



SAPIENZA
UNIVERSITÀ DI ROMA

Machine Learning per Segnali Sensoristici: Classificazione di Serie Temporalì in Ambienti Reali

Machine Learning for Sensor Signals:
Time Series Classification in Real-World Environments

Facoltà di Scienze Matematiche Fisiche e Naturali
Laurea Triennale in Scienze Matematiche per l'Intelligenza Artificiale (L-35)

Michele Magrini
Matricola 2066963

Relatore
Marco Sciandrone

Anno Accademico 2024/2025

Sommario

Questa tesi affronta il problema della classificazione automatica di segnali sensoristici multivariati, raccolti da dispositivi sviluppati da *Sensichips S.r.l.*, per identificare 17 diverse sostanze chimiche. Sono stati esplorati quattro approcci distinti: classificazione su singoli istanti temporali, su finestre temporali, tramite rappresentazione compressa con *Radial Basis Functions* (RBF) e mediante elaborazione sequenziale continua con reti neurali ricorrenti (RNN). Per ciascun approccio, sono stati progettati e confrontati modelli neurali o modelli classici (Random Forest), valutando il compromesso tra accuratezza, dimensione in memoria e robustezza. I risultati sperimentali mostrano che l'approccio a finestre temporali offre il miglior equilibrio tra prestazioni e complessità, mentre le RNN rappresentano un'alternativa promettente per l'elaborazione temporale continua. L'intero codice, comprensivo di preprocessing, addestramento e analisi, è disponibile pubblicamente per garantire la riproducibilità e favorire estensioni future.

This thesis addresses the problem of automatic classification of multivariate sensor signals, collected by devices developed by Sensichips S.r.l., for the identification of 17 different chemical substances. Four distinct approaches were explored: classification at individual time instants, within fixed-length time windows, using compact representations via Radial Basis Functions (RBF), and through continuous sequential processing with recurrent neural networks (RNN). For each approach, both neural and classical models (Random Forest) were designed and compared, evaluating the trade-off between accuracy, memory footprint, and robustness. Experimental results show that the time window approach provides the best balance between performance and complexity, while RNNs emerge as a promising alternative for continuous temporal processing. The full codebase, including preprocessing, training, and evaluation scripts, is publicly available to ensure reproducibility and support future extensions.

An English version of this thesis is available at the following link:

https://github.com/mich1803/sensor-ts-classification/blob/main/docs/report_eng.pdf

Machine Learning per Segnali Sensoristici: Classificazione di Serie Temporali in Ambienti Reali

Sapienza Università di Roma – Facoltà di Scienze Matematiche Fisiche e Naturali

Autore: Michele Magrini (magrini.2066963@studenti.uniroma1.it)

Relatore: Marco Sciandrone (marco.sciandrone@uniroma1.it)

Sensichips S.R.L. (sensichips.com)

Indice

Introduzione	1
1 Il Dataset Sensoristico SCA	3
1.1 Descrizione generale del dataset	3
1.2 Modalità di acquisizione dei dati	3
1.3 Filtraggio e preparazione iniziale	4
1.4 Esplorazione statistica e visiva del dataset	5
1.5 Considerazioni sul contesto reale	7
2 Classificazione su Singoli Istanti Temporal	8
2.1 Multilayer Perceptron	8
2.1.1 Funzionamento e apprendimento del modello	9
2.1.2 Il Common MLP Block	10
2.2 Encoder MLP	10
2.3 Impostazioni comuni ai modelli	12
2.4 Risultati e osservazioni	12
2.4.1 Definizione delle Metriche	12
3 Classificazione con Finestre Temporal	15
3.1 Struttura del dato: finestre temporal	15
3.2 Column Selective Encoder MLP	16
3.3 Reti Neurali Convoluzionali	17
3.4 Risultati e osservazioni	18
4 Analisi Temporale delle Predizioni	22
4.1 Motivazione e utilità dell'esperimento	22
4.2 Risultati e osservazioni	22
5 Esperimenti di Classificazione Binaria	24
5.1 Classificazione binaria su singoli istanti temporal	25
5.2 Classificazione binaria su finestre temporal	25
5.3 Osservazioni	26
6 Classificazione con Interpolazione RBF	27
6.1 Descrizione dell'approccio	27
6.2 Struttura del dato in ingresso	27
6.3 Random Forest	28

6.4	Risultati e osservazioni	29
7	Classificazione con Elaborazione Temporale Continua	30
7.1	Recurrent Neural Networks	30
7.2	Reti Gated Recurrent Unit	31
7.3	Reti Long Short-Term Memory	31
7.4	Classificazione di Serie Temporali Complete e di Lunghezza Variabile	32
7.4.1	Risultati e Osservazioni	32
7.5	Applicazione dell'RNN alle Finestre Temporali	33
7.5.1	Risultati e Osservazioni	33
8	Discussione Comparativa dei Modelli	35
	Conclusioni	38
	Bibliografia	39

Elenco delle figure

1.1	Distribuzione degli esperimenti (a sinistra) e dei sample (a destra) divisi per etichetta e colorati in base agli split di train, validation o test.	5
1.2	Analisi visiva e statistica del dataset. In alto a sinistra: matrice di correlazione tra le 16 feature. In alto a destra: proiezione 3D dei dati attraverso PCA. In basso: contributo assoluto delle feature alla formazione delle prime tre componenti principali (PC1, PC2, PC3).	6
2.1	Confronto tra MLP semplice (sinistra) e Encoder MLP (destra). Il secondo introduce uno spazio latente (verde) prima del <i>Common MLP Block</i> .	11
2.2	Learning curve del modello MLP sul singolo istante durante l'addestramento: andamento della loss e dell'accuracy su training e validation.	13
2.3	Matrici di confusione sul test set per il modello MLP sul singolo istante. A sinistra: classificazione <i>sample-level</i> , a destra: classificazione <i>experiment-level</i> .	14
3.1	Esempio di rappresentazione di un campione da finestra temporale della dimensione 100×16 (time steps $\times$ feature). Si notano chiaramente differenze strutturali tra le colonne, indicative della risposta eterogenea delle diverse feature al segnale chimico.	15
3.2	Architettura del modello CSEMLP . Il segnale è elaborato da un blocco selettivo per colonna (CS) e un blocco fully connected (L), prima di essere processato dal <i>Common MLP Block</i> .	17
3.3	Architettura del modello CNN1D . La sequenza temporale 100×16 viene trasposta in un tensore 16×100 e processata con convoluzioni 1D lungo l'asse temporale. L'output viene poi appiattito e passato al <i>Common MLP Block</i> .	17
3.4	Learning curve del modello EMLPs sulle finestre temporali durante l'addestramento: andamento della loss e dell'accuracy su training e validation.	20
3.5	Matrici di confusione sul test set per il modello EMLPs sulle finestre temporali. A sinistra: classificazione <i>sample-level</i> , a destra: classificazione <i>experiment-level</i> .	20

4.1	Distribuzione temporale delle predizioni corrette nei due approcci. A sinistra: MLP su singoli istanti temporali; a destra: EMLP su finestre temporali. Ogni riga rappresenta un esperimento, ordinato per classe chimica.	23
6.1	Esempio di approssimazione tramite RBF ($m = 5$) sulla feature SnAu_150_200.0_IN-PHASE . Le linee tratteggiate rappresentano le curve originali di alcuni esperimenti con AMMONIUM_CHLORIDE , mentre le linee continue mostrano il fit ottenuto con le basi radiali.	28
7.1	Architettura di una generica RNN. Il vettore x_t si collega all'hidden state h_t (analogo all'hidden layer che si trovava nell'MLP Block nei capitoli precedenti) che a sua volta è collegato a h_{t-1} e h_{t+1} . Infine l'output layer comunica solo con l'ultimo hidden state h_N	30
7.2	Curve di apprendimento del modello RNN GRU su Serie Temporali di lunghezza variabile.	33
7.3	Risultati dell'applicazione dell'RNN GRU sull'approccio a finestre temporali. A sinistra le confusion matrix Sample-Level ed Experiment-level, a destra le distribuzioni temporali associate (analoghe alla figura 4.1).	34
8.1	Relazione tra accuratezza a livello di esperimento e dimensione in memoria dei modelli.	36

Elenco delle tabelle

2.1	Prestazioni dei modelli nell'approccio ad istante temporale.	13
3.1	Prestazioni dei modelli nell'approccio a finestre temporali, riportate in termini di accuratezza sui singoli campioni (S_{acc}) e a livello di esperimento (E_{acc}), come definito nella Sezione 2.4.1.	18
5.1	Prestazioni medie e per fold del modello MLP su task binario tra ETHANOL e ACETONE nell'approccio a istanti temporali.	25
5.2	Prestazioni medie e per fold del modello EMLPs su task binario tra ETHANOL e ACETONE nell'approccio a finestre temporali.	25
6.1	Performance dei diversi modelli nell'approccio RBF.	29
7.1	Prestazioni dei modelli GRU e LSTM sull'intera sequenza.	32

Introduzione

Negli ultimi decenni, il *Machine Learning* (ML) si è affermato come uno strumento fondamentale per analizzare e interpretare grandi quantità di dati eterogenei. La classificazione automatica di segnali raccolti da sensori ha assunto particolare rilevanza tra le molteplici applicazioni di tecniche di ML, un compito complesso ma cruciale in contesti industriali⁽⁴⁾, ambientali⁽³⁾ e biomedici^(6;10).

La presente tesi si concentra sullo sviluppo e la valutazione di modelli di apprendimento automatico applicati a serie temporali multivariate ottenute da sensori chimici. I dati utilizzati provengono da una collaborazione con l'azienda *Sensichips S.r.l.*, che ha fornito una collezione di acquisizioni sperimentali sottoforma di matrici condotte in condizioni controllate e in ambiente reale.

Ciascun esperimento consiste in una sequenza temporale di misure, organizzate in tabelle con 16 colonne, corrispondenti a differenti caratteristiche elettriche rilevate da due sensori distinti, operanti a frequenze e modalità diverse. Ogni esperimento è associato a una sola sostanza chimica tra 17 possibili, che costituisce l'etichetta di classificazione.

L'obiettivo del lavoro è sviluppare algoritmi in grado di apprendere dai dati storici e di prevedere correttamente, a partire da una nuova sequenza di misure, quale sostanza sia presente. Si tratta di un problema di classificazione supervisionata su serie temporali, complicato dalla natura rumorosa e variabile dei dati raccolti in ambienti reali, nonché dalla variabilità nella durata e struttura delle acquisizioni.

Per affrontare questa sfida, sono stati progettati e implementati quattro approcci distinti, ciascuno con un diverso grado di sfruttamento dell'informazione temporale e della struttura dei dati.

- **Approccio all'istante temporale:** tratta ogni singola riga delle misurazioni come un campione indipendente, rappresentato da un vettore di 16 dimensioni. È una strategia semplice che ignora la dipendenza temporale tra osservazioni consecutive.
- **Approccio a finestre temporali:** considera sottosequenze (finestre) di misure contigue nel tempo come unità di input. Ogni finestra, di dimensione fissa $N \times 16$, permette al modello di apprendere anche la dinamica locale del segnale.
- **Approccio RBF:** ogni colonna (serie temporale) viene approssimata mediante una combinazione di funzioni di base radiali, ottenendo una rappresentazione compatta dell'intero esperimento. Tale rappresentazione riduce la dimensionalità e consente una classificazione più stabile.

- **Approccio con elaborazione temporale continua:** utilizza modelli ricorrenti per processare l'intera sequenza temporale di ciascun esperimento, senza suddividerla in finestre. Questo approccio permette al modello di apprendere rappresentazioni globali sfruttando la dipendenza temporale completa tra i dati.

Tutti gli approcci sono stati valutati secondo la capacità di classificare correttamente l'intera serie temporale. Se un modello non analizza l'intera serie temporale allora la predizione finale è decisa attraverso un voto di maggioranza sulle predizioni parziali.

Il contributo di questa tesi è duplice: da un lato l'implementazione e il confronto sistematico tra approcci differenti, dall'altro la riflessione critica su come il contesto reale influenzi le scelte modellistiche e i risultati.

Capitolo 1

Il Dataset Sensoristico SCA

Il presente capitolo è dedicato alla descrizione del dataset utilizzato per l'addestramento e la valutazione dei modelli di classificazione. Questo insieme di dati, fornito dall'azienda *Sensichips S.r.l.*, rappresenta una collezione eterogenea di esperimenti condotti in condizioni reali e controllati mediante sensori chimici avanzati. La comprensione della struttura e delle modalità di acquisizione di questi dati è essenziale per motivare le scelte modellistiche esposte nei capitoli successivi.

1.1 Descrizione generale del dataset

Il dataset, denominato **SCA Sensor Dataset**, è composto da numerose tabelle, ognuna delle quali rappresenta un **esperimento** individuale. Ogni esperimento consiste in una sequenza temporale di misure raccolte durante l'esposizione dei sensori a una specifica sostanza chimica. Complessivamente, sono presenti dati relativi a 17 classi di sostanze, tra cui etanolo, acetone, ammoniaca e altre sostanze volatili comunemente utilizzate come target per sensori ambientali.

Ciascuna tabella contiene 16 colonne di feature, corrispondenti a misurazioni elettriche acquisite a differenti frequenze e modalità operative. Il numero di righe di ciascun file è variabile e rappresenta l'evoluzione temporale del segnale nel tempo: ogni riga può essere vista come una singola acquisizione istantanea di tutte le feature. Le etichette (labels) non variano riga per riga, ma ogni file è interamente associato a una sola classe, ovvero una sola sostanza.

