POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Matematica

Tesi di Laurea Magistrale

Deep Learning e CFD: sviluppo di modelli per la previsione della visibilità in scenari d'incendio in galleria

Relatore:
Prof. Luigi Preziosi

Candidata:
Ilaria Anselmi

Correlatrice:
Prof.ssa Maria Strazzullo

Tutor Aziendale:
Ing. Michele Fronterrè

Anno Accademico 2023-2024

Al Professor Piero Tempesta

Sommario

L'elaborato si propone di offrire un'analisi approfondita sull'evoluzione e l'integrazione dell'intelligenza artificiale (AI) nella progettazione antincendio basata sulle prestazioni (PBD), con particolare attenzione alle reti neurali convoluzionali trasposte e alla loro applicazione nella previsione della stratificazione dei fumi e della visibilità in scenari di incendio all'interno di gallerie stradali. Lo studio inizia con una panoramica sul ruolo della fluidodinamica computazionale (CFD) nella simulazione dei suddetti eventi critici e prosegue con una disamina dello sviluppo dell'AI in ambito PBD, con un approfondimento sullo stato dell'arte dell'ingegneria antincendio supportata dal deep learning. Viene quindi realizzata l'implementazione di simulazioni fluidodinamiche computazionali per la generazione di dati, utilizzati come base per l'addestramento del modello di AI. Successivamente, si passa alla costruzione dell'architettura della rete neurale, composta da strati convoluzionali trasposti e strati densi, e ci si concentra sul processo di ottimizzazione degli iperparametri tramite l'algoritmo Hyperband. Particolare attenzione viene dedicata alla scelta delle funzioni di loss, con una transizione dal Mean Squared Error (MSE) all'integrazione dello Structural Similarity Index (SSIM), al fine di migliorare la fedeltà strutturale delle immagini generate. L'analisi evidenzia i vantaggi di una loss combinata nel migliorare la qualità delle previsioni, soprattutto per scenari di incendio con modesti tassi di rilascio di calore (HRR). Vengono inoltre analizzate le immagini di output ottenute tramite l'impiego di matrici di confusione per valutare la capacità del modello di distinguere tra aree visibili e non visibili durante l'incendio, mostrando come l'ultimo modello sviluppato riduca i falsi positivi (FP) con maggiore efficacia di quanto siano in grado di fare quelli tradizionali. Infine, vengono presentati i risultati ottenuti dalle previsioni della rete in diversi scenari, con una stima accurata delle aree compromesse e dell'altezza dei fumi, offrendo un contributo pratico per ottimizzare le procedure di evacuazione, in particolare per quanto attiene al calcolo del tempo di uscita disponibile (ASET). Attraverso un rigoroso approccio sperimentale e l'applicazione di tecniche avanzate di deep learning, l'elaborato propone soluzioni innovative per migliorare la sicurezza nelle gallerie stradali, favorendo un sistema di progettazione antincendio più efficace e accurato, in un'ottica di prevenzione dei rischi per l'ambiente, gli utenti e gli impianti.

Indice

1 Introduzione

La necessità di incrementare le vie di comunicazione per trasportare merci e collegare, in tempi sempre più brevi, luoghi e persone, ha costretto gli esperti nell'ambito della realizzazione di infrastrutture a fare i conti con le caratteristiche morfologiche del nostro territorio. L'Italia, oggigiorno, presenta un ingente numero di gallerie, tunnel e trafori, molti dei quali non a norma di legge per quanto concerne la manutenzione di volte, manti stradali e il rispetto dei principali requisiti antincendio, come la ventilazione o gli impianti di estinzione ed aspirazione. Questa situazione ed i numerosi incidenti che si sono registrati nel tempo, come quello del Traforo del Monte Bianco - solo per citarne uno - hanno reso di stretta attualità il tema della sicurezza. Gli incendi nelle gallerie stradali, pur non essendo eventi estremamente frequenti, rappresentano un rischio significativo a causa della gravità delle loro conseguenze. Essi non derivano quasi mai da incidenti convenzionali, quali potrebbero essere lo scontro tra due veicoli o contro una parete: più frequentemente sono guasti meccanici, come problemi di carburazione, malfunzionamenti di cambio, ruote, freni o impianti elettrici, ad innescare le fiamme. Le condizioni ambientali delle gallerie, caratterizzate da spazi chiusi, visibilità limitata e vie di fuga complesse, ne aumentano rapidamente la pericolosità, con fumi densi, alte temperature e gas tossici, che possono risultare letali.
Il principale pericolo per chi si trova in una galleria in fiamme è rappresentato dal fumo, che riduce la percezione visiva e può rendere impossibile la fuga, pertanto, per avere un controllo efficace dell'incendio, bisogna avere una buona conoscenza della gestione dello stesso, che può essere ottenuta basandosi su un'approfondita comprensione del suo sviluppo e propagazione. La stratificazione dei fumi, ovvero la tendenza dei gas caldi a concentrarsi sopra quelli più freddi, in questo ambito è un fenomeno chiave da analizzare, poiché consente inizialmente una visibilità migliore nella zona sottostante, favorendo la messa in salvo di utenti e veicoli. Questo studio si focalizza sulla previsione di tale stratificazione per il calcolo del tempo a disposizione per la fuga (ASET), valutato attraverso la progettazione basata sulle prestazioni (PBD). L'uso di simulazioni tramite il Fire Dynamics Simulator (FDS) consente di riprodurre scenari realistici d'incendio, convalidati da esperimenti su larga scala che hanno dimostrato una buona corrispondenza tra misurazioni sperimentali e simulazioni, specialmente nella fase iniziale del pennacchio. Tuttavia, l'alto costo computazionale delle simulazioni di fluidodinamica computazionale (CFD) limita la possibilità di esplorare un numero elevato di scenari. Negli ultimi anni, questo campo ha iniziato a beneficiare dell'integrazione di metodi di deep learning, che hanno mostrato

la capacità di prevedere la stratificazione e l'evoluzione del fumo in diverse circostanze, ottenendo risultati comparabili a quelli delle simulazioni CFD, ma con tempi drasticamente ridotti. In tale ottica si intende esplorare l'uso delle reti neurali convoluzionali trasposte per prevedere la visibilità in galleria. In questa tesi sono stati simulati 72 scenari per una durata di 900 s ciascuno, variando parametri come l'Heat Release Rate (HRR), la geometria delle gallerie e la posizione d'incendio, utilizzando il software FDS. I fotogrammi ricavati dalle simulazioni ritraenti la visibilità tra 0 e 30 m, con una frequenza temporale di 10 s, sono stati utilizzati per allenare la rete al fine di renderla in grado di prevedere dinamiche di fumo in scenari non ancora presi in esame. I risultati sono stati analizzati per individuare zone critiche con visibilità inferiore ai 10 m, soglia minima necessaria per consentire la fuga degli utenti in galleria e per calcolare l'altezza dello strato di fumo o la distanza tra l'incendio e il momento in cui esso tocca il suolo, fondamentale per determinare l'ASET.
Gran parte della ricerca sulle reti neurali si è concentrata sulla definizione di nuove architetture mentre la funzione di loss, che è il vero motore dell'apprendimento nelle reti, ha ricevuto meno attenzione. La funzione di costo più comunemente utilizzata è la norma quadratica ℓ_2 dell'errore, come nel caso del Mean Squared Error (MSE). Tuttavia, studi del genere di quelli condotti da Wang et al. [26] dimostrano che l'integrazione di metriche strutturali, come lo Structural Similarity Index (SSIM) con una funzione ℓ_1 o ℓ_2, può portare a miglioramenti significativi anche senza modificare l'architettura della rete.
Nel presente elaborato, sono stati implementati due modelli di rete: uno che utilizza MSE come funzione di loss e uno che integra SSIM con l'obiettivo di studiare come l'uso combinato di queste metriche sia in grado di rilevare cambiamenti strutturali e migliori la capacità di catturare l'evoluzione del pennacchio di fumo e la stratificazione durante un incendio.
La tesi è strutturata in quattro capitoli principali. Nel primo, viene esplorata l'evoluzione della PBD e l'integrazione dell'Artificial Intelligence (AI). Si procede, quindi, nel secondo capitolo, ad esaminare il funzionamento delle reti neurali, per passare nel terzo a descrivere l'implementazione del modello AI nella simulazione degli incendi. Infine, nel quarto capitolo, vengono discussi i risultati ottenuti e le possibili applicazioni pratiche.

2 Integrazione dell'AI nella PBD

Questa parte esplora l'integrazione dell'AI nella PBD dell'ingegneria antincendio, evidenziandone il potenziale nel superare le limitazioni dei metodi tradizionali. La sezione 1.1 analizza la nascita e lo sviluppo della PBD, mettendo in luce le sfide e i benefici derivanti dall'uso di strumenti computazionali avanzati. La sezione 1.2 discute lo stato dell'arte nell'uso dell'AI per migliorare l'efficienza e l'accuratezza delle previsioni in scenari d'incendio. La sezione 1.3 esamina vantaggi e limiti delle reti neurali applicate alla sicurezza antincendio. Infine, la sezione 1.4 inquadra il ruolo del progettista AI all'interno del design concettuale dell'ingegneria antincendio.

2.1 La fluidodinamica computazionale

La CFD è una disciplina che impiega l'analisi numerica per studiare la dinamica dei fluidi, risolvendo le equazioni differenziali che ne governano il comportamento, note come *Equazioni di Navier-Stokes*. Queste equazioni descrivono come le sei variabili incognite del flusso — le tre componenti della velocità $\vec{u} = (u, v, w)$, la pressione p, la temperatura T e la densità ρ — variano all'interno di un fluido in movimento, in conformità ai principi fondamentali di conservazione della massa, della quantità di moto e dell'energia. Questo sistema di equazioni è fortemente accoppiato e caotico, il che implica che anche piccoli cambiamenti nelle condizioni iniziali possono condurre a traiettorie nello spazio delle soluzioni completamente differenti. Le equazioni di Navier-Stokes per un sistema termo-fluidodinamico sono quindi le seguenti:

Equazioni di conservazione della massa:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = \dot{m}'''$$

dove $\dot{m}'''$ rappresenta la sorgente di massa per unità di volume dovuta a processi fisici come evaporazione o reazioni chimiche;

Equazioni della quantità di moto:

$$\frac{\partial (\rho \vec{u})}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) = -\nabla p + \nabla \cdot \tau + \rho \vec{g} + \vec{f}$$

dove τ è il tensore degli sforzi viscosi e $\vec{g}$ è il vettore dell'accelerazione gravitazionale;

Equazione dell'energia:

$$\frac{\partial(\rho h)}{\partial t} + \nabla \cdot (\rho h \vec{u}) = \frac{Dp}{Dt} + \nabla \cdot (k \nabla T) + \dot{q} + \phi$$

dove h è l'entalpia specifica, k è la conducibilità termica, $\dot{q}$ è il tasso di rilascio del calore per unità di volume e ϕ rappresenta i termini di dissipazione viscosa.
L'obiettivo della CFD è risolvere numericamente questo complesso sistema di equazioni per generare una simulazione dettagliata delle variabili fisiche all'interno di un dominio di interesse. Quest'ultimo viene suddiviso in una griglia composta da celle o volumi di controllo, all'interno delle quali le equazioni differenziali vengono discretizzate in equazioni algebriche, più semplici da risolvere numericamente, permettendo di ottenere soluzioni approssimate delle variabili del flusso in ciascuna di esse.
Ogni simulazione CFD segue diverse fasi ben definite:

- creazione della mesh: la scelta del tipo di mesh (strutturata, non strutturata o adattativa) dipende dalla geometria del dominio e dalla complessità del fenomeno da simulare. La qualità della mesh è un elemento cruciale per garantire la precisione della simulazione: una griglia più dettagliata consente una maggiore risoluzione spaziale delle variabili fisiche in ogni punto del dominio, ma comporta anche un elevato costo computazionale, poiché il numero di celle da calcolare cresce esponenzialmente;

- definizione delle condizioni al contorno: una volta creata la griglia, è necessario impostare le condizioni iniziali e al contorno, che includono, rispettivamente, i valori delle variabili fisiche all'istante iniziale e sulle superfici del dominio. Questi parametri, come la pressione del fluido all'ingresso di un condotto o la temperatura delle pareti, sono fondamentali perché influenzano l'evoluzione delle variabili all'interno del dominio;

- iterazioni numeriche: in questa fase operativa, le equazioni vengono risolte iterativamente per ogni cella della griglia, aggiornando progressivamente le variabili fisiche fino a raggiungere la convergenza;

- post-elaborazione: consiste nell'analisi e visualizzazione dei risultati, utile per comprendere il comportamento del fluido e per estrarre informazioni con fini progettuali.

Uno dei principali vantaggi della CFD è la sua capacità di eseguire simulazioni per diverse condizioni al contorno in modo più agevole rispetto all'indagine

sperimentale. Grazie alla possibilità di simulare indipendentemente dal fattore di scala, si evitano i problemi che nascono quando si cerca di osservare i componenti e i parametri fluidodinamici su modelli reali, che spesso richiedono soluzioni costose e complesse. Questo permette di valutare il comportamento di elementi come i materiali da costruzione, le barriere antincendio e le persone in pericolo, in situazioni operative molto simili a quelle reali. L'incremento della potenza di calcolo, poi, ha reso possibile simulare scenari estremamente complessi, come il movimento del fumo durante un incendio in galleria, con un livello di precisione sempre maggiore, offrendo soluzioni progettuali ottimali in tempi ridotti.

CFD nella PBD Con l'entrata in vigore del D.M. 9 maggio 2007, denominato "Direttive per l'attuazione dell'Approccio Ingegneristico alla Sicurezza Antincendio", si è ufficialmente introdotto l'utilizzo della **Fire Safety Engineering** (Ingegneria della Sicurezza Antincendio) in ambito normativo, sancendo l'adozione dell'approccio prestazionale nella progettazione della sicurezza antincendio, che tradizionalmente era di tipo prescrittivo, ovvero le normative fissavano rigide caratteristiche del sistema, sufficienti a garantire la sicurezza, assicurando così la conformità a determinati standard. L'approccio PBD è invece basato su una logica prestazionale, per cui il progettista ha maggiore libertà nella scelta e nel dimensionamento degli impianti di sicurezza, purché dimostri in maniera quantitativa l'efficacia di tali sistemi nel ridurre il rischio. La PBD conta tra gli obiettivi principali quello di garantire che il sistema d'esodo progettato possa assicurare agli occupanti di raggiungere un luogo sicuro in sicurezza e di permanervi, senza incontrare gli effetti dell'incendio. Ciò però non è sempre possibile, soprattutto nel caso di infrastrutture come le gallerie stradali. In questo contesto, la CFD svolge un ruolo fondamentale, poiché permette ai progettisti di simulare scenari d'incendio complessi, valutando il comportamento dei flussi di fumo e calore e la loro interazione con le vie di esodo, configurandosi quindi come strumento essenziale per dimostrare l'efficacia delle soluzioni progettuali. Il progettista, variando accuratamente i parametri, può ottenere informazioni dettagliate e quantitative sull'evoluzione dell'incendio e determinare se le misure adottate garantiscono che gli occupanti possano raggiungere un luogo sicuro senza essere esposti agli effetti pericolosi dell'incendio. In tali contesti, nei pressi della fonte d'innesco, gli occupanti possono essere esposti agli effetti dell'incendio per un tempo che dipende dalla loro capacità di allontanarsi dalla sorgente, influenzando così l'efficacia del sistema di esodo stesso, da cui la necessità di definire un ASET per ogni casistica di interesse. La valutazione

della visibilità è fondamentale per la progettazione della sicurezza in scenari d'incendio. In particolare, la riduzione della visibilità influenza direttamente l'ASET, per cui un corretto calcolo della stessa consente di determinare i tempi critici entro cui è possibile evacuare in sicurezza un ambiente prima che la densità di fumo riduca eccessivamente la percezione visiva.
Uno dei software più utilizzati nel campo delle simulazioni CFD è il **FDS**, un software open-source sviluppato dal NIST (National Institute of Standards and Technologies) e basato su codici di calcolo implementati in linguaggio FORTRAN. FDS gestisce il calcolo della visibilità in funzione della densità ottica D del fumo, che esprime la capacità di un mezzo di assorbire e disperdere la luce che lo attraversa. In altre parole, D misura quanta luce viene attenuata per unità di distanza nel mezzo ed è direttamente correlata alla quantità di particelle di fumo presenti nell'aria dalla relazione:

$$D = \frac{\kappa m_f}{\rho},$$

dove:

- κ rappresenta il coefficiente di estinzione specifico del fumo, legato alla capacità del fumo di assorbire e disperdere la luce;
- m_f è la massa di fumo per unità di volume;
- ρ è la densità dell'aria.

La riduzione dell'intensità della luce $I(x)$ che avviene mentre questa attraversa un mezzo con densità ottica D è espressa dalla legge di Lambert-Beer [18]:

$$I(x) = I_0 e^{-\kappa x},$$

dove:

- I_0 è l'intensità della luce incidente (prima che attraversi il fumo);
- κ è il coefficiente di estinzione, legato alla densità ottica del fumo;
- x è la distanza percorsa dalla luce nel mezzo.

È quindi chiaro che maggiore è la densità ottica del fumo, più rapidamente la luce viene attenuata quando lo attraversa. In termini di visibilità, ossia la distanza alla quale un oggetto può essere appena distinguibile attraverso il fumo, il modello utilizza la seguente relazione:

$$V = \frac{3}{\kappa}.$$

Questa formula implica che la visibilità V è inversamente proporzionale al coefficiente di estinzione κ. Poiché κ varia da un punto all'altro nel dominio, anche la visibilità V varia di conseguenza. Il fattore 3 si basa su uno standard che considera la soglia di trasmittanza luminosa del 5% come criterio per la visibilità minima.
Il coefficiente di estinzione κ varia in base alla composizione del fumo e alle caratteristiche del combustibile che brucia. In FDS, questo parametro può essere impostato in funzione del materiale coinvolto nell'incendio, permettendo di simulare accuratamente l'effetto del fumo sulla visibilità in scenari specifici. Una corretta scelta di κ è essenziale per ottenere una simulazione fedele del comportamento del fumo e della conseguente riduzione della visibilità. Durante una simulazione, FDS aggiorna continuamente i valori della densità ottica del fumo in ciascuna cella della griglia in base alla produzione di fumo derivante dall'HRR, alla dispersione del fumo attraverso i flussi d'aria e alla sedimentazione delle particelle. I risultati della simulazione sono espressi in termini di visibilità media per ciascuna cella della griglia, permettendo di valutare come la visibilità vari nel tempo e nello spazio durante lo sviluppo dell'incendio.

2.2 Nascita e sviluppo dell'AI nella PBD

L'integrazione della PBD nell'ingegneria antincendio ha beneficiato dell'avanzamento di strumenti di ingegneria computazionale sofisticati, come il Computer-aided design (CAD), la CFD e i modelli di zona. Questi strumenti, che utilizzano modelli fisici validati tramite dati sperimentali per analizzare e prevedere i comportamenti del fuoco, sono ormai considerati standard nell'industria e sono impiegati dagli ingegneri antincendio per valutare i sistemi di controllo del fumo e le strategie di evacuazione. Malgrado ciò, l'uso di taluni presenta notevoli sfide a causa della complessità degli scenari e della conseguente necessità di definire configurazioni inedite nei software di simulazione. Nonostante la PBD sia stata ampiamente applicata per la sicurezza antincendio degli edifici nel 21° secolo, infatti, essa presenta ancora due problemi significativi [1]:

- alti costi e tempi lunghi: ogni caso di PBD richiede ingenti quantità di tempo e risorse umane per i processi di progettazione, documentazione, revisione e approvazione;
- validazione dei modelli: i modelli numerici utilizzati sono validati sperimentalmente su un set specifico di configurazioni. L'utilizzo di tali modelli in condizioni molto diverse da quelle sperimentali solleva dubbi sull'affidabilità dei risultati.

I modelli CFD, in particolare, sono comunemente impiegati nell'ingegneria della protezione antincendio per prevedere flussi complessi utili al controllo del fumo. L'alta risoluzione spazio-temporale richiesta, però, comporta significativi costi computazionali, con tempi di simulazione che possono estendersi per ore, giorni o addirittura settimane, a seconda della complessità della stessa. Per questo motivo, il costo per riprodurre grandi domini o condurre studi parametrici può risultare proibitivo. Per risolvere simili problemi, lo sviluppo emergente dell'intelligenza artificiale, come strumento complementare alla PBD, offre nuovi approcci e soluzioni progettuali sia per i progettisti che per le autorità competenti.
L'integrazione dell'AI sugli scenari d'incendio viene spesso messa in pratica a posteriori rispetto alle simulazioni CFD, allenandola sugli scenari generati dai software di risoluzione numerica per insegnarle a riconoscere dinamiche tipiche come la stratificazione dei fumi e prevederne il comportamento una volta che si varino i parametri che definiscono una configurazione.
Più nel dettaglio l'applicazione degli strumenti di AI nella PBD segue tre fasi principali:

- **Pre-elaborazione e input**: in questa fase vengono selezionati i fattori chiave di input, come l'HRR, le dimensioni strutturali ed eventuali tipologie di impianti per ventilazione del fumo, come verrà meglio esposto nel prossimo capitolo. Analogamente alla PBD convenzionale, è possibile fare alcune approssimazioni, come la geometria e il tasso di crescita dell'incendio, con lo scopo di semplificare la modellazione iniziale;

- **Previsione AI**: utilizzando le informazioni di input fornite, il modello di AI, opportunamente addestrato, è in grado di prevedere lo sviluppo del movimento del fumo e il profilo di visibilità durante l'incendio in una scala temporale di pochi minuti, rendendo il processo estremamente rapido ed efficiente;

- **Output e applicazione**: grazie alle immagini di visibilità riprodotte, il progettista può quantificare l'altezza dei fumi con un grado di approssimazione legato alla qualità delle stesse.

Nonostante il tempo necessario per la generazione dei dati, la progettazione e l'addestramento della rete sia considerevole , una volta completata questa fase, un modello generativo è capace di fornire previsioni su nuovi scenari in modo rapido. Questo metodo elimina il lungo e costoso processo di prova e correzione tipico della modellazione CFD per ogni caso di esame e fa in modo che le autorità competenti possano utilizzare lo strumento AI

per verificare velocemente l'affidabilità della PBD tradizionale senza dover eseguire processi di modellazione antincendio CFD simili. Di conseguenza, il processo di progettazione e revisione può essere significativamente ridotto, passando da mesi, come avviene con la PBD tradizionale, a poche ore o addirittura minuti con l'approccio basato sull'AI.
Per tali motivi, la modellazione generativa è particolarmente promettente nell'analisi dei rischi per grandi strutture con topologie ripetitive, come sistemi di tunnels, impianti di stoccaggio sotterranei e miniere, poiché consente di fornire valutazioni preliminari per gli ingegneri antincendio e riduce il numero di simulazioni necessarie per comprendere le prestazioni di sicurezza, giudicare la validità dei risultati delle simulazioni e migliorare l'efficienza lavorativa.

2.3 Stato dell'arte dell'ingegneria antincendio basata sull'AI

Negli ultimi anni, come già accennato, l'applicazione dell'intelligenza artificiale nell'ingegneria antincendio ha visto significativi progressi, specialmente nel rilevamento e previsione dei profili di fumo e temperatura. Diversi studi hanno esplorato l'uso di modelli di apprendimento automatico e reti neurali per migliorare l'efficacia e la precisione delle previsioni durante gli incendi. Questo sezione esamina lo stato attuale dei progetti sull'AI focalizzati su questi aspetti, citando i principali autori e le loro ricerche.
Lo studio di modelli di previsioni del fumo e della temperatura è già stato oggetto d'indagine dai primi anni 2000. Ricerche come quella a carico di Kurioka et al. [11], i quali hanno condotto un test su piccola scala utilizzando come banco di prova un tunnel in miniatura e hanno proposto una correlazione empirica per la temperatura massima del soffitto e l'HRR, hanno gettato le basi per ulteriori approfondimenti sull'analisi delle temperature durante gli incendi in galleria. A seguire infatti, Li et al. [17] hanno indagato la relazione che lega l'aumento della temperatura del fumo e HRR sotto varie condizioni di ventilazione trasversale, monitorando il tasso di ventilazione adimensionale. Questo studio ha fornito una comprensione più approfondita delle dinamiche del fumo e del calore in presenza di vento. Sulla loro scia, nel 2013 , Wei et al. [6] hanno esaminato gli effetti dei fattori delle pareti sulla distribuzione della temperatura delle volte negli incendi in galleria, concentrandosi sull'influenza delle componenti strutturali sulla propagazione del calore e del fumo. Con la maggior conoscenza ed il sempre più massivo utilizzo delle tecniche di smart-firefighting basate su algoritmi di AI, Internet of Things (IoT) e sensori, negli ultimi dieci anni sono

arrivate anche le prime importanti soluzioni che contemplano l'integrazione dell'AI con modelli empirici e simulazioni CFD. Hodges et al. [8] hanno utilizzato una rete neurale convoluzionale trasposta (TCNN) e i risultati delle simulazioni condotte con il FDS per prevedere la distribuzione della temperatura all'interno delle stanze dei compartimenti. Questo approccio ha dimostrato la capacità delle reti neurali di migliorare la precisione delle previsioni termiche. Naser et al. [20] hanno costruito un database di test antincendio su strutture in legno e previsto la resistenza al fuoco di quest'ultime tramite un modello di AI. Il modello allenato su questi dati ha previsto la posizione e le dimensioni della fonte d'incendio e gli eventi critici (ad esempio, la velocità di ventilazione critica), ma ha anche abilitato la previsione in tempo reale avanzata dell'evoluzione futura dei campi di incendio. Zeng et al. [1] hanno sviluppato l'Intelligent Fire Engineering Tool (IFETool), un software di progettazione antincendio basato sull' AI, per velocizzare l'analisi della sicurezza antincendio e identificare rapidamente i limiti di progettazione. Gli autori hanno creato un ampio database numerico di incendi in atri e addestrato un modello di deep learning per prevedere l'evoluzione della visibilità del fumo, della temperatura e della concentrazione di CO con un'accuratezza del 97%. Questa tesi trae ispirazione direttamente dai risultati e dai metodi sviluppati da [1].

