Politecnico di Torino

Politecnico di Torino

Laurea Magistrale in Ingegneria Matematica

A.a. 2024/2025

Sessione di laurea Ottobre 2025

Predizione del Rischio di Collasso dei Ponti Mediante Deep Learning e dati InSAR

Relatore: Dott. Francesco Della Santa

Candidato: Federico Sisci

Introduzione

Ponti, gallerie e molte altre infrastrutture rappresentano da sempre elementi essenziali per lo sviluppo tecnologico dell'uomo. Il monitoraggio di tali opere è stata ed è tutt'ora una delle sfide più rilevanti della società moderna con implicazioni in molti ambiti relativi sia alla sicurezza pubblica sia alla gestione del territorio. Infatti, il degrado di tali opere può avere importanti ripercussioni in termini economici e, soprattutto, può essere un rischio per la popolazione. Le tradizionali tecnologie per il monitoraggio sfruttavano principalmente misurazioni visive da parte di operatori specializzati e sensori installati in loco. Sfortunatamente, tranne in casi evidenti, questi metodi si sono mostrati inefficaci e costosi al fine di prevedere un incremento di rischio, per esempio legato a crolli parziali o totali delle strutture.
In seguito ha preso piede una nuova tecnologia basata sul confronto di immagini create mediante un segnale radar emesso da satelliti orbitanti attorno alla terra. La tecnologia interferometrica SAR, InSAR, utilizza due o più immagini per misurare con precisione millimetrica spostamenti di porzioni di strutture o della superficie al fine di progettare interventi mirati di manutenzione. Essendo i dati acquisiti tramite piattaforme satellitari, il monitoraggio può essere esteso a vaste aree e, inoltre, può essere ripetuto più volte nel tempo con un costo inferiore risultando poco invasivo.
Tuttavia, l'enorme quantità di dati a disposizione richiede metodologie di elaborazione maggiormente avanzate rispetto ad anni fa. Perciò, in questo contesto, si colloca il Deep Learning che rappresenta uno strumento moderno di estrazione di informazioni da ingenti quantità di dati.
L'obiettivo della tesi è di integrare i dati sui displacements ricavati dalla tecnologia InSAR con i moderni modelli di Deep Learning che utilizzano la *Multi-Heads Attention* al fine di ricavare relazioni complesse tra i dati di input. In particolare, si vuole dimostrare la loro grande utilità del prevedere l'incremento del rischio del crollo di una struttura al fine di poter agire con tempestività alla manutenzione.

L'elaborato si divide in cinque capitoli.
Nel primo capitolo si effettua una panoramica sui principali strumenti di monitoraggio di strutture, dalle ispezioni visive alle moderne tecniche InSAR. Si procede

spiegando diverse tipologie di queste ultime applicate a numerosi casi studio riguardanti i ponti, le gallerie e la superficie terrestre.
Nel secondo capitolo si esamina la storia del Deep Learning, dalla formalizzazione del Percettrone ai modelli di rete neurale profonda più moderni come i Transformers. In particolare, si approfondiscono le principali architetture di rete neurale come il *Fully Connected Layer*, *Multy-Heads Attention* ed il *Normalization Layer*. Infine analizzano le tecniche utilizzate durante l'addestramento dei modelli creati come la *Random Search*, il *Drop Out* ed i *Callbacks*.
Nel terzo capitolo si formalizza il problema trattato. Quindi, si approfondisce la definizione e la creazione delle immagini SAR e dell'interferometria SAR. In particolare, si spiegano le metodologie di acquisizione per i ponti di Tadcaster e Cantiano, che sono i due casi studio della tesi. Inoltre, si formalizzano i due modelli sviluppati: *GeoDispNet*, che considera solamente i displacements, e *HydroGeoDispNet*, che considera anche elementi naturali esterni come la portata dell'acqua. Infine, si definiscono le metriche utilizzate per valutarne l'accuratezza.
Nel quarto capitolo si approfondisce la creazione del dataset relativo ai due ponti in esame spiegando le tecniche di *data augmentation* e *data trasformation* utilizzate. Inoltre, si definiscono gli indici di incremento di rischio. Si procede definendo gli iperparametri e gli addestramenti effettuati. Infine, si mostrano i risultati ottenuti in differenti scenari valutando singolarmente i modelli e ponendoli a confronto con metriche comuni.
Nel quinto capitolo, si discutono le conclusioni, alla luce dei risultati ottenuti nel capitolo precedente. Inoltre, si effettua una panoramica di possibili sviluppi.

Il lavoro svolto in questa tesi ha contribuito alla produzione di un articolo da conferenza, a seguito della conferenza ARTISTE 2025, e di un articolo scientifico destinato ad una rivista. Entrambi i lavori saranno a breve sottoposti a peer review.

Indice

Capitolo 1

Tecniche di monitoraggio

Fin dall'antichità, l'uomo ha cercato di superare i limiti imposti dalla natura, costruendo opere che hanno segnato il cammino del progresso mediante l'uso degli strumenti a disposizione. Molte di tali strutture, come i ponti, le dighe o le gallerie non sono semplici costruzioni, ma rappresentano elementi essenziali per lo sviluppo tecnologico dell'uomo. Infatti, queste realizzazioni collegano territori e persone, diminuendo la distanza ed le tempistiche del percorso, o permettono di controllare elementi della natura per sfruttarli a proprio vantaggio. Il fine ultimo è di trasformare l'ambiente per soddisfare bisogni primari come la comunicazione, l'approvvigionamento energetico e la sicurezza.
Le infrastrutture più antiche, quindi progettate e costruite in altre epoche, così come quelle moderne, sono soggette all'azione di erosione degli elementi naturali di tutti giorni. In alcuni casi, esse sono sottoposte a forti stress strutturali legati a fenomeni naturali estremi, come i terremoti. Sfortunatamente, negli ultimi anni, si è registrato un aumento di eventi meteorologici estremi dovuti al cambiamento climatico [1]. In particolare, si tratta di piogge intense ed alluvioni che possono provocare fenomeni erosivi accelerati, provocando infiltrazioni che minano la resistenza della struttura. O ancora, raffiche di vento con intensità sopra la media che possono indebolire la stabilità delle opere più esposte.
Inizialmente, la manutenzione delle strutture si basava su ispezioni visive che rappresentano la forma più semplice di monitoraggio, ma possono essere influenzate dalla soggettività e dalla bravura di chi effettua le rilevazioni. Successivamente, si è passati ad un approccio definito *run-to-failure*, secondo il quale l'elemento danneggiato veniva sostituito. Quindi, tale strategia risulta priva di costi di monitoraggio, ma implica indirettamente rischi elevanti a livello di sicurezza. Un principio di prevenzione è stato introdotto con l'approccio *time-based maintenance*, con il quale vengono svolti interventi con tempistiche regolari e fissate a priori indipendentemente dallo stato reale dell'elemento da riparare. In tale scenario, vi è il rischio di effettuare azioni manutentive non necessarie, con il conseguente

spreco di risorse su componenti che non presentano effettive criticità a discapito di elementi, al contrario, che ne avrebbero bisogno. Infine, ha preso piede la tecnica del monitoraggio strutturale, SHM. Essa permette di agire sulla manutenzione basandosi su misure relative allo stato reale della struttura nel corso del tempo. In generale, tale processo sfrutta sensori installati e configurati per registrare grandezze fisiche sia legate alla struttura, come deformazioni, spostamenti e relative accelerazioni, sia degli agenti climatici, come temperatura e vento [2].
L'evoluzione tecnologica ha permesso un'ulteriore passo in avanti nel monitoraggio grazie all'utilizzo di satelliti orbitanti attorno alla terra. Infatti, questi ultimi consentono di osservare tutta la superficie terrestre sfruttando l'interazione tra le onde elettromagnetiche emesse dai satelliti e le strutture che si trovano ad una considerevole distanza [3]. Il processo è innovativo rispetto alle tecniche precedentemente citate, per alcune delle quali è richiesta l'installazione e la calibrazione di sensori in loco. Tra i vantaggi, vi è la possibilità di acquisire dati su vaste porzioni di superficie in maniera continua e regolare nel corso del tempo, indipendentemente dalle condizioni di luce solare o di meteo avverse, le quali rappresentano fattori che in precedenza potevano limitare tali rilevazioni. Ciò è reso possibile grazie a sensori radar che utilizzano microonde [3].
Un'ulteriore evoluzione di tale processo sono particolari tipologie di radar che sfruttano una tecnica di acquisizione ad apertura sintetica, chiamata SAR. Essa permette di ottenere immagini ad alta risoluzione grazie alla combinazione delle osservazioni fatte dallo stesso satellite in più posizioni lungo la propria traiettoria. Le immagini SAR sono create da informazioni sull'intensità e sulla fase del segnale riflesso [4]. A titolo esemplificativo, in Fig. 1.1 è possibile visualizzare un'immagine SAR del vulcano Etna in Sicilia.
Tra le tecniche di elaborazione di immagini SAR, l'interferometria SAR, in breve InSAR, ricopre un ruolo fondamentale nel monitoraggio strutturale. Essa permette di confrontare due immagini SAR che raffigurano la stessa area ma in due istanti temporali differenti. Da tale processo è possibile ottenere mappe e grafici di varia natura relative ai displacements con precisione millimetrica [4]. Tale strumento e la creazione delle relative immagini sono spiegate nel dettaglio nel Cap. 3.
La tecnologia InSAR ha subito diverse evoluzioni a seconda dell'utilizzo e della finalità degli interferogrammi. In particolare, la Differential InSAR, DInSAR, consiste nel rimuovere la componente topografica al fine di porre l'attenzione unicamente sulle deformazioni nel corso del tempo [1].
Inoltre, vi sono particolari tecniche che consentono di combinare più di due immagini per stimare i movimenti nel tempo: è il caso del Multi-Temporal InSAR. In particolare, una specifica implementazione è la SBASA InSAR, che combina solo coppie di immagini con piccola distanza temporale e spaziale. L'obiettivo è di minimizzare il più possibile la decorrelazione interferometrica al fine di ottenere interferogrammi di alta qualità. Per esempio, tale processo è stato utilizzato per

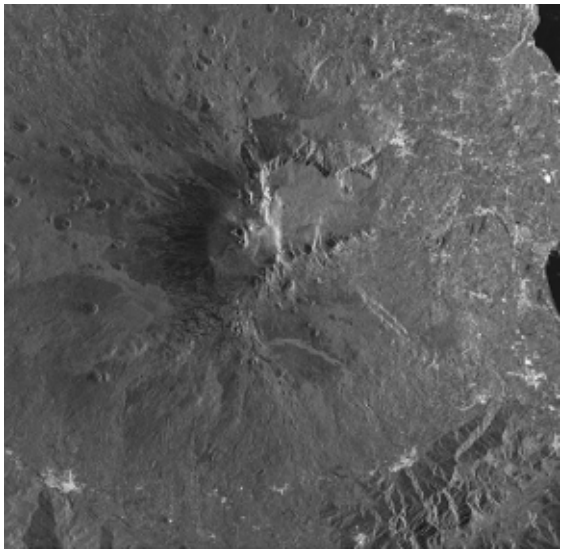

Figura 1.1: Immagine SAR del vulcano Etna [5].

elaborare immagini acquisite mediante la costellazione di satelliti COSMO-SkyMed al fine di osservare eventuali fenomeni di subsidenza: possibili e lenti abbassamenti della superficie terrestre. Ciò è stato applicato al tracciato della linea C della metropolitana di Roma calcolando le velocità medie lungo la "*line-of-sight*" (LOS, cioè la linea di vista), per ogni punto campionato. Nella Fig. 1.2 si osservano punti gialli e rossi vicino alle linee tratteggiate in nero, che rappresentano la porzione di linea della metro già edificata. Da tale analisi, si evince che vi sono stati fenomeni di subsidenza del territorio adiacente nel periodo di costruzione della linea C [1].
Un'ulteriore sviluppo della Multi-Temporal InSAR è la tecnica che sfrutta i Persistent Scatterrers (PS), ovvero pixel che mantengono una maggiore coerenza interferometrica nel tempo. Spesso, questi ultimi corrispondono a elementi riflettenti, come i piloni di un ponte, o strutture metalliche. La tecnica prende il nome di PSInSAR e utilizza centinaia di immagini SAR acquisite lungo la stessa orbita per identificare l'andamento della deformazione di ogni PS. Tale tecnologia applicata al monitoraggio di ponti risulta particolarmente promettente, in quanto consente di individuare eventuali cedimenti della struttura e spostamenti con risoluzione millimetrica. Inoltre, essendo i ponti costituiti da un'elevata quantità di componenti metallici e superfici lisce, è possibile avere una maggiore densità di PS che costituiscono un mappa dettagliata della struttura. Un esempio applicativo riguarda il ponte Palatino, situato a Roma, a cui è stato applicato un approccio PSInSAR con selezione automatica dei PS. Inoltre, si è eseguita una stima della velocità media di deformazione lungo la LOS che ha prodotto il risultato visibile in Fig. 1.3. Si noti la presenza di punti principalmente di colore giallo, sintomo di

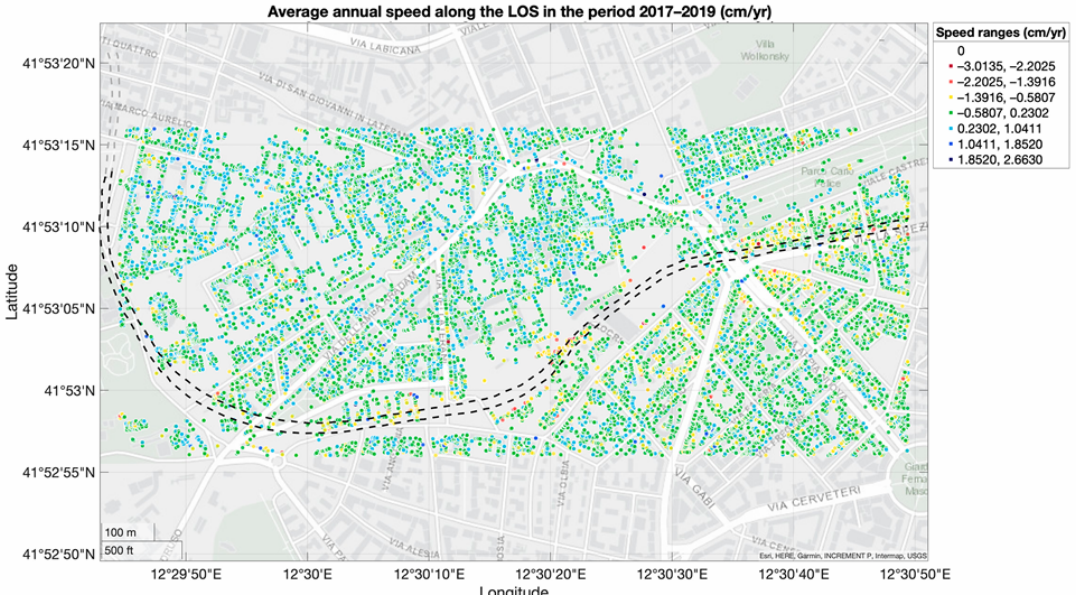

Figura 1.2: Dispersione della velocità media annuale lungo la LOS. Il percorso in nero si riferisce alla parte già in costruzione,in grigio si riferisce a parti non ancora costruite [1].

una stabilità generale della struttura. Inoltre, si osservi come i PS sono disposti principalmente sui bordi del ponte [5].
Un altro esempio di utilizzo di tecnologia InSAR per il monitoraggio di ponti viene presentata da Talledo et al. [6]. Gli autori sfruttano un dataste open source al fine di mappare le porzioni dei ponti in celle di 20 metri. Successivamente, vengono calcolate statistiche chiave, come la velocità di spostamento lungo la LOS, per ciascuno dei PS tramite dati MT-DInSAR. Inoltre, si combinano i dati acquisiti lungo le traiettorie ascendenti e discendenti per stimare le componenti verticali della velocità. Un esempio viene riportato in Fig. 1.4, nella quale sono visibili le celle che costituiscono i ponti e gli scatter di colore differente in base alla velocità media della LOS. Tale approccio viene applicato al tratto stradale tra Tirano e Marghera, che conta circa 300 ponti di tipologie differenti. Esso ha permesso di identificare un numero significativo di ponti che presentano velocità verticali di spostamento rilevanti, come in Fig. 1.4. Quindi, tale strumento risulta utile per propiziare ispezioni in loco e pianificare manutenzioni mirate [6].
Le tecniche InSAR non trovano applicazioni esclusivamente nel monitoraggio di strutture civili, come quelle precedentemente citate, ma sono uno strumento di fondamentale importanza anche per studiare e capire le cause di numerosi eventi ambientali. Per esempio, l'analisi delle frane avviene mediante la MT-InSAR. Tramite quest'ultima sono ricavate le serie temporali dell'andamento della frana

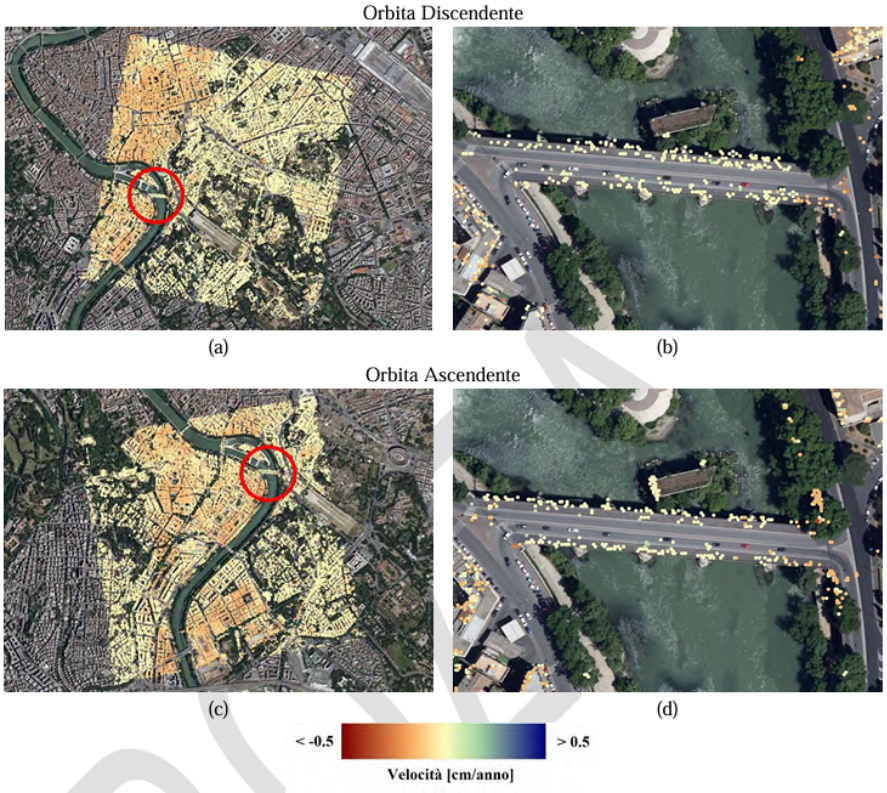

Figura 1.3: Disposizione dei PS nell'area del ponte Palatino e relativo zoom sul ponte. Le immagini si dividono in orbita ascendente e discendente del satellite [5].

e le zone cinematiche sul pendio. Successivamente, tali dati vengono combinati con quelli relativi alle piogge per un'analisi più approfondita. L'obiettivo non è solamente la misurazione dei displacements ma anche di riuscire ad interpretare i processi che governano le frane, al fine di progettare ed effettuare interventi mirati di prevenzione. In Fig. 1.5, è possibile osservare la mappa dei displacements di un versante instabile lungo la LOS in un periodo temporale di circa 3 mesi. L'immagine mostra nitidamente che alcune porzioni della frana hanno raggiunto spostamenti ingenti, dell'ordine di 350 millimetri. Essa è stata ottenuta mediante la tecnica Groud-Base InSAR, GB-InSAR, che permette di ottenere immagini in due dimensioni dei displacements stimando lo spostamento lungo la linea di vista [7].
In generale, le immagini SAR e le tecnologie sviluppate da esse sono state al centro di altri numerosi studi recenti [8], [9], [10], [11], [12], etc.. Si è osservato come i dati ricavati da tali strumenti costituiscono un'importante fonte di sapere al fine di progettare ed eseguire interventi mirati sulle strutture e sul territorio. Un ambito di crescente interesse, che sfrutta tali dati, è l'intelligenza artificiale (IA).

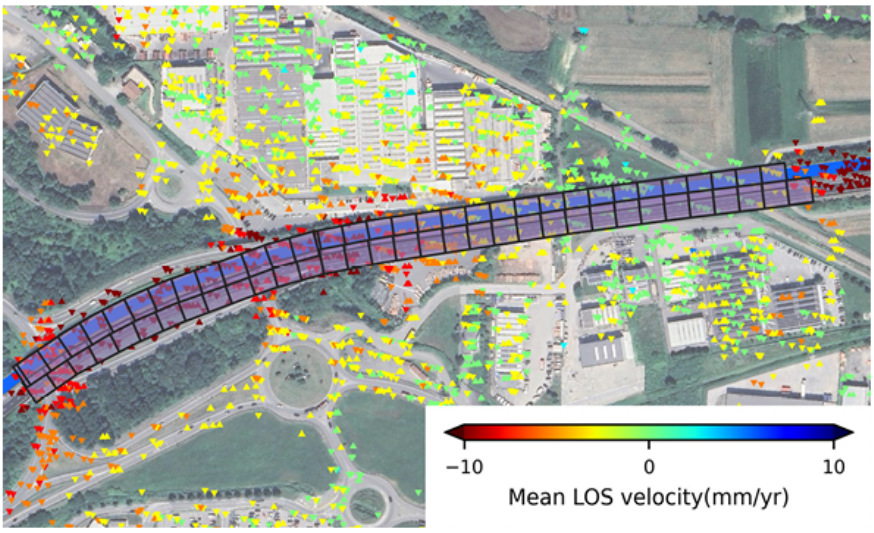

Figura 1.4: Vista di due ponti, uno per ogni direzione di traffico, segmentati in porzioni [6].

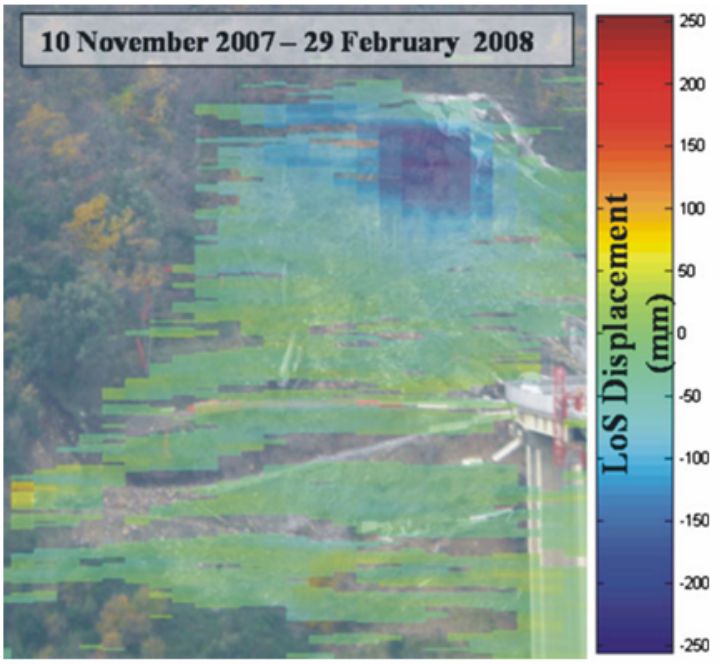

Figura 1.5: Mappa di spostamento cumulato lungo la LOS della zona franosa [7].

Attraverso algoritmi di Machine Learning [2] e Deep Learning, è possibile estrarre conoscenza anche in scenari in cui un pattern ben definito non è visibile né nelle

mappe di velocità della LOS o nei modelli di deformazione né nei grafici temporali dei displacements. Quindi, l'AI permette di individuare relazioni non lineari e complesse tra i dati consentendo di estrarre andamenti non immediatamente visibili, per esempio a causa del rumore negli strumenti di acquisizione. Ciò risulta di grande utilità nell'individuare il comportamento futuro passando da un'analisi descrittiva del fenomeno ad una predittiva. Inoltre, l'AI permette di gestire e analizzare datasets di grandi dimensioni consentendo di applicare queste analisi su vasta scala e di supportare le decisioni in scenari complessi.

Capitolo 2

Storia del Deep Learning e strumenti utilizzati

In questo capitolo si effettua una panoramica del Deep Learning attraverso una prospettiva storica, che illustri l'evoluzione di tale disciplina dalle sue origini fino alle più recenti scoperte scientifiche. Per esempio, si analizza il modello dei primi neuroni artificiali ispirati al cervello umano. O ancora, si indaga il meccanismo di backpropagation che ha segnato il passaggio dal semplice neurone alle reti neurali. Infine, si esaminano alcune scoperte recenti come il meccanismo di attenzione che è alla base dei Trasformers.
Successivamente, si analizzano i principali layers (letteralmente, strati di neuroni) utilizzati all'interno della rete neurale e le più rilevanti strategie applicate all'addestramento dei modelli.

2.1 Il Deep Learning

Il concetto di *Deep Learning* (DL), o Apprendimento Profondo, è fortemente interconnesso a quelli di *Artificial Intelligence* (AI), o Intelligenza Artificiale, e di *Machine Learning* (ML), o Apprendimento Automatico. La loro storia ripercorre tutto il secolo precedente e vede uno sviluppo sempre maggiore nei giorni nostri grazie alle molteplici innovazioni nel campo dell'informatica e nella creazione di nuove componenti per computer, come le GPU, sempre più performanti e adatte agli algoritmi di queste discipline.
Al fine di comprendere il legame tra AI, ML e DL, risulta utile considerare il diagramma proposto in Fig. 2.1. L'intelligenza artificiale costituisce l'involucro più esterno ed ampio tra le tre discipline. Essa riproduce la capacità di integrare l'intelligenza umana all'interno delle macchine mediante l'uso di regole e algoritmi codificati da esseri umani al fine si risolvere compiti specifici. L'AI permette ai

computer di imitare il cervello umano nelle sue capacità di pensiero. Il Machine Learning può essere considerato un sottoinsieme dell'intelligenza artificiale. Esattamente come farebbe un essere umano, il ML sfrutta dataset ed algoritmi per insegnare al computer in che modo imparare da esempi. Gli algoritmi vengono addestrati su dati strutturati ed etichettati con lo scopo di effettuare classificazioni e previsioni, al fine di estrarre informazioni chiave utili a guidare processi decisionali. Nel caso si abbia a che fare con dati non strutturati, solitamente vengono sottoposti ad un processo di pre-elaborazione per trasformarli in un formato di tipo strutturato. In questo contesto, quindi, l'apprendimento autonomo dipende maggiormente dall'intervento umano durante la fase di apprendimento.
Infine, il Deep Learning rappresenta il sottoinsieme più specifico in Fig. 2.1. Il suo punto di forza è la capacità di automatizzare quasi completamente il processo di estrazione delle caratteristiche al fine di ridurre notevolmente la dipendenza dall'azione dell'uomo che, invece, caratterizzava il ML. L'algoritmo alla base del DL è costituito dalle reti neurali profonde che permettono al computer di apprendere autonomamente una struttura nei dati di input. Tale caratteristica rende il DL significativamente efficacie con grandi quantità di dati non strutturati senza la necessità di una pre-elaborazione umana [13].

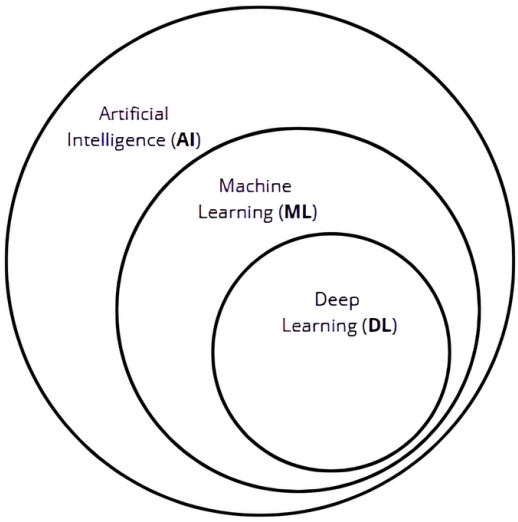

Figura 2.1: Relazione tra AI, ML e DL.

Storicamente, il punto di partenza del DL si identifica con il 1943: Warren S. McCulloch e Walter Pitts, rispettivamente un neurofisico e un matematico statunitensi, gettano le basi per lo sviluppo di tale disciplina attraverso la comprensione computazionale del sistema nervoso. Nel loro articolo [14], gli autori partono dalla natura *all-or-none* dell'attività nervosa, spiegando come il singolo neurone possa

essere rappresentato mediante una proposizione logica: la sua attivazione deriva da una funzione booleana degli input prevenienti dagli altri neuroni.
Successivamente, nel 1949, Donald Olding Hebb, padre della neuropsicologia, formula una teoria sul rafforzamento delle connessioni sinaptiche, che sarà il fondamento di studi seguenti. Nel suo elaborato [15], l'autore sintetizza quella che oggi viene chiamata Legge di Hebb con la frase "*Cells that fire together, wire together*", letteralmente "Cellule che si attivano insieme, si collegano insieme". Hebb esprime quindi il concetto secondo il quale un'attivazione simultanea e ripetuta di alcuni neuroni rafforza la loro connessione; di conseguenza, l'accensione di uno rende più probabile l'accensione dell'altro in futuro.
Successiamente, nel 1958, lo psicologo e informatico Frank Rosemblatt scrive un articolo [16] che rappresenta il cardine della storia dell'AI. Egli getta le basi delle reti neurali ispirandosi al sistema nervoso umano, attraverso l'introduzione del *Perceptron*, ovvero il percettrone. L'architettura di quest'ultimo si compone di tre elementi principali, visibili nella Fig.2.2, le cui connessioni seguono una logica *feed-forward*: non sono presenti collegamenti che formino cicli. In particolare, *S-points* è il primo stadio costituito da un insieme di unità sensoriali, le quali ricevono uno stimolo esterno, attivandosi secondo la logica *all-or-none* citata precedentemente. Successivamente, gli impulsi vengono trasmessi ad uno strato di *A-units*, unità di associazione, tramite connessioni sia eccitatorie sia inibitorie. Un'unità si accende se la somma degli impulsi in input supera una determinata soglia θ.
Infine, un sottoinsieme casuale di *A-units* forma un *source-set*, il quale si collega alle *R-units* che gestiscono l'output del perceptron.

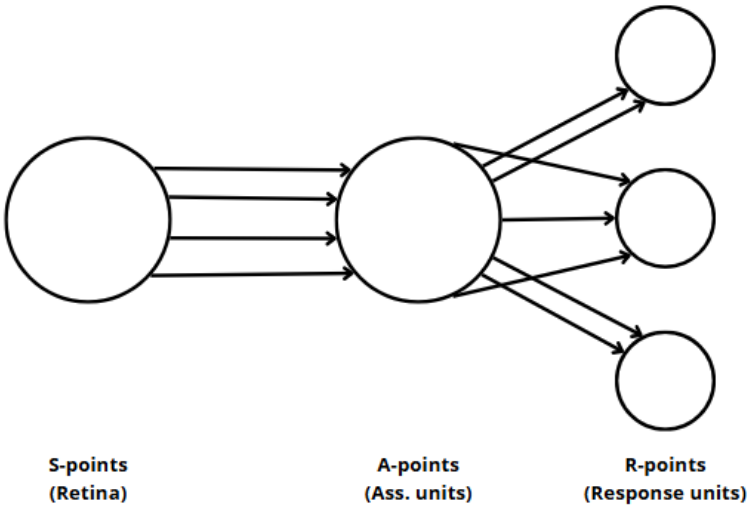

Figura 2.2: Rappresentazione perceptron di Rosemblatt.

