MATR. N. 18164

UNIVERSITÀ CAMPUS BIO-MEDICO DI ROMA

FACOLTÀ DIPARTIMENTALE DI INGEGNERIA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA DEI SISTEMI INTELLIGENTI

DEEP LEARNING FOR BIGDATA

PROGETTO ANOMALY DETECTION & PREDICTION

Studenti:
Francesco Pinsone
Claudia Marano
Edoardo Caliano

Anno Accademico 2023/2024

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Inquadramento del Problema

Il problema affrontato in questo studio riguarda la rilevazione e la previsione di anomalie in un sistema industriale simulato (testbed) che riproduce un processo fisico di riempimento e svuotamento di tank.
Il testbed è composto da un insieme di componenti fisici:

- **8 tank**: Tank1, Tank2, Tank3, Tank4, Tank5, Tank6, Tank7, Tank8
- **6 pompe**: Pump1, Pump2, Pump3, Pump4, Pump5, Pump6
- **4 sensori di flusso**: Flowsensor1, Flowsensor2, Flowsensor3, Flowsensor4
- **22 valvole**: Valv 1, Valv 2, Valv 3, Valv 4, Valv 5, Valv 6, Valv 7, Valv 8, Valv 9, Valv 10, Valv 11, Valv 12, Valv 13, Valv 14, Valv 15, Valv 16, Valv 17, Valv 18, Valv 19, Valv 20, Valv 21, Valv 22

Questi componenti sono interconnessi tramite il protocollo di comunicazione Modbus e gestiti da un Programmable Logic Controller (PLC).
Nel sistema, i sensori raccolgono dati relativi al riempimento e svuotamento dei serbatoi, al flusso attraverso le pompe e alla posizione delle valvole (apertura/chiusura).
In condizioni operative normali, il processo segue un comportamento ciclico ben definito: le pompe e le valvole si attivano in sequenza per garantire un flusso regolare, e i serbatoi si riempiono e si svuotano secondo una logica predeterminata.
Tuttavia, in caso di un attacco cibernetico, il comportamento del sistema viene alterato. L'apertura e la chiusura delle pompe e delle valvole seguono un ciclo anomalo, causando deviazioni significative nel normale funzionamento del testbed. Queste anomalie, che si manifestano sia a livello fisico che di rete, vengono indicate nel dataset con una

label pari a 1, mentre i comportamenti normali sono etichettati con una label pari a 0. Il dataset fornito contiene due categorie di dati:

- **Dati fisici**: informazioni relative al riempimento, svuotamento e al flusso all'interno del sistema.
- **Dati di Rete**: comunicazioni registrate tramite il protocollo Modbus tra il PLC e i componenti del testbed.

L'obiettivo dello studio è sviluppare modelli in grado di:

- **Rilevare anomalie (Anomaly Detection)**: identificare istantaneamente comportamenti anomali nel sistema.
- **Prevedere anomalie (Anomaly Prediction)**: anticipare la comparsa di anomalie con almeno un timestamp di preavviso.

Per raggiungere questi obiettivi, saranno utilizzati sia modelli di rete densa che modelli specifici per serie temporali, valutando le performance su entrambe le tipologie di dati forniti.

1.2 Descrizione del Dataset

Il dataset è composto da un insieme di file in formato CSV che fanno riferimento a diverse tipologie di misurazioni o estrazioni di informazioni relative ad un impianto dotato di strumentazione di vario tipo: pompe, valvole, serbatoi e sensori di flusso. Questi file possono quindi contenere record relativi a misurazioni fisiche effettuate sull'impianto oppure record relativi ai pacchetti appartenenti al traffico di rete intercettato in entrata e in uscita dall'impianto.

1.2.1 File con Misurazioni Fisiche

Rappresentano dati di misurazioni fisiche relativi a un impianto, con rilevazioni su serbatoi, pompe, sensori di flusso e valvole. Ogni riga del file corrisponde a un'istantanea dello stato dell'impianto in un determinato momento.

Descrizione dei Campi

- `Time`: Timestamp che indica l'istante della misurazione. Formato: `DD/MM/YYYY HH:MM:SS`.

- `Tank_1, ..., Tank_8`: Stato o livello dei serbatoi (valori numerici).
- `Pump_1, ..., Pump_6`: Stato delle pompe (valori booleani: `true` per attivo, `false` per inattivo).
- `Flow_sensor_1, ..., Flow_sensor_4`: Letture dei sensori di flusso (valori numerici).
- `Valv_1, ..., Valv_22`: Stato delle valvole (valori booleani: `true` per aperto, `false` per chiuso).
- `Label_n`: Etichetta numerica associata al record per analisi (ad esempio, categorizzazione o classificazione).
- `Label`: Etichetta testuale che descrive lo stato generale (ad esempio, `normal` per condizioni normali).

Osservazioni

- I dati rappresentano uno stato statico dell'impianto in un istante specifico.
- Le variabili booleane (`true`/`false`) consentono di analizzare il comportamento di pompe e valvole in relazione agli altri componenti.
- `Label_n` e `Label` sono utili per classificazioni e analisi di tipo supervisionato (ad esempio, rilevamento di anomalie).

1.2.2 File con Pacchetti di Rete

Rappresentano dati sul traffico di rete con informazioni relative a pacchetti, connessioni e protocolli. Ogni riga corrisponde a un record che contiene dettagli relativi a un pacchetto di rete.

Descrizione dei Campi

- `Time`: Timestamp che indica l'istante in cui il pacchetto è stato registrato. Formato: `YYYY-MM-DD HH:MM:SS.ssssss`.
- `mac_s`: Indirizzo MAC sorgente del pacchetto.
- `mac_d`: Indirizzo MAC di destinazione del pacchetto.
- `ip_s`: Indirizzo IP sorgente del pacchetto.

- `ip_d`: Indirizzo IP di destinazione del pacchetto.
- `sport`: Porta sorgente del pacchetto.
- `dport`: Porta di destinazione del pacchetto.
- `proto`: Protocollo utilizzato (ad esempio, `Modbus`).
- `flags`: Flag di controllo TCP/UDP associati al pacchetto, rappresentati come una sequenza binaria.
- `size`: Dimensione del pacchetto in byte.
- `modbus_fn`: Funzione del protocollo Modbus (ad esempio, `Read Coils Request` o `Read Coils Response`).
- `n_pkt_src`: Numero di pacchetti inviati dalla sorgente.
- `n_pkt_dst`: Numero di pacchetti inviati alla destinazione.
- `modbus_response`: Contenuto della risposta Modbus (ad esempio, `[0]` indica una risposta specifica o `N/A` se non applicabile).
- `label_n`: Etichetta numerica associata al record per analisi (sembra usata per classificazione o categorizzazione).
- `label`: Etichetta testuale del record (ad esempio, `normal` per traffico regolare).

