Previsione di moti caotici e implementazione dell'informazione fisica in un'Echo State Network

Claudio Chilin
Emanuele Ciccarelli
Michele Viscione

Febbraio 2022

Sommario

L'obiettivo finale del documento è ottenere una previsione efficace dell'evoluzione di un moto caotico attraverso l'uso di una Echo State Network. Nella prima parte vi è un'introduzione a questo tipo di rete neurale, dopodiché vengono trattati i possibili aggiustamenti utili a migliorare l'orizzonte di previsione, sfruttando anche approcci nuovi come l'introduzione dell'informazione fisica nella *loss function* e soluzioni originali come la funzione di attivazione *CMT2*.

Indice

1 Introduzione

Al giorno d'oggi, l'uso delle reti neurali *feed-forward* ha consentito di risolvere problemi di ampissimo respiro. Tuttavia l'implementazione per la risoluzione di problemi con dipendenza temporale risulta poco naturale e a volte addirittura svantaggiosa. Sta diventando invece sempre più popolare l'uso delle reti neurali ricorrenti (RNN), cioè reti in cui l'output non è semplicemente una funzione dell'input, ma nelle quali entrano in gioco anche gli input e gli stati della rete passati. Fin dal modello proposto da Hopfield [1], la presenza di punti fissi è stata fondamentale per il corretto funzionamento dell'architettura. Le RNN sono allora strumenti molto potenti per risolvere problemi ad alta complessità come sistemi dinamici caotici.

In particolare, si parla di *Reservoir Computing* riferendosi a metodi di machine learning che sfruttano sistemi ad elevata complessità, nel nostro caso una riserva di neuroni, per codificarvi l'informazione nella dinamica transiente forzata dall'input, praticamente trasformando un'informazione spazio-temporale in una completamente spaziale attraverso lo stato della riserva.

Un grosso problema di questo tipo di reti sta nel costo computazionale dell'aggiornamento di strutture particolarmente intricate. La soluzione a questo problema è stata proposta indipendentemente da Jeager tramite la Echo State Network [2] e Maass con la Liquid State Machine [3] proponendo uno schema in cui la riserva è generata casualmente e lasciata invariata per tutto il tempo, modificando unicamente i pesi che vanno dalla riserva all'input.

Ciò ha il doppio vantaggio di essere non solo molto più veloce, ma anche più versatile. Nel lavoro di Jaeger a cui facciamo riferimento, infatti, viene mostrato come la stessa struttura può essere utilizzata per risolvere molti problemi di diversa natura, dall'imparare un segnale periodico (ovvero un attrattore a molti punti) a comportarsi come un flip-flop (cioè imparare molteplici attrattori).

Una caratteristica importante per il funzionamento delle RNN è che abbiano una memoria a breve termine. Ciò è necessario affinché l'apprendimento non avvenga su un passato troppo lontano, creando una dipendenza troppo forte da una storia troppo lunga. Questa caratteristica si traduce nelle nostre ESN nella cosiddetta *echo state property*, ovvero la proprietà di dimenticare i propri stati nel lontano passato e dipendere unicamente da una (relativamente) breve sequenza di input precedente al corrente: forzando la riserva tramite un segnale, questo "echeggerà" all'interno di essa per un certo periodo di tempo. Si potrebbe vedere l'addestramento che avviene sui pesi tra la riserva e l'output come l'apprendimento di quali informazioni passate andare a richiamare per costruire la previsione. La echo state property è regolata dal raggio spettrale della matrice di adiacenza della riserva, λ_{max}.

Un ulteriore interessante vantaggio del basarsi su strutture casuali e dover allenare unicamente i pesi in output, sui quali c'è una dipendenza lineare, è che è possibile costruire il modello su substrati non convenzionali, come chip fotonici [4] o strutture atomiche [5].

In questo documento viene definita in maniera dettagliata una Echo State Network, mostrandone le capacità. Dopodiché il nostro studio si concentra sull'utilizzo di questa rete per effettuare previsioni sul moto nel modello di Lorentz, studiandone l'efficacia rispetto a degli iperparametri, nonché migliorandone la capacità predittiva attraverso l'inserimento dell'informazione fisica nel modello. È inoltre descritto un metodo di effettuare l'aggiornamento dello stato della riserva, da noi denominato CMT2, che consente di aggiornare dinamicamente il raggio spettrale della matrice di adiacenza che la descrive.

2 Echo State Network

In questa sezione viene illustrato il modello basilare di ESN, andandone poi a toccare "con mano" le capacità attraverso una serie di piccoli esperimenti mirati proprio ad esplicitarne il comportamento. Infine, viene introdotto il concetto di *informazione fisica*, che sarà utilizzato nella sezione 3 e che vedremo migliorare la capacità predittiva del modello.

2.1 ESN "classica"

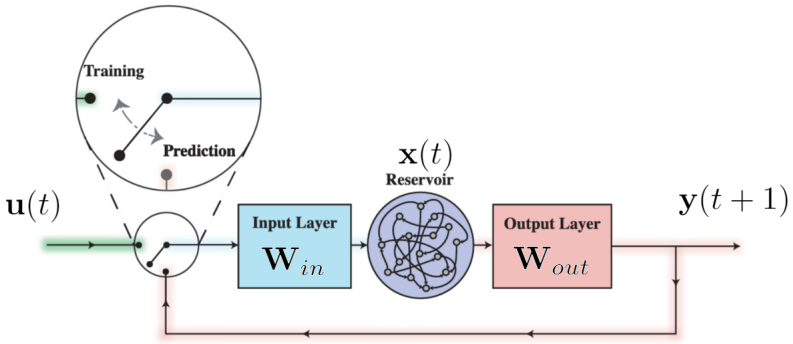


Figura 1: Rappresentazione schematica del modello.

Una ESN è un tipo di rete ricorrente: il suo stato non dipende unicamente dall'attuale input, come per una classica rete feedforward, ma tiene anche conto dei propri stati passati, comportandosi così come un sistema dinamico. Proprio per ciò, questa rete è in grado di predire in maniera piuttosto efficace l'andamento futuro del segnale dato in ingresso, arrivando a riuscire a fare delle previsioni accurate anche per un moto caotico.

Nel modello più basilare di ESN (per una rappresentazione schematica si veda la figura 1) un segnale $\mathbf{u}(t)$ (vettore a valori reali di dimensione k) viene codificato attraverso una matrice di pesi $\mathbf{W}_{in}$ in una *riserva* il cui stato al tempo t è definito come $\mathbf{x}(t) \in [-1,1]^N$. Ad ogni passo temporale $\mathbf{x}(t)$ viene aggiornato attraverso l'equazione

$$x_i(t+1) = f((\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}\mathbf{x}(t))_i) \tag{1}$$

in cui la matrice $\mathbf{W}$ è una matrice $N \times N$, $N \gg k$, a valori reali e la funzione f è una funzione di attivazione non-lineare. Nella più classica delle formulazione delle ESN $f(x) = \tanh x$; ma, come mostrato nella sezione 4, è possibile utilizzare altre funzioni, andando così anche ad influenzare la velocità di apprendimento della rete. Laddove non specificato diversamente sarà sottinteso l'uso di $\tanh x$ come funzione di attivazione.

Compiuto l'aggiornamento dello stato della riserva tramite l'equazione 1, l'output $\mathbf{y}(t+1)$ (genericamente di dimensione k', ma nel nostro lavoro e nella stragrande maggioranza dei casi $k' = k$) si ottiene invece tramite $\mathbf{y}(t+1) = \mathbf{W}_{out}\mathbf{x}(t+1)$.

In questo modo, lo stato della riserva $\mathbf{x}(t)$ sarà funzione, oltre che dell'attuale stato dello strato di input $\mathbf{u}(t)$, anche dei propri stati passati $\{\mathbf{x}(t-1),\ \mathbf{x}(t-2),\ ...\}$, e di conseguenza degli input passati $\{\mathbf{u}(t-1),\ \mathbf{u}(t-2),\ ...\}$.

Perché la rete funzioni è necessario che abbia *stati di eco*, cioè che esista un tempo in cui lo stato della riserva sia indipendente dalla condizione iniziale $\mathbf{x}(0)$, ma dipenda solo dalla sequenza di input. In realtà è possibile dimostrare [2] che la presenza di stati di eco implica *memoria evanescente*, cioè la dipendenza dello stato della riserva non risiede in tutti gli stati di input, bensì solo ad una sequenza limitata $\{\mathbf{u}(t), ..., \mathbf{u}(t-\tau)\}$, $0 < \tau < t$. La presenza di stati di eco nel sistema va sotto il nome di *echo state property*.

Le matrici $\mathbf{W}_{in}$, $\mathbf{W}$, $\mathbf{W}_{out}$ e lo stato iniziale della riserva $\mathbf{x}(0)$ sono inizializzati con valori casuali, ma avendo delle accortezze [2]: le matrici di accoppiamento sono sparse, cioè il sistema è debolmente connesso nelle sue parti. Nella fattispecie, seguendo l'approccio in [6], le righe di $\mathbf{W}_{in}$ hanno un solo elemento non nullo di valore estratto a caso in maniera uniforme nell'intervallo $[-\sigma_{in}, \sigma_{in}]$, mentre $\mathbf{W}$ ha un grado medio (cioè il numero medio di entrate non nulle per riga) $\langle d \rangle = 3$, e viene riscalata di un fattore che sarà in

seguito motivato e quantificato perché la *echo state property* sia soddisfatta. La sparsità di queste matrici è un requisito affinché il sistema non sia eccessivamente forzato dall'attuale stato dell'input e mantenga quindi una migliore traccia degli stati passati nella sua dinamica. È possibile inizializzare $\mathbf{W}_{out}$ in maniera ottimale cosicché l'addestramento converga più velocemente sfruttando la ridge regression: $\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \gamma\mathbb{I})$. Dove $\mathbf{X}$, $\mathbf{Y}$ sono rispettivamente le matrici ottenute concatenando input e corrispondente output nei vari passi temporali di una fase detta di pre-allenamento, cioè $(\mathbf{Y})_{i,j} = y_i(t = j)$. γ è il cosiddetto fattore di regolarizzazione di Tikhonov, posto pari a 0.0001 in tutto il lavoro.

In fase di addestramento, che avviene in maniera supervisionata, sarà la sola $\mathbf{W}_{out}$ ad essere modificata, tramite una semplice minimizzazione dell'errore rispetto al segnale "maestro" $\hat{\mathbf{y}}(t)$. Data la proprietà di memoria evanescente della rete, la parte iniziale dei dati viene scartata. In questo modo l'addestramento non dipenderà dalla condizione iniziale casuale (*washout*).

