UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Progetto e sviluppo di un sistema di analisi del traffico di rete

Tesi di Laurea Magistrale in Ingegneria Informatica

G. Acuto

Relatore: Prof. Giuseppe Lo Re

Correlatore: Ing. Marco Morana

Progetto e sviluppo di un sistema di analisi del traffico di rete

Tesi di Laurea di
Dott. Giuseppe Acuto

Relatore:
Ch.mo Prof. Giuseppe Lo Re

Controrelatore:
Ch.mo Prof. Salvatore Gaglio

Correlatore:
Ing. Marco Morana

Sommario

Le tecniche di classificazione del traffico di rete permettono di associare ad una sequenza di pacchetti scambiati tra due host l'applicazione o il protocollo applicativo da cui il flusso è stato generato. Il processo di identificazione riveste una grande importanza in diversi ambiti, dalla gestione della Qualità del Servizio al monitoraggio di rete per l'individuazione di eventi di interesse. Le tecniche più comunemente utilizzate per il riconoscimento delle applicazioni si basano sull'ispezione diretta del payload o su metodi di *Machine Learning* in grado di classificare i dati sulla base di alcune particolari *features*. Nel presente lavoro viene descritto l'utilizzo di tecniche basate su *Deep Learning* per la classificazione del traffico di rete a partire dall'analisi del payload estratto dai pacchetti TCP. Nel Capitolo 1 vengono mostrati i differenti scenari in cui è possibile effettuare la classificazione del traffico di rete. Nel Capitolo 2 viene descritto lo stato dell'arte riguardante le tecniche di analisi del traffico di rete e le tecniche di *Deep Learning*; queste ultime comunemente utilizzate in molti scenari applicativi, diversi da quello oggetto del presente lavoro, con risultati sorprendenti. Nel Capitolo 3 vengono descritte le tecniche di *Deep Learning* utilizzate per la realizzazione del sistema di analisi oggetto della presente tesi. Nel Capitolo 4 viene descritto il dataset che è stato utilizzato per testare le tecniche di classificazione e il processo di estrazione e di elaborazione dei flussi di traffico. Nel Capitolo 5 sono illustrati i risultati che si sono ottenuti dal processo di classificazione operato su cinque scenari differenti. Infine, nel Capitolo 6 sono presentati e discussi i risultati sperimentali, evidenziando alcuni punti di forza e criticità che presentano le tecniche utilizzate.

Indice

1 Introduzione

La diffusione di un numero sempre maggiore di dispositivi in grado di connettersi ad Internet rende oggi particolarmente rilevante il problema del controllo del traffico di rete. Si osserva, infatti, un numero sempre crescente di utenti che spaziano dai semplici utilizzatori domestici che posseggono più dispositivi agli amministratori di reti più complesse. Il traffico di rete è, dunque, eterogeneo e consiste di flussi provenienti da diverse applicazioni, molte delle quali hanno caratteristiche uniche ed esigenze specifiche rispetto ai parametri di rete, come il ritardo e il throughput. Il soddisfacimento di questi requisiti influenza notevolmente la qualità e la fruibilità di un'applicazione e, quindi, uno dei principali motivi per cui è necessaria la gestione del traffico di rete è quello di garantire che tali requisiti siano soddisfatti in modo da assegnare la corretta larghezza di banda.

Se, da un lato, per un Internet Service Provider (ISP) o un amministratore di rete, risulta relativamente semplice misurare il volume complessivo di traffico che transita sui collegamenti (link) della propria rete, l'informazione ottenuta da una misurazione così aggregata e priva di dettagli si restringe ad una semplice valutazione complessiva del carico al quale sono sottoposti i link e i nodi (router).

L'analisi del traffico di rete nasce, in generale, per risolvere due problemi. In primo luogo, per conoscere le caratteristiche dei flussi di traffico che transitano nella rete, come la durata dei flussi, il volume di traffico che ognuno di essi trasporta, la loro velocità di trasmissione e il grado di variabilità di questa velocità. Questo tipo di analisi è denominata *flow analysis* (analisi a livello di flusso) e permette una gestione delle risorse più efficiente di quella ottenuta da una semplice misurazione del livello di carico complessivo dei link. Il secondo problema è quello di riconoscere la tipologia di applicazione e il protocollo che ha generato il flusso di traffico. L'analisi che permette di ottenere questa informazione è l'identificazione (o classificazione) del traffico e rappresenta solo il primo passo in un meccanismo di gestione dei flussi di Internet. Tipicamente, infatti, una volta che un flusso di rete viene classificato come appartenente a uno specifico protocollo o a una particolare applicazione, viene contrassegnato o marcato in modo da aiutare gli ISP o gli amministratori di rete a determinare le appropriate politiche di servizio da applicare a quel flusso. L'implementazione di tale analisi nei router potrebbe consentire la messa a punto di tecniche avanzate di controllo delle risorse, che possa superare i

classici modelli di controllo della congestione [57], [58], [59], [60], abilitando nuovi servizi in grado di offrire differenti qualità del servizio Qualità del Servizio (QoS).

L'identificazione del traffico è infatti un utile strumento per una gestione della QoS differenziata per le diverse applicazioni. Infatti, se si riconosce l'applicazione che sta generando un dato flusso di traffico, si possono identificare i requisiti di QoS specifici per quell'applicazione (per esempio, in termini di throughput e ritardo) e quindi allocare le risorse necessarie per garantire che questi requisiti siano rispettati, con una conseguente soddisfazione degli utenti. Questa attività è usualmente denominata *QoS management*. Un'altra attività di gestione del traffico, realizzata tipicamente dagli ISP, è il cosiddetto *traffic engineering*, un'operazione che consiste nel determinare, per un dato flusso di traffico o per una data categoria di flussi, il percorso migliore all'interno della rete, per meglio rispettare i requisiti di qualità del servizio delle applicazioni. Un'attività di *traffic engineering* mirata alla gestione della QoS richiede che i requisiti di qualità del servizio siano noti e, quindi, che si conduca un'attività di identificazione del traffico.

Inoltre, una corretta identificazione delle applicazioni e dei protocolli del traffico di rete è essenziale per implementare politiche di sicurezza adeguate. La classificazione delle applicazioni e dei protocolli permette, infatti, di realizzare ed applicare politiche volte ad incrementare la sicurezza in special modo di reti private appartenenti ad aziende che hanno bisogno di prevenire attacchi dall'esterno e di renderle il meno vulnerabili possibile. Questa attività di *anomaly detection* consiste, per esempio, nell'individuare un carico di traffico anomalo per una data applicazione in rete. Una situazione di questo tipo può essere generata da attacchi di tipo *Denial of Service* o *Distributed Denial of Service*, nel quale un insieme di computer "infettati" genera un carico di traffico focalizzato verso risorse mirate (per esempio un sito web che si intende inabilitare). Se si rileva una distribuzione anomala del traffico, si possono intraprendere misure reattive per controbattere questo tipo di attacchi.

È opportuno citare, infine, la possibilità di fornire risorse differenziate a diverse tipologie di applicazioni. È chiaro, infatti, che se il traffico peer-to-peer consuma la maggior parte delle risorse di rete senza autorizzazione, un ISP potrebbe fornire meno banda alle applicazioni che lo generano, o bloccarle direttamente. Proprio grazie alla classificazione del traffico di rete, nel 2009, si è notato come il

traffico generato dalle applicazioni di tipo peer-to-peer per il file sharing (come ad esempio eMule) ammontava a circa l'80% del complessivo traffico di Internet [1].

Il lavoro che viene affrontato in questa tesi mira a presentare i risultati ottenuti dall'utilizzo di due tecniche di *Deep Learning* per la classificazione del traffico di rete. Sebbene esistano diverse tecniche comunemente utilizzate oggi, queste si basano principalmente su metodi di *Deep Packet Inspection* o di *Machine Learning*. Nel primo caso viene effettuata un'ispezione profonda del *payload* di ogni flusso che richiede un alto onere computazionale e le rende poco utilizzabili in contesti di classificazione online. Le seconde, invece, richiedono un grande lavoro preliminare nella ricerca di caratteristiche abbastanza rappresentative di ogni flusso e possono essere utilizzate in un processo di classificazione online. In ogni caso è sempre richiesto un continuo aggiornamento di tali tecniche per tenerle al passo con l'evoluzione della rete. Le tecniche di *Deep Learning*, invece, si fondano su algoritmi di apprendimento che permettono di estrarre in maniera automatica un insieme di features a partire dai flussi di rete che analizzano. La fase di apprendimento, come per le tecniche di Machine Learning classiche, viene effettuata offline, ma il processo di classificazione consiste nel semplice calcolo di una funzione e risulta quindi vantaggioso per un utilizzo online. In questo contesto, però, l'analisi dei flussi di traffico richiede un ridimensionamento dei dati tramite l'applicazione di un vettore di features in modo da poter fornire un vettore di dati di dimensione fissa, indispensabile per le tecniche di *Deep Learning*. Per tale motivo i risultati che verranno analizzati saranno confrontati con quelli ottenuti dall'utilizzo di un albero di decisione, una tecnica di Machine Learning che, in letteratura, ha portato i risultati migliori nell'ambito della classificazione del traffico di rete.

Il processo di classificazione viene effettuato sia a livello di protocollo che a livello di applicazione, prendendo in considerazione scenari differenti e basandosi unicamente sull'utilizzo delle informazioni contenute nel payload dei pacchetti TCP. In una prima analisi vengono presentati e analizzati i risultati ottenuti dalla classificazione di flussi di traffico completi. Questo processo non permette l'utilizzo di tali tecniche in un contesto real-time, visto l'eccessivo ammontare di dati, ma permette di dare una prima valutazione dei metodi che vengono utilizzati. Vengono quindi presi in considerazione tre contesti differenti in cui il processo di classificazione viene applicato lavorando, rispettivamente, sul payload del primo pacchetto, dei primi quattro pacchetti e dei primi dieci pacchetti. Questa analisi consente la valutazione del comportamento dei metodi di *Deep Learning* in un

contesto online, permettendo di confrontare i risultati con quelli ottenuti dall'albero di decisione. Infine, per utilizzare tutte le potenzialità dei metodi di *Deep Learning* e valutare la loro applicazione in un processo di classificazione online, vengono analizzati i risultati ottenuti lavorando direttamente sui primi 1024 byte del payload del primo pacchetto senza darne alcun altro tipo di rappresentazione. Si noterà come in questo ultimo caso si otterranno i risultati migliori.

2 Stato dell'arte

In questo capitolo viene presentato lo stato dell'arte relativo all'analisi del traffico di rete e alle tecniche di *Deep Learning* che vengono utilizzate in questo lavoro per il processo di classificazione.

Nella prima parte vengono descritte le tecniche esistenti per la classificazione del traffico di rete, evidenziando le idee che ne stanno alla base e i motivi che portano a ricercare soluzioni sempre più innovative. Nella seconda parte del capitolo viene introdotto il *Deep Learning* e le sue applicazioni in scenari differenti da quello oggetto della presenti tesi, mostrando in particolar modo i risultati ottenuti nell'ambito dell'elaborazione delle immagini.

2.1 Tecniche di analisi del traffico di rete

Per classificazione del traffico di rete si intende l'utilizzo di tutte quelle tecniche che mirano ad analizzare ed ispezionare il flusso di dati proveniente dalla rete di Internet e che si differenziano tra loro per i metodi utilizzati e la scelta dei parametri decisionali. Il processo di classificazione viene tipicamente utilizzato per identificare i protocolli applicativi e le diverse applicazioni che vengono continuamente sviluppate ed utilizzate in rete.

