Denoising di segnali tramite software Knime utilizzando il modello LSTM Autoencoder

Internal Report

november 2022

AUTHOR
Luca Boni
lboni@daimon.bo.ismn.cnr.it

VERSION
1.0

Lista contenuti

Abstract

Nello studio di segnali complessi è necessario l'ausilio di algoritmi che comprendano il dato in più dimensioni. A questo proposito, se viene fornito un dataset abbastanza ampio, le reti neurali ricorrenti LSTM permettono tramite una rete ricorsiva l'apprendimento della struttura e la predizione del comportamento di segnali in ingresso anche molto complessi. Inoltre, esse possiedono dei meccanismi interni chiamati gate che possono regolare il flusso di informazioni e apprendere quali dati in una sequenza è importante conservare o eliminare, permettendo la conservazione di informazioni passate lontane. L'utilizzo di questi strumenti, seppur ampiamente evoluto fino ad oggi, necessita comunque di competenze di base molto ampie, il che implica la maggior parte delle volte il coinvolgimento di un programmatore specializzato. In realtà, è possibile sfruttare il potenziale delle reti neurali anche in modo grafico e intuitivo, adattando software di data science, quale Knime, a questo scopo.

Introduzione

Knime è un software di data science solitamente usato nell'analisi dei dati di attività commerciali. Grazie all'ausilio degli algoritmi del deep learning framework Keras mette a disposizione degli utenti la possibilità di utilizzare vari modelli di reti neurali. La storia del loro utilizzo su Knime e i più recenti articoli vedono impiegate le tecniche di machine learning soprattutto per aumentare le performance nella gestione e accessibilità di documenti o nella sicurezza informatica per il rilevamento delle intrusioni (IDS); sono invece praticamente nulli i suoi utilizzi nell'ambito della ricerca scientifica. Il lavoro presentato in questo report mira a compensare questa mancanza, esponendo un'applicazione delle capacità di deep learning di Knime per il denoising di segnali. Il framework sviluppato può potenzialmente trovare applicazioni nell'analisi di segnali ottenuti da dispositivi per interfacce con cellule neuronali sviluppati da istituti del CNR (Ana Isabel Borrachero-Conejo,Emanuela Saracino,Marco Natali,Federico Prescimone,Saskia Karges,Simone Bonetti,Grazia Paola Nicchia,Francesco Formaggio,Marco Caprini,Roberto Zamboni,Francesco Mercuri,Stefano Toffanin,Michele Muccini,Valentina Benfenati, 19/12/2018, *Electrical Stimulation by an Organic Transistor Architecture Induces Calcium Signaling in Nonexcitable Brain Cells*). In particolare, viene utilizzato il modello LSTM Autoencoder capace di scomporre il segnale in ingresso, capirne la struttura e ricostruirlo. Questo modello viene utilizzato per rilevare anomalie o per filtrare segnali, come ad esempio il suono. Similmente all'obiettivo di questo elaborato, il modello è stato utilizzato per il denoising dei segnali dell'attività cardiaca (ECG) dove si è dimostrato che grazie all'ausilio di reti neurali di deep learning si riescono a ottenere risultati sufficientemente affidabili. L'obiettivo, quindi, è riuscire nello stesso modo a pulire segnali complessi con una dinamica specifica; per rendere possibile una futura implementazione nella lettura dei segnali neurali.

1. Introduzione alle reti neurali

Le reti neurali sono modelli computazionali che permettono ai programmi informatici di riconoscere modelli e risolvere problemi comuni nei campi di AI, machine learning e deep learning, riflettendo il comportamento del cervello umano.

1.1 Neuroni

I neuroni sono i componenti di base della rete neurale. Essi prendono degli input, li moltiplicano con un peso, li sommano con una distorsione e poi la somma viene passata attraverso una funzione di attivazione per portare gli ingressi (che sono in un range illimitato) in una forma migliore, più prevedibile (sigmoid function = portare da (-infinito,+infinito) a (0,1)).

1.2 Funzioni di attivazione

Le funzioni di attivazione sono operatori matematici che vengono applicati ai time step sampling della RNN per farli passare da un modello lineare illimitato a uno non lineare limitato (come la tangente iperbolica tanh [da –infinito,+infiniti a –1,+1] o il rectified linear unit ReLU [da –infinito,+infinito a 0,+infinito])
Altre importanti sono:

- Soft-max: usata per problemi probabilistici (per esempio trovare un elemento specifico all'interno di un gruppo di campioni, dando una probabilità a ogni campione e prendendo quella più alta, considerando che la somma di tutti i valori di probabilità risulta 1 [quella selezionata per esempio varrebbe 0,75]).
- Sigmoide: usata per lo più nel metodo di apprendimento regressione della rete neurale, dove il modello effettua stime su dei valori numerici . Essa restituisce un valore compreso tra 0 e 1 tramite la funzione matematica da cui prende il nome.

1.3 LOSS

La funzione LOSS permette di arrivare al minore degli errori possibili: si tratta del criterio usato dalla rete neurale per identificare l'errore e che gli permette di ridurlo il più possibile. La più comune è il mean square error (MSE):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

n= numero di campionamenti, y= variabile predetta, ytrue= valore reale, ypred= output della nostra rete

1.4 Funzione di ottimizzazione

La funzione di ottimizzazione cambia il valore dei pesi durante i vari step di apprendimento basandosi sui risultati della LOSS. Tra i più comuni c'è lo Stochastic Gradient Descent (SGD)

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

η è il learning rate ed esprime quanto velocemente viene allenata la rete.
Facendo questo per tutti i pesi e le distorsioni del sistema si riduce l'errore, per poi riportare i risultati ottenuti all'ingresso e ripetere l'operazione, facendo tendere l'errore a 0.

2. Reti Neurali Ricorrenti (RNN)

2.1 Capacità

Nelle reti feed forward i neuroni non possono comunicare tra loro sullo stesso livello ma solo coi neuroni del livello successivo, mentre i neuroni delle RNN possono ammettere loop e/o comunicare con quelli dei livelli precedenti (introducendo il concetto di memoria).
La differenza si nota anche nei tipi di dati compatibili all'utilizzo: le feed forward elaborano ingressi statici (come immagini), mentre le RNN possono elaborare anche segnali variabili nel tempo (come video) e distinguerne le variazioni (ad esempio diversi comportamenti).

Immagine tratta da https://www.domsoria.com/2019/11/rnn-recurrent-neural-network/

2.2 Funzionamento

La RNN funziona con un meccanismo simile alle approssimazioni successive, utilizzando una retroazione con memoria nell'hidden state.

