ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Corso di laurea in Ingegneria Meccanica

Valutazione delle prestazioni delle Reti Neurali Ricorrenti di tipo LSTM nella stima di indici sintetici di combustione per motori endotermici alternativi

Tesi di laurea in Laboratorio Di Motori A Combustione Interna

Relatore:
Prof. Alessandro Brusa

Presentata da:
Leonardo Liverani

ANNO ACCADEMICO 2023/2024

Alla mia famiglia,

Eleonora,

Paolo

e Lorenzo.

Abstract

Lo scopo di questo elaborato di tesi è la creazione di modelli, nel nostro caso sviluppati tramite Matlab, per la predizione degli indici di combustione in un motore ad accensione comandata. Questi modelli sono basati sul machine learning, ovvero apprendimento automatico, il quale può essere considerato una branca dell'intelligenza artificiale. Un sottoinsieme del machine learning è costituito dalle reti neurali, esse sono un algoritmo basato su nodi, che vengono chiamati neuroni e sono in grado di risolvere problemi come il riconoscimento di immagini o la traduzione automatica. Nel particolare, il modello da me creato è basato su reti neurali ricorrenti (RNN) di tipo LSTM (Long Short-Term Memory), dette a lungo termine. Il modello è in grado di predire l'andamento del MFB50, un indice di combustione che esprime l'angolo in cui è bruciata il 50% della miscela, considerando come input quattro variabili: i giri motore (RPM), l'anticipo d'accensione, il carico ed infine il lambda. Al modello vengono passati dei dati sottoforma di file Excel registrati in laboratorio in cui queste variabili cambiano al variare delle condizioni di funzionamento del motore, i dati di train ovvero quelli usati per allenare il modello sono dati stazionari, mentre i dati di test usati per validare il modello sono formati da due profili di guida e sono quindi dati dinamici. Così facendo il modello è in grado di allenarsi e di predire l'andamento del MFB50 per cicli di guida che vengono utilizzati come test, confrontando il valore di MFB50 predetto con quello reale e ottenendo un certo errore.

Indice

Lista delle figure

Capitolo 1

Introduzione al machine learning e alle reti neurali

1.1 Machine learning

Il machine learning o apprendimento automatico è un ramo dell'intelligenza artificiale, grazie ad esso i computer sono in grado di apprendere dai dati e migliorare su una specifica funzione, è basato su cinque fattori fondamentali: i dati, il modello, l'algoritmo, il training ed infine il test. I dati sono l'aspetto cruciale del machine learning , vengono utilizzati per addestrare il modello e possono essere in tabelle come dati numerici o non strutturati nel caso di immagini o audio; il modello è una rappresentazione matematica in grado di fare previsioni su nuovi dati; l'algoritmo è un metodo usato per trovare relazioni nei dati e formare il modello; il training è il processo in cui l'algoritmo apprende grazie ai dati, qui vengono calibrati i parametri del modello per minimizzare l'errore; il test ha lo scopo di valutare la capacità del modello sulla predizione di nuovi dati. Un altro aspetto di elevata importanza è dato dal problema di overfitting ed underfitting; l'overfitting si presenta nel caso in cui il modello è troppo complesso, adattandosi perfettamente ai dati di training, ma creando elevati errori nella predizione su nuovi dati. L'underfitting, al contrario, si verifica quando il modello è troppo semplice, non riuscendo a catturare le relazioni tra i dati, portando ad elevati errori sia sul training che sul test.

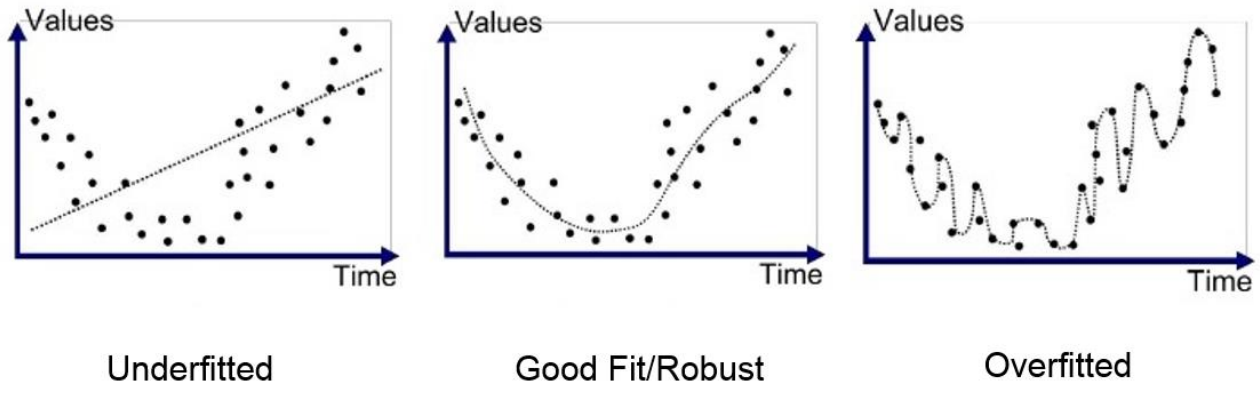


Figura 1.1: Rappresentazione di tre modelli differenti, partendo da sinistra notiamo l'underfitting, al centro è presente un modello giustamente calibrato ed infine a destra l'overfitting.

Esistono tre categorie principali in cui è suddiviso il machine learning:

- L'apprendimento supervisionato, in questa tipologia i dati di training sono costituiti sia dagli input che dagli output corretti; a sua volta l'apprendimento supervisionato può essere diviso in due sottoproblemi durante l'elaborazione dei dati: la classificazione, grazie la quale, utilizzando un algoritmo si è in grado di assegnare con elevata precisione i dati dei test in specifiche categorie; la regressione viene invece utilizzata per comprendere la relazione tra variabili dipendenti e indipendenti, l'obiettivo è quindi quello di prevedere un valore continuo, ecco perché viene spesso utilizzata per la previsione dei prezzi o delle vendite.

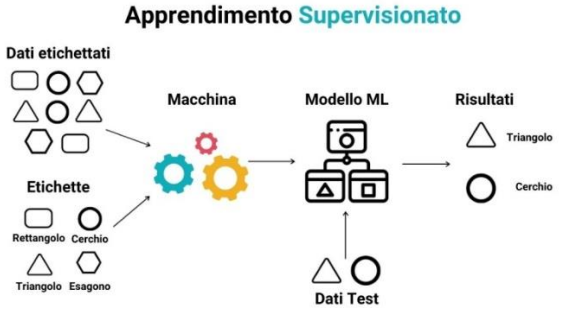


Figura 1.2: Schema dell'apprendimento supervisionato.

- L'apprendimento non supervisionato è caratterizzato dall'assenza di dati etichettati, in cui l'algoritmo ha lo scopo di trovare schemi e relazioni tra i dati, questo può essere fatto raggruppando gli elementi simili, questo processo è detto clustering, o riducendo la dimensionalità cioè il numero di variabili per semplificare l'analisi; l'apprendimento non supervisionato è spesso utilizzato come metodo esplorativo, per analizzare le relazioni tra i dati, senza avere un obiettivo di previsione.

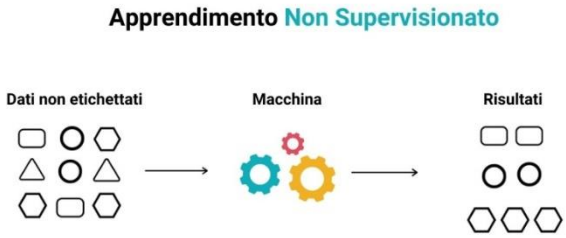


Figura 1.3: Schema dell'apprendimento non supervisionato.

- L'apprendimento per rinforzo è una tecnica di apprendimento, il cui obiettivo è quello di creare degli agenti in grado di compiere delle scelte atte a portare a termine determinati compiti tramite le relazioni con l'ambiente in cui sono immersi.

Il machine learning è quindi un approccio alla modellazione empirica particolarmente promettente grazie al quale i sistemi sono in grado di migliorarsi con l'esperienza, data la sua elevata capacità di analizzare dati in grandi quantità, fare previsioni future e trovare relazioni e strutture, viene utilizzato in molti campi come il riconoscimento delle immagini in sistemi come quelli di Google Photos per riconoscere oggetti o volti, ma anche per diagnosi medica o in ambito finanziario per fare previsioni di mercato.

1.2 Reti neurali

Le reti neurali, anche dette *Artificial Neural Networks* (ANN), possono essere considerate un sottoinsieme del machine learning, la loro architettura e il loro funzionamento è molto simile a quello del cervello umano, sono costituite da una rete di nodi interconnessi, detti neuroni, i quali sono organizzati in tre strati:

- **Strato di input** *(input layer)*: è il primo dei tre strati, il suo scopo è quello di ricevere i dati grezzi in entrata, il numero di neuroni in questo strato sarà quindi uguale al numero di caratteristiche nell'input.
- **Strati nascosti** *(hidden layers)*: sono gli strati intermedi, compresi tra quello di input e quello di output, in questi strati avviene la maggior parte dell'elaborazioni computazionali della rete. Il numero degli strati intermedi varia in base al tipo di rete neurale, come anche il numero di neuroni per strato, che sono considerati iperparametri della rete.
- **Strato di output** *(output layer)*: è lo strato finale della rete il cui compito è quello di produrre il risultato finale, ricevendo le informazioni processate dagli stati precedenti; come negli altri strati il numero di neuroni può variare in base al problema considerato.

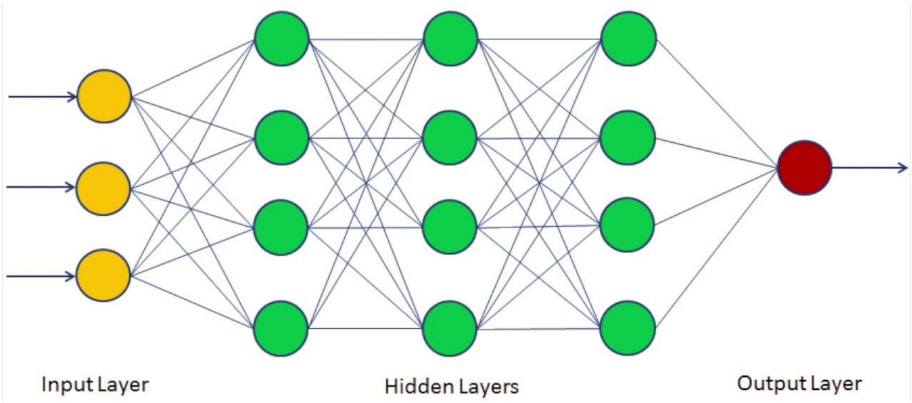


Figura 1.4: Architettura di una rete neurale artificiale.