Terminata la fase di addestramento, l'output del sistema viene collegato con l'input, tale che $\mathbf{u}(t) = \mathbf{y}(t)$. In questo modo la rete produce previsioni sul sistema che ha appreso. Un approccio piuttosto comune sta nel dare un input dal segnale "maestro" utilizzato in fase di allenamento dopo una serie di previsioni (in numero fissato o variabile), per migliorare l'orizzonte di previsione della rete.

Echo State Property

Per andare a definire in maniera precisa la echo state property sono necessarie delle considerazioni preliminari: Innanzitutto, è necessario che il dominio dell'input sia compatto: $\mathbf{u}(t) \in U \ \forall\ t$, U chiuso e limitato. Lo stato della riserva nel corso dell'evoluzione temporale evolve come un sistema caotico non lineare, ed in quanto tale rimane anche esso confinato in un insieme compatto A. Chiamo queste condizioni **condizioni standard di compattezza**.

Inoltre, per mantenerci il più possibile generici, supponiamo che l'evoluzione del sistema sia definita da un operatore $\mathbf{T}(\mathbf{u}, \mathbf{x}) : \ U \times A \to A$ continuo negli argomenti tramite $\mathbf{x}(t+1) = \mathbf{T}(\mathbf{u}(t+1), \mathbf{x}(t))$

Definizione 1 *Un sistema ha **stati di eco** se lo stato della riserva $\boldsymbol{x}(t)$ per $t \to \infty$ è univocamente determinato dalla sequenza degli input $\boldsymbol{u}^t \equiv \{\boldsymbol{u}(t-1), ...\boldsymbol{u}(0)\}$.*
Cioè $\forall \boldsymbol{u}^t \in U^t$ sequenza di input, $\forall \boldsymbol{x}^t,\ \boldsymbol{x'}^t \in A$ stati della riserva, definiti tramite $\boldsymbol{x}(i) = \boldsymbol{T}(\boldsymbol{u}(i), \boldsymbol{x}(i-1))$, $\boldsymbol{x'}(i) = \boldsymbol{T}(\boldsymbol{u}(i), \boldsymbol{x'}(i-1))$, il sistema si dice soddisfare la echo state property se $|\boldsymbol{x}(t) - \boldsymbol{x'}(t)| \xrightarrow{t\to\infty} 0$

In tal caso, posso definire le **echo input function** $\mathbf{E} = \{e_1(\mathbf{u}^t), ..., e_N(\mathbf{u}^t)\}$ tali che, per $t \to \infty$, $x_i(t) = e_i(t)$.

È possibile dimostrare [2] che un sistema in cui esistono stati di eco ha una cosiddetta *memoria evanescente*, ovvero la traccia degli input passati svanisce col tempo. Matematicamente ciò si formula dicendo che per ogni $\epsilon > 0$, per ogni $\mathbf{u}^t, \mathbf{u'}^t$ tali che $|\mathbf{E}(\mathbf{u}^t) - \mathbf{E}(\mathbf{u'}^t)| < \epsilon$, esistono $\delta > 0, h > 0$ per cui $|\mathbf{u}(t') - \mathbf{u'}(t')| < \delta \ \forall\ t' \in [t-h, t]$. Questo ci dice che l'informazione trasmessa al serbatoio tramite l'input sarà rilevante solo per un tempo finito.

Condizione sufficiente perché esistano stati di eco è che il massimo autovalore λ_{max} della matrice $\mathbf{W}$ sia minore di 1 in modulo. Assumiamo infatti che la funzione di attivazione $f(x) < x \ \forall\ x$, allora:

$$
\begin{aligned}
|\mathbf{x}(t) - \mathbf{x'}(t)| &= |\mathbf{T}(\mathbf{u}(t), \mathbf{x}(t-1)) - \mathbf{T}(\mathbf{u}(t), \mathbf{x'}(t-1))| \\
&\leq |(\mathbf{W}\mathbf{x}(t-1) + (\mathbf{W}_{in}\mathbf{u}(t)) - (\mathbf{W}\mathbf{x'}(t-1) + (\mathbf{W}_{in}\mathbf{u}(t))| \\
&= |\mathbf{W}(\mathbf{x}(t-1) - \mathbf{x'}(t-1))| \leq |\lambda_{max}|\ (\mathbf{x}(t-1) - \mathbf{x'}(t-1))
\end{aligned}
$$

Grazie a questa condizione, ricaviamo che perché il nostro modello funzioni è necessario riscalare la matrice $\mathbf{W}$ di un fattore $\alpha < 1/|\lambda_{max}|$.

2.2 Sistemi dinamici

L'obiettivo che ci poniamo con questo documento è l'utilizzo di una ESN per riuscire a prevedere l'evoluzione di un sistema dinamico caotico.

Un sistema dinamico è una mappa $\psi^t(\mathbf{x}) : \ \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}^l$, $\mathbf{x} \in \mathbb{R}^l$ definita da una equazione differenziale ordinaria come segue:

$$
\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}(t)) \qquad \text{caso di tempo continuo}
$$

$$\mathbf{x}(t+1) = \mathbf{F}(\mathbf{x}(t)) \qquad \text{caso di tempo discreto}$$

F è detto il campo vettoriale associato alla mappa. La caoticità del moto sta nella crescita esponenziale delle distanza tra due soluzioni, caratterizzata dagli esponenti di Lyapunov del sistema.

È immediato notare che l'equazione 1 può essere vista come descrittiva di un sistema dinamico. Supponiamo che la rete sia stata addestrata, e che quindi sia nella configurazione chiusa in cui $\mathbf{u}(t) = \mathbf{y}(t)$. Lo stato complessivo della rete è quindi completamente descritto da $(\mathbf{x}(t), \mathbf{y}(t))$, che chiamiamo rispettivamente variabili nascoste e visibili del sistema. La rete evolve allora in maniera deterministica. Un sistema dinamico di questo tipo, avente delle unità nascoste, si dice un **sistema dinamico neurale**. È possibile dimostrare [7] che sistemi di questo tipo possono approssimare con arbitraria precisione un sistema dinamico a dimensionalità inferiore tramite le variabili visibili.

Ovviamente ciò richiede delle premure, ad esempio è importante notare che affinché l'approssimazione sia valida non è sufficiente imporre delle particolari condizioni iniziali al solo strato visibile, bensì è necessario che ad esse corrispondano delle specifiche condizioni iniziali anche alle variabili della riserva. La echo state property ci viene incontro in questa richiesta, permettendo di "dimenticare" la condizione iniziale arbitraria della riserva (che sarà infatti scartata) e creando invece una corrispondenza 1-a-1 tra la sequenza di input ed $\mathbf{x}(t)$.

Quando la rete viene inizializzata, lo stato della riserva è completamente casuale. È necessario forzare il sistema mandandogli una porzione di traiettoria in input. La riserva si stabilizzerà così su un attrattore che corrisponde ad un qualche attrattore del sistema dinamico. Possiamo dire che, in un certo senso, addestrando $\mathbf{W}_{out}$, stiamo cercando l'operatore che proietti lo stato della riserva in maniera tale che approssimi il sistema dinamico voluto.

La mappatura dinamica non riflette unicamente il comportamento di un insieme di orbite del sistema, bensì determina una legge di come i punti dello spazio delle configurazioni del sistema evolvono nel tempo. In questo modo, con l'addestramento la rete non si limita ad imparare un insieme di orbite, bensì impara in maniera efficace ad approssimare il sistema dinamico a cui siamo interessati.

2.3 Esperimenti

Di seguito sono riportati dei semplici esperimenti, scelti per mostrare in maniera chiara la echo state property e la memoria evanescente. Inizialmente prendiamo in esame come la rete risponde all'input di un segnale sinusoidale, vediamo quanto rapidamente la condizione iniziale viene dimenticata in base al raggio spettrale e come una rete allenata chiusa in loop converga ad un punto fisso. Successivamente, cerchiamo di mostrare la capacità della rete di passare da un'orbita asintotica all'altra in base a improvvisi cambiamenti nello strato di input.

2.3.1 Segnale sinusoidale

In questo esperimento il nostro scopo è mostrare come, dopo un iniziale transiente in cui viene cancellata l'informazione della condizione iniziale, la riserva $\mathbf{x}(t)$ si allinei su di una precisa orbita che corrisponde alla mappatura del segnale in input.

Fissate le matrici di adiacenza $\mathbf{W}$, $\mathbf{W}_{in}$, abbiamo mandato m volte lo stesso segnale $\mathbf{u}(t)$ su condizioni iniziali della riserva $\mathbf{x}^{(j)}(0), j = 0, ..., m-1$ diverse, trascurando l'output (e quindi l'allenamento). Per quantificare la distanza tra due stati di riserva, abbiamo scelto uno stato di riferimento $\mathbf{x}^{(0)}(t)$ e definito le funzioni $\delta x_j(t) \equiv \sum_{i=1}^{m-1} |x_i^{(j)} - x_i^{(0)}|^2(t)$, dove con $x_i^{(j)}(t)$ abbiamo indicato il valore del i-esimo neurone della riserva al tempo t della j-esima ripetizione dell'esperimento. In figura 2 sono riportati i risultati ottenuti.

Come previsto, qualsiasi fosse la distanza iniziale tra i due stati, le funzioni $\delta x_i(t)$ vanno a zero molto rapidamente in un tempo finito. Chiamiamo il valor medio di questo tempo sui nostri esperimenti t_{clean}. Abbiamo ritenuto interessante studiare la dipendenza di t_{clean} dal parametro λ_{max}, cioè il massimo autovalore della matrice di adiacenza W. Abbiamo svolto quindi uno studio su una stessa matrice W, riscalandola ciascuna volta per avere come massimo autovalore l'ascissa riportata in figura e mediando t_{clean} su 100 condizioni iniziali diverse (ma forzando il sistema sempre con lo stesso input $u(t)$). Tale andamento è risultato indipendente dalla taglia della riserva N. La curva ricavata è riportata in figura 3. È evidente l'andamento lineare fino ad un certo valore di $\lambda_{max} > 1$, dopo il quale il t_{clean} diverge molto rapidamente.