Osservazioni

- I dati sono dettagliati e permettono di analizzare il traffico di rete su più livelli (MAC, IP, porta, protocollo).
- La presenza di campi come `modbus_fn` e `modbus_response` suggerisce che il file sia stato catturato in un ambiente industriale dove il protocollo Modbus è utilizzato per la comunicazione tra dispositivi.
- `label_n` e `label` sono utili per analisi di tipo supervisionato, come il rilevamento di anomalie o attacchi.

Capitolo 2

Anomaly Detection e Prediction con Dense Neural Network

2.1 Anomaly Detection

Nel seguente paragrafo vengono discusse le modalità di risoluzione e lo schema di processo adottato per l'implementazione del task di *Anomaly Detection* mediante l'utilizzo di **reti neurali**. Si è scelto inoltre di non accorpare il dataset fisico e di rete implementando quindi uno script per fare detection sul dataset fisico e uno per fare detection sul dataset di rete.

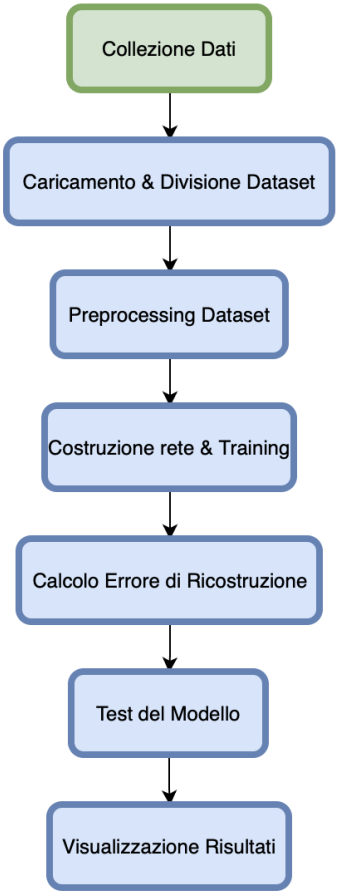

Figura 2.1: Schema del processo di Anomaly Detection

Come è possibile immaginare dalla precedente figura i due script, lavorando su due diversi dataset, differiscono solo per il preprocessing.

2.1.1 Physical Detection

Il è stato eseguito attraverso l'utilizzo di un approccio basato su *autoencoder*. Il modello è stato addestrato esclusivamente su dati normali, consentendo di identificare comportamenti anomali in fase di test attraverso l'errore di ricostruzione.

Caricamento e Segmentazione del Dataset

Il dataset è stato caricato e suddiviso in intervalli di un minuto. Ogni intervallo rappresenta un blocco temporale omogeneo dei dati. Durante questa fase:

- Caricamento del dataset che non contiene misurazioni corrispondenti a situazioni anomale (`phy_norm.csv`).
- Ordinamento dei dati in base ai timestamp per garantire coerenza temporale.
- Gli intervalli non conformi (ovvero con un numero di campioni diverso da 60) sono stati esclusi per mantenere l'uniformità.

Preprocessing

I dati segmentati sono stati concatenati in un unico array per consentire la normalizzazione tramite *StandardScaler*. Dopo la normalizzazione, i dati sono stati nuovamente suddivisi nei segmenti originali. Questo approccio garantisce che i dati di input per il modello siano privi di bias derivanti da scale diverse.

Costruzione e Addestramento del Modello

Il modello utilizzato per il rilevamento delle anomalie è un *autoencoder* composto da:

- Strati di codifica con funzioni di attivazione *LeakyReLU*.
- *Batch Normalization* per accelerare l'addestramento e migliorare la stabilità.
- *Dropout* per prevenire l'overfitting.
- Strati di decodifica simmetrici per ricostruire l'input originale.

L'addestramento è stato eseguito esclusivamente su segmenti normali con un *loss function* basato sull'errore quadratico medio (*MSE*). È stato utilizzato un meccanismo di *early stopping* per interrompere l'addestramento in caso di mancati miglioramenti nella validazione.

Calcolo dell'Errore di Ricostruzione

Per definire una soglia di anomalia:

- È stato calcolato l'errore medio di ricostruzione (*MSE*) sui dati di addestramento.
- La soglia è stata fissata al 95° percentile dell'errore di ricostruzione, considerando come anomali i segmenti con errore superiore a tale soglia.

Test del Modello

Il modello è stato testato su un dataset contenente anomalie (`phy_att_3.csv`). Durante questa fase:

- I dati di test sono stati segmentati e normalizzati utilizzando lo stesso *scaler* impiegato per i dati di addestramento.
- L'autoencoder ha ricostruito i segmenti di test, e l'errore di ricostruzione è stato calcolato per ciascun segmento.
- I segmenti con errore superiore alla soglia sono stati classificati come anomali.

Risultati

Il modello è stato in grado di rilevare blocchi anomali nei dati di test con successo, identificando le anomalie sulla base del comportamento osservato durante l'addestramento.

Conclusione

Questo approccio basato su *autoencoder* ha dimostrato l'efficacia nel rilevamento di anomalie fisiche nel dataset. La metodologia utilizzata può essere estesa ad altri scenari con opportuni adattamenti ai dati e al modello.

2.1.2 Network Detection

Di seguito si descrivere il processo implementato per rilevare anomalie nei dati di rete. Il file Python contiene uno script che utilizza un autoencoder, una rete neurale non supervisionata, per identificare intervalli di tempo con comportamenti anomali rispetto al traffico normale.

Dataset & Preprocessing

Il caricamento e la segmentazione del dataset avvengono come segue:

- **Input**: Il dataset originale contiene pacchetti di rete annotati con timestamp, informazioni di protocollo, indirizzi di rete e altre caratteristiche rilevanti.
- **Suddivisione**: Il dataset viene suddiviso in intervalli temporali di 100 millisecondi. Ogni intervallo costituisce un segmento che rappresenta una finestra temporale di traffico di rete.

- **Filtraggio**: Vengono mantenuti solo i segmenti con esattamente 200 campioni per garantire uniformità nei dati di input.

Normalizzazione

Le caratteristiche dei segmenti vengono preprocessate utilizzando un *ColumnTransformer* che:

1. Esegue l'**One-Hot Encoding** delle variabili categoriali.
2. Applica una **MinMax Normalization** alle variabili numeriche.

I segmenti normalizzati sono quindi divisi e preparati per l'addestramento.

Struttura del Modello Autoencoder

L'autoencoder è progettato per apprendere una rappresentazione compressa dei dati normali e rilevare deviazioni significative. La struttura è composta da:

- **Encoder:** Una sequenza di strati densi con:
  - Attivazione *LeakyReLU* per gestire gradienti piccoli.
  - *Batch Normalization* per migliorare la convergenza.
  - *Dropout* per prevenire l'overfitting.
- **Decoder:** Ricostruisce i dati originali partendo dalla rappresentazione codificata.

L'autoencoder viene addestrato esclusivamente su dati senza anomalie.

Calcolo Treshold

Dopo l'addestramento:

1. L'errore di ricostruzione (MSE) è calcolato su tutti i segmenti normali.
2. Una **soglia** è definita utilizzando il **95° percentile** dell'errore di ricostruzione. I segmenti con un errore superiore a questa soglia sono considerati anomali.

