

Detecting Trajectory Spoofing Attacks on AIS

Cybersecurity M



Studenti:

Andrea La Rocca
Cristina Tomaciello

Docente:

Prof. Michele Colojanni

Anno Accademico 2025/2026

Abstract

Il **Sistema di Identificazione Automatica (AIS)** rappresenta lo standard globale per il tracciamento marittimo e la prevenzione delle collisioni. Tuttavia, la mancanza di autenticazione e cifratura nel protocollo di trasmissione lo rende intrinsecamente vulnerabile ad **attacchi cyber-fisici di spoofing**.

Questo progetto affronta la problematica della sicurezza marittima proponendo un sistema di **Anomaly Detection** basato su **Deep Learning**, finalizzato al riconoscimento di attacchi di *trajectory spoofing* che violano la coerenza cinematica della navigazione.

La ricerca confronta due distinte architetture neurali di tipo Autoencoder: le classiche **Long Short-Term Memory (LSTM)**, che operano a tempo discreto, e le innovative **Liquid Neural Networks (LNN)**, basate su equazioni differenziali ordinarie per la modellazione a tempo continuo. Per la validazione sperimentale è stata sviluppata una pipeline di *data curation* su un dataset reale della *U.S. Maritime Administration* composto da oltre 215 milioni di messaggi, arricchito tramite l'iniezione sintetica di quattro categorie di minacce: **Speed Spoofing**, **Teleport**, **Ghost Ship** e **Silent Drift**.

I risultati sperimentali dimostrano che **entrambe le architetture raggiungono tassi di rilevamento del 100%** per le anomalie cinematiche e progressive. L'analisi critica evidenzia che la presunta superiorità delle LNN nello scenario **Ghost Ship** non deriva da una migliore capacità di apprendimento, ma dalla calibrazione della soglia operativa di allarme, identificando nel trade-off tra sensibilità e falsi positivi il vero fattore determinante per la sicurezza. Lo studio conclude che, sebbene i modelli apprendano efficacemente le leggi fisiche del moto, soffrono di una "cecità spaziale" rispetto alla geografia assoluta, suggerendo la necessità futura di sistemi ibridi contestuali.

Indice

1 Il Sistema di Identificazione Automatica (AIS) e le Sue Vulnerabilità	3
1.1 Introduzione e Funzionamento dell'AIS	3
1.2 Panoramica delle Minacce alla Sicurezza Marittima	4
1.3 Classificazione degli Attacchi di Spoofing di Traiettoria	4
2 Deep Learning per Serie Temporali (LSTM e LNN)	5
2.1 Introduzione al Rilevamento di Anomalie (Anomaly Detection)	5
2.2 Modellazione di Sequenze con Reti Ricorrenti (RNN)	5
2.3 L'Architettura Autoencoder per il Rilevamento di Anomalie	6
2.3.1 Meccanismo di Rilevamento tramite MAE	6
2.4 L'AutoEncoder LSTM (Long Short-Term Memory) - Approccio a Tempo Discreto	7
2.4.1 Architettura e Meccanismo delle Celle	7
2.5 L'AutoEncoder LNN (Liquid Neural Network)-Approccio a Tempo Continuo .	8
2.5.1 Celle CfC (Closed-form Continuous-time)	8
2.5.2 Architettura AutoNCP e Wiring Sparso	8
3 Dataset e Tassonomia degli Attacchi	9
3.1 Acquisizione e Caratterizzazione dei Dati	9
3.2 Pipeline di Preprocessing e Feature Engineering	9
3.2.1 Data Cleaning e Filtraggio	10
3.2.2 Segmentazione e Regolarizzazione Temporale	10
3.2.3 Sliding Window e Normalizzazione	11
3.3 Tassonomia e Modellazione degli Attacchi	11
3.3.1 Speed Spoofing (Falsificazione della Velocità)	12
3.3.2 Teleport (Incoerenza Cinematica)	12
3.3.3 Ghost Ship (Incoerenza di Rotta)	13
3.3.4 Silent Drift (Deriva Progressiva)	13
4 Implementazione dei Modelli e Risultati Sperimentali	14
4.1 Metodologia di Addestramento	14
4.2 Configurazione delle Architetture	14
4.2.1 LSTM Autoencoder	14
4.2.2 LSTM AutoEncoder: Analisi della Ricostruzione	15
4.2.3 Liquid Neural Network (LNN)	16
4.2.4 LNN AutoEncoder: Analisi della Ricostruzione	17
4.2.5 Confronto Preliminare	18
4.3 Risultati di Rilevamento	19
4.3.1 Anomalie Cinematiche (Speed Spoofing e Teleport)	19
4.3.2 Anomalie Progressive (Silent Drift)	19
4.3.3 Anomalie Semantiche (Ghost Ship)	19
4.3.4 Risultati LSTM	19

4.3.5	Risultati LNN	21
5	Analisi Critica e Discussione dei Risultati	24
5.1	Dinamicità della Soglia e Trade-off Operativo	24
5.1.1	La Relatività del Rilevamento	24
5.2	Analisi Comparativa: Stabilità del Modello e Separabilità delle Classi	24
5.2.1	Varianza e Definizione del Margine di Sicurezza	24
5.3	Limiti Intrinseci: Cecità Spaziale e Apprendimento Cinematico	25
5.3.1	Analisi dell'Apprendimento Effettivo	25
6	Sviluppi Futuri	26
6.1	Risoluzione della Cecità Spaziale	26
6.2	Soglia Adattiva tramite Clustering Contestuale	26
6.3	Interpretabilità e Supporto Decisionale	26
	Conclusioni	28

Capitolo 1

Il Sistema di Identificazione Automatica (AIS) e le Sue Vulnerabilità

1.1 Introduzione e Funzionamento dell'AIS

Il Sistema di Identificazione Automatica (AIS) (*Automatic Identification System*) è un transponder e un sistema di tracciamento automatico essenziale nel settore marittimo, originariamente sviluppato come standard dall'Organizzazione Marittima Internazionale (IMO). La sua funzione primaria è duplice: aumentare la sicurezza della navigazione attraverso la prevenzione delle collisioni in mare e supportare la gestione globale del traffico marittimo. L'AIS opera trasmettendo e ricevendo in modo continuo e autonomo dati vitali via radiofrequenze VHF, fornendo in tempo reale informazioni sulla posizione, la rotta, la velocità (SOG/COG) e lo stato di navigazione di un'imbarcazione alle altre navi e alle stazioni a terra. Questo lo rende un componente cruciale non solo per la sicurezza, ma anche per la logistica marittima, il coordinamento delle operazioni di salvataggio e il monitoraggio di attività illecite, come la pesca non autorizzata.

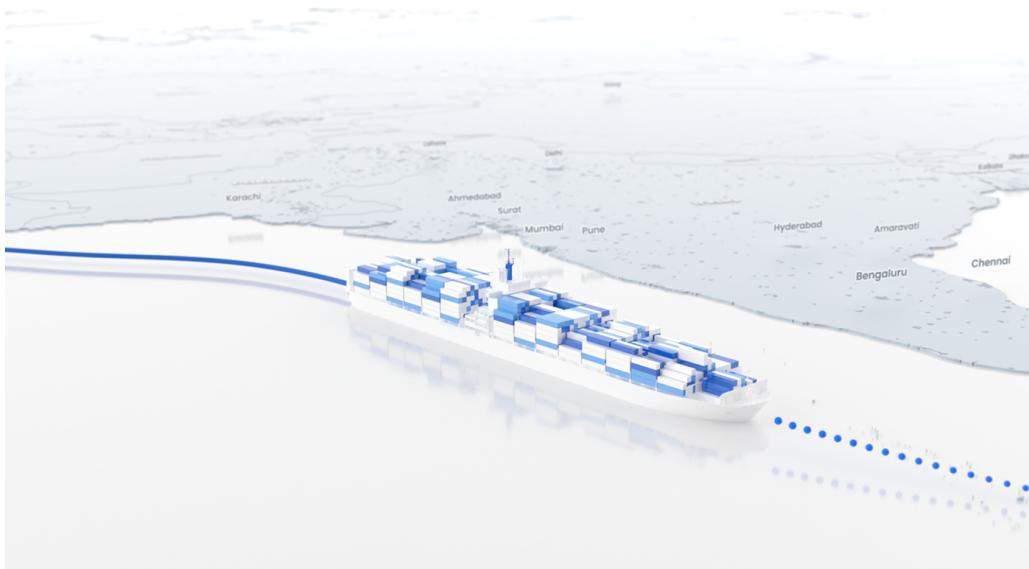


Figura 1.1: Traiettoria composta da dati AIS

1.2 Panoramica delle Minacce alla Sicurezza Marittima

Nonostante l'AIS sia fondamentale per la sicurezza e il coordinamento del traffico, la sua natura di **sistema di trasmissione non autenticata** via radiofrequenze lo rende intrinsecamente vulnerabile agli attacchi. La sicurezza marittima moderna, infatti, non si limita più alle minacce fisiche tradizionali, ma deve affrontare un crescente spettro di minacce cyber che mirano a compromettere l'affidabilità dei dati di navigazione. Queste vulnerabilità sono sfruttate per diverse finalità malevole:

- **Guerra Ibrida e Disinformazione:** L'iniezione di false informazioni di posizione o identità può servire a nascondere o simulare la presenza di navi, creando confusione (es. la creazione di "navi fantasma").
- **Attività Illegali:** Attori non statali, come trafficanti o contrabbandieri, possono manipolare i dati AIS per nascondere il loro percorso effettivo o le loro destinazioni.
- **Dirottamenti e Furti Stealth:** La disabilitazione o la manipolazione selettiva dell'AIS (spoofing) può consentire a un'imbarcazione di "scomparire" dai sistemi di tracciamento o di deviare in modo invisibile, facilitando un dirottamento o un furto di carico.
- **Attacchi Cinetici:** La falsificazione dei dati di navigazione (in particolare GPS o AIS) può indurre una nave a seguire rotte pericolose o a causare collisioni, con l'obiettivo di danneggiare infrastrutture critiche o l'ambiente.

