**ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA**

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO di
INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"Guglielmo Marconi"
DEI

**CORSO DI LAUREA IN INGEGNERIA ELETTRONICA**

**TESI DI LAUREA**
**in**
**Elaborazione Dei Segnali Nei Sistemi Elettronici M**

# PIATTAFORME EDGE PER L'UTILIZZO DI RETI NEURALI PROFONDE IN AMBITO INTERNET OF THINGS

CANDIDATO
Marco Bonazzi

RELATORE
Prof. Mauro Mangia

CORRELATORI
Chiar.mo Prof. Riccardo Rovatti
Dott. Alex Marchioni

Anno Accademico 2018/2019

Sessione III

# INDICE

# 1 INTRODUZIONE

## 1.1 SENSORI, EDGE E CLOUD

Al giorno d'oggi la velocità e versatilità dei sistemi di telecomunicazione hanno permesso di connettere alla rete un imponente numero di dispositivi a cui viene data la possibilità non solo di scambiare immense quantità di dati, ma anche di beneficiare di risorse computazionali e di archiviazione notevolmente superiori a quelle disponibili nel nodo di accesso. In particolare l'opportunità di demandare compiti da un dispositivo a un altro, purché entrambi siano connessi alla rete, ha permesso lo sviluppo del paradigma conosciuto come cloud computing, modello secondo il quale un sistema utente si avvale di risorse appartenenti ad un sistema fornitore per mezzo della rete internet. Questo paradigma di calcolo distribuito implica una naturale gerarchia fra i dispositivi, distinguendoli in base alle funzionalità e alle risorse a disposizione in tre famiglie:

- Nodi sensori: sono i sistemi che solitamente dispongono di meno risorse, ma anche quelli più economici, e nel modello ricoprono il ruolo di richiedenti del servizio;
- Nodi edge: si tratta di sistemi che hanno una disponibilità di risorse di solito superiore a quella offerta dai nodi sensori e rappresentano il contatto tra questi ultimi e il cloud. Il loro compito è quello di facilitare le comunicazioni tra i due livelli, trasferendo i dati relativi a più nodi sensori e pre-elaborarli;
- Nodi cloud: sono i sistemi con la maggior quantità di risorse che ricoprono il ruolo di fornitori dei servizi richiesti dai nodi sensori, interfacciandosi ad essi mediante i nodi edge.

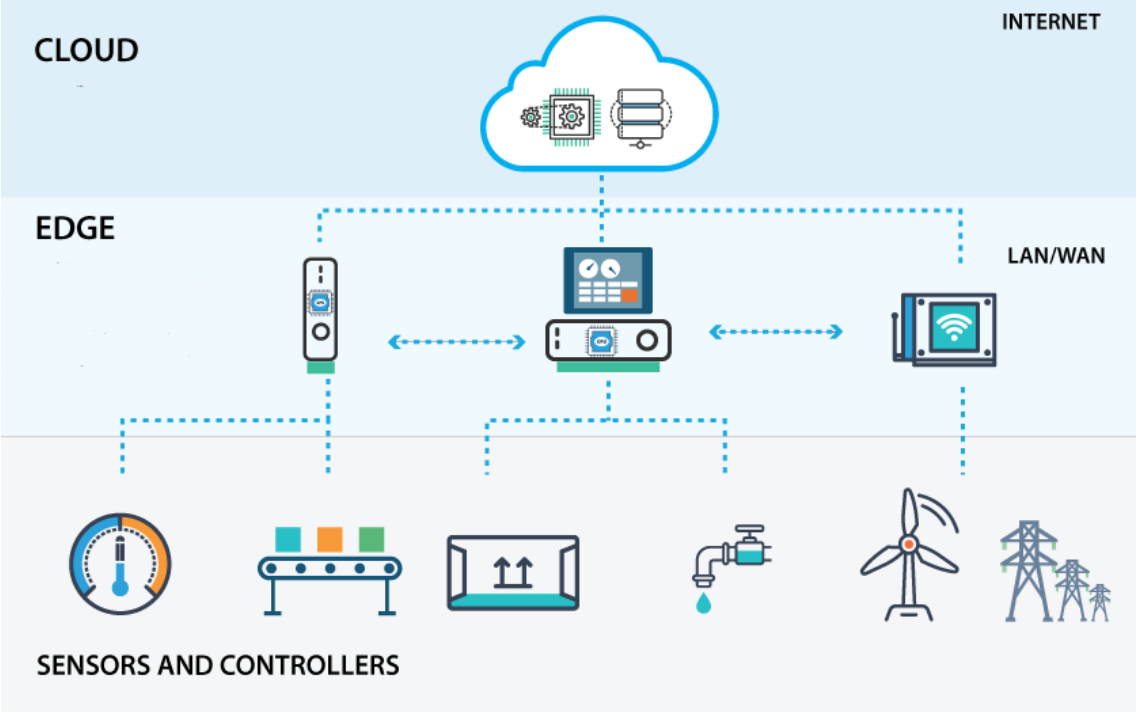



*Figura 1-1: Diagramma del cloud computing [1]*

I vantaggi offerti dal cloud computing sono numerosi e tra questi sicuramente spiccano le prestazioni che si possono raggiungere mediante questo paradigma. Infatti le risorse offerte dal cloud, sia per quanto riguarda la velocità di esecuzione dei compiti sia per quanto riguarda lo spazio di archiviazione disponibile, sono ordini di grandezza superiori rispetto quelle messe a disposizione dai dispositivi richiedenti.

A questo fatto va aggiunta la prerogativa del cloud di essere estremamente affidabile. Infatti i data center che ospitano i server del cloud sono caratterizzati da un hardware fortemente ridondante e flessibile, che permette sia di preservare in modo efficiente più copie dei dati sia di spostare i compiti da una macchina all'altra in modo totalmente invisibile all'utente finale. Questa elasticità, oltre a tutelare da eventuali malfunzionamenti dell'hardware, concede di eseguire parallelamente più compiti di più utenti diversi in modo isolato, ridimensionando secondo le necessità le risorse a disposizione per ciascun richiedente di servizi. In questo modo è possibile gestire in maniera estremamente efficiente il carico computazionale permettendo di abbattere i costi, dal momento che si trae il massimo beneficio dalle spese per l'acquisto, l'alimentazione e il mantenimento dei server.

I servizi di cloud computing si distinguono in tre categorie, differenziate a seconda del livello di completezza delle funzioni offerte:

- Infrastructure as a Service (IaaS): con questa soluzione si affitta l'intera infrastruttura che comprende la sede fisica in cui è situato il data center, le macchine virtuali, i server e la loro rete di interconnessione;
- Platform as a Service (PaaS): con questa modalità si ottiene un ambiente che integra un sistema operativo già configurato e provvisto di tutti gli strumenti necessari allo sviluppo, alla gestione e alla distribuzione di applicazioni software;
- Software as a Service (SaaS): questo metodo permette di distribuire una applicazione mediante la rete, lasciando al provider del servizio il compito della manutenzione dell'hardware e del software sottostante.

Nonostante i numerosi vantaggi che il cloud computing può offrire, con l'avvento dell'Internet of Things (IoT) e l'esponenziale aumento di dispositivi connessi alla rete, la versatilità di questo paradigma è risultata in parte limitata in diversi contesti applicativi. Infatti l'accrescimento così veloce del numero di potenziali client denota un conseguente incremento dei dati da trasferire, richiedendo uno sforzo maggiore alla rete. Questo problema risulta particolarmente spinoso in diverse applicazioni in cui la latenza tollerata è minima. Inoltre, anche nei casi in cui i vincoli temporali sono più rilassati, l'aumento di dispositivi sovraccarica la rete rendendo comunque difficoltoso concentrare l'intera elaborazione dei dati in un unico data center.

Per queste ragioni parallelamente al cloud computing si è andato a delineare un nuovo paradigma di calcolo distribuito, l'edge computing, caratterizzato dalla decentralizzazione delle risorse di calcolo avvicinandole ai nodi sensori, riducendo notevolmente le latenze e le risorse richieste alla rete. Il dislocamento del calcolo dei dati porta con sé notevoli vantaggi, soprattutto perché questo nuovo paradigma si adatta perfettamente al nuovo carico di lavoro richiesto dall'IoT.

Infatti l'edge computing, oltre a risolvere i principali problemi di latenza e congestionamento della rete, si dimostra più efficace in numerose situazioni, permettendo l'impiego di soluzioni più mirate al variare dell'ambito di applicazione. Sebbene così facendo si perda la generalità e la versatilità tipiche delle soluzioni cloud computing, la decentralizzazione del calcolo su piattaforme edge, o eventualmente sui nodi sensori, risulta essere attualmente l'unica soluzione facilmente realizzabile per affrontare in modo

efficiente lo sviluppo dell'internet of things, che porterà ad avere sempre più dispositivi connessi alla rete che produrranno una quantità di dati tale da non poter essere spostata in modo adeguato.

Questo però non significa che l'edge computing andrà a sostituire la controparte cloud. Tuttavia, mentre al primo paradigma di calcolo verranno demandati incarichi più leggeri come analisi dati basilare, aggregazione e pre-elaborazione, il secondo rimarrà focalizzato su tutte le attività che richiedono uno sforzo computazionale elevato, impossibile da dislocare sui nodi edge. Infatti l'edge computing si dimostra essere più limitato nelle risorse, rispetto al classico paradigma del cloud computing, presentando vincoli più stringenti per quanto riguarda risorse computazionali, spazio di archiviazione e consumi.

Un altro aspetto da considerare è la sicurezza dei dati. Infatti in uno scenario edge computing la maggior parte dei dati rimane all'interno del nodo edge, riducendo notevolmente le possibilità di furto di dati.

Se si considera lo scenario dell'internet of things, è fondamentale ragionare anche sul tipo di dato che questo nuovo ecosistema andrà a produrre. Infatti si prevede non solo che la mole di dati aumenterà, ma che la tipologia stessa cambierà: la quantità di dati prodotta dall'IoT si presenta più eterogenea e variegata rispetto alle classiche strutture, necessitando di un maggiore sforzo per estrarre l'informazione utile, e l'aumento delle comunicazioni machine-to-machine (M2M) renderanno necessarie nuove tecniche di telecomunicazione in grado di ottimizzare il trasferimento nei diversi contesti. Questo non solo richiederà nuovi algoritmi per l'elaborazione finalizzati all'estrazione dell'informazione e all'ottimizzazione dei segnali trasmessi, ma anche dispositivi in grado di implementare in modo efficiente tali sistemi a livello edge in maniera autonoma, senza bisogno di interfacciarsi al cloud.

## 1.2 INTELLIGENZA ARTIFICIALE E RETI NEURALI

Le capacità di calcolo dei moderni dispositivi hanno permesso di sviluppare e realizzare algoritmi di elaborazione del segnale particolarmente complessi, come riconoscimento di immagini, elaborazione del linguaggio naturale e guida autonoma, che presentano comportamenti estremamente affini a quelli tipici dell'intelletto umano, prendendo pertanto il generico nome di intelligenza artificiale (Artificial Intelligence, AI).

Oggigiorno la maggioranza dei sistemi che integrano programmi di AI si basa sul Machine Learning (ML), tecnica che permette di svolgere funzioni anche particolarmente complesse senza programmare esplicitamente la macchina, ma regolando l'apprendimento del programma, rappresentato da un insieme di variabili interne, attraverso esempi. La grande adattabilità di questa classe di algoritmi ne ha favorito la diffusione nei più svariati campi e, soprattutto grazie al machine learning, ha permesso di risolvere problemi che le tecniche di programmazione tradizionale faticherebbero a gestire. La procedura di addestramento può avvenire per mezzo di dati opportunamente classificati, in modo da disporre della corrispondente uscita. Questa procedura necessita solitamente di un'elaborazione non automatizzata sui dati e per questo prende il nome di apprendimento supervisionato, contrapposto a quello non supervisionato che non richiede alcuna classificazione sui dati di addestramento.

Sebbene esistano svariate tecniche di data mining che si basano sul machine learning, le reti neurali risultano essere tra le più diffuse in quanto sono estremamente vantaggiose nel livello di parallelizzazione. La loro complessità può essere adattata in base alla precisione richiesta e presentano una estrema adattabilità nei diversi contesti applicativi. Il nome delle reti neurali deriva dalla somiglianza del funzionamento della struttura del neurone, elemento alla base della rete, con le cellule neurali umane. Infatti il neurone può essere visto come un sistema a $m$ ingressi e singola uscita connesso in una rete di suoi simili con cui interagisce scambiando segnali. L'elaborazione degli ingressi che avviene all'interno di questa struttura è divisibile in una parte lineare e una non lineare, confinata all'uscita del sistema. Ciascuno degli $m$ ingressi $x_j$, prima di essere sommato a tutti gli altri assieme a un bias $b_k$, viene moltiplicato per un peso $w_{k,j}$. Talvolta il bias viene indicato come peso di indice zero $w_{0,k}$ a cui corrisponde un ingresso unitario ($x_0 = 1$), in modo da utilizzare un unico vettore di parametri e facilitarne la lettura e la messa a punto.

Il risultato della sommatoria $v_k$ rappresenta l'uscita della componente lineare del sistema ed è l'ingresso di una funzione che prende il nome di funzione di attivazione $\varphi$, l'ultimo stadio di elaborazione in cui risiede anche l'unica non linearità della struttura.

$$v_k = \sum_{j=1}^{m} w_{k,j} x_j + b_k = \sum_{j=0}^{m} w_{k,j} x_j$$

$$y_k = \varphi(v_k) = \varphi\left( \sum_{j=0}^{m} w_{k,j} x_j \right)$$

La scelta della funzione di attivazione è totalmente arbitraria e dipende dalla specifica applicazione, anche se esistono diversi modelli di rete che sono più propensi a lavorare con certe classi di funzioni.

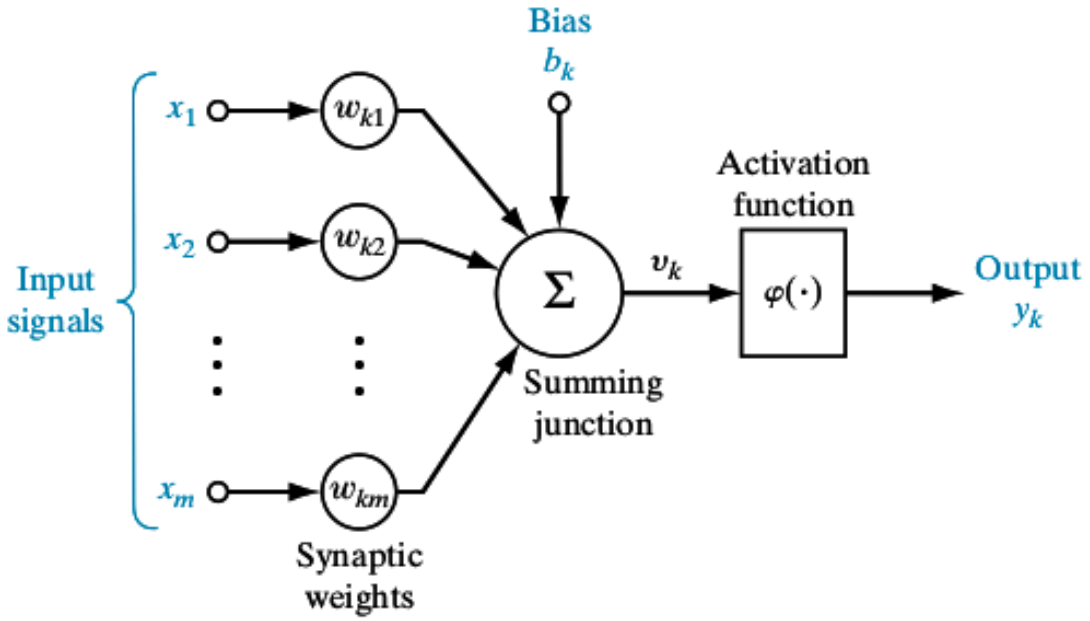



*Figura 1-2: Struttura di un neurone [2]*

La molteplicità di configurazioni con cui più neuroni, chiamati anche nodi, possono essere interconnessi e interfacciarsi con ingressi e uscite è smisurata e permette di ottenere risultati completamente differenti in base alla topologia scelta. In letteratura esistono molteplici soluzioni ma la distinzione principale avviene in base alla presenza di anelli chiusi di nodi all'interno della struttura: in caso affermativo si è in presenza di reti ricorrenti, altrimenti si parla di reti feed-forward.

Le reti più comuni e più semplici sono le reti feed-forward dal momento che non presentano comportamenti dinamici e, una volta addestrate impostandone i parametri, ai morsetti si comportano esattamente come una funzione analitica. Fra le reti feed-forward le più utilizzate sono presenti le reti convoluzionali (Convolutional Neural Network o CNN), il cui compito è quello di emulare l'elaborazione visiva umana attraverso convoluzioni successive sulle matrici o tensori di ingresso, e le reti multi-strato o stratificate (Multi-Layers Perceptron o MLP), in cui i neuroni sono disposti in gruppi successivi e le connessioni avvengono solo tra uno strato precedente e quello successivo. Solitamente i nodi appartenenti allo stesso strato presentano la stessa funzione di attivazione e le reti stratificate prendono anche il nome di reti pienamente connesse (fully connected o dense) a causa del fatto che la rappresentazione matematica si riduce a una serie di prodotti matrice-vettore.

Nelle reti pienamente connesse ciascun neurone dello strato precedente è collegato a ognuno dei nodi dello strato successivo, originando così una matrice di pesi $\boldsymbol{W}$ in cui il coefficiente $w_{k,j}$ rappresenta il fattore da moltiplicare all'uscita del j-esimo nodo dello strato precedente prima di entrare nel k-esimo nodo dello strato successivo. Ad ogni strato vi è quindi una moltiplicazione matriciale le cui dimensioni dipendono dal numero di elementi negli strati precedente e successivo.

Considerando che le dimensioni del vettore di ingresso e quelle di uscita sono dati, il minimo numero di strati di cui una rete neurale è composta è due e prendono i nomi di strato di ingresso, il primo, e strato di uscita, il secondo. In queste circostanze è richiesta una singola moltiplicazione matriciale. Spesso le reti neurali stratificate possiedono almeno anche uno strato intermedio denominato strato nascosto e nell'eventualità vi sia più di uno strato nascosto si parla di rete neurale stratificata profonda (Deep Neural Network o DNN). In questo caso lo sforzo computazionale cresce e sarà necessaria una moltiplicazione matriciale aggiuntiva per ogni strato che si inserisce ma allo stesso tempo la flessibilità della rete aumenta notevolmente. Infatti, a parità di dimensioni di ingresso e uscita, la rete sarà in grado di mappare meglio funzioni più articolate tanto più il numero di strati nascosti e il numero di nodi che li compongono è elevato.

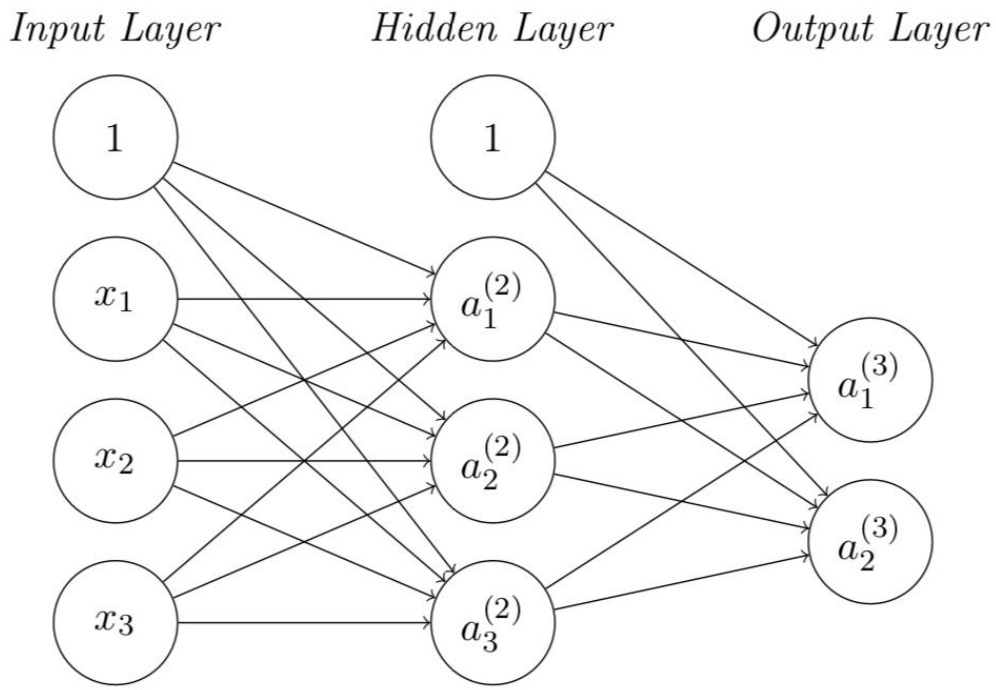



*Figura 1-3: Struttura di una rete neurale feed-forward stratificata a singolo strato nascosto [3]*

Una volta scelta la struttura della rete e selezionate le funzioni di attivazione di ciascuno strato il sistema è pronto per essere addestrato, ovvero è possibile avviare un algoritmo che ottimizzi i gradi di libertà rimanenti in modo da insegnare alla rete a svolgere la funzione richiesta. Questo apprendimento nel caso delle reti neurali avviene per mezzo della sintonizzazione delle matrici di peso. L'addestramento delle reti neurali avviene quasi esclusivamente in modo supervisionato fatta eccezione per alcuni casi particolari quali gli autoencoders, in cui ingresso e uscita sono uguali. Pertanto l'apprendimento richiede un insieme di dati sufficientemente grande da descrivere in modo esaustivo la funzione che si vuole mappare e abbini per ciascun vettore in ingresso un vettore di uscita corrispondente.

Per prima cosa è necessario definire una funzione scalare di errore $E(\boldsymbol{w})$ che permetta di quantificare la differenza tra il vettore d'uscita stimato e il vettore d'uscita corretto. Ovviamente questa differenza dipenderà dai valori del vettore $\boldsymbol{w}$, che in questo caso rappresenta l'insieme di tutti i coefficienti all'interno della rete. L'intero comportamento della rete è governato dai parametri contenuti in $\boldsymbol{w}$, pertanto l'addestramento della rete si pone come obiettivo di trovare i coefficienti che permettano di svolgere la funzione

desiderata. Successivamente i pesi sono impostati a un valore casuale, dopodiché viene fatta la predizione di una porzione del data set, ovvero si calcolano le uscite a fronte degli ingressi. Tramite il calcolo della funzione errore non solo è possibile stimare l'accuratezza della rete, ma anche calcolarne il gradiente e modificare i pesi spostandosi in direzione opposta. Questo sistema permette di raggiungere un minimo della funzione aggiornando il vettore pesi periodicamente ed è rappresentato dalla seguente formula:

$$\boldsymbol{w'} = \boldsymbol{w} - \lambda \nabla E(\boldsymbol{w})$$

in cui $\boldsymbol{w'}$ rappresenta in nuovo vettore dei pesi e $\lambda$ indica un fattore di apprendimento tramite il quale è possibile controllare la velocità con cui ci si muove contro gradiente. Questa fase prende il nome di retropropagazione dell'errore, con la quale si dovrebbe ridurre lo scarto tra l'uscita desiderata e quella ottenuta.

Solitamente sono necessarie più iterazioni dell'intero training set, che prendono il nome di epoche, per ottenere una buona rappresentazione della funzione desiderata, ma le ripetizioni dipendono molto dalla complessità della funzione richiesta e dall'insieme di dati per l'addestramento. Infatti è necessario che i dati impiegati siano rappresentativi di tutte le possibili casistiche che la funzione si impone di rappresentare e in caso contrario si rischia di non riuscire a raggiungere il livello di accuratezza previsto.

Un altro aspetto cruciale nella fase di addestramento è la scelta della funzione errore. Infatti non è sufficiente che essa sia monotona nell'intorno del punto di minimo ma è importante che questa sia convessa in tutto lo spazio considerato poiché, dal momento che il punto di partenza è casuale, il rischio è quello che l'addestramento porti in un punto di minimo locale, risultando una soluzione subottimale.

Nella fase di addestramento è molto importante anche studiare il criterio di interruzione grazie al quale l'addestramento termina. Infatti si è studiato che oltre un certo numero di epoche di allenamento, solitamente abbastanza elevato ma comunque dipendente dalla rete e dai dati, la rete cominci a perdere di efficienza nei casi reali, nonostante continui a ridurre l'errore tra le uscite stimate e quelle effettive nella fase di insegnamento. Questo fenomeno prende il nome di overfitting ed è dovuto al rumore di fondo presente su dati impiegati nell'addestramento, che compromette le prestazioni della rete qualora questa tenti in qualche modo di mapparlo. Per ridurre questo fenomeno e spostare il punto in cui l'allenamento produce un effetto negativo è necessario aumentare il numero e la varietà

di dati impiegati per addestrare la rete. Tuttavia è necessario non scontrarsi con il fenomeno opposto, l'underfitting, in cui si interrompe l'allenamento troppo presto, rinunciando a un possibile miglioramento delle prestazioni. Il punto di ottimo non solo dipende dalla quantità e dalla qualità dei dati, ma anche dalla complessità della rete e dalla funzione che si desidera emulare, pertanto non esiste un unico criterio di interruzione.