Le prime tecniche di classificazione del traffico di rete per l'identificazione delle applicazioni sono state utilizzate fin dai primi tempi di internet, in particolar modo a partire dagli anni novanta, e vengono tutt'oggi utilizzate in combinazione a nuovi approcci. Queste si basavano sulla conoscenza dei numeri di porta noti, i quali venivano assegnati e registrati presso l'autorità internazionale IANA [2]. Questa soluzione è risultata essere molto efficiente e relativamente accurata per molto tempo, fino all'avvento delle prime applicazioni peer-to-peer (in particolar modo a partire dal 2002 con la nascita del progetto eMule) che segnarono un primo

punto di svolta rispetto alle tradizionali applicazioni. Se da una parte gli amministratori di rete erano soliti rispondere al grande consumo di banda da parte di queste applicazioni bloccando direttamente le porte che esse utilizzavano di default, dall'altra parte gli sviluppatori di questa famiglia di applicazioni cominciarono ad utilizzare nuovi metodi per raggirare il problema. Il primo espediente è stato quello di utilizzare porte dinamiche o, molto spesso, le stesse porte utilizzate da altre applicazioni note, in modo da rendere definitivamente inefficienti le tecniche di classificazione utilizzate fino a quel momento. Come risposta, negli stessi anni, la comunità scientifica introdusse nuovi metodi di classificazione del traffico di rete che presero il nome di *Deep Packet Inspection* (DPI). Questi metodi, che hanno avuto un grande sviluppo a partire dagli anni 2000 e che attualmente continuano ad essere utilizzati, si basavano su un'ispezione diretta del contenuto dei pacchetti trasmessi in rete, alla ricerca di alcuni pattern nei payload in grado di caratterizzare ogni applicazione. Il punto debole di questi metodi risiedeva, però, nell'alto costo computazionale che richiedeva una tale operazione. Nel frattempo, alcune applicazioni iniziarono a cifrare il contenuto dei payload trasmesso in rete, costringendo i metodi basati su DPI a ricercare nuove soluzioni. Per ovviare a queste limitazioni cominciano ad essere presentati in letteratura nuovi metodi che si basano su tecniche di *Machine Learning*. In generale, tutte le tecniche di *Machine Learning* che tutt'ora vengono utilizzate ricercano, in una fase offline, la relazione che lega un insieme di caratteristiche estratte dal contenuto informativo di un certo numero di pacchetti di rete (come ad esempio i numeri di porta, la dimensione del payload o l'intervallo di una trasmissione) all'applicazione che ha generato quei pacchetti. Questo insieme di caratteristiche viene utilizzato per costruire un modello che viene poi applicato per identificare il traffico di rete in una fase online. Per poter ridurre, almeno parzialmente, la precisione delle tecniche basate su metodi di *Machine Learning*, le applicazioni di rete hanno iniziato ad implementare metodi di offuscamento che, in maniera del tutto casuale, modificano le caratteristiche delle informazioni che trasportano.

Oltre alle tecniche di *Machine Learning* e DPI, sono stati proposti altri approcci che non mostrano, però, miglioramenti significativi rispetto alle tecniche sopra citate. Ad esempio, sono stati implementati dei metodi alternativi che adottano un approccio differente, conosciuti come metodi basati sul comportamento dell'host (*host-behavior-based methods*). Questi metodi attuano una classificazione che si basa sul comportamento degli utenti finali che producono traffico nella rete.

Un'altra soluzione è rappresentata dalle tecniche basate sugli indirizzi IP (*IP-based techniques*), simili a quelle basate sulle porte note. Questo approccio utilizza le informazioni che vengono estratte, in una fase offline, da un insieme di indirizzi IP che provengono da una applicazione ben nota (ad esempio Youtube) per classificare il traffico legato a queste applicazioni. Sebbene molte delle soluzioni esistenti possono raggiungere alti valori di accuratezza, non esiste un metodo universale applicabile ad ogni possibile scenario di rete.

Nei prossimi paragrafi analizzeremo nel dettaglio alcune di queste tecniche.

2.1.1 **L'architettura di Internet**

Tramite le reti di calcolatori, più host possono essere messi in grado di comunicare tra loro, permettendo il trasferimento di dati di diverso genere e l'espletamento di servizi (e-mail, trasferimento file, navigazione Web, ecc…). Qualunque sia la natura dei dati scambiati, gli host devono seguire delle regole ben precise e concordate a priori tramite l'esecuzione di specifici protocolli. Un protocollo definisce il formato e l'ordine dei messaggi scambiati tra due o più entità in comunicazione, cosi come le azioni intraprese in fase di trasmissione e ricezione di un messaggio o di un altro evento.

L'architettura di una rete di calcolatori viene definita da una struttura stratificata di livelli, ognuno dei quali è regolamentato da un insieme di protocolli di comunicazione. Nel corso degli anni sono stati proposti due modelli per descrivere la struttura dei protocolli nei vari livelli, la quale prende il nome di pila di protocolli. Il primo modello prende il nome di ISO/OSI (Open System Interconnection) ed è stato introdotto dall'ISO (International Organization for Standardization). Il modello che invece si è affermato come standard con l'evoluzione di Internet è rappresentato dalla suite TCP/IP, che prende il nome dei due principali protocolli che ne fanno parte, TCP (Transmission Control Protocol) e IP (Internet Protocol). Il modello OSI, proposto negli anni settanta, prese forma quando i protocolli, che sarebbero diventati i protocolli di Internet, erano ai loro esordi ed erano solo uno dei tanti insiemi di protocolli in via di sviluppo. Secondo tali architetture, le funzionalità dei protocolli sono divise in strati, ognuno dei quali:

- fornisce servizi allo strato superiore
- richiede servizi allo stato inferiore
- comunica con gli strati adiacenti tramite un'interfaccia standard

- comunica con lo strato di pari livello (*peer*) di un'altra entità secondo un protocollo assegnato

Il modello TCP/IP (Figura 1) è organizzato in cinque livelli (invece dei 7 dello standard ISO/OSI). Lo strato *n* di un nodo di destinazione riceve un messaggio (o pacchetto o frame, a seconda del livello considerato) identico a quello trasmesso dallo stesso livello nel nodo sorgente.

Il livello di applicazione è la sede delle applicazioni di rete e dei relativi protocolli applicativi. Tale strato include molti protocolli, quali ad esempio HTTP (che consente la richiesta e il trasferimento di documenti web), SMTP (che consente il trasferimento di messaggi di posta elettronica) o FTP (per il trasferimento di file tra due sistemi remoti). Un protocollo di livello applicativo è distribuito su più sistemi terminali e permette lo scambio di informazioni (che prendono il nome di messaggi) tra terminali in cui è presente la stessa applicazione.

Figura 1 Pila di protocolli modello TCP/IP

Il livello di trasporto di Internet trasferisce i messaggi del livello di applicazione tra un modulo client e un modulo server di un'applicazione. In Internet i due protocolli di trasporto sono TCP e UDP. Il primo fornisce alle applicazioni un servizio orientato alla connessione, mentre il secondo fornisce alle proprie applicazioni un servizio senza connessione. I pacchetti a questo livello prendono il nome di segmenti ed in Figura 2 viene mostrata la struttura di un tipico segmento TCP.

Il livello di rete si occupa di trasferire i pacchetti (datagrammi) attraverso la rete da un host ad un altro ed utilizza il protocollo IP.

Il livello di collegamento instrada un datagramma attraverso una serie di router tra l'origine e la destinazione e i pacchetti di questo livello prendono il nome di frame.

Il compito del livello fisico, infine, è quello di trasferire i singoli bit di un frame da un nodo a quello successivo. I protocolli di questo livello sono dipendenti dal collegamento e dall'effettivo mezzo trasmissivo.

Nel lavoro oggetto di questa tesi, il processo di classificazione viene applicato ad un insieme di informazioni estratte dal livello applicativo. La maggior parte delle applicazioni di rete consiste di coppie di processi comunicanti che si scambiano dei messaggi, accedendo alla rete sottostante per mezzo di una porta. Un processo invia e riceve i messaggi in rete tramite un'interfaccia presente tra il livello di applicazione e il livello di trasporto interno ad un host, che prende il nome di socket.

Figura 2 Struttura di un segmento TCP

I messaggi che vengono inviati tramite socket vengono definiti dal protocollo applicativo utilizzato, che definisce la tipologia di messaggi scambiati, la struttura e la semantica dei campi del messaggio e le regole per determinare il modo in cui un processo deve inviare o rispondere ad un messaggio. Non tutti i protocolli applicativi sono di pubblico dominio, come lo sono HTTP, SMTP e POP3, che vengono specificati dalle RFC, alcuni, infatti, sono proprietari e non rendono pubbliche tutte le loro specifiche, come ad esempio eMule.

Questa è la ragione per cui, nel corso degli anni, sono state ricercate diverse tecniche per permettere l'identificazione di tutta quella famiglia di applicazioni che ha pesantemente influenzato la rapida crescita di Internet.

2.1.2 **Metodi basati sulle porte**

I metodi basati sulle porte rappresentano una delle prime tecniche adoperate dai *service providers* e dagli amministratori di rete per monitorare il traffico di Internet. Tali tecniche si basavano sulla conoscenza delle porte note, che venivano assegnate a specifici servizi applicativi dalla IANA (*Internet Assigned Numbers Authority*), un'organizzazione internazionale responsabile dell'assegnazione degli indirizzi IP.

L'identificazione del traffico di rete si riduceva, dunque, ad un semplice confronto tra il numero di porta estratto da un pacchetto IP e l'elenco delle porte note registrate dalla IANA, tipicamente quelle che vanno da 0 a 1023. Ad esempio, le web application utilizzavano la porta 80 e le applicazioni per l'invio di email la porta 25. Nella Tabella 1 vengono portati alcuni esempi di porte note e corrispettivi protocolli.

Porta	Protocollo
20	**FTP – data**
22	**SSH**
23	**Telnet**
25	**SMTP**
80	**HTTP**
88	**Kerberos**
110	**POP3**

Tabella 1 Esempio di porte note registrate tramite IANA

Con l'avvento di nuove applicazioni, in particolar modo con le applicazioni *peer-to-peer* (P2P), questo approccio inizia a risultare inapplicabile. Diventa, infatti, impossibile riuscire ad identificare tutta quella nuova generazione di applicazioni che richiedono un utilizzo massiccio della banda di rete in quanto, queste ultime, cominciano ad utilizzare nuove strategie per mascherare la propria identità. Si iniziano a vedere le prime applicazioni che utilizzano porte dinamiche o porte ben note utilizzate e ad associate ad applicazioni che erano conosciute come affidabili, come ad esempio la porta 80 del protocollo HTTP. Ad oggi le porte utilizzate dalle applicazioni vengono utilizzate in altre tecniche in aggiunta ad altre informazioni più caratterizzanti.

2.1.3 **Deep Packet Inspection**

Una delle prime alternative al metodo basato sulle porte è stata l'ispezione diretta dei payload dei pacchetti per l'identificazione del traffico di rete. Questo approccio comprende tutti quei meccanismi che vengono più comunemente denominati di *Deep Packet Inspection* (DPI). Molti di questi metodi utilizzano la *Signature Analysis* per l'analisi e l'ispezione di applicazioni differenti, la quale permette di verificare la presenza o meno di alcune *signatures* che caratterizzano ogni applicazione.

Le *signatures* rappresentano dei pattern unici che vengono associati ad ogni applicazione. Ogni applicazione viene studiata per ricercare alcune sue caratteristiche peculiari e successivamente viene creato un database di riferimento contenente tutti i pattern associati ad ogni applicazione. A questo punto, il metodo di classificazione consiste nel confrontare il traffico che si intende analizzare con il database che è stato precedentemente creato in modo da identificare l'applicazione esatta.

Application	Protocol	Feature string	Position
Oicq	UDP	0x02	0x2a
Thunder	UDP	0x3b000000 0x3c000000 0x32000000	0x2a-0x2d
	TCP	0x36000000	0xb7-0xba
Bit Torrent	TCP	0x13426974	0x36-0x39
QQDownload	UDP	0xfe+length (payload-3)	0x2a-0x2c
QQlive	UDP	0xfe+length (payload-3)	0x2a-0x2c
PPlive	UDP	0x010000	0x34-0x36

PPS	UDP	0x80/0x84	0x2b
	UDP	length(payload-6)+0x4300	0x2a-0x2d
eDonkey/Emule	UDP	e3 or c5 or d4	0x2a
	TCP	e3 or c5 or d4	0x36

Tabella 2 Codici caratteristici di alcune applicazioni P2P

Questo approccio comporta un aggiornamento periodico del database per tenerlo sempre aggiornato sia con le nuove applicazioni che vengono costantemente create, che con i continui sviluppi a cui sono soggette le applicazioni già esistenti. Il lavoro che viene compiuto con questo nuovo approccio mira soprattutto ad identificare applicazioni di tipo P2P in quanto, come già accennato precedentemente, usano in maniera assidua strategie per camuffare ed evitare la propria identificazione. La Tabella 2 riporta un esempio di pattern utilizzati per alcune applicazioni P2P [3].

Esistono diversi metodi di *Signature Analysis* e i più diffusi software di DPI, quali ad esempio PACE e NBAR, differiscono proprio per il metodo o i metodi utilizzati.

