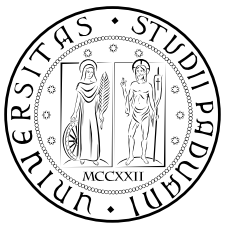


Università degli Studi di Padova

Dipartimento di Tecnica e Gestione dei Sistemi Industriali
Corso di Laurea Magistrale in Ingegneria Meccatronica

Tesi Magistrale

MODELLI DI IMPEDENZA DATA-DRIVEN PER INVERTER CONNESSI ALLA RETE

Relatore: prof. Paolo Mattavelli

Correlatore: Ing. Andrea Zilio

Laureando: Nicola Balasso
6056396

Anno Accademico: 2023/2024

Questo lavoro è stato composto con LaTeX e la classe suftesi di Ivan Valbusa. Il font con grazie è il Palatino di Hermann Zapf, mentre il font lineare è l'Iwona di Janusz M. Nowacki.

Indice

Elenco delle tabelle

Elenco delle figure

Introduzione

Nel prossimo futuro, sempre più risorse energetiche rinnovabili saranno disponibili e connesse alla rete elettrica, aumentando la capacità di fornire energia e in maniera più efficiente. Per connettere alla rete tali risorse occorrono convertitori in grado di adattare tale sorgente in una tensione alternata di opportuna ampiezza e frequenza. Tuttavia, l'interazione tra convertitori e la rete può generare instabilità, anche se presi singolarmente sono stabili. Tra i vari criteri di stabilità, quello basato sull'impedenza di uscita, applicando il criterio di Nyquist, risulta particolarmente comodo in quanto è sufficiente conoscere solo il rapporto tra l'impedenza di uscita del convertitore e quella di rete per verificare la stabilità del sistema. Tali parametri sono relativamente semplici da calcolare, anche per via sperimentale, tramite misure di corrente e tensione, quindi senza apportare modifiche invasive al sistema in esame.

In questa tesi si vedrà come modelli data-driven, quindi basati su reti neurali, possono essere impiegati nella stima dell'impedenza di uscita o, equivalentemente, dell'ammettenza dei convertitori connessi a rete. Tale approccio viene migliorato dall'impiego del transfer learning (TL), una tecnica di machine learning che consente di aumentare l'efficienza dell'utilizzo dei dati di addestramento sfruttando la conoscenza acquisita su un problema per risolverne un altro simile. In questo contesto, il TL consente di trasferire le informazioni tra diverse topologie di convertitori o tra convertitori analoghi ma con diversi parametri. Esistono due scenari in cui il TL risulta particolarmente comodo. Il primo è quando sia ha a disposizione un modello analitico del sistema dal quale è possibile derivare un dataset iniziale per addestrare un modello. Successivamente, l'addestramento di fino viene fatto partendo dal modello preaddestrato e aggiungendo dei dati sperimentali. Il secondo è quando sono disponibili modelli addestrati su convertitori della stessa topologia o simili e, partendo da questi, è possibile ottenere il modello di un nuovo convertitore con meno dati rispetto ad un addestramento da zero.

Nella prima parte della tesi si tratteranno gli aspetti teorici nella progettazione di una rete neurale e del criterio di stabilità basato sull'impedenza. Successivamente, si andrà più in dettaglio nell'analisi della struttura di un inverter trifase connesso a rete e di come i vari anelli di controllo influenzano l'ammettenza del convertitore, arrivando ad un modello analitico che sarà usato poi nella costruzione di un dataset. L'ultima parte tratterà le varie architetture delle reti neurali utilizzate per stimare le componenti dell'ammettenza, mostrando anche come si possano ottenere dei modelli migliori ottimizzando gli iperparametri delle reti con l'utilizzo dei tuner integrati nelle librerie. Infine, si vedranno i vantaggi offerti dall'utilizzo del TL, in particolare di due metodi di TL, per stimare l'ammettenza di un convertitore diverso da quelli usati precedentemente, fornendo solo poche centinaia di dati per l'addestramento della rete, contro le migliaia degli altri dataset.

Capitolo 1

Artificial Intelligence e Deep Learning

Il deep learning e le intelligenze artificiali (IA) sono concetti strettamente legati e stanno prendendo piede velocemente in numerosi settori, dall'informatica alla medicina, dalla finanza al settore industriale. L'intelligenza artificiale si occupa di creare sistemi informatici in grado di eseguire compiti che richiedono solitamente l'intelligenza umana, come, ad esempio, riconoscimento di immagini, traduzione automatica ecc... Il deep learning è una sottocategoria dell'IA che si concentra sull'addestramento di reti neurali artificiali. Queste reti neurali sono in grado di apprendere automaticamente relazioni complesse tra i dati forniti attraverso l'esposizione a grandi quantità di esempi (dataset). Questo processo è detto "addestramento" e consiste nel fornire ripetutamente alla rete i dati di addestramento, che adatta i pesi dei neuroni per minimizzare l'errore tra le previsione della rete e i valori attesi. Questi concetti saranno meglio presentati nella sezione 1.1.

Le reti neurali, una volta addestrate, sono in grado di svolgere compiti anche molto complessi. Uno degli utilizzi principali è quello di approssimare una funzione non lineare, spesso ignota o difficile da calcolare. In ambito dell'elettronica di potenza può voler dire stimare la relazione tra dei parametri di un sistema e le sue uscite. Un esempio è la stima della relazione tra alcuni parametri di controllo di un convertitore e della tensione, corrente e potenza prodotte. Il calcolo di tale relazione può essere piuttosto complicato e non sempre possibile. Si pensi al caso di componenti le cui caratteristiche e parametri non sono noti, in questo caso l'approccio black-box concesso dalle reti neurali è un grande vantaggio.

Uno dei tipi di rete più semplici ed utilizzati è la feedforward, utilizzata per questa tesi.

1.1 Feedforward Neural Networks

Una rete feedforward o FNN è una rete in cui i nodi al suo interno non formano dei cicli, ovvero, l'informazione scorre unidirezionalmente la rete. L'elemento fondamentale costituente una rete neurale è il neurone artificiale, la cui struttura generale è mostrata in figura 1.1.

Come si vede dall'immagine, ogni neurone è un sistema Multiple Input Single Output (MISO), il quale esegue una combinazione lineare pesata (ω_i) di tutti gli input, aggiunge un bias (ω_b), e il risultato viene passato attraverso una funzione di attivazione. Essa è di solito non lineare proprio per introdurre la capacità di rappresentare andamenti non lineari alla rete neurale. Tipiche funzioni di attivazioni sono, ad esempio, Sigmoide, tangente iperbolica, relu.

Quando più neuroni vengono connessi insieme si ottiene una rete neurale la quale è in grado di rappresentare relazioni ingresso-uscita anche molto complesse. La struttura così ottenuta è mostrata in figura 1.2. Le connessioni tra neuroni possono essere di diversi tipi dando origine a diverse classi di reti, ma in ogni caso sono organizzati in strati (layer). Questi layer si possono dividere in tre categorie:

- **Layer di input**: i neuroni di questo layer ricevono i dati di ingresso e li manda ai layer successivi. Cosa importante è che il numero di neuroni in questo strato coincida con la dimensione di parametri di addestramento.
- **Layer nascosti**: questi layer separano quelli di ingresso e uscita e possono essercene diversi composti da molti neuroni. I pesi di questi neuroni vengono costantemente aggiornati durante l'addestramento, normalmente con unvalore compreso tra 0 e 1.
- **Layer di uscita**: ultimo strato della rete che restituisce i risultati stimati.

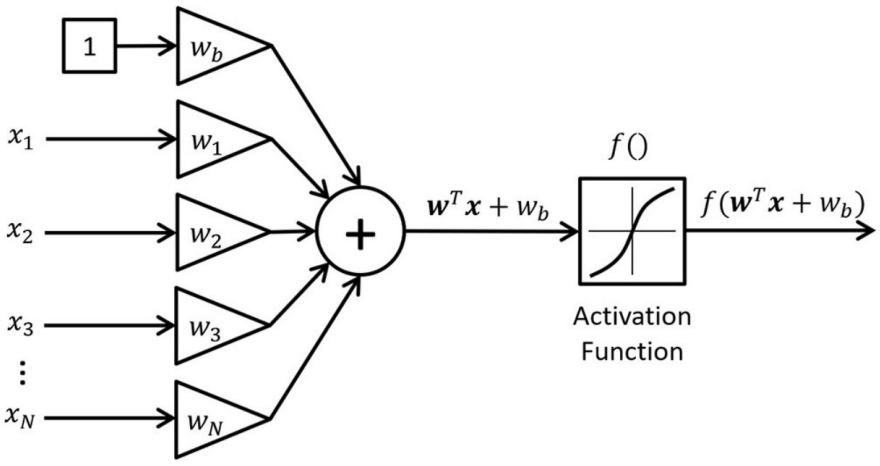


Figura 1.1. Struttura generale di un neurone artificiale [7].

Una rete viene detta "fully connected" se gli ingressi dei neuroni di un layer sono connessi con le uscite di quelli del layer precedente e le loro uscite sono connesse con gli ingressi dei neuroni del layer successivo.

Da tenere presente che una FANN è in grado di rappresentare solo funzioni statiche, cioè gli output dipendono solo dal valore attuale degli input e rimangono costanti per una data combinazione tempo-invariante degli input. Da questo si capisce che le FANN non sono indicate per modellizzare sistemi dinamici descritti da funzioni differenziali, proprio perché non contengono nessun elemento di memoria. L'analisi fatta in questa tesi, essendo in frequenza, consente, però, l'utilizzo di questo tipo di reti.

Considerando una FANN con un layer di ingresso, uno di uscita ed N layer nascosti, in generale, l'i-esimo layer ha k_i neuroni e il valore di questo è definito dalla seguente equazione

$$\boldsymbol{H}^{(i)} = \sigma\left(\boldsymbol{\omega}^{(i)}\boldsymbol{H}^{(i-1)} + \boldsymbol{b}^{(i)}\right) \tag{1.1}$$

dove σ è la funzione di attivazione, $\boldsymbol{H}$ la matrice dei valori dei neuroni, $\boldsymbol{\omega}$ i pesi e $\boldsymbol{b}$ i bias dell'i-esimo layer, così definiti

$$\boldsymbol{H}^{(i)} = \begin{bmatrix} H_1^{(i)} & H_2^{(i)} & \dots & H_{k_i}^{(i)} \end{bmatrix}^T, \tag{1.2}$$

$$\boldsymbol{b}^{(i)} = \begin{bmatrix} b_1^{(i)} & b_2^{(i)} & \dots & b_{k_i}^{(i)} \end{bmatrix}^T, \tag{1.3}$$

$$\boldsymbol{\omega}^{(i)} = \begin{bmatrix} \omega_{1,1}^{(i)} & \omega_{2,1}^{(i)} & \dots & \omega_{k_{i-1},1}^{(i)} \\ \omega_{1,2}^{(i)} & \omega_{2,2}^{(i)} & \dots & \omega_{k_{i-1},2}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{1,k_i}^{(i)} & \omega_{2,k_i}^{(i)} & \dots & \omega_{k_{i-1},k_i}^{(i)} \end{bmatrix} \tag{1.4}$$

Nel caso del primo layer, la matrice dei valori è

$$\boldsymbol{H}^{(1)} = \sigma\left(\boldsymbol{\omega}^{(i)}\boldsymbol{I} + \boldsymbol{b}^{(1)}\right) \tag{1.5}$$

dove la matrice $\boldsymbol{I} = [I_1, I_2 \dots I_n]^T$ contiene gli n neuroni di ingresso, rappresentativi dei parametri di ingresso della rete. Mentre, per il layer di uscita, la matrice dei valori è

$$\boldsymbol{O} = \boldsymbol{\omega}^{(N+1)}\boldsymbol{H}^{(N)} + \boldsymbol{b}^{(N+1)} \tag{1.6}$$

dove $\boldsymbol{O} = [O_1, O_2 \dots O_m]^T$ contiene gli m neuroni di uscita, rappresentativi dei parametri di uscita della rete.

Durante l'addestramento sono di particolare importanza due funzioni: la funzione di costo (cost function) e perdita (loss function). La funzione di costo, anche chiamata funzione obiettivo, è una misura della discrepanza tra

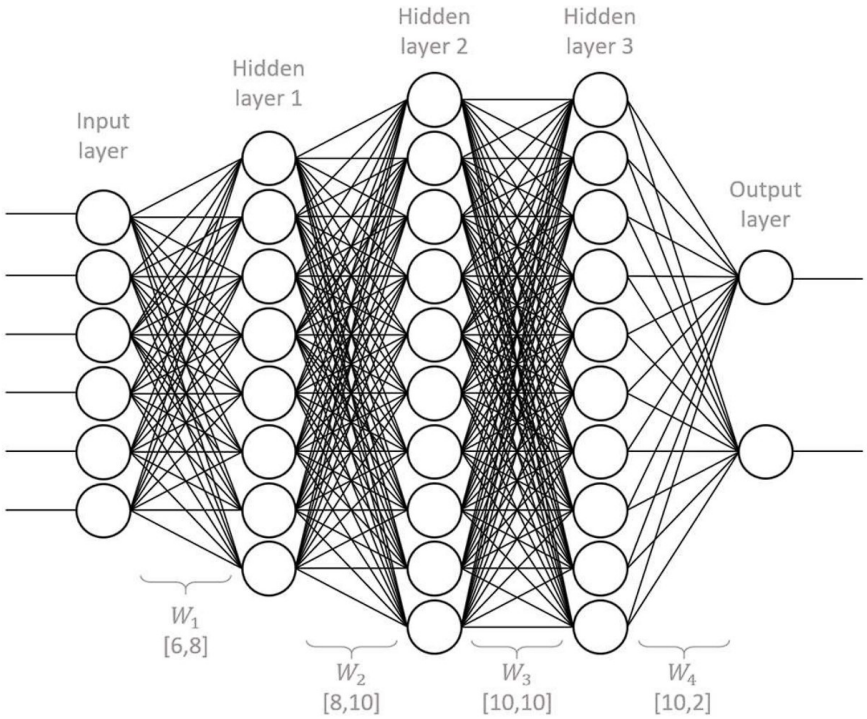


Figura 1.2. Esempio di struttura di una rete neurale fully-connected con un layer di ingresso, uno di uscita e 3 layer nascosti, [7].

l'output previsto della rete neurale e l'output desiderato per un dato di addestramento. L'obiettivo dell'addestramento della rete neurale è minimizzare questa funzione di costo, in modo che il modello possa produrre risultati sempre più accurati. Ci sono diverse funzioni di costo, a seconda del tipo di problema. Ad esempio, la funzione di costo quadratica è spesso usata per problemi di regressione, mentre la funzione di entropia incrociata (cross entropy loss) è popolare per problemi di classificazione.

Di seguito si vedrà più in dettaglio quali sono i passaggi principali per addestrare correttamente una rete.

1.2 Creazione del dataset

Prima ancora di iniziare l'addestramento vero e proprio, è importante preparare il dataset che verrà poi diviso in parti utilizzate nelle varie fasi dell'addestramento. È logico pensare che più cura verrà usata in questa fase, migliori saranno i risultati ottenuti con l'addestramento.

Prima di tutto, è necessario raccogliere i dati che veramente sono necessari per la rete. È inutile, e sbagliato, fornire qualsiasi tipo di dato che non sia strettamente necessario. Così facendo si rende inutilmente complesso l'addestramento rischiando di non arrivare a risultati soddisfacentielaremesso di avere a disposizione dei dati che rispettano questa condizione, si suddivide il dataset in tre parti:

- **Set di addestramento**: questo è la porzione più grande che contiene i dati usati per addestrare il modello. Esso deve includere ogni possibile caso che il modello possa incontrare.
- **Set di test**: questa porzione di dati serve per verificare il corretto funzionamento del modello addestrato. Si utilizzano dei dati che la rete non ha mai visto, simulando, quindi, esempi di un contesto reale. La fase di test permette di capire se il modello è in grado di generalizzare sufficientemente e di fornire previsioni accurate.
- **Set di validazione**: questa porzione viene usata durante l'addestramento e serve per regolare i parametri della rete e valutare le performance.

Suddivisioni tipiche del dataset sono mostrate in fig. 1.3 [1].

1 https://www.v7labs.com/blog/train-validation-test-set

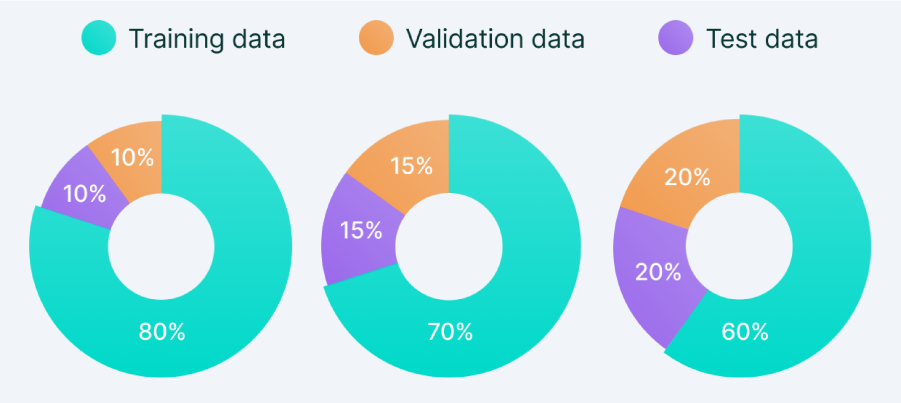


Figura 1.3. Tipiche percentuali di spartizione di un dataset.

1.2.1 Preprocessing dei dati

Dati mancanti o duplicati, estremi o inconsistenti possono disturbare l'addestramento del modello e creare errori e previsioni non consistenti. Perciò la prima fase di preprocessing è molto importante e si può dividere in 4 passaggi: pulizia dei dati, integrazione, trasformazione e riduzione della dimensione.

Pulizia dei dati

La pulizia dei dati comprende operazioni come l'inserimento di dati mancanti, filtraggio, rimozione di dati sporadici ecc...Quando si hanno dati mancanti, soluzioni tipiche possono essere l'inserimento di tali valori tramite metodi di stima o l'inserimento manuale.

A volte i dati possono essere ottenuti da misure e, quindi, affetti da rumore sovrapposto. In questo caso la qualità dei sensori e del sistema di acquisizione devono essere di buona qualità per ridurre al minimo il rumore a priori. Esistono tecniche di elaborazione che possono ulteriormente filtrare i dati, una volta acquisiti. Metodi comuni sono il binning, regressione e il clustering. Il binning consiste nel suddividere i dati in gruppi più piccoli, indipendenti tra di loro, che vengono elaborati individualmente. I dati presenti in ogni sottoinsieme possono poi essere sostituiti dalla loro media o essere elaborati in altro modo. Metodi di regressione sono solitamente usati per previsioni e approssimano i dati raccolti in una funzione lineare o polinomiale. Infine, il clustering consiste nel raggruppare i dati simili tra loro. I valori che non rientrano nelle categorie possono essere trattati come rumore ed eliminati.

Integrazione dei dati

Spesso le fonti dei dati sono diverse e c'è la necessità di unificare il tutto in un unico grande dataset. Questo può portare a problemi di compatibilità tra diversi formati dei dati o valori ridondanti o non consistenti.

Trasformazione dei dati

Una volta filtrati e organizzati i dati, è opportuno adattarli agendo sul loro valore e forma attraverso la normalizzazione. Ciò permette di scalare il dataset entro un certo range, spesso 0-1, e riduce la dispersione dei dati, semplificando l'addestramento del modello. È importante tenere a mente che, se si addestra una rete neurale con valori compresi tra 0 e 1, ogni dato che verrà fornito successivamente dovrà essere normalizzato alla stessa maniera e le previsioni dovranno essere denormalizzate per riacquistare un significato fisico.

Riduzione della dimensione

Arrivati a questo punto, si può avere un dataset molto esteso e potrebbe essere troppo grande per poter essere gestito e analizzato facilmente. Spesso è sufficiente un dataset più piccolo per ottenere un modello sufficientemente

addestrato. La dimensione giusta dipende da caso a caso e spesso la si trova per tentativi. L'importante è che la riduzione non sia tale da compromettere troppo i risultati forniti dal modello.

1.3 Addestramento

Una volta raccolti dati sufficienti, filtrati, uniti e normalizzati si può passare all'addestramento vero e proprio. Per eseguire un addestramento efficace è utile comprendere come esso viene fatto e quali sono i criteri in base ai quali una rete adatta i propri pesi. Fondamentali in questa fase sono la scelta della funziona di costo e l'algoritmo di ottimizzazione. L'addestramento equivale ad un problema di minimizzazione della funzione costo ed esistono diversi algoritmi di ottimizzazione, basati sul gradiente o meno. Tra i più famosi esiste lo stochastic gradient descent (SGD) o algoritmi più recenti ed evoluti, come l'Adam che sarà presentato successivamente.

1.3.1 SGD

L'SGD è un algoritmo iterativo che si basa sostanzialmente sulla discesa del gradiente, ma anziché calcolare il gradiente sull'intero set di dati in una singola iterazione, seleziona casualmente un sottoinsieme di dati, chiamato mini-batch, e viene calcolato il gradiente solo su questo sottoinsieme. Questo rende l'algoritmo più efficiente a livello computazionale. Si prenda un esempio per comprendere meglio.

Si supponga di voler minimizzare la funzione $f(x) = x^2$, il minimo si trova in $x = 0$. Partendo da un valore casuale, ad esempio $x = 2$, l'algoritmo calcola il gradiente in quel punto, che è 4. Ciò significa che avanzando di un piccolo step nella direzione crescente delle x la funzione crescerà. Perciò occorre spostarsi nella direzione opposta. Procedendo in questo modo si arriverà al minimo locale (che corrisponde anche al minimo globale) della funzione. La grandezza dello step prende il nome di learning rate, parametro molto importante nell'addestramento delle reti neurali. Una situazione ottimale è mostrata in fig. 1.4[2]. Traducendo questo concetto nell'ambito del machine

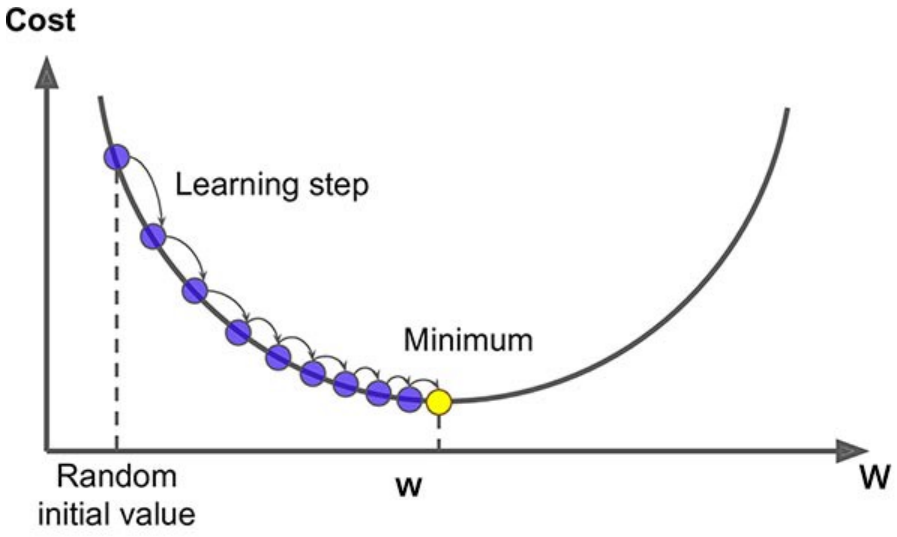


Figura 1.4. Convergenza del gradiente discendente.

learning, ciò che si vuole è modificare i pesi del modello per minimizzare la funzione costo. Ad ogni iterazione viene calcolato il gradiente della funzione costo e viene fatto uno spostamento nella direzione opposta pari al learing rate scelto, così facendo la funzione diminuirà fino a trovare un minimo locale. L'aggiornamento dei singoli pesi corrisponde alla seguente regola:

$$\omega_j = \omega_{j-1} \cdot lr \cdot \frac{\partial f}{\partial \omega_j} \tag{1.7}$$

dove lr è il learning rate ed f la funzione costo. È importante scegliere un valore adeguato di questo parametro: se lo si sceglie troppo grande c'è il rischio di oltrepassare il punto di minimo, se è troppo piccolo l'addestramento sarà

2 https://mlpills.dev/es/machine-learning-es/gradiente-descendente/

molto lento. Esitono varianti dell'SGD che possiedono migliori capacità di raggiungere il punto di minimo delle funzioni costo, una molto comune è l'SGD con momento. Tale algoritmo introduce il concetto di "inerzia" durante l'aggiornamento dei parametri del modello, che vengono modificati non solo tenendo conto del gradiente attuale ma anche quello dei passi precedenti. Ciò può aiutare ad accelerare l'addestramento.

Esistono anche algoritmi più evoluti che sono in grado di adattare il proprio learning rate, come l'Adam che è stato usato per l'addestramento delle reti in questa tesi.

1.3.2 Adam

Adam è un algoritmo di ottimizzazione con tasso di apprendimento adattivo ed è una combinazione di vari algoritmi, tra cui l'SGD con momento. Durante l'aggiornamento dei parametri del modello, viene calcolato il momento del primo ordine, che tiene conto dei gradienti delle iterazioni precedenti. Inoltre, Adam utilizza un momento del secondo ordine, che è una stima dei gradienti quadrati. Questo momento tiene traccia dell'ampiezza dei gradienti lungo ciascuna dimensione del parametro. Questo aiuta ad adattare il tasso di apprendimento per ciascun parametro in base alla sua importanza e alla variazione della funzione di costo lungo quella dimensione. Per ogni parametro del modello, Adam calcola una stima del momento del primo e secondo ordine. Queste stime vengono utilizzate per aggiornare il parametro, insieme a un tasso di apprendimento adattivo.

Adam risulta efficace in molti scenari, tuttavia è importante regolare attentamente gli iperparametri come il tasso di apprendimento e i coefficienti di decadimento dei momenti.

Durante l'addestramento di un modello possono insorgere due problemi tipici: l'overfitting e underfitting. Per riconoscerli occorre monitorare attentamente l'errore dei dati di addestramento e validazione. L'overfitting è il problema in cui il modello si è specializzato troppo sul dataset di addestramento e non è in grado di generalizzare il problema. In questo cas, il modello avrà un errore piccolo sui dati fornitegli ma molto più grande su dati nuovi. Il problema di underfitting, al contrario, si verifica quando il modello non è riuscito ad estrapolare una relazione sufficientemente accurata tra i dati fornitegli. Per comprendere meglio questi concetti si veda un esempio nel caso di preblemi di regressione.

Un buon modello dovrebbe avvicinarsi abbastanza alla curva che meglio approssima la relazione tra i vari punti, come in fig. 1.5[3], anche se non perfetto. La linea gialla è la previsione del modello, quella grigia la curva che meglio approssima i punti.

