UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche, Informatiche e Matematiche
Corso di Laurea in Informatica

Anomaly e Novelty Detection ed algoritmi di confronto tra curve applicati all'innovazione tecnologica ed industriale

Nicholas Bernardoni

Tesi di Laurea

Relatore:
Prof. Riccardo Martoglia

Anno Accademico 2019/2020

RINGRAZIAMENTI

Ringrazio l'Ing. Riccardo Martoglia, mio relatore, per la continua disponibilità e assistenza durante l'intero percorso universitario.
Ringrazio l'Ing. Marcello Arletti, mio relatore aziendale, per l'aiuto datomi soprattutto durante la stesura delle tesi.
Ringrazio soprattutto la mia famiglia per avermi accompagnato e per non avermi fatto mancare nulla durante questo percorso.
Un ringraziamento speciale va ai miei amici di Marano che mi hanno sempre supportato e spronato a dare il meglio.

PAROLE CHIAVE

LSTM

Novelty Detection

Confronto

FFT

Innovazione

Indice

Parte II – Sviluppo ed Implementazione

Introduzione

Al giorno d'oggi, le aziende richiedono sempre di più tecnologie che siano in grado di evitare dei guasti all'interno dei macchinari produttivi e che possano quindi comunicare eventuali stati di errore con un certo anticipo agli operatori, prima che possano verificarsi danni considerevoli all'attività produttiva. Le aziende possono perdere consistenti somme di denaro a causa di inefficienze all'interno dei processi produttivi, come:

- cambi preventivi di componenti: alcune imprese, per evitare rotture degli elementi che costituiscono i macchinari che potrebbero arrestare la produzione, decidono di sostituire alcune parti delle macchine con un certo anticipo, anche se il componente potrebbe essere ancora capace di continuare la sua attività senza generare alcun problema;
- rotture dei componenti: a causa di errori umani o di scarso controllo dello stato di salute dei componenti, possono verificarsi rotture di elementi delle macchine che possono portare ad un fermo della produzione per periodi anche discretamente lunghi;

Grazie agli enormi passi avanti fatti dalla tecnologia coadiuvata dall'Intelligenza Artificiale, dal Machine Learning, dalle reti neurali, oggi è possibile sviluppare tecniche e metodi che permettono alle aziende di predire guasti all'interno dei componenti prima che questi si verifichino. Tutto questo è possibile utilizzando diverse tecniche di approccio, ma sempre attraverso un'analisi storica dei dati sul medio-lungo periodo, che vengono rilevati direttamente dal campo (ad esempio: tramite sensori montati sui macchinari) e storicizzati su database a formare il data lake necessario all'impiago dell'intelligenza artificiale.

Proteo Engineering SRL ha deciso di investire in questo progetto di ricerca e sviluppo per portare innovazione sul mercato, investendo su tecniche nuove.

Il lavoro condotto all'interno di quest'azienda riguarda due parti principali:

- sviluppo di tecniche di confronto che permettessero di avere una metrica di non similarità tra due segnali provenienti da vari sensori;

- sviluppo di tecniche di Novelty ed Anomaly Detection che, attraverso reti neurali, permettessero di predire un cambiamento all'interno dei dataset di valori forniti da sensori montati sui cilindri di una cartiera.

Il seguente elaborato è strutturato in cinque capitoli:

- nel primo capitolo verranno illustrate le parti teoriche riguardanti le trasformate, in particolare le FFT, e la realtà applicativa di Proteo Engineering SRL con l'analisi dei sensori, del progetto Vibromet e degli obiettivi;
- nel secondo capitolo verranno spiegati ed analizzati i problemi affrontati e le tecnologie utilizzate per lo sviluppo di metodi che permettessero di confrontare due curve e di metodi riguardanti Novelty ed Anomaly Detection;
- nel terzo capitolo si andranno ad affrontare la progettazione dei metodi di confronto e delle LSTM;
- nel quarto capitolo si andrà ad analizzare il codice sviluppato per implementare i metodi esplicati nei precedenti capitoli;
- nel quinto capitolo verranno analizzati i risultati dei test effettuati.

Parte I

Il Caso di studio

Capitolo 1

Caso di studio e realtà applicativa

Nella prima parte di questo capitolo, si andranno ad analizzare gli aspetti teorici riguardanti le trasformate ed in particolare la trasformata di Fourier che svolge un ruolo fondamentale nell'analisi dei segnali. Nella seconda parte del capitolo si andrà ad analizzare la realtà applicativa di Proteo Engineering SRL, il progetto Vibromet e gli obbiettivi per questa attività di ricerca e sviluppo.

1.1 Cos'è una trasformata

Una trasformata è un'applicazione, generalmente lineare, definita in uno spazio di funzioni rispetto ad un altro spazio di funzioni. In pratica trasforma una funzione in un'altra. Tale trasformata è solitamente applicata ad una funzione per semplificare alcune operazioni o in generale per risolvere più facilmente dei problemi.

Sia dato un problema A (complesso), si potrebbe procedere come segue:

1. si trasforma il problema A in un altro problema B più semplice;
2. si risolve il problema B;
3. si antitrasforma la soluzione del problema B nella soluzione del problema A.

1.2 Trasformata di Fourier

La trasformata di Fourier [1] permette di scomporre un'onda, anche molto complessa e rumorosa (per esempio il suono di uno strumento musicale o la voce) in più sotto-componenti (ampiezza, fase e frequenza) delle onde sinusoidali che, sommate tra loro, danno origine al segnale di partenza.

Una funzione f(x), arbitraria a patto che sia periodica di periodo T, può essere decomposta come somma infinita di seni e di coseni (serie trigonometrica). In pratica è uno sviluppo del genere:

$$f(x) \sim \sum_{k=0}^{\infty} (a_k cos(k\omega x) + b_k sin(k\omega x)), \qquad \omega = \frac{2\pi}{T}$$

Figura 1.1: Serie di Fourier

Le serie di Fourier (figura 1.1) permettono lo studio di un segnale periodico, mentre la trasformata di Fourier permette l'analisi di un segnale x(t) il cui andamento nel tempo non è per forza periodico.

1.3 DFT e FFT

La trasformata di Fourier consente di trattare segnali a tempo continuo secondo le loro componenti armoniche. Analogamente, la trasformata di Fourier a tempo discreto, permette l'analisi della frequenza dei segnali a tempo discreto.

Ogni tipo di segnale a tempo discreto può essere approssimato attraverso segnali digitali. L'analisi in frequenza di questi segnali può essere effettuata attraverso la Trasformata Discreta di Fourier (DFT) [2]. Un maggiore utilizzo della DFT nell'analisi dei segnali è stata la scoperta di una classe di algoritmi veloci, chiamata FFT (Fast Fourier Transform), che consente il calcolo della trasformata di Fourier a tempo discreto in tempo quasi lineare nella dimensione dei dati. Il calcolo diretto di questa trasformata è costoso, in quanto richiede **$O(N^2)$** operazioni di prodotto, dove N è la dimensione dello spazio su cui si applica la trasformata. Utilizzando principalmente tecniche "divide et impera", si sono scoperti algoritmi FFT che calcolano la DFT riducendo a **$O(N\log_2 N)$** il numero delle operazioni di moltiplicazione.

La trasformata di Fourier a tempo discreto può essere applicata a segnali campionati; tali segnali sono a tempo discreto e con frequenza normalizzata nel continuo [0, 2π). Per poter trattare opportune approssimazioni di tali segnali con tecniche digitali, dobbiamo:

1. considerare solo un numero finito di campioni nel tempo;
2. effettuare un campionamento anche in frequenza, così da considerare solo un numero finito di frequenze anziché l'intervallo continuo [0, 2π).

L'obiettivo del punto uno può essere raggiunto con l'approssimazione dell'informazione contenuta in un segnale f(t) con quella ottenuta dal vettore x formato da N campioni del segnale campionato con passo τ:

$$x = [x(0), \ldots, x(N-1)], con x(n) = f(n\tau), n = 0, \ldots, N-1$$

Il vettore x contiene quindi l'informazione del segnale all'interno dell'intervallo temporale [0, (N−1) τ].

Il secondo punto tiene in considerazione il vettore X che è formato da N campioni della trasformata a tempo discreto X(Ω), campionata ad intervalli di ampiezza 2π/N:

$$X = [X(0), \ldots, X(N-1)], con X(k) = X((2\pi/N)k), k = 0, \ldots, N-1$$

dove si denota, con la stessa variabile, il vettore X(k) che rappresenta la DFT e l'applicazione complessa che rappresenta la trasformata di Fourier a tempo discreto X(Ω). Sotto l'ipotesi che l'energia del segnale sia essenzialmente contenuta negli N campioni x(0), . . ., x(N−1) vale che:

$$X((2\,\pi/N)k) = \sum_{n=-inf}^{+inf} x(n)\mathrm{e}^{-i\frac{2\pi}{N}kn} \approx \sum_{n=0}^{N-1} x(n)\mathrm{e}^{-i\frac{2\pi}{N}kn} = X(k)$$

Si può quindi approssimare la trasformata di Fourier con la seguente trasformazione tra vettori complessi N-dimensionali.

$$X(k) = \sum_{n=0}^{N-1} x(n)\mathrm{e}^{-i\frac{2\pi}{N}kn} = X(k)$$

Quella indicata sopra è detta Trasformata Discreta di Fourier.

1.4 Calcolo veloce della DFT

La DFT è ritenuta molto importante perché esistono classi di algoritmi che ne permettono un calcolo veloce; essi adottano una tecnica ricorsiva divide-et-impera che consiste nella scomposizione della DFT di partenza in trasformate che sono la metà di quella di partenza.

La DFT richiede la moltiplicazione di un vettore a N componenti [x(0) ,..., x(N− 1)] per la matrice N x N la cui componente alla riga n e colonna k è:

$$W_N^{kn} = \mathrm{e}^{-i\frac{2\pi}{N}kn}$$

Tale calcolo può essere effettuato con $O(N^2)$ operazioni di somma e prodotto, poiché il calcolo di ognuna delle N componenti X(k) richiede O(N) operazioni, ma ciò non è utilizzabile nella realtà applicativa. Sono quindi stati proposti algoritmi più efficienti per effettuare questo calcolo, come quello proposto da Runge e Konig nel 1924 e da Cooley e Tukey [3] nel 1965. Questi algoritmi richiedono O(NlogN) invece di $O(N^2)$ operazioni e sono chiamati algoritmi FFT.

1.5 Realtà applicativa

1.5.1 Proteo Engineering SRL

Proteo Engineering SRL è specializzata da oltre nella realizzazione di impianti automatici e di potenza "chiavi in mano". Negli anni, il team di Proteo ha accumulato diverse esperienze che li ha affermati a livello mondiale su molteplici settori.

Alcune delle mansioni e ambiti di cui si occupa Proteo:

- controllo di processo;
- automazione e robotica;
- automazioni industriali complete di hardware e software;
- soluzioni software e di raccolta dati, monitoraggio, sistemi MES;
- attività di ricerca e sviluppo ed innovazione tecnologica.

Proteo Engineering ha sedi in varie parti del mondo tra cui Russia e Stati Uniti.

1.5.2 Progetto Vibromet

Vibromet [4] è un sistema di raccolta dati provenienti da sensori triassiali, che elabora i dati ricevuti in ingresso per effettuare una manutenzione predittiva.

Oltre ad analizzare le accelerazioni, è possibile selezionare il controllo delle frequenze prodotte (FFT), impostando limiti di ampiezza e frequenza per ogni singolo asse.

La strumentazione memorizza i livelli delle varie energie prodotte, in modo tale da poterli analizzare ed averli come base per avere un segnale di allarme più preciso. Quest'ultimo viene analizzato dall'operatore addetto al macchinario tramite un monitor che segue lo stato del sensore. Il sistema, inoltre, fornisce un set di API che permettono di attivare vari sensori all'interno della macchina a cui viene applicato.

In realtà, esistono varie fasi di questo processo: si parte dal sensore che viene applicato alla macchina di cui si vogliono analizzare i segnali. I dati raccolti vengono poi inviati tramite TCP/IP o tramite rete Wireless al software Vibromet. Di questi dati ne viene poi tenuta traccia attraverso il loro storage e in modo tale che possano essere utilizzati come base per analisi predittive. Infine, i dati vengono visualizzati ed analizzati da un operatore. Possono essere eseguiti report ed analisi anche sul lungo periodo.

Grazie all'autoapprendimento, questo sistema potrebbe, in linea teorica, essere applicato a qualsiasi settore industriale e su moltissime tipologie di macchinari, come ad esempio atomizzatori,

miscelatori, linee di verniciature e presse. Inoltre, può essere applicato su impianti industriali, come nelle Acciaierie, nei sistemi di dosaggio, nei forni, ma anche a livello delle infrastrutture come nei ponti ferroviari, nelle scale mobili o nei trasformatori. Di seguito un'immagine (figura 1.2) dei possibili punti in cui montare questi tipi di sensore:

Figura 1.2 Sensori applicati su una scala mobile

Il sistema consente di acquisire periodicamente dati provenienti dagli accelerometri (i quali rilevano l'energia dispersa dai componenti di un elemento meccanico), in modo tale da poter calcolare la tendenza di ciascun gruppo armonico, prevedendone così gli sviluppi futuri, segnalare l'errore nel caso si arrivi ad una condizione che si allontani dalla normalità ed evitare così possibili guasti o effetti negativi come, ad esempio, fermi della produzione o problemi della qualità. Nella figura 1.3 viene mostrata un grafico di un'analisi dei dati generica.