Tra gli approcci più utilizzati, riportati in Figura 3 [4], possiamo distinguere i seguenti metodi:

- *Pattern analysis*
- *Numerical analysis*
- *Behavioral analysis*
- *Heuristic analysis*
- *Protocol/state analysis*

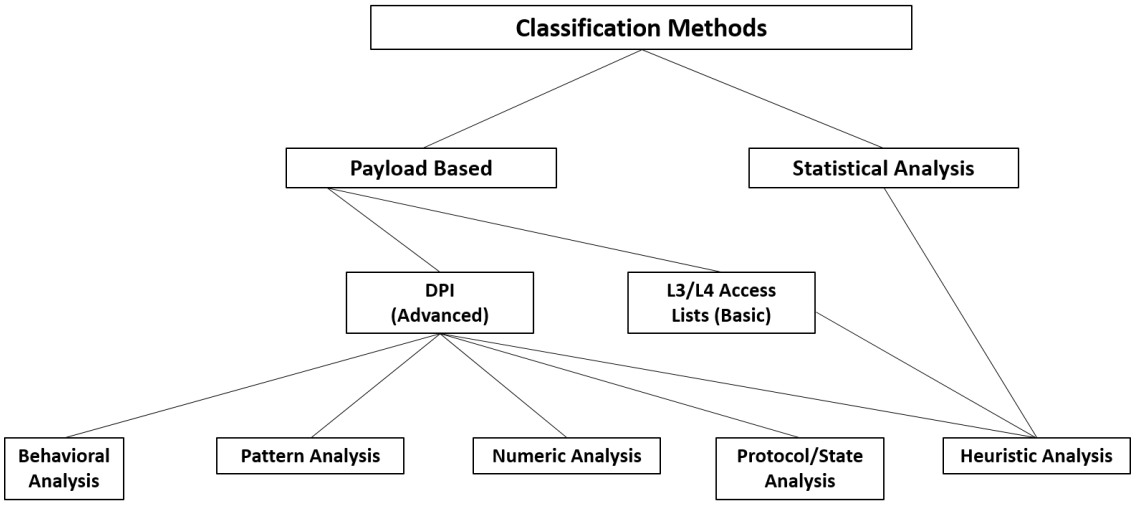

Figura 3 Tecniche e metodi di classificazione

I metodi di *pattern analysis* rendono esplicita l'assunzione che alcune applicazioni includono nei payload dei loro pacchetti alcuni pattern (composti da byte, caratteri o stringhe) che possono essere utilizzati dal processo di classificazione per identificare alcuni protocolli. Per alcune applicazioni queste sequenze non sono sempre collocate nella stessa posizione ma possono essere presenti in qualsiasi altro posto all'interno del payload. Sebbene queste tecniche riescano a classificare correttamente i pacchetti generati da queste applicazioni, non tutti i protocolli prevedono sequenze speciali di caratteri o stringhe nei loro pacchetti e dunque questo approccio non funziona con questi ultimi.

I metodi di analisi numerica, invece, prevedono la ricerca di caratteristiche numeriche all'interno del contenuto del pacchetto, come la dimensione del payload o il numero di pacchetti ricevuti come risposta ad una richiesta. Ad esempio, le prime versioni di Skype prevedevano un messaggio di richiesta da parte del client di dimensione pari a 18 byte e la risposta che ricevevano era di solito costituita da un messaggio di 11 byte. Dal momento che questo tipo di analisi viene tipicamente fatta su un certo numero di pacchetti consecutivi, il processo di classificazione richiede un tempo considerevole.

I metodi di analisi del comportamento e di analisi delle euristiche vengono tipicamente utilizzati insieme in molti programmi antivirus per identificare *malware* e *worm*. Il primo viene solitamente utilizzato per analizzare il comportamento di un certo flusso di traffico di rete per produrre una descrizione delle applicazioni che possono averlo generato. Il secondo, invece, genera delle statistiche dai pacchetti che ispeziona e viene solitamente utilizzato per classificare il protocollo utilizzato.

Infine, i metodi di *protocol/state analysis* vengono utilizzati per quelle applicazioni in cui il protocollo applicativo segue una prestabilita sequenza di step o azioni. Ad esempio, una tipica richiesta FTP GET da parte di un client è seguita da una risposta di validazione da parte del server. I comportamenti di alcuni protocolli vengono dunque utilizzati per classificare alcune applicazioni.

Nel momento in cui molte applicazioni hanno iniziato a cifrare il traffico da loro generato, però, molti di questi metodi sono diventati inutilizzabili se utilizzati da soli. I metodi di analisi del comportamento e delle euristiche possono però aiutare ad identificare alcune applicazioni. In generale, ormai, i software commerciali di DPI utilizzano tutti questi metodi insieme, o un sottoinsieme di essi. Inoltre, sebbene tali software riescano a raggiungere un alto grado di accuratezza per la classificazione di molte applicazioni, essi richiedono un alto consumo di risorse che comporta l'utilizzo di hardware dedicato e spesso costoso per poter lavorare online.

2.1.4 **Tecniche basate su Machine Learning**

Le tecniche basate su *Machine Learning* vengono ampiamente utilizzate da molto tempo in tantissimi campi, come il *search engine* o la diagnostica medica solo per citarne alcuni. Una delle sue prime applicazioni nell'ambito delle reti [5] risale al 1990 con l'obiettivo di ottimizzare il controllo del flusso di una rete, che a quel tempo era prevalentemente quella telefonica. Il *Machine Learning* viene introdotto, nell'ambito dell'analisi del traffico di rete, in modo da poter superare le limitazioni dovute ai metodi basati su ispezione diretta del payload (DPI). L'obiettivo era quello di trovare un meccanismo alternativo che permettesse l'identificazione delle applicazioni di rete senza la necessità di ispezionare il contenuto di un pacchetto. Le tecniche di *Machine Learning* possono essere suddivise, in base allo scenario d'utilizzo, in tecniche di classificazione e tecniche di *clustering*.

La classificazione, Figura 4, fa parte delle tecniche di apprendimento supervisionato, le quali costruiscono un modello a partire da un insieme di addestramento già classificato che sia in grado di separare i campioni dell'insieme e di classificarne di nuovi. Il *clustering*, invece, fa parte delle tecniche di apprendimento non supervisionato, che non necessitano di un insieme di addestramento pre-classificato ma che cercano una suddivisione naturale, in *cluster*, in base alla similarità dei campioni che compongono l'insieme (Figura 5).

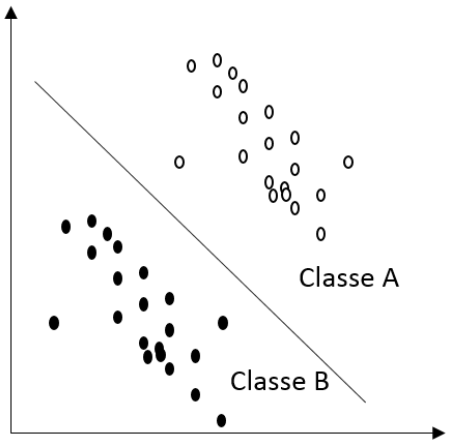

Figura 4 Classificazione

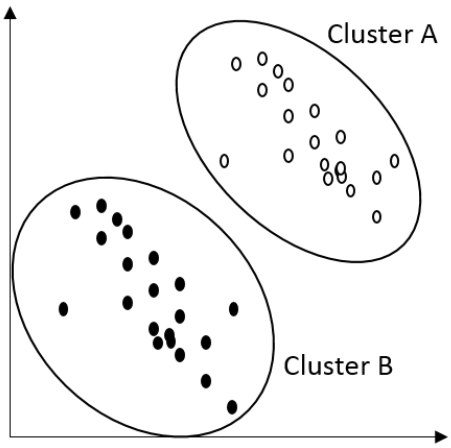

Figura 5 Clustering

In generale i metodi di *Machine Learning*, sia supervisionati che non supervisionati, utilizzano un dataset dal quale sono state estratte precedentemente un insieme di *features*. Questo insieme rappresenta l'input dal quale la tecnica di *Machine Learning* estrae un modello, che prende il nome di struttura di conoscenza, che viene restituito sotto forma di struttura. Dalla tecnica di ML utilizzata dipende la tipologia di struttura che viene prodotta, che ad esempio può essere un albero decisionale, un insieme di regole o un insieme di *cluster*.

Nei prossimi due paragrafi viene data una breve descrizione dei due tipi di apprendimento, che nell'ambito della classificazione delle applicazioni di rete vengono descritti e confrontati nel dettaglio in [6].

2.1.4.1 Apprendimento supervisionato

Tra i metodi supervisionati, anche conosciuti come metodi di classificazione, fanno parte diversi algoritmi. L'obiettivo di ognuno di questi algoritmi è la costruzione di una struttura di conoscenza, ovvero il classificatore, che sia in grado di associare ad ogni campione una classe di appartenenza. Tali algoritmi ricevono come input un insieme di campioni con una classe pre-assegnata, dal quale costruiscono il proprio modello di classificazione secondo una specifica struttura. Il processo di apprendimento supervisionato inizia con un insieme di addestramento, *TS*, definito come un insieme di coppie di tipo input/output $TS = < x_1, y_1 > , < x_2, y_1 >, \cdots, < x_N, y_M >$, dove x_i è il vettore di input contenente i valori delle caratteristiche del i-esimo campione e y_i è il valore della classe in uscita. Tale processo mira a scoprire le relazioni che collegano le diverse istanze date in input e produce come output una struttura, tipicamente un albero decisionale o un insieme

di regole di classificazione, che permette di classificare le istanze in un insieme discreto, $y_1, y_2, \cdots, y_M$, di categorie. Successivamente si passa ad una fase di testing, con la quale si misura il corretto comportamento del modello. In questa fase viene dato in pasto al modello un insieme di nuovi dati classificati non facenti parte dell'insieme di addestramento. Dai risultati ottenuti da questo nuovo insieme si valuta il comportamento finale della struttura di conoscenza.

2.1.4.2 Apprendimento non supervisionato

A differenza dei metodi di apprendimento supervisionato, i metodi non supervisionati, anche conosciuti come metodi di *clustering*, non necessitano di un insieme di addestramento completamente etichettato. Infatti, il comportamento di tali tecniche non prevede di classificare le istanze date come input in un numero di classi predefinito, ma è il modello stesso che scopre il naturale raggruppamento dei dati. In generale raggiungono un livello di accuratezza leggermente minore rispetto ai metodi supervisionati, ma hanno una fase di addestramento computazionalmente meno onerosa. Dal momento che i metodi di *clustering* restituiscono come uscita un cluster con nessun valore semantico, quando vengono utilizzati per la classificazione del traffico di rete occorre anche descrivere il cluster esportato dalla fase di addestramento e assegnargli una classe o categoria per poter attuare il processo di classificazione, come mostrato in Figura 5.

Date le caratteristiche sopra elencate, i metodi di classificazione del traffico di rete basati su *Machine Learning* risultano essere la soluzione più adottata in letteratura. Ad ogni modo, come per i metodi basati su *Deep Packet Inspection*, queste tecniche necessitano di essere aggiornate periodicamente per adattare il modello di classificazione alle nuove applicazioni.

2.1.5 Approccio basato sullo studio del comportamento degli host

Un altro approccio che mira a risolvere le limitazioni costituite dalle tecniche basate sull'analisi del payload, è rappresentato da un insieme di tecniche che mirano ad acquisire informazioni estraendo delle caratteristiche a partire dal comportamento dell'host. Karagiannis [7] e Xu [8] hanno sviluppato un metodo di classificazione basato proprio sull'analisi del comportamento dell'utente. In particolare, l'approccio proposto da Karagiannis studia il comportamento di un insieme di host presentando tre livelli distinti:

- Un livello sociale in cui viene valutata la popolarità di un host, prendendo in considerazione il numero di comunicazioni che esso effettua con altri host
- Un livello funzionale in cui viene valutato il ruolo dell'host all'interno della rete, in cui si cerca di scoprire se esso è un fornitore o un consumatore di un servizio, o se esso partecipa ad un sistema di comunicazioni collaborative. Ad esempio, un host che effettua un gran numero di connessioni utilizzando la stessa porta è molto probabile che sia un provider, ovvero un fornitore di un servizio su quella porta.
- Un livello applicativo in cui vengono catturate le interazioni tra host a livello di trasporto cercando di identificare l'applicazione d'origine.

Nel lavoro di Karagiannis vengono utilizzate alcune euristiche, in aggiunta a queste tre metriche, per la classificazione del comportamento delle applicazioni. Sebbene tale metodo riesce a classificare i protocolli applicativi basandosi su comportamenti di gruppi di host, esso non è in grado di classificare esattamente le applicazioni.