La novità di Rusemblatt consiste nel meccanismo con cui il perceptron apprende

e si basa sulla legge di Hebb precedentemente citata. In particolare, ad ogni *A-unit* viene associato un valore V dinamico che cambia durante l'addestramento e che aumenta se l'unità viene attivata. L'incremento di questo valore potenzia le connessioni delle unità coinvolte nella risposta desiderata e crea un percorso privilegiato all'interno della rete. In questo modo, tale flusso verrà maggiormente considerato nelle iterazioni future rendendo più probabile l'output corretto quando si presenterà lo stesso input.
Nonostante il forte entusiasmo derivante dai progressi fin qui citati, che scaldarono la comunità scientifica, nel 1969 gli informatici Marvin Minsky e Seymour Papert misero in luce alcuni limiti del perceptron. All'interno del loro libro [17], essi si concentrano sulla funzione logica *XOR* tipicamente rappresentata dal simbolo $\oplus$. Quest'ultima, equivalente all'addizione in modulo 2, prende in ingresso due inputs α, β e restituisce in uscita un solo output; questi ultimi possono avere valori *TRUE* e *FALSE*, rispettivamente 1 e 0 in binario. L'output viene rappresentato dalla tabella di verità 2.1. Gli autori mostrarono come il perceptron, essendo un classificatore lineare, non fosse in grado di risolvere tale funzione che necessiterebbe invece un separatore non lineare.

Input α	**Input β**	**Output**
0	0	0
0	1	1
1	0	1
1	1	0

Tabella 2.1: Outputs della funzione logica *XOR*.

Tali limiti determinarono una diminuzione dell'interesse della comunità scientifica riguardo le reti neurali. Minsky e Papert, però, mostrano che reti artificiali più complesse, come i *perceptron multilivello*, ovvero percettroni aventi più unità centrali, riuscissero a superare tali ostacoli almeno in linea teorica. La difficoltà più grande consisteva nel fatto che gli algoritmi di addestramento conosciuti fino ad allora non erano applicabili a tali strutture.
Un svolta avvenne nel 1986, quando David E. Rumelhart, psicologo, Geoffrey E. Hinton, informatico, e Ronald J. Williams, docente di informatica, pubblicarono l'articolo [18] nel quale descrivevano una nuova procedura di apprendimento. Quest'ultima si diffuse rapidamente riaccendendo la corsa della ricerca in questo campo e segna definitivamente una svolta nello sviluppo delle reti neurali profonde che sono alla base del Deep Learning. L'algoritmo è noto come *backpropagation* e consiste nello sfruttare le regole di derivazione della funzione composta per calcolare il gradiente dei pesi w della rete rispetto una funzione di errore (o di Loss) da minimizzare per addestrare con successo il modello. Più nel dettaglio, come

mostrato in Fig. 2.3, la rete neurale considerata è feed-forward, composta da un livello di input, uno o più livelli nascosti, chiamati *hidden layers*, e un livello di output.
In generale, ogni nodo j riceve un input totale u_j definito come una somma pesata:

$$u_j = \sum_i w_{ij}\bar{y}_i \tag{2.1}$$

dove w_{ij} è il peso della connessione tra i nodi i e j mentre $\bar{y}_i$ è l'output dei nodi precedenti dato da una funzione non lineare applicata a u_j. L'errore citato precedentemente viene definito, per ogni coppia u$= (u_j, \bar{y}_j)$ come:

$$E = \sum_{\mathrm{u}} \sum_j \frac{1}{2}(\bar{y}_{j,\mathrm{u}} - y_{j,\mathrm{u}})^2 \tag{2.2}$$

dove y_j^u è l'output desiderato. L'obiettivo è minimizzare tale errore attraverso la discesa del gradiente, ovvero calcolare la derivata parziale di E rispetto ai pesi w_{ij}. Per calcolare ciò è utile considerare la derivata parziale dell'errore, per ognuna delle unità di output, sia rispetto all'output $\bar{y}_j$

$$\frac{\partial E}{\partial \bar{y}_j} = \bar{y}_j - y_j \tag{2.3}$$

sfruttando l'equazione 2.2, sia rispetto all'input u_j

$$\frac{\partial E}{\partial u_j} = \frac{\partial E}{\partial \bar{y}_j} \cdot \frac{d\bar{y}_j}{du_j} = (\bar{y}_j - y_j) \cdot f'(u_j) \tag{2.4}$$

considerando l'equazione 2.3 e la definizione di $\bar{y}_j$. Si ottiene quindi

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{ij}} = \frac{\partial E}{\partial u_j} \cdot \bar{y}_i$$

utilizzando l'equazione 2.4 e la definizione 2.1. Infine si procede aggiornando i pesi:

$$\Delta w_{ij} = -\xi \cdot \frac{\partial E}{\partial w_{ij}}$$

dove ξ è il tasso di apprendimento (in inglese, learning rate).
La procedura presentata da Rumelhart et al. fonda le basi su studi precedenti e consente l'addestramento efficace di reti neurali che presentano livelli nascosti. Matematicamente, il metodo di calcolo di $\frac{\partial E}{\partial w_{i,j}}$ è una particolare applicazione ricorsiva della *regola della catena* applicata ad una funzione composta, la funzione d'errore composta con la rete neurale, nella quale ogni layer rappresenta una funzione non lineare.

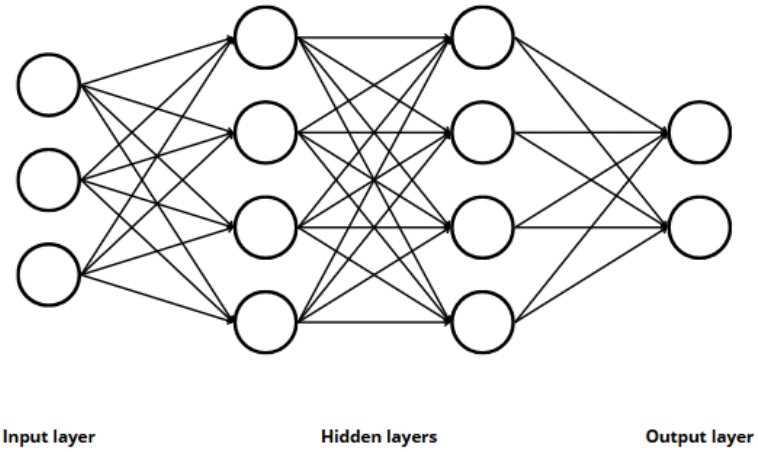

Figura 2.3: Rete neurale multistrato con due livelli nascosti.

Successivamente, la ricerca scientifica si concentrò sull'aumentare il numero di nodi dei singoli layers e sull'incrementare la quantità di hidden layers, rendendo le reti neurali sempre più *profonde*, dando il via al Deep Learning. Ciò è stato reso possibile anche grazie ad un maggior supporto tecnologico accompagnato da uno sviluppo costante delle componenti software e hardware dei computer nel corso del tempo. Una figura di riferimento in questo contesto è stato Geoffrey Hinton, che aveva precedentemente collaborato allo sviluppo dell'algoritmo di back-propagation. Tramite alcuni lavori, egli ha mostrato come l'applicazione di questo nuovo strumento in diversi contesti possa superare le prestazioni di altri algoritmi considerati come lo stato dell'arte. Di particolare rilevanza è l'articolo [19] pubblicato nel 2009 da Hinton in collaborazione con Abdel-rahman Mohamed e George Dahl, nel quale si mostra che le *Deep Belief Networks* sono di grande utilità nella modellazione acustica per il riconoscimento fonetico. Tale procedura, sfrutta la profondità dell'architettura, al fine di creare dipendenze ad alto ordine nei segnali vocali superando i modelli utilizzati fino ad ora in tale contesto.
Pochi anni più tardi, Hinton insieme a Alex Krizhevsky e Ilya Sutskever pubblica un articolo [20] nel quale viene presentata un particolare tipo di rete neurale chiamata *Convolutional Neural Network*, CNN, che vince la ImageNet Large-Scale Visual Recognition Challenge, competizione basata sul riconoscimento di immagini. La vittoria è dovuta all'efficace combinazione dell'architettura profonda con un dataset ampio e all'introduzione di alcune tecniche, che diventeranno pratiche standard negli anni a seguire. Un'ulteriore innovazione che ha permesso di ridurre notevolmente l'errore ed i tempi di addestramento è quella di elaborare il training su due *GPU* in parallelo. L'architettura è progettata in modo tale che metà dei

neuroni di ogni layer risieda su una sola GPU, mentre la comunicazione tra queste ultime avvenga solo in specifici layer.
Un altro ambito in cui il Deep Learning si è sviluppato è quello riguardante la trattazione di sequenze come frasi, testi o sequenze temporali. Fino al 2010, i principali modelli utilizzati sono state delle reti neurali profonde con caratteristiche differenti come le *Reti Neurali Ricorrenti*, RNN ed i *Long Short Term Memory*, LSTM. Le prime sono caratterizzate da un meccanismo di ricorrenza mediante il quale mantengono una memoria interna degli stati precedenti [21]. Le LSTM, invece, sono state progettate appositamente per risolvere il problema della scomparsa del gradiente nelle sequenze lunghe in input [22]. Tali algoritmi hanno rappresentato per anni lo stato dell'arte in questo contesto e sono alla base dei modelli introdotti in un articolo [23] da Ilya Sutskever, Oriol Vinyals e Quoc V. Le di Google e prendono il nome di *Sequente-to-Sequence*, Seq2Seq. Essi vengono impiegati con successo in compiti di traduzione automatica e alla loro base vi sono due componenti chiave: un *encoder* che comprime l'input in un vettore di dimensione fissata e un *decoder* che usa tale vettore per generare l'output sequenziale. Nonostante i buoni risultati ottenuti, questa struttura possiede una limitazione: poiché l'intero input deve essere compresso in un unico vettore, i SeqtoSeq presentano un collo di bottiglia informativo in particolar modo per sequenze lunghe.
Tale limite viene superato per la prima volta nel 2015 grazie agli studi di Dzmitry Bahdanau, KyungHyun Cho e Yoshua Bengio, che presentano un primo meccanismo di attenzione. Nel loro articolo [24], essi mostrano come tramite il meccanismo di attenzione venga calcolata una ponderazione dinamica su tutti i layer nascosti dell'encoder che viene passata al decoder insieme all'output dell'encoder. In altre parole, è come se il modello fornisse maggiore importanza ad alcune parti della sequenza in input. Tale meccanismo viene inizialmente applicato ai modelli SeqtoSeq, ma nel 2017 Vaswani et al. di Google propongono una nuova struttura all'interno del loro articolo "*Attention is All You Need*" [25]. Tale processo si basa unicamente su un'estensione del meccanismo di attenzione che prende il nome di *Self-Attention*: esso permette ad ogni pezzo della sequenza di interagire direttamente con tutti i restanti, calcolando dei pesi di attenzione per ogni coppia. Ciò consente di catturare dipendenze globali precedentemente difficili da individuare e segna un incredibile passo in avanti in questo contesto.

2.2 Architetture delle reti neurali

Al fine di comprendere al meglio il codice sviluppato, si analizzano nel dettaglio i principali layers utilizzati. In particolare, si approfondisce il *Fully Connected Layer*, che è uno strato in cui ogni neurone è connesso a tutti i neuroni dello strato precedente. Il suo utilizzo permette di combinare globalmente le informazioni

mediante l'applicazione di una trasformazione lineare e di una funzione di attivazione (solitamente, non lineare). Successivamente, si indaga il *Multi-Head Attention Layer*, che rappresenta il cuore dell'encoder e del deconder presenti nell'architettura dei Transformers, visibile in Fig. 2.4. Esso permette al modello di calcolare in parallelo più valori di attenzione al fine di individuare relazioni tra porzioni diverse dei dati di input, anche molto distanti tra loro. Considerando il caso esaminato nella tesi, l'obiettivo è di catturare dipendenze tra gli spostamenti verticali dei punti, le coordinate e, per uno dei due modelli, la portata del fiume. Infine, si esamina il *Normalization Layer* avente lo scopo di stabilizzare l'addestramento: infatti, esso permette di ridurre fenomeni come l'esplosione o la scomparsa del gradiente durante il processo di ottimizzazione dei pesi del modello.

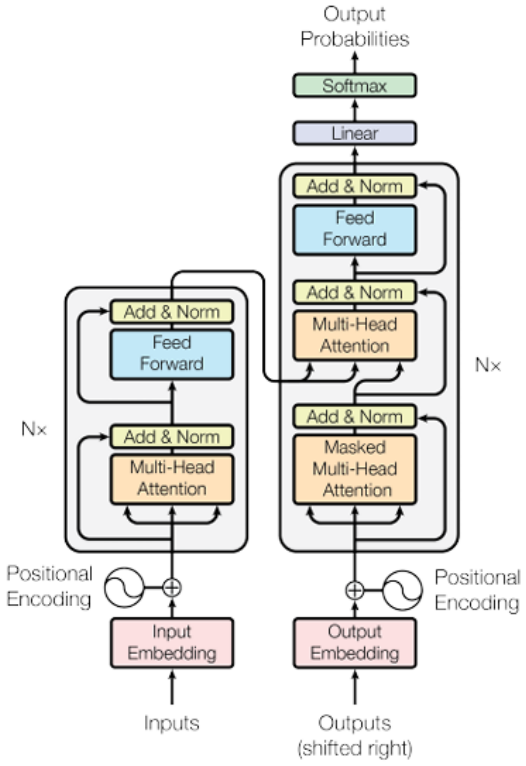

Figura 2.4: Architettura dei Transformer dal paper "*Attention is All You Need*" [25].

2.2.1 Fully-Connected Layer

Il *Fully-Connected*, o *Dense*, *Layer*, in breve FC-Layer, è la forma più semplice e, storicamente, la prima architettura usata in una rete neurale artificiale. Esso

ha segnato l'inizio dello sviluppo delle reti neurali e costituisce un elemento di fondamentale importanza nei modelli sviluppati ancora oggi. La sua struttura, semplice quanto all'epoca rivoluzionaria, permette di combinare linearmente le informazioni e di introdurre non linearità risultando una componente cruciale per la combinazione di informazioni imparate dal modello. Come è possibile notare in Fig 2.5, la caratteristica principale è che ogni neurone di un singolo layer è connesso ad ogni neurone del layer precedente. Tali elementi risultano utili per le

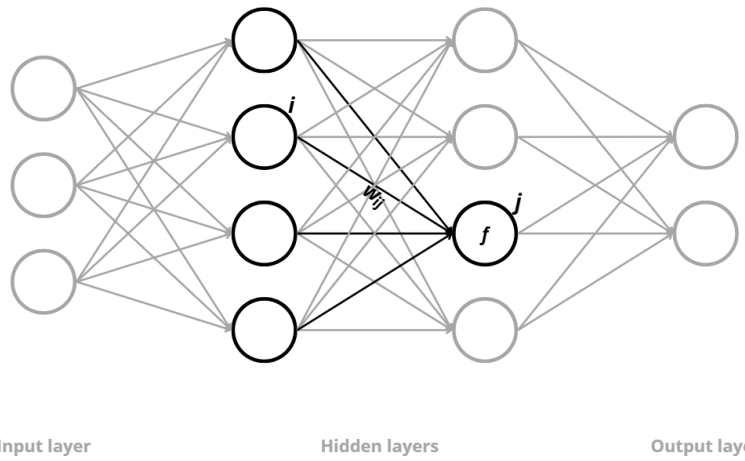

Figura 2.5: Esempio di fully-connected layer in una rete neurale.

reti neurali profonde e, a differenza dei primi strati della rete in cui vengono estratte informazioni locali, agiscono su tutte le features contemporaneamente estraendo informazioni globali [26].
Dalla documentazione di TensorFlow [27] è possibile estrarre la seguente definizione di un FC-layer:

$$\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$$

ovvero il dense layer calcola l'output moltiplicando l'input per una matrice di pesi chiamata kernel e, al risultato, somma un termine bias. Infine applica una funzione di attivazione, che svolge un ruolo fondamentale all'interno del modello. Se quest'ultimo fosse composto solamente da una sequenza di layer sprovvisti della funzione di attivazione, si potrebbe considerare, indipendentemente dalla profondità della rete, come un'unica grande trasformazione lineare. Tale struttura rappresenta un forte limite, in quanto i dati di input di una rete neurale provenienti dal mondo reale sono principalmente non-lineari. Introducendo la funzione di attivazione per ogni nodo, viene immessa non-linearità nella rete, che permette al modello di apprendere relazioni complesse ed eventuali gerarchie nei dati forniti [28].

Matematicamente, il fully-connected layer si esprime come una funzione $\mathcal{L}(x) : \mathbb{R}^i \to \mathbb{R}^o$ tale che [29]:

$$\mathcal{L}(x) = \sigma(W^T u + b); \qquad \forall u \in \mathbb{R}^i$$

dove $u \in \mathbb{R}^i$ è un vettore di input, $W \in \mathbb{R}^{i \times o}$ è una matrice di pesi tale che l'elemento w_{ij} rappresenta il peso della connessione tra i nodi i e j. Inoltre, $\hat{b} = (\hat{b}_1, ..., \hat{b}_o) \in \mathbb{R}^o$ è un vettore di bias e $\sigma : \mathbb{R}^o \to \mathbb{R}^o$ è una *element-wise function*: funzione che viene applicata elemento per elemento ad ogni vettore che riceve in input e rappresenta la funzione di attivazione.

2.2.2 Multi-Head Attention

L'idea alla base di ogni meccanismo di attenzione è quella di modellare relazioni implicite tra due o più parti di una sequenza data in input, chiamate *token*. Dopo il 2015, vi sono stati diversi tentativi di migliorare quanto fatto da Bahdanau descritto nella sezione precedente. Un primo step viene effettuato da Cheng et al. e prende il nome di *intra-attention*. Quest'ultima consente al modello di creare un collegamento tra il token in analisi e le sue rappresentazioni precedentemente create, definendo un valore di attenzione e salvandolo su una memoria interna chiamata *memory tape*. In questo modo, il modello sarà in grado di determinare relazioni tra token all'interno della stessa sequenza [30].
Un'ulteriore modifica viene apportata da Luong. Quest'ultimo presenta diversi metodi per semplificare e, soprattutto, migliorare il meccanismo di attenzione, ma il contributo più importante riguarda principalmente l'utilizzo della moltiplicazione al posto dell'addizione all'interno dell'algoritmo. Da qui nasce la *dot-product attention*, un particolare tipo di attenzione che utilizza il prodotto scalare per calcolare le similarità tra i token [31].
Successivamente, Vaswani et al. sviluppano ulteriormente l'attention: introducono una moltiplicazione per uno *scaling factor* definendo l'attenzione come *Scaled Dot-Product Attention* [25]. Nel dettaglio, un tensore di input generico $X \in \mathbb{R}^{\mathrm{n} \times \mathrm{K}}$, dove n e K sono rispettivamente il numero di token dell'input e la dimensione delle features associate a ciascun token, è proiettato in tre spazi distinti. Questi ultimi sono queries Q, keys K e values V:

$$Q = XW_Q \oplus b_Q^T, \qquad K = XW_K \oplus b_K^T, \qquad V = XW_V \oplus b_V^T, \qquad Q, K, V \in \mathbb{R}^{\mathbf{n} \times \mathrm{F}}$$

dove $W_Q, W_K, W_V \in \mathbb{R}^{\mathrm{K} \times \mathrm{F}}$ sono le matrici di proiezione, $b_Q, b_K, b_V \in \mathbb{R}^{\mathrm{F}}$ sono i vettori bias e F è la dimensione dello spazio in cui ogni elemento viene proiettato dopo tale operazione. Tali concetti vengono formalizzati per la prima volta dagli autori in questo paper. In particolare, la query rappresenta il token per il quale si sta calcolando il peso dell'attention e, nel contesto della self-attention, la posizione

di ogni token è una query. Le keys sono le rappresentazioni di tutti gli elementi della sequenza che potenzialmente possono essere interrogati dalla query e sono utili al fine di calcolare la similarità con quest'ultima mediante il prodotto scalare. Infine, i values contengono l'informazione concreta che viene aggregata all'output: una volta calcolati i pesi dell'attention attraverso la somiglianza query-keys, essi vengono usati nel conteggio di una media pesata che produrrà l'output. In seguito, viene

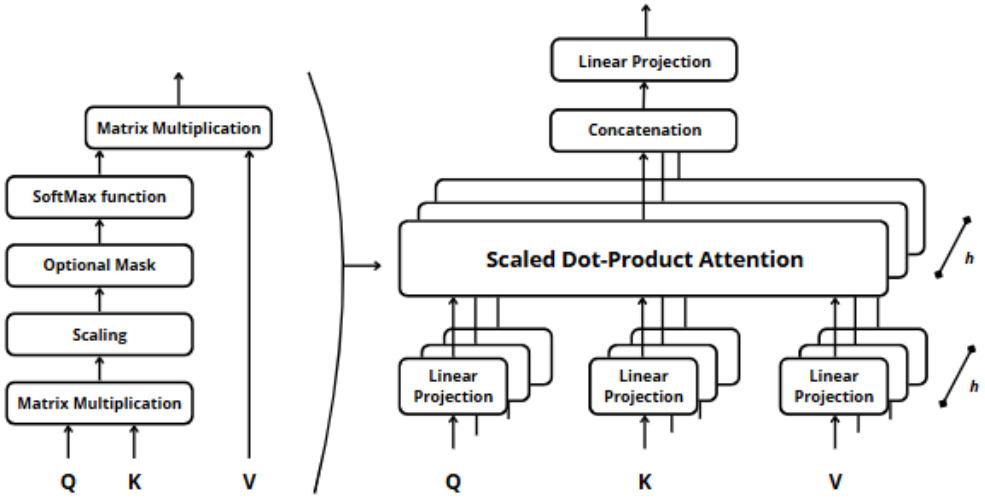

Figura 2.6: Rappresentazione grafica della Scaled Dot-Product Attention, a sinistra, e della Multi-Head Attention, a destra.

calcolato il prodotto tra Q e K^T attraverso la moltiplicazione matriciale, blocco Matrix Multiplication in Fig 2.6, al fine di calcolare le similarità tra ogni query e tutte le chiavi. In seguito, il risultato viene scalato per il fattore $\frac{1}{\sqrt{\mathrm{F}}}$, blocco Scaling. Inoltre, si applica una maschera, blocco Mask, al fine di preservare la proprietà auto-regressiva e la funzione *Soft-Max*, blocco SoftMax Function. Quest'ultima fa parte della famiglia delle funzioni di attivazione e trasforma un vettore di valori in un vettore con elementi non-negativi a somma uno. In generale, dato un vettore di input $u = (u_1, u_2, ..., u_K) \in \mathbb{R}^K$, la funzione soft-max è definita come segue [32]

$$\mathrm{softmax}(u_i) = \frac{e^{u_i}}{\sum_{j=1}^{K} e^{u_j}}.$$

In sintesi, ipotizzando di applicare il meccanismo di attenzione contemporaneamente a Q, K e V, si ottiene la seguente formula:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{\mathrm{F}}}\right)V$$

L'uso del prodotto matriciale rappresenta una svolta in questo contento, soprattutto considerando la maggior velocità di calcolo fortemente ottimizzato.
Vaswani et al. sviluppano ulteriormente tale meccanismo definendo la *Multi-Head Attention*, MHA. Come è possibile vedere in Fig 2.6, essa consiste nel proiettare linearmente l'input Q, K e V un numero h di volte. Successivamente, ognuna di queste proiezioni viene passata come input a h blocchi di scaled dot-product attention che lavorano in parallelo. In seguito, gli output vengono concatenati e proiettati un'ultima volta. In sintesi, si ottiene:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{H}_1, ..., \text{H}_h)W^0$$

dove

$$\text{H}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

con $W_i^Q \in \mathbb{R}^{D\times \text{F}}$, $W_i^K \in \mathbb{R}^{D\times \text{F}}$ e $W_i^V \in \mathbb{R}^{D\times \text{F}}$, e $W^0 \in \mathbb{R}^{h\cdot \text{F}\times D}$ con $D = \text{F} \cdot h$.
Il MHA permette quindi al modello di prestare attenzione a informazioni provenienti da rappresentazioni dei token di sottospazzi differenti in parallelo. Ciò non era possibile con un singolo meccanismo di attenzione.

2.2.3 Normalization Layer

All'interno del codice prodotto viene usato il *Normalization layer* mediante i comando di TensorFlow [27]. Tale tecnica è impiegata in una rete neurale al fine di normalizzare le attivazioni di strati differenti. Come spiegano Ba et al. all'interno del loro articolo [33], tale strumento supera alcune limitazioni della *Batch Normalization*. Quest'ultima è stata ideata con lo scopo di accelerare e stabilizzare la fase di addestramento delle reti profonde. Matematicamente, essa normalizza la somma degli input di ogni neurone u_i di un singolo layer l sfruttando media e varianza calcolate sull'intero mini-batch i:

$$\overline{u}_{i,l} = \frac{u_{i,l} - \mu_i}{\sigma_i}$$

dove la media sul batch è determinata da

$$\mu_i = \mathbb{E}_{\text{u}\sim P(\text{u})}[u_{i,l}] = \int u_i(\text{u})P(\text{u})d\text{u}$$

con u esempio di input del mini-batch e $P(\text{u})$ distribuzione di probabilità dei dati di input. Inoltre

$$\sigma_i = \sqrt{\mathbb{E}_{\text{u}\sim P(\text{u})}[(u_{i,l} - \mu_i)^2]}$$

definisce la deviazione standard calcolata sul batch. Infine l'output del layer l è definito come

$$\bar{y}_i = \kappa_i \overline{u}_i + b_i$$

dove κ_i e b_i rappresentano rispettivamente i fattori di *gain* e *bias* appresi dal modello durante la fase di addestramento. In conclusione, la batch normalization richiede un batch sufficientemente grande al fine di stimare correttamente media e varianza; inoltre, viene applicata al singolo neurone e su tutti gli esempi del batch introducendo, di conseguenza, una dipendenza tra questi ultimi.
Al contrario, il Normalization layer non necessita di una dimensione minima del batch e calcola media e varianza su tutte le unità di un layer per singolo input a_i:

$$\hat{u}_i = \frac{u_i - \mu}{\sigma}$$

dove la media è

$$\mu = \frac{1}{L}\sum_{i=1}^{L} u_i$$

con L numero di neuroni del singolo strato. La deviazione standard è calcolata come

$$\sigma = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(u_i - \mu)^2}.$$

Infine l'output del layer è definito da

$$\bar{y}_i = \iota_1 \hat{u}_i + \iota_2$$

con ι_1 e ι_2 fattori appresi durante l'addestramento.
Di conseguenza, tale trasformazione normalizza per singolo layer e per singolo esempio evitando dipendenza tra questi ultimi.

2.3 Strategie per l'addestramento

Si esaminano del dettagli alcuni elementi utili a efficentare l'addestramento dei modelli. In particolare, si analizza la tecnica della *Random Search* utilizzata con l'obiettivo di individuare gli iperparametri che permetto al modello di offrire le migliori prestazioni. In seguito, si approfondisce il *Drop out*, usato per disattivare una piccola percentuale di neuroni durante la fase di addestramento al fine di ridurre il rischio di overfitting. Infine, si indagano i principali *Callbacks* implementati nella fase di training del modello. Essi sono sfruttati per ottimizzare l'addestramento mediante la riduzione del learning rate o l'interruzione preventiva del processo nel caso non vi fosse alcun miglioramento di una specifica metrica.

2.3.1 Iperparametri e Random Search

Una distinzione rilevante nell'ambito del deep learning è quella tra *parametri* e *iperparametri* del modello. I primi si riferiscono ai valori interni di quest'ultimo

e sono, per esempio, i pesi ed i bias della rete neurale appresi durante la fase di addestramento attraverso l'ottimizzazione della loss. Invece, gli iperparametri sono esterni al modello e ne descrivono la configurazione. Essi devono essere scelti manualmente prima dell'avvio della fase di training e possono essere, per esempio, il learning rate, la dimensione del batch o il numero di layer della rete [13].
La scelta di tali iperparametri rappresenta un passaggio cruciale nello sviluppo di un modello in quanto essa può influenzare notevolmente le prestazioni o il tempo di esecuzione. La tecnica utilizzata nel codice sviluppato è la *Random Search* [34], che permette di selezionare alcune combinazioni di valori degli iperparametri in modo casuale da uno spazio definito a priori. Tale tecnica è in contrapposizione con un altro metodo chiamato *Grid Search*, che consiste nel testare tutte le possibili combinazioni degli iperparametri a disposizione. Formalmente, lo spazio degli iperparametri è definito come

$$\Lambda = \Lambda_1 \times \cdots \times \Lambda_i \times \cdots \times \Lambda_s \qquad i = 1, ..., s$$

dove Λ_i è un insieme di valori numerici e non che può assumere un determinato iperparametro e sr è il numero totale di quest'ultimi. La Grid Search valuta ogni possibile combinazione di valori

$$\lambda_t = (\lambda_{1,t}, \lambda_{2,t}, ..., \lambda_{s,t}) \qquad t = 1, ..., T$$

con T numero delle combinazioni possibili. In contrapposizione, la Random Search estrae un numero n prefissato di combinazioni $\lambda_t \sim \mathcal{U}(\Lambda)$ in modo indipendente e casuale da una distribuzione uniforme.
L'intuizione alla base di tale tecnica è che non tutti i parametri influenzano in ugual modo il modello. Pertanto, esplorare casualmente Λ consente di ottenere in generale prestazioni migliori. Infatti, come dimostrano Bergstra e Bengio [35], a parità di numero di tentativi, la Random Search individua soluzioni migliori rispetto alla Grid Search, a parità di combinazioni valutate. Infine, nel constesto della tesi, è stata scelta la Random Search in quanto Λ presenta una dimensionalità elevata. Di conseguenza, a causa della crescita esponenziale del numero di combinazioni possibili, la Grid Search risulta inefficiente a causa del tempo di esecuzione richiesto.