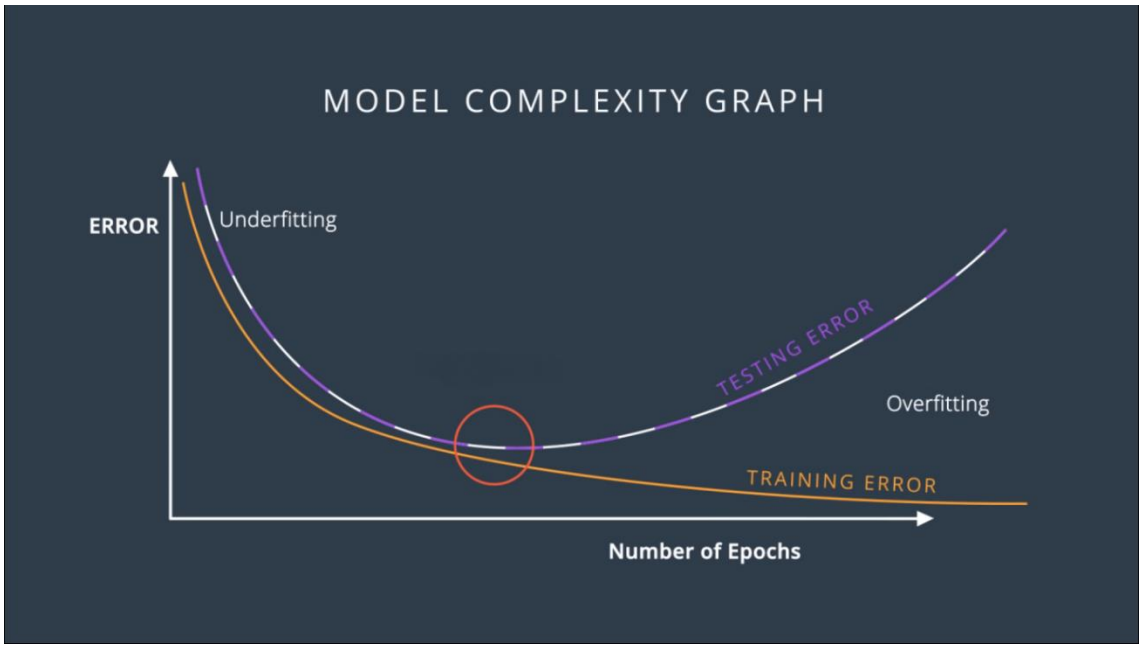



*Figura 1-4: Andamento dell'errore all'aumentare del numero di epoche di addestramento [4]*

Un meccanismo analogo è tipico nella scelta della complessità della rete: una rete troppo articolata (overfitting) o troppo minimalista (underfitting) risulta una scelta subottimale, ma anche in questo caso tutto dipende dalla funzione desiderata e dai dati impiegati nell'addestramento.

Una menzione importante va fatta sulle possibilità di parallelizzazione delle reti neurali. Infatti la loro diffusione è in parte anche dovuta alla enorme capacità di parallelizzare il calcolo, assente o molto limitata in buona parte degli algoritmi classici. Infatti una rete neurale idealmente può essere implementata con parallelismo a livello di neurone, considerando che ogni nodo è una entità funzionale autonoma e pertanto, se gli ingressi sono disponibili, può essere calcolato il risultato. Se a questo poi si aggiunge che in diversi modelli diffusi di rete neurale i nodi sono organizzati in modo da ridurre il calcolo della rete a un prodotto matriciale, è immediato capire quanto parallelismo possa essere

sfruttato in questo modo. Proprio questa caratteristica ha permesso di demandare gran parte del calcolo alle GPU, rendendo l'utilizzo di reti neurali più accessibile e veloce anche in dispositivi dalle prestazioni limitate.

# 2 DISPOSITIVI

## 2.1 SENSORI

I nodi sensori (o mote) rappresentano i punti di partenza dell'elaborazione del segnale nei paradigmi di edge e cloud computing e sono i dispositivi che comunemente si interfacciano con l'ambiente e gli utenti finali. Proprio per questa ragione la varietà con cui i nodi sensori si possono presentare è smisurata: a causa delle diverse condizioni di lavoro, dell'ambiente operativo, delle funzionalità offerte e dei vincoli a cui sono sottoposti, i nodi sensori dispongono di una estrema variabilità e flessibilità.

Proprio per questo la caratterizzazione generale di un nodo sensore è estremamente vaga ma i requisiti che di norma si richiedono sono in merito ai consumi, alle risorse a disposizione e al prezzo finale. Infatti, come gli stessi modelli di calcolo distribuito stabiliscono, a un dispositivo di questa categoria è richiesto prevalentemente di scambiare dati con il livello superiore (edge) e di interagire con l'ambiente mediante sensori e attuatori.

La computazione a questo livello è molto limitata dal momento che tutte le operazioni complesse avvengono per mezzo di richieste alla rete e l'elaborazione in locale è ridotta al minimo. Questo avviene perché la moltitudine di nodi sensori presenti rende necessario un costo per singolo elemento molto contenuto, oltre al fatto che in moltissimi casi anche l'energia a disposizione per alimentare lo stesso è estremamente limitata. Proprio per questo è raro trovare un nodo sensore con capacità di calcolo sufficienti per l'esecuzione di algoritmi elaborati e per lo stesso motivo esistono sistemi di Dynamic Power Management (DMP) che spengono o limitano alcuni blocchi del nodo sensore in modo da ridurre i consumi. L'essenzialità alla base dei nodi sensori rende facile individuare alcuni elementi ricorrenti:

- Controllore: serve a gestire l'intero sistema, controllando tutti gli altri moduli che ne fanno parte, e ha anche il compito di eseguire le operazioni di elaborazione dei dati demandate al nodo sensore. Solitamente è realizzato mediante un microcontrollore grazie al basso costo e all'ottima flessibilità offerti, ma esistono

anche realizzazioni mediante FPGA o ASIC dedicati in base alle esigenze. Più raro ma possibile è anche l'impiego di DSP e microprocessori più performanti, considerati però sono in applicazioni specifiche;

- ADC e DAC: qualora il controllore non contenga già convertitori per interfacciarsi al mondo analogico è necessario inserirli all'interno del nodo sensore dal momento che rappresentano l'unico modo per collegarsi con i sensori e gli attuatori, a meno che questi non ne siano già provvisti. Solitamente è utile abbinare anche unità accessorie come timer, controller PWM e comparatori, ma questi dipendono esclusivamente dal tipo di applicazione richiesta;
- Memoria non volatile: solitamente è costituita da una memoria flash di modeste dimensioni e contiene il programma eseguito dal controllore e le informazioni necessarie alla sua esecuzione, oltre a eventuali dati che non vengono inviati o salvati in RAM;
- Fonte di alimentazione: sicuramente rappresenta una delle parti più importanti e limitanti del sistema, dal momento che il suo compito è fornire energia a tutti gli altri blocchi del nodo. L'energia può essere fornita mediante un alimentatore o, più frequentemente, per mezzo di batterie e la sua distribuzione è fortemente dipendente dai contesti d'impiego. Sempre più frequente è l'adozione di tecniche di energy harvesting, tramite le quali si incamera l'energia disponibile nell'ambiente in modo da ridurre i consumi netti o eventualmente rendere energeticamente indipendente il nodo sensore;
- Modulo di comunicazione: si tratta del blocco che ha il compito di gestire il collegamento con il dispositivo edge a cui il nodo sensore e associato. Le comunicazioni possono avvenire mediante rete cablata o, più frequentemente, mediante rete wireless. Esistono numerosi protocolli e sistemi di telecomunicazione incentrati su questo tipo di collegamenti M2M che si differenziano in base alla quantità e alla tipologia di dati, agli intervalli di trasmissione, la quantità di nodi da interconnettere, ai requisiti energetici, alla banda occupata e alla distanza di copertura;
- Sensori e attuatori: si tratta dei blocchi funzionali che effettivamente si interfacciano con l'ambiente circostante, con il compito di misurare una grandezza fisica o di intervenire apportando modifiche su di essa. La loro varietà è elevata a causa delle molteplici situazioni e richieste, tuttavia nel caso degli

attuatori è sempre richiesta energia proveniente dall'alimentazione tramite un circuito di potenza comandato dal controllore. È comune che all'interno di un singolo nodo vi siano più sensori ed eventualmente anche uno o più attuatori, in modo da poter condividere le risorse computazionali ed energetiche evitando di dover sdoppiare il dispositivo.

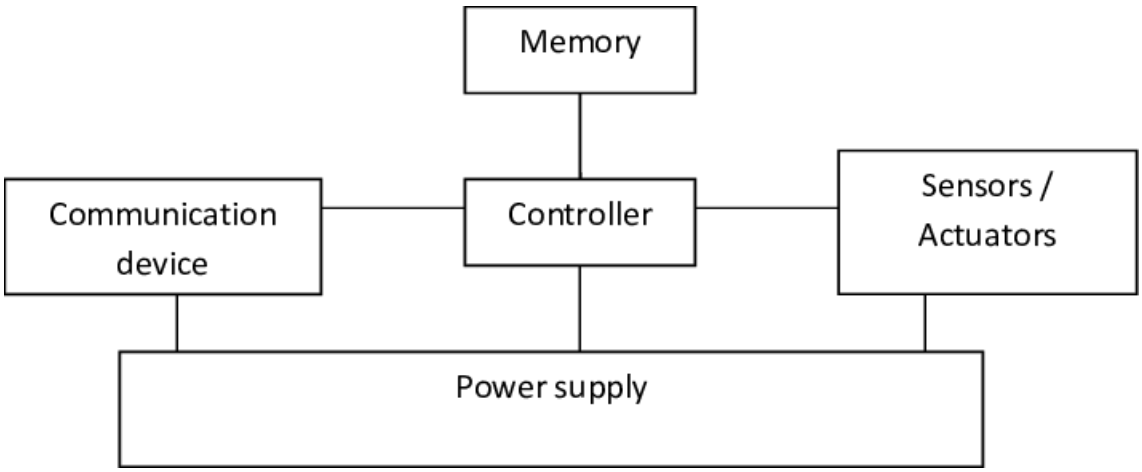



*Figura 2-1: Schema a blocchi di un nodo sensore [5]*

Una particolare configurazione di nodi sensori è la Wireless Sensor Network (WSN), una specifica architettura distribuita caratterizzata dalla capacità dei nodi di comunicare tra loro tramite collegamenti radio appositamente studiati e in maniera del tutto automatica, in modo da rendere facilmente gestibile un numero di nodi molto elevato.

Questo avviene grazie alle comunicazioni autonome tra i nodi sensori che permettono di aggiornare con facilità un gateway grazie a una rete multi-hop, che consente di ridurre il numero di comunicazioni al dispositivo principale e allo stesso tempo di aumentare il raggio di copertura. Infatti una rete multi-hop permette di usare i nodi che la costituiscono come ripetitori, dando la possibilità anche ai sensori che non sono in diretto contatto con il gateway di comunicarvi. Questo consente anche di non dover gestire troppi dispositivi tramite il gateway, evitando di sovraccaricare la rete passando da una configurazione a stella a una mesh. In questo modo è possibile connettere a internet ogni nodo della rete mediante il gateway, anche se il collegamento non avviene direttamente. Una rete di questo tipo risulta particolarmente versatile in diverse situazioni e inoltre dimostra un'ottima scalabilità.

Infatti in un sistema WSN è sempre possibile aggiungere nodi senza dover riconfigurare la rete, dal momento che la struttura permette ai nodi di interagire liberamente. Questo consente di anche di avere alti requisiti di affidabilità, poiché in caso di guasto di un nodo gli altri hanno la capacità di arginare il problema, continuando a comunicare tra loro. Questi ultimi due aspetti di scalabilità e affidabilità sono fondamentali non solo nelle WSN ma anche in tutte le reti che connettano nodi sensore.

Le limitazioni che riguardano le wireless sensor network sono quelle tipiche della maggior parte dei nodi sensori, vale a dire la ridotta capacità di calcolo e i vincoli energetici molto stringenti. A queste vanno aggiunte una latenza superiore alle altre tipologie di rete dovuta alle comunicazioni multi-hop. Per limitare questi problemi che affiggono questa particolare tipologia di rete, sono stati sviluppati appositi protocolli di comunicazione finalizzati a migliorare la sincronizzazione, la copertura e la latenza delle reti multi-hop, considerando anche la bassa disponibilità energetica. Questi sistemi di telecomunicazione solitamente risolvono il problema delle poche risorse energetiche, limitando banda e potenza di trasmissione e accendendo i transceiver solo in determinati intervalli temporali. I problemi di sincronizzazione e latenza invece risultano più difficili da eliminare anche se esistono protocolli ad hoc finalizzati a ottimizzare questi aspetti. Ciononostante le reti WSN risultano essere tra le più utilizzate in ambito IoT proprio grazie alla loro enorme scalabilità e versatilità, che permette di adattarle in numerosissimi contesi applicativi.

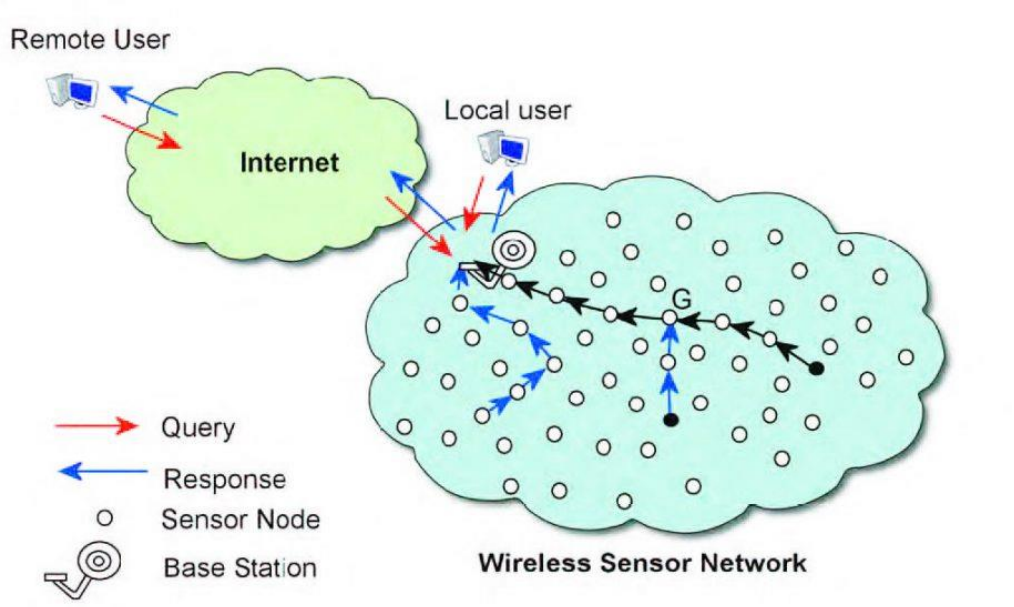



*Figura 2-2: Struttura di una Wireless Sensor Network (WSN) [6]*

## 2.2 PIATTAFORME EDGE

I dispositivi edge comprendono una vasta gamma di apparecchiature che, nei modelli di calcolo distribuito, si collocano tra i nodi sensori e il cloud. Questi sistemi sono caratterizzati da avere maggiori risorse computazionali ed energetiche rispetto ai nodi sensori e possono permettersi di avere un costo superiore, anche se la differenza principale rispetto alle altre due categorie di dispositivi risiede nelle funzionalità. Infatti un nodo edge è chiamato a svolgere particolari compiti che lo contraddistinguono da un nodo sensore:

- Comunicazioni: un nodo edge, data la sua particolare posizione nella catena di elaborazione, è specializzato nelle comunicazioni machine to machine, ottimizzando la distribuzione di dati tra cloud e sensori. In particolare si differenzia da un nodo sensore grazie alla capacità di gestire un maggiore flusso di dati provenienti da più sorgenti. Proprio per questo accade spesso che, nei sistemi meno articolati, un nodo edge svolga anche il compito di gateway nella rete, gestendo i nodi sensori in maniera autonoma e locale, evitando un interfacciamento diretto tra i sensori e internet;

- Elaborazione: diversamente dai nodi sensore, gli edge sono caratterizzati da una capacità computazionale nettamente superiore che permette di eseguire algoritmi di elaborazione del segnale di complessità elevata, fino a implementare tecniche di intelligenza artificiale e machine learning. Questo permette non solo a questi dispositivi di fungere da aggregatori, ma anche compiere significative trasformazioni dei segnali in ingresso, talvolta evitando di dover ricorrere al cloud;
- Archiviazione: le piattaforme edge dispongono di memorie di ordini di grandezza superiori a quelle dei nodi sensore, pertanto possono immagazzinare una discreta quantità di dati provenienti dai sistemi sottostanti, alleggerendo il cloud e la rete dalla trasmissione e dall'accumulo della grande quantità di informazioni proveniente dai sensori;
- Complemento al cloud: in generale i nodi edge sono chiamati a svolgere sempre più attività fortemente affini a quelle offerte dal cloud, spostando l'elaborazione e l'archiviazione sempre più verso i nodi periferici in modo da non appesantire la rete e ridurre le latenze sui dati. Questo nuovo funzionamento alleggerisce anche il cloud dall'esecuzione di molti compiti ausiliari, lasciando ai nodi centrali solo gli incarichi a maggiore sforzo computazionale. Pertanto uno dei compiti più importanti dei nodi edge è quello di ottimizzare la quantità di informazioni inviate al cloud in modo da garantirne la massima efficienza nell'elaborazione finale.

Come già spiegato, la possibilità di eseguire elaborazione in prossimità della sorgente dati rappresenta un enorme vantaggio in termini di latenza e sicurezza dei dati, pertanto è particolarmente interessante studiarne le potenzialità a livello computazionale, per capire in quali situazioni l'edge computing può subentrare al cloud.

In particolare poter implementare modelli di intelligenza artificiale all'interno di nodi edge senza la necessità di ricorrere alla rete rappresenta una svolta per l'IoT e per numerosi campi applicativi, in primis per i settori automotive e biomedicale. Questo ha portato allo sviluppo di dispositivi appositamente studiati all'esecuzione di determinati algoritmi di intelligenza artificiale, grazie a specifici acceleratori hardware, che permettono di ottenere prestazioni elevate a un costo contenuto e con un basso consumo energetico.

Parallelamente all'ottimizzazione dell'hardware mediante specifici acceleratori, sono stati sviluppati algoritmi, framework e librerie appositamente studiati per le risorse di un nodo edge, riducendo la richiesta di risorse a fronte di una minor complessità del modello e della conseguente accuratezza. Questa ottimizzazione software ha accresciuto notevolmente le opportunità, permettendo di poter integrare intelligenza artificiale anche in piattaforme edge dalle risorse computazionali più limiate o perfino in alcuni nodi sensori.

Grazie alla versatilità dei modelli di intelligenza artificiale e la possibilità di implementare lo stesso algoritmo su differenti dispositivi hardware, è possibile impiegare apparecchiature diverse per l'esecuzione dello stesso compito, variando prevalentemente prestazioni, prezzo e consumo energetico. Proprio per questo si sono considerate diverse soluzioni commerciali per implementare un nodo edge, studiandone prima le specifiche e poi verificandone le prestazioni nell'esecuzione di reti neurali. Questo studio ha l'obiettivo di comprendere come le prestazioni cambino al variare della rete neurale eseguita, in modo da individuare quale piattaforma sia la più idonea a seconda del contesto applicativo e delle prestazioni richieste.

Le piattaforme prese in considerazione sono cinque: Nvidia Jetson Nano Developer Kit, Google Coral Dev Board, Raspberry Pi, STM32MP157A-DK1 e STM32 Nucleo-H743ZI2. Nei paragrafi successivi verranno illustrate nel dettaglio le specifiche di ciascuna scheda per poi poterle confrontare e prevedere il loro comportamento nell'esecuzione di modelli di AI. Si tratta di un insieme di dispositivi volutamente molto eterogeneo, a dimostrazione della varietà con cui si possono realizzare nodi edge.

### 2.2.1 Nvidia Jetson Nano Developer Kit

La Nvidia Jetson Nano Developer Kit è una scheda di sviluppo di intelligenza artificiale che integra unità di calcolo volte all'accelerazione di algoritmi di machine learning e computer vision. Sebbene tale prodotto si rivolga prevalentemente al mercato della prototipazione, i suoi costi contenuti e la grande capacità computazionale, unite all'ottima efficienza energetica, la rendono appetibile anche in parecchie applicazioni commerciali che si basano su reti neurali profonde, specialmente nel settore dell'internet of things,

come classificazione di immagini, riconoscimento di oggetti, processing multimediale, GPU computing e deep learning.

La Jetson Nano si basa su un modulo GPU Nvidia Maxell a 128 core affiancato da una CPU quad-core ARM Cortex-A57 con parallelismo 64 bit e frequenza di clock di 1,43 GHz, supportate da una RAM LPDDR4 da 4GB con bus a 64-bit a 1600 MHz che permette un trasferimento dati fino a 25,6 GB/s. Questa configurazione consente di raggiungere prestazioni di 472 GFLOP e, grazie a specifici acceleratori VPU (Video Processing Unit), velocità di codifica video 4K secondo gli standard H.264/H.265 di 30 frame al secondo, che aumentano a 60 in caso di decodifica. Il punto di forza di questa configurazione hardware, oltre al prezzo molto competitivo di appena 100 $, è che tali risultati sono ottenuti mantenendo i consumi stimati compresi tra i 5 e i 10 W, grazie alla notevole efficienza energetica dei componenti scelti.

La scheda è inoltre fornita di una ricca connettività che permette l'utilizzo nei numerosi contesti applicativi a cui è rivolta. La Jetson Nano possiede 40 GPIO attraverso i quali è possibile accedere alle interfacce seriali $I^2C$, $I^2S$, SPI e UART; in aggiunta dispone di 4 USB 3.0 tipo A e una micro USB 2.0 di tipo B. Infine sono presenti uno slot di espansione M.2, un connettore RJ45 per l'Ethernet gigabit, lettore di schede microSD, porte video HDMI 2.0 e DisplayPort 1.4 e un connettore per la videocamera MIPI CSI-2 in grado di supportare fino a 4 stream video simultanei (18 Gb/s). L'unica lacuna nella connettività della Jetson Nano è l'assenza di un modulo radio, che limita l'utilizzo della scheda in alcune applicazioni wireless che prevedono uso di reti come Wi-Fi o Bluetooth, mancanza che può essere facilmente risolta mediante l'espansione attraverso altre interfacce.

L'alimentazione della scheda avviene mediante un apposito connettore oppure per mezzo della porta micro USB, richiedendo 5 V e 4 A nel primo caso e 5 V e 2 A nel secondo. In alternativa è possibile alimentare il kit di sviluppo anche per mezzo della porta Ethernet, in quanto supporta l'alimentazione PoE (Power over Ethernet). All'interno della scheda è presente una memoria flash eMMC da 16 GB, sulla quale non è però possibile installare il sistema operativo, rendendo necessario l'impiego di una scheda microSD nella maggior parte delle applicazioni.

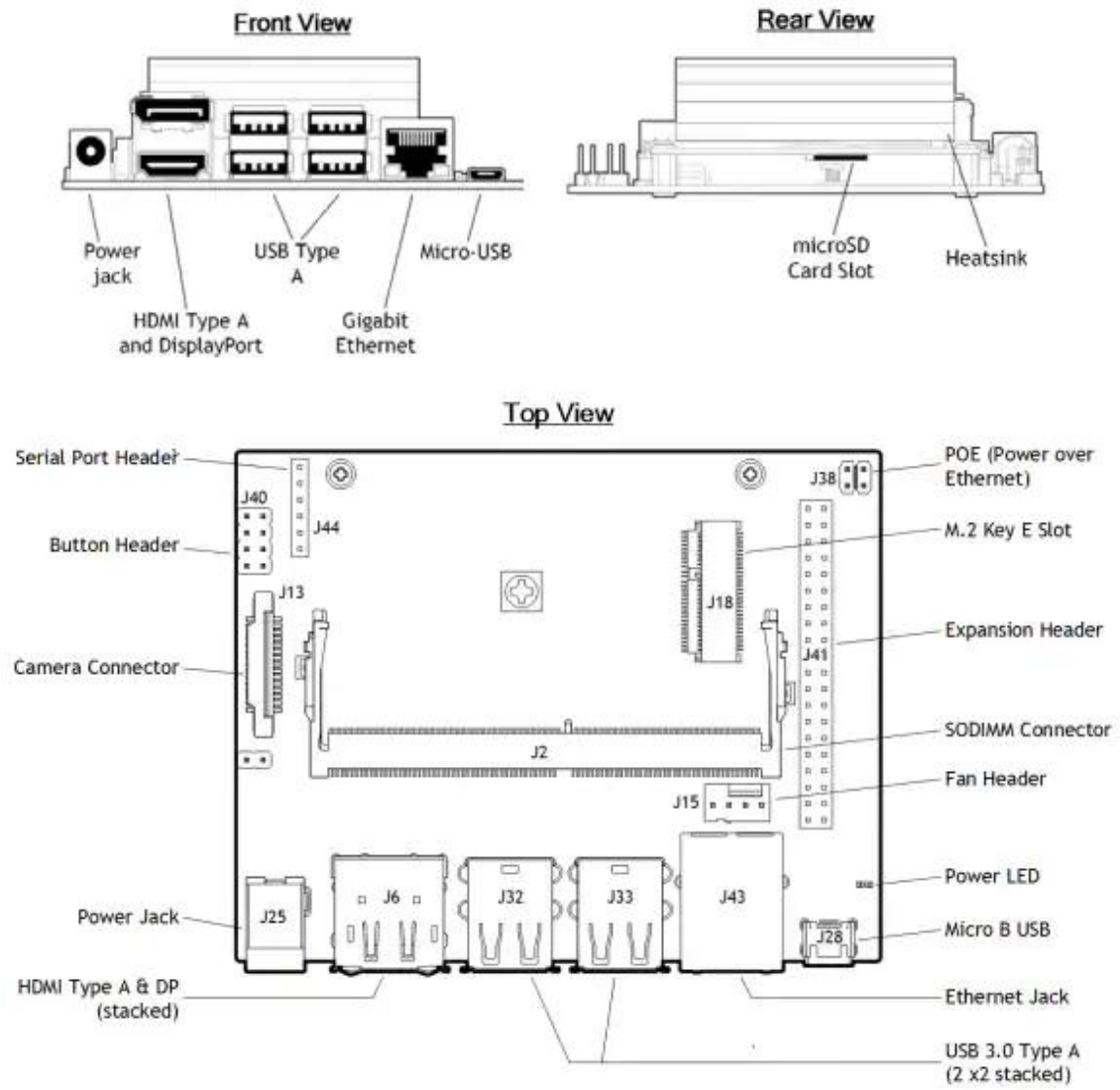



*Figura 2-3: Interfacce della Nvidia Jetson Nano Developer Kit*

La Jetson Nano è supportata dal Nvidia JetPack SDK, un Software Development Kit sviluppato da Nvidia per tutti i suoi moduli della serie Jetson, che permette di gestire in maniera ottimale i prodotti dell'azienda, fornendo un software studiato per garantire le massime prestazioni nelle applicazioni di intelligenza artificiale sulla base dell'hardware adottato. JetPack, oltre a fornire un sistema operativo, mette a disposizione librerie, esempi, strumenti di sviluppo e documentazione, garantendo un supporto software dei prodotti Nvidia efficiente ed aggiornato. Il sistema operativo contenuto in JetPack 4.3 è il Nvidia L4T nella versione 32.3.1, derivato da Linux Ubuntu 18.04 in versione 64 bit con kernel 4.9, che fornisce automaticamente numerose librerie e driver, sufficienti per la maggior parte delle applicazioni. Oltre a questo si trovano preinstallati diversi driver

per la gestione corretta ed efficiente dell'hardware della piattaforma e alcuni tool di controllo e utilità. In particolare, oltre al kernel Linux, al filesystem e al bootloader, L4T integra un Board Support Package (BSP) che contiene tutte le librerie necessarie per lo sviluppo e la prototipazione di applicazioni IoT sulle schede Nvidia. Inoltre in L4T sono inclusi diversi gruppi di librerie aggiuntive, divise a seconda delle funzioni che sono chiamate a svolgere:

- Artificial Intelligence: include Nvidia TensorRT e Nvidia Cuda Deep Neural Network (cuDNN) che rappresentano le API grazie alle quali è possibile raggiungere notevoli miglioramenti nello sviluppo, validazione e ottimizzazione di reti neurali profonde;
- Computer Vision: gruppo composto dalle librerie VisionWorks e OpenCV, grazie alle quali è possibile ottimizzare le direttive di Computer Vision;
- Camera & Display (libArgus, NVC-DRM, X Driver e Weston): API di basso livello per la gestione accelerata delle interfacce video in ingresso e uscita;
- Cuda Toolkit: librerie per lo sviluppo in C e C++ per applicazioni sviluppate su sistemi a GPU;
- Graphics: insieme di librerie grafiche di base 2D e 3D che includono OpenGL, OpenGL ES, EGL e Vulkan;
- Developer Tools (Tegra System Profile, Tegra Graphics Debugger, PerfKit): librerie per l'ottimizzazione e il debug delle applicazioni.

