Università degli Studi della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

---

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea

Previsione del traffico stradale tramite general-purpose sensing e machine learning

Relatore

Prof. Giancarlo Fortino
Prof. Noel Crespi
Ing. Roberto Minerva
Ing. Claudio Savaglio

Candidato

Stefano Morrone
Matr. 235388

Anno Accademico 2022-2023

A mio fratello Alessandro

"Non accontentarti mai, ciò che pare buono basta poco per renderlo ottimo"
- Alessandro Morrone

Sommario

La previsione del flusso del traffico riveste un ruolo cruciale nell'ambito delle Smart City, contribuendo non solo a ottimizzare la gestione della viabilità da parte delle autorità competenti, ma anche a guidare gli automobilisti nella scelta delle strade migliori per raggiungere le loro destinazioni. Un elemento essenziale per una gestione efficace del traffico è proprio la previsione accurata delle condizioni del traffico stradale.

Questo lavoro di tesi propone un approccio innovativo alla previsione del traffico stradale, sfruttando dati relativi all'inquinamento atmosferico e alle condizioni meteorologiche. L'utilizzo di dati provenienti da sensori *general purpose*, come quelli utilizzati per il rilevamento dell'inquinamento atmosferico e le stazioni meteo, offre un vantaggio significativo: infatti, tale approccio consente di ridurre la dipendenza da sensori specifici a vantaggio di sensori più versatili che possono essere impiegati per diverse finalità. In questo studio sono stati analizzati dati inerenti specifici gas atmosferici, tra i quali monossido di carbonio, monossido di azoto, nitrito, ossido di azoto e ozono, in quanto strettamente correlati all'intensità del traffico stradale. Sono stati quindi considerati parametri meteorologici come pressione atmosferica, temperatura, direzione e velocità del vento, poiché essi giocano un ruolo cruciale nella dispersione dei gas sopra menzionati. I dati relativi al flusso del traffico e all'inquinamento atmosferico sono stati raccolti dal portale open data della città di Madrid, Spagna, mentre i dati meteorologici sono stati ottenuti dal sito visualcrossing. Sono state infine utilizzate e combinate reti neurali ricorrenti e convoluzionali per fondere tali dati ed effettuare accurate previsioni del traffico: è stato così possibile dimostrare l'efficacia e l'efficienza dell'approccio, basato su sensori general purpose opportunamente supportati da strumenti di Machine Learning. Lo studio ha conseguito risultati positivi, con un errore di previsione del traffico stradale minimo del 10% e un valore massimo del 16%, rispetto al valore reale, evidenziando, la robustezza dei modelli proposti per la previsione del traffico. In particolare, il modello ibrido, che fonde reti ricorrenti e convoluzionali, ha mostrato una maggiore precisione con un margine di errore inferiore rispetto agli altri modelli nella previsione del flusso del traffico stradale.

Indice

Capitolo 1

Introduzione

Nel panorama contemporaneo, l'Internet of Things (IoT) rappresenta un paradigma emergente di comunicazione che ridefinisce il modo in cui interagiamo con il mondo che ci circonda. Questo concetto rivoluzionario prevede che oggetti comuni della vita quotidiana siano dotati di microcontrollori, dispositivi di comunicazione digitale e protocolli adeguati per comunicare ed essere pienamente integrati nella rete Internet. L'obiettivo fondamentale dell'IoT infatti è rendere Internet ancora più immersiva e pervasiva, aprendo le porte a un accesso e a un'interazione più agevole con una vasta gamma di dispositivi molto eterogenei tra di loro. Questi dispositivi spaziano dagli elettrodomestici alle telecamere di sorveglianza, dai sensori di monitoraggio agli attuatori, dai display ai veicoli e molto altro. L'ampia quantità e varietà di dati generati da tali dispositivi rappresenta una risorsa senza precedenti e apre la strada a un universo di applicazioni innovative che possono supportare le imprese, ottimizzare le operazioni delle amministrazioni pubbliche e migliorare la vita dei cittadini. Uno degli impieghi più significativi dell'IoT emerge nel contesto urbano, dove l'adozione di soluzioni ICT (Information and Communications Technology) dà vita a un nuovo paradigma: le Smart City. Queste città intelligenti, aumentate dalla connettività ubiqua e dalla vasta rete di dispositivi interconnessi, mirano a creare ambienti urbani più efficienti e sostenibili, potenziando simultaneamente la qualità della vita dei cittadini. In questo scenario, l'IoT svolge un ruolo chiave, consentendo la generazione, la raccolta e l'analisi dei dati per migliorare l'infrastruttura, i servizi pubblici e l'interazione cittadina.

Nel contesto delle Smart Cities, la gestione del traffico veicolare emerge come una questione di fondamentale importanza, poiché la mobilità intelligente costituisce uno dei servizi chiave, incidendo direttamente sulla qualità della vita dei cittadini, dando origine a significativi problemi di natura ambientale, sociale ed economica. La capacità di prevedere con anticipo il flusso del traffico riveste un'importanza cruciale nel tentativo di prevenire le congestioni stradali. Questa previsione offre agli automobilisti l'opportunità di selezionare percorsi meno congestionati per raggiungere le loro destinazioni, o di adattare gli orari di viaggio in modo da mitigare eventuali ritardi dovuti al traffico.

La previsione del traffico è una procedura che mira a stimare o prevedere il flusso di veicoli nel prossimo futuro. Uno dei principali fattori che contribuisce all'inquinamento atmosferico urbano è l'incremento delle emissioni causate dal traffico veicolare. Secondo l'Organizzazione Mondiale della Sanità (OMS), una considerevole percentuale dell'inquinamento atmosferico in ambiente urbano è attribuibile al settore dei trasporti. Per affrontare queste sfide, molte città hanno iniziato a implementare una rete diffusa di sensori in grado di monitorare l'intensità del traffico e la qualità dell'aria. L'inquinamento atmosferico derivante dal traffico è influenzato da una serie di variabili, che spaziano dalla tipologia di alimentazione dei veicoli al livello di congestione stradale, dalla quantità di tempo trascorso negli ingorghi, fino alle condizioni atmosferiche e alle caratteristiche geografiche del territorio urbano, tra molte altre. Diverse città in tutto il mondo, tra cui Madrid, Santander e Barcellona in Spagna, Singapore, Seul e Copenaghen, hanno già implementato una vasta rete di sensori per raccogliere dati preziosi destinati a migliorare le previsioni e affrontare questi problemi in maniera più efficace. Stabilire le correlazioni tra i livelli di inquinamento e l'entità del traffico può avere impatti significativi, come ad esempio una migliore gestione della qualità dell'aria. Inoltre, ciò consentirebbe un monitoraggio più efficace del traffico veicolare, riducendo la necessità di numerosi sensori di traffico e quindi contribuendo a contenere i costi di manutenzione. Questi risparmi potrebbero essere reinvestiti nell'implementazione di una più ampia infrastruttura di gestione ambientale. Inoltre, questo approccio potrebbe favorire una transizione da rilevamenti e monitoraggi specifici a strumenti di rilevamento di uso generale adatti a contesti urbani di dimensioni considerevoli. Infine, permetterebbe l'integrazione e l'utilizzo di altre forme di controllo ambientale, come i dati satellitari.

1.1 Obiettivi e Organizzazione

Questo lavoro di Tesi è il risultato di un periodo di mobilità Erasmus, e della collaborazione tra il laboratorio del Professore Noel Crespi (DICE Lab, Telecom SudParis,Parigi - Francia) e il laboratorio del Professore Giancarlo Fortino (SPEME Lab, Unical, Rende - Italia).

L'obiettivo fondamentale è sviluppare un sistema di previsione del traffico basato esclusivamente su dati ambientali e meteorologici. Questo approccio permette di utilizzare sensori più generici, capaci di monitorare una vasta gamma di situazioni urbane, non limitandosi solamente al traffico stradale. L'impiego di sensori generici riduce notevolmente la necessità di dispositivi specifici, che spesso hanno un utilizzo limitato, aumentando così i costi di manutenzione.

Lo sviluppo di un tale sistema potenzialmente può migliorare la sicurezza stradale, poichè il traffico rappresenta una potenziale fonte di rischi: distanze ridotte tra i veicoli, frenate improvvise, ridotta visibilità aumentano il numero di incidenti che possono coinvolgere anche i pedoni. Inoltre, l'ottimizzazione dei flussi veicolari consente di migliorare la qualità della vita all'interno della Smart City: infatti, il traffico impatta negativamente sulla salubrità dell'aria e sul tempo speso in auto dai cittadini.

Per l'implementazione del sistema appena introdotto sono state analizzate le potenzialità delle reti neurali ricorrenti (RNN) e reti neurali convoluzionali (CNN) unidimensionali nel contesto della previsione del traffico stradale in un ambiente Smart City.
Le RNN rappresentano una classe di reti neurali artificiali specializzate nell'elaborazione di dati sequenziali o serie temporali. Questi algoritmi sono ampiamente utilizzati per affrontare problemi che coinvolgono dati ordinati nel tempo, come la traduzione automatica, l'elaborazione del linguaggio naturale (NLP) e il riconoscimento vocale. A differenza delle reti neurali feed-forward e convoluzionali, le RNN si distinguono per la presenza di una memoria interna, che consente loro di catturare le relazioni tra i dati di input nel corso del tempo. Le reti neurali ricorrenti come le LSTM (Long Short-Term Memory) e le GRU (Gated Recurrent Unit), sono varianti specializzate di RNN sviluppate per mitigare alcune delle sfide tipiche delle RNN tradizionali. Un aspetto distintivo delle RNN è che l'output generato dipende strettamente dai dati di input precedenti, permettendo loro di modellare relazioni complesse in dati sequenziali.
Le CNN sono un tipo di rete neurale artificiale specializzata nell'analisi e nell'elaborazione di dati visivi e immagini. Queste reti sono state progettate per emulare la capacità di percezione visiva umana, consentendo loro di riconoscere modelli e caratteristiche all'interno delle immagini. Le CNN sono particolarmente efficaci nella classificazione e nel rilevamento di oggetti in immagini, ma la loro versatilità si estende ben oltre l'ambito della visione artificiale. Possono essere utilizzate con successo anche per l'analisi di serie temporali, riuscendo a catturare sia le informazioni spaziali che temporali nelle sequenze di dati. Tale capacità di estrarre e comprendere pattern complessi e strutture nascoste rende le CNN strumenti preziosi nella previsione di serie storiche, ad esempio nel campo finanziario o meteorologico, dove la rilevazione di correlazioni e tendenze può fornire previsioni accurate e utili.

Il lavoro di tesi è strutturato come segue:

- Nel **secondo capitolo** è presente lo studio della letteratura sulla previsione del traffico stradale e l'influenza dei fattori inquinanti assieme ai fondamenti teorici delle tecnologie adottate per sviluppare i modelli di Machine Learning.
- Nel **terzo capitolo**, sono illustrate le tecnologie e i software utilizzati nello sviluppo dei modelli.
- Nel **quarto capitolo** vengono descritti tutti i passi necessari per la formazione e la creazione del dataset necessario per l'attività di ricerca, e tutti gli step successivi sino all'implementazione dei vari modelli.
- Il **quinto capitolo** riporta un'analisi accurata dei risultati ottenuti dalla validazione e dell'inferenza dei modelli.
- Infine, nel **sesto capitolo**, verranno esposte le conclusioni e le direzioni future.

Capitolo 2

Stato dell'arte e Background

Il capitolo è diviso in due parti. Nella prima parte sarà condotta un'analisi completa sullo stato attuale delle previsioni del traffico stradale e sull'influenza dei gas inquinanti. Attraverso la consultazione di una serie di ricerche e studi presenti in letteratura, sarà fornita una panoramica delle diverse prospettive e metodologie esistenti in questo campo. Questa indagine ci permetterà di acquisire una comprensione approfondita delle soluzioni proposte per la previsione del traffico stradale e, al contempo, offrirà alcuni spunti per valutare, migliorare le metodologie proposte dai vari ricercatori o svilupparne di nuove.

Nella seconda parte del capitolo, sarà esplorata la teoria che sta alla base dei modelli di machine learning utilizzati per la previsione del traffico stradale. Questa analisi seguirà il seguente flusso: Si inizierà con una panoramica sulle Reti Neurali Ricorrenti (RNN) e sulle Long Short-Term Memory (LSTM), con particolare attenzione alle varianti unidirezionali e bidirezionali. Verrà esaminata anche l'architettura delle Gated Recurrent Unit (GRU) come alternativa nello studio dei dati sequenziali. Inoltre, saranno affrontate le Reti Convoluzionali (CNN) e il concetto di dimensionalità, così da consentire la distinzione tra reti bidimensionali, unidimensionali e tridimensionali.

2.1 Analisi della Letteratura

Pilar Rey del Castillo [1] fornisce un'analisi approfondita del traffico nella città di Madrid. Nel suo lavoro, propone indicatori a breve termine per comprendere l'evoluzione del traffico urbano. Questo studio del 2020 ha sfruttato una vasta gamma di algoritmi di apprendimento automatico, tra cui K-means, alberi decisionali e K-nearest neighbors, insieme a modelli di traffico, dati meteorologici e dati relativi agli eventi di Madrid per prevedere la congestione del traffico in città. Majumdar et al. [2] utilizzano una LSTM per prevedere la congestione distribuita sulla rete stradale. Questo studio si è basato sui dati stradali raccolti nella città di Buxton, nel Regno Unito. Le caratteristiche chiave considerate sono state la velocità, la lunghezza, il flusso e l'avanzamento del traffico. Wang et al. [3] presentano un metodo integrato che combina il Group Data

management Method (GMDH) e il Seasonal Autoregressive Integrated Moving Average (SARIMA) per prevedere il flusso del traffico nel distretto di Guiyang, in Cina. Hou et al. [4] presentano un modello ibrido che unisce un algoritmo di rete neurale wavelet e un algoritmo di media mobile integrata autoregressiva (ARIMA) per la previsione del traffico. Nel loro studio, utilizzano solo due attributi chiave, ovvero tempo e flusso del traffico. Zhu et al.[5] sfruttano i dati GPS per proporre una tecnica di previsione del traffico. Nel loro lavoro introducono un modello basato su una rete neurale artificiale che utilizza un algoritmo di percorso ottimale per determinare il flusso di traffico a breve termine. In una prospettiva simile, Ketabi et al. [6] conducono un'analisi approfondita delle reti neurali ricorrenti e delle tecniche convenzionali per prevedere la congestione del traffico. Tuttavia, nel loro studio, sono stati considerati principalmente i dati provenienti dalle telecamere di sorveglianza stradale, senza considerare l'impatto dell'inquinamento atmosferico. Wei et al. [7] espongono una tecnica di previsione del traffico basata sull'utilizzo di un Autoencoder e una rete neurale ricorrente LSTM. Nel loro dataset, si sono concentrati su tre variabili chiave: l'occupazione, la velocità e il flusso del traffico. Shahid et al. [8] impiegano i livelli di inquinanti atmosferici, tra cui monossido di carbonio (CO), Nitrito (NO_2), anidride solforosa (SO_2) e ozono (O_3), per prevedere il traffico stradale. Questo studio ha impiegato sette diversi modelli di regressione, utilizzando dati aperti raccolti dalla città di Aarhus, in Danimarca. Russo et al. [9] si avvalgono di dati atmosferici, tra cui temperatura e direzione e intensità del vento, insieme a gas inquinanti come NO_2, NO (monossido di azoto) e CO, come input per una rete neurale finalizzata alla previsione delle concentrazioni di particolato sottile (PM_{10}). Lana et al. [10] sfruttano i dati storici meteorologici e di traffico per costruire modelli di regressione finalizzati a prevedere i diversi livelli di inquinanti atmosferici presenti nell'aria, tra cui CO,NO,NO_2,O_3 e PM_{10}. Questa ricerca ha fatto ampio uso dei dati del 2015 provenienti dalla città di Madrid. Ly et al. [11] hanno predetto le concentrazioni di NO_2 e CO utilizzando dati provenienti da dispositivi multisensore e tre variabili meteorologiche chiave: temperatura, umidità relativa e umidità assoluta. Batterman et al. [12] sviluppano un modello di dispersione noto come Research Line Source (R-LINE) insieme a un inventario di emissioni al fine di prevedere le concentrazioni orarie di $PM_{2,5}$ e ossidi di azoto (NO_x) a Detroit. Molto importanti sono ovviamente le due ricerche effettuate da Awan et Al. Nel primo studio [13], hanno sviluppato un approccio di previsione del traffico che sfrutta l'utilizzo di gas inquinanti come CO,NO, NO_2, NO_x e O_3, insieme a variabili meteorologiche, quali temperatura, direzione e velocità del vento, nonché il numero di veicoli. Hanno addestrato una rete neurale ricorrente, nello specifico una LSTM, per effettuare previsioni del traffico. Nel secondo studio [14], hanno adottato un approccio simile, utilizzando dati sull'inquinamento acustico e dati relativi al traffico stradale. Anche in questo caso, hanno impiegato una rete neurale ricorrente LSTM per prevedere il traffico. In entrambi i casi, hanno confrontato i risultati ottenuti utilizzando solo i dati relativi al traffico stradale con quelli ottenuti includendo le altre variabili menzionate in precedenza. È fondamentale sottolineare che i dati meteorologici e quelli relativi agli inquinanti sono utilizzati congiuntamente ai dati del traffico per effettuare le previsioni e migliorarne la qualità, ma tali previsioni non si basano solamente su tali

tipologie di variabili.

2.2 Deep Learning models

Tecniche classiche come la regressione multilineare e il noto modello Auto-Regressive Integrated Moving Average (ARIMA), sono state ampiamente utilizzate per affrontare le sfide legate alla previsione di serie temporali. Tuttavia, questi approcci statistici spesso impongono assunzioni come la stazionarietà e la correlazione lineare tra i dati storici, circoscrivendo la loro adattabilità in contesti in cui tali presupposti non sono pienamente soddisfatti. Le limitazioni intrinseche di questi metodi possono compromettere la loro capacità di identificare e catturare pattern non lineari e complessi presenti nelle serie temporali, rendendo difficile lo sviluppo di modelli di previsione robusti e affidabili. Negli ultimi anni, i metodi e le tecniche di deep learning si sono rivelati estremamente efficaci nel risolvere problematiche complesse di previsione delle serie temporali. Questi approcci forniscono un quadro più flessibile per gestire la natura spesso rumorosa e caotica dei dati di serie temporali, consentendo di ottenere previsioni più precise. In particolare, le **reti ricorrenti** (RNN) e le **reti neurali convoluzionali** (CNN) si sono affermate come alcune delle tecniche di deep learning più efficaci e ampiamente utilizzate. Nel contesto della previsione delle serie temporali, le RNN sono progettate per catturare con efficienza le informazioni relative ai pattern sequenziali, sfruttando la loro architettura specifica. D'altra parte, le CNN sono in grado di filtrare il rumore presente nei dati in ingresso ed estrarre le caratteristiche più rilevanti, contribuendo così alla costruzione di modelli di previsione più precisi. Va comunque sottolineato che, sebbene le CNN siano ben adattate per gestire dati con autocorrelazione spaziale, spesso non sono ottimali per gestire dipendenze temporali complesse e prolungate. Le RNN, d'altra parte, sono progettate per affrontare specificamente le correlazioni temporali, ma la loro capacità di previsione dipende in gran parte dalle caratteristiche presenti nell'insieme di addestramento. Pertanto, la scelta del modello più adatto dipende dalle caratteristiche specifiche dei dati e dalle relazioni temporali coinvolte nel problema di previsione [15][16].

2.2.1 Reti neurali ricorrenti - RNNs

Una **rete neurale ricorrente** (RNN) è un tipo di rete neurale artificiale che utilizza dati sequenziali o dati di serie temporali [17]. Questi algoritmi di deep learning sono comunemente utilizzati per problemi ordinali o temporali, come la traduzione linguistica, l'elaborazione del linguaggio naturale (NLP), il riconoscimento vocale e i sottotitoli delle immagini; sono incorporati in applicazioni popolari come Siri e Google Translate.

La distinzione fondamentale tra le Reti Neurali Ricorrenti (RNN) e le Reti Neurali Feedforward, conosciute anche come Perceptron Multistrato (MLP), risiede nella modalità in cui l'informazione fluisce attraverso la rete. Mentre nelle Reti Feedforward, l'informazione scorre senza cicli, nelle RNN sono presenti cicli che consentono all'informazione di essere trasmessa a se stessa. Questa caratteristica amplia le capacità delle RNN, poiché

tiene in considerazione non solo l'input corrente X_t, ma anche gli input precedenti $X_{0:t-1}$ [18]. Tale differenza viene evidenziata nella Figura 2.1, che presenta una visione ad alto livello delle due reti.

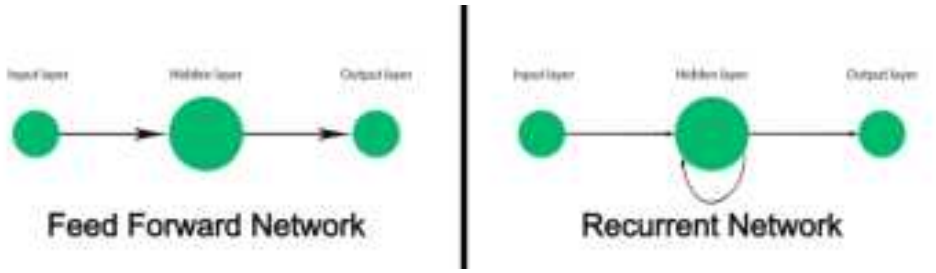


Figura 2.1: *Visione ad alto livello di MLP (Multilayer Perceptron) e RNN (Recurrent Neural Network) che evidenzia le differenze nella modalità di diffusione dell'informazione.*

Nella Figura 2.2 invece, viene mostrata la struttura della rete in seguito allo "srotolamento" del ciclo:

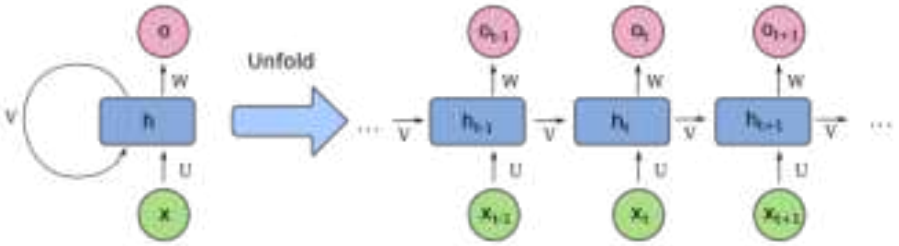


Figura 2.2: *"Srotolamento" ciclo RNN*

In dettaglio, possiamo esaminare il processo attraverso il quale l'informazione viene trasmessa dalla precedente iterazione al livello nascosto utilizzando notazione matematica. Denotiamo l'*hidden state* e l'input al tempo t come segue: $H_t \in R^{n \times h}$ e $X_t \in R^{n \times d}$. In questa notazione, n rappresenta il numero di campioni, h rappresenta il numero di unità nascoste, e d rappresenta il numero di input per ciascun campione. Per modellare questa trasmissione, facciamo uso di una matrice dei pesi $W_{xh} \in R^{d \times h}$, una matrice da stato nascosto a stato nascosto $W_{hh} \in R^{h \times h}$ e un parametro di bias $b_h \in R^{1 \times h}$. Infine, tutte queste informazioni vengono elaborate attraverso una funzione di attivazione Φ, spesso la funzione tangente iperbolica o la funzione sigmoidea, per preparare i gradienti utilizzati nella fase di backpropagation. Riunendo queste notazioni, otteniamo le seguenti equazioni:

$$H_t = \Phi_h(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{2.1}$$

come variabile nascosta, e

$$O_t = \Phi_o(H_t W_{ho} + b_o) \tag{2.2}$$

come variabile di output.