2.3.2 Overfitting e Dropout

Il codice è stato implementato cercando di minimizzare l'*overfitting*. Quest'ultimo, come descritto in [13], rappresenta un problema rilevante nel contesto del DL e, in generale, nell' AI. Tale fenomeno si verifica quando il modello non generalizza sufficientemente i dati osservati, i quali definiscono il *training set*, a dati nuovi mai visti dall'algoritmo che costituiscono il *test set*. Di conseguenza, il modello ottiene ottime performance sui dati di addestramento ma fallisce con quelli di test :

tipicamente, si osserva un divario sempre più crescente tra l'errore sull'insieme di addestramento, il quale tende a valori piccoli, e l'errore sull'insieme di test, che aumenta successivamente ad una prima diminuzione. Le cause principali sono: la dimensione eccessivamente piccola del training set che, quindi, risulta essere poco rappresentativo; la smisurata complessità del modello che diventa troppo accurato sui dati di training e che, quindi, diventa sensibile a piccole variazioni nei dati.
In contrapposizione a tale fenomeno, vi è l'*underfitting*, che si verifica quando il modello risulta semplice a tal punto da non riuscire ad individuare la struttura di base dei dati. Ciò implica delle basse performance sia sul training set sia sul test set.
Questa distinzione può essere rappresentata e spiegata graficamente come in Fig. 2.7. Si ipotizzi un sistema di due assi cartesiani. Sulle ascisse è misurata la complessità del modello (*Model complexity*), definita, per esempio, dal numero o dalla tipologia di layers, oppure dal numero di neuroni costituenti un singolo strato, mentre le ordinate rappresentano l'errore (*Error*) calcolato tra l'output predetto dal modello e quello reale attraverso una specifica metrica. Sul grafico vengono tracciati gli andamenti degli errori misurati sul training set e sul test set.
L'underfitting si presenta quando il modello è semplice e le due tipologie di errore sono assimilabili possedendo un valore significativo. Se la complessità del modello viene aumentata, i due errori tendono a diminuire con un andamento differente fino ad una situazione definita ottimale (*Best fit*) in cui vi è un compromesso tra underfitting e overfitting. In questo contesto, il modello non è particolarmente complesso e riesce a generalizzare bene su dei dati di test. Infine, superata tale soglia, l'errore sui dati di addestramento tende a diminuire sempre di più ma quello sui dati di test riprende a crescere, sintomo che il modello è troppo specializzato sul training set e incapace di generalizzare. Al fine di evitare il fenomeno appena descritto, nel modello è stato utilizzato il *Dropout*, sfruttando la libreria di Tensorflow [27]. Questa tecnica è stata introdotta da Srivastava et al. nel 2014. All'interno del suo articolo [36], viene descritto come il dropout permette di mitigare l'overfitting nella rete neurale. Esso consiste nell'inibire dei neuroni seguendo una determinata distribuzione di probabilità. Più formalmente, ad ogni training step, ogni nodo del singolo strato ha una probabilità p di essere spento ponendo il suo output uguale a zero. Una volta disattivati questi ultimi, le probabilità restanti vengono scalate nel modo seguente

$$\frac{1}{1-\mathrm{p}}$$

al fine di mantenere costante la somma delle attivazioni.
Graficamente, può essere visualizzata nella Fig. 2.8 in cui è presente una rete neurale, nella quale sono stati disattivati due neuroni in entrambi gli hidden layers (nella figura sono individuati con una x).
In conclusione, mediante questa tecnica, il modello si affida meno al singolo neurone

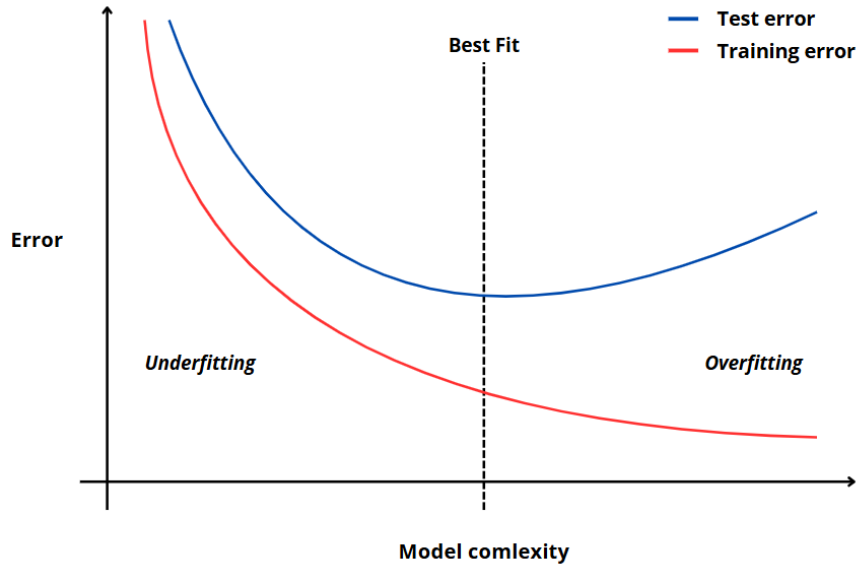

Figura 2.7: Rappresentazione concettuale di Overfitting e Underfitting.

e impara rappresentazioni più robuste e generalizzabili. Inoltre, si riscontra un miglioramento delle prestazioni e una riduzione della varianza del modello.

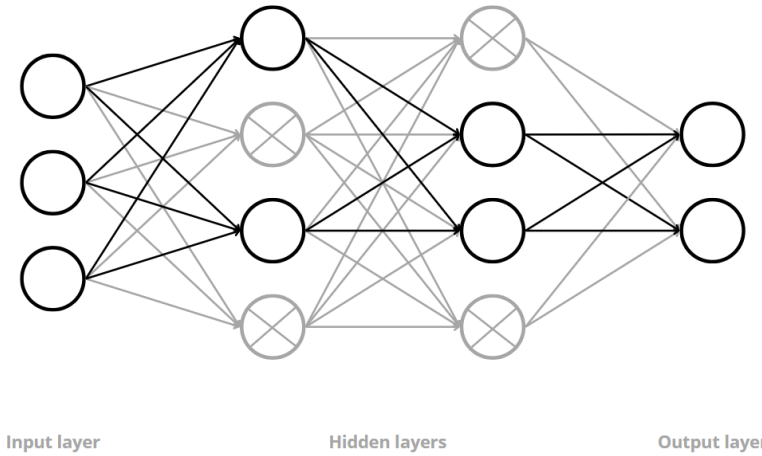

Figura 2.8: Esempio di applicazione della tecnica del dropout ad una rete neurale.

2.3.3 Callbacks

Al fine di eseguire dei controlli durante l'addestramento di un modello di rete neurale, è possibile utilizzare i *callbacks*: metodi chiamati più volte durante il processo di training con lo scopo di modificare o salvare alcuni parametri del modello. Di particolare rilevanza sono l'*Earlystopping* e il *ReduceLROnPlateau*, entrambi presenti nella libreria di TensorFlow [27] e utilizzati nel codice sviluppato. Il primo di questi ultimi è una tecnica utilizzata principalmente per ridurre l'overfitting. L'idea alla base è di interrompere la fase di addestramento in anticipo rispetto al completamento del numero di epoche impostato. Tale arresto anticipato evita che il modello impari a memoria i dati di training e permette una maggiore capacità di generalizzare sui dati di test [37]. Nella pratica, l'interruzione avviene quando il valore di una determinata quantità monitorata non presenta miglioramenti per un numero definito di epoche consecutive [27]. Tale misura è la *loss*, anche definita come funzione di perdita, che svolge un ruolo centrale nei modelli di Deep Learning. Essa viene introdotta nell'ambito della matematica classica, ma assume un ruolo operativo per il DL grazie agli studi di Rumelhart et al. pubblicati nel paper [18] riguardante la retro-propagazione dell'errore. La loss è utilizzata per ottimizzare i parametri di un modello durante il processo di addestramento di quest'ultimo. Nel dettaglio, essa quantifica la differenza tra l'output atteso e quello predetto dall'algoritmo. A tal fine, esistono differenti tipologie di loss, ognuna con caratteristiche diverse. Nel codice sviluppato per la tesi, la loss utilizzata è il MSE, *Mean Squared Error*: esso misura la media del quadrato della differenza tra i valori predetti $\hat{y}$ e quelli reali y. Matematicamente [38]:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)^2$$

dove n è il numero di campioni del mini-batch considerando la singola iterazione. Tale quantità è non negativa e uguale a zero se e solo se ogni predizione coincide con il valore reale. Inoltre, si noti che tende a penalizzare le differenze maggiori in quanto elevate al quadrato, ma riduce quelle minori di uno.
La seconda tecnica utilizzata è il *ReduceLROnPlateau*, ovvero la riduzione del *learning rate* tramite uno specifico fattore. Tale callback monitora la loss e, se quest'ultima non migliora, procede con una diminuzione del learning rate [27]. Quest'ultimo è un iperparametro fondamentale per i modelli di deep learning in quanto esso regola l'ampiezza dei passi nella direzione del gradiente durante l'aggiornamento dei parametri del modello.
A tal fine, esistono differenti tipologie di ottimizzatori che regolano iterativamente i parametri del modello. Nel codice sviluppato per la tesi, viene utilizzato l'ottimizzatore ADAM, *Adaptive Moment Estimation*, introdotto da Kingma e Ba [39]. Esso rappresenta un'evoluzione rispetto alla SGD, *Stochastic gradient descent*, basata

sul gradiente della funzione di perdita e introdotto da Bengio et al[40]:

$$w_t = w_{t-1} - \xi_t \frac{\partial E(z_t, w)}{\partial w} \tag{2.5}$$

dove wt e w_{t-1} sono rispettivamente i vettori di parametri del modello all'iterazione t e $t-1$ e z_t è il mini-batch utilizzato all'iterazione t. Inoltre E rappresenta la loss calcolata su z_t all'iterazione t e ξ_t è il learning rate in t.
Bengio et al sottolineano come la scelta di un valore ottimale di quest'ultimo ξ^* risulta cruciale al fini della riuscita dell'addestramento del modello. Come si può notare in Fig. 2.9, un learning rate troppo elevato, $\xi_{high} > \xi^*$, può causare divergenza durante il training portando, di conseguenza, ad un aumento della loss. Al contrario, un valore troppo basso, $\xi_{low} < \xi^*$, può rallentare considerevolmente la convergenza o bloccare il modello in minimi locali [40].

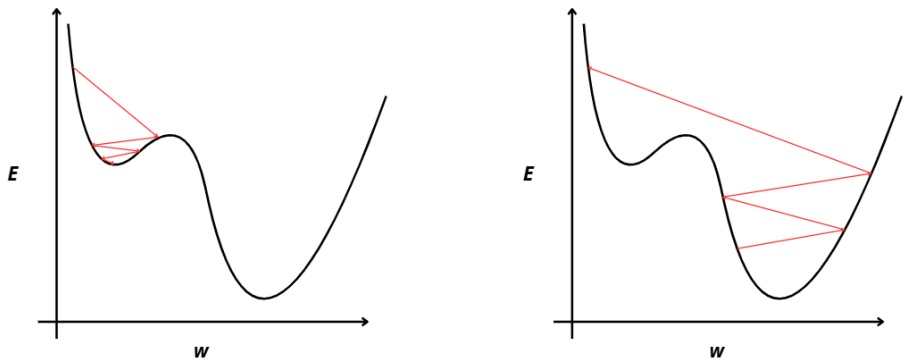

Figura 2.9: A sinistra, esempio qualitativo di learning rate basso, a destra, esempio qualitativo di learning rate alto.

Invece, ADAM risulta efficace e, allo stesso tempo, stabile, in quanto permette di combinare i vantaggi di due tecniche: il momentum e l'algoritmo del learning rate adattivo. In particolare, il primo si riferisce alla media del gradiente della loss, ovvero il primo momento, che consente di accelerare la convergenza e di ridurre le oscillazioni. Inoltre, il secondo permette di modificare automaticamente il passo di aggiornamento dei parametri basandosi sulla varianza del gradiente considerando ogni parametro singolarmente. Di conseguenza, i gradienti più instabili hanno aggiornamenti del learning rate più piccoli portando un vantaggio riguardo la stabilità e la precisione durante la discesa del gradiente.

Matematicamente, ADAM si può definire nel modo seguente:

$$w_{t+1} = w_t - \tilde{\alpha} \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \bar{\epsilon}}}$$

in cui θ_{t+1} e θ_t sono definiti come nel SGD ma all'iterazioni $t+1$ e t, $\tilde{\alpha}$ è il learning rate e $\bar{\epsilon}$ è un termine di stabilità numerica inserito al fine di evitate divisioni per zero. $\hat{m}_t$ e $\hat{v}_t$ sono rispettivamente versioni corrette mediante un bias di m_t, stima del primo momento, e v_t, stima del secondo momento, in quanto quest'ultimi risultano troppo vicini a zero:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

β_1^t e β_2^t sono rispettivamente i coefficienti di decadimento per m_t e v_t che sono definiti come segue.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\bar{g}_t \qquad v_t = \beta_2 v_{t-1} + (1 - \beta_2)\bar{g}_t^2$$

dove $\bar{g}_t$ è il gradiente della funzione di perdita rispetto ai parametri w.
Kingma e Ba evidenziano che l'utilizzo di un ottimizzatore come ADAM è più vantaggioso in differenti casi pratici. Tale metodo ottiene prestazioni comparabili o migliori con una minore sensibilità al tuning degli iperparametri e una maggiore convergenza. Tuttavia, risulta ancora valido il discorso precedentemente introdotto riguardo l'uso di un learning rate adeguato evitando valori troppo alti o bassi.

Capitolo 3

Sviluppo del problema

Al fine di motivare la provenienza e l'utilizzo dei dati mostrati nel Cap. 4, si procede analizzando la creazione delle immagini SAR e la tecnologia InSAR alla base delle misurazione dei displacements. Inoltre si presentano i ponti di Tadcatser e Cantiano, utilizzati come training e testing dei modelli.
Successivamente, si procede formalizzando i due modelli creati come funzioni parametriche. In particolare, si pone l'attenzione sulla costruzione dell'input e sulla composizione dell'output dei modelli.

3.1 Generazione dati sintetici da immagini SAR

Le immagini SAR, *Synthetic Apertur Radar*, rappresentano una delle più recenti ed innovative fonti di dati utili per monitorare il territorio e le relative infrastrutture. Esse sono create mediante sistemi radar montati su satelliti che orbitano attorno alla terra ad una distanza massima di circa 800 km. Inoltre, essi percorrono traiettorie inclinate di pochi gradi rispetto ai meridiani e la loro orbita viene definita discendente, dal polo Nord al polo Sud, e ascendente, dal polo Sud al polo Nord. I radar montati sui satelliti emettono segnali elettromagnetici verso la superficie terrestre lungo la LOS, *line of sight*, che corrisponde alla direzione lungo la quale i sensori acquisiscono le immagini e le relative informazioni sulla superficie terrestre [1]. Successivamente, i sensori ricevono il segnale retrodiffuso.
Tali strumenti sfruttano il concetto di Apertura Sintetica: il satellite utilizza un'antenna di dimensione minime montata su di esso che viene spostata da un sensore lungo la direzione di volo. Come è visibile nell'immagine 3.1, le posizioni coperte dall'antenna sono sequenziali e, da ognuna di esse, l'antenna trasmette un impulso e riceve il suo eco di ritorno. In seguito, quest'ultima sintetizza un vettore che rappresenta l'allineamento di antenne con lunghezza fissata lungo la direzione di volo. Il risultato è un'antenna con una apertura di segnale maggiore di quella

reale[5].

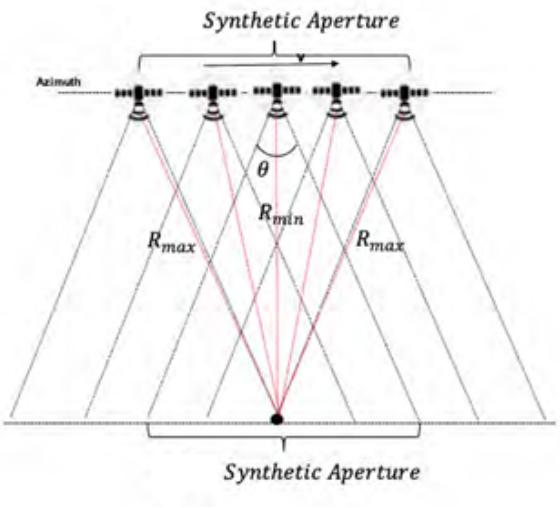

Figura 3.1: Rappresentazione qualitativa dell'apertura sintetica [5]; R_{max} e R_{min} sono rispettivamente il raggio massimo e minimo.

L'elaborazione del segnale è un processo chiamato Focalizzazione [41] e consente di ottenere un'immagine digitale rappresentata da una matrice complessa di pixel. Ognuno di questi ultimi, contiene l'informazione sull'ampiezza del segnale retrodiffuso, che dipende dalla forza della riflessione della superficie, e la fase, la quale dipende dalle proprietà geometriche ed elettromagnetiche della scena osservata. L'immagine risultante presenta differenti tonalità di grigio in base alla luminosità dell'area: infatti, maggiore è la luminosità, maggiore è la capacità di riflettere il segnale derivante dal radar. Tale proprietà dipende dalla tipologia di segnale trasmesso ma soprattutto dalle caratteristiche fisiche e geometriche della scena illuminata. Per esempio, le superfici piatte, come le strade, hanno una bassa retrodiffusione: infatti, nella immagini SAR appaiono molto scure risultando facilmente distinguibili. Invece, le zone con molte costruzioni, per esempio i centri abitati, risultano molto chiare in quanto vi è la possibilità che il segnale rimbalzi più volte prima di tornare indietro.
É importante sottolineare che un'immagine SAR risulta differente, non solo nei colori ma anche geometricamente, da un'immagine ottica che ritrae la stessa superficie. Inoltre, gli effetti delle distorsioni geometriche nelle immagini SAR cambiano notevolmente a seconda dell'angolo di vista con il quale il sensore satellitare opera: di conseguenza, due immagini SAR che ritraggono la stessa superficie possono risultare differenti tra di loro [5].
Un'importante applicazione consiste nel confrontare due o più immagini SAR acquisite dallo stesso punto di vista e che ritraggono la stessa porzione di superficie,

al fine di rilevare spostamenti dell'ordine di centimetri e millimetri. Tale tecnica è chiamata interferometria SAR, InSAR, e sfrutta due immagini successive al processo di Focalizzazione con le proprietà descritte in precedenza [42]. Dal loro confronto è possibile definire un interferogramma: un'immagine complessa della quale ciascun pixel rappresenta la differenza di cammino ottico tra il radar e la superficie, e viceversa, tra i due ritratti. Inoltre, ogni pixel è matematicamente definito come un numero complesso. Quindi, l'interferogramma è la moltiplicazione di un numero complesso con il proprio coniugato. Al fine di produrre un risultato di qualità, ovvero privo di frange rumoreose o informazioni parzialmente errate, è necessario procedere ad una coregistrazione accurata delle immagini. Tale tecnica viene effettuata stimando e correggendo traslazioni, rotazioni e deformazioni non lineari tra le due immagini SAR. L'obiettivo è di rendere minimo l'errore residuo nell'associazione tra due pixel [4].
Basandosi sulla geometria dalla quale sono state scattate le immagini, si possono estrarre informazioni differenti. Infatti, se esse provengono da orbite diverse, almeno in parte, la fase relativa permette di stimare l'altezza dei punti sulla superficie, definiti attraverso due coordinate spaziali. Da tali misure si producono modelli digitali di elevazione, DEM, aventi la risoluzione spaziale tipica di un'immagine SAR. Invece, se esse sono create con la medesima geometria di acquisizione, ma in istanti temporali differenti, la differenza di fase permette di identificare displacements anche millimetrici della superficie lungo la linea di di vista del radar [42].
La tecnologia InSAR è stata applicata ad entrambi i ponti usati per popolare il dataset di training e test in questa tesi. Il primo è il ponte di Tadcaster, che è situato nella cittadina di Tadcaster, nel North Yorkshire, in Inghilterra. Esso è un ponte ad archi in muratura lungo circa 100 metri e largo 10 metri costituito da due porzioni affiancate e realizzate in epoche differenti. Il ponte venne edificato nel 1698-1699 e fu ampliato nel 1791-1792. Durante il corso della sua vita, esso non ha subito ingenti danni fino al 29 dicembre 2015: un periodo di intense piogge e inondazioni ha causato il cedimento a monte di un pilastro, come visibile in basso a destra in Fig. 3.2. Analisi approfondite hanno riscontrato che il crollo parziale fu causato del fenomeno dello scour, ovvero l'erosione del materiale del letto del fiume attorno alle fondamenta del pilone che è crollato di conseguenza. A tale struttura è stata applicata la tecnica SBAS InSAR, mediante la quale è stata prodotto un grafico di movimenti temporali. Esso mostra i displacements di alcuni punti, appartenenti al ponte e limitrofi ad esso, nella LOS del satellite SAR nel tempo, tracciato come movimento relativo rispetto alla posizione del ponte rilevata durante la prima acquisizione [10].
Il secondo ponte analizzato è un ponte pedonale collocato in Via Flaminia Sud circa all'altezza del civico 9 del comune di Cantiano. Grazie alla modalità Street View di Google Maps, in Fig. 3.3, si può notare la situazione nel mese di giugno 2011, a sinistra, ed la struttura rimanente a ottobre 2022 dopo l'alluvione che ha

colpito Cantiano nel settembre dello stesso anno. Le immagini SAR sono state elaborate, mediante la tecnologia A-DInSAR con dati acuisiti dalla costaellazione di satelliti COSMO-SkyMed, e sottoposte ad un processo di post elaborazione dei dati. In particolare, si sono ottenute serie temporali di spostamento con accuratezza millimetrica dalle quali è possibile estrarre informazioni sulle eventuali deformazioni [11].

Figura 3.2: Schema del ponte che mostra la porzione crollata in rosso relativa al quinto pilone; in basso a destra, una foto del ponte dopo il collasso successiva al ritirarsi della piena del fiume [10].

Figura 3.3: Zona del ponte pedonale di Cantiano nel mese di luglio del 2011, a sinistra [43], e nell'ottobre del 2022, a destra [44].

3.2 Formalizzazione del problema di predizione

Il problema affrontato consiste nel predire con il minor errore possibile un indice di rischio, la cui definizione è spiegata nel dettaglio nel Cap. 4. Tale obiettivo può essere complicato in termini di risorse computazionali. Di conseguenza, si definisce un problema più semplice ma efficiente in egual misura. In particolare, si illustra un modello che ha l'obiettivo di quantificare correttamente gli incrementi relativi a un determinato indice di rischio in una finestra temporale di 14 giorni al fine di risalire, in seguito, al rischio di crollo su tutto il periodo relativo alle misurazioni effettuate. Di conseguenza, si considerano due modelli differenti. Il primo è chiamato GeoDispNet, *Geospatial Displacements Network* e prende in input i dati temporali di spostamenti verticali misurati in millimetri, i displacements, e le loro coordinate geometriche. Invece, il secondo è HydroGeoDispNet, *Hydrological data Geospatial Displacements Network* e, oltre ai dati sopracitati, tiene conto anche della portata del fiume. Di seguito, i dettagli di entrambi i modelli sviluppati relativi l'input, i target e l'output del modello. Invece, gli elementi rilevanti dell'architettura sono approfonditi nel Cap. 4.

3.2.1 GeoDispNet

L'input ed i target sono definiti attraverso due funzioni create ad hoc, alle quali sono dati come input un dataset contenente le coordinate, D_c, e uno relativo ai displacements, D_s. Una, f_{rand}, è usata nella fase di training, validation e testing del modello, mentre l'altra, f_{full}, è utilizzata per effettuare delle predizioni su scenari che non rientrano nel set di addestramento.
In generale, si ha l'obiettivo di definire un modello che sia in grado di prevedere il rischio indipendentemente dal numero di punti del ponte passati in input. Infatti, considerando la procedura di creazione del dataset approfondita nel Cap. 4, è possibile avere fino ad un massimo di nove punti. Al fine di gestire tale variabilità, per i ponti che presentano un numero minore di punti, vengono creati rispettivamente dei punti fittizi con relativi attributi scelti a priori. Lo scopo di tali scelte è che il modello impari a distinguere i punti reali dai punti creati a mano, conferendo a questi ultimi un'importanza minore.
Inizialmente, si crea un tensore tridimensionale, Fig. 3.5, e definito come

$$C \in \mathbb{R}^{m \times n \times 3}$$

dove m è il numero totale di finestre temporali che vengono campionate, mentre n è il numero massimo possibile di punti del ponte. In particolare, ogni elemento di C è così definito

$$C_{j,i} = (x_{j,i}, z_{j,i}, b_{j,i}), \quad j = 1, ..., m; \quad i = 1, ..., n$$

dove (x_i, z_i) sono le coordinate spaziali del generico punto i, per ogni j e $b_i \in \{-1,1\}$ rappresenta rispettivamente un punto fuori dal ponte o del ponte. Tali valori fanno riferimento ad una colonna di D_c avente valori in $\{0,1\}$. La scelta di effettuare la trasformazione $\{0,1\} \longrightarrow \{-1,1\}$ permette di avere una variabile con una distribuzione bilanciata e centrata in zero rispetto ad una sbilanciata con min = 0 e max = 1. Ciò può facilitare l'addestramento in quanto può portare ad una riduzione del rumore in alcuni strati del modello, per esempio nei Dense Layer.
Al fine di migliorare la robustezza del modello riguardo rotazioni o traslazioni del ponte, si procede con un riordinamento ed una normalizzazione delle coordinate. In particolare, i punti del ponte vengono orientati nello spazio applicando la PCA basata solamente sui punti del ponti, $b_i = 1$. Quindi, ogni punto i si dispone seguendo le due componenti principali $(PC_{1,i}, PC_{2,i})$ che corrispondono rispettivamente alla lunghezza l e alla larghezza w del ponte. Si presuppone che, in generale, un ponte abbia $l > w$: quindi, in seguito alla PCA, si prevede che il ponte sia disposto con l allineato all'asse orizzontale. Successivamente, si effettua una traslazione di quest'ultimo e si definiscono le nuove coordinate:

$$(x_i^*, z_i^*) = (PC_{1,i}, PC_{2,i}) - (x_{mid}, z_{mid})$$

dove $(x_{mid}, z_{mid}) = \frac{1}{2}(P_{max} + P_{min})$ sono le coordinate del centroide del ponte con $P_{max} = (PC_{1,max}, PC_{2,max})$ e $P_{min} = (PC_{1,min}, PC_{2,min})$, rispettivamente coordinate massime e minime dopo l'applicazione della PCA. Si procede con una scalatura tale che la proiezione orizzontale dei punti sia nell'intervallo $[-1,1]$ ottenendo le coordinate finali $(\hat{x}_i^*, \hat{z}_i^*)$. Infine, se il ponte presenta un numero di punti strettamente inferiore a nove, vengono creati manualmente punti p_{out} esterni al ponte in numero pari alla differenza, quindi $b_{p_{out}} = -1$. Inoltre, essi sono posizionati a sinistra della struttura, dunque, hanno coordinate $(x_{p_{out}}, z_{p_{out}}) = (\min_i(\hat{x_i}^* - 1), 0)$. Il risultato di tale processo è visibile in Fig. 3.4. Nel primo plot, a sinistra, è possibile vedere il ponte di Tadcaster originale, mentre nel secondo si nota il riorientamento tramite PCA e la scalatura. Infine, nel terzo è visibile anche il punto *p_out* inserito manualmente a sinistra del ponte in una posizione esterna.
Successivamente, C viene combinato con un altro tensore tridimensionale, anch'esso visibile in Fig. 3.5:

$$X \in \mathbb{R}^{m \times n \times d},$$

dove m, n sono definiti come prima e d rappresenta la lunghezza della finestra temporale, pari a 14 giorni. Gli elementi $X_{j,i,}$ sono, per ogni finestra temporale $j = 1, ..., m$, i displacements del punto i nella finestra temporale avente d giorni. Anche in questo caso, se il ponte ha strettamente meno di nove punti, si aggiungono dei punti p_{out} fittizi tali da avere un spostamento verticale nullo in tutta la finestra temporale, quindi $disps_{s,p_{out}} = 0$, con $s = 1, ..., d$. Infine, l'input del modello è la concatenazione dei tensori X e C, $I = (X, C) \in \mathbb{R}^{m \times n \times (d+3)}$.

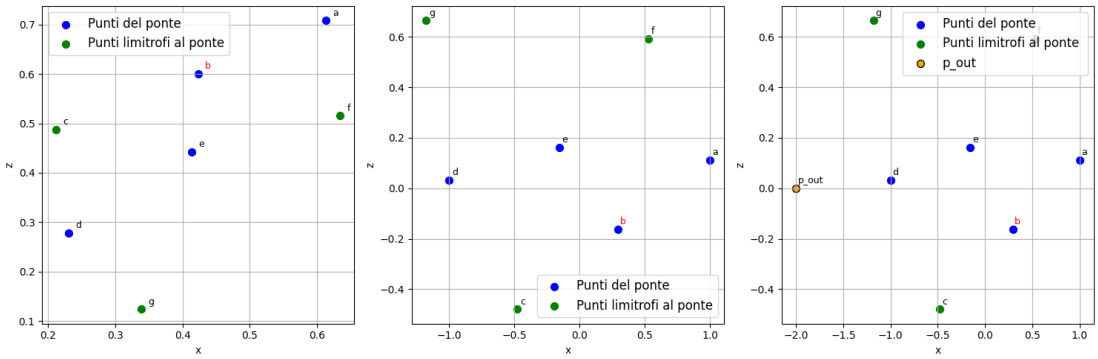

Figura 3.4: Applicazione della PCA al ponte di Tadcaster e posizionamento manuale dei punti p_{out}.