Oltre a questo, JetPack fornisce un sistema a interfaccia grafica per la gestione del software e delle librerie, rendendo semplice e intuitivo integrare moduli aggiuntivi ottimizzati nel caso l'applicazione faccia uso di librerie non presenti nell'installazione di base di L4T.

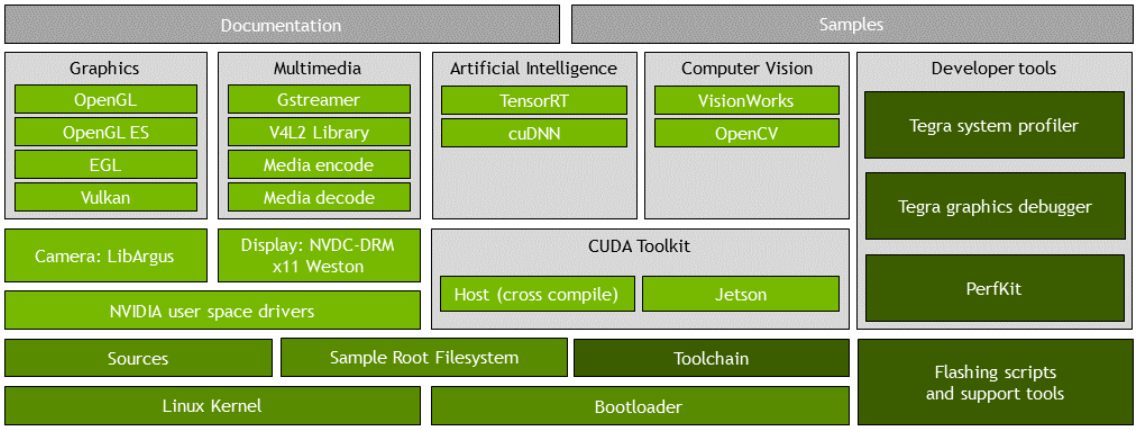



*Figura 2-4: Architettura del Board Support Package (BSP) della Nvidia Jetson Nano Developer Kit*

## 2.2.2 Google Coral Dev Board

La Google Coral Dev Board è una scheda di sviluppo progettata per la prototipazione e l'utilizzo di intelligenza artificiale a basso consumo di potenza e alto rendimento, in modo da poter offrire prestazioni elevate con un impatto energetico limitato, rendendola una soluzione estremamente conveniente sia in ambito IoT sia in campo mobile. L'hardware presente in questa piattaforma è estremamente ottimizzato e adatto all'esecuzione di reti neurali profonde, comprendendo un insieme di acceleratori rivolti a incrementare le prestazioni e migliorare la resa energetica. Il sistema ha come nucleo computazionale l'Edge TPU SoM (System-on-Module) che integra differenti unità di calcolo ognuna con un compito specifico:

- Il SoC NXP i.MX 8M è formato da due CPU con architettura ARM: un Cortex-A53 quad-core a 64 bit con frequenza di clock di 1,5 GHz per l'esecuzione del sistema operativo e dei task general purpose, e un Cortex-M4 a 266 MHz e parallelismo 32 bit, con il compito di gestire le periferiche di basso livello ed eseguire operazioni DSP floating point in tempo reale a basso consumo energetico. Ciascun Cortex-A53 è dotato di una cache di primo livello di 32 kB per i dati e 32 kB per le istruzioni, oltre ad avere una cache di secondo livello unificata di 1 MB; diversamente il Cortex-M4 possiede un solo livello di cache divisa in 16 kB per i dati e 16 kB per le istruzioni e affiancata da una memoria TCM (Tightly-Coupled Memory) di 256 kB per ridurre la latenza e le miss in memoria e migliorare le prestazioni in tempo reale. Il Cortex-A53 integra inoltre

un Media Processing Engine (MPE) basato sulla tecnologia ARM NEON, architettura a supporto SIMD avanzato grazie alla quale è possibile parallelizzare il calcolo di più istanze con una sola istruzione.

- La Google Edge TPU ML rappresenta la vera innovazione della scheda: un ASIC dedicato ad aumentare la velocità di esecuzione degli algoritmi di intelligenza artificiale e machine learning. Questa TPU (Tensor Processing Unit) permette, oltre ad accelerare i prodotti tensoriali alla base dei motori di inferenza, di ridurre drasticamente i consumi, risultando una soluzione ottimale per le piattaforme edge con disponibilità energetica limitata che implementano algoritmi di machine learning. Quantitativamente Google dichiara prestazioni di 2 TOPS/W fino a un massimo di 4 TOPS a 2W. La TPU si interfaccia con il modulo principale di calcolo NXP i.MX 8M mediante collegamenti PCIe e $I^2C$.
- La Vivante GC7000Lite è una GPU general purpose che permette di velocizzare le operazioni grafiche grazie ai 16 core alla frequenza di 1 GHz, ai 4 shaders e al supporto a numerose librerie grafiche tra cui le OpenGL ES 1.1, 2.0, 3.0, 3.1 e le Vulkan. Le prestazioni raggiunte sono di 32 GFLOPS con aritmetica a 32 bit, che raddoppiano in caso di operazioni a 16 bit. Oltre alle prestazioni grafiche notevoli questa unità risulta essere particolarmente indicata in caso di calcolo eterogeno grazie alla tecnologia HSA e le librerie OpenCL 1.2 che sfruttano il bus AMBA ACE-Lite per gestire la coerenza di cache tra CPU e GPU.
- Gli Acceleratori VPU (Video Processing Unit) sono delle unità ad hoc per la codifica e la decodifica video secondo gli standard H.264 e H.265 che permettono di raggiungere i 60 fotogrammi al secondo con la risoluzione massima di 4K.
- Il modulo ATECC608A è un coprocessore crittografico finalizzato all'accelerazione degli algoritmi di sicurezza basati su chiavi asimmetriche di firma (pubblica e privata) con protocolli ECDSA (Elliptic Curve Digital Signature Algorithm) ed ECC (Elliptic Curve Cryptography).

Queste unità di elaborazione sono affiancate da una memoria ROM da 128 kB riservata all'avvio del sistema e due moduli di RAM interni al chip da 128 kB e 32 kB. Esternamente è presente una SDRAM LPDDR4 da 1 GB interfacciata direttamente al controller DDR del modulo NXP i.MX 8M mediante un bus a 32 bit con frequenza di funzionamento di 1600 MHz, raggiungendo velocità di trasferimento di 12,8 GB/s. La scheda è provvista anche di una memoria flash NAND eMMC da 8 GB affiancata da un

lettore microSD con interfaccia SDIO 3.0 (Secure Digital Input Output) e con supporto all'avvio da scheda, che permette di alloggiare il sistema operativo esternamente.

La scheda di sviluppo integra un connettore da 40 pin GPIO, di cui 16 abilitati alla generazione di interrupt, che permette di connettersi al modulo mediante le diverse interfacce seriali di cui è fornita: 2 UART, 2 $I^2C$, 4 linee PWM e 2 SPI. Inoltre sono presenti una USB 3.0 con connettore type-C e supporto OTG, una USB 3.0 con connettore di tipo A e un connettore RJ45 per la connessione Ethernet gigabit secondo lo standard IEEE 802.3. In ambito di connettività multimediale la scheda dispone di uscita HDMI 2.0a con risoluzione fino a 4K, una linea MIPI CSI-2 con due connettori per videocamera che supporta fino a 4 stream video con risoluzione 1080p a 60 frame al secondo, un connettore MPI DSI per il display, un jack audio da 3,5 mm e un connettore stereo a 4 terminali. La connettività wireless è gestita dal modulo Murata LBEE5U91CQ che permette collegamenti mediante Bluetooth in versione 4.2, con supporto BLE e interfaccia UART, e Wi-Fi MIMO 2x2 secondo gli standard 802.11a/b/g/n/ac. La scheda si completa di un connettore USB Tipo C per l'alimentazione a 5 V e un massimo di 2 A e una porta Micro USB tramite la quale si può stabilire un collegamento seriale per la programmazione.

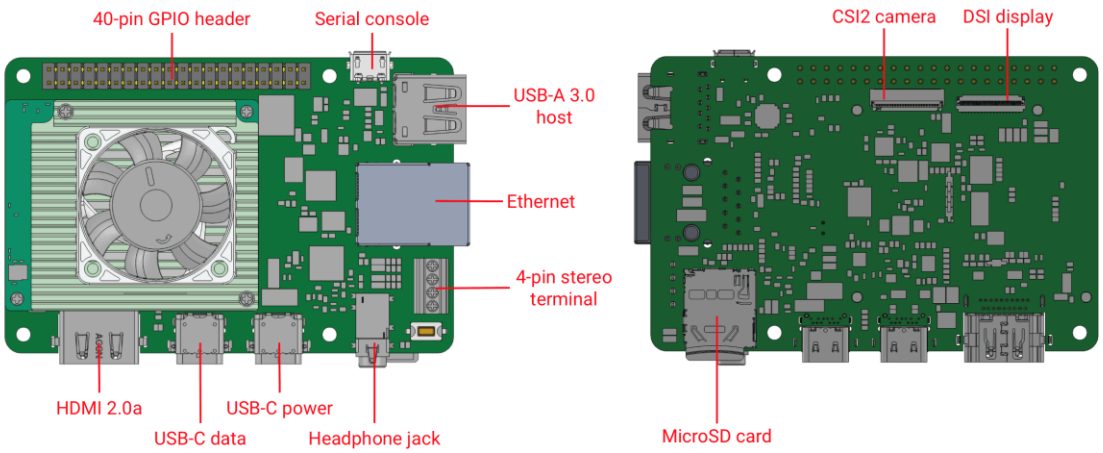



*Figura 2-5: Interfacce della Google Coral Dev Board*

La scheda di sviluppo è supportata da un sistema operativo ad hoc distribuito da Google di nome Mendel Linux, che permette di accedere alle potenzialità fornite dall'hardware in modo semplice e automatico. Questo software open source deriva dalla distribuzione Linux Debian con kernel alla versione 4.14, appositamente alleggerita e ottimizzata in base alle risorse a disposizione, perfezionando non solo sul sistema operativo in genere,

ma anche programmi, librerie e framework al suo interno quali Python 3.7, OpenCV, OpenCL e TensorFlow Lite, garantendo prestazioni migliorate nell'impiego di questi applicativi.

### 2.2.3 Raspberry Pi 4

La Raspberry Pi 4 Model B è una scheda finalizzata all'esecuzione di sistemi operativi basati su kernel Linux e dal costo molto contenuto, risultando una delle soluzioni più diffuse nella prototipazione e nelle applicazioni a ridotto costo computazionale; caratteristiche la rendono una scelta molto valida in svariati campi, incluso quello dell'internet of things.

Nonostante non vi siano acceleratori specificatamente finalizzati all'incremento delle prestazioni di reti neurali profonde, l'impiego di questa scheda per svolgere task come la classificazione di immagini e il deep learning risulta essere comunque una interessante ed economica soluzione general purpose a problemi ad alto carico computazionale, nonché un buon metro di paragone per il confronto con altre architetture più ottimizzate ed efficaci.

Essendo la versione più aggiornata della scheda, la Raspberry Pi 4 beneficia di un hardware attuale ed efficiente basato su una CPU Broadcom BCM2711 a quattro core Cortex-A72 a 64 bit con una frequenza di clock di 1,5 GHz. La scheda è disponibile nei tagli di RAM da 1, 2 e 4 GB LPDDR4 con bus a 64 bit e frequenza di 3200 MHz raggiungendo una velocità di trasferimento di 51,2 GB/s. Nell'applicazione richiesta verrà utilizzata una Raspberry Pi 4 con il massimo ammontare di memoria RAM pari a 4 GB. Pur non avendo una GPU dedicata la scheda è provvista di acceleratori per la codifica e la decodifica video (VPU) secondo gli standard H.264/H.265 e supporto alle librerie grafiche OpenGL ES 3.0, garantendo una buona efficienza e velocità nelle applicazioni multimediali, raggiungendo una decodifica 4K a 60 frame al secondo e una codifica 1080p a 60 frame al secondo.

Un notevole punto di forza della Raspberry Pi è l'abbondanza di porte con le quali ci si può interfacciare, rendendola estremamente versatile in moltissimi contesti applicativi. Oltre ai 40 GPIO è possibile utilizzare 6 $I^2C$, 5 SPI e 6 UART, insieme alle 2 USB 3.0 e alle 2 USB 2.0 di tipo A. L'alimentazione della scheda può avvenire mediante USB di

tipo C oppure per mezzo dei GPIO, richiedendo in entrambi i casi 5 V e fino 3 A, imponendo che i consumi non superino i 15 W. Inoltre e disponibile una porta Ethernet gigabit con connettore RJ45, attraverso la quale è anche possibile alimentare la scheda (PoE). Non avendo una memoria flash interna, la Raspberry Pi 4 necessita di un lettore di schede microSD per l'alloggiamento del sistema operativo.

Per quanto riguarda le interfacce multimediali la Raspberry Pi 4 è dotata di 2 uscite micro-HDMI 2.0, che permettono di controllare due monitor esterni con risoluzione massima di 4K ciascuno, una porta MIPI CSI a due vie per connettere fino a due videocamere e una porta MPI DSI a due vie, potendo così collegare uno o due display. È inoltre disponibile un jack audio da 3,5 mm a 4 poli, attraverso il quale si può accedere al segnale audio stereo in uscita e al video composito. Infine la scheda è dotata di un modulo radio che le fornisce connettività wireless secondo gli standard 802.11b/g/n/ac e Bluetooth 5.0 con supporto a Bluetooth Low Energy (BLE).

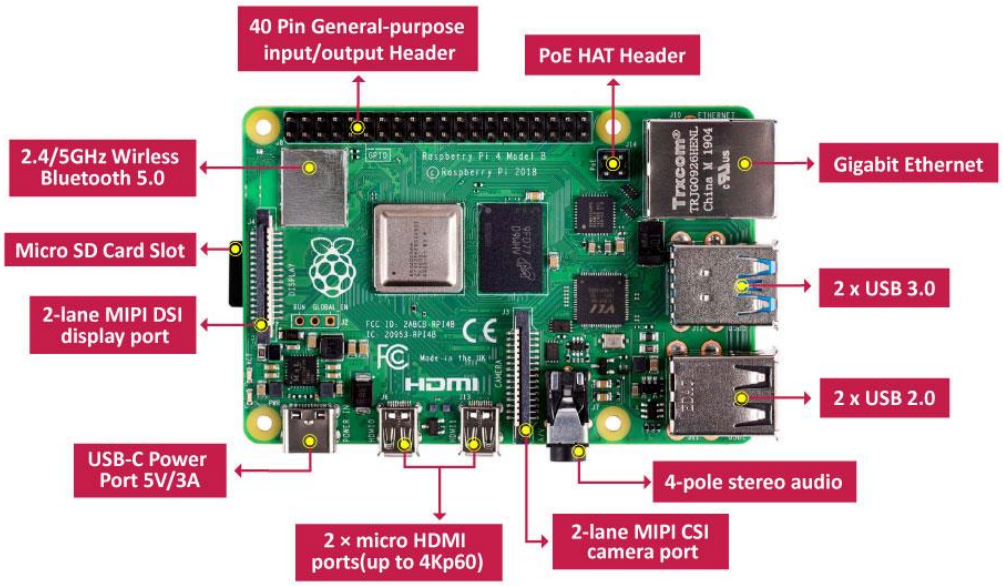



*Figura 2-6: Interfacce della Raspberry Pi 4*

Grazie alla sua ampia diffusione e ai suoi diversi contesti di applicazione, la Raspberry Pi 4 dispone di diversi sistemi operativi di terze parti realizzati e ottimizzati appositamente per la scheda. Tuttavia il sistema operativo ufficiale e più conosciuto, nonché lo stesso diffuso dalla stessa Raspberry Pi Foundation, è Raspbian. Questo è

derivato dalla distribuzione Linux Debian 10 Buster ottimizzato appositamente per lavorare con architetture ARM e tutti i componenti presenti all'interno della scheda. Infatti il sistema operativo dispone già di tutte le librerie necessarie a gestire l'hardware nel miglior modo possibile e numerosi software preinstallati, oltre a essere compatibile con tutti i programmi aggiuntivi disponibili per Linux Debian. La Raspberry Pi Foundation fornisce anche un tool software denominato NOOBS (New Out Of the Box Software) per l'installazione in modo semplice e intuitivo di Raspbian sulla microSD.

### 2.2.4 STM32MP157A-DK1

La STM32MP157A-DK1 è una scheda di sviluppo prodotta dalla STMicroelectronics basata sul modulo di calcolo STM32MP157, un'architettura eterogenea e versatile prevalentemente pensata per l'intelligenza artificiale e l'elaborazione grafica in tempo reale che aggrega una CPU ARM Cortex-A7 dual-core e una ARM Cortex-M4, entrambe a 32 bit, e una GPU specializzata nella grafica 3D. La soluzione a due processori permette di avere una unità più potente ed energivora, come il Cortex-A7, utilizzata negli incarichi più onerosi (come l'esecuzione del sistema operativo), ma allo stesso tempo garantisce una buona gestione delle interfacce in tempo reale demandando le attività di comunicazione con le periferiche e di signal processing basilari al Cortex-M4. Questa configurazione di calcolo, se si considera anche la GPU, risulta essere molto interessante se si tiene presente il contesto applicativo dell'internet of things e del machine learning, nei quali le piattaforme edge sono spesso chiamate ad eseguire algoritmi di complessità notevole con disponibilità energetiche molto spesso limitate.

Il modulo di calcolo STM32MP157 si compone due ARM Cortex-A7 con frequenza di funzionamento di 650 MHz e due livelli di cache, rispettivamente da 64 kB (32 kB per le istruzioni e 32 kB per i dati) e 256 kB, e un ARM Cortex-M4 con clock a 209 MHz e con il supporto alle istruzioni DSP e unità a virgola mobile a singola precisione. L'unità di elaborazione STM32MP157 integra anche una GPU 3D compatibile con le librerie OpenGL ES 2.0 e con frequenza di funzionamento di 533 MHz per velocizzare le operazioni di elaborazione di immagini e video e il calcolo parallelo. Internamente il modulo dispone di 256 kB di RAM per il sistema, 384 kB di RAM per le funzioni da microcontrollore e 64 kB di retention RAM; inoltre la scheda incorpora esternamente,

mediante bus DDR3L da 16 bit a 533 MHz, ulteriore modulo di memoria RAM per il sistema da 512 MB. Il modulo non è dotato di memorie flash interne ma possiede diverse interfacce tra le quali 4 UART, 4 USART, 6 $I^2C$, 3 $I^2S$, 6 SPI, 2 ADC a 16 bit e 2 DAC a 12 bit e i tutti controller necessari al funzionamento delle periferiche aggiuntive sulla scheda (HDMI, GPIO, USB e Ethernet). L'interfaccia del kit di sviluppo si completa con 4 porte USB tipo A, un connettore per la videocamera MIPI-CSI, connettore HDMI, slot microSD, modulo per la programmazione e il debug STLINK-V2.1, connettore RJ45 per il collegamento gigabit Ethernet, jack stereo da 3,5 mm, connettore di espansione per collegare la scheda ad Arduino Uno V3 e 40 GPIO compatibili con le espansioni per Raspberry Pi. L'alimentazione della scheda avviene mediante connettore USB tipo C, mediante GPIO o STLINK, con un assorbimento massimo di 15 W (5 V e 3 A).

Nonostante sia possibile programmare la scheda con gli ambienti di sviluppo comunemente diffusi per i microcontrollori (come IAR, Keil e gli IDE GCC) STMicroelectronics fornisce un sistema operativo open-source derivato da Linux Yocto Project, distribuzione di Linux ottimizzata per sistemi embedded per l'internet of things, di nome OpenSTLiunx Distribution, specifico per operare con schede basate su processori Cortex-A7 a singolo o doppio core. In particolare questo sistema operativo si basa sull'accostamento di soluzioni per l'accelerazione di task che richiedono un elevato ammontare di risorse computazionali e supporto grafico, congiuntamente a procedure finalizzate all'ottenimento della massima resa energetica e al controllo in tempo reale. Questo è possibile grazie all'hardware molto adattabile sostenuto da un codice di supporto e da driver opportunamente studiati e ottimizzati che permettono di beneficiare delle risorse al meglio. Tale sistema viene memorizzato esternamente sulla microSD.

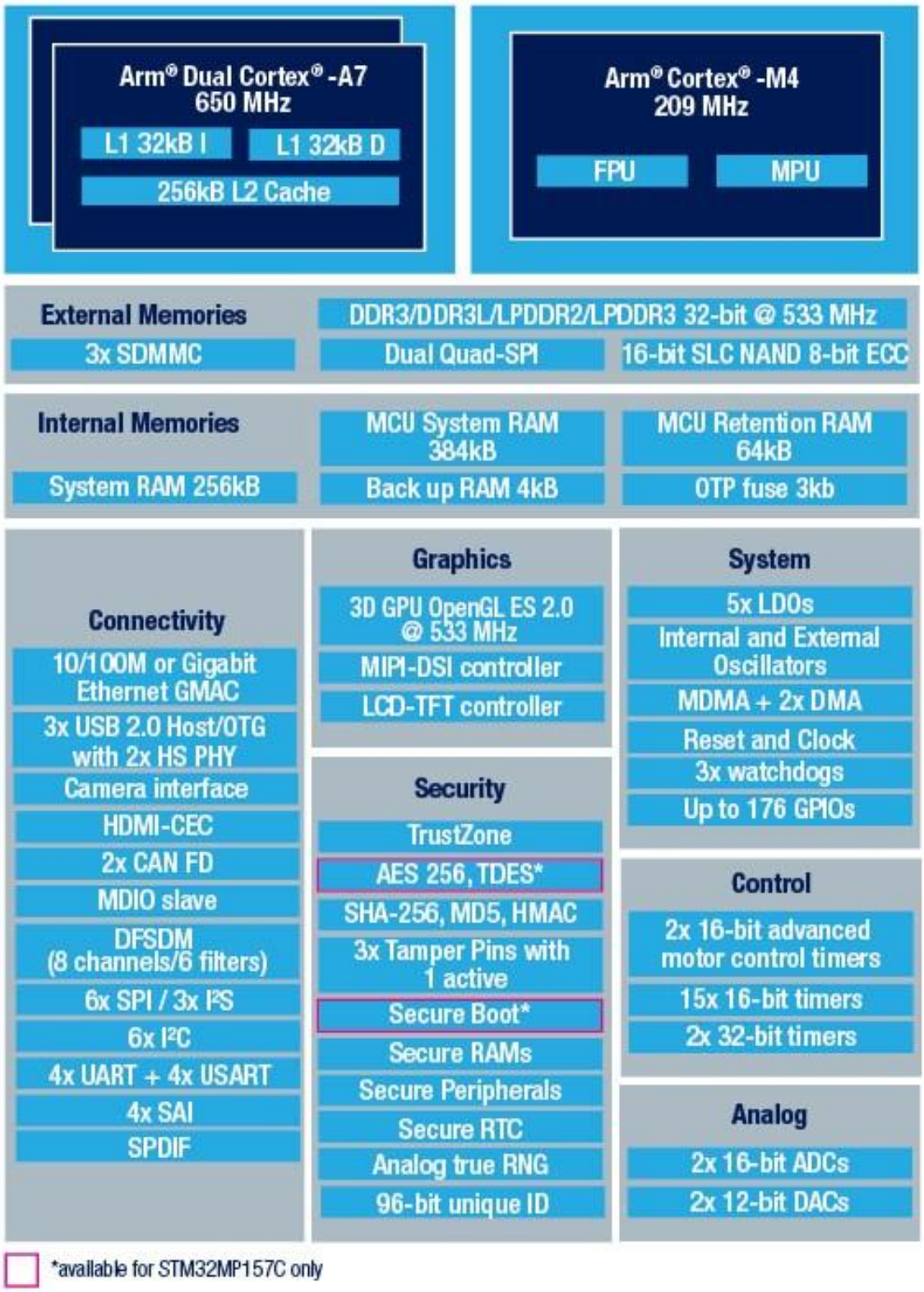



*Figura 2-7: Schema a blocchi del modulo STM32MP157*

### 2.2.5 STM32 Nucleo-H743ZI2

La STM32 Nucleo-H743ZI2 è una scheda prodotta dalla STMicroelectronics basata sul microcontrollore STM32H743II, MCU ad alte prestazioni con architettura ARM Cortex-M7. Nonostante il Cortex-M7 si tratti del MCU più performante della sua famiglia, come accade per molti prodotti che si basano unicamente sulla serie Cortex-M, le potenzialità di questo processore sono abbastanza ridotte a fronte di un consumo energetico estremamente limitato se paragonato alle altre piattaforme edge considerate. Infatti, se si escludono i consumi delle periferiche, la CPU richiede a pieno regime soli 275 µW/MHz che, alla frequenza di lavoro massima di 480 MHz e alimentazione a 3,3 V, risultano essere di circa 0,44 W. Da questo consegue che un prodotto di questo tipo è estremamente indicato in applicazioni IoT in cui la disponibilità di energia è notevolmente limitata ma allo stesso tempo sono richieste capacità di calcolo superiori alla media di un comune nodo sensore.