2.4 Vantaggi e limiti delle reti neurali

Le reti neurali sono capaci di apprendere e modellare relazioni complesse e non lineari nei dati, rendendole estremamente versatili per una vasta gamma di applicazioni, che spaziano dal settore finanziario a quello medico, dalla robotica al riconoscimento di immagini e voce.
La loro natura distribuita conferisce particolare resistenza agli errori e al rumore nei dati di input: in una rete neurale, l'informazione non è centralizzata ma diffusa attraverso interi layer. Il gran numero di neuroni interconnessi che la compongono crea questa forma di ridondanza: le informazioni o i pattern appresi sono dipendenti non da un singolo neurone, ma piuttosto parte di una funzione collettiva. Questa struttura consente un'alta tolleranza ai guasti: se un neurone fallisce o viene meno, il carico che esso gestiva può essere redistribuito agli altri neuroni nella rete, assicurando che l'elaborazione continui senza interruzioni significative. Dopo un adeguato addestramento, le reti neurali possono generalizzare da dati precedentemente non visti, eseguendo predizioni e classificazioni in tempo reale e prevedendo risultati accurati anche su nuovi set di dati, il che le rende strumenti preziosi per applicazioni che richiedono scalabilità e velocità.
Nonostante l'efficacia dei modelli prodotti, è importante riconoscere che le

reti neurali si comportano come “black box”, ovvero non offrono spiegazioni chiare sul perché specifiche decisioni o predizioni sono state fatte.
Inoltre, per un efficace apprendimento supervisionato, richiedono grandi quantità di dati etichettati per l’addestramento affinchè apprendano automaticamente le caratteristiche nascoste al loro interno, il che può risultare costoso e impegnativo. Questa condizione amplifica la complessità della loro implementazione, non solo in termini di raccolta dati ma anche per quanto riguarda la loro gestione e preparazione. Tutti i modelli di AI esistenti devono essere perfezionati prima di essere applicati per risolvere problemi concreti, spesso attraverso un cospicuo numero di iterazioni di addestramento su un grande database. Pertanto, un database antincendio ben strutturato è anche un prerequisito per le applicazioni di AI nell’ingegneria antincendio.
Senza una corretta regolazione e validazione, inoltre, c’è il rischio che le reti neurali si adattino eccessivamente ai dati di addestramento, perdendo così la capacità di generalizzare. Questo fenomeno, noto come “overfitting”, può compromettere l’utilità del modello quando esposto a nuovi scenari.
Per di più, i risultati ottenuti dalle reti neurali possono variare significativamente a seconda della scelta dei parametri di addestramento, come il tasso di apprendimento, il numero di epoche e la dimensione del batch. La sensibilità a tali parametri richiede un’attenzione scrupolosa per assicurare che il modello non solo performi bene sui dati di test ma sia anche stabile e affidabile sotto diverse condizioni operative, come vedremo nella sezione 3.

2.5 Design concettuale della sicurezza antincendio

L’ultimo lavoro sviluppato da Huang et al. [1] ha dimostrato la fattibilità di adottare un algoritmo di deep learning e un database CFD di incendi preesistente per prevedere il movimento del fumo all’interno di un semplice atrio e stimare l’ASET in pochi secondi. Lo schema presente (Figura 6) mostra come lo strumento di intelligenza artificiale si inserisce nel design della progettazione antincendio per l’ Authority Having Jurisdiction (AHJ) nella rapida valutazione dell’affidabilità della PBD, nella revisione delle simulazioni CFD di incendi e nell’individuazione dei problemi.

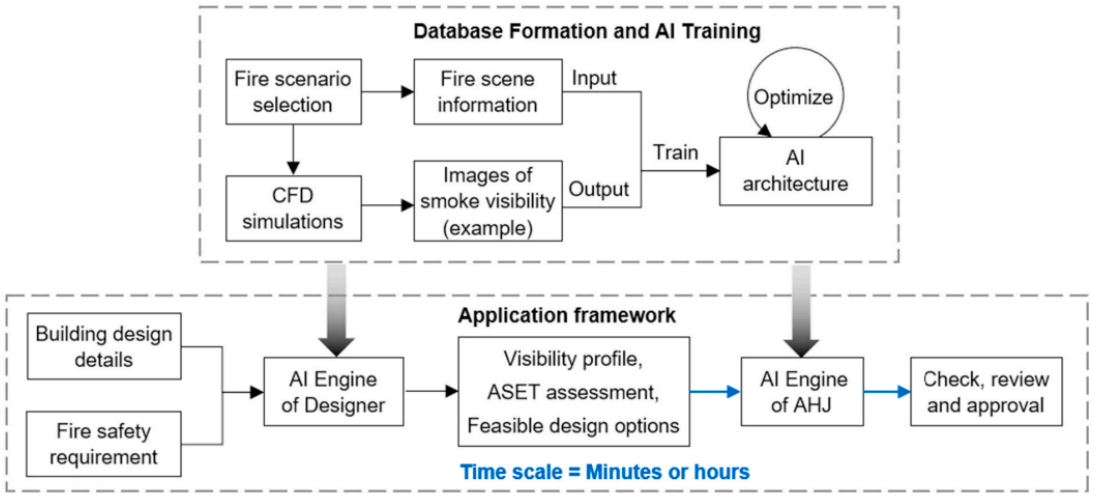


Figura 1: Database formation and AI training. Immagine tratta da [1].

1. **Formazione del database e addestramento dell'AI**:

   (a) Selezione dello scenario d'incendio:
   si scelgono scenari d'incendio specifici che rappresentano possibili situazioni di emergenza;

   (b) Informazioni sulla scena dell'incendio:
   si raccolgono le informazioni di interesse sugli scenari d'incendio selezionati;

   (c) Simulazioni CFD:
   si eseguono simulazioni CFD per analizzare il comportamento del fuoco e del fumo negli scenari selezionati;

   (d) Immagini di visibilità del fumo:
   le simulazioni CFD generano immagini che rappresentano la visibilità del fumo. Queste immagini vengono utilizzate come etichette per l'addestramento dell'architettura AI;

   (e) Ottimizzazione:
   l'architettura AI viene addestrata con i dati in ingresso (informazioni sugli scenari d'incendio) e le etichette (immagini di visibilità del fumo) e viene ottimizzata per migliorare le sue prestazioni;

   (f) Dettagli della progettazione dell'edificio e requisiti di sicurezza antincendio:
   si forniscono i dettagli della progettazione dell'edificio e i requisiti di sicurezza antincendio;

2. **Processo di verifica e convalida da parte dell'Autorità competente**

   (a) Ricezione dei risultati dal progettista:
   il progettista utilizza il proprio motore AI per generare profili di visibilità, proporre opzioni di progettazione basate sui dettagli del design dell'edificio e sui requisiti di sicurezza antincendio. Questi risultati vengono quindi inviati all'autorità competente per la revisione;

   (b) Verifica e revisione:
   L'AHJ utilizza il proprio motore AI per simulare il comportamento del fumo e la visibilità negli stessi scenari d'incendio elaborati dal progettista. Confrontando le immagini generate dal suo AI con quelle prodotte dal progettista, l'AHJ verifica se i risultati siano allineati agli standard di sicurezza. Questo passaggio è fondamentale per garantire che le soluzioni progettuali rispettino i criteri di sicurezza, poiché consente una validazione incrociata: il progettista propone una soluzione, e l'AHJ, attraverso il test del proprio modello AI, valuta se i risultati ottenuti siano conformi alle normative e ai requisiti di sicurezza stabiliti, permettendo di individuare eventuali discrepanze o problemi nelle proposte del progettista e fornire feedback;

   (c) Controllo e approvazione:
   Una volta che l'AI dell'AHJ ha verificato e convalidato i risultati, l'autorità competente effettua un controllo finale; se tutto è conforme ai requisiti, l'AHJ approva il progetto, se ci sono problemi o la proposta non soddisfa i criteri, l'AHJ può richiedere ulteriori modifiche o chiarimenti al progettista.

Il progetto sviluppato nell'ambito di questa tesi si focalizza sulla fase iniziale di questo processo collaborativo, con particolare attenzione alla creazione del dataset, all'addestramento e alla validazione del modello AI, fondamentali per supportare le autorità competenti nella valutazione e approvazione delle progettazioni antincendio basate su prestazioni.

3 Reti neurali nella Computer Vision

Questa sezione esamina l'applicazione delle reti neurali nel campo della Computer Vision, concentrandosi sulle loro architetture e sul processo di addestramento, nonché sull'utilizzo di specifiche funzioni di perdita per migliorare la qualità delle immagini generate. La Sezione 2.1. approfondisce l'importanza della suddivisione del dataset in training, validation e test set, evidenziando come ciascuno di questi contribuisca al successo dell'addestramento e alla capacità di generalizzazione del modello. La Sezione 2.2 esplora la struttura delle reti neurali, descrivendo il modello di neurone artificiale e la rete multistrato di tipo Fully Connected Neural Networks. La Sezione 2.3 discute in dettaglio il processo di addestramento delle reti neurali, suddiviso in *feedforward propagation* e *backpropagation*, con specifica sulle tecniche di ottimizzazione di Stochastic Gradient Descent e Adam Optimizer. Le Sezioni 2.4 e 2.5 affrontano i due principali problemi riscontrabili durante l'addestramento della rete: il *vanishing and exploding gradient* e l'*overfitting* e le tecniche di regolarizzazione adottabili, come il *dropout*. Infine, la Sezione 2.6. analizza le funzioni di perdita utilizzate nella Computer Vision, con particolare attenzione al *Mean Squared Error* e allo *Structural Similarity Index*, discutendo i loro vantaggi, limitazioni e applicazioni specifiche.

3.1 Training, Validation e Test Set

Nell'ambito delle reti neurali per la generazione di immagini, i dati rappresentano una pietra angolare per lo sviluppo e la valutazione di un modello, poiché costituiscono la base su cui esso impara a riconoscere pattern, relazioni e caratteristiche specifiche che gli permettono di generare nuovi campioni con precisione e coerenza. Per garantire che una rete non solo apprenda efficacemente dai dati, ma sia anche in grado di generalizzare su nuovi scenari, il dataset viene suddiviso in tre distinti sottoinsiemi: il set di addestramento, il set di validazione e il set di test.

Training set Le reti neurali vengono addestrate attraverso un processo iterativo in cui osservano migliaia o milioni di esempi, dai quali apprendono pattern, relazioni e strutture presenti nei dati. Il training set è tipicamente il più grande dei tre sottoinsiemi, poiché è cruciale che il modello abbia a disposizione una quantità sufficiente di dati per un apprendimento efficace. I campioni che lo compongono permettono al modello di comprendere come certi elementi visivi si correlano tra loro, come vengono distribuiti e come variano in presenza di condizioni diverse, come ad esempio l'evoluzione della

distribuzione dei fumi cambiando le dimensioni geometriche della galleria. Perché una rete neurale sia utile in applicazioni reali, deve essere in grado di generalizzare, ovvero fare previsioni accurate su nuovi dati che non ha mai visto durante l'addestramento. Affinché ciò accada, il training set deve essere sufficientemente diversificato per evitare che il modello si limiti a memorizzare i dati, ma piuttosto impari a riconoscere pattern e regole sottostanti che gli permettano di adattarsi bene a nuovi scenari.

Validation set Mentre il modello viene addestrato sul training set, il validation set viene utilizzato per monitorare le sue prestazioni su dati non visti durante l'addestramento, fornendo una misura di come il modello potrebbe comportarsi su nuovi dati. Questo set ha un ruolo cruciale nel prevenire il sovradattamento del modello ai dati, assicurando che mantenga la sua capacità di generalizzazione.

Test set Infine, il set di test fornisce una valutazione imparziale della capacità della rete di generalizzare su dati completamente nuovi, garantendo che il modello possa fare previsioni accurate in condizioni diverse da quelle viste durante l'addestramento. Questo set viene utilizzato una volta completato l'addestramento per ottenere una misura finale delle prestazioni del modello.

3.2 Modello di neurone artificiale e rete multistrato

Le reti neurali sono modelli di elaborazione dati ispirati al funzionamento dei neuroni biologici nel cervello umano. Progettate per simulare il modo in cui il cervello processa le informazioni, queste architetture sono particolarmente efficaci per l'apprendimento automatico e l'analisi di dati complessi.
Una rete neurale è composta da numerose unità interconnesse, chiamate neuroni, ciascuno dei quali è collegato a molti altri tramite connessioni ponderate. Il funzionamento di ciascun neurone può essere descritto come segue: riceve un vettore di input $\mathbf{x} \in R^m$, esegue un prodotto scalare con i suoi pesi $\mathbf{w} \in R^m$, aggiunge un termine di bias b e infine applica una funzione di attivazione al risultato:

$$\begin{aligned} z &= \mathbf{w} \cdot \mathbf{x} + b, \\ h &= \mathrm{f}(z). \end{aligned} \tag{1}$$

La funzione di attivazione ha il compito di introdurre non linearità e limitare l'uscita del neurone a un intervallo specifico. Tra le funzioni di attivazione

più comuni nell'elaborazione delle immagini troviamo *Tanh*, *Sigmoid* e *ReLU*. I pesi e il bias di ciascun neurone, inizialmente impostati a valori casuali molto vicini allo zero, vengono appresi durante la fase di addestramento. I neuroni artificiali si distribuiscono su una struttura organizzata in layer $[l_1, \ldots, l_n] \in L$. Ogni rete è caratterizzata da un *input layer* l_1, che raccoglie i neuroni di ingresso, un *output layer* l_n, che presenta i risultati in base al compito assegnato, e da eventuali *hidden layers*, attraverso i quali si propaga l'informazione. Gli strati sono interconnessi in modo tale che l'output di un layer l_i diventi l'input del layer successivo l_{i+1}.
Nel caso più standard di reti neurali completamente connesse (*Fully Connected Neural Networks*), ogni neurone di un hidden layer è collegato a tutti i neuroni del layer precedente e successivo, come mostrato in Figura 2. Specificamente, ogni neurone negli hidden layers riceve inputs da vari altri neuroni e, attraverso il processo di pesatura, somma e attivazione, trasmette un output agli strati successivi.

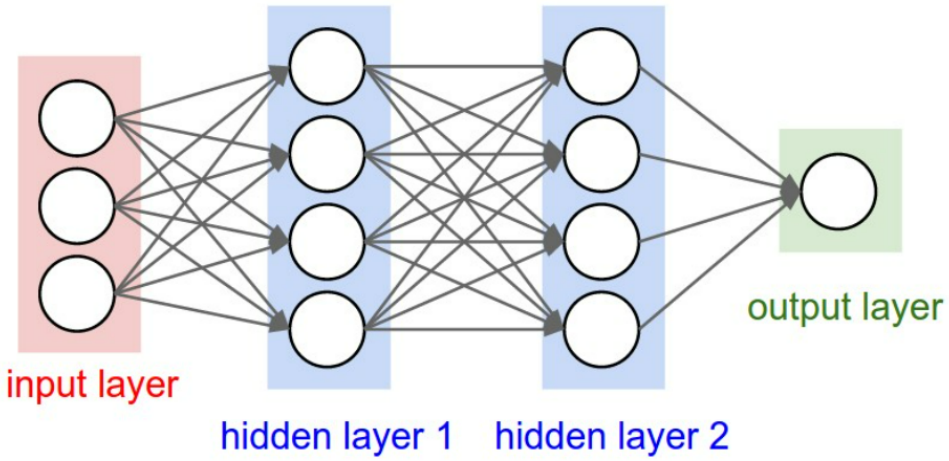


Figura 2: Rete densa a tre strati. Immagine tratta da [13].

3.3 Training

Durante il training supervisionato, la rete impara ad adattare i pesi e i bias per minimizzare la differenza tra l'output previsto e quello noto, divenendo in grado di prevedere nuovi dati non ancora etichettati. Questo processo di apprendimento avviene alternando due fasi: la feedforward propagation e la backpropagation.

3.3.1 Feedforward propagation

Durante la fase di *feedforward*, i dati di input vengono trasformati attraverso la rete, passando per vari strati fino a raggiungere l'output layer. Questo strato finale produce i risultati in base al task specifico per cui la rete è stata addestrata. L'output prodotto viene valutato tramite una funzione di perdita (*loss function*) che stima quanto si discosta da quello desiderato.

Dense layer I *dense layers* costituiscono la base delle Fully Connected Neural Networks, in cui ogni neurone è collegato a tutti i neuroni dei layer precedenti e successivi, trasformando i dati di input secondo il procedimento mostrato nell'equazione (1). Questa architettura permette alla rete di apprendere relazioni complesse tra i dati, sfruttando al massimo la capacità computazionale per modellare in modo efficace input complessi e non lineari. Il processo di feedforward propagation all'interno di un dense layer è uno degli approcci più comuni nell'elaborazione delle informazioni in una rete neurale. In questo contesto, lo strato fully connected ha un solo iperparametro da configurare: il numero di neuroni K. Questo parametro determina la dimensione dell'output, che nel caso di questo modello è un singolo vettore di dimensione $1 \times 1 \times K$.
Il seguente algoritmo descrive il processo di propagazione in una rete composta unicamente da dense layers:

Algorithm 1 Feedforward propagation in a fully connected neural network

Require: $\mathbf{x}$: Input vector
1: $\hat{\mathbf{h}}^0 \leftarrow \mathbf{x}$
2: **for** $l = 1$ **to** $N - 1$ **do**
3: $\quad \mathbf{z}^l \leftarrow \mathbf{W}^l \hat{\mathbf{h}}^{l-1} + \mathbf{b}^l$
4: $\quad \hat{\mathbf{h}}^l \leftarrow f(\mathbf{z}^l)$
5: $\hat{\mathbf{y}}^N \leftarrow \mathbf{W}^N \hat{\mathbf{h}}^{N-1} + \mathbf{b}^N$
6: **return** $\hat{\mathbf{y}}^N$

dove N è il numero totale di layer che compone la rete.

Convolutional layer In una rete neurale un *convolutional layer* è essenziale per l'estrazione delle caratteristiche dall'input, che di solito è un'immagine rappresentata come matrice di interi che dispone i neuroni in tre dimensioni: larghezza, altezza e profondità (espressa in termini di canali). Esso è costituito da un insieme di filtri apprendibili (detti anche *kernel*), che

sono matrici tridimensionali con una larghezza e un'altezza ridotte, ma con una profondità uguale a quella del volume di input. Ad esempio, dato un input di dimensioni $128 \times 128 \times 3$, i filtri hanno la forma $k \times h \times 3$, dove i valori tipici per h e k variano tra 1 e 7 e sono solitamente di uguale valore $h = k$.
Il layer è responsabile dell'applicazione di questi filtri all'immagine tramite un processo convolutivo in cui il kernel viene spostato sistematicamente sui dati di input e, ad ogni posizione, esegue un'operazione di moltiplicazione elemento per elemento (element-wise) seguita da una somma. Il risultato di questa operazione è un singolo valore di output, che contribuisce a formare una mappa di attivazione bidimensionale, comunemente chiamata *feature map*.
I pesi del filtro, che vengono appresi durante l'addestramento, rappresentano la struttura o la caratteristica che il filtro rileverà. La rete che contiene un convolutional layer, quindi, inizia con pesi casuali, che attraverso l'addestramento vengono gradualmente aggiustati in modo tale da migliorare la sua capacità di identificare le correlazioni tra i pixel.
Gli iperparametri che devono essere fissati a priori sono: la dimensione del kernel, lo stride e il padding.
La **grandezza del kernel** determina quanto la dimensione dell'output è ridotta rispetto all'input: più è grande, più piccole saranno le dimensioni della mappa di funzionalità, con implicazioni dirette su quanta parte dell'immagine originale viene aggregata in ogni singolo valore di uscita. Se il kernel ha dimensioni ridotte, ogni valore nell'output rappresenta informazioni da un piccolo gruppo di pixel dell'immagine originale, quindi la mappa delle funzionalità risultante sarà sensibile ai dettagli locali, come bordi o texture fini. Un filtro di dimensioni maggiori, invece, copre un'area più vasta dell'immagine a ogni passaggio, il che implica che viene generato un numero inferiore di valori di output, poiché ogni valore rappresenta una porzione più ampia dell'immagine originale, riuscendo a catturare meglio correlazioni a lungo raggio.
Quando si applica un kernel di dimensione $k \times k$ su un input di dimensione $n \times n$ senza l'utilizzo di padding, la dimensione dell'output per lato è determinata dalla seguente formula:

$$\text{dim output} = n - k + 1.$$

Il numero di filtri utilizzati, invece, influisce sulla profondità dell'output. Più filtri sono utilizzati, maggiore sarà la varietà delle caratteristiche estratte, poiché ogni filtro contribuirà con una mappa di funzionalità unica, risultando in una rappresentazione più completa e diversificata dell'immagine di input.

Il passo di spostamento del filtro sull'immagine è definito dallo **stride**, che stabilisce di quanti pixel il filtro si sposta sull'immagine ad ogni applicazione successiva sia lungo l'asse dell'altezza sia lungo la larghezza. Strides maggiori riducono la dimensione della mappa delle funzionalità poiché il filtro coprirà l'immagine con meno applicazioni, mentre strides minori consentono un'analisi più dettagliata.
Un altro fenomeno di cui bisogna tener conto durante l'applicazione del filtro sono gli effetti di bordo che vengono gestiti attraverso il **padding**. Per impostazione predefinita, un filtro inizia a sinistra dell'immagine, con il suo lato sinistro posizionato sui pixel dell'estremo sinistro dell'immagine. Viene poi spostato una colonna alla volta finché il lato destro del filtro non raggiunge i pixel dell'estremo destro dell'immagine, per poi iterare il procedimento alla riga sottostante. Questo processo implica che i pixel ai bordi dell'immagine non possano contribuire pienamente alla mappa delle funzionalità [3]. Ciò si risolve aggiungendo pixel di valore zero ai bordi, una tecnica che viene utilizzata soprattutto in forma di *same padding*, per assicurare che l'output abbia la stessa dimensione dell'input quando lo stride s è 1. La dimensione dell'output per lato in tale caso è calcolabile come:

$$\text{dim output} = \lfloor \frac{\text{n}}{\text{s}} \rfloor. \tag{2}$$

Questo approccio assicura che ogni pixel abbia l'opportunità di trovarsi al centro del filtro e di non alterare il calcolo del prodotto scalare durante l'applicazione, dal momento che i pixel aggiunti col padding hanno valore nullo.
Per comprendere meglio il funzionamento di questo tipo di layer è necessario far riferimento a un esempio operativo: l'immagine 3 mostra come un filtro convoluzionale venga utilizzato per processare l'input, producendo una feature map.

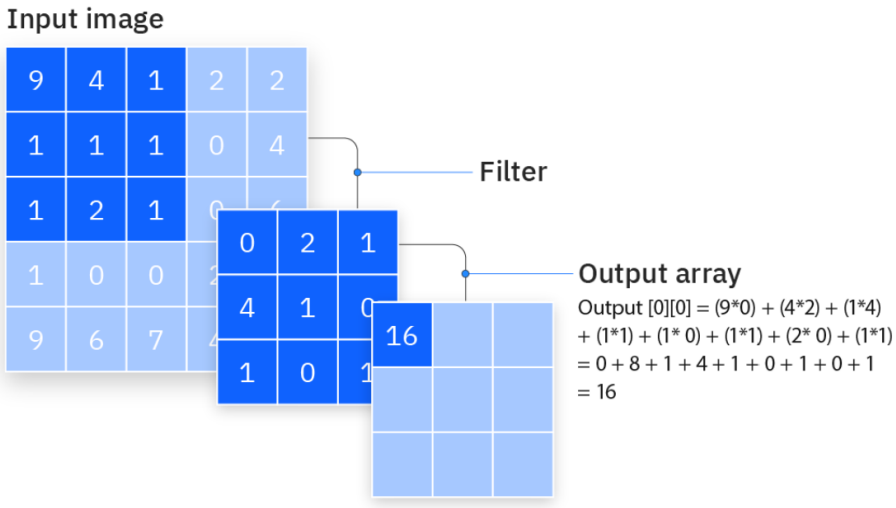


Figura 3: Convolutional Process. Immagine tratta da [12].

$$\text{Matrice di Input} = \begin{bmatrix} 9 & 4 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 4 \\ 1 & 2 & 1 & 0 & 4 \\ 1 & 0 & 0 & 2 & 4 \\ 9 & 6 & 7 & 4 & 2 \end{bmatrix}, \quad \text{Filtro} = \begin{bmatrix} 0 & 2 & 1 \\ 4 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Non viene applicato alcun padding, quindi l'output è direttamente influenzato dalla dimensione del filtro e dall'input, assumendo dimensioni pari a:

$$\text{dim output} = (5 - 3) + 1 = 3, \text{ quindi sarà una matrice } 3 \times 3.$$

Il filtro viene applicato tramite l'operazione di convoluzione alla prima porzione della matrice di input:

$$\text{output}[0][0] = (9 \times 0) + (4 \times 2) + (1 \times 1) + (1 \times 4) + (1 \times 1) + ... = 16.$$

In seguito viene spostato di una colonna a destra e applicato alla porzione successiva della matrice:

$$\text{Output}[0][1] = (4 \times 0) + (1 \times 2) + (2 \times 1) + (1 \times 4) + (0 \times 1) + ... = 14.$$

Il processo viene iterato per le restanti porzioni, fino a completamento dell'immagine.