I primi dati vengono trasformati in vettori leggibili dalla macchina. Quindi la RNN elabora la sequenza dei vettori uno per uno grazie a un ciclo for. L'hidden state funge da memoria nelle RNN ed è lo step successivo della sequenza elaborata per essere ricordata. L'input e l'hidden state precedenti vengono poi combinati e i vettori arrivano così a contenere informazioni sull'input attuale e sugli input precedenti.
Il numero di cicli di elaborazione della RNN è limitato da parametri che possono indicare quando la percentuale di errore è abbastanza bassa da terminare il processo (iperparametri).

2.3 Unfolding della rete

Concetto importante per comprendere come addestrare una rete neurale ricorrente. Consiste nella trasformazione di una rete RNN in una di tipo feed forward decidendo a priori il numero di passi temporali su cui effettuare l'analisi. La cella è una parte della rete che conserva uno stato interno h(t) per ogni istante di tempo (costituita da un numero prefissato di neuroni, considerabile come un livello della rete).
RNN a N step temporali imposti = feed forward a N livelli, permettendo di svolgere i calcoli per la RNN come fatto per la feed forward:

$$h(t) = \varphi(x_t * w_x + h_{(t-1)} * w_h + b)$$

dove w_x e w_h sono i pesi, b è il bias e φ è la funzione di attivazione.

3. Long Short Term Memory (LSTM)

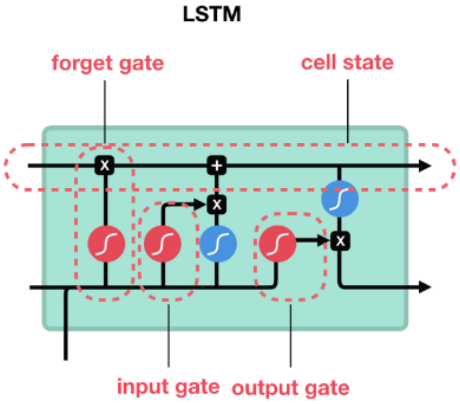
A differenza delle reti neurali classiche, la LSTM è una rete neurale artificiale con connessioni di feedback. Tale rete neurale ricorrente può elaborare non solo singoli dati (come immagini), ma anche intere sequenze.

3.1 Capacità

Le funzioni di ottimizzazione usate per aggiornare i pesi delle reti neurali diminuiscono di valore più si propagano indietro nel tempo, generando problemi con grandi serie di dati in ingresso (gradiente di scomparsa).
Ciò avviene perché i livelli che ottengono un piccolo miglioramento nel gradiente smettono di apprendere.
La LSTM ha dei meccanismi interni chiamati gate che possono regolare il flusso di informazioni; queste porte possono apprendere quali dati in una sequenza è importante conservare o eliminare.

3.2 Struttura

Immagine tratta da
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21



- FORGET GATE: l'input e lo stato interno precedente entrano in un sigmoid che decide quale informazione deve essere gettata o tenuta.
- GATE DI INPUT: portiamo il precedente stato interno e l'ingresso corrente a passare in una funzione di sigmoid che ne quantifica la rilevanza in scala da 0 a 1 e in un tanh che lo ricalibra tra –1 e 1 per aiutare a regolare la rete, dopodichè vengono moltiplicati nel gate l'uscita di tanh con quella della funzione sigmoid facendo rimanere solo le informazioni ritenute importanti dal sigmoid.
- STATO DELLA CELLA: passa dal gate di forget e viene abbassato se vicino a 0, poi viene sommato con il gate di input aggiornando lo stato della cella a nuovi valori che la rete trova rilevanti fornendoci il nuovo stato della cella.
- GATE DI OUTPUT: decide quale sarà il prossimo stato interno: vengono mandati in una sigmoid l'input corrente e lo stato interno precedente; l'uscita dei sigmoid viene moltiplicata con il valore della cella passato in un tanh dando vita al nuovo stato interno (=output) da trasmettere al passaggio temporale successivo.

3.3 Utilizzo della LSTM

I problemi principali da affrontare nella creazione di un framework utilizzatore di una rete LSTM sono:

- Cosa mettere in X e Y: dato dipendente e dato dipendenza. Non bisogna limitarsi a pensarli come array, poiché entrambi possono essere insiemi di dati in più dimensioni (quadrati di dati, cubi di dati o altro). Per modellare i dati da mettere in X e Y si utilizza il criterio della rolling window, con cui poi far apprendere la rete neurale. Questo metodo consiste nel riempire una lista con righe di dati lunghe quanto un numero di dati precedentemente scelto (m).
  La riga successiva parte dal secondo dato della riga precedente e ne mette alla fine uno nuovo della lista di partenza, e così via fino a terminare i dati e ad avere come ultimi elementi delle righe le n predizioni dei m-1 dati precedenti. La rolling window viene utilizzata soprattutto nelle analisi temporali.

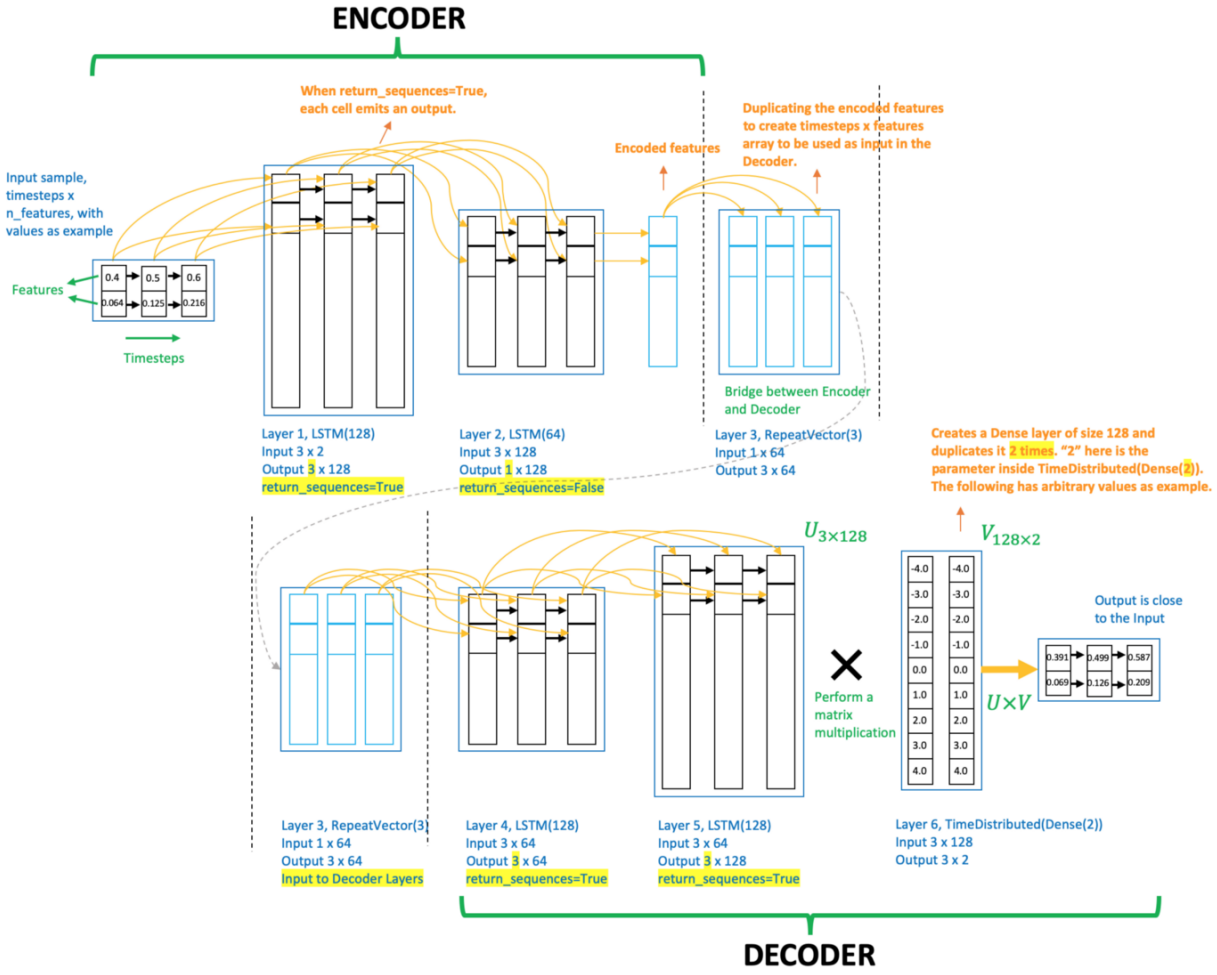
- Modello: quanti layer si vogliono utilizzare, le loro caratteristiche e in che modo vengono collegati (sequential …) per la profondità della rete.
- Algoritmo: scelta e applicazione della funzione di ottimizzazione (GDU, ADAM) e loss (MSE) che permettono il calcolo dell'errore e il cambio dei pesi nella rete neurale. Nel grafico finale dell'andamento dell'errore durante l'apprendimento il valore deve tendere a 0.

