POLITECNICO DI TORINO

Collegio di Ingegneria Chimica e dei Materiali

Corso di Laurea Magistrale
in Ingegneria Chimica e dei Processi Sostenibili

Tesi di Laurea Magistrale

Tecnologia *transformers* per l'identificazione della dinamica di processi industriali

Relatore

prof. Davide Fissore

Candidato

Bersano Gabriele

Ottobre 2025

INDICE

Capitolo 1

Introduzione

Con l'avvento della rivoluzione 4.0, l'industria di processo e in particolare quella chimica sta vivendo una profonda trasformazione grazie all'introduzione del *machine learning* [1]. In questo contesto, l'identificazione di processo a partire da dati d'impianto riveste un ruolo fondamentale: essa consente di sviluppare modelli predittivi accurati in grado di rappresentare fedelmente il comportamento dinamico del sistema [2]. Tali modelli non solo aumentano la sicurezza operativa, ma permettono anche di ottimizzare l'efficienza del processo, riducendo al contempo la dipendenza da strumenti diagnostici tradizionali, spesso complessi e costosi. Le reti neurali, grazie alla loro capacità di apprendere rappresentazioni complesse e in parallelo dai dati, puntano a dare previsioni accurate e a identificare quelle relazioni non lineari tipiche di impianti chimici complessi [3]. In particolare, in questo contesto verrà adottata la metodologia dei *transformer*, una delle tecniche più avanzate nell'ambito dell'analisi delle serie temporali. Grazie alla loro architettura basata sui meccanismi di attenzione, i *transformer* risultano particolarmente efficaci nel catturare le dipendenze a lungo raggio tra le variabili di processo. Queste caratteristiche li rendono una soluzione all'avanguardia, in grado di superare le prestazioni dei modelli tradizionali e di migliorare significativamente la capacità predittiva del sistema [4]. La Tesi sarà articolata in due sezioni volte entrambe a catturare la dinamica del processo in esame. Verranno analizzati due sistemi MISO, cioè in cui si ha una variabile controllata e più di una variabile manipolabile. La prima parte della Tesi si concentrerà su un sistema MISO 2x1, in cui si hanno 2 variabili manipolate e una controllata. Si tratta di una colonna di separazione propano-propilene, modellata in dettaglio attraverso un simulatore di processo commerciale, usato come surrogato del processo vero e proprio. In questo caso, l'attenzione è rivolta all'analisi della purezza del prodotto in testa alla colonna, basandosi su simulazioni condotte in ambiente virtuale conoscendo le condizioni operative d'impianto. La seconda parte riguarda l'analisi di dati reali provenienti da un impianto industriale, in particolare da un forno per la conversione di etano in etilene. Qui il sistema analizzato è di 3x1, cioè con 3 variabili manipolate, cosa che rende più complessa l'identificazione e la predizione.

In una fase iniziale, vengono analizzati i dati raccolti in condizioni operative stazionarie; successivamente, si generano dati di sintesi mediante un modello del sistema, al fine di prevedere la risposta dinamica del processo a perturbazioni di tipo gradino. Lo svolgimento di questa Tesi è stato possibile grazie alla preziosa collaborazione con l'azienda Alpha Process Control, alla quale desidero esprimere la mia più sincera gratitudine per il supporto fornito, l'assistenza costante e la grande professionalità dimostrata durante tutte le fasi del lavoro.

1.1. Le serie temporali e disturbi a gradino

Una serie temporale è un insieme di valori raccolti in successione con intervalli di tempo costanti (minuti, ore, giorni, settimane), quindi una raccolta di valori discreti nel tempo [5]. In questo progetto di Tesi sono state utilizzate serie temporali multivariate, serie temporali in cui più variabili sono state misurate simultaneamente e analizzate in funzione del tempo. Ogni osservazione è costituita da un vettore di valori scalari, dove ogni componente rappresenta una variabile diversa [5]. Sempre rimanendo nel contesto di questo lavoro di Tesi, una serie temporale multivariata potrebbe corrispondere alla portata della alimentazione in ingresso, alla pressione d'uscita (COP) e alla temperatura d'uscita (COT) da un forno ad etano in funzione del tempo, come illustrato in Figura 1.1.

Figura 1.1: Serie temporale multivariata: portata della alimentazione in ingresso, pressione d'uscita (COP) e temperatura d'uscita (COT) da un forno ad etano in funzione del tempo.

Questo tipo di serie temporale in condizioni di lavoro pressoché stazionarie, non permette tuttavia la totale comprensione del processo essendo tali dati poco variabili. Per questo sono

stati creati dei disturbi a gradino, un esempio dei quali è illustrato in Figura 1.2, per comprendere a pieno la dinamica del sistema (come si analizzerà meglio nei prossimi capitoli).

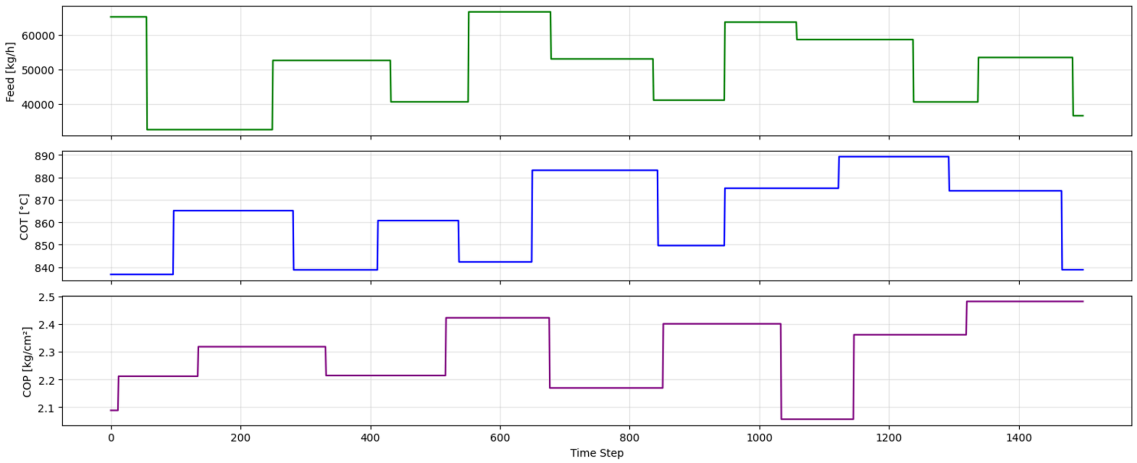


Figura 1.2: disturbi a gradino delle variabili manipolate in ingresso ad un forno ad etano

1.2. Fondamenti dello *splitter* propano-propilene

La distillazione è un processo di separazione termica usato per separare componenti di una miscela liquida in base alle differenze di volatilità. È largamente impiegata in processi della chimica industriale e raffinerie. Una colonna di distillazione continua opera in condizioni stazionarie con un flusso costante di alimentazione (*feed*) e si basa su due componenti principali il condensatore in cima e il ribollitore in fondo alla colonna. L'alimentazione liquida o una miscela binaria, viene immessa a una certa altezza nella colonna. Il ribollitore fornisce calore (spesso tramite vapore) alla base della colonna e una parte del liquido viene vaporizzata e risale la colonna e il residuo liquido meno volatile viene prelevato dal fondo. Il contatto tra vapore ascendente e liquido discendente avviene nei piatti (*trays*) oppure in materiali di riempimento (*packing*). In cima alla colonna avviene la condensazione, Il vapore ricco del componente più volatile arriva in cima e un condensatore lo trasforma in liquido. Questo liquido condensato in parte può essere reimmesso nella colonna per migliorare la purezza, il resto viene prelevato come distillato.

La separazione tra propano e propilene rappresenta una delle sfide più complesse nella progettazione di sistemi di distillazione industriale. Entrambi i componenti, appartenenti alla famiglia degli idrocarburi C3, presentano proprietà fisiche molto simili, con punti di ebollizione a bassa temperatura e una volatilità relativa che si aggira intorno a 1.1–1.2. Questo

valore così vicino all'unità implica che la spinta termodinamica alla separazione è estremamente debole, e pertanto è necessario ricorrere a colonne molto efficienti e ben ottimizzate per ottenere una buona separazione [6].

Il dispositivo impiegato per questa operazione è noto come *splitter* propano-propilene (o *C3 splitter*), una colonna di distillazione a frazionamento elevato che può contenere anche 150-200 piatti teorici e per questo motivo spesso si divide il processo in due colonne (figura 1.3). Il propilene, più volatile, tende a raccogliersi in testa alla colonna, mentre il propano, leggermente meno volatile si arricchisce in fase liquida e viene ritirato al fondo. Tuttavia, data la minima differenza di volatilità e la ristretta finestra di separazione, spesso 1–2 °C per stadio teorico, l'intero processo deve essere progettato con estrema precisione.

Le condizioni operative tipiche vedono la colonna funzionare a pressioni comprese tra 20 e 30 bar, in modo da alzare il punto di ebollizione dei componenti, con un *reflux ratio* elevato (circa 10) per favorire la purezza in testa e una potenza termica significativa (4 MMkcal/hr) al ribollitore per mantenere il profilo di temperatura desiderato lungo tutta la colonna.

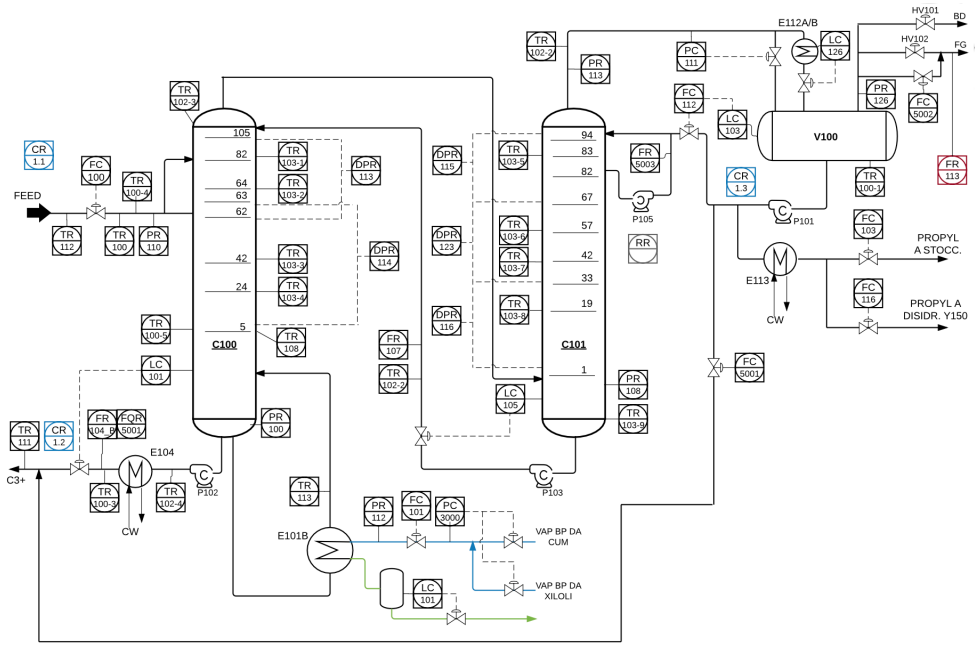


Figura 1.3: Schema del C3 *splitter*. Figura fornita da Alpha Process Control

La colonna di separazione propano–propilene (*C3 splitter*) è un componente fondamentale nella fase di *post-cracking* dell'*ethylene unit.* Il *feed* della C3 splitter proviene direttamente

dall'impianto di *steam cracking*, dove materie prime come nafta o gas leggeri vengono sottoposte a cracking termico ad alta temperatura (~850 °C) in presenza di vapore. Questo processo genera una miscela complessa di idrocarburi leggeri, principalmente C1–C4, tra cui propano (C3H8) e propilene (C3H6). I prodotti gassosi del *cracking* vengono successivamente inviati alla sezione di frazionamento gas (*gas concentration/fractionation*), un insieme di colonne che separano i componenti in base al numero di atomi di carbonio (figura 1.4) [7]. In particolare, il *depropanizer* rimuove la frazione C3 (propano + propilene) separandola dai componenti più pesanti (C4+). È a valle di questo stadio che entra in gioco il *C3 splitter*, deputato alla separazione finale tra propano e propilene. Questa separazione è di fondamentale importanza dal punto di vista industriale. Il propilene è infatti una delle principali materie prime dell'industria petrolchimica, utilizzato per la produzione di una vasta gamma di prodotti a elevato valore aggiunto, come il polipropilene (PP), uno dei polimeri termoplastici più diffusi al mondo. Al contrario, il propano rappresenta un sottoprodotto a minor valore commerciale, generalmente utilizzato come combustibile o gas di processo. Riuscire a separare in modo efficace il propilene dal propano è dunque essenziale per massimizzare la redditività dell'impianto e fornire alle unità a valle (come gli impianti di polimerizzazione) un flusso di propilene ad alta purezza.

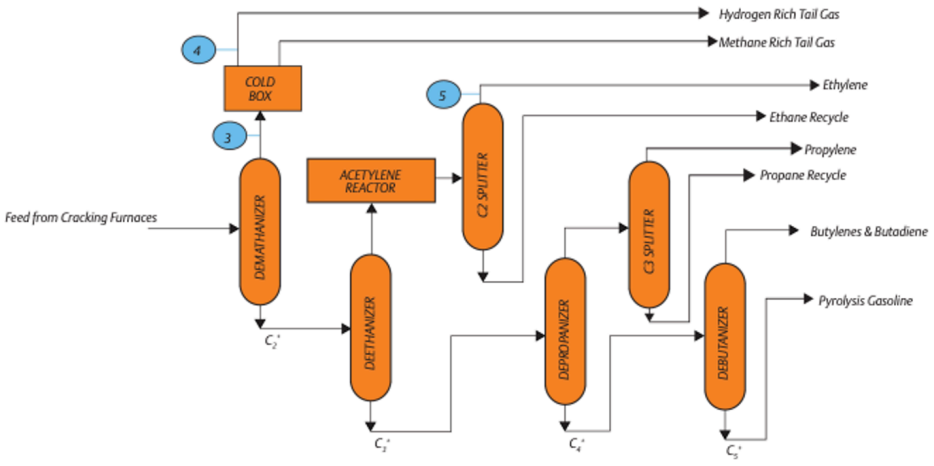


Figura 1.4: Schema riassuntivo della fase di *gas concentration/fractionation.* Figura tratta da [7] e riprodotta con modifiche.

1.3. Fondamenti del forno di *steam cracking* dell'etano

Il processo di *steam cracking* dell'etano rappresenta uno degli snodi fondamentali nella produzione industriale di etilene, una delle olefine più importanti nel panorama petrolchimico. Questo processo si basa su una pirolisi termica, in cui l'etano viene scaldato fino a temperature elevate, tipicamente intorno agli 800–850 °C, in presenza di vapore d'acqua (figura 1.5). Il vapore non partecipa direttamente alla reazione chimica, ma ha un ruolo essenziale come diluente, riducendo la pressione parziale dell'etano e ostacolando la formazione di *coke* [8].

La reazione avviene in condizioni estremamente rapide: il tempo di residenza del gas all'interno dei tubi del forno è dell'ordine di millisecondi. Questo è un compromesso ingegneristico: tempi troppo lunghi favorirebbero la formazione di prodotti indesiderati e di coke, mentre tempi troppo brevi non permetterebbero una conversione efficace. Per bloccare la reazione al momento ottimale, il gas in uscita dal forno viene immediatamente raffreddato in un *Transfer Line Exchanger* (TLE), un'unità che consente un "quench" termico istantaneo e che recupera calore generando vapore ad alta pressione [8].

Nonostante le elevate temperature, la conversione dell'etano in un singolo passaggio è relativamente bassa, spesso inferiore al 70%, motivo per cui si implementano circuiti di ricircolo che reinviano l'etano non reagito all'ingresso del forno. Questo migliora il rendimento complessivo del processo.

L'obiettivo centrale del *cracking* è massimizzare la resa in olefine leggere (soprattutto etilene), riducendo la formazione di prodotti secondari come *tail gas* (metano, idrogeno, ecc.). La selettività verso l'etilene è favorita da condizioni di *cracking* "dolci", cioè alta temperatura e bassi tempi di permanenza, nonché da un buon controllo della cinetica di reazione attraverso il profilo di temperatura all'interno del forno.

Per arrivare a questo obbiettivo posso agire principalmente su 3 variabili manipolate: la temperatura, la velocità, la pressione (figura 1.6).

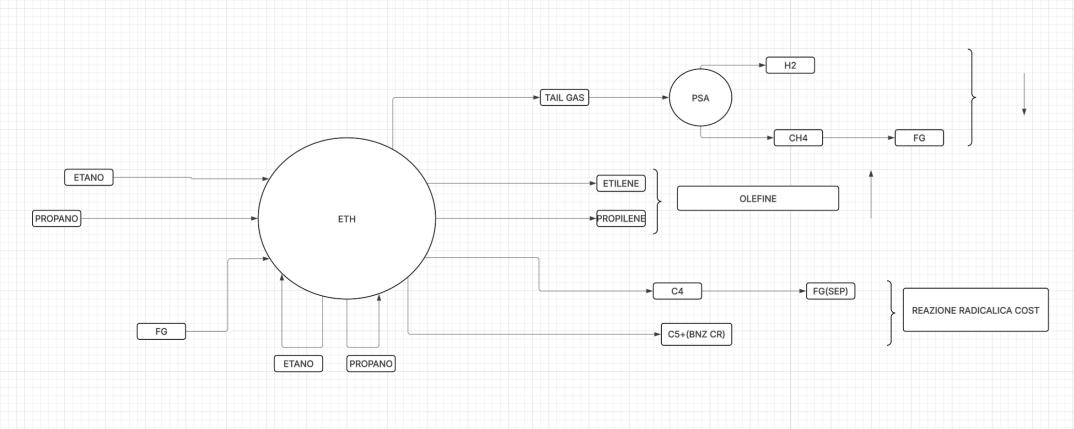


Figura 1.5: schema del forno ad etano

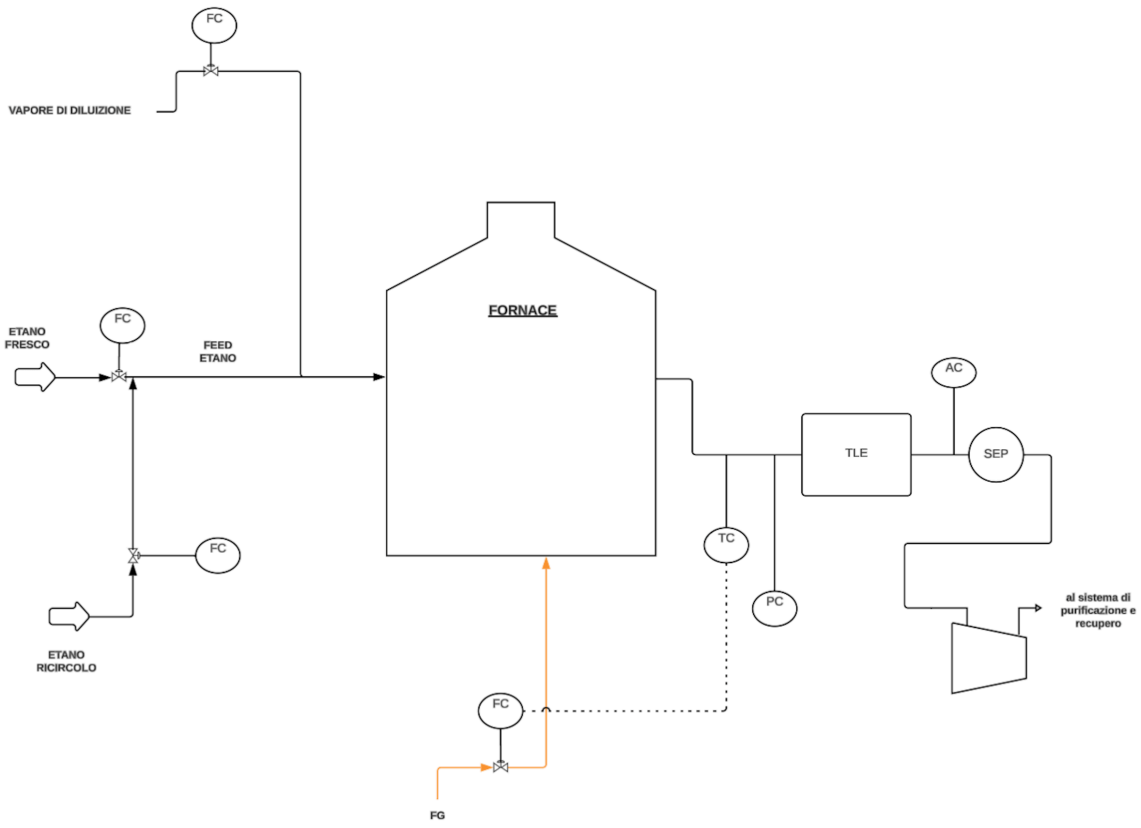


Figura 1.6: *Process flow diagram* forno ad etano.

Capitolo 2

Modelli teorici implementati

2.1. Reti Neurali

Per poter parlare dei *transformers* bisogna prima fare una precisazione sulle reti neurali artificiali (*Artificial Neural Networks*, ANN). Le ANN sono modelli computazionali ispirati al funzionamento dei neuroni biologici. Il loro scopo è apprendere una funzione di mappatura tra un input e un output attraverso un processo di addestramento. Questi modelli sono oggi alla base di molte applicazioni nell'ambito del *machine learning* e del *deep learning* [9].

2.1.1 Reti Neurali a singolo strato

Il principio base di funzionamento di queste reti risiede nei neuroni che ricevono in input delle informazioni $x=(x_1, x_2, \ldots, x_n)$ che vengono matematicamente combinate per ottenere un output o un vettore di output y. La rete di questo sistema è composta da uno strato di input e uno di output. Ogni neurone effettua il seguente calcolo:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.1}$$

I valori di x vengono combinati attraverso una somma pesata dove w_i per $i = 1,2, \ldots, n$ sono i pesi associati alle n variabili di input, introducendo un valore costante detto *bias* (b) ed infine viene applicata una funzione di attivazione f (viene spiegata meglio nel capitolo seguente). Per casi più complessi, quali la gestione di relazioni non lineari bisogna introdurre uno o più strati intermedi, detti strati nascosti [9].

2.1.2 Funzione di attivazione

Una funzione di attivazione è una funzione matematica che trasforma l'input lineare di un neurone (somma pesata degli input più bias) in un output non lineare. Senza di essa, una rete neurale non riuscirebbe a modellare fenomeni complessi [10].

La funzione di attivazione più usata nel *deep learning,* soprattutto nei *layer* nascosti è la ReLU (*Rectified Linear Unit*). È una funzione molto semplice (figura 2.1):

$$ReLU(x) = max(0, x) \tag{2.2}$$

cioè:

- Se x>0 → restituisce x
- Se x≤0 → restituisce 0

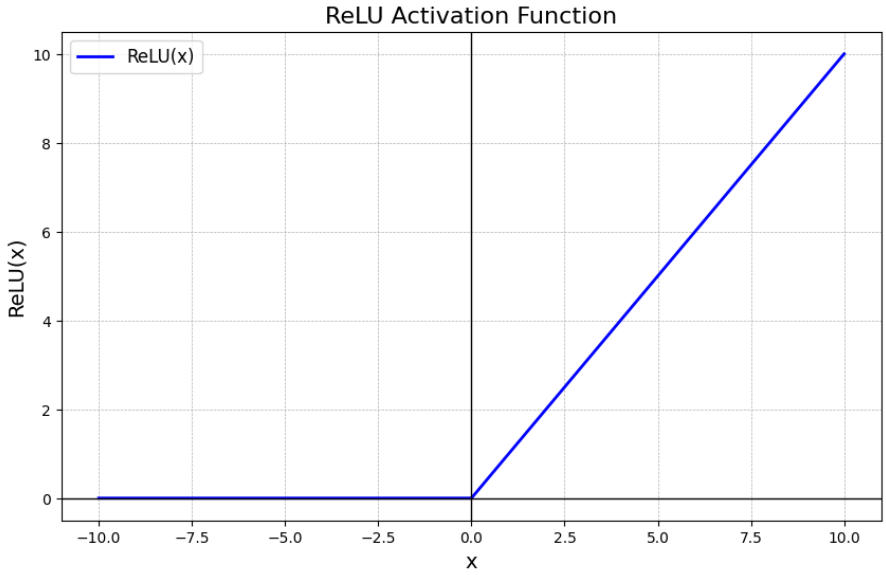


Figura 2.1: Funzione di attivazione ReLU.

È molto usata per la sua semplicità computazionale e permette di apprendere relazioni complesse. Il grosso problema di questa funzione di attivazione risiede nei cosiddetti neuroni morti: se un neurone entra nella regione in cui l'input è sempre x≤0, il suo output sarà sempre zero, e il peso non viene aggiornato e quel neurone diventa "morto". Per evitare questo, spesso si usa Leaky ReLU, che sostituisce lo zero con un valore negativo piccolo.

2.1.3 Reti Neurali *Multilayer*

Per apprendere correlazioni più complesse fra le variabili in ingresso, come nei casi studio di questo lavoro, bisogna aumentare il numero di *layer* nascosti e introdurre il caso di *Multy-Layers Perceptron* (MLP) o di *Deep Neural Network* (DNN) (figura 2.2) [9].

In questi modelli profondi oltre al numero di *layers*, vengono definiti il numero di neuroni che compongono ogni strato nascosto e influenzano la bontà del mio risultato finale.