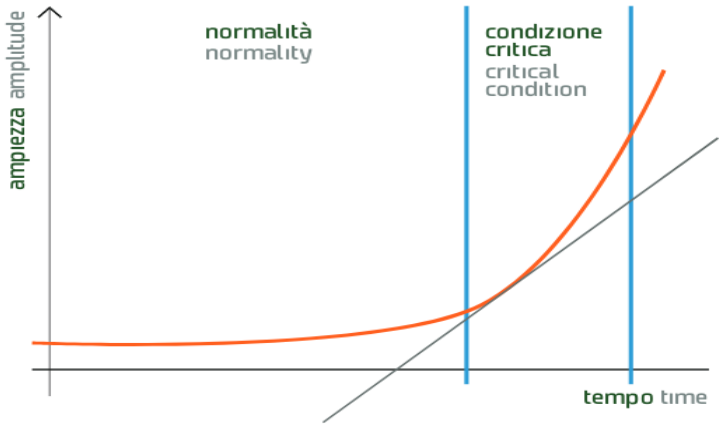


Figura 1.3 Grafico di analisi dei dati

1.5.3 Sensori utilizzati: Shock and Vibrations Sensor MEMS

I sensori utilizzati per realizzare il progetto Vibromet sono basati sulla tecnologia MEMS (Micro Electro-Mechanical Systems) e servono per il monitoraggio degli shock e delle vibrazioni. In figura 1.4 ne abbiamo un esempio.

È composto da vari LED di diversi colori:

- verde: indica quando si stanno eseguendo le operazioni standard;
- giallo: indica un ciclo di scrittura oppure di programmazione della memoria;
- blu: indica il transito di pacchetti su bus RS-485 (viene rilevato un pacchetto in transito);
- rosso: indica l'interruzione dell'accelerometro (quando si esce dalla soglia della "normalità" del sistema).

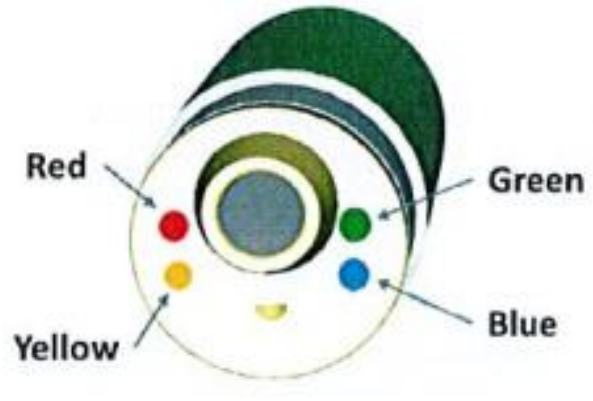


Figura 1.4 Sensore MEMS

Quando acceso, dopo i vari check del sistema, il dispositivo richiamerà dalla memoria l'ultima configurazione salvata e torna ad eseguire le operazioni riportare dal LED verde. L'output nella configurazione standard mostra il valore delle accelerazioni registrate sull'asse X. Bisogna inoltre considerare in che modo viene montato il sensore: nel caso in cui non venga montato in corrispondenza di un'asse, bisogna calcolare le due corrispondenti componenti (seno e coseno dell'angolo) e su questo normalizzare il segnale.

Per evitare collisioni e/o errori di comunicazione, tutti i comandi sono incapsulati in pacchetti. Questi ultimi sono suddivisi in due categorie: quelli a "short syntax" e quelli ad "extended syntax". Solitamente, i pacchetti che rientrano nella prima categoria sono quelli che vengono inviati senza parametri, mentre quelli compresi nella seconda categoria comprendono parametri e sono protetti da

controlli di checksum. Inoltre, il sensore risponde con un ACK (segnale di Acknowledge) che viene emessa in risposta ad un comando, solo se questo è stato ricevuto correttamente.

1.6 Obiettivi e applicazioni

Il progetto che sono andato a sviluppare si è svolto principalmente in due fasi:

- **prima fase**: in cui sono andato ad implementare un algoritmo che permettesse di confrontare due segnali FFT dati in ingresso e che permettesse di stabilirne il grado di non similarità;
- **seconda fase**: avendo dei dati provenienti da sensori di una cartiera, sono andato ad effettuare un'attività di ricerca e sviluppo, in cui sono andato ad implementare un algoritmo basato su LSTM, cercando di andare a predire un possibile cambiamento dei dati, come ad esempio una possibile condizione di errore, prima che questa si verifichi.

Sempre di più, al giorno d'oggi, ci sono aziende che richiedono metodi che permettano di poter anticipare un cambiamento all'interno della produzione. Questo permette di evitare problemi di qualità nel prodotto stesso, con conseguenti danni alla produzione, sia problemi sui macchinari e sugli impianti, ad esempio: la rottura di un componente all'interno di un macchinario.

Prendiamo l'esempio di una ceramica: se la produzione dovesse fermarsi, a causa della rottura di uno stampo all'interno di una pressa, questo provocherà il fermo dell'attività produttiva e un'ingente perdita economica.

Ogni tipo di azienda media o grande che sia, vuole evitare di perdere profitti a causa di un guasto che potrebbe essere evitato. Nell'ultimo periodo storico è stato portato avanti un nuovo paradigma: la manutenzione preventiva. L'azienda, grazie a dati storici e statistici, sa che un certo componente X ha una vita media di utilizzo di due anni e dopo un anno decide di sostituirlo. Questo è un caso che ipoteticamente salvaguarda l'azienda dalla rottura per usura ma non ottimizza la gestione dei costi. Potrebbe verificarsi uno spreco di denaro, in quanto quello specifico componente è stato sostituito pur avendo ancora vita residua e non è utilizzato al massimo del suo potenziale.

L'obiettivo che Proteo Engineering SRL si prefigge con questo progetto di ricerca e sviluppo, è quello di fornire alle aziende un sistema che possa realmente aiutarle nell'evitare buona parte delle tipologie di problema derivante da un calcolo umano errato o da una condizione di anomalia imprevista dei macchinari e che possa portare ad un mancato introito per le aziende. In questo specifico momento, sul mercato, non vi è una offerta di un sistema così completo, pur essendo in uno stato di grossa domanda. Per questo Proteo Engineering ha deciso di investire in questo progetto, cercando di portare innovazione nel settore, investendo sui metodi di Intelligenza Artificiale, come la Novelty Detection, che faranno parte sempre di più della nostra quotidianità.

Capitolo 2

Problemi affrontati e tecnologie utilizzate

All'interno di questo capitolo si andranno ad analizzare le attività che sono state affrontate durante l'intera attività all'interno di Proteo Engineering SRL. Nella prima parte verranno riportati i metodi riguardanti la prima fase dell'attività in cui ho sviluppato un codice che permettesse che mi permettesse di confrontare due curve provenienti da un segnale. Nella seconda parte, invece, verranno introdotti vari concetti e tecniche che mi hanno portato a sviluppare un programma basato sulle Reti Neurali.
Nell'ultima parte vengono illustrate le tecnologie che hanno permesso la costruzione di tali modelli.

2.1 Confronto di due curve

Il problema che mi sono trovato ad affrontare all'inizio della mia attività di tirocinio, è stato quello di dover implementare un algoritmo, sviluppato in Python, che preso il modulo di due FFT, relative al medesimo segnale ma a due intervalli diversi, andasse a rilevare le differenze nello spettro del segnale. Si aveva quindi:

- una curva di riferimento
- una curva di confronto

La prima curva veniva confrontata con la seconda in modo tale da rilevarne la non similarità. Dato che l'algoritmo verrà poi applicato ad un segnale FFT, è stato ipotizzato che il numero di campioni delle due curve fosse il medesimo e che, a ciascun campione, corrispondesse la medesima frequenza, ovvero è stato possibile trascurare la normalizzazione sull'asse delle ascisse.
L'invarianza del numero di campioni e delle ordinate, cioè il campione *i*-esimo, corrisponderà sempre alla medesima frequenza e ciò ha portato a pensare che sarebbe stato possibile utilizzare tecniche che consentissero il confronto di distribuzioni, come ad esempio il **Kolmogorov-Smirnov Test [5]**.
Il Kolmogorov-Smirnov Test (KS test) è un test non parametrico che viene utilizzato per decidere se un campione deriva da una popolazione con una specifica distribuzione.
La statistica Kolmogorov-Smirnov quantifica una distanza tra la funzione di distribuzione empirica del campione e la funzione di distribuzione cumulativa della funzione di riferimento o tra due funzioni di distribuzioni empiriche di due campioni.

Il risultato del test riporta la massima differenza tra due distribuzioni cumulative e calcola il P-value su queste. Il valore P aiuta a capire se la differenza tra il risultato osservato e quello ipotizzato è dovuta alla casualità del campionamento, oppure se la differenza è statisticamente significativa, cioè difficilmente spiegabile attraverso la casualità del campionamento. Quando questo valore è piccolo, si può concludere che i due campioni derivano da una popolazione con differenti distribuzioni. La popolazione potrebbe variare per: la mediana, la varianza oppure per la forma della distribuzione.
Il KS test, effettuato su due campioni, è uno dei metodi parametrici più utilizzati ed è quello che è stato utilizzato per confrontare due curve.
Un altro metodo che è stato preso in considerazione per confrontare tra loro due FFT, è stato quello di prendere in analisi i singoli punti che componevano i segnali di input e confrontarli tra di loro attraverso una sorta di *Scarto Quadratico Medio*. Sono stati considerati due segnali, di cui ne è stata calcolata la differenza puntuale tra il primo ed il secondo segnale. Di tale differenza ne è stata calcolata la media, lo scarto e lo scarto quadratico medio, che è il valore significativo, che si voleva ottenere.
Ultimo metodo che è stato utilizzato, è quello di studiare le aree di due integrali, confrontandole attraverso un rapporto. Questo permette di osservare quanto le due aree sottese ed i relativi segnali siano simili tra di loro. Più il valore ritornato dal rapporto è vicino a 0 e più i due segnali possono considerarsi equivalenti.

2.2 Anomaly e Novelty Detection

Nella seconda parte della mia attività ho utilizzato modelli di intelligenza artificiale che mi permettessero di costruire un modello predittivo. Mi sono trovato a sviluppare un programma che potesse rilevare, con un certo anticipo, una novità o un'anomalia all'interno dei dati.

2.2.1 Cosa si intende con Anomaly Detection

Il processo di Anomaly Detection [6] consiste nel riconoscimento di eventi inattesi all'interno dell'insieme dei dati, che differiscono dalla condizione di normalità. Dati anomali possono indicare incidenti critici, come glitch tecnici, oppure potenziali opportunità, ad esempio un cambio di comportamento nei comportamenti dei consumatori.

Partendo dalla base, un'anomalia è un evento che si verifica raramente e deve essere significativo: se si verifica spesso, non può essere definito "anomalo" ma è per forza riconducibile ad una qualche regola. Un valore anomalo, in statistica, viene definito come ***outlier***.

Quando cerchiamo di identificare una condizione anomala, ciò che ci si prospetta davanti può essere più o meno complesso ed è dipendente dal numero di variabili che si vogliono prendere in considerazione. Con riferimento a questo, possiamo fare una distinzione tra Anomaly Detection univariata e monovariata: nel primo caso, gli outlier possono emergere osservando la distribuzione dei valori relativi ad un singolo parametro, mentre nel secondo caso possono emergere dall'osservazione di due variabili.

Altra distinzione che viene fatta, riguarda il tipo di approccio che può essere supervisionato o non supervisionato. Quando si vuole adoperare un *approccio supervisionato*, la natura dei dati deve essere nota in partenza: l'algoritmo viene addestrato con dei dati disponibili fino a quando non riconosce l'anomalia in maniera autonoma.

Se, invece, si adotta un approccio non supervisionato, gli elementi che possono provocare un risultato anomalo non sono noti in partenza.

2.2.2 Novelty e Outlier Detection

Esistono principalmente due metodi che utilizzano l'Anomaly Detection:

- l'Outlier Detection [7], in cui i dati di training contengono valori anomali che sono definiti come osservazioni lontane dalle altre. Gli stimatori provano ad eseguire una fit sulla regione in cui i dati di training sono maggiormente concentrati;
- la Novelty Detection [8], in cui i dati di training non sono alterati da valori anomali e quindi siamo interessati ad individuare se una nuova osservazione è un outlier o meno. In questo contesto, l'anomalia è chiamata anche novità.

Il primo metodo è conosciuto anche come ricerca dell'anomalia non supervisionata, la seconda invece è semi-supervisionata.

La Novelty Detection rappresenta un importante task in molti sistemi diagnostici e di monitoraggio. Nei sistemi di sicurezza è essenziale rilevare le occorrenze anomale degli eventi il più velocemente possibile, prima che ci siano significativi risultati di regressione. Questo può essere ottenuto attraverso il continuo monitoraggio delle deviazioni del sistema dal comportamento normale dei pattern. Consideriamo, ad esempio, un dataset di *n* osservazioni derivanti dalla stessa distribuzione. Ora vogliamo aggiungere un'osservazione a questo set di dati: se la nuova osservazione è diversa dalle altre, possiamo suppore che questa non rappresenti la normalità, altrimenti, se è molto simile a quelle precedenti, non possiamo distinguerla. La Novelty Detection è questo che cerca di andare a capire.

I passaggi classici per andare ad implementare un modello di Novelty Detection sono mostrati in figura 2.1: la procedura inizia con un processing offline dei dati grezzi (ad esempio le temperature o gli stati del sensore) e successivamente viene "istruito" il modello sui dati che abbiamo estratto.