1.3 Classificazione degli Attacchi di Spoofing di Traiettoria

Il progetto è specificamente focalizzato sul rilevamento degli attacchi che violano la **coerenza fisica e/o dinamica** della traiettoria di una nave, noti come *trajectory spoofing attacks*. Questa classe di attacchi si basa sull'iniezione di dati AIS che, sebbene plausibili nel formato, sono **impossibili** da realizzare in termini cinematici (es. accelerazioni istantanee o salti nello spazio) per un'imbarcazione reale. Ai fini della validazione del modello, la nostra metodologia si è concentrata sulla capacità di rilevare quattro distinte categorie di anomalie che rappresentano le minacce più sofisticate e insidiose:

- **Attacco Cinematico (GPS/Teleport):** L'anomalia è creata da una discontinuità istantanea della posizione (Salto GPS), violando il principio di inerzia fisica.
- **Speed Spoofing:** Consiste nell'iniezione di dati di velocità estremamente rumorosi o impossibili.
- **Ghost Ship (Rotta Inversa):** In questo scenario, la nave trasmette una velocità nominale, ma il **vettore di rotta (COG)** è falsificato (es. un errore di 180°), indicando una direzione opposta a quella effettiva.
- **Silent Drift:** L'attacco più *stealth*, dove la posizione devia lentamente e progressivamente mentre i dati di velocità sono nominali. Questo simula un trascinamento o un dirottamento lento, difficile da individuare con i metodi di rilevamento tradizionali.

Capitolo 2

Deep Learning per Serie Temporali (LSTM e LNN)

2.1 Introduzione al Rilevamento di Anomalie (Anomaly Detection)

Il **Rilevamento di Anomalie** (*Anomaly Detection*) è un campo critico della statistica e del Machine Learning focalizzato sull'identificazione di pattern o eventi che non sono conformi al comportamento atteso o "normale" di un set di dati. In un contesto di **serie temporali** come i dati di tracciamento navale AIS, un'anomalia è definita come **una deviazione significativa e inattesa** dalla sequenza o dalla dinamica di movimento precedentemente appresa. L'obiettivo primario non è semplicemente classificare un punto, ma comprendere se l'evoluzione temporale di una sequenza di dati (posizione, velocità, rotta) viola la **coerenza intrinseca o la firma fisica** del sistema che la genera. Nel nostro scenario di *trajectory spoofing*, l'anomalia corrisponde a una falsificazione intenzionale che rompe le leggi cinematiche di un'imbarcazione, distinguendosi dal rumore naturale o dagli errori di misurazione.

2.2 Modellazione di Sequenze con Reti Ricorrenti (RNN)

La peculiarità dei dati AIS (posizione, velocità, rotta) risiede nella loro natura di **serie temporale sequenziale**, dove ogni osservazione è intrinsecamente dipendente dalle osservazioni precedenti. Per modellare efficacemente la dinamica di un veicolo marittimo, è essenziale utilizzare architetture di Machine Learning in grado di catturare le dipendenze a lungo termine e di apprendere il "contesto" del movimento in corso. Le **Reti Neurali Ricorrenti (RNN)** sono specificamente progettate per gestire input sequenziali. A differenza delle reti neurali feed-forward standard, le RNN mantengono uno **stato interno (memoria)** che viene aggiornato ad ogni passo temporale. Questo meccanismo di *ricorrenza* permette alla rete di collegare informazioni del passato con l'output corrente, rendendole ideali per:

1. **Mantenere lo Stato:** Ricordare la posizione e la rotta di una nave su un lungo intervallo di tempo, essenziale per capire se una manovra attuale è coerente con la sua storia.
2. **Modellare la Dipendenza Temporale:** Imparare le transizioni legali tra uno stato cinematico (es. velocità e accelerazione) e quello successivo, incapsulando di fatto le **leggi fisiche e l'inerzia del veicolo**.

Tuttavia, le RNN "tradizionali" soffrono del problema del vanishing gradient, che limita la loro capacità di apprendere dipendenze molto lunghe. Per superare questa limitazione, sono

state sviluppate architetture più avanzate, come la **Long Short-Term Memory (LSTM)** e le **Liquid Neural Networks (LNN)**, che costituiscono il nucleo comparativo del nostro progetto.

2.3 L'Architettura Autoencoder per il Rilevamento di Anomalie

Per sfruttare la capacità di modellazione temporale delle Reti Ricorrenti nel contesto dell'identificazione di comportamenti anomali, si utilizza l'architettura **Autoencoder (AE)**. Un Autoencoder è una rete neurale non supervisionata progettata per apprendere una rappresentazione efficiente (*o codifica*) di un insieme di dati, riducendone la dimensionalità. L'AE è composto da due parti fondamentali:

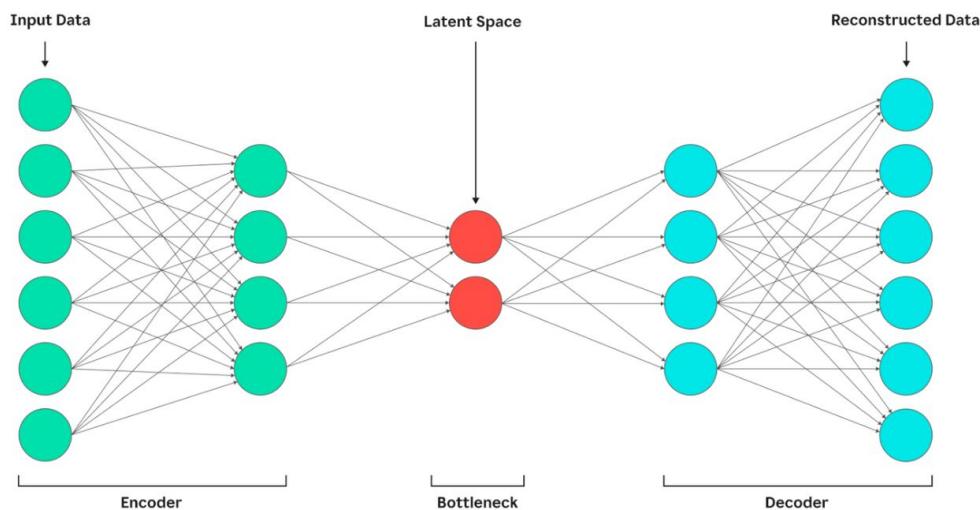


Figura 2.1: Composizione di un Autoencoder (AE)

1. **Encoder:** Accetta una sequenza di input (es. una porzione di traiettoria AIS) e la comprime in una rappresentazione a bassa dimensione, nota come spazio latente o bottleneck.
2. **Decoder:** Riceve la rappresentazione dallo spazio latente e tenta di ricostruire la sequenza di input originale.

2.3.1 Meccanismo di Rilevamento tramite MAE

Il principio fondamentale nel *Rilevamento di Anomalie* con un Autoencoder si basa sull'addestramento del modello esclusivamente su dati che rappresentano il **movimento normale e lecito**.

- **Apprendimento Normale:** Durante l'addestramento, l'AE impara a ricostruire le sequenze normali con un errore molto basso. Lo spazio latente cattura quindi la "firma" del movimento lecito.
- **Rilevamento Anomalo:** Quando viene presentato un input anomalo (ad esempio, un attacco di *Teleport o Speed Spoofing*), l'AE non è in grado di codificare e decodificare correttamente quel pattern poiché non corrisponde alla firma appresa.

La misura di questa incapacità è l'**Errore di Ricostruzione**, calcolato tramite il **Mean Absolute Error (MAE)**. Un alto MAE di ricostruzione è quindi il segnale diretto di un'anomalia.

2.4 L'AutoEncoder LSTM (Long Short-Term Memory) - Approccio a Tempo Discreto

L'AutoEncoder basato su **Long Short-Term Memory (LSTM)** rappresenta una delle soluzioni più robuste e consolidate per la modellazione di serie temporali complesse, caratteristica essenziale per le traiettorie navali, fungendo quindi da approccio di **riferimento a tempo discreto** nel nostro progetto.

2.4.1 Architettura e Meccanismo delle Celle

Il potere della LSTM risiede nella sua **cella di memoria (cell state)**, il cui flusso di informazioni è regolato da tre *gate* interni:

1. **Forget Gate:** Determina quali informazioni della cella di memoria precedente devono essere dimenticate.
2. **Input Gate:** Decide quali nuove informazioni dell'input corrente devono essere memorizzate nello stato della cella.
3. **Output Gate:** Controlla quale parte dello stato della cella viene esposta come output (stato nascosto) al passo temporale successivo.

Nel contesto dell'Autoencoder, la LSTM è implementata sia nell'Encoder che nel Decoder:

1. L'Encoder LSTM comprime la traiettoria di input (sequenza di posizioni, velocità) in un vettore di stato latente.
2. Il Decoder LSTM utilizza questo stato per ricostruire la traiettoria originale, passo dopo passo, gestendo le sequenze in intervalli temporali definiti.

L'efficacia della LSTM nella ricostruzione del movimento normale e la sua gestione precisa delle sequenze la rendono una scelta valida, nonostante sia basata su una logica di aggiornamento che lavora per passi discreti nel tempo.

