Utilizzo di dispositivi Internet of Things e reti neurali convoluzionali per l'analisi di correlazione tra CO2 e Rumore in ambienti domestici

Progetto per l'esame di Programmazione per l'IoT a.a. 2023-2024
Corso di Laurea Magistrale in Informatica Applicata

10/07/2024

Docente:

Prof. Emanuele Lattanzi

Candidato:

Matteo Marco Montanari

Indice della presentazione

Abstract

In questo articolo presentiamo un sistema IoT per la **raccolta e l'analisi** di dati sperimentali riguardanti i livelli di CO2 e rumore in ambienti domestici. Utilizziamo un microcontrollore **ESP32** per la raccolta e l'invio dei dati su un server IoT e una **Raspberry Pi** per l'analisi di tali dati tramite reti neurali convoluzionali.

Questo sistema IoT permette di avere un **ridotto consumo energetico** e un basso costo di realizzazione, caratteristiche che lo rendono impiegabile agevolmente anche in ambienti domestici. I risultati mostrano una correlazione tra i livelli di CO2 e rumore rilevati sperimentalmente, evidenziando l'**efficacia** del nostro sistema di registrare ed individuare le relazioni che intercorrono tra variabili di natura fisica, quali CO2 e rumore, in ambienti chiusi e controllati.

Per la raccolta e l'analisi dei dati sono stati utilizzati i seguenti **dispositivi**:

- **ESP32**: fa parte di una serie di microcontrollori System on a Chip (SoC) a basso costo e ridotto consumo energetico, dotati di Wi-Fi e Bluetooth integrati. ESP32 è prodotto e sviluppato da Espressif Systems ed è la versione migliorata del microcontrollore ESP8266.
- **Raspberry Pi Model 3**: è un Single Board Computer (SBC) sviluppato dalla Raspberry Pi Foundation ed è stato progettato per scopi didattici e di ricerca.
- **Sensore MH-Z19B (CO2)**: modulo per la rilevazione di gas tramite un sensore di piccole dimensioni che utilizza in particolare **infrarossi non dispersivi (NDIR)** per rilevare la presenza di CO2 nell'aria.
- **Sensore ARD2-2238 (rumore)**: semplice dispositivo di dimensioni ridotte dotato di un microfono per rilevare l'intensità del suono. Presenta sia un'uscita digitale che un'uscita analogica. La soglia di sensibilità del sensore può essere regolata tramite il potenziometro posizionato su di esso.

Dispositivi e software utilizzati

Software utilizzati

I **software** e framework necessari per il funzionamento del sistema IoT sono:

- **Framework Arduino**: è una piattaforma di sviluppo open source sviluppata principalmente per la programmazione di microcontrollori tramite i linguaggi **C/C++** in contesti didattici e di ricerca.
- **InfluxDB**: è un database per serie temporali (TSDB) open source sviluppato dalla società InfluxData. Viene utilizzato per l'archiviazione e il recupero di dati riguardanti serie temporali provenienti in particolare da sensori IoT.
- **Tensorflow**: è uno dei più utilizzati framework open source per il Machine Learning e l'Intelligenza Artificiale. In particolare, per eseguire su Raspberry Pi Model 3 è stato necessario utilizzare una specifica versione del framework sviluppata appositamente per dispositivi embedded chiamata **Tensorflow Lite**.
- **Google Colaboratory (Colab)**: è un servizio basato su tecnologia Jupyter Notebook ed è ospitato su **Google Cloud**, non richiede alcuna configurazione per l'utilizzo e fornisce accesso gratuito alle risorse di elaborazione, incluse GPU e TPU.

Metodi utilizzati

Raccolta dati

Per la raccolta dei dati, abbiamo utilizzato un microcontrollore ESP32 equipaggiato con un sensore di CO2 (MH-Z19B) e un sensore di rumore (ARD2-2238). I dati raccolti sono stati trasmessi in tempo reale su un **server IoT** e conservati in un database per serie temporali (**InfluxDB**).

I dati di CO2 e rumore sono stati raccolti a intervalli regolari di 5 minuti per una durata totale di circa 5 giorni. Il microcontrollore ESP32 è stato programmato tramite il framework **Arduino** per acquisire e inviare i dati tramite Wi-Fi direttamente al server IoT.