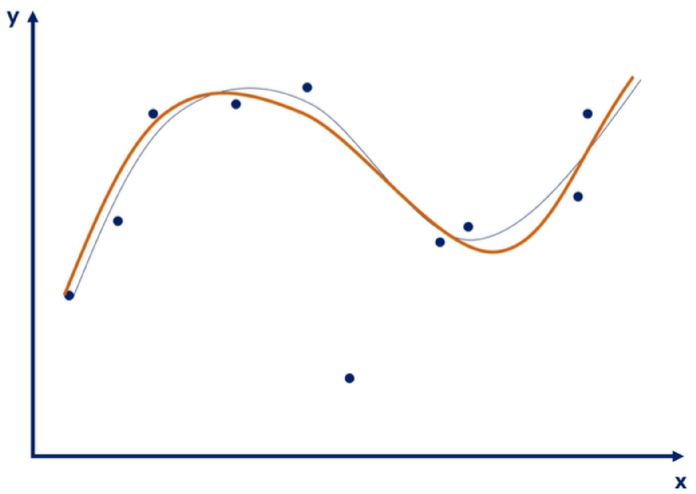


Figura 1.5. Buon modello che approssima la relazione cercata. Basso errore e alta accuratezza.

Nel caso di overfitting, il modello restituisce una relazione molto buona dei punti di training. Può essere confusa con un ottimo addestramento, tuttavia il modello interpreta anche il rumore come informazione. Un esempio si vede in fig. 1.6[3]. Infine, il problema dell'underfitting si verifica quando il modello propone una soluzione che non è sufficientemente accurata per rappresentare i dati e quindi non ha una capacità adeguata di previsione. Nel caso in esame, una soluzione lineare è una soluzione troppo semplice, come si vede in fig. 1.7[3].

3 https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/

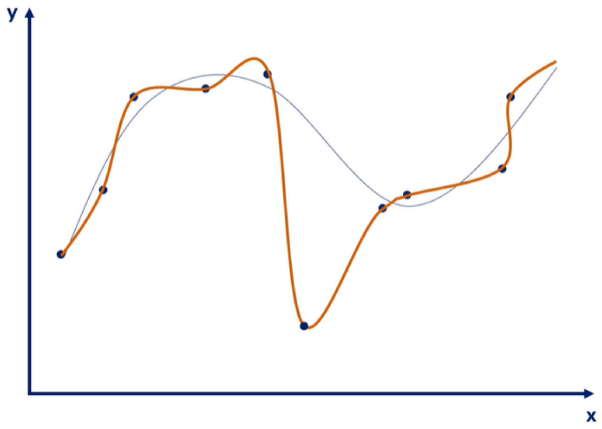


Figura 1.6. Problema di overfitting. Il modello si è specializzato sui dati di training "perdento di vista" il problema generale. Basso errore, bassa accuratezza.

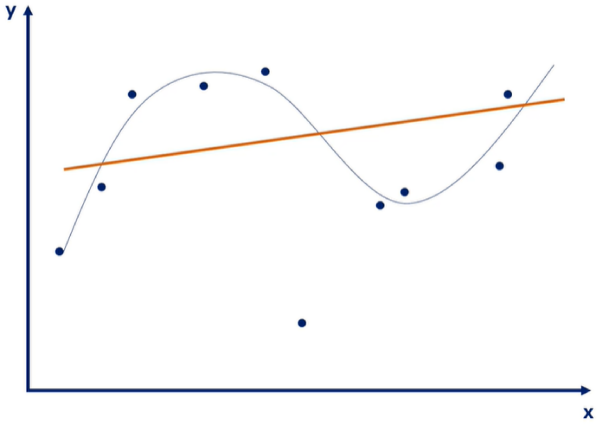


Figura 1.7. Problema di underfitting. La soluzione proposta è inconsistente con il problema esaminato. Alto errore e bassa accuratezza.

Durante l'addestramento è importante monitorare l'errore dei dati di addestramento e paragonarlo con quelli di validazione. Spesso è possibile riconoscere l'inizio di overfitting quando l'errore di training si riduce mentre quello di validazione tende a salire, proprio perché il modello comincia a perdere di generalità. Questa situazione è rappresentata in fig. 1.8[4]. Una soluzione spesso adottata è interrompere l'addestramento quando l'errore di validazione comincia a salire, ovvero quando il modello comincia ad entrare nella zona di overfitting.

1.4 Transfer Learning

Il transfer learning è una tecnica di machine learning in cui una IA preaddestrata per un compito viene riusata per un nuovo compito, simile al precedente. Solitamente, i modelli vengono addestrati da zero per ogni nuovo compito. Il transfer learning utilizza un approccio diverso: invece di creare un nuovo modello utilizzando un grande dataset, se ne utilizza uno addestrato in precedenza come punto di partenza, che possiede già caratteristiche utili per risolvere il nuovo problema. Concettualmente, è molto simile a voler insegnare a sciare ad un pattinatore. I due compiti sono simili tra loro e la persona possiede già una buona conoscenza di base. Ciò, intuitivamente, lo renderà più avvantaggiato nell'imparare la nuova attività e ci metterà meno tempo.

Il tranfer learning si può dividere in due step principali.

4 https://datahacker.rs/018-pytorch-popular-techniques-to-prevent-the-overfitting-in-a-neural-networks/

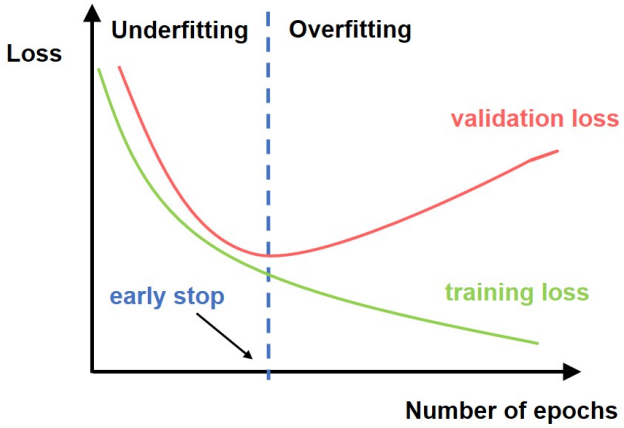


Figura 1.8. Problema di overfitting e underfitting durante l'addestramento.

1. **Preaddestramento**: in questo passo, viene creato un modello addestrato su un grande dataset per un certo compito. Questo fornisce al modello delle conoscenze generali utili successivamente.
2. **Addestramento specifico**: dopo il preaddestramento, si procede con un secondo addestramento sul problema specifico che si vuole risolvere. Durante questo processo si fornisce un dataset molto più ridotto relativo al problema specifico.

Tale approccio fornisce diversi vantaggi [3].

- Addestramento più rapido. Dal momento che il modello parte da una conoscenza preesistente, di solito sono richieste meno epoche per ottenere risultati soddisfacenti.
- Dipendenza dai dati ridotta. Il transfer learning può mitigare la necessità di un dataset molto vasto.
- Migliore generalizzazione. Le informazioni apprese dal modello preaddestrato sono spesso più robuste e generalizzabili, migliorando le performance sul compito specifico.
- Potenza di calcolo inferiore. Addestrare un modello da zero può richiedere un onere computazionale molto intenso, con il transfer learning si possono ridurre di molto le risorse richieste.

La tecnica del transfer learning risulta utile, ad esempio, in due scenari comuni [3]:

- Estrapolazione di un modello basato su dati reali a partire da uno basato su dati analitici. Si supponga di avere un modello addestrato su un gran numero di dati analitici derivati da un tipo di convertitore A, i cui parametri sono tutti noti. Successivamente si riaddestra il modello con pochi dati reali ottenuti da un secondo convertitore B, analogo ad A ma con parametri diversi e ignoti. In questo modo, è possibile ottenere un modello sufficientemente accurato di entrambi i convertitori, dal momento che derivano da dati reali.
- Estrapolazione di un modello da un convertitore ad un altro. Si supponga di avere un modello preaddestrato su un grande dataset relativo ad un convertitore A. Con un dataset più limitato, relativo ad un convertitore B, si può riaddestrare il modello per ottenere i parametri del secondo convertitore.

Il transfer learning può essere fatto, come si vedrà nel capitolo 4, in due modi. Il primo consiste nel caricare un modello preaddestrato con un grande dataset. A questo punto, per non perdere l'informazione acquisita dal preaddestramento, si "congelano" i pesi della rete, ovvero si blocca l'aggiornamento di questi durante l'addestramento. Successivamente, si rimuovono alcuni layer dal modello e se ne aggiungono di nuovi , che sao proprio questi ad essere aggiornati. Così facendo, si ottiene una sorta di modello ibrido che conserva delle informazioni sul problema precedente sul quale era addestratogenerale e in più ha la capacità di szarsi su un nuovo problema, simile al precedente, grazie ai nuovi neuroni aggiunti. La rete neurale così ottenuta viene infine addestrata su un dataset raelativamente piccolo, poche centinaia di unti spesso sono sufficienti.

Il secondo metodo, invece, prevede il riaddestramento di tutto il modello senza modificarlo. In questo modo si aggiornano tutti i pesi della rete sul nuovo dataset, che rimane comunque di dimensione ridotta come nel primo metodo. L'unico parametro di controllo sul quale si può agire per migliorare il riaddestramento è il learning rate. In questa fase è necessaria qualche prova per capire quale sia il valore più adatto. Se troppo grande si rischia, oltre

ai problemi già citati, di alterare troppo le informazioni generali possedute dal modello preaddestrato, se troppo piccolo si rischia di non modificare a sufficienza i pesi della rete per specializzarsi sui nuovi dati.

Entrambi i metodi sono efficaci, come si vedrà, tuttavia ci sono alcune differenze generali che si possono dire. Il secondo metodo è generalmente più efficace perché si aggiornano tutti i pesi per convergere alla soluzione ottimale, a differenza del primo dove una buona parte rimane fissa. Questo comporta, però, una potenza di calcolo superiore e non trascurabile per modelli molto complessi. Il primo metodo è più vantaggioso da questo punto di vista, anche se la sua efficacia dipende dal numero di layer rimossi e aggiunti. In conclusione, la scelta del metodo migliore dipende dalla situazione specifica e dall'accuratezza che si vuole raggiungere. Nel caso di modelli semplici e dataset contenuti, i due metodi risultato equivalenti e non c'è una differenza sostanziale sull'errore offerto da entrambi.

Capitolo 2

Criterio di stabilità per convertitori connessi alla rete

2.1 Introduzione al problema

Negli ultimi anni, l'interazione tra inverter connessi alla rete è stata argomento di forte interesse, dovuto al fatto che sempre più risorse rinnovabili sono interfacciate alla rete tramite convertitori. Un aspetto cruciale è garantire la stabilità della rete e dei singoli inverter ad essa connessi. È noto che un'alta impedenza di rete può destabilizzare il controllo di corrente di un inverter e portare a risonanze o altre forme di instabilità.

Metodi tradizionali per l'analisi della stabilità del sistema sono, ad esempio, lo studio del luogo delle radici o altre tecniche nel dominio del tempo o frequenza. Tali metodi si basano sulla accurata conoscenza del modello dell'inverter, nonché l'effetto dell'accoppiamento con altri convertitori e dell'impedenza di rete in varie condizioni di lavoro.

Una tecnica ormai ben nota per l'analisi della stabilità di tali sistemi è il criterio basato sull'impedenza.

2.2 Criterio di stabilità basato sull'impedenza

Il concetto alla base di questo criterio è suddividere il sistema in esame in un sottosistema sorgente e uno di carico. Il primo viene modellato come un circuito equivalente di Thévenin o di Norton e il secondo è costituito dalla sua impedenza di ingresso. Il rapporto tra l'impedenza di uscita della sorgente e l'impedenza di ingresso del carico deve rispettare il criterio di stabilità di Nyquist affinché il sistema in questione sia stabile. L'utilizzo di questo approccio comporta diversi vantaggi rispetto ad altri metodi [4].

1. Verificare la stabilità del sistema in un dato punto di lavoro comporta il calcolo dell'impedenza del sottosistema sorgente e carico con la teoria dei sistemi lineari.
2. Aggiungere o togliere una sorgente o un carico, oppure cambiare punto operativo del carico modifica solamente una impedenza del sistema.
3. Le impedenze possono essere ottenute per via analitica, simulazioni o sperimentalmente.
4. Le analisi possono suggerire come devono essere modificate le impedenze per rendere il sistema stabile.

Nel caso semplice di un sistema AC, ad esempio un inverter è connesso a rete, occorre prima linearizzare il sistema con un'analisi ai piccoli segnali. Il sistema risultante è rappresentato da un generatore di tensione (V_s) in serie con la sua impedenza (Z_s), rappresentato, quindi, come un circuito equivalente di Thévenin. La rete viene rappresentata come un'impedenza di carico (Z_l). Lo schema risultante è mostrato in figura 2.1.

Nel dominio della frequenza, la corrente circolante è data da:

$$I(s) = \frac{V_s(s)}{Z_l(s) + Z_s(s)} = \frac{V_s(s)}{Z_l(s)} \cdot \frac{1}{1 + Z_s(s)/Z_l(s)} \tag{2.1}$$

Se la sorgente e il carico sono stabili, la stabilità del sistema si può studiare osservando il secondo termine della eq. (2.1). Infatti, ricorda la funzione ad anello chiuso di un sistema retroazionato negativamente dove il guadagno di

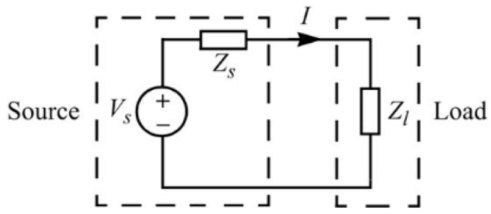


Figura 2.1. Circuito equivalente ai piccoli segnali di un generatore di tensione con carico [6].

anello è 1 e il guadagno di retroazione è $Z_s(s)/Z_l(s)$. La teoria dei sistemi lineari afferma che tale sistema è stabile se e solo se $Z_s(s)/Z_l(s)$ soddisfa il criterio di Nyquist.

C'è da sottolineare che il criterio di stabilità basato sull'impedenza presuppone che il generatore di tensione sia stabile quando non è collegato ad un carico. Tale condizione è spesso soddisfatta dal momento che la maggior parte dei generatori reali lavorano come generatori di tensione e rimangono stabili quando non collegati ad un carico. Tuttavia, gli inverter connessi a rete lavorano solitamente come generatori di corrente, percui non è possibile applicare il criterio così presentato [6]. Nel caso di sistemi controllati in corrente, torna utile una rappresentazione di Norton, anziché di Thévenin, fig. 2.2, costituita da un generatore ideale di corrente (I_s) in parallelo ad un'ammettenza (Y_s). Il carico è rappresentato dalla sua ammettenza di ingresso (Y_l)

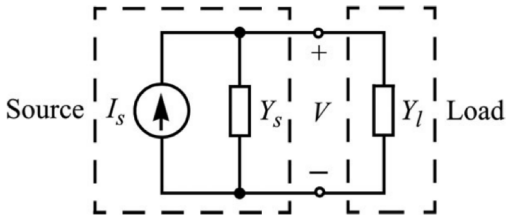


Figura 2.2. Circuito equivalente ai piccoli segnali di un generatore di corrente con carico [6].

La tensione ai capi del carico è data da:

$$V(s) = \frac{I_s(s)}{Y_s(s) + Y_l(s)} = \frac{I_s(s)}{Y_l(s)} \cdot \frac{1}{1 + Y_s(s)/Y_l(s)} \tag{2.2}$$

Similmente al caso precedente, il generatore di corrente si suppone stabile quando non collegato al carico (corto-circuito in uscita) e il carico è stabile quando alimentato da un generatore ideale di corrente. Sotto queste ipotesi, il sistema risulta stabile se e solo se $Y_s(s)/Y_l(s)$ soddisfa il criterio di Nyquist. Le condizioni necessarie, in questo caso, sono opposte al caso della rappresentazione di Thévenin, anche se i due circuiti sono matematicamente equivalenti. Occorre, quindi, distinguere sistemi controllati in tensione da quelli controllati in corrente perché questi ultimi non possono rimanere stabili quando la loro uscita è un circuito aperto e non viene fornito un percorso per la corrente. Di conseguenza, sistemi considerati come generatori di corrente non possono essere accuratamente rappresentati con Thévenin, ma occorre usare una rappresentazione del tipo di fig. 2.2.

2.2.1 Inverter connessi alla rete

Il modello più comune della rete elettrica è un generatore di tensione in serie con un'impedenza (Z_g), solitamente una resistenza in serie ad un'induttanza. Considerando il modello di un inverter come un generatore di corrente in parallelo con la sua impedenza di uscita, il sistema inverter-rete può essere rappresentato come in fig. 2.3. Questo può essere visto come un sistema ibrido costituito da un generatore di tensione e uno di corrente. Concordemente con le ipotesi precedenti, la rete risulta stabile se non connessa all'inverter e quest'ultimo risulta stabile se l'impedenza di rete è nulla.

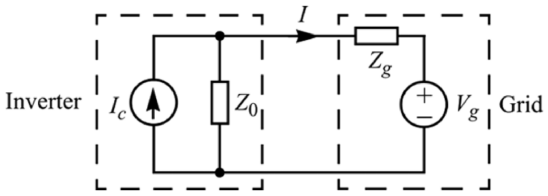


Figura 2.3. Circuito equivalente ai piccoli segnali di un inverter connesso a rete [6].

La corrente in uscita all'inverter, dopo alcuni passaggi, risulta:

$$I(s) = \left[I_c(s) - \frac{V_g(s)}{Z_0(s)} \right] \cdot \frac{1}{1 + Z_g(s)/Z_0(s)} \tag{2.3}$$

Con le considerazioni fatte e le ipotesi viste poco sopra, un inverter connesso a rete risulta stabile se $Z_g(s)/Z_0(s)$ soddisfa il criterio di Nyquist. Questa condizione suggerisce che un inverter dovrebbe avere un'impedenza di uscita più grande possibile per poter operare in varie situazioni della rete.

2.3 Criterio di Nyquist

Esistono numerosi approcci allo studio della stabilità di un sistema, come ad esempio i diagrammi di Bode, modellizzazione in spazio di stato, luogo delle radici ecc... Uno dei modi più usati è il criterio di Nyquist che si basa sullo studio della risposta in frequenza del sistema e sulla posizione dei poli e zeri nel piano complesso.

Considerando il sistema retroazionato di fig. 2.4, se $G(s)$ è la funzione di trasferimento ad anello aperto e $K(s)$ la funzione di feedback, la funzione di trasferimento ad anello chiuso è

$$H(s) = \frac{G(s)}{1 + G(s)K(s)} \tag{2.4}$$

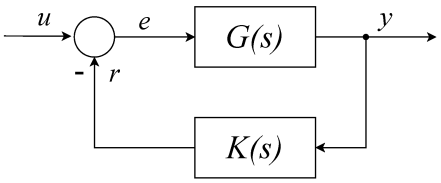


Figura 2.4. Sistema SISO retroazionato.

La stabilità del sistema (2.4) è data dagli zeri dell'equazione $1 + G(s)K(s)$. Il criterio di Nyquist afferma che:

Teorema 2.3.1 (Criterio di Nyquist). *Dato un sistema di controllo retroazionato con funzione di trasferimento H(s), il sistema è stabile se e solo se il diagramma completo di Nyquist della funzione ad anello aperto $G(s)K(s)$ non attraversa mai il punto critico (-1,0) del piano complesso e il numero di giri effettuati attorno tale punto in senso antiorario è pari al numero di poli di H(s) che giacciono nel semipiano destro del piano complesso.*

Tale teorema è valido per sistemi SISO, se si vogliono analizzare sistemi MIMO occorre fare riferimento ad criterio generalizzato di Nyquist.

Teorema 2.3.2 (Criterio di Nyquist generalizzato). *Sia dato un sistema retroazionato multivariabile il cui guadagno ad anello aperto sia privo di modi non controllabili e/o non osservalbili che giacciono nel semipiano destro. Allora, il sistema ad anello chiuso è stabile se e solo se la somma dei giri in senso antiorario attorno al punto -1+j0 da parte della funzione di trasferimento inversa di **L(s) = G(s)K(s)** è pari al numero di poli nel semipiano destro di **G(s)** e **K(s)**.*

2.4 Misure in dq

La descrizione di un sistema AC, ad esempio trifase, con una singola equazione è particolarmente complesso, data la tempo varianza delle grandezze del sistema. A volte può non bastare la linearizzazione ai piccoli segnali e in questi casi la rappresentazione nel sistema di riferimento sincrono *dqo* risulta particolarmente utile. La trasformazione di una terna dal sistema di riferimento *abc* a quello *dqo* è data come segue

$$T_{\text{abc-dqo}} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos(\theta) & \cos(\theta - \frac{2}{3}\pi) & \cos(\theta + \frac{2}{3}\pi) \\ -\sin(\theta) & -\sin(\theta - \frac{2}{3}\pi) & -\sin(\theta + \frac{2}{3}\pi) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} x_d \\ x_q \\ x_0 \end{bmatrix} = T_{\text{abc-dqo}} \cdot \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \tag{2.6}$$

Nel caso di sistemi bilanciati e simmetrici, privi di connessione di neutro, l'asse *o* può essere rimosso. Le variabili di un sistema trifase trasformate diventano, perciò, delle grandezze costanti, facilitando la linearizzazione. Nel caso di un sistema trifase bilanciato, esso diviene 2 circuiti DC interconnessi.

Per contro, però, si complica la rappresentazione dell'impedenza che diventa, ora, una matrice 2x2 in cui compaiono dei termini di accoppiamento tra la corrente di asse *q* e la tensione di asse *d* e viceversa. L'impedenza nel sistema *dq* viene di solito indicata come segue:

$$\boldsymbol{Z_{dq}}(s) = \begin{bmatrix} Z_{dd}(s) & Z_{dq}(s) \\ Z_{qd}(s) & Z_{qq}(s) \end{bmatrix} \tag{2.7}$$

La rappresentazione ai piccoli segnali di un generatore di tensione collegato a un carico è ancora valida (fig. 2.5), solo che il sistema non è più SISO ma MIMO.

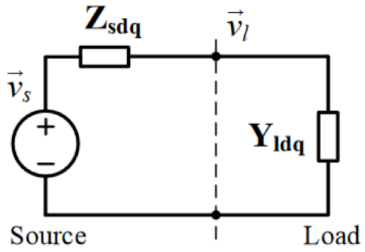


Figura 2.5. Circuito equivalente ai piccoli segnali di un generatore di tensione nel sistema di riferimento *dq* [4].

Anche in questo caso è possibile applicare il teorema di Nyquist, in particolare quello generalizzato per i sistemi MIMO.

2.4.1 Misura di ammettenza in *dq*

In questo capitolo si è visto come per garantire la stabilità di un inverter connesso a rete occorre conoscere la sua impedenza di uscita, o equivalentemente la sua ammettenza. In questa sezione si vogliono presentare delle tecniche per misurare l'ammettenza nel sistema *dq*. Esistono essenzialmente due modi per farlo: perturbare la tensione nel punto di connessione tra due sottosistemi o perturbare la corrente, come mostrato in fig. 2.6. La scelta tra perturbazione di tensione e corrente dipende dalle caratteristiche del sistema in esame. Perturbare la corrente conviene per sistemi con impedenza relativamente bassa, mentre perturbare la tensione conviene per sistemi con una impedenza relativamente alta.

Sfruttando la risposta di tensione e corrente al disturbo iniettato, è possibile calcolare l'ammettenza in *dq*. Dal momento che la trasformazione in *dq* rende il sistema MIMO con termini di accoppiamento tra gli assi, una variazione nella corrente in un asse, *d* o *q*, provoca una variazione nella tensione e corrente di entrambi gli assi. Va sottolineato, però, che tutte le ammettenze considerate sono valide per piccoli segnali. Questo permette di applicare la sovrapposizione degli effetti. Per ottenere misure accurate, il sistema deve rimanere a regime durante

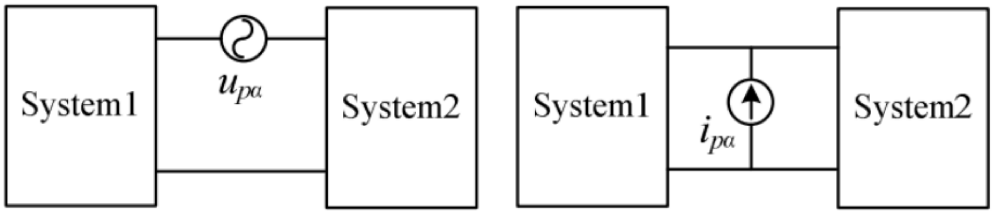


Figura 2.6. Tipi di perturbazione per sistemi AC [4].

l'iniezione del disturbo. Ciò significa che i vettori di tensione e corrente nel riferimento dq sono mappati come un singolo punto.

Il calcolo dell'ammettenza prevede di risolvere due sistemi lineari indipendenti. Ogni gruppo di equazioni corrisponde ad una eccitazione diversa del sistema e il modello dell'ammettenza ai piccoli segnali può essere calcolato linearizzando il sistema attorno al punto di lavoro, come mostrato nell'eq. (2.8).

$$\begin{cases} \begin{bmatrix} i_{d1}(s) \\ i_{q1}(s) \end{bmatrix} = \begin{bmatrix} Y_{dd}(s) & Y_{dq}(s) \\ Y_{qd}(s) & Y_{qq}(s) \end{bmatrix} \begin{bmatrix} v_{d1}(s) \\ v_{q1}(s) \end{bmatrix} \\ \begin{bmatrix} i_{d2}(s) \\ i_{q2}(s) \end{bmatrix} = \begin{bmatrix} Y_{dd}(s) & Y_{dq}(s) \\ Y_{qd}(s) & Y_{qq}(s) \end{bmatrix} \begin{bmatrix} v_{d2}(s) \\ v_{q2}(s) \end{bmatrix} \end{cases} \tag{2.8}$$

Dal momento che l'ammettenza non cambia, le equazioni si possono dividere per asse:

$$\begin{cases} \begin{bmatrix} i_{d1}(s) \\ i_{d2}(s) \end{bmatrix} = \begin{bmatrix} v_{d1}(s) & v_{q1}(s) \\ v_{d2}(s) & v_{q2}(s) \end{bmatrix} \begin{bmatrix} Y_{dd}(s) \\ Y_{dq}(s) \end{bmatrix} \\ \begin{bmatrix} i_{q1}(s) \\ i_{q2}(s) \end{bmatrix} = \begin{bmatrix} v_{d1}(s) & v_{q1}(s) \\ v_{d2}(s) & v_{q2}(s) \end{bmatrix} \begin{bmatrix} Y_{qd}(s) \\ Y_{qq}(s) \end{bmatrix} \end{cases} \tag{2.9}$$

Prendendo la trasposta e raggruppandole si ottiene:

$$\begin{bmatrix} i_{d1}(s) & i_{d2}(s) \\ i_{q1}(s) & i_{q2}(s) \end{bmatrix} = \begin{bmatrix} Y_{dd}(s) & Y_{dq}(s) \\ Y_{qd}(s) & Y_{qq}(s) \end{bmatrix} \begin{bmatrix} v_{d1}(s) & v_{q1}(s) \\ v_{d2}(s) & v_{q2}(s) \end{bmatrix}^T \tag{2.10}$$

La matrice delle ammettenze risulta quindi

$$\begin{bmatrix} Y_{dd}(s) & Y_{dq}(s) \\ Y_{qd}(s) & Y_{qq}(s) \end{bmatrix} = \begin{bmatrix} i_{d1}(s) & i_{d2}(s) \\ i_{q1}(s) & i_{q2}(s) \end{bmatrix} \begin{bmatrix} v_{d1}(s) & v_{q1}(s) \\ v_{d2}(s) & v_{q2}(s) \end{bmatrix}^{-T} \tag{2.11}$$

I vettori di tensione devono essere linearmente indipendenti per poter ottenere l'inversa della trasposta e occorrono due perturbazioni linearmente indipendenti per ottenere tutte le informazioni necessarie. All'inizio viene perturbata la tensione dell'asse d alla frequenza desiderata mantenendo la tensione dell'asse q a zero ($v_{q1}(s) = 0$). Successivamente, viene perturbato l'asse q mantenendo la tensione di asse d a zero ($v_{d2}(s) = 0$). Con queste considerazioni, la matrice delle ammettenze si calcola come:

$$\begin{bmatrix} \frac{i_{d1}(s)}{v_{d1}(s)} & \frac{i_{d2}(s)}{v_{q1}(s)} \\ \frac{i_{q1}(s)}{v_{d1}(s)} & \frac{i_{q2}(s)}{v_{q2}(s)} \end{bmatrix} = \begin{bmatrix} Y_{dd}(s) & Y_{dq}(s) \\ Y_{qd}(s) & Y_{qq}(s) \end{bmatrix} \tag{2.12}$$

Capitolo 3

Test case: Inverter trifase

I convertitori connessi a rete possono essere categorizzati in due tipi: inverter grid-forming (GFM) e grid-following (GFL). Entrambe le tipologie iniettano potenza in rete ma lo fanno in maniera diversa. Gli inverter GFM gestisce la tensione AC, contribuendo attivamente alla formazione di una sorgente di tensione. È in grado di sincronizzarsi alla rete grazie al controllo droop della frequenza, assomigliando al comportamento di un generatore sincrono. Un inverter GFL, invece, regola la corrente AC iniettata ed è in grado di sincronizzarsi automaticamente alla rete tramite un phase-locked loop (PLL). Ciò significa che questo tipo di convertitore necessita di una tensione di riferimento alla quale agganciarsi. Perciò, queste due tipologie hanno rappresentazioni equivalenti diverse. Un inverter GFM è descritto come un generatore di tensione in serie ad una piccola impedenza, mentre un inverter GFL viene descritto come un generatore di corrente pilotato in parallelo ad una grande impedenza. Da qui si intuisce il modo diverso che hanno le due tipologie di iniettare potenza in rete. I convertitori GFL forniscono potenza controllando la quantità di corrente iniettata, mentre i GFM controllano direttamente la tensione ai loro terminali di uscita. Questa distinzione è importante per rappresentare la configurazione equivalente corretta quando si utilizza il criterio basato sull'impedenza. In questa tesi si fa riferimento a convertitori GFL.

