POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea Magistrale

Politecnico
di Torino

Predizione di campi fluidodinamici mediante approcci basati sul Machine Learning

Relatori

Prof. Iuso Gaetano

Ing. Cafiero Gioacchino

Candidato

Picciolo Michele

Aprile 2022

A

Nastasi Maria

Picciolo Giovanni

Ullo Giovanni

Abstract

This thesis work it is aimed to exploit the most recent technological discoveries to make a prediction about the behavior of turbulent flows. Proper Orthogonal Decomposition allows to obtain a decomposition of a fluid dynamic field as a superposition of modes. This decomposition is fed to a neural network previously trained to remember long-term dependencies called "Long Short-Term Memory" (LSTM), to predict the behavior of the flow field in subsequent time intervals. The behavior of LSTM neural networks on canonical flow will be analyzed in depth, in order to highlight their peculiarities and limitations. The aim is to obtain full awareness of the behavior of these networks, in order to gain an ample control, to use them in more complex environments. Such knowledge allows to predict the behavior of quantities of interest in the field (for example pressure), or even to enrich and improve the temporal resolution of experimental measurements with limited acquisition frequencies. The potential of the technique opens the path to future developments where it can be implemented to rapidly study the behavior of the flow field when subjected to active control, in order to minimize the cost in terms of experiments and at the same time guarantee exploration of a large parametric space.

Sommario

In questo lavoro di tesi si propone di sfruttare le più recenti scoperte tecnologiche per effettuare una predizione del comportamento di flussi turbolenti. La Proper Orthogonal Decomposition permette di ottenere una decomposizione di un campo fluidodinamico come sovrapposizione di modi. Tale decomposizione viene fornita ad una rete neurale precedentemente addestrata a ricordare dipendenze a lungo termine denominata "Long Short-Term Memory" (LSTM), per predirre il comportamento del campo di moto in istanti di tempo successivi. Si andrà ad analizzare a fondo il comportamento delle reti neurali LSTM su campi relativamente semplici,al fine di evidenziarne le peculiarità e gli eventuali limiti. Lo scopo è ottenere una piena consapevolezza del comportamento di queste reti, in modo da ottenere una buona padronanza delle stesse ed utilizzarle in campi fluidodinamici più complessi. Tale conoscenza consente di predirre il comportamento di grandezze di interesse del campo (ad esempio la pressione), o anche per arricchiare migliorare la risoluzione temporale di misure sperimentali con frequenze di acquisizione limitate. Le potenzialità della tecnica aprono il campo a sviluppi futuri dove essa può essere implementata per studiare in maniera rapida il comportamento del campo di moto quando soggetto al controllo attivo, al fine di minimizzare il costo in termini di esperimenti e garantendo allo stesso tempo l'esplorazione di un ampio spazio parametrico.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Nonostante sia oggetto di studi da ormai più di un secolo, il campo della fluidodinamica ad oggi mostra ancora molte zone buie ed inesplorate oltre a presentare fenomenologie non ancora del tutto comprese. Questa problematica è sicuramente riconducibile alla considerevole complessità del comportamento del fluido quando viene posto in movimento, il quale origina dei moti che possono essere sia laminari e sia turbolenti.In particolar modo i flussi turbolenti sono oggetto di studi da decenni: a causa della loro distribuzione su varie scale, dalla più piccola (scala di Kolmogorov) alle macroscale integrali, presentano un comportamento difficilmente prevedibile e delle dinamiche non ancora del tutto ben comprese. A cento anni dal primo volo dei fratelli Wright sono stati compiuti passi da gigante in questa direzione: ad oggi è possibile risolvere, seppur in maniera approssimata, le equazioni di governo di Navier-Stokes le quali hanno consentito nel tempo lo sviluppo di codici numerici di fluidodinamica computazionale per riprodurre virtualmente, grazie ai più moderni calcolatori, ciò che si verificherebbe nelle condizioni al vero, permettendo così di conoscere a priori il comportamento del fluido in determinate condizioni.

Grazie all'ausilio delle più recenti unità di calcolo è possibile effettuare anche simulazioni che prevedano la risoluzione diretta delle equazioni di Navier-Stokes, chiamate *Direct Numerical Simulation* o *DNS*. Tuttavia esse risultano essere eccessivamente onerose in termini di costi computazionali, poiché devono essere risolte tutte le scale di moto, a partire dalla scala di Kolmogorov ν fino alla scala integrale L.

A differenza di simulazioni effettuate utilizzando dei modelli (vedasi le simulazioni RANS o LES), la simulazione DNS non prevede alcuna approssimazione se non le sole strettamente necessarie alla discretizzazione della griglia spaziale: la maggior correttezza dei campi, però, viene pagata in termini di potenza computazionale richiesta, che è proporzionale alla velocità del fluido e alla grandezza del campo di moto che si vuole studiare. Pope ha stimato che per una simulazione DNS di un

campo turbolento ed isotropo il costo computazionale sia pari a Re_L^3.[1]

Inoltre ammettendo anche di disporre di una potenza di calcolo sufficientemente adeguata al campo di moto da analizzare, ci si troverebbe in un secondo momento con un'immensa quantità di dati, i quali sarebbero pressoché impossibili da analizzare al fine di un'immediata comprensione del fenomeno, sarebbe invece opportuno sviluppare un modello che, a partire dai dati di input, possa fornire un campo semplificato ma comunque accurato.
Una buona soluzione a queste problematiche può essere rappresentata dai *Modelli ad ordine ridotto* o *ROM*. Qui i dati di partenza vengono riorganizzati, scartando i meno significativi in termini energetici, in modo da ottenere una diminuzione dei gradi di libertà del problema originale:. Questo permette di catturare le strutture principali del campo di moto con un costo computazionale ridotto, pur sempre mantenendo una descrizione accurata della dinamica del sistema; inoltre i risultati ottenuti sono subito leggibili e forniscono un mezzo con il quale è possibile effettuare una prima indagine quantitativa, potenzialmente utile come base per analisi più approfondite o per l'implementazione di analisi fluidodinamiche computazionali citate poc'anzi. Ovviamente l'accuratezza di questi modelli diminuisce con l'aumentare del numero di gradi di libertà che vengono trascurati ma dipende anche dalle proprietà di convergenza del ROM adottato.[2]
Lu et al. annoverano tra i metodi ROM più comuni:

- *Center Manifold.*
- *Lyapunov–Schmidt (L-S).*
- *Galerkin.*
- *Modal Synthesis.*
- *Proper Orthogonal Decomposition (POD).*

Per gli scopi di questa tesi magistrale è stato ritenuto opportuno proseguire gli studi con quest'ultimo approccio.

1.1 Cenni sulla POD

Questo approccio fu introdotto per la prima volta per scopi fluidodinamici da Lumley nel 1967, ma era già conosciuto in altri settori come decomposizione di Karhunen-Love o PCA (Principal Components Analysis) [4], e si propone di decomporre un generico campo vettoriale in un set di funzioni deterministiche ortonormali tra loro, chiamate "modi POD". Essi permettono poi la ricostruzione del campo

originale, se si effettua il procedimento inverso, o di un campo ulteriormente semplificato se si effettuano degli opportuni troncamenti nei ranghi delle matrici che si ottengono, ulteriori approfondimenti saranno effettuati nei prossimi paragrafi. I modi POD sono calcolati in modo che si minimizzi l'errore quadratico medio tra il campo originale e la sua approssimazione di ordine ridotto. L'energia associata ai modi POD invece esprime la bontà della ripartizione dell'energia caratteristica del flusso: prendendo ad esempio il caso di un campo vorticoso si parlerà di energia cinetica come enstrofia. [5]

Un esempio di interpretazione dei dati grazie alla POD è stato realizzato da Frouzakis et al., i quali, recuperati dei dati riguardanti una fiamma a diffusione idrogeno/aria a getto contrapposto ottenuti mediante DNS, hanno proceduto a calcolare i modi POD scoprendo che al fine di una rappresentazione accurata del sistema, per un range esteso di velocità del flusso, sono necessari soltanto i primi sei modi a fronte di più di 26'000 gradi di libertà. Successivamente hanno sfruttato la possibilità presentata poc'anzi di ricostruire il campo di moto originale utilizzando solo i primi modi selezionati, ottenendo un campo "filtrato" : in particolare è stato ricostruito il campo delle temperature ottenendo un errore massimo percentuale di circa il 5-6 %, il che è un ottimo risultato considerando che il numero di modi utilizzati per la ricostruzione è di 4 ordini di grandezza inferiore ai gradi di libertà.

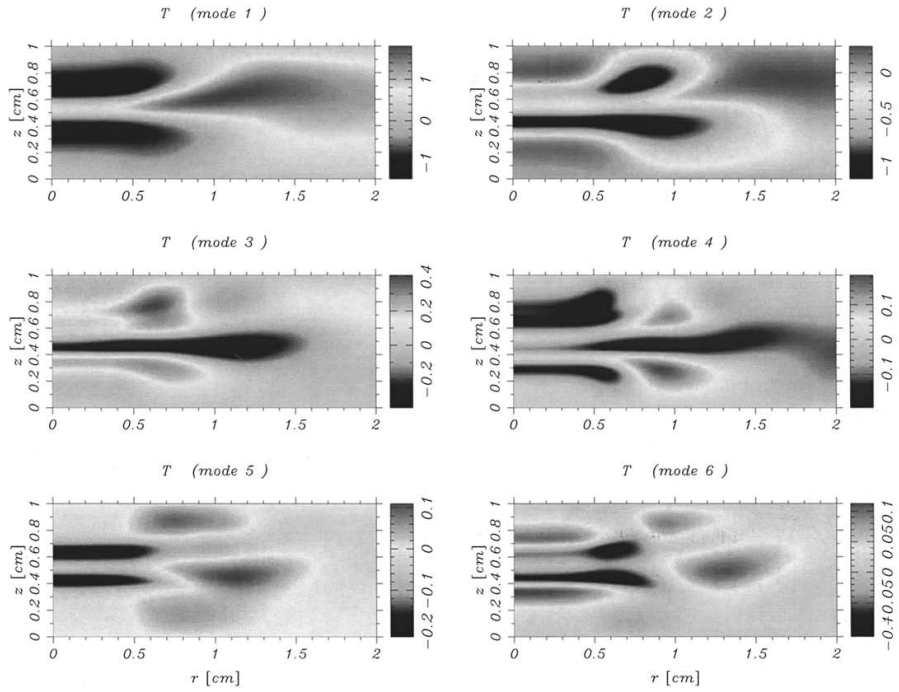


Figure 1.1: I primi sei modi calcolati da Frouzakis et al.

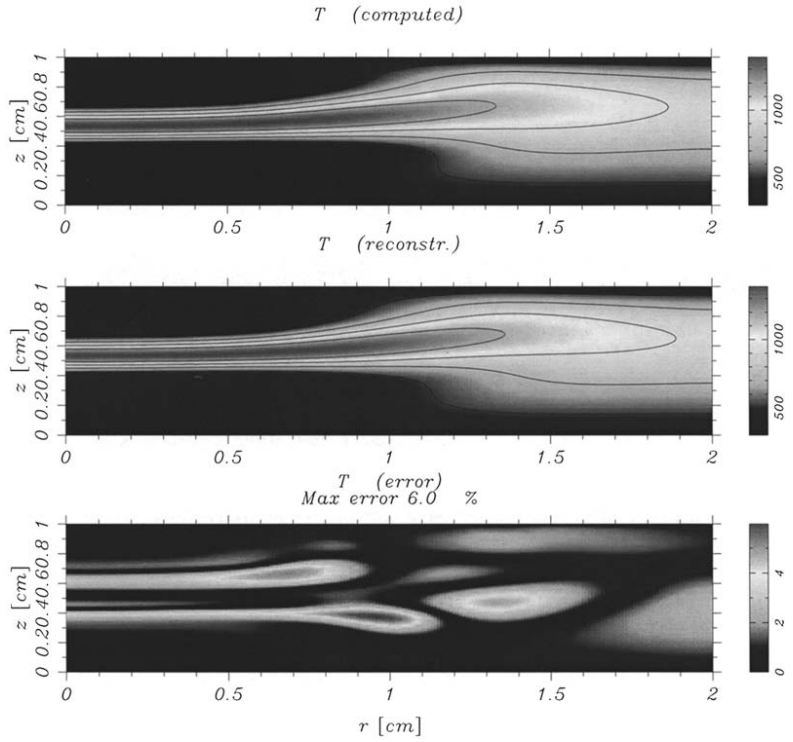


Figure 1.2: Confronto tra i campi di temperatura ottenuti da Frouzakis et al. mediante DNS e ricostruzione POD con i primi 6 modi e relativo errore percentuale.

Inoltre, un'oculata scelta di rango per il troncamento nella POD, consente l'ottenimento di una riduzione di disturbi (identificati come "rumore") presenti a seguito dell'acquisizione di dati sperimentali (problema non presente nel caso di simulazioni effettuate su dati numerici). Infatti il troncamento delle matrici ad un basso rango potrebbe rivelarsi utile al fine di scartare degli effetti indesiderati che si presenterebbero a seguito di acquisizioni (vibrazioni, errori operatore etc.): data la ripartizione delle energie tipica della POD, e considerando che in quanto rumore ad esso è associata sicuramente una bassa energia, poiché la maggior parte sarà dedicata al fenomeno studiato, tipicamente questi effetti vengono rappresentati in modi a basso contenuto energetico. Ad esempio del-Castillo-Negrete et al. hanno focalizzato lo studio su questa possibilità a proposito di simulazioni basate su particelle di interesse per i plasma confinati magneticamente, ottenendo ottimi risultati. Infatti partendo da un campo numerico di test non perturbato, hanno applicato un forte rumore di tipo Gaussiano e successivamente hanno applicato un filtro POD effettuando una ricostruzione con i primi 5 modi, ottenendo una notevole riduzione del rumore, riportando il campo in condizioni molto affini allo stato di partenza.

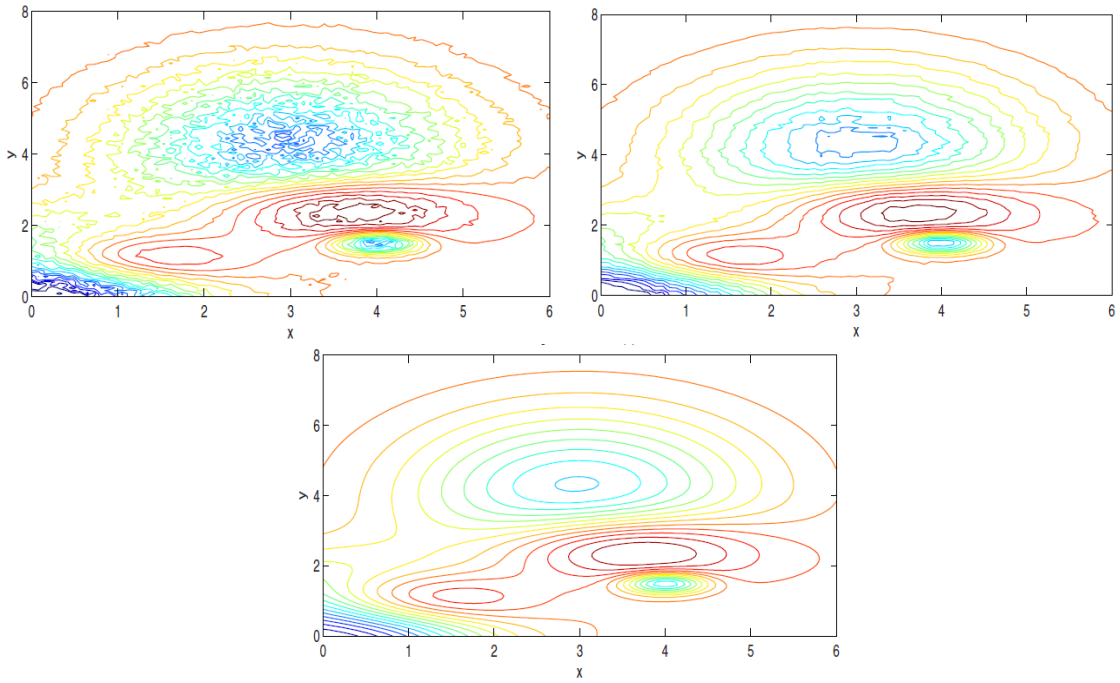


Figure 1.3: In basso: il campo originale. In alto: a sinistra il campo perturbato, a destra il campo ricostruito con 5 modi POD.[7]

I vantaggi della POD sono, però, molteplici: Rahman et al. utilizzando un approccio ibrido, basato su POD ed algoritmi di intelligenza artificiale, sono stati in grado di effettuare una predizione su dei moti quasi stazionari. In particolare, questi algoritmi hanno permesso di predirre l'andamento dei coefficienti modali basandosi soltanto sulla precedente evoluzione temporale degli stessi, permettendo così la ricostruzione dei campi quasi-geostrofici di moti oceanici in istanti di tempo successivi.

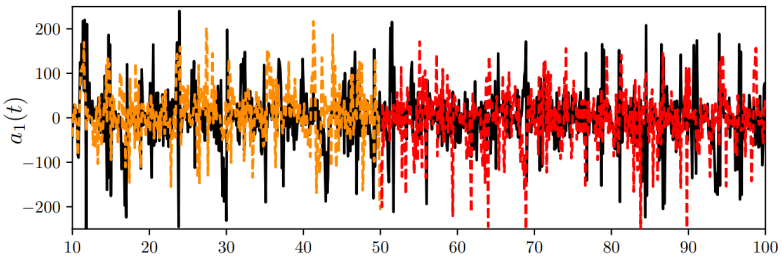


Figure 1.4: In nero l'andamento del primo coefficiente temporale; in giallo la parte di dati utilizzata dall'algoritmo per predirre; in rosso la parte predetta dall'algoritmo.[8]

Sfruttando una combinazione di reti neurali ed algoritmi di deep learning quindi, è possibile effettuare una predizione nel tempo semplicemente basandosi sui coefficienti modali POD calcolati in precedenza.

1.2 Cenni sul machine learning

Il deep learning è un ramo del machine learning che si concentra sull'apprendimento automatizzato del calcolatore: a differenza del machine learning classico dove la predizione è effettuata secondo dei dati "depurati" e algoritmi forniti dall'utente, nel deep learning è possibile fornire grandi dataset di dati che saranno elaborati automaticamente.

Ad esempio nel caso del riconoscimento di un segnale stradale, utilizzando un algoritmo di ML sarebbe necessario estrarre dal segnale stradale le caratteristiche principali (forma,colore, posizioni stradali etc.), successivamente fornirle ad un algoritmo preimpostato per riconoscimento immagini e controllare che il tutto funzioni correttamente; invece un algoritmo di DL, che si avvale dell'utilizzo di reti neurali, è in grado di imparare autonomamente quali features estrarre dal segnale stradale, ed in caso di errore è in grado di rivedere la sua complessa struttura modificando le connessioni all'interno della suddetta rete.

Nell'ultimo decennio si è assistito all'utilizzo sempre più vasto del deep learning con reti neurali sempre più sofisticate, grazie anche alla vertiginosa crescita delle potenze di calcolo e della memoria a disposizione. Basti pensare che un PC avanzato nei primi anni 2000 disponeva di un microprocessore Intel Pentium MMX (frequenza di 233 MHz con velocità bus di 66 MHz) , 64 MB di RAM, un disco rigido da 3,2 GB ed una scheda grafica con 32MB di memoria dedicata, per un costo di oltre 3000 euro odierni. Ad oggi invece il dispositivo utilizzato per allenare la rete neurale utilizzata in questo lavoro di tesi magistrale dispone di una CPU I7-9750H (frequenza di 4,5 GHz con velocità bus di 8GT/s), 16 GB di RAM, un disco rigido da 2 TB ed una scheda grafica con 6 GB di memoria dedicata, per un costo di 1000 euro. Grazie a questi progressi è stato possibile ridurre gli errori commessi dagli algoritmi, riducendo drasticamente i tempi di calcolo e soprattutto potendo implementare il DL . Infatti i due ostacoli principali del DL sono la gestione di grandi quantità di dati, necessari per l'apprendimento della rete, e la potenza computazionale per processare questi dati.

Le reti neurali o *Artificial Neural Networks (ANN)* risultano essere quindi il fulcro del deep learning. Il loro funzionamento si basa sull'idea di riprodurre le reti neurali biologiche, adoperando neuroni artificiali o *nodi* che inviano e ricevono dati tra loro e li utilizzano per apprendere come rispondere correttamente in futuro, emulando il comportamento dei loro corrispettivi biologici. I nodi sono collegati tra loro e organizzati in strati o *layers*, i quali a loro volta sono suddivisi in:

- **Input layer** È lo strato iniziale che accetta i dati di input.
- **Hidden layer** A seconda del tipo di rete neurale implementata si possono avere più strati interni. Ai nodi di questi strati vengono assegnati dei *pesi*, delle costanti *bias* e delle *soglie*: in particolare i primi due vengono utilizzati dalla ANN per determinare quanto influirà il valore di un singolo neurone sui calcoli, mentre gli ultimi rappresentano dei valori che l'output deve oltrepassare per attivare il livello successivo della rete e procedere al passaggio dello stesso, secondo quanto definito dalle *funzioni di attivazione* che regolarizzano l'output. Poiché non sono classificabili nè tra gli input nè tra gli output, questi strati vengono chiamati anche "nascosti". Convenzionalmente, in presenza di 3 o più strati, si parla di *Deep Neural Networks*.
- **Output layer** È lo strato che riceve l'informazione processata dai precedenti strati, in modo da fornire il risultato finale.

Il problema delle reti neurali tradizionali, però, è la mancanza di memoria rispetto alle informazioni transitate precedentemente nello strato di input. Infatti, se questo può non essere rilevante nel caso di input non correlati tra loro, così non è per quanto riguarda le serie temporali o l'andamento di funzioni. Qui ogni dato in ingresso è correlato ai dati precedenti e futuri, palesando la necessità che si tenga conto della "storia temporale", e che i precedenti dati possano andare a fornire un feedback alla rete.

Questo problema è potenzialmente risolvibile grazie all'introduzione delle reti neurali ricorrenti o *Recurrent Neural Network (RNN)*. Il motivo risiede nell'architettura propria delle reti neurali ed il conseguente addestramento: le ANN sono costruite con un'architettura denominata *Feed-Forward Neural Network* dove cioè l'informazione in output non viene sfruttata dagli stessi strati in feedback ma solo fornita agli strati successivi come input.

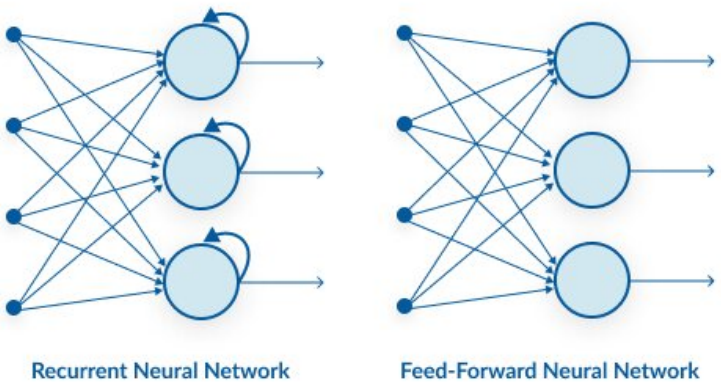


Figure 1.5: Differenza tra le architetture RRN (a sinistra) e FFNN (a destra).[9]

È evidente la differenza tra le due: mentre nella rete FFNN l'informazione viaggia solo in avanti, nelle reti ricorrenti l'informazione processata dagli strati non viene semplicemente portata come output ma si ha una sorta di "retroazione"; in breve l'output all'istante t e funzione dell'input al tempo t ma anche dell'output al tempo *t-1*, cioè dell'output all'istante precedente. Questo ciclo permette alla rete di ricordare le informazioni in modo da poter fronteggiare situazioni in cui la memoria è fondamentale.

Un particolare tipo di RNN è la rete *LSTM* o *Long Short-Term Memory*. Presentata per la prima volta da Hochreiter and Schmidhuber, essa si propone di risolvere alcuni problemi tipici come il *"vanishing or exploding gradient"* che affligge le reti RNN durante l'addestramento, come evidenziato da Bengio et al.(questo argomento verrà approfondito nel capitolo dedicato).

In particolare rispetto all'architettura RNN tradizionale, si aggiunge un "forget gate" (letteralmente "cancello dell'oblio"), che aiuta la rete a decidere se tenere in memoria dei dati ritenuti utili al fine dei successivi output, secondo quanto appreso durante l'algoritmo di allenamento, oppure se scartarli se non ritenuti necessari. Per questo motivo le reti LSTM sono molto utilizzate per la previsione di serie temporali o, nel nostro caso, degli andamenti di una funzione.

Di seguito si andrà ad analizzare in dettaglio la POD, spiegandone i principi matematici, e nel seguente capitolo si andranno ad analizzare nel dettaglio le principali caratteristiche delle reti LSTM. Infine gli ultimi due capitoli prevederanno la preparazione dei codici necessari alla presentazione dei risultati, presenti nell'ultimo capitolo che rappresenta il fulcro di questo lavoro di tesi magistrale.

Capitolo 2

I modelli di ordine ridotto: la POD

Come già riportato nel capitolo 1, fu introdotta per la prima volta per scopi fluidodinamici da Lumley nel 1967, nel tentativo di semplificare il campo randomico della turbolenza introducendo delle funzioni, ciascuna delle quali che potesse rappresentare una porzione dell'energia dello spettro del fluido ed aiutare a visualizzare delle strutture organizzate che sarebbero potute sfuggire nel caos dei moti turbolenti. [5]

Vi è la necessità di fornire un opportuno numero di snapshot perché il dataset dev'essere rappresentativo della dinamica del problema da rappresentare. Queste funzioni sono anche denominate *Modi POD* e ciascuno di questi rappresenta una porzione dell'energia cinetica fluttuante del flusso.

2.1 L'algoritmo POD

I modi POD sono organizzati in ordine decrescente, dove il primo è quello con il contenuto energetico più alto e gli ultimi invece rappresentano soltanto una minima parte dell'energia. Proprio a causa dell'infima energia ad essi associata, solitamente questi ultimi modi vengono scartati poiché non caratteristici del flusso osservato ma rappresentanti "rumore" energetico. In particolare possono essere disturbi legati agli strumenti di acquisizione o errori dovuti nel processare i dati, la loro eliminazione può apportare effetti benefici in termine di ricostruzione (argomento trattato in sezione 2.3) del segnale "pulito". La ricerca è quindi di un sottospazio vettoriale opportuno, affinché le energie rappresentate come valori scalari, se sommate tra loro diano il valore massimo ottenibile in qualsiasi configurazione. L'obiettivo era quello di "depurare" il campo turbolento, rappresentando soltanto le strutture coerenti

tipiche della turbolenza in modo da permettere di ottenere una comprensione qualitativa sulla disposizione del flusso osservato anche in situazioni di campi fortemente complessi.

Per poter sfruttare al meglio la tecnica POD risulta necessario disporre di un numero di *snapshot* opportuno, ove ciò non sia possibile è da preferire un quantitativo minore ma più rappresentativo dell'intero fenomeno piuttosto che avere molte istantanee del fenomeno ma in brevi lassi di tempo: ad esempio, considerando di prendere in esame il fenomeno del *vortex shedding* (ovvero sfilamento di vortici) di un cilindro investito da una corrente con *Re* idoneo, si considererà un intervallo di tempo sufficientemente prolungato affinché vengano catturati gli istanti in cui la corrente, lambendo il cilindro, si distaccherà generando una prima coppia di vortici alternati (chiamata anche *scia di Von Karman*), permettendo quindi di ottenere la dinamica completa; se invece venisse effettuata un'acquisizione, anche a frequenze raddoppiate, senza catturare lo sfilamento di vortici si otterrebbe un'errata ricostruzione del campo di moto.

Supponiamo di rappresentare un campo vettoriale come la velocità $u(x,t)$ di un flusso statisticamente stazionario. La sua componente non stazionaria viene rappresentata come:

$$u' = u - \bar{u} \tag{2.1}$$

dove u rappresenta il campo vettoriale della velocità e $\bar{u}$ rappresenta la sua media nel tempo. È quindi possibile rappresentare u' come la sommatoria di determinate funzioni spaziali $\Phi_k(x)$ moltiplicate "per un peso" temporale $a_k(t)$:

$$u' = \sum_{k=1}^{\infty} (a_k(t)\Phi_k(x)) \tag{2.2}$$

Questa rappresentazione non è univoca, in quanto è possibile trovare infinite combinazioni, la POD invece tende a trovare la combinazione ottimale dei coefficienti contenuti in (2.2) affinché si abbia il numero più basso possibile di modi a patto che venga massimizzata l'energia contenuta nei primi k modi **spaziali**.

Un altro parametro che viene tenuto in considerazione nella decomposizione riportata in 2.2 è quello della ortonormalità tra i modi. Questa proprietà permette di rendere indipendenti tra loro i coefficienti temporali. $a_k(t)$ dipenderà soltanto dal modo $\Phi_k(x)$.

La POD può essere effettuata in vari modi, in questo lavoro di tesi verranno illustrati i tre ritenuti fondamentali ai fini degli studi eseguiti.[13]

2.1.1 Metodo classico o *spaziale*

È possibile fornire vari tipi di campi, sia scalari (es. pressione, temperatura) che vettoriali (es. velocità, vorticità) per discreti intervalli di tempo t con una griglia

spaziale che può avere più dimensioni indicate con n. In generale l'algoritmo POD restituirà sempre, ordinando secondo un criterio di energia discendente, un set di coefficienti modali ortogonali tra loro $\Phi(x)$ con i loro corrispondenti coefficienti temporali $a(t)$ ed i rispettivi contenuti energetici.

In questo metodo sarà necessario preparare gli snapshot m come una matrice in cui le colonne rappresenteranno il valore del campo vettoriale $u(t)$ in un determinato punto (x_k, y_k, z_k) ed ogni colonna risulterà associata ad un determinato istante di tempo $t_1, t_2, ..., t_n$. Indichiamo con i il numero di righe, e j il numero di colonne. Successivamente si andrà ad effettuare una media del campo vettoriale:

$$\bar{u}_i(x) = \frac{\sum_{j=1}^{n}(u_{i,j}(x,t))}{n} \tag{2.3}$$

dove per evitare di rendere la notazione troppo gravosa è stata sostituita la dipendenza delle variabili x, y, z con x. Una volta effettuata la media, sarà necessario sottrarla al campo vettoriale iniziale al fine di ottenere la componente fluttuante dello stesso:

$$u'(x,t) = u(x,t) - \bar{u}(x) \tag{2.4}$$

Ecco che quindi la matrice degli snapshot avrà una forma del tipo:

$$U = [u'_1(t) \quad u'_2(t) \quad u'_3(t) \quad ... \quad u'_n(t)] \in \mathbb{R}^{n\,x\,m} \tag{2.5}$$

Una volta organizzata la matrice U è necessario trovare un vettore di basi ottimali (*proper*) che possano rappresentare al meglio i dati forniti, cioè riscrivendo l'equazione 2.2 come:

$$u(x,t) - \bar{u}(x) = \sum_{k=1}^{m}(a_k(t)\Phi_k(x)) \tag{2.6}$$

si cercano dei vettori $\Phi(x)$ che possano rappresentare il vettore $u(x)$ con il minor numero di modi possibile e l'energia rappresentativa più alta.

Una soluzione può essere quella di risolvere il problema agli autovalori sul sistema:

$$C\phi_j = \lambda_j\phi_j \qquad con \quad \phi_j \in \mathbb{R}^n, \qquad \lambda_1 \geq \lambda_2 ... \geq \lambda_n \geq 0 \tag{2.7}$$

con C matrice di covarianza del vettore $u'(x,t)$. Si definisce matrice di covarianza la matrice tale che:

$$C = \sum_{i=1}^{m} u'(t_i)\, u'^{T}(t_i) \; = \frac{UU^T}{m} \qquad \in \mathbb{R}^{n\,x\,n} \tag{2.8}$$

Gli autovettori calcolati in 2.7 risultano essere i modi POD, mentre gli autovalori trovati esprimono l'energia catturata dai modi POD corrispondenti.

