Cybersecurity Project Documentation

Daniele Russo, Nicola Modugno

November 2025

Contents

1 Introduzione

1.1 Contesto e motivazione

L'industria della sicurezza degli endpoint ha adottato in misura crescente strumenti basati sull'apprendimento automatico (ML) come componenti integrali delle proprie strategie di difesa. In particolare, i classificatori che utilizzano feature derivate dall'analisi statica dei binari sono comunemente impiegati per la rilevazione e la prevenzione rapida prima dell'esecuzione (pre-execution detection), fungendo spesso da prima linea di difesa per gli utenti finali. Tali sistemi, tuttavia, sono esposti ad attacchi avversari. Storicamente, l'attenzione principale si è concentrata sugli attacchi di evasione, in cui l'avversario mira ad alterare il punto dati in fase di inference per indurre una classificazione errata. Tuttavia, in questo contesto di sicurezza, un problema insidioso è rappresentato dagli attacchi di poisoning, che tentano di influenzare il processo di training del modello ML, e in particolare dagli attacchi di backdoor poisoning. Il pipeline di training di molti vendor di sicurezza presenta un punto di iniezione naturale per tali attacchi, poiché le aziende si affidano spesso a feed di minacce crowdsourced per ottenere un flusso ampio e diversificato di binari etichettati (sia goodware che malware) necessari per raggiungere prestazioni di rilevamento soddisfacenti. In un attacco backdoor, l'avversario introduce un modello scelto con cura (trigger o watermark) nello spazio delle feature affinché il modello vittima impari ad associare la sua presenza a una classe scelta dall'attaccante. Gli attacchi di backdoor possono essere resi particolarmente difficili da rilevare quando rientrano nella categoria dei "clean-label backdoor attacks", dove l'avversario inietta campioni benigni con la backdoor senza manipolare l'etichetta originale assegnata dai sistemi di analisi di terze parti. Il presente progetto si propone di affrontare la difficoltà di difesa contro tali attacchi furtivi, esplorando l'uso di tecniche di scarsità (sparsity) per tentare di rilevare i campioni avvelenati nel dataset.

1.2 Obiettivi del progetto

L'obiettivo primario di questo progetto è utilizzare tecniche di scarsità e iniezione di rumore per sviluppare un rilevatore di dataset poisoning. Per raggiungere questo scopo, il progetto prevede le seguenti fasi operative

1. Reperire o costruire un dataset avvelenato di malware. Questo dataset dovrà contenere campioni backdoored che simulino attacchi di poisoning contro classificatori malware (come quelli studiati nel contesto di Windows PE, PDF o applicazioni Android).

2. Addestrare una rete neurale come classificatore di malware (detector) utilizzando il dataset avvelenato.

3. Introdurre rumore nei pesi interni della rete neurale o sparsificare la rete (ad esempio, utilizzando tecniche di pruning che possono generare sottoreti sparse, un concetto centrale nell'ipotesi del "biglietto della lotteria"

o LTH). Le tecniche di pruning possono ridurre il numero di parametri mantenendo l'accuratezza, rendendo l'inferenza più efficiente.

4. Verificare una correlazione tra il risultato della classificazione ottenuto dopo l'aggiunta di rumore e la natura avvelenata dei campioni.

1.3 Ipotesi di lavoro: resilienza dei campioni avvelenati alla misclassificazione

L'ipotesi centrale che questo progetto intende verificare è che i campioni avvelenati (o backdoored) possiedano una resilienza alla misclassificazione quando viene introdotto rumore nella rete, e che l'introduzione di rumore o sparsità possa quindi aiutare a identificare i campioni avversari.
Questa ipotesi si fonda sull'idea che le porzioni di rete che codificano l'effetto della backdoor (il trigger) possano comportarsi in modo differente rispetto alle porzioni che codificano le feature legittime. Le tecniche di scarsità (sparsity) implicano l'eliminazione di pesi non necessari dalle reti neurali (pruning). L'ipotesi del biglietto della lotteria (LTH) suggerisce che le reti neurali dense e inizializzate casualmente contengono sottoreti sparse ("winning tickets") che, se addestrate isolatamente, possono raggiungere un'accuratezza paragonabile alla rete originale. È stato osservato che l'inizializzazione dei pesi è cruciale per l'efficacia di una sottorete sparsa. In effetti, anche se l'aggiunta di rumore gaussiano all'inizializzazione di un winning ticket tende a ridurne l'accuratezza, i winning tickets mostrano una sorprendente robustezza al rumore (in particolare a bassi livelli) e continuano a sovraperformare i modelli ri-inizializzati casualmente. Se i campioni avvelenati sfruttano un percorso specifico o una configurazione di pesi (analoga, per la sua unicità, a un winning ticket o a un "percorso furtivo") che è stato stabilito durante il poisoning, la loro risposta alla manipolazione dei pesi (rumore o sparsity) potrebbe essere distintiva. In particolare, se l'effetto della backdoor è codificato in pesi che rimangono insolitamente stabili o che sono particolarmente importanti (ad esempio, pesi che si muovono più lontano durante il training, come evidenziato in uno studio sugli LTH), l'iniezione di rumore potrebbe non alterare la loro classificazione attesa (la misclassificazione intenzionale), rivelando così la loro natura avversaria. L'uso di rumore (noise injection) è stato storicamente esplorato anche nel contesto degli attacchi avversari, dove l'avvelenamento deliberato del dataset con rumore ben congegnato mirava a impedire la generazione di firme utili (come nel caso del sistema Polygraph). Questo progetto inverte la logica, usando il rumore non per attaccare, ma per diagnosticare e rilevare la presenza di attacchi di poisoning già esistenti.

1.4 Osservazioni: Implicazioni della "Lottery Ticket Hypothesis" sulla Resilienza del Backdoor

L'ipotesi operativa del progetto sostiene che i campioni avvelenati siano resilienti agli errori di misclassificazione e che l'introduzione di rumore nella rete (o

la sua sparsificazione) possa rivelare i campioni avversari attraverso una correlazione distintiva nei risultati di classificazione. Tuttavia, i risultati empirici derivanti dalla ricerca sulla scarsità nelle reti neurali (come la Lottery Ticket Hypothesis - LTH) suggeriscono che i meccanismi di apprendimento che portano a rappresentazioni efficaci possono essere intrinsecamente resistenti alle perturbazioni, un fattore che potrebbe complicare la strategia di rilevamento basata sull'iniezione di rumore.

1.4.1 Resilienza delle Sottoreti Sparse al Rumore

La Lottery Ticket Hypothesis postula che le reti neurali dense e inizializzate in modo casuale contengano delle sottoreti sparse, definite "biglietti vincenti" (winning tickets), che, se addestrate isolatamente, possono raggiungere un'accuratezza paragonabile alla rete originale. L'importanza dell'inizializzazione originale è cruciale per il successo di questi winning tickets, poiché le sottoreti sparse ri-inizializzate casualmente non riescono a eguagliare le prestazioni della rete originale. Un'indagine specifica sulla robustezza di questi winning tickets rivela che l'aggiunta di rumore gaussiano alle loro inizializzazioni (un test di robustezza correlato all'introduzione di rumore nei pesi) riduce l'accuratezza e rallenta l'apprendimento. Ciononostante, i winning tickets mostrano una sorprendente robustezza al rumore. Ad esempio, anche dopo l'aggiunta di rumore con una deviazione standard pari a 3σ, i winning tickets continuano a superare le prestazioni ottenute con una ri-inizializzazione casuale.

1.4.2 Potenziale Implicazione per il Rilevamento del Poisoning

Se si assume che l'effetto backdoor (la mappatura tra il trigger e la classificazione errata come "benigno") sia codificato in una sottorete particolarmente ottimizzata che ha "vinto la lotteria dell'inizializzazione" per quel compito specifico di misclassificazione, ne consegue che questa porzione di rete (il backdoor) potrebbe essere robusta per natura. Un attacco di backdoor poisoning efficace mira a creare un'area di densità nello spazio delle feature affinché il classificatore adatti il suo confine decisionale. Strategie di attacco più furtive, come la "Greedy Combined Selection", cercano di far mimetizzare il backdoor nelle aree ad alta densità di campioni benigni legittimi, rendendo il trigger semanticamente coerente e difficile da rilevare tramite tecniche di clustering o anomalia. Considerando la robustezza al rumore dei winning tickets, è plausibile che il subnetwork che implementa l'associazione trigger→benigno non venga significativamente perturbato dall'aggiunta di rumore o dalla sparsificazione della rete. Proiezione dell'Esito (Ipotesi Critica): Se la robustezza del backdoor è analoga alla robustezza dei winning tickets al rumore, la conseguenza diretta per il progetto potrebbe essere che la classificazione errata (misclassificazione dei campioni avvelenati come benigni) persisterà anche dopo l'introduzione del rumore o della sparsità. Questo fenomeno indebolirebbe la capacità di stabilire una chiara correlazione tra la variazione della classificazione indotta dal rumore e la presenza del campione avvelenato. In sintesi, l'esito del progetto potrebbe non es-

sere l'identificazione di una rottura nella classificazione dei campioni avvelenati, ma al contrario, la conferma della loro stabilità di classificazione (resilienza), suggerendo che la loro robustezza potrebbe derivare da un meccanismo di apprendimento ottimale, seppur avversario.

2 Stato dell'arte

2.1 Data Poisoning e Backdoor Attacks nei Modelli ML

I sistemi di sicurezza degli endpoint utilizzano sempre più strumenti basati sull'apprendimento automatico (ML), in particolare classificatori che impiegano feature derivate dall'analisi statica dei binari, per la rilevazione e la prevenzione rapida prima dell'esecuzione. Tuttavia, questi modelli sono esposti a vulnerabilità legate agli attacchi avversari. Gli attacchi avversari contro i modelli ML si dividono in due categorie principali: attacchi di evasione e attacchi di poisoning. Gli attacchi di evasione mirano a perturbare un campione in fase di testing per indurre una misclassificazione. Gli attacchi di poisoning, invece, tentano di manipolare i dati di training, iniettando nuovi dati o modificando quelli esistenti, al fine di influenzare il processo di addestramento e causare misclassificazioni al momento dell'inferenza. Il contesto della sicurezza informatica offre un punto di iniezione naturale per gli attacchi di poisoning, poiché molte aziende si affidano a feed di minacce provenienti da fonti esterne (crowdsourced) per ottenere la grande quantità di dati etichettati necessaria per l'addestramento.