1.2 Modalità di acquisizione dei dati

I dati sono stati acquisiti utilizzando il sensore *SCA*⁽²⁾, progettato da Sensichips, e basato su due diversi componenti sensibili:

- **Sensore SnO₂+Au su hotplate:** un sensore a base di ossido di stagno funzionalizzato con nanoparticelle d'oro. Questo sensore è stato operato secondo la tecnica **Temperature Cycled Operation (TCO)**, ovvero riscaldato a sei diversi livelli di temperatura compresi tra 150°C e 400°C. Per ogni livello, sono state acquisite due misure (IN-PHASE) a 78.125 Hz e 200 Hz.

- **Sensore Al_2O_3 su interdigit:** un sensore su allumina anodizzata stimolato elettricamente a due diversi livelli di tensione. Qui viene impiegata la tecnica **Voltage Cycled Operation (VCO)** e vengono registrate sia le componenti *in-phase* che *quadrature*, ma solo a 78.125 Hz.

Le misurazioni sono state eseguite in ambienti chiusi, dove il composto chimico veniva lasciato evaporare in modo lento e controllato, oppure in ambienti aperti con turbolenza ambientale.

1.3 Filtraggio e preparazione iniziale

Per garantire una qualità minima dei dati utilizzati nei modelli di classificazione, è stato necessario applicare una serie di criteri di filtraggio. In particolare, sono stati considerati validi solo gli esperimenti che soddisfacevano entrambe le seguenti condizioni:

- Ciascun esperimento doveva contenere almeno **200 righe** (ossia almeno 200 misurazioni temporali), al fine di permettere l'applicazione dell'approccio a finestre temporali con sufficiente scorrimento.
- Ogni classe (etichetta) doveva essere rappresentata da almeno **10 esperimenti validi**, per assicurare un minimo di varietà e significatività statistica all'interno di ciascun gruppo.

Suddivisione del dataset: train, validation e test

Un aspetto fondamentale del processo di preparazione è la suddivisione del dataset in tre sottoinsiemi: **train** (addestramento), **validation** (validazione) e **test** (valutazione finale). Questo step ha lo scopo di separare le fasi di apprendimento, ottimizzazione degli iperparametri e verifica delle prestazioni generali del modello.

Nel caso specifico del dataset fornito da Sensichips, la divisione è stata effettuata a livello di esperimento: tutte le righe appartenenti a un dato esperimento sono sempre assegnate allo stesso split. Questa scelta è cruciale per evitare che informazioni temporali o caratteristiche specifiche dell'esperimento stesso vengano condivise tra training e test, con conseguente rischio di *data leakage* e sovrastima delle prestazioni.

La suddivisione è avvenuta secondo le seguenti proporzioni, calcolate separatamente per ogni etichetta:

- **Validation set:** 10% degli esperimenti disponibili per ciascuna classe, con un minimo di 2 esperimenti per etichetta.
- **Test set:** 10% degli esperimenti per ciascuna classe, anche in questo caso con un minimo di 2 esperimenti per etichetta.
- **Train set:** tutti gli esperimenti restanti, utilizzati per l'apprendimento effettivo del modello.

Tale strategia di suddivisione garantisce una rappresentazione bilanciata e coerente delle classi in ogni fase del processo, rispettando allo stesso tempo la struttura intrinseca dei dati.

È importante sottolineare che tutti i modelli multiclasse sviluppati nel corso di questo lavoro sono stati addestrati e valutati utilizzando esattamente la stessa suddivisione del dataset. Questa coerenza nello split dei dati assicura che le prestazioni dei diversi modelli siano confrontabili in modo equo, eliminando variabili legate a differenti partizioni dei dati e garantendo la conformità dei risultati ottenuti.

Successivamente, le feature numeriche sono state normalizzate, ovvero centrate rispetto alla media e scalate in base alla deviazione standard, entrambe calcolate esclusivamente sul set di addestramento. Questa operazione ha lo scopo di armonizzare le scale delle diverse feature e migliorare la stabilità numerica degli algoritmi di apprendimento.

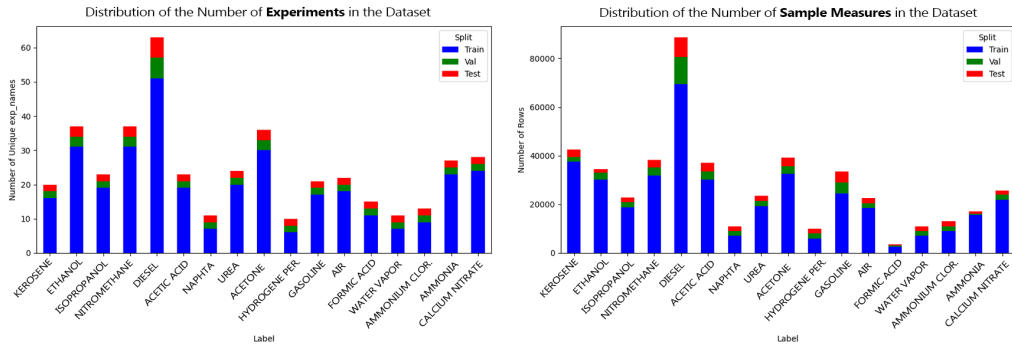


Figura 1.1 Distribuzione degli esperimenti (a sinistra) e dei sample (a destra) divisi per etichetta e colorati in base agli split di train, validation o test.

1.4 Esplorazione statistica e visiva del dataset

L'esplorazione visiva e statistica dei dati è un passaggio fondamentale per comprenderne la struttura intrinseca e guidare eventuali scelte di riduzione dimensionale o normalizzazione. Tutti i grafici riportati in Figura 1.2 condividono lo stesso ordinamento delle feature, che sono disposte come segue:

- Le prime 6: misure acquisite dal sensore **SnO₂+Au** a **200Hz**, con temperature dai 150°C ai 400°C.
- Le successive 6 (7–12): misure dello stesso sensore a **78.125Hz**, con le stesse temperature.
- Le ultime 4 (13–16): misure del sensore **Al₂O₃** registrate a 78.125Hz, suddivise tra componenti *in-phase* e *quadrature*.

La prima visualizzazione (Fig 1.2 in alto a sinistra) mostra la **matrice di correlazione** tra le feature calcolata a livello di sample. È evidente che le misure del sensore SnO₂+Au a diverse temperature risultano fortemente correlate tra loro,

suggerendo che questi canali misurano lo stesso fenomeno ma con sensibilità diverse. Analogamente, anche le quattro feature relative al sensore Al_2O_3 mostrano alta correlazione tra loro.

Questa ridondanza tra feature motiva l'uso di una tecnica di **riduzione dimensionale**, come la *Principal Component Analysis* (PCA). La PCA è una trasformazione lineare che proietta i dati originali in uno spazio a dimensionalità ridotta, massimizzando la varianza lungo le nuove direzioni principali (componenti principali). Le componenti sono ortogonali tra loro e costruite come combinazioni lineari delle feature originarie.

Nel **plot 3D della PCA** (Fig. 1.2 in alto a destra) ogni punto rappresenta una finestra temporale proiettata nei primi tre assi principali. I dati formano delle “curve” distinte per ciascuna sostanza, che sembrano facilmente separabili nel nuovo spazio, indicando che la PCA conserva bene la struttura discriminativa tra le classi.

Infine, la parte inferiore della figura 1.2 illustra il contributo assoluto delle feature per ciascuna delle prime tre componenti (PC1, PC2 e PC3). Questi grafici sono stati prodotti estraendo i coefficienti della matrice dei **loadings** della PCA. In particolare, se $\mathbf{W} \in \mathbb{R}^{d \times 3}$ è la matrice di trasformazione, ogni colonna rappresenta una combinazione delle $d = 16$ feature originali.

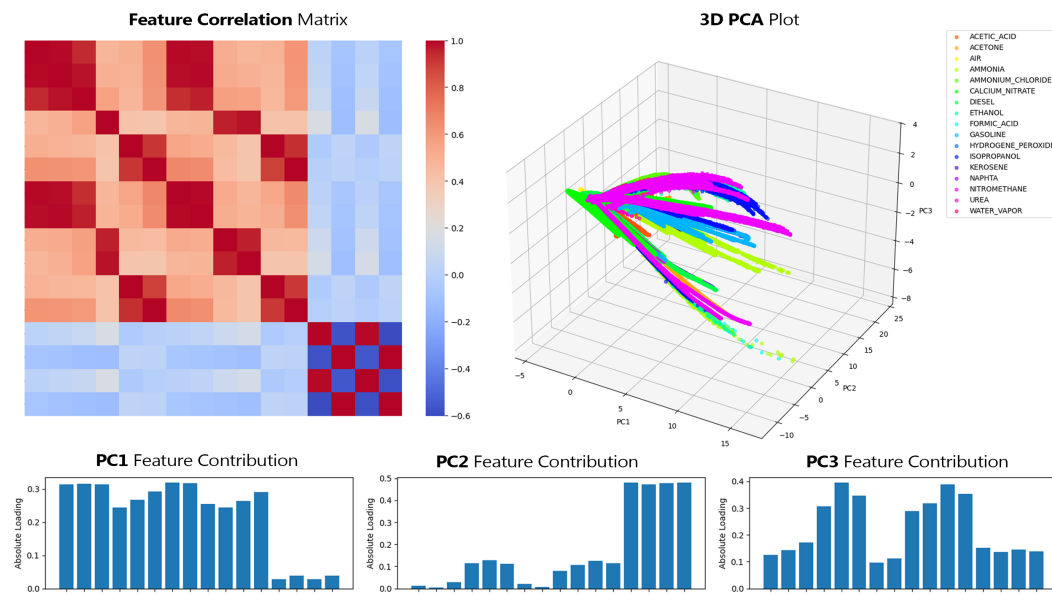


Figura 1.2 Analisi visiva e statistica del dataset. In alto a sinistra: matrice di correlazione tra le 16 feature. In alto a destra: proiezione 3D dei dati attraverso PCA. In basso: contributo assoluto delle feature alla formazione delle prime tre componenti principali (PC1, PC2, PC3).

Dai plot si osserva che:

- La **PC1** è dominata dalle feature del sensore SnO_2+Au (in particolare quelle a 200Hz).
- La **PC2** è principalmente influenzata dalle misure del sensore Al_2O_3 .
- La **PC3** presenta un mix di contributi significativi da entrambi i sensori.

Questa analisi conferma che gran parte dell'informazione discriminativa è contenuta in poche componenti principali. Ciò suggerisce l'opportunità di sfruttare tecniche di compressione e proiezione nello sviluppo di modelli di classificazione leggeri ed efficienti.

1.5 Considerazioni sul contesto reale

Uno degli aspetti distintivi di questo dataset è la sua origine da esperimenti in ambienti reali, non sintetici. Questo introduce un livello di variabilità (ad esempio rumore, fluttuazioni termiche, effetti di contaminazione) che rende il task di classificazione particolarmente interessante dal punto di vista della ricerca. Come evidenziato in studi recenti sulla sensoristica ambientale^(1;7), la robustezza degli algoritmi ML in presenza di rumore reale è uno dei criteri fondamentali per la loro applicabilità industriale.

La presenza di classi chimiche con proprietà simili — come ad esempio etanolo e acetone — costituisce un'ulteriore sfida, poiché richiede che il modello apprenda rappresentazioni sottili ma significative.

Capitolo 2

Classificazione su Singoli Istanti Temporali

In questo capitolo viene analizzato un primo approccio alla classificazione delle serie temporali sensoristiche, che considera ogni istante temporale in maniera indipendente. Tale strategia, sebbene ignori completamente le dipendenze temporali tra misurazioni successive, permette di ridurre drasticamente la complessità del modello e rappresenta un valido punto di partenza per confronti successivi.

Questo tipo di tecnica è simile a quanto proposto in contesti industriali e biomedicali dove ogni misura è trattata come un'osservazione autonoma⁽⁹⁾. In particolare, tale approccio è spesso utilizzato in situazioni in cui non è disponibile una struttura temporale chiara, oppure quando si vuole costruire un modello leggero per classificazioni in tempo reale.

2.1 Multilayer Perceptron

Il primo modello testato in questo approccio è un **Multilayer Perceptron (MLP)**, una rete neurale di tipo *feed-forward* composta da strati lineari intercalati da funzioni di attivazione non lineari. L'MLP rappresenta una delle architetture fondamentali nel deep learning, largamente impiegata in compiti di classificazione e regressione⁽¹²⁾.

Architettura utilizzata

Nel nostro caso, l'input del modello è un vettore $\mathbf{x} \in \mathbb{R}^{16}$, corrispondente a un singolo istante temporale con 16 feature. Il modello è strutturato come segue:

$$\mathbb{R}^{16} \xrightarrow{\text{Linear}} \mathbb{R}^{64} \xrightarrow{\text{ReLU} + \text{Dropout}} \mathbb{R}^{64} \xrightarrow{\text{Linear}} \mathbb{R}^{17}$$

Il blocco finale da 64 a 17 neuroni, utilizzato anche nei modelli successivi, è definito nella Sezione 2.1.2 come *Common MLP Block*.

2.1.1 Funzionamento e apprendimento del modello

Durante l'allenamento, l'MLP cerca di minimizzare una funzione di perdita confrontando le predizioni con le etichette reali. L'ottimizzazione dei pesi avviene tramite una combinazione di due elementi chiave:

1. La **discesa del gradiente**, che aggiorna i pesi nella direzione che riduce la funzione di perdita;
2. L'algoritmo **backpropagation**, che calcola i gradienti attraverso i livelli della rete.