In sintesi, lo strato convoluzionale è responsabile dell'estrazione delle caratteristiche tramite l'applicazione di uno o più filtri a seconda del numero di canali dell'input, che vengono spostati sull'immagine con uno stride definito, e, se necessario, con l'aggiunta di padding per gestire i bordi dell'immagine. Questo processo, insieme all'uso di funzioni di attivazione nonlineari, consente alle reti che utilizzano layer di questo tipo, come le *Convolutional Neural Networks* (CNNs), di eseguire compiti di classificazione delle immagini e riconoscimento degli oggetti in modo efficiente e scalabile, superando i metodi manuali di estrazione delle caratteristiche utilizzati in passato.

Transposed convolutional layer Il *transposed convolutional layer* è anche erroneamente noto come deconvolutional layer. Quest'ultimo inverte il funzionamento di un convolutional layer: se l'output generato attraverso uno strato convoluzionale standard viene deconvoluto, si ottiene indietro l'input originale. Tuttavia, la convoluzione trasposta non inverte la convoluzione standard in base ai valori, bensì solo alle dimensioni.
Nel processo, il kernel viene applicato a ciascun pixel dell'input e i risultati vengono posizionati e sommati nella matrice di output, permettendo una ricostruzione spazialmente espansa dell'immagine originale. Successivamente, si combinano tutte queste matrici insieme secondo le posizioni iniziali nel livello di input, sommando i valori sovrapposti.
Le dimensioni della matrice di output dipendono ancora una volta dalle dimensioni del kernel, dello stride e del padding utilizzato.
La dimensione del kernel determina quanta parte dell'immagine originale contribuisce a formare un singolo valore nell'output. Al contrario delle convoluzioni normali, nelle convoluzioni trasposte, in assenza di padding, un kernel di dimensioni maggiori non riduce l'output, ma piuttosto "disperde" ogni numero dell'input su un'area più ampia della matrice in uscita. Questo comporta che un kernel più grande in una convoluzione trasposta non solo aumenti la dimensione dell'output, ma riesca a catturare informazioni globali su una porzione più ampia. In assenza di padding la dimensione dell'output per lato nella convoluzione trasposta è determinata dalla seguente formula:

$$\text{dim output} = (n-1) \times s + k.$$

Questo significa che il solo kernel espande la dimensione dell'input di $k-1$ pixel per lato.
Nelle convoluzioni trasposte, il parametro stride indica la velocità con cui il kernel si muove sul livello di uscita: più grandi sono i passi, maggiore è la matrice di output, in assenza di padding. L'uso del padding, come nel

caso del same padding, permette di controllare la dimensione dell'output, assicurando che questa sia coerente con le dimensioni desiderate, soprattutto in relazione allo stride: la forma di output per lato è costretta a diventare la forma di input moltiplicata per il passo:

$$\text{dim output} = \text{s} \times \text{n}. \tag{3}$$

Ne consegue che quando s è pari a 1, in entrambi i casi di convoluzione vista, l'uscita ha le stesse dimensioni dell'ingresso.
Per comprendere meglio il funzionamento di questo tipo di layer è necessario far nuovamente riferimento a un esempio operativo: l'immagine 4 mostra come un filtro venga utilizzato per espandere l'input, producendo un output di dimensioni maggiori.

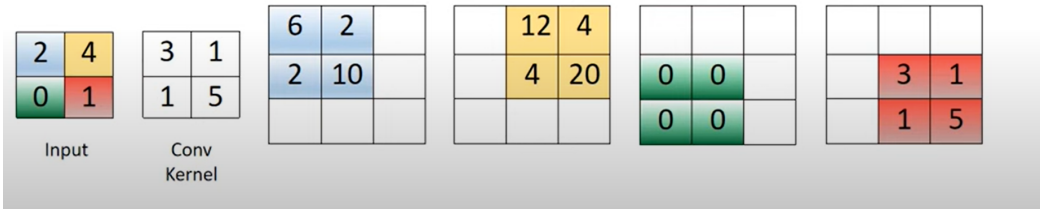


Figura 4: Transposed Convolution. Immagine tratta da [27].

$$\text{Matrice di Input} = \begin{bmatrix} 2 & 4 \\ 0 & 1 \end{bmatrix}, \quad \text{Filtro} = \begin{bmatrix} 3 & 1 \\ 1 & 5 \end{bmatrix}.$$

Non viene applicato alcun padding, quindi l'output è direttamente influenzato dalla dimensione del filtro e dall'input, assumendo dimensioni pari a:

$$\text{dim output} = (2-1) \times 1 + 2 = 3, \text{quindi è una matrice } 3 \times 3.$$

Si inizia considerando il primo valore della matrice di input moltiplicato per il kernel:

$$2 \times \begin{bmatrix} 3 & 1 \\ 1 & 5 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ 2 & 10 \end{bmatrix}.$$

Questo risultato viene posizionato nell'angolo in alto a sinistra della matrice di output con il primo pixel della matrice ottenuta in posizione corrispondente al pixel processato in questo primo passo.
Si prende, successivamente, il secondo valore dell'input e lo si moltiplica per il kernel:

$$4 \times \begin{bmatrix} 3 & 1 \\ 1 & 5 \end{bmatrix} = \begin{bmatrix} 12 & 4 \\ 4 & 20 \end{bmatrix}.$$

La matrice in uscita viene posizionata nell'output spostata di una colonna a destra rispetto alla posizione precedente, poiché il pixel da cui viene calcolato si trova nella seconda colonna dell'input.
Il procedimento viene iterato per gli altri pixel dell'input. Le matrici ottenute vengono poi combinate insieme nella matrice di output, cosicché ogni valore di quest'ultima rappresenti una combinazione ponderata delle caratteristiche dell'input, con contributi dai pixel circostanti tramite la moltiplicazione con il kernel.
Le *Transposed Convolutional Neural Networks* (TCNNs), reti che presentano transposed convolutional layers, sono architetture neurali che svolgono un ruolo complementare rispetto alle CNNs. Mentre queste ultime sono progettate per ridurre la dimensione spaziale dell'input, esse sono utilizzate per eseguire il processo inverso: ricostruire una versione spazialmente più grande dell'input a partire dalle rappresentazioni astratte generate dalla rete.

3.3.2 Backpropagation

Dopo aver calcolato l'output, si procede alla valutazione dell'errore (*loss*) tra il risultato ottenuto e quello reale. L'errore commesso è calcolato tramite la loss function, che assume forme diverse a seconda del task per cui è progettata la rete e che sarà oggetto di minimizzazione:

$$\min_{\mathbf{w} \in R^B} L(\mathbf{w}). \tag{4}$$

Questo calcolo è fondamentale per il processo di apprendimento ed è utilizzato per aggiornare i pesi e i bias della rete attraverso la **backpropagation**.
È doveroso ricordare che si sta parlando in termini di errore medio su tutti i campioni a disposizione del training set: $L = \frac{1}{n} \sum_i L_i$, quindi la minimizzazione della loss function implica una migliore performance della rete su tutti i campioni.
Dopo aver calcolato il valore della perdita, inizia la fase di backpropagation, durante la quale i pesi e i bias dei neuroni vengono aggiornati secondo una determinata politica di ottimizzazione. La più comune è la discesa del gradiente (*Gradient Descent*), che aggiorna i pesi seguendo la direzione opposta al gradiente della funzione di perdita rispetto ai pesi di ciascun layer della rete. Questa collezione di derivate, noto come gradiente rispetto ai pesi (∇L), misura quanto la funzione di perdita cambi in risposta a

piccole variazioni dei pesi della rete. Scendendo lungo il gradiente, il processo di ottimizzazione minimizza la funzione di perdita, migliorando così le prestazioni della rete. Questo passaggio si fa avvalendosi della regola della catena, che rende necessaria la differenziabilità della loss function.
Applicare il Gradient Descent (GD) significa partire dall'attuale valore che assume il peso di un layer e determinare in quale direzione muoversi per ridurre la loss. In presenza di un solo peso, si calcola la derivata rispetto al peso stesso che, in un contesto geometrico unidimensionale, si traduce nella pendenza della retta tangente alla curva valutata nel valore assunto. Se questa è positiva , comporta uno spostameno verso sinistra, se è negativa, il contrario. Generalizzando ad un contesto multidimensionale, le strategie di minimizzazione cercano di scendere iterativamente lungo le curve di livello della funzione di perdita (Figura 6). Localmente, la direzione migliore di discesa è quella opposta al gradiente, che rappresenta la direzione di massima crescita. L'aggiornamento del primo hidden layer di pesi $\mathbf{w_1}$ della Figura 2 alla prima iterazione diventa dunque:

$$\mathbf{w_1}^{(1)} = \mathbf{w_1}^{(0)} - \eta \nabla L(\mathbf{w_1}), \tag{5}$$

dove η è denominato *Learning rate*. Questo iperparametro è molto importante perchè determina la dimensione del passo di aggiornamento nella direzione indicata dal gradiente, influenzando in maniera diretta l'efficacia della discesa. Un valore basso rallenta l'addestramento, richiedendo più tempo per raggiungere la convergenza, ma permette aggiornamenti più precisi, riducendo il rischio di overfitting, soprattutto in presenza di pochi dati. Al contrario, un learning rate alto può accelerare la convergenza, ma comporta il rischio di oscillazioni e di una convergenza instabile, potendo superare i minimi della funzione di perdita e compromettere l'ottimizzazione.
In maniera più generale, possiamo estendere questa formula al vettore comprensivo di tutti i parametri il quale all'iterazione t subisce l'aggiornamento dettato dall'espressione:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \nabla L(\mathbf{w}_{t-1}). \tag{6}$$

Dal momento che η è un numero sempre positivo - varia nel range [0,1] - ogni componente del vettore del gradiente negativo fornisce due informazioni fondamentali:

- **Direzione**: il segno di ciascuna componente (positivo o negativo) indica la direzione in cui la corrispondente componente del vettore di input deve essere spostata, cioè se deve essere aumentata o diminuita;

- **Magnitudine relativa**: le magnitudini relative di tutte le componenti ci dicono quali modifiche sono più importanti, cioè quali componenti del vettore di input dovrebbero essere modificate maggiormente per ottenere un miglioramento significativo della funzione obiettivo.

Stochastic Gradient Descent Quando si lavora con dataset di dimensioni molto grandi, calcolare il gradiente su tutti i campioni per ogni aggiornamento del GD può diventare computazionalmente estremamente costoso, poichè ogni iterazione richiede un passaggio completo su tutto il dataset per calcolare la somma dei gradienti, il che può essere impraticabile in termini di tempo e risorse, specialmente con datasets che contengono migliaia o addirittura milioni di campioni (n molto grande).
Questa sfida ha portato allo sviluppo dello Stochastic Gradient Descent (SGD), una variante del GD che mira a rendere l'algoritmo di ottimizzazione più efficiente e scalabile. L'idea chiave alla base dello SGD è di evitare il calcolo del gradiente su tutto il dataset in ogni iterazione: si seleziona, invece, un piccolo sottoinsieme casuale di campioni (detto "mini-batch") per approssimare il gradiente della funzione di perdita. Il processo di aggiornamento dei pesi viene quindi operato dopo aver visionato ogni mini-batch e la regola (5) diventa:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \cdot \frac{1}{M} \sum_{i \in S_M} \nabla L_i(\mathbf{w}_{t-1}) \quad \text{dove} \quad |S_M| = M, \quad \text{con} \quad M \ll n. \tag{7}$$

Questa metodologia introduce alcuni importanti benefici [19], come:

- Velocizzazione del processo di convergenza: poiché ogni iterazione aggiorna i pesi dopo aver calcolato il gradiente su un sottoinsieme molto più piccolo di dati, l'algoritmo può eseguire molte più iterazioni nello stesso tempo rispetto al GD tradizionale. Questo permette allo SGD di esplorare più rapidamente la superficie della funzione di perdita, avvicinandosi al minimo con maggiore velocità;

- Riduzione del costo computazionale: l'aggiornamento dei pesi basato su un mini-batch o su un singolo campione riduce significativamente il costo computazionale per ogni iterazione. Ciò è particolarmente importante nei casi in cui le risorse hardware siano limitate o quando il dataset sia così grande da non poter essere completamente caricato in memoria;

- Adattabilità: lo SGD è più adattabile a scenari in cui i dati vengono aggiornati o aggiunti continuamente, poichè consente di allenare il modello sui nuovi senza dover rielaborare l'intero dataset.

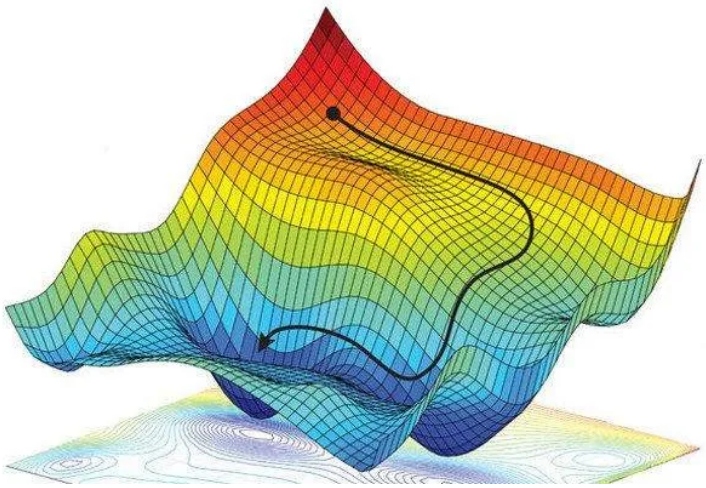


Figura 5: Stochastic Gradient Descent. Immagine tratta da [19].

In sintesi il processo di **training** delle rete si riassume nei seguenti punti:

1. **Divisione del dataset**: l'intero dataset di addestramento viene diviso in più mini-batch;

2. **Aggiornamento del modello**: per ogni mini-batch, viene calcolata la loss confrontando le predizioni con i valori reali, successivamente si valuta il suo gradiente rispetto ai pesi della rete tramite backpropagation e infine si prosegue con l'aggiornamento dei pesi della rete utilizzando un algoritmo di ottimizzazione, come lo SGD o suoi derivati basato sui gradienti calcolati;

3. **Ripetizione**: questo processo viene ripetuto per tutti i mini-batch (un'epoca).

L'intero procedimento descritto viene iterato per un numero scelto di epoche, fino a quando la funzione di perdita non diminuisce più in modo significativo.

Algorithm 2 Stochastic Gradient Descent (SGD)

Require: α: Stepsize (learning rate)
Require: $f(\mathbf{w})$: Stochastic objective function with parameters $\mathbf{w}$
Require: $\mathbf{w}_0$: Initial parameter vector
1: $t \leftarrow 0$
2: **while** $\mathbf{w_t}$ not converged **do**
3: $\quad t \leftarrow t + 1$
4: $\quad \nabla L_t \leftarrow \nabla_{\mathbf{w}} f_t(\mathbf{w}_{t-1})$
5: $\quad \mathbf{w_t} \leftarrow \mathbf{w}_{t-1} - \alpha \cdot \nabla L_t$
6: **return** $\mathbf{w}_t$

Adaptive Moment Estimation L'algoritmo di ottimizzazione specifica la metodologia utilizzata per aggiornare i pesi della rete durante l'addestramento. Un ottimizzatore che prende le mosse dall'algoritmo SGD e trova largo impiego nell'ambito delle reti profonde utilizzate nella Computer Vision è l'*Adaptive Moment Estimation* (Adam).
Adam si distingue per la sua capacità di adattare dinamicamente il learning rate per ciascun parametro della rete neurale, il che lo rende di più alto livello rispetto allo SGD.
Durante l'addestramento, per ogni parametro w_t della rete dell'iterazione t-esima, Adam memorizza la media esponenziale dei gradienti m_t (stima del gradiente medio o momento primo) e la media esponenziale dei gradienti quadrati v_t (stima del gradiente quadrato medio, detta anche momento secondo).
L'algoritmo aggiorna quindi i pesi della rete in modo iterativo e in ogni iterazione (ovvero dopo ogni batch di dati), m_t e v_t vengono ricalcolati secondo le formule [14]:

$$m_t = \beta_1^t \cdot m_{t-1} + (1 - \beta_1^t) \cdot \nabla L_t,$$
$$v_t = \beta_2^t \cdot v_{t-1} + (1 - \beta_2^t) \cdot \nabla L_t^2,$$

dove:

- m_{t-1} è la stima del gradiente medio calcolata nell'iterazione precedente;
- ∇L_t è il gradiente della funzione di costo rispetto ai pesi, calcolato nell'iterazione corrente;
- β_1 e β_2 sono iperparametri che controllano il tasso di decadimento esponenziale per i momenti (tipicamente impostati a 0.9 e 0.999, rispettivamente).

Poiché i momenti m_t e v_t sono inizialmente messi a zero, nelle prime iterazioni dell'allenamento essi tendono ad essere sottostimati.
Adam applica quindi una **correzione del bia**s:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t},$$

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

Infine, i parametri $\mathbf{w}_t$ vengono aggiornati utilizzando le stime corrette del momento e del gradiente quadrato, secondo la formula:

$$\mathbf{w}_{t+1} = \mathbf{w_t} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_\mathbf{t}} + \epsilon} \cdot \hat{\mathbf{m}}_\mathbf{t}.$$

Qui, η rappresenta il *global learning rate* e ϵ è un piccolo valore per evitare la divisione per zero (tipicamente 10^{-8}).
In sintesi, l'algoritmo in cui consiste Adam si compone dei seguenti passi:

1. **Inizializzazione:** all'inizio dell'addestramento si impone un valore a m_0, che tipicamente è inizializzato a zero. Essendo la prima iterazione, infatti, non c'è ancora alcuna informazione sui gradienti, quindi si tratta di una scelta naturale.

2. **Aggiornamento iterativo:** ad ogni iterazione:

   - si calcola il gradiente ∇L_t della funzione di perdita rispetto ai pesi;
   - si aggiorna il momento primo m_t e il momento secondo v_t utilizzando le formule sopracitate;
   - si aggiornano i pesi.

3. **Ripetizione:** questo processo viene ripetuto ad ogni iterazione fino al termine dell'addestramento.

Algorithm 3 Adam: A Method for Stochastic Optimization

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\mathbf{w})$: Stochastic objective function with parameters $\mathbf{w}$
Require: $\mathbf{w}$: Initial parameter vector

$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
$t \leftarrow 0$
while $\mathbf{w}_t$ not converged **do**
    $t \leftarrow t + 1$
    $\nabla L_t \leftarrow \nabla_{\mathbf{w}} f_t(\mathbf{w}_{t-1})$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla L_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \nabla L_t^2$
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
    $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
return $\mathbf{w}_t$

La regolazione automatica del passo di aggiornamento di ciascun parametro è una tecnica estremamente efficace in quanto alcuni potrebbero necessitare di aggiornamenti più grandi per convergere rapidamente, altri richiedere cambiamenti più piccoli per evitare di oscillare troppo. E' ciò che succede nella fasi inziali dell'addestramento, quando i gradienti possono essere grandi e un learning rate adattativo aiuta a prevenire oscillazioni nei pesi. Nelle fasi avanzate, invece, l'adattamento del learning rate assicura che la rete continui a fare progressi verso la convergenza, anche se più lentamente.

Confronto tra backpropagation nei layer densi e convoluzionali Nei *layer densi* (fully connected) in cui ogni neurone è connesso a tutti i neuroni del layer precedente, la formula per il calcolo dell'output di ciascun neurone i nel layer l è data da:

$$z_i^{(l)} = \sum_j w_{ij}^{(l)} h_j^{(l-1)},$$

dove $h_j^{(l-1)}$ è l'output del neurone j nel layer $l-1$. L'attivazione $h_i^{(l)}$ del neurone i nel layer l è ottenuta applicando una funzione di attivazione f:

$$h_i^{(l)} = f(z_i^{(l)}).$$

La derivata della funzione di perdita L rispetto al peso $w_{ij}^{(l)}$ è data da:

$$\frac{\partial L}{\partial w_{ij}^{(l)}} = \frac{\partial L}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} \cdot h_j^{(l-1)}, \tag{8}$$

avendo indicato con $\delta_i^{(l)} = \frac{\partial L}{\partial z_i^{(l)}}$ il gradiente dell'errore rispetto all'input netto del neurone i nel layer l. In questo caso, il peso $w_{ij}^{(l)}$ influenza direttamente un singolo neurone i nel layer l, cioè un singolo output $z_i^{(l)}$.
Differentemente, nei layers convoluzionali, i pesi che compongono i filtri sono condivisi tra diverse regioni dell'input.
Consideriamo un filtro di dimensioni $M \times N$ come una piccola matrice di pesi $w_{mn}^{(l)}$, dove m e n indicano la posizione all'interno del filtro. L'output $z_{ij}^{(l)}$ del pixel (i, j) nella feature map del layer l è calcolato come una convoluzione del filtro sulla regione corrispondente dell'input $h^{(l-1)}$, ottenuto dal layer precedente:

$$z_{ij}^{(l)} = \sum_{m=1}^{M} \sum_{n=1}^{N} w_{mn}^{(l)} \cdot h_{(i+m-1)(j+n-1)}^{(l-1)}.$$

La derivata della funzione di perdita L rispetto a un peso $w_{mn}^{(l)}$ all'interno del filtro è data da:

$$\frac{\partial L}{\partial w_{mn}^{(l)}} = \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\partial L}{\partial z_{ij}^{(l)}} \cdot h_{(i+m-1)(j+n-1)}^{(l-1)},$$

dove:

- H e W sono le dimensioni della feature map risultante,
- $\frac{\partial L}{\partial z_{ij}^{(l)}}$ è il gradiente dell'errore rispetto all'output $z_{ij}^{(l)}$ del pixel (i, j).

In questo caso, il peso $w_{mn}^{(l)}$ viene applicato ripetutamente su diverse regioni dell'input, influenzando più pixel $z_{ij}^{(l)}$ della mappa delle feature risultante. Di conseguenza, la derivata della perdita rispetto a $w_{mn}^{(l)}$ è la somma dei contributi di tutti i pixel $z_{ij}^{(l)}$ a cui $w_{mn}^{(l)}$ ha contribuito.
La convoluzione trasposta segue lo stesso ragionamento della convoluzione standard, ma al contrario: invece di ridurre le dimensioni dell'input, espande l'output. La formula per la derivata della perdita rispetto ai pesi in un layer convoluzionale trasposto è analoga a quella della convoluzione standard, ma applicata nel senso inverso.

Questa differenza nella condivisione e applicazione dei pesi nei layer convoluzionali e convoluzionali trasposti rispetto alla connessione diretta nei layer densi è ciò che rende i layer convoluzionali particolarmente efficienti in termini di parametri e adatti per operazioni di riconoscimento oggetti o generazione di immagini.

3.3.3 Vanishing and Exploding Gradient

Il concetto di **vanishing gradient** si riferisce alla riduzione progressiva delle derivate della funzione di loss rispetto ai pesi mentre vengono propagate all'indietro attraverso una rete neurale durante la backpropagation, provocando aggiornamenti dei pesi insignificanti. Al contrario, se le derivate sono molto grandi, si può incorrere nel problema opposto, noto come **exploding gradient**, in cui gli aggiornamenti dei pesi diventano eccessivi. Questi tipi di problemi sono particolarmente manifesti nei dense layers, dove ogni neurone è completamente connesso con i neuroni del layer successivo. A causa di queste connessioni complete, piccoli gradienti si moltiplicano attraverso i layers, riducendosi ulteriormente e causando una difficoltà nell'aggiornamento efficace dei pesi. Considerando l'esempio della rete neurale con tre strati della Figura 2, il gradiente della funzione di perdita L rispetto al peso della prima connessione del primo layer diventa:

$$\frac{\partial L}{\partial w_{11}^{(1)}} = x_1 \cdot \sum_k \left(\delta_k^{(3)} \cdot w_{k1}^{(3)} \cdot f'(z_2) \cdot w_{1k}^{(2)} \cdot f'(z_1) \right).$$

Se la funzione di attivazione $f(z)$ è una funzione che satura, come la sigmoide o la tangente iperbolica (mostrate in Figura 6), le derivate $f'(z)$ tendono ad essere molto piccole quando z è in un intervallo lontano da zero (molto positivo o molto negativo).