Test su Dati Anomali

Il modello viene testato su un dataset separato contenente anomalie. Il processo include:

1. Segmentazione e preprocessing dei dati di test, seguendo lo stesso schema dei dati di addestramento.
2. Predizione utilizzando l'autoencoder.
3. Calcolo dell'errore di ricostruzione per ciascun segmento.
4. Identificazione delle anomalie confrontando l'errore di ricostruzione con la soglia predefinita.

Risultati

- **Errore di ricostruzione (MSE):** Gli errori di ricostruzione sono analizzati per distinguere tra comportamenti normali e anomalie.
- **Anomalie rilevate:** È stato quantificato il numero totale di segmenti anomali rilevati dal modello.

Conclusioni

Questo approccio sfrutta l'autoencoder per apprendere il comportamento normale del traffico di rete. Il modello è in grado di rilevare anomalie basandosi esclusivamente sulle deviazioni nell'errore di ricostruzione, dimostrando la sua efficacia nel task di rilevamento di anomalie nel dataset di rete.

2.2 Anomaly Prediction

Nel seguente paragrafo vengono discusse le modalità di risoluzione per il task di *Anomaly Prediction* mediante l'utilizzo di **Dense NN**. Come per il caso di Anomaly Prediction, si è scelto di non unire il dataset fisico e quello di rete, andando a sviluppare uno script per la parte physical e uno per la parte network.

2.2.1 Physical Prediction

L'analisi per la previsione di anomalie nel dataset fisico è stata effettuata implementando una pipeline basata su Dense Neural Networks (DNN). Di seguito vengono riportati i dettagli delle fasi principali del processo e i risultati ottenuti.

Caricamento e Segmentazione del Dataset

Il dataset è stato caricato e segmentato in intervalli temporali di un minuto. Ogni segmento rappresenta un insieme di osservazioni omogenee nel tempo. Durante questa fase sono stati seguiti i seguenti passi:

- **Caricamento del dataset**: I dati normali (`phy_norm.csv`) e di attacco (`phy_att_*.csv`) sono stati importati e uniti in un unico dataset.
- **Pulizia e Ordinamento**: I dati sono stati ordinati temporalmente sulla base della colonna `Time`, assicurando coerenza temporale per ogni segmento.
- **Filtraggio**: Solo i segmenti con una lunghezza esattamente pari a 60 campioni sono stati mantenuti, garantendo uniformità nei dati.
- **Validazione delle Etichette**: Sono state mantenute solo le osservazioni con etichette valide (0 per dati normali e 1 per dati anomali).

Preprocessing

Dopo la segmentazione, i dati sono stati preprocessati per garantire un input appropriato al modello. Durante questa fase:

- Le feature sono state normalizzate utilizzando `StandardScaler`, riducendo ogni feature a una distribuzione con media 0 e deviazione standard 1.
- I segmenti sono stati ricostruiti mantenendo le etichette originali (`label_n`), preservando l'integrità delle informazioni per il modello.

Preparazione dei Dati per il Modello

I dati preprocessati sono stati preparati per l'addestramento e la valutazione del modello:

- **Caratteristiche e Target**: Ogni segmento è stato trasformato in un array unidimensionale (*flattened*) per l'input della rete, con la colonna `label_n` come target.
- **Bilanciamento**: I dati sono stati suddivisi in training (70%), validation (15%), e test set (15%) mantenendo la distribuzione originale delle classi.

Architettura del Modello

Il modello utilizzato è una rete DNN progettata con i seguenti strati:

- **Input Layer**: Configurato per accettare i dati preprocessati.
- **Hidden Layers**: Due strati densi con:
  - Funzione di attivazione `ReLU`.
  - Dropout del 30% per prevenire overfitting.
- **Output Layer**: Uno strato denso con funzione di attivazione sigmoide per produrre una classificazione binaria.

Il modello è stato compilato utilizzando la funzione di perdita `binary_crossentropy` e l'ottimizzatore `Adam` con un learning rate di 0.001.

Addestramento

L'addestramento è stato eseguito per un massimo di 50 epoche utilizzando:

- **Batch Size**: 32.
- **Early Stopping**: Per interrompere l'addestramento in caso di mancati miglioramenti nel validation loss dopo 5 epoche consecutive.

Risultati

I risultati principali del modello sul test set sono riportati di seguito:

- **Confusion Matrix:**

$$\begin{bmatrix} 31 & 0 \\ 5 & 0 \end{bmatrix}$$

- **Classification Report:**

Classe	Precision	Recall	F1-score
Normale (0)	0.86	1.00	0.93
Anomalia (1)	0.00	0.00	0.00

Tabella 2.1: Classification Report per le classi Normale (0) e Anomalia (1).

- **ROC-AUC Score:** 0.5000

Discussione

Il modello ha mostrato una capacità eccellente di identificare comportamenti normali, con precisione e recall pari a 1.00 per la classe normale. Tuttavia, non è stato in grado di rilevare anomalie, come evidenziato dai valori pari a 0.00 per la precisione e il recall della classe anomala. Questo risultato può essere attribuito alla significativa predominanza di esempi normali nel dataset, che ha reso difficile l'apprendimento delle caratteristiche associate alle anomalie.

Conclusione

Questo esperimento ha dimostrato che l'approccio basato su Dense Neural Networks può essere efficace nel classificare comportamenti normali, ma richiede ulteriori adattamenti per migliorare la capacità di rilevamento delle anomalie. Possibili estensioni future includono:

- L'uso di tecniche di bilanciamento del dataset per mitigare lo squilibrio di classe.
- L'adozione di modelli ibridi che combinano approcci supervisionati e non supervisionati per migliorare il rilevamento delle anomalie.

2.2.2 Network Prediction

L'analisi delle anomalie nel dataset di rete è stata condotta utilizzando una pipeline di previsione basata su *Dense Neural Networks (DNN)*. Di seguito vengono descritti i dettagli delle fasi principali del processo, i risultati ottenuti e una discussione sulle criticità riscontrate.

Dataset e Preprocessing

Il dataset utilizzato comprendeva traffico di rete registrato in un ambiente industriale, con informazioni relative a timestamp, indirizzi MAC e IP, porte di comunicazione, protocolli, dimensioni dei pacchetti, e informazioni specifiche del protocollo Modbus. Durante la fase di preprocessing sono state implementate le seguenti operazioni:

- **Analisi preliminare delle feature:** Tutti i file CSV sono stati analizzati per verificare la presenza di caratteristiche comuni e specifiche per ciascun file. Sono state identificate 15 feature comuni tra tutti i file, tra cui `Time`, `mac_s`, `mac_d`, `sport`, e `dport`.
- **Unione dei file CSV:** I dati provenienti da diversi file sono stati concatenati, ottenendo un dataset finale contenente 549,737 righe.

