Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

UTILIZZO DI SOFT-BROWNIAN-OFFSET PER LA GENERAZIONE DI ATTACCHI AI FINI DELL'ADDESTRAMENTO DI RILEVATORI DI INTRUSIONI

APPLICATION OF SOFT-BROWNIAN-OFFSET TO GENERATE CYBER-ATTACKS TO TRAIN INTRUSION DETECTORS

GUGLIELMO BARTELLONI

Relatore: *Relatore*
Correlatore: *Correlatore*

Anno Accademico 2022-2023

INDICE

ELENCO DELLE FIGURE

"If you put water into a cup, it becomes the cup. You put water into a bottle, it becomes the bottle. You put it in a teapot, it becomes the teapot. Now, water can flow or it can crash. Be water, my friend."

— Bruce Lee

1

INTRODUZIONE

Un sistema di rilevamento di intrusioni, anche detto intrusion detection system (IDS), è un'applicazione o un dispositivo volta a monitorare continuamente la rete per identificare attività malevole.

In particolare un IDS controlla il traffico di rete oppure i log di sistema, alla ricerca di possibili anomalie che potrebbero indicare la presenza di attacchi.

Per migliorare il rilevamento delle intrusioni, nell'ultimo decennio, si è iniziato ad utilizzare algoritmi di Machine Learning e Deep Learning attingendo informazioni dai Big Data. [4]

Questo però ha portato a nuove problematiche, come ad esempio il fatto che i modelli di Machine Learning sono molto sensibili ai dati di addestramento.

Un dataset contenente pacchetti di rete sarà composto per la maggior parte da pacchetti "normali", cioè pacchetti di traffico abituale e, per la restante parte, da pacchetti di attacchi. I vari dataset sul traffico di rete presenti oggi però, non hanno una quantità sufficiente di attacchi per poter addestrare al meglio i modelli.

Un modello addestrato su un dataset con una percentuale di attacchi troppo bassa, non sarà in grado di rilevare bene gli attacchi [5].

Un metodo per sopperire a questa mancanza è quello di generare nuovi dati a partire da quelli già esistenti (Data Augmentation). I dati generati devono essere però sufficientemente differenti di modo da rappresentare meglio le anomalie che si hanno nella rete durante un attacco.

In questa tesi esploreremo un algoritmo di Data Augmentation chiamato Soft-Brownian-Offset (SBO) [6] e lo utilizzeremo per generare nuovi dati per addestrare un IDS.

In particolare Soft-Brownian-Offset permette di generare campioni così detti out-of-distribution (OOD), cioè campioni che non fanno parte della distribuzione dei dati di partenza. Questo è coerente col fatto che di solito i pacchetti di attacchi sono fuori dalla distribuzione dei pacchetti comuni.

2

FONDAMENTI

2.1 INTRUSION DETECTION SYSTEM

Un intrusione può essere definita come un evento che compromette integrità, confidenzialità e la disponibilità di un sistema informatico [7].

Gli Intrusion Detection System (IDS) sono delle soluzioni hardware o software che, posti all'interno di una rete o di un sistema, rilevano eventuali intrusioni.

Questi strumenti sono essenziali per tenere al sicuro le persone da attacchi informatici [1].

[8] Le principali funzioni degli IDS sono:

- Monitorare ed analizzare sia le attività utente che di sistema
- Tracciare le violazioni delle policy utente
- Analizzare le configurazioni e le vulnerabilità del sistema
- Rilevare tipici attacchi di rete
- Analizzare di attività anomale

[9] Solitamente gli IDS vengono classificati in base al tipo di analisi che effettuano e come questi rilevano le minacce. Ne esistono di tre tipi principali:

- Signature-Based Detection (SD)
- Anomaly-based Detection (AD)
- Stateful Protocol Analysis (SPA)

2.1.1 *Signature-Based Detection*

Questo tipo di rilevamento utilizza la firma di un attacco per poterlo rilevare. Quindi conoscendo questa firma, gli IDS la comparano agli eventi catturati della rete. Dato che questi attacchi hanno bisogno di una conoscenza pregressa sono anche chiamati Knowledge-based.

2.1.2 *Anomaly-based Detection*

Gli Anomaly-based Intrusion Detection Systems (AIDS) sono stati introdotti per sopperire alle mancanze del Signature-Based Detection. Questo tipo di rilevamento utilizza un modello che rappresenta il normale comportamento della rete. Se viene rilevato un evento che non è coerente con il modello di riferimento, allora viene segnalata un'anomalia. Questo tipo di rilevamento è chiamato anche Behavior-Based.

Il principale vantaggio di questo tipo di IDS è la possibilità di rilevare gli attacchi zero-day [10], in quanto questi sistemi, non si basano sulla firma dei dati o su regole rigide. Tra gli altri vantaggi si ha che risulta essere difficile, per un eventuale criminale, capire quale sia il comportamento normale di un utente senza produrre un segnale da parte del sistema [1].

Gli AIDS possono essere classificati in base al metodo utilizzato per la loro implementazione:

- Basati sulla Statistica (Statistical-Based)
- Basati sulla Conoscenza (Knowledge-Based)
- Basati sull'apprendimento automatico (Machine Learning-Based)

Gli AIDS Statistical-Based dopo aver registrato i dati di una porzione di elementi, derivano il modello statistico di un utente nella rete.

I Knowledge-based invece, utilizzano regole predefinite per generare il modello di riferimento. Dall'altra parte troviamo gli AIDS Machine Learning-Based, che utilizzano un algoritmo di apprendimento automatico per generare il modello di riferimento.

La figura 1 mostra più in dettaglio questo tipo di classificazione e i metodi di implementazione associati.

