POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Matematica

Tesi di Laurea Magistrale

Ricerca di anomalie su sistemi ingegneristici attraverso metodi di intelligenza artificiale

Relatore:
Prof. Francesco Vaccarino
Correlatori:
Antonio Mastropietro
Alessandro De Gregorio

Candidato:
Andrea Gardetto

Marzo 2021

Sommario

In questa tesi è affrontato il problema dell'identificazione di anomalie su serie temporali multivariate attraverso modelli di Deep Learning.
Il caso d'uso consiste in un'analisi di dati provenienti da simulazioni di sensori applicati a sospensioni di autoveicoli sportivi. A causa del processo progressivo di deterioramento del materiale composito di cui sono prodotte le sospensioni, è necessario introdurre un sistema di monitoraggio delle condizioni strutturali che rilevi quando una o più sospensioni sono danneggiate. Tuttavia tra i metodi per controllare la salute di una struttura si può distinguere in test distruttivi, che deteriorano o distruggono pezzi della struttura da monitorare ricadendo quindi sui costi da sostenere, e test non distruttivi. L'obiettivo della tesi è trovare un test non distruttivo che ottenga buoni risultati riguardanti l'individuazione dei danni, ovvero che identifichi le anomalie senza produrre eccessivi falsi positivi.
Grazie alla capacità dei modelli di Deep Learning di ottenere ottime prestazioni anche per dati di grande dimensionalità e per dati che presentano strutture non-lineari difficili da catturare, è sempre più attuale la ricerca di reti neurali artificiali che svolgano questo compito. In particolare sono stati analizzati due modelli che, sebbene si basano sullo stesso principio di rilevare le anomalie attraverso una codifica e decodifica dei dati, presentano proprietà differenti.

Ringraziamenti

Vorrei ringraziare il Professor Francesco Vaccarino e i correlatori Antonio Mastropietro e Alessandro De Gregorio per la disponibilità dimostrata e per la presenza costante con cui mi hanno permesso di svolgere questo lavoro di tesi. Inoltre li ringrazio perché sono stati attenti a seguirmi, fornendomi stimoli, idee e consigli in maniera continuativa.

Ringrazio inoltre il centro interdipartimentale Smartdata@PoliTo per l'ospitalità, fornendomi la possibilità di svolgere il tirocinio e fare esperimenti utili alla tesi.

Grazie ai miei genitori per l'affetto smisurato, sostegno, stima e aiuto che hanno sempre rivolto nei miei confronti. Mi hanno sostenuto nei momenti in cui credevo di non farcela, mi hanno sempre aiutato cercando di togliermi pesi di dosso ma soprattutto per il bene che mi hanno sempre voluto.

Grazie ai miei familiari: mia zia, mio cugino, mia sorella e i miei nipoti, Armando, zio Massimo e zia Anna. Sebbene non siamo una grande famiglia, il loro affetto, interesse e supporto sono state spinte per ottenere questo risultato.

Un grazie enorme va a Laura per il bene che mi dimostra sempre, per i suoi consigli e la sua presenza continua che mi hanno permesso di andare avanti in alcuni momenti, trovando conforto costante in una persona speciale.

Ringrazio inoltre i miei amici. Soprattutto coloro che hanno creduto in me, coloro che mi hanno supportato, chi attraverso consigli e chi attraverso presenza e affetto.

Grazie al Maestro Aurelio che con la sua passione e dedizione mi ha fatto appassionare della Matematica.

Infine il grazie più grande va alla mia "Nonnina Bella" che avrebbe tanto voluto vedermi laureato in Ingegneria. Grazie perché è sempre stata presente fino all'ultimo momento, supportandomi e implicitamente spingendomi a far bene, oltre che per me, anche per lei e i miei cari. La ringrazio perché è stata più di una Nonna, più di una seconda madre e, sebbene sia poco, questa laurea la dedico alla mia cara Nonna Gina.

Indice

1 Introduzione 1

2 Machine Learning 5
- 2.1 Minimizzazione del rischio empirico . . . 7
- 2.2 Capacità, underfitting e overfitting . . . 8
- 2.3 Iperparametri e dataset di validazione . . . 9
- 2.4 Metodi di discesa del gradiente (GD) e discesa stocastica del gradiente (SGD) 9

3 Deep Learning 15
- 3.1 Introduzione . . . 15
- 3.2 Reti neurali feed-forward (MLP) . . . 17
- 3.3 Algoritmo di Backpropagation . . . 18
- 3.4 Reti convoluzionali . . . 20
  - 3.4.1 Operazione di convoluzione . . . 20
  - 3.4.2 Pooling . . . 23
- 3.5 Reti ricorrenti . . . 24
  - 3.5.1 Varianti reti ricorrenti . . . 25
  - 3.5.2 Reti ricorrenti bidirezionali . . . 27
  - 3.5.3 Backpropagation lungo il tempo (BTTP) . . . 27
  - 3.5.4 Problema dell'esplosione/scomparsa del gradiente . . . 28
  - 3.5.5 Reti Long short-term memory (LSTM) . . . 29
- 3.6 Reti autoencoder . . . 30
  - 3.6.1 Autoencoder incompleti e autoencoder regolarizzati . . . 31
  - 3.6.2 Applicazioni . . . 32
- 3.7 Modelli applicati . . . 33
  - 3.7.1 Primo modello: LSTM Encoder-Decoder . . . 33
  - 3.7.2 Secondo modello: MSCRED . . . 36

4 Anomaly detection 41
- 4.1 Panoramica sull'individuazione delle anomalie . . . 41
- 4.2 Monitoraggio delle condizioni strutturali (SHM) . . . 44
- 4.3 Caso d'uso . . . 46

5 Analisi dataset e risultati 47
5.1 Statistica descrittiva dei dataset 47
5.2 Metriche e grafici utili alla valutazione di una classificazione binaria 51
5.3 Risultati ottenuti 54
5.3.1 Risultati LSTM Encoder-Decoder 55
5.3.2 Risultati MSCRED 62
5.4 Confronto modelli 69

6 Conclusioni e sviluppi futuri 71

Elenco delle tabelle 75

Elenco delle figure 77

Capitolo 1

Introduzione

I materiali compositi sono una valida soluzione nel settore automobilistico in quanto permettono di produrre veicoli più leggeri e minori emissioni, senza tuttavia rinunciare alla robustezza e all'integrità strutturale.
Questi materiali però subiscono un processo di progressivo deterioramento, dovuto dall'impatto energetico di piccoli eventi o dal ciclo di fatica, che portano alla degradazione di moduli elastici e delle resistenze. Per evitare che il deterioramento causi un guasto imprevisto dei componenti strutturali, si ricorre a metodi economici per monitorare in maniera continuativa lo stato di salute del materiale composito. Siccome è nota la relazione tra quantità di danno e riduzione del modulo elastico, si può effettuare un monitoraggio rispetto alla degradazione della rigidezza per rilevare la posizione e la quantità di danno che si sono verificati.
Il procedimento che permette un controllo di questo tipo prende il nome di Monitoraggio delle condizioni strutturali (Structural Health Monitoring, SHM). In antichità si richiedeva agli ingegneri di ideare e realizzare un progetto tecnico che soddisfacesse le esigenze a cui era adibito, al minor costo possibile e che durasse nel tempo resistendo alle peggiori condizioni possibili. In precedenza la resistenza di una struttura veniva stimata attraverso una previsione e senza monitorarla continuamente. Questo ha portato numerosi fallimenti e alla necessità di introdurre sistemi di monitoraggio per mezzo dei quali l'ingegnere può rilevare con precisione la presenza di un danno [12].
Il sistema di monitoraggio si basa su cinque operazioni: monitoraggio del carico, identificazione del danno, individuazione della posizione in cui questo è avvenuto e delle relative dimensioni e gravità.
In questo lavoro di tesi si vuole risolvere il secondo punto precedentemente definito, ovvero l'identificazione del danno. Questa ricerca viene compiuta attraverso l'utilizzo di modelli di reti neurali artificiali. In particolare vengono analizzati dati provenienti da simulazioni di sensori a reticolo di Bragg in fibra che vengono posti su alcuni braccetti delle sospensioni di un veicolo. Le simulazioni vengono fatte per un veicolo con componenti compositi sani, in presenza di un'anomalia collocata in un punto del veicolo e presenza di un'altra anomalia situata in un'ulteriore posizione. Il tragitto percorso dal veicolo prevede la presenza di accelerazioni longitudinali e laterali di varia natura, registrando tramite i sensori le variazioni di lunghezza rispetto a quelle che si osservano a riposo.

Il lavoro svolto permette di ottenere un sistema di monitoraggio a basso costo che risulta essenziale per evitare guasti nei veicoli e per la sicurezza.
Sebbene il monitoraggio di materiali compositi sia un tema attuale ed esistano numerosi studi a riguardo, non è altrettanto cospicua l'analisi rivolta al settore automobilistico.
Nel lavoro svolto viene fatta un'assunzione tanto forte quanto ragionevole: misurazioni fatte su materiali compositi sani differiscono in qualche modo da quelle fatte su materiali danneggiati.
L'obiettivo del lavoro consiste nel cercare di ottenere un classificatore che identifichi un danno in maniera autonoma, ovvero una rete neurale che, ricevute in ingresso alcune misurazioni, stabilisca se l'oggetto analizzato sia danneggiato o meno. Questo scopo viene perseguito per mezzo di due modelli di Deep Learning. Queste reti neurali si propongono di identificare le anomalie, ciascuna catturando caratteristiche differenti dei dati.

Il lavoro di tesi è organizzato nel seguente modo.
Il *capitolo 2* consiste in una panoramica sul Machine Learning. Sono trattati gli ambiti in cui questa forma di statistica trova applicazione, i tipi di problema che si possono incontrare in funzione dei dati a disposizione e il problema di ottimizzazione che rende possibile la riuscita di questi modelli. Inoltre sono descritte alcune particolarità che accadono nella fase di addestramento di un algoritmo e da cui deriva direttamente la bontà del risultato finale. Infine sono riportati dei metodi di ottimizzazione utilizzati nel Machine Learning ma che trovano applicazione soprattutto nel Deep Learning, o direttamente o attraverso varianti di questi.

Nel *capitolo 3* si introducono le reti neurali artificiali e sono riportate le tipologie più comuni e utili alla risoluzione del caso d'uso. Inizialmente sono definite le reti neurali feed-forward, che aiutano ad avere una comprensione semplice dell'idea alla base dei modelli di Deep Learning, e l'algoritmo di Backpropagation, che permette alle reti di imparare dai dati attraverso l'aggiornamento dei pesi di cui è composta la rete stessa. In seguito vengono riportate le reti convoluzionali, definendo l'operazione matematica che viene utilizzata nei modelli e le funzioni di pooling che svolgono un ruolo essenziale introducendo invarianza locale, proprietà utile ad esempio nell'individuazione di oggetti nelle immagini che permette di riconoscere questi anche in posizioni differenti.
In questo capitolo inoltre vengono riportate le reti ricorrenti, ponendo attenzione al problema dell'esplosione/scomparsa del gradiente e introducendo quindi una variante che sopperisce alla difficoltà di addestrare i modelli ricorrenti semplici.
Una sezione è dedicata ad un tipo di rete neurale utile a catturare informazioni intrinseche nei, codificando e decodificando i record in modo tale che le ricostruzioni preservino solamente le strutture che si ripetono, eliminando quindi il rumore nei dati o caratteristiche che vengono osservate raramente dalla rete.
Infine in questo capitolo vengono illustrati i due modelli utilizzati: il primo consiste in un autoencoder che si serve di strati ricorrenti LSTM per catturare le dipendenze temporali. Il secondo modello invece è una rete autoencoder in cui vengono applicate operazioni di convoluzione e strati ricorrenti ConvLSTM, che differiscono dai precedenti LSTM in quanto catturano le strutture temporali ma al posto delle operazioni di moltiplicazione presentano operazioni di convoluzione, mantenendo quindi la quadridimensionalità. Tutti

e due i modelli utilizzati si fondano sulla medesima idea di addestrare una rete utilizzando solo una classe, quella sana, per poi riconoscere osservazioni anomale laddove la ricostruzione è meno precisa, supponendo quindi che dati dati anomali e sani presentino caratteristiche differenti. Quel che differenzia i modelli tuttavia è il metodo utilizzato per ricostruire. La prima rete infatti ricostruisce le serie temporali originarie mentre il secondo modello prima genera matrici che rappresentano le dipendenze tra le variabili, simili a covarianze, lungo finestre temporali differenti e poi impara a modellare il mutamento nel tempo delle relazioni tra le variabili [18].

Il *capitolo 4* fornisce una panoramica generale sul problema dell'individuazione di anomalie, illustrando gli ambiti in cui queste si presentano e le caratteristiche che influenzano la scelta del modello da utilizzare [3]. Successivamente è introdotto il Monitoraggio delle condizioni strutturali, che è un ambito in cui uno dei compiti equivale all'analisi fatta nel lavoro di tesi, ovvero l'identificazione di anomalie. Infine viene definito il caso d'uso, specificando che tipo di dati vengono utilizzati e come sono stati generati.

Il *capitolo 5* inizialmente presenta il dataset utilizzato, mostrando le statistiche riassuntive delle variabili suddividendo l'analisi per valori sani, anomali di tipo 1 e anomali di tipo 2. In seguito sono riportate anche delle matrici di correlazione, sempre separando ogni caso, che vista la difficoltà nel percepire differenze tra la classi dalla tabella con le statistiche, forniscono informazioni utili su cosa aspettarsi dall'analisi che verrà svolta.
Una sezione di questo capitolo è dedicata alle metriche utilizzate nella classificazione binaria, con particolare attenzione alla precisione, richiamo e all'introduzione dell'F_1 score. Questa misura è particolarmente importante perché permette di valutare contemporaneamente un compromesso tra le due metriche precedenti in modo tale che si trovino modelli che non accettino di avere buone prestazioni rispetto ad una misura e pessime rispetto all'altra, ma che cercano un via di mezzo.
Infine per ogni modello sono riportate le caratteristiche con cui questo è stato addestrato, validato e testato e successivamente sono comparate le prestazioni registrate sulle osservazioni di test.

Capitolo 2

Machine Learning

Il Machine Learning è una branca dell'intelligenza artificiale, "una forma di statistica applicata che pone maggiore enfasi sull'uso dei computer per stimare, per mezzo della statistica, funzioni complicate e una minore enfasi nel dimostrare gli intervalli di confidenza attorno a queste funzioni" [6].
L'obiettivo di un algoritmo di Machine Learning è imparare direttamente dai dati una funzione che soddisfi requisiti specifici. Di seguito sono riportati i problemi più frequenti per cui si ricorre a questi modelli:

- *regressione*: nel seguente problema lo scopo consiste nel predire un valore numerico per una variabile risposta y, assumendo che esista una relazione tra le variabili dell'insieme di dati che lo permetta. Le informazioni utili alla previsione possono essere tutte o solo alcune delle covariate x e si vuole trovare la funzione f che definisca questa relazione tra x e y.
  Sia quindi $x \in \mathbb{R}^m$, l'obiettivo di un modello di regressione è trovare f / y=f(x), con f : $\mathbb{R}^m \to \mathbb{R}$;

- *classificazione*: si parla di classificazione quando, partendo da valori delle covariate x disponibili per l'analisi, si vuole ottenere una variabile risposta y di tipo categorico, che dipenda dalle covariate x e possa essere trovata tramite una funzione f.
  Sia quindi $x \in \mathbb{R}^m$, l'obiettivo di un modello di regressione è trovare f / y=f(x), con f : $\mathbb{R}^m \to \{1, ..., k\}$.
  Il valore k si riferisce al numero di categorie, anche chiamate classi, a cui possono appartenere le osservazioni. Nel caso particolare in cui k=2 si parla di classificazione binaria, mentre quando k>2 si denomina come classificazione multi-classe;

- *clustering*: lo scopo di questi modelli è raggruppare elementi di un dataset in sottogruppi all'interno dei quali i dati sono più omogenei. E' necessario quindi definire misure di somiglianza, ovvero utilizzare metriche per il calcolo della distanza tra le osservazioni, tra cui ad esempio è molto diffusa la distanza Euclidea. In questi algoritmi, diversamente dai due precedenti, la variabile risposta y o non è presente oppure non viene utilizzata per la scelta del raggruppamento ma solo per verificare a posteriori la bontà del modello.

E' possibile catalogare i modelli di Machine Learning in relazione al modo in cui viene addestrato un algoritmo, ovvero in base alla tipologia di dati a disposizione e all'accessibilità di un feedback durante l'addestramento. Questi modelli si suddividono, a grandi linee in quanto ci sono ripartizioni più fini a seconda dei connotati nell'insieme di dati, in tre grandi categorie:

- *apprendimento supervisionato*: quando, oltre alle covariate x, è possibile addestrare l'algoritmo anche attraverso le variabili risposta y, chiamate *etichette*, che sono o le soluzioni desiderate o un aiuto all'addestramento;
- *apprendimento non-supervisionato*: quando non si è a disposizione delle etichette e questo implica la ricerca di proprietà intrinseche e relazioni tra le covariate x nei dati;
- *apprendimento per rinforzo*: molto differente dai due precedenti. Il modello in questione è chiamato agente e il suo scopo è di effettuare azioni che ottimizzino dei compiti prestabiliti, interagendo con un ambiente dinamico e quindi modificando le proprie azioni in funzione delle risposte ottenute da questo.

In questa tesi non verrà trattato l'apprendimento per rinforzo, non essendo pertinente al problema illustrato in seguito.

L'assunzione necessaria alla base di questa tipologia di algoritmi è che tutti i dataset utilizzati, ovvero quello usato per addestrare il modello, quello per validarlo e quello per testare i risultati ottenuti, si considerano generati da un'unica distribuzione di probabilità, chiamata *processo di generazione dei dati*. Assumendo inoltre che gli esempi nei dataset siano indipendenti gli uni dagli altri e che i dataset siano identicamente distribuiti, allora il processo di generazione dei dati può essere descritto attraverso una distribuzione di probabilità su un singolo esempio, chiamata *distribuzione di generazione dei dati* (p_{data}).

Grazie a queste assunzioni è ragionevole pensare di addestrare gli algoritmi per mezzo di un dataset e pensare che l'algoritmo ottenga buoni risultati su altri insiemi di dati, sebbene questi non siano stati presi in considerazione durante l'addestramento. Tale abilità del modello è chiamata *generalizzazione* ed è l'obiettivo principale del Machine Learning.

2.1 Minimizzazione del rischio empirico

L'allenamento del modello consiste nella definizione e risoluzione, attraverso una o più iterazioni lungo il dataset apposito, di un problema di ottimizzazione, che come tale può essere sempre riformulato come un equivalente problema di minimizzazione di una funzione obiettivo. Si consideri il caso supervisionato non regolarizzato (ma si può estendere facilmente lo sviluppo al caso non-supervisionato o regolarizzato).
L'obiettivo del Machine Learning è trovare una soluzione al seguente problema di ottimizzazione:

$$\min_{\boldsymbol{\theta}} J^*(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) \tag{2.1}$$

dove $L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$ è la funzione di perdita e $J^*(\boldsymbol{\theta})$ viene definito *rischio*, o equivalentemente detto errore di generalizzazione. Siccome non è possibile essere in possesso di (x,y) provenienti dalla distribuzione di generazione dei dati p_{data} , in quanto si è in possesso solo di una porzione di questi provenienti da una distribuzione empirica $\hat{p}_{\text{data}}$ che è il dataset di addestramento, al posto di ottimizzare l'equazione (2.1) si ottimizza la funzione obiettivo:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\mathbf{y})\sim \hat{p}_{\text{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) = \min_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right), y^{(i)}\right) \tag{2.2}$$

dove m è la cardinalità del dataset, che corrisponde alla *minimizzazione del rischio empirico*, o equivalentemente detto errore sul dataset di addestramento. Se valgono le assunzioni fatte precedentemente e sotto ulteriori condizioni, la minimizzazione del rischio empirico implica anche quella del rischio. Tuttavia raramente si minimizza la funzione obiettivo dell'equazione 2.2, preferendogli spesso funzioni alternative. Questo è dovuto a due motivi: il primo è che la minimizzazione del rischio empirico porta al fenomeno dell'overfitting per modelli con un'elevata capacità (sezione 2.2) e il secondo è che alcune funzioni di perdita risultano difficili da ottimizzare attraverso il metodo della discesa del gradiente o le sue varianti 2.4.
E' necessario puntualizzare che, a causa dell'impossibilità di conoscere la vera distribuzione generatrice dei dati, l'obiettivo degli algoritmi di Machine Learning differisce da quello dell'ottimizzazione in quanto in quest'ultimo ambito lo scopo trova compimento nella minimizzazione in sé mentre per i primi modelli l'ottimizzazione è solo un tramite.

2.2 Capacità, underfitting e overfitting

In un modello di Machine Learning viene definita *capacità* l'abilità del medesimo ad adattarsi ai dati di addestramento, ovvero a diminuire l'errore su quel dataset.
Possono verificarsi però due fenomeni indesiderati durante l'allenamento di un modello: *underfitting* e *overfitting*.
Il primo avviene quando l'algoritmo non riesce ad imparare le caratteristiche dei dati, non riducendo in maniera soddisfacente l'errore su quel dataset, dovuto ad una scarsa capacità del modello.
Il comportamento opposto si presenta quando l'algoritmo possiede una elevata capacità tanto da memorizzare quasi perfettamente i dati, cosa non desiderata in quanto l'obiettivo dell'addestramento è la diminuzione dell'errore di generalizzazione.

