



ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Dipartimento di Informatica - Scienza e Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

Intrusion Detection Systems adattivi con Data Compensation per il rilevamento di attacchi cyber

TESI DI LAUREA IN CYBERSECURITY

Relatore

Chiar.mo Prof.
Michele Colajanni

Laureando

Antonio Sarchione

Correlatrice

Dott.ssa
Isabella Marasco

Anno Accademico 2025/2026

Abstract

Nell'ambito della cybersecurity, gli Intrusion Detection Systems fondati su modelli di Machine Learning statici non riescono ad adattarsi alla continua evoluzione delle minacce cyber, rendendo necessari riaddestramenti periodici dall'elevato costo computazionale. Il Continual Learning si propone come soluzione efficace per superare tale limite, ma la sua adozione introduce due criticità significative, ovvero il Catastrophic Forgetting e il Concept Drift. In questa tesi viene proposto Dark Adaptive and Reactive Knowledge Experience Replay (DARKER), un framework di apprendimento incrementale innovativo che ottimizza la stabilità e la plasticità del modello. DARKER si basa su due componenti: un meccanismo adattivo che aggiorna dinamicamente il coefficiente di distillazione dei logits sulla base del forgetting osservato e una strategia di campionamento stratificato del buffer per tutelare la rappresentatività delle classi di attacco minoritarie. La metodologia è stata validata sperimentalmente sui due dataset ToN-IoT e CICIDS2017, confrontando le prestazioni di DARKER con le baseline DER e iCaRL. I risultati sperimentali evidenziano come DARKER registri un Forgetting inferiore rispetto agli approcci di confronto su entrambi i dataset e una maggiore robustezza contro i fallimenti sistemici di classificazione, preservando prestazioni superiori anche su Recall, Precision e F1-score per la quasi totalità della sequenza temporale.

Indice

Abstract

Introduzione

1 Continual Learning negli Intrusion Detection Systems

1.1	Intrusion Detection Systems	1
1.2	Continual Learning	6
1.3	Catastrophic Forgetting	9
1.3.1	Principali strategie	11
1.4	Concept Drift	12
1.5	Continual Learning come soluzione per IDS	16

2 Stato dell'Arte

2.1	Related Works	18
2.2	Principali metodologie	20
2.2.1	Elastic Weight Consolidation	20
2.2.2	Gradient Episodic Memory	22
2.2.3	Incremental Classifier and Representation Learning	25
2.2.4	Dark Experience Replay	28
2.3	Gap applicativo	32

3 Metodologia

3.1	Data Pre-processing	36
3.2	Architettura Rete Neurale	40
3.3	Model Implementation	44
3.3.1	Replay selettivo della memoria episodica	44
3.3.2	Distillazione adattiva dei logits	46

3.4	Model Training	48
3.5	Model Evaluation	52
3.5.1	Metriche di valutazione	52
3.5.2	Valutazione intra-task	56
4	Risultati	
4.1	ToN-IoT	59
4.2	CICIDS2017	69

Conclusione

Bibliografia

Elenco delle figure

1.1	Host-Based Intrusion Detection System	3
1.2	Network-Based Intrusion Detection System	4
1.3	Signature-Based Intrusion Detection System	5
1.4	Anomaly-Based Intrusion Detection System	5
1.5	Continual Learning Pipeline	7
2.1	Confronto tra EWC, normalizzazione $L2$ e baseline senza vincoli	21
2.2	Calcolo del gradiente in GEM e A-GEM	24
2.3	Comportamento interno di iCaRL	28
2.4	Comportamento interno di DER	29
2.5	Comportamento interno di DER++	30
3.1	Fasi dell'implementazione di DARKER	36
3.2	Dizionario per la standardizzazione del dataset ToN-IoT	38
3.3	Colonne da rimuovere nel pre-processing del dataset ToN-IoT	39
4.1	Accuracy media per task su ToN-IoT	60
4.2	Accuracy media per classi su ToN-IoT	60
4.3	Forgetting medio per task su ToN-IoT	61
4.4	Forgetting medio per classi su ToN-IoT	62
4.5	Trade-off tra Accuracy e Forgetting su ToN-IoT	63
4.6	Recall media per task su ToN-IoT	63
4.7	Recall media per classi su ToN-IoT	64
4.8	Precision media per task su ToN-IoT	65
4.9	Precision media per classi su ToN-IoT	65
4.10	F1-score medio per task su ToN-IoT	66
4.11	F1-score medio per classi su ToN-IoT	66
4.12	DARKER Confusion Matrix su ToN-IoT	67

4.13	DER Confusion Matrix su ToN-IoT	68
4.14	iCaRL Confusion Matrix su ToN-IoT	69
4.15	Accuracy media per task su CICIDS2017	70
4.16	Accuracy media per classi su CICIDS2017	71
4.17	Forgetting medio per task su CICIDS2017	72
4.18	Forgetting medio per classi su CICIDS2017	72
4.19	Trade-off tra Accuracy e Forgetting su CICIDS2017	73
4.20	Recall media per task su CICIDS2017	74
4.21	Recall media per classi su CICIDS2017	74
4.22	Precision media per task su CICIDS2017	75
4.23	Precision media per classi su CICIDS2017	75
4.24	F1-score medio per task su CICIDS2017	76
4.25	F1-score medio per classi su CICIDS2017	77
4.26	DARKER Confusion Matrix su CICIDS2017	78
4.27	DER Confusion Matrix su CICIDS2017	79
4.28	iCaRL Confusion Matrix su CICIDS2017	80

Elenco delle formule

1.1	Concept Drift	12
2.1	EWC Loss Function	20
2.2	GEM Loss Function Gradient	23
2.3	iCaRL NME prototype	25
2.4	iCaRL classification	25
2.5	iCaRL representation	26
2.6	iCaRL Loss Function	27
2.7	DER Loss Function	29
2.8	DER++ Loss Function	30
3.1	DARKER Loss Function	47
3.2	Termine di distillazione in DARKER	47

3.3	Calcolo adattivo del coefficiente di distillazione	48
3.4	Accuracy media	52
3.5	Forgetting relativo	53
3.6	Forgetting medio	53
3.7	Recall	53
3.8	Precision	54
3.9	F1-score	55
3.10	Confusion Matrix multi-classe	55
3.11	Confusion Matrix binaria	55
3.12	True Positives	56
3.13	False Positives	56
3.14	False Negatives	56
3.15	True Negatives	56
3.16	Funzione per il calcolo del Forgetting medio per il task i-esimo	57

Introduzione

La continua evoluzione delle minacce informatiche evidenzia come la staticità degli attuali sistemi di difesa costituisca il loro principale punto debole. La capacità di un Intrusion Detection System di adattarsi in tempo reale rappresenta un requisito fondamentale per evitare l'obsolescenza, soprattutto nei contesti in cui un'azione repentina può essere necessaria.

Al centro di questo problema c'è il fenomeno del Concept Drift, ovvero il cambiamento nelle proprietà statistiche del traffico di rete nel tempo, per effetto sia della comparsa di nuove tipologie di attacco sia della normale variazione del traffico legittimo. Per superare la rigidità dei riaddestramenti periodici, viene adottato il Continual Learning nei sistemi di difesa e monitoraggio, per permettere ad un sistema di imparare da un flusso continuo di dati, integrando nuove conoscenze senza dover riaddestrare da zero. Tuttavia, l'adozione di questa metodologia porta con sé il fenomeno del Catastrophic Forgetting, ovvero la tendenza dei modelli a sovrascrivere le informazioni precedentemente apprese non appena ne arrivano di nuove. Nonostante la letteratura recente abbia proposto alcune strategie per mitigare questo problema, molte di esse si concentrano in particolar modo sulla conservazione della memoria storica, ottenendo modelli che risultano o troppo rigidi per adattarsi ai repentini cambiamenti del traffico o, al contrario, troppo instabili per garantire una protezione sul lungo periodo.

L'obiettivo di questa tesi è di contribuire al superamento di tale divario, introducendo *Dark Adaptive and Reactive Knowledge Experience Replay* (DARKER), una nuova metodologia di apprendimento incrementale, il cui principale contributo scientifico risiede nell'introduzione di nuove strategie di aggiornamento e campionamento selettivo del buffer, progettate per preservare la rappresentatività delle classi nel tempo, e nello sviluppo di un algoritmo originale per la determinazione dinamica del coefficiente di distillazione dei logits, adattato in funzione del Forgetting osservato sul task corrente. L'integrazione di queste due componenti realizza una strategia di Data Compensation ideata per dimostrare come una gestione consapevole della memoria storica possa prevenire il Catastrophic Forgetting e contemporaneamente agire come

un meccanismo attivo di compensazione, migliorando la reattività e l'accuratezza dei sistemi IDS verso le minacce emergenti. Gli esperimenti realizzati in questo lavoro sono stati condotti su due dataset di riferimento nel panorama della sicurezza informatica e i risultati ottenuti hanno dimostrato come la metodologia proposta ottenga prestazioni superiori lungo la sequenza temporale rispetto alle principali soluzioni presenti in letteratura.

La tesi verterà su quattro capitoli. Nel Capitolo 1 vengono introdotti i concetti fondamentali degli IDS e del Continual Learning, approfondendo le criticità cruciali legate al Catastrophic Forgetting e al Concept Drift in ambienti operativi. Nel Capitolo 2 si esaminano i lavori correlati e si analizzano i fondamenti metodologici delle principali soluzioni presenti in letteratura, evidenziandone il gap applicativo. Nel Capitolo 3 viene presentata nel dettaglio l'implementazione del modello proposto, illustrandone i criteri di progettazione e di valutazione. Nel Capitolo 4, infine, si effettua un'analisi approfondita dei risultati sperimentali ottenuti sui due dataset, confrontando l'efficacia di DARKER con i principali approcci di riferimento in letteratura.

Capitolo 1

Continual Learning negli Intrusion Detection Systems

1.1 Intrusion Detection Systems

La progressiva evoluzione delle infrastrutture moderne ha trasformato i sistemi informatici in componenti fondamentali di qualsiasi organizzazione. Tuttavia, tale dipendenza dalla tecnologia ha ampliato in modo proporzionale la superficie esposta ad attività ostili, favorendo una profonda evoluzione del panorama delle minacce informatiche, le quali, da fenomeni sporadici, si sono trasformate in operazioni strutturate condotte da organizzazioni specializzate[1]. La crescente complessità di questo scenario ha reso strutturalmente inadeguati gli approcci basati su regole statiche, poiché i comportamenti ostili sono progettati per mimetizzarsi nel traffico legittimo e per adattarsi alle contromisure implementate. È in questo contesto che gli Intrusion Detection Systems (IDS) si sono affermati come componente cruciale delle infrastrutture di sicurezza, finalizzata all'identificazione di attività malevole, utilizzi impropri o accessi non autorizzati. Il loro obiettivo primario consiste nel rilevare tempestivamente comportamenti riconducibili ad intrusioni, a prescindere dalla loro origine, mediante il monitoraggio continuo degli eventi generati dai sistemi e la successiva analisi dei pattern comportamentali. All'interno delle moderne architetture di rete, gli IDS agiscono come presidi a tutela della triade CIA, costituita da *Confidentiality*, *Integrity*, *Availability*, che rappresenta il paradigma di riferimento per la valutazione della sicurezza dei sistemi informativi[2][1]. Nello specifico, l'adozione di un IDS non si limita alla salvaguardia della riservatezza o alla sola tutela dell'integrità informativa, ma ricopre un ruolo cruciale nel garantire anche la disponibilità del servizio, identificando in

tempo reale anomalie riconducibili ad attacchi, come ad esempio *Denial of Service (DoS)*, potenzialmente in grado di compromettere o bloccare l'operatività dell'intera infrastruttura.

Sotto il profilo architetturale, un IDS può essere modellato come un sistema hardware o software adibito all'automazione del processo di rilevamento, articolato attorno ad un ciclo operativo che comprende la raccolta di informazioni dai pacchetti di rete, il loro pre-processing finalizzato all'estrazione delle feature più significative, l'analisi mediante un motore di rilevamento e la successiva generazione di alert in presenza di eventi sospetti o di deviazioni significative rispetto al comportamento atteso. Uno degli aspetti principali nella progettazione di un IDS riguarda la definizione delle sue proprietà desiderabili. In particolare, un sistema efficace deve essere in grado di operare in maniera continua e autonoma, riducendo al minimo la necessità di intervento umano e garantendo al contempo elevata affidabilità operativa in contesti operativi prolungati. La sostenibilità di un simile ciclo operativo esige che la struttura di un IDS sia configurata secondo logiche di modularità e disaccoppiamento dei componenti, affinché sia garantita una solida stabilità del sistema. Proprio per questa ragione, gli IDS moderni devono incorporare capacità di adattamento dinamico, col fine di modellare l'evoluzione dei pattern di utilizzo in contesti reali, nei quali il traffico di rete varia nel tempo come conseguenza dell'introduzione di nuove applicazioni e di aggiornamenti infrastrutturali[2][3].

La valutazione prestazionale di un IDS ruota attorno alla precisione del rilevamento. Un sistema efficace deve bilanciare la riduzione dei falsi positivi e dei falsi negativi, poiché un'elevata incidenza di allarmi non rilevanti compromette l'affidabilità operativa e incrementa il carico di analisi da parte degli operatori umani, mentre la mancata identificazione di attacchi effettivi riduce drasticamente l'efficacia complessiva del sistema. Il trade-off tra sensibilità e specificità si conferma, dunque, uno dei problemi centrali nella progettazione e ottimizzazione delle moderne soluzioni di rilevamento delle minacce[2].

La costante evoluzione delle minacce informatiche e la crescente complessità dei contesti applicativi hanno imposto una necessaria diversificazione tecnologica degli IDS. Ogni diverso sistema è il risultato di una specifica scelta progettuale volta a rispondere a precise esigenze di monitoraggio e questa eterogeneità viene di solito organizzata in base a variabili cruciali, ovvero il perimetro di osservazione e la tipologia di rilevamento. In letteratura, la distinzione tra *Host-Based IDS (HIDS)* e *Network-Based IDS (NIDS)*, dal punto di vista strutturale e quindi del perimetro di osservazione, riflette la necessità di scegliere tra la profondità analitica degli eventi locali al singolo endpoint e la prospettiva sistemica sulle interazioni di rete dell'intera

organizzazione[4]. Gli HIDS, la cui architettura è schematizzata nella Figura 1.1, questa tipologia opera direttamente sui dispositivi da proteggere e monitora elementi interni del sistema come, ad esempio, i file, i processi, le modifiche ai permessi e le chiamate al kernel dei sistemi operativi. Il loro sviluppo nasce dalla necessità di rilevare attacchi che non sono subito visibili a livello di traffico di rete; il loro principale vantaggio consiste nell'elevata visibilità locale, che spiega il motivo del loro ampio impiego nelle attività di analisi forense. Tuttavia, questi sistemi di monitoraggio presentano diversi limiti, tra cui il significativo consumo di risorse computazionali, la difficoltà di gestione in infrastrutture di grandi dimensioni e la possibilità che un attaccante con privilegi amministrativi possa disabilitarli.

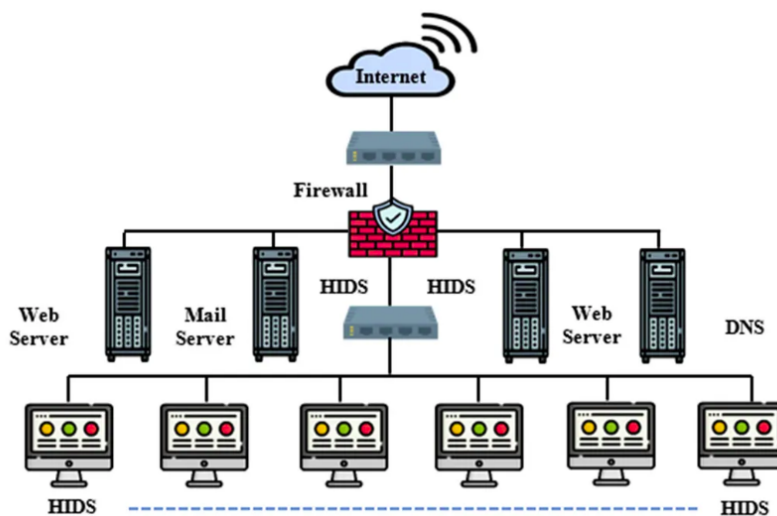


Figura 1.1: Host-Based Intrusion Detection System

L'altra categoria è costituita dai NIDS, la cui architettura è delineata nella Figura 1.2. Essa nasce dalla necessità di analizzare il traffico di rete transitante attraverso i vari nodi delle infrastrutture, come router o gateway. Questi sistemi monitorano pacchetti di rete e ricostruiscono le sessioni di comunicazione per identificare le varie tipologie di minacce e il trasferimento non autorizzato di dati sensibili. Il principale vantaggio consiste nella capacità di effettuare un monitoraggio centralizzato della rete di un'intera infrastruttura senza alcuna installazione di agenti sui singoli host. Nonostante ciò, anche i NIDS presentano importanti criticità legate al sempre più standardizzato utilizzo del traffico cifrato mediante appositi protocolli, che limitano in maniera concreta la possibilità di ispezionare il contenuto dei pacchetti di rete. Oltre a questo limite, emergono problematiche di scalabilità legate al fatto che le reti moderne producono enormi volumi di traffico che devono essere elaborati in un tempo quasi nullo[4].

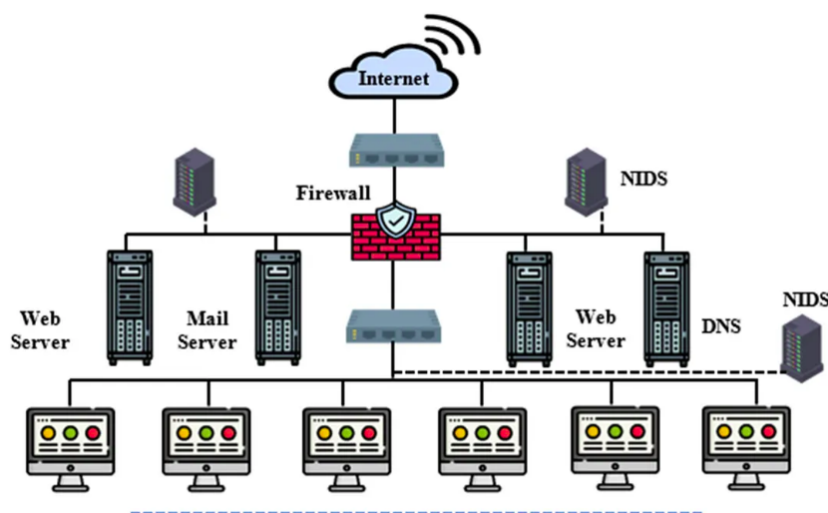


Figura 1.2: Network-Based Intrusion Detection System

Un'ulteriore diversificazione per i sistemi IDS riguarda la logica utilizzata per il rilevamento delle minacce, per cui vengono distinte differenti tipologie di sistemi: i *Signature-Based IDS* e gli *Anomaly-Based IDS*. La prima categoria permette di identificare le intrusioni confrontando il traffico osservato con un database di firme associate ad attacchi precedentemente noti. Le firme utilizzate per il rilevamento possono contenere sequenze di byte malevoli o anche pattern comportamentali che consentano di identificare una specifica minaccia. Questa tipologia di sistemi garantisce un'elevata accuratezza e un tasso di falsi positivi molto basso soprattutto nel rilevamento di minacce conosciute. Nonostante ciò, il limite principale di questi sistemi consiste nell'incapacità di rilevare attacchi "zero-day" o varianti sconosciute, proprio perché tali sistemi dipendono in primo luogo dalla disponibilità preventiva delle firme[5]. I *Signature-Based IDS*, la cui struttura è riprodotta nella Figura 1.3, richiedono aggiornamenti continui del database e sono vulnerabili a tecniche evasive che rendono il codice malevolo difficile da analizzare o rilevare, come ad esempio accade con la tecnica dell'*obfuscation*.

Come alternativa ai precedenti e per superarne i limiti sono stati introdotti gli *Anomaly-Based IDS*, la cui struttura è visibile nella Figura 1.4. Essi sono basati sull'identificazione di deviazioni rispetto al comportamento normale della rete o del sistema. Questo metodo si fonda sulla definizione di una baseline del comportamento legittimo del traffico, classificando come sospetta ogni attività che manifesti una deviazione rilevante rispetto al modello di riferimento. Grazie alla loro capacità di prescindere dalla conoscenza preventiva di una specifica minaccia, i sistemi di rilevamento che utilizzano questa tecnica sono considerati una delle frontiere più avanzate della ricerca, riuscendo ad essere efficienti anche nell'identificazione di vulnerabilità "zero-day"[4].

Una criticità rilevante riguarda il numero di falsi positivi, che, in alcuni casi, può diventare fuorviante, poiché questi sistemi possono considerare sospette anche variazioni legittime del traffico, generando così un'elevata quantità di alert superflui. Un altro problema riguarda il punto di vista computazionale, dato che questi sistemi richiedono elevate risorse sia durante la fase di addestramento che durante quella di inferenza.

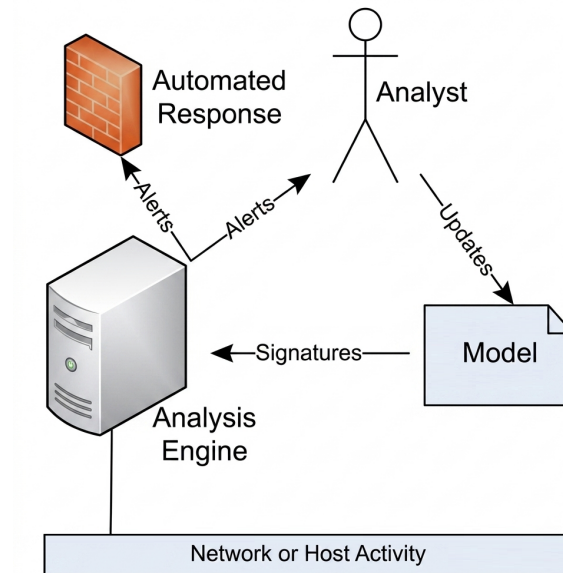


Figura 1.3: Signature-Based Intrusion Detection System

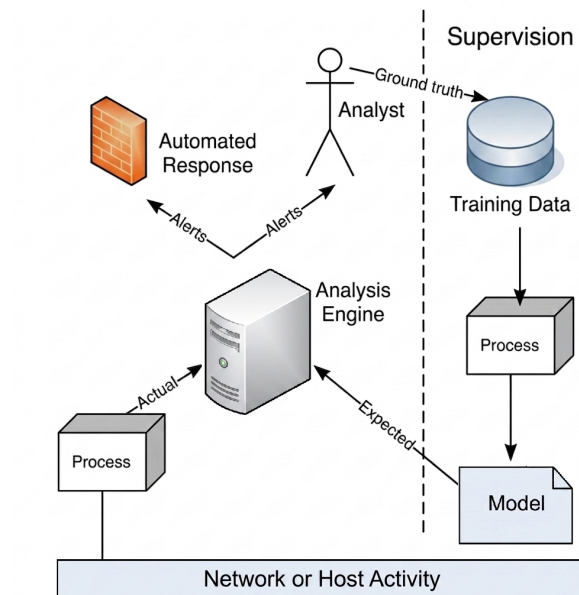


Figura 1.4: Anomaly-Based Intrusion Detection System

L'ultima categoria è quella dei *Specification-Based IDS*, nei quali il comportamento corretto del sistema viene definito attraverso regole formali dettate dagli amministratori di rete. Questa

tipologia viene per lo più adottata in ambienti industriali, in cui i comportamenti del traffico di rete sono molto più prevedibili rispetto agli ambienti dinamici. Il vantaggio di questa tecnica consiste nella riduzione dei falsi positivi e nella non dipendenza da firme predefinite; mentre lo svantaggio principale sta nella definizione manuale stessa che richiede una profonda conoscenza dell'infrastruttura monitorata e che rende questo tipo di approccio non scalabile in sistemi complessi[6].

I sistemi analizzati in questa sezione costituiscono diverse tipologie di IDS statici, in cui il modello viene costruito a partire da dati storici e rimane immutato durante la sua operatività. La loro stazionarietà va in contrasto con la natura dinamica del traffico di rete: la distribuzione dei dati di rete e le sue caratteristiche variano nel tempo per effetto di aggiornamenti dei protocolli di rete e della continua evoluzione delle tecniche utilizzate per la realizzazione delle minacce informatiche. Negli IDS statici, il riaddestramento periodico del modello sull'intero dataset aggiornato risulta spesso impraticabile in contesti operativi reali perché richiede la disponibilità di dati sempre aggiornati, comporta costi computazionali considerevoli e impone delle finestre temporali in cui il sistema è indisponibile per essere aggiornato. L'onere computazionale e gestionale legato all'aggiornamento costante dei database per gli IDS statici, unito all'incapacità di questi sistemi di adattarsi autonomamente alle normali mutazioni del traffico, evidenzia l'inadeguatezza delle soluzioni tradizionali all'interno di infrastrutture di rete moderne, caratterizzate da forte dinamismo ed elevata instabilità. Questi limiti strutturali degli IDS statici hanno orientato la ricerca in letteratura verso la realizzazione di sistemi di rilevamento basati su Continual Learning, in grado di aggiornarsi in tempo reale all'arrivo di nuove minacce, senza dover rieseguire l'addestramento sull'intero dataset e senza annullare le conoscenze acquisite in precedenza.

1.2 Continual Learning

Il Continual Learning [7] è un approccio metodologico che prevede l'addestramento continuo nel tempo di un modello, tramite l'integrazione di nuove conoscenze e la conservazione dell'esperienza pregressa. Essa costituisce una branca del Machine Learning, ma, a differenza di quest'ultimo, basato sul presupposto di catturare e analizzare una grande distribuzione statica dei dati, il Continual Learning affronta scenari in cui i dati evolvono nel tempo e presentano distribuzioni dinamiche. Questo concetto prende ispirazione dalla capacità degli esseri umani

di acquisire, mantenere e trasferire conoscenze nel tempo, adattandosi a contesti mutevoli, senza perdere le abilità precedentemente acquisite. La principale peculiarità di questa metodologia risiede nella non stazionarietà delle distribuzioni dei dati. Nel contesto del Continual Learning, raffigurato nella Figura 1.5, infatti, l'obiettivo si sposta dall'addestramento su un dataset statico e completamente noto a priori alla progettazione di modelli capaci di adattarsi all'arrivo di nuove informazioni, integrandole nel tempo senza compromettere quelle già acquisite.

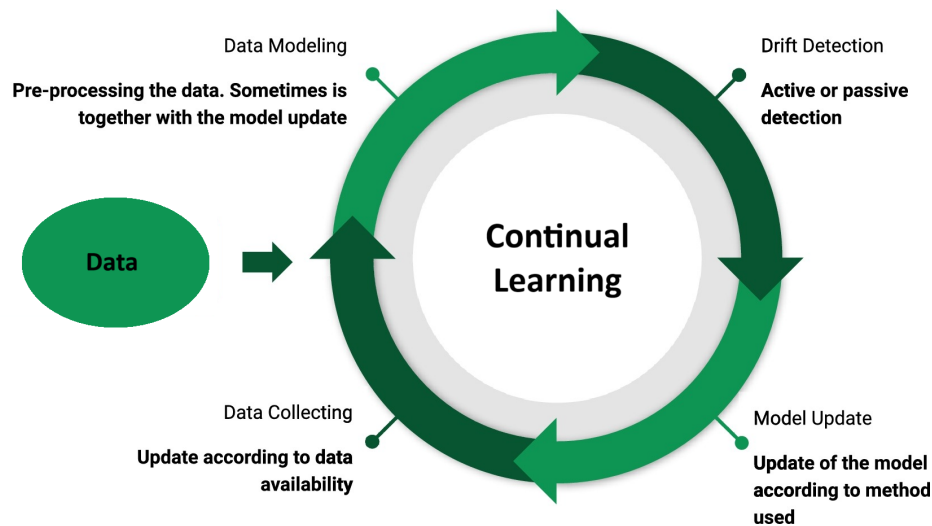


Figura 1.5: Continual Learning Pipeline

Dunque, nel Continual Learning il modello viene esposto ad una sequenza di task, ciascuno dei quali è associato ad un dataset e ad una specifica distribuzione dei dati. Ad ogni passaggio, il sistema ha il compito di apprendere il task corrente aggiornando i propri parametri, preservando le conoscenze acquisite nei task precedenti. Al contempo, la valutazione avviene sull'intero insieme dei task osservati fino a quel momento, con l'obiettivo di minimizzare la perdita media complessiva lungo l'intera sequenza. Questo obiettivo diventa particolarmente complesso da raggiungere nel contesto di applicazioni reali, dove l'accesso ai dati relativi ai task precedenti risulta spesso limitato, parziale o addirittura del tutto assente: non si tratta di una mancanza occasionale, ma di una condizione strutturale che accompagna gran parte dei sistemi in produzione.

