprendere a ricostruire immagini più dettagliate, non ha fornito i risultati sperati.

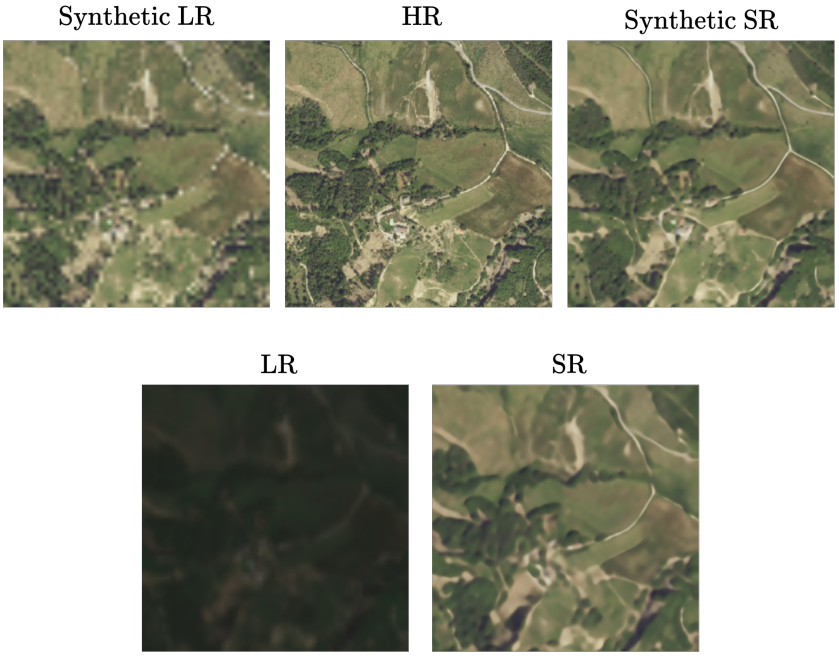

Figura 4.6: Confronto tra ricostruzione dei dati sintetici e dati reali tramite fine tuning.

Il modello infatti con i dati sintetici riesce a produrre immagini di qualità migliore, come osservabile in Figura 4.6, probabilmente a causa della migliore base di partenza. Tuttavia con la tipologia di fine tuning adottata la ricostruzione delle immagini reali è peggiorata rispetto a un addestramento tradizionale. Una possibile causa potrebbe essere la differenza tra i dati sintetici e i dati reali, che nonostante il downsample bicubico, rimangono comunque molto diversi tra loro. Rimane da valutare se una funzione di degradazione più complessa e realistica o un approccio al fine-tuning diverso possano portare a risultati migliori.

4.5 Utilizzo di Loss Multiple

Per quanto riguarda l'utilizzo di più funzioni di perdita, i risultati degli esperimenti condotti sono i seguenti:

Loss	**PSNR**	**PSNR-RGB**	**PSNR-NIR**	**SSIM**	**LPIPS**
Ch.	21.48	**22.07**	20.19	0.574	**0.592**
Ch. + SSIM	21.50	22.03	20.34	**0.589**	0.576
Ch. + LPIPS	21.40	21.84	**20.37**	0.547	0.357
Ch. + HF	**21.52**	**22.07**	20.29	0.578	**0.592**
NCC + SSIM	21.51	21.97	20.46	0.585	0.581

Tabella 4.5: Risultati delle diverse combinazioni di funzioni di perdita testate.

Come si può osservare dalla Tabella 4.5, l'utilizzo di più loss ha portato a miglioramenti complessivamente ridotti rispetto all'utilizzo della sola Charbonnier. L'aggiunta di LPIPS, nonostante la graduale diminuzione di questa durante l'allenamento, ha peggiorato tutti i risultati, compresa la metrica LPIPS stessa. Come visibile dalla Figura 4.7, la ricostruzione presenta un pattern ripetuto, probabilmente dovuto alla natura della loss. La discesa della loss di LPIPS durante l'allenamento, in contrasto con i risultati ottenuti