2.1.6 **Approccio basato su indirizzi IP**

Gli indirizzi IP possono trasportare importanti informazioni legate al traffico che essi producono. Seguendo un approccio simile alle prime tecniche che si basavano sulla conoscenza delle porte note, questi metodi usano le informazioni legate agli indirizzi IP per classificare il traffico di rete. Possiamo individuare due differenti approcci che fanno suo di queste tecniche.

La prima tecnica [9] si basa sull'utilizzo dei soli indirizzi IP appartenenti alle applicazioni più conosciute per effettuare una classificazione del traffico analizzato. Fondamentalmente, questa tecnica prevede una fase offline per individuare tutti gli indirizzi IP che appartengono ai più famosi siti web, come ad esempio Google e Twitter. Una volta identificati, essi vengono utilizzati per classificare direttamente trasmissioni che comunicano con tali indirizzi. Ad esempio, l'indirizzo IP 31.13.83.16 appartiene a Facebook, dunque tutto il traffico che comunica o proviene da quell'indirizzo viene direttamente etichettato come appartenente all'applicazione Facebook. Esistono, inoltre, lavori che permettono di fare un analisi del traffico generato attraverso i social network per individuare eventi di interesse [10] [11].

La seconda tecnica prevede l'utilizzo di metodi basati sul servizio (*Service-based methods*). Un servizio viene definito come quella tripletta <*IP, Porta, Protocollo*> assegnata a una specifica applicazione. La lista dei servizi deve essere creata in una fase offline usando un dataset di flussi di traffico già classificati. Ad esempio, nel lavoro proposto da Yoon [12] i flussi di traffico, raccolti e classificati in una rete interna, vengono raggruppati in base alle loro triplette. Dopo la fase di raggruppamento, il metodo proposto da Yoon riesce a classificare correttamente un servizio applicativo soltanto quando una tripletta possiede un numero minimo di flussi, scelto arbitrariamente, entro un certo intervallo di tempo.

Sebbene queste tecniche abbiano presentato risultati molto accurati, come per molti degli altri approcci descritti fin qui, necessitano comunque di un aggiornamento periodico per poter essere applicate mantenendo la loro precisione. Inoltre, la migrazione di molte applicazioni verso le *Content Delivery Networks*, ovvero reti globali composte da server *proxy* distribuiti in più *data center* per poter fornire contenuti web all'utente finale con alte performance, ha sensibilmente degradato il loro utilizzo.

2.1.7 **Confronto tra i diversi approcci**

Sebbene siano state studiate diverse tecniche per la classificazione di applicazioni di rete, non risulta chiaro quali di queste utilizzino il metodo migliore per identificare un'applicazione in tempo reale. A tal proposito, sono stati pubblicati diversi lavori dalla comunità scientifica per presentare i risultati ottenuti utilizzando diverse tecniche [13] [14] [15].

Nel 2006 Williams, Zander e Armitage hanno presentato, nel lavoro [16], un confronto tra cinque algoritmi di *Machine Learning* con apprendimento supervisionato: *Naive Bayes* con Discretizzazione (NBD), *Naive Bayes* con *Density Kernel Extimation* (NBK), albero di decisione C.45, *Bayes Network* e *Naive Bayes Tree*. Nel loro studio dimostrano che alcuni degli algoritmi presi in esame portano a risultati che non differiscono molto tra di loro, portando gli autori a concentrarsi anche sull'analisi delle prestazioni, confrontando le velocità di classificazione e i tempi di addestramento. Per i loro esperimenti hanno utilizzato il dataset pubblico messo a disposizione dal NLANR [17] dal quale hanno estratto 22 differenti caratteristiche. Dai loro esperimenti conclusero che l'algoritmo C.45 è il più accurato, con una percentuale di accuratezza finale del 99,4%, seguita dal *Bayes Tree* con un'accuratezza del 99,32%. Inoltre, l'albero di decisione C.45 è risultato

essere il metodo che ha richiesto il minor tempo di addestramento ed è stato capace di classificare un insieme di istanze centinaia di volte più veloce di una SVM.

Nello stesso anno viene proposto un lavoro che presenta un confronto tra una tecnica di classificazione supervisionata, *Naive Bayes,* e un approccio non supervisionato, utilizzando il metodo *AutoClass*. Il lavoro conclude che la tecnica non supervisionata raggiunge un'accuratezza del 91%, più alta del 9% di quella raggiunta dalla Naive Bayes, del 93%.

Nel 2015 Bujlow, Carela-Español e Barlet-Ros pubblicano un articolo intitolato "*Independent Comparison of Popular DPI Tools for Traffic Classification*" [15] in cui confrontano le prestazioni di sei differenti software di *Deep Packet Inspection* su un dataset di tracce di rete da loro creato e reso pubblico, contenente più di 750 mila flussi. Nello specifico, utilizzano due prodotti commerciali di DPI, PACE e NBAR, e 4 *tools* open source (OpenDPI, L7-filter, nDPI, Libprotoident). Lo studio viene fatto su diversi livelli di classificazione (protocollo, applicazione e web service) e le prestazioni vengono valutate su diversi scenari, confrontando i risultati ottenuti dall'analisi di tutto il flusso, dall'analisi dei flussi troncati o utilizzando solo pacchetti troncati. I risultati mostrano come la classificazione a livello di applicazione presenti molti più problemi se confrontata con risultati ottenuti a livello di protocollo, in quanto nessuno dei software utilizzati riesce a classificare il 100% dei flussi appartenenti a una specifica applicazione. Nel loro lavoro concludono che non si evince una grande differenza, in termini di accuratezza, nella classificazione effettuata tra flussi completi e flussi troncati e che in generale un peggioramento di prestazioni lo si evince quando il processo di analisi viene applicato ai pacchetti troncati.

Sebbene i metodi di DPI siano quelli che comunemente danno valori di accuratezza migliori, gli algoritmi di *pattern matching* per la classificazione online sono computazionalmente complessi e solitamente richiedono hardware costoso. Dall'altra parte, le tecniche di *ML* risultano più adatte per la classificazione online ma richiedono un attento studio delle features da utilizzare. Come si può notare, possiamo concludere in accordo con la letteratura, che non esiste alcuna tecnica che presenta prestazioni chiaramente superiori rispetto a tutte le altre.

2.2 Tecniche di Deep Learning e applicazioni

Le tecniche di *Deep Learning* sono molto utilizzate in diversi ambiti ma, al meglio della nostra conoscenza, non esistono approcci in cui il *Deep Learning* è stato applicato all'analisi del traffico di rete. In questa sezione viene fornita una panoramica delle tecniche esistenti e delle loro applicazioni in svariati scenari applicativi.

Il *Deep Learning* si riferisce a una classe di tecniche di *Machine Learning* per la classificazione di dati e per l'estrazione di *features,* largamente sviluppate a partire dal 2006 [18]. Tali tecniche consentono di apprendere una rappresentazione dei dati su diversi livelli di astrazione e vengono applicate a modelli computazionali con una struttura stratificata. La loro applicazione, negli ultimi anni, ha portato un'ondata di innovazione in diversi settori, grazie ai risultati ottenuti nell'ambito del riconoscimento di immagini, della localizzazione di oggetti, della scoperta di nuovi farmaci e della genetica, solo per citarne alcuni. Gli algoritmi utilizzati permettono di estrarre la complessa struttura interna di un grande insieme di dati e indicano al modello stratificato in che modo modificare i propri parametri interni. Questi parametri, in ogni strato, vengono aggiornati per creare una propria rappresentazione dei dati a partire dalla rappresentazione presente al livello precedente.

I metodi di *Deep Learning* vengono anche chiamati metodi di apprendimento della rappresentazione (*representation-learning methods*) a livelli multipli di rappresentazione, ognuno dei quali è ottenuto tramite la composizione di moduli semplici ma non lineari. Ogni modulo, partendo da un insieme di dati grezzi, trasforma la rappresentazione da un livello ad un altro con un grado di astrazione sempre maggiore. La composizione di un certo numero di trasformazioni permette di apprendere funzioni molto complesse. Nell'ambito della classificazione, i livelli di rappresentazione più alti permettono di amplificare alcuni aspetti importanti dell'input ed eliminare variazioni irrilevanti. Un'immagine, ad esempio, assume la forma di un array contenente i valori dei pixel e le *features* apprese nel primo livello di rappresentazione, tipicamente, indicano la presenza o meno di *edges* in particolari posizioni e con diverse orientazioni. Il secondo *layer*, invece, ricerca dei motivi nell'immagine individuando particolari disposizioni degli *edges* a prescindere da piccole variazioni e dalle loro posizioni. Il terzo livello assembla i motivi producendo un grande numero di forme, e i *layer* successivi individuano gli oggetti

dell'immagine come combinazione di queste forme. L'aspetto chiave del *Deep Learning* risiede nel fatto che questi livelli di features non vengono studiati e progettati dall'uomo, ma vengono appresi dai dati in input utilizzando una procedura di apprendimento di tipo general-purpose.

Il *Deep Learning* ha portato eccellenti risultati nello scoprire le strutture intricate in dati di grandi dimensioni ed è stato quindi applicato in molti campi della scienza, dell'economia e della medicina. Oltre a battere record nell'ambito del riconoscimento di immagini [19] [20] o nel riconoscimento vocale [21] [22], ha superato altre tecniche di *Machine Learning* nel predire l'attività di potenziali nuove molecole di farmaci [23], analizzare i dati provenienti da un acceleratore di particelle [24] [25], ricostruire i circuiti cerebrali [26] e nel prevedere gli effetti di alcune malattie provocate da mutazioni del DNA [27]. Forse ancora più sorprendente, il *Deep Learning* ha prodotto risultati estremamente promettenti nella comprensione del linguaggio naturale [28], la classificazione di particolari argomenti e la traduzione linguistica [29]. Infine, esistono anche alcuni lavori in cui le tecniche di *Deep Learning* vengono utilizzate per l'identificazione e classificazione di software malevoli [30], in tal caso però, data la dimensione variabile dei dati trattati, vengono utilizzati dei metodi per estrarre manualmente un insieme di features in modo da ridurre la dimensione dei dati di input e fornire una nuovo livello di rappresentazione.

Le architetture che vengono utilizzate per il *Deep Learning* normalmente si riferiscono a reti neurali composte da diversi strati, ognuno dei quali è costituito da un certo numero di nodi, chiamati neuroni, che effettuano trasformazioni non lineari sui dati provenienti dallo strato precedente.

In generale, possiamo distinguere tra tre differenti architetture, ognuna caratterizzata dalla tipologia di tecniche di apprendimento utilizzate:

- Architetture generative: utilizzano un tipo di apprendimento non supervisionato e il loro obiettivo è quello di ridurre la dimensionalità dei dati di input o di dare una rappresentazione della funzione di probabilità congiunta, $p(x,y)$, a partire da un insieme finito che, una volta appresa, permette di generare nuovi campioni a partire da essa.

- Architetture discriminative: utilizzano un tipo apprendimento di tipo supervisionato con l'obiettivo di effettuare una classificazione su di un insieme di dati.
- Architetture ibride: utilizzano un'architettura generativa per effettuare un pre-addestrameno non supervisionato e successivamente effettuano un addestramento supervisionato a fini della classificazione.

2.2.1 **Deep Learning per l'elaborazione delle immagini**

Uno dei settori in cui le tecniche di *Deep Learning* hanno portato al raggiungimento di risultati più straordinari è quello dell'elaborazione delle immagini, in particolar modo con l'utilizzo delle *convolutional neural networks*, la cui struttura e funzionamento viene descritta nel dettaglio nel capitolo successivo.

A partire dai primi anni 2000 il *Deep Learning* è stato applicato con grande successo in diversi ambiti per il rilevamento, la segmentazione e il riconoscimento degli oggetti e delle regioni presenti nelle immagini. Nell'ambito delle immagini, infatti, si erano costruiti dataset di grandi dimensioni con tutti i campioni etichettati. Tra questi, si iniziano a vedere interessanti risultati nella classificazione di immagini rappresentanti segnali stradali [31], nella segmentazione di immagini biologiche [32], nel rilevamento dei volti [33], del testo e dei corpi umani in immagini naturali.