La limitazione deriva principalmente da due tipologie di vincoli, differenti ma legati fra loro. Da un lato, vincoli di natura implementativa, legati alla necessità di garantire scalabilità e gestire in modo sostenibile le risorse di memoria, soprattutto quando il volume dei dati cresce in modo significativo nel tempo; dall'altro, vincoli di natura normativa e di privacy, che entrano in gioco quando il sistema opera su dati sensibili, imponendo restrizioni severe sulla loro conservazione,

consultazione e riutilizzo. In entrambi i casi, il modello deve essere in grado di adattarsi e operare in modo efficace anche in assenza di uno storico completo.

In letteratura, il Continual Learning viene formalizzato attraverso differenti scenari, definiti in funzione della natura dell'evoluzione dei dati e del livello di informazione[7][8]:

- **Task-Incremental Learning:** in questo scenario, il modello viene addestrato su una sequenza di task ben identificabili dalla presenza di etichette. L'elemento caratterizzante è che, durante la fase di test, l'identità del task corrente è nota al modello, il che costituisce una semplificazione significativa rispetto agli altri scenari. Questa informazione aggiuntiva consente al modello di specializzarsi su ciascun task in modo separato, riducendo il rischio di interferenza tra le conoscenze acquisite in momenti diversi. L'obiettivo principale non si limita quindi alla sola preservazione delle conoscenze passate, ma include anche un'efficiente condivisione delle rappresentazioni interne tra task differenti, in modo da sfruttare eventuali similarità e favorire un trasferimento della conoscenza. Il Task-Incremental Learning, nonostante sia lo scenario concettualmente più facile tra quelli considerati, pone comunque importanti sfide nella progettazione di architetture capaci di bilanciare specializzazione e generalizzazione.
- **Domain-Incremental Learning:** in questo scenario, la struttura del problema rimane invariata nel tempo e ciò che evolve è la distribuzione dei dati di input. Nel Domain-Incremental Learning le classi da predire sono le stesse, il formato dell'output non cambia e il modello effettua le stesse rilevazioni ma in contesti sempre diversi. Il modello non dispone di informazioni esplicite sull'identità del task corrente durante l'addestramento e deve sviluppare una capacità di adattamento interna. La principale difficoltà non risiede tanto nell'espansione della conoscenza, quanto nel mantenimento della stabilità delle prestazioni di fronte a variazioni della distribuzione, che possono essere sia gradualmente che piuttosto brusche. Questo scenario è considerato molto rilevante in applicazioni industriali, dove le condizioni operative tendono a mutare nel tempo in modo continuo e imprevedibile.
- **Class-Incremental Learning:** il modello viene esposto ad una sequenza di task in cui nuove classi vengono introdotte gradualmente nel tempo, espandendo progressivamente lo spazio delle possibili etichette. La complessità dello scenario nasce dalla combinazione di due fattori che si sovrappongono e si amplificano a vicenda, ovvero la necessità di

discriminare correttamente nonostante il numero crescente di classi e l'assenza di qualsiasi informazione esplicita sull'identità del task durante la fase di addestramento.

Quest'ultimo scenario è in grado di riflettere la reale evoluzione delle minacce in un'infrastruttura di rete, dove nuovi attacchi appaiono senza etichette predefinite. Tra i tre scenari, il Class-Incremental Learning è quello che rispecchia meglio la realtà nell'ambito della cybersecurity e, più nello specifico, nell'operatività degli IDS. In questi sistemi, infatti, la continua evoluzione degli attacchi, senza una chiara separazione tra generazioni successive di minacce, comporta una naturale espansione dello spazio delle classi nel tempo, a cui solo quest'ultimo scenario può far fronte. La valutazione dei modelli di Continual Learning richiede metriche specifiche, in grado di descrivere in modo accurato il comportamento del modello lungo l'intero processo di acquisizione di informazioni. A differenza degli scenari tradizionali, non basta misurare la performance su un singolo insieme di test, ma bisogna capire come quelle performance vengano mantenute nel tempo e a quale costo.

1.3 Catastrophic Forgetting

A differenza degli scenari tradizionali, in cui il modello viene addestrato una sola volta su un dataset statico, nei contesti reali i dati tendono ad arrivare in sequenza e a cambiare nel tempo e questo dinamismo implica che il modello debba essere aggiornato all'arrivo di nuove informazioni. L'integrazione di nuove informazioni, spesso caratterizzate da distribuzioni differenti rispetto alle precedenti, può interferire con le conoscenze già apprese e a causa di questo problema il modello tende a perdere col tempo la capacità di riconoscere correttamente i dati passati e quindi di conservare la memoria storica. Questo fenomeno, definito come Catastrophic Forgetting, comporta un rapido deterioramento delle prestazioni del modello e, in casi più critici, la completa sostituzione delle informazioni già acquisite con quelle più recenti. Esso rappresenta una conseguenza diretta dell'apprendimento sequenziale: un sistema che aggiorna i propri parametri un task alla volta non può garantire coerenza con ciò che ha visto prima[7].

Il problema principale che genera il Catastrophic Forgetting risiede nel modo in cui il modello codifica le informazioni e aggiorna i parametri durante l'apprendimento. In pratica, quando un modello viene addestrato su nuovi task, l'ottimizzazione dei pesi tende a privilegiare le informazioni più recenti, modificando le rappresentazioni interne che in precedenza erano ritenute importanti per i task già appresi e sovrascrivendo i parametri precedenti per minimizzare la

funzione di perdita corrente, ovvero la metrica matematica fondamentale utilizzata per quantificare l'errore del modello nel prevedere i risultati corretti. Un ulteriore fattore critico per il modello è dato dalla non stazionarietà dei dati. Nel Continual Learning i dati cambiano di continuo e le classi aumentano nel tempo e questo altera lo spazio rappresentativo appreso dal modello. In assenza di strategie che permettano di non perdere del tutto la memoria storica, il modello tende ad adattarsi alla distribuzione più recente (*distribution shift*), modificando le proprie rappresentazioni interne e riducendo gradualmente la capacità di descrivere senza errori i dati osservati in precedenza.

Questo comportamento mette in evidenza la difficoltà nel bilanciare l'adattamento a nuove informazioni con la necessità di preservare le conoscenze già acquisite: la stabilità assoluta equivale ad una sorta di immobilità cognitiva del modello, mentre la plasticità assoluta produce una memoria fragile e instabile. Questa difficoltà porta al cosiddetto *stability-plasticity dilemma*, cioè la necessità di trovare un equilibrio tra la capacità del modello di apprendere nuove classi e quella di preservare la conoscenza già acquisita[9]. Da un lato, un sistema deve conservare un grado di stabilità tale da mantenere nel tempo le conoscenze già acquisite; dall'altro, deve anche restare abbastanza plastico da integrare nuove informazioni senza irrigidirsi. Questo equilibrio è però fragile per definizione a causa del modo in cui le reti neurali rappresentano la conoscenza[10][11]. Nei modelli, infatti, le informazioni vengono distribuite attraverso insiemi di parametri condivisi durante l'apprendimento che vengono modificati per apprendere nuove informazioni. La complessità di questo dilemma deriva proprio da questa natura distribuita della rappresentazione delle informazioni, che da un lato favorisce la generalizzazione della rete e dall'altro aumenta anche la sensibilità del modello agli aggiornamenti successivi[7]. Per questo motivo, la ricerca sul Continual Learning si concentra molto sul controllo del grado di plasticità del sistema, dato che la riduzione degli aggiornamenti più distruttivi permette una maggiore preservazione della capacità adattiva della rete.

Tra le principali strategie considerate per mitigare il Catastrophic Forgetting, il *Joint Training*, ovvero il riaddestramento del modello sull'unione di tutti i dati osservati fino al task corrente, rappresenta il metodo ottimale, poiché garantisce l'accesso alle distribuzioni passate durante ogni fase di aggiornamento. Tuttavia, questo approccio risulta incompatibile con i requisiti operativi del Continual Learning: il costo computazionale e di memorizzazione aumenta al crescere del numero dei task osservati in sequenza, rendendo questa tipologia di addestramento impraticabile in questo ambito[7][12]. A questi vincoli si aggiungono restrizioni di natura

normativa che limitano la conservazione e il riutilizzo dei dati già processati in numerosi contesti applicativi. Questi limiti strutturali hanno orientato la ricerca verso lo sviluppo di metodologie alternative, in cui il modello, in assenza dello storico delle distribuzioni passate, deve aggiornare i propri parametri facendo riferimento solo ai campioni più recenti[13].

1.3.1 Principali strategie

La letteratura classifica le principali strategie per evitare che i modelli dimentichino ciò che hanno appreso secondo tre approcci e ognuno di essi interviene con tecniche diverse, ma con l'obiettivo comune di garantire una memoria a lungo termine alla rete neurale[7][13]:

- **Replay Methods:** è la metodologia più utilizzata e consolidata nel Continual Learning. Questi metodi operano attraverso la reintegrazione di informazioni relative a distribuzioni già osservate durante la fase di aggiornamento sui nuovi task. L'idea di fondo è quella di ancorare in modo incrementale l'ottimizzazione ad una porzione della distribuzione già osservata, così da mantenere la capacità discriminativa sulle conoscenze pregresse e non destabilizzare le rappresentazioni interne[12][14]. In letteratura si distinguono due principali tipologie di replay, tra cui quella più classica, legata alla memoria episodica contenente un sottoinsieme di esempi reali provenienti dai task passati, che vengono poi riutilizzati insieme ai nuovi durante il training, e una più recente, legata al replay generativo, dove i dati del passato non vengono conservati in chiaro, ma vengono ricreati in forma sintetica, permettendo di rispettare i vincoli legati alla privacy. Nel tempo, il replay si è arricchito di varianti che oltre a riutilizzare esempi cercano anche di preservare in modo più fine la conoscenza del modello, come, ad esempio, avviene per il *Gradient Episodic Memory* o per il *Dark Experience Replay*, che operano applicando ulteriori operazioni oltre alla conservazione della memoria storica per tenere intatti sia il contenuto informativo sia la struttura della conoscenza che il modello aveva sviluppato.
- **Regularization Methods:** è una metodologia che si basa sull'integrazione di un vincolo nella funzione di perdita per penalizzare in modo selettivo i pesi ritenuti importanti per i task precedenti. Questa tecnica permette al modello di evolvere verso nuovi task preservando la conoscenza pregressa senza dover archiviare esempi dei dati passati, poiché la gestione del bilanciamento tra plasticità e stabilità viene gestita agendo direttamente sul processo di ottimizzazione dei parametri[12][13]. All'interno di questa categoria,

emerge una distinzione in base all'approccio utilizzato: regolarizzazione parametrica e regolarizzazione funzionale. Nel primo caso, l'obiettivo è la protezione dei pesi della rete che avviene tramite l'applicazione di una penalità che impedisce variazioni drastiche ai pesi più importanti[15]. Con l'altro approccio, invece, vengono applicati dei vincoli sul comportamento del modello, ovvero sulle sue predizioni. In questo modo, la conoscenza pregressa viene mantenuta forzando il modello a mantenere output coerenti con le funzioni apprese in precedenza, anche in assenza dei dati originali. Dunque, la regolarizzazione effettuata da questi metodi si può intendere quasi come una strategia intenta a proteggere la conoscenza pregressa, intervenendo sul processo di aggiornamento del modello: agendo sui pesi, sulle risposte del modello o su una loro combinazione, in alcuni casi[15][12]. Tuttavia, l'efficacia di questi metodi dipende dalla precisione dei criteri di stima dell'importanza dei pesi e dalla qualità della conoscenza distillata.

- **Parameter Isolation Methods:** è una metodologia che affronta il problema limitando la condivisione dei parametri tra i diversi task e isolandone le conoscenze. Ad ogni task viene assegnata una parte dedicata dell'architettura evitando che i parametri vengano sovrascritti. A differenza degli altri metodi, la memoria non viene più difesa in anticipo ma è protetta strutturalmente, impedendo che i nuovi dati sovrascrivano i pesi già ottimizzati[16]. Un esempio che adotta questa strategia è dato dalle *Progressive Neural Networks*, dove per ogni nuovo task viene aggiunta una colonna alla rete lasciando intatte quelle vecchie. Questa strategia è considerata molto efficace, soprattutto in contesti in cui l'identità del task è nota; tuttavia, l'inevitabile crescita della complessità strutturale rischia di compromettere la scalabilità del sistema, soprattutto in scenari di apprendimento continuo caratterizzati da una lunga operatività del modello.

1.4 Concept Drift

La definizione di Concept Drift è stata introdotta per descrivere il fenomeno in cui il concetto da imparare muta nel tempo, rendendo progressivamente inadeguato qualunque modello addestrato su dati storici. Esso viene definito come una variazione della distribuzione tra input e output nel tempo:

$$\exists t : P_t(X, y) \neq P_{t+1}(X, y), \quad (1.1)$$

dove $P_t(X, y)$ è la distribuzione di probabilità che governa contemporaneamente le feature X e il target y al tempo t , ovvero la legge che spiega come gli input e le loro etichette si relazionano in quel momento[17]. La formula indica come esista almeno un istante t in cui la distribuzione che descrive i dati cambia rispetto all'istante successivo e, quando questo avviene, il modello addestrato su P_t sta applicando regole apprese in un contesto che non esiste più. Dato che $P(X, y) = P(X) \cdot P(y | X)$, la variazione rispetto all'istante precedente può essere causata da due componenti distinte: se cambia $P(X)$ mentre $P(y | X)$ resta stabile, allora la distribuzione delle feature si allontana ma il confine decisionale del classificatore rimane corretto e viene generato il cosiddetto "virtual drift"; invece, se avviene il contrario, allora cambia la relazione fondamentale tra input e output e il modello diventa sbagliato perché la realtà rappresentata non esiste più, generando il "real drift".

In ambito IDS, questa distinzione tra i due diversi tipi di drift si ritrova in scenari pratici: se il drift è causato da un aggiornamento infrastrutturale che modifica la composizione del traffico di rete senza introdurre nuove minacce allora si parla di virtual drift, mentre se il drift è causato dall'evoluzione degli attacchi informatici si parla di real drift. Nonostante le due cause possano normalmente sovrapporsi, il real drift resta l'argomento principale su cui la maggior parte degli studi in questo ambito e anche il presente elaborato si focalizzano, poiché genera conseguenze dirette sulla capacità di rilevazione del modello.

La letteratura distingue solitamente differenti tipologie di Concept Drift in base alla velocità e alla modalità con cui la distribuzione cambia nel tempo[17]. La distinzione è sottile ma fondamentale perché ciascuna tipologia reagisce in modo diverso ai metodi di rilevazione e se si sceglie lo strumento sbagliato si rischia quasi con certezza di avere prestazioni insoddisfacenti o costi computazionali troppo elevati. La tipologia più evidente è il *sudden drift*, per cui la distribuzione cambia in modo brusco e definitivo in un singolo istante temporale, quello che in letteratura viene chiamato "change point"[18]. Prima di quel momento i campioni rilevati appartengono ad un concetto ed un istante dopo ad un altro, senza nessuna fase di transizione. L'effetto che si ha in un modello è un crollo immediato delle prestazioni, generando un segnale di errore netto, che rende questo tipo di drift anche il più facilmente rilevabile.

Diverso è il comportamento del *gradual drift*, dove il passaggio da un concetto all'altro avviene attraverso una fase transitoria durante la quale i campioni sono ottenuti da una sovrapposizione delle distribuzioni dei due concetti[18]: all'inizio prevale il vecchio concetto, poi entrambi contribuiscono in modo graduale e infine prevale il nuovo, quindi non esiste un vero e proprio

punto di rottura facilmente individuabile. In cybersecurity, questa transizione è imprevedibile e può durare anche migliaia di esempi, rendendo difficile separare il drift dal normale rumore delle fasi iniziali. Un chiaro esempio per questa tipologia potrebbe essere la sostituzione progressiva di un vettore di attacco noto con una variante leggermente modificata per eludere i detector.

Vi è poi l'*incremental drift*, in cui non esiste un concetto di partenza o un concetto di arrivo definibile con certezza, perché la distribuzione si trasforma attraverso una sequenza continua di piccoli stati intermedi, indistinguibili l'uno dall'altro[18][19]. L'accumulo di tanti piccoli errori nel rilevamento porta a una divergenza totale tra la distribuzione attuale e quella di addestramento. La difficoltà principale risiede nel fatto che il segnale di drift rimane al di sotto della soglia minima di rilevazione del detector, rendendo il fenomeno invisibile nelle fasi iniziali della sua manifestazione. In assenza di variazioni significative nel breve periodo, il sistema non dispone di elementi sufficienti per attivare un meccanismo di allerta tempestiva. Per individuare questa tipologia di drift sarebbe necessario confrontare distribuzioni estratte da finestre temporali molto distanti tra loro, con un duplice svantaggio: da un lato, un costo computazionale più elevato rispetto ai metodi di rilevazione convenzionali; dall'altro, un ritardo di adattamento che si accumula progressivamente prima che il problema risulti evidente. Rientra in questa categoria il graduale mutamento del profilo del traffico di rete legittimo nel corso di mesi, un processo in cui ogni singola variazione locale risulta trascurabile, ma il cui effetto cumulativo può compromettere in misura sostanziale le prestazioni del modello.

L'ultima tipologia è il *recurring drift* che si distingue dalle precedenti perché in esso le distribuzioni osservate in passato tendono a ripresentarsi con cadenza regolare o irregolare, generando un drift ciclico[19]. In cybersecurity, questo fenomeno si osserva nelle campagne di attacco che si riattivano periodicamente, anche con variazioni minime rispetto all'attacco precedente. Il problema introdotto da questa tipologia è evidente, ovvero i sistemi che scartano i modelli storici perché ritenuti obsoleti perdono la capacità di riconoscere concetti già visti, mentre quelli che li conservano devono gestire troppi campioni rispetto alla disponibilità limitata della memoria storica. Nella realtà applicativa queste quattro tipologie non si presentano quasi mai in forma pura, ma come una sovrapposizione di più tipi di drift nello stesso stream di dati. Il traffico di rete non è mai stazionario e questa caratteristica distingue l'ambito della sicurezza informatica dagli altri domini in cui il Concept Drift viene studiato. L'instabilità della rete è dovuta a diverse cause e motivazioni che si sovrappongono senza sosta, rendendo difficile isolare una singola causa per progettarne una risposta mirata. Una parte del problema nasce dall'evoluzione del

traffico legittimo, dovuta ad aggiornamenti applicativi e all'uso di nuovi protocolli.

Un sistema di rilevamento che considera normali dei pattern generati dai dati storici potrebbe classificare come anomalo del traffico del tutto legittimo ma semplicemente "diverso dal solito", solo perché è cambiata la baseline di riferimento. In questo caso, trattandosi quasi sempre di virtual drift, il confine decisionale sarebbe comunque corretto, ma la distanza tra la distribuzione reale e la distribuzione utilizzata per l'addestramento cresce fino a rendere inaffidabili le stime di confidenza dei modelli. A rendere questo ambito di studio ancora più complesso c'è l'effettiva evoluzione delle tecniche di attacco per aggirare gli IDS, una componente che non esiste negli altri domini e che sovrappone in particolar modo il virtual drift al real drift, generando instabilità nel decidere quelli che sono i limiti predefiniti per il confine decisionale nei modelli. Alcuni esempi di queste evoluzioni possono essere delle varianti di malware che alterano le proprie chiamate API, dei botnet che cambiano la distribuzione temporale dei pacchetti oppure delle campagne di phishing che aggiornano i template per aggirare i filtri definiti per la loro individuazione[20]. La situazione diventa veramente critica quando il drift non è dovuto solo alla variazione di un concetto noto ma deriva dalla comparsa di una classe del tutto nuova, mai vista durante l'addestramento. In questo caso si parla di attacchi "zero-day" che, a differenza delle cause descritte in precedenza, non sono affrontabili con i soli strumenti classici del Concept Drift; infatti, aggiornare il modello su nuovi campioni della distribuzione corrente non basta se il sistema non è in grado di riconoscere che un certo campione non appartiene ad alcuna delle distribuzioni note, anzi rischia di generare un'attribuzione verso la classe più vicina anche se quest'ultima non ha nulla in comune con il campione osservato. Le poche soluzioni che esistono attualmente in letteratura per questo problema richiedono addirittura un intervento umano che quindi può introdurre latenze pericolose in scenari ad alta velocità.

Il quadro finale che emerge da questa analisi rende il Concept Drift in ambito IDS un problema molto più complesso rispetto a qualunque altro contesto applicativo e motiva la ricerca di soluzioni che utilizzino le tecniche di Continual Learning nei meccanismi di rilevazione e che le adattino ad ambienti non stazionari ed ostili come questo. L'obiettivo del presente elaborato è proprio quello di cercare di ridurre al minimo l'influenza di queste componenti nei modelli attuali per ottenere un rilevamento ed una individuazione delle minacce più pulita e più concreta sia nel breve che nel lungo periodo. Una migliore gestione del problema del Concept Drift porta ovviamente all'ottenimento di valori ottimali nelle metriche di valutazione e ad un modello in grado di soffrire meno a tutte quelle che sono le cause che portano al degradamento della

rilevazione e del monitoraggio in tempo reale.

1.5 Continual Learning come soluzione per IDS

L'integrazione del Continual Learning negli IDS nasce dall'esigenza di rispondere alla continua evoluzione della cybersecurity. Nelle reti moderne, le quali sono caratterizzate da infrastrutture in continua evoluzione e da minacce sempre più sofisticate, i modelli di difesa statici sono strutturalmente inadeguati a garantire una protezione duratura nel tempo. L'apprendimento incrementale permette di integrare nuove conoscenze preservando le capacità già acquisite, evitando così la necessità di onerosi riaddestramenti. È proprio in questo contesto che il Continual Learning trova una collocazione naturale: la capacità di aggiornarsi senza dimenticare si traduce direttamente in robustezza e stabilità rispetto al Concept Drift tipico dei flussi di traffico reali[21][22]. Il Concept Drift descrive esattamente questa situazione per cui le proprietà statistiche del traffico si spostano nel tempo in modo imprevedibile, allargando il divario tra la distribuzione vista in fase di addestramento e quella che il modello incontra in produzione. In ambito IDS, questo si traduce in un deterioramento progressivo dell'accuratezza man mano che i pattern del traffico legittimo o le firme degli attacchi si discostano da ciò che il modello ha appreso[22].

Adottare il Continual Learning per il rilevamento di minacce informatiche ha molto senso perché il Class-Incremental Learning è lo scenario che più si avvicina alla realtà operativa dei sistemi di difesa: un modello deve saper distinguere tra un numero crescente di classi di attacco senza poter contare su segnali espliciti sulla loro identità[23]. Questo implica una tensione concreta tra due esigenze opposte: un IDS deve restare abbastanza plastico da adattarsi alle minacce emergenti, senza però perdere la solidità delle rappresentazioni già consolidate[7]. Per fronteggiare queste sfide, la ricerca si è orientata verso varie soluzioni, basate su strategie diverse fra loro. Tra quelle basate sulla tecnica della regolarizzazione dei parametri, l'*Elastic Weight Consolidation (EWC)*[15] affronta il problema penalizzando le modifiche ai parametri ritenuti critici per i task precedenti, così da proteggere ciò che il modello ha già appreso senza bloccare del tutto la sua capacità di aggiornarsi. Un approccio diverso è quello adottato da *Gradient Episodic Memory (GEM)* e dalla sua versione ottimizzata (*A-GEM*)[24], che invece di agire direttamente sui pesi utilizzano una strategia basata sulla memoria episodica, ovvero una specie di archivio, dove, quando il modello impara qualcosa di nuovo, il sistema controlla che i cambiamenti non vadano

ad intralciare ciò che è presente in questo archivio, agendo come un vincolo per proteggere i dati già immagazzinati. A queste due metodologie, infine, si affiancano quelle basate sul replay selettivo e, in particolare, figurano l'*Incremental Classifier and Representation Learning (iCaRL)*[14], il quale seleziona gli esempi più rappresentativi di ciascuna classe e li reintroduce durante i task successivi, combinando questa strategia con la distillazione della conoscenza per stabilizzare le classi già apprese, e il *Dark Experience Replay (DER)*[25], che, a differenza del precedente metodo, sposta l'attenzione sulle uscite interne del modello: anziché riproporre gli input grezzi, conserva le distribuzioni di confidenza prodotte in passato, la cosiddetta "*dark knowledge*", e le usa come riferimento per mantenere coerenza tra i task senza imporre vincoli rigidi sulla plasticità.

Nella pratica, tuttavia, nessuna di queste soluzioni si trasferisce agli IDS senza limitazioni strutturali di rilievo. La gestione del buffer di memoria rappresenta forse il nodo più critico: la conservazione di un numero sufficiente di esemplari rappresentativi, requisito fondamentale per metodi come GEM o iCaRL, può generare un overhead significativo in termini di spazio e di latenza, limitando la scalabilità del sistema su archi temporali estesi o in ambienti operativi con risorse computazionali vincolate. Tale criticità risulta ancora più marcata quando il dominio applicativo è soggetto a un elevato volume di traffico o a una rapida variazione delle distribuzioni, condizioni che imporrebbero un aggiornamento frequente e costoso del buffer stesso[7]. A questi limiti di natura tecnica si aggiungono le criticità legate alla privacy e al quadro normativo definito dalla General Data Protection Regulation (GDPR), che introduce vincoli stringenti sulla conservazione e sul trattamento dei dati personali. La necessità di mantenere campioni storici del traffico di rete entra pertanto in tensione diretta con i principi di minimizzazione e limitazione della conservazione sanciti dalla normativa, rendendo alcune di queste architetture difficilmente adottabili in contesti istituzionali o aziendali senza l'implementazione di opportune misure di anonimizzazione dei dati.

Capitolo 2

Stato dell'Arte

2.1 Related Works

All'interno della letteratura sul Continual Learning, i primi studi che ne affrontano l'applicazione specifica agli IDS basavano principalmente sull'applicazione di metodi di Machine Learning tradizionali, come alberi decisionali, algoritmi genetici e regressione logistica, e venivano utilizzati per la sola classificazione delle minacce in base a caratteristiche di attacco già note[26]. Il progresso nella ricerca ha dimostrato come l'applicazione del Continual Learning a questo ambito sia diventata col tempo necessaria per far fronte alla costante evoluzione del traffico di rete, consentendo l'aggiornamento continuo dei modelli predittivi senza comprometterne la capacità discriminativa sulle classi già osservate. Tra i contributi fondamentali per la presente trattazione vi sono gli studi condotti da Amalapuram et al.[22][27], i quali affrontano l'applicazione del Continual Learning all'interno di Anomaly-Based NIDS. Gli autori sottolineano l'importanza di due aspetti cruciali: l'elevata instabilità delle prestazioni metriche all'ordine sequenziale con cui si presentano i task al modello e l'impatto indotto dallo sbilanciamento delle classi. Per risolvere questi aspetti, gli autori propongono una strategia di replay basata sull'algoritmo *Class Balancing Reservoir Sampling*, dimostrando come l'efficacia del Continual Learning sui dati del traffico di rete sia strettamente legata alla distribuzione statistica delle classi nel tempo. Questo risultato evidenzia come nei sistemi di rilevamento la sfida principale non risiede solo nella diminuzione del Catastrophic Forgetting, ma soprattutto nella gestione della memoria episodica. In continuità con questa premessa, si colloca lo studio comparativo effettuato da Bhurani et al.[28], il quale costituisce un riferimento essenziale per comprendere l'efficacia di metodologie consolidate, quali *Elastic Weight Consolidation (EWC)* e *Gradient Episodic Memory (GEM)*, nel

contesto della cybersecurity. All'interno di questo lavoro, vengono valutati e confrontati diversi approcci di Class-Incremental Learning applicati a flussi di dati di rete, tra cui i metodi basati sulla regolarizzazione parametrica, come EWC, e i metodi fondati sul rehearsal con vincoli geometrici sui gradienti, di cui GEM costituisce il paradigma principale. Dagli esperimenti condotti in questo lavoro, risulta evidente come in ambienti IDS caratterizzati da un'elevata non stazionarietà l'efficacia di EWC manifesti chiari limiti strutturali; al contrario, GEM si dimostra più idoneo a preservare l'accuratezza sulle classi storiche poiché impone un vincolo diretto sull'aggiornamento dei gradienti, impedendo che l'ottimizzazione sulle minacce correnti degradi in modo eccessivo la memoria del passato.