All'interno del microcontrollore STM32H743II è presente un singolo processore basato su architettura ARM Cortex-M7 a 32 bit, una memoria flash EPROM da 2 MB, su cui è possibile caricare il firmware, e una memoria SRAM da 1 MB, che comprende da una ITCM (Instruction Tightly Coupled Memory) da 64 kB e una DTCM (Data Tightly Coupled Memory) da 128 kB, memorie a bassa latenza per ridurre i ritardi nella fase di instruction fetch e nelle operazioni load/store. In aggiunta la CPU dispone di una cache di primo livello, con 16 kB riservati alle istruzioni e 16 kB per i dati. Il microcontrollore permette inoltre di fare uso di memorie aggiuntive tramite l'unità FMC (Flexible Memory Controller), grazie alla quale è possibile interfacciarsi in modo efficiente a memorie flash NAND, flash NOR, SRAM e SDRAM esterne al chip grazie a un bus con clock regolabile fino a 100 MHz.

Il microcontrollore integra inoltre 4 DMA controller che permettono di evitare il sovraccarico della CPU dovuto alle operazioni di trasferimento in memoria. Questa configurazione hardware permetter di raggiungere performance di 2,14 DMIPS/MHz fino a un massimo di 1027 DMIPS. Nonostante non vi siano GPU o hardware dedicato per le applicazioni video, il microcontrollore STM32H743 dispone anche di un acceleratore dedicato al codec JPEG e il Chrom-ART accelerator, un'unità finalizzata ad alleggerire la CPU dalle operazioni grafiche e dai conseguenti trasferimenti in memoria con un throughput di un pixel a ciclo. Le funzionalità del sistema sono arricchite anche da

un'unità per il calcolo a virgola mobile a doppia precisione (FPU a 64 bit) e la compatibilità della CPU con le istruzioni DSP, permettendo migliori prestazioni in tempo reale e una maggiore velocità di esecuzione nell'elaborazione dei segnali. Le librerie CMSIS (Cortex-M Software Interface Standard) permettono di sfruttare al meglio queste unità garantendo una compilazione ottimizzata in grado di impiegare al meglio i moduli hardware disponibili. All'interno del microcontrollore sono presenti anche 2 DAC a 12 bit, 3 ADC a 16 bit, 10 timer a 16 bit, 2 timer a 32 bit e 2 controller PWM a 16 bit che permettono di interagire con segnali analogici esterni. Tra le interfacce digitali seriali sono invece presenti 4 USART, 5 UART di cui una a basso consumo, 6 SPI, 4 $I^2C$ e un controller USB 2.0 OTG.

La scheda, oltre al microcontrollore, integra il modulo ST-LINK-V3 che permette di programmare il sistema e svolgere operazioni di debug supportando JTAG e SWD senza bisogno di ricorrere a componenti esterni, ma semplicemente connettendo tramite il connettore micro USB un computer che disponga di un ambiente di sviluppo supportato, come IAR, Keil o altri IDE basati sul compilatore GCC. In alternativa è possibile connettersi alla scheda mediante un apposito modulo di programmazione e debug interfacciato tramite connettore CN5 MIPI-10.

Parti sostanziali della connettività della Nucleo-H743ZI2 e che ne aumentano notevolmente la versatilità sono il connettore morpho expansion, che offre la possibilità di espandere la scheda con altri moduli accessori della STMicroelectronics, il connettore ST Zio, che permette di collegare alla scheda un Arduino Uno R3, e la presenza di connessione Ethernet tramite interfaccia fisica RJ45. La scheda dispone inoltre di 144 contatti GPIO configurabili, tramite i quali si può alimentare la scheda se non si vuole utilizzare la USB o il modulo ST-LINK-V3.


*Figura 2-8: Schema a blocchi del microcontrollore STM32H743II*

### 2.2.6 Confronto tra i dispositivi edge considerati

Considerata la notevole varietà di schede prese in esame, è difficile paragonarle senza introdurle in uno specifico contesto applicativo, dal momento che ciascun dispositivo risulta pensato per una categoria di applicazioni diversa, con conseguenti peculiarità che lo differenziano da tutti gli altri. Volendo però classificare le piattaforme a seconda delle capacità computazionali, dei moduli hardware disponibili e del prezzo si possono identificare tre principali famiglie di dispositivi: le schede con maggiore disponibilità di risorse e specifiche per l'esecuzione di algoritmi di intelligenza artificiale (Nvidia Jetson Nano e Google Coral), schede general purpose sprovviste di acceleratori specifici ma comunque molto prestanti (Raspberry Pi 4 e STM32MP157A-DK1) e schede a minori prestazioni e minori consumi, basate su un microcontrollore di fascia alta, come la STM32 Nucleo-H743ZI2.

La prima classe di schede comprende la Nvidia Jetson Nano e la Google Coral, entrambe soluzioni con un hardware dedicato per l'elaborazione video e l'implementazione di modelli di AI, supportato da un abbondante quantitativo di memoria RAM e di archiviazione. Entrambe le schede fanno uso di un sistema operativo basato su kernel Linux e limitano i consumi massimi a 10 W. Questa tipologia di schede richiede una spesa pari o perfino superiore a 100 $ e rappresentano il vertice per i dispositivi edge ad alte prestazioni, comunque limitando i consumi. Nonostante la Google Coral risulti più completa a livello di acceleratori e connettività, la principale differenza tra le due piattaforme è come vengono implementati gli algoritmi di intelligenza artificiale: la Nvidia Jetson Nano impiega una GPU a 128 core Maxwell mentre, sebbene provvista di una GPU dedicata, la Google Coral fa uso del coprocessore Edge TPU ML. Questi due diversi approcci saranno da tenere presente nella fase seguente, in cui le schede verranno testate, in particolare considerando che questa differenza hardware si riflette in una differenza a livello di framework impiegati.

Nel secondo gruppo sono presenti due schede meno performanti e ottimizzate delle precedenti ma che comunque, grazie ai prezzi più contenuti e alle buone prestazioni, permettono di raggiungere risultati interessanti, sufficienti per buona parte delle applicazioni. La Raspberry Pi 4 e la STM32MP157A-DK1 sono entrambe schede basate sistema operativo Linux, con consumi massimi di 15 W e con prezzi più moderati rispetto alle precedenti: 35 $ per la prima e 65 $ per la seconda. La scheda della

STMicroelectronics possiede una GPU dedicata e un sistema a doppio processore, mentre la Raspberry Pi, d'altra parte, vanta una CPU con frequenza maggiore, una memoria RAM più capiente e veloce, una migliore connettività e acceleratori dedicati per la codifica e la decodifica video.

L'ultima categoria di dispositivi è composta dalla sola STM32 Nucleo-H743ZI2 e si tratta di quella a minori prestazioni, minor costo e minori consumi, risultando meno versatile delle precedenti ma anche la più conveniente in parecchi casi in cui non sono necessarie risorse elevate. In questa fascia non si fa uso di sistemi operativi Linux ma si programma direttamente mediante linguaggio C e la memoria disponibile come la velocità del processore sono nettamente ridotte rispetto alle altre schede. L'hardware impiegato si avvicina a quello di un nodo sensore ad alte prestazioni, pertanto le funzionalità sono estremamente scarse a fronte di una migliore resa energetica che permette alla scheda di avere consumi distintamente inferiori alle altre soluzioni.

È importante ricordare che ogni insieme ha i propri punti di forza e, nel valutare le prestazioni di una piattaforma, è indispensabile considerare la classe in cui è inserito, dimensionando le aspettative a seconda della categoria in cui il sistema si colloca. Proprio per questo non è possibile valutare in maniera assoluta le schede solamente per le prestazioni dimostrate nell'esecuzione di modelli di AI, ma è necessario considerare anche consumi, prezzo e versatilità in modo da avere un quadro più completo. Proprio per questo l'analisi successiva, che si focalizza sulle prestazioni dei dispositivi, cercherà comunque di fare riferimenti agli altri aspetti, anche se non testati, in modo da inquadrare in maniera più dettagliata le piattaforme considerate.

| | **Nvidia Jetson Nano Developer Kit** | **Google Coral Dev Board** | **Raspberry Pi 4** | **STM32MP 157A-DK1** | **STM32 Nucleo H743ZI2** |
|---|---|---|---|---|---|
| **CPU** | ARM Cortex-A57 quad-core @ 1,43 GHz | NXP i.MX 8M SoC (ARM Cortex-A53 dual-core @ 1,5 GHz + ARM Cortex-M4 @ 266 MHz) | Broadcom BCM2711 (ARM Cortex-A72 quad-core @ 1,5 GHz) | STM32MP 157 (ARM Cortex-A7 dual-core @ 650 MHz + ARM Cortex-M4 @ 209 MHz) | STM32H7 (ARM Cortex-M7 @ 480 MHz) |
| **GPU** | Nvidia Maxwell 128-core | Vivante GC7000Lite | - | 3D OpenGL ES @ 533 MHz | - |
| **Acceleratori** | - | Google Edge TPU ML coprocessor + cryptographic coprocessor | - | - | - |
| **RAM integrata** | - | 160 kB | - | 256 kB system + 448 kB MCU | 1 MB SRAM 32 bit |
| **RAM esterna** | 4 GB LPDDR4 64 bit @ 1600 MHz (25,6 GB/s) | 1 GB DDR4 32 bit @ 1600 MHz (12,8 GB/s) | 4GB LPDDR4 64 bit @ 3200 MHz (51,2 GB/s) | 512 MB DDR3L 16 bit @ 533 MHz (2,13 GB/s) | - |
| **ROM flash** | 16 GB eMMC | 128 kB Boot ROM + 8 GB eMMC | - | - | 2 MB flash EPROM |
| **VPU encode** | 4K @ 30 | 4K @ 60 | 1080p @ 30 | - | - |
| **VPU decode** | 4k @ 60 | 4K @ 60 | 4k @ 60 | - | - |
| **Sistema operativo** | L4T (Nvidia JetPack) | Mendel Linux | Raspbian | Open STLinux/ Debian | - |
| **Consumi massimi stimati** | 10 W | 10 W | 15 W | 15 W | 440 mW |
| **Prezzo** | 100 $ | 150 $ | 35 $ | 65 $ | 25 $ |

*Tabella 2-1: Riassunto delle principali caratteristiche delle piattaforme edge considerate*

## 2.3 CLOUD

Le piattaforme cloud rappresentano il vertice della catena di elaborazione del segnale nel paradigma di calcolo distribuito classico, presentando una enorme disponibilità di risorse e spazio di archiviazione. Solitamente questi servizi vengono offerti da compagnie che prendono il nome di cloud provider, che mettono a disposizione i propri data center e i propri servizi ad altre società o direttamente agli utenti finali. Disponendo di una quantità di memoria e capacità computazionali estremamente grandi e difficili da paragonare a causa della ripartizione a vari utenti, tali provider si distinguono prevalentemente per come le risorse vengono gestite, suddividendole tra i diversi client, e in base al livello di completezza delle piattaforme (IaaS, PaaS, SaaS) e dei servizi ad esse collegate.

Esistono tre tipologie principali modelli di implementazione del cloud computing:

- Cloud pubblico: l'intera infrastruttura è gestita da un provider che mette a disposizione le proprie risorse ai vari client. Rappresenta il modello che solitamente offre più prestazioni con minori costi e con maggiore flessibilità ma allo stesso tempo risulta esser il più vulnerabile ad attacchi informatici e furti di dati;
- Cloud privato: soluzione analoga alla precedente con la differenza che l'intera struttura è gestita unicamente dall'ente che utilizza il servizio. Spesso dispone di risorse inferiori e a un costo maggiore rispetto a quelle di un'infrastruttura a modello pubblico ma può far uso di hardware dedicato in base alla specifica applicazione e permette di raggiungere livelli di sicurezza superiori alla controparte pubblica;
- Cloud ibrido: si tratta di una soluzione che fa uso di entrambi i precedenti modelli di cloud, impiegando la flessibilità del cloud pubblico in caso di carichi di lavoro intensi ma allo stesso tempo fa uso di data center privati per archiviare dati sensibili. Spesso viene utilizzato questo modello nel community cloud, in cui più compagnie a cloud privato condividono parte dei loro data center in una rete di più società.

Ad oggi la maggioranza del cloud è rappresentata da quello ibrido, mentre sono sempre più rare le soluzioni private. L'impiego del solo cloud pubblico ha visto una crescita negli ultimi anni, risultando sempre più vantaggioso rispetto a quello privato.

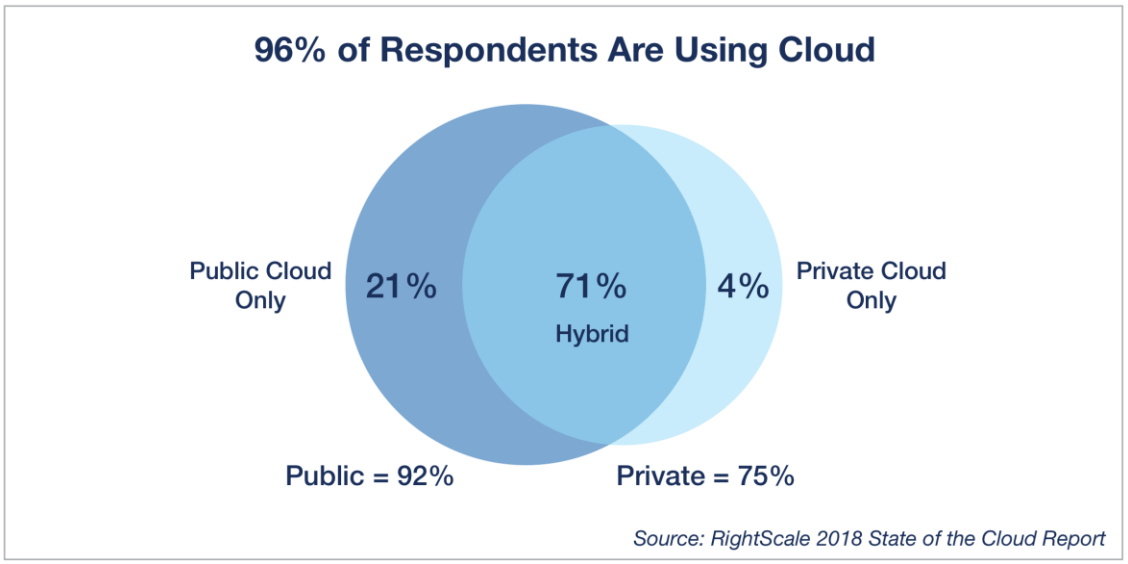



*Figura 2-9: Divisione dei metodi di implementazione del cloud computing [7]*

Il cloud che fa uso di data center pubblici, interamente o in parte, risulta essere quello più diffuso perché è presenta una maggiore elasticità nell'uso di risorse e minori costi, rivelandosi il metodo che meglio si adegua nella maggior parte dei casi al business dell'IoT. Proprio per questo è necessario capire come sono gestiti i carichi di lavoro all'interno di un data center pubblico, per comprendere le dinamiche tramite le quali si distribuisce il carico di lavoro. Infatti alla base della gestione di una piattaforma cloud pubblica è fondamentale il corretto impiego delle risorse presenti nei data center, dal momento che il costo finale dipende prevalentemente da questo aspetto. Proprio per questo le spese di acquisto, manutenzione e mantenimento dei server sono quelle che poi incidono maggiormente sul costo finale, pertanto è necessario sfruttare al meglio l'hardware disponibile. Per fare questo occorre dimensionare in maniera opportuna le risorse con le richieste, cercando di avere sempre un carico di lavoro commisurato alle potenzialità dei server, risultando l'unico modo per ammortizzare in maniera adeguata i costi del data center. Un aspetto fondamentale e caratterizzante per i vari cloud provider a modello pubblico è la modalità con cui vengono gestiti i carichi di lavoro e gli spazi di archiviazione, e di conseguenza le tariffe.

Di seguito sono riportati i principali cloud provider, fornendone una breve descrizione ed evidenziandone le principali caratteristiche:

- Amazon Web Services (AWS): si tratta di un provider che fornisce servizi di cloud computing on-demand per quanto riguarda elaborazione, archiviazione,

networking, database, analisi, strumenti di sviluppo e strumenti specifici per l'internet of things. Opera a livello IaaS, PaaS e SaaS e offre i propri servizi ad aziende e privati. Possiede data center in oltre 20 paesi e rappresenta il maggior leader nel settore del cloud computing, possedendo un terzo della quota di mercato totale. Tra i 175 servizi che offre più conosciuti sono l'Amazon Elastic Compute Cloud (EC2), per l'elaborazione dati, e l'Amazon Simple Storage Service (S3), per l'archiviazione;

- Microsoft Azure: è il servizio di cloud computing di Microsoft che fornisce servizi IaaS, PaaS e SaaS a società esterne e privati. Rappresenta il secondo maggior provider di servizi cloud dopo AWS e offre oltre 600 prodotti per l'elaborazione dati, l'archiviazione, lo sviluppo software e la programmazione. Tra le numerose funzionalità offerte vanno ricordate le macchine virtuali per l'elaborazione dati e l'Azure Storage per lo stoccaggio di dati;
- Google Cloud Platform (GCP): risulta essere il terzo cloud provider su scala globale e offre servizi di vario tipo IaaS, PaaS e SaaS. Offre oltre 100 prodotti che comprendono elaborazione dati, archiviazione, AI e machine learning, networking e strumenti per sviluppatori. Tra questi si ricordano il Google Compute Engine per l'elaborazione, il Google Cloud Storage per l'archiviazione e la G suite come SaaS;
- IBM Cloud: cloud provider che offre servizi ai tre livelli IaaS, PaaS e SaaS e in diversi campi tra cui elaborazione dati, archiviazione, networking, AI, IoT e strumenti di sviluppo. Tra i 170 prodotti offerti vanno ricordati i server di calcolo Bare Metal e l'IBM Cloud Storage che permette l'archiviazione dei dati.

| | Prodotti offerti | Elaborazione | Archiviazione |
|---|---|---|---|
| **Amazon Web Services (AWS)** | 175 | Amazon Elastic Compute Cloud (EC2) | Amazon Simple Storage Service (S3) |
| **Microsoft Azure** | 100 | Macchine Virtuali | Azure Storage |
| **Google Cloud Platform (GCP)** | 100 | Google Compute Engine | Google Cloud Storage |
| **IBM Cloud** | 170 | Bare Metal Server | IBM Cloud Storage |

*Tabella 2-2:Riassunto delle principali caratteristiche dei cloud provider considerati*

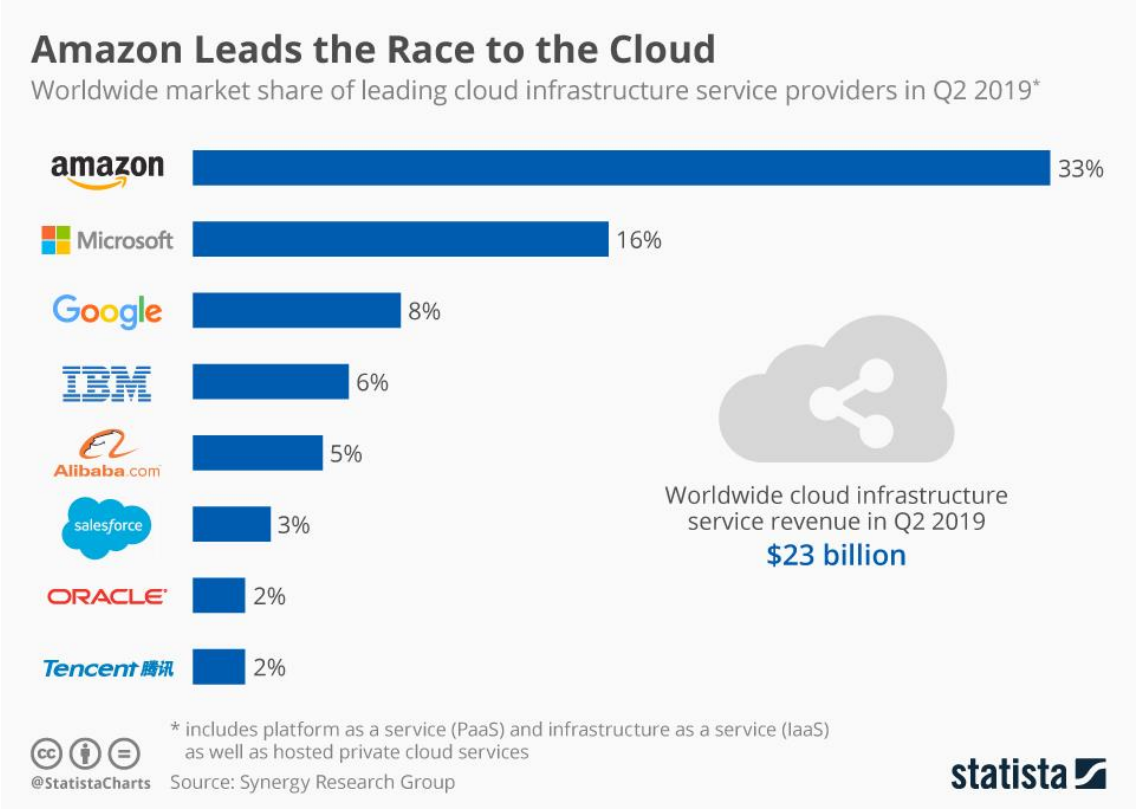



*Figura 2-10: Divisione del mercato tra i vari cloud provider [8]*

Vista la notevole varietà di servizi e soluzioni offerte dai cloud provider considerati è difficile paragonarli senza focalizzarsi su una specifica necessità, tuttavia è possibile confrontare alcune caratteristiche generali che permettono di differenziare le potenzialità di ciascuna compagnia. È importante notare come tutti i provider considerati supportino il modello a cloud ibrido, caratteristica non scontata che ha permesso di ampliare il mercato delle aziende. Un altro aspetto da considerare è che molti dei servizi offerti dalle compagnie sono delle declinazioni dei sistemi di calcolo e archiviazione di base con opportune ottimizzazioni e personalizzazioni a seconda del caso specifico. Questo rende ancora più difficile valutare i servizi offerti in generale dal momento che ciascuna funzionalità specifica può dimostrarsi più efficiente di altre nella particolare applicazione.

Se si considera AWS, esso rappresenta il provider più solido e maturo, e dispone del maggior numero di data center e di servizi offerti che ne assicurano un'ottima efficienza, elasticità e affidabilità. Proprio grazie a questo la rete di servizi Amazon è quella che ha una maggiore copertura a livello globale, anche se le tariffe finali sono medialmente le più alte.

Microsoft Azure ha come punto di forza, oltre ai prezzi ridotti rispetto a AWS, l'ottima integrazione con le tecnologie e gli strumenti di sviluppo Microsoft come Office e .NET che lo rende il cloud ideale per sviluppatori e programmatori di questa piattaforma. Nonostante questi aspetti, Azure non ha la stessa versatilità ed elasticità del competitor Amazon, a causa di una minore disponibilità di data center.

Google Cloud Platform risulta essere il cloud più economico soprattutto per carichi di lavoro intensi grazie ai Sustained Use Discount (SUD) e ai Committed Use Discount (CUD), per mezzo dei quali si abbassano le tariffe in caso di un uso prolungato e costante delle risorse o occupazione temporale precisa che permetta una migliore programmazione dello sforzo computazionale per ottimizzare il carico di lavoro. Anche gli altri provider fanno uso di sconti in caso di programmazione delle risorse occupate ma non raggiungono l'efficienza di Google. A livello di funzionalità GCP è analogo ad Azure, risultando meno completo e flessibile di AWS ma comunque mantenendo un buon grado di versatilità.

Il cloud di IBM rappresenta il meno consistente, a livello di divisione del mercato, dei provider trattati, nonostante offra un numero di servizi e configurazioni quasi al pari di AWS, superando quindi Azure e GCP. Come tariffe risulta allineato con il servizio di Microsoft, anche presenta una raggiungibilità globale superiore, quasi pari a quella offerta

da Amazon, nonostante disponga di meno data center. Nonostante questo, l'impiego di IBM cloud risulta essere quello meno user-friendly delle soluzioni trattate, riducendone la diffusione in buona parte delle applicazioni.

# 3 EDGE COMPUTING MEDIANTE RETI NEURALI

## 3.1 SCALABILITÀ DELLE RETI NEURALI SU PIATTAFORME EDGE

Sebbene l'addestramento di reti neurali sia stata da sempre un'attività ad alta richiesta di risorse che perfino tuttora sia quasi interamente affidata al cloud, la loro esecuzione è sempre più orientata ad hardware più scalati come le piattaforme edge o i nodi sensori. Ovviamente la possibilità di inserire reti neurali all'interno di hardware dalle risorse limitate dipende in gran parte dalla complessità del modello e dalle approssimazioni che si è disposti a fare; tuttavia i nodi edge moderni sono sempre più indicati allo svolgimento di algoritmi per l'intelligenza artificiale, in particolare le reti neurali.