È importante sottolineare che, dal momento che le immagini possono essere etichettate a livello dei pixel, il *Deep Learning* rende possibile la loro applicazione in diversi ambiti tecnologici, come la progettazione di auto con *self-drive* [34]. Aziende come *Mobileye* e *NVIDIA* hanno iniziato ad utilizzare metodi basati sulle *convolutional neural networks* nei loro sistemi di visione per le auto. Altre applicazioni che stanno guadagnando importanza coinvolgono la comprensione del linguaggio naturale e il riconoscimento vocale. Nonostante questi successi le *convolutional networks* sono state in gran parte messe da parte dalle tradizionali comunità di *computer vision* e *Machine Learning* fino alla competizione *ImageNet* del 2012. In quell'occasione le reti convoluzionali, applicate ad un insieme di dati costituito da circa un milione di immagini prese dal web e che contenevano più di 1000 classi diverse, hanno ottenuto risultati straordinari, riuscendo quasi a dimezzare i tassi di errore dei migliori approcci concorrenti. Questo successo ha portato una rivoluzione nella *computer vision*; le reti convoluzionali, adesso, sono

diventate l'approccio dominante in quasi tutti i lavori di riconoscimento e di individuazione delle immagini.

Le prestazioni dei sistemi di visione artificiale basati su reti convoluzionali hanno portato le più importanti aziende tecnologiche, tra cui Google, Facebook, Microsoft, IBM, Yahoo!, Twitter e Adobe, cosi come un numero in rapida crescita di strart-up, ad avviare attività di ricerca e progetti di sviluppo per distribuire prodotti basati sul riconoscimento delle immagini. Inoltre, diverse aziende hanno iniziato a sviluppare chip specializzati [35] [36] per l'implementazione hardware di queste reti, per consentire di implementare applicazioni di visione artificiale real-time per smartphone, fotocamere, robot e macchine con self-drive.

Nel prossimo capitolo analizziamo più nel dettaglio alcune delle tecniche che sono state utilizzate in questo lavoro.

3 Metodi di Deep Learning

In questo capitolo vengono introdotti e descritti i metodi di *Deep Learning* che sono stati utilizzati nel lavoro oggetto della tesi per l'analisi del traffico di rete. Nel prossimo paragrafo si descrivono le due tipologie di architetture discriminative che vengono prese in esame al fine di identificare i protocolli e le applicazioni che hanno generato i flussi di rete. In particolar modo viene descritto il funzionamento di una *Deep Neural Network* e di una *Deep Convolutional Neural Network*. Inoltre, in modo da dare una visione più ampia dei metodi di *Deep Learning*, viene analizzata una famiglia di reti che non verrà utilizzata in questo lavoro ma che appartiene alle architetture generative, e che consiste nelle *Deep Belief Neural Networks*.

3.1 Architetture Discriminative

3.1.1 **Deep Neural Network**

Le reti neurali sono modelli discriminativi rappresentati come grafi orientati in cui ogni nodo (neurone) è composto da una funzione di attivazione e da un bias, e in cui ad ogni collegamento viene assegnato un peso. I neuroni in una rete neurale di tipo *feed-forward* (Figura 6) vengono raggruppati in livelli che possono avere connessioni entranti soltanto con i neuroni del livello precedente e connessioni uscenti con il livello superiore. Inoltre, non sono permesse connessioni tra i neuroni dello stesso livello. Dunque, la struttura orientata del grafo comporta che in una rete neurale classica il flusso di informazione proceda dal livello più basso al livello più alto senza tornare indietro.

La struttura e il numero di livelli rappresentano un aspetto fondamentale in questo tipo di reti per la loro abilità di estrarre particolari *features* a partire dai dati in ingresso in modo da ottenere una rappresentazione più complessa e significativa. Infatti, partendo dal primo strato che prende il nome di strato visibile, è possibile attivare un numero di neuroni nel livello successivo (*hidden layer*) che catturano *features* sempre più complesse man mano che si avvicina allo strato finale, aumentando, strato dopo strato, il livello di astrazione.

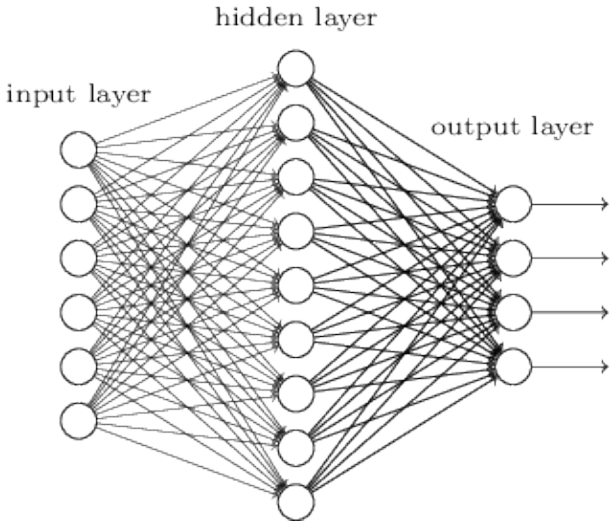

Figura 6 Deep Neural Network

Ogni neurone in una rete neurale è composto da una funzione di attivazione che deve essere calcolata per produrre il valore di uscita, secondo la formula:

$$y_j = f\left(\sum_i W_{ij}x_i + b_j\right)$$

dove j rappresenta l'indice del neurone all'interno del livello che si sta considerando, i rappresenta l'indice di un neurone del livello precedente, x_i è il valore di attivazione di ogni neurone i, W_{ij} e b_j sono, rispettivamente, i pesi di ogni collegamento che unisce il neurone i al neurone j e il valore di bias del neurone j, ed infine $f()$ è una funzione di attivazione non lineare. In problemi di classificazione il livello di output rappresenta la classe alla quale alla rete neurale mappa l'input corrente. L'uscita della rete viene calcolata utilizzando i valori di attivazione dei neuroni presenti in questo livello e viene modellata tramite una rappresentazione *one-hot,* in cui il numero di neuroni dello strato precedente combacia col numero di classi da discriminare. Tra questi neuroni viene attivato soltanto quello con la cui funzione di attivazione restituisce il valore più alto.

Per addestrare il modello viene definita una funzione di errore che, per ogni campione che viene processato, permette di dare una misura della distanza tra il valore di uscita predetto dalla rete e il valore di uscita desiderato. Durante la fase di addestramento, la rete modifica i propri parametri interni in modo da ridurre il valore della funzione di errore. Per aggiornare correttamente il valore dei pesi si utilizza l'algoritmo di discesa lungo il gradiente. L'algoritmo calcola il gradiente del vettore dei pesi, che per ognuno di essi indica di quanto aumenta o diminuisce la

funzione di errore se uno specifico peso viene incrementato di una piccola quantità. Il vettore dei pesi viene così aggiornato nella direzione opposta al gradiente. La variante più utilizzata dell'algoritmo prende il nome di *stochastic gradient descent* (SGD), che consiste nel mostrare alla rete i campioni un piccolo insieme alla volta, calcolare i valori di output e la funzione di errore, calcolare il valore medio del gradiente per ogni gruppo di campioni e aggiornare i pesi di conseguenza. Il processo viene ripetuto per molti piccoli gruppi di campioni dell'insieme di addestramento fin quando il valore della funzione di errore non smette di decrescere. Viene chiamato stocastico in quanto ogni gruppo di campioni di addestramento permette di dare una stima approssimata del valore del gradiente medio su tutti i campioni dell'insieme di addestramento. Questa procedura trova sorprendentemente un buon insieme di pesi che permette di far convergere la funzione di errore in maniera rapida se confrontata a tecniche di ottimizzazione più elaborate. Dopo la fase di addestramento, le prestazioni della rete vengono misurate su un insieme differente di campioni che fanno parte dell'insieme di test, per misurare la capacità di generalizzazione.

In generale, esistono diverse implementazioni di reti neurali che differiscono tra loro, ad esempio, per il tipo di connettività tra due strati consecutivi, per le funzioni di attivazione utilizzate o per i livelli da cui sono composte. In questo lavoro ci riferiremo alla rete appena analizzata, conosciuta in molti casi come *Multi-Layer Perceptron*, come a una generica *Deep Nerual Network*, per differenziarla dalle *Deep Convolutional Neural Networks* che verranno adesso analizzate.

3.1.2 **Deep Convolutional Neural Network**

Le *Deep Convolutional Neural Networks* rappresentano uno specifico tipo di reti neurali progettate, in particolar modo, per l'elaborazione di dati sotto forma di array multi dimensionali, ad esempio immagini a colori costituite da tre array 2-dimensionali contenti il valore di intensità di ogni pixel sui tre canali di colore. Rispetto alle *Deep Neural Networks*, permettono di codificare e estrarre un insieme di caratteristiche dall'insieme di dati sui quali vengono addestrate riducendo sensibilmente il numero di parametri della rete. Inoltre, i neuroni che compongono i vari layer sono organizzati su 3 dimensioni: larghezza (*width*), altezza (*height*) e profondità (*depth*). L'architettura di una tipica rete convoluzionale è strutturata in diversi stadi, come mostrato in Figura 7. I primi stadi sono composti da due tipi di livelli: livelli convoluzionali e livelli di pooling.

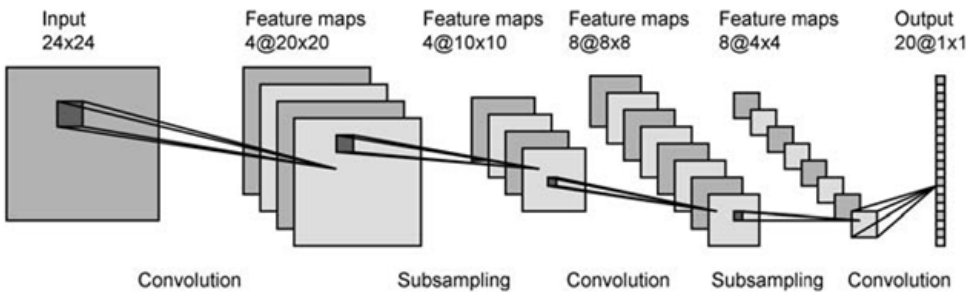

Figura 7 Deep Convolutional Neural Networks

Le unità base di un livello convoluzionale non vengono più raggruppate come un tipico strato di neuroni, ma vengono organizzate in *feature maps*. Per mezzo di queste, ogni unità è connessa solamente a una piccola porzione (*patch*) delle *features maps* del livello precedente, tramite un insieme di pesi che prendono il nome di banco di filtri (*filter banks*). Il risultato ottenuto da questa somma pesata locale viene successivamente elaborato tramite una funzione di non linearità, tipicamente una funzione di *ReLU*. Tutte le unità facenti parte di una *feature map* condividono lo stesso banco di filtri, mentre le diverse *feature maps* appartenenti a uno stesso livello utilizzano differenti banchi di filtri.

La scelta di questa architettura presenta un duplice vantaggio. Per prima cosa, in un array contenente dei dati, come ad esempio le immagini, alcuni gruppi di valori locali sono spesso fortemente correlati tra loro, formando pattern locali distintivi che possono facilmente essere identificati. Inoltre, alcune statistiche locali delle immagini, o in generale di qualsiasi altro tipo di dato, risultano non dipendere dalla posizione in cui vengono applicate. In altre parole, se un *pattern* è presente in una determinata regione dell'immagine, potrebbe anche essere presente in una qualsiasi altra posizione, da qui il motivo per cui diverse unità che operano su porzioni differenti dell'input condividono gli stessi pesi per tentare di trovare lo stesso pattern in diverse parti dell'array.

Sebbene il ruolo del livello convoluzionale sia quello di ricercare associazioni di *features* locali nel livello precedente, il ruolo del livello di pooling è quello di unire caratteristiche simili in un'unica *feature* finale. Solitamente una *convolutional neural network* è composta da due o tre strati di livelli convoluzionali e di pooling con a seguire uno o più livelli fully-connected [37]. L'addestramento di questi reti avviene, come per le normali rete neurali, attraverso una tecnica di discesa lungo il gradiente che sistema il valore dei banchi di filtri tramite minimizzazione di una funzione di errore.

Di seguito analizziamo nel dettaglio il funzionamento e la struttura dei diversi livelli che compongono una CNN.