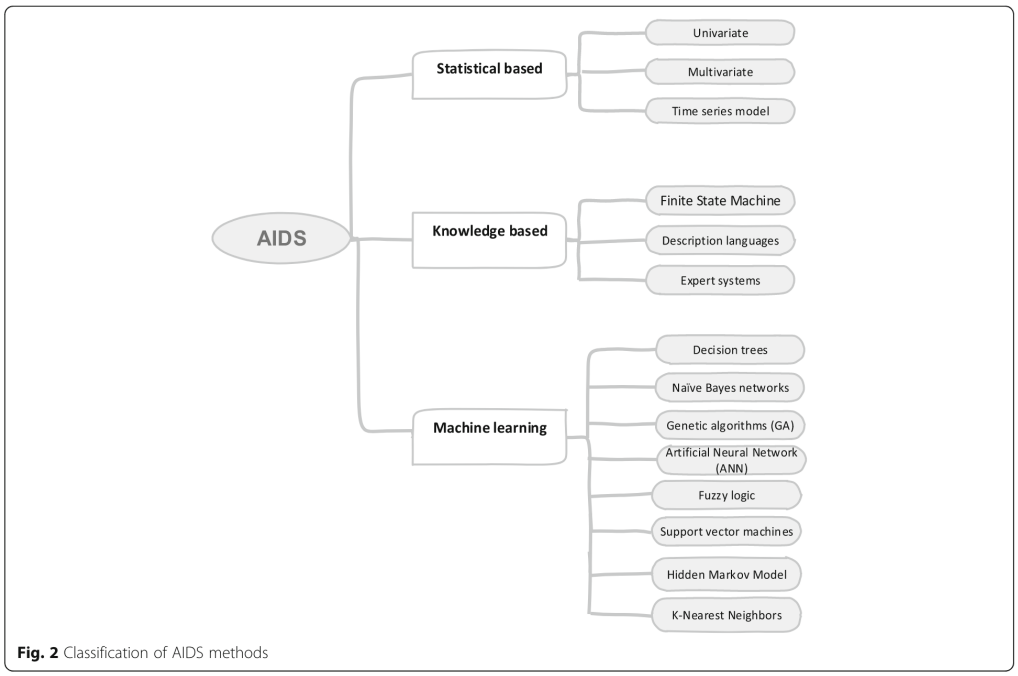

Fig. 2 Classification of AIDS methods

Figura 1: Schema che riassume i vari metodi di implementazione degli AIDS, da [1]. I modelli sono divisi nelle tre categorie descritte precedentemente, ma sono presenti anche i vari tipi di algoritmi utilizzati.

2.1.3 *Stateful-Protocol-Analysis*

In questo caso gli IDS conoscono lo stato e le specifiche del protocollo utilizzato. Vengono quindi rilevati degli eventi che non rispettano gli standard del protocollo, generalmente quelli da specifica e.g. IEEE.

Potrebbe sembrare che gli AD e gli SPA siano simili, in realtà i primi, conoscono il comportamento di una specifica rete, mentre i secondi, conoscono solo gli standard dei protocolli.

2.2 MACHINE LEARNING PER RILEVAMENTO DI INTRUSIONI

Il Machine Learning (ML) è un sottoinsieme dell'intelligenza artificiale e permette ai sistemi di imparare dai dati e di migliorare le loro prestazioni nel tempo senza essere esplicitamente programmati. Nel caso degli Intrusion Detection Systems, gli algoritmi di ML permettono di rilevare in maniera precisa e rapida gli attacchi per grandi quantità di dati in poco tempo [11].

Solitamente questi algoritmi sono divisi in:

- Supervised
- Unsupervised

Inoltre gli algoritmi di ML possono essere usati per la classificazione o per la predizione.

Il processo di classificazione si compone di vari passi tra cui:

- Preprocessamento dei dati
- Definizione dei dati di test
- Scelta dell'algoritmo
- Impostazione dei parametri
- Addestramento del modello
- Test del modello

come mostrato in figura 2.

Il modello viene addestrato su un dataset con classi conosciute e cerca di trovare una relazione tra le caratteristiche dei dati (features) e le classi stesse. In particolare un classificatore mappa una funzione:

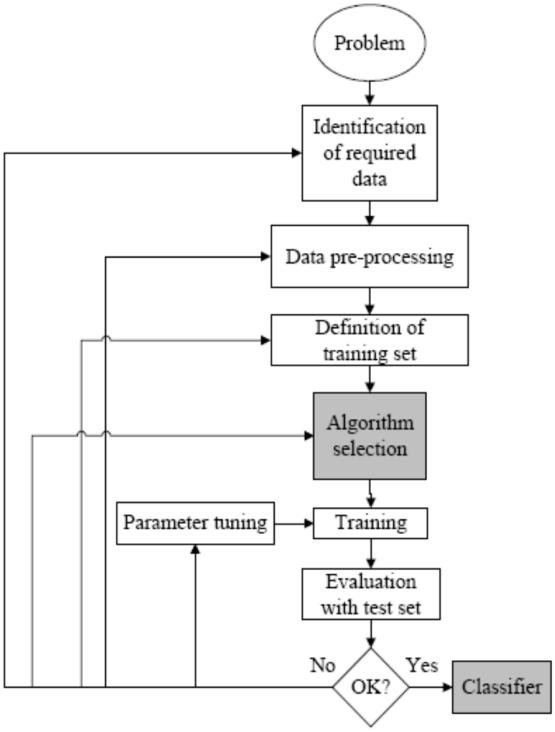

Figura 2: Il processo del Supervised Machine Learning, da [2]

$$f x_1, \ldots x_S \; \Theta$$

che assegna uno scalare facente parte di un insieme C di elementi disgiunti (le classi), ad un insieme di S elementi vettoriali (le caratteristiche) [12].

Nella regressione invece, si cerca di approssimare la funzione:

$$f x_1, \ldots x_S \; y$$

a partire dalle caratteristiche. La funzione f è approssimata utilizzando varie tecniche di interpolazione, estrapolazione, analisi di regressione e curve fitting [12].

Nel nostro caso utilizzeremo un algoritmo di classificazione.

Supervised Machine Learning

Il Supervised Machine Learning utilizza un dataset con classi completamente etichettate cercando di trovare una relazione tra gli elementi del dataset e le classi di questi dati. I dataset utilizzati per l'addestramento, necessitano di essere catalogati precedentemente. Ci sono vari algoritmi di questo tipo e.g. Support Vector Machine (SVM), Naïve Bayes, Reti Neurali, Alberi di Decisione, Random Forest.