È proprio dai noti vantaggi dell'edge computing e dall'esecuzione di algoritmi di intelligenza artificiale su queste piattaforme che nasce l'interesse per verificare le potenzialità di questi sistemi, esaminando quali sono i reali comportamenti in fase di esecuzione. Lo studio non può che trattarsi di un'attività empirica, dal momento che un'analisi a monte sarebbe estremamente complicata, sia per la quantità di parametri da considerare, sia per la mancanza di gran parte delle informazioni precise sulle architetture, volutamente non pubblicate dalle case costruttrici, evitando possibili imitazioni dalla concorrenza.

Alla luce di questo sono state scelte delle reti neurali di test, che simulino una possibile soluzione a un problema reale, per stabilire come le architetture considerate si comportino nell'esecuzione di tali algoritmi. Nonostante vi siano parecchie grandezze che possano risultare interessanti nell'esecuzione di reti neurali, come consumi e occupazione di memoria, l'unico parametro che verrà considerato approfonditamente sarà il tempo di esecuzione, in particolare la latenza del sistema, ovvero il tempo tra l'applicazione degli ingressi e l'ottenimento delle relative uscite. Questo è dovuto a diversi fattori. In primis, trattandosi di un sistema di edge computing, la latenza è il parametro più importante, per

il quale si predilige questo paradigma a quello del cloud computing. In secondo luogo l'analisi dei consumi risulta particolarmente difficoltosa in numerose situazioni e, inoltre, non è particolarmente significativa, visto che le piattaforme risultano avere specifiche sulla potenza dissipata teorica molto simili in gran parte dei casi. Infine lo studio della memoria occupata non viene fatto direttamente, dal momento che in gran parte dei casi considerati la memoria è sufficiente all'esecuzione dell'intera rete e, in aggiunta, una saturazione della memoria porterebbe a un aumento di latenza consistente o perfino all'impossibilità di eseguire il modello. Proprio per questo considerare come sola metrica i tempi di esecuzione della rete risulta sufficiente.

La scelta delle reti di test non poteva che essere mirata considerando l'applicazione finale, dal momento che esistono una moltitudine di classi di possibili reti neurali, e considerarle tutte sarebbe stato impensabile e inutile a livello applicativo. Proprio per questo si è scelto di focalizzarsi su reti feed-forward multistrato pienamente connesse (MLP), tipologia che poi verrà impiegata successivamente nell'algoritmo di ricostruzione di segnali sottoposti a compressed sensing. Questo ha dato modo di studiare meglio le variabili che impattano sulle risorse di calcolo richieste al sistema, vale a dire numero di nodi e profondità della rete. Infatti il primo parametro rappresenta il numero di operazioni MAC richieste ed è anche collegato alla quantità di memoria impiegata, mentre il secondo misura la capacità della piattaforma di eseguire efficientemente calcoli in cascata, riducendo il numero di accessi in memoria nei casi più ottimizzati. Proprio per questo si è deciso di variare questi due parametri in maniera più possibilmente indipendente, tentando di capire in che modo siano collegati al tempo finale di esecuzione.

Le reti considerate si possono dividere in due gruppi: la prima tipologia, caratterizzata da un singolo strato nascosto di dimensione crescente che permette di aumentare il numero di operazioni MAC senza modificare il numero di livelli, e la seconda, contraddistinta da strati nascosti di numero crescente ma grandezza costante. In entrambi i casi si sono mantenute costanti le dimensioni di ingresso e uscita della rete e pari a 100 nodi. La dimensione dello strato nascosto della prima tipologia di reti varia da 100 a 20000, mentre per la seconda categoria sono state fissate le dimensioni di 100 e 1000 nodi in un numero di strati nascosti variabile da 1 a 10.

Come funzione di attivazione è stata scelta per tutti gli strati una softmax, particolare funzione molto utilizzata nelle reti neurali che ha la proprietà di comprimere un vettore in modo che la somma delle sue componenti sia unitaria:

$$\begin{cases} \sigma: \mathbb{R}^K \to \mathbb{R}^K \\ \sigma(\boldsymbol{z})_i = \dfrac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \ per\ i = 1, \dots, K\ e\ \boldsymbol{z} = (z_1, \dots, z_K) \in \mathbb{R}^K \end{cases}$$

Questa scelta è giustificata dal fatto che la softmax è una funzione molto utilizzata nelle classificazioni non binarie e ha una complessità maggiore rispetto a rettificatori e funzioni sigmoidee, che presentano solo valori positivi di uscita, rischiando di semplificare eccessivamente l'aritmetica nell'implementazione. Di seguito è riportata la tabella con le reti neurali di test utilizzate in cui sono riassunte le principali caratteristiche di ciascun modello:

| Nome Rete | Dimensione ingresso | Dimensione uscita | Numero strati nascosti | Nodi strato nascosto | Totale pesi e MAC |
|---|---|---|---|---|---|
| 1-S | 100 | 100 | 1 | 100 | 20200 |
| 1.1-S | 100 | 100 | 1 | 150 | 30250 |
| 1.2-S | 100 | 100 | 1 | 200 | 40300 |
| 2-S | 100 | 100 | 1 | 250 | 50350 |
| 2.1-S | 100 | 100 | 1 | 300 | 60400 |
| 2.2-S | 100 | 100 | 1 | 350 | 70450 |
| 2.3-S | 100 | 100 | 1 | 400 | 80500 |
| 2.4-S | 100 | 100 | 1 | 450 | 90550 |
| 3-S | 100 | 100 | 1 | 500 | 100600 |
| 3.1-S | 100 | 100 | 1 | 550 | 110650 |
| 3.2-S | 100 | 100 | 1 | 600 | 120700 |
| 4-S | 100 | 100 | 1 | 750 | 150850 |
| 5-S | 100 | 100 | 1 | 1000 | 201100 |
| 6-S | 100 | 100 | 1 | 1250 | 251350 |
| 7-S | 100 | 100 | 1 | 1500 | 301600 |
| 8-S | 100 | 100 | 1 | 1750 | 351850 |
| 9-S | 100 | 100 | 1 | 2000 | 402100 |
| 10-S | 100 | 100 | 1 | 2250 | 452350 |
| 11-S | 100 | 100 | 1 | 2500 | 502600 |
| 12-S | 100 | 100 | 1 | 2750 | 552850 |
| 13-S | 100 | 100 | 1 | 3000 | 603100 |
| 14-S | 100 | 100 | 1 | 3250 | 653350 |
| 15-S | 100 | 100 | 1 | 3500 | 703600 |
| 16-S | 100 | 100 | 1 | 3750 | 753850 |
| 17-S | 100 | 100 | 1 | 4000 | 804100 |
| 18-S | 100 | 100 | 1 | 4250 | 854350 |
| 19-S | 100 | 100 | 1 | 4500 | 904600 |
| 20-S | 100 | 100 | 1 | 4750 | 954850 |
| 21-S | 100 | 100 | 1 | 5000 | 1005100 |
| 22-S | 100 | 100 | 1 | 5250 | 1055350 |
| 23-S | 100 | 100 | 1 | 5500 | 1105600 |
| 24-S | 100 | 100 | 1 | 5750 | 1155850 |
| 25-S | 100 | 100 | 1 | 6000 | 1206100 |
| 26-S | 100 | 100 | 1 | 6250 | 1256350 |
| 27-S | 100 | 100 | 1 | 6500 | 1306600 |
| 28-S | 100 | 100 | 1 | 6750 | 1356850 |
| 29-S | 100 | 100 | 1 | 7000 | 1407100 |
| 30-S | 100 | 100 | 1 | 7250 | 1457350 |
| 31-S | 100 | 100 | 1 | 7500 | 1507600 |
| 32-S | 100 | 100 | 1 | 7750 | 1557850 |
| 33-S | 100 | 100 | 1 | 8000 | 1608100 |

| | | | | | |
|---|---|---|---|---|---|
| 34-S | 100 | 100 | 1 | 8250 | 1658350 |
| 35-S | 100 | 100 | 1 | 8500 | 1708600 |
| 36-S | 100 | 100 | 1 | 8750 | 1758850 |
| 37-S | 100 | 100 | 1 | 9000 | 1809100 |
| 38-S | 100 | 100 | 1 | 9250 | 1859350 |
| 39-S | 100 | 100 | 1 | 9500 | 1909600 |
| 40-S | 100 | 100 | 1 | 9750 | 1959850 |
| 41-S | 100 | 100 | 1 | 10000 | 2010100 |
| 42-S | 100 | 100 | 1 | 10250 | 2060350 |
| 43-S | 100 | 100 | 1 | 10500 | 2110600 |
| 44-S | 100 | 100 | 1 | 10750 | 2160850 |
| 45-S | 100 | 100 | 1 | 11000 | 2211100 |
| 46-S | 100 | 100 | 1 | 11250 | 2261350 |
| 47-S | 100 | 100 | 1 | 11500 | 2311600 |
| 48-S | 100 | 100 | 1 | 11750 | 2361850 |
| 49-S | 100 | 100 | 1 | 12000 | 2412100 |
| 50-S | 100 | 100 | 1 | 12250 | 2462350 |
| 51-S | 100 | 100 | 1 | 12500 | 2512600 |
| 52-S | 100 | 100 | 1 | 12750 | 2562850 |
| 53-S | 100 | 100 | 1 | 13000 | 2613100 |
| 54-S | 100 | 100 | 1 | 13250 | 2663350 |
| 55-S | 100 | 100 | 1 | 13500 | 2713600 |
| 56-S | 100 | 100 | 1 | 13750 | 2763850 |
| 57-S | 100 | 100 | 1 | 14000 | 2814100 |
| 58-S | 100 | 100 | 1 | 14250 | 2864350 |
| 59-S | 100 | 100 | 1 | 14500 | 2914600 |
| 60-S | 100 | 100 | 1 | 14750 | 2964850 |
| 61-S | 100 | 100 | 1 | 15000 | 3015100 |
| 62-S | 100 | 100 | 1 | 15250 | 3065350 |
| 63-S | 100 | 100 | 1 | 15500 | 3115600 |
| 64-S | 100 | 100 | 1 | 15750 | 3165850 |
| 65-S | 100 | 100 | 1 | 16000 | 3216100 |
| 66-S | 100 | 100 | 1 | 16250 | 3266350 |
| 67-S | 100 | 100 | 1 | 16500 | 3316600 |
| 68-S | 100 | 100 | 1 | 16750 | 3366850 |
| 69-S | 100 | 100 | 1 | 17000 | 3417100 |
| 70-S | 100 | 100 | 1 | 17250 | 3467350 |
| 71-S | 100 | 100 | 1 | 17500 | 3517600 |
| 72-S | 100 | 100 | 1 | 17750 | 3567850 |
| 73-S | 100 | 100 | 1 | 18000 | 3618100 |
| 74-S | 100 | 100 | 1 | 18250 | 3668350 |
| 75-S | 100 | 100 | 1 | 18500 | 3718600 |
| 76-S | 100 | 100 | 1 | 18750 | 3768850 |
| 77-S | 100 | 100 | 1 | 19000 | 3819100 |

| | | | | | |
|---|---|---|---|---|---|
| 78-S | 100 | 100 | 1 | 19250 | 3869350 |
| 79-S | 100 | 100 | 1 | 19500 | 3919600 |
| 80-S | 100 | 100 | 1 | 19750 | 3969850 |
| 81-S | 100 | 100 | 1 | 20000 | 4020100 |
| 1-M1 | 100 | 100 | 1 | 100 | 20200 |
| 2-M1 | 100 | 100 | 2 | 100 | 30300 |
| 3-M1 | 100 | 100 | 3 | 100 | 40400 |
| 4-M1 | 100 | 100 | 4 | 100 | 50500 |
| 5-M1 | 100 | 100 | 5 | 100 | 60600 |
| 6-M1 | 100 | 100 | 6 | 100 | 70700 |
| 7-M1 | 100 | 100 | 7 | 100 | 80800 |
| 8-M1 | 100 | 100 | 8 | 100 | 90900 |
| 9-M1 | 100 | 100 | 9 | 100 | 101000 |
| 10-M1 | 100 | 100 | 10 | 100 | 111100 |
| 1-M2 | 100 | 100 | 1 | 1000 | 201100 |
| 2-M2 | 100 | 100 | 2 | 1000 | 1202100 |
| 3-M2 | 100 | 100 | 3 | 1000 | 2203100 |
| 4-M2 | 100 | 100 | 4 | 1000 | 3204100 |
| 5-M2 | 100 | 100 | 5 | 1000 | 4205100 |
| 6-M2 | 100 | 100 | 6 | 1000 | 5206100 |
| 7-M2 | 100 | 100 | 7 | 1000 | 6207100 |
| 8-M2 | 100 | 100 | 8 | 1000 | 7208100 |
| 9-M2 | 100 | 100 | 9 | 1000 | 8209100 |
| 10-M2 | 100 | 100 | 10 | 1000 | 9210100 |

*Tabella 3-1: Proprietà delle reti utilizzate per il test*

È importante ricordare che le reti utilizzate non sono state allenate ma solamente inizializzate tramite valori casuali, pertanto non è richiesto di definire una funzione errore o altri parametri che ne regolino l'apprendimento.

Un altro aspetto da considerare è che le dimensioni delle reti di test sono state volutamente ingrandite ben oltre le alle misure di una comune rete neurale, dal momento che in diverse soluzioni reali il numero di nodi e la profondità è notevolmente più piccolo, come sarà il caso applicativo successivo. Questa scelta è stata fatta appositamente per mettere in difficoltà le piattaforme testate, spingendole al limite delle loro prestazioni, in modo da valutarne al meglio le capacità di parallelismo ed individuare eventuali limiti dovuti all'hardware.

Tali reti sono state definite mediante il pacchetto Keras del linguaggio Python, integrato all'interno del framework TensorFlow Core dalla versione 2.0, grazie al quale è stato

possibile esportare in maniera semplice ed automatizzata i modelli. Il formato con cui Keras esporta i propri modelli contiene all'interno la struttura della rete, le funzioni di attivazione e i pesi in aritmetica a virgola mobile in singola precisione (con estensione .h5). Tuttavia non tutte le piattaforme edge sono in grado di utilizzare questo formato, date le grandi dimensioni e la necessità di essere gestito ad alto livello e tradotto. Proprio per questo, contestualmente alla generazione delle reti, sono stati eseguiti alcuni passaggi per la conversione ad altri formati. Prima di tutto si è operata una esportazione in formato TensorFlow Lite (con estensione .tflite), specifico per piattaforme mobile e basato sulla piattaforma FloatBuffers, permettendo di passare da una descrizione ad alto livello della rete a una serie di operazioni matriciali, non richiedendo più l'impiego di Keras.

Oltre a questo, per la compatibilità con la Google Coral Dev Board, e per completezza nel confronto anche per la Raspberry Pi 4 e la STM32MP157A-DK1, è stato necessario impiegare un'ulteriore procedura di post-training quantization, con la quale si quantizzano i modelli TensorFlow Lite. In particolare la quantizzazione richiesta è stata la full-integer quantization, in cui si riduceva l'aritmetica delle operazioni a 8 bit, sia per i pesi salvati sia per le funzioni di attivazione. Queste semplificazioni successive hanno portato con loro un notevole risparmio di dimensioni dei modelli, dovuti alla riduzione della precisione degli stessi. Quantitativamente parlando si tratta di una diminuzione di circa 3 volte nel passaggio da un modello TensorFlow Core/Keras a uno TensorFlow Lite, e un'ulteriore riduzione di quasi 4 volte nella quantizzazione successiva. Questa modalità permette di risparmiare quasi 12 volte nelle dimensioni del modello, oltre a farlo risultare notevolmente più leggero in fase di esecuzione, riducendo sia le dimensioni delle variabili in memoria, sia la complessità delle operazioni richieste.

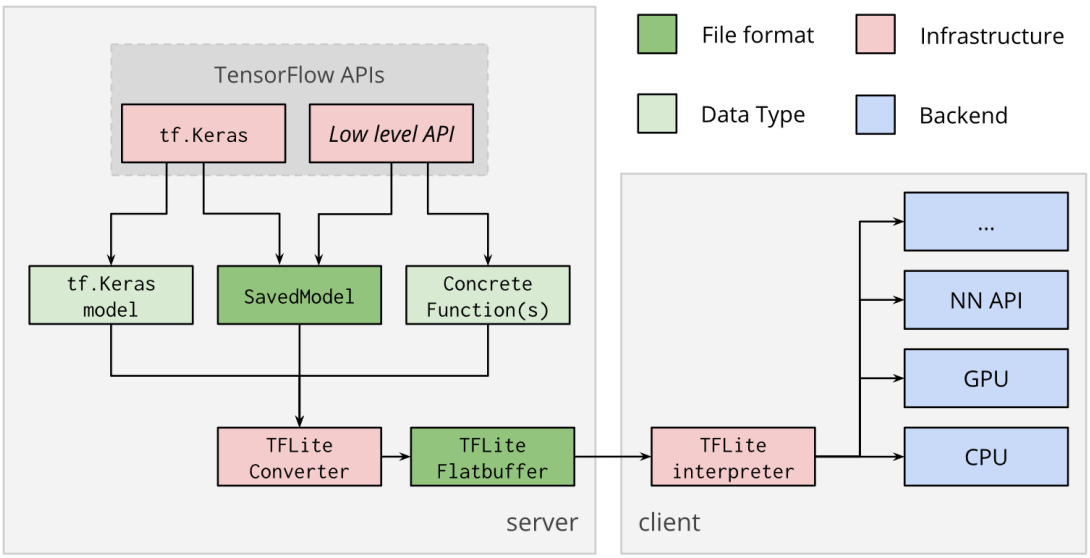



*Figura 3-1: Diagramma di flusso per la conversione in un modello TensorFlow Lite*

Questa procedura tuttavia riduce notevolmente anche la risoluzione numerica della rete, che potrebbe pertanto perdere di efficienza nelle predizioni. Proprio per questo esiste la possibilità di quantizzare il modello durante l'allenamento, permettendo di ottimizzare la procedura di quantizzazione adattando le soglie in fase di apprendimento, massimizzandone l'efficacia (quantization aware training). Questa via prevede prima l'esportazione della rete in un grafico contenente i pesi e le funzioni di attivazione quantizzate (estensione .pb) per poi convertirlo in un modello TensorFlow Lite. Nel caso specifico si è scelto di non adottare questa procedura dal momento che non interessa la precisione del modello ma solamente quanto questa scelta impatti sulle tempistiche, lasciando le valutazioni sull'impatto della quantizzazione ai casi specifici di studio.

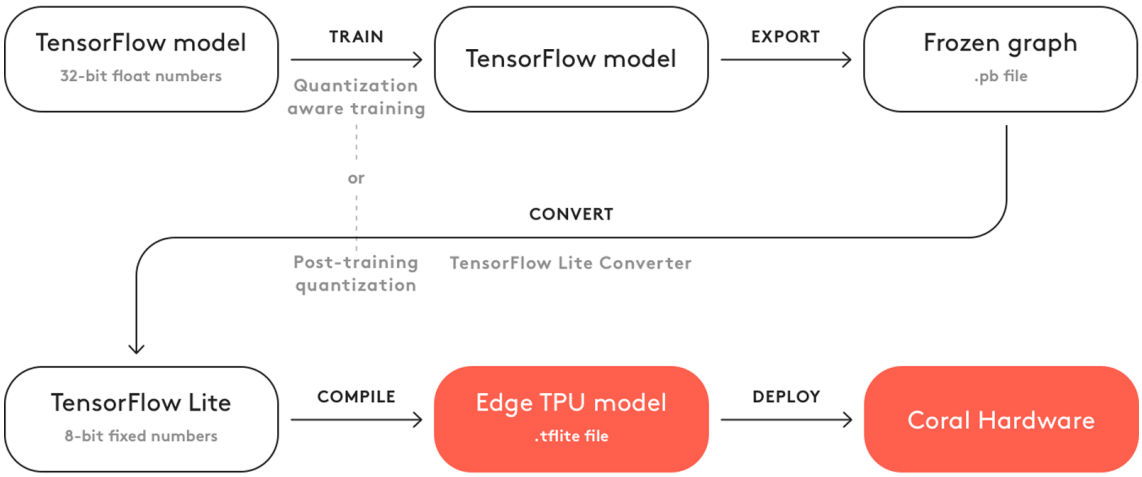



*Figura 3-2: Processo di addestramento e conversione a un modello quantizzato*

### 3.1.1 Setup delle piattaforme

Una volta definite ed esportate le reti nel corretto formato è stato possibile copiarle all'interno delle piattaforme considerate per poterle eseguire. Prima di fare questo però si è dimostrata necessaria per ogni scheda una fase di inizializzazione, installando su ciascuna piattaforma i framework, i pacchetti e le librerie necessarie per avere un'esecuzione massimamente ottimizzata sull'hardware a disposizione.

Delle cinque schede considerate, quattro si basano su un sistema a kernel Linux. Pertanto la prima fase di inizializzazione prevedeva l'installazione del sistema operativo su ciascuna piattaforma. Per la Google Coral, la Nvidia Jetson Nano e la Raspberry Pi 4 sono stati utilizzati i sistemi operativi forniti dalle case costruttrici, vale a dire Mendel Linux, L4T e Raspbian. Diversamente per la STM32MP157A-DK1 si è scelto di adottare una distribuzione di Linux Debian Buster adattata appositamente per la scheda. Questo deriva dal fatto che, diversamente dagli altri sistemi operativi sopra citati, OpenSTLinux risultava estremamente scarno di componenti necessari all'esecuzione delle reti e la loro installazione sarebbe risultata più difficoltosa che adottare una diversa distribuzione che presentasse lo stesso livello di ottimizzazione ma più completezza. Proprio grazie a questa scelta su nessuna scheda è stato necessario installare l'interprete comandi per Python 3 e il pacchetto Numpy per il calcolo vettoriale, dal momento che erano già presenti nel sistema.

La Nvidia Jetson Nano Developer Kit, dopo aver appositamente scritto L4T all'interno della scheda microSD, presenta l'interprete Python in versione 3.6.9 e dispone di interfaccia grafica molto simile a un computer desktop, grazie alla quale è risultato più semplice installare i pacchetti mancanti. In particolare per l'esecuzione della rete e stato richiesto di aggiungere alcune librerie e pacchetti necessari alla corretta esecuzione di TensorFlow GPU in versione 2.0.0. L'impiego di quest'ultimo ha permesso di parallelizzare al massimo il calcolo sulla Nvidia Maxwell a 128 core e allo stesso tempo di non dover installare Keras, dal momento che risulta già integrato in TensorFlow GPU.

In maniera analoga alla Nvidia Jetson Nano è stato caricato Raspbian all'interno della microSD della Raspberry Pi 4, verificando poi che anch'esso disponeva di interprete Python nella più recente versione 3.7.3. Vista la necessità di dover utilizzare più framework su questa scheda come metro di paragone, è risultato vantaggioso installare Virtualenv, un software che permette di creare dei contenitori, chiamati ambienti, di librerie e pacchetti in versioni diverse senza che essi vadano in conflitto. In questo modo è stato possibile installare in due ambienti differenti, uno con TensorFlow Core 1.14 affiancato da Keras 2.3.1, l'altro provvisto TensorFlow Lite 2.1.0. Proprio grazie a questa separazione è stato possibile avere un ambiente ottimizzato per l'esecuzione di modelli Keras (.h5) e un altro idoneo ad eseguire modelli TensorFlow Lite (.tflite) con dovute ottimizzazioni dell'interprete per l'hardware della Raspberry Pi 4. In realtà lo stesso TensorFlow Core possiede un interprete per i modelli TensorFlow Lite ma l'ottimizzazione e l'impiego di risorse è completamente diverso, pertanto per valutarne le prestazioni in maniera adeguata è stato necessario installare entrambi i framework.

La Google Coral ha una procedura di installazione di Mendel Linux più complessa rispetto alle schede precedenti, necessitando di una scrittura della microSD tramite la scheda di sviluppo stessa connessa a un computer con sistema operativo Linux e presentando una interfaccia solamente a riga di comando per mezzo di una connessione seriale. Una volta installato il sistema, già provvisto di interprete Python 3.7.3, è stato necessario aggiungere una versione appositamente ottimizzata di TensorFlow Lite 2.1.0, in modo da supportare il particolare hardware della scheda. Oltre a questo, per ottenere le massime prestazioni dalla Google Coral, è stato necessario utilizzare un ulteriore software di ottimizzazione dei modelli, di nome Edge TPU Compiler, che premettesse di suddividere il codice in base alle operazioni richieste e alla compatibilità con l'hardware. In particolare il software, già presente all'interno della scheda, si fa carico

dell'ottimizzazione del codice presente in un modello TensorFlow Lite completamente quantizzato, decidendo quali istruzioni debbano essere eseguite sulla TPU e quali sulla CPU, prediligendo quest'ultima solo in caso le operazioni non siano supportate dall'acceleratore tensoriale. Nel caso specifico delle reti considerate, la TPU riesce ad eseguire senza problemi tutte le moltiplicazioni matriciali, quindi a implementare bene la parte lineare della rete, mentre non supporta la funzione di attivazione softmax, come avviene per tante altre funzioni di attivazione, il cui calcolo è demandato alla CPU. Oltre a questo il processore si deve fare carico anche della quantizzazione degli ingressi e dalla dequantizzazione delle uscite, overhead che diversamente dai precedenti rimangono costanti allo scalare della rete.