2.1.2 Metodo SVD applicato direttamente alla matrice degli snapshot

La *Singular Value Decomposition* o *SVD* è una tecnica di fattorizzazione di matrici rettangolari: in particolare si pone l'obiettivo di diagonalizzare una matrice, previa pre e post moltiplicazione per delle opportune matrici. A differenza della scomposizione agli autovalori, la quale può essere applicata solo a certe matrici quadrate , la SVD può essere applicata a qualunque matrice rettangolare. Si può anche dimostrare come le due tecniche siano legate tra loro, ed in particolare che, se viene data una matrice A $\in \mathbb{R}^{n\,x\,m}$ hermitiana (cioè che coincide con la propria trasposta), allora il quadrato dei *singular values* coincide con gli autovalori di A.

La dimostrazione non viene riportata poiché esula dallo scopo di questa tesi, ma si può dimostrare che data una matrice A $\in \mathbb{R}^{r\,x\,n}$ $\exists$ {U $\in \mathbb{R}^{n\,x\,n}$; V $\in \mathbb{R}^{m\,x\,m}$ } tale che:

$$A = U\Sigma V^T \tag{2.9}$$

La matrice Σ avrà le stesse dimensioni di A, ed essendo stata diagonalizzata conterrà tutti valori nulli nelle posizioni extra-diagonale, mentre lungo la diagonale principale si andranno a disporre i valori positivi σ_i con i=1,2,...,r (rango di troncamento), i quali sono detti *singular values* di A. Lungo le colonne di U e V invece si disporranno i cosiddetti *singular vectors* rispettivamente sinistro e destro, i quali corrispondono agli autovettori presentati in eq.2.7. La SVD si dice "unica" se per ogni *singular value* σ è possibile trovare un solo *singular vector*, altrimenti la SVD è detta *degenere*.
Si noti come le matrici U e V siano ortonormali tra loro, cioé viene soddisfatta la seguente relazione:

$$UU^T = U^TU = I \tag{2.10}$$

$$U^T = U^{-1} \tag{2.11}$$

Le relazioni 2.10 e 2.11 si applicano anche alla matrice V. Le matrici U e V rappresentano rispettivamente i *singular vector* sinistro e destro della matrice A, mentre Σ è una matrice diagonale i cui elementi non sono altro che i *singular values* σ_i con i = 1,2, ..., m, $\sigma_i > 0$ della matrice A.

2.1.3 Metodo degli Snapshot[14]

Il primo metodo presentato diventa inutilizzabile nel caso si disponga di una matrice degli snapshot composta da un dataset molto vasto. Questo perché la matrice di covarianza ha dimensioni *n x n*. Questo problema è facilmente valicabile grazie alle

proprietà della POD. Più nello specifico, si ricorda come i coefficienti modali siano indipendenti tra loro, ma soprattutto che il campo vettoriale possa essere ricostruito sia sfruttando i coefficienti modali ϕ e sia sfruttando i coefficienti temporali a_k effettuando un'opportuna moltiplicazione per la matrice Ψ.

La matrice degli snapshot $U(t_i)$ viene troncata in modo da mantenere solo una parte del dataset iniziale, secondo il parametro "r" indicato anche come *rango della matrice degli snapshot*. Secondo questa procedura si avranno degli indici come: $i = t_1, t_2, t_3, ..., t_r$ con $r << n$.

La scelta del rango della matrice non è banale, poiché non esiste un metodo universale. Convenzionalmente viene considerato un rango abbastanza piccolo da permettere di ridurre il costo computazionale del problema agli autovalori ma anche abbastanza elevato al fine di permettere la ricostuzione delle macrostrutture più importanti nel campo vettoriale esaminato. Volendo essere ancora più precisi solitamente viene scelto in modo che l'energia catturata dai modi POD, calcolati sulla matrice troncata, sia di circa il 90% mentre in letteratura è possibile trovare applicazioni fino al 99%. Il problema agli autovalori da risolvere diventa quindi:

$$U^T(t_i)\, U(t) \Psi_j = \lambda_j \Psi_j, U \in \mathbb{R}^{(m\, x\, m)} \tag{2.12}$$

È bene notare come, a differenza dell'approccio classico, a seguito della risoluzione del problema agli autovalori non otteniamo direttamente i modi POD ϕ bensì la matrice Ψ, lungo le cui colonne si troveranno i relativi autovettori. Essi risultano essere correlati ai modi POD (ovvero gli autovettori della matrice di covarianza) secondo la seguente relazione:

$$\phi_j = U\, \Psi_j \frac{1}{\sqrt{\lambda_j}}\ conj = 1{,}2, ..., m \tag{2.13}$$

2.2 La matrice dei dati

Per far si che i vari algoritmi POD appena presentati possano generare dei risultati sensati è necessario che la matrice,la quale contiene i dati del campo vettoriale studiato, venga opportunamente "preparata" disponendo i dati al suo interno secondo una ben precisa distribuzione. Non è obbligatorio che vi sia un *timestep* fissato (cioè che i dati siano equispaziati nel tempo) nella matrice di partenza, a patto che la stessa venga successivamente elaborata in modo da ottenere i dati opportunamente ridimensionati per risiedere in una matrice uniforme.

Bisogna organizzare la matrice in modo da generare delle colonne di dati ognuna relativa ad un istante temporale. Nel caso di dati a più dimensioni, la matrice degli snapshot manterrà sempre la stessa conformazione con un passaggio di *stacking* o "aggancio" delle coordinate spaziali. Andando più nello specifico la matrice degli

snapshot, ad esempio nel caso 3D, vedrà ogni colonna con le coordinate lungo le tre direzioni impilate come indicato in figura 2.1. Successivamente per visualizzare i campi verrà effettuata un'operazione di *unstacking* ripristinando e dividendo le tre dimensioni.

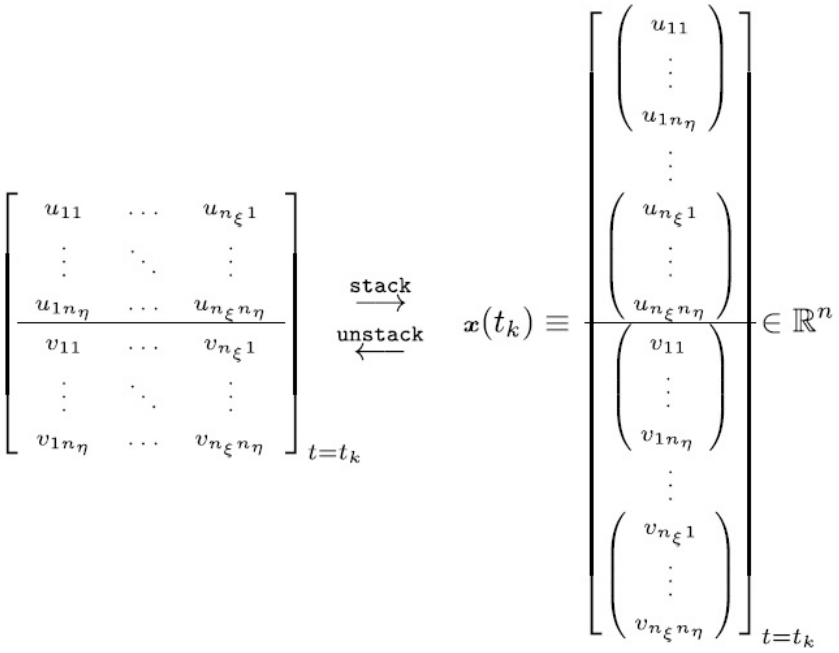


Figure 2.1: Operazione di *stacking* ed *unstacking* per una matrice di velocità 2D (u,v) con direzioni principali (ξ,η).[13]

2.3 Il troncamento e la successiva ricostruzione

Ogni rilevamento sperimentale dei dati porta con sè un certo errore rispetto alla misura reale dovuto anche alla precisione finita della catena di acquisizione (ad esempio il fondo scala nei trasduttori o l'angolazione di alcune particolari sonde); inoltre in caso di acquisizioni con strumentazione elettronica non di rado si acquisisce il segnale della misura ma anche altri segnali "spuri" come ad esempio disturbi elettronici dovuti alla rete elettrica che alimenta la catena di misura o, nel caso di rilevamenti fluidodinamici, dei flussi esterni al fenomeno studiato. Tutto questo si traduce in un disturbo del segnale acquisito, identificato come *rumore*. È possibile effettuare un troncamento delle matrici ottenute a seguito della diagonalizzazione presentata in 2.1.2 del dataset, al fine di semplificare il campo vettoriale. A seguito di questa operazione, in caso di acquisizioni sperimentali, vi è un secondo effetto non

trascurabile: come mostrato nel capitolo introduttivo, il rumore viene attenuato (ed in alcuni casi quasi del tutto eliminato) se si effettua una scelta oculata del rango di troncamento.
È possibile ricostruire il campo di moto in seguito al troncamento seguendo due vie:

- **Ricostruzione mediante matrici**: Questo metodo è utilizzabile a seguito dell'applicazione dei metodi basati sulla decomposizione SVD descritti nelle rispettive sezioni 2.1.2 e 2.1.3. È il metodo più rapido tra i due ed è utilizzato soprattutto nel caso non vi sia necessità di effettuare predizioni o, in generale, di riferirsi ai coefficienti temporali a. Infatti esso prevede che venga effettuata la decomposizione SVD descritta in equazione 2.9. Successivamente le matrici ottenute verranno troncate a seconda di quale sia stato scelto come rango delle matrici, ed infine è possibile ricostruire il campo vettoriale semplificato, semplicemente effettuando una ri-moltiplicazione tra le matrici:

  $$A = U_r \cdot \Sigma_r \cdot V_r^T \tag{2.14}$$

  con il pedice "r" si identificano le matrici troncate.

- **Ricostruzione mediante coefficienti temporali** a_k: Principalmente questo metodo viene utilizzato a seguito dell'applicazione del metodo degli snapshot descritto in sezione 2.1.3 ma anche nel caso di approccio ibrido POD-LSTM poiché esso si basa sui coefficienti temporali predetti dalla stessa rete. Infatti una volta eseguita la *Singular Value Decomposition* ed ottenuti i modi pod ϕ secondo la formula 2.13 è possibile ottenere i coefficienti temporali effettuando la moltiplicazione tra la matrice degli snapshot trasposta A^T e la matrice dei modi POD ϕ:

  $$a \; = \; A^T \cdot \phi$$

  Una volta effettuate le operazioni relative alla rete LSTM è possibile ricostruire il campo di moto sfruttando la proprietà dei coefficienti temporali secondo cui

  $$u'_{rec} = \tilde{u} + A_{rec} \cdot a_{pred}.$$

Capitolo 3

Machine learning: RNN ed LSTM

A seguito dell'esposizione dettagliata riguardo la POD nel capitolo 2 risulta chiaro come avvenga la decomposizione di un generico campo vettoriale e come vengano generati i coefficienti modali ϕ_k ed i coefficienti temporali a_k, utilizzati per la ricostruzione del campo vettoriale come spiegato nei rispettivi paragrafi 2.1.2 e 2.1.3. Tuttavia al fine di avere una perfetta comprensione delle prove e dei risultati svolti in questa tesi di laurea è necessario che vengano esposti dei concetti chiave riguardanti il machine learning come preannunciato nel capitolo 1. Infatti nei successivi capitoli verrà introdotta la ricostruzione dei coefficienti temporali a_k mediante algoritmi di DL ed in particolare grazie ad una rete neurale LSTM.

3.1 Recurrent Neural Networks

Come presentato nel capitolo 1 le RNN riescono a risolvere il problema della mancanza di memoria delle normali reti neurali feed-forward. Questa persistenza dell'informazione avviene grazie alla presenza di un loop che permette all'output di essere influenzato sia dal valore di input che dal valore dell'output stesso all'istante di tempo precedente. Questo concetto può sembrare ostico e misterioso (cosa avviene durante il loop?) ma è chiarificatrice la figura 3.1. Osservandola infatti si nota come in realtà il flusso dell'informazione possa essere ricondotto ad una normale rete *feed-forward*.
L'output del generico neurone h al tempo t (O_t) è funzione sia dell'input al tempo t (x_t) che di O_{t-1} e così via. In teoria quindi la RNN dovrebbe essere in grado di ricordare perfettamente ogni dato necessario anche per lunghi istanti di tempo, data la dipendenza da ogni output.

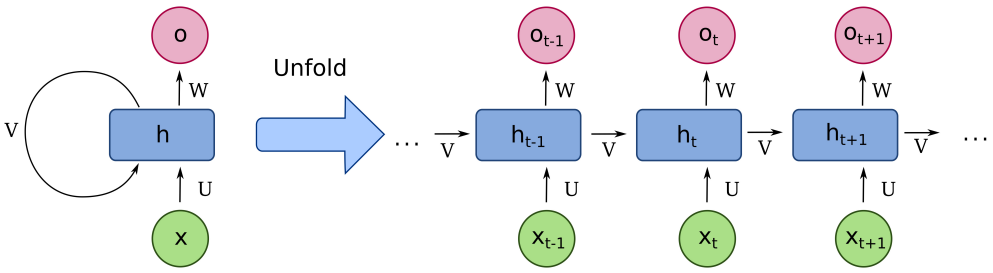


Figure 3.1: L'architettura base di una rete RNN.[15]

3.1.1 Exploding or Vanishing Gradient

La teoria però è ben diversa dalla pratica, infatti provando ad utilizzare un'architettura classica RNN la rete riuscirà a fornire il giusto output, nel caso di predizioni dove le informazioni necessarie sono contenute in istanti di tempo relativamente brevi, ma per dipendenze molto lunghe inizirebbero a sorgere delle complicazioni. Per esempio se utilizzassimo una rete RNN per predirre l'ultima parola in una frase molto semplice come "È nuvoloso, sta per *piovere*" probabilmente una rete ben allenata riuscirebbe a dare come output la parola corretta. Tuttavia se si provasse a predirre l'ultima parola della frase "La Mole Antonelliana, opera dell'architetto Alessandro Antonelli, è un edificio monumentale della città di *Torino*", la rete restituirebbe un output errato poichè si andrebbe sicuramente incontro al fenomeno del cosiddetto *exploding or vanishing gradient*. Esso è una caratteristica non voluta dello schema di allenamento *backpropagation*, il quale, una volta rilevata la differenza tra valore desiderato e valore di output effettivo, modifica i valori dei pesi partendo dall'ultimo strato ed arrivando al primo.
Il tipo di problema che si presenta dipende dalle funzioni di attivazione, si parla di:

- **Vanishing gradient** se vengono usate funzioni di attivazione non lineari (es. tangente iperbolica). Il loro gradiente infatti è compreso tra 0 e 1, risulta palese come la moltiplicazione di molti numeri in questo intervallo possa portare il valore finale ad avvicinarsi rapidamente allo zero,andando incontro ad un fenomeno di *evanescenza* del gradiente stesso.

- **Exploding gradient** se vengono usate funzioni lineari (es. rettificatore), poiché il valore del gradiente può essere superiore all'unità, portando rapidamente il gradiente a crescere andando incontro ad un fenomeno di *esplosione* dello stesso. [16]

3.1.2 Addestramenti

L'addestramento rappresenta la parte più importante e delicata del processo di utilizzo delle reti LSTM e più in generale delle reti neurali. Infatti è qui che la rete modifica i propri parametri in modo da essere capace di poter fornire una risposta corretta in seguito a determinati input ricevuti in ingresso. Un'errata impostazione dei training provocherebbe un'inservibilità delle reti con conseguente spreco di tempo ed energia, poiché si renderebbe necessario un ulteriore addestramento.

L'addestramento vero e proprio ha inizio con l'inizializzazione di tutti i pesi a valori casuali, a seguire viene poi scelto l'approccio seguendo due vie maestre:

- Addestramento senza supervisione: Non viene fornito alcun feedback alla rete sul suo operato, motivo per cui è essa stessa che valuta cosa imparare sulla base della "correttezza" degli output restituiti. Sebbene possa sembrare inutile sotto certi aspetti, questa tecnica potenzialmente consente di ottenere una rete addestrabile "infinite volte" su tanti campi diversi riguardanti input mai incontrati prima. Tuttavia ad oggi non è ancora sufficientemente affidabile per adoperarla su di una rete neurale, infatti la maggior parte degli algoritmi prevede un *addestramento con supervisione*.

- Addestramento con supervisione: Si forniscono alla rete i valori di output desiderati, la quale calcolerà l'errore rispetto agli output restituiti. Lo stesso errore sarà utilizzato per modificare i pesi e i bias (delle costanti) della rete secondo lo schema di *backpropagation*. Al termine di questa operazione viene ripetuto l'intero ciclo, denominato *epoca*. Ad ogni epoca la rete riesce ad affinare sempre più il suo grado di accuratezza, finchè non si raggiunge il numero massimo di epoche o l'operatore termina l'addestramento. Al termine del training molto spesso viene svolta una parte di "test" con una porzione dei dati del training che viene utilizzata per effettuare una predizione di prova utilizzando la rete appena addestrata. Si noti come sia possibile che la rete non riesca ad arrivare a convergenza, il che può dipendere da un dataset in input non correlato agli output desiderati, ad un numero insufficienti di dati in ingresso oppure ad un'architettura della rete neurale da rivedere. Data la sua comprovata affidabilità è l'algoritmo di addestramento utilizzato in questo lavoro di tesi.

3.2 Long Short Term Memory

La rete LSTM o *Long Short-Term Memory*, si propone di risolvere i problemi appena descritti grazie all'introduzione di un nuovo tipo di neurone, molto più complesso del semplice percettrone. Nel 1997 Hochreiter and Schmidhuber la progettarono

con lo scopo ben preciso di evitare l'insorgere di fenomeni che potessero risultare d'intralcio nel ricordare dati a lungo termine

La peculiarità del neurone LSTM è la presenza di diversi cancelli o *gate* presenti all'interno dello stesso, il cui scopo è quello di far persistere le informazioni senza però andarle a cercare all'interno della rete, ma trattenendole secondo quanto appreso durante la fase di allenamento.

- **Forget gate** È il primo cancello che incontra l'informazione in arrivo nonché il più importante: qui infatti viene presa una decisione sul mantenere l'informazione in memoria oppure dimenticarla.Esso riceve l'informazione al tempo precedente $h_(t-1)$ (come feedback) e l'informazione al tempo t. Ad esse viene applicata una funzione di attivazione, la quale verrà moltiplicata per l'output precedente. Se il risultato è prossimo allo 0 allora dimenticherà, viceversa verrà ricordato l'output precedente.
- **Input gate** Riceve anch'esso l'informazione al tempo precedente h(t-1) (come feedback) e l'informazione al tempo t e decide se possono essere elaborati insieme o meno. Il suo output andrà ad incontrarsi con l'uscita del forget gate nel seguente cancello.
- **Input modulation gate o Cell State** Questo gate serve a trasportare l'informazione, verrà approfondito nel paragrafo 3.2.1.
- **Output gate** Sempre in base agli ingressi forniti qui si decide se lo stato attuale può essere fornito come output, nonché retroazione per il prossimo time step.

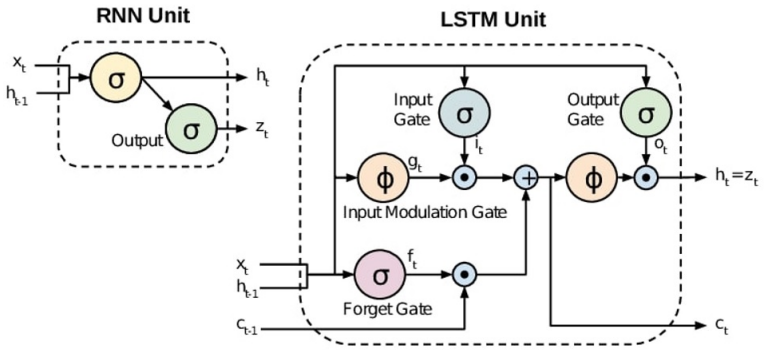


Figure 3.2: La differenza tra una generica unità RNN ed un neurone LSTM.[16]

3.2.1 Il neurone LSTM

È possibile esaminare ancor più nello specifico il neurone LSTM, in particolare andando a capire come funzionano i *gate* e perché risolvono il problema del gradiente.

Sicuramente una delle chiavi fondamentali del successo di questa rete è rappresentata dal cosiddetto "stato di cella" o *cell state*: infatti, come in una catena di montaggio, l'informazione in input scorre indisturbata (tranne in alcuni punti) lungo ciò che può essere identificato come una linea retta che attraversa il neurone da una parte all'altra. L'input però può ricevere un'aggiunta o rimozione di informazioni controllata dai *gate* precedentemente introdotti in .

I *gate* vengono rappresentati da funzioni di tipo sigmoide, le quali simulano il comportamento fisico di un cancello: se l'output è 0 il cancello resta chiuso e non si ha passaggio di informazioni, se invece è pari ad 1 il cancello è aperto e permette il passaggio di qualunque informazione attraverso esso. Se invece si ha un output pari ad un valore compreso tra i due estremi allora si ha un passaggio parziale delle informazioni. È evidente quindi come i gate fungano anche da "protezione" nei confronti dei *cell state*. Al fine di comprendere meglio la parte seguente, la quale approfondirà il funzionamento dei layer, si ritiene utile introdurre la seguente terminologia:

- f_t funzione attivazione forget gate layer;
- σ funzione sigmoide $\frac{1}{1+e^{-x}}$;
- W Matrice pesi (con pedici relativi al rispettivo layer);
- x_t vettore input fornito al layer LSTM;
- h_{t-1} vettore hidden state al tempo t=t-1;
- b vettore dei bias (con pedici relativi al rispettivo layer);
- i_t cell state activation vector;
- C_t Stato di cella al tempo t;
- C_{t-1} Stato di cella al tempo t=t-1;
- o_t Output gate activation layer;
- h_{t-1} vettore hidden state al tempo t;

Forget Gate layer Questo layer è utilizzato per decidere quali informazioni debbano essere mantenute nel *cell state* e quali invece non siano più necessarie e possano essere dimenticate. Infatti il cancello sigmoide, ricevendo in input h_{t-1} e x_t deciderà singolarmente per ogni numero presente nel *cell state* C_{t-1} generando un output compreso tra 0 (l'informazione va dimenticata) e 1 (l'informazione va mantenuta). Si noti come in questo step venga deciso cosa

fare (dimenticare o ricordare l'informazione) ma non venga ancora eseguita l'azione!

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

Input Gate Layer Successivamente verrà presa una decisione riguardo a quali informazioni in input sia necessario memorizzare nel *cell state*. Questa operazione in realtà si articola in due parti: la prima verrà svolta da questo layer, rappresentato anch'esso da un cancello sigmoide, che deciderà in base ai valori di output (coerentemente con i valori esposti precedentemente) se ricordare o meno determinate informazioni; la seconda parte invece verrà svolta da un layer con funzione di attivazione a *tangente iperbolica tanh* la quale creerà un vettore con le possibili combinazioni di informazioni da salvare. Le informazioni dei due layer si andranno poi a combinare in modo da creare un aggiornamento da inoltrare sul *cell state*.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ C_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned}$$

Passo di update A questo punto si effettua l'aggiornamento vero e proprio dello stato di cella C_{t-1} che, in base alle operazioni precedentemente svolte, diventerà lo stato di cella C_t secondo la seguente equazione: (in breve ft è quanto vogliamo dimenticare, it quanto vogliamo ricordare del nuovo)

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{3.2}$$

Output Gate layer Questo layer rappresenta l'ultimo step: qui viene elaborato quello che poi sarà l'output restituito dal neurone. Infatti non verrà restituito direttamente C_t ma verranno svolte delle pre-operazioni al fine di restituire solo il valore desiderato: viene applicato uno strato sigmoide per decidere quale parte del *cell state* sarà dato come output coerentemente con quanto visto finora (con valori sempre compresi tra 0 e 1), ed infine viene applicato uno strato con tangente iperbolica

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t \cdot tanh(C_t) \end{aligned}$$

In letteratura esistono alcune varianti del neurone LSTM o delle versioni semplificate (GRU), le quali però non verranno esaminate poiché esulano dallo scopo di questa tesi.

Capitolo 4

Analisi dei dati e implementazione in MATLAB

Data la complessità delle reti LSTM, soprattutto per quanto concerne la fase di addestramento, si è ritenuto opportuno testarne le capacità su dei problemi con complessità e dinamiche differenti al fine di evidenziarne gli eventuali limiti.

4.1 I dati

Tra i vari casi presi in esame si è scelto di iniziare con dei dati numerici rappresentanti un campo vettoriale di vorticità intorno ad un cilindro a basso numero di Reynolds[1]. Essi rappresentano un buon punto di partenza, in quanto non essendo rilevazioni sperimentali non sono affetti da errori o *rumore*. Inoltre, essendo un dataset non eccessivamente oneroso in termini computazionali, è possibile effettuare la POD secondo la metodologia presentata nella sezione 2.1.2.

Successivamente, dati i risultati promettenti, è stato preso in esame un dataset più complesso quale un getto assial-simmetrico studiato tramite tecnica ottica PIV. Di questo era disponibile una griglia spaziale molto elevata con diverse migliaia di snapshot. Data l'enorme quantità di dati sono state effettuate diverse operazioni (che verranno esplicate ulteriormente nelle sezioni seguenti) per risparmiare spazio. Inoltre è stato necessario adoperare il metodo degli snapshot presentato nella sezione 2.1.3. Questo campo è stato quello più problematico poiché nonostante la

[1]I dati sono stati recuperati dal sito http://dmdbook.com/

grande mole di dati a disposizione, la rete LSTM sembrava non rispondere alle aspettative ed ogni tentativo prevedeva un'attesa non indifferente. Al termine della rispettiva sezione vengono elencati i probabili motivi della nascita di queste difficoltà.

Al fine di confermare le supposizioni effettuate al termine della sezione relativa al getto si è passati ad un dataset sperimentale acquisito tramite tecnica ottica PIV, contenente dati acquisiti ad una frequenza almeno doppia rispetto alla frequenza tipica del fenomeno studiato, tali dati vengono comunemente definiti *time resolved*. Essi riguardano il fenomeno della convezione di Rayleigh-Benard ed in particolare l'analisi di una cella di Bénard, si noti come non è stata aumentata la frequenza di acquisizione ($\sim$ 15 Hz), bensì risulta molto bassa la frequenza tipica del sistema ($\sim$ 5 Hz).

Per comprendere appieno come alcuni parametri potessero interferire con la buona riuscita della ricostruzione eseguita con approccio ibrido POD-LSTM si è studiato un caso molto semplice quale un'onda sinusoidale numerica. Qui sono stati effettuati diversi studi definendo quindi i limiti entro cui è possibile spingere la rete LSTM, sia in termini di potenzialità del metodo che in termini di potenza computazionale disponibile.

4.2 Algoritmo POD in MATLAB

In MATLAB è già implementato il comando per la SVD. In particolare è possibile ottenere le matrici presentate nell'equazione 2.9 utilizzando il comando:

$$[U, \Sigma, V^T] = svd(A) \tag{4.1}$$

dove con "A" si intende la matrice degli snapshot.
Lo stesso non si applica nel caso del metodo presentato nel paragrafo 2.1.3, per il quale è stato quindi scritta una parte di codice seguendo il metodo presentato da Raiola et al. da utilizzare in presenza di grandi quantità di dati. Il metodo si basa sul principio di intercambiabilità tra la matrice dei coefficienti temporali Ψ e la matrice dei coefficienti modali Φ, e risulta essere molto vantaggioso in caso si disponga di un dataset con un numero di snapshot elevati in confronto al numero di grid points.

Infatti, supponendo di disporre di una matrice degli snapshot A con dimensioni n x p ,dove n è il numero degli *snapshot* e p è il numero di *grid points* a disposizione, essa può essere decomposta come:

$$A = \Psi \, \Sigma \, \Phi \tag{4.2}$$

Con Φ e Ψ presentate precedentemente, e Σ matrice diagonale che contiene i *singular values* del campo vettoriale lungo la sua diagonale. Le basi ottimali e ortonormali cercate dalla POD in modo da formare le matrici appena citate sono ottenibili dagli automodi normalizzati della matrice di covarianza spaziale:

$$A^T A = \Phi^T \Lambda \Phi \tag{4.3}$$

Tuttavia viene riportato che i modi POD possono essere anche calcolati dagli automodi normalizzati della matrice di covarianza temporale:

$$A A^T = \Psi \Lambda \Psi^T \tag{4.4}$$

E poiché $A A^T$ è una matrice Hermitiana definita positiva allora i suoi autovalori sono sicuramente maggiori di zero, rappresentando quindi in modulo il rispettivo contributo energetico di ogni modo; essi sono presenti in numero pari al numero di snapshot. Per gli scopi di questa tesi è stato quindi implementato l'algoritmo riportato in eq. 4.4, previa riorganizzazione della matrice A, in modo tale da far corrispondere le sue dimensioni, e sottrazione del valor medio dalla stessa, in modo da ottenere solo le fluttuazioni del campo di moto.

4.3 LSTM in MATLAB

Benchè in un primo momento si sia pensato ad implementare la rete LSTM mediante librerie già esistenti su Python, in un secondo momento si è ritenuto più conveniente sfruttare il *Deep Learning Toolbox™* già presente su MATLAB a causa di un'interfaccia più user-friendly e della maggior facilità con cui è possibile costruire reti neurali, grazie ad algoritmi già esistenti. L'architettura della rete LSTM pre-impostata di MATLAB rispetta appieno quanto presentato finora, come illustrato nella figura 4.1. Invece, per quanto riguarda le reti LSTM utilizzate per classificazioni di contenuti multimediali, vengono effettuate delle modifiche in alcuni layer che però non verranno illustrate poiché esulano dallo scopo di questa tesi.

Figure 4.1: Schema semplificato dell'implementazione di una rete LSTM in MATLAB

Affinché tutto funzioni correttamente è necessario che vengano rispettati tutti i punti riportati nel sottostante elenco, rigorosamente nell'ordine riportato:

1. Bisogna creare un vettore contenente tutti i layer precedentemente specificati. Nello pseudocodice sopra riportato, esso è indicato come *layers*.

2. L'***input layer*** creato deve avere la stessa dimensione dei dati forniti nella sequenza di addestramento (nel caso della predizione temporale dei coefficienti a_k quindi la dimensione sarà rappresentata dal numero di snapshot).

3. Il ***layer LSTM*** viene pre-impostato semplicemente fornendo il numero di neuroni desiderato (indicato come *numHiddenUnits*) ed il tipo di operazione da eseguire, che nel caso degli studi effettuati in questa tesi sarà del tipo *sequence-to-sequence* prevedendo quindi l'aggiunta di un'opzione 'OutputMode','sequence'.

4. Il ***Fully Connected Layer*** creato deve avere dimensioni pari al numero di output desiderato, esso inoltre verrà collegato a tutti i neuroni dell'ultimo layer.

5. È necessario chiudere il vettore indicato all'inizio di questa lista con un ***regression layer*** nel caso di predizione temporale.