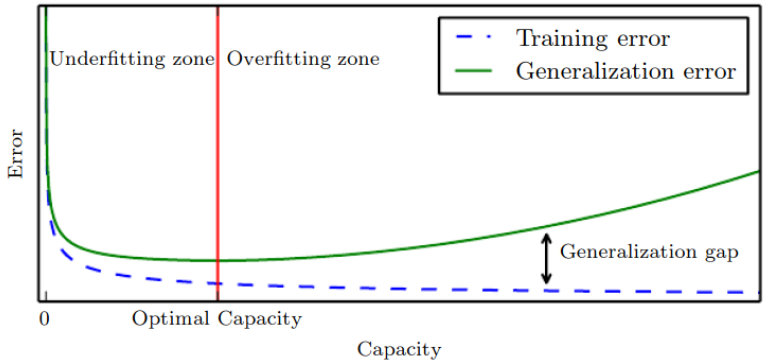

Figura 2.1: Relazione tra capacità di un modello, overfitting e underfitting [6]

Dalla Figura 2.1 si può notare come la capacità di un modello aumenta, sotto un opportuno addestramento e scelta degli iperparametri (sezione 2.3), all'aumentare delle iterazioni. Si passa infatti una bassa capacità (cioè alto errore sul dataset di addestramento) durante le prime ripetizioni, in quanto il modello non ha avuto abbastanza tempo per imparare le caratteristiche dei dati, ad una maggiore all'aumentare di queste. Dopo molte iterazioni si verifica quindi il fenomeno dell'overfitting, del quale si può osservare l'inizio in concomitanza con l'aumentare della differenza tra l'errore di generalizzazione e l'errore sul dataset di addestramento.
Il metodo principale utilizzato per evitare il problema dell'overfitting risulta salvare come modello ottimale quello che presenta l'errore di generalizzazione inferiore, tecnica anche chiamata *early-stopping*.

2.3 Iperparametri e dataset di validazione

Un ruolo fondamentale in un modello di Machine Learning lo ricoprono gli iperparametri. Questi sono valori che non vengono addestrati o modificati durante l'addestramento.
Per poter scegliere i miglior iperparametri spesso bisogna provare varie combinazioni e salvare il modello che presenta l'errore più basso su un apposito dataset, ovvero quello di validazione.
Le combinazioni di iperparametri possono essere ottenute in diversi modi. Il più tradizionale è la ricerca esaustiva, anche noto come grid search, ovvero la sperimentazione di tutte le possibili combinazioni.

k-fold cross-validation

La k-fold cross validation consiste in una tecnica statistica utilizzata per calcolare l'errore di validazione e quindi utile anche a trovare i migliori iperparametri di un modello, tecnica alternativa, più robusta ma anche più costosa computazionalmente rispetto all'utilizzo di un apposito dataset di validazione.
Il metodo consiste nel suddividere il dataset di addestramento in k sottoinsiemi di uguale numerosità, a meno di arrotondamenti, e di addestrare il modello k volte, pescando senza rimpiazzo un sottoinsieme ogni volta e utilizzandolo come dataset di validazione, mentre i restanti k-1 vengono considerati come dataset di addestramento. Si calcolano per tutti e k i modelli gli errori di validazione e infine si fa una media campionaria, la quale sarà considerata l'errore sul dataset di validazione per quel modello.
Questa tecnica è utile a prevenire l'overfitting e il problema del campionamento asimmetrico del campione osservato che si presenta quando si suddivide un dataset in più parti e si allena il modello considerandone solo una. Può accadere infatti che non tutte le informazioni contenenti nel dataset di validazione siano presenti in quello di addestramento, verificando così una distorsione nel modello.

2.4 Metodi di discesa del gradiente (GD) e discesa stocastica del gradiente (SGD)

Discesa del gradiente

Come visto nella sezione 2.2, lo scopo di un algoritmo di Machine Learning è quello di minimizzare l'errore di generalizzazione, ovvero trovare i parametri θ tali che

$$\theta = \underset{\theta}{argmin}\ \frac{1}{m}\sum_{i=1}^{m} L\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right), y^{(i)}\right)$$

In questa sezione è utile introdurre una notazione alternativa e più sintetica rispetto a quella utilizzata finora, per semplicità e coerenza con [6].
Siano quindi:

- $x = \theta$
- $f(x) = \frac{1}{m}\sum_{i=1}^{m} L\left(f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right), y^{(i)}\right)$

Si consideri prima il caso monodimensionale, che aiuta ad avere una comprensione più immediata, e poi il caso multidimensionale in cui si formalizzano i risultati:

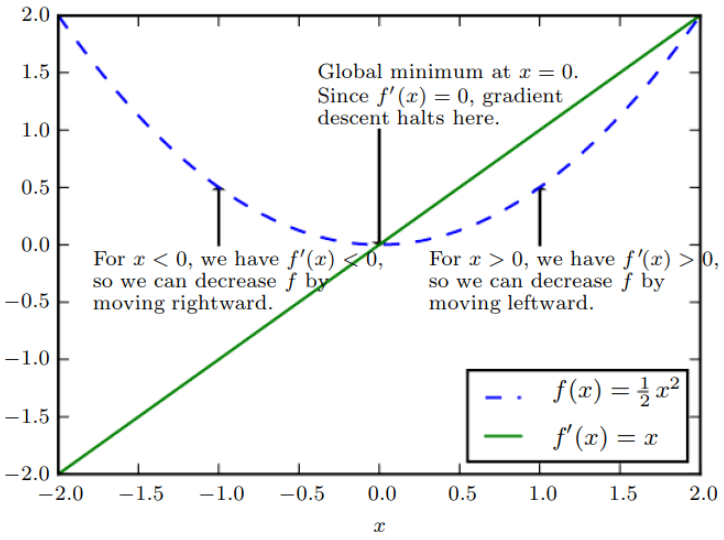

Figura 2.2: Rapporto derivata e minimo di una funzione [6]

Sia $x \in \mathbb{R}$, $f(x) \in \mathbb{R}$ e $f'(x) \in \mathbb{R}$ la derivata di f, allora la funzione f(x) raggiunge il suo valore minimo quando f'(x)=0. Si nota inoltre che

- per x<0 , f'(x)<0
- per x>0 , f'(x)>0

Questo esempio illustra come muovendosi nella direzione opposta della derivata ci si avvicini al minimo della funzione. Definendo come ϵ il tasso di apprendimento, di cui sarà discusso a breve, si può riportare la formula iterativa che permette la minimizzazione di f(x):

$$x' = x - \epsilon \cdot f'(x)$$

dove ad ogni passo si aggiorna il valore x con x'. Questa tecnica di minimizzazione prende il nome di *metodo del gradiente discendente* o *metodo della discesa più rapida.*

Ora si può generalizzare al caso multidimensionale, dove $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ e $f'(x) \in \mathbb{R}$, e fornire una giustificazione formale rispetto all'intuizione di notare una decrescita della funzione nella direzione opposta alla derivata.

Sia $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ il gradiente di f(x), ovvero un vettore con elemento i-esimo $(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}))_i = \frac{\partial f(x)}{\partial x_i}$, allora per trovare la direzione di maggior decrescita u bisogna risolvere

il seguente problema:

$$\min_{\boldsymbol{u},\ \boldsymbol{u}^\top \boldsymbol{u}=1} \boldsymbol{u}^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{u},\ \boldsymbol{u}^\top \boldsymbol{u}=1} \|\boldsymbol{u}\|_2 \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2 \cos\beta$$

e siccome u non dipende da β si può riscrivere come

$$\min_{\boldsymbol{u},\ \boldsymbol{u}^\top \boldsymbol{u}=1} \|\boldsymbol{u}\|_2 \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2$$

la cui soluzione risulta essere $u = -\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$.
Quindi il metodo del gradiente discendente si può scrivere come:

$$x' = x - \epsilon \cdot \nabla_x f(x)$$

Tasso di apprendimento

In un algoritmo di Machine Learning un ruolo fondamentale lo ricopre la scelta dell'iperparametro *tasso di apprendimento*, che consiste nella lunghezza del passo che si considera ad ogni iterazione mentre ci si sposta verso un minimo di una funzione di perdita.
Per capire le problematiche risultanti da una scelta errata di questo iperparametro risulta conveniente considerare una funzione di perdita simile ad una parabola. Sono riportate di seguito due figure raffiguranti l'impatto del tasso di apprendimento sulla minimizzazione dell'errore di test. E' utile rappresentarle tutte e due dato che, rispettivamente, il primo aiuta a capire il problema da un punto di vista più geometrico mentre il secondo aiuta a verificare, a posteriori di un addestramento, se la scelta del tasso di apprendimento è stata corretta o sbagliata e in che modo aggiustarla, mostrando come varia l'errore al crescere delle iterazioni. In tutti e due i casi si vede che un tasso di apprendimento troppo basso implica una convergenza al minimo troppo lenta e uno troppo alto può portare addirittura ad una crescita incontrollata dell'errore.

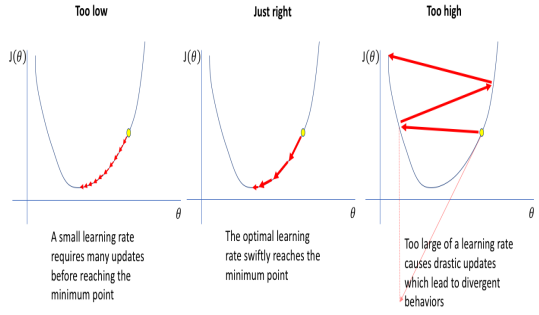

Figura 2.3: Impatto del tasso di apprendimento sulla funzione di perdita

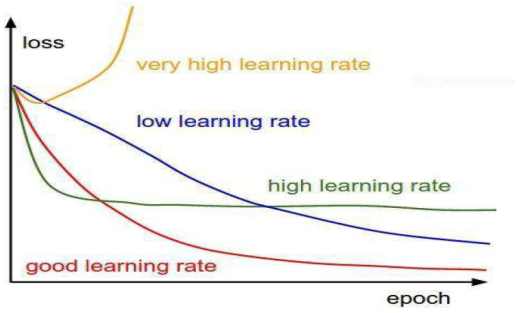

Figura 2.4: Andamento dell'errore di addestramento al variare del tasso di apprendimento

Discesa stocastica del gradiente

La *discesa stocastica del gradiente* è una variante del metodo descritto precedentemente. E' importante in quanto è alla base di quasi tutti i modelli di Deep Learning.
"Un problema ricorrente nel Machine Learning è che per ottenere una buona generalizzazione sono necessari grandi dataset di addestramento, ma grandi dataset sono anche più costosi computazionalmente"[6].
Infatti per come è definita la funzione di perdita, ovvero

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{\mathrm{p}}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right) \tag{2.3}$$

il metodo del gradiente discendente richiederebbe di calcolare

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right) \tag{2.4}$$

il cui costo computazionale è di $O(m)$. Se si pensa che vengono fatte così tante operazioni per un solo aggiornamento dei pesi e che per minimizzare la funzione sono necessari molteplici aggiornamenti, allora si intuisce come per dataset molto ampi diventa difficile arrivare a convergenza in tempi ragionevoli.
Considerando che il gradiente di equazione 2.4 è un valore atteso, risulta possibile quindi approssimarlo utilizzando solo una parte degli m campioni, ovvero un numero m'<m presi dal dataset iniziale in maniera random. Gli m' campioni utilizzati per un singolo aggiornamento dei pesi formano un sottoinsieme $\mathcal{B} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m')}\right\}$ chiamato *minibatch* e il gradiente viene approssimato con:

$$\boldsymbol{g} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right) \tag{2.5}$$

La formula della discesa stocastica del gradiente risulta quindi

$$\theta' = \theta - \epsilon \cdot g \tag{2.6}$$

Problema con minimi locali e punti di sella

Per come sono definiti gli algoritmi descritti precedentemente per la minimizzazione della funzione di rischio empirico, che utilizzano la derivata del gradiente della funzione di perdita per raggiungere la soluzione ottimale, si riscontrano problemi con i minimi locali e i punti di sella, che spesso sono presenti in funzioni non convesse. Infatti in tutti e due i casi si tratta di *punti stazionari*, ovvero $x^\star \ / \ \nabla f(x^\star) = 0$.
Quindi accade spesso che un algoritmo di ottimizzazione smetta di progredire verso la soluzione ottima, ovvero il minimo globale $x^\star \ / \ f(x^\star) < f(x) \ \forall x \in Dom(f)$, una volta arrivato su un minimo locale che però non è anche globale o in un punto di sella. Come mostrato in Figura 2.6, anche se non si raggiunge il minimo globale della funzione, alcuni minimi locali possono essere accettabili in quanto il gap con il valore ottimale è basso.

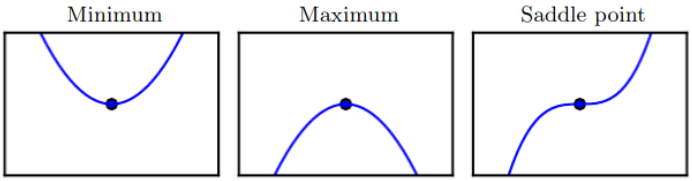

Figura 2.5: Punto di minimo, punto di massimo e punto di sella [6]

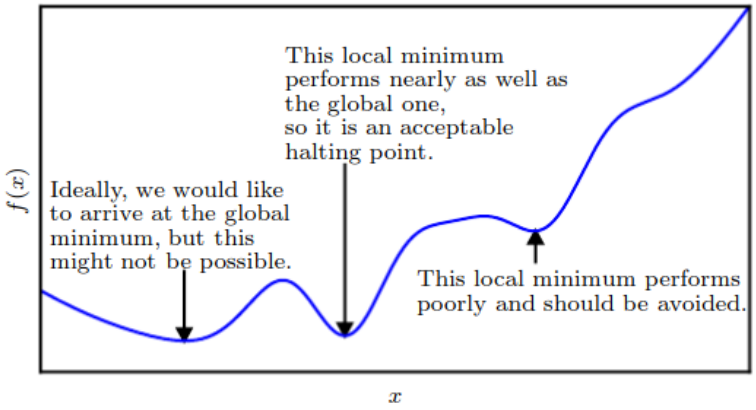

Figura 2.6: Minimi locali e globali di una funzione [6]

Capitolo 3

Deep Learning

3.1 Introduzione

"Il Deep Learning è un tipo specifico di Machine Learning"[6].
Lo sviluppo di questa categoria è giustificato dagli ostacoli riscontrati con i più semplici algoritmi di Machine Learning nella risoluzione dei problemi centrali dell'intelligenza artificiale, quali ad esempio il riconoscimento vocale e di oggetti. Gli ostacoli principali derivano da:

- problemi dei semplici algoritmi di apprendimento automatico ad imparare funzioni in spazi di grande dimensionalità. Infatti al crescere della quantità di variabili, il numero di possibili configurazioni distinte di un insieme di variabili aumenta in maniera esponenziale. Questo fenomeno prende il nome di "Maledizione della dimensionalità";

- convergenza, all'aumentare della cardinalità del dataset, a valori inferiori di accuratezza dei modelli di Machine Learning rispetto a quelli di Deep Learning

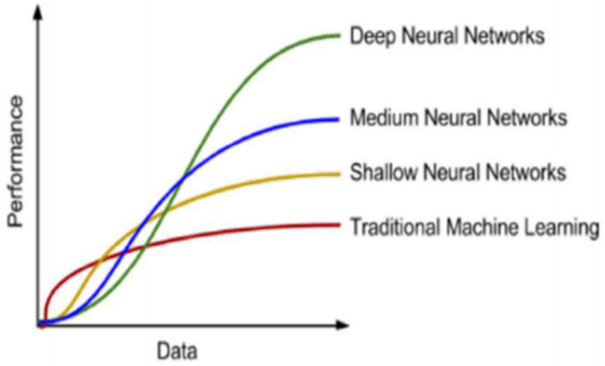

Figura 3.1: Confronto prestazioni tra differenti tipi di algoritmi di Deep Learning e Machine Learning al variare della dimensione dei dati [10]

I modelli di Deep Learning consistono in reti neurali artificiali profonde.
Con *reti neurali artificiali* ci si riferisce a funzioni che mappano un input x in un output $\hat{y} = f(x;\theta)$, dove θ sono i parametri della rete che vengono addestrati con lo scopo di raggiungere una buona approssimazione di f^*, funzione ideale tale per cui $y = f^*(x)$. Si definiscono in questa maniera per la somiglianza nella rappresentazione grafica alle reti neurali biologiche. Queste funzioni infatti si possono raffigurare come dei grafi composti da nodi, chiamati neuroni, dove vengono effettuate le operazioni, e da archi che indirizzano l'output di un nodo verso un altro nodo. Inoltre il grafo viene suddiviso in strati, ovvero blocchi di neuroni. Esempi di strati sono quello di input , dove ad ogni nodo è associato un valore in ingresso alla rete, gli strati intermedi che vengono anche chiamati nascosti, e infine il risultato uscente dalla rete, che può essere di dimensione arbitraria.
Il termine *profonda* invece è dovuto all'utilizzo nella rete di molti strati .
La particolarità di questi modelli è che, sebbene utilizzino prevalentemente semplici operazioni, la loro ripetizione e la presenza di funzioni di attivazione, che introducono non linearità e stabilità numerica, permette alle reti neurali profonde di raggiungere ottimi risultati, spesso impossibili da ottenere attraverso funzioni note.

Esempi di *funzioni di attivazione* sono:

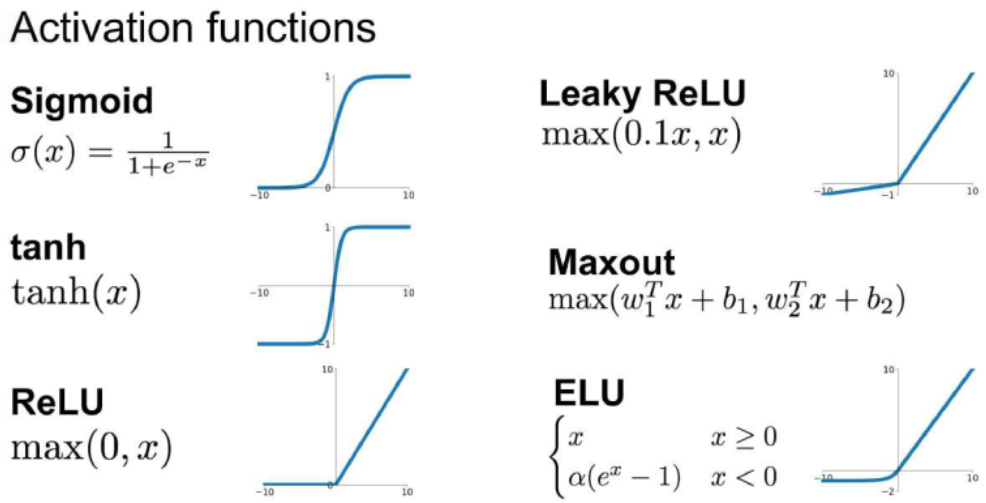

Figura 3.2: Esempi di funzioni di attivazione

3.2 Reti neurali feed-forward (MLP)

Le reti neurali feed-forward sono il tipo di reti più semplici che si possano costruire. Il loro nome deriva dal fatto che l'informazione nella rete scorre sempre in avanti, non creando quindi cicli fra le connessioni tra gli strati.
Questo tipo di rete è molto utilizzato soprattutto nei problemi di classificazione. Spesso però non viene implementata una rete feed-forward profonda, sia per i problemi di sovradattamento ai dati di addestramento di cui potrebbe soffrire il modello sia perché talvolta è preferibile utilizzare alcuni strati feed-forward pienamente connessi solo in seguito ad altri tipi. Ad esempio nella classificazione di immagini generalmente si usano prima strati convoluzionali, che riducono le dimensioni e catturano solo l'informazione rilevanti delle immagini a diverse scale, e poi si usano strati feed-forward per ottenere una classificazione dell'immagine, che non si sarebbe potuta ottenere per mezzo delle sole reti feed-forward profonde.

Osservazione 3.2.1. *In una rete neurale, uno* ***strato*** *si dice* ***pienamente connesso*** *se ogni nodo di quello strato è connesso con ogni nodo dello strato successivo.*

Osservazione 3.2.2. *Una* ***rete*** *si dice* ***pienamente connessa*** *se è composta solamente da strati pienamente connessi.*

In figura 3.3 è illustrato un esempio di rete feed-forward pienamente connessa con un solo strato nascosto. Questo modello riceve in input l'n-esima osservazione del dataset x_n, che è un vettore D-dimensionale, e attraverso delle operazioni di moltiplicazione e addizione , seguite da dall'applicazione di funzioni di attivazione, utilizzando per ogni nodo nascosto tutti i nodi in input calcola i valori z_n, ovvero i neuroni H-dimensionali appartenenti allo strato nascosto. Infine l'informazione tramite operazioni analoghe a prima arriva ai nodi in output, vettore C-dimensionale, utilizzando tutti i neuroni nascosti.

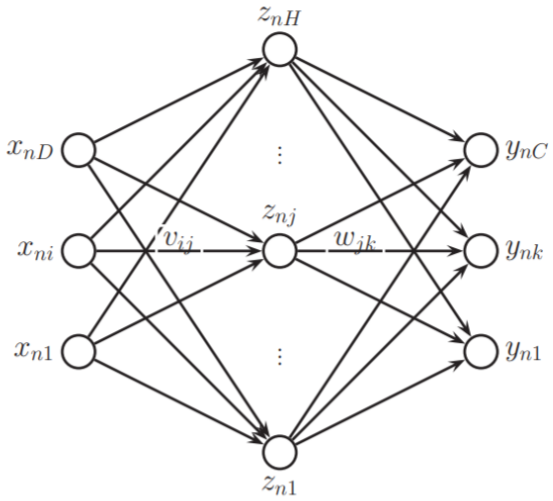

Figura 3.3: Esempio di rete feed forward, pienamente connessa e con uno strato nascosto [13]

3.3 Algoritmo di Backpropagation

Come visto precedentemente 2.4, l'apprendimento delle reti neurali profonde, e quindi dei parametri che intervengono nelle funzioni interne alla rete neurale, richiede che vengano calcolati i gradienti della funzione di perdita rispetto ai parametri, per poi aggiornarli e man mano cercare di avvicinarsi alla funzione ideale, ovvero quella che minimizza l'errore di generalizzazione.
Data la difficoltà nel calcolare i gradienti della funzione di perdita rispetto ai parametri, si può vedere una rete neurale come una composizione di funzioni

$$f = f^{(1)} \circ f^{(2)} \circ \ldots \circ f^{(L-1)}$$

dove $f^{(i)} = g^{(i,j)}(z^{(i,j)}())$, con $g^{(i,j)}$ funzione di attivazione del j-esimo strato nascosto e i-esimo nodo in quello strato. $z^{(i,j)}$ invece è una trasformazione, di solito di tipo affine $z = W \cdot x + b$, e quindi risulta più semplice ed efficiente, in quanto si risparmiano calcoli, l'utilizzo ricorsivo della *regola della catena* per il calcolo dei gradienti.