In questo capito si vedrà la struttura generale del convertitore esaminato, derivando la matrice dell'ammettenza a partire dai vari blocchi costituenti. Si vedrà poi il confronto tra il modello analitico e quello simulativo, arrivando infine alla creazione di un dataset da usare per l'addestramento della rete neurale.

3.1 Struttura del convertitore

La struttura del convertitore in esame è mostrata in fig. 3.1 ed è basata su quattro anelli di controllo:

1. **Controllo di corrente** (ACC): questo è l'anello più interno e veloce e ha il compito di controllare le correnti nel sistema di riferimento dq.
2. **Phase-Locked Loop** (PLL): questo sistema è usato per sincronizzare l'inverter con la tensione di rete. L'angolo θ prodotto dal PLL viene usato per tutte le trasformazioni in dq. Il corretto allineamento garantisce che la corrente i_d sia responsabile della potenza attiva iniettata e la corrente i_q di quella reattiva.
3. **Controllo di tensione DC** (DVC): questo anello di controllo si occupa di mantenere la tensione sul DC-link costante gestendo la corrente assorbita dal condensatore. Esso fornisce il riferimento di corrente i_d^{ref}.
4. **Compensazione della potenza reattiva** (AVC): normalmente si desidera ridurre al minimo la potenza reattiva circolante in rete. Questo controllo si occupa di ciò, fornendo il riferimento di corrente i_q^{ref}.

   Di seguito si mostreranno più in dettaglio i quattro anelli di controllo.

3.1.1 Notazioni usate

Prima di procedere all'analisi dei vari componenti, è opportuno definire le convenzioni usate. Data la natura multivariabile delle equazioni, conviene utilizzare una notazione complessa per le variabili. Una generica grandezza è quindi rappresentata come $\underline{x} = x_r + jx_i$. Per rappresentare forme matriciali si userà il grassetto ($\mathbf{Z}$).

Di seguito si farà uso di vari sistemi di riferimento, oltre a quello dq. La tensione di rete nel punto di connessione comune (PCC), nel riferimento $\alpha\beta$, è E^s. La tensione di rete nel riferimento dq è $E = E^s e^{-j\theta_g}$ e la tensione nel

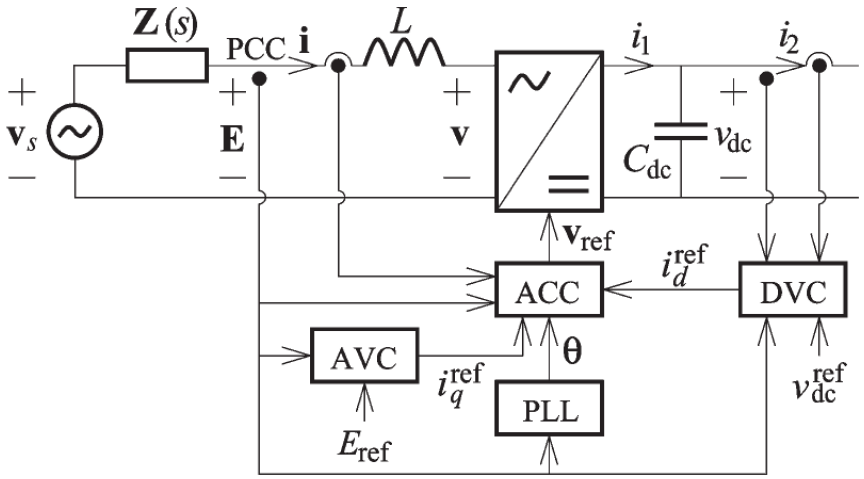


Figura 3.1. Struttura principale dell'inverter connesso a rete [2].

riferimento del convertitore è pari a $E^c = E^s e^{-j\theta_c}$, dove θ_c è la fase ottenuta dl PLL. Le due tensioni sono legate tra loro da:

$$E^c = e^{-j\Delta\theta} E, \quad \Delta\theta = \theta_g - \theta_c \tag{3.1}$$

3.1.2 Controllo di corrente

L'equazione che descrive la tensione ai capi del filtro L (o impedenza R+jω_gL se considerata anche una componente resistiva) è data da:

$$L\frac{d\underline{i}}{dt} + j\omega_g L\underline{i} = \underline{E} - \underline{v} \tag{3.2}$$

che nel riferimento del convertitore corrisponde a:

$$L\frac{d\underline{i}^c}{dt} + j(\omega_g + \Delta\omega)L\underline{i}^c = \underline{E}^c - \underline{v}^c \tag{3.3}$$

Dove $\Delta\omega = d\Delta\theta/dt$ è solitamente trascurabile rispetto a ω_g. Per poter disaccoppiare i_d da i_q, occorre progettare un regolatore PI per il riferimento di tensione PWM come segue [4]:

$$\underline{v}^{ref} = -\left(k_p + \frac{k_i}{s}\right)(\underline{i_{ref}} - \underline{i}^c) - j\omega_g L\underline{i}^c \tag{3.4}$$

Dove il riferimento di corrente $\underline{i_{ref}} = i_d^{ref} + ji_q^{ref}$ è generato dal controllo di tensione DC e AC. Dal momento che il controllore è progettato per disaccoppiare le correnti, l'anello di controllo può essere rappresentato come in fig. 3.2. Dove $PI(s)$ è la funzione di trasferimento del regolatore PI e $1/sL$ è la funzione ad anello aperto. Il termine

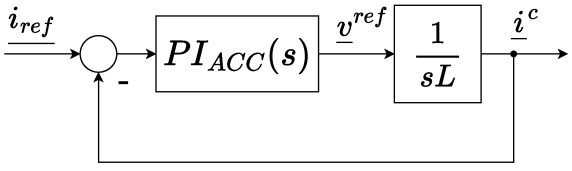


Figura 3.2. Schema del controllo di corrente.

proporzionale del regolatore viene definito a partire dalla banda passante desiderata, α_c, come $k_p = \alpha_c L$. Il termine integrale viene inserito per azzerare l'errore a regime dovuto a disaccoppiamento non perfetto e cadute di tensione sulla resistenza del filtro. Un valore che garantisce sufficiente margine di fase e buona reiezione ai disturbi di tensione è $k_i = \alpha_c^2 L/20$.

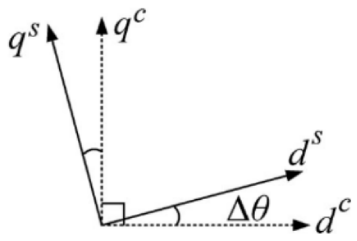


Figura 3.3. Disallineamento tra il sistema di riferimento *dq* del controllore e del sistema [4].

3.1.3 Phase-Locked-Loop (PLL)

Il PLL è un sistema di controllo retroazionato che ha lo scopo di fornire la fase di un segnale di ingresso, in modo da avere, a regime, il sistema di riferimento *dq* del controllore in fase con quello del vero del sistema. Quando la tensione di rete viene perturbata, la posizione del sistema *dq* viene cambiata e, a causa della dinamica limitata del PLL, il riferimento *dq* di controllo non è più allineato. Tale disallineamento è mostrato in fig. 3.3. Una perturbazione di tensione, quindi, si propaga all'angolo di uscita del PLL e fino al controllo di corrente. La perturbazione prosegue fino alla tensione generata dall'inverter e, infine, alla corrente di uscita. Ciò significa che la dinamica del PLL influenza l'impedenza di uscita dell'inverter e ne va tenuto conto. Lo schema a blocchi di un PLL è mostrato in fig. 3.4. Nel caso di una perturbazione ai piccoli segnali della fase $\tilde{\theta}$, questa viene retroazionata fino alla matrice di

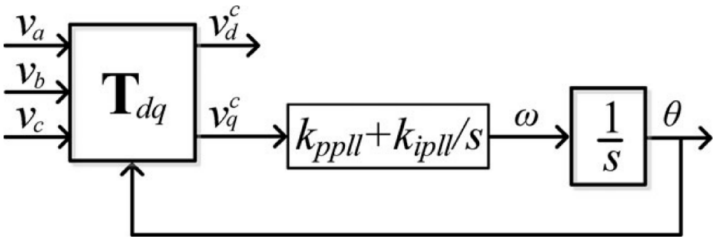


Figura 3.4. Schema a blocchi di un PLL [4].

trasformazione che genera un valore perturbato di tensione $\tilde{v_q}$. Per variazioni di fase $\Delta\theta$ tra i due riferimenti *dq* sufficientemente piccole, vale:

$$\tilde{v_q} = |v_{dq}| \cdot sin(\Delta\theta) \approx |v_{dq}| \cdot \Delta\theta \tag{3.5}$$

Di conseguenza, la funzione ad anello aperto è:

$$F_{PLL}(s) = PI_{PLL}(s) \cdot \frac{|v_{dq}|}{s} \tag{3.6}$$

Il design del regolatore PI è fatto in modo da garantire la larghezza di banda desiderata, solitamente di qualche Hz, e il margine di fase desiderato.

3.1.4 Controllo di tensione DC (DVC)

Il compito di questo controllo è di mantenere la tensione sul DC-link costante agendo sulla corrente i_d. Trascurando le perdite nel convertitore e considerando il controllo di tensione AC molto più veloce del DVC, è possibile legare la potenza attiva assorbita, P, con la potenza associata al DC-link, $P = i_1 \cdot v_{dc}$. Con un bilancio di potenza si ottiene:

$$\frac{1}{2} C_{dc} \frac{dv_{dc}^2}{dt} = P - P_L \tag{3.7}$$

dove P_L è la potenza del carico al lato DC: $P_L = v_{dc} \cdot i_2$. Se il DVC operasse direttamente sull'errore $v_{dc}^{ref} - v_{dc}$, la dinamica ad anello chiuso sarebbe dipendente dal punto operativo v_{dc0}. Per ovviare a questo problema, il PI del

DVC non opera sull'errore $v_{dc}^{ref} - v_{dc}$ ma su $\frac{(v_{dc}^{ref})^2 - v_{dc}^2}{2}$ [2]. Il riferimento di potenza attiva in uscita dal controllore diventa:

$$P_{ref} = PI_{DVC}(s) \cdot \frac{(v_{dc}^{ref})^2 - v_{dc}^2}{2} \tag{3.8}$$

Per ottenere il riferimento di corrente, la potenza P_{ref} viene divisa per la tensione nel punto PCC:

$$i_d^{ref} = \frac{P_{ref}}{|E|} \tag{3.9}$$

Perciò, la funzione ad anello aperto risulta:

$$F_{DVC}(s) = PI_{DVC}(s) \cdot \frac{1}{sC_{dc}} \tag{3.10}$$

Il design del PI segue lo stesso ragionamento di quello del PLL: si impone la banda passante desiderata, in questo caso qualche decina di Hz è sufficiente, e il margine di fase desiderato.

3.1.5 Compensazione della potenza reattiva (AVC)

Questo controllo fornisce il riferimento di corrente i_q^{ref} in modo da ridurre la potenza reattiva introdotta in rete. L'equazione del controllore è la seguente:

$$i_q^{ref} = PI_{AVC}(s) \cdot (E_{ref} - |E|) \tag{3.11}$$

Il design del PI dipende dall'impedenza della rete, quindi dalle condizioni e dal punto in cui viene collegato l'inverter.

3.2 Derivazione dell'ammettenza dal modello a piccoli segnali

In questa sezione si vedrà come derivare l'ammettenza a partire dai singoli componenti, precedentemente visti. Il calcolo dell'ammettenza si basa principalmente su [2] e [4]. Il punto di partenza sono le equazioni degli anelli di controllo presentate nella sezione precedente e sullo schema di fig: 3.1.

3.2.1 Anello di corrente

In questa analisi si trascura il ritardo del controllo digitale e dei tempi morti dell'inverter. Supponendo di ottenere un disaccoppiamento perfetto tra gli assi d e q e si trascurano gli anelli di controllo esterni (PLL e DVC), si può ottenere lo schema di fig. 3.5. Dallo schema è facile derivare la seguente equazione:

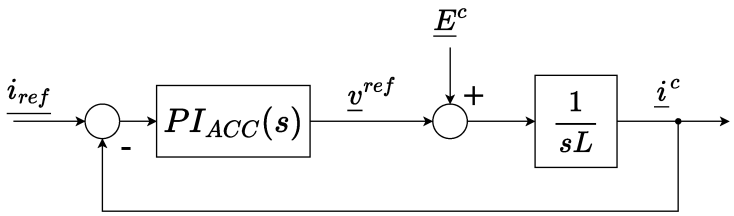


Figura 3.5. Schema dell'anello di corrente considerando anche l'influenza della tensione.

$$\underline{i}^c = \underline{G_c}\,\underline{i_{ref}} + \underline{Y_i}\,\underline{E^c} \tag{3.12}$$

Dove $\underline{G_c}$ e $\underline{Y_i}$ sono, rispettivamente, la funzione di trasferimento ad anello aperto e l'ammettenza di ingresso. Esse si possono esprimere come:

$$\underline{G_c}(s) = \frac{PI_{ACC}(s) \cdot \frac{1}{sL}}{1 + PI_{ACC}(s) \cdot \frac{1}{sL}} = g_c(s) \tag{3.13}$$

$$\underline{Y_i}(s) = \frac{\frac{1}{sL}}{1 + PI_{ACC}(s) \cdot \frac{1}{sL}} = y_i(s) \tag{3.14}$$

In presenza di una componente resistiva nel filtro del convertitore, la (3.13) e (3.14) diventano:

$$\underline{G_c}(s) = \frac{PI_{ACC}(s)}{sL + R + PI_{ACC}(s)} = g_c(s) \tag{3.15}$$

$$\underline{Y_i}(s) = \frac{1}{sL + R + PI_{ACC}(s)} = y_i(s) \tag{3.16}$$

In forma matriciale risulta:

$$\underline{i^c} = \underbrace{\begin{bmatrix} g_c(s) & 0 \\ 0 & g_c(s) \end{bmatrix}}_{\boldsymbol{G_c}(s)} \underline{i_{ref}} + \underbrace{\begin{bmatrix} y_i(s) & 0 \\ 0 & y_i(s) \end{bmatrix}}_{\boldsymbol{Y_i}(s)} \underline{E^c} \tag{3.17}$$

La scelta della banda passante del controllo di corrente, come visto, determina il valore dei parametri del regolatore PI. Come riportato in [4], più grande è il guadagno k_i, più grande è l'impedenza Z_{dd}. Questa considerazione è da tenere a mente in fase di progettazione, in quanto è desiderabile avere un'impedenza di uscita più alta possibile per poter operare in maniera stabile in più situazioni possibili.

3.2.2 Anello di tensione DC

Per analizzare l'influenza di questo anello sull'ammettenza di ingresso dell'inverter, occorre linearizzare il sistema. Il punto PCC è considerato come riferimento per la fase. Per ottenere il modello ai piccoli segnali, vengono considerate piccole perturbazioni indicate con il simbolo $\sim$: $\underline{E} = E_0 + \tilde{E_d} + j\tilde{E_q}$ e $\underline{i} = I_{d0} + \tilde{i_d} + j(I_{q0} + \tilde{i_q})$. In base a [2], la potenza attiva e reattiva possono essere calcolate come:

$$P = \mathfrak{Re}\left\{\underline{Ei^*}\right\} \approx \underbrace{E_0 I_{d0}}_{P_0} + \underbrace{I_{d0}\tilde{E_d} + I_{q0}\tilde{E_q} + E_0\tilde{i_d}}_{\tilde{P}} \tag{3.18}$$

$$Q = \mathfrak{Im}\left\{\underline{Ei^*}\right\} \approx \underbrace{-E_0 I_{q0}}_{Q_0} + \underbrace{I_{d0}\tilde{E_q} - I_{q0}\tilde{E_d} - E_0\tilde{i_q}}_{\tilde{Q}} \tag{3.19}$$

Come si vede dall'equazione, le correnti i_d e i_q sono legate rispettivamente alla potenza attiva e reattiva. Assumendo un convertitore ideale (privo di perdite) e un anello di controllo AC molto più veloce del DVC, è possibile eguagliare la potenza attiva assorbita con quella associata al DC-link. L'equazione della dinamica del DC-link diventa quindi:

$$C_{dc} v_{dc0} \frac{d\tilde{v}_{dc}}{dt} = \tilde{P} - \tilde{P_L} \tag{3.20}$$

dove v_{dc0} è il riferimento di tensione del DC-link. Se il controllore è progettato in base alle equazioni (3.9) e (3.8), tali equazioni di possono linearizzare come:

$$\tilde{P}_{ref} = -v_{dc0} PI_{DVC}(s)\tilde{v}_{dc} \tag{3.21}$$

$$\tilde{i_d}^{ref} = \frac{\tilde{P}_{ref}}{E_0} = -\frac{V_{dc0}}{E_0} PI_{DVC}(s)\tilde{v}_{dc} \tag{3.22}$$

Usando le equazioni (3.12), (3.15) e (3.16)), si ottiene:

$$\tilde{i_d} = -\frac{v_{dc0}}{E_0} g_c(s) PI_{DVC}(s)\tilde{v}_{dc} + y_i(s)\tilde{E_d} \tag{3.23}$$

Quest'ultima equazione viene sostituita nella (3.18) e la risultante $\tilde{P}$ viene sostituita nella (3.20). Ne risulta:

$$\tilde{v}_d c = \frac{[E_0^2 y_i(s) + P_0]\tilde{E_d} - Q_0\tilde{E_q}}{E_0 V_{dc0}[sC_{dc} + g_c(s) PI_{DVC}(s)]} \tag{3.24}$$

Infine, è possibile scrivere l'equazione finale di $\tilde{i_d}^{ref}$ sostituendo la (3.24) nella (3.22):

$$\tilde{i_d}^{ref} = -G_{dc}^d(s)\tilde{E_d} + G_{dc}^q(s)\tilde{E_q} \tag{3.25}$$

dove:

$$G_{dc}^{d}(s) = \frac{[y_i(s) + P_0/E_0^2]PI_{DVC}(s)}{sC_{dc} + g_c(s)PI_{DVC}(s)} \tag{3.26}$$

$$G_{dc}^{q}(s) = \frac{Q_0 PI_{DVC}(s)}{E_0^2[sC_{dc} + g_c(s)PI_{DVC}(s)]} \tag{3.27}$$

In forma matriciale, la (3.25) risulta:

$$\tilde{i_d}^{ref} = \underbrace{\begin{bmatrix} -G_{dc}^{d}(s) & G_{dc}^{q}(s) \\ 0 & 0 \end{bmatrix}}_{\boldsymbol{G_{Ei}}(s)} \tilde{E} \tag{3.28}$$

Nel caso si implementi un anello di controllo della tensione alternata, la componente *qd* della matrice $\boldsymbol{G_{Ei}(s)}$ è pari a $-PI_{AVC}(s)$. In [2] viene suggerito di scegliere una banda passante dell'anello DVC che sia non più grande di $0.1\alpha_c$, soprattutto in casi in cui si hanno risonanze poco smorzate in bassa frequenza (circa 10 Hz).

3.2.3 Anello di sincronizzazione (PLL)

Come spiegato precedentemente, il PLL ha lo scopo di sincronizza il riferimento *dq* del controllore con quello del sistema. Di conseguenza, la componente *q* della tensione del PCC viene mantenuta a zero. Dalla struttura del PLL si deduce che:

$$\frac{d\theta_c}{dt} = \omega_{g0} + \Delta\omega = \omega_{g0} + PI_{PLL}(s)\mathfrak{Im}\left\{\underline{E^c}\right\} \tag{3.29}$$

dove ω_{g0} è la costante aggiunta in uscita al PI, pari a $2\pi\cdot 50$ o 60 Hz. Dalla (3.1) si deriva la seguente espressione:

$$\underline{E_c} \approx (1 - j\Delta\theta)(E_0 + \underline{\tilde{E}}) \approx E_0 + \underline{\tilde{E}} - jE_0\Delta\theta \Rightarrow \mathfrak{Im}\left\{\underline{E^c}\right\} = \mathfrak{Im}\left\{\underline{\tilde{E}}\right\} - E_0\Delta\theta \tag{3.30}$$

Dato che $\Delta\theta = \theta_c - \theta_g$, si può scrivere:

$$\frac{d\Delta\theta}{dt} = \omega_{g0} - \omega_g + PI_{PLL}(s)(\mathfrak{Im}\left\{\underline{\tilde{E}}\right\} - E_0\Delta\theta) \tag{3.31}$$

e la funzione di trasferimento risulta:

$$\Delta\theta = \underbrace{\frac{PI_{PLL}(s)}{s + E_0 PI_{PLL}(s)}}_{G_{PLL}(s)} \mathfrak{Im}\left\{\underline{\tilde{E}}\right\} \tag{3.32}$$

A regime, $\mathfrak{Im}\left\{\underline{\tilde{E}}\right\} = 0$, perciò un regolatore proporzionale è sufficiente a mantenere $\Delta\theta$ a zero. È tuttavia consigliabile l'aggiunta di un termine integrale per rimuovere l'errore a regime quando ω_c differisce da ω_g. La larghezza di banda α_p deve essere relativamente piccola, in modo da reiettare eventuali armoniche di tensione nel PCC. Valori consigliati sono $\alpha_p \leq 0.1\alpha_c$ [2]. La relazione tra il riferimento del convertitore e della rete è dato dalle seguenti:

$$\underline{\tilde{E}^c} = \underline{\tilde{E}} - jE_0\Delta\theta = \underline{\tilde{E}} - jE_0 G_{PLL}(s)\mathfrak{Im}\left\{\underline{\tilde{E}}\right\} \tag{3.33}$$

$$\underline{\tilde{i}} = \underline{\tilde{i}^c} + jE_0\Delta\theta = \underline{\tilde{i}^c} + jE_0 G_{PLL}(s)\mathfrak{Im}\left\{\underline{\tilde{E}}\right\} \tag{3.34}$$

che in forma matriciale diventano:

$$\underline{\tilde{E}^c} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 - E_0 G_{PLL}(s) \end{bmatrix}}_{\boldsymbol{G_{PLL}^s}(s)} \tilde{E} \tag{3.35}$$

$$\tilde{i} = \tilde{i}^c + \underbrace{\begin{bmatrix} 0 & \frac{Q_0}{E_0} G_{PLL}(s) \\ 0 & \frac{P_0}{E_0} G_{PLL}(s) \end{bmatrix}}_{\boldsymbol{G_{PLL}^p}(s)} \tilde{E} \tag{3.36}$$

3.2.4 Valore finale dell'ammettenza

Linearizzando le equazioni del sistema, è possibile dedurre il legame tra la tensione $\tilde{E}$ e la corrente $\tilde{i}$ corrispondente allo schema di fig. 3.6. Il valore finale dell'ammettenza si può calcolare come:

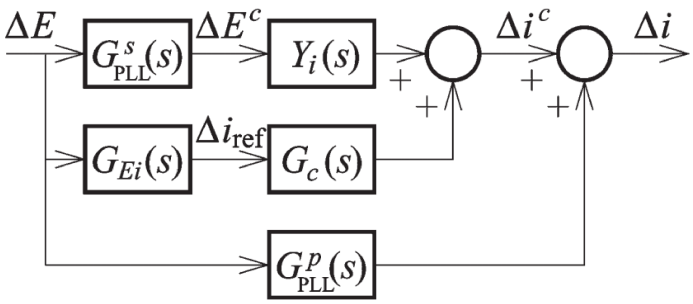
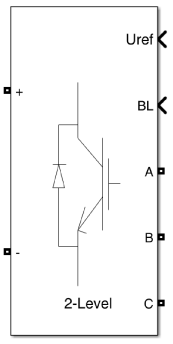


Figura 3.6. Schema a blocchi dell'ammettenza di un inverter GFL linearizzato [2].

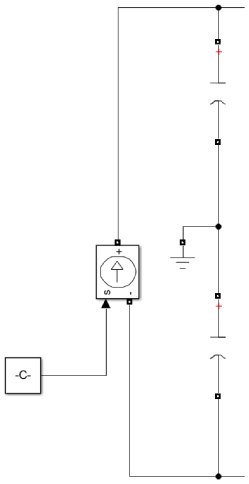
$$\boldsymbol{Y}(s) = \boldsymbol{Y_i}(s)\boldsymbol{G^s_{PLL}}(s) + \boldsymbol{G_c}(s)\boldsymbol{G_{Ei}}(s) + \boldsymbol{G^p_{PLL}}(s) \tag{3.37}$$

3.3 Comparazione tra modello analitico e simulazione

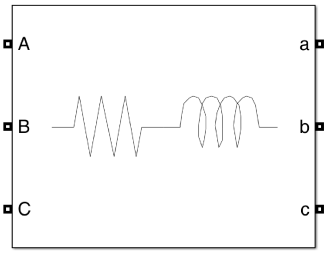
Il modello analitico ricavato nella sezione precedente è stato poi comparato con un modello simulativo in Simulink. Per poter analizzare il funzionamento anche in presenza di potenza reattiva, il controllo AVC è stato trascurato ed è stato fornito un riferimento i_q^{ref} direttamente. Di seguito vengono elencati i principali componenti utilizzati nel modello Simulink.