Grazie a questa fase iniziale di training, il modello durante il monitoraggio online andrà ad analizzare le nuove acquisizioni e a determinare se i nuovi dati corrispondono a scenari conosciuti (derivanti dalla fase di training) oppure se presentano caratteristiche differenti e possono essere considerate novità. In caso di scenari conosciuti, l'analisi continua, altrimenti questa non può essere portata avanti in quanto l'affidabilità dell'analisi sarebbe compromessa. In questo caso, la presenza di questi dati viene riportata e dopo la supervisione di un esperto, le misurazioni sono salvate in modo tale da poter aggiornare il modello corrente, poter effettuare di nuovo il training e continuare l'analisi. Nella figura 2.1 viene mostrato visivamente il processo di Novelty Detection.

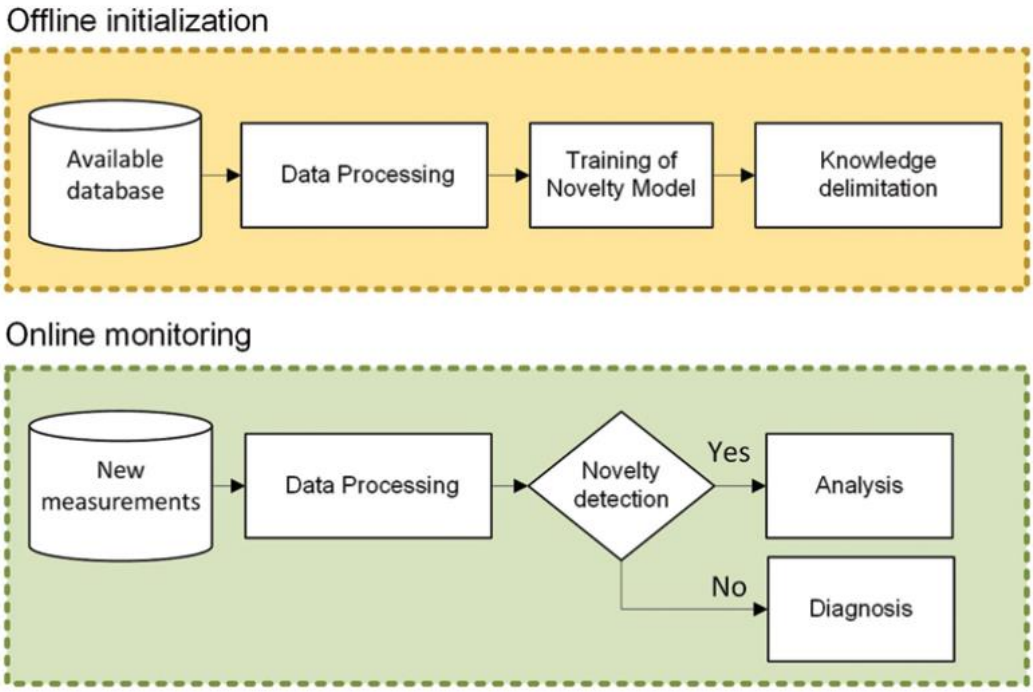


Figura 2.1: Processo di Novelty Detection

2.3 Reti Neurali Artificiali

2.3.1 Cosa sono

Le Reti Neurali Artificiali [9], o semplicemente Reti Neurali, sono sistemi di computazione che si inspirano alle reti neurali biologiche che costituiscono la mente animale.
Una rete neurale è composta da una collezione di neuroni artificiali collegati fra loro. Ogni connessione trasmette un segnale agli altri neuroni. Il segnale è rappresentato da un numero reale è calcolato da alcune funzioni non lineari come somma dei suoi input.
Le connessioni vengono chiamate **archi,** come in un grafo. Solitamente, gli archi ed i neuroni hanno un proprio peso, che cresce o decresce la forza del segnale di connessione. Inoltre, i neuroni possono avere una soglia che permette di inviare un segnale solamente se viene superato tale valore.
Le reti neurali imparano processando esempi, ognuno contenente un input ed un risultato conosciuti, formando un'associazione di probabilità pesata tra i due, che viene immagazzinata all'interno di una struttura dati nella rete stessa. Solitamente la fase di training di una rete neurale viene eseguita calcolando la differenza tra l'output calcolato della rete e l'output che ci si aspetta: questo viene chiamato errore. In seguito, la rete aggiusta i pesi delle varie associazioni secondo le regole di apprendimento e utilizzando l'errore appena calcolato. Grazie a questi vari aggiustamenti, il valore dell'output predetto sarà sempre più simile al valore reale.

2.3.2 Struttura

Le ANN (Artificial Neural Network) [10] cercano di simulare il comportamento del cervello animale che è composto dai neuroni. Sono composte da nodi, che rappresentano i neuroni, e da archi che li connettono, rappresentanti le sinapsi. Quindi una rete neurale è essenzialmente un grafo orientato.
Queste reti sono spesso organizzate in livelli (layer). I vari layer sono costituiti da vari nodi interconnessi che contengono una funzione di attivazione. I livelli principali vengono così suddivisi:

- **input layer**: riceve il valore delle variabili esplicative per ogni osservazione e presenta i dati alla rete attraverso la comunicazione con uno o più "hidden layer". I nodi di input sono passivi, cioè non cambiano i dati ma li duplicano per restituirli in output e poterli mandare ai nodi successivi;
- **hidden layer**: applica una data trasformazione ai dati all'interno della rete. I valori entranti in un "hidden node" vengono moltiplicati per un peso e poi sommati per restituire un unico valore. Viene poi connesso ad altri "hidden nodes" oppure ai nodi di output;

- **output layer**: è collegato a “hidden layers” o “input layer” e ritorna un valore di output che corrisponde alla predizione della variabile. Nei problemi di classificazione, ad esempio, il nodo rappresentante l’output è solamente uno. In figura 2.2 viene mostrata la struttura di una rete neurale artificiale.

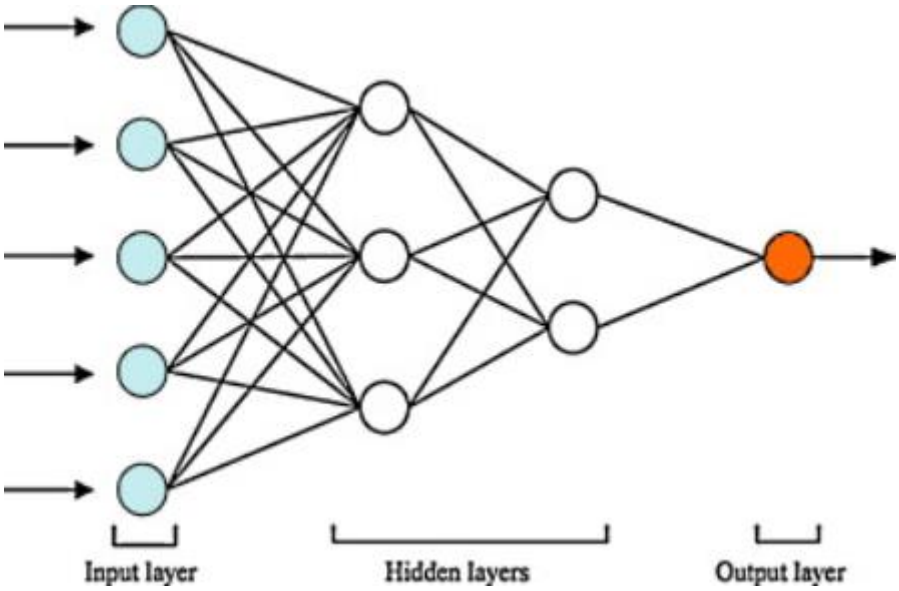


Figura 2.2: Struttura Artificial Neural Network

2.3.3 Reti Neurali Ricorrenti

Le reti neurali tradizionali non sono pensate per possedere una memoria che permetta loro di conservare i dati per un certo periodo di tempo in modo tale da poter completare un task.

Le reti neurali ricorrenti (RNN) [11] rappresentano una classe delle reti neurali artificiali in cui le connessioni vanno a formare un grafo diretto. Queste reti, durante il loro training, ricordano e prendono decisioni che sono influenzate dal passato. Per fare in modo che l’informazione persista, sono presenti dei loop al loro interno. Le RNN possono prendere uno o più valori di input e ciò che restituiranno in output sarà influenzato non solo dal peso applicato sugli input layer, come avveniva per le ANN, ma anche dai vettori che rappresentano lo stato della rete, basato sui precedenti output: uno stesso input può produrre differenti output a seconda delle precedenti serie di valori in ingresso.

Le RNN vengono impiegate in moltissimi task di previsione e sono capaci di analizzare **serie temporali** e prevedere eventi futuri, come ad esempio il prezzo delle azioni o le traiettorie dei veicoli.

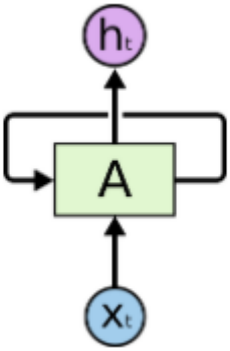


Figura 2.3: RNN con loop

Nella figura 2.3, abbiamo una rete neurale, A, che riceve un generico input x_t e restituisce un output h_t. Il loop permette alla rete di passare l'informazione da una parte all'altra della rete.
Una rete neurale ricorrente può essere vista anche come copie multiple della stessa rete, ognuna che trasmette un messaggio al successore, come mostrato in figura 2.4.

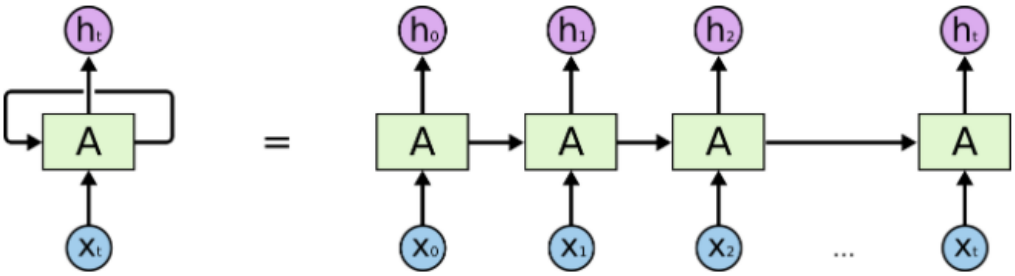


Figura 2.5: RNN con distensione del loop

Uno dei problemi delle RNN è quello delle **dipendenze a lungo termine**. Per esempio, se si vuole cercare in un modello lessicale la parola che termina la frase, in alcuni casi non abbiamo bisogno di un contesto: ad esempio nella frase "le nuvole sono nel …", è abbastanza ovvio che la parola finale sarà "cielo". In altri casi però, dedurre un'informazione a distanza di alcune frasi, può essere complicato e quindi serve il contesto, che permette di dedurre l'informazione mancante senza basarsi solo sulle parole precedenti. Più il gap cresce all'interno della frase e meno le RNN saranno in grado di connettere le varie informazioni.

2.4 Long Short Term Memory

Le Long Short Term Memory (LSTM) [12] sono delle particolari reti neurali ricorrenti in grado di assimilare concetti dipendenti tra loro nel lungo periodo: questo è il loro comportamento standard. Tutte le reti neurali ricorrenti presentano la forma di una catena di moduli che si ripetono all'interno della rete. Anche nelle LSTM si verifica questo, ma in maniera più complessa: invece di avere un solo layer, questi tipi di reti ne hanno quattro che interagiscono tra loro. La struttura di un LSTM viene mostrata in figura 2.6.

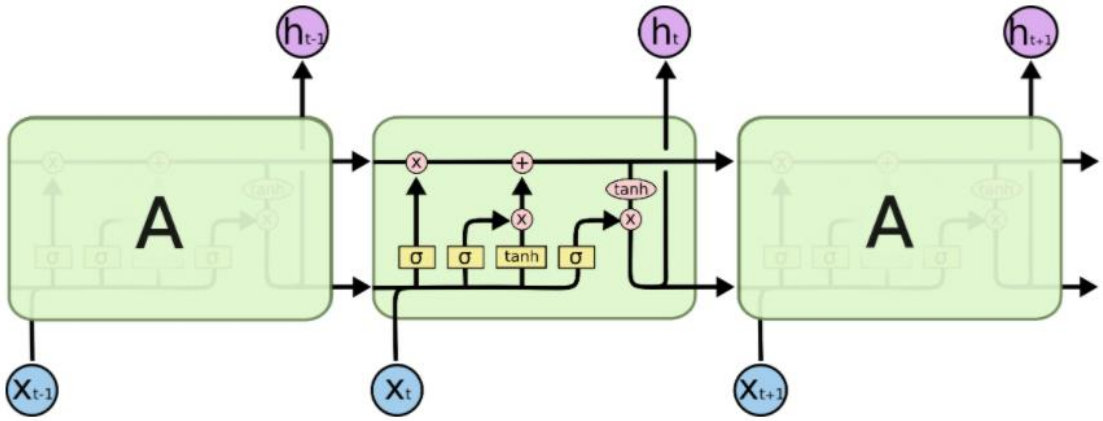


Figura 2.6: LSTM

Una parte molto importante all'interno di una LSTM è la **cella di stato** che percorre la parte alta di ogni singolo modulo. Queste reti hanno la capacità di aggiungere o rimuovere informazioni alla cella di stato; quest'operazione viene controllata da particolari strutture chiamate **gates**. I gates sono formati da una layer sigmoidale della rete neurale (funzione sigma) e da un'operazione di moltiplicazione. Questo strato della rete restituisce un valore numerico compreso tra 0 e 1: se il valore di ritorno è 0, il gate non lascia passare nulla, altrimenti, se corrisponde a 1, lascia passare tutto. A seconda del valore restituito passa più o meno informazione. Un LSTM possiede tre gates. Un singolo gate viene mostrato in figura 2.7.