e i pattern visibili, suggerisce che il modello non stia migliorando la qualità dell'immagine ma stia overfittando, generando artefatti indesiderati. È possibile che il peso assegnato a LPIPS (0.1) sia troppo elevato, portando il modello a concentrarsi eccessivamente su questa loss e sarebbe dunque utile sperimentare con pesi molto più bassi per vedere se questo porti ai risultati sperati senza introdurre artefatti. In generale, dai vari esperimenti condotti emerge come i valori di LPIPS tendano a essere inversamente proporzionali al PSNR: l'aumento di uno corrisponde alla diminuzione dell'altro, probabilmente dovuto al fatto che miglioramenti di singoli pixel sparsi possano non essere percepiti come miglioramenti a livello di feature globali.

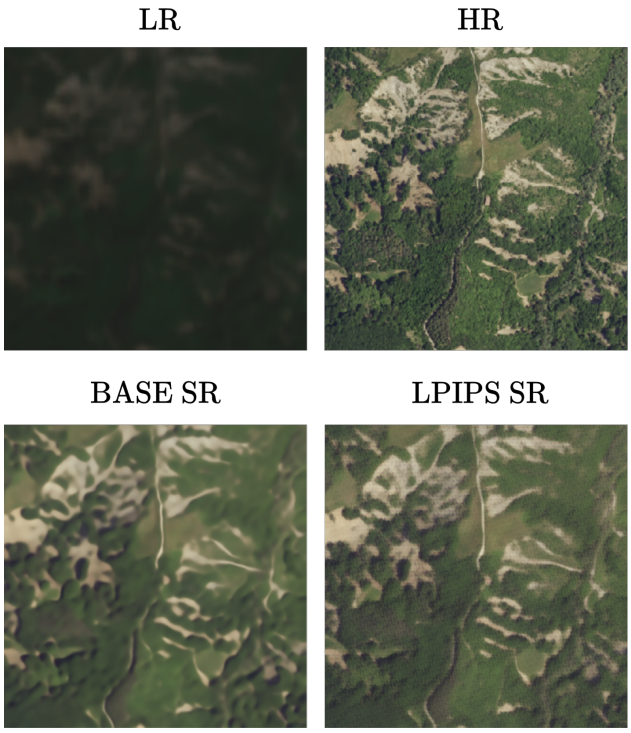

Figura 4.7: Esempio di artefatti generati dall'utilizzo di LPIPS come funzione di loss aggiuntiva rispetto al modello standard.

In ogni caso considerato che la valutazione è stata eseguita su un numero di patch elevato, mantenendo iperparametri invariati e un seed per la rando-

micità fisso, i risultati nonostante la loro vicinanza sono comunque affidabili. La combinazione che nel complesso è risultata più efficace è stata Charbonnier e HF, che ha migliorato ogni metrica tranne LPIPS, che è rimasta invariata. L'utilizzo di più funzioni di loss, sebbene in letteratura sia spesso vantaggioso, in questo caso non ha portato a miglioramenti significativi. Una possibile spiegazione è data dalla ancora bassa qualità delle immagini ricostruite, che nonostante gli ottimi risultati ottenuti rispetto alle immagini LR di partenza, rimangono comunque lontane dalle HR. Per questo motivo, la rete tende a concentrarsi principalmente nell'aumentare la qualità generale dell'immagine, piuttosto che migliorare dettagli specifici come texture o bordi netti, che potrebbero essere enfatizzati da funzioni di perdita aggiuntive.