Di seguito è riportato il codice scritto per pre-impostare la rete neurale, fornito da MATLAB.

```
numFeatures = 12;
numHiddenUnits = 125;
numResponses = 1;
layers = [ ...
sequenceInputLayer(numFeatures)
lstmLayer(numHiddenUnits,'OutputMode','sequence')
fullyConnectedLayer(numResponses)
regressionLayer];

```

È anche possibile prevedere delle reti LSTM più complesse aggiungendo ulteriori LSTM layers avendo però cura di evitare l'insorgere del fenomeno di *overfitting* aggiungendo degli appositi layer chiamati *dropout Layer*, spesso usati per regolarizzare la rete. L'overfitting è un fenomeno tipico delle reti neurali che si verifica durante l'addestramento delle stesse: la rete si adatta troppo ai dati a disposizione a causa dell'alto numero di parametri in relazione ai dati, comportandosi equivalentemente ad uno studente che "impara a memoria" una lezione e non saprà quindi rispondere correttamente a domande al di fuori di quanto studiato; la rete infatti risponderà in maniera errata nel caso di dati diversi da quelli usati durante il training. Essa, invece di imparare a memoria i dati, dovrebbe cercare di carpire

gli andamenti dei dati e riuscire così ad effettuare una sorta di "generalizzazione" del loro comportamento.

In letteratura è possibile trovare vari metodi per evitare il fenomeno dell'overfitting tra cui:

- **Early Stopping** Consiste nel fermare l'addestramento della rete nel momento in cui, venendo testata sui dati di *validazione* [2] , non si osservano miglioramenti in termini di accuratezza di previsione.
- **Aumento dati** È il metodo più ovvio ma non è il più semplice da attuare, infatti potrebbero sorgere problemi legati alla natura dei dati (ad esempio impossibilità ad aumentare la frequenza di acquisizione) oppure problemi computazionali nel processare una quantità di dati eccessiva.
- **Regolarizzazione mediante dropout** Onde evitare il fenomeno dell'overfitting vengono omesse alcune *hidden units* dalla rete già esistente, in modo totalmente casuale e ripetuto per ogni epoca: questo spinge i neuroni rimanenti a modificare i suoi pesi per adattarsi alla nuova configurazione. Alla fine di ogni epoca vengono ripristinati i neuroni "sospesi" e ne vengono omessi degli altri sempre in modo casuale. Esistono anche altre tecniche di regolarizzazione ma per non appesantire la trattazione non verranno descritte.

[2] I dati di validazione vengono estratti dal dataset di allenamento, con lo scopo di verificare l'accuratezza di previsione della rete al termine di ogni epoca onde evitare il fenomeno di overfitting.

Capitolo 5

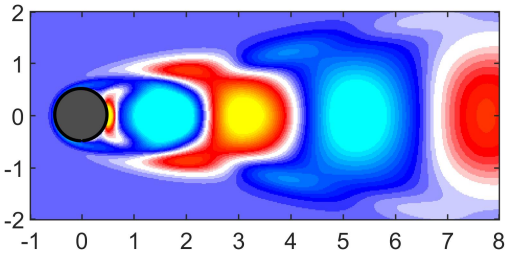
Risultati

In questo capitolo vengono mostrati i risultati ottenuti a seguito dell'implementazione della metodologia POD-LSTM appena presentata. Nella sezione 3.2.1 è stata descritta la rete LSTM in generale, e soprattutto sono stati analizzati i vari tipi di allenamento possibili. I risultati che seguono sono stati ottenuti utilizzando la medesima procedura di addestramento: si effettua una divisione dei dati forniti, in modo da utilizzare una prima parte dei dati per allenare la rete, ed una seconda parte per un successivo test del comportamento della stessa; convenzionalmente è stato fissato questo splitting come 90% - 10%, con la prima parte relativa al train e la seconda relativa al test. Tuttavia, al fine di investigare con maggior dettaglio i limiti ed i comportamenti anomali della rete, queste percentuali sono state modificate in alcuni passaggi, come verrà specificato nelle apposite sezioni. Non sono stati invece mantenuti fissi i parametri della rete LSTM: infatti per migliorare i risultati ottenuti, in alcune trattazioni si è reso necessario aumentare il numero di neuroni (identificati come *numHiddenUnits*), soprattutto in caso di fenomeni complessi (si pensi al getto assial-simmetrico) o nel caso di una matrice con un numero elevato di snapshot (ad esempio la cella di convezione di Rayleigh-Bénard). Secondo quanto presentato in 1 però, all'aumentare del numero di neuroni seguirà anche un aumento della complessità della rete con un aumento della difficoltà di addestramento: per questa ragione sono stati effettuati anche dei vari tentativi al fine di creare una rete che potesse dare il risultato migliore possibile mantenendo il numero di neuroni al minimo necessario. Nelle successive sezioni si evidenzieranno i risultati ottenuti ed i limiti della tecnica.

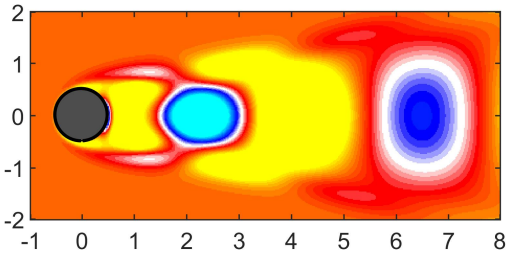
5.1 Il cilindro a $Re = 100$

Come primo ambiente di test cui sottoporre la metodologia è stato scelto un dataset riguardante un fenomeno noto, cioè un cilindro che viene investito lungo il suo

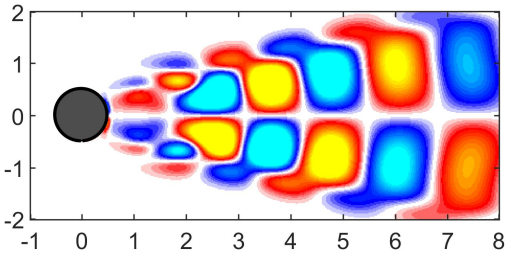
asse principale da una corrente fluida con numero di Reynolds pari a 100. Secondo quanto appreso dalla teoria, per correnti con $Re > 40$ si svilupperà uno sfilamento di vortici, o *vortex shedding* dal retro del corpo. Per numeri di Reynolds compresi tra 40 e 150 lo shedding viene definito *regolare* e lo sfilamento è periodico sia nel tempo che nello spazio, prendendo il nome di *Scia di Von-Karman*. In particolare viene analizzato un campo vettoriale di vorticità numerico, non affetto quindi da errori di misurazione o disturbi di altro genere derivanti da attività sperimentali, che prevede una griglia spaziale di 449x199 punti con 151 snapshot temporali. Dato il basso numero di snapshot e la dimensione accettabile della griglia, è stata implementata la metodologia presentata in 2.1.2 al fine del calcolo dei modi POD, i quali vengono riportati di seguito.



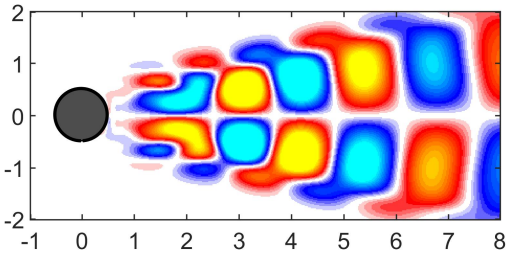
(a) Primo modo POD



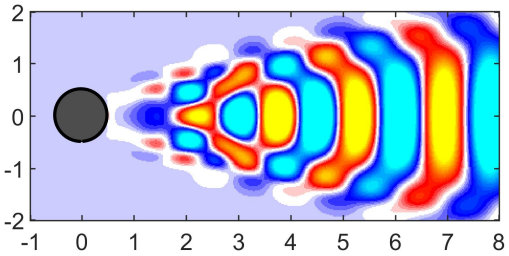
(b) Secondo modo POD



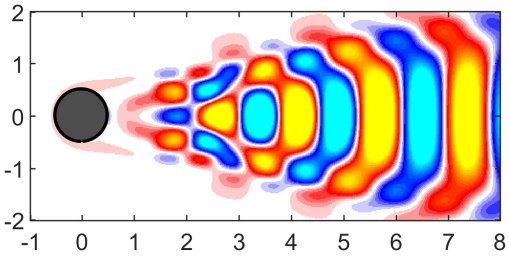
(c) Terzo modo POD



(d) Quarto modo POD

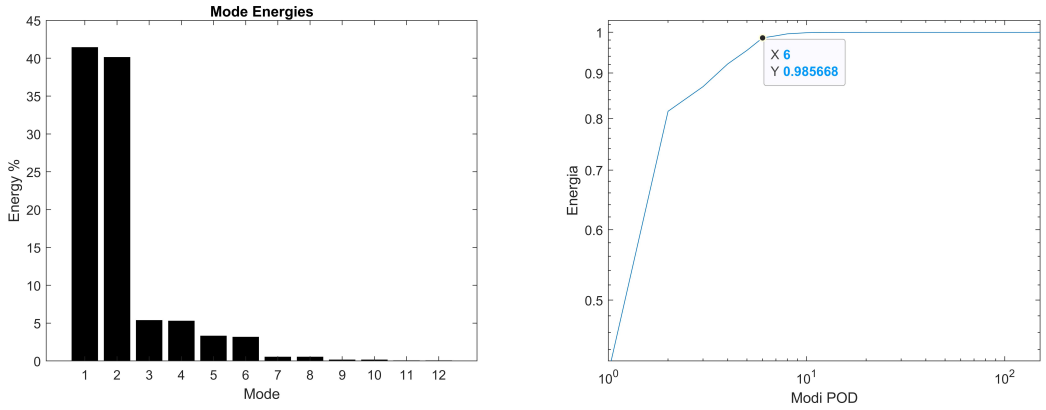


(a) Quinto modo POD



(b) Sesto modo POD

Figure 5.2: I primi sei modi POD



(a) Distribuzione energia associata ai modi. **(b)** Percentuale di energia catturata dai 6 modi pari a circa il 99%.

Figure 5.3: Energia associata ai modi POD.

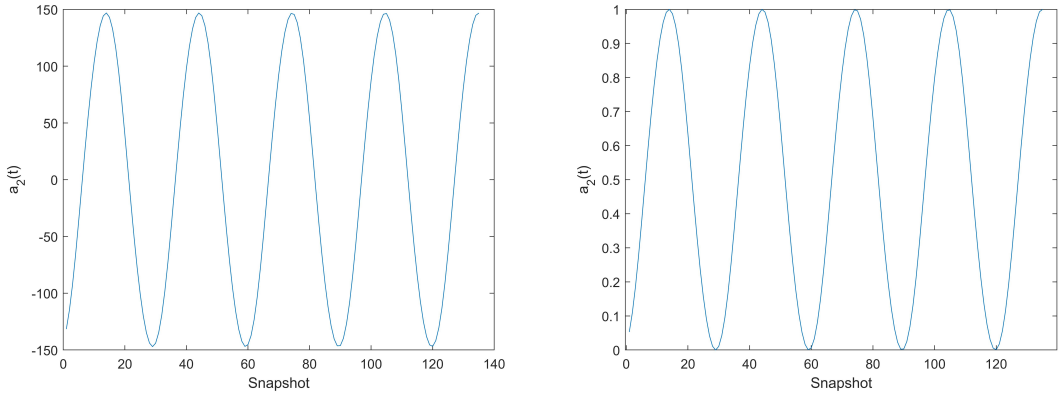
Come anticipato nel capitolo 2 la POD ha subito mostrato le principali strutture del flusso. Infatti già dando uno sguardo ai primi due modi è evidente come il comportamento della vorticità attorno al cilindro sia perfettamente simmetrico, ciò si continua ad evidenziare anche nella scia, con le zone "rosse" che indicano una vorticità positiva e le zone "blu" che indicano una zona con vorticità negativa. Le figure 5.1c e 5.1d sono chiarificatrici: viene mostrata una chiara periodicità spaziale della scia, con le zone rosse e blu che si alternano lungo l'asse orizzontale, indicando proprio una struttura molto affine alla scia di Von Karman. I successivi modi iniziano invece ad essere poco interpretabili fisicamente anche a causa della bassa energia associata ad essi, con la difficoltà a trovarne un riscontro nell'effettivo fenomeno fisico. Difatti in figura 5.3 vengono riportate le energie associate ai primi 12 modi POD in punti percentuali. Risulta palese come circa l'80% sia attribuito ai primi due modi, con i modi successivi al sesto che possono essere trascurati data la scarsa energia ad essi associata (circa 3-4 punti percentuali).

5.1.1 Preprocessing dati

Si è deciso quindi di effettuare un troncamento comprendendo tutti i modi fino al sesto, in modo da catturare circa il 99% del contenuto energetico della vorticità. Non sarebbe stato un problema scartare il quinto e sesto modo, poichè si sarebbe comunque catturata una buona parte di energia, tuttavia data l'elevata potenza computazionale a disposizione si è optato per ottenere dei risultati di maggior qualità, ricordando che i modi complessivi risultano essere pari al numero di

snapshot $n = 151$.

Prima di fornire la sequenza di addestramento alla rete LSTM è necessario effettuare una normalizzazione dei dati, che in questo caso corrispondono ai coefficienti temporali a_k con $k = 0,1,..,r$; r = rango di troncamento. La normalizzazione previene dei picchi che potrebbero disturbare l'addestramento della rete, il quale è già abbastanza delicato in condizioni ottimali; infatti tutti i coefficienti temporali vengono scalati nell'intervallo $0 \div 1$ mantenendo la loro forma d'onda.



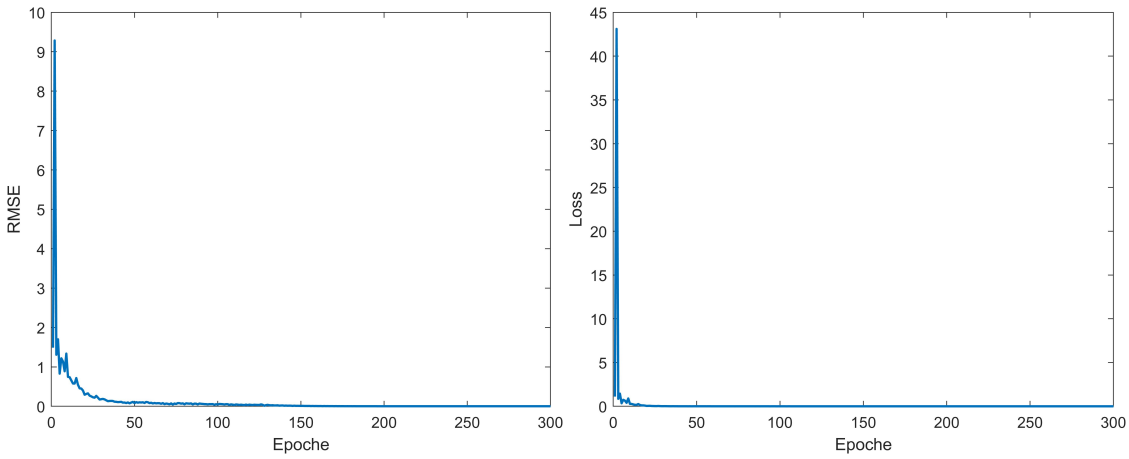
(a) Non normalizzato **(b)** Normalizzato

Figure 5.4: Coefficiente temporale a_2 normalizzato e non normalizzato.

Una volta normalizzati, i coefficienti temporali, vengono divisi in due parti: una destinata all'addestramento ed una dedicata ai test, coerentemente con quanto presentato nell'introduzione del capitolo 5.

5.1.2 Addestramento rete LSTM e ricostruzione

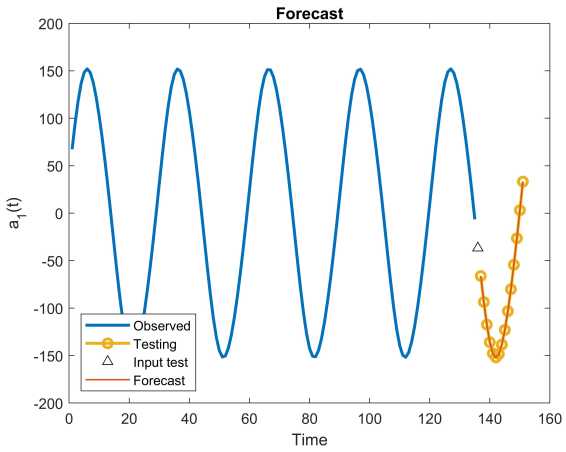
I risultati che seguono sono stati ottenuti con 700 hidden units ed un solo LSTM layer, secondo lo schema riportato in figura 4.1.



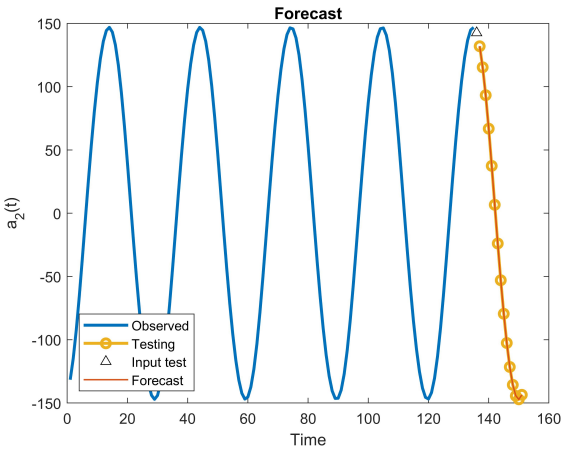
(a) Root Mean Square Error. **(b)** Funzione di perdita.

Figure 5.5: Parametri forniti durante l'addestramento della rete neurale LSTM con il *Deep Learning Toolbox* di MATLAB.

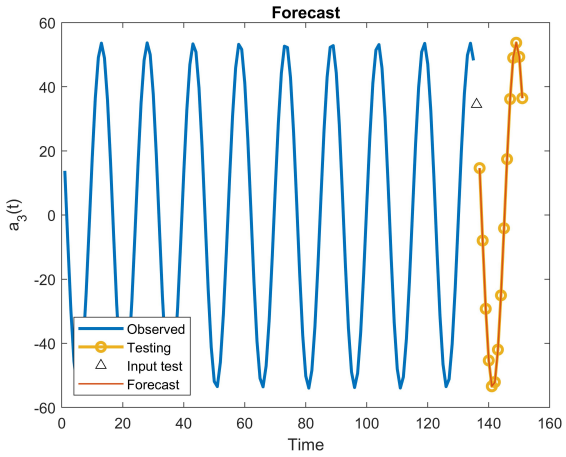
In figura 5.5 vengono riportati: in 5.5a il *root mean square error* ed in 5.5b il *training loss*, due controlli relativi all'addestramento eseguito dalla rete LSTM che vengono ripetuti al termine di ogni epoca. Il RMSE non è altro che la deviazione standard dei residui, i quali a loro volta sono la misura della "distanza" che separa i dati dalla curva di best-fit originata dalla rete neurale, con un valore che si stabilizza intorno a 0.0015. La funzione Loss indica invece quanto i valori predetti si discostano dal valore atteso. Di seguito invece vengono riportati i risultati della ricostruzione ottenuti con la rete appena allenata.



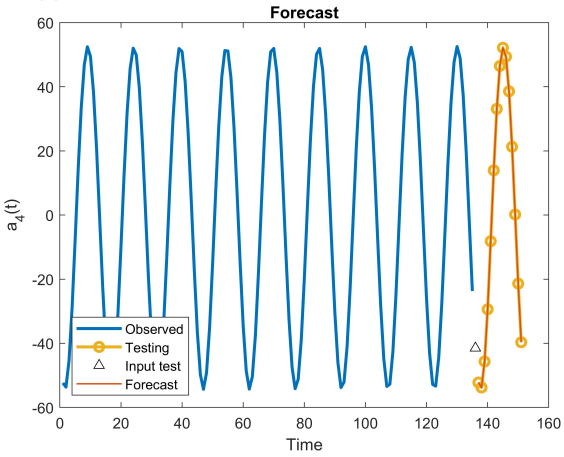
(a) Ricostruzione coefficiente temporale $a_1(t)$



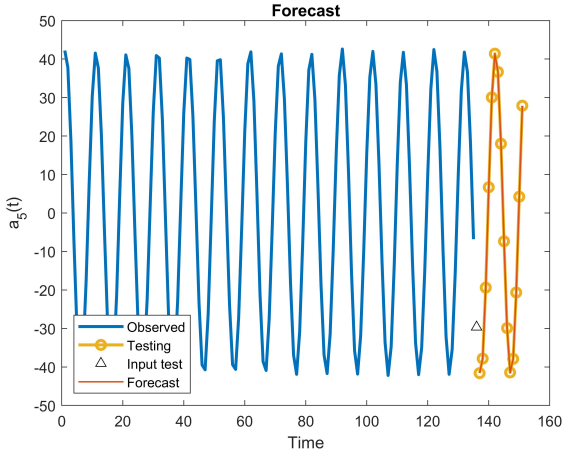
(b) Ricostruzione coefficiente temporale $a_2(t)$



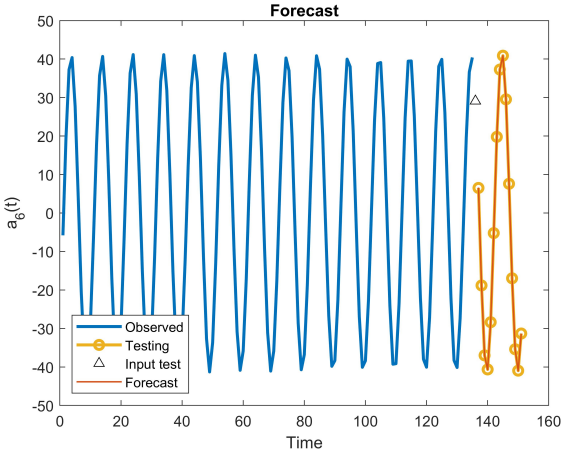
(c) Ricostruzione coefficiente temporale $a_3(t)$



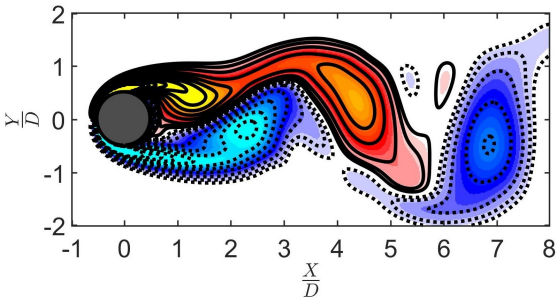
(d) Ricostruzione coefficiente temporale $a_4(t)$



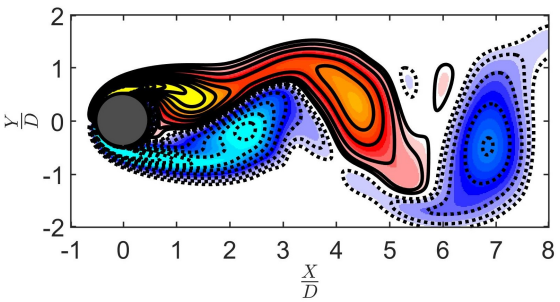
(e) Ricostruzione coefficiente temporale $a_5(t)$



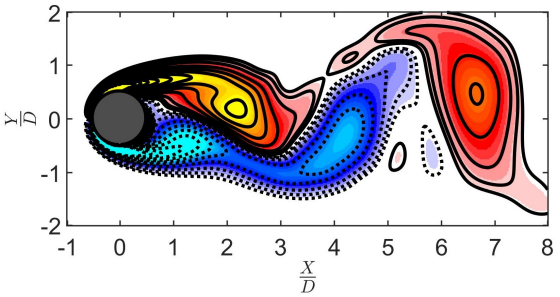
(f) Ricostruzione coefficiente temporale $a_6(t)$



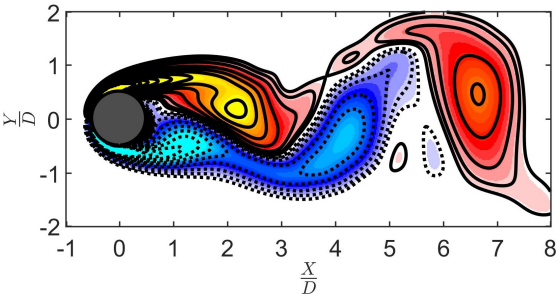
(a) 136° snapshot (primo della serie temporale



(b) Ricostruzione del 136° snapshot



(c) 151° snapshot (ultimo della serie temporale)



(d) Ricostruzione del 151° snapshot

Figure 5.7: Campi di vorticità relativi al cilindro.

I primi sei coefficienti temporali a riportati vengono ricostruiti con un ottimo grado di accuratezza, poiché la linea di previsione o *forecast* segue fedelmente la linea di testing, mostrando quindi un corretto funzionamento della rete LSTM. Questa tesi è avvalorata anche dai campi vettoriali di vorticità ricostruiti, i quali sono mostrati nei primi snapshot successivi all'addestramento della rete dove si mostra una ricostruzione molto fedele all'originale, in cui sono presentidelle impercettibili inaccuratezze che non pregiudicano la bontà della rete, in quanto si ricorda come lo scopo di questo approccio POD-LSTM sia quello di ricostruire le macroscale, alleggerendo significativamente il peso computazionale delle simulazioni con un piccolo prezzo di pagare in termine di perdite di accuratezza. Nell'ultimo snapshot ricostruito la rete continua a prevedere con ottimi risultati il fenomeno del vortex shedding replicando molto bene il comportamento reale, catturando la fenomenologia di macroscala.

5.2 Il getto

In questa sezione verrà illustrata la metodologia utilizzata nell'analizzare dei dati sperimentali a disposizione su di un getto assial-simmetrico, non verranno tuttavia presentati risultati in quanto i campi ottenuti risultano essere errati. Dati gli ottimi risultati ottenuti nella sezione 5.1 si è proceduto ad applicare la metodologia POD-LSTM su dei dati riguardanti un campo più complesso rispetto al vortex-shedding presente a valle di un flusso che investe un cilindro. Difatti nel caso del getto si ha la transizione del flusso da laminare (cuore potenziale) a turbolento con la nascita di strutture turbolente complesse ma coerenti con quanto presentato in sezione 1. Un'altra difficoltà è rappresentata dalla natura sperimentale dei dati acquisiti. Essi infatti a differenza dei dati numerici sono affetti da errori, i quali possono derivare da errori dell'operatore (parallasse, movimentazione manuale etc.) o causati dalla precisione della strumentazione di cui si dispone (Fondo scala trasduttori, sensibilità sonde etc.). Inoltre, rispetto ai dati numerici relativi al cilindro, è stato acquisito un dataset molto ricco, come viene riportato in tabella:

Dimensione griglia	2000 x 1000
Numero Snapshot	6000

Avere a disposizione una tale mole di dati può sicuramente rappresentare un vantaggio in termini di accuratezza nelle simulazioni e completezza nel rappresentare più periodi completi , tuttavia può facilmente trasformarsi in un'arma a doppio taglio in quanto la potenza computazionale necessaria alla gestione di questo dataset supera anche le disponibilità del PC presentato in sezione 1. Il requisito più stringente è la memoria RAM, necessaria per caricare in memoria i dati utili allo sviluppo su MATLAB. A causa di queste limitazioni sono state effettuate delle operazioni preliminari utilizzando una workstation fornita di 64GB di RAM, disponibile presso il laboratorio di aeronautica "Modesto Panetti" del Politecnico di Torino. Si è inoltre resa necessaria l'implementazione del metodo degli snapshot presentato in 2.1.3. In particolare sono stati caricati i campi vettoriali di vorticità i quali sono stati oggetto di analisi per poter effettuare un "taglio" della griglia spaziale. È stata seguita l'idea del ridurre la griglia spaziale eliminando le zone laminari, poichè di scarso interesse al fine delle ricostruzioni, e riducendo le zone turbolente alle sole zone di transizione ed una piccola porzione di scia turbolenta al fine di ridurre i tempi computazionali. Questa scelta si è rivelata vincente poiché la metodologia POD-LSTM non ha restituito i risultati attesi, per cui si sarebbero allungati i tempi per poi ottenere comunque un risultato negativo.

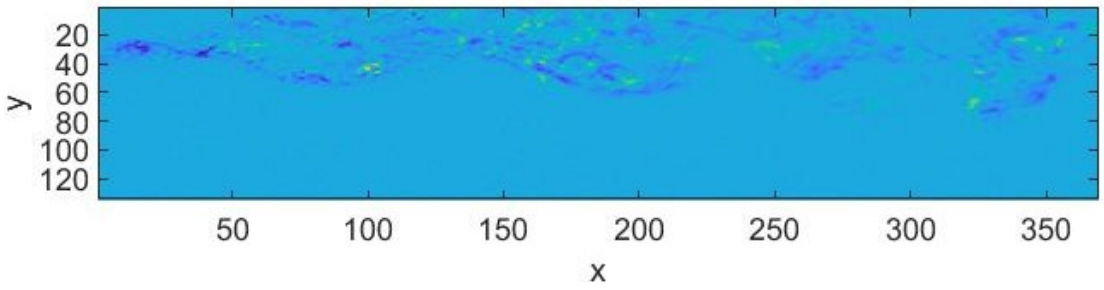


Figure 5.8: Vorticità della porzione di getto turbolento presa in esame

I risultati ottenuti a seguito della ricostruzione sono stati oggetto di profonde analisi per poter comprendere quale potesse essere stato l'errore commesso, con una revisione di tutti i codici utilizzati e diverse prove che consistevano nel variare i parametri fondamentali presentati nei precedenti capitoli; in particolare sono state eseguite prove su:

- **Modifica del rango di troncamento della matrice degli snapshot** Poiché si disponeva di 6000 snapshot, i modi risultavano essere elevati, sebbene con un rango di circa 50, secondo quanto calcolato, si stimava una cattura di circa il 90% della vorticità. Il rango è stato anche impostato a valori molto alti (e ritenuti non necessari ai fini dei calcoli) per scartare l'ipotesi che la non accuratezza potesse essere dovuta all'inclusione di un numero non sufficiente di modi POD. Si è potuto concludere che non fosse questo il problema, poiché sebbene si osservassero dei minimi miglioramenti nei primi snapshot ricostruiti, gli istanti di tempo successivi non mostravano nemmeno l'andamento medio del campo; si tenga inoltre presente che questo comportamento era tenuto dagli snapshot ricostruiti inizialmente, ma tale comportamento non è ritenuto sufficiente al fine di una corretta ricostruzione, dato che l'interesse è incentrato sulle fluttuazioni di velocità.
- **Modifica del numero di snapshot a disposizione** Sebbene in un primo momento si sia optato per una riduzione degli snapshot forniti al calcolatore per un risparmio sia in termini temporali che computazionali, dati i risultati ottenuti, si è optato per fornire il massimo numero di snapshot a disposizione alla rete LSTM. Sono anche stati forniti snapshot per un periodo più lungo ma ridotti in numero per limiti computazionali. Tuttavia entrambe le soluzioni non hanno trovato riscontro nei campi ricostruiti.
- **Modifica della griglia spaziale** Al fine di ottenere una buona ricostruzione si è provato ad analizzare diverse zone del campo di moto, sia spaziando tra la zona completamente turbolenta e la zona di transizione, sia cercando di sfruttare una parziale simmetria lungo l'asse y. È anche stata modificata

la risoluzione della griglia spaziale, sia ampliandola che riducendola (seppur mantenendo al suo interno delle strutture caratteristiche, previa individuazione visiva sui campi originali).

- **Modifica parametri rete LSTM** Tra le quattro modifiche questa è quella che ha apportato più risultati, sebbene tali risultati non siano stati sufficienti a determinare una corretta ricostruzione del campo di moto. Lo scopo principale è stato quello di aumentare il numero di neuroni, al fine di fornire "più memoria" alla rete data la complessità del campo. Sono stati forniti fino a circa 5000 neuroni, limite oltre cui non ci si è potuti spingere per limiti hardware. È stata quindi aumentata la profondità della rete introducendo più LSTM layer, opportunamente intervallati da dropout layer secondo le tecniche presentate in 4.3. Quest'ultima tecnica però, se da un lato aumenta le capacità a disposizione della rete LSTM, dall'altro aumenta sensibilmente anche la difficoltà di addestramento: infatti si sono ottenuti dei risultati parzialmente accettabili (sopratutto considerando i gravi errori iniziali) nei primissimi snapshot seguenti l'addestramento, con una perdita completa dell'informazione negli snapshot seguenti.