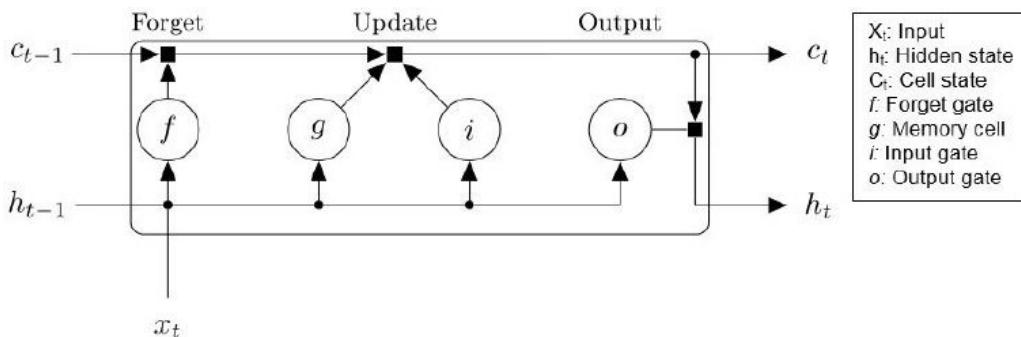


Figura 2.2: Flusso di dati alla fase temporale t per una unità LSTM

2.5 L'AutoEncoder LNN (Liquid Neural Network)-Approccio a Tempo Continuo

Le **Liquid Neural Networks (LNN)** rappresentano l'avanguardia nell'ambito della modellazione di serie temporali, introducendo un approccio a tempo continuo che contrasta con la logica a passi discreti delle LSTM. Questo approccio è particolarmente adatto a modellare sistemi fisici complessi, come la dinamica inerziale di una nave, offrendo una stabilità superiore al rumore e una maggiore robustezza nella gestione di dati irregolari o incompleti.

2.5.1 Celle CfC (Closed-form Continuous-time)

L'elemento chiave delle LNN è costituito dalle **Celle CfC** (*Closed-form Continuous-time*). A differenza delle LSTM che utilizzano gate per aggiornare lo stato di memoria in modo discreto, le celle CfC modellano la dinamica del veicolo attraverso la risoluzione di **Equazioni Differenziali Ordinarie (ODE)**. In questo modo, le LNN non si limitano a ricordare cosa è successo, ma riescono a modellare il modo in cui il sistema fisico si evolve nel tempo, anche tra i punti di campionamento discreti. Di conseguenza, questa modellazione continua conferisce al modello una migliore capacità di generalizzazione.

2.5.2 Architettura AutoNCP e Wiring Sparso

Per ottimizzare ulteriormente l'efficienza e la capacità predittiva, l'Autoencoder LNN utilizza l'architettura **AutoNCP** (*Neural Circuit Policies*). Questa configurazione si ispira alla connettività neurale biologica per creare un **wiring sparso** (connettività ottimizzata e non completamente connessa). Il wiring sparso non solo riduce il numero di parametri da addestrare, aumentando l'efficienza, ma contribuisce anche alla **robustezza complessiva** del modello, rendendo le LNN una soluzione scalabile e altamente affidabile per la protezione dei sistemi di navigazione.

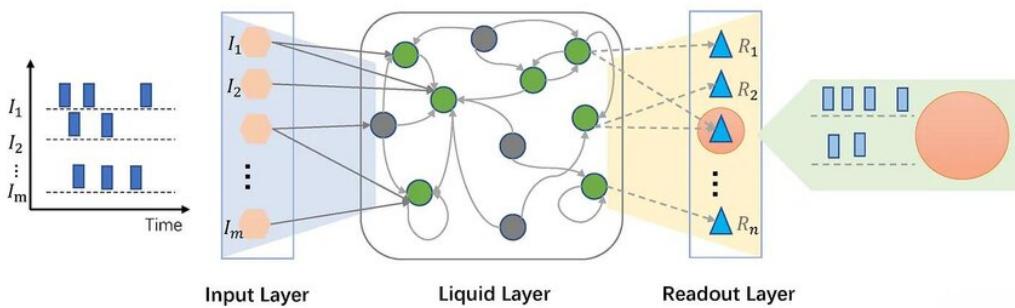


Figura 2.3: Architettura di una rete LNN

Capitolo 3

Dataset e Tassonomia degli Attacchi

L'efficacia di un sistema di rilevamento delle intrusioni dipende in modo imprescindibile dalla qualità dei dati utilizzati per l'addestramento. Poiché le reti neurali operano secondo il principio del *Garbage In, Garbage Out*, gran parte dello sforzo progettuale è stato dedicato alla costruzione di una pipeline di dati robusta. Questo capitolo descrive il ciclo di vita del dato, dall'acquisizione dei messaggi grezzi fino alla generazione sintetica di scenari di attacco complessi, necessari per validare le capacità di rilevamento del sistema.

3.1 Acquisizione e Caratterizzazione dei Dati

Per garantire l'apprendimento di dinamiche di navigazione realistiche da parte del modello, è stato selezionato il dataset pubblico della **U.S. Maritime Administration (MARAD)**. La finestra temporale scelta copre l'anno fiscale da **Luglio 2024 a Giugno 2025**.

I dati grezzi, nativamente distribuiti in file CSV giornalieri, sono stati convertiti nel formato colonnare **Apache Parquet**. Questa operazione di ingegneria dei dati è stata necessaria per gestire l'enorme mole di informazioni. Il volume totale dei dati processati ammonta a circa **215 milioni di messaggi AIS**, generati da oltre **1.2 milioni di identificativi unici (MMSI)**. Ogni messaggio cattura un'istantanea dello stato cinematico della nave, definito dalle seguenti feature essenziali:

- **MMSI (Maritime Mobile Service Identity)**: L'identificativo univoco della nave.
- **Latitudine e Longitudine**: La posizione geospaziale nel sistema di riferimento WGS84.
- **SOG (Speed Over Ground)**: La velocità scalare della nave rispetto al fondale.
- **COG (Course Over Ground)**: L'angolo di rotta della nave rispetto al nord vero.
- **Timestamp**: L'istante temporale esatto della ricezione del segnale.

3.2 Pipeline di Preprocessing e Feature Engineering

I dati AIS sono notoriamente rumorosi: soffrono di packet loss, errori dei sensori GPS a bordo e irregolarità nella trasmissione. Per trasformare questo flusso grezzo in dati papabili per le reti neurali (LSTM e LNN), è stata sviluppata una pipeline di preprocessing a tre stadi.

3.2.1 Data Cleaning e Filtraggio

La prima fase ha riguardato la pulizia del dataset per rimuovere il rumore che avrebbe potuto confondere il modello durante l'apprendimento della fisica navale. Sono state applicate regole di filtraggio rigide:

1. **Rimozione dello Stazionamento:** Dato che l'obiettivo del progetto è rilevare anomalie sulla traiettoria in movimento, sono stati scartati tutti i record con una velocità (SOG) inferiore a **2.0 nodi** perché i dati di navi ormeggiate o all'ancora rappresentano un elemento che non contribuisce all'apprendimento della dinamica inerziale.
2. **Validazione dei Sensori:** Sono stati eliminati i messaggi contenenti valori di default del protocollo AIS che indicano “dato non disponibile” (come un COG fisso a 511.0) o coordinate geografiche impossibili (fuori dai range ± 90 latitudine, ± 180 longitudine).
3. **Integrità degli Identificativi:** Sono stati rimossi gli MMSI non conformi allo standard ITU (lunghezza diversa da 9 cifre) o palesemente fintizi (es. “123456789”), spesso frutto di errori di configurazione dei transponder che non corrispondono a navi reali.

3.2.2 Segmentazione e Regolarizzazione Temporale

Una delle sfide più complesse nell'analisi dei dati AIS è che le navi non trasmettono a intervalli regolari. Infatti, in condizioni normali la frequenza varia da 2 secondi a 3 minuti a seconda della velocità e della manovra. Le reti neurali classiche, tuttavia, performano meglio con passi temporali costanti. Per risolvere questo problema, è stata implementata una logica di **Segmentazione e Resampling**:

1. **Segmentazione delle Traiettorie:** I dati sono stati raggruppati per nave (MMSI) e ordinati temporalmente. Poiché una nave può spegnere il transponder e riaccenderlo giorni dopo in un luogo diverso, è stata introdotta una soglia di taglio (**Time Gap Threshold**) di **60 minuti**. Se tra due messaggi consecutivi passa più di un'ora, la sequenza viene spezzata e considerata come una nuova traiettoria indipendente, evitando di interpolare posizioni attraverso lunghi periodi di silenzio radio.
2. **Filtraggio delle Traiettorie:** Considerando la grande mole di dati a disposizione si è scelto di optare per un filtraggio **aggressivo** andando ad eliminare tutte le traiettorie con almeno 1 gap di più di **10 minuti** tra le varie righe.
3. **Interpolazione sulla Griglia (Snap-to-Grid):** Le traiettorie valide sono state ricamionate su una griglia temporale fissa di **1 minuto**.
 - Per Latitudine, Longitudine e Velocità, è stata usata un'interpolazione lineare.
 - Per la Rotta (COG), è stata adottata un'**interpolazione vettoriale**. Poiché gli angoli sono ciclici (359° è vicino a 1°), interpolare linearmente avrebbe creato errori grossolani. Il COG è stato quindi decomposto in Seno e Coseno, interpolato nello spazio vettoriale e poi riconvertito in angolo, garantendo la continuità della rotazione.

```

# Interpolazione Coerente
final_batch = joined.with_columns([
    pl.col("Latitude").interpolate().over("TrajectoryID"),
    pl.col("Longitude").interpolate().over("TrajectoryID"),
    pl.col("SOG").interpolate().clip(0, 200).over("TrajectoryID"),
    pl.col("cog_sin").interpolate().over("TrajectoryID"),
    pl.col("cog_cos").interpolate().over("TrajectoryID"),
    pl.col("MMSI").forward_fill().backward_fill().over("TrajectoryID")
])

# Ricostruzione COG
final_batch = final_batch.with_columns([
    (np.arctan2(pl.col("cog_sin"), pl.col("cog_cos")) * 180 / np.pi).alias("COG_new")
]).with_columns([
    ((pl.col("COG_new") + 360) % 360).alias("COG")
]).rename({"GridTime": "Timestamp"})