Definizione 3.3.1. *(Regola della catena - caso scalare) Siano $x \in \mathbb{R}$, f,g funzioni : $\mathbb{R} \to \mathbb{R}$ differenziabili e tali che y=g(x) e che z=f(y)=f(g(x)), allora per la* ***regola della catena*** *vale che*

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

La regola della catena si può generalizzare anche oltre al caso scalare:

Definizione 3.3.2. *(Regola della catena - caso multidimensionale) Siano $x \in \mathbb{R}^m$, f,g funzioni : $\mathbb{R}^m \to \mathbb{R}^n$ differenziabili e tali che y=g(x) e che z=f(y)=f(g(x)), allora per la* ***regola della catena*** *vale che*

$$\frac{\partial z}{\partial x_i} = \sum_j \left(\frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i} \right)$$

che può essere riscritto anche in notazione vettoriale come:

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \right)^\top \cdot \nabla_{\boldsymbol{y}} z = J^\top \cdot \nabla_{\boldsymbol{y}} z$$

dove J è la matrice Jacobiana di dimensioni (n x m) della mappa g.

L'algoritmo di *Backpropagation* consiste nell'applicazione ricorsiva della regola della catena lungo la rete, dove ogni tensore può essere convertito in un vettore, attraverso una concatenazione, per poi essere riconvertito in un tensore dopo l'applicazione della backpropagation.

3.4 Reti convoluzionali

"Le reti neurali convoluzionali sono un tipo specifico di reti applicate a dati che presentano una topologia nota, simile a una griglia. Esempi di questi dati sono le serie temporali, che possono essere pensate come un griglia monodimensionale che prende valori a istanti di tempo regolari, e le immagini, che possono essere considerate come una griglia bidimensionale di pixel"[6].

3.4.1 Operazione di convoluzione

Di seguito sono riportate le definizioni degli operatori di convoluzione, anche detti *filtri*, ognuna per un diverso tipo di valore x di input:

Definizione 3.4.1 (Convoluzione per funzioni)**.** *Siano x,k due funzioni. La* ***convoluzione*** *di x e k è definita come*

$$(Kx)(t) = (k * x)(t) = \int_{-\infty}^{\infty} x(\tau)k(t-\tau)d\tau = \int_{-\infty}^{\infty} x(t-\tau)k(\tau)d\tau$$

dove K è chiamato operatore di convoluzione associato a k [17].

Siccome di solito si analizzano i dati sotto forma di successioni, necessario introdurre la stessa formula ma valida nel caso discreto:

Definizione 3.4.2 (Convoluzione per successioni)**.** *Siano x,k due successioni. La* ***convoluzione*** *di x e k è definita come*

$$(Kx)_n = (k * x)_n = \sum_{l\in\mathbb{Z}} x_l k_{n-l} = \sum_{l\in\mathbb{Z}} k_l x_{n-l}$$

dove K è chiamato operatore di convoluzione associato a k [17].

Infine siccome l'interesse è rivolta ad un'analisi di dati di dimensione finita, per poter fornire la definizione di convoluzione per successioni di lunghezza finita, è necessario formalizzare una cosa.

Osservazione 3.4.1. *(Estensione successione di lunghezza finita) Sia x una successione di lunghezza finita con dominio*

$$n \in \{0,1,\ldots,N-1\}$$

per un N intero positivo. Una sequenza di lunghezza finita può essere vista o come una successione i cui elementi non nulli possono essere solo quelli per $n \in \{0,1,\ldots,N-1\}$ *oppure come un periodo di una sequenza periodica con*

$$x_{n+cN} = x_n, \quad c \in \mathbb{Z}$$

Definizione 3.4.3 (Convoluzione circolare)**.** *Siano x,k due successioni di lunghezza N. La* ***convoluzione circolare*** *di x e k è definita come*

$$(Kx)_n = (k * x)_n = \sum_{l=0}^{N-1} x_l k_{(n-l) mod N} = \sum_{l=0}^{N-1} k_l x_{(n-l) mod N}$$

dove K è chiamato operatore di convoluzione circolare associato a k [17].

E' utile necessario introdurre la definizione di convoluzione per segnali bidimensionali, ovvero immagini I, dove I(i,j) è il pixel in posizione (i,j):

Definizione 3.4.4 (Convoluzione bidimensionale)**.** *Siano I,K rispettivamente un segnale e un filtro bidimensionali. La* ***convoluzione bidimensionale*** *S di I e K è definita come*

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n)$$

Da tutte le definizioni si può notare come l'operatore di convoluzione sia un *operatore commutativo.*

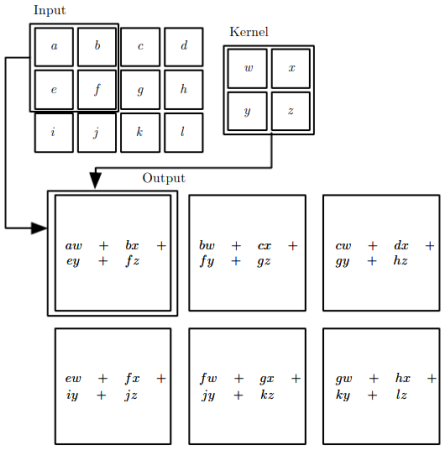

Figura 3.4: Esempio dell'azione di un filtro su un segnale bidimensionale [6]

L'esempio in Figura 3.4 mostra il significato di applicare una convoluzione bidimensionale da un punto di vista interpretativo. Da un punto di vista dell'implementazione su un computer però questo consiste in una semplice moltiplicazione per una matrice, che nel caso di segnali finiti unidimensionali equivale alla *Matrice di Toeplitz* e nel caso bidimensionale invece corrisponde alla *Matrice circolante a doppio blocco.*
Generalmente, siccome la dimensione dei filtri è piuttosto bassa rispetto alla dimensione dei segnali, queste matrici sono *sparse*, che rende l'operazione di convoluzione computazionalmente efficiente.

$$y = \begin{bmatrix} \vdots \\ y_{-2} \\ y_{-1} \\ \boxed{y_0} \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & h_0 & h_{-1} & h_{-2} & h_{-3} & h_{-4} & \cdots \\ \cdots & h_1 & h_0 & h_{-1} & h_{-2} & h_{-3} & \cdots \\ \cdots & h_2 & h_1 & \boxed{h_0} & h_{-1} & h_{-2} & \cdots \\ \cdots & h_3 & h_2 & h_1 & h_0 & h_{-1} & \cdots \\ \cdots & h_4 & h_3 & h_2 & h_1 & h_0 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}}_{H} \begin{bmatrix} \vdots \\ x_{-2} \\ x_{-1} \\ \boxed{x_0} \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} = Hx.$$

Figura 3.5: Convoluzione come prodotto per matrice infinita dell'operatore H [17]

$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} h_0 & h_{N-1} & h_{N-2} & \cdots & h_1 \\ h_1 & h_0 & h_{N-1} & \cdots & h_2 \\ h_2 & h_1 & h_0 & \cdots & h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \cdots & h_0 \end{bmatrix}}_{H} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} = Hx.$$

Figura 3.6: Convoluzione circolare come prodotto per matrice dell'operatore H [17]

L'operazione di convoluzione introdotta grazie all'utilizzo di queste reti presenta principalmente tre vantaggi:

1) *interazioni sparse*: generalmente gli strati delle reti sono pienamente connessi, presentando quindi oltre ai parametri delle moltiplicazioni, che tipicamente non sono sparse, anche numerosi parametri che collegano ogni input a ogni output dello strato successivo.
   Nelle reti convoluzionali invece, siccome si usano filtri di piccola dimensione, è necessario registrare solo pochi parametri e questo è efficiente sia statisticamente sia, soprattutto, computazionalmente.
   Se da un punto i vista computazionale risulta immediato il motivo, dovuto alle moltiplicazioni di matrici sparse, è interessante notare come, se queste reti sono abbastanza profonde, allora sebbene due nodi di strati diversi e lontani non siano direttamente connessi, è possibile tuttavia che ci sia un'interazione. "Questo permette alla rete di descrivere in maniera efficiente alcune interazioni complicate tra le variabili"[6].

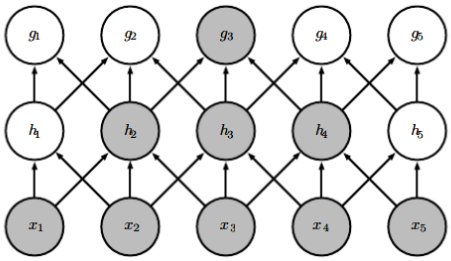

Figura 3.7: Esempio della possibilità di interazioni in reti convoluzionali profonde malgrado la mancanza di connessione diretta [6]

2) *condivisione dei parametri*: quando si utilizza una rete convoluzionale si applicano gli stessi filtri per ogni nodo di uno strato, condividendo quindi gli stessi parametri all'interno dello strato, a differenza delle classiche reti per cui invece è utilizzato esattamente una sola volta. Questo non riduce il numero di operazioni da fare ma la quantità di insieme di parametri da immagazzinare, e quindi riduce la memoria usata.

3) *equivarianza per traslazione*: In questi tipi di rete, la condivisione di parametri fa in modo che gli strati convoluzionali abbiano una particolare proprietà, la equivarianza per traslazione, che lega direttamente i cambiamenti nell'input ai cambiamenti nell'output.

Definizione 3.4.5. *Siano f,g due funzioni. Si dice che f è* ***equivariante*** *a g se*

$$f(g(x)) = g(f(x))$$

3.4.2 Pooling

In queste reti inoltre è tipico l'utilizzo di *strati di pooling* , ovvero funzioni che aiutano a ridurre le dimensioni, rendendo la rete efficiente da un punto di vista computazionale, e a trasformare il segnale originario in un altro più omogeneo. Questo accade in quanto ogni neurone in input allo strato si rimpiazza con una statistica riassuntiva locale, ovvero che viene calcolata solo per nodi vicini. Le funzioni di pooling più utilizzate sono quelle che sostituiscono all'input il valore medio e massimo, cioè rispettivamente *average pooling* e *max pooling*.
La diminuzione della dimensione avviene in quanto, una volta applicato il pooling, è possibile sottocampionare grazie alla ridondanza di valori uguali vicini.
Una conseguenza dell'omogenizzazione locale del segnale invece è che la rappresentazione risulta *invariante* a piccole traslazioni dell'input. Questa proprietà è utile in molti ambiti, ad esempio se si vuole riconoscere un oggetto in un'immagine senza tuttavia essere interessati a dove sia collocato precisamente.
L'applicazione del pooling su output di convoluzioni ottenute utilizzando filtri differenti può portare il modello ad essere invariante per alcune trasformazioni, come illustrato in figura 3.8

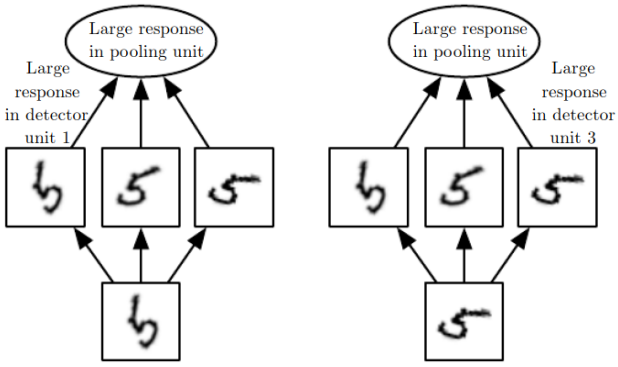

Figura 3.8: Esempio di invarianza per trasformazioni [6]

3.5 Reti ricorrenti

Le reti neurali ricorrenti sono modelli di Deep Learning introdotti per il processamento di dati sequenziali, ovvero dati in cui l'ordine delle osservazioni risulta importante. Questi algoritmi trovano applicazione soprattutto nell'analisi delle serie temporali, nel riconoscimento vocale e testuale e nell'analisi dei video.
L'idea fondamentale alla base delle reti ricorrenti è la *condivisione dei parametri*, ovvero l'utilizzo degli stessi parametri lungo posizioni, istanti di tempo o più in generale passi della sequenza, per due motivi principali:

- possibilità di applicazione del modello, in fase di test, su sequenze di lunghezza differente da quelle viste dall'algoritmo in fase di addestramento
- riduzione del numero di parametri e capacità di riconoscere l'informazione in diverse posizioni lungo la sequenza. Ad esempio una rete senza condivisione per imparare le frasi "ho mangiato una mela ieri" e "ieri ho mangiato una mela" utilizzerebbe il doppio dei parametri utilizzati da una che possiede questa proprietà, in quanto la prima non sarebbe capace di riconoscere che si tratta della stessa frase.
  Più in generale, l'assenza di condivisione dei parametri implicherebbe la necessità di addestrare la rete attraverso tutte le varianti che la frase può assumere, non rendendo possibile comunque il riconoscimento corretto di un'alterazione di questa in fase di test che non sia stata vista nell'addestramento dal modello.

Generalmente lo strato ricorrente di questi modelli può essere descritto attraverso un sistema dinamico con equazione ricorsiva:

$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right) \tag{3.1}$$

dove

- $h^{(t)}$ è lo stato dinamico del sistema chiamato *unità nascosta*
- $x^{(t)}$ è il segnale esterno in input al sistema al tempo t

- θ e f sono rispettivamente i parametri e la funzione parametrizzata dagli stessi. La condivisione dei parametri si può intuire dalla seguente formula osservando che sia θ che f non dipendono da t

Un ruolo importante in queste reti è svolto dalle unità nascoste $h^{(t)}$. Infatti "quando la rete ricorrente viene addestrata a svolgere un'attività che richiede la previsione del futuro attraverso il passato, tipicamente la rete impara ad utilizzare $h^{(t)}$ come una specie di riassunto con perdite degli aspetti rilevanti per l'attività della sequenza di input fino a t. Questo riassunto in generale è necessariamente affetto da perdite dal momento che $h^{(t)}$ mappa una sequenza di lunghezza arbitraria $(x^{(t)}, x^{(t-1)}, ..., x^{(1)})$ in un vettore di lunghezza fissata" [6].

3.5.1 Varianti reti ricorrenti

Esistono principalmente tre modi per implementare una rete ricorrente, caratterizzati da quanti valori vengono utilizzati per il calcolo della funzione di perdita e dalle connessioni che permettono alla rete di utilizzare l'informazione passata:

1) connessioni unità nascoste - unità nascoste e sequenza in output

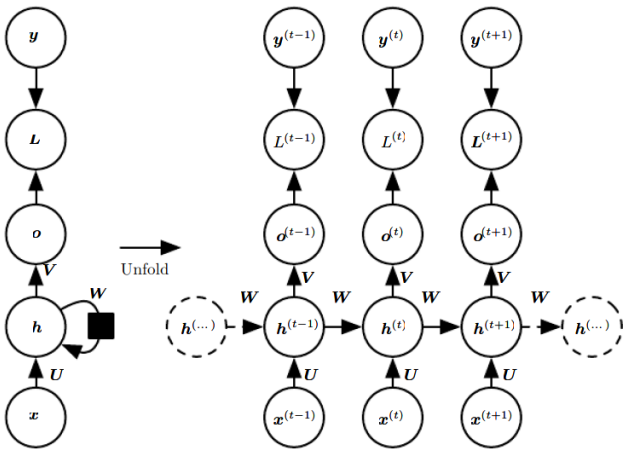

Figura 3.9: Reti ricorrenti: sequenza in output e collegamento unità nascoste - unità nascoste [6]

Il grafico sulla sinistra è una rappresentazione compatta, attraverso l'introduzione di cicli sul grafo, di quello sulla destra che enfatizza l'utilizzo dei medesimi parametri nella rete. Gli elementi che compongono il grafico sono: il segnale esterno $\{x^{(t)}\}_t$ in input, le unità nascoste $\{h^{(t)}\}_t$, gli output $\{o^{(t)}\}_t$ uscenti dalle unità nascoste che tramite delle trasformazioni diventano i valori stimati $\{\hat{y}^{(t)}\}_t$ delle vere etichette $\{y^{(t)}\}_t$ e infine la sequenza di funzioni di perdita $\{L^{(t)}\}_t$, utili in quanto la loro somma costituisce la perdita totale sulla sequenza $L = \sum_t L^{(t)}$.

2) connessioni unità output - unità nascoste e sequenza in output

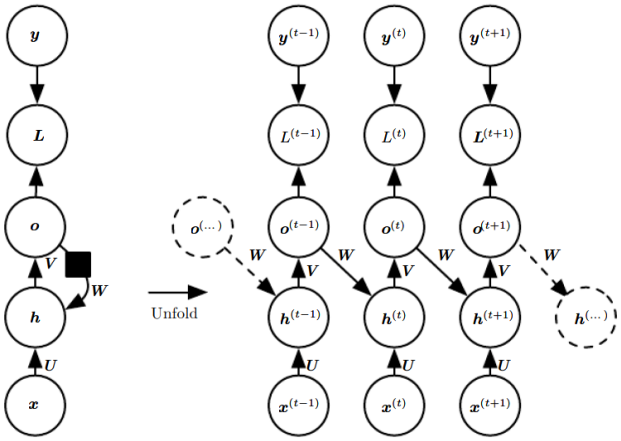

Figura 3.10: Reti ricorrenti: sequenza in output e collegamento unità output - unità nascoste [6]

Il modello descritto precedentemente, che deriva dall'applicazione diretta della formula 3.1, risulta costoso da allenare in quanto non è possibile parallelizzare i calcoli in fase di addestramento. Si può pensare però di riformulare l'equazione precedente come:

$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{o}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right) \tag{3.2}$$

ottenendo quindi una connessione indiretta tra le unità nascoste. Questa modifica ha il vantaggio di permettere la parallelizzazione in fase di addestramento, che risulta possibile dal momento che ora i passi della sequenza di addestramento sono disaccoppiati.
La rete definita così però presenta uno svantaggio che deriva dal fatto che generalmente l'output $o^{(t)}$ è inferiore come dimensione a $h^{(t)}$, facendo passare quindi meno informazione rispetto alla rete con connessione diretta tra le unità nascoste.
Modelli come questi possono essere addestrati tramite la *forzatura dell'insegnante*, ovvero l'utilizzo in fase di addestramento dell'etichetta vera $y^{(t-1)}$ al posto di $o^{(t-1)}$, come dato in input per il calcolo di $h^{(t)}$. In formule quindi si ha che:

$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{y}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right) \tag{3.3}$$

3) connessioni unità nascoste - unità nascoste e singolo output

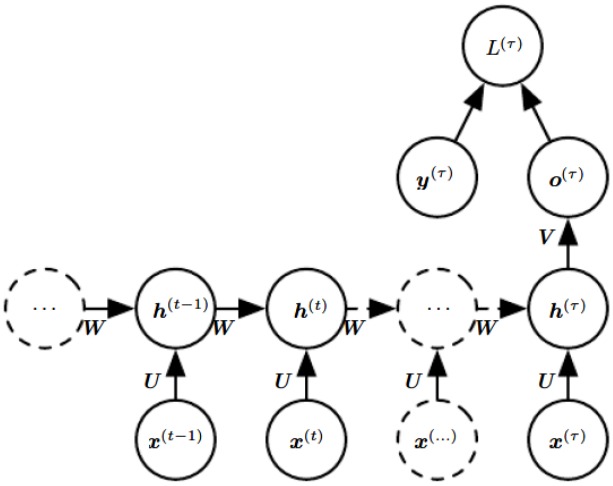

Figura 3.11: Reti ricorrenti: singolo output e collegamento unità nascoste - unità nascoste [6]

La differenza tra questi modelli e i due precedenti è la presenza di un'unica funzione di perdita, calcolata all'ultimo istante della sequenza. Questi modelli vengono principalmente utilizzati quando l'interesse è rivolto solamente all'ultimo istante della sequenza in input, utilizzando però tutta l'informazione passata.

3.5.2 Reti ricorrenti bidirezionali

In molte applicazioni dove l'obiettivo non è la previsione di uno o più valori futuri partendo dai valori passati (noti), si possono utilizzare reti neurali ricorrenti bidirezionali. Questa tipologia di algoritmi utilizza due strati ricorrenti nella stessa rete, ma senza che vi siano connessioni dirette tra loro, sfruttando quindi sia l'informazione nell'ordine passato in input al modello che anche quello opposto, risolvendo quindi una limitazione delle classiche reti ricorrenti a discapito dell'introduzione di parametri.
Le maggiori applicazioni in cui queste reti sono utilizzate sono il riconoscimento vocale, la traduzione testuale e l'analisi delle dipendenze.

3.5.3 Backpropagation lungo il tempo (BTTP)

Nelle reti neurali ricorrenti i parametri vengono aggiornati tramite il classico algoritmo di backpropagation, che nel caso specifico di queste reti prende il nome di backpropagation lungo il tempo.
Per semplicità si consideri come rete quella raffigurata in figura 3.9, assumendo che l'ultimo istante di tempo della sequenza è τ e si calcolino le derivate rispetto alla matrice di parametri W, ma si potrebbe generalizzare anche ad altre versioni della rete e a parametri

differenti.