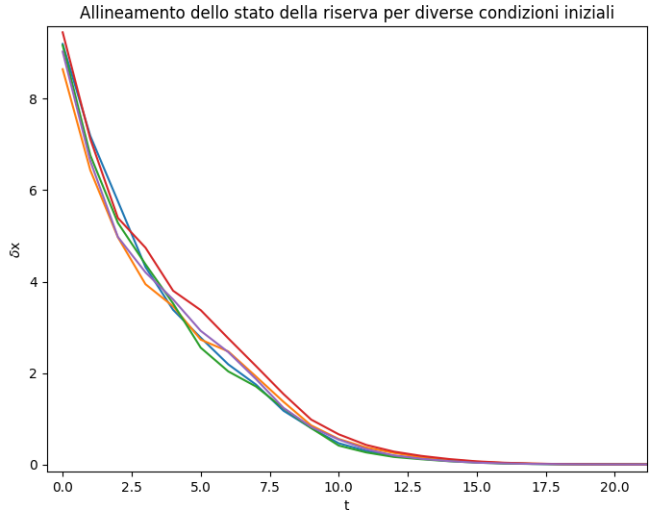


Figura 2: Andamento di $\delta x_j(t)$, definite nel testo, per 5 differenti condizioni iniziali e stessa matrice di adiacenza W con $\lambda_{max} = 0.9$

Il valore per cui avviene la divergenza è risultato differente per diverse matrici W, ma il comportamento di $t_{clean}(\lambda)$ rimane lo stesso. Un fit lineare sulla parte regolare ha fornito un coefficiente pari a $m = 21.3 \pm 0.1$

A questo punto ci siamo interessati a vedere come lo stato della riserva si discosta dall'orbita forzata dall'input una volta che la rete è stata allenata ed input ed output sono chiusi in loop. In figura 4 è riportato l'andamento temporale dell'output (immagine in alto) e di tre neuroni della riserva per un sistema forzato (ed allenato, seppur non perfettamente, come evidente) fino a $T/2$ e chiuso in feedback tra input ed output per il resto del tempo. Si noti come l'andamento dei neuroni abbia lo stesso periodo dell'input in fase forzata e come di come seguano fedelmente l'output in fase chiusa a loop, convergendo ad un punto fisso. La convergenza ad un punto fisso sembra essere un comportamento comune a qualsiasi ESN chiusa in questa configurazione. Sebbene non sia particolarmente evidente in questo specifico esperimento, i neuroni hanno comportamenti periodici differenti. Per un segnale diverso sarebbe stato possibile anche osservare la presenza di armonici differenti in ciascun neurone. Questo comportamento è influenzato dalle connessioni che ciascun neurone ha e di conseguenza anche dal raggio spettrale λ_{max}.

Vale la pena inoltre menzionare che per una buona riuscita di questo esperimento è stato necessario selezionare un periodo tale che l'orizzonte di previsione non ne fosse un multiplo. In questo modo si evita di allenare la macchina sempre sullo stesso dataset, cosa che diminuirebbe l'informazione da essa acquisita.

2.3.2 Flip flop

Abbiamo mostrato come forzare la rete tramite l'input porta lo stato della riserva in un'orbita periodica e che questa può essere tradotta nell'output desiderato addestrando $\mathbf{W}_{out}$. Con il presente esempio, ripreso da [2], è possibile mostrare invece la capacità della rete di imparare più attrattori e di come il passaggio da un'orbita attrattiva ad un'altra sia imponibile tramite l'input.

Il setup è il seguente: all'ingresso della riserva sono presenti m input. Ad ogni passo temporale uno degli input può mandare una "spike" nel sistema, cioè uno solo di essi sarà non nullo con probabilità $p = 0.02$. L'output corrispondente a ciascun input si porrà allora pari ad 0.5 dopo l'arrivo di una spike nel sistema e tornerà a -0.5 qualora una nuova spike arrivi da un differente input. Per fare sì che lo stato dell'output influenzi quello della riserva, l'equazione 1 è stata modificata inserendo all'interno di $f(x)$ anche un termine $\mathbf{W}_{back}\mathbf{y}(t)$, dove $\mathbf{W}_{back}$ ha un quinto delle entrate non nulle, con valori ± 0.1. Nella matrice W_{in} le entrate non nulle sno impostate a ± 5, in modo che l'input abbia un peso importante sulla dinamica. Il raggio spettrale è $\lambda = 0.4$ per avere una perdita di memoria sufficientemente veloce.

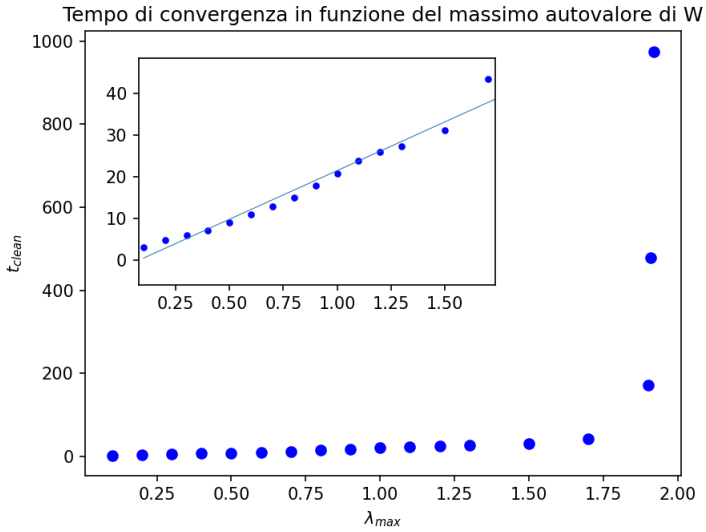


Figura 3: Dipendenza della memoria da λ_{max}. L'immagine interna ha lo scopo di evidenziare la linearità dei parametri al di sotto del valore di divergenza.

Purtroppo non siamo riusciti a riprodurre in maniera efficace l'esperimento, ma abbiamo ritenuto comunque interessante mostrare e commentare il risultato dell'articolo. In figura 5 sono riportati gli andamenti di alcuni input, output e neuroni della riserva scelti a caso a scopo esemplificativo.

Abbiamo visto che in assenza di input il sistema tende a convergere verso un punto fisso stabile: la riserva sarà in uno stato stazionario. Quando la rete riceve una spike in input, il grande peso dato ad essa tramite $\mathbf{W}_{in}$ farà sì che per l'aggiornamento della riserva lo stato precedente sia trascurabile. La riserva verrà mandata fuori dall'equilibrio ed attraverserà un regime transiente che porterà il sistema nel punto fisso corrispondente all'input ricevuto, visualizzato esternamente tramite l'input. La presenza del feedback $\mathbf{W}_{back}$ assicura che il sistema si congeli nel giusto stato. L'allenamento ha in questo caso non solo lo scopo di fornire la giusta mappatura dei punti fissi della riserva nell'output ma anche di supportare la convergenza verso di essi.

2.4 Informazione fisica

Si è visto come questo tipo di rete neurale risulti naturalmente calzante per la previsione delle traiettorie di sistemi dinamici anche caotici. La conoscenza di equazioni che governano l'evoluzione del sistema (vedi le ODE descriventi un sistema dinamico, così come leggi di conservazione associate al moto, eccetera) possono supportare l'addestramento. Sta allora diventando piuttosto popolare l'introduzione della conoscenza del modello nella rete, ottenendo così un approccio ibrido (si veda ad esempio [8]). Ciò è possibile anche nel caso di una ESN, ma abbiamo ritenuto più interessante un approccio differente, proposto in [6], che si mantiene *model free* inserendo l'informazione fisica nella funzione loss da minimizzare.

Con questo metodo, congiuntamente alla minimizzazione dell'errore quadratico medio E_d riportata nella sezione 2.1, si intende richiedere anche che il sistema rispetti una particolare legge fisica. In particolare, detta $\frac{d\mathbf{y}(t)}{dt} = \mathcal{F}(\mathbf{y})$ l'equazione differenziale che regge il sistema dinamico che ci interessa studiare, il vincolo che vogliamo imporre è

$$W(\mathbf{y}) = \mathcal{F}(\hat{\mathbf{y}}) - \dot{\mathbf{y}} = 0 \tag{2}$$

Dove $\dot{y}$ sarà la derivata temporale della traiettoria del sistema predetto, cioè il rapporto incrementale tra due punti successivi della previsione del sistema, mentre $\hat{y}$ è il punto medio tra i due, in cui sarà calcolata la derivata della traiettoria tramite $\mathcal{F}$.

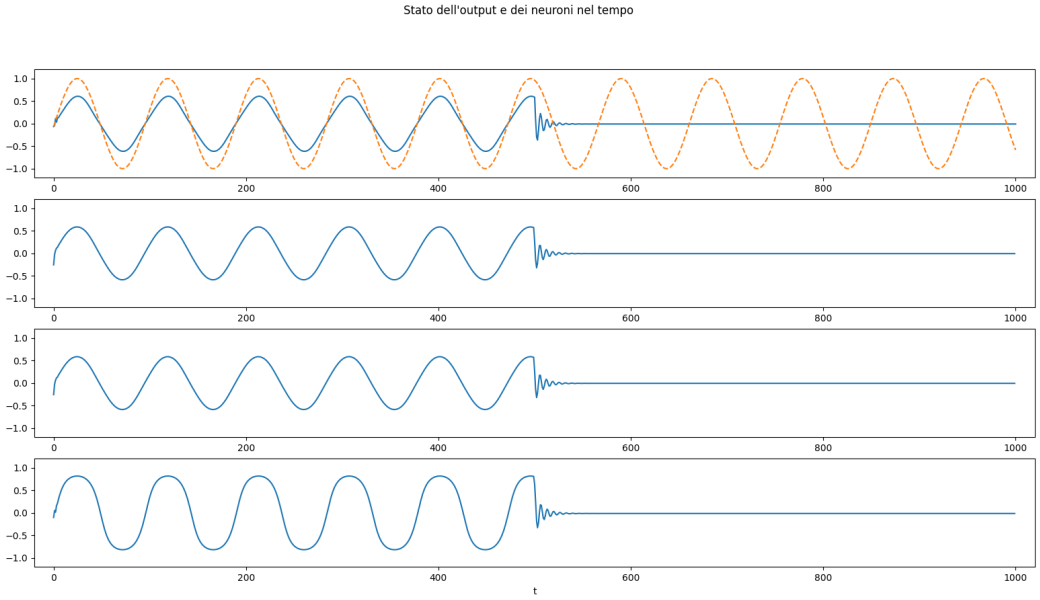


Figura 4: Andamento temporale di segnale maestro e output (rispettivamente linea tratteggiata e continua in alto) e di tre neuroni della riserva scelti casualmente, per una rete allenata fino a $T/2 = 500$ e chiusa in loop tra input ed output fino alla fine della simulazione