2.1.1 Attacchi flip-label, clean-label e backdoor feature-based

Gli attacchi di poisoning si suddividono ulteriormente in base al loro obiettivo, inclusi gli attacchi di availability (che degradano l'accuratezza generale), gli attacchi targeted (che causano la misclassificazione di una singola istanza) e gli attacchi di backdoor. In un attacco di backdoor, l'avversario introduce un modello o schema scelto con cura (trigger o watermark) nello spazio delle feature. Il modello vittima impara ad associare la presenza di questo pattern a una classe scelta dall'attaccante. Quando lo stesso watermark viene iniettato in un campione di test, esso ne forza la classificazione desiderata. Questo approccio, originariamente introdotto per le reti neurali nel riconoscimento di immagini, offre agli aggressori una capacità di evasione generica. I tipi di attacco di backdoor includono:

- Attacchi Flip-label (o con etichetta manipolata): L'avversario manipola l'etichetta del campione avvelenato. Nel contesto della sicurezza, questo approccio è spesso impraticabile poiché le etichette per i dati crowdsourced sono generate da più motori antivirus di terze parti, che l'attaccante non può controllare direttamente.
- Attacchi Clean-label: Questi attacchi avanzati e furtivi aggirano l'ostacolo non manipolando l'etichetta originale dei dati di poisoning. L'attaccante inietta campioni benigni con la backdoor nel training set, pur mantenendo l'etichetta benigna. L'obiettivo è alterare il processo di training in modo che il classificatore associ la backdoor a una classificazione benigna.
- Attacchi Feature-based: Il progetto si concentra su attacchi contro classificatori che utilizzano feature statiche estratte dai binari (come Windows PE, PDF o applicazioni Android). Le metodologie di backdoor possono

essere rese model-agnostic (indipendenti dal modello) sfruttando tecniche di Explainable Machine Learning (come SHAP) per guidare la selezione delle feature più efficaci per creare il watermark. L'attacco di tipo Greedy Combined Selection, in particolare, è risultato essere il più furtivo, poiché genera backdoor in regioni ad alta densità di campioni legittimi, rendendole difficili da rilevare.

2.1.2 Rumore intenzionale e inquinamento del dataset

Il concetto di inquinamento intenzionale dei dati per eludere i sistemi di rilevamento non è nuovo e precede gli attacchi backdoor contro i moderni sistemi ML. Già in studi precedenti sugli worm signature generators (come Polygraph), è stato dimostrato che l'iniezione deliberata di rumore (deliberate noise injection) può impedire in modo efficace la generazione di firme utili. Polygraph, un generatore di firme basato sulla sintassi, era stato progettato per resistere a livelli di rumore "normale" fino all'80% nel pool di flussi sospetti. Tuttavia, gli aggressori possono utilizzare worm polimorfici ingannevoli per creare e inviare flussi anomali falsi (fake anomalous flows) al fine di inquinare intenzionalmente il dataset utilizzato per l'estrazione delle firme. Quando il rumore è ben congegnato, un livello molto inferiore, ad esempio solo il 50% di rumore (corrispondente all'invio di un flusso anomalo falso per ogni variante di worm reale), è sufficiente a impedire al sistema di generare firme affidabili. L'attacco si basa sulla creazione di flussi anomali falsi che contengono invarianti falsi (FI) anziché i veri invarianti del worm (TI). Una tecnica specifica, l'iniezione di score multiplier strings, è usata per contrastare le Bayes signatures: queste stringhe vengono scelte per avere una probabilità di occorrenza non trascurabile nel traffico innocuo, ma sufficiente per ottenere un punteggio alto nel modello. L'attaccante frammenta queste stringhe e le inietta, ottenendo un effetto moltiplicatore sullo score che rende impossibile impostare una soglia di rilevamento (threshold) che garantisca contemporaneamente un basso tasso di falsi positivi (FP) e un basso tasso di falsi negativi (FN).

2.2 Tecniche di rilevazione e mitigazione

La difesa contro gli attacchi backdoor, in particolare nel contesto model-agnostic e clean-label, è intrinsecamente difficile. Molte delle attuali contromisure sono focalizzate su reti neurali profonde per la visione artificiale e assumono che gli attaccanti manipolino le etichette. Le strategie di mitigazione valutate includono:

1. Spectral Signatures: Questo metodo, adattato per operare su uno spazio di feature ridotto, si basa sulla decomposizione SVD (Singular Value Decomposition) per rilevare due sottopopolazioni ϵ - separabili spettralmente. I campioni con i punteggi di outlier più alti vengono filtrati.

2. HDBSCAN (Hierarchical Density-Based Clustering): Ipotizzando che i campioni watermarked formino un subspace ad alta densità, questo approccio di clustering densità-basato può identificare i cluster anomali.

3. Isolation Forest: Un algoritmo di rilevamento anomalie non supervisionato che identifica punti rari e diversi. Si è osservato che questa tecnica è efficace nell'isolare i punti backdoored, ma solo se addestrata su un dataset trasformato (ridotto alle feature più importanti). Se applicato allo spazio delle feature originale, l'algoritmo rileva solo una piccola frazione dei punti avvelenati, confermando che le sottopopolazioni non sono sufficientemente separabili nello spazio originale.

È stato empiricamente dimostrato che gli attacchi generati tramite la strategia Combined (che si mimetizzano in regioni ad alta densità di campioni legittimi) sono molto più furtivi rispetto agli attacchi Independent e sono difficili da isolare con queste tecniche di mitigazione. La difficoltà nella difesa deriva dalla combinazione della bassa separabilità delle sottopopolazioni indotta dagli attacchi clean-label e della difficoltà di distinguere le regioni dense generate dall'attacco dalle regioni dense che si verificano naturalmente nei dataset benigni diversi.

2.3 Sparsità e robustezza dei modelli neurali

Le tecniche di sparsità (sparsity) si concentrano sull'eliminazione dei pesi non necessari dalle reti neurali (pruning), riducendo il conteggio dei parametri di oltre il 90% senza compromettere l'accuratezza, migliorando così l'efficienza dell'inferenza. Il concetto chiave in questo ambito è l'Ipotesi del Biglietto della Lotteria (Lottery Ticket Hypothesis - LTH), la quale afferma che le reti neurali dense e inizializzate in modo casuale contengono sottoreti sparse (winning tickets) che, se addestrate isolatamente dalla loro inizializzazione originale, raggiungono un'accuratezza di test paragonabile alla rete originale in un numero simile di iterazioni. Punti salienti sulla LTH:

- Importanza dell'Inizializzazione: I winning tickets hanno "vinto la lotteria dell'inizializzazione". La loro efficacia dipende in modo cruciale dai loro valori di peso iniziali; se vengono re-inizializzati casualmente, le prestazioni decadono rapidamente.
- Identificazione: I winning tickets vengono identificati addestrando la rete, potando i pesi di minore entità (pruning) e quindi ripristinando i pesi rimanenti ai loro valori iniziali originali. L'approccio di pruning iterativo si è dimostrato più efficace nel trovare winning tickets più piccoli rispetto al pruning one-shot.
- Generalizzazione e Velocità di Apprendimento: I winning tickets imparano spesso più velocemente della rete originale e raggiungono una migliore accuratezza di test (generalizzano meglio).

Il concetto di robustezza, cruciale per il progetto, si interseca con la LTH nel modo seguente:

- Robustezza al Rumore: È stato studiato l'effetto dell'aggiunta di rumore gaussiano all'inizializzazione dei winning tickets. Nonostante il rumore riduca l'accuratezza e rallenti l'apprendimento, i winning tickets dimostrano una sorprendente robustezza al rumore; anche con l'aggiunta di rumore elevato (fino a 3 σ), mantengono prestazioni superiori rispetto alle reti re-inizializzate casualmente.

Questa robustezza intrinseca delle sottoreti sparse al rumore offre una potenziale analogia per la resilienza dei campioni avvelenati. Se il meccanismo della backdoor sfrutta un percorso di peso ottimizzato (un "biglietto vincente" avversario), questa struttura potrebbe essere robusta alle perturbazioni introdotte (come rumore o sparsificazione), portando alla persistenza della classificazione errata (la misclassificazione come benigno).

3 Metodologia

3.1 Costruzione o scelta del dataset

3.1.1 Dataset di malware avvelenato (es. MalwareBackdoors)

Per la sperimentazione è stato utilizzato il dataset **EMBER** (*Endgame Malware BEnchmark for Research*), disponibile nelle versioni 2017 e 2018, ciascuna di dimensione approssimativa pari a 10 GB. Tale dataset è stato scelto in quanto rappresenta uno standard de facto per la ricerca nel campo del malware detection basato su machine learning, fornendo feature estratte staticamente da eseguibili Windows PE.

Il dataset contiene oltre un milione di campioni etichettati come *benigni* o *malware*, e per ragioni computazionali è stato selezionato un sottoinsieme rappresentativo. A partire dal dataset pulito, è stato introdotto un **avvelenamento controllato** mediante un attacco di tipo *Label Flipping*, consistente nella modifica del 20% delle etichette di campioni malware verso la classe *benigna*, con l'obiettivo di simulare un'alterazione realistica dei dati di training come descritto in [1].

3.1.2 Preprocessing e feature extraction

Il preprocessing è stato eseguito utilizzando la libreria ufficiale `ember`, che consente l'estrazione e normalizzazione delle 2851 feature statiche disponibili. Sono stati condotti esperimenti con tre diverse configurazioni di feature:

- **Top-17 feature** selezionate mediante mutua informazione (MI) e correlazione inferiore a 0.98;
- **Top-40 feature** per un bilanciamento tra complessità e capacità discriminativa;
- **Tutte le 2851 feature** per valutare le performance complete del classificatore.

Questa strategia ha permesso di analizzare la sensibilità del modello al numero di feature e la robustezza in condizioni di ridotta dimensionalità, replicando il principio dell'*explanation-guided feature selection* proposto da Severi et al. [1].

3.2 Architettura del classificatore di malware

3.2.1 Reti neurali utilizzate

Il classificatore implementato è una **rete neurale feed-forward fully connected**, composta da tre layer nascosti di dimensione 4000, 2000 e 100 neuroni, seguiti da un layer di output sigmoide. Ogni layer è seguito da un blocco di *Batch Normalization* e da una funzione di attivazione ReLU. È stato applicato

un tasso di dropout pari a 0.5 per ridurre l'overfitting, e una penalizzazione L2 con fattore di weight decay pari a $1e^{-5}$.

Il modello è stato addestrato per 10 epoche con una dimensione di batch di 256, utilizzando l'ottimizzatore Adam con learning rate iniziale di 0.001. La scelta di questa architettura è motivata dal compromesso tra capacità di rappresentazione e sostenibilità computazionale su macchine locali.

3.2.2 Metriche di valutazione

Per valutare le prestazioni del modello nelle tre condizioni sperimentali (*clean*, *poisoned*, *noisy*) sono state utilizzate le seguenti metriche:

- **Accuracy**, **Precision**, **Recall**, **F1-Score**, e **AUC-ROC**;
- Analisi di **specificità** e **sensibilità** per valutare la robustezza rispetto a falsi positivi e falsi negativi;
- Ottimizzazione della soglia di decisione tramite massimizzazione del valore F1.

I risultati principali per la configurazione con 17 feature mostrano:

- Dataset pulito: Accuracy = 0.7659, F1 = 0.7712, AUC = 0.778;
- Dataset avvelenato: Accuracy = 0.7617, F1 = 0.7849, AUC = 0.773;
- Dataset rumoroso: Accuracy = 0.5, AUC = 0.741.