Infine a seguito di ricerche bibliografiche si è ipotizzato come le cause potessero originarsi principalmente da due caratteristiche dei dati:

Lunghezza del periodo Ovvero acquisizioni effettuate per istanti di tempo non sufficientemente lunghi alla cattura di un periodo significativo (cioè che possa catturare tutti i comportamenti del fluido). Questa ipotesi è stata ovviamente scartata poiché, data la frequenza propria del getto, si è acquisito sicuramente per tempi sufficientemente lunghi.

Dati non time-resolved cioè dati acquisiti con una frequenza molto più bassa rispetto alle frequenze caratteristiche del fenomeno. Questa ipotesi è stata verificata ed è stata fornita come spiegazione alla fallita ricostruzione.

Effettivamente i dati di cui si disponeva sono stati acquisiti ad una frequenza di 15 Hz, nonostante dalla letteratura le frequenze caratteristiche risultino essere dell'ordine delle migliaia di Hz.

5.3 La convezione di Rayleigh-Bénard

La naturale prosecuzione degli esperimenti è stata la ricerca di un dataset che disponesse di dati acquisiti ad una frequenza tale da poter definire i dati time-resolved. La scelta è ricaduta su dati derivanti da un'acquisizione sperimentale con tempi di macroscala molto lunghi, consentendo quindi rilevamenti accurati

e frequenze di acquisizione sicuramente più alte di quelle proprie del sistema. Il fenomeno studiato è la convezione di Rayleigh-Bénard, fenomeno convettivo naturale nei liquidi immersi in un campo gravitazionale, che dà origine a delle strutture organizzate in celle dette *celle di Bénard*. La nascita di queste celle si deve ad un apporto di energia termica nella parte inferiore del sistema, che genererà un gradiente di temperatura, il quale a sua volta andrà a modificare anche le pressioni e le densità dei vari strati di liquido sovrastanti con il conseguente moto del sistema.

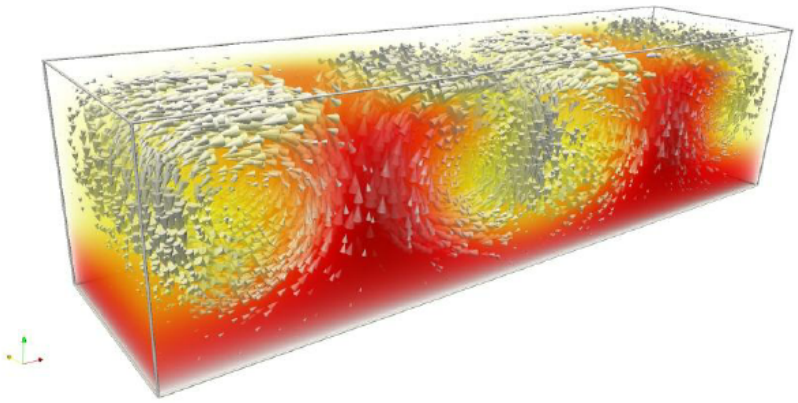


Figure 5.9: Celle di Bènard: in rosso sono indicate le zone più calde, in giallo le zone più fredde.[18]

5.3.1 Importazione dati e test 2D

Il dataset su cui sono stati effettuati i primi test è formato da 600 snapshot con una risoluzione di griglia normalizzata ed espressa in coordinate vettoriali pari a 69 x 36. Si è deciso di iniziare con un campo bidimensionale per procedere eventualmente per gradi, in caso di ottenimento di risultati quantomeno accettabili estraendo il piano corrispondente a Z= 18 dal campo 3D. Data la presenza di gradienti nel campo si è deciso di importare i dati relativi alla vorticità del campo, in modo da poter avere anche un riscontro rispetto a quanto precedentemente svolto sul getto assial-simmetrico affrontato in sezione 5.2. Nonostante la griglia spaziale abbia dimensioni compatibili con il metodo svd presentato in 2.1.2, si è optato per applicare il metodo degli snapshot (si veda la sezione 2.1.3) a causa dell'elevato numero di istantanee temporali di cui si dispone: infatti questa metodologia risulta essere particolarmente conveniente affrontando problemi in cui la dimensione temporale risulta essere maggiore rispetto alla dimensione spaziale. L'analisi delle energie POD riportate in figura 5.10 suggeriscono come sia possibile effettuare un troncamento al quinto modo POD con una cattura di circa il 99% di energia (con un risparmio computazionale non indifferente dato che si parla di 600 snapshot,

corrispondenti a 600 modi).

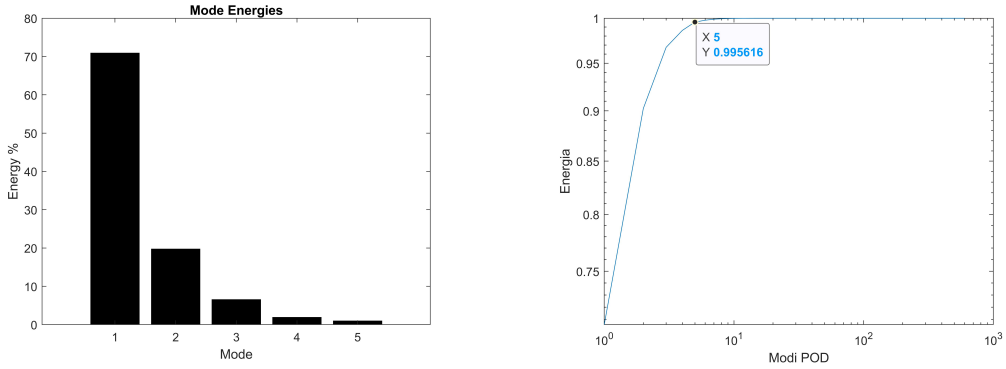
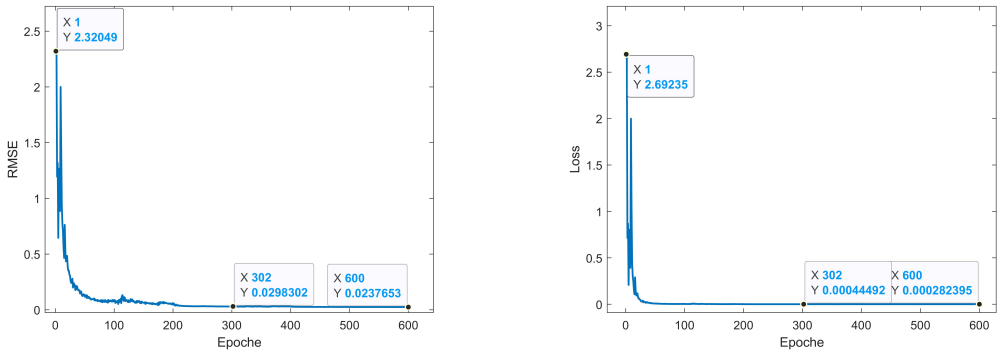


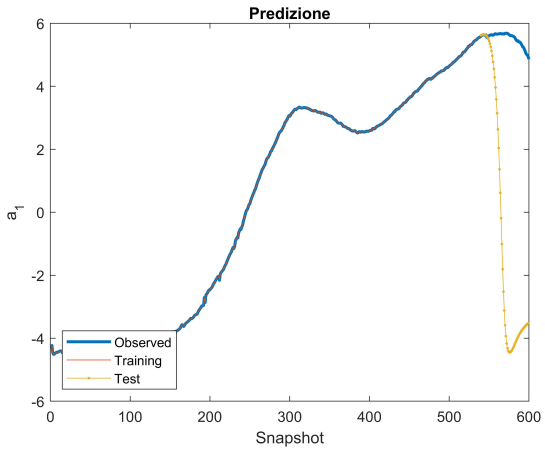
Figure 5.10: Modi energia

I dati a disposizione sono stati quindi processati in modo da ottenere i coefficienti temporali a_k *con* $k = 1, ...,5$ i quali a loro volta sono stati divisi secondo la proporzione $90 \div 10$ in modo da ottenere un dataset di *training* ed uno di *test*. Essi sono stati sottoposti al processo di standardizzazione effettuando un'operazione di sottrazione del valor medio e successivamente divisione rispetto alla deviazione standard, in modo da far ricadere ogni dato nell'intervallo $0 \div 1$. I seguenti risultati sono stati ottenuti con un addestramento effettuato con 500 epoche e 800 hidden units, mentre l'errore tra i campi originali semplificato ed il suo corrispondente ricostruito è stato calcolato e normalizzato come: $\varepsilon = \sqrt{\frac{(w-w_{rec})^2}{w^2}}$ dove con w ci si riferisce al campo di vorticità semplificato con cinque modi POD e con w_{rec} allo stesso campo ma ricostruito dalla rete LSTM.

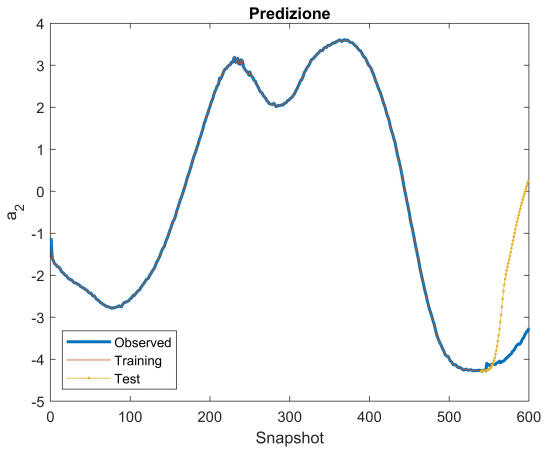


(a) RMSE per 500 epoche. **(b)** Funzione Loss per 500 epoche.

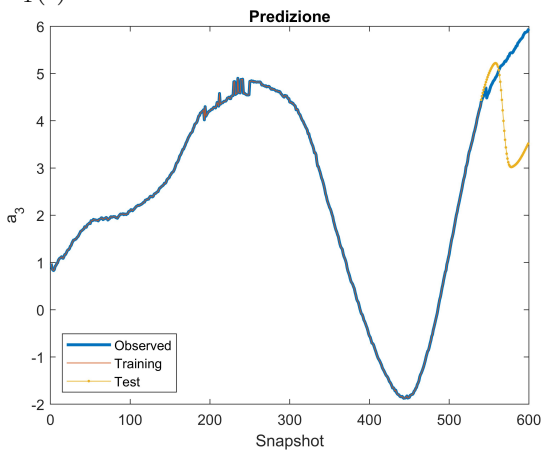
Figure 5.11: Parametri relativi all'allenamento della rete LSTM



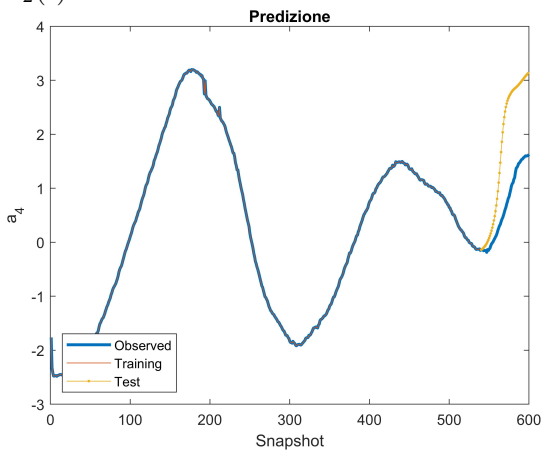
(a) Ricostruzione coefficiente temporale $a_1(t)$



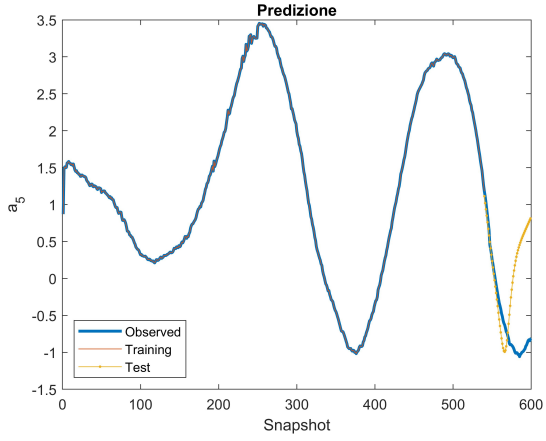
(b) Ricostruzione coefficiente temporale $a_2(t)$



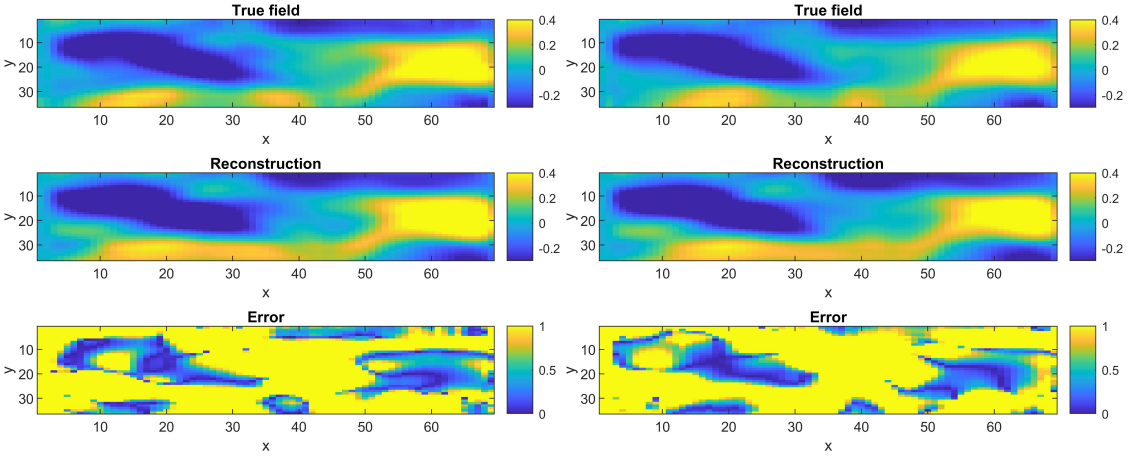
(c) Ricostruzione coefficiente temporale $a_3(t)$



(d) Ricostruzione coefficiente temporale $a_4(t)$



(e) Ricostruzione coefficiente temporale $a_5(t)$



(a) 1°snapshot della serie temporale (corrispondente allo snapshot 541)

(b) 20° snapshot della serie temporale

(c) 40° snapshot della serie temporale

(d) 60°snapshot della serie temporale (corrispondente al 600° ed ultimo snapshot)

Figure 5.13: Campi ricostruiti riportati secondo il seguente schema: in alto il campo vettoriale semplificato (*true field*), al centro il campo ricostruito (*Reconstruction*) ed in basso l'errore commesso (*Error*).

I risultati riportati sembrano restituire dei dati ottimi in quanto in figura 5.11 viene illustrato il buon andamento dell'addestramento della rete LSTM utilizzata. Infatti il Root Mean Square Error e la funzione Loss si attestano su valori molto bassi ed inferiori a 0.1, il che potrebbe portare a credere che la rete funzioni quasi perfettamente. Tuttavia la ricostruzione dei coefficienti temporali mostra come vi siano dei gravi errori soprattutto nei modi più energetici a_1 ed a_2. In particolare tutti i coefficienti temporali ricostruiti mostrano degli errori, con a_5 ed a_4 che nei primi snapshot relativi al dataset di test sembrano avere un andamento che

potrebbe essere ritenuto accettabile (soprattutto a_5), però è altrettanto evidente come con il crescere degli istanti temporali cresca anche l'errore commesso nel prevedere il suo andamento. Inoltre si ricorda come questi due coefficienti siano relativi a dei modi energetici con basso contenuto energetico, per cui gli andamenti più importanti sono relativi ai primi tre modi. I coefficienti temporali ricostruiti a_1 ed a_2 divergono rapidamente a partire dal ventesimo snapshot di test, e questo è visivamente apprezzabile nei campi ricostruiti riportati in figura 5.13. Infatti si nota come l'errore riportato in figura 5.13b si mantenga circa costante, tranne alcune zone circoscritte, rispetto a quanto riportato in figura 5.13a; effettuando un'analisi visiva i campi ricostruiti sembrano essere qualitativamente simili al campo originale semplificato, ma l'errore commesso risulta essere prossimo al 100% in molte porzioni di campo. Una spiegazione plausibile potrebbe essere che trattandosi di valori molto piccoli, una piccola variazione potrebbe risultare in un errore molto alto; questo spiegherebbe anche perché visivamente, su una scala molto ampia, alcune zone sembrano essere uguali. La similitudine precedentemente evidenziata inizia a scomparire in figura 5.13c e la causa può essere ricondotta al divergere dei coefficienti temporali dall'andamento previsto.

Merita invece un approfondimento quanto riportato in figura 5.13d, relativa alla ricostruzione dell'ultimo snapshot di test, poiché seguendo l'andamento indicato finora, tenuto conto del comportamento dei coefficienti temporali i quali divergono dalla curva attesa, ci si aspetterebbe un errore elevato con dei campi completamente irriconoscibili. Infatti il campo ricostruito risulta essere diverso nel complesso dal campo originale semplificato corrispondente. Ciononostante non si ha una perdita completa delle informazioni, infatti alcune zone del campo, seppur con valore diverso, continuano a rappresentare l'andamento qualitativo del campo, il che è parzialmente rappresentato nel rispettivo grafico d'errore.

Questo comportamento anomalo potrebbe essere dovuto alla ridotta dimensione degli spostamenti (sia spaziali che temporali), poichè piccoli spostamenti implicano piccoli errori. Per confermare questa teoria, e verificare l'esistenza di queste discrepanze, si è passati ad analizzare lo stesso dataset però in tre dimensioni.

5.3.2 Test 3D

Il test è stato eseguito sugli stessi dati appena trattati, con l'unica differenza che è stata acquisita anche la terza dimensione, in modo da ottenere una griglia spaziale con dimensioni pari a 69x36x36. Anche in questo caso è stata preferita l'applicazione del metodo degli snapshot trattato in sezione 2.1.3, sia per la struttura 3D dei dati e sia per quanto precedentemente spiegato in sezione 5.3.1. L'analisi dell'energia dei modi POD riportata in figura 5.14 mostra come, con l'aumentare delle dimensioni, si sia andati incontro ad una modifica delle ripartizioni energetiche nei modi.

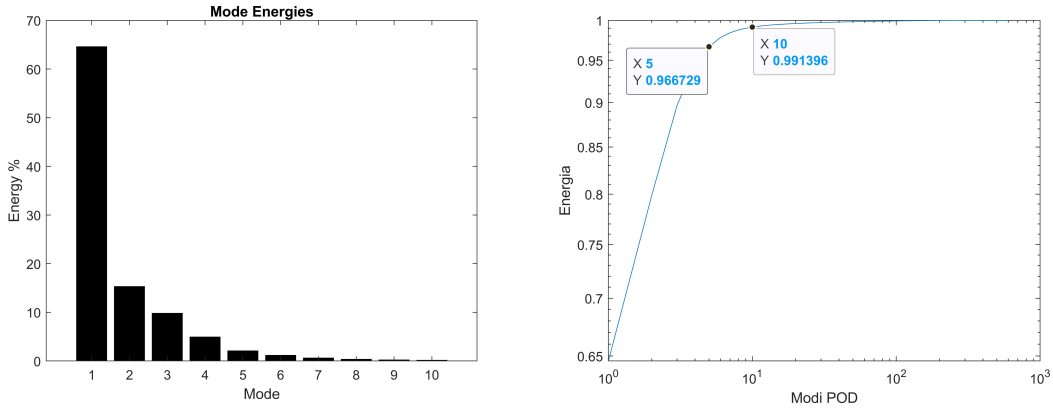
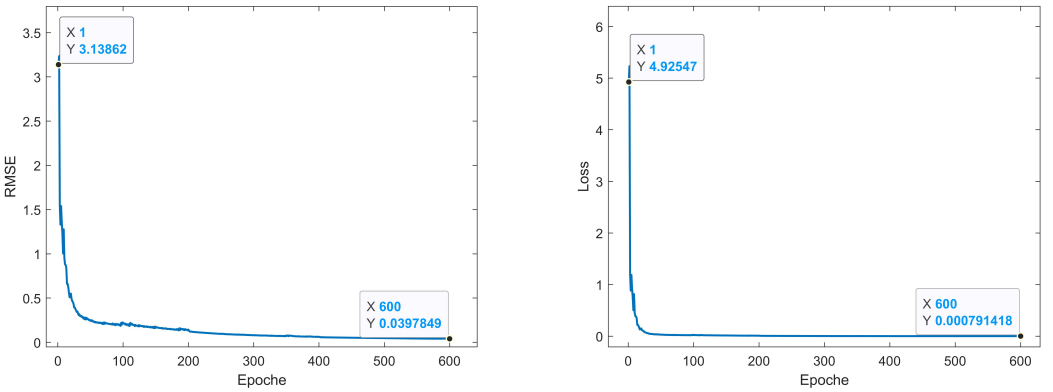


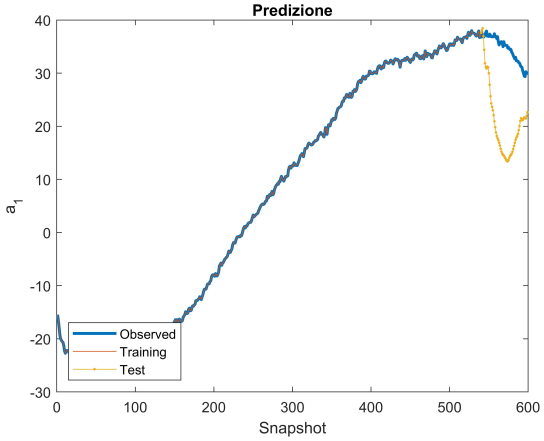
Figure 5.14: Energia associata ai modi POD del campo vorticoso semplificato tridimensionale.

Nel caso 2D, ai primi 5 modi veniva associata un'energia pari a circa il 99% dell'energia, mentre adesso ne contengono circa il 96%. Al fine di ripristinare il contenuto energetico catturato è necessario utilizzare 10 modi POD, i quali infatti sono associati a più del 99% dell'energia. È stato quindi effettuato il calcolo dei coefficienti temporali a_k *con* $k = 1, ...,10$ ciascuno dei quali è stato diviso in due parti, ottenendo un dataset di *training* (contenente 540 punti) ed uno di *test* (contenente 60 punti). Entrambi i dataset sono stati sottoposti al processo di standardizzazione sottraendo loro la media e dividendo per la deviazione standard. I seguenti risultati sono stati ottenuti con un addestramento effettuato con 1000 hidden units e 600 epoche.

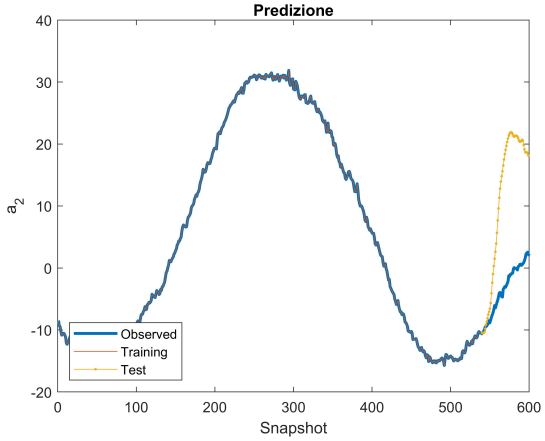


(a) RMSE per 600 epoche. **(b)** Funzione Loss per 600 epoche.

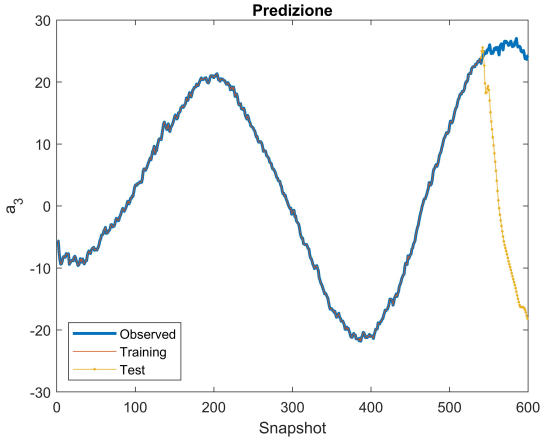
Figure 5.15: Parametri relativi all'addestramento della rete LSTM



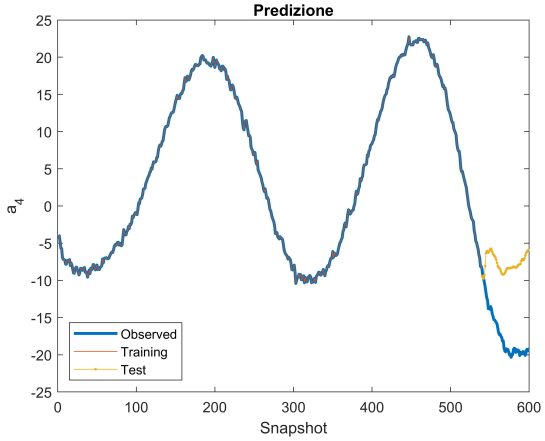
(a) Ricostruzione coefficiente temporale $a_1(t)$



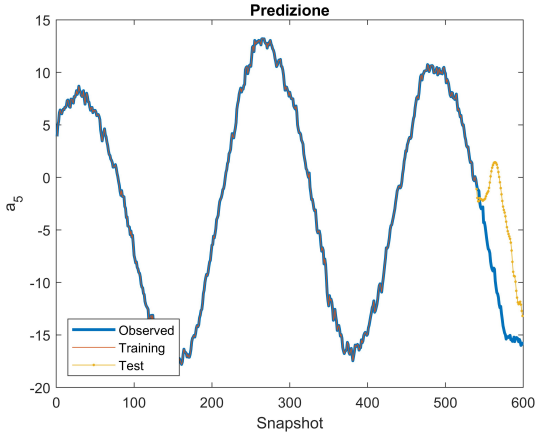
(b) Ricostruzione coefficiente temporale $a_2(t)$



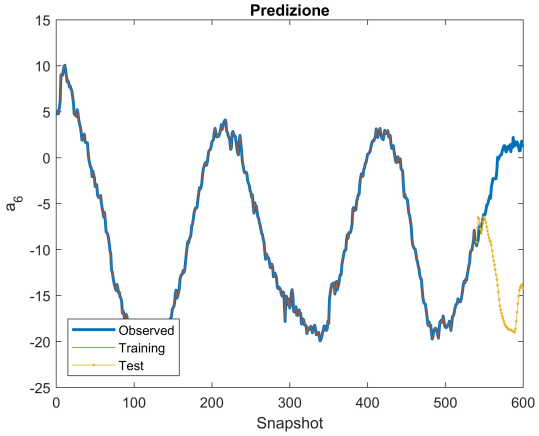
(c) Ricostruzione coefficiente temporale $a_3(t)$



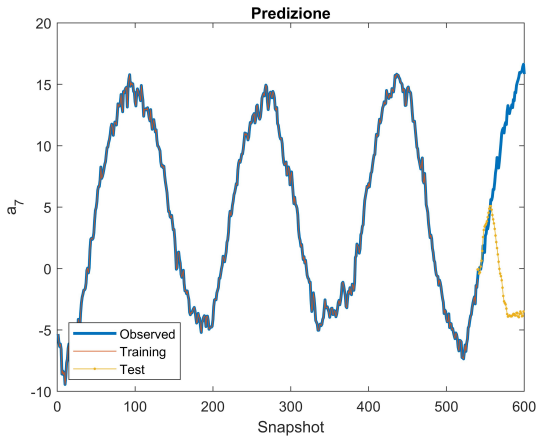
(d) Ricostruzione coefficiente temporale $a_4(t)$



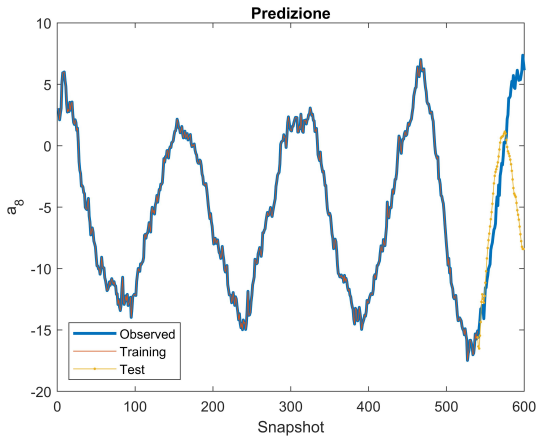
(e) Ricostruzione coefficiente temporale $a_5(t)$



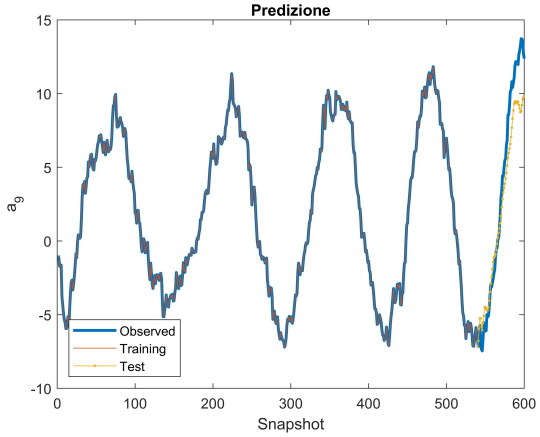
(f) Ricostruzione coefficiente temporale $a_6(t)$



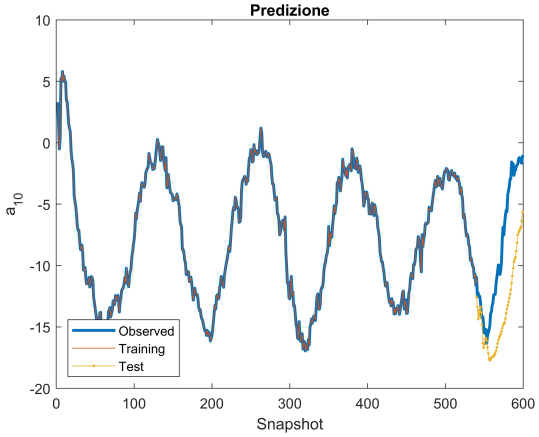
(a) Ricostruzione coefficiente temporale $a_7(t)$



(b) Ricostruzione coefficiente temporale $a_8(t)$



(c) Ricostruzione coefficiente temporale $a_9(t)$

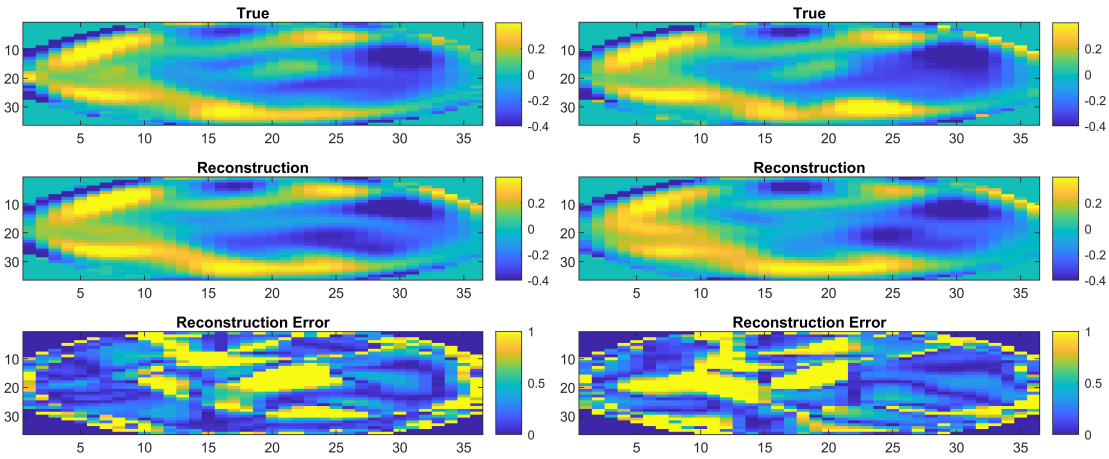


(d) Ricostruzione coefficiente temporale $a_{10}(t)$

Figure 5.17: Ricostruzione coefficienti temporali per campo 3D.