Discesa del gradiente

Data una funzione di perdita $\mathcal{L}(\theta)$, dove θ rappresenta i parametri del modello (pesi e bias), la discesa del gradiente aggiorna i parametri secondo la regola:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} \mathcal{L}(\theta^{(t)})$$

dove η è il *learning rate*, e $\nabla_{\theta} \mathcal{L}$ è il gradiente della funzione di perdita rispetto ai parametri. Questo metodo può essere applicato su tutto il dataset (batch), su piccoli sottoinsiemi (mini-batch), o istante per istante (stocastico).

Backpropagation

Il gradiente $\nabla_{\theta} \mathcal{L}$ è calcolato tramite la tecnica della *backpropagation*, un'applicazione ricorsiva del teorema della catena. Se $z^{(l)} = W^{(l)} h^{(l-1)} + b^{(l)}$ è l'input lineare del layer l e $h^{(l)} = \sigma(z^{(l)})$ è l'output dopo attivazione, allora l'errore del layer l è:

$$\delta^{(l)} = \left(\frac{\partial \mathcal{L}}{\partial h^{(l)}} \right) \cdot \sigma'(z^{(l)})$$

Il gradiente rispetto ai pesi è quindi:

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} (h^{(l-1)})^T$$

Ottimizzazione con Adam

Nel nostro lavoro abbiamo utilizzato l'ottimizzatore **Adam** (Adaptive Moment Estimation), una variante avanzata della discesa del gradiente stocastica che mantiene medie mobili dei gradienti e dei loro quadrati. I passi di aggiornamento sono:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Dove g_t è il gradiente corrente, m_t è la stima del primo momento (media), v_t del secondo momento (varianza), e ϵ è un termine per evitare divisioni per zero.

Cross-Entropy Loss

La funzione di perdita utilizzata è la **Cross-Entropy Loss**, adatta ai problemi di classificazione multiclasse. Se $\mathbf{y} \in \{0, 1\}^C$ è il vettore one-hot dell'etichetta e $\hat{\mathbf{y}} \in [0, 1]^C$ è la distribuzione predetta (dopo softmax), allora la perdita è:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{c=1}^C y_c \log(\hat{y}_c)$$

dove $C = 17$ è il numero di classi. Questa funzione penalizza fortemente predizioni lontane dalla classe corretta e incoraggia probabilità concentrate sulla classe target.

2.1.2 Il Common MLP Block

L'ultima parte di questo MLP, ovvero la transizione da 64 a 17 neuroni, viene definita come **Common MLP Block** e sarà riutilizzata anche nei modelli successivi. Essa consiste in:

- **Dropout** con $p = 0.2$
- **Linear layer** da 64 a 17 neuroni

Questo blocco ha il compito di mappare le rappresentazioni latenti interne del modello verso le probabilità associate a ciascuna delle 17 classi chimiche. L'output finale è elaborato tramite una funzione **softmax**, che assegna una probabilità normalizzata a ciascuna classe.

2.2 Encoder MLP

In aggiunta al semplice MLP descritto nella Sezione 2.1, è stato testato anche un modello leggermente più complesso, che introdurremo come **Encoder MLP** (abbreviato **EMLP**). Questo modello si basa sull'idea di proiettare l'input in uno spazio latente a bassa dimensionalità, dal quale si ottiene poi la predizione finale attraverso il Common MLP Block già definito nella Sezione 2.1.2.

Motivazione

L'introduzione di uno spazio latente serve a comprimere l'informazione contenuta nelle 16 feature originarie in una rappresentazione più compatta, che possa aiutare il modello a generalizzare meglio ed evitare l'overfitting. Questo è particolarmente utile quando il numero di campioni è limitato o quando le feature sono in parte ridondanti.

Struttura del modello

L'architettura complessiva del modello EMLP è composta da due blocchi principali:

1. **Encoder**: un layer lineare che proietta l'input $\mathbf{x} \in \mathbb{R}^{16}$ in uno spazio latente di dimensione d , con $d \in \{4, 8\}$.

2. **Common MLP Block**: come descritto precedentemente, un blocco Dropout + Linear che proietta da 64 a 17 classi.

L'architettura può quindi essere rappresentata come:

$$\mathbb{R}^{16} \xrightarrow{\text{Linear}} \mathbb{R}^d \xrightarrow{\text{ReLU} + \text{Dropout}} \mathbb{R}^d \xrightarrow{\text{Linear}} \mathbb{R}^{64} \xrightarrow{\text{Common MLP Block}} \mathbb{R}^{17}$$

Le due varianti del modello utilizzano:

- **EMLPs (Small)**: dimensione latente $d = 4$
- **EMLPb (Big)**: dimensione latente $d = 8$

Questa distinzione consente di valutare l'effetto della capacità rappresentativa del modello sul trade-off tra accuratezza e generalizzazione.

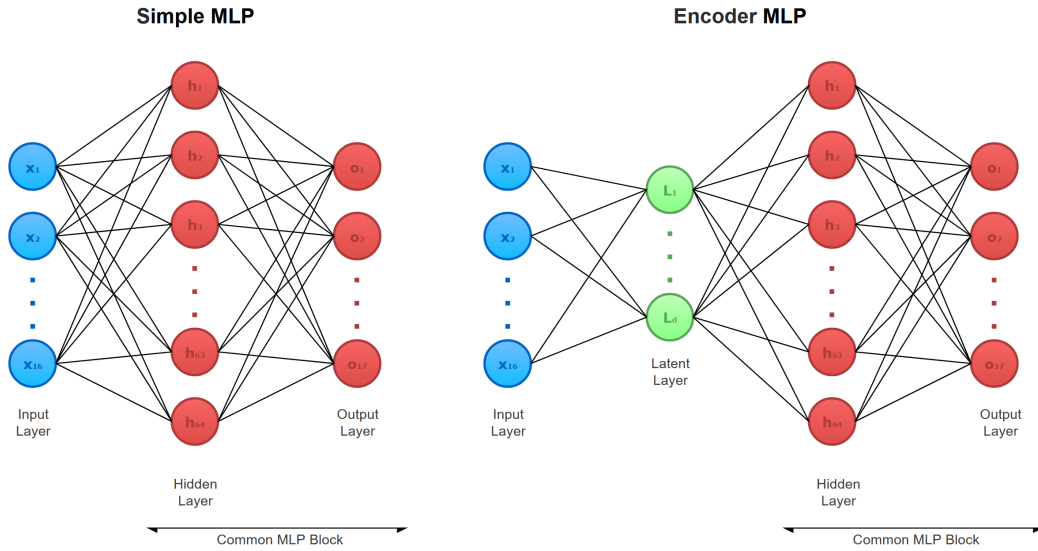


Figura 2.1 Confronto tra **MLP** semplice (sinistra) e **Encoder MLP** (destra). Il secondo introduce uno spazio latente (verde) prima del *Common MLP Block*.

2.3 Impostazioni comuni ai modelli

Tutti i modelli di classificazione multiclasse presentati in questo capitolo (e nei successivi) sono stati addestrati utilizzando le medesime impostazioni iperparametriche, al fine di garantire una valutazione comparabile e rigorosa delle rispettive prestazioni.

- **Learning Rate (LR):** il tasso di apprendimento è stato impostato a 1×10^{-4} . Questo parametro controlla la dimensione dei passi effettuati dall'ottimizzatore nella discesa del gradiente.
- **Weight Decay:** un valore di 1×10^{-5} è stato utilizzato per penalizzare pesi di grandi dimensioni, fungendo da regolarizzazione per evitare l'overfitting.
- **Numero massimo di epoche:** è stato fissato a **50**. Ogni epoca rappresenta un passaggio completo su tutto il dataset di training.
- **Early Stopping:** è stato utilizzato un meccanismo di interruzione anticipata con **patience** pari a 5 (ossia $\#Epoche/10$), bloccando l'addestramento se la perdita di validazione non migliorava per 5 epoche consecutive.
- **Dropout:** è stato introdotto un *dropout* con probabilità $p = 0.2$ per prevenire l'overfitting. Questo meccanismo disattiva casualmente alcune unità durante il training, forzando il modello a non dipendere da singole attivazioni.

Queste impostazioni sono mantenute in tutti gli esperimenti, salvo diversa indicazione.

2.4 Risultati e osservazioni

2.4.1 Definizione delle Metriche

Nel valutare le prestazioni dei modelli, distinguiamo due livelli di accuratezza:

- **Sample-level Accuracy - S_{acc} :** indica la percentuale di istanti temporali (singole righe delle tabelle) classificati correttamente. È calcolata come:

$$S_{acc} = \frac{\# \text{ sample corretti}}{\# \text{ totale sample}}$$

- **Experiment-level Accuracy - E_{acc} :** indica la percentuale di esperimenti (intere tabelle) classificati correttamente tramite *majority vote* sui sample contenuti. Per ogni esperimento viene selezionata la classe più predetta tra tutte le sue righe. La formula è:

$$E_{acc} = \frac{\# \text{ esperimenti corretti}}{\# \text{ totale esperimenti}}$$

Risultati La tabella 2.1 riporta i risultati ottenuti dai modelli MLP e EMLP, in termini di accuratezza sui set di training, validazione e test, sia a livello di sample che a livello di esperimento. Le figure 2.2 e 2.3 ci mostrano le curve di apprendimento e le matrici di confusione di quello che è risultato il miglior modello in questo approccio: l'MLP.

Model	Size (MB)	Epochs	Train S_{acc}	Val. S_{acc}	Test S_{acc}	Train E_{acc}	Val. E_{acc}	Test E_{acc}
MLP	0.009	50	0.750	0.636	0.470	0.799	0.658	0.439
EMLPb	0.007	17	0.469	0.334	0.334	0.475	0.317	0.292
EMLPs	0.006	49	0.438	0.393	0.371	0.478	0.341	0.341

Tabella 2.1 Prestazioni dei modelli nell'approccio ad istante temporale.

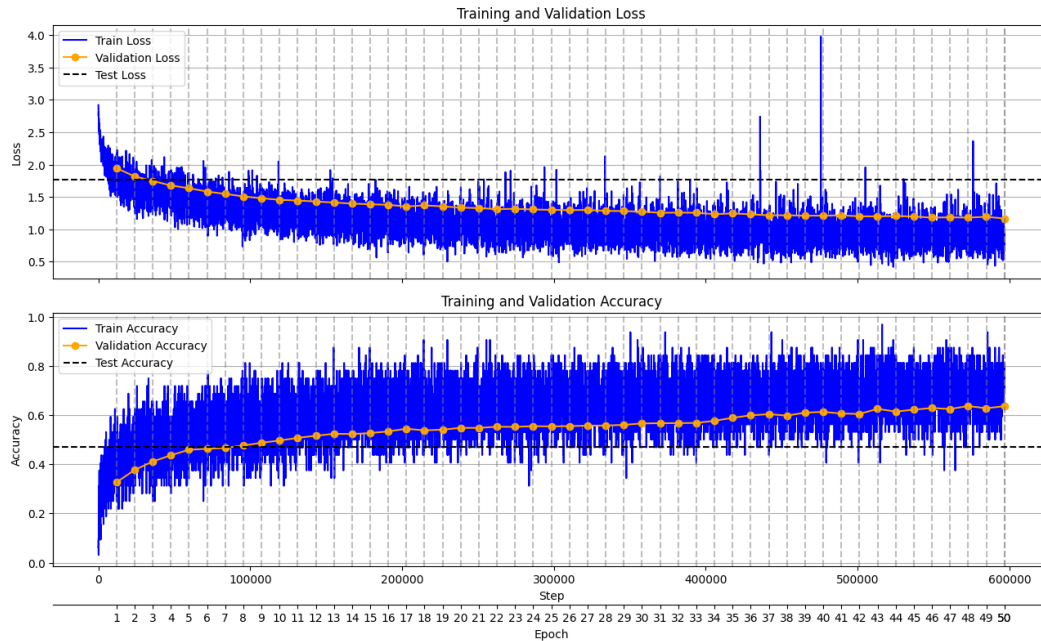


Figura 2.2 Learning curve del modello MLP sul singolo istante durante l'addestramento: andamento della loss e dell'accuracy su training e validation.

Osservazioni Il modello **MLP** ha ottenuto le migliori prestazioni su tutti i fronti, sia in termini di accuratezza (**47.0%** a livello di sample) che di performance a livello di esperimento (**43.9%**). Nonostante sia il modello architetturealmente più semplice, risulta quello più efficace in questo scenario. Questo risultato può essere interpretato come un segnale che, in presenza di input molto compatti (16 feature) e trattati come istanti indipendenti, l'introduzione di uno spazio latente aggiuntivo, come nel

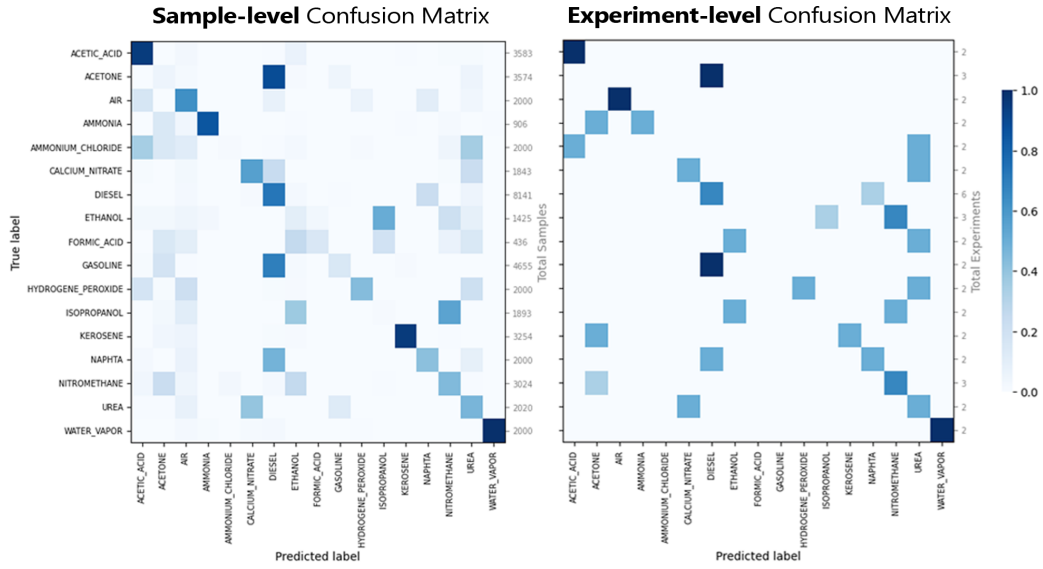


Figura 2.3 Matrici di confusione sul test set per il modello MLP sul singolo istante.