Il target Y associato a ciascuna finestra temporale è rappresentato da una sequenza di incrementi di rischio *incr* nella stessa finestra temporale di I:

$$Y \in \mathbb{R}^{m \times d}.$$

Il singolo elemento è $Y_{j,:} = (y_t, y_{t+1}, ..., y_{t+d-1})$ dove y_t rappresenta l'incremento di rischio al tempo t. É importante notare che ogni y_t è stato normalizzato dividendolo per un fattore α_{inc} pari al 97 % del valore massimo di tale indice osservato nel training set $\mathcal{D}_{\text{train}}$:

$$\alpha_{inc} = 0.97 \cdot \max_{Y_{j,i} \in \mathcal{D}_{\text{train}}} Y_{j,i}.$$

Lo scopo è di avere un insieme di training di incrementi di rischio con valori anche lievemente maggiori di uno, per rendere robusto il modello rispetto la capacità di generalizzazione.
La funzione f_{rand} prende casualmente le m finestre temporali di dimensione d al fine di strutturare il tensore in modo bilanciato e vario, utile per l'addestramento e la validazione. Al contrario, f_{full} viene usata per effettuare testing su un intero dataframe: infatti, le finestre temporali sono generate in modo sequenziale e coprono l'intero periodo di tempo. Quindi $m = T - d + 1$ con T numero totale delle date delle misurazioni. Ciò permette di ricostruire l'incremento del rischio su tutto l'orizzonte temporale.
GeoDispNet è rappresentato qualitativamente con un quadrato in Fig. 3.5 e è definito mediante una funzione parametrizzata

$$g_{base,\theta} : \mathbb{R}^{m \times n \times (d+3)} \longrightarrow \mathbb{R}^{m \times d}$$

tale che, considerando una finestra temporale campionata j, predice l'output

$$\hat{y}^{\,j} = g_{base,\theta}(I_{j,:,:}) = (\hat{y}^j_t, \hat{y}^j_{t+1}, ..., \hat{y}^j_{t+d-1}) \cdot \alpha_{inc}.$$

che rappresenta l'incremento di rischio predetto in ognuno dei d giorni della finestra temporale riscalato. Si noti che le finestre temporali create possono avere uno o più giorni t in comune tra loro. Al fine di ottenere un unico incremento di rischio in ognuna delle date del periodo temporale, si definisce una media delle predizioni in ogni istante t ignorando gli incrementi nulli:

$$\tilde{y}_t = \frac{1}{|\mathcal{J}(t)|} \sum_{j \in \mathcal{J}(t)} \hat{y}_t{}^j$$

dove $\mathcal{J}(t)$ è l'insieme delle finestre temporali che includono t mentre $|\mathcal{J}(t)|$ rappresenta la sua cardinalità. Tale processo, iterato per ogni istante t, produce la serie temporale degli incrementi di rischio predetti:

$$\tilde{y}_1, ..., \tilde{y}_T.$$

Infine, per ottenere l'indice di rischio predetto all'istante t, si effettua una somma cumulata degli incrementi predetti in t:

$$\tilde{r}_t = \sum_{\tau=1}^{t} \tilde{y}_\tau, \quad \text{con } t = 1, ..., T. \tag{3.1}$$

Tale misura permette di valutare visivamente se l'andamento del rischio stimato segue l'evoluzione temporale del rischio reale r_k con k che identifica il tipo di rischio. La rappresentazione grafica fornisce un paragone immediato riguardo la correttezza della predizione. Al fine di valutare quantitativamente tale predizione, si determina l'errore assoluto tra il rischio predetto e quello reale per visualizzare gli istanti in cui il modello commette errori in misura maggiore lungo la serie temporale. Quindi, si calcolano:

$$e_{abs,t} = |\tilde{y}_t - y_t|, \qquad e_{sqt,t} = (\tilde{y}_t - y_t)^2 \tag{3.2}$$

con $t = 1, ..., T$.
Al fine di valutare l'addestramento del modello, si quantificano le sue performance sul test set mediante due metriche chiamate Loss_{test} e MAE_{org} che considerano α_{inc}:

$$\text{Loss}_{test} = \frac{1}{M} \sum_{i=1}^{M} \left(\hat{y}_i - \frac{y_i}{\alpha_{inc}} \right)^2 \qquad \text{MAE}_{org} = \frac{1}{M} \sum_{i=1}^{M} |\hat{y}_i - y_i| \tag{3.3}$$

dove M è il numero di campioni del test set.
Al fine di misurare più approfonditamente l'accuratezza delle predizioni del modello, si calcolano MAE, *mean absolute error*, e MSE, *mean squared error* su scenari differenti. Considerano gli incrementi predetti all'interno della j-esima finestra temporale $\hat{y}^j$, si calcola:

$$\text{MAE}_{pred} = \frac{1}{m \cdot d} \sum_{j=1}^{m} \sum_{\tau=1}^{d} |\hat{y}_\tau^j - y_\tau^j|, \qquad \text{MSE}_{pred} = \frac{1}{m \cdot d} \sum_{j=1}^{m} \sum_{\tau=1}^{d} (\hat{y}_\tau^j - y_\tau^j)^2 \tag{3.4}$$

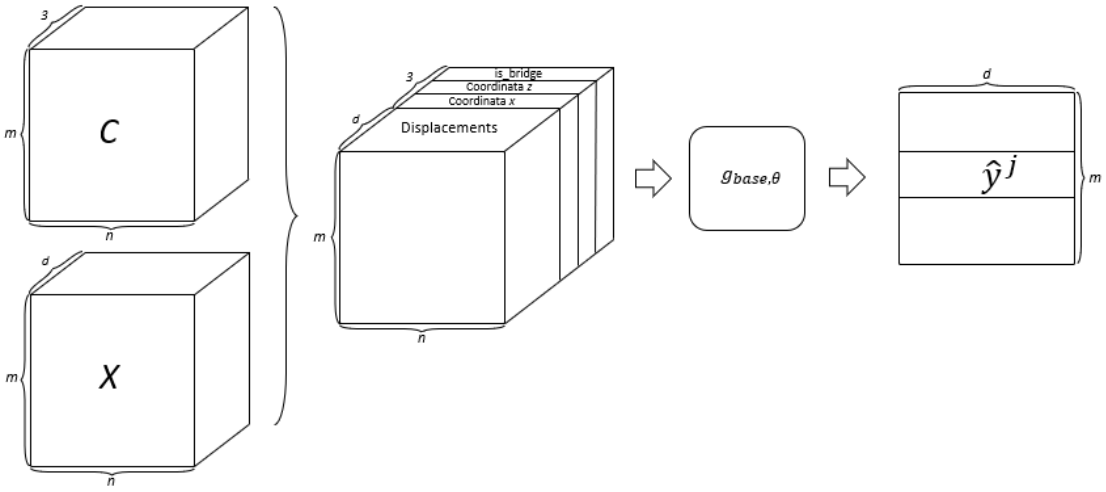

Figura 3.5: Schema qualitativo della formalizzazione del modello GeoDispNet.

con m e d definite come sopra. Tale misurazione permette di confrontare la performance del modello riguardo l'output precedente a successive elaborazioni considerando tutti i giorni di tutte le finestre temporali.
Inoltre, si definiscono:

$$\text{MAE}^k_{day} = \frac{1}{m}\sum_{j=1}^{m}|\hat{y}^j_{t+k-1} - y^j_{t+k-1}|, \qquad \text{MSE}^k_{day} = \frac{1}{m}\sum_{j=1}^{m}(\hat{y}^j_{t+k-1} - y^j_{t+k-1})^2 \quad (3.5)$$

con $k = 1, ..., d$. In tal caso, vengono prese in considerazione tutte le m finestre temporali ma si misurano le metriche per ognuno dei k giorni.
Dopo aver calcolato $\tilde{y}$, si definiscono nuovamente

$$\text{MAE}_{post} = \frac{1}{T}\sum_{t=1}^{T}|\tilde{y}_t - y_t|, \qquad \text{MSE}_{post} = \frac{1}{T}\sum_{t=1}^{T}(\tilde{y}_t - y_t)^2 \quad (3.6)$$

con T definito come sopra. Tali misure considerano gli incrementi successivi al calcolo della media che esclude i valori nulli e li valuta su tutto l'arco temporale.

3.2.2 HydroGeoDispNet

In quanto il secondo modello presenta molte caratteristiche in comune con il primo, di seguito verranno analizzate solamente le principali differenze.
Come è possibile visualizzare in Fig. 3.6, l'input di HydroGeoDispNet è composto dall'unione di due elementi. Il primo è la coppia (X, C), che rappresenta l'informazione riguardo i displacements e le coordinate. Il secondo, anch'esso creato

mediante f_{rand} e f_{full}, è un tensore bidimensionale

$$G \in \mathbb{R}^{m \times d}$$

tale che $G_{j,i}$, con $j = 1, ..., m$ e $i = 1, ..., d$, rappresenta il valore della portata del fiume Wharfe in ogni i-esimo giorno per ciascuna finestra temporale j. Quindi, l'input è

$$I = (X, C) \cup G.$$

Il modello è definito come

$$g_{gdf,\theta} : (\mathbb{R}^{m \times n \times (d+3)}, \mathbb{R}^{m \times d}) \longrightarrow \mathbb{R}^{m \times d},$$

che, considerando la j-esima finestra temporale, restituisce la sequenza degli incrementi di rischio

$$\hat{y}^{\,j} = g_{gdf,\theta}((X_{j,:,:}, C_{j,:}), G_{j,:}) = (\hat{y}^j_t, \hat{y}^j_{t+1}, ..., \hat{y}^j_{t+d-1}).$$

La procedura per creazione dell'indice di rischio e le metriche utilizzate per verificare l'accuratezza del modello sono comparabili a quelle utilizzate per GeoDispNet.

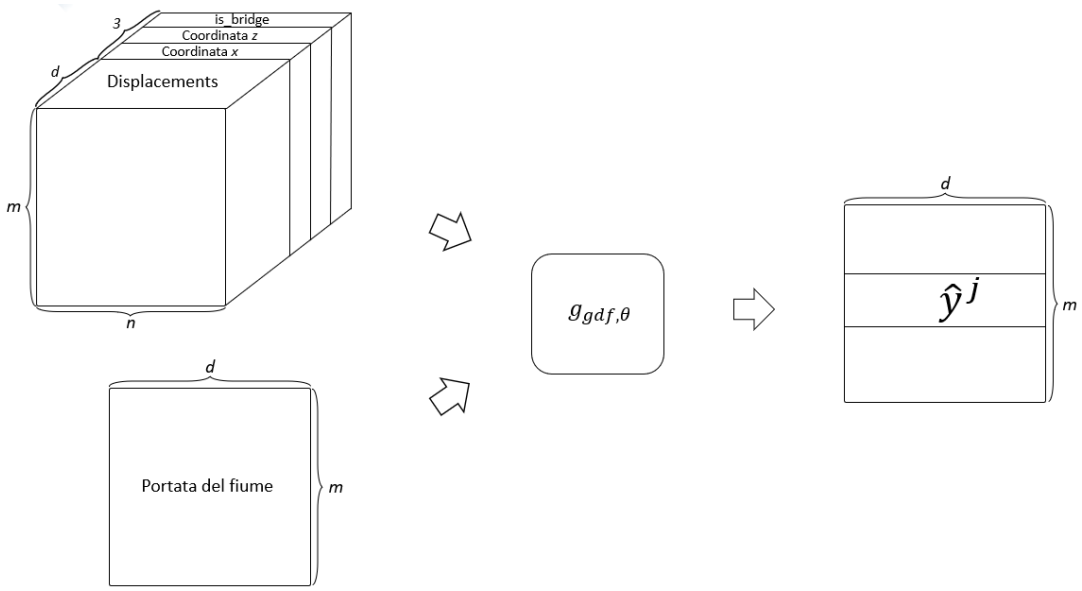

Figura 3.6: Schema qualitativo della formalizzazione del modello HydroGenDispNet.

Capitolo 4

Esperimenti

In questo capitolo, si analizza il dataset a disposizione. In particolare, si approfondisce la struttura dei due file principali contenenti le informazioni spaziali, gli spostamenti verticali dei punti, gli indici di rischio ed i relativi incrementi. Inoltre, si spiegano le tecniche di *data augmentation* applicate ai files del ponte di Tadcaster e di *data trasformation* al fine di realizzare i files del ponte di Cantiano. Si pone l'attenzione sul calcolo di indici di rischio *data driven* basati sui displacements e, per Tadcaster, sulla portata del fiume Wharfe.
Successivamente, si analizza l'architettura dei due modelli, formalizzati nel Cap. 3, spiegando il flusso dei dati di input attraversi i Layers che ne costituiscono la struttura.
Infine, si effettua un'approfondita analisi dei risultati ottenuti dai due modelli in più scenari. I risultati sono valutati qualitativamente, attraverso grafici di predizione e di andamento di errori, e quantitativamente, mediante l'utilizzo delle metriche introdotte nel capitolo precedente. In ultimo, si valutano i due modelli con l'ausilio di due metriche comuni.

4.1 Datatset

Al fine di costruire il dataset utilizzato, vengono considerate due strutture differenti: il ponte di Tadcaster, utilizzato in tutte le fasi del processo di sperimentazione costituito da tuning degli iperparametri, addestramento e test del modello, ed il ponte di Cantiano, impiegato unicamente come test del modello. Per ognuno di essi, sono stati esaminati due file distinti: *points* e *inferred*, corrispondenti rispettivamente a D_c e D_s, e approfonditi nei paragrafi a seguire. In generale, il primo contiene le coordinate dei punti sul ponte e limitrofi. Invece, il secondo comprende i displacements espressi in millimetri dei punti presenti in *points* ed i corrispondenti indici di rischio. Tali file vengono elaborati e modificati per essere

integrati con il modello. La procedura seguita è simile per entrambi i ponti ma presenta delle differenze non trascurabili che vengono analizzate di seguito.

4.1.1 Ponte di Tadcaster

File di partenza

I dati a disposizione per tale struttura derivano da un processo di estrazione delle informazioni dal paper [10] eseguito a priori. I relativi valori sono riportati interamente nei due file precedentemente introdotti. Nella Tabella 4.1 sono riportati i dati del file *points*. In particolare, vi sono i nomi identificativi dei sette punti del ponte presi in considerazione, colonna *point*, mentre, nella colonna b, viene riportato un valore numerico uguale a uno se il punto corrisponde ad una porzione di ponte o zero altrimenti. Infine, nelle colonne x e z sono indicate la latitudine e la longitudine normalizzate tra zero e uno. La loro disposizione si può vedere in Fig. 4.1 nella quale i punti evidenziati in blu rappresentano luoghi sul ponte, quelli colorati in verde sono limitrofi al ponte ed, infine, il punto b identifica la porzione crollata.

point	b	x	z
a	1	0.613208	0.708097
b	1	0.423544	0.600994
c	0	0.212594	0.487996
d	1	0.230884	0.278521
e	1	0.413625	0.442218
f	0	0.633992	0.515512
g	0	0.338683	0.123869

Tabella 4.1: Rappresentazione del file *points* del ponte di Tadcaster. Valori delle colonne x e z approssimati alla sesta cifra decimale.

Le principali caratteristiche dei punti del ponte si trovano nel file *inferred*. Come mostrato in Tabella 4.2, la colonna *date* contiene le date in cui sono stati misurati i displacements che coprono un periodo temporale di 623 giorni dal 14 marzo 2014 al 26 novembre 2015. In seguito, vi è una colonna *step* che contiene dei valori numerici equispaziati tra zero e uno: essa è utile per definire e stampare plots. Le colonne successive sono pari al numero di punti del ponte e sono chiamate con il nome del punto corrispondente riportato nella colonna *point* della Tabella 4.1. In quest'ultime, sono riportati i valori dei displacements e, in particolare, si ha a disposizione una misurazione al giorno per ogni punto considerato. Si noti che i displacements relativi alla prima data sono impostati a zero in quanto si considera il primo giorno di rilevazione come punto di partenza dalla quale vengono calcolati

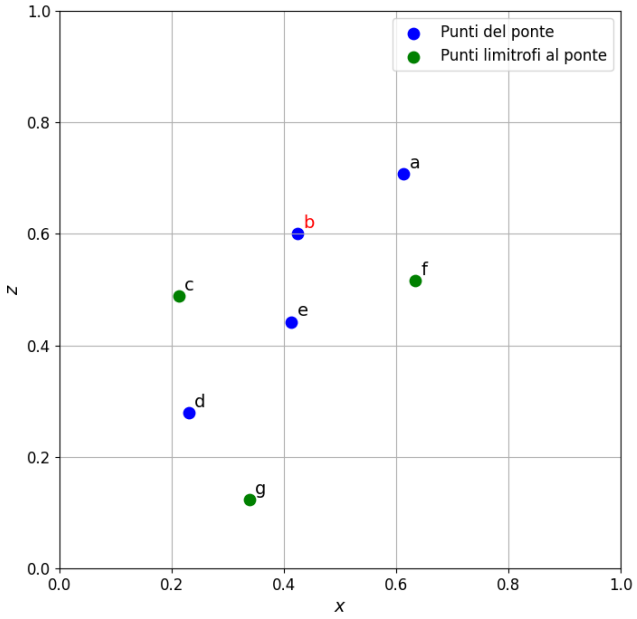

Figura 4.1: Plot dei punti riferiti al ponte di Tadcaster.

gli spostamenti verticali dei giorni successivi.
A titolo esemplificativo, si mostra l'andamento dei displacements del punto b, $\Delta_{t,b}$, che rappresenta la porzione di ponte crollata, del punto sul ponte e, $\Delta_{t,e}$, e di un punto limitrofo c, $\Delta_{t,c}$, nella Fig. 4.2. Si noti come l'andamento del punto b è rapidamente decrescente a partire da un determinato istante temporale in avanti.

date	*step*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
2014-03-14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2014-03-15	0.001616	-0.072404	-0.249532	0.109668	0.301884	0.026386	-0.093642	0.102178
2014-03-16	0.002424	-0.165415	-0.478953	0.020121	0.317585	0.081577	0.053158	0.147250
2014-03-17	0.003232	-0.134015	-0.329929	-0.055375	0.237279	0.021060	0.203994	0.176461
2014-03-18	0.004848	-0.038448	-0.113063	0.002897	0.083162	0.007622	0.108425	0.088656
2014-03-19	0.006464	0.010469	0.013346	-0.076899	0.111867	0.158586	0.150165	0.122606
2014-03-20	0.008080	-0.052825	-0.065792	-0.093375	0.161405	0.100918	0.134017	0.112952
2014-03-21	0.008888	0.076800	0.059275	-0.063162	0.088612	0.019524	0.100058	0.116668
2014-03-22	0.010505	0.016339	0.045610	-0.015093	0.076395	0.047806	0.106646	0.096338
2014-03-23	0.011313	0.013627	-0.001648	0.004128	0.037430	0.009415	0.089851	0.093202

Tabella 4.2: Rappresentazione delle prime dieci righe della prima parte del file *inferred* del ponte di Tadcaster. Valori delle colonne approssimati alla sesta cifra decimale.

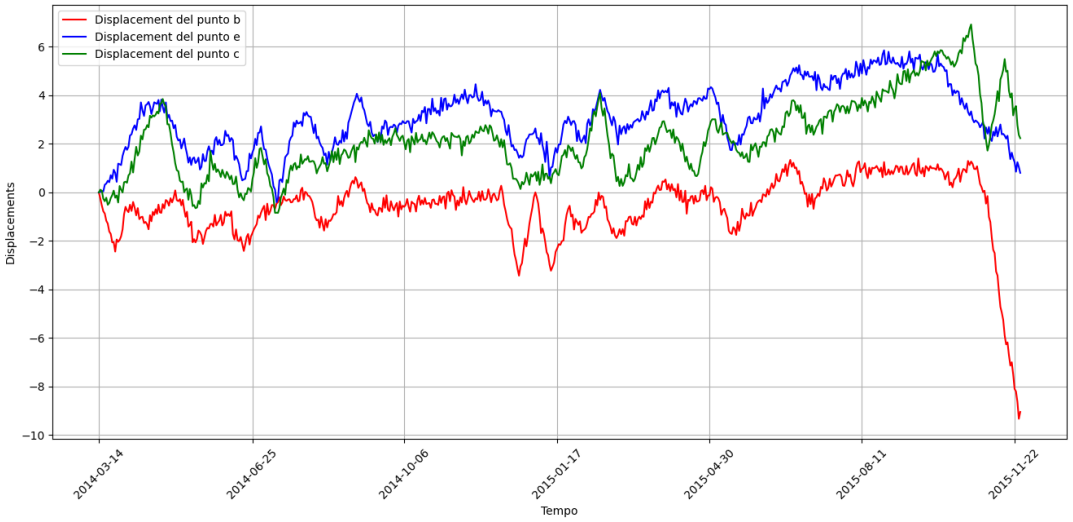

Figura 4.2: Andamento dei displacemnts dei punti b, e e c del ponte di Tadcaster.

Prima di illustrare la definizione e la creazione degli indici di rischio ed i rispettivi incrementi, risulta utile esporre come siano stati ricavati i valori riguardanti la portata del fiume.
Nel paper di riferimento [10], è presente un grafico in cui vengono comparati l'andamento degli spostamenti verticali della porzione crollata del ponte, punto b, e la portata giornaliera del fiume espressa in m^3/sec. I valori relativi a quest'ultima sono stati ricreati mediante il software online gratuito *WebPlotDigitizer* [45], sviluppato per estrarre dati numerici da immagini salvate in differenti formati. Il suo funzionamento si articola in più fasi di cui, la prima, consiste nel caricamento del grafico dal quale si desiderano estrarre i dati. Successivamente, si effettua una definizione e calibrazione degli assi. In questo caso, l'asse delle ascisse è rappresentato dalle date, mentre l'asse delle ordinate descrive i valori della portata. Inoltre, vengono inseriti i punti di riferimento per questi ultimi al fine di permettere al programma di stabilire la corrispondenza tra i pixel ed i valori reali. Attraverso la modalità semi-automatica, il software rileva i pixel di uno specifico colore all'interno di un'area quadrata sull'immagine. La tonalità dei pixel ed il numero di questi ultimi che definiscono i lati di tale spazio sono definiti a priori dall'utente. Infine, individuati i valori, è possibile salvarli in un file in formato CSV avente due colonne nominate *date* e *gdf*, che contengono i valori individuati dal software, rispettivamente le date e la portata.
Successivamente si procede con la trasformazione di quest'ultimo in un dataframe tramite libreria Pandas [46] al fine di effettuare una pulizia dei dati. Infatti, l'operazione effettuata con *WebPlotDigitizer* fornisce più di un valore della portata per singola data, soprattutto nel caso in cui il numero di pixel è impostato basso al fine

di individuare più pixel possibili e, di conseguenza, più valori della portata. Per facilitare la successiva operazione, si procede con un ordinamento cronologico delle date. Infine, si raggruppano i valori della portata in base alla data selezionando, per ciascun giorno, il suo valore massimo. Tale scelta, a discapito del valore medio o minimo, permette di esaltare i massimi mantenendoli fedeli al plot originale ed evita rispettivamente arrotondamenti troppo evidenti o di dare più importanza ai minimi della serie. Il risultato di tale processo viene mostrato in Fig. 4.3.

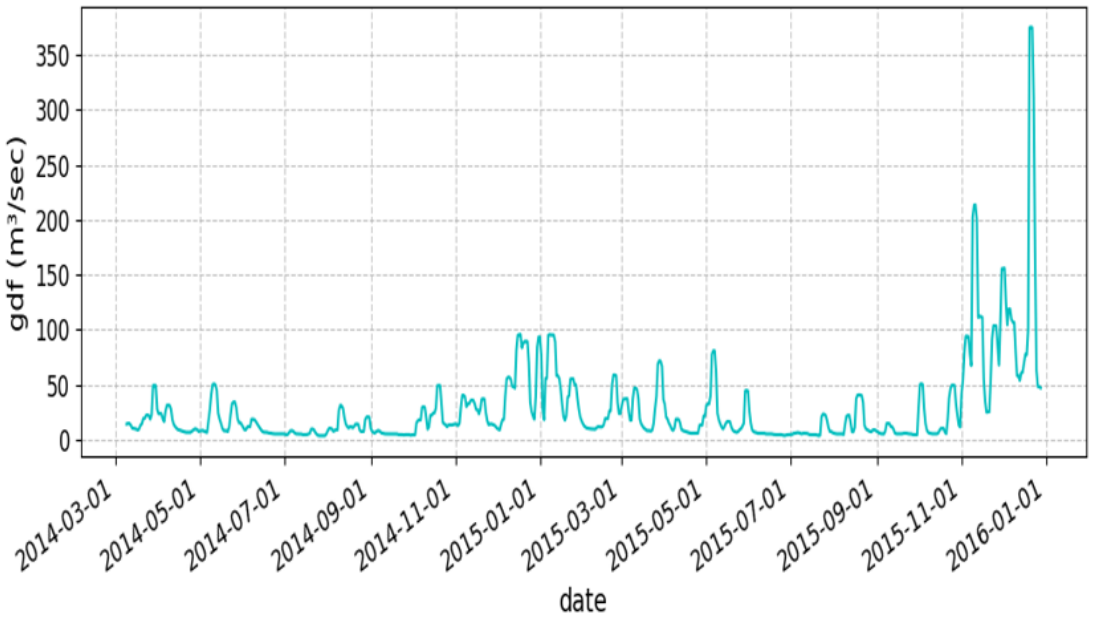

Figura 4.3: Andamento della portata per il fiume Wharfe espressa in m^3/sec.

Si procede ora con l'analizzare la creazione dei cinque differenti indici di rischio con i relativi incrementi presi in esame per il ponte di Tadcaster:

$$\mathrm{R} = \{r_k\} \quad \mathrm{e} \quad \mathrm{Y} = \{y_k\}, \qquad k = 1, ..., 5.$$

In particolare, r_k ha valori compresi nell'intervallo [0,1] mentre il relativo y_k è calcolato come la differenza tra due indici in sequenza. Matematicamente, se $r_{k,i}$ è il valore del rischio al tempo $i-$esimo e $r_{k,i+1}$ rappresenta quello al tempo successivo, l'incremento è definito come segue:

$$y_{k,i} = r_{k,i} - r_{k,i-1} \qquad i = 1, .., T \tag{4.1}$$

dove T è l'istante temporale dell'ultima misurazione e $r_{k,0}$ e $y_{k,0}$ sono entrambi nulli. Tale definizione rende gli incrementi non negativi. Una visione d'insieme di

r_1	y_1	r_2	y_2	r_3	y_3	r_4	y_4	r_5	y_5
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.82e-12	6.82e-12	0.007567	0.007567	0.003783	0.003783	0.006490	0.006490	0.003245	0.003245
1.09e-10	1.02e-10	0.015906	0.008339	0.007953	0.004170	0.013644	0.007153	0.006822	0.003576
5053e-10	4.44e-10	0.028980	0.013075	0.014490	0.006537	0.024859	0.011215	0.012429	0.005608
1.45e-9	1.19e-9	0.040611	0.011630	0.020305	0.005815	0.034835	0.009976	0.017418	0.004988
4.26e-9	2.52e-9	0.053878	0.013267	0.026939	0.006634	0.046215	0.011380	0.023108	0.005690
8.84e-9	4.59e-9	0.068995	0.015117	0.034498	0.007589	0.059182	0.012967	0.029591	0.006484
1.64e-8	7.54e-9	0.083880	0.014886	0.041940	0.007443	0.071951	0.012769	0.035976	0.006384
2.80e-8	1.16e-8	0.101719	0.017839	0.050860	0.008919	0.087253	0.015302	0.043626	0.007651
4.48e-8	1.68e-8	0.119819	0.018099	0.059910	0.009050	0.1027783	0.015525	0.051389	0.007763

Tabella 4.3: Rappresentazione degli indici di rischio e dei loro incrementi nel file *inferred* per il ponte di Tadcaster. Valori delle colonne approssimati alla sesta cifra decimale.

quest'ultimi è disponibile nella Tab. 4.3.
Il primo di essi è r_1 che contiene valori generati applicando la funzione $f(x) = x^4$ alla colonna *step* della Tab. 4.2. La scelta di tale trasformazione deriva dalla volontà di poter analizzare un rischio che non dipende direttamente dai dati sperimentali e che, quindi, è scelto a priori. Il suo andamento, mostrato in Fig. 4.4, risulta utile al fine di rappresentare la progressiva crescita del rischio durante il periodo di osservazione fino al valore massimo, 1, che rappresenta il crollo certo del ponte. Infatti, la funzione è caratterizzata inizialmente da un incremento moderato seguito da una crescita resa più accentuata avvicinandosi agli istanti prima del crollo.
Gli altri indici di rischio, visibili nella Tab. 4.3, sono stati creati seguendo un approccio *data-driven*: per la loro creazione, si sono considerati i dati a disposizione come i displacements e la portata. L'idea alla base è che il superamento di una determinata soglia di quest'ultimi influisse sul rischio causando un aumento di esso proporzionale all'ammontare del superamento della soglia. Si procede, dunque, con il calcolo di tale misura rispettivamente per i displacements Δ dei punti $p = \{a, b, e\}$ e per la portata γ:

$$\eta_p = \hat{\Delta}_p - \sigma_p \qquad \eta_\gamma = \hat{\gamma} + \sigma_\gamma$$

dove $\hat{\Delta}_p$ e $\hat{\gamma}$ rappresentano il valore medio delle misurazioni degli spostamenti verticali e della portata. Matematicamente:

$$\hat{\Delta}_p = \frac{1}{T}\sum_{i=1}^{T} \Delta_{p,i} \qquad \hat{\gamma} = \frac{1}{T}\sum_{i=1}^{T} \gamma_i$$

con $\Delta_{p,i}$ e γ_i che sono rispettivamente il valore del displacament del punto p e la portata rilevati all'istante temporale i-esimo; T definito come sopra. Invece, σ_p e σ_γ sono le deviazioni standard campionarie calcolate per i dispalcements e la

portata come segue:

$$\sigma_p = \sqrt{\frac{\sum_{i=1}^{T}(\Delta_{p,i} - \hat{\Delta}_p)^2}{T-1}} \qquad \sigma_\gamma = \sqrt{\frac{\sum_{i=1}^{T}(\gamma_i - \hat{\gamma})^2}{T-1}}$$

La decisione di creare due soglie differenti per i punti e per la portata deriva dalla relazione tra quest'ultimi e l'incremento dell'indice di rischio. Si considerino i primi: l'aumento del rischio che il ponte crolli è legato ad un movimento verticale e verso il basso di una porzione di ponte insolita rispetto alla tendenza abituale. In particolare, i punti significativi sono tali da avere un displacement con valori tali da essere minori di η_p. Al contrario, una portata rilevante è tale da avere valori superiori a η_γ. Infatti, un aumento dei metri cubi del flusso di acqua, quindi un incremento significativo del livello e della forza de fiume, può causare danni alla struttura del ponte e, dunque, aumentare il rischio di crollo.
Al fine di creare gli indici di rischio, non si considerano direttamente gli spostamenti verticali o la portata, bensì i corrispondenti salti percentuali oltre soglia così definiti

$$\rho_{p,i} = \max\left(\frac{-\Delta_{p,i} + \eta_p}{|\eta_p|},0\right), \qquad p = \{a,b,e\} \;\text{ e }\; i = 1,...,T$$

$$\rho_{\gamma,i} = \max\left(\frac{\gamma_i - \eta_\gamma}{|\eta_\gamma|},0\right), \qquad p = \{a,b,e\} \;\text{ e }\; i = 1,...,T.$$

Inoltre, si desiderano creare indici di rischio basati su due scenari differenti: il primo considera unicamente gli spostamenti verticali, mentre il secondo tiene conto anche della portata. Quindi, si calcola il valore medio per ogni istante temporale dei salti appena introdotti come segue:

$$\rho_{od,i} = \frac{1}{3}\left(\rho_{a,i} + \rho_{b,i} + \rho_{e,i}\right),$$

$$\rho_{all,i} = \frac{1}{4}\left(\rho_{a,i} + \rho_{b,i} + \rho_{e,i} + \rho_{\gamma,i}\right),$$

dove $\rho_{od,i}$ tiene conto solo di Δ, mentre $\rho_{all,i}$ considera Δ e γ.
Facendo riferimento alla Tabella 4.3, r_2 e r_3 sono calcolati basandosi sui displacements. In particolare, il primo indice di rischio è calcolato come la somma cumulata della media dei salti e, successivamente, normalizzata per ottenere valori compresi tra zero e uno:

$$r_{2,i} = \frac{\sum_{j=1}^{i}\rho_{od,j}}{\sum_{j=1}^{T}\rho_{od,j}}, \qquad i,j = 1,...,T.$$

Invece, al fine di definire il secondo, si calcola la somma in ogni istante temporale tra r_1 e r_2, ovvero $\mathrm{r}_{3,i} = r_{1,i} + r_{2,i}$. A tale addizione, si applica la normalizzazione

Min-Max la quale è una trasformazione lineare che consente di ridefinire ogni valore all'interno dell'intervallo [0,1]:

$$r_{3,i} = \frac{\mathrm{r}_{3,i} - \min(\mathrm{r}_{3,i})}{\max(\mathrm{r}_{3,i}) - \min(\mathrm{r}_{3,i})}.$$

Gli incrementi relativi agli indici appena introdotti, y_2 e y_3, sono calcolati mediante la formula 4.1.
I rischi r_4 e r_5, invece, si calcolano seguendo la logica sopraccitata ma considerando la portata insieme ai displacements. In particolare, si ha:

$$r_{4,i} = \frac{\sum_{j=1}^{i} \rho_{all,j}}{\sum_{j=1}^{T} \rho_{all,j}}, \qquad i, j = 1, ..., T$$

$$r_{5,i} = \frac{\mathrm{r}_{5,i} - \min(\mathrm{r}_{5,i})}{\max(\mathrm{r}_{5,i}) - \min(\mathrm{r}_{5,i})}, \quad \text{con} \;\; \mathrm{r}_{5,i} = r_{1,i} + r_{4,i}$$

I corrispondenti incrementi, y_4 e y_5 sono calcolati tramite la formula 4.1. In Fig. 4.4 e Fig. 4.4 è possibile visualizzare il risultato di tali operazioni. A differenza di r_1 che presenta un andamento polinomiale e definito, gli altri indici di rischio mostrano una crescita monotona ma irregolare. In particolare, si possono notare in maniera netta dei periodi temporali in cui r_2 e r_4 hanno rispettivamente $y_2 = y_4 \approx 0$. Tale scenario non si manifesta con r_3 e r_5 grazie all'aggiunta di $r_1 \neq 0$ in ogni istante temporale.