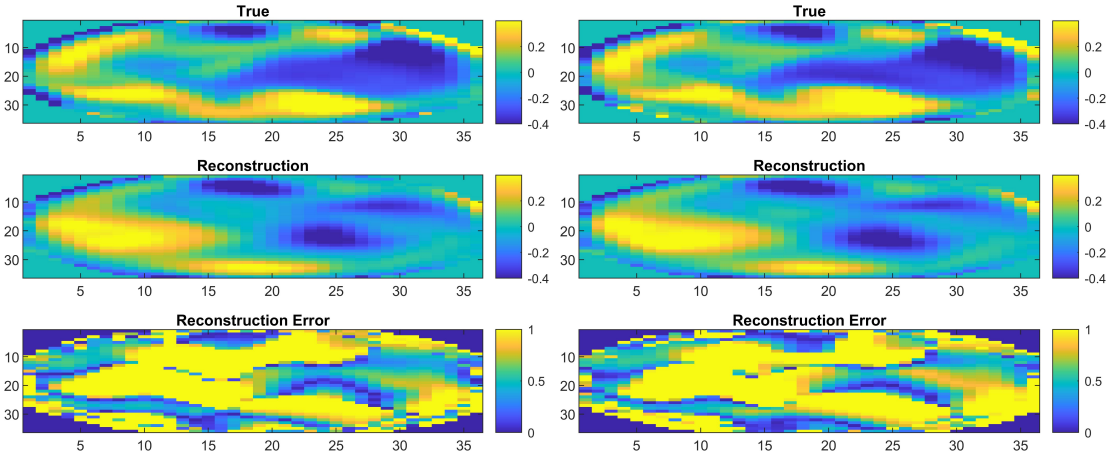
Una volta ottenuti i coefficienti temporali necessari, è stato possibile ricostruire i campi di moto utilizzando la procedura descritta in sezione 2.3. Trattandosi di un campo di vorticità tridimensionale esso è stato rappresentato fissando gli assi X,Y,Z e proiettando quindi rispettivamente sui tre piani Y-Z,X-Z,X-Y. Ogni figura rappresenta il campo originale semplificato (troncato con 10 modi POD), il campo ricostruito e l'errore tra i due campi. L'errore è stato calcolato come:

$$\varepsilon = \sqrt{\frac{(\omega_{real} - \omega_{rec})^2}{\omega_{real}^2}} \tag{5.1}$$



(a) 1° snapshot nel Piano Y-Z (corrispondente allo snapshot 541).

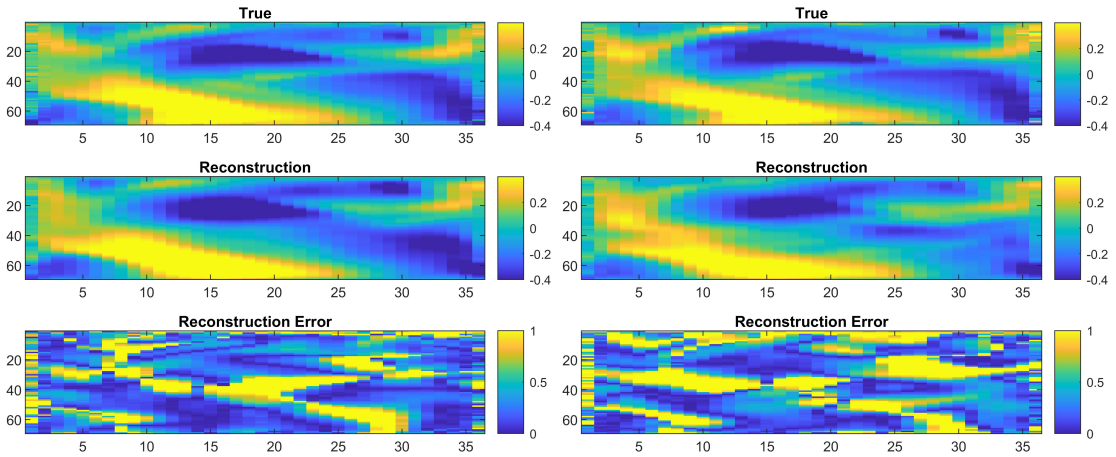
(b) 20° snapshot nel Piano Y-Z.



(c) 40° snapshot nel Piano Y-Z.

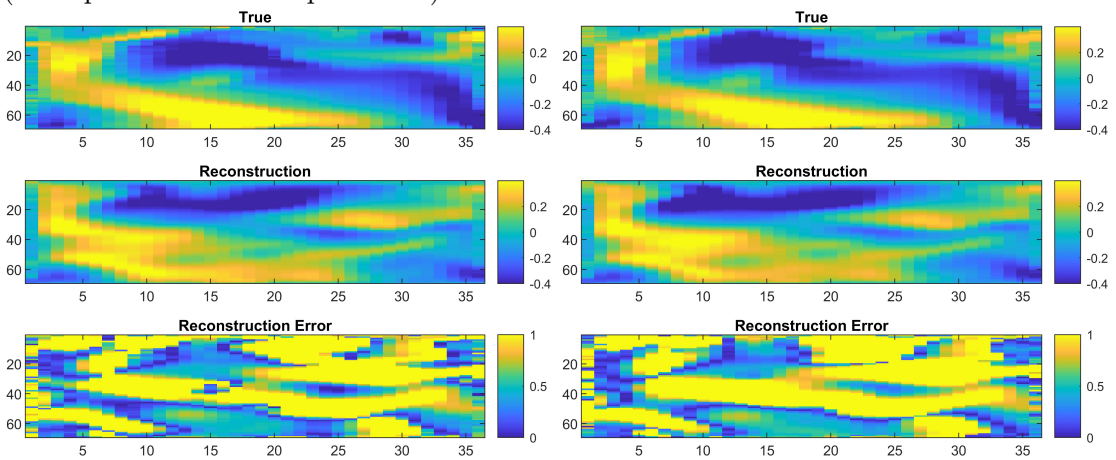
(d) 60° snapshot nel Piano Y-Z.

Figure 5.18: Campi ricostruiti nel piano Y-Z con X=35,riportati secondo il seguente schema: in alto il campo vettoriale semplificato (*true field*), al centro il campo ricostruito (*Reconstruction*) ed in basso l'errore commesso (*Error*).



(a) 1° snapshot nel Piano X-Z (corrispondente allo snapshot 541).

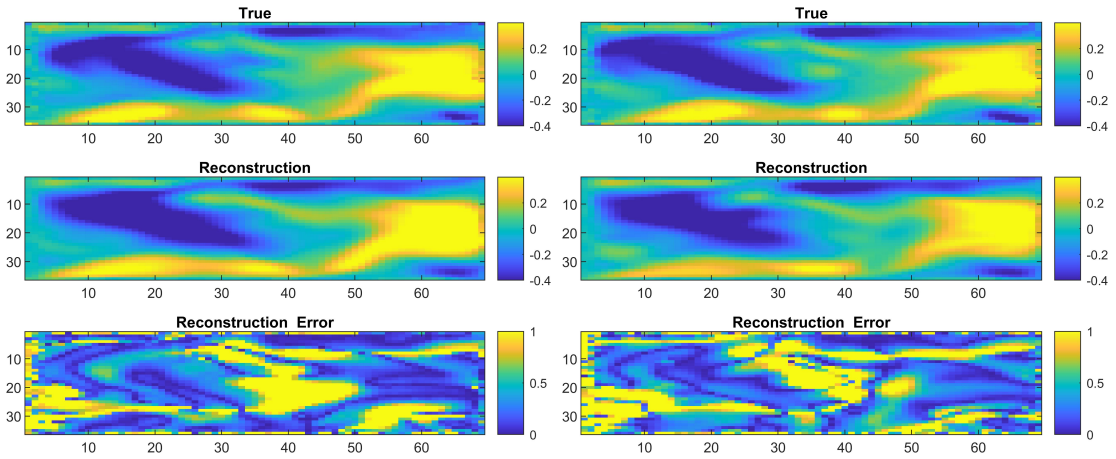
(b) 20° snapshot nel Piano X-Z.



(c) 40° snapshot nel Piano X-Z.

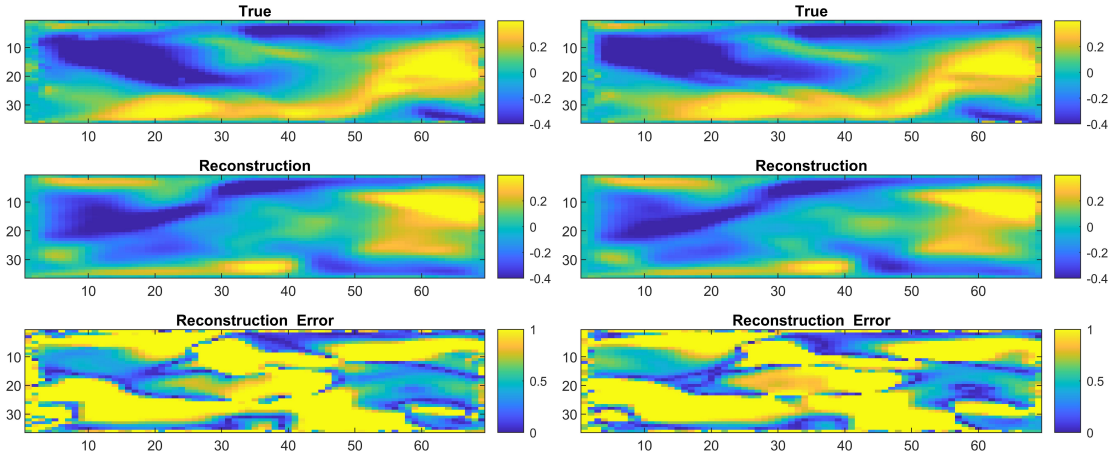
(d) 60° snapshot nel Piano X-Z.

Figure 5.19: Campi ricostruiti nel piano X-Z con Y=18,riportati secondo il seguente schema: in alto il campo vettoriale semplificato (*true field*), al centro il campo ricostruito (*Reconstruction*) ed in basso l'errore commesso (*Error*).



(a) 1° snapshot nel Piano X-Y (corrispondente allo snapshot 541).

(b) 20° snapshot nel Piano X-Y.



(c) 40° snapshot nel Piano X-Y.

(d) 60° snapshot nel Piano X-Y.

Figure 5.20: Campi ricostruiti nel piano X-Y con Z=18,riportati secondo il seguente schema: in alto il campo vettoriale semplificato (*true field*), al centro il campo ricostruito (*Reconstruction*) ed in basso l'errore commesso (*Error*).

Rispetto al caso analizzato in 5.3.1, i risultati riportati sembrano non essere molto differenti. Dai grafici relativi all'addestramento riportati in figura 5.15, sembrerebbe che la rete sia riuscita a modificare i propri parametri in modo da ottenere degli output soddisfacenti: il Root Mean Square Error si attesta su valori inferiori a 0.1 insieme con la funzione Loss. Tuttavia i test effettuati sui coefficienti temporali a restituiscono un altro quadro della situazione. A differenza del precedente caso, dove alcuni coefficienti temporali, seppur legati a bassi contenuti energetci, venivano ricostruiti sufficientemente bene, in questo caso le curve relative ai coefficienti temporali divergono abbastanza rapidamente, con alcuni andamenti apparentemente

inspiegabili. In particolare i coefficienti a_1 , a_2 , a_3 , i quali rappresentano il 90% circa dell'energia totale del campo, mostrano degli andamenti senza un senso logico, il che induce a pensare che la rete non sia riuscita effettivamente a carpirne l'effettivo andamento. Seppur gli andamenti dei restanti coefficienti possa essere ritenuto sufficientemente corretto nei primi snapshot ricostruiti (rispetto ai primi tre), la ricostruzione del campo di vorticità sarà senza dubbio compromessa a causa degli errori in a_1 , a_2 *e* a_3 . Ancora una volta invece, sorprendentemente, non si ha una completa perdita dell'informazione e, seppur i campi ricostruiti non possano essere considerati corretti, si ritiene opportuno effettuare delle considerazioni.

Nelle ricostruzioni relative al piano Y-Z si ha una vista "frontale" della cella di Bénard, infatti è apprezzabile anche a prima vista la forma tipica della cella convettiva. Il primo snapshot ricostruito inizia a mostrare già alcune zone con errori prossimi al 100% , però effettuando un'analisi visiva quantitativa è apprezzabile come la maggior parte del campo rispetti le macroscale, con i pattern principali presenti sia nel campo originale semplificato e sia in quello ricostruito ed un errore che, seppur possa sembrare alto, è localizzato e non completamente diffuso. Nel 20° snapshot ricostruito (corrispondente allo snapshot 560) l'errore appare aumentare di poco ma principalmente spostarsi seguendo l'evoluzione del campo, il che è un comportamento del tutto atteso dato il divergere precedentemente analizzato dei primi tre coefficienti temporali. Ancora una volta però il campo ricostruito continua a mostrare le macroscale relative al campo "vero", seppur iniziando a perdere delle informazioni relative al modulo della vorticità. Il 40° ed il 60° snapshot perdono quasi completamente le informazioni relative sia ai pattern delle macroscale e sia al modulo della vorticità, coerentemente con i coefficienti temporali che divergono molto in questi ultimi istanti di ricostruzione; coerentemente con l'andamento degli errori, i quali risultano essere in prossimità del 100% in quasi tutte le zone, ad eccezione di porzioni ristrette del campo.

Le ricostruzioni relative al piano X-Z sono state effettuate coerentemente con quanto appena esposto per il piano Y-Z. Il primo snapshot ricostruito rappresenta molto bene le macroscale con i pattern ben evidenziati. Sono presenti degli errori relativi al modulo della vorticità con un errore massimo che si attesta solo in alcune zone circoscritte. Il 20° snapshot ricostruito presenta un comportamento simile al suo omologo del piano Y-Z: infatti se nel precedente campo iniziavano ad emergere i primi errori ma erano ancora apprezzabili i pattern del campo originale, adesso il campo di vorticità perde molto meno rapidamente le informazioni relative al campo originale con una buona approssimazione delle macroscale. Qui il modulo mantiene sempre le proporzioni tra le varie zone (ad esempio confrontando due zone non si ha un'inversione di massimi e minimi) però l'errore inizia ad accrescersi, come dimostra l'ampliarsi delle zone ad errore massimo. Gli snapshot riportati nelle figure 5.19c e 5.19d, rispettivamente 40° e 60°, mostrano una perdita molto elevata delle informazioni, coerentemente con quanto visto per il precedente piano, con i

pattern quasi irriconoscibili se non per alcune zone ed i moduli non corrispondenti. Ancora una volta l'errore massimo si attesta su valori del 100% in quasi tutta la porzione di campo preso in esame, seppur con una distribuzione inferiore all'errore calcolato nel piano Y-Z. Lo stesso comportamento è tenuto anche nel piano X-Y. Risulta quindi palese come la rete LSTM abbia incontrato delle difficoltà nel ricostruire fedelmente il campo di vorticità sia in due dimensioni che in tre dimensioni. Gli andamenti sono sempre simili tra loro: l'informazione iniziale, la quale viene ricostruita in maniera discretamente corretta, lentamente si "degrada" nel tempo; gli ultimi snapshot relativi alle ricostruzioni infatti risultano essere abbastanza diversi rispetto alle relative controparti originali. Tuttavia si ricorda come la ricostruzione sia migliorata rispetto a quanto analizzato nel capitolo 5.2, grazie a dei dati time-resolved. Di seguito verranno descritti alcuni dei tentativi effettuati nel cercare di migliorare le capacità predittive della rete LSTM adoperata.

5.3.3 Smoothing a

La predizione dei coefficienti temporali, mostrata in figura 5.17 , non ha restituito i risultati attesi. Osservando le curve ricostruite non è semplice trovare una spiegazione agli andamenti mostrati. Una prima ipotesi che si è avanzata è stata che la rete LSTM potesse essere stata tratta in inganno dalla "seghettatura" delle curve: in esse infatti è possibile distinguere un andamento medio, facilmente approssimabile tracciando una semplice linea, a cui si aggiungono delle fluttuazioni più o meno forti. Considerando l'architettura neurale utilizzata si ricorda come il neurone LSTM decida in autonomia cosa memorizzare a seconda dell'addestramento effettuato, inoltre non ha una visione ampia della curva completa bensì vede una serie di punti senza identificare un andamento medio che l'occhio umano invece riesce a creare in automatico. Inoltre dato il grande quantitativo di punti e le grandi distanze che intercorrono tra le dipendenze utili vi è il conseguente pericolo di sviluppare le problematiche descritte in sezione. Per queste motivazioni si è pensato che il malfunzionamento sarebbe potuto essere ricondotto all'errata memorizzazione di pattern non necessari al fine della ricostruzione. È stato quindi effettuato uno smoothing di prova su tutti i coefficienti temporali, al fine di eliminare queste fluttuazioni.

Lo smoothing è stato effettuato utilizzando la funzione *smoothdata* presente nelle librerie di MATLAB ed tra le varie funzioni di smoothing,la scelta è ricaduta sulla funzione *sgolay*, la quale risulta essere la meno "invasiva"[1] ed utilizza un filtro Savitzky-Golay.

[1]Per non invasiva si intende un filtro che possa aumentare la precisione dei dati senza andare ad intaccare l'andamento medio della curva.

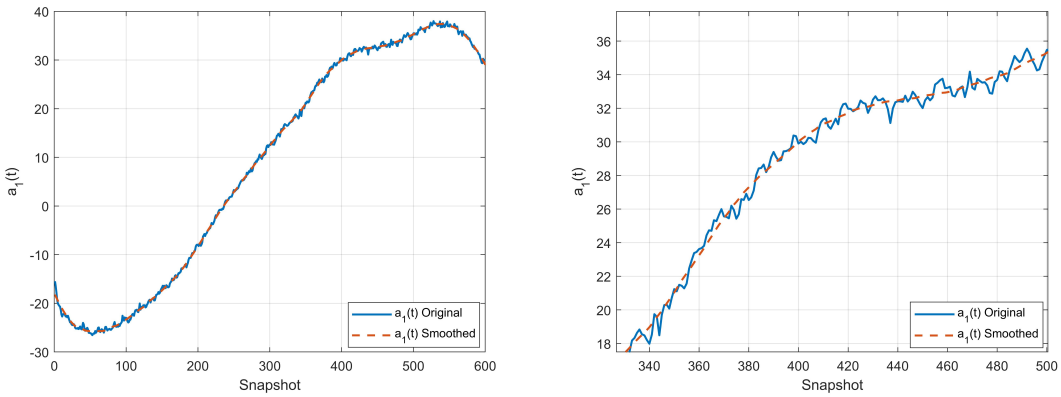
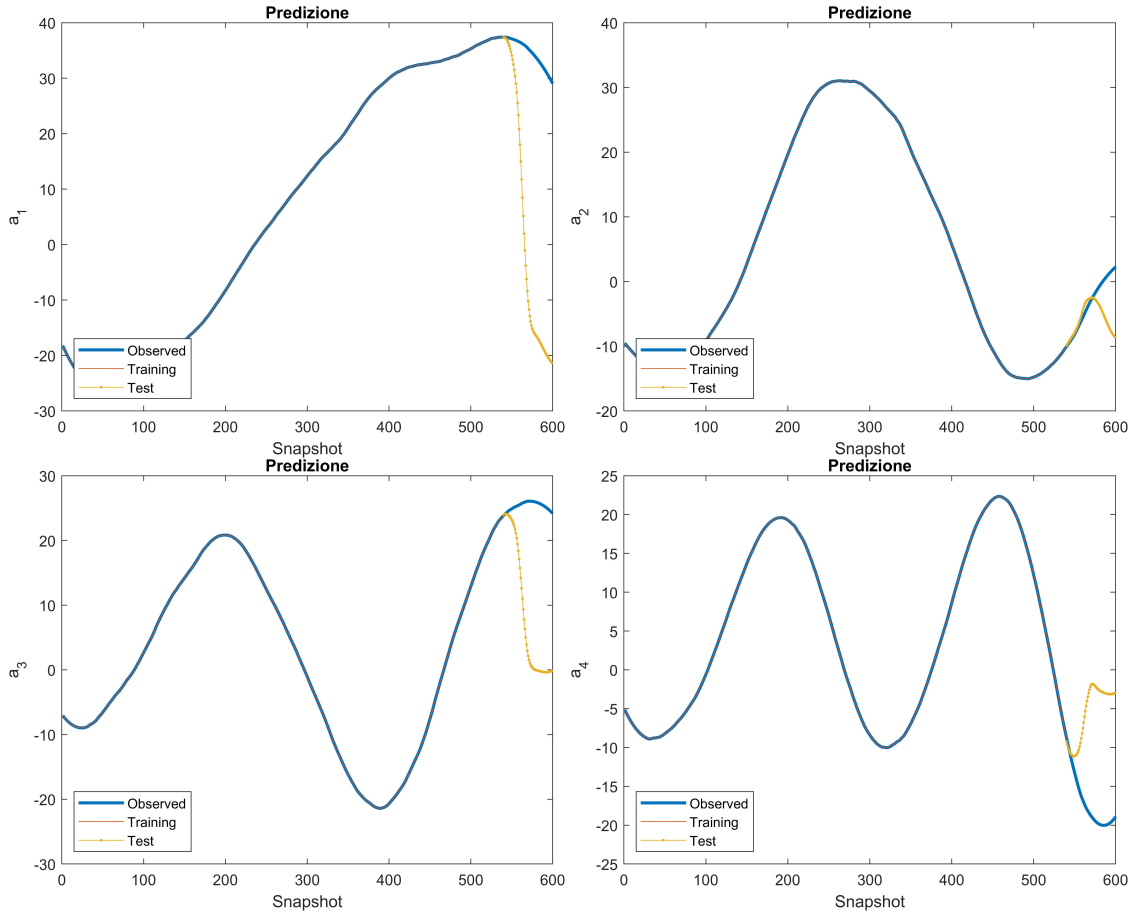


Figure 5.21: Coefficiente temporale a_1 in versione originale ed a seguito dell'applicazione del filtro, rispettivamente in blu e rosso.



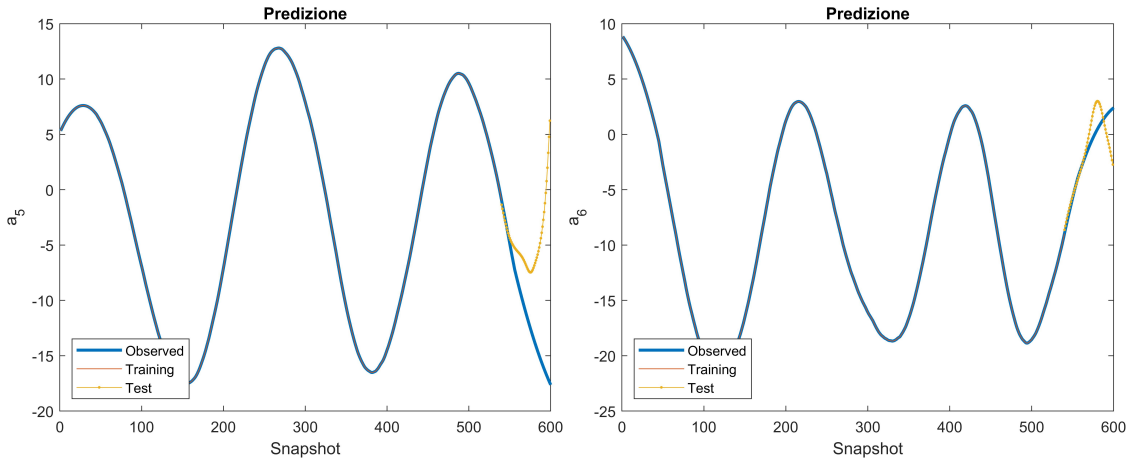
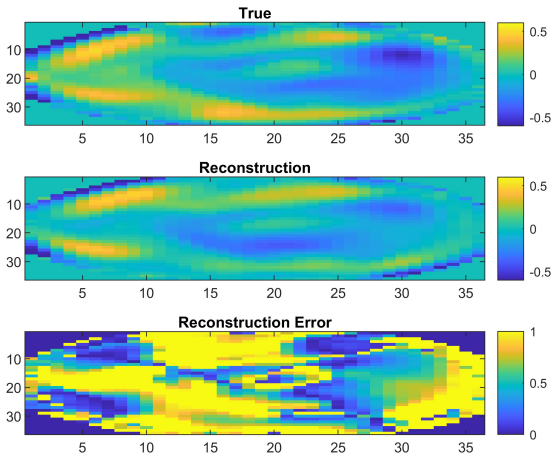


Figure 5.23: Predizione dei primi 6 coefficienti temporali "levigati".

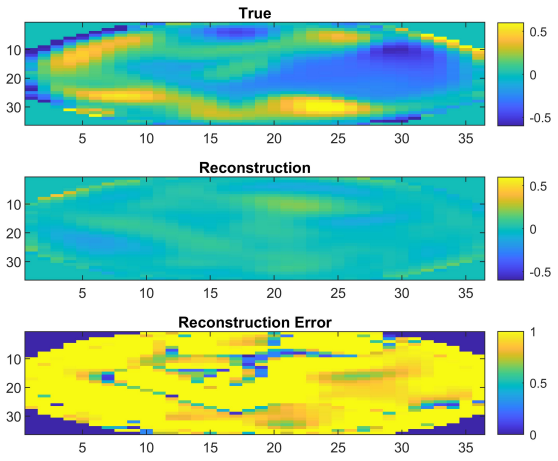
In figura 5.21 si ottengono ottimi risultati a seguito dell'applicazione di questa tecnica, con l'andamento medio della curva che viene preservato. In figura 5.23 vengono riportati invece i risultati ottenuti. La rete LSTM utilizzata possiede le stesse caratteristiche della stessa precedentemente utilizzata per predirre i coefficienti temporali non levigati. Si può notare come questa operazione abbia dato dei risultati contrastanti: infatti confrontando con quanto ottenuto in figura 5.17 è evidente come il coefficiente temporale a_2 abbia ottenuto benefici non indifferenti, con una predizione che risulta essere quasi perfetta fino allo snapshot 575 circa dove poi si ha un'inversione della pendenza della curva con valori che però non si discostano di molto dal risultato atteso. Anche il coefficiente temporale a_6 ha beneficiato di questa operazione seppur sia quasi ininfluente ai fini della ricostruzione. Ciononostante non sono stati riscontrati gli stessi benefici nella predizione degli altri quattro coefficienti temporali [2]: infatti si notano enormi peggioramenti nel coefficiente a_1, mentre si mantengono circa costanti gli errori relativi ai coefficienti a_3, a_4 ed a_5.

Di seguito vengono invece riportati i risultati relativi al campo di moto tridimensionale ottenuto a seguito di smoothing sui coefficienti temporali.

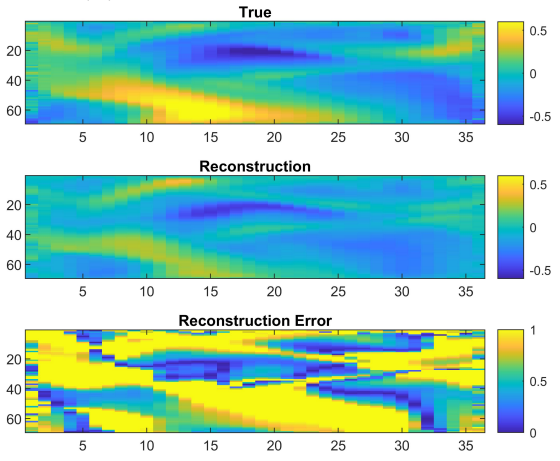
[2]Si ricorda che ci si è focalizzati sui primi 6 modi poiché rappresentano il 97% del contenuto energetico del flusso.



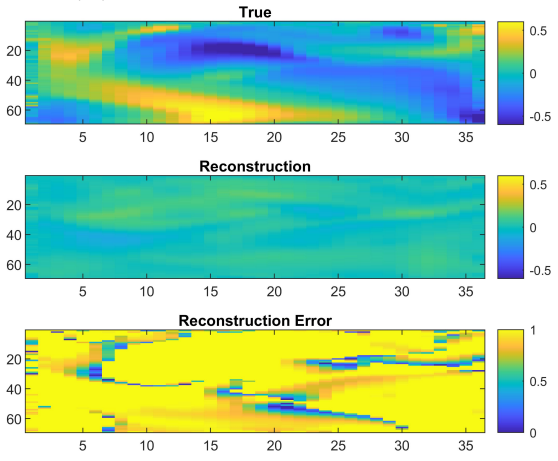
(a) 1° snapshot nel Piano Y-Z.



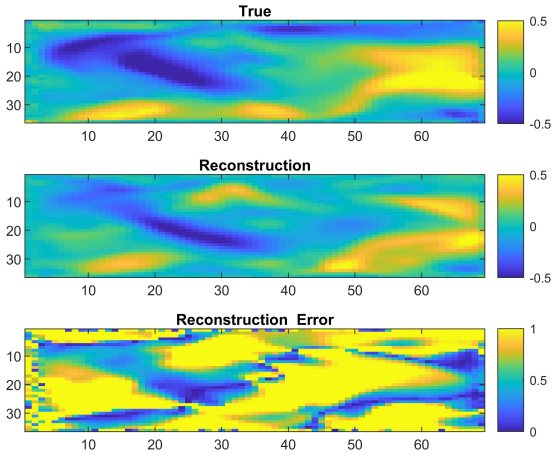
(b) 30° snapshot nel Piano Y-Z.



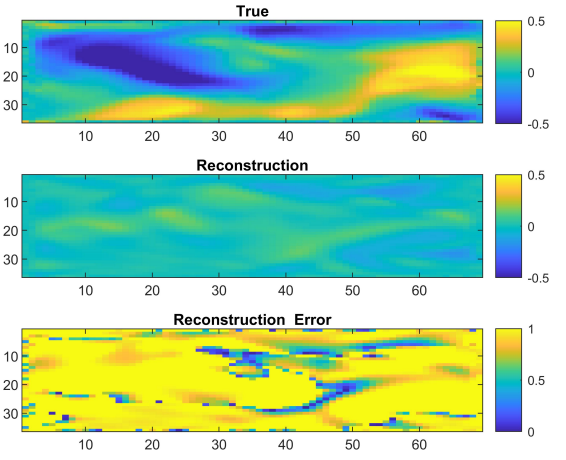
(c) 1° snapshot nel Piano X-Z.



(d) 30° snapshot nel Piano X-Z.



(e) 1° snapshot nel Piano X-Y.



(f) 30° snapshot nel Piano X-Y.

Figure 5.24: Campi ricostruiti con coefficienti temporali levigati.

Come da previsioni i campi non vengono migliorati rispetto alla ricostruzione effettuata senza smoothing. Risulta evidente come già al trentesimo snapshot, cioè a metà del percorso di predizione, gli errori siano distribuiti su quasi tutto il campo con errori prossimi al 100%. Questi errori inoltre risultano già essere presenti a partire dal primo istante di ricostruzione, il che non è un risultato inaspettato dato che al coefficiente temporale a_1 è associata un'energia pari al 65%. Si ipotizza quindi come in quelle fluttuazioni che si sono cercate di appianare, possa essere presente una certa periodicità, sfuggente all'occhio umano, che risulta avere effetti benefici sulla memoria dei neuroni LSTM.