- **Gestione dei dati mancanti:** Sono stati individuati valori mancanti significativi in feature come `modbus_response` e `modbus_fn`. I valori mancanti sono stati riempiti utilizzando la mediana per i dati numerici e un valore predefinito per quelli categoriali.
- **Rimozione di campi ridondanti:** Feature come `label`, `modbus_response` e `modbus_fn` sono state eliminate, poiché ridondanti o non rilevanti per il task.
- **Normalizzazione delle feature numeriche:** Variabili come `sport`, `dport`, `size`, `n_pkt_src`, e `n_pkt_dst` sono state normalizzate utilizzando il metodo `MinMaxScaler`.
- **Codifica delle feature categoriali:** Variabili come `proto`, `flags`, `mac_s`, e `mac_d` sono state codificate tramite `LabelEncoder`.
- **Shift del target:** La colonna `label_n` è stata shiftata di due timestamp per implementare la previsione di anomalie future.

Il dataset preprocessato è stato quindi suddiviso in training set (70%), validation set (15%) e test set (15%), mantenendo una distribuzione bilanciata delle classi.

Architettura del Modello

Il modello utilizzato per la previsione delle anomalie è una rete *Dense Neural Network (DNN)* con la seguente architettura:

- **Input Layer:** Riceve un vettore con 13 feature normalizzate.
- **Hidden Layers:** Due livelli densi con 64 e 32 nodi rispettivamente, funzioni di attivazione ReLU e dropout al 30% per ridurre l'overfitting.
- **Output Layer:** Un livello sigmoide che restituisce una probabilità binaria.

Il modello è stato compilato utilizzando:

- **Loss Function:** `binary_crossentropy`.
- **Optimizer:** `Adam` con un learning rate pari a 0.001.
- **Metriche di valutazione:** `accuracy`.

Addestramento e Valutazione

L'addestramento del modello è stato effettuato utilizzando `EarlyStopping` con un massimo di 5 epoche senza miglioramenti nella loss di validazione. L'addestramento ha raggiunto una convergenza dopo 5 epoche, come illustrato nei grafici della Figura 2.2.

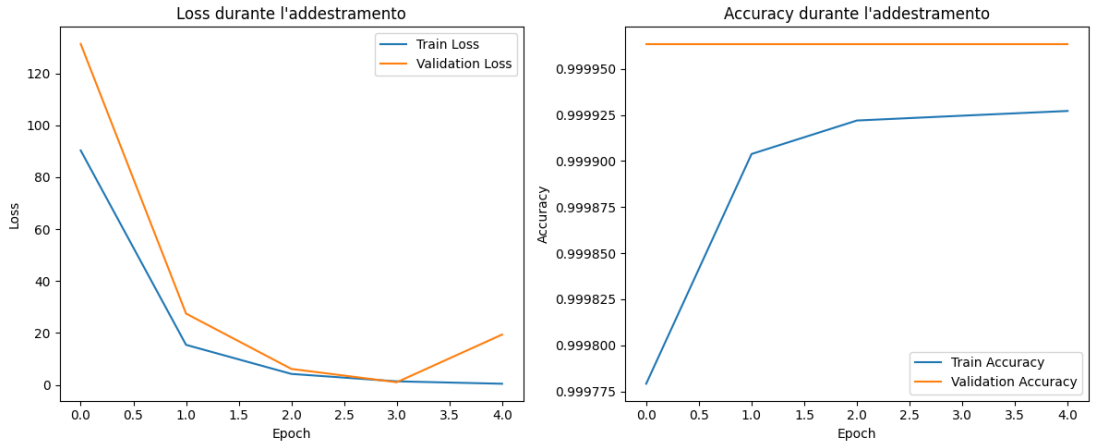

Figura 2.2: Andamento della loss e dell'accuracy durante l'addestramento del modello DNN sul dataset di rete.

Risultati

I risultati principali del modello sul test set sono riassunti di seguito:

- **Confusion Matrix:**

$$\begin{bmatrix} 82458 & 0 \\ 3 & 0 \end{bmatrix}$$

- **Classification Report:**

Classe	Precision	Recall	F1-score
Normale (0)	1.00	1.00	1.00
Anomalia (1)	0.00	0.00	0.00

Tabella 2.2: Classification Report per le classi Normale (0) e Anomalia (1).

- **ROC-AUC Score:** 0.5000.

Discussione

Nonostante l'elevata accuratezza complessiva del modello (100%), i risultati evidenziano una totale incapacità di rilevare le anomalie, con precisione e recall pari a 0.00

per la classe `1`. Questa discrepanza è attribuibile all'elevato squilibrio tra classi, dove solo 3 esempi nel test set appartengono alla classe `1`.

Criticità identificate:

- **Squilibrio del dataset:** La classe anomala rappresenta una frazione trascurabile dei dati totali.
- **Modello non bilanciato:** L'addestramento su un dataset altamente sbilanciato porta il modello a ignorare le anomalie.

Proposte per miglioramenti futuri:

- **Bilanciamento del dataset:** L'uso di tecniche come il *oversampling* (SMOTE) o il bilanciamento durante il training.
- **Data Augmentation:** Generazione sintetica di esempi anomali per aumentare la rappresentatività della classe `1`.
- **Architetture avanzate:** Modelli che integrano caratteristiche di serie temporali, come RNN o LSTM.

Capitolo 3

Anomaly Detection e Prediction per Time Series

Questo documento presenta l'implementazione dei task di Anomaly Detection e Anomaly Prediction su serie temporali.
Una Time Series (o serie temporale) è una collezione di osservazioni (o misurazioni) effettuate e registrate in ordine cronologico, spesso a intervalli di tempo regolari. In altre parole, è un insieme di dati che descrivono l'evoluzione di una o più variabili nel corso del tempo (ad esempio, la temperatura di un serbatoio, il livello di riempimento, oppure il volume di traffico di rete in un determinato intervallo).
Le serie temporali analizzate derivano da dati acquisiti durante un processo fisico (riempimento e svuotamento di serbatoi, definito come Dataset Fisico) e da dati di rete (definito come Dataset di Network), che si riferiscono allo scambio di informazioni associato al processo.

3.1 Schema del processo

Per implementare i task di Anomaly Detection e Anomaly Prediction, il processo è stato suddiviso in diverse fasi:

1. **Preprocessing**: Questa fase include il preprocessing del *Dataset Fisico* e del *Dataset di Network*. Ogni dataset viene trattato separatamente per preparare i dati grezzi alla fase successiva.