Se $L = \sum_{t=1}^{\tau} L^{(t)}$, allora:

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W} \tag{3.4}$$

sapendo che i pesi W sono condivisi nel calcolo di tutti gli $h^{(t)}$, vale che:

$$\begin{cases} \frac{\partial h^{(t)}}{\partial W} = \sum_{j=0}^{t} \frac{\partial h^{(t)}}{\partial h^{(j)}} \cdot \frac{\partial h^{(j)}}{\partial W} \\ \frac{\partial h^{(t)}}{\partial h^{(j)}} = \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial h^{(t-2)}} \cdot \ldots \cdot \frac{\partial h^{(j+1)}}{\partial h^{(j)}} \end{cases} \implies \frac{\partial h^{(t)}}{\partial W} = \sum_{j=0}^{t} \left(\prod_{k=j+1}^{t} \frac{\partial h^{(k)}}{\partial h^{(k-1)}} \right)$$

Quindi

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \sum_{j=0}^{t} \frac{\partial L^{(t)}}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \left(\prod_{k=j+1}^{t} \frac{\partial h^{(k)}}{\partial h^{(k-1)}} \right) \tag{3.5}$$

3.5.4 Problema dell'esplosione/scomparsa del gradiente

Il problema della scomparsa del gradiente si verifica sia per reti neurali feed-forward profonde [13] sia per le reti ricorrenti. Quest'ultime infatti consistono nella composizione della stessa funzione ripetuta molteplici volte, una per ogni passo della sequenza, che possono essere non lineari. Per dimostrare il problema tipico che si ha nel calcolo della backpropagation, per semplicità si assuma una rete in cui tutte le non-linearità siano rimosse. Senza queste quindi, la relazione ricorsiva tra $h^{(t-1)}$ e $h^{(t)}$ si può scrivere anche come:

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top} \boldsymbol{h}^{(t-1)} \tag{3.6}$$

che si può scrivere equivalentemente come

$$\boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^{t}\right)^{\top} \boldsymbol{h}^{(0)} \tag{3.7}$$

Se W ammette una decomposizione della forma $\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top}$, con $\boldsymbol{Q}$ ortogonale, allora la formula 3.7 si può riscrivere come

$$\boldsymbol{h}^{(t)} = \boldsymbol{Q}^{\top}\boldsymbol{\Lambda}^{t}\boldsymbol{Q}\boldsymbol{h}^{(0)} \tag{3.8}$$

La matrice $\boldsymbol{\Lambda}^t$, essendo una matrice diagonale, è composta dagli autovalori di W elevati a t, cioè $(\lambda^{(i)})^t$.

Per valori di t abbastanza grandi, quindi per dipendenze a lungo termine, si ha che:
- se $|\lambda^{(i)}| < 1$, allora $(\lambda^{(i)})^t \to 0$
- se $|\lambda^{(i)}| > 1$, allora $(\lambda^{(i)})^t \to M$, dove M è un valore elevato

Illustrato il problema, si può applicare l'idea alla funzione di ricorrenza mostrata nell'equazione 3.5.
Nel pratico il problema dell'esplosione/scomparsa del gradiente fa si' che i parametri della rete o non vengano aggiornati (quando gradiente svanisce) o lo siano in maniera eccessiva (quando invece esplode).

3.5.5 Reti Long short-term memory (LSTM)

Le reti LSTM sono una versione modificata delle reti ricorrenti, introdotte per risolvere il problema della scomparsa del gradiente. Questi modelli però non risolvono il problema analogo dell'esplosione dello stesso, infatti "sebbene gli LSTM tendano a non soffrire del problema della scomparsa del gradiente, essi possono avere gradienti esplosivi"[16].
L'idea che permette a questi modelli di funzionare è l'introduzione di una memoria ulteriore rispetto a quella a breve termine $h^{(t)}$, ovvero lo *stato della cella* o memoria a lungo termine $c^{(t)}$, che viene aggiornata tramite l'uso di *porte*, che svolgono il compito di decidere quali informazioni rimuovere o aggiungere allo stato della cella.
Le reti LSTM quindi sono composte da celle collegate tra loro dove ognuna di essere al suo interno è suddivisa in tre blocchi: *porta di dimenticanza* (f), *porta di ingresso* (i) e *porta d'uscita* (o).

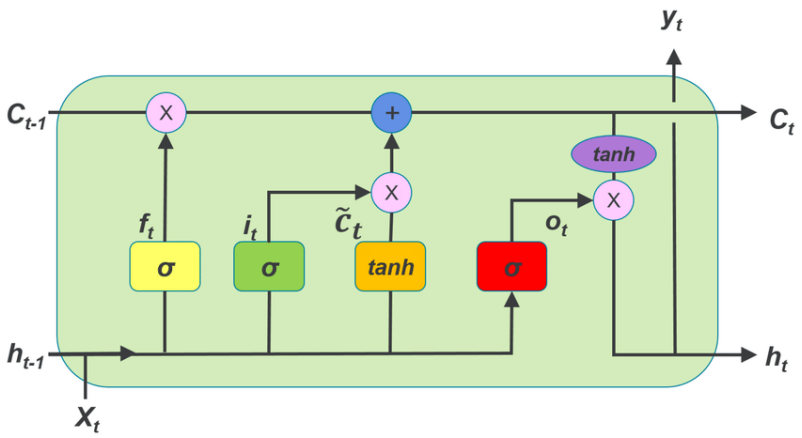

Figura 3.12: Esempio di una singola cella LSTM e del flusso di informazioni che scorre attraverso[9]

dove le operazioni in Figura 3.12 sono definite come:

- $\sigma = \frac{1}{1+e^{-x}}$
- $\odot$ prodotto di Hadamard (moltiplicazione elemento per elemento)
- W_{lm} dove l={x,h} e m={f,i,c,o}
- b_l vettore di distorsione utilizzato nella porta p
- $\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right)$

- $\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right)$
- $\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right)$
- $\tilde{\mathbf{c_t}} = \tanh\left(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right)$
- $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c_t}}$
- $\mathbf{h}_t = \tanh\left(\mathbf{o}_t \odot \mathbf{c}_t\right)$

Per una spiegazione più approfondita, si consideri una cella in un generico istante t.
Il primo passo è svolto dalla porta di dimenticanza che decide quali informazioni eliminare da c_{t-1} e in che misura. Tale decisione viene presa moltiplicando, elemento per elemento, lo stato della cella c_{t-1} per f_t, cioè funzione logistica applicata a h_{t-1} e a x_t. Siccome il vettore f_t presenta valori contenuti in [0,1], valori di f_t vicini a 0 implicano che lo stato della cella "dimentichi" l'informazione corrispondente, mentre valori vicini a 1 permettono all'informazione di essere perfettamente mantenuta o quasi.
Il secondo passo consiste nell'aggiunta allo stato della cella di nuovi nuovi elementi, opportunamente calcolati attraverso due passaggi: il primo avviene calcolando una funzione logistica che decide quali valori dovranno essere aggiornati i_t e calcolando i valori candidati ad essere sommati allo stato della cella $\tilde{c_t}$ attraverso una funzione iperbolica; il secondo passaggio invece consiste nella moltiplicazione dei due vettori ottenuti, trovando così i valori che saranno sommati allo stato della cella c_{t-1} e forniranno c_t.
Infine l'ultimo passo consiste nel calcolo dell'output relativo a quella cella h_t, che risulta essere anche l'unità nascosta in input alla cella successiva. L'output viene calcolato in questo modo: prima si utilizza una funzione logistica applicata a h_{t-1} e a x_t che decide quali parti della cella c_t verranno mantenuti in output, poi si calcola il prodotto tra i valori appena calcolati e $\tanh(c_t)$, dove l'applicazione della funzione tangente iperbolica è utile a stabilizzare i valori in modo che appartengano all'intervallo [-1,+1].

3.6 Reti autoencoder

Gli autoencoder sono un tipo di rete neurale non supervisionato dove si cerca di codificare e decodificare i dati in input al modello per ottenere in output una ricostruzione degli stessi che soddisfi alcune proprietà. Il procedimento di codifica e decodifica permette, sotto opportune condizioni, di catturare le caratteristiche rilevanti nei dati, filtrando quindi i connotati caratterizzanti solo alcune osservazioni del dataset.
La rete è composta da due funzioni: un *codificatore* C e un *decodificatore* D. Se si definisce x l'insieme dei valori in input al modello, l'output del modello sarà $\hat{x} = D(C(x))$.
La funzione di perdita risulta uguale a quella dei precedenti modelli ad eccezione di utilizzare come variabile risposta y i dati che sono passati in input al modello x. Nel caso in cui la rete sia sufficientemente capace, accade che $\hat{x} = x$, riuscendo a riprodurre perfettamente i dati in input ma senza fornire la possibilità estrarre informazioni rilevanti dagli stessi.

3.6.1 Autoencoder incompleti e autoencoder regolarizzati

Per risolvere questo problema, e quindi ottenere una rappresentazione in output leggermente differente dai dati in input, sono introdotti due tipi di autoencoder che possono anche essere combinati tra loro:

Autoencoder incompleti

L'idea più intuitiva e utilizzata per non permettere alla rete di ricostruire perfettamente i dati in input consiste nell'imporre la riduzione in fase di codifica della dimensionalità lungo la rete, e quindi agli strati che la compongono, per poi avere un aumento della dimensione in fase di decodifica, per cercare di ricostruire i dati con le loro dimensioni iniziali. Gli autoencoder che presentano la struttura appena descritta sono definiti autoencoder incompleti.

Autoencoder regolarizzati

Al posto di ridurre la dimensionalità, per ridurre la capacità di un modello preservando però le caratteristiche rilevanti si può pensare di modificare la funzione di perdita in modo da ottenere diverse proprietà.

Un esempi di rete regolarizzata sono gli autoencoder sparsi. La funzione di perdita in questi modelli è definita come

$$L(\boldsymbol{x}, D(C(\boldsymbol{x}))) + \Omega(\boldsymbol{C}) \tag{3.9}$$

ovvero come la funzione di perdita classica, con y=D(C(x)), a cui però si somma una funzione dei parametri del codificatore $\Omega(\boldsymbol{C})$, che rende sparsa la rete, analogamente a quel che avviene nella regressione con i modelli RIDGE e LASSO.

Un altro esempio di rete regolarizzata sono una diretta variante dell'esempio precedente. In questo caso la funzione di perdita è definita come

$$L(\boldsymbol{x}, D(C(\boldsymbol{x}))) + \lambda \sum_i \|\nabla_{\boldsymbol{x}} C_i\|^2 \tag{3.10}$$

ovvero la funzione di perdita classica, con y=D(C(x)), a cui è aggiunta una funzione di penalizzazione che permette alla rete di imparare una funzione i cui cambiamenti sono controllati da un iperparametro $\lambda \in \mathbb{R}^+$:
per $\lambda \to 0$, la funzione di perdita risulta simile a quella classica;
per $\lambda \to +\infty$, la funzione imparata dalla rete non cambia per piccole variazioni dell'input

3.6.2 Applicazioni

Le reti autoencoder sono particolarmente utili per svolgere i seguenti compiti:

- *riduzione del rumore* che di solito affiggono segnali audio e immagini;
- *riduzione della dimensionalità* in modo da permettere una migliore visualizzazione dei dati. Nel caso in cui vengano utilizzati autoencoder con funzioni codificatrice e decodificatrice lineari e l'errore quadratico medio come funzione di perdita, allora un autoencoder impara lo stesso sottospazio generato dalla PCA, analisi delle componenti principali. Quindi si può notare come questi modelli risultino una generalizzazione della PCA utile per riduzioni non lineari delle dimensioni;
- *rilevamento di anomalie*. Infatti questi algoritmi, se addestrati a ricostruire in maniera fedele i dati, possono essere usati per calcolare la distribuzione dell'errore di ricostruzione che verrà utilizzata in seguito, anche possibilmente attraverso trasformazioni, come indicatore di normalità/anomalia;
- *riconoscimento del discorso*, *componimento di didascalie di immagini* e *traduzioni di frasi*. Nella sezione riguardante le reti ricorrenti 3.5 è stato illustrato come sia possibile mappare sequenze di input in output di dimensione fissa, ovvero o uguale alla lunghezza dell'input oppure corrispondente ad un unico valore monodimensionale. Molte volte però è necessario che le dimensioni dell'input e dell'output non siano le medesime, si pensi ad una traduzione tra frasi di due lingue diverse ad esempio. Questo problema è risolto attraverso l'utilizzo combinato di un autoencoder e di uno o più strati ricorrenti.

3.7 Modelli applicati

3.7.1 Primo modello: LSTM Encoder-Decoder

Il primo modello utilizzato per l'analisi del problema consiste in un autoencoder composto da strati ricorrenti LSTM.
L'idea su cui si fonda il modello è di addestrare una rete a "ricostruire il comportamento normale della serie temporale, per poi in seguito usare l'errore di ricostruzione per identificare le anomalie"[11]. Infatti, siccome la rete in fase di addestramento ha utilizzato solo dati normali, ci si aspetta che presenti errori di ricostruzione superiori per osservazioni etichettate come anomale. Si tratta quindi di una rete neurale a una classe, poiché le osservazioni anomale sono usate solo in fase di validazione e test.
L'utilizzo dell'autoencoder nel modello è giustificato dalla necessità di ricostruire dei dati catturando solo gli aspetti rilevanti degli stessi. Inoltre, a causa della natura dei dati di tipo sequenziale, è necessario utilizzare reti ricorrenti, di cui LSTM è un tipo specifico adatto a ridurre la difficoltà introdotta dalla scomparsa del gradiente, problema che non permette un ottimale calcolo e aggiornamento dei pesi lungo la rete.

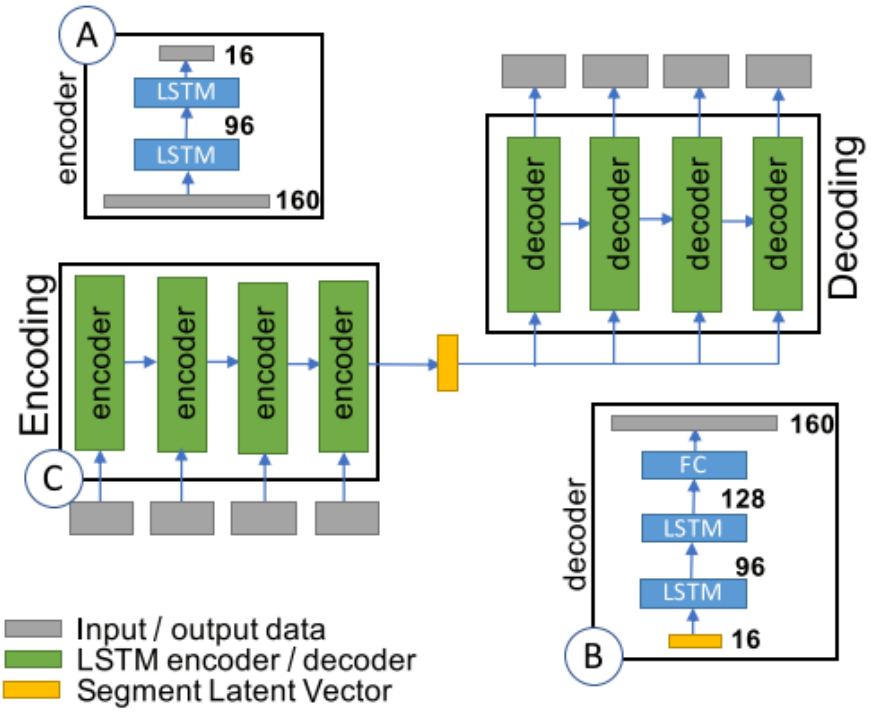

Figura 3.13: Primo modello: LSTM Encoder-Decoder [15]

Il modello utilizzato per il caso d'uso è ispirato a quello illustrato in Figura 3.13. Rispetto a questo però sono state utilizzate dimensioni differenti dei valori in input e all'interno del

modello. Inoltre allo strato pienamente connesso stata applicata una tecnica di *dropout*. Questo metodo, anche chiamato diluizione, è utile a regolarizzare una rete neurale per ridurre il problema dell'overfitting di un modello sui dati di addestramento. Infatti "l'overfitting può essere ridotto utilizzando il dropout per prevenire complessi co-adattamenti sui dati di addestramento"[8], dove per co-adattamenti si riferisce a quando diverse unità nascoste in una rete neurale hanno un comportamento altamente correlato, riducendo la capacità del modello. Questa tecnica consiste nell'eliminare, sugli strati in cui è definita, momentaneamente alcuni nodi dalla rete prima del elaborazione di ogni osservazione. Questa eliminazione avviene in maniera casuale e solo in fase di addestramento. La probabilità con cui i nodi vengono eliminati è un iperparametro $p \in [0,1]$ che deve essere definito prima dell'addestramento e tipicamente è inferiore a 0.5.

Per applicare il modello, in cui ogni osservazione in input è una sequenza di record del dataset originario M, si crea un tensore X in cui ogni osservazione è una successione ordinata temporalmente e campionata dal dataset iniziale, procedendo di passo 1. Quindi se si definisce T la lunghezza della sequenza, che nel caso d'uso è un iperparametro che assume valore 60, V il numero di variabili che compongono la singola osservazione del dataset originario, che coincide a 30 sensori nel problema analizzato, e n la cardinalità del dataset M, allora

$$X^{(t)} = \left(M^{(t-T+1)}, M^{(t-T+2)}, ..., M^{(t)}\right) \quad \forall t \in [T, n]$$

passando quindi da una matrice M di dimensioni (n,V) a un tensore X di misura ((n-T+1),T,V), dove all'elemento $X^{(i)(j)(k)}$ corrisponde il valore della variabile k nell'i-esima osservazione e j-esimo elemento della sequenza.
La rete utilizzata riceve in input una singola osservazione $X^{(t)}$, di dimensioni (1,T,V) e attraverso un strato LSTM produce in output un tensore di dimensioni (1,T,16). A quest'ultimo è ulteriormente applicato un altro strato LSTM che produce in output un tensore di dimensioni (1,1,4), codificando di fatto l'informazione passata in input al modello.
La ricostruzione avviene prima ripetendo T volte l'informazione codificata, ottenendo quindi un tensore di dimensioni (1,T,4), poi applicando due strati LSTM i cui output sono rispettivamente di dimensione (1,T,4) e (1,T,16). Infine è applicato uno strato il dropout con p=0.1 a uno strato pienamente connesso il cui output, di dimensioni (1,T,V), è la ricostruzione $\hat{X}^{(t)}$ dell'osservazione di input.

La funzione di perdita utilizzata per addestrare la rete è l'errore medio assoluto (Mean Absolute Error). Per il record t-esimo in input al modello, ovvero $X^{(t)}$, quindi è calcolata come:

$$Loss^{(t)} = \frac{\sum_{i=t-T+1}^{t} \sum_{j=1}^{V} |X^{(t)(i)(j)} - \hat{X}^{(t)(i)(j)}|}{T \cdot V} \tag{3.11}$$

Valutazione del modello: calcolo punteggio di anomalia e delle due soglie

Una volta trovato il modello migliore, per identificare le anomalie è stato necessario calibrare una soglia τ che ha la funzione di classificatore binario, dove 0 equivale ad un'osservazione normale e 1 ad un valore anomalo.
Per trovare il valore ottimale della soglia è necessario introdurre le seguenti variabili:

- $E^{(t)} = |X^{(t)(T)} - \hat{X}^{(t)(T)}|$
  tensore che rappresenta l'errore di ricostruzione per l'istante t. E quindi equivale ad una matrice di dimensioni (n-T,V);
- $\chi^{(t)(j)} = \begin{cases} 1 & \text{se } E^{(t)(j)} > \alpha \\ 0 & \text{altrimenti} \end{cases}$
  funzione indicatrice che, per ogni istante t e variabile j, assume valore 1 se l'errore di ricostruzione è maggiore di un iperparametro α;
- $a^{(t)} = \sum_{j=1}^{V} \chi^{(t)(j)}$
  punteggio di anomalia. Svolge la funzione di contatore di quante variabile hanno superato la soglia α;
- $\hat{y}^{(t)} = \begin{cases} 1 & \text{se } a^{(t)} > \tau \\ 0 & \text{altrimenti} \end{cases}$
  anomalie predette.

La scelta degli iperparametri α e τ avviene attraverso una ricerca esaustiva, utilizzando vari valori per tutte e due gli iperparametri, dove il criterio per la scelta degli iperparametri equivale alla massimizzazione della statistica F1 score, ovvero il valore che confronta contemporaneamente precisione e richiamo di una classificazione binaria.

3.7.2 Secondo modello: MSCRED

Il secondo modello utilizzato, Multi-Scale Convolutional Recurrent Encoder-Decoder [18], consiste in un'architettura codificatrice-decodificatrice in cui l'idea sottostante per rilevare le anomalie risulta simile al modello precedente, ovvero si assume che per osservazioni appartenenti a dataset anomali la rete non riesca a ricostruire bene i record, poiché durante la fase di addestramento ha imparato i pattern appartenenti alle osservazioni non anomale.