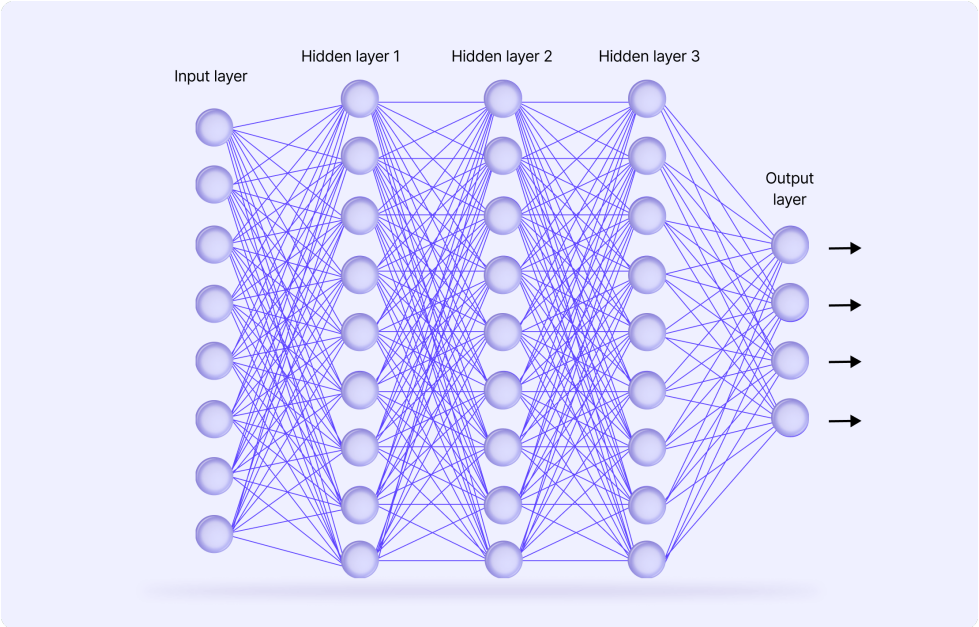


Figura 2.2: esempio di architettura di rete neurale con 3 *hidden layers*. Figura tratta da https://encord.com/blog/activation-functions-neural-networks/

Se la rete ha L strati, con $\mathbf{h}^{(l)}$ che rappresenta il vettore risultante nello strato l, possiamo scrivere:

$$\mathbf{h}^{(l)} = \mathrm{f}^{(l)}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2.3}$$

dove $W^{(l)}$ è la matrice dei pesi dello strato l, $\mathbf{b^{(l)}}$ è il vettore dei *bias* e $f^{(l)}(\cdot)$ è la funzione di attivazione del medesimo strato. L'output della rete dopo l'ultimo strato sarà quindi:

$$\boldsymbol{y} = f^{(L)}(\boldsymbol{W}^{(L)}\boldsymbol{h}^{(L-1)} + \boldsymbol{b}^{(L)}) \tag{2.4}$$

2.1.4 Standarizzazione

I dati nelle reti neurali prima di essere analizzati per trovare possibili correlazioni vengono standardizzati. Questo garantisce che tutte le *feature* abbiano lo stesso ordine di grandezza ed evita che *feature* con scale diverse dominino l'addestramento. Inoltre, facilita la convergenza dell'ottimizzatore: nel contesto del *deep learning*, un ottimizzatore aggiorna i pesi della rete per ridurre la funzione di perdita. Per un MLP, alcuni ottimizzatori comuni sono lo *Stocastic Gradient Descent* (SGD) e Adam, il primo funziona con aggiornamenti regolari seguendo il gradiente negativo della funzione di perdita (*loss*). Il secondo combina SGD con *momentum* adattivo: il *momentum* aggiunge inerzia all'aggiornamento dei pesi, ovvero limita il movimento fisico: se un peso si sta già muovendo in una direzione, continuerà in quella direzione. L'Adam è il più usato per MLP grazie alla sua efficienza e convergenza rapida.

La standarizzazione è un metodo di *preprocessing* che trasforma le variabili in modo che abbiano media zero e deviazione standard unitaria [11]. L'equazione che governa tale calcolo è la seguente:

$$X' = \frac{X-\mu}{\sigma} \tag{2.5}$$

dove X è il valore originale, μ è la media così calcolata:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.6}$$

dove:

- x_i sono i valori della variabile originale
- n è il numero totale di osservazioni

σ rappresenta la deviazione standard:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}} \tag{2.7}$$

2.1.5 Fase di addestramento e ottimizzatore

L'addestramento di una rete neurale ha inizio con l'inizializzazione in modo casuale dei pesi e dei *bias*, poi si prosegue con l'inizio della fase di *feed - forward* in cui i dati in ingresso al modello vengono trasmessi dallo strato di input ai successivi strati nascosti fino ad arrivare allo strato di output. Per ogni passo i neuroni di uno strato eseguono una somma pesata degli input ricevuti, li elaborano con una funzione di attivazione, ed inviano il risultato allo strato successivo. Alla fine, la rete produce una previsione basata sui pesi attuali e viene valutata la funzione di perdita o errore, che misura la differenza tra l'output previsto dalla rete e il valore reale desiderato. La funzione di perdita (*loss*) che è stata utilizzata è il *Mean Squared Error* (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.8}$$

dove:

- N: numero totale di campioni
- y_i valore reale (ground truth) del campione i
- $\hat{y}_i$ valore predetto dal modello per il campione i

Una volta calcolato l'errore, si passa alla fase di *back propagation*, dove l'errore viene trasmesso all'indietro attraverso la rete, aggiornando i pesi in modo da ridurre l'errore per le

future previsioni. Durante la fase di *back propagation*, viene calcolata la derivata della funzione di perdita rispetto ai pesi della rete. Questo permette di determinare sia la direzione sia l'entità della modifica necessaria per ciascun peso al fine di ridurre l'errore. Il calcolo sfrutta la regola della catena del calcolo differenziale e consente di aggiornare i pesi procedendo dall'output verso l'input, uno strato alla volta. Nel lavoro sviluppato in questa Tesi, l'aggiornamento dei pesi è stato gestito utilizzando Adam, un algoritmo ottimizzatore avanzato. Adam è dotato di un meccanismo di adattamento automatico del *learning rate* per ciascun parametro, favorendo così una maggiore stabilità e una più rapida convergenza del processo di addestramento [11]. L'intero processo di aggiornamento dei pesi viene infine ripetuto per molte epoche (iterazioni sull'intero set di dati).

2.1.6 Numero di epoche

Un'epoca rappresenta un passaggio completo su tutto il *dataset* di addestramento. Durante un'epoca, il modello legge i dati a *batch*, poi aggiorna i pesi attraverso *backpropagation* e migliora (idealmente) le sue prestazioni sull'obiettivo (es. MSE). Non vi è una regola fissa per determinare il numero di epoche necessarie a priori; tuttavia, bisogna stare attenti ad avere troppe epoche (*overfitting*), cosa che indica che il modello sta memorizzando i dati invece di generalizzare su nuovi input o ad avere troppe poche epoche (*underfitting*), nel qual caso il modello non apprende abbastanza relazioni da poter generare una buona previsione. Nel modello implementato nell'ambito di questo lavoro viene usata la tecnica dell'*Early Stopping* per via del suo basso costo computazionale [12] [4]. È una tecnica automatica per fermare l'addestramento quando la prestazione non migliora più sul *validation set,* cioè quella parte dei dati totali (10-20 %) che serve per valutare il modello durante l'addestramento, ma senza aggiornare i pesi. Il suo funzionamento è molto semplice quando questa metrica smette di migliorare per un certo numero di epoche (ad esempio 5), l'addestramento viene interrotto e i pesi vengono riportati al miglior modello osservato fino a quel momento. Questa tecnica permette di evitare l'*overfitting* ottenendo un modello che generalizza meglio sui dati nuovi (figura 2.3).

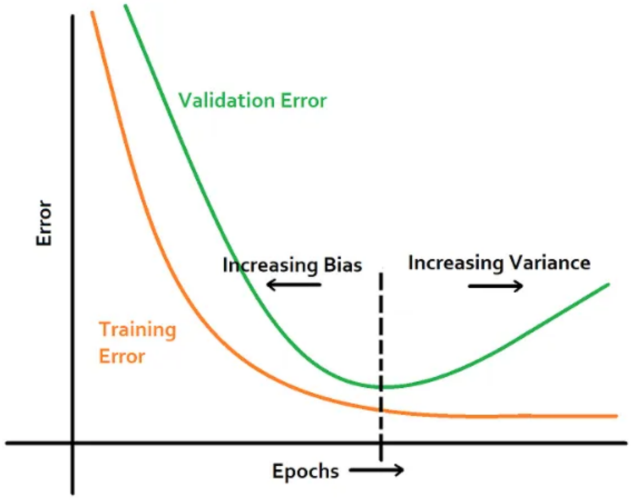


Figura 2.3: Tecnica di validazione *Early Stopping*. Figura tratta da https://medium.com/@rahuljain13101999/why-early-stopping-works-as-regularization-b9f0a6c277

2.2. Architettura RNN

Le Reti Neurali Ricorrenti (RNN) rappresentano un'evoluzione delle classiche reti *feed-forward* (come le MLP), progettate per lavorare su dati sequenziali o temporali, dove l'informazione passata è rilevante per il contesto attuale [13].

A differenza delle MLP, le RNN mantengono uno stato interno, detto stato nascosto, che viene aggiornato a ogni passo temporale. Questo meccanismo consente alla rete di memorizzare informazioni precedenti e usarle per generare output futuri.

L'architettura base di una RNN si può descrivere in questo modo:

$$\boldsymbol{h}_t = \tanh\left(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}\right) \tag{2.9}$$

$$\boldsymbol{y}_t = \boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{c} \tag{2.10}$$

dove:

- x_t è l'input al tempo t
- h_t e h_{t-1} sono rispettivamente lo stato nascosto e lo stato nascosto al tempo precedente

- W_x e W_h sono le matrici dei pesi per input e stato precedente
- W_y matrice dei pesi di output
- b, c bias
- y_t è l'output.

Nonostante la loro struttura dinamica, le RNN tradizionali soffrono di: decadimento/esplosione del gradiente, memoria a breve termine limitata, difficoltà nell' apprendere dipendenze lontane nel tempo.

2.3. Architettura generale *transformer*

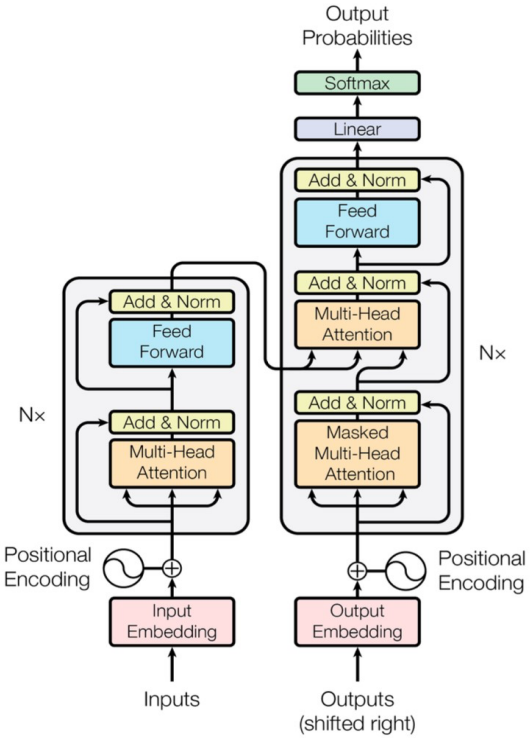


Figura 2.4: Architettura generale *transformer* utilizzata nei contesti NLP. Figura tratta da [14] e riprodotta con modifiche.

I *transformer* sono stati progettati per risolvere i problemi incontrati dalle architetture RNN (reti neurali ricorrenti), quali la parallelizzazione e la gestione delle dipendenze a lungo termine. Essi rappresentano una delle architetture di *deep learning* più rivoluzionarie degli

ultimi anni (Figura 2.4). Introdotti per la prima volta da Vaswani (2017) per compiti di traduzione automatica, sono stati rapidamente adottati in numerosi ambiti grazie alla loro capacità di modellare dipendenze a lungo termine in sequenze, senza l'uso di strutture ricorrenti. Il cuore dell'architettura è costituito dal meccanismo di *self-attention*, che consente di confrontare ogni elemento della sequenza con tutti gli altri, pesandone l'importanza relativa [14]. Queste architetture possiedono una potenzialità maggiore delle RNN come dimostrato in studi comparativi [15], ma presentano limiti di complessità rendendo difficile la loro applicazione in ambiti differenti da quello NLP (*natural language processing*). Infatti, la *self-attention* ha una complessità computazionale quadratica rispetto alla lunghezza dell'input O (L^2) [16], questo rende difficile l'applicazione a sequenze temporali molto lunghe. Per questa ragione, gran parte della ricerca recente sui *transformer* si concentra sullo sviluppo di approssimazioni del meccanismo di attenzione per ridurre il costo computazionale [17]. I *transformer* utilizzano un'architettura *encoder-decoder*: l'*encoder* processa la sequenza storica, arricchita con codifiche posizionali per mantenere l'ordine temporale. Il *decoder* invece genera le previsioni future basandosi sull'output dell'*encoder* e su eventuali input noti futuri. (figura 2.5)

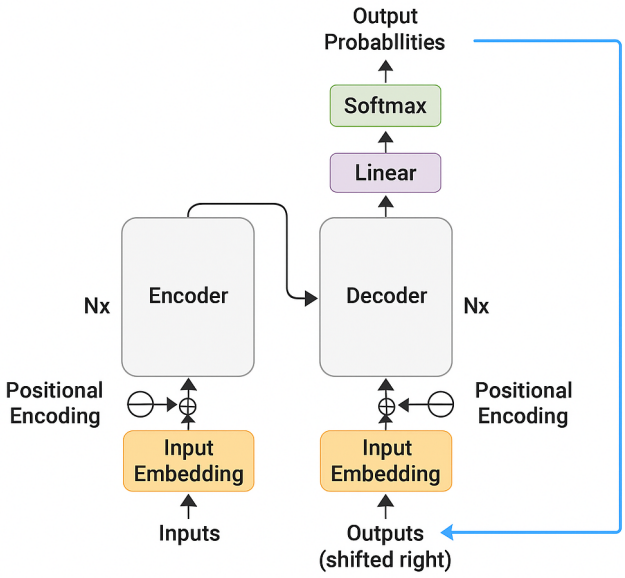


Figura 2.5: Architettura generale *Encoder-Decoder* semplificata.

La prima parte del processo di NLP consiste nella *tokenizzazione*, ossia la suddivisione della frase in unità più piccole, chiamate *token*, che possono essere parole oppure sotto-parole della

frase di partenza. Per elaborare le parole bisogna eseguire una serie di operazione numeriche; quindi, è nata la necessità di eseguire una conversione tra quello che è lo spazio delle parole e quello dei numeri. Vengono dunque utilizzati gli *embedding*: questi trasformano i *token* in vettori numerici, preservando e codificando informazioni semantiche e contestuali. l principale obiettivo degli *embedding* è quello di rappresentare il significato e le relazioni tra parole all'interno di uno spazio continuo multidimensionale. In pratica, ciascun *token* viene trasformato in un vettore posizionato in uno spazio tale che la distanza tra i punti rifletta la loro somiglianza semantica [18]. Tuttavia, poiché i modelli *transformer* elaborano tutte le parole simultaneamente, hanno bisogno di un modo per riconoscere l'ordine delle parole. Questo viene ottenuto tramite il *positional encoding*, che assegna codifiche sinusoidali alla posizione delle parole, garantendo che il modello comprenda l'ordine della sequenza [19].

Con il passaggio attraverso i vari livelli dell'architettura, queste rappresentazioni vettoriali evolvono, consentendo al modello di costruire una comprensione sempre più ricca e contestualizzata degli elementi in input.

Questa trasformazione progressiva è fondamentale per la creazione di rappresentazioni di ordine superiore, che emergono man mano che l'informazione viene raffinata. Una volta generati, gli *embedding* vengono trasmessi ai *transformer blocks*, i moduli centrali del modello, dove hanno luogo le principali operazioni di elaborazione.

Ogni *transformer block* è composto da tre componenti essenziali: il meccanismo di attenzione, che permette ai *token* di "guardare" gli altri *token*, scambiando informazioni contestuali. Poi continua con *add+norm*; con *add* (connessione residuale) l'output della sotto-struttura (es. attenzione) viene sommato all'input originale della stessa. Questo si chiama *residual connection* e aiuta a preservare il gradiente durante il *backpropagation*, evitando la degradazione del segnale nei modelli profondi. Con *norm* (Layer Normalization) dopo la somma, si applica la normalizzazione dei valori lungo la dimensione delle *feature*, mantenendo inalterate le informazioni posizionali, questo stabilizza e velocizza l'apprendimento.

Infine, è presente una rete neurale *feed-forward* (MLP), la cui funzione è quella di elaborare in modo indipendente ciascun *token*, migliorandone la rappresentazione locale [14]. Mentre l'attenzione serve ad arricchire ogni *token* con informazioni provenienti da altri punti della sequenza, l'MLP interviene per raffinare individualmente tali rappresentazioni.

Dopo essere stati processati dagli strati del *transformer*, i dati raggiungono il *decoder*, la parte del modello responsabile della generazione dell'output. Il *decoder* riceve due input principali: le rappresentazioni prodotte dall'*encoder* e gli *embedding* della sequenza generata fino a quel momento. Per assicurare che la generazione avvenga in modo autoregressivo, viene utilizzato il meccanismo di *masked self-attention*, una variante del meccanismo standard che impedisce

al modello di accedere ai *token* futuri nella sequenza [14]. In concreto, per ciascuna posizione, il modello può consultare solo i *token* precedenti, generando così il testo da sinistra verso destra, un elemento essenziale per preservare la coerenza grammaticale e semantica. Questo mascheramento si ottiene applicando una maschera triangolare alla matrice di attenzione, che oscura le posizioni future.

Segue quindi uno strato di *cross-attention*, nel quale il *decoder* integra l'informazione generata fino a quel punto con quella proveniente dall'*encoder*, arricchendo ulteriormente il contesto disponibile. Dopo un ulteriore passaggio in un MLP, il risultato finale viene proiettato in uno spazio vocabolario attraverso uno strato lineare, cui segue una funzione *softmax* che calcola la distribuzione di probabilità sui possibili *token* successivi.

A ogni step, viene scelto il *token* con la massima probabilità, che viene poi aggiunto alla sequenza generata. Si procede iterativamente fino ad ottenere l'intera frase o sequenza finale desiderata.

2.3.1 Analogia con l'utilizzo per le serie temporali

L'architettura *transformer*, inizialmente sviluppata per il *Natural Language Processing* (NLP), ha trovato ampie applicazioni anche nella previsione di serie temporali multivariate [20]. Sebbene i domini di utilizzo siano diversi, entrambi condividono lo stesso obiettivo: individuare correlazioni e dipendenze significative tra gli elementi della sequenza.

Nel contesto delle serie temporali, ciascun istante viene trattato come un *token*, il cui vettore denso di caratteristiche o osservazioni multivariate (*embedding*) rappresenta lo stato del sistema in quel momento. Il meccanismo di *self-attention* consente al modello di determinare quali istanti passati sono più rilevanti per una predizione accurata del futuro [21]. Analogamente all'NLP, dove l'*embedding* delle parole cattura relazioni semantiche, nel dominio temporale tali rappresentazioni dense codificano le relazioni tra variabili e il loro andamento nel tempo.

Un'ulteriore analogia risiede nell'utilizzo delle codifiche posizionali: nel linguaggio naturale esse indicano l'ordine delle parole nella frase, mentre nelle serie temporali forniscono al modello informazioni sulla sequenza cronologica dei dati, elemento cruciale per la previsione [14]. Inoltre, i *transformer* impiegano strati *feed-forward* dopo l'attenzione, e il meccanismo di *multi-head attention* consente di catturare in parallelo diverse dipendenze temporali e inter-variabili [21] [22]. Ad esempio, nel presente lavoro, alcune teste di attenzione potrebbero apprendere le relazioni tra il flusso di alimentazione e la conversione, mentre altre potrebbero specializzarsi nelle interazioni tra temperatura COT e conversione.

Un importante vantaggio dei *transformer* è la possibilità di eseguire il *training* in parallelo, elaborando intere sequenze simultaneamente, a differenza delle reti ricorrenti che operano in maniera sequenziale. Questa proprietà li rende particolarmente adatti a *dataset* di grandi dimensioni, come quello utilizzato in questo studio [22].

Tuttavia, un limite rispetto all'NLP riguarda la natura meno strutturata dei dati temporali: mentre il linguaggio naturale segue regole sintattiche ben definite, le serie temporali sono spesso affette da rumore, irregolarità e dati mancanti. Nonostante ciò, la flessibilità e la capacità di astrazione dei *transformer* dimostrano la loro efficacia anche in ambiti distanti da quello per cui furono originariamente progettati.

2.3.2 La tecnica delle *sliding windows*

Nel modello *transformer* utilizzato è stato implementata la tecnica delle *sliding windows* per suddividere la serie temporale in segmenti di lunghezza fissa, dividendo quindi l'intero *dataset* in sotto finestre temporali (Figura 2.6) [23]. Questo meccanismo è fondamentale per adattare modelli *transformer* (originariamente pensati per NLP) all'ambito delle serie temporali. Il meccanismo delle *sliding windows* suddivide la serie temporale in sequenze strutturate, permettendo di generare finestre temporali utilizzabili nei processi di addestramento, validazione e test. Questa strategia consente al modello di cogliere *pattern* dinamici significativi, mantenendo al contempo un'efficienza computazionale ottimale.

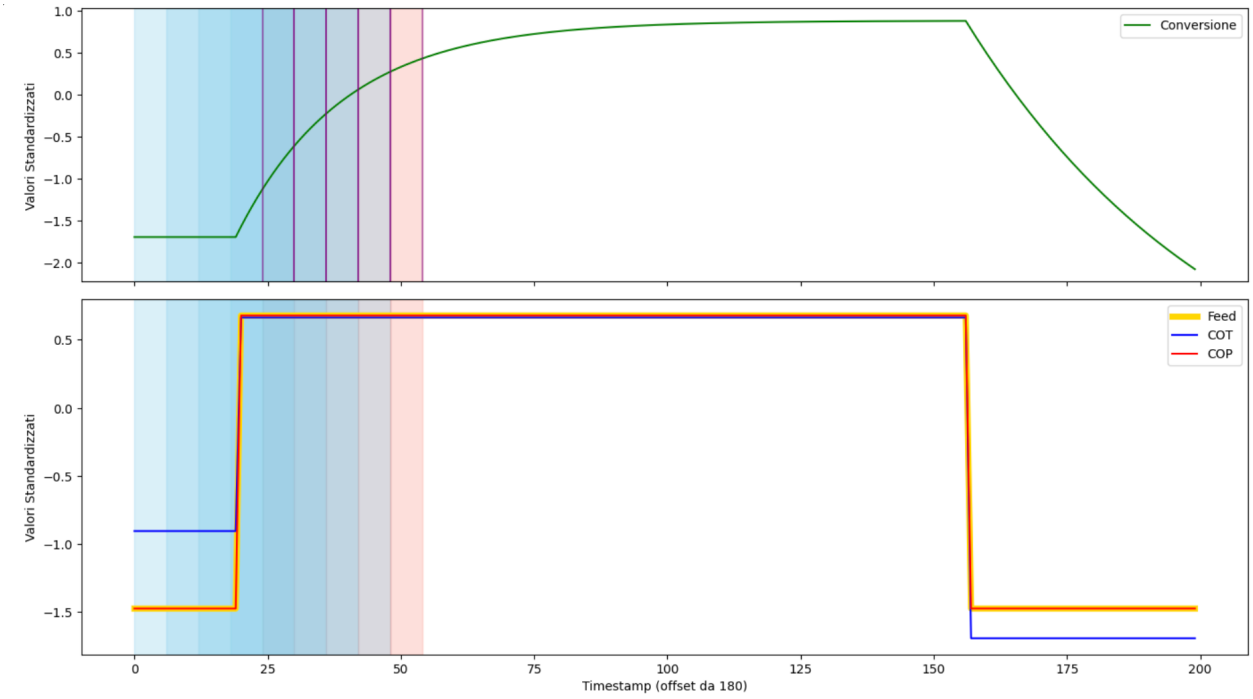


Figura 2.6: Visualizzazione delle finestre temporali sui minuti da 180-380 del *dataset* di *training*

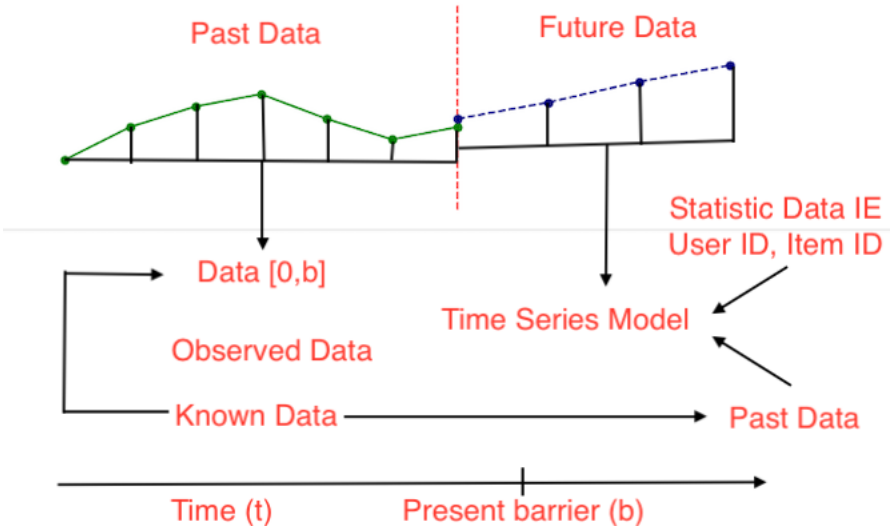


Figura 2.7: *Workflow* delle previsioni delle serie temporali per un *Transformer*. Schema rielaborato da https://www.geeksforgeeks.org/deep-learning/transformer-for-time-series-forecasting/

Il metodo delle *sliding windows* consiste nel creare finestre mobili di dati passati, che vengono utilizzate per prevedere un certo numero di valori futuri, chiamati *Horizon* (figura 2.7). Una volta divisa la serie temporale multivariata in una iniziale serie temporale dedicata al *training*, la adiacente e successiva serie dedicata alla *validation* e la finale per la fase di *testing*, si procede alla generazione delle finestre temporali. Se si considera una serie temporale $X = [x_1, x_2, \ldots, x_T]$, con un *context length* L_c e un *prediction length* L_p, il modello estrae sotto-sequenze della forma $[x_t, x_{t+1}, \ldots, x_{t+Lc-1}]$ come input e $[x_{t+Lc}, x_{t+Lc+1}, \ldots, x_{t+Lc+Lp-1}]$ come *target* (Figura 2.8).