Ciascun neurone è connesso al neurone dello strato successivo tramite una connessione pesata, questo peso determina l'importanza del collegamento tra due neuroni, nel processo di training questi pesi tra neuroni vengono continuamente aggiornati in modo da rendere minimo l'errore tra l'output della rete e quello reale.

Ad ogni neurone è applicata una funzione di attivazione per determinare l'output che trasmetterà ai neuroni successivi, tra le più comuni funzioni di attivazione troviamo la sigmoide:

$$f(x)=\frac{1}{1+e^{-x}} \quad (1.1)$$

che comprime l'output in un range tra 0 ed 1; altrimenti la tangente iperbolica:

$$f(x)\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (1.2)$$

che comprime l'output in un range tra -1 ed 1.

Nella fase di propagazione in avanti *(forward propagation)* i dati passano dallo strato di input a quello di output, passando per gli strati nascosti in cui all'input vengono moltiplicati i pesi associati, viene sommata una costante ed infine è applicata la funzione di attivazione per determinare l'output. Nella fase di addestramento *(training)* viene utilizzata la tecnica opposta ovvero la propagazione all'indietro *(backpropagation)* in modo da aggiornare i pesi della rete e ottimizzare l'errore tra l'output reale e quello predetto.

Sono presenti diversi tipi di reti neurali, ognuna caratterizzata da una diversa architettura in base al problema da risolvere; le reti neurali feedforward *(FNN)* sono considerate le più semplici perché i dati fluiscono in un'unica direzione, dallo strato di input all'output. Un loro difetto è dato dal fatto che sono prive di una memoria interna, nel momento in cui un input è stato elaborato e viene calcolato il risultato, la rete non lo considera più, per questo motivo non sono adatte a dati con dipendenze temporali, gli input vengono considerati indipendenti gli uni dagli altri e vengono elaborati in isolamento senza memoria del passato.

Nello studio di dati basati su serie temporali o sequenziali è invece necessario tenere conto della serie temporale, in quanto ogni dato ha una relazione con quello precedente e quello futuro, le reti neurali ricorrenti o *recurrent neural network (RNN)* sono in grado di risolvere questo problema. Le RNN hanno connessioni tra i neuroni che permettono di ricevere input da neuroni di strati precedenti o dello stesso strato, andando così a creare dei cicli che consentono alla rete di memorizzare informazioni su dati passati. Per i motivi sopra elencati le FNN vengono principalmente utilizzate per problemi di classificazione delle immagini, in cui i dati in input non hanno dipendenze temporali; le RNN entrano in gioco in problemi di traduzione automatica o generazione di testo, in cui l'ordine degli input è estremamente rilevante.

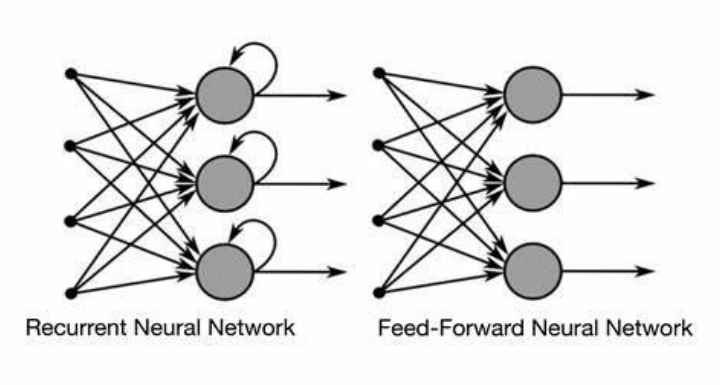


Figura 1.5: Confronto tra RNN a sinistra e FNN a destra.

1.3 Recurrent Neural Network

Siccome l'obiettivo di questo elaborato è la predizione di serie temporali di dati, ovvero valori registrati in continuo nel dominio del tempo, è utile concentrarsi sulle RNN. Come anticipato in precedenza le RNN sono dotate di una memoria, grazie la quale riescono a conservare informazioni su dati precedenti, rendendole utili nel caso di analisi di dati in cui l'ordine degli input è cruciale o in serie temporali. In questa tipologia di rete, i neuroni oltre che ricevere input dallo strato precedente, li possono ricevere anche da sé stessi o da neuroni dello stesso strato, grazie a queste connessioni è possibile avere uno strato nascosto che si aggiorna e migliora nel tempo.In una rete RNN ogni neurone oltre che elaborare un input x_t all' istante *t*, elabora anche il proprio stato nascosto h_{t-1} derivante dall'istante precedente, l'output h_t di ogni neurone è una funzione dell'input in quel istante e dello stato nascosto precedente. Si può avere un'idea più chiara di questo concetto guardando la figura 1.6 in cui si può notare che all'ingresso di ogni neurone è presente un input x_t e lo stato nascosto precedente h_{t-1}.

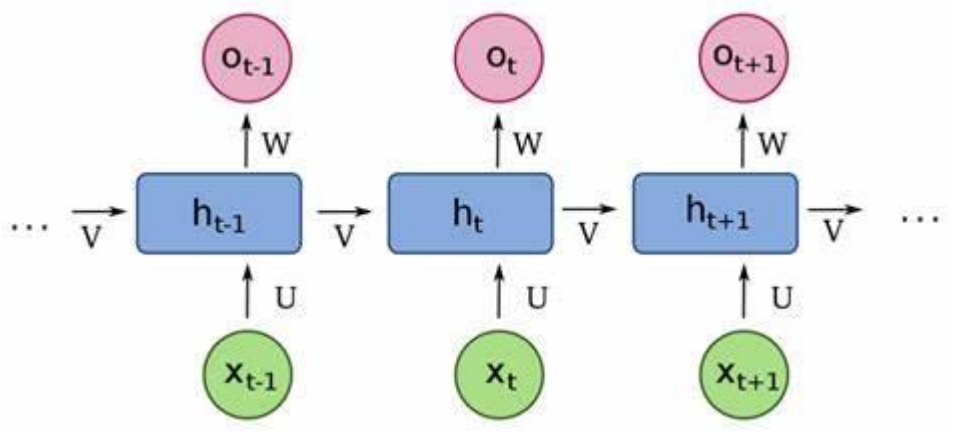


Figura 1.6: Architettura di una rete RNN.

L'output h_t di ogni neurone può essere espresso tramite la seguente formula:

$$h_t = f(W_{xh}\, x_t + W_{hh} h_{t-1} + b_h) \quad (1.3)$$

In cui:

- $\boldsymbol{W_{xh}}$ rappresenta la matrice dei pesi tra l'input e lo strato nascosto.
- $\boldsymbol{W_{hh}}$ rappresenta la matrice dei pesi tra lo strato nascosto nell'istante attuale e quello precedente.
- $\boldsymbol{b_h}$ rappresenta il bias.
- $\boldsymbol{f}$ rappresenta la funzione di attivazione.

Analizzando la formula 1.3 notiamo che è formata da un primo termine corrispondente alla matrice dei pesi dell'input W_{xh} moltiplicato per l'input corrente x_t, i pesi della matrice indicano quanto ogni elemento della matrice influisce lo strato nascosto corrente; il secondo termine è costituito dal prodotto tra la matrice dei pesi ricorrenti W_{hh} e lo strato nascosto precedente h_{t-1} , in questo caso i pesi determinano quanto le informazioni passate devono rimanere in memoria nel calcolo del nuovo strato nascosto; entrambi i valori delle matrici vengono costantemente aggiornati durante il processo di training per minimizzare l'errore della rete tramite il processo di BPTT. L'ultimo termine b_h detto bias viene sommato al risultato dei due prodotti sopra descritti in modo da spostare la funzione di attivazione, così facendo la rete può adattarsi meglio ai dati, nel caso di tutti i pesi nulli e in assenza del bias tutte le trasformazioni fatte dalla rete passerebbero dall'origine, grazie alla sua presenza si ha una maggiore flessibilità nell'elaborazione dei dati, anche il valore del bias viene aggiornato durante il training. Alla somma di questi tre termini è applicata la funzione di attivazione f introducendo non linearità nel modello, le tre più utilizzate sono la sigmoide (1.1), la tangente iperbolica (1.2) e la Rectified Linear Unit (ReLU) utilizzata spesso nelle reti LSTM.

Le RNN sono caratterizzate da un'elevata capacità nel gestire dati sequenziali e serie temporali, ma nonostante questo presentano due limiti principali:

- Nel processo di training viene utilizzata una variante della *backpropagation through time* (BPTT) per minimizzare l'errore tra dati reali e dati predetti, in cui l'errore viene propagato all'indietro nel tempo in modo da modificare i pesi tenendo conto anche degli errori passati, può capitare che gli errori stimati diventino molto piccoli spostandosi all'indietro creando un aggiornamento non efficace dei pesi, questo problema può essere definito come vanishing gradient e porta complicazioni nel caso di lunghe sequenze temporali con dipendenze a lungo termine.

- Il contrario del vanishing gradient succede nel caso in cui gli errori diventano estremamente grandi, causando aggiornamenti troppo elevati dei pesi e portando all'instabilità del modello durante il training, questo fenomeno è detto exploding gradient.

Per oltrepassare questi limiti e riuscire ad elaborare dati con relazioni a lungo termine sono state create delle varianti:

- **Long Short-Term Memory (LSTM):** una versione avanzata delle RNN create per risolvere il problema del gradiente evanescente tramite un meccanismo di gating. Sono presenti tre "cancelli" principali (gate) grazie ai quali vengono regolati i flussi di informazioni: il forget gate stabilisce le informazioni dello stato precedente che non devono essere ricordate, l'input gate decide quali informazioni devono essere memorizzate nello strato nascosto ed infine l'output gate determina un output dallo strato nascosto attuale.

- **Gated Recurrent Unit (GRU):** sono una versione più semplice rispetto le LSTM ed utilizzano due gate: il reset gate stabilisce la quantità di informazioni dello strato nascosto precedente che deve essere dimenticata e l'update gate aggiorna lo strato nascosto attuale con nuovi dati. Anche le GRU come le LSTM sono in grado di elaborare dati con dipendenze a lungo termine, pur essendo più semplici e veloci da allenare rispetto alle LSTM.

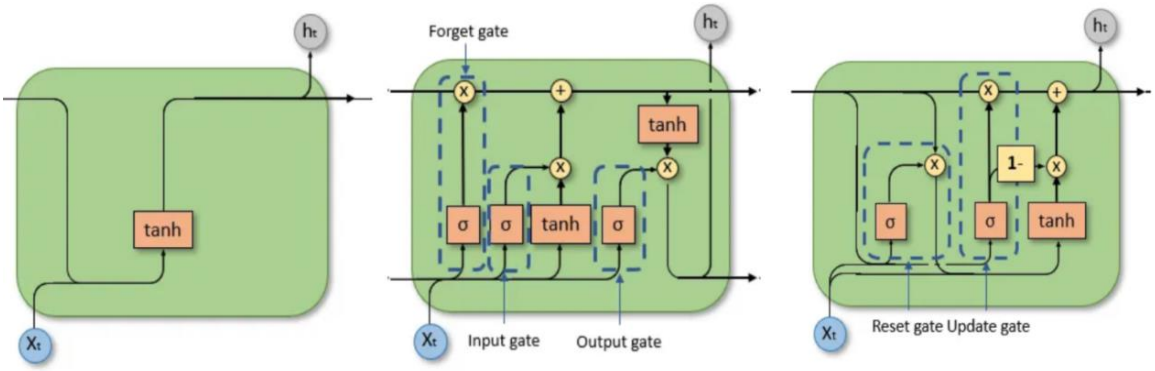


Figura 1.7: Confronto tra l'architettura di una rete RNN (a sinistra), di una rete LSTM (al centro) e di una GRU (a destra).