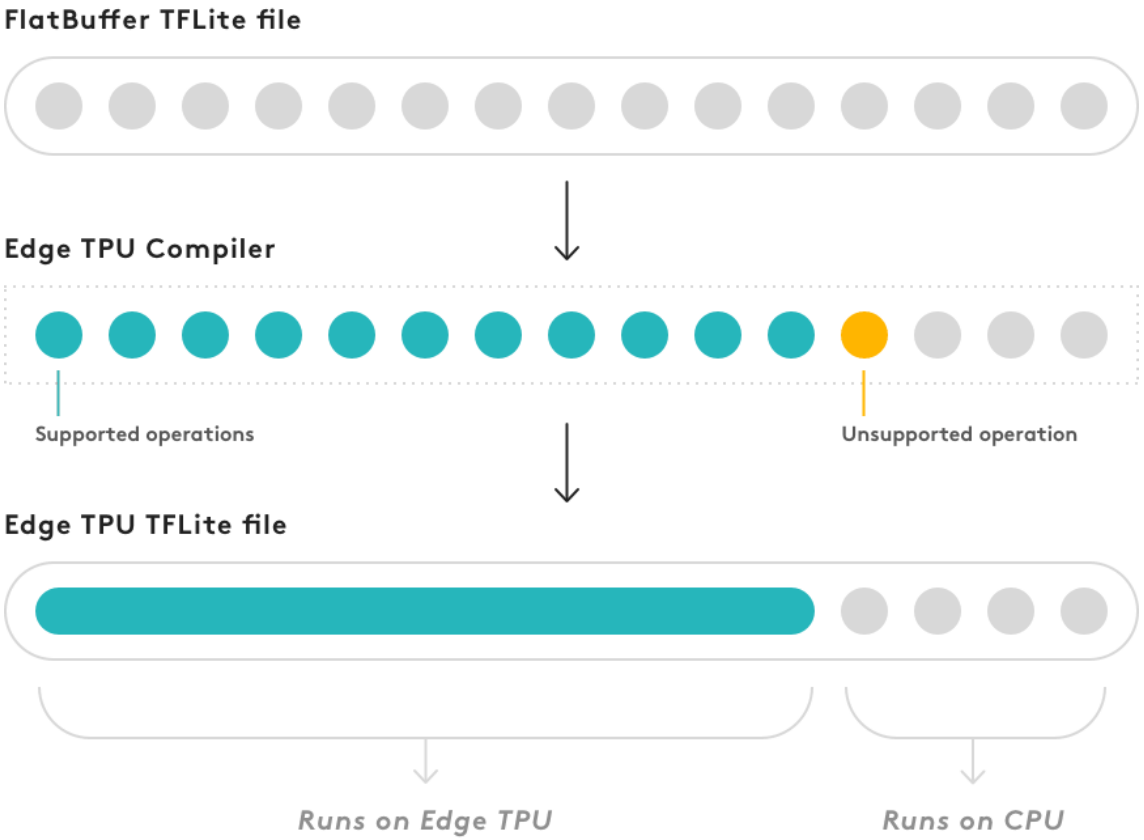



*Figura 3-3: Ottimizzazione di un modello TensorFlow Lite quantizzato per la Google Coral Dev Board*

In maniera analoga alla Raspberry Pi 4 e alla Nvidia Jetson Nano e stato possibile scrivere Linux Debian sulla STM32MP157A-DK1. Tale sistema, pur non essendo provvisto di interfaccia grafica, possiede l'interprete Python in versione 3.7.3. A questo, date le prestazioni della scheda e le necessità di confronto, è stato aggiunto solamente l'interprete TensorFlow Lite 2.1.0 e le librerie necessarie al suo funzionamento, nonostante non

avesse particolari ottimizzatori che ne migliorassero le prestazioni, diversamente dalle altre due schede che fanno uso di modelli quantizzati.

Infine l'impiego della STM32 Nucleo-H743ZI2 come piattaforma edge per l'utilizzo di reti neurali risulta estremamente vincolato alle scelte di programmazione e al livello di ottimizzazione richiesto. Infatti diversamente dalle altre schede, questo microcontrollore non dispone né di un sistema operativo, né del pacchetto TensorFlow, motivo per cui non è facile confrontarla con le altre soluzioni. Il metodo che inizialmente può sembrare più intuitivo è quello di utilizzare uno strumento apposito per l'intelligenza artificiale di STMicroelectronics che permette di tradurre un modello Keras in codice C direttamente compilabile per il microcontrollore, tuttavia il programma non risulta particolarmente ottimizzato. Proprio per questo si è deciso di scrivere un codice C partendo da zero che implementasse le reti neurali scelte, in modo da ottimizzare l'esecuzione utilizzando le librerie CMSIS, che permettono di ottenere un notevole incremento delle prestazioni grazie a un migliore impiego delle risorse hardware.

### 3.1.2 Esecuzione delle reti neurali

Dopo aver opportunamente impostato le schede per l'esecuzione di reti neurali è stato possibile eseguire un algoritmo che permettesse di ottenere una misura delle tempistiche necessarie all'esecuzione di questi modelli. In particolare, per le schede provviste di sistema operativo, è stato preparato uno script Python che permettesse di automatizzare il processo di acquisizione dei dati per ogni rete, salvando le misure in un file testuale. In particolare tale procedimento può essere riassunto nei seguenti passaggi:

1. Caricamento dei pacchetti richiesti;
2. Inizializzazione di 100 vettori casuali;
3. Caricamento della rete neurale;
4. Predizione dei tutti i vettori di uscita misurando il tempo necessario per le 100 predizioni;
5. Ripetizione del punto 4 per 12 volte, eliminando l'iterazione dal risultato maggiore e quella dal risultato minore;
6. Calcolo della media delle 10 misure rimanenti e salvataggio del valore nel file di uscita;

7. Ripetizione dello stesso algoritmo dal punto 3 per ogni differente rete neurale considerata.

Grazie a questo procedimento sono stati trovati i tempi medi per 100 previsioni di ciascuna rete neurale. Il numero di 100 vettori di ingresso e altrettanti vettori d'uscita serve per rendere i tempi di latenza sufficientemente grandi da essere misurati senza risentire troppo della risoluzione temporale del sistema operativo e dell'interprete comandi Python, ma contemporaneamente senza impiegare troppo tempo nel processo di misura. L'impiego di 12 iterazioni invece serve per ridurre la varianza della misura, dal momento che il sistema operativo rende i tempi di esecuzione aleatori. Diversamente lo scartare la misura maggiore e minore serve per eliminare tempi troppo brevi o troppo lunghi dovuti a cause ricorrenti nel sistema, come i frequenti overhead che si verificano sulle prime iterazioni a causa del caricamento dalla memoria dei valori.

Un altro appunto va fatto sulle differenze tra TensorFlow GPU o TensorFlow Core e TensorFlow Lite. Infatti nei primi due pacchetti è possibile eseguire la predizione di un'intera matrice di ingresso mentre nella versione Lite è concesso solamente di ricavare la predizione di un singolo vettore alla volta. Questo fatto permette di ottimizzare notevolmente il parallelismo nei primi due casi, risultando non più paragonabili con le versioni alleggerite e quantizzate. Proprio per questo si è deciso, anche in un'ottica di implementazione in un sistema reale di predizione, di elaborare un singolo vettore alla volta anche nei casi in cui il framework supporti la predizione matriciale, permettendo di paragonare le diverse piattaforme edge.

Di seguito sono riportati in forma tabellare i risultati ottenuti dall'esecuzione di 100 predizioni su ciascuna scheda provvista di sistema operativo Linux, prestando particolare attenzione che durante l'esecuzione nessuna delle piattaforme saturasse la memoria:

| Nome Rete | Totale pesi e MAC | Nvidia Jetson Nano (TF) [ms] | Raspberry Pi 4 (TF) [ms] | Raspberry Pi 4 (TFL) [ms] | Google Coral (TFL) [ms] | STM32 MP157A-DK1 (TFL) [ms] |
|---|---|---|---|---|---|---|
| 1-S | 20200 | 365 | 138 | 3 | 36 | 14 |
| 2-S | 50350 | 309 | 153 | 5 | 30 | 24 |
| 3-S | 100600 | 301 | 170 | 7 | 31 | 40 |
| 4-S | 150850 | 302 | 186 | 10 | 28 | 59 |
| 5-S | 201100 | 304 | 196 | 12 | 35 | 78 |
| 6-S | 251350 | 343 | 210 | 15 | 33 | 99 |
| 7-S | 301600 | 328 | 221 | 18 | 35 | 119 |
| 8-S | 351850 | 340 | 226 | 21 | 33 | 139 |
| 9-S | 402100 | 345 | 297 | 23 | 34 | 161 |
| 10-S | 452350 | 371 | 254 | 28 | 32 | 178 |
| 11-S | 502600 | 378 | 263 | 30 | 40 | 206 |
| 12-S | 552850 | 380 | 270 | 32 | 35 | 227 |
| 13-S | 603100 | 382 | 283 | 36 | 36 | 233 |
| 14-S | 653350 | 412 | 288 | 38 | 33 | 249 |
| 15-S | 703600 | 425 | 306 | 40 | 33 | 267 |
| 16-S | 753850 | 436 | 312 | 43 | 35 | 285 |
| 17-S | 804100 | 446 | 404 | 47 | 35 | 314 |
| 18-S | 854350 | 486 | 334 | 50 | 35 | 327 |
| 19-S | 904600 | 491 | 338 | 53 | 35 | 337 |
| 20-S | 954850 | 528 | 342 | 55 | 35 | 356 |
| 21-S | 1005100 | 510 | 362 | 58 | 34 | 378 |
| 22-S | 1055350 | 560 | 366 | 61 | 35 | 430 |
| 23-S | 1105600 | 534 | 374 | 63 | 35 | 450 |
| 24-S | 1155850 | 551 | 386 | 66 | 34 | 428 |
| 25-S | 1206100 | 541 | 538 | 68 | 36 | 445 |
| 26-S | 1256350 | 525 | 409 | 70 | 35 | 463 |
| 27-S | 1306600 | 484 | 417 | 74 | 35 | 473 |
| 28-S | 1356850 | 511 | 420 | 76 | 32 | 496 |
| 29-S | 1407100 | 496 | 441 | 80 | 36 | 523 |
| 30-S | 1457350 | 542 | 444 | 81 | 35 | 541 |
| 31-S | 1507600 | 545 | 438 | 84 | 36 | 548 |
| 32-S | 1557850 | 570 | 473 | 86 | 36 | 598 |
| 33-S | 1608100 | 522 | 626 | 90 | 36 | 650 |
| 34-S | 1658350 | 614 | 523 | 96 | 322 | 665 |
| 35-S | 1708600 | 607 | 528 | 96 | 332 | 633 |
| 36-S | 1758850 | 625 | 518 | 97 | 347 | 636 |
| 37-S | 1809100 | 610 | 516 | 100 | 363 | 660 |
| 38-S | 1859350 | 653 | 531 | 103 | 373 | 671 |
| 39-S | 1909600 | 685 | 517 | 104 | 375 | 685 |
| 40-S | 1959850 | 705 | 548 | 108 | 384 | 705 |
| 41-S | 2010100 | 660 | 778 | 110 | 405 | 722 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 42-S | 2060350 | 717 | 573 | 113 | 406 | 742 |
| 43-S | 2110600 | 712 | 572 | 115 | 418 | 808 |
| 44-S | 2160850 | 719 | 569 | 118 | 434 | 851 |
| 45-S | 2211100 | 690 | 644 | 121 | 439 | 808 |
| 46-S | 2261350 | 749 | 592 | 124 | 454 | 803 |
| 47-S | 2311600 | 758 | 603 | 125 | 462 | 824 |
| 48-S | 2361850 | 770 | 641 | 127 | 471 | 845 |
| 49-S | 2412100 | 783 | 900 | 131 | 478 | 863 |
| 50-S | 2462350 | 795 | 680 | 135 | 494 | 881 |
| 51-S | 2512600 | 829 | 655 | 137 | 499 | 893 |
| 52-S | 2562850 | 846 | 681 | 138 | 514 | 918 |
| 53-S | 2613100 | 787 | 677 | 140 | 522 | 942 |
| 54-S | 2663350 | 830 | 672 | 144 | 527 | 997 |
| 55-S | 2713600 | 826 | 758 | 146 | 534 | 1026 |
| 56-S | 2763850 | 824 | 751 | 148 | 549 | 993 |
| 57-S | 2814100 | 692 | 1024 | 150 | 555 | 1003 |
| 58-S | 2864350 | 757 | 731 | 156 | 564 | 1047 |
| 59-S | 2914600 | 752 | 744 | 157 | 574 | 1030 |
| 60-S | 2964850 | 777 | 744 | 160 | 580 | 1063 |
| 61-S | 3015100 | 745 | 767 | 161 | 590 | 1085 |
| 62-S | 3065350 | 820 | 769 | 165 | 598 | 1101 |
| 63-S | 3115600 | 788 | 774 | 168 | 604 | 1116 |
| 64-S | 3165850 | 803 | 831 | 170 | 612 | 1205 |
| 65-S | 3216100 | 749 | 1108 | 173 | 624 | 1247 |
| 66-S | 3266350 | 824 | 845 | 184 | 630 | 1295 |
| 67-S | 3316600 | 828 | 867 | 189 | 639 | 1262 |
| 68-S | 3366850 | 846 | 844 | 185 | 647 | 1222 |
| 69-S | 3417100 | 798 | 848 | 182 | 656 | 1235 |
| 70-S | 3467350 | 804 | 851 | 189 | 665 | 1247 |
| 71-S | 3517600 | 776 | 819 | 191 | 674 | 1251 |
| 72-S | 3567850 | 805 | 856 | 190 | 685 | 1276 |
| 73-S | 3618100 | 730 | 1231 | 192 | 692 | 1294 |
| 74-S | 3668350 | 808 | 848 | 195 | 697 | 1313 |
| 75-S | 3718600 | 805 | 914 | 202 | 706 | 1427 |
| 76-S | 3768850 | 836 | 871 | 199 | 717 | 1443 |
| 77-S | 3819100 | 760 | 932 | 204 | 722 | 1367 |
| 78-S | 3869350 | 835 | 873 | 205 | 728 | 1373 |
| 79-S | 3919600 | 831 | 878 | 210 | 738 | 1386 |
| 80-S | 3969850 | 855 | 923 | 210 | 749 | 1409 |
| 81-S | 4020100 | 789 | 1326 | 214 | 756 | 1429 |

*Tabella 3-2:Tempi di esecuzione delle reti a singolo strato nascosto sulle schede con sistema Linux*

| Nome Rete | Totale pesi e MAC | Nvidia Jetson Nano (TF) [ms] | Raspberry Pi 4 (TF) [ms] | Raspberry Pi 4 (TFL) [ms] | Google Coral (TFL) [ms] | STM32 MP157A-DK1 (TFL) [ms] |
|---|---|---|---|---|---|---|
| 1-M1 | 1 | 287 | 136 | 3 | 35 | 14 |
| 2-M1 | 2 | 327 | 151 | 4 | 28 | 19 |
| 3-M1 | 3 | 375 | 160 | 6 | 34 | 24 |
| 4-M1 | 4 | 404 | 166 | 7 | 35 | 29 |
| 5-M1 | 5 | 441 | 173 | 8 | 39 | 33 |
| 6-M1 | 6 | 476 | 180 | 9 | 39 | 38 |
| 7-M1 | 7 | 504 | 189 | 11 | 41 | 44 |
| 8-M1 | 8 | 535 | 196 | 12 | 43 | 49 |
| 9-M1 | 9 | 571 | 210 | 13 | 45 | 54 |
| 10-M1 | 10 | 605 | 219 | 14 | 47 | 59 |
| 1-M2 | 1 | 298 | 192 | 12 | 35 | 79 |
| 2-M2 | 2 | 421 | 332 | 79 | 46 | 429 |
| 3-M2 | 3 | 532 | 472 | 142 | 61 | 802 |
| 4-M2 | 4 | 654 | 601 | 204 | 75 | 1147 |
| 5-M2 | 5 | 615 | 782 | 263 | 90 | 1428 |
| 6-M2 | 6 | 653 | 892 | 331 | 109 | 1844 |
| 7-M2 | 7 | 646 | 1023 | 393 | 127 | 2169 |
| 8-M2 | 8 | 689 | 1192 | 454 | 146 | 2508 |
| 9-M2 | 9 | 714 | 1274 | 516 | 203 | 2852 |
| 10-M2 | 10 | 793 | 1430 | 579 | 487 | 3169 |

*Tabella 3-3:Tempi di esecuzione delle reti a strato nascosto multiplo sulle schede con sistema Linux*

Successivamente sono rappresentati i grafici che permettono di visualizzare i dati appena riportati. Si può notare che all'aumentare del numero di nodi, pari al numero di operazioni MAC da eseguire, tutte le schede presentano un andamento prevalentemente crescente nei tempi di esecuzione. Tuttavia una prima differenza che salta subito all'occhio è che le schede sprovviste di acceleratori dedicati alla AI presentano un trend approssimabile a una funzione affine, caratterizzato da un offset particolarmente modesto in caso di modelli quantizzati e leggermente più grande per modelli ad aritmetica a virgola mobile e framework completo.

In particolare le schede che fanno uso di modelli quantizzati senza acceleratori hardware (Raspberry Pi 4 e STM32MP157A-DK1) presentano un andamento particolarmente lineare, dovuto all'utilizzo di un singolo core supportato da TensorFlow Lite. In questo caso si osservano differenze nella pendenza di circa un fattore 6 a sfavore della scheda di

STMicroelectronics, che viene perfettamente giustificato dalla diversa frequenza di funzionamento delle CPU e dalla diversa velocità della memoria.

Per completezza di paragone si è utilizzato TensorFlow Core con un modello non quantizzato sulla stessa Raspberry Pi 4, così da avere un riferimento sull'impatto dei diversi framework e livelli di completezza dei modelli. In particolare l'impiego di TensorFlow Core permette alla piattaforma di sfruttare tutti e 4 i core della CPU. Tuttavia i tempi necessari per lo stesso numero di predizioni diventano più grandi: pur permanendo il comportamento affine l'offset iniziale si decuplica e la pendenza triplica. Questo è perfettamente comprensibile se si considera la maggior complessità del framework e la diversa tipologia di dato.

Diversamente è possibile osservare che per la Nvidia Jetson Nano e la Google Coral i trend risultano completamente differenti, pur trattandosi di funzioni prevalentemente crescenti. Infatti la prima presenta inizialmente tempi abbastanza lunghi e superiori a tutte le altre piattaforme considerate, ma all'aumentare dei parametri mostra un andamento logaritmico che la rende fortemente concorrenziale alle altre nei carichi di lavoro più pesanti. Inoltre, considerando che i modelli della Nvidia Jetson Nano non sono quantizzati come per quasi tutte le altre schede, i risultati sono pienamente in linea con le aspettative, dimostrando una capacità di parallelismo notevole della scheda.

Diversamente la Google Coral, facendo uso di modelli quantizzati, risulta molto prestante fin da subito mantenendo un andamento praticamente costante all'aumentare dei parametri fino a circa 1,6 milioni, in cui vi è un rapido incremento nella caratteristica e il trend prosegue linearmente risultando equiparabile alle altre soluzioni sprovviste di acceleratori. Si ipotizza che questo comportamento sia dovuto all'architettura della TPU, progettata per un numero limitato di parametri, oltre il quale è necessario l'intervento della CPU, che presenta capacità di parallelizzazione notevolmente inferiori.

È anche possibile osservare come, impiegando i framework TensorFlow Core o GPU, le curve siano notevolmente meno omogenee, con andamenti meno regolari e picchi periodici. Questo si giustifica pensando che il pacchetto utilizzato supporta una parallelizzazione multi-core e che le reti non sono scalate di potenze di 2, pertanto è normale che periodicamente vi siano tempi di esecuzione più lunghi. Un'altra differenza tra i due framework si evince sulla memoria occupata, che può superare i 1800 MB in caso di TensorFlow Core o GPU con i modelli più complessi, mentre risulta essere quasi

costante e di poco più di 20 MB nella versione Lite. Un'ultima peculiarità che si può notare in fase di esecuzione è che usando TensorFlow Lite con modelli quantizzati il tempo impiegato nella prima iterazione è praticamente uguale alle altre 11, diversamente nel caso di framework completo la prima previsione può richiedere fino a un incremento di 3 volte il tempo necessario alle normali previsioni, fattore dovuto probabilmente alla necessità di dover allocare le variabili in memoria. Questo giustifica la scelta di aver scartato l'iterazione massima e minima per ciascuna misura, permettendo di ottenere valori più fedeli a quelli che si presenterebbero in un contesto real-time.

Considerando il parallelismo su più strati, poco si riesce a estrapolare dal secondo grafico in cui gli strati nascosti sono di dimensione molto piccola e i trend si presentano tutti affini con offset notevolmente diversi, in cui i modelli quantizzati vincono rispetto a quelli a virgola mobile. Diversamente nel secondo caso multistrato, dove il numero di nodi degli strati nascosti diventa più consistente, si vede confermato quanto l'analisi precedente aveva evidenziato: le schede con acceleratore dedicato presentano una crescita molto più lenta dei tempi di esecuzione all'aumentare degli strati nascosti, segno che la parallelizzazione di queste schede è tale da riuscire a gestire in modo efficiente più strati. In particolare i sistemi che si affidano alla sola CPU riportando un andamento rettilineo, diversamente dalla Google Coral e dalla Nvidia Jetson Nano la cui crescita è più smorzata.

Molto interessante è l'andamento della caratteristica relativa alla Google Coral, che non presenta il rapido incremento che si vedeva nel caso a singolo strato nascosto, nonostante il numero finale di parametri sia perfino superiore a quello del primo esperimento. Questo conferma l'ipotesi precedentemente fatta: l'hardware della Coral supporta singoli strati nascosti non superiori a circa 1,6 milioni di parametri, ma qualora si abbiano più strati di dimensione inferiore a questo limite, anche se complessivamente il numero di nodi è estremamente elevato, la TPU riesce a gestire in modo efficiente il parallelismo.

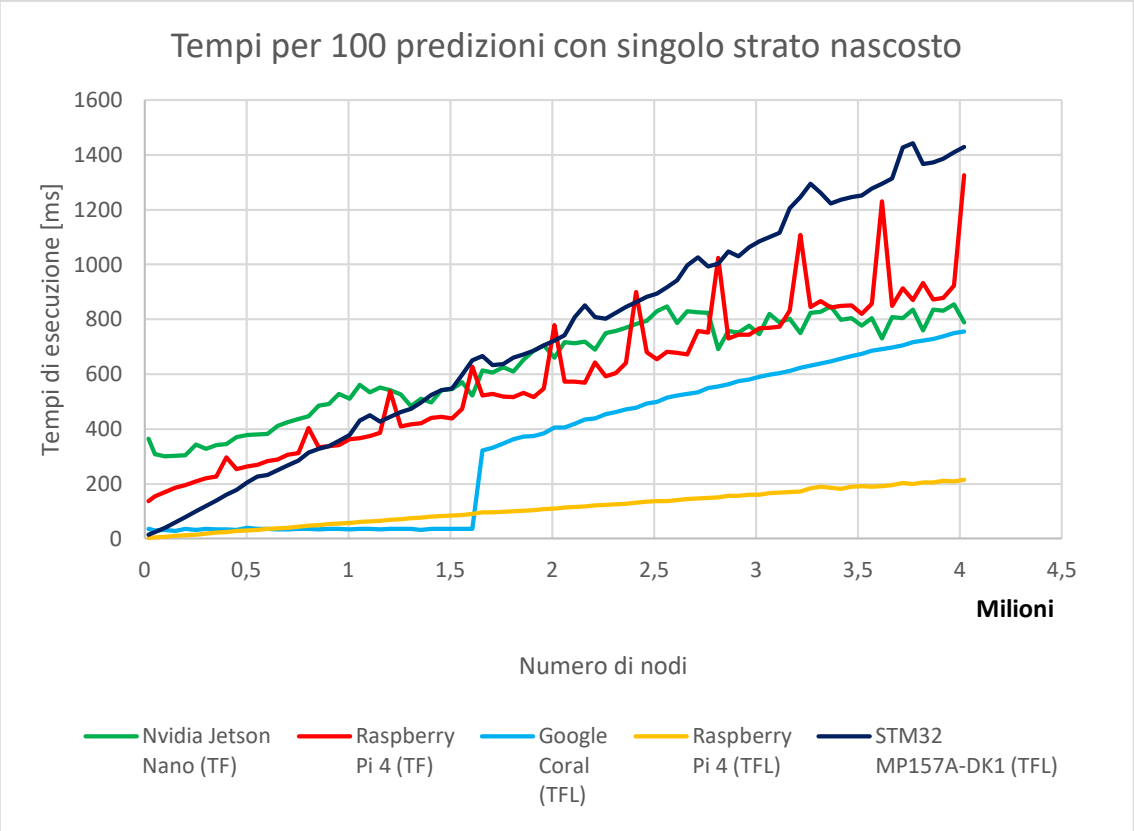
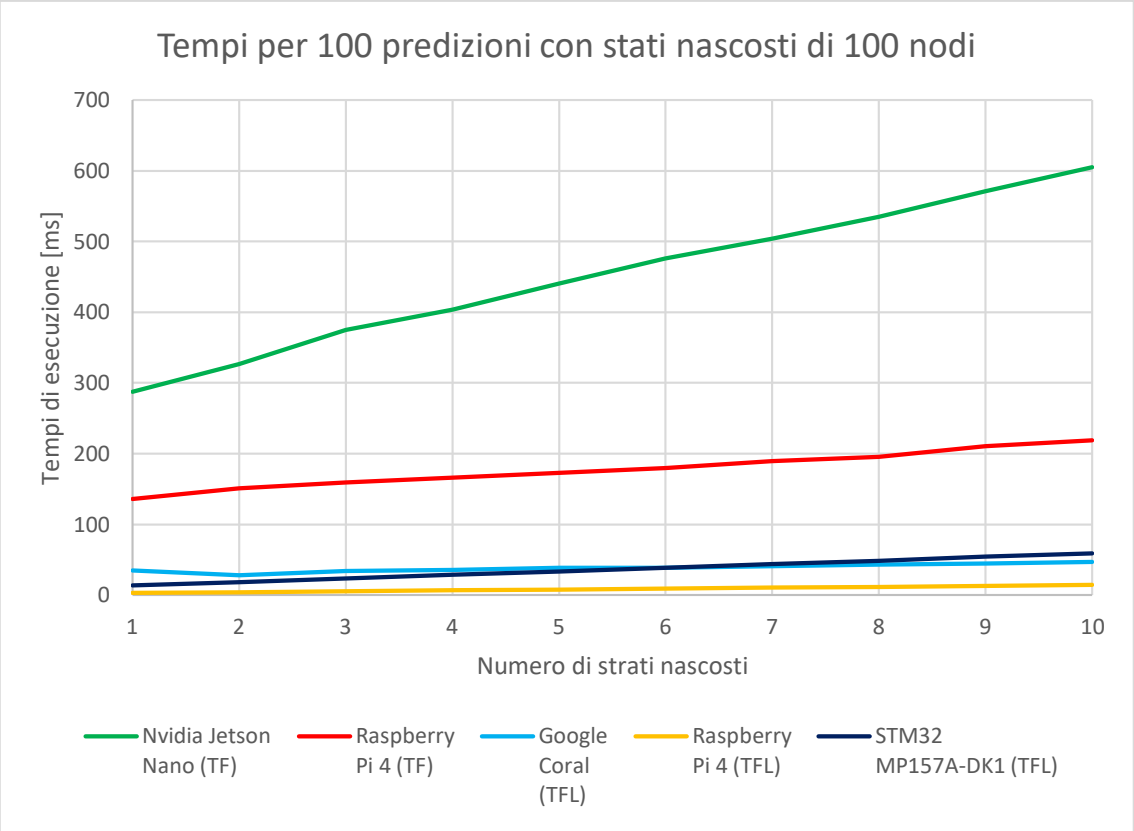