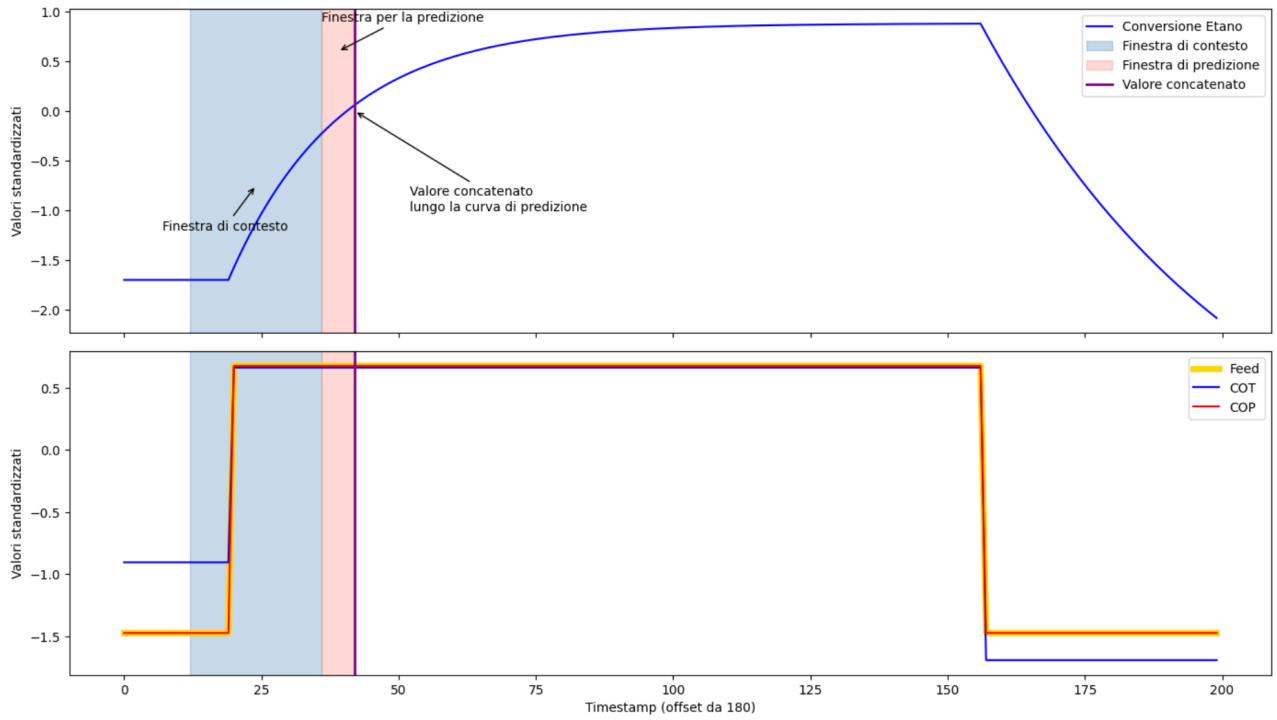


Figura 2.8: Visualizzazione di una finestra temporale del *dataset* di *training*.

Questo sta a significare che l'*encoder* riceverà la sequenza caratterizzata dalla lunghezza *context length* mentre il *decoder* riceverà la sequenza di lunghezza *prediction length*. Considerando come esempio il *dataset* di *training*, la finestra viene fatta avanzare lungo la serie temporale utilizzando un parametro chiamato *stride*, che definisce di quanti passi la finestra si sposta in avanti per generare una nuova finestra di *training*. Uno *stride* piccolo come un solo un istante temporale, comporta una maggiore sovrapposizione tra le finestre consecutive, aumentando il numero di esempi di *training* disponibili, mentre uno *stride* più grande, per esempio impostando *stride* uguale all'orizzonte di predizione *horizon,* si riduce il numero di finestre generate, diminuendo la ridondanza ma rischiando di perdere informazioni rilevanti. Nella fase di *test*, il modello *transformer* applica i pesi ottimizzati durante l'addestramento per analizzare le finestre temporali del *dataset* di *test*, che contengono informazioni mai incontrate in precedenza. Questi pesi, affinati attraverso la minimizzazione dell'errore sulle sequenze di *training*, restano invariati: il modello non modifica più i propri parametri, ma si limita a eseguire previsioni sulle nuove sequenze. In questa fase il modello non impara, ma si limita a svolgere il compito di predizione sui nuovi dati passati al modello. Nel nostro caso studio basato su dati sintetici, suddiviso nei due sotto casi del forno ad etano e dello *splitter C3*, attraverso un *tuning* del modello sono stati identificati in modo preciso i parametri *context* e *predictive length* per le due applicazioni.

2.3.3 Struttura *Encoder - Decoder*

L'architettura *encoder-decoder* implementata in questo modello *transformer* (Figura 2.9) ha un flusso dei dati che segue un percorso ben preciso ed elaborato. Si incomincia con la prima fase di generazione degli *embedding* relativi all'*input*, dove le finestre temporali multivariate in ingresso vengono proiettate in uno spazio latente di una specifica dimensione (ottimizzata durante la fase di *tuning* del modello) mediante una trasformazione lineare. Una volta ottenuto l'*embedding*, si procede con il *positional encoding,* garantendo che il modello comprenda l'ordine della sequenza. I dati così elaborati entrano nel *transfer block,* fluiscono attraverso l'*encoder*, che è composto da una sequenza di blocchi *transformer* che processano iterativamente l'informazione. Ogni blocco *transformer* contiene il meccanismo della *self-attention* che consente a ciascun punto della sequenza di interagire con tutti gli altri (attraverso l'utilizzo delle matrici di *Query*, *Key* e *Value*), cercando di trovare le dipendenze tra le variabili. Il meccanismo divide la rappresentazione in teste multiple (di solito da 4 a 16) che operano in parallelo (MHA), ognuna avente all'interno il meccanismo della *self-attention*, permettendo al modello di focalizzarsi su diverse relazioni contemporaneamente. Questo processo è implementato attraverso le proiezioni delle matrici di Q e K, seguite dal calcolo dei punteggi di attenzione (*matmul*) e poi normalizzati (*scale*) e trasformati in pesi da *softmax*. Infine, con *matmul* applica questi pesi di attenzione ai valori della matrice *Value*, creando l'output per questa testa. Gli output di tutte le teste vengono concatenati e poi passati attraverso un'altra *Linear Transformation* (LT). Dopo la *self-attention*, i dati attraversano un *feed-forward network* (MLP) con attivazione *GELU* che espande e poi ricomprime le informazioni con lo scopo di aumentarne la rappresentatività, mentre connessioni residue e normalizzazione a strati sono necessari per avere un *training* con una certa stabilità. Il passaggio di informazioni dall'*encoder* al *decoder,* la cosiddetta *cross-attention,* è importante dato che va a fornire il contesto per la generazione delle previsioni future. Il *decoder* prende in input una sequenza di *timestep*, ottenuta dagli ultimi elementi dell'output dell'*encoder*. Questa scelta garantisce continuità temporale durante la generazione. La lunghezza di questa sequenza iniziale è un iperparametro che viene ottimizzato durante la fase di *tuning*. All'interno di ogni strato del *decoder*, il flusso dei dati si arricchisce attraverso tre componenti fondamentali: prima una *masked self attention* che elabora la sequenza *decoder* finora generata, poi una *cross-attention* che integra l'informazione dell'*encoder* permettendo al *decoder* di accedere all'intero contesto storico, e infine un *feed-forward network* che espande ulteriormente la capacità rappresentativa. Ognuno di questi blocchi è accompagnato da meccanismi di normalizzazione e connessioni residue che

agevolano l'ottimizzazione e migliorano la stabilità durante l'addestramento. La lunghezza della sequenza nel *decoder* rimane costante grazie a una finestra temporale che scorre a ogni passo di generazione. Infine, l'output prodotto dal *decoder* viene passato attraverso un livello lineare che lo trasforma in previsioni per la variabile target. Queste previsioni vengono accumulate passo dopo passo per costruire l'intera sequenza prevista e, successivamente, vengono denormalizzate per riportarle nelle unità originali.

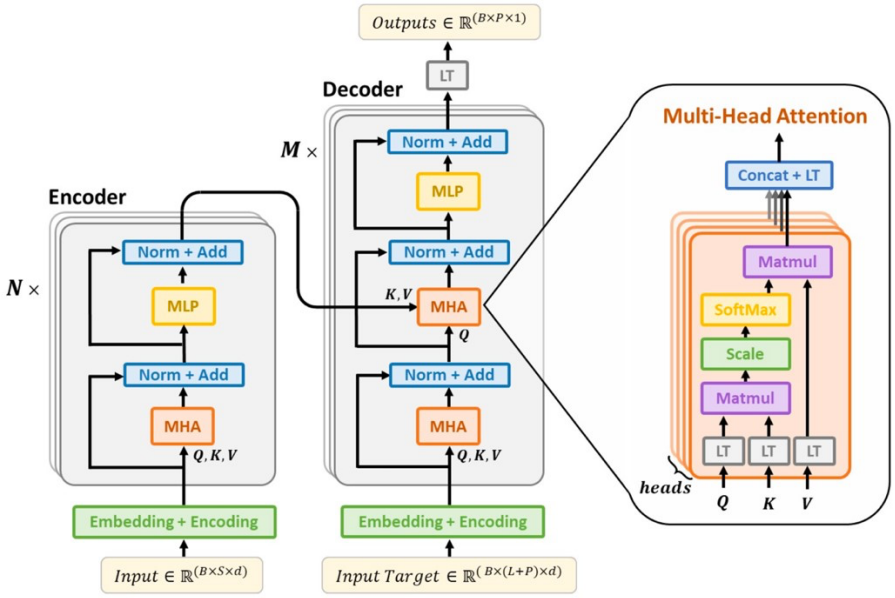


Figura 2.9: Rappresentazione della struttura *encoder - decoder* dove B è il *batch size*, S è la lunghezza di contesto, P è la lunghezza della predizione ed L è la lunghezza della sequenza iniziale del *decoder*. Figura tratta da [24] e riprodotta con modifiche.

Nei capitoli a seguire verrà studiato il flusso dei dati attraverso i meccanismi principali della struttura, *step* successivi alla creazione delle finestre temporali ed alla divisione in strutture *batch*. Verranno osservati e analizzati nel dettaglio i processi di generazione degli *embedding* ed il meccanismo della *self-attention*, processi chiave di questa architettura.

2.3.4. La suddivisione in *batch* e la generazione degli *embedding*

Dopo l'applicazione del meccanismo delle *sliding window*, ciascuna finestra acquisita corrisponde a una sequenza temporale di lunghezza prefissata, contenente i valori delle diverse

variabili osservate nel tempo. L'insieme di tutte le finestre può essere interpretato come una matrice tridimensionale, in cui:

- la prima dimensione rappresenta il numero totale di finestre estratte (ossia il numero di sequenze)
- la seconda corrisponde alla lunghezza temporale di ciascuna finestra (numero di *timestep*)
- la terza indica il numero di variabili (o *feature*) osservate in ogni istante.

Le dimensioni dei dati dopo lo *sliding windows* sono quindi le seguenti:

[Numero di finestre, Lunghezza finestra, Numero di variabili]

Una volta finita la generazione di tutte le finestre temporali, avremo che queste sono divise in tre grandi *batch*, uno comprendente i dati di *training* (il primo 60% del *dataset* completo nel nostro caso), uno per il *validation* (il successivo 20%) ed infine l'ultimo per il *testing* (l'ultimo 20%). Il passo successivo consiste nel suddividere le finestre temporali, raccolte all'interno dei tre *macrobatch*, in *batch* di dimensioni più ridotte (Figura 2.10). Questa suddivisione ha l'obiettivo di ottimizzare l'efficienza computazionale durante la fase di addestramento del modello. In questo modo, si ottiene una sequenza di *mini-batch* che appartengono ai tre insiemi distinti iniziali. Durante ogni iterazione di *training*, l'architettura *transformer* processerà un *batch* avente la seguente struttura dimensionale:

[*Batch size*, Lunghezza finestra, Numero di variabili]

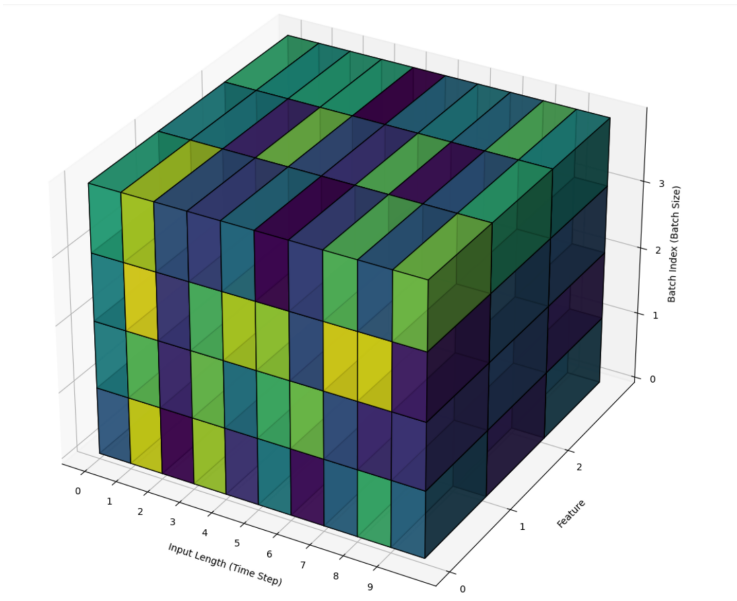


Figura 2.10: Visualizzazione delle dimensioni dopo la creazione delle finestre temporali ed il raggruppamento di queste in *batch*.

L'adozione del *batching* durante l'addestramento del modello consente un'elaborazione efficiente in parallelo e contribuisce a stabilizzare il processo di ottimizzazione. Infatti, sia il calcolo delle derivate che l'aggiornamento dei pesi vengono eseguiti su una media di più esempi simultaneamente, riducendo la variabilità nei gradienti e migliorando la convergenza. Inoltre, questo approccio risulta particolarmente vantaggioso nei contesti industriali, dove i dati possono essere affetti da rumore: l'aggregazione su più istanze rende il modello più robusto e meno sensibile a disturbi locali. Una volta suddivisi in *batch*, i dati all'interno di ciascuna finestra temporale vengono proiettati in uno spazio vettoriale di dimensione fissa mediante uno strato di *embedding*. Questo passaggio è fondamentale per l'elaborazione da parte dell'architettura *transformer*, la quale richiede input in forma vettoriale omogenea. Per ogni istante temporale contenuto nella finestra, l'intero vettore di input (che comprende tutte le variabili osservate in quell'istante) viene trasformato in un singolo *embedding* vettoriale attraverso una trasformazione lineare. In altre parole, tutte le misurazioni simultanee di un *time-stamp* vengono aggregate in una rappresentazione congiunta, anziché essere codificate individualmente. Questo approccio consente al modello di catturare con efficacia le relazioni tra variabili coesistenti, come le interazioni tra temperatura, pressione o portata, facilitando l'individuazione di *pattern* complessi e correlazioni rilevanti. Tuttavia, una possibile limitazione di questa strategia è la perdita di specificità: alcune caratteristiche uniche delle

singole variabili potrebbero venire attenuate, in particolare se certe variabili hanno un'influenza più significativa di altre sul comportamento del sistema. Nonostante ciò, questo metodo rappresenta un buon compromesso tra espressività e semplicità del modello, risultando efficace per una vasta gamma di problemi industriali reali, soprattutto in presenza di multi variabilità e interazioni tra segnali.

2.3.5. Il meccanismo di attenzione

Ogni punto della serie temporale, dopo essere stato trasformato in un *embedding* numerico, entra nel *transfer block* dove viene elaborato attraverso il meccanismo di attenzione (Figura 2.11). Per determinare quali informazioni all'interno di una sequenza temporale debbano essere valorizzate maggiormente, il modello *transformer* utilizza un meccanismo noto come attenzione. Ogni elemento della sequenza viene trasformato in tre vettori distinti: *query* (Q), *key* (K) e *value* (V). Applicando questa trasformazione a tutta la sequenza, si ottengono tre matrici omonime *query, key* e *value*, su cui si fonda il calcolo dell'attenzione. Nel contesto dell'analisi di serie temporali, come nel caso in esame, non tutte le informazioni passate hanno la stessa rilevanza per la previsione di un valore futuro. È dunque fondamentale individuare, per ogni passo temporale, quali osservazioni precedenti contengano le informazioni più significative. Il meccanismo di attenzione è progettato esattamente per questo scopo: consente al modello di valutare dinamicamente il contributo di ciascun punto nella sequenza passata, in relazione al punto corrente. In particolare, ogni vettore *query* viene confrontato con tutte le *key* della sequenza tramite un prodotto scalato, il cui risultato viene normalizzato tramite una funzione *softmax*. Questo produce una distribuzione di pesi che quantifica quanto ogni osservazione precedente debba essere considerata, dando luogo a un filtro contestuale. Tali pesi vengono poi applicati ai rispettivi *value*, generando una rappresentazione ponderata che enfatizza i dati più rilevanti rispetto al contesto.

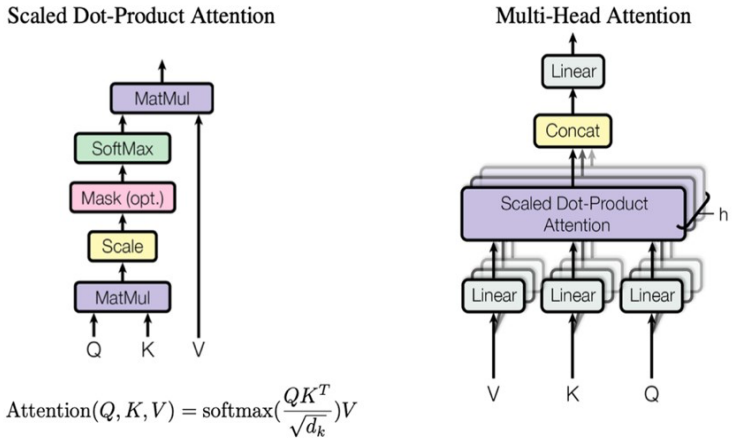


Figura 2.11: Rappresentazione del meccanismo di attenzione (a sinistra) e del meccanismo della *multi-head attention* (a destra). Figura tratta da [14] e riprodotta con modifiche.

Nel presente lavoro è stata adottata una *multi-head attention*, un'estensione del meccanismo appena descritto. In questo approccio, l'operazione di attenzione viene eseguita in parallelo su più teste indipendenti (h), ciascuna delle quali ha la possibilità di apprendere *pattern* e relazioni differenti tra i dati. Ogni testa lavora con una propria porzione della rappresentazione, ottenuta suddividendo lo spazio degli *embedding* lungo la dimensione dei canali.

I risultati delle varie teste vengono infine ricombinati: i singoli output vengono concatenati e successivamente proiettati attraverso uno strato lineare finale (Figura 2.11). Il risultato è un'unica rappresentazione vettoriale arricchita, che riflette molteplici prospettive sulle dipendenze all'interno della sequenza.

2.4. Software commerciale Honeywell

Nel caso studio si è voluto mettere in risalto le performance del *transformers* rispetto a quelle del software commerciale (Honeywell) normalmente utilizzato dall'azienda Alpha Process Control per l'identificazione e la predizione della dinamica del processo. Il software per l'identificazione e la previsione della risposta dinamica utilizza un modello *Finite Impulse Response* (FIR). Vengono calcolate le metriche di valutazione rispetto ai dati di sintesi per il caso *transformers* e per il modello commerciale, andando a confrontare i risultati. Inoltre, si mette in risalto la differenza di risultato graficamente.

2.4.1. Modello *Finite Impulse Response* (FIR)

I modelli lineari FIR (*Finite Impulse Response*) rappresentano un sistema lineare o debolmente non lineare tramite la sua risposta all'impulso, troncata dopo un certo orizzonte temporale. In altre parole, descrivono come l'uscita *y(t)* di un processo reagisce a un impulso unitario in ingresso (come se si aprisse e chiudesse velocemente una valvola) assumendo che dopo un tempo finito la risposta si annulli. Questi modelli sono concettualmente simili ai modelli a risposta gradino finita (figura 2.12)

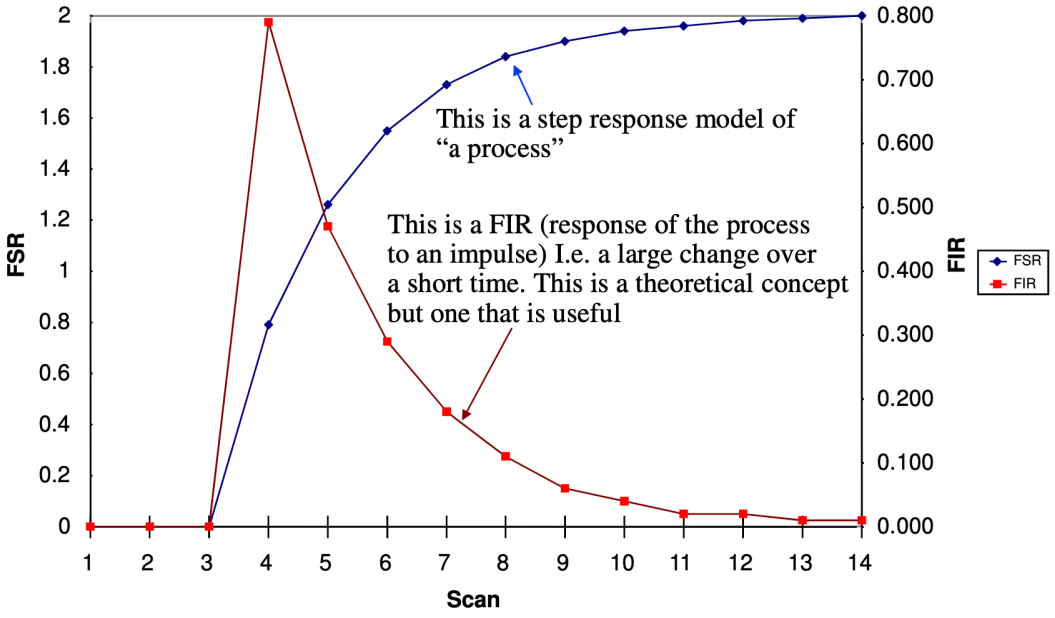


Figura 2.11: confronto risposta a gradini (FSR) con risposta ad impulso (FIR). Tratto da [25] e riprodotta con modifiche.

Dal punto di vista matematico, un modello FIR discreto esprime l'uscita come somma convolutiva finita degli ingressi passati. Nella sua forma posizionale (assoluta) possiamo scrivere ad esempio:

$$y(k) = b_0 u(k) + b_1 u(k-1) + \cdots + b_N u(k-N) + bias. \qquad (2.9)$$

Oppure si predice la variazione, con una forma incrementale (forma "in velocità") che elimina il *bias*:

$$\Delta y(k) = b_0 \Delta u(k) + b_1 \Delta u(k-1) + \cdots + b_N \Delta u(k-N) \qquad (2.10)$$

Bisogna evidenziare che sono semplicemente due rappresentazioni equivalenti del medesimo modello FIR di base.

I modelli FIR sono ampiamente utilizzati in identificazione di sistemi e in controllo predittivo. In ambito d'identificazione, costituiscono un approccio non parametrico (detto anche *structure free*): invece di assumere a priori un modello differenziale di un certo ordine (ad esempio primo o secondo ordine), si stima direttamente la sequenza di impulsi che descrive il sistema. In pratica, per ottenere un modello FIR di un processo si procede eseguendo test sperimentali: si applicano ingressi noti (come segnali a gradino come nel nostro caso, oppure impulsi brevi o sequenze pseudocasuali tipo PRBS) al sistema e si misura l'uscita. Attraverso metodi di regressione lineare o analisi di correlazione si estraggono quindi i coefficienti b_i della risposta impulsiva che meglio riproducono l'uscita misurata a partire dall'ingresso (minimizzando l'errore in media quadratica, ad esempio). Il documento originale sottolinea l'uso di *step*, *pulse* e *PRBS* come segnali di eccitazione e menziona l'importanza di tecniche come correlazione o trasformate per passare dai dati grezzi al modello (figura 2.12) [25]. Il modello FIR agisce come un predittore lineare del sistema, fornendo stime immediate dell'output futuro senza dover risolvere equazioni differenziali o integrare modelli complessi. Va sottolineato, tuttavia, che l'accuratezza delle predizioni FIR dipende dalla qualità dell'identificazione dei coefficienti e dalla validità delle ipotesi di linearità e tempo-invarianza nel periodo di predizione. In un contesto di tesi dedicato all'identificazione e predizione mediante *transformers*, i modelli FIR rappresentano dunque un termine di paragone fondamentale: essi costituiscono la base lineare semplice con cui confrontare le prestazioni di modelli più sofisticati.

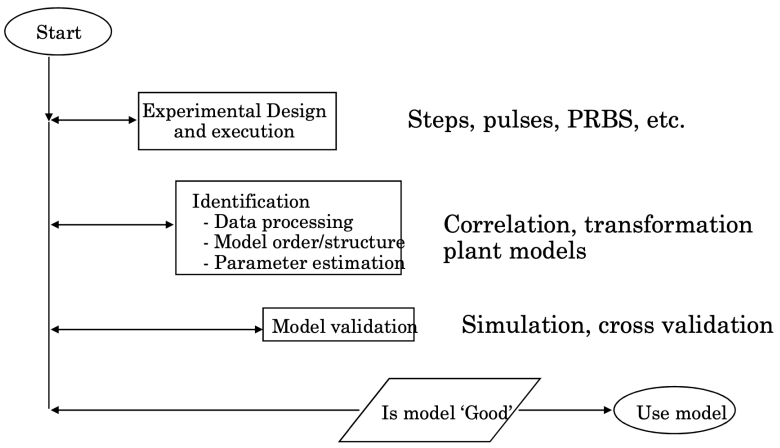


Figura 2.11: Workflow identificazione con modello FIR. Tratto da [25] e riprodotto con modifiche.

2.5. Metriche di valutazione

Per valutare l'efficacia e le prestazioni dei modelli predittivi utilizzati nei due casi studio del forno ad etano e del C3 *splitter* sono state utilizzate tre metriche: il coefficiente di determinazione R^2 nella sua formulazione classica, il *Mean Squared Error* (MSE) e il *Mean Absolute Error* (MAE) [26]. Il coefficiente di determinazione R^2 fornisce una misura della capacità del modello di spiegare la variabilità della variabile dipendente utilizzando l'intero *dataset* per l'addestramento. Il coefficiente di determinazione multiplo R^2 viene definito come:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.11}$$

dove y_i rappresenta il valore reale della i-esima osservazione, $\hat{y}_i$ è il valore predetto dal modello, $\bar{y}$ è la media dei valori osservati, n indica il numero totale delle osservazioni.

Per una valutazione più completa delle prestazioni sono state considerate anche due altre metriche di valutazione, chiamate errore quadratico medio (*Mean Squared Error*, MSE) ed errore assoluto medio (*Mean Absolute Error*, MAE). L'errore quadratico medio (MSE) è stato calcolato come:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.12}$$

L'errore assoluto medio (MAE) è stato determinato tramite la formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.13}$$

L'R^2 è una misura relativa della bontà del modello: più il suo valore è vicino a 1, maggiore è la capacità del modello di spiegare la variabilità dei dati. Valori prossimi a 0 indicano invece una scarsa capacità predittiva. Il *Mean Squared Error* (MSE), invece, va a valutare la dispersione dei residui elevando al quadrato le differenze tra valori reali e predetti. Questo è maggiormente sensibile agli errori più grandi. Al contrario, il *Mean Absolute Error* (MAE) fornisce una stima di più facile interpretazione della deviazione media tra le previsioni e i valori osservati, senza penalizzare eccessivamente gli *outlier*. Il confronto fra queste metriche ottenute nel caso del *transformers* e del modello commerciale ci permette di comparare e valutare le prestazioni dei modelli.