Unsupervised Machine Learning

In questo caso, gli algoritmi di ML non hanno un dataset con classi etichettate. Questi algoritmi traovano delle relazioni tra i dati, cercando di raggrupparli in base a delle caratteristiche comuni.

L'unsupervised machine learning mostra però basse prestazioni per quanto riguarda il rilevamento quindi non possono essere utilizzati come principale strumento per gli IDS. Il motivo di questo scarso rendimento è dovuto al fatto che è molto probabile generare dei Falsi Positivi (vengono rilevati attacchi quando in realtà non cene sono) oppure generare dei Falsi Negativi (non vengono rilevati attacchi quando in realtà ce ne sono) questo va a degradare le performance generali del modello.

D'altra parte però questi modelli si sono dimostrati opinabilmente migliori per rilveare gli attacchi zero-day. [10].

2.3 XGBOOST

XGBoost (eXtreme Gradient Boosting) è un algoritmo di Machine Learning che utilizza alberi di decisione per la classificazione e la regressione.

Nel mondo reale è utilizzato anche la predizione dei click per le pubblicità [13] e come algoritmo per le competizioni su Kaggle [14].

Il suo punto di forza principale è la grande scalabilità in tutti gli scenari. XGBoost ha prestazioni superiori di dieci volte rispetto alle soluzioni esistenti. Inoltre riesce a sfruttare al meglio la potenza di calcolo della macchina su cui viene eseguito [14]. È stato scelto XGBoost come algoritmo in questa tesi per i motivi sopra citati.

2.3.1 *Alberi di Decisione*

Nella sua parte fondamentale XGBoost utilizza gli alberi di decisione.

Un albero di decisione è uno strumento relativamente semplice per classificare i dati dove vengono poste delle domande che riguardano le features. Ogni domanda è contenuta in un nodo dell'albero, ed oguno di questi punta ad un figlio che rappresenta una possibile risposta. In questo modo le domande formano una gerarchia (Figura 3), codificata come un albero [3].

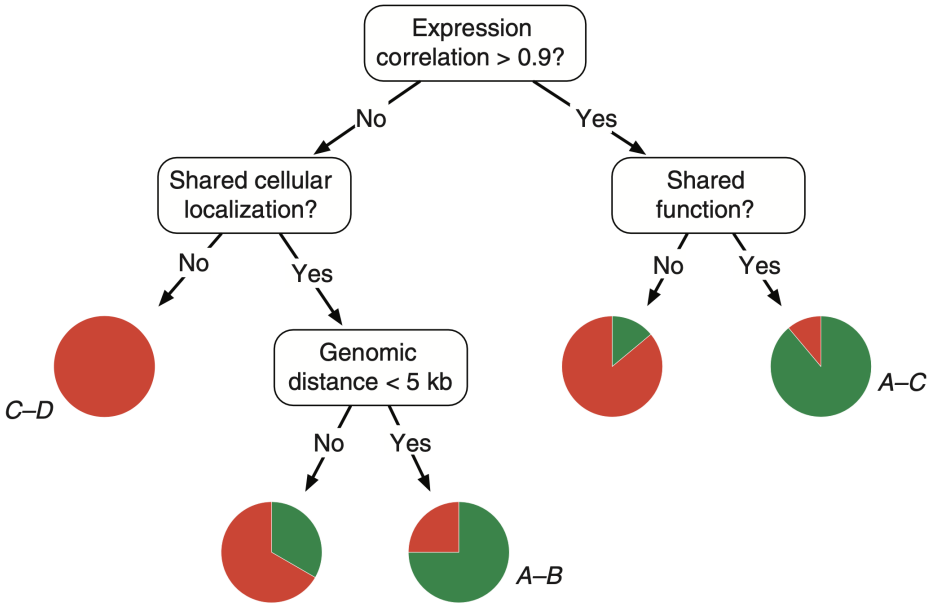

Figura 3: L'immagine mostra un esempio di albero di decisione di Carl Kingsford e Steven L Salzberg [3]

L'algoritmo più conosciuto che utilizza alberi di decisione è il C4.5 [15].

2.4 DATASET

I dataset sono un insieme di dati utilizzati per l'addestramento e il test dei modelli di ML. In questa tesi sono stati utilizzati tre dataset: ADFA-NET, CIDDS e CIC-IDS18.

2.4.1 *ADFA-NET*

ADFA è un dataset creato da un gruppo di ricerca dell'Università di New South Wales (Australia) nel 2013 e include dieci tipi di attacchi. Contiene in particolare attacchi di Brute Force, Java Based Meterpreter, Add new Superuser, Linux Meterpreter payload e attacchi alla C100 Webshell. [16]

Questo è il più semplice dataset su cui sono stati effettuati i test.

2.4.2 *CIDDS-18*

Solitamnete i dataset non rappresentano al meglio la realtà, in quanto per motivi di privacy questi vengono offuscati e anonimizzati perdendo molta della diversificazione utile per l'addestramento di un modello. Per sopperire a queste mancanze è stato creato CIC-IDS cercando di mantenere il più possibile caratteristiche di uno scenario reale. Esso contiene una grande quantità di dati ricoprendo le principali tipologie di attacchi che si possono trovare in una rete, tra questi abbiamo DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltrazione, Port scan e Botnet [16].

Questo risulta di gran lunga il più complesso dataset utilizzato in questa tesi contenendo più di settanta colonne e quasi duecentomila righe.

3

SOFT BROWNIAN OFFSET

Soft Brownian Offset (SBO) è un algoritmo per la generazione di dati sintetici fuori dalla distribuzione (Out of Distribution OOD).

È stato sviluppato da [6] inizialmente per poter migliorare il rilevamento dei dati fuori dalla distribuzione, si pensi ad esempio nel caso dei cyber-physical systems (CPS) e.g. auto a guida autonoma, dove il rilevamento di eventi anomali è di vitale importanza [17].

Per questo scopo i dati generati necessitano di essere non troppo dissimili da quelli originali, perché altrimenti il modello potrebbe soffrire di "overconfidence" [18].

SBO si basa sul "Gaussian Hyperspheric Offset" che cerca di creare i dati OOD campionando attraverso una ipersfera attorno ai dati iniziali nella distribuzione (In Distribution).