Poiché H_t incorpora in modo ricorsivo H_{t-1}, e questo processo si ripete per ogni passaggio temporale, la RNN tiene traccia di tutti gli stati nascosti che l'hanno preceduta, inclusa H_{t-1}. Se confrontiamo queste notazioni delle RNN con notazioni simili per le reti neurali feedforward, possiamo chiaramente distinguere la differenza descritta in precedenza. Nella prima equazione, vediamo il calcolo per la variabile nascosta, mentre la seconda equazione rappresenta la variabile di output:

$$H = \Phi_h(X W_{xh} + b_h) \tag{2.3}$$

come variabile nascosta, e

$$O = \Phi_o(H W_{ho} + b_o) \tag{2.4}$$

Un'altra cosa che si può notare dalle formule precedentemente riportate, è che vengono utilizzati gli stessi parametri per ciascun input. Ciò riduce la complessità dei parametri, a differenza di altre reti neurali [19]. Uno dei principali vantaggi delle RNN risiede nella loro capacità di connettere informazioni passate a quelle attuali. In alcuni scenari, non è sufficiente avere solo informazioni a breve termine; è necessario avere una memoria delle informazioni precedenti. Un esempio che illustra questa esigenza è la previsione della parola successiva in base alle parole precedenti:

- Se ci troviamo a prevedere l'ultima parola nella frase "le nuvole sono nel cielo", non richiediamo un contesto aggiuntivo: è abbastanza evidente che la parola successiva sarà "cielo". In situazioni in cui la distanza tra le informazioni rilevanti e il punto in cui sono richieste è limitata, le RNN possono apprendere a utilizzare le informazioni passate.

- Tuttavia, se cerchiamo di prevedere l'ultima parola nella frase "Sono cresciuto in Italia ... parlo fluentemente italiano", le informazioni recenti suggeriscono che la parola successiva sarà probabilmente il nome di una lingua. Tuttavia, per restringere ulteriormente il campo e ottenere il contesto specifico dell' "Italia", dobbiamo fare riferimento a informazioni più remote. In alcuni casi, il divario tra le informazioni rilevanti e il punto in cui sono necessarie può diventare notevole. Purtroppo, all'aumentare di tale divario, le RNN potrebbero non essere più in grado di collegare in modo efficace le informazioni.

In teoria, le RNN sono assolutamente in grado di gestire tali "dipendenze a lungo termine". Un essere umano potrebbe scegliere con attenzione i parametri per risolvere problemi di questa tipologia. Purtroppo, in pratica, le RNN non sembrano essere in grado di apprenderli. Tale limitazione è dovuta alla cosiddetta scomparsa o esplosione dei gradienti (rispettivamente **vanishing gradients** e **exploding gradients**):

- i **gradienti "sfumati"** si verificano quando i valori del gradiente sono troppo piccoli, causando l'interruzione dell'apprendimento del modello o impiegando troppo tempo.
- i **gradienti "esplosivi"** si verificano invece, quando i valori del gradiente sono troppo grandi;

Questi fenomeni possono portare a problemi di stabilità della rete durante l'addestramento e portare a oscillazioni e instabilità nei pesi della rete. Per comprendere appieno le sfide legate a questi problemi, è utile esaminare il processo di **backpropagation** (propagazione all'indietro) nelle reti neurali ricorrenti [20]. La propagazione all'indietro comporta la trasmissione dell'errore dalla previsione ai pesi e ai bias della rete. Nelle RNN, questo processo è talvolta denominato **Back Propagation Through Time** (BPTT), poiché si estende attraverso tutti i passaggi temporali, anche se le matrici dei pesi e dei bias rimangono invariate.

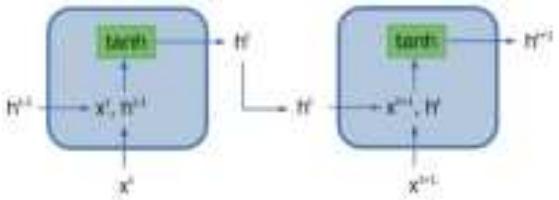


Figura 2.3: *Processo di feedforward nelle RNN*

La Figura 2.3 mostra una sezione di una tipica RNN con due ingressi. Il rettangolo verde rappresenta il calcolo in feedforward degli input e delle loro attivazioni nello stato nascosto, utilizzando la funzione di tangente iperbolica (tanh). Questi calcoli utilizzano lo stesso insieme di parametri (pesi e bias) in tutti i passaggi temporali.

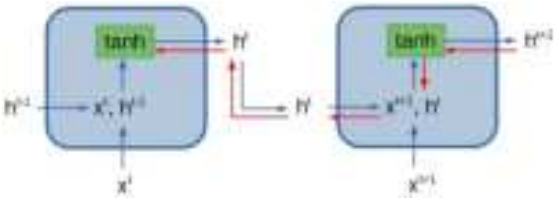


Figura 2.4: *Processo di BPTT nelle RNN*

Nella Figura 2.4 invece, è evidenziato il percorso di **BPTT** in rosso. Per sequenze molto lunghe, i calcoli si accumulano notevolmente. Questo è un aspetto critico poiché può generare un fattore esponenziale che dipende dai valori dei pesi. Ogni volta che retrocediamo di un passo temporale, dobbiamo calcolare il prodotto interno tra il nostro gradiente corrente e la matrice dei pesi. Per dare un'idea della portata del problema, supponiamo che la nostra matrice dei pesi sia scalare e abbia un valore assoluto compreso tra 0,9 e 1,1. Immaginiamo anche di avere una sequenza di 100 passaggi temporali.

Il fattore esponenziale generato dalla moltiplicazione di questi valori per cento volte produrrebbe un problema di gradiente "sfumato" per 0,9 e un problema di gradiente "esplosivo" per 1,1:

$$0,9^{100} = 0,000017(...) \tag{2.5}$$

$$1,1^{100} = 13780,61(...) \tag{2.6}$$

In pratica, i calcoli per il BPTT all'ultimo passaggio temporale assomiglierebbero a quanto segue:

$$\frac{dL}{dW_1} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{dh^n} \frac{dh^n}{dh^{n-1}}(...) \frac{dh^3}{dh^2} \frac{dh^2}{dh^1} \frac{dh^1}{dW^1} \tag{2.7}$$

$$\frac{dL}{dW_1} = \frac{dL}{d\hat{y}} (W_h)^T (W_n)^T (...) (W_3)^T (W_2)^T \frac{dh^1}{dW^1} \tag{2.8}$$

Da notare che questa rappresentazione non è completamente accurata, ma offre una buona comprensione dell'accumulo esponenziale delle matrici dei pesi nel BPTT di una RNN con n input. In questo contesto, W_h rappresenta la matrice dei pesi dell'ultimo strato lineare della RNN. Il processo di aggiornamento dei pesi prevede l'applicazione di una correzione ai pesi esistenti utilizzando la formula:

$$W_{new} = W_{old} - \eta \frac{dL}{dW} \tag{2.9}$$

Successivamente, aggiungiamo una frazione di questi valori alle matrici dei pesi e dei bias. È evidente che questo approccio porta a miglioramenti minimi dei parametri o rischia di generare problemi se cerchiamo di apportare correzioni troppo significative.

2.2.1.1 Long Short Term Memory networks - LSTMs

Le **Long Short Term Memory networks**, comunemente abbreviate come "LSTMs", costituiscono una categoria speciale di reti neurali ricorrenti (RNN) in grado di catturare dipendenze a lungo termine [21]. Questi modelli dimostrano una performance eccellente in una vasta gamma di applicazioni ed hanno ormai una diffusa adozione. L'architettura degli LSTM è stata progettata esplicitamente per risolvere il problema delle dipendenze a lungo termine. Tutte le reti neurali ricorrenti condividono una struttura di base composta da moduli replicati. Nelle RNN tradizionali, questi moduli sono solitamente costituiti da strati semplici con funzioni di attivazione come la tangente iperbolica (tanh). Anche nelle LSTM, la struttura a catena è presente, ma si differenziano per il modulo ripetuto che assume una forma distintiva. Invece di essere composto da un singolo strato di rete neurale, sono costituiti da quattro strati che interagiscono in modo particolare.
Nella Figura 2.5, ciascuna freccia trasporta un vettore completo, collegando l'output di un nodo all'input di altri nodi. I cerchi colorati in rosa rappresentano gli strati della rete neurale. Le linee unite rappresentano la concatenazione di informazioni, mentre la linea che si biforca indica che il contenuto viene duplicato e inviato a posizioni diverse.

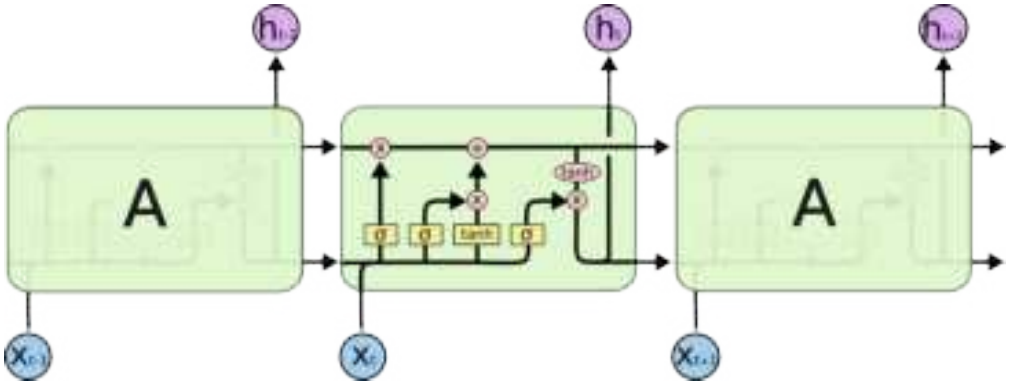


Figura 2.5: *Modulo LSTM*

L'architettura di rete LSTM [22] è composta da tre componenti, ciascuna delle quali svolge un ruolo specifico:

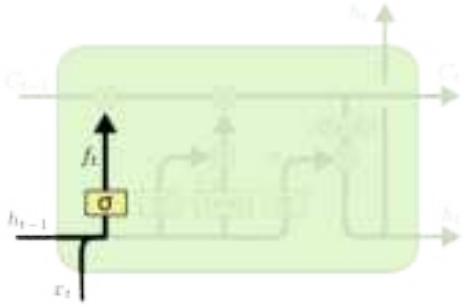
- *Decidere cosa ricordare o dimenticare*: La prima componente si occupa di determinare se le informazioni provenienti dal passaggio temporale precedente siano rilevanti e debbano essere mantenute in memoria o siano invece irrilevanti e possano essere dimenticate.
- *Apprendere nuove informazioni*: La seconda componente cerca di acquisire nuove informazioni dall'input corrente e aggiornare lo stato interno della cella LSTM.
- *Trasmettere le informazioni aggiornate*: Infine, la terza componente si occupa di comunicare le informazioni aggiornate dal passaggio temporale corrente a quello successivo.

Queste tre parti lavorano in sinergia per consentire alle reti LSTM di catturare relazioni a lungo termine e gestire con successo le sequenze di dati. Le unità LSTM sono suddivise in tre componenti conosciute come porte (**gate**). Queste porte regolano il flusso di informazioni all'interno e all'esterno della cella di memoria LSTM. La prima porta è chiamata "**Forget gate**", la seconda "**Input gate**", e l'ultima "**Output gate**". Un'unità LSTM è costituita da queste tre porte e da una cella di memoria, che può essere pensata come un layer di neuroni in una rete feedforward tradizionale, in cui ciascun neurone ha uno stato nascosto e uno stato corrente. Come nelle reti neurali ricorrenti standard, un'unità LSTM ha uno stato nascosto in cui h_{t-1} rappresenta lo stato nascosto al timestamp precedente e h_t è lo stato nascosto al timestamp corrente. In aggiunta a ciò, un'unità LSTM possiede uno *stato della cella* rappresentato da C_{t-1} e C_t, rispettivamente per i timestamp precedente e corrente. In questa architettura, lo **stato nascosto** funge da **memoria a breve termine**, mentre lo **stato della cella** funge da **memoria a lungo termine**.

Cell state e gates L'elemento chiave all'interno di una rete LSTM è lo **stato della cella**, raffigurato come una *linea orizzontale* che attraversa l'intero diagramma. Lo

stato della cella funge da sorta di nastro trasportatore che si estende per tutta la lunghezza della sequenza, con poche interazioni lineari lungo il percorso. Questa struttura agevola il flusso continuo di informazioni senza modifiche. Gli LSTM sfruttano l'abilità di rimuovere o aggiungere informazioni allo stato della cella, un processo attentamente regolato da componenti noti come "porte". Queste porte costituiscono un meccanismo per il passaggio selettivo delle informazioni e sono composte da uno strato di reti neurali sigmoide associato a un'operazione di moltiplicazione puntuale. Lo strato sigmoideo emette valori compresi tra zero e uno, che indicano la quantità di ciascun componente da far passare. Un valore vicino a zero implica "non far passare nulla," mentre un valore prossimo a uno implica "lasciar passare tutto." Un'architettura LSTM dispone di tre di queste porte, che servono a proteggere e regolare lo stato della cella.

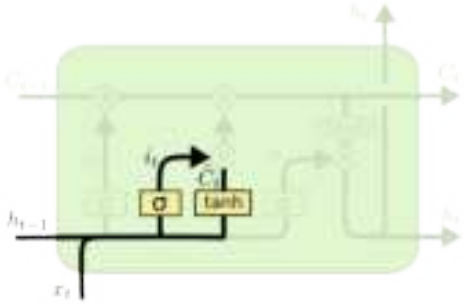
Forget gate layer Nel primo passo di una rete LSTM, si compie una decisione cruciale: determinare quali informazioni conservare e quali scartare dallo stato della cella. Questa decisione è affidata a uno strato sigmoideo noto come il 'livello di porta dell'oblio' (**forget gate layer**). Questo strato riceve in input h_{t-1} e x_t e produce in output un valore compreso tra 0 e 1 per ciascun elemento all'interno dello stato della cella C_{t-1}. Un valore di 1 indica che l'informazione corrispondente verrà mantenuta, mentre un valore di 0 indica che verrà dimenticata.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Figura 2.6: *Forget gate layer*

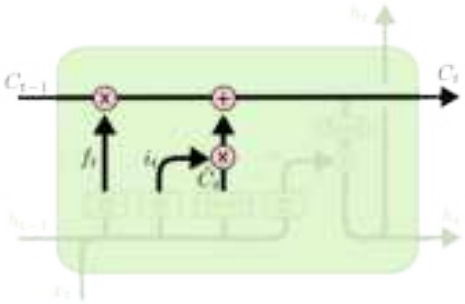
Input gate layer and candidate value Il passo successivo comporta la selezione delle nuove informazioni da memorizzare nello stato della cella. Questa operazione è divisa in due parti. Inizialmente, uno strato sigmoideo denominato "**input gate layer**" determina quali valori saranno soggetti a un aggiornamento. Successivamente, uno strato tanh genera un vettore di nuovi valori candidati, $\widetilde{C}_t$, che potrebbe essere sommato allo stato attuale. Nell'ulteriore passaggio, saranno uniti questi due elementi al fine di ottenere un aggiornamento per lo stato della cella.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 2.7: *Input gate layer e vettore dei nuovi valori candidati*

Ora è il momento di effettuare l'aggiornamento del vecchio stato della cella, C_{t-1}, al nuovo stato della cella C_t. I passaggi precedenti hanno portato alla selezione dei coefficienti appropriati, ed è giunto il momento di applicarli. Per procedere, verrà moltiplicato il vecchio stato per f_t al fine di dimenticare le informazioni precedentemente selezionate. Successivamente, verrà aggiunto il prodotto tra i_t e $\widetilde{C}_t$. Questi sono i nuovi valori candidati, opportunamente pesati in base alle decisioni prese durante il processo di aggiornamento di ciascun valore di stato.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 2.8: *Inserimento del vettore dei nuovi valori candidati nel nuovo stato attuale*

Ouput gate Infine, è necessario definire il processo di generazione dell'output. Questo output sarà basato sullo stato della cella, ma sarà sottoposto a un processo di filtraggio. Inizialmente, verrà applicato uno strato sigmoideo per determinare quali parti dello stato della cella contribuiranno all'output finale. Successivamente, i valori dello stato della cella verranno elaborati attraverso una funzione tangente iperbolica (tanh), che li ridurrà in un intervallo compreso tra -1 e 1. Infine, questi valori saranno moltiplicati per l'output della funzione sigmoidea, selezionando così solo le parti rilevanti per la produzione dell'output.

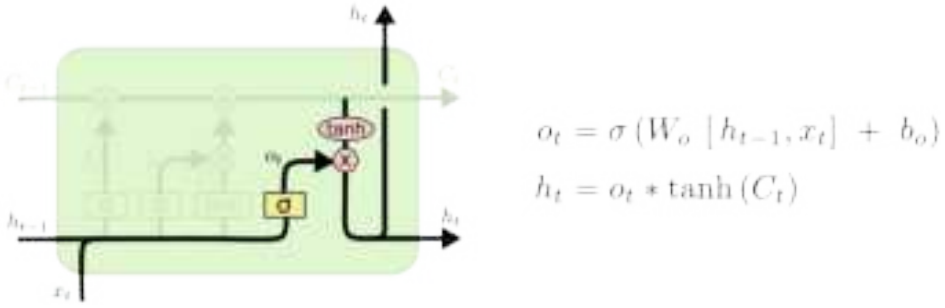


Figura 2.9: *Output Gate*

2.2.1.1.1 Bidirectional Long Short Term Memory networks

Le **Bidirectional Long Short Term Memory networks**, conosciute come LSTM bidirezionali, costituiscono un'architettura di rete neurale ricorrente (RNN) in grado di elaborare dati di input sia in direzione avanzata che retrograda. In una RNN LSTM tradizionale, le informazioni scorrono solo dal passato al futuro, consentendo di effettuare previsioni basate sul contesto precedente. Al contrario, gli LSTM bidirezionali considerano il contesto sia passato che futuro, acquisendo dipendenze in entrambe le direzioni. Questa architettura prevede due strati LSTM, uno che processa la sequenza di input in avanti e l'altro all'indietro, consentendo alla rete di accedere contemporaneamente alle informazioni da passaggi temporali passati e futuri. Gli LSTM bidirezionali risultano particolarmente efficaci in attività che richiedono una comprensione completa delle sequenze di input, come l'elaborazione del linguaggio naturale, comprese l'analisi del sentiment e la traduzione automatica. Integrando informazioni da entrambe le direzioni, gli LSTM bidirezionali potenziano la capacità del modello di acquisire dipendenze a lungo termine e migliorano la precisione delle previsioni nei dati sequenziali complessi.

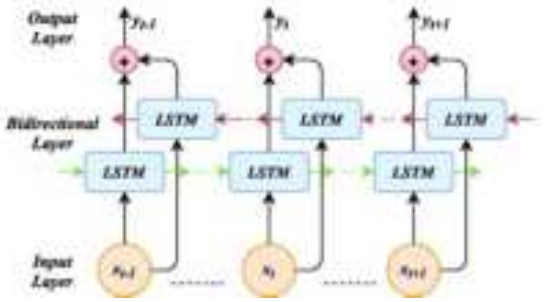


Figura 2.10: *Bidirectional LSTM*

2.2.1.2 Gated Recurrent Unit networks - GRUs

Le reti **Gated Recurrent Unit** (GRU) rappresentano una variante delle LSTM, caratterizzate da una struttura semplificata. Le GRU sono dotate di *due porte*, mentre le LSTM ne possiedono tre. A differenza delle LSTM, le GRU si basano esclusivamente su uno stato nascosto per il trasferimento di informazioni tra le unità ricorrenti, eliminando l'utilizzo di uno stato di cella separato. Inoltre nelle LSTM, la porta di input controlla cosa verrà aggiunto nello stato interno, mentre nelle GRU, tale compito è gestito dall'update gate. Nelle LSTM, il forget gate decide quali informazioni precedenti verranno mantenute, mentre nelle GRU, la porta di reset decide quali informazioni precedenti saranno azzerate.Le LSTM tendono a catturare sequenze più lunghe e complesse grazie alla loro struttura a tre porte, ma sono più complesse da addestrare. Le GRU hanno meno parametri e possono essere più facili da addestrare, ma potrebbero essere meno adatte per modellare sequenze lunghe.[23].

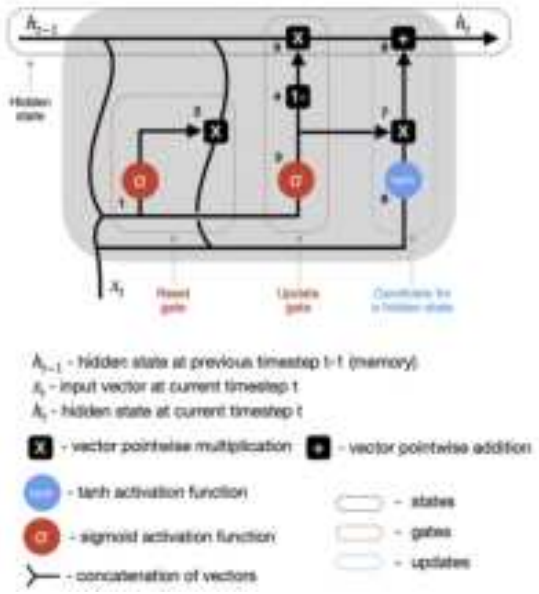


Figura 2.11: *Componenti Gated Recurrent Unit*

Nel dettaglio varranno spiegati tutti i passaggi riportati nel diagramma della Figura 2.11 [24].

- **1-2 Porta di reset**: Questo processo inizia combinando lo stato nascosto precedente (h_{t-1}) con l'input corrente (x_t), moltiplicando ciascun valore per i rispettivi pesi e sommando i bias. Questo risultato viene poi fatto passare attraverso una porta di reset, che utilizza una funzione sigmoidea. La funzione sigmoidea genera valori tra 0 e 1, determinando quali informazioni scartare (0), ricordare completamente (1) o memorizzare parzialmente (valori tra 0 e 1). Il *secondo passaggio* coinvolge la ridefinizione dello stato nascosto precedente, ottenuto moltiplicando l'output del passaggio uno con lo stato nascosto precedente.

- **3-4-5 Porta di aggiornamento**: Il *terzo passaggio* può sembrare simile al primo, ma è importante notare che qui vengono utilizzati pesi e bias diversi, generando un output sigmoideo differente. Questo output, ottenuto attraverso la funzione sigmoidea, viene quindi sottratto da un vettore costituito da tutti 1 (*passaggio quattro*) e moltiplicato per lo stato nascosto precedente (*passaggio cinque*). Questo rappresenta un'operazione di aggiornamento dello stato nascosto con nuove informazioni.