3.4 Modello LSTM Autoencoder

Gli autoencoder sono un tipo di modello di apprendimento auto supervisionato in grado di apprendere una rappresentazione compressa dei dati di input. Le reti neurali profonde usano questo modello durante l'addestramento di livelli nascosti per trovare una rappresentazione comune dei dati dall'input.
Grazie alla loro implementazione con le reti neurali LSTM, l'autoencoder riesce ad apprendere composizioni di dati complesse anche in sequenza nel tempo, come filmati o segnali temporali.
Il modello in questione è una rete neurale tipicamente costituita da un solo livello nascosto; imposta i valori target in modo che siano uguali all'input, comprendendo quindi la struttura che gli viene data in ingresso ricostruendola in uscita.
Dato un insieme di esempi di input χ, il training dell'autoencoder consiste nel trovare parametri θ={W1,W2,b1,b2] che minimizzino l'errore di ricostruzione, che corrisponde a minimizzare la seguente funzione obiettivo (LOSS):

$$\mathcal{J}(\theta) = \sum_{x \in \mathcal{X}} \|x - \tilde{x}\|^2$$

Qui un’immagine della struttura di autoencoder utilizzata in questo report



Nella parte di encoder il segnale viene compattato e scomposto per poi essere ricomposto nella parte di decoder, durante l’apprendimento la rete aggiorna i pesi che utilizza nella parte di decoder per ricostruire l’ingresso fino a che non è in grado di ricostruire con un errore adeguatamente basso il segnale che gli è stato dato

4. RNN autoencoder per il denoising di un segnale periodico

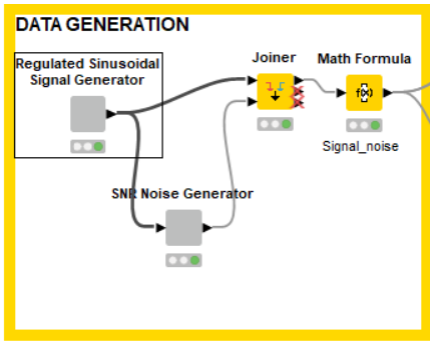
Prima di lavorare direttamente con segnali complessi come quelli derivanti da sensori elettrofisiologici, si è preferito testare la rete neurale autoencoder e le potenzialità di Knime su un segnale più semplice e meglio modellabile come quello sinusoidale. Questo ha reso più lineare la costruzione del framework che in seguito è stato adattato ai segnali complessi interessati.

4.1 Software utilizzato

Per la progettazione di questa rete viene utilizzato il software KNIME, ideato per creare e produrre data science, Ha anche al suo interno componenti di machine learning, in particolare nodi che utilizzano script di KERAS e contengono le strutture delle principali reti neurali e nodi per il loro apprendimento.

4.2 Generazione del dataset

L'architettura utilizza come segnale di ingresso inizialmente un generatore sinusoidale a cui viene sommato un rumore bianco. Il valore di quest'ultimo è distribuito tramite una gaussiana, una distribuzione di probabilità continua utilizzata per descrivere variabili casuali, il cui termine di varianza dipende dalla potenza media del rumore desiderato. Per scegliere questa potenza si utilizza la teoria dell'SNR (rapporto segnale rumore): trasformando il valore medio della potenza del segnale sinusoidale generato nel tempo in decibel, si può sottrarre il valore di SNR da quest'ultimo, trovando il valore medio del rumore; anti trasformandolo poi, possiamo metterlo come valore della varianza della distribuzione gaussiana. Quindi l'SNR viene scelto dall'utente per decidere l'entità del rumore da voler apportare al segnale.