Partendo da quest'ultimo algoritmo, SBO permette di utilizzare qualsiasi punto del dataset e traslarlo in modo da avere una certa distanza minima dagli altri punti. I dati vengono processati da un autoencoder (AE) addestrato sul dataset di riferimento. La figura 4 mostra graficamente questo approccio.

Soft Browninan Offset è ispirato al moto browniano e traduce il concetto nella sua rappresentazione in R^D usando le ipersfere.

L'algoritmo cerca di traslare i punti, iterativamente, in modo da avere una distanza minima chiamata d_{min} dal punto originale. All'aumentare del valore d_{min} si aumenta quindi la distanza dei punti sintentici dai punti originali, questo lo si può notare dalla figura 5 dove è presente anche il parametro $softness$ utilizzato per regolare il riempimento delle aree vuote.

3.1 AUTOENCODER

Un autoencoder è una particolare rete neurale che ha il compito di codificare un input e comprimerlo in una rappresentazione rilevamente per poi decodificarlo e ricostruirlo nel modo più fedele possibile [19].

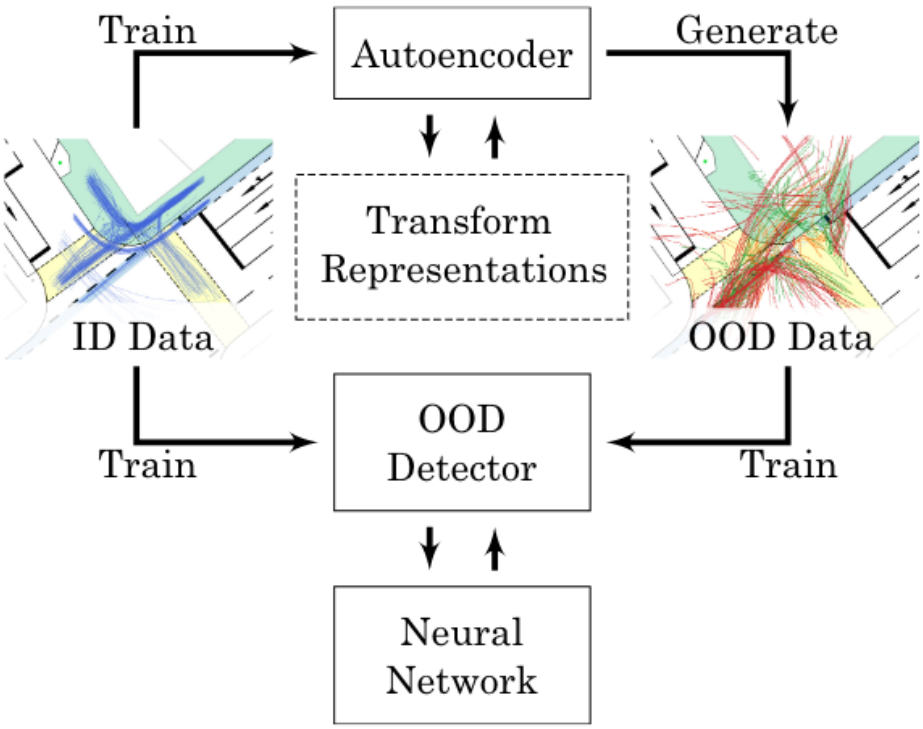

Figura 4: Schema che rappresenta l'approccio utilizzato da SBO. Inizialmente i dati ID vengono usati per addestrare l'autoencoder e successivamente vengono generati i dati OOD tramite SBO. Questi poi vengono decodificati dall'AE per integrarsi nel dataset iniziale. Infine, viene addestrata una rete neurale, nel nostro caso invece, utilizzeremo XGBoost.

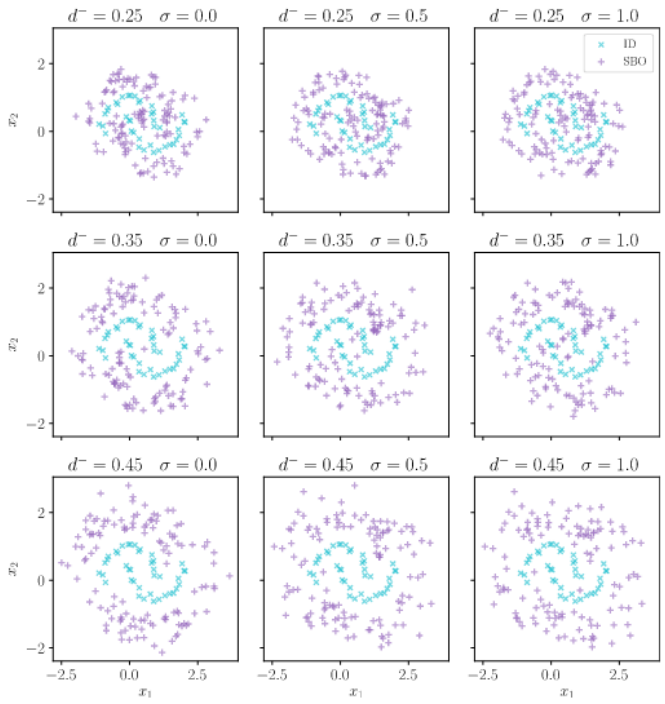

Figura 5: La figura rapprenta come si distribuiscono i dati sintettici (in lilla) rispetto ai dati originali (in azzurro) al variare del parametro d_{min} e $softness$.

Il loro compito principale è quindi di imparare, in modo non supervisionato, una rappresentazione ridotta significativa dei dati.

Il problema può essere formalmente definito [20], come trovare le funzioni $A{:}R^n \rightarrow R^p$ (encoder) e $B{:}R^p \rightarrow R^n$ (decoder) in modo che:

$$argmin_{A,B} E[\Delta(x, B(A(x)))]$$

dove E è la distribuzione di x che ci si aspetta, e Δ è la funzione obbiettivo della ricostruzione, che misura la distanza tra l'output del decoder e l'input.

La figura 6 mostra il processo del modello autoencoder.