La leggera variazione dei valori tra training pulito e avvelenato dimostra la **resilienza parziale del modello all'avvelenamento**, mentre l'introduzione di rumore sui pesi porta a un degrado evidente delle performance.

3.3 Introduzione di rumore e sparsificazione

3.3.1 Perturbazione dei pesi interni

Per valutare la robustezza interna della rete, è stato introdotto un **rumore gaussiano additivo** con deviazione standard $\sigma = 0.0001$ su tutti i layer densi e di normalizzazione. L'analisi statistica ha mostrato variazioni trascurabili della media e deviazione standard dei pesi (ordine di 10^{-7}), confermando che la perturbazione agisce come *micro-noise* controllato e non distruttivo. Tuttavia, tale perturbazione ha generato un drastico calo prestazionale nel test finale, indicando una fragilità strutturale del modello.

3.3.2 Applicazione di tecniche di pruning (Lottery Ticket Hypothesis)

Successivamente è stata esplorata la possibilità di applicare **pruning strutturale** ispirato alla *Lottery Ticket Hypothesis* di Frankle e Carbin [2]. L'obiettivo

è identificare sotto-reti (*winning tickets*) capaci di mantenere la stessa accuratezza del modello completo, ma con una significativa riduzione del numero di parametri.

Il pruning iterativo è stato condotto rimuovendo progressivamente i pesi a minor magnitudine, e re-inizializzando la rete ai pesi originali pre-training. Questa strategia permette di isolare subnet capaci di apprendere efficacemente, anche in presenza di rumore o avvelenamento parziale dei dati, aumentando la generalizzazione del modello.

3.4 Ipotesi sperimentale

3.4.1 Campioni avvelenati come outlier resilienti al rumore

L'ipotesi alla base dell'esperimento è che i **campioni avvelenati** introdotti durante il training possano comportarsi come *outlier resilienti al rumore*, mantenendo un'influenza rilevante sul processo di apprendimento anche dopo l'iniezione di rumore o pruning.

Tale fenomeno è coerente con quanto osservato da Perdisci et al. [3], secondo cui l'iniezione deliberata di "rumore informativo" può impedire ai sistemi di generazione automatica di firme di distinguere pattern genuini da quelli manipolati. Nel contesto di classificazione di malware, i campioni avvelenati si comportano come falsi negativi strutturali, in grado di distorcere la decision boundary e rendere inefficace la resilienza del modello a perturbazioni gaussiane o sparsificazioni.

4 Esperimenti

4.1 Setup sperimentale

4.1.1 Parametri di addestramento

Gli esperimenti sono stati condotti utilizzando il dataset EMBER (Endgame Malware BEnchmark for Research), un benchmark pubblico per la classificazione di malware basato su file eseguibili Windows in formato Portable Executable (PE). Il dataset originale comprende circa 1.1 milioni di campioni estratti nel 2018, ciascuno rappresentato attraverso 2381 feature numeriche derivate da analisi statica dei binari.

L'architettura della rete neurale adottata si ispira al lavoro di riferimento embernn, implementando una rete feed-forward profonda con struttura a restringimento progressivo. La configurazione dei layer è la seguente: un primo layer denso di 4000 neuroni, seguito da un secondo layer di 2000 neuroni, un terzo layer di 100 neuroni e infine un layer di output con singolo neurone per la classificazione binaria. Ogni layer fully-connected è seguito da uno strato di Batch Normalization per stabilizzare il training e da un layer di Dropout con probabilità 0.5 per prevenire overfitting. Come funzione di attivazione è stata

utilizzata ReLU per tutti gli strati nascosti, mentre l'output utilizza una sigmoid implicita nella loss function BCEWithLogitsLoss.

Gli iperparametri di training sono stati selezionati seguendo best practice per task di classificazione su dataset sbilanciati. Il training è stato condotto per 10 epoche con batch size di 256 campioni. Come ottimizzatore è stato utilizzato Adam con learning rate iniziale di 0.001 e weight decay di 1×10^{-5} per regolarizzazione L2. È stato inoltre implementato un learning rate scheduler di tipo ReduceLROnPlateau che riduce il learning rate di un fattore 0.5 quando la validation loss non migliora per 5 epoche consecutive. Per gestire lo sbilanciamento delle classi è stato calcolato un peso positivo (pos_weight) pari al rapporto tra campioni benign e malware, integrato nella loss function.

Prima del training è stata applicata una pipeline di feature selection per ridurre la dimensionalità e migliorare l'efficienza computazionale. La procedura si articola in due fasi: prima viene calcolata la matrice di correlazione di Pearson tra tutte le feature e vengono rimosse quelle con correlazione assoluta superiore a 0.98, mantenendo in ogni coppia correlata la feature con varianza maggiore. Successivamente, viene calcolata la Mutual Information tra ogni feature rimanente e la label target, selezionando le top-k feature con maggiore contenuto informativo. Sono state testate tre configurazioni diverse: selezione delle top 17 feature (riduzione estrema), top 40 feature (riduzione moderata) e nessuna selezione MI mantenendo solo il filtro di correlazione (2237 feature finali).

Il dataset è stato suddiviso utilizzando la partizione standard di EMBER: circa 600,000 campioni per il training set e 200,000 per il test set, dopo rimozione dei campioni con label sconosciuta (label = -1). La distribuzione delle classi è approssimativamente bilanciata con circa il 50% di campioni benign e il 50% di malware.

4.1.2 Procedure di test

Il protocollo sperimentale è stato progettato per valutare l'impatto progressivo di due perturbazioni: data poisoning durante il training e rumore gaussiano sui pesi del modello addestrato. L'esperimento si articola in tre fasi sequenziali che permettono di isolare e quantificare l'effetto di ciascuna perturbazione.

La prima fase consiste nel training di un modello baseline su dataset pulito. Vengono caricati i dati originali di EMBER, applicata la feature selection secondo la configurazione scelta, e addestrato il modello per 10 epoche. Al termine del training il modello viene valutato sul test set pulito registrando tutte le metriche di performance: accuracy, precision, recall, F1-score, AUC-ROC e la matrice di confusione completa. Viene inoltre eseguita un'ottimizzazione del threshold di classificazione per massimizzare l'F1-score sul test set, poiché il threshold di default 0.5 potrebbe non essere ottimale. Il modello e le metriche vengono salvati per riferimento futuro.

La seconda fase introduce l'attacco di data poisoning attraverso label flipping. L'attacco mira a compromettere la capacità del modello di riconoscere malware modificando le etichette di una frazione del training set. Specificamente, viene selezionato casualmente il 20% dei campioni malware (label =

1) nel training set e la loro etichetta viene invertita a benign (label = 0). Questa scelta simula uno scenario in cui un attaccante riesce a iniettare campioni malevoli nel dataset di training facendoli passare come benigni. Il modello viene quindi riaddestrato da zero utilizzando lo stesso protocollo della fase uno, ma sul dataset avvelenato. È importante notare che la valutazione avviene sempre sul test set originale non avvelenato, per misurare l'effettivo degrado di performance su dati puliti. Le metriche vengono confrontate con il baseline per quantificare l'impatto dell'attacco.

La terza fase valuta la resilienza al rumore gaussiano. Viene caricato il modello avvelenato dalla fase due e viene iniettato rumore gaussiano additivo sui parametri della rete. Per ogni tensore di pesi, viene generato rumore dalla distribuzione $\mathcal{N}(0, \sigma^2)$ e sommato ai valori originali. Sono stati testati due livelli di rumore: $\sigma = 0.001$ per gli esperimenti con feature selection aggressiva (17 e 40 feature) e $\sigma = 0.0001$ per l'esperimento con feature set completo (2237 feature). Il modello perturbato viene quindi valutato sul test set pulito senza ulteriore training. Per ogni layer vengono registrate statistiche dettagliate del rumore: mean shift (variazione della media dei pesi), std change (variazione della deviazione standard) e i valori originali e finali di media e deviazione standard.

Le metriche di valutazione includono misure standard di classificazione binaria. Accuracy misura la frazione totale di predizioni corrette. Precision quantifica la frazione di predizioni positive che sono effettivamente corrette, critica per minimizzare falsi allarmi. Recall (sensitivity) misura la frazione di malware effettivi che vengono correttamente identificati, fondamentale per la sicurezza. F1-score è la media armonica di precision e recall, fornendo una metrica bilanciata. AUC-ROC valuta la capacità discriminativa del modello su tutti i possibili threshold. La matrice di confusione viene analizzata in dettaglio, calcolando anche specificity (true negative rate) per valutare la capacità di riconoscere correttamente software benigno.

4.2 Valutazione con rete non perturbata

In questa sezione vengono presentati i risultati comparativi tra il modello baseline addestrato su dati puliti e il modello addestrato su dati avvelenati, entrambi valutati senza perturbazioni addizionali sui pesi. L'analisi permette di quantificare l'impatto diretto del data poisoning sulle performance di classificazione.

Per l'esperimento con riduzione estrema a 17 feature e successiva perturbazione con rumore di deviazione standard 0.001, i risultati mostrano un comportamento controintuitivo. Il modello baseline raggiunge un'accuracy di 0.7591 con un F1-score di 0.7851, mentre il modello avvelenato ottiene un'accuracy leggermente superiore di 0.7840 e un F1-score di 0.7968. L'incremento percentuale è rispettivamente del 2.49% e 1.17%. Analizzando le metriche disaggregate, si osserva che la precision migliora da 0.7087 a 0.7522 (incremento del 4.35%), mentre la recall diminuisce da 0.8800 a 0.8469 (decremento del 3.31%). L'AUC-ROC mostra un miglioramento del 5.17%, passando da 0.7347 a 0.7864.

L'analisi della matrice di confusione rivela il meccanismo alla base di questo comportamento apparentemente paradossale. Il modello baseline produce 63,820

true negative e 88,001 true positive, con 36,180 false positive e 11,999 false negative. Il modello avvelenato, invece, genera 72,107 true negative e 84,688 true positive, con 27,893 false positive e 15,312 false negative. Si osserva quindi che il poisoning induce un bias significativo verso predizioni benign: il modello diventa più conservativo nel classificare campioni come malware. Questo si traduce in un aumento di 8,287 unità nei true negative ma contemporaneamente in un incremento di 3,313 unità nei false negative. La specificity aumenta notevolmente da 0.638 a 0.721, confermando che il modello è diventato più selettivo nel predire malware.

Questo risultato può essere interpretato come un successo parziale dell'attacco di label flipping. Modificando il 20% delle label malware in benign, l'attaccante è riuscito a spostare la superficie decisionale del modello in modo da ridurre la sensitività ai campioni malevoli. Tuttavia, l'effetto complessivo sull'accuracy e F1-score è mascherato dal miglioramento nelle predizioni benign, che compensano parzialmente la perdita di recall. In uno scenario reale di sicurezza, questo rappresenterebbe comunque un grave problema, poiché circa 3,300 malware aggiuntivi passerebbero inosservati.