- **6-7-8 Candidato per lo stato nascosto**: Dopo aver reimpostato lo stato nascosto precedente nel passaggio due, gli output vengono combinati con i nuovi input (x_t), moltiplicati per i rispettivi pesi e aggiunti i bias. Successivamente, il risultato passa attraverso una funzione di attivazione tangente iperbolica (*passaggio sei*). Il candidato stato nascosto ottenuto viene quindi moltiplicato per l'output di una porta di aggiornamento (*passaggio sette*) e sommato allo stato nascosto precedentemente modificato (h_{t-1}), creando così il nuovo stato nascosto h_t.

2.2.2 Convolutionale Neural Networks - CNNs

Le **Convolutional Neural Networks**, o CNN, sono un tipo di rete neurale artificiale progettata per affrontare problemi di visione artificiale, in cui l'obiettivo è analizzare e riconoscere modelli all'interno di immagini o dati bidimensionali. Questa architettura di rete è ispirata dalla percezione visiva umana e ha dimostrato di essere estremamente efficace nella risoluzione di compiti di classificazione e rilevamento di oggetti. Sebbene le CNN siano state originariamente progettate per problemi di visione artificiale, il loro utilizzo è stato esteso con successo a compiti di analisi di serie temporali. Le CNN possono apprendere caratteristiche spaziali e temporali nelle sequenze, rendendole efficaci nella previsione di serie storiche, come dati finanziari, dati meteorologici e altro ancora. Questo approccio è particolarmente utile quando le serie temporali contengono pattern complessi e strutture nascoste che le CNN possono rivelare ed estrarre. Una Convolutional Neural Network è caratterizzata da tre livelli principali che lavorano insieme per estrarre e apprendere automaticamente caratteristiche rilevanti dai dati in ingresso[25]:

- **convolutional layer**
- **pooling layer**
- **fully connected layer**

Convolutional Layer Il cuore pulsante di una CNN è rappresentato dal suo **strato convoluzionale**, responsabile della stragrande parte dei calcoli. Questo strato richiede alcuni componenti chiave: i dati di input, un filtro e una mappa delle caratteristiche. Per comprenderne il funzionamento, consideriamo un'immagine a colori come input, la quale è essenzialmente una matrice tridimensionale di pixel, ciascuno con tre dimensioni corrispondenti ai canali RGB. Il **filtro**, noto anche come *kernel*, si sposta all'interno dell'immagine, controllando la presenza di caratteristiche. Questa operazione è chiamata

convoluzione. Il filtro in questione è una matrice bidimensionale di pesi, che rappresenta una parte dell'immagine. Anche se le sue dimensioni possono variare, spesso è una matrice 3x3, determinando così la dimensione del campo recettivo. Il filtro viene applicato a una porzione dell'immagine, calcolando un prodotto scalare tra i pixel dell'input e il filtro. Il risultato di questo prodotto scalare viene quindi inserito in un'array di output. Il filtro si sposta gradualmente, iterando questo processo fino a coprire l'intera immagine. Il risultato finale della serie di prodotti scalari tra l'input e il filtro è conosciuto come **mappa di feature**, **mappa di attivazione** o **feature di convoluzione**. È importante notare che i pesi nel filtro rimangono costanti mentre scorre sull'immagine, un principio noto come *condivisione dei parametri*. Durante il processo di allenamento tramite backpropagation e discesa del gradiente, alcuni parametri, come i valori dei pesi, verranno regolati. Tuttavia, tre iperparametri fondamentali che influenzano la dimensione del volume di output devono essere configurati prima dell'addestramento della rete neurale. Questi includono

1. Il **numero di filtri** (width), che influisce sulla profondità dell'output. Ad esempio, tre filtri distinti produrrebbero tre diverse mappe di caratteristiche, creando quindi una profondità di tre.

2. Il **passo** (stride) è la distanza, o numero di pixel, che il kernel sposta sulla matrice di input. Sebbene valori di falcata pari o superiori a due siano rari, un passo più grande produce un risultato inferiore.

3. Il **riempimento con zero** (padding), che viene solitamente utilizzato quando i filtri non si adattano all'immagine di input. Ciò imposta a zero tutti gli elementi che ricadono al di fuori della matrice di input, producendo un output più grande o di dimensioni uguali

Dopo ogni operazione di convoluzione, una CNN applica una trasformazione Rectified Linear Unit (ReLU) alla mappa delle caratteristiche, introducendo la non linearità nel modello.

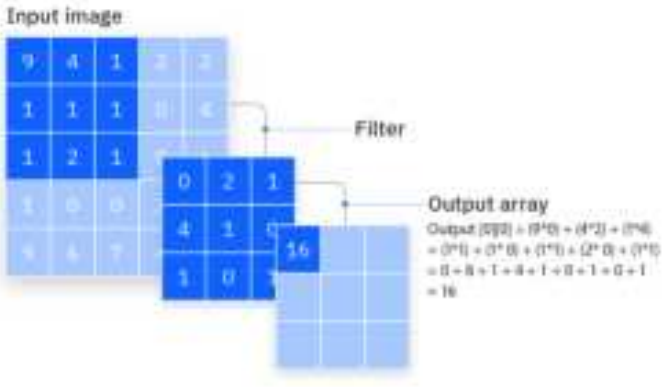


Figura 2.12: *Operazione di convoluzione*

Pooling Layer I **livelli di pooling**, noti anche come "*downsampling*", svolgono una critica funzione di riduzione della dimensionalità, contribuendo a contenere il numero di parametri nell'input della rete. Questo strato condivide alcune similitudini con il livello convoluzionale, ma con un'importante differenza: mentre il livello convoluzionale applica filtri con pesi specifici, il livello di pooling utilizza filtri senza pesi. In altre parole, il filtro di pooling esegue un'operazione di aggregazione dei valori all'interno di una specifica regione dell'input, generando un output basato su questa aggregazione. Esistono principalmente due varianti di pooling che svolgono ruoli distinti:

- **Max Pooling**: quando il filtro si sposta sull'input, seleziona il pixel con il valore massimo da inviare all'array di output.
- **Average Pooling**: quando il filtro si sposta sull'input, calcola il valore medio all'interno del campo ricettivo da inviare all'array di output.

Anche se durante il processo di pooling si verifica una perdita di informazioni, questa operazione comporta diversi vantaggi fondamentali per le CNN. Tra questi, vi è la riduzione della complessità del modello, un incremento dell'efficienza computazionale e la mitigazione del rischio di overfitting.

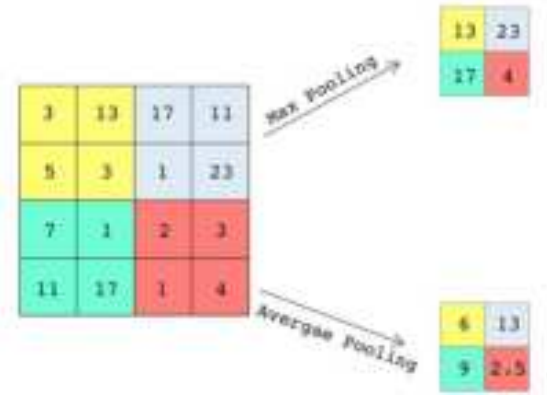


Figura 2.13: *Operazione di pooling*

Fully Connected Layer I neuroni in questo strato sono interconnessi con tutti i neuroni sia nel livello precedente che in quello successivo, seguendo una struttura simile alle reti neurali completamente connesse (FCNN o Fully Connected Neural Network). Di conseguenza, le operazioni in questo strato possono essere calcolate attraverso la moltiplicazione di matrici, seguita dall'aggiunta di bias, come è tipico nelle reti neurali completamente connesse. Il ruolo principale di questo strato è quello di mappare le rappresentazioni tra l'input e l'output, contribuendo così alla capacità della rete di apprendere e generare previsioni significative[26]

Quando ci riferiamo alle Convolutional Neural Networks (CNN), nella maggior parte dei casi si fa riferimento a reti **bidimensionali** utilizzate per classificare immagini. Tut-

tavia, nell'ambito delle reti neurali convoluzionali, esistono anche due altri tipi di reti: le CNN **monodimensionali** e **tridimensionali**. La caratteristica distintiva tra questi tipi di reti è la dimensione attraverso cui il kernel si sposta (monodimensionale = una sola dimensione, bidimensionale = due dimensioni, e così via). In particolare, le reti CNN monodimensionali sono progettate per elaborare dati che presentano una sola dimensione. Un esempio concreto di dati che richiedono il movimento del kernel in una sola dimensione e presentano caratteristiche spaziali sono le serie temporali. Per comprendere meglio questa applicazione, consideriamo i dati raccolti da un accelerometro indossato da una persona sul braccio. Questi dati rappresentano l'accelerazione rilevata lungo tutti e tre gli assi. La CNN monodimensionale può essere impiegata per riconoscere le attività svolte dalla persona, ad esempio se sta ferma, cammina o salta, basandosi sui dati dell'accelerometro. Questi dati sono bidimensionali: una dimensione rappresenta il tempo e l'altra riporta i valori dell'accelerazione lungo i tre assi. Per chiarire ulteriormente il concetto, immaginiamo il kernel che scorre sui dati dell'accelerometro. Ogni riga nel grafico rappresenta l'accelerazione registrata in un dato momento temporale lungo gli assi. Tuttavia, il kernel può spostarsi solo in una direzione, seguendo la dimensione temporale dell'accelerazione. Questa applicazione delle CNN monodimensionali illustra come tali reti possano essere utilizzate per l'analisi di dati di serie temporali con informazioni spaziali, fornendo una visione più chiara delle attività svolte [27].

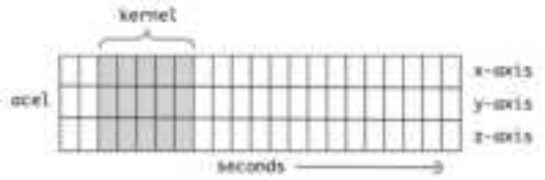


Figura 2.14: *Processo di scorrimento del kernel lungo i dati di un accelerometro*

Capitolo 3

Tecnologie utilizzate

In questo capitolo verranno esposte le tecnologie e i framework impiegati nella progettazione delle reti neurali per la previsione del traffico stradale. Gli oggetti di studio di questo capitolo saranno: **Python**, linguaggio di programmazione ampiamente utilizzato nell'ambito del deep learning e dell'analisi dei dati, il quale si distingue per la sua notevole versatilità nell'ambito della prototipazione rapida e nello sviluppo di modelli di machine learning; **Tensorflow** e **Keras**, due framework di deep learning ampiamente utilizzati; **Google Colab**, piattaforma di cloud computing gratuita, particolarmente indicata per lo sviluppo di progetti di deep learning.

Python è un linguaggio di programmazione interpretato e di alto livello, ampiamente utilizzato in diversi settori, tra cui l'apprendimento automatico, l'intelligenza artificiale, l'analisi dei dati, lo sviluppo web e molte altre discipline. La sua crescente popolarità è il risultato delle sue caratteristiche distintive e della sua ampia comunità di sviluppatori. Questo linguaggio è noto per la sua facilità d'uso, che rende la scrittura del codice più accessibile e comprensibile per gli sviluppatori. Inoltre, Python vanta una potente libreria standard e una semantica dinamica, che consente la creazione di programmi in modo più efficiente e flessibile. Supporta diversi paradigmi di programmazione, tra cui l'orientamento agli oggetti, la programmazione strutturata e molte caratteristiche della programmazione funzionale. Alcune delle caratteristiche più riconoscibili includono l'uso di variabili non tipizzate e l'indentazione come sintassi specifica, al posto delle parentesi più comuni in altri linguaggi. Questo conferisce al codice Python una struttura leggibile e chiara. Python è particolarmente adatto per il machine learning grazie alla sua vasta gamma di librerie specializzate. Queste librerie, tra cui NumPy, Pandas, Scikit-Learn e TensorFlow, forniscono strumenti essenziali per la manipolazione dei dati, l'implementazione di algoritmi di apprendimento automatico e la valutazione delle prestazioni dei modelli. La presenza di queste librerie semplifica notevolmente il processo di sviluppo di modelli di machine learning, consentendo agli sviluppatori di concentrarsi sulla progettazione e sperimentazione di algoritmi senza doversi preoccupare di implementazioni complesse. In sintesi, Python è un linguaggio di programmazione versatile e potente,

ampiamente utilizzato in molteplici settori, ma con una rilevanza particolare nel campo dell'apprendimento automatico grazie alla sua robusta collezione di librerie specializzate.

3.1 Tensorflow e Keras

Keras e **TensorFlow** sono entrambi framework essenziali nell'ambito dell'apprendimento automatico e delle reti neurali. Sebbene abbiano somiglianze, è importante comprenderne le differenze e il ruolo complementare che svolgono: **TensorFlow** è una libreria open source completa e flessibile per l'apprendimento automatico. Offre una vasta gamma di funzionalità, dall'implementazione di modelli di machine learning a basso livello alla gestione delle risorse di calcolo per l'addestramento di reti neurali. TensorFlow può essere utilizzato in modo indipendente per progetti complessi e richiede una certa conoscenza delle API di basso livello. **Keras**, d'altra parte, è un'interfaccia ad alto livello o un wrapper che opera su TensorFlow (o altri backend di libreria open source). Il suo obiettivo principale è semplificare il processo di sviluppo delle reti neurali. Keras rende più intuitivo, visuale e modulare l'utilizzo di TensorFlow. In sintesi, TensorFlow è una libreria end-to-end open source che offre un'ampia gamma di possibilità e può essere utilizzata per progetti di machine learning di qualsiasi complessità. Keras è un'interfaccia di alto livello che semplifica l'utilizzo di TensorFlow, rendendo più accessibile la creazione di reti neurali.

Figura 3.1: *Relazione presente tra Tensorflow e Keras*

3.2 Google Colab

Google Colab, o Colaboratory, è una potente piattaforma basata su cloud offerta da Google che consente agli utenti di eseguire codice direttamente online senza dover preoccuparsi di configurazioni complesse o di installare software sul proprio computer locale. Per utilizzare Google Colab, è necessario disporre di un account Google, che consente di effettuare l'accesso e di accedere alla piattaforma. L'elemento chiave di Google Colab è l'utilizzo dei **Jupyter Notebook**, che sono documenti interattivi in cui è possibile scrivere, eseguire e documentare il proprio codice. Questi notebook consentono di suddividere il codice in celle, ognuna delle quali può contenere sia codice che testo informativo

formattato in Markdown. Questo rende i notebook estremamente utili per i professionisti di data science e machine learning, in quanto consentono di eseguire tutti i passaggi di analisi dati o sviluppo di modelli e di descrivere il processo in linguaggio naturale. Google Colab è particolarmente adatto per lo sviluppo, l'esecuzione e la condivisione di progetti di machine learning, analisi dati e programmazione in generale. Ciò è reso possibile dal fatto che Google Colab fornisce accesso pay-per-use alle GPU (Graphics Processing Units) e alle TPU (Tensor Processing Units), che consentono di addestrare modelli di machine learning in modo significativamente più veloce rispetto a una CPU tradizionale. Inoltre, la piattaforma è preconfigurata con una vasta gamma di librerie e framework popolari per il machine learning, come TensorFlow, PyTorch, scikit-learn e molti altri. In breve, Google Colab è una piattaforma cloud versatile e potente che semplifica lo sviluppo e l'esecuzione di progetti di machine learning e analisi dati, offrendo accesso a risorse hardware avanzate e facilitando la condivisione di progetti con altri utenti.

Capitolo 4

Metodologia

Nel seguente capitolo, verrà esplorato l'intero processo seguito per condurre la nostra attività di ricerca. Si inizierà spiegando la nostra idea iniziale, affrontando le sfide che hanno portato alla sua modifica e illustrando come abbiamo risolto tali problemi durante il nostro percorso. Successivamente, verranno forniti dettagli sulla creazione del dataset utilizzato, comprese le operazioni necessarie per la sua formazione, l'identificazione della tipologia di serie temporali affrontate, la metodologia di alimentazione delle reti neurali e la loro implementazione. La base di partenza per questa ricerca risiede nei lavori precedenti di Faraz Malik Awan, condotti insieme ai professori Roberto Minerva e Noel Crispi. L'idea originale era quella di rilevare stazioni di monitoraggio dell'inquinamento dell'aria, stazioni di monitoraggio dell'inquinamento acustico, stazioni meteorologiche e stazioni di rilevamento del traffico stradale entro una distanza reciproca di 600 m. Sono state individuate tutte le combinazioni di sensori che rientrano entro una distanza reciproca di 600 metri al fine di ottimizzare la correlazione delle informazioni dei sensori con il traffico stradale. Questa selezione è stata guidata dalla volontà di assicurare un'associazione stretta tra le informazioni raccolte dai sensori e il traffico. L'aumento della distanza tra i sensori avrebbe comportato un aumento della probabilità che le informazioni non fossero più interconnesse, consentendo ad altre fonti di inquinamento di influire sui dati rilevati dai sensori, con particolare riferimento ai sensori di inquinamento acustico e atmosferico. Tuttavia, abbiamo adottato un approccio differente concentrandoci sull'uso esclusivo di dati generali, come quelli relativi all'inquinamento atmosferico, ai dati meteorologici e all'inquinamento acustico, per prevedere l'intensità del traffico stradale. Nel paragrafo successivo, verra spiegato da dove provengono i dati e le fasi di elaborazione necessarie per ottenere il nostro dataset finale.

4.1 Dati

I dati utilizzati sono stati acquisiti dal portale aperto di Madrid, il quale fornisce numerosi dataset aperti generati da numerosi sensori presenti nella città. Nell'ambito della nostra ricerca è importante definire 4 tipologie di sensori presenti nella città:

Sensori per l'inquinamento dell'aria Nella città di madrid sono presenti **24** stazioni remote automatiche che raccolgono informazioni di base per la sorveglianza atmosferica e presentano gli analizzatori necessari per la corretta misurazione dei livelli di gas e particelle. Nella Figura 4.1 vengono visualizzate le posizioni dei vari sensori.

Figura 4.1: *Posizioni stazioni di inquinamento atmosferico*

Sensori dell'inquinamento acustico Nella città di madrid sono presenti **31** stazioni di rilevamento dell'inquinamento acustico. Nella Figura 4.2 vengono visualizzate le posizioni dei vari sensori.

Figura 4.2: *Posizioni stazioni di inquinamento acustico*

Sensori per le condizioni meteorologiche Il sistema completo di qualità dell'aria del Consiglio comunale di Madrid comprende la rete meteorologica comunale che presenta

26 stazioni di rilevamento. Nella Figura 4.3 vengono visualizzate le posizioni dei vari sensori.

Figura 4.3: *Posizioni stazioni meteorologiche*

Sensori del traffico Nella città di madrid sono presenti due tipologie di sensori del traffico:

- sensori permanenti
- sensori non permanenti

Figura 4.4: *Posizioni sensori permanenti del traffico*

I primi sono presenti in un numero relativamente ristretto rispetto ai secondi, infatti i primi sono pari **60**, mentre per la seconda tipologia si hanno **7.360** sensori, nello specifico:

- 7.360 rilevatori di veicoli con le seguenti caratteristiche:
  - 71 includono dispositivi di lettura della targa
  - 158 hanno sistemi di visione artificiale ottica controllati dal Mobility Management Center
  - 1.245 sono specifici per le autostrade e l'accesso alla città
  - e il resto dei 5.886, vengono gestiti con sistemi di controllo dei semafori di base.
- Più di 4.000 punti di misurazione:
  - 253 con sistemi per il controllo della velocità, la caratterizzazione dei veicoli e il doppio ciclo di lettura
  - 70 di loro costituiscono le stazioni di misurazione specifiche della città.

Nella Figura 4.4 vengono presentate solamente le posizioni di riferimento dei 60 sensori di traffico permanente, poiché l'altra tipologia non presenta delle posizioni stabili. Per tutte e 4 le stazioni vengono forniti dei file csv contenti numerose colonne, tra cui:

- **codice della stazione**
- **latitudine**
- **longitudine**

Attraverso tali informazioni è stato sviluppato un codice al fine di rilevare tutte le possibili combinazioni delle 4 stazioni a distanza reciproca inferiore a 600m. Di seguito viene riportato il codice implementato per l'individuazione di ciascun gruppo di sensori.

```
def near_stations(dfW,dfT,dfN,dfA):
""" Evaluate if this 4 type of stations are close to each other """

 #creation of the result dataframe
 result_df = pd.DataFrame(columns=[ 'AS_CODE', 'WS_CODE',
                                    'TS_CODE', 'NS_CODE',
                                    'lat_W','lon_W',
                                    'lat_T','lon_T',
                                    'lat_N','lon_N',
                                    'lat_A','lon_A',
                                    'distance_WT','distance_WN',
                                    'distance_WA','distance_TN',
                                    'distance_TA','distance_NA' ])
 # iteration on every point of the first dataframe
 for i_W, rowW in dfW.iterrows():

  for i_T, rowT in dfT.iterrows():

```

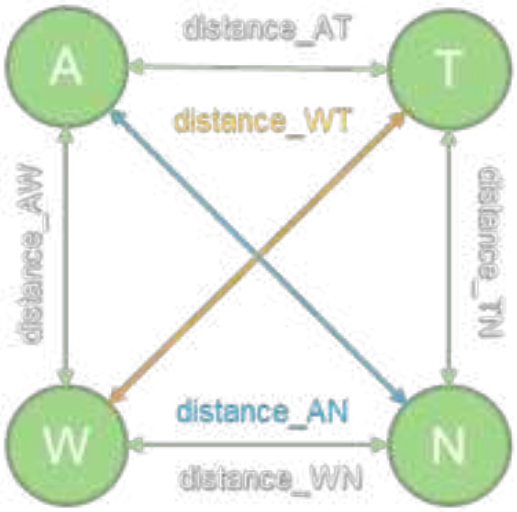
```
19    for i_N, rowN in dfN.iterrows():
20
21     for i_A, rowA in dfA.iterrows():
22
23      dist_WT = distance((rowW['LATITUDE'], rowW['LONGITUDE']),
        ↪  (rowT['LATITUDE'], rowT['LONGITUDE'])).m
24
25      dist_WN = distance((rowW['LATITUDE'], rowW['LONGITUDE']),
        ↪  (rowN['LATITUDE'], rowN['LONGITUDE'])).m
26
27      dist_WA = distance((rowW['LATITUDE'], rowW['LONGITUDE']),
        ↪  (rowA['LATITUDE'], rowA['LONGITUDE'])).m
28
29      dist_TN = distance((rowT['LATITUDE'], rowT['LONGITUDE']),
        ↪  (rowN['LATITUDE'], rowN['LONGITUDE'])).m
30
31      dist_TA = distance((rowT['LATITUDE'], rowT['LONGITUDE']),
        ↪  (rowA['LATITUDE'], rowA['LONGITUDE'])).m
32
33      dist_NA = distance((rowN['LATITUDE'], rowN['LONGITUDE']),
        ↪  (rowA['LATITUDE'], rowA['LONGITUDE'])).m
34
35      if(dist_WT < 600 and dist_WN < 600 and dist_WA < 600 and dist_TN <
        ↪  600 and dist_TA < 600 and dist_NA < 600):
36
37       df_new_row = pd.DataFrame({
38                                  'AS_CODE':[rowA['CODE']],
                                    ↪  'WS_CODE':[rowW['CODE']],
39                                  'TS_CODE':[rowT["Number"]],
                                    ↪  'NS_CODE':[rowN['VIA_CODE']],
40                                  'lat_W': [rowW['LATITUDE']],'lon_W':
                                    ↪  [rowW['LONGITUDE']],
41                                  'lat_T': [rowT['LATITUDE']] ,'lon_T':
                                    ↪  [rowT['LONGITUDE']],
42                                  'lat_N': [rowN['LATITUDE']] ,'lon_N':
                                    ↪  [rowN['LONGITUDE']],
43                                  'lat_A': [rowA['LATITUDE']] ,'lon_A':
                                    ↪  [rowA['LONGITUDE']],
44                                  'distance_WT': [dist_WT], 'distance_WN':
                                    ↪  [dist_WN],
45                                  'distance_WA': [dist_WA], 'distance_TN':
                                    ↪  [dist_TN],
46                                  'distance_TA': [dist_TA], 'distance_NA':
                                    ↪  [dist_NA]})
47
48
49       result_df = pd.concat([result_df,df_new_row])
50
```

```
51    return result_df.reset_index(drop=True)
```

Nella funzione vengono generate tutte le combinazioni, e vengono inserite all'interno di un dataframe, valore di ritorno della funzione, solo quelle combinazioni per le quali la distanza reciproca è inferiore a 600m. Sono inseriti i codici di tutte le stazioni e tutte le distanze presenti tra coppie di stazioni, la Figura 4.5a aiuta a comprendere meglio il significato di ogni colonna presente nel dataframe risultante:

- **A** indica le stazioni di inquinamento dell'aria (*Air pollution*)
- **T** indica le staziondi di Traffico (*Traffic*)
- **W** indica le stazioni metereologiche (*Weather*)
- **N** indica le stazioni di inquinamento acustico (*Noise Pollution*)

distance_WA indica la distanza tra la stazione meteorologica e la stazione di inquinamento dell'aria e così di seguito. È importante segnalare che per la valutazione della distanza tra le stazioni attraverso l'ausilio di latitudine e longitudine è stata usata la libreria python Geopy, che semplifica il calcolo delle distanze geografiche tra due punti e consente di decidere l'unità di misura dei valori di distanza (metri, kilometri ecc.).