(a) Blocco Simulink dell'inverter a 2 livelli.



(b) Sorgente di tensione DC dell'inverter.



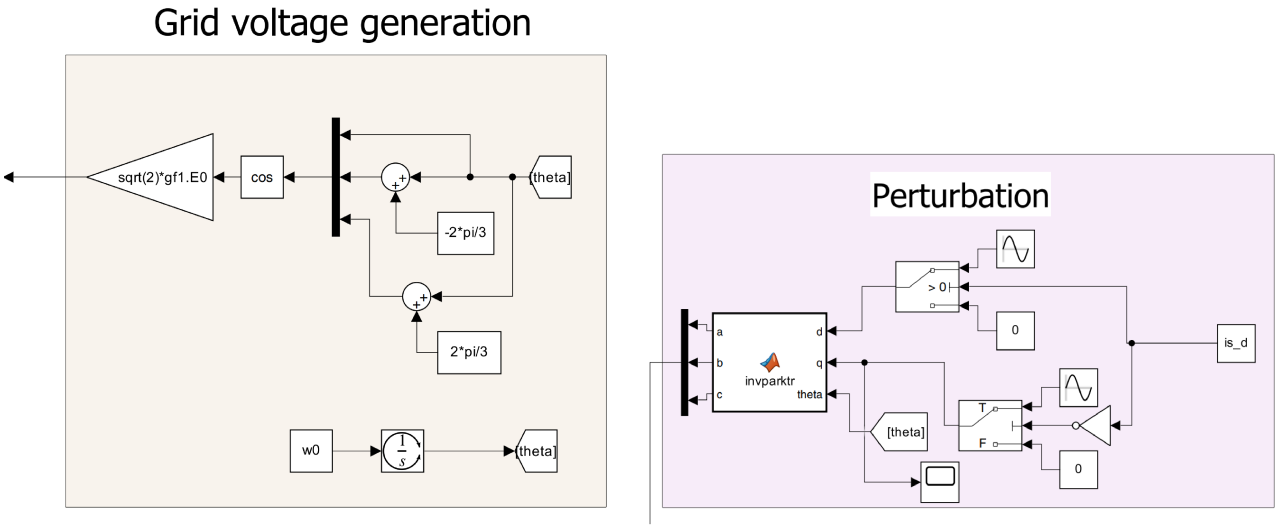
(c) Filtro del convertitore.

Figura 3.7. Blocchi principali del convertitore.

1. Inverter:
   In fig. 3.7a è mostrato il modello Simulink dell'inverter usato. Il blocco in questione è un modello medio controllato con la tensione di riferimento. Questa approssimazione è accettabile perché l'analisi riguarda l'interazione tra l'inverter e la rete e non a instabilità interne del convertitore.
2. DC bus:
   La sorgente di tensione DC del convertitore è modellizzata con due condensatori alimentati da un generatore

di corrente costante. La corrente generata è presa con il segno negativo per rispettare la convenzione dello schema 3.1 e il suo valore dipende dalla potenza attiva che si vuole iniettare in rete. Il valore di C_{dc} è pari a $200/\omega_0$ p.u., dove ω_0 è la pulsazione della rete, in questo caso $2\pi \cdot 50$ rad/s. Con i valori scelti, C_{dc} risulta pari a 0.636 p.u. La tensione del DC-link è impostata a 3 p.u., nel caso in cui 1 p.u. corrisponda alla tensione RMS nel PCC. Il valore minimo risulta comunque $2\sqrt{2} \cdot E_0$.

3. Filtro del convertitore:
   Il filtro usato è composto da un induttore e una resistenza in serie. I valori scelti sono R = 0.01 p.u. e L = $0.15/\omega_0$ p.u.
4. Rete:
   La rete è stata creata con tre generatori di tensione sfasati di 120° l'uno dall'altro e di ampiezza $\sqrt{2} \cdot E_0$. La perturbazione della tensione viene realizzata imponendo delle tensioni sinusoidali nel riferimento *dq* e successivamente trasformate in una terna di sinusoidi in *abc*. L'ampiezza della perturbazione è non più grande di 0.1 V. Lo schema Simulink utilizzato è mostrato in fig. 3.8. La variabile *is_d* serve per selezionare l'asse *d* o *q*.



(a) Schema Simulink della rete.

(b) Schema Simulink della perturbazione di tensione.

Figura 3.8. Blocchi per la generazione e perturbazione della tensione di rete.

5. Controllo di tensione DC (DVC):
   Il controllo DVC si basa sull'equazione (3.8). Il PI è stato realizzato con la funzione *designPI* di Matlab a seconda della larghezza di banda desiderata e la funzione ad anello aperto considerata è $1/sC_{dc}$. In fig. 3.9 è riportato lo schema Simulink del controllo DVC. I livelli di saturazione del riferimento di corrente i_d^{ref} sono stati impostati a ±10 p.u.

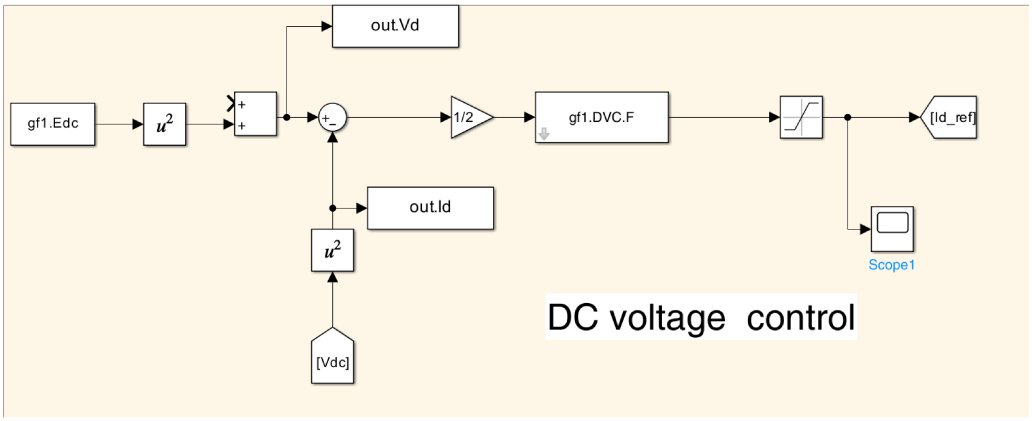


Figura 3.9. Schema Simulink del controllo DVC.

6. Controllo di corrente:
   Il controllo di corrente riceve i riferimenti dal DVC e dal AVC, che in questo caso è stato bypassato. Il riferimento i_q^{ref} viene scelto in modo da iniettare in rete la potenza reattiva scelta, pari a $i_q^{ref} = -Q/\sqrt{3}E_0$. La progettazione del PI si basa sulla (3.4) con la matrice di disaccoppiamento pari a:

$$D = \begin{bmatrix} 0 & \omega_g L \\ -\omega_g L & 0 \end{bmatrix} \tag{3.38}$$

   Lo schema Simulink è riportato in fig. 3.10.

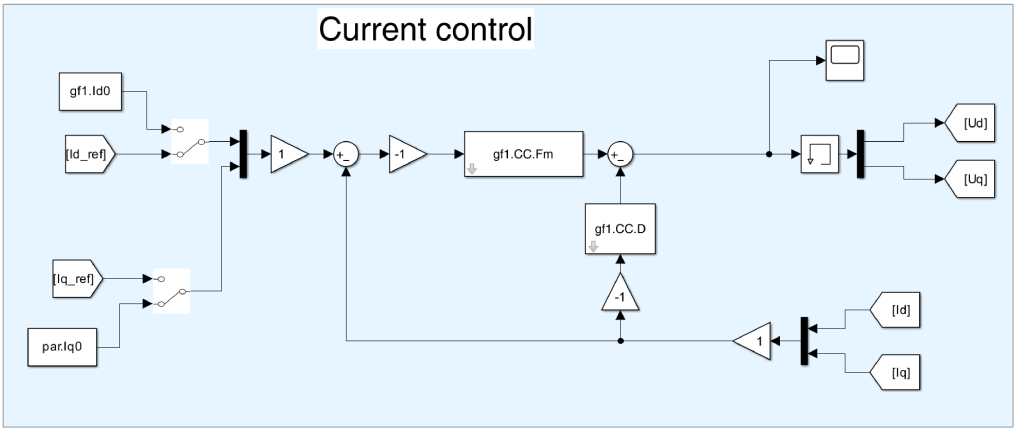


Figura 3.10. Schema Simulink del controllo di corrente.

7. PLL:
   Il PI del PLL è stato progettato anch'esso con la funzione *designPI* di Matlab considerando la (3.29). In fig. 3.11 è riportato lo schema Simulink.

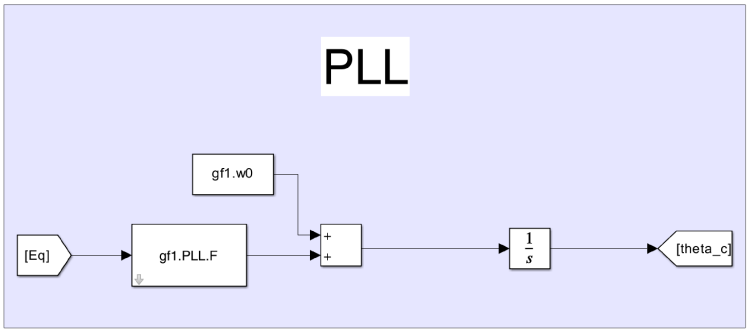


Figura 3.11. Schema Simulink del PLL.

Come detto nel capitolo 2, esistono due modi possibili di misurare un'impedenza o, in questo caso, una ammettenza: perturbare la corrente o la tensione tra due sottosistemi. In questo caso si è scelto di perturbare la tensione di rete, come mostrato in fig. 3.8b. La perturbazione viene effettuata prima sull'asse d e successivamente sull'asse q a frequenze diverse. Il range di frequenze è selezionabile all'inizio delle simulazioni. Una scelta valida è partire da una frequenza bassa, ad esempio 1 Hz, e salire fino a oltre la banda passante del controllo di corrente, in modo da verificare la presenza o meno di risonanze o malfunzionamenti del controllo. La misura consiste nell'analizzare la risposta della tensione e corrente nel sistema dq, togliendo la componente continua dalle misure, ed effettuare una analisi in frequenza. A questo scopo esistono diverse soluzioni, tra cui la Fast Fourier Transform (FFT) o la Cross Power Spectral Density (CPSD). Entrambe restituiscono la risposta in frequenza cercata: $Y(j\omega) = \frac{I(j\omega)}{V(j\omega)}$, tuttavia la CSPD è consigliata nel caso di misure affette da rumore ed è stata usata questa in Simulink.

In generale, ogni misura è affetta da rumore, il che rende problematico calcolare direttamente il rapporto tra due risposte in frequenza come $H(j\omega) = Y(j\omega)/X(j\omega)$, in cui l'uscita del sistema in esame $Y(j\omega)$ e l'ingresso $X(j\omega)$

hanno rumore sovrapposto. Una soluzione per mitigare l'effetto del rumore è l'utilizzo di stimatori. Un esempio è la CPSD:

$$\tilde{H}(j\omega) = \frac{P_{yx}(j\omega)}{P_x(j\omega)} \tag{3.39}$$

dove $P_{yx}(j\omega)$ è la densità spettrale di potenza incrociata di X e Y e $P_x(j\omega)$ la densità di potenza di X. Si supponga di avere una misura Y affetta da rumore N_0. La stima di H usando il rapporto dei due segnali risulta:

$$\tilde{H}(j\omega) = \frac{Y(j\omega) + N_0}{X(j\omega)} = H + \frac{N_0}{X} = H + \frac{1}{SNR} \tag{3.40}$$

dove SNR è il rapporto segnale-rumore. Quindi la stima della risposta in frequenza è tanto più errata quanto più basso è il SNR. La densità spettrale di potenza riduce tale problema. Dalla definizione di correlazione incrociata si ha:

$$\begin{aligned} P_{yx}(j\omega) =& \mathbb{E}[\overline{X(j\omega)} \cdot (Y(j\omega) + N_0)] \\ =& \mathbb{E}[\overline{X(j\omega)} \cdot Y(j\omega) + \overline{X(j\omega)} \cdot N_0] \\ =& \mathbb{E}[\overline{X(j\omega)} \cdot H(j\omega)X(j\omega) + \overline{X(j\omega)} \cdot N_0] \\ =& H(j\omega) \cdot \mathbb{E}[\overline{X(j\omega)}X(j\omega)] + \mathbb{E}[\overline{X(j\omega)}N_0] \\ =& H(j\omega)P_x(j\omega) + \mathbb{E}[\overline{X(j\omega)} \cdot N_0] \end{aligned} \tag{3.41}$$

Dove $\overline{X(j\omega)}$ è il complesso coniugato di $X(j\omega)$. Supponendo che il rumore sia scorrelato dal segnale X, si ha:

$$\mathbb{E}[\overline{X(j\omega)} \cdot N_0] = 0 \tag{3.42}$$

La CPSD di due segnali è proporzionale alla correlazione incrociata, quindi lo stimatore diventa:

$$\tilde{H}(j\omega) = \frac{P_{yx}(j\omega)}{P_x(j\omega)} = H(j\omega) \tag{3.43}$$

cioè si ha uno stimatore perfetto. Nella pratica, tuttavia, il termine $\mathbb{E}[\overline{X(j\omega)} \cdot N_0]$ non arriva mai a zero ma raggiunge valori molto bassi, fornendo una buona stima di $H(j\omega)$.

In Matlab, una volta misurati i segnali di tensione e corrente per ogni asse, si è calcolata la CPSD e fatto il rapporto per ottenere la risposta in frequenza. Nelle simulazioni si è fatta variare sia la potenza attiva, sia quella reattiva da 0 a 1 p.u. In fig. 3.12 sono mostrate le simulazioni variando la potenza attiva e mantenendo la potenza reattiva a zero. Le componenti Y_{dq} e Y_{qd} sono entrambe nulle in corrispondenza di potenza reattiva nulla, infatti le simulazioni riportano valori estremamente bassi. In questi casi, l'errore della fase non è rilevante dal momento che si tratta di valori assoluti così bassi. Le altre componenti corrispondono perfettamente all'andamento atteso. Nel caso in cui la potenza reattiva non sia nulla, allora la componente Y_{qd} non è più nulla, come mostrato in fig. 3.13. Anche in questo caso la simulazione corrisponde ai valori analitici.

3.4 Creazione del dataset

Confermato il fatto che il modello analitico e simulativo corrispondono, è possibile creare un dataset per l'addestramento della rete neurale a partire dal modello Simulink creato. Per semplicità e risparmio di tempo, si è deciso di utilizzare i valori analitici per creare il dataset, anche se si è dimostarto che il modello Simulink è fedele ai risultati derivati in [2]. Per la realizzazione del dataset si sono scelti 50 punti in frequenza e 8 punti operativi in potenza, così definiti:

$$\begin{aligned} P &= [0.1 \;\; 0.25 \;\; 0.4 \;\; 0.5 \;\; 0.6 \;\; 0.75 \;\; 0.9 \;\; 1] \;\; p.u. \\ Q &= [0 \;\; 0.2 \;\; 0.4 \;\; 0.5 \;\; 0.6 \;\; 0.75 \;\; 0.9 \;\; 1] \;\; p.u. \end{aligned}$$

Il vettore delle frequenze è definito con la funzione *logspace* di Matlab: $f = logspace(0, log10(3e3), 50)$. In questo modo si ottiene un vettore di 50 punti equispaziati in una scala logaritmica da 1 Hz a 3 kHz. Il numero di punti

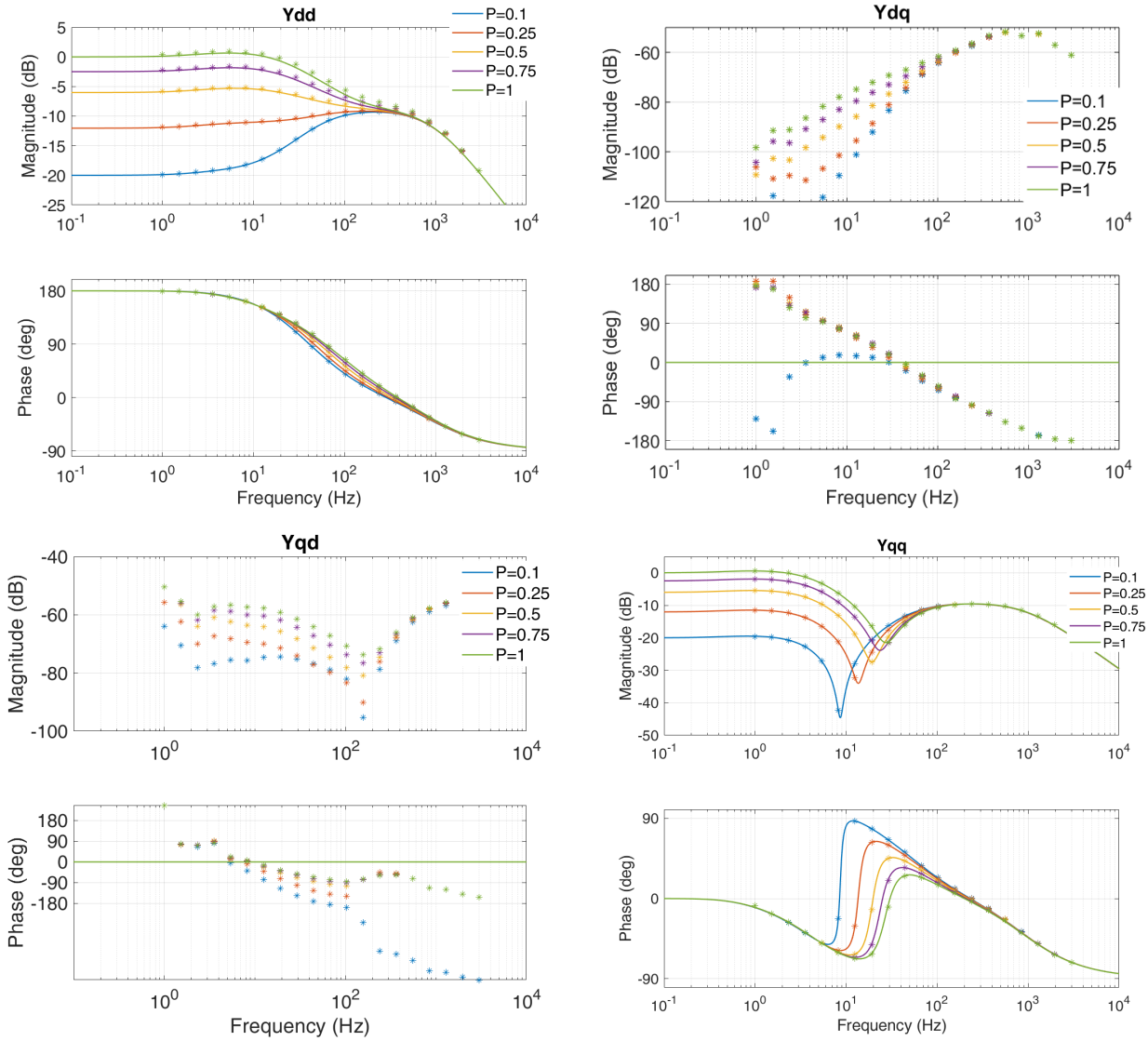


Figura 3.12. Simulazione dell'ammettenza: confronto tra modello analitico (linea continua) e simulato (∗).

Convertitore 1		**Convertitore 2**		**Convertitore 3**		**Convertitore 4**	
P_n	300 kW	P_n	500 kW	P_n	100 kW	P_n	800 kW
BW CC	1 kHz	*BW* CC	750 Hz	*BW* CC	2 kHz	*BW* CC	600 Hz
BW DVC	25 Hz	*BW* DVC	15 Hz	*BW* DVC	25 H	*BW* DVC	15 Hz
BW PLL	5 Hz	*BW* PLL	5 Hz	*BW* PLL	10 Hz	*BW* PLL	2 Hz

Tabella 3.1. Caratteristiche dei convertitori.

del dataset è quindi 3200 (8×8×50). Siccome l'intento finale è quello di testare l'efficacia del transfer learning, si sono realizzati diversi dataset per rappresentare diversi convertitori con caratteristiche differenti. In totale si sono realizzati 4 convertitori, con relativi dataset, definiti da diverse bande passanti degli anelli di controllo e diverse potenze nominali, le cui caratteristiche sono riportate nella tabella 3.1.

I valori sono stati scelti considerando il fatto che, solitamente, più alta è la potenza nominale, più sono ridotte le bande passanti degli anelli di controllo. Nel contesto del transfer learning, come detto nel capitolo 1, occorre disporre di un grande dataset comprensivo di più convertitori per poi utilizzare un numero ridotto di dati da un nuovo dataset per specializzare il modello. Per fare ciò, i primi tre convertitori sono stati usati per il preaddestramento delle reti neurali e il quarto per fornire dati specifici per l'addestramento di fino.

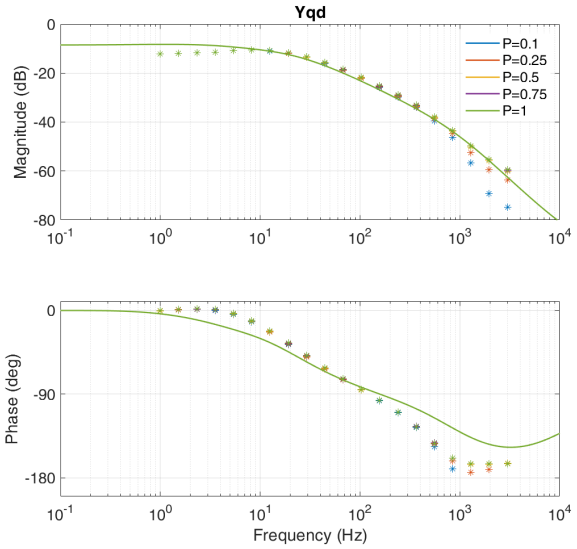


Figura 3.13. Simulazione dell'ammettenza Y_{qd} con Q=0.1 p.u..

Capitolo 4

Addestramento delle reti neurali e Transfer Learning

Nel capitolo 1 si sono viste le caratteristiche principali di una rete neurale, le potenzialità e i possibili problemi che si possono verificare durante l'addestramento. In questo capitolo si farà una breve introduzione alle librerie usate per la creazione delle reti neurali e il loro addestramento. Si analizzerà la struttura delle singole reti (numero di neuroni, layer, funzione di attivazione...) per capire quale sia la configurazione più adatta nei vari casi. Il lavoro fatto in [3] si nota che è stata utilizzata una rete neurale per la parte reale immaginaria di ogni componente dell'ammettenza, quindi 8 reti totali. In questa tesi si è scelto di utilizzare una rete per stimare sia la parte reale che immaginaria di ogni componente. Questo, comprensibilmente, aumenta la complessità del problema e di conseguenza della rete. In linea teorica si potrebbe utilizzare un'unica rete per stimare l'intera matrice dell'ammettenza, ma probabilmente porterebbe ad avere un modello estremamente complesspiù complesso e meno accurato. Per tali motivi si è scelto di utilizzare 4 reti totali. accertato il corretto funzionamento delle reti, si vedrà l'applicazione del transfer learning. A tale scopo si sono confrontati due metodi: 1) la rimozione di alcuni layer dal modello preaddestrato e l'aggiunta di nuovi, addestrando solo questi ultimi; 2) il riaddestramento completo del modello preaddestrato modificando il tasso di apprendimento. Entrambi i metodi fanno uso di un numero limitato di dati presi casualmente dal dataset del quarto convertitore. Gli altri tre sono stati utilizzati per ottenere i modelli preaddestrati dai quali partire.

4.1 Libreria Tensorflow

Tensorflow [1] è una delle librerie più usate per lo sviluppo di reti neurali, srilasciata da Google Brain nel 2015. È opensource ed è stata progettata con diverse risorse integrate che facilitano lo sviluppo di modelli di IA, come la libreria integrata Keras [2] che ornisce una API ad alto livello per la creazione di reti neurali. Tensorflow offre, inoltre, il supporto per l'accelerazione hardware tramite GPU e TPU che consente di eseguire addestramenti su grandi dataset in modo significativamente più veloce rispetto all'utilizzo della sola CPU.

Di seguito si vedranno in dettaglio le singole reti e i risultati ottenuti durante l'addestramento. Si vedranno da prima delle configurazioni scelte manualmente per poi confrontarle con delle architetture ottimizzate dal tuner integrato nella libreria keras. Questo metodo permette di esplorare in maniera semplice e automatizzata numerose scelte degli iperparametri e di trovare quale sia l'architettura più adatta. Il dataset preso come riferimento è quello del convertitore 1 e in tutti i casi si è suddiviso il dataset in 80% addestramento, 10% validazione e 10% test. I dati sono stati poi normalizzati tra 0 e 1.

Nello sviluppo di reti neurali non ci sono delle regole nette su come scegliere gli iperparametri e occorre provare varie soluzioni. Solitamente, un buon punto di partenza è usare 3 layer con un numero modesto di neuroni, a seconda della complessità del problema. Se l'errore risulta alto con questa configurazione, allora è probabile che ci siano pochi neuroni e si aumentano il numero di layer e/o neuroni. Una prima scelta della funzione di attivazione è la *relu*, in quanto fornisce buoni risultati nella maggior parte dei casi. L'ottimizzatore usato è Adam

1 https://www.tensorflow.org/?hl=it
2 ://keras.io/

con le impostazioni predefinite di learning rate e coefficienti dei momenti. Essenso un problema di regressione, la funzione di costo scelta è l'errore quadratico medio sui dati di validazione(RMSE).

4.2 Architettura di Y_{dd}

Tenendo a mente le considerazioni generali appena menzionate, la configurazione iniziale proposta per Y_{dd} è una rete composta da 3 layer, 26 neuroni e 218 parametri. Da tenere a mente che l'ultimo layer ha il numero di neuroni fissato a 2 dal momento che la rete deve stimare due valori numerici, corrispondenti alla parte reale e immaginaria di Y_{dd}. La funzione di attivazione scelta è stata la *relu*. Nell'ultimo layer, quello di uscita, si è scelto di usare la funzione *linear*. Per l'addestramento si è scelto un batch size di 64. Valori tipici sono 32, 64, 128 o anche 1024, a seconda della dimensione del dataset di addestramento. Il numero di epoche massimo è di 2000 epoche con l'aggiunta di una condizione di stop in base al valore dell'RMSE di validazione: se questo rimaneva inferiore alla soglia di 0.0005 per almeno 100 epoche, allora l'addestramento si fermava. Questa condizione permette di avere addestramenti più rapidi ma anche di evitare overfitting.

In fig. 4.1 si vede come l'errore di addestramento e validazione siano molto simili e che calano rapidamente, segno che la rete si sta addestrando bene e non c'è over o underfitting. L'addestramento si può dire ben riuscito ed è il momento di verificare il modello con dati di test che, si ricorda, sono valori nuovi che la rete non ha mai visto e che rappresentano dati in un contesto reale.