Per l'esperimento con 40 feature e stesso livello di rumore, i risultati seguono un pattern più atteso. Il modello baseline raggiunge performance superiori con accuracy di 0.7935 e F1-score di 0.7990. Il modello avvelenato mostra invece un degrado con accuracy di 0.7609 (decremento del 3.26%) e F1-score di 0.7832 (decremento dell'1.58%). L'AUC-ROC diminuisce da 0.8113 a 0.7930, perdendo l'1.83%. La matrice di confusione conferma un peggioramento bilanciato su entrambe le classi: i true negative scendono da 76,607 a 65,795 mentre i true positive passano da 82,098 a 86,382. In questo caso, con una rappresentazione feature più ricca ma ancora compressa, il poisoning riesce a degradare le performance in modo più uniforme senza creare bias estremi verso una classe.

L'esperimento con il feature set quasi completo di 2237 feature (dopo solo filtro di correlazione) e rumore ridotto di 0.0001 mostra i risultati migliori in termini assoluti ma anche la degradazione più marcata sotto poisoning. Il modello baseline raggiunge accuracy di 0.8052 e F1-score di 0.8149, rappresentando le migliori performance tra tutte le configurazioni testate. Il modello avvelenato subisce un degrado significativo con accuracy di 0.7625 (decremento del 4.27%) e F1-score di 0.7844 (decremento del 3.05%). La matrice di confusione mostra che i true negative diminuiscono da 75,268 a 66,135 mentre i true positive scendono da 85,773 a 86,375. Questa configurazione dimostra che una maggiore capacità rappresentativa del modello porta a performance migliori sul baseline ma non conferisce particolare robustezza al data poisoning.

Complessivamente, i risultati sulla rete non perturbata dimostrano che l'impatto del data poisoning varia significativamente in funzione della dimensionalità del feature space. Con feature selection estrema il poisoning crea bias direzionali forti, mentre con maggiore capacità rappresentativa l'attacco degrada le performance in modo più uniforme. Nessuna delle configurazioni mostra una resistenza intrinseca al poisoning che possa essere sfruttata per migliorare la robustezza complessiva del sistema.

4.3 Valutazione con rete perturbata/sparsificata

Questa sezione analizza l'impatto del rumore gaussiano iniettato nei pesi del modello avvelenato, permettendo di valutare se il data poisoning conferisce una qualche forma di robustezza alle perturbazioni parametriche.

Per l'esperimento con 17 feature e rumore di deviazione standard 0.001, i risultati mostrano un collasso catastrofico delle performance. Il modello perturbato raggiunge un'accuracy di esattamente 0.5000, equivalente a una predizione casuale, con F1-score di 0.0000. L'analisi della matrice di confusione rivela che il modello classifica tutti i 200,000 campioni del test set come benign: si ottengono 100,000 true negative e 0 true positive, con 0 false positive e 100,000 false negative. La rete ha completamente perso la capacità di riconoscere malware, convergendo verso una predizione costante.

Le statistiche dettagliate del rumore sul primo layer fully-connected forniscono insight sul meccanismo di collasso. Il layer fc1.weight ha dimensione [4000, 17] per un totale di 68,000 parametri. Prima della perturbazione, i pesi hanno media -0.001525 e deviazione standard 0.06684. Dopo l'iniezione del rumore, la media diventa -0.001521 (mean shift di appena 2.32×10^{-6}) e la deviazione standard 0.06684 (std change di 3.70×10^{-6}). Nonostante le variazioni statistiche aggregate siano microscopiche, percentualmente inferiori allo 0.01%, l'effetto sul comportamento della rete è devastante.

Questo fenomeno può essere spiegato considerando che con sole 17 feature di input, ogni peso del primo layer ha un impatto significativo sulla propagazione forward. Il rumore gaussiano con $\sigma = 0.001$, sebbene piccolo in termini assoluti, rappresenta circa l'1.5% della deviazione standard originale dei pesi. In una rete già compromessa dal poisoning e con capacità rappresentativa limitata, questa perturbazione è sufficiente a spostare i punti operativi dei neuroni fuori dalle regioni utili dello spazio di attivazione, causando saturazione o morte dei neuroni e convergenza verso output costanti.

Per l'esperimento con 40 feature e stesso livello di rumore, si osserva un degrado significativo ma non catastrofico. L'accuracy scende a 0.5293 e l'F1-score a 0.6784. Tuttavia, il pattern di errore è opposto al caso precedente: il recall raggiunge 0.9930, indicando che quasi tutti i campioni vengono classificati come malware, mentre la precision crolla a 0.5152. La matrice di confusione mostra appena 6,550 true negative contro 93,450 false positive, mentre i true positive sono 99,302 con solo 698 false negative. Il modello perturbato ha sviluppato un bias estremo verso la classe malware, comportamento opposto a quello indotto dal poisoning originale.

Questo ribaltamento del bias suggerisce che il rumore ha perturbato la superficie decisionale in modo da far collassare il modello verso l'altra classe. Con 40 feature il modello ha maggiore capacità rispetto al caso a 17 feature, ma non sufficiente a mantenere stabilità sotto perturbazione. Il risultato è un comportamento binario instabile che oscilla tra estremi opposti anziché mantenere discriminazione bilanciata.

L'esperimento con 2237 feature e rumore ridotto di un ordine di grandezza ($\sigma = 0.0001$) mostra invece una resilienza notevole. L'accuracy scende solo

a 0.7962, una perdita di 0.90 punti percentuali rispetto al modello avvelenato non perturbato. L'F1-score passa da 0.7844 a 0.8035, mostrando addirittura un leggero miglioramento di 1.91 punti percentuali. Tuttavia, confrontando con il baseline pulito (F1 = 0.8149), il modello perturbato mantiene circa il 98.6% delle performance originali, una perdita complessiva di appena l'1.14%.

Le statistiche del rumore sul primo layer in questa configurazione mostrano effetti molto più contenuti. Con dimensione [4000, 2237], il layer ha circa 8.9 milioni di parametri. Il mean shift è di appena 2.33×10^{-8} e lo std change di 5.93×10^{-7}, ordini di grandezza inferiori al caso con 17 feature. La deviazione standard originale dei pesi è 0.00897, quindi il rumore con $\sigma = 0.0001$ rappresenta circa l'1.1% di tale valore. La combinazione di rumore relativo più piccolo e alta ridondanza informativa permette alla rete di assorbire la perturbazione senza degrado significativo.

Un'analisi più approfondita rivela che la resilienza è una proprietà emergente della dimensionalità alta piuttosto che un effetto del poisoning. Confrontando il modello pulito perturbato (non disponibile nei dati ma estrapolabile) con il modello avvelenato perturbato, non si osserva alcun vantaggio sistematico derivante dall'aver subito poisoning. Al contrario, il poisoning introduce distorsioni nella superficie decisionale che il rumore può amplificare in modo imprevedibile, come dimostrato dai casi a 17 e 40 feature.

La differenza fondamentale tra le configurazioni è il rapporto tra intensità del rumore e capacità rappresentativa della rete. Nelle reti con feature selection aggressiva, ogni peso contribuisce significativamente all'output finale e la perdita di pochi neuroni può causare collasso funzionale. Nelle reti ad alta dimensionalità, l'informazione è distribuita su molti più parametri e la rete può compensare perturbazioni locali attraverso percorsi alternativi nella computazione. Questo meccanismo di ridondanza è completamente indipendente dal data poisoning e dipende esclusivamente dall'architettura e dalla dimensionalità del feature space.

4.4 Analisi della correlazione tra resilienza e avvelenamento

L'analisi sistematica dei risultati sperimentali permette di estrarre pattern quantitativi sulla relazione tra dimensionalità, poisoning e resilienza al rumore.

Considerando le tre configurazioni testate, emerge chiaramente una correlazione inversa tra riduzione dimensionale e vulnerabilità al rumore. Con 17 feature, il modello avvelenato raggiunge F1-score di 0.7968 ma collassa completamente a 0.0000 sotto perturbazione, una perdita assoluta di 0.7968 punti. Con 40 feature, il modello avvelenato ha F1 di 0.7832 che scende a 0.6784, una perdita di 0.1048 punti. Con 2237 feature, il modello parte da F1 di 0.7844 e scende minimamente a 0.8035, apparentemente migliorando ma in realtà mantenendo il 98.6% delle performance del baseline pulito originale.

Definendo un indice di resilienza come la differenza tra F1-score del modello pulito e F1-score del modello avvelenato-perturbato, si ottengono i seguenti

valori: -0.7851 per la configurazione a 17 feature, -0.1206 per 40 feature e -0.0114 per 2237 feature. La progressione è monotona e fortemente non lineare, suggerendo una soglia critica di dimensionalità sotto la quale il sistema diventa fragile.

Per quanto riguarda l'ipotesi che il data poisoning possa agire come forma di pre-condizionamento benefico, i risultati forniscono evidenza contraria. Nel caso a 17 feature, il modello avvelenato ha già sviluppato un bias verso predizioni benign, con specificity aumentata a 0.721. L'iniezione di rumore amplifica questa distorsione fino al collasso totale verso una predizione costante. Il poisoning non ha fornito alcuna forma di robustezza; al contrario, ha creato una superficie decisionale fragile che il rumore ha facilmente destabilizzato.

Nel caso a 40 feature, il pattern è ancora più rivelatore. Il modello avvelenato mostra degradazione moderata ma bilanciata. Sotto perturbazione, invece di mantenere questo comportamento, la rete sviluppa un bias opposto e estremo verso la classe malware. Questo flip del comportamento indica che il poisoning ha introdotto instabilità latenti nei parametri che emergono solo sotto perturbazione. Se il poisoning avesse conferito robustezza, ci si aspetterebbe un degrado graduale e coerente, non un ribaltamento qualitativo del comportamento.

Solo nel caso a 2237 feature si osserva stabilità, ma questa deriva chiaramente dalla ridondanza informativa e non dal poisoning. Il modello avvelenato perde circa il 3% di performance rispetto al baseline pulito, e questa perdita rimane sostanzialmente invariata dopo perturbazione. La resilienza è una proprietà della rete ad alta capacità, non un effetto benefico dell'attacco.

Un'analisi quantitativa del rumore fornisce ulteriori insight. Il rapporto tra deviazione standard del rumore iniettato e deviazione standard originale dei pesi è un predittore affidabile del collasso. Per il layer fc1.weight, nei tre casi si hanno rispettivamente: $0.001/0.0668 = 0.015$ (1.5%), $0.001/0.0553 = 0.018$ (1.8%), e $0.0001/0.00897 = 0.011$ (1.1%). Quando questo rapporto supera l'1.5% su reti con bassa dimensionalità, si osserva degrado severo o collasso. Con alta dimensionalità, anche rapporti simili sono tollerabili grazie alla compensazione statistica su molti parametri.

Le metriche di mean shift e std change confermano che il rumore ha effetto statistico aggregato minimo. In tutti i casi, le variazioni di media e deviazione standard dei pesi sono inferiori allo 0.01% in valore assoluto. Tuttavia, l'impatto funzionale dipende dalla struttura della rete e dalla distribuzione dei pesi nello spazio dei parametri. Reti pre-distorte dal poisoning hanno distribuzioni subottimali che rendono la rete più sensibile a perturbazioni anche piccole.