    - **Preprocessing del Dataset di Network**: Nella fase di preprocessing del dataset di rete, vengono effettuate operazioni volte a preparare i dati grezzi per l'analisi. Queste includono:
        - La gestione di informazioni temporali

- Il filtraggio di eventuali valori mancanti o incoerenti
- Selezione delle caratteristiche più rilevanti per il task: in particolare, vengono utilizzate tutte le feature ad accezione di "label", in quanto ridondante essendoci già "label_n", presente in formato numerico.
- Aggregazione e campionamento dei dati su intervalli di tempo regolari: in particolare, vengono aggregati i dati relativi ad ogni feature considerando un intervallo temporale di un secondo. Per aggregare i dati, per ogni feature vengono calcolate metriche come il numero di indirizzi unici, protocolli, porte e dimensioni dei pacchetti, così da creare una rappresentazione sintetica e strutturata del traffico di rete.
  Questa fase include quindi un processo di **Feature Extraction**, in seguito alla quale si ottengono le seguenti nuove feature:
  * **unique_mac_src_count** e **mac_src_count**: Calcolate dalla feature iniziale *mac_s* come numero di indirizzi unici (*nunique*) e conteggio totale (*count*).
  * **unique_mac_dst_count** e **mac_dst_count**: Derivate da *mac_d* come numero di indirizzi unici (*nunique*) e conteggio totale (*count*).
  * **unique_ip_src_count** e **unique_ip_dst_count**: Derivate rispettivamente da *ip_s* e *ip_d* come numero di indirizzi IP unici (*nunique*).
  * **unique_sport_count** e **unique_dport_count**: Ottenute da *sport* e *dport* come numero di porte uniche (*nunique*).
  * **unique_proto_count**: Calcolata da *proto* come numero di protocolli unici (*nunique*).
  * **unique_flags_count**: Derivata da *flags* come numero di flags unici (*nunique*).
  * **size_mean**, **size_sum** e **size_std**: Calcolate da *size* come media (*mean*), somma (*sum*) e deviazione standard (*std*).
  * **modbus_request_count**: Derivata da *modbus_fn* come conteggio totale (*count*).
  * **total_packets_sent** e **total_packets_received**: Ottenute da *n_pkt_src* e *n_pkt_dst* come somma totale (*sum*).
  * **modbus_response_count**: Calcolata da *modbus_response* come conteggio totale (*count*).
  * **label_n**: Ottenuta dalla stessa feature iniziale *label_n* come valore massimo nella finestra temporale (*max*).

Tabella 3.1: Nuove feature ottenute dalla fase di Feature Extraction e relativi metodi di calcolo.

Feature	**Feature Iniziale**	**Metodo di Calcolo**
unique_mac_src_count	mac_s	Numero di indirizzi unici (*nunique*)
mac_src_count	mac_s	Conteggio totale (*count*)
unique_mac_dst_count	mac_d	Numero di indirizzi unici (*nunique*)
mac_dst_count	mac_d	Conteggio totale (*count*)
unique_ip_src_count	ip_s	Numero di indirizzi unici (*nunique*)
unique_ip_dst_count	ip_d	Numero di indirizzi unici (*nunique*)
unique_sport_count	sport	Numero di porte uniche (*nunique*)
unique_dport_count	dport	Numero di porte uniche (*nunique*)
unique_proto_count	proto	Numero di protocolli unici (*nunique*)
unique_flags_count	flags	Numero di flags unici (*nunique*)
size_mean	size	Media (*mean*)
size_sum	size	Somma (*sum*)
size_std	size	Deviazione standard (*std*)
modbus_request_count	modbus_fn	Conteggio totale (*count*)
total_packets_sent	n_pkt_src	Somma totale (*sum*)
total_packets_received	n_pkt_dst	Somma totale (*sum*)
modbus_response_count	modbus_response	Conteggio totale (*count*)
label_n	label_n	Valore massimo nella finestra temporale (*max*)

    - **Preprocessing del Dataset Fisico**: Nella fase di preprocessing del dataset fisico viene verificata la presenza della colonna temporale e questa viene convertita in un formato datetime. Inoltre, viene rimossa una colonna superflua (Label).

2. **Unione dei Dataset Preprocessati**: I due dataset vengono combinati in un unico dataset unificato. Successivamente, vengono applicate operazioni di standardizzazione per uniformare le scale delle variabili e ulteriore preprocessing per garantire la consistenza dei dati.

3. **Divisione in Training e Test**: Il dataset unificato viene suddiviso in due sottoinsiemi: uno per l'addestramento del modello (*training set*) e uno per la valuta-

zione delle prestazioni (*test set*). Poiché si tratta di task di anomaly detection e prediction, i modelli vengono addestrati esclusivamente con i dati etichettati come normali e testati su sequenze contenenti anche dati anomali. La suddivisione viene quindi eseguita seguendo questa logica: i dataset normali vengono combinati in un unico file di training, mentre i dataset contenenti dati anomali vengono unificati in un unico file di test, mantenendo la coerenza temporale dei dati.

4. **Feature Selection**: Viene effettuata una selezione delle feature più rilevanti per ridurre la dimensionalità del dataset e migliorare l'efficienza e l'accuratezza del modello.
   Per fare questo, vengono eseguite le seguenti operazioni:

   - **Analisi della correlazione**: Calcola la matrice di correlazione tra le feature numeriche e individua coppie di feature altamente correlate (correlazione superiore a una soglia predefinita).
   - **Importanza delle feature**: Valuta l'importanza delle feature basandosi sulla loro varianza e visualizza le feature con maggiore variabilità sia prima che dopo la rimozione di feature ridondanti.
   - **Rimozione delle feature correlate**: Elimina una delle feature in ciascuna coppia altamente correlata o non necessaria.
   - **Preparazione dei dataset**: Aggiorna i dataset di train e test rimuovendo le feature selezionate e conserva un elenco ordinato delle feature rimaste per l'addestramento e la valutazione dei modelli.
   - **Visualizzazione**: Genera grafici per rappresentare la matrice di correlazione e l'importanza delle feature rimanenti, fornendo una panoramica visiva dei dati.

5. **Data Cleaning e Normalizzazione**: Questa fase si occupa di preparare i dataset per l'addestramento e la valutazione del modello.
   Inizialmente, vengono identificate e rimosse eventuali colonne di tipo temporale (datetime) dai dataset di train e test. Successivamente, le feature vengono selezionate escludendo la colonna del target (*label*) e normalizzate utilizzando il metodo *MinMaxScaler*, scalando i valori su un intervallo standard. Infine, i dataset vengono separati in input (feature normalizzate) e target (*label*), pronti per le fasi di addestramento e valutazione.

6. **Creazione delle Sequenze**: Vengono create sequenze di dati di lunghezza fissa a partire dai dataset di input e dai rispettivi target. Per il training, vengono con-

siderate solo le istanze con etichetta normale (‘label = 0‘), generando sequenze di lunghezza specificata in cui tutte le etichette all’interno della sequenza sono uguali. Per il test, vengono create sequenze includendo anche i dati con anomalie. Le sequenze e le etichette risultanti vengono restituite come array, pronte per essere utilizzate nei modelli di apprendimento basati su serie temporali..

7. **Scelta del Modello e Addestramento**: Si seleziona il modello più adatto per il task e si procede all’addestramento utilizzando il *training set*.

8. **Valutazione dei Risultati**: Infine, il modello addestrato viene valutato sul *test set* per misurare le sue prestazioni nel rilevare e prevedere anomalie.

Lo schema a blocchi, riportato nella Figura 3.2, fornisce una rappresentazione grafica intuitiva di questo processo. Ogni blocco corrisponde a una fase del workflow descritto sopra.