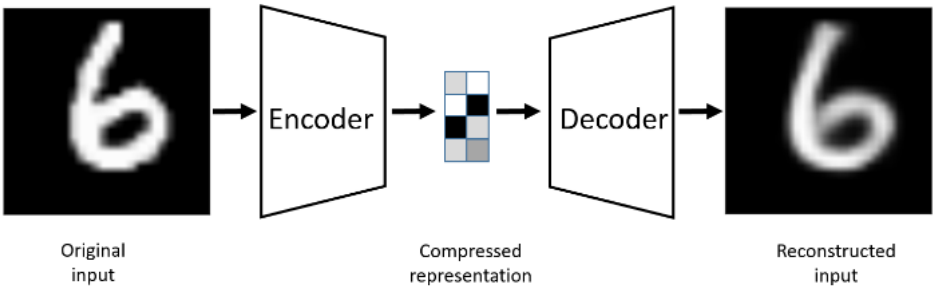

Figura 6: Schema che rappresenta il modello autoencoder. Da [20]

L'approccio utilizzato per Soft Brownian Offset è quello di addestrare un autoencoder sul dataset di partenza, riducendo lo spazio di lavoro utilizzando l'encoder, generare i dati OOD ed infine ricostruire il dataset attraverso il decoder. In questo modo il dataset è più semplice da elaborare.

4

GENERAZIONE DI ATTACCHI USANDO SOFT BROWNIAN OFFSET

La procedura adottata per questa tesi è la seguente:

- Preparazione dei dataset
- Generazione dei pacchetti Out of Distribution (OOD)
- Analisi dei dati ottenuti
- Addestramento del modello
- Valutazione del modello

Nel capitolo 5 verranno mostrati i risultati.

4.1 METODOLOGIE

Ogni dataset è stato analizzato, per capire quali fossero i dati più significativi per l'addestramento del modello.

È stato necessario filtrare alcune colonne che contenevano valori non numerici come, nel caso di CIC-IDS, la colonna "Timestamp" che includeva la data e l'ora del pacchetto.

Inoltre le righe contenenti valori come "Inf" o "NaN" sono state modificate perché, se lasciate intatte, avrebbero causato errori durante la generazione dei dati out-of-distribution.

Entrambi i dataset hanno una colonna "Label" per indicare la classificazione del pacchetto, nello specifico, sono presenti le varie tipologie di attacchi e.g. "Bruteforce", "Dos", "PingScan", "PortScan". Dato che, nel nostro caso, non ci interessa sapere nello specifico la catalogazione del pacchetto, tutti gli attacchi, sono stati etichettati come "attack".

Dopo aver fatto queste operazione di preprocessamento, si è passati alla generazione dei sample OOD.

Gen. \ Addes.	N+OOD	N+AV+OOD	N+AV
OOD_{CMP}	N+OOD_{CMP}	N+AV+OOD_{CMP}	/
OOD_N	N+OOD_N	N+AV+OOD_N	/
OOD_{AV}	N+OOD_{AV}	N+AV+OOD_{AV}	/

Tabella 1: Combinazioni di generazione dei pacchetti e addestramento del modello, dove N = pacchetti normali, AV = pacchetti di attacchi veri, CMP = pacchetti normali + pacchetti di attacchi, OOD = pacchetti generati. Si trova poi AV OOD, N OOD, N+AV OOD, rispettivamente dati generati a partire da attacchi, dati generati a partire da pacchetti normali, dati generati a partire dal dataset completo

In particolare possiamo distinguere vari casi sia, per la generazione dei pacchetti sia, per l'addestramento del modello. Abbiamo infatti, tre tipologie di generazione:

- Generazione a partire da soli pacchetti normali.
- Generazione a partire da soli pacchetti di attacchi.
- Generazione a partire sia da pacchetti normali che da pacchetti di attacchi (dataset completo).

Per l'addestramento abbiamo:

- Addestramento del modello con i pacchetti normali e i pacchetti OOD.
- Addestramento del modello con i pacchetti normali, i pacchetti OOD e i pacchetti di attacchi.
- Addestramento del modello con i pacchetti normali e i pacchetti di attacchi.

Le combinazioni di addestramento e generazione danno luogo ad esiti diversi. La tabella 1 riassume le varie combinazioni.

Un metodo possibile che non è stato approfondito è quello di generare i pacchetti filtrando quelli troppo "vicini", in termini di caratteristiche, ai pacchetti normali. Si ottiene così un dataset che ha la giusta quantità di dati out-of-distribution e potrebbe migliorare l'apprendimento di un modello.

La prima prova effettuata è stata quella di utilizzare Soft Brownian Offset a partire dal dataset completo senza distinzioni di classificazione di pacchetto,

per vedere se era possibile in questo modo, generare degli attacchi verosimili. Questa soluzione però genera dei pacchetti che sono troppo vicini, in termini di caratteristiche, a quelli normali perché i dataset hanno una quantità molto maggiore di dati normali rispetto a quelli di attacchi.

Un ulteriore metodo è quello di utilizzare solo i pacchetti normali per la generazione, questo rappresenta maggiormente uno scenario reale in quanto solitamente, i dati di attacco a disposizione sono pochi.

Evidenzieremo quindi le differenze che ci sono tra i vari approcci.

Per ogni metodo, la generazione tramite Soft Brownian Offset verrà eseguita quattro volte attraverso un ciclo "for" per testare i vari parametri, linearmente intervallati, di d_min (d^-) e softness (σ). Il valore di d_off (d) invece sarà calcolato come $d_{min} \cdot 0.7$ come da documentazione.

A seguito della generazione dei pacchetti, questi devono essere uniti al dataset iniziale e verranno etichettati come pacchetti di attacco.

Per avere un'idea più concreta della qualità della generazione, si procede con la visualizzazione grafica del nuovo dataset così creato, tramite l'algoritmo di riduzione di dimensionalità UMAP. Data la grande quantità di colonne che i dataset possiedono, la riduzione della dimensionalità dei dati è necessaria per permettere la loro rappresentazione su un piano in due dimensioni.

E' stato scelto UMAP, rispetto alla sua alternativa TSNE, per la velocità di esecuzione [21].

Per i grafici è stata utilizzata la libreria Plotly per permettere l'esplorazione dei dati salvati in formato HTML.

Dunque si procede all'addestramento di XGBoost con gli iperparametri impostati sui valori di fabbrica.