Tra i lavori più recenti fra quelli presenti in letteratura, ricopre un ruolo di spicco quello proposto da Di Monda et al.[29] poiché introduce un cambiamento di prospettiva rispetto ai lavori precedentemente esaminati. Di Monda et al. propongono un approccio di *Few-Shot Class-Incremental Learning* per contesti NIDS, adattando la logica della metodologia *Incremental Classifier and Representation Learning (iCaRL)* alla classificazione dei flussi di rete attraverso una sua variante, denominata *iCaRL+*. Questo contributo dimostra la validità del paradigma basato sui prototipi, il quale si rivela veramente efficace qualora i nuovi attacchi emergano con una disponibilità esigua di campioni e debbano essere integrati in un classificatore già addestrato. Rispetto ai metodi analizzati in precedenza, quest'ultimo non ha come obiettivo la manipolazione del processo di ottimizzazione, bensì il mantenimento di una rappresentazione stabile dello spazio delle classi osservate e l'aggiornamento della memoria degli esemplari in modo coerente con l'espansione dello spazio delle minacce.

La letteratura recente focalizzata sul dominio IDS manifesta particolare interesse sulla metodologia *Dark Experience Replay (DER)*[25], in grado di coniugare il rehearsal di campioni storici con una componente di distillazione basata sulla memoria grezza, costituita dai logits. A tal proposito, Prasath et al.[21] effettuano un'analisi comparativa tra DER e molteplici strategie su benchmark di riferimento per il rilevamento delle intrusioni, come NSL-KDD e CICIDS2017. I risultati proposti da questo lavoro evidenziano come i metodi strutturati sulla combinazione di replay e distillazione superino le tradizionali soluzioni, riducendo di gran lunga gli effetti del Catastrophic Forgetting. L'analisi effettuata delinea DER come una strategia particolarmente idonea per i sistemi IDS poiché combina l'assimilazione di nuovi pattern d'attacco alla salvaguardia della stabilità decisionale sulle classi precedentemente apprese, utilizzando la complessa struttura decisionale e le relazioni di incertezza tra le classi fornite dai logits.

2.2 Principali metodologie

Nella sezione che segue vengono esaminati nel dettaglio i fondamenti procedurali dei principali algoritmi di Continual Learning. L'applicazione di tali approcci al rilevamento di intrusioni sul traffico di rete richiede un'analisi approfondita delle loro assunzioni di fondo, al fine di evidenziare le problematiche che emergono quando queste metodologie devono gestire l'eterogeneità, la non stazionarietà e lo sbilanciamento tipici del traffico reale: dalla rigidità della regolarizzazione parametrica ai vincoli geometrici sui gradienti, fino alle complessità di gestione della distillazione della conoscenza in presenza di flussi fortemente asimmetrici.

2.2.1 Elastic Weight Consolidation

Tra i metodi di regolarizzazione, l'Elastic Weight Consolidation (EWC) affronta il Catastrophic Forgetting vincolando la modifica dei parametri rilevanti per i task precedenti. L'importanza di ciascun parametro viene stimata tramite la matrice di informazione di Fisher, che approssima la curvatura della funzione di verosimiglianza calcolata sul task precedente. Questa stima viene incorporata nella funzione obiettivo come termine di penalità quadratica, che penalizza gli aggiornamenti dei parametri in misura proporzionale alla loro importanza, preservando così la conoscenza acquisita nel corso dell'apprendimento dei task successivi[15]. La protezione dei parametri appresi non avviene allo stesso modo per tutti i task, infatti ogni peso riceve un livello diverso di rigidità a seconda della sua importanza per il task appena appreso. L'importanza di ciascun peso viene stimata attraverso la diagonale della *Fisher Information Matrix*, la quale funge da misura della sensibilità della funzione di perdita rispetto a ciascun parametro. Dunque, se un peso ha un ruolo rilevante nel mantenere le prestazioni sul task precedente, il modello viene penalizzato quando prova a modificarlo per adattare il proprio apprendimento a favore del nuovo task; invece, se il peso è poco influente, può essere aggiornato con maggiore libertà. Questo rende EWC una tecnica più raffinata di una semplice regolarizzazione globale perché introduce una sorta di protezione selettiva nei confronti dei parametri appresi.

La funzione di perdita standard di EWC è definita come:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_{\text{new}}(\theta) + \frac{\lambda}{2} \sum_i F_i (\theta_i - \theta_i^*)^2, \quad (2.1)$$

dove $\mathcal{L}_{\text{new}}$ è la funzione di perdita del task corrente, θ^* rappresenta la soluzione ottimale del task precedente, F_i è il peso assegnato al parametro i dalla matrice di Fisher e λ controlla il

bilanciamento tra apprendimento nuovo e conservazione del passato[15]. Questa equazione descrive come in questo metodo l'aggiornamento dei pesi avviene tenendo conto di un vincolo che impedisce al modello di allontanarsi troppo dai valori ottimali. L'EWC non fa altro che aggiungere un termine di penalità alla normale funzione di errore che "punisce" il modello ogni volta che si allontana troppo da quello che aveva imparato prima per non stravolgere i pesi ritenuti importanti per le vecchie classi. Dunque, negli approcci basati su questa strategia, si addestra il modello su un task, si salvano i parametri finali, si stima l'importanza di ciascun peso e, infine, si passa al task successivo aggiungendo alla funzione di perdita corrente la penalità EWC, la quale viene rappresentata nella Figura 2.1.

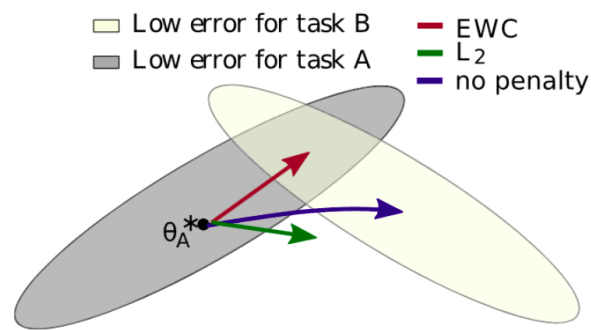


Figura 2.1: Confronto tra EWC, normalizzazione L_2 e baseline senza vincoli

Con l'aumento del numero dei task, il ragionamento si estende accumulando l'informazione relativa a tutti i task precedenti, il che introduce un problema strutturale importante, ovvero la crescita del termine di penalità che nella formulazione è lineare con il crescere del numero di task, poiché viene aggiunto un nuovo contributo per ciascun task osservato. In sistemi come gli IDS, dove le classi di attacco si accumulano nel tempo su orizzonti temporali estesi, questa crescita rende nel tempo il modello più costoso e difficile da gestire. Per affrontare questo limite, in letteratura è stata proposta una variante della tecnica denominata *Online EWC*[30], la quale mantiene una stima unica della matrice di Fisher aggiornata in modo incrementale, evitando l'accumulo di un termine separato per ogni task passato e migliorando sensibilmente la scalabilità di questo metodo. Un approccio che utilizza un concetto molto simile a questo è quello della *Synaptic Intelligence (SI)*[31], che stima l'importanza dei parametri online durante il training stesso e non a posteriori come avviene in EWC. Questa caratteristica rende SI più adatta a scenari continui in cui non è possibile effettuare una stima separata del termine di ogni task, collocandola insieme ad EWC e alle sue varianti all'interno di una famiglia più grande di metodi basati sulla protezione selettiva dei parametri.

I vantaggi principali di questo approccio risiedono nel fatto che l'EWC non necessita di un buffer di esempi passati, eliminando così le problematiche di memoria e di conformità normativa precedentemente discusse, e non impone modifiche architetturali al modello, integrandosi in modo trasparente con le procedure di ottimizzazione standard basate su gradiente. Dal punto di vista delle prestazioni, questa tecnica consente di ridurre significativamente il Catastrophic Forgetting, dimostrando una efficacia trasversale tanto in contesti di apprendimento supervisionato quanto in scenari di Reinforcement Learning, ambiti caratterizzati da dinamiche di aggiornamento differenti. Per la sua capacità di unire efficacia e leggerezza computazionale, l'EWC è oggi considerato uno dei metodi fondamentali del Continual Learning, costituendo un riferimento per la ricerca successiva e per lo sviluppo di approcci più avanzati[15].

Tuttavia, le limitazioni di EWC emergono con maggiore evidenza quando i task sono molto diversi tra loro, come avviene negli scenari IDS, nei quali la distribuzione del traffico di rete e anche la natura delle minacce evolvono in maniera continua e non stazionaria. In contesti così dinamici, la regolarizzazione introdotta da EWC può mostrarsi troppo vincolante, limitando le capacità del modello di adattarsi ai nuovi task e compromettendo di conseguenza le sue prestazioni nel rilevamento di nuove tipologie di attacco[15][13]. Un altro limite rilevante riguarda l'impiego della diagonale della matrice di Fisher per stimare l'importanza dei parametri: un'approssimazione ottenuta in questo modo è conveniente dal punto di vista computazionale e quindi adeguata a scenari di apprendimento continuo, ma non consente di modellare le dipendenze tra i parametri della rete, fornendo una rappresentazione solo parziale della struttura dell'informazione. Per questi motivi, in contesti caratterizzati da elevata variabilità, di solito EWC viene integrato con ulteriori strategie, per mirare ad ottenere una stima più accurata dell'importanza dei pesi e per preservare in modo più efficace il bilanciamento tra stabilità e plasticità del modello.

2.2.2 Gradient Episodic Memory

Il Gradient Episodic Memory (GEM) sfrutta una memoria episodica per conservare parte degli esempi dei task passati per ridurre il Catastrophic Forgetting[24]. Questa tecnica mira a controllare i gradienti di aggiornamento in modo che il nuovo apprendimento non peggiori le prestazioni sui task precedenti, affrontando così il problema del forgetting imponendo un vincolo sulla direzione in cui il modello può muoversi nello spazio dei parametri. Per ogni task precedente viene mantenuto un piccolo insieme di campioni rappresentativi, che viene poi usato

per valutare quanto un nuovo aggiornamento potrebbe danneggiare la conoscenza acquisita. All'arrivo di ciascuna sezione del nuovo task, il modello calcola il gradiente della funzione di perdita corrente, ma prima di applicarlo verifica se quell'aggiornamento causerebbe un aumento della funzione sui task passati; nel caso in cui il gradiente rischia di peggiorare un task precedente, allora viene proiettato verso una regione dello spazio dei parametri compatibile con la conservazione delle prestazioni precedenti, definita come "zona sicura"[24].

La formulazione matematica del gradiente calcolato dal GEM viene definita come:

$$\begin{aligned} g &= \nabla_{\theta} \mathcal{L}_t(\theta), \quad g_k = \nabla_{\theta} \mathcal{L}_k(\theta) \\ \tilde{g} &= \arg \min_{g'} \frac{1}{2} \|g' - g\|^2 \quad \text{tale che} \quad g'^{\top} g_k \geq 0 \quad \forall k < t \end{aligned} \quad (2.2)$$

dove g è il gradiente della funzione di perdita del task corrente, g_k è il gradiente calcolato sulla memoria episodica del task passato k , mentre $\tilde{g}$ è il gradiente proiettato, ottenuto imponendo il vincolo $g'^{\top} g_k \geq 0 \quad \forall k < t$, il quale garantisce che l'aggiornamento non aumenti la funzione dei task precedenti.

La logica alla base di questa ottimizzazione, raffigurata in Figura 2.2, offre una nuova prospettiva per affrontare il problema del Catastrophic Forgetting. Infatti, a differenza dell'EWC, il quale penalizza i pesi ritenuti più importanti, il GEM lavora sugli aggiornamenti, imponendo un vincolo che evita al gradiente di scegliere direzioni che possano degradare il passato. In questo modo, in base al vincolo, non va ad annullare ciò che era stato imparato prima, ma va a deformare il meno possibile il gradiente proiettato per correggerne la direzione in modo da non farlo allontanare troppo da quella che è la direzione originale. Il GEM, dunque, mantiene una memoria episodica con esempi selezionati dai task precedenti, poi calcola il gradiente per il task corrente e, infine, risolve un problema di ottimizzazione per ottenere il gradiente proiettato. La presenza di un vincolo per ciascun task passato implica un costo computazionale che aumenta all'aumentare del numero dei task e della dimensione della memoria; proprio per ridurre questa complessità computazionale, successivamente è stata introdotta una sua versione ottimizzata, definita come Averaged Gradient Episodic Memory (A-GEM), che semplifica il vincolo usando un solo riferimento medio della memoria invece di una serie di vincoli separati per ciascun task passato[32].

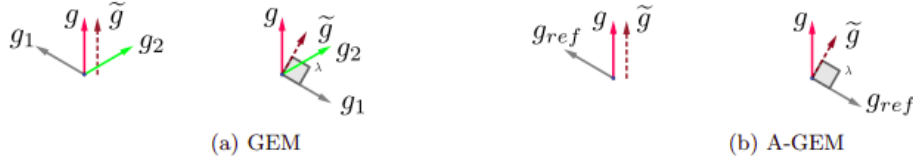


Figura 2.2: Calcolo del gradiente in GEM e A-GEM

Questa semplificazione ha però un costo concreto, ovvero, in presenza di task con distribuzioni molto eterogenee, il vincolo medio può non essere sufficientemente discriminativo per proteggere ciascun task passato, portando a un leggero calo di prestazioni rispetto a GEM. In ambito IDS, dove le firme di attacchi diversi possono occupare regioni dello spazio delle feature molto distanti tra loro, questa limitazione di A-GEM merita ancora più attenzione quando si sceglie di applicare una delle due varianti.

GEM rappresenta un metodo molto rigoroso nel controllo del forgetting e per questo motivo viene spesso descritto come uno dei metodi centrali nella letteratura sul Continual Learning in ambito di cybersecurity come base concettuale per molti sviluppi successivi[7][13]. Questa sua efficienza lo rende una solida base di partenza nel rilevamento di attacchi sul lungo periodo, ma il suo elevato costo computazionale non lo rende completamente adatto per l'analisi del traffico dati, che richiede risposte immediate per l'ottenimento di risultati concreti. In particolare, l'A-GEM può risultare una scelta interessante in ambito IDS perché offre un buon compromesso tra prestazioni e costi. Il limite, però, è che questa tecnica dipende comunque da una memoria episodica ridotta e da campioni rappresentativi del passato, e in un IDS, se la memoria non cattura bene attacchi rari, il modello riesce ad adattarsi solo in modo parziale ai cambiamenti reali, rischiando di compromettere in modo eccessivo il rilevamento[32]. GEM è il metodo che più di tutti punta al controllo rigoroso del Catastrophic Forgetting, imponendo un vincolo esplicito sulla direzione degli aggiornamenti del gradiente in modo che nessun task passato venga danneggiato. Questa tecnica è concettualmente raffinata ma ha un costo computazionale che cresce esponenzialmente rispetto al numero di task e alla dimensione della memoria episodica, poiché richiede la soluzione di un problema di programmazione quadratica ad ogni step di aggiornamento. A-GEM mitiga questo problema sostituendo i vincoli ad ogni task con un unico vincolo medio, ma introduce una perdita di Precision che si manifesta di più proprio in contesti come quelli IDS dove esistono molti task eterogenei distanti tra loro nello spazio delle feature.

2.2.3 Incremental Classifier and Representation Learning

L'Incremental Classifier and Representation Learning (iCaRL) combina tre elementi che di solito vengono trattati separatamente, i quali sono memoria episodica, apprendimento incrementale della rappresentazione e classificazione basata su prototipi, con l'obiettivo di ridurre il Catastrophic Forgetting mantenendo al tempo stesso una buona capacità di generalizzazione[14][23], perciò è ritenuto uno dei metodi più efficienti nel Continual Learning. La sua rilevanza è stata confermata anche nel dominio della cybersecurity, dove ha ispirato varianti specifiche per la classificazione delle minacce nel traffico di rete. Tra queste, una delle più significative è iCaRL+, proposta da Di Monda et al.[29], che applica i principi fondamentali di iCaRL nell'ambito NIDS. In tale contesto, il metodo viene utilizzato per integrare progressivamente nuove tipologie di traffico e nuove classi di attacco disponendo soltanto di un numero ridotto di esempi per ciascuna classe emergente. La strategia su cui si basa iCaRL, come si può osservare in Figura 2.3, consiste nel conservare per ciascuna classe osservata un numero limitato di esempi rappresentativi, denominati "esemplari", i quali vengono utilizzati per mantenere informazioni sulle classi precedenti e per costruire il classificatore incrementale interno del modello. A differenza dei metodi tradizionali che utilizzano un classificatore finale standard aggiornato tramite backpropagation, come, ad esempio, avviene per GEM, questa tecnica, invece, adotta un approccio basato sul principio del *nearest-mean-of-exemplars (NME)*, tramite il quale per ogni classe y viene calcolato un prototipo definito come la media delle rappresentazioni associate agli esemplari memorizzati:

$$\mu_y = \frac{1}{|P_y|} \sum_{p \in P_y} \phi(p), \quad (2.3)$$

dove P_y indica l'insieme degli esemplari della classe y e $\phi(p)$ indica la funzione di estrazione delle caratteristiche appresa dal modello. Un prototipo, quindi, è una specie di rappresentazione media di ciascuna classe nello spazio delle feature, quasi come se fosse il baricentro della classe y . La classificazione di un nuovo campione x avviene assegnando l'etichetta associata al prototipo più vicino nello spazio delle feature:

$$y^* = \arg \min_{y=1,\dots,t} \|\phi(x) - \mu_y\|, \quad (2.4)$$

dove $\|\phi(x) - \mu_y\|$ indica la distanza tra il campione x e il prototipo della classe y , calcolata tramite distanza euclidea. Dunque, la classe assegnata y^* è quella del prototipo più vicino.

L'utilizzo di questa strategia è molto utile nei contesti dinamici di lunga durata perché mantiene coerenza tra classificatore e rappresentazione anche quando il modello modifica lo spazio delle feature durante la fase di addestramento, dato che i prototipi vengono aggiornati di conseguenza, evitando il disallineamento che caratterizza una gran parte dei classificatori tradizionali[14]. Tuttavia, l'efficacia di questa classificazione dipende fortemente dalla stabilità dello spazio delle feature nel tempo. Se il backbone si aggiorna in modo significativo durante i task successivi, i prototipi calcolati sui task precedenti diventano progressivamente obsoleti, anche nel caso in cui gli esemplari fossero stati selezionati in modo ottimale. Questa dipendenza dalla qualità della rappresentazione è un aspetto critico di iCaRL che emerge quando il numero di task cresce e il modello è costretto ad aggiornare profondamente le proprie rappresentazioni interne per favorire l'interpretazione di nuove classi molto diverse da quelle già viste.

La selezione degli esemplari costituisce uno degli elementi fondamentali di questo metodo: iCaRL conserva i campioni di ciascuna classe tramite una procedura di *herding*, finalizzata a scegliere gli esempi che meglio approssimano la distribuzione reale della classe nello spazio delle feature. La selezione avviene ripetutamente in modo che la media delle loro rappresentazioni sia il più possibile vicino alla media calcolata sull'intera classe:

$$p_k = \arg \min_{x \in X} \left\| \mu - \frac{1}{k} \left[\phi(x) + \sum_{j=1}^{k-1} \phi(p_j) \right] \right\|, \quad (2.5)$$

dove μ indica la media globale della classe e p_k indica l'esemplare scelto al passo k . iCaRL non può mantenere tutti gli esempi in memoria, quindi deve scegliere quali tenere. La procedura di *herding*, ad ogni passo, sceglie l'esempio che, aggiunto agli esemplari già scelti, fa in modo che la loro media nello spazio delle feature sia il più vicino possibile alla media dell'intera classe μ e che quindi la rappresentino meglio anche se in numero ristretto.

Questo approccio costruisce una memoria episodica compatta ma sufficientemente rappresentativa, anche se le risorse sono limitate, poiché il numero di esemplari mantenuti per ciascuna classe vista viene ridotto man mano che il numero totale di classi osservate aumenta, così da rispettare un budget di memoria prefissato[14]. In un IDS con molte classi di attacco accumulate nel tempo, questo squilibrio può portare ad un degrado sistematico della capacità di riconoscere minacce già osservate in passato[23]. A questo si aggiunge anche il problema dello sbilanciamento delle classi, che in ambito IDS è una condizione strutturale: il traffico malevolo è tipicamente minoritario rispetto al traffico legittimo e alcune tipologie di attacco presentano

pochissimi campioni nel training set originale. La procedura di herding, progettata per operare con classi relativamente bilanciate, perde efficacia in questi contesti, poiché se una classe di attacco dispone di pochi campioni la media della classe nello spazio delle feature è poco stabile per natura e gli esemplari selezionati rischiano di non essere abbastanza rappresentativi della sua distribuzione reale.

Per completare la strategia, iCaRL introduce anche una componente di *knowledge distillation* per limitare il forgetting durante l'aggiornamento del modello, unendo i nuovi campioni agli esemplari delle classi precedenti per generare il training set definitivo. La funzione obiettivo utilizzata da iCaRL combina quindi una componente supervisionata relativa alle nuove classi e una componente di distillazione finalizzata a mantenere coerente il comportamento del modello fino al task corrente:

$$\mathcal{L}_{\text{iCaRL}} = - \sum_{(x_i, y_i) \in D} \left[\sum_{y=s}^t \delta_{y=y_i} \log g_y(x_i) + \sum_{y=1}^{s-1} (q_i^y \log g_y(x_i) + (1 - q_i^y) \log(1 - g_y(x_i))) \right], \quad (2.6)$$

dove $g_y(x_i)$ indica l'output del modello per la classe y , mentre q_i^y indica la probabilità prodotta dal modello precedente prima dell'aggiornamento. La prima componente della funzione favorisce l'acquisizione di informazioni in modo standard sulle nuove classi, invece la seconda cerca di non degradare le conoscenze acquisite, senza utilizzare le etichette vere, ma utilizzando le rappresentazioni del modello precedente come target per mantenere lo stesso modo di ragionare utilizzato fino a quel momento.

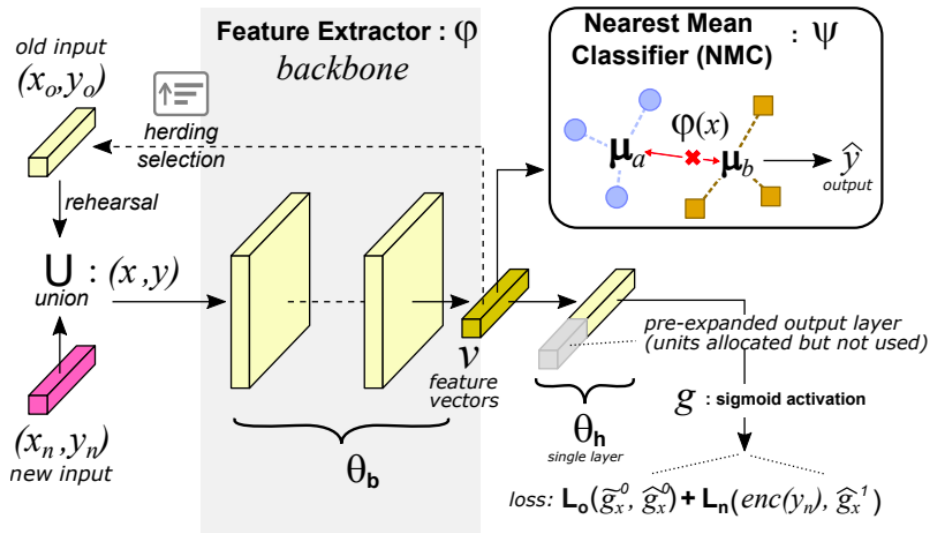


Figura 2.3: Comportamento interno di iCaRL

Nel contesto della cybersecurity, questa metodologia si presenta come una soluzione molto interessante perché in contesti dinamici è in grado di aggiornare il sistema all'arrivo di nuovi attacchi, senza dover riaddestrare il modello su tutto il traffico storico[21]. Inoltre, la classificazione basata su prototipi è coerente con la necessità di adattare continuamente la rappresentazione del traffico di rete senza perdere la capacità di riconoscere minacce già osservate. Tuttavia, gli IDS reali presentano delle caratteristiche che rendono il problema più complesso, infatti il traffico di rete evolve molto rapidamente, le classi presentano un forte sbilanciamento e molte minacce emergenti non appartengono a categorie già note. Le strategie basate solo su iCaRL mostrano alcuni limiti strutturali, poiché tale metodo nasce in principio per ambienti supervisionati e chiusi, nei quali l'arrivo di nuove classi avviene in modo controllato. Di conseguenza, per ottenere un approccio efficiente per i sistemi di rilevamento moderni, iCaRL viene integrato con tecniche aggiuntive di gestione avanzata del drift per garantire una maggiore robustezza nonostante la non stazionarietà dei dati[21][4][1].

2.2.4 Dark Experience Replay

Il Dark Experience Replay (DER) è un metodo di Continual Learning che si differenzia dagli altri precedentemente analizzati perché la sua idea distintiva, come si può osservare in Figura 2.4, è quella di conservare in una memoria episodica alcuni esempi del passato e, soprattutto, i logits prodotti dal modello nel momento in cui questi esempi vengono osservati, ovvero i valori grezzi prodotti dall'ultimo layer prima della funzione di attivazione finale. Tali valori, indicati

con il termine "*dark knowledge*", contengono molte più informazioni delle sole etichette[25] e, per questo motivo, il loro utilizzo verrà riproposto anche nella nuova metodologia presentata nell'elaborato. Il DER ottimizza una funzione obiettivo composta da due termini, dove da un lato si ha la perdita supervisionata, tramite la funzione di Cross-Entropy, sul task corrente e dall'altro un termine di distillazione che forza la coerenza tra i logits attuali e quelli già presenti in memoria:

$$\mathcal{L}_{\text{DER}}(\theta) = \mathcal{L}_{\text{CE}}(y, f_{\theta}(x)) + \alpha \mathbb{E}_{(x_0, z_0) \sim M} \left[\frac{1}{2} \|z_0 - h_{\theta}(x_0)\|_2^2 \right], \quad (2.7)$$

dove M indica la memoria episodica, la coppia (x_0, z_0) indica un vecchio esempio e i logits che il modello produceva quando aveva visto quell'esempio per la prima volta e α è un parametro numerico utilizzato per bilanciare il contributo del replay rispetto all'apprendimento del task corrente[25]. Nella prima componente, così come avviene per iCaRL, viene utilizzata la classica funzione di perdita Cross-Entropy, che calcola le differenze tra la distribuzione di probabilità delle classi prevista dal modello e la distribuzione reale dei dati; mentre con la seconda componente, si vuole minimizzare la distanza tra i logits memorizzati z_0 e quelli che il modello sta producendo in quel momento $h_{\theta}(x_0)$ sullo stesso esempio, bilanciando con α quanto peso dare al vincolo generato da questa componente rispetto al nuovo apprendimento. Il modello deve continuare ad apprendere i dati nuovi, ma, allo stesso tempo, non deve allontanarsi troppo dal comportamento che aveva sugli esempi già visti, cercando di mantenere la continuità funzionale della rete e non soltanto la correttezza delle etichette passate.

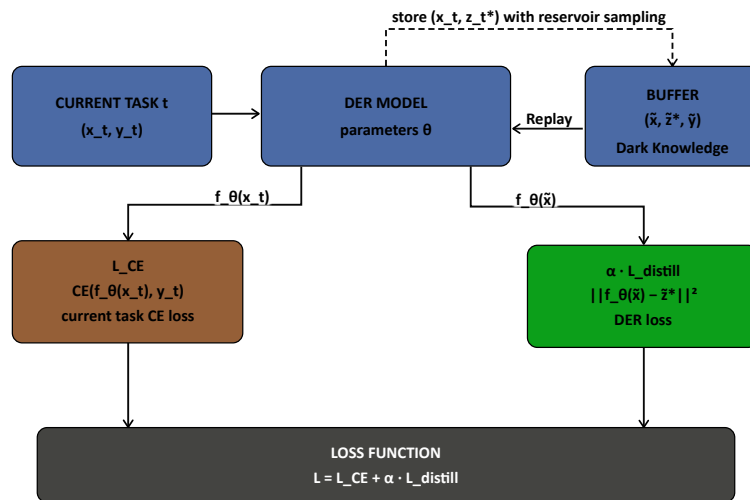


Figura 2.4: Comportamento interno di DER

In letteratura, viene proposta una variante della tecnica, denominata *DER++*, che aggiunge alla funzione di costo un terzo termine di Cross-Entropy applicato direttamente agli esemplari in memoria con le loro etichette:

$$\begin{aligned} \mathcal{L}_{\text{DER++}}(\theta) = & \mathcal{L}_{\text{CE}}(y, f_{\theta}(x)) \\ & + \alpha \mathbb{E}_{(x_0, z_0) \sim M} \left[\frac{1}{2} \|z_0 - h_{\theta}(x_0)\|_2^2 \right] \\ & + \beta \mathbb{E}_{(x_0, y_0) \sim M} [\mathcal{L}_{\text{CE}}(y_0, f_{\theta}(x_0))], \end{aligned} \quad (2.8)$$

dove β controlla il peso del nuovo termine e y_0 indica le etichette vere degli esempi in memoria[25]. Questo terzo contributo forza il modello a classificare correttamente gli esempi memorizzati non solo a livello di logits ma anche a livello di etichette, combinando il vincolo sulla struttura interna delle rappresentazioni con un vincolo sulla correttezza della classificazione. *DER++* risulta generalmente più stabile di *DER* in caso di distribuzioni con forte variabilità tra task; tuttavia, l'aggiunta di un termine di Cross-Entropy introduce una tensione interna non trascurabile. Il vincolo sull'etichetta, visibile nella rappresentazione del comportamento interno di *DER++* in Figura 2.5, tratta gli esempi in memoria in modo rigido, assumendo implicitamente che le etichette originali restino target validi anche dopo che il modello ha riorganizzato il proprio spazio delle feature per accomodare le classi nuove.