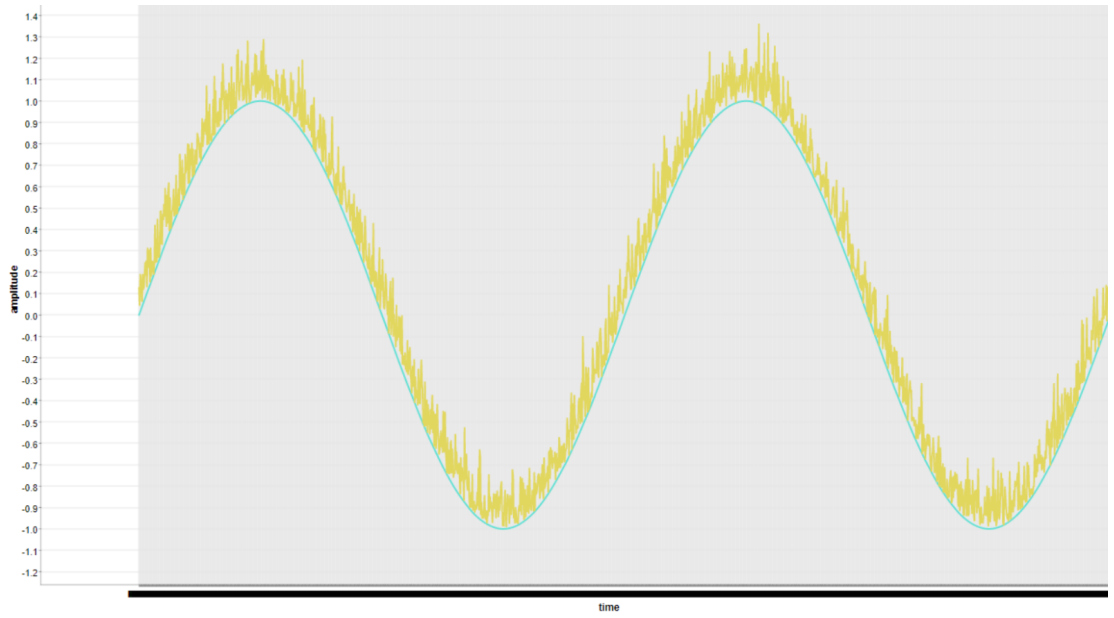
Più l'SNR ha un valore basso, più vi è un alto contenuto in termini di rumore.

4.3 Divisione tra train e test e rolling window

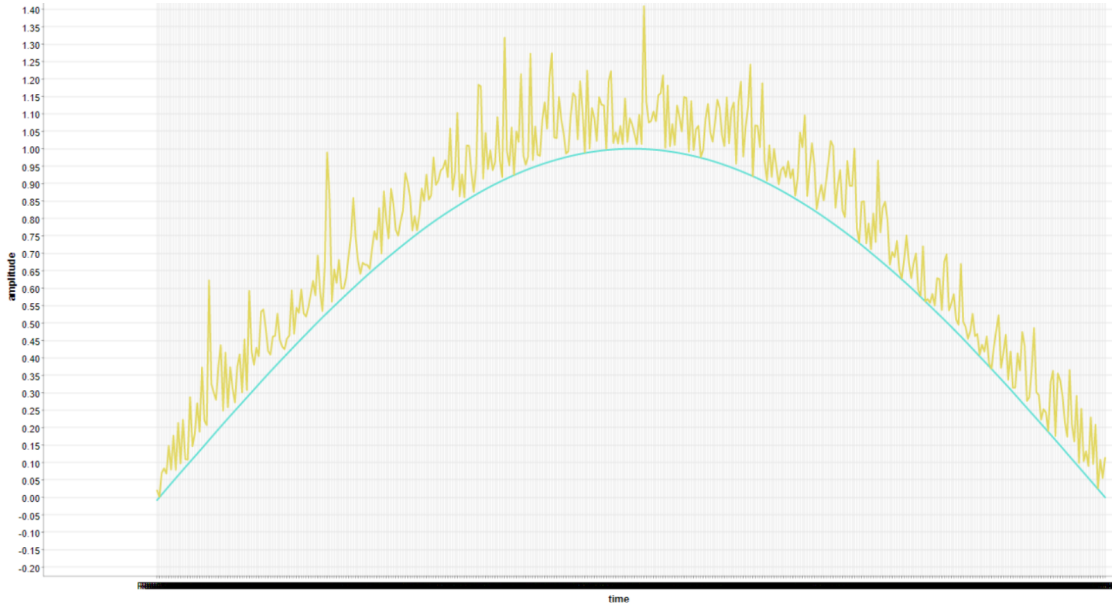
viene effettuato un sampling del segnale per un valore deciso in precedenza, per questo progetto l'intervallo di 100ms viene campionato 2000 volte, generando quindi 2000 dati da dover dividere in dati per allenare la rete e dati per testarla (1600 per allenarla e 400 per testarla):

Rappresentazione grafica del segnale di train e di test:

TRAIN_DATA

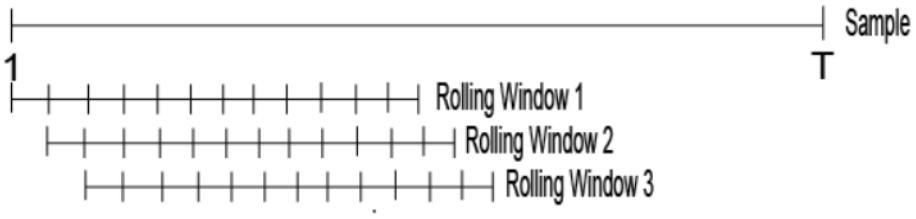


TEST_DATA

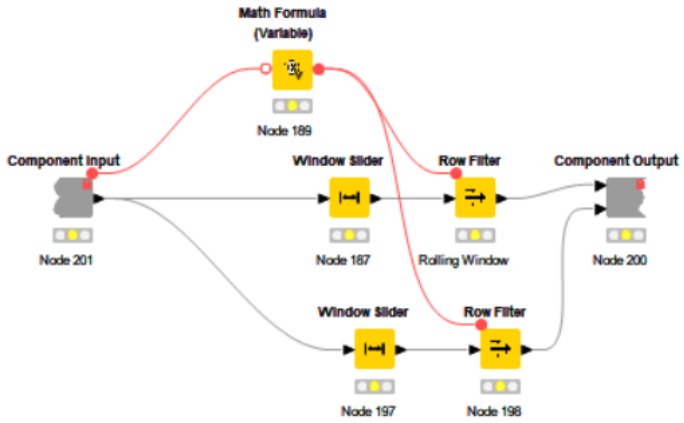


Per rendere poi la tabella di dati di Training interfacciabile con la rete di LSTM si rimodella quest'ultima nella forma Rolling-window: una distribuzione dei dati sorgente, che permette la previsione del dato futuro di una serie temporale basata non solo sull'istante prima, ma su una "finestra" di dati passati. Viene utilizzata per modellare le serie temporali come quelle che elabora la LSTM e ha la seguente struttura:

Immagine tratta da https://www.mathworks.com



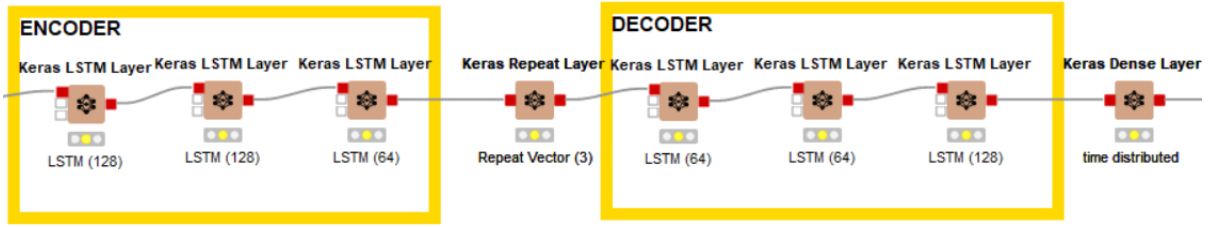
Viene implementata su KNIME utilizzando una rolling window di 40 elementi



4.4 Modello e architettura della rete

Modellati tutti i dati necessari per il training della rete, si definisce l'architettura utilizzata. Essendo lo scopo del progetto il denoising del segnale in ingresso, si utilizza la rete LSTM Autoencoder esposta in precedenza.