Infine si valuta il modello utilizzando la metrica "Metthew's correlation coefficient" (MCC) prendendo come dati di test il dataset iniziale (senza pacchetti OOD). Si è scelto di utilizzare questa metrica per il calcolo delle prestazioni del modello in quanto, rispetto all'accuracy test, MCC è efficace quando si ha un dataset sbilanciato [22] come nel nostro caso (abbiamo molti più pacchetti normali che pacchetti di attacchi).

Le metriche raccolte per ogni strategia di generazione sono state poi confrontate, per capire così quali sono gli effetti delle generazioni.

5

RISULTATI

Di seguito verranno mostrati i dati più significativi prodotti atraverso del codice Python. Sono state utilizzate diverse librerie, tra cui:

- Pandas e Numpy, per la manipolazione dei dataset
- Plotly, per la visualizzazione dei dati
- UMAP, per la riduzione della dimensionalità dei dataset
- SBO, per la generazione dei dati fuori dalla distribuzione
- XGBoost, il modello addestrato
- Scikit-learn, per la valutazione del modello e per il preprocessamento dei dati

Tutto il codice utilizzato si trova su Github [23] .

5.1 PREPARAZIONE DEI DATI

Come detto nel capitolo 4, i dati non possono essere utilizzati come sono, ma hanno bisogno di un preprocessamento. In particolare, sono stati eliminati i dati non numerici attraverso il metodo di Pandas "pandas.DataFrame.replace" come segue:

```
input_data.replace([np.inf, -np.inf], -1, inplace=True)
input_data.replace(np.nan, -1, inplace=True)
```

Si estrae poi la colonna label dal dataset, che dovrà essere utilizzata successivamente per il modello.

Si procede a rinominare le etichette dei vari attacchi in "attack":

```
attacks_packets_types =['Bot', 'DDOS attack-HOIC', 'DDOS
                                    attack-LOIC-UDP',
                        'DoS attacks-Hulk', 'DoS attacks-SlowHTTPTest',
                        'FTP-BruteForce',
                        'Infilteration', 'SSH-Bruteforce']
attacks_packets.replace(attacks_packets_types, 'attack', inplace=True)
```

Si estraggono poi i pacchetti normali e quelli di attacco:

```
normal_packets =input_data[input_data['label'] =="normal"]
attack_packets =input_data[input_data['label'] !="normal"]
```

5.2 GENERAZIONE DEI PACCHETTI

Si è generato i nuovi pacchetti attraverso la libreria Python SBO fornita da [6]. Il metodo "soft_brownian_offset" della libreria richiede:

- i dati da cui generare i nuovi pacchetti
- d_min (d^-)
- d_off (d)
- softness (σ)
- numero di pacchetti da generare

```
data_ood =soft_brownian_offset(data_i, d_min_, d_off_,
                               softness=softness_,
                              n_samples=n_ood_samples)
```

Questo metodo ritorna i dati generati all'interno di una struttura dati numpy.

Questi dati sintetici vengono uniti al dataset iniziale ed etichettati come "attack" nel modo seguente:

```
#Dati
np.concatenate((normal_data.drop(columns=['label']),
                                    attacks_data.drop(columns=['label']),
                                    ood_data))

#Etichette
np.concatenate((normal_data.label, attacks_data.label, ['attack' for x
                                    in range(len(ood_data))]))
```

5.3 RAPPRESENTAZIONE GRAFICA DEI DATI

In questa sezione esploreremo i dati generati da Soft Brownian Offset graficamente utilizzando la libreria UMAP [21] per ridurre la dimensionalità.

I test sono stati eseguiti con i seguenti parametri:

```
umap_2d =UMAP(n_components=2, init='random', random_state=0,
              n_neighbors=70, min_dist=.8)
```

che risultano avere il giusto compromesso tra velocità e accuratezza.

Tramite la rappresentazione grafica dei dati è possibile notare in maniera se i pacchetti generati sono più simili a quelli di attacco oppure a quelli normali. L'obbiettivo è quello di generare dati che siano graficamente vicini ai pacchetti di attacco ma lontani da quelli normali, questo infatti dovrebbe tradursi, in un addestramento del modello migliore. Le tecniche di generazioni più efficaci quindi, saranno quelle che riescono a generare pacchetti non troppo distanti da quelli di attacco.

I dati inoltre sono stati generati attraverso varie impostazioni di SBO manipolando i parametri d^{-}, d^{off} e σ.

Analizziamo quindi gli approcci citati nel capitolo 4.

5.3.1 *Generazione a partire dai pacchetti normali*

Questo metodo di generazione dei pacchetti, in uno scenario reale, è quello che potrebbe essere utilizzato maggiormente, in quanto solitamente non si hanno a disposizione tanti dati su attacchi informatici all'interno della propria rete, ma solo dati di traffico comune. In figura 7 si nota, come questo metodo non sia molto efficace infatti, ci sono delle evidenti sovrapposizioni tra pacchetti normali e pacchetti generati mentre, in corrispondenza degli attacchi, questo non accade. Le figure successive 9 e 10 sono invece i test eseguiti su CICIDS che ottengono i medesimi risultati.

E' evidente che i pacchetti generati, siano troppo simili ai pacchetti normali e troppo dissimili da quelli di attacco. Questa valutazione qualitativa potrebbe già darci un'idea sull'efficacia di addestramento del modello.

5.3.2 *Generazione a partire dai pacchetti attacco*

Questa è la modalità opposta rispetto a quella precedentemente citata, dove si generano i dati sintetici a partire dai pacchetti di attacco. Come vediamo in figura 11, in questo caso i pacchetti OOD, sono separati dai dati normali

Figura 7: (Dataset ADFANET - Generazione da normali) In giallo i pacchetti di attacco, in viola i pacchetti generati ed in verde i pacchetti normali. Questi grafici sono stati prodotti a partire da una porzione di dati ristretta per permettere una migliore visualizzazione.

Figura 8: (Dataset ADFANET - Generazione da normali) Questa figura presenta i dati della figura 7 ingranditi per permettere una migliore visualizzazione.