Il **circuito** utilizzato per la raccolta dati è costituito da una ESP32 alimentata a 5V e collegata direttamente ai sensori di CO2 e rumore (MH-Z19B e ARD2-2238). Nel dettaglio:

- **MH-Z19B (CO2)**: il pin V_{in} è alimentato a 5V, GND è connesso a massa. I due pin del seriale (RXD e TXD) sono connessi al seriale 2 di ESP32 (rispettivamente GPIO16 e GPIO17).
- **ARD2-2238 (rumore)**: il pin A0 (analogico) è collegato al pin GPIO32 di ESP32. Il + è connesso a 3.3V di alimentazione e G è collegato a massa (GND). Il pin digitale (D0) non viene utilizzato.

Metodi utilizzati

Raccolta dati: schema circuitale

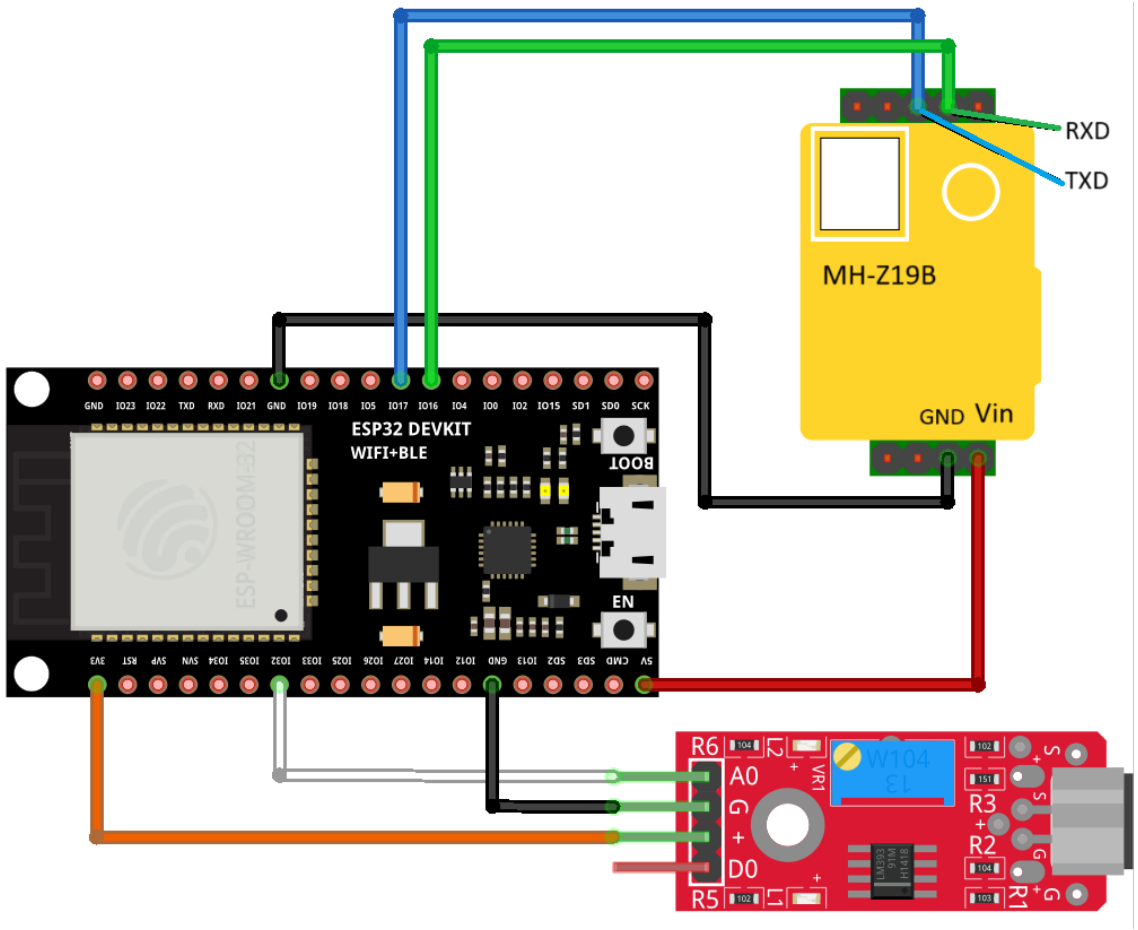


Figura: Circuito per la raccolta dati con ESP32, MH-Z19B e ARD2-2238

Metodi utilizzati

Raccolta dati: codice

La tecnica di programmazione utilizzata per il campionamento e l'invio di dati è quella tipica dei sistemi ebbedded: il **superloop**. In sintesi, il chip ESP32 esegue all'infinito una serie di istruzioni costituite da:

- misurazione dei valori di CO2 e rumore nella stanza;
- invio dei dati raccolti sul server IoT;
- attesa di 5 minuti prima di ricominciare il ciclo.

Il **codice completo** sviluppato per questo progetto è disponibile al repository pubblico GitHub: `https://www.github.com/MatteoMarcoM/Progetto_IoT/`.

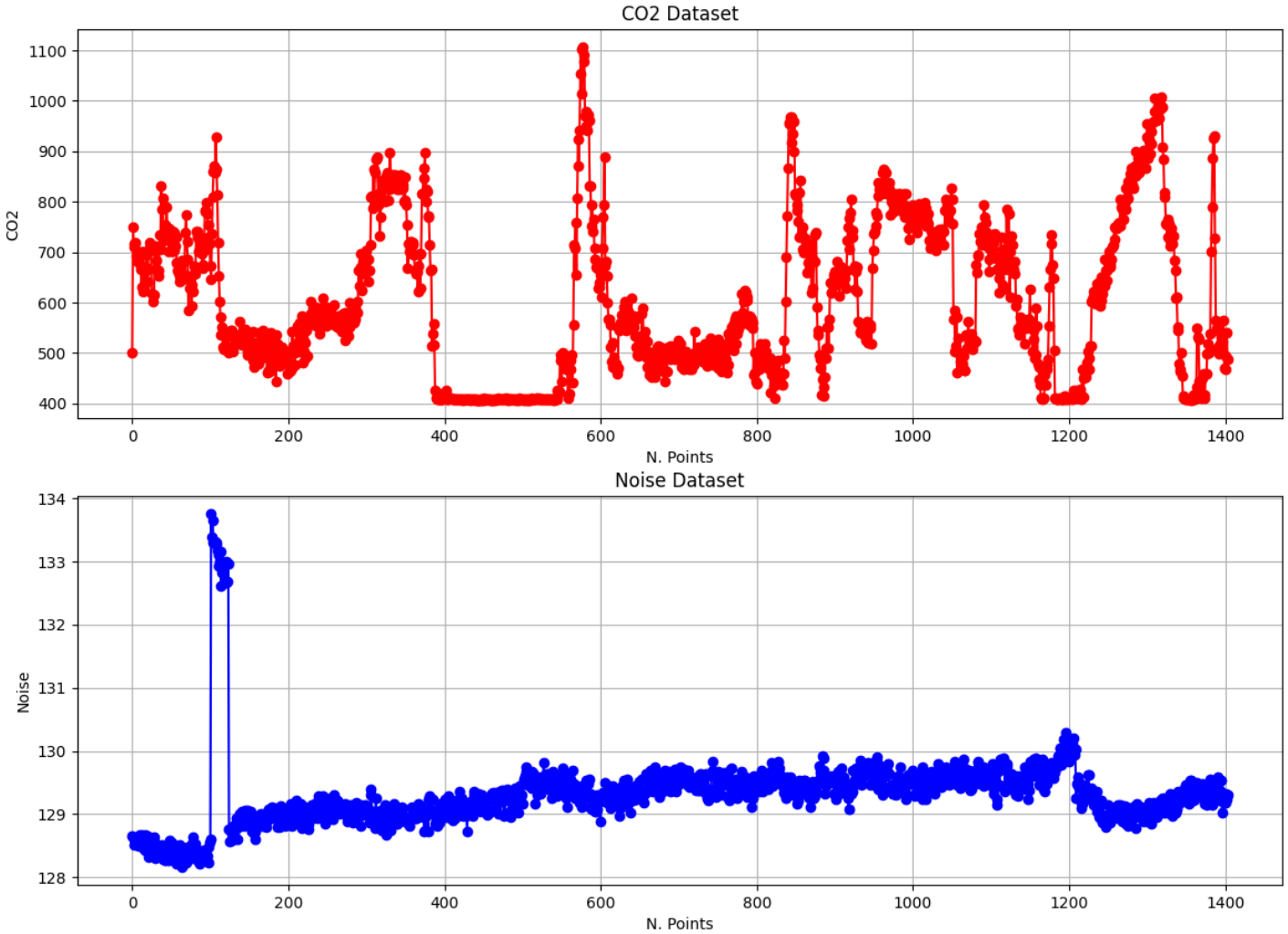
Metodi utilizzati

Raccolta dati: campionamento

Per il campionamento dei valori di CO2 viene semplicemente letto il valore del sensore MH-Z19B a intervalli di 5 minuti tramite i pin collegati alla porta seriale n. 2 (RXD2 e TXD2). Per il campionamento del rumore, invece, assumendo una **frequenza di campionamento** del segnale di 22 kHz, occorre utilizzare degli accorgimenti specifici per l'esecuzione su sistemi con **poca memoria RAM** (520 kB). Dovendo campionare a 22 kHz per 10 secondi, sono necessari in totale 22 000 samples / sec $\times$ 10 sec = 220 000 samples. Considerando che ogni sample è un valore intero a 32 bit = 4 Byte allora sono necessari $4 \times 220\,000$ B $\approx$ 880 kB di memoria RAM per raccogliere i dati di una singola misurazione, che sono più della capienza massima della RAM disponibile su ESP32. Per questo motivo, si è deciso di calcolare la **media aritmetica** predisponendo una variabile globale chiamata *somma* (inizializzata a 0) da utilizzare come **somma cumulativa** dei campioni ottenuti fino a quel momento. Per ottenere il valore medio di rumore, da inviare al server IoT, occorre poi dividere la somma di tutti i campioni per il numero di samples acquisiti (220 000). Dovendo campionare a 22 kHz, il **delay** tra un sample e l'altro è di 1/22 000 sec $\approx$ 0.000 045 sec = 45 μ sec.