Generazione matrici e tensori

Diversamente dal modello utilizzato precedentemente, dove si modellava la rete per imparare a ricostruire la serie temporale multivariata, in questo modello prima si generano delle matrici apposite e poi si utilizzano quelle per addestrare, validare e testare la rete.
Queste matrici sono ottenute calcolando le dipendenze tra le singole variabili della serie temporale multivariata, che nel caso d'uso equivalgono a sensori, e quindi l'obiettivo della rete è imparare a riconoscere come variano queste dipendenze nel tempo e a rilevare le anomalie come istanti di tempo in cui queste relazioni si comportano differentemente da quanto ci si aspetterebbe.
La rete utilizzata è un modello multi-scala in quanto le relazioni tra le variabili vengono calcolate, e in seguito modellate dalla rete, utilizzando tre scale differenti, ovvero tre diverse finestre temporali, assumendo che si possa trarre informazioni differenti da ciascuna.
Le dipendenze tra le variabili quindi si modellano come una covarianza tra variabili aleatorie, a meno della sottrazione della media da queste, calcolato tramite le osservazioni precedenti, in un lasso di tempo lungo quanto la scala. Quindi per ogni osservazione si otterranno tre matrici.
Se si definisce:

- w uno dei tre iperparametri di scala che vengono forniti in input al modello, con w $\in$ {10,30,60}
- T la lunghezza massima della serie
- x la serie temporale multivariata, con $x_i^{(t)}$ valore dell'i-esimo sensore al tempo t
- $M^{(w)}$ la matrice simmetrica generata con $m_{ij}^{(t)(w)}$ l'elemento (i,j)-esimo della matrice ottenuta per mezzo della scala w al tempo t

allora ogni elemento della matrice è calcolato come:

$$m_{ij}^{(t)(w)} = \frac{\sum_{\delta=0}^{w} x_i^{(t-\delta)} x_j^{(t-\delta)}}{w} \quad \forall t \in [60, T]$$

Ottenendo tre matrici simmetriche di dimensioni (30,30), il dataset iniziale viene sostituito dall'utilizzo di uno nuovo composto da tensori di dimensioni (30,30,3).
A causa della profondità del modello è delle dimensioni delle matrici, si può sottocampionare il dataset prendendo un'osservazione ogni gap_time, che è un iperparametro passato in input al modello. Nell'analisi svolta questo iperparametro ha assunto valore uguale a 10.

Struttura del modello

Per addestrare il modello non in modo tale che ricostruisca l'informazione al tempo t non basta passare in input solamente il tensore corrispondente. Ciò che viene fatto è utilizzare h tensori (30,30,3), dove h è un iperparametro che per il caso d'uso è stato settato uguale a 5, in modo tale che entrambi apportino un contributo alla ricostruzione. Quindi per analizzare il record al tempo t si utilizzano le osservazioni fino al tempo t-h+1. Si può quindi ridefinire il dataset di tensori (30,30,3) in un altro di dimensioni (5,30,30,3), che possiede 4 record in meno rispetto al precedente.
L'input del modello è quindi un tensore di dimensioni (5,30,30,3), come si può vedere dall'immagine seguente dove h blocchi vengono passati in input:

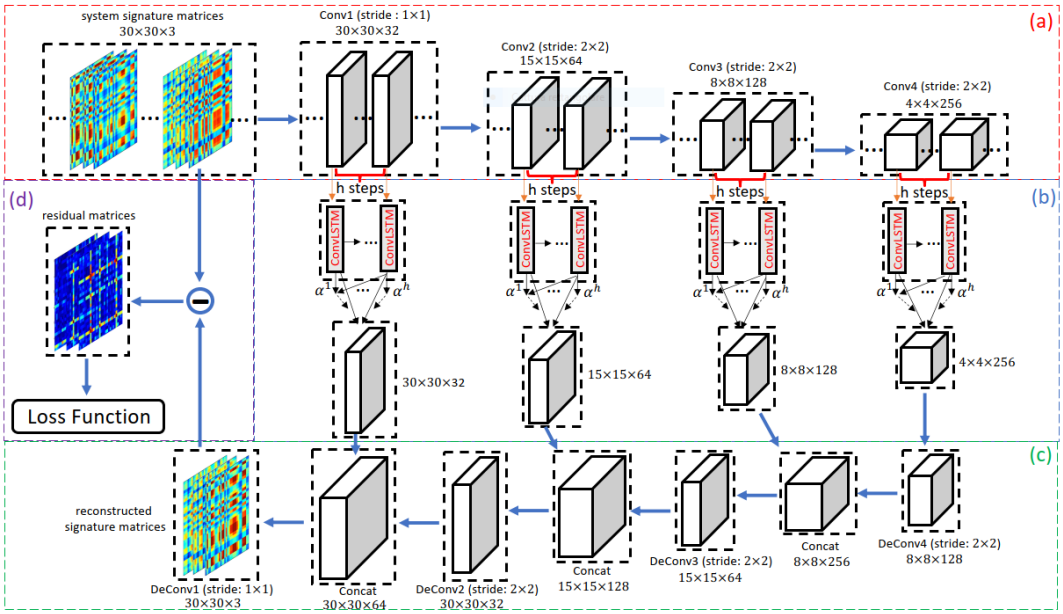

Figura 3.14: Secondo modello: MSCRED [18]

Dalla Figura 3.14 si nota come il compito di codificare è svolto dalle fasi (a) e (b). In quest'ultima è presente anche una tecnica di attenzione che permette di ridurre la dimensione da h blocchi ad uno solo, utilizzando solo le informazioni rilevanti. La fase (c) consiste in una decodifica e ricostruzione dell'ultimo tensore e nella fase (d) si calcolano le matrici residue, che saranno utilizzate infine nella funzione di perdita.

Fase (a)

La fase (a) consiste in quattro strati convoluzionali in cui viene preservata dimensione h, che indica il numero di blocchi e si applica una convoluzione bidimensionale alle matrici, codificando quindi l'informazione attraverso l'uso di molteplici filtri. Di seguito sono riportate rispettivamente le dimensioni in input e output per ogni strato:

- strato 1: (h,30,30,3)-(h,30,30,32)
- strato 2: (h,30,30,32)-(h,15,15,64)
- strato 3: (h,15,15,64)-(h,8,8,128)
- strato 4: (h,8,8,128)-(h,4,4,256)

Fase (b)

La fase (b) si suddivide in due parti:
la prima consiste nell'utilizzo di quattro strati ConvLSTM applicati direttamente ai tensori ottenuti dalla fase (a). Questo tipo di strati è una variante degli strati LSTM dove vengono semplicemente sostituite le operazioni di moltiplicazione interne alle celle LSTM con operazioni di convoluzione utili a preservare la dimensione in entrata dei tensori. Attraverso gli strati ConvLST, quindi si catturano le strutture temporali nei dati in cui vogliamo catturare i pattern temporali. Gli input di questi strati sono i tensori uscenti dalla fase (a), ovvero di dimensioni:

- (h,30,30,32) per lo strato 1
- (h,15,15,64) per lo strato 2
- (h,8,8,128) per lo strato 3
- (h,4,4,256) per lo strato 4

La seconda parte della fase (b) consiste in una somma pesata degli h tensori in uscita dalle celle ConvLSTM, dove i pesi α sono calcolati autonomamanete dalla rete, e quindi senza bisogno di introdurre parametri che rallenterebbero l'addestramento della rete, attraverso una funzione softmax che assegna al peso α^i un valore proporzionale a quanto il tensore i-esimo è simile all'ultimo tensore, ovvero quello che si riferisce al tempo t che si vuole ricostruire.
Di seguito è riportata la formula per il calcolo degli α^i:

$$\hat{\mathcal{H}}^{t,l} = \sum_{i \in (t-h+1,t)} \alpha^i \mathcal{H}^{i,t}, \alpha^i = \frac{\exp\left\{\frac{\mathrm{Vec}\left(\mathcal{H}^{t,l}\right)^{\mathrm{T}} \mathrm{Vec}\left(\mathcal{H}^{i,l}\right)}{h}\right\}}{\sum_{i \in (t-h+1,t)} \exp\left\{\frac{\mathrm{Vec}(\mathcal{H}^{t,l})^{\mathrm{T}} \mathrm{Vec}(\mathcal{H}^{i,l})}{h}\right\}}$$

dove Vec() è un operatore che vettorizza una matrice passata in input, $\mathcal{H}^{j,l}$ consiste nel j-esimo output, dove j=t-h+1,...,t , dell'l-esimo strato tra i quattro ConvLSTM. Il tensore risultante dalla somma pesata è il tensore $\hat{\mathcal{H}}^{j,l}$.
Quindi alla fine della fase (b) si ottiene in output tensori di dimensioni:

- (1,30,30,32)
- (1,15,15,64)
- (1,8,8,128)
- (1,4,4,256)

Fase (c)

La fase (c) consiste nell'applicazione di strati deconvoluzionali in cui, attraverso una concatenazione degli output della fase (b) e dei precedenti tensori deconvoluzionati, tranne che per l'ultimo strato in cui questo non può avvenire, si arriva alla ricostruzione dell'ultimo tensore passato in input al modello tra gli h, ovvero $\hat{X}$ che ha dimensioni (1,30,30,3).

Fase (d)

Nella fase (d) si calcolano i tensori residui come la differenza tra l'h-esimo passato in input al modello X e quello ricostruito $\hat{X}$. Su questa differenza si calcola una funzione di perdita, definita come la somma per ogni scala del quadrato della norma di Frobenius. Se si indica con $X^{(t)(c)}$ l'ultima matrice passata in input al modello al tempo t calcolata con finestra temporale di lunghezza c e $\hat{X}^{(t)(c)}$ la corrispondente ricostruzione, allora la funzione di perdita si scrive come:

$$Loss^{(t)} = \sum_{c=1}^{3} \left\| X^{(t)(c)} - \hat{X}^{(t)(c)} \right\|_F^2$$

Valutazione del modello: calcolo punteggio di anomalia e delle due soglie

Una volta addestrato opportunamente il modello, utilizzando un dataset di convalida per scegliere il modello che presenta il minor errore di generalizzazione, si ricercano le soglie α e τ utili alla ricostruzione. Sebbene il modello sia stato addestrato utilizzando tre scale diverse, in cui nella funzione di perdita ognuna apportava un contributo, e quindi l rete è stata addestrata ad ottenere buoni risultati per ogni scala, in fase di validazione e di test si utilizzano solamente la scala più piccola, sia per ottenere un risparmio computazionale sia perché si è riscontrato che le ricostruzioni della scala più piccola permettono di raggiungere una migliore classificazione.

Il procedimento per la ricerca delle soglie è simile a quello utilizzato per il precedente modello. La differenza è che per ogni istante si considera un punteggio di anomalia per ogni sensore, per poi registrare come punteggio di anomalia per quell'istante solamente il valore più alto tra quelli calcolati. In pratica si considera per ogni istante qual è il punteggio di anomalia per il sensore ricostruito in maniera peggiore.
Sia quindi:

- $E^{(t)} = |X^{(t)(10)} - \hat{X}^{(t)(10)}|$
  la matrice residua per il record t di dimensioni (30,30) ed elemento (i,j)-esimo $E^{(t)}_{(i,j)}$;
- $I^{(t)}_{(i,j)} = \begin{cases} 1 & \text{se } E^{(t)}_{(i,j)} > \alpha \\ 0 & \text{altrimenti} \end{cases}$
  funzione indicatrice rispetto all'osservazione t che restituisce come valori diversi diversi da 0, e uguali ad 1, solo gli errori di ricostruzione superiori alla soglia α;

- $v^{(t)(i)} = \sum_{j=1}^{30} (I^{(t)}_{(i,j)} + I^{(t)}_{(j,i)})$
  elemento i-esimo del vettore contenente i punteggi di anomalia al tempo t per il sensore i. Si sarebbe anche potuto utilizzare per il calcolo di $v^{(t)(i)}$ una sola funzione indicatrice, $I^{(t)}_{(i,j)}$ ad esempio, in quanto si sospetta che la rete ricostruisca in maniera simmetrica. A causa della possibilità di avere errate ricostruzioni però è preferibile utilizzare una sommatoria sia sulle righe che sulle colonne di $I^{(t)}$;

- $a^{(t)} = \max_i v^{(t)(i)}$
  punteggio di anomalia al tempo t, equivalente al numero di ricostruzioni errate oltre la soglia α sia sulle righe sia sulle colonne da parte del peggior sensore;

- $\hat{y}^{(t)} = \begin{cases} 1 & \text{se } a^{(t)} > \tau \\ 0 & \text{altrimenti} \end{cases}$
  anomalie predette.

Per la scelta del miglior modello si utilizza la metrica F_1 score, cercando i valori di α e τ che massimizzano quindi il compromesso tra precisione e richiamo.

Capitolo 4

Anomaly detection

4.1 Panoramica sull'individuazione delle anomalie

"L'identificazione di anomalie si riferisce al problema di trovare delle strutture nei dati non conformi al comportamento atteso"[4].
Gli ambiti in cui si applica un'analisi del genere sono molteplici. Proprio per questo le conseguenze della comparsa di anomalie generano danni che possono essere sia di lieve entità sia di proporzioni di gran lunga maggiori. Si ricorre alla ricerca di anomalie per motivi di sicurezza, salute e molto spesso per evitare o ridurre perdite in ambito aziendale. Quest'analisi trova applicazione nella sicurezza riguardante l'abuso e intrusione di rete, nella videosorveglianza e nel rilevamento di frodi bancarie. Inoltre è tipico utilizzare una ricerca di questo tipo per monitorare ad esempio mezzi di trasporto, strutture, ponti ed edifici la cui osservazione può prevenire ingenti danni.
Questo tipo di analisi è utilizzato anche in ambito medico, riscontrando la presenza di patologie e talvolta potendo così monitorare continuamente la salute del paziente.
Infine il settore in cui viene più impiegata un'indagine di questo tipo è l'ambito aziendale. Infatti l'obiettivo è prevenire guasti applicando sensori che monitorino gli oggetti di interesse per evitare che si verifichino rotture e conseguenti blocchi della produzione.
Ci sono due tipi di assunzioni che si fanno in questo tipo di analisi:

1) anomalie sono fenomeni che si verificano solo raramente

2) le caratteristiche dei dati anomali sono diverse rispetto a quelle delle dei dati normali

E' sempre più comune l'utilizzo di modelli di Deep Learning per l'individuazione delle anomalie rispetto a metodi di statistica tradizionale. Questo è dovuto principalmente dalle complesse strutture di dati che si cercano e inoltre dall'utilizzo di dataset di dimensioni elevate, che rende impossibile ottenere buone prestazioni per mezzo di metodi di statistica tradizionale. Questi ultimi metodi inoltre risultano inadeguati soprattutto nella ricerca delle anomalie attraverso le immagini, come ad esempio di tipo medico per la ricerca di patologie, e nelle sequenze temporali in quanto spesso sono composte da complicate strutture [3].

La scelta del modello appropriato di Deep Learning da utilizzare per un'analisi di questo tipo è fondamentale e dipende principalmente dalle caratteristiche dei dati, delle anomalie e dell'obiettivo dello statistico. Nei paragrafi successivi sono illustrate le caratteristiche da tenere in considerazione prima di scegliere un modello per l'individuazione di anomalie.

Tipo di anomalia

Esistono tre modi in cui si può verificare un'anomalia:

1) *puntuale*: quando si è in presenza di una osservazione che assume valori distanti da tutti quanti gli altri presenti nel dataset;

2) *contestuale*: quando un record non presenta valori anomali in assoluto, cioè considerando tutti gli elementi del dataset, ma risulta anomalo contestualmente alla posizione in cui si trova;

3) *collettiva*: "Una collezione anomala di punti individuali è detta una anomalia collettiva o di gruppo, in cui ciascuno dei punti individuali in isolamento appare come istanze di dati normali mentre osservato in un gruppo mostra caratteristiche insolite"[3].

Tipologia di dati da analizzare

I dati utilizzati possono essere di tipo sezionale, serie storiche o longitudinali [14].
I primi sono dati in cui le osservazioni vengono raccolte nello stesso periodo di tempo, o senza che le differenze temporali abbiano un impatto sull'analisi, registrando una o più variabili riguardanti soggetti diversi. Generalmente l'analisi è volta a estrapolare le relazioni che sussistono tra le variabili.
Le serie temporali invece sono dati ottenuti osservando un singolo soggetto in istanti di tempo differenti. Tra le molteplici analisi che si possono svolgere, la principale è la previsione.
I dati longitudinali infine possono considerarsi una generalizzazione dei due precedenti in quanto sono registrate osservazioni di soggetti diversi e il tempo svolge un ruolo importante

Disponibilità etichette

Un ruolo chiave nella scelta del modello lo svolge la presenza di etichette nel dataset. Nei problemi di ricerca di anomalie risulta costosa la produzione di valori anomali e talvolta persino impossibile. In funzione quindi del possesso o meno di una variabile categorica caratterizzante lo stato delle osservazioni, che può avere valori binari (anomalia/non-anomalia) ma può anche avere diversi livelli di anomalia, il modello che ne deriva può essere definito a grandi linee in quattro modi.
Un modello si dice *supervisionato* quando in fase di addestramento si dispone di sufficienti etichette per ogni classe. Quasi mai in questo tipo di problemi si ha a che fare con un insieme di dati che presenta queste caratteristiche.
Quando il dataset invece è sbilanciato a favore di dati non-anomali si parla di modello *semi-supervisionato*.
Alcune volte è conveniente invece impostare un algoritmo che in fase di addestramento

utilizzi solo valori non-anomali ma che in fase di validazione di eventuali soglie classificatrici e in fase di test sia in possesso di tutte quante le classi. Questo tipo di modelli si chiamano *reti a una classe*.
Infine gli algoritmi *non-supervisionati* non presentano alcuna etichetta. Questi modelli ancor più degli altri basano la ricerca delle anomalie sull'assunzione che ci siano caratteristiche differenti tra valori sani e anomali. Quindi lo scopo di questi modelli è catturare le proprietà intrinseche dei dati sani oppure utilizzare algoritmi di densità da cui è possibile notare in maniera non supervisionata una separazione nei dati in funzione di metriche di distanza.

Output del modello

I valori restituiti dalla rete possono essere o variabili categoriche, cioè etichette indicanti ad esempio classe sana e uno o più tipi di non-sane, oppure un punteggio di anomalia, ovvero variabili numeriche. Quest'ultimo risulta utile ad esempio in un monitoraggio di sensori quando oltre alla classificazione in anomalo/non-anomalo si è anche interessati a risalire alla causa dell'anomalia e quindi ai sensori che più di tutti hanno deviato dalla loro abituale comportamento. Inoltre il punteggio di anomalia può essere utilizzato in una fase di calibrazione di una o più soglie per arrivare a delle etichette, quindi in questo caso svolge la funzione di un classificatore.

Obiettivo della ricerca

Ci sono principalmente due motivi per svolgere una ricerca di anomalie: riparare e prevenire. Quando si vuole ad esempio riparare un macchinario che ha subito un guasto inatteso si potrebbe essere interessati a risalire alla causa dell'anomalia per effettuare la riparazione. Un altro esempio di ricerca di anomalie di questo tipo può essere la ricerca di frodi bancarie, trovando il danno e sistemando l'operazione. Invece alcune volte si fa un'analisi per prevenire un'anomalia. Questo accade quando si è in presenza di manifestazioni antecedenti all'anomalia che rivelano un comportamento inatteso. Come ad esempio potrebbe avvenire prima di una rottura di un macchinario in cui molto probabilmente potrebbe esserci una finestra di tempo dove i sensori cominciano ad assumere valori differenti a quelli standard.

4.2 Monitoraggio delle condizioni strutturali (SHM)

Uno dei compiti degli ingegneri è ideare e realizzare un progetto o costruzione che, oltre a soddisfare delle esigenze tipiche del settore, abbia una lunga durata e resista alle peggiori condizioni e lo faccia al minor costo possibile.
Tuttavia può capitare che in fase di produzione si fabbrichino oggetti con errori infinitesimali che non riescono ad essere identificati inizialmente ma che incidono sui tempo di vita o sulla sicurezza a lungo termine del progetto. Questo problema rende impossibile quindi pensare di prevedere la durata di una costruzione all'inizio e procedere alla manutenzione al tempo stabilito senza effettuare un monitoraggio periodico della costruzione.
Se l'utilità di usare un sistema di monitoraggio risulta evidente per costruzioni la cui sicurezza ricade sulla salute delle persone, altrettanto risulta per rilevare guasti in macchinari o strutture di vario genere. Infatti, sebbene sia necessario sostenere costi superiori al dovuto per un'analisi di questo genere, si ottengono vantaggi a livello di costo rispetto a non applicarla:

1) costi diretti della rottura: molto spesso accade che la rottura di un macchinario comporti perdite ingenti, evitabili per mezzo di un controllo effettuato e una successiva manutenzione;

2) costi indiretti della rottura: tra le conseguenze di una rottura la principale è la sospensione della produzione che ricade su costi opportunità. Quest'ultimo alcune volte può essere evitato o mitigato attraverso il monitoraggio. Infatti in quest'ultimo caso è possibile decidere quando sospendere la produzione, cercando quindi di far coincidere la riparazione a momenti in cui l'azienda risulta inoperosa, annullando completamente o mitigando quindi i costi opportunità.

Test distruttivi e test non distruttivi

Per testare lo stato di salute di un oggetto si utilizzano due tipi di test: distruttivi e non distruttivi.
I *test distruttivi* avvengono modificando o distruggendo un pezzo di costruzione su cui si sta svolgendo l'analisi per verificare lo stato di salute dello stesso. Malgrado la loro affidabilità, in quanto si può osservare esplicitamente lo stato del componente, risultano dispendiosi, poiché gli oggetti esaminati non risultano più utilizzabili o presentano danni da deterioramento, e tuttavia in alcuni contesti non sono applicabili a causa di possibili violazioni della sicurezza. Rientrano in questa tipologia di test ad esempio la prova di trazione, la rottura da stress, il test di scorrimento e il test della durezza.
I *test non distruttivi* invece utilizzano un diverso approccio, monitorando i componenti senza alterarne la loro struttura. I vantaggi principali di questi controlli rispetto ai precedenti sono:

- risparmio dei costi dei materiali: in questo tipo di test non vengono deteriorati e quindi sono completamente utilizzabili;

- risparmio di tempo: poiché i test distruttivi necessitano di effettuare passaggi manuali che richiedono tempo e sforzo maggiore rispetto ai processi semplificati che si hanno nei test non distruttivi;
- sicurezza: si immagini ad esempio l'analisi delle fondamenta di una struttura. Un test distruttivo potrebbe portare a conseguenze immediate o future, evitabili utilizzando un sistema di monitoraggio non distruttivo che non avrebbe effetti sulla condizione delle fondamenta;
- perdita informazioni: alcune volte la ricerca in maniera distruttiva di difetti nascosti produce una perdita di indicazioni, specialmente nel caso di grandi infrastrutture dove si perdono informazioni riguardanti forme di progettazione più grandi.