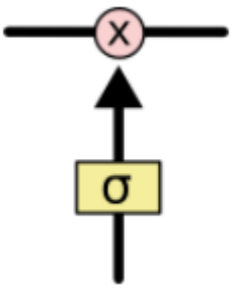
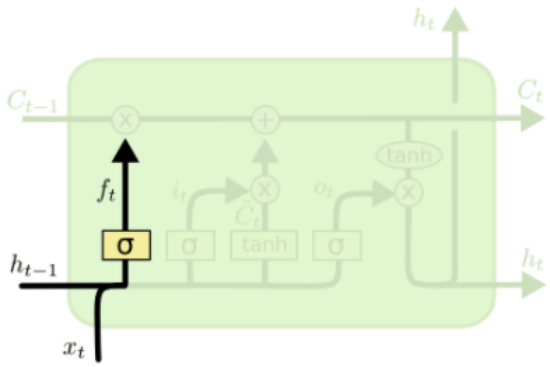


Figura 2.7: Gate in una LSTM

2.4.1 Funzionamento

Il primo passo fatto all'interno di un LSTM è quello di decidere quale informazione conservare e quale informazione scartare: questo viene eseguito da un layer sigmoidale chiamato "forget gate layer" (figura 2.8). Questo strato della rete guarda all'output della rete precedente e all'input corrente e restituirà un numero compreso tra zero ed uno per ogni valore all'interno della cella di stato C_{t-1}. La funzione di uscita viene calcolata dando in pasto alla funzione sigma il vettore dei pesi moltiplicato per l'output del livello t-1 e l'input del livello t, a cui viene poi aggiunto un fattore di normalizzazione b_f, rappresentante il bias.



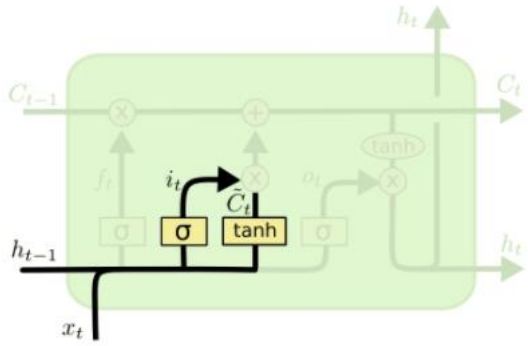
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figura 2.8: Forget Gate Layer

Il prossimo step all'interno di un LSTM è quello di decidere quale informazione andrà ad essere immagazzinata nella cella di stato. Per fare questo, il procedimento si divide in due parti:

- la prima parte serve a decidere quale valore verrà aggiornato e viene fatto attraverso una funzione sigmoidale;
- la seconda parte crea un certo numero di candidati che andranno a rappresentare lo stato corrente e andranno aggiunti alla cella di stato. I candidati vengono calcolati attraverso una tangente iperbolica.

Le due parti vengono mostrate in figura 2.9.

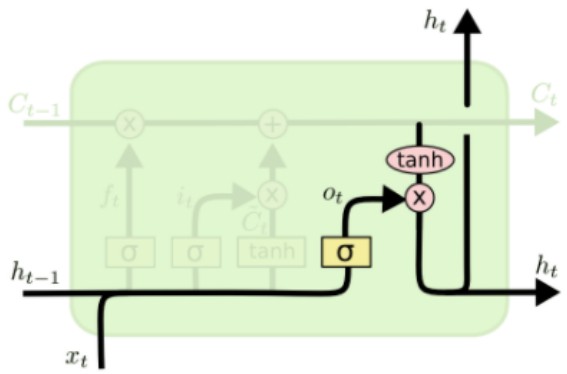


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figura 2.9: Calcolo valori dello stato corrente

Ora viene aggiornato la cella dello stato precedente, C_{t-1}, nel valore attuale C_t. Moltiplichiamo il valore calcolato nel primo step, f_t, in modo tale da scartare le informazioni che non ci interessano e successivamente sommiamo i valori calcolati al passo due, cioè C_t*i_t. Lo stato della rete viene quindi aggiornato.

Come ultimo step, viene deciso cosa viene mandato in output, che sarà una versione filtrata della cella di stato della rete. Viene applicata una funzione sigmoidale che andrà a decidere quale valore della cella di stato verrà restituito. A quest'ultima, invece, viene applicata una tangente iperbolica in modo tale da portare il valore tra 1 e -1 per essere poi moltiplicata per l'output della funzione sigmoidale (figura 2.10).



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figura 2.10: Calcolo output attuale

2.4.2 Applicazioni

Le principali applicazioni in cui vengono impiegati le LSTM sono:

- robot control;
- time series prediction;
- time series anomaly detection;
- speech recognition;
- handwriting recognition.

2.5 Tecnologie utilizzate

Durante la fase di tirocinio, ho utilizzato diversi strumenti per poter analizzare i dati e poterli elaborare nella maniera più opportuna.

2.5.1 Python

Python [13] è un linguaggio di programmazione general-purpose, interpretato, interattivo, object-oriented e di alto livello.

Python è uno dei linguaggi più utilizzati al giorno d'oggi. Alcune motivazioni del suo successo sono:

- la facilità con è possibile apprendere questo linguaggio: possiede parole chiave, strutture dati semplici e ha una sintassi chiara e definita;
- la facilità con cui è possibile leggere e capire il codice e quindi anche la facilità con cui è possibile effettuare delle manutenzioni;
- l'interattività con cui è permesso testare e debbuggare pezzi di codice;
- la portabilità.

Oggi Python viene spesso utilizzato nell'ambito dell'intelligenza artificiale, perché è un linguaggio di scrittura veloce ed open source e permette di catturare idee complesse con poche righe di codice. Esistono, infatti, librerie come "Keras" e "TensorFlow", che contengono molte informazioni sulle funzionalità dell'apprendimento automatico.

2.5.2 Tensorflow

Tensorflow [14] è una libreria open source per il calcolo numerico e per il machine learning su larga scala. Questa libreria riunisce una serie di modelli di machine learning e deep learning ed algoritmi per poterli rendere utilizzabili. Usa Python per fornire un API front-end per costruire le applicazioni, mentre esegue le applicazioni in C++, ad alte prestazioni.

Tensorflow offre più livelli di astrazione. Crea ed addestra modelli utilizzando l'API Keras, che semplifica il machine learning. Inoltre, consente di addestrare e distribuire facilmente un modello.

2.5.3 Keras

Keras [15] è una delle API dominanti nel settore delle reti neurali ad alto livello ed è scritta in Python. Keras è stata creata per essere user-friendly, modulare e facile da estendere. I vari moduli, come i

layer neurali, le funzioni di costo o le funzioni di attivazione, sono indipendenti tra loro e possono essere combinati per creare nuovi modelli.
I modelli sono l'idea chiave dietro Keras e ne esistono due principali in Keras:

- il **Sequential model**: è rappresentato da una pila di layer che possono essere descritti in maniera molto semplice. Questo tipo di modello è semplice ma limitato, in quanto non possono essere creati modelli complessi;
- la **Model class**: è collegata con l'API sequenziale e serve per creare modelli più complessi come i DAG. Fornisce gli stessi modelli del Sequential model, ma risulta più flessibile nell'unirli insieme.

Keras è stata sviluppata con l'obbiettivo di permettere alle persone di scrivere i propri script senza aver bisogno di imparare i dettagli presenti nel backend.
La differenza principale tra Keras e Tensorflow è che la prima non riesce a gestire i cambiamenti del modello a basso livello: per quello serve Tensorflow, anche se più complessa da capire.
Keras è stata utilizzata nel mio lavoro per creare il modello sul quale si basa la parte di addestramento del modello e ricerca della "novità" all'interno dei dati.

2.5.4 Pandas

Pandas [16] (Python Data Analysis Library) è una libreria software utilizzata in Python per la manipolazione e l'analisi dei dati. Quello che Pandas fa è di prendere dei file, ad esempio in formato "CSV" o database "sql", e crea oggetti Python con righe e colonne chiamati "dataframe" che sono simili a delle tabelle.
Pandas può essere utilizzato per:

- convertire liste Python, dizionari o array di tipo NumPy in un dataframe;
- aprire file locali, solitamente file "CSV";
- aprire file remoti o database.

Questa libreria è stata utilizzata per analizzare i dati provenienti da un file CSV che ho creato estraendo i dati da un database.

2.5.5 Numpy

Numpy [17] è una libreria che è composta da array multidimensionali ed una collezione di routine per poter processare questi array. Il cuore del funzionamento di NumPy sono gli "ndarray". A differenza delle normali liste in Python, questi elementi devono essere tutti dello stesso tipo.

All’interno dell’attività di tirocinio, questa libreria è stata utilizzata principalmente per eseguire operazioni all’interno dei dataframe, come ad esempio le medie o le operazioni di casting dei valori.

Parte II

Sviluppo ed Implementazione

Capitolo 3

Progettazione

Nella prima parte del capitolo si andrà a vedere come sono stati pensati e strutturati i vari algoritmi che permettono a due segnali di essere confrontati. Nella seconda parte, invece, si andrà a mettere in risalto l'analisi dei sensori vibrazionali installati su macchinari produttivi di una cartiera e l'implementazione di una rete neurale che permetta di andare a rilevare una novità all'interno dei dati, che non per forza dev'essere un'anomalia.

3.1 Algoritmi di confronto tra curve

Questa prima attività è stata svolta per Proteo Engineering SRL per poter individuare eventuali anomalie che si possono verificare su segnali provenienti da sensori posizionati sulle macchine e segnalare il fenomeno con allarmi.
Si hanno due segnali:

- uno di **riferimento;**
- uno di **confronto**.

Ognuno di questi segnali è formato da 1024 valori che vengono campionati ogni 3 secondi. La curva di riferimento viene calcolata sui primi dieci campionamenti. La curva di confronto viene rilevata dal sensore in tempo reale e poi comparata con quella di riferimento: se il valore restituito, calcolato dai vari algoritmi che verranno analizzati, è superiore ad una certa soglia, verrà segnalato un allarme, altrimenti l'analisi continua.
Durante i vari test da me effettuati, in una prima parte si è considerato un campione di valori molto ridotto per testare più approfonditamente la correttezza della soluzione adottata e per scegliere con più accuratezza il metodo che verrà poi utilizzato all'interno del progetto Vibromet. Nella seconda parte, invece, sono state considerate sequenze di 1024 valori che mi permettessero di testare i vari pezzi di codice nella loro completezza, in modo tale da avere casistiche il più simili possibili alla realtà applicativa.

3.1.1 KS test

Il primo metodo preso in esame utilizza il Kolmogorov-Smirnov test [18]. Come spiegato nel capitolo 2, è un test non parametrico utilizzato per decidere se un campione deriva o meno da una certa

distribuzione. Questo metodo può essere utilizzato per il confronto tra due distribuzioni perché il numero di campioni rimane invariato insieme alla frequenza, quindi si avrà che il campione i-esimo corrisponderà sempre alla medesima frequenza.

Per implementare questo metodo, si è utilizzata una funzione (*ks_2samp*) presente all'interno della libreria "*Scipy*". Questa funzione calcola il valore assoluto della differenza delle due curve in ingresso. Vengono quindi definite le ipotesi di validità o meno del risultato:

- ipotesi nulla: i due campioni sono da considerare provenienti dalla medesima distribuzione;
- ipotesi alternativa: i due campioni provengono da distribuzioni differenti;
- livello di significatività: è rappresentato da un fattore alpha ed è il valore critico che viene confrontato con il valore statistico restituito dal test. Solitamente, il fattore alpha corrisponde a 0,05.

Dalla funzione Python vengono restituiti due valori:

```
KstestResult(statistic=0.4, pvalue=0.41752365281777043)
```

Figura 3.1: Risultati del KS test

Ora si deve verificare che i risultati ottenuti si possano considerare accettabili. Il valore statistico deve essere confrontato col valore del critical value di default, che corrisponde a 0,05.

In questo caso possono avvenire due casistiche:

- **statistic > critical value**: i due campioni provengono dalla stessa distribuzione e quindi possono essere confrontati tra loro;
- **statistic < critical value**: i due campioni provengono da due distribuzioni diverse e quindi non possono essere considerati confrontabili.

Per quanto riguarda il *P-value*, questo valore tanto più è vicino ad uno quanto più le due curve si possono considerare simili.

Per quanto valida, si è deciso di non utilizzare questa tecnica di confronto in quanto nonostante restituisse valori discreti, non si riusciva a capire quanto davvero questi segnali fossero effettivamente simili tra loro; inoltre, con alcuni valori, per quanto le due curve risultassero simili, venivano restituiti valori non accettabili e non congrui con quello che ci si aspettava.

3.1.2 Metodo SQM

Altro metodo preso in considerazione per poter confrontare due campioni di valori è quello dello **Scarto Quadratico Medio (SQM)**. Lo SQM è un indicatore di dispersione di una distribuzione di valori. Viene calcolato come segue:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}}$$

Figura 3.2: Deviazione standard o Scarto Quadratico Medio

dove $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ e rappresenta la media aritmetica del campione X, quindi il valor medio.

Prima di aver effettuato queste operazioni, sono stati calcolati due valori:

- un massimo assoluto;
- un minimo assoluto.

Questi vengono calcolati tra tutti i punti dei segnali dati in ingresso, in modo tale da poter aver una metrica di confronto omogenea, che permettesse di confrontare tutti i dati. Se non si fossero presi in considerazione tali valori, non sarebbe stato possibile confrontare tra loro i risultati delle varie prove, in quanto cambiando entrambe le curve e non si ha un riferimento fisso.

Successivamente, viene effettuata la differenza tra i valori delle due curve nel punto i-esimo. Su questi valori, andrà calcolata una media per poter poi calcolare il singolo scarto (media delle differenze – differenza calcolata nel punto i).

I valori degli scarti calcolati, vengono poi normalizzati attraverso un rapporto con il massimo ed il minimo assoluti, in modo tale da avere tutti i valori degli scarti compresi tra 0 ed 1.