L'idea di utilizzo dell'autoencoder per il denoising è di forzare il livello nascosto a recuperare funzionalità più solide e impedirgli di apprendere semplicemente l'identità, l'autoencoder viene addestrato a ricostruire l'input da una versione danneggiata di esso.
Su Knime questo viene implementato tramite i layer Keras nel seguente modo:

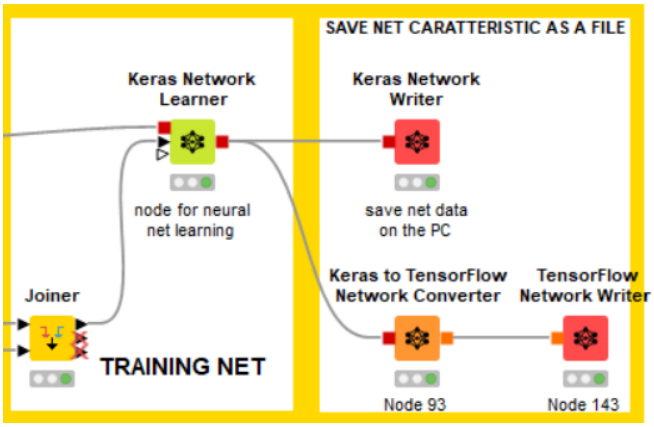


Sono stati utilizzati 3 layer LSTM per l'encoding (la decomposizione del segnale) e 3 per il decoding (ricostruzione del segnale), in questo modo scompone il segnale sporco e impara a ricostruire un segnale pulito basato sul segnale sporco appena decomposto.

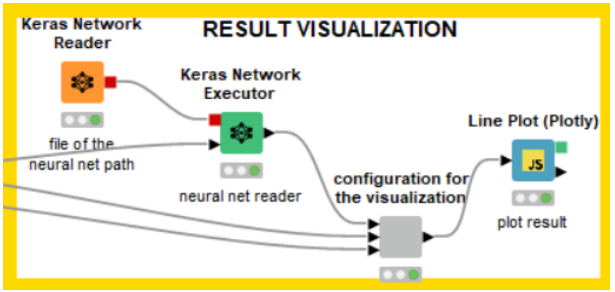
4.5 Allenamento della rete e risultati

All'ingresso del layer di learning Keras sono state date in input le rolling windows del segnale sporco e pulito di training, in seguito viene allenata la rete autoencoder con 100 epoche (iperparametro che definisce il numero di volte in cui l'algoritmo di apprendimento funzionerà attraverso l'intero set di dati di addestramento) e una batch size di 32 (iperparametro che definisce quanti campioni elaborare prima di aggiornare i parametri del modello intero). Terminato il training, vengono salvati i pesi e i dati riguardanti la rete in un file .h5 da cui poi si potrà andare a leggere per testare la nostra rete.

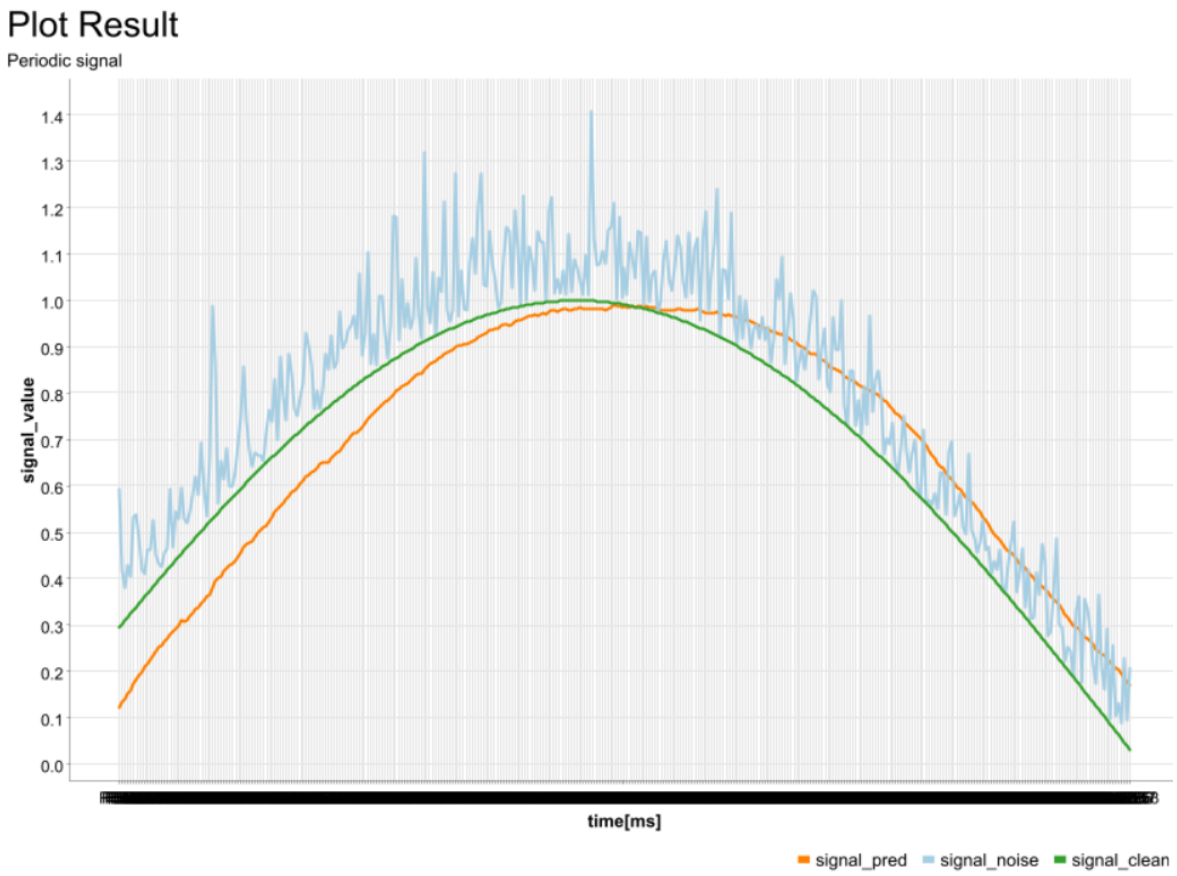
La rete di nodi Knime utilizzata per il training è la seguente:



Di seguito riporto la rete usata per la visualizzazione dei risultati e i grafici correlati:



Andamento della LOSS:



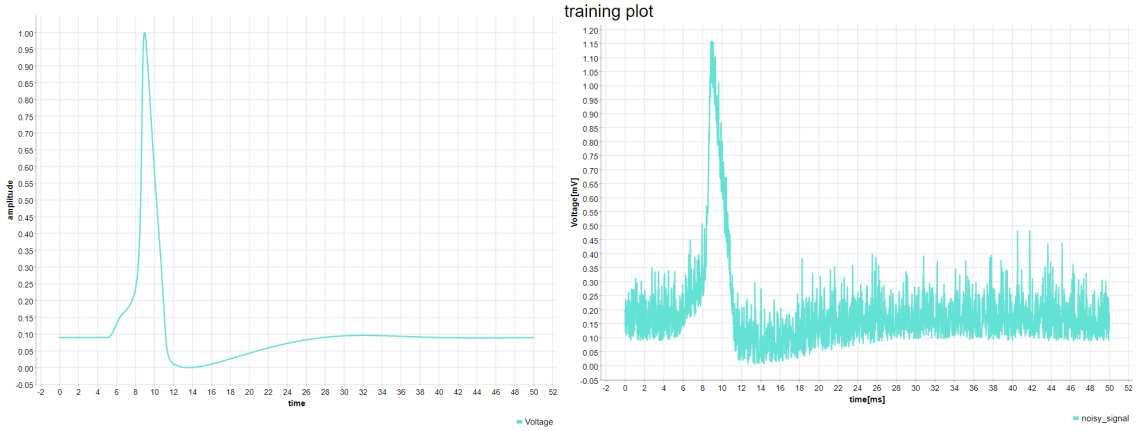
Si può notare dall'andamento della LOSS, che l'errore della predizione diminuisce come atteso, facendo tendere l'errore quadratico medio a 0, questo confermato dal grafico ampiezza-tempo che rende visibili il segnale predetto, il segnale sporco e il segnale pulito reale; notando che il segnale predetto ha una forma sufficientemente fedele al segnale pulito reale.

5. Denoising di un segnale complesso con dinamica specifica

5.1 Implementazione del framework

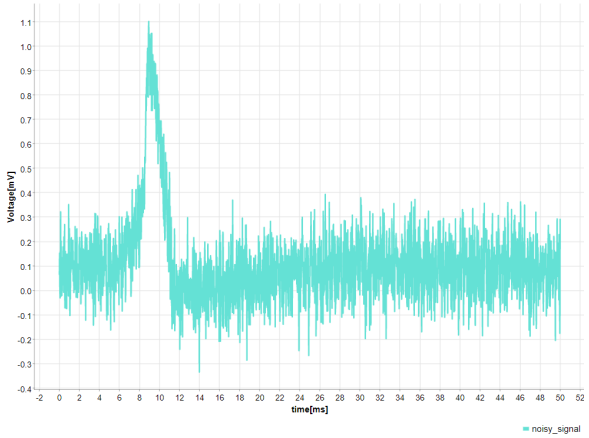
Utilizzando il framework del Progetto precedentemente esposto si variano gli input della rete, mandando in ingresso alla parte di training un file .csv contenente 2000 dati che descrivono l'impulso pulito di un segnale complesso. Quest'ultimo è stato generato tramite la libreria di python che sfrutta l'ambiente di simulazione Neuron: un potente motore per la simulazione di neuroni e reti neurali biofisiche. Cercando in questo modo di far avvicinare il più possibile la struttura del segnale generato con quella dell'impulso di un singolo neurone.

I valori del segnale, quindi, vengono normalizzati tra 0 e 1 e gli viene aggiunto un rumore bianco con la tecnica SNR come fatto in precedenza sul segnale sinusoidale.

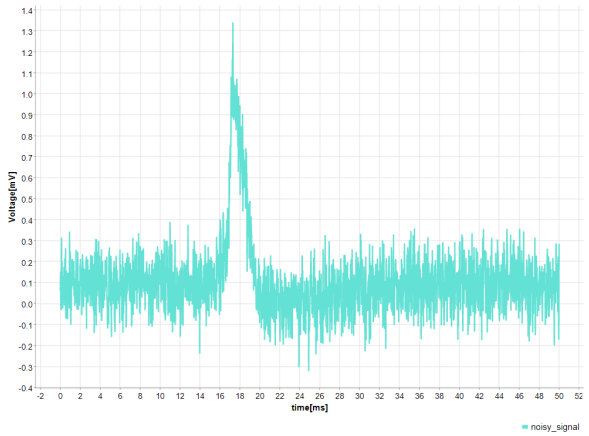


Per quanto riguarda il segnale di test, è stato utilizzato un segnale composto da 2000 dati normalizzati, come per quello di training, a cui è stato aggiunto un rumore con scala di ampiezza e valore SNR differenti. La sua forma è stata generata in modo da esaminare 3 diverse casistiche :

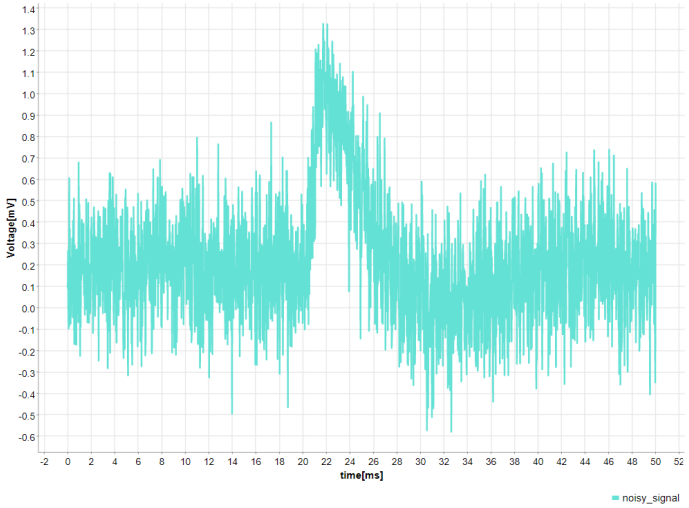
1. Viene utilizzato lo stesso segnale usato nel training ma con noise diverso in modo da accertarsi della ripetibilità dei risultati della rete prima di procedere con test più complessi



2. Vengono variate struttura del segnale e posizione dell'impulso ma mantenendo la stessa scala di valori. Questo test determina se la rete è in grado di pulire un qualsiasi segnale della stessa tipologia di quello usato per allenare la rete.