(a) Grafico per la comprensione delle colonne del dataframe

(b) Libreria utilizzata per calcolare la distanza tra le stazioni

Figura 4.5

Al termine dell'applicazione del codice il risultato ottenuto è stato quello riportato nella Tabella 4.1 e Tabella 4.2.
La funzione ha restituito 14 possibili combinazioni di stazioni, in particolare: 5 stazioni di inquinamento dell'aria, 6 stazioni meteorologiche, 11 stazioni di traffico e 5 stazioni di inquinamento acustico.

Tabella 4.1: *Dataframe restituito dalla funzione near_stations*

AS_CODE	WS_CODE	TS_CODE	NS_CODE	distance_WT	distance_WN	distance_WA	distance_TN	distance_TA	distance_NA
28079004	28079004	2	273600	245.711856	0.0	0.0	245.711856	245.711856	0.0
28079004	28079004	26	273600	426.806775	0.0	0.0	426.806775	426.806775	0.0
28079008	28079008	18	18900	224.187192	0.0	0.0	224.187192	224.187192	0.0
28079008	28079008	45	18900	340.159099	0.0	0.0	340.159099	340.159099	0.0
28079035	28079035	8	145800	133.723923	0.0	0.0	133.723923	133.723923	0.0
28079035	28079035	25	145800	229.834537	0.0	0.0	229.834537	229.834537	0.0
28079035	28079035	26	145800	511.407832	0.0	0.0	511.407832	511.407832	0.0
28079035	28079035	34	145800	517.046585	0.0	0.0	517.046585	517.046585	0.0
28079035	28079035	35	145800	366.960000	0.0	0.0	366.960000	366.960000	0.0
28079038	28079038	15	554550	385.068832	0.0	0.0	385.068832	385.068832	0.0
28079038	28079038	19	554550	337.907196	0.0	0.0	337.907196	337.907196	0.0
28079056	28079056	10	293200	505.048867	0.0	0.0	505.048867	505.048867	0.0
28079008	28079111	18	18900	166.529713	205.216783	205.216783	224.187192	224.187192	0.0
28079008	28079111	45	18900	363.823602	205.216783	205.216783	340.159099	340.159099	0.0

Tabella 4.2: *Posizioni delle stazioni con i relativi codici*

CODICE	LATITUDINE	LONGITUDINE	TIPO
28079004 (273600)	40.423882	-3.712257	Meteorologica, Inquinamento dell'aria (Inquinamento acustico)
28079008 (18900)	40.421553	-3.682316	Meteorologica, Inquinamento dell'aria (Inquinamento acustico)
28079035 (145800)	40.419209	-3.703166	Meteorologica, Inquinamento dell'aria (Inquinamento acustico)
28079038 (5545550)	40.445544	-3.707130	Meteorologica, Inquinamento dell'aria (Inquinamento acustico)
28079056 (293200)	40.385034	-3.718768	Meteorologica, Inquinamento dell'aria (Inquinamento acustico)
28079111	40.422649	-3.680369	meteorologiche
2	40.426068	-3.712709	Traffico
26	40.422228	-3.707717	Traffico
18	40.421237	-3.679707	Traffico
45	40.424310	-3.684064	Traffico
8	40.420409	-3.703295	Traffico
25	40.420559	-3.701113	Traffico
34	40.416405	-3.708030	Traffico
35	40.417716	-3.707023	Traffico
15	40.446924	-3.711294	Traffico
19	40.443473	-3.704212	Traffico
10	40.385879	-3.724612	Traffico

Una cosa che si può evincere dalle tabelle precedentemente presentate è che molte delle stazioni di inquinamento dell'aria e meteorologiche coincidono, infatti la distanza tra le due stazioni è nulla e i codici identificativi combaciano, mentre per quanto riguarda le stazioni di inquinamento dell'aria e dell'inquinamento acustico hanno distanza nulla ma codice identificativo differente (codice presente tra parentesi nella Tabella 4.2). Dopo esserci accertati che vi fossero stazioni vicine tra di loro, il passo successivo è stato quello di verificare che i dataset presenti avessero una cadenza tale da poter effettuare uno studio accurato. Purtroppo mentre i dati meteorologici, di inquinamento dell'aria e del traffico presentavano dati giornalieri orari (nel caso del traffico dati con cadenza di ogni 15 minuti per le stazioni non permanenti), i dati sull'inquinamento acustico erano presenti in forma cumulativa, ovvero un unico dato riassuntivo per la singola giornata. Come effettuato da Awan nel suo studio, abbiamo cercato di contattare il comune di Madrid per

cercare di ottenere i dati orari a grana più fine, con cadenza oraria, purtroppo però i dati disponibili erano unicamente quelli riportati sul sito. A causa di questo inconveniente quindi abbiamo dovuto riadattare il lavoro di ricerca all'utilizzo di sole **3** tipologie di stazione, ovvero: *stazione meteorologica*, *stazione di inquinamento dell'aria* e *stazione di rilevamento del traffico*. Di conseguenza la funzione precedentemente riportata è stata modificata al fine di rilevare combinazioni delle sole tre stazioni. Il risultato ottenuto presenta le medesime soluzioni ottenute dall'esecuzione della precedente funzione escludendo le stazioni di inquinamento acustico. Nella Figura 4.6 viene riportata la mappa

Figura 4.6: *Posizioni delle stazioni riportate nella Tabella 4.2*

su cui sono state posizionate tutte le stazioni ad una distanza reciproca inferiore a 600m. Nella mappa sono state evidenziate con il simbolo dell'*auto* le stazioni di traffico, con il simbolo del *fuoco* le stazioni di rilevamento dei gas inquinanti e con la *nuvola* le stazioni meteorologiche. Come evidenziato prima, stazioni meteorologiche e di inquinamento del gas in realtà sono un'unica stazione per cui, nella mappa, saranno rappresentate da una sola stazione meteorologica, poiché le altre coincidono per latitudine e longitudine. Una volta individuate le stazioni che rispettavano i criteri da noi ricercati, il passo successivo è stato quello di verificare la qualità dei datasets presenti sul portale aperto di Madrid. Di seguito verrà riportato un sottoparagrafo per l'analisi di ciascun dataset esplorato. Per ciascuna stazione sono stati analizzati i dati relativi all'intero anno 2021.

4.1.1 Dati inquinamento atmosferico

Il primo dataset esplorato è quello relativo all'inquinamento dell'aria. Il portale di Madrid fornisce un file per ogni mese, a partire dal 2001, in cui vengono riportati i livelli di inquinamento dei vari gas per ogni ora del giorno. I dati forniti presentano 55 colonne (i nomi delle colonne riportate sono in spagnolo), tra le quali:

- PROVINCIA
- MUNICIPIO
- MAGNITUDE
- PUNTO MUESTREO
- ANO
- MES
- DIA
- H01
- VO1

Di seguito, esplicitiamo il significato di tre colonne in particolare:

- La colonna **PUNTO MUESTREO** contiene al suo interno una stringa contenente 3 informazioni:
  - il codice della stazione
  - il gas rilevato
  - la tecnica di misurazione utilizzata

  tali informazioni vengono divise dall'interno della stringa attraverso il simbolo di underscore.
- la colonna **H**X e **V**X, la X che segue la H indica l'orario al quale è stata effettuata la misurazione, quindi sono presenti 24 colonne per tale tipologia, il valore all'interno della colonna rappresenta il valore della misurazione del gas rilevato specificato nella colonna "PUNTO MUESTREO", mentre la colonna "VX" indica se il valore riportato nella colonna "HX" sia valido o meno, quindi anche per tale tipologia abbiamo 24 colonne, i valori all'interno sono **"V"** per valid e **"N"** per not valid.

Dall'analisi delle colonne presenti nel dataset si intuisce che non tutti i sensori di inquinamento dell'aria rilevano tutti i medesimi gas, quindi il primo passo è stato comprendere i gas rilevati dalle stazioni. In totale, tutti i gas rilevati sono 17 ma, nel dataset fornito con la locazione dei vari sensori, vengono riportate solo 7 tipologie di gas rilevati, quindi, al fine di comprendere quali gas rilevasse ogni stazione, è stato necessario prendere il primo mese di rilevazioni e verificare tutti i valori presenti nella colonna "PUNTO MUESTREO" per ciascun sensore. Nella Tabella 4.3 vengono riportati tutti i gas rilevati e i relativi codici utilizzati all'interno del dataset per indicare tali gas.

Tabella 4.3: *Gas rilevati e relativi codici*

GAS	CODICE	FORMULA O ABBREVIAZIONE
Diossido di zolfo	01	$S0_2$
Monossido di carboni	06	CO
Monossido di azoto	07	NO
Diossido di azoto	08	NO_2
Particelle ¡ 2.5 µm	09	$PM_{2.5}$
Particelle ¡ 10 µm	10	PM_{10}
Ossido di azoto	12	NO_x
Ozono	14	O_3
Toluene	20	TOL
Benzene	30	BEN
Etilbenzene	35	EBE
Metaxilene	37	MXY
Paraxilene	38	PXY
Ortossilene	39	OXY
Idrocarburi totali (esano)	42	TCH
Metano	43	CH_4
Idrocarburi non metanici (esano)	44	NMHC

Nel nostro caso siamo interessati solo ai gas correlati al traffico stradale quali: **CO**,**NO**,**NO$_2$**,**NO$_x$** e **O3**. Andando a verificare quali delle 5 stazioni precedemente trovate li rilevasse, si scopre che solo 3 delle cinque stazioni rilevano i gas di nostro interesse, ovvero:

- **28079008**
- **28079035**
- **28079056**

Successivamente, abbiamo esaminato attentamente le stazioni per assicurarci che ciascuna di esse avesse registrazioni complete per ogni gas in questione. Dato il periodo limitato considerato per il nostro studio, che è l'intero anno 2021, la gestione di dati mancanti per periodi prolungati sarebbe stata impraticabile. L'utilizzo di metodi di *riempimento* dei dati mancanti per lunghi intervalli di tempo avrebbe potuto influenzare negativamente l'integrità dello studio, poiché avrebbe comportato la generazione di dati sintetici. Inoltre, l'opzione di **escludere** ulteriori finestre temporali sarebbe stata altrettanto problematica, poiché avrebbe significato ridurre notevolmente la quantità di dati a nostra disposizione per l'analisi. Purtroppo, in seguito a tale analisi, si è scoperto che la stazione con codice "28079056" presentava 7 mesi di rilevazioni mancanti (dal mese di giugno fino alla fine dell'anno). Per fortuna le altre stazioni, tenendo in considerazione i dati mancanti e le rilevazioni non considerate valide (valore del campo "VX" pari a "N"), non presentavano grosse problematiche. Quindi le stazioni rimaste a nostra disposizione sono solo la stazione: 28079008 e 28079035. Il passo conclusivo di questa analisi è stato quello di modificare il formato di partenza dei dati così da renderlo più comprensibile, riassuntivo e consono ad un analisi delle serie temporali. Il formato finale presenta le seguenti colonne: datetime, STATION_CODE,CO,NO,NO2,NOx,O3.

4.1.2 Dati metereologici

Il formato dei dati fornito dal portale di Madrid per i dati meteorologici è il medesimo fornito per i dati relativi all'inquinamento atmosferico, però il campo "PUNTO MUESTREO" presenta il tipo di parametro atmosferico registrato. Per quanto riguarda i dati atmosferici rilevati, le stazioni registrano i parametri riportati nella Tabella 4.4.

Tabella 4.4: *Parametri atmosferici rilevati e relativi codici*

CODICE	PARAMETRO
81	VELOCITÀ VENTO
82	DIREZIONE VENTO
83	TEMPERATURA
86	UMIDITÀ RELATIVA
87	PRESSIONE BARIOMETRICA
88	RADIAZIONE SOLARE
89	PRECIPITAZIONE

Come per il caso precedente, è stata effettuata una analisi sulla tipologia del parametro atmosferico rilevato da ciascuna stazione. Alla fine delle analisi il risultato ottenuto è stato quello riportato nella Tabella 4.5.

Tabella 4.5: *Parametri atmosferici rilevati da ciascuna stazione*

STAZIONE	PARAMETRO
28079008	temperatura, umidità relativa
28079035	temperatura e umidità relativa
28079056	velocità e direzione del vento, temperatura e umidità relativa
28079111	temperatura e umidità relativa

Dalla Tabella 4.5, emerge che solo una stazione è dotata di informazioni cruciali, come la velocità e la direzione del vento, fattori che possono influire sulla dispersione dei gas inquinanti. Tuttavia, è importante sottolineare che questa stazione è la medesima stazione di monitoraggio dell'aria precedentemente esclusa a causa della mancanza di tali dati, ed era accoppiata con la stazione di rilevamento del traffico stradale identificata dal codice 10, la quale rappresentava l'unica combinazione presente nella tabella precedente. Di conseguenza, la stazione in questione non poteva essere utilizzata ai fini della nostra ricerca. Per ovviare a questa mancanza, abbiamo individuato il portale **Visual Crossing** per reperire le informazioni mancanti di interesse.

4.1.2.1 Visual Crossing Data

Visual Crossing è una piattaforma web che consente di acquisire dati meteorologici storici da diversi paesi. Ricercando lo stato e la città di interesse, il sito consente di visualizzare le stazioni presenti in quella città e le loro locazioni, inoltre è possibile selezionare il range storico dei dati di interesse, selezionando il livello di granularità dei dati (quotidiani e ogni ora) e la tipologia di unità di misura per ciascuno dei parametri meteorologici rilevati.
Per quanto riguarda Madrid sono presenti 12 stazioni, disseminate per tutta la città. Tra le varie informazioni presenti nel dataset vi sono: *velocità del vento*, *direzione del vento* e *stations*, che rappresenta tutte le stazioni dove i vari dati sono stati rilevati. Da una analisi effettuata risulta che due stazioni (**08221099999** e **08223099999**) sono presenti nel 98% (con precisione 98,58% la prima e 98,06% la seconda). Tali stazioni coprono una ampia zona del centro di Madrid, di conseguenza si è deciso di utilizzare questi dati per il nostro lavoro di ricerca. Inoltre, l'insieme di dati in esame non mostra alcuna assenza di informazioni per l'anno considerato. Attraverso tale dataset siamo riusciti ad

Figura 4.7: *Posizione delle stazioni meteorologiche, in verde le stazioni 08221099999 e 08223099999*

integrare informazioni fondamentali quali la *velocità del vento* e la *direzione*, insieme alla *temperatura*, l'*umidità* e la *pressione atmosferica*. In questo caso non è stato necessario modificare il formato dei dati poiché la struttura fornita dal sito consentiva di lavorare con le serie temporali. Una volta risolto il problema dei dati meteorologici si è passato all'esplorazione dei dati inerenti al flusso del traffico stradale.

4.1.3 Dati sul traffico

Durante l'esame dei dati del traffico nel dataset, è emerso che molte stazioni di rilevamento del traffico fisse presentavano due direzioni di rilevamento, indicando la presenza di strade a doppia corsia. Questa scoperta avrebbe potuto comportare ulteriori complessità nell'analisi che intendevamo condurre. Abbiamo quindi valutato che, per lo scopo specifico del nostro studio, una strada a singola corsia fosse una scelta più adatta. Questo perché avremmo potuto concentrarci con maggiore precisione sul flusso di traffico in

una sola direzione, riducendo al minimo l'impatto di variabili impreviste che avrebbero potuto influire sulle dinamiche delle previsioni. Con l'obiettivo di semplificare la procedura, abbiamo inizialmente esaminato se le stazioni mobili di rilevamento del traffico registrassero dati relativi al traffico in una sola direzione. Una volta confermato che queste stazioni operavano in una sola direzione, abbiamo individuato una stazione vicina a una delle due stazioni di monitoraggio dell'inquinamento dell'aria. Dovendo affrontare una serie di complessità nella procedura di individuazione delle stazioni, abbiamo fatto una scelta strategica, optando per la selezione di una singola stazione che soddisfacesse diversi requisiti. Questa stazione è stata scelta non solo per la sua stabilità in termini di posizionamento nel tempo ma anche per la sua prossimità alle stazioni di rilevamento dell'inquinamento dell'aria. Ciò ci ha permesso di mantenere una forte correlazione tra i dati sull'inquinamento dell'aria e le emissioni dei veicoli che percorrevano la strada. L'esito della ricerca è stato favorevole, conducendoci all'individuazione della stazione di traffico "4303". Questo sensore, posizionato a soli 39 metri dalla stazione di rilevamento dell'aria "28079035", ha dimostrato di essere una scelta ideale per il nostro studio. Inoltre, si trova su una strada a senso unico che sembra non essere afflitta da numerose fonti esterne, come ad esempio industrie, che potrebbero influenzare i dati relativi ai gas inquinanti che abbiamo preso in esame. Questa selezione si è rivelata estremamente vantaggiosa per gli scopi della nostra ricerca.

(a) *Posizione stazioni finali selezionate*

(b) *Zoom stazioni di traffico e di inquinamento*

Figura 4.8: *Posizioni delle stazioni finali selezionate*

L'introduzione di questa nuova tipologia di stazione ha portato a un nuovo formato di dati, diverso da quelli precedenti in termini di frequenza. Mentre i dati precedenti, come quelli relativi all'inquinamento dell'aria e alle condizioni meteorologiche, venivano registrati ogni ora, questi nuovi dati vengono forniti ogni 15 minuti. Di conseguenza,

per uniformare il formato dei dati, è stato necessario aggregarli. È essenziale notare il processo di aggregazione, specialmente nel caso dei dati di inquinamento. Ad esempio, i dati relativi all'inquinamento iniziano a essere registrati dall'una di notte. Ciò significa che il valore rilevato all'una rappresenta una somma cumulativa dei dati registrati dalla mezzanotte all'una. Questo principio si applica a ciascuna delle registrazioni successive. Di conseguenza, i dati relativi al traffico stradale sono stati aggregati in modo coerente con il formato dei dati precedentemente esaminati. Inoltre, va notato che i dati sull'intensità del traffico erano originariamente espressi in veicoli per ora. Pertanto, per ottenere il conteggio dei veicoli in un determinato istante, è stato necessario dividere i valori per 4. In seguito a questa operazione è stato possibile sommare i valori in un intero range.

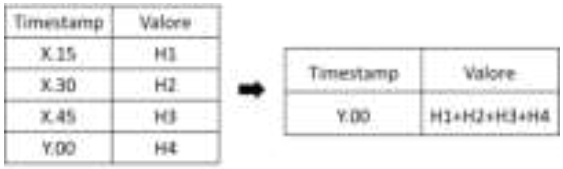


Figura 4.9: *Figura esplicativa del processo di aggregazione dei dati del traffico, con Y = X+1*

4.2 Dataset finale

Dopo aver completato la fase di selezione delle stazioni in conformità con i criteri precedentemente delineati e aver apportato le necessarie modifiche ai dataset, i tre insiemi di dati sono stati consolidati in un unico dataset omogeneo. La chiave di unione per questa procedura è stata la colonna **datetime**, opportunamente introdotta in ciascun dataset. Sfruttando la colonna datetime come punto di riferimento, è stato possibile creare un'unica riga di dati per ciascun istante temporale. È rilevante notare che i dataset relativi all'inquinamento atmosferico e ai dati meteorologici presentavano una frequenza oraria. Attraverso le modifiche precedentemente apportate, il dataset inerente al traffico stradale è stato adattato per condividere la stessa frequenza oraria, agevolando così la fusione dei dati tramite la colonna datetime.

```
df_28079035_4303 = df_pollution_28079035.merge(df_traffico, on =
↪  'datetime')
df_28079035_4303_W = df_28079035_4303.merge(df_weather, on = 'datetime')
df_28079035_4303_W.drop_duplicates(inplace = True)
```

Dopo questa procedura, il dataset finale rappresenta l'intero anno 2021, partendo dall'una del primo gennaio (inizio delle informazioni disponibili per i dati sulla qualità dell'aria) e terminando alle 23 del 31 dicembre. Considerando la frequenza oraria dei dati, il dataset consiste in 8759 osservazioni. Le features che costituiscono il dataset completo sono riportate nella Tabella 4.6.

Tabella 4.6: *Features del dataset unito e relative unità di misura*

FEATURE	UNITÀ DI MISURA
CO	mg/m^3
NO	$\mu g/m^3$
NO_2	$\mu g/m^3$
NO_x	$\mu g/m^3$
Flusso del traffico	Veicoli/h
Temperatura	°C
Umidità	%
Velocità del vento	Kph
Direzione del vento	Gradi
Pressione	mb

Dopo l'integrazione dei singoli insiemi di dati in un dataset unificato, è emersa la necessità di gestire i valori mancanti, un aspetto cruciale quando si affrontano serie temporali. A tal fine, è stato implementato un processo di interpolazione lineare per completare le osservazioni temporali mancanti, assicurando così la continuità della serie e la coerenza dei dati nel contesto temporale considerato.

4.2.1 Linear Interpolation

La problematica dei valori mancanti è comune nei dati del mondo reale, compresi i dati di serie temporali. Queste lacune nei dati possono sorgere da diverse fonti, tra cui la corruzione dei dati o la mancanza di registrazione in momenti specifici. Questi problemi possono derivare da varie circostanze, come possibili malfunzionamenti tecnici nei sensori, che talvolta si disattivano, o da errori nella memorizzazione dei dati su un server. Tuttavia, è importante sottolineare che i modelli di serie temporali richiedono dati completi per condurre analisi e modellazione accurate. Pertanto, prima di procedere con qualsiasi attività di modellazione o analisi di serie temporali, è cruciale affrontare e risolvere adeguatamente la questione dei dati mancanti Per affrontare questa sfida, esistono due approcci principali:

- **eliminare la riga con il valore mancante**: Questo approccio, tuttavia, è spesso inappropriato poiché può comportare la perdita di informazioni significative, inclusa la correlazione con le osservazioni circostanti.
- **Imputare le informazioni mancanti**: Questo è l'approccio preferito, in quanto stima o imputa i valori mancanti, consentendo di mantenere il set di dati completo.