A sinistra: classificazione *sample-level*, a destra: classificazione *experiment-level*.

caso degli EMLP, non solo non apporta beneficio, ma può addirittura degradare le prestazioni.

In particolare, entrambi gli EMLP mostrano forti segni di underfitting: accuracies e performance molto basse sia su training che su test, a indicare che il modello fatica ad apprendere strutture significative dai dati. Ciò può essere imputato sia alla bassa capacità rappresentativa (in particolare per EMLPs con $d = 4$) sia alla maggiore difficoltà di ottimizzazione introdotta da uno spazio intermedio non necessario.

Questi risultati confermano che, per l’approccio a istante temporale, un’architettura semplice ma ben calibrata è spesso più efficace di modelli più complessi, specialmente quando il contesto temporale viene ignorato. In scenari del genere, è preferibile massimizzare l’efficienza e la capacità discriminativa diretta, piuttosto che cercare astrazioni latenti.

La Figura 2.3 mostra le matrici di confusione relative al modello **MLP** nell’approccio a istante temporale, sia a livello di sample (sinistra) che di esperimento (destra).

A livello **sample-level**, si osservano buone prestazioni nella classificazione di classi ben distinte come *WATER VAPOR* e *ACETIC ACID*, che mostrano una forte diagonale. Tuttavia, molte altre classi presentano una significativa dispersione, indicando incertezza o confusione tra sostanze chimicamente simili. Ad esempio, le predizioni per *AMMONIA*, *ETHANOL* e *ACETONE* sono distribuite su più etichette, riflettendo la loro vicinanza nei pattern sensoriali.

Nella matrice a **experiment-level**, ottenuta tramite voto di maggioranza sui singoli istanti, si evidenzia un miglioramento nella coerenza delle predizioni, ma rimangono alcuni errori sistematici. È interessante notare che il modello riesce comunque a mantenere una certa accuratezza anche su esperimenti con dati particolarmente variabili, confermando la robustezza della strategia di voto rispetto al rumore locale.

Capitolo 3

Classificazione con Finestre Temporali

In questo capitolo viene approfondito un approccio alternativo alla classificazione delle misurazioni sensoristiche: l'**approccio a finestre temporali**. A differenza di quanto visto nel Capitolo 2, dove ogni istante temporale veniva trattato come un campione indipendente, qui si considera un'intera sequenza temporale come unità di input per il modello.

3.1 Struttura del dato: finestre temporali

In questo approccio, ogni esperimento viene segmentato in finestre temporali di dimensione fissa $N \times 16$, dove $N = 100$. Ogni finestra rappresenta un campione di addestramento composto da 100 osservazioni consecutive sulle 16 feature disponibili.

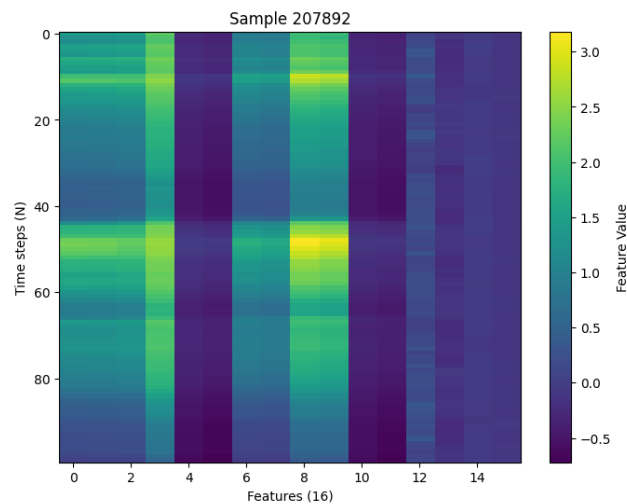


Figura 3.1 Esempio di rappresentazione di un campione da finestra temporale della dimensione 100×16 (time steps $\times$ feature). Si notano chiaramente differenze strutturali tra le colonne, indicative della risposta eterogenea delle diverse feature al segnale chimico.

Per massimizzare la quantità di dati disponibili, è stato adottato un meccanismo di **overlapping**, in cui ciascuna finestra successiva scorre di un solo passo temporale ($N - 1$ di overlap). Questo garantisce un numero elevato di campioni per ciascun esperimento, migliorando la qualità dell'apprendimento.

Ogni finestra è etichettata con la classe associata all'intero esperimento da cui è stata estratta, mantenendo così la coerenza semantica del dato.

Modelli utilizzati

Tutti i modelli descritti in questa sezione condividono le stesse impostazioni iperparametriche dettagliate nella Sezione 2.3, e impiegano nel blocco finale il *Common MLP Block* descritto nella Sezione 2.1.2.

MLP Il primo modello testato è un semplice MLP, già introdotto nella Sezione 2.1, applicato alla versione flattenata della finestra temporale 100×16 . La sua architettura è la seguente:

$$\mathbb{R}^{100 \times 16} \xrightarrow{\text{Flatten}} \mathbb{R}^{1600} \xrightarrow{\text{Linear}} \mathbb{R}^{64} \xrightarrow{\text{Common MLP Block}} \mathbb{R}^{17}$$

EMLP È stato inoltre utilizzato l'EMLP, anch'esso già descritto nella Sezione 2.2, che prevede un layer di compressione in uno spazio latente di dimensione d , prima della proiezione verso i 64 neuroni intermedi. Le due varianti testate impiegano $d \in \{20, 80\}$:

$$\mathbb{R}^{100 \times 16} \xrightarrow{\text{Flatten}} \mathbb{R}^{1600} \xrightarrow{\text{Linear}} \mathbb{R}^d \xrightarrow{\text{Linear}} \mathbb{R}^{64} \xrightarrow{\text{Common MLP Block}} \mathbb{R}^{17}$$

3.2 Column Selective Encoder MLP

Il modello **CSEMLP** introduce una variante dell'EMLP in cui le connessioni tra input e layer latente sono **selettive per colonna**. Invece di connettere completamente il vettore di input ai neuroni latenti, si definiscono:

- M neuroni connessi solo ai dati di ciascuna colonna (feature) del segnale;
- K neuroni completamente connessi a tutto il vettore di input.

L'architettura diventa quindi:

$$\mathbb{R}^{100 \times 16} \xrightarrow{\text{Flatten}} \mathbb{R}^{1600} \xrightarrow{\text{Selective Linear}} \mathbb{R}^{16 \cdot M + K} \xrightarrow{\text{Linear}} \mathbb{R}^{64} \xrightarrow{\text{Common MLP Block}} \mathbb{R}^{17}$$

Sono stati testati due setting:

- **CSEMLPs**: $M = 1$, $K = 3$ (totale 19 dimensioni latenti)
- **CSEMLPb**: $M = 5$, $K = 3$ (totale 83 dimensioni latenti)

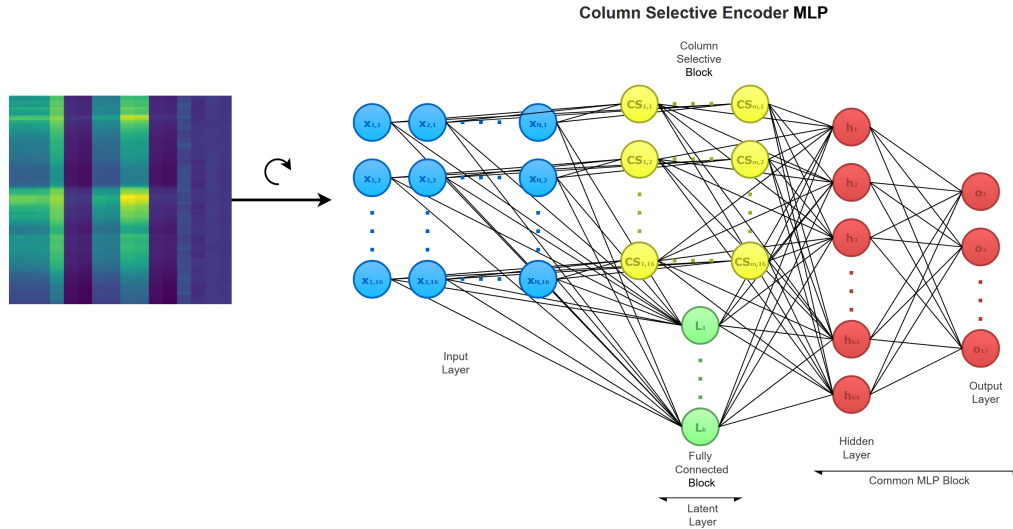


Figura 3.2 Architettura del modello **CSEMLP**. Il segnale è elaborato da un blocco selettivo per colonna (CS) e un blocco fully connected (L), prima di essere processato dal *Common MLP Block*.

3.3 Reti Neurali Convoluzionali

Le **Convolutional Neural Networks** (CNN) sono una classe di reti neurali particolarmente efficaci per l'elaborazione di dati con struttura spaziale o temporale. L'idea di base è quella di applicare filtri convoluzionali che apprendano rappresentazioni locali e traslazionalmente invarianti del dato. Le CNN sono ampiamente utilizzate in vari campi dell'intelligenza artificiale, grazie alla loro capacità di apprendere rappresentazioni gerarchiche dai dati. Recenti studi hanno evidenziato le loro prestazioni superiori in compiti di classificazione, rilevamento di oggetti e segmentazione semantica, rendendole una scelta preferenziale in molte applicazioni moderne⁽⁵⁾.

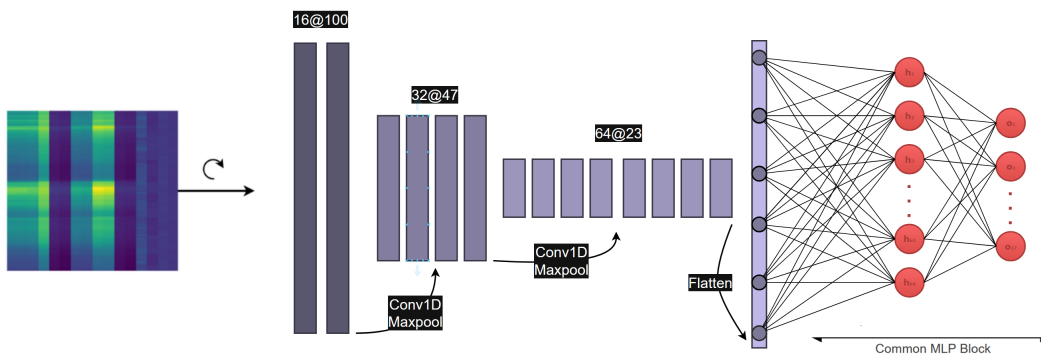


Figura 3.3 Architettura del modello **CNN1D**. La sequenza temporale 100×16 viene trasposta in un tensore 16×100 e processata con convoluzioni 1D lungo l'asse temporale. L'output viene poi appiattito e passato al *Common MLP Block*.

Blocco convoluzionale:

$$\begin{aligned}
&\mathbb{R}^{100 \times 16} \xrightarrow{\text{Transpose}} \mathbb{R}^{16 \times 100} \xrightarrow{\text{Conv1d}(16 \rightarrow 32, \text{kernel}=3)} \mathbb{R}^{32 \times 98} \xrightarrow{\text{MaxPool1d}(2) + \text{Dropout}} \mathbb{R}^{32 \times 49} \\
&\xrightarrow{\text{Conv1d}(32 \rightarrow 64, \text{kernel}=3)} \mathbb{R}^{64 \times 47} \xrightarrow{\text{MaxPool1d}(2) + \text{Dropout}} \mathbb{R}^{64 \times 23} \xrightarrow{\text{Flatten}} \mathbb{R}^{1472}
\end{aligned}$$

Blocco fully connected:

$$\mathbb{R}^{1472} \xrightarrow{\text{Linear}(1472 \rightarrow 64)} \mathbb{R}^{64} \xrightarrow{\text{ReLU} + \text{Dropout}} \mathbb{R}^{64} \xrightarrow{\text{Linear}(64 \rightarrow 17)} \mathbb{R}^{17}$$

Questa architettura sfrutta la natura sequenziale del dato e applica convoluzioni 1D lungo l'asse temporale, modellando pattern locali nelle serie temporali multivariate.