Sigmoid:	Tanh:	ReLU:
$\sigma(x) = \frac{1}{1+e^{-x}}$	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\text{ReLU}(x) = \max(0, x)$
$\sigma : R \to (0, 1)$	$\tanh : R \to (-1, 1)$	$\text{ReLU} : R \to R^+$

Quando gli input cadono in regioni sature, i gradienti si avvicinano allo zero, con conseguente scarso aggiornamento dei pesi dello strato precedente. Il gradiente calcolato in ogni strato è il prodotto di molteplici derivate, che includono quelle delle funzioni di attivazione e i pesi delle matrici. Se anche uno di questi termini è molto piccolo, il gradiente totale può diminuire drasticamente. Quando questo fenomeno si ripete strato dopo

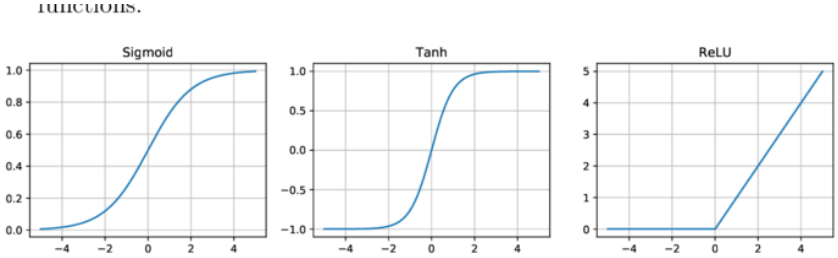


Figura 6: Sigmoid, Tanh and RELU Activation Function. Immagine tratta da [16].

strato i gradienti che arrivano ai layer più vicini all'input (come $\mathbf{w}_1$) diventano estremamente piccoli.
Nelle reti semplici ciò non rappresenta un grosso problema, ma man mano che vengono aggiunti più layer, questi piccoli gradienti, che si moltiplicano tra gli strati, decadono in modo significativo, comportando un mancato aggiornamento dei primi strati della rete che hanno un ruolo fondamentale nella cattura delle correlazioni del modello. Un indicatore chiave di questo fenomeno è la stagnazione nel miglioramento dei parametri prestazionali del modello nel corso dei periodi di addestramento [4].
L'effetto complessivo è che, durante l'addestramento, gli strati più profondi potrebbero apprendere correttamente, mentre gli strati più vicini all'input rimarrebbero praticamente invariati. Questo porta a un modello poco efficiente, poiché l'intera rete non è in grado di ottimizzarsi completamente.
I **convolutional layers**, invece, hanno una struttura che limita il numero di connessioni tra i neuroni, poiché i filtri convoluzionali operano su piccole regioni locali dell'input. Questo approccio preserva meglio l'informazione e riduce la probabilità di amplificare gradienti piccoli, mitigando in parte il fenomeno. Nel contesto di reti profonde le funzioni di attivazione come *ReLU (Rectified Linear Unit)* e *Leaky ReLU* offrono significativi vantaggi poichè sono esenti dal problema del vanishing gradient.
La ReLU restituisce direttamente l'input quando questo è positivo, mantenendo una derivata costante pari a 1. Questo significa che, per input positivi, il gradiente non si riduce mai a zero durante la propagazione all'indietro. Queste proprietà consentono a ReLU di mantenere gradienti più consistenti e di accelerare la convergenza di SGD, come dimostrato in [33].
Il problema dell'exploding gradient, al contrario, si verifica quando, durante la propagazione all'indietro, le derivate degli strati della rete neurale diventano progressivamente più grandi man mano che ci si sposta verso gli strati iniziali. La causa principale di questo problema risiede nei pesi della rete: valori elevati portano a derivate corrispondentemente grandi, causando

variazioni significative nei nuovi valori dei pesi rispetto a quelli precedenti. Di conseguenza, il gradiente non riesce a convergere e può far oscillare il modello attorno ai minimi locali, rendendo difficile il raggiungimento del minimo globale. Durante il processo di addestramento, si possono incontrare valori "NaN" nella loss function o in altri calcoli intermedi [4]. La funzione di perdita, in tal caso, mostra un comportamento irregolare, oscillando invece di diminuire costantemente, suggerendo che i pesi della rete vengono aggiornati eccessivamente da grandi gradienti, impedendo una convergenza uniforme.

3.3.4 Overfitting

Le reti neurali profonde contengono più livelli nascosti non lineari, il che le rende in grado di apprendere relazioni molto complesse tra i loro input e output. Tuttavia, con dati di addestramento limitati come il caso trattato in questa tesi, il modello non solo memorizza i pattern o le relazioni utili esistenti ma anche i dettagli irrilevanti o casuali che non sono rappresentativi delle caratteristiche generali del problema, come outliers o altre forme di rumore presenti. Questo fenomeno, noto come *overfitting*, si manifesta quando il modello si adatta così strettamente ai dati di addestramento da perdere la capacità di generalizzare quando deve predire il comportamento di scenari nuovi. Per fronteggiare questa tendenza sono state implementate tecniche di regolarizzazione tra le quali spicca il *dropout* [24].
Durante l'addestramento di una rete neurale il suddetto processo è responsabile dello spegnimento casuale di un certo numero di neuroni in ogni iterazione, che viene conseguentemente escluso dal calcolo dell'output e dall'aggiornamento dei pesi. Questo significa che, per ogni passo di aggiornamento del training, solo una parte dei neuroni è attiva e contribuisce alla formazione del modello. Tuttavia, grazie alla natura casuale del dropout, nel corso di molte iterazioni tutti i neuroni verranno attivati abbastanza frequentemente da avere i loro pesi aggiornati in modo equilibrato.
La probabilità che un neurone rimanga attivo durante un dato passaggio è determinata da un parametro chiamato p. Ad esempio, se p è uguale a 0.5, c'è una probabilità del 50% che un neurone sia attivo in un'iterazione specifica. Questa tecnica introduce una sorta di ensemble learning all'interno della stessa rete neurale: durante l'addestramento, diverse versioni diradate della rete (vedi Figura 8) vengono addestrate, poiché ogni iterazione lavora su un sottoinsieme diverso di neuroni.

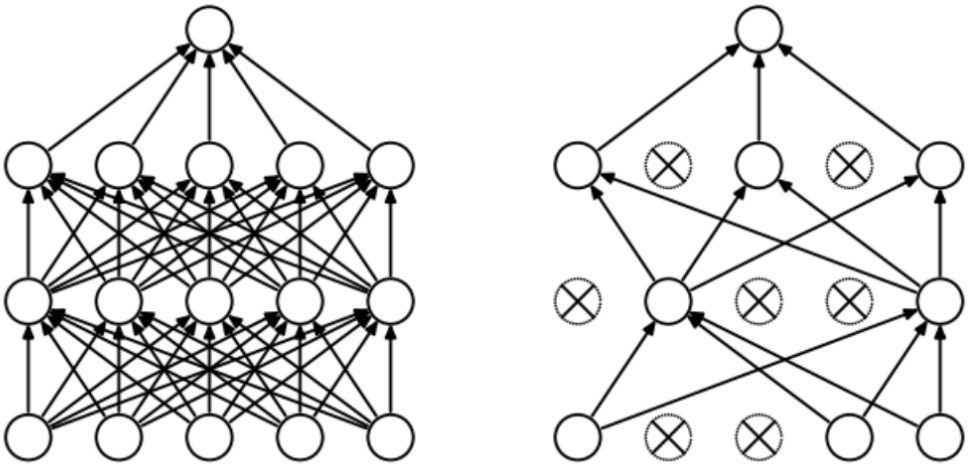
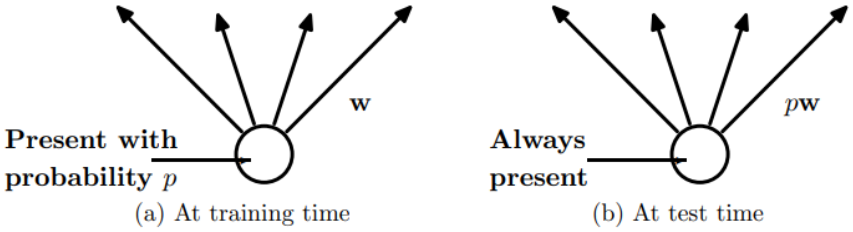


Figura 7: Dropout. Immagine tratta da [24].

Non potendo fare affidamento su una particolare combinazione di neuroni per fare predizioni, la possibilità che il modello si adatti eccessivamente ai dati di addestramento si riduce, contribuendo in tal modo alla sua robustezza e capacità di generalizzazione.
Durante la fase di test, il dropout non viene applicato e tutti i neuroni della rete sono attivi. Tuttavia, per garantire coerenza con il processo di addestramento, i pesi dei neuroni vengono scalati moltiplicandoli per la probabilità p utilizzata durante l'addestramento. Questo accorgimento serve a compensare l'attivazione simultanea di tutti i neuroni, assicurando che le prestazioni della rete in fase di predizione riflettano fedelmente quanto appreso durante l'addestramento con dropout. Senza questa correzione, infatti, la rete potrebbe generare output incoerenti, compromettendo l'affidabilità delle previsioni.



(a) At training time (b) At test time

Figura 8: A sinistra: il neurone durante l'addestramento è presente con probabilità p ed è collegato alle unità nel livello successivo con pesi w.
A destra: durante il test, l'unità è sempre presente e i pesi sono moltiplicati per p. Immagine tratta da [24].

3.4 Funzioni di loss

Nella Computer Vision, le funzioni di loss L svolgono un ruolo cruciale nel processo di addestramento dei modelli, poiché determinano il modo in cui viene misurata la discrepanza tra le previsioni del modello e i risultati attesi. Tra le funzioni di loss più comunemente utilizzate in questo campo spiccano il Mean Squared Error (MSE) e lo Structural Similarity Index (SSIM), due metriche che offrono approcci distinti per valutare la qualità delle immagini generate o modificate da un modello.

3.4.1 Mean Squared Error

Il **MSE** è una metrica utilizzata nell'elaborazione delle immagini per quantificare l'accuratezza di un modello predittivo, misurando direttamente l'entità dell'errore tra un'immagine originale e la sua versione distorta. Essa viene calcolata prendendo la media dei residui al quadrato, dove il residuo è la differenza tra il valore previsto e il valore effettivo per ciascun pixel:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{9}$$

n è il numero di pixel nell'immagine, y_i e $\hat{y}_i$ sono rispettivamente il valore reale e previsto.
Ogni differenza al quadrato $(y_i - \hat{y}_i)^2$ rappresenta una misura della deviazione di una previsione rispetto al valore reale. Prendendo la somma di queste differenze quadratiche e poi calcolandone la media, il MSE fornisce una misura aggregata di quanto le previsioni si discostino in media dai valori reali. Per tale motivo può essere interpretata come una misura di distanza, sebbene

non sia da intendersi nel senso geometrico del termine poichè, nonostante soddisfi le proprietà di non negatività, identità e simmetria richieste, viola la disuguaglianza triangolare in senso stretto, perché la somma dei quadrati delle differenze può crescere non linearmente quando si confrontano diverse coppie di immagini.

$$\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{10}$$

A differenza della distanza euclidea di uno spazio n-dimensionale (10), infatti, non è una misura diretta della distanza tra due punti, ovvero una norma L^2 e per tale motivo può intendersi come una metrica derivata da essa.
Questa interpretazione del MSE comporta la comprensione della magnitudine dell'errore e delle sue implicazioni sulle prestazioni del modello. Un MSE più basso indica che le previsioni del modello sono più vicine ai valori effettivi (l'immagine distorta è molto simile all'originale), segnalando una migliore accuratezza. Al contrario, un MSE più alto suggerisce che le previsioni si discostano maggiormente dai valori reali, indicando prestazioni inferiori.
Spesso questa funzione viene utilizzata come loss nel processo di apprendimento, in quanto convessa, simmetrica e differenziabile. Queste proprietà facilitano l'uso di algoritmi di ottimizzazione come lo SGD per l'addestramento del modello consentendo una retropropagazione stabile ed efficiente con convergenza della rete verso una soluzione ottimale facilmente computabile.
Tra gli altri vantaggi che apporta troviamo la capacità di fornire una misura completa dell'accuratezza del modello, che gli ha permesso di affermarsi come standard convenzionale per valutare e confrontare le prestazioni degli algoritmi di elaborazione delle immagini; la semplicità e intuitività della formulazione matematica che lo definisce: per ogni pixel dell'immagine, è necessario eseguire una sottrazione, elevare al quadrato il risultato, sommare e poi dividere per il numero totale di pixel; la capacità di valutare l'errore su ogni campione in modo indipendente, senza necessità di considerare i precedenti; la sensibilità agli errori sia piccoli che grandi. Tuttavia, proprio quest'ultimo punto nasconde un'importante limitazione associata all'MSE: esso penalizza gli errori grandi in modo sproporzionato rispetto a quelli piccoli, poiché le differenze vengono elevate al quadrato comportando una potenziale distorsione della valutazione complessiva delle prestazioni del modello. Tra le altre limitazioni associate ai derivati della norma L^2 c'è il fatto che quando si tratta di valutare la qualità delle immagini, non si correla bene con la qualità percepita da un osservatore umano. Questo è dovuto a diverse assunzioni implicite nel suo utilizzo, come esposto da Zhou Wang and Alan C. Bovik in [25]:

1. **Assunzione 1**: l'MSE è indipendente dalle relazioni spaziali tra i pixel dell'immagine originale. Ignorando completamente la struttura spaziale delle immagini, se il segnale originale e quello distorto vengono riordinati casualmente nello stesso modo, l'MSE tra i due rimarrà invariato. Tuttavia, cambiando l'ordine spaziale dei pixel, la struttura visiva dell'oggetto rappresentato nell'immagine può essere alterata significativamente. In un'immagine di un incendio, i pixel non sono indipendenti: la loro disposizione spaziale è cruciale per rappresentare correttamente le fiamme, il fumo e le aree circostanti. L'Assunzione 1 implica che, se l'immagine viene riordinata casualmente, l'MSE non cambierà, anche se il risultato potrebbe non rappresentare più un incendio in modo realistico. Questo evidenzia una grave limitazione dell'MSE nel catturare la struttura spaziale necessaria per un'accurata rappresentazione visiva degli incendi.

2. **Assunzione 2**: l'MSE è indipendente da qualsiasi relazione tra il segnale originale e il segnale di errore. Esso misura, infatti, solo l'entità dell'errore, senza considerare come questo si distribuisca rispetto al contenuto dell'immagine originale. In altre parole, se due immagini hanno lo stesso segnale di errore aggiunto, l'MSE sarà il medesimo per entrambe, anche se l'errore è più visivamente percepibile in una rispetto all'altra. Nel contesto delle simulazioni di incendi, l'immagine prevista è spesso un'approssimazione dell'immagine reale e le differenze tra le due (ovvero il rumore) possono essere distribuite in modo non uniforme. L'MSE non distingue tra errori che avvengono in aree critiche, come i bordi del fuoco, e quelli che si verificano in zone meno rilevanti. Di conseguenza potrebbe non riflettere accuratamente la qualità dell'immagine ricostruita se l'errore è concentrato in aree visivamente importanti, sottovalutando così l'impatto delle distorsioni.

3. **Assunzione 3**: la fedeltà del segnale è indipendente dai segni dei campioni del segnale di errore. Se in un'area dell'immagine originale il valore del pixel viene aumentato (errore positivo) o diminuito (errore negativo) rispetto al valore corretto, l'MSE tratterà entrambi gli errori allo stesso modo. Nel contesto delle immagini, questo significa che l'MSE non fa differenza tra un errore positivo (quando un pixel nell'immagine distorta è più luminoso rispetto all'originale) e un errore negativo (quando un pixel nell'immagine distorta è più scuro rispetto all'originale).

3.4.2 Structural Similarity Index

Lo **SSIM** è una metrica sviluppata per valutare la somiglianza tra due immagini, basata sul presupposto che il sistema visivo umano (HVS) sia altamente adatto ad estrarre informazioni strutturali dal campo visivo. Lo SSIM viene calcolato localmente all'interno di una finestra mobile, generalmente rappresentata da un filtro gaussiano G_σ che si sposta pixel per pixel attraverso l'immagine, risultando in una mappa. Questo filtro assegna pesi maggiori ai pixel vicini al centro della finestra e pesi minori ai pixel più lontani, riflettendo l'idea che quelli centrali siano più rilevanti per la valutazione della qualità strutturale. È importante notare che, affinché l'intensità complessiva dell'immagine rimanga invariata, i pesi del filtro gaussiano sono normalizzati in modo tale che la loro somma sia pari a 1.
Lo SSIM è progettato per catturare la somiglianza tra l'immagine originale x e quella distorta y attraverso tre componenti principali: luminanza $l(x, y)$, contrasto $c(x, y)$ e struttura $s(x, y)$. Queste componenti sono espresse utilizzando statistiche locali facilmente calcolabili e vengono combinate per formare lo SSIM locale.

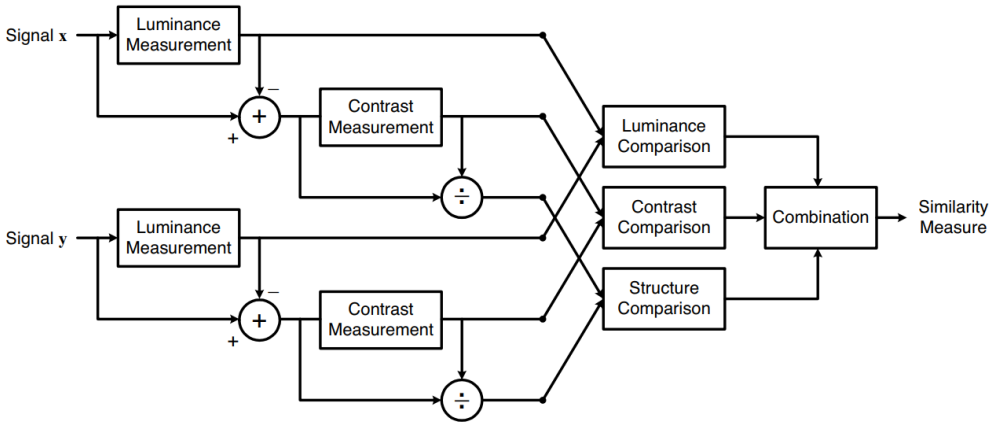


Figura 9: Structural Similary Index. Immagine tratta da [26].

Il diagramma 9 illustra questo processo di confronto che si compone dei seguenti step [26]:

1. **Misurazione della Luminanza (Luminance Measurement):** entrambi i segnali, x e y, passano attraverso un blocco di *Luminance Measurement* che calcola la luminanza media per ciascuna immagine. Le medie μ_x e μ_y della luminosità delle due immagini sono una media pesata dei valori dei pixel all'interno della finestra locale centrata su

un pixel, con pesi determinati dal filtro gaussiano G_σ [26]:

$$\mu_x = \sum_{i=1}^{N} w_i x_i, \tag{11}$$

dove w_i rappresenta il peso determinato dal filtro e x_i i valori dei pixel nella finestra locale.

2. **Misurazione del Contrasto (Contrast Measurement):** dopo aver ottenuto la luminanza, ciascun segnale passa attraverso un blocco di *Contrast Measurement*, dove viene calcolata la varianza locale. Le varianze σ_x^2 e σ_y^2 sono una media pesata delle differenze al quadrato tra i valori dei pixel nella finestra locale e la media:

$$\sigma_x^2 = \sum_{i=1}^{N} w_i (x_i - \mu_x)^2. \tag{12}$$

La normalizzazione del contrasto viene eseguita dividendo la varianza calcolata per la luminanza media ottenuta nel passaggio precedente.

3. **Confronto di Luminanza, Contrasto e Struttura:** i risultati della misurazione della luminanza e del contrasto vengono quindi confrontati tra i due segnali x e y rispettivamente nei blocchi *Luminance Comparison*

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{13}$$

e *Contrast Comparison*:

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}. \tag{14}$$

Parallelamente, la *Structure Comparison* confronta la correlazione strutturale tra le due immagini x e y attraverso la covarianza σ_{xy}. Quest'ultima è calcolata come una media pesata dei prodotti delle differenze tra i valori dei pixel nelle due immagini e le rispettive medie locali [26]:

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y), \tag{15}$$

e confluisce nella formula di confronto come:

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \tag{16}$$

È importante notare che le strutture degli oggetti nella scena sono indipendenti dall'illuminazione. Di conseguenza, per esplorare l'informazione strutturale in un'immagine, si è reso necessario separare l'influenza dell'illuminazione.

4. **Combinazione e Misura di Similarità (Combination and Similarity Measure):** i risultati di questi tre confronti (luminanza, contrasto e struttura) vengono poi combinati nel blocco *Combination* per calcolare l'indice di somiglianza finale:

$$\text{SSIM(p)} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} = lc(p) \cdot s(p). \qquad (4)$$

Gli elementi C_1, C_2 e C_3 sono piccole costanti positive che stabilizzano ciascun termine, in modo che statistiche prossime allo zero non causino instabilità numerica. In realtà, anche se $C_1 = C_2 = C_3 = 0$, lo SSIM funziona generalmente molto bene [10].

Per ottenere una stima globale della somiglianza strutturale tra le due immagini, i valori di SSIM per tutte le regioni dell'immagine vengono mediati per ricavare un singolo punteggio. Ciò rende questa metrica particolarmente utile in applicazioni in cui la qualità percepita è fondamentale, come la compressione delle immagini, la super risoluzione e la riduzione del rumore. Lavorando su finestre di pixel è infatti in grado di catturare le strutture locali dell'immagine, un aspetto critico per mantenere dettagli visivi importanti che altre metriche, come il MSE, potrebbero non considerare adeguatamente.
Tuttavia, non manca di alcune limitazioni [28]. Una delle principali è la sua minore sensibilità ai piccoli cambiamenti uniformi nelle regioni piatte dell'immagine, dove la struttura è meno evidente. Inoltre, l'efficacia dello SSIM può dipendere fortemente dalla dimensione del filtro utilizzato: un filtro troppo grande favorisce la somiglianza su larga scala rischiando di non catturare bene i dettagli, mentre uno troppo piccolo rende lo SSIM più sensibile alle variazioni locali, ma potrebbe non accorgersi delle macro somiglianze.
Studi recenti, come [25] [28], hanno mostrato come la funzione di perdita per SSIM per una porzione di immagine P individuata da una finestra centrata sul pixel p-esimo possa essere espressa impostando:

$$L_{SSIM}(p) = 1 - SSIM(p),$$

e la loss complessiva L_{SSIM} su tutti i pixel dell'immagine:

$$L_{SSIM} = \frac{1}{N} \sum_{P} \left(1 - SSIM(p)\right),$$

dove N rappresenta il numero totale di patch nell'immagine. L'idea è che minimizzando questa perdita, si massimizzi il valore dello SSIM e quindi si ottenga un'immagine più simile a quella di riferimento. In sintesi, si vuole addestrare la rete neurale affinché produca immagini che, quando valutate con lo SSIM, risultino altamente simili alle immagini di riferimento, il che implica massimizzare la covarianza tra le strutture locali delle due immagini. È importante notare che, come evidenziato nell'equazione (4), il calcolo di $SSIM(p)$ richiede l'analisi di una regione attorno al pixel p di dimensione pari al supporto del filtro G_σ. Di conseguenza, sia $L_{\text{SSIM}}(p)$ che le sue derivate non possono essere calcolate nelle aree di confine di P senza l'applicazione di padding. Questo vincolo non si applica a funzioni di perdita più semplici come ℓ_1 o ℓ_2 e alle loro derivate, che considerano solo il valore della patch di riferimento e della patch elaborata al pixel p. Va inoltre sottolineato che la funzione di perdita basata sullo SSIM è differenziabile rispetto ai pixel dell'immagine, permettendo di calcolare la derivata parziale di $L_{\text{SSIM}}(p)$ rispetto a $x(q)$ come segue:

$$\frac{\partial L_{\text{SSIM}}(p)}{\partial x(q)} = -\frac{\partial}{\partial x(q)} SSIM(p) = -\left(\frac{\partial l(p)}{\partial x(q)} \cdot cs(p) + l(p) \cdot \frac{\partial cs(p)}{\partial x(q)} \right).$$

4 Implementazione del modello

Questa sezione si concentra sull'implementazione di un modello che integri la simulazione fluidodinamica con l'intelligenza artificiale per l'analisi della stratificazione del fumo negli incendi in galleria. La Sezione 3.1 approfondisce la fisica dell'incendio, mentre la Sezione 3.1.1 esamina l'evoluzione della stratificazione dei fumi e il gradiente termico, evidenziandone la reciproca dipendenza. La Sezione 3.2 descrive lo studio della configurazione del modello, analizzando l'influenza dei parametri relativi alla posizione dell'incendio e alla geometria dell'edificio sul comportamento del fumo dopo lo scoppio dell'incendio. La Sezione 3.3 presenta la fase di pre-processing della simulazione fluidodinamica, in cui vengono descritte la definizione del dominio computazionale, la mesh di calcolo e le condizioni al contorno necessarie per settare correttamente il simulatore. La Sezione 3.4 si occupa della formazione del dataset, spiegando come le immagini CFD siano utilizzate per addestrare il modello di intelligenza artificiale. La Sezione 3.5 analizza l'architettura della rete, giustificando la struttura del modello di deep learning utilizzato. Le Sezioni 3.6 e 3.7 si concentrano sulla scelta degli iperparametri della rete e la loro ottimizzazione attraverso l'uso dell'algoritmo Hyperband. Infine, la Sezione 3.8 discute l'utilizzo del MSE e la successiva transizione a una Custom Loss con lo SSIM, per migliorare la qualità strutturale delle previsioni.

4.1 Fisica dell'incendio

La dinamica di un incendio in un ambiente confinato, come una galleria, è fortemente influenzata dai cambiamenti sia di temperatura che di pressione, i quali interagiscono tra loro in modo complesso e interdipendente. Quando un incendio si innesca, il materiale combustibile inizia a bruciare, generando calore e provocando un rapido aumento della temperatura dell'aria circostante, determinando così una diminuzione della densità dell'aria. Concomitantemente, si verifica anche una diminuzione della pressione locale nella zona sovrastante. Quest'area di bassa pressione genera un gradiente rispetto all'ambiente circostante, dove la pressione è più alta.
La differenza creatasi favorisce l'ingresso di aria fresca dai portali o dalle aperture situate nelle parti inferiori della galleria: l'aria fresca, più densa e pesante, viene richiamata verso l'incendio per bilanciare la pressione, alimentando ulteriormente la combustione con un apporto di ossigeno.
Parallelamente, il fumo prodotto, avente temperatura maggiore e quindi densità minore dell'aria circostante, tende a salire verso la parte superiore della galleria, mentre l'aria fredda viene richiamata nella parte inferiore. Le

differenze di pressione e temperatura agiscono quindi in sinergia, generando movimenti convettivi all'interno della galleria. Questo fenomeno porta alla formazione di una stratificazione dell'aria, con uno strato caldo e fumoso che si accumula nella parte superiore e uno strato più freddo e denso nella parte inferiore.