Perturbazione dei file di partenza

I file relativi al ponte di Tadcaster vengono usati come punto di partenza per popolare il training set. In particolare, attraverso tecniche di *data augmentation*, vengono creati ulteriori file a partire da *points* e *inferred*. Tale operazione si articola in più fasi ed ha lo scopo di arricchire i dati di input del modello: in particolare, l'obiettivo è di creare coppie di file che presentino piccole differenze riguardanti i valori delle coordinate e dei displacements ma aventi gli stessi indici di rischio.
Inizialmente, si creano quattro differenti funzioni che rappresentano delle trasformazioni geometriche di riflessione all'interno del quadrato unitario nel quale sono definiti i punti del ponte. Dunque, si vogliono creare delle copie del ponte originale ma con le coordinate orientate spazialmente in modo differente. Tali trasformazioni sono:

- φ_{or}, esegue una riflessione dei punti rispetto all'asse orizzontale $z = 0.5$ ottenendo come nuove coordinate (x_{new}, z_{new}) = (x_{old}, 1 - z_{old}). Si noti che x rimane invariata;
- φ_{vr}, invece, effettua una riflessione rispetto all'asse verticale $x = 0.5$ ricavando (x_{new}, z_{new}) = ($1 - x_{old}$, z_{old}). Si noti che la coordinata z rimane invariata;

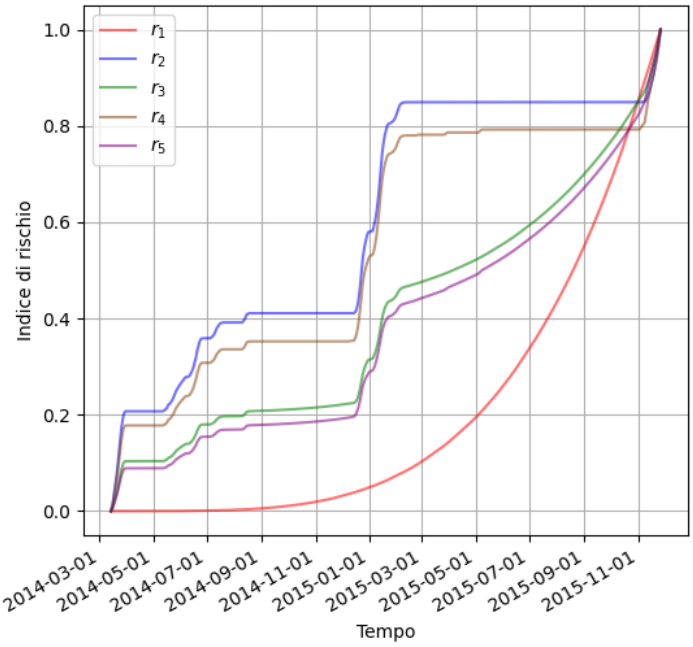

Figura 4.4: Indici di rischio r_k, $k = 1, ..., 5$.

- φ_{d1}, esegue una riflessione rispetto alla diagonale principale del quadrato unitario che ha equazione $z = x$. Quinid, di ottiene $(x_{new}, z_{new}) = (z_{old}, x_{old})$;

- φ_{d2}, al contrario, effettua una riflessione rispetto alla diagonale secondaria del quadrato unitario che ha equazione $z = -x + 1$. Si ottengono le nuove coordinate $(x_{new}, z_{new}) = (1 - z_{old}, 1 - x_{old})$.

Successivamente si creano tutte le possibili combinazioni di tali funzioni e si applicano alle coordinate originali del ponte. Si noti che alcune combinazioni restituiscono un output avente coordinate uguali al ponte di partenza oppure forniscono risultati uguali tra loro. Per esempio, si consideri un punto di riferimento avente coordinate (x_0, z_0). La combinazione di tutte le funzioni genera le coordinate

di partenza, infatti

$$\begin{aligned}
&\varphi_{or} \circ \varphi_{vr} \circ \varphi_{d1} \circ \varphi_{d2}((x_0, y_0)) \\
&= \varphi_{or}(\varphi_{vr}(\varphi_{d1}(\varphi_{d2}((x_0, y_0))))) = \\
&= \varphi_{or}(\varphi_{vr}(\varphi_{d1}((1 - y_0,\, 1 - x_0)))) = \\
&= \varphi_{or}(\varphi_{vr}((1 - x_0,\, 1 - y_0))) = \\
&= \varphi_{or}((x_0,\, 1 - y_0)) = \\
&= (x_0,\, y_0).
\end{aligned}$$

O ancora, la riflessione orizzontale combinata con quella rispetto alla diagonale principale fornisce lo stesso risultato di quest'ultima combinata con la riflessione verticale. Infatti, si ha

$$\begin{aligned}
&\varphi_{or} \circ \varphi_{d1}((x_0, z_0)) = \\
&= \varphi_{or}(\varphi_{d1}((x_0, z_0))) = \\
&= \varphi_{or}((z_0, x_0)) = \\
&= (z_0, 1 - x_0) = \\
&= \varphi_{d1}((1 - x_0, z_0)) = \\
&= \varphi_{d1}(\varphi_{vr}((x_0, z_0))) = \\
&= \varphi_{d1} \circ \varphi_{vr}((x_0, z_0)).
\end{aligned}$$

Al fine di evitare l'inserimento nel training set di ponti prodotti dalle combinazioni aventi le stesse coordinate, si effettua un controllo sul nuovo ponte. Se quest'ultimo rientra in uno dei due scenari appena descritti, il ponte viene scartato e non viene inserito nel set per l'addestramento.
Nella creazione dell'input del modello si applica la PCA alla posizione dei punti al fine di orientare il ponte. Attraverso delle verifiche sperimentali, si nota che applicando tale processo ai ponti appena creati si producono ulteriori duplicati. Per evitare ciò si aggiunge una piccola quantità di rumore alle coordinate: in particolare, si applica un *white noise*, o *rumore bianco*, a x e z. Matematicamente, si consideri un vettore di variabili casuali indipendenti $\upsilon = (\upsilon_1, ..., \upsilon_n)$, $n = 7$ numero totale dei punti sul ponte, il quale ha distribuzione normale:

$$\upsilon_i \sim \mathcal{N}(\mu_\upsilon, \sigma_\upsilon^2)$$

con media $\mu_\upsilon = 0$ e deviazione standard $\sigma_\upsilon^2 = \frac{0.05}{3}$. Tali valori sono scelti per avere una minima perturbazione dei punti al fine di avere un risultato simile come struttura e forma al ponte originale. Infine, il vettore υ viene sommato a x e z. Nella Fig. 4.5 è possibile visualizzare un confronto tra il ponte oirginale, in alto a sinistra, e quelli creati mediante il processo di riflessione e perturbazione

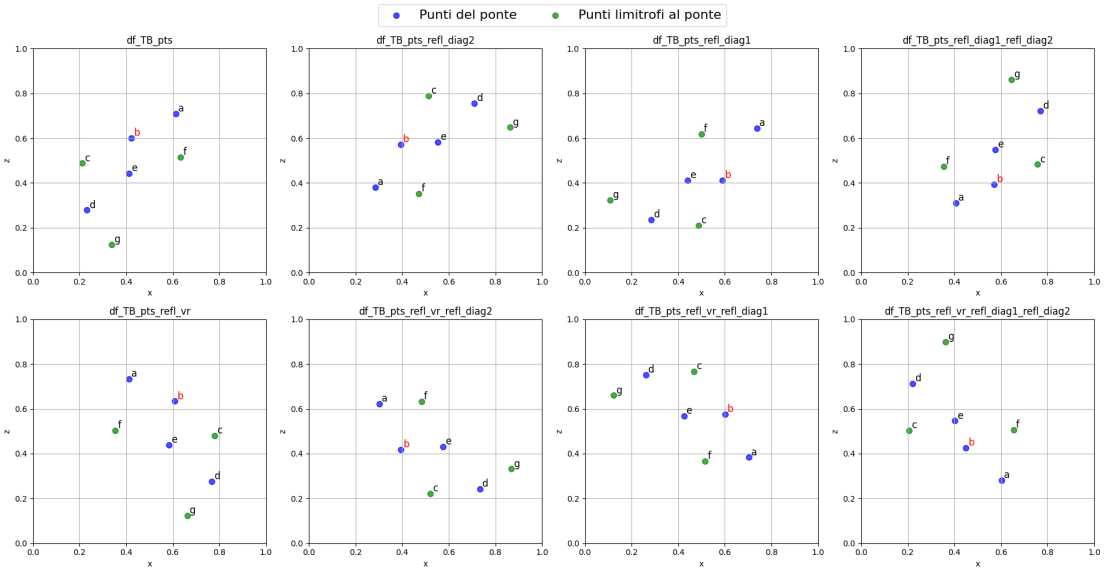

Figura 4.5: Coordinate del ponte originale, in alto a sinistra, e coordinate dei ponti dopo le riflessioni e le perturbazioni.

appena descritto. Si noti che il titolo di ogni sotto-figura è definito dalle riflessioni combinate per crearlo.
Ad ognuno degli sette ponti creati fino a questo momento ed al ponte originale, si applica tre volte ogni funzione dell'insieme F = $\{f_{new}, f_\epsilon, f_\nu\}$. Al completamento di tale operazione, si avranno a disposizione per il training set un totale di 80 ponti, di cui 72 creati e 8 preesistenti. Di seguito, i dettagli di tali trasformazioni.
La funzione f_{new} permette di aggiungere un nuovo punto p_{new} sul ponte. Inizialmente, si crea l'inviluppo convesso definito dai punti del ponte $\mathcal{P} = \{p_1, ..., p_4\} = \{a, b, d, e\}$:

$$Conv(\mathcal{P}) = \left\{ \sum_{i=1}^{4} \phi_i p_i : \phi_i \geq 0, \sum_{i=1}^{4} \phi_i = 1 \right\}.$$

Ciò implica che Conv($\mathcal{P}$) è l'insieme convesso più piccolo che contiene tutti i punti appartenenti a $\mathcal{P}$. Al suo interno viene costruita la triangolazione di Delaunay al fine di suddividere in sezioni l'inviluppo. In particolare, essa può essere definita come $\mathcal{T} = \{T_1, ..., T_h\}$ con $i = 1, ..., h$ e $T_i \in \mathcal{P}$. Si noti che l'unione dei singoli triangoli copre interamente l'inviluppo convesso: infatti,

$$\bigcup_{i=1}^{h} T_i = \text{Conv}(\mathcal{P}).$$

Inoltre, due triangoli distinti, $T_j \neq T_k$, non presentano sovrapposizioni.
Successivamente, si crea un area rettangolare così definita

$$\mathcal{A} = [x_{min}, x_{max}] \times [z_{min}, z_{max}]$$

dove (x_{max}, z_{max}) e (x_{min}, z_{min}) sono i valori massimi e minimi delle coordinate dei punti in $\mathcal{P}$. Si generano casualmente le coordinate di p_{new} sfruttando una distribuzione uniforme definita per x e z:

$$x_{new} \sim \mathcal{U}(x_{min}, x_{max}) \qquad z_{new} \sim \mathcal{U}(z_{min}, z_{max}).$$

Tali valori vengono accettati come coordinate del nuovo punto se

$$(x_{new}, z_{new}) \in \mathcal{A} \cap T_i$$

dove $T_i \in \mathcal{T}$, $i = 1, ..., h$. In Fig. 4.6 è possibile visualizzare i principali passi del processo appena descritto.
Un ulteriore controllo che viene effettuato è che p_{new} sia abbastanza distante dai punti preesistenti. Si calcola quindi la distanza euclidea tra le coordinate di questi ultimi e quelle del nuovo punto: se tale misura è minore di una soglia fissata, p_{new} non viene accettato e si ripete il processo con la generazione casuale di altre coordinate.
L'ultimo passo consiste nel costruire i dispacements del nuovo punto, un valore per ogni giorno del periodo temporale considerato e aggiungerli nella tabella relativa al file *inferred*. Si considerano l'insieme di punti sul ponte $\mathcal{P}$, i relativi displacements Δ_p con $p \in \mathcal{P}$ e il nuovo punto p_{new}. Inizialmente, si calcolano le distanze euclidee δ_p tra p_{new} e i punti in $\mathcal{P}$ e, per ognuna di esse, si definisce un peso inverso alla distanza

$$\hat{\delta}_p = \frac{\min(\delta_p)}{\delta_p}.$$

Tale definizione permette di assegnare maggiore importanza ai punti più vicini penalizzando quelli più distanti da p_{new}. Infatti, viene dato un peso unitario al punto in $\mathcal{P}$ che presenta distanza minima e dei pesi inferiori proporzionali alla distanza da p_{new}. Si calcolano i nuovi displacements tramite una media pesata:

$$\Delta_{new,i} = \frac{\sum_{p \in \mathcal{P}} \hat{\delta}_j \cdot \Delta_{p,i}}{\sum_{p \in \mathcal{P}} \hat{\delta}_j} \quad \text{con} \quad i = 1, ..., T.$$

Mediante funzione f_ϵ, si ha lo scopo di perturbare i displacements scalando i valori giornalieri rispetto a un fattore moltiplicativo. Quindi, si definisce un vettore casuale normale

$$\epsilon = (\epsilon_1, .., \epsilon_T), \qquad \epsilon_i \sim \mathcal{N}(\mu_\epsilon, \sigma_\epsilon^2) \quad \text{con} \quad i = 1, ..., T$$

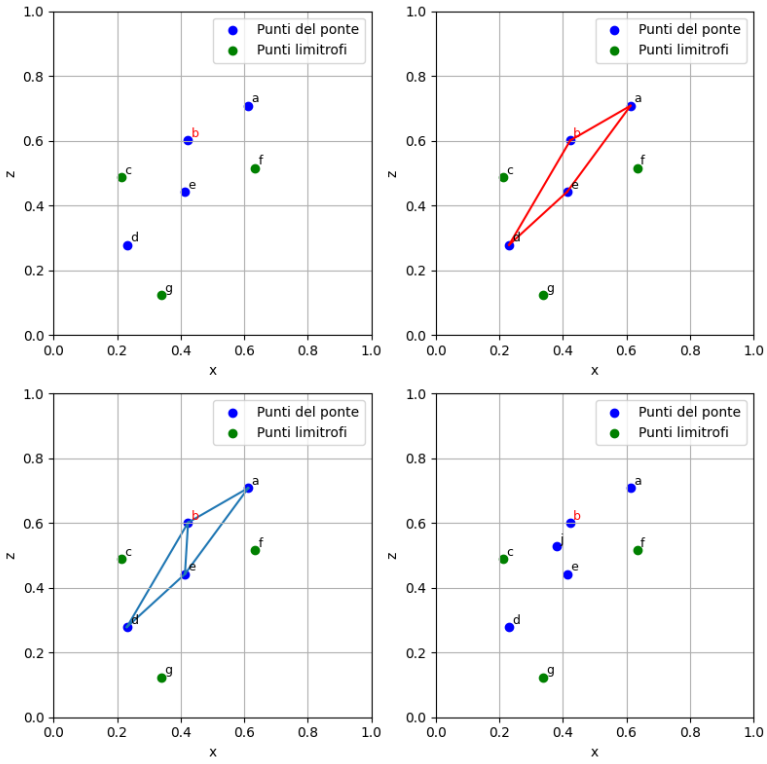

Figura 4.6: Principali step della creazione del nuovo punto mediante la funzione f_{new}. In alto a sinistra, il ponte di partenza; in alto a destra, l'inviluppo convesso disegnato in rosso; in basso a sinistra, la triangolazione di Delaunay colorata in azzurro e costruita sull'inviluppo convesso; in basso a destra, il ponte con il nuovo punto j.

dove T è l'ultimo istante temporale, $\mu_\epsilon = 1$ è la media e $\sigma_\epsilon^2 = \frac{0.1}{3}$ è la deviazione standard. Tali valori sono selezionati per la natura moltiplicativa e non additiva di ϵ e, inoltre, hanno l'obiettivo di mantenere la forma e l'andamento dei displacements successivi alla trasformazione fedele all'originale.
Si considerano solamente le colonne contenenti gli spostamenti verticali del file *inferred* e si applica ad esse la trasformazione

$$\Delta_{j,i}^{new} = \Delta_{j,i} \cdot \epsilon_i \quad \text{con} \quad j = 1, ..., n \quad \text{e} \quad i = 1, ..., T$$

dove n è il totale dei punti considerati.
Infine, la funzione f_ν permette di applicare del *white noise* ai displacements. Quindi, si definisce un vettore di variabili casuali $\nu = (\nu_1, ..., \nu_T)$ che segue una distribuzione normale. Infatti,

$$\nu_i \sim \mathcal{N}(\mu_\nu, \sigma_\nu^2) \quad \text{con} \quad i = 1, ..., T$$

dove i valori della media $\mu_\nu = 0$ e $\sigma_\nu^2 = \frac{1}{3}$ sono scelti per la natura additiva della funzione e hanno lo scopo di non snaturare l'andamento originale degli spostamenti verticali. I nuovi displacments sono definiti come segue:

$$\Delta_{j,i}^{new} = \Delta_{j,i} + \nu_i \quad \text{con} \quad j = 1, ..., n \quad \text{e} \quad i = 1, ..., T$$

dove n è definito come sopra.
In Fig. 4.7 si visualizza uno zoom dell'andamento dei displacements del punto b prima e dopo l'applicazione delle funzioni f_ϵ e f_ν. La curva rossa rappresenta i Δ precedenti alle trasformazioni, mentre la curva viola e quella verde identificano i Δ scalati rispettivamente mediante f_ϵ e f_ν. Si noti che i nuovi andamenti sono simili a quello originale. Però, f_ν introduce una maggiore variabilità visibile con differenze più importanti. Al contrario, f_ϵ produce una differenza minima.

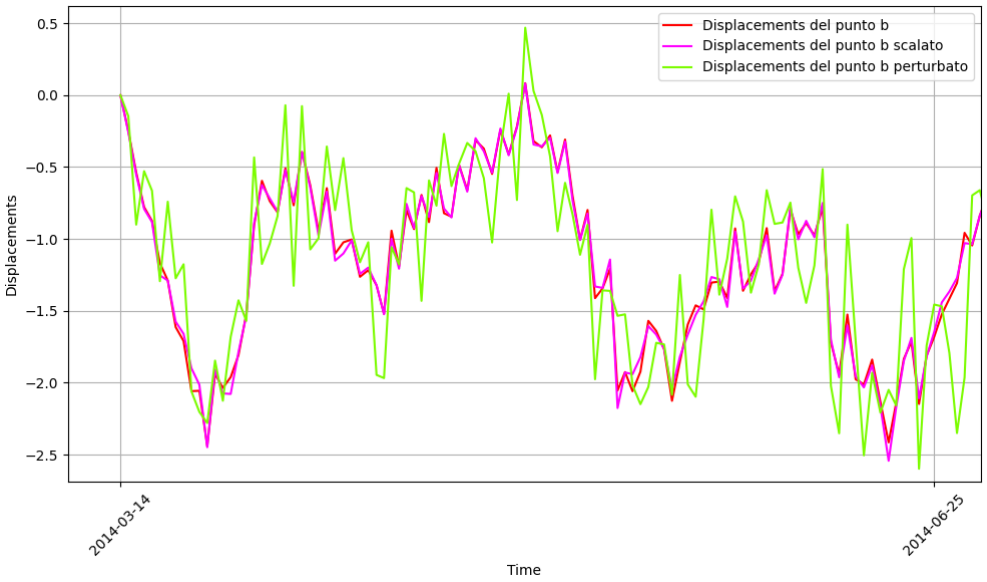

Figura 4.7: Esempio di applicazione delle funzioni f_ϵ e f_ν a Δ_b.

Alla fine di tale processo, i nuovi ponti creati vengono aggiunti al training set che è popolato da strutture con 7 e 8 punti. Poichè il ponte di Cantiano, usato unicamente come test, presenta 9 punti, si vuole allineare l'insieme di addestramento a tale quantità. Perciò, si applica un'ultima volta la funzione f_{new} a tutti i ponti in modo da avere anche file con 9 punti all'interno del set di addestramento. Quindi, si aggiungono ulteriori 80 file per un totale di 160.

4.1.2 Ponte di Cantiano

Al fine di valutare le performance del modello su un ponte diverso da quello di Tadcaster, si considera il ponte del comune di Cantiano. I dati a disposizione presentano un formato e una struttura differente rispetto a quelli che il modello riceve in input. Di conseguenza, si effettua un processo di *data trasformation* al fine di uniformare i dati.
I file di partenza sono *C7_polygon* e *Cantiano_Asc*. Il primo è in formato KML, ovvero Keyhole Markup Language, il quale viene usato per rappresentare dati geospaziali su mappe. Esso contiene una lista di coordinate geografiche che descrivono il poligono P rappresentante il ponte visto dall'alto. Il secondo, invece, contiene delle caratteristiche relative a punti del ponte e limitrofi, come ID, latitudine, longitudine o i displacements divisi per date.
Il primo step consiste nel classificare ogni punto presente in *Cantiano_Asc* sfruttando il poligono proveniente dal file *C7_polygon*: gli elementi che si trovano all'interno del poligono sono etichettati come punti del ponte mentre i punti oltre il poligono sono catalogati come esterni al ponte. In particolare, i primi sono sei e sono visibili in blu nella Fig. 4.8 mentre i rispettivi valori sono presenti nelle prime sei righe della Tab. 4.4. Si noti che, a differenza di Tadcaster, la colonna *point* contiene un valore numerico come identificativo del punto. Tale disparità è trascurabile ai fini del modello in quanto la colonna *point* non viene inserita nell'input.
Dato l'elevato numero di punti collocati all'esterno del ponte, si procede con la selezione di solo una parte di essi considerando quelli più vicini al ponte. In particolare, si calcolano le coordinate $(x_{\mathcal{C}}, z_{\mathcal{C}})$ del centroide $\mathcal{C}$ tramite libreria Shapely [47] ottenendo:

$$\mathcal{C} = (x_{\mathcal{C}}, z_{\mathcal{C}}) = \frac{1}{A} \int \int_P (x_P, z_P) dA$$

dove A è l'area del poligono P e (x_P, z_P) sono le coordinate del poligono in due dimensioni. Successivamente, per calcolare le distanza tra il centroide ed i punti esterni al ponte, si utilizza la formula dell'Hanverseno invece della distanza euclidea. Per fare ciò, si ipotizzi un generico punto esterno al ponte con coordinate (x_j, z_j) e si trasformano quest'ultime in radianti

$$x_j^* = x_j \cdot \frac{\pi}{180}, \qquad z_j^* = z_j \cdot \frac{\pi}{180}.$$

Successivamente, si calcola un parametro c tale che

$$c = 2 \cdot \text{arctan2}(\sqrt{a}, \sqrt{1-a})$$

dove arctan2 è una variazione dell'arcotangente avente però due input. Il valore a è definito come

$$a = \sin^2\left(\frac{dz}{2}\right) + \cos(z_{\mathcal{C}}^*) \cdot \cos(z_j^*) \cdot \sin^2\left(\frac{dx}{2}\right)$$

dove $dx = x_j^* - x_{\mathcal{C}}^*$ e $dz = z_j^* - z_{\mathcal{C}}^*$ sono le distanze tra le coordinate del punto considerato e quelle del centroide. Infine, la distanza è calcolata come

$$\mathrm{d} = R_t \cdot c$$

con $R_t = 6371000$ metri, che rappresenta il raggio della terra. Tale procedura viene applicata a tutti i punti esterni al ponte. Avendo ora a disposizione tali misurazioni, è possibile selezionare tra questi ultimi i più vicini ad esso tali che $d < 15$ metri. Si procede con un'ulteriore spoglio: vengono considerati i due punti tali da avere il più ampio range tra i displacemnts massimo e minimo. Essi sono rappresentati in verde nella Fig. 4.8 mentre i relativi valori numerici sono visibili nella Tabella 4.4, nelle ultime due righe.

point	*b*	*x*	*z*
8180	1	0.361671	0.526855
8181	1	0.361628	0.526469
8240	1	0.362743	0.528560
8281	1	0.363987	0.531587
8329	1	0.364931	0.532247
8330	1	0.364931	0.532192
fake_left	1	0.357789	0.521086
7953	0	0.356738	0.522232
8131	0	0.360856	0.527240

Tabella 4.4: Rappresentazione del file *points.csv* del ponte di Cantiano. Valori delle colonne x e z approssimati alla sesta cifra decimale.

In quest'ultima, è possibile vedere un ulteriore elemento chiamato "fake_left". Esso è stato scelto in modo casuale nella parte sinistra del ponte. In particolare, si consideri l'area che costituisce il ponte di Cantiano A ed i valori massimi e minimi delle coordinate di quest'ultima, ovvero (x_{min}, x_{max}) e (z_{min}, z_{max}). Inotre, si introduca una coordinata $\tilde{x}$ che si trova nella parte sinistra del ponte definita come

$$\tilde{x} = x_{min} + \frac{x_{max} - x_{min}}{4}.$$

É possibile utilizzare tale formula in quanto, attraverso un plot di prova, è emerso che x_{min} è collocato nella porzione di sinistra. In seguito, le coordinate di fake_left, (x_f, z_f), vengono generate casualmente sfruttando una distribuzione uniforme

$$x_f \sim \mathcal{U}(x_{min}, \tilde{x}) \qquad z_f \sim \mathcal{U}(z_{min}, z_{max}).$$

Tali proposte vengono accettate come coordinate nel nuovo punto se

$$(x_f, z_f) \in \mathcal{A}_P \cap ([x_{min}, \tilde{x}] \times [z_{min}, z_{max}]).$$

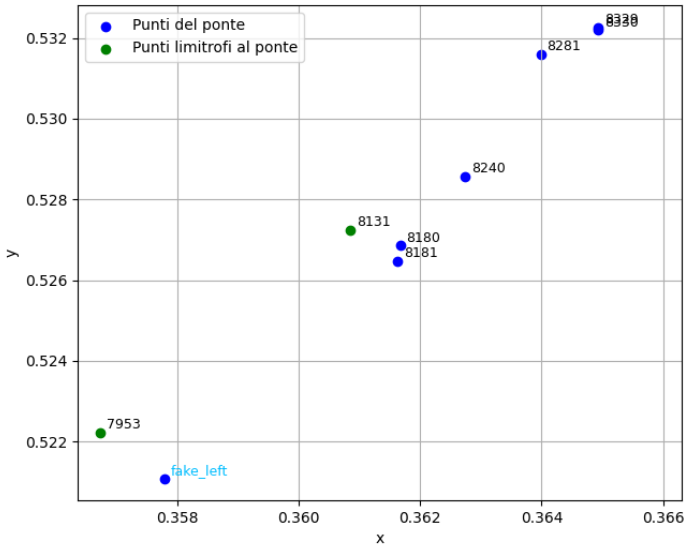

Figura 4.8: Plot dei punti riferiti al ponte di Cantiano. Si noti che i punti 8329 e 8330 sono vicini a tal punto da risultare visivamente sovrapposti.

La procedura appena descritta viene applicata al fine di evitare una problematica emersa con l'applicazione della PCA e della scalatura introdotta nel Cap. 3. In Fig. 4.9 sono riportati i risultati di tali trasformazioni in due casi: a sinistra, vi è il ponte senza fake_left, mentre a destra, viene incluso il punto fake_left. Si consideri il primo: l'orientamento ottenuto è influenzato dalla distribuzione dei punti del ponte in quanto quest'ultimi si trovano principalmente nella porzione di destra. Di conseguenza, i punti esterni al ponte risulteranno più distanti di quanto non lo siano in realtà. Con l'aggiunta di fake_left, i punti esterni vengono calibrati maggiormente consentendo un orientamento più simmetrico: infatti, è possibile notare una disposizione spaziale più fedele all'originale nel plot a destra. Tale correzione è positiva riguardo la coerenza geometrica e, quindi, consente di avere performance migliori da parte dei modelli.
Una volta definiti i punti, si assegnano i relativi valori della colonna b come fatto per Tadcaster nella sezione precedente. Infine, si normalizzano le coordinate all'interno del quadrato unitario. Il risultato è visibile nella Fig. 4.8.
Successivamente, si procede con la creazione del file *inferred*, il cui risultato è visibile nella Tabella 4.5 per i displacements e nella Tabella 4.6 per gli indici di rischio R $= \{r_k\}$ ed i relativi incrementi Y $= \{y_k\}$, con $k = 1, ..., 3$.
Inizialmente, si inseriscono i valori dei dispalcements relativi ai punti selezionati che si trovano nel file *Cantiano_Asc*. Si noti che, in questo caso, il periodo coperto dalle

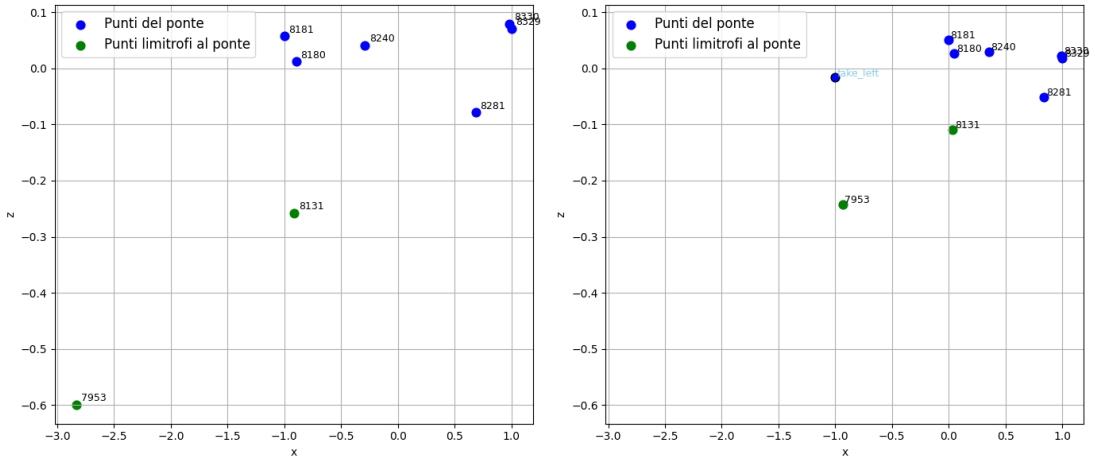

Figura 4.9: Confronto visivo dell'applicazione della PCA al ponte di Cantiano senza il punto fake_left, plot a sinistra, e con il punto fake_left, a destra, in due quadrantia venti le stesse dimensioni.

date	*step*	8180	8181	8240	8281	8329	8330	7953	8131	fake_left
2011-06-02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2011-06-03	0.000228	0.057216	-0.264718	0.064190	0.270566	-0.033159	-0.331172	-0.018007	0.023963	-0.046957
2011-06-04	0.000456	-0.140517	-0.509331	0.05457	0.288863	-0.156777	-0.355530	-0.168537	0.053359	-0.166026
2011-06-05	0.000684	-0.234778	-0.803212	-0.268707	0.527851	-0.148587	-0.625101	-0.250208	0.021856	-0.313344
2011-06-06	0.000912	-0.190693	-1.235023	-0.221306	0.662570	-0.183130	-0.760711	-0.390984	-0.130133	-0.400105
2011-06-07	0.001141	-0.431639	-1.520209	-0.108072	0.850705	-0.368708	-1.080710	-0.780173	0.000994	-0.537600
2011-06-08	0.001369	-0.404808	-1.814557	-0.389507	0.876275	-0.140854	-1.254426	-0.765254	-0.090303	-0.640612
2011-06-09	0.001597	-0.570406	-1.975707	-0.362667	0.993249	-0.253670	-1.357840	-1.002546	-0.382356	-0.721808
2011-06-10	0.001825	-0.526191	-2.364655	-0.267563	1.262215	-0.322762	-1.712558	-1.016746	0.1786372	-0.802897
2011-06-11	0.002053	-0.590205	-2.6468515	-0.334322	1.450374	-0.423271	-1.856454	-1.232369	-0.265823	-0.900402

Tabella 4.5: Rappresentazione delle prime dieci righe della prima parte del file *inferred* del ponte di Cantiano. Valori delle colonne approssimati alla sesta cifra decimale.

misurazioni è superiore rispetto a Tadcaster: infatti, la data del primo dispacement è il 02/06/2011 mentre l'ultima è il 22/10/2022, un periodo di 4159 giorni, rispetto ai 653 di Tadcaster. Inoltre, le date non sono sequenziali come nel caso di Tadcaster. Per esempio, tra la prima data e la seconda, 04/07/2011, vi è un periodo temporale di 32 giorni. Si procede, dunque, con la creazione delle date mancanti e con l'inserimento dei relativi dispacements nel seguente modo. Siano $\mathcal{D} = \{t_1, ..., t_{T^*}\}$ e $\mathcal{V} = \{v_1, ..., v_{T^*}\}$ rispettivamente l'insieme delle date iniziali e quello dei relativi displacements, con T^* riferito all'ultimo istante temporale. Per ogni data mancante $t \in (t_i, t_{i+1})$, il relativo spostamento verticale è definito come l'interpolazione lineare

r_1	y_1	r_2	y_2	r_3	y_3
0.0	0.0	0.0	0.0	0.0	0.0
2.71e-15	2.71e-15	0.000107	0.000107	5.35e-05	5.35e-05
4.33e-14	4.06e-14	0.000213	0.000106	0.000106	5.29e-05
2.19e-13	1.76e-13	0.000304	9.11e-05	0.000152	4.56e-05
6.93e-13	4.74e-13	0.000387	8.29e-05	0.000193	4.14e-05
1.69e-12	9.99e-13	0.000458	7.13e-05	0.000229	3.57e-05
3.51e-12	1.82e-12	0.000528	6.98e-05	0.000264	3.49e-05
6.50e-12	2.99e-12	0.000590	6.26e-05	0.000295	3.13e-05
1.11e-11	4.59e-12	0.000665	7.48e-05	0.000333	3.74e-05
1.78e-11	6.67e-12	0.000741	7.61e-05	0.000371	3.80e-05

Tabella 4.6: Rappresentazione degli indici di rischio e dei loro incrementi nel file inferred per il ponte di Cantiano. Valori delle colonne approssimati alla sesta cifra decimale.

sfruttando la formula della retta passante tra due punti, v_i e v_{i+1}:

$$\hat{v}_t = v_i + \frac{t - t_i}{t_{i+1} - t_i}(v_{i+1} - v_i), \quad \text{con} \quad t_i < t < t_{i+1}.$$

Al fine di simulare la variabilità delle misurazioni, si introduce del rumore

$$\varepsilon_t \sim \mathcal{N}(0, \sigma^2 I_{T^*})$$

che si aggiunge a $\hat{v}_t$ ottenendo $\tilde{v}_t = \hat{v}_t + \varepsilon(t)$. In Fig. 4.10 è il risultato di tale processo.
Si noti che l'ultima data a disposizione, 22 ottobre 2022, non coincide con la data dell'alluvione, 15 settembre 2022, a seguito della quale si è verificato il crollo del ponte. Perciò, si procede considerando solo le date e le relative misurazioni fino all'alluvione.
Inoltre, si noti che il punto fake_left, essendo stato creato manualmente, non presenta dei valori di displaceemnts preesistenti all'interpolazione lineare. Quindi, prima di applicare la procedura appena descritta, si procede con la definizione manuale dei displacements seguendo la logica introdotta per il ponte di Tadcaster nel paragrafo precedente. Si utilizza, dunque, una media pesata sfruttando l'inverso delle distanze tra il punto fake_left e gli altri elementi del ponte di Cantiano.
Successivamente si creano gli indici di rischio e i relativi incrementi mediante la procedura spiegata nel paragrafo precedente. Per questa struttura, non si hanno a disposizione i dati relativi al flusso del fiume, quindi si definisco solamente tre indici. In particolare, nella Fig. 4.11 si ha r_1, rischio scelto a priori come per Tadcaster. Invece, nella Fig. 4.11, è possibile visualizzare r_2, che tiene conto unicamente dei salti relativi dei displacements oltre soglia e r_3 che è costruito a partire da r_1 e r_2. Si noti che r_2 presenta poche porzioni con incremento nullo, di conseguenza, l'aggiunta di r_1 per definire r_3, non risulta così evidente.