Per le motivazioni sopra elencate le RNN sono utilizzate in ambiti in cui compaiono dati sequenziali come il riconoscimento vocale per trascrivere file audio in testo, le serie temporali come la modellazione dei dati meteo o infine la generazione di musica ed arte basandosi su modelli già esistenti.

1.4 Long Short-Term Memory

Come anticipato in precedenza le reti LSTM sono una variante delle reti RNN, caratterizzate da una struttura più articolata e più complicate da allenare, ma in grado di oltrepassare il problema del vanishing gradient ed elaborare quindi in maniera più efficace lunghe sequenze di dati con dipendenze a lungo termine.

Le reti LSTM sono state introdotte da Sepp Hochreiter e Jurgen Schmidhuber nel 1997 e si sono fin da subito dimostrate utili nell'elaborazione di sequenze lunghe tramite una struttura interna che consente di decidere quali informazioni tenere e quali dimenticare.

A differenza di una rete RNN in cui è presente un solo stato nascosto, nella cella LSTM sono presenti due stati differenti che gestiscono le informazioni:

- $\boldsymbol{h_t}$ lo stato nascosto: può essere considerato come una memoria a breve termine
- $\boldsymbol{C_t}$ lo stato della cella: può invece essere considerato come una memoria a lungo termine

Questi due stati interagiscono tra loro attraverso tre porte:

1. Forget Gate o porta di dimenticanza: il suo compito è stabilire la quantità di informazioni contenute nello stato della cella C_{t-1} che vanno conservate o dimenticate, essendo espressa da una funzione sigmoide, la quale produce un valore nell'intervallo tra zero ed uno, nel caso in cui il valore si avvicini allo zero l'informazione sarà perduta, nel caso contrario in cui il valore è prossimo ad uno l'informazione sarà mantenuta. La forget gate è calcolata tramite la seguente formula:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1.4)$$

   Dove:
   - σ è la funzione sigmoide.
   - W_f è la matrice dei pesi legata alla forget gate.
   - $[h_{t-1}, x_t]$ è l'unione tra lo stato nascosto precedente h_{t-1} e l'input attuale x_t.
   - b_f è il bias della forget gate.

2. Input Gate o porta di input: il suo obiettivo è quello di decidere quali informazioni dell'input attuale devono essere aggiunte allo stato della cella, questo viene fatto in due passaggi: la funzione sigmoide i_t stabilisce quali valori aggiornare, successivamente la funzione tangente iperbolica *tanh* crea un vettore di nuovi valori $\tilde{C}_t$ che verranno aggiunti allo stato della cella.
   I nuovi valori da aggiornare determinati tramite la sigmoide sono espressi dalla seguente formula:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (1.5)$$

   Mentre il vettore di nuovi valori è espresso attraverso la seguente formula:

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (1.6)$$

   Dove:
   - W_i e W_c sono le matrici dei pesi rispettivamente della input gate e dei nuovi valori.
   - b_i e b_c sono i bias associati.
   - σ e tanh sono rispettivamente la funzione sigmoide e la funzione tangente iperbolica.

Determinate le nuove informazioni da aggiungere e quelle da mantenere, lo stato attuale della cella C_t viene calcolato tramite la combinazione dello stato precedente C_{t-1} ed i nuovi valori $\tilde{C}_t$, in formule lo stato attuale della cella può essere scritto come:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (1.7)$$

Osservando l'equazione si nota come la rete LSTM sia in grado di mantenere informazioni tramite la forget gate e aggiornandole attraverso l'input gate.

3. Output Gate o porta di output: stabilisce la parte dello stato della cella C_t che deve essere utilizzata per calcolare l'output della cella per lo stato attuale, la combinazione tra questa porta e lo stato della cella consente di produrre l'output.
   L'output gate è calcolato come:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (1.8)$$

Mentre l'output h_t è calcolato come:

$$h_t = o_t \cdot \tanh(C_t) \quad (1.9)$$

In cui il termine $\tanh(C_t)$ normalizza lo stato della cella in un range tra -1 e 1.

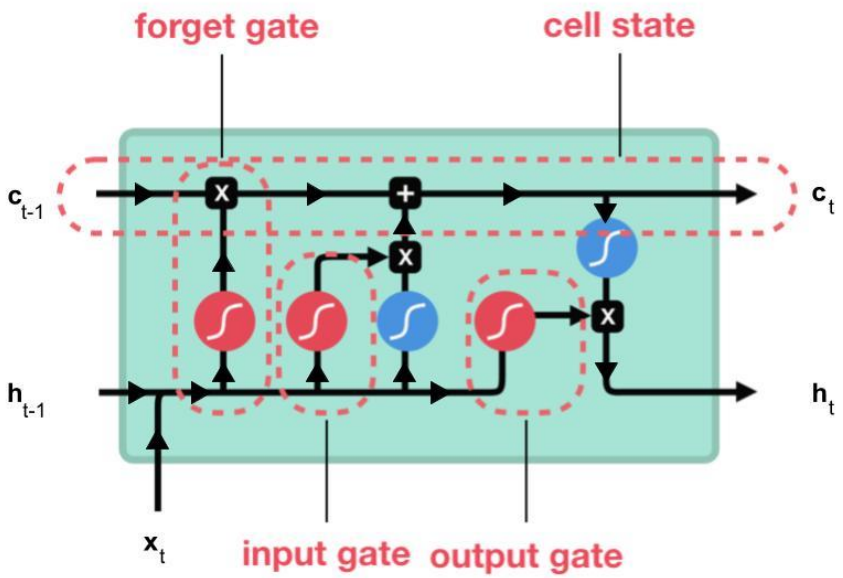


Figura 1.8: Architettura di una rete LSTM.

Riferendoci alla figura 1.8 possiamo avere un'idea più chiara sulla struttura di una cella LSTM, sul suo funzionamento e sulla divisione nelle tre porte principali. Ripercorrendo il flusso dei dati nella cella inizialmente si passa nella forget gate in cui si decide quali informazioni della cella precedente devono essere dimenticate, si passa poi dalla input gate in cui si scelgono le nuove informazioni che devono essere aggiunte alla cella, a questo punto si ha un aggiornamento dello stato della cella combinando le informazioni della input e della forget gate per aggiornare la memoria della cella, infine si giunge all'output gate che determina la parte della cella utilizzata per creare l'output.
Grazie alla struttura a blocchi formata dalle varie porte si è in grado di controllare il flusso delle informazioni in maniera precisa, uno svantaggio è però dato dal fatto che richiedono tempi di allenamento più lunghi e maggiori risorse computazionali.

Capitolo 2

Analisi dei dati

2.1 Datatrain e Datatest

Come anticipato nel capitolo precedente le reti neurali, in particolare quelle ricorrenti di tipo LSTM hanno bisogno di un'elevata quantità di dati per poter essere calibrate al meglio, questo set di dati è diviso in:

- Datatrain o *Train Set* : grazie a quest' insieme di dati la rete viene allenata, riesce a relazionare gli input agli output modificando ed ottimizzando i suoi pesi e bias in base all'errore calcolato dato dalla differenza tra la previsione del modello ed il valore reale.
- Datatest o *Test Set* : quest'insieme di dati viene invece utilizzato per determinare le prestazioni del modello e valutare le sue performance su dati che non ha mai visto, questo passaggio è fondamentale soprattutto nelle reti RNN che lavorando su lunghe sequenze di dati incorrono facilmente nel problema dell'overfitting, in cui la rete creata si adatta molto bene ai dati di training ma è caratterizzata da elevati errori su nuovi dati.

In software come Matlab sono presenti funzioni come "cvpartition" che partendo da un data set e stabilendo una percentuale restituisce un train set e un test set formati da una quantità di dati proporzionale alla percentuale inserita nella funzione, solitamente viene utilizzata una percentuale del 80%-85% per il training e del 20%-15% per il test.

Nel nostro caso specifico andremo invece ad utilizzare come train set un concatenamento di tre file registrati in laboratorio al banco prova su un motore termico V8 a combustione interna ed accensione comandata, l'insieme del train set è formato da:

- Lambda sweep: in un motore a combustione interna il parametro lambda (λ) è definito come il rapporto tra aria e combustibile (AFR), λ viene considerato un parametro molto importante in un motore ed infatti per questo ne parleremo più nel dettaglio a breve, mentre ora focalizziamoci sulla procedura del lambda sweep. E' una tecnica utilizzata per valutare il comportamento del motore al variare del parametro λ, il motore viene posizionato sul banco prova e collegato al freno a correnti parassite in modo da poter controllare i giri rpm del motore tramite il freno, viene stabilito un certo intervallo Δ con cui viene fatto variare λ e così facendo si nota la variazione delle prestazioni del motore come la potenza, la coppia, la temperatura dei gas di scarico, le emissioni di inquinanti e il consumo, questi dati vengono registrati e convertiti in un file Excel.

- Spark sweep: considerando un motore ad accensione comandata, con il termine spark advance o anticipo d'accensione si fa riferimento al momento in cui si crea la scintilla tra gli elettrodi della candela e di conseguenza si avvia il processo di combustione. Questo parametro viene misurato in gradi dell'albero motore rispetto al punto morto superiore (PMS), si parla di anticipo d'accensione nel caso in cui la scintilla si generi prima del PMS altrimenti di ritardo d'accensione o di anticipo d'accensione ma con valore negativo se la scintilla viene scoccata dopo il PMS. L'anticipo d'accensione è un parametro fondamentale perché ad esso sono relazionati molti aspetti come la trasformazione da energia chimica ad energia meccanica, il fenomeno della detonazione e la formazione di inquinanti, per questi motivi l'anticipo d'accensione è di fondamentale importanza e verrà approfondito più avanti mentre ora ci focalizziamo sulla procedura di spark sweep. È una tecnica utilizzata per valutare le prestazioni e gli effetti del motore al variare dell'anticipo d'accensione, come nel caso precedente si collega il motore al freno e si fa variare di un certo intervallo l'anticipo, cercando di mantenere costanti le altre variabili, si ripetono i cicli a diversi regimi di condizione e si registrano i dati, convertendoli poi in file Excel.