*Figura 3-4: Tempi di esecuzione all'aumentare del numero di nodi per schede a sistema Linux*



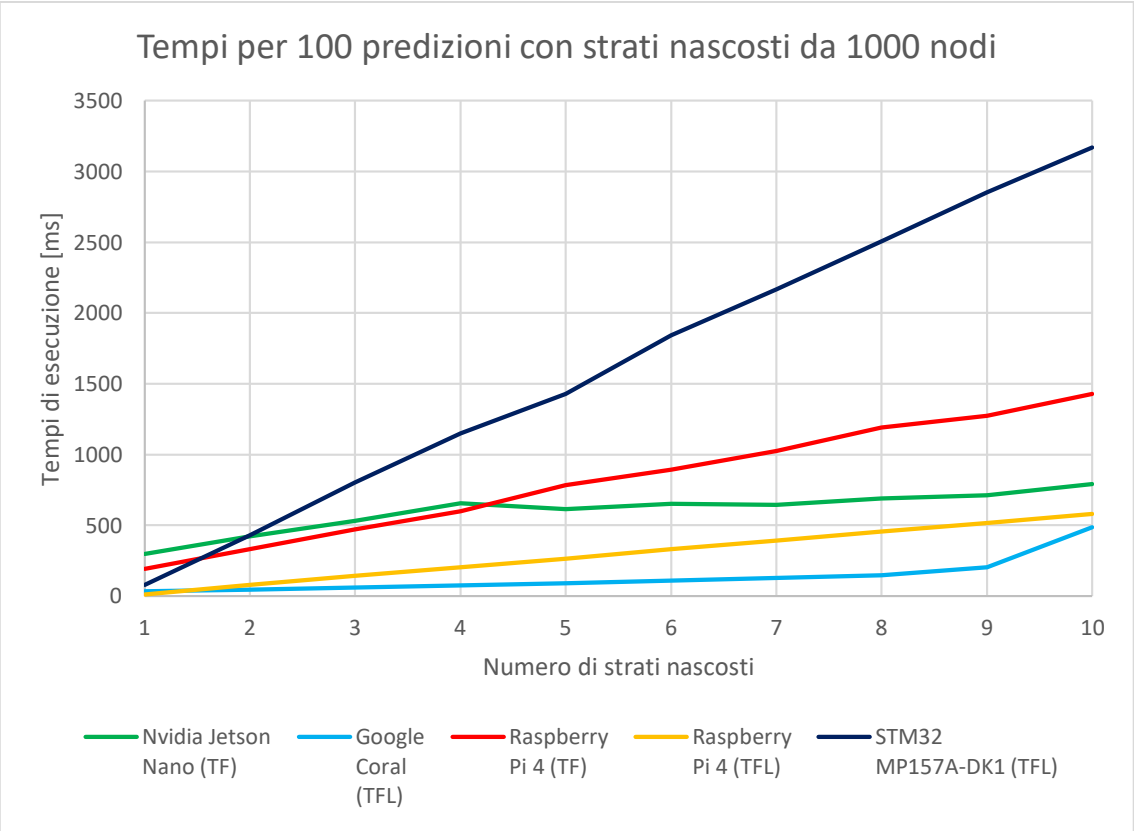




*Figura 3-5: Tempi di esecuzione all'aumentare del numero di strati nascosti per schede a sistema Linux*

Un discorso completamente diverso va fatto per la STM32 Nucleo-H743ZI2, in cui l'assenza di sistema operativo rende inutile ripetere le misure, dal momento che il risultato temporale è deterministico. Anche in questo caso sono state ripetute 100 misure per avere una latenza sufficientemente grande da poter essere misurata senza troppe incertezze. In questo esperimento la misura è stata eseguita mediante il timer Systick del microcontrollore e si è fatto uso di una porta seriale per comunicare la misura temporale esternamente. Infatti l'utilizzo della modalità release anziché di quella di debug, tramite l'IDE che ha permesso di implementare il codice, fornisce una valutazione più realistica delle tempistiche, ma allo stesso tempo rende necessario dover usare una porta seriale per trasmettere i risultati, vista l'impossibilità di accedere ai registri interni del microcontrollore.

Sono stati svolti alcuni test sulla prima rete neurale implementata e l'impiego delle librerie CMSIS, unito alla modalità release, ha permesso di ottenere un incremento di velocità notevole. Quantitativamente parlando, senza ottimizzazioni e in modalità debug l'esecuzione di 100 previsioni di una rete a singolo strato nascosto di 100 nodi richiede 2876 ms che diventano 1516 ms in modalità release e scendono ulteriormente a 604 ms aggiungendo anche l'utilizzo delle librerie CMSIS. Inoltre tali librerie permettono diversi livelli di quantizzazione dei dati in ingresso e nel caso specifico si è scelto di confrontare tutti i livelli di approssimazione possibile, considerando aritmetica a virgola mobile a 32 bit e aritmetica a virgola fissa a 32 e 16 bit (Q31 e Q15). Le reti neurali che si sono potute caricare su questo sistema sono notevolmente più piccole rispetto a quelle impiegate nelle piattaforme edge a sistema Linux. Il limite che è sopraggiunto per primo è quello riguardante la memoria: la STM32 Nucleo-H743ZI2 non riesce a gestire reti con un numero di parametri superiore ai 130000 circa, dato perfettamente in linea con le specifiche del microcontrollore, considerando la RAM di solo 1 MB. Tuttavia è stato possibile valutarne le prestazioni sia in caso di numero di nodi del singolo strato nascosto crescente, sia all'aumentare del numero di strati:

| Nome Rete | Totale pesi e MAC | Floating point 32 bit [ms] | Fixed point 32 bit (Q31) [ms] | Fixed point 16 bit (Q15) [ms] |
|---|---|---|---|---|
| 1-S | 20200 | 604 | 682 | 323 |
| 1.1-S | 30250 | 903 | 1018 | 470 |
| 1.2-S | 40300 | 1088 | 1352 | 655 |
| 2-S | 50350 | 1501 | 1687 | 816 |
| 2.1-S | 60400 | 1781 | 2021 | 928 |
| 2.2-S | 70450 | 2076 | 2356 | 1085 |
| 2.3-S | 80500 | 2166 | 2690 | 1231 |
| 2.4-S | 90550 | 2666 | 3028 | 1389 |
| 3-S | 100600 | 2706 | 3359 | 1528 |
| 3.1-S | 110650 | 3256 | 3693 | 1753 |
| 3.2-S | 120700 | 3246 | 4026 | 1832 |

*Tabella 3-4: Tempi di esecuzione delle reti a singolo strato nascosto su STM32 Nucleo-H743ZI2*

| Nome Rete | Totale pesi e MAC | Floating point 32 bit [ms] | Fixed point 32 bit (Q31) [ms] | Fixed point 16 bit (Q15) [ms] |
|---|---|---|---|---|
| 1-M1 | 1 | 604 | 682 | 323 |
| 2-M1 | 2 | 822 | 1024 | 497 |
| 3-M1 | 3 | 1210 | 1364 | 681 |
| 4-M1 | 4 | 1494 | 1705 | 852 |
| 5-M1 | 5 | 1793 | 2047 | 1026 |
| 6-M1 | 6 | 2093 | 2391 | 1131 |
| 7-M1 | 7 | 2191 | 2732 | 1320 |
| 8-M1 | 8 | 2690 | 3073 | 1471 |
| 9-M1 | 9 | 2742 | 3413 | 1712 |
| 10-M1 | 10 | 3017 | 3755 | 1821 |

*Tabella 3-5: Tempi di esecuzione delle reti a strato nascosto multiplo su STM32 Nucleo-H743ZI2*

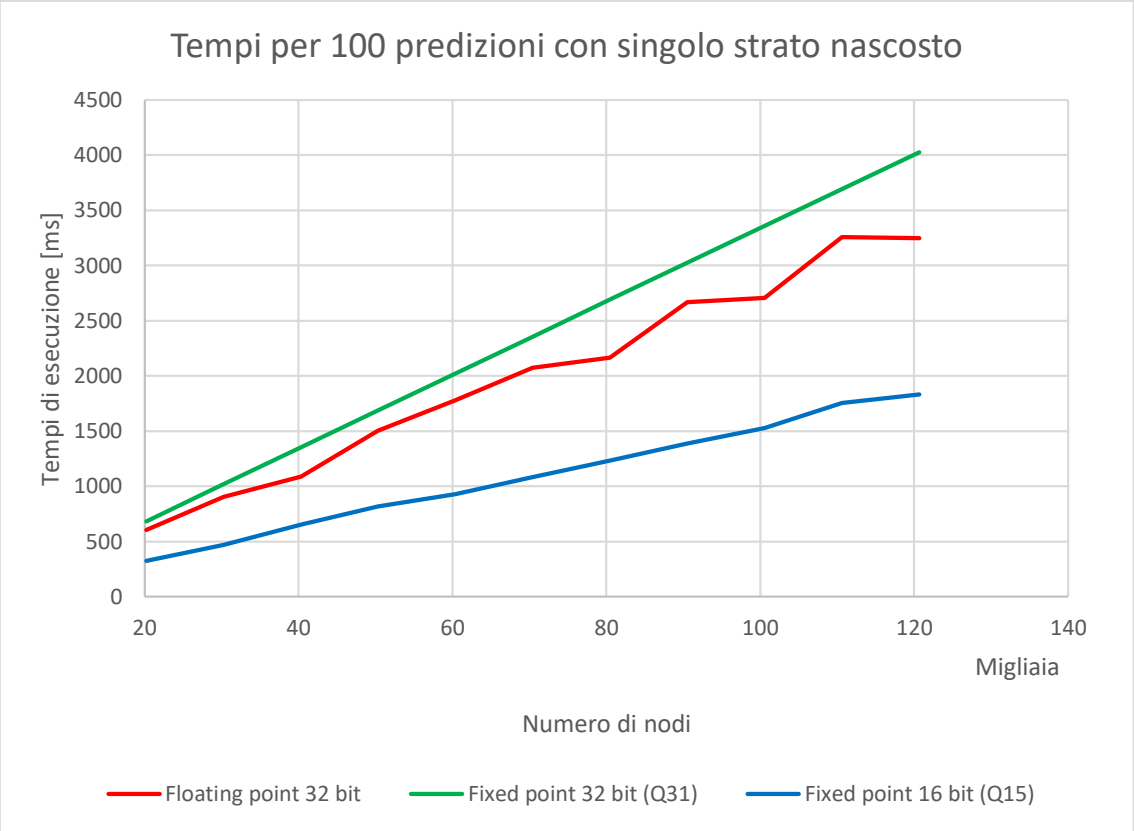



*Figura 3-6: Tempi di esecuzione all'aumentare del numero di nodi per STM32 Nucleo-H743ZI2*

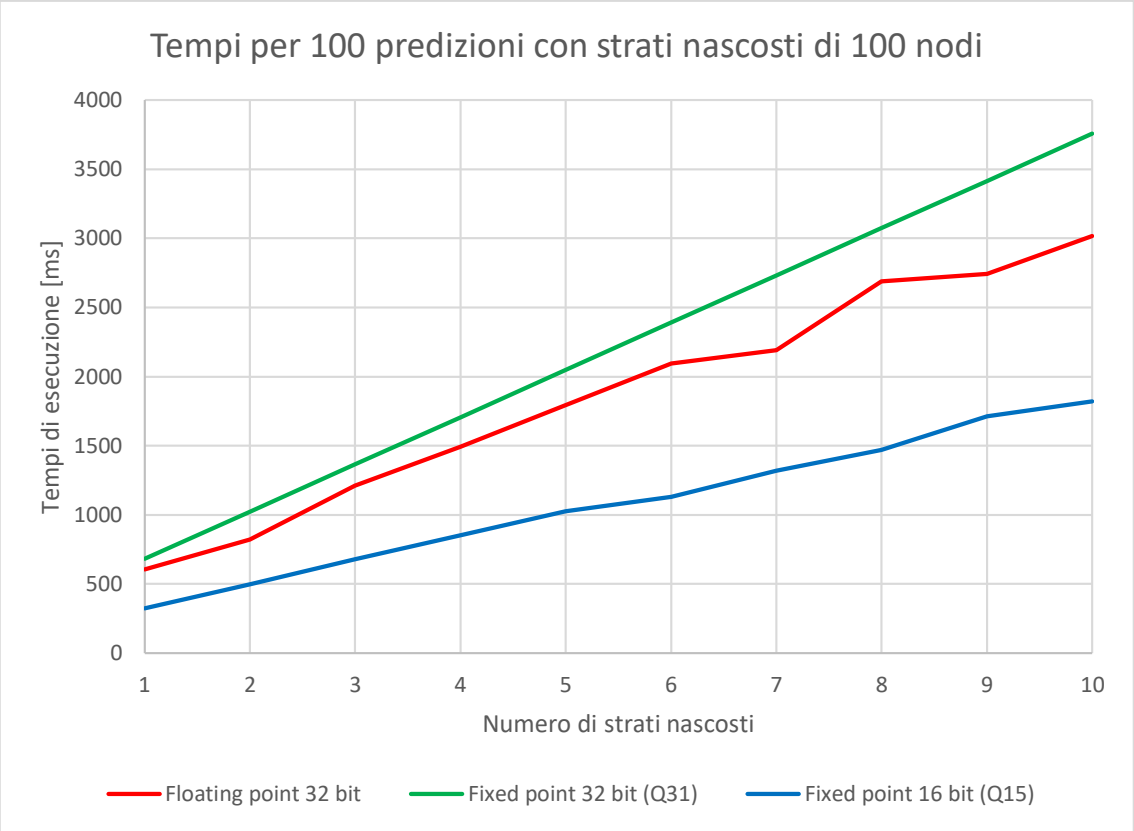



*Figura 3-7: Tempi di esecuzione all'aumentare del numero di strati nascosti per STM32 Nucleo-H743ZI2*

Come si può notare dai dati riportati e dai grafici che li rappresentano, in questo caso i tutti gli andamenti sono crescenti ed equiparabili a lineari, sia nel caso di singolo strato nascosto crescente, sia nel caso di strati nascosti multipli. Questo è in linea con le aspettative dal momento che, nonostante le librerie CMSIS permettano di ottimizzare l'hardware per le moltiplicazioni matriciali, il sistema si comporta comunque come una piattaforma a microprocessore, pertanto non possedendo acceleratori appositamente studiati per l'intelligenza artificiale non ci sono particolari ottimizzazioni aggiuntive all'aumentare dei parametri. È interessante notare come la presenza dell'unità per il calcolo a virgola mobile a 32 bit (FPU) permetta di ottimizzare questo tipo di calcoli rendendoli perfino più efficienti di circa un 20% rispetto alla corrispondente controparte a virgola fissa a 32 bit. Come ci si aspetta poi riducendo poi la precisione dei dati a virgola fissa a 16 bit i tempi si dimezzano rispetto alla configurazione fixed-point a 32 bit, risultando la soluzione più veloce e dimostrando un corretto funzionamento delle librerie CMSIS per la parallelizzazione del calcolo.

## 3.2 IMPIEGO DELLE RETI NEURALI NEL COMPRESSED SENSING

Il Compressed Sensing (CS) è una tecnica di elaborazione del segnale che permette di ridurre il numero di campioni rappresentativi mediante sottocampionamento, non rispettando i vincoli imposti dal teorema di Whittaker-Nyquist-Kotelnikow-Shannon. Si tratta di una procedura che non prevede perdita di informazione qualora vengano rispettati i vincoli imposti e in assenza di rumore, risultando estremamente utile nelle applicazioni in cui le risorse nel nodo trasmettitore sono particolarmente limitate. Infatti l'elaborazione avviene in modo fortemente asimmetrico, richiedendo un modesto sforzo computazionale per la compressione e un impiego di risorse più oneroso in fase di ricostruzione.

Questa caratteristica si adatta perfettamente ai paradigmi del cloud computing e dell'edge computing, in cui in cui le risorse aumentano progressivamente nei livelli di elaborazione. In particolare il CS permette di comprimere il segnale nel nodo sensore con uno sforzo computazionale ridotto, per poi ricostruirlo nei nodi edge o cloud dove vi è maggiore disponibilità di risorse. Questo permette di ridurre notevolmente il numero di bit trasmessi dal nodo sensore al nodo edge, consentendo di rilassare le specifiche sui canali di telecomunicazione, in modo da risparmiare energia a parità di informazioni inviate.

Il compressed sensing può essere effettuato su tutti quei segnali che possono essere rappresentati mediante una base di sparsità, ovvero segnali la cui rappresentazione su una particolare base ortonormale a $n$ dimensioni possiede al più $k$ elementi non nulli, quantità che prende il nome di sparsità del segnale. Pertanto definendo $x$ segnale di $n$ dimensioni di sparsità $k$, $S$ la matrice che ha sulle righe i vettori che compongono la base ortonormale e il vettore $\xi$ contenente i coefficienti si ha:

$$x = S\,\xi$$

con $\xi$ vettore $n$-dimensionale che al suo interno non possiede più di $k$ elementi non nulli. Un segnale di questo tipo si dimostra [9] poter essere compresso in un vettore $m$-dimensionale, con $m < n$, mediante la moltiplicazione per una matrice antipodale $A^{\pm}$, formata da variabili binarie indipendenti di valori egualmente probabili $+1$ e $-1$ e di dimensione $m\,x\,n$. Si dimostra [9] che se $m \geq 2k$ è possibile ricostruire il segnale, limitando il rapporto $n/m$, che rappresenta la compressione del segnale. Pertanto

l'encoder risulta di complessità lineare, limitatosi a una semplice moltiplicazione matriciale. Detto $y$ il segnale $m$-dimensionale all'uscita dell'encoder si ha:

$$y = A^{\pm}x = A^{\pm}S\,\xi$$

A questo punto al nodo ricevente è possibile ricavare il vettore di sparsità $\xi$, e conseguentemente $x$, dalla sola conoscenza di $y$, $A^{\pm}$ e $S$. Infatti si può calcolare $\hat{\xi}$, la stima di $\xi$, come controimmagine relativa a $y$ che tra le infinite che minimizza l'errore di ricostruzione:

$$\hat{\xi} = \arg\min_{\xi\in\mathbb{R}} \|\xi\|_1 \;\; s.t. \|y - A^{\pm}S\,\xi\| \leq \tau$$

in cui ponendo $\tau = 0$, nel caso ideale in assenza di rumore, si ottiene la perfetta ricostruzione.

La procedura descritta tuttavia è sempre da soggetta a elementi di disturbo, come il rumore di quantizzazione, che ne compromettono la perfetta ricostruzione. Per quantificare quanto questi agenti compromettano il procedimento sono state definite due figure di merito che rappresentano il rapporto segnale rumore di ricostruzione sul segnale in ingresso all'encoder (Reconstruction Signal-to-Noise Ratio, RSNR) e sul segnale compresso (Reconstruction Measurements-to-Noise Ratio, RMNR):

$$RSNR = \frac{\|x\|_2}{\|x - \hat{x}\|_2} dB$$

$$RMNR = \frac{\|y\|_2}{\|y - \hat{y}\|_2} dB$$

in cui $\hat{x}$ rappresenta il segnale ricostruito e $\hat{y}$ la compressione del segnale ricostruito $\hat{y} = A^{\pm}\hat{x}$.

Questa procedura risulta estremamente complessa nel nodo ricevente soprattutto all'aumentare della lunghezza delle finestre temporali sul segnale $x$. Proprio per questo sono state ricercate tecniche alternative per la ricostruzione del segnale, in modo da non dover risolvere il problema di minimizzazione vincolata. Un metodo alternativo per la ricostruzione di un segnale sottoposto a compressed sensing prevede l'impiego di una rete neurale, grazie alla quale non occorre risolvere il problema di minimizzazione. Questo metodo prevede, mediante la rete appositamente addestrata e con soglie di uscita ottimizzate, la stima del supporto di $\xi$ indicato con $\hat{s}$, tramite il quale è possibile ricavare

il vettore $\xi$ stesso con una semplice matrice inversa, riducendo il costo computazionale di ricostruzione. Infatti sapendo che $y = A^{\pm} S\,\xi$ e conoscendo il supporto $\hat{s}$ di $\xi$ è possibile invertire la relazione tra $y$ e $\xi$ tramite la pseudo-inversione di Moore-Penrose, ottenendo gli elementi non nulli del vettore di sparsità ($\xi_{|\hat{s}}$):

$$\xi_{|\hat{s}} = (A^{\pm} S_{|\hat{s}})^{\dagger} y$$

A questo punto conoscendo il supporto di $\xi$ ($\hat{s}$) e gli elementi non nulli che lo compongono ($\xi_{|\hat{s}}$) si può stimare il vettore di sparsità e di conseguenza calcolare la stima del segnale iniziale mediante $S$.

La rete neurale che ha il compito di identificare il supporto del segnale è feed-forward stratificata pienamente connessa (MLP) ed è formata da uno strato di ingresso di $m$ nodi, da due strati nascosti con $2n$ nodi con funzioni di attivazione rettificatrici (ReLu), da uno strato nascosto con $n$ nodi con la medesima funzione di attivazione e da uno strato di uscita a $n$ nodi con funzione sigmoidea per l'attivazione. La rete è stata addestrata per i valori $n = 128$, $k = 24$ e per diversi valori di $m$ (48, 54 e 64) e nell'addestramento è stata considerata la matrice di compressione $A$, che non risulta più antipodale ma ottimizzata come uno strato aggiuntivo alla rete. Grazie a questo, la compressione permette di ottenere gli $m$ valori rappresentativi del segnale grazie ai quale la rete meglio funziona nella ricostruzione del supporto.

L'addestramento ha utilizzato come funzione l'entropia incrociata sulle classi come errore e 1000 epoche su un training set di 10000 elementi e valutata su un test set di 15000. All'interno del decoder è poi possibile calcolare il RMNR, figura di merito che identifica quanto la ricostruzione sia avvenuta bene tramite una successiva compressione del segnale ricostruito, calcolando la differenza tra il vettore ricevuto e quello calcolato.

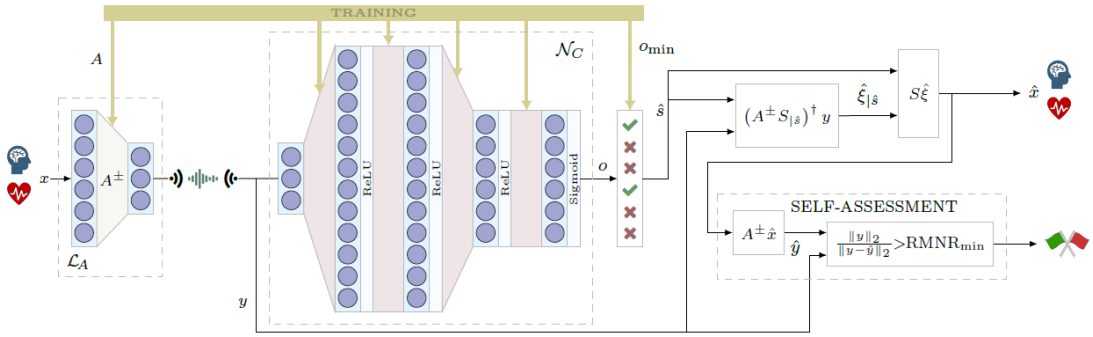



*Figura 3-8: Schema a blocchi di un decoder per segnali sottoposti a compressed sensing implementato mediante rete neurale [9]*

Questa procedura è possibile qualora esista una rappresentazione sparsa per la classe di segnali considerata. Nel caso specifico sono stati adottati dei segnali ECG sintetici, sui quali la rete è stata allenata, la cui rappresentazione risulta sparsificata usando come base le funzioni wavelet symlet 6.

Proprio grazie all'impiego di reti neurali, questo algoritmo di ricostruzione si presta particolarmente all'implementazione su piattaforma edge, risultando estremamente utile in campo applicativo, dal momento che la compressione ha un costo molto limitato. I livelli di compressione raggiunti sono nel caso specifico variabili da 2 a 2,7 in base al valore di $m$ adottato. Inoltre, se si considera la rete neurale utilizzata, essa risulta estremamente più semplice rispetto a quelle impiegate per testare le piattaforme nel paragrafo precedente, rendendo l'utilizzo della tecnica del compressed sensing ancora più appetibile nei casi pratici. Di seguito è riportata una tabella in cui sono indicati i parametri che caratterizzano le reti impiegate al variare di $m$:

| Valori di m | Dimensione ingresso | Dimensione uscita | Numero strati nascosti | Totale pesi e MAC |
|---|---|---|---|---|
| 48 | 48 | 128 | 3 | 127744 |
| 54 | 54 | 128 | 3 | 129280 |
| 64 | 64 | 128 | 3 | 131840 |

*Figura 3-9: Parametri delle reti neurali utilizzate per la ricostruzione di segnali sottoposti a CS*

Come si può notare sia la profondità della rete, sia il numero totale di pesi risultano essere pienamente gestibili dalle schede operanti con sistema Linux ed equipaggiate con TensorFlow o TensorFlow Lite. Nel caso specifico va ricordato che l'impiego di modelli

quantizzati potrebbe compromettere notevolmente il segnale, aumentando il rumore di quantizzazione e risultando particolarmente fastidioso in caso di ricostruzione. Tuttavia questo aspetto è da valutare in base alla classe di segnali considerata.