```

Figura 3.1: Interpolazione

3.2.3 Sliding Window e Normalizzazione

L'ultimo stadio prepara l'input per la rete. Le lunghe traiettorie continue sono state suddivise in finestre scorrevoli (**Sliding Windows**) di lunghezza fissa $W = 30$. Questo significa che il modello riceve in input gli ultimi 30 minuti di navigazione per predire il comportamento futuro. Tutte le feature sono state infine normalizzate utilizzando uno **StandardScaler** (media 0, varianza 1). Questo passaggio è cruciale per evitare che variabili con scale diverse (es. la rotta in gradi 0-360 vs la velocità in nodi 0-20) destabilizzino la discesa del gradiente durante il training.

```

COLONNE_DA_NORMALIZZARE = ['Latitude', 'Longitude', 'SOG', 'COG']
SCALER_PATH = 'scaler.joblib'

scaler = StandardScaler()

try:
    for i, file_path in enumerate(TRAIN_FILES):
        print(f"Processando file {i+1}/{len(TRAIN_FILES)}: {os.path.basename(file_path)}")
        df_chunk = pd.read_parquet(file_path, columns=COLONNE_DA_NORMALIZZARE)
        scaler.partial_fit(df_chunk)
        del df_chunk
        gc.collect()
    print("\nAdattamento completato su tutti i 16 file di training.")
    joblib.dump(scaler, SCALER_PATH) #Salva file
    print(f"\nScaler salvato come '{SCALER_PATH}'.")

except Exception as e:
    print(f"\n Errore durante la preparazione dello scaler: {e}")

```

Figura 3.2: Creazione `scaler.joblib` tramite `StandardScaler`

3.3 Tassonomia e Modellazione degli Attacchi

Uno dei problemi principali nella cybersecurity marittima è la mancanza di dataset pubblici contenenti attacchi reali annotati (“Spoofing dataset”). Per ovviare a ciò e validare scientificamente il sistema, è stata sviluppata una metodologia di **Data Injection**: partendo dai dati

reali (assunti come “Normali”), sono state iniettate perturbazioni matematiche controllate per generare un *Ground Truth* sintetico contenente quattro specifiche classi di minaccia.

3.3.1 Speed Spoofing (Falsificazione della Velocità)

Questo attacco simula la manipolazione del campo velocità nel messaggio AIS, lasciando inalterata la posizione. È uno scenario tipico per nascondere lo stato operativo della nave (es. farla sembrare in transito rapido mentre sta compiendo attività illecite a bassa velocità).

- **Modellazione:** Al valore normalizzato della velocità (SOG) viene aggiunto un rumore gaussiano con media 5.0 e deviazione standard 2.0.
- **Effetto:** Si crea una discrepanza fisica immediata. La distanza percorsa calcolata dalle posizioni GPS non corrisponde più all’integrale della velocità dichiarata, violando le leggi del moto.

```
# 1. Speed Spoofing (Rumore forte su SOG)
# Aggiungiamo rumore significativo alla velocità
noise_speed = np.random.normal(5.0, 2.0, X_mixed[idx_speed, :, 2].shape)
X_mixed[idx_speed, :, 2] += noise_speed
y_true[idx_speed] = 1
attack_type[idx_speed] = 1
```

Figura 3.3: Codice Speed Spoofing

3.3.2 Teleport (Incoerenza Cinematica)

L’attacco *Teleport* rappresenta una discontinuità spaziale improvvisa, simulando uno spoofing GNSS grossolano o un errore critico del sensore.

- **Modellazione:** Viene aggiunto un offset spaziale fisso e significativo (+0.5 nello spazio normalizzato) sia alla latitudine che alla longitudine.
- **Dinamica:** L’anomalia viene iniettata solo negli **ultimi 5 minuti** della finestra temporale di 30 minuti. Questo serve a testare la reattività del modello nel rilevare un cambio di stato improvviso dopo una fase di normalità.

```
# 2. Teleport (Salto di posizione finale)
# Salto improvviso negli ultimi 5 step
X_mixed[idx_tele, -5:, 0] += 0.5
X_mixed[idx_tele, -5:, 1] += 0.5
y_true[idx_tele] = 1
attack_type[idx_tele] = 2
```

Figura 3.4: Codice Teleport

3.3.3 Ghost Ship (Incoerenza di Rotta)

Lo scenario *Ghost Ship* è una minaccia più sottile e complessa. La nave mantiene posizioni e velocità realistiche, ma trasmette una rotta (COG) falsa, indicando di muoversi in una direzione diversa (spesso opposta) a quella reale.

- **Modellazione:** Il valore del COG viene invertito di 180 gradi. Inoltre, viene aggiunto un rumore di “jitter” (tremolio).
- **Obiettivo:** Questo attacco sfida la capacità del modello di correlare la variazione delle coordinate geografiche (il vettore spostamento effettivo) con il vettore direzione dichiarato nel messaggio.

```
# 3. Ghost Ship (Inversione Rotta + Jitter)
# Inversione COG (180 gradi circa) + rumore
X_mixed[idx_ghost, :, 3] = (X_mixed[idx_ghost, :, 3] + 0.5) % 1.0
X_mixed[idx_ghost, :, 3] += np.random.normal(0, 0.2, X_mixed[idx_ghost, :, 3].shape)
y_true[idx_ghost] = 1
attack_type[idx_ghost] = 3
```

Figura 3.5: Codice Ghost Ship

3.3.4 Silent Drift (Deriva Progressiva)

Il *Silent Drift* è l’attacco più sofisticato e difficile da rilevare dato che simula un dirottamento lento o una deriva causata da correnti, dove la nave viene fatta deviare dalla rotta prevista in modo molto graduale, cercando di rimanere sotto la soglia di allarme dei sistemi di monitoraggio tradizionali.

- **Modellazione:** A differenza del *Teleport*, l’errore non è un gradino ma una **rampa lineare**. Viene applicato un coefficiente di deriva ($\alpha = 0.01$) che moltiplica il tempo trascorso.
- **Dinamica:** All’inizio della finestra l’errore è nullo; man mano che il tempo scorre, la posizione falsificata si allontana sempre di più dalla realtà. Questo scenario è cruciale per testare la “memoria a lungo termine” delle reti ricorrenti (LSTM/LNN), poiché l’anomalia è visibile solo osservando l’accumulo dell’errore nel tempo, non il singolo istante.

```
# 4. Silent Drift (Deriva progressiva)
# Drift lineare da 0 a 0.01 (più sottile per testare la sensibilità)
drift_rate = 0.01
steps = np.arange(WINDOW_SIZE)
drift_vector = steps * drift_rate
X_mixed[idx_drift, :, 0] += drift_vector
X_mixed[idx_drift, :, 1] += drift_vector
y_true[idx_drift] = 1
attack_type[idx_drift] = 4
```

Figura 3.6: Codice Silent Drift

Capitolo 4

Implementazione dei Modelli e Risultati Sperimentali

4.1 Metodologia di Addestramento

La fase di addestramento rappresenta il nucleo della pipeline di rilevamento. Poiché l’obiettivo è identificare anomalie mai viste prima, si è optato per un approccio *Self-Supervised* basato su architetture *Sequence-to-Sequence Autoencoder*. Il principio operativo è la ricostruzione: le reti neurali vengono addestrate esclusivamente su dati di traffico considerati “legittimi” (dataset MARAD). La rete impara a comprimere la sequenza di input di 30 minuti (X) in uno spazio latente (Z) e a ricostruirla in uscita (\hat{X}). L’assunzione è che il modello fallirà nel ricostruire traiettorie attaccate, generando un errore di ricostruzione (MAE) elevato.

4.2 Configurazione delle Architetture

4.2.1 LSTM Autoencoder

Il modello *Long Short-Term Memory* (LSTM) è stato implementato come riferimento standard per serie temporali discrete.

- **Encoder:** Layer LSTM (128 unità) che restituisce l’ultimo stato nascosto (vettore di contesto).
- **Decoder:** Layer `RepeatVector` seguito da LSTM (128 unità) per ricostruire la sequenza.
- **Output:** Layer `TimeDistributed Dense` per mappare le 4 feature (Lat, Lon, SOG, COG).
- **Ottimizzazione:** Loss MAE, Ottimizzatore Adam ($LR = 0.001$).

```

n_features = len(COLONNE_FEATURES)
latent_dim = 128

# Encoder
inputs = Input(shape=(WINDOW_SIZE, n_features))
# Il primo LSTM comprime l'input
lstm_encoder = LSTM(latent_dim, return_sequences=False)(inputs)

# Decoder
# Ripete il vettore compresso per ogni timestep
repeat_vector = RepeatVector(WINDOW_SIZE)(lstm_encoder)

# Il secondo LSTM "legge" il vettore compresso e ricostruisce la sequenza
lstm_decoder = LSTM(latent_dim, return_sequences=True)(repeat_vector)

# Un layer finale per rimappare l'output
output = TimeDistributed(Dense(n_features))(lstm_decoder)

model_lstm = Model(inputs, output)
model_lstm.compile(optimizer='adam', loss='mae')

print("Modello LSTM-Autoencoder creato e compilato.")
model_lstm.summary()

```

Figura 4.1: LSTM AutoEncoder

4.2.2 LSTM AutoEncoder: Analisi della Ricostruzione

Le capacità predittive del modello LSTM sono state valutate osservando la ricostruzione delle dinamiche su navi del test set non viste durante l'addestramento. L'analisi dei grafici evidenzia il comportamento tipico delle reti ricorrenti discrete quando applicate a segnali reali spesso affetti da quantizzazione (particolarmente visibile nei grafici della velocità SOG, che presentano un andamento "a gradini"). Il modello LSTM tende ad agire come un potente filtro passabasso: approssima i salti discreti dei dati reali con curve continue e morbide. Sebbene il trend macroscopico a lungo termine venga catturato efficacemente, il modello fatica a replicare le variazioni impulsive e ad alta frequenza, introducendo un errore evidente sia nella velocità che nei picchi della rotta (COG).