Esistono diverse tecniche di imputazione dei dati, tra cui l'**interpolazione**, che stima i valori mancanti basandosi sui dati noti nelle vicinanze temporali. Nelle tecniche di

interpolazione, si considerano solo i dati passati e futuri noti per imputare i valori mancanti, contribuendo così a mantenere l'integrità del set di dati [28]. Nel nostro specifico contesto, abbiamo rilevato l'occorrenza di valori mancanti in alcuni timestamp dei dati provenienti dai sensori di flusso del traffico. Sebbene la presenza di valori mancanti non fosse diffusa su larga scala, è stato essenziale affrontare questa sfida, specialmente considerando la natura dei dati temporali. Per risolvere tale questione, abbiamo adottato l'approccio dell'**interpolazione lineare**. L'interpolazione lineare è una popolare tecnica di riempimento dei valori mancanti nei dataset. Questa tecnica cerca di individuare timestamp simili a quelli che presentano valori mancanti e riempie ciascun valore mancante con un valore medio. L'interpolazione lineare afferma che tra un punto di campionamento e il punto successivo c'è un gradiente costante. Considerando questa ipotesi, se l'ampiezza del punto i-esimo è x_i e l'ampiezza del punto i + 1 -esimo è x_{i+1}, quindi mantenendo il gradiente costante, il punto j-esimo tra x_i e x_{i+1} può essere calcolato nel seguente modo [29]:

$$\frac{x_{i+1} - x_i}{(i+1) - i} = \frac{x_j - x_i}{j - i} \tag{4.1}$$

$$x_j = \frac{x_{i+1} - x_i}{(i+1) - i} * (j - i) + x_i \tag{4.2}$$

4.2.2 Correlazioni tra il Traffico e i fattori inquinanti

Prima di procedere con l'addestramento delle reti neurali, è stato eseguito un ulteriore passo per valutare la **correlazione** tra i diversi gas selezionati. Questa selezione di gas è stata effettuata in conformità allo studio condotto da Awan et al.[13]. In statistica, la correlazione è un indice che misura la relazione tra due variabili. Quando diciamo che due variabili, A e B, sono correlate, intendiamo che i valori di A tendono a variare in modo coerente con i valori di B. Questa relazione non implica una causa ed effetto diretta, ma piuttosto rappresenta la capacità di una variabile di cambiare in risposta alle variazioni dell'altra. Nella valutazione della correlazione, è essenziale considerare il tipo di relazione tra le variabili e la sua forma. La relazione può essere di natura lineare o non lineare. Nel caso di una correlazione lineare, i punti sul piano cartesiano seguono una retta, mentre in una correlazione non lineare, la relazione tra le variabili può avere un andamento curvilineo, come una parabola o un'iperbole. Questa distinzione è fondamentale per comprendere come due variabili si influenzino reciprocamente [30].

Pearson correlation Il coefficiente di correlazione di Pearson (noto anche come "coefficiente di correlazione prodotto-momento") è una misura dell'associazione lineare tra due variabili continue X e Y . Ha un valore compreso tra -1 e 1. Più il valore si avvicina a zero, più la correlazione lineare è debole. Un valore positivo è indice di una correlazione positiva, in cui i valori delle due variabili tendono ad aumentare in parallelo. Un valore negativo è indice di una correlazione negativa, in cui il valore di una variabile tende ad

aumentare quando l'altra diminuisce [31].Di seguito viene riportata la formula [32]:

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.3}$$

dove $\bar{v}$ è la il valore medio della variabile v:

$$\bar{x} = \frac{\frac{1}{n}}{\sum_{i=1}^{n} v_i} \tag{4.4}$$

Calcolando le correlazioni di Pearson tra le variabili, abbiamo ottenuto i risultati riportati nella Figura 4.10.

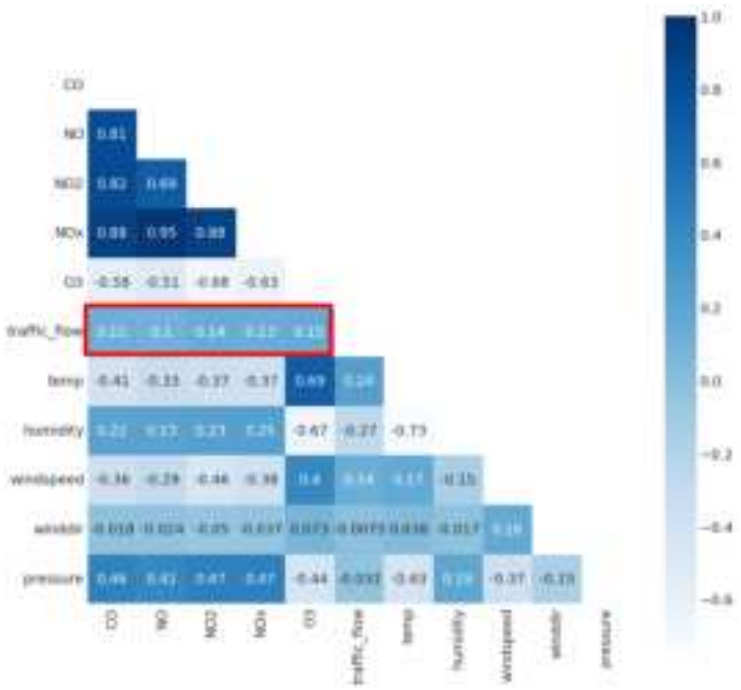


Figura 4.10: *Valori di correlazione di Pearson tra tutte le variabili*

Analizzando la correlazione tra il traffico e gli inquinanti, si osserva che sebbene i valori ottenuti non siano elevati, mantengono comunque una tendenza positiva. Questi risultati suggeriscono che vi è una relazione tra l'aumento del traffico stradale e l'aumento dei livelli di inquinanti. Come menzionato in precedenza, il coefficiente di correlazione di Pearson è in grado di rilevare correlazioni lineari tra due variabili. Tuttavia, al fine di esaminare la presenza di eventuali correlazioni non lineari, è stato impiegato il coefficiente di correlazione di Spearman.

Spearman La correlazione di Spearman [33] è un concetto fondamentale in statistica che misura il grado di relazione tra due variabili. Una delle caratteristiche distintive di questa misura è che l'unico requisito necessario per applicarla è che le variabili siano ordinabili e, se possibile, continue. Questa flessibilità la rende particolarmente utile quando non è possibile soddisfare l'assunzione di linearità richiesta dalla correlazione di Pearson o quando le variabili coinvolte sono rappresentate in scala ordinale. A differenza del metodo di Pearson, la correlazione di Spearman opera utilizzando i *ranghi* dei dati invece dei loro valori effettivi. Questo aspetto la rende una misura *non parametrica*, cioè senza l'assunzione di una distribuzione specifica per le variabili coinvolte. La correlazione di Spearman valuta la direzione e la forza della relazione monotona tra le variabili, assegnando valori compresi tra -1 e 1, con -1 che rappresenta una relazione inversamente proporzionale, 1 che rappresenta una relazione direttamente proporzionale e 0 che indica l'assenza di correlazione. È importante notare che questa misura si concentra solo sulla relazione monotona tra le variabili e non tiene conto di eventuali relazioni non monotone o non lineari. Dal punto di vista applicativo, il coefficiente r rappresenta un'istanza specifica del coefficiente di correlazione di Pearson, ma con la distinzione che i dati vengono convertiti in ranghi prima di calcolare il coefficiente. I ranghi sono una forma di trasformazione dei dati in cui le osservazioni vengono convertite in posizioni ordinali in base al loro valore in un insieme di dati. In altre parole, assegnare ranghi significa attribuire un numero a ciascun dato in modo che rifletta la sua posizione nella distribuzione dei dati, dall'osservazione più piccola (il rango più basso) a quella più grande (il rango più alto) [34].

$$\rho_s = \frac{\sum_{i=1}^{n}(r_i - \bar{r})(s_i - \bar{s})}{\sqrt{\sum_{i=1}^{n}(r_i - \bar{r})^2}\sqrt{\sum_{i=1}^{n}(s_i - \bar{s})^2}} \tag{4.5}$$

Spesso la formula 4.5 viene sostituita dalla formula seguente, che esegue un calcolo più semplice, in quanto calcola la differenza D tra i ranghi delle due misure di osservazione, ottenendo così:

$$\rho_s = 1 - \frac{\sum_i(D_i^2)}{N(N^2 - 1)} \tag{4.6}$$

Dove $D_i = r_i - s_i$ è la differenza dei ranghi, essendo r_i e s_i rispettivamente il rango della prima variabile e della seconda variabile della i-esima osservazione ed N è il numero complessivo di osservazioni. Calcolando le correlazioni di Spearman tra le variabili, abbiamo ottenuto i risultati presenti nella Figura 4.11. Come si può vedere cambiando tipo di correlazione i valori non sono troppo elevati, ma il legame delle variabili aumenta, questo rafforza ancora di più la nostra idea di relazione tra gas inquinanti e traffico stradale.

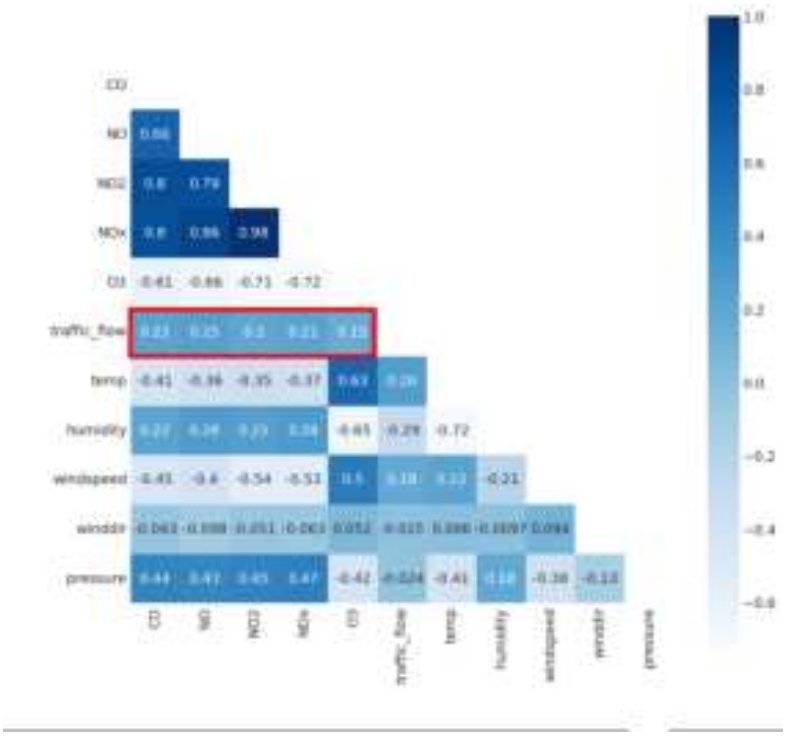
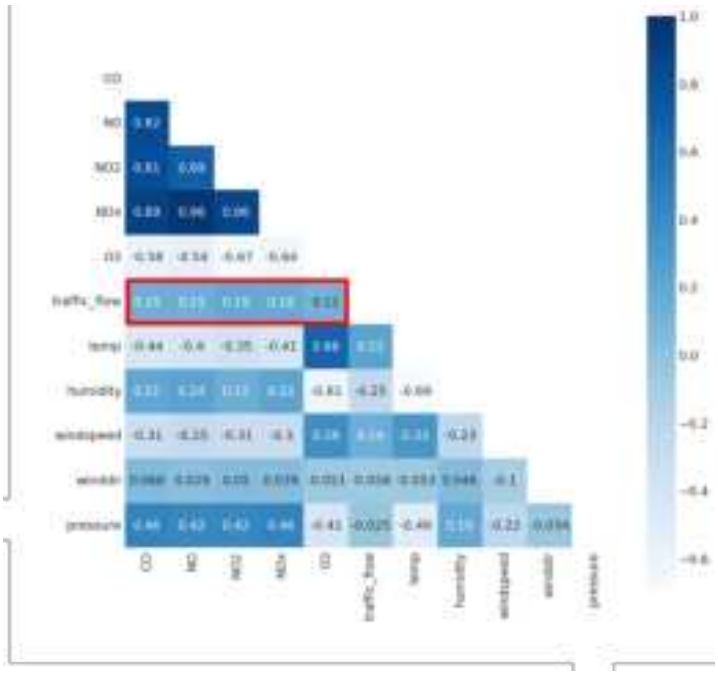


Figura 4.11: *Valori di correlazione di Spearman tra tutte le variabili*

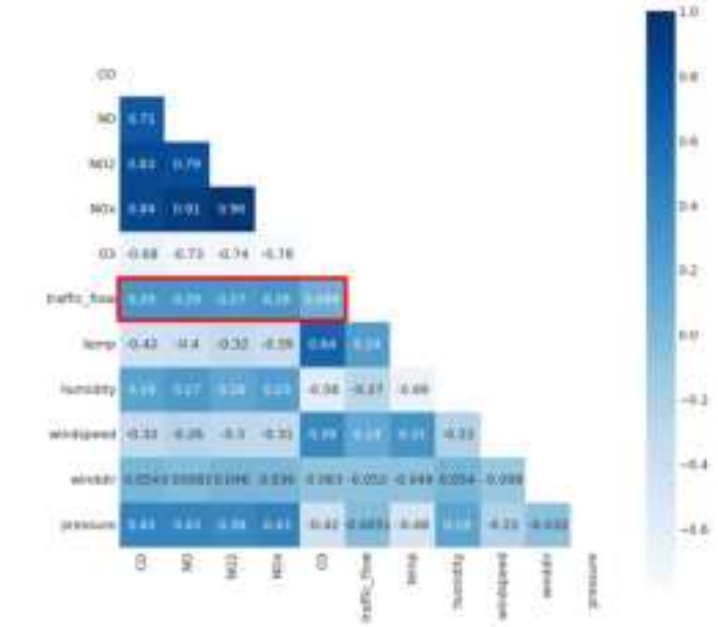
Come spiegato nell'abstracrt, sono stati introdotti alcuni fattori ambientali, in particolare la velocità del vento, che influisce sulla dispersione dei vari gas inquinanti. Per tale motivo si è cercato di calcolare le varie correlazioni con le sole righe del dataset che presentassero valori bassi in relazione alla velocità del vento, così da simulare una potenziale situazione ideale in assenza di vento.

```
df = df_28079035_4303_W.copy()
max_windspeed = df['windspeed'].max()
min_windspeed = df['windspeed'].min()
df['windspeed'] = (df['windspeed'] - min_windspeed) / (max_windspeed -
↪  min_windspeed)
low_windspeed = df.loc[df['windspeed'] < 0.15]
```

I valori ottenuti a seguito di tale operazione per entrambi i valori di correlazione sono riportati nella Figura 4.12.
Osservando i risultati ottenuti con la parziale assenza di vento, si nota un aumento dei valori di correlazione, suggerendo che la presenza o assenza di vento potrebbe influire sulla dispersione dei gas inquinanti. Nonostante i valori di correlazione non siano elevati, si intuisce una relazione positiva tra il traffico e gli inquinanti. Tuttavia, la correlazione misura la relazione tra due variabili, trascurando l'influenza combinata di tutte le variabili su una singola variabile target. Inoltre, dai risultati complessivi delle correlazioni,



(a) *Correlazione di Pearson*



(b) *Correlazioe di Spearman*

Figura 4.12: *Valori di Correlazione senza vento*

emerge che fattori meteorologici come temperatura, umidità e pressione sono correlati ai valori dei gas inquinanti. Pertanto, l'utilizzo di reti neurali offre la possibilità di rilevare le complesse relazioni multiple tra tutte le variabili nel dataset.

4.2.3 Feature engineering

La **feature engineering** (ingegneria delle caratteristiche) è una fase fondamentale nel processo di preparazione dei dati per l'analisi e la modellazione. In questa fase, ci concentriamo sulla *creazione* e sulla *trasformazione* delle variabili (caratteristiche) in modo da renderle più informative e adatte all'obiettivo dell'analisi. L'obiettivo della feature engineering è migliorare la qualità dei dati, aumentare le prestazioni dei modelli e consentire loro di catturare meglio i pattern sottostanti nei dati. Questa fase può includere la rimozione di variabili inutili o ridondanti, la creazione di nuove caratteristiche basate su quelle esistenti e la trasformazione delle caratteristiche per renderle più adatte all'analisi statistica o all'apprendimento automatico. Una delle prime operazioni effettuate in tale fase è stata l'operazione di trasformazione riguardante le variabili di direzione del vento e velocità del vento. La direzione del vento, espressa in gradi, potrebbe creare problemi quando viene utilizzata come input per un qualsiasi modello di machine learning. Ad esempio, se il vento soffia a 360° o 0°, dovrebbero essere considerati vicini e coerenti tra loro, ma in termini matematici potrebbero sembrare distanti. Inoltre, la direzione del vento potrebbe non avere importanza quando il vento è assente. Per facilitare l'interpretazione del modello, abbiamo convertito queste informazioni in un vettore di vento, che rappresenta in modo più adeguato la relazione tra la direzione e la velocità del vento [35]. Di seguito viene riportato il codice utilizzato per l'esecuzione di tale operazione:

```
wv = df.pop('windspeed')

# Convert to radians.
wd_rad = df.pop('winddir')*np.pi / 180

# Calculate the wind x and y components.
df['Wx'] = wv*np.cos(wd_rad)
df['Wy'] = wv*np.sin(wd_rad)
```

Oltre alla precedente modifica, sono state introdotte anche due variabili:

- **Ora**
- **Giorno della settimana**

L'introduzione di queste variabili è motivata dal fatto che il lavoro si basa su serie temporali, e tali feature si rivelano di fondamentale importanza durante la fase di addestramento, in quanto consentono alle diverse reti neurali di rilevare schemi e tendenze all'interno delle serie temporali. Una volta terminata questa fase sono state completate tutte le features a nostra disposizione per il training delle varie reti. Nella Tabella 4.7 espone tutte le feature a nostra disposizione.

Tabella 4.7: *Features complete del dataset*

FEATURE
CO
NO
NO_2
NO_x
Flusso del traffico
Temperatura
Umidità
Pressione
Wx
Wy
Ora
Giorno della settimana

4.3 Definizione del problema

Una volta raccolti tutti i dati necessari per il training delle reti neurali, è stato fondamentale categorizzare la tipologia di serie temporali con cui stavamo lavorando. Il primo passo è stato determinare se ci trovassimo di fronte a una serie temporale univariata o multivariata. Ecco una distinzione tra le due [36]:

- **Serie temporali univariate**: Questo tipo di serie è caratterizzato da una *singola variabile dipendente dal tempo*. Ad esempio, immaginiamo di raccogliere dati da un sensore che registra la temperatura di una stanza ogni secondo. In questo caso, la serie contiene solo un'unica dimensione, ossia la temperatura. L'analisi di una serie temporale univariata si basa sull'osservazione dei pattern nei dati passati di questa variabile singola.
- **Serie temporali multivariate**: Le serie temporali multivariate coinvolgono *più di una variabile di serie temporale*. Ogni variabile dipende non solo dai suoi valori precedenti ma è anche influenzata da altre variabili presenti nella serie. Queste interdipendenze tra le variabili sono utilizzate per la previsione dei valori futuri.

Nel nostro caso, ci troviamo di fronte a una **serie temporale multivariata**. La decisione di utilizzare reti neurali per affrontare questa tipologia di serie temporale è altamente valida. Questi modelli sono in grado di catturare non solo le dinamiche dei dati passati, ma anche l'interazione e l'influenza reciproca tra le diverse variabili presenti nella nostra serie temporale. L'approccio basato su reti neurali offre la flessibilità necessaria per modellare complesse relazioni tra le variabili e catturare pattern che potrebbero sfuggire ad altri approcci. È quindi un'opzione promettente per l'analisi e la previsione di serie

temporali multivariate.

Una distinzione fondamentale da considerare riguarda il tipo di modelli da utilizzare, in particolare tra modelli "single step" e modelli "multi-step":

- I modelli **single step** sono la scelta più semplice e prevedono il valore di una singola caratteristica, ovvero prevedono ciò che accadrà in un singolo passaggio temporale, ad esempio, un'ora nel futuro, *basandosi esclusivamente sulle condizioni attuali.*
- Tuttavia, un modello "single step" ha limitazioni significative. Questi modelli non tengono conto del contesto temporale e delle variazioni nei dati di input nel corso del tempo. Per superare queste limitazioni e ottenere previsioni più accurate, è necessario ricorrere a modelli **multi-step** che permettono di accedere a **informazioni da più fasi temporali** durante il processo di previsione

(a) *Single Step* ***(b)*** *Multi-step*

Figura 4.13: *Modelli per tipologia di input*

In altre parole, la scelta tra modelli "single step" e "multi-step" è cruciale per assicurarsi che il modello sia in grado di considerare il contesto temporale e le dinamiche complesse dei dati al fine di migliorare la qualità delle previsioni. Nel nostro caso, abbiamo adottato un approccio basato su modelli **multi-step**, come precedentemente illustrato. Questa scelta ci consente di tenere in considerazione il contesto temporale e le complesse dinamiche presenti nei dati.

Inoltre, vi è una distinzione ulteriore in base alla tipologia di output generato dai nostri modelli. Si distinguono due categorie principali:

- **Modello a singolo output (Single-Output Model)**: Questo modello genera *una sola previsione in uscita*, prevedendo il valore della variabile target in un singolo istante temporale avanti rispetto alla finestra temporale utilizzata per la previsione.
- **Modello a output multipli (Multi-Output Model)**: In questo caso, il modello genera *più previsioni in uscita*, utilizzando i dati di input passati per prevedere più valori futuri.

(a) Single-output *(b) Multi-output*

Figura 4.14: *Modelli per tipologia di output*

Nella nostra attività di ricerca, data la complessità del problema in esame, abbiamo adottato l'approccio con modelli **single-output**. I modelli single step si concentrano sulla previsione del valore della variabile target in un singolo passo temporale futuro, basandosi sulle condizioni attuali e sulle relazioni intercorrenti tra le variabili. Tuttavia, i modelli multi-output, che mirano a prevedere più valori futuri in un'unica previsione, possono essere complessi da implementare e richiedere una maggiore quantità di dati di addestramento. Inoltre, questi modelli possono essere più suscettibili a errori cumulativi man mano che ci si sposta ulteriormente nel futuro, a causa della propagazione degli errori in ciascun passo successivo. L'adozione di modelli single-output ci ha permesso di affrontare in modo più specifico le dinamiche tra le variabili e di mantenere una visione chiara delle relazioni tra input e output, semplificando così il processo di modellazione e l'interpretazione dei risultati. Al termine di questa indagine quindi la tipologia di modello da noi scelta è stata quella di un modello **multi-step**,**single output** al fine di poter lavorare al meglio con il nostro problema di **serie temporale multivariata**.