Capitolo 3

Casi studio e risultati

In questo capitolo vengono presentati e discussi i risultati ottenuti nel corso del lavoro di Tesi. L'analisi è suddivisa in due sezioni distinte, corrispondenti a due casi studio caratterizzati da crescente complessità nella struttura del sistema MISO (*Multiple Input, Single Output*). In particolare, si passa da un sistema con due variabili manipolate, relativo a uno *splitter* di una miscela propano-propilene (C3 *splitter*), a un sistema con tre variabili manipolate, rappresentato da un forno ad etano. Per questi casi studio sappiamo che ci troviamo di fronte ad un comportamento non lineare delle variabili manipolate rispetto alla variabile controllata. In entrambi i casi, il *dataset* utilizzato per identificare la dinamica del processo è composto da dati continui acquisiti con frequenza di un minuto.

Il primo caso studio riguarda l'identificazione e la previsione del comportamento dinamico di un C3 *splitter*, con l'obiettivo specifico di modellare l'andamento dell'impurezza di propano in testa alla colonna di separazione. Poiché l'impianto in esame non dispone di un analizzatore per la misura diretta di tale impurezza, ma sono note le condizioni operative del processo, è stato sviluppato un *dataset* sintetico tramite simulazioni effettuate con un simulatore di processo commerciale. Le variabili manipolate in ingresso sono state variate secondo una sequenza a gradini, al fine di stimolare adeguatamente la risposta del sistema.

La seconda parte dell'analisi è dedicata all'identificazione e previsione della dinamica di un forno ad etano, con l'obiettivo di costruire un modello capace di stimare la conversione dell'etano. In questo caso, si dispone di dati reali di processo acquisiti da sensori in condizioni operative standard. Tuttavia, la bassa variabilità dei dati in tali condizioni limita l'efficacia dell'identificazione. Inoltre, questi analizzatori subiscono fenomeni di sporcamento dovuto al coke e per questo motivo sono poco utilizzati nell'impianto in questione. Per ovviare a tale problema, si è fatto ricorso a un modello del forno basato su equazioni di bilancio, attraverso il quale sono stati generati dati con maggiore variabilità, nuovamente mediante variazioni a gradino delle variabili manipolate.

In entrambi i casi studio è stata utilizzata la stessa architettura di tipo *transformer*, opportunamente personalizzata nei suoi parametri principali per adattarsi alle specificità di

ciascun sistema. L'obiettivo è stato quello di massimizzare la capacità predittiva del modello sull'output di interesse.

3.1. Colonna di separazione propano-propilene

Come introdotto in precedenza, questa sezione dell'elaborato si concentrerà sulla colonna di separazione propano–propilene (*C3 splitter*), componente fondamentale nella fase di *post-cracking* dell'*ethylene unit.* Il *feed* della colonna arriva a monte dal *depropanizer* che divide i componenti idrocarburici C3, cioè propano e propilene dai C4+.

L'impurezza del propano nel prodotto di testa viene calcolata come frazione molare percentuale di propano. Per avere una migliore separazione e ridurre l'impurezza di propano nel prodotto in testa alla colonna, le due principali variabili da manipolare nella colonna sono il calore fornito al ribollitore (E101B) e il rapporto di riflusso in testa alla colonna. Il *range* di variazione di queste due variabili è stato scelto in base al valore di normale operatività dell'impianto. Per quanto riguarda il calore fornito al ribollitore, esso viene già fornito da impianto e viene calcolato come calore latente quindi come prodotto fra la portata di vapore FC101 e l'entalpia di vaporizzazione. Invece per quanto riguarda il riflusso, il valore iniziale da cui far partire le variazioni della variabile viene fornito ed è calcolato come rapporto fra il valore letto dal sensore FC112 diviso la somma delle portate del propilene FC103 e FC116 (figura 3.1).

Per sviluppare l'architettura *transformer* sono stati creati dati di processo vista la mancanza di un analizzatore, essenziale per registrare il valore dell'impurezza in testa alla colonna.

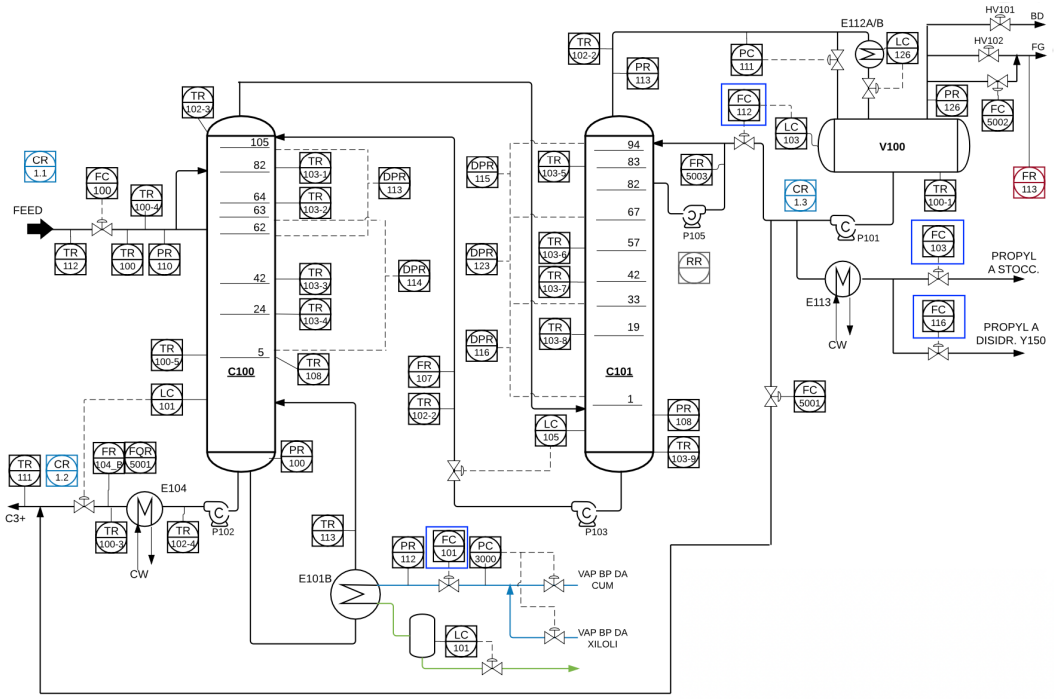


Figura 3.1: Schema d'impianto della colonna *splitter* C3 con focus su variabili manipolate utilizzate per trovare il calore al *reboiler* e il riflusso in testa alla colonna.

3.1.1. Creazione dati

Viene disegnato uno schema semplificato per il nostro caso studio dello *splitter* propano-propilene su un simulatore commerciale (figura 3.2). La frequenza del *dataset* creato per il sistema MISO 2x1 è di un dato al minuto, ciò è dovuto al fatto che il *transformers* per funzionare deve poter lavorare con dati dove il controllo richiede sensibilità a cambiamenti rapidi. I dati di sintesi per le variabili manipolate sono stati creati come *step test*, ricreando così la procedura identificativa usata in impianto per studiare la dinamica del processo (figura 3.3). Inserendo le condizioni operative e variando il valore di queste variabili manipolabili, il simulatore di processo fornisce come risultato i valori dell'impurezza statica, cioè raggiunto lo stazionario nel prodotto di testa (figura 3.4).

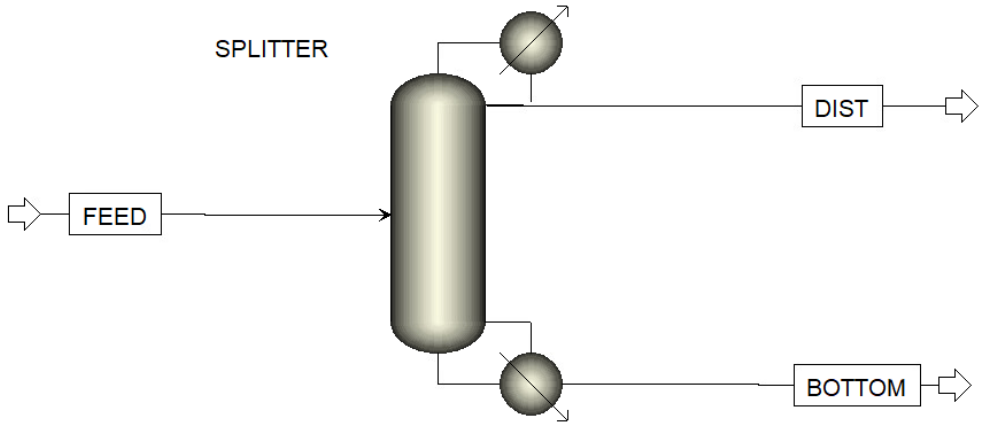


Figura 3.2: schema semplificato dello *splitter* C3

In un secondo momento a questi valori statici di frazione molare del propano viene inserita la dinamica del processo, assunta essere del 1 ordine con ritardo, con una costante di tempo che dipende dal riflusso, variando fra 150-180 minuti e il ritardo costante di 20 minuti, ottenendo così il *dataset* continuo dell'impurezza di propano in colonna da cui far partire le nostre analisi (figura 3.4). Per quanto riguarda il ritardo di processo, esso non viene considerato direttamente nella fase di identificazione, ossia i dati di ingresso e uscita utilizzati per addestrare il modello vengono trattati come se non vi fosse ritardo. Successivamente, una volta generate le previsioni, viene applicato uno *shift temporale* alle uscite stimate, in modo da riallinearle con i dati reali. Questa strategia è stata adottata dopo aver sperimentato diverse identificazioni con il ritardo incluso nei dati di *training*. Tali prove hanno mostrato che, nell'intorno dei primi 20 minuti di previsione, il modello tendeva a produrre oscillazioni e comportamenti non coerenti con la dinamica reale del processo. L'approccio scelto ossia ignorare il ritardo durante l'identificazione e introdurlo solo a posteriori sulle previsioni consente invece di ottenere curve più stabili e meglio allineate ai valori simulati, migliorando la qualità complessiva della previsione.

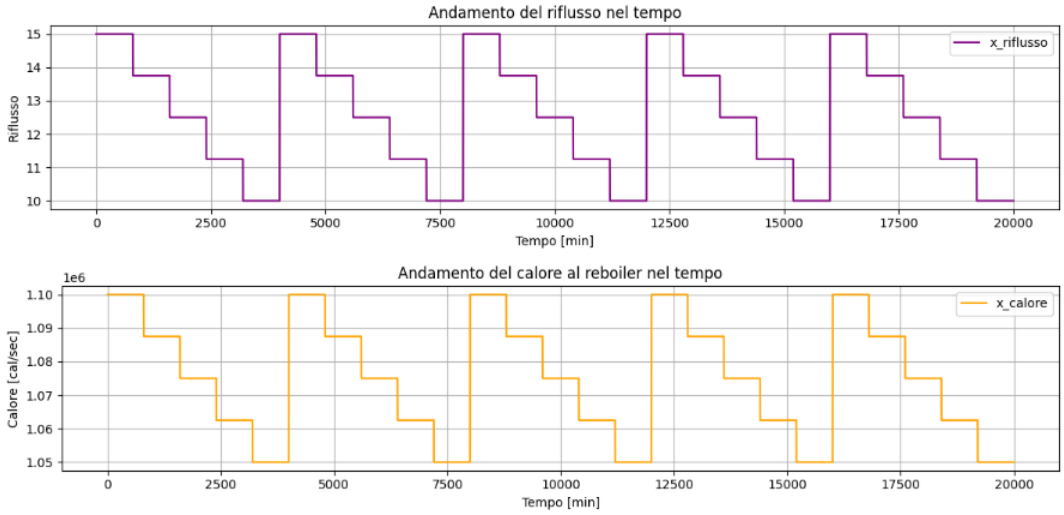


Figura 3.3: input manipolate a *step test*

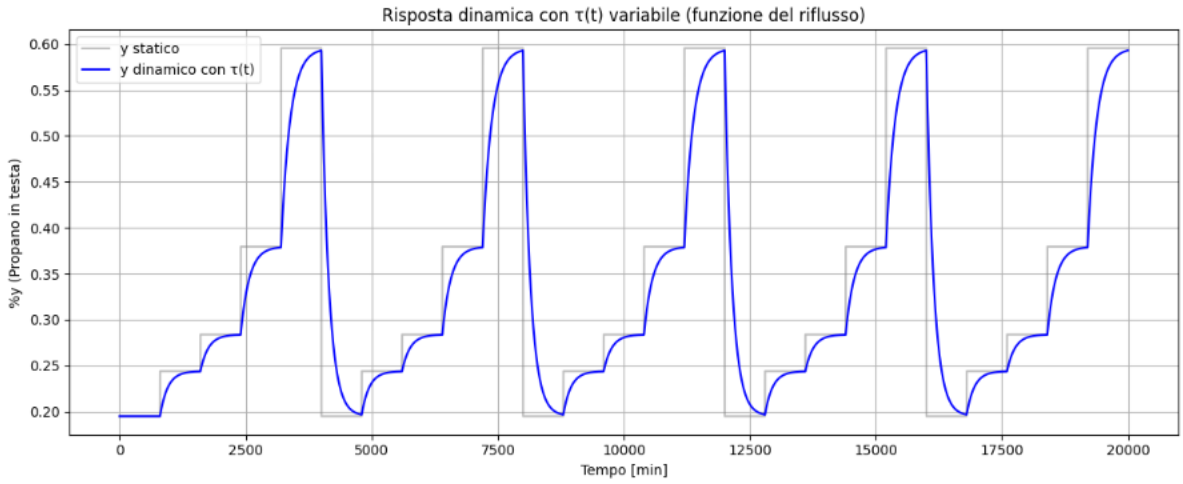


Figura 3.4: insieme dei dati di concentrazione molare del propano da analizzare

La quantità di dati fornita all'architettura *transformers* deve essere consistente, altrimenti il modello non riesce ad apprendere le rapide variazioni della variabile controllata e l'andamento del processo. Per questo motivo dopo una serie di tentativi è stato identificato il valore ideale di 20000 punti, uno per minuto, circa 13 giorni.

Il *dataset* utilizzato è composto da 20000 punti, suddivisi in blocchi da 4000 campioni ciascuno, all'interno dei quali le variabili manipolate seguono un andamento decrescente. È importante sottolineare che, ai fini dell'identificazione, risulterebbe equivalente utilizzare un andamento crescente o decrescente; nel nostro caso si è scelto il *trend* decrescente per coerenza nella costruzione dei dati. Tale configurazione consente di mettere in evidenza la capacità del

modello *transformer* di identificare correttamente la dinamica del sistema e di confrontarla con le prestazioni del modello commerciale. Si può notare che il riflusso varia da 15 a 10 con step di 1, mentre il calore fornito al *reboiler* si sposta da 3,96 MMkcal/h (1100000 cal/s) a 3,78 MMkcal/h (1050000 cal/s) con gradini di 10000 cal/s. Quando il rapporto di riflusso diminuisce implica che estraggo più prodotto in testa e rinvio meno liquido in colonna, questo causa un peggioramento nella separazione avendo un minore contatto fra la fase liquido e la fase vapore. Un discorso simile può essere fatto per il calore al *reboiler*: quando il calore fornito al ribollitore diminuisce le impurezze vanno ad aumentare, in quanto ho meno vapore che risale la colonna e quindi un minore contatto fra la fase liquido e la fase vapore. Il grado di impurezza nella corrente varia da circa 0.6% a 0.2% in frazione molare. Considerando che la specifica del propilene impone un limite massimo di 0.5 mol% di propano, sarà necessario individuare i valori di controllo ottimali per mantenerla al di sotto di tale soglia. Si osserva che l'andamento della frazione molare di propano in testa ha un andamento non costante; infatti, applicando sempre lo stesso valore delta del gradino in ingresso si osserva una variazione del propano diversa ogni volta. In particolare, per valori più bassi di riflusso e calore al ribollitore, si nota un guadagno maggiore in impurezza di propano. In un secondo momento viene fornito uno *step test* sempre nel *range* operativo delle variabili, ma in modo casuale senza seguire più un andamento preciso per testare ulteriormente l'affidabilità del modello.

3.1.2. Modello commerciale Honeywell

I modelli normalmente utilizzati in ambito commerciale, come quelli impiegati dall'azienda Alpha Process Control, sono modelli di tipo lineare. Tali modelli presentano delle limitazioni intrinseche: essi sono infatti in grado di descrivere soltanto dinamiche lineari o, al più, andamenti debolmente non lineari, senza riuscire a cogliere in maniera completa i comportamenti complessi del sistema.

Per questa predizione che usa il modello FIR vengono passati come *training* al modello 80% dei dati, per fare una predizione generale su tutti i dati. Si andrà poi a selezionare solo la parte finale, il 20% dei dati finali, quella di nostro interesse, che sarà di confronto con il nostro modello *transformers.*

I risultati ottenuti sul *test set* evidenziano i limiti del modello lineare Honeywell nell'apprendere correttamente la dinamica del processo. Come mostrato nel grafico in figura

3.5, le previsioni del modello (curva colore blu) non riescono a seguire accuratamente l'andamento reale del sistema (curva nera).

In particolare, si osserva che a fronte di variazioni a gradino delle variabili manipolate, l'impurezza di propano in testa alla colonna mostra risposte con guadagni diversi a seconda del valore iniziale. Tuttavia, il modello prevede sempre un incremento simile in corrispondenza di ogni gradino crescente, dimostrando di non cogliere la dipendenza del guadagno dal punto di inizio della variazione. Questa limitazione porta a una stima della risposta incapace di riflettere la variabilità della dinamica osservata nel processo reale.

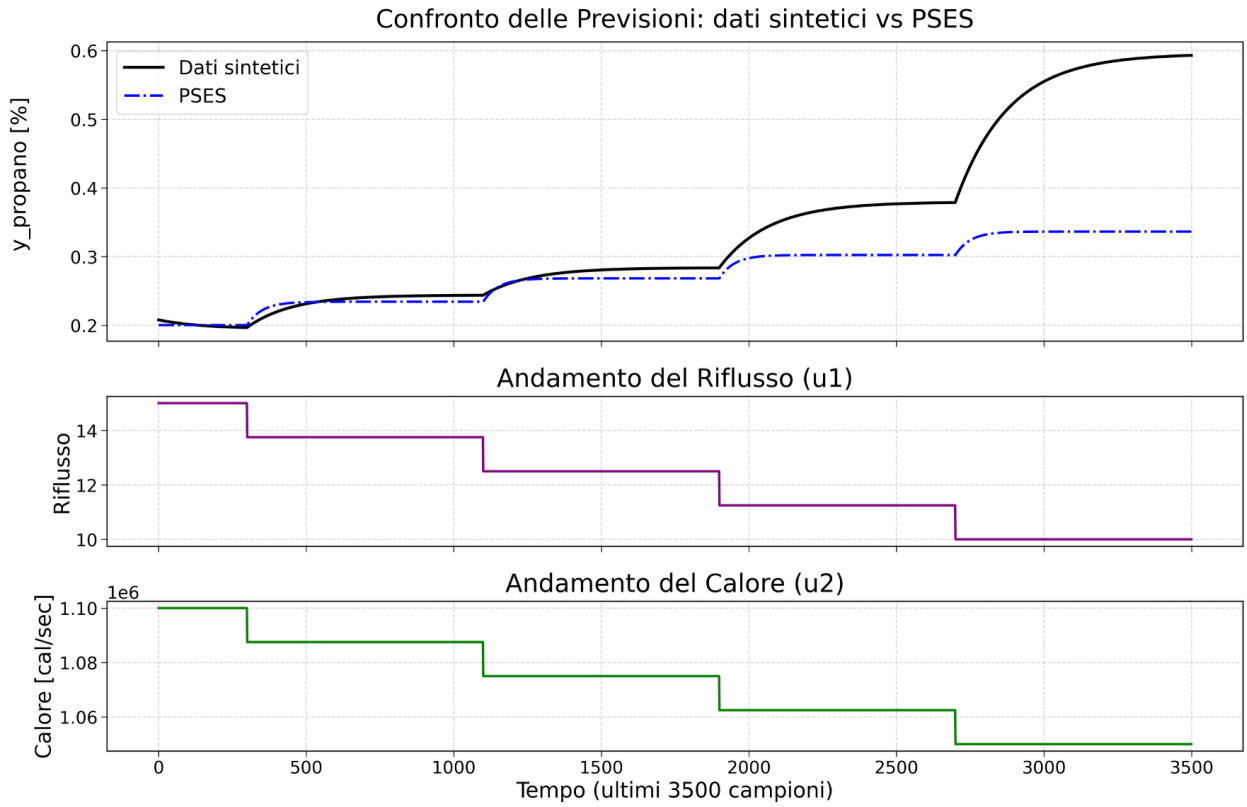


Figura 3.5: Andamento delle variabili manipolate (riflusso e calore) e confronto tra i valori reali e le previsioni dell'impurezza di propano ottenute con il modello Honeywell, sui dati del *test set*.

Anche l'analisi quantitativa tramite le metriche di errore conferma la difficoltà del modello nel rappresentare correttamente la relazione tra ingressi e uscita controllata. Il valore di MSE = 0.0076 indica un errore quadratico medio ancora significativo, con deviazioni relativamente ampie tra i valori previsti e quelli reali. Inoltre, il valore di MAE = 0.0700 conferma una

distanza media piuttosto elevata tra le stime e i dati osservati. Tuttavia, è il coefficiente di determinazione $R^2 = 0.5200$ a mostrare in modo più chiaro i limiti del modello: questo valore suggerisce che solo il 52% della varianza della variabile *target* viene spiegata dalle previsioni del modello. Un R^2 inferiore a 0.7 è generalmente considerato insufficiente nei contesti ingegneristici o di controllo avanzato, in cui è richiesta un'elevata fedeltà nella modellazione delle dinamiche del sistema. Nel complesso, queste metriche suggeriscono che il modello PSES non è sufficientemente accurato per un impiego affidabile in contesti di controllo predittivo o ottimizzazione avanzata.

3.1.3. Architettura *transformer*

Per catturare la dinamica di queste serie temporali multivariate è stata implementata una architettura *transformers* opportunamente personalizzata. Questa tecnologia permette di catturare le relazioni non lineari presenti in questo processo grazie al meccanismo di attenzione che lavora in parallelo. La suddivisione del *dataset* usata nel nostro modello dopo varie prove ha mostrato come *training* 60%, *validation* 20% e *test* 20% sia la suddivisione più adatta, essendo un buon compresso per il *fitting* e di comprensione del modello, ma senza finire in *overfitting.* L'architettura sviluppata si basa su un modello di tipo *encoder-decoder*, in cui l'*encoder* elabora una sequenza storica delle variabili di processo, mentre il *decoder* si occupa della generazione dei valori futuri della variabile target. Il *decoder* utilizza sia le rappresentazioni latenti prodotte dall'*encoder*, sia la porzione già nota della sequenza temporale nel suo orizzonte di previsione, integrando progressivamente i dati disponibili fino a un dato istante. Il modello è stato configurato attraverso un processo iterativo di ottimizzazione, che ha portato alla selezione di 8 teste di attenzione, una componente *feed-forward* con dimensione 2048, e una profondità di 2 livelli per entrambi i blocchi *encoder* e *decoder*. La normalizzazione dei dati è stata eseguita calcolando media e deviazione standard sull'intero *training set*. L'analisi è stata condotta in due fasi distinte. Nella prima, si è utilizzato il *dataset* di test fornendo all'*encoder* i valori reali della variabile *target* come contesto. Nel *decoder*, già durante la prima analisi, le previsioni erano basate esclusivamente sui valori generati fino all'i-esimo istante all'interno della finestra temporale complessiva di lunghezza T destinata alla previsione. Questa modalità, pur semplificata rispetto alla realtà operativa, ha consentito di valutare il potenziale predittivo del modello in condizioni ideali, mostrando una buona capacità di seguire l'evoluzione temporale della variabile di interesse. Tuttavia, poiché l'obiettivo finale è la sostituzione dell'analizzatore, si è affrontato il problema dell'assenza dei

valori reali in fase operativa. Per simulare uno scenario completamente autonomo, è stata quindi implementata una seconda configurazione in cui anche l'*encoder* riceve in input solo valori predetti, rimuovendo del tutto la dipendenza dai dati reali. Questo approccio ha evidenziato la possibilità di propagazione degli errori dovuta all'uso autoregressivo prolungato, ma rappresenta un test più vicino all'applicazione industriale desiderata. Nel corso degli esperimenti, si è testato il modello con due valori di dimensione degli *embedding* vettoriali: 256 e 512. La scelta di tali valori è il risultato di un compromesso tra efficienza computazionale (in termini di tempo di addestramento) e precisione delle previsioni. Per ciascun *embedding*, sono stati variati anche i parametri relativi alla lunghezza del contesto storico e all'orizzonte di previsione. La prima controlla quanti punti temporali precedenti vengono forniti all'*encoder*, mentre la seconda definisce quanto nel futuro il *decoder* deve spingersi nella previsione. I risultati delle combinazioni testate verranno mostrati nei prossimi paragrafi, accompagnati da grafici comparativi e metriche statistiche, con l'obiettivo di analizzare le prestazioni del modello.

Per testare il modello e identificare la configurazione ottimale da confrontare con il modello PSES, è stata condotta una serie di esperimenti. La fase di *tuning* è stata suddivisa in due set distinti, uno con *embedding* pari a 512 (Figura 3.6) e l'altro con *embedding* pari a 256 (Figura 3.7). Le metriche MSE e MAE mostrano in generale valori molto buoni, con oscillazioni comprese tra 0.0005 e 0.004. Tuttavia, per la valutazione comparativa si è scelto di utilizzare il coefficiente di determinazione R^2 come metrica principale, rappresentata tramite apposite *heatmap*. L'esperimento con *embedding* 512 ha evidenziato ottime prestazioni nella modalità *teacher forcing*, in particolare quando viene utilizzata una sequenza storica di 180 punti. Al contrario, nella modalità autoregressiva, si osserva in generale un peggioramento delle performance dovuto all'accumulo progressivo degli errori: ogni errore di predizione tende infatti a propagarsi nei passi successivi, compromettendo l'accuratezza complessiva. Rimangono comunque delle combinazioni ottimali anche per la modalità autoregressiva, in particolare sono molto soddisfacenti per orizzonti di predizione brevi (10 punti). L'obiettivo del lavoro però è quello di ottenere buone prestazioni anche su predizioni più lunghe. Pertanto, considerando i valori di R^2 nei casi con *pred_len* pari a 20 e 30, si è individuata come combinazione ottimale quella con *seq_len* 180 e *pred_len* 30.