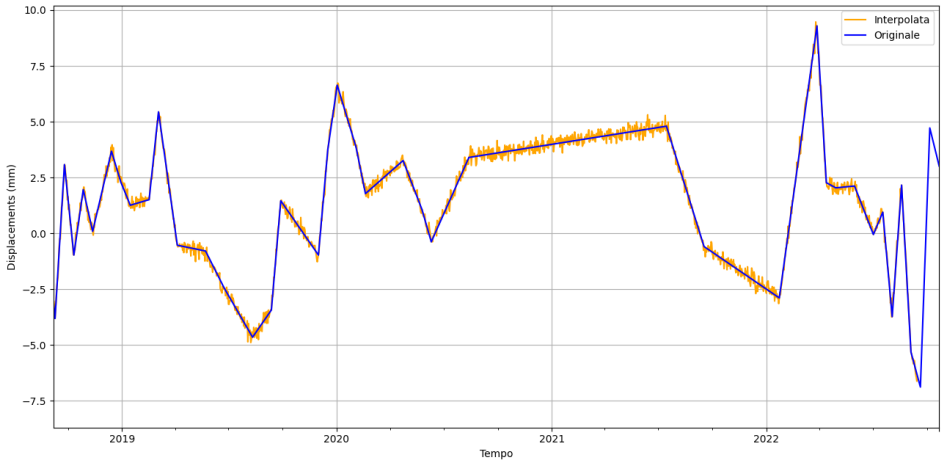

Figura 4.10: Zoom del confronto tra i displacements originali e quelli successivi al processo di interpolazione.

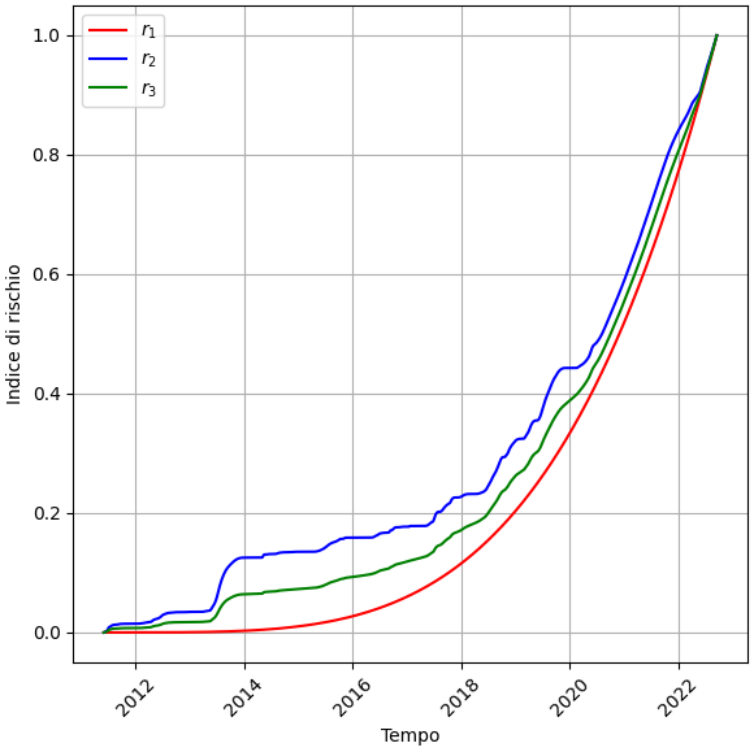

Figura 4.11: Indice di rischio r_k, $k = 1, ..., 3$.

4.2 Architettura dei modelli

Di seguito viene illustrata del dettaglio l'architettura dei modelli formalizzati nel Cap. 3. In quest'ultimo, entrambi i modelli sono definiti come funzioni parametriche che ricevono in input tensori aventi come prima dimensione m corrispondente al numero di finestre temporali. Dal punto di vista operativo, l'architettura è progettata per elaborare una singola finestra temporale alla volta composta da n punti e d giorni. Quindi, l'analisi che segue è illustrata considerando una j-esima finestra temporale. Alla luce di ciò, nelle immagini che seguono la prima dimensione è identificata come *None*. Si noti che i plot sono creati tramite il comando *plot_model* disponibile nella libreria Tensorflow [27].
In generale, l'architettura si articola principalmente in tre componenti comuni sia a GeoDispNet sia a HydroGeoDispNet. Inizialmente vi è uno step di *Pre-processing* al fine di assegnare all'input un ordinamento preciso, uguale ed indipendentemente dai valori dei dati e di prepararlo per il modello. Successivamente, vi è il cuore dell'architettura costituita da blocchi ripetuti che contengono un meccanismo di attenzione. Tali elementi consentono ad ogni punto del ponte di interagire con tutti gli altri. Infine, si effettua un'aggregazione degli elementi elaborati per creare la predizione.

4.2.1 GeoDispNet

Si consideri ora il primo modello. Come è possibile visualizzare nella Fig. 4.12, si effettua un riordinamento stabile dei displacements, $X \in \mathbb{R}^{n\times d}$, e delle coordinate, $C \in \mathbb{R}^{n\times 3}$, mediante *SortAndReorder Layer*. Si definisce la permutazione π, tale che ordina le coordinate prima per y e, in secondo luogo, per x se $z_i = z_j$ con $i, j \in \mathcal{P}$. Quindi, si ha

$$X^\pi = P_\pi X, \quad C^\pi = P_\pi C$$

dove P_π è l'operatore di π. Tale operazione è introdotta in quanto i punti del ponte non presentano un ordinamento naturale e costante: infatti, tramite π, essi sono successivamente disposti sempre nello stesso modo rendendo l'input del modello indipendente rispetto all'ordine originale del ponte. A beneficiare di ciò è anche la MHA, approfondita in seguito. In tale blocco, la matrice di attenzione riflette meglio la struttura geometrica in quanto elementi vicini rimangono vicini anche nella rappresentazione matriciale. Inoltre, il modello non deve apprendere più volte la stessa relazione in contesti differenti, azione necessaria se i punti del ponte fossero presenti in ordini diversi nei batch. Successivamente, le coordinate C sono passate ad un *Dense Layer* con d unità:

$$\tilde{P} = \text{Dense}(C^\pi) = \sigma(C^\pi W_{\tilde{P}} + \hat{b}_{\tilde{P}}) \in \mathbb{R}^{n\times d}$$

con σ funzione di attivazione, $W_{\tilde{P}}$ matrice dei pesi e $\hat{b}_{\tilde{P}}$ vettore di bias. Tale operazione rappresenta un encoding delle coordinate: infatti, si proietta (x, y, b) in uno spazio latente della dimensione dei dati temporali al fine di renderli confrontabili con X^{π}. Di conseguenza, è possibile concatenare i due input tramite il *Multiply Layer* che esegue una moltiplicazione elemento per elemento:

$$Z_0 = X^{\pi} \odot \tilde{C}, \qquad Z_0 \in \mathbb{R}^{n \times d}.$$

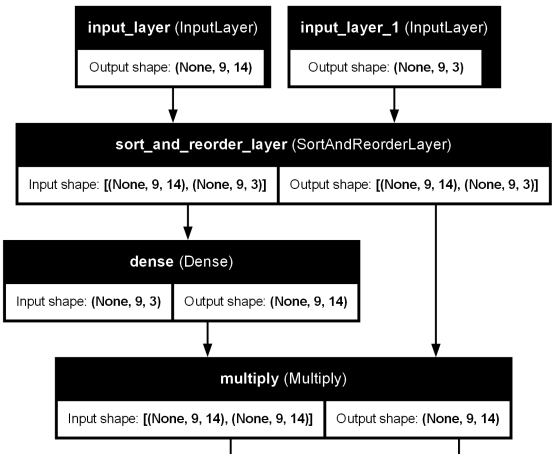

Figura 4.12: Layers che gestiscono l'input di GeoDispNet prima di passare ai blocchi sequenziali. input_layer e input_layer_1 sono rispettivamente X e C.

Nella Fig. 4.13, si visualizza Z_0 dato come input ad un blocco composto da più layer differenti e ripetuto sequenzialmente B volte. Considerando l'i-esimo blocco, con $i = 1, ..., B$, il primo strato è un *Multi-Head Attention Layer*, dettagliato nel Cap. 2. Formalmente, si ha:

$$H_i = \mathrm{MHA}(Z_{i-1}), \qquad H_i \in \mathbb{R}^{n \times d}.$$

dove i token sono rappresentati dagli n punti, mentre le features sono i d giorni. Tale layer permette di modellare dipendenze tra i punti del ponte difficilmente individuabili con strati più semplici, per esempio i Dense Layer. In seguito, H_i viene normalizzato mediante un *Normalization Layer* al fine di rendere l'ottimizzazione più stabile evitando esplosioni del gradiente. Inoltre, si applica un *Residual Layer* al fine di sommare l'informazione originale a quella elaborata. Quindi si ha:

$$\tilde{Z}_i = \mathrm{LayerNorm}(H_i) + Z_{i-1}.$$

Si applicano due *Dense Layer* sequenziali con due unità differenti, $d_1 > d$ e $d_2 = d$

$$F_i = \mathrm{Dense}_1(\tilde{Z}_i), \qquad E_i = \mathrm{Dense}_2(F_i)$$

e, infine, si esegue nuovamente la coppia *Normalization Layer* e *Residual Layer*, indicato con *add* nelle immagini, ottenendo

$$Z_i = \text{LayerNorm}(E_i) + \tilde{Z}_i.$$

Considerando il codice sviluppato, successivamente ai due *Dense Layers* sequenziali, vi è un *Drop Out Layer* opzionale. A seguito di *Random Search* ed esperimenti su un dataset di dimensione ridotta, si è osservato che le performance risultano migliori senza l'utilizzo di tale layer.

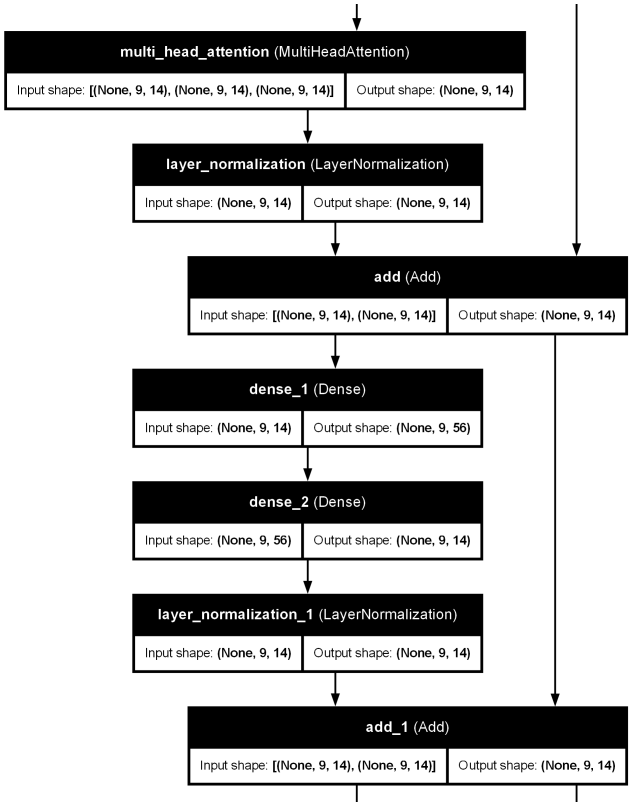

Figura 4.13: Blocco ripetuto in sequenza B volte in GeoDispNet.

Infine, si procede con l'ultimo step visualizzabile nella Fig. 4.14: si effettua una somma aggregata, *Reduce and Sum Layer*, al fine di concatenare l'informazione proveniente da tutti i punti in un singolo vettore $S \in \mathbb{R}^d$:

$$S = \sum_{p=1}^{n} Z_B[p, :]$$

dove Z_B rappresenta l'output dell'ultimo blocco. Tale procedura, permette di ottenere una rappresentazione globale che rispecchia la struttura complessiva del

ponte. Infine, si applicano due *Dense Layers* sequenziali, l'ultimo dei quali ha l'obiettivo di proiettare S nello spazio dell'output, $\mathbb{R}^d$:

$$\tilde{S} = \text{Dense}_{13}(S), \qquad \hat{y} = \text{Dense}_{14}(\tilde{S}) = \text{softplut}(W_{\hat{y}}\tilde{S} + b_{\hat{y}}).$$

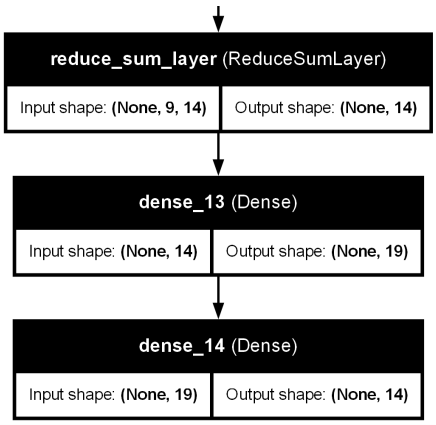

Figura 4.14: Layers che elaborano gli elementi uscenti dal blocco al fine di costruire la predizione di GeoDispNet.

4.2.2 HydroGeoDispNet

L'architettura del modello che elabora anche la portata del fiume presenta la stessa struttura di GeoDispNet: infatti, essa è composta dal ordinamento dell'input, la codifica spaziale delle coordinate ed i blocchi sequenziali ripetuti formati dai layer sopracitati. La differenza principale, visibile in Fig. 4.16, risiede nella presenza di un terzo input relativo alla portata del fiume nella finestra temporale di d giorni:

$$G \in \mathbb{R}^d.$$

Esso viene elaborato mediante un *Dense Layer*, in figura *flow_dense_1*,

$$\hat{G} = \text{Dense}_{f,1}(G)$$

prima di essere concatenato all'output della somma aggregata successiva all'ultimo blocco. Si osservi che i layers *dense_11* e *dense_12* corrispondono rispettivamente ai layers *dense_13* e *dense_14* di GeoDispNet.
Successivamente si procede come in GeoDispNet. Relativamente all'uso del *Drop out Layer*, si evidenzia come si ottengano performance migliori non utilizzandolo con l'input G relativo alla portata al fine di mantenere la totalità dell'informazione.

É, però, possibile utilizzarlo nei singoli blocchi. Infatti, è visibile in Fig. 4.15 in due posizioni: tra la MHA ed il *Normalization Layer* e tra il secondo *Dense Layer* ed il *Normalization Layer*. L'uso di tale strato, permette di diminuire il costo computazionale degli addestramenti senza compromettere le performance.

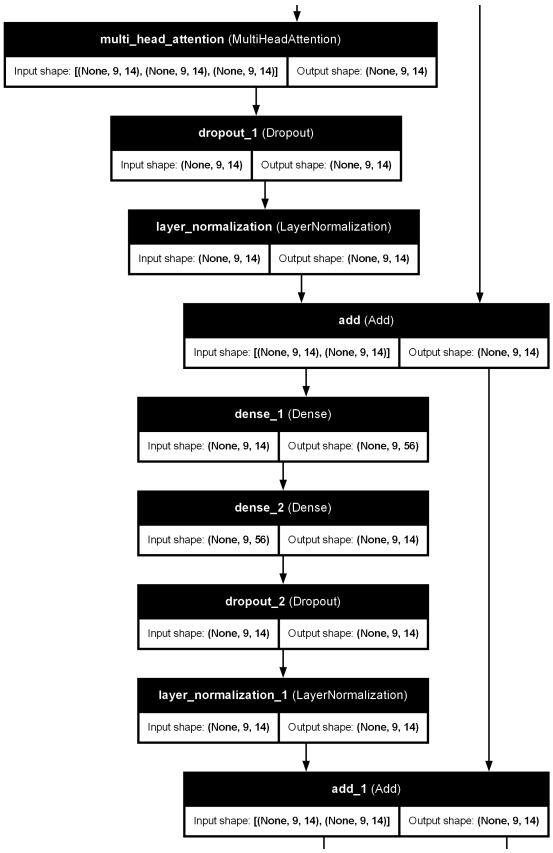

Figura 4.15: Blocco ripetuto in sequenza B volte in HydroGeoDispNet. Si noti la presenza di due *Drop Out Layers*.

4.3 Definizione degli iperparametri

Prima di descrivere gli addestramenti effettuati, si procede con la definizione degli iperparametri λ ottenuti mediante il tuner del *Random Search*, tecnica approfondita nel Cap. 2. In particolare, si effettua una ricerca causale della combinazione di iperparametri migliore per il modello all'interno dello spazio di ricerca Λ. Per fare ciò, il tuner addestra il modello su 20 epoche per un massimo di 15 trials; ognuno di questi ultimi presenta una combinazione differente dei λ. Inoltre, vengono definiti alcuni callbacks: per esempio, l'addestramento di uno specifico trial può terminare

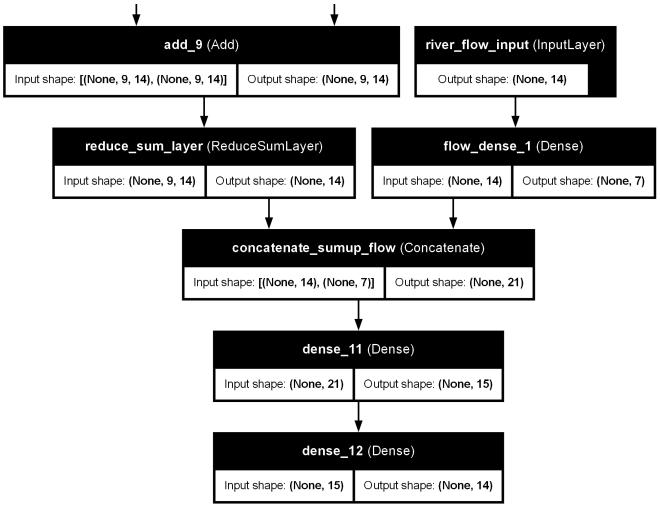

Figura 4.16: Layers che elaborano gli elementi uscenti dal blocco al fine di costruire la predizione di HydroGeoDispNet; river_flow_input rappresenta il terzo input G.

mediante l'*EarlyStopping*, che forza la fine del training se la loss non migliora dopo 5 epoche, oppure il *TerminateOnNan*, che interrompe l'addestramento se vi è la presenza di valori NaN per la loss. Infine, l'obiettivo del tuner è di minimizzare la loss calcolata sul validation dataset e considera r_1 come rischio target per il training con entrambi i modelli.
La definizione dei λ per questi ultimi, lo spazio di ricerca, esplicitato nel codice con l'oggetto *hp* di Keras [34], ed i valori scelti sono visibili in Tab. 4.7. Si noti che i λ riguardanti i layer che elaborano l'input (X, C) sono gli stessi sia per GeoDispNet sia per HydroGeoDispNet. Solamente per quest'ultimo, il tuner definisce gli iperparametri per l'input G.
In particolare, λ_1 si riferisce alla funzione di attivazione usata nel layers $\text{Dense}_{1,2,3}$ e lo spazio di ricerca è stato definito con l'obiettivo di minimizzare la possibilità di gradienti nulli per il modello e di migliorare la convergenza.
Il numero di blocchi è comparabile per entrambi i modelli mentre il numero di teste e la dimensione delle keys del MHA sono uguali. Inoltre, anche λ_7 e λ_8 risultano simili tra i due modelli e la dimensione del batch è la medesima. In GeoDispNet, λ_5 viene impostato come False: infatti, il Drop out non viene usato. Invece, in HydroGeoDispNet λ_5 e λ_{11} risultano True dalla ricerca effettuata dal tuner. Tuttavia, esperimenti eseguiti su datasets ridotti hanno dimostrato che utilizzare tale tecnica su entrambe le informazioni, displacements e portata del fiume, porta ad una convergenza verso un minimo locale della loss quando il numero di epoche supera quelle impostate per il tuner. Al contrario, impostando $\lambda_{11} =$

False, quindi passando al modello l'informazione completa sulla portata, permette di ottenere un miglioramento nelle prestazioni. Ciò accade in quanto il Drop Out, applicato durante la *Random Search*, restituisce i λ ottimali su quella specifica configurazione composta da un datataset ridotto con un numero di epoche minimo. Tali condizioni, però, possono non coincidere con quelle del training completo durante il quale il modello riceve un dataset più grande ed un numero di epoche maggiore. Di conseguenza, il modello potrebbe convergere a minimi locali e avere prestazioni peggiori se il Drop Out è mantenuto in uso anche durante il training.
Si noti che la dimensione del batch, λ_9, è uguale per entrambi i modelli e che il *learning rate* non è stato definito tramite il tuning degli iperparametri. Esso è stato impostato manualmente pari a 5e-4 al fine di avere un buon compromesso tra convergenza e stabilità numerica.
Il tuning degli iperparametri, gli addestramenti ed i test, spiegati nelle sezioni successive, sono stati eseguiti su una macchina che non possiede alcuna GPU: di conseguenza, ogni operazione è stata effettuata su CPU. In questo scenario, Tensorflow esegue alcune operazioni in modo ottimizzato tramite oneDNN, Deep Neural Network Library, la quale è una libreria di Intel volta a migliorare le performance soprattutto su CPU. Tale utilizzo può portare ad una lieve, ma in tale contesto rilevante, modifica dei risultati numerici rispetto ad una versione non ottimizzata a causa, per esempio, di errori di arrotondamento. Per esempio, è possibile visualizzare una differenza in alcuni iperparametri eseguendo due volte il tuning; o ancora, il training, nella prima esecuzione, può convergere ad un minimo locale, mentre, nella seconda, può convergere al minimo globale. Al fine di evitare risultati differenti tra esecuzioni diverse, si sono impostate delle variabili di ambiente, oltre al random seed della libreria Numpy [48], che garantiscono la totale riproducibilità dei risultati in quanto permettono di disattivare tali ottimizzazioni.

Iperparametro	**Spazio di Ricerca**	**GeoDispNet**	**HydroGeoDispNet**
λ_1 = funzione di attivazione	$\{\text{elu}, \tanh, \text{leaky\_relu}, \text{swish}\}$	leaky_relu	elu
λ_2 = num. di blocchi	$\{1, 2, 3, \ldots, 8\}$	6	5
λ_3 = num. di teste	$\{4, 6, 8, 10, 12\}$	12	12
λ_4 = dim. di key	$\{8, 9, 10, \ldots, 16\}$	13	13
λ_5 = uso del Drop Out	$\{\text{True}, \text{False}\}$	False	True*
λ_6 = tasso del Drop Out	$\{0.00, 0.05, 0.10, \ldots, 0.30\}$	0.1	0.15
λ_7 = unità di $\text{Dense}_1()$	$\{2d, 3d, 4d\}$	56	56
λ_8 = unità di $\text{Dense}_{13}()$	$\{\frac{d}{2}, \frac{d}{2}+4, \frac{d}{2}+8, \ldots, \frac{d}{2}+20\}$	19	15
λ_9 = dim. del batch	$\{32, 64\}$	32	32
λ_{10} = unità di $\text{Dense}_{f,1}()$	$\{\frac{d-2}{4}, \frac{d-2}{4}+4, \frac{d-2}{4}+8, d\}$	-	7
λ_{11} = uso del Drop Out fiume	$\{\text{True}, \text{False}\}$	-	True*
λ_{12} = tasso del Drop Out fiume	$\{0.00, 0.05, 0.10, \ldots, 0.30\}$	-	0.05

Tabella 4.7: Da sinistra verso destra: definizione degli iperparametri, spazio di ricerca per ogni iperparametro, valori numerici o booleani e funzioni trovate dal tuner per i due modelli in esame.

4.4 Addestramenti effettuati

Impostati gli iperparametri tramite il tuner per entrambi i modelli, si effettuano differenti training con i due modelli al fine di valutare nel complesso le loro performance. Quindi, si procede addestrando sia GeoDispNet sia HydroGeoDispNet con gli indici di incremento di rischio relativi al ponte di Tadcaster. Inizialmente, viene usato l'indice scelto a priori, y_1, come variabile target. Successivamente, considerando i rischi *datadriven*, ogni modello è addestrato sul rispettivo y_k in accordo con la metodologia utilizzata per la loro creazione. Per esempio, GeoDispNet viene addestrato utilizzando i dati relativi sia a y_2 sia a y_3, entrambi definiti unicamente mediante i displacements. Invece, HydroGeoDispNet viene addestrato su y_4 e, in seguito, su y_5, creati usando gli spostamenti verticali e la portata. Tale logica ha l'obiettivo, verificato nelle sezioni seguenti, di validare la capacità predittiva dei modelli con dati di test coerenti con quelli di training.
Ogni addestramento viene effettuato su ponti aventi un numero massimo di punti pari a 9 e creando 500 finestre temporali di 14 giorni mediante la funzione f_{rand}:

$$m = 500, \qquad d = 14, \qquad n = 9.$$

Inoltre, il dataset riservato al training deriva dalla scomposizione di quest'ultimo in tre insieme di dimensione differente. Il *training set* rappresenta il 40% del dataset creato, utile per definire i pesi ottimali del modello. Il *validation set* è di dimensione minore, 10%, e permette di rilevare l'overfittig. Infine, il *test set* rappresenta il 50% ed è utilizzato per valutare le prestazioni del modello fornendo una stima sulla capacità di quest'ultimo di generalizzare su dati mai visti.
I primi esperimenti, relativi al comprendere quale fosse l'architettura migliore per il modello, sono eseguiti su un dataset ridotto, ma mantenendo tali percentuali, e con un numero di epoche basso al fine di ridurre il tempo di calcolo per eseguire verifiche rapide. Superata tale fase, si è fissato un numero massimo di epoche pari a 1500 per ogni addestramento. L'obiettivo è di consentire al modello di raggiungere condizioni di training prolungato per ambire al miglior risultato. Eventuali interruzioni anticipate o riduzioni del *learninng rate* sono gestite in autonomia dai meccanismi di Callbacks precedentemente citati nel Cap. 2. In particolare, il *ReduceLROnPlateau* riduce di un fattore pari a 0.5 il *learning rate*, fino ad un minimo di 1e-6, ogni volta che dopo 75 epoche consecutive la validation loss non migliora rispetto al valore minimo fin ora trovato. Mentre il *EarlyStopping* interrompe anticipatamente l'addestramento se per 200 epoche consecutive la validation loss non migliora.
Si procede esaminando la bontà dell'addestramento di entrambi i modelli analizzandone la convergenza, all'aumentare del numero di epoche, ed i possibili fenomeni di overfitting. Ciò è valutato mediante la training e validation loss che calcola il MSE in ogni epoca. Tale risultato viene riportano in un grafico per ogni modello al fine di operare un confronto visivo tra gli indici di rischio. Inoltre, viene calcolato

anche il MAE: esso è utilizzato solamente per monitorare un'altra metrica durante il processo ma, a differenza del MSE, non influenza il processo di ottimizzazione. Quindi, quest'ultima non viene riportata nel lavoro di tesi.
Infine, entrambi i modelli sono stati valutati sul test set preventivamente separato dal training e dal validation set. Per farlo, si sono analizzate due metriche su tale insieme: Loss_{test} e MAE_{org}, definite mediante le formule 3.3 nel Cap. 3. Si osservi che la seconda metrica, essendo moltiplicata per $alpha_{inc}$, presenta la stessa unità degli incrementi di rischio. Di conseguenza, si ha a disposizione un valore interpretabile e confrontabile con i valori reali di y_k.

4.4.1 Addestramento di GeoDispNet

Prima di analizzare i valori della loss, è necessario approfondire l'indice di rischio r_2. Gli addestramenti preliminari su un dataset ridotto e un numero minore di epoche massime, circa 500, hanno evidenziato una criticità su tale indice. Infatti, come è visibile in Fig. 4.17, il modello riesce a prevedere solamente un andamento debolmente crescente ma non si avvicina alla forma reale del rischio.

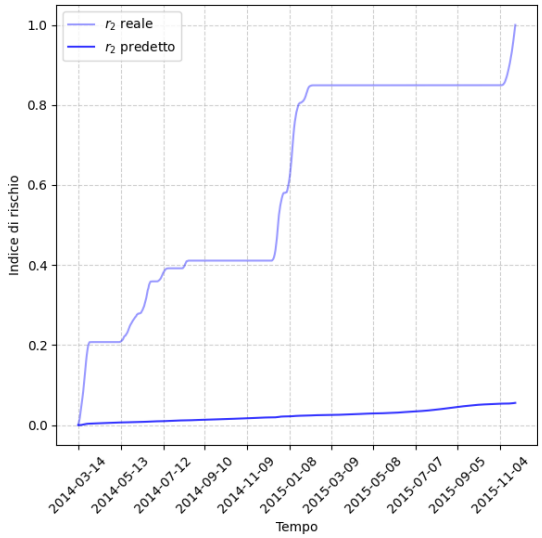

Figura 4.17: Errore sulla predizione dell'indice di rischio r_2 con GeoDispNet.