Confronto Dinamica su 3 Navi Casuali

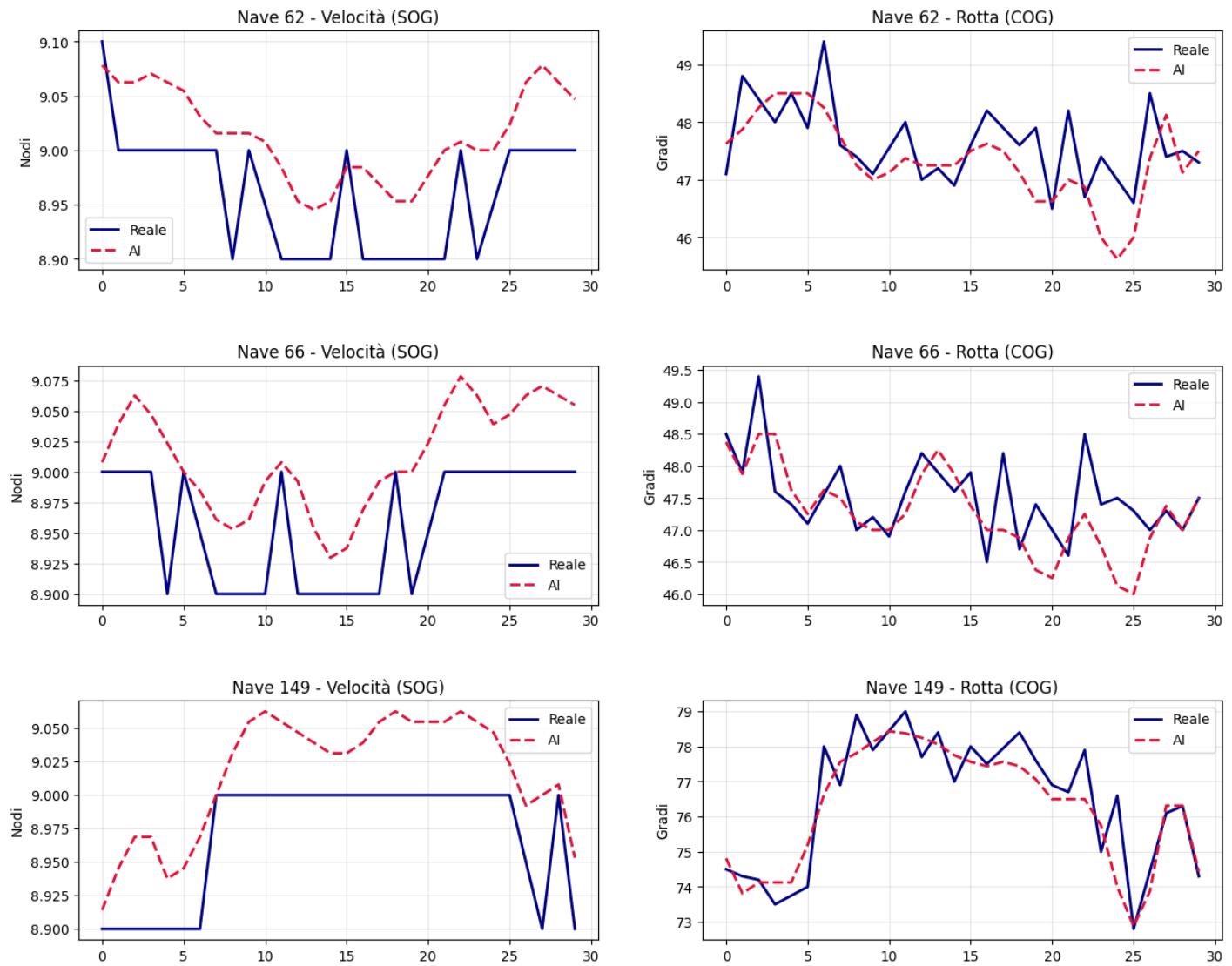


Figura 4.2: LSTM Capacità Predittiva Su Dati "Normali"

4.2.3 Liquid Neural Network (LNN)

Per la configurazione LNN, l'implementazione si concentra sulla gestione della dinamica continua tramite un cablaggio neurale sparso.

- **Wiring Sparso (AutoNCP):** È stata utilizzata la funzione AutoNCP(128, 64), generando un circuito con 128 unità ricorrenti e 64 di comando. Questo riduce drasticamente le sinapsi attive rispetto a una rete densa, migliorando l'interpretabilità.
- **Celle CfC:** Sono state impiegate celle *Closed-form Continuous-time* con `mixed_memory=True`, permettendo la gestione di dipendenze a lungo termine senza solver numerici lenti.
- **Stabilizzazione:** Applicato *Gradient Clipping* (0.5) e scheduler *Cosine Decay with Restarts* per gestire la complessità di addestramento delle ODE.

```

n_features = len(COLONNE_FEATURES)
latent_dim = 128
output_dim = 64
wiring = AutoNCP(latent_dim,output_dim) # Definisce una wiring sparsa

# Encoder
inputs = Input(shape=(WINDOW_SIZE, n_features))
# LAYER LIQUIDO 1 (Encoder): USIAMO WIRING SPARSA
lnn_encoder = CfC(wiring, return_sequences=False, mixed_memory=True)(inputs)

# Decoder
repeat_vector = RepeatVector(WINDOW_SIZE)(lnn_encoder)
lnn_decoder = CfC(wiring, return_sequences=True, mixed_memory=True)(repeat_vector)

output = TimeDistributed(Dense(n_features))(lnn_decoder)

model_lnn = Model(inputs, output)

model_lnn.summary()

```

Figura 4.3: LNN AutoEncoder

4.2.4 LNN AutoEncoder: Analisi della Ricostruzione

La valutazione del modello LNN mostra risultati qualitativamente diversi, derivanti dalla sua natura a tempo continuo. Le caratteristiche salienti delle ricostruzioni LNN sono la fluidità e la coerenza fisica delle traiettorie generate. Di fronte a dati di velocità fortemente quantizzati (es. Nave 44), la rete non cerca di replicare il "gradino" artificiale del sensore, ma interpola una dinamica di transizione continua e fisicamente plausibile tra i punti. Anche nel tracciamento della rotta (COG), il modello dimostra un'ottima capacità di seguire le oscillazioni principali. Vi è, tuttavia, una forte regolarizzazione che può portare a una leggera sottostima dei picchi più acuti, privilegiando una dinamica stabile.

Confronto Dinamica su 3 Navi Casuali

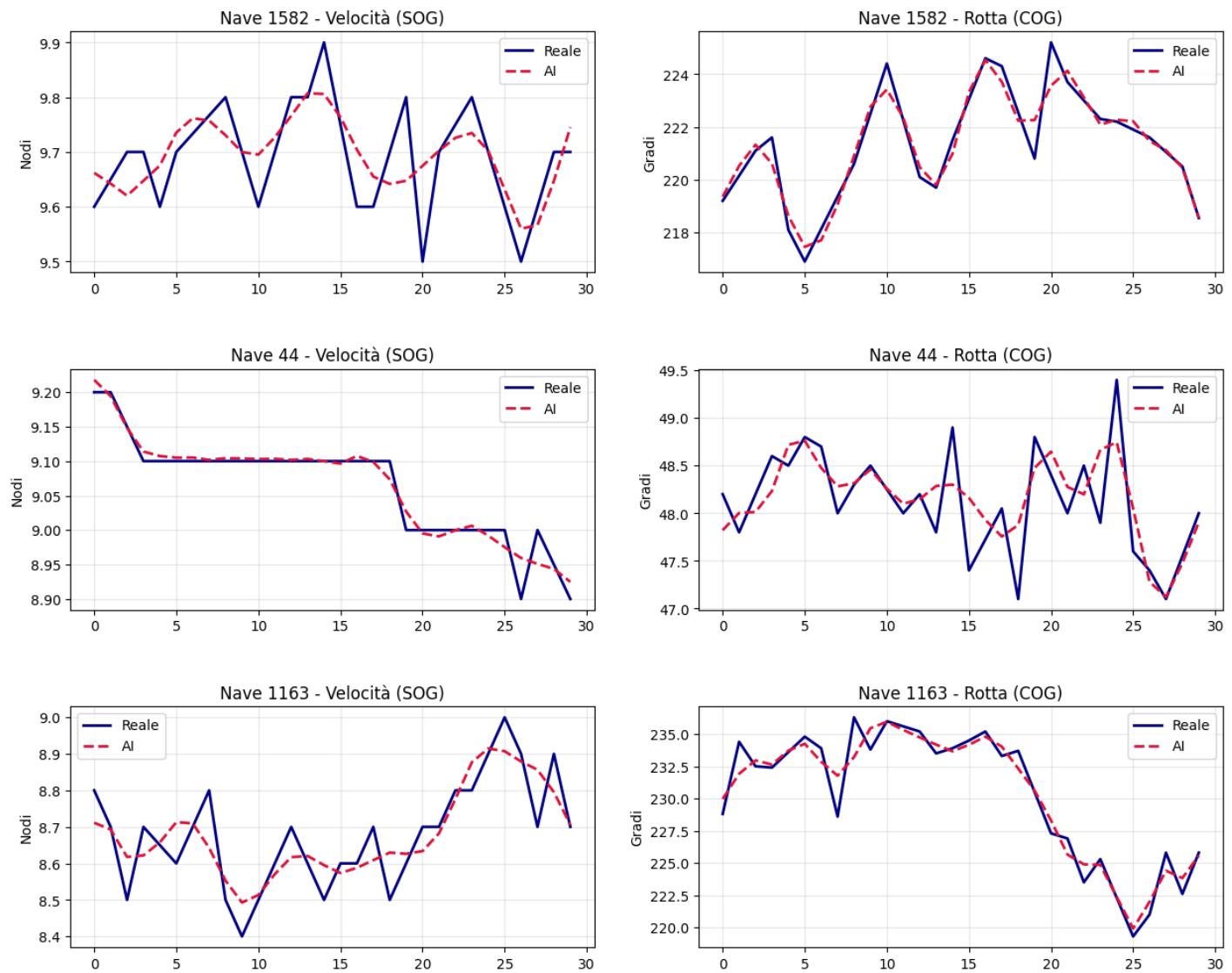


Figura 4.4: LNN Capacità Predittiva Su Dati "Normali"

4.2.5 Confronto Preliminare

L’analisi comparata delle ricostruzioni non decreta una superiorità netta di un modello sull’altro, ma evidenzia due approcci complementari alla modellazione delle serie storiche navali, ciascuno con un punto di forza distintivo:

- **LSTM:** Il principale pregio risiede nell’essere un approssimatore universale per sequenze discrete. Questo è dimostrato da un’elevata capacità di adattarsi alla distribuzione dei dati, minimizzando l’errore di ricostruzione. La sua architettura è estremamente efficace nel catturare pattern ricorrenti complessi all’interno della finestra temporale fissa.
- **LNN:** Il punto di forza è la capacità di apprendere non solo la sequenza di punti, ma la legge di variazione del sistema. Questo garantisce una rappresentazione indipendente dalla frequenza di campionamento e potenzialmente più stabile in scenari dove la dinamica temporale è prioritaria.