Come ultimo passaggio, e come riportato anche dalla formula, si va a calcolare la radice dello scarto quadratico medio: si calcola il quadrato, si sommano e si dividono gli scarti per la dimensione del campione analizzato e viene ritornato all'utente questo valore. In questo caso, più il valore ritornato sarà prossimo all'uno, più i due campioni saranno simili.

Non è stato scelto questo metodo da utilizzare all'interno del software Vibromet dell'azienda perché non è risultato possibile calcolare tutti i vari massimi e minimi assoluti e far sì che tutti i vari valori

ritornati potessero essere confrontati tra loro, in quanto la normalizzazione poteva risultare diversa ogni 1024 valori.

3.1.3 Metodo Integrale

Ultimo metodo preso in considerazione è stato quello di avvalersi degli integrali. Con questo metodo si è voluto calcolare il rapporto tra le due aree sottese relative alle curve di riferimento e di confronto. In una prima fase preliminare sono stati calcolati dei valori di soglia sopra e sotto la curva di riferimento. Questi valori rappresentano l'1% del valore i-esimo: in pratica è come se i punti avessero una fascia che li avvolge. In questa zona di tolleranza, i valori della curva che verrà utilizzata per il confronto, vengono considerati parte della curva di riferimento e quindi la loro differenza con quest'ultima risulta nulla.

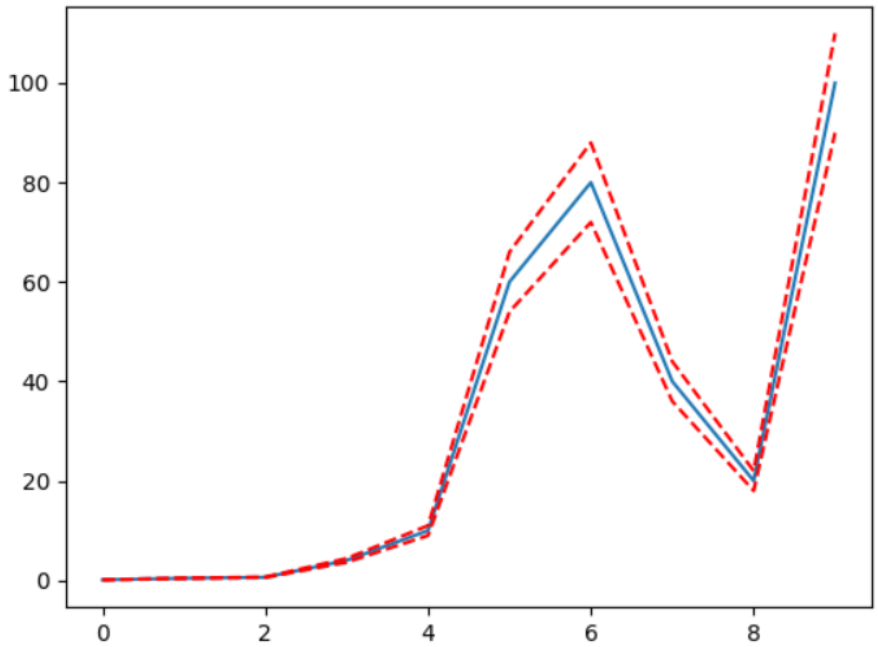


Figura 3.3: Rappresentazione della curva di riferimento e dei valori di tolleranza

Si è pensato fosse meglio un valore di tolleranza che fosse calcolato sul singolo punto e che non fosse fisso, consentendo così una maggiore deviazione se il valore del punto è maggiore.
Successivamente si procede con il calcolo dei punti esterni alle due soglie di tolleranza: se il valore i-esimo è all'esterno della zona di tolleranza, dev'essere calcolata la differenza, in valore assoluto, tra il valore della curva di riferimento e quello della curva di confronto nell'i-esima posizione.

Dopo questa fase preliminare di elaborazione dei dati vengono calcolate effettivamente le due aree attraverso l'integrale e successivamente ne viene effettuato il rapporto: più questo valore è vicino a 0, più i due segnali sono simili.

Questa è stata ritenuta la scelta migliore per lo scopo di progetto, si è deciso di implementare all'interno del software Vibromet questa soluzione. Si è inoltre deciso, su indicazione dell'azienda, di ampliare le informazioni restituite all'utente in modo tale da avere a disposizione più informazioni che permettessero di avere un quadro più ampio della situazione. Sono stati quindi aggiunti:

- il valore di non similarità in percentuale;
- i contributi superiori e inferiori rispetto alla curva di riferimento: in pratica, si sono separati i valori esterni alla zona di tolleranza in due, quelli superiori e quelli inferiori, cosicché si potesse capire in che proporzione avessero contribuito ad ottenere il risultato di non similarità;
- un flag: permette di capire se l'area della curva di confronto è maggiore di quella di riferimento. In questo caso il flag viene portato ad uno ed il valore di non similarità percentuale viene portato al 100%.

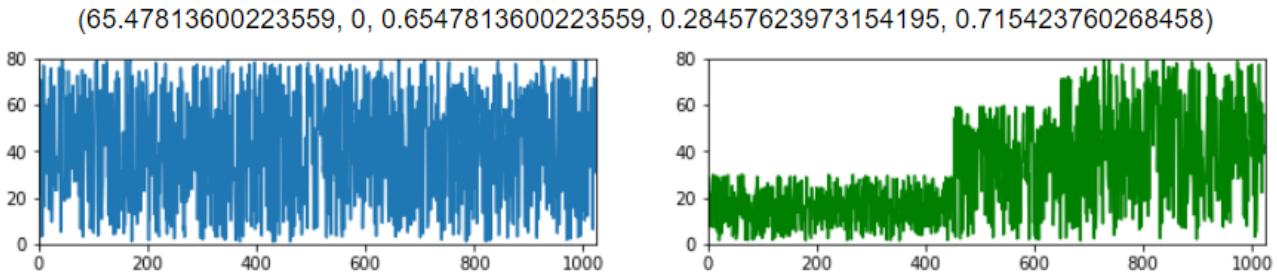


Figura 3.4: Esempio di due segnali (riferimento a sx, confronto a dx) e del l'output dell'algoritmo dell'integrale

3.2 Novelty Detection con LSTM

Nella seconda parte dell'attività di tirocinio, sono andato sviluppare un progetto riguardante l'AI, utilizzando le reti neurali.

3.2.1 Dati utilizzati

All'interno di una Cartiera sono stati installati quattro sensori da fine luglio a fine gennaio 2020. Secondo quanto riportato, due sensori hanno smesso di funzionare dopo uno o due mesi, mentre gli altri due hanno funzionato correttamente per tutta la durata della sperimentazione. I due sensori che hanno portato a termine il lavoro sono quelli relativi ai cilindri 35 e 41 che hanno raccolto rispettivamente 3649 e 3084 campioni per ciascuna frequenza.

Analizzando i vari campioni raccolti per ciascuna frequenza, si è notato che nel sensore relativo al cilindro 41, sull'asse X alla frequenza di 1200Hz, a partire da gennaio 2020 è stato rilevato un generale aumento di dispersione vibrazionale in corrispondenza di questa frequenza.

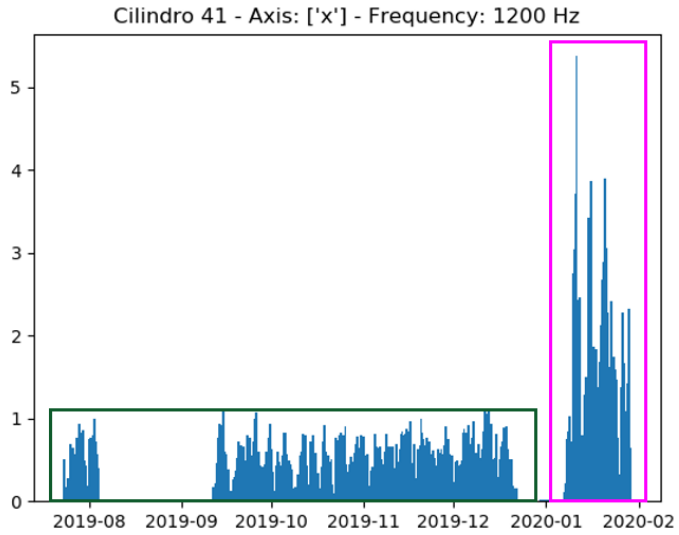


Figura 3.5: Dati relativi alla frequenza di 1200Hz sull'asse X restituiti dal sensore montato sul cilindro 41

Avendo a disposizione il database con i relativi dati riguardanti l'intero periodo, per l'analisi si è deciso di prendere in considerazione solo i valori delle frequenze relativi al cilindro 41, in particolare quella relativa ai 1200Hz che contiene informazioni rilevanti per la ricerca. Si è quindi creato uno script Python che fosse in grado di estrarre tutti i dati associati allo specifico sensore preso in esame. Dato il volume cospicuo del database, si è successivamente effettuato un'ulteriore estrazione dei dati, in modo tale che venissero salvati solo le misure di interesse.

Una rappresentazione dei dati, estratti attraverso Pandas, viene mostrata di seguito:

```
2019-08-01 00:33:48  0.1209
2019-08-01 01:36:07  0.1482
2019-08-01 02:38:07  0.1482
2019-08-01 03:39:57  0.1989
2019-08-01 04:41:58  0.5304
2019-08-01 05:43:48  0.5577
2019-08-01 06:45:58  0.5655
2019-08-01 07:48:12  0.4173
2019-08-01 08:50:20  0.3315
2019-08-01 09:52:23  0.2574
2019-08-01 10:54:14  0.4992
2019-08-01 11:56:18  0.3315
2019-08-01 12:58:20  0.5811
```

Figura 3.6: Dati estratti dal sensore

Ogni valore rilevato dal sensore è relativo ad un orario. Queste letture del sensore venivano sono state una volta all'ora.

3.2.2 Caricamento e pre-processing dei dati

In questa fase è stato utilizzato TensorFlow come backend e Keras come libreria per lo sviluppo dei modelli. Il primo passo effettuato è quello dell'estrazione dei dati. È stato quindi creato un dataframe attraverso Pandas in modo tale che potesse contenere tutti i valori relativi alla frequenza di 1200Hz. Questo dataframe è stato poi suddiviso in due: la prima parte prende i dati relativi al primo periodo di installazione del sensore (dal 01.08.2019 al 31.10.2019) e serve come dataset per addestrare la rete, mentre la seconda parte seleziona i valori che partono da inizio novembre 2019 fino ad arrivare al 31.12.2020 e questo dataset servirà come "test set" per la rete, in cui si andrà a cercare di osservare un comportamento diverso dal normale.

Successivamente i valori devono essere normalizzati in modo da rientrare in un range di valori compreso tra 0 e 1.

Le reti LSTM ricevono un input in un formato tridimensionale. Le tre dimensioni sono le seguenti:

- **sample**: rappresenta un campione di dati, nel nostro caso la dimensione del "train set" e del "test set";
- **time step**: rappresenta quante volte viene eseguita una rete neurale. Nel nostro caso è impostato ad 1;
- **feature**: rappresenta un'osservazione ad ogni time step.

Ciò significa che il layer di input si aspetta un array 3D di dati quando vengono effettuate le operazioni di “fit” e di predizione sul modello. Il “reshape” dei dati è quindi da effettuare prima di dare in pasto i valori a queste reti.

3.2.3 Neural Network Model

Come modello di rete neurale, è stato utilizzato un **autoencoder** [19]. Gli autoencoder sono un particolare tipo di rete neurale che permette di codificare i dati in una maniera efficiente. Questo tipo di rete neurale consente di apprendere una rappresentazione codificata di un set di dati, addestrando la rete ed ignorando il rumore all’interno del segnale, e successivamente riuscire a ricostruire i dati che erano stati passati precedentemente come input alla rete. Questo tipo di modello si può applicare molto bene con le immagini ad esempio.

L’idea che sta dietro all’utilizzo di un autoencoder per la Novelty Detection è quella di “addestrare” il modello sui dati ritenuti “normali” e successivamente determinare la ricostruzione dell’errore. Quando si incontreranno dei dati che sono fuori dalla norma delle precedenti rilevazioni e si proverà a ricostruirli, vedremo un incremento dell’errore dovuto alla generazione dei valori che il modello non è in grado di ricostruire in quanto non è mai stato addestrato a riconoscerli. Per costruire questo modello è stata utilizzata la libreria Keras.

L’autoencoder è stato suddiviso in cinque strati:

- La prima coppia di layer crea una rappresentazione compressa dei dati di input e questi formano l’**encoder**.
- Il layer successivo è costituito da un “repeat vector” che permette di ripetere l’input per tutti i time step del decoder.
- I due livelli finali permettono di ricostruire i dati di input.