L'errore da minimizzare sarà così $E_{tot} = E_d + \alpha E_p$, l'errore fisico sarà dato da

$$E_p = \frac{1}{N_t} \sum_f \frac{|W(\mathbf{y}_f)|^2}{k} \tag{3}$$

dove la somma si estenderà su tutti gli N_t punti $\mathbf{y}_f$ previsti dalla rete in fase di training, gli stessi utilizzati per calcolare l'errore rispetto al segnale "maestro". Il fattore α, empiricamente posto uguale a 30, è stato necessario affinché l'inclusione della fisica fosse rilevante. È opportuno osservare che il termine $\dot{\mathbf{y}}$ può essere computato attraverso due approcci differenti. Il primo di questi consiste nell'utilizzo di una derivazione numerica, che ripone il vincolo nella forma

$$W(\mathbf{y}(t)) = \mathcal{F}(\hat{\mathbf{y}(n)}) - \frac{\mathbf{y}(t) - \mathbf{y}(t-1)}{\epsilon} = 0 \tag{4}$$

Oppure, in una forma meno rumorosa

$$W(\mathbf{y}(t)) = \epsilon \mathcal{F}(\hat{\mathbf{y}(t)}) - \mathbf{y}(t) + \mathbf{y}(t-1) = 0 \tag{5}$$

Un tale vincolo tuttavia porterà ad una rappresentazione distorta dell'effettivo campo vettoriale che guida le curve integrali del nostro sistema. Ad esempio, nel caso del modello di Lorentz, il termine $\mathcal{F}(\hat{\mathbf{y}(t)})$, calcolato in $\mathbf{y}(t)$ anziché nel punto medio porta il sistema ad apprendere una rappresentazione errata delle orbite periodiche attorno agli attrattori, che invece convergono ad un punto fisso, come si può osservare nella figura 6 che mostra l'*n-forecasting* (dove la rete deve prevedere l'intero orizzonte di allenamento del sistema a partire dalle sole condizioni iniziali) di una ESN allenata in questo modo. Questo comportamento è comprensibile se si considera che una tale distorsione su $\mathcal{F}(\cdot)$ corrisponde ad "anticipare" i vettori del campo vettoriale lungo l'orbita periodica, rendendoli di fatto più convergenti all'attrattore rispetto al reale comportamento del modello. Questo comportamento emerge nonostante la rete sia allenata sulle orbite reali del sistema, indice del fatto che l'immissione dell'informazione fisica "esterna" ai dati ha avuto effetto.

La nostra soluzione per rendere più affidabile la computazione di $\mathcal{F}(\cdot)$ è quindi quella di calcolarlo nel punto medio fra $\mathbf{y}(t)$ e $\mathbf{y}(t-1)$.

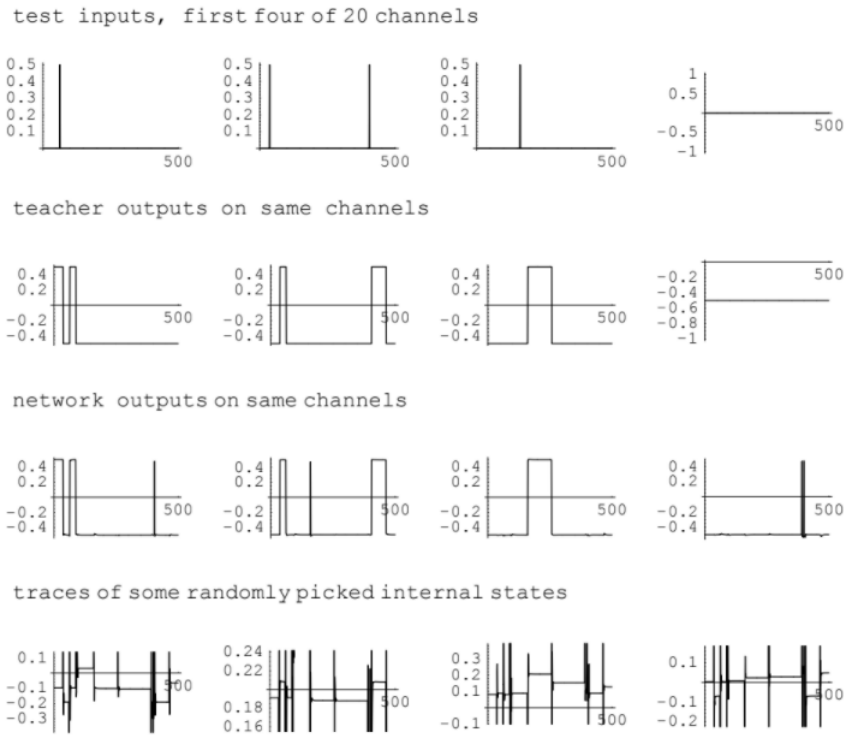


Figura 5: Input, output e risposta di alcuni neuroni nell'esperimento di [2] per molteplici attrattori.

L'altro approccio utilizzato per calcolare il vincolo 2 consiste nello sfruttare il meccanismo dell'auto-differenziazione (*backpropagation*). Tuttavia un tale schema necessita che la nostra rete abbia una dipendenza diretta dal tempo t, poiché essa è la variabile rispetto al quale vogliamo differenziare. Abbiamo provato a modificare il nostro modello rendendolo direttamente dipendente dal tempo, ma il calcolo dei gradienti soffriva di tipiche ([9]) patologie di azzeramento e rigidità. Una tale soluzione potrebbe essere esplorata meglio tramite l'utilizzo di algoritmi diversi di ottimizzazione, come L-BFGS-B usato in [6], o approcci *gradient-free*. Per completezza mostriamo nella figura 7 l'efficacia dell'informazione fisica tramite auto-differenziazione nel caso di una rete *feed-forward* per la regressione sulla dinamica di un oscillatore armonico smorzato, mostrando anche le sue notevoli capacità di reiezione del rumore presente nei dati di allenamento.

Occorre precisare che l'utilizzo della derivata numerica non implica necessariamente una peggiore comprensione dell'informazione fisica. A seconda del contesto la derivazione tramite auto-differenziazione, ignorando le maggiori difficoltà in fase di addestramento, può non essere la scelta migliore [10].

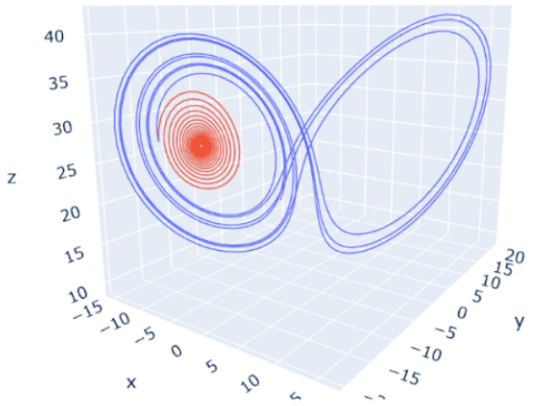


Figura 6: Segnale maestro integrato con Lorentz (in blu) e moto previsto dalla rete (in rosso) con informazione fisica distorta che porta alla convergenza nel punto fisso, teoricamente instabile

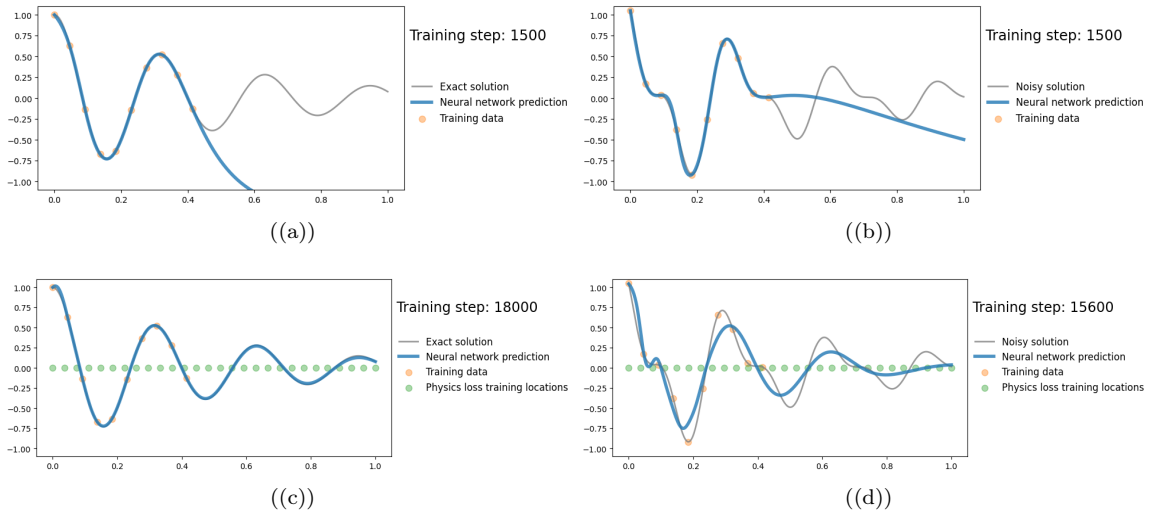


Figura 7: Regressione di una rete feed-forward. (a): Regressione senza informazione fisica e con dati non rumorosi. La dinamica trovata rimane uguale anche per epoche di addestramento molto maggiori. (b): Regressione senza informazione fisica e con dati rumorosi. La dinamica trovata fa *overfitting* sul rumore. (c): Regressione con informazione fisica e dati non rumorosi. Il comportamento del segnale viene replicato perfettamente. (d): Regressione con informazione fisica e dati rumorosi. Vi è una buona reiezione del rumore e il comportamento del segnale viene replicato efficacemente quando non si hanno più dati rumorosi come deviazione.

3 Previsione dei moti caotici

Come anticipato, lo scopo del nostro studio è di prevedere l'evoluzione di un sistema caotico approssimandolo tramite la ESN. Il sistema che abbiamo scelto è il ben noto modello di Lorentz, che ricordiamo essere retto dalle equazioni.

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = \rho x - xz - y \\ \dot{z} = xy - \beta z \end{cases} \tag{6}$$

Dove abbiamo scelto i valori tipicamente usati $\sigma = 10$, $\beta = 8/3$, $\rho = 28$ per cui il moto è caotico. Come massimo esponente di Lyapunov del sistema abbiamo considerato $\lambda = 0.934$ [6]. In questa sezione, quando saranno riportate delle scale temporali sui grafici, l'unità temporale dovrà considerarsi il centesimo del tempo di Lyapunov: $\lambda t/100$.