Per far si che i modelli fossero tutti della tipologia multi step, è stato necessario generare delle finestre temporali [37], scorrevoli, necessarie per la fase di addestramento. Di seguito viene riportato il codice utilizzato per la generazione delle finestre temporali:

```
def format_for_training(no_records,X,y):
  X_ret=[]
  Y_ret=[]
  for i in range(hops,no_records):
    X_ret.append(X[i-hops:i])
    Y_ret.append(y[i][0])
  X_ret,Y_ret = np.array(X_ret),np.array(Y_ret)

  return X_ret,Y_ret
```

Il seguente codice genera finestre temporali scorrevoli dai dati di una serie temporale. Il parametro **hops** indica quanti passi temporali precedenti devono essere considerati in ciascuna finestra. Per chiarire ulteriormente: sia x_t una variabile generica all'istante temporale t, e sia data la serie temporale X $= \{x_1, x_2, x_3, x_4\}$ e la serie Y $= \{y_1, y_2, y_3, y_4\}$. Utilizzando il codice proposto e impostando il valore del parametro hops su 3, si gene-

rerà la prima finestra X_ret= $\{x_1, x_2, x_3\}$ e Y_ret = $\{y_4\}$. Le finestre scorrono di un solo valore temporale alla volta, e la finestra temporale successiva sarà X_ret = $\{x_2, x_3, x_4\}$ e Y_ret= $\{y_5\}$. Nel nostro approccio, abbiamo configurato il parametro "hops" a 168, che corrisponde al numero totale di ore in una settimana. Questa selezione di 168 ore, rappresentante una settimana, è stata fatta per catturare l'evoluzione degli inquinanti atmosferici in un periodo in cui si verificano modelli ricorrenti, come quelli legati al traffico durante i giorni feriali e nei fine settimana. L'obiettivo è discernere le differenze tra i giorni lavorativi e i fine settimana. Inoltre, in questa finestra temporale, gli inquinanti hanno il tempo di consolidarsi, con alcuni di essi che possono permanere nell'aria per diverse ore o più. Questa scelta di periodo potrebbe consentire una previsione più accurata, considerando che l'intensità del traffico mostra pattern differenti nei giorni feriali rispetto ai fine settimana. Ad esempio, una settimana all'interno di un particolare mese, come dicembre prima delle festività natalizie, potrebbe presentare un aumento significativo del traffico e, di conseguenza, dell'inquinamento. Diversi mesi possono mostrare notevoli differenze nei livelli di traffico e inquinamento. La decisione di adottare una finestra temporale di una settimana è motivata dalla volontà di catturare queste variazioni, mantenendo al contempo un periodo di osservazione breve e rilevante per l'acquisizione dei dati. Per condurre la nostra analisi, abbiamo introdotto una distinzione tra la variabile target e le altre variabili. Questo approccio ci consente di utilizzare gas inquinanti e parametri atmosferici per predire l'intensità del traffico.

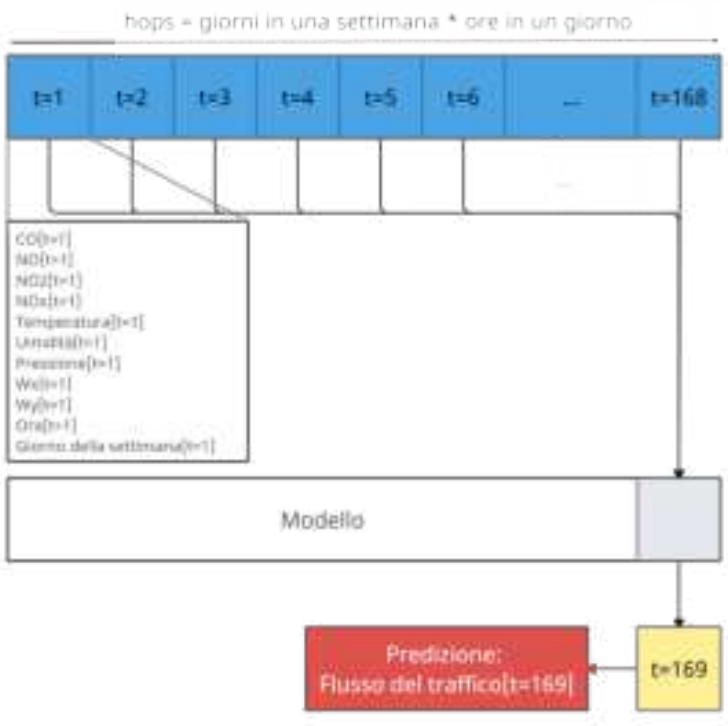


Figura 4.15: *Schema riassuntivo della tipologia di modelli realizzati*

Nella Figura 4.15, presentiamo il modello che abbiamo sviluppato, la scelta della dimensionalità della finestra, e la variabile target che intendiamo predire.

4.4 Modelling

Nella seguente sezione verranno presentati i vari modelli creati ed utilizzati. Il Primo modello implementato è stata una rete neurale **LSTM**, di seguito si riporta il codice:

```
model = Sequential()
#Adding the first LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True, input_shape =
↪  (hops,no_cols)))
model.add(Dropout(0.2))
# Adding a second LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
# Adding a third LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))
# Adding a fourth LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50))
model.add(Dropout(0.2))
# Adding the output layer
model.add(Dense(units = 1))
```

Nel processo di sviluppo del modello predittivo, abbiamo implementato una rete neurale ricorrente (LSTM) utilizzando il framework Keras. La struttura del modello è stata progettata per catturare le relazioni complesse e ricorsive nei dati di serie temporali. La scelta di utilizzare quattro strati LSTM, ognuno seguito da un layer di Dropout per la regolarizzazione, è stata motivata dalla complessità intrinseca dei dati atmosferici e del traffico. Il primo strato LSTM è configurato per restituire sequenze complete, consentendo al modello di apprendere pattern a diversi livelli di dettaglio. Gli strati successivi continuano a lavorare su sequenze, mantenendo la capacità di acquisire informazioni contestuali importanti. Ogni strato è seguito da uno strato di Dropout per prevenire overfitting e migliorare la generalizzazione del modello. L'ultimo strato consiste in un layer di output denso con un singolo neurone, in quanto stiamo generando un modello single-output per prevedere il valore dell'intensità del traffico. L'intero modello è stato progettato per apprendere in maniera efficiente dai dati di serie temporali, acquisendo rappresentazioni utili dei pattern presenti nei gas inquinanti e nei parametri atmosferici.

Dropout Il termine **dropout** si riferisce alla pratica di eliminare temporaneamente alcuni nodi (input e hidden layer) in una rete neurale, come mostrato in Figura 4.16. Tutte le connessioni in avanti e all'indietro con un nodo eliminato vengono temporaneamente rimosse, creando così una nuova architettura di rete derivata dalla rete di partenza. I nodi vengono eliminati con una probabilità di dropout, indicata con p. Nel

(a) Rete neurale Standard *(b) Dopo aver applicato il dropout*

Figura 4.16: *Esempio di Dropout in una rete neurale.*

problema dell'overfitting, il modello apprende il rumore statistico. In sostanza, l'obiettivo principale dell'addestramento è ridurre la funzione di perdita, considerando tutte le unità (neuroni). Nell'overfitting, un'unità potrebbe cambiare in modo da correggere gli errori delle altre unità. Questo porta a co-adattamenti complicati, che a loro volta generano il problema dell'overfitting perché questa complessa co-adattazione non riesce a generalizzare sul dataset non visto. Utilizzando il dropout, si impedisce a queste unità di correggere gli errori delle altre unità, evitando così la co-adattazione. Quindi, eliminando casualmente alcune unità (nodi), si costringono gli strati ad assumersi più o meno responsabilità dell'input, adottando un approccio probabilistico. Ciò assicura che il modello si generalizzi e, di conseguenza, riduca il problema dell'overfitting [38].

Il secondo modello implementato è stato una **GRU**, di seguito si riporta il codice:

```
# Create GRU model
model2 = Sequential()
# Input layer
model2.add(GRU (units = 64, return_sequences = True, input_shape =
↪  (hops,no_cols)))
model2.add(Dropout(0.2))
# Hidden layer
model2.add(GRU(units = 64))
model2.add(Dropout(0.2))
model2.add(Dense(units = 1))
```

Il modello GRU (Gated Recurrent Unit) rappresenta una scelta avanzata per la nostra analisi delle serie temporali. A differenza delle reti LSTM, le unità GRU introducono una struttura più semplice ma altrettanto potente per catturare i pattern temporali nei dati. Nel nostro modello GRU, abbiamo configurato un layer di input che restituisce sequenze, ideale per catturare le relazioni complesse tra le variabili di input nei diversi istanti temporali. Successivamente, abbiamo aggiunto uno strato nascosto GRU con 64 unità, che sfrutta le sue capacità di apprendimento per identificare i pattern ricorrenti nei dati. Per evitare il rischio di overfitting, abbiamo introdotto dropout con un tasso

del 20% dopo entrambi i layer. Infine, il layer di output, con un'unità, è progettato per generare la previsione finale dell'intensità del traffico.

Il modello successivo che è stato implementato è stato quello di una **LSTM bidirezionale** semplice con un unico livello. Di seguito si riporta il codice:

```
model3 = Sequential()
model3.add(Bidirectional(LSTM(50, activation='relu'),
↪  input_shape=(hops,no_cols)))
model3.add(Dense(1)
```

Il modello Bidirectional LSTM rappresenta un'ulteriore avanzamento nella nostra analisi delle serie temporali. Introducendo la bidirezionalità, questo modello è in grado di catturare le relazioni nei dati non solo dai passati istanti temporali ma anche dai futuri. Nel nostro caso, abbiamo implementato un layer Bidirectional LSTM con 50 unità e attivazione ReLU. Questo strato è progettato per apprendere pattern complessi all'interno dei dati, sfruttando la sua capacità di elaborare informazioni sia in avanti che all'indietro nel tempo. Successivamente, abbiamo aggiunto un layer di output con un'unità, che genererà la previsione finale dell'intensità del traffico. La scelta di Bidirectional LSTM riflette la nostra attenzione alla complessità delle dinamiche temporali nei dati, consentendo al modello di acquisire una comprensione più approfondita delle relazioni causalità. Inoltre, è importante sottolineare che l'attivazione ReLU nel layer LSTM contribuisce alla non linearità del modello, consentendo di catturare relazioni complesse nei dati. Questo modello Bidirectional LSTM si inserisce all'interno della nostra strategia complessiva per l'analisi delle serie temporali.

Il penultimo modello implementato è stato quello di una **rete neurale convoluzionale**. Di seguito si riporta il codice:

```
model4 = Sequential()
model4.add(InputLayer((hops,no_cols)))
model4.add(Conv1D(200, kernel_size=2, activation='relu'))
model4.add(MaxPooling1D(pool_size=4))
model4.add(Flatten())
model4.add(Dense(8, 'relu'))
model4.add(Dense(1, 'linear'))
```

Il modello proposto presenta un approccio innovativo all'analisi delle serie temporali, incorporando un layer Conv1D nella nostra architettura. A differenza degli approcci basati su reti ricorrenti, l'utilizzo di un layer Conv1D ci consente di sfruttare le caratteristiche di convoluzione per estrarre pattern rilevanti dai dati temporali. In questo modello, abbiamo inizializzato il layer Conv1D con 200 filtri e una dimensione di kernel pari a 2, utilizzando l'attivazione ReLU per introdurre non linearità. Il layer successivo, MaxPooling1D con una dimensione di pool di 4, contribuisce a ridurre la dimensionalità dell'output convoluzionale, consentendo di mantenere le caratteristiche più significative.

Successivamente, abbiamo introdotto un layer Flatten per preparare l'output convoluzionale per l'input al layer Dense. Il primo layer Dense ha 8 unità con attivazione ReLU e fornisce uno strato denso con connessioni completamente collegate per apprendere relazioni complesse. Infine, il layer di output Dense con attivazione lineare è progettato per generare la previsione finale dell'intensità del traffico. Questo modello Conv1D rappresenta una prospettiva alternativa nella nostra ricerca, sfruttando le capacità di convoluzione per rilevare pattern temporali in modo più efficiente. Questa diversificazione nella nostra architettura riflette il nostro impegno a esplorare diverse tecniche per migliorare la precisione delle previsioni sulle serie temporali complesse.

Nel modello finale, abbiamo adottato un approccio **ibrido** che sfrutta sia le reti LSTM che i layer convoluzionali. Questa combinazione strategica mira a utilizzare appieno le capacità distintive di entrambe le architetture. Le LSTM sono note per la loro abilità di catturare dipendenze a lungo termine nelle sequenze temporali, consentendo al modello di comprendere relazioni complesse. Al contempo, l'integrazione di layer convoluzionali aggiunge un elemento di apprendimento delle caratteristiche temporali, consentendo alla rete di identificare pattern significativi in modo efficiente. Questa sinergia tra LSTM e layer convoluzionali riflette la nostra volontà di massimizzare la capacità predittiva del modello, sfruttando le forze complementari di entrambe le architetture neurali. Il risultato è un modello robusto in grado di cogliere sia le relazioni a lungo termine che le caratteristiche temporali rilevanti nel nostro contesto. Di seguito viene riportato il codice relativo a tale modello:

```
model5 = Sequential()
model5.add(Conv1D(filters=168, kernel_size=2, padding='same',
↪  activation='relu',input_shape(hops, no_cols)))
model5.add(MaxPooling1D(pool_size=4))
model5.add(LSTM(64))
model5.add(Dropout(0.2))
model5.add(Dense(1, 'linear'))
```

Il primo layer convoluzionale, con 168 filtri e una dimensione del kernel pari a 2, è progettato per catturare e apprendere caratteristiche rilevanti nella serie temporale del traffico. Successivamente, un layer di MaxPooling riduce la dimensionalità delle caratteristiche apprese, preparando il terreno per il successivo layer LSTM. Il layer LSTM, con 64 unità, consente al modello di catturare dipendenze a lungo termine nelle sequenze temporali, contribuendo così a una comprensione più approfondita dei modelli di traffico nel corso del tempo. Per mitigare il rischio di overfitting, è stato aggiunto un layer di Dropout con una probabilità del 20%. Infine, il modello si conclude con un layer denso con attivazione lineare, che produce la previsione del traffico.

Capitolo 5

Addestramento delle Reti Neurali e Analisi dei Risultati

Questo capitolo si concentra sul processo di addestramento delle reti neurali. Verrà approfondita la suddivisione del dataset in training set, validation set e test set, esaminando le metriche utilizzate per l'addestramento delle reti e la valutazione delle loro prestazioni. Saranno presentate dettagliatamente le curve di apprendimento, che forniscono un'analisi dettagliata dell'evoluzione del processo di addestramento nel corso del tempo, e verranno messi a confronto i risultati ottenuti da ciascuna rete neurale.

5.1 Training delle reti

5.1.1 Suddivisione in Training, Validation e Test set

Un passaggio essenziale prima di procedere con l'addestramento delle reti neurali è la suddivisione del dataset in tre insiemi distinti: *Training Set*, *Validation Set* e *Test Set*.

- Il **Training Set** contiene i dati dedicati esclusivamente all'addestramento, permettendo al modello di apprendere le relazioni tra le variabili di input (X) e l'output desiderato (Y), che rappresenta la variabile da predire. Tuttavia, utilizzando solo il Training Set, potremmo incorrere nel rischio di overfitting, in cui il modello potrebbe adattarsi eccessivamente ai dati di addestramento, ma faticare a generalizzare su nuovi dati. Per mitigare questo problema, viene introdotto il *Validation Set*, un insieme di dati mai visto dal modello durante l'addestramento.
- Il **Validation Set** è utilizzato per validare i risultati ottenuti durante la fase di addestramento, offrendo una valutazione imparziale delle prestazioni del modello.
- Infine, il **Test Set** viene impiegato una volta completato l'addestramento delle reti neurali. Questo set di dati consente di valutare e visualizzare le prestazioni complessive del modello, fornendo una misura finale del suo funzionamento.

La suddivisione da noi effettuata è stata la seguente: il 70% dei dati è stato utilizzato per il training set (6131 ore), dal primo gennaio fino alle ore 11 del 13 settembre; Il 20% dei dati è stato utilizzato per il validation set (1751 ore), dalle 12 del 13 di settembre fino alle ore 10 del 24 di novembre; il 10% dei dati è stato utilizzato per il test set (877 ore), dalle ore 11 del 24 novembre fino alle 23 del 31 dicembre.

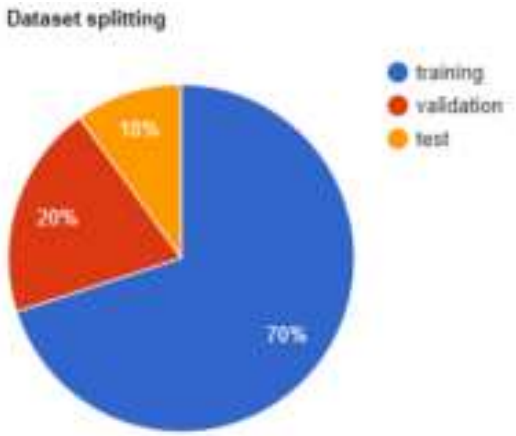


Figura 5.1: *Suddivisione del Dataset*

Prima di procedere con l'addestramento delle reti, è stato eseguito un ulteriore passo preliminare per normalizzare i valori dei dati.

5.1.2 Data Normalizzation

La **normalizzazione** dei dati è una delle fasi cruciali nella pre-elaborazione dei dati, garantisce la qualità dei dati prima di utilizzarli come input nei modelli di machine/deep learning ed è necessaria quando le caratteristiche hanno diverse scale di valori. Questa differenza di scala può compromettere le prestazioni di un modello di machine/deep learning, che potrebbe, ad esempio, essere intrinsecamente influenzato da una feature avente scala maggiore, ma questo non significa che sia la feature più importante. La normalizzazione contribuisce a ridurre il tempo di addestramento. Esistono diverse tecniche di normalizzazione dei dati, tra cui min-max, normalizzazione mediana e scalatura decimale Z-score. Nel nostro lavoro di ricerca tesi, abbiamo utilizzato la tecnica di normalizzazione più popolare, la normalizzazione min-max.

Min-Max Normalizzation La normalizzazione min-max [39] mappa i dati in intervalli predefiniti, [0,1] o [−1,1]. I valori di ogni attributo nei dati vengono definiti in base ai loro valori minimo e massimo. Se indichiamo l'attributo nei dati con *Atr*, il suo valore con *a_val*, il suo valore normalizzato con *a_norm* e l'intervallo predefinito come [*lower_lim*, *higher_lim*], allora l'equazione seguente può essere utilizzata per calcolare i

valori normalizzati nell'intervallo [*lower_lim*, *higher_lim*]:

$$a\_norm = lower\_lim + \frac{(higher\_lim - lower\_lim) \times (a\_val - min(Atr)}{max(Atr) - min(Atr))} \tag{5.1}$$

Considerando higher_lim = 1 e lower_lim = 0 quindi:

$$a\_norm = \frac{a\_val - min(Atr)}{max(Atr) - min(Atr)} \tag{5.2}$$

5.1.3 Parametri Training

In Keras attraverso il codice *model.compile()*, si configurano i dettagli del processo di addestramento del modello. Di seguito verrà riportata una sottosezione per ciascun parametro settato in tale comando.

5.1.3.1 Loss

Il parametro "**loss**" all'interno di *model.compile()* specifica la funzione di perdita da minimizzare durante l'addestramento. Nel nostro contesto si sono individuate tre metriche di valutazione: *Mean Absolute Error*, *Mean Squared Error* e *Root Mean Squared Error* [40].

Mean Absolute Error MAE è la forma più semplice di metrica che si può utilizzare per problemi di previsione. Si prende la media dei residui del valore effettivo e del valore previsto. Vengono presi i valori assoluti di ciascun residuo, così che residui positivi e negativi non si annullino a vicenda.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5.3}$$

In questa metrica ogni residuo sarà proporzionale all'errore totale, il che significa che anche gli errori di grandi dimensioni saranno lineari all'errore totale (tratta tutti gli errori allo stesso modo). Un valore basso di MAE suggerirà che il nostro modello è ottimo per le previsioni, mentre un valore elevato di MAE suggerisce che il nostro modello è cattivo nelle previsioni. Tuttavia, miriamo a sviluppare modelli che non commettano frequenti errori di dimensioni significative, quindi necessitiamo di una metrica che sanzioni in modo più severo gli errori maggiori rispetto a quelli minori. Nonostante questo difetto, tale metrica può essere utilizzata per valutare il miglioramento dei modelli, anche se in maniera non troppo dettagliata.

Mean Squared Error MSE rappresenta la media della differenza quadrata (diversamente da prima in cui veniva utilizzato il valore assoluto) tra i valori originali e quelli

previsti nel set di dati. Misura la varianza dei residui.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5.4}$$

A differenza del MAE dove ogni residuo è proporzionale all'errore totale, l'errore cresce quadraticamente in MSE. Ciò significa, in definitiva, che MSE avrà un errore totale maggiore a causa della presenza di valori anomali rispetto a quanto avverrebbe in MAE. Un MSE più elevato indica che il modello sarà penalizzato per aver fatto previsioni che differiscono notevolmente dal valore effettivo. Questo comporta che una grande differenza tra i valori previsti e quelli effettivi sarà più penalizzata in MSE che in MAE. MSE è una metrica utile, ma è difficile da interpretare in quanto, per definizione, prevede la quadratura dei termini di errore e, pertanto, non ha le stesse unità di misura del valore che vogliamo prevedere.

Root Mean Squared Error RMSE è la radice quadrata dell'errore quadratico medio. Misura la deviazione standard dei residui.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{5.5}$$

MSE ha un valore più alto perché viene elevato a potenza. Per portare questo valore sulla stessa scala di quello dell'errore di previsione, utilizziamo RMSE e ciò facilita l'interpretazione in quanto sia MSE che RMSE, elevando a potenza i residui, risultano ugualmente influenzati dai valori anomali.

Ai fini dell'addestramento di tutte e 5 le reti neurali è stata utilizzata come funzione di loss la **MeanSquaredError()**, tutte le tre metriche precedentemente esposte verranno utilizzate per la valutazione complessiva dei modelli.

5.1.3.2 Optimizer

Il parametro "**optimizer**" all'interno di *model.compile()* consente di scegliere la funzione di ottimizzazione per aggiornare i pesi del modello durante l'addestramento. Le strategie di ottimizzazione svolgono un ruolo cruciale nell'addestramento delle reti neurali profonde. Affrontare la complessità di addestrare reti di dimensioni considerevoli richiede l'impiego di ottimizzatori altamente performanti per migliorare l'efficienza e la rapidità del processo. Gli ottimizzatori gestiscono l'aggiornamento dei pesi della rete per minimizzare la funzione di perdita durante il processo di apprendimento. Keras mette a disposizione diversi algoritmi di ottimizzazione, ciascuno con i suoi vantaggi e svantaggi. Nel nostro lavoro di ricerca abbiamo usato l'algoritmo **Adam** [41], un'estensione dello **stochastic gradient descent** (SGD). L'approccio di ottimizzazione di Adam si basa sulla regolazione del tasso di apprendimento per ciascun parametro, utilizzando

informazioni derivanti dai gradienti calcolati in iterazioni precedenti. Ciò favorisce una convergenza più rapida e precisa rispetto ai metodi con tasso di apprendimento fisso, come l'SGD. L'algoritmo Adam combina gli aspetti positivi di altri due algoritmi di ottimizzazione, **Momentum** [42] e **RMSProp**. Immaginiamo una palla da bowling che si muove lungo una discesa su una superficie liscia: inizialmente, la palla parte con una velocità modesta, ma nel tempo accumula slancio, raggiungendo la sua massima velocità. Questo concetto costituisce la base dell'algoritmo di ottimizzazione del momento. Ad ogni iterazione, l'algoritmo del momento sottrae il gradiente corrente dal vettore del momento m e aggiorna i pesi di conseguenza. Questo mette in evidenza che il ruolo del gradiente è accelerare il processo, non determinare direttamente la velocità. Per simulare un meccanismo di attrito e impedire un aumento eccessivo del momento, l'algoritmo introduce un nuovo iperparametro β, che può essere impostato tra 0 (alto attrito) e 1 (nessun attrito). Un valore comune per β è 0.1.