Una volta costruito questo modello, viene istanziato e compilato utilizzando un “Adam optimizer” [20], come “neural network optimizer”, ed una media assoluta sull’errore per calcolare la funzione obbiettivo o “loss function”. Questo tipo di optimizer è un’estensione del metodo della discesa stocastica del gradiente e viene utilizzata per aggiornare iterativamente i pesi sui dati di training.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1, 1)              0
_________________________________________________________________
lstm_1 (LSTM)                (None, 1, 16)             1152
_________________________________________________________________
lstm_2 (LSTM)                (None, 4)                 336
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 1, 4)              0
_________________________________________________________________
lstm_3 (LSTM)                (None, 1, 4)              144
_________________________________________________________________
lstm_4 (LSTM)                (None, 1, 16)             1344
_________________________________________________________________
time_distributed_1 (TimeDist (None, 1, 1)              17
=================================================================
Total params: 2,993
Trainable params: 2,993
Non-trainable params: 0
```

Figura 3.7: Riassunto del modello creato

3.2.4 Model fit

Dopo aver ottenuto finalmente il modello, possiamo effettuare un'operazione di "fitting" [21], cioè verificare la bontà dell'adattamento del modello prodotto rispetto ai dati osservati. Questa operazione verrà effettuata per 100 "epoche", cioè ripetuta per 100 volte, in modo tale da avere una rappresentazione più precisa. Il "model fitting" è una misura di quanto un modello di machine learning generalizzi bene i dati su cui il modello ha effettuato il "training". Un modello "well-fitted" produce risultati più accurati.

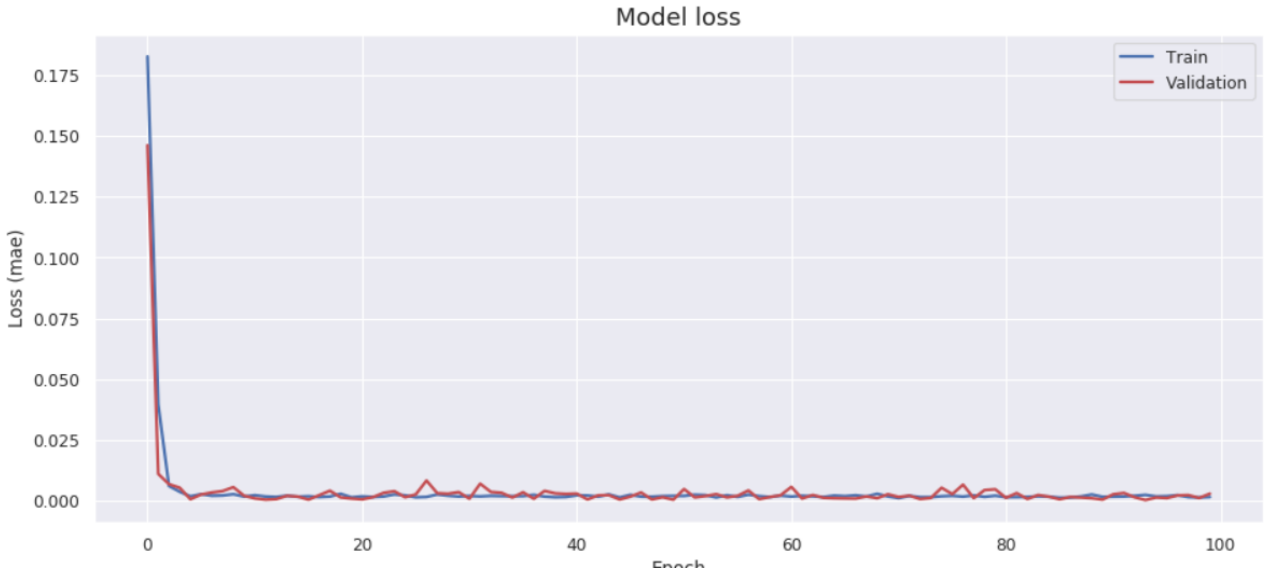


Figura 3.8: Fit del modello rispetto ai dati di train

Facendo il grafico della distribuzione dell'errore, calcolato sui dati di training, si può identificare un valore di soglia che ci permette di identificare una "novità" o un'anomalia all'interno dei dati.

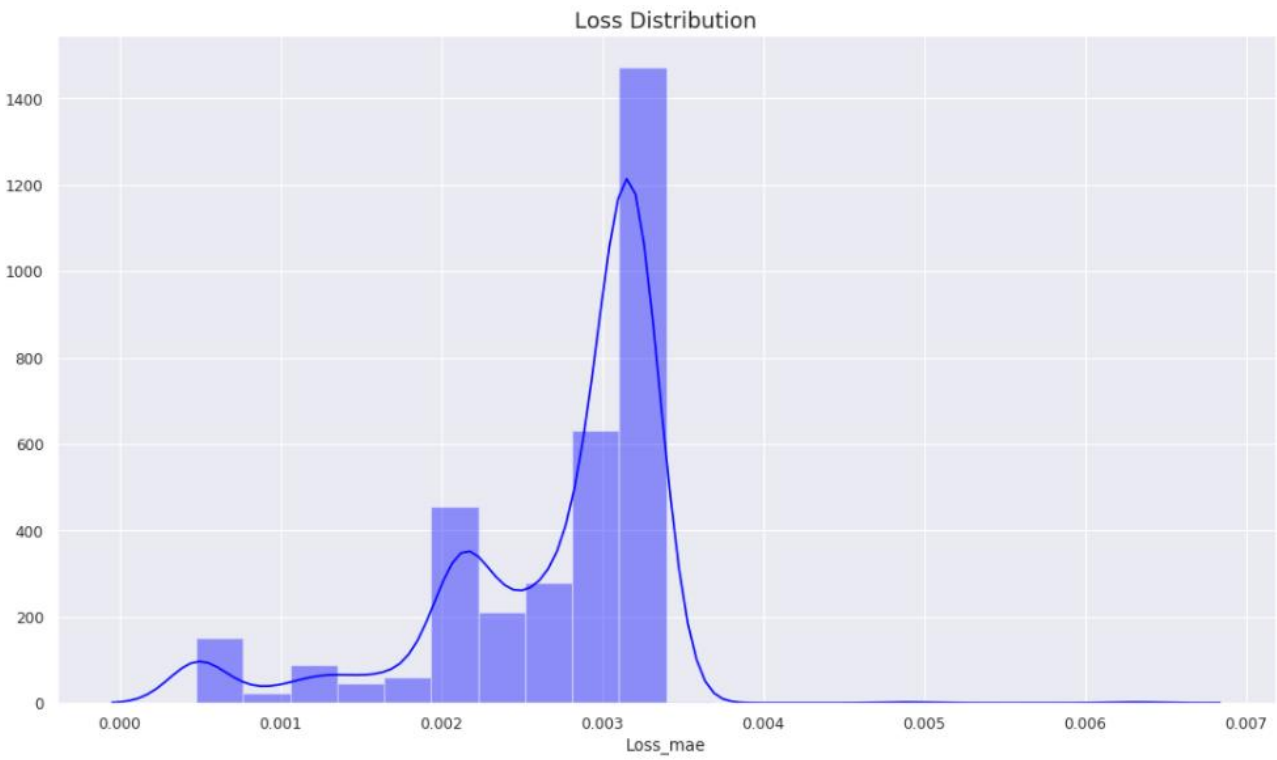


Figura 3.9: Distribuzione dell'errore

In questo caso, il valore di soglia assume un valore pari a 0.0068 ed è il valore oltre il quale saranno presenti valori che non si considerano più nella norma, ma non per forza anomali.

3.2.5 Novelty detection nel modello

La fase successiva è stata quella di ricostruire l'errore nel train e nel test dataset in modo tale da determinare quali valori all'interno dei segnali superassero effettivamente la soglia che individua la normalità. Sono stati poi uniti i due dataframe (train e test) in modo da avere una rappresentazione unica dell'intero set di dati.

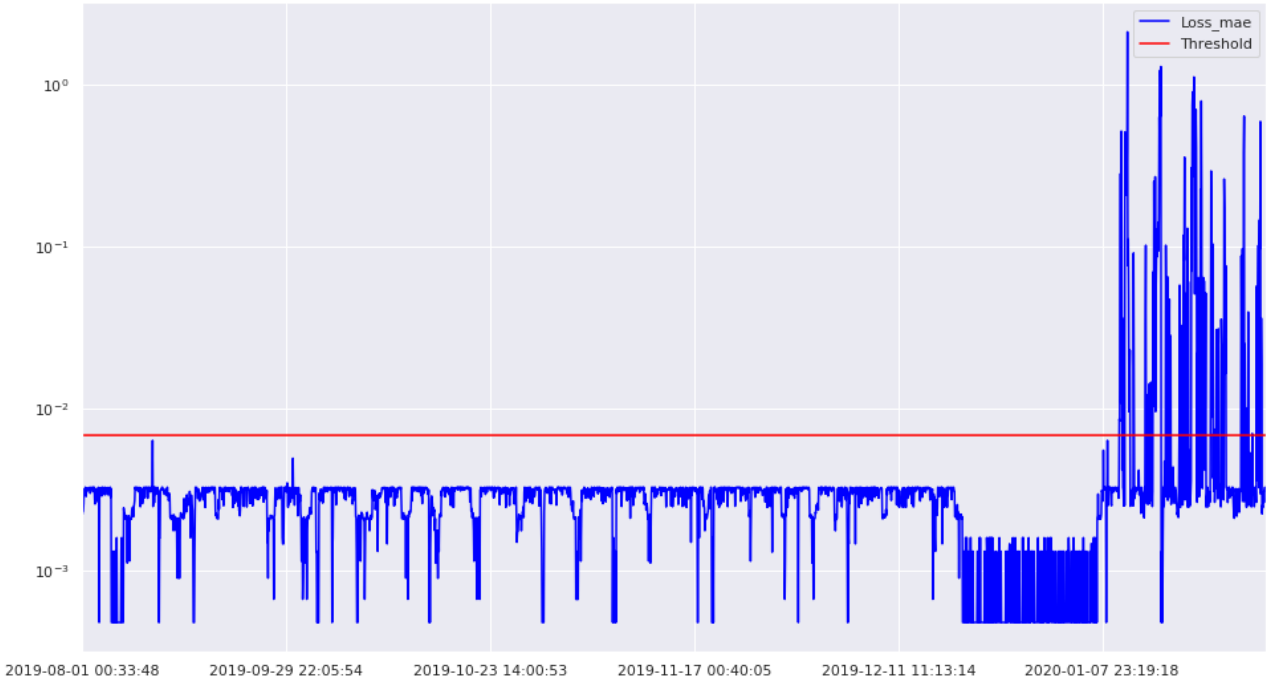


Figura 3.10: Novelty Detection

3.2.6 Risultati

Questo approccio basato su LSTM ha portato risultati interessanti. Si è riusciti a rilevare effettivamente un cambiamento all'interno dei dati, riuscendo a distinguere i dati "nella norma" e quelli che si discostavano da questa condizione. Quello che non è stato possibile fare è la predizione dell'anomalia o della novità con un certo anticipo, anche solo di un'ora prima dato che il sensore campiona i dati una volta all'ora. Non avendo un cambiamento progressivo e graduale dei valori, si riesce solamente ad individuare il punto di cambiamento, senza avere un'effettiva predizione. Questa potrebbe essere raggiungibile nel lungo periodo con una collezione di dati più lunga ed ampia.
Per questa attività di ricerca e sviluppo, si ritiene di aver ottenuto ugualmente ottimi risultati.

Capitolo 4

Implementazione

In questo capitolo si andranno ad illustrare le parti più interessanti all'interno del codice. Quello che verrà analizzato sarà l'implementazione relativa all'algoritmo utilizzato per confrontare due segnali e quella relativa all'applicazione di LSTM per la Novelty Detection.

L'implementazione del codice relativo al confronto di due segnali, è composto dalle seguenti funzioni:

- **calcolo**: la funzione principale da cui vengono chiamate le altre funzioni che eseguono i vari calcoli;
- **calcola_contributi**: calcola i valori sopra e sotto la soglia impostata dal segnale di riferimento e li restituisce;
- **calcola_val_esterni**: effettua il calcolo dei valori che escono dalla zona di tolleranza e ne restituisce i valori esterni a queste zone;
- **integrale_trap**: calcola il valore dell'integrale all'interno dei due segnali passati in ingresso e ne restituisce il rapporto.

Per quanto riguarda l'esecuzione del codice relativo alla creazione di un Autoencoder basato su LSTM, in figura 4.1 ne viene riportata una visione d'insieme.

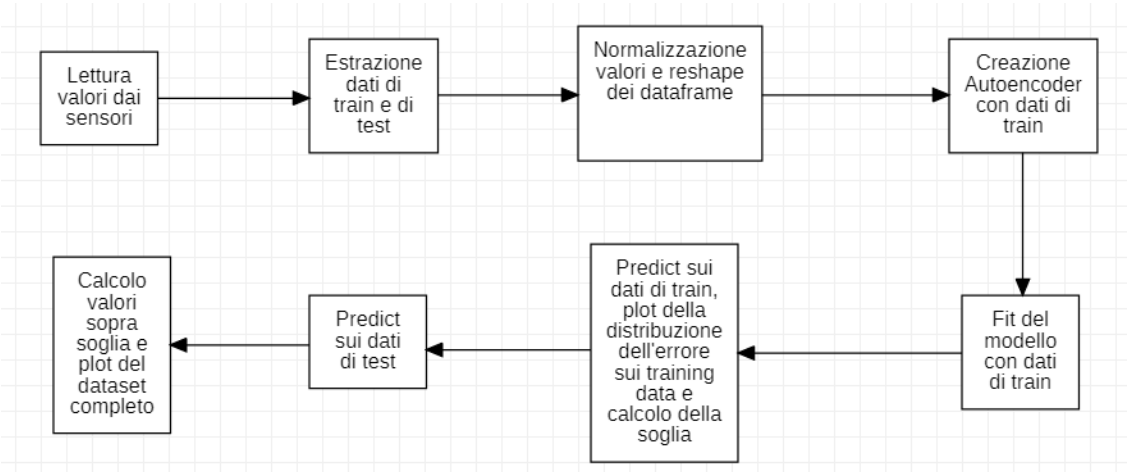


Figura 4.1: Processo schematizzato di creazione di un Autoencoder Model per Novelty Detection

4.1 Metodo di confronto di due segnali

Questo metodo, come spiegato nei precedenti capitoli, permette di ottenere una metrica di non similarità tra due segnali che vengono passati in ingresso.