Metodi utilizzati

Raccolta dati: grafici in serie temporali



Metodi utilizzati

Elaborazione dei dati

Per lo studio di correlazione tra le due variabili fisiche analizzate abbiamo utilizzato un metodo di **Machine Learning** basato su reti neurali artificiali. I dati sperimentali ottenuti dai sensori sono stati utilizzati per addestrare delle **reti neurali** allo scopo di individuare una relazione funzionale che rappresenti la correlazione tra le due variabili.

Metodi utilizzati

Elaborazione dei dati: definizione del modello

Per mantenere intatta la **natura temporale** dei dati raccolti, si è deciso di predire un singolo valore di una delle due variabili a partire da una sequenza di λ valori consecutivi dell'altra variabile. Assumendo quindi che il valore di CO2 sia la variabile indipendente mentre il rumore (*noise*) quella dipendente, l'obiettivo principale è stato di far apprendere alla rete neurale la **relazione funzionale** (dipendente da parametri Θ), $f_\Theta : \mathbb{R}^\lambda \mapsto \mathbb{R}$, intercorrente tra i dati raccolti sperimentalmente:

$$\text{noise} = f_\Theta(\text{CO2}^{(1)}, \ldots, \text{CO2}^{(\lambda)}) \tag{1}$$

Simmetricamente, assumendo invece che il rumore (*noise*) sia la variabile indipendente e la CO2 quella dipendente, l'**obiettivo secondario** è stato di far apprendere alla rete neurale la relazione funzionale (dipendente da parametri Φ), $h_\Phi : \mathbb{R}^\lambda \mapsto \mathbb{R}$, intercorrente tra i dati raccolti sperimentalmente:

$$\text{CO2} = h_\Phi(\text{noise}^{(1)}, \ldots, \text{noise}^{(\lambda)}) \tag{2}$$