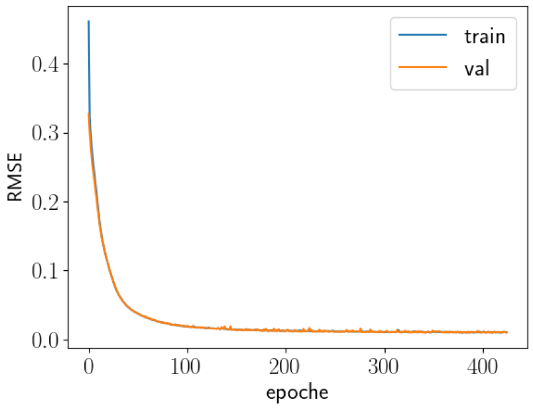


Figura 4.1. Errore di addestramento e validazione durante l'addestramento.

In fig. 4.2 si vede quanto si discosta la previsione del modello dai dati di test. Sull'asse delle ascisse sono riportati i dati di test e sull'asse delle ordinate le previsioni del modello. I punti rossi corrispondono ai dati di test, quindi quelli veri, e giacciono sulla bisettrice del piano. Mentre i punti blu sono le previsioni del modello e, idealmente, dovrebbero essere uguali ai dati di test, quindi sovrapposti. Visivamente sono risultati piuttosto buoni già con una rete così semplice.

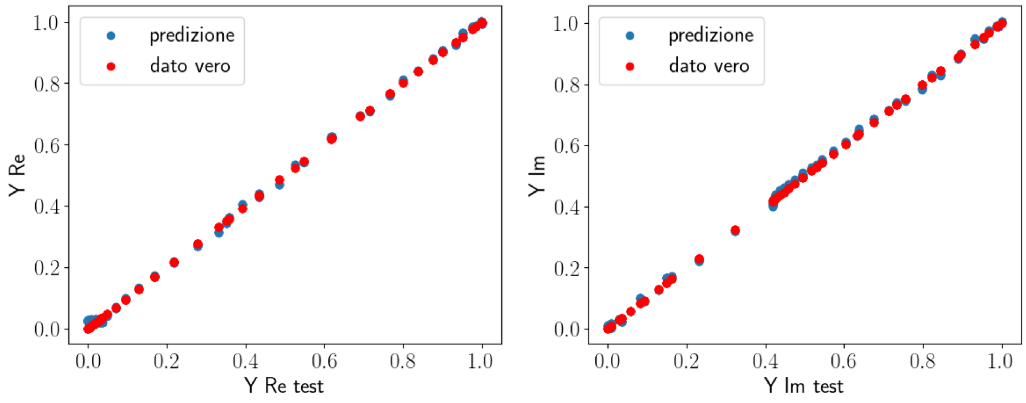


Figura 4.2. Previsione sui dati di test del modello più semplice.

L'errore ottenuto alla fine sui dati di test è riportato nella tabella 4.1. Per il modulo si è scelto di calcolare lo scarto quadratico medio (RMSE), invece, per la fase si è preferito usare un'altra metrica: l'errore mediano assoluto. Questo tipo di errore è meno sensibile ai punti anomali lontani dala valore atteso rispetto all'RMSE, e dà quindi un miglior indice di accuratezza. L'esistenza di questi punti anomali è dovuta al fatto che nel calcolo della fase vicino a 180° accade spesso che ci siano cambi di segno della fase dati dalla funzione *arctan2*, se pur il valore assoluto è molto vicino a quello vero.

	Errore	**min**	**max**
Modulo	0.003 S (RMSE)	0.1 S	0.342 S
Fase	0.93° (MdAE)	-71.13°	179.92°

Tabella 4.1. Errore sui dati di test con il modello più semplice.

Già con una rete relativamente semplice (218 parametri) è stato possibile ottenere un errore relativo del 3% sul modulo rispetto al valore minimo. Si sono provate architetture più complesse, ad esempio con 5 layer, 54 neuroni e 678 parametri e usando *tanh* come funzione di attivazione, ottenendo errori ancora più bassi. In fig. 4.3 è mostrato il confronto tra le previsioni del modello e i dati di test. Visibilmente, ora, il modello è in grado di stimare meglio i

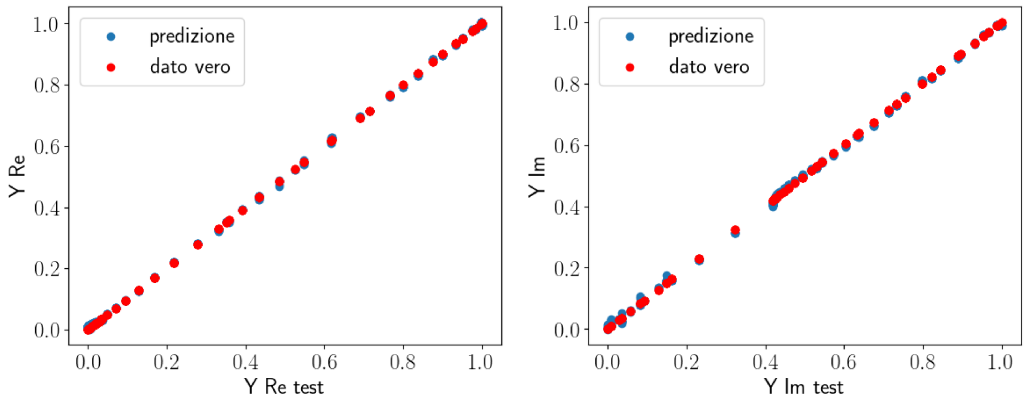


Figura 4.3. Previsione sui dati di test con modello più complesso.

	Errore	**min**	**max**
Modulo	0.001 S (RMSE)	0.1 S	0.342 S
Fase	0.458° (MdAE)	-71.13°	179.92°

Tabella 4.2. Errore sui dati di test con il modello più semplice.

valori attesi, come mostrato nella tabella 4.2 con un errore circa 3 volte più piccolo. Le prove fatte dimostrano che già una rete composta da 3 layer e 24 neuroni è in grado di stimare l'ammettenza con errori di circa il 3%. Reti più complesse offrono errori ancora minori ma bisogna tenere a mente quanto complesso si vuole realizzare il modello e qual è l'errore accettabile. È alquanto inutile utilizzare modelli complessi quando si hanno già ottimi risultati con molti meno neuroni. Per visualizzare ancora meglio le performance del modello ottenuto, si può graficare il diagramma di Bode di un punto operativo, ad esempio (P,Q) = (0.25,0.25) p.u. In fig. 4.4 e 4.5 sono mostrati i diagrammi di Bode del modello più semplice e quello più complesso, rispettivamente. Come si vede, il secondo modello è più fedele al grafico atteso.

4.2.1 Ottimizzazione degli iperparametri

È chiaro che partire da configurazioni relativamente semplici è una buona soluzione di partenza. Eventualmente si può aumentare la complessità nel caso l'errore sia eccessivo, ma i gradi di libertà offerti sono molti e non è

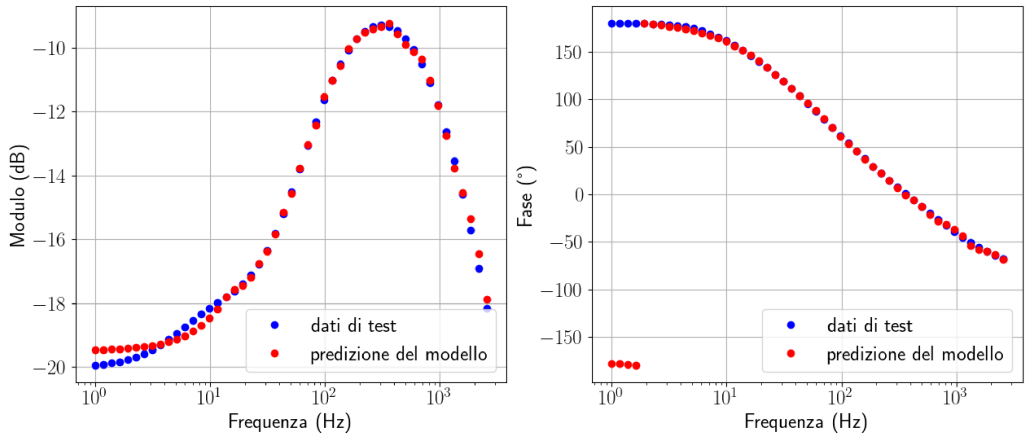


Figura 4.4. Diagramma di Bode del modello a 218 parametri.

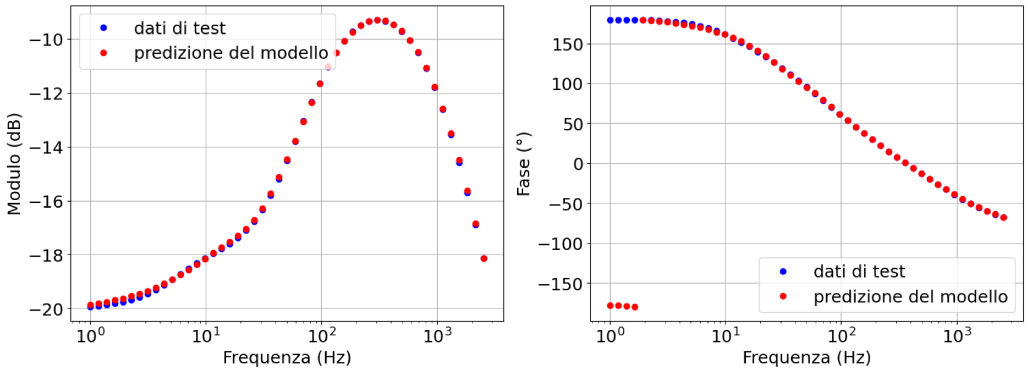


Figura 4.5. Diagramma di Bode del modello a 678 parametri.

semplice trovare la combinazione migliore, anche perché alcuni iperparametri potrebbero contrastarsi a vicenda e vanificare il miglioramento offerto da altri. Analizzare ogni combinazione manualmente richiederebbe molto tempo e molte soluzioni non sarebbero ottime. Per esplorare una grande vastità di scelta in maniera efficiente tornano molto utili i tuner offerti dalle librerie di machine learning. Keras offre una discreta scelta su come ottimizzare gli iperparametri di una rete neurale. Esistono delle classi predefinite di ricerca dei parametri ottimali: RandomSearch, BayesianOptimization e Hyperband. L'algoritmo scelto è il BayesianOptimization perché offre un buon compromesso tra complessità computazionale ed efficienza di ricerca. I parametri che sono stati fatti variare sono:

- **Numero di layer interni**: oltre al layer di ingresso e quello di uscita, si è fatto variare il numero di layer interni della rete tra 1 e 3, per un totale di 5 layer per non ottenere architetture troppo complesse.
- **Numero di neuroni**: per ogni layer si è fatto variare il numero di neuroni da 2 a 30, eccetto il layer di uscita.
- **Funzione di attivazione**: le funzioni di attivazioni testate sono state la relu, tanh e sigmoid, che sono tra le più usate.
- **Learning rate**: essendo un parametro che influenza molto l'addestramento si è fatto variare da 0.0005 a 0.005. Si ricorda che il valore di default è 0.001.
- **Batch size**: anche il valore del batch size può influire sulla convergenza perciò si è voluto analizzare come cambiava la rete al variare di esso. I valori di prova sono stati 32, 64, 128 e 200.

I passaggi da fare per ottimizzare gli iperparametri sono spiegati nelle guide di keras (`https://keras.io/guides/keras_tuner/`), ma in breve occorre:

1. Definire il modello e lo spazio di ricerca degli iperparametri.
2. Definire il tuner, ad esempio il BayesianOptimization, impostando la funzione obiettivo (RMSE di validazione) e il numero di prove da eseguire, in questo caso 10.

3. Eseguire la ricerca degli iperparametri ottimali.

Una volta terminata la ricerca, la funzione di ricerca restituisce i modelli con gli iperparametri migliori tra quelli proposti. Non solo, un altro vantaggio di usare il tuner è quello di poter ottenere un modello con i pesi migliori in assoluto, rispetto a qualsiasi epoca. A differenza dell'addestramento tradizionale con la funzione *fit()*, si ottiene il modello con gli iperparametri dell'ultima epoca di addestramento, che potrebbero non essere quelli ottimali, soprattutto se l'errore oscilla molto.

Nel caso di Y_{dd}, il miglior modello ottenuto è composto da 5 layer, 91 neuroni e 1659 parametri. Il learning rate e batch size ottimali sono stati 0.003 e 32 rispettivamente. La funzione di attivazione migliore è risultata essere la tanh. L'errore medio ottenuto è riportato nella tabella 4.3. Come si vede, l'errore è diminuito circa di un fattore

	Errore	**min**	**max**
Modulo	0.56 mS (RMSE)	0.1 S	0.342 S
Fase	0.112° (MdAE)	-71.13°	179.92°

Tabella 4.3. Errore sui dati di test con il modello migliore.

5 rispetto al modello a 218 parametri. Come c'era da aspettarsi, per ottenere risultati migliori, è stato necessario aumentare di molto il numero di neuroni. Tuttavia, c'è da tenere conto della dimensione dello spazio di ricerca e dei tentavi fatti: se il numero di combinazioni è molto alto e i tentativi permessi sono pochi c'è il rischio di trovare dei modelli performanti ma più complessi del necessario. Infatti, il secondo miglior modello fornito dalla ricerca è composta da 5 layer, 42 neuroni e soli 450 parametri con un errore appena superiore (vedi tabella 4.4). Ciò sta a significare che non è necessario spingersi a configurazioni con tanti neuroni per ottenere errori più piccoli di una configurazione scelta manualmente. L'importante è definire saggiamente lo spazio di ricerca dell'ottimizzatore. Perciò, la soluzione migliore tra errore e complessità della rete è questo secondo caso.

	Errore	**min**	**max**
Modulo	0.82 mS (RMSE)	0.1 S	0.342 S
Fase	0.28° (MdAE)	-71.13°	179.92°

Tabella 4.4. Errore sui dati di test del secondo miglior modello.

Per un confronto grafico si veda la fig. 4.6 e il diagramma di Bode in fig. 4.7. In questo caso le previsioni del modello sono praticamente perfette.

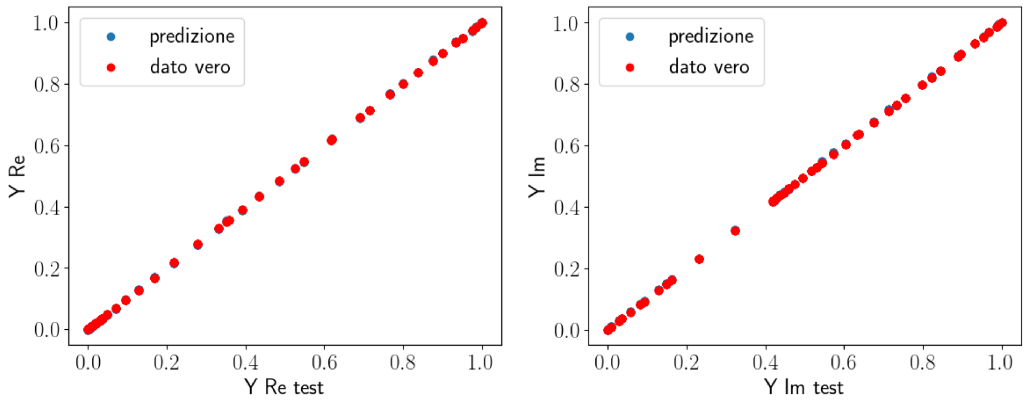


Figura 4.6. Previsione sui dati di test del modello migliore.

4.2.2 Influenza della dimensione del dataset

Come detto più volte, i dati forniti al modello durante l'addestramento sono di fondamentale importanza. Uno degli aspetti più importanti di un dataset è la sua dimensione: più grande è, migliore sarà il modello ottenuto.

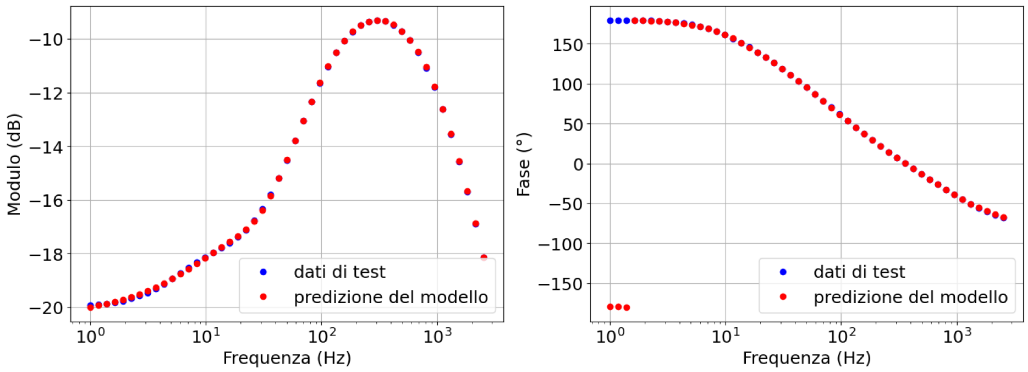
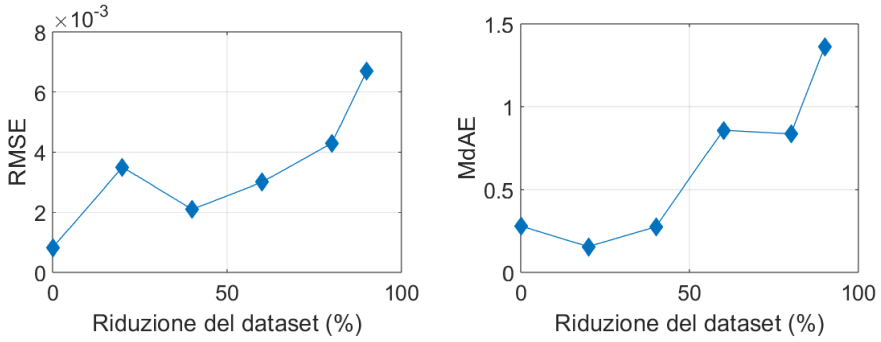


Figura 4.7. Diagramma di Bode del modello migliore.

Tuttavia, non sempre è facile raccogliere sufficienti dati, perciò, in questa prova si è voluto analizzare le prestazioni del modello al variare della dimensione del dataset per verificare se con un dataset più piccolo sia possibile ottenere buoni risultati. Si ricorda che il set di dati originale è composto da 3200 punti di cui l'80% usato per l'addestramento.

Si prenda come riferimento il modello da 678 parametri, si è utilizzato un batch size di 64 per permettere un addestramento con pochi dati. Le suddivisioni tra dati di addestramento, validazione e test sono le stesse. I risultati ottenuti sono riportati nel grafico di fig. 4.8.



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.8. Errore del modulo e fase al variare della dimensione del dataset.

Si nota che l'errore comincia a salire significativamente dopo una riduzione di circa l'80%, che equivale a 640 punti totali. I risultati mostrano che con un dataset di poche centinaia di dati è sufficiente ad avere errori inferiori al 7% rispetto al modulo minimo. La dimensione ideale del dataset dipende dall'errore che si vuole ottenere e dalla disponibilità e facilità di ottenere tali dati.

4.3 Architettura di Y_{dq}

Il caso di Y_{dq} è piuttosto semplice visto che il suo valore rimane a zero in ogni condizione. In questo caso non è necessaria una rete neurale per stimare tale componente, e comunque è sufficiente una rete abbastanza semplice per farlo. Una soluzione molto buona trovata è composta da 3 layer, 12 neuroni e 110 parametri. Siccome l'uscita da stimare è sempre zero, si è abbassato il learning rate a 0.0002 in modo da ridurre la modifica dei pesi della rete e non incorrere in oscillazioni dell'errore. Come prevedibile, l'errore è presto converso a valori molto bassi, riportati nella tabella 4.5. L'errore sulla fase è trascurabile dal momento che il modulo è estremamente basso.

In fig. 4.9 è riportato il confronto tra la previsione del modello e il valore atteso.

Essendo questo un caso banale non si è fatta l'ottimizzazione degli iperparametri.

	Errore	min	max
Modulo	$5{\cdot}10^{-6}$ S (RMSE)	0 S	0 S
Fase	9.17° (MdAE)	0°	0°

Tabella 4.5. Errore sui dati di test.

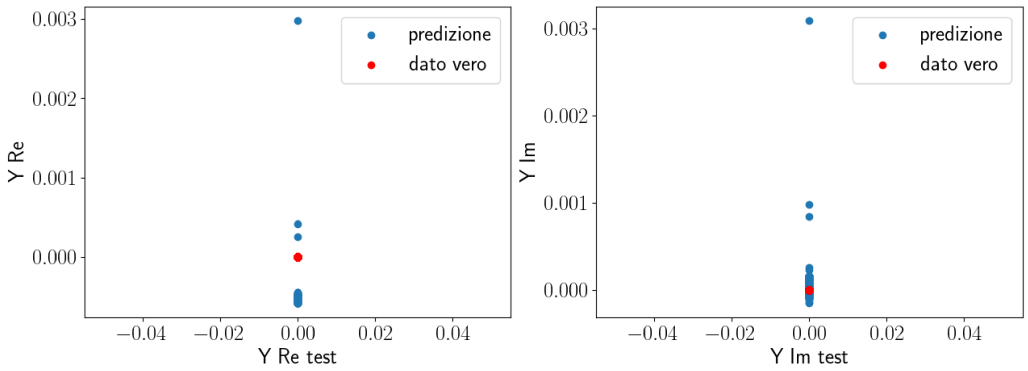


Figura 4.9. Previsione sui dati di test del modello.

4.4 Architettura di Y_{qd}

La componente Y_{qd}, si ricorda, è nulla quando la potenza reattiva è nulla, altrimenti assume valori diversi da zero. Analogamente a quanto fatto per Y_{dd}, si è partiti da una configurazione semplice con pochi neuroni per poi aumentare fino ad ottenere un errore RMSE paragonabile al caso di Y_{dd}. L'architettura risultante è una rete composta da 4 layer, 50 neuroni e 650 parametri. Il learning rate usato è 0.002 e un batch size di 128, la funzione di attivazione, tanh. Le previsioni del modello sui dati di test sono mostrate in fig. 4.10, mentre in fig. 4.11 è mostrato il diagramma di Bode, sempre nel punto operativo (0.25,0.25) p.u.

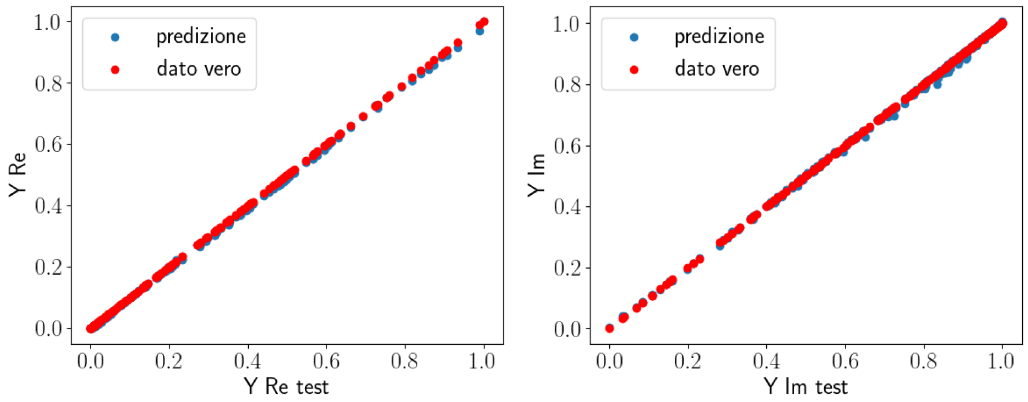


Figura 4.10. Previsione sui dati di test del modello.

L'errore sui dati di test è riportato in tabella 4.6. L'errore della fase è relativamente alto, dovuto probabilmente ad errori numerici nel calcolo della fase quando il numero complesso si trova vicino a 180°. Infatti, l'errore massimo della fase sui dati di test è stato di 177°. In fig. 4.11 si vede questo errore negli ultimi punti.

4.4.1 Ottimizzazione degli iperparametri

Anche in questo caso si è usato l'ottimizzatore di keras per cercare una eventuale configurazione più efficace. Si è provato a verificare se con un massimo di 2 layer interni, quindi 4 layer totali, si riusciva ad ottenere un errore minore. L'ottimizzatore ha trovato una configurazione di 4 layer, 52 neuroni e 497 parametri, contro i 650 di prima,

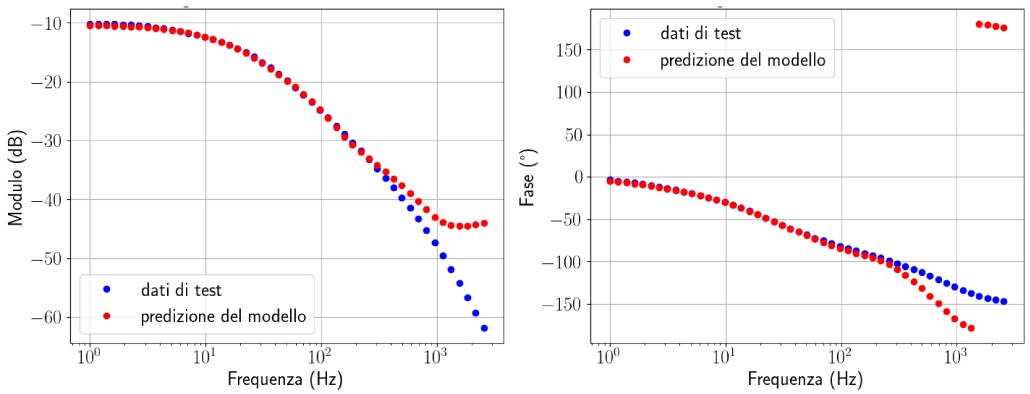


Figura 4.11. Diagramma di Bode.

	Errore	min	max
Modulo	0.0074 S (RMSE)	0 S	1.544 S
Fase	1.04° (MdAE)	-147°	0°

Tabella 4.6. Errore sui dati di test.

che presenta un errore MSRE del modulo di circa la metà. Gli iperparametri ottimali sono risultati essere un learning rate di 0.0037, batch size di 64 e sigmoid come funzione di attivazione. In fig. 4.12 e 4.13 sono mostrate le previsioni del modello e il digramma di Bode del modello ottimo.

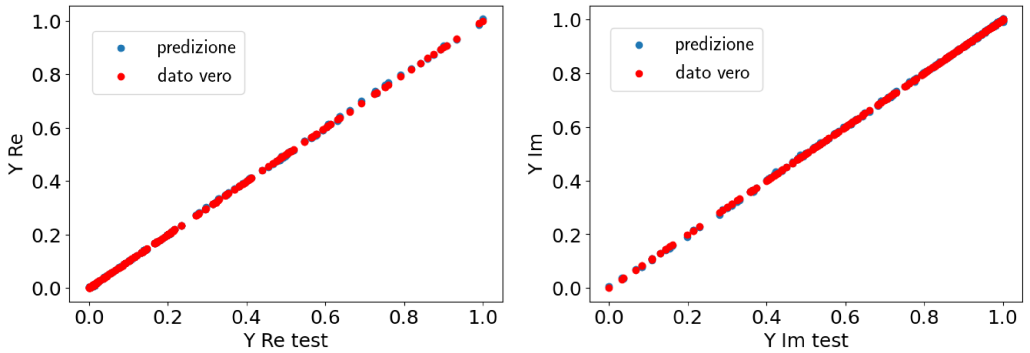


Figura 4.12. Previsione sui dati di test del modello ottimizzato.