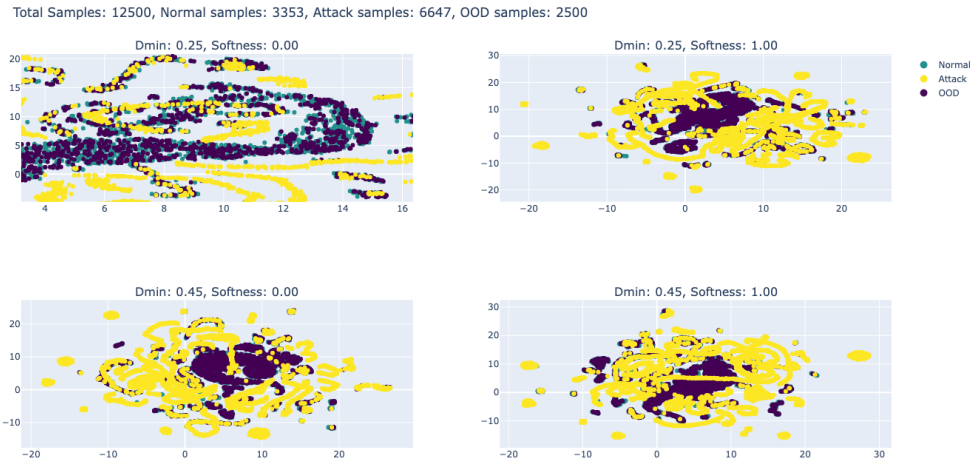

Figura 9: (Dataset CICIDS - Generazione da normali) Si nota la maggiore complessita di questo dataset, ma si presentano gli stessi problemi di ADFANET.

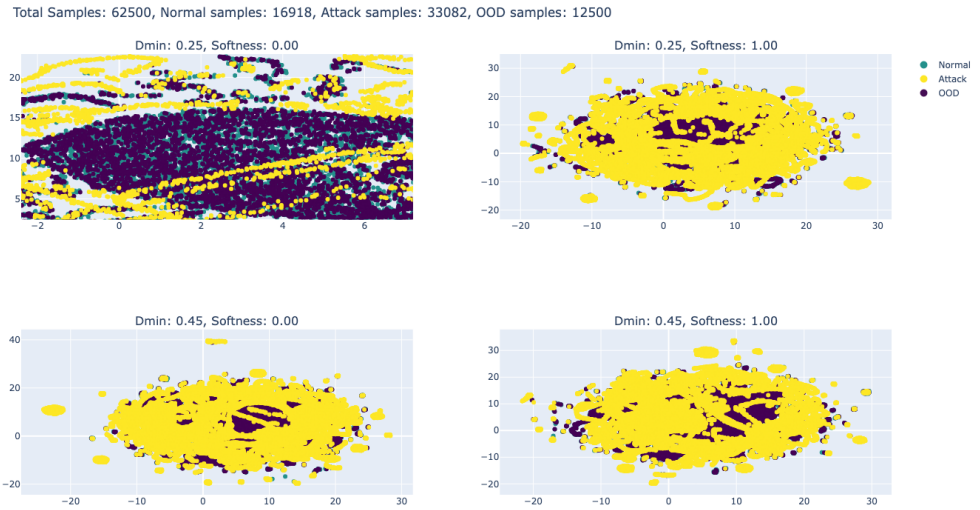

Figura 10: (Dataset CICIDS - Generazione da normali) Questa immagine contiene grafici con più di 62 000 pacchetti. Nel grafico in alto a sinistra si nota come i pacchetti generati siano sovrapposti a quelli normali.

ma vicini a quelli di attacco. Questa, osservando i grafici, sembra essere una buona soluzione di generazione.

Figura 11: (Dataset CICIDS - Generazione da attacchi) Questa immagine contiene grafici con più di 62 000 pacchetti. Nel grafico in alto a sinistra si nota come i pacchetti generati siano sovrapposti a quelli di attacco.

5.4 ADDESTRAMENTO DEL MODELLO

Per addestrare il modello XGBoost si è utilizzata la libreria annessa con i suoi parametri di fabbrica. Dopo il training, ogni modello è stato salvato in formato JSON per poi essere riutilizzato:

```
model =xgb.XGBClassifier()
# Addestra il modello
model.fit(X_train, y_train)

# Salva il moddello
model.save_model(model_path)
```

La parte più onerosa, in termini computazionali, non è stata l'addestramento del modello bensì la generazione dei dati sintetici che, nel caso di CICIDS, ha impiegato diverse ore.

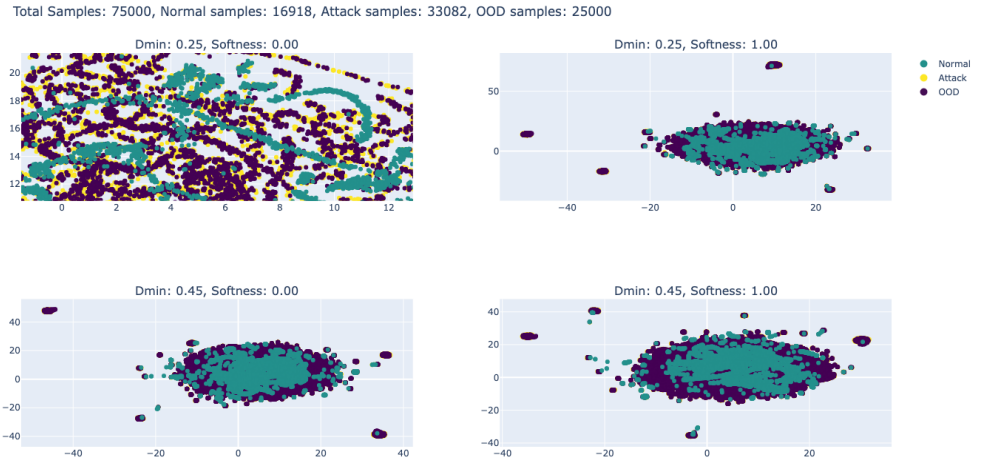

Figura 12: (Dataset CICIDS - Generazione da attacchi) Questa immagine contiene grafici con più di 75 000 pacchetti. Nel grafico in alto a sinistra si nota come i pacchetti generati siano sovrapposti a quelli di attacco. Ma data la grande complessità del dataset, i pacchetti sono in generale tutti molto vicini.