4.3 Risultati di Rilevamento

I modelli addestrati sono stati sottoposti a quattro scenari di attacco distinti. In questa fase presentiamo i tassi di rilevamento (*Recall*) puri, definiti come la percentuale di attacchi che hanno superato la soglia di sicurezza preliminare.

4.3.1 Anomalie Cinematiche (Speed Spoofing e Teleport)

Le violazioni macroscopiche della fisica sono state identificate con precisione assoluta.

- **Speed Spoofing:** Rilevato al **100%** da entrambi i modelli. La discrepanza tra velocità scalare e spazio percorso è immediata.
- **Teleport:** Rilevato al **100%** da entrambi i modelli. Il salto di coordinate viola i vincoli di inerzia appresi.

4.3.2 Anomalie Progressive (Silent Drift)

Nello scenario *Silent Drift* (deriva con rateo $\alpha = 0.01$), entrambi i modelli hanno raggiunto un Detection Rate del **100%**. L'errore accumulato lungo la finestra di 30 minuti risulta evidente per architetture dotate di memoria a lungo termine.

4.3.3 Anomalie Semantiche (Ghost Ship)

Nello scenario *Ghost Ship*, dove la rotta (COG) è incoerente con il vettore spostamento, si osserva la prima divergenza numerica significativa:

Modello	Detection Rate (Recall)
LSTM	59.43%
LNN	86.75%

Tabella 4.1: Tassi di rilevamento preliminari per lo scenario Ghost Ship.