Il modello di Lorentz è un sistema dissipativo: la mappa associata non è simplettica, cioè non conserva il volume dello spazio delle configurazioni, che invece si riduce durante il moto. Ciò significa che qualsiasi sia la condizione iniziale la traiettoria tenderà ad andare in una zona limitata dello spazio delle configurazioni. Per i parametri che abbiamo scelto, la mappa ha due punti fissi instabili: $\mathbf{R}^*_{\pm} = (\pm 6\sqrt{2}, \pm 6\sqrt{2}, 27)$. La traiettoria quindi, per qualsiasi condizione iniziale, andrà ad orbitare intorno a questi due punti fissi, passando in maniera imprevedibile dal girare attorno ad uno in senso orario ad un altro in senso antiorario. Tale attrattore, che si denomina "attrattore strano", non è un punto fisso né un'orbita periodica: le orbite sono quasi-periodiche ed il passaggio da un punto fisso all'altro avviene in maniera estremamente irregolare al leggero variare delle condizioni iniziali [11].

Dovrebbe essere quindi chiaro il motivo di tanto interesse nel cercare di far riprodurre alla rete un sistema di tale complessità. L'addestramento è stata eseguito utilizzando come segnale maestro una traiettoria ottenuta integrando l'equazione 6 tramite l'algoritmo Runge Kutta al quarto ordine con passo temporale $\epsilon = 0.01 \cdot \lambda$. In figura 8 è mostrato un esempio di confronto tra la traiettoria integrata con l'algoritmo (in blu) e la previsione della rete (in rosso, fatta con *1-forecasting*, cioè fornendo in input il passo successivo, ottenuto con Runge Kutta, e prevedendo il passo seguente tramite la rete).

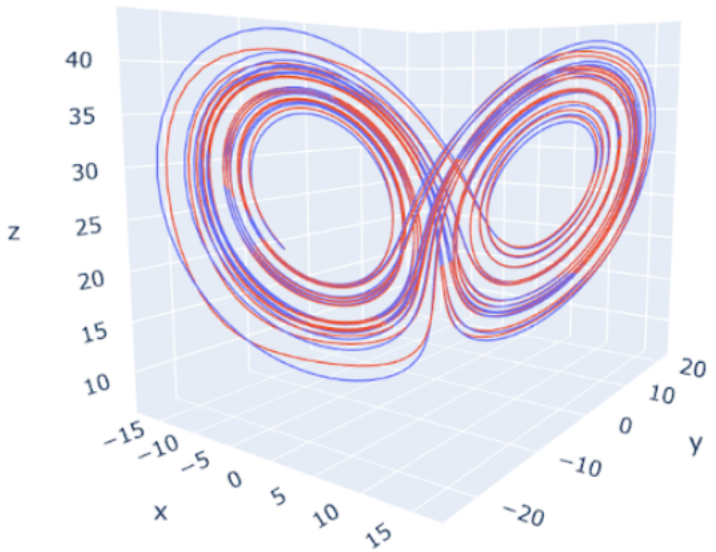
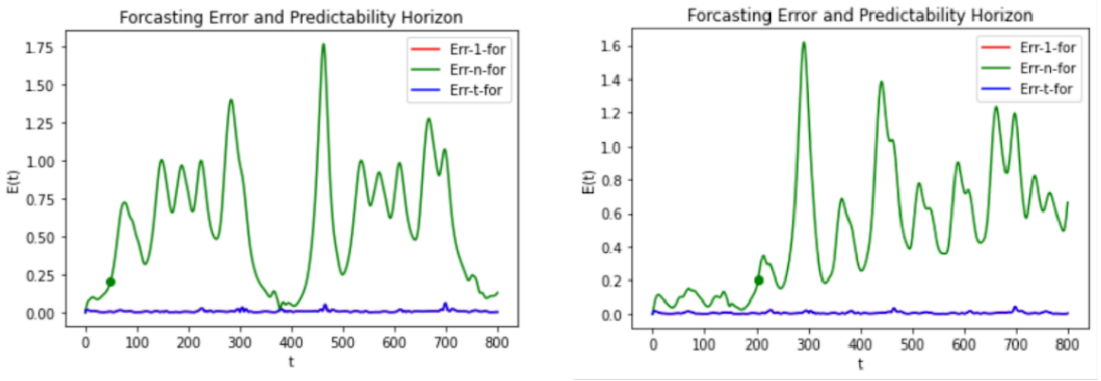


Figura 8: Previsione della traiettoria del modello di Lorentz, fino a tempo $T = 1000$, allenato per 10000 epoche per una riserva con $N = 200$ neuroni, $\sigma_{in} = 0.15$, raggio spettrale $\lambda_{max} = 0.4$

L'obiettivo finale di questo documento sarà trovare i parametri (e la migliore funzione di attivazione f dell'equazione 1) per cui la rete abbia il migliore *orizzonte di previsione* possibile. L'orizzonte di previsione



((a)) ((b))

Figura 9: Errore sulle previsioni, ESN classica (a) vs ESN con informazione fisica (b). Le reti sono addestrate sull'1-forecasting e vengono messe alla prova sia su quest'ultimo che sull'n-forecasting. In questo caso specifico il t-forecasting (forecasting di allenamento) coincide con l'1-forecasting. Il pallino verde indica il punto in cui l'errore $E(t)$ diventa pari a 0.2, cioè il nostro orizzonte di predittività.

è quantificato tramite l'errore relativo tra la previsione $\mathbf{y}(t)$ e la traiettoria integrata con Runge Kutta $\hat{\mathbf{y}}(t)$: $E(t) = \frac{|\mathbf{y}(t)-\hat{\mathbf{y}}(t)|}{\langle|\mathbf{y}|^2\rangle^{1/2}}$, dove $\langle\cdot\rangle$ indica la media temporale. In particolare, definiamo *orizzonte di previsione* il primo valore t per il quale $E(t) \geq 0.2$, in modo analogo a [6] .In figura 9 possiamo osservare gli errori sulle previsioni di due ESN, allenate con e senza informazione fisica. I rispettivi orizzonti di previsione ottenuti sono 8 tempi di Lyapunov per l'*1-forecasting* (previsione dello passo successivo del sistema dinamico) per entrambe le reti, ovvero il valore massimo ottenibile in questo esperimento. Per quanto riguarda l'orizzonte di previsione relativo all'*n-forecasting* (previsione dell'intera evoluzione del sistema a partire dalla condizione iniziale), abbiamo ottenuto 2 tempi di Lyapunov di previsione per l'ESN con informazione fisica e 0.5 tempi di Lyapunov per l'ESN classica. Risulta quindi immediato notare i benefici in termini di generalizzazione dell'informazione fisica per la previsione del comportamento di sistemi caotici.

4 Transizioni

In questa sezione vogliamo approfondire il discorso della transizione dello stato della riserva. In 2 abbiamo visto che l'approccio tipico è quello di utilizzare una tanh come funzione di attivazione, ma, più in generale, qualsiasi funzione sigmoide che non permetta ai valori dei neuroni di crescere indefinitamente può svolgere lo stesso ruolo. Partiamo dunque descrivendo l'approccio classico, dove la transizione è descritta dall'equazione

$$x_i(t+1) = \tanh((\mathbf{W}_{in})_i \mathbf{u}(t+1) + (\mathbf{W})_i \mathbf{x}(t)) \tag{7}$$

In questo schema la transizione non possiede parametri da allenare e, come abbiamo visto in 2.1, l'ESP è garantita dal raggio spettrale λ con cui inizializziamo $\mathbf{W}$. In questo caso, la dipendenza di $\mathbf{x}(t)$ dallo stato iniziale della riserva scala al più come $\lambda^t|\mathbf{x}(0)|$ e in modo analogo scala la dipendenza dagli ingressi nel tempo. Ci proponiamo ora di presentare un nuovo approccio, introducendo una transizione con parametri che possa pesare la dipendenza dagli ingressi in maniera condizionale allo stato corrente. Introduciamo quindi la Convolutionally Modulated Tanh (CMT2).

CMT2

Partiamo innanzitutto definendo la Neighboring Neural Activity (NNA) $\eta_i(t)$ dell'i-esimo neurone al tempo t come

$$\eta_i(t) = \sum_j (\mathbf{W})_{i,j} x_j(t) \tag{8}$$

Questa rappresenta la somma delle attività al tempo t dei neuroni che influenzano il neurone i-esimo, moltiplicate per i rispettivi pesi degli archi dati da $\mathbf{W}$. Chiamiamo $\tilde{\mathbf{W}}(t)$ la matrice le quali righe sono definite come $(\tilde{\mathbf{W}}(t))_i = (\mathbf{W})_i \odot \mathbf{x}(t)$ dove con $\odot$ intendiamo il prodotto di Hadamard. Sia $\eta(t)$ il vettore delle $\eta_i(t)$ e $\vec{\mathbf{1}}$ il vettore di dimensione $1 \times N$ e componenti 1, possiamo riformulare 8 come

$$\eta(t) = \tilde{\mathbf{W}}(t)\vec{\mathbf{1}} \tag{9}$$

L'equazione 9 può essere vista come l'applicazione di una Convolutional Neural Network (CNN) con filtro di dimensione $1 \times N$ e componenti fisse a 1 ad una matrice $\tilde{\mathbf{W}}(t)$, muovendo il filtro riga per riga. Risulta quindi immediata l'estensione di tale approccio con l'utilizzo una effettiva CNN che possa trovare un filtro $1 \times N$ dimensionale che pesi in maniera differente le attività dei vicini che influenzano il neurone i-esimo. Chiamiamo $\tilde{\eta}(t)$ il vettore delle NNA-pesate ottenute in questo modo e θ_C il vettore dei parametri del filtro appreso durante la fase di addestramento.

$$\tilde{\eta}_i(t) = (\tilde{\mathbf{W}}(t))_i \theta_C = \sum_j (\mathbf{W})_{i,j} x_j(t) \theta_{C,j} \tag{10}$$

Alle luce di questa introduzione definiamo la transizione CMT2 come

$$\mathbf{x}(t+1) = \alpha(t+1) \tanh((\mathbf{W}_{in})_i \mathbf{u}(t+1) + (\mathbf{W})_i \mathbf{x}(t)) \tag{11}$$

Dove $\alpha(t+1)$ è uno scalare ottenuto attraverso l'applicazione di una rete convoluzionale con filtro $1 \times N$ dimensionale su $\tilde{\mathbf{W}}(t)$ e la successiva composizione con un layer lineare, con dimensione di ingresso N, di uscita 1 e parametri allenabili θ_L, in modo da far dipendere $\alpha(t+1)$ dalle $\tilde{\eta}_i(t)$ in maniera pesata. Infine, il codominio di questa funzione viene schiacciato tra 0 e 2 attraverso l'utilizzo di una tangente iperbolica.

$$\alpha(t+1) = 1 + \tanh(\text{Linear}(\text{Conv}(\tilde{\mathbf{W}}(t), \theta_C), \theta_L)) \tag{12}$$

$$\alpha(t+1) = 1 + \tanh(\sum_i \theta_{L,i} \sum_j (\mathbf{W})_{i,j} x_j(t) \theta_{C,j}) \tag{13}$$

Possiamo inoltre introdurre un bias su entrambi i layer allenabili, in modo tale da permettere la rottura eventuali simmetrie nella modulazione di comportamenti simmetrici della riserva.