Il **dataset** utilizzato per addestrare le reti neurali è composto dalla successione di tutte le coppie costituite dai valori di CO2 e rumore ottenuti sperimentalmente (al medesimo istante di tempo) tramite il sistema IoT da noi sviluppato. In simboli:

$$DataSet := \{(CO2_i, noise_i)\}_{i=1}^{N} \tag{3}$$

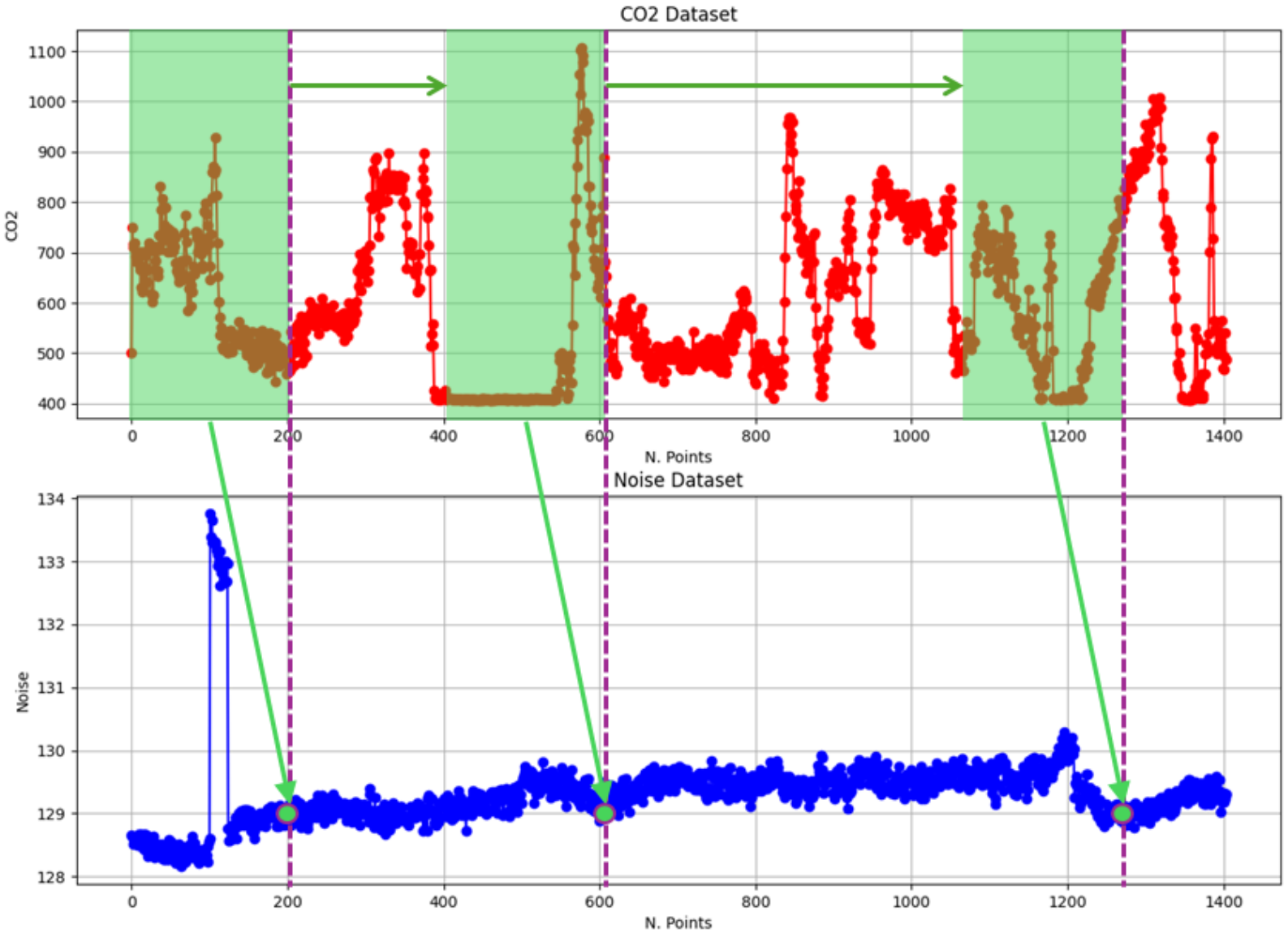
Data la **natura temporale** dei dati raccolti, considerando il modello f_Θ, il dataset deve essere preprocessato nel modo seguente prima di essere fornito in input alla rete neurale (se si utilizza il modello h_Φ, basta invertire $CO2_i$ con $noise_i$).