- VVT sweep: con il termine variable valve timing (VVT) ci si riferisce alla possibilità di poter variare la fasatura delle valvole di aspirazione e di mandata in relazione alle condizioni operative del motore, così facendo si può ottimizzare l'aspirazione di aria nel cilindro e l'espulsione dei gas di scarico per poter migliorare le prestazioni e diminuire i consumi e le emissioni. Nei motori tradizionali la fasatura delle valvole è fissa e non può essere variata , tramite il processo di VVT sweep si studia il comportamento del motore al variare della fasatura delle valvole, come nelle tecniche precedenti si posiziona il motore al banco prova e si collega al freno, si fa quindi variare la fasatura delle valvole agendo sul ritardo o sull'anticipo nella loro chiusura ed apertura, ma anche sulla durata di apertura, così facendo si registrano i dati ottenuti tramite i sensori presenti nel motore come la sonda lambda che rileva la presenza di ossigeno nei gas di scarico oppure le temperature dei gas di scarico. Potendo variare la fasatura delle valvole si può arrivare al caso limite in cui entrambe le valvole (aspirazione e scarico) siano aperte, questa condizione è detta di incrocio e favorisce il ricircolo interno dei gas di scarico e riduce le emissioni di NO_x.

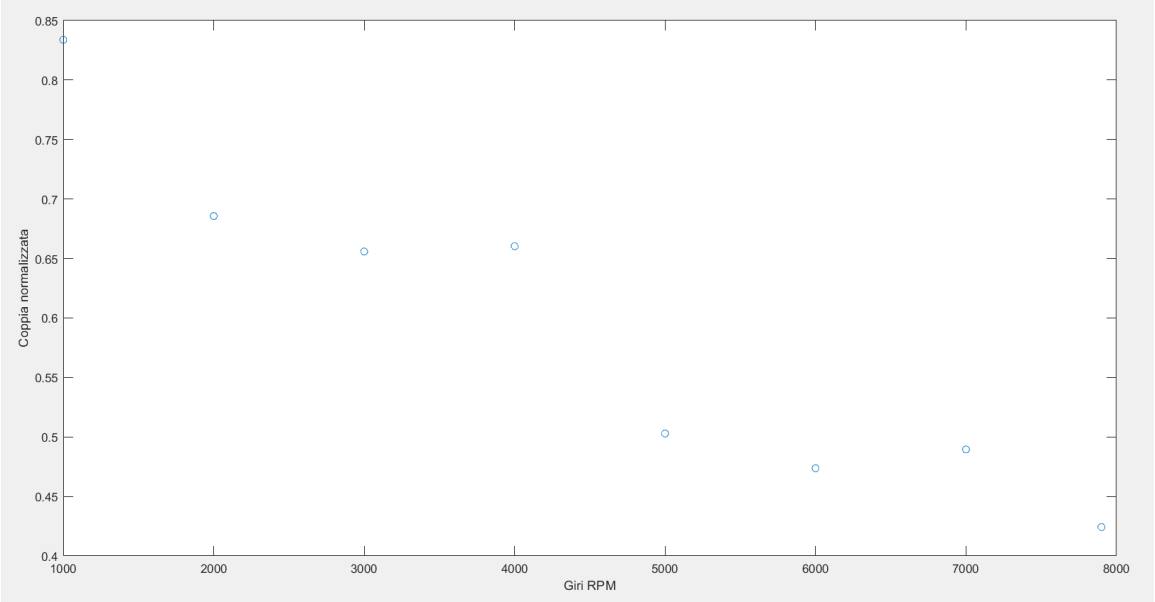
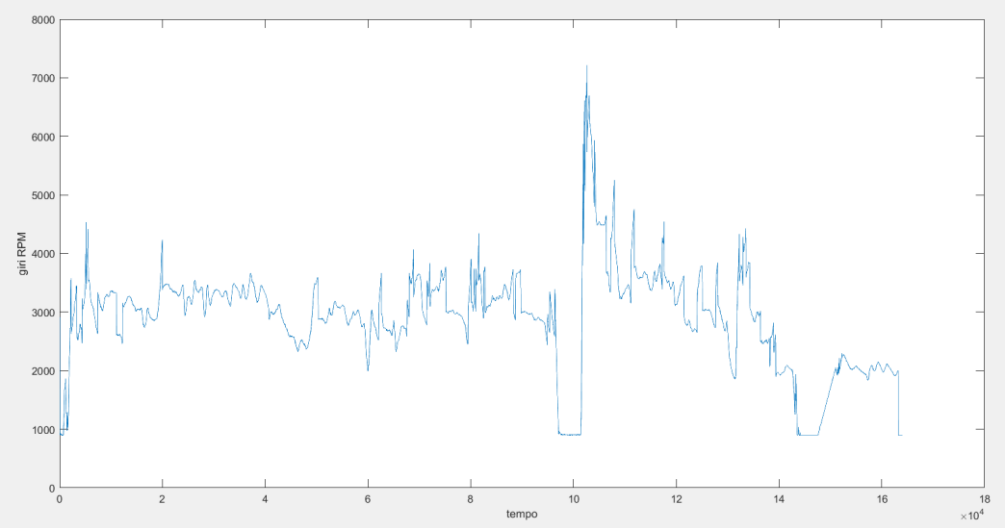


Figura 2.1: Coppia media normalizzata in funzione dei giri per i dati del file Spark sweep.

Il test set è invece formato da due file, in questo caso si tratta di prove dinamiche essendo registrate una su strada ed una su pista, andremo quindi a denominare rispettivamente questi cicli di guida di test “ Road test “ e “ Track test “.



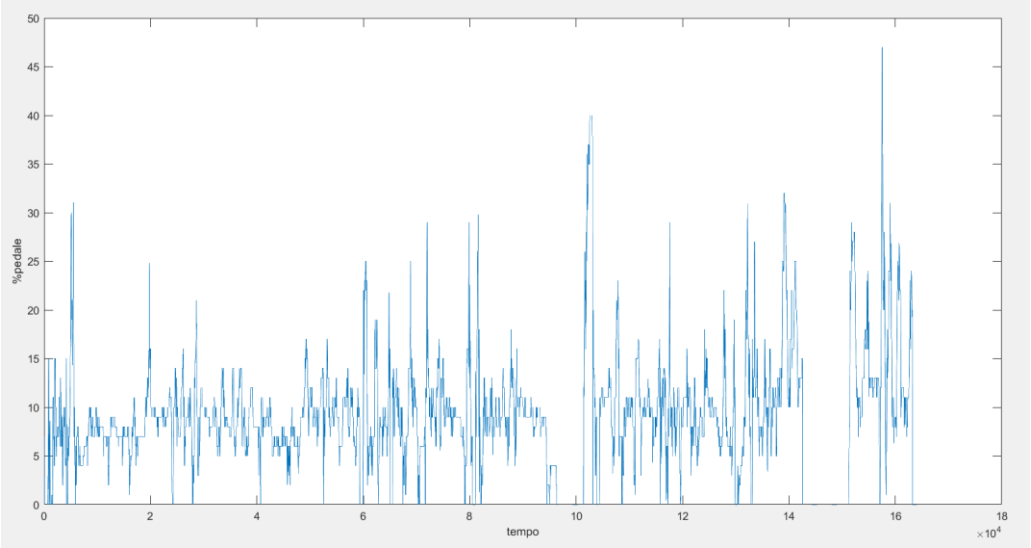


Figura 2.2: Grafici raffiguranti i giri e la % di pedale in funzione del tempo per il Road Test.

I file Excel utilizzati come data set contengono centinaia di parametri, alcuni valori sono legati alle condizioni della sala prove in cui vengono registrati i dati, mentre altri sono legati al comportamento del motore e sono quelli che interessano a noi; come visto in precedenza al modello vanno passati dati in input per potersi allenare e creare output che vanno poi confrontati con quelli reali del test set. Nel nostro modello utilizzeremo quattro input in grado di poter caratterizzare al massimo le condizioni di funzionamento del motore e sono: i giri (RPM), il lambda (lamsoni), l'anticipo d'accensione (zwist) ed il carico (rlsol_w). I termini tra parentesi sopra riportati sono rappresentativi del nome usato per rappresentare i parametri nel file Excel. L'output è invece dato dal MFB50, di cui verrà fatta una media tra quello del cilindro 5,6 e 7 studiando una singola bancata, viene tralasciato quello del cilindro 8 in quanto ha comportamenti differenti dagli altri tre. La rete si calibrerà quindi tramite il train set e verrà poi testata attraverso il test set dove verranno predetti i valori di MFB50 e calcolato l'errore tra il valore reale e quello predetto dal modello. Andiamo ora ad analizzare nello specifico i vari parametri di input e output.

2.2 MFB50

Il parametro MFB50 (50% of mass fraction burned) rappresenta l'angolo in gradi in cui il 50% della massa di miscela formata da aria e combustibile è stata bruciata rilasciando l'energia chimica che possedeva.

Per poter calcolare il parametro MFB50 è necessario introdurre il rilascio di calore che rappresenta l'energia termica liberata durante il processo di combustione, ci si riferisce poi al rilascio di calore cumulato normalizzato il quale è dato dall'integrale del rilascio di calore istantaneo rispetto al calore totale disponibile, questo valore rappresenta la frazione di massa bruciata. È importante introdurre il sistema indicating, che attraverso la misurazione della pressione all'interno del cilindro, e la sua relazione con l'angolo di manovella, è in grado di fornire informazioni sul ciclo di combustione, la pressione è infatti un parametro fondamentale nel calcolo del rilascio di calore.

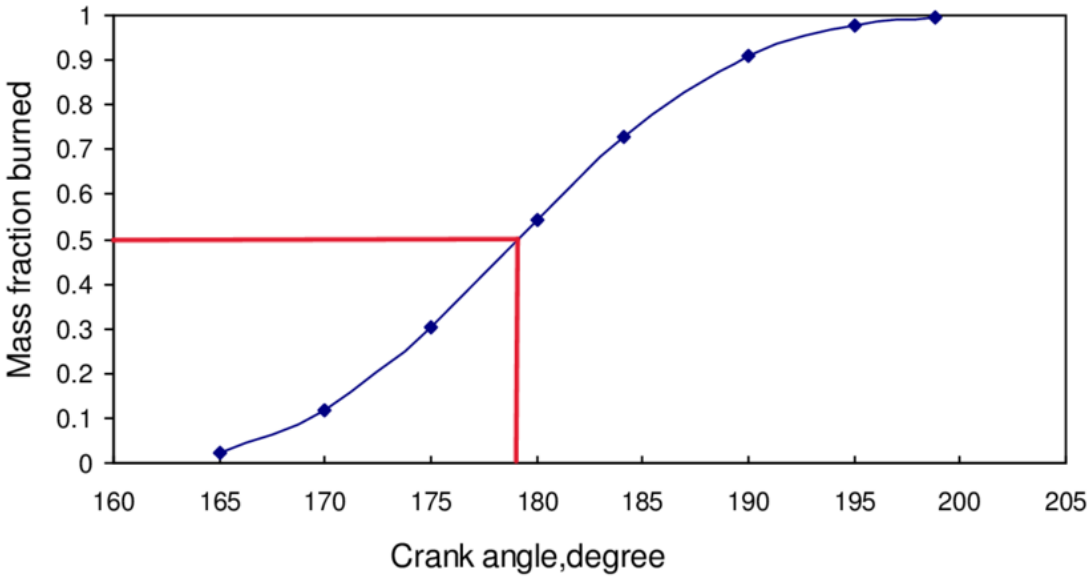


Figura 2.3: Relazione tra la massa di miscela bruciata e l'angolo di manovella.