Nella parte iniziale del modello, due layer convoluzionali con kernel di dimensione 3 agiscono lungo l'asse temporale per catturare strutture locali e dipendenze a breve termine tra i segnali. Ogni convoluzione è seguita da un'operazione di *MaxPooling1D* con **kernel size pari a 2**, che ha il compito di ridurre progressivamente la risoluzione temporale e, di conseguenza, la dimensionalità computazionale, mantenendo al contempo le informazioni più salienti. Questa combinazione di convoluzione e pooling permette di estrarre rappresentazioni gerarchiche sempre più astratte, che vengono poi appiattite e passate al *Common MLP Block* per la classificazione finale.

3.4 Risultati e osservazioni

Risultati La tabella 3.1 riporta i risultati ottenuti dai modelli MLP, EMLP, CSEMLP e CNN in termini di accuratezza sui set di training, validazione e test, sia a livello di sample che a livello di esperimento. Le figure 3.4 e 3.5 ci mostrano le curve di apprendimento e le matrici di confusione di quello che è risultato il miglior modello in questo approccio: l'**EMLPs**.

Model	Size (MB)	Epochs	Train S _{acc}	Val. S _{acc}	Test S _{acc}	Train E _{acc}	Val. E _{acc}	Test E _{acc}
MLP	0.414	10	0.719	0.628	0.481	0.903	0.683	0.536
EMLPs	0.138	30	0.812	0.680	0.621	0.885	0.707	0.683
EMLPb	0.537	12	0.750	0.696	0.575	0.956	0.780	0.634
CSEMLPs	0.035	9	0.750	0.606	0.501	0.840	0.731	0.585
CSEMLPb	0.077	9	0.688	0.636	0.516	0.882	0.707	0.585
CNN	0.413	9	0.844	0.607	0.563	0.985	0.658	0.682

Tabella 3.1 Prestazioni dei modelli nell'approccio a finestre temporali, riportate in termini di accuratezza sui singoli campioni (S_{acc}) e a livello di esperimento (E_{acc}), come definito nella Sezione 2.4.1.

Osservazioni Il modello **EMLPs** (Encoder MLP con spazio latente di dimensione 20) si distingue come il più performante in questo approccio, ottenendo la miglior accuratezza sia a livello di sample (**62.1%**) che a livello di esperimento sul test set (**68.3%**), con un peso contenuto in memoria (**0.138 MB**). Questa configurazione sembra offrire un ottimo compromesso tra capacità rappresentativa e generalizzazione, risultando particolarmente adatta alla struttura temporale aggregata dei dati. Inoltre, si osserva una notevole capacità del modello nel non incorrere in overfitting: l'accuratezza sul training (81.2%) rimane contenuta rispetto a modelli più complessi, suggerendo una buona regolarizzazione e un'efficace capacità di astrazione.

Il modello **CNN** si colloca subito dietro, con un'**experiment-level accuracy** di **68.2%**, praticamente identica a quella di EMLPs, ma con un footprint in memoria decisamente maggiore (**0.413 MB**). Tuttavia, a differenza di EMLPs, la CNN raggiunge un'accuratezza sul training molto elevata (**98.5%**) in sole 9 epoche, evidenziando una tendenza marcata all'**overfitting**. Questo comportamento suggerisce che la CNN ha un'alta capacità di apprendere le specificità del training set, ma fatica maggiormente a generalizzare sui dati non visti.

Tale osservazione non va letta in modo esclusivamente negativo: il fatto che la CNN riesca a raggiungere risultati così competitivi nonostante il sovradattamento indica un potenziale non ancora pienamente sfruttato. Con opportuni accorgimenti, come l'adozione di tecniche di regolarizzazione più efficaci (es. aumento del dropout, early stopping più aggressivo, o data augmentation) e una ricerca più mirata degli iperparametri (numero di canali, kernel size, stride, padding), è plausibile ipotizzare un ulteriore miglioramento delle prestazioni, anche se al costo di una maggiore complessità computazionale rispetto a EMLPs.

Il modello **MLP**, pur mantenendo una buona solidità (53.6% a livello di esperimento), è superato dai modelli encoder-based, confermando l'utilità della compressione in spazi latenti quando si opera su finestre temporali più lunghe.

Le performance dei modelli **CSEMLP**, seppur interessanti, risultano inferiori rispetto agli EMLP tradizionali.

Infine, se confrontiamo questi risultati con quelli ottenuti nell'approccio a istante temporale (Sezione 2.4), emerge un dato molto interessante: le *sample-level accuracy* sono comparabili tra i due approcci, ma la **experiment-level accuracy** è decisamente superiore nell'approccio a finestre temporali. Questo suggerisce che l'uso di finestre consente di catturare una maggiore coerenza temporale e fornisce una rappresentazione più robusta dell'esperimento nel suo complesso.

La Figura 3.5 mostra le confusion matrix per il modello **EMLPs** nell'approccio a finestre temporali, che si è rivelato il più efficace nel task di classificazione multiclasse.

A livello di **sample**, si osserva una netta diagonalizzazione della matrice, con ottime prestazioni su molte classi, in particolare per sostanze come *WATER VAPOR*, *ACETIC ACID* e *UREA*, che risultano essere classificate correttamente nella quasi totalità dei casi. Alcune confusioni rimangono su classi chimicamente affini come *ACETONE* e *ETHANOL*, ma in misura decisamente inferiore rispetto all'approccio su singoli istanti.

La matrice a **livello di esperimento** conferma ulteriormente la solidità del modello: in questo caso, la maggioranza delle etichette viene predetta correttamente, con un'elevata coerenza tra gli istanti all'interno dello stesso esperimento. Questo

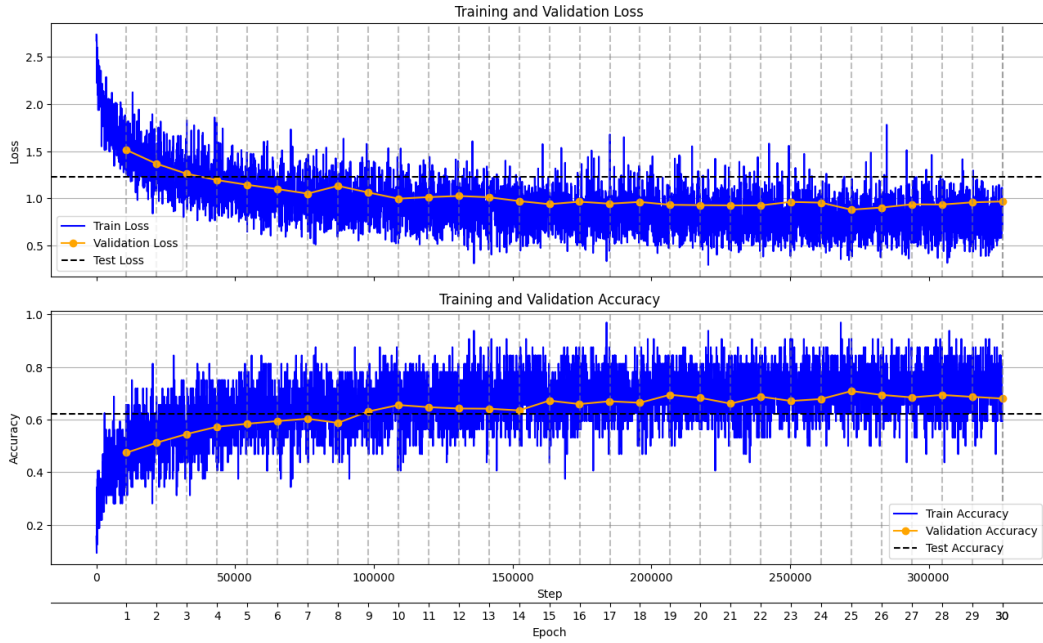


Figura 3.4 Learning curve del modello EMLPs sulle finestre temporali durante l'addestramento: andamento della loss e dell'accuracy su training e validation.

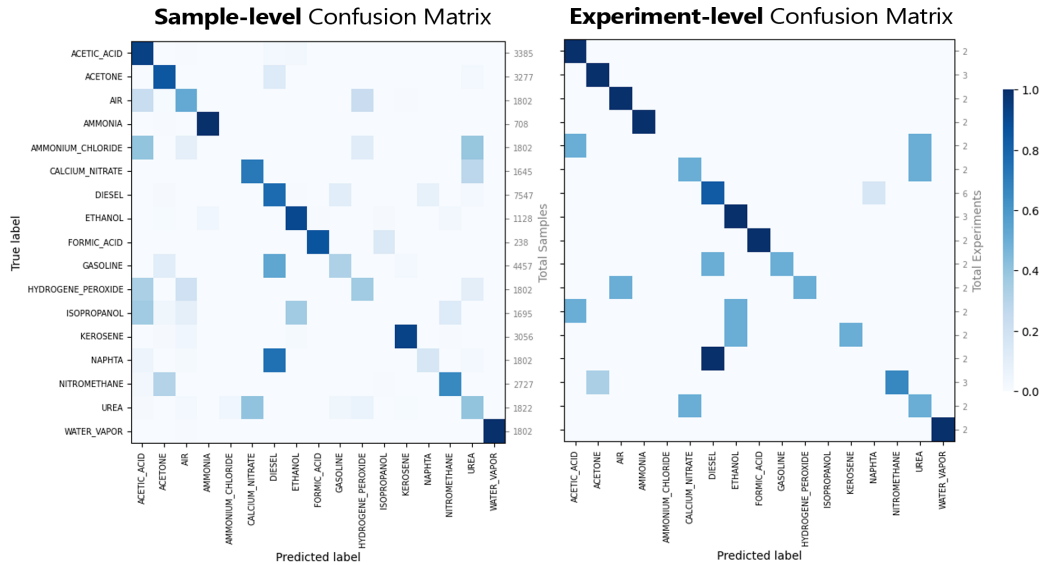


Figura 3.5 Matrici di confusione sul test set per il modello EMLPs sulle finestre temporali. A sinistra: classificazione *sample-level*, a destra: classificazione *experiment-level*.

indica che il modello ha appreso rappresentazioni che non solo funzionano su base locale, ma si mantengono robuste lungo la sequenza temporale.

A confronto con la confusion matrix del modello MLP nel caso di singoli istanti temporali (Figura 2.3), si nota una sostanziale riduzione dell'ambiguità tra le classi e una migliore distribuzione delle predizioni corrette. Questo rafforza l'idea che l'informazione temporale aggregata offerta dalle finestre sia cruciale per ottenere

prestazioni affidabili in scenari sensoristici reali.

Un aspetto ricorrente che emerge da entrambe le confusion matrix, sia nell'approccio a istanti temporali che in quello a finestre, è la tendenza di molte classi a essere erroneamente predette come **DIESEL**. Questo fenomeno può essere attribuito allo **squilibrio intrinseco del dataset**, in cui la classe *DIESEL* è rappresentata da un numero di esperimenti e campioni sensibilmente superiore rispetto alle altre (come si può notare dalla figura 1.1). La predominanza statistica di tale classe può portare i modelli a preferirla come predizione, specialmente nei casi ambigui o borderline, evidenziando così l'importanza di strategie di bilanciamento nei futuri sviluppi.

Capitolo 4

Analisi Temporale delle Predizioni

Questo capitolo è dedicato all'analisi della **distribuzione temporale delle predizioni** generate dai modelli sviluppati nei capitoli precedenti. L'obiettivo è comprendere come la capacità del modello di individuare correttamente la classe di appartenenza di un esperimento si evolva nel tempo, fornendo così informazioni cruciali sulla stabilità, reattività e coerenza delle predizioni.

4.1 Motivazione e utilità dell'esperimento

Quando si ha a che fare con serie temporali, è importante non limitarsi a valutare l'accuratezza complessiva su un insieme di campioni indipendenti, ma osservare *quando e con quale confidenza* il modello riesce a riconoscere correttamente la sostanza analizzata. Questo tipo di analisi può mettere in luce:

- se il modello tende ad attivarsi correttamente solo dopo un certo tempo di esposizione,
- se vi sono fluttuazioni anomale nelle predizioni durante l'acquisizione,
- quanto è stabile la predizione lungo l'esperimento (continuità decisionale),
- se le finestre o gli istanti classificati correttamente si concentrano in determinate zone temporali.

Queste informazioni sono particolarmente utili in contesti applicativi reali, dove la tempestività e la stabilità della risposta sono elementi fondamentali.

4.2 Risultati e osservazioni

Risultati La Figura 4.1 riporta un confronto visivo tra due modelli: il **MLP** sull'approccio a *istante temporale* (a sinistra) e l'**EMLPs** sull'approccio a *finestre temporali* (a destra). Per ciascun campione all'interno degli esperimenti di test sono riportate due rappresentazioni:

- **Valore di softmax della classe corretta:** mostrato come mappa di colore (da bianco a blu), indica la probabilità assegnata dal modello alla classe corretta. Un blu più scuro rappresenta una maggiore sicurezza.
- **Ranking della classe corretta:** rappresentato tramite codifica a colori:
 - Verde: la classe corretta è classificata come **Top-1**.
 - Giallo: la classe corretta è tra le **Top-5**.
 - Rosso: la classe corretta è al di fuori delle prime 5 (predizione errata).