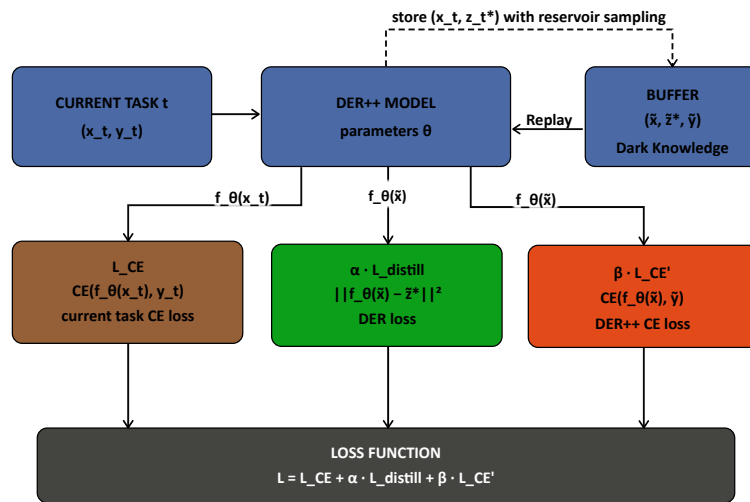


Figura 2.5: Comportamento interno di *DER++*

Quando questa riorganizzazione è profonda, il termine aggiuntivo può entrare in conflitto con il termine di distillazione sui logits che invece agisce in modo più morbido preservando le

relazioni di incertezza tra classi piuttosto che imporre confini decisionali fissi. Nel contesto della rilevazione di minacce sul traffico di rete, dove i pattern di attacchi diversi condividono caratteristiche simili e dove i confini tra classi non sono netti, questa rigidità del vincolo sulle etichette rischia di rendere DER++ meno adatto di DER in scenari con forte sovrapposizione e con elevata variabilità tra classi[25]. Un altro aspetto che caratterizza DER è l'utilizzo di un buffer di memoria dinamico, reso tale dalla tecnica del *reservoir sampling*, per cui si cerca di mantenere un campione casuale e rappresentativo dello stream di dati, così da mantenere la memoria di dimensione costante lungo l'intero flusso, quindi anche per un periodo di lunga durata. Questo aspetto ha implicazioni concrete anche in termini di efficienza della memoria, poiché ogni campione memorizzato porta con sé i logits associati, perciò ogni slot del buffer trasporta una quantità di informazioni maggiore rispetto a quanto otterrebbe un metodo come iCaRL con lo stesso budget, dove ciascun esemplare conserva solo l'input. DER è quindi strutturalmente meno sensibile alla riduzione del buffer rispetto ai metodi basati esclusivamente sul replay degli esempi e questa è una proprietà molto rilevante in ambienti IDS dove le risorse di memoria sono limitate. Nonostante l'impiego di un buffer adattivo, tramite l'utilizzo dei logits, questa tecnica non cerca solo di ricordare la classe corretta ma cerca di preservare il modo in cui il modello interpretava i campioni nel suo spazio decisionale interno.

Vale anche la pena mettere a confronto DER con iCaRL sul piano dell'uso della distillazione dato che entrambi i metodi ne fanno uso ma con obiettivi diversi. iCaRL distilla sulle probabilità di output per stabilizzare la classificazione delle classi vecchie, lavorando sulla decisione finale del modello. DER, invece, distilla sui logits grezzi per preservare la geometria interna dello spazio decisionale, ovvero il modo in cui il modello distribuisce l'incertezza e le relazioni tra le classi ancora prima che queste vengano normalizzate in probabilità. Quando la struttura delle rappresentazioni porta informazioni che le sole etichette non catturano, così come avviene quando si analizza il traffico di rete per individuare le minacce, il DER è concettualmente più adatto a preservare la capacità discriminativa del modello nel tempo. E proprio per questo motivo, l'idea alla base di questo approccio di utilizzare i logits per preservare anche il modo di ragionare del modello verrà poi utilizzata nel metodo proposto nella tesi perché ritenuta la tecnica più valida fra quelle presenti in letteratura. Infatti, rispetto ai metodi descritti in precedenza, DER ha un vantaggio strutturale netto perché conserva sia esempi reali sia la risposta funzionale del modello su quegli esempi, invece di focalizzarsi sull'eccessiva protezione dei dati del passato o sull'imposizione di vincoli per evitare aggiornamenti distruttivi. In questo modo unisce replay

e distillazione in un modo tale da essere il metodo ritenuto più convincente per un contesto di cybersecurity, dove i sistemi operano per analizzare il traffico e gli attacchi che mutano continuamente nel tempo. Alcuni studi relativi all'ambito degli IDS dove vengono confrontati diversi approcci di Continual Learning su diversi dataset hanno dimostrato come le tecniche basate sul DER ottengano un'accuratezza migliore e un FPR (False Positive Rate) più basso rispetto ad approcci più tradizionali, riuscendo anche ad apprendere in modo incrementale nuovi pattern di traffico[21]. Oltre questo, studi recenti hanno dimostrato anche come i modelli basati sul replay siano considerati tra i più promettenti per questi contesti dinamici in cui è necessaria una risposta immediata, nonostante restino aperti alcuni problemi come lo sbilanciamento delle classi, la disponibilità limitata di etichette in memoria per non sovraccaricare il costo computazionale e la gestione del drift[22][12].

2.3 Gap applicativo

Le metodologie di Continual Learning esaminate in precedenza hanno trovato ciascuna una propria applicazione nel dominio della sicurezza informatica: EWC e GEM sono state validate come soluzioni per il Class-Incremental Learning sul traffico di rete; iCaRL ha dimostrato la propria efficacia in scenari caratterizzati da scarsità di dati; DER si è imposta come una delle strategie più promettenti grazie all'efficace preservazione della "*dark knowledge*" tramite la distillazione dei logits. Nonostante ciò, questi metodi nascono per fini diversi e il loro utilizzo nell'ambito del rilevamento di attacchi cyber avviene spesso in ambienti sperimentali, senza affrontare in modo congiunto tutte le criticità che rendono questo dominio strutturalmente diverso dagli altri. Dall'analisi di questi approcci emergono tre famiglie di limiti strutturali che nessuno dei metodi affronta in modo congiunto per un sistema di rilevamento delle intrusioni basato su Continual Learning:

- **Rigidità del controllo sullo "stability-plasticity" dilemma:** EWC penalizza i parametri sulla base di una stima della loro importanza calcolata a posteriori al termine di ogni task e applica questo vincolo con la stessa intensità indipendentemente dal forgetting osservato, senza introdurre alcuna forma di adattamento rispetto all'andamento delle prestazioni[15]. Allo stesso modo, sia DER sia DER++ trattano il coefficiente di distillazione come un iperparametro che viene fissato prima dell'addestramento e mai aggiornato lungo la sequenza di task[25]: una scelta concettuale che non risulta molto compatibile con un

dominio in cui l'intensità del Concept Drift tra un task e il successivo non è nota e può variare in modo repentino se compaiono nuove classi di attacco. Un vincolo fisso rischia di essere insufficiente quando il forgetting è elevato e troppo conservativo quando il drift è contenuto, limitando la plasticità del modello.

- **Gestione inefficiente della memoria episodica in presenza di forte sbilanciamento delle classi:** GEM e A-GEM non prevedono alcun meccanismo per il riempimento della memoria episodica, limitandosi a conservare un sottoinsieme di esempi passati senza alcun criterio[24][32]. Invece, iCaRL alloca un budget di esemplari per classe che si riduce all'aumentare del numero di attacchi osservati, generando il fenomeno del "recency bias" e risultando poco efficace quando la classe dispone già in origine di pochi campioni su cui calcolare una media stabile nello spazio delle feature[14]. Il DER, infine, affida la composizione del buffer episodico alla tecnica del Reservoir Sampling puramente casuale che, in un flusso dominato dal traffico legittimo, tende a sovrarappresentare le classi maggioritarie rispetto a quelle malevole più rilevanti in ambito IDS[25]. Il problema dello sbilanciamento delle classi in questo ambito è stato segnalato in letteratura come una delle cause principali dell'instabilità delle metriche di valutazione[27][28]. Infatti, nessuno dei metodi esaminati combina una gestione della memoria episodica realmente sensibile allo sbilanciamento delle classi con un meccanismo di regolarizzazione o distillazione che si adatti dinamicamente al forgetting effettivamente misurato durante l'addestramento.
- **Perdita di robustezza a fronte di sovrapposizione tra classi ed elevata eterogeneità tra task:** GEM impone un vincolo sulla direzione del gradiente per cui è necessario risolvere un problema di programmazione quadratica ad ogni passo di aggiornamento, richiedendo un costo computazionale che cresce parallelamente alla dimensione della memoria episodica e al numero di task; mentre, la semplificazione introdotta da A-GEM, pur riducendo questo costo, perde capacità discriminativa quando i task sono molto eterogenei tra loro[24][32]. Inoltre, vi è la classificazione per prototipi di iCaRL che dipende dalla stabilità dello spazio delle feature nel tempo e che perde di efficacia quando il backbone si aggiorna in modo significativo per favorire l'apprendimento di classi molto diverse da quelle note[14][23]. Il DER++, invece, introduce un termine di Cross-Entropy sugli esempi del buffer che entra in tensione con il contributo di distillazione sui logits ogni volta che il modello deve riorganizzare lo spazio delle feature in modo marcato[25]. In un dominio in cui i confini decisionali tra classi di attacco diverse sono raramente ben

definiti, questa tensione rischia di incidere in modo elevato sulla stabilità e sulla robustezza del modello nel lungo periodo.

Capitolo 3

Metodologia

La continua evoluzione delle minacce cyber, unita alla naturale variabilità del traffico di rete, causa l'amplificazione del Concept Drift: gli attaccanti modificano le caratteristiche degli attacchi con l'obiettivo specifico di eludere i meccanismi di rilevamento esistenti e per questo motivo un modello addestrato su una distribuzione di attacchi nota perde efficacia. Questa dinamica evidenzia l'importanza di continui aggiornamenti, poiché ogni finestra temporale in cui un IDS non riesce a rilevare nuove tipologie di attacco si traduce in un aumento di attacchi che attraversano il sistema di rilevamento senza essere identificati. Allo stesso tempo, un aggiornamento mal gestito e un addestramento che non permetta un rapido adattamento delle rappresentazioni interne rischiano di compromettere la capacità del modello di riconoscere le minacce già note, alimentando il Catastrophic Forgetting e esponendo nuovamente l'infrastruttura ad attacchi mitigati in precedenza.

Il Continual Learning si propone come risposta a questa duplice esigenza, ma, nonostante l'area di ricerca sia in costante sviluppo, il divario tra l'efficacia dei modelli dimostrata sui benchmark presenti in letteratura e il loro reale rendimento nei sistemi di produzione costituisce ancora oggi un ostacolo primario. Il presente lavoro di tesi propone *Dark Adaptive and Reactive Knowledge Experience Replay* (DARKER), una metodologia innovativa concepita per colmare questo divario. Combinando la distillazione della "*dark knowledge*" con il replay selettivo della memoria storica su un'architettura a doppio buffer, questa tecnica implementa una strategia di Data Compensation che interviene direttamente sul delicato bilanciamento tra stabilità e plasticità del modello, preservando le informazioni già apprese senza perdere la capacità di adattarsi a nuovi attacchi.

Nel capitolo che segue viene analizzato il modello realizzato, strutturando l'esposizione secondo

le fasi logiche, osservabili nel diagramma a blocchi in Figura 3.1, che hanno caratterizzato il workflow sperimentale: dalla fase di Data Understanding e Pre-processing iniziale alla progettazione dell'architettura MLP, per giungere, infine, alle fasi di implementazione, training e validazione finale dei risultati, fornendo così una visione d'insieme dell'intero processo di ricerca.

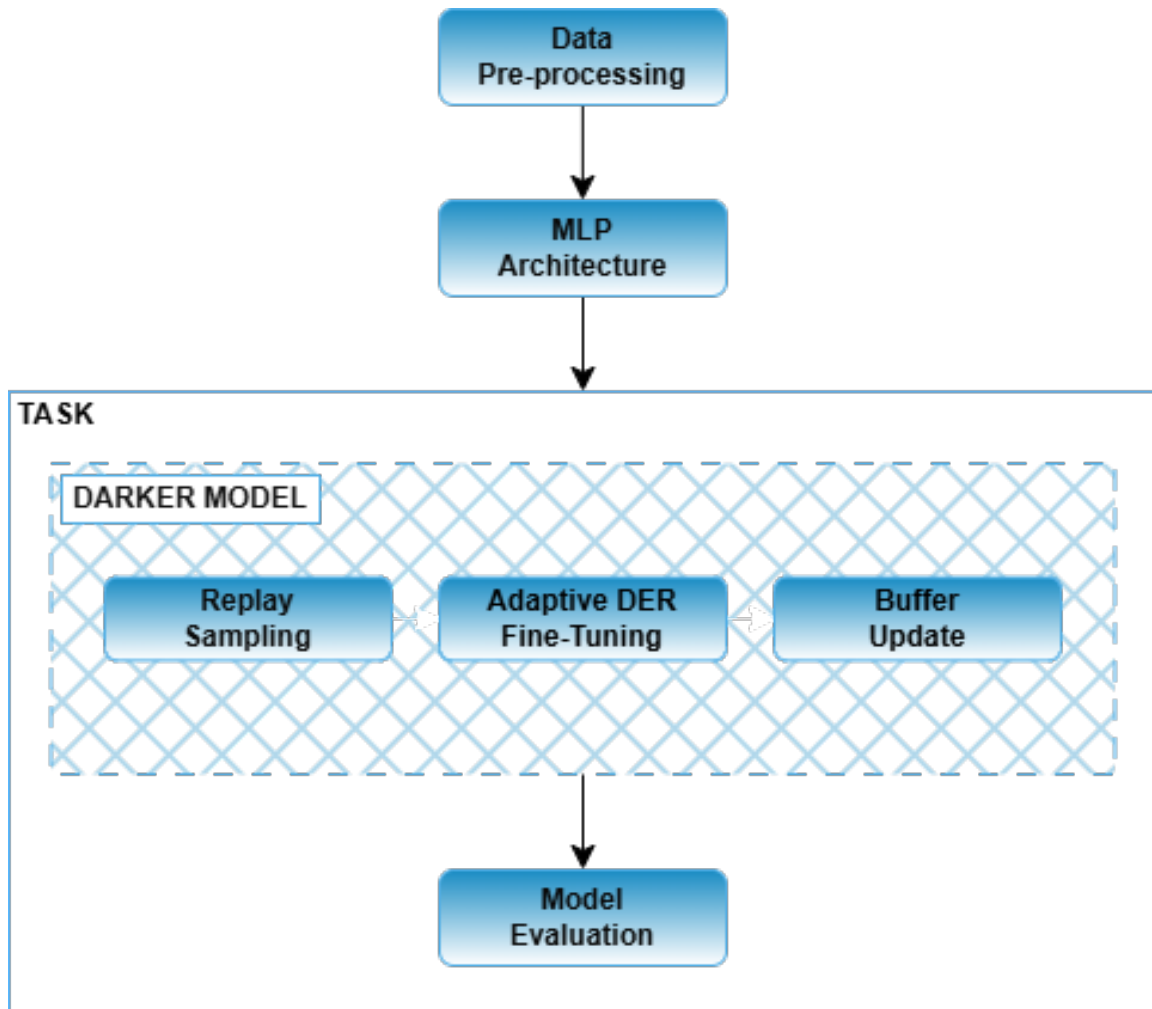


Figura 3.1: Fasi dell'implementazione di DARKER

3.1 Data Pre-processing

Per l'implementazione di DARKER sono stati utilizzati due dataset in particolare, ovvero il CICIDS2017, rilasciato dal Canadian Institute for Cybersecurity (CIC) dell'Università di New Brunswick, Canada[33], e il ToN-IoT, rilasciato dal gruppo di ricerca sulla sicurezza informatica dell'Università del New South Wales (UNSW) di Canberra, Australia[34]. Questi due dataset sono stati preferiti perché la differenza tra le loro distribuzioni e tra la loro tassonomia di attacco

consente di osservare il comportamento di DARKER in scenari distinti, in cui il Concept Drift si manifesta con caratteristiche diverse.

La scelta dei dataset in ambito IDS riveste un'importanza cruciale per poter validare i risultati ottenuti. Il CICIDS2017 rappresenta uno dei dataset di riferimento per la ricerca sugli IDS basati su Continual Learning[21][22]. Esso è stato generato in modo da avere una prevalenza significativa di traffico benigno e una distribuzione degli attacchi a sua volta asimmetrica: alcune categorie, come PortScan e DoS Hulk, raggiungono centinaia di migliaia di campioni, mentre altre, come Heartbleed, sono rappresentate da un numero minimo di istanze nell'intero dataset. Nel Continual Learning questo sbilanciamento mette in difficoltà qualunque strategia basata sulla memoria episodica perché la probabilità che istanze delle classi rare vengano campionate nel buffer diminuisce all'aumentare della cardinalità dei campionamenti, rendendo il loro ricordo nel tempo più fragile rispetto alle classi maggioritarie[33]. A differenza del primo, il ToN-IoT è il risultato di una modellazione artificiale di un traffico di rete che replica le condizioni operative di una rete industriale, ottenuta dall'interazione di tecnologie di complessità differenti. Lo sbilanciamento tra le classi è presente anche nel ToN-IoT, ma con una distribuzione differente rispetto al CICIDS2017: il traffico benigno è ben rappresentato ma non prevale nettamente su tutte le minacce presenti nel dataset, poiché sono presenti alcune categorie, come Scanning o DDoS, che sono sovrarappresentate[34].

Nella fase di pre-processing, per entrambi i dataset utilizzati per l'implementazione del modello, disponibili sotto forma di diversi file in formato CSV, è stato creato un dizionario *columns* per la standardizzazione dei nomi, visibile in Figura 3.2. In esso il nome di ciascuna colonna osservata viene associato al nome descrittivo utilizzato nella fase di addestramento, garantendo l'unicità di ciascuna colonna, in modo che ogni task condivida con i precedenti la stessa rappresentazione dello spazio delle feature in modo che il replay dei campioni storici risulti coerente con la distribuzione corrente[13].

```

columns = {
    "src_ip": "Source IP",
    "src_port": "Source Port",
    "dst_ip": "Dest IP",
    "dst_port": "Dest Port",
    'source_ip': 'Source IP',
    'source_port': 'Source Port',
    'destination_ip': 'Dest IP',
    'destination_port': 'Dest Port',
    "proto": "Protocol",
    "service": "Service",
    "duration": "Duration",
    "src_bytes": "Source bytes",
    "dst_bytes": "Dest bytes",
    "conn_state": "Connection State",
    "missed_bytes": "Missed bytes",
    "src_pkts": "Source packets",
    "src_ip_bytes": "Source IP bytes",
    "dst_pkts": "Dest packets",
    "dst_ip_bytes": "Dest IP bytes",
    "dns_query": "DNS query",
    "dns_qclass": "DNS qclass",
    "dns_qtype": "DNS qtype",
    "dns_rcode": "DNS rcode",
    "dns_AA": "DNS AA",
    "dns_RD": "DNS RD",
    "dns_RA": "DNS RA",
    "dns_rejected": "DNS rejected",
    "ssl_version": "SSL version",
    "ssl_cipher": "SSL cipher",
    "ssl_resumed": "SSL resumed",
    "ssl_established": "SSL established",
    "ssl_subject": "SSL subject",
    "ssl_issuer": "SSL issuer",
    "http_trans_depth": "HTTP transaction depth",
    "http_method": "HTTP method",
    "http_uri": "HTTP uri",
    "http_referrer": "HTTP referrer",
    "http_version": "HTTP version",
    "http_request_body_len": "HTTP request body len",
    "http_response_body_len": "HTTP response body len",
    "http_status_code": "HTTP status code",
    "http_user_agent": "HTTP user agent",
    "http_orig_mime_types": "HTTP orig mime types",
    "http_resp_mime_types": "HTTP resp mime types",
    "weird_name": "Weird name",
    "weird_add1": "Weird add1",
    "weird_notice": "Weird notice",
    "label": "Label",
    "type": "Category"
}

```

Figura 3.2: Dizionario per la standardizzazione del dataset ToN-IoT

Dopo la standardizzazione dei nomi, è stato creato un array *columns_to_drop* contenente i nomi delle colonne da rimuovere dal dataframe perché contengono informazioni di carattere identificativo, come si può osservare in Figura 3.3.

```
columns_to_drop = [  
    "uid",  
    "Label",  
    "Source IP",  
    "Source Port",  
    "Dest IP",  
    "Dest Port",  
    "Source bytes",  
    "Source IP bytes",  
    "Dest bytes",  
    "Dest IP bytes",  
    "Timestamp"  
]
```

Figura 3.3: Colonne da rimuovere nel pre-processing del dataset ToN-IoT

La rimozione di queste feature corrisponde ad una necessità documentata in letteratura: la colonna "uid" contiene gli identificatori univoci di flusso, assegnati nel momento in cui viene processata ogni connessione, ed è fondamentale rimuoverla perché contiene valori da cui il modello non imparerebbe nulla e che invece introdurrebbero un rumore casuale nella rappresentazione interna che interferisce con l'addestramento; la colonna "Label" contiene una classificazione binaria tra traffico normale e traffico malevolo, che, come la colonna precedente, non fornirebbe informazioni utili per l'apprendimento; le colonne "Source IP", "Dest IP", "Source Port", "Dest Port", "Source IP bytes", "Dest IP bytes", "Source bytes", "Dest bytes", contengono rispettivamente le informazioni relative agli indirizzi IP, alle corrispondenti porte e ai byte scambiati in ciascuna direzione che, se venissero utilizzate per l'addestramento, potrebbero fungere da scorciatoie per i classificatori, i quali rischierebbero di imparare a distinguere il traffico benigno da quello malevolo in base ad attributi specifici della rete di test anziché sulle proprietà comportamentali dei flussi[35]; la colonna "Timestamp" contiene un'informazione legata al momento di cattura della connessione e la sua inclusione nel dataframe renderebbe il modello sensibile all'ordine temporale, con il rischio di introdurre dipendenze tra l'istante temporale e l'etichetta di un attacco[35], causando una perdita considerevole di accuratezza del modello nel tempo, indipendente dal vero drift semantico generato dalle informazioni del traffico di rete.

Tra le varie operazioni di pre-processing, oltre la rimozione delle colonne non rilevanti, vi sono la sostituzione dei valori infiniti presenti nel dataset con NaN (Not a Number) e la successiva eliminazione, insieme a quella dei record incompleti. La conservazione di questi elementi avrebbe introdotto valori critici nello spazio delle feature, distorcendo le distribuzioni e producendo campioni non rappresentativi all'interno della memoria episodica del modello. In secondo luogo,

tutte le feature categoriche sono state codificate attraverso *LabelEncoder*, uno strumento della libreria "scikit-learn" che assegna a ciascun valore osservato un intero univoco, permettendo al modello di processare questi attributi come fossero delle feature numeriche[21]. Si è preferito l'utilizzo di questo strumento rispetto ad altri, come One-Hot Encoding, per evitare un'eccessiva espansione dello spazio delle feature e un costo computazionale incrementale che avrebbe causato a sua volta un ritardo nel replay dei dati. Infine, per ogni dataframe è stata utilizzata la funzione *df_shrink* della libreria "fastai.tabular.core", la quale ottimizza l'uso di memoria di un dataframe convertendo le colonne numeriche in tipi di dato più piccoli, senza perdere informazioni. A completamento della fase di pre-processing dei dataframe estratti dai file CSV, vi è la rimozione delle righe duplicate poiché la loro presenza distorcerebbe la distribuzione della classi nello spazio delle feature.

3.2 Architettura Rete Neurale

Per la realizzazione di DARKER è stata implementata un'architettura **MLP (Multilayer Perceptron)**, ovvero una rete neurale feedforward in cui ogni layer riceve in input le informazioni dal precedente e in cui l'addestramento avviene tramite backpropagation. La capacità di un MLP di modellare relazioni tra feature di input e classi di output deriva dalla composizione di più trasformazioni non lineari successive, ognuna delle quali introduce un livello aggiuntivo di astrazione rispetto alla precedente, catturando pattern che non sarebbero individuabili con una classificazione lineare. La scelta di questa architettura risponde alla natura dei dati utilizzati: un MLP risulta più efficiente sui dati tabulari se confrontata con una *CNN (Convolutional Neural Network)* perché non impone strutture relazionali sui dati di input e per questo motivo risulta anche la scelta più utilizzata in letteratura. Oltre a questo, la sua struttura sequenziale di layer densi si adatta in modo naturale al contesto del Continual Learning perché il layer di output del modello può essere espanso progressivamente all'arrivo di nuove classi di attacco senza modifiche all'architettura interna.

Per l'implementazione dell'architettura MLP è stata realizzata la funzione ***build_model()***, dove l'input del modello è costituito da feature di traffico di rete rappresentate da attributi numerici precedentemente codificati. L'architettura è organizzata in tre blocchi nascosti con larghezza decrescente: 512, 256 e 128 neuroni. Questa struttura ad imbuto riflette una logica di compressione progressiva delle rappresentazioni, dove i layer più grandi e più vicini all'input hanno

il compito di estrarre feature di basso livello del traffico, mentre i layer successivi operano su rappresentazioni sempre più astratte. In questo modo si riesce a catturare le proprietà strutturali delle classi di attacco in modo sempre più discriminativo, riducendo al tempo stesso la dipendenza dai dettagli specifici del singolo campione. Questo comportamento è fondamentale in un sistema di replay dove i campioni storici devono essere in grado di produrre un segnale di distillazione significativo anche dopo aver osservato molti task[7].

I primi due blocchi seguono la stessa sequenza: *Dense* $\rightarrow$ *LayerNormalization* $\rightarrow$ *Dropout*. La funzione di attivazione *ReLU (Rectified Linear Unit)*[36], all'interno dello strato Dense, garantisce la non linearità necessaria perché la rete non collassi in un semplice classificatore lineare, azzerando i neuroni che producono un valore negativo. In secondo luogo, la regolarizzazione *L2*, applicata ai kernel di ciascuno strato denso con coefficiente *l2* pari a $1e-3$, aggiunge un termine di penalità alla funzione di perdita che ostacola la creazione di pesi molto grandi su singoli neuroni, distribuendo in modo più uniforme il contributo delle feature di input e riducendo il rischio di overfitting in presenza di classi molto sbilanciate.

Una caratteristica distintiva nell'architettura MLP del modello proposto nella tesi è l'utilizzo di *LayerNormalization*, rispetto alla più comune BatchNormalization. Quest'ultima calcola la media e la varianza sulla dimensione del batch durante l'addestramento e mantiene delle statistiche "in movimento" che vengono utilizzate per la fase di test. In un contesto di apprendimento sequenziale non stazionario, come quello del rilevamento di minacce sul traffico di rete, queste statistiche rischiano di diventare troppo sbilanciate verso la distribuzione del task più recente, amplificando così le differenze di distribuzione e contribuendo al Catastrophic Forgetting. Al contrario, la LayerNormalization normalizza lungo la dimensione delle feature per ogni singolo campione, rimanendo indipendente dalle statistiche di distribuzione del task corrente e producendo un segnale di normalizzazione coerente anche quando i campioni provengono da distribuzioni diverse, una caratteristica fondamentale in un sistema di replay dove ogni batch di addestramento contiene campioni del task corrente e campioni estratti dal buffer episodico.