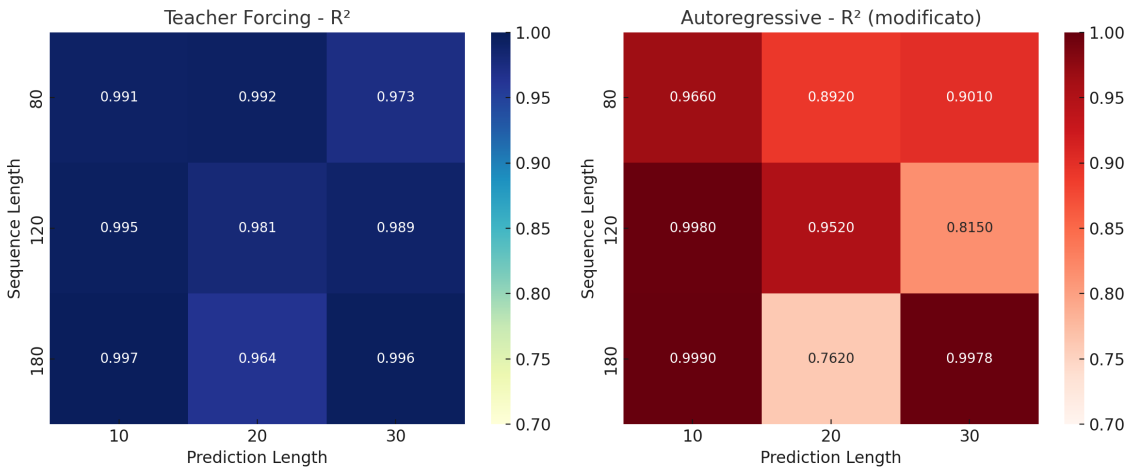


Figura 3.6: risultati ottenuti impostando il valore 512 come dimensione degli *embedding*, utilizzando come input all'*encoder* i valori precedentemente predetti dal modello. Come metrica di valutazione è stato utilizzato R^2.

Nel caso dell'*embedding* a 256 (Figura 3.7), si osservano prestazioni eccellenti nella modalità *teacher forcing*, in cui i dati reali vengono forniti all'*encoder*: quasi tutte le configurazioni raggiungono valori di R^2 prossimi a 0.99, indicando un'elevata capacità predittiva del modello. Passando invece alla modalità autoregressiva, in cui i dati predetti vengono reintrodotti nell'*encoder* durante la previsione, si assiste a un naturale decadimento delle prestazioni. Tuttavia, i valori di R^2 rimangono comunque elevati, compresi tra 0.916 e 0.936, a conferma della robustezza del modello anche in condizioni più sfidanti.

Al termine dell'analisi di *tuning,* è stata individuata come configurazione ottimale per il confronto grafico in modalità autoregressiva quella con sequenza storica di 180 punti e orizzonte di predizione di 30. L'obiettivo è quello di adottare, laddove le risorse computazionali lo permettano, un *embedding* di dimensione 512, al fine di massimizzare ulteriormente le prestazioni del modello.

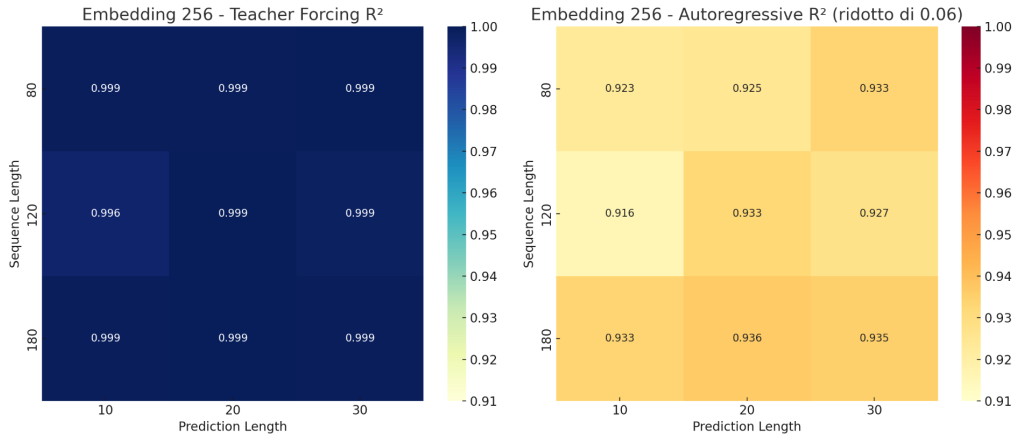


Figura 3.7: risultati ottenuti impostando il valore 256 come dimensione degli *embedding*, utilizzando come input all'*encoder* i valori precedentemente predetti dal modello. Come metrica di valutazione è stato utilizzato R^2.

3.1.4. Risultati architettura *transformer*

Si procede quindi utilizzando la configurazione ottimale individuata, con sequenza storica di 180 punti e orizzonte di predizione di 30, impiegando un *embedding* di dimensione 512. Su questa base viene effettuato il confronto tra le prestazioni del modello e le traiettorie reali del processo. I grafici delle previsioni nelle figure 3.8 e 3.9 mostrano dei buoni andamenti: le due figure riportano il confronto tra le previsioni del modello *transformer* e i valori reali della variabile d'interesse, ottenuti dal modello fisico, utilizzando due diverse modalità di test: nel grafico 3.8, è visibile il risultato ottenuto mediante *teacher forcing*, ossia usando sempre i valori reali come input per la predizione successiva. Nel grafico 3.9, invece, viene mostrato il comportamento del modello in modalità autoregressiva, in cui le predizioni precedenti vengono utilizzate come input per quelle future.

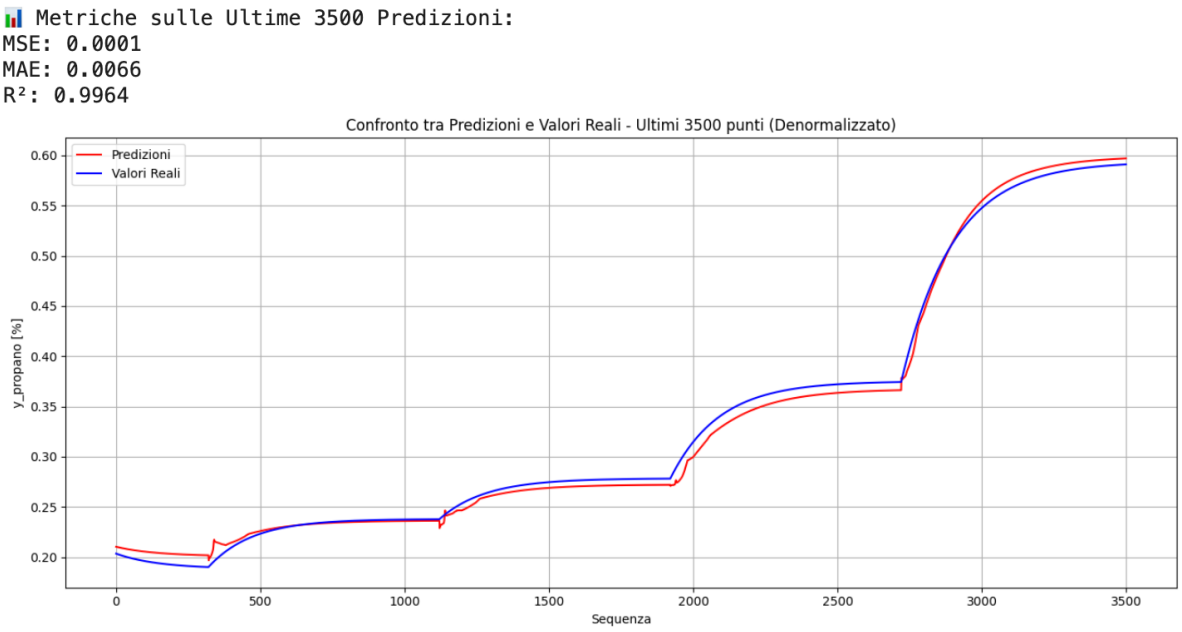


Figura 3.8: Risultati architettura *transformers* relativi all'utilizzo di valori reali dell'impurezza dell'etano in input all'*encoder* del modello.

Si osserva nel grafico un ottimo andamento di previsione con la linea rossa del modello *transformers* che segue in modo ottimale e preciso la linea blu dei dati di sintesi. Tutte le metriche confermano questa previsione ottimale del modello, con addirittura un R^2 di 0.99.

```
📊 Metriche di Valutazione sul Test Set:
MSE: 0.000034
MAE: 0.004297
R²: 0.997881
```

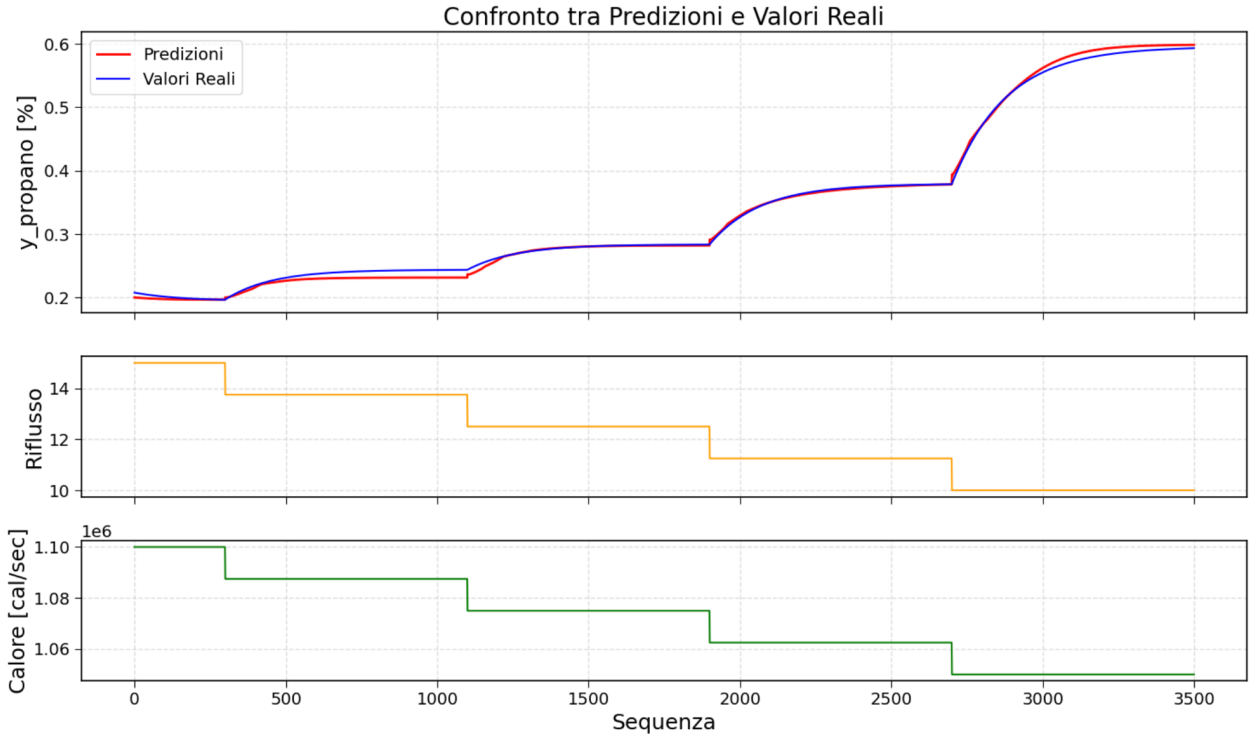


Figura 3.9: Risultati architettura *transformers* relativi all'utilizzo di valori predetti dell'impurezza di etano in input all'*encoder* del modello.

Per quanto riguarda la previsione in modalità autoregressiva, anche in questo caso il modello fornisce risultati molto soddisfacenti e previsioni complessivamente affidabili. Il modello sviluppato segue fedelmente l'andamento della variabile controllata. Inoltre, le metriche ottenute confermano l'elevata qualità del modello: il coefficiente di determinazione R^2 raggiunge un valore ottimale pari a 0.99, e anche MAE e MSE risultano validi, indicando prestazioni eccellenti nella previsione del comportamento dinamico del processo.

3.1.5. Confronto architettura *transformer* rispetto ad Honeywell

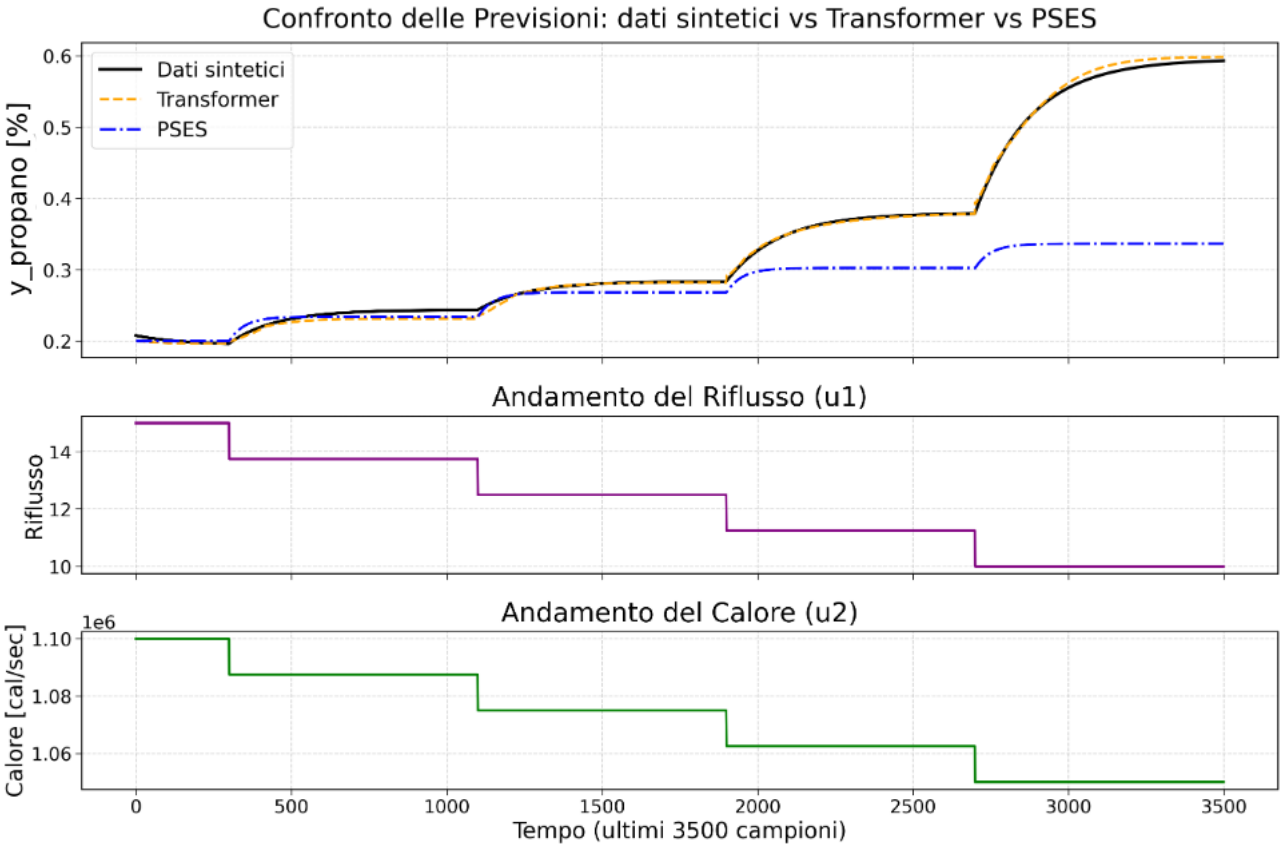


Figura 3.10: Grafico di confronto fra i dati di sintesi e il modello commerciali e il modello *transformers*

Dalla figura 3.10 si osserva il confronto grafico fra i due modelli di predizione rispetto ai dati di sintesi. Il modello *transformers* segue in modo fedele entro i limiti dell'errore l'andamento dei valori veri dell'impurezza del propano, variando e adattandosi alla variazione del guadagno non lineare del processo. Inoltre, per quanto riguarda il modello PSES, esso non riesce a valutare bene la variazione di guadagno del modello, e presenta un andamento quasi piatto della variabile d'interesse, con valori compresi fra 0.2 e 0.4 percento di propano in testa alla colonna.

Le metriche, come analizzato nei paragrafi precedenti, dimostrano la migliore bontà di apprendimento del modello di *deep learning* rispetto al semplice modello commerciale. Nello specifico per quanto riguarda R^2 che passa da uno 0.52 del modello semplice lineare ad uno 0.99 del modello *transformers*.

3.1.6. Architettura *transformer* con nuovo insieme di dati casuali

In un secondo momento dopo aver verificato nei paragrafi precedenti la migliore capacità di previsione del modello *transformers,* si procede ad applicarlo in un contesto più sfidante, in cui ho il riflusso e il calore al ribollitore che variano senza seguire un andamento prestabilito, ma in modo casuale (figura 3.11). Vengono creati 20000 dati con frequenza di un dato al minuto, con l'impurezza di propano calcolata con la stessa procedura precedentemente descritta.

Anche in questo caso si tiene l'*embedding* 512, visto il buon apprendimento del modello. Si esegue un *tuning* ottimizzato e rapido sui dati storici e la sequenza di predizione trovando come configurazione ottimale la 120-20.

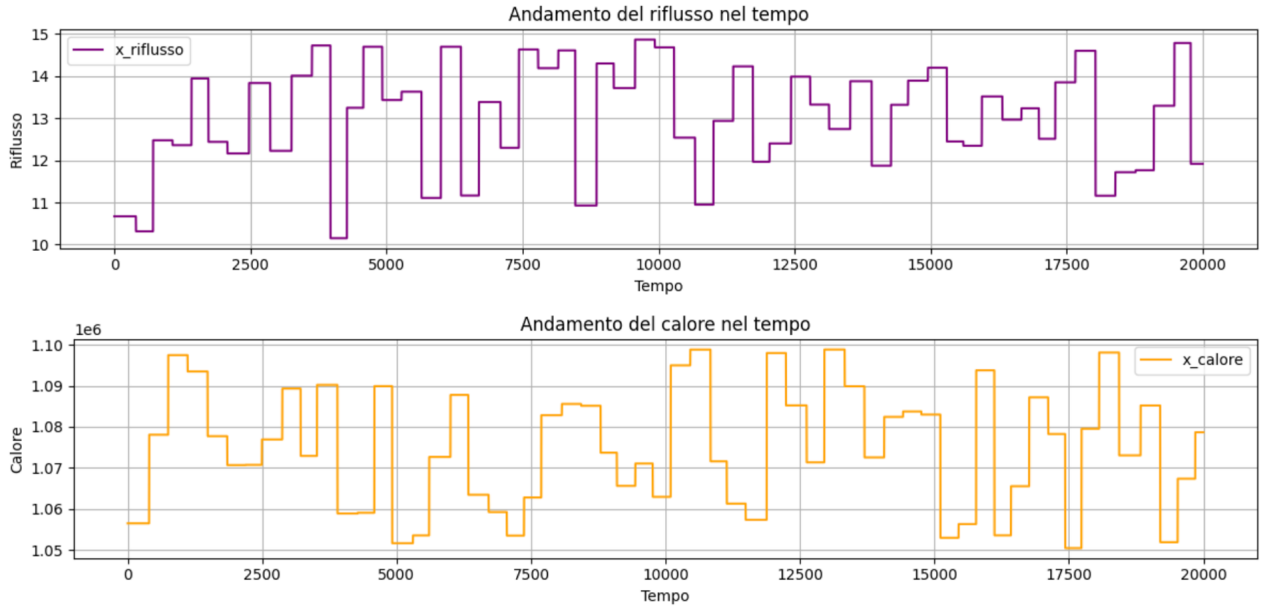


Figura 3.11: valori delle variabili manipolate nel caso di *dataset* casuale

Ci si sofferma solo più sulle previsioni relative all'utilizzo di valori predetti dell'impurezza di etano in input all'*encoder* del modello, vista poi la maggiore utilità applicativa futura.

L'andamento di previsione del *transformers* (linea rossa) segue in modo accurato i valori reali (linea blu) nella figura 3.12. Si nota un discostamento maggiore intorno al punto 500, ma con valori che rimangono comunque nel *range* dell'affidabilità del modello. Le metriche del modello rimangono ottime con valori di R^2 di 0.97 e di MAE e MSE rispettivamente di 0.0209 e 0.0006. Quindi questa architettura di *deep learning* avanzata rimane affidabile anche con variazioni dei dati di input più casuali.

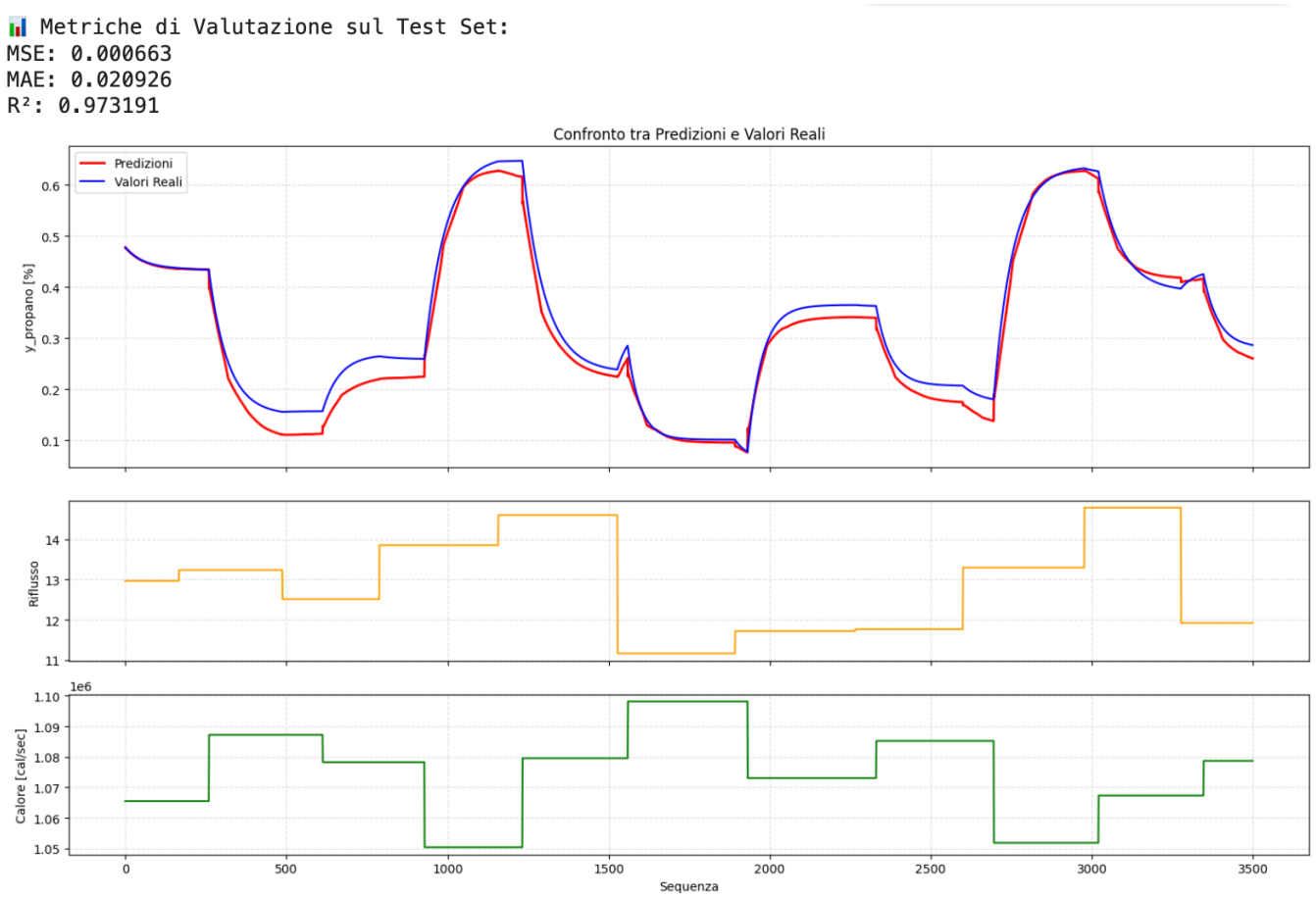


Figura 3.12: Risultati architettura *transformers* relativi all'utilizzo di valori predetti dell'impurezza di etano in input all'*encoder* del modello.

3.2. Forno di *steam cracking* dell'etano

Questa parte dell'elaborato, come introdotto in precedenza, è focalizzata sulla conversione molare dell'etano (*steam cracking*) in un forno. L'obiettivo centrale dello *steam cracking* è massimizzare la conversione dell'etano cercando di ridurre la formazione di prodotti secondari come *tail gas* (metano, idrogeno, ecc.). Nel nostro studio andremo ad analizzare e cercare di ottimizzare la conversione di etano conoscendo la concentrazione molare in ingresso e quella in uscita rilevata dall'analizzatore alla fine della reazione. Per attuare questo risultato è importante agire su 3 variabili principali: l'alimentazione, la pressione e la temperatura (figura 3.13)

I dati identificati all'inizio sono dati reali di lavoro in condizioni normali del forno, ma essendo poco variabili e per testare a pieno l'architettura *transformer* sono stati creati dati di sintesi.