3.1.2.1 Convolutional Layer

Il livello convoluzionale rappresenta il blocco principale di una CNN ed esegue la maggior parte del calcolo computazionale. I parametri che la rete deve apprendere a questo livello sono rappresentati da gruppi di unità che prendono il nome di banco di filtri. Ogni gruppo di filtri viene inizializzato su due dimensioni (larghezza e altezza) ma si estende lungo tutta la profondità del volume di input. Ad esempio, se applicato per il riconoscimento di immagini a colori, un tipico filtro per il primo livello convoluzionale ha dimensione 5x5x3. Si applica il banco di filtri su tutto il volume di input ed ad ogni passo si effettua il prodotto puntuale tra la matrice dei pesi e la *patch* che si sta considerando. Questa operazione, da un punto di vista matematico, consiste nell'applicare l'operatore di convoluzione tra la matrice rappresentante il volume di input e il banco di filtri. Il risultato di questa operazione consiste nella creazione di una mappa di attivazione 2D, ovvero in una *feature map*, per ogni filtro distinto. L'unione di tutte le *featurs maps* 2D lungo la dimensione di profondità produce il volume d'output specifico di quel livello convoluzionale.

La scelta delle dimensioni dei parametri che caratterizzano questo livello dipende da due concetti importanti*: local connectivity* e disposizione spaziale.

Local Connectivity. Quando si trattano input di grandi dimensioni, come le immagini, risulta impraticabile collegare ogni neurone a tutti i neuroni del volume precedente. Per ovviare a questo problema, in una rete convoluzionale, ogni neurone viene connesso soltanto ad una regione locale del volume di input, come mostrato in Figura 8. L'estensione spaziale della connettività rappresenta un iperparametro della rete, chiamato *receptive field* del neurone, e trova il suo corrispettivo nella dimensione del filtro. L'estensione della connettività lungo la dimensione di profondità, invece, coincide proprio con la profondità del volume di input (nel caso di immagini a colori esso ha valore 3). Le connessioni tra i neuroni sono quindi locali nello spazio, in termini di altezza e larghezza, ma complete lungo la dimensione di profondità

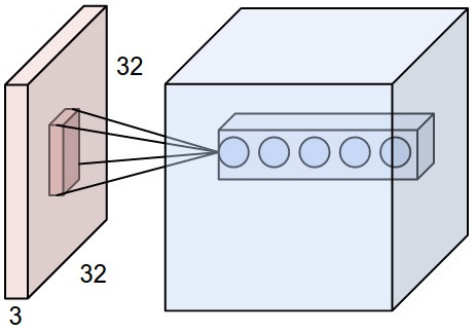

Figura 8 Local Connectivity

Disposizione spaziale. Esistono tre iperparametri che controllano la dimensione del volume di uscita e che quindi controllano il numero di neuroni da utilizzare e il modo in cui devono disporsi:

- La *profondità* del volume di output, che indica il numero di filtri che si vogliono utilizzare, ognuno dei quali apprende una caratteristica differente dal volume di input.
- Lo *stride,* che rappresenta il passo con cui si fa scorrere un filtro sul volume di input. Se posto a 1 il filtro si muove di un elemento alla volta, se assume valori maggiori di 1, il volume di uscita assume una dimensione spaziale ridotta.
- *Lo zero padding,* che permette di controllare la dimensione spaziale del volume di uscita.

3.1.2.2 ReLU layer

Rappresenta il livello in cui viene applicata la funzione di attivazione *ReLU*. Normalmente questa funzione viene applicata direttamente all'uscita del livello convoluzionale e per tale motivo quando parliamo di livello convoluzionale in realtà ci riferiamo a un'operazione di convoluzione seguita dall'applicazione della funzione di attivazione.

ReLU è l'abbreviazione di *Rectified Linear Units* e rappresenta una funzione di attivazione non lineare, definita come:

$$f(x) = \max(0, x)$$

E' stato dimostrato che l'utilizzo di questa funzione, invece di altre come la tangente iperbolica e la funzione sigmoide (Figura 9), aumenta il comportamento

discriminativo in una rete convoluzionale ed, inoltre, rende l'apprendimento di questo tipo di reti molto più rapido [38]. La sua applicazione, con il vincolo a zero, crea sparsità nel modello, infatti soltanto un piccolo insieme di neuroni viene attivato mentre i restanti vengono posti a zero.

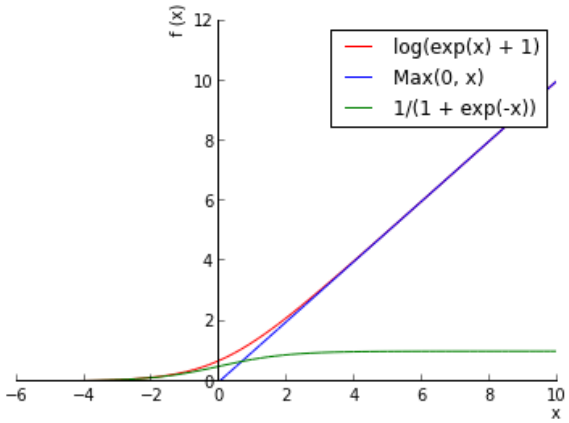

Figura 9 Confronto tra funzioni di attivazione

3.1.2.3 Max pooling layer

Un'operazione di *max pooling* è semplicemente definita come l'applicazione di una funzione di massimo su una finestra di input (Figura 10 e Figura 11). Questo livello viene applicato all'uscita del livello convoluzionale e ha la funzione di ridurre progressivamente la dimensione spaziale del volume di input per diminuire, di conseguenza, il numero di parametri e di operazioni all'interno della CNN e controllare l'overfitting.

Il *pooling layer* opera indipendentemente su ogni uscita proveniente dal livello precedente e solitamente consiste nell'applicazione di filtri di massimo di dimensione 2x2 ed un passo (*stride*) di 2, in modo da dimezzare ad ogni strato la dimensione del volume.

Come per il livello di ReLU, anche questo livello consiste di una funzione già determinata e non contiene al suo interno parametri che la rete deve apprendere.

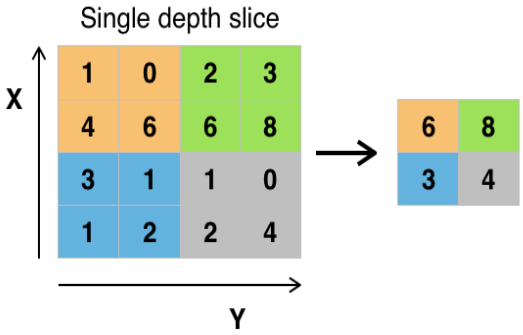

Figura 10 Max pooling 2D

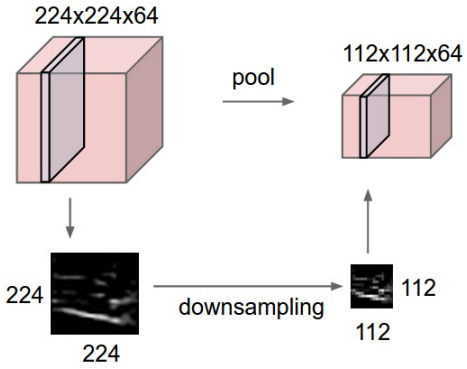

Figura 11 Max pooling 3D

3.1.2.4 Fully-connected layer

I livelli appena descritti possono essere ripetuti diverse volte ma devono sempre comparire nello stesso ordine. A questi seguono uno o più livelli completamente connessi che rappresentano gli ordinari livelli di una di una *Deep Neural Network*, in cui ogni neurone è collegato a tutti i neuroni del livello precedente. Vengono inseriti al termine della struttura della rete convoluzionale proprio perché le applicazioni delle funzioni del livello di *max pooling* permettono di avere in ingresso un volume di dimensioni notevolmente ridotte e permettono di trovare una correlazione tra i dati ad un livello di astrazione molto più alto.

3.1.2.5 Output layer

L'ultimo livello di una rete convoluzionale rappresenta l'uscita finale. Per problemi di classificazione è utile ottenere la probabilità condizionata per ogni classe, tramite la tecnica di *Softmax Regression* [39]. Tale tecnica è una forma più generale di *logistic regression*, che viene spesso utilizzata in ambito di classificazione multi classe dove le classi sono mutuamente esclusive. Per ottenere il valore di probabilità di una certa classe *j*, si utilizza la seguente funzione di softmax, che prende il nome di *normalized exponential*:

$$P(Y = j|x) = softmax_c(x) = \frac{e^{\sum_j W_{ij} x_j + b_i}}{\sum_k e^{W_{kj} x_j + b_k}}$$

Per l'addestramento della rete, invece, la funzione di costo adoperata dalla *Softmax Regression*, nota come *cross entropy*, è definita come:

$$H_{y'}(y) = -\sum_i y'_i \log(y'_i)$$

dove la somma viene calcolata sull'insieme di addestramento prendendo n campioni.

3.2 Architetture generative

Con architetture generative si intendono tutti quei modelli di *Deep Learning* con apprendimento non supervisionato. Queste architetture risultano essere molto interessanti in quanto sono in grado di apprendere una distribuzione di probabilità a partire da un insieme finito di dati e di generare nuovi campioni a partire da essa. I modelli generativi, spesso, possono essere rappresentati tramite dei grafi in cui i nodi rappresentano delle variabili casuali e gli archi forniscono informazioni sul tipo di dipendenza tra le variabili che vengono prese in considerazione. La distribuzione di probabilità congiunta tra tutte le variabili in gioco può essere calcolata in termini di prodotti tra un nodo e tutti i suoi vicini nel grafo. Tra i modelli generativi esiste una famiglia che prende il nome di modelli basati sull'energia (*energy-based models*) che associano ad ogni configurazione di variabili un certo valore di energia. A configurazioni con un alto valore di probabilità il modello cerca di associare un basso valore di energia e, viceversa, a configurazioni con bassi valori di probabilità associare un alto valore di energia. Di solito per aumentare il potere espressivo del modello si aggiungono un insieme di variabili nascoste, *h,* e si cerca di calcolare la probabilità congiunta tra le variabili osservate, ovvero quelle che si vogliono studiare, e queste variabili fittizie.

Sebbene il calcolo della distribuzione di probabilità congiunta risulti impraticabile per modelli in cui il grafo rappresentativo contiene dei collegamenti tra i nodi dello stesso livello, esiste una famiglia di modelli generativi, chiamati *Restricted Boltzmann Machines*, che rendono questo calcolo più semplice.

3.2.1 **Restricted Boltzman Machine**

Le Macchine di *Boltzman* (*BMs*), Figura 12, appartengono ai modelli basati sull'energia e sono costituite da due tipi di variabili: variabili osservate, *x*, e variabili nascoste, *h.* Vista la loro struttura a grafo, esse hanno la proprietà di avere la funzione di energia esprimibile secondo l'equazione:

$$E(x,h) = -b'x - c'h - h'Wx - x'Ux - h'Vh$$

dove le matrici U e V vengono assunte simmetriche. A causa del termine quadratico in h, il calcolo analitico per questo tipo di modelli risulta essere non praticabile.

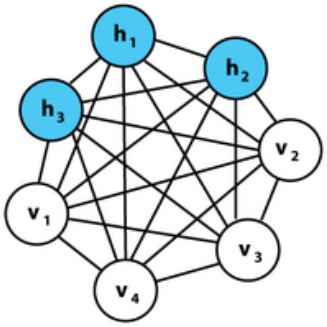

Figura 12 Boltzmann Machine

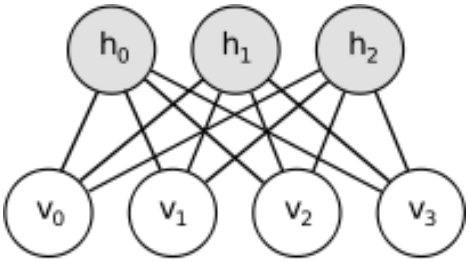

Figura 13 Restricted Boltzmann Machine

Le *Restricted Bolttzman Machines*, invece, sono un particolare tipo di macchine di *Boltzman* in cui non sono presenti connessioni tra neuroni dello stesso livello, come mostrato in Figura 13. Il vincolo di non avere connessioni di tipo *visibile-visibile* e *hidden-hidden* permette di rappresentare questi modelli come grafi completi bipartiti. La funzione di energia diventa bilineare e assume la forma:

$$E(x, h) = -bx - ch - h'Wx$$

dove W rappresenta la matrice dei pesi che collegano le unità nascoste a quelle visibili, mentre b e c rappresentano gli offset, rispettivamente, delle unità visibili e delle unità nascoste.

Con questi vincoli è possibile definire la funzione di energia libera, caratterizzante ogni modello basato sull'energia, nel modo seguente:

$$\mathcal{F}(v) = -bx - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i x)}$$

Grazie alla specifica struttura delle *RBMs*, inoltre, le unità visibili e le unità nascoste sono condizionatamente indipendenti data una terza. Utilizzando questa proprietà è possibile scrivere:

$$p(h \mid x) = \prod_i p(h_i|x)$$

$$p(x \mid h) = \prod_j p(x_j|h)$$

Le *RBMs* risultano particolarmente interessanti perché possono essere utilizzate in entrambi i versi, o a partire dalle unità visibili o a partire dalle unità nascoste.