$$X_k := \{CO2_i\}_{i=k}^{k+\lambda-1} \qquad (k = 1, \dots N - \lambda) \tag{4}$$

$$y_k := [noise_i]_{i=k+\lambda} \qquad (k = 1, \dots N - \lambda) \tag{5}$$

Metodi utilizzati

Elaborazione dei dati: data preprocessing



Metodi utilizzati

Elaborazione dei dati: addestramento della rete

Pertanto, lo scopo delle reti neurali implementate per l'analisi della correlazione tra CO2 e rumore è quello di risolvere un problema di **regressione statistica** (non lineare): individuare una relazione funzionale (dipendente da parametri) tra variabili misurate sulla base di dati sperimentali. Questo problema, nell'ambito del Machine Learning, è un esempio di **apprendimento supervisionato** che prevede di minimizzare una opportuna funzione di costo (**Mean Squared Error**) per addestrare la rete neurale e individuare i **parametri ottimi** delle funzioni modello (Θ^*, Φ^*).

Dopo l'addestramento, la rete neurale assumerà il ruolo di **modello parametrizzato** in base al valore dei dati sperimentali acquisiti: $NN^{\Theta} \equiv f_{\Theta^*}(\cdot)$ oppure $NN^{\Phi} \equiv h_{\Phi^*}(\cdot)$. Fornendo in input alla rete neurale una sequenza di λ dati ricavati sperimentalmente (di CO2 o rumore a seconda del modello utilizzato) otterremo il singolo **valore predetto** corrispondente (rumore o CO2 rispettivamente) all'istante di tempo successivo all'ultimo campione di input.

Metodi utilizzati

Elaborazione dei dati: Convolutional Neural Networks

Le **reti neurali convoluzionali (CNN)** sono una classe di reti neurali profonde (**deep learning**) particolarmente adatte per l'elaborazione di dati sequenziali come le serie temporali. A differenza delle classiche reti **Feed-Forward**, le CNN consentono di imparare automaticamente le features (**feature extraction**) tramite l'utilizzo di filtri convoluzionali chiamati **kernel**, che vengono applicati ripetutamente alle varie porzioni dei dati di input. Applicando più **layer convoluzionali** in cascata, è possibile estrarre features di *alto livello* a partire da caratteristiche più semplici individuate nei dati di input. Per il nostro studio abbiamo dunque utilizzato le librerie **TensorFlow** e **Keras** per costruire due reti neurali di tipo Convolutional Neural Network (CNN) con lo scopo di individuare una relazione funzionale che rappresenti la correlazione tra le due variabili tramite un task di regressione su dati sperimentali.

Metodi utilizzati

Elaborazione dei dati: architettura CNN1

L'architettura che **empiricamente** risulta la migliore per l'apprendimento del modello f_Θ, che rappresenta la correlazione tra i valori di CO2 e rumore in base ai dati prodotti dal nostro esperimento, è la seguente (**CNN1**):

- Layer di **Input** di dimensione $(\lambda, 1)$, con $\lambda = 40$.
- Layer **Convoluzionale 1D** con 64 kernel di dimensione 3 e funzione di attivazione lineare.
- Layer **Convoluzionale 1D** con 32 kernel di dimensione 2 e funzione di attivazione lineare.
- Layer **Convoluzionale 1D** con 32 kernel di dimensione 2 e funzione di attivazione lineare.
- Layer **Flatten** per adattare l'output degli strati convoluzionali agli strati densi finali.
- Layer **Denso** con 30 neuroni e funzione di attivazione lineare.
- Layer **Denso** con 1 neurone (per task di regressione) e funzione di attivazione lineare.