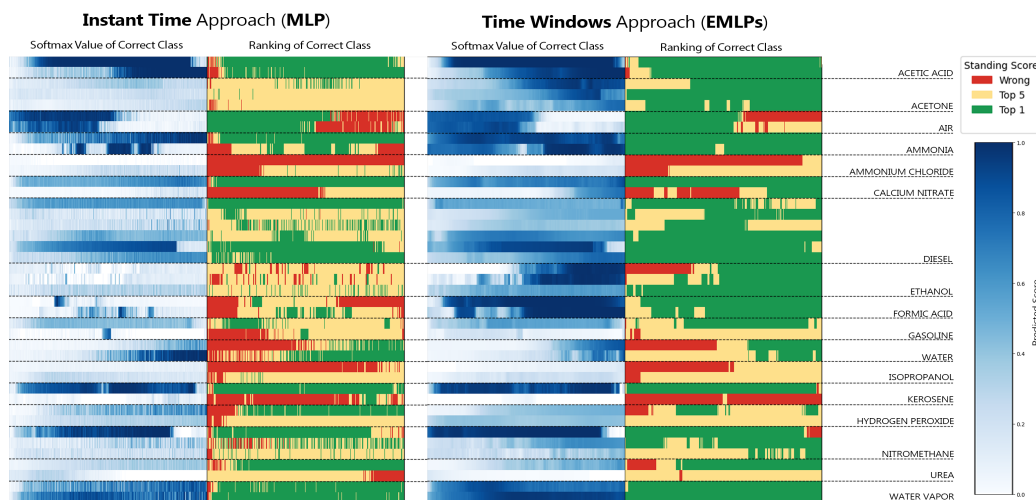


Figura 4.1 Distribuzione temporale delle predizioni corrette nei due approcci. A sinistra: MLP su singoli istanti temporali; a destra: EMLP su finestre temporali. Ogni riga rappresenta un esperimento, ordinato per classe chimica.

Osservazioni La visualizzazione della Figura 4.1 mostra in maniera chiara la maggiore robustezza del modello basato su finestre temporali (EMLPs) rispetto a quello basato su singoli istanti (MLP). A livello qualitativo:

- Nel modello **MLP**, i valori di softmax appaiono spesso bassi e discontinui lungo la sequenza temporale (colonne di intensità molto variabile), suggerendo che le predizioni siano altamente sensibili al singolo istante osservato. Le bande rosse nella seconda colonna (ranking) evidenziano una frequente incertezza o errore nel riconoscimento della classe corretta.
- Nel caso dell'**EMLPs**, al contrario, si osserva una maggiore coerenza temporale: le predizioni sono più stabili (colorazione blu uniforme) e il ranking corretto tende a rimanere nella Top-1 o Top-5 per l'intera durata dell'esperimento. Questo conferma l'efficacia dell'integrazione temporale nel migliorare la comprensione globale del segnale.

Capitolo 5

Esperimenti di Classificazione Binaria

Per approfondire la capacità discriminativa dei modelli proposti nei capitoli precedenti, è stato condotto un ulteriore esperimento di classificazione binaria. In particolare, si è scelto di confrontare le due classi **ETHANOL** e **ACETONE**, selezionate per la loro similarità chimica e per un numero di esperimenti comparabile, rendendo il compito sfidante ma significativo.

A differenza degli esperimenti principali di classificazione multiclasse, l'obiettivo qui è verificare se i modelli siano in grado di cogliere differenze anche sottili tra due sole classi in uno scenario più focalizzato. Questo approccio permette inoltre di valutare più finemente la generalizzazione su un task meno complesso, ma comunque rilevante dal punto di vista applicativo.

Per ciascuno dei due approcci principali (istante temporale e finestra temporale), è stato utilizzato il **modello migliore emerso nei capitoli precedenti**:

- Per l'approccio a istante temporale, è stato utilizzato l'**MLP semplice** descritto nella Sezione 2.1, che aveva mostrato le performance migliori in termini di accuratezza e robustezza.
- Per l'approccio a finestra temporale, si è fatto ricorso al **modello EMLPs** (Encoder + MLP con spazio latente di dimensione 20), individuato come il più performante nel Capitolo 3.

L'esperimento è stato eseguito utilizzando una **validazione incrociata a 10 fold**, selezionando in ciascun fold 3 esperimenti per classe come validation set, 3 come test set e il resto come training. Questa strategia assicura una valutazione robusta e riduce la varianza dovuta alla specifica scelta di partizionamento.

5.1 Classificazione binaria su singoli istanti temporali

Fold	Epochs	Train S _{acc}	Val. S _{acc}	Test S _{acc}	Train E _{acc}	Val. E _{acc}	Test E _{acc}
Fold 1	25	1.000	0.865	0.750	0.984	0.833	0.833
Fold 2	20	1.000	0.784	0.816	1.000	0.833	0.833
Fold 3	18	0.812	0.753	0.843	0.967	0.833	1.000
Fold 4	23	0.935	0.779	0.562	0.951	0.833	0.500
Fold 5	25	0.917	0.589	0.813	0.951	0.500	0.833
Fold 6	18	0.933	0.771	0.751	0.967	0.833	0.667
Fold 7	25	0.846	0.766	0.885	0.967	0.667	1.000
Fold 8	25	0.778	0.886	0.805	0.951	1.000	0.833
Fold 9	25	0.750	0.873	0.771	0.967	1.000	0.833
Fold 10	6	0.857	0.675	0.840	0.852	0.833	1.000
Mean	20.5	0.883	0.775	0.764	0.956	0.820	0.837

Tabella 5.1 Prestazioni medie e per fold del modello MLP su task binario tra ETHANOL e ACETONE nell’approccio a istanti temporali.

5.2 Classificazione binaria su finestre temporali

Fold	Epochs	Train S _{acc}	Val. S _{acc}	Test S _{acc}	Train E _{acc}	Val. E _{acc}	Test E _{acc}
Fold 1	20	0.935	0.968	0.711	1.000	1.000	0.667
Fold 2	6	1.000	0.767	0.843	1.000	0.667	0.833
Fold 3	6	0.889	0.794	0.874	1.000	0.833	0.833
Fold 4	10	1.000	0.898	0.640	1.000	1.000	0.667
Fold 5	10	1.000	0.830	0.812	0.984	0.833	0.833
Fold 6	21	1.000	0.938	0.827	1.000	1.000	0.833
Fold 7	6	1.000	0.781	0.903	1.000	0.833	0.833
Fold 8	7	1.000	0.878	0.956	1.000	0.833	1.000
Fold 9	18	0.923	0.945	0.739	1.000	1.000	0.833
Fold 10	6	1.000	0.673	0.654	1.000	0.667	0.667
Mean	11.0	0.975	0.847	0.788	0.998	0.867	0.790

Tabella 5.2 Prestazioni medie e per fold del modello EMLPs su task binario tra ETHANOL e ACETONE nell’approccio a finestre temporali.

5.3 Osservazioni

I risultati della classificazione binaria evidenziano un miglioramento significativo rispetto al task multiclasse, come atteso, con performance elevate su entrambi gli approcci.

L'approccio *a finestre temporali*, pur richiedendo modelli leggermente più complessi, si dimostra vantaggioso nella classificazione a livello di singolo campione (**78.8%** vs **76.4%**), grazie alla sua capacità di catturare dinamiche temporali locali. Tuttavia, sorprendentemente, l'approccio *a istante temporale* raggiunge una performance media per esperimento leggermente superiore (**83.7%** vs **79.0%**), suggerendo che la maggiore granularità del dato, unita a un maggior numero di predizioni da aggregare tramite majority voting, può talvolta compensare la mancanza di contesto temporale.

Questa inversione di tendenza rispetto al caso multiclasse è particolarmente interessante: mentre in precedenza le finestre temporali risultavano nettamente superiori, nel setting binario entrambe le strategie mostrano prestazioni comparabili e robuste.

Nel complesso, i risultati dimostrano che entrambi gli approcci sono capaci di generalizzare efficacemente, con performance stabili nei diversi fold della cross-validazione. Ciò conferma la solidità del preprocessing adottato e l'efficacia delle architetture testate in uno scenario meno affollato e più bilanciato.

Capitolo 6

Classificazione con Interpolazione RBF

6.1 Descrizione dell'approccio

In questo capitolo analizziamo un approccio alternativo alla classificazione delle serie temporali basato sull'uso delle **Radial Basis Functions (RBF)**. L'idea alla base di questa metodologia è quella di approssimare ogni singola serie temporale (una colonna di una tabella, quindi una feature del sensore) con una combinazione lineare di m funzioni radiali centrate in punti fissi lungo l'asse temporale.

Formalmente, una serie temporale $\{y(t_i)\}_{i=1}^M$ viene trattata come un campione rumoroso di una funzione continua sottostante $y : \mathbb{R} \rightarrow \mathbb{R}$, e approssimata nella forma:

$$\tilde{y}(t) = \sum_{j=1}^m a_j \phi(|t - c_j|)$$

dove:

- ϕ è una funzione radiale, tipicamente gaussiana: $\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$,
- c_j sono i centri delle basi, equidistanti nel dominio della serie temporale,
- a_j sono i coefficienti da determinare tramite regressione lineare.

Questo approccio, già utilizzato con successo per compiti di riconoscimento in segnali ECG⁽¹⁰⁾, consente di ridurre drasticamente la dimensionalità del problema e migliora la robustezza del modello rispetto al rumore e alle variazioni locali.

6.2 Struttura del dato in ingresso

Applicando questo metodo a ciascuna delle 16 serie temporali associate a un esperimento, otteniamo una rappresentazione compressa costituita da m coefficienti per serie, per un totale di $m \times 16$ feature per esperimento. Con $m = 5$, come nel nostro caso, il vettore risultante ha dimensione:

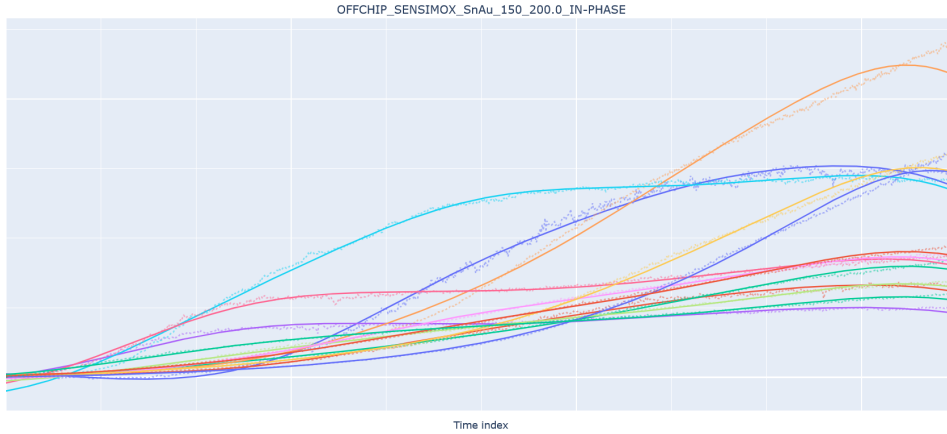


Figura 6.1 Esempio di approssimazione tramite RBF ($m = 5$) sulla feature `SnAu_150_200.0_IN-PHASE`. Le linee tratteggiate rappresentano le curve originali di alcuni esperimenti con `AMMONIUM_CHLORIDE`, mentre le linee continue mostrano il fit ottenuto con le basi radiali.

$$\mathbb{R}^{5 \times 16} = \mathbb{R}^{80}$$

Questo vettore rappresenta quindi l'intero esperimento e costituisce l'input per i modelli di classificazione.

Modelli utilizzati

MLP Sono state testate le seguenti architetture MLP:

$$\mathbb{R}^{80} \rightarrow \mathbb{R}^{64} \rightarrow \mathbb{R}^{17}, \quad \mathbb{R}^{80} \rightarrow \mathbb{R}^{256} \rightarrow \mathbb{R}^{17}, \quad \mathbb{R}^{80} \rightarrow \mathbb{R}^{64} \rightarrow \mathbb{R}^{64} \rightarrow \mathbb{R}^{17}$$

Tutte le reti sono allenate con i medesimi iperparametri di base usati in precedenza nella Sezione 2.3, tuttavia, data la natura statica dei dati, le epoche massime per l'allenamento sono state fissate a **1000**, con **early stopping** attivo e pazienza pari a **100** epoche.

6.3 Random Forest

Oltre ai modelli neurali, è stato testato un classificatore Random Forest, particolarmente adatto a contesti con dati *tabulari* come quelli prodotti dall'approccio RBF. Le Random Forest sono insiemi di alberi decisionali addestrati su sottoinsiemi casuali dei dati, con predizione finale ottenuta tramite voto di maggioranza.

Ogni albero viene addestrato su una porzione casuale del dataset (bootstrap), mentre in ogni nodo è selezionato casualmente un sottoinsieme delle feature per effettuare la divisione. Questo meccanismo introduce diversità e riduce il rischio di overfitting.

Nel nostro esperimento, è stato utilizzato un modello con numero di stimatori (alberi) = 100

Le Random Forest si sono dimostrate particolarmente efficaci in vari contesti di classificazione biologica e ambientale, offrendo buone performance anche in assenza di pre-processing complessi o ottimizzazione profonda dei parametri. In ambito ecologico, ad esempio, sono state utilizzate con successo per modellare la distribuzione delle specie, grazie alla loro capacità di gestire dati complessi e non lineari⁽¹¹⁾. Inoltre, nelle competizioni di machine learning, le Random Forest hanno spesso ottenuto risultati di rilievo, come nel caso della competizione "Leaf Classification" su Kaggle, dove sono state impiegate per classificare efficacemente le specie di foglie⁽⁸⁾.

6.4 Risultati e osservazioni

Model	Size (MB)	Epochs	Train Acc.	Val. Acc.	Test Acc.
MLP_64	0.025	598	0.312	0.341	0.268
MLP_256	0.100	716	0.688	0.463	0.390
MLP_64_64	0.042	1000	0.562	0.439	0.414
RandomForest	3.540	–	1.000	0.658	0.439

Tabella 6.1 Performance dei diversi modelli nell'approccio RBF.