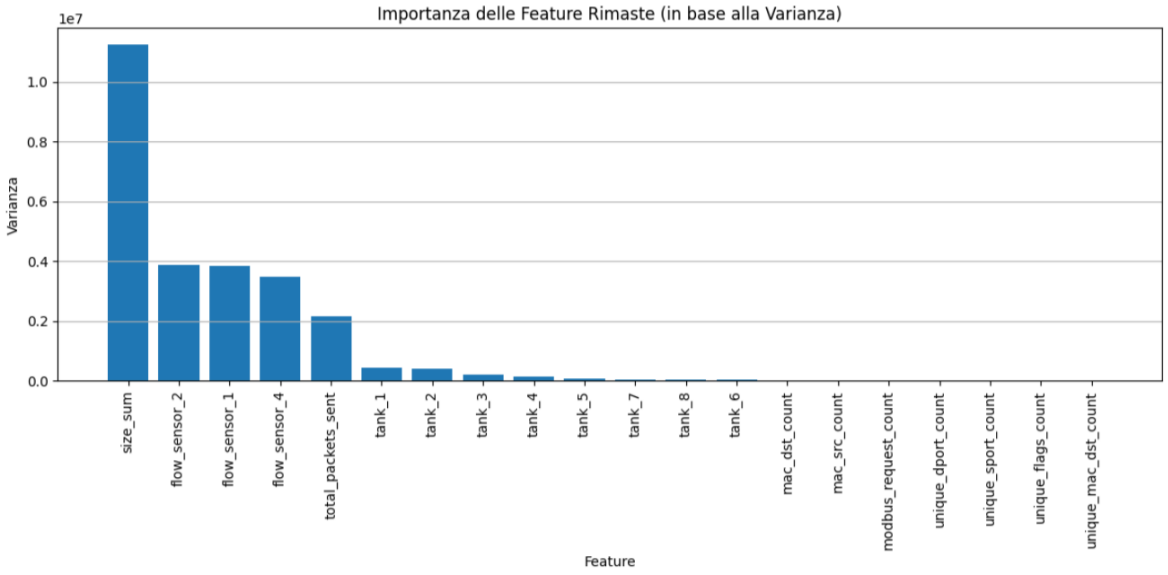

Figura 3.1: Varianza delle feature

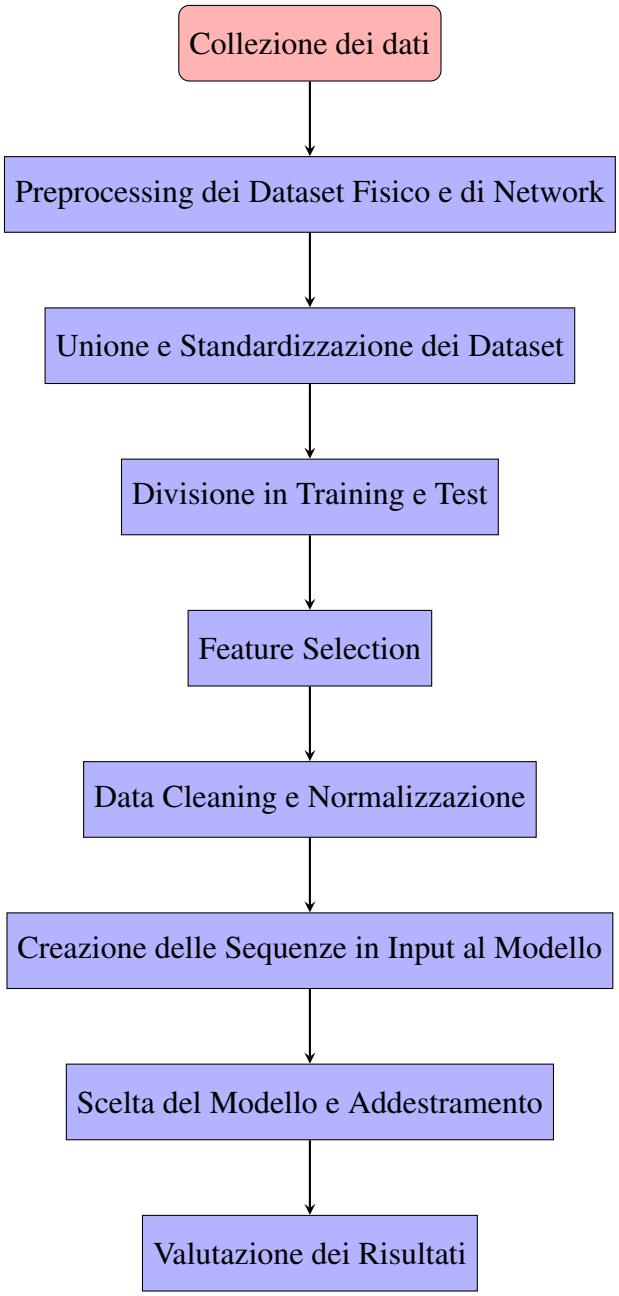

Figura 3.2: Schema a blocchi del processo di Anomaly Detection e Prediction per serie temporali.

3.2 Anomaly Detection

Per effettuare il task di Anomaly Detection per Time Series utilizziamo un **VAE (Variational Autoencoder)**: è un modello generativo che apprende rappresentazioni probabilistiche dei dati in uno spazio latente, combinando un encoder (che mappa i dati nello spazio latente) e un decoder (che ricostruisce i dati dagli embedding latenti).
In Anomaly Detection per Time Series, il VAE viene addestrato su sequenze normali,

imparando a ricostruire fedelmente questi dati. Durante il test, le anomalie generano errori di ricostruzione elevati poiché i dati anomali non seguono il modello appreso.

3.2.1 Costruzione del modello VAE

Nel processo descritto, viene costruito un modello di Variational Autoencoder (VAE) progettato per rilevare anomalie nei dati.
Il VAE è composto da un **Encoder** che comprime i dati di input in una rappresentazione latente di dimensioni ridotte, e un **Decoder** che cerca di ricostruire i dati originali a partire da questa rappresentazione.

Layer (type)	Output Shape	Param #
encoder (Functional)	[(None, 5), (None, 5)]	6,842
decoder (Functional)	(None, 160, 51)	6,915

Total params: 13,757 (53.74 KB)
Trainable params: 13,741 (53.68 KB)
Non-trainable params: 16 (64.00 B)

Figura 3.3: Modello del VAE

3.2.2 Addestramento del modello VAE

Durante la fase di addestramento, il modello viene allenato su dati normali con l'obiettivo di minimizzare la differenza tra i dati originali e quelli ricostruiti. Per migliorare la robustezza, ai dati di input viene aggiunto un rumore controllato.
La funzione di perdita del VAE combina due componenti principali: una perdita di ricostruzione, che misura la qualità della ricostruzione, e una perdita KL, che regolarizza la distribuzione latente.
Una volta addestrato, il VAE viene valutato su un dataset contenente sia dati normali che anomali. Il modello ricostruisce i dati e si calcola l'errore di ricostruzione per ogni sequenza. Gli errori vengono analizzati per identificare anomalie, con una soglia ottimale determinata sulla base delle curve di precision-recall. La soglia consente di classificare ogni sequenza come normale o anomala.