Metodi utilizzati

Elaborazione dei dati: architettura CNN2

Di contro, l'architettura che empiricamente risulta la migliore per l'**apprendimento del modello** h_Φ, che rappresenta la correlazione tra i valori di rumore e CO2 in base ai dati prodotti dal nostro esperimento, è la seguente (**CNN2**):

- Layer di **Input** di dimensione $(\lambda, 1)$, con $\lambda = 40$.
- Layer **Convoluzionale 1D** con 64 kernel di dimensione 3 e funzione di attivazione lineare.
- Layer **Convoluzionale 1D** con 32 kernel di dimensione 2 e funzione di attivazione lineare.
- Layer **Flatten** per adattare l'output degli strati convoluzionali agli strati densi finali.
- Layer **Denso** con 30 neuroni e funzione di attivazione lineare.
- Layer **Denso** con 1 neurone (per task di regressione) e funzione di attivazione lineare.

Metodi utilizzati

Elaborazione dei dati: esecuzione su Raspberry Pi

L'esecuzione dei modelli, precedentemente addestrati in Cloud tramite **Google Colaboratory**, è stata effettuata su un dispositivo embedded tipico dei sistemi IoT: **Raspberry Pi Model 3**. Il codice utilizzato per implementare le reti neurali è scritto in linguaggio **Python 3** e si basa sulle librerie di calcolo numerico: **NumPy, Matplotlib, Pandas** e sui framework di Intelligenza Artificiale e Machine Learning **TensorFlow e Tensorflow Lite**.

Problema Diretto La prima rete neurale da noi sviluppata (**CNN1**) è stata in grado di predire i valori di rumore a partire dai dati sperimentali di CO2 immediatamente precedenti, collezionati in una finestra temporale di $\lambda = 40$ punti consecutivi. Questi risultati mostrano l'*effettiva capacità* del nostro modello di rete neurale di **apprendere** la relazione funzionale corrispondente al modello f_Θ che individua la **correlazione** tra CO2 e rumore.

Problema Inverso Simmetricamente, considerando il problema di predire i valori di CO2 a partire dai dati di rumore immediatamente precedenti, la seconda rete neurale da noi sviluppata (**CNN2**) *non è stata così efficace* a predire tali valori (si è utilizzata una finestra temporale di $\lambda = 40$ punti consecutivi). Questi risultati mostrano una scarsa capacità del nostro modello di rete neurale di **apprendere** la relazione funzionale corrispondente al modello h_Φ, che dovrebbe individuare la **correlazione** tra rumore e CO2.