Risultati

Osservazioni Dalla Tabella 6.1 emerge chiaramente che il modello **Random Forest** ottiene le migliori prestazioni su validation (**65.8%**) e test (**43.9%**). L'accuratezza perfetta sul training (**100%**) è attesa, poiché la profondità massima degli alberi è impostata su **None**, consentendo una suddivisione completa dei dati fino alla purezza. Questo non implica necessariamente overfitting, ma riflette l'elevata capacità del modello di adattarsi ai dati di partenza.

Tra le reti neurali, **MLP_64_64** risulta il più efficace, con una test accuracy di **41.4%**. Il raggiungimento del limite massimo di epoche (**1000**) senza attivare l'*early stopping* suggerisce un apprendimento ancora in corso, con possibilità di miglioramento se il training fosse stato esteso.

Il modello **MLP_256**, con maggiore capacità, mostra buone prestazioni (**39.0%** su test), ma tende a sovra-adattarsi leggermente. Al contrario, **MLP_64** evidenzia una capacità troppo limitata per il compito, con performance deboli su tutti gli split.

È importante notare che, sebbene il modello Random Forest offra le migliori prestazioni tra quelli testati, il suo **peso in memoria (3.54 MB)** è significativamente superiore a quello delle reti neurali, il che potrebbe rappresentare un vincolo in scenari di deploy su dispositivi embedded o edge computing.

Confrontando questi risultati con gli approcci precedenti, notiamo che le performance migliori dell'RBF (**RF: 43.9%**) e dell'MLP a doppio strato sono in linea con quelle dell'MLP su istanti temporali (anch'esso **43.9%**), ma nettamente inferiori rispetto all'approccio a finestre temporali, dove l'EMLPs ha raggiunto una test accuracy di **68.3%** a livello di esperimento. Questo conferma il valore dell'informazione temporale nella classificazione di segnali sensoristici.

Capitolo 7

Classificazione con Elaborazione Temporale Continua

7.1 Recurrent Neural Networks

In questo capitolo si introduce il concetto generale di elaborazione temporale continua, un paradigma particolarmente adatto per la modellazione di dati sequenziali come le serie temporali. Questo approccio si basa sull'idea che l'informazione acquisita a ogni istante temporale sia correlata a quella degli istanti precedenti, e che la sequenza stessa sia l'entità da modellare. A differenza dei modelli tradizionali che trattano ogni osservazione come indipendente, i modelli con memoria ricorrente utilizzano meccanismi interni che permettono di “ricordare” informazioni passate.

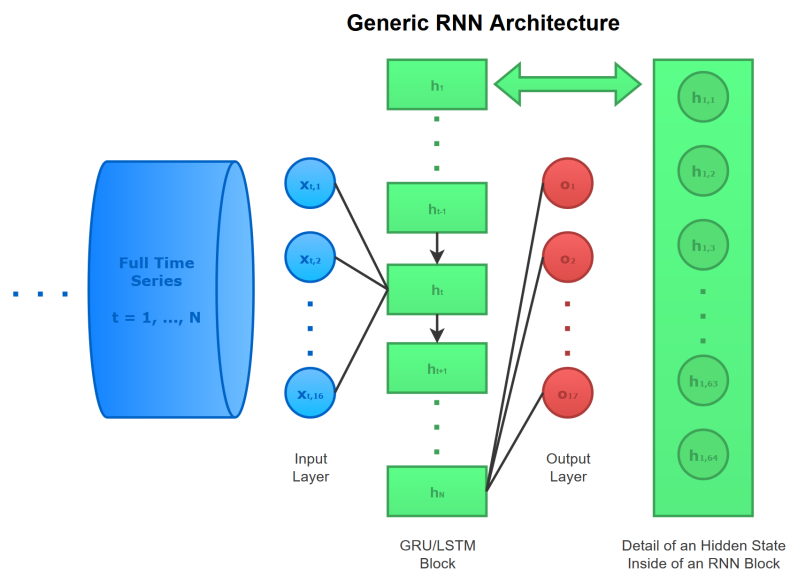


Figura 7.1 Architettura di una generica RNN. Il vettore x_t si collega all'hidden state h_t (analogo all'hidden layer che si trovava nell'MLP Block nei capitoli precedenti) che a sua volta è collegato a h_{t-1} e h_{t+1} . Infine l'output layer comunica solo con l'ultimo hidden state h_N

Le **Reti Neurali Ricorrenti** (Recurrent Neural Networks, RNN, Fig. 7.1) sono progettate proprio con questo obiettivo: consentire un'elaborazione sequenziale dei dati mantenendo uno stato interno aggiornato a ogni step temporale. Questo le rende particolarmente adatte alla classificazione di serie temporali, dove la dinamica del segnale contiene informazioni cruciali per la predizione.

7.2 Reti Gated Recurrent Unit

Il modello Gated Recurrent Unit (GRU) rappresenta un'estensione delle RNN classiche, progettata per alleviare il problema della attenuazione del gradiente nei processi di apprendimento su lunghe sequenze. Ogni unità GRU gestisce due porte principali:

- **Reset gate** r_t : decide quanto del precedente stato nascosto h_{t-1} dimenticare.
- **Update gate** z_t : decide quanto del nuovo stato candidato $\tilde{h}_t$ verrà usato per aggiornare lo stato corrente.

Le equazioni che regolano il comportamento della GRU sono le seguenti:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}), \quad r_t = \sigma(W_r x_t + U_r h_{t-1})$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t \odot h_{t-1})), \quad h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

dove x_t rappresenta l'input al tempo t , h_t lo stato nascosto, e σ la funzione sigmoide. Le porte z_t e r_t controllano il flusso dell'informazione, permettendo un apprendimento più stabile su sequenze anche lunghe.

7.3 Reti Long Short-Term Memory

Le Long Short-Term Memory (LSTM) sono una generalizzazione ancora più potente delle RNN, che gestiscono una **memoria esplicita a lungo termine**, rappresentata dallo stato della cella c_t . Ogni unità LSTM è composta da tre porte principali:

- **Input gate** i_t : controlla quanto dell'input corrente va a influenzare lo stato della cella.
- **Forget gate** f_t : determina quanto dello stato precedente c_{t-1} deve essere mantenuto.
- **Output gate** o_t : regola quanto dello stato della cella c_t verrà emesso come output h_t .

Il funzionamento completo è dato dalle equazioni:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \quad f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \quad o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t)$$

A differenza delle GRU, le LSTM separano chiaramente lo *stato nascosto* h_t dalla *memoria della cella* c_t , il che conferisce maggiore flessibilità nella gestione della memoria su tempi lunghi. Tuttavia, ciò comporta anche un maggior numero di parametri da apprendere e, di conseguenza, una maggiore complessità computazionale.

Differenze tra GRU e LSTM. Sebbene le LSTM siano più espressive, le GRU si distinguono per la loro efficienza e semplicità architetturale. In molti contesti pratici, le GRU riescono a ottenere prestazioni paragonabili alle LSTM con un numero inferiore di parametri e un addestramento più stabile, specialmente su dataset di dimensioni moderate come quello analizzato in questa tesi.

7.4 Classificazione di Serie Temporali Complete e di Lunghezza Variabile

Nel nostro contesto, ogni esperimento viene trattato come una sequenza temporale completa, mantenendo la variabilità nella durata tra esperimenti diversi. Il modello GRU/LSTM riceve in input l'intera sequenza $(x_1, x_2, \dots, x_N)$, dove N è la lunghezza variabile della serie. Ogni step x_t viene processato sequenzialmente, aggiornando uno stato nascosto che riassume l'informazione fino a quel punto:

$$h_t = \text{RNN}(x_t, h_{t-1})$$

Alla fine della sequenza, l'ultimo stato h_N viene utilizzato per la classificazione dell'intero esperimento. Questo approccio permette al modello di apprendere una rappresentazione dinamica del segnale nel suo insieme, catturando dipendenze temporali anche lontane.

Tutti i setting iperparametrici sono rimasti invariati rispetto a quelli descritti nella Sezione 2.3, ad eccezione del numero massimo di epoche, fissato a 1000, e della pazienza dell'early stopping, aumentata a 100. Tale scelta è coerente con quanto già adottato nell'approccio RBF (Capitolo 6), data la natura di input a livello di esperimento.

7.4.1 Risultati e Osservazioni

Modello	Size (MB)	Epoche	Train E_{acc}	Val E_{acc}	Test E_{acc}
GRU	0.067	1000	0.667	0.530	0.390
LSTM	0.088	287	0.430	0.368	0.268

Tabella 7.1 Prestazioni dei modelli GRU e LSTM sull'intera sequenza.

Dal confronto tra GRU e LSTM emerge chiaramente la superiorità del primo. Il modello GRU, con una test accuracy di **0.390**, si avvicina sensibilmente alle prestazioni del miglior MLP nell'approccio RBF (0.414).

Tuttavia, la GRU rimane ben al di sotto delle performance ottenute dal modello **EMLPs** con finestre temporali, che ha raggiunto una test accuracy a livello di esperimento pari a **0.683**. Questo risultato evidenzia come, sebbene la GRU offra una buona capacità di modellare sequenze intere e di lunghezza variabile, l'approccio

di compressione spaziale-temporale tramite finestre latenti si dimostri ancora il più efficace.

Va infine sottolineato che, rispetto alla LSTM, la GRU si comporta meglio anche in termini di stabilità durante il training: il modello LSTM ha interrotto precocemente l'addestramento (287 epoche), segno di un'ottimizzazione più problematica e un rischio maggiore di overfitting. Questo conferma l'efficacia strutturale delle GRU per dataset di dimensioni moderate e segnali rumorosi, come nel nostro caso.

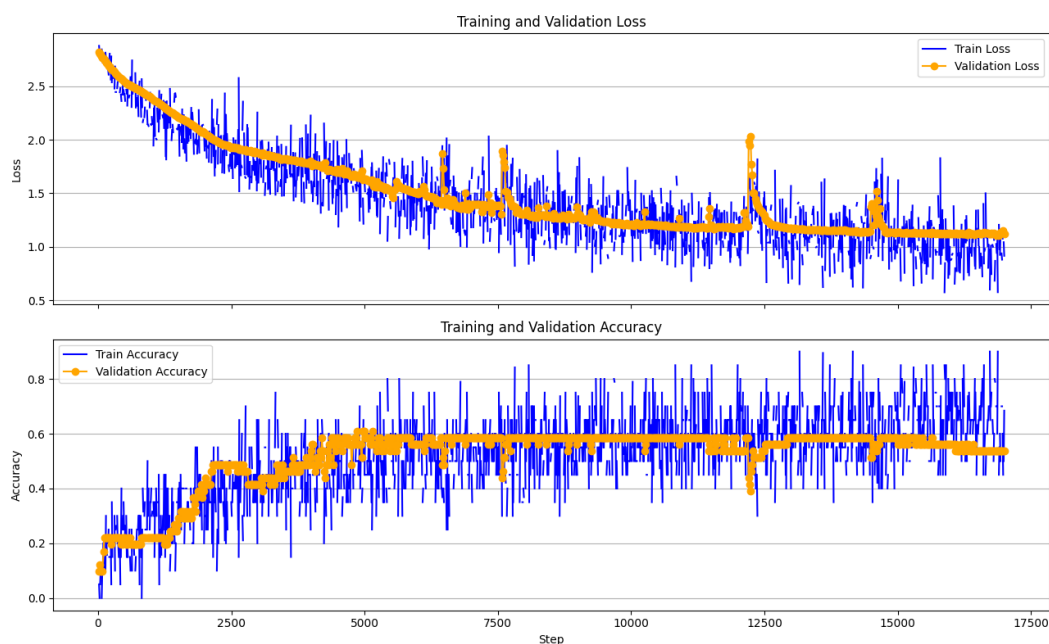


Figura 7.2 Curve di apprendimento del modello RNN GRU su Serie Temporali di lunghezza variabile.

7.5 Applicazione dell'RNN alle Finestre Temporali

In questa sezione si valuta l'efficacia dell'utilizzo del modulo GRU per la classificazione su finestre temporali di lunghezza fissa ($N = 100$), analogamente a quanto fatto nel Capitolo 3. L'obiettivo è testare se un modello con memoria temporale riesce a migliorare la classificazione delle singole finestre rispetto agli MLP utilizzati in precedenza. L'intera predizione sull'esperimento viene poi ottenuta aggregando le predizioni di ciascuna finestra tramite majority vote.

7.5.1 Risultati e Osservazioni

Dall'analisi con il modello GRU applicato a finestre temporali di lunghezza fissa ($N = 100$), e classificazione finale mediante majority voting sull'intero esperimento, emergono diversi spunti interessanti. Le matrici di confusione e le visualizzazioni delle predizioni (Figura 7.3) mostrano una performance complessivamente solida, con $E_{acc} = 0.54$, superiore a molti modelli MLP valutati a livello di finestra.