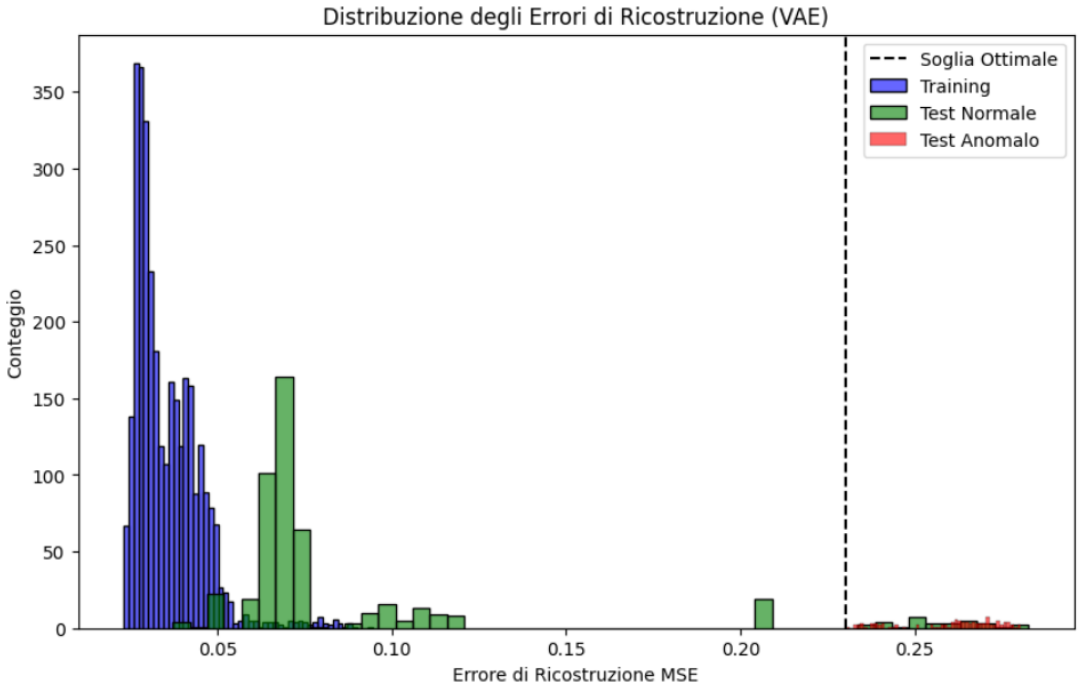

Figura 3.4: Distribuzione degli Errori di Ricostruzione del VAE

3.2.3 Prestazioni del modello

Le prestazioni del modello vengono misurate utilizzando metriche standard come la curva ROC, l'AUC score e un report di classificazione, fornendo un'analisi quantitativa dell'efficacia del VAE nel rilevare anomalie. In particolare, dal report di classificazione emergono alti valori della **Sensibilità (Recall)** e della **Precisione** della classe Anomala (rispettivamente 0.99 e 0.77), evidenziando che il modello costruito riesce a rilevare correttamente quasi tutte le anomalie presenti del dataset. Le buone prestazioni offerte dal modello risultando evidenti anche dall'alto valore dell'AUC (Area Under the Curve) del modello e dalla visualizzazione della curva ROC.

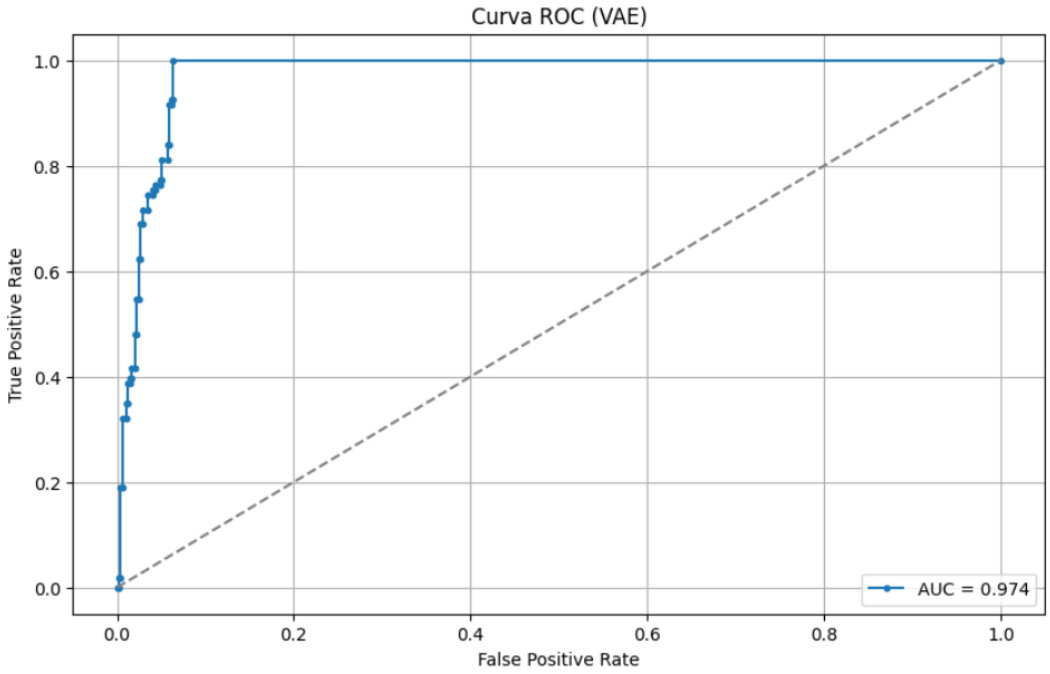

Figura 3.5: Curva ROC del VAE

3.3 Anomaly Prediction

Per effettuare il task di Anomaly Prediction per Time Series utilizziamo un modello che sfrutta LSTM (Long-Short Term Memory) e GRU (Gated Recurrent Units).
Le LSTM e le GRU sono due tipi di reti neurali ricorrenti (RNN) progettate per gestire dati sequenziali e temporali. Entrambe sono in grado di catturare relazioni a lungo termine nelle sequenze grazie a meccanismi interni che controllano il flusso di informazioni, ma differiscono nella loro complessità e struttura.
Le LSTM utilizzano celle di memoria con tre porte principali: una porta di input, una di output e una porta di dimenticanza. Queste porte regolano quali informazioni devono essere mantenute, aggiornate o dimenticate. Questo consente alle LSTM di apprendere relazioni a lungo termine senza che il segnale si degradi, problema noto come vanishing gradient. Per l'anomaly prediction, la capacità di una LSTM di ricordare eventi passati cruciali nel contesto di una sequenza è particolarmente utile per rilevare anomalie che potrebbero derivare da dipendenze temporali complesse.
Le GRU, invece, sono una versione semplificata delle LSTM. Utilizzano due porte principali: una porta di aggiornamento e una porta di reset. Questa struttura le rende meno complesse dal punto di vista computazionale e spesso più efficienti, pur mantenendo una capacità simile nel catturare dipendenze a lungo termine. La semplicità delle GRU le rende adatte quando i dati non richiedono una gestione molto detta-

gliata della memoria a lungo termine, ma beneficiano comunque di un'elaborazione temporale robusta.