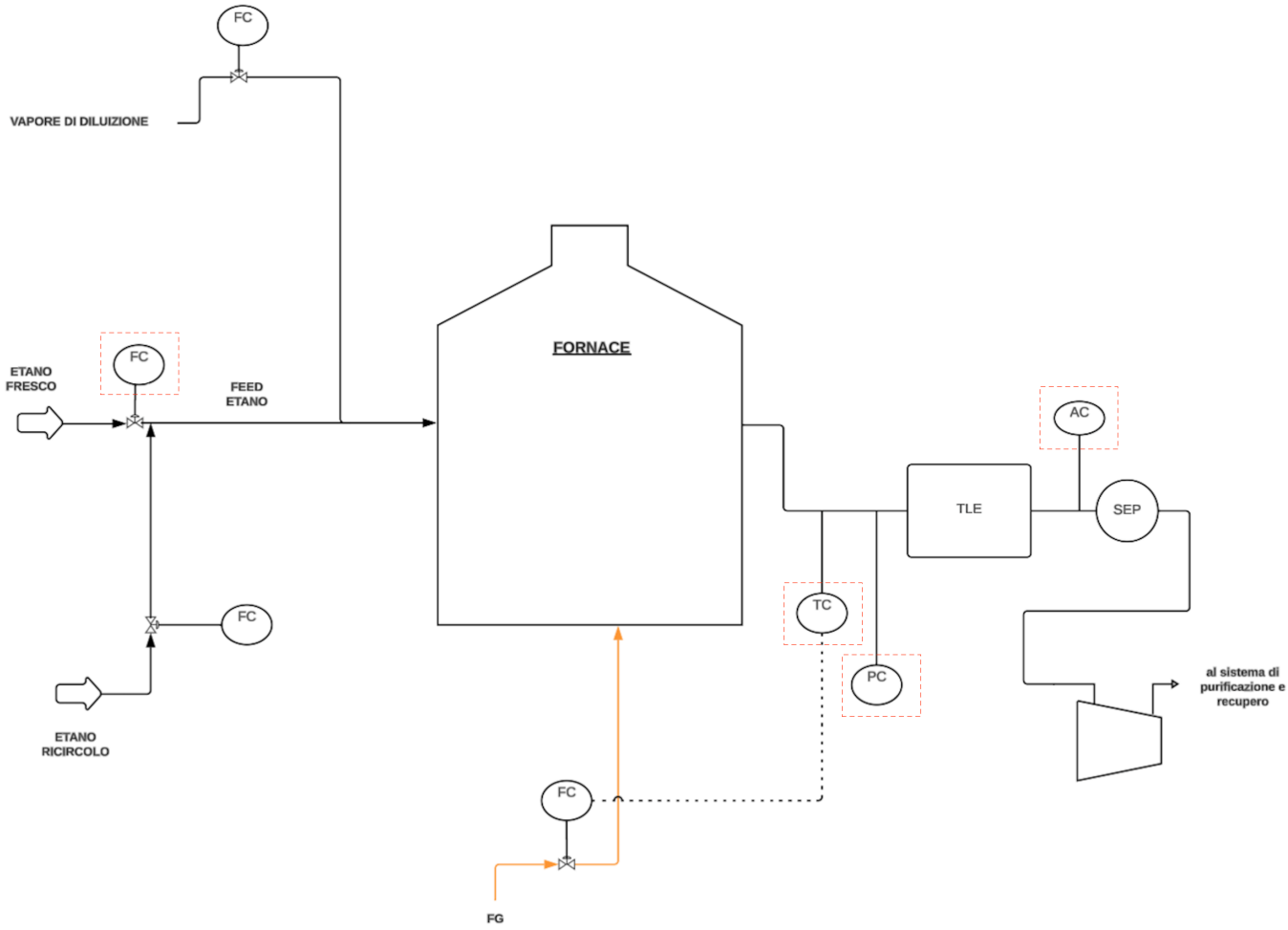


Figura 3.13: Forno ad etano con *focus* su sensori da cui sono stati presi i dati reali.

3.2.1 Dati reali d'impianto

In una fase preliminare si analizzano i dati reali d'impianto dal punto 2000 in poi in cui il forno si è stabilizzato dopo il cambio marcia (figura 3.14). In generale, si osserva che per gran parte dei 13000 punti le variabili manipolate rimangono costanti o variano per rumore di misurazione e in pochi casi vengono cambiati dall'operatore. L'alimentazione varia fra 36000 e 37000 kg/h, la COT 845 e 855 °C, la COP fra 3.5 e 3.7 kg/cm^2. Questo fa si che nella variabile conversione all'uscita del forno ci sia un range fra 62% e 64%. Quindi per questo motivo, in un secondo momento vista la poca variabilità si creano dati di sintesi.

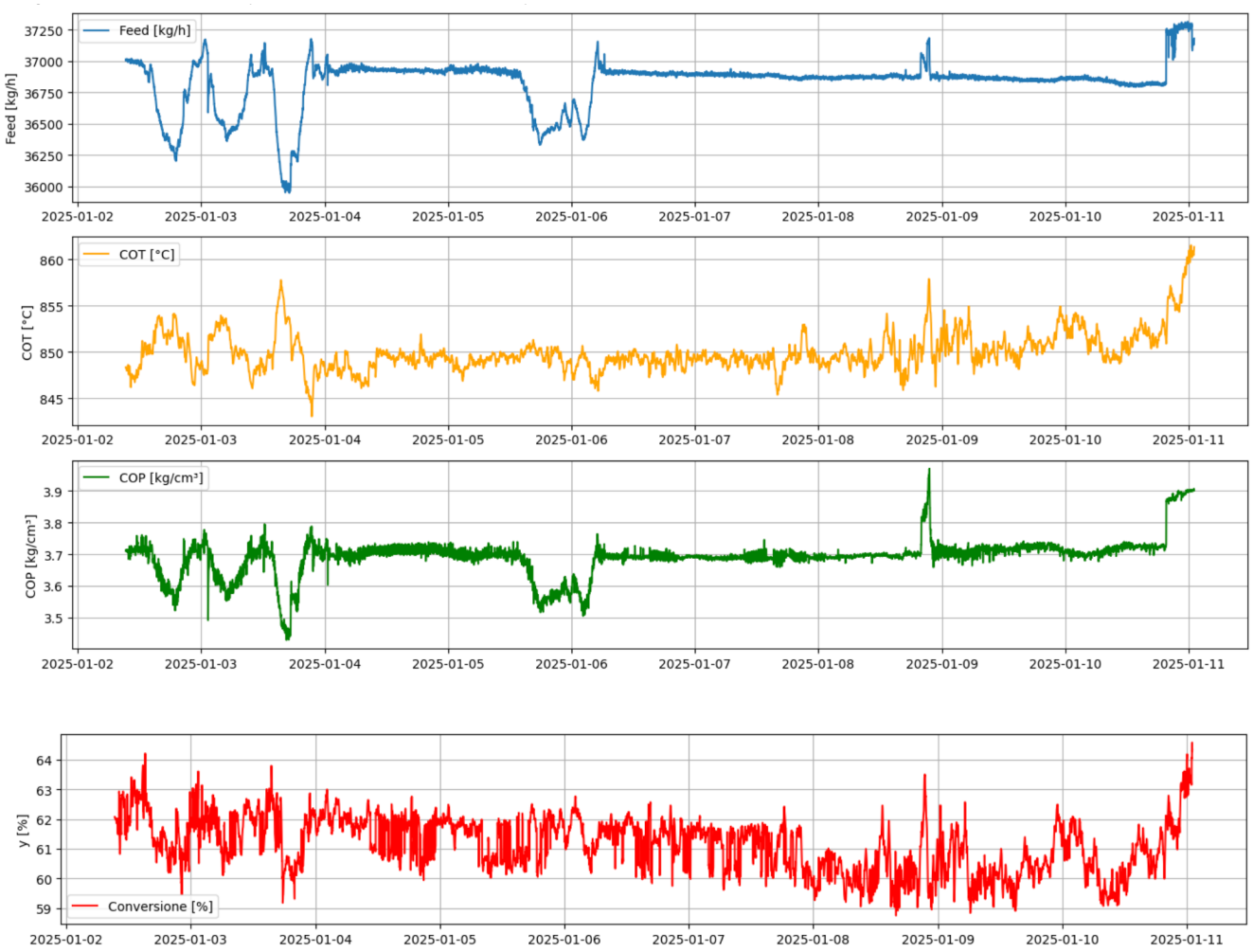


Figura 3.14: Forno ad etano dati reali d'impianto

3.2.2 Creazione dati di sintesi

Nel forno ad etano per le variabili manipolate vengono creati degli *step test,* simulando lo *step test* in impianto. Questi *step* sono definiti entro certi *range* di lavoro normali dell'impianto: la *feed* varia fra 30000 e 70000 kg/h, la COT fra 830 e 890 °C e infine la COP fra 2 e 2.5 kg/cm^2 (figura 3.15). Il *dataset* creato in questo caso è di 8000 punti, circa 6 giorni, con dati di frequenza pari ad un valore al minuto in modo da garantire più informazioni temporali da passare al modello, rendendo più affidabili le previsioni. Per quanto riguarda la conversione con questo *dataset*, anche qui si conoscono i dati statici a cui in un secondo momento viene aggiunta la dinamica. Viene usato il modello rigoroso che lega la conversione alle 3 variabili manipolate basato sul bilancio di materia e sul modello cinetico rigoroso. Una volta trovati i dati statici si inserisce la dinamica del 1 ordine del sistema con una costante di tempo

inversamente proporzionale all'alimentazione (20-50 minuti) e senza ritardo, essendo una reazione molto rapida che avviene in millisecondi. In questo modo ottengo il grafico dell'andamento dinamico della conversione di etano da analizzare con la mia architettura *transformers.*

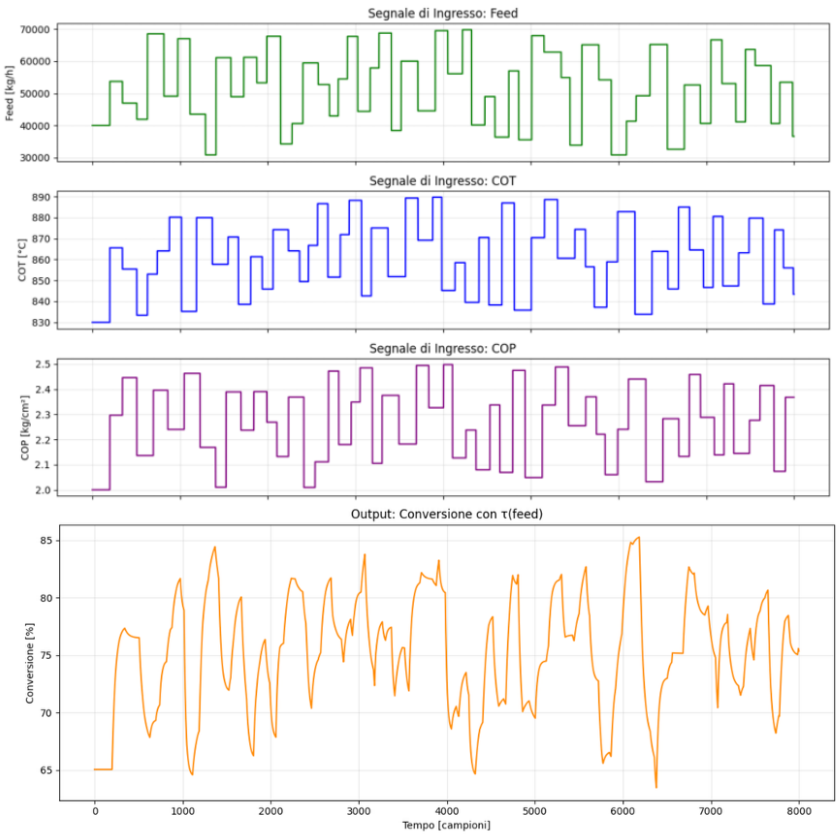


Figura 3.15: input variabili manipolate a *step test* e valori conversione

3.2.3. Modello commerciale Honeywell

I modelli normalmente utilizzati in ambiti commerciali, come quello dell'azienda Alpha Process Control, sono modelli lineari, modelli che hanno delle limitazioni e possono trovare solo certe dipendenze fra le variabili, senza riuscire però a cogliere a pieno certi andamenti. Per questa predizione che usa il modello FIR vengono passati come *training* al modello 80% dei dati, per fare una predizione su tutto l'insieme di dati. Poi si andranno a selezionare solo la parte finale d'interesse, cioè il 20% dei dati finali, che sarà di confronto con il nostro modello *transformers.* (figura 3.16)

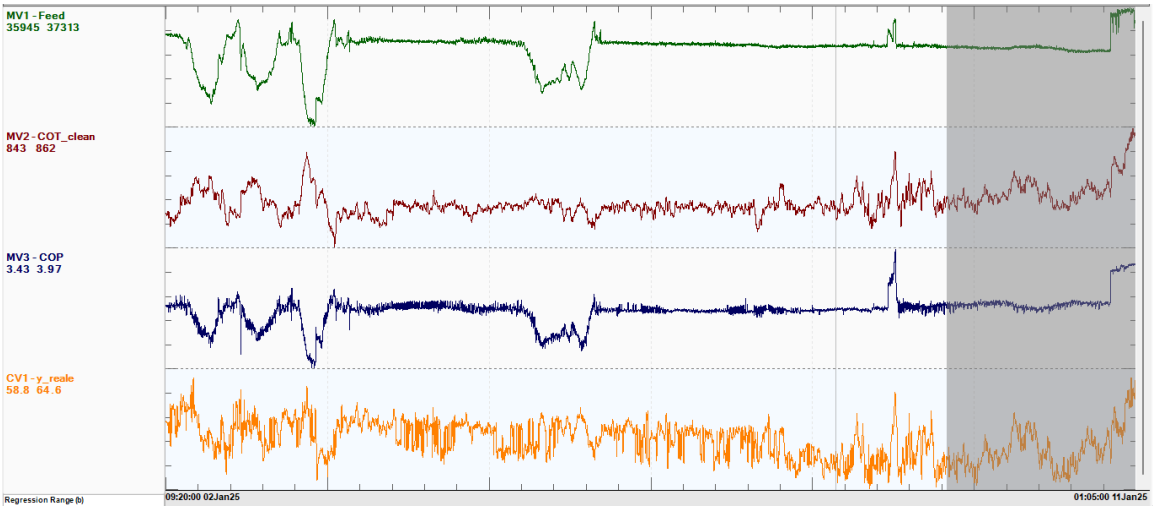


Figura 3.16: suddivisione dati di *training* e *test* su software Honeywell.

3.2.4. Architettura *transformer*

Per catturare la dinamica di queste serie temporali multivariate è stata implementata una architettura *transformers* opportunamente personalizzata. Questa tecnologia permette di catturare le relazioni non lineari presenti in questo processo grazie al meccanismo di attenzione che lavora in parallelo. La suddivisione del *dataset* usata nel nostro modello dopo varie prove ha mostrato come *training* 60%, *validation* 20% e *test* 20% sia la suddivisione più adatta, essendo un buon compresso per il *fitting* e di comprensione del modello ma senza andare in *overfitting*. L'architettura sviluppata si basa su un modello di tipo *encoder-decoder*, in cui l'*encoder* elabora una sequenza storica delle variabili di processo, mentre il *decoder* si occupa della generazione dei valori futuri della variabile target. Il *decoder* utilizza sia le rappresentazioni latenti prodotte dall'*encoder*, sia la porzione già nota della sequenza temporale nel suo orizzonte di previsione, integrando progressivamente i dati disponibili fino a un dato istante. Il modello è stato configurato attraverso un processo iterativo di ottimizzazione, che ha portato alla selezione di 8 teste di attenzione, una componente *feed-forward* con dimensione 2048, e una profondità di 2 livelli per entrambi i blocchi *encoder* e *decoder*. La normalizzazione dei dati è stata eseguita calcolando media e deviazione standard sull'intero *training set*. L'analisi è stata condotta in due fasi distinte. Nella prima, si è utilizzato il *dataset* di test fornendo all'*encoder* i valori reali della variabile *target* come contesto. Nel *decoder*, già durante la prima analisi, le previsioni erano basate esclusivamente sui valori generati fino all'i-esimo istante all'interno della finestra temporale complessiva di lunghezza T destinata alla previsione. Questa modalità, pur semplificata rispetto alla realtà operativa, ha

consentito di valutare il potenziale predittivo del modello in condizioni ideali, mostrando una buona capacità di seguire l'evoluzione temporale della variabile di interesse. Tuttavia, poiché l'obiettivo finale è la sostituzione dell'analizzatore, si è affrontato il problema dell'assenza dei valori reali in fase operativa. Per simulare uno scenario completamente autonomo, è stata quindi implementata una seconda configurazione in cui anche l'*encoder* riceve in input solo valori predetti, rimuovendo del tutto la dipendenza dai dati reali. Questo approccio ha evidenziato la possibilità di propagazione degli errori dovuta all'uso autoregressivo prolungato, ma rappresenta un test più vicino all'applicazione industriale desiderata. Nel corso degli esperimenti, si è testato il modello con due valori di dimensione degli *embedding* vettoriali: 256 e 512. La scelta di tali valori è il risultato di un compromesso tra efficienza computazionale (in termini di tempo di addestramento) e precisione delle previsioni. Per ciascun *embedding*, sono stati variati anche i parametri relativi alla lunghezza del contesto storico e all'orizzonte di previsione. La prima controlla quanti punti temporali precedenti vengono forniti all'*encoder*, mentre la seconda definisce quanto nel futuro il *decoder* deve spingersi nella previsione. I risultati delle combinazioni testate verranno mostrati nei prossimi paragrafi, accompagnati da grafici comparativi e metriche statistiche, con l'obiettivo di analizzare l'effetto di ciascun parametro sulle prestazioni del modello.

3.2.5 Risultati modelli con dati d'impianto

La prima serie di esperimenti è stata condotta utilizzando dati reali d'impianto, già precedentemente analizzati e caratterizzati da una scarsa variabilità. Proprio a causa di questa limitazione intrinseca del *dataset*, non è stato eseguito un *tuning* approfondito dei parametri del modello, ma sono state effettuate alcune prove esplorative. Di queste, vengono riportati i risultati più significativi (figura 3.17 e 3.18), benché sia evidente che la qualità limitata dei dati costituisca il principale vincolo prestazionale.

Nel primo scenario, il modello riceve come contesto storico i valori reali della variabile *target* (*y*), mostrando una buona capacità di apprendimento e generazione di previsioni accurate. Tuttavia, questa modalità non riflette le condizioni operative reali di un impianto industriale, dove l'obiettivo è ridurre la dipendenza dai sensori.

Per affrontare questo limite, è stato considerato un secondo approccio, in cui il modello opera in modalità autoregressiva: le previsioni generate vengono utilizzate come nuovo contesto in

ingresso per i passi successivi. Tuttavia, per questa previsione i grafici e i dati statistici ottenuti mostrano una previsione non accurata.

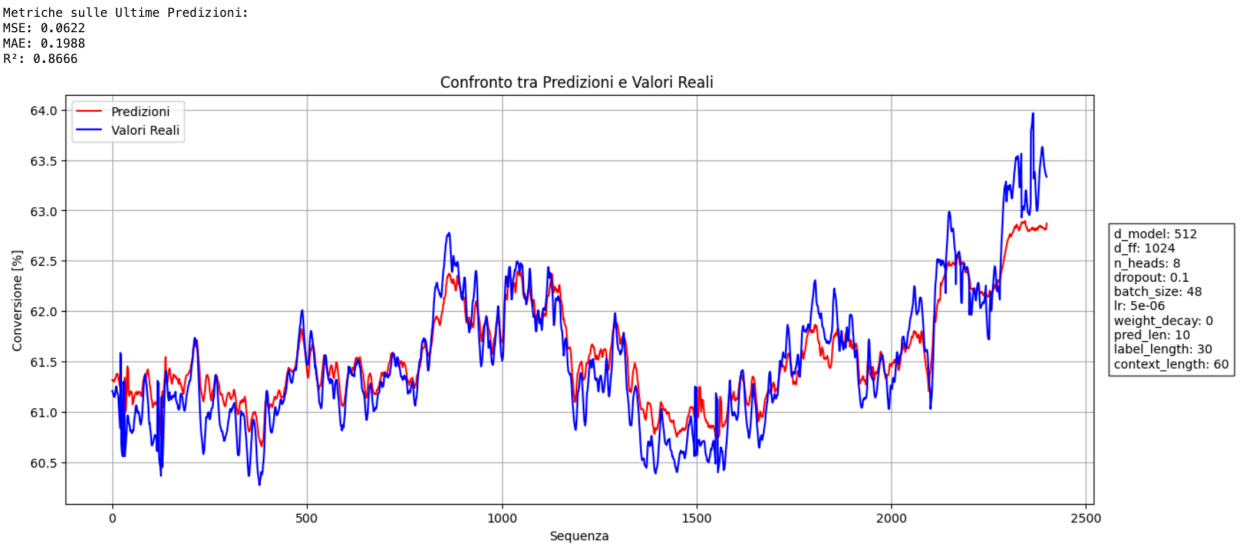


Figura 3.17: Risultati architettura *transformers* relativi all'utilizzo di valori reali della conversione dell'etano in input all'encoder del modello.

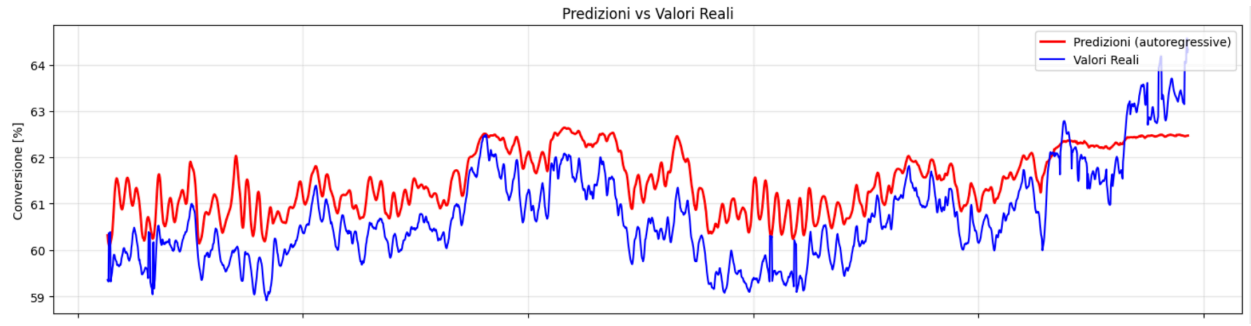


Figura 3.18: Risultati architettura *transformers* relativi all'utilizzo di valori predetti della conversione di etano in input all'encoder del modello.

In questo caso si osserva un notevole peggioramento delle prestazioni del modello, come evidenziato dalle metriche di valutazione calcolate sul test set:

- MSE: 0.8585
- MAE: 0.8536
- R^2: 0.1692

Questi risultati confermano la difficoltà di previsione in assenza dei valori reali della variabile *target* tra gli input del modello. La scarsa varianza nei dati di ingresso e l'impossibilità di fornire una traccia reale di riferimento al modello compromettono la qualità della previsione, rendendo i risultati non accettabili per un'applicazione pratica.

Per affrontare questa limitazione e testare in modo più controllato la capacità del modello di apprendere dinamiche anche non lineari, si è quindi deciso di passare ad un insieme di dati di sintesi, generati in modo da garantire una maggiore variabilità nei segnali di ingresso e una struttura più adatta alla fase di apprendimento.

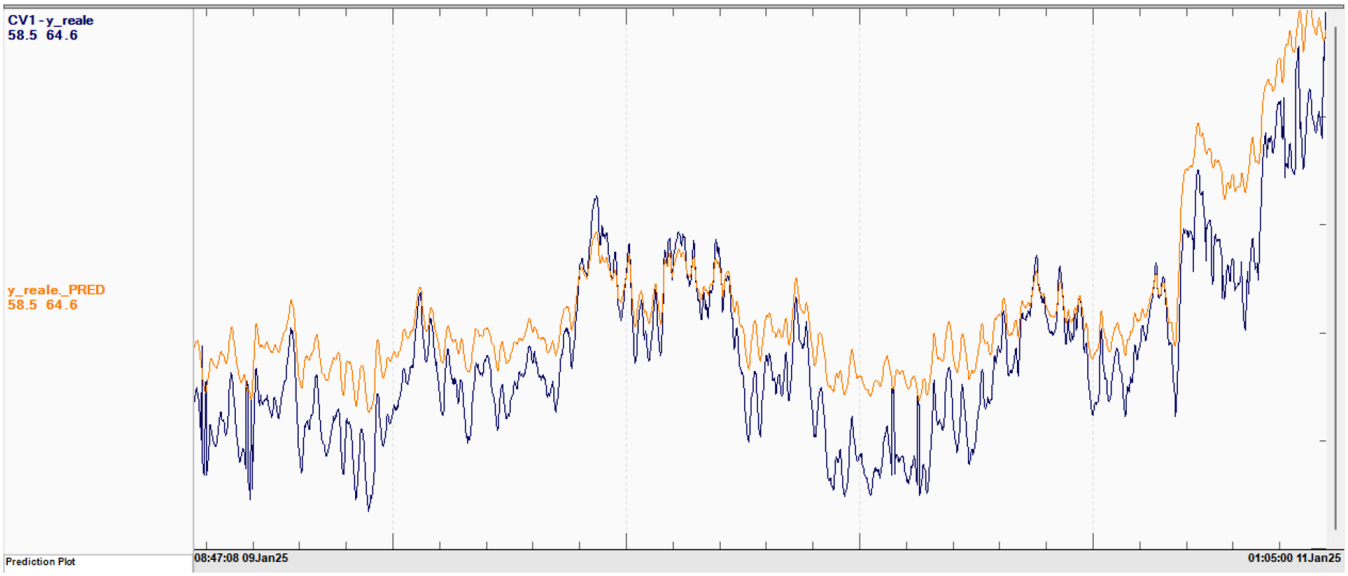


Figura 3.19: Risultati software Honeywell

Per quanto riguarda il software Honeywell (figura 3.19) anche quest'ultimo non eccelle nel predire il valore della conversione e si attesta sui risultati simili al *transformers*. Questo è dovuto al fatto che entrambi i modelli di predizione hanno troppi pochi dati variabili e non riescono ad apprendere il possibile andamento; quindi, non si può osservare il beneficio nell'utilizzo di uno dei due metodi rispetto all'altro.

3.2.6 Risultati con dati di sintesi

Con il secondo set di esperimenti svolto questa volta sui dati di sintesi si osservano risultati buoni sia per il caso in cui il contesto storico siano valori reali o previsioni. Le metriche di valutazione sono ottime per entrambi i casi con un andamento oscillante per R^2 fra 0.95 e 0.98, che per questo motivo non può essere preso come parametro principale di confronto. Per quanto riguarda MSE e MAE vengono costruite delle *heat map* per i due casi:

- per il caso con *teacher forcing*:

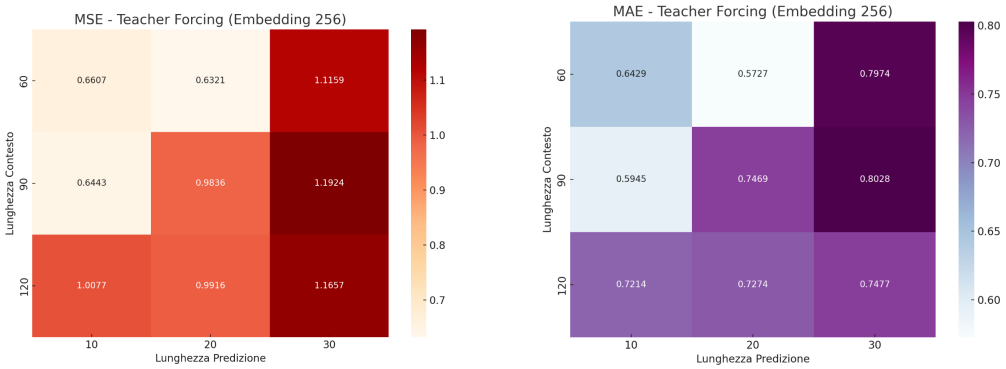


Tabella 3.20. risultati ottenuti impostando il valore 256 come dimensione degli *embedding*, utilizzando come input all'*encoder* i valori reali. Come metrica di valutazione sono stati utilizzati l'errore quadratico medio MSE e l'errore assoluto medio MAE

In queste *heat map* si osserva che le previsioni con lunghezza predetta 30 minuti hanno metriche molto peggiori rispetto agli altri casi e per questo motivo sono da scartare. La metrica migliore da scegliere in questo caso è quella con lunghezza di predizione 20 minuti passando un contesto storico di dati di 60 minuti.