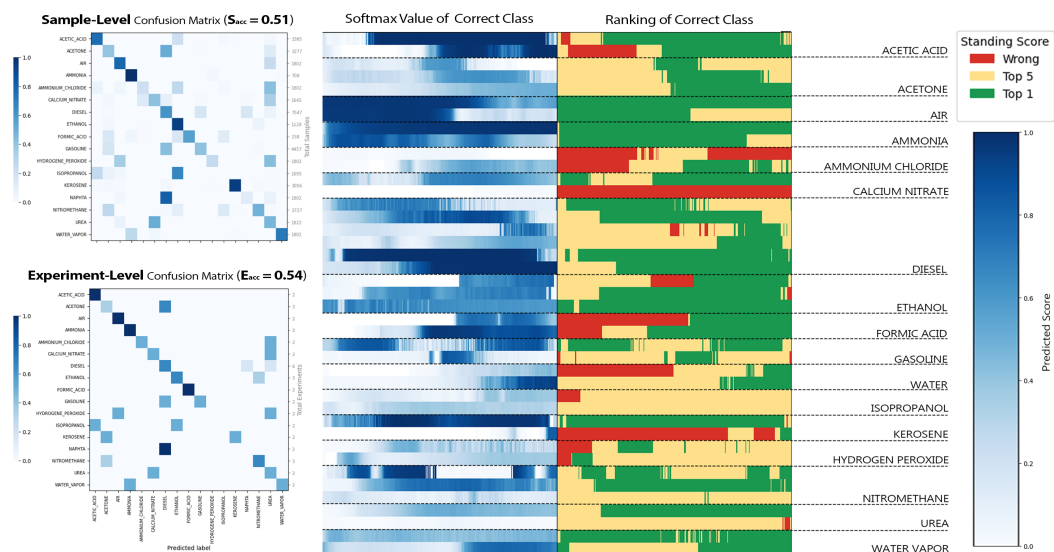


Figura 7.3 Risultati dell'applicazione dell'RNN GRU sull'approccio a finestre temporali. A sinistra le confusion matrix Sample-Level ed Experiment-level, a destra le distribuzioni temporali associate (analoghe alla figura 4.1).

Il modello GRU riesce a distinguere con particolare efficacia sostanze come KEROSENE, AIR e NITROMETHANE, dove il pattern temporale è probabilmente più distintivo o ripetitivo. Tuttavia, soffre su sostanze come CALCIUM NITRATE e AMMONIUM CHLORIDE, per le quali la memoria temporale potrebbe introdurre più ambiguità che informazione utile. Questo risultato suggerisce che l'attenzione temporale conferita dal GRU può accentuare variazioni significative del segnale per alcune classi, ma risulta più fragile quando la discriminazione richiede attenzione a dettagli stazionari o locali.

Rispetto al modello **EMLPs**, che nel capitolo sulle finestre temporali ha raggiunto $E_{acc} = 0.683$, la GRU su finestre risulta meno efficace. Tuttavia, si comporta comparabilmente al modello **MLP** (0.536).

Capitolo 8

Discussione Comparativa dei Modelli

Dopo aver presentato e analizzato i singoli approcci nei capitoli precedenti, in questo capitolo sintetizziamo i principali risultati in un confronto trasversale tra modelli, sia in termini di accuratezza che di efficienza parametrica. L'obiettivo è mettere in luce i compromessi tra complessità architetturale, dimensione in memoria e prestazioni ottenute nella classificazione delle sostanze chimiche.

Prestazioni e Complessità a Confronto

La Figura 8.1 mostra la **relazione tra dimensione del modello (in MB) e accuratezza a livello di esperimento**, mettendo a confronto i principali modelli testati nei quattro approcci: *istante temporale*, *finestre temporali*, *approssimazione RBF* e *elaborazione temporale continua* (RNN). Ogni punto rappresenta un modello, colorato in base all'approccio di appartenenza, mentre linee orizzontali identificano le performance di modelli outlier per dimensione esclusi dallo scatter per mantenere la leggibilità del grafico.

Osservazioni Comparative

L'analisi della Figura 8.1 consente alcune considerazioni chiave:

- **Approccio a finestre temporali** — I modelli in verde (basati su finestre) dominano la parte superiore del grafico, confermandosi come il miglior compromesso tra accuratezza e dimensione. Il modello **EMLPs_win** si distingue per una test accuracy del **68.3%** e una dimensione contenuta (0.138 MB). Anche i modelli **CSEMLP** offrono buone prestazioni, pur adottando architetture più compatte.
- **Approccio a istanti temporali** — I modelli blu, estremamente leggeri, sacrificano però la performance: il semplice **MLP** ottiene una test accuracy del **43.9%** con appena 9 KB, rendendolo adatto a scenari embedded ma meno competitivo in termini assoluti.

- **Approccio RBF** — I modelli arancioni offrono buoni risultati pur mantenendo una complessità ridotta. Il **MLP_64_64** raggiunge il **41.4%** con soli 42 KB. La **Random Forest**, pur simile in performance (43.9%), risulta un outlier per memoria con **3.54 MB**.
- **Approccio RNN** — I modelli in rosso, basati su sequenze complete, si collocano in una zona intermedia del grafico. Il **GRU**, con 0.067 MB, raggiunge una test accuracy del **39.0%**, mostrando buone proprietà generative pur senza superare le performance dei modelli a finestre. Il **LSTM**, pur più complesso (0.088 MB), mostra una capacità inferiore (26.8%). Anche la versione **GRU_win** (addestrata su finestre) si posiziona vicino ai CSEMLP, evidenziando la flessibilità di questo tipo di architettura.
- **Outlier per dimensione** — Alcuni modelli si collocano fuori scala rispetto al rapporto tra dimensione e accuratezza. La **Random Forest** (3.54 MB), pur avendo performance comparabili ad alcuni MLP (43.9%), presenta un peso nettamente superiore. I modelli **MLP_win** (0.414 MB) e **MLPb_win** (0.537 MB) risultano sproporzionati rispetto al guadagno in performance. In questo contesto, la **CNN**, con i suoi **0.413 MB**, si avvicina al limite superiore della scala di complessità, ma a differenza degli altri outlier mostra un margine di miglioramento: pur essendo più pesante di modelli semplici come EMLPs, raggiunge performance quasi equivalenti e potrebbe beneficiarne ulteriormente con una più attenta ottimizzazione degli iperparametri. Questo suggerisce che, seppur più costosa, una CNN ben calibrata può rappresentare una valida alternativa, soprattutto in scenari dove l'accuratezza è prioritaria rispetto alla massima compressione.

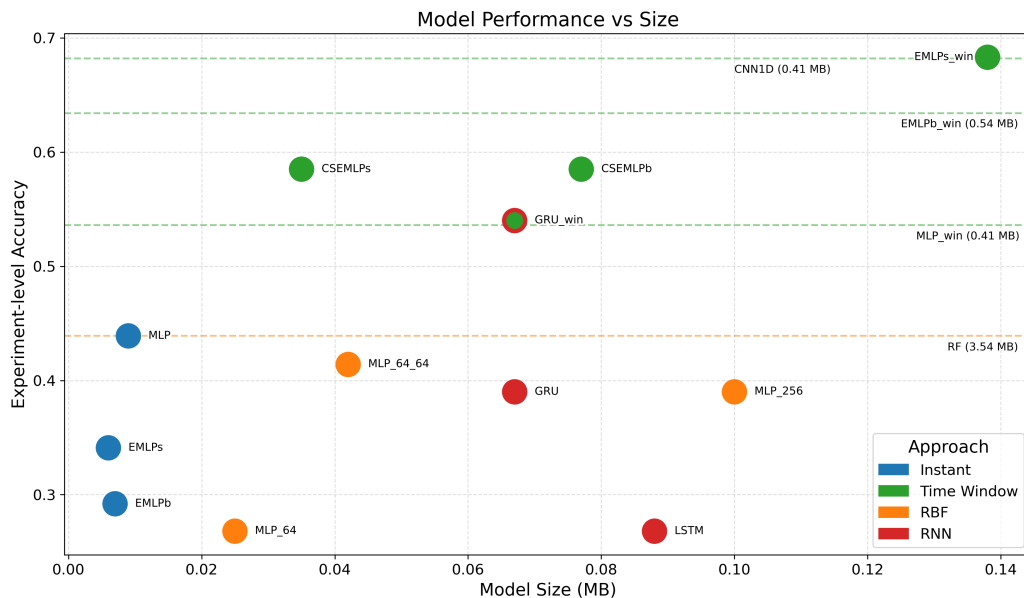


Figura 8.1 Relazione tra accuratezza a livello di esperimento e dimensione in memoria dei modelli.

Considerazioni Finali

L'approccio a **finestre temporali** si conferma il più competitivo, sia per accuratezza sia per efficienza, a fronte di modelli compatti e ben progettati. L'approccio **RBF** rappresenta una valida alternativa per scenari a bassa capacità computazionale, mentre l'approccio RNN si dimostra promettente grazie alla sua capacità di modellare l'intera dinamica temporale della sequenza, con performance simili ai modelli RBF, ma potenzialmente più robuste con quantità di dati maggiori.

Un'ultima riflessione emerge dall'osservazione delle matrici di confusione nei capitoli precedenti: i modelli tendono a favorire le classi con maggiore rappresentatività nei dati di training, penalizzando quelle sottorappresentate. Per aumentare ulteriormente la generalizzazione e l'equità nella classificazione, sarà fondamentale in futuro bilanciare il dataset aumentando il numero di esperimenti per le classi più rare.

Nel complesso, emerge chiaramente come l'informazione temporale sia un elemento centrale per la classificazione dei segnali sensoristici. A seconda del vincolo di memoria o della priorità in accuratezza, il progettista può scegliere tra soluzioni ultra-compatte (MLP) o più sofisticate e prestanti (EMLPs_win, GRU_win o CNN).

Conclusioni

Il presente lavoro ha esplorato diverse strategie di classificazione applicate a dati sensoristici complessi forniti da *Sensichips S.r.l.*, con l'obiettivo di identificare sostanze chimiche a partire da segnali multivariati rumorosi raccolti in ambienti reali. Quattro approcci principali sono stati sviluppati e confrontati:

- L'approccio **a istante temporale**, semplice ma estremamente compatto, ha mostrato che anche architetture leggere possono raggiungere risultati soddisfacenti, con una test accuracy a livello di esperimento pari al 43.9%.
- L'approccio **a finestre temporali** si è rivelato il più efficace, grazie alla sua capacità di catturare dinamiche locali del segnale: il modello **EMLPs_win** ha raggiunto la miglior performance sperimentale con un'accuratezza del 68.3%.
- L'approccio **RBF**, basato su una rappresentazione compressa delle serie temporali, ha evidenziato una buona efficacia soprattutto nei modelli MLP, pur mantenendo un'alta efficienza parametrica.
- L'approccio **con elaborazione temporale continua**, realizzato tramite RNN (GRU e LSTM), ha permesso di modellare l'intera sequenza sperimentale come un flusso temporale coerente. Il modello GRU ha mostrato un comportamento promettente, soprattutto se accostato all'approccio con finestre temporali.

Attraverso un'analisi sistematica delle prestazioni, è emersa con chiarezza l'importanza dell'informazione temporale e della scelta della rappresentazione in input. L'approccio a finestre temporali si conferma il più solido, mentre l'utilizzo di RNN offre una via alternativa in grado di valorizzare la struttura sequenziale completa dei dati, con potenzialità future ancora da esplorare.

Le confusion matrix e l'analisi qualitativa dei risultati hanno inoltre evidenziato la tendenza dei modelli a favorire le classi più rappresentate nel dataset, suggerendo come prossima direzione la gestione più accurata dello sbilanciamento tra classi per evitare bias nelle predizioni e migliorare la generalizzazione.

Questo studio dimostra come l'integrazione tra progettazione architeturale, analisi statistica e comprensione del dominio applicativo possa portare a modelli efficaci e interpretabili anche in scenari realistici.

Tutti i codici sorgente sviluppati per questa tesi, inclusi i notebook di preprocessing, addestramento e analisi, sono disponibili in forma pubblica nella seguente repository GitHub:

<https://github.com/mich1803/sensor-ts-classification>

Bibliografia

- [1] Jiayue Han, Huizi Li, Jiangong Cheng, Xiang Ma, and Yanyan Fu. Advances in metal oxide semiconductor gas sensor arrays based on machine learning algorithms. *J. Mater. Chem. C*, 13:4285–4303, 2025.
- [2] Sensichips S.r.l. sensichips.com/air-sensor, 2025. Accessed: 2025-05-21.
- [3] Emran Alotaibi and Nadia Nassif. Artificial intelligence in environmental monitoring: in-depth analysis. *Discover Artificial Intelligence*, 4(1):84, 2024.
- [4] Adebimpe Ige, Peter Adepoju, and Anfo Pub. Machine learning in industrial applications: An in-depth review and future directions. *International Journal of Multidisciplinary Research and Growth Evaluation*, 06:36–44, 12 2024.
- [5] Xia Zhao, Limin Wang, Yufei Zhang, Xuming Han, Muhammet Deveci, and Milan Parmar. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4):99, 2024.
- [6] Cesar F. Caiafa, Zhe Sun, Toshihisa Tanaka, Pere Marti-Puig, and Jordi Solé-Casals. Special issue “machine learning methods for biomedical data analysis”. *Sensors*, 23(23):9377, 2023.
- [7] Usman Yaqoob and Mohammad I. Younis. Chemical gas sensors: Recent developments, challenges, and the potential of machine learning. *Sensors*, 21(8), 2021.
- [8] Ivan Sosnovik. Leaf classification competition: 1st place winner’s interview, 2017. Accessed: 2025-05-21.
- [9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, chapter 10, pages 373–431. MIT Press, Cambridge, MA, 2016.
- [10] A. De Gaetano, S. Panunzi, F. Rinaldi, A. Risi, and M. Sciandrone. A patient adaptable ecg beat classifier based on neural networks. *Applied Mathematics and Computation*, 213(1):243–249, 2009.
- [11] Richard D Cutler, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jonathan Gibson, and Joshua J Lawler. Random forests for classification in ecology. *Ecological applications*, 17(1):81–94, 2007.
- [12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.