3.3.1 Costruzione del modello LSTM e GRU

Nel task considerato, viene utilizzato un modello che integra LSTM e GRU in un'unica architettura, il che rende possibile sfruttare i punti di forza di entrambe. Le GRU, posizionate in uno strato bidirezionale, aiutano a catturare pattern temporali complessi guardando sia al passato che al futuro della sequenza, mentre le LSTM unidirezionali concentrano l'elaborazione su relazioni a lungo termine in una sola direzione. Questo approccio consente di bilanciare la capacità del modello di rappresentare sequenze complesse con l'efficienza computazionale, garantendo che le anomalie derivanti da dipendenze temporali siano rilevate con maggiore precisione.
La struttura del modello integra diversi livelli per un'elaborazione avanzata dei dati temporali. Inizia con un livello convoluzionale che individua pattern locali nella sequenza. I dati elaborati passano quindi attraverso un livello GRU bidirezionale, che cattura relazioni temporali in entrambe le direzioni, passate e future.
Successivamente, un meccanismo di attenzione analizza l'intera sequenza per determinare le parti più rilevanti, migliorando la capacità del modello di concentrarsi su eventi critici.
Infine, un livello LSTM unidirezionale riduce ulteriormente la sequenza a una rappresentazione compatta e ricca di informazioni, che viene utilizzata per la classificazione finale. Questa combinazione di livelli consente al modello di elaborare informazioni temporali in modo profondo ed efficiente, rendendolo particolarmente adatto a compiti di anomaly prediction.

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 170, 36)	0	-
conv1d (Conv1D)	(None, 170, 4)	436	input_layer[0][0]
batch_normalization (BatchNormalization)	(None, 170, 4)	16	conv1d[0][0]
bidirectional (Bidirectional)	(None, 170, 32)	2,112	batch_normalization[0…
dropout (Dropout)	(None, 170, 32)	0	bidirectional[0][0]
lstm (LSTM)	(None, 8)	1,312	dropout[0][0]
attention_layer (AttentionLayer)	(None, 32)	202	dropout[0][0]
dropout_1 (Dropout)	(None, 8)	0	lstm[0][0]
concatenate (Concatenate)	(None, 40)	0	attention_layer[0][0], dropout_1[0][0]
dense (Dense)	(None, 1)	41	concatenate[0][0]

Figura 3.6: Architettura modello LSTM e GRU

3.3.2 Addestramento del modello LSTM e GRU

L'addestramento e la predizione delle anomalie con un modello basato su LSTM, GRU e meccanismi di attenzione seguono un approccio che si fonda sull'analisi degli errori di ricostruzione e sull'uso di una soglia per identificare comportamenti anomali.
Durante l'addestramento, il modello apprende a prevedere una sequenza futura o a ricostruire i dati di input sequenziali. Per questo scopo, il modello viene addestrato esclusivamente su dati considerati "normali", cioè privi di anomalie. Il modello impara quindi una rappresentazione del comportamento normale e, grazie ai meccanismi di memoria delle LSTM e GRU, può catturare relazioni temporali complesse a breve e lungo termine.
Una volta addestrato, il modello viene applicato a dati di test che possono includere sia comportamenti normali che anomalie. La procedura segue questi passaggi:

- **Predizione o Ricostruzione:** Per ogni sequenza di input $\mathbf{X} \in \mathbb{R}^{T \times F}$, dove T è la lunghezza della finestra temporale e F è il numero di feature, il modello genera una predizione $\hat{\mathbf{X}} \in \mathbb{R}^{T \times F}$ che rappresenta la ricostruzione della sequenza.

- **Calcolo dell'Errore di Ricostruzione:** L'errore di ricostruzione viene calcolato per ogni sequenza utilizzando il *Mean Squared Error* (MSE):

$$\text{Errore di Ricostruzione} = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{X}_i - \hat{\mathbf{X}}_i \right)^2, \tag{3.1}$$

dove $\mathbf{X}_i$ rappresenta il valore reale e $\hat{\mathbf{X}}_i$ il valore ricostruito dal modello per il i-esimo elemento della sequenza.

- **Definizione della Soglia:** Per distinguere tra sequenze normali e anomale, si definisce una soglia τ sull'errore di ricostruzione. Due approcci comuni per determinare τ sono:
  - *Percentile dell'Errore di Ricostruzione:* Si calcola il percentile (ad esempio, il 95-esimo percentile) degli errori di ricostruzione sui dati normali:

$$\tau = \text{Percentile}_{95}\left(\text{Errore di Ricostruzione}_{\text{normale}}\right). \tag{3.2}$$

  - *Curva Precision-Recall:* Utilizzando le etichette binarie nei dati di test, si calcola la curva *Precision-Recall* variando la soglia τ. La soglia ottimale è quella che massimizza il punteggio $F1$:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.3}$$

- **Classificazione delle Anomalie:** Le sequenze vengono classificate come normali o anomale in base al confronto tra l'errore di ricostruzione e la soglia τ:
  - Se Errore di Ricostruzione $\leq \tau$, la sequenza è considerata **normale**.
  - Se Errore di Ricostruzione $> \tau$, la sequenza è considerata **anomala**.

Il motivo per cui si utilizza questo approccio è che il modello, addestrato esclusivamente su dati normali fatica a ricostruire sequenze che non seguono il comportamento atteso, come le anomalie. Questo porta a un errore di ricostruzione significativamente maggiore per i dati anomali rispetto ai dati normali. L'uso della soglia τ consente di sfruttare questa differenza per separare automaticamente i due tipi di comportamento, e quindi dati anomali da quelli normali.

3.3.3 Prestazioni del modello

In modo simile al task di Anomaly Detection, le prestazioni del modello vengono misurate utilizzando metriche standard come il report di classificazione.
Nel caso della Anomaly Prediction, le prestazioni del modello sono state valutate facendo riferimento alla soglia ottimale, ovvero quella che massimizza l'F1-score.
In questo caso, si osserva che la **Sensibilità (Recall)** per la classe anomala rimane alta

(0.87), indicando che il modello è in grado di rilevare quasi tutte le anomalie presenti. Tuttavia, la **Precisione** è significativamente più bassa rispetto al task di Anomaly Detection (0.30), suggerendo una maggiore quantità di falsi positivi. Questo comportamento implica che il modello spesso identifica sequenze come anomale anche quando non lo sono, il che potrebbe essere accettabile in contesti dove i falsi positivi sono tollerabili, come sistemi di monitoraggio preventivo o di manutenzione predittiva, in cui rilevare tutte le possibili anomalie è prioritario. D'altro canto, in contesti critici, dove l'accuratezza assoluta è essenziale e i falsi positivi rappresentano un costo o un rischio significativo, l'uso di questo modello potrebbe non essere raccomandabile.