Nei casi più comunemente studiati in cui si utilizzano unità binarie (dove x_j e $h_i \in \{0,1\}$), otteniamo una versione probabilistica della usuale funzione di attivazione di un neurone:

$$P(h_i = 1 \mid v) = sigm(c_i + W_i x)$$
$$P(x_j = 1 \mid h) = sigm(b_j + W_j' h)$$

La formulazione dell'energia libera di una *RBM* con unità binarie si semplifica:

$$\mathcal{F}(v) = -bv - \sum_i \log(1 + e^{(c_i + W_i v)})$$

Proprio a partire da questa formula si può monitorare il comportamento di una RBM durante l'addestramento. Se le unità non sono binarie, occorre scegliere una distribuzione di probabilità che più si adatta ai dati (ad esempio una distribuzione gaussiana o una distribuzione esponenziale) e modificare la formula corrispettiva oltre che il processo di campionamento che viene descritto di seguito.

La funzione principale di una RBM è quella di trovare i campioni di una certa distribuzione *p(x)*, non conosciuta a priori, a partire dai campioni presenti nell'insieme di addestramento, in modo da rappresentare con buona approssimazione la funzione di probabilità congiunta *p(x,h)* che meglio descrive i legami tra le unità visibili e le unità nascoste. Questi campioni si ottengono eseguendo una catena di *Markov* fino alla sua convergenza, utilizzando un metodo, conosciuto come campionamento di *Gibbs*, come operatore di transizione.

Il campionamento di *Gibbs* per una funzione di probabilità con *N* variabili aleatorie random, $S = (S_1, \cdots, S_N)$, si ottiene attraverso una sequenza di *N* sotto step di campionamento della forma $S_i \sim p(S_i \mid S_{-i})$, dove S_{-i} rappresenta le *N-1* variabili aleatorie di *S* escludendo S_i. In una *RBM*, *S* rappresenta l'insieme delle unità visibili e delle unità nascoste. Dal momento che per tale modello esse risultano essere condizionatamente indipendenti, il processo di campionamento avviene simultaneamente. Le unità visibili vengono campionate fissati i valori delle unità nascoste e, viceversa, le unità nascoste vengono campionate fissando i nuovi valori calcolati per le unità visibili. In una catena di *Markov*, un singolo step di campionamento di *Gibbs* corrisponde all'esecuzione consecutiva delle seguenti formule, in cui si è scelta una sigmoide come funzione di attivazione:

$$h^{(n+1)} \sim sigm(W' x^{(n)} + c)$$
$$x^{(n+1)} \sim sigm(W' h^{(n+1)} + b)$$

In Figura 14 viene rappresentato graficamente il processo.

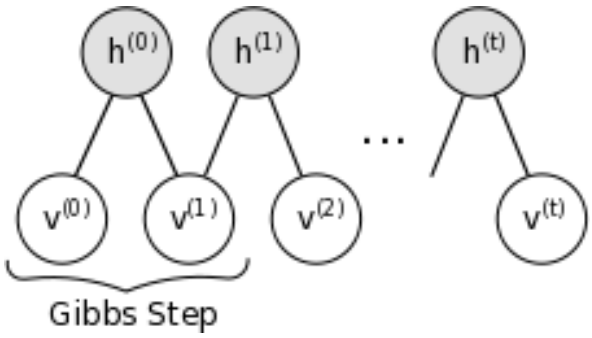

Figura 14 Campionamento di Gibbs

Per $t \to \infty$, è garantito che i campioni $(v^{(t)}, h^{(t)})$, riescano a descrivere in modo accurato la distribuzione di probabilità *p(x,h)*. In linea teorica il processo di convergenza richiede di applicare un campionamento di *Gibbs* un numero elevato di volte per garantire la convergenza, tuttavia, l'algoritmo di *Contrastive Divergence* garantisce la convergenza in numero finito di step. Tale algoritmo consiste di due trucchi per accelerare il processo di campionamento:

- La catena di *Markov* viene inizializzata con i campioni dell'insieme di addestramento.
- Non si attende che la catena converga, i campioni si ottengono soltanto dopo *k-step* si campionamento di *Gibbs*. Nella pratica, scegliere un valore di *k* uguale ad 1 o a 10 rende il processo sorprendentemente accurato.

3.2.2 **Deep Belief Neural Networks**

Una *Restricted Boltzmann Machine* rappresenta il blocco principale di una *Deep Belief Neural Network (DBN)*, in quanto condivide i propri parametri con ogni suo layer.

Le *DBNs* (Figura 15) rappresentano modelli grafici, particolarmente utilizzati dopo il 2006 [18], che apprendono ad estrarre in maniera non supervisionata una rappresentazione gerarchica dei dati facenti parte di un insieme di addestramento. L'obiettivo è quello di modellare la distribuzione congiunta tra il vettore *x* dei dati in ingresso e i vettori h^k delle unità nascoste distribuite sui diversi strati:

$$P(x, h^1, \cdots, h^l) = (\prod_{k=0}^{l-2} P(h^k | h^{k+1}))P(h^{l-1}, h^l)$$

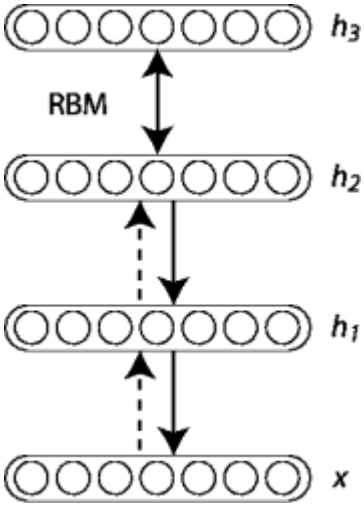

Figura 15 Deep Belief Neural Network

Nella formula precedente x rappresenta l'insieme dei dati in ingresso e si assume che $x = h^0$, $P(h^{k-1}, h^k)$ rappresenta la distribuzione di probabilità delle unità visibili condizionata dalle unità nascoste al livello k e, infine, $P(h^{l-1}, h^l)$ rappresenta la probabilità congiunta tra le unità visibili e le unità nascoste dell'ultimo strato *RBM*, come mostrato in Figura 15. In generale le *DBN* prevedono due fasi distinte:

- Una fase di *pre-training* in cui si applica un algoritmo di addestramento non supervisionato ad una *RBM* alla volta in maniera sequenziale.
- Una fase di raffinamento, detta *fine-tuning*, che consiste in un addestramento supervisionato, volto alla classificazione, e che quindi prevede che tutti i dati di addestramento siano stati già classificati precedentemente. Questa fase è detta di raffinamento in quanto l'addestramento parte dalla configurazione di pesi che sono stati precedentemente calcolati nella fase di *pre-training*.

L'algoritmo di addestramento per questo modello, come spiegato da Hinton [18], inizia con la creazione di una prima *RBM* per addestrare i pesi che collegano il primo strato visibile al primo strato nascosto, assumendo che la matrice dei pesi per i restanti livelli dell'intera rete mantenga gli stessi valori. Una volta appresi, tali pesi vengono utilizzati per campionare le unità visibili del primo livello in modo da creare una nuova rappresentazione per le unità nascoste del primo *hidden layer*. Dopo l'inizializzazione di questo strato viene creata una seconda *RBM* che viene addestrata allo stesso modo e la procedura viene ripetuta ricorsivamente per tutti gli strati della

rete, ad eccezione dell'ultimo. La fase di pre-addestramento porta la rete a trovare una matrice dei pesi che si presume riesca a dare una rappresentazione più generale dei dati che vengono trattati. Successivamente, viene applicata una procedura di addestramento supervisionato che prevede un insieme di addestramento già classificato e che a partire dal nuovo stato della rete mira a trovare una nuova configurazione della matrice dei pesi tale da consentire un processo di classificazione più raffinato. In questa fase si utilizzano algoritmi di addestramento già visti per i modelli discriminativi.

4 Sistema di analisi del traffico di rete

In questo capitolo vengono descritti i metodi di *Deep Learning* che sono stati addestrati per il lavoro oggetto della tesi e le tecniche che sono state utilizzate per l'estrazione e l'elaborazione delle tracce di traffico di rete da classificare. Nella prima parte viene descritta la struttura dei due metodi di *Deep Learning* utilizzati, presentando il numero di *layers*, di neuroni e i parametri di apprendimento. Nella seconda parte vengono descritte le tecniche per l'elaborazione delle tracce di rete. Viene presentato il dataset contenente la raccolta di pacchetti di rete che si è scelto di utilizzare e il processo di *parsing* che ha permesso di elaborare i dati presenti nel dataset in modo da estrarre i payload TCP e salvarli in un database. Successivamente, viene descritto il vettore di *features* che è stato estratto da ogni flusso di rete per ridurne la dimensione ed, infine, viene presentato il *framework* utilizzato per l'implementazione dei metodi di *Deep Learning*.

4.1 Metodi basati su Deep Learning

*** OMISSIS ***

4.2 Implementazione

4.2.1 **I dati**

I metodi di *Deep Learning* sono stati applicati ai dati presenti nel dataset pubblico utilizzato nel lavoro di Bujlow, Carela-Espanol e Barlet-Ros [15] e disponibile presso il sito web [40]. Questo dataset, a differenza di altri utilizzati e resi disponibili dalla comunità scientifica, contiene una raccolta di pacchetti IP che permette di ricostruire le intere trasmissioni ed, inoltre, fornisce il contenuto del payload completo ed in chiaro. La maggior parte dei dataset pubblici utilizzati e resi disponibili per testare le tecniche di classificazione del traffico di rete, infatti, non forniscono il contenuto completo di ogni pacchetto che, per motivi di privacy, viene troncato o cifrato, rendendo di fatto difficile un confronto reale tra le diverse

tecniche. Ad esempio, la raccolta di tracce di traffico di Internet rese pubbliche in MAWI [41], un *repository* che periodicamente rende disponibili diversi flussi di traffico, viene pubblicata troncando il payload contenuto nei pacchetti IP, rendendone disponibili soltanto i primi 96 byte.

Ogni flusso presente nel dataset, che prende il nome di *COMNET14*, è stato propriamente contrassegnato in modo da darne una descrizione. In particolare, ad ogni flusso è stata associata un'etichetta per permetterne la classificazione su tre livelli distinti di precisione: protocollo applicativo, applicazione e web service. Ad ogni flusso possono essere associate diverse *labels*, mentre flussi che non hanno associata una label non vengono presi in considerazione quando vengono estratti dal file PCAP.

Protocollo applicativo	**Megabytes**
DNS	7.66
HTTP	7325.44
ICMP	2.34
IMAP-STARTTLS	36.56
IMAP-TLS	410.39
NETBIOS Name Service	11.13
NETBIOS Session Service	0.01
SAMBA Session Service	450.39
NTP	6.12
POP3-PLAIN	189.25
POP3-TLS	147.68
RTMP	2353.67
SMTP-PLAIN	62.27
SMTP-TLS	3.37
SOCKSv5	898.31
SSH	844.87
Webdav	59.91

Tabella 3 Protocolli applicativi dataset

Applicazione	**Megabytes**
4Shared	13.39
America's Army	61.15
BitTorrent clients (encrypted)	3313.98
BitTorrent clients (non-encrypted)	6779.95
Dropbox	128.66
eDonkey clients (obfuscated)	8178.74
eDonkey clients (non-obfuscated	8480.48
FreeNet	538.28
FTP clients (active)	341.17
FTP clients (passive)	270.46
iTunes	75.4
League of Legends	124.14
Pando Media Booster	13.3
PPlive	83.86
PPStream	390.4
RDP clients	13257.65
Skype (all)	102.99
Skype (audio)	4.85
Skype (file transfer)	25.74
Skype (video)	41.16
Sopcast	109.34
Spotify	195.15
Steam	255.84
TOR	47.14
World of Warcraft	1.98

Tabella 4 Applicazioni nel dataset

Per l'assegnazione corretta dei nomi delle applicazioni, gli autori hanno utilizzato un *Volunteer-Based System* (VBS), sviluppato dall'Università di Aalborg [42]. L'obiettivo del progetto VBS è quello di collezionare informazioni sui flussi di traffico provenienti da Internet insieme alle informazioni dettagliate specifiche di ogni pacchetto. Ad esempio, per quanto concerne i flussi di traffico, vengono raccolte informazioni come l'istante iniziale di una trasmissione, il numero di pacchetti inviati, gli indirizzi IP del mittente e del destinatario, le porte locali e remote e il protocollo utilizzato a livello di trasporto. Per ogni pacchetto appartenente a una determinata trasmissione, invece, vengono raccolte informazioni sulla direzione di trasmissione, dimensione del payload, *flag* TCP e il timestamp. Per ogni flusso, inoltre, il sistema VBS ottiene il nome del processo che lo ha generato grazie ai vari sistemi di *sockets* in cui è stato installato.