Il risultato di CMT2 è quello di modulare la classica transizione basata su Tanh in base all'attività dei neuroni della riserva. Questo poiché il nostro scopo non è quello di alterare la dinamica della transizione,

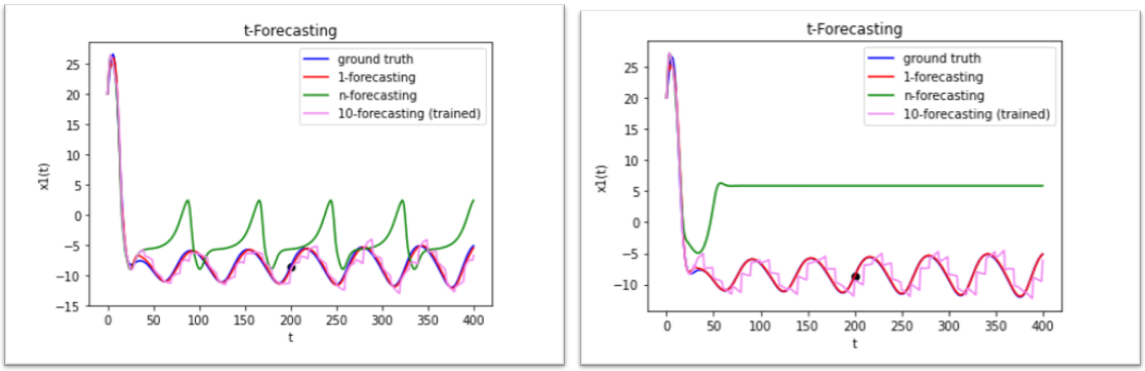
mantenendo la generalità dell'evoluzione imposta da $\mathbf{W}$, ma quello di velocizzare la convergenza a punti fissi della riserva e, soprattutto, di alterare dinamicamente il raggio spettrale. Infatti possiamo osservare che, chiamando $\bar{\mathbf{x}}(t)$ lo stato ottenuto attraverso 7, l'equazione 11 può essere riscritta come

$$\mathbf{x}(t+1) = \alpha(t+1)\bar{\mathbf{x}}(t+1) = \alpha(t+1)\tanh((\mathbf{W}_{in})_i\mathbf{u}(t+1) + (\mathbf{W})_i\alpha(t)\bar{\mathbf{x}}(t)) \tag{14}$$

In cui la matrice di transizione della riserva al tempo t è $\mathbf{W}\alpha(t-1)$, con raggio spettrale $\lambda\alpha(t-1)$. In questo modo stiamo permettendo alla riserva di ricordare (aumentando λ) o dimenticare (diminuendolo) più efficacemente gli ingressi nel tempo condizionatamente allo stato della riserva. Infatti lo stato $\mathbf{x}(t+1)$ verrà influenzato dall'ingresso $\mathbf{u}(t)$ al più come $\alpha(t)\mathbf{W}\mathbf{W}_{in}\mathbf{u}(t)$ mentre da $\mathbf{u}(t-1)$ come $\alpha(t)\alpha(t-1)\mathbf{W}^2\mathbf{W}_{in}\mathbf{u}(t-1)$. Ad esempio, la rete potrebbe imparare ad abbassare inizialmente il raggio spettrale per velocizzare la pulizia (*washout*) della dipendenza dello stato iniziale $\mathbf{x}(0)$. Inoltre, un comportamento tipico è quello di facilitare il lavoro alla $\mathbf{W}_{out}$ visto che la rimodulazione del segnale di uscita, generato osservando le dinamiche della riserva, può essere effettuata in collaborazione con CMT2. Ovviamente l'introduzione di una tale rete comporta un rallentamento del tempo di addestramento (≈ 3 volte) poiché, dato k la dimensione del sistema da emulare, passeremo da circa $k \times N$ parametri da allenare a $(k+2) \times N$.

CMT2 vs Tanh

Le proprietà finora citate permettono a CMT2 una maggiore capacità di generalizzazione rispetto a Tanh, fermo restando che il comportamento di quest'ultima può essere riprodotto imparando $\alpha(t) = 1$, $\forall t$. A supporto di questa affermazione, in figura 10 si può osservare il risultato di un confronto diretto fra queste due transizioni. In particolare l'esperimento è stato svolto sul modello di Lorentz, di cui è mostrata la prima componente, fornendo in ingresso solo due delle componenti del sistema e facendo emulare internamente alla riserva il comportamento della mancante come handicap (ciò è possibile poiché le dinamiche del modello sono accoppiate). Nel grafico vengono messe a paragone l'*n-forecasting*, l'*1-forecasting* e il *10-forecasting*, dove quest'ultimo è quello su cui le reti sono state allenate. Dal comportamento delle due reti si può osservare come la CMT2 ha introdotto una maggiore precisione ed una più efficace generalizzazione, comprendendo meglio la periodicità del modello anche quando messa completamente in *feedback* (*n-forecasting*). L'addestramento delle due reti è stato effettuato con lo stesso numero di epoche nonostante la disparità nelle dimensioni dei modelli.



((a)) ((b))

Figura 10: Confronto tra CMT2 (a) e Tanh (b). Nei grafici vengono rappresentati 1-forecasting (rosso), 10-forecasting(rosa), n-forecasting (verde) e la reale dinamica (blu). Il punto nero indica che da $t = 200$ in poi la rete si trova in un orizzonte temporale inesplorato durante il training.

Come ulteriore esempio, prendiamo in esame la risposta della nostra rete all'addestramento su un onda quadra, in maniera simile a quanto visto in 2.3.1. In figura 12 possiamo osservare l'effetto della modulazione nel rendere il segnale di uscita più vicino a quello desiderato, mentre in figura 11 viene riportato il corrispondente comportamento di $\alpha(t)$. Risulta evidente, dati questi grafici, come il segnale basso dell'onda quadra sia

rappresentato meglio con CMT2 (fig 12(a)) grazie alla riduzione improvvisa della modulazione $\alpha(t)$ in corrispondenza del passaggio alto-basso. Infatti, grazie a questo, la rete è in grado di ridurre improvvisamente il proprio raggio spettrale in modo da "dimenticare" più velocemente gli ingressi (alti) precedenti.

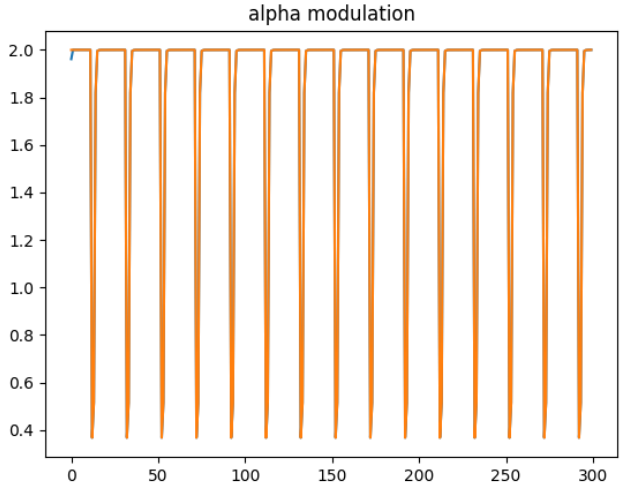
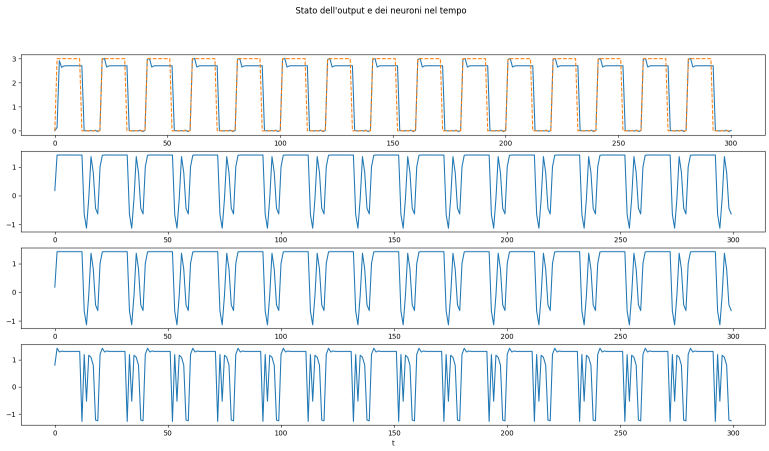
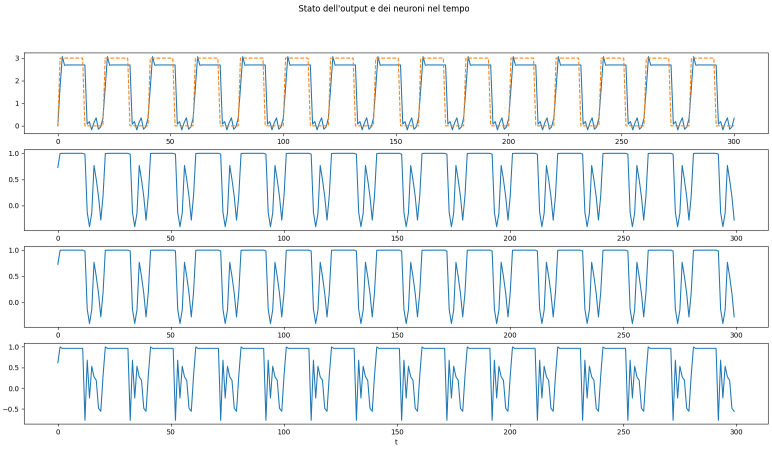


Figura 11: Modulazione $\alpha(t)$ della transizione nel tempo.