Anche qui, in alta frequenza la stima comincia a divergere dai valori attesi, tuttavia la differenza è ridotta rispetto a prima. In tabella 4.7 è riportato l'errore del modello migliore. Sia il modulo che la fase, adesso, hanno un errore minore, soprattutto la fase. Segno che l'ottimizzatore è riuscito a trovare una soluzione ottimale.

È ovviamente possibile ottenere architetture più performanti aumentando il numero di layer e/o neuroni dello spazio di ricerca, ma l'obiettivo qui era di trovare una configurazione con una complessità simile ma più efficace. Perciò si è optato per il modello proposto dall'algoritmo il più semplice possibile.

4.5 Architettura di Y_{qq}

La componente Y_{qq} è la più complessa da approssimare, essendoci un'antirisonanza e un cambio di fase brusco. Quindi ci si aspetta che la rete neurale sia un po' più complessa delle altre. In questo caso può tornare molto utile l'ottimizzatore. Scegliendo una configurazione di 4 layer, 50 neuroni e 650 parametri, tanh come funzione di attivazione e batch size di 128, si è ottenuto un buon modello, come mostrato in fig. 4.14 e 4.15. L'errore misurato è riportato in tabella 4.8.

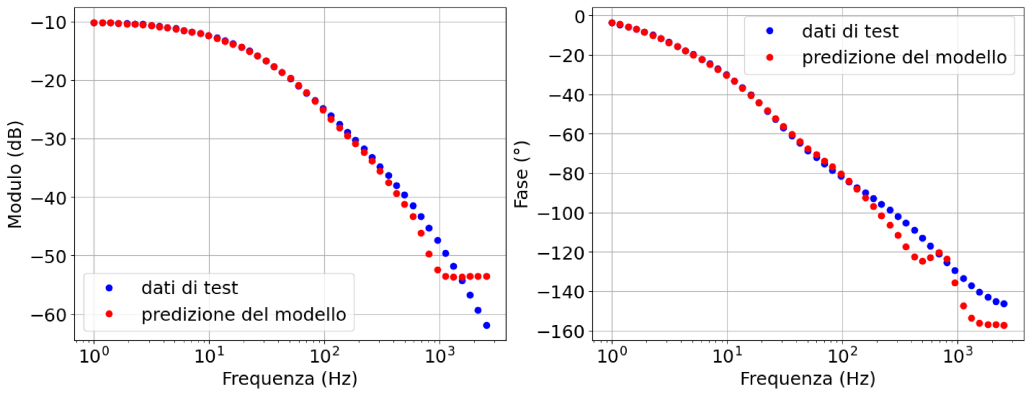


Figura 4.13. Diagramma di Bode del modello ottimizzato.

	Errore	min	max
Modulo	0.0035 S (RMSE)	0 S	1.542 S
Fase	0.28° (MdAE)	-147°	0°

Tabella 4.7. Errore sui dati di test.

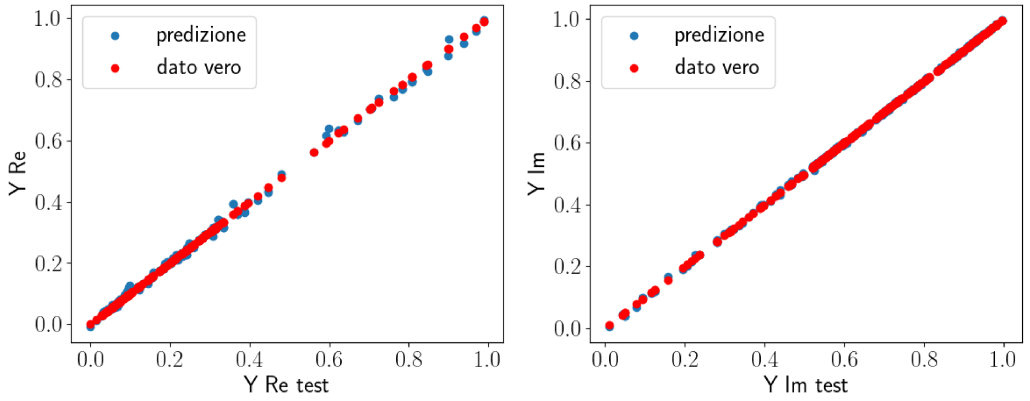


Figura 4.14. Previsione sui dati di test.

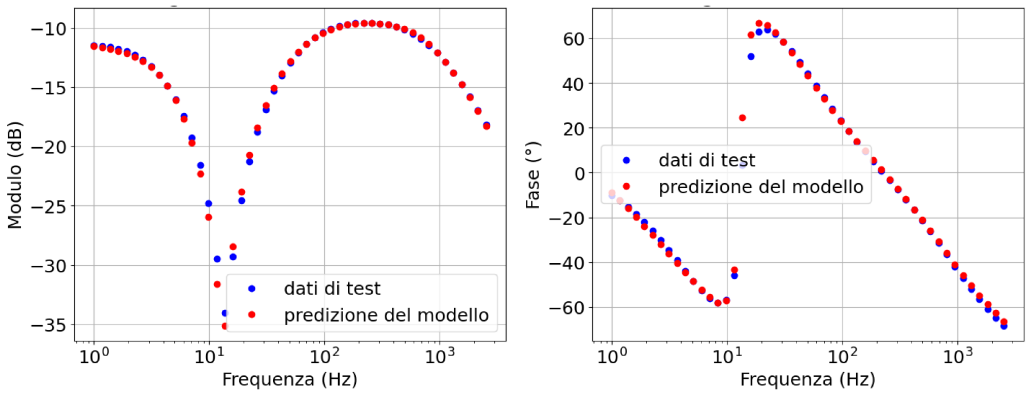


Figura 4.15. Diagramma di Bode.

4.5.1 Ottimizzazione degli iperparametri

Il modello così è già un ottimo modello. L'ottimizzatore può mostrare se esistono delle configurazioni altrettanto efficaci ma più semplici. Si è provato a cercare una soluzione con al massimo 3 layer. Il miglior risultato è stata una

	Errore	min	max
Modulo	0.0078 S (RMSE)	0.032 S	1.066 S
Fase	0.372° (MdAE)	-71.59°	87.05°

Tabella 4.8. Errore sui dati di test.

rete con 3 layer e 300 parametri ma con un errore superiore al modello ottenuto manualmente. Perciò si deduce che per questa funzione sia necessaria una rete con almeno 4 layer per ottenere errori più piccoli. In questo caso si è trovata una soluzione composta da 4 layer, 42 neuroni e 553 parametri, quindi comunque più semplice rispetto all'originale, con learning rate di 0.001 e batch size di 64. I risultati sono mostrati in fig. 4.16 e 4.17. In tabella 4.9 è riportato l'errore del modello migliore.

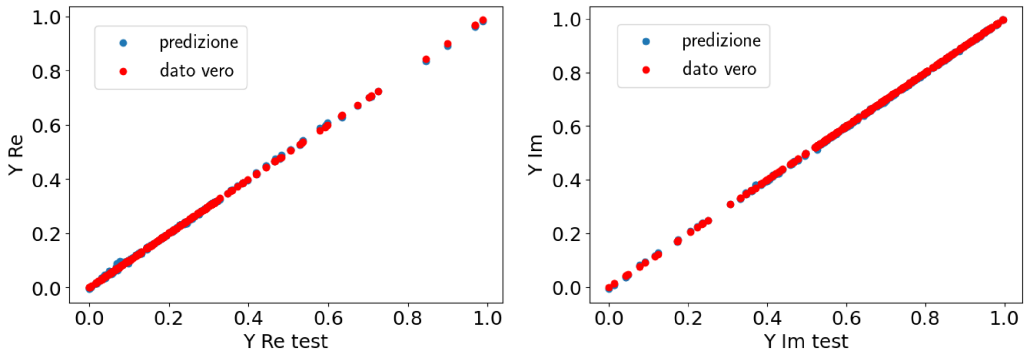


Figura 4.16. Previsione sui dati di test del modello ottimizzato.

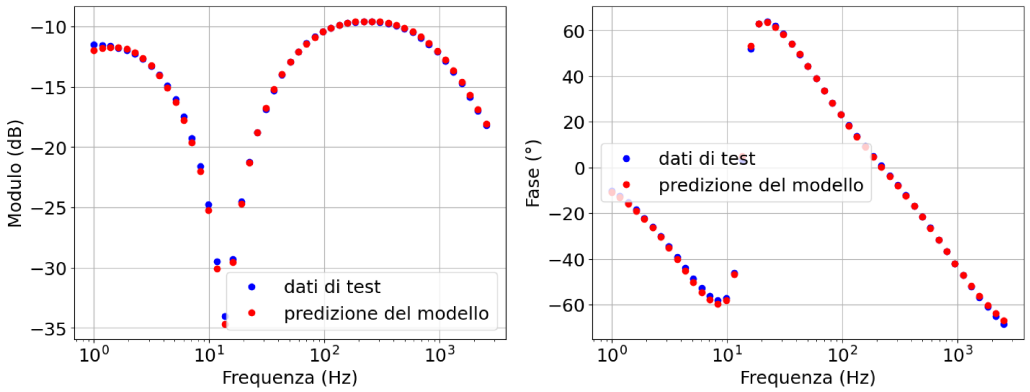


Figura 4.17. Diagramma di Bode del modello ottimizzato.

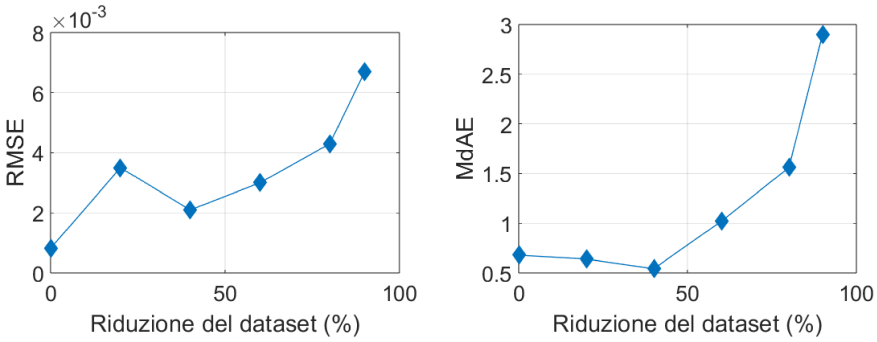
4.5.2 Influenza della dimensione del dataset

Essendo questa la funzione più complessa da approssimare, ha senso verificare l'influenza della dimensione del dataset. Il test è stato fatto sul modello ottenuto manualmente da 650 parametri abbassando il batch size a 64. Il resto delle impostazioni è rimasto inalterato. In fig. 4.18 sono mostrati i risultati.

In questo caso si nota una forte dipendenza dalla dimensione del dataset: oltre il 60% di riduzione si cominciano ad avere errori molto più alti. Ciò significa che per funzioni relativamente complesse è necessario un dataset di dimensione maggiore. Comunque sia, l'errore si mantiene abbastanza basso fino ad un dataset del 40% rispetto all'originale, pari a 1280 punti. Anche qui si capisce che la quantità ottimale di punti non è ben definita ma dipende dalla complessità del problema e dall'accuratezza che si vuole raggiungere.

	Errore	min	max
Modulo	0.0045 S (RMSE)	1.066 S	S
Fase	0.678° (MdAE)	-71.57°	87.05°

Tabella 4.9. Errore sui dati di test del modello ottimizzato.



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.18. Errore del modulo e fase al variare della dimensione del dataset.

4.6 Transfer Learning

Finora si sono ottenuti dei modelli relativi ad uno specifico inverter e con un dataset numeroso. Si vuole ora verificare la fattibilità e l'efficacia del TL applicato su un convertitore nuovo, il quarto. Per fare ciò, occorre prima di tutto addestrare una rete su tutti e tre i convertitori, generando così un modello generico che possiede informazioni su diversi casi. Intuitivamente, questo modello non sarà specializzato in nessun caso particolare e quindi ci si aspetta un errore maggiore rispetto ai singoli casi visti precedentemente. Successivamente, come accennato nel capitolo 1, si vedranno due approcci per specializzare le reti sul convertitore 4. Si vedrà anche qual è l'influenza del numero di dati presi per l'addestramento di fino per capire se esiste un numero ideale di punti da scegliere.

4.6.1 Y_{dd} con rimozione di layer

Il primo metodo che si vuole presentare è quello solitamente utilizzato per il TL, ovvero rimuovere alcuni layer (di solito gli ultimi) e riaddestrare la rete mantenendo, però, inalterati i pesi dei layer originali. Quindi, di fatto, addestrare solo i nuovi layer aggiunti. Per creare il dataset comprensivo dei tre convertitori si sono uniti i dataset rispettivi, creandone uno unico da 9600 punti. Il preaddestramento è stato fatto con l'aiuto dell'ottimizzatore per trovare il modello più adatto. Il modello migliore è risultato essere composto da 3 layer, 45 neuroni e 614 parametri. La bontà delle previsioni del modello è peggiore rispetto ai singoli casi, come previsto (vedasi fig. 4.19).

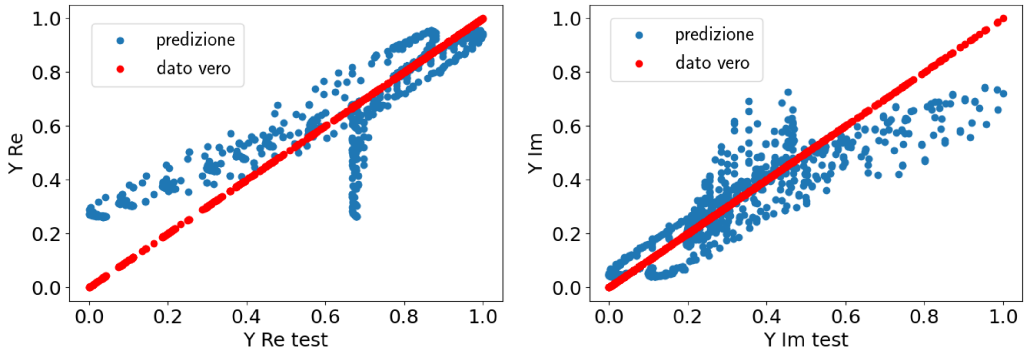


Figura 4.19. Previsione sui dati di test del modello preaddestrato.

L'errore del modello risulta piuttosto alto, come si vede dalla tabella 4.10.

	Errore	min	max
Modulo	0.178 S (RMSE)	0.094 S	1.076 S
Fase	8.86° (MdAE)	-71°	179.92°

Tabella 4.10. Errore sui dati di test del modello preaddestrato.

Ottenuto il modello preaddestrato, è possibile rimuovere alcuni layer e sostituirli con dei nuovi. È intuibile che più neuroni si aggiungono, migliori saranno le prestazioni. Ma c'è da fare attenzione a non rimuovere troppi layer, altrimenti si rischia di togliere troppe informazioni date dal preaddestramento e vanificare il processo. Una volta caricato il modello preaddestrato si sono rimossi gli ultimi due layer e sostituiti con altri due costituiti da 12 e 2 neuroni rispettivamente, per un totale di 354 parametri. Il numero di punti presi dal dataset del convertitore 4 è stato scelto pari a 500 per iniziare. Questo valore è stato poi abbassato per analizzare l'andamento dell'errore al variare del numero di punti. I risultati ottenuti sono stati piuttosto promettenti. La rete è riuscita a specializzarsi sull'ultimo convertitore con buone previsioni, come si vede in fig. 4.20 e 4.21.

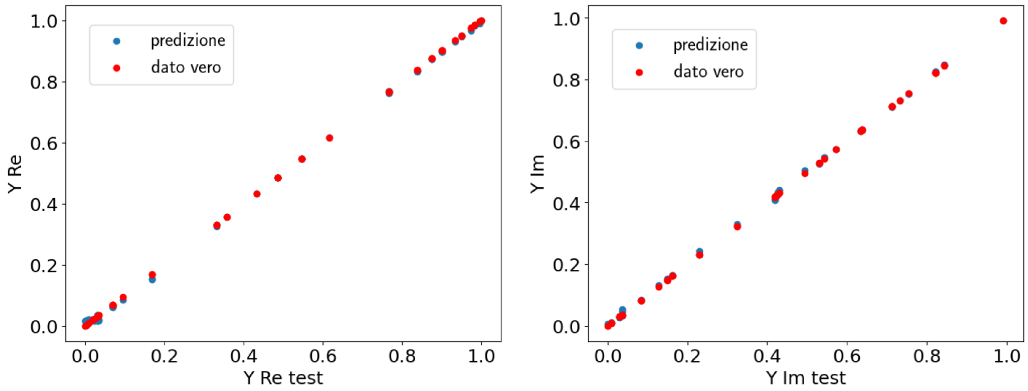


Figura 4.20. Previsioni del modello con TL rimuovendo 2 layer.

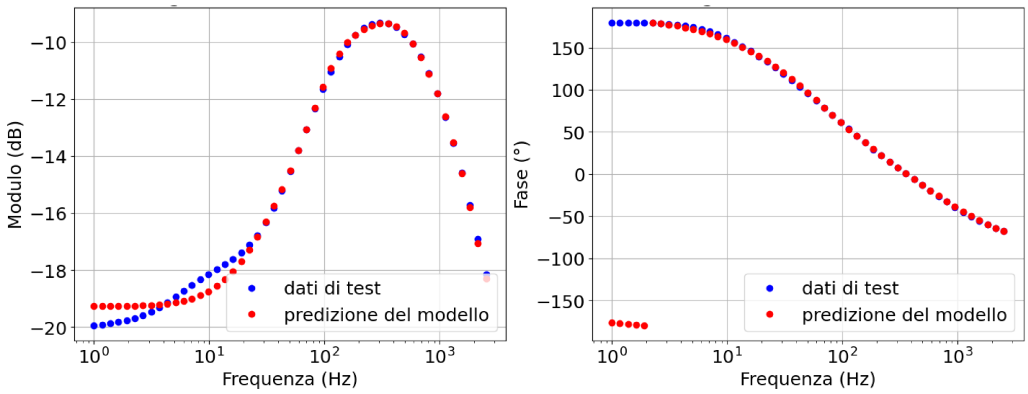


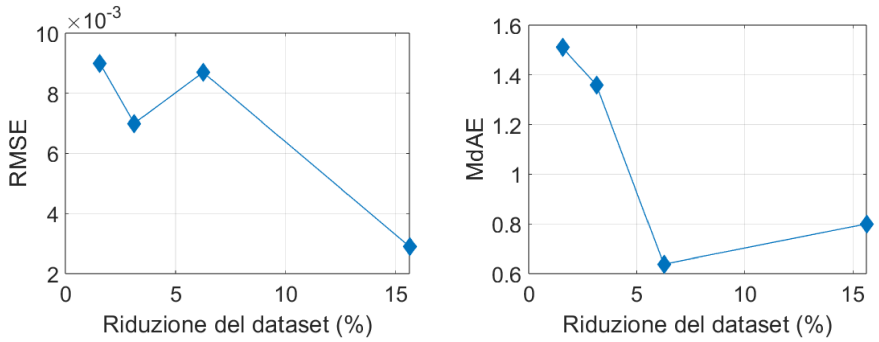
Figura 4.21. Diagramma di Bode con TL rimuovendo 2 layer.

In tabella 4.11 è riportato l'errore del modello ottenuto.

	Errore	min	max
Modulo	0.0029 S (RMSE)	0.1 S	0.342 S
Fase	0.9° (MdAE)	-71°	179.92°

Tabella 4.11. Errore sui dati di test con TL rimuovendo 2 layer.

Questi risultati dimostrano che il TL funziona ed è possibile addestrare un modello e specializzarlo per un problema specifico partendo da uno problema più generico. Il tutto con un numero limitato di punti, si ricorda che si è usato solo il 15% del dataset totale. È lecito chiedersi, ora, quale sia il numero minimo di punti che permetta alla rete di addestrarsi in maniera efficace. Concettualmente si può ritornare alle conclusioni fatte precedentemente sull'influenza della dimensione del dataset. Quindi la dimensione corretta dipende dall'errore che si è disposti a tollerare. Si è analizzato l'errore al variare del numero di punti presi e i risultati raccolti sono riportati in fig. 4.22. Si vede che l'errore del modulo rimane sotto il 10% rispetto al modulo minimo fino a 50 punti (l'1.5% del dataset



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.22. Errore del modulo e fase al variare del numero di punti presi, riferito al dataset completo di 3200 punti.

originale). Ciò è molto promettente perché mostra l'efficacia del TL anche con pochissimi dati.

4.6.2 Y_{dd} con riaddestramento

Vista l'efficacia del metodo basato sull'aggiunta di nuovi layer, si vuole ora confrontarlo con il metodo che prevede il riaddestramento completo della rete. Si è ripreso anche in questo caso lo stesso modello preaddestrato e, anziché modificarne la struttura o i pesi, si è riaddestrato completamente fornendo solo un numero limitato di punti del nuovo dataset. Per facilitare il riaddestramento si è aumentato leggermente il learning rate da 0.001 a 0.002. I risultati sono stati migliori del primo metodo come si vede dalle immagini sotto. La bontà della stima del modello riaddestrato è mostrata in fig. 4.23 e in fig. 4.24 si vede il diagramma di Bode.

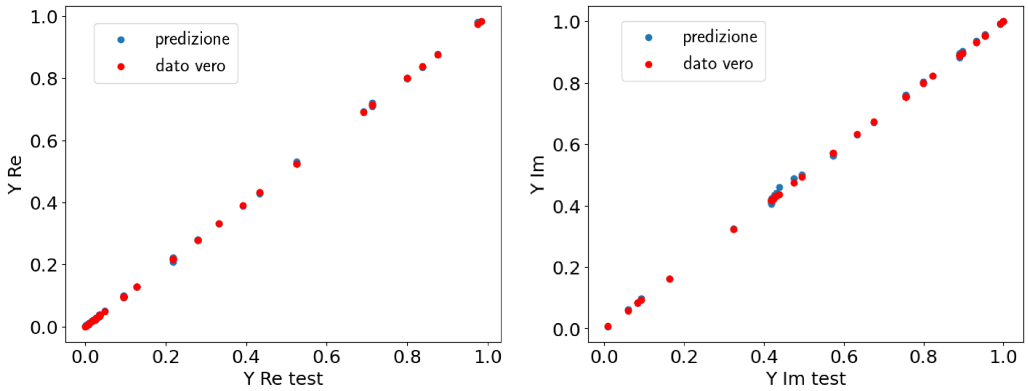


Figura 4.23. Previsioni del modello riaddestrato.

Si nota dalla fig. 4.24 che ora la previsione del modello è più fedele ai dati attesi. Questo comportamento è spiegabile con il fatto che riaddestrando l'intero modello è possibile raggiungere pesi più ottimizzati rispetto al primo metodo che mantiene i primi layer fissi. Comunque l'addestramento risulterà più semplice e rapido dal momento che il punto di partenza sono dei pesi vicini a quelli ottimali. Ulteriore conferma della bontà del metodo è l'errore riportato in tabella 4.12. Si nota un miglioramento rispetto al primo metodo e si ripercuote anche sull'influenza del numero di punti usati per il riaddestramento. Questo secondo metodo consente un errore minore a parità di dati, come mostrato in fig. 4.25.

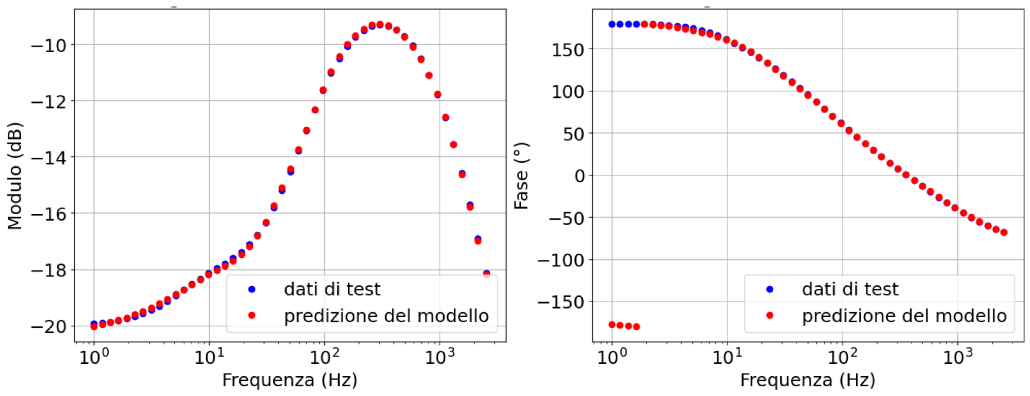
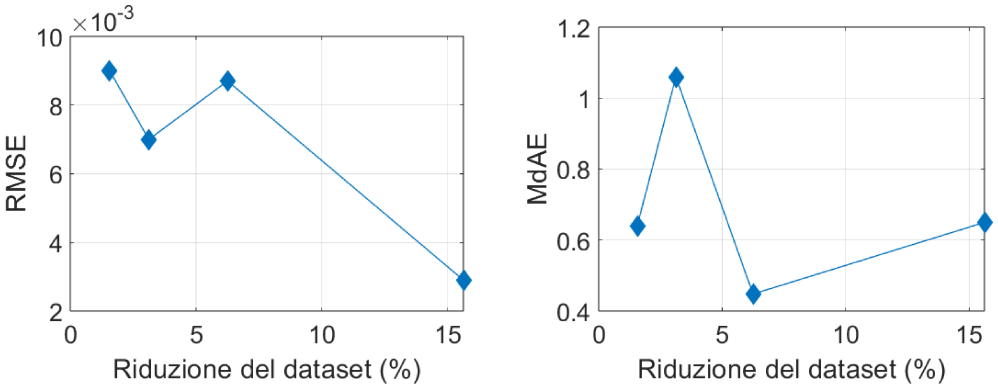


Figura 4.24. Diagramma di Bode del modello riaddestrato.

	Errore	min	max
Modulo	0.0012 S (RMSE)	0.1 S	0.342 S
Fase	8.86° (MdAE)	-71°	179.92°

Tabella 4.12. Errore sui dati di test.



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.25. Errore del modulo e fase al variare del numero di punti presi.

4.6.3 Y_{dq} con rimozione di layer

Essendo questa componente costantemente nulla ci si aspetta di ottenere risultati equivalenti con entrambi i metodi. L'errore raggiunto nel preaddestramento è stato di appena $0.56 \cdot 10^{-3}$ S per il modulo, per ovvi motivi si eviterà di portare i grafici di Bode e di previsione del modello. Una volta fatto il preaddestramento con il dataset comprensivo dei tre convertitori, si sono rimossi gli ultimi due layer, lasciandone solo uno, e se ne sono aggiunti due di nuovi da 4 e 2 neuroni rispettivamente. L'addestramento del nuovo modello è stato fatto prendendo 500 punti e ha raggiunto addirittura valori di errore nulli e le previsioni del modello erano esattamente zero. Per tale ragione si evita di riportare ulteriori grafici e tabelle. Sono state fatte prove con meno punti ma i risultati sono stati analoghi.