Algorithm 1 Algoritmo di ottimizzazione momento

$m \leftarrow m - \eta \nabla_\theta J(\theta)$ ▷ Update momentum
$\theta \leftarrow \theta + m$ ▷ Update weights using momentum

L'algoritmo di **discesa del gradiente** (GD) sceglie, ad ogni iterazione, la direzione di massima diminuzione, che potrebbe non essere immediatamente orientata verso l'ottimo globale. Sarebbe vantaggioso se l'algoritmo fosse in grado di correggere la sua direzione in modo da orientarsi anticipatamente verso l'ottimo globale. L'algoritmo **AdaGrad** implementa questa correzione riducendo il vettore del gradiente lungo le direzioni più ripide.

Algorithm 2 Algoritmo di ottimizzazione AdaGrad

$s \leftarrow s + \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$ ▷ Update accumulated squared gradient
$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta) \oslash \sqrt{s + \epsilon}$ ▷ Update model parameters with AdaGrad

L'algoritmo AdaGrad, sebbene efficace nel ridurre la dimensione del passo lungo le direzioni delle feature più ripide, presenta un inconveniente nella sua tendenza a ridurre la velocità troppo rapidamente, rischiando di non convergere all'ottimo globale. Per mitigare questo problema, l'algoritmo RMSProp adotta un approccio diverso. A differenza di AdaGrad, RMSProp accumula solo i gradienti delle iterazioni più recenti, introducendo un apposito iperparametro di attenuazione.

Algorithm 3 Algoritmo di ottimizzazione RMSProp

$s \leftarrow \beta s + (1 - \beta)(\nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta))$ ▷ Update accumulated squared gradient
$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta) \oslash \sqrt{s + \epsilon}$ ▷ Update model parameters with AdaGrad

L'algoritmo Adam, abbreviativo di "**Adaptive Moment Estimation**", combina le idee alla base degli algoritmi di ottimizzazione del momento e RMSProp.

5.1.3.3 Model Checkpoint

La tecnica di **checkpointing** rappresenta una strategia di tolleranza agli errori per i processi di lunga durata, in cui viene creata un'istantanea dello stato del sistema in caso di eventuali guasti. Questo approccio consente di mitigare gli effetti negativi di un'interruzione, consentendo l'utilizzo diretto del checkpoint o come punto di partenza per una nuova esecuzione, riprendendo l'attività dal punto in cui è stata interrotta. Nell'ambito dell'addestramento di modelli di deep learning, il punto di controllo corrisponde ai pesi del modello. Questi pesi possono essere sfruttati per effettuare previsioni dirette o come base per un ulteriore addestramento continuo. Keras fornisce un'implementazione di questa funzionalità attraverso un'API di *callback*. La classe di **callback ModelCheckpoint** [43] offre la possibilità di definire la posizione in cui salvare i pesi del modello, specificare il nome del file e indicare le condizioni che devono verificarsi per creare un checkpoint del modello. L'API consente inoltre di specificare quale metrica monitorare, come ad esempio perdita o accuratezza nel set di dati di training o di convalida. È possibile indicare se la ricerca di un miglioramento deve massimizzare o minimizzare il valore della metrica. Infine, è possibile personalizzare il nome del file utilizzato per memorizzare i pesi, incluso l'uso di variabili come il numero di epoche o il valore della metrica. Durante il processo di addestramento del modello, il ModelCheckpoint può essere integrato chiamando la funzione *fit()* sul modello, finalizzando così la fase di training della rete. Tramite le opzioni fornite da questa funzione è possibile preservare il modello con la performance migliore, determinata dalla grandezza monitorata. Impostando il parametro 'save_best_only' su True, il salvataggio sarà riservato esclusivamente al modello migliore. Nel nostro caso quindi la funzione di callback, per ciascuno dei cinque modelli, è stata impostata con i seguenti parametri:

```
cpC = ModelCheckpoint("filepath", monitor="val_loss", save_best_only=True,
↪  mode='min')
```

5.1.4 Early Stopping

Keras supporta l'interruzione anticipata dell'addestramento tramite una *callback* chiamata **EarlyStopping** [44]. Questa callback consente di specificare la misura delle prestazioni da monitorare e, una volta che tale metrica smette di soddisfare la condizione prestabilita, interrompe il processo di addestramento. Il "monitor" permette di specificare la misura di prestazione da monitorare al fine di terminare l'allenamento. In base alla scelta della misura di prestazione, con l'argomento "mode" dovrà essere specificato se l'obiettivo della metrica scelta è aumentare (massimizzare o 'max ') o diminuire (minimizzare o 'min '). Possiamo tenere conto di ciò aggiungendo un ritardo all'attivazione in termini di numero di epoche in cui non vorremmo vedere alcun miglioramento. Questo può essere fatto impostando l'argomento "patience ". Nel nostro caso questa callback è stata utilizzata congiuntamente alla Model Checkpoint solo nell'ultimo modello. La funzione è stata impostata con i seguenti parametri:

```
cpE = EarlyStopping(monitor='val_loss', mode='min', patience=25)
```

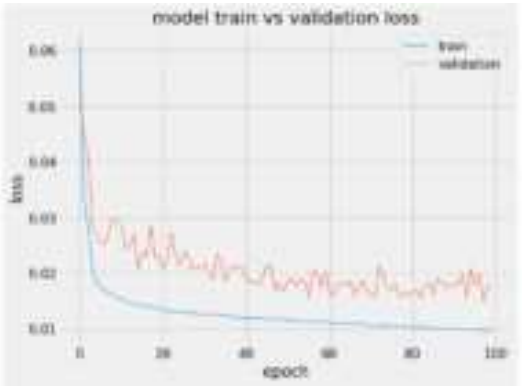
Una volta definiti tutti i parametri per tutti e 5 i modelli sono stati definiti i seguenti comandi:

```
model.compile(loss=MeanSquaredError(),
↪  optimizer=Adam(learning_rate=0.0001))
history = history = model.fit(X_train, Y_train,
↪  validation_data=(X_validation, y_validation),
↪  epochs=100,batch_size=64, callbacks=[cpC])
```

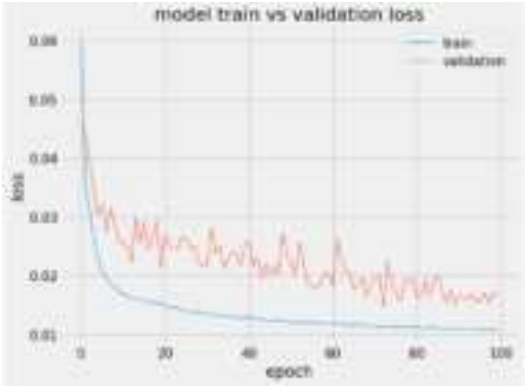
L'unica variazione si riscontra nel numero di epoche del modello Bidirezionale, ridotto a 32 anziché 100. Tale scelta è motivata dalla natura più semplice del modello e da problematiche riscontrate nell'andamento delle curve di apprendimento, soprattutto oltre un certo numero di epoche. Nel caso dell'ultimo modello, nell'array *callbacks* è stata inclusa anche la variabile 'cpE', precedentemente definita.

5.1.5 Learning Curves

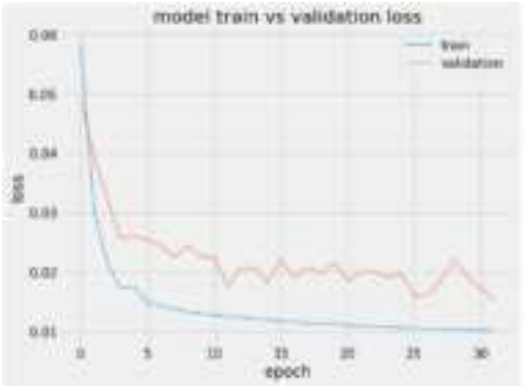
Le curve di apprendimento, essenziali nel contesto dei modelli di machine learning, rappresentano uno strumento cruciale per monitorare e valutare le performance di un modello durante il processo di addestramento e validazione. Forniscono informazioni dettagliate sull'apprendimento del modello dai dati e rivelano eventuali segnali di *underfitting* o *overfitting*. L'**underfitting** si verifica quando il modello non riesce ad apprendere in modo sufficiente dai dati di addestramento, risultando incapace di ridurre adeguatamente l'errore su questo insieme. Al contrario, l'**overfitting** si manifesta quando il modello si adatta eccessivamente ai dati di addestramento, includendo dettagli insignificanti o rumore statistico. Questo fenomeno rende il modello troppo specializzato, compromettendo la sua capacità di generalizzare correttamente su nuovi dati e aumentando l'errore di generalizzazione. Le curve di training mostrano l'andamento della loss del modello sui dati di addestramento in funzione delle epoche o del tempo. La loss di training misura quanto bene il modello si adatta ai dati di addestramento. D'altra parte, le curve di validazione illustrano la dinamica della loss su dati completamente estranei al modello, fornendo un indicatore critico delle sue performance in situazioni in cui non ha esperienza diretta. La loss di validazione è essenziale per valutare la capacità del modello di estendere le sue conoscenze ed effettuare previsioni accurate su nuovi dati. L'analisi della forma e dell'andamento delle curve di apprendimento guida la valutazione del comportamento del modello e suggerisce eventuali modifiche per migliorare il processo di apprendimento e le prestazioni complessive. Questa sezione presenta i risultati derivanti dall'analisi delle curve di apprendimento.



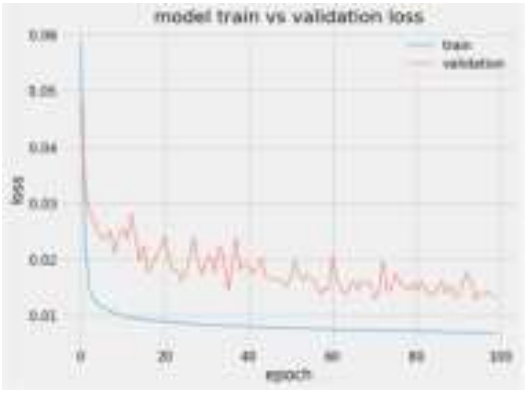
(a) *Curve di apprendimento LSTM*



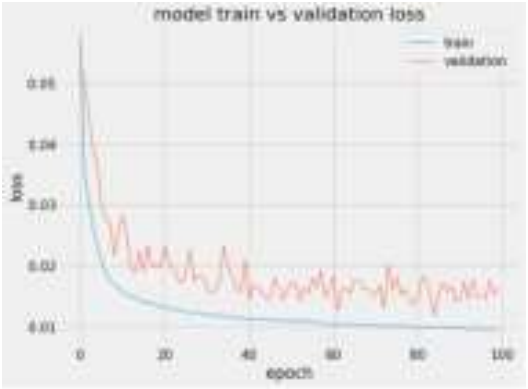
(b) *Curve di apprendimento GRU*



(c) *Curve di apprendimento Bidirectional LSTM*



(d) *Curve di apprendimento Rete Convoluzionale*



(e) *Curve di apprendimento LSTM-CONV*

Figura 5.2: *Curve di apprendimento per i vari modelli.*

Le curve di apprendimento dei cinque modelli di machine learning implementati sono illustrate nelle immagini nella Figura 5.2. La curva *blu* rappresenta l'andamento del *training set*, mentre la curva *arancione* rappresenta il *validation set*. In tutte e cinque le immagini, le curve di training mostrano una tendenza regolare, con alcune fluttuazioni che conferiscono una leggera irregolarità. D'altra parte, le curve di validation mostrano un andamento meno regolare, con valori che salgono e scendono. Nonostante le irregolarità, in tutti e cinque i casi, l'errore calcolato sul validation set tende a diminuire, indicando un miglioramento nel corso delle epoche.

Le curve con fluttuazioni di ampiezza minore sono osservabili nel primo modello implementato, LSTM, e nell'ultimo modello che combina LSTM e rete convoluzionale. Questi modelli presentano fluttuazioni con valori più contenuti. Nel caso del modello di rete ricorrente bidirezionale, il numero di epoche è inferiore a causa dell'andamento più irregolare della curva, che non mostrava miglioramenti significativi con il passare delle iterazioni. Nonostante l'irregolarità, si nota che le curve del validation set rimangono costantemente al di sopra di quelle del training set. Non emergono segni evidenti di underfitting, e non si osserva un overfitting significativo, poiché le curve non si discostano in modo eccessivo, e tutte mostrano una tendenza al declino dei valori di loss nel validation set nel corso delle epoche. I modelli che evidenziano performance superiori nei dati di validation sono il secondo e l'ultimo modello implementato, caratterizzati da un andamento decrescente leggermente più pronunciato rispetto alle curve precedenti, come si può osservare dai grafici.

Per prevenire l'overfitting, è stata utilizzata la funzione di callback ModelCheckpoint per tutti e cinque i modelli, consentendo di salvare il modello con il valore minimo di loss sul validation set. Tale modello è stato successivamente impiegato nella fase di inferenza. Nonostante le irregolarità, le curve di validation mantengono una distanza accettabile rispetto ai corrispondenti valori di loss nel training set, seguendo il trend decrescente. Inoltre, per quanto riguarda l'ultimo modello, sono presenti tutte le 100 epoche, nonostante l'uso della funzione di callback EarlyStopping, indicando un miglioramento del valore di validation nel tempo.

Nell'ambito della sezione dedicata all'addestramento delle reti neurali, è cruciale sottolineare una delle sfide predominanti in questo contesto: i considerevoli tempi richiesti per un corretto addestramento. La durata dell'addestramento è direttamente proporzionale sia al numero di dati impiegati durante questa fase, sia alle dimensioni individuali di ciascun dato. In molti casi, l'addestramento di una singola rete neurale può estendersi per diversi giorni. Nel contesto specifico del nostro lavoro, la quantità limitata di dati impiegati, pari a 6131 righe nel set di addestramento (con una dimensione del file di circa 483 kB), e la natura dei dati stessi - essendo testuali e privi di componenti multimediali come immagini - hanno rappresentato una significativa mitigazione di questa sfida. Tale configurazione, unitamente a una accurata scelta dei processi di ottimizzazione delle varie reti, ha permesso di completare con successo l'addestramento di tutte le reti neurali

nell'arco di una singola giornata.

5.2 Risultati dei modelli sul Test set

Di seguito si riportano i vari risultati ottenuti da ciascuno dei modelli attraverso l'utilizzo delle metriche precedentemente definite nella sezione delle Loss. Verranno riportati i risultati ottenuti da ciascun modello quindi in base al valore di MAE, MSE e RMSE. Prima di esporre i seguenti grafici però è necessaria una breve parentesi per comprendere al meglio i risultati ottenuti e a cosa essi corrispondo.

Quando si chiama *MinMaxScaler.fit(Y_train)*, ciò che avviene è il calcolo dei valori massimi e minimi basati sui dati presenti in Y_train. Successivamente, chiamando *transform()*, si trasformano tutte le caratteristiche sottraendo il minimo e dividendolo per la differenza tra massimo e minimo. Per comodità, queste due chiamate di funzione possono essere eseguite in un unico passaggio utilizzando *fit_transform()*. La ragione per cui si desidera addestrare lo scaler solo con i dati di addestramento è perché non si vuole influenzare il modello con informazioni provenienti dai dati di test. Se si eseguisse *fit()* sui dati di test, verrebbero calcolati valore massimo e valore minimo nuovamente. In teoria, questi valori potrebbero essere molto simili se i set di test e di addestramento avessero la stessa distribuzione, ma nella pratica questo non è sempre detto. Invece, si desidera trasformare solo i dati di test utilizzando i parametri calcolati sui dati di addestramento. In questo modo, si applicano le stesse trasformazioni a entrambi i set, garantendo coerenza nelle scale delle caratteristiche.
Di seguito si riportano i passaggi matematici che portano quindi alla valutazione del Mean Squared Error con i valori scalati all'interno del range [0,1] e come è possibile riportare questi valori nel range di partenza.

Sia definito con min il minimo valore della generica variabile y da predire, nel nostro caso l'intensità del traffico, e con max il massimo valore. Con n si indicano tutte le misurazioni della variabile y all'interno del training set.

$$min := \min_{\text{n}} y_n \tag{5.6}$$

$$max := \max_{\text{n}} y_n \tag{5.7}$$

Sia definito con y'_n il valore della variabile target scalato

$$y'_n = \frac{y - min}{max - min} \tag{5.8}$$

Sia definito invece definito con p'_n il valore predetto scalato

$$p'_n = \frac{p_n - min}{max - min} \tag{5.9}$$

Il valore di MSE scalato sarà quindi uguale a

$$\begin{aligned} mse_{scaled} &= \frac{1}{n}\sum_n |y'_n - p'_n|^2 \\ &= \frac{1}{n}\sum_n |\frac{y - min}{max - min} - \frac{p_n - min}{max - min}|^2 \\ &= \frac{1}{n}\sum_n |\frac{y - min}{max - min} - \frac{min}{max - min} - \frac{p_n - min}{max - min} + \frac{min}{max - min}|^2 \\ &= \frac{1}{n}\sum_n |\frac{y_n}{max - min} - \frac{p_n}{max - min}|^2 \\ &= \frac{1}{(max - min)^2}\frac{1}{n}\sum_n |y_n - p_n|^2 \end{aligned} \tag{5.10}$$

Una volta riscritta l'equazione del MSE con i valori scalati notiamo che l'ultimo termine, escludendo la frazione della differenza tra il massimo e il minimo valore al quadrato, è esattamente il valore di MSE senza valori scalati.

$$mse_{original} = \frac{1}{n}\sum_n |y_n - p_n|^2 \tag{5.11}$$

Quindi possiamo definire la formula per ricavare il valore di MSE non scalato nell'intervallo [0,1] a partire dal valore di MSE scalato.

$$mse_{original} = \frac{1}{n}\sum_n |y_n - p_n|^2 = (max - min)^2 mse_{scaled} \tag{5.12}$$

Applicando lo stesso ragionamento applichiamo la formula ottenuta per MAE e RMSE.

$$MAE_{original} = (max - min)MAE_{scaled} \tag{5.13}$$

$$RMSE_{original} = (max - min)RMSE_{scaled} \tag{5.14}$$

Nel nostro caso essendo il massimo valore del traffico presente pari a 100 e il valore minimo pari a 0, sarà neccessario moltiplicare i valori ottenuti per 100 (tale valore andrà elevato al quadrato nel caso di MSE) così da poter ritornare al numero di veicoli come unità di misura.

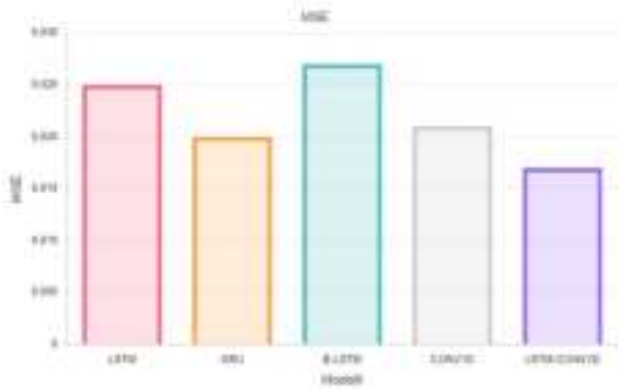
Chiusa questa parentesi riportiamo i grafici dei vari modelli per ciascuna delle metriche.

MSE Come si può vedere dalla Figura 5.3 il valore migliore di MSE è quello relativo all'ultimo modello implementato, mentre il valore peggiore di MSE è associato al modello Bidirectional LSTM, essendo quest'ultimo il modello più semplice implementato e che presentava più problematiche nelle curve di addestramento. Ricordiamo che MSE, avendo l'elevamento a potenza, perde l'unità di misura della variabile da predire, ma,

comparando i vari risultati, si può selezionare il modello ottimale in base a tale metrica considerando il valore minore.

MAE Dalla Figura 5.4 si evince che, come nel caso precedente, il valore migliore è associato all'ultimo modello. Inoltre, per tale misura è possibile convertire il valore ottenuto nel numero di macchine che, in questo caso, risulta essere pari a 10, che possono essere previste in più (o in meno) dal nostro modello. Il modello invece che presenta il peggior valore di MAE è anche in questo caso il modello bidirezionale, con un errore pari a 13 macchine. In termini di percentuali, ciò equivale a un errore di previsione del traffico stradale massimo del 13% e un errore minimo del 10%. Poiché MSE contiene il quadrato dei residui, in linea teorica dovrebbe avere un valore maggiore rispetto al MAE ma, essendo i valori della variabile target compresi nell'intervallo [0,1], avviene il contrario. Applicando le formule precedentemente ricavate si può notare come ciò venga garantito.

RMSE Questa ultima metrica rispecchia l'andamento delle precedenti ma, come precedentemente spiegato, attribuisce un peso maggiore a gli errori più grandi (come MSE). Per questa ragione, i valori ottenuti sono maggiori rispetto alla precedente misurazione che non ne teneva conto. Il modello migliore presenta un errore di previsione di 13 macchine, mentre il modello peggiore presenta un errore di predizione di 16 macchine. In termini di percentuali, ciò equivale a un errore di previsione del traffico stradale massimo del 16% e un errore minimo del 13%.
Nella Figura 5.5 vengono riportati i risultati di tutti e 5 i modelli.

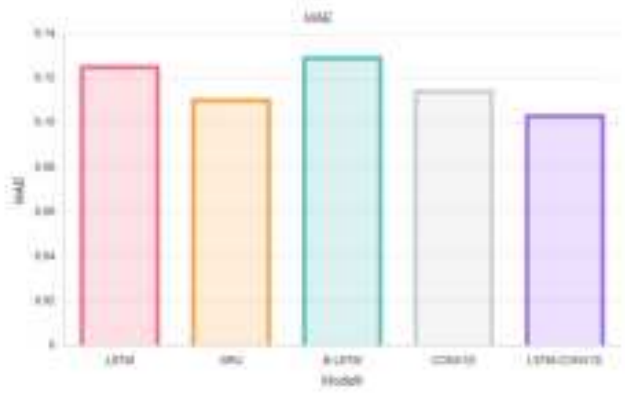


(a) Valori di MSE, nell'attività di predizione del traffico, dei cinque modelli messi a confronto

Modello	MSE
LSTM	0,025
GRU	0,020
Bidirectional-LSTM	0,027
Conv1d	0,021
LSTM-Conv1d	0,017

(b) Tabella riassuntiva dei valori di MSE di ciascuno dei cinque modelli

Figura 5.3: *Risultati di MSE per i cinque modelli.*

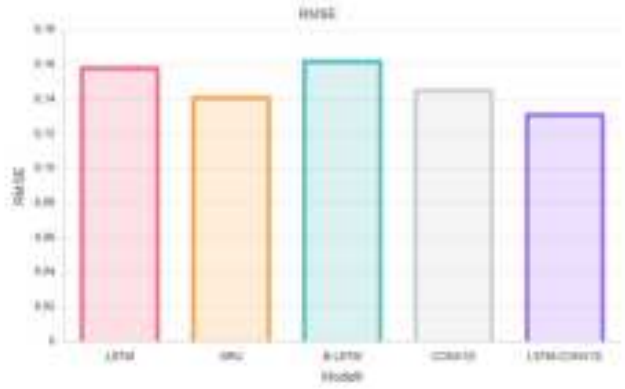


(a) Valori di MAE, nell'attività di predizione del traffico, dei cinque modelli messi a confronto

Modello	MAE
LSTM	0,126
GRU	0,111
Bidirectional-LSTM	0,130
Conv1d	0,115
LSTM-Conv1d	0,104

(b) Tabella riassuntiva dei valori di MAE di ciascuno dei cinque modelli

Figura 5.4: *Risultati di MAE per i cinque modelli.*



(a) Valori di RMSE, nell'attività di predizione del traffico, dei cinque modelli messi a confronto

Modello	RMSE
LSTM	0,159
GRU	0,142
Bidirectional-LSTM	0,163
Conv1d	0,146
LSTM-Conv1d	0,132

(b) Tabella riassuntiva dei valori di RMSE di ciascuno dei cinque modelli

Figura 5.5: *Risultati di RMSE per i cinque modelli.*

La Tabella 5.1 presenta un riepilogo dei valori di tutte le metriche per ciascun modello.