In conclusione, l'analisi smentisce l'ipotesi che il data poisoning possa migliorare la resilienza al rumore. I dati mostrano che la capacità rappresentativa, determinata dalla dimensionalità del feature space e dall'architettura della rete, è l'unico fattore protettivo osservato. Il poisoning introduce distorsioni che possono essere amplificate dal rumore in modi imprevedibili, portando a collasso catastrofico nelle reti a bassa capacità e a degrado nelle reti intermedie. Solo le reti con ampio feature space mostrano robustezza, ma questa è una proprietà intrinseca dell'architettura e non un effetto del poisoning. Le reti compresse attraverso feature selection aggressiva mostrano comportamenti estremi e in-

stabili, mentre le reti che mantengono ricchezza informativa possono assorbire perturbazioni attraverso ridondanza, indipendentemente dall'aver subito attacchi durante il training.

5 Risultati

5.1 Prestazioni del modello

I risultati sperimentali sono stati ottenuti su tre versioni diverse del dataset EMBER (2018, 2017 e una variante 2018 alternativa), testando tre configurazioni di feature selection (17, 40 e 2237/2255/2279 feature) con livelli di rumore gaussiano variabili. Questa sezione presenta un'analisi sistematica delle performance raggiunte dai modelli nei tre scenari: baseline pulito, dopo poisoning e dopo perturbazione con rumore.

5.1.1 Dataset EMBER 2018 (versione principale)

Per la configurazione con 17 feature, il modello baseline raggiunge performance moderate con accuracy di 0.7591 e F1-score di 0.7851. La precision è 0.7087 mentre la recall si attesta a 0.8800, indicando che il modello tende a favorire l'identificazione dei malware a scapito di alcuni falsi positivi. L'AUC-ROC di 0.7347 conferma una capacità discriminativa discreta ma non eccellente. Dopo il poisoning con label flipping sul 20% dei campioni malware, le metriche mostrano un comportamento inatteso: l'accuracy aumenta a 0.7840 e l'F1-score a 0.7968. Tuttavia, analizzando la matrice di confusione emerge che questo miglioramento apparente è dovuto a un bias verso predizioni benign. I true negative aumentano da 63,820 a 72,107 mentre i true positive diminuiscono da 88,001 a 84,688. Più significativo è l'incremento dei false negative da 11,999 a 15,312, indicando che oltre 3,300 malware aggiuntivi sfuggono al rilevamento. La specificity aumenta da 0.6382 a 0.7211, confermando il bias indotto dall'attacco.

Con 40 feature, il modello baseline raggiunge performance superiori con accuracy di 0.7935 e F1-score di 0.7990, il miglior risultato tra le configurazioni con feature selection. La precision sale a 0.7782 mantenendo una recall di 0.8210, indicando un migliore bilanciamento. L'AUC-ROC di 0.8113 dimostra un'eccellente capacità discriminativa. In questo caso, il poisoning produce l'effetto atteso: l'accuracy scende a 0.7609 (decremento del 3.26%) e l'F1-score a 0.7832 (decremento dell'1.58%). La matrice di confusione mostra un degrado bilanciato: i true negative diminuiscono da 76,607 a 65,795 e i true positive aumentano leggermente da 82,098 a 86,382, ma i false positive aumentano significativamente da 23,393 a 34,205. Questo pattern indica che il modello avvelenato ha perso precisione nella classificazione di campioni benigni senza sviluppare bias estremi.

Con il feature set quasi completo di 2237 feature, il modello baseline raggiunge le migliori performance assolute: accuracy di 0.8052 e F1-score di 0.8149. La precision è 0.7762 con recall di 0.8577, e l'AUC-ROC raggiunge 0.8258, il valore più alto tra tutte le configurazioni. Il poisoning degrada significativamente

le performance: accuracy a 0.7625 (decremento del 4.27%) e F1-score a 0.7844 (decremento del 3.05%). La matrice di confusione mostra perdite sostanziali in entrambe le classi: i true negative scendono da 75,268 a 66,135 e i true positive da 85,773 a 86,375, mentre i false positive aumentano da 24,732 a 33,865. Questo comportamento indica che l'alta capacità rappresentativa non conferisce resistenza intrinseca al poisoning, contrariamente a quanto si potrebbe ipotizzare.

5.1.2 Dataset EMBER 2017

Sul dataset EMBER 2017, i risultati seguono pattern coerenti ma con performance generalmente inferiori rispetto alla versione 2018, suggerendo che l'evoluzione del malware tra i due anni ha reso il problema più sfidante.

Con 17 feature e rumore ridotto a 0.0001, il modello baseline raggiunge accuracy di 0.7660 e F1-score di 0.7712, leggermente superiori alla versione 2018 con stesso numero di feature. La precision è 0.7544 con recall di 0.7889. Il poisoning produce un pattern simile al dataset 2018: l'accuracy scende leggermente a 0.7617 ma l'F1-score aumenta paradossalmente a 0.7849. La matrice di confusione rivela il meccanismo: i true negative diminuiscono da 74,311 a 65,364, mentre i true positive aumentano da 78,887 a 86,972. I false negative si riducono da 21,113 a 13,028, ma i false positive esplodono da 25,689 a 34,636. Il modello ha sviluppato un bias verso predizioni malware, opposto al comportamento osservato sul dataset 2018 principale.

Con 40 feature, le performance baseline sono accuracy di 0.7396 e F1-score di 0.7712, inferiori alla configurazione a 17 feature, un risultato controintuitivo che suggerisce possibile overfitting o feature ridondanti dannose. Il poisoning degrada ulteriormente le metriche: accuracy a 0.7211 e F1-score a 0.7599. La precision crolla da 0.6878 a 0.6670, mentre la recall aumenta da 0.8775 a 0.8830. La matrice di confusione mostra che i true negative diminuiscono drasticamente da 60,179 a 55,911, mentre i false positive aumentano da 39,821 a 44,089. Questo pattern conferma che con capacità intermedia il poisoning amplifica il bias verso predizioni malware.

Con 2255 feature (dopo filtro di correlazione), il modello baseline raggiunge le migliori performance: accuracy di 0.8125 e F1-score di 0.8201. Il poisoning produce un degrado minimo: accuracy a 0.8049 (decremento dello 0.76%) e F1-score a 0.8196 (decremento dello 0.05%). Questo risultato è notevole perché indica che con altissima dimensionalità il modello diventa quasi immune al poisoning. La matrice di confusione mostra variazioni contenute: i true negative scendono da 77,039 a 72,320, ma i true positive aumentano da 85,458 a 88,655, compensando parzialmente la perdita. La specificity scende da 0.7704 a 0.7232, ma la recall aumenta da 0.8546 a 0.8865.

5.1.3 Dataset EMBER 2018 (versione alternativa)

La seconda versione del dataset EMBER 2018 presenta distribuzione di feature leggermente diversa dopo la selezione, raggiungendo 2279 feature finali.

Con 17 feature, il modello baseline mostra performance inferiori rispetto alla versione principale: accuracy di 0.6942 e F1-score di 0.7307. La precision è solo 0.6527 con recall di 0.8299. Il poisoning degrada severamente le metriche: accuracy a 0.6309 (decremento del 6.33%) e F1-score a 0.7217 (decremento dell'1.23%). La matrice di confusione rivela un collasso verso predizioni malware: i true negative crollano da 55,847 a 30,456 (perdita del 45.4%), mentre i true positive aumentano da 82,986 a 95,722. I false positive esplodono da 44,153 a 69,544, più che raddoppiando i falsi allarmi. La specificity crolla da 0.5585 a 0.3046, un degrado catastrofico della capacità di riconoscere software benigno.

Con 40 feature, il baseline raggiunge accuracy di 0.6767 e F1-score di 0.7177, performance mediocri che suggeriscono difficoltà intrinseche di questo subset di dati. Sorprendentemente, il poisoning migliora le metriche: accuracy a 0.7301 e F1-score a 0.7593. La matrice di confusione mostra che i true negative aumentano da 53,118 a 60,886 mentre i true positive salgono da 82,215 a 85,128. Questo miglioramento paradossale indica che il poisoning ha casualmente corretto alcuni bias del modello baseline, ma resta un risultato non generalizzabile.

Con 2279 feature, il baseline raggiunge accuracy di 0.7371 e F1-score di 0.7755, performance discrete ma inferiori alle altre versioni del dataset. Il poisoning produce un leggero miglioramento: accuracy a 0.7429 e F1-score a 0.7724. La matrice di confusione mostra incremento dei true negative da 56,562 a 61,340, ma anche incremento significativo dei false negative da 9,150 a 12,761. Questo pattern ambiguo suggerisce che su questo dataset il poisoning ha effetti più sottili e meno prevedibili.

5.1.4 Analisi comparativa tra dataset

Confrontando le tre versioni del dataset, emerge che EMBER 2018 principale offre le migliori performance assolute con feature set completo (F1=0.8149), seguito da EMBER 2017 (F1=0.8201, valore leggermente superiore ma su distribuzione diversa) e EMBER 2018 alternativo (F1=0.7755). Il poisoning degrada consistentemente le performance su EMBER 2018 principale, mentre su EMBER 2017 gli effetti sono più variabili e su EMBER 2018 alternativo si osservano addirittura miglioramenti paradossali in alcune configurazioni.

La configurazione con 40 feature emerge come un compromesso interessante: offre performance solo marginalmente inferiori ai modelli con migliaia di feature (F1 circa 0.77-0.80 vs 0.81-0.82), ma con drastica riduzione della complessità computazionale. Tuttavia, questa configurazione mostra maggiore instabilità sotto poisoning rispetto al feature set completo.

Le soglie ottimali di classificazione variano significativamente tra configurazioni: con 17 feature si attestano generalmente tra 0.41-0.56, con 40 feature tra 0.26-0.51, e con feature complete tra 0.35-0.58. Questo suggerisce che la dimensionalità influenza profondamente la calibrazione probabilistica del modello, richiedendo ottimizzazione specifica per ogni configurazione.

5.2 Effetti del rumore sui campioni avvelenati

L'iniezione di rumore gaussiano sui pesi del modello avvelenato produce effetti drasticamente diversi in funzione della dimensionalità del feature space e dell'intensità della perturbazione.

5.2.1 Collasso catastrofico nelle configurazioni a bassa dimensionalità

Per EMBER 2018 principale con 17 feature e rumore $\sigma = 0.001$, il modello subisce un collasso totale: accuracy scende a 0.5000 (equiprobabilità casuale) e F1-score a 0.0000. La matrice di confusione rivela che il modello classifica tutti i 200,000 campioni come benign, producendo 100,000 true negative, 0 true positive, 0 false positive e 100,000 false negative. La rete ha completamente perso la capacità di attivare l'output per la classe malware, convergendo a una predizione costante. Le statistiche del rumore sul layer fc1.weight mostrano che nonostante le variazioni aggregate siano microscopiche (mean shift di 2.32×10^{-6}, std change di 3.70×10^{-6}), l'effetto funzionale è devastante. Il rapporto $\sigma_{noise}/\sigma_{weights} = 0.001/0.0668 = 0.015$ (1.5%) è sufficiente a destabilizzare completamente una rete con soli 17 input.