Conclusioni

In questo lavoro di tesi sono stati studiati e confrontati diversi modelli di super-risoluzione per immagini satellitari, con l'obiettivo di migliorare la risoluzione spaziale delle immagini Sentinel-2 per il monitoraggio delle frane. Sono stati selezionati tre modelli rappresentativi delle due principali architetture utilizzate per la super-risoluzione: RCAN, basato su reti neurali convoluzionali (CNN), e DRCT e Swin2MoSE, basati su Transformer. I modelli sono stati allenati e valutati sui dati di 4 comuni italiani, utilizzando metriche quali PSNR, SSIM e LPIPS per misurare la qualità delle ricostruzioni. I risultati ottenuti mostrano che RCAN ha raggiunto le migliori prestazioni in termini di PSNR e SSIM, nonostante la letteratura suggerisca che i modelli basati su Transformer superino le CNN per compiti di super-risoluzione. Questo potrebbe essere dovuto al fatto che i modelli Transformer richiedono generalmente dataset più grandi e tempi di addestramento più lunghi per superare le prestazioni delle CNN, specialmente in scenari con un fattore di upscale elevato come 5×. Swin2MoSE nel test di confronto ha mostrato prestazioni comparabili a DRCT, motivo per cui i successivi esperimenti fatti utilizzano un solo modello Transformer, scegliendo DRCT per la sua implementazione più semplice e diretta. Dai test effettuati variando il numero di epoche e la dimensione del dataset per l'allenamento di RCAN e DRCT, si è osservato che vista la quantità limitata di dati a disposizione, il modello CNN ha beneficiato maggiormente sia di un addestramento più lungo che di un dataset più ampio. L'andamento della loss tuttavia suggerisce come DRCT, e presumibilmente anche Swin2MoSE, sottoperformi a causa del dataset piccolo, e

che una eventuale espansione possa portare a un superamento di RCAN. In quanto le immagini LR fornite presentano una differenza spettrale significativa rispetto alle immagini HR, si è deciso di provare a pre-allenare RCAN su immagini degradate artificialmente con un downsample bicubico, per poi eseguire un fine-tuning sul dataset reale. Per quanto il pre-allenamento, vista la migliore qualità delle immagini sintetiche, abbia portato a un miglioramento delle prestazioni in termini di PSNR e SSIM, il fine-tuning sul dataset reale non ha portato a benefici rispetto ad un addestramento classico. L'utilizzo di loss multiple invece, come viene generalmente fatto, ha portato a un incremento seppur minimo delle prestazioni, probabilmente dovuto al fatto che le immagini ricostruite non hanno una qualità sufficientemente alta per trarre vantaggio da loss che migliorano dettagli più fini rispetto alla qualità generale.

Nel complesso, i risultati ottenuti dimostrano l'efficacia dei modelli di super-risoluzione studiati per migliorare la risoluzione delle immagini satellitari. Nonostante il fattore di scala elevato e la scarsa qualità delle immagini a bassa risoluzione le diverse architetture hanno ricostruito immagini di qualità molto superiore a quella di partenza, con RCAN che si è dimostrato il modello più performante. Questi risultati dimostrano la fattibilità delle tecniche utilizzate, permettendo in particolare l'utilizzo dei dati Sentinel-2 per compiti che richiedono dettagli maggiori rispetto a quelli generalmente disponibili. Nonostante resti da valutare l'effettivo impatto delle immagini ricostruite all'interno della pipeline per il monitoraggio delle frane, i risultati ottenuti per questo step sono promettenti.

Come passi successivi di questo lavoro, si potrebbe considerare l'ampliamento del dataset utilizzando alcuni di quelli già esistenti per la super-risoluzione di immagini satellitari. Inoltre per limiti temporali non è stato possibile testare l'utilizzo di code di upsampling più complesse e/o specifiche per immagini RGB+NIR, che potrebbero portare a ulteriori miglioramenti. Andrebbe inoltre considerato di utilizzare *Generative Adversarial Network* (GAN), per migliorare in modo significativo la qualità visiva delle immagini

ricostruite, per quanto sia necessario verificare che i dettagli aggiunti, in quanto artefatti generati per una maggiore qualità percepita, non interferiscano con la successiva fase di segmentazione delle frane. Per limiti hardware non è stato invece possibile provare a passare in input patch di dimensioni maggiori, che potrebbero migliorare le prestazioni dei modelli, fornendo un contesto più ampio per la ricostruzione. Infine nonostante il modello RCAN abbia ottenuto i risultati migliori, sarebbe da integrare le migliorie implementate in Swin2MoSE all'interno di DRCT e valutare le sue prestazioni.