4.3.4 Risultati LSTM

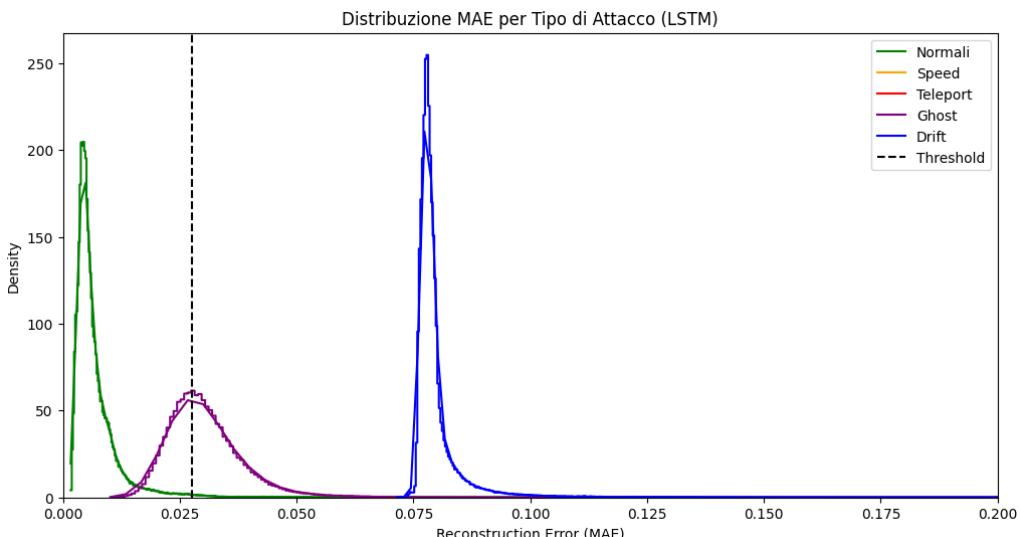


Figura 4.5: Distribuzione Errori per tipo di Attacco - LSTM

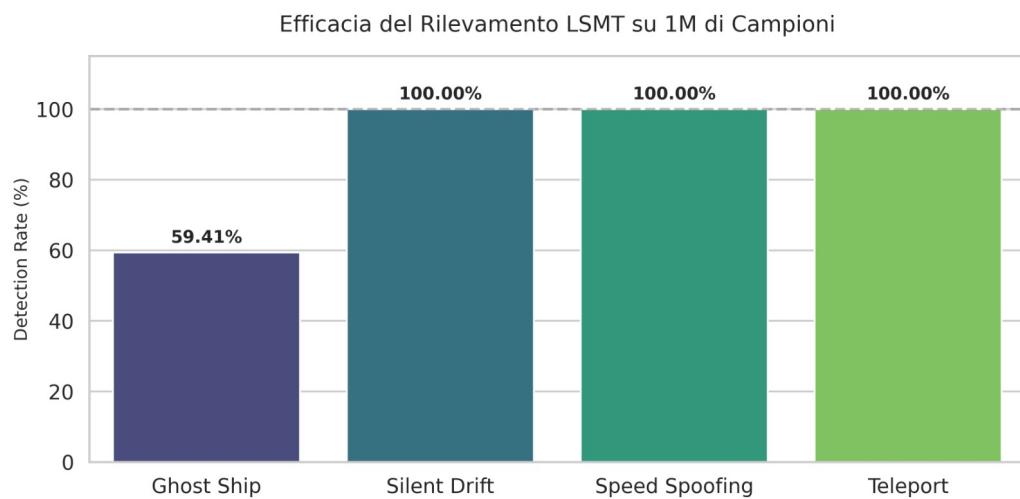


Figura 4.6: Efficacia - LSTM

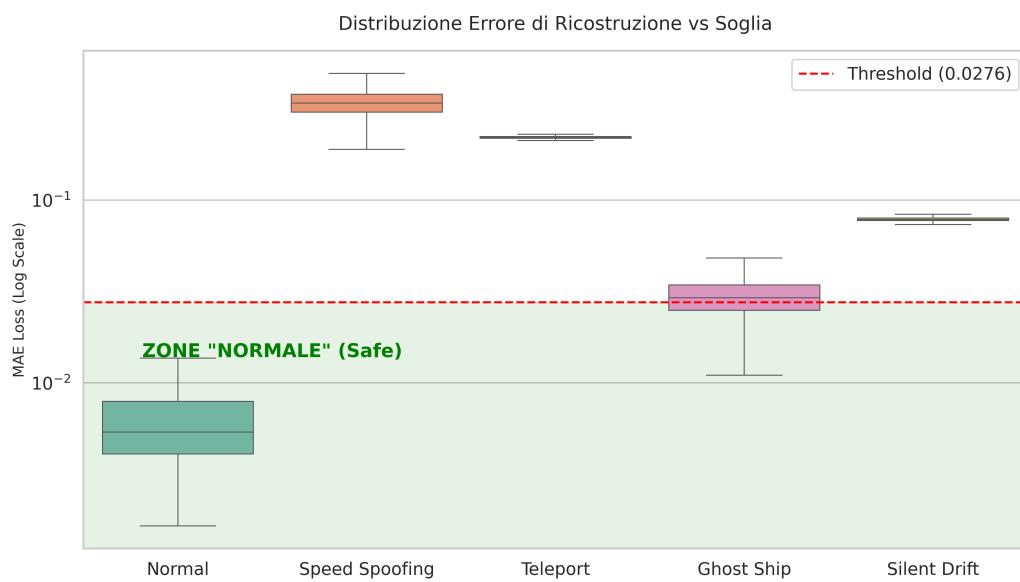


Figura 4.7: Distribuzione Errore di ricostruzione Soglia - LSTM

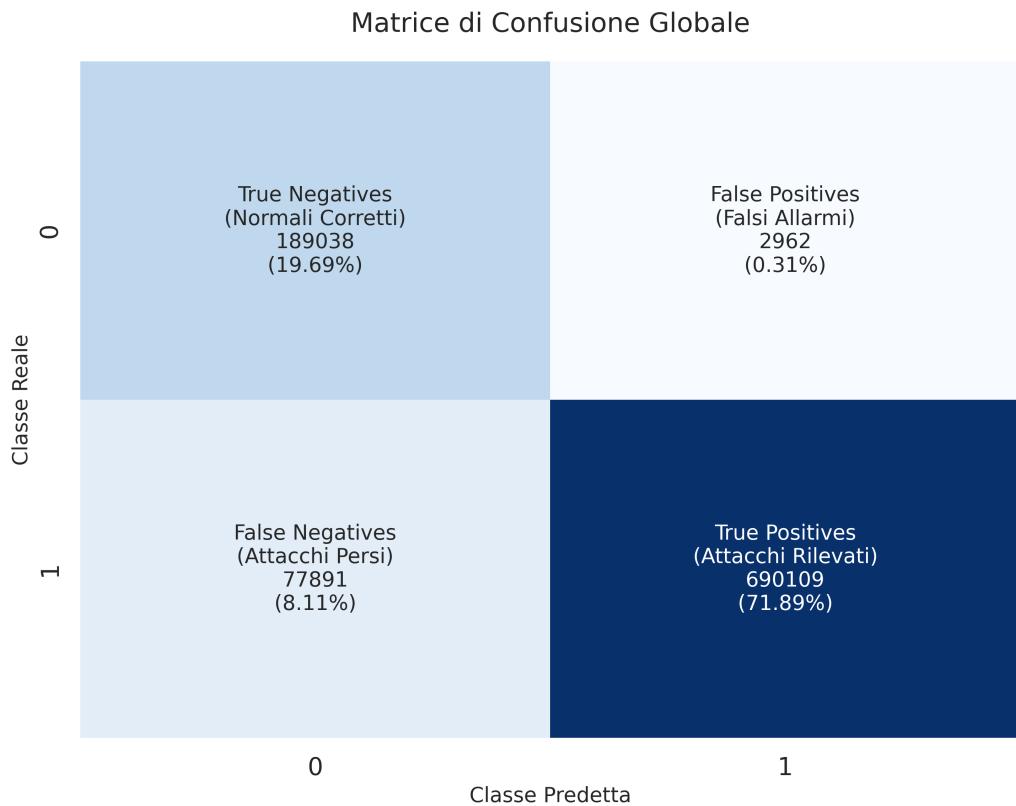


Figura 4.8: Matrice di confusione su dataset attaccato - LSTM

4.3.5 Risultati LNN

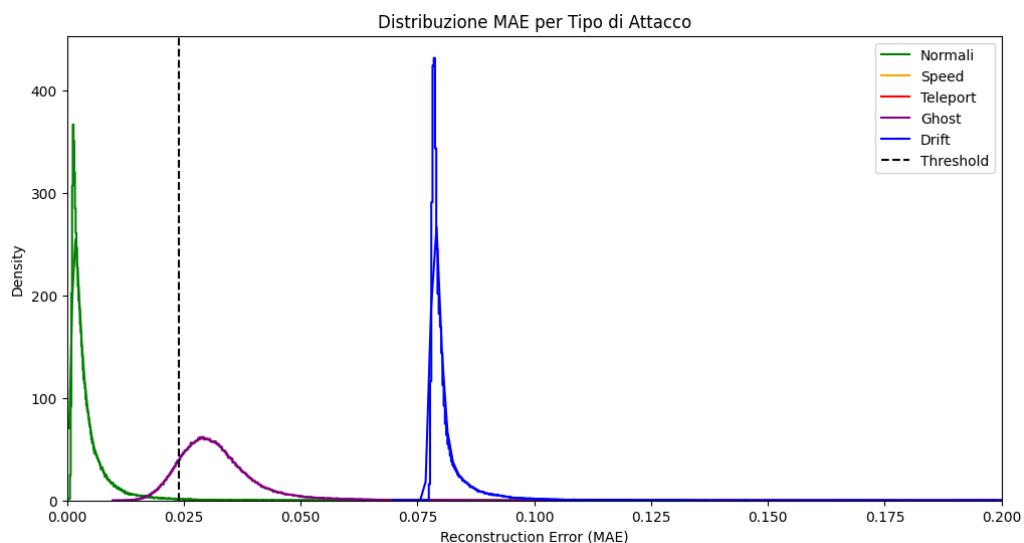


Figura 4.9: Distribuzione Errori per tipo di Attacco - LNN

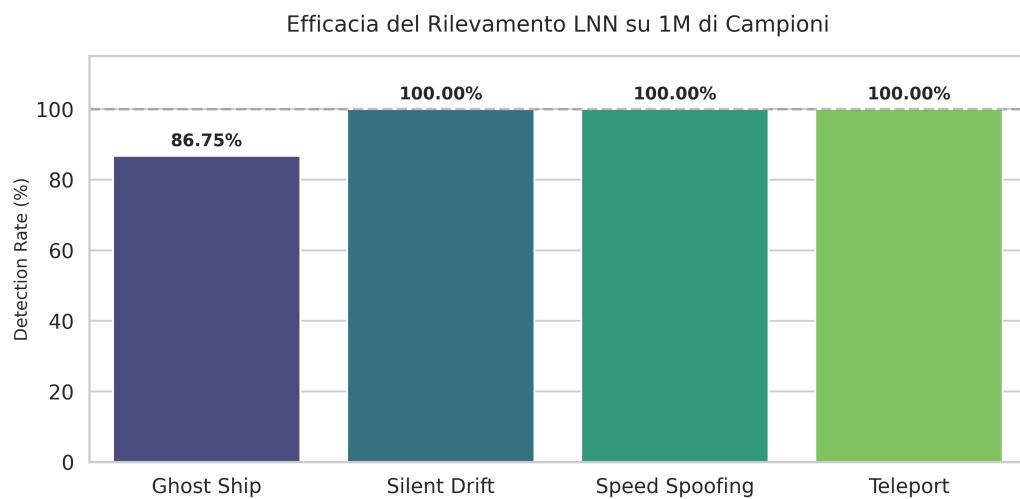


Figura 4.10: Efficacia - LNN

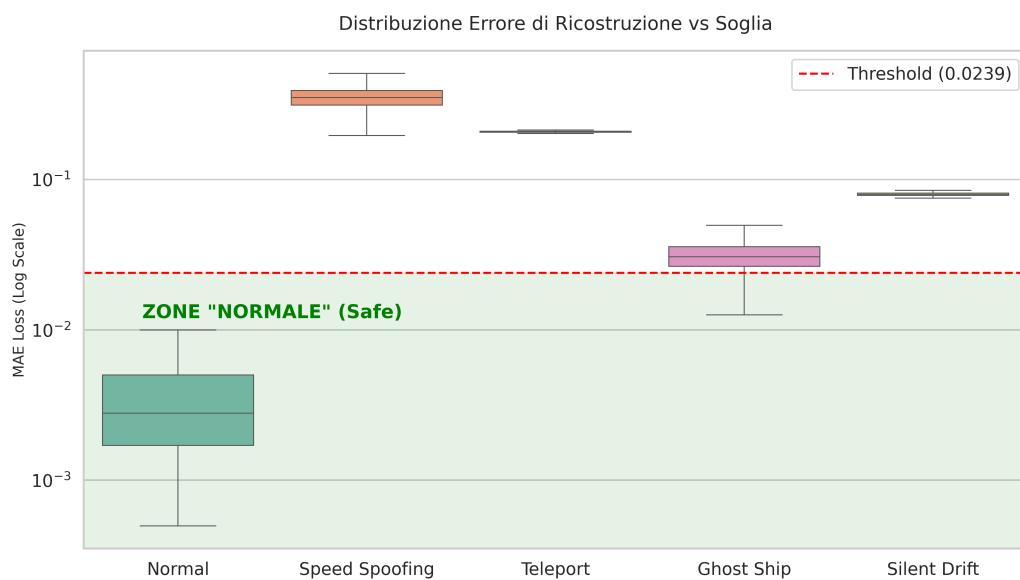


Figura 4.11: Distribuzione Errore di ricostruzione Soglia - LNN

Matrice di Confusione Globale

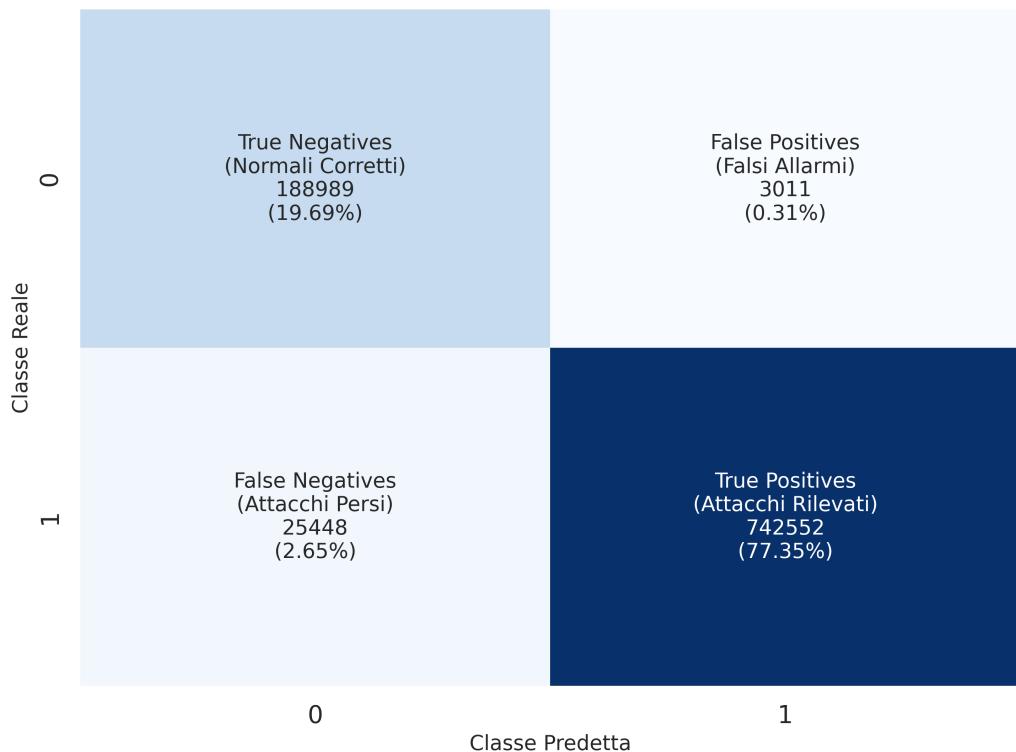


Figura 4.12: Matrice di confusione su dataset attaccato - LNN

Capitolo 5

Analisi Critica e Discussione dei Risultati

5.1 Dinamicità della Soglia e Trade-off Operativo

Un’analisi approfondita delle prestazioni non può prescindere dalla natura della soglia di allarme (τ). In questo studio, la soglia non è una costante universale, ma un valore **dinamico** che dipende interamente dalla distribuzione statistica del dataset. La formula utilizzata ($\tau = \mu + 3\sigma$) è una convenzione statistica standard, ma non rappresenta un limite fisico invalicabile.

5.1.1 La Relatività del Rilevamento

La soglia rappresenta un parametro puramente operativo che definisce la sensibilità del sistema di sicurezza:

- **Soglia Alta (Conservativa):** Mantenendo la soglia attuale, si privilegia la continuità operativa. Si accetta che alcune anomalie sottili (con errore di ricostruzione $MAE < \tau$) non vengano rilevate pur di mantenere i falsi allarmi a un livello gestibile ($\sim 2.5\%$).
- **Soglia Bassa (Aggressiva):** Se l’obiettivo fosse la “perfezione di rilevamento” ($Recall \rightarrow 100\%$), la soglia potrebbe essere arbitrariamente abbassata. Tuttavia, i dati dimostrano che per catturare il 100% degli attacchi *Ghost Ship* o *Silent Drift* iniziali, dovremmo accettare un aumento dei Falsi Positivi.