Passaggi del monitoraggio delle condizioni strutturali

Il campo dell'ingegneria che si occupa di monitorare nel tempo componenti o costruzioni è chiamato *Monitoraggio delle condizioni strutturali* e può essere riassunto in tre passaggi chiave [12]:

1) selezionare i punti critici per apporre sopra una rete di sensori. Questo viene fatto simulando e identificando le aree sensibili e quindi utili a fornire informazioni. Inoltre in questa fase si specificano il tipo di sensori da utilizzare, l'ambiente in cui devono operare, la dimensione massima che devono avere e la precisione delle misurazioni che effettuano;
2) la seconda fase consiste nell'ottenimento delle misurazioni dai sensori. Risulta fondamentale in questo passaggio essere a conoscenza dei carichi applicati alle strutture e delle altre caratteristiche che qualora trascurate potrebbero portare a misurazioni inadeguate;
3) infine l'ultimo passaggio consiste nel confronto tra le misurazioni effettuate dai sensori con lo standard previsto per quel sensore.

Sistemi di monitoraggio dell'integrità strutturale

Per controllare l'integrità di una costruzione è necessario servirsi di una rete di sensori. I più utilizzai sono: reti in fibra ottica e emissioni acustiche.
Le *reti in fibra ottica* sono composte da sensori in fibra ottica che sono considerati la principale strumentazione adibita al monitoraggio, grazie al loro peso inferiore rispetto ad altre tecnologie che li rende quindi più convenienti ad esempio quando devono ricoprire una superficie vasta o quando sono disposti su dei veicoli. Requisiti fondamentali per una ragionevole costruzione sono la velocità nel rilevare l'informazione, l'accuratezza nella trasmissione all'unità di acquisizione dati e l'indipendenza del trasferimento dei dati da fattori esterni come radiazioni elettromagnetiche.
Le *emissioni acustiche* riguardano la formazione di onde elastiche transitorie generate attraverso la ridistribuzione del campo di stress. Infatti quando una struttura subisce un'alterazione vengono generati degli spostamenti che inducono un rilascio di energia

sotto forma di onde che si propagano in superficie, dove si ottengono le misurazioni per mezzo di sensori applicati.

4.3 Caso d'uso

Un materiale composito è un sistema costituito da due o più fasi, costruito in modo tale che la combinazione dei materiali abbia proprietà e prestazioni superiori a quelle che avrebbero le singole componenti indipendentemente.
Questo tipo di materiale si rivela un'ottima soluzione nel settore automobilistico grazie al suo peso leggero che permette di ridurre le emissioni dei veicoli, senza tuttavia rinunciare alla robustezza e all'integrità strutturale. In particolare i materiali fibro-rinforzati a matrice polimerica (FRP) rappresentano un'alternativa valida ai metalli ad alta resistenza. Questi materiali però sono soggetti a deterioramenti progressivi, dovuti dal ciclo di fatica o da piccoli eventi di impatto energetico, che portano alla degradazione di moduli elastici e delle resistenze. Per evitare un guasto imprevisto dei componenti strutturali si ricorre a metodi economici per monitorare continuamente lo stato di salute del materiale composito. Poiché esiste una relazione ben nota tra la riduzione del modulo elastico e la quantità di danno, la degradazione della rigidezza può essere monitorata per rilevare la posizione e la quantità di danno che ha avuto luogo [5]. Questo tipo di controllo corrisponde a quel che precedentemente è stato definito Sistema di monitoraggio delle condizioni strutturali (SHM). Christian Boller delinea cinque livelli di SHM [1]: monitoraggio del carico, identificazione del danno, individuazione del luogo in cui il guasto è avvenuto, dimensioni e gravità del danneggiamento.
Il lavoro svolto in questa tesi si riferisce al secondo punto, ovvero la ricerca del danno che viene effettuata per mezzo di reti neurali artificiali.
I materiali compositi permettono che vengano applicati in-situ sensori che possono essere nascosti nei materiali, avendo così il vantaggio di venir protetti da agenti esterni e di ricevere informazioni dal nucleo del materiale. Vista la provata validità dell'applicazione di fibre ottiche nei materiali compositi, in quest'analisi sono stati considerati dati provenienti da sensori a reticolo di Bragg in fibra, utili per la loro capacità a ottenere misurazioni della deformazioni assiali con una precisione di $1\mu\epsilon$.
In particolare in questo lavoro sono stati analizzati dati provenienti da simulazioni di sensori a reticolo di Bragg in fibra che vengono posti su alcuni braccetti delle sospensioni di un veicolo. I dati sono raccolti con una frequenza di 3 Hz, riproducendo il tragitto di un veicolo su un percorso che prevede la presenza di accelerazioni longitudinali e laterali di varia natura. I dati sono raccolti prima simulando un veicolo completamente sano e poi vengono ripetute due altri scenari, ciascuno caratterizzato da un danno in un diverso punto del veicolo.
Per il caso d'uso sono stati utilizzati due differenti modelli che, sebbene utilizzino entrambi la comune idea di identificare il danno per mezzo di ricostruzioni avvenute male, si differenziano nel modo in cui queste ricostruzioni vengono prodotte. Il primo modello infatti restaura i dati stessi attraverso l'applicazione di una rete neurale. Il secondo invece fonda la sua validità sull'assunzione che il danno può essere individuato nei materiali compositi monitorando le correlazioni tra le misure dei sensori [7].

Capitolo 5

Analisi dataset e risultati

5.1 Statistica descrittiva dei dataset

Nel caso d'uso si utilizza un dataset composto dalla concatenazione di 3 serie temporali multivariate aventi lo stesso numero di variabili e la stessa cardinalità. Si ha quindi un insieme di dati per la classe sana (classe 0), anomalia di tipo 1 (classe 1) e anomalia di tipo 2 (classe 2).
Il dataset corrisponde quindi ad una matrice di dimensioni 31x210576, dove la prima dimensione indica il numero di variabili e la seconda invece il numero di istanti di tempo che definiscono la serie temporale, ovvero tre serie da 70192 istanti. Delle 31 variabili, una è la variabile risposta y che indica lo stato del veicolo, cioè l'etichetta che prende valori in 0,1,2, e le rimanenti 30 sono variabili quantitative che misurano la variazione di lunghezza dei sensori rispetto a quella che si osserva a riposo.
Nella tabella che segue sono illustrate le statistiche principali suddivise per ogni classe per evidenziare comportamenti differenti. Sono mostrate quindi la media, la deviazione standard e la moda per ogni classe.
Poiché i sensori rilevano variazioni relative di lieve entità, nella tabella sono riportate le statistiche ognuna moltiplicata per 10^5, per facilitare il confronto.

	Classe 0			Classe 1			Classe 2		
	media	std	moda	media	std	moda	media	std	moda
Var1	-9.13	10.12	-1.71	1.41	1.25	0.84	0.66	0.75	0.19
Var2	20.32	19.41	5.73	-3.13	14.97	-2.71	-3.79	5.53	-0.71
Var3	0.03	1.60	-0.51	1.61	5.33	0.33	0.57	1.15	0.08
Var4	-0.16	2.12	-0.73	1.85	5.27	0.45	0.74	1.40	0.11
Var5	15.62	12.82	5.18	-6.84	7.79	-2.75	-2.22	4.88	-0.66
Var6	16.18	13.92	5.00	-6.63	7.50	-2.70	-2.07	4.81	-0.63
Var7	-1.20	2.71	-1.32	2.60	5.60	0.80	1.09	1.59	0.19
Var8	14.36	14.53	3.60	-4.97	6.75	-1.92	-1.17	4.69	-0.44
Var9	-1.68	2.74	-1.54	1.41	5.71	0.94	1.15	1.38	0.22
Var10	-1.89	2.52	-1.53	2.85	5.54	0.93	1.12	1.32	0.21
Var11	11.27	12.18	2.44	-2.84	3.13	-1.23	-0.80	2.82	-0.29
Var12	10.74	11.63	2.32	-2.68	2.93	-1.17	0.77	2.65	-0.28
Var13	10.26	11.07	2.23	-2.55	2.74	-1.12	-0.76	2.48	-0.27
Var14	-2.43	1.64	-1.29	2.37	4.92	0.75	0.80	0.78	0.17
Var15	7.40	7.68	1.70	-1.75	2.20	-0.88	-0.69	1.47	-0.21
Var16	6.03	6.40	1.36	-1.51	1.66	-0.70	-0.51	1.41	-0.17
Var17	5.59	6.16	1.20	-1.43	1.61	-0.61	-0.39	1.51	-0.15
Var18	4.33	6.75	0.46	-1.26	5.39	-0.18	0.25	2.58	-0.05
Var19	-2.02	2.61	-0.98	0.19	6.34	0.55	0.80	1.56	0.13
Var20	0.51	7.81	-1.37	0.10	11.33	0.86	1.58	4.33	0.19
Var21	-2.05	1.90	-0.80	0.38	3.58	0.44	0.67	1.06	0.12
Var22	2.92	5.07	0.15	-0.88	5.57	-0.02	0.44	1.95	0.02
Var23	1.42	1.27	0.41	-0.56	0.73	-0.22	-0.44	0.44	-0.10
Var24	2.81	2.33	0.87	-1.08	1.07	-0.48	-0.50	0.55	-0.12
Var25	2.32	1.90	0.73	-0.86	0.84	-0.39	-0.43	0.44	-0.10
Var26	1.04	1.52	0.09	-0.17	1.02	-0.03	0.08	0.39	0.00
Var27	2.06	1.70	0.64	-0.73	0.73	-0.35	-0.36	0.37	-0.09
Var28	-0.47	0.41	-0.15	0.24	0.36	0.08	0.03	0.14	0.01
Var29	2.13	1.86	0.62	-0.65	0.71	-0.33	-0.30	0.32	-0.07
Var30	-0.64	0.57	-0.18	0.23	0.26	0.09	0.05	0.13	0.02

Tabella 5.1: Statistiche riassuntive dati caso d'uso

A causa dell'elevato numero di variabili risulta difficile scorgere differenze significative nei dati rispetto a ciascuna classe.
Può essere utile mostrare le matrici di correlazione suddividendo per classi l'analisi, indagando così se differenti tipi di classi rispecchiano un comportamento differente nelle correlazioni tra i sensori o meno.

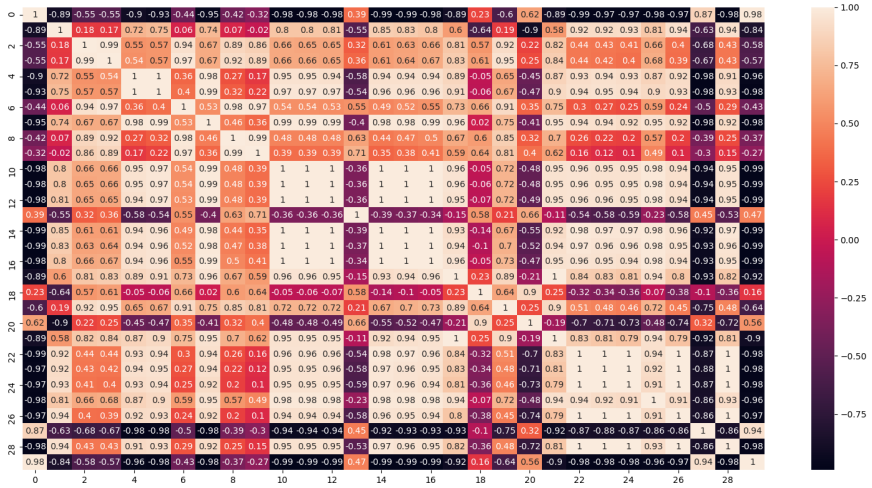

Figura 5.1: Matrice di correlazione classe 0

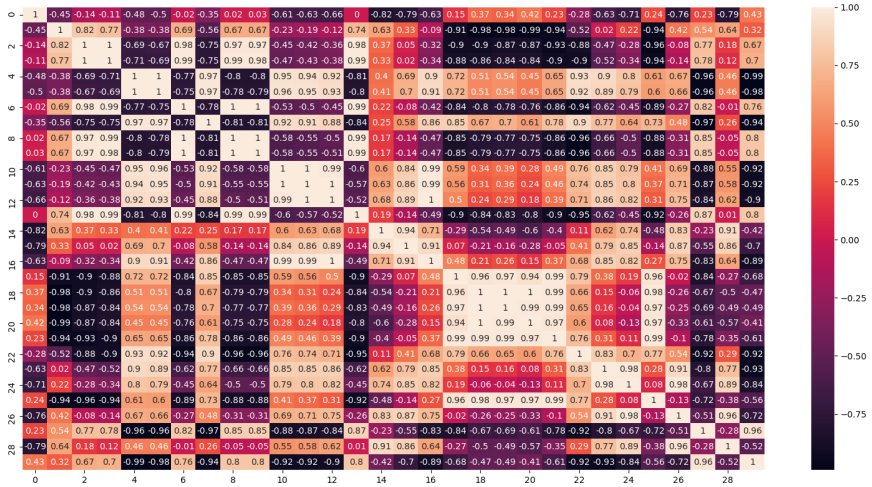

Figura 5.2: Matrice di correlazione classe 1

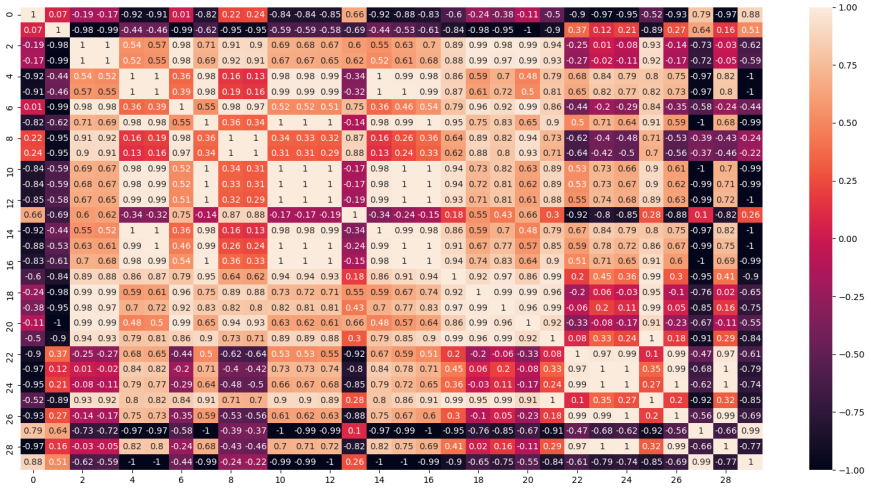

Figura 5.3: Matrice di correlazione classe 2

Da queste matrici si può notare come le correlazioni tra i sensori per l'anomalia di tipo 1 si discosti molto dalle correlazioni calcolate sul dataset di valori sani. La classe 0 infatti presenta generalmente correlazioni positive tra i sensori, nello specifico ad esempio tra i sensori 10-11-12, 14-15-16-17 e 2-23-24-25-26 per cui ci sono valori vicini al massimo che è 1, ma in generale a meno di eccezioni vicino alla diagonale si presentano blocchi più o meno grandi di sensori positivamente correlati.
Per quanto riguarda invece la classe 1 si evidenzia un notevole cambio nelle correlazioni, mantenendo sempre una prevalenza di correlazioni positive lungo i blocchi della diagonale ma introducendo invece forti legami di tipo opposto inattesi da valori appartenenti alla classe sana. Tra i principali blocchi di correlazione negativa si evidenziano i sensori 2-3-4 con 17-18-19-20-21-22 e i sensori 6-8-9 con 17-8-19-20-21-22.
La classe 2 invece non evidenzia valori particolarmente differenti rispetto alla classe 0, supponendo quindi che sia più difficile ottenere una buona separazione di questi valori rispetto ai valori sani nel modello MSCRED il quale fonda la propria efficacia sulla modellazione delle dipendenze simili alle correlazioni lungo finestre mobili appartenenti alle serie temporali. Si suppone a priori quindi che una grande differenza riguardante le correlazioni aiuti il modello ad ottenere una buona classificazione delle anomalie, rendendo quindi più facile la ricerca delle anomalie di tipo 1 rispetto a quelle di tipo 2.

5.2 Metriche e grafici utili alla valutazione di una classificazione binaria

Esistono numerosi metodi per valutare la bontà di una classificazione binaria. Generalmente si utilizzano vari metodi rispetto ad uno singolo in quanto ciascuno aiuta a valutare diversi aspetti della classificazione effettuata.
Per introdurre le metriche utili alla valutazione è necessario definire i seguenti valori con cui si caratterizzano le singole osservazioni predette. Sia y_i la classe vera del record e $\hat{y}_i$ il valore predetto, allora $\hat{y}_i$ è un:

- vero positivo: se $\hat{y}_i = 1$ e $\hat{y}_i = 1$
- falso positivo: se $\hat{y}_i = 1$ e $\hat{y}_i = 0$
- vero negativo: se $\hat{y}_i = 0$ e $\hat{y}_i = 0$
- falso negativo: se $\hat{y}_i = 0$ e $\hat{y}_i = 1$

Questi valori sono utilizzati per calcolare l'accuratezza, una misura che riassume in un unico valore la qualità della classificazione, ovvero tra tutte le predizioni si calcola la percentuale di classificazioni corrette.

Accuratezza

Sia quindi TP il numero di veri positivi, FP il numero di falsi positivi, TN il numero di veri negativi e FN il numero di falsi negativi. L'accuratezza allora, per un dataset di dimensione n, è calcolata come:

$$Accuratezza = \frac{TP + TN}{n} \tag{5.1}$$

Sebbene l'accuratezza sia uno strumento utile, non fornisce informazioni relative alla classificazione delle singole classi. Si riportano quindi altre metriche che mostrano il rendimento del modello su ciascuna classe.

Matrice di confusione

Una matrice di confusione è il modo più intuitivo per valutare una classificazione osservando i risultati ottenuti su entrambe le classi. Nel caso di una classificazione binaria consiste in una matrice di dimensioni 2x2 definita come:

	Valori predetti classe 0	Valori predetti classe 1
Valori veri classe 0	TN	FP
Valori veri classe 1	FN	TP

Tabella 5.2: Esempio matrice di confusione per classificazione binaria

Questo strumento mette in evidenza i valori classificati correttamente e quelli classificati in maniera errata, ciascuno con le rispettive quantità.
La matrice di confusione può essere riformulata normalizzando i valori sulle righe, ottenendo quindi la percentuale di classificazioni corrette e sbagliate per entrambe le classi.

Richiamo

Per ogni classe è utile inoltre porre attenzione sul richiamo, anche chiamato recupero.
Questa metrica indica la percentuale di osservazioni classificate correttamente per ogni classe rispetto al numero totale di record appartenenti a quella classe, ovvero coincide con i valori sulla diagonale di una matrice di confusione normalizzata rispetto ai valori veri.
La formula per il richiamo definita per entrambe le classi è:

$$Richiamo_{classe1} = \frac{TP}{TP + FN} \tag{5.2}$$

$$Richiamo_{classe0} = \frac{TN}{TN + FP} \tag{5.3}$$

Precisione

Per entrambe le classi inoltre è possibile calcolare la precisione. Questa misura indica la percentuale di classificazioni corrette, della classe per cui si calcola, rispetto alle predizioni totali avvenute nei confronti di quella classe. Quindi:

$$Precisione_{classe1} = \frac{TP}{TP + FP} \tag{5.4}$$

$$Precisione_{classe0} = \frac{TN}{TN + FN} \tag{5.5}$$

La precisione calcolata rispetto alla classe 0 viene anche definita specificità e rappresenta il tasso di veri negativi.
Inoltre quando non è specificata la classe per cui viene calcolata la precisione, si sottintende che sia calcolata rispetto alla classe 1. In questo caso prende il nome di sensibilità e rappresenta il tasso di veri positivi.

Precisione e recupero sono due misure importanti poiché forniscono informazioni dettagliate riguardo alla classificazione rispetto ad ogni classe. Queste due metriche però hanno comportamenti opposti, ovvero accade che all'aumentare di una misura si ha una diminuzione dell'altra.
Per validare un modello quindi è necessario introdurre un'altra metrica che tenga in considerazione sia precisione che recupero, cercando un compromesso tra i due.

F_1 score

L'F_1 score, anche chiamata misura F, è una media armonica tra precisione e recupero.
E' preferita la media armonica a quella aritmetica in quanto la prima attribuisce un maggior peso ai valori piccoli di precisione e recupero, evitando quindi di ottenere un valore alto della misura F anche nel caso in cui una metrica tra precisione e richiamo è alta e l'altra molto bassa. Questo caso porterebbe ad una buona media aritmetica ma ad una pessima media armonica. Utilizzando la media armonica quindi si ottiene un compromesso più equilibrato.
Definiti p la precisione e r il richiamo, l'F_1 score è calcolato come:

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}} = 2 \cdot \frac{p \cdot r}{p + r} \tag{5.6}$$

Curva ROC e AUC

Le curve ROC sono un altro strumento che aiuta a trovare il compromesso tra richiamo e precisione. "La curva ROC mostra il compromesso tra richiamo e specificità quando si modifica il limite per determinare come classificare un record" [2].
Per ottenere una curva ROC si eseguono due operazioni:

1) "ordinare i record in base alla probabilità prevista di essere 1"[2], partendo dal meno probabile e finendo col più probabile

2) calcolare la specificità e il richiamo cumulativi in base ai record ordinati

Infine si rappresentano questi due valori cumulativi in un grafico dove l'asse delle y indica la sensibilità e l'asse delle x la specificità.
In questo grafico di solito viene anche rappresentata una retta diagonale, rappresentate cioè le prestazioni che ottiene un classificatore random o costante su una classe. Questa rappresentazione è utile perché mette in relazione un classificatore casuale con un altro addestrato.
Per valutare la bontà di una curva ROC attraverso una singola misura è possibile porre l'attenzione sull'area sotto la curva (AUC) del classificatore. Infatti tanto più questo valore compreso tra 0 e 1 è alto, tanto migliore sarà il classificatore. Considerando che un classificatore casuale ha un AUC equivalente a 0.5, si considerano come buoni i classificatori aventi AUC > 0.8 circa.