Bibliografia

[1] Yu Duan, Mingtao Ding, Yufeng He, Hao Zheng, Ricardo Delgado-Téllez, Sergey Sokratov, Francisco Dourado e Sven Fuchs. "Global projections of future landslide susceptibility under climate change". In: *Geoscience Frontiers* 16.4 (2025), p. 102074. ISSN: 1674-9871. DOI: https://doi.org/10.1016/j.gsf.2025.102074. URL: https://www.sciencedirect.com/science/article/pii/S1674987125000799.

[2] Lucas Pedrosa Soares, Helen Cristina Dias, Guilherme Pereira Bento Garcia e Carlos Henrique Grohmann. "Landslide Segmentation with Deep Learning: Evaluating Model Generalization in Rainfall-Induced Landslides in Brazil". In: *Remote Sensing* 14.9 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14092237. URL: https://www.mdpi.com/2072-4292/14/9/2237.

[3] Alessandro Novellino, Catherine Pennington, Kathryn Leeming, Sophie Taylor, Itahisa Gonzalez Alvarez, Emma McAllister, Christian Arnhardt e Annie Winson. "Mapping landslides from space: A review". In: *Landslides* 21.5 (2024), pp. 1041–1052. DOI: 10.1007/s10346-024-02215-x. URL: https://doi.org/10.1007/s10346-024-02215-x.

[4] Muratbek Kudaibergenov, Serik Nurakynov, Berik Iskakov, Gulnara Iskaliyeva, Yelaman Maksum, Elmira Orynbassarova, Bakytzhan Akhmetov e Nurmakhambet Sydyk. "Application of Artificial Intelligence in Landslide Susceptibility Assessment: Review of Recent Progress". In: *Remote Sensing* 17.1 (2025). ISSN: 2072-4292. DOI: 10.3390/rs17010034. URL: https://www.mdpi.com/2072-4292/17/1/34.

[5] European Space Agency. *Sentinel-2 Data Information*. URL: https://sentiwiki.copernicus.eu/web/s2-mission.

[6] European Space Agency. *ESA Sentinel-2*. URL: https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2.

[7] Sha Huang, Lina Tang, Joseph P. Hupy, Yang Wang e Guofan Shao. "A commentary review on the use of normalized difference vegetation index (NDVI) in the era of popular remote sensing". In: *Journal of Forestry Research* 32.1 (2021), pp. 1–6. ISSN: 1993-0607. DOI: 10.1007/s11676-020-01155-1. URL: https://doi.org/10.1007/s11676-020-01155-1.

[8] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong e Yun Fu. *Image Super-Resolution Using Very Deep Residual Channel Attention Networks*. 2018. arXiv: 1807.02758 [cs.CV]. URL: https://arxiv.org/abs/1807.02758.

[9] Chih-Chung Hsu, Chia-Ming Lee e Yi-Shiuan Chou. "DRCT: Saving Image Super-resolution away from Information Bottleneck". In: (2024). arXiv: 2404.00722 [cs.CV]. URL: https://arxiv.org/abs/2404.00722.

[10] Leonardo Rossi, Vittorio Bernuzzi, Tomaso Fontanini, Massimo Bertozzi e Andrea Prati. "Swin2-MoSE: A new single image supersolution model for remote sensing". In: *IET Image Processing* 19.1 (gen. 2025). ISSN: 1751-9667. DOI: 10.1049/ipr2.13303. URL: http://dx.doi.org/10.1049/ipr2.13303.

[11] Andreas Lugmayr, Martin Danelljan, Luc Van Gool e Radu Timofte. *SRFlow: Learning the Super-Resolution Space with Normalizing Flow*. 2020. arXiv: 2006.14200 [cs.CV]. URL: https://arxiv.org/abs/2006.14200.

[12] Yunliang Qi, Meng Lou, Yimin Liu, Lu Li, Zhen Yang e Wen Nie. *Advancing Image Super-resolution Techniques in Remote Sensing: A Comprehensive Survey*. 2025. arXiv: 2505.23248 [eess.IV]. URL: https://arxiv.org/abs/2505.23248.

[13] Honggang Chen, Xiaohai He, Linbo Qing, Yuanyuan Wu, Chao Ren e Ce Zhu. *Real-World Single Image Super-Resolution: A Brief Review*. 2021. arXiv: 2103.02368 [eess.IV]. URL: https://arxiv.org/abs/2103.02368.