4.1.1 Evoluzione della stratificazione dei fumi e gradiente di temperatura

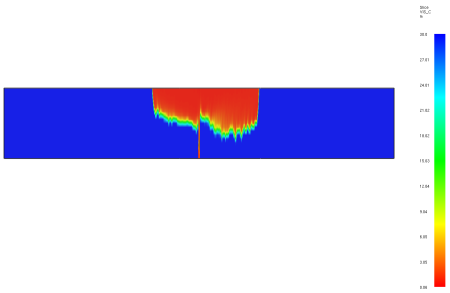
La forza del gradiente termico influisce direttamente sull'intensità delle correnti convettive e sulla stabilità della stratificazione [22]. In assenza di un forte gradiente termico, le correnti convettive saranno più deboli, risultando in una stratificazione meno marcata, cioè in una distribuzione più uniforme della temperatura lungo la galleria.
Quando il gradiente termico assume un profilo importante, ovvero quando c'è una grande differenza di temperatura tra l'aria calda e quella circostante, che si trova a temperatura ambiente, la stratificazione diventa inizialmente molto evidente, con un netto accumulo di aria calda e fumosa nella parte superiore della galleria. Tuttavia, le correnti convettive che si sviluppano in queste condizioni sono più intense, il che può portare a un rapido mescolamento di aria calda e fredda, che tende a rompere la stratificazione, riducendo la durata della distinzione tra gli strati e distribuendo il calore in altre parti della galleria. Questo processo può peggiorare la visibilità, mettendo a rischio gli utenti in fuga e i soccorritori.
L'estremizzazione di questa dinamica si presenta in gallerie in pendenza e prende il nome di *effetto camino*. In questo caso, l'aria calda e i gas di combustione tendono a risalire lungo la pendenza, accelerando il movimento dell'aria verso l'alto e attirando ulteriormente aria fredda dal basso. Questo fenomeno intensifica l'incendio e velocizza la propagazione del fumo e del calore lungo il tunnel, rendendo ancora più complessa la gestione dell'emergenza. In questo caso, l'aria calda e i gas di combustione tendono a risalire lungo la pendenza della galleria, accelerando il movimento dell'aria verso l'alto e attirando ulteriormente aria fredda dal basso. Questo fenomeno intensifica l'incendio e accelera la propagazione del fumo e del calore lungo la galleria, rendendo ancora più complessa la gestione dell'emergenza.
Le immagini in Figura 10 mostrano l'evoluzione del profilo di visibilità all'interno di una galleria con pendenza moderata, illustrando come la visibilità varia su un piano verticale longitudinale posto lungo l'asse. La colorbar indica i colori associati a una visibilità compresa tra 0 e 30 m. La normativa internazionale è concorde nel considerare il valore di visibilità

pari a 10 m come limite di sicurezza per l'evacuazione degli utenti dall'area dell'incendio. In generale, possiamo distinguere tre macro aree:

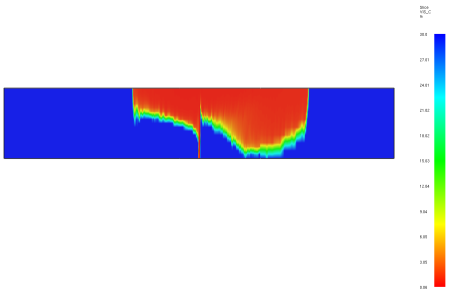
- **Rosso**: indica le aree con visibilità più ridotta, corrispondenti alle regioni dove il fumo è più denso e la temperatura è più elevata;
- **Verde e Giallo**: rappresentano le regioni di transizione, dove la densità del fumo e la temperatura iniziano a diminuire. Queste aree sono tipicamente situate tra la zona a bassa visibilità (rossa) e la zona di visibilità più alta (blu);
- **Blu**: suggerisce aree con visibilità migliore, dove il fumo è meno presente o la temperatura è più bassa, ovvero zone meno influenzate dall'incendio.

La visibilità tende a peggiorare vicino alla fonte dell'incendio, con un progressivo miglioramento man mano che ci si allontana, come è evidente nella colorazione che passa dal rosso (scarsa visibilità) al blu (buona visibilità) nelle immagini. In condizioni di incendio, la stratificazione del fumo e il gradiente di temperatura creano un ambiente dove la visibilità è drasticamente ridotta nelle zone più calde e dense di fumo (rosso), mentre migliora gradualmente man mano che si scende verticalmente o ci si allontana dalla fonte di calore.

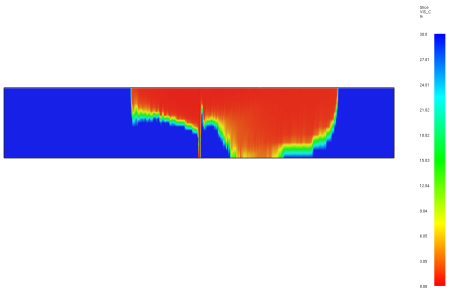
Figura 10: Modello di incendio con HRR=8 MW, pendenza=3%, posizione dell'incendio centrale, sezione della galleria=83 m^2, altezza 7,2 m, larghezza 12,8 m.



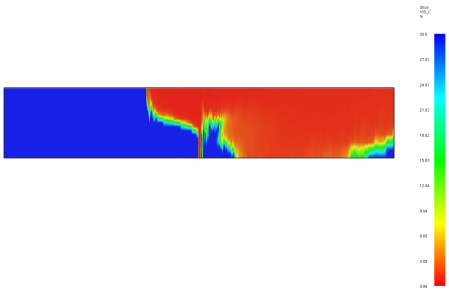
(a) Profilo di visibilità a 80 s.



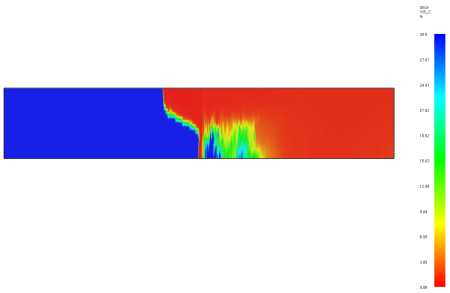
(b) Profilo di visibilità a 160 s.



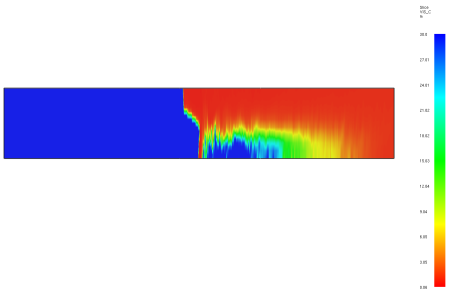
(c) Profilo di visibilità a 210 s.



(d) Profilo di visibilità a 330 s.



(e) Profilo di visibilità a 480 s.



(f) Profilo di visibilità a 860 s.

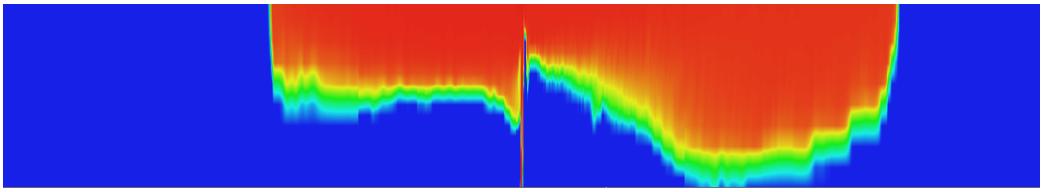
4.2 Studio della configurazione

In ambito ingegneristico, la progettazione di edifici sottoposti a rischi di incendio si basa su compartimenti antincendio, spazi delimitati entro i quali si prevede lo sviluppo e il contenimento del fuoco. La valutazione spaziale dell'ambiente in cui può svilupparsi un incendio è essenziale per comprendere

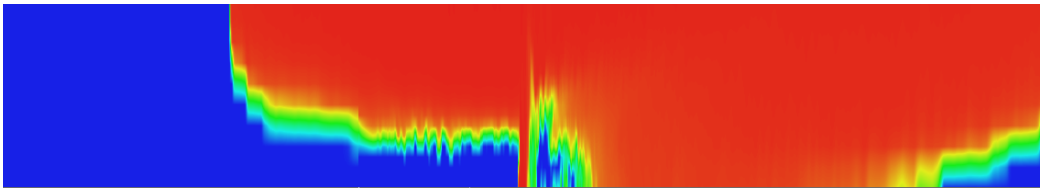
la dinamica del fuoco all'interno di un compartimento specifico. Questo processo viene sintetizzato con la denominazione *studio della configurazione* che, nell'ambito della simulazione dell'evento, diventa quindi il parametro di riferimento principale.
Per formare un database numerico che copra una vasta gamma di scenari di PBD, vengono considerati quattro parametri chiave che influenzano la struttura della galleria: la sua geometria, la sua pendenza, la dimensione dell'incendio e la posizione in cui ha origine.

- **Geometria dell'edificio**: la *NFPA 502, Standard for Road Tunnels, Bridges, and Other Limited Access Highways* [21], stabilisce che l'altezza e la larghezza delle gallerie siano determinate dal progettista, a condizione che permettano il passaggio sicuro dei veicoli, inclusi quelli di emergenza. Il dimensionamento della galleria di interesse è stato definito scegliendo due larghezze diverse dei portali: 12,8 m e 16 m, simulando un tunnel rispettivamente a 2 e 3 corsie, con un'altezza fissa di 7,2 m. Sia la larghezza dei portali che l'altezza possono cambiare la superficie trasversale della galleria, che varia da 83 m^2 a 123 m^2.

- **Dimensione dell'incendio**: la dimensione dell'incendio, definita in termini di HRR nella galleria, ha assunto tre valori diversi: 4 MW, 8 MW e 15 MW con lo stesso HRRPUA di 1 MW/m^2. Studi recenti hanno infatti mostrato che l'HRR di picco di un incendio di un veicolo leggero in una galleria è compreso tra 0,56 MW e 15 MW.
  Un HRR più alto genera forti correnti convettive che mescolano l'aria calda con l'aria fredda, riducendo le differenze di temperatura e rompendo la stratificazione. Un HRR più debole, d'altra parte, permette alla stratificazione di durare più a lungo.



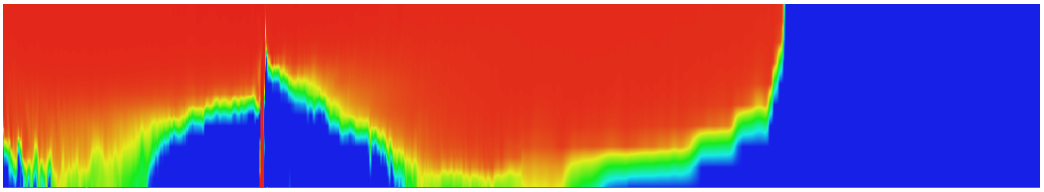
(a) Scenario 1: HRR pari a 4 MW, pendenza 1.5%, posizione centrale, sezione 83 m^2, istante 310.



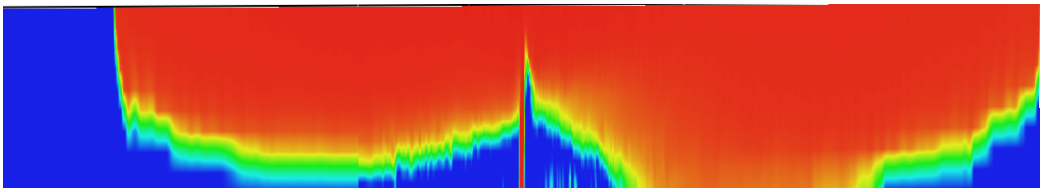
(b) Scenario 2: HRR pari a 15 MW, pendenza 1.5%, posizione centrale, sezione 83 m^2, istante 310.

Figura 11: Grafici di visibilità tra 0 e 30 m. Lo Scenario 1 rappresenta un incendio con HRR pari a 4 MW, dove si osserva una stratificazione del fumo più stabile e duratura a causa della minore intensità delle correnti convettive. Lo Scenario 2, invece, simula un incendio con un HRR di 15 MW, dove la potenza termica più elevata genera forti correnti convettive che mescolano l'aria calda e fredda, rompendo la stratificazione e rendendo il profilo di temperatura meno uniforme lungo l'altezza della galleria.

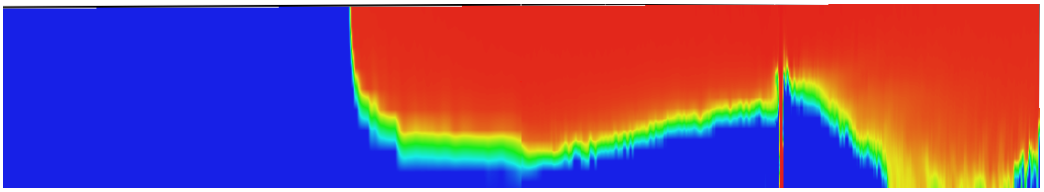
- **Posizione dell'incendio**: sono state simulate tre posizioni della fonte di incendio, con il fuoco posto al centro, al 25% della galleria o al 75% della stessa. La posizione dell'incendio influisce sull'effetto camino e quindi sulla distribuzione del fumo lungo la galleria e sulla stratificazione.



(a) Scenario 1: HRR 8 MW, pendenza 0.5%, sezione 83 m^2, posizione 25% della galleria, istante 430.



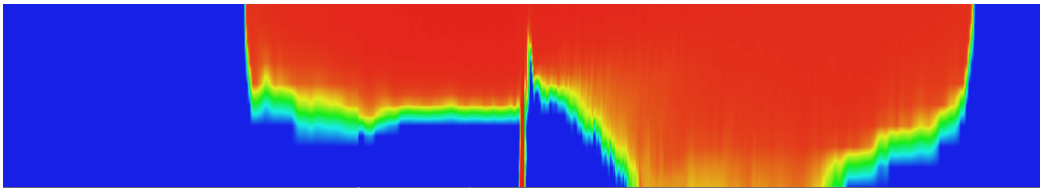
(b) Scenario 2: HRR 8 MW, pendenza 0.5%, sezione 83 m^2, posizione centrale, istante 430.



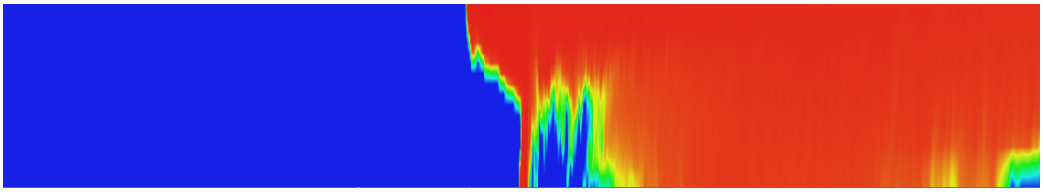
(c) Scenario 3: HRR 8 MW, pendenza 0.5%, sezione 83 m^2, posizione 75% della galleria, istante 430.

Figura 12: Grafici di visibilità tra 0 e 30 m. Lo Scenario 1, all'ingresso della galleria, mostra un flusso direzionale del fumo; lo Scenario 2, al centro, genera un doppio gradiente termico con fumo propagato in entrambe le direzioni; lo Scenario 3, alla fine della galleria, mostra il fumo orientato verso l'estremità opposta.

- **Pendenza**: la pendenza della galleria condiziona la stratificazione del fumo tramite l'effetto camino. La simulazione di pendenze variabili (0.5%, 1.5%, 3%, 5%) ha dimostrato come l'inclinazione possa accelerare la risalita del fumo verso le zone più alte della struttura (vedi Figura 13).



(a) Scenario 1: HRR 8 MW, pendenza 1.5%, sezione 83 m^2, posizione 50% della galleria, istante 300 s.



(b) Scenario 2: HRR 8 MW, pendenza 5%, sezione 83 m^2, posizione 50% della galleria, istante 300 s.

Figura 13: Grafici di visibilità tra 0 e 30 m. Lo Scenario 1 rappresenta una pendenza dell'1.5%, dove il fumo si stratifica ma mantiene una distribuzione relativamente uniforme lungo la galleria. Lo Scenario 2 mostra una pendenza del 5%, dove l'effetto camino è più pronunciato, accelerando il movimento del fumo verso l'estremità più alta della struttura.

I valori di questi parametri sono riassunti nella Tabella 1.

Parametri	**Intervallo di Valori**
Posizione dell'incendio	Al centro, al 25% e al 75% della galleria
Pendenza	0.5%, 1.5%, 3%, 5%
Dimensione dell'incendio (HRR)	4 MW, 8 MW, 15 MW
Area sezione trasversale	83 m^2, 100 m^2
Altezza galleria	7.2 m
Larghezza galleria	12.8 m, 16 m

Tabella 1: Parametri variati nella simulazione del modello d'incendio.

4.3 Pre processing

La simulazione fluidodinamica si suddivide in diverse fasi, ognuna delle quali è fondamentale per ottenere risultati accurati e affidabili. Prima di arrivare alla risoluzione vera e propria delle equazioni che descrivono il moto dei fluidi e i fenomeni fisici coinvolti è necessario:

- definire il dominio computazionale che delimita lo spazio in cui si svolgerà la simulazione;
- creare la mesh, cioè la griglia di calcolo che suddivide il dominio in piccole celle;
- impostare delle proprietà dei fluidi e dei fenomeni fisici e chimici che influenzeranno il comportamento del sistema;
- definire delle condizioni al contorno, che descrivono come il sistema interagisce con l'ambiente esterno.

4.3.1 Definizione del dominio computazionale e mesh di calcolo

Un elemento cruciale nelle simulazioni di incendio è la corretta scelta delle dimensioni della mesh, che devono essere adeguate alla scala del fuoco e ai fenomeni fisici coinvolti. L'HRR gioca un ruolo centrale in queste simulazioni, poiché fornisce una misura diretta dell'intensità dell'incendio e influisce sulla dinamica dei flussi convettivi, la produzione di fumo e il trasferimento di calore. Più nello specifico, il valore dell'HRR determina la quantità di combustibile che brucia per unità di tempo e, di conseguenza, la quantità di fumo prodotta.
L'HRR $\dot{Q}$ è espresso in termini di potenza termica (KW o MW) attraverso la seguente relazione:

$$\dot{Q} = m \cdot \Delta H_c,$$

dove:

- m è la massa di combustibile bruciato per unità di tempo (kg/s),
- ΔH_c rappresenta il calore di combustione (energia rilasciata per unità di massa del combustibile bruciato).

La dimensione caratteristica della fiamma, rappresentata dal diametro D^*, è strettamente correlata all'HRR dalla seguente formula:

$$D^* = \left(\frac{\dot{Q}}{\rho_\infty c_p T_\infty g} \right)^{2/5},$$

dove:

- ρ_∞ è la densità dell'aria;
- c_p è la capacità termica specifica dell'aria;
- T_∞ è la temperatura dell'aria ambiente;
- g è l'accelerazione gravitazionale.

Questa grandezza influisce direttamente sulla risoluzione della mesh, ovvero sulla dimensione delle celle Δx, la cui scelta è fondamentale per stabilire un equilibrio tra l'accuratezza della simulazione e il costo computazionale.
Per garantire una corretta rappresentazione del comportamento del fuoco, il rapporto tra il diametro caratteristico della fiamma D^* e la dimensione della cella Δx, indicato come N, deve essere scelto con attenzione:

$$N = \frac{D^*}{\Delta x}.$$

Le linee guida dell'FDS 5 User Guide [18] suggeriscono che N dovrebbe essere compreso tra 4 e 16. Valori più elevati di N (tra 10 e 16) comportano una maggiore copertura dell'incendio con un numero maggiore di celle, il che si traduce in una risoluzione più dettagliata, ma anche un aumento del costo computazionale, poiché sono richieste più risorse per risolvere le equazioni del flusso. Per simulazioni meno precise, un N compreso tra 4 e 8 può essere considerato adeguato. Il valore ottimale di N garantisce un compromesso accettabile tra accuratezza della simulazione e costi computazionali.
La corretta impostazione delle dimensioni della griglia di calcolo in relazione all'HRR si è dimostrata cruciale per ottenere simulazioni esatte ed efficienti dal punto di vista computazionale.
In particolare, la scelta di un passo di griglia più fine, come nel caso di simulazioni con un HRR di 4 MW, ha portato a un significativo aumento del tempo di calcolo: il tempo di simulazione dell'incendio ha raggiunto circa 24 h. Al contrario, quando si è optato per un passo di griglia più ampio, ad esempio nelle simulazioni con HRR di 8 MW e 15 MW, che avevano una dimensione di cella di 0.4 m nella mesh contenente il focolaio, il tempo di simulazione si è ridotto a circa 10-11 h. Questo dimostra come la scelta del passo di griglia influenzi non solo la precisione della simulazione, ma anche i

costi computazionali, e come sia cruciale trovare un equilibrio adeguato tra risoluzione e durata del calcolo.

Tabella 2: Calcolo del rapporto $N = \frac{D^*}{\Delta x}$ per i diversi HRR simulati.

HRR (MW)	**Diametro caratteristico D^* (m)**	**Dimensione cella Δx (m)**	**Rapporto $N = \frac{D^*}{\Delta x}$**
4 MW	1.5 m	0.2 m	7.5
8 MW	2.2 m	0.4 m	5.5
15 MW	2.8 m	0.4 m	7.0

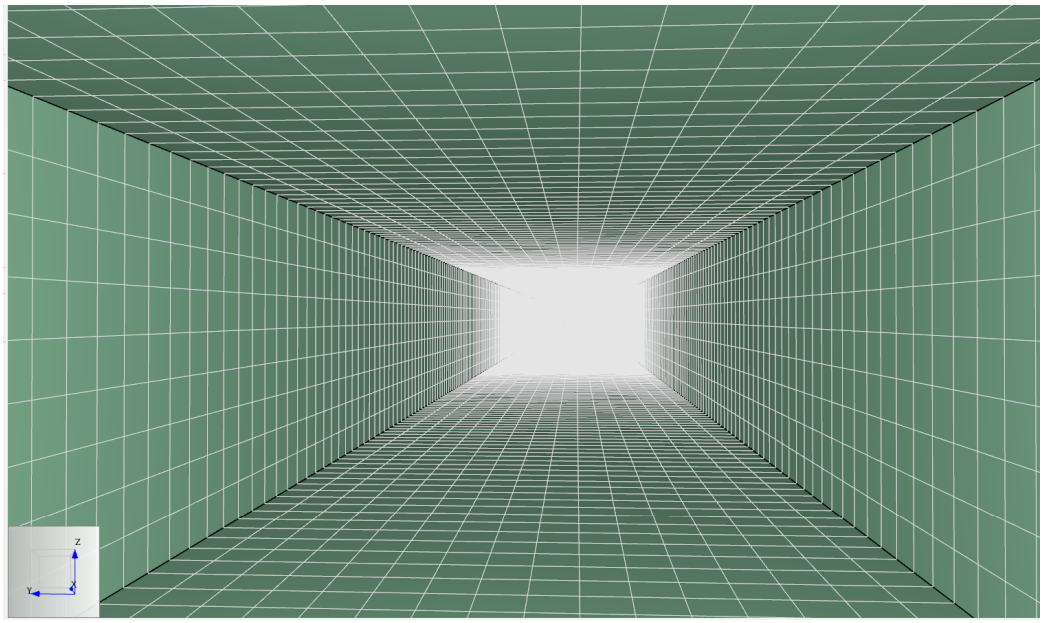


Figura 14: Mesh di calcolo.

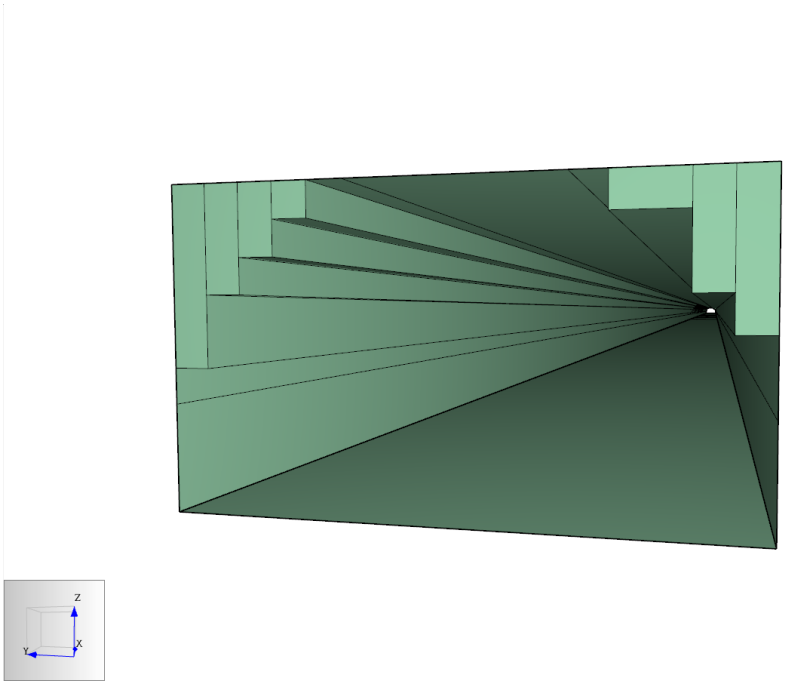


Figura 15: Struttura della galleria di studio.