5.3 Risultati ottenuti

Entrambi i modelli sono stati analizzati attraverso un dataset composto da 70192 osservazioni per la classe 0, 15128 record per la classe 1 e 15128 per la classe 2.
Queste dimensioni sono giustificate dal fatto che si vuole comparare i due modelli sui medesimi istanti di tempo. Infatti per ottenere una buona separazione tra le parti il secondo modello necessita di scartare 64 osservazioni ogni volta che si estrae un sottoinsieme dal dataset.
Entrambi i modelli vengono addestrati per mezzo di osservazioni appartenenti alla classe 0. Tuttavia in fase di validazione delle soglie classificatrici e di test vengono utilizzati anche i record appartenenti alle classi rimanenti. Risulta interessante quindi considerare sia come i modelli riescano a separare i valori sani da quelli anomali, caso che contrappone la classe 0 alle altre due, sia capire se le reti riescono a riconoscere più un'anomalia di un tipo rispetto che di un altro. I casi analizzati dunque sono:

1) classe 0 - classe 1 - classe 2
2) classe 0 - classe 1
3) classe 0 - classe 2

La figura 5.4 fornisce un'idea qualitativa di come sia stato utilizzato il dataset nei vari casi. Per quanto riguarda il secondo e terzo caso, il dataset di validazione e di test dei valori sani risultano dimezzati rispetto al primo caso, per fare in modo che il dataset rimanga bilanciato come avviene nel primo caso.

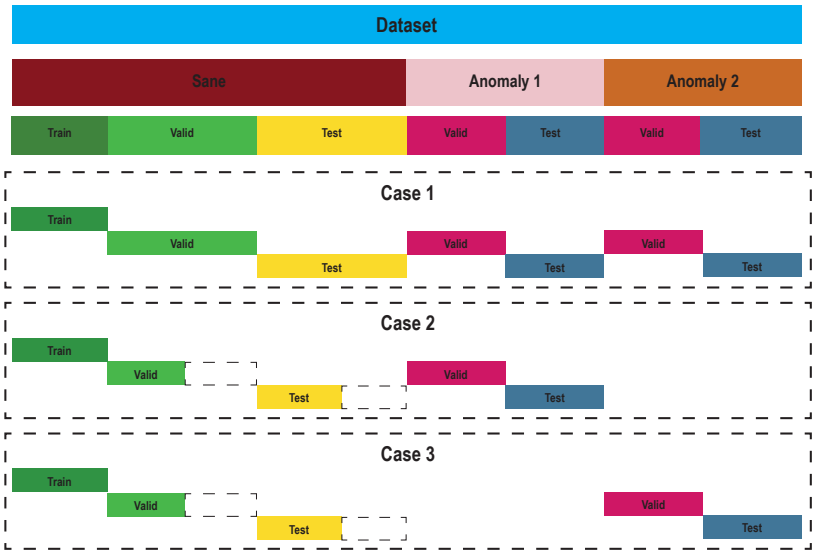

Figura 5.4: Suddivisione dataset nel caso d'uso

5.3.1 Risultati LSTM Encoder-Decoder

Il primo modello è stato addestrato attraverso l'utilizzo di 28000 record. Per scegliere il modello migliore è stata utilizzata la tecnica dell'early-stopping, stabilendo come criterio di arresto la minimizzazione dell'errore su un dataset di validazione, composto da 12000 osservazioni. Come ottimizzatore è stato utilizzato Adam, variante dell'algoritmo SGD che sfrutta metodi adattivi del tasso di apprendimento per ottenerne uno individuale per ogni parametro.

Gli iperparametri utilizzati nell'addestramento del modello sono:
- batch = 50
- numero iterazioni = 50
- tasso di apprendimento = 0.0001
- tasso di decadimento esponenziale = 0.96

Siccome lo scopo dell'analisi è trovare un classificatore binario che predica se un'osservazione è considerata anomala o meno, è necessario riformulare le etichette y, appartenenti alle classi {0,1,2}, in $\tilde{y}$ dove

$$\tilde{y}^{(t)} = \begin{cases} 0 & \text{se } y^{(t)} = 0 \\ 1 & \text{se } y^{(t)} = \{1,2\} \end{cases}$$

che sono le variabili risposta che effettivamente vengono confrontate con $\hat{y}$, ovvero le etichette predette.

1) classe 0 - classe 1 - classe 2

Per l'analisi del modello utilizzando entrambi i tipi di anomalie in fase di calibrazione e test, per avere un dataset bilanciato sono stati utilizzati in fase di validazione 10000 campioni della classe 0 e 10000 equamente suddivisi tra classe 1 e 2. Per la fase di test invece sono stati considerati 20000 record appartenenti alla classe 0 e ulteriori 20000 suddivisi in parti uguali tra classe 1 e 2.
Per prima cosa risulta utile illustrare la matrice di confusione normalizzata rispetto alle vere etichette, in quanto fornisce un'indicazione immediata, anche se non dettagliata, di come siano state classificate le osservazioni:

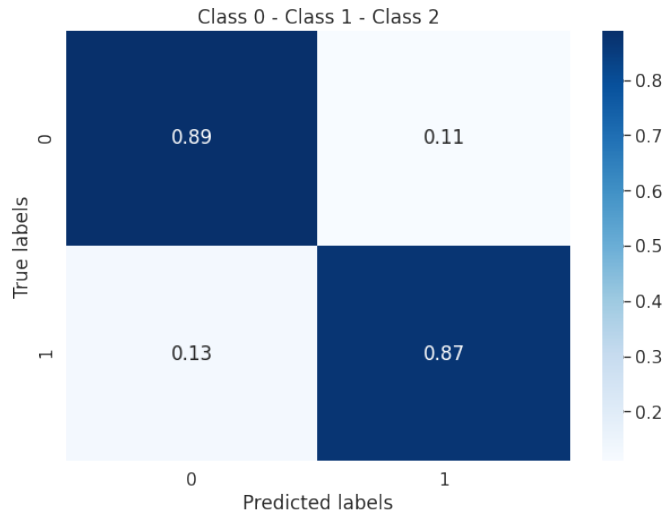

Figura 5.5: Matrice di confusione con modello LSTM-ED (caso 1)

Con un'accuratezza dell' 88% il modello, calibrato considerando tutte e due i tipi di anomalie, risulta un buon classificatore. Per un'analisi più accurata è riportata di seguito una tabella che include le statistiche più utilizzate per valutare la bontà di un classificatore di tipo binario:

	Precisione	Richiamo	F_1 score
Classe 0	0.87	0.89	0.88
Classe 1 - Classe 2	0.89	0.87	0.88

Tabella 5.3: Statistiche per la valutazione della classificazione utilizzando il modello LSTM-ED per il caso 1

Dalla tabella emergono risultati equilibrati. Il richiamo risulta inferiore per la classe 0, sebbene sia preferibile il contrario, ovvero ottenere un alto richiamo delle anomalie a discapito di falsi positivi.
Un ulteriore strumento per valutare i risultati ottenuti è il grafico della curva ROC, che mostra l'andamento del tasso di veri positivi rispetto a quello dei falsi positivi.

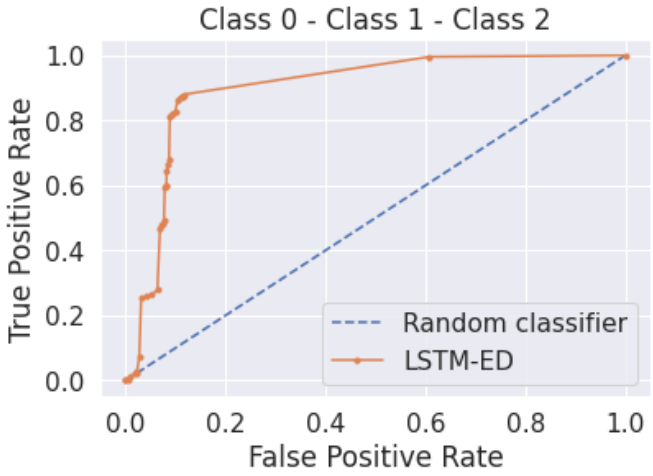

Figura 5.6: Curva ROC utilizzando LSTM-ED (caso 1)

Dalla curva ROC si può ricavare un risultato che identifica la bontà del classificatore, ovvero l'area sotto la curva, che in questo caso equivale al 90%. E' un buon risultato se paragonato ad un classificatore random la cui area sotto la curava sarebbe stata uguale al 50%.
Infine sono rappresentate le distribuzioni dei punteggi di anomalia, normalizzate rispetto alla cardinalità della classe nel dataset di test, utili per vedere da una diversa prospettiva la bontà del modello. I valori anomali corrispondono agli istogrammi blu mentre quelli rossi e marroni corrispondono rispettivamente alle anomalie di tipo 1 e di tipo 2.

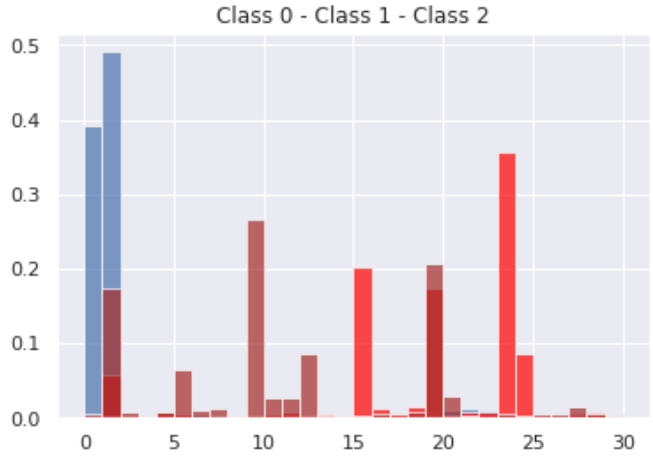

Figura 5.7: Distribuzioni punteggi di anomalia con LSTM-ED (caso 1)

Dalla Figura 5.22 emerge una separazione tra le classi, a conferma dei buoni risultati illustrati precedentemente in tabella.

2) classe 0 - classe 1

Può essere interessante fare un'analisi per capire quale danno venga rilevato più facilmente tra il primo e il secondo.
In questa analisi si utilizzano in fase di validazione delle soglie 5000 campioni per entrambe le classi, mentre nella fase di test 10000.
La matrice di confusione normalizzata rispetto ai veri valori risulta:

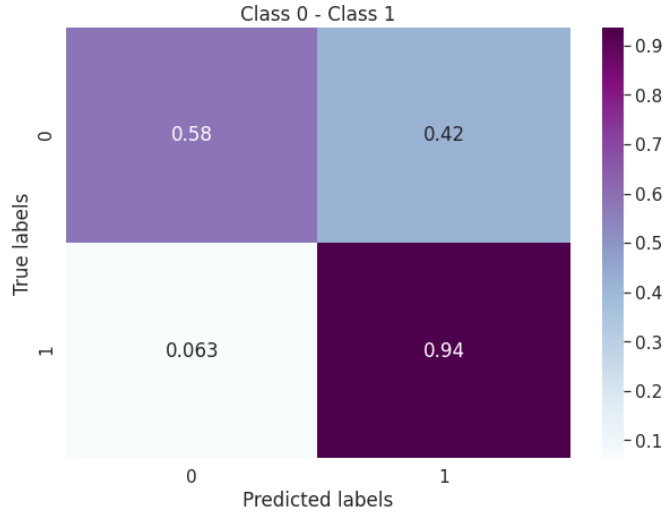

Figura 5.8: Matrice di confusione con modello LSTM-ED (caso 2)

L'accuratezza in questo caso è del 76%. Infatti sebbene sia alto il valore di richiamo per la classe 1, il recupero per la classe 0 risulta decisamente basso. Quindi il modello ha scelto una soglia troppo bassa in modo tale da ottenere molti falsi positivi, quasi la metà della classe 0.
La precisione, il recupero e la misura F sono riportati nella seguente tabella:

	Precisione	Richiamo	F_1 score
Classe 0	0.90	0.58	0.71
Classe 1	0.69	0.94	0.80

Tabella 5.4: Statistiche per la valutazione della classificazione utilizzando il modello LSTM-ED per il caso 2

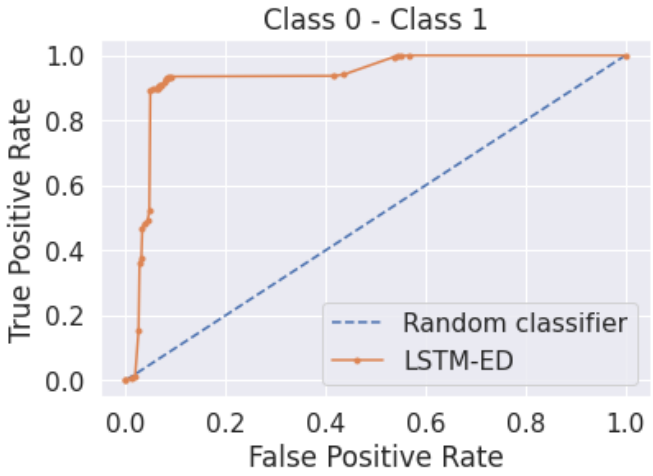

Figura 5.9: Curva ROC utilizzando LSTM-ED (caso 2)

In questo caso si ottiene una curva ROC con un'area sotto la curva equivalente al 93%. Infine si illustrano le distribuzioni del punteggio di anomalia sia per la classe 0 che per la classe 1:

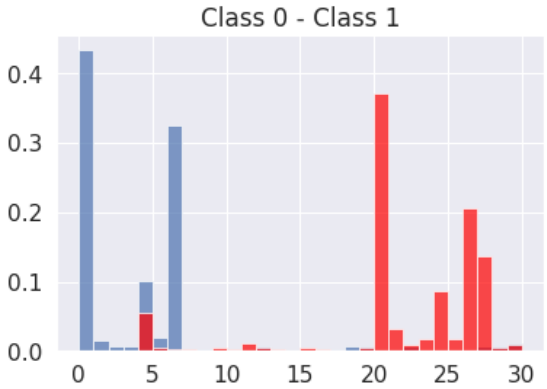

Figura 5.10: Distribuzioni punteggi di anomalia con LSTM-ED (caso 2)

Anche in questo caso le distribuzioni risultano ben separate. I valori bassi di precisione del richiamo per la classe 0 e di precisione per la classe 1 derivano dal fatto che il modello in fase di calibrazione ha selezionato una soglia troppo bassa.

3) classe 0 - classe 2

Il terzo caso è stato analizzato utilizzando, come per il secondo caso, 10000 campioni della classe 2 e gli stessi 10000 della classe 0 utilizzati nel precedente confronto.
La matrice di confusione normalizzata rispetto alle vere etichette risulta:

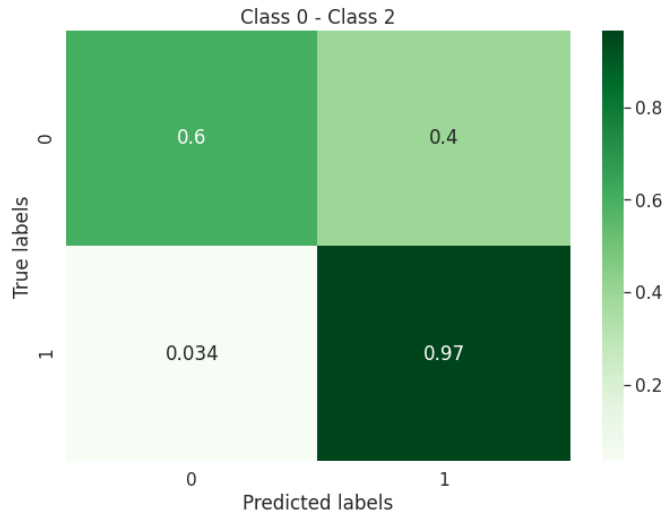

Figura 5.11: Matrice di confusione con modello LSTM-ED (caso 3)

L'accuratezza risulta del 78%. Anche in questo caso, come nel precedente, il modello classifica bene i valori anomali a discapito di ottenere un numero elevato di falsi positivi. E' riportata sotto la tabella con le statistiche utili per la classificazione per il terzo caso:

	Precisione	Richiamo	F_1 score
Classe 0	0.95	0.60	0.74
Classe 2	0.71	0.97	0.82

Tabella 5.5: Statistiche per la valutazione della classificazione utilizzando il modello LSTM-ED per il caso 3

Si può notare dalla tabella che sia il valore di richiamo per la classe 0 che per la classe 2 risultano maggiori rispetto all'analisi precedente, seppur di poco.

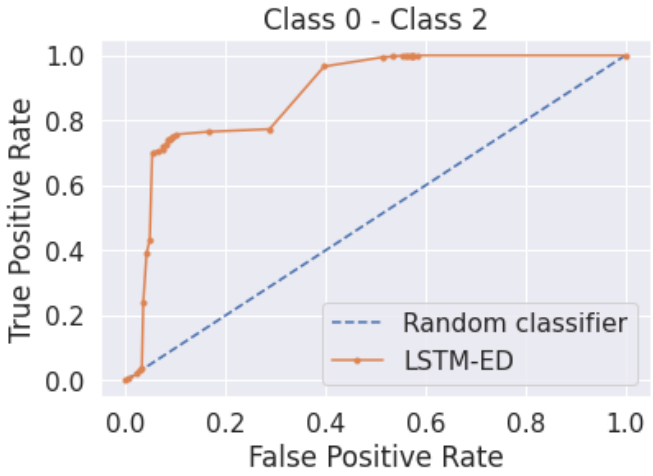

Figura 5.12: Curva ROC utilizzando LSTM-ED (caso 3)

Per questa analisi l'area calcolata sotto la curva vale 88%.
Infine si illustrano le distribuzioni del punteggio di anomalia sia per la classe 0 che per la classe 2:

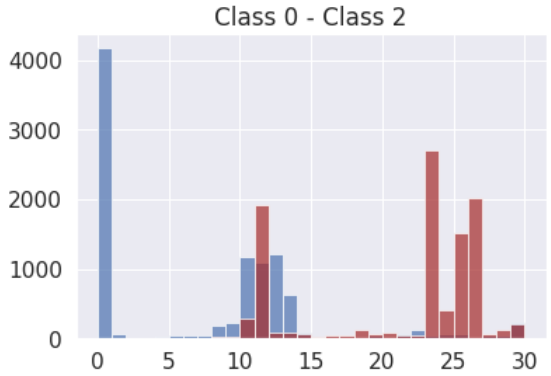

Figura 5.13: Distribuzioni punteggi di anomalia con LSTM-ED (caso 3)

Risulta anche in questo caso evidente una separazione tra le classi, sebbene meno accentuata rispetto al caso 2.

5.3.2 Risultati MSCRED

Il secondo modello è stato addestrato utilizzando 28000 record. Per scegliere il modello migliore anche in questo caso è stata utilizzata la tecnica dell'early-stopping, stabilendo come criterio di arresto la minimizzazione dell'errore su un dataset di validazione, composto da 12000 osservazioni. Anche in questo caso come ottimizzatore è stato scelto Adam. Gli iperparametri utilizzati in questo modello sono:
- batch = 1
- numero iterazioni 10
- tasso di apprendimento = 0.00001

Come per il precedente modello, siccome l'obiettivo è trovare un classificatore binario, è necessario riformulare le etichette in modo che siano utili per all'analisi seguente. Quindi le variabili y che indicano la classe di appartenenza dei record vengono sostituite nell'analisi da $\tilde{y}$ dove:

$$\tilde{y}^{(t)} = \begin{cases} 0 & \text{se } y^{(t)} = 0 \\ 1 & \text{se } y^{(t)} = \{1,2\} \end{cases}$$

che sono le variabili risposta che effettivamente vengono confrontate con $\hat{y}$, ovvero le etichette predette.

1) classe 0 - classe 1 - classe 2

Similmente all'analisi precedente, per avere un dataset bilanciato sono stati utilizzati in fase di validazione 10000 campioni della classe 0 e 10000 equamente suddivisi tra classe 1 e 2. Per la fase di test invece sono stati considerati 20000 record appartenenti alla classe 0 e ulteriori 20000 suddivisi in parti uguali tra classe 1 e 2.

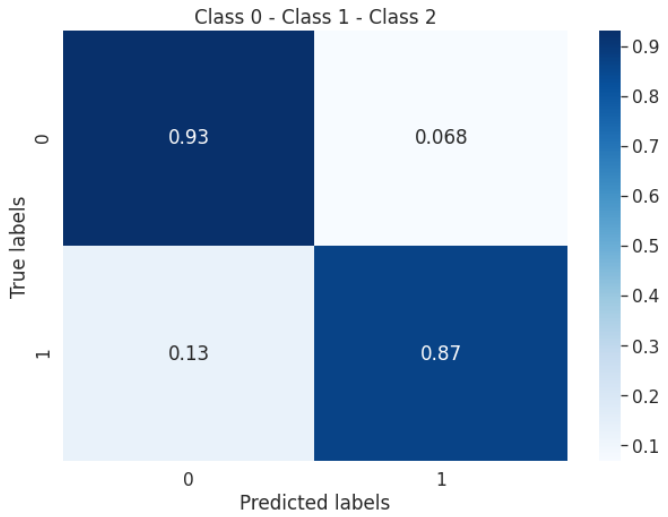

Figura 5.14: Matrice di confusione con modello MSCRED (caso 1)

Il modello ottiene un'accuratezza del 90%. Si può notare come per il caso 1 tenda a classificare meglio la classe 0 rispetto a quelle anomale.