In sintesi, la capacità di rilevamento non è una proprietà intrinseca della rete neurale isolata, ma una funzione diretta del costo (in termini di falsi allarmi) che l’operatore è disposto a pagare.

5.2 Analisi Comparativa: Stabilità del Modello e Separabilità delle Classi

Osservando i risultati aggregati del Capitolo 4, la LNN mostra una prevalenza significativa sulla LSTM nello scenario *Ghost Ship* ($\sim 87\%$ vs $\sim 58\%$ di Recall). Un’analisi approfondita dei valori di errore di ricostruzione (MAE) permette di attribuire questo divario non a una diversa sensibilità all’attacco, ma a una diversa stabilità sul traffico lecito.

5.2.1 Varianza e Definizione del Margine di Sicurezza

Entrambe le reti, LSTM e LNN, hanno raggiunto un’ottima convergenza durante il training. Tuttavia, l’analisi della distribuzione dell’errore ha rivelato due aspetti importanti:

- **Risposta all'Attacco:** Di fronte all'incoerenza cinematica di un attacco Ghost Ship, entrambe le reti reagiscono generando un errore di ricostruzione assoluto simile (≈ 0.026).
- **Comportamento sul Normale:** La LNN presenta una varianza (σ) inferiore sul traffico lecito rispetto alla LSTM.

Questa differenza di varianza ha implicazioni dirette sul calcolo della soglia. La LSTM, essendo intrinsecamente più "rumorosa" sul traffico normale, impone una soglia di sicurezza più alta ($\tau \approx 0.0276$). Di conseguenza, l'errore generato dall'attacco (0.026) cade al di sotto di questa soglia, risultando in un Falso Negativo. Al contrario, la maggiore stabilità della LNN permette di fissare una soglia più stretta ($\tau \approx 0.0239$). In questo scenario, lo stesso errore di attacco (0.026) supera la soglia, venendo correttamente classificato come anomalia.

Profilo Operativo	Soglia (τ)	Detection Rate	FPR
Aggressivo	0.01572	99.50%	~6.53%
Conservativo	0.02761	57.80%	~2.50%

Tabella 5.1: Performance del modello LSTM su 1M campioni di test

Profilo Operativo	Soglia (τ)	Detection Rate	FPR
Aggressivo	0.01693	99.50%	~4.18%
Conservativo	0.02395	87.03%	~2.51%

Tabella 5.2: Performance del modello LNN su 1M campioni di test

5.3 Limiti Intrinseci: Cecità Spaziale e Apprendimento Cinematico

Un risultato fondamentale emerso dall'analisi qualitativa delle ricostruzioni 2D (visualizzazione delle traiettorie predette vs reali) è l'assenza di ancoraggio geografico da parte dei modelli. Sia in situazione di attacco sia in circostanze "normali", le reti non riescono a riposizionare correttamente la nave sulla mappa assoluta. Le predizioni non risultano essere traiettorie caotiche, bensì curve cinematicamente plausibili ma affette da un significativo *offset* spaziale, che le rende geometricamente incoerenti rispetto al riferimento cartografico.

5.3.1 Analisi dell'Apprendimento Effettivo

Questo comportamento evidenzia che le reti non hanno appreso una mappa topologica implicita (non distinguono terra da mare, né identificano corridoi marittimi assoluti), ma hanno appreso esclusivamente la **legge di correlazione differenziale** tra le feature. I modelli hanno interiorizzato che una specifica velocità (SOG) e un angolo di rotta (COG) devono corrispondere matematicamente a un preciso incremento vettoriale (ΔLat , ΔLon) nel passo temporale successivo.

La "Cecità Spaziale" è confermata sperimentalmente: i modelli non rilevano anomalie posizionali statiche (es. attraversamento di terraferma con dinamica coerente), ma reagiscono istantaneamente a incongruenze vettoriali (es. discrepanza tra direzione dichiarata e spostamento effettivo).

In conclusione, il sistema attuale opera come un **validatore di coerenza cinematica differenziale** (verifica la derivata del moto), ma fallisce come sistema di monitoraggio geografico assoluto.

Capitolo 6

Sviluppi Futuri

Alla luce delle limitazioni emerse durante la sperimentazione, in particolare la “cecità spaziale” dei modelli e la sensibilità operativa alla soglia di allarme, si propongono tre direzioni di sviluppo.

6.1 Risoluzione della Cecità Spaziale

L’analisi sperimentale ha evidenziato che l’attuale architettura (LNN/LSTM), ottimizzando la funzione di costo MAE, tende a privilegiare la coerenza cinematica (SOG, COG) a discapito della precisione posizionale assoluta. Per superare questa limitazione intrinseca senza ricorrere a validatori esterni, si propone l’evoluzione del modello in un **Conditional Autoencoder**. Tale approccio modifica il paradigma di apprendimento da una stima non vincolata a una stima condizionata dalla posizione iniziale $c = (LAT_0, LON_0)$. Questa soluzione garantisce che la rete mantenga la sensibilità alle anomalie cinematiche, risolvendo al contempo la "cecità spaziale" mediante un vincolo geometrico strutturale e non appreso.

6.2 Soglia Adattiva tramite Clustering Contestuale

Lo studio ha dimostrato che una soglia fissa ($\tau = \mu + 3\sigma$) è sub-ottimale: è troppo permissiva per navi in navigazione rettilinea (rischio di *Silent Drift* non rilevati) e troppo severa per navi in manovra complessa (rischio di falsi positivi). Uno sviluppo futuro realizzabile prevede l’abbandono della soglia statica in favore di una **Soglia Contestuale** basata su tecniche di Clustering non supervisionato (es. K-Means o DBSCAN).

- **Identificazione dello Stato:** Prima di passare all’Autoencoder, i dati vengono classificati in cluster operativi (es. “Ormeggio”, “Manovra in Porto”, “Navigazione in Mare Aperto”).
- **Soglie Dedicate:** Per ogni cluster viene calcolata una soglia τ specifica. Ad esempio, in “Mare Aperto” la varianza attesa è bassa, quindi la soglia sarà molto stretta (aumentando la sensibilità agli attacchi sottili); in “Manovra”, la soglia sarà più alta per tollerare la varianza naturale senza generare falsi allarmi.

6.3 Interpretabilità e Supporto Decisionale

Attualmente, il sistema agisce come una “Black Box”: fornisce un valore di errore aggregato (MAE totale) senza spiegare la causa dell’anomalia. Per un operatore umano, sapere che c’è

un attacco non basta, serve sapere cosa sta succedendo. Un’evoluzione naturale del software consiste nel passare da un errore scalare a un **Vettore di Errore**:

- Invece di mediare l’errore su tutte le feature, il sistema dovrebbe calcolare e mostrare il contributo di errore delle singole componenti ($MAE_{Lat/Lon}$, $MAE_{Velocit}$, MAE_{Rotta}).
- **Diagnostica Automatica:** Se il picco di errore proviene solo dalla velocità, il sistema suggerisce un probabile “Speed Spoofing”; se proviene dalla rotta ma non dalla posizione, suggerisce un “Ghost Ship”. Questo approccio non richiede nuove reti neurali, ma solo una diversa post-elaborazione dell’output già esistente, aumentando drasticamente l’utilità del tool.

Conclusioni

Il presente lavoro ha affrontato una delle sfide presenti nell'ambito della sicurezza delle infrastrutture critiche: la protezione del sistema di tracciamento navale AIS da attacchi di spoofing di traiettoria. Attraverso la progettazione, l'addestramento e la validazione di due architetture di Deep Learning, una basata sullo standard consolidato delle **Long Short-Term Memory (LSTM)** e una sull'approccio innovativo delle **Liquid Neural Networks (LNN)**, è stato possibile dimostrare come l'Intelligenza Artificiale possa trascendere i limiti dei sistemi di controllo tradizionali.

Superiorità Tecnologica delle LNN

L'analisi comparativa condotta su oltre 215 milioni di messaggi ha delineato un quadro chiaro sull'evoluzione delle architetture neurali. Sebbene entrambi i modelli abbiano raggiunto l'eccellenza nel rilevamento di anomalie cinematiche macroscopiche, la **Liquid Neural Network** si distingue come un significativo passo in avanti nella modellazione di sistemi fisici complessi.

- **LSTM:** Si conferma una soluzione robusta e matura per serie storiche discrete capace di generalizzare efficacemente le leggi del moto con un costo computazionale contenuto.
- **LNN (Il Passo Avanti):** L'approccio a tempo continuo basato su equazioni differenziali ha dimostrato una **stabilità superiore**. La minore varianza riscontrata sul traffico lecito è la prova che il modello "comprende" la dinamica fisica meglio della controparte discreta.

Mentre la LSTM richiede soglie di sicurezza più ampie per gestire l'incertezza, la LNN permette una calibrazione più fine e aggressiva. Pertanto, pur non essendoci un divario netto in termini di Recall su attacchi semplici, la LNN rappresenta l'architettura di riferimento per gli sviluppi futuri, offrendo una base più solida per distinguere sottili anomalie dal rumore di fondo.

Prospettive Future

In conclusione, questo studio ha gettato le basi per un sistema di difesa proattivo. Il futuro di questa ricerca non risiede nella mera sostituzione di un modello con l'altro, ma nella loro evoluzione architettonica verso una maggiore consapevolezza del contesto. In conclusione, il passaggio da modelli teorici a strumenti di difesa operativi richiede ora di ancorare saldamente l'analisi cinematica al contesto geografico reale. Solo colmando questo divario sarà possibile garantire un monitoraggio affidabile e proteggere efficacemente il dominio marittimo dalle minacce cyber-fisiche più insidiose.