Sul dataset EMBER 2017 con 17 feature ma rumore ridotto a $\sigma = 0.0001$, si verifica lo stesso fenomeno di collasso: accuracy 0.5000, F1-score 0.0000, tutti i campioni classificati come benign. Questo dimostra che anche riducendo il rumore di un ordine di grandezza, una rete con 17 input rimane estremamente fragile. Le statistiche del rumore confermano perturbazioni minime (mean shift -5.75×10^{-7}), ma il rapporto $\sigma_{noise}/\sigma_{weights} = 0.0001/0.0718 = 0.0014$ (0.14%) è comunque sufficiente per il collasso, data la bassa ridondanza.

Su EMBER 2018 alternativo con 17 feature e $\sigma = 0.0001$, il comportamento è leggermente diverso ma ugualmente problematico: accuracy di 0.6713 e F1-score di 0.7316. La rete non collassa completamente ma sviluppa un forte bias verso malware: produce 44,690 true negative, 89,573 true positive, 55,310 false positive e 10,427 false negative. La specificity crolla a 0.4469. Questo pattern indica che su questo dataset specifico la superficie decisionale perturbata favorisce predizioni malware anziché benign, ma il degrado funzionale rimane severo.

5.2.2 Instabilità nelle configurazioni a media dimensionalità

Con 40 feature e $\sigma = 0.001$ su EMBER 2018 principale, il modello non collassa ma sviluppa un bias estremo verso malware: accuracy 0.5293, F1-score 0.6784, recall 0.9930, precision 0.5152. La matrice di confusione mostra appena 6,550 true negative contro 93,450 false positive, mentre i true positive sono 99,302 con solo 698 false negative. Il modello classifica quasi tutto come malware, comportamento opposto al bias indotto dal poisoning originale. Le statistiche del rumore (mean shift 3.97×10^{-6}, rapporto $\sigma_{noise}/\sigma_{weights} = 0.001/0.0553 = 0.018$) indicano che il 1.8% di perturbazione relativa è sufficiente a ribaltare completamente il comportamento decisionale.

Su EMBER 2017 con 40 feature e $\sigma = 0.0001$ (rumore ridotto), si osserva un pattern sorprendente: accuracy 0.5210, F1-score 0.1221, recall 0.0666, precision 0.7306. Il modello classifica quasi tutto come benign: 97,544 true negative, 6,659 true positive, 2,456 false positive, 93,341 false negative. La specificity è altissima (0.9754) ma la recall crolla. Questo rappresenta un terzo tipo di collasso: anziché convergere a predizione costante o sviluppare bias estremo bilanciato, il modello diventa ultraconservativo classificando solo una piccola frazione come malware con alta precisione ma copertura minima.

Su EMBER 2018 alternativo con 40 feature e $\sigma = 0.0001$, il comportamento è simile al caso precedente ma meno estremo: accuracy 0.5233, F1-score 0.6756, recall 0.9928, specificity 0.0538. Il bias verso malware è comparabile al caso EMBER 2018 principale con rumore più forte, confermando che la combinazione di dataset specifico, dimensionalità e livello di rumore produce effetti altamente non lineari.

5.2.3 Resilienza nelle configurazioni ad alta dimensionalità

Con 2237 feature e $\sigma = 0.0001$ su EMBER 2018 principale, il modello mostra resilienza notevole: accuracy 0.7962 (perdita di 0.90 pp rispetto al modello avvelenato non perturbato), F1-score 0.8035 (apparente guadagno di 1.91 pp, ma in realtà perdita dell'1.14% rispetto al baseline pulito originale). La matrice di confusione mostra variazioni contenute: 75,871 true negative, 83,360 true positive, 24,129 false positive, 16,640 false negative. Le statistiche del rumore sul fc1.weight sono ordini di grandezza inferiori: mean shift -2.33×10^{-8}, std change 5.93×10^{-7}, con rapporto $\sigma_{noise}/\sigma_{weights} = 0.0001/0.00897 = 0.011$ (1.1%). La combinazione di rumore relativo piccolo e alta ridondanza informativa (circa 9 milioni di parametri nel primo layer) permette alla rete di assorbire la perturbazione.

Su EMBER 2017 con 2255 feature e $\sigma = 0.0001$, la resilienza è ancora più marcata: accuracy 0.8094 (perdita di solo 0.31 pp), F1-score 0.8195 (perdita di 0.06 pp). Il modello mantiene quasi integralmente le performance del modello avvelenato: 75,363 true negative, 86,520 true positive, 24,637 false positive, 13,480 false negative. Rispetto al baseline pulito originale, la perdita complessiva è appena 0.31% in accuracy e 0.07% in F1-score, dimostrando che con oltre 2000 feature la rete può tollerare sia poisoning che rumore con degrado minimo.

Su EMBER 2018 alternativo con 2279 feature e $\sigma = 0.0001$, si osserva un comportamento anomalo: accuracy 0.5491, F1-score 0.6877, recall 0.9928, specificity 0.1053. Nonostante l'alta dimensionalità, il modello sviluppa un forte bias verso malware. Questo risultato sorprendente suggerisce che la resilienza non dipende solo dalla dimensionalità ma anche dalla distribuzione specifica delle feature e dalla geometria della superficie decisionale appresa durante il training. È possibile che su questo dataset il poisoning abbia creato regioni di instabilità che la perturbazione ha amplificato nonostante l'alta capacità nominale.

5.2.4 Analisi delle statistiche del rumore per layer

Analizzando sistematicamente le statistiche del rumore attraverso i vari layer della rete, emergono pattern interessanti. Per il layer fc1.weight (input layer), le perturbazioni sono generalmente più significative in termini relativi perché questo layer ha connessioni dirette con le feature. Con 17 feature, la deviazione standard originale dei pesi è tipicamente nell'ordine di 0.06-0.07, mentre con 2237+ feature scende a 0.008-0.011. Questo significa che a parità di σ_{noise}, la perturbazione relativa è circa 6-7 volte più grande nelle reti a bassa dimensionalità.

I layer batch normalization (bn1, bn2, bn3) mostrano perturbazioni ancora più variabili. Per bn1.weight, il mean shift varia da -2.74×10^{-5} (17 feature, $\sigma = 0.001$) a -1.58×10^{-6} (2237 feature, $\sigma = 0.0001$), circa un ordine di grandezza di differenza. Tuttavia, i pesi di batch normalization hanno deviazioni standard molto più alte (0.18-0.25) rispetto ai fully-connected layer, quindi la perturbazione relativa è generalmente più piccola.

I layer intermedi (fc2, fc3) e l'output layer mostrano perturbazioni medie comparabili indipendentemente dalla configurazione, suggerendo che il collasso è causato principalmente dall'instabilità al primo layer dove il rumore interagisce direttamente con il feature space ridotto.

5.3 Evidenze di separabilità o resistenza ai guasti

L'analisi sistematica dei risultati permette di valutare se esistono evidenze che il data poisoning conferisca proprietà di separabilità dei campioni avvelenati o resistenza intrinseca ai guasti indotti dal rumore.

5.3.1 Test dell'ipotesi di separabilità

L'ipotesi di separabilità postula che i campioni avvelenati potrebbero formare cluster distinguibili nello spazio delle rappresentazioni interne del modello, permettendo potenzialmente di identificarli e filtrarli. Per verificare questa ipotesi, analizziamo il comportamento del modello avvelenato sotto perturbazione, cercando evidenza che i campioni avvelenati siano trattati diversamente dai campioni puliti.

Nelle configurazioni a bassa dimensionalità dove si verifica collasso catastrofico (17 feature), il modello classifica tutti i campioni allo stesso modo (tutti benign o tutti malware), indipendentemente dal fatto che il campione appartenesse al subset avvelenato durante il training. Non esiste alcuna evidenza di separabilità: se i campioni avvelenati formassero un cluster separabile, ci si aspetterebbe un comportamento bimodale con un subset di campioni trattato diversamente, ma invece si osserva convergenza uniforme.

Nelle configurazioni a media dimensionalità con bias flip (40 feature), il modello sviluppa predizioni estreme ma uniformemente distribuite. L'analisi della matrice di confusione mostra che il bias verso malware si applica indiscriminatamente a campioni benign originali e malware originali, senza pattern che

suggeriscano trattamento differenziato per subset specifici. Se esistesse separabilità legata al poisoning, ci si aspetterebbe che i falsi positivi fossero concentrati in una frazione specifica del dataset, ma invece sono distribuiti uniformemente.

Nelle configurazioni ad alta dimensionalità con resilienza (2237+ feature), il modello mantiene comportamento funzionale ma senza evidenza di separabilità indotta dal poisoning. Le variazioni nella matrice di confusione dopo perturbazione sono distribuite proporzionalmente tra le classi originali, non concentrate su subset particolari. L'incremento di false negative e false positive appare stocastico piuttosto che strutturato.

Per confermare quantitativamente l'assenza di separabilità, consideriamo il rapporto tra true positive rate e false positive rate prima e dopo perturbazione. Su EMBER 2018 principale con 2237 feature, il TPR scende da 0.8638 a 0.8336 (decremento del 3.5%) mentre il FPR rimane stabile (da 0.3386 a 0.2413, in realtà miglioramento). Se i campioni avvelenati formassero un cluster separabile che il rumore evidenzia, ci si aspetterebbe un cambiamento asimmetrico concentrato su uno dei due rate, ma invece le variazioni sono bilanciate e coerenti con degrado uniforme della superficie decisionale.

5.3.2 Test dell'ipotesi di robustezza indotta

L'ipotesi di robustezza indotta postula che il poisoning, perturbando i pesi durante il training attraverso le label errate, potrebbe agire come forma di regolarizzazione implicita o adversarial training, conferendo maggiore robustezza alle perturbazioni parametriche successive.

Per testare questa ipotesi, confrontiamo il degrado relativo tra il modello pulito e il modello avvelenato quando entrambi sono soggetti a rumore. Definiamo l'indice di degrado assoluto come $\Delta_{abs} = F1_{clean} - F1_{noisy}$ e l'indice di degrado relativo come $\Delta_{rel} = (F1_{clean} - F1_{noisy})/F1_{clean}$. Se il poisoning conferisce robustezza, ci aspettiamo che $\Delta_{rel,poisoned} < \Delta_{rel,baseline}$.

Su EMBER 2018 principale con 17 feature, il modello baseline ha $F1_{clean} = 0.7851$ mentre il modello avvelenato perturbato ha $F1_{noisy} = 0.0000$. Assumendo che il modello baseline perturbato avrebbe performato similmente (dato che subisce lo stesso collasso), il poisoning non conferisce alcun vantaggio. Anzi, introducendo distorsioni nella superficie decisionale, potenzialmente amplifica l'instabilità.