Una galleria, con le sue pareti, racchiude un volume che può essere idealmente paragonato a un parallelepipedo. Questo volume è chiuso su quattro lati, mentre i restanti due, corrispondenti ai portali, rimangono aperti, consentendo il passaggio dei flussi d'aria, fondamentali per il comportamento dei tunnel. Il modello geometrico è stato realizzato definendo accuratamente le superfici che delimitano il dominio del fluido da analizzare. La galleria, costruita in calcestruzzo, è stata rappresentata all'interno di un dominio a forma di parallelepipedo con due diverse dimensioni: nel primo caso $12.8\,m \times 7.2\,m \times 1000\,m$ e nel secondo caso $16\,m \times 7.2\,m \times 1000\,m$. Sono state posizionate due VENT in corrispondenza dei portali, per considerare che lungo l'asse y il dominio fosse aperto. Sono state impostate 5 sezioni 2D nel piano y per registrare la visibilità, la temperatura e la concentrazione di CO, allo scopo di visualizzare l'evoluzione delle condizioni di visibilità sul profilo verticale, parametri fondamentali da ottenere in output dalla simulazione.

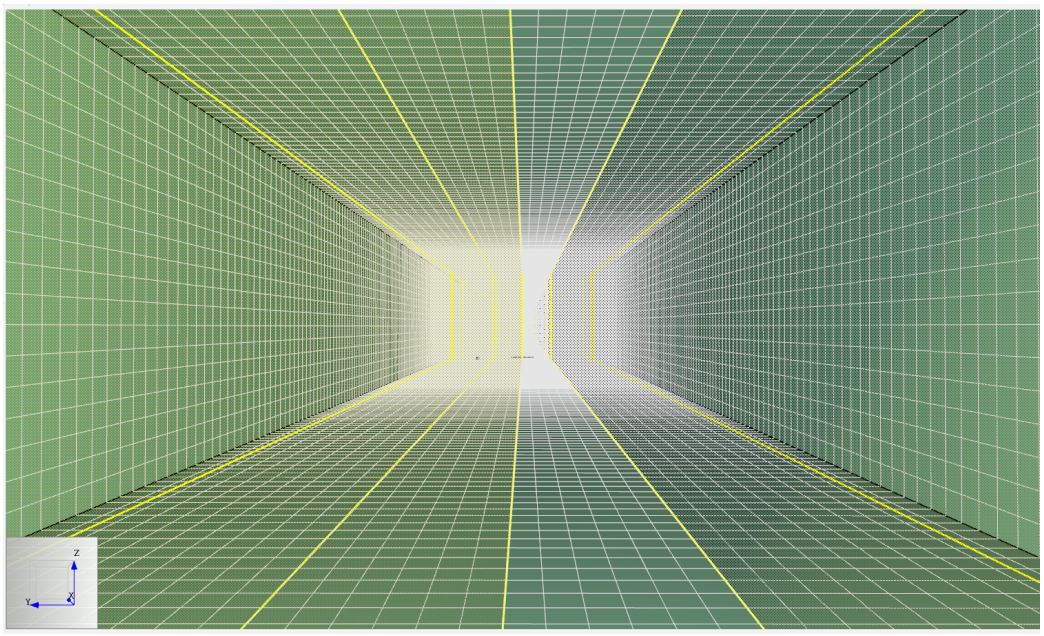


Figura 16: Piano 2D per misurare visibilità, CO e temperatura.

4.3.2 Simulazione di free burning

Al centro della galleria è stata posizionata una VENT di tipo BURNER, progettata per simulare un incendio attivo rilasciando calore. Questa VENT è configurata per mantenere costante il tasso di rilascio di calore per unità di superficie, noto come HRRPUA (Heat Release Rate per Unit Area), che rappresenta la quantità di calore rilasciata per metro quadrato di superficie bruciante.
La relazione che lega il tasso di rilascio di calore totale all'area bruciante e all'HRRPUA è espressa dalla formula:

$$HRR = S \times HRRPUA.$$

dove:

- HRR è il tasso totale di rilascio di calore (in kW o MW),
- S è la superficie della VENT (in m^2),
- HRRPUA è il tasso di rilascio di calore per unità di area (in kW/m^2).

In un modello di **free burning**, la combustione viene simulata come un processo continuo ed espansivo, senza tenere conto dell'esaurimento del combustibile o delle variazioni delle condizioni ambientali. Questo

modello presuppone che il fuoco continui a bruciare a un tasso costante, indipendentemente dalla disponibilità di combustibile o da altri fattori esterni.
Poiché l'HRRPUA è mantenuto costante, un aumento della superficie attiva S comporta un incremento proporzionale del tasso totale di rilascio di calore. Variando la superficie della VENT si ha un controllo diretto sull'intensità dell'incendio senza cambiarne il tasso di combustione per unità di superficie, garantendo così la coerenza del comportamento complessivo del fuoco. Inoltre, è importante sottolineare che la risoluzione della mesh attorno alla VENT ha un impatto significativo sulla precisione della simulazione: una griglia troppo grossolana potrebbe non rappresentare correttamente la dinamica del calore e del fumo generati dal fuoco, mentre una griglia più fine garantisce una migliore accuratezza nella simulazione del trasporto di calore e gas all'interno del dominio.

4.3.3 Condizioni a contorno

Il confine termico delle superfici del dominio è impostato come parete predefinita con una temperatura ambiente di 25°C, che considera il trasferimento di calore tra le pareti e l'aria circostante. Analogamente, la pressione è assunta pari a quella ambiente, con un valore di 1,01325E+5 Pa. Le **open boundary conditions** (condizioni al contorno aperte) sono applicate in corrispondenza dei portali, permettendo uno scambio continuo tra l'interno della galleria e l'ambiente esterno. Queste condizioni consentono al fumo, al calore e alle onde di pressione di uscire dal dominio di calcolo, evitando riflessioni o disturbi indesiderati alle estremità del dominio. Quest'ultime sono infatti generate dal rilascio di calore da parte dell'incendio. La rapida espansione dei gas provoca la nascita di onde di pressione, che si propagano molto più velocemente rispetto ai flussi di fumo e calore, poiché la loro velocità è legata a quella del suono nell'aria. A causa di questa differenza di velocità, le onde di pressione possono raggiungere le estremità di una galleria di dimensioni ridotte in tempi molto brevi, generando una rapida variazione della pressione interna.

4.4 Formazione del dataset

Dopo la simulazione, i risultati e i parametri dello scenario d'incendio vengono utilizzati per creare un ampio database destinato all'addestramento del modello di AI. I dati sulla visibilità registrati nella sezione verticale situata al centro della galleria lungo l'asse y vengono esportati dai risultati della modellazione e visualizzati tramite Smokeview. Con una durata dell'incendio

di 900 s e una frequenza di esportazione di un fotogramma ogni 10 s, si ottengono 91 campioni per ogni scenario. In totale, si contano 6552 campioni per i 72 scenari di incendio simulati.
I parametri del modello CFD, quali le informazioni sul tunnel e sull'incendio, tra cui l'area della sezione trasversale e la pendenza della galleria, la dimensione dell'incendio e la sua posizione, vengono utilizzati come input per il modello di AI. Data la natura dinamica del movimento del fumo durante un incendio, il tempo trascorso dall'accensione diventa un parametro aggiuntivo da considerare nella simulazione. I dati vengono organizzati in una grande tabella, dove ogni colonna rappresenta un parametro di input e ogni riga rappresenta un vettore di caratteristiche identificativo di uno scenario, corredato dall'istante temporale in cui è osservato.
Un vettore di input contenente anche il tempo specifico, insieme a una matrice di output corrispondente allo scenario d'incendio, formano una coppia di campioni di addestramento. Ogni colonna della tabella di input e ogni fotogramma della matrice di output vengono normalizzati nell'intervallo [0,1] utilizzando la funzione di normalizzazione min-max. Questo processo è essenziale affinchè tutte le feature abbiano la stessa influenza sull'output della rete.
Successivamente, gli indici identificativi di ogni scenario vengono riordinati casualmente e suddivisi in dataset di addestramento, validazione e test con un rapporto di 0.64, 0.16 e 0.20 rispettivamente, in modo tale che la rete si alleni su determinati scenari e venga testata su situazioni inedite. Il dataset di addestramento viene appunto utilizzato per addestrare il modello, quello di validazione per valutarlo durante l'addestramento, mentre il dataset di test per stimare la qualità del modello su campioni non visti dopo l'addestramento.
Durante il processo di addestramento, i campioni non vengono letti singolarmente dalla rete per aggiornare i coefficienti iterativamente, né vengono caricati tutti contemporaneamente, a causa dell'esiguità della memoria fisica del computer. Per bilanciare l'efficienza dell'addestramento e la limitatezza delle risorse, essi vengono raggruppati in mini-batch. Nel caso in questione, l'addestramento è stato eseguito su un computer con 8 GB di memoria, e il 60% del dataset (3931 campioni), destinato al training, è stato suddiviso in 67 batch da 64 elementi ciascuno.

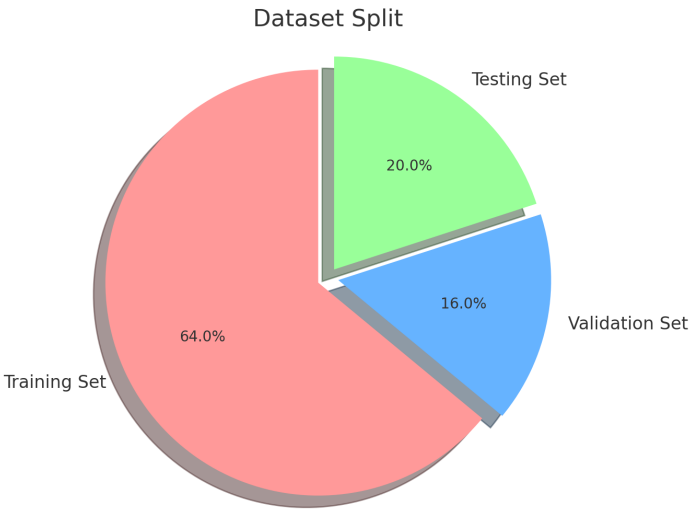


Figura 17: Dataset Split.

4.5 Architettura della rete

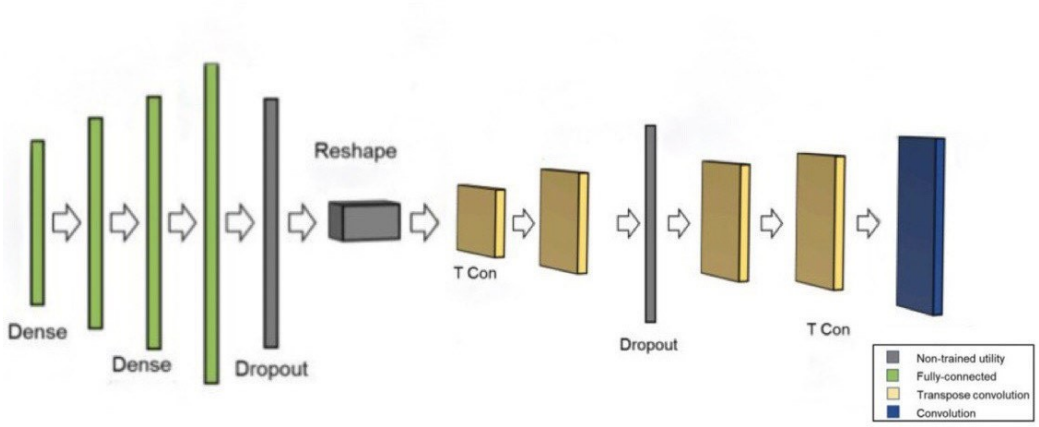


Figura 18: Modello di deep learning composto da layer di tipo fully connected, TCNN e CNN. Immagine tratta da [1].

Il modello di deep learning è progettato per iniziare senza alcuna conoscenza preliminare del calcolo della visibilità all'interno di una galleria e acquisire tale competenza attraverso un processo di addestramento continuo. Studi precedenti hanno già dimostrato la capacità delle reti neurali con strati di convoluzione trasposta di calcolare la distribuzione spaziale della

temperatura in uno stato stazionario all'interno di ambienti chiusi [9]. Partendo da questo approccio, si è valutata l'efficacia di un modello che unisce strati densi e convoluzionali di tracciare l'evoluzione di una grandezza fisica dal comportamento analogo. Questa architettura si è dimostrata particolarmente adatta per compiti che richiedono l'apprendimento e la generazione di dati spaziali su larga scala, come la simulazione di fenomeni fisici legati alla sicurezza antincendio.

Strati Densi Preliminari La rete inizia con una sequenza di strati densi, soggetti a una progressiva espansione nel numero di unità. I cinque parametri relativi all'ambiente della galleria e alle caratteristiche dell'incendio che si sono rivelati più influenti nell'evoluzione dei fumi - ovvero l'area e la pendenza della galleria, l'HRR, la posizione del fuoco e l'istante dallo scoppio dell'incendio in cui si considera lo scenario - vengono raccolti in un vettore e forniti alla rete. Le informazioni nascoste nell'input vengono arricchite dai successivi 4 strati densi, che agiscono per estrarre ed apprendere interazioni non lineari tra queste caratteristiche, espandendo la rappresentazione.
Poiché questa combinazione di layer tende ad avere un gran numero di parametri, aumentando la possibilità di overfitting, è stato inserito uno strato di Dropout alla fine della sequenza di strati densi. Applicato con tassi moderati, esso riveste un ruolo particolarmente utile sulle rappresentazioni altamente dimensionali, riducendo l'attivazione di neuroni specifici e promuovendo una migliore generalizzazione.

Ridimensionamento e Trasformazione Spaziale Dopo la fase di elaborazione negli strati densi preliminari, i dati devono essere convertiti in una rappresentazione che permetta un'analisi spaziale dettagliata. In questa fase, uno strato di Reshape modifica la struttura dell'output proveniente dagli strati densi, trasformandolo da una forma vettoriale in una matrice bidimensionale (2D), interpretabile come un'immagine. Questo passaggio è essenziale per abilitare la successiva elaborazione convoluzionale, poiché consente alla rete di passare da una rappresentazione astratta e globalmente densa delle caratteristiche a una distribuzione spaziale di queste stesse informazioni.

Strati Convoluzionali Trasposti Quattro strati convoluzionali trasposti costituiscono il cuore della rete. Ogni layer di questo tipo lavora per raffinare ulteriormente i dettagli delle feature map in output dallo strato precedente, migliorando la precisione con cui viene simulata la distribuzione del fumo. Nei primi strati, l'utilizzo di kernel di dimensioni maggiori (ad esempio

5×5) e con più canali consente di applicare filtri con un campo ricettivo più ampio, catturando le informazioni globali necessarie dalla rappresentazione compressa. Man mano che l'immagine viene riportata a una risoluzione maggiore, è più funzionale concentrarsi su un numero minore di canali, ma più dettagliati, per migliorare la definizione dei particolari. Coerentemente, kernel di dimensioni più piccole (ad esempio, 3×3 o 1×1) sono utilizzati per affinare bordi e texture.
Il Dropout posto tra gli strati convoluzionali trasposti svolge una funzione cruciale nella regolarizzazione del modello. In questa fase, l'obiettivo è prevenire la specializzazione eccessiva di determinati canali, garantendo che la rete mantenga la capacità di generalizzare su scenari diversi. In altre parole, il dropout aiuta a distribuire il focus della rete su un insieme più ampio di neuroni, impedendo che essa si concentri eccessivamente su dettagli specifici della mappa di caratteristiche che potrebbero non essere rilevanti o che potrebbero portare a un modello eccessivamente complesso e non generalizzabile. Ciò ha anche un duplice vantaggio in termini di complessità computazionale, che viene ridotta nelle fasi finali quando i dettagli sono già stati ricostruiti a una risoluzione più alta, e di regolarizzazione, diminuendo il rischio di overfitting.

Strato di Output Infine, la rete termina con uno strato convoluzionale che genera un'immagine di visibilità RGB. Questa rappresenta la distribuzione finale del fumo all'interno della galleria, fornendo una raffigurazione della visibilità su tutto il dominio simulato.

4.6 Funzioni di attivazione: ReLU e Sigmoid

In questo studio sono state adottate le funzioni di attivazione ReLU e Sigmoid per mappare gli input attraverso i vari strati della rete. La funzione di attivazione ReLU è stata impiegata negli hidden layers, mentre la funzione Sigmoid è stata scelta per l'ultimo strato della rete, che produce l'immagine finale.
La ReLU, introdotta per la prima volta in una rete dinamica da Hahnloser et al. [7] nel 2011, è un metodo efficace per migliorare l'addestramento delle reti profonde rispetto ad altre funzioni di attivazione ampiamente utilizzate, diventando la scelta più popolare per gli strati convoluzionali. La funzione di attivazione ReLU è stata selezionata principalmente per la natura dei dati di input che, essendo per lo più strettamente positivi, la rendono particolarmente adatta. La ReLU è infatti definita come:

$$\text{ReLU}(x) = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{se } x \leq 0 \end{cases},$$

Il che significa che gli input a destra dello zero rimangono invariati. Questo comportamento evita qualsiasi distorsione o trasformazione indesiderata dei dati all'inizio del processo di elaborazione, permettendo alla rete di propagare informazioni cruciali attraverso i primi strati senza perdita di rilevanza. Sebbene, in un primo momento, si comporti in maniera lineare, l'effetto cumulativo del suo utilizzo attraverso più livelli introduce la non linearità necessaria, consentendo alla rete di apprendere relazioni complesse tra i dati. Un altro motivo fondamentale per l'uso di questa funzione, già introdotto nella sezione 2.3.3., è la sua capacità di prevenire il vanishing gradient, un problema comune nelle reti profonde che si manifesta soprattutto in presenza di funzioni di attivazione come Tanh e Sigmoid, che causano una diminuzione del valore del gradiente durante la propagazione all'indietro, rallentando o bloccando l'apprendimento, da cui ReLU è esente, grazie alla struttura della funzione che la definisce.
Tuttavia, proprio poiché l'uscita della ReLU non è limitata, potrebbe essere soggetta al fenomeno di exploding gradient, ma impostando i pesi iniziali in modo appropriato, i gradienti riescono a mantenere una magnitudine ragionevole e la convergenza è raggiunta rapidamente.
Per garantire un addestramento stabile ed efficiente della rete, la scelta dell'inizializzazione dei pesi è ricaduta sulla tipologia He Uniform, progettata specificamente per funzionare in sinergia con ReLU [23]. I pesi sono campionati da una distribuzione uniforme nell'intervallo [-a, a], dove n rappresenta il numero di neuroni di input che alimentano il layer e $a = sqrt(6/n)$ è derivato dall'analisi matematica della funzione di attivazione ReLU. Impostando l'intervallo in questo modo, l'inizializzazione He Uniform consente alle attivazioni di mantenere una varianza non nulla, prevenendo il problema dei "dead neurons", di cui la ReLU soffre.
ReLU permette quindi di gestire efficacemente gli input positivi, introducendo la non linearità necessaria per l'apprendimento di pattern complessi, mentre l'inizializzazione He Uniform garantisce stabilità durante l'addestramento, assicurando un apprendimento efficace.
Per l'ultimo strato della rete, che genera l'immagine finale è stata scelta la funzione di attivazione Sigmoid. La funzione Sigmoid è definita come:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}},$$

è ampiamente utilizzata in contesti in cui l'output deve essere interpretato come una probabilità o un valore normalizzato compreso tra 0 e 1. La scelta della Sigmoid per l'output finale è stata guidata dalla necessità di mappare i valori dell'immagine generata in un intervallo compreso tra 0 e 1, corrispondente ai valori normalizzati delle immagini target. Questo approccio assicura che l'output della rete rimanga all'interno dei limiti previsti, garantendo che il risultato sia coerente con le etichette attese, e salvaguardia la stabilità durante la fase di backpropagation, grazie alla differenziabilità della funzione.

4.7 Scelta degli iperparametri

La scelta degli iperparametri è una fase cruciale nella progettazione di modelli di deep learning: una configurazione subottimale può compromettere l'efficacia del modello sia in termini di stabilità dell'apprendimento che di capacità di generalizzazione, indipendentemente dalla qualità dei dati di addestramento. Tuttavia, la selezione degli iperparametri rappresenta una sfida complessa, poiché equivale a risolvere un problema di ottimizzazione ad alta dimensionalità, la cui regolarità è spesso sconosciuta. Questo processo richiede una combinazione di esperienza, conoscenza del dominio e sperimentazione. Di conseguenza, la ricerca ha concentrato notevoli sforzi sullo sviluppo di tecniche automatizzate, come Hyperband, Sequential Model-based Algorithm Configuration (SMAC), algoritmi genetici e ParamILS, senza escludere metodi più tradizionali come Random Search e Bayesian Optimization [2]. Nell'ambito di questa tesi, gli iperparametri relativi al numero di unità nascoste, alla dimensione del kernel e al tasso di apprendimento sono stati ottimizzati utilizzando l'algoritmo Hyperband.

4.7.1 Funzionamento di Hyperband

La ricerca degli iperparametri richiede un bilanciamento tra l'esplorazione di un'ampia gamma di configurazioni e l'allocazione delle risorse necessarie per valutare correttamente ciascuna di esse. Hyperband affronta questo problema combinando i vantaggi della Random Search e della tecnica di Successive Halving. Il primo consente di indagare molti possibili set di iperparametri, mentre il secondo gestisce l'allocazione delle risorse in modo adattivo: progressivamente riduce il numero di configurazioni da testare e aumenta le risorse (le epoche di addestramento) assegnate a quelle più promettenti. Questo processo si ripete fino a raggiungere un insieme ottimale di iperparametri.

Entrando nel dettaglio, Hyperband vuole in input tre parametri chiave:

- **min_epochs**: indica il numero minimo di epoche per cui ogni configurazione verrà addestrata prima di essere valutata. Il valore predefinito è 1.
- **max_epochs**: definisce il numero massimo di epoche di addestramento. In questo caso, **max_epochs** è impostato a 15 per garantire che le configurazioni più promettenti ricevano risorse sufficienti.
- **factor**: stabilisce il fattore di riduzione nel processo di Successive Halving. Con **factor** fissato a 3, ad ogni iterazione il numero di configurazioni viene ridotto di un terzo, mentre le risorse allocate a quelle rimanenti vengono triplicate.

L'algoritmo si articola in una struttura a due livelli: i cicli esterni, chiamati *brackets*, e i cicli interni. Ogni *bracket* rappresenta un ciclo che gestisce una quantità iniziale diversa di risorse e configurazioni.

- Nel primo bracket, tutte le configurazioni vengono addestrate per 1 epoca; dopo ogni ciclo, si eliminano le configurazioni meno performanti, riducendone il numero di circa un terzo e si riallenano quelle rimaste con un numero di epoche triplicato. Questo processo continua fino a raggiungere il limite massimo di 15 epoche, momento in cui si individua la configurazione migliore.
- Nel secondo bracket, invece, le configurazioni partono con un numero maggiore di risorse, in questo caso 3 epoche di addestramento. Come nel primo bracket, le configurazioni con le peggiori performance vengono scartate e quelle sopravvissute vengono riallenate per 9 epoche, continuando il processo fino a individuare la configurazione migliore.
- Brackets successivi: ogni nuovo bracket parte con un numero inferiore di configurazioni rispetto ai precedenti, ma con un'allocazione iniziale di risorse maggiore (ad esempio, 9 epoche).

In sintesi, in ogni bracket si inizia con una quantità di risorse diversa e si riduce progressivamente il numero di configurazioni, allocando più risorse a quelle che mostrano le migliori performances.

4.7.2 Iperparametri ottimizzati nella rete

Nella fase di ottimizzazione degli iperparametri tramite *Keras Tuner Hyperband*, sono stati definiti intervalli per i principali iperparametri, quali

il numero di unità nei layer densi, la dimensione del kernel nei layers convoluzionali e il *learning rate*, in modo da rispettare la struttura e la funzione di ciascun componente della rete.

Unità degli Strati Densi Gli intervalli relativi al numero di unità nei layer densi sono stati impostati in maniera crescente, coerentemente con la funzione di questi strati, che mirano a espandere progressivamente la rappresentazione dell'input e catturare interazioni non lineari.
- Nei primi strati densi, sono stati utilizzati intervalli più piccoli (ad esempio [16, 512], [64, 512]), con l'obiettivo di iniziare a estrarre le caratteristiche più rilevanti.
- Negli strati successivi, gli intervalli sono stati ampliati (ad esempio [256, 512], [512, 2048]), per consentire una maggiore complessità della rappresentazione: aumentando il numero di unità, la rete è in grado di apprendere meglio interazioni complesse tra le caratteristiche, necessarie per il compito di predizione.

Dimensione del kernel Gli intervalli relativi alla dimensione del kernel nei layer convoluzionali trasposti e al numero di unità sono stati impostati in modo decrescente lungo i vari strati, coerentemente con la necessità di focalizzarsi inizialmente su informazioni globali per poi passare alla risoluzione di dettagli più fini.
- Nei primi strati convoluzionali, sono stati definiti kernel più grandi (ad esempio [3, 5]), con l'obiettivo di catturare pattern spaziali su larga scala. Questo approccio è utile per ottenere una visione globale del fenomeno, cruciale nelle fasi iniziali dell'elaborazione.
- Man mano che il modello progredisce, sono stati utilizzati kernel più piccoli (nell'intervallo [1, 3]), per focalizzarsi su dettagli locali e affinare la rappresentazione. In questo modo, la rete è in grado di migliorare la risoluzione e i dettagli dell'immagine generata, ottimizzando la precisione della previsione.