Il *Dropout* viene applicato solo nei primi due blocchi, con tassi decrescenti di 0.2 e 0.1. Questa asimmetria verso il terzo blocco è stata pensata perché i layer di estrazione e compressione operano su uno spazio di rappresentazione ancora ampio, in cui il numero elevato di parametri può portare il modello a costruire dipendenze specifiche dal singolo campione anziché generalizzare sulle proprietà della sua classe di appartenenza. Il Dropout tenta di contrastare questo rischio disattivando casualmente una parte dei neuroni durante ogni passo di addestramento, forzando il

modello a sviluppare rappresentazioni più robuste. Il terzo blocco, quello di affinamento, riceve rappresentazioni già compresse e stabilizzate e l'aggiunta di Dropout rischierebbe di disperdere le distinzioni minime tra le classi, penalizzando proprio la funzione di affinamento di questo blocco. Per la stessa ragione, anche la LayerNormalization viene omessa nel terzo blocco nascosto perché le rappresentazioni in ingresso sono già normalizzate dall'uscita del secondo blocco e una seconda normalizzazione rischierebbe di alterare la scala che il classificatore utilizzerà per costruire il confine decisionale.

Il layer finale non presenta alcuna funzione di attivazione, ma è un semplice strato Dense che produce logits grezzi, ovvero i valori di attivazione non normalizzati prima di qualsiasi trasformazione probabilistica. La motivazione di questa scelta è legata all'utilizzo dei logits nel meccanismo di distillazione di DARKER, per cui nel buffer episodico vengono conservati sia i campioni di input che i logits prodotti dal modello al momento della prima osservazione di ciascun campione[25]. Lavorare con i logits, e non con le probabilità Softmax, ad esempio, permette di preservare la geometria interna dello spazio decisionale del modello, catturando in modo molto più fedele il modo in cui esso distribuisce l'incertezza tra le classi nel momento in cui il campione viene visto per la prima volta.

La funzione di perdita scelta per la compilazione del modello è la *SparseCategoricalCrossentropy* che si distingue dalla classica CategoricalCrossentropy per il fatto di accettare le etichette come interi scalari invece di vettori one-hot, eliminando la necessità di una trasformazione preliminare delle etichette e riducendo il consumo di memoria in presenza di un numero elevato di classi. L'algoritmo di ottimizzazione utilizzato è *Adam* con un learning rate di $1e-3$ che, rispetto ad altri valori più conservativi, permette al modello di convergere più rapidamente su ciascun task, riducendo il numero di epoche necessarie e il costo computazionale complessivo dell'addestramento. La combinazione di un algoritmo di ottimizzazione più reattivo e di una forte regolarizzazione L2 costituisce una soluzione ottimale, in cui la velocità di adattamento del task corrente è bilanciata da un vincolo strutturale che preserva la stabilità delle rappresentazioni tra un task e l'altro.

Il numero delle classi che il modello deve distinguere non è fisso e cresce nel tempo man mano che nuove tipologie di attacco vengono introdotte dai task successivi. In un classificatore standard, il layer di output ha una dimensione fissa pari al numero di classi viste durante l'addestramento e non può essere modificata senza ricostruire il modello. Per affrontare questo problema in modo compatibile con la logica del Continual Learning è stata implementata la

funzione *expand_model()* per gestire l'espansione dello spazio di output del classificatore ogni volta che vengono introdotte delle classi mai viste prima. La logica dietro l'operazione di espansione si articola in tre fasi successive. Nella prima, viene costruito un nuovo modello con la stessa architettura interna del precedente ma con un layer di output di dimensione pari al numero totale di classi osservate fino al task corrente, calcolato come la somma delle classi già note e delle nuove classi introdotte in quel task. Nella seconda fase, i pesi dell'intero backbone vengono trasferiti dal vecchio modello al nuovo. Il trasferimento avviene escludendo l'ultimo layer, ovvero quello del classificatore, e copiando in ordine i pesi di tutti i blocchi intermedi. In questo modo, la parte del modello che si occupa della trasformazione delle feature di input nelle rappresentazioni interne su cui poi opera il classificatore rimane invariata dopo l'espansione, garantendo il mantenimento integrale della conoscenza accumulata sui pattern di traffico osservati nei task precedenti. La conservazione delle conoscenze pregresse è la condizione necessaria affinché l'espansione non provochi di per sé una forma di forgetting sulle classi vecchie. Infine, la terza fase riguarda il trasferimento selettivo di pesi del layer di output. Un layer Dense è parametrizzato da una matrice di pesi W e da un vettore di bias b , quindi, quando il numero di classi aumenta, occorre espandere entrambi. Per far ciò, come prima operazione vengono recuperati la matrice di pesi e il vettore di bias dall'output layer del nuovo modello realizzato nelle fasi precedenti. Successivamente, nelle prime old_n colonne della nuova matrice new_W viene copiata la matrice old_W del vecchio modello e nei primi old_n elementi del nuovo vettore new_b viene copiato il vettore old_b del vecchio modello. In questo modo, il confine decisionale per le classi già viste è preservato com'era prima dell'arrivo di nuove classi, mantenendo gli stessi pesi e gli stessi bias e producendo gli stessi logits, senza che venga introdotto alcun rumore dall'espansione. I neuroni corrispondenti alle nuove classi, invece, mantengono l'inizializzazione casuale della prima fase e verranno modellati dall'addestramento del task corrente. Questa strategia di espansione con trasferimento selettivo dei pesi produce sempre un output che copre l'intero spazio delle classi osservate in tempo reale senza necessitare di informazioni sull'identità del task durante la fase di test.

La combinazione delle due funzioni definisce un ciclo di vita del modello in grado di gestire la rilevazione di nuove classi tramite l'estensione del layer di output e la preservazione integrale delle conoscenze pregresse. Questa integrazione, quindi, assicura di ottenere un'architettura che si aggiorni costantemente adattandosi al numero corrente di classi e al volume dei dati.

3.3 Model Implementation

Il nucleo dell'implementazione di DARKER nasce da un'importante constatazione, ovvero che il replay episodico classico e la distillazione della conoscenza, se presi singolarmente, affrontano il Concept Drift e, quindi, il Catastrophic Forgetting da angolazioni complementari ma parziali. Il replay classico espone il modello ai campioni storici reintroducendo il segnale di supervisione delle classi già apprese [37], ma vincola il modello soltanto alla correttezza della predizione finale sui campioni del buffer, senza imporre alcun vincolo sulla struttura dello spazio decisionale. Questo limite può essere superato affiancando al replay un secondo segnale di supervisione costruito sui logits che il modello produceva sui campioni storici nel momento in cui li ha osservati per la prima volta e non sulle etichette delle classi[25]. I logits codificano non solo la classe predetta ma anche l'intera distribuzione dell'incertezza tra le classi e il loro utilizzo permette di preservare la geometria dello spazio decisionale e non solo il suo esito finale. DARKER supera questa dicotomia integrando le due strategie in un'unica architettura a doppio buffer: il *Replay Buffer*, che conserva coppie (x_i, y_i) e alimenta il replay episodico, e il *DER Buffer*, che conserva triple (x_i, y_i, z_i) , dove z_i è il vettore di logits prodotto dal modello alla prima osservazione di x_i , e che alimenta la componente di distillazione. I due buffer agiscono in parallelo durante ogni task: il primo garantisce che il modello continui a classificare correttamente i campioni storici, il secondo garantisce che la classificazione avvenga mantenendo la stessa struttura decisionale interna che aveva costruito alla prima rilevazione.

3.3.1 Replay selettivo della memoria episodica

La costruzione del Replay Buffer è ottenuta dalla funzione *update_replay_buffer()*, la quale è stata implementata tramite il *Reservoir Sampling*[38][25], un algoritmo di campionamento casuale che garantisce che ogni campione osservato nell'intera sequenza di task abbia la stessa probabilità di essere presente nel buffer al termine dell'aggiornamento, indipendentemente dall'ordine di arrivo o dal numero di dati processati. Per ogni nuovo campione (x_i, y_i) estratto dal task corrente occorre fare una distinzione tra due differenti casistiche:

1. se il buffer non ha ancora raggiunto la capacità massima, il campione viene aggiunto direttamente in coda
2. se il buffer è pieno, viene generato un intero casuale j nell'intervallo $[0, seen+1)$, dove *seen* è il contatore cumulativo di tutti i campioni osservati dall'inizio dell'addestramento,

e se il valore casuale è inferiore alla capacità massima del buffer, allora il campione in posizione j viene sostituito dal campione corrente

L'utilizzo del denominatore $seen+1$ invece della capacità massima del buffer è la proprietà distintiva del Reservoir Sampling, la quale assicura che la probabilità di ogni campione di essere inserito nel buffer diminuisca proporzionalmente al numero di campioni già osservati, generando così una distribuzione uniforme sull'intera sequenza storica. Rispetto ad altre strategie di selezione, questa procedura ha complessità $O(1)$ per ogni aggiornamento e non richiede la valutazione del modello sui campioni storici, una proprietà fondamentale in un contesto operativo in cui la latenza causata dagli aggiornamenti è vincolante.

Una volta popolato, il Replay Buffer viene campionato dalla funzione *sample_replay_buffer()* attraverso una strategia di campionamento stratificato. La campionatura casuale pura risulta inadeguata in questo contesto perché in un buffer che riflette la distribuzione naturale del traffico di rete le classi di attacco rare tendono ad essere sotto-campionate nei batch di replay, producendo un segnale di supervisione asimmetrico che accelera il forgetting proprio sulle classi più critiche per la sicurezza di un sistema[32]. La funzione opera in due fasi sequenziali distinte:

1. per ogni classe c_k presente nel buffer, viene selezionato un insieme di min_sample campioni dalla sua partizione, garantendo una quota minima per classe indipendentemente dalla sua frequenza
2. i campioni rimanenti dalla prima selezione fino alla dimensione massima complessiva $n_samples$ vengono selezionati casualmente dalla porzione del buffer non ancora inclusa, senza distinzione di classe

La combinazione di queste due fasi genera in output un batch di replay in cui tutte le classi storiche, comprese quelle minoritarie, contribuiscono con un numero minimo garantito di campioni al segnale di supervisione, riducendo l'asimmetria che alimenta il forgetting selettivo sulle classi rare. Questo campionamento stratificato nella prima fase garantisce un peso minimo per tutte le classi, mentre nella seconda fase cerca di mantenere fedelmente la distribuzione reale del traffico di rete per la porzione rimanente, evitando di distorcerla completamente per non generare un batch che risulti troppo distante dalla realtà operativa.

3.3.2 Distillazione adattiva dei logits

La componente di distillazione all'interno di DARKER è stata ottenuta tramite l'integrazione di diverse funzioni, ognuna delle quali con un compito ben specifico.

La costruzione del DER Buffer è stata realizzata tramite la funzione *update_der_buffer()*. Essa risulta più articolata di quella utilizzata per la realizzazione del Replay Buffer, poiché deve mantenere una distribuzione per classe quanto più bilanciata possibile, in modo che il segnale di distillazione sia uniformemente distribuito su tutte le classi storiche durante l'intero addestramento. A caratterizzare questa funzione vi sono l'utilizzo di una struttura ausiliaria, definita come *der_class_pos*, ovvero un dizionario che associa a ciascuna classe l'insieme delle posizioni nel buffer occupate dai suoi campioni, e l'imposizione di limite superiore al numero di campioni per classe, indicato con l'intero *limit*.

L'aggiornamento del DER Buffer si articola in tre casi distinti:

- **CASO 1:** il buffer non ha ancora raggiunto la sua capacità massima e il nuovo campione viene semplicemente accodato al suo interno.
- **CASO 2:** il buffer ha raggiunto la capacità massima, ma la classe a cui appartiene il nuovo campione è ancora sotto il limite superiore. In questo caso, vengono identificate tutte le classi che superano il limite, definite come *overflow_classes*, viene selezionata quella con il maggiore eccesso rispetto al limite e uno dei suoi campioni viene rimpiazzato con il nuovo, spostando la distribuzione del buffer verso un equilibrio più uniforme. In assenza di classi sovrarappresentate, la sostituzione avviene in modo casuale sull'intero buffer.
- **CASO 3:** il buffer ha raggiunto la capacità massima e la classe a cui appartiene il nuovo campione ha anch'essa raggiunto il suo limite superiore. In quest'ultimo caso, viene applicata una sostituzione intra-classe, ovvero uno dei campioni già presenti per quella classe viene rimpiazzato casualmente con quello nuovo, mantenendo invariata la cardinalità della classe e la distribuzione del buffer fino a quel momento.

L'obiettivo finale di questa logica è preservare un segnale di distillazione bilanciato su tutte le classi nel tempo poiché una distribuzione sbilanciata del DER Buffer amplificherebbe il vincolo di distillazione sulle classi sovrarappresentate, riducendo l'efficacia del termine di distillazione per le classi minoritarie, producendo così forgetting in maniera asimmetrica.

In modo analogo al Replay Buffer, dopo essere stato popolato, il DER Buffer viene campionato dalla funzione *sample_der_buffer()*, attraverso la stessa strategia di campionamento stratificato,

con l'unica differenza che ogni campione estratto include anche il vettore di logits z_i memorizzato al momento della sua prima osservazione. Questa tripla (x_i, y_i, z_i) costituisce l'input per il termine di distillazione nella funzione di perdita.

Il cuore computazionale di DARKER è costituito dalla funzione *der_train_step()*. Ad ogni passo, essa ottimizza la seguente funzione di perdita:

$$\mathcal{L}_{\text{DARKER}}(\theta) = \mathcal{L}_{\text{CE}}(f_{\theta}(x_{\text{cur}}), y_{\text{cur}}) + \alpha \cdot \mathcal{L}_{\text{DER}}(f_{\theta}(x_{\text{rep}}), z_{\text{rep}}) \quad (3.1)$$

dove $\mathcal{L}_{\text{CE}}$ è la Cross-Entropy sul batch corrente e $\mathcal{L}_{\text{DER}}$ è il termine di distillazione pesato dal coefficiente α .

Nella funzione i gradienti di entrambi i termini vengono calcolati tramite un'unica *Gradient-Tape()*, garantendo che l'aggiornamento dei pesi del modello tenga conto simultaneamente del segnale di apprendimento sul task corrente e del vincolo di coerenza sui campioni storici. Il termine $\mathcal{L}_{\text{CE}}$ viene calcolato eseguendo una *forward pass* del modello sul batch corrente, in modo che Dropout e LayerNormalization si comportino come durante la fase di addestramento; invece, il termine $\mathcal{L}_{\text{DER}}$ è definito come l'errore quadratico medio tra i logits prodotti dal modello corrente sui campioni del DER Buffer e i logits memorizzati z_{rep} :

$$\mathcal{L}_{\text{DER}} = \frac{1}{|B_{\text{rep}}|} \cdot \sum_{(x_i, z_i) \in B_{\text{rep}}} \|f_{\theta}(x_i) - z_i\|_2^2 \quad (3.2)$$

La scelta di applicare il *Mean Squared Error (MSE)* sui logits grezzi invece di una normalizzazione Softmax è stata preferita poiché il MSE preserva la direzione e la dimensione della risposta del modello, catturando con maggiore fedeltà il modo in cui il modello distribuisce l'incertezza tra le classi nell'istante in cui il campione è stato osservato per la prima volta[25], a differenza della Softmax che avrebbe proiettato i logits su una distribuzione di probabilità, comprimendo l'informazione relativa all'intensità della risposta del modello. Il fondamento teorico alla base della realizzazione di questa funzione risiede nello sfruttare il concetto di "*dark knowledge*"[25]: i logits di un modello non codificano solo la classe predetta, ma anche le relazioni di similitudine tra tutte le classi presenti, riflettendo quanto il modello ritenga corretta ciascuna alternativa per un dato input. Preservare questa distribuzione permette di preservare anche la struttura relazionale dello spazio decisionale nello stesso modo in cui era stata costruita al momento della prima osservazione, impedendo che le nuove classi introdotte dai task successivi alterino i confini decisionali consolidati in precedenza.

Un'ulteriore caratteristica che distingue DARKER dall'implementazione originale di DER è l'adattamento dinamico del coefficiente α attraverso la funzione `get_der_alpha()`. Questo coefficiente viene solitamente utilizzato come un iperparametro fisso scelto prima dell'addestramento e questa scelta può risultare inadeguata in un contesto non stazionario poiché un valore costante non è in grado di adattarsi alla variabilità del drift tra task consecutivi, rischiando di sottostimare il vincolo di distillazione nei momenti in cui il forgetting è più elevato.

In DARKER, il coefficiente α viene calcolato dinamicamente al termine di ogni task secondo la relazione:

$$\alpha_t = \text{clip}\left(\alpha_{\min} + F_{t-1}^{\text{Recall}}, \alpha_{\min}, \alpha_{\max}\right) \quad (3.3)$$

dove F_{t-1}^{Recall} è il valore di forgetting medio sulla Recall misurato al termine del task precedente e $[\alpha_{\min}, \alpha_{\max}]$ è l'intervallo di variazione consentito, i cui estremi sono impostati rispettivamente a 0.3 e 0.99. Quando il forgetting sulla Recall è elevato, α cresce e il termine di distillazione esercita un vincolo più stringente sul modello, frenando il drift delle rappresentazioni interne verso le nuove classi e offrendo maggiore stabilità; invece, quando il forgetting è basso, α rimane vicino al suo minimo e la funzione di perdita concede al modello una maggiore plasticità per adattarsi al task corrente. La funzione `clip` garantisce semplicemente che il coefficiente rimanga all'interno dell'intervallo ammissibile per cercare di preservare la stabilità numerica dell'ottimizzazione anche in scenari di drift estremo.

La scelta di utilizzare la Recall come segnale guida per questa logica, anziché la più comunemente controllata Accuracy, è motivata dalla natura del problema. In un sistema IDS la metrica più critica è costituita dalla sensibilità del modello verso le classi di attacco già apprese[2] e non la percentuale di classificazioni corrette. Un incremento di forgetting sulla Recall segnala che il modello sta perdendo la capacità di rilevare minacce già conosciute e questo rappresenta una grave vulnerabilità, tale da giustificare un intervento immediato sul peso della distillazione.

3.4 Model Training

La fase di addestramento di DARKER è orchestrata da un training loop sequenziale che itera sulla sequenza di task in ordine temporale, replicando le condizioni operative di un sistema di rilevamento delle minacce in produzione in cui i dati arrivano in flusso continuo e il modello non può accedere anticipatamente all'intera storia del traffico. In questa sezione, prima di

analizzare nel dettaglio le diverse fasi del training loop, viene descritta la fase di inizializzazione degli iperparametri, la quale determina il comportamento complessivo del sistema lungo l'intera sequenza di task.

Ogni task elabora un blocco di lunghezza pari a 100.000 campioni del dataset, una dimensione scelta per bilanciare la granularità dell'apprendimento del modello con la disponibilità di dati sufficienti a sviluppare rappresentazioni stabili per le classi di ciascun task. Il numero di epoche di addestramento per task è fissato a 8, un valore che permette al modello di convergere su ciascun task senza richiedere un numero eccessivo di iterazioni che aumenterebbe il costo computazionale complessivo.

La dimensione del Replay Buffer è fissata al 20% della dimensione del task, mentre la dimensione del batch di campionamento stratificato è pari al 30% della memoria episodica. Le stesse proporzioni sono state applicate al DER Buffer, in modo da mantenere un rapporto bilanciato tra i campioni del task corrente e quelli estratti dai due buffer durante ciascun batch di addestramento. La proporzione scelta per la memoria episodica costituisce un compromesso tra la fedeltà della rappresentazione storica e il costo di memoria, in linea con le scelte adottate in letteratura per scenari con un numero elevato di task[25].

Il coefficiente iniziale di distillazione è fissato a 0.3, corrispondente al valore minimo dell'intervallo di variazione della funzione *get_der_alpha()*, descritta nella sezione precedente. Questo valore viene utilizzato solo per il primo task, poiché per tutti i task successivi il coefficiente viene ricalcolato dinamicamente in funzione del forgetting sulla Recall osservato al task precedente. La normalizzazione delle feature è realizzata tramite l'algoritmo **MinMaxScaler()**, con "feature_range" pari a (0, 1) e "clip" posto a *true*, in modo che i valori vengano compressi nell'intervallo unitario e che eventuali valori anomali fuori dalla distribuzione di addestramento non producano attivazioni fuori scala nella rete.

La lista dei task viene costruita prima dell'avvio dell'addestramento suddividendo l'intero dataframe in blocchi contigui. Questa suddivisione preserva l'ordine temporale originale del traffico, garantendo che ciascun task rappresenti una finestra temporale distinta e che la sequenza di apprendimento rispecchi il flusso reale dei dati.

All'inizio di ogni task, il training loop analizza le etichette del blocco corrente e aggiorna la struttura *class_to_index*, un dizionario che mantiene la mappatura tra il nome originale di ciascuna classe di attacco e un indice progressivo. Se una classe presente nel task corrente non è ancora stata osservata in nessun task precedente, le viene assegnato un nuovo indice e il

suo nome viene aggiunto alla lista *index_to_class*. Questa struttura cresce lungo la sequenza di task e definisce in ogni istante il vocabolario completo delle classi conosciute dal modello, che corrisponde alla dimensione attuale del layer di output. Le etichette originali del task corrente vengono poi convertite nel corrispondente vettore di indici interi *y_mapped*, che costituisce il vettore di supervisione utilizzato per il calcolo della funzione di perdita.

Poi, i dati di ogni task vengono suddivisi in training set e test set tramite *train_test_split()* con proporzioni 80/20 e il parametro "shuffle" posto a *false*. La scelta di disattivare lo shuffle è data dal fatto che ogni task rappresenta un blocco temporalmente ordinato del traffico di rete e mescolare i campioni prima della partizione introdurrebbe una contaminazione temporale tra i due set di dati, rendendo la valutazione non rappresentativa delle condizioni operative reali in cui il modello viene aggiornato su dati passati e testato su dati futuri.

La normalizzazione viene applicata dopo la partizione: lo *scaler* viene adattato, tramite il metodo *fit()*, solo sul training set del primo task e da quel momento in poi viene applicato con il metodo *transform()* nello stesso modo a tutti i task successivi, inclusi i campioni estratti dal Replay Buffer, poiché anche questi ultimi vengono conservati nella loro forma grezza non normalizzata. In questo modo, si garantisce che tutti i campioni vengano proiettati nello stesso spazio delle feature su cui il modello è stato addestrato in origine.

Prima dell'inizio della fase di addestramento sul task corrente, vi è la fase del replay selettivo della memoria storica in cui il Replay Buffer viene campionato tramite la funzione *sample_replay_buffer()* e i campioni estratti, dopo la normalizzazione, vengono concatenati al training set corrente. Questa operazione produce il training set effettivo su cui il modello verrà addestrato, composto da campioni del task corrente e da campioni storici provenienti dai task precedenti. Il Replay Buffer viene aggiornato con i dati grezzi del task corrente tramite il metodo *update_replay_buffer()* dopo la concatenazione, in modo che i campioni storici campionati in precedenza non vengano sovrascritti prima del loro utilizzo nell'addestramento.

Dopo l'aggiornamento del training set avviene la costruzione ed espansione del modello che segue una logica incrementale, coerente con il principio del Class-Incremental Learning in cui il numero delle classi non è noto a priori ma viene aggiornato nel corso della sequenza di task. Al primo task, quando il modello non è stato ancora inizializzato, viene invocata la funzione *build_model()* con il numero di feature del training set e il numero di classi osservate fino a quel momento. A partire dal secondo task e per tutti i task successivi, invece, viene invocata la funzione *expand_model()* per estendere il layer di output preservando i pesi appresi in precedenza.

Questa strategia è più fedele alle condizioni operative reali rispetto ad un’inizializzazione con il numero totale delle classi, la quale richiederebbe una conoscenza anticipata dell’intero spazio delle etichette, informazione non disponibile a priori in un ambito in cui nuove categorie di attacco possono emergere in qualsiasi momento.

L’ultima fase analizzata in questa sezione, ovvero quella dell’addestramento del modello, è costituita da più fasi. Innanzitutto, il training set aggiornato tramite il replay selettivo viene infine convertito in un oggetto *tf.data.Dataset* e rimescolato in ordine casuale, in modo che l’ordine dei campioni venga rideterminato ad ogni epoca. Il mescolamento ha il fine di evitare che il modello sviluppi dipendenze dall’ordine di presentazione dei campioni e che produca pattern di ottimizzazione non generalizzabili. Successivamente, il dataset viene suddiviso in batch di dimensione fissa.

Il valore del coefficiente *DER_ALPHA* viene determinato prima dell’inizio dell’addestramento e per tutti i task successivi viene ricalcolato dalla funzione *get_der_alpha()* in relazione al forgetting sulla Recall del task precedente. Per ciascuna delle epoche di addestramento, il loop itera su tutti i batch del dataset e per ciascuno di essi il DER Buffer viene campionato tramite il metodo *sample_der_buffer()* con una quota minima per classe pari a *DER_REPLAY_SIZE // 2*, estraendo un insieme bilanciato di campioni storici accompagnati dai rispettivi logits di riferimento. Il batch corrente e il campione del DER Buffer vengono passati in input alla funzione *der_train_step()*, la quale calcola e minimizza la funzione di perdita combinata di DARKER. Il campionamento del DER Buffer avviene ad ogni batch per garantire che il segnale di distillazione vari continuamente durante l’addestramento, esponendo il modello a insiemi diversi di campioni storici ad ogni step e riducendo il rischio che il vincolo di distillazione si concentri su un sottoinsieme fisso del buffer.

Al termine dell’addestramento, il modello viene utilizzato per calcolare i logits sull’intero training set del task corrente e i logits ottenuti, insieme alle feature e alle etichette corrispondenti, vengono passati in input al metodo *update_der_buffer()* per aggiornare il DER Buffer con i campioni del task corrente e i rispettivi vettori di riferimento. Questo aggiornamento avviene dopo il completamento dell’addestramento sul task corrente, in modo che i logits memorizzati riflettano le rappresentazioni finali del modello su quel task e non uno stato intermedio dell’ottimizzazione e non generino un segnale di distillazione inconsistente per i task successivi che rischierebbe di distorcere il vincolo di coerenza che il termine $\mathcal{L}_{\text{DER}}$ impone sullo spazio decisionale.

3.5 Model Evaluation

3.5.1 Metriche di valutazione

Le metriche di valutazione hanno l'obiettivo di quantificare l'accuratezza di un modello e di chiarire come viene ottenuta questa accuratezza. Esse permettono di distinguere tra modelli che mantengono le conoscenze pregresse in modo stabile e modelli che, pur raggiungendo buone prestazioni finali, tendono a degradare significativamente su task precedenti. L'efficienza di un modello di Continual Learning non può essere valutata attraverso una singola metrica, ma richiede un insieme di misure complementari, come indicatori di performance globale, metriche specifiche per la quantificazione del forgetting e misure più sensibili alla distribuzione dei dati. L'analisi di queste metriche nel loro insieme consente di ottenere una valutazione più solida e realistica del comportamento del modello, mettendone in luce sia le capacità che i limiti.

Accuracy

Il punto di partenza per ogni analisi prestazionale è l'Accuracy, poiché indica la percentuale di istanze classificate in maniera corretta. Nel Continual Learning, però, il suo significato va oltre la semplice bontà della predizione su un singolo insieme di dati; l'accuratezza finale esprime quanto il modello riesce a mantenere buone prestazioni su tutti i task appresi fino a quel momento, dopo aver incorporato nuove informazioni nel tempo.

Indicando con $a_{i,j}$ l'Accuracy ottenuta sul task j dopo l'addestramento sul task i , l'Accuracy media finale è:

$$ACC = \frac{1}{T} \sum_{j=1}^T a_{T,j} \quad (3.4)$$

Questa misura restituisce una valutazione complessiva del modello al termine dell'intero processo incrementale. Un valore elevato di Accuracy non è sufficiente, da solo, per comprendere se il modello abbia costruito o meno una conoscenza soddisfacente oppure se abbia privilegiato i task più recenti a discapito dei precedenti[24]: un modello che raggiunge un valore alto di ACC , ignorando completamente i task vecchi e agendo bene solo sull'ultimo otterrebbe comunque un punteggio decente. Per questo da sola non basta, ma serve qualcosa di più specifico.

Forgetting

Per quantificare il costo dell'apprendimento incrementale in termini di perdita di memoria, si introduce il concetto di Forgetting. Esso misura in modo diretto quanto il modello perde le conoscenze acquisite in passato dopo l'introduzione di nuovi task; quindi, misura quanto l'acquisizione di nuove informazioni interferisca con ciò che il modello aveva già memorizzato[32].