Infine, come ultima prova delle nostre affermazioni, in figura 13, confrontiamo, a parità di epoche di addestramento, la previsione della dinamica del modello di Lorentz (di cui riportiamo una componente) tramite una ESN con transizione Tanh e informazione fisica (a), la stessa della figura 9, e una ESN con transizione CMT2 e informazione fisica. Prendiamo in esame l'n-forecasting di queste due figure: Nonostante la maggiore precisione iniziale della Tanh, la rete con CMT2 riesce a generalizzare meglio la dinamica, compiendo dapprima dei giri introno al primo attrattore per poi spostarsi sul secondo (del quale dispone di minori informazioni durante l'addestramento) dove infine, erroneamente, converge. Al contrario, la rete con Tanh non riesce a generalizzare l'esistenza di un secondo attrattore e rimane quindi vincolata ad orbitare intorno al primo. La ragione per cui la rete con CMT2 converge all'attrattore è dovuta al fatto che, al tempo dell'esperimento, computavamo ancora il vincolo differenziale dell'informazione fisica in $\mathbf{y}(t)$ invece che nel punto medio. Come abbiamo già discusso in 2.4, questo comporta una distorsione del campo vettoriale appreso, con il risultato di una insolita convergenza all'attrattore. Purtroppo non vi è stato il tempo di replicare l'esperimento con le nuove accortezze. Tuttavia, è giusto precisare che CMT2 riesce a generalizzare l'esistenza dei due attrattori anche se sprovvista di informazione fisica, come si può osservare nella figura 14. Ricordiamo infine che la causa della minore precisione nella dinamica iniziale può essere dovuta alla maggiore lentezza nel trovare la giusta configurazione dei parametri di CMT2, ulteriormente aggravata dall'introduzione dell'informazione fisica, o al suo tentativo di generalizzare il secondo attrattore (è più facile imparare un'orbita fissa come per Tanh).

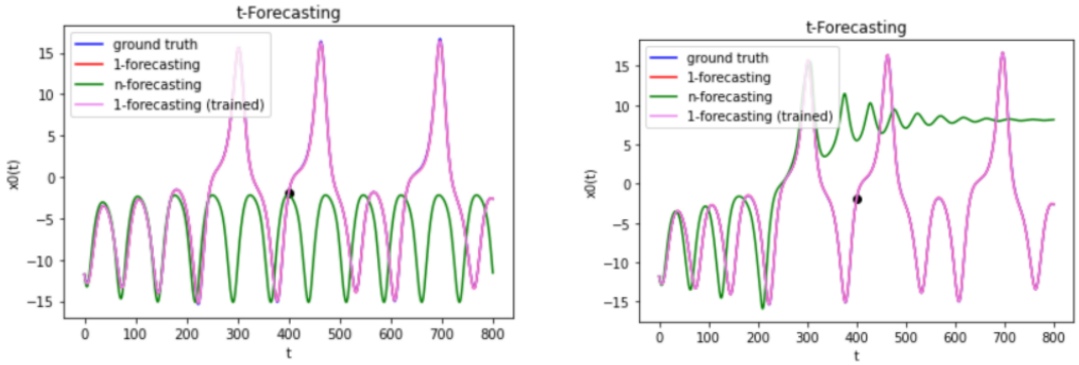


((a))



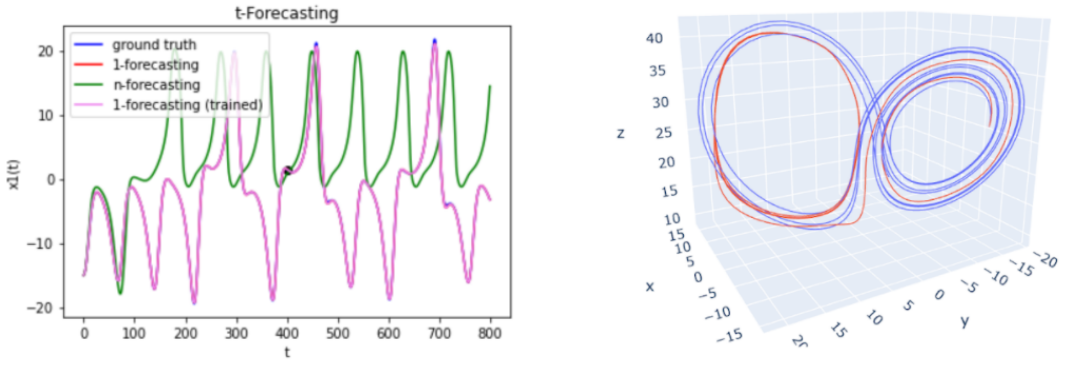
((b))

Figura 12: In maniera analoga alla figura 4, vengono riportate le dinamiche di uscita (blu) contro il segnale desiderato (arancione) e la dinamica di tre neuroni casuali (sotto-grafici 2,3,4). In particolare, la figura (a) fa riferimento all'ESN con transizione CMT2 mentre la figura (b) fa riferimento all'ESN con transizione classica.



((a)) ((b))

Figura 13: 1-forecasting (rosa) e n-forecasting (verde) del modello di Lorentz per una ESN con Tanh (a) e CMT2 (b). Entrambe le reti sono allenate con l'aggiunta dell'informazione fisica. Il punto (nero) indica la fine dell'orizzonte temporale esplorato durante l'allenamento, dopo di esso la rete replica una dinamica mai vista.



((a)) ((b))

Figura 14: N-forecasting di una ESN con CMT2 ma senza informazione fisica

5 Ruolo dei parametri

5.1 Caratteristiche generali

Riportiamo qui alcune caratteristiche generali tipiche di tutti gli esperimenti svolti, per poi addentrarci invece nella descrizione di parametri più specifici, ove si sono testati diversi valori studiandone gli effetti sull'apprendimeto della rete. In generale, come tipico nella definizione di una loss function per una rete neurale, si è usato, per tutti gli esperimenti, un regolarizzatore di tikhonov con valore di $\gamma = 0.0001$. Tale valore è stato preso in linea con il valore usato in [6].

Altre caratteristiche tipiche a tutti gli esperimenti sono il test della rete su orizzonti temporali tipicamente più lunghi rispetto a quelli usati nel training, per verificare se la rete ha imparato a generalizzare bene il comportamento del sistema, e l'utilizzo dell'algoritmo di ottimizzazione ADAM nel training della rete.

Infine, ulteriore caratteristica comune a tutti i training della rete è l'utilizzo della cpu invece che della gpu. Anche se tipicamente nel contesto delle reti neurali è frequente il contrario, nel nostro contesto i parametri che possono essere allenati sono in generale pochi e il vero problema è costituito dalle continue approssimazioni svolte per calcolare, tramite Runge Kutta, l'evoluzione del sistema che vogliamo che la rete apprenda. Dai vari esperimenti svolti è di fatto risultato un drastico aumento del tempo di allenamento della rete all'aumentare dell'orizzonte temporale considerato, evidenziando dunque il collo di bottiglia dell'addestramento.

Ciò e ricollegabile anche ad un altro fattore tipico di tutte le esecuzioni della rete, cioè l'assenza di *early stop*. Tale tecnica è di fatti una pratica molto comune, usata per evitare che la reti faccia *overfitting*, tuttavia i tempi di allenamento sono in generale fin troppo lunghi, sopratutto al crescere dell'orizzonte temporale, rendendo la possibilità di overfitting molto remota. Si è dunque optato per non utilizzare l'early stop durante i vari esperimenti eseguiti.

Cominciamo quindi a descrivere i vari iperparametri presenti nella rete e come questa si comporti al variare di quest'ultimi.

5.2 Forecasting

Un primo parametro da stabilire nella rete è il *forecasting*, cioè su quanti passi temporali la rete si allenerà a predire l'output del sistema evolvendo autonomamente, chiudendo a feedback output e input. Con k-forecasting, $1 \leq k \leq N_t$, N_t orizzonte di allenamento, indichiamo che ogni k passi il segnale del sistema fisico viene immesso nuovamente nella rete.

Nella seguente trattazione indicheremo con n-forecasting il forecasting dell'intero orizzonte temporale del sistema. In questa caso l'errore da minimizzare è relativo alla differenza tra l'intera dinamica del sistema lungo N_t e l'output della rete fatta evolvere in completa autonomia. In generale, in questo lavoro ci siamo concentrati principalmente sull'1-forecasting, usando l'n-forecasting per comprendere quanto la rete sia riuscita a generalizzare la dinamica del sistema. Allenamenti invece su generici k-forecasting, per $k \neq 1$, risultano in generale molto più difficili per il training della rete ottenendo, come mostrato dagli esperimenti qui riportati in figura 15, una minore precisione per lo stesso ordine di epoche. Nonostante singolarmente i k-forecasting, $k \neq 1$, risultino molto difficili per il training della rete, se opportunamente combinati tra loro, e sopratutto con l'1-forecasting, possono invece aumentare molto la generalizzazione della rete.

5.2.1 Curriculum learning

L'idea di base di tale training è quella di far allenare la rete inizialmente su forecasting molto piccoli, specialmente l'1-forecasting, al fine di ottenere una buona predicibilità locale del sistema, per poi andare gradualmente ad aumentare il forecasting sul quale si allena, per migliorarne la generalizzazione.

Nella pratica, ad ogni epoca viene fatto un k-forecasting con k estratto da una distribuzione gaussiana che varia durante le varie epoche. In particolare la media di tale gausianna è ottenuta parametrizzando una funzione sigmoide esponenziale con codominio $[1, C]$ (C valore massimo del forecasting su cui vogliamo addestrare la rete) come si può vedere in figura 16, mentre la deviazione standard di quest'ultima segue il grafico riportato in figura 17. Da un primo esperimento abbiamo osservato, come prevedibile, un aumento sostanziale del rumore nella loss function ma con una migliore generalizzazione da parte della rete. Purtroppo il tempo a nostra disposizione non ha reso possibile l'esecuzione di ulteriori esperimenti che rendessero statisticamente significative le nostre ipotesi.