Learning Rate Il learning rate è stato ottimizzato in un intervallo logaritmico compreso tra $1e^{-4}$ e $1e^{-2}$, per consentire una ricerca efficace tra valori piccoli e grandi. Valori più piccoli sono stati inclusi per garantire stabilità nella convergenza, mentre valori più grandi hanno permesso di esplorare la possibilità di un addestramento più rapido in casi specifici.

4.8 Mean Square Error e Custom Loss

4.8.1 Utilizzo Iniziale del Mean Squared Error

All'inizio del processo di sviluppo del modello per la generazione di immagini, è stato utilizzato il MSE come funzione di loss principale. Questa scelta è stata motivata dall'accuratezza che il MSE introduce, poiché la metrica permette di misurare direttamente la differenza quadratica media tra i pixel dell'immagine originale e quelli dell'immagine generata. In un primo momento, il MSE ha fornito risultati soddisfacenti, allineati con le prestazioni riportate in letteratura [1]. Tuttavia, in vista di applicazioni pratiche, come il calcolo dell'altezza dei fumi o la distanza tra la zona di propagazione del fuoco e lo scoppio dell'incendio, sono emerse alcune delle sue limitazioni, specialmente in termini di preservazione della struttura visiva complessiva.

4.8.2 Transizione alla Custom Loss con lo SSIM

Nell'ambito della generazione di immagini, lo SSIM nella funzione di loss ha dimostrato significativi vantaggi rispetto all'utilizzo del solo MSE [28], compensandone le limitazioni. Una combinazione convessa di queste metriche consente, infatti, di sfruttare i punti di forza di entrambe, migliorando la fedeltà visiva delle immagini prodotte dal modello. In particolare, una funzione di loss combinata con una ponderazione di $0.6 \times \text{MSE} + 0.4 \times \text{SSIM}$ permette di bilanciare la precisione numerica (fornita dal MSE) con la qualità percepita (assicurata dallo SSIM), offrendo al contempo una maggiore robustezza agli errori locali.
Il MSE fornisce una misura rigorosa della somiglianza punto per punto tra l'immagine predetta e quella di riferimento, concentrandosi sulla minimizzazione delle differenze assolute tra i pixel e garantendo che l'immagine generata sia la più vicina possibile a quella originale in termini di valore medio. Tuttavia, questa metrica non considera come queste differenze siano distribuite spazialmente nell'immagine e potrebbe, quindi, non riflettere accuratamente la percezione umana della qualità visiva: il MSE tratta tutte le differenze tra i pixel allo stesso modo, senza distinguere tra errori che si verificano in zone critiche, come i bordi o le transizioni, e quelli in aree più uniformi. Di conseguenza, un piccolo errore in una zona uniforme può essere penalizzato allo stesso modo di un errore lungo un bordo, nonostante sia molto più significativo per la visualizzazione e il calcolo dell'altezza dello strato dei fumi.
D'altra parte, lo SSIM si focalizza sulla preservazione della struttura visiva complessiva, valutando non solo le differenze in termini di intensità dei pixel, ma cercando anche di mantenere la coerenza delle relazioni spaziali, il che è

cruciale per garantire che l'immagine generata conservi un aspetto realistico e fedele all'originale. Lo SSIM tiene conto delle relazioni locali tra i pixel, garantendo che dettagli importanti, come i bordi degli oggetti o le transizioni tra aree con diverse proprietà visive, siano rappresentati in modo accurato, anche se vi sono piccole differenze di intensità e valutando queste discrepanze in modo più equilibrato rispetto al MSE.

5 Risultati ottenuti

Questa sezione si concentra sull'analisi dei risultati ottenuti dai modelli proposti. Inizialmente vengono introdotte le metriche utilizzate per valutare le prestazioni del modello, l'R-squared R^2 e l'indice di somiglianza strutturale SSIM, che vengono impiegate sia durante la fase di addestramento che in quella di test. Particolare attenzione è riservata all' R^2, utile per valutare l'accuratezza numerica del modello. In seguito, vengono descritte le architetture dei due modelli con un focus sugli iperparametri e sulle funzioni di loss impiegate. Vengono presentati grafici che mostrano l'andamento della loss, del R^2 e dello SSIM, permettendo di comprendere l'evoluzione del modello durante le fasi di training e validazione. La sezione 4.3 fornisce un confronto visivo e numerico tra i modelli, mettendo in luce i rispettivi punti di forza e debolezza. Nel contesto della classificazione delle aree di visibilità, la sezione 4.4 analizza le matrici di confusione, utili per verificare la capacità dei modelli di distinguere tra aree visibili e non visibili. Viene presentata un'analisi delle percentuali di falsi positivi e falsi negativi, che consente un confronto diretto tra le performance dei modelli. Infine, vengono presentati esempi visivi della distribuzione dei fumi a 600 s dall'inizio dell'incendio, momento cruciale per valutare la propagazione e le zone di pericolo. La sezione 4.5 conclude con un'analisi delle applicazioni pratiche, come il calcolo dell'altezza del fumo e della distanza percorribile prima che questo raggiunga il suolo, dimostrando l'importanza di tali misurazioni per la sicurezza delle persone all'interno della galleria.

5.1 Metriche di bontà del modello

Per valutare l'efficacia del modello durante l'addestramento e la fase di test, vengono impiegate due metriche fondamentali: l'R^2 e lo SSIM. Queste metriche forniscono una valutazione complessiva delle prestazioni del modello, consentendo di analizzare sia la precisione numerica che l'aspetto visivo delle immagini generate, ovvero forniscono un'analisi sia qualitativa che quantitativa.
Mentre la logica che governa la metrica SSIM è stata ampiamente trattata nella sezione 3.4.2, questa parte verterà sull'approfondimento dell' R^2.

R^2 squared L'R^2 viene comunemente utilizzato come indicatore della bontà di adattamento per i modelli di regressione, specialmente nel contesto delle immagini. Esso misura quanto bene il modello sia in grado di prevedere i valori dei pixel, in relazione alla varianza presente nell'immagine originale. L'espressione dell' R^2 è data da:

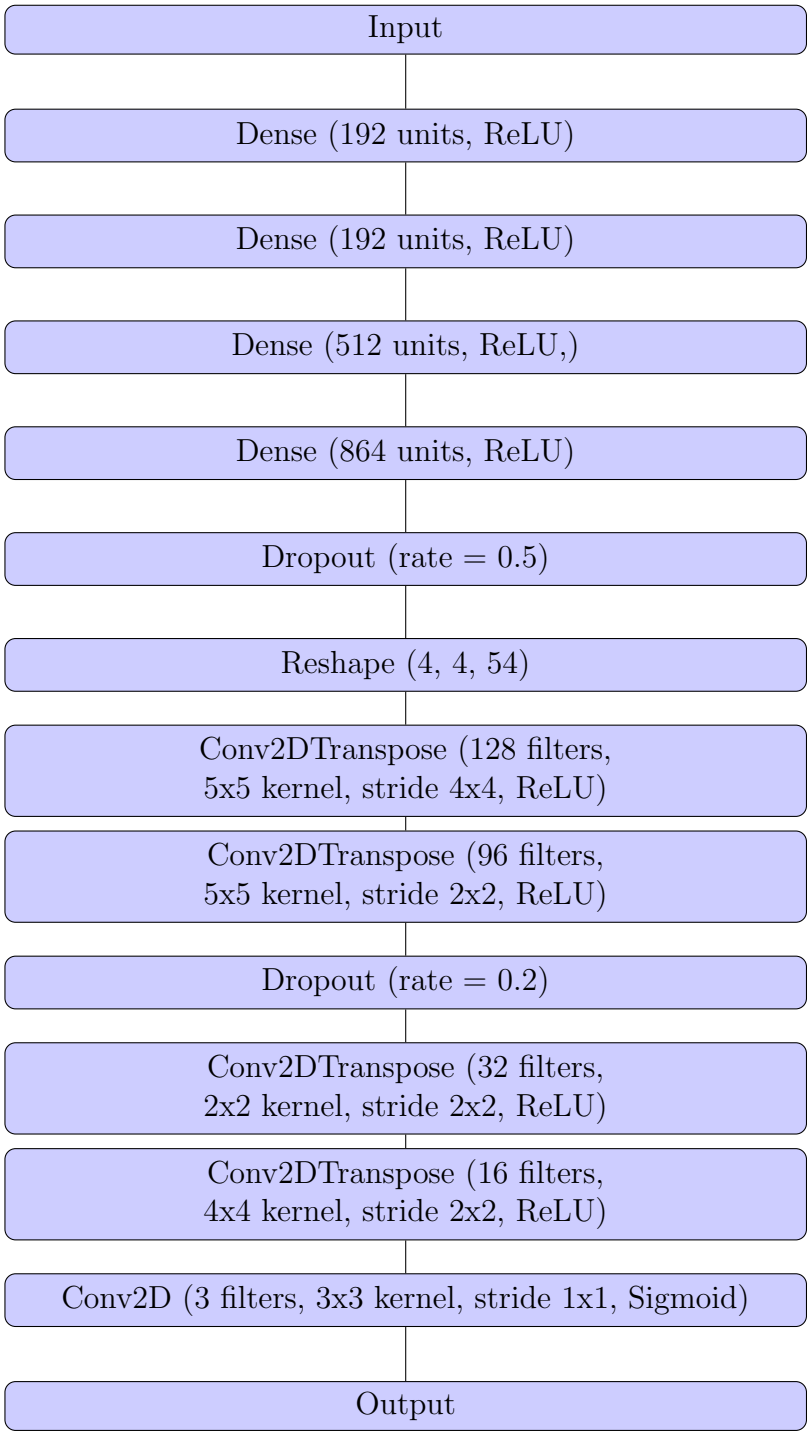
$$R^2 = 1 - \frac{\text{SSres}}{\text{SStot}},$$

dove:

- SSres è la somma dei quadrati dei residui, ovvero le differenze tra i valori dei pixel reali e quelli predetti;
- SStot rappresenta la somma dei quadrati delle differenze tra i valori reali e la loro media, ovvero la varianza dei pixel nell'immagine originale.

R^2 quantifica quindi quanto bene il modello riproduca i dettagli visivi di un'immagine, confrontando le previsioni con la varianza dell'immagine stessa. Un valore di R^2 vicino a 1 indica che il modello prevede con alta precisione i valori dei pixel, spiegando gran parte della varianza presente nell'immagine originale. Al contrario, un valore inferiore riflette una minore accuratezza predittiva, evidenziando quanto la varianza nell'immagine non venga catturata dalle previsioni del modello.
L'utilizzo congiunto di R^2 e SSIM permette di ottenere una valutazione più completa delle prestazioni del modello. Mentre l'R^2 fornisce una misura quantitativa della precisione predittiva, lo SSIM offre una valutazione qualitativa della fedeltà visiva delle immagini generate. Questa combinazione garantisce che il modello non solo preveda correttamente i valori dei pixel, ma anche che le immagini risultanti siano strutturalmente coerenti e visivamente simili a quelle originali. In questo modo, si assicura che il modello mantenga un equilibrio tra accuratezza numerica e qualità percettiva, essenziale per applicazioni in cui l'aspetto visivo è di primaria importanza.

5.2 Modello 1


Input
Dense (192 units, ReLU)
Dense (192 units, ReLU)
Dense (512 units, ReLU,)
Dense (864 units, ReLU)
Dropout (rate = 0.5)
Reshape (4, 4, 54)
Conv2DTranspose (128 filters,
5x5 kernel, stride 4x4, ReLU)
Conv2DTranspose (96 filters,
5x5 kernel, stride 2x2, ReLU)
Dropout (rate = 0.2)
Conv2DTranspose (32 filters,
2x2 kernel, stride 2x2, ReLU)
Conv2DTranspose (16 filters,
4x4 kernel, stride 2x2, ReLU)
Conv2D (3 filters, 3x3 kernel, stride 1x1, Sigmoid)
Output

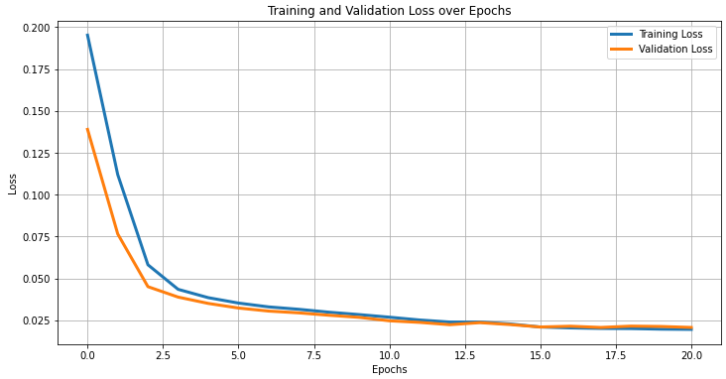


Figura 19: Training and Validation Loss over Epochs.

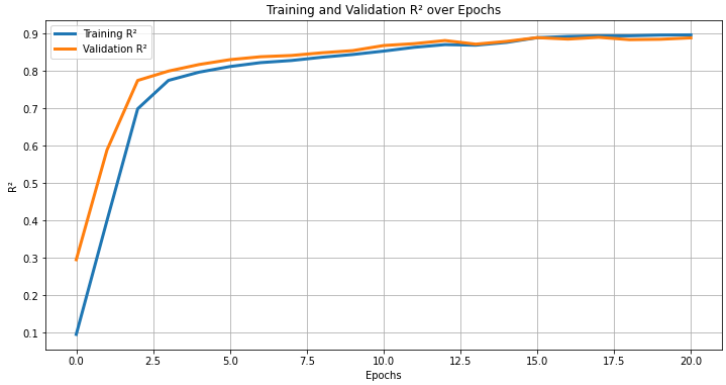
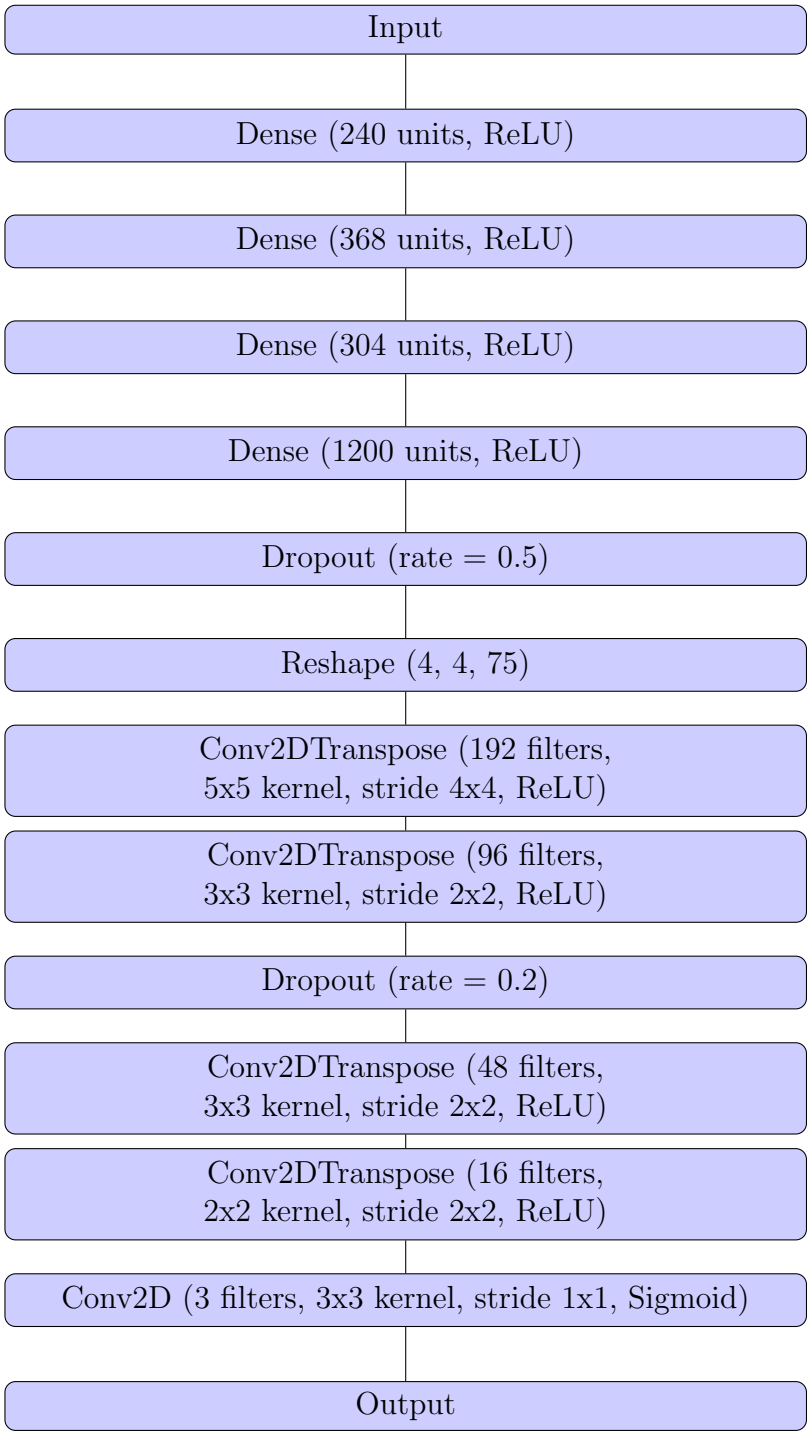


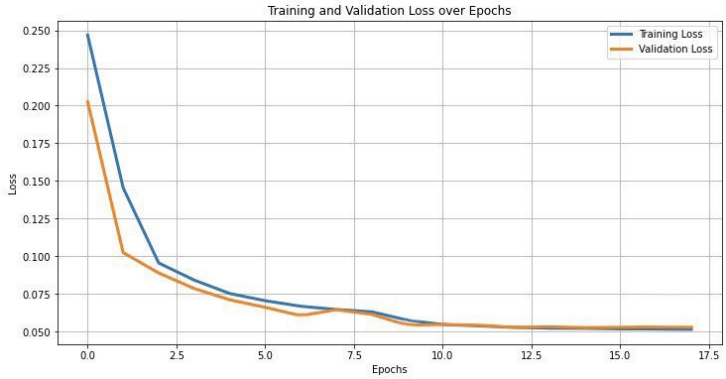
Figura 20: Training and Validation R^2 over Epochs.

Il primo modello dimostra un apprendimento rapido, come evidenziato dal forte calo della training loss nelle prime epoche. Dopo circa 13 iterazioni sull'intero dataset, la loss si stabilizza, indicando che il modello ha raggiunto la convergenza in un periodo di tempo relativamente breve, con un tempo di training complessivo di circa 14 *min*. Questo comportamento, con un apprendimento stabile e poche oscillazioni, riflette una buona scelta degli iperparametri, riducendo il rischio di overfitting o underfitting. Anche la validation loss segue un andamento parallelo, dimostrando che il modello è capace di generalizzare bene su dati non visti. L'apprendimento risulta stabile, senza oscillazioni significative o divergenze tra i set di training e validation, come confermato dalle metriche di bontà del modello: l'R^2 e lo SSIM.
L'R^2 di training cresce rapidamente, stabilizzandosi intorno a 0.90, mentre quello di validation si arresta a 0.89, evidenziando la buona capacità

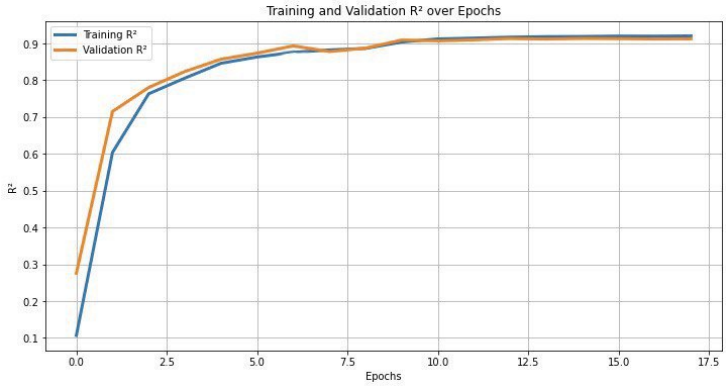
predittiva del modello anche sui dati di validazione. Per quanto riguarda lo SSIM, una metrica importante per i modelli di ricostruzione o generazione di immagini, il Modello 1 raggiunge un valore di 0.77 sul dataset di validazione. Sebbene questo indichi che il modello riesce a preservare le caratteristiche strutturali delle immagini, il risultato non è ottimale, lasciando spazio a miglioramenti.

5.3 Modello 2

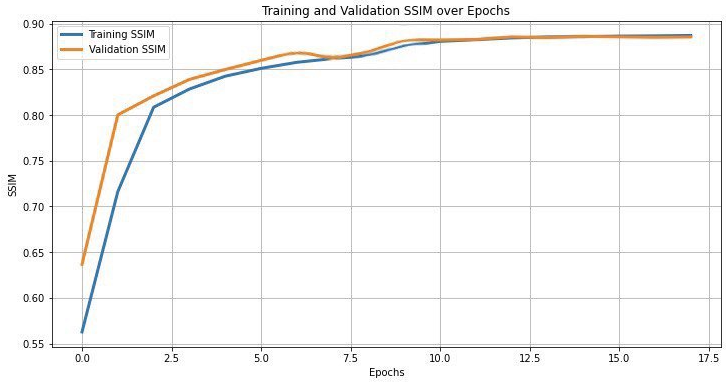

Input
Dense (240 units, ReLU)
Dense (368 units, ReLU)
Dense (304 units, ReLU)
Dense (1200 units, ReLU)
Dropout (rate = 0.5)
Reshape (4, 4, 75)
Conv2DTranspose (192 filters,
5x5 kernel, stride 4x4, ReLU)
Conv2DTranspose (96 filters,
3x3 kernel, stride 2x2, ReLU)
Dropout (rate = 0.2)
Conv2DTranspose (48 filters,
3x3 kernel, stride 2x2, ReLU)
Conv2DTranspose (16 filters,
2x2 kernel, stride 2x2, ReLU)
Conv2D (3 filters, 3x3 kernel, stride 1x1, Sigmoid)
Output



(a) Training and Validation Loss over Epochs



(b) Training and Validation R^2 over Epochs.



(c) Training and Validation SSIM over Epochs.

Anche il secondo modello dimostra una buona capacità di adattamento e generalizzazione, come evidenziato dall'andamento della validation loss, che si mantiene costantemente al di sotto della training loss. L'abbattimento avviene in modo analogo al primo modello, ma si raggiunge la convergenza leggermente prima, dopo circa 10 iterazioni complete sul dataset, con

un tempo di training complessivo di circa 10 *min*. Ciò conferma che il modello riesce ad apprendere rapidamente, bilanciando l'addestramento e la generalizzazione.
Le metriche di R^2 e SSIM riflettono una performance superiore rispetto al Modello 1. Infatti, il R^2 in fase di validazione arriva a toccare un valore di 0.92, un miglioramento rispetto allo 0.90 del primo modello. La differenza più significativa è osservata nella metrica SSIM, che raggiunge un valore di 0.89, dimostrando una qualità superiore nella generazione di immagini in termini di somiglianza strutturale, come si riscontra nella sezione 4.5 nei plot generati dal modello.
Questo risultato evidenzia i vantaggi derivanti dall'uso di una loss combinata che integra MSE e SSIM, permettendo di eccellere sia nella precisione che nella qualità strutturale delle immagini generate, rispetto a un approccio che si basa esclusivamente sul MSE.
Il Modello 2 non solo raggiunge migliori risultati nelle metriche, ma lo fa anche con una convergenza più rapida, rendendolo una scelta complessivamente più performante.

5.4 Matrici di confusione

Le seguenti matrici di confusione (vedi Figure 22 e 23) sono riferite alla visibilità tra 0 e 10 m a seguito dello scoppio dell'incendio. Avendo a che fare con classi sbilanciate, (**Visible**=positivo, **Non Visible**=negativo), si è adoperata una normalizzazione atta a mostrare la proporzione di ciascuna classe, fornendo un quadro più chiaro delle prestazioni del modello. La normalizzazione per colonna, in particolare, permette di valutare quanto le predizioni del modello corrispondano effettivamente alla classe reale, mettendo in luce eventuali confusioni tra le classi. Si parla di *falsi positivi* (FP) quando il modello predice "Visible" un pixel che, invece, è investito dal fuoco e quindi dovrebbe appartenere alla classe "Non Visible". Al contrario, i *falsi negativi* (FN) sono i pixel predetti come "Non Visible" quando, in realtà, appartengono a zone sicure per i pedoni. L'obiettivo del modello è quello di minimizzare la somma di falsi positivi e falsi negativi, garantendo così una maggiore accuratezza nella previsione della visibilità. Come si può osservare nella prima matrice di confusione, il Modello 1 mostra una tendenza significativa a confondere alcuni esempi di "Non Visible" con "Visible", con un tasso di errore dell'11% per la classe "Non Visible". Questo indica che commette più FN, predicendo che un pixel è "Non Visible" quando in realtà è "Visible", ovvero non investito dal fumo o dal fuoco.
In termini di FP, entrambi i modelli mostrano prestazioni simili, con un basso

tasso di errore del 4% per il primo e del 2% per il secondo, il che suggerisce che le predizioni delle aree visibili sono molto accurate. Questo è particolarmente importante in contesti di sicurezza, poiché riduce il rischio di sottovalutare le zone sicure per i pedoni.
Nel complesso, il Modello 1 risulta essere più conservativo rispetto al Modello 2, poiché predice erroneamente un pixel come "Non Visible" con una frequenza maggiore (11% contro 7%). Questa tendenza allarmista potrebbe portare a evacuazioni non necessarie o ad altre reazioni precauzionali, riducendo l'efficienza delle operazioni di gestione dell'emergenza.
La seconda matrice di confusione, relativa al Modello 2, mostra chiaramente una maggiore accuratezza nella classificazione dei "Non Visible", con un tasso di errore ridotto al 2%. In un contesto di sicurezza, come in un incendio, dichiarare che c'è visibilità quando non ce n'è potrebbe portare le persone a prendere decisioni sbagliate, come intraprendere un percorso pericoloso o ritardare l'evacuazione, sottovalutando la gravità della situazione. Per tale motivo il Modello 2 risulta più affidabile e accurato, oltre a dimostrare complessivamente una miglior capacità nel discriminare tra le due classi.