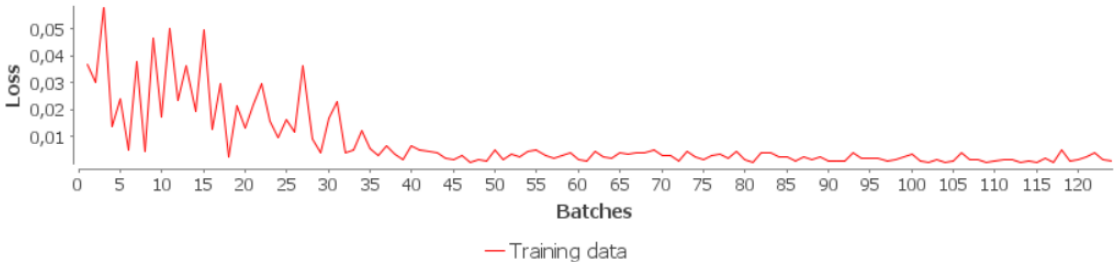


3. Infine viene sconvolta la struttura e la scala di valori del segnale, mantenendo solo le macro caratteristiche in modo da testare i limiti di precisione di questa rete.

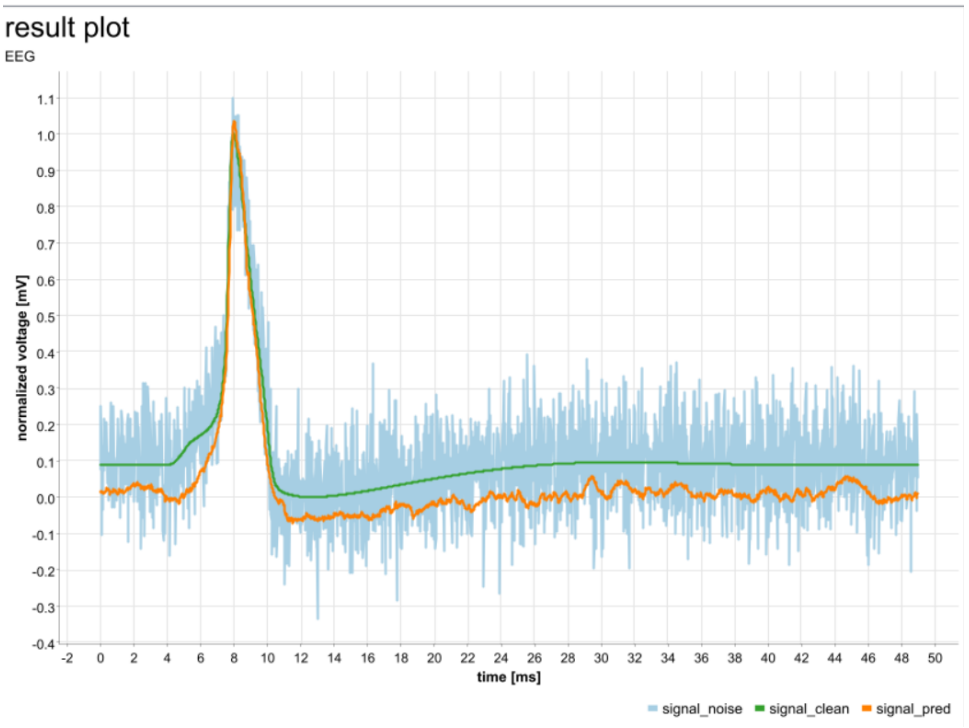


5.2 Risultati

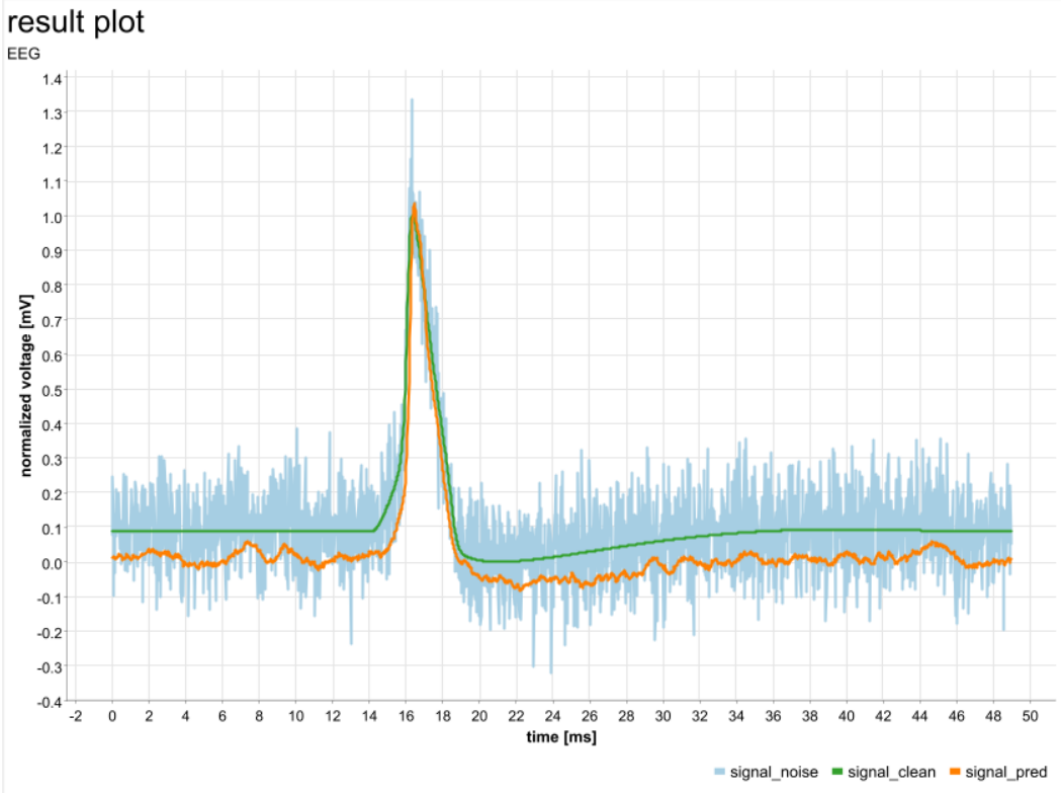
Andamento della LOSS



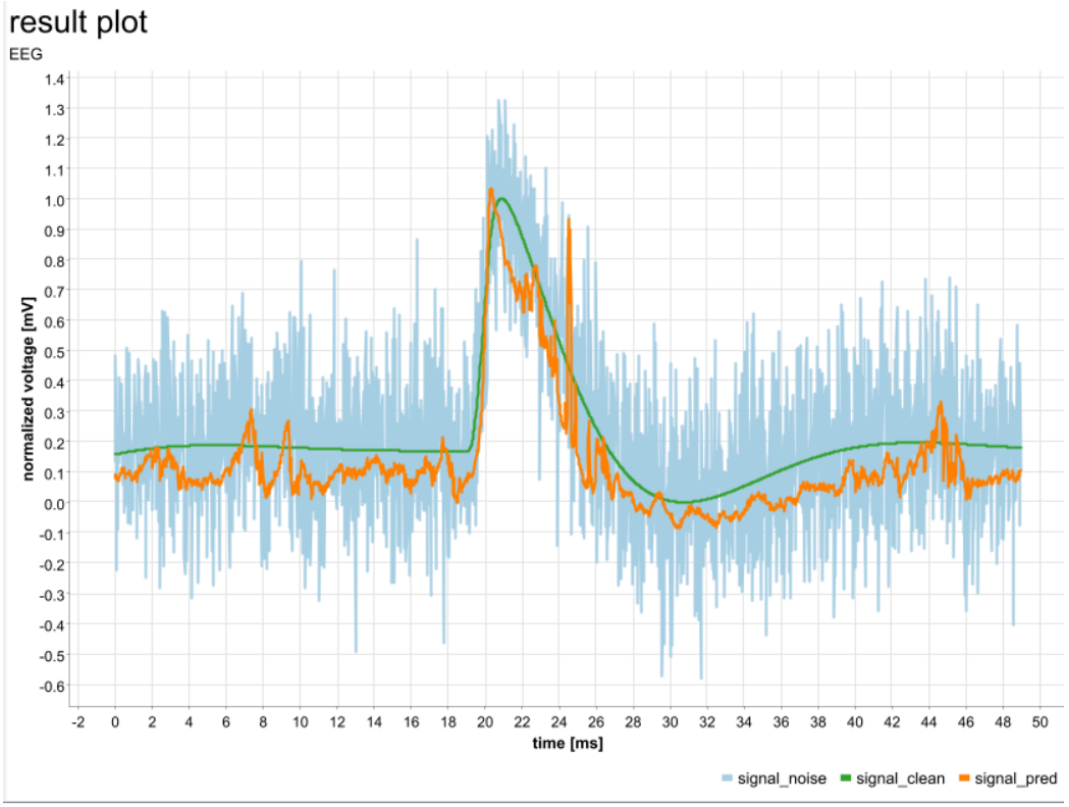
1. Stesso segnale del training



2. Segnale diverso dal training ma stessa scala di valori



3. Segnale completamente variato rispetto a quello di training



Conclusioni

Si è presentata una soluzione per rendere più veloce e precisa la lettura di segnali complessi, sottoposti a rumori ingenti, che ne compromettono la comprensione. Questo è stato possibile grazie alla costruzione “step by step” di un framework di denoising, tramite il modello LSTM autoencoder e le sue applicazioni in ambito di analisi dei segnali. I risultati ottenuti dallo studio di questi specifici segnali artificiali, essendo la rappresentazione ideale dell’impulso di un singolo neurone, portano potenzialmente questo progetto a poter essere utilizzato in futuro nello studio degli astrociti compiuto dalla CNR di bologna (Emanuela Saracino, Luca Maiolo, Davide Polese, M. Semprini, Ana Isabel Borrachero-Conejo, Jacopo Gasparetto, Stefano Murtagh, Margherita Sola, Lorenzo Tomasi, Francesco Valle, Luca Pazzini, Francesco Formaggio, Michela Chiappalone, Saber Hussain, Marco Caprini, Michele Muccini, Luigi Ambrosio, Guglielmo Fortunato, Roberto Zamboni, Annalisa Convertino, and Valentina Benfenati, *A Glial-Silicon Nanowire Electrode Junction Enabling Differentiation and Noninvasive Recording of Slow Oscillations from Primary Astrocytes*) come alternativa alle passate tecniche utilizzate per la visualizzazione dei risultati, basate sulla trasformata di Fourier. Tutto questo tramite il software Knime,utilizzato in ambito di ricerca scientifica, portando ipoteticamente l’utilizzo delle reti neurali anche alla portata di ricercatori o in generale utenti senza una particolare formazione a riguardo; essendo, questa rete, stata progettata senza l’ausilio di alcuno script informatico, ma con solo le rappresentazioni grafiche funzionali di cui questo software dispone.