Solitamente i valori ottimali di MFB50 sono quelli compresi in un intervallo tra gli 8 e i 12 gradi dopo il punto morto superiore (PMS), ovvero il punto in cui il pistone è più vicino alla testata, un MFB50 compreso tra questi valori consente di sfruttare al massimo l'energia liberata dalla combustione e trasformarla in lavoro meccanico nella fase di espansione. Nel caso in cui il valore del MFB50 fosse troppo anticipato si otterrebbe una pressione nel cilindro elevata creando perdite meccaniche ed il fenomeno della detonazione, ovvero un processo in cui una parte degli end gas cioè la zona della carica fresca non ancora investita dal fronte di fiamma si trova in condizioni di autoaccensione a causa delle elevate temperature e pressioni, incentivando la propagazione di onde di pressione. Nel caso contrario in cui il valore di MFB50 sia troppo ritardato, la combustione avverrebbe in una posizione molto distante dal PMS riducendo drasticamente la potenza utile generata.

Il calcolo del MFB50 è basato sulla pressione all'interno del cilindro e dall'espressione del calore rilasciato durante la combustione che può essere ottenuta combinando il primo principio della termodinamica e l'equazione di stato dei gas ideali, il calore rilasciato durante la combustione può essere espresso come:

$$dQ = dU + pdV \quad (2.1)$$

in cui:

- dQ rappresenta il calore rilasciato.
- dU rappresenta la variazione di energia interna.
- P rappresenta la pressione.
- dV rappresenta la variazione di volume del cilindro.

Determinata l'equazione in grado di rappresentare il calore rilasciato, calcoliamo il calore cumulativo che può essere visto come la somma di tutti i contributi di calore da $\theta_{inizale}$ in cui ha inizio la combustione fino ad un determinato angolo θ, possiamo quindi scrivere:

$$Q_{cum}(\theta) = \int_{\theta_{inizale}}^{\theta} dQ \quad (2.2)$$

La frazione di massa di miscela bruciata all'angolo θ è espressa dal rapporto tra il calore rilasciato fino a quel punto e il calore totale rilasciato:

$$MFB(\theta) = \frac{Q_{cum}(\theta)}{Q_{tot}} \quad (2.3)$$

Dove:

- $MFB(\theta)$ è la frazione di massa di miscela bruciata all'angolo θ.
- $Q_{cum}(\theta)$ è il calore rilasciato fino all'angolo θ.
- Q_{tot} è il calore totale rilasciato.

Calcolata quindi la funzione in grado di esprimere la frazione di massa di miscela bruciata, il parametro MFB50 è il punto in cui il 50% della massa è stata bruciata, cioè:

$$MFB(\theta_{50}) = 0.50 \quad (2.4)$$

In cui θ_{50} è l'angolo per il quale è stata bruciata il 50% della massa di miscela, solitamente si ha una rappresentazione grafica che mostra la relazione tra frazione di massa bruciata e angolo di manovella per il calcolo del MFB50, come quella in figura 2.1.

Sono presenti diversi parametri in grado di influenzare il MFB50, l'anticipo d'accensione o spark advance (SA) ha un effetto diretto sul valore di MFB50, nel caso in cui il valore d'anticipo fosse troppo elevato anche il valore di MFB50 avverrà prima rispetto alla giusta posizione, nel caso contrario in cui l'accensione sia troppo ritardata, il MFB50 avverrà in ritardo rispetto alla posizione ottimale causando uno spreco di energia utile.

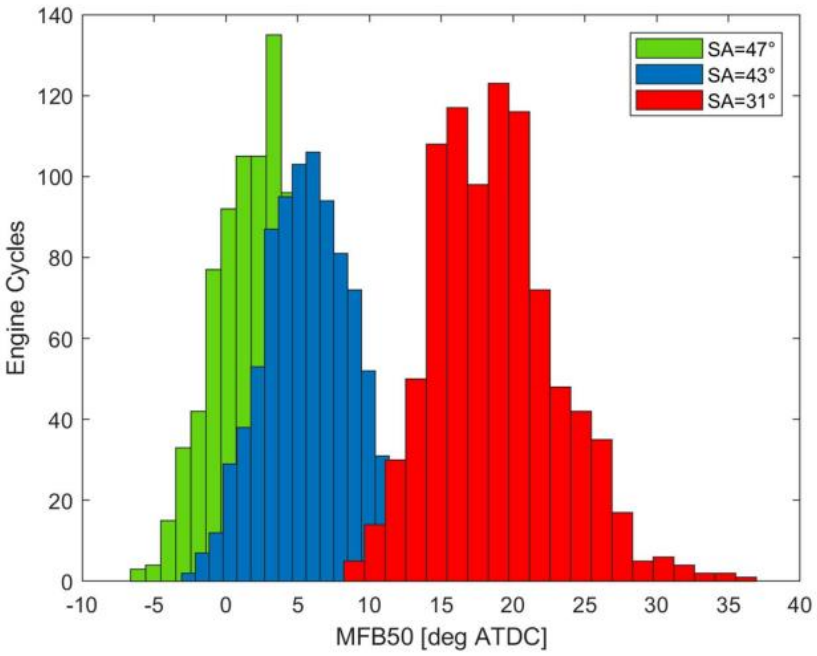


Figura 2.4: Relazione tra MFB50 e anticipo d'accensione (SA).

Sono presenti altri parametri che possono influenzare il valore di MFB50 come:

- Condizioni di carico e giri motore: Il MFB50 varia in base alle condizioni del motore, nel caso di carichi elevati e giri motore alti, il tempo in cui può avvenire la combustione è ridotto quindi l'anticipo d'accensione deve essere elevato in modo da avere un giusto MFB50.

- Geometria del motore: Parametrici geometrici come il rapporto di compressione, l'alesaggio, il rapporto corsa diametro e la forma della camera di combustione possono influenzare il flusso dell'aria e quindi la combustione, creando effetti sul valore di MFB50.

- Tipologia di carburante: I carburanti per motori ad accensione comandata sono caratterizzati da un numero di Ottano per determinare la loro resistenza alla detonazione, nel caso in cui la resistenza alla detonazione sia scarsa si ha una combustione rapida portando ad un anticipo del MFB50.

2.3 Anticipo d’accensione (SA)

L’anticipo d’accensione o spark advance (SA) esprime il numero in gradi, in cui viene fatta scoccare la scintilla prima del punto morto superiore (PMS), è necessario che la scintilla avvenga prima del PMS in quanto la combustione non è un processo istantaneo, ma è presente un intervallo di tempo in cui il fronte di fiamma si espande e si completa , grazie a questo anticipo d’accensione si riesce ad ottenere la massima pressione in camera di combustione nel momento ottimale, ovvero poco dopo il PMS, avendo un’ elevata efficienza di conversione dell’energia e diminuendo le perdite. E’ necessario regolare in maniera calibrata l’anticipo in quanto se troppo elevato porta ad un’elevata pressione in camera di combustione provocando stress sui componenti e detonazione, con la propagazione di onde di pressione; nel caso contrario in cui l’anticipo sia troppo ritardato, la combustione si completa nel momento in cui il pistone si è allontanato di molto dal PMS, diminuendo la pressione e la corsa disponibile per generare potenza utile causando un calo di efficienza e di prestazioni.

Sono presenti diverse variabili in grado di influenzare l’anticipo d’accensione, come ad esempio:

- Rapporto di compressione: In caso di motori con elevati rapporti di compressione è necessario avere anticipi d’accensione precisi, in quanto a causa delle maggiori pressioni in camera la miscela si comprime di più e la combustione è più rapida.

- Giri del motore (RPM): A giri elevati l’anticipo deve essere maggiore perché il tempo a disposizione per completare la combustione è minore.

- Qualità del carburante: Carburanti con elevata resistenza alla detonazione cioè con elevati numeri di Ottano consentono anticipi più elevati in quanto meno soggetti al fenomeno della detonazione.

- Rapporto aria-combustibile: Miscele magre ovvero formate da più aria e meno combustibile in base al rapporto stechiometrico bruciano più lentamente in confronto a quelle ricche e hanno bisogno quindi di un anticipo maggiore.

Il grafico riportato sotto rappresenta l’andamento della pressione nel cilindro nel caso di combustione (firing) e senza combustione (motoring), si può notare la variazione di pendenza della pressione a circa -30 ° ovvero il punto in cui inizia la combustione.

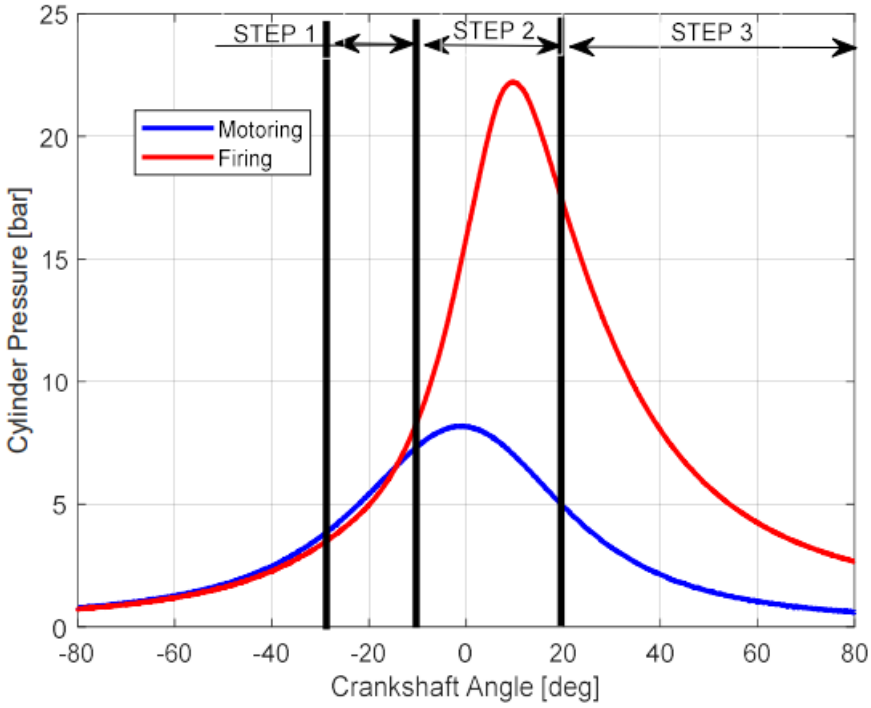


Figura 2.5: Andamento della pressione in funzione dell'angolo di manovella.