4.6.4 Y_{dq} con riaddestramento

Utilizzando il secondo metodo si sono ottenuti ottimi risultati anche se non perfetti come il primo metodo. Le previsioni del modello non erano esattamente nulle anche se erano dell'ordine di 10^{-6}. L'errore RMSE del modulo è stato pari a $0.65 \cdot 10^{-3}$ S. L'errore sulla fase non è stato considerato visto che si tratta di moduli così bassi e il valore risultante sarebbe dovuto ad errori numerici.

4.6.5 Y_{qd} con rimozione di layer

La rete che meglio approssimava il dataset combinato è risultata essere composta da 4 layer, 51 neuroni e 451 parametri. Le previsioni del modello risultante sono mostrate in fig. 4.26. La tabella 4.13 mostra l'errore misurato del modello preaddestrato.

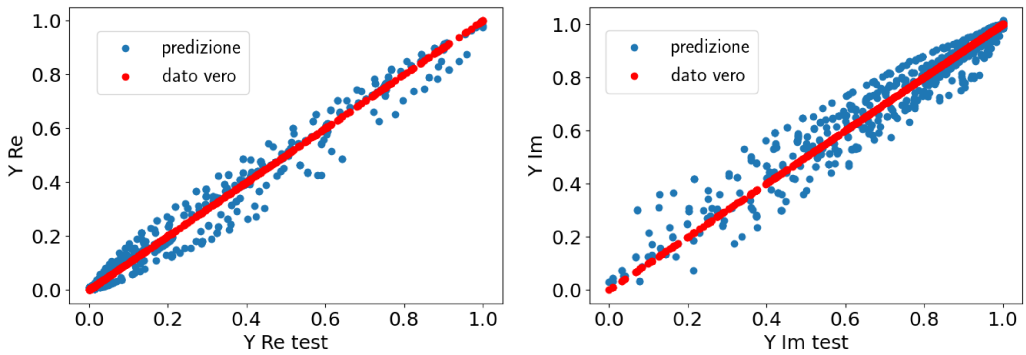


Figura 4.26. Predizione del modello preaddestrato.

	Errore	min	max
Modulo	0.05 S (RMSE)	0 S	1.543 S
Fase	7.67° (MdAE)	-147°	0°

Tabella 4.13. Errore sui dati di test del modello preaddestrato.

Anche qui si sono rimossi gli ultimi due layer e con l'algoritmo di ottimizzazione si sono aggiunti 3 layer per un totale di 50 neuroni e 959 parametri. Con 500 punti si sono ottenuti buoni risultati, come mostrato in fig. 4.27 e 4.28.

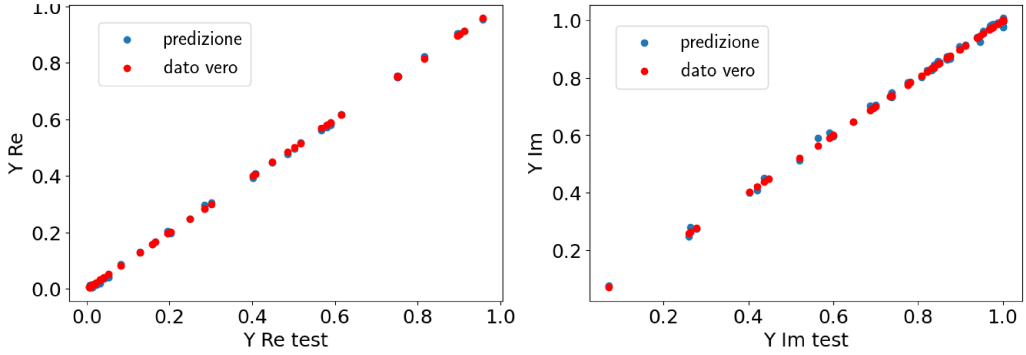


Figura 4.27. Previsioni del modello con TL rimuovendo 2 layer.

L'errore ottenuto con questo primo metodo di TL è riportato in tabella 4.14. Abbassando il numero di punti si è ottenuto, come negli altri casi, un peggioramento dell'errore, come visibile nel grafico 4.29. In questo caso l'errore rimane stabile fino a riduzioni molto spinte (sotto il 5%).

	Errore	min	max
Modulo	0.0087 S (RMSE)	0 S	1.52 S
Fase	0.895° (MdAE)	-139.65°	0°

Tabella 4.14. Errore sui dati di test con TL rimuovendo 2 layer.

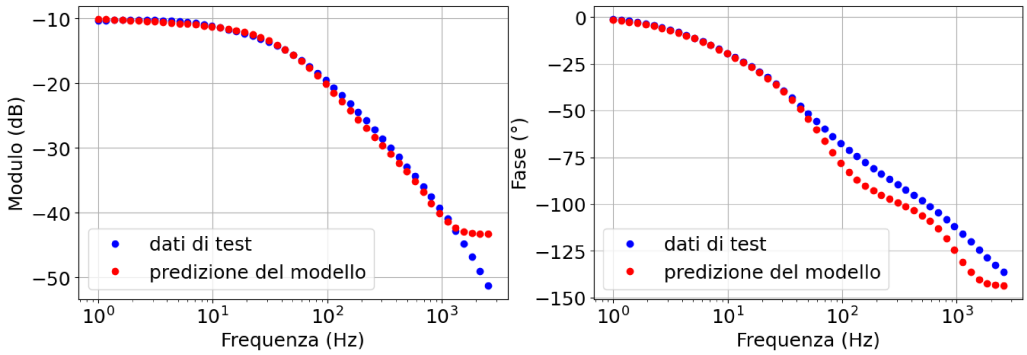
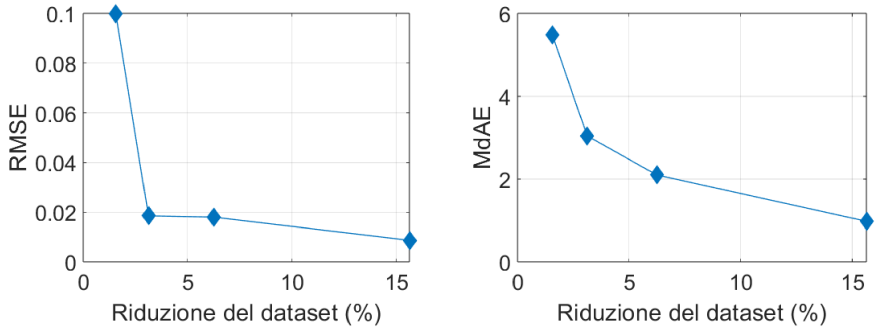


Figura 4.28. Diagramma di Bode con TL rimuovendo 2 layer.



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.29. Errore del modulo e fase al variare del numero di punti presi.

4.6.6 Y_{qd} con riaddestramento

Partendo dal modello preaddestrato e impostando un learning rate di 0.002 e un batch size di 32, si è ottenuto un nuovo modello che ha riportato errori, stavolta, leggermente peggiori rispetto al primo metodo di TL. Con 500 punti si sono ottenuti i risultati di fig. 4.30 e 4.31.

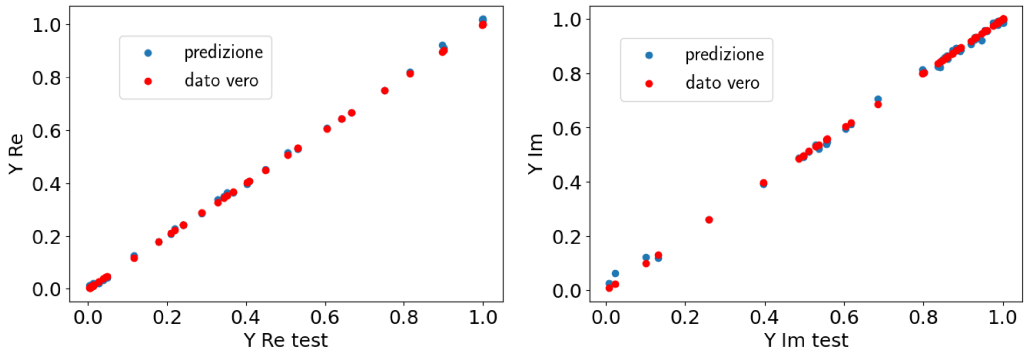


Figura 4.30. Previsioni del modello riaddestrato.

L'errore misurato in questo caso è riportato in tabella 4.15. L'errore massimo si è rivelato essere minore ma più sensibile al numero di punti, vedasi grafici 4.32.

4.6.7 Y_{qq} con rimozione di layer

Con l'utilizzo dell'ottimizzatore si è trovata una rete ottimale per ridurre il più possibile l'errore durante il preaddestramento. La configurazione trovata è costituita da 4 layer, 60 neuroni e 881 parametri con sigmoid come funzione di attivazione. L'errore ottenuto è riportato nella tabella 4.16. Sorprendentemente è più basso dell'errore

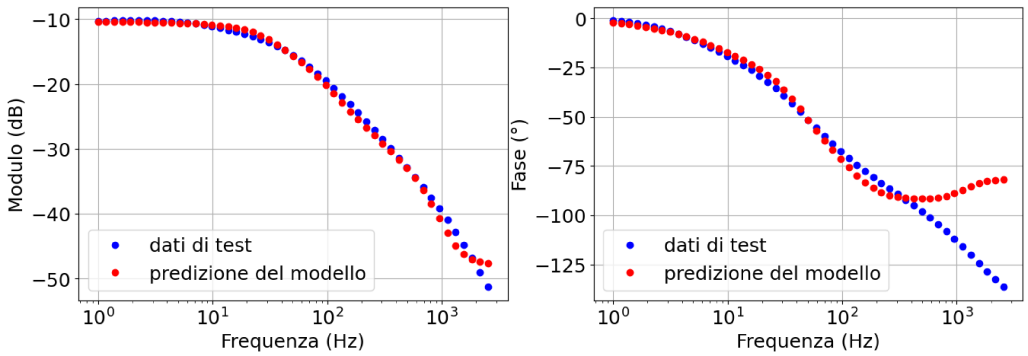
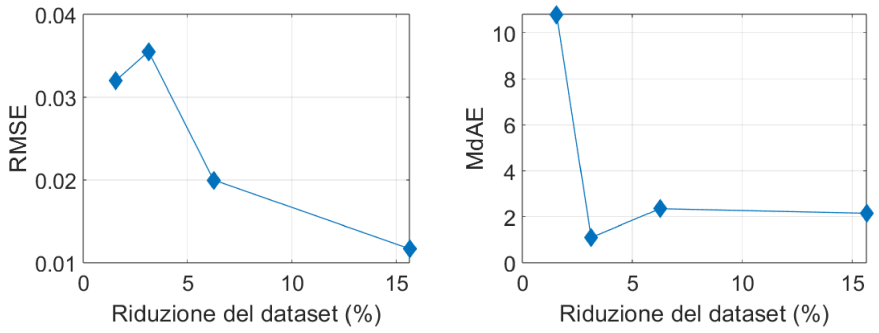


Figura 4.31. Diagramma di Bode del modello riaddestrato.

	Errore	min	max
Modulo	0.0117 S (RMSE)	0 S	1.52 S
Fase	2.15° (MdAE)	-139.65°	0°

Tabella 4.15. Errore sui dati di test riaddestrando il modello.



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.32. Errore del modulo e fase al variare del numero di punti presi.

trovato con il modello addestrato esclusivamente sul convertitore 1 (vedasi tabella 4.9). Questo fatto è probabilmente dovuto ad una architettura particolarmente ottimizzata per questo dataset e dal fatto che Y_{qq} sia molto simile tra i vari inverter e la rete ha potuto approssimare molto bene tale funzione, avendo a disposizione un dataset molto grande.

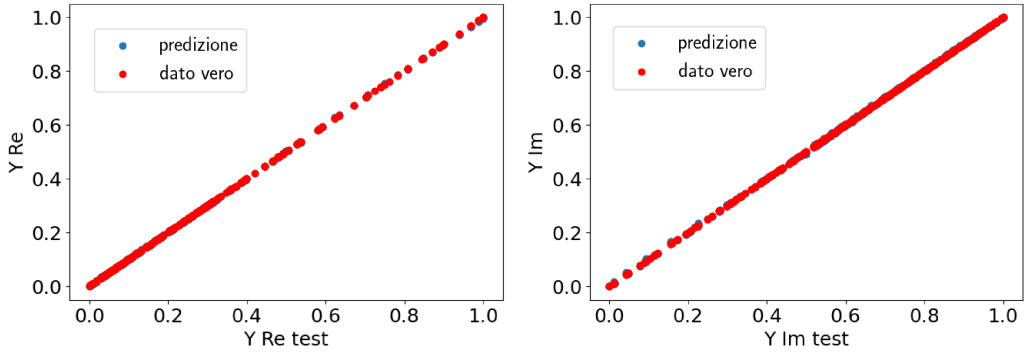
La predizione del modello preaddestrato è praticamente perfetta, come si vede dalle fig. 4.33 e 4.34. Avendo ottenuto risultati così buoni, ci si aspetta che la specializzazione sui dati del quarto inverter sia altrettanto buona con entrambi i metodi. Anche qui si sono rimossi gli ultimi due layer e sostituiti con altri due nuovi da 24 e 2 neuroni rispettivamente. Dopo qualche prova si è ottenuto un modello con un errore riportato in tabella 4.17, utilizzando 500 punti.

Sorprendentemente, si sono ottenuti leggermente peggiori al caso preaddestrato. Probabilmente perché la rimozione di alcuni layer e il conseguente cambio di pesi, mantenendone alcuni inalterati, ha creato un modello non ottimizzato come nel modello preaddestrato. Comunque sia, il modello risultante è in grado di predire molto bene i dati di test, come si vede dalle fig. 4.35 e 4.36.

La componente Y_{qq} si è visto essere piuttosto simile per ogni inverter, perciò si può supporre che la relativa rete neurale sia più immune alla riduzione del numero di punti usati. Infatti, in fig. 4.37 è mostrato l'andamento dell'errore al variare del numero di punti presi. Sebbene ci sia un aumento per riduzioni significative del dataset, i valori massimi registrati sono comunque tra i più bassi misurati, mostrando la bontà del metodo.

	Errore	min	max
Modulo	0.0014 S (RMSE)	0.0067 S	1.07 S
Fase	0.25° (MdAE)	-71.6°	87°

Tabella 4.16. Errore sui dati di test del modello preaddestrato.



!\[h\]

Figura 4.33. Previsione del modello preaddestrato.

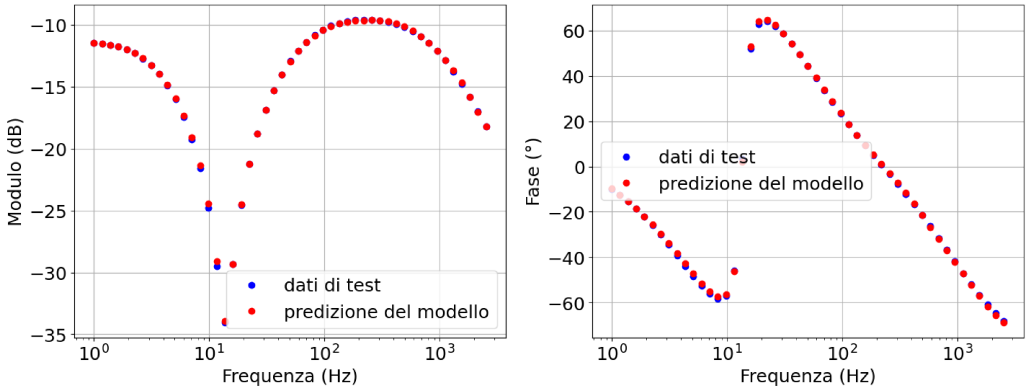


Figura 4.34. Diagramma di Bode del modello preaddestrato.

	Errore	min	max
Modulo	0.0084 S (RMSE)	0.05 S	0.99 S
Fase	0.72° (MdAE)	-67°	87°

Tabella 4.17. Errore sui dati di test con TL rimuovendo 2 layer.

4.6.8 Y_{qq} con riaddestramento

Per le osservazioni fatte poco prima, si presume che questo metodo sia più efficace per il transfer learning, in quanto vengono aggiornati di poco tutti i pesi della rete e quindi si riesce a raggiungere un modello più ottimizzato. Infatti, sono state necessarie appena un centinaio di epoche per riaddestrare il modello sui dati del quarto inverter e l'errore raggiunto è stato molto più basso rispetto al primo metodo di transfer learning, mostrato nella tabella 4.18.

In fig. 4.38 e 4.39 sono mostrati le previsioni e il diagramma di Bode del modello riaddestrato. Si vede che il modello ottenuto in questo modo è in grado di approssimare ancora meglio la funzione.

Oltre a garantire errori più bassi, riaddestrare il modello consente di avere una minore dipendenza dal numero di dati usati per il riaddestramento (vedasi fig. 4.40). Questo era prevedibile dal momento che il modello preaddestrato possiede già un'ottima conoscenza dell'andamento generale di Y_{qq} e anche se si utilizzano pochi punti nel riaddestramento il modello rimane in grado di fare delle buone stime. C'è comunque un tendente aumento

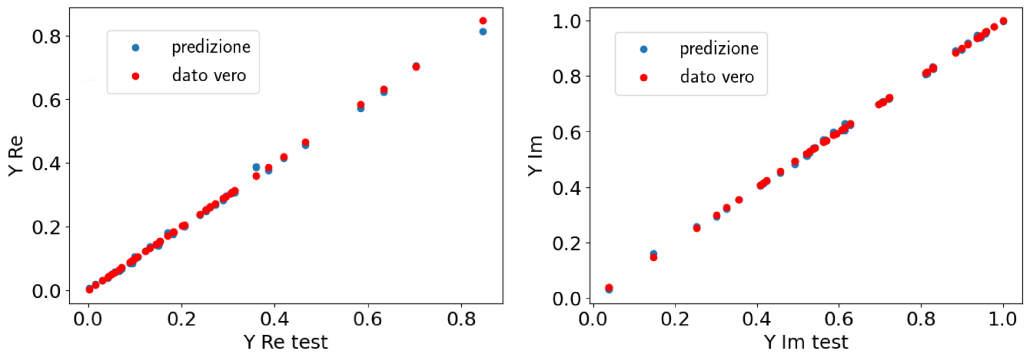


Figura 4.35. Previsione del modello con TL rimuovendo 2 layer.

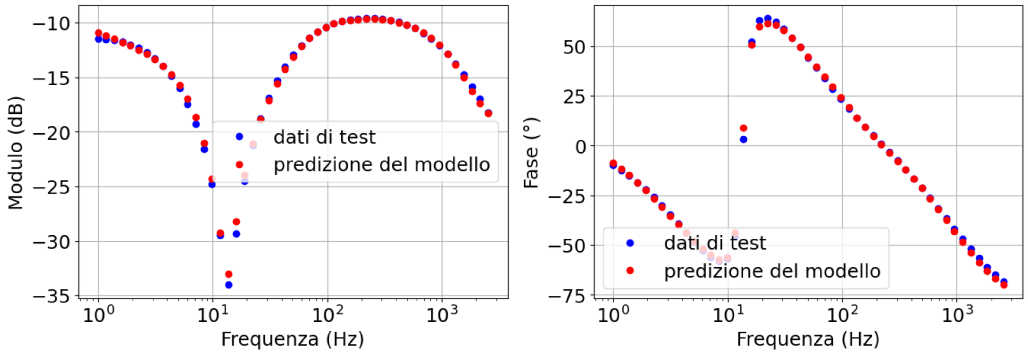
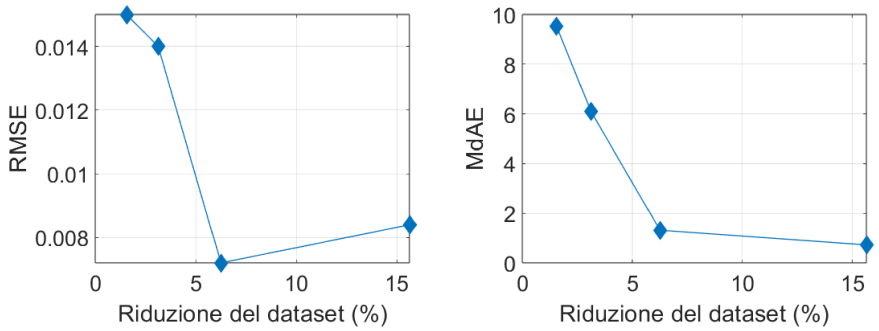


Figura 4.36. Diagramma di Bode con TL rimuovendo 2 layer.



(a) RMSE del modulo.

(b) MdAE della fase.

Figura 4.37. Errore del modulo e fase al variare del numero di punti presi.

dell'errore, in particolare quello del modulo, ma rimane a valori estremamente bassi se confrontato con quello di fig. 4.37.

	Errore	min	max
Modulo	0.0012 S (RMSE)	0.06 S	0.89 S
Fase	0.56° (MdAE)	-52.7°	41.8°

Tabella 4.18. Errore sui dati di test riaddestrando il modello.

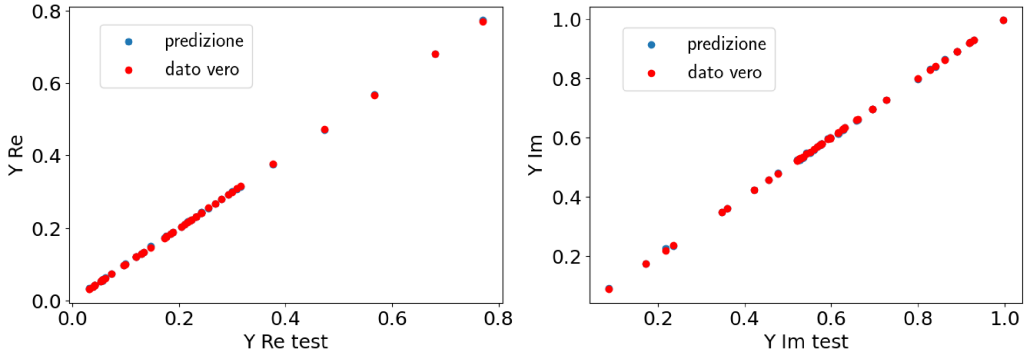


Figura 4.38. Previsione del modello riaddestrato.

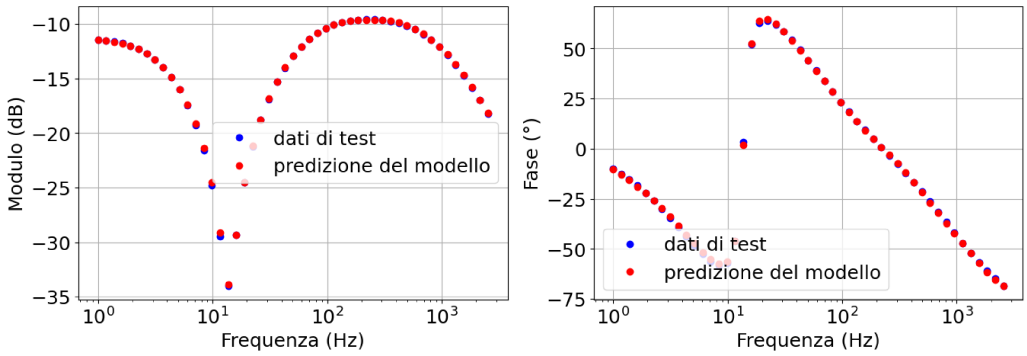
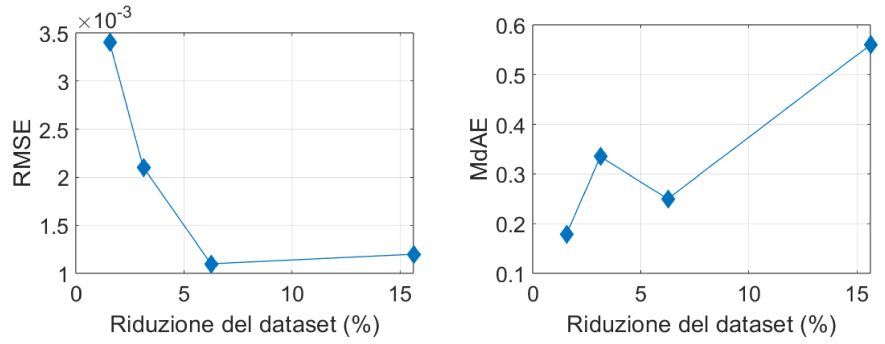


Figura 4.39. Diagramma di Bode del modello riaddestrato.



(a) RMSE del modulo del modello riaddestrato. (b) MdAE della fase del modello riaddestrato.

Figura 4.40. Errore del modulo e fase al variare del numero di punti presi.

4.7 Confronto con una singola rete neurale

Finora si sono usate 4 reti neurali per suddividere il problema in sottoproblemi più semplici e semplificare, di conseguenza, la complessità delle reti neurali. Sorge spontanea la domanda di come si comporterebbe un'unica rete neurale per stimare l'intera matrice dell'ammettenza. In quel caso si deve avere una rete con un layer di uscita composto da 8 neuroni: 2 (parte reale e immaginaria) × 4 (componenti da stimare). In questa sezione si vogliono confrontare i risultati finora ottenuti con quelli di un modello realizzato secondo questo criterio. Si presume che una singola rete che stimi tutte le 8 componenti dell'ammettenza sia necessariamente più complessa delle singole reti. Tuttavia, se le uscite hanno degli andamenti simili tra loro o, semplicemente, molti valori in comune, si riduce la complessità totale della rete necessaria. Secondo questa considerazione ci si aspetta un modello più complesso ma non di molto.

Per verifica si è usato il dataset relativo al primo convertitore, mantenendo la stessa dimensione (3200 punti). Con l'aiuto dell'ottimizzatore di keras si è cercata una architettura che minimizzasse l'errore e composta al massimo da 5 layer. La ricerca ha restituito un modello da 4 layer, 77 neuroni e 1359 parametri, circa il doppio rispetto alle singole reti. La bontà delle previsioni è stata sorprendente: gli errori medi del modulo hanno raggiunto valori appena superiori a quelli ottenuti con i singoli modelli ottimizzati con il tuner, vedasi la tabella 4.19. In fig. 4.41 sono visibili le previsioni del modello per tutte le 8 componenti.

	Y_{dd}	Y_{dq}	Y_{qd}	Y_{qq}
RMSE modulo	0.68 mS	0.002 S	0.0057 S	0.0057
MdAE fase	0.166°	128.1°	1.18°	0.37°
min modulo	0.1 S	0 S	0 S	0.006 S
min fase	-71.13°	0°	-147°	-71.57°
max modulo	0.342 S	0 S	1.544 S	1.06 S
max fase	179.92°	0°	0°	87°

Tabella 4.19. Errore sui dati di test.

È interessante verificare anche il comportamento di questo modello con il TL e verificare come si comporta la singola rete con i due metodi.

4.7.1 TL con rimozione di layer

Per cominciare si è ottenuto il modello preaddestrato sui primi 3 inverter, ottenendo gli errori di tabella 4.20.

	Y_{dd}	Y_{dq}	Y_{qd}	Y_{qq}
RMSE modulo	0.185 S	0.0048 S	0.049 S	0.009
MdAE fase	8.58°	85.66°	6.81°	1.02°
min modulo	0.094 S	0 S	0 S	0.006 S
min fase	-71.22°	0°	-147°	-71.57°
max modulo	1.076 S	0 S	1.544 S	1.07 S
max fase	179.92°	0°	0°	87°

Tabella 4.20. Errore sui dati di test del modello preaddestrato.