5.5 CALCOLO DELLE METRICHE

Per il calcolo delle metriche si è utilizzato il metodo fornito da SCIkit-learn per il Matthews Correlation Coefficient facendo classificare i pacchetti al modello e confrontando i risultati con le etichette effettive del dataset. In questo modo si può osservare quanto il modello sia efficace per il rilevamento dei tipi di pacchetti.

Abbiamo quindi raccolto le metriche sul modello addestrato con varie quantità di dati di partenza e di dati sintetici. Ogni modello poi è stato testato per dieci volte ed è stata fatta la media della metrica MCC. Nelle tablelle 2 e 3 sono mostrati i migliori risultati ottenuti per ogni metodo utilizzato riprendendo 1. Come si può osservare in generale, il moddello addestrato insieme ai dati sintetici ottiene risultati peggiori rispetto al modello senza dati aggiuntivi. Bisogna considerare inoltre, che la generazione dei dati OOD impiega molto tempo anche su dataset non troppo complessi come ADFANET, quindi i miglioramenti non solo sono relativamente bassi ma in molte circostanze risultano essere deterioranti.

La lista di dati completi può essere consultata nell'appendice **??**.

Gen. \ Addes.	N+OOD	N+AV+OOD	N+AV
OOD_{CMP}	34.52%	99.854%	99.842%
OOD_{N}	33.37%	99.839%	99.842%
OOD_{AV}	44.04%	99.865%	99.842%

Tabella 2: Risultati migliori ottenuti per ogni metodo sul dataset ADFANET

Gen. \ Addes.	N+OOD	N+AV+OOD	N+AV
OOD_{CMP}	−12.06%	92.493%	93.596%
OOD_{N}	−11.53%	92.772%	93.596%
OOD_{AV}	−13.66%	93.428%	93.596%

Tabella 3: Risultati migliori ottenuti per ogni metodo sul dataset CICIDS

6

CONCLUSIONI

BIBLIOGRAFIA

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, p. 20, July 2019. (Citato nelle pagine 3, 9, 10, and 11.)

[2] Babcock University, O. F.Y, A. J.E.T, A. O, H. J. O, O. O, and A. J, "Supervised Machine Learning Algorithms: Classification and Comparison," *International Journal of Computer Trends and Technology*, vol. 48, pp. 128–138, June 2017. (Citato nelle pagine 3 and 13.)

[3] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Nature Biotechnology*, vol. 26, pp. 1011–1013, Sept. 2008. (Citato nelle pagine 3 and 15.)

[4] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, Jan. 2021. (Citato a pagina 7.)

[5] S. S. Gopalan, D. Ravikumar, D. Linekar, A. Raza, and M. Hasib, "Balancing Approaches towards ML for IDS: A Survey for the CSE-CIC IDS Dataset," in *2020 International Conference on Communications, Signal Processing, and Their Applications (ICCSPA)*, (Sharjah, United Arab Emirates), pp. 1–6, IEEE, Mar. 2021. (Citato a pagina 7.)

[6] F. Moller, D. Botache, D. Huseljic, F. Heidecker, M. Bieshaar, and B. Sick, "Out-of-distribution Detection and Generation using Soft Brownian Offset Sampling and Autoencoders," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (Nashville, TN, USA), pp. 46–55, IEEE, June 2021. (Citato nelle pagine 7, 17, and 26.)

[7] E. Biermann, E. Cloete, and L. Venter, "A comparison of Intrusion Detection systems," *Computers & Security*, vol. 20, pp. 676–683, Dec. 2001. (Citato a pagina 9.)

[8] A. S. Ashoor and S. Gore, "Importance of Intrusion Detection System," vol. 2, no. 1, 2010. (Citato a pagina 9.)

[9] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, pp. 16–24, Jan. 2013. (Citato a pagina 9.)

[10] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Unsupervised Algorithms to Detect Zero-Day Attacks: Strategy and Application," *IEEE Access*, vol. 9, pp. 90603–90615, 2021. (Citato nelle pagine 10 and 14.)

[11] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. Khan, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review," *Procedia Computer Science*, vol. 171, pp. 1251–1260, 2020. (Citato a pagina 12.)

[12] F. Hoffmann, T. Bertram, R. Mikut, M. Reischl, and O. Nelles, "Benchmarking in classification and regression," *WIREs Data Mining and Knowledge Discovery*, vol. 9, Sept. 2019. (Citato a pagina 14.)

[13] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela, "Practical Lessons from Predicting Clicks on Ads at Facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, (New York NY USA), pp. 1–9, ACM, Aug. 2014. (Citato a pagina 15.)

[14] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (San Francisco California USA), pp. 785–794, ACM, Aug. 2016. (Citato a pagina 15.)

[15] S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," *Machine Learning*, vol. 16, pp. 235–240, Sept. 1994. (Citato a pagina 16.)

[16] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, (Funchal, Madeira, Portugal), pp. 108–116, SCITEPRESS - Science and Technology Publications, 2018. (Citato a pagina 16.)

[17] M. Yuhas, Y. Feng, D. J. X. Ng, Z. Rahiminasab, and A. Easwaran, "Embedded out-of-distribution detection on an autonomous robot platform," in *Proceedings of the Workshop on Design Automation for CPS and IoT*, (Nashville Tennessee), pp. 13–18, ACM, May 2021. (Citato a pagina 17.)

[18] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," July 2016. Comment: 29 pages. (Citato a pagina 17.)

[19] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," Apr. 2021. Comment: Book chapter. (Citato a pagina 17.)

[20] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," (Citato nelle pagine 3 and 20.)

[21] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," Sept. 2020. Comment: Reference implementation available at http://github.com/lmcinnes/umap. (Citato nelle pagine 23 and 27.)

[22] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, p. 6, Dec. 2020. (Citato a pagina 23.)

[23] G. Bartelloni, "APPLICATION OF SOFT-BROWNIAN-OFFSET TO GENERATE CYBER-ATTACKS TO TRAIN INTRUSION DETECTORS," Apr. 2023. (Citato a pagina 25.)