2.4 Lambda (λ)

L'insieme di centinaia di reazioni intermedie per passare dai reagenti ai prodotti durante il processo di combustione può essere semplificato tramite la seguente formula:

$$C_{43}H_{84} + 64O_2 \rightarrow 43CO_2 + 42H_2O \quad (2.5)$$

Il rapporto tra massa di ossigeno e massa di combustibile è dato da:

$$m_{fuel} = 43 \cdot 12 + 84 = 600 \quad (2.6)$$

$$m_{o2} = 64 \cdot 32 = 2048 \quad (2.7)$$

$$\frac{m_{o2}}{m_{fuel}} = \frac{2048}{600} \quad (2.8)$$

Considerando che in ogni kg di aria sono contenuti 0.23 kg di ossigeno, il rapporto aria-combustibile o air fuel ratio (AFR) in condizioni stechiometriche è dato da:

$$(AFR)_{st} = \frac{m_{air}}{m_{fuel}} = \frac{m_{air}}{m_{o2}} \cdot \frac{m_{o2}}{m_{fuel}} = \frac{1}{0.23} \cdot \frac{2048}{600} = 14.84 \quad (2.9)$$

Il rapporto aria-combustibile stechiometrico varia in base all'idrocarburo, per la benzina solitamente è vicino a 14.65, mentre per il gasolio è 14.5, a causa di questa variabilità si preferisce fare riferimento ad un indice normalizzato rispetto al rapporto stechiometrico, l'indice d'aria anche detto lambda (**λ**), definito come:

$$\lambda = \frac{AFR}{AFR_{st}} \quad (2.10)$$

Il parametro lambda misura quindi la deviazione dal rapporto stechiometrico e può assumere tre valori:

- $\lambda = 1$: Ci troviamo in condizioni stechiometriche, in questo caso la combustione è efficiente e vengono minimizzate le emissioni di inquinanti.

- $\lambda < 1$: Ci troviamo nel caso di una miscela grassa, ovvero in eccesso di combustibile rispetto all'aria, in questa situazione avremo un elevato consumo di carburante in quanto non tutto il combustibile riuscirà a partecipare al processo di combustione, questa configurazione viene spesso utilizzata per diminuire le temperature dei gas di scarico in quanto il carburante in eccesso evaporando assorbe calore.

- $\lambda > 1$: Ci troviamo nel caso di una miscela magra in cui è presente più aria rispetto al carburante, questa configurazione è usata a bassi regimi di rotazione per poter diminuire i consumi avendo una quasi completa combustione del carburante.

Il parametro lambda viene costantemente regolato in base alle informazioni che arrivano dalla sonda lambda, un sensore che misura la quantità di ossigeno nei gas di scarico ed invia un segnale alla centralina (ECU); se il sensore rileva ossigeno significa che la miscela è magra e viene quindi iniettato più carburante, nel caso contrario in cui viene rilevato poco ossigeno nei gas di scarico viene ridotto l'apporto di carburante.

È fondamentale tenere sotto controllo il lambda in quanto è strettamente legato alla produzione di emissioni inquinanti come il monossido di carbonio (CO), gli ossidi d'azoto (NO_x) e gli idrocarburi incombusti (HC). Si cerca di mantenere un lambda vicino alla stechiometria in quanto i catalizzatori hanno un'elevata efficienza di conversione in quel range.

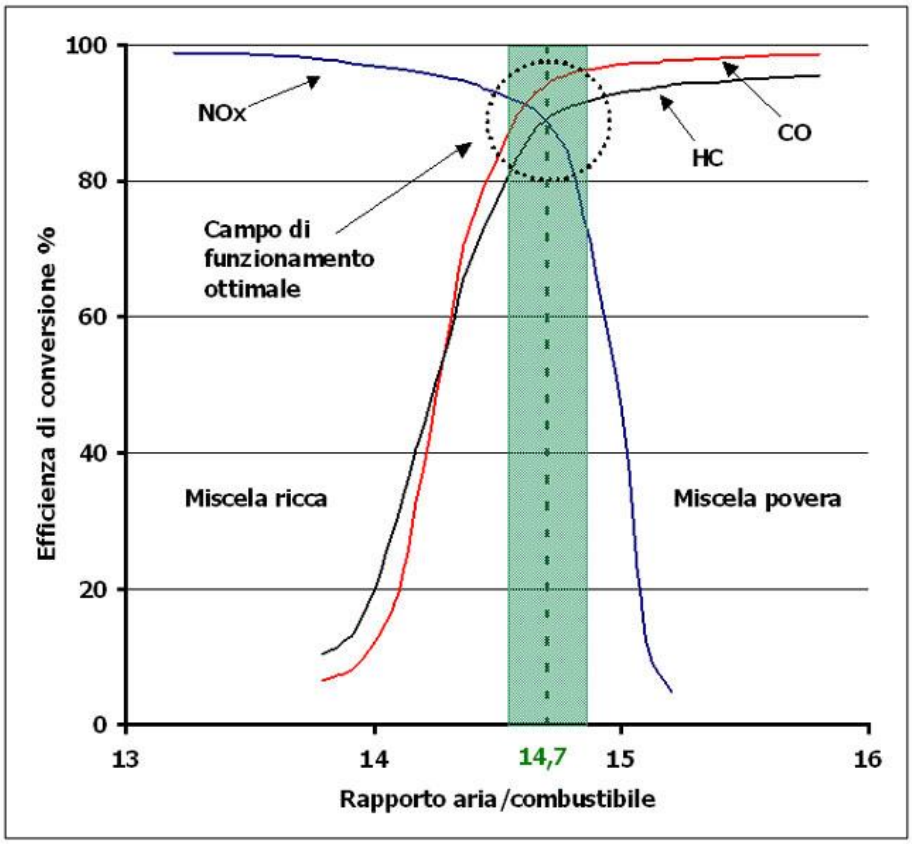


Figura 2.6: Efficienza di conversione delle emissioni inquinanti in funzione di lambda.

2.5 Giri motore (RPM)

I giri motore o RPM sono un parametro fondamentale per valutare le condizioni di funzionamento del motore ed indicano il numero di volte che l'albero motore compie una rotazione completa in un minuto.
Gli RPM sono strettamente legati alla coppia e alla potenza, sono presenti numerosi fattori in grado di influenzarli come ad esempio la differenza tra motori ad accensione per compressione e motori ad accensione comandata, in questi ultimi si raggiungono RPM più elevati grazie a componenti più leggeri in camera di combustione, nei motori ad accensione per compressione si hanno pressioni maggiori dovute a rapporti di compressioni più elevati e questo comporta la necessità di componenti più massicci dovendo quindi diminuire la velocità di rotazione.

2.6 Coefficiente di carico volumetrico

L'ultimo parametro utilizzato come input è definito come relative load e può essere considerato come un'efficienza volumetrica, questo valore di input esprime il rapporto tra massa d'aria effettivamente aspirata e massa d'aria teoricamente aspirabile nelle condizioni ideali. Questo coefficiente è strettamente legato alle condizioni ambientali, in particolare alla pressione ambiente e alla temperatura perché sono in grado di influenzare la densità dell'aria, un altro fattore correlato a questo coefficiente è dato dai moti di carica, i quali sono proporzionali all'efficienza volumetrica del motore, cioè la capacità di riempire i cilindri con carica fresca ad ogni ciclo d'aspirazione.

Capitolo 3

Implementazione del modello in Matlab

3.1 Acquisizione dei dati

Il primo passo per la creazione del modello è l'acquisizione dei dati in Matlab dai file Excel registrati in laboratorio e dai cicli di guida, per poter scegliere i file presenti nel computer da esportare su Matlab viene usata la funzione " **uigetfile** " che apre una finestra di dialogo in cui sono elencati i file che potremo andare a selezionare in base ai filtri utilizzati.

Successivamente tramite la funzione " **xlsread** " è possibile convertire i file Excel prima selezionati in matrici.

A questo punto andremo a creare una matrice con i dati che ci interessano per la creazione del modello, tralasciando quelli superflui. Otteniamo quindi una matrice formata da cinque colonne, che in ordine rappresentano rispettivamente: i giri motore (rpm), il coefficiente di carico volumetrico, l'anticipo d'accensione, il lambda ed infine l'MFB50 di cui viene fatta una media tra il cilindro 5, 6 e 7.

Il procedimento sopra riportato viene effettuato per i tre file di training ovvero " lambda sweep ", " spark sweep " e " vvt sweep ", le tre matrici ottenute vengono quindi concatenate in modo da formare il datatrain e poter allenare il modello, lo stesso procedimento viene usato per i due file dinamici di test e formare il datatest su cui verrà testato il modello.

```
% acquisizione dati dal file lambda sweep%
[fName_banco,folder_banco]=uigetfile('*.xlsx','Multiselect','on','Select bench files');
raw=xlsread([folder_banco,fName_banco]);
data=raw(37:end,:);
rpm_1=data(:,69);
rlsol_w_1=data(:,124);
zwist_1=data(:,157);
lamsoni_1=data(:,99);
mfb2_cyl5_1=data(:,194);
mfb2_cyl6_1=data(:,195);
mfb2_cyl7_1=data(:,196);
for i=1:92
   media_mfb2_1(i,:)=(mfb2_cyl5_1(i)+mfb2_cyl6_1(i)+mfb2_cyl7_1(i))/3;
end
```

Figura 3.1: Codice per acquisizione dati dal file "lambda sweep" e creazione della matrice.

3.2 Creazione della rete e scelta del numero di HiddenSizes

Tramite la funzione " **layrecnet** " è possibile creare una rete neurale ricorrente a livelli di tipo LSTM, la rete creata è modellabile in base a tre parametri :

- LayerDelays: Il vettore di riga dei ritardi.
- HiddenSizes: Il vettore di riga del livello nascosto.
- TrainFcn: La funzione di addestramento in retropropagazione.

Il parametro HiddenSizes rappresenta il numero di dimensioni dei livelli nascosti ed è modificabile in modo da diminuire l'errore tra i valori predetti e quelli reali.

Utilizzando un ciclo for per determinare l'andamento dell'errore RMSE sui dati di train al variare del numero di HiddenSizes si è potuto creare un vettore " errore_rmse_train_Hiddensize " contenente il valore dell'errore in funzione del numero di livelli nascosti.

```
%Ciclo for per determinare la variazione di RMSE training al variare di HiddenSizes%
for b=1:40
net=layrecnet(1:2,b);
net= train(net,dataTrain1_transp,media_mfb2_trans);
mfb50pred=net(dataTrain1_transp);
ET=rmse(mfb50pred,media_mfb2_trans);
errore_rmse_train_Hiddenzsize(b)=ET;
end
```

Figura 3.2: Codice del ciclo for per la creazione del vettore " errore_rmse_train_Hiddensize " contenente il valore dell'errore in funzione del numero di livelli nascosti.