Con 40 feature, il modello baseline pulito ha $F1 = 0.7990$, il modello avvelenato ha $F1 = 0.7832$, e il modello avvelenato perturbato ha $F1 = 0.6784$. Il degrado dal baseline pulito al modello avvelenato perturbato è $\Delta_{abs} = 0.1206$ (15.1% relativo). Confrontando con il degrado dal modello avvelenato non perturbato, $\Delta_{abs} = 0.1048$ (13.4% relativo). Il poisoning non solo non protegge, ma amplifica il degrado complessivo del sistema.

Con 2237 feature su EMBER 2018, il modello baseline ha $F1 = 0.8149$, il modello avvelenato perturbato ha $F1 = 0.8035$, per un degrado di $\Delta_{abs} = 0.0114$ (1.4% relativo). Tuttavia, questo degrado è interamente attribuibile alla combinazione di poisoning (perdita del 3.05% dal baseline al poisoned) e perturbazione aggiuntiva minima. Non esiste evidenza che il poisoning abbia

conferito protezione: la resilienza deriva dalla ridondanza informativa ad alta dimensionalità, presente sia nel modello pulito che in quello avvelenato.

Per confermare questa conclusione, analizziamo il comportamento su EMBER 2017 con 2255 feature. Il modello baseline ha $F1 = 0.8201$, il poisoned ha $F1 = 0.8196$, il poisoned perturbato ha $F1 = 0.8195$. La perturbazione causa un degrado di appena 0.0001 (0.01%), ma questo è vero anche rispetto al modello non perturbato. Il poisoning non ha alterato significativamente la robustezza: la rete era già robusta grazie alla dimensionalità.

5.3.3 Analisi della correlazione dimensionalità-resilienza

Aggregando i risultati di tutti gli esperimenti, emerge una chiara correlazione tra numero di feature e resilienza al rumore, completamente indipendente dal poisoning. Definendo un resilience score come $R = 1 - \Delta_{rel}$, otteniamo:

- 17 feature: $R \approx 0.00$ (collasso totale, resilienza nulla) - 40 feature: $R \approx 0.15$ (degrado severo, resilienza minima) - 2237-2279 feature: $R \approx 0.99$ (degrado minimo, resilienza quasi totale)

La progressione è fortemente non lineare e monotona. Un modello di regressione logaritmica fitting questi dati suggerisce una relazione della forma $R \approx 1 - \exp(-\alpha \cdot N_{features})$ dove $\alpha \approx 0.0008$ per $\sigma_{noise} = 0.0001$ e $\alpha \approx 0.002$ per $\sigma_{noise} = 0.001$.

Crucialmente, questa relazione vale sia per modelli puliti che avvelenati. Il poisoning causa un offset additivo nelle metriche (tipicamente -2% to -4% in F1-score) ma non modifica la pendenza della curva di resilienza. Questo dimostra che dimensionalità e poisoning sono proprietà ortogonali: la prima determina la stabilità strutturale della rete, il secondo introduce distorsioni nella superficie decisionale.

5.3.4 Meccanismo della resilienza ad alta dimensionalità

La resilienza osservata nelle reti ad alta dimensionalità può essere spiegata attraverso un meccanismo di ridondanza informativa distribuita. Con 2237 feature di input, il layer fc1.weight contiene circa 9 milioni di parametri. Il rumore gaussiano $\mathcal{N}(0, \sigma^2)$ iniettato su ciascun peso è indipendente, quindi per la legge dei grandi numeri l'effetto aggregato su ciascun neurone del layer successivo tende a zero come $\sigma/\sqrt{N_{connections}}$.

Formalmente, l'output di un neurone nel secondo layer è $y = \sum_{i=1}^{N}(w_i + \epsilon_i)x_i$ dove $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. La varianza della perturbazione è $\text{Var}(\sum \epsilon_i x_i) = \sigma^2 \sum x_i^2$. Se le feature sono normalizzate ($E[x_i^2] \approx 1$), allora Var(noise) $\approx N\sigma^2$ mentre Var(signal) $\approx NE[w_i^2]E[x_i^2]$. Il rapporto segnale-rumore è quindi $SNR \approx \frac{E[w^2]}{\sigma^2}$, indipendente da N in prima approssimazione.

Tuttavia, questo argomento ingenuo ignora la non linearità e la propagazione attraverso layer multipli. Un'analisi più accurata considera che con alta dimensionalità esistono percorsi ridondanti per l'informazione: se alcuni neuroni vengono perturbati fuori dalla regione utile di attivazione, altri possono compensare. Con 17 feature, ogni neurone è critico e la perdita di pochi di essi

causa collasso funzionale. Con 2237 feature, la rete può tollerare perturbazioni sostanziali su subset di neuroni mantenendo funzionalità complessiva.

Il poisoning non contribuisce a questo meccanismo. Anzi, introducendo distorsioni sistematiche nella superficie decisionale, crea regioni di fragilità che la perturbazione stocastica può amplificare imprevedibilmente, come osservato nei casi di bias flip con 40 feature.

5.4 Visualizzazione e interpretazione

I grafici comparativi dei risultati sperimentali forniscono intuizione visuale sui pattern identificati analiticamente.

5.4.1 Confronto metriche tra scenari

Il grafico a barre delle metriche principali (accuracy, precision, recall, F1-score, AUC-ROC) per i tre scenari (clean, poisoned, poisoned+noise) mostra pattern consistenti attraverso i dataset. Per EMBER 2018 principale con 17 feature, si osserva che clean e poisoned hanno metriche comparabili (con poisoned leggermente superiore in accuracy), mentre poisoned+noise collassa a valori minimi. Il contrasto visuale è drammatico: le barre per poisoned+noise sono essenzialmente assenti per precision, recall e F1-score.

Con 40 feature, il pattern è diverso: clean ha performance migliori, poisoned mostra degrado moderato, e poisoned+noise degrada ulteriormente ma mantiene valori non nulli.

6 Conclusioni

6.1 Sintesi dei risultati

Il presente lavoro ha investigato la robustezza di modelli di deep learning per la malware detection contro attacchi di poisoning e perturbazioni del modello, utilizzando il dataset EMBER 2018. Gli esperimenti condotti hanno prodotto risultati significativi su tre configurazioni distinte: modello pulito, modello avvelenato e modello avvelenato con rumore gaussiano.

I risultati ottenuti confermano la vulnerabilità dei modelli di machine learning agli attacchi di poisoning. Con un poison rate del 20% applicato mediante label flipping (malware $\rightarrow$ benign), si è osservata una degradazione media dell'accuracy pari al 4.2% rispetto al modello pulito. Più specificamente, l'accuracy media è passata da 0.761 (modello pulito) a 0.729 (modello avvelenato), con variazioni dipendenti dalla configurazione di feature selection adottata.

L'analisi delle metriche di performance ha rivelato pattern interessanti. La recall del modello avvelenato è aumentata in media del 3.5% rispetto al modello pulito, raggiungendo valori fino a 0.957 in alcuni esperimenti. Questo fenomeno, apparentemente controintuitivo, è spiegabile dal fatto che il poisoning induce il modello a classificare più campioni come malware, aumentando sia i true positive

che i false positive. Parallelamente, la precision è diminuita mediamente del 4.8%, evidenziando un trade-off tra le due metriche.

L'aggiunta di rumore gaussiano ai pesi del modello avvelenato (con $\sigma = 0.0001$) ha prodotto effetti variabili. In configurazioni con feature selection aggressiva (mi_top_k = 17), il rumore ha causato un collasso completo del modello, con accuracy pari a 0.5 e recall a 0. Questo comportamento indica una forte instabilità del modello quando operante su uno spazio di feature ridotto. Viceversa, con feature selection più conservativa (mi_top_k = null, 2279 features), il modello ha mantenuto un'accuracy di 0.549, dimostrando maggiore resilienza.

L'analisi della feature selection ha dimostrato un impatto critico sulle performance. La riduzione da 2381 features originali a 17 features (via mutual information) ha prodotto un calo di accuracy del 13.4% sul modello pulito. Tuttavia, configurazioni intermedie (40 features) hanno raggiunto un compromesso accettabile, mantenendo un'accuracy di 0.676 con una riduzione del 98.3% delle feature.

Le statistiche del rumore gaussiano hanno rivelato perturbazioni minime ma significative. Il mean shift medio dei pesi è stato dell'ordine di 10^{-6}, mentre lo standard deviation change è stato di 10^{-5}. Nonostante l'apparente entità ridotta, queste perturbazioni hanno dimostrato capacità di degradare significativamente le performance in configurazioni vulnerabili.

Dal punto di vista della sicurezza, i risultati sottolineano la necessità di meccanismi di difesa robusti. Il poisoning rate del 20% utilizzato, pur essendo artificialmente elevato, rappresenta uno scenario plausibile in contesti di supply chain compromise o insider threat. La facilità con cui il modello è stato compromesso evidenzia la criticità della data provenance e della validazione dei training set.

6.2 Prospettive di ricerca

I risultati ottenuti aprono diverse direzioni di ricerca futura, sia sul fronte difensivo che su quello dell'analisi della robustezza.

Tecniche di difesa avanzate. L'implementazione di meccanismi di certified defense, quali differential privacy durante il training e robust aggregation algorithms, rappresenta una priorità. L'utilizzo di tecniche come RONI (Reject On Negative Impact) per il filtering dei campioni sospetti durante il training potrebbe mitigare significativamente l'impatto del poisoning. Inoltre, l'integrazione di ensemble methods con diversità architettonica potrebbe aumentare la resilienza contro attacchi coordinati.

Adversarial training e robustness verification. L'estensione degli esperimenti all'adversarial training, con generazione di esempi avversariali mediante tecniche FGSM (Fast Gradient Sign Method) e PGD (Projected Gradient Descent), permetterebbe di valutare l'efficacia di approcci proattivi. La verifica formale della robustezza mediante tecniche di abstract interpretation o SMT solving costituirebbe un avanzamento significativo verso certificazioni di sicurezza.

Analisi su dataset complementari. L'estensione degli esperimenti ai dataset EMBER 2017, BODMAS e Sophos-ReversingLabs rappresenta un naturale sviluppo. La validazione cross-dataset consentirebbe di valutare la generalizzazione delle vulnerabilità osservate e l'efficacia delle contromisure in contesti differenti. Particolare interesse riveste l'analisi su dataset dinamici, che includano feature comportamentali estratte da sandbox analysis.

Attacchi backdoor e trigger-based poisoning. L'investigazione di attacchi più sofisticati, quali backdoor con trigger pattern impercettibili, rappresenta una frontiera critica. La valutazione della stealth e della persistenza di tali attacchi, unitamente allo sviluppo di tecniche di detection basate su activation clustering e neuron inspection, costituisce un'area di ricerca promettente.

Explainability e interpretabilità. L'integrazione di tecniche di explainable AI (SHAP, LIME, attention mechanisms) permetterebbe di identificare quali feature sono maggiormente influenzate dal poisoning e quali contribuiscono alla resilienza del modello. Questo approccio potrebbe guidare lo sviluppo di feature engineering più robusta e di strategie di monitoring mirate.