5.3.4 Ripetizione periodo

Dopo aver accantonato l'idea di effettuare uno smoothing sui coefficienti temporali a sono stati effettuati dei tentativi per individuare se il problema potesse risiedere in una mancanza di dati utili a ricostruire un periodo completo. Al fine di simulare quest'effetto sono stati ripetuti parte dei dati iniziali posizionandoli in coda nella matrice degli snapshot.

```
1    k_rep = 0.25;
2    NImg = 1:round((k_rep+1)*nn);
3
4    for i=1:ceil(k_rep)
5        X_3 = cat(4,X_3,X_3);
6    end
7
8    X_3 = X_3(:,:,:,NImg); %X_3 matrice degli Snapshot.
```

Questo codice permette di gestire la quantità di snapshot da ripetere, in modo da ottenere un periodo più o meno lungo. La prima analisi è stata effettuata con una ripetizione del 25% degli snapshot, mentre i parametri della rete e la preparazione dei dati necessari all'addestramento sono rimasti invariati rispetto agli esperimenti precedentemente svolti. Dato l'aumento del 25% nel numero di snapshot (da 600 a 750), e le percentuali fissate 90÷10 relative al data training e data test, la rete è stata testata su 75 snapshot.

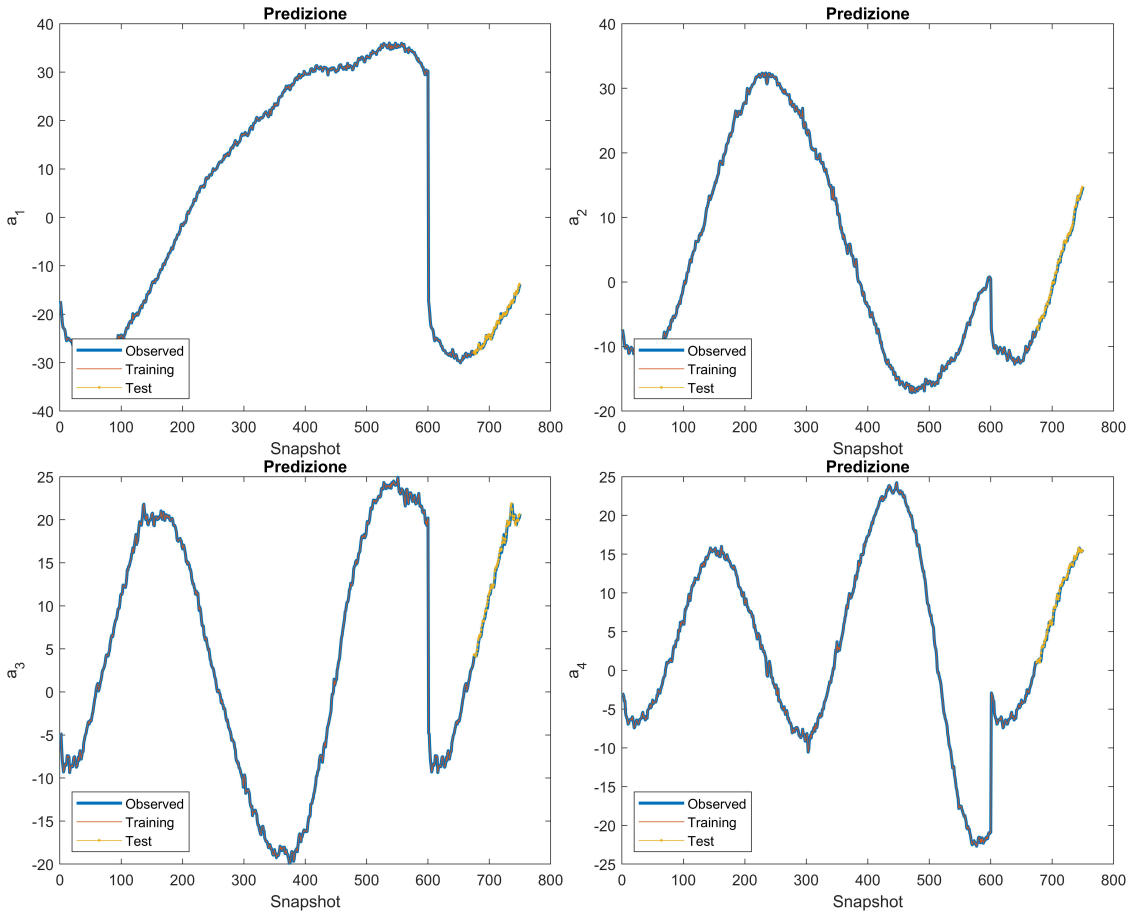
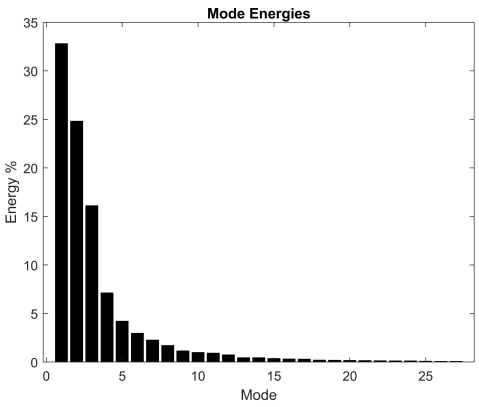


Figure 5.25: Predizione dei primi 4 coefficienti temporali a seguito di ripetizione del 25% dei dati.

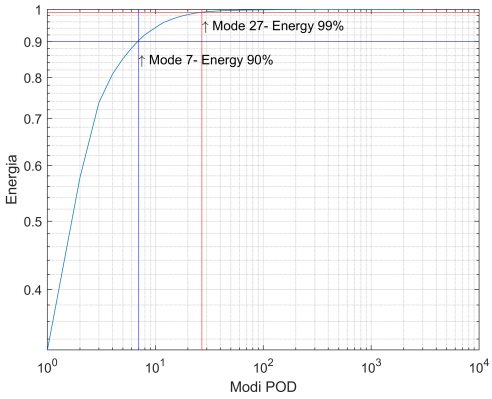
I risultati ottenuti ed evidenziati in figura 5.25, mostrano chiaramente come la rete riesca a ricostruire alla perfezione i coefficienti temporali. Tuttavia si ricorda come queste parti che vengono ricostruite siano già state incontrate dalla rete durante l'addestramento, poiché non corrispondono ad effettivi dati acquisiti ma semplicemente agli stessi dati ripetuti nel tempo, per cui si potrebbe essere indotti a pensare che tale precisione sia frutto di overfitting.

5.3.5 Incremento numero di snapshot

Al fine di confermare quanto appena ottenuto sono stati importati ulteriori 2000 snapshot acquisiti sperimentalmente come i primi 600 tramite tecnica PIV. Grazie ad essi è stato possibile aumentare il tempo totale di acquisizione ed è stata quindi creata una nuova matrice contenente 2600 snapshot, il che implica la necessità di effettuare nuove analisi sulla distribuzione energetica dei modi (adesso pari a 2600).



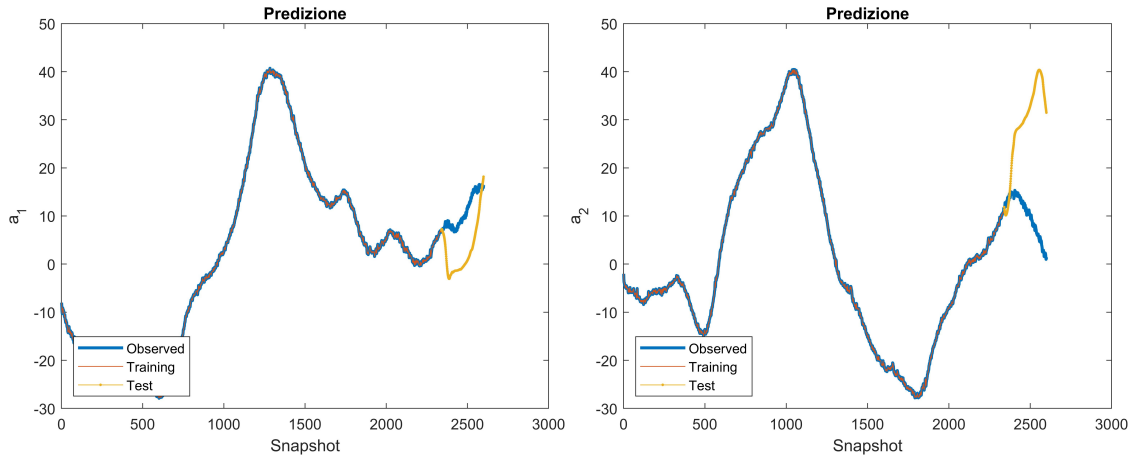
(a) Distribuzione energia tra i modi POD.



(b) Modi necessari per raggiungere il 90-99% dell'energia.

Figure 5.26: Grafici relativi alla distribuzione energetica a seguito dell'aumento numero di snapshot.

La figura 5.26 mostra come l'energia distribuita tra i modi POD sia sensibilmente variata, infatti per rappresentare il 90% dell'energia cinetica turbolenta sono necessari 7 modi POD, mentre per il 99% si ha una crescita esponenziale in quanto sono necessari 27 modi POD; il che resta comunque un enorme risparmio poiché vengono estratti 27 modi invece di 2600. Data l'enorme mole di dati da processare, con conseguente carico di informazioni da memorizzare a cura dei neuroni LSTM, si è deciso di incrementare le capacità della rete neurale fornendola di 3000 hidden units. Gli snapshot dedicati al training ed al test sono stati divisi come di consueto, ottenendo 260 istantanee su cui eseguire il test della rete. Di seguito verranno illustrate le ricostruzioni relative ai primi 6 modi POD, nonché i più utili al fine di una accurata ricostruzione (energia associata pari a $\sim 88\%$).



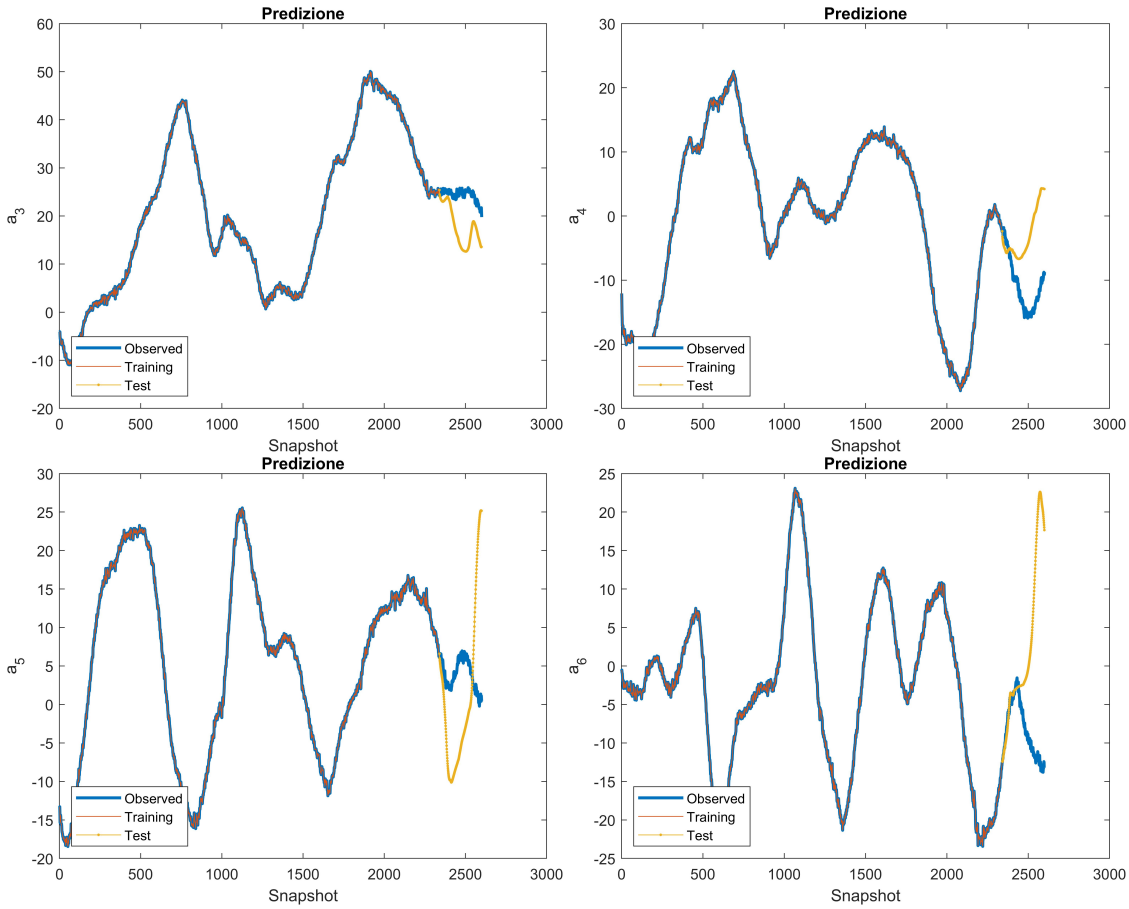


Figure 5.28: Modi POD ricostruti su 2600 Snapshot.

I risultati riportati in figura 5.28 evidenziano problemi nella fase di ricostruzione. In particolare nonostante il periodo sia stato aumentato del 400% l'andamento dei coefficienti temporali a non sembra avere un andamento periodico, il che può avere due spiegazioni:

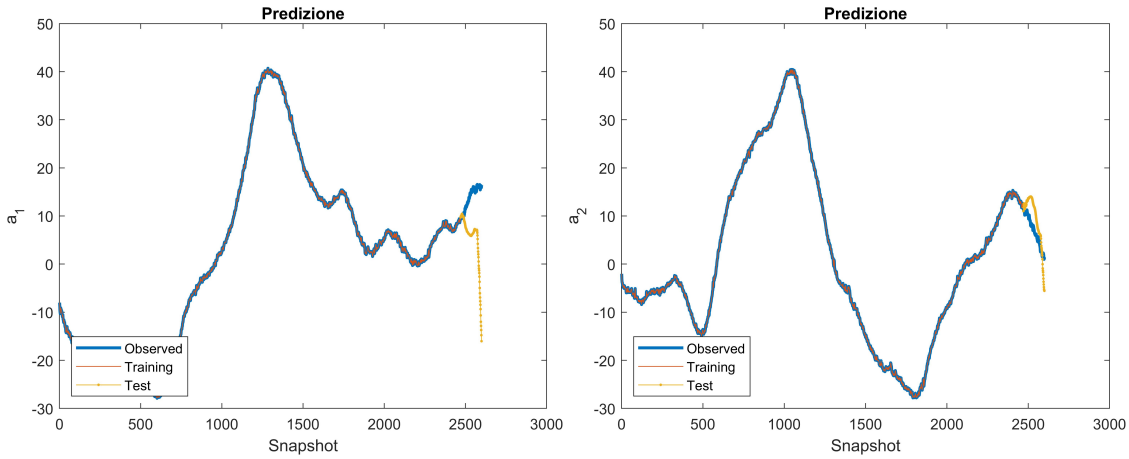
Fenomeno non periodico: Il fenomeno da cui sono estratti i modi POD non segue un andamento periodico ma si svolge caoticamente senza alcuna correlazione tra i vari istanti. Questa ipotesi è stata prontamente scartata poiché secondo quanto presentato nell'introduzione del capitolo 5.3, la convezione di Rayleigh-Bénard è fortemente periodica.

Periodo non completo: I dati a disposizione forniscono informazioni riguardo un periodo che non rappresenta appieno il periodo tipico del fenomeno. Questa ipotesi è sicuramente più accettabile rispetto alla precedente, in quanto guardando ai lavori presenti in letteratura, alcuni dei quali presentati nel capitolo 1, la predizione effettuata con le reti LSTM è eseguita su una lunga

time-history fortemente periodica. I coefficienti temporali presi in oggetto invece, nonostante siano state scattate 2600 istantanee, non evidenziano periodicità.

Effettuando un'analisi più approfondita, quanto appena esposto è avallato dai grafici presentati poc'anzi: infatti guardando alla predizione del secondo coefficiente temporale a_2 si nota come la rete riconosca un andamento simile a quanto precedentemente incontrato nell'intorno dello snapshot 1000 replicando, previa opportuna modifica dovuta alle sue variabili interne, il comportamento appreso durante l'addestramento. Essa infatti non ha mai incontrato una curva con un andamento simile a quanto ci si aspetterebbe che venisse seguito, sottolineando ancora una volta l'importanza di avere una time-history più lunga ai fini di perfezionare l'addestramento.

Al fine di ottimizzare i dati a disposizione per poter sfruttare quanti più snapshot possibili durante l'addestramento si è anche provato a modificare la percentuale di splitting tra data test e data training, fornendo 2470 snapshot alla rete LSTM per addestrarsi (pari al 95%) ed utilizzando i restanti 130 snapshot per testarla (pari al restante 5%).



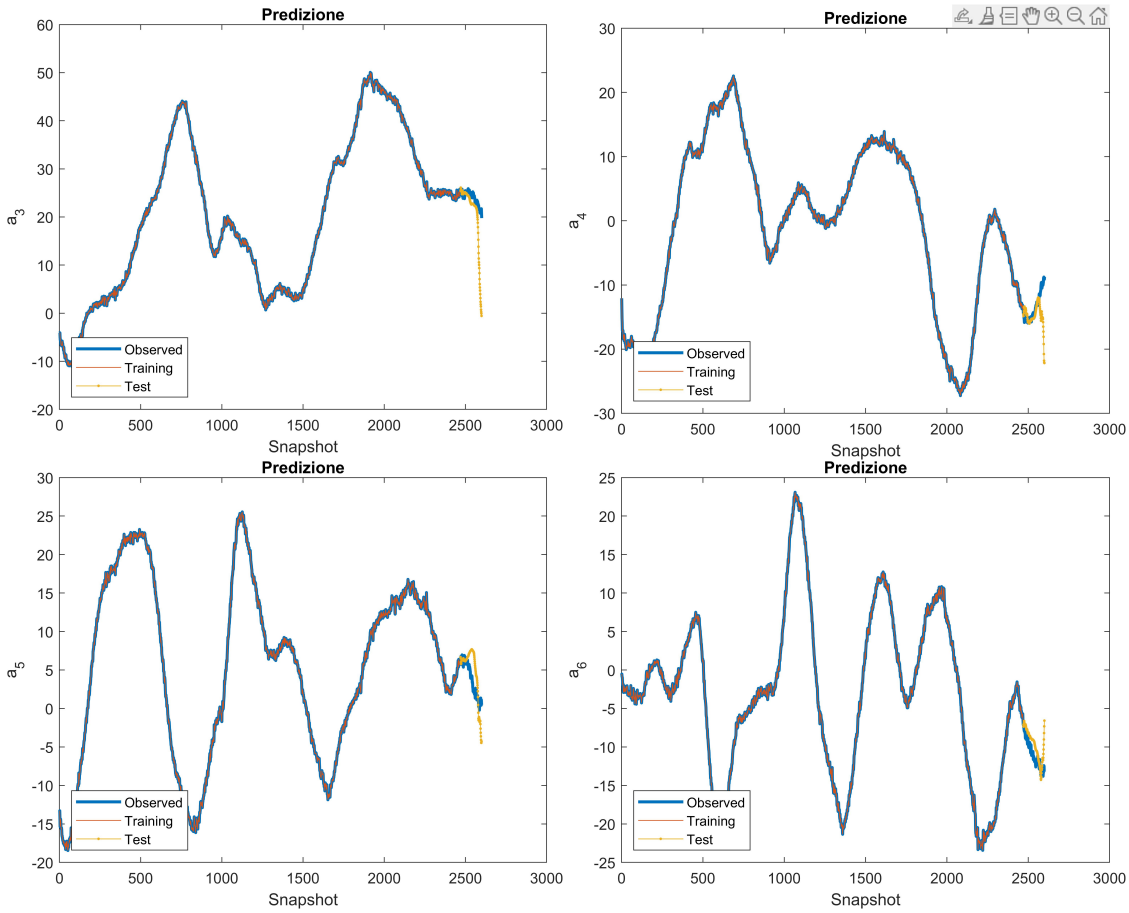
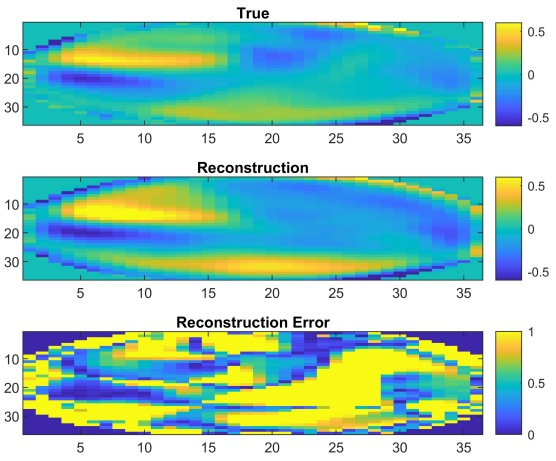
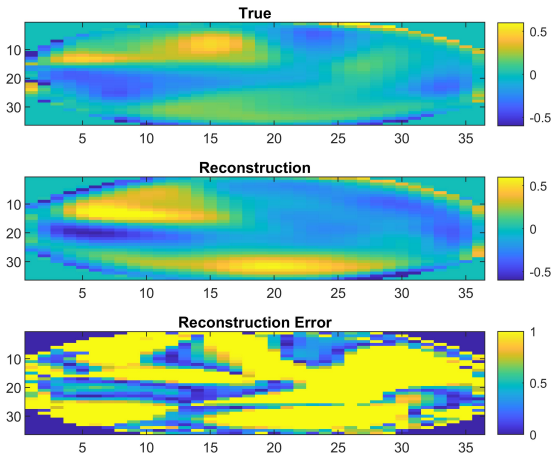


Figure 5.30: Modi POD ricostruti su 2600 Snapshot con data training modificati.

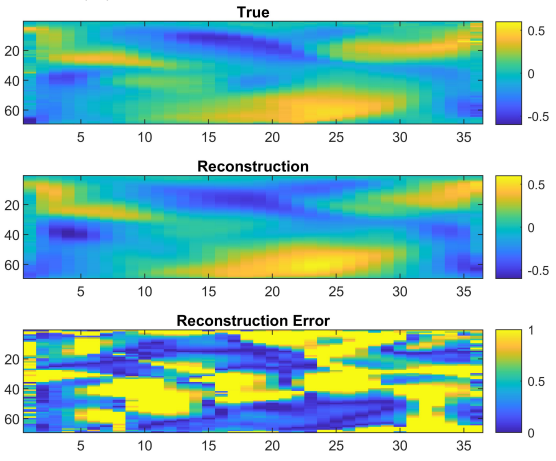
I risultati mostrati in figura 5.30 mostrano come un incremento dei dati utilizzati per l'addestramento rispecchi un miglioramento nelle capacità predittive della rete LSTM. Infatti effettuando un raffronto con quanto diagrammato in figura 5.28 si può notare come i coefficienti temporali vengano predetti con un errore molto più contenuto: ad esempio guardando al plot del coefficiente a_2, in condizioni di addestramento con minor quantità di dati si ha una importantissima divergenza che porta la predizione a non essere fedele a quanto ci si sarebbe atteso; invece guardando allo stesso coefficiente ma in condizioni di data training incrementato si nota come la parte di a_2 predetta riesca a seguire in maniera molto ravvicinata la curva che dovrebbe effettivamente tracciare.



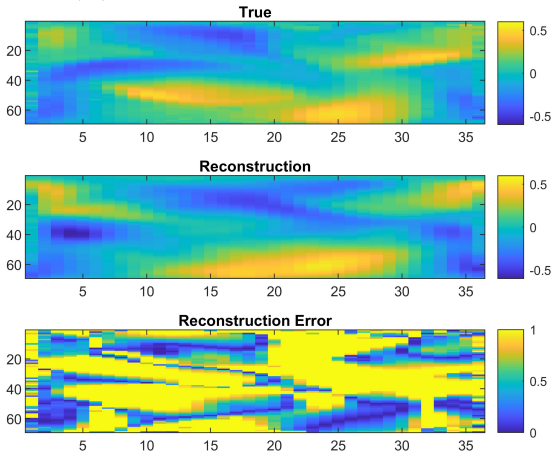
(a) 1° snapshot nel Piano Y-Z.



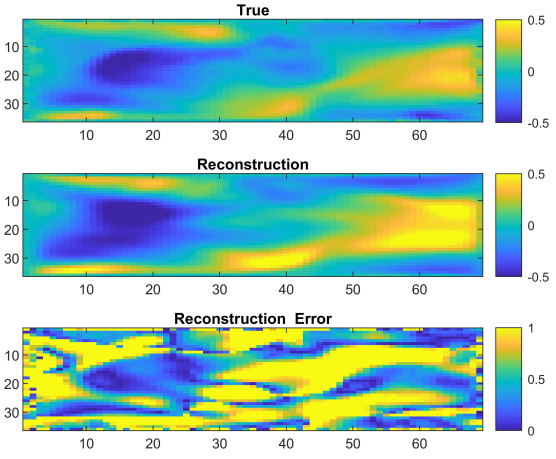
(b) 60° snapshot nel Piano Y-Z.



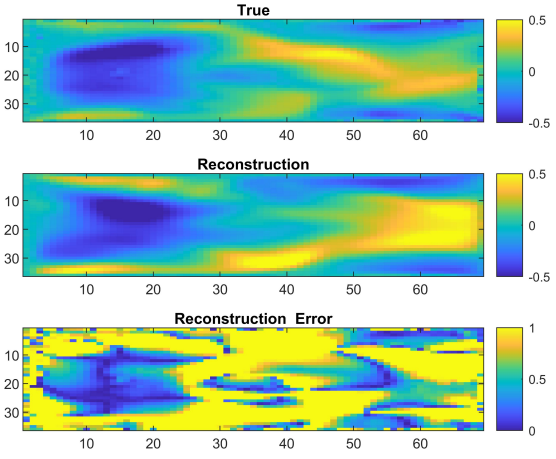
(c) 1° snapshot nel Piano X-Z



(d) 60° snapshot nel Piano X-Z.



(e) 1° snapshot nel Piano X-Y.



(f) 1° snapshot nel Piano X-Y.

Figure 5.31: Campi ricostruiti nei tre piani X-Y,X-Z,Y-Z riportanti in alto il campo reale, al centro il campo ricostruito ed in basso l'errore relativo tra i due campi.

La figura 5.31 mostra come, nonostante si siano ottenuti dei discreti risultati nella predizione dei coefficienti temporali, i campi di vorticità ricostruiti continuino a presentare degli errori che in alcuni punti risultano essere importanti. In particolare confrontando gli snapshot 1-60 di ogni piano, non si distingue alcun cambiamento tra i campi ricostruiti, cambiamento che invece è evidenziato in parte nel diagramma di errore. Effettivamente guardando al coefficiente temporale a_1 precedentemente diagrammato, si nota come la ricostruzione si discosti dall'andamento atteso e questo potrebbe riflettersi nell'errore di ricostruzione del campo vorticoso,considerando che ad a_1 è associata l'energia del flusso pari a $\sim 35\%$.

Queste affermazioni aprono ad un'ulteriore ipotesi, che sia quella di avere degli snapshot troppo vicini nel tempo rispetto all'andamento delle curve (con possibile generazione del fenomeno di overfitting). Infatti nonostante si benefici di una vasta quantità di snapshot, essi non sono distribuiti in maniera da facilitare il lavoro della rete LSTM in quanto la stessa curva potrebbe essere ricostruita con un numero molto inferiore di punti a patto di perdere le fluttuazioni che sono state eliminate in sezione 5.3.3 ma guadagnando una vasta gamma di pattern da cui la rete LSTM potrebbe trarre benefici non indifferenti.

5.4 Il seno

A causa delle ricostruzioni non molto accurate ottenute con l'approccio ibrido POD-LSTM si è deciso di tornare ad un dataset molto semplice per effettuare alcune analisi, quale il caso canonico di un'onda sinusoidale. Si vuole approfondire il comportamento della rete al fine di comprendere come migliorarne il comportamento. Si è deciso di testare l'influenza del rumore sui dati di addestramento e sopratutto la lunghezza minima di dati necessari al fine di ottenere una buona previsione. L'onda sinusoidale aiuta in questo senso, poiché avendo un andamento periodico è possibile escludere i problemi di qualsivoglia natura variando i parametri in ingresso singolarmente.

Questo approccio prevede di accantonare temporaneamente la POD poiché si ritiene che le difficoltà risiedano nella parte concernente la rete neurale, in quanto si è visto come la ricostruzione realizzata mediante coefficienti temporali a, in assenza di predizione LSTM, venga realizzata alla perfezione.

5.4.1 Onda sinusoidale con $f = 1Hz$ e normalizzazione

Come primo caso test si è utilizzata un'onda sinusoidale molto semplice con un periodo pari a $T = 1s$ per un tempo pari a $t = 5s$ con una risoluzione spaziale alta (400 punti) al fine di evitare distorsioni nella ricostruzione dell'onda (questa problematica verrà affrontata più avanti, prima con la riduzione di punti e poi con l'aggiunta di un disturbo gaussiano).

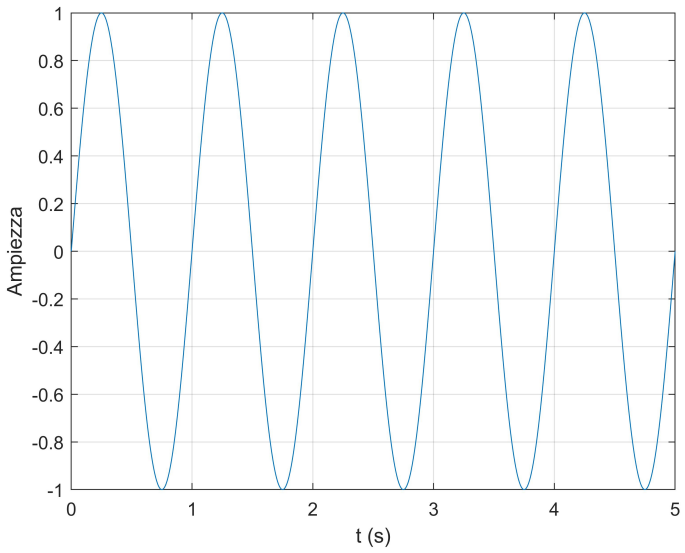


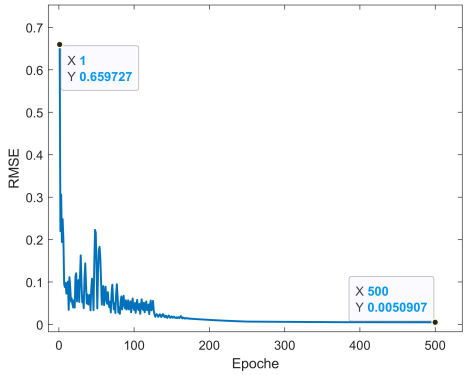
Figure 5.32: Onda sinusoidale con $f = 1Hz$ generata per un tempo di 5 secondi.

È stata anche eseguita la normalizzazione dei dati, nonostante in questo problema non sia necessario, al fine di validare anche la procedura di normalizzazione già utilizzata in precedenza. Il dataset è stato diviso secondo lo schema training 90% - 10% test (rispettivamente 360 e 40 punti).

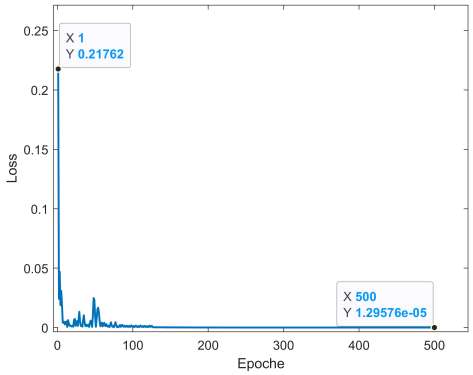
In seguito si sono normalizzati i dati utilizzando la seguente funzione:

```
1     function value=norm_fun(a)
2     value.Max = max(a,[],2);
3     value.Min = min(a,[],2);
4
5     value.norm_value = (a-value.Min)./(value.Max-value.Min);
6 end
```

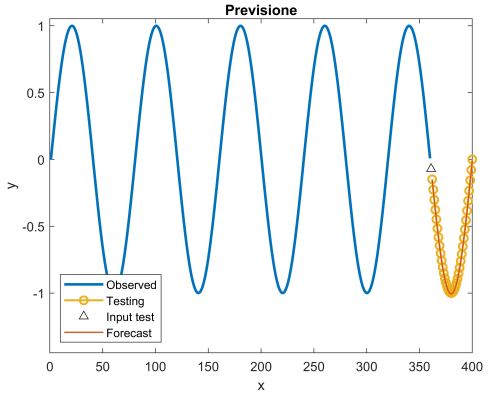
Essa prende in input un dataset, ne cerca il massimo ed il minimo e sottrae ad ogni valore il minimo, dividendo poi per la differenza tra massimo e minimo, in modo da far rientrare qualunque dato nell'intervallo 0 - 1. Di seguito vengono riportati i risultati ottenuti con una rete LSTM con 200 hidden units.