[14] Wenjie Luo, Yujia Li, Raquel Urtasun e Richard Zemel. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. A cura di D. Lee, M. Sugiyama, U. Luxburg, I. Guyon e R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf.

[15] Chao Dong et al. *Image Super-Resolution Using Deep Convolutional Networks*. 2015. URL: https://arxiv.org/abs/1501.00092.

[16] Jiwon Kim et al. *Accurate Image Super-Resolution Using Very Deep Convolutional Networks*. 2016. URL: https://arxiv.org/abs/1511.04587.

[17] Bee Lim et al. *Enhanced Deep Residual Networks for Single Image Super-Resolution*. 2017. URL: https://arxiv.org/abs/1707.02921.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser e Illia Polosukhin. "Attention Is All You Need". In: (2017). arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit e Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image

Recognition at Scale". In: (2021). arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton e Jeff Dean. "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". In: (2017). arXiv: 1701.06538 [cs.LG]. URL: https://arxiv.org/abs/1701.06538.

[21] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers e Neil Houlsby. *Scaling Vision with Sparse Mixture of Experts*. 2021. arXiv: 2106.05974 [cs.CV]. URL: https://arxiv.org/abs/2106.05974.

[22] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool e Radu Timofte. "SwinIR: Image Restoration Using Swin Transformer". In: (2021). arXiv: 2108.10257 [eess.IV]. URL: https://arxiv.org/abs/2108.10257.

[23] Marcos V. Conde, Ui-Jin Choi, Maxime Burchi e Radu Timofte. *Swin2SR: SwinV2 Transformer for Compressed Image Super-Resolution and Restoration*. 2022. arXiv: 2209.11345 [cs.CV]. URL: https://arxiv.org/abs/2209.11345.

[24] European Space Agency. *Sentinel-2 Data Processing*. URL: https://docs.sentinel-hub.com/api/latest/data/sentinel-2-l2a/.

[25] Adéla Hamplová, Tomáš Novák, Miroslav Žáček e Jiří Brožek. "Effects of Normalised SSIM Loss on Super-Resolution Tasks". In: *CMES - Computer Modeling in Engineering and Sciences* 143.3 (2025), pp. 3329–3349. ISSN: 1526-1492. DOI: https://doi.org/10.32604/cmes.2025.066025. URL: https://www.sciencedirect.com/science/article/pii/S1526149225001997.

[26] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. URL: https://arxiv.org/abs/1801.03924.

[27] Jonathan T. Barron. *A General and Adaptive Robust Loss Function*. 2019. arXiv: 1701.03077 [cs.CV]. URL: https://arxiv.org/abs/1701.03077.

[28] Younghyun Jo, Sejong Yang e Seon Joo Kim. *Investigating Loss Functions for Extreme Super-Resolution*. 2020.

Ringraziamenti

Con la consegna di questa tesi si conclude la mia triennale di Informatica. Questo percorso è stato al tempo stesso lungo e breve per tutte le persone conosciute e le esperienze vissute che per me sono state preziose e indimenticabili. Mi sembra dunque doveroso ringraziare di cuore tutti i colleghi di corso che in questi anni mi hanno aiutato a raggiungere questo importante traguardo, e che posso orgogliosamente chiamare amici. Allo stesso modo non posso non dire grazie ai miei amici conosciuti alle superiori che nonostante li veda poco, sembra sempre come se fosse passato un solo giorno dall'ultima volta. (E che non mi hanno ucciso per averli traditi scegliendo l'Informatica alla Chimica.) Voglio poi ringraziare infinitamente la mia famiglia senza la quale non sarei quello che sono oggi, che mi ha sempre supportato incondizionatamente, quali che fossero le mie scelte. Infine, sono grato anche a me stesso, per la scelta di essermi iscritto a questo corso, non essendovi stato un singolo giorno in cui io mi sia pentito della mia decisione.
Aspettando quanto il futuro ha in serbo, metto al capitolo un punto finale.