Le cause risiedono nei valori di y_2 rispetto a quelli di y_1. Si noti che la distribuzione di y_2 è differente rispetto quella di y_1: infatti, tale incongruenza è visibile nel grafico di sinistra della Fig. 4.18. Inoltre, si osserva che y_2 possiede una quantità di valori nulli maggiore di y_1:

$$|\{i \mid y_{1,i} = 0\}| = 1, \qquad |\{i \mid y_{2,i} = 0\}| = T_0,$$

dove il primo insieme ha cardinalità unitaria grazie alla definizione di y_1. Al fine di rendere compatibile la distribuzione di valori nulli di y_2 con y_1, quindi ottenere:

$$|\{i \mid \check{y}_{2,i} = 0\}| = 1,$$

si procede con una perturbazione di un numero di zeri di y_2 sommando valori molto piccoli al fine di non stravolgere la forma e l'andamento di r_2. Successivamente alla perturbazione, si ha

$$\check{y}_{2,i} = y_{2,i} + \ell_{1,i} + \ell_{2,i}$$

dove $\ell_{1,i}$ e $\ell_{2,i}$ sono elementi estratti da due vettori che seguono due distribuzioni casuali uniformi tali che:

$$\ell_1 \sim \mathcal{U}(2 \cdot 1e-4, 14 \cdot 1e-4), \qquad \ell_2 \sim \mathcal{U}(0, 5e-5).$$

Tale combinazione ed i relativi parametri sono stati scelti a seguito di esperimenti che rendessero casuale la perturbazione. Successivamente, poichè dopo tale operazione, la somma cumulata degli incrementi ha prodotto un indice di rischio con il massimo superiore ad uno, si è eseguita una normalizzazione tra zero ed uno del nuovo indice perturbato. Il risultato è visibile nel grafico di destra della Fig. 4.18. Le componenti di r_2 perturbato che possono sembrare lineari in realtà sono composte da piccole oscillazioni.

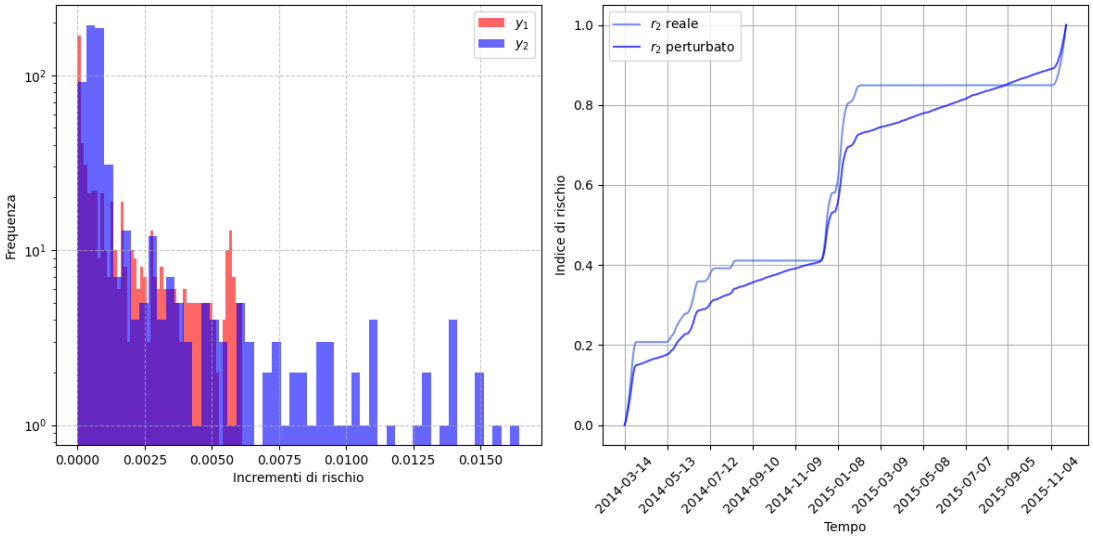

Figura 4.18: A sinistra, si confrontano le distribuzioni degli incrementi di rischio di r_1 e r_2. A destra, si visualizzano r_2 prima e successivamente alla perturbazione applicata.

Si procede ora con l'analisi della loss sul training e validatiion set relativa agli

addestramenti su GeoDispNet. In Fig. 4.19 è visibile il suo andamento con gli incrementi di rischio associati a r_1, r_2, successivo alla perturbazione, e r_3. Si noti che tutte le curve di training e validation mostrano una forte decrescita iniziale per poi rallentare e convergere ad un minimo stabile: ciò risalta l'avvenuto apprendimento da parte del modello. Inoltre, non si osservano rilevanti fenomeno di overfitting in quanto le curve di training e validaton non divergono in nessuno scenario in modo significativo. L'indice y_2 risulta più difficile da predire in quanto ha la training loss maggiore rispetto a quelle relative a y_3 e y_1. Si osserva però che y_1 ha una training loss più bassa di quella associata a y_2 e y_3 quindi il modello riesce a predire bene sui dati di training. Allo stesso tempo, y_1 presenta la validation loss più alta: ciò implica che GeoDispNet non è in grado di predire bene in egual misura anche su dati mai visti e, quindi, generalizza peggio. Infine, si noti che solamente l'addestramento su y_1 si ferma preventivamente poco dopo le 600 epoche grazie ai Callbacks: infatti, si nota dal grafico che la validation loss, dall'epoca 400, subisce una lenta crescita rispetto alla training loss.

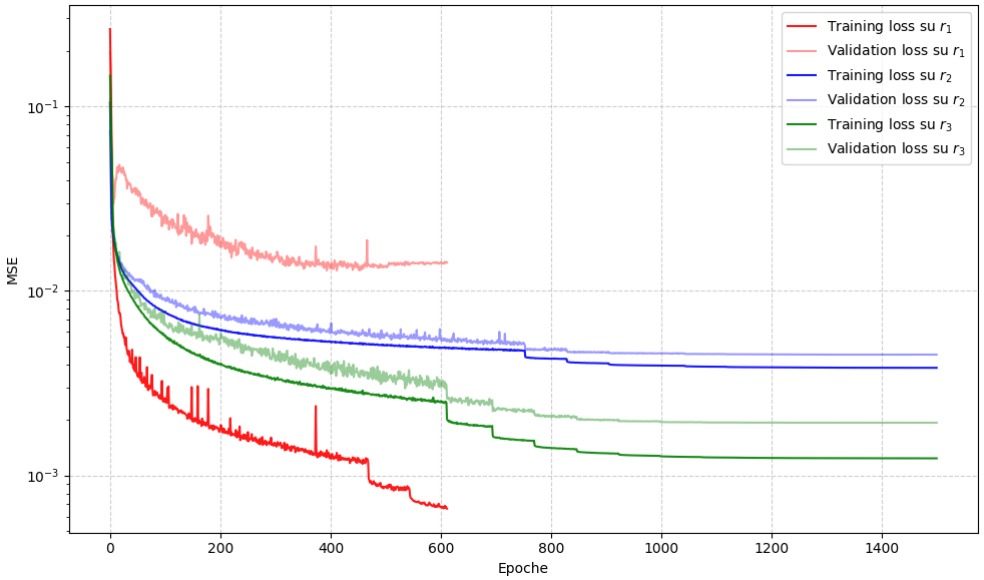

Figura 4.19: Confronto tra gli andamenti delle loss.

Si consideri ora il test set al fine di valutare GeoDispNet successivamente all'addestramento su dati non presenti nel training set. Come si evince dalla Fig. 4.20, r_1 presenta i valori di Loss_{test} e MAE_{org} maggiori: infatti, esso l'indice target di rischio con cui il modello generalizza peggio su dati mai visti. Al contrario, GeoDispNet addestrato su r_3 ottiene le migliori prestazioni. Inoltre, si osservi che i dati nel grafico di sinistra seguono la stessa gerarchia del grafico di destra. Infine, si noti che

i valori del MAE_{org} confrontato con y_k, risultano particolarmente piccoli mostrando una buona capacità predittiva da parte del modello.

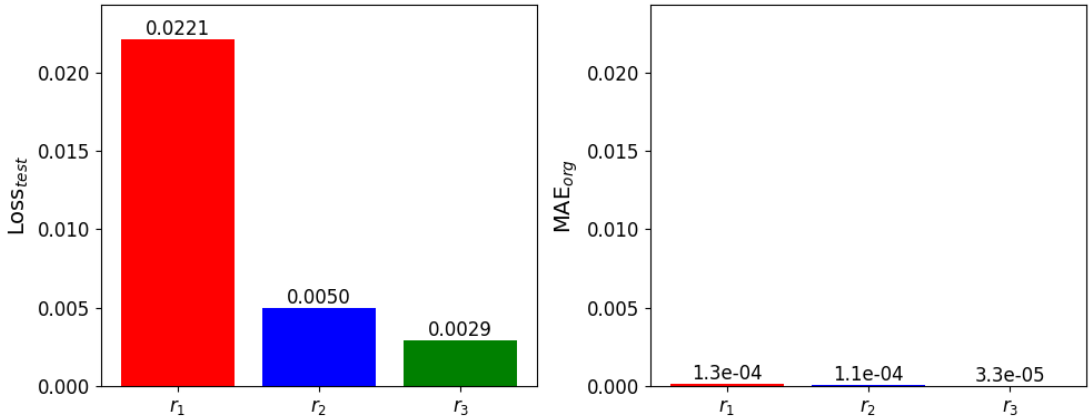

Figura 4.20: Confronto di Loss_{test}, a sinistra, e MAE_{org}, a destra.

4.4.2 Addestramento di HydroGeoDispNet

L'andamento di training e validation loss su HydroGeoDispNet viene valutato sugli incrementi relativi ai rischi r_1, r_4 e r_5. Come è possibile vedere in Fig. 4.21, l'evoluzione di tale metrica è simile a quanto descritto nella sezione precedente. Considerando entrambi gli scenari, si nota come y_1 venga predetto meglio sui dati di training ma, allo stesso tempo, entrambi i modelli addestrati su tale indice, generalizzino peggio, infatti hanno una validation loss maggiore. Inoltre, si noti che le loss associate a r_4 ed r_5 sono molto vicine tra loro e manifestano una convergenza più marcata: quindi, il modello non presenta overfitting ed è in grado di generalizzare.
I risultati di Loss_{test} e MAE_{org}, sono visibili in Fig. 4.22. In generale, i risultati sono comparabili con quelli ottenuti con GeoDispNet. In particolare, i valori più elevati si ottengo con r_1, confermando che anche HydroGeoDispNet ha difficoltà a generalizzare se addestrato su r_1.

4.5 Risultati

Ogni modello addestrato su un particolare y_k viene testato sul medesimo indice con $k \in \{1, ...,5\}$ per il ponte di Tadcaster e $k \in \{1,2,3\}$ per quello di Cantiano. Si noti che, essendo quest'ultimo utilizzato unicamente come test, i pesi addestrati sono riferiti al ponte di Tadcaster. L'obiettivo di tale fase di testing è di valutare

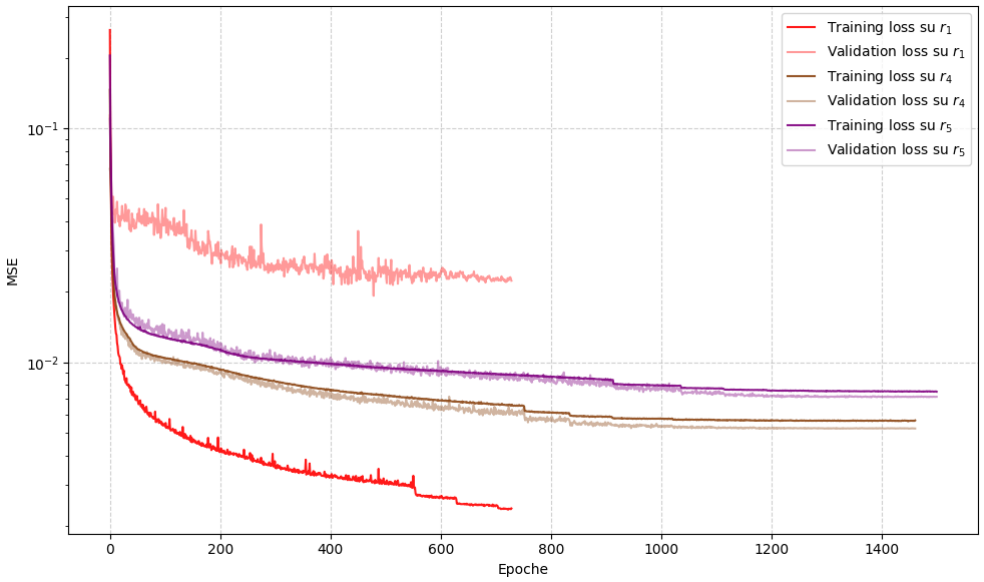

Figura 4.21: Confronto tra gli andamenti delle loss.

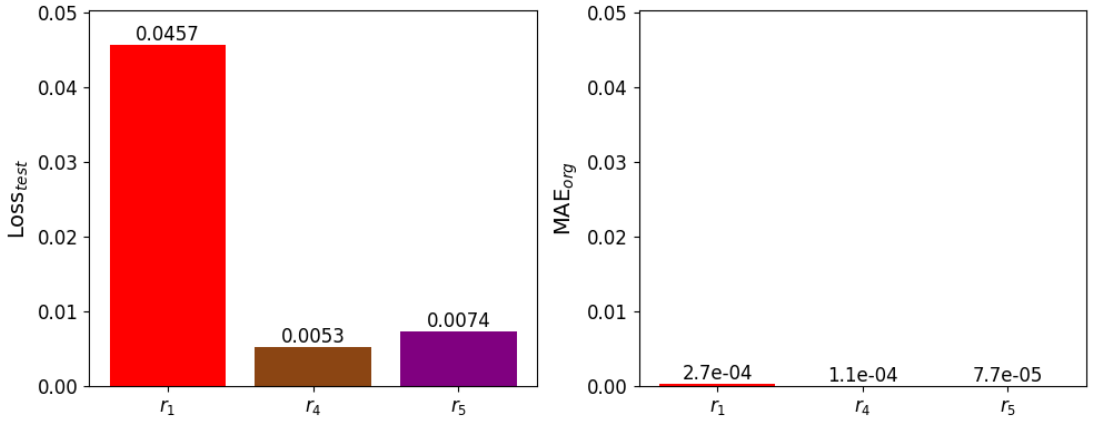

Figura 4.22: Confronto di Loss_{test}, a sinistra, e MAE_{org}, a destra.

la capacità di generalizzare del modello ad un contesto che non è incluso nei dati di addestramento.
Le valutazioni sulle prestazioni dei modelli sono eseguite mediante metriche e rappresentazioni grafiche, ognuna delle quali ha lo scopo di evidenziare un aspetto specifico. In particolare, è visualizzato un confronto qualitativo e quantitativo tra i valori del rischio reale e quelli del rischio predetto. Tale paragone viene effettuato rispettivamente mediante due grafici: il primo mostra i due indici di rischio al fine di evidenziare visivamente la differenza tra l'elemento predetto, identificato dalla

formula 3.1, e quello originale; il secondo, invece, mostra il valore di e^t_{abs}, calcolato mediante la formula 3.2 in ogni istante temporale in scala logaritmica. In seguito, viene visualizzato il valore del MAE in ogni istante t della finestra temporale di previsione mediante la fomula 3.5. Quest'ultimo viene calcolato su tutte le finestre temporali indipendentemente dalla posizione di essa rispetto all'arco temporale T delle misurazioni. Ciò permette di quantificare una possibile difficoltà del modello nella predizione in funzione di t. Infine, si mostrano i valori del MAE calcolato in due scenari differenti. Il primo considera tutti i giorni di tutte le finestre temporali dell'output del modello, formula 3.4, al fine di quantificare la bravura del modello nel predire finestre di dati. Invece, il secondo considera il rischio costruito dalle predizioni, formula 3.6, e misura quanto il modello è accurato nel prevedere l'andamento del rischio reale nell'intero arco temporale.

4.5.1 Valutazioni su Tadcaster mediante GeoDispNet

Si consideri la Fig. 4.23. Visivamente, la previsione dei rischi r_1 e r_3 risulta indistinguibile dal loro andamento reale. Invece, r_2 perturbato è sovrastimato anche se la forma del rischio viene predetta quasi correttamente. Tale risultato, sebbene migliore rispetto a quanto mostrato in Fig. 4.17, è peggiore rispetto agli altri due indici di rischio.

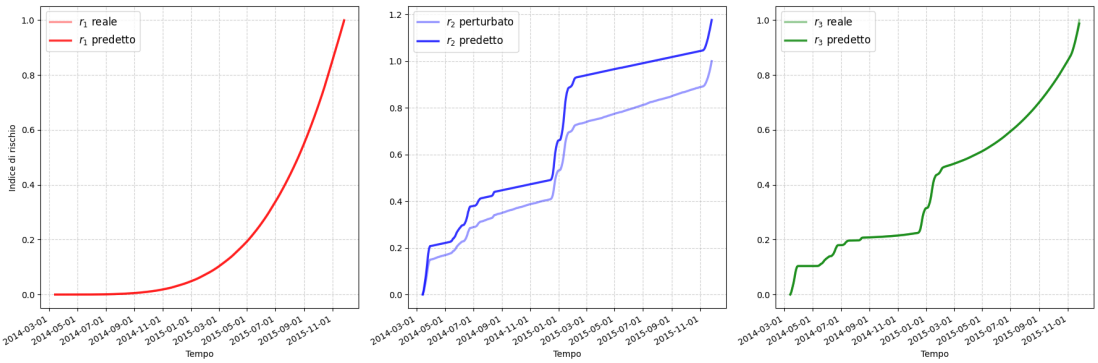

Figura 4.23: Confronto tra i rischi r_1, r_2 e r_3 reali e predetti.

Tale scenario è confermato in modo quantitativo dalla Fig. 4.24: infatti, si nota come l'andamento dell'errore assoluto $e_{abs,t}$, $t = 1, ..., T$, relativo a r_2 sia generalmente maggiore rispetto a quelli di r_1 ed r_3 che risultano invece paragonabili.
In Fig.4.25, è possibile visualizzare l'andamento del MAE_{day} per ogni giorno k di ogni finestra temporale, calcolato mediante la formula 3.5. Tutte le curve presentano un andamento stabile nel tempo e non si nota una crescita o decrescita rilevante. Tale risultato implica che il modello riesce a prevedere l'incremento di rischio in

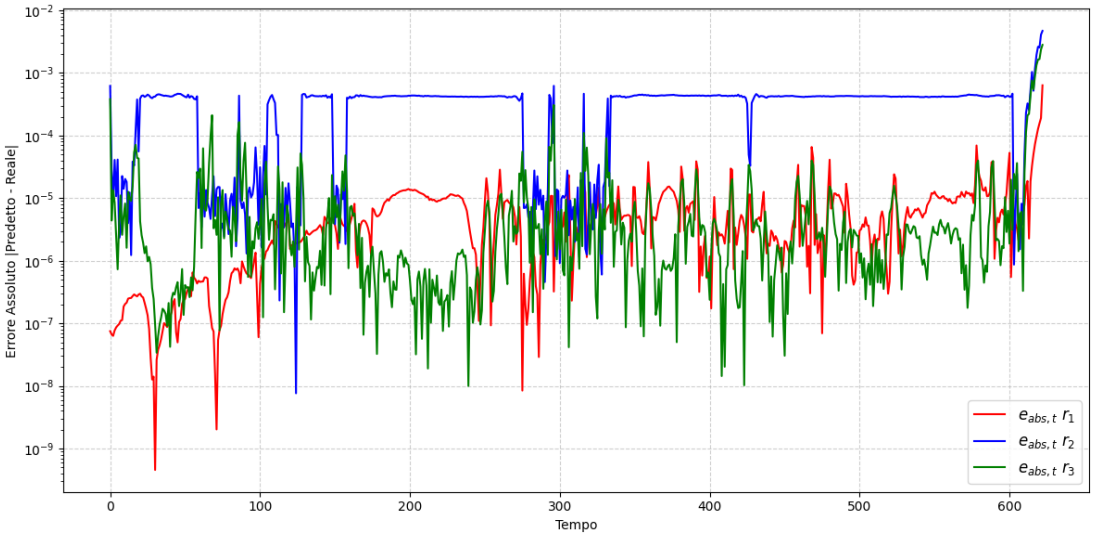

Figura 4.24: Confronto degli errori assoluti dei rischi r_1, r_2 e r_3.

ogni giorno con lo stesso margine di errore. Più nello specifico, si nota che MAE_{day} associato a y_1 è il più basso: infatti, il modello riesce a prevedere meglio y_1 e, di conseguenza, r_1. Invece, i valori relativi a y_2 e y_3 sono rispettivamente circa 10 e 20 volte più grandi, nonostante siano degli errori piccoli.

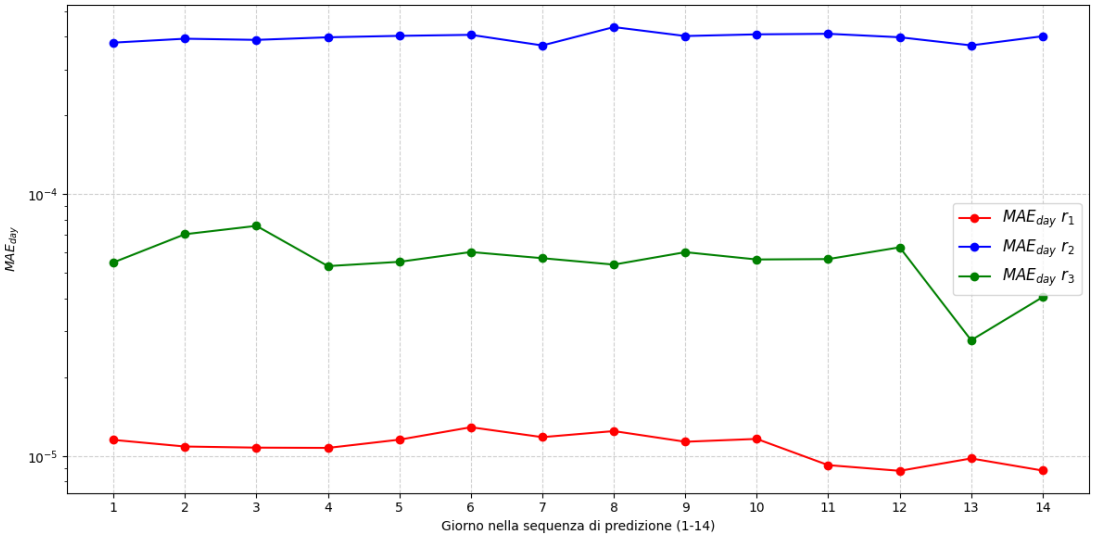

Figura 4.25: Confronto dei MAE_{day} relativi agli incrementi di rischio y_1, y_2 e y_3.

4.5.2 Valutazioni su Tadcaster mediante HydroGeoDispNet

Si consideri la Fig. 4.26. Tutti e tre i rischi sono sottostimati rispetto al rischio reale. In particolare, anche r_1 viene predetto peggio nonostante sia il rischio più semplicemente definito essendo il risultato di una funzione polinomiale. Si noti, inoltre, che in tale scenario, r_4 viene predetto senza essere perturbato come è stato fatto per r_2. Ciò accade grazie all'informazione sulla portata dell'acqua che viene elaborata da HidroGeoDisp.

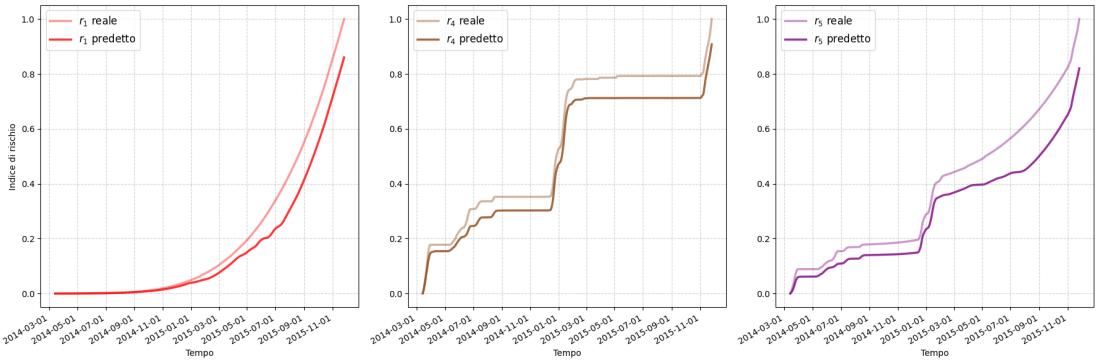

Figura 4.26: Confronto tra i rischi r_1, r_4 e r_5 reali e predetti.

Visionando la Fig. 4.27, si coglie che $e_{abs,t}$ di $r_4 \sim \mathcal{O}(10^{-10})$ in alcuni periodi temprali. Infatti, è possibile notare che questi ultimi corrispondono alle zone in cui il rischio cresce maggiormente e nelle quali r_4 viene predetto meglio, quindi mostra un errore assoluto minore. In generale, si nota come l'ordine di grandezza degli errori è paragonabile a quelli generati con il modello GeodispNet riportati in Fig. 4.24.
L'analisi della Fig. 4.28 risulta simile a quella effettuata per il grafico del MAE_{day} di GeoDispNet. Anche in questo scenario, non è visibile una crescita o una decrescita tale da far ipotizzare che il modello riesca a prevedere meglio, quindi produce un errore minore, in alcuni giorni k della finestra temporale. Inoltre, la metrica associata a y_1 risulta minore di quella relativa sia a y_4 sia a y_5.

4.5.3 Valutazioni su Cantiano mediante GeoDispNet

Al fine di analizzare i risultati ottenuti da GeoDispNet, addestrato su y_k con $k = \{1,2,3\}$, riguardo i medesimi incrementi di rischio per il ponte di Cantiano, è necessario approfondire alcuni aspetti al fine di preparare il dataset relativo a tale ponte. Per prima cosa, vi è una discrepanza tra gli archi temporali ai quali si riferiscono i dati del ponte di Tadcaster e di quello di Cantiano: infatti, essi fanno riferimento rispettivamente a 623 giorni e 4124 giorni. Al fine di rendere

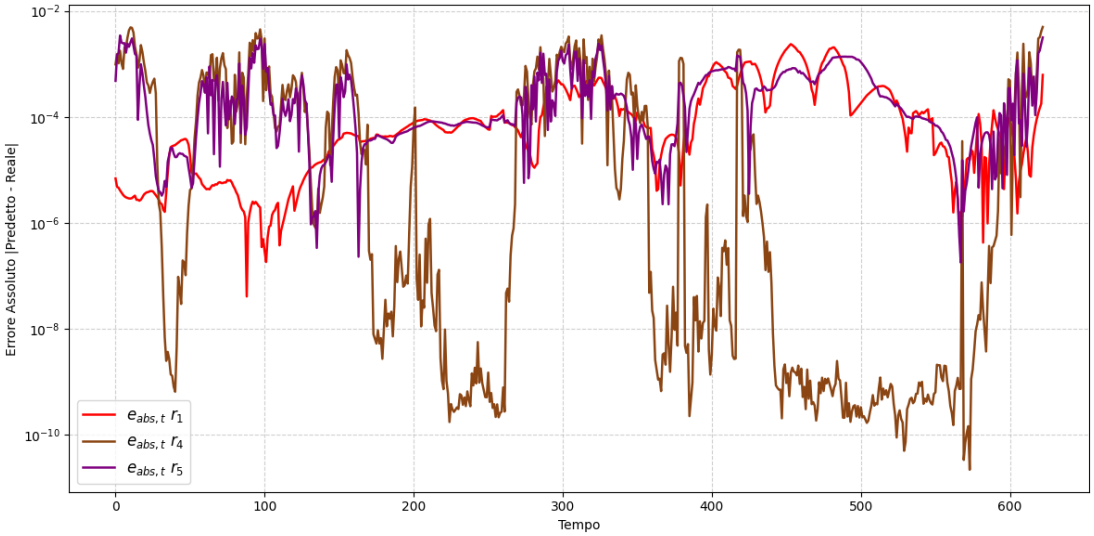

Figura 4.27: Confronto tra gli errori assoluti dei rischi r_1, r_4 e r_5.