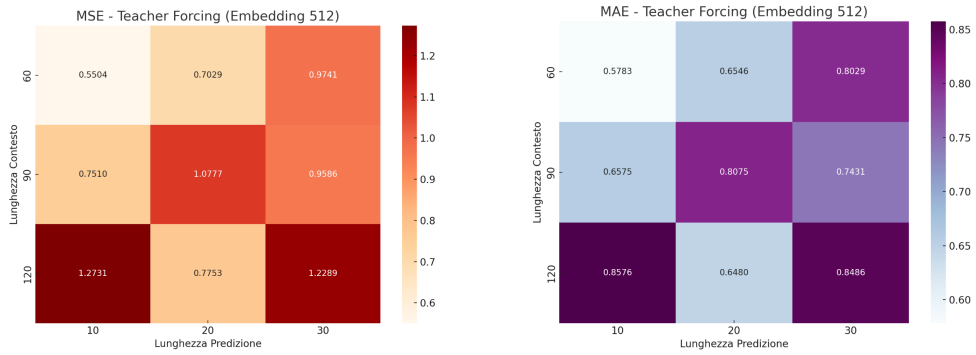


Tabella 3.21. risultati ottenuti impostando il valore 512 come dimensione degli *embedding*, utilizzando come input all'*encoder* i valori reali. Come metrica di valutazione sono stati utilizzati l'errore quadratico medio MSE e l'errore assoluto medio MAE

Nel confronto tra le diverse configurazioni di lunghezza del contesto e dell'orizzonte di previsione, si osservano metriche peggiori per l'accoppiamento contesto 120 e previsione 30 minuti, che viene quindi escluso.

Il miglior risultato complessivo si ottiene con l'accoppiamento 60–10, ma questo fornisce una finestra predittiva limitata a soli 10 punti. Per questo motivo, si cercano le migliori prestazioni tra gli accoppiamenti con orizzonte di previsione di 20 o 30 punti, individuando come opzione

ottimale la configurazione 90–20, che rappresenta un buon compromesso tra accuratezza e lunghezza della previsione. Per andare a fare un confronto grafico con i valori reali si utilizzerà la configurazione a 256 avendo metriche di errore leggermente migliore e che hanno un costo computazionale minore.

- Per il caso autoregressivo:

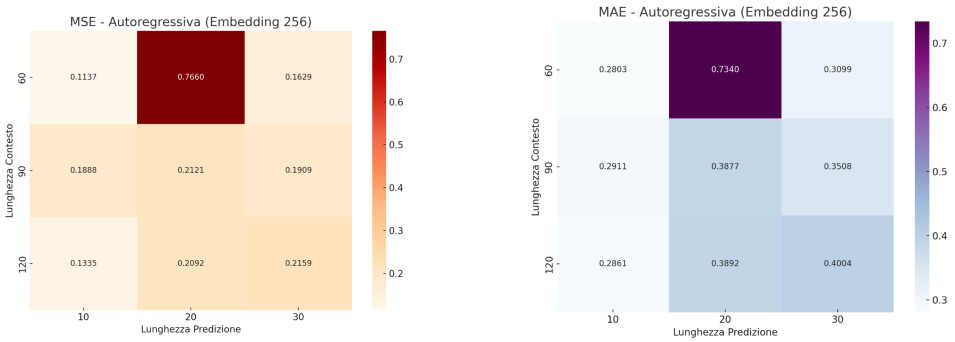


Figura 3.22. risultati ottenuti impostando il valore 256 come dimensione degli *embedding*, utilizzando come input all'encoder i valori precedentemente predetti dal modello. Come metrica di valutazione è stato utilizzato l'errore quadratico medio MSE e l'errore assoluto medio MAE.

In questa configurazione di *heat map* si osserva come ci sia una configurazione da evitare la 60-20. Inoltre, per lunghezza di predizione 10 minuti si ottengono i risultati migliori ma ovviamente l'orizzonte predetto sarà appunto limitato. Quindi si sceglie una previsione di 30 minuti con lunghezza di contesto 60 che da 0.1629 come MSE e 0.3099 come MAE.

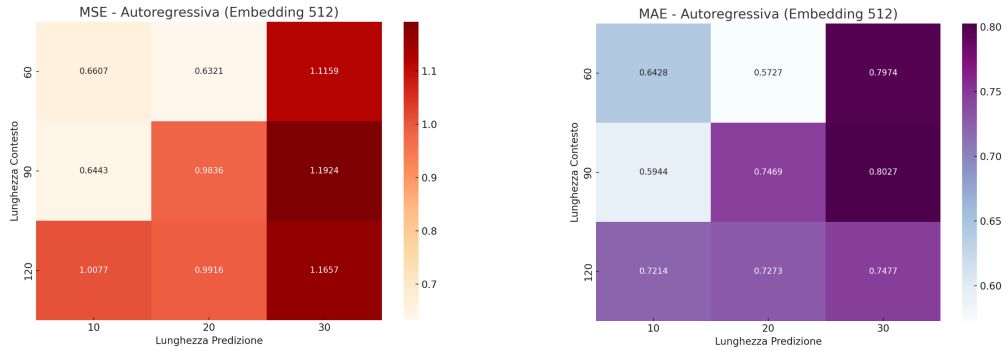


Figura 3.23. risultati ottenuti impostando il valore 512 come dimensione degli *embedding*, utilizzando come input all'encoder i valori precedentemente predetti dal modello. Come metrica di valutazione è stato utilizzato l'errore quadratico medio MSE e l'errore assoluto medio MAE.

Per il caso autoregressivo con *embedding* 512 si vede come MSE e MAE abbiano dei valori peggiori in generale in tutti gli esperimenti. Si osserva in particolare un netto peggioramento

quando si effettua una previsione di 30 punti futuri. Le migliori modellazioni si ottengono anche qui con lunghezza di predizione 10 o 20 minuti e limitando la lunghezza di contesto a 60 minuti. Questa configurazione a 512 di *embedding* verrà scartata preferendogli quella più semplice a 256.

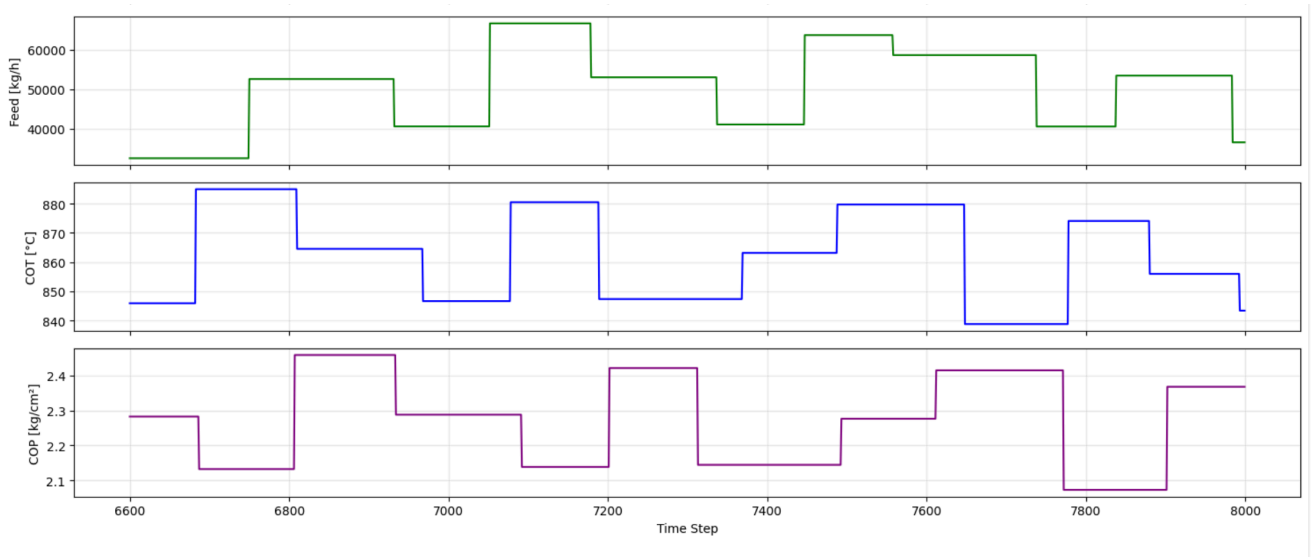


Figura 3.24: valori delle manipolate nel *test set*

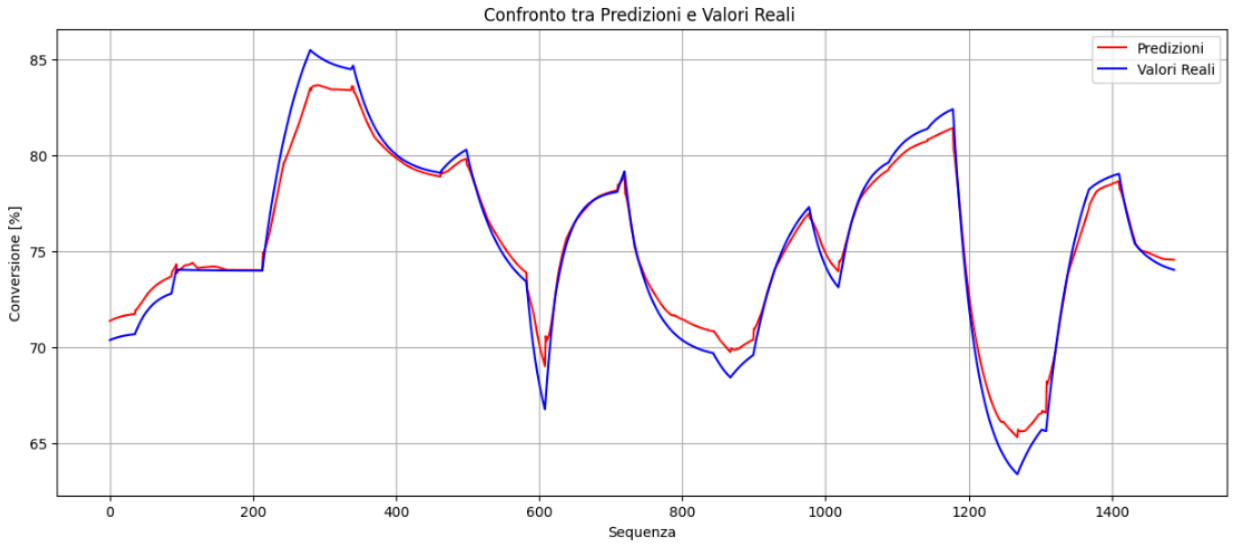


Figura 3.25: Risultati architettura *transformers* relativi all'utilizzo di valori reali della conversione dell'etano in input all'*encoder* del modello.

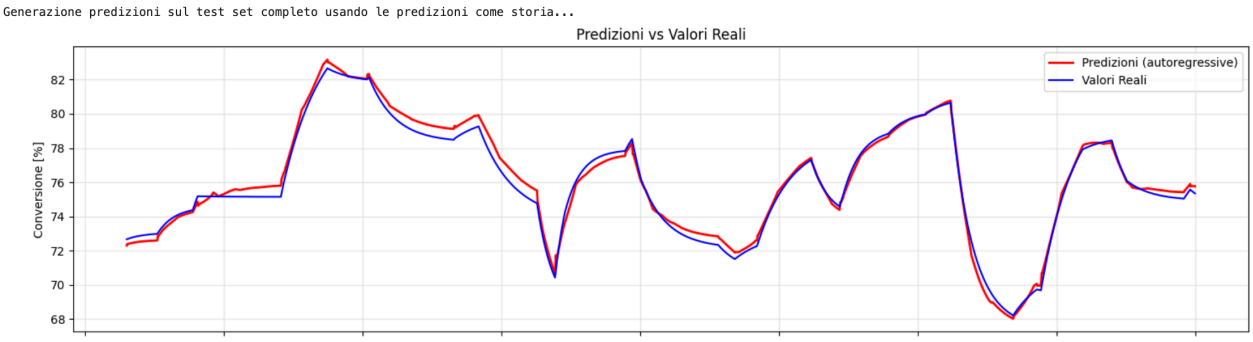
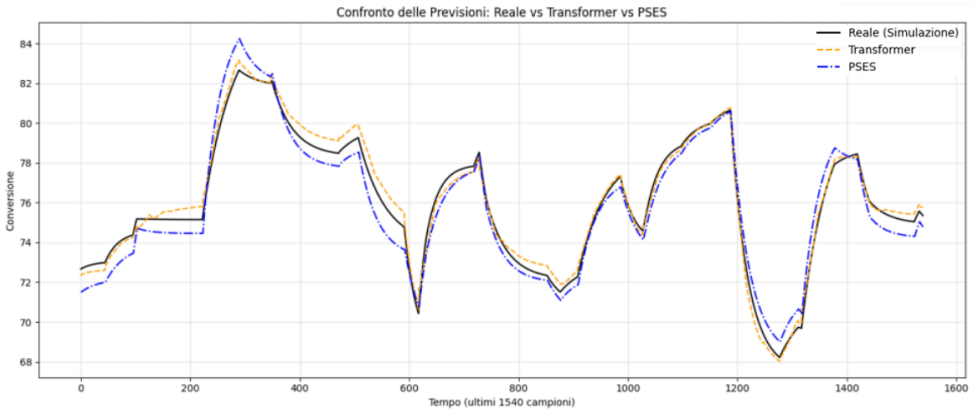


Figura 3.26: Risultati architettura *transformers* relativi all'utilizzo di valori predetti della conversione di etano in input all'*encoder* del modello.

I grafici delle previsioni nelle figure 3.25 e 3.26 mostrano dei buoni andamenti: le due figure riportano il confronto tra le previsioni del modello *transformer* e i valori reali di conversione, ottenuti dal modello del processo, utilizzando due diverse modalità di test: nel grafico 3.25, è visibile il risultato ottenuto mediante *teacher forcing*, ossia usando sempre i valori reali come input per la predizione successiva. Nel grafico 3.26, invece, viene mostrato il comportamento del modello in modalità autoregressiva, in cui le predizioni precedenti vengono utilizzate come input per quelle future.

A livello qualitativo si nota come l'approccio autoregressivo segua in maniera più coerente l'evoluzione dei valori reali (curva blu), soprattutto nelle fasi dinamiche più accentuate come i picchi e le discese ripide. Al contrario, il modello testato in *teacher forcing* mostra una maggiore instabilità e una minore aderenza, in particolare nei transitori e nelle zone ad alta variabilità.




```
📊 Metriche di Errore rispetto al Reale:
--------------------------------------------------
Transformer:
  MSE  = 0.1629
  MAE  = 0.3099
  R²   = 0.9928
--------------------------------------------------
PSES:
  MSE  = 0.5614
  MAE  = 0.6498
  R²   = 0.9462
```

Figura 3.27: Confronto risultati software Honeywell (linea blu) e *transformers* (linea gialla) nella previsione dei dati sintetici (linea nera)

La figura mostra il confronto tra le previsioni ottenute dal modello transformer (linea gialla), il modello commerciale Honeywell PSES (linea blu) e i valori reali derivanti dalla simulazione fisica (linea nera). Si osserva che il modello *transformer* tende a seguire con maggiore precisione l'andamento reale, in particolare nelle fasi di variazione rapida della variabile di conversione. Tuttavia, la differenza rispetto al modello Honeywell non appare così marcata da poter essere considerata conclusiva. Dal punto di vista quantitativo, le metriche (MSE, MAE e R^2) confermano un miglioramento a favore del *transformer*, ma la portata di tale differenza richiede ulteriori esperimenti e validazioni per essere definitivamente consolidata.

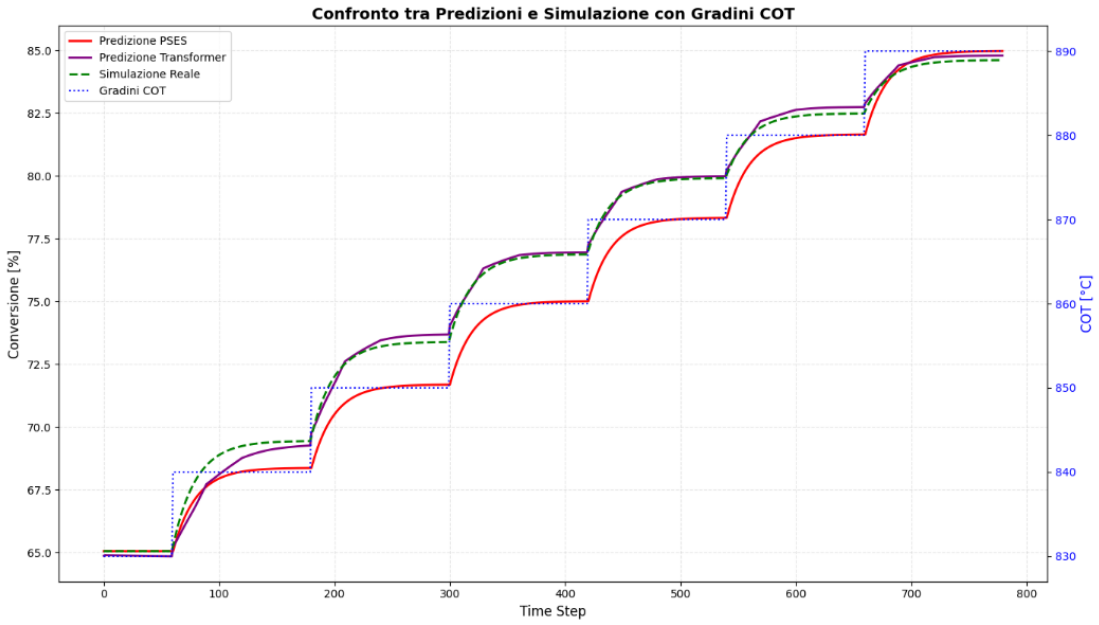
3.2.7 Risultati variazione della singola variabile

Nel grafico riportato sopra, si evidenzia la superiore capacità predittiva del modello *transformer* rispetto al modello commerciale (PSES by Honeywell). Per mettere in risalto questa differenza di apprendimento, sono stati eseguiti test isolati sulle singole variabili manipolate, mantenendo costanti le restanti. Il modello *transformer* è stato testato con gli stessi parametri e pesi già ottimizzati nel *training* generale, variando solamente il *dataset* di test. Per quanto riguarda l'alimentazione, entrambi i modelli, *transformer* e PSES, hanno mostrato prestazioni equivalenti. Questo suggerisce una relazione pressoché lineare tra il segnale di alimentazione e la conversione, che risulta facilmente catturabile anche da un modello commerciale con una costante di guadagno fissa di -0.00021.

Al contrario, la variabile COT ha mostrato una dinamica decisamente più complessa. In particolare, variando la COT da 830°C a 890°C, si osserva un incremento della conversione fino al 20%. Analizzando il comportamento della risposta, emerge una non linearità marcata: i primi gradini a basse temperature determinano un aumento più accentuato della conversione, mentre il guadagno decresce progressivamente con l'aumentare della temperatura.

Il modello PSES di Honeywell non riesce a cogliere questa variazione di guadagno in funzione della temperatura: restituisce un comportamento lineare con un guadagno medio approssimato a +0.33. Di conseguenza, nelle prime fasi (temperature basse) sottostima l'incremento di conversione, accumulando errore; nelle fasi successive, con temperature più alte, l'errore si riduce in quanto la media del modello si avvicina al guadagno reale. Al contrario, il modello *transformer* è in grado di adattarsi dinamicamente al variare della temperatura, cogliendo la diminuzione del guadagno e restituendo previsioni molto più accurate lungo tutto l'intervallo di test. Questa analisi conferma che, in presenza di comportamenti non lineari e dinamici, il

modello *transformer* riesce ad apprendere variazioni locali e strutture complesse che un modello commerciale a guadagno fisso non è in grado di rappresentare. (figura 3.28)




```
📈 Metriche di confronto (rispetto alla simulazione reale):

PSES vs Reale:
🔹 MSE: 1.647165
🔹 MAE: 1.122222
🔹 R²: 0.955970
🔹 Differenza massima: 2.014125

Transformer vs Reale:
🔹 MSE: 0.079618
🔹 MAE: 0.213677
🔹 R²: 0.997872
🔹 Differenza massima: 1.031437
```

Figura 3.28: Risultati software Honeywell vs *transformers* per COT a gradini crescenti

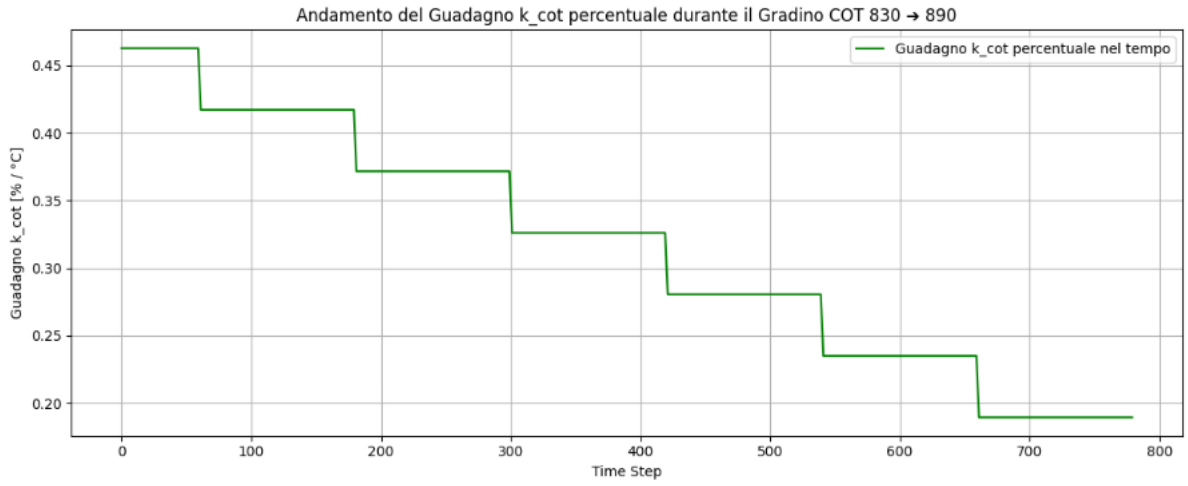
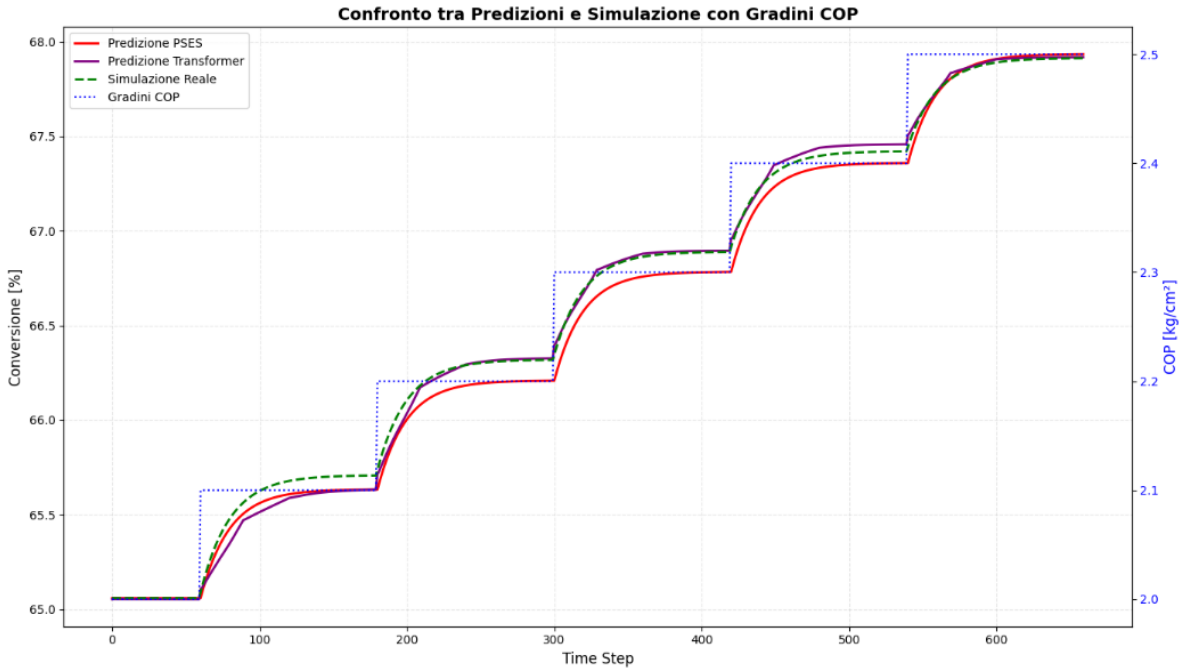


Figura 3.29: grafico guadagno predetto dal modello *transformers*

Un andamento analogo a quello osservato per la COT si riscontra anche per la variabile COP, seppur in forma meno marcata. L'effetto della COP sulla conversione risulta infatti più contenuto: variando la pressione nel suo *range* operativo ottimale in impianto, la conversione si sposta da circa 65% al 68%.