Link al progetto su GitHub

https://github.com/boni3099/Autoencoder-project-CNR-.git

Fonti utilizzate

- https://neuronsimulator.github.io/nrn/tutorials/scripting-neuron-basics.html#What-is-NEURON?
- https://www.mathworks.com
- https://towardsdatascience.com/
- https://www.domsoria.com/2019/11/rnn-recurrent-neural-network/
- Marchi, E., Vesperini, F., Eyben, F., Squartini, S., Schuller, B. (2015). *A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).
- G Ciullo, *Introduzione al Laboratorio di Fisica, 2014 – Springer*
- Karol Antczak, *Deep Recurrent Neural Networks for ECG Signal Denoising*
- D. Puthankattil Subha, Paul K. Joseph, Rajendra Acharya U, Choo Min Lim (2019) *EEG Signal Analysis: A Survey*
- Jason Brownlee on November 5, 2018 in Long Short-Term Memory Networks. *A Gentle Introduction to LSTM Autoencoders*
- Syed Muzamil Basha1 , K. Bagyalakshmi , C. Ramesh , Robbi Rahim , R. Manikandan and Ambeshwar Kumar (2019). *Comparative Study on Performance of Document Classification Using Supervised Machine Learning Algorithms: KNIME*
- Munther Abualkibash (18/11/2019). *Machine Learning in Network Security Using KNIME Analytics*
- H.D.Nguyen, K.P.Tran, S.Thomassey, M.Hamad (04/2021). Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management
- Mohamed R. Ibrahim, James Haworth, Aldo Lipani, Nilufer Aslam,Tao Cheng, Nicola Christie (28/01/2021) *Variational-LSTM autoencoder to forecast the spread of coronavirus across the globe*
- Maximilian Grobbelaar, Souvik Phadikar, Ebrahim Ghaderpour, Aaron F. Struck, Nidul Sinha, Rajdeep Ghosh, d. Zaved Iqubal Ahmed (2022). *A Survey on Denoising Techniques of Electroencephalogram Signals Using Wavelet Transform*
- Emanuela Saracino, Luca Maiolo, Davide Polese, M. Semprini, Ana Isabel Borrachero-Conejo, Jacopo Gasparetto, Stefano Murtagh, Margherita Sola, Lorenzo Tomasi, Francesco Valle, Luca Pazzini, Francesco Formaggio, Michela Chiappalone, Saber Hussain, Marco Caprini, Michele Muccini, Luigi Ambrosio, Guglielmo Fortunato, Roberto Zamboni, Annalisa Convertino, and Valentina Benfenati, *A Glial-Silicon Nanowire Electrode Junction Enabling Differentiation and Noninvasive Recording of Slow Oscillations from Primary Astrocytes*