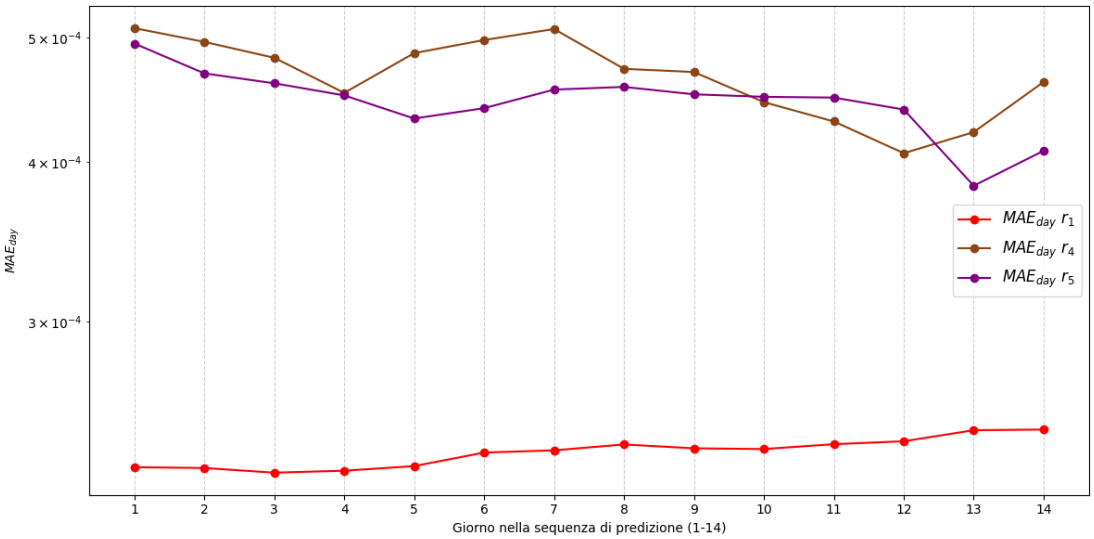

Figura 4.28: Confronto dei MAE_{day} relativi agli incrementi di rischio y_1, y_4 e y_5.

compatibile il modello ad un periodo di tempo maggiore, si procede scalando gli incrementi di rischio predetti per un fattore pari a $\frac{623}{4124}$.
Inoltre, per garantire la coerenza statistica tra i dati di Cantiano e i dati di Tadcaster, utilizzati come training set, si procede ad una trasformazione dei valori dei displacements di Cantiano. L'obiettivo è che le distribuzioni delle colonne considerate abbiano media e deviazione standard coerenti con quelle del dataset

usato come training. Quindi, i displacements successivi a tale modifica $\Delta'_{i,j}$ sono così definiti:

$$\Delta'_{i,j} = \left(\frac{\Delta_{i,j} - \mu_C}{\sigma_C} \right) \cdot \sigma_{tr} + \mu_{tr} \qquad i = 1, ..., T_C, \quad j \in \mathcal{P}_C,$$

dove T_C e $\mathcal{P}_C$ sono rispettivamente l'ultimo istante temporale di Cantiano e l'insieme dei punti del ponte di Cantiano. μ_C e μ_{tr} sono le medie calcolate su tutti i Δ del ponte di Cantiano e dell'insieme di training, mentre σ_{tr} e σ_C sono le deviazioni standard.
Si procede con l'analizzare i risultati ottenuti sul ponte di Cantiano. In Fig. 4.29 è possibile visualizzare la predizione degli indici di rischio confrontate con i valori reali. Come si poteva prevedere, si notano performance peggiori rispetto ai risultati ottenuti con il dataset riferito a Tadcaster, usato come base per il dataset di training e punto di partenza per la creazione dei datasets sintetici. Di conseguenza, si nota una minore accuratezza nei rischi predetti ed una maggiore difficoltà del modello a generalizzare. In generale, si noti come l'andamento dei rischi predetti è uguale ad una retta in tutti gli scenari analizzati indipendentemente da quello reale. Quindi il modello è in grado di prevedere solamente un aumento del rischio nel tempo ma non il suo reale andamento. In particolare, si osserva che r_1, rischio definito a priori, è sottostimato in modo evidente, anche con l'utilizzo del fattore correttivo. Tuttavia, tale risultato è di minore importanza non essendo r_1 un rischio *data driven*. Al contrario, r_2 e r_3 sono di maggiore interesse. Le predizioni generate direttamente dall'output del modello, linea gialla, sovrastimano il rischio reale in modo evidente. La nota positiva, però, è che l'ordine di grandezza di questi ultimi diventa comparabile con i rischi reali, dopo aver applicato il fattore correttivo, linea tratteggiata, nonostante la previsione non rispecchi l'andamento reale. Si osservi che r_3 risulta sovrastimato, soprattutto nella parte centrale dell'arco temporale. Al contrario, la predizione di r_2 risulta più fedele. É importante sottolineare che gli incrementi nulli di quest'ultimo indice di rischio sono stati perturbati, in quanto GeoDispNet è stato addestrato su un dataset con incrementi nulli anch'essi perturbati.
In generale, ciò mostra una difficoltà del modello di fornire stime coerenti su datasets che non rientrano nell'insieme di training.
In Fig. 4.30, è possibile visualizzare quantitativamente tale risultato. Si noti che i valori riportati sono in media di ordini di grandezza maggiori rispetto ai valori ottenuti con i test sul ponte di Tadcaster. Ciò conferma una maggiore difficoltà di GeoDispNet quando viene testato su dataset che non rientra nel training. Si noti che $e_{abs,t}$ di r_1 presenta un andamento relativamente regolare e poco rumoroso. Tale comportamento era di facile previsione in quanto rispecchia il valore assoluto della differenza di una funzione lineare e un polinomio di quarto grado. In particolare, dal 2014 in poi si assiste ad un aumento dell'errore: infatti, l'andamento del rischio

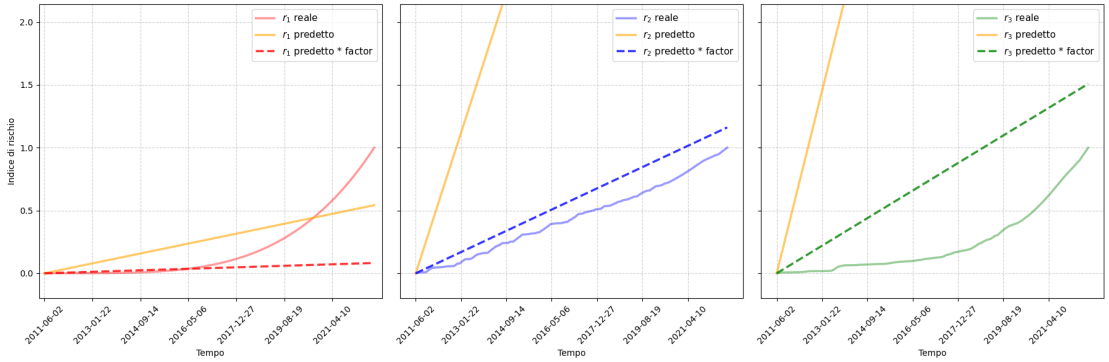

Figura 4.29: Confronto tra i rischi r_1, r_2 e r_3 reali e predetti, con e senza il fattore correttivo.

reale e predetto divergono in maniera significativa in Fig. 4.29. Invece, r_2 ed r_3 presentano andamenti più rumorosi senza mostrare un netto peggioramento come per r_1. Tale scenario rispecchia la maggiore complessità dei rischi reali.

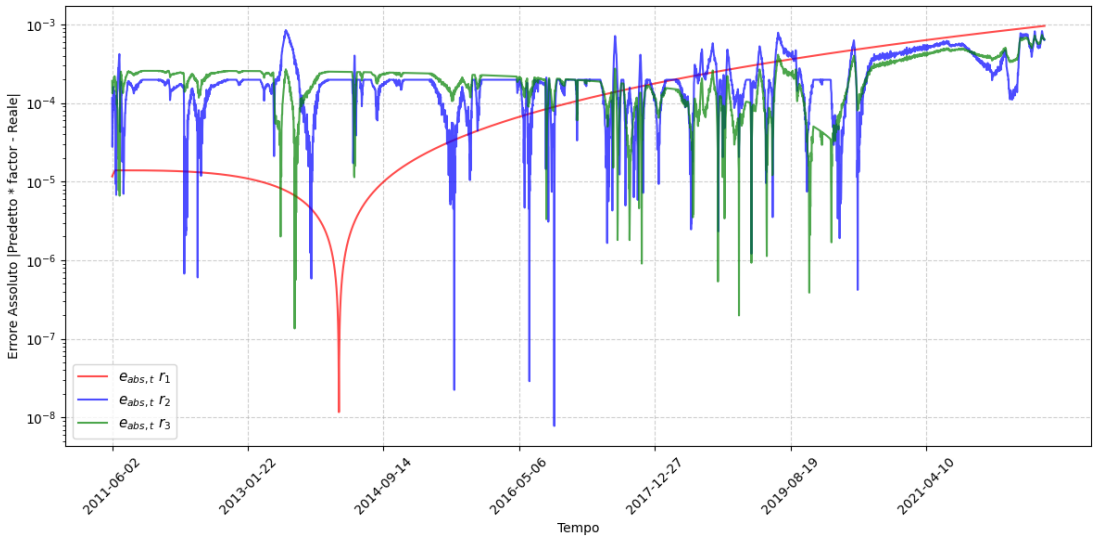

Figura 4.30: Confronto tra gli errori assoluti dei rischi predetti r_1, r_2 e r_3 moltiplicati per il fattore correttivo.

La Fig. 4.31 illustra l'andamento del MAE_{day} per il ponte di Cantiano. I valori hanno un ordine di grandezza accettabile e possono sembrare in contraddizione con quanto mostrato fino ad ora. Infatti, tali valori sono comparabili a quelli delle Fig. 4.25 e 4.28, relative a previsioni sul rischio di Tadcaster, le cui prestazioni sono state valutate come superiori. Tuttavia, è importante sottolineare che il MAE_{day} è

calcolato direttamente sull'output del modello, che è la predizione dell'incremento di rischio, e non sull'indice di rischio. Di conseguenza, è plausibile osservare valori di MAE_{day} con tale ordine di grandezza nonostante la stima del rischio risultante sia qualitativamente e quantitativamente peggiore. Tale fenomeno è maggiormente evidente su r_1: infatti, si osserva un MAE_{day} minore ma, allo stesso tempo, un rischio predetto lontano da quello reale.

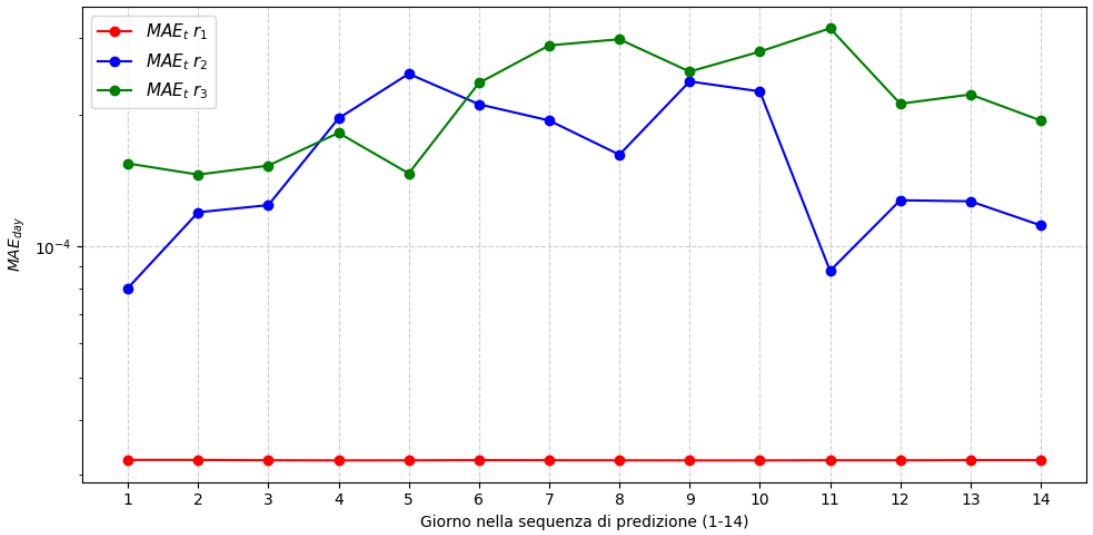

Figura 4.31: Confronto dei MAE_{day} relativi agli incrementi di rischio y_1, y_2 e y_3 con il fattore correttivo.

4.5.4 Confronto numerico tra modelli

Infine, si opera un confronto numerico tra le performance ottenute con GeoDispNet e HydroGeoDispNet sui ponti di Tadcaster e Cantiano. Per ognuno di essi, è calcolato il MAE_{pred}, che considera direttamente le predizioni del modello, ed il MAE_{post}, che è basato sulle predizioni aggregate, definiti rispettivamente mediante le formule 3.4 e 3.6. A tal fine si consideri la Fig. 4.32, nella quale vi sono due grafici che mostrano gli stessi scenari per ognuna delle due metriche. In particolare, si osserva r_1 analizzato con entrambi i modelli su Tadcaster e con GeoDispNet su Cantiano. Invece, r_2 ed r_3 sono visibili insieme a r_4 ed r_5: per esempio, considerando r_2, si comparano le prestazioni di GeoDispNet su Tadacaster e su Cantiano con le performance di HydroGeoDispNet su Tadcaster su r_4.
In generale, i risultati sono in linea con quanto ottenuto fino ad ora. In particolare, si consideri il grafico di sinistra. In esso, si noti che i valori di MAE_{pred}, calcolato considerando direttamente l'output del modello, sono minimi e si hanno per le

predizioni sul ponte di Tadcaster mediante GeoDispNet. Mentre HydroGeoDispNet, applicato allo stesso ponte, ottiene valori di poco più alti: infatti, le predizioni mostrate in Fig. 4.26 risultano peggiori. Al contrario, si osserva un MAE_{pred} maggiore su Cantiano, soprattutto con i rischi *data driven*.
MAE_{post}, che considera l'output successivo all'aggregazione delle finestra temporali, ha valori più simili tra i vari scenari. In particolare, GeoDispNet su Tadcaster presenta quantità minime: infatti, il rischio predetto e quello reale sono sovrapposti nella Fig. 4.23, quindi indistinguibili visivamente. HydroGeoDispNet su Tadcaster ha valori di MAE_{post} superiori e paragonabili a GeoDispNet su Tadcaster, infatti tali scenari sono predetti in modo peggiore. Si noti come le performance di Cantiano diventano paragonabili grazie alle modifiche apportate dai fattori correttivi riguardanti il periodo temporale e la distribuzione dei dispalcements introdotti precedentemente.
Infine, si osservi che, da tali considerazioni, sembrerebbe che l'aggiunta dell'informazione sull'acqua porti a errori maggiori e, di conseguenza, a predizioni peggiori. In realtà, il punto di forza di tale scenario è la possibilità di prevedere con un minimo errore il rischio totalmente *data driven* senza la necessità di perturbare gli incrementi nulli.

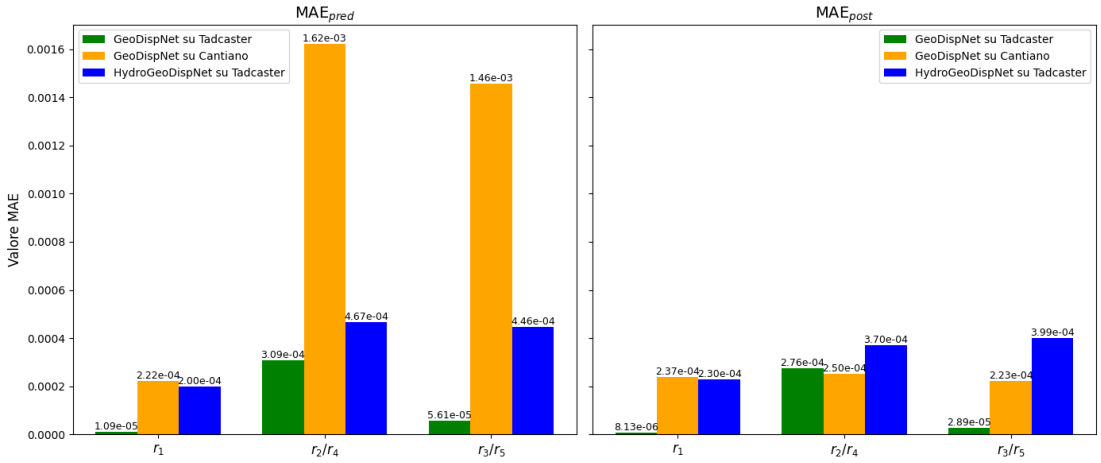

Figura 4.32: Confronto di MAE_{pred} e MAE_{post} relativi a y_k con $k = \{1, ..., 5\}$.

Capitolo 5

Conclusioni e possibili sviluppi

La presente tesi ha avutolo scopo di mostrare come il Deep Learning applicato al monitoraggio di ponti possa avere un ruolo fondamentale nella prevenzione di crolli. In particolare, si sono sfruttate le serie temporali relative ai displacements ottenute mediante elaborazione di immagini SAR e delle varie tecniche derivanti dall' InSAR.
Al fine di perseguire tale obiettivo, si sono creati due modelli: GeoDispNet, che considera in input gli spostamenti verticali in millimetri, e HydroGeoDispNet, che prende in esame anche la portata dell'acqua. Gli iperparametri dei due modelli sono stati definiti attraverso la tecnica della Random Search. Inoltre, entrambi i modelli sono stati addestrati su un dataset sintetico creato a partire da un dataset originale relativo al ponte di Tadcaster. L'andamento della loss evidenzia che in nessuno degli scenari analizzati vi sono fenomeni di overfitting. Inoltre, esso mostra che l'incremento di rischio relativo a r_1, definito a priori, è più facile da prevedere per i modelli in quanto presenta valori minori della funzione di perdita. Allo stesso tempo, la curva della validation loss si distacca molto, sintomo che i modelli non sono in grado di generalizzare, ovvero replicare il risultato ottenuto sul training set anche nel validation set. Al contrario, gli altri indici di rischio *data driven* presentano curve di validation e training loss vicine tra loro, quindi il modello riesce a generalizzare su dati mai visti
Inizialmente i modelli sono stati valutati separatamente. I risultati con GeoDispNet su Tadcaster mostrano una buona capacità predittiva, visivamente perfetta per r_1 e r_5 mentre per r_4, l'indice di rischio perturbato, vi è una sovrastima. Tale risultato è confermato anche dall'errore assoluto e dal MAE_{day}. Quindi, si evince che il rischio definito a priori viene predetto in modo migliore.
Considerando ancora il ponte di Tadcaster, le valutazioni con HydroGeoDisoNet

indicano che r_1, r_4 e r_5 vengono qualitativamente sottostimatiti. Il MAE_{day} suggerisce, anche in tal caso, che y_1 è l'incremento di rischio predetto meglio. Il risultato più rilevante è che l'aggiunta della portata dell'acqua permette di non perturbare r_4 prima dell'addestramento: infatti, il modello riesce a prevederne l'andamento ed è in grado di prevedere intervalli temporali in cui y_4 risulta nullo.
GeoDispNet è stato valutato anche sul ponte di Cantiano, il quale non è presente all'interno del training set. In tale scenario il modello ha prestazioni inferiori in quanto le predizioni, che sono delle linee rette, non rispecchiano gli andamenti reali. In particolare, r_1 ha la previsione peggiore: tale risultato non incide le conclusioni in quanto questo rischio è l'unico che non si basa sui dati. Al contrario, le previsioni di r_2 e di r_3 possiedono un ordine di grandezza paragonabile ai valori reali. In particolare, la predizione di r_2 è la più fedele al rischio reale. Visivamente, la qualità delle predizioni non è paragonabili a quelle per Tadcaster, invece i grafici di $e_{abs,t}$ e MAE_{day} sono comparabili. Ciò mostra una difficoltà del modello con un ponte non presente nel training set.
Infine, si sono comparati i due modelli mediante le metriche MAE_{pred} e MAE_{post}, le quali mostrano continuità rispetto ai risultati ottenuti in precedenza.
In conclusione, i risultati ottenuti su r_1 sono i migliori in termini training loss ma, allo stesso tempo, hanno prestazioni peggiori sul validation loss manifestando, quindi, una peggiore capacità di generalizzazione. I rischi *data driven* colmano questa differenza. Inoltre, HydroGeoDispNet mostra buoni risultati e permette di prevedere anche incrementi di rischio nulli. Infine, GeoDispNet applicato al ponte di Cantiano mostra performance inferiori, in quanto il modello prevede l'incremento del rischio ma non il reale andamento.

Un ipotetico futuro sviluppo potrebbe consistere nel migliorare la qualità del training e del test set a disposizione. Per esempio includendo più ponti nell'addestramento in modo da aumentare la variabilità dei dati di input. Oppure aumentando il numero di punti su cui sono calcolati i displacements in modo da avere una rappresentazione più completa del ponte. Un ulteriore ampliamento riguarderebbe l'includere ulteriori elementi naturali che possono influire sulla stabilità della struttura, come le piogge ed i venti, ed fenomeni estremi. Inoltre, un'evoluzione, sarebbe il ridefinire il campionamento degli intervalli di incremento di rischio al fine di evitare scenari simili a quello di r_4 in cui è stato necessario perturbare in parte gli elementi nulli per ottenere una previsione ragionevole. Infine, per queste analisi sono stati usati due ponti crollati: uno sviluppo interessante, consisterebbe nell'applicare tale metodologia a ponti tuttora in vita al fine di valutare le prestazioni predittive.

Bibliografia

[1] Giulia Delo, Marco Civera, Erica Lenticchia, Gaetano Miraglia, Cecilia Surace e Rosario Ceravolo. «*Interferometric Satellite Data in Structural Health Monitoring: An Application to the Effects of the Construction of a Subway Line in the Urban Area of Rome*». In: *Applied Sciences* 12.3 (2022), p. 1658 (cit. alle pp. 1–4, 27).

[2] Charles R. Farrar e Keith Worden. *Structural Health Monitoring: A Machine Learning Perspective*. Chichester, UK: John Wiley & Sons, 2012. ISBN: 1-118-44321-7 (cit. alle pp. 2, 6).

[3] James B. Campbell e Randolph H. Wynne. *Introduction to Remote Sensing*. New York, NY, USA: Guilford Press, 2011. ISBN: 1-60918-177-8 (cit. a p. 2).

[4] Ernesto Rodriguez e John Martin. «*Theory and Design of Interferometric Synthetic Aperture Radars*». In: *IEE Proceedings F - Radar and Signal Processing* 139.2 (1992), pp. 147–159. DOI: 10.1049/ip-f-2.1992.0020 (cit. alle pp. 2, 29).

[5] ReLUIS, Consorzio Rete dei Laboratori Universitari di Ingegneria Sismica e CNR-IREA, Istituto per il Rilevamento Elettromagnetico dell'Ambiente. *Linee guida per l'utilizzo dei dati interferometrici satellitari ai fini dell'interpretazione del comportamento strutturale delle costruzioni*. Rapp. tecn. Ver. 7. Italia: ReLUIS – CNR IREA, ott. 2023 (cit. alle pp. 3–5, 28).

[6] Diego Alejandro Talledo e Anna Saetta. «*A Multi-Level Semi-Automatic Procedure for the Monitoring of Bridges in Road Infrastructure Using MT-DInSAR Data*». In: *Remote Sensing* 17.14 (2025), p. 2377. DOI: 10.3390/rs17142377. URL: https://doi.org/10.3390/rs17142377 (cit. alle pp. 4, 6).

[7] Francesca Bozzano, Ivan Cipriani, Paolo Mazzanti e Alberto Prestininzi. «*Displacement Patterns of a Landslide Affected by Human Activities: Insights from Ground-Based InSAR Monitoring*». In: *Natural Hazards* 59 (2011), pp. 1377–1396. DOI: 10.1007/s11069-011-9832-4 (cit. alle pp. 5, 6).

[8] Amin Tavakkoliestahbanati, Pietro Milillo, Hao Kuai e Giorgia Giardina. «*Pre-collapse spaceborne deformation monitoring of the Kakhovka dam, Ukraine, from 2017 to 2023*». In: *Communications Earth & Environment* 5 (2024), p. 145. DOI: 10.1038/s43247-024-01465-0 (cit. a p. 5).

[9] Said Quqa, Antonio Palermo e Alessandro Marzani. «*Regional-scale bridge condition monitoring using InSAR displacements and environmental data*». In: *Structural Health Monitoring* 24.4 (2024). First Published 2024. DOI: 10.1177/14759217241302369. URL: https://doi.org/10.1177/14759217241302369 (cit. a p. 5).

[10] S. Selvakumaran, S. Plank, C. Geiß, C. Rossi e C. Middleton. «*Remote Monitoring to Predict Bridge Scour Failure Using Interferometric Synthetic Aperture Radar (InSAR) Stacking Techniques*». In: *International Journal of Applied Earth Observation and Geoinformation* 73 (2018), pp. 463–470. DOI: 10.1016/j.jag.2018.07.004 (cit. alle pp. 5, 29, 30, 38, 40).

[11] NHAZCA S.R.L. *Studio delle deformazioni del suolo tramite interferometria SAR satellitare (InSAR) in corrispondenza di 7 ponti crollati durante l'alluvione del settembre 2022 nelle Marche*. Relazione Tecnica NZ4870-$U_1$90625. Cliente: Politecnico di Torino, CIG: B672621889. Roma, Italia: NHAZCA S.R.L., giugno 2025 (cit. alle pp. 5, 30).

[12] Yassir Hamzaoui, Marco Civera, Andrea Miano, Manuela Bonano, Francesco Fabbrocino, Andrea Prota e Bernardino Chiaia. «*Hierarchical Clustering and Small Baseline Subset Differential Interferometric Synthetic Aperture Radar (SBAS-DInSAR) for Remotely Sensed Building Identification and Risk Prioritisation*». In: *Remote Sensing* 17.1 (2025), p. 128. ISSN: 2072-4292. DOI: 10.3390/rs17010128. URL: https://www.mdpi.com/2072-4292/17/1/128 (cit. a p. 5).

[13] I. Goodfellow, Y. Bengio e A. Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613. URL: https://www.deeplearningbook.org (cit. alle pp. 9, 21).

[14] W. S. McCulloch e W. Pitts. «*A Logical Calculus of the Ideas Immanent in Nervous Activity*». In: *Bulletin of Mathematical Biology* 52.1/2 (1990). Ristampato da Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133 (1943), pp. 99–1115 (cit. a p. 9).

[15] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley, 1949 (cit. a p. 10).

[16] F. Rosenblatt. «*The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*». In: *Psychological Review* 65.6 (1958), pp. 386–395 (cit. a p. 10).

[17] M. L. Minsky e S. A. Papert. *Perceptrons: An Introduction to Computational Geometry*. Expanded. Cambridge, MA: MIT Press, 1988, p. 258. ISBN: 0-262-63111-3 (cit. a p. 11).

[18] D. E. Rumelhart, G. E. Hinton e R. J. Williams. «*Learning Representations by Back-Propagating Errors*». In: *Nature* 323.6088 (1986), pp. 533–536 (cit. alle pp. 11, 24).

[19] A. Mohamed, G. Dahl e G. Hinton. «*Deep Belief Networks for Phone Recognition*». In: *Advances in Neural Information Processing Systems 22 (NIPS 2009)*. Curran Associates, Inc., 2009 (cit. a p. 13).

[20] A. Krizhevsky, I. Sutskever e G. E. Hinton. «*ImageNet Classification with Deep Convolutional Neural Networks*». In: *Advances in Neural Information Processing Systems 25 (NIPS 2012)*. 2012, pp. 1097–1105 (cit. a p. 13).

[21] L. R. Medsker e L. C. Jain, cur. *Recurrent Neural Networks: Design and Applications*. The CRC Press International Series on Computational Intelligence. CRC Press, 2001 (cit. a p. 14).

[22] S. Hochreiter e J. Schmidhuber. «*Long Short-Term Memory*». In: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. a p. 14).

[23] I. Sutskever, O. Vinyals e Q. V. Le. «*Sequence to Sequence Learning with Neural Networks*». In: *Advances in Neural Information Processing Systems* 27 (2014), pp. 3104–3112 (cit. a p. 14).

[24] D. Bahdanau, K. Cho e Y. Bengio. «*Neural Machine Translation by Jointly Learning to Align and Translate*». In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. 2015. URL: `https://arxiv.org/abs/1409.0473` (cit. a p. 14).

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser e I. Polosukhin. «*Attention Is All You Need*». In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 5998–6008 (cit. alle pp. 14, 15, 17).

[26] Y. LeCun, Y. Bengio e G. Hinton. «*Deep Learning*». In: *Nature* 521.7553 (2015), pp. 436–444. DOI: `10.1038/nature14539` (cit. a p. 16).

[27] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software disponibile su tensorflow.org. 2015. URL: `https://www.tensorflow.org/` (cit. alle pp. 16, 19, 22, 24, 57).

[28] S. R. Dubey, S. K. Singh e B. B. Chaudhuri. «*Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*». In: *Neurocomputing* (2022). arXiv: `2109.14545 [cs.LG]` (cit. a p. 16).

[29] F. Della Santa. «*Data-Driven Deep Learning Methods for Physically-Based Simulations*». Doctoral dissertation. Politecnico di Torino, 2021 (cit. a p. 17).

[30] A. P. Parikh, O. Täckström, D. Das e J. Uszkoreit. «*A Decomposable Attention Model for Natural Language Inference*». In: *arXiv preprint arXiv:1606.01933* (2016). URL: https://arxiv.org/abs/1606.01933 (cit. a p. 17).

[31] M.-T. Luong, H. Pham e C. D. Manning. «*Effective Approaches to Attention-based Neural Machine Translation*». In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. Lisbon, Portugal, 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166 (cit. a p. 17).

[32] J. Ren e H. Wang. *Mathematical Methods in Data Science*. 1ª ed. Chapter 3, Section 3.7.2.4 Softmax function. Elsevier, 2023. ISBN: 9780443186790 (cit. a p. 18).

[33] J. L. Ba, J. R. Kiros e G. E. Hinton. «*Layer Normalization*». In: *arXiv preprint arXiv:1607.06450* (2016). URL: https://arxiv.org/abs/1607.06450 (cit. a p. 19).

[34] Keras Team. *Keras Tuner: RandomSearch API*. Ultimo accesso: 2 agosto 2025. 2024. URL: https://keras.io/keras_tuner/api/tuners/random/ (cit. alle pp. 21, 62).

[35] J. Bergstra e Y. Bengio. «*Random Search for Hyper-Parameter Optimization*». In: *Journal of Machine Learning Research* 13.2 (2012), pp. 281–305 (cit. a p. 21).

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever e R. Salakhutdinov. «*Dropout: A Simple Way to Prevent Neural Networks from Overfitting*». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958 (cit. a p. 22).

[37] L. Prechelt. «*Early Stopping - But When?*» In: *Neural Networks: Tricks of the Trade*. A cura di G. Dreyfus e G. Orr. Also appeared as technical report TR-01-97, Fakultät für Informatik, Universität Karlsruhe, 1997. Springer, 1998, pp. 55–69 (cit. a p. 24).

[38] J. R. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, E. A. Chavez-Urbiola e J. A. Romero-Gonzalez. *Loss Functions and Metrics in Deep Learning*. arXiv preprint. 2024. URL: https://arxiv.org/abs/2307.02694 (cit. a p. 24).

[39] D. P. Kingma e J. Ba. «*Adam: A Method for Stochastic Optimization*». In: *arXiv preprint arXiv:1412.6980* (2015) (cit. a p. 24).

[40] Y. Bengio. «*Practical Recommendations for Gradient-Based Training of Deep Architectures*». In: *Neural Networks: Tricks of the Trade* (2012), pp. 437–478 (cit. a p. 25).

[41] Giorgio Franceschetti e Riccardo Lanari. *Synthetic Aperture Radar Processing*. Boca Raton, FL: CRC Press, 1999 (cit. a p. 28).

[42] Roland Bürgmann, Paul Rosen e Eric J. Fielding. «*Synthetic Aperture Radar Interferometry to Measure Earth's Surface Topography and Its Deformation*». In: *Annual Review of Earth and Planetary Sciences* 28 (2000), pp. 169–209. DOI: 10.1146/annurev.earth.28.1.169 (cit. a p. 29).

[43] Google. *Google Maps Street View di Via Flaminia Sud 9, Cantiano, Marche, Italia*. Ultimo accesso: 18 settembre 2025. 2024. URL: https://www.google.it/maps/place/43%C2%B028'12.2%22N+12%C2%B037'36.9%22E/@43.4700704,12.6267309,3a,75y,43.66h,73.41t/data=!3m8!1e1!3m6!1sES1uxRM-_QyTyycMzeQTRA!2e0!5s20110601T000000!6shttps:%2F%2Fstreetviewpixels-pa.googleapis.com%2Fv1%2Fthumbnail%3Fcb_client%3Dmaps_sv.tactile%26w%3D900%26h%3D600%26pitch%3D16.59134777407705%26panoid%3DES1uxRM-_QyTyycMzeQTRA%26yaw%3D43.66299909297969!7i13312!8i6656!4m4!3m3!8m2!3d43.4700556!4d12.6269167?entry=ttu&g_ep=EgoyMDI1MDkxNS4wIKXMDSoASAFQAw%3D%3D (cit. a p. 30).

[44] Google. *Google Maps Street View di Via Flaminia Sud 9, Cantiano, Marche, Italia*. Ultimo accesso: 18 settembre 2025. 2024. URL: https://www.google.it/maps/place/43%C2%B028'12.2%22N+12%C2%B037'36.9%22E/@43.4700462,12.6268465,3a,75y,60.98h,90t/data=!3m8!1e1!3m6!1sbEtrmQQOnQZ52yuiJ3TM6Q!2e0!5s20221001T000000!6shttps:%2F%2Fstreetviewpixels-pa.googleapis.com%2Fv1%2Fthumbnail%3Fcb_client%3Dmaps_sv.tactile%26w%3D900%26h%3D600%26pitch%3D0%26panoid%3DbEtrmQQOnQZ52yuiJ3TM6Q%26yaw%3D60.97652!7i16384!8i8192!4m4!3m3!8m2!3d43.4700556!4d12.6269167?entry=ttu&g_ep=EgoyMDI1MDkxNS4wIKXMDSoASAFQAw%3D%3D (cit. a p. 30).

[45] A. Rohatgi. *WebPlotDigitizer*. Ultimo accesso: 11 agosto 2025. 2025. URL: https://automeris.io/WebPlotDigitizer (cit. a p. 40).

[46] J. Reback et al. «*pandas-dev/pandas: Pandas*». In: *Zenodo* (2020). DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134 (cit. a p. 40).

[47] S. Gillies et al. *Shapely: Manipulation and Analysis of Geometric Objects*. 2025. URL: https://shapely.readthedocs.io/ (cit. a p. 51).

[48] C. R. Harris et al. «*Array Programming with NumPy*». In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2 (cit. a p. 63).