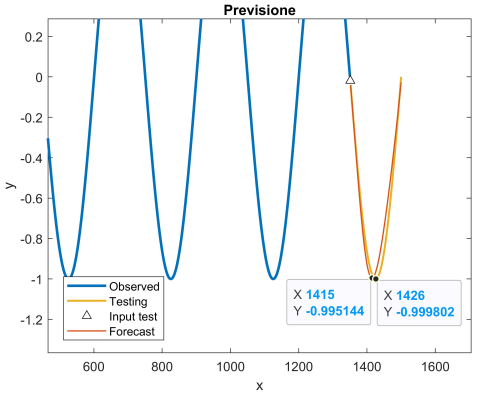
(a) RMSE addestramento rete LSTM per 500 epoche.



(b) Funzione Loss della rete LSTM per 500 epoche.



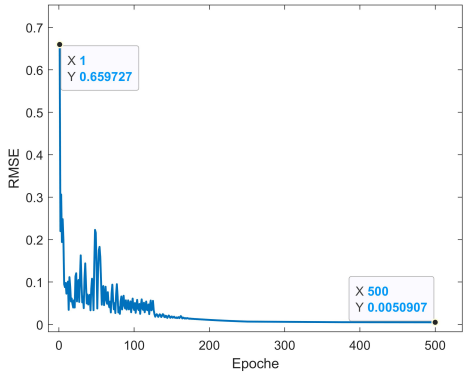
(a) Seno ricostruito con 400 punti



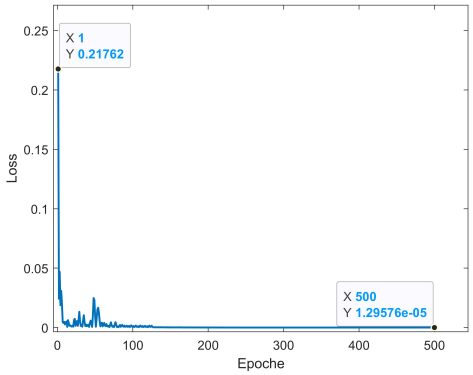
(b) Seno ricostruito con 1500 punti. Il punto X=1426 è relativo alla curva di testing, mentre il punto X=1415 appartiene alla funzione *forecast*

.

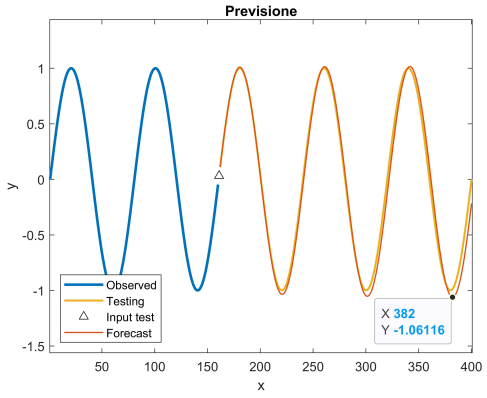
Come si può notare in figura 5.34a la funzione seno predetta dalla rete LSTM (in arancione) segue perfettamente la funzione seno prestabilita. Aumentando però il numero di punti forniti alla rete si può notare in figura 5.34b come la predizione inizi a mostrare una certa inaccuratezza, per certi versi inaspettata data la semplicità della funzione studiata. Tale comportamento risulta essere molto amplificato se la rete LSTM viene addestrata con una percentuale inferiore di dati: di seguito si riportano i dati di addestramento relativi sempre alla stessa funzione sinusoidale ma con una diversa ripartizione del dataset tra dati di training e dati di test, in particolare vengono forniti due periodi come training, ritenuti più che sufficienti al fine di una previsione accurata.



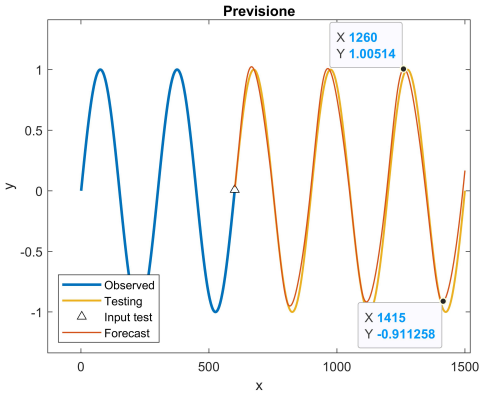
(a) RMSE per 500 epoche.



(b) Funzione Loss per 500 epoche.



(a) Seno ricostruito con 400 punti

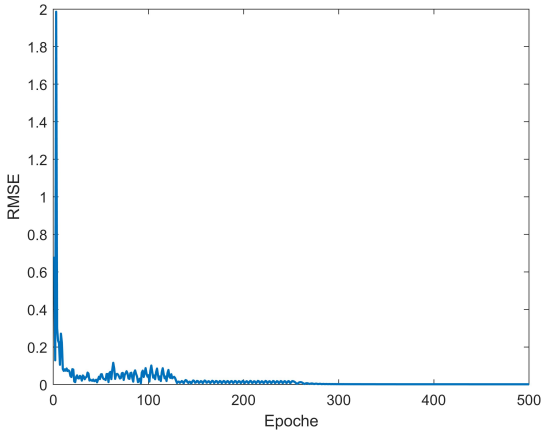


(b) Seno ricostruito con 1500 punti. I punti evidenziati sono relativi alla funzione *forecast*

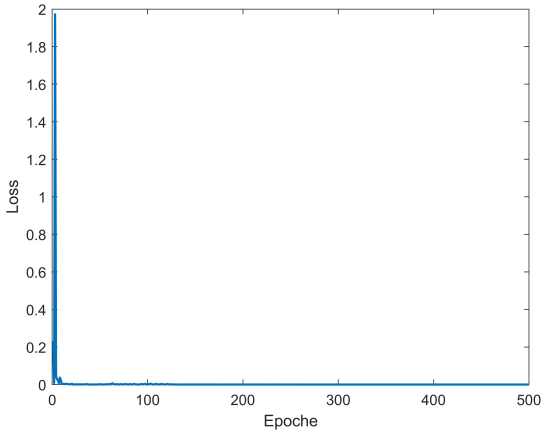
.

Come ci si aspettava la funzione predetta dalla rete peggiora, come evidenziato nella figura 5.36a, tuttavia ci si sarebbe aspettati che la rete avrebbe continuato a predirre nell'intervallo 0-1 sia perché la funzione originale prevedeva quest'intervallo, ed a maggior ragione a seguito della normalizzazione dei dati stessi. Invece si evidenzia come la funzione superi il valore -1 arrivando al valore minimo di -1.06. In figura 5.34b invece vi è un'inversione di tendenza, infatti viene raggiunto il valore massimo pari a 1.005 ma il valore minimo sale a -0.91.

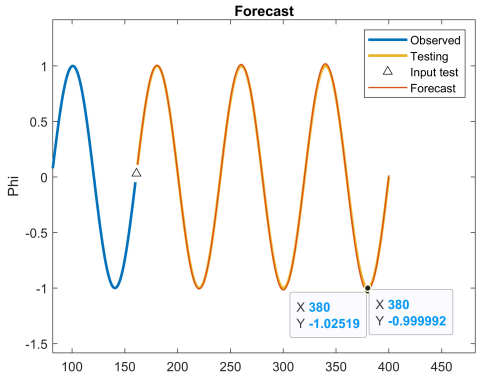
Nell'ipotizzare quali potessero essere le cause, sono stati ripetuti i test senza effettuare la normalizzazione.



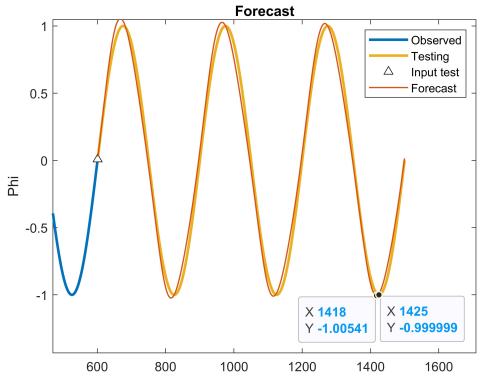
(a) RMSE per 500 epoche.



(b) Funzione Loss per 500 epoche.



(a) Seno ricostruito con 400 punti



(b) Seno ricostruito con 1500 punti. X=1418 appartiene alla linea di *forecast* a differenza di X=1425

.

I risultati sembrerebbero quindi migliorare guardando alla figura 5.38a poiché seppur si abbia una funzione $sin(x) > 1$ in generale la curva di *forecast* segue molto bene la funzione di test. Tuttavia tale comportamento non si ripete in figura 5.38b in quanto la funzione sinusoidale inizia ad essere errata sia in ampiezza che in fase: infatti seppur in figura sia evidenziato il punto più basso raggiunto (appena -1.00541) è evidente come il massimo di tale funzione sia ben oltre l'unità.

5.4.2 Gestione del rumore da parte della rete LSTM

Un aspetto meritevole di approfondimenti è la gestione del "rumore" da parte della rete LSTM. In questa sezione si vuole approfondire quale sia la distorsione massima

che essa può sopportare restituendo una ricostruzione accurata. I dataset analizzati in questa tesi magistrale infatti derivano per la maggior parte da acquisizioni sperimentali e, secondo quanto esposto nella sezione 2.3, essi sono per forza di cose affette da vari tipi di rumore. Nonostante con il troncamento realizzato con la tecnica POD sia possibile eliminare la maggior parte di questi disturbi, una parte di disturbi resterà sempre all'interno dei dati forniti alla rete neurale. Un altro aspetto che rende importante effettuare questo tipo di analisi è quello legato agli andamenti dei coefficienti temporali a: in figura 5.28 essi presentano un andamento fortemente fluttuante, il che può essere, in un certo senso, ricondotto ad un andamento medio più una componente fluttuante dovuta a rumore generico.
Per poter eseguire questa analisi è stato inizialmente creato un vettore con valori scelti casualmente in un intervallo compreso tra -0.15 e +0.15. Tale vettore viene sommato all'onda sinusoidale in modo da generare l'onda "disturbata" fittizia.

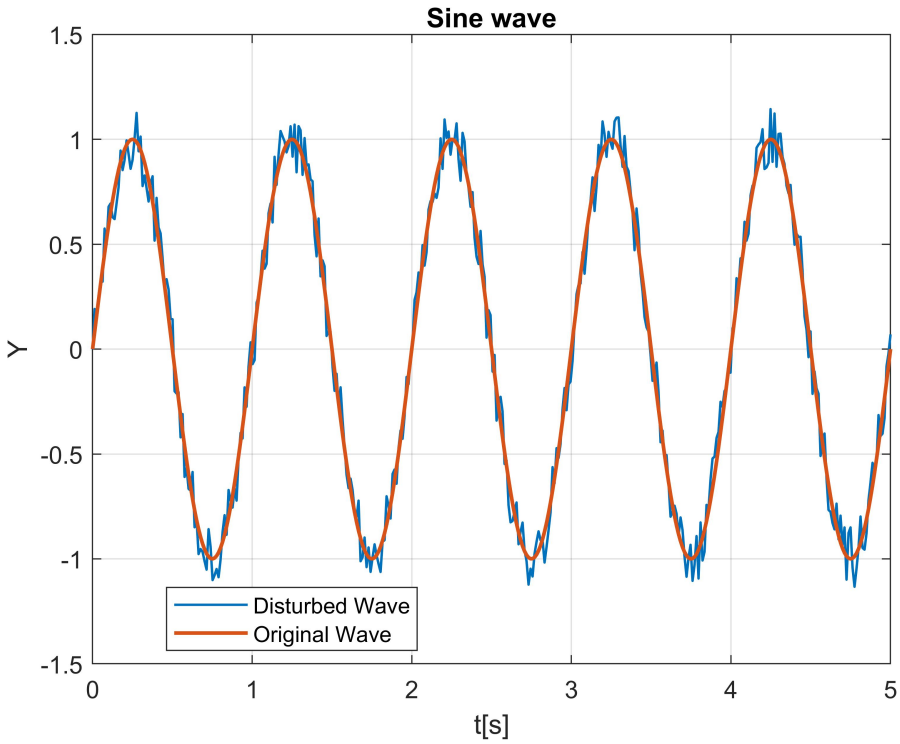
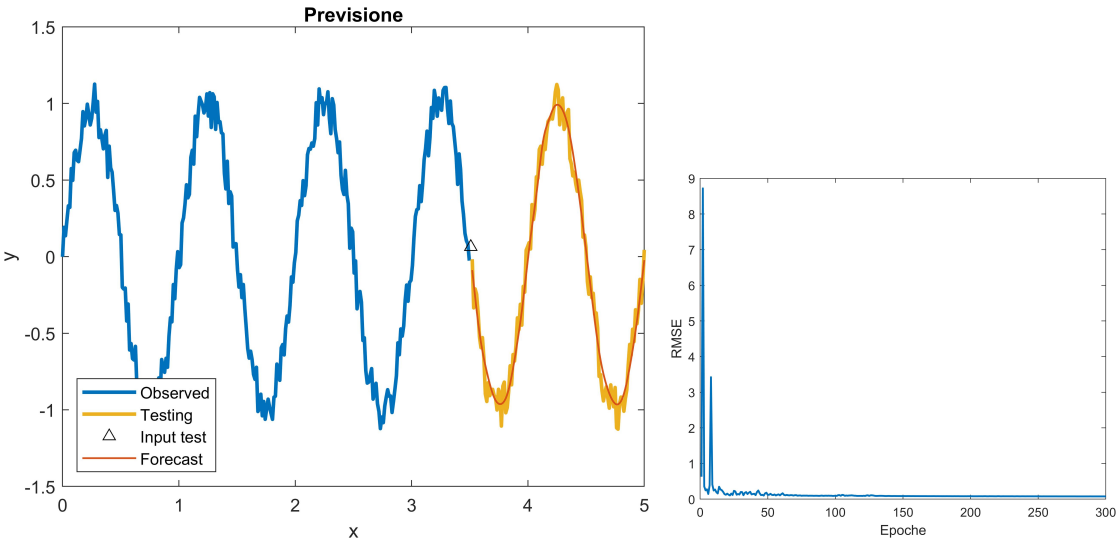


Figure 5.39: Onda sinusoidale con rumore pari a ± 0.15.

In questo caso i dati a disposizione sono stati ridotti a 400 punti, onde evitare di incorrere nel fenomeno di overfitting, successivamente divisi in data training $70\% \div 30\%$ data test. La rete LSTM è stata fornita di 500 hidden units e, data la velocità delle operazioni di calcolo, sono state effettuate iterazioni per 600 epoche.

La figura 5.40a dimostra come la rete neurale riesca a ricostruire con successo



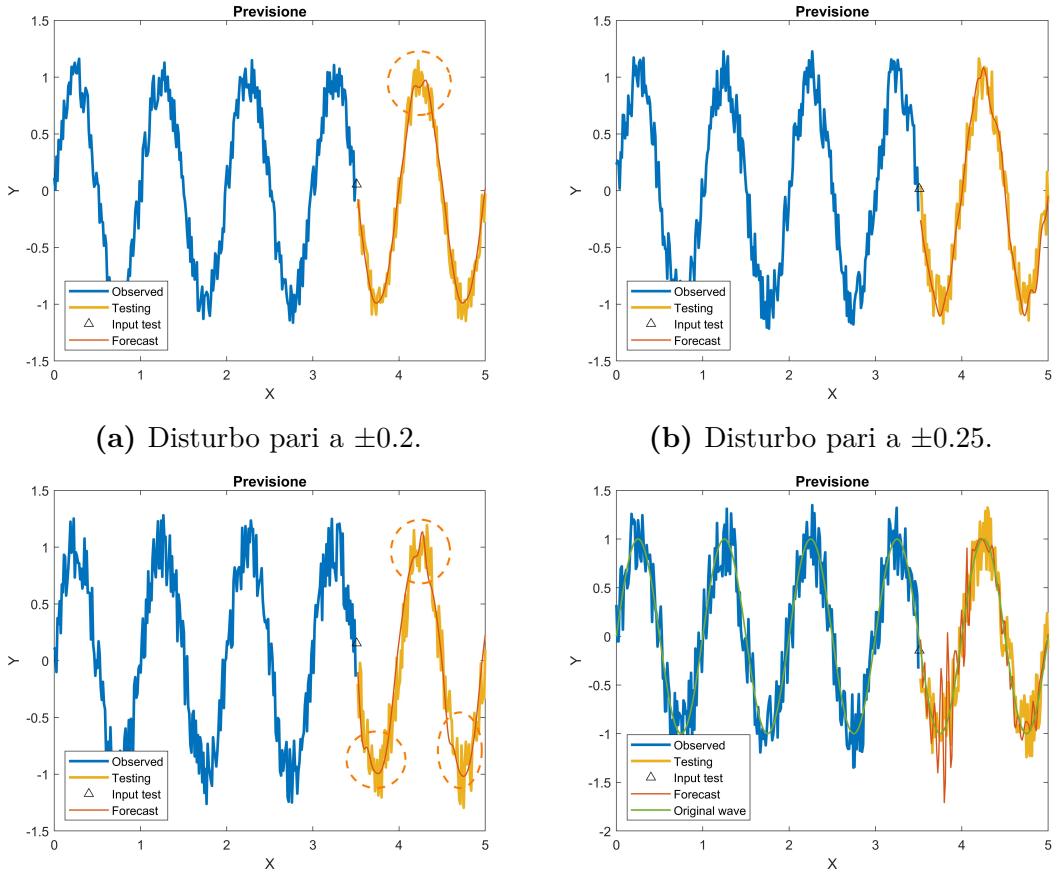
(a) Predizione andamento onda sinusoidale con disturbo pari a ±0.15 e 70% dei dati dedicati all'addestramento.

(b) Parte del training eseguito: l'RMSE decade molto velocemente.

Figure 5.40: Risultati primo tentativo con onda disturbata.

parte dell'onda sinusoidale, pari ad un periodo $T \sim 1.5$. In questo caso la rete è riuscita ad identificare l'andamento medio della curva, scorporando il rumore dall'onda sinusoidale ed agendo come una sorta di filtro aggiuntivo[3]. Appurata la possibilità di ricostruzione su questo tipo di problema è stato progressivamente aumentato il rumore sull'onda sinusoidale al fine di valutarne le capacità predittive.

[3]Si ricorda come i casi precedentemente studiati derivino per la maggior parte da acquisizioni sperimentali con un primo filtraggio effettuato dall'analisi POD.



(a) Disturbo pari a ±0.2.

(b) Disturbo pari a ±0.25.

(c) Disturbo pari a ±0.3.

(d) Disturbo pari a ±0.4. In verde l'onda sinusoidale originale.

Figure 5.41: Risultati ricostruzione onda sinusoidale con disturbo crescente.

L'aumento del disturbo pone in difficoltà le capacità predittive della rete LSTM, tuttavia esso non è risolvibile aumentando il numero di neuroni fornito alla rete, infatti si sono fornite fino a 1500 hidden units senza ottenere risultati diversi. In figura 5.41a viene cerchiata una zona in cui l'onda sinusoidale non viene ben ricostruita, con il rumore che inizia a non essere più così ben distinto rispetto al segnale originale; in figura 5.41c sono presenti dei disturbi localizzati lungo tutta la zona ricostruita segno che fa ipotizzare come la rete abbia raggiunto una situazione di "saturazione" rispetto al rumore gestibile; difatti in figura 5.41c oltre ai disturbi appena evidenziati nella precedente figura sono presenti delle fluttuazioni agli estremi della curva cerchiati in arancione; infine tutti questi disturbi provocano una vera e propria divergenza in figura 5.41d dove si ipotizza che la rete LSTM non riesca più a comprendere quali informazioni memorizzare durante l'addestramento, questo si riflette nelle sue capacità predittive dove viene ricostruita un'onda con forti

fluttuazioni e sfasamenti. In figura 5.41d si è riportato in verde l'onda sinusoidale originale per evidenziare i problemi appena descritti.

Alla luce di quanto analizzato in queste sezioni si giunge quindi alla conclusione che un numero eccessivo di punti o eccessivo rumore non è facilmente gestibile dalla rete LSTM, mentre gli errori dovuti alla normalizzazione sembrano essere trascurabili nel caso semplice di una funzione sinusoidale. Per questo motivo nella sezione seguente verrà analizzata una doppia onda sinusoidale con normalizzazione e minor quantità di punti, in modo da testare le capacità di previsione della rete neurale su due onde simultanee (si ricorda che durante le previsioni dei coefficienti temporali a si vanno a prevedere tante onde contemporaneamente quanto più il rango di troncamento è alto).

5.4.3 Doppia onda sinusoidale

Un'altra ipotesi avanzata riguardo le scarse capacità predittive della rete LSTM riguardava la possibilità che l'addestramento effettuato su più onde simultaneamente potesse essere compromesso. Per effettuare questa prova sono state utilizzate due onde sinusoidali con periodo $T = 1s$ ma con frequenze differenti e pari rispettivamente a $f = 1Hz - 1.5Hz$, come illustrato in figura 5.42.

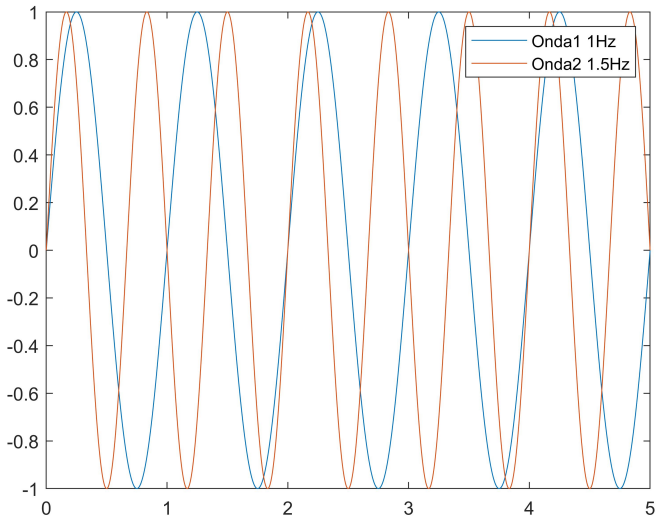
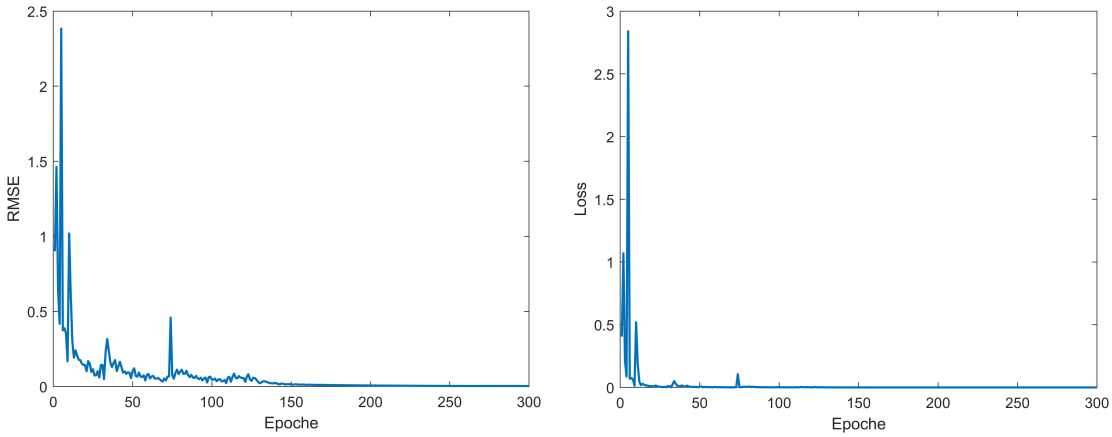


Figure 5.42: Onde sinusoidali con $f = 1 - 1.5Hz$ generate per un tempo di 5 secondi.

L'addestramento della rete neurale è stato effettuato utilizzando una quota fissata di hidden units pari a 500. Onde evitare fenomeni di overfitting le curve

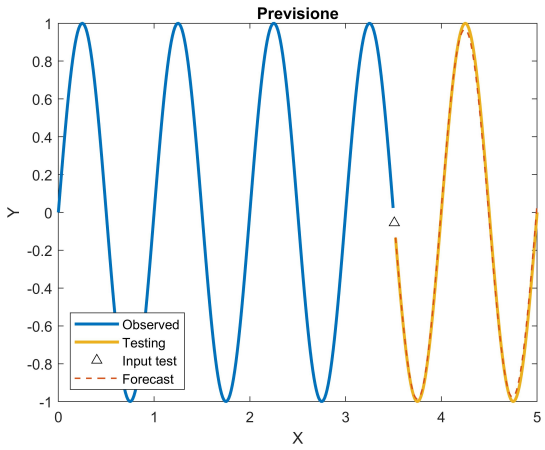
sono state generate utilizzando 400 punti. Tali punti sono stati divisi tra data training e data test secondo le percentuali 70% ÷ 30%. Al fine di verificare la correttezza della normalizzazione adoperata finora le curve sono state normalizzate, seppur non fosse strettamente necessario.



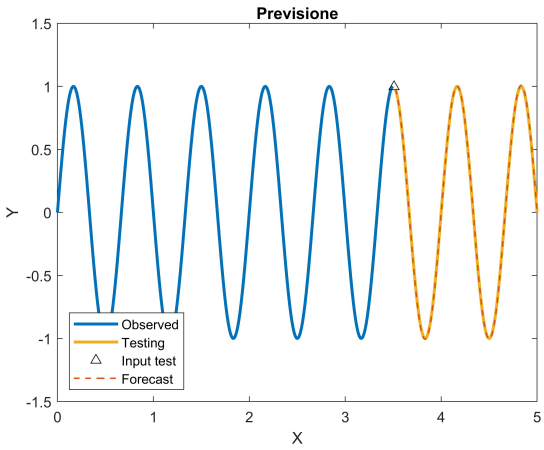
(a) RMSE per 300 epoche. **(b)** Funzione Loss per 300 epoche.

Figure 5.43: Addestramento rete LSTM per la predizione della doppia onda sinusoidale.

L'addestramento è stato svolto per 600 epoche con valore minimo di Root Mean Square Error pari a 0.0027 e valore minimo della funzione Loss pari a 3.7e-6, in figura 5.43 vengono mostrate solo il 50% delle epoche poiché si sarebbe ottenuto un grafico poco rappresentativo altrimenti.



(a) Onda sinusoidale con $f = 1Hz$ ricostruita.



(b) Onda sinusoidale con $f = 1.5Hz$ ricostruita.

Figure 5.44: Predizione andamento onde sinusoidali generate con 400 punti.

In figura 5.44 si può notare come la predizione dell'andamento di entrambe le onde sinusoidali sia stata effettuata con ottima precisione da parte della rete LSTM. Questo conferma quanto precedentemente ipotizzato, cioè che il predirre più onde contemporaneamente non vada ad inficiare sulle capacità predittive.

5.5 I limiti della predizione

Un ultimo test effettuato è stato quello di spingere al limite tali capacità predittive in condizioni favorevoli, le quali vengono ricapitolate di seguito per semplicità:

- Numero di punti forniti non eccessivo: Secondo quanto precedentemente illustrato, la rete LSTM non riesce a gestire una quantità di dati ridondanti eccessiva.

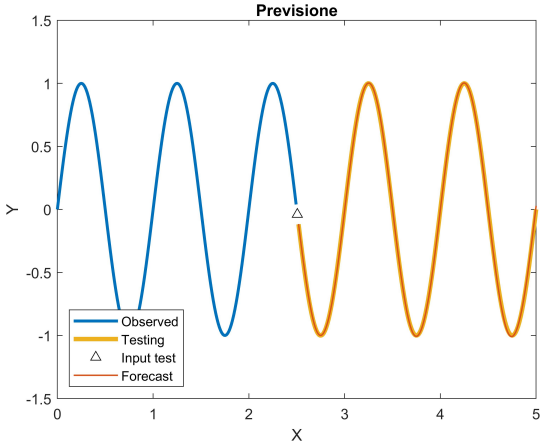
- Nessuna normalizzazione: Le operazioni di normalizzazione e denormalizzazione vanno a destabilizzare la delicata fase di predizione e ricostruzione.

- Dati non affetti da rumore: il rumore può destabilizzare, soprattutto se intenso e continuativo, le capacità del neurone LSTM di identificare le corrette peculiarità della curva al fine di effettuare una corretta ricostruzione.

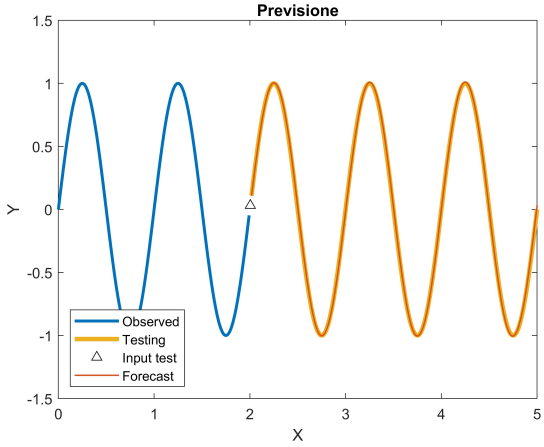
I dati a disposizione che soddisfano i tre requisiti di cui sopra sono l'onda sinusoidale ed i dati numerici relativi alla scia del cilindro 2D.

5.5.1 Test riduzione dati onda sinusoidale

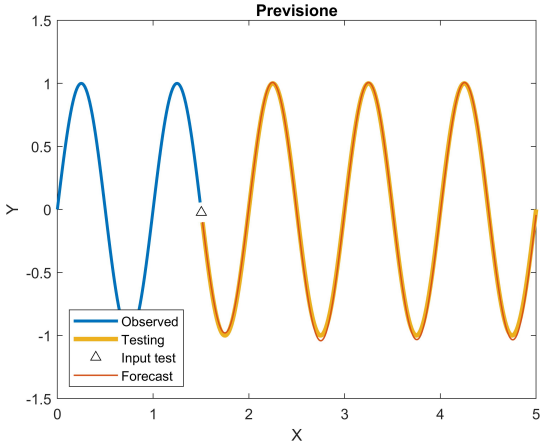
In questa sezione verrà affrontato un problema di fondamentale importanza in quanto la curva che si andrà a ricostruire sarà una delle più semplici che si possano incontrare in campo fluidodinamico. Inoltre tenendo anche conto delle semplificazioni effettuate, i risultati ottenuti forniranno una stima affidabile su quanto sia possibile spingersi in termini di riduzione dei dati di training, poiché un'errata predizione su questo problema si tradurrebbe sicuramente in errori molto più ampi su problemi complessi.



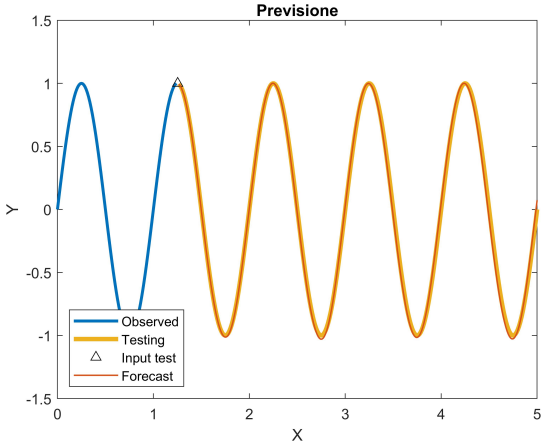
(a) Onda sinusoidale ricostruita con il 50% dei dati(T=2.00).