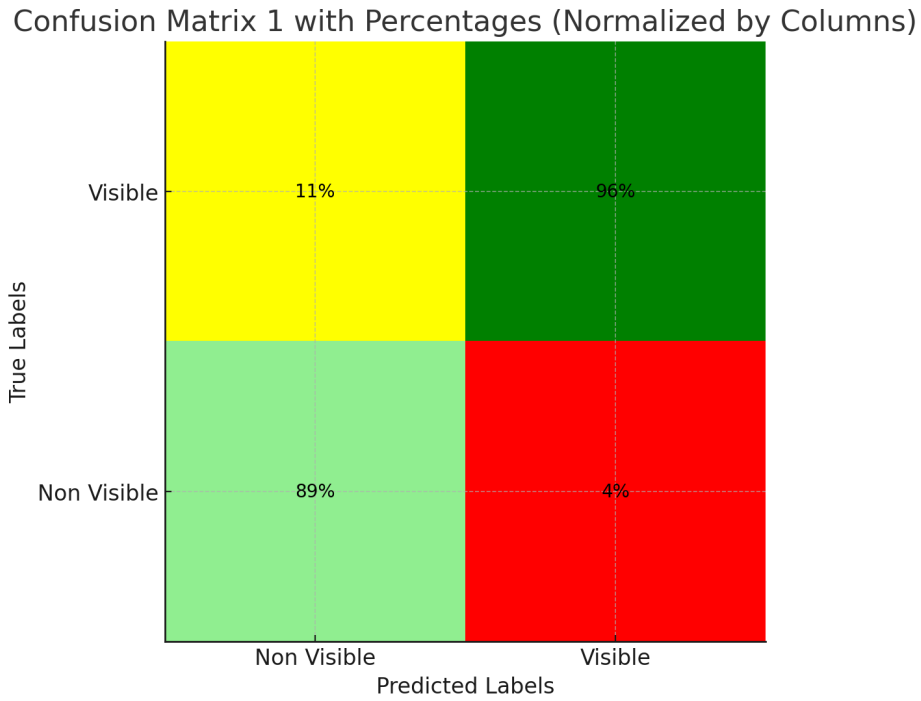


Figura 22: Confusion Matrix Modello 1. Visibilità tra 0 e 10 m.

Classe predetta: Non Visible

- L'89.24% delle volte in cui il modello ha predetto **Non Visible**, la classe reale era effettivamente **Non Visible**.
- Il 10.76% delle volte in cui il modello ha predetto **Non Visible**, la classe reale era invece **Visible** (FN).

Classe predetta: Visible

- Il 96.36% delle volte in cui il modello ha predetto **Visible**, la classe reale era effettivamente **Visible**.
- L'3.64% delle volte in cui il modello ha predetto **Visible**, la classe reale era invece **Non Visible** (FP).

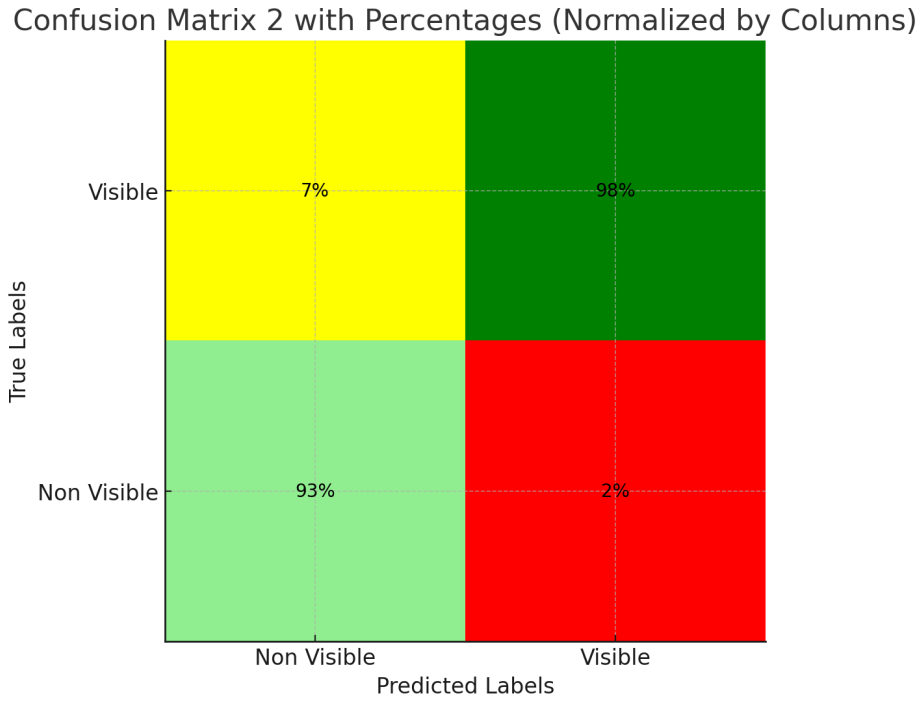


Figura 23: Confusion Matrix Modello 2. Visibilità tra 0 e 10 m.

Classe predetta: Non Visible

- Il 93.02% delle volte in cui il modello ha predetto **Non Visible**, la classe reale era effettivamente **Non Visible**.
- Il 6.98% delle volte in cui il modello ha predetto **Non Visible**, la classe reale era invece **Visible** (FN).

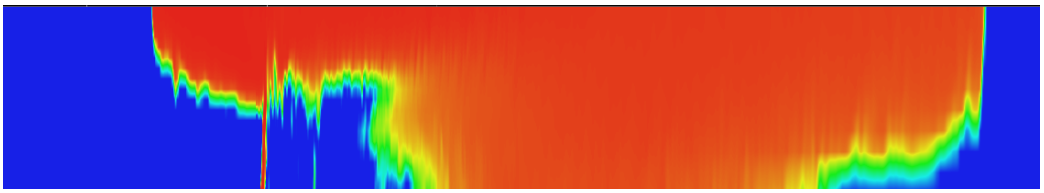
Classe Predetta: Visible

- Il 98.12% delle volte in cui il modello ha predetto **Visible**, la classe reale era effettivamente **Visible**.
- L'1.88% delle volte in cui il modello ha predetto **Visible**, la classe reale era invece **Non Visible** (FP).

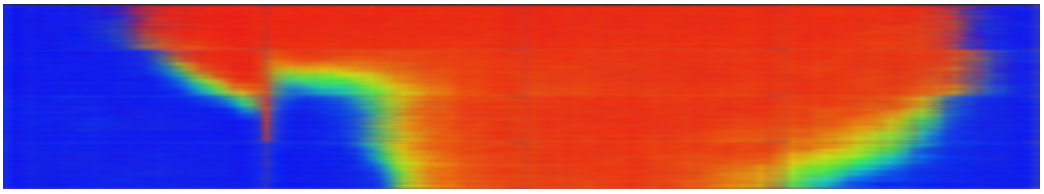
5.5 Plot degli output

A 600 s dall'inizio dell'incendio si ottiene una visualizzazione chiara della distribuzione dei fumi nella galleria. Per questo motivo, tale istante è stato scelto come momento significativo per confrontare la previsione dei modelli.

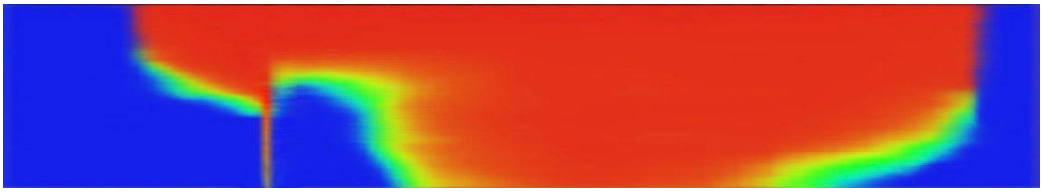
5.5.1 Esempio campione 1



(a) Scenario: HRR 4 MW, pendenza 3%, sezione 100 m^2, posizione 25% della galleria, istante 600. Immagine in output dalle simulazioni con FDS.



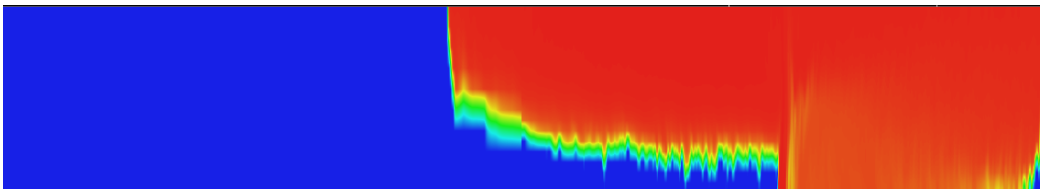
(b) Scenario 2: HRR 4 MW, pendenza 3%, sezione 100 m^2, posizione 25%, istante 600. Immagine in output dal Modello 1.



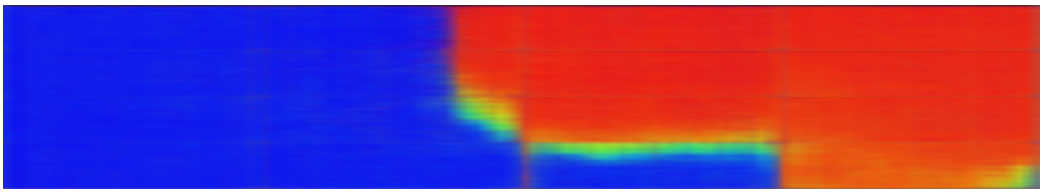
(c) Scenario 3: HRR 4 MW, pendenza 3%, sezione 100 m^2, posizione 25% della galleria, istante 600. Immagine in output dal Modello 2.

Dal confronto con la simulazione reale, è evidente che per bassi valori di HRR, il Modello 1 (raffigurato nel secondo plot) presenti difficoltà nel catturare il sottile pennacchio di fumo generato dallo scoppio dell'incendio, il quale appare molto più sfumato e frantumato, suggerendo evidenti limitazioni nella rappresentazione delle transizioni fini delle aree sottili di fumo.
Inoltre, il Modello 1 tende a sovrastimare le zone di "Non Visible" ai bordi dell'incendio, prevedendo aree di visibilità compromessa che risultano troppo ampie e distanti da quanto ci si aspetterebbe nella realtà, sovrastima che potrebbe portare a una valutazione errata delle aree sicure in situazioni di emergenza.
D'altra parte il Modello 2 (terzo plot), grazie all'integrazione della metrica SSIM nella funzione di loss, dimostra una capacità superiore nel catturare le transizioni strutturali di colore, producendo un pennacchio di fumo più definito, che si traduce in un risultato molto più simile alla simulazione reale. L'integrazione della metrica SSIM ha quindi fatto da ago della bilancia, migliorando notevolmente l'accuratezza in termini di pixel predetti correttamente. Nonostante entrambi i modelli presentino alcune limitazioni nella rappresentazione della natura frastagliata dei bordi tra le aree di media visibilità, il Modello 2 si dimostra chiaramente più affidabile nel catturare le dinamiche complesse delle transizioni.

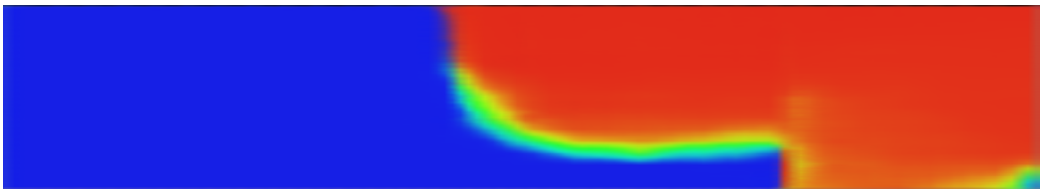
5.5.2 Esempio campione 2



(a) Scenario: HRR 15 MW, pendenza 1.5%, sezione 83 m^2, posizione 75% della galleria, istante 600. Immagine in output dalle simualzioni con FDS.



(b) Scenario 2: HRR 15 MW, pendenza 1.5%, sezione 83 m^2, posizione 75%, istante 600. Immagine in output dal Modello 1.



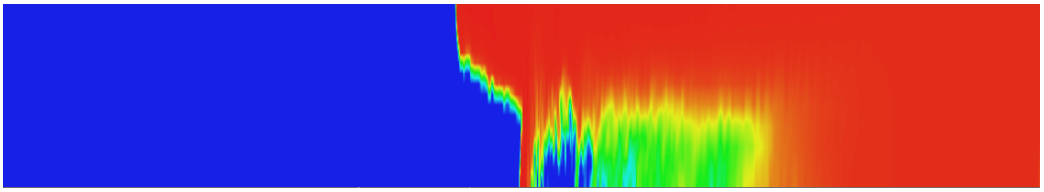
(c) Scenario 3: HRR 15 MW, pendenza 1.5%, sezione 83 m^2, posizione 75% della galleria, istante 600. Immagine in output dal Modello 2.

Il Modello 1 (secondo plot) predice un pennacchio di fumo aggiuntivo e centrale che non è presente nella simulazione reale. Questo comportamento suggerisce una sovrastima dell'estensione del fumo e delle aree compromesse. Di conseguenza, il modello tende a commettere più FN (falsi negativi), classificando come "Non Visible" zone che in realtà sono "Visible". Questo risultato è coerente con quanto mostrato dalla matrice di confusione associata sopracitata, che evidenzia un tasso di FN pari all'11%, sovrastimando così il pericolo.
Il Modello 2 (terzo plot) predice un andamento del fumo molto più simile a quello della simulazione reale. A differenza del Modello 1, non introduce pennacchi di fumo aggiuntivi, mostrando una maggiore precisione nella rappresentazione delle aree di "Non Visible".

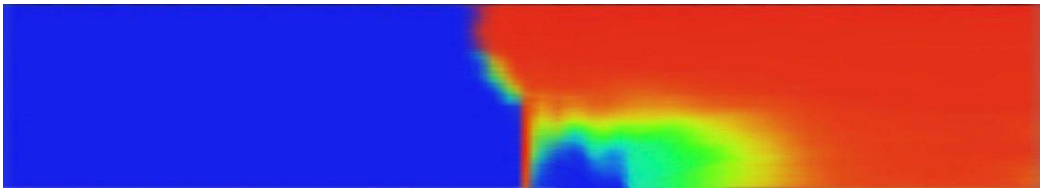
I plot evidenziano inoltre come il Modello 2, grazie alla sua maggiore densità, riesca a catturare meglio le dinamiche complesse, soprattutto in scenari con HRR più elevati. Questo gli conferisce una maggiore affidabilità nel rappresentare incendi più intensi e realistici, migliorando la previsione delle aree di visibilità compromessa.
In sintesi, il Modello 1 sovrastima la diffusione del fumo, portando a predizioni meno precise delle aree di "Non Visible", mentre il Modello 2 offre una rappresentazione più accurata, riducendo in modo significativo il tasso di errori, in particolare i FN.

5.6 Applicazioni pratiche

Considerata la comprovata superiorità del secondo modello nel confronto delle prestazioni, le stime verranno effettuate utilizzando le immagini generate da quest'ultimo. Per investigare il massimo errore commesso, è stato selezionato il campione a 600 s dallo scoppio con il maggior numero di pixel erroneamente classificati in termini di visibilità tra 0 e 10 m. Questo approccio consente di valutare l'accuratezza nei casi più critici, dove l'errore nella classificazione può avere un impatto maggiore sulla previsione e quindi sulla sicurezza delle persone presenti nell'ambiente. Lo scenario con il più alto tasso di falsi negativi e falsi positivi corrisponde a un HRR pari a 8 MW, una pendenza del 3%, una sezione di 83 m^2 e un focolaio localizzato a metà della galleria. Dalle immagini sottostanti (vedi Figura 26), emerge una discrepanza significativa tra la simulazione del modello e la realtà osservata. Nella prima figura, generata dal software FDS, la distribuzione dei fumi appare più complessa, con una colonna di fumo che si espande in modo irregolare, evidenziando una variabilità cromatica che indica una distribuzione non uniforme e flussi turbolenti all'interno della galleria. L'espansione dei fumi copre una zona più ampia sia in altezza che in lunghezza, mostrando come i gas si disperdano rapidamente verso il soffitto e i lati. Nella seconda immagine (output del Modello 2), si notano differenze significative in altezza e forma: la distribuzione appare più compatta e regolare. Il colore rosso, che rappresenta visibilità nulla, domina la parte superiore della galleria, mentre una fascia verde, corrispondente con visibilità media, si estende verso il basso in modo relativamente uniforme. Questa visione più rigida del flusso dei fumi suggerisce che il modello ha sottostimato la turbolenza e l'espansione irregolare osservati nella realtà.



(a) Scenario: HRR 8 MW, pendenza 3%, sezione 83 m^2, posizione 50%, istante 600. Immagine in output dalle simulazioni con FDS.



(b) Scenario 2: HRR 8 MW, pendenza 3%, sezione 83 m^2, posizione 50%, istante 600. Immagine in output dal Modello 2.

Figura 26: Confronto tra simulazioni di distribuzione fumi: FDS e Modello 2.

Queste intuizioni visive sono confermate dagli errori rilevati nei calcoli. Il modello ha stimato una distanza percorribile di 4 m tra il focolaio e il punto in cui i fumi tornano a toccare il suolo, mentre l'osservazione reale ha evidenziato che tale distanza è di 2 m, mostrando un errore significativo. Inoltre, il modello ha previsto che i fumi toccassero un'altezza di 1.20 m, mentre invece la stessa è stata misurata a 1.40 m dalle simulazioni. Sebbene l'errore sull'altezza sia meno pronunciato rispetto a quello della distanza, la constatazione di queste imprecisioni ha evidenziato la necessità di migliorare ulteriormente la precisione del modello per garantire una maggiore affidabilità nelle situazioni critiche.
La scelta di indagare il caso peggiore, ovvero il campione con il maggior numero di pixel erroneamente classificati in termini di visibilità, ha permesso di identificare i limiti del modello nei casi più complessi e di lavorare verso soluzioni più affidabili. I risultati ottenuti indicano che, sebbene il secondo modello mostri generalmente una maggiore accuratezza, esso può ancora incorrere in errori significativi in scenari estremi.

6 Conclusioni

L'obiettivo di questa tesi era quello di esplorare l'utilizzo delle reti neurali convoluzionali trasposte per la previsione della stratificazione dei fumi e della visibilità all'interno di gallerie stradali in scenari d'incendio. Il lavoro ha dimostrato che l'integrazione di un modello di deep learning nella previsione di scenari inediti può contribuire al miglioramento della regolamentazione antincendio basata su PBD, fornendo previsioni affidabili se comparate a quelle ottenute da simulatori CFD, come FDS. Con il modello di AI pre-addestrato, è possibile valutare i nuovi progetti di sicurezza antincendio elaborati dalle istituzioni in termini di visibilità in presenza di fumo, in meno di 1 s.

Il primo contributo significativo di questa ricerca è stata la conferma che una rete neurale composta da strati densi e convoluzionali trasposti, allenata con una funzione di loss standard ℓ_2, è in grado di rilevare con un buon grado di veridicità le zone di pericolo in galleria a seguito dell'evento critico. Sebbene questa configurazione si sia rivelata efficace nel ridurre l'errore numerico, ha presentato limiti nella capacità di riprodurre la linea di propagazione delle fiamme.

Al contrario, un approccio basato sull'integrazione dello SSIM attraverso la definizione di una loss combinata con il MSE ha permesso di ottenere un miglioramento nella qualità strutturale delle immagini generate, come evidenziato dai confronti visivi tra il Modello 1 e il Modello 2, fornendo risultati più affidabili nella previsione delle zone di "Non Visible" e nella rappresentazione del pennacchio di fumo, soprattutto negli scenari con valori più elevati di HRR.

Nonostante i risultati positivi, la ricerca ha evidenziato alcuni limiti. Sebbene il Modello 2 abbia dimostrato prestazioni superiori in scenari estremi, di cui è stato analizzato un esempio, ha comunque registrato errori significativi nel calcolo di misure chiave per la definizione dell'ASET, come la stima della distanza percorribile prima che i fumi raggiungano il suolo. Questo indica che il suddetto, pur essendo più accurato, potrebbe essere ulteriormente suscettibile di perfezionamento al fine di gestire scenari più complessi, caratterizzati da turbolenze o geometrie irregolari. Prestazioni migliori del modello di AI potrebbero essere conseguite simulando un numero maggiore di scenari e parametri chiave per ottenere un database più ampio. Inoltre, il modello dovrebbe essere testato su incendi reali o su simulazioni di larga scala, una volta completata la costruzione dell'edificio specifico per cui è stato progettato il sistema di sicurezza, ovvero l'edificio target. L'utilizzo di questa tipologia di reti neurali, in conclusione, può facilitare l'implementazione di strategie di sicurezza antincendio nelle gallerie, permettendo, con simulazioni

rapide e accurate, di esplorare una più ampia gamma di scenari rispetto alle modalità tradizionali, ottimizzando le procedure di evacuazione nell'ottica di un'ingegneria per la sicurezza.
Per il futuro, si propone di esplorare ulteriori approcci per migliorare l'accuratezza delle previsioni. L'uso di architetture più complesse, come le Convolutional graph neural networks (CGNN) [15] o le Generative adversarial networks (GAN) [5], potrebbe portare ad una rappresentazione ancora più precisa delle dinamiche del fumo. Inoltre, un'ulteriore ottimizzazione della funzione di loss attraverso l'integrazione di nuove metriche strutturali emergenti, potrebbe migliorare la capacità del modello di cogliere le dinamiche più complesse nelle zone di transizione.

Riferimenti bibliografici

[1] Ling chu Su, Xiqiang Wu, Xiaoning Zhang, and Xinyan Huang. Smart performance-based design for building fire safety: Prediction of smoke motion via ai. *Journal Name*, 2024. Department of Building Services Engineering, Hong Kong Polytechnic University, Hong Kong; Ove Arup and Partners Hong Kong Limited, Hong Kong; Research Institute for Sustainable Urban Development, Hong Kong Polytechnic University, Hong Kong.

[2] 2020 Blog For. Hyperband: A novel bandit-based approach to hyperparameter optimization. Online blog post, 2020. Accessed: 2024-08-26.

[3] GeeksforGeeks. Cnn — introduction to padding. Website, 2023. Accessed: 2024-08-04.

[4] GeeksforGeeks. Vanishing and exploding gradients problems in deep learning. Website, 2023. Accessed: 2024-08-04.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] Xiaochen Gu, Jiandu Zhang, Yong Pan, Yuqing Ni, Congming Ma, Wei Zhou, and Yuanyuan Wang. Hazard analysis on tunnel hydrogen jet fire based on cfd simulation of temperature field and concentration field. *Journal Name*, 2023.

[7] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

[8] J.L. Hodges. *Predicting Large Domain Multi-Physics Fire Behavior Using Artificial Neural Networks*. PhD thesis, Virginia Polytechnic Institute and State University, 2018.

[9] J.L. Hodges, B.Y. Lattimer, and K.D. Luxbacher. Compartment fire predictions using transpose convolutional neural networks. *Fire Safety Journal*, 108:102854, 2019.

[10] A. Hore and D. Ziou. Image quality metrics: Psnr vs. ssim. *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[11] L.H. Hu, R. Huo, W. Peng, W.K. Chow, and R.X. Yang. On the maximum smoke temperature under the ceiling in tunnel fires. *Journal Name*, X(X):X–X, 2003.

[12] IBM. What are convolutional neural networks? Website, 2023. Accessed: 2024-08-04.

[13] Andrej Karpathy. Convolutional neural networks (cnn / convnets). http://cs231n.github.io/, 2018. Accessed: 2018-10-15.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Accessed: 2024-08-04.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2017.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates Inc., 2012.

[17] K.B. Lee and H.S. Shin. An application of a deep learning algorithm for automatic detection of unexpected accidents under bad cctv monitoring conditions in tunnels. In *Proceedings - 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications, Deep-ML 2019*, 2019.

[18] R. McDermott, S. Hostikka, J. Floyd, and K. McGrattan. *Fire Dynamics Simulator (FDS) User's Guide*. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2021. Accessed: 2024-08-26.

[19] Mohit Mishra. Stochastic gradient descent: A basic explanation. Online article on Medium, 2023. Accessed: 2024-09-24.

[20] M.Z. Naser. Fire resistance evaluation through artificial intelligence - a case for timber structures. *Fire Safety Journal*, 105:1–18, 2019.

[21] National Fire Protection Association. *NFPA 502: Standard for Road Tunnels, Bridges, and Other Limited Access Highways*. National Fire Protection Association, Quincy, MA, 2020.

[22] Yoshiyuki Oka and Graham T Atkinson. Control of smoke flow in tunnel fires. *Fire Safety Journal*, 25:305–322, 1995.

[23] OpenGenus. He initialization in deep learning, 2024. Accessed: 2024-08-31.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[25] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[26] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1284395.

[27] YouTube. Reti neurali - back propagation — teoria di deep learning — deep learning tutorial italiano. YouTube video, 2023. Accessed: 2024-08-26.

[28] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):1–11, 2017.