	Precisione	Richiamo	F_1 score
Classe 0	0.88	0.93	0.90
Classe 1 - Classe 2	0.93	0.87	0.90

Tabella 5.6: Statistiche per la valutazione della classificazione utilizzando il modello MSCRED per il caso 1

La classificazione in questo caso risulta bilanciata, ottenendo un valore minimo tra richiamo e precisione dell'87% considerando entrambe le classi.

Figura 5.15: Curva ROC utilizzando MSCRED (caso 1)

Il modello MSCRED calibrato considerando tutte le classi presenta un'area sotto la curva del 93%.

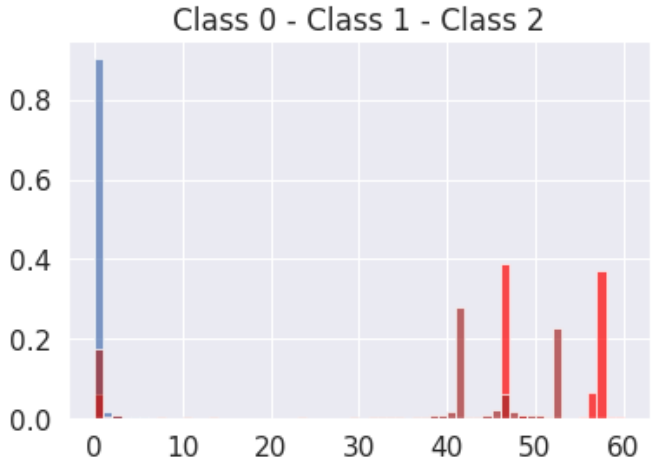

Figura 5.16: Distribuzioni punteggi di anomalia con MSCRED (caso 1)

Dal grafico delle distribuzioni infine si nota una evidente separazione tra classe sana, che assume quasi sempre il valore 0, e le classi anomale.

2) classe 0 - classe 1

In questo caso vengono utilizzate gli stessi campioni utilizzati precedentemente. Per avere una prima informazione utile a capire la bontà del modello si mostra la matrice di confusione:

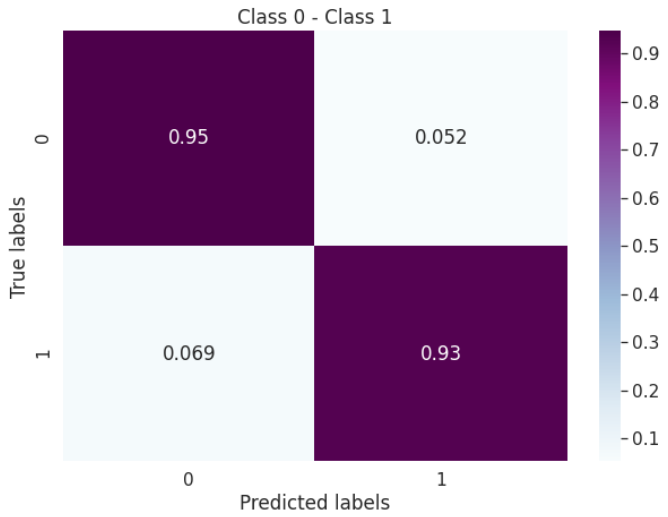

Figura 5.17: Matrice di confusione con modello MSCRED (caso 2)

Il modello raggiunge un'accuratezza del 94%.

	Precisione	Richiamo	F_1 score
Classe 0	0.93	0.95	0.94
Classe 1	0.95	0.93	0.94

Tabella 5.7: Statistiche per la valutazione della classificazione utilizzando il modello MSCRED per il caso 2

Come si può notare in tabella, il modello riconosce molto bene il danno di tipo 1, ottenendo valori di precisione e richiamo sopra il 93%.

Figura 5.18: Curva ROC utilizzando MSCRED (caso 2)

Come prevedibile dalla bontà del modello, si ottiene una buona curva ROC che presenta un'area sotto la curva equivalente a 96%.

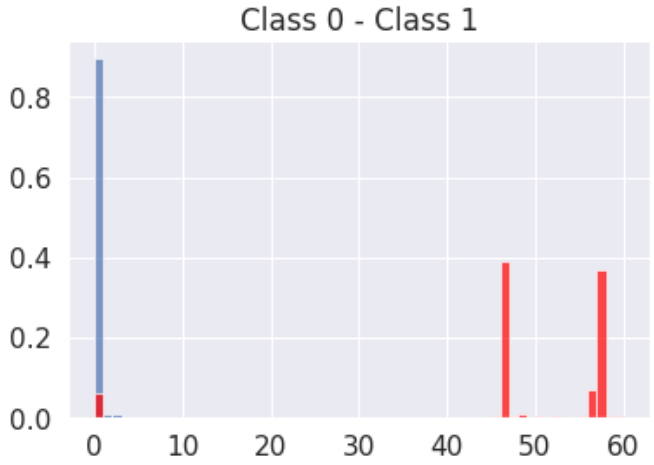

Figura 5.19: Distribuzioni punteggi di anomalia con MSCRED (caso 2)

Infine gli istogrammi confermano una ottima separazione tra e classi. La classe 0 infatti risulta quasi tutta concentrata su un punteggio di anomalia uguale a 0 mentre la classe 1 è quasi tutta concentrata oltre un punteggio di 45, a meno di pochi falsi negativi.

3) classe 0 - classe 2

L'analisi del caso 3 avviene utilizzando gli stessi dataset utilizzati nel modello precedente. La matrice di confusione in questa analisi risulta:

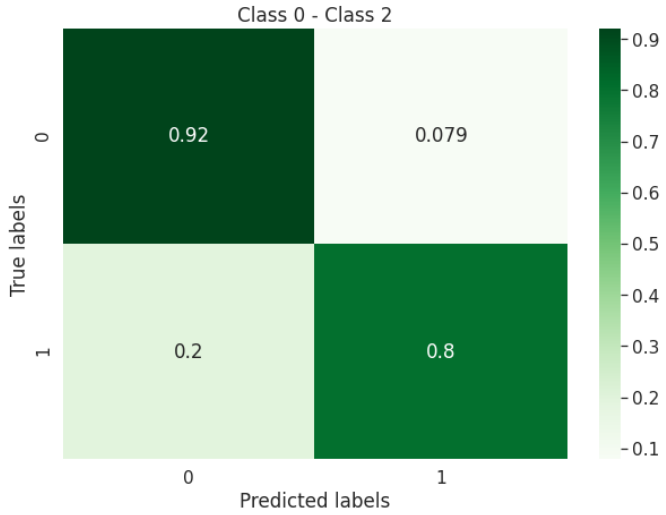

Figura 5.20: Matrice di confusione con modello MSCRED (caso 3)

L'accuratezza ottenuta sul dataset equivale all'86%. Però il richiamo risulta basso per la classe 1, a favore della classe 0 per cui risulta del 92%.

	Precisione	Richiamo	F_1 score
Classe 0	0.83	0.82	0.87
Classe 2	0.91	0.80	0.85

Tabella 5.8: Statistiche per la valutazione della classificazione utilizzando il modello MSCRED per il caso 3

Dalla tabella si nota come la precisione sia alta per la classe 2, questo perché sebbene sia stata rilevata meno la classe 2 rispetto alla classe 0, i valori identificati differiscono nettamente dagli altri della classe 0.

Figura 5.21: Curva ROC utilizzando MSCRED (caso 3)

La curva ROC ha un'area sotto la curva equivalente a 89%. Sebbene risulti inferiore sia in precisione sia in richiamo rispetto al modello precedente, è considerato un buon risultato.

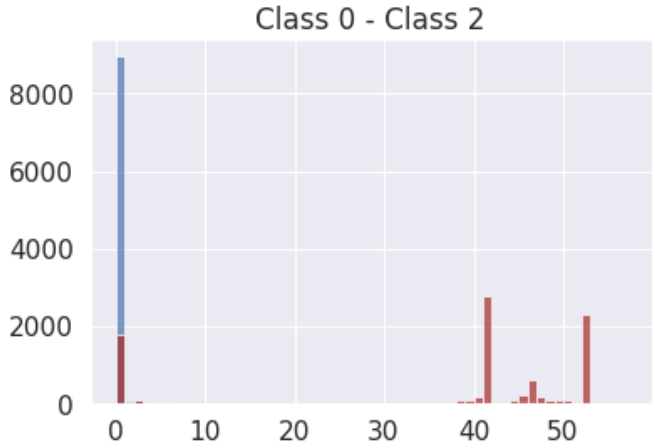

Figura 5.22: Distribuzioni punteggi di anomalia con MSCRED (caso 3)

Dal grafico delle distribuzioni si vede che nonostante sia presente una separazione tra dati sani e anomali, molti record della classe 2 risultano falsi positivi. In particolare se paragonato al caso precedente si intuisce come la rete abbia più difficoltà nel riconoscere l'anomalia di tipo 2, quando analizzati separatamente.

5.4 Confronto modelli

I due modelli utilizzati nell'analisi precedente presentano caratteristiche differenti sia a livello di risultati ottenuti sia a livello di costo computazionale. I tempi di di calcolo per il modello LSTM autoencoder risultano:
- 25 minuti per l'addestramento e convalida del modello utilizzando 50 iterazioni
- 4 minuti per la calibrazione delle soglie
- 2 minuti per la fase di test

Il secondo modello invece consiste in una rete neurale più profonda che utilizza tensori di grandi dimensioni che non sono sparsi. I tempi per questo modello sono:
- 4 ore e mezza per l'addestramento e convalida del modello utilizzando 10 iterazioni
- 4 ore per la fase di calibrazione delle soglie
- 20 minuti per la fase di test

Per confrontare meglio i risultati ottenuti ci si può limitare a riportare i valori dell'F1 score, mantenendo separata l'analisi per tutti e tre i casi studiati.

	LSTM-ED	MSCRED
Caso 1	88.19%	**89.80**%
Caso 2	79.60%	**93.88**%
Caso 3	**88.15**%	85.44%

Tabella 5.9: Confronto modelli LSTM-ED e MSCRED

Il primo modello risulta migliore del secondo nell'analisi del terzo caso, ovvero contrapponendo nella classificazione solamente classe 0 e classe 2.
Il secondo modello utilizzato, MSCRED, risulta migliore quando si svolge un'analisi in cui vuole essere riconosciuto un valore anomalo senza la necessità di indagare la tipologia del danno risulta migliore. Questo modello inoltre raggiunge buone prestazioni soprattutto quando mette a confronto valori normali con anomalie di tipo 1, arrivando ad un valore dell'F_1 score vicino al 94%.

Considerando le prestazioni ottenute, il secondo modello risulta generalmente migliore. Infatti oltre ad avere valore dell'F_1 score più alti in due casi su tre, un aspetto importante è che la sua peggior classificazione ottiene comunque un valore di 85.44% , più alto rispetto al peggior valore del primo modello. In funzione di grandi risorse computazionali si può pensare di utilizzare il secondo modello nel monitoraggio di una struttura a causa delle sue prestazioni superiori rispetto al primo. Altrimenti per un notevole risparmio computazionale, il primo modello sarebbe adeguato.

Capitolo 6

Conclusioni e sviluppi futuri

In questo lavoro di tesi sono stati presentati gli argomenti alla base del Machine Learning. Sono state introdotte quindi le assunzioni che permettono di sviluppare i modelli, la funzione di perdita che viene ottimizzata attraverso algoritmi di ottimizzazione, quali ad esempio la discesa del gradiente e la sua variante stocastica.
Dopo la parte iniziale è stata presentata la branca del Machine Learning che in seguito è stata utilizzata direttamente per analizzare il caso d'uso, ovvero il Deep Learning. Tra i modelli che compongono questo ambito sono stati descritti alcuni dei principali: reti feed-forward, reti convoluzionali, ricorrenti e autoencoder. Inoltre è stato illustrato il ragionamento che sta alla base dell'algoritmo di backpropagation, che permette di calcolare in modo efficiente i pesi delle reti neurali, ovvero le parti addestrabili di questi modelli che attraverso l'osservazione di record imparano ad adattarsi ai dati.
In seguito sono stati illustrati i due modelli utilizzati per l'analisi. Il primo consiste in un autoencoder al cui interno sono presenti strati LSTM, in modo tale da poter trattare dati sequenziali e catturarne le dipendenze temporali. Il secondo modello invece consiste sempre in un autoencoder ma applicato non ai dati originali, bensì a matrici generate appositamente. Queste matrici vengono costruite modellando le dipendenze tra le variabili nel tempo per mezzo di finestre temporali mobili di diversa lunghezza, rendendo il modello di tipo multi-scala.
Successivamente è stato descritto l'ambito dell'individuazione di anomalie, caratterizzando le particolarità che incidono sulla scelta degli algoritmi da utilizzare.
E' stata fornita una panoramica sia sul tema del Monitoraggio delle condizioni strutturali sia sul caso d'uso, il quale si riferisce ad un sistema di monitoraggio nell'ambito automobilistico.
Dopo una descrizione del dataset sono state illustrate le misure e i grafici che vengono utilizzati per valutare una classificazione di tipo binario.
Infine sono stati riportati i risultati di entrambi i modelli rispetto a tre casi considerati, Per quanto riguarda il modello LSTM-ED, nel primo caso ottiene buone prestazioni con valori minimi di precisione e richiamo dell'87%. Questo modello nel secondo caso presenta un ottimo valore di richiamo per la classe anomala ma un valore molto basso per valori sani. Questo è indice del fatto che il classificatore non riesca a separare bene le due distribuzioni, impostando quindi una soglia molto bassa a discapito di ottenere troppi

falsi positivi. Lo stesso comportamento del modello si riscontra nel terzo caso, seppur con risultati leggermente migliori.
Per quanto riguarda invece i risultati ottenuti con il secondo modello, MSCRED, nel primo caso risultano superiori a quelli corrispondenti ottenuti con LSTM-ED. MSCRED risulta un ottimo modello per identificare anomalie di tipo 1. Infatti per il secondo caso si ottengono risultati alti sia per precisione sia per richiamo di entrambe le classi, ottenendo valori minimi di precisione e richiamo del 93%. La stessa bontà di classificazione non si riscontra nell'analisi del terzo caso, ottenendo un richiamo dell'80% sulla classe 0.
Infine nell'ultima parte della tesi sono riportati in una tabella i valori dell'F_1 score che permettono di visualizzare immediatamente in quali casi gli algoritmi sono preferiti agli altri, evidenziando, come l'algoritmo MSCRED raggiunga prestazioni migliori a discapito di tempi computazionali superiori rispetto a LSTM-ED.
Questo lavoro di tesi può essere utilizzato come paragone per l'utilizzo di ulteriori modelli di Deep Learning applicati al monitoraggio delle condizioni strutturali, ponendo maggiore attenzione sulle prestazioni ottenute e sui tempi necessari di calcolo, che è una caratteristica fondamentale da tener conto nello sviluppo di un modello che classifichi in maniera continua i dati ottenuti dalla struttura.
Inoltre risulta interessante effettuare uno o più confronti con modelli di statistica tradizionale allo stato dell'arte, investigando se questi ultimi raggiungono le prestazioni dei modelli di Deep Learning e quali sono i tempi di calcolo.

Bibliografia

[1] Christian Boller, Fu-Kuo Chang, and Yozo Fujino. *Encyclopedia of Structural Health Monitoring*. 01 2009.

[2] Peter Bruce, Andrew Bruce, and Peter Gedeck. *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*. O'Reilly Media, 2020.

[3] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[5] Alberto Ciampaglia, Antonio Mastropietro, Alessandro De Gregorio, Francesco Vaccarino, Giovanni Belingardi, and Enrico Busto. Artificial intelligence for damage detection in automotive composite parts: A use case. Technical report, SAE Technical Paper, 2021.

[6] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[7] Luigi Grassia, Michele Iannone, America Califano, and Alberto D'Amore. Strain based method for monitoring the health state of composite structures. *Composites Part B: Engineering*, 176, 2019.

[8] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[9] Aya Abdelsalam Ismail, Timothy Wood, and Héctor Corrada Bravo. Improving long-horizon forecasts with expectation-biased lstm networks. *arXiv preprint arXiv:1804.06776*, 2018.

[10] Safaa Laqtib, Khalid El Yassini, and Moulay Lahcen Hasnaoui. A deep learning methods for intrusion detection systems based machine learning in manet. In *Proceedings of the 4th International Conference on Smart City Applications*, pages 1–8, 2019.

[11] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[12] Sophia Metaxa, Konstantinos Kalkanis, Constantinos S Psomopoulos, Stavros D Kaminaris, and George Ioannidis. A review of structural health monitoring methods for composite materials. *Procedia Structural Integrity*, 22:369–375, 2019.

[13] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[14] Avishek Pal and PKS Prakash. *Practical time series analysis: master time series data processing, visualization, and modeling using python*. Packt Publishing Ltd, 2017.

[15] Jingyi Shen, Runqi Wang, and Han-Wei Shen. Visual exploration of latent space for traditional chinese music. *Visual Informatics*, 4(2):99–108, 2020.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[17] Martin Vetterli, Jelena Kovačević, and Vivek K Goyal. *Foundations of signal processing*. Cambridge University Press, 2014.

[18] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1409–1416, 2019.

Elenco delle tabelle

5.1 Statistiche riassuntive dati caso d'uso 48
5.2 Esempio matrice di confusione per classificazione binaria 51
5.3 Statistiche per la valutazione della classificazione utilizzando il modello LSTM-ED per il caso 1 56
5.4 Statistiche per la valutazione della classificazione utilizzando il modello LSTM-ED per il caso 2 58
5.5 Statistiche per la valutazione della classificazione utilizzando il modello LSTM-ED per il caso 3 60
5.6 Statistiche per la valutazione della classificazione utilizzando il modello MSCRED per il caso 1 63
5.7 Statistiche per la valutazione della classificazione utilizzando il modello MSCRED per il caso 2 65
5.8 Statistiche per la valutazione della classificazione utilizzando il modello MSCRED per il caso 3 67
5.9 Confronto modelli LSTM-ED e MSCRED 69

Elenco delle figure

2.1 Relazione tra capacità di un modello, overfitting e underfitting [6] 8
2.2 Rapporto derivata e minimo di una funzione [6] 10
2.3 Impatto del tasso di apprendimento sulla funzione di perdita 11
2.4 Andamento dell'errore di addestramento al variare del tasso di apprendimento 11
2.5 Punto di minimo, punto di massimo e punto di sella [6] 13
2.6 Minimi locali e globali di una funzione [6] 13

3.1 Confronto prestazioni tra differenti tipi di algoritmi di Deep Learning e Machine Learning al variare della dimensione dei dati [10] 15
3.2 Esempi di funzioni di attivazione 16
3.3 Esempio di rete feed forward, pienamente connessa e con uno strato nascosto [13] 18
3.4 Esempio dell'azione di un filtro su un segnale bidimensionale [6] 21
3.5 Convoluzione come prodotto per matrice infinita dell'operatore H [17] . . . 22
3.6 Convoluzione circolare come prodotto per matrice dell'operatore H [17] . . 22
3.7 Esempio della possibilità di interazioni in reti convoluzionali profonde malgrado la mancanza di connessione diretta [6] 23
3.8 Esempio di invarianza per trasformazioni [6] 24
3.9 Reti ricorrenti: sequenza in output e collegamento unità nascoste - unità nascoste [6] 25
3.10 Reti ricorrenti: sequenza in output e collegamento unità output - unità nascoste [6] 26
3.11 Reti ricorrenti: singolo output e collegamento unità nascoste - unità nascoste [6] 27
3.12 Esempio di una singola cella LSTM e del flusso di informazioni che scorre attraverso[9] 29
3.13 Primo modello: LSTM Encoder-Decoder [15] 33
3.14 Secondo modello: MSCRED [18] 37

5.1 Matrice di correlazione classe 0 49
5.2 Matrice di correlazione classe 1 49
5.3 Matrice di correlazione classe 2 50
5.4 Suddivisione dataset nel caso d'uso 54
5.5 Matrice di confusione con modello LSTM-ED (caso 1) 56

5.6 Curva ROC utilizzando LSTM-ED (caso 1) 57
5.7 Distribuzioni punteggi di anomalia con LSTM-ED (caso 1) 57
5.8 Matrice di confusione con modello LSTM-ED (caso 2) 58
5.9 Curva ROC utilizzando LSTM-ED (caso 2) 59
5.10 Distribuzioni punteggi di anomalia con LSTM-ED (caso 2) 59
5.11 Matrice di confusione con modello LSTM-ED (caso 3) 60
5.12 Curva ROC utilizzando LSTM-ED (caso 3) 61
5.13 Distribuzioni punteggi di anomalia con LSTM-ED (caso 3) 61
5.14 Matrice di confusione con modello MSCRED (caso 1) 63
5.15 Curva ROC utilizzando MSCRED (caso 1) 64
5.16 Distribuzioni punteggi di anomalia con MSCRED (caso 1) 64
5.17 Matrice di confusione con modello MSCRED (caso 2) 65
5.18 Curva ROC utilizzando MSCRED (caso 2) 66
5.19 Distribuzioni punteggi di anomalia con MSCRED (caso 2) 66
5.20 Matrice di confusione con modello MSCRED (caso 3) 67
5.21 Curva ROC utilizzando MSCRED (caso 3) 68
5.22 Distribuzioni punteggi di anomalia con MSCRED (caso 3) 68