Per ogni task j , il Forgetting relativo è:

$$F_j = \max_{l \in \{1, \dots, T-1\}} a_{l,j} - a_{T,j} \quad (3.5)$$

e il Forgetting medio sull'intera sequenza:

$$F = \frac{1}{T-1} \sum_{j=1}^{T-1} F_j \quad (3.6)$$

Questa metrica misura quindi la stabilità del modello nel tempo, catturando il divario tra la migliore performance ottenuta su un task e quella al termine dell'intero addestramento. Un valore alto di F è il segnale più chiaro che l'apprendimento dei nuovi task sta sovrascrivendo le rappresentazioni interne. Il Forgetting è la metrica analizzata maggiormente negli esperimenti, poiché un modello che dimentica attacchi già visti può rappresentare una vulnerabilità reale nei sistemi di monitoraggio.

Recall

Per analizzare al meglio la capacità di un sistema di non ignorare minacce reali bisogna utilizzare la Recall, la quale aiuta a valutare la capacità del modello di riconoscere correttamente le istanze appartenenti ad una classe. La Recall è una metrica rilevante soprattutto quando l'obiettivo principale non è solo quello di minimizzare il numero totale degli errori, ma anche quello di garantire la corretta identificazione degli eventi significativi[39] e questo è proprio quello che avviene negli IDS durante il rilevamento di attacchi informatici e quello che il metodo proposto mira a garantire.

Per una classe c , la Recall è:

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \quad (3.7)$$

dove TP_c sono i *True Positives* della classe c , ovvero il numero di istanze che appartengono a quella classe e che vengono correttamente classificate dal modello, e FN_c sono i *False Negatives*, ovvero il numero delle istanze che appartengono a quella classe ma che vengono classificate male dal modello, attribuendo loro l'etichetta di un'altra classe. Nel Continual Learning questa metrica è prioritaria perché permette di verificare se il modello continua a riconoscere bene una classe anche dopo l'arrivo di nuovi task e se quest'ultima è bassa per una classe appresa, significa che il modello sta perdendo sensibilità proprio su quella classe[2].

Precision

Spostando l'attenzione sull'affidabilità delle segnalazioni, la Precision serve per valutare quanto le predizioni positive del modello siano in realtà corrette. Questa metrica misura quanto ci si possa fidare delle predizioni positive del modello: una Precision alta si ha quando il modello segnala un evento positivo che sia molto probabilmente corretto, generando pochissimi falsi allarmi, mentre una Precision bassa si ha quando il modello genera molti falsi allarmi[39].

Per una classe c , la Precision è:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \quad (3.8)$$

Nel Continual Learning questa metrica è molto importante perché una sua riduzione indica spesso che il modello sta iniziando a generalizzare in modo eccessivo o errato sui nuovi task, classificando come positivi dei pattern che in realtà appartengono a classi diverse[2]. Insieme, la Precision e la Recall rappresentano due aspetti importanti nel rilevamento delle minacce perché permettono di valutare sia la performance globale sia la qualità della distinzione tra le varie classi nel tempo, aspetto fondamentale nel rilevamento di intrusioni dove i set di dati sono quasi sempre sbilanciati.

F1-score

In un modello, la Recall e la Precision rappresentano due metriche opposte: massimizzare la Recall significa rilevare più attacchi possibili, rischiando di ottenere un numero elevato di falsi allarmi, mentre aumentare la Precision riduce i falsi positivi, ma rischia di far passare numerosi attacchi. L'F1-score è la media armonica delle due metriche e serve proprio per trovare un punto

di equilibrio. Per una classe c , l’F1-score viene definito come:

$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \quad (3.9)$$

Questa metrica è molto utile negli IDS perché valuta sia se il modello rileva correttamente gli attacchi sia quanto sono affidabili le sue segnalazioni[2]. A differenza dell’Accuracy, che può risultare poco informativa e che va interpretata con cautela in presenza di classi sbilanciate, l’F1-score restituisce una misura più equilibrata della qualità delle predizioni, limitando sia i *False Negatives*(FN) che i *False Positives*(FP)[39].

Confusion Matrix

A completamento delle metriche introdotte e qualora ci sia la necessità di un’ispezione più dettagliata degli errori commessi dal modello, la Confusion Matrix è uno strumento diagnostico per interpretare nel dettaglio il comportamento di un modello. A differenza delle metriche precedenti, le quali riassumono la performance in un singolo valore, la Confusion Matrix mostra come le istanze vengano distribuite tra classe reale e classe predetta, rendendo visibili sia le classificazioni corrette, concentrate sulla diagonale principale, sia gli errori, che compaiono, invece, al di fuori di essa[40]. Dunque, la Confusion Matrix rappresenta uno strumento essenziale per interpretare il tipo di errore prodotto dal modello e per leggere con maggiore chiarezza il suo comportamento in scenari non stazionari, oltre che un supporto visivo alle metriche principali. Nel caso generale di un problema multi-classe con K classi, la Confusion Matrix è definita come una matrice quadrata $\mathbf{C} \in \mathbb{R}^{K \times K}$, in cui ogni elemento $C_{i,j}$ denota il numero di istanze appartenenti alla classe reale i che vengono classificate come classe j :

$$C_{i,j} = \sum_{n=1}^N \mathbb{I}(y_n = i \wedge \hat{y}_n = j) \quad (3.10)$$

dove y_n indica la classe reale dell’istanza n -esima, mentre $\hat{y}_n$ indica la predizione del modello. La diagonale principale della matrice ($C_{i,i}$) rappresenta le classificazioni corrette, mentre gli elementi fuori diagonale quantificano le confusioni tra classi diverse.

Nel caso binario, la Confusion Matrix si riduce alla forma classica:

$$\mathbf{C} = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \quad (3.11)$$

dove:

$$TP : \text{True Positives} = \sum_{i=1}^N \mathbb{I}(y_i = 1 \wedge \hat{y}_i = 1) \quad (3.12)$$

$$FP : \text{False Positives} = \sum_{i=1}^N \mathbb{I}(y_i = 0 \wedge \hat{y}_i = 1) \quad (3.13)$$

$$FN : \text{False Negatives} = \sum_{i=1}^N \mathbb{I}(y_i = 1 \wedge \hat{y}_i = 0) \quad (3.14)$$

$$TN : \text{True Negatives} = \sum_{i=1}^N \mathbb{I}(y_i = 0 \wedge \hat{y}_i = 0) \quad (3.15)$$

Un elevato numero di FP può causare affaticamento negli analisti (alert fatigue), mentre la presenza di FN indica attacchi non rilevati che rimangono liberi di operare nel sistema, rappresentando la criticità maggiore per la sicurezza.

Nel Continual Learning, e più nello specifico per lo scenario Class-Incremental Learning, la distribuzione corretta delle classi assume un ruolo molto importante, poiché il calo di prestazioni non coincide sempre e soltanto con il Catastrophic Forgetting in senso stretto. In particolar modo, il modello deve essere in grado di distinguere tutte le classi osservate fino a quel momento senza poter contare sull'identità del task durante la fase di addestramento. Quindi, una parte degli errori può anche dipendere dal task confusion, cioè dalla difficoltà del modello nel separare in modo corretto classi appartenenti a task diversi[23]. Di conseguenza, questo aspetto assume ancor più rilevanza se si parla di IDS, dato che gli studi sulla visualizzazione degli errori permettono di capire quali attacchi vengono mancati e quanto questo tipo di errori possano compromettere l'operatività di un sistema[2].

3.5.2 Valutazione intra-task

La valutazione intra-task descrive il protocollo con cui DARKER viene valutato al termine di ciascun task e costituisce il meccanismo operativo attraverso cui le metriche introdotte nella sezione precedente vengono effettivamente calcolate. La logica di fondo è quella di valutare il modello sul test set del task corrente e su tutti i test set accumulati fino a quel momento dopo ogni task t , tipica del contesto del Continual Learning.

La valutazione dell'Accuracy del modello sul singolo task si articola in due fasi distinte:

1. Nella prima fase, il modello elabora i campioni del test set X_{eval} tramite il metodo `model.predict()`, ottenendo una distribuzione di probabilità sulle classi per ciascun in-

put. Successivamente, l'applicazione del metodo *argmax()* seleziona la classe con la probabilità più elevata, determinando il vettore delle classi predette *y_pred*.

2. Nella seconda fase, il modello effettua un confronto elemento per elemento tra il vettore delle classi predette dal modello *y_pred* e il vettore delle classi reali presenti nel test set *y_eval*, generando un array di valori booleani che indica per ogni elemento se la predizione sia corretta o meno. A questo array viene poi applicato il calcolo della media aritmetica, tramite la funzione *mean()*, generando un singolo valore che rappresenta la performance del modello sul test set corrente.

Per il calcolo delle altre metriche fondamentali per la valutazione delle performance del modello su ciascun task viene impiegata la funzione *compute_metrics()*. Essa riceve in input il vettore delle etichette reali *y_true*, le predizioni del modello *y_pred* e l'insieme delle classi presenti nel test set corrente *labels_present*, restituendo la Precision, la Recall e l'F1-score. L'utilizzo delle sole classi realmente presenti nel test set permette di evitare l'introduzione di penalità artificiali per le classi assenti che distorcerebbe la stima delle performance su ogni task. L'aggregazione è stata ottenuta tramite l'utilizzo della strategia "macro averaging"[41]: per ciascuna metrica, il valore viene calcolato per ogni classe presente e poi mediato in modo non pesato, assegnando ad ogni classe lo stesso contributo indipendentemente dalla sua frequenza nel test set. La scelta di questa strategia è motivata dalla natura dei dati poiché, in un contesto IDS con classi fortemente sbilanciate, la media pesata tenderebbe a mascherare la degradazione del modello sulle classi di attacco minoritarie, le quali contribuirebbero poco al valore aggregato proprio perché poco rappresentate. Al contrario, la media macro tratta ogni categoria di attacco allo stesso modo, rendendo visibile anche il calo di performance sulle classi rare. Il parametro "zero_division" posto a zero gestisce il caso in cui il modello non produca alcuna predizione positiva per una classe, assegnando valore zero alla metrica corrispondente. Questo è uno scenario che si verifica frequentemente nelle prime fasi dell'addestramento, quando il modello ha appena espanso il proprio layer di output per accogliere nuove classi ma non ha ancora sviluppato rappresentazioni in grado di discriminare le diverse classi con correttezza.

La funzione *compute_forgetting()*, invece, implementa la definizione di Forgetting medio introdotta nella sezione precedente. Per ogni task precedente, la funzione identifica il valore massimo raggiunto dall'Accuracy su quel task in tutte le valutazioni precedenti all'ultimo:

$$F_i = \max_{t \in [i, T-1]} \{acc_matrix[t][i]\} - acc_matrix[T-1][i] \quad (3.16)$$

dove l'indice t indica al termine di quale task effettuare la valutazione, l'indice i indica il task i -esimo su cui effettuare la valutazione e T rappresenta il numero totale di task che il modello ha incontrato fino a quel momento. Il massimo è calcolato a partire dall'indice $t = i$, ovvero dal momento in cui il task è stato addestrato per la prima volta, fino all'indice $T - 2$ incluso. Il confronto viene poi effettuato con il valore dell'Accuracy registrato all'indice $T - 1$, che corrisponde alla valutazione più recente disponibile sul task i -esimo. Questa implementazione riflette la definizione classica di Forgetting[32], secondo la quale occorre misurare la differenza rispetto alla migliore performance raggiunta, rendendo la stima più fedele alla reale perdita di conoscenza. Inoltre, il Forgetting è stato tracciato anche sulle altre metriche e, in particolar modo, sulla Recall, poiché quest'ultima costituisce la principale metrica analizzata per l'adattamento dinamico del coefficiente α del termine di distillazione nella funzione di perdita di DARKER.

Capitolo 4

Risultati

Nel seguente capitolo vengono esaminati e discussi i risultati ottenuti dalla valutazione sperimentale di DARKER sui dataset ToN-IoT e CICIDS2017, posti a confronto con le metodologie DER e iCaRL. L'analisi condotta secondo il protocollo di valutazione intra-task è finalizzata a verificare se DARKER permette di ottenere un compromesso più solido tra stabilità e plasticità rispetto alle principali strategie presenti in letteratura nell'ambito IDS. I risultati sono presentati separatamente per ciascun dataset, le cui caratteristiche distributive permettono di osservare il comportamento del modello in scenari di Concept Drift differenti.

4.1 ToN-IoT

Accuracy

All'interno della Figura 4.1, nella quale viene confrontata l'Accuracy media delle diverse metodologie lungo l'intera sequenza di task, si osserva come il modello DARKER, a differenza di DER e iCaRL, mantenga un andamento regolare, caratterizzato da oscillazioni circoscritte e dall'assenza di collassi. Dopo un transitorio iniziale nei primi task, in cui tutti i metodi esibiscono prestazioni comparabili, la curva relativa a DARKER si stabilizza in un intervallo che non scende mai sotto la soglia di 0.77, evidenziando come la combinazione di buffer class-aware e distillazione adattiva limiti l'interferenza tra task successivi, persino in presenza di un flusso di dati fortemente sbilanciato.

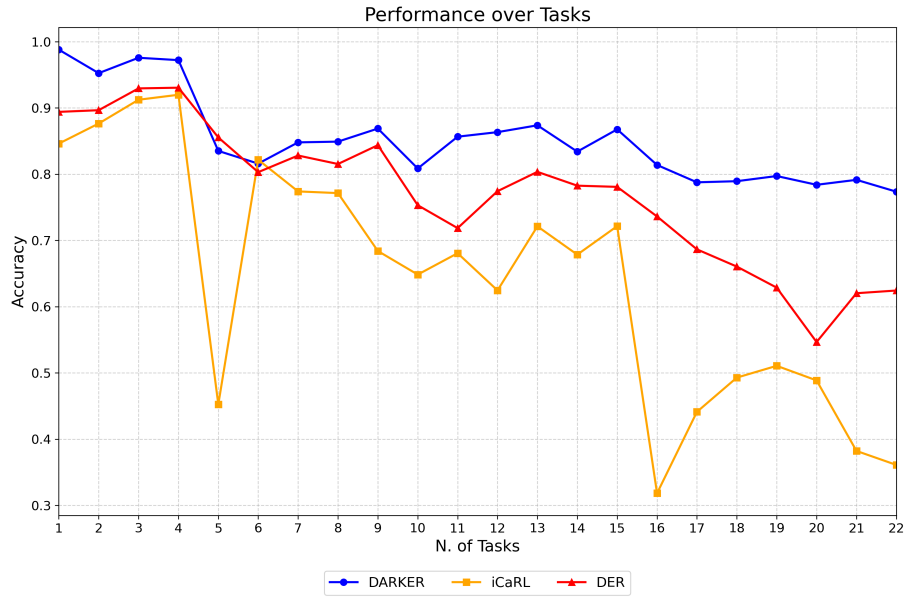


Figura 4.1: Accuracy media per task su ToN-IoT

La Figura 4.2, la quale isola l'effetto dell'incremento del numero di classi osservate a parità di task, conferma le precedenti osservazioni in modo più netto: DARKER mostra un decadimento molto limitato all'aumentare del numero di classi. Il vantaggio di DARKER si manifesta principalmente nella robustezza della curva in corrispondenza di un elevato numero di classi note.

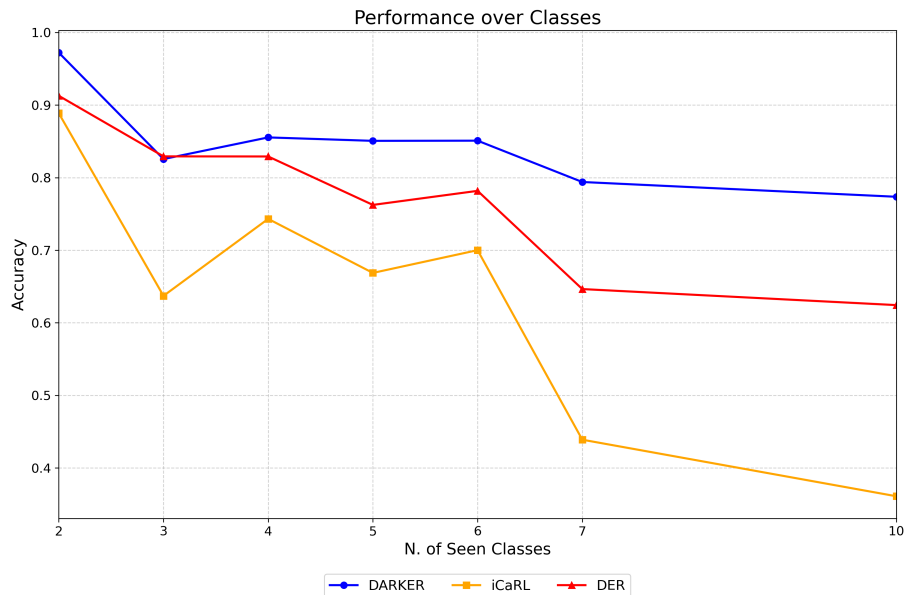


Figura 4.2: Accuracy media per classi su ToN-IoT

L'analisi congiunta dei due grafici offre una chiara evidenza di come l'assenza di una delle due componenti dell'architettura di DARKER, ovvero l'adozione di un buffer non class-aware per

DER e l'utilizzo di una componente di distillazione statica per iCaRL, si traduca in instabilità del modello nella corretta classificazione delle minacce all'espandersi dello spazio delle classi.

Forgetting

L'analisi del Forgetting medio delle diverse metodologie lungo l'intera sequenza di task, illustrato in Figura 4.3, risulta dominata dall'instabilità di iCaRL, in perfetta coerenza con quanto osservato nell'analisi dell'Accuracy. Gli altri due approcci restano entrambi vicini allo zero, con DER che dal task 10 in avanti mostra oscillazioni più ampie, rispetto a DARKER, la cui curva mostra una crescita più graduale e controllata. Nei task conclusivi 21 e 22, contraddistinti dalla presenza di classi minoritarie che, a causa della loro scarsa numerosità, dispongono di un supporto estremamente ridotto nei set di valutazione, DARKER registra un valore di Forgetting medio leggermente superiore rispetto a DER.

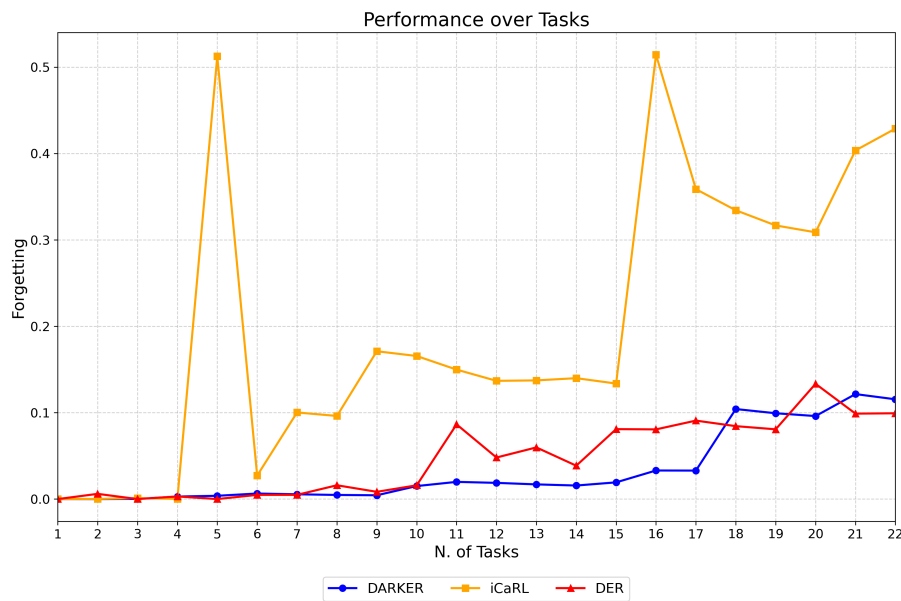


Figura 4.3: Forgetting medio per task su ToN-IoT

Il grafico in Figura 4.4 amplifica visivamente la medesima dinamica: l'introduzione di nuove classi minoritarie compromette in modo cumulativo la conoscenza già acquisita da iCaRL. Al contrario, DARKER e DER preservano lo stesso comportamento descritto in precedenza, con DARKER che resta tendenzialmente inferiore a DER, fino al raggiungimento dei task finali caratterizzati da un ridotto volume di campioni.

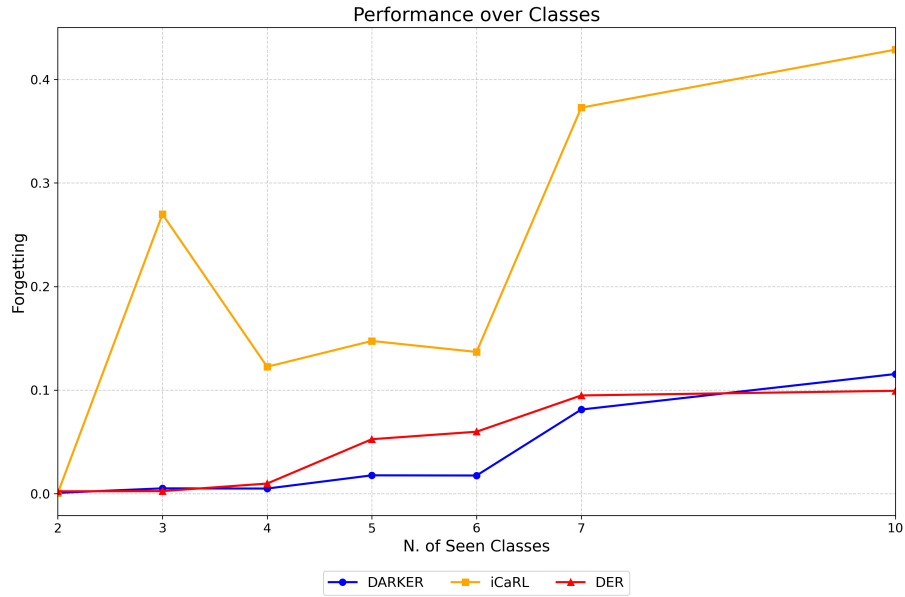


Figura 4.4: Forgetting medio per classi su ToN-IoT

In Figura 4.5 viene illustrato lo spazio di trade-off tra Accuracy e Forgetting medi delle diverse metodologie considerate. I risultati evidenziano come i punti associati a DARKER presentino una densità di concentrazione superiore all'interno o in prossimità della cosiddetta *Best Region*. La distribuzione dei risultati ottenuti consente di dividere il grafico in tre regioni sequenziali lungo l'asse del Forgetting: nella fascia caratterizzata da Forgetting quasi nullo, i tre metodi condividono elevati livelli di Accuracy, con una concentrazione maggiore di DARKER e DER rispetto ad iCaRL; nella fascia intermedia, dove il Forgetting è contenuto, DARKER preserva un valore di Accuracy nettamente superiore rispetto agli altri approcci, fornendo un'evidenza quantitativa dell'efficacia della distillazione adattiva; infine, la fascia ad alto Forgetting risulta occupata esclusivamente dalle rilevazioni di iCaRL.

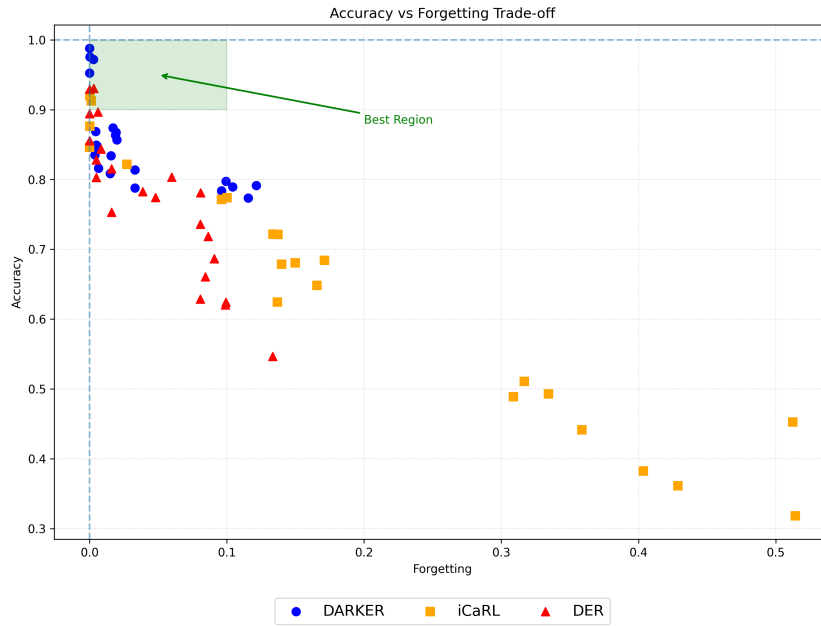


Figura 4.5: Trade-off tra Accuracy e Forgetting su ToN-IoT

Recall

La Figura 4.6 descrive il confronto relativo alla Recall delle tre metodologie lungo l'intera sequenza di task del dataset ToN-IoT. In seguito a una fase iniziale caratterizzata da prestazioni lievemente inferiori, il comportamento di DARKER si attesta su livelli analoghi a quelli di DER, manifestando tuttavia oscillazioni più contenute e una maggiore regolarità all'avanzare dei task. Rispetto ad iCaRL, l'analisi evidenzia un degrado prestazionale costante di quest'ultimo, legato alla procedura di herding impiegata per la gestione del buffer di memoria.

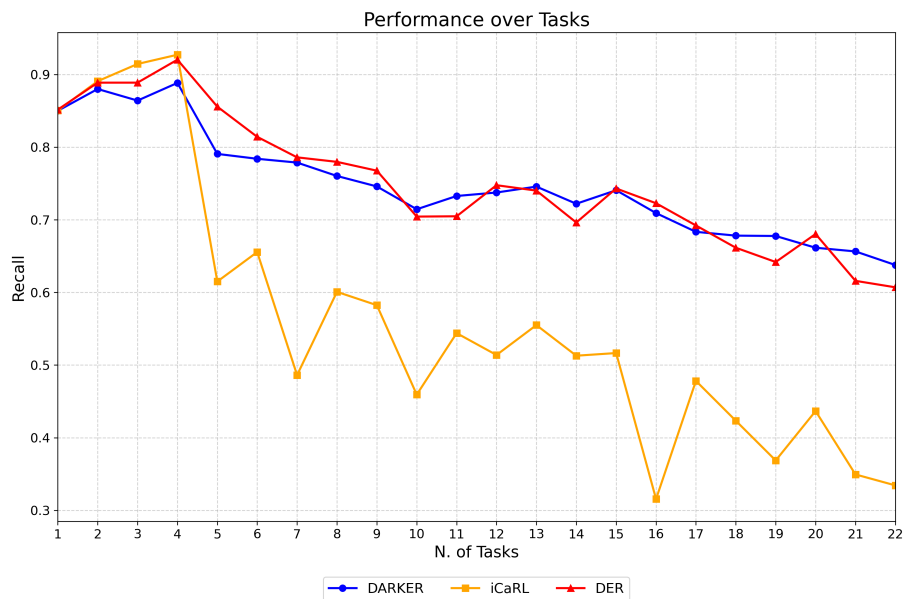


Figura 4.6: Recall media per task su ToN-IoT

Coerentemente con quanto rilevato sulla sequenza di task, l'avvio di DARKER risulta più debole di quello di DER nelle prime fasi del processo. Tuttavia, come evidenziato in Figura 4.7, all'aumentare del numero di classi osservate e, in particolare, dalle 5 classi in poi, DARKER esibisce un comportamento più stabile e lineare rispetto alle metodologie concorrenti, mitigando l'effetto di degradazione nel lungo periodo.



Figura 4.7: Recall media per classi su ToN-IoT

Precision

Per quanto concerne la Precision, il vantaggio empirico a favore di DARKER appare più robusto e persistente. Nel grafico in Figura 4.8, nelle fasi in cui il buffer risulta scarsamente popolato e il meccanismo di distillazione ancora debole, DARKER mostra un margine di superiorità contenuto. All'incrementarsi del numero dei task, tuttavia, il modello delinea una curva con andamento crescente, nonostante oscillazioni locali. Al contrario, le altre due metodologie mostrano un comportamento del tutto differente: le rispettive curve manifestano una marcata instabilità lungo l'intera sequenza dei task e rimangono stabilmente al di sotto di quella espressa da DARKER, con l'unica eccezione rappresentata da iCaRL in corrispondenza del task 15.