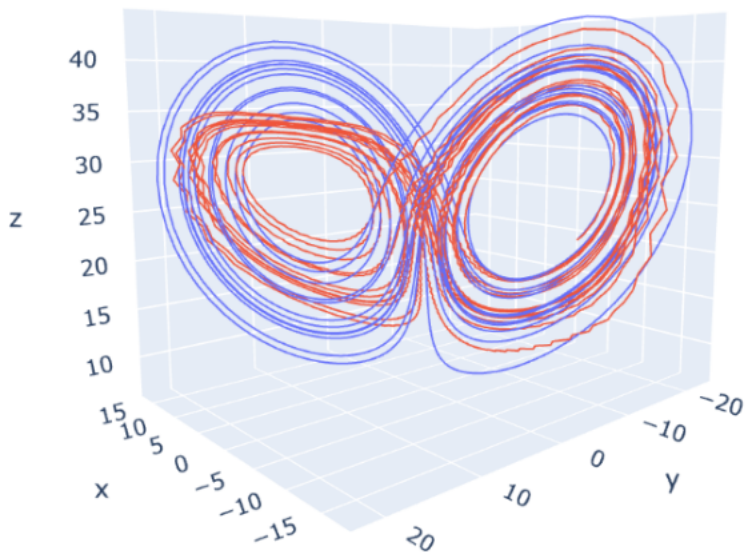


Figura 15: Previsione della traiettoria del modello di Lorentz, fino a tempo $T = 1000$, allenato per 10000 epoche su un 2-forecasting per una riserva con $N = 200$ neuroni, $\sigma_{in} = 0.15$, raggio spettrale $\lambda_{max} = 0.4$

5.2.2 Phased curriculum learning

Citiamo qui una generalizzazione del precedente discusso curriculum learning: il phased curriculum learning. Quest'ultimo consiste semplicemente nell'eseguire più cicli di curriculum learning durante il periodo di training. Questo ovviamente aggrava il problema di rumorosità della loss, sopratutto nel passaggio da forecasting con orizzonti temporali molto elevati all'1-forecasting che avviene quando si riavvia un ciclo di curriculum learning.

5.3 Taglia della riserva

Come detto in [6] l'orizzonte di predicibilità medio aumenta all'aumentare della dimensione della riserva. In generale la maggior parte degli esperimenti è stata eseguita avendo una riserva di dimensione pari a 200, tuttavia una prima serie di esperimenti variando la dimensione della riserva sembra corroborare la suddetta ipotesi.

Riportiamo qui una serie di esperimenti eseguiti allenando una rete con la classica funzione di transizione *tanh*, purtroppo dato che al crescere della dimensione della riserva cresce naturalmente anche il numero di parametri che possono essere allenati, ciò ha reso proibitivo, in termini di tempo, lo sviluppo di anche esperimenti della rete con funzione di transizione CMT2, dato che quest'ultima risulta in genere considerevolmente più lenta da allenare. Diminuendo la dimensione della riserva a 100, possiamo osservare un degradamento nelle performance dell' 1-forecasting come si può vedere in figura 18, ma sopratutto risulta una grande perdita di generalizzazione sull'n-forecasting. Aumentando invece la dimensione della riserva possiamo invece generalmente osservare un miglioramento della generalizzazione della rete, che qui ricordiamo essere intesa come quanto n-forecasting riesce ad essere preciso.

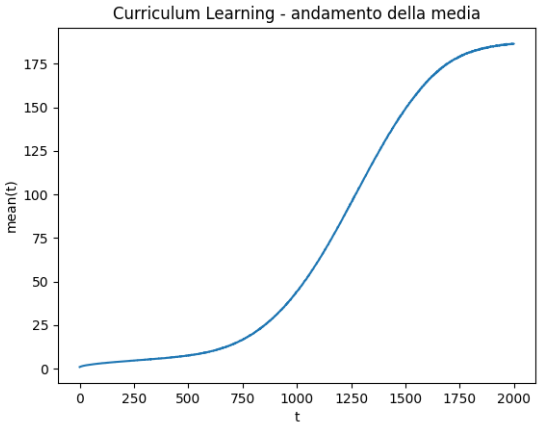


Figura 16: Media della gaussiana usata per curriculum learning

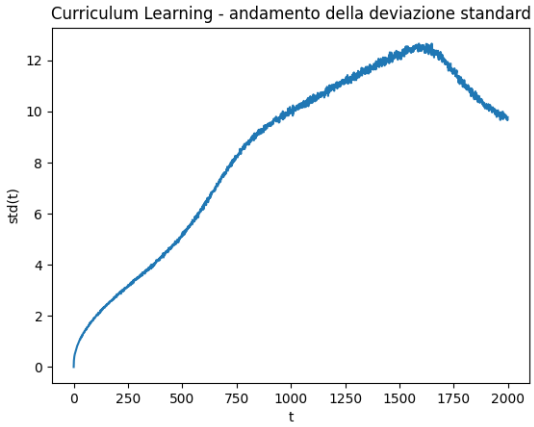


Figura 17: Deviazione standard della gaussiana usata per curriculum learning

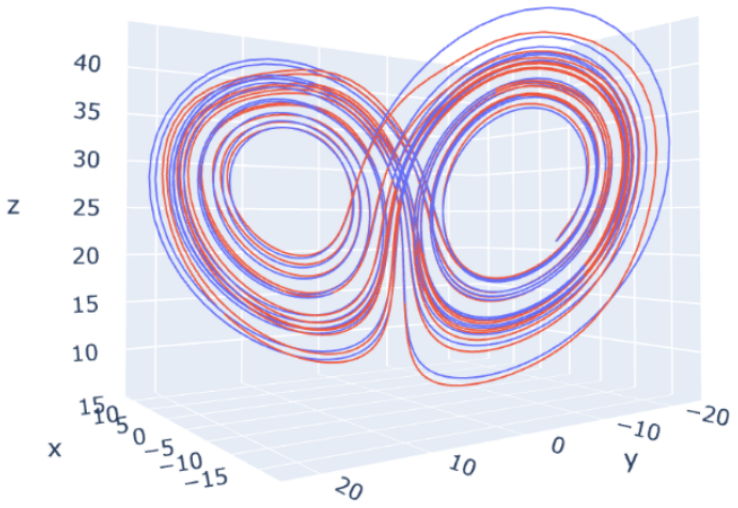


Figura 18: Previsione della traiettoria del modello di Lorentz, fino a tempo $T = 1000$, allenato per 10000 epoche per una riserva con $N = 100$ neuroni, $\sigma_{in} = 0.15$, raggio spettrale $\lambda_{max} = 0.4$

5.4 Loss multiple

Sebbene sia il curriculum learning sia il phased curriculum learning possano rappresentare due alternative molto allettanti per cercare di migliorare la generalizzazione della rete, il problema dell'eccessivo rumore introdotto nell'allenamento da questi due metodi potrebbe essere un problema non trascurabile. Una possibile soluzione a tale problema potrebbe essere di "integrare" i vari forecasting direttamente nella loss: con integrare intendiamo qui la costruzione di una loss che sia la somma di varie loss corrispondenti a diversi k-forecasting, ciò richiederebbe tuttavia la necessità di tenere in memoria più stati della rete, ognuno dei quali relativo ad un diverso forecasting.

6 Conclusioni

Nella sezione 2.3 abbiamo potuto osservare un interessante comportamento in t_{clean}, tempo necessario alla rete per cancellare la propria condizione iniziale: fino ad un certo valore del raggio spettrale $\lambda_{max} > 1$ il valore di t_{clean} cresce linearmente in λ con coefficiente lineare $m = 21.3 \pm 0.1$ indipendentemente dalla taglia della riserva N, dopodiché perde questo andamento in un piccolo intervallo, per raggiungere una divergenza all'infinito, cioè gli stati della riserva non si allineano più e si perde la *echo state property*. Il valore di λ_{max} per cui avviene la divergenza dipende dalla matrice $\mathbf{W}$, ma il coefficiente lineare della retta è gli è indipendente.

Siamo riusciti ad implementare con successo l'informazione fisica nel nostro modello ottenendo così un miglioramento delle prestazioni della rete.

L'utilizzo della funzione di attivazione CMT2 ha migliorato ulteriormente la capacità di apprendimento del sistema, risultando equivalente alla possibilità di modificare dinamicamente il raggio spettrale della matrice di adiacenza $\mathbf{W}$ ma senza modificare la struttura della riserva, diventando così uno strumento interessante per future versioni di questa ESN. Abbiamo sperimentato diversi metodi e parametri per il modello, ma, purtroppo, la rete ha dimostrato di aver bisogno di lunghi addestramenti e grande potenza computazionale e non siamo riusciti ad ottenere risultati competitivi. Proprio per questo motivo, i futuri sviluppi di questo progetto partirebbero da un attività sperimentale più intensiva, dove potremmo osservare i limiti delle architetture prese in esame. Inoltre sarebbe opportuno verificare l'impatto sulle performance di algoritmi di ottimizzazione differenti e di una più generale ottimizzazione della fase di addestramento, ad esempio pre-computando un *ensemble* di traiettorie su cui allenare la rete. Siamo curiosi di sperimentare diversi approcci per l'inclusione dell'informazione fisica, come nei modelli ibridi [8], oppure attraverso l'uso delle Neural Projections [12]. Inoltre, sarebbe interessante utilizzare qualcosa di diverso da un layer lineare per generare le predizioni a partire dall'evoluzione della riserva, come ad esempio una rete NEAT [13], in modo da trovare un sottoinsieme ottimale e informativo dei neuroni su cui basare le predizioni.

Riferimenti bibliografici

[1] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, pp. 2554–8, 05 1982.

[2] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, 01 2001.

[3] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 11 2002. [Online]. Available: https://doi.org/10.1162/089976602760407955

[4] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature communications*, vol. 5, p. 3541, 03 2014.

[5] H. O. Sillin, R. Aguilera, H.-H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Gimzewski, "A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing," *Nanotechnology*, vol. 24, no. 38, p. 384004, sep 2013. [Online]. Available: https://doi.org/10.1088/0957-4484/24/38/384004

[6] N. Doan, W. Polifke, and L. Magri, "Physics-informed echo state networks," *Journal of Computational Science*, vol. 47, p. 101237, Nov 2020. [Online]. Available: http://dx.doi.org/10.1016/j.jocs.2020.101237

[7] M. Kimura and R. Nakano, "Learning dynamical systems by recurrent neural networks from orbits," *Neural Networks*, vol. 11, no. 9, pp. 1589–1599, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608098000987

[8] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, no. 4, p. 041101, Apr 2018. [Online]. Available: http://dx.doi.org/10.1063/1.5028373

[9] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," *CoRR*, vol. abs/2001.04536, 2020. [Online]. Available: https://arxiv.org/abs/2001.04536

[10] L. Cedergren, "Physics-informed neural networks for biopharma applications," 2021.

[11] M. Cencini, F. Cecconi, and A. Vulpiani, *Chaos: from simple models to complex systems.* WORLD SCIENTIFIC, 2009. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/7351

[12] S. Yang, X. He, and B. Zhu, "Learning physical constraints with neural projections," 2020.

[13] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.