Riportando in un grafico l'errore RMSE sui dati di train in funzione del numero di livelli nascosti e utilizzando la funzione " **fit** " per approssimare l'andamento della curva si giunge al compromesso di utilizzare 5 come numero di livelli nascosti.

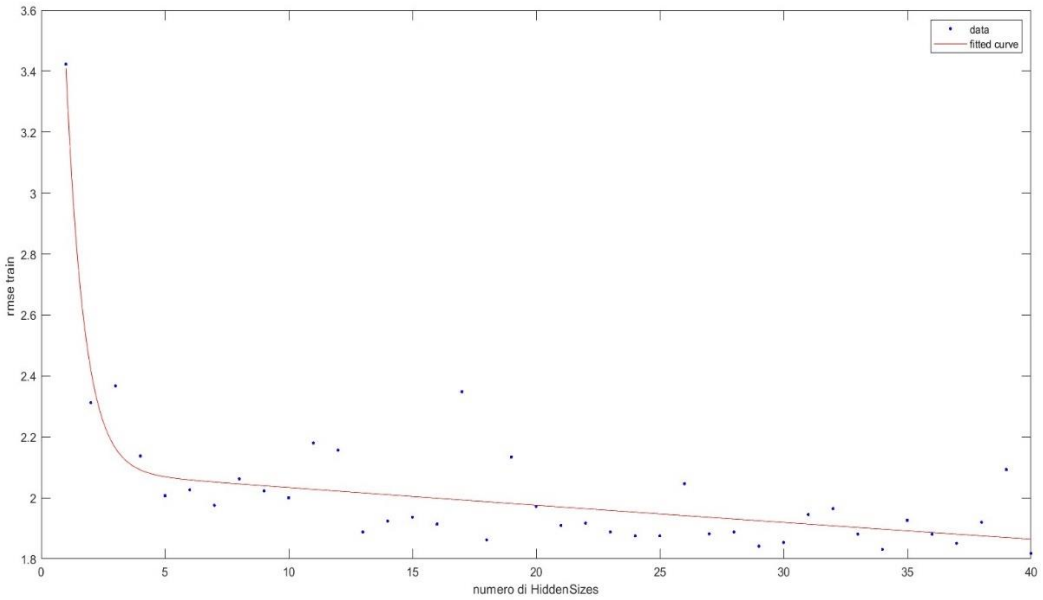


Figura 3.3: Grafico che riporta l'errore RMSE sui dati di train in funzione del numero di HiddenSizes.

Utilizzando 5 come numero di livelli nascosti si ha un giusto compromesso in quanto si può notare dal grafico 3.3 che l'andamento dell'errore RMSE è decrescente ma in maniera molto debole oltre il valore 5 di HiddenSizes, allo stesso tempo usando 5 come valore di HiddenSizes si riesce ad oltrepassare il problema dell'overfitting e dell'underfitting di cui avevamo fatto un accenno nell'introduzione. L'overfitting è caratterizzato da un'elevata capacità del modello ad elaborare i dati di training portando però ad elevati errori sui dati di test e nel nostro caso può essere causato da un eccessivo numero di livelli nascosti, al contrario l'underfitting è causato da un numero troppo basso di livelli nascosti creando così un modello troppo semplice e caratterizzato da scarse prestazioni.

Usando la funzione " **layrecnet** " creiamo quindi la nostra rete specificando 5 come numero di HiddenSizes ed usiamo poi la funzione " **train** " per allenare la rete tramite il datatrain.

```
%Creazione della rete ed addestramento %
net=layrecnet(1:2,5);
net=train(net,dataTrain1_transp,media_mfb2_trans);
```

Figura 3.4: Codice per la creazione della rete e per l'addestramento.

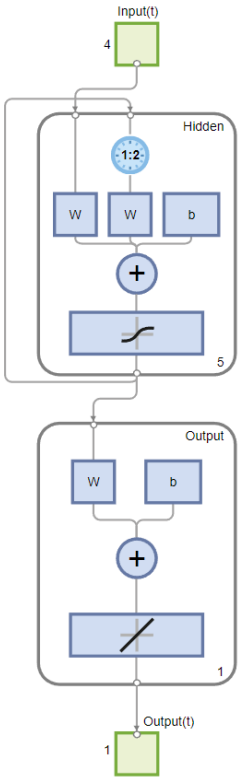


Figura 3.5: Schema rappresentativo della rete creata.

È utile rappresentare graficamente la relazione tra i valori di MFB50 realmente misurato e i valori di MFB50 predetto per i dati di train, più i valori di MFB50 (ovvero i punti blu nella figura 3.6) si avvicinano alla retta rossa, minore sarà l'errore del modello e quindi caratterizzato da migliori performance.

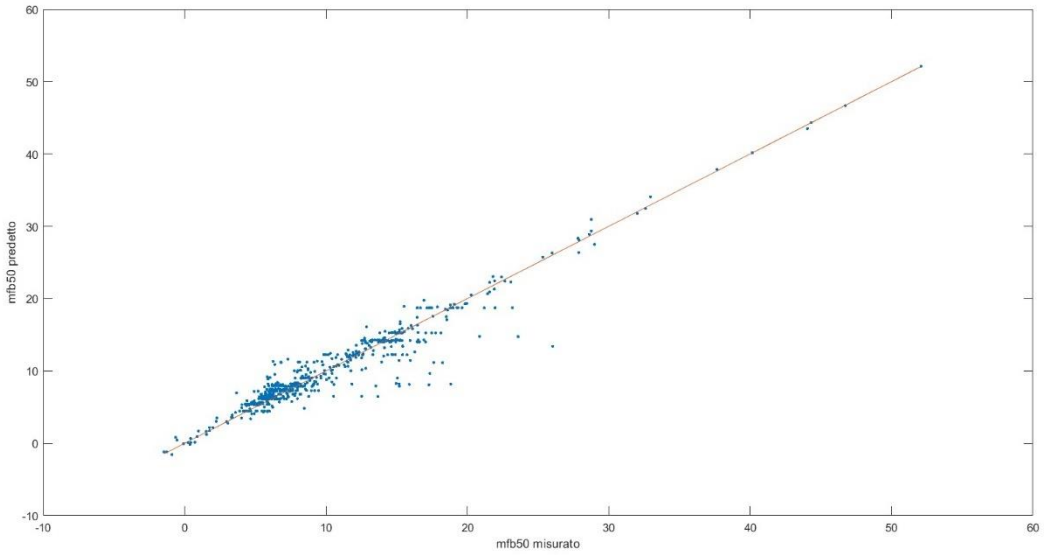


Figura 3.6: Grafico raffigurante la relazione tra MFB50 misurato e predetto sui dati di train.

Capitolo 4

Risultati

4.1 Errore RMSE

Prima di analizzare i risultati del modello sui due cicli di guida è necessario introdurre l'errore RMSE o Root Mean Squared Error, viene infatti considerato il parametro fondamentale per stabilire le performance di un modello e le sue capacità di predizione. La formula per calcolare l'errore RMSE tra due vettori y_{reale} ed y_{pred} è la seguente:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |y_{reale,i} - y_{pred,i}|^2} \quad (4.1)$$

Dove:

- y_{reale} : Rappresenta il vettore dei valori reali o misurati.
- y_{pred} : Rappresenta il vettore dei valori predetti o stimati.
- n : Rappresenta il numero totale di osservazioni.

L'errore RMSE fornisce una stima di quanto i valori predetti differiscono da quelli realmente misurati e dalla formula 4.1 si nota che vengono penalizzati maggiormente gli errori più grandi a causa del quadrato delle differenze tra valori reali e valori predetti. Ovviamente valori più bassi di RMSE indicano prestazioni più elevate del modello.

In Matlab è presente la funzione " **rmse** " che prende in input il vettore dei valori predetti e il vettore dei valori reali, che ovviamente devono avere la stessa lunghezza e restituisce l'errore RMSE.

Utilizzando 5 come numero di HiddenSizes si ottiene un errore RMSE di 1.8 sui dati di train, un valore che comunque si può ritenere accettabile in quanto vengono considerati validi errori fino a 4-5 gradi per i valori di MFB50, l'errore sui dati di train sarà però ovviamente minore rispetto a quelli di test su cui andremo a validare le performance del nostro modello.

4.2 Road Test

Dopo aver allenato il modello sui dati di train è necessario testarlo sui dati di test, nel nostro caso il Datatest è formato da due cicli di guida dinamici che abbiamo chiamato rispettivamente “ Road test “ e “ Track test”. Per ottenere il vettore di MFB50 predetti (che nel codice è denominato “ pred6 “) è sufficiente inserire i dati di input del Road test all’interno del modello allenato precedentemente denominato “ net “.

Infine utilizzando la funzione “ **rmse** “, di cui abbiamo parlato nel paragrafo precedente, è possibile determinare l’errore tra i valori predetti dal modello ed i valori reali di MFB50 misurati nel Road test. L’errore risultante su questo ciclo di guida è 12, un valore abbastanza elevato rispetto ai 4/5 gradi accettabili, il modello creato non si può quindi ritenere valido su questo ciclo di guida a causa dell’errore troppo elevato che però è causato dai picchi che si possono notare in figura 4.2, questi picchi non possono essere predetti in quanto sono generati dal sistema indicating quando ci sono delle mancate combustioni, e di conseguenza il valore di MFB50 non può essere calcolato, in questo caso specifico i picchi sono generati dalla centralina, che nei bruschi rilasci di pedale attua delle mancate iniezioni per rallentare più rapidamente il regime motore.

```
pred6=net(dataTest2.');
E6=rmse(pred6.',media_mfb2_5);
```

Figura 4.1: Codice per determinare l’errore sul Road test.

È utile riportare in un grafico come variano i valori di MFB50 reale ed MFB50 predetto in funzione del tempo, in particolare in rosso è riportato il valore reale mentre in verde quello predetto.