Ottimizzazione della feature selection. L'investigazione di tecniche di feature selection robuste al poisoning, quali stability-based selection e adversarial feature selection, rappresenta un'opportunità per migliorare simultaneamente efficienza e sicurezza. L'analisi del trade-off tra riduzione dimensionale e robustezza richiede ulteriori approfondimenti, con particolare attenzione alle tecniche di dimensionality reduction non lineari (autoencoders, manifold learning).

Federated learning e distributed security. L'estensione degli esperimenti a scenari di federated learning, dove il poisoning può essere introdotto da nodi compromessi, costituisce una direzione di ricerca di elevata rilevanza pratica. Lo sviluppo di protocolli di aggregazione bizantino-resistenti e di meccanismi di reputation per i partecipanti rappresenta una sfida aperta.

Transfer learning e domain adaptation. L'analisi della propagazione del poisoning in scenari di transfer learning, dove modelli pre-addestrati vengono fine-tuned su task specifici, richiede particolare attenzione. La valutazione dell'impatto del poisoning sul source model e la sua persistenza dopo il fine-tuning costituisce un aspetto critico per deployment reali.

Real-time monitoring e detection. Lo sviluppo di sistemi di monitoring continuo delle performance del modello in produzione, con detection di anomalie e drift, rappresenta un complemento essenziale alle tecniche di hardening. L'integrazione di approcci statistici (CUSUM, sequential hypothesis testing) con machine learning-based anomaly detection potrebbe fornire early warning di compromissione.

In conclusione, il presente lavoro ha dimostrato la vulnerabilità intrinseca dei modelli di malware detection agli attacchi di poisoning, evidenziando al contempo l'importanza di approcci di sicurezza by design. Le prospettive di ricerca delineate sottolineano la necessità di un approccio olistico alla sicurezza dei sistemi di machine learning, che integri tecniche difensive, verifica formale e monitoring continuo.

Appendici

A. Dettagli implementativi

Ambiente di sviluppo

- Python 3.8+
- PyTorch 2.0+ con supporto MPS (Apple Silicon), CUDA (NVIDIA) e CPU fallback
- Librerie scientifiche: NumPy 1.24+, Pandas 2.0+, Scikit-learn 1.3+
- Libreria EMBER: versione ufficiale da GitHub (endgameinc/ember)
- Hardware testato: Apple M-series (MPS acceleration), NVIDIA GPU (CUDA 11.8+)

Architettura del modello EmberMalwareNet

```
Layer             Output Shape        Parameters
-------------------------------------------------
fc1               [batch, 4000]       Input_dim × 4000 + 4000
BatchNorm1d       [batch, 4000]       8000
Dropout(0.5)      [batch, 4000]       0
ReLU              [batch, 4000]       0

fc2               [batch, 2000]       4000 × 2000 + 2000
BatchNorm1d       [batch, 2000]       4000
Dropout(0.5)      [batch, 2000]       0
ReLU              [batch, 2000]       0

fc3               [batch, 100]        2000 × 100 + 100
BatchNorm1d       [batch, 100]        200
Dropout(0.5)      [batch, 100]        0
ReLU              [batch, 100]        0

output            [batch, 1]          100 × 1 + 1
-------------------------------------------------
Total parameters (with input_dim=2381): ~17.1M
```

Funzione di loss e ottimizzazione

- Loss function: BCEWithLogitsLoss con pos_weight calcolato automaticamente dal class imbalance
- Optimizer: Adam (lr=0.001, weight_decay=1e-5)
- Scheduler: ReduceLROnPlateau (factor=0.5, patience=5)
- Early stopping: basato su F1-score (con monitoring del best model)

Feature selection pipeline

1. Rimozione feature costanti (varianza = 0)
2. Correlation-based filtering: calcolo matrice di correlazione di Pearson, rimozione feature con $|corr| > 0.98$, mantenimento feature con varianza maggiore
3. Mutual information selection: calcolo MI tra ogni feature e label, selezione top-k feature per MI score
4. Parallelizzazione: joblib con n_jobs=-1 per accelerazione
5. Sampling: utilizzo di subset di 10,000 campioni per accelerare il calcolo

Poisoning attack implementation

- Tipo: Label flipping (Malware $\rightarrow$ Benign)
- Selezione campioni: random sampling con seed=42 per riproducibilità
- Poison rate: 20% dei campioni malware del training set
- Validazione: nessuna modifica alle feature, solo flip delle label

Gaussian noise injection

- Target: tutti i layer di tipo Linear (weight tensors)
- Distribuzione: $\mathcal{N}(0, \sigma^2)$ con $\sigma = 0.0001$
- Applicazione: in-place addition su parametri del modello
- Tracking: statistiche pre/post perturbazione per ogni layer

Metriche e valutazione

- Threshold optimization: ricerca esaustiva del threshold che massimizza F1-score
- Confusion matrix: calcolo TN, FP, FN, TP per ogni esperimento
- ROC-AUC: Area under ROC curve per valutazione threshold-independent
- Optimal threshold: memorizzato e applicato per risultati finali

Gestione memoria e performance

- Batch processing: dimensione batch=256 (configurable)
- Data loading: TensorDataset e DataLoader per efficienza
- Mixed precision: supporto nativo PyTorch (se disponibile su hardware)
- Model checkpointing: salvataggio automatico del best model per F1-score

B. Codice e configurazioni

Struttura del progetto

```
project/
 main.py                   # Orchestrazione esperimenti
 preprocessing/
    data_loader.py        # Caricamento e feature selection
 attack/
    poisoning.py          # Poisoning e noise injection
 network/
    model.py              # Definizione EmberMalwareNet
    trainer.py            # Training loop e valutazione
 utils/
    metrics.py            # Calcolo metriche
    visualization.py      # Plotting e grafici
    io_utils.py           # I/O e serializzazione
 dataset/
    ember2018/            # Dataset vettorizzato
 results/
     experiment_results.json
     comparison_plot.png
```

Configurazione esperimento (ExperimentConfig)

```
DATA_DIR = "dataset/ember_dataset_2018_2"
EPOCHS = 10
BATCH_SIZE = 256
LEARNING_RATE = 0.001
DROPOUT_RATE = 0.5
WEIGHT_DECAY = 1e-5
POISON_RATE = 0.2
NOISE_STD = 0.0001
ATTACK_TYPE = 'Label Flipping (Malware->Benign)'
CORR_THRESHOLD = 0.98
MI_TOP_K = None  # o 17, 40 per feature selection
```

Esempio di esecuzione

```
# Vettorizzazione dataset (one-time)
python vectorize_ember.py dataset/ember2018

# Training completo con feature selection
python main.py dataset/ember2018

# Diagnostica dataset
python diagnostic.py dataset/ember2018
```

Output formato JSON

```
{
  "experiment_date": "2025-11-12T15:36:26.907176",
  "config": { /* parametri esperimento */ },
  "clean": {
    "test": {
      "accuracy": 0.73706,
      "precision": 0.6765,
      "recall": 0.9085,
      "f1_score": 0.7755,
      "auc_roc": 0.8100,
      "confusion_matrix": { /* TN, FP, FN, TP */ }
    }
  },
  "poisoned": { /* metriche modello avvelenato */ },
  "noisy": { /* metriche con rumore */ },
  "poisoning_info": { /* dettagli poisoning */ },
  "noise_statistics": { /* statistiche per layer */ }
}
```

Requisiti sistema

- RAM: minimo 16GB raccomandati (32GB per dataset completo)
- Storage: 10GB per dataset vettorizzato + 1GB per modelli
- GPU: opzionale ma raccomandato (10x speedup sul training)
- Tempo esecuzione: 2-5 minuti per epoca (dipende da hardware)

C. Dataset e link di riferimento

Dataset EMBER

- Nome completo: Endgame Malware BEnchmark for Research (EMBER)
- Versioni: EMBER 2017 (1.1M samples), EMBER 2018 (1.1M samples)
- Repository GitHub: `https://github.com/elastic/ember`
- Paper: Anderson & Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models", ArXiv:1804.04637, 2018
- Download: `https://ember.elastic.co/ember_dataset_2018_2.tar.bz2`
- Dimensione: 2.8GB compresso, 6.5GB estratto

Struttura dataset

- Training set: 900,000 samples (600k train, 200k validation, 100k unlabeled)

- Test set: 200,000 samples (100k benign, 100k malware)
- Features: 2,381 feature statiche estratte da PE header, import/export, section, byte histogram
- Label: 0 (benign), 1 (malware), -1 (unlabeled)
- Formato originale: JSONL compresso
- Formato vettorizzato: NumPy .dat files (X_train.dat, y_train.dat, X_test.dat, y_test.dat)

Feature categories EMBER

1. General file info (10 features): dimensioni, virtual size, presence of debug info
2. Header (62 features): Machine type, characteristics, timestamp
3. Imports (1280 features): histogram di librerie e funzioni importate
4. Exports (128 features): numero e tipologia di simboli esportati
5. Section (255 features): caratteristiche delle sezioni PE (size, entropy, permissions)
6. Datadirectories (30 features): presenza e dimensione delle directory
7. Byte histogram (256 features): distribuzione dei byte nel file
8. String info (104 features): numero e tipologia di stringhe printable

Librerie e tool

- PyTorch: `https://pytorch.org/`
- EMBER library: `pip install git+https://github.com/elastic/ember.git`
- Scikit-learn: `https://scikit-learn.org/`
- Pandas: `https://pandas.pydata.org/`
- Joblib: `https://joblib.readthedocs.io/`

Paper di riferimento

1. Anderson, H. S., & Roth, P. (2018). "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models". ArXiv:1804.04637.
2. Biggio, B., et al. (2012). "Poisoning Attacks against Support Vector Machines". ICML 2012.
3. Gu, T., et al. (2017). "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". NIPS 2017 Workshop.

4. Tramèr, F., et al. (2020). "Ensemble Adversarial Training: Attacks and Defenses". ICLR 2018.

5. Carlini, N., & Wagner, D. (2017). "Towards Evaluating the Robustness of Neural Networks". IEEE S&P 2017.

Repository e risorse aggiuntive

- Adversarial Robustness Toolbox (ART): `https://github.com/Trusted-AI/adversarial-robustness-t`
- CleverHans: `https://github.com/cleverhans-lab/cleverhans`
- Foolbox: `https://github.com/bethgelab/foolbox`
- SecML: `https://secml.gitlab.io/`

Standard e framework di sicurezza

- NIST AI Risk Management Framework: `https://www.nist.gov/itl/ai-risk-management-framework`
- MITRE ATLAS (Adversarial Threat Landscape): `https://atlas.mitre.org/`
- OWASP Machine Learning Security Top 10: `https://owasp.org/www-project-machine-learning-sec`

Riferimenti bibliografici

References

[1] G. Severi, J. Meyer, S. Coull, A. Oprea. *Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers*. arXiv:2003.01031, 2021.

[2] J. Frankle, M. Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. ICLR, 2019.

[3] R. Perdisci et al. *Misleading Worm Signature Generators Using Deliberate Noise Injection*. IEEE S&P, 2006.