Tabella 5.1: *Valori di MSE, MAE e RMSE per ciascun modello*

MODELLO	MSE	MAE	RMSE
LSTM	0,025	0,126	0,159
GRU	0,020	0,111	0,142
Bidirectional-LSTM	0,027	0,130	0,163
Conv1d	0,021	0,115	0,146
LSTM-Conv1d	0,017	0,104	0,132

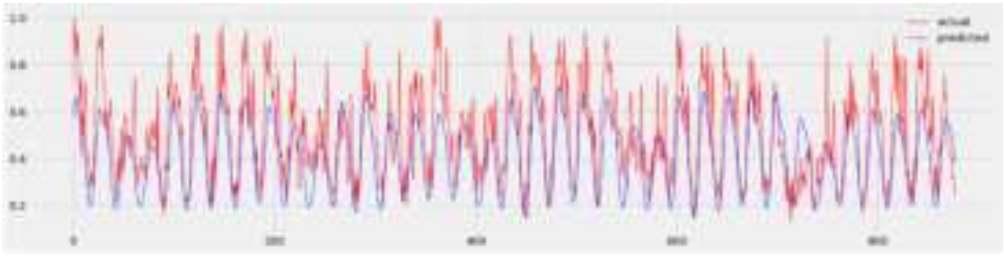
Oltre a vedere i singoli valori delle varie metriche ottenuti sul test set è importante vedere l'andamento delle curve predette rispetto alle curve reali.

Nel contesto della Figura 5.6, l'*andamento reale* del traffico è rappresentato in *rosso*, mentre l'*andamento predetto* dai modelli specifici è evidenziato in *blu*. I primi tre grafici illustrano le curve predette da ciascuno dei tre modelli di rete ricorrente (LSTM, GRU, Bidirectional LSTM), evidenziando che tali modelli riescono complessivamente a riprodurre accuratamente l'andamento delle curve, ad eccezione di alcune situazioni limite o picchi evidenziati nei grafici. Il modello LSTM-Bidirezionale, tuttavia, mostra una minore capacità nel replicare l'andamento delle curve, presentando valori predetti inferiori rispetto ai valori reali in alcune circostanze.

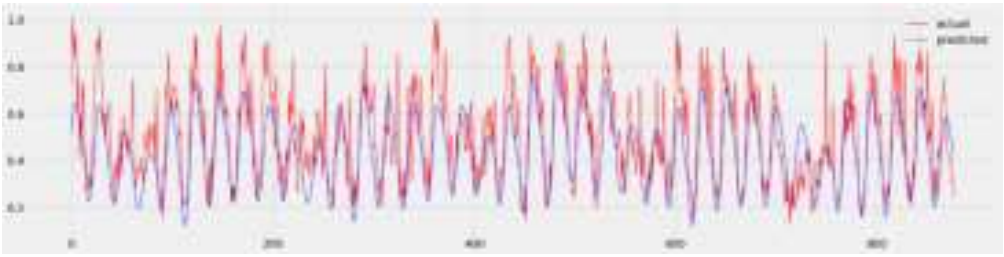
Le reti ricorrenti, come la GRU, dimostrano una buona capacità predittiva rispetto all'andamento reale delle curve, sebbene tali andamenti risultino essere caratterizzati da una certa "dolcezza", in quanto non riescono a catturare completamente le sottili irregolarità rispetto a una curva di traffico. Questa specificità viene meglio catturata dal modello di rete convoluzionale, il quale è in grado di riprodurre curve più irregolari rispetto ai modelli precedenti.

Complessivamente, il modello finale si distingue per la sua capacità di ricreare in modo ottimale l'andamento della curva reale del traffico, evidenziando un'efficace cattura anche dei valori di picco rispetto ai modelli precedenti.

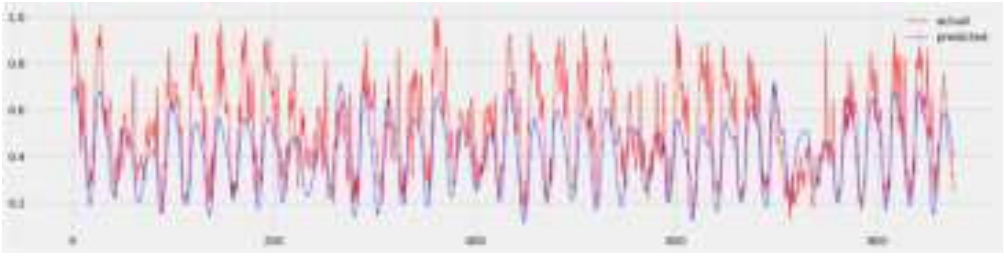
In generale, si noti che tutti e cinque i modelli, compresi LSTM, GRU, Bidirectional LSTM e i modelli basati su reti convoluzionali, mostrano una propensione a sottostimare l'intensità del traffico nei casi in cui si verificano errori, piuttosto che sovrastimarla. Tale tendenza è evidente nelle situazioni in cui i valori predetti risultano inferiori rispetto ai valori reali, sottolineando una propensione comune verso una stima più conservativa dell'intensità del traffico da parte di tutti i modelli implementati.



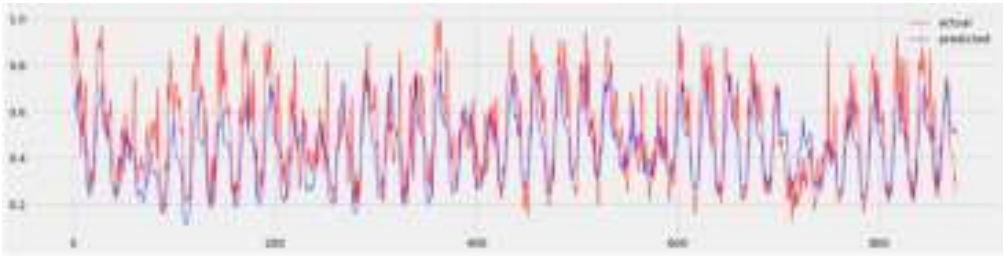
(a) *Valori predetti e reali del flusso del traffico stradale tramite LSTM*



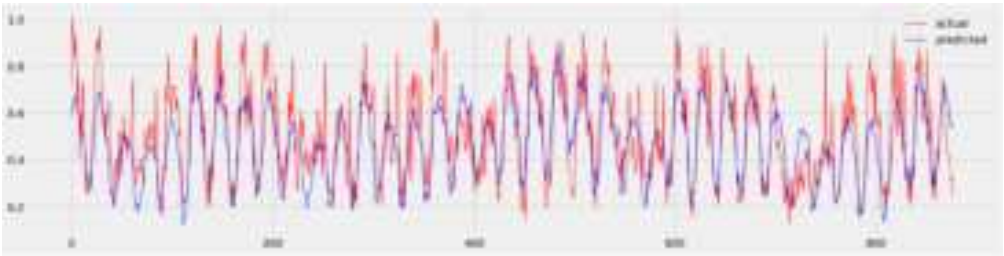
(b) *Valori predetti e reali del flusso del traffico stradale tramite GRU*



(c) *Valori predetti e reali del flusso del traffico stradale tramite Bidirectional LSTM*



(d) *Valori predetti e reali del flusso del traffico stradale tramite rete Convoluzionale*



(e) *Valori predetti e reali del flusso del traffico stradale tramite LSTM-Conv1d*

Figura 5.6: *Confronto tra i valori predetti e reali del traffico per i vari modelli.*

Capitolo 6

Conclusioni e possibili sviluppi

La previsione del flusso del traffico urbano rappresenta un elemento critico per ottimizzare la gestione degli spostamenti nelle grandi città. Questo lavoro di tesi ha proposto un approccio innovativo (poichè basato esclusivamente su dati di sensori *general-purpose*) per la previsione del traffico e della sua intensità, combinando i dati relativi agli inquinanti atmosferici e alle condizioni meteorologiche.

L'approccio proposto consente di ridurre significativamente la dipendenza da sensori specifici, utilizzabili solo in contesti limitati e caratterizzati da costi di manutenzione elevati. Ciò rappresenta un'opzione politicamente, socialmente ed ecologicamente più sostenibile rispetto alla distribuzione di sensori di traffico specializzati, spostando la complessità dalle infrastrutture hardware a quelle software. Inoltre, l' approccio proposto non solo consente di abbattere i costi di implementazione e gestione, ma anche di ampliare le funzioni e i servizi offerti, contribuendo così a dissipare le incertezze di alcuni cittadini legate al concetto di Smart City.

La crescente adozione dei modelli di deep learning nell'ambito della ricerca, specialmente per la previsione delle serie temporali, ha messo in luce la loro straordinaria potenza e versatilità nel gestire dati complessi. La capacità di catturare modelli intricati nei dati ha reso tali approcci ampiamente utilizzati, riflettendo la tendenza a impiegare reti neurali anche nel presente lavoro. All'interno di questa ricerca, la scelta di utilizzare modelli di deep learning è stata guidata dalla consapevolezza della loro efficacia nel trattare dati complessi, apprendere automaticamente caratteristiche rilevanti e adattarsi a una vasta gamma di problemi. In particolare, si è optato per l'implementazione di reti ricorrenti, note per la loro capacità di catturare dipendenze a lungo termine nei dati e di reti convoluzionali, che permettono di rilevare pattern complessi e strutture nascoste nelle serie temporali. Tuttavia, è fondamentale riconoscere che, nonostante i loro indiscutibili punti di forza, questi modelli presentano sfide significative, tra cui, la necessità di affrontare l'explainability e l'interpretabilità. Un'analisi approfondita dei risultati ottenuti dai cinque modelli esaminati fornisce spunti interessanti. Mentre le

reti neurali ricorrenti si confermano essenziali per la previsione del traffico, emergono delle difficoltà nell'identificare le irregolarità nelle curve effettive del flusso veicolare. Di contro, il modello di rete convoluzionale ha dimostrato maggiore efficacia nel catturare tali singolarità, offrendo una visione più dettagliata dell'andamento reale delle curve. Particolarmente rilevante è la combinazione di una rete ricorrente e una rete convoluzionale, che ha prodotto risultati superiori alle altre soluzioni. Questa sinergia tra le due architetture ha dimostrato di catturare in modo più accurato l'andamento delle curve del traffico, ottenendo nel contempo il valore più basso nell'errore di previsione rispetto al numero effettivo di macchine previste. Il risultato evidenzia la i benefici derivanti dalla combinazione di diverse architetture neurali nella progettazione di modelli di previsione del traffico più efficaci e precisi. Questo successo è attribuibile alla sinergia ottenuta mediante la fusione di due modelli, la quale consente di preservare e sfruttare i punti di forza distintivi di entrambe le architetture. Tale approccio integrato si rivela cruciale nel superare le limitazioni individuali delle architetture singole, contribuendo così alla realizzazione di modelli di previsione del traffico più completi e sofisticati. I risultati ottenuti nelle valutazioni delle metriche sono soddisfacenti, considerando la complessità e le molteplici variabili coinvolte nella previsione del numero di macchine passanti. È importante sottolineare che diversi fattori esterni possono influenzare i risultati, come le fonti di inquinamento atmosferico che impattano sulla rilevazione dei gas emessi dalle vetture. La presenza sempre più diffusa di veicoli elettrici, che non contribuiscono alle emissioni inquinanti, aggiunge ulteriore complessità alla previsione. Nonostante lo studio abbia cercato di ricreare condizioni ideali è inevitabile che fattori esterni, come l'inquinamento proveniente da fonti diverse e la presenza di veicoli elettrici, possano influire sulla precisione delle previsioni. L'introduzione di ulteriori informazioni, quali i dati sull'inquinamento acustico, potrebbe rappresentare un passo in avanti, offrendo un ulteriore parametro da correlare al traffico stradale. Tuttavia, anche in questo caso, sarebbe necessario considerare possibili influenze esterne che potrebbero impattare sulle valutazioni di tali informazioni.

In prospettiva futura, si potrebbero condurre studi a periodi di tempo più estesi, incorporando dati che coprono diversi anni al fine di individuare pattern non solo settimanali e giornalieri ma anche mensili. Esplorare la variazione dei risultati dei vari modelli considerando diverse tipologie di strade, sia a singola corsia che a più corsie, consentirebbe di comprendere le sfide specifiche di ciascun contesto e la qualità delle predizioni ottenute. Ulteriori ricerche potrebbero focalizzarsi sulla ricerca di fonti esterne per integrare l'inquinamento acustico e valutare in che misura i risultati variano rispetto a quelli ottenuti senza tale contributo. Un'analisi interessante potrebbe concentrarsi sull'effetto dell'incremento della finestra di predizione nel numero di ore sull'errore dei diversi modelli. Parallelamente, si potrebbe infine esplorare l'utilizzo di misurazioni satellitari sull'inquinamento atmosferico, valutandone l'integrazione con i dati da terra per migliorare l'identificazione delle correlazioni e fornire predizioni più accurate.

Bibliografia

[1] Pilar Rey del Castillo. *Analyzing traffic flows in Madrid city*. Disponibile online: https://cros-legacy.ec.europa.eu/content/s06p2-analizing-traffic-flows-madrid-city_en.

[2] Sharmila Majumdar et al. ≪Congestion prediction for smart sustainable cities using IoT and machine learning approaches≫. In: *Sustainable Cities and Society* 64 (2021), p. 102500. ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2020.102500. URL: https://www.sciencedirect.com/science/article/pii/S2210670720307198.

[3] Wei Wang et al. ≪An interpretable model for short term traffic flow prediction≫. In: *Mathematics and Computers in Simulation* 171 (2020). International Conference in Mathematics and Applications, held in Bangkok, Thailand, on December 16-18, 2018, pp. 264–278. ISSN: 0378-4754. DOI: https://doi.org/10.1016/j.matcom.2019.12.013. URL: https://www.sciencedirect.com/science/article/pii/S0378475419303726.

[4] Qinzhong Hou et al. ≪An adaptive hybrid model for short-term urban traffic flow prediction≫. In: *Physica A: Statistical Mechanics and its Applications* 527 (2019), p. 121065. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2019.121065. URL: https://www.sciencedirect.com/science/article/pii/S0378437119306508.

[5] Dongjie Zhu et al. ≪Research on Path Planning Model Based on Short-Term Traffic Flow Prediction in Intelligent Transportation System≫. In: *Sensors* 18.12 (dic. 2018), p. 4275. ISSN: 1424-8220. DOI: 10.3390/s18124275. URL: http://dx.doi.org/10.3390/s18124275.

[6] Roozbeh Ketabi et al. ≪Vehicular Traffic Density Forecasting through the Eyes of Traffic Cameras; a Spatio-Temporal Machine Learning Study≫. In: *Proceedings of the 9th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*. DIVANet '19. Miami Beach, FL, USA: Association for Computing Machinery, 2019, pp. 81–88. ISBN: 9781450369077. DOI: 10.1145/3345838.3356002. URL: https://doi.org/10.1145/3345838.3356002.

[7] Wangyang Wei, Honghai Wu e Huadong Ma. «An AutoEncoder and LSTM-Based Traffic Flow Prediction Method». In: *Sensors* 19.13 (lug. 2019), p. 2946. ISSN: 1424-8220. DOI: 10.3390/s19132946. URL: http://dx.doi.org/10.3390/s19132946.

[8] Nimra Shahid et al. «Towards greener smart cities and road traffic forecasting using air pollution data». In: *Sustainable Cities and Society* 72 (2021), p. 103062. ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2021.103062. URL: https://www.sciencedirect.com/science/article/pii/S2210670721003462.

[9] Ana Russo et al. «Neural network forecast of daily pollution concentration using optimal meteorological data at synoptic and local scales». In: *Atmospheric Pollution Research* 6.3 (2015), pp. 540–549. ISSN: 1309-1042. DOI: https://doi.org/10.5094/APR.2015.060. URL: https://www.sciencedirect.com/science/article/pii/S1309104215302245.

[10] Ibai Laña et al. «The role of local urban traffic and meteorological conditions in air pollution: A data-based case study in Madrid, Spain». In: *Atmospheric Environment* 145 (2016), pp. 424–438. ISSN: 1352-2310. DOI: https://doi.org/10.1016/j.atmosenv.2016.09.052. URL: https://www.sciencedirect.com/science/article/pii/S1352231016307683.

[11] Hai-Bang Ly et al. «Development of an AI Model to Measure Traffic Air Pollution from Multisensor and Weather Data». In: *Sensors* 19.22 (nov. 2019), p. 4941. ISSN: 1424-8220. DOI: 10.3390/s19224941. URL: http://dx.doi.org/10.3390/s19224941.

[12] Stuart Batterman, Rajiv Ganguly e Paul Harbin. «High Resolution Spatial and Temporal Mapping of Traffic-Related Air Pollutants». In: *International Journal of Environmental Research and Public Health* 12.4 (apr. 2015), pp. 3646–3666. ISSN: 1660-4601. DOI: 10.3390/ijerph120403646. URL: http://dx.doi.org/10.3390/ijerph120403646.

[13] Faraz Malik Awan, Roberto Minerva e Noel Crespi. «Improving Road Traffic Forecasting Using Air Pollution and Atmospheric Data: Experiments Based on LSTM Recurrent Neural Networks». In: *Sensors* 20.13 (lug. 2020), p. 3749. ISSN: 1424-8220. DOI: 10.3390/s20133749. URL: http://dx.doi.org/10.3390/s20133749.

[14] Faraz Malik Awan, Roberto Minerva e Noel Crespi. «Using Noise Pollution Data for Traffic Prediction in Smart Cities: Experiments Based on LSTM Recurrent Neural Networks». In: *IEEE Sensors Journal* 21.18 (2021), pp. 20722–20729. DOI: 10.1109/JSEN.2021.3100324.

[15] Sneh Jain, Roopam Gupta e Asmita A. Moghe. «Stock Price Prediction on Daily Stock Data using Deep Neural Networks». In: *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*. 2018, pp. 1–13. DOI: 10.1109/ICACAT.2018.8933791.

[16] Ioannis E. Livieris, Emmanuel Pintelas e Panagiotis Pintelas. ≪A CNN–LSTM Model for Gold Price Time-Series Forecasting≫. In: 32.23 (dic. 2020), pp. 17351–17360. ISSN: 0941-0643. DOI: 10.1007/s00521-020-04867-x. URL: https://doi.org/10.1007/s00521-020-04867-x.

[17] IBM. *What are recurrent neural networks?* https://www.ibm.com/topics/recurrent-neural-networks.

[18] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. arXiv: 1912.05911 [cs.LG].

[19] *Introduction to Recurrent Neural Network*. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/. 2023.

[20] *How the LSTM improves the RNN*. https://towardsdatascience.com/how-the-lstm-improves-the-rnn-1ef156b75121. 2021.

[21] Sepp Hochreiter e Jürgen Schmidhuber. ≪Long Short-Term Memory≫. In: *Neural Comput.* 9.8 (nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[22] *Long Short-Term Memory*. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/. 2023.

[23] *LSTM Vs GRU in Recurrent Neural Network: A Comparative Study*. https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/. 2021.

[24] *GRU Recurrent Neural Networks — A Smart Way to Predict Sequences in Python*. https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6. 2022.

[25] *What are convolutional neural networks?* https://www.ibm.com/topics/convolutional-neural-networks.

[26] *Convolutional Neural Networks, Explained*. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[27] *Understanding 1D and 3D Convolution Neural Network — Keras*. https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610.

[28] *4 Techniques to Handle Missing values in Time Series Data*. https://towardsdatascience.com/4-techniques-to-handle-missing-values-in-time-series-data-c3568589b5a8. 2022.

[29] Koredianto Usman e Mohammad Ramdhani. ≪Comparison of Classical Interpolation Methods and Compressive Sensing for Missing Data Reconstruction≫. In: *2019 IEEE International Conference on Signals and Systems (ICSigSys)*. 2019, pp. 29–33. DOI: 10.1109/ICSIGSYS.2019.8811057.

[30] *Correlazione Statistica*. https://www.yimp.it/correlazione-statistica/. 2022.

[31] *Pearson Correlation Coefficient (r) — Guide Examples*. https://www.scribbr.com/statistics/pearson-correlation-coefficient/. 2023.

[32] *Pearson's correlation coefficient: a beginner's guide*. https://datapeaker.com/en/big--data/Pearson's-Correlation-Coefficient-A-Beginner's-Guide/.

[33] *Spearman's rank correlation coefficient*. https://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient.

[34] *Coefficiente di correlazione per ranghi di Spearman*. https://it.wikipedia.org/wiki/Coefficiente_di_correlazione_per_ranghi_di_Spearman.

[35] *Time series forecasting*. https://www.tensorflow.org/tutorials/structured_data/time_series.

[36] *Multivariate Time Series Analysis With Python for Forecasting and Modeling (Updated 2023)*. https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/.

[37] *Multivariate Time Series Forecasting using RNN(LSTM)*. https://medium.com/mlearning-ai/multivariate-time-series-forecasting-using-rnn-lstm-8d840f3f9aa7.

[38] *Dropout in Neural Networks*. https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9.

[39] Jiaqi Pan, Yan Zhuang e Simon Fong. ≪The Impact of Data Normalization on Stock Market Prediction: Using SVM and Technical Indicators≫. In: vol. 652. Set. 2016, pp. 72–88. ISBN: 978-981-10-2776-5. DOI: 10.1007/978-981-10-2777-2_7.

[40] *Quali sono i diversi modi per valutare un modello di regressione lineare?* https://ichi.pro/it/quali-sono-i-diversi-modi-per-valutare-un-modello-di-regressione-lineare-209018250907954.

[41] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[42] B.T. Polyak. ≪Some methods of speeding up the convergence of iteration methods≫. In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(64)90137-5. URL: https://www.sciencedirect.com/science/article/pii/0041555364901375.

[43] *How to Checkpoint Deep Learning Models in Keras*. https://machinelearningmastery.com/check-point-deep-learning-models-keras/.

[44] *Use Early Stopping to Halt the Training of Neural Networks At the Right Time*. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/.