Il dataset è stato creato per studiare le prestazioni di sei software di DPI su tre livelli di precisione ed è composto da 767690 trasmissioni, per un ammontare di 53.31 GB. Viene fornito insieme ad un file informativo che provvede a dare, per ogni flusso, informazioni sul nome dell'applicazione, il tempo iniziale e finale della trasmissione e le *5-tuple* (porta mittente, indirizzo IP mittente, porta destinatario, indirizzo IP destinatario, protocollo di trasporto). Il nome dell'applicazione viene fornito per il 98,96% dei dati, mentre i restanti dati risultano non etichettati in quanto appartenenti a trasmissioni in cui il sistema di VBS non è stato in grado ricavare il nome del processo. I protocolli applicativi insieme al numero di flussi e le dimensioni sono fornite nella Tabella 3, mentre applicazioni sono mostrate in Tabella 4.

4.2.2 **Parsing**

L'insieme di tracce, contenute nel dataset *COMNET14*, viene reso disponibile in formato PCAP. I file PCAP (*Packet-CAPture*) nascono nel campo del *networking* per poter memorizzare il traffico di rete catturato da specifici programmi. Nei sistemi operativi basati su Unix, *pcap* viene implementato grazie alla libreria *libpcap* [43], mentre nei sistemi Windows viene implementato grazie a *WinPcap* [44]. L'utilizzo di questo software è fondamentale per catturare il traffico di rete e procedere ad una successiva fase di analisi.

*** OMISSIS ***

5 Risultati Sperimentali

*** OMISSIS ***

6 Conclusioni

In questo lavoro sono state progettate e sviluppate due tecniche basate su *Deep Learning* per studiare le loro prestazioni nell'ambito della classificazione del traffico di rete. I risultati sperimentali sono stati ottenuti applicando tali tecniche su una raccolta di flussi di rete messa a disposizione dalla comunità scientifica. Ai fini della progettazione dei due modelli, si è utilizzato un descrittore per estrarre un vettore di *features* e permettere la creazione di un insieme di addestramento i cui campioni avessero stesse dimensioni. Inoltre, per una valutazione più accurata, i risultati sono stati confrontati con quelli ottenuti dall'applicazione di un albero di decisione, la tecnica di *Machine Learning* che in molti lavori di classificazione del traffico di Internet ha presentato il comportamento migliore.

*** OMISSIS ***

Riferimenti Bibliografici

[1] A. Callado, C. Kamienski, G. Szabó e B. P. Geró, «A Survey on,» IEEE Communications Surveys & Tutorials, 2009.

[2] «IANA,» [Online]. Available: http://www.iana.org/.

[3] Y. Y. Yan Liu, «Analysys of P2P Traffic Identification Methods,» Journal of Emerging Trends in Computing and Information Sciences, vol. 4, n. 5, pp. 490-493, 2013.

[4] CISCO, «CISCO WAN and Application Optimization Solution Guide,» [Online].Available:http://www.cisco.com/c/en/us/td/docs/nsite/enterprise/wan/wan_optimization/wan_opt_sg.html.

[5] B. Silver, «Netman: A learning network traffic controller,» Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, vol. 2, pp. 923-931, 1990.

[6] T. Nguyen e G. Armitage, «A survey techniques for internet traffic classification using machine learning,» IEEE Communications Surveys and Tutorials, vol. 10, n. 4, 2008.

[7] T. Karagiannis, K. Papagiannaki, N. Taft e M. Faloutsos, «Profiling the End Host,» Passive and active network measurement, 2007.

[8] K. Xu e Z, «Profiling Internet Backbone Traffic: Behavior Models and Applications,» 2009.

[9] T. Mori, R. Kawahara, H. Hasegawa e S. Shimogawa, «Characterizing traffic flows originating from large-scale video sharing services,» Proceedings of Traffic Monitoring and Analysis, pp. 17-31, 2010.

[10] S. Gaglio, G. L. Re e M. Morana, «A framework for real-time Twitter data analysis,» in Journal of Computer Communications, Elsevier, ISSN 0140-3664.

[11] S. Gaglio, G. L. Re e M. Morana, «Real-Time Detection of Twitter Social Events from the User's Perspective,» in Proceedings of the 2015 IEEE

International Conference on Communications (ICC2015), 2015.

[12] S. Yoon, J. Park, J. Park, Y. Oh e M. Kim, «Internet application traffic classification using fixed IP-port,» Manag. Enabling the Future Internet for Changing Bus. and New Comput. Serv., vol. 5787, pp. 21-30, 2009.

[13] J. Erman, A. Mahanti e M. Arlitt, «Internet traffic classification using machine learning,» IEEE GLOBCOM, 2006.

[14] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloustos e H. Lee, «Internet traffic classification demystified: myths, caveauts and the best practices,» ACM CoNEXT, 2008.

[15] T. Bujlow, V. Carela-Espanol e P. Barlet-Ros, «Indipendent Comparison of Popular DPI Tools for Traffic Classification,» Computer Networks: The International Journal of Computer and Telecommunications Networking, pp. 75-89, 2015.

[16] N. Williams, S. Zander e G. Armitage, «A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification,» ACM SIGCOMM Comput. Commun. Rev., vol. 36, n. 5, 2006.

[17] «NLANR traces,» [Online]. Available: http://pma.nlanr.net/Special/.

[18] G. E. Hinton, S. Osindero e Y.-W. Teh, «A Fast Learning Algorithm for Deep Belief Nets,» Neural Computation, 2006.

[19] A. Krizhevsky, I. Sutskever e G. Hinton, «ImageNet classification with deep convolutional neural networks,» In Proc. Advances in Neural Information Processing Systems, 2012.

[20] C. Farabet, C. Couprie, L. Najman e Y. LeCun, «Learning hierarchical features for scene labelling,» IEEE Trans. Pattern Anal. Mach. Intell, 2013.

[21] T. Mikolov, A. Deoras, D. Povey, L. Burget e J. Cernocky, «Strategies for training large scale neural network language models,» In Proc. Automatic Speech Recognition and Understandig, 2011.

[22] G. Hinton, «Deep neural networks for acoustic modeling in speech recognition,» IEEE Signal Processing Magazine, 2012.

[23] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl e V. Svetnik, «Deep neural nets as a method for quantitative structure-activity relationships,» J. Chem. Inf. Model, 2015.

[24] T. Ciodaro, D. Deva, J. d. Seixas e D. Damazio, «Online particle detection with neural networks based on topological calorimetry information,» J. Phys. Conf, 2012.

[25] Kaggle, «Higgs boson machine learning challenge,» 2014. [Online]. Available: https://www.kaggle..

[26] M. Helmstaedter, «Helmstaedter, M. et al. Connectomic reconstruction of the inner plexiform layer in the mouse retina,» Nature, 2013.

[27] M. K. Leung, H. Y. Xiong, L. J. Lee e B. J. Frey, «Deep learning of the tissue regulated splicing code,» Bioinformatics, 2014.

[28] R. C. N. l. p. (. f. scratch, «Natural language processing (almost) from scratch,» 2011.

[29] S. Jean, K. Cho, R. Memisevic e Y. Bengio, «On using very large target vocabulary for neural machine translation,» 2015.

[30] J. Saxe e K. Berlin, «Deep neural network based malware detection using two dimensional binary program features,» 10th International Conference on: Malicious and Unwanted Software (MALWARE) , 2015.

[31] D. Ciresan, U. Meier, J. Masci e J. Schmidhuber, «Multi-column deep neural network for traffic sign classification,» Neural Netwoek , 2012.

[32] F. Ning, «Toward automatic phenotyping of developing embryos from videod,» IEEE Trans. Image Process, 2005.

[33] C. Garcia e M. Delakis, «Convolutional face finder: a neural architecture for fast and robust face detection,» IEEE Trans. Pattern Anal. Machine Intell, 2004.

[34] R. Hadsell, «Learning long-range vision for autonomous off-road driving,» J. Field Robot, 2009.

[35] B. Boser, E. Sackinger, J. Bromley, Y. LeCun e L. Jackel, «Ana analog neural network processor with programmable topology,» J. Solid State Circuits,

1991.

[36] C. Farabet, «Large-scale FPGA-based convolutional networks,» In Scaling up Machine Learning: Parallel and Distributed Approaches, 2011.

[37] Y. LeCun, Y. Bengio e G. Hinton, «Deep Learning,» Nature, pp. 436-444, 2015.

[38] X. Glorot, A. Bordes e Y. Bengio, «Deep sparse rectifier networks.,» In Proc. of the 14th International Conference on Artificial Intelligence and Statistics, vol. 15, pp. 315-323, 2011.

[39] S. University, «Softmax Regression,» [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/.

[40] «Traffic classification at the Universitat Politècnica de Catalunya (UPC),» 2014. [Online]. Available: http://www.cba.upc.edu/monitoring/traffic-classification.

[41] MAWI, «MAWI Working Group Traffic Archive,» [Online]. Available: http://mawi.wide.ad.jp/mawi/.

[42] T. Bujlow, K. Balachandran, S. L. N. Hald, T. Riaz e J. M. Pedersen, «Volunteer-Based System for research on Internet traffic,» TELFOR Journal, vol. 4, n. 1, pp. 2-7, 2012.

[43] «TCPDUMO & LIBPCAP,» [Online]. Available: http://www.tcpdump.org/.

[44] «WinPcap,» [Online]. Available: https://www.winpcap.org/.

[45] «WIRESHARK,» [Online]. Available: https://www.wireshark.org/.

[46] «Libreria DPKT,» [Online]. Available: https://pypi.python.org/pypi/dpkt.

[47] «SQLite3,»[Online].Available:https://docs.python.org/2/library/sqlite3.html.

[48] A. Juvonen e T. Supola, «Adaptive framework for network traffic classification using dimensionality reduction and clustering,» Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012.

[49] C. Wressnegger, G. Shwenk, D. Arp e K. Rieck, «A close look on n-grams in intrusion detection: anomaly detection vs. classification,» AISec '13 ACM

workshop on Artificial intelligence and security, 2013.

[50] K. Ingham, «Anomaly Detection for HTTP Intrusion Detection: Algorithm Comparisons the Effect of Generalization on Accuracy,» PhD thesis, Department of Computer Science, University of New Mexico, 2007.

[51] «Tensorflow,» [Online]. Available: https://www.tensorflow.org/.

[52] J. C. Prather, D. F. Lobach, L. Goodwin, J. W. Hales, M. L. Hage e W. E. Hammond, «Medical data mining: knowledge discovery in a clinical data warehouse,» AMIA Annual Fall Symposium, 1997.

[53] M. Scigocki e S. Zander, «Improving Machine Learning Network Traffic Classification with payload-based features,» 2012.

[54] J. R. Quinlan, «Induction of Decision Trees,» 1986.

[55] T. U. o. Waikato, «Weka 3: Data Mining Software in Java,» [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[56] M. Umanol, H. Okamoto, H. Tamura, F. Kawachi, S. Umedzu e J. Kinoshita, «Fuzzy decision tree by fuzzy ID3 algorithm and its application to diagnosis systems,» IEEE World Congress on Computational Intelligence, 1994.

[57] G. Di Fatta, S. Gaglio, G. Lo Re, M. Ortolani, Adaptive routing in active networks., IEEE Openarch 3, 23-24, 2000;

[58] G. Di Fatta, G. Lo Re, A. Urso, A fuzzy approach for the network congestion problem, International Conference on Computational Science, 286-295, 2002;

[59] G. Di Fatta, G. Lo Re, A. Urso, Parallel genetic algorithms for the tuning of a fuzzy AQM controller, International Conference on Computational Science and Its Applications, 417-426, 2003;

[60] A. De Paola, S. Fiduccia, S. Gaglio, L. Gatani, G. Lo Re, A. Pizzitola, M. Ortolani, P. Storniolo, A. Urso, Rule based reasoning for network management, Proceeding of the Seventh International Workshop on Computer Architecture for Machine Perception, 2005.

Indice delle figure

Indice delle Tabelle