```
def calcolo(rif, conf, param):
    # calcolo tolleranza
    tolleranza_sup = [x + x * 0.01 for x in rif]
    tolleranza_inf = [x - x * 0.01 for x in rif]

    valori = np.array(calcola_val_esterni(conf, tolleranza_sup, tolleranza_inf))

    # se la curva di confronto è uguale a 0, verrà ritornato sempre 0: 0/rif
    if abs(sum(valori) == 0.0):
        return 0.0, 0.0, 0.0, 0.0, 0.0
    # per evitare la divisione per 0
    elif abs(sum(rif)) == 0.0:
        rif = [x + 0.1 for x in rif]

    contributi_pos, contributi_neg = calcola_contributi(conf, tolleranza_sup, tolleranza_inf)

    area, flag, raw = integrale_trap(rif, valori, param)
    return area, flag, raw, contributi_pos, contributi_neg
```

Figura 4.2: Funzione principale per il calcolo della non similarità

Come mostrato in figura 4.2, questa è la funzione chiave per il calcolo della similarità. Nella prima parte vengono effettuate le operazioni preliminari del calcolo delle zone di tolleranza e dei valori esterni a questa zona. I valori esterni sono calcolati come differenza tra il valore del punto della zona di tolleranza ed il valore del punto della curva di confronto.

Vengono poi effettuati i controlli per evitare operazioni illegali, come la divisione per 0. Nel caso la curva di riferimento fosse costituita da valori nulli, viene sommata con un valore per evitare l'arresto preventivo del programma.

La fase successiva consiste nel calcolo dei contributi superiori e inferiori al segnale di riferimento. Questa feature è stata aggiunta per poter capire quale parte contribuisca maggiormente al valore di non similarità.

L'ultima parte consiste nel calcolo vero e proprio delle aree delle due curve e quindi il calcolo effettivo del valore di confronto.

```python
def integrale_trap(rif, conf, param=1):
    area_rif = integrate.trapz(rif)

    # normalizzo e ritorno
    area_raw = (abs(integrate.trapz(conf)) / area_rif) / param

    # valore che se a 0 notifica che la soglia non è stata superata, 1 altrimenti
    flag = 0

    # in caso in cui l'area sia maggiore di 1
    if area_raw > 1:
        area_tot = 100
        flag = 1
        return area_tot, flag, area_raw

    # ritorna la percentuale
    return area_raw * 100, flag, area_raw
```

Figura 4.3: Calcolo del rapporto tra le aree dei due segnali

Nella figura 4.3 viene mostrato il calcolo del rapporto tra le due aree dei segnali, quello di riferimento e quello di confronto. Il valore di flag è utilizzato nel caso il valore dell'area della curva di confronto fosse maggiore dell'area di quella di riferimento. In questo caso, il valore di flag viene impostato ad uno ed il valore che viene assegnato per identificare la non similarità, corrisponde al 100%, cioè le due curve non si assomigliano.

4.2 Creazione della rete LSTM

Per poter creare un una rete che riuscisse ad individuare un cambiamento all'interno dei dati, si sono utilizzate le LSTM. Prima di poter procedere con queste, sono state effettuate operazioni preliminari, come la separazione del dataset in due parti:

- parte di train;
- parte di test.

Successivamente viene seguita la normalizzazione dei valori (in modo tale che potessero essere compresi tra 0 e 1) e il reshape di questi due dataframe di valori.

```
train_set_1 = merged_data['2019-08-01 00:33:48':'2019-10-31 22:44:23']
test_set_1 = merged_data['2019-11-01 00:48:38':]


#NORMALIZE THE DATA
scaler = MinMaxScaler()
X_train = scaler.fit_transform(train_set_1) #normalizza tra 0 e 1
X_test = scaler.transform(test_set_1)
scaler_filename = "scaler_data_PROTEO"
joblib.dump(scaler, scaler_filename) #salva un file su disco


# reshape inputs for LSTM [samples, timesteps, features]
X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1]) #shape conta il numero di righe e colonne
print("Training data shape:", X_train.shape)
X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1]) #facciamo il reshape per poterlo dare in input alla rete LSTM
```

Figura 4.4: Operazioni preliminari

Il passaggio successivo è stato quello di creare un A*utoencoder Model*, che permette di apprendere una rappresentazione codificata dei dati.

```
#AUTOENCODER MODEL

def autoencoder_model(X):
    inputs = Input(shape=(X.shape[1], X.shape[2]))
    L1 = LSTM(16, activation='selu', return_sequences=True,
              kernel_regularizer=regularizers.l2(0.00))(inputs)
    L2 = LSTM(4, activation='selu', return_sequences=False)(L1)
    L3 = RepeatVector(X.shape[1])(L2)
    L4 = LSTM(4, activation='selu', return_sequences=True)(L3)
    L5 = LSTM(16 , activation='selu', return_sequences=True)(L4)
    output = TimeDistributed(Dense(X.shape[2]))(L5)
    model = Model(inputs=inputs, outputs=output)
    return model

model = autoencoder_model(X_train)
model.compile(optimizer='adam', loss='mae')
model.summary()

# fit the model to the data
nb_epochs = 100
batch_size = 10
history = model.fit(X_train, X_train, epochs=nb_epochs, batch_size=batch_size,validation_split=0.05).history
```

Figura 4.5: Definizione del modello

Come è possibile vedere in figura 4.5, il modello è rappresentato da cinque layer:

- due di input;
- uno che rappresenta la ripetizione dei valori di input per i time step del decoder;
- gli ultimi che rappresentano la ricostruzione dei dati di input.

Dopo aver creato il modello, gli viene passato il dataset di training normalizzato e viene compilato utilizzando un optimizer.

Infine, viene eseguita una "fit" del modello, in modo tale da avere una misura della bontà dei dati ottenuti rispetto a quelli dati in input al modello.

```
X_pred = model.predict(X_train)
X_pred = X_pred.reshape(X_pred.shape[0], X_pred.shape[2])
X_pred = pd.DataFrame(X_pred, columns=train_set_1.columns)
X_pred.index = train_set_1.index

scored = pd.DataFrame(index=train_set_1.index)
Xtrain = X_train.reshape(X_train.shape[0], X_train.shape[2])
scored['Loss_mae'] = np.mean(np.abs(X_pred-Xtrain), axis = 1)
#soglia = list(dict(X_pred[-1:])[0])[0]

soglia = scored['Loss_mae']
plt.figure(figsize=(16,9), dpi=80)
plt.title('Loss Distribution', fontsize=16)
ax = sns.distplot(scored['Loss_mae'], bins = 20, kde= True, color = 'blue')
plt.show()
```

Figura 4.6: Calcolo della distribuzione dell'errore sul training set

Nella figura 4.6 viene mostrato come viene calcolato il valore di soglia andando a calcolare la distribuzione dell'errore sui dati di training. La funzione predict prende in ingresso i dati di input e su questi andrà ad effettuare una predizione dei valori. Facendo il grafico di questa distribuzione, si andrà ad ottenere un valore che corrisponderà alla soglia oltre la quale si avranno valori che si discostano dalla norma.

La predict viene calcolata anche sui dati di test. Infine, vengono uniti i due dataframe, quello di train e quello di test, e si andrà ad osservare nel grafico il punto in cui viene rilevata un'effettiva novità all'interno dei dati. In figura 4.7 vengono calcolati i valori sopra soglia e viene creato un array di booleani che identifica se il punto i-esimo è o meno sopra soglia.