Risultati e conclusioni

Problema diretto: risultati CNN1

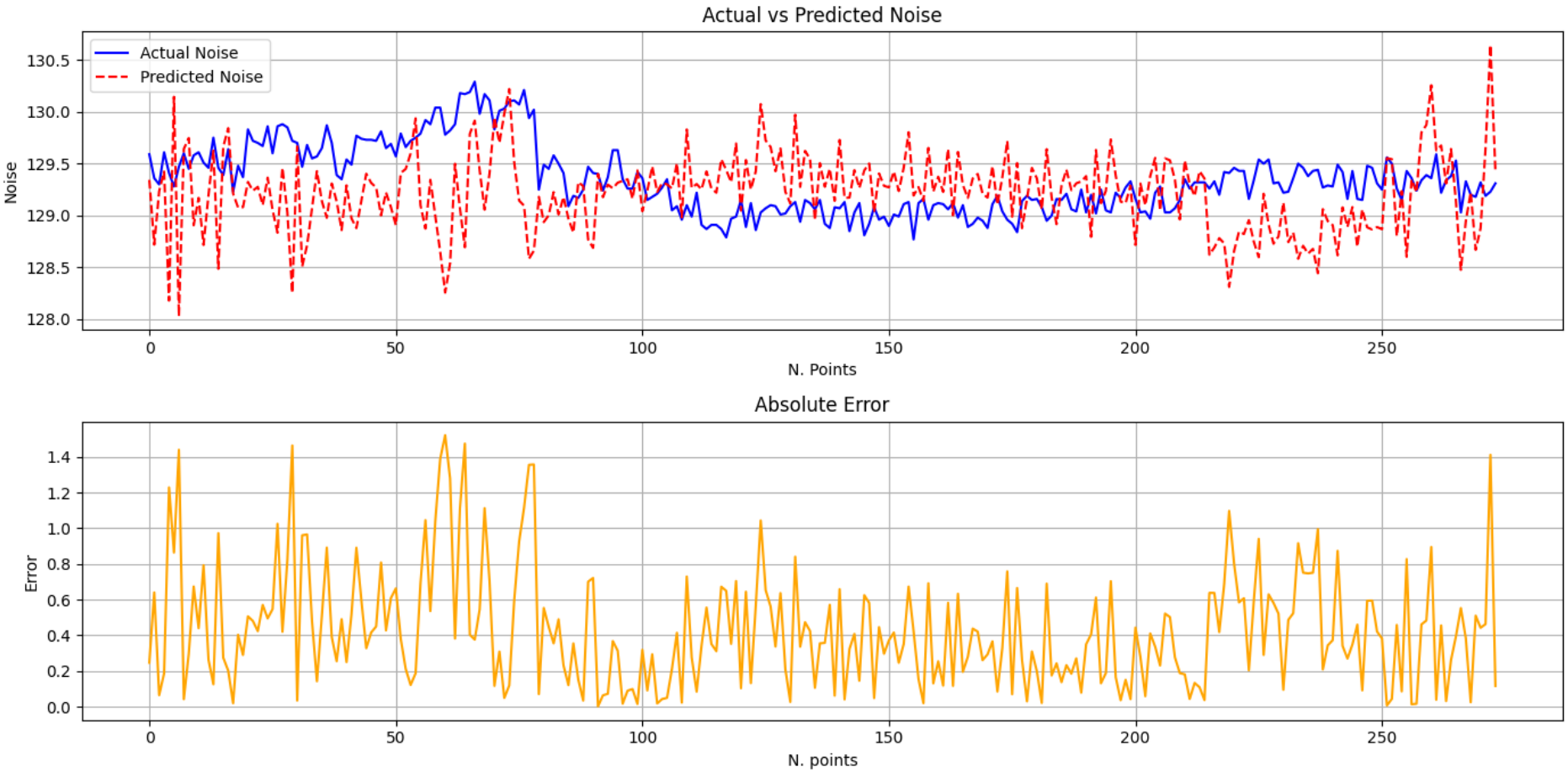


Figura: (**Problema diretto**) Andamento dei valori di rumore, predetti ed effettivi, relativi ai dati di test e grafico dell'errore assoluto.

Risultati e conclusioni

Problema inverso: risultati CNN2

Figura: (**Problema inverso**) Andamento dei valori di CO2, predetti ed effettivi, relativi ai dati di test e grafico dell'errore assoluto.

La Tabella seguente mostra le due **metriche di valutazione** del modello per le due istanze di rete neurale addestrate (corrispondenti al problema diretto e inverso). MSE è la Mean Squared Error (**loss function**) mentre la **Mean Absolute Error (MAE)** è la media degli errori assoluti calcolata sui dati di test (i valori di input *non* sono stati *normalizzati*).

NN	MSE	MAE
CNN1	0.294	0.432
CNN2	33444.141	154.797

Gli **scarsi risultati** della CNN2 sono dovuti, a nostro avviso, alla **bassa sensibilità** del sensore di rumore che ha prodotto di conseguenza dei dati che seguono un andamento pressocché lineare da cui è molto difficile ricavare i valori di CO2 ottenuti sperimentalmente, che sono invece molto variabili.

Ulteriori sviluppi potrebbero riguardare l'utilizzo di altre tipologie di reti neurali per l'analisi di correlazione (nonché di altre tecniche di Machine Learning classiche), oltre la possibilità di sperimentare l'utilizzo di ulteriori dispositivi e sensori per la rilevazione di altre grandezze fisiche allo scopo di suggerire analisi di correlazione più complesse che le coinvolgano. In particolare, sembra promettente l'utilizzo delle reti neurali ricorrenti come le **Long Short Term Memory (LSTM)**, un'architettura progettata specificamente per modellare le dipendenze a lungo termine su dati sequenziali. Un iniziale tentativo in questo senso è stato fatto ma ancora non ci è stato possibile implementare tali tipologie di reti su Raspberry Pi Model 3 per motivi di **compatibilità** col framework **Tensorflow Lite**, cosa che impone lo spostamento su dispositivi diversi o l'utilizzo di hardware più recenti.

1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Grazie per l'attenzione