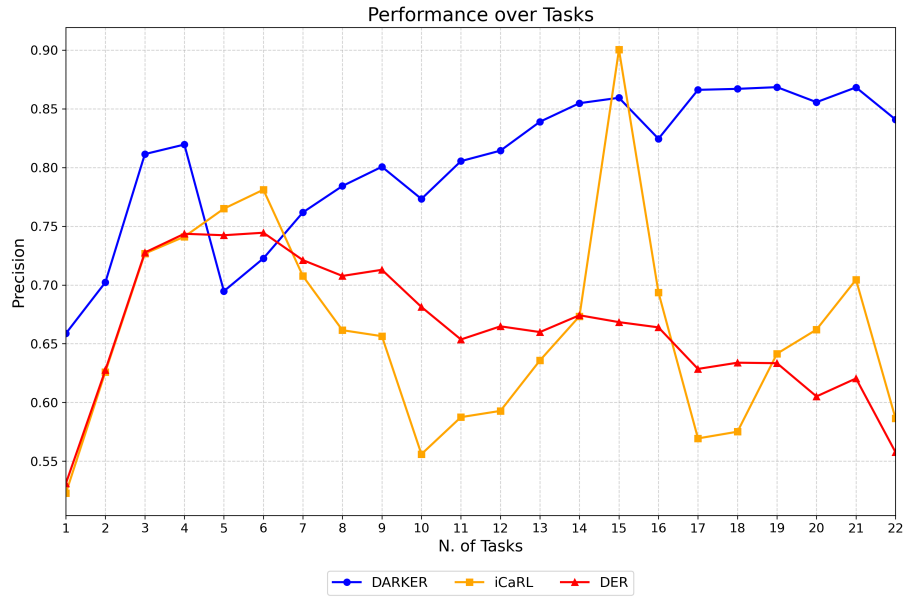


Figura 4.8: Precision media per task su ToN-IoT

Il comportamento analizzato lungo la sequenza dei task si riflette anche nel grafico di Figura 4.9 in funzione del numero di classi viste. Fatta eccezione del caso in cui le classi osservate sono 3, in cui DARKER subisce un lieve calo, l'approccio proposto si colloca sistematicamente al di sopra degli altri, ampliando il proprio distacco in modo progressivo: al raggiungimento delle 10 classi osservate, DARKER supera gli altri metodi di oltre 25 punti percentuali, dimostrando come il modello riesca a preservare un'elevata affidabilità predittiva all'aumentare della complessità dello spazio delle classi, a differenza di quanto mostrato da DER e iCaRL.



Figura 4.9: Precision media per classi su ToN-IoT

F1-score

Il quadro complessivo derivante dall'analisi della metrica F1-score sul dataset ToN-IoT, riportato in Figura 4.10, evidenzia una dominanza totale di DARKER rispetto alle soluzioni concorrenti lungo l'intera sequenza temporale.

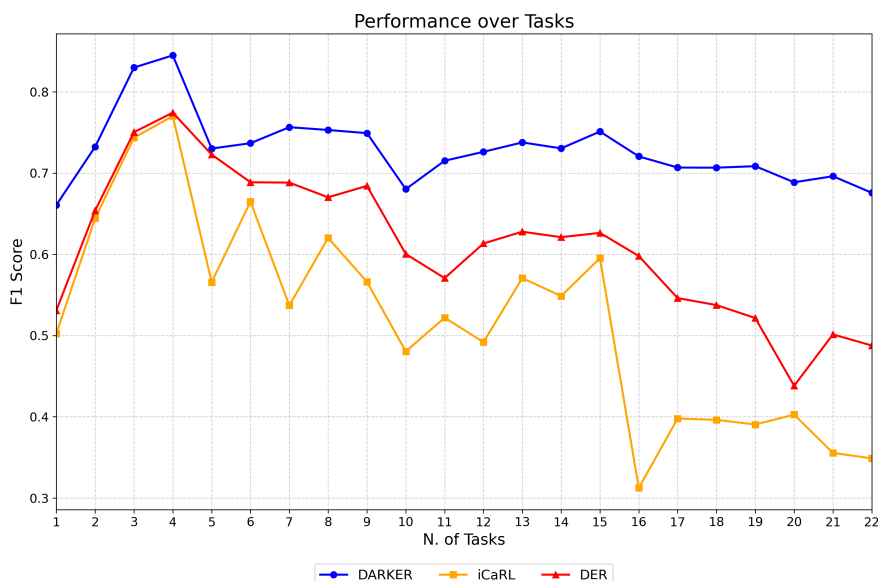


Figura 4.10: F1-score medio per task su ToN-IoT

Lo stesso profilo di superiorità incondizionata di DARKER si conferma e si estende anche al crescere del numero di classi monitorate, come illustrato in Figura 4.11. Questo riscontro convalida l'efficacia globale di DARKER, in grado di ottimizzare sia la capacità di intercettazione delle minacce sia l'accuratezza delle predizioni fornite.



Figura 4.11: F1-score medio per classi su ToN-IoT

Confusion Matrix

La Confusion Matrix di DARKER sul ToN-IoT, analizzabile in Figura 4.12, mostra una diagonale principale ben definita in corrispondenza delle classi dotate di maggiore supporto campionario, che rappresentano i vettori d'attacco predominanti. Le principali anomalie rilevabili riguardano la parziale sovrapposizione tra le classi "Injection" e "XSS", dovuta alla similarità semantica delle due classi nei sistemi IoT, e l'insufficiente capacità di convergenza sulle classi "Backdoor" e "MITM", penalizzate dal loro ridotto supporto statistico all'interno del dataset.

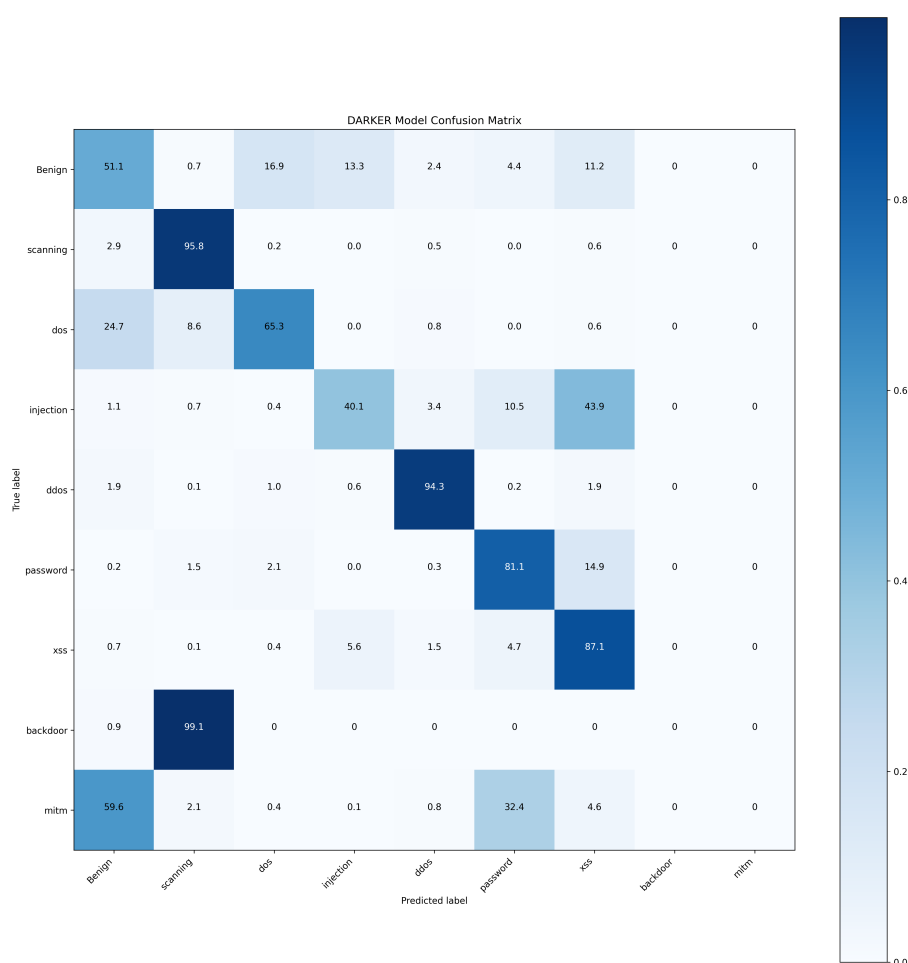


Figura 4.12: DARKER Confusion Matrix su ToN-IoT

All'interno della Confusion Matrix di DER, riportata in Figura 4.13, emerge un marcato bias di conservazione verso la classe "Benign" e in un contesto operativo IDS tale distorsione si traduce in inaffidabilità complessiva dell'infrastruttura, determinando un numero critico di falsi negativi. Oltre a ciò, altre classi, come, ad esempio, quelle di "DoS" e "XSS", evidenziano la presenza di confini decisionali degradati, che inducono il classificatore a far convergere erroneamente gli attacchi verso altre classi.

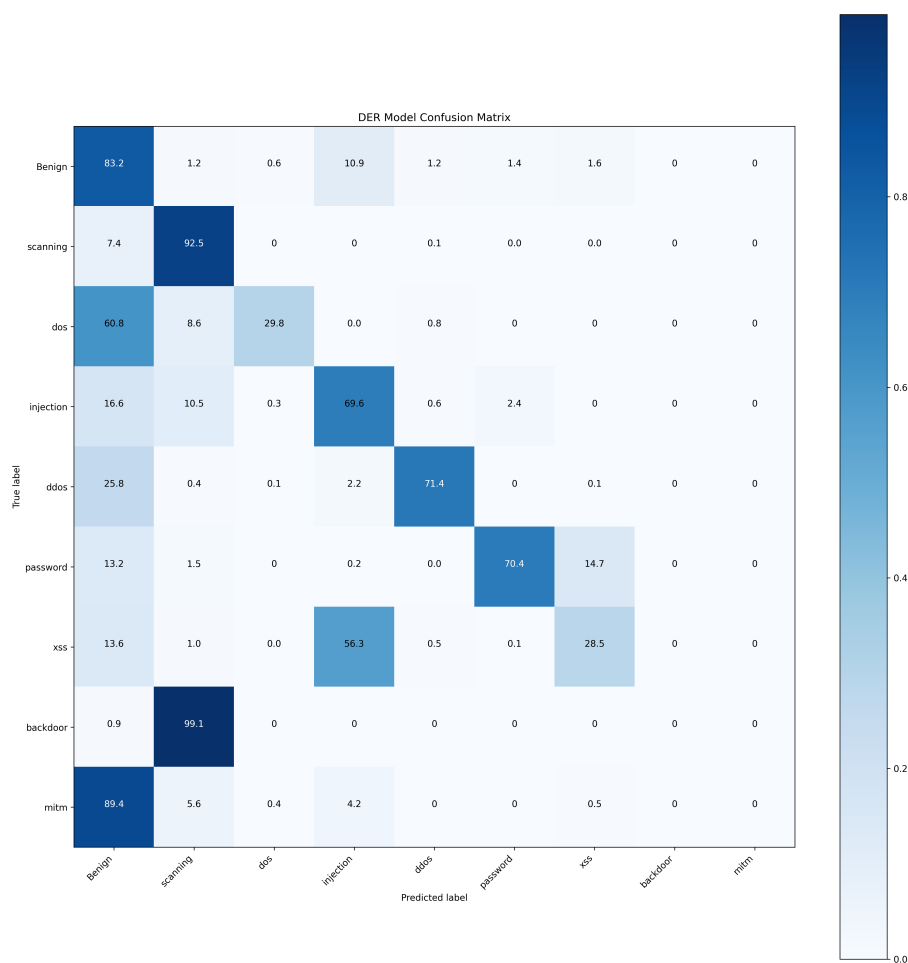


Figura 4.13: DER Confusion Matrix su ToN-IoT

La Confusion Matrix associata a iCaRL, rappresentata in Figura 4.14, dimostra l'elevata instabilità del modello, palesando una forte dispersione della conoscenza tra classi vicine, evidente soprattutto nelle classi "Scanning" e "XSS". Sebbene iCaRL si dimostri l'unico modello capace di discriminare la classe "Backdoor", nonostante il ridotto volume campionario, il collasso della classe "Benign" evidenzia una vulnerabilità sistemica: in uno scenario operativo reale, un crollo simile comporterebbe un tasso insostenibile di falsi allarmi, invalidando l'efficacia del sistema nella corretta distinzione del traffico lecito da quello malevolo.

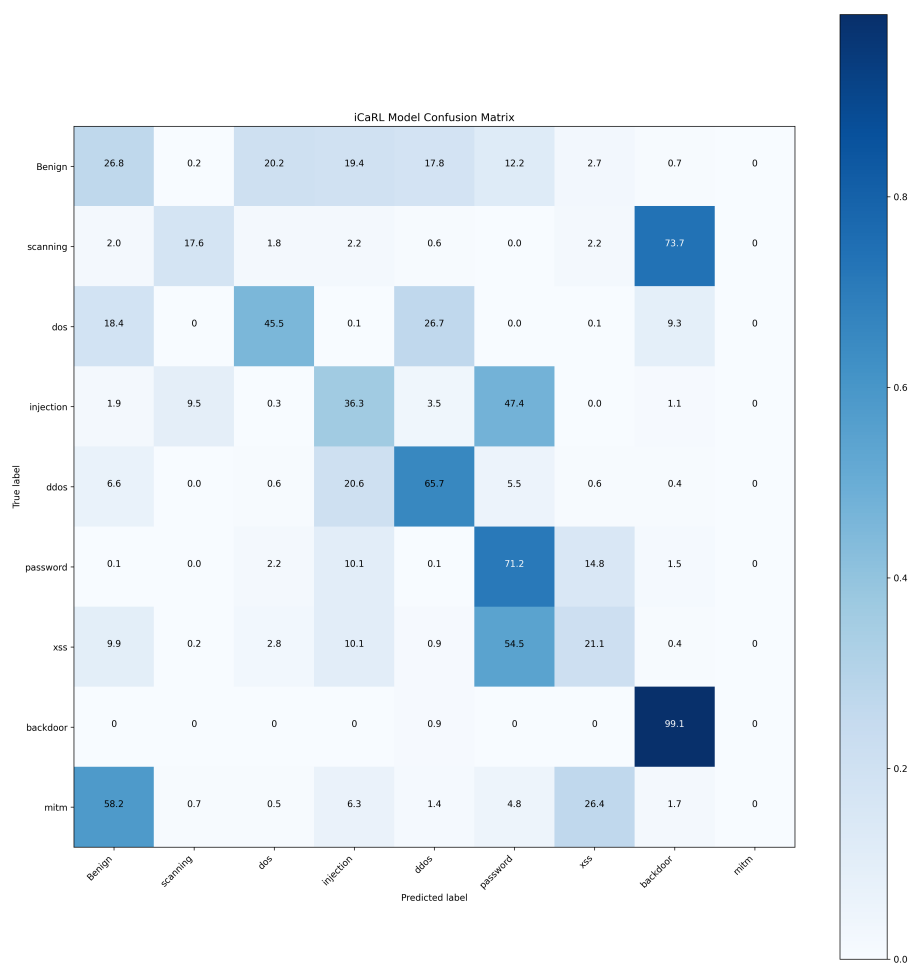


Figura 4.14: iCaRL Confusion Matrix su ToN-IoT

L'analisi comparativa evidenzia come la matrice di DARKER superi complessivamente le altre soluzioni in corrispondenza delle classi a maggior supporto numerico. Nel dominio della sicurezza informatica, l'errore asimmetrico più severo consiste nella classificazione di un vettore d'attacco come traffico legittimo e DARKER attenua efficacemente tale rischio, riducendo la conversione verso la classe "Benign" a livelli trascurabili per tutte le minacce principali. Inoltre, a differenza delle altre metodologie confrontate, DARKER dimostra una migliore capacità nello strutturare confini decisionali netti e discriminanti tra le varie classi di attacco.

4.2 CICIDS2017

Accuracy

Il quadro generale delineato dal dataset CICIDS2017 varia nettamente da quanto riscontrato con il dataset ToN-IoT, poiché in questo contesto il traffico lecito costituisce oltre l'80% dell'intero

set di dati, sovrastando le classi malevole minoritarie. Tale premessa evidenzia come la metrica di valutazione relativa all'Accuracy del modello, in virtù delle proprietà metodologiche discusse nel capitolo precedente, non può costituire l'unico metro di paragone tra le diverse metodologie considerate. Nella Figura 4.15, si osserva come DARKER e DER esibiscano un andamento quasi sovrapponibile lungo l'intera sequenza di task, sebbene il modello proposto mantenga in modo costante un lieve margine di superiorità rispetto a DER.

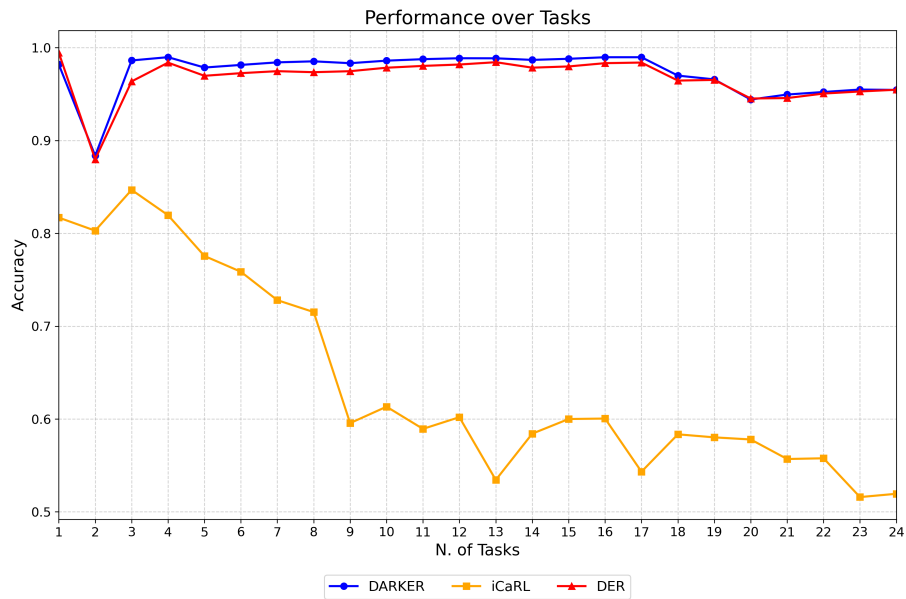


Figura 4.15: Accuracy media per task su CICIDS2017

Anche nell'analisi condotta in funzione del numero di classi viste, osservabile in Figura 4.16, il vantaggio incrementale di DARKER rispetto a DER appare più contenuto rispetto a quanto registrato su ToN-IoT, nonostante l'estensione dello spazio delle classi sia considerevole.



Figura 4.16: Accuracy media per classi su CICIDS2017

Questo risultato dimostra come il beneficio della distillazione adattiva di DARKER rispetto a quella statica di DER emerga in modo più nitido in scenari con una distribuzione delle classi più sbilanciata, attenuandosi parzialmente in configurazioni con uno sbilanciamento più contenuto. Oltre questo, il confronto con iCaRL conferma come l'adozione di un buffer class-aware, volto a mitigare il "recency bias" sul lungo termine, combinato con la distillazione adattiva dei logits, introduca miglioramenti sostanziali nell'accuratezza complessiva della classificazione.

Forgetting

In Figura 4.17, l'analisi del Forgetting medio tra i tre approcci rivela una netta divergenza a favore di DARKER. Sia DARKER sia DER si attestano su intervalli prossimi allo zero, con il primo che preserva una stabilità quasi assoluta fino al task 18; al contrario, iCaRL manifesta un Forgetting elevato e persistente per tutta la sequenza temporale. In questo contesto, a differenza dei trend osservati su ToN-IoT, DARKER conserva un vantaggio costante, interrotto soltanto da due brevi sovrapposizioni nella fase finale, confermando la capacità del meccanismo adattivo di minimizzare la perdita di conoscenza pregressa.

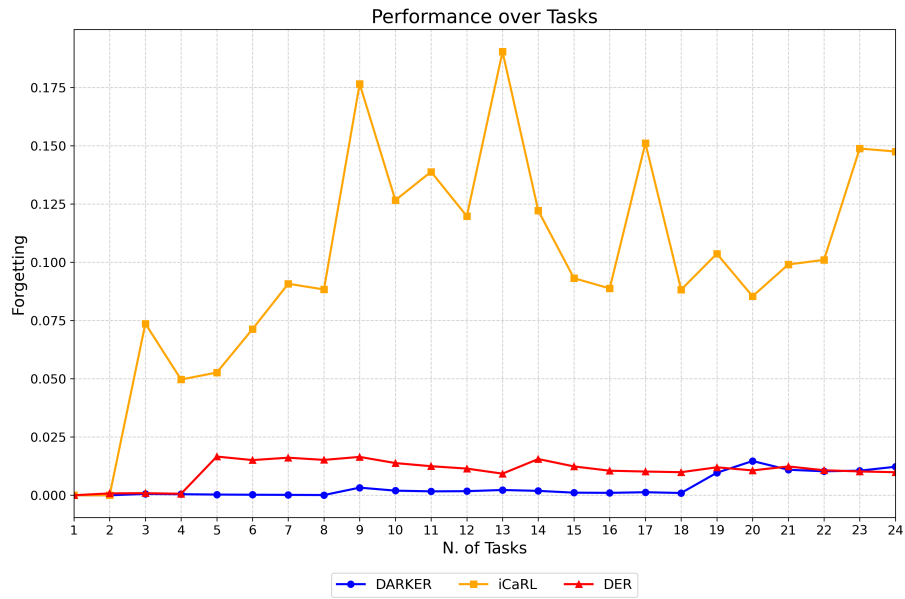


Figura 4.17: Forgetting medio per task su CICIDS2017

Il grafico del Forgetting medio in funzione del numero di classi osservate, illustrato in Figura 4.18, completa le precedenti considerazioni: mentre iCaRL mostra una rapida crescita all'espandersi dello spazio delle classi, la curva di DARKER si mantiene sostanzialmente piatta e prossima allo zero, per poi crescere più lentamente di DER fino alla convergenza quasi speculare nella fase terminale.

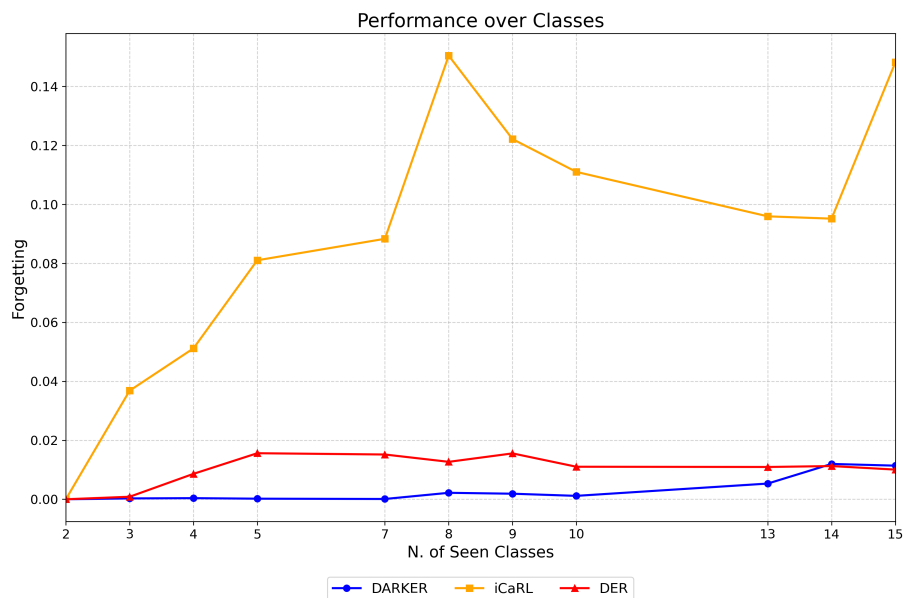


Figura 4.18: Forgetting medio per classi su CICIDS2017

In Figura 4.19 viene illustrato lo spazio di trade-off tra Accuracy e Forgetting medi sul dataset CICIDS2017. La distribuzione dei punti delinea una netta separazione tra due regioni: da

un lato, un'area più compatta in prossimità della *Best Region*, caratterizzata da Forgetting minimo e Accuracy elevata, racchiude la quasi totalità dei punti rappresentativi di DARKER e DER; dall'altro, un'area in cui il Forgetting è più elevato e l'Accuracy è minore contiene una distribuzione più dispersiva costituita esclusivamente dai punti che rappresentano iCaRL. Nonostante all'interno della *Best Region* non risulti una netta distinzione tra DARKER e DER, si nota una maggiore concentrazione dei punti di DARKER nell'estremità superiore sinistra, a testimonianza di un comportamento complessivo migliore rispetto a quello di DER.

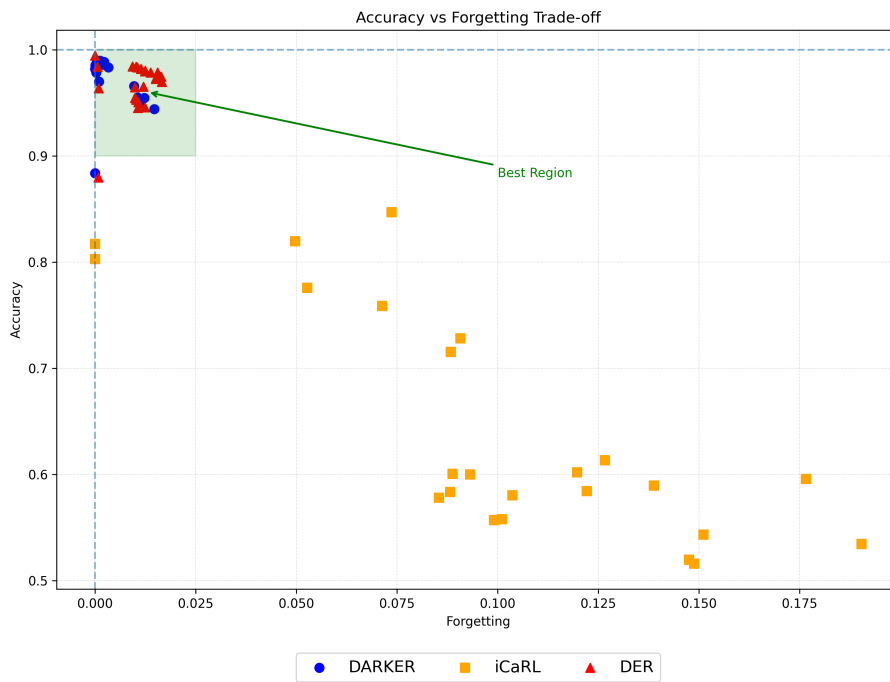


Figura 4.19: Trade-off tra Accuracy e Forgetting su CICIDS2017

Recall

Dall'analisi della Recall media riportata in Figura 4.20, si osserva come tutti gli approcci presentino un forte calo al task 2. Tale anomalia è riconducibile alla configurazione specifica del task, il quale introduce una classe minoritaria non rappresentata nei set di dati impiegati durante le fasi di addestramento dei modelli. Successivamente, la curva associata a DARKER si colloca stabilmente al di sopra delle altre, dimostrando come la combinazione di buffer class-aware e di distillazione adattiva dei logits consenta al modello di preservare un'elevata sensibilità di rilevamento lungo l'intera sequenza temporale.

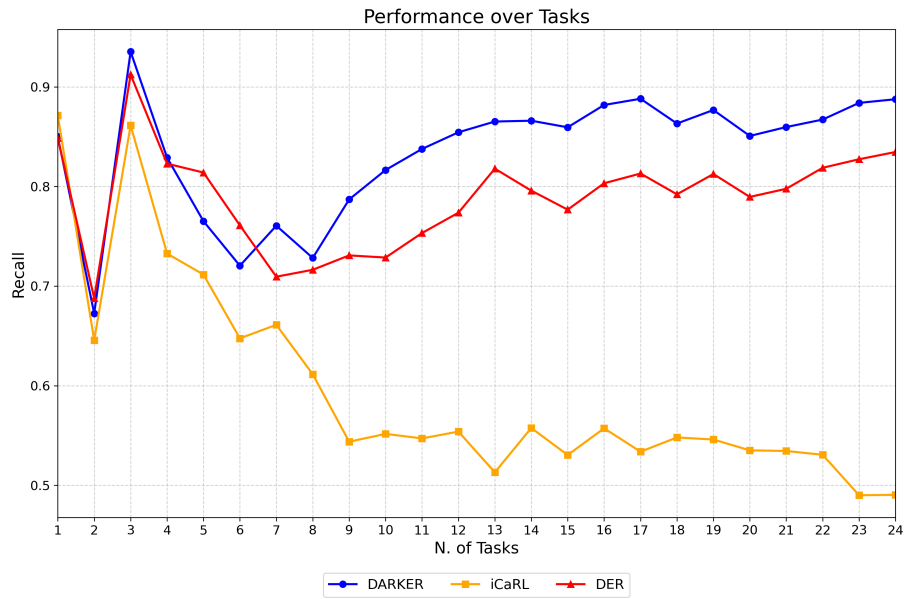


Figura 4.20: Recall media per task su CICIDS2017

La medesima dinamica si riflette nell'analisi prestazionale espressa in funzione del numero di classi (Figura 4.21): nonostante l'incremento progressivo delle classi osservate, DANKER garantisce una stabilità della Recall superiore nel lungo termine rispetto alle metodologie concorrenti.