Per il primo metodo si sono tolti gli ultimi due layer e con l'ottimizzatore si è cercata una configurazione ottimale composta da non più di 5 layer. È risultato un modello composta da proprio 5 layer, 95 neuroni e 1697 parametri totali. Anche qui, per l'addestramento si sono usati 500 punti iniziali del dataset del quarto convertitore. L'errore del modello così ottenuto è riportato in tabella 4.21. Gli errori risultano abbastanza simili a quelli misurati per le singole reti, soprattutto gli errori del modulo, mostrando un'ottima efficacia anche con una singola rete.

A livello grafico si può vedere la capacità del modello nel ricostruire tutte le componenti dell'ammettenza nei diagrammi di Bode di fig. 4.42. Si nota che alcune componenti sono meglio approssimate di altre, ma in generale sono piuttosto fedeli e si ottengono comportamenti simili a quelli trovati in precedenza.

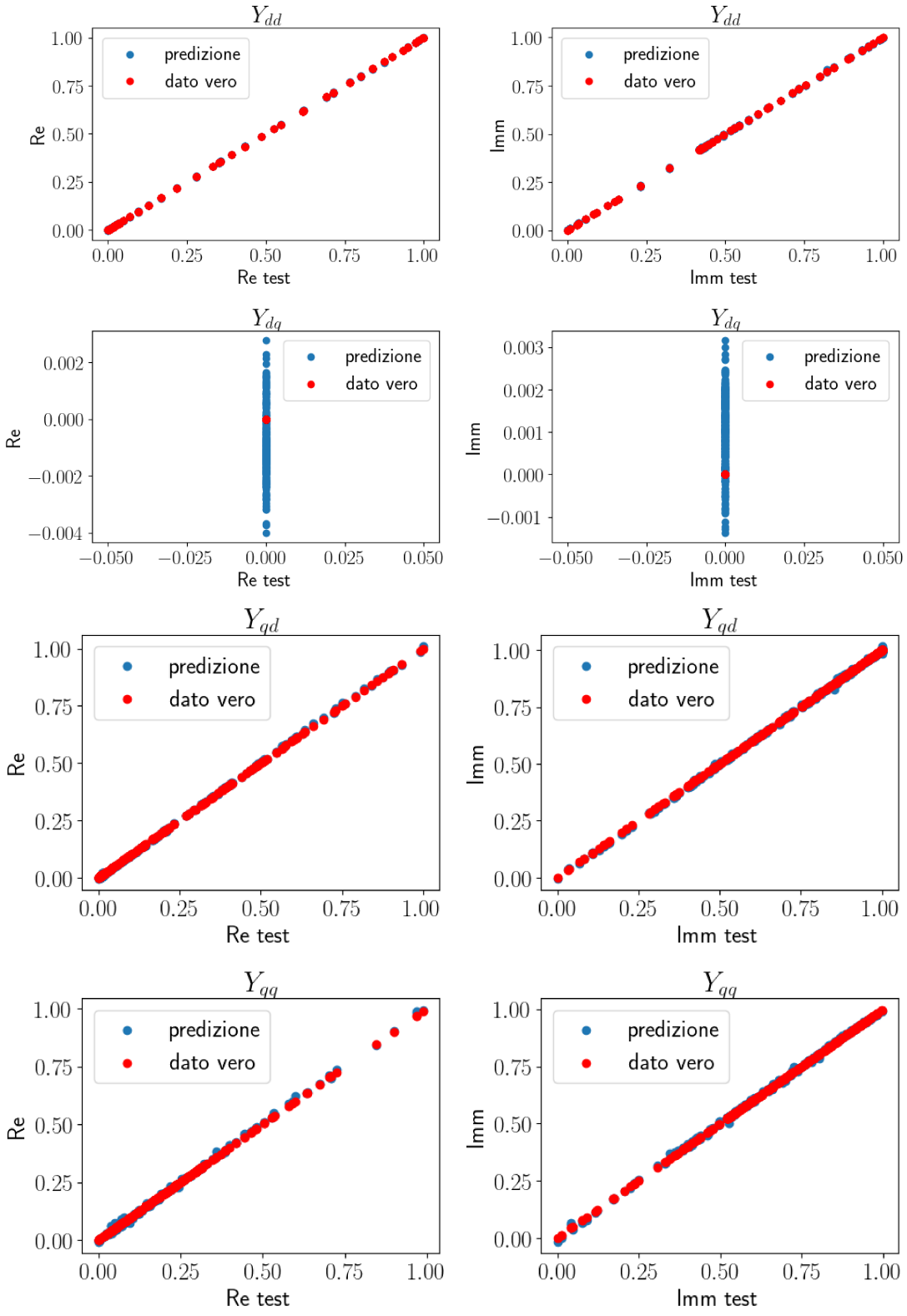


Figura 4.41. Previsioni della singola rete neurale.

4.7.2 TL con riaddestramento

Nei singoli casi si era visto che, in generale, si ottenevano errori più piccoli riaddestrando l'intero modello sui dati del quarto convertitore, perciò ci si aspetta un comportamento simile anche qui. Durante il riaddestramento si è notata la necessità di ridurre il tasso di apprendimento, altrimenti la funzione costo aveva delle oscillazioni

	Y_{dd}	Y_{dq}	Y_{qd}	Y_{qq}
RMSE modulo	0.0034 S	0.0047 S	0.01 S	0.012 S
MdAE fase	0.6°	130.17°	1.14°	1.35°
min modulo	0.1 S	0 S	0 S	0.039 S
min fase	-71.22°	0°	-139.65°	-71.4°
max modulo	0.342 S	0 S	1.544 S	1.07 S
max fase	179.92°	0°	0°	74°

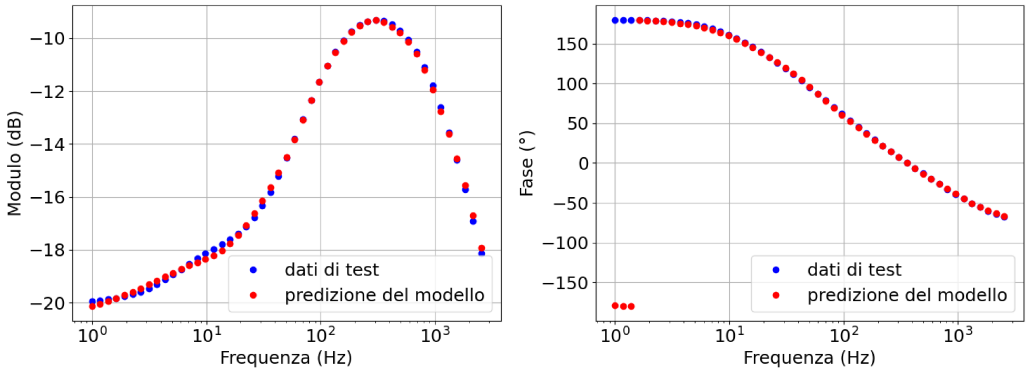
Tabella 4.21. Errore sui dati di test con TL rimuovendo 2 layer.

eccessive che non permettevano di convergere ad una soluzione ottimale. Un valore buono si è visto essere 0.0005. La tabella 4.22 riporta l'errore misurato con questo metodo.

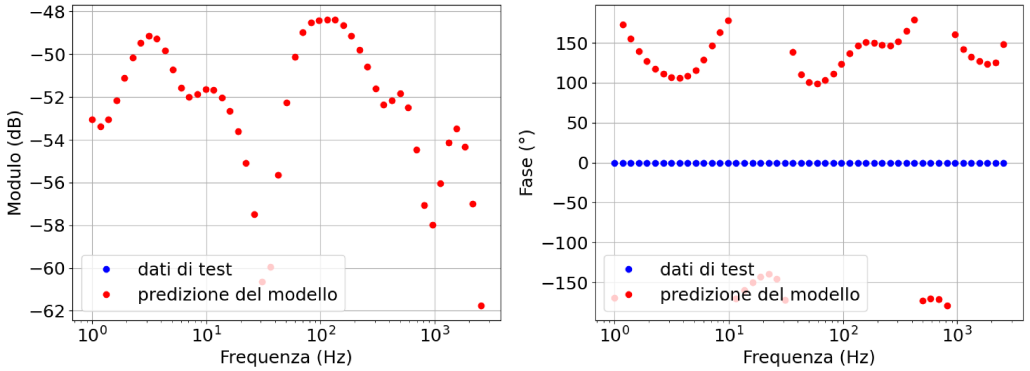
	Y_{dd}	Y_{dq}	Y_{qd}	Y_{qq}
RMSE modulo	0.0028 S	0.0032 S	0.007 S	0.0072
MdAE fase	0.43°	104.33°	0.74°	0.55°
min modulo	0.1 S	0 S	0 S	0.019 S
min fase	-71.22°	0°	-120°	-71.4°
max modulo	0.342 S	0 S	1.389 S	1.07 S
max fase	179.92°	0°	0°	87°

Tabella 4.22. Errore sui dati di test del modello riaddestrato.

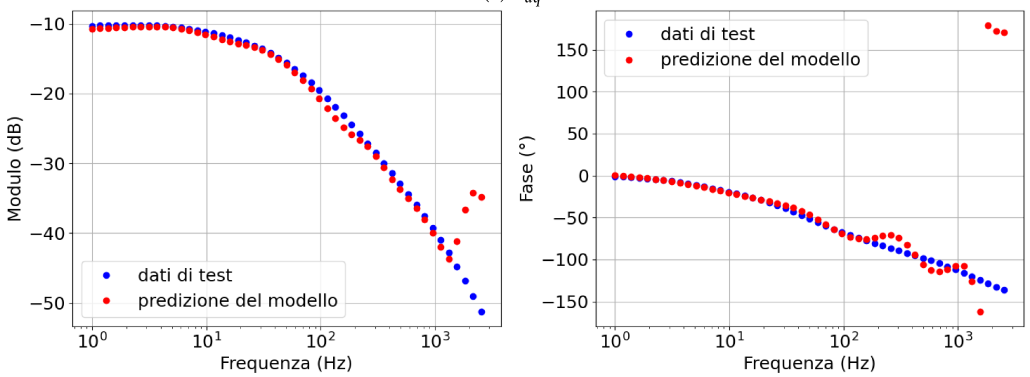
Si può affermare che questo metodo rimane il più efficace, anche se il margine di miglioramento rispetto al primo metodo è piuttosto limitato. Forse all'aumentare della complessità della rete si ha una convergenza dei due metodi a soluzioni molto simili tra loro. In fig. 4.43 sono visibili i grafici di Bode del modello riaddestrato e si vede che la differenza con la fig.4.42 è limitata, segno che i due approcci hanno avuto equivalente efficacia.



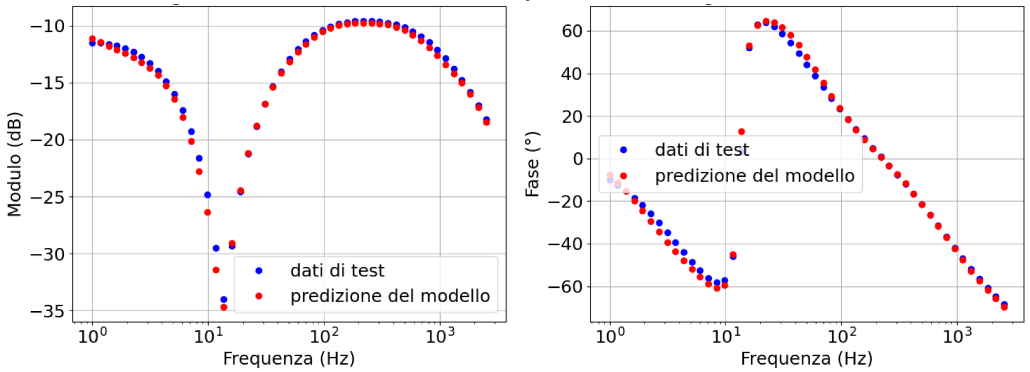
(a) Y_{dd}.



(b) Y_{dq}.

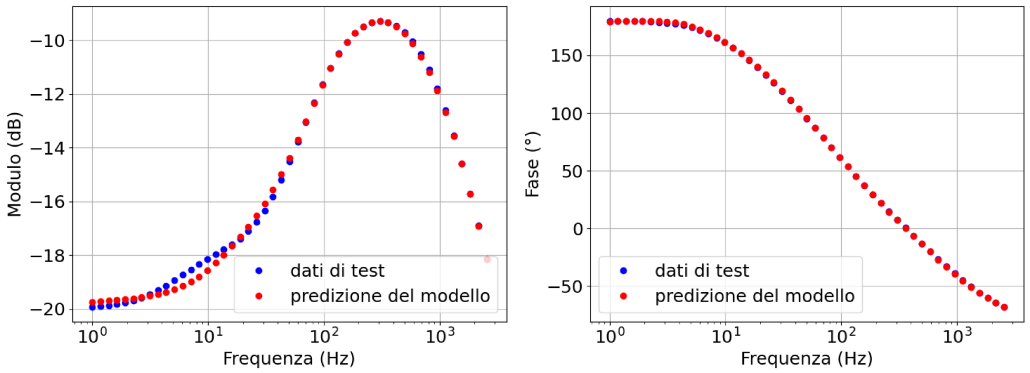


(c) Y_{qd}.

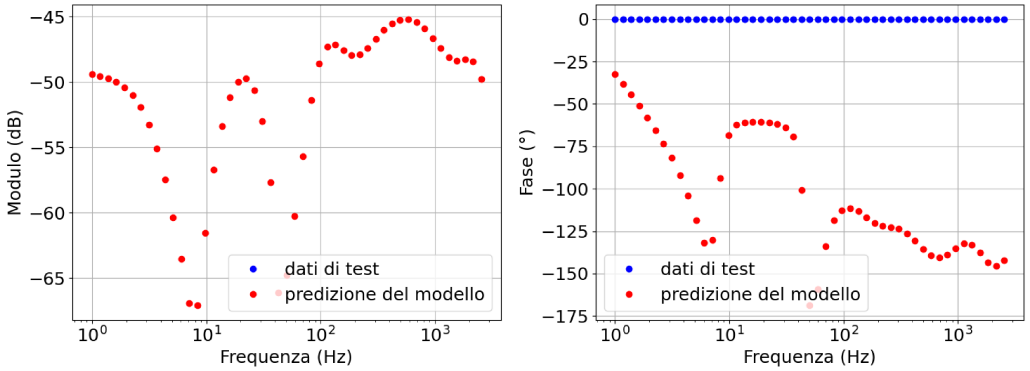


(d) Y_{qq}.

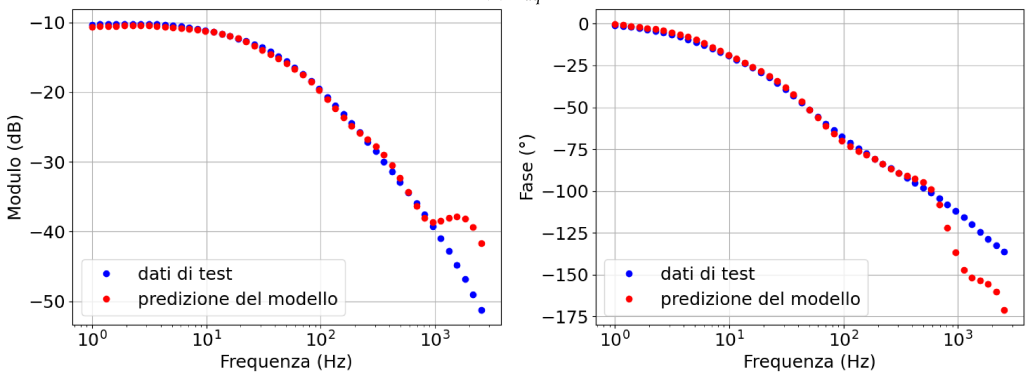
Figura 4.42. Diagramma di Bode con TL rimuovendo 2 layer.



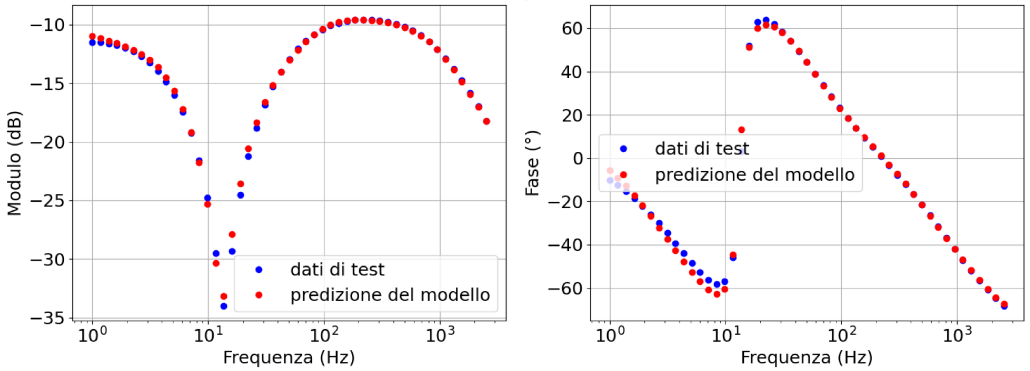
(a) Y_{dd}.



(b) Y_{dq}.



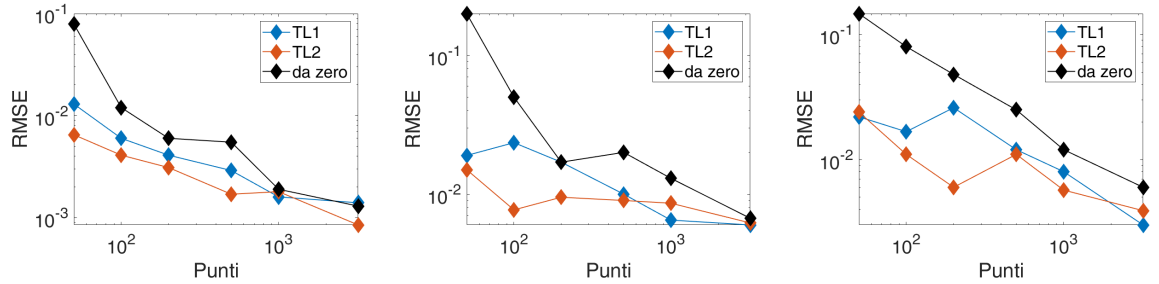
(c) Y_{qd}.



(d) Y_{qq}.

Figura 4.43. Diagramma di Bode del modello riaddestrato.

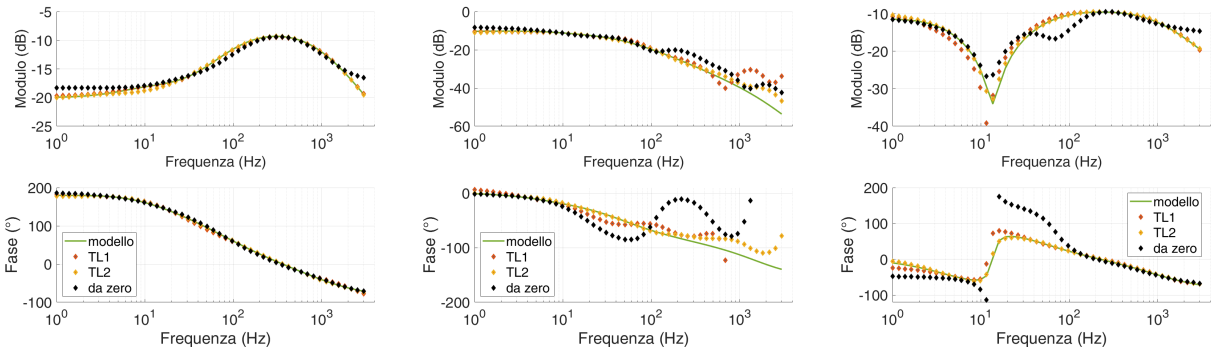
Come ultimo confronto, è interessante analizzare l'errore al variare del numero di punti usati per i due metodi di TL e un modello addestrato da zero sul convertitore 4. In fig. 4.44 è mostrato l'errore RMSE del modulo delle componenti Y_{dd}, Y_{qd} e Y_{qq}. Y_{dq} non si è considerata, visto che il suo valore è costante a zero e non avrebbe senso monitorare l'errore del modello. Il primo metodo di TL è stato nominato "TL1" e ilo secondo "TL2". I dati sono stati ottenuti addestrando la rete per 1000 epoche. Si vede che l'RMSE dato dal TL è quasi sempre minore di quello di un modello addestrato da zero e per ottenere lo stesso errore occorrono migliaia di punti. Si nota che già con 50 punti prelevati dal dataset l'errore è circa 10 volte minore rispetto al modello addestrato da zero. Ciò è un'ulteriore conferma dell'efficacia e dei vantaggi offerti dal TL.



(a) Y_{dd}. (b) Y_{qd}. (c) Y_{qq}.

Figura 4.44. Andamento dell'RMSE del modulo per un modello definito da zero e i due metodi di TL per il convertitore 4.

Volendo visualizzare la capacità di previsione dei vari modelli, in fig. 4.45 sono mostrati i diagrammi di Bode di Y_{dd}, Y_{qd} e Y_{qq} ottenuti con soli 100 punti per l'addestramento. Sebbene non ci siano grosse differenze tra TL1 e TL2, ci sono errori significativi tra la funzione attesa e quella restituita dal modello addestrato da zero.



(a) Y_{dd}. (b) Y_{qd}. (c) Y_{qq}.

Figura 4.45. Confronto tra i diagrammi di Bode del convertitore 4 con TL1, TL2 e un modello definito da zero addestrati con 100 punti.

Capitolo 5

Conclusioni

L'utilizzo sempre più esteso delle risorse rinnovabili connesse a rete ha stimolato lo studio di dispositivi e tecniche sempre più efficienti per gestire l'immissione di energia nella rete stessa. In questo senso i convertitori DC/AC giocano un ruolo fondamentale e la loro progettazione deve essere fatta con cura per garantire la funzionalità e la stabilità dei convertitori stessi, ma anche della rete alla quale sono collegati. Si è visto come la stabilità può essere garantita in un vasto range di situazioni basandosi sull'impedenza di uscita o, equivalentemente, sull'ammettenza dell'inverter. Come si è visto esistono diversi modi per ricavarla, anche sperimentalmente, e in questa tesi si è voluto dimostrare l'efficacia dell'utilizzo delle reti neurali per stimare tale parametro. Questo metodo consente di ricavare l'ammettenza con pochi parametri di ingresso e tutti ricavabili senza necessariamente misurarli (frequenza, potenza attiva e reattiva immesse).

Si è mostrata anche l'efficacia della tecnica del TL che ha dimostrato ottimi risultati, sopperendo ad uno degli aspetti più delicati e complessi nel deep learning: la creazione di un dataset. Normalmente si dovrebbero raccogliere sufficienti dati, che possono essere anche diverse migliaia di punti, a seconda dell'accuratezza desiderata, per addestrare ogni modello su uno specifico convertitore. Ciò risulta assai dispendioso e lungo. Con il TL si è dimostrato che, partendo da un modello preaddestrato su una collezione di convertitori diversi, con poche centinaia di punti relativi ad uno specifico convertitore è stato possibile ottenere un modello con caratteristiche equivalenti ad uno ottenuto da un addestramento tradizionale. Ciò consente, ipoteticamente, di creare dei modelli generici di partenza che possono essere addestrati sul componente specifico fornendo solo alcuni dati ottenuti da semplici misure sperimentali o simulative. A differenza di precedenti studi come [3], si sono visti due possibili modi di applicare tale tecnica, entrambi molto efficaci. Il metodo basato sul riaddestramento si è rivelato il più accurato, tuttavia c'è da tenere conto della potenza di calcolo necessaria per riaddestrare tutti i neuroni. In questo lavoro raramente si sono superati i 1000 parametri per le reti neurali, perciò l'addestramento richiedeva pochi minuti. Tuttavia, nel caso di modelli molto più complessi che richiedono diverse ore di addestramento e potenze di calcolo non indifferenti, questo approccio può risultare svantaggioso. Invece, il metodo basato sulla rimozione e aggiunta (o direttamente l'aggiunta) di nuovi layer consente di raggiungere ottimi risultati con uno sforzo limitato. Infatti, occorre solamente addestrare i neuroni aggiunti che, se sono molti meno di quelli preesistenti, si può avere un notevole risparmio di tempo e risorse rispetto all'altro metodo. Questo approccio ha anche il vantaggio di poter specializzare il modello su un problema in cui il numero di uscite è diverso dal modello preaddestrato.

L'efficacia dell'utilizzo dell'intelligenza artificiale nel settore dell'elettronica di potenza è già stata dimostrata in molti studi e questa tesi ha dimostrato che la sua applicazione può essere resa più efficiente in termini di richiesta dati, tempo e risorse di calcolo utilizzando modelli preaddestrati e un limitato numero di misure. Il caso di studio è stato un inverter trifase ma tale approccio può essere applicato idealmente a qualsiasi sistema.

Bibliografia

[1] Gerald Francis, Rolando Burgos, Dushan Boroyevich, Fred Wang e Kamiar Karimi. «An algorithm and implementation system for measuring impedance in the D-Q domain». In: *2011 IEEE Energy Conversion Congress and Exposition*. 2011 IEEE Energy Conversion Congress and Exposition (ECCE). Phoenix, AZ, USA: IEEE, set. 2011, pp. 3221–3228. ISBN: 978-1-4577-0542-7. DOI: 10.1109/ECCE.2011.6064203. URL: http://ieeexplore.ieee.org/document/6064203/ (visitato il 13/03/2024).

[2] Lennart Harnefors e Stefan Lundberg. «Input-Admittance Calculation and Shaping for Controlled Voltage-Source Converters». en. In: *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS* 54.6 (2007).

[3] Yufei Li et al. *Machine Learning at the Grid Edge: Data-Driven Impedance Models for Model-Free Inverters*. en. preprint. Ago. 2023. DOI: 10.36227/techrxiv.22137905.v2. URL: https://www.techrxiv.org/doi/full/10.36227/techrxiv.22137905.v2 (visitato il 08/02/2024).

[4] Lorenzo Mastronardi. *Stability analysis of a grid-connected STATCOM using the impedance-based method*. 2023. URL: https://hdl.handle.net/20.500.12608/61059.

[5] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[6] Jian Sun. «Impedance-Based Stability Criterion for Grid-Connected Inverters». en. In: *IEEE Transactions on Power Electronics* 26.11 (nov. 2011), pp. 3075–3078. ISSN: 0885-8993. DOI: 10.1109/TPEL.2011.2136439. URL: http://ieeexplore.ieee.org/document/5741855/ (visitato il 08/02/2024).

[7] Andrew Wunderlich e Enrico Santi. «Digital Twin Models of Power Electronic Converters Using Dynamic Neural Networks». en. In: *2021 IEEE Applied Power Electronics Conference and Exposition (APEC)*. Phoenix, AZ, USA: IEEE, giu. 2021, pp. 2369–2376. ISBN: 978-1-72818-949-9. DOI: 10.1109/APEC42165.2021.9487201. URL: https://ieeexplore.ieee.org/document/9487201/ (visitato il 08/02/2024).