Il modello commerciale PSES di Honeywell assume un guadagno medio fisso di circa +5.73 per tutta la curva. Tuttavia, la realtà mostra un comportamento non lineare: il guadagno varia in modo inversamente proporzionale al valore della pressione. In particolare, il contributo alla conversione è maggiore a pressioni più basse e si riduce progressivamente all'aumentare della COP. Di conseguenza, il modello commerciale tende a sovrastimare o sottostimare l'effetto reale nei diversi tratti del gradino. L'errore di previsione risulta massimo nel tratto centrale degli step test, dove il guadagno reale si discosta maggiormente dalla media stimata dal modello. Questo conferma, anche per la variabile COP, la superiorità del modello *transformer* nell'adattarsi alla variazione dinamica e non lineare del sistema, fornendo previsioni più accurate lungo tutto il *range* operativo. (figura 3.30)




```
📈 Metriche di confronto (rispetto alla simulazione reale):

PSES vs Reale:
🔹 MSE: 0.006080
🔹 MAE: 0.067605
🔹 R²: 0.992596
🔹 Differenza massima: 0.137688

Transformer vs Reale:
🔹 MSE: 0.002080
🔹 MAE: 0.031088
🔹 R²: 0.997467
🔹 Differenza massima: 0.131316
```

Figura 3.30: Risultati software Honeywell vs *transformers* per COP a gradini crescenti

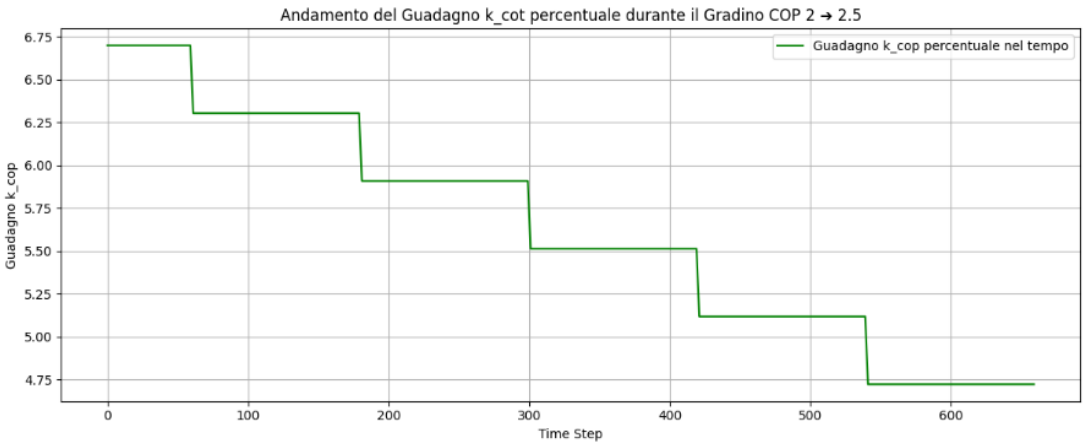


Figura 3.31: grafico del guadagno predetto dall'architettura *transformers*

3.2.8 Risultati *range* amplificato

Viste le buone performance ottenute dal modello *transformer* sui dati di sintesi iniziali, si è deciso di condurre un ulteriore esperimento utilizzando un insieme di dati con un *range* amplificato delle variabili manipolate, al fine di testare la capacità del modello di apprendere comportamenti più complessi e non lineari. L'obiettivo di questa espansione dei *range* operativi è puramente esplorativo e di validazione del modello, poiché in un contesto impiantistico reale le manipolate tendono a operare entro valori limitati, difficilmente soggetti a variazioni così estese. In particolare:

- l'alimentazione (*feed*) è stata mantenuta costante, in quanto dalle analisi precedenti è emerso che il suo contributo è tendenzialmente lineare;
- la temperatura (COT) è stata estesa nel *range* 830 °C – 1030 °C. Questa scelta è motivata dal fatto che oltre una certa soglia il guadagno diventa negativo, introducendo un comportamento fortemente non lineare (si veda Figura *3.33-3.34*);
- la pressione (COP) è stata invece estesa nel *range* 2 – 5 kg/cm², anch'essa per osservare l'emergere di un'inversione del guadagno e un incremento dell'effetto non lineare (si veda Figura *3.35-3.36*).

Per rappresentare correttamente questi comportamenti e garantire un'adeguata copertura del nuovo spazio di input, si è generato un insieme di dati di sintesi casuale con una frequenza di campionamento pari ad un valore al minuto, raddoppiando il numero di dati a 20000 punti. Questa scelta è stata dettata dalla necessità di fornire al modello *transformer* una quantità di

dati sufficiente per cogliere le variazioni più profonde, complesse e non lineari introdotte dalla nuova configurazione.

A partire da questo nuovo *dataset*, si è proceduto con una fase di *tuning* limitata, mantenendo il valore dell'*embedding dimension* a 256 (dato l'equilibrio osservato tra accuratezza e complessità computazionale nei test precedenti) e variando solamente la lunghezza della sequenza storica e dell'orizzonte di predizione. Si è trovato che la migliore configurazione è anche in questo caso la 60-20, con 60 dati passati come contesto storico e 20 punti predetti futuri.

L'attenzione si è concentrata in particolare sulle previsioni in modalità autoregressiva, in cui i valori predetti della conversione di etano vengono reintrodotti come input all'*encoder* del modello. Questa modalità è infatti la più rilevante in vista di un possibile utilizzo del modello in un sistema di controllo predittivo. Come evidenziato in figura 3.32, i risultati ottenuti sul *test set* risultano soddisfacenti: l'andamento della conversione viene approssimato in maniera coerente in ogni tratto della conversione, confermando l'efficacia del *transformer* nel modellare dinamiche complesse fuori dal *range* operativo nominale.

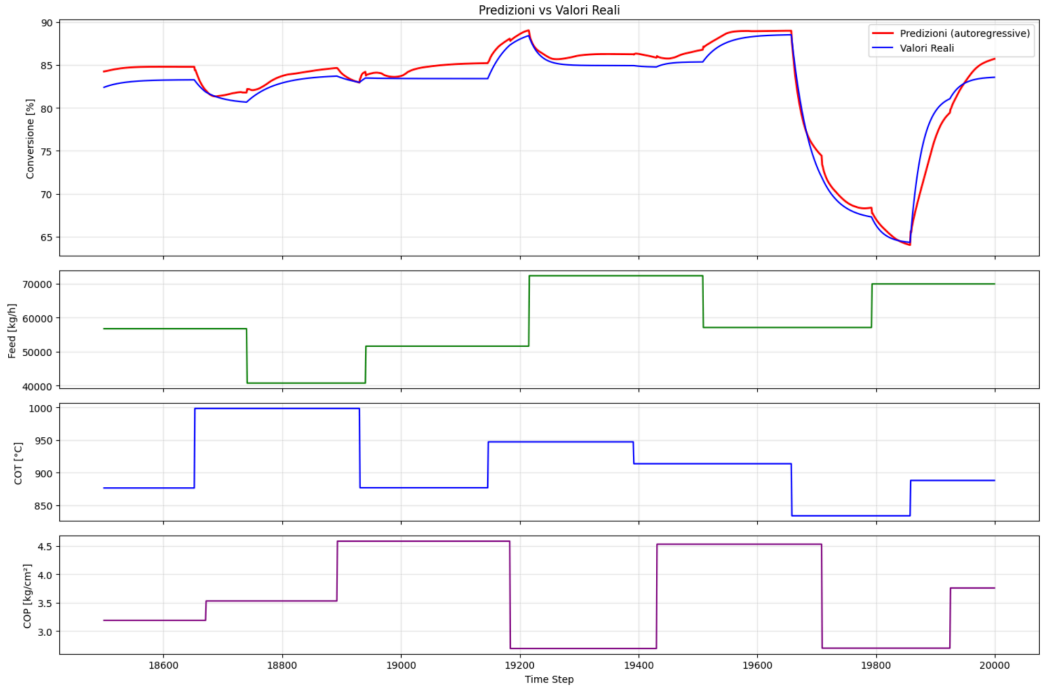


Figura 3.32: risultati relativi all'utilizzo di valori predetti della conversione di etano in input all'*encoder* del modello.

L’andamento della conversione viene predetto in modo accurato dal modello *transformer* sia nella fase iniziale, in cui la conversione si mantiene elevata (tra l’80% e il 90%), sia nella parte finale della sequenza, caratterizzata da un calo marcato fino a circa il 65% per valori di COP e COT più bassi (figura 3.32). Questo comportamento conferma la capacità del modello di adattarsi a variazioni significative all’interno della sequenza temporale.

Le metriche di valutazione sul *test set* risultano lievemente inferiori rispetto al caso precedente con variabili nel *range* operativo ottimale: si osserva infatti un *Mean Squared Error* (MSE) pari a 1.5767, un *Mean Absolute Error* (MAE) pari a 1.0554 e un coefficiente di determinazione R^2 pari a 0.9524. Sebbene vi sia un leggero peggioramento rispetto ai test precedenti, tali valori rimangono comunque ottimi e indicativi di un apprendimento efficace da parte del modello, anche in presenza di dinamiche non lineari accentuate.

Come ultima fase di validazione, sono stati eseguiti test mirati su gradini isolati delle due variabili manipolate caratterizzate da comportamenti non lineari (COT e COP). I risultati ottenuti confermano la robustezza dell’architettura *transformer* nel catturare correttamente sia la risposta nella regione operativa ottimale, dove il guadagno delle variabili è positivo e sia nelle zone fuori scala, in cui il guadagno cambia segno. Questa capacità di rappresentare correttamente la variazione di segno nel guadagno rappresenta un aspetto cruciale nell’identificazione di sistemi non lineari e nel successivo utilizzo del modello in contesti predittivi o di controllo.

Figura 3.33: risultati relativi all'utilizzo di valori predetti della conversione di etano in input all'*encoder* del modello per step a gradino della COT da 830 a 890.

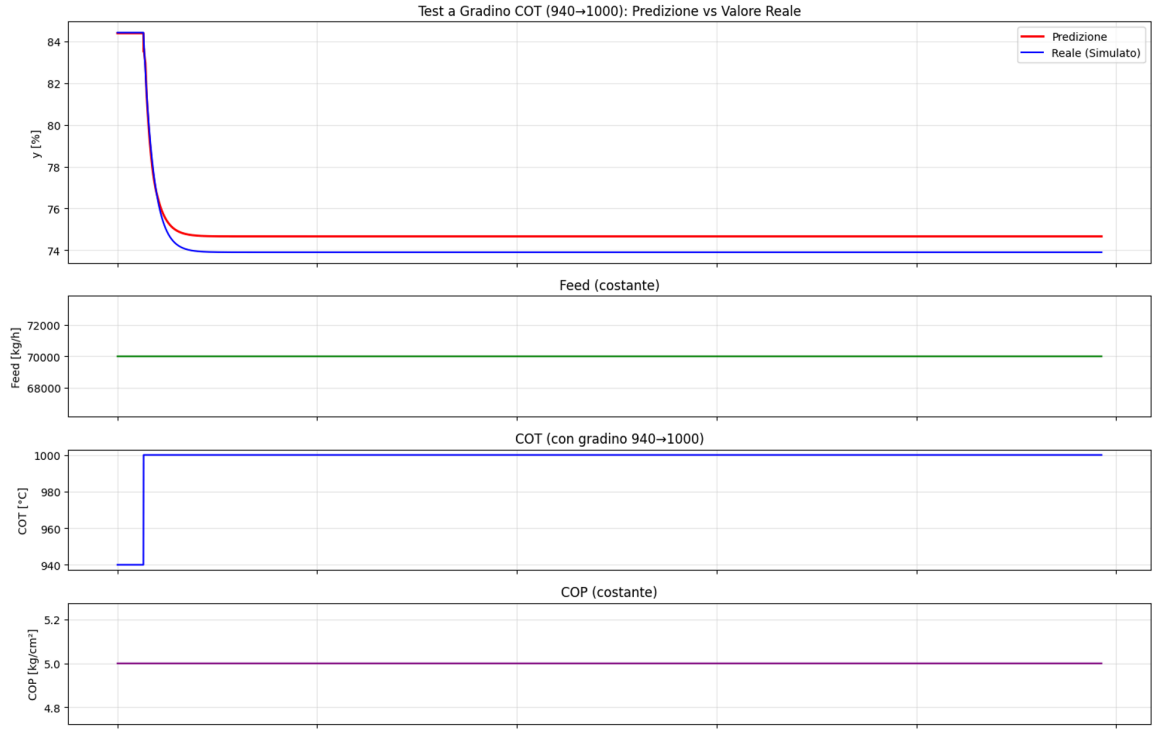


Figura 3.34: risultati relativi all'utilizzo di valori predetti della conversione di etano in input all'*encoder* del modello per step a gradino della COT da 940 a 1000.

Per quanto riguarda la variabile COT, si osserva che nel *range* ottimale, step test tra 830 e 890 °C (figura 3.33), il modello fornisce una previsione molto accurata, con scostamenti minimi rispetto al riferimento. Anche nel caso di una variazione al di fuori del *range* ottimale (figura 3.34), con COT compresa tra 940 e 1000 °C, il modello riesce comunque a riprodurre correttamente l'andamento della variabile d'interesse, pur senza raggiungere il valore stazionario atteso del 74%. Il guadagno associato alla variabile COT mostra un comportamento fortemente non lineare: nel primo test assume un valore positivo di circa +0,33, mentre nel secondo diventa negativo, intorno a -0.16. Nonostante questa marcata variazione di segno e valore assoluto, il nostro modello è in grado di prevedere con buona affidabilità l'andamento del sistema, catturando correttamente la transizione non lineare.

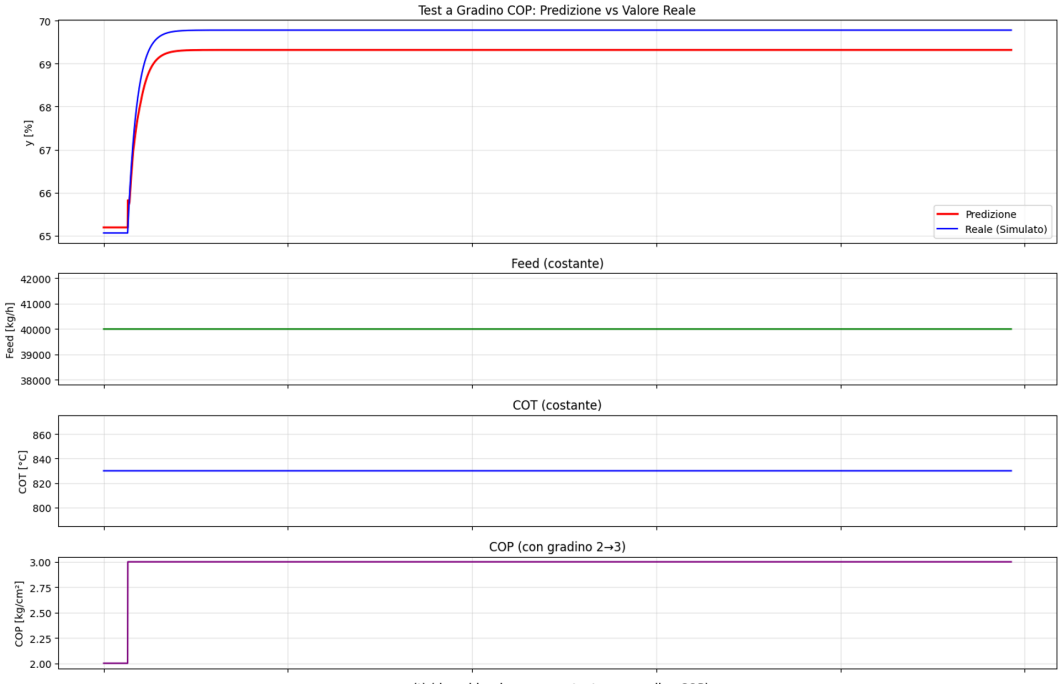


Figura 3.35: risultati relativi all'utilizzo di valori predetti della conversione di etano in input all'*encoder* del modello per step a gradino della COP da 2 a 3.

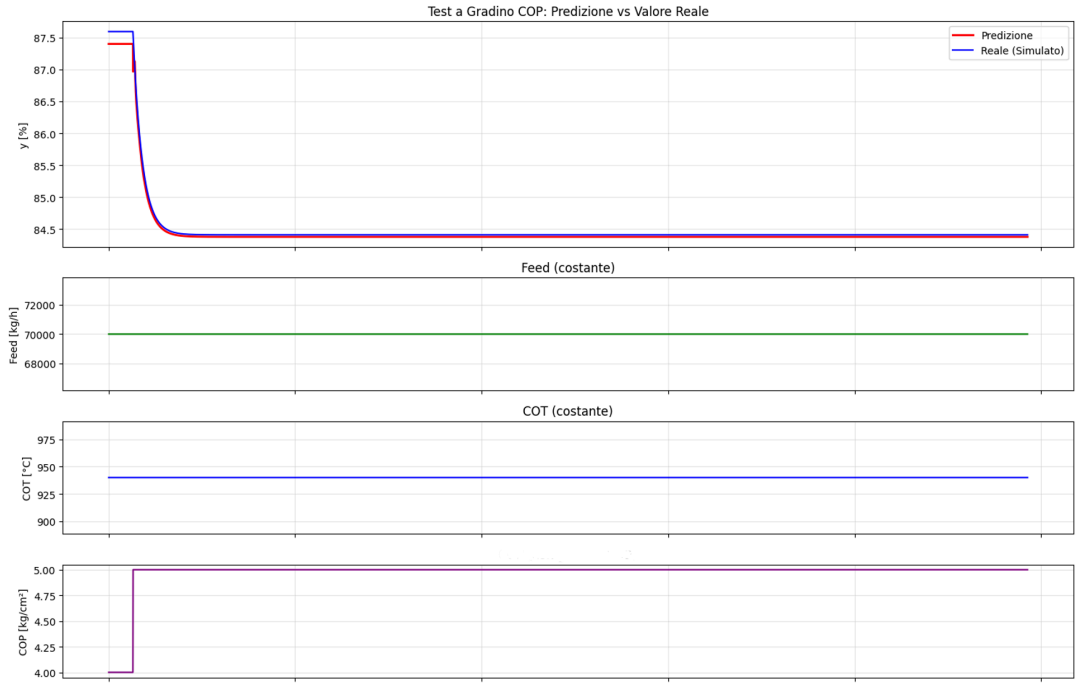


Figura 3.36: risultati relativi all'utilizzo di valori predetti della conversione di etano in input all'*encoder* del modello per step a gradino della COP da 4 a 5.

Per quanto riguarda la variabile COP, si osserva che nel test a gradino da 2 a 3 kg/cm², la previsione è buona, ma non perfettamente accurata: il modello si avvicina allo stazionario atteso del 70%, ma si stabilizza leggermente sotto, attorno al 69%. Al contrario, nel caso dello step fuori *range* da 4 a 5 kg/cm², la risposta predetta è perfettamente allineata al comportamento atteso, raggiungendo correttamente lo stazionario intorno all' 84,5%. Il guadagno associato alla variabile COP evidenzia anch'esso una forte non linearità: nel primo test assume un valore positivo di circa +4,4, mentre nel secondo diventa negativo, intorno a −3,0. Nonostante questa marcata variazione di segno e valore assoluto, il nostro modello riesce a prevedere con buona affidabilità la risposta del sistema, rappresentando correttamente la transizione non lineare osservata.

Capitolo 4

Conclusioni

In questo lavoro di Tesi è stata dimostrata l'efficacia delle architetture *transformer* nell'identificazione della dinamica di processi industriali non lineari. L'utilizzo di modelli avanzati di *deep learning* si è rivelato particolarmente vantaggioso rispetto ai metodi lineari commerciali, evidenziando un miglioramento significativo nella qualità delle previsioni. Nel primo caso studio, relativo alla colonna di separazione C3 *splitter*, sistema MISO 2x1, il modello ha mostrato ottime capacità predittive sia su *dataset* con andamento controllato e regolare, sia su *dataset* casuali. L'architettura *transformer* è stata in grado di adattarsi efficacemente a differenti regimi operativi, modellando con precisione la dinamica del sistema anche in presenza di variazioni non lineari del guadagno e introducendo soluzioni pratiche per la gestione del ritardo di processo. Nel secondo caso, riferito a un forno per lo *steam cracking* dell'etano (sistema MISO 3x1), il *transformer* ha affrontato una sfida più complessa dovuta al maggior numero di variabili manipolate. Anche in questo contesto ha evidenziato una notevole capacità di previsione, sia su dati di sintesi costruiti per simulare scenari realistici, sia in test specifici su singole variabili. L'analisi ha incluso anche casi con *range* operativi amplificati, dove il modello ha dimostrato di saper cogliere e apprendere dinamiche non lineari più marcate, mantenendo comunque ottime performance ($R^2 > 0.95$). Un aspetto rilevante è la versatilità dell'architettura nel passare da modalità *teacher forcing* a modalità completamente autoregressiva. Infine, i risultati ottenuti indicano che i *transformer* rappresentano una soluzione promettente e scalabile per l'identificazione e la previsione di processi industriali complessi, soprattutto in assenza di analizzatori in tempo reale. Un naturale sviluppo di questo lavoro potrebbe consistere nell'applicazione del modello a dati reali ottenuti da *step test* effettuati direttamente in impianto, in condizioni operative reali. In tal contesto, sarebbe fondamentale valutare la robustezza del modello nei confronti del rumore di misura, tipico dei sensori industriali, e verificare la sua capacità di generalizzare anche in presenza di disturbi. Questa validazione su campo rappresenterebbe un passaggio cruciale per l'adozione del modello in ambito produttivo e per il suo eventuale impiego all'interno di strategie di controllo avanzato.

Bibliografia

[1] Niranjan Sitapure, Joseph Sang-Il Kwon (2023). Exploring the potential of time-series transformers for process modeling and control in chemical systems: An inevitable paradigm shift? Chemical Engineering Research and Design, volume 194, pag 461-477 https://doi.org/10.1016/j.cherd.2023.04.028

[2] Ankur Kumar, Jesus Flores-Cerrillo (2023). Machine Learning in Python for Dynamic Process Systems, MLforPSE.

[3] Pieter P. Plehiers,Steffen H. Symoens,Ismaël Amghizar,Guy B. Marin,Christian V. Stevens, Kevin M. Van Geem 2019. Artificial Intelligence in Steam Cracking Modeling: A Deep Learning Algorithm for Detailed Effluent Prediction. Engineering volume 5(6), pag 1027-1040. https://doi.org/10.1016/j.eng.2019.02.013

[4] Tesi Lazzini Matteo (2025). Applicazione dei transformers nella Predizione delle Proprietà dei prodotti: Un nuovo Paradigma per il Controllo Avanzato dei processi, Politecnico di Torino.

[5] Pena Daniel, Tiao George C., Tsay Ruey S (2001). A Course in Time Series Analysis, Wiley. Universidad Carlos III de Madrid, Spain and University of Chicago, United States.

[6] Maheshwari, U. et al. (2011). Optimization of Propylene-Propane Distillation Process using Aspen Plus. Presented at CHEMCON 2011 – Annual Indian Chemical Engineering Congress, Birla Institute of Technology and Science, Pilani, India.

[7] Emerson (n.d.). Online Process Gas Analysis in Ethylene Production Plants. https://www.emerson.com/documents/automation/application-note-online-process-gas-analysis-in-ethylene-production-plants-rosemount-en-71816.pdf

[8] Daison Y. Caballero, Lorenz T. Biegler, Reginaldo Guirardello (2015). Simulation and Optimization of the Ethane Cracking Process to Produce Ethylene. Elviser, volume 37, pag 917-922 https://doi.org/10.1016/B978-0-444-63578-5.50148-1.

[9] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Cap 6-7

[10] Sharma, S., Sharma, S., & Athaiya, A. (2020). Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology, 4(12), 310-316.

[11] Kingma, Diederik P.(2014) Adam: A method for stochastic optimization.

[12] Prechelt, L. (1998). Early Stopping - But When? In: Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524, pag 55-69. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-49430-8_3

[13] Z. C. Lipton, J. Berkowitz, and C. Elkan (2015). A critical review of recurrent neural networks for sequence learning, Cornell university. articolo online: arXiv:1506.00019v4.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 31. https://arxiv.org/abs/1706.03762

[15] S. Karita *et al*. (2019). A Comparative Study on Transformer vs RNN in Speech Applications, IEEE Automatic Speech Recognition and Understanding Workshop *(ASRU)*, Singapore, 2019, pp. 449-456, doi: 10.1109/ASRU46091.2019.9003750.

[16] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, *35*(12), 11106-11115. https://doi.org/10.1609/aaai.v35i12.17325

[17] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, Xifeng Yan (2020). Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. University of California, Santa Barbara. Articolo online: arXiv:1907.00235v3

[18] Molino, P., Wang, Y., & Zhang, J. (2019). Parallax: Visualizing and understanding the semantics of embedding spaces via algebraic formulae. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 165–180), Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-3028

[19] Philipp Dufter, Martin Schmitt, Hinrich Schütze; Position Information in Transformers: An Overview. Computational Linguistics 2022; 48 (3): 733–763. https://doi.org/10.1162/coli_a_00445

[20] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting https://doi.org/10.48550/arXiv.2012.07436

[21] Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting https://doi.org/10.48550/arXiv.1912.09363

[22] Wu, H., Xu, J., Wang, J., & Long, M. (2021). *Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting*. https://doi.org/10.48550/arXiv.2106.13008

[23] Ralf-Peter Mundani, Jérôme Frisch, Vasco Varduhn, Ernst Rank (2015). sliding window technique for interactive high-performance computing scenarios https://doi.org/10.1016/j.advengsoft.2015.02.003

[24] E. Demir and A. E. Tümer (2022). Spatio-temporal wind speed forecasting using graph networks and novel transformer architectures, in *Proc. 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Naples, Italy, pp. 425–430. https://doi.org/10.1109/RO-MAN53752.2022.9900743

[25] Manuale funzionamento modello commerciale Honeywell

[26] Chicco, Davide, Matthijs J. Warrens, and Giuseppe Jurman, (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. Peerj computer science 7, e623. https://doi.org/10.7717/peerj-cs.623

Ringraziamenti

Desidero esprimere la mia più profonda gratitudine a tutte le persone che hanno reso possibile la realizzazione di questo lavoro.

Un grazie speciale va alla mia famiglia, che con amore e pazienza mi ha sempre sostenuto, incoraggiandomi a perseguire i miei obiettivi anche nei momenti di maggiore preoccupazione e difficoltà. In particolare, mia sorella è stata una presenza costante e insostituibile, sempre pronta a offrirmi affetto e sostegno nei momenti in cui ne avevo più bisogno.

Un ringraziamento sentito va al mio relatore, Prof. Davide Fissore, per la disponibilità, la competenza e la guida con cui ha accompagnato l'intero percorso di ricerca e la stesura di questa tesi. Un pensiero di riconoscenza va anche all'azienda Alpha Process Control, che mi ha accolto offrendomi un ambiente stimolante e professionale, trasformando questa esperienza non solo in un importante traguardo accademico, ma anche in una significativa occasione di crescita personale e lavorativa.

Sono profondamente grato anche ai miei amici, di lunga data e più recenti, che hanno contribuito a rendere questo cammino più ricco e significativo. Un grazie particolare a Lorenzo, Stefano e Matteo, amici sinceri con i quali ho condiviso tanto e che non hanno mai fatto mancare la loro presenza. Un grazie sincero anche ad Andrea e Livio, amici che mi accompagnano fin dai tempi del liceo e con cui ho condiviso gioie e preoccupazioni, rendendo questo percorso ancora più prezioso.

Infine, voglio ringraziare i miei colleghi di lavoro e i compagni di università: con loro ho vissuto momenti di confronto, collaborazione e crescita che hanno reso questo percorso ancora più stimolante e arricchente.