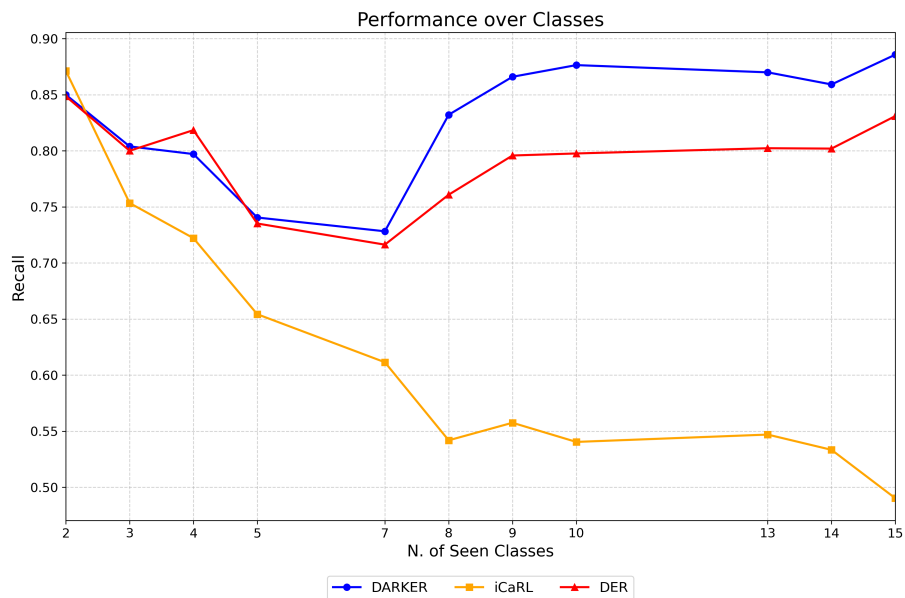


Figura 4.21: Recall media per classi su CICIDS2017

Precision

L'analisi riguardante la Precision sul CICIDS2017 convalida ulteriormente l'efficacia del modello proposto. Come illustrato in Figura 4.22, DANKER e DER esibiscono comportamenti

sovrapponibili nelle fasi iniziali dell'apprendimento, ma, a partire dal task 6, DARKER consolida un margine di superiorità costante che si estende fino al termine della sequenza di task.

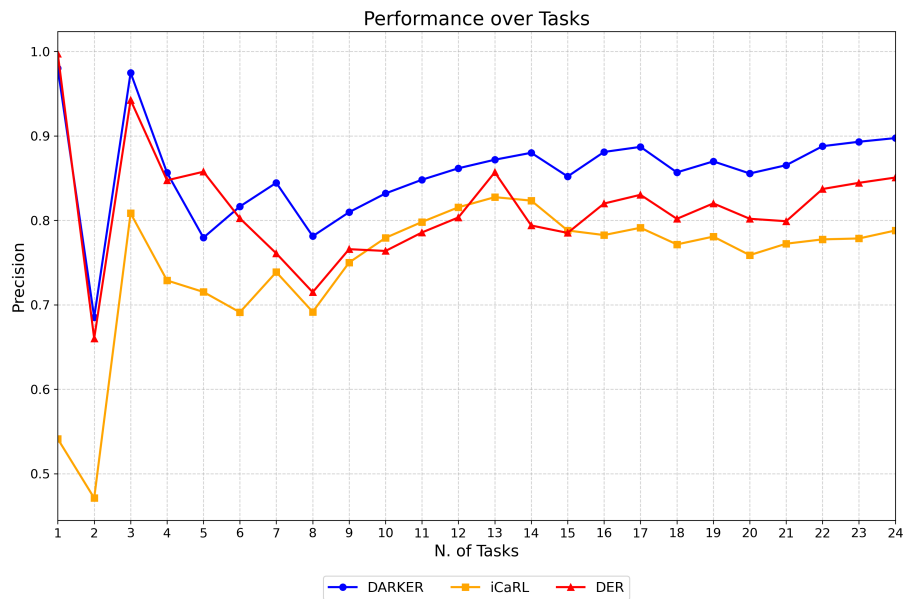


Figura 4.22: Precision media per task su CICIDS2017

Il grafico presente in Figura 4.23 conferma tale andamento in funzione dell'incremento delle classi osservate. Ad eccezione di una fase iniziale in cui l'andamento di DARKER è sovrapponibile a quello di DER, il modello proposto mantiene un distacco costante e persistente rispetto agli altri metodi, confermando la sua capacità di garantire un'elevata affidabilità predittiva all'aumentare delle classi nel dataset CICIDS2017.



Figura 4.23: Precision media per classi su CICIDS2017

F1-score

Il quadro generale nella valutazione della metrica F1-score evidenzia una netta separazione tra i metodi DARKER e DER da un lato e iCaRL dall'altro. Come evidenziato in Figura 4.24, in seguito a una fase iniziale caratterizzata da un comportamento analogo, a partire dal task 6 DARKER delinea un margine incrementale significativo e duraturo, evolvendo da uno scarto iniziale contenuto fino a un vantaggio cospicuo fino alla conclusione della sequenza temporale.

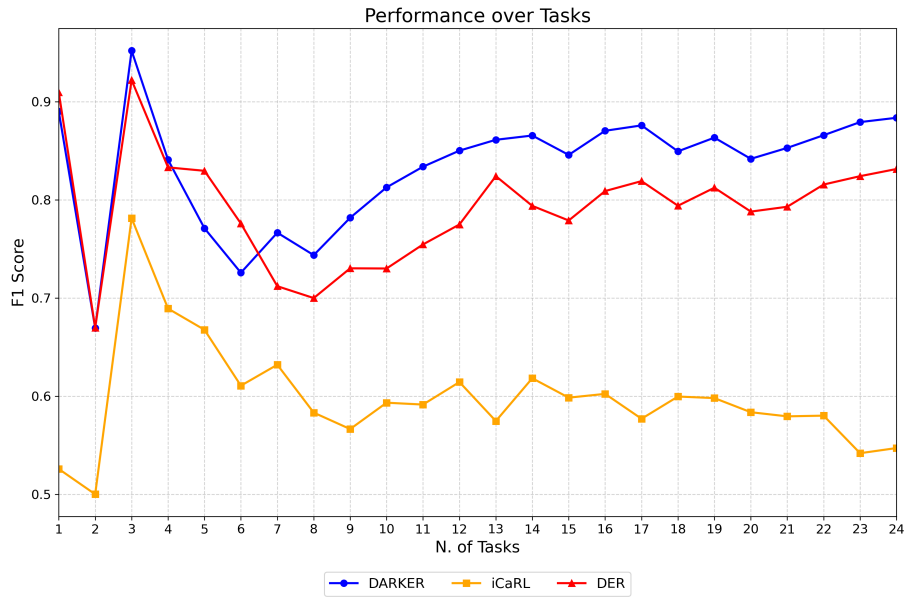


Figura 4.24: F1-score medio per task su CICIDS2017

La stessa tendenza evolutiva si osserva nell'analisi della metrica F1-score rispetto al numero di classi osservate, visibile in Figura 4.25. Dopo una fase iniziale caratterizzata da sovrapposizioni tra DARKER e DER, l'approccio proposto incrementa in modo progressivo il proprio distacco all'espandersi dello spazio delle classi, confermando l'ottimizzazione simultanea della capacità di rilevamento e della precisione delle risposte.



Figura 4.25: F1-score medio per classi su CICIDS2017

Confusion Matrix

La Confusion Matrix di DARKER sul CICIDS2017, riportata in Figura 4.26, mostra un'accurata classificazione delle classi caratterizzate da un volume elevato e da una netta separazione nello spazio delle feature, registrando un tasso di errore marginale in proporzione al volume degli attacchi. All'interno di questo scenario, si identificano due sole criticità circoscritte: la prima è relativa alla classe "Infiltration", la quale presenta una Recall pari a zero, a causa del numero limitato di campioni disponibili sia all'interno dell'intero dataset sia nella rispettiva ripartizione di test; la seconda riguarda la classe "DoS Slowhttptest", anch'essa con Recall nulla, penalizzata dalla sua somiglianza semantica con la classe "DoS slowloris", la quale ha indotto il modello a un'errata classificazione.

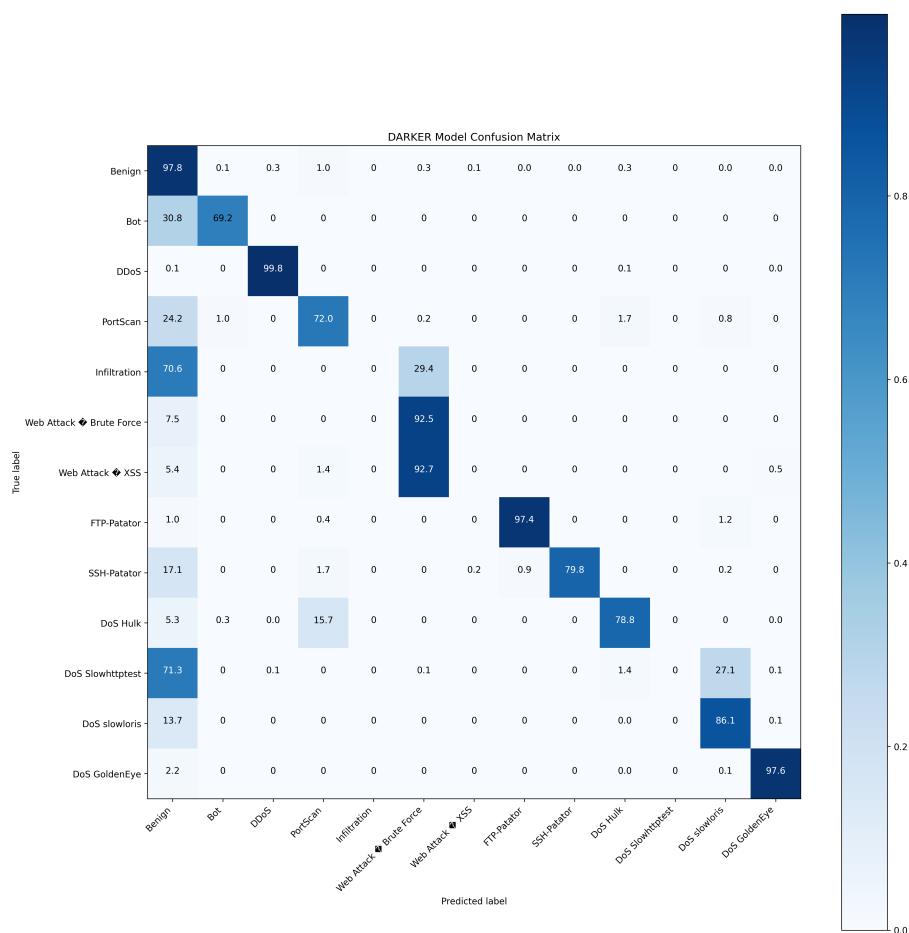


Figura 4.26: DARKER Confusion Matrix su CICIDS2017

La Confusion Matrix di DER, illustrata in Figura 4.27, dimostra come il modello sia in grado di classificare correttamente le classi con un volume particolarmente elevato. Tuttavia, il pattern dominante della matrice è un collasso sistemico: molteplici classi registrano infatti una Recall pari a zero, con una deviazione quasi totale dell'errore verso la classe di traffico legittimo, etichettato come "Benign". Tra le varie classi classificate in maniera errata, quella su cui il modello ha avuto prestazioni peggiori è la classe "DoS Hulk"; pur essendo caratterizzata da un alto volume di traffico, DER crolla in termini di Recall, classificando circa un terzo dei vettori d'attacco come traffico legittimo.

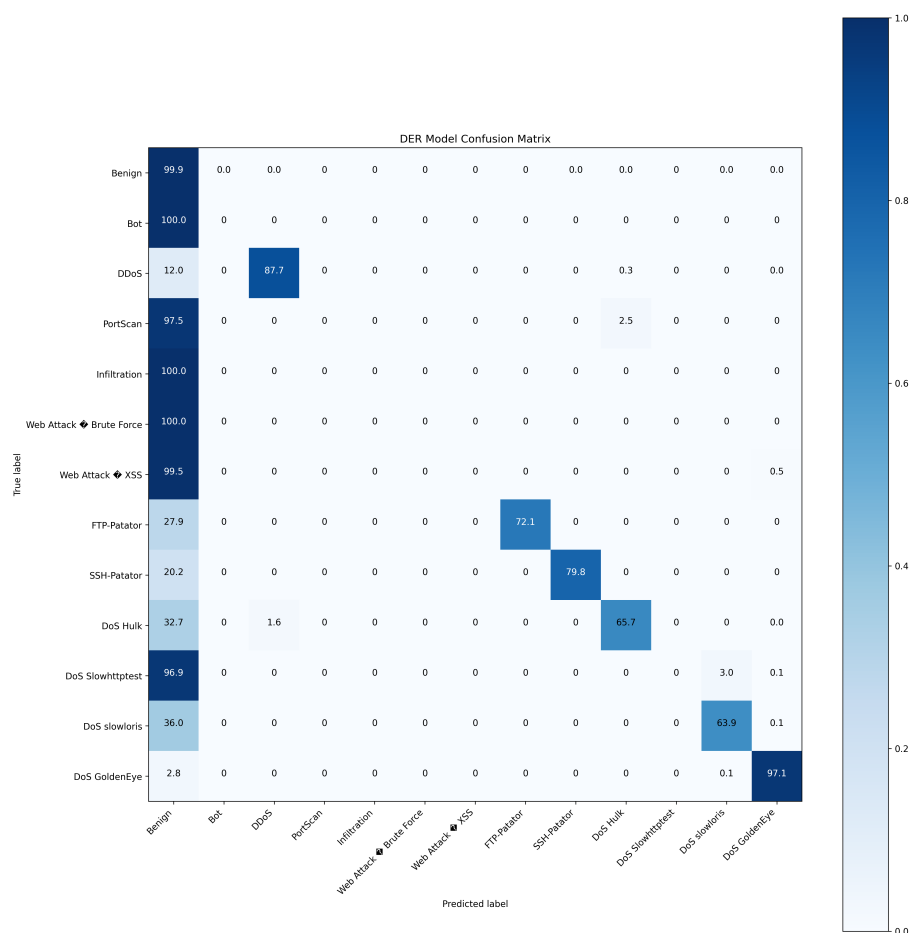


Figura 4.27: DER Confusion Matrix su CICIDS2017

In Figura 4.28 viene presentata la Confusion Matrix di iCaRL, la quale mostra un pattern di errore qualitativamente diverso, caratterizzato da una dispersione diffusa su classi di attacco vicine. Tale instabilità è comune a numerose classi, inclusa "Benign", delineando una criticità che determinerebbe un volume insostenibile di falsi allarmi in ambiente produttivo.

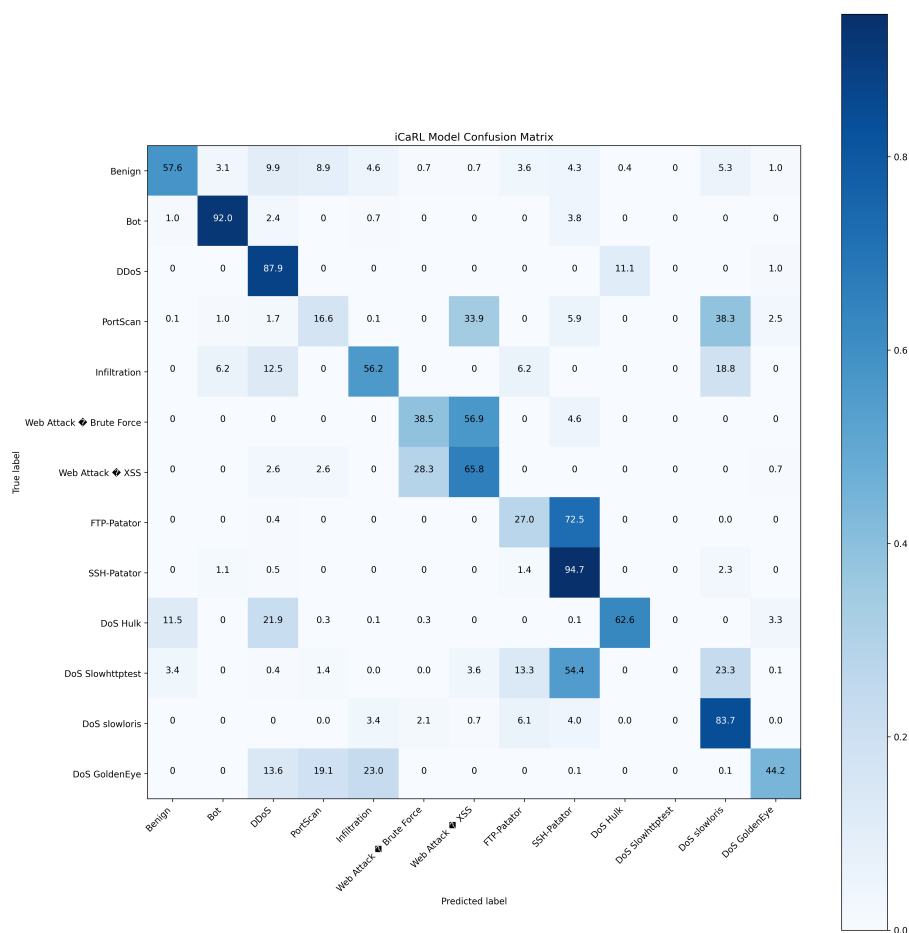


Figura 4.28: iCaRL Confusion Matrix su CICIDS2017

Tra le tre matrici analizzate, DARKER si distingue come l'unico approccio capace di evitare fallimenti sistemici su entrambe le dimensioni cruciali per un sistema IDS: il rischio di mancata identificazione delle minacce e la potenziale inefficacia operativa dovuta alla generazione di falsi allarmi. Come illustrato in Figura 4.26, DARKER riduce il numero di attacchi non rilevati in misura nettamente superiore rispetto a DER, ottenendo prestazioni simili ad iCaRL; tuttavia, a differenza di quest'ultimo, il modello riesce anche a mantenere un'elevata accuratezza nel controllo dei falsi positivi, ottimizzando le due dimensioni senza richiedere un compromesso sulle prestazioni del sistema.

Conclusione

Il presente lavoro di tesi ha esaminato le complesse problematiche connesse all'applicazione del Continual Learning nell'ambito degli Intrusion Detection Systems, un settore che, nonostante la sua continua evoluzione, mostra ancora una parziale frammentarietà metodologica rispetto ad altri domini applicativi. Il punto di partenza dell'analisi condotta nell'elaborato è rappresentato dall'osservazione critica per cui i fenomeni di Catastrophic Forgetting e Concept Drift vengono affrontati in modo disgiunto dalle soluzioni attualmente presenti in letteratura, le quali mirano a privilegiare la conservazione della memoria storica a scapito della plasticità verso il continuo mutamento dei reali contesti operativi. In risposta a questa dicotomia, il contributo scientifico di questa tesi si concretizza nella formalizzazione e validazione di DARKER, un framework innovativo progettato per superare la rigidità dei modelli presenti in letteratura attraverso l'introduzione di due soluzioni metodologiche: un algoritmo adattivo che calcola dinamicamente la componente di distillazione dei logits in funzione del reale degrado prestazionale registrato sulla metrica di Recall e una politica di campionamento stratificato del buffer episodico, ideata per proteggere le classi di attacco minoritarie dal rischio di sovrascrittura da parte delle classi predominanti.

Gli esperimenti condotti sul dataset ToN-IoT confermano l'efficacia del modello proposto, il quale ha dimostrato un valore di Accuracy stabile lungo l'intera sequenza dei task, preservando un Forgetting medio prossimo allo zero per la quasi totalità del processo di addestramento e consolidando un vantaggio competitivo anche sulle altre metriche valutate rispetto ai modelli di confronto DER e iCaRL, salvo due circoscritte eccezioni: una Recall inizialmente inferiore rispetto a quella di DER, che si stabilizza e la supera nei task successivi, e un lieve incremento di Forgetting nelle fasi finali della sequenza, caratterizzate dall'estrema scarsità campionaria delle classi minoritarie. L'analisi effettuata sulle matrici di confusione ha evidenziato come DARKER rappresenti l'unico approccio in grado di mitigare contemporaneamente i rischi di fallimento sistemico su entrambe le dimensioni cruciali del rilevamento: la mancata intercettazione delle

minacce e la proliferazione di falsi positivi. Al contempo, i risultati sperimentali ottenuti sul dataset CICIDS2017 hanno mostrato un quadro in parte differente, legato soprattutto alla composizione del dataset stesso. In tale scenario, DARKER e DER hanno esibito trend di Accuracy simili per gran parte della sequenza, con un margine di superiorità costante ma contenuto a favore del metodo proposto. Lo scarto si è consolidato in modo più netto nelle metriche di Precision e F1-score solo dopo le fasi iniziali, sottolineando come il beneficio predittivo introdotto dalla distillazione adattiva tenda a manifestarsi proporzionalmente allo sbilanciamento del traffico di rete: nel ToN-IoT, caratterizzato da una forte disparità tra le classi, l'adattamento dinamico del coefficiente di distillazione α trova le condizioni ottimali; in presenza di uno sbilanciamento più attenuato, come nel CICIDS2017, il vantaggio rispetto a DER si riduce, senza mai annullarsi. Tuttavia, l'analisi delle relative matrici di confusione ribadisce la robustezza di DARKER, capace di contenere gli attacchi non rilevati, evitando il collasso sistemico manifestato da DER. Nel complesso, i riscontri ottenuti su entrambi i dataset forniscono una conferma a sostegno dell'ipotesi fondante di DARKER, dimostrando come l'integrazione di una gestione class-aware del buffer episodico e di una distillazione dinamica dei logits consenta di delineare un compromesso più equilibrato tra stabilità e plasticità rispetto alle strategie presenti in letteratura. Questo miglior equilibrio non si traduce in una dominanza assoluta di DARKER, ma in un comportamento complessivo più robusto contro improvvisi cedimenti strutturali, in grado di evitare fallimenti sistemici.

Le prospettive di sviluppo futuro derivanti dal presente studio si concentrano su due direttrici principali. Una prima estensione riguarda la transizione dal replay basato su campioni storici reali verso paradigmi di Generative Replay, la quale consentirebbe di superare i vincoli strutturali di allocazione di memoria e i requisiti di conformità normativa in materia di privacy, preservando i benefici legati alla distillazione dei logits. Parallelamente, una seconda estensione concerne l'arricchimento del segnale di feedback utilizzato per la regolazione del coefficiente di distillazione. Tale arricchimento permetterebbe l'implementazione di una funzione di controllo capace di intercettare tempestivamente forme di degradazione prestazionale ancora più lievi.

In conclusione, il presente lavoro di tesi dimostra come l'integrazione di questi meccanismi possa contribuire a colmare il divario esistente tra l'efficacia teorica del Continual Learning e le stringenti esigenze operative degli Intrusion Detection Systems, mentre DARKER si configura come un avanzamento concreto verso lo sviluppo di sistemi di sicurezza adattivi capaci di rilevare attacchi cyber senza compromettere la memoria storica.

Bibliografia

- [1] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *Journal of network and computer applications*, vol. 36, no. 1, pp. 16–24, 2013.
- [2] L. H. Yeo, X. Che, and S. Lakkaraju, “Understanding modern intrusion detection systems: a survey,” *arXiv preprint arXiv:1708.07174*, 2017.
- [3] K. Scarfone, P. Mell *et al.*, “Guide to intrusion detection and prevention systems (idps),” *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.
- [4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
- [5] M. Masdari and H. Khezri, “A survey and taxonomy of the fuzzy signature-based intrusion detection systems,” *Applied Soft Computing*, vol. 92, p. 106301, 2020.
- [6] L. O. Nweke, “A survey of specification-based intrusion detection techniques for cyber-physical systems,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, 2021.
- [7] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural networks*, vol. 113, pp. 54–71, 2019.
- [8] G. M. Van de Ven, T. Tuytelaars, and A. S. Tolias, “Three types of incremental learning,” *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.
- [9] M. Mermillod, A. Bugaiska, and P. Bonin, “The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects,” p. 504, 2013.
- [10] W. C. Abraham and A. Robins, “Memory retention—the synaptic stability versus plasticity dilemma,” *Trends in neurosciences*, vol. 28, no. 2, pp. 73–78, 2005.

-
- [11] M. De Lange, G. van de Ven, and T. Tuytelaars, “Continual evaluation for lifelong learning: Identifying the stability gap,” *arXiv preprint arXiv:2205.13452*, 2022.
- [12] L. Wang, X. Zhang, H. Su, and J. Zhu, “A comprehensive survey of continual learning: Theory, method and application,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 8, pp. 5362–5383, 2024.
- [13] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.
- [14] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.
- [15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.
- [17] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.
- [18] F. Hinder, V. Vaquet, and B. Hammer, “One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift,” *Frontiers in Artificial Intelligence*, vol. 7, p. 1330257, 2024.
- [19] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, “A survey on machine learning for recurring concept drifting data streams,” *Expert Systems with Applications*, vol. 213, p. 118934, 2023.

-
- [20] M. Soltani, K. Khajavi, M. Jafari Siavoshani, and A. H. Jahangir, "A multi-agent adaptive deep learning framework for online intrusion detection," *Cybersecurity*, vol. 7, no. 1, p. 9, 2024.
- [21] S. Prasath, K. Sethi, D. Mohanty, P. Bera, and S. R. Samantaray, "Analysis of continual learning models for intrusion detection system," *IEEE Access*, vol. 10, pp. 121 444–121 464, 2022.
- [22] S. K. Amalapuram, A. Tadvai, R. Vinta, S. S. Channappayya, and B. R. Tamma, "Continual learning for anomaly based network intrusion detection," in *2022 14th International Conference on COMMunication Systems & NETWORKS (COMSNETS)*. IEEE, 2022, pp. 497–505.
- [23] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer, "Class-incremental learning: survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2022.
- [24] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017.
- [25] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *Advances in neural information processing systems*, vol. 33, pp. 15 920–15 930, 2020.
- [26] N. Xing, S. Zhao, Y. Wang, K. Ning, and X. Liu, "A dynamic intrusion detection system capable of detecting unknown attacks," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 7, 2023.
- [27] S. K. Amalapuram, T. T. Reddy, S. S. Channappayya, and B. R. Tamma, "On handling class imbalance in continual learning based network intrusion detection systems," in *Proceedings of the First International Conference on AI-ML Systems*, 2021, pp. 1–7.
- [28] P. Bhurani, S. S. Chouhan, and N. Mittal, "Study of class incremental learning strategies for intrusion detection system," in *International Conference on Frontiers of Intelligent Computing: Theory and Applications*, 2022, pp. 281–291.

-
- [29] D. Di Monda, A. Montieri, V. Persico, P. Voria, M. De Ieso, and A. Pescapè, “Few-shot class-incremental learning for network intrusion detection systems,” *IEEE Open Journal of the Communications Society*, vol. 5, pp. 6736–6757, 2024.
- [30] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, “Progress & compress: A scalable framework for continual learning,” in *International conference on machine learning*. PMLR, 2018, pp. 4528–4537.
- [31] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in *International conference on machine learning*. Pmlr, 2017, pp. 3987–3995.
- [32] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, “Efficient lifelong learning with a-gem,” *arXiv preprint arXiv:1812.00420*, 2018.
- [33] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” *ICISSp*, vol. 1, no. 2018, pp. 108–116, 2018.
- [34] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, “Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems,” *Ieee Access*, vol. 8, pp. 165 130–165 150, 2020.
- [35] L. D’hooge, M. Verkerken, B. Volckaert, T. Wauters, and F. De Turck, “Establishing the contaminating effect of metadata feature inclusion in machine-learned network intrusion detection models,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2022, pp. 23–41.
- [36] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.
- [37] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, “Experience replay for continual learning,” *Advances in neural information processing systems*, vol. 32, 2019.
- [38] J. S. Vitter, “Random sampling with a reservoir,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.
- [39] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.

-
- [40] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” *arXiv preprint arXiv:2008.05756*, 2020.
- [41] T. Gowda, W. You, C. Lignos, and J. May, “Macro-average: rare types are important too,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 1138–1157.