```
scored_train['Anomaly'] = scored_train['Loss_mae'] > scored_train['Threshold']
```

Figura 4.7: Calcolo dei valori sopra soglia

Capitolo 5

Test e sviluppi futuri

In questo capitolo verranno analizzate le prove per verificare la correttezza delle soluzioni sviluppate ed implementate nei precedenti capitoli. Nella prima parte verranno analizzati e mostrati i risultati dei test condotti su segnali formati da 1024 valori, come veniva richiesto effettivamente da Proteo Engineering SRL, in modo tale che potesse rispecchiare un esempio di realtà applicativa.
Nella seconda parte verranno analizzati i test effettuati sulla rete LSTM, nei quali si è cercato di andare a migliorare la soluzione proposta.

5.1 Testing di non similarità tra due segnali

Questa sezione riporta i file che sono stati generati per testare le varie funzionalità software. Si è deciso di creare autonomamente dei file contenenti delle successioni di valori, in modo tale da permettere di conoscere con accuratezza la conformazione del dato di input e di prevedere, dal punto di vista teorico, il comportamento ed i risultati del sistema a quel determinato input.
Se il sistema, anche dal punto di vista pratico, si comporta come atteso teoricamente, ne rafforza il risultato.
I valori ritornati sono cinque e sono i seguenti:

- il primo rappresenta un valore percentuale che esprime il grado di non similarità;
- il secondo valore rappresenta un flag che è settato a 0 di default e nel caso il valore dell'area di confronto dovesse superare quella dell'area del segnale di riferimento verrà settato ad uno;
- il terzo valore rappresenta il valore "raw" dell'area, non in percentuale;
- il quarto valore rappresenta il contributo superiore dovuto al calcolo dell'area dell'integrale;
- il quinto valore mostra il contributo inferiore.

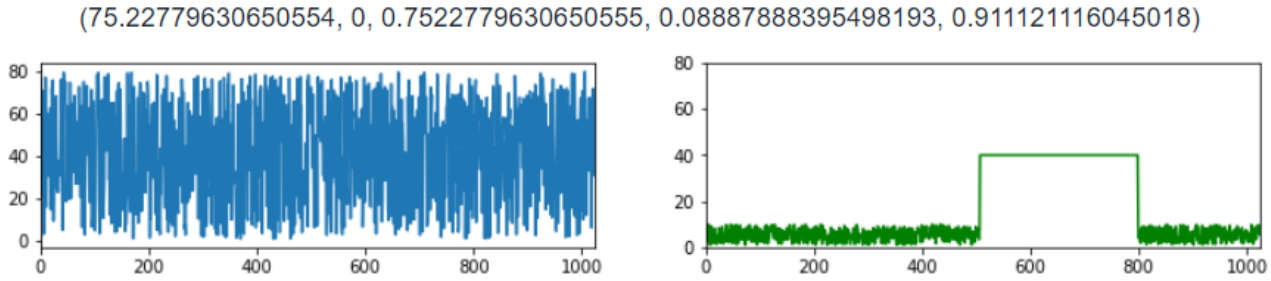


Figura 5.1: Confronto di due segnali non simili

In figura 5.1 vengono mostrati due segnali: quello a sinistra rappresenta il segnale di riferimento, mentre quello di a destra rappresenta il segnale di confronto. Il primo segnale è stato generato come una successione di valori casuali compresi tra 0 ed 80, mentre il secondo è stato creato inserendo del "rumore" alla prima parte del segnale e all'ultima, mentre al centro è presente una parte di onda quadra. In questo caso possiamo osservare, sia graficamente, che dai valori restituiti, che i due segnali in ingresso sono diversi tra loro in quanto il valore percentuale è rappresentato da un 75%: più questo valore è prossimo al 100% e più i due segnali sono diversi.
In caso di applicazione all'interno di un software, questa situazione dovrebbe generare un allarme.

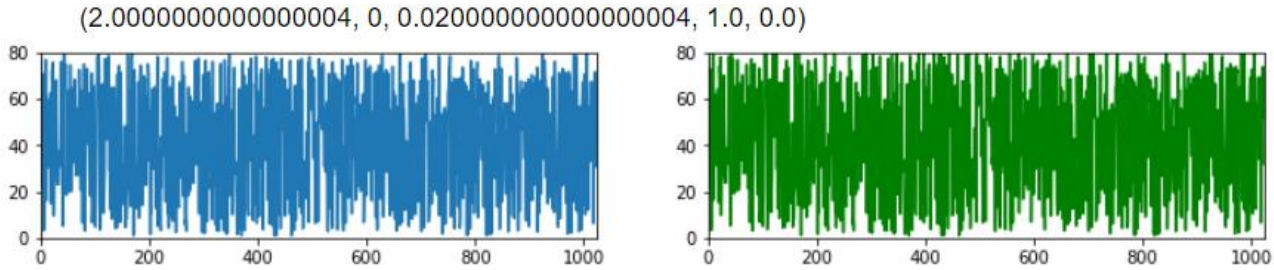


Figura 5.2: Confronto di due segnali identici

In figura 5.2 un esempio di due segnali identici. In questo caso, i due segnali sono stati generati nello stesso modo, aggiungendo solo, nel secondo segnale, un leggero "rumore". In questo caso, sia visivamente, che dai valori ritornati, possiamo concludere che i due segnali ritornati siano effettivamente uguali.

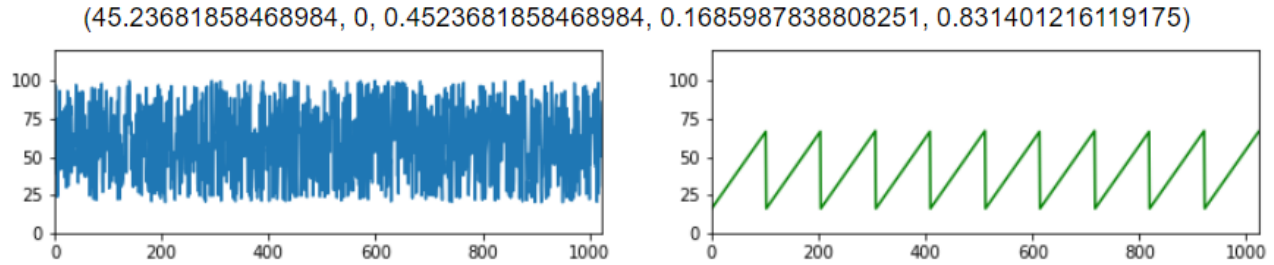


Figura 5.3: Confronto di due segnali simili

In figura 5.3 viene riportato un esempio di due segnali che possono considerarsi simili. Il calcolo dell'integrale viene effettuato sul segnale di riferimento e sul segnale di confronto che esce dalle zone di tolleranza del primo segnale. Graficamente possono risultare diverse, ma molti valori rientrano

nella zona di tolleranza e non vengono quindi considerati. Non sempre, in questi casi, i calcoli e i grafici rappresentano lo stesso risultato.
Il segnale di confronto rappresenta un'onda con andamento a dente di sega e quando confrontata con il segnale di riferimento produce i risultati sopra riportati. Il 45% come percentuale di non similarità è corretto in quanto, per come sono stati impostati i valori per la creazione dei segnali, ci si aspettava che la maggioranza rientrasse nelle zone di tolleranza o comunque in prossimità di queste. In questa situazione non dovrebbe presentarsi una condizione di allarme del sistema, ma questo dipende dal tipo di macchinario sul quale è montato e con che frequenza si ripetono certi segnali.

Dai risultati di test ci si ritiene soddisfatti di come è stato sviluppato il software.

5.2 LSTM testing

Avendo riportato buoni risultati con la creazione di un Autoencoder model formato da LSTM, si è deciso di apportare delle modifiche ai valori dati in pasto al modello. Queste modifiche sono state apportate su:

- dimensione del dataset di train e di test;
- cambiamento dell'optimizer per la compilazione del modello;
- dimensionamento del numero di neuroni.

5.2.1 Dimensionamento dei dati di train e di test

In questa fase si è provato ad aumentare il numero di valori di train forniti in input all'encoder. Aumentando il numero dei valori di train dati in pasto alla rete, si ipotizzava si potesse avere una diminuzione dell'errore all'interno dei dati, ma questo non è avvenuto, lasciando inalterato l'errore nei dati.

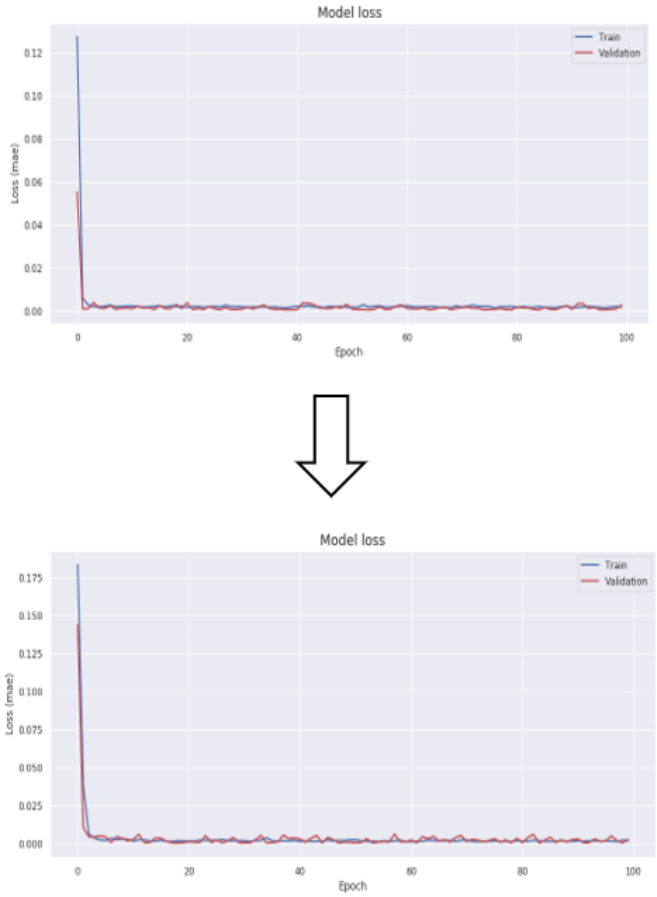


Figura 5.4: Confronto errore con cambio dei dataset

5.2.2 Cambiamento dell'activation function

In questa fase si è provato a cambiare l'activation function [22] che viene utilizzata all'interno di ogni nodo della rete LSTM e definisce l'output di quel nodo su un dato set di input. In pratica le activation function convertono input lineari in output non lineari che permettono una facilitazione dell'apprendimento, in quanto possiedono una complessità maggiore e possono rappresentare più informazione.

Esistono vari tipi di funzioni di attivazione, come ad esempio la *Selu* [23] e la *Softmax Activation Function* [24]. Il primo tipo di funzione di attivazione è capace di apprendere in modo migliore e più velocemente rispetto alle altre, mentre la *Softmax Activation Function* trasforma ogni vettore di numeri che riceve in input in un set di probabilità, in modo da cercare di capire quale di questi valori si andrà a ripetere con più probabilità. Questa funzione di attivazione si applica bene a problemi di classificazione come, ad esempio, quello di distinguere le mail spam e non spam.

Come si può notare in figura 5.5, la *Softmax Activation Function* non rappresenta bene i dati che gli sono passati in input e quindi presenta un valore di errore all'interno dei dati piuttosto elevato rispetto alla prima funzione di attivazione. Si è deciso quindi di continuare con la Selu Activation Function a discapito della Softmax.

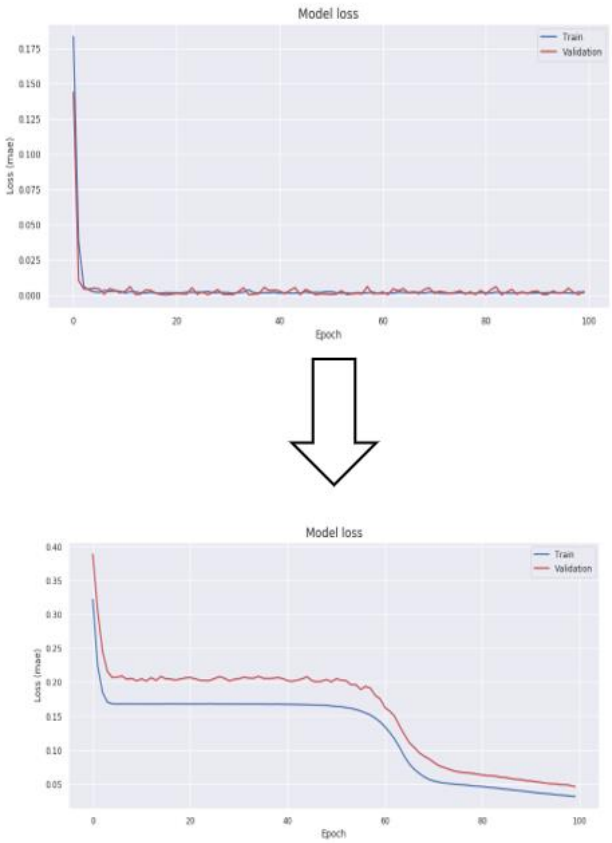


Figura 5.5: Confronto errore con cambio di activation function

5.2.3 Dimensionamento del numero di neuroni

Durante questa fase sono stati effettuati dei test cambiando il numero di neuroni all'interno di ogni layer LSTM dell'encoder. Aumentando il numero di neuroni, l'errore non tendeva a diminuire in maniera significativa. Dimezzando il numero di neuroni all'interno della rete, si è vista una tendenza all'aumento dell'errore.

La condizione iniziale del sistema è così strutturata:

- sedici neuroni nel primo e nell'ultimo layer;
- quattro neuroni nei layer centrali.

In questo caso si è ritenuto che questa condizione fosse la migliore all'interno dei singoli nodi in quanto probabilmente si è raggiunta una condizione di saturazione oltre la quale non è possibile migliorare le prestazioni del sistema.

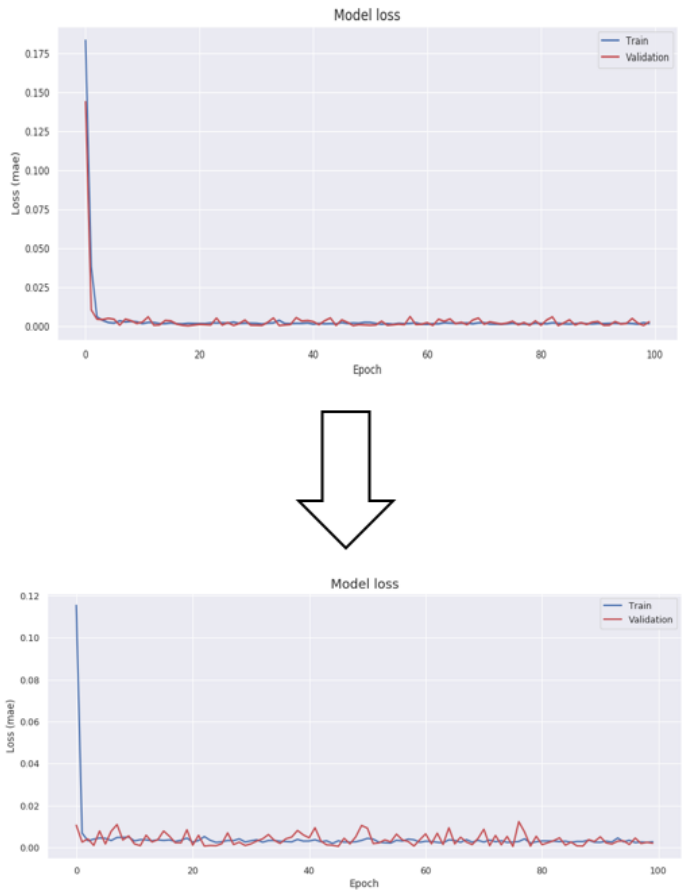


Figura 5.6: Confronto errore con modifica del numero di neuroni

Conclusioni

In questo elaborato sono stati utilizzati metodi di Intelligenza Artificiale e funzioni che sono state studiate durante il corso di laurea. Sono state analizzate ed implementate tecniche di Novelty ed Anomaly Detection, che hanno permesso di rilevare i cambiamenti all'interno dei set di dati, e tecniche che consentito di avere una metrica di non similarità tra due segnali.

La ricerca e lo sviluppo di questi metodi, ha permesso all'azienda di utilizzarli all'interno di un progetto più ampio, il Vibromet. Il mio lavoro in questo progetto è continuato anche dopo l'attività di tirocinio, andando a validare le varie parti del software che non erano mai state testate, come la gestione dei singoli allarmi rilevati dai sensori vibrazionali e la validazione del segnale FFT restituito.

Quello che in futuro si spera di ottenere è un software capace di rilevare con un certo anticipo quello che avverrà all'interno dei macchinari, tramite un attento storage dei dati ed analisi degli stessi.

Proteo Engineering SRL investe molto in progetti all'avanguardia, per fornire soluzioni che possano portare le aziende verso un futuro di innovazione e sviluppo tecnologico.

Bibliografia e Sitografia

[1]https://it.qwe.wiki/wiki/Fast_Fourier_transform, http://ens.di.unimi.it/dispensa/cap5.pdf, http://www.mat.uniroma3.it/didattica_interattiva/aa_11_12/am310/10.fourier.pdf

[2] https://it.qwe.wiki/wiki/Discrete_Fourier_transform

[3] https://it.qwe.wiki/wiki/Cooley%E2%80%93Tukey_FFT_algorithm

[4]https://www.proteoeng.com/wpcontent/uploads/2019/10/PROTEO_BROCHURE%20VIBROMET.pdf, Pro_Vib_01.pdf

[5] https://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm, https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test, https://www.graphpad.com/guides/prism/8/statistics/interpreting_results_kolmogorov-smirnov_test.htm

[6] https://www.spindox.it/it/blog/anomaly-detection-larte-di-riconoscere-linatteso/, https://www.anodot.com/blog/what-is-anomaly-detection/,

[7] https://scikit-learn.org/stable/modules/outlier_detection.html

[8]https://www.intechopen.com/books/fault-diagnosis-and-detection/evaluation-of-novelty-detection-methods-for-condition-monitoring-app

[9] https://en.wikipedia.org/wiki/Artificial_neural_network

[10] https://medium.com/@rinu.gour123/artificial-neural-network-for-machine-learning-structure-layers-2a275f73f473#:~:text=The%20structure%20of%20a%20neural,architecture'%20or%20'topology'.&text=The%20simplest%20structure%20is%20the,is%20equal%20to%20the%20input.

[11] https://en.wikipedia.org/wiki/Recurrent_neural_network, https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce, https://andreaprovino.it/rnn-recurrent-neural-network/,

[12] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13]https://www.tutorialspoint.com/python/index.htm#:~:text=Python%20is%20a%20high%2Dlevel,syntactical%20constructions%20than%20other%20languages.

[14] https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html#:~:text=Created%20by%20the%20Google%20Brain,way%20of%20a%20common%20metaphor.

[15] https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html, https://www.upgrad.com/blog/the-whats-what-of-keras-and-tensorflow/

[16] https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673#:~:text=Pandas%20stands%20for%20%E2%80%9CPython%20Data%20Analysis%20Library%20%E2%80%9D.&text=What's%20cool%20about%20Pandas%20is,Excel%20or%20SPSS%20for%20example.

[17] https://en.wikipedia.org/wiki/NumPy

[18] https://medium.com/@abhyankar.ameya/kolmogorov-smirnov-two-sample-test-with-python-70c309107c78

[19] https://towardsdatascience.com/lstm-autoencoder-for-anomaly-detection-e1f4f2ee7ccf, https://en.wikipedia.org/wiki/Autoencoder

[20] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[21] https://www.datarobot.com/wiki/fitting/

[22] https://en.wikipedia.org/wiki/Activation_function

[23] https://towardsdatascience.com/gentle-introduction-to-selus-b19943068cd9

[24] https://www.upgrad.com/blog/types-of-activation-function-in-neural-networks/, https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60