(b) Onda sinusoidale ricostruita con il 40% dei dati(T=1.75).



(c) Onda sinusoidale ricostruita con il 30% dei dati (T=1.5).

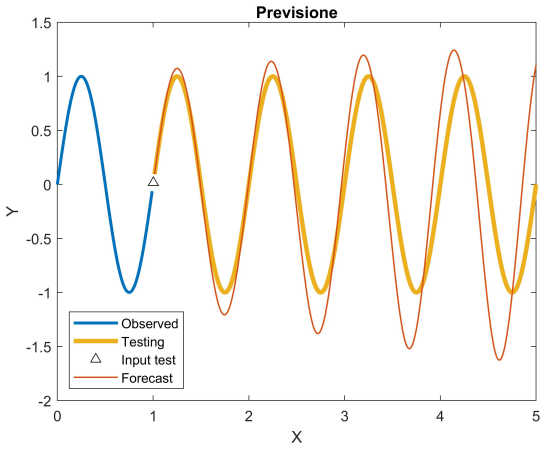


(d) Onda sinusoidale ricostruita con il 25% dei dati (T=1.25).

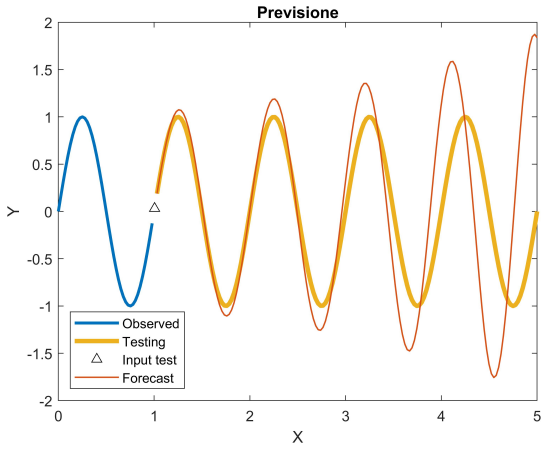
Figure 5.45: Predizioni andamento onda sinusoidale effettuate con successo.

In figura 5.45 sono rappresentate le ricostruzioni effettuate con successo dalla

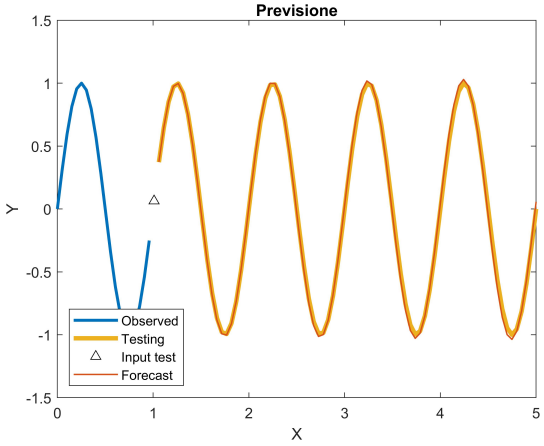
rete neurale precedentemente descritta sull'onda sinusoidale costituita da 400 punti. Ulteriori diminuzioni dei dati di addestramento permetterebbero una corretta ricostruzione solo riducendo il numero di punti con cui viene costruita l'onda.



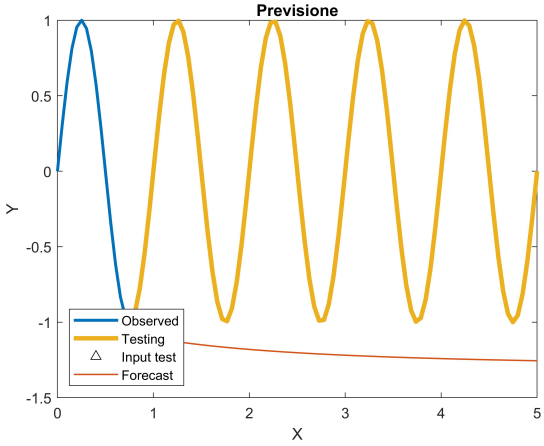
(a) Onda sinusoidale non ricostruita con il 20% dei dati e 400 punti.



(b) Onda sinusoidale non ricostruita con il 20% dei dati e 200 punti.



(c) Onda sinusoidale ricostruita con il 20% dei dati e 100 punti (T=1).



(d) Onda sinusoidale non ricostruita con il 15% dei dati (T=0.75).

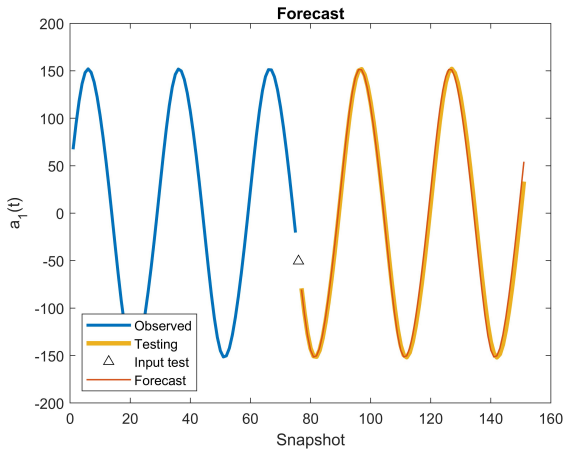
Figure 5.46: Limite predizioni andamento onda sinusoidale.

L'onda viene ricostruita solo riducendo il numero di punti come illustrato in figura 5.46c, il che si traduce in una evidente difficoltà nel rappresentare le zone con minor raggio di curvatura, tuttavia si ritiene accettabile questa piccola perdita di precisione localizzata se confrontata con la possibilità di ottenere predizioni affidabili. Questo consente di affermare che è possibile utilizzare un solo periodo di onda sinusoidale per ricostruirne il proseguo, a patto di porre la rete LSTM in condizioni da evitare fenomeni di overfitting. Non è possibile spingersi oltre come

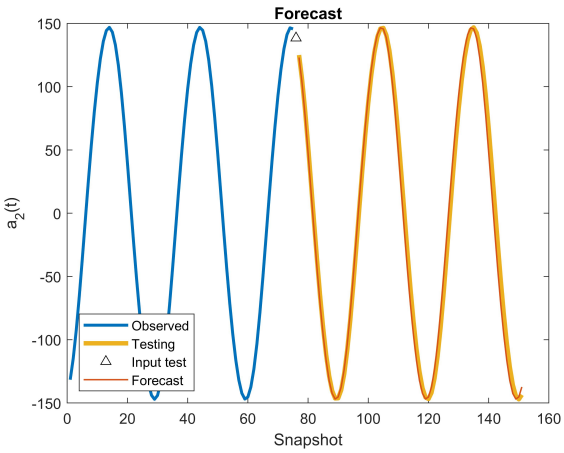
illustrato in figura 5.46d poiché la rete si trova ad affrontare una porzione di curva mai vista durante l'addestramento con conseguente andamento pseudo-casuale.

5.5.2 Test riduzione dati scia cilindro

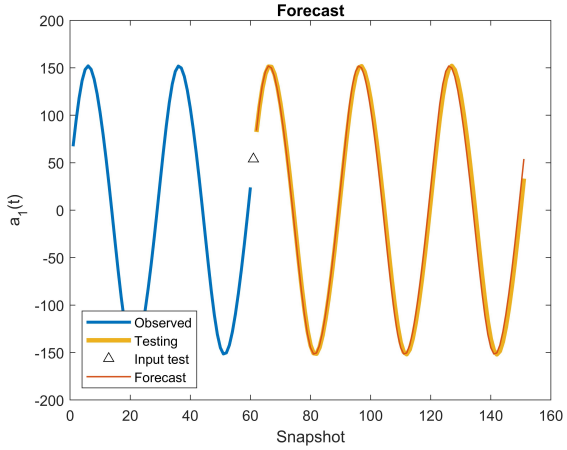
In questa sezione si cercheranno di replicare i risultati ottenuti nel caso dell'onda sinusoidale 1D per un campo bidimensionale. Il caso preso in esame è il campo di vorticità di un cilindro investito da flusso con Reynolds tale da generare il fenomeno del vortex shedding, già trattato in sezione 5.1: esso è ricostruito utilizzando solo sei modi POD e, per le analisi necessarie in questo capitolo, i coefficienti temporali $a_k \ ; k = 1, ...,6$ non sono stati normalizzati, coerentemente con quanto precedentemente esposto. Rispetto al caso unidimensionale è anche possibile visualizzare i campi ricostruiti al fine di valutare l'effettiva correttezza dei risultati.



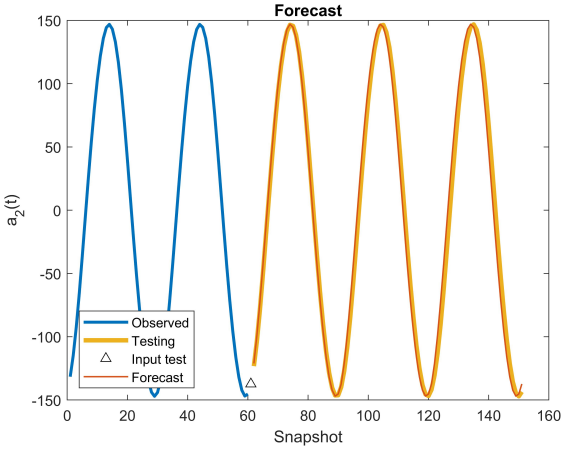
(a) Coefficiente temporale a_1 con dataset al 50%.



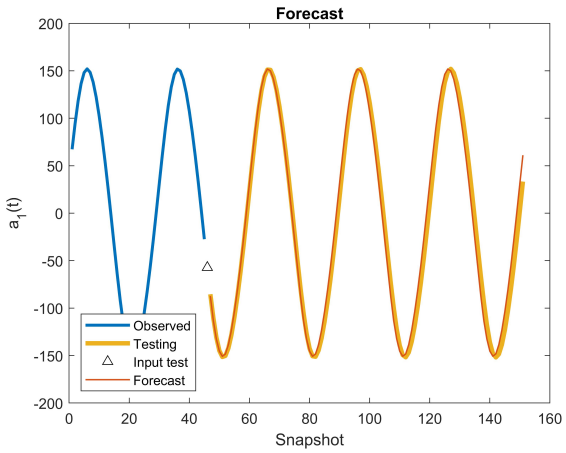
(b) Coefficiente temporale a_2 con dataset al 50%.



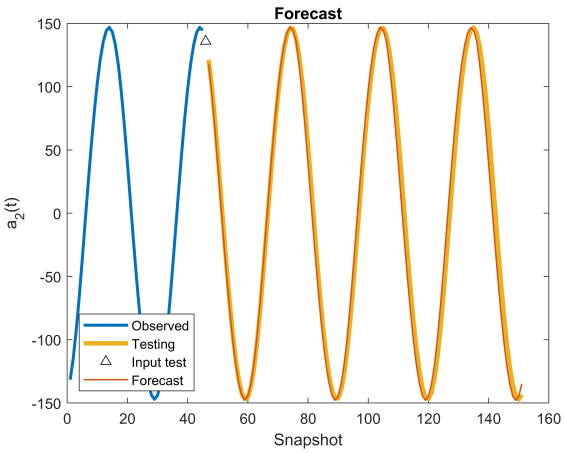
(c) Coefficiente temporale a_1 con dataset al 40%.



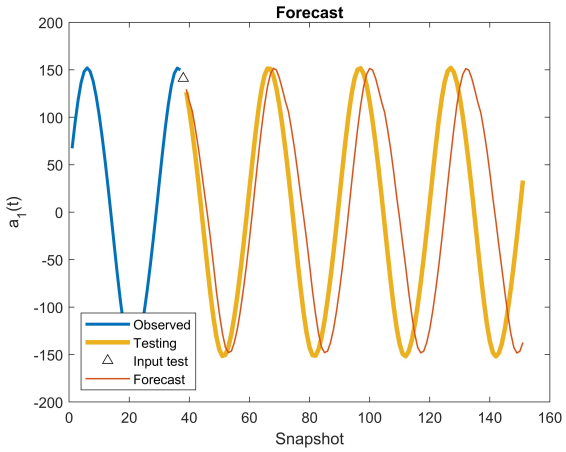
(d) Coefficiente temporale a_2 con dataset al 40%.



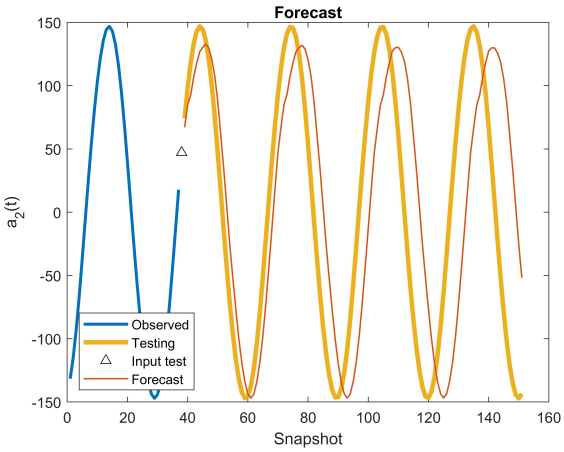
(a) Coefficiente temporale a_1 con dataset al 30%.



(b) Coefficiente temporale a_2 con dataset al 30%.



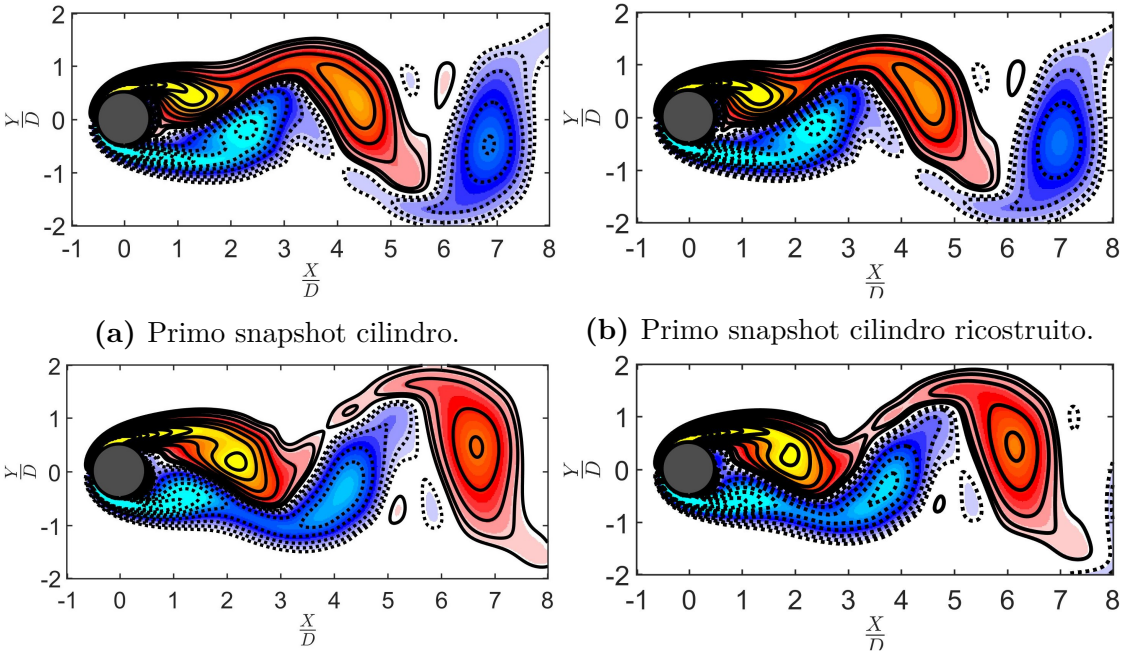
(a) Coefficiente temporale a_1 con dataset al 25%.



(b) Coefficiente temporale a_2 con dataset al 25%.

Figure 5.49: Limite predizioni coefficienti temporali a_1 , a_2.

In figura 5.49 vengono riportati i risultati relativi ai coefficienti temporali a, dove si evince chiaramente come il limite inferiore sia di circa un periodo e mezzo di onda. Infatti destinando solo il 25% del dataset all'addestramento si evidenzia un comportamento sfasato rispetto alla curva che si dovrebbe seguire. Inoltre nonostante i coefficienti temporali risultino essere ben ricostruiti per una percentuale del 30% dei dati destinati all'addestramento, è presente un leggero sfasamento tra la curva predetta e la curva effettiva.



(a) Primo snapshot cilindro.

(b) Primo snapshot cilindro ricostruito.

(c) Ultimo snapshot cilindro.

(d) Ultimo snapshot cilindro ricostruito.

Figure 5.50: Campi di vorticità relativi al cilindro con 45 snapshot (su 151 disponibili) dedicati all'addestramento.

Impercettibilmente, lo sfasamento evidenziato nella ricostruzione dei coefficienti temporali si ripercuote anche nella predizione dei campi di moto come è possibile notare osservando attentamente le figure 5.50c e 5.50d. Tale sfasamento non è invece rilevato nella predizione dei campi temporali relativi al primo snapshot di test (46° snapshot del dataset), a testimonianza della ottima capacità di predirre delle reti LSTM se poste nelle condizioni adeguate. Inoltre si ricorda come vengano utilizzati sei modi POD per la ricostruzione, per cui lo sfasamento potrebbe essere anche ricondotto agli ulteriori 4 modi, che secondo la decomposizione POD rappresentano l'alternanza tra i vortici. Tuttavia in precedenza si è scelto di diagrammare solo i primi due modi, che rappresentano circa l'80% del contenuto energetico, per evitare di appesantire la trattazione.

5.6 Conclusioni e sviluppi futuri

L'approccio ibrido POD-LSTM viene proposto per riuscire a ridurre i tempi necessari ad effettuare simulazioni fluidodinamiche per cui potrebbero anche essere necessari dei giorni, senza considerare che sarebbe necessario effettuare nuove simulazioni partendo da zero nel caso fosse necessario modificare dei parametri

relativi al campo di moto, o peggio, effettuare delle piccole variazioni geometriche che comporterebbero l'intera revisione del modello computazionale. Il metodo analizzato in questa tesi, una volta effettuato l'addestramento iniziale della rete neurale LSTM è in grado di rispondere agli input che riceve la rete generando gli output desiderati in pochi istanti. Il costo principale da affrontare, come esposto nei risultati presentati, è quello relativo all'addestramento: infatti vi è la necessità di fornire una matrice di dati contenenti informazioni essenziali (non ridondanti) per generare in risposta l'output desiderato. Il costo di tale operazione non è fisso, ma variabile in funzione principalmente della complessità del campo vettoriale studiato e della precisione che si vuole ottenere; inizialmente potrebbe anche risultare più oneroso rispetto alle comuni analisi CFD, ma si avrà comunque un guadagno in termini di versatilità a lungo termine. Trattandosi di un approccio basato su una metodologia che coinvolge a sua volta un metodo statistico-matematico (POD) ed una nuovissima metodologia computazionale (LSTM) è stato importante comprendere se tale sinergia tra i due potesse avvenire in diversi campi senza incorrere in incompatibilità.

I risultati ottenuti mostrano una tecnica sicuramente valida per quanto riguarda la parte di analisi POD. Si è dimostrato come essa permetta di ridurre sensibilmente i dati in ingresso insieme con il rumore acquisito ed il costo computazionale. La maggior parte dei problemi si sono concentrati intorno alla rete neurale LSTM, poiché se da un lato la POD ha una procedura ben prestabilita, l'addestramento e l'implementazione della rete LSTM dipendono da molte variabili: il primo infatti rappresenta un tassello fondamentale per la buona riuscita della predizione. La fase di addestramento è risultata particolarmente sensibile a:

- **Dati in ingresso:** In letteratura si fa cenno rispetto alla natura dei dati, i quali devono contenere informazioni utili per la rete in modo che possa generare output corretti, tuttavia l'autore non ha riscontrato feedback riguardo quanto questi dati possano essere fitti, poiché si è dimostrato come una quantità eccessiva di dati, seppur forniscano maggiori informazioni alla rete, rischino invece di risultare controproducenti. Si rischia inoltre il fenomeno di overfitting.

- **Architettura rete:** Se da un lato l'aumento di neuroni fornisce potenzialmente maggiore memoria e capacità di apprendimento alla rete, dall'altro aumenta la complessità della stessa con conseguente difficoltà all'addestramento. Nei vari tentativi di predizione effettuati nel corso di questa tesi magistrale si è notato come un banale problema fosse risolvibile utilizzando pochi neuroni (ordine delle centinaia). Affrontando lo stesso problema con una rete formata da migliaia di neuroni l'addestramento non risultava più efficace come prima e, seppur il *Root Mean Square Error* risultasse basso, si ottenevano ricostruzioni errate con andamenti che si focalizzavano più sulle fluttuazioni che sull'andamento medio, come se la rete non riuscisse ad avere una visione

di insieme a causa della non perfetta sinergia tra tutti i neuroni. Questo fenomeno si presentava anche aumentando il numero di "LSTM Layer" presenti all'interno della rete.

Allo stato attuale l'approccio ibrido POD-LSTM con la rete addestrata in questo lavoro è utilizzabile per predirre con buoni risultati degli andamenti fortemente oscillatori e protratti nel tempo, con un numero di punti non eccessivo. Si ritiene che la predizione di campi più complicati possa essere possibile focalizzando gli studi sui seguenti punti:

- Deep Neural Network: Un incremento nella difficoltà del problema necessariamente deve essere affrontato con un incremento nella complessità della rete, che da un lato consente di ottenere più memoria e più potenza computazionale ma dall'altro rischia di provocare facilmente overfitting. Vanno quindi approfonditi i meccanismi di interazione al fine di organizzare un addestramento che possa risultare efficace.

- Funzioni di attivazione: La rete LSTM in questo lavoro adopera solo funzioni di attivazione di tipo sigmoide ed a tangente iperbolica. Tuttavia in letteratura sono presenti molte altre alternative, Farzad et al. ne elencano 23 con annessi studi su come potrebbero migliorare le prestazioni. Inoltre Tondak nel suo articolo scrive espressamente come la funzione di tangente iperbolica non permetta una perfetta ricostruzione di sequenze molto lunghe.

È inoltre possibile affinare ulteriormente le capacità della rete LSTM approfondendo concetti minori quali la scelta di funzioni di normalizzazione univoche, "sfoltimento" di dati se in presenza di quantità eccessive e modifica alle tecniche di addestramento per problemi particolarmente complessi.

Al momento della scrittura di questo lavoro, non risultano noti all'autore approcci ibridi POD-LSTM dedicati alla predizione di campi fluidodinamici, eccezion fatta per il lavoro di Rahman et al. il quale si basa comunque su campi quasi-geostrofici di grande scala, mentre in questo lavoro ci si è focalizzati su problemi più interessanti ai fini dello studio aerodinamico e fluidodinamico. Una volta eseguito un buon addestramento ed ottenuta una rete funzionante, sarà possibile esplorare ogni variazione parametrica che rientri nei dati di addestramento, con risultati pressoché immediati. Questo apre le porte all'implementazione di tale modello in un sistema di controllo aerodinamico attivo, sia in ambito aeronautico che automobilistico: una rete addestrata con dati sufficienti potrebbe essere in grado di comprendere, ottenendo dati da sensori posizionati in opportune sedi, la condizione in cui ci si trova a viaggiare e prevedere come evolverà la stessa, implementando anticipatamente delle tecniche volte a ridurre la resistenza all'avanzamento mediante attuatori

oppure, ad esempio in vetture da competizione, modificando l'assetto aerodinamico al fine di aumentare la deportanza. Le possibilità di tale tecnica spaziano anche sul miglioramento di tecniche già esistenti: Deng et al., partendo da un campo quasi time-resolved ottenuto tramite tecnica ottica PIV($f = 5\,Hz$), sono riusciti a renderlo time-resolved soddisfando il teorema di Nyquist-Shannon[4]. Infatti la frequenza tipica del problema analizzato è $\sim 2.8\,Hz$ per cui è necessario acquisire ad una frequenza di almeno $\sim 5.6\,Hz$. Un interessante approccio sarebbe quello di aumentare i punti ad alta risoluzione al fine di effettuare un sostanziale aumento della risoluzione della tecnica PIV.

[4]Tale teorema sostiene la necessità di effettuare acquisizioni ad almeno una frequenza doppia rispetto alla frequenza tipica del sistema.

Ringraziamenti

La mia carriera universitaria non è stata facile, compresa la stesura di questa tesi. Non ce l'avrei mai fatta da solo, per questo ho alcuni ringraziamenti da fare.

Vorrei ringraziare il Prof. Gaetano Iuso e l'Ing. Gioacchino Cafiero per avermi seguito durante il mio percorso di tesi e per avermi concesso una grande opportunità di apprendimento all'interno del laboratorio di aeronautica. Ringrazio anche l'Ing. Enrico Amico per essere stato sempre pronto ad aiutarmi durante l'intero percorso di tesi ed avermi affiancato durante gli esperimenti.

Grazie ad Andrea, Ciccio, Daniele, Enrico, Gabriele, Giulia, Ilaria, Lorenzo e tutti gli altri amici per avermi sostenuto quando pensavo di non farcela, per avermi ascoltato e consigliato, per la spensieratezza con una passeggiata, un pranzo insieme o semplicemente una folle uscita in notturna. Grazie a Giovanni per essere stato non solo un coinquilino, ma un migliore amico che è sempre stato presente e su cui ho sempre potuto fare affidamento. Un grazie anche a quelle persone che non hanno un nome, ma che in un modo o nell'altro mi hanno aiutato durante la vita di tutti i giorni e, alcune più di altre, mi hanno insegnato delle lezioni di vita che porterò sempre con me.

Grazie a Padre Tonino per essere sempre lì pronto ad aiutarmi con i suoi saggi consigli e la sua preghiera, perché certi traguardi sono impossibili da raggiungere senza un aiuto dall'alto.

Grazie a tutti gli zii, cugini, ex vicini di casa che sono stati presenti nella mia vita durante il mio percorso universitario e che in un modo o nell'altro mi hanno aiutato ad andare avanti, con un allenamento in palestra, un weekend insieme, un giro in bici o più semplicemente scambiando quattro chiacchiere.

Grazie alla mia famiglia, Papà Mamma e Luca, per l'aiuto sia morale ma anche economico, frutto di importanti sacrifici. Per avermi sempre sostenuto ed incoraggiato quando le cose non andavano bene ed essere stati il mio porto sicuro durante

le tempeste. Probabilmente non ve ne sarete resi conto, ma anche solo la vostra presenza è stata per me uno sprone per andare avanti, cercando di rendervi sempre orgogliosi di me.

Infine un grazie a me, perché so cosa ho passato in tutti questi anni, quello che ho dovuto superare e quanto ho subito, ma non sono mai caduto. Sono andato avanti ripetendo "Dream it Possible". Ho sempre trovato il modo per rialzarmi e non ho mai fatto spegnere la fiamma anche nei momenti più bui, nonostante mille problemi sono sempre andato avanti, per non deludere chi crede in me, ricordandomi poi, che anch'io sono uno di quelli. .

Probabilmente ho dimenticato qualcuno da ringraziare, ma sono grato ad ogni persona che mi possa aver aiutato in questo percorso, perché è facile stare dalla parte dei vincenti, ma non è facile farlo quando perdono.

Bibliografia

[1] Stephen B Pope. *Turbulent Flows. The Science of Microfabrication*. Cambridge University Press, 2000 (cit. on pp. 1, 2).

[2] David J. Lucia, Philip S. Beran, and Walter A. Silva. «Reduced-order modeling: new approaches for computational physics». In: *Progress in Aerospace Sciences* 40.1 (2004), pp. 51–117. ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2003.12.001 (cit. on p. 2).

[3] Kuan Lu, Kangyu Zhang, Haopeng Zhang, Xiaohui Gu, Yulin Jin, Shibo Zhao, Chao Fu, and Yongfeng Yang. «A Review of Model Order Reduction Methods for Large-Scale Structure Systems». In: *Shock and Vibration* 2021 (May 2021), e6631180. DOI: 10.1155/2021/6631180 (cit. on p. 2).

[4] G Berkooz, P Holmes, and J L Lumley. «The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows». In: *Annual Review of Fluid Mechanics* 25.1 (1993), pp. 539–575. DOI: 10.1146/annurev.fl.25.010193.002543 (cit. on p. 2).

[5] Julien Weiss. «A tutorial on the proper orthogonal decomposition». In: (2019), p. 3333 (cit. on pp. 3, 9).

[6] C.E. Frouzakis, Y.G. Kevrekidis, J. Lee, K. Boulouchos, and A.A. Alonso. «Proper orthogonal decomposition of direct numerical simulation data: Data reduction and observer construction». In: *Proceedings of the Combustion Institute* 28.1 (2000), pp. 75–81. ISSN: 1540-7489. DOI: https://doi.org/10.1016/S0082-0784(00)80197-6 (cit. on pp. 3, 4).

[7] D. del-Castillo-Negrete, D. A. Spong, and S. P. Hirshman. «Proper orthogonal decomposition methods for noise reduction in particle-based transport calculations». In: *Physics of Plasmas* 15.9 (2008), p. 092308. DOI: 10.1063/1.2979680 (cit. on pp. 4, 5).

[8] Sk M Rahman, Suraj Pawar, Omer San, Adil Rasheed, and Traian Iliescu. «Nonintrusive reduced order modeling framework for quasigeostrophic turbulence». In: *Physical Review E* 100.5 (2019), p. 053306 (cit. on pp. 5, 78).

[9] Aravind Pai. URL: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/ (cit. on p. 7).

[10] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 8, 18).

[11] Y. Bengio, P. Simard, and P. Frasconi. «Learning long-term dependencies with gradient descent is difficult». In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181 (cit. on p. 8).

[12] JL Lumley. «Rational approach to relations between motions of differing scales in turbulent flows». In: *The Physics of Fluids* 10.7 (1967), pp. 1405–1408 (cit. on p. 9).

[13] Kunihiko Taira et al. «Modal analysis of fluid flows: An overview». In: *Aiaa Journal* 55.12 (2017), pp. 4013–4041 (cit. on pp. 10, 14).

[14] Lawrence Sirovich. «Turbulence and the dynamics of coherent structures. II. Symmetries and transformations». In: *Quarterly of Applied mathematics* 45.3 (1987), pp. 573–582 (cit. on p. 12).

[15] URL: https://it.wikipedia.org/wiki/Rete_neurale_ricorrente (cit. on p. 17).

[16] Cristiano Casadei. *Le reti neurali ricorrenti*. URL: https://www.developersmaggioli.it/blog/le-reti-neurali-ricorrenti (cit. on pp. 17, 19).

[17] Marco Raiola, Stefano Discetti, and Andrea Ianiro. «On PIV random error minimization with optimal POD-based low-order reconstruction». In: *Experiments in fluids* 56.4 (2015), pp. 1–15 (cit. on p. 23).

[18] Renato Elias, Jose Camata, Albino Aveleda, and Alvaro Coutinho. «Evaluation of parallel and communication models in the finite element solution of coupled viscous flow and transport problems». In: (Jan. 2009) (cit. on p. 37).

[19] Amir Farzad, Hoda Mashayekhi, and Hamid Hassanpour. «A comparative performance analysis of different activation functions in LSTM networks for classification». In: *Neural Computing and Applications* 31.7 (2019), pp. 2507–2521 (cit. on p. 78).

[20] Akshay Tondak. *Introduction to Recurrent Neural Networks (RNN)*. URL: https://k21academy.com/datascience/machine-learning/recurrent-neural-networks/ (cit. on p. 78).

[21] Zhiwen Deng, Yujia Chen, Yingzheng Liu, and Kyung Chun Kim. «Time-resolved turbulent velocity field reconstruction using a long short-term memory (LSTM)-based artificial intelligence framework». In: *Physics of Fluids* 31.7 (2019), p. 075108 (cit. on p. 79).