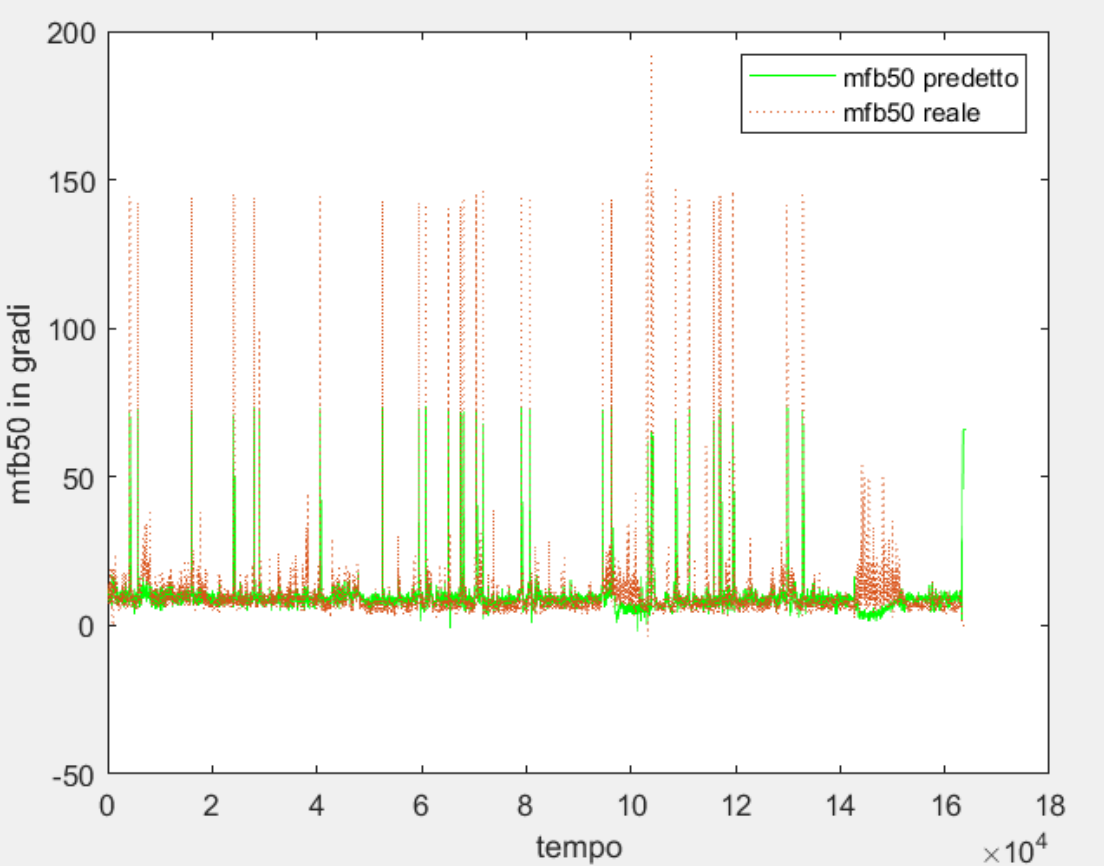


Figura 4.2: Relazione grafica tra MFB50 reale e predetto nel Road test.

4.3 Track Test

Per poter testare il modello sul secondo ciclo di guida, ovvero il Track test, è sufficiente ripetere i passaggi eseguiti precedentemente ma utilizzare come dati di input quelli del Track test, inserendo questi dati all'interno del modello si ottiene il vettore degli MFB50 predetti ed infine combinando questo vettore con quello degli MFB50 reali all'interno della funzione " **rmse** " si ottiene l'errore.

```
pred5=net(dataTest1.');
E5=rmse(pred5.',media_mfb2_3);
```

Figura 4.3: Codice per determinare l'errore sul Track test.

Su questo ciclo di guida si ottiene un errore di 4.6, un valore che comunque si può ritenere accettabile e quindi possiamo considerare affidabile il nostro modello su questo ciclo di guida.

Come nel caso precedente è utile riportare in un grafico l'andamento tra i valori di MFB50 reale e predetto in funzione del tempo, in rosso sono rappresentati i valori di MFB50 reale ed in verde quelli predetti.

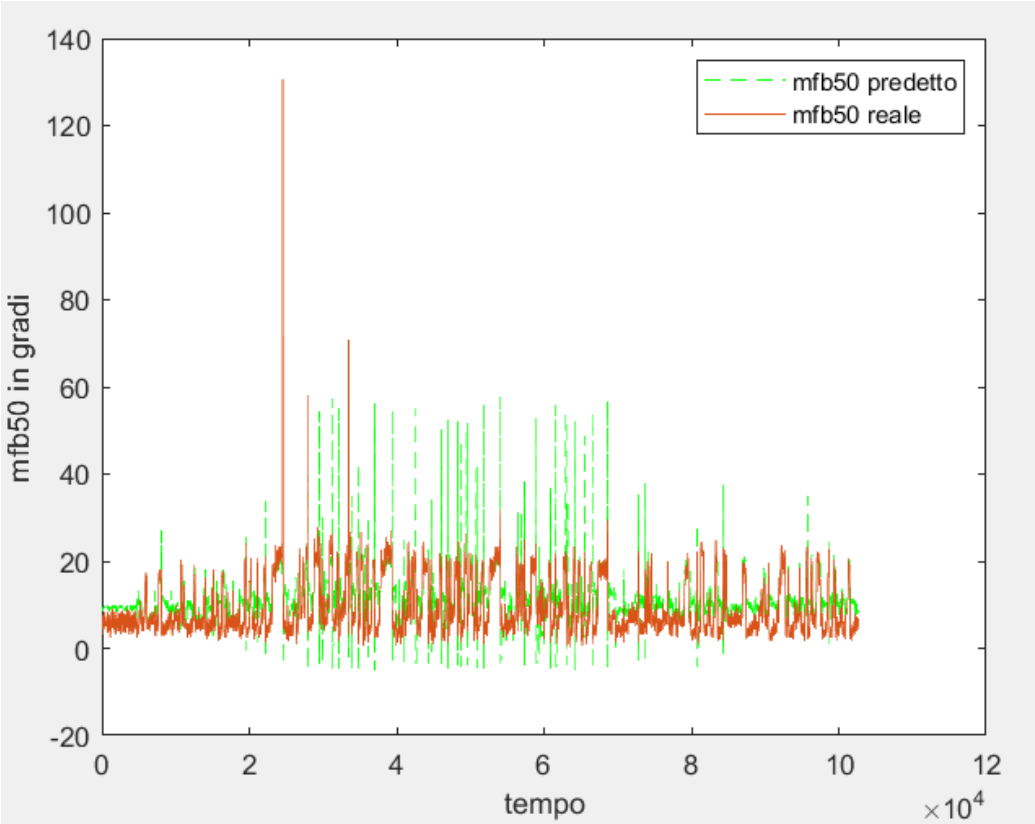


Figura 4.4: Relazione grafica tra MFB50 reale e predetto nel Track test.

4.4 Conclusioni

Tramite il machine learning ed il software Matlab si è potuto creare un modello in grado di predire i valori di MFB50 in relazione a quattro input in grado di caratterizzare le condizioni di funzionamento del motore, il modello creato restituisce un errore di 12 e 4.6 sui due cicli di guida su cui viene testato, l'elevata differenza di errore tra i due cicli di guida è data dal fatto che sul primo, ovvero sul Road Test, vengono effettuati molti più fuel cut-off rispetto all'altro ciclo perché sono presenti più frenate essendo in strada, la centralina è quindi costretta ad effettuare delle mancate iniezioni per rallentare più velocemente il regime motore. A causa dell'eccessivo errore sul primo ciclo non si può considerare come un modello valido per la predizione del MFB50 su ogni ciclo di guida, ma può essere considerato come un punto di partenza, infatti sono presenti numerosi parametri che si possono modificare per poter migliorare le prestazioni del modello, in questo elaborato ci siamo focalizzati in particolare sulla scelta ottimale del numero di HiddenSizes, ma si può anche agire sugli iperparametri della rete neurale oppure aumentare i dati di training per poter diminuire l'errore.

Bibliografia

[1] MathWorks , documentazione di Matlab , https://it.mathworks.com/help/matlab/

[2] Giovanni Busetti, *Performance assessment of recurrent neural network-based engine models for the prediction of combustion indexes under transient conditions*.

[3] Picciolo Michele, *Predizione di campi fluidodinamici mediante approcci basati sul Machine Learning.*

[4] Delia Esposito, *Torque model calibration of a motorcycle internal combustion engine.*

[5] A. Narayan Bhatt and N. Shrivastava. *Application of artificial neural network for internal combustion engines.* A state of the art review.

[6] L. Petrucci, F. Ricci, and et al. F. Mariani. *Performance analysis of artificial neural networks for control in internal combustion engines.*

[7] J. Fu, R. Yang, X. Li, X. Sun, Y. Li, Z. Liu, Y. Zhang, and B. Sunden. *Application of artificial neural network to forecast engine performance and emissions of a spark ignition engin.*

[8] Y. Tan and M. Saif. *Neural-networks-based nonlinear dynamic modeling for automotive engines.*

Ringraziamenti

Mi sembra doveroso ringraziare il prof. Alessandro Brusa per il suo grande impegno e la sua continua disponibilità durante tutto il periodo di preparazione della tesi, è stato un grande piacere potersi confrontare con lui e partecipare ai vari test condotti in laboratorio.

Voglio ringraziare il nonno Gigi, anche se non potrà leggere queste parole, perché se ho intrapreso questo percorso per parte è anche merito tuo che mi hai sempre detto che sarei diventato un bravo ingegnere meccanico ed alla fine ce l'abbiamo fatta nonno, ed anche la nonna Rosa per ogni aiuto.

Voglio ringraziare i miei nonni Bruno e Rosalba che si ricordavano dei miei esami più di me, sono sempre stati i primi ad interessarsi su come fosse andato l'esame e questo perché hanno sempre voluto il meglio da me e dal mio futuro.

Voglio ringraziare i miei genitori Paolo ed Eleonora, mi avete sempre supportato in ogni mia scelta, anche se sicuramente ne ho fatte tante di sbagliate ma forse affrontare questo percorso è stata una delle più giuste, mi avete aiutato in ogni dubbio ed in ogni scelta, anche se a volte abbiamo discusso alla fine è anche merito vostro se sono arrivato alla fine di questo percorso e vi voglio bene.

Voglio ringraziare Francesca per avermi sopportato in questo viaggio, in ogni momento di crisi e difficoltà, per ogni volta che ho pensato di mollare, per ogni pianto e problema ha sempre saputo ascoltarmi, nei momenti di incertezza ho sempre contato sulla tua presenza e di questo ti sono veramente grato.

Voglio ringraziare ogni mio amico e persona che in questo periodo mi ha dato anche solo un consiglio, che mi ha insegnato qualcosa o anche solo che mi ha strappato un sorriso, in particolare Marco e Jacopo che mi hanno aiutato a staccare la testa e a non pensare quando ce n'era bisogno con una colazione fuori e due chiacchiere o con un giro in moto.

Infine devo dire grazie a me stesso perché ce l'hai fatta a concludere questo percorso Leo, non è stato facile anche se è passato velocemente, certi esami sembravano impossibili ma un passo alla volta si sorpassa ogni ostacolo, sicuramente Leo incontrerai tante altre sfide nella tua vita ma ricordati che come hai vissuto momenti difficili in questi ultimi tre anni allora riuscirai ad affrontare tutto come alla fine hai sempre fatto e per una volta puoi veramente dirti bravo.