Diversamente dalle altre piattaforme edge, la STM32 Nucleo-H743ZI2 potrebbe presentare alcuni problemi nell'implementazione di tali reti per via della memoria limitata. Infatti il numero totale di pesi delle reti è al limite della gestibilità della scheda, risultando troppo elevato soprattutto nel caso di $m = 64$, in cui il numero di parametri supera 130000.

# 4 RIVELAZIONE DI ANOMALIE MEDIANTE COMPRESSED SENSING

## 4.1 PROCEDIMENTO DI IDENTIFICAZIONE E ANOMALIE CONSIDERATE

Dopo aver valutato come le varie piattaforme commerciali si adattino all'esecuzione di reti neurali per implementare un decodificatore di segnali sottoposti a compressed sensing, si è deciso di analizzare come questa tecnica permetta di individuare eventuali anomalie presenti nel segnale. Questa è una funzione aggiunta che permette di eseguire due compiti in maniera simultanea: comprimere il segnale senza perdere informazione e determinare se questo presenta anomalie. Questo aspetto risulta un notevole valore aggiunto al procedimento, dal momento che non richiede risorse aggiuntive e permette di eseguire una funzione tipica dell'intelligenza artificiale, risultando estremamente utile in diversi contesti applicativi.

La rilevazione di anomalie avviene per mezzo del blocco di autovalutazione contenuto all'interno del sistema di ricostruzione. Infatti, sapendo che tutto l'apparato di compressione e decompressione è stato addestrato per lavorare al meglio con una certa classe di segnali, qualora si presenti un segnale con caratteristiche che non rientrano in quelle che il sistema si aspetta, il risultato della ricostruzione sarà di qualità nettamente inferiore rispetto a quella attesa. Con questo semplice ragionamento è possibile implementare un predittore binario il cui compito è quello di determinare se un segnale è anomalo rispetto alla classe considerata.

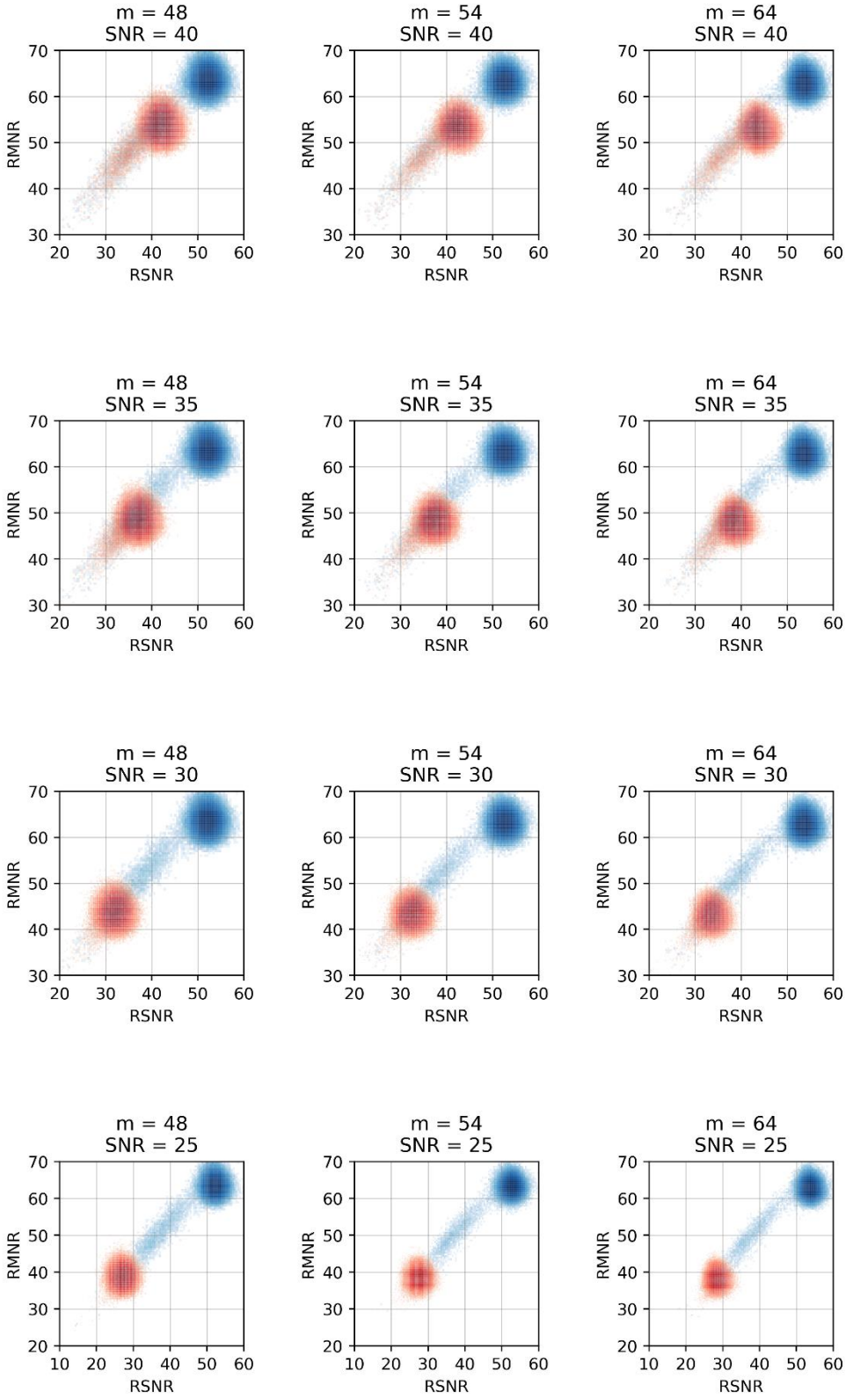
La valutazione di quanto la ricostruzione è avvenuta bene avviene mediante le figure di merito RSNR e RMNR descritte nel capitolo precedente. Queste quantità permettono di dare una stima quantitativa di quanto il segnale ricostruito si avvicini a quello originale,

nel caso del RSNR, o quanto il segnale nuovamente compresso si allontani da quello ricevuto, nel caso del RMNR. La figura di merito che più interessa è il RSNR dal momento che si occupa direttamente del segnale in forma estesa e non della sua versione compressa. Nonostante ciò, è possibile verificare come queste due figure di merito siano strettamente correlate, rendendo il RMNR una valida alternativa al RSNR. Infatti, se si vuole implementare questo algoritmo in un sistema reale, il RMNR è l'unica metrica disponibile nel nodo ricevente, dal momento che non si possiede la versione originale del segnale ma solo della sua ricostruzione, impedendo così il calcolo del RSNR in ricezione.

Nel caso specifico è stata considerata come anomalia l'aggiunta di rumore gaussiano bianco ai segnali ECG sintetici di un data set di 160000 elementi ed è stato subito possibile verificare come la qualità del segnale stesso e della ricostruzione si degradasse al diminuire del SNR. La scelta di questo tipo di anomalia è dovuto prevalentemente alla sua semplicità implementativa, nonostante il procedimento descritto sia valido per qualunque tipo di variazione del segnale che ne alteri le proprietà descritte dai segnali del training set. Per rientrare in uno scenario più realistico si è scelto di considerare un noise-floor di base dovuto alla quantizzazione che rappresentasse il caso fault-free. Proprio per questo si è utilizzato come base un SNR di 50 dB, valore ragionevole che rappresenta il rumore di quantizzazione dovuto alla precisione della macchina. Ovviamente in questa situazione sono stati considerati numeri floating point a singola precisione, ricordando poi che il valore del rumore di quantizzazione andrebbe cambiato qualora si faccia uso di una diversa risoluzione numerica della macchina. A questo punto si è deciso di considerare anomalia tutti i segnali di base a cui sono stati aggiunti diversi livelli di rumore. In particolare i livelli di SNR considerati sono 40, 30, 25, 20, 10 e 0 dB. In questa configurazione è stato considerato fault-free ogni segnale a cui era stato applicato un rumore tale da raggiungere i 50 dB di SNR (il rumore di quantizzazione), mentre tutti gli altri con un livello inferiore di SNR sono stati etichettati come affetti da anomalia.

Una volta stabilita questa classificazione binaria è stato possibile verificare come le figure di merito del compressed sensing variassero al variare del SNR, tentando di individuare una correlazione, per mezzo della quale poi potrà essere costruito un predittore. In particolare quello che ci si aspetta è che al diminuire del SNR cali anche la qualità della ricostruzione e di conseguenza anche le figure di merito ad esso associate. Per verificare questa relazione è stata utilizzata una rappresentazione a punti (scatter plot) per i vari livelli di SNR elencati, considerando per ciascun segnale del data set i relativi valori di

RSNR e RMNR al variare del rumore aggiunto. Questa procedura è stata iterata per ciascuno dei tre valori di $m$ considerati nella fase di ricostruzione, in modo da verificare se il rapporto di compressione incidesse sulla capacità di individuazione delle anomalie.


m = 48
SNR = 40
m = 54
SNR = 40
m = 64
SNR = 40
m = 48
SNR = 35
m = 54
SNR = 35
m = 64
SNR = 35
m = 48
SNR = 30
m = 54
SNR = 30
m = 64
SNR = 30
m = 48
SNR = 25
m = 54
SNR = 25
m = 64
SNR = 25
RMNR
RSNR

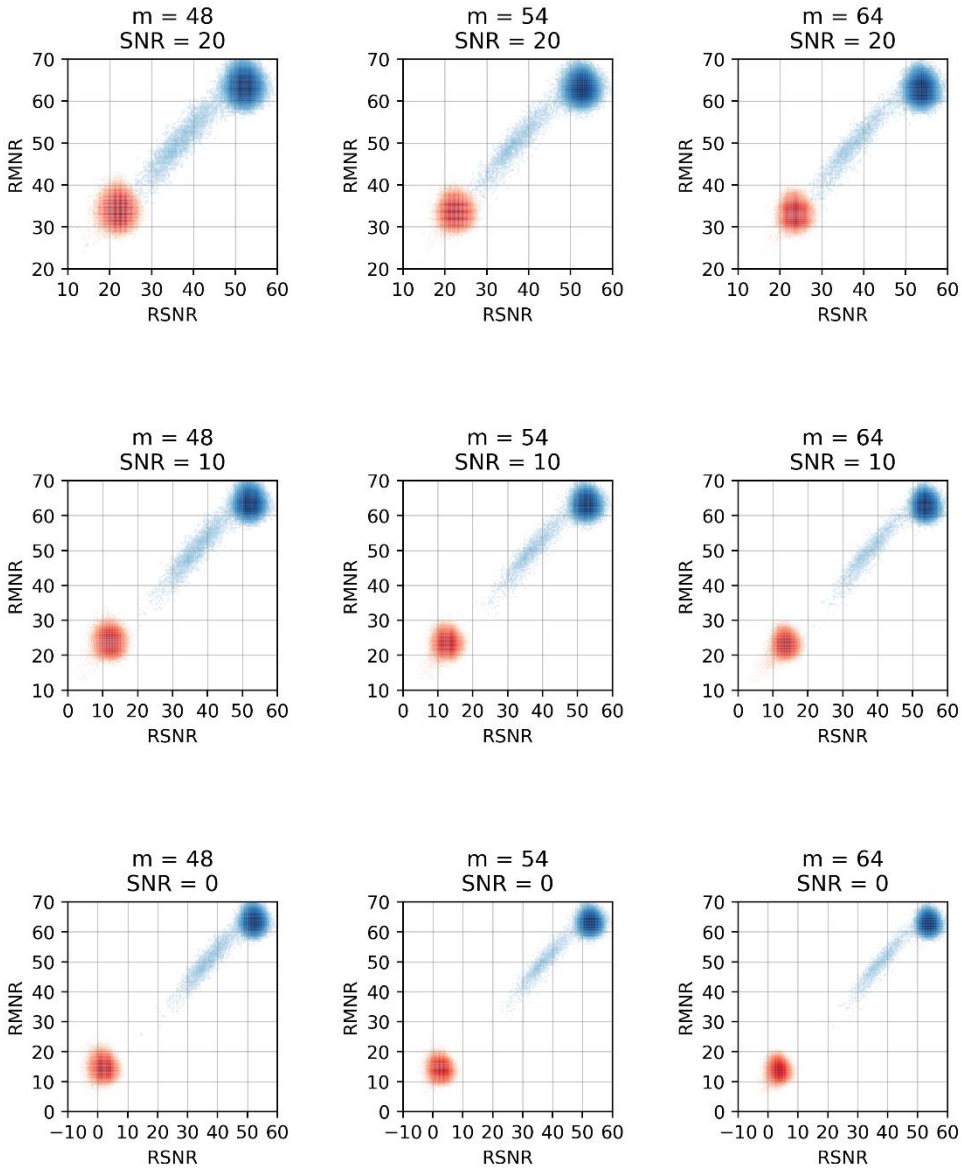



*Figura 4-1: Grafici di dispersione delle figure di merito RSNR e RMNR al variare del SNR e m*

*In blu sono rappresentati i segnali secondo il caso fault-free con SNR pari a 50 dB mentre in rosso sono raffigurati i segnali con livelli di rumore superiore. Tanto più i cluster sono separati tanto più è facile classificare il segnale mediante le figure di merito indicate.*

Si può notare come al calare del rapporto segnale-rumore i due cluster che rappresentano i segnali affetti da rumore aggiuntivo e quelli soggetti al solo rumore di quantizzazione si allontanino e risultino sempre più separati, ovvero più facilmente classificabili dal predittore binario. Non si rilevano differenze particolarmente significative al variare di $m$, ma più il SNR è alto, più i cluster presentano una coda allungata verso valori minori di RSNR e RMNR, indicando che anche in alcuni casi a basso rumore la ricostruzione

può risultare meno efficiente del previsto. Questo è particolarmente problematico qualora si voglia discernere tra anomalie e casi in assenza di rumore, dal momento che queste code possono creare falsi positivi. Diversamente al diminuire del SNR le code tendono ad accorciarsi, indicando come la ricostruzione in questi casi abbia risultati più scarsi ma meno variabili. Infine è possibile notare una fortissima correlazione tra RSNR e RMNR, garantendo che pur considerando la figura di merito che agisce sui segnali compressi, comunque si ha un indice di come avviene la ricostruzione sul segnale originale.

## 4.2 REALIZZAZIONE DEL PREDITTORE BINARIO

Dopo aver definito le modalità con cui identificare le anomalie è stato possibile realizzare un predittore binario che permettesse di riconoscere in modo efficiente quali segnali fossero affetti da disturbi, considerando anche un'ottica implementativa. Per la costruzione del predittore è stato necessario l'utilizzo dello stesso data set impiegato nell'addestramento delle reti, tramite il quale poi è stata anche valutata l'accuratezza, dividendo i 160000 elementi in 10000 per la scelta delle soglie e 150000 per le stime finali.

Se si considera un predittore binario è possibile identificare quattro possibili esiti della predizione:

- Veri positivi (True Negative, TN): rappresentano gli elementi caratterizzati da una anomalia che il predittore classifica come anomali;
- Veri negativi (True Positive, TP): rappresentano gli elementi esenti da anomalie che il predittore non classifica come anomali;
- Falsi positivi (False Positive, FP): rappresentano gli elementi esenti da anomalie che il predittore classifica come anomali;
- Falsi negativi (False Negative, FN): rappresentano gli elementi caratterizzati da una anomalia che il predittore non classifica come anomali.

Lo scopo nella realizzazione del predittore binario è quello di posizionare la soglia di attivazione in modo da ottimizzare una certa figura di merito. Nel caso specifico la figura di merito da ottimizzare era l'accuratezza complessiva, vale a dire la media tra le percentuali di veri positivi e veri negativi:

$$ACC = \frac{TNR + TPR}{2}$$

in cui $TNR$ rappresenta la True Negative Rate e la $TPR$ rappresenta la True Positive Rate, ovvero rispettivamente la percentuale di veri positivi e negativi sull'intero test set.

Nella ottimizzazione delle soglie del predittore binario, per prima cosa è risultato necessario definire dei valori di probabilità che rappresentassero i vari livelli di veri negativi imposti sul training set, considerando i segnali con solo il rumore di quantizzazione ($SNR = 50\ dB$). Infatti, imponendo una certa True Negative Rate (TNR) del predittore, è possibile ottenere delle soglie sul valore di RMNR al variare di $m$ (48, 54 e 64), grazie alle quali è possibile ricavare una ripartizione del training set secondo la percentuale richiesta. Questo ha permesso di ottenere un valore di RMNR per ciascuno dei 3 valori di compressione considerati e per ciascuna probabilità imposta.

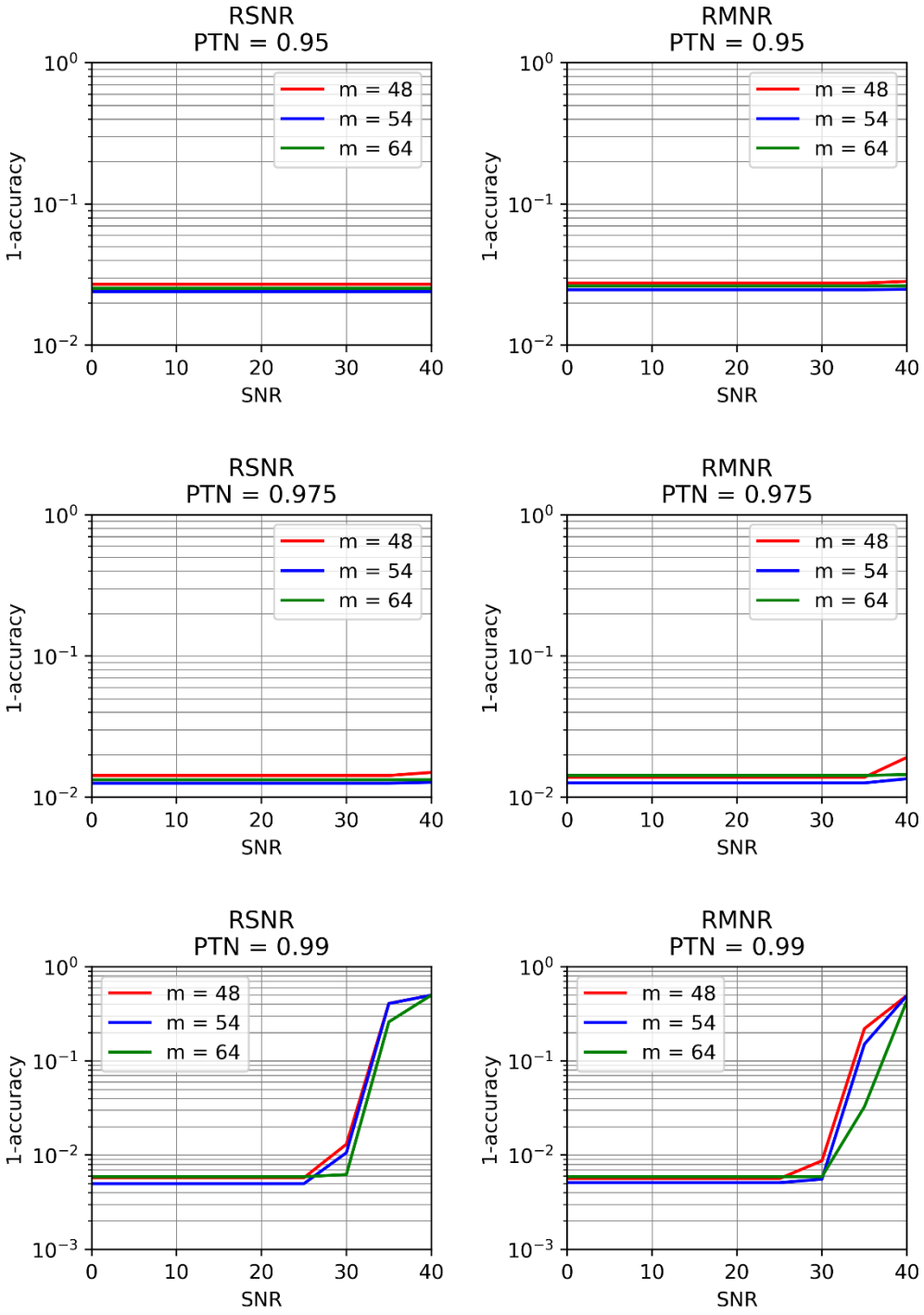
Si sono considerati 15 diversi valori di probabilità di veri positivi, risultando un primo grado di libertà nell'ottimizzazione del predittore: 95%, 97,5%, 99%, 99,25%, 99,5%, 99,75%, 99,9%, 99,925% 99.95%, 99,975%, 99,99%, 99,9925%, 99,995%, 99,9975%, 99,999%. In seguito si è potuta calcolare la TNR sul test set, verificando che le percentuali imposte risultassero molto simili a quelle ricavate, a dimostrazione che il training set è ben rappresentativo dell'intero data set. Per confronto è stato eseguito lo stesso procedimento per i valori di RSNR, ottenendo un totale di 90 soglie e altrettanti valori di true negative rate calcolate sul test set (2 metriche, 3 fattori di compressione e 15 livelli di probabilità imposta).

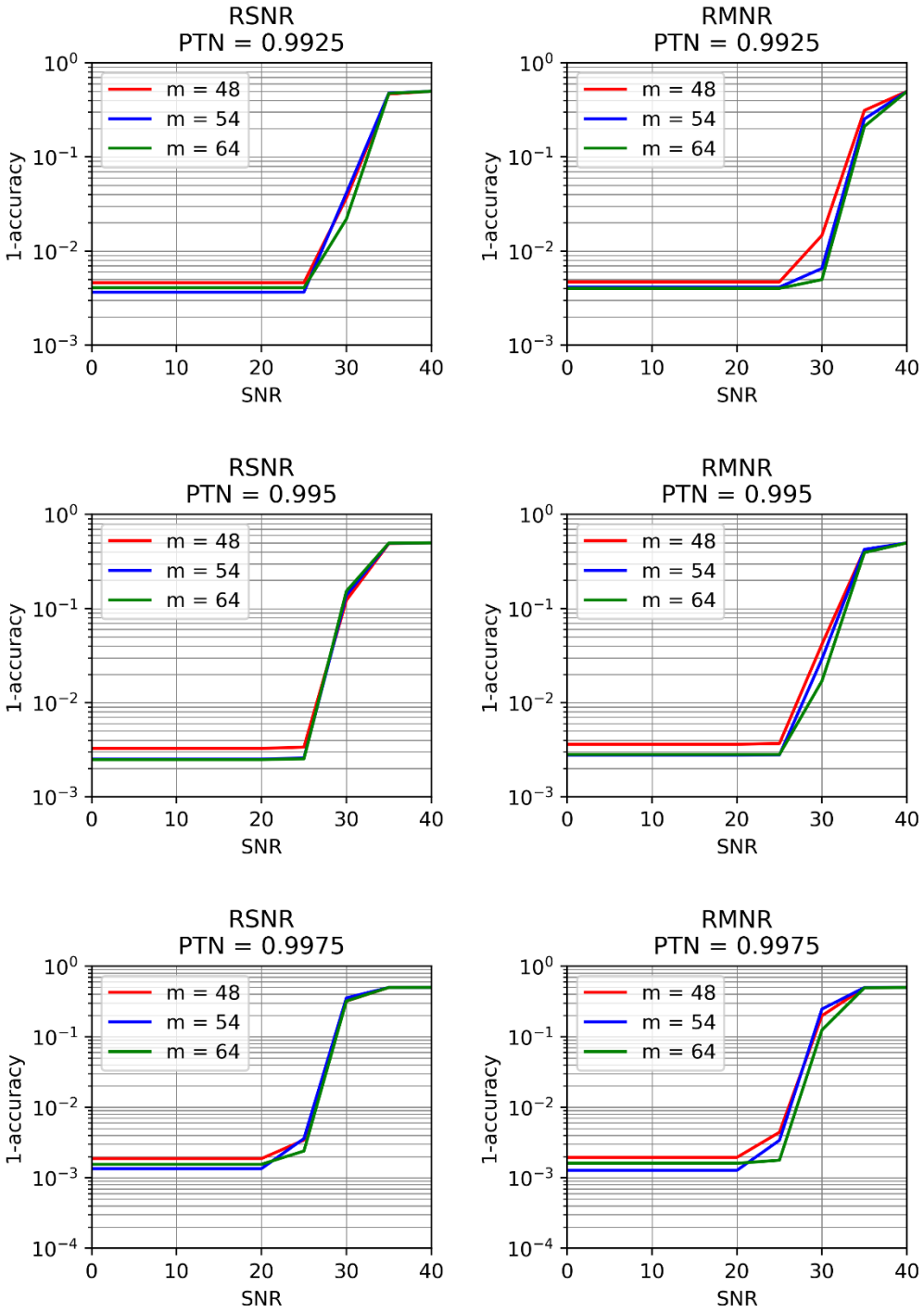
Definendo le soglie si sono identificati altrettanti predittori binari con diverse TNR. Per scegliere quale di questi fosse il migliore è risultato necessario valutare la corrispondente TPR e calcolarne conseguentemente l'accuratezza, figura di merito che in questo caso si vuole ottimizzare. Per fare questo si sono considerati i test set con i diversi livelli di SNR considerati nel paragrafo precedente (40, 30, 25, 20, 10, 0). Nel classificare i segnali rumorosi, che in un caso ideale sarebbero dovuti tutti essere identificati come anomali, si è potuto calcolare il rapporto tra anomalie individuate e numero di elementi considerati, ottenendo la true positive rate. Per ciascun valore di $m$, per ciascun livello di rumore, per ciascuna figura di merito e per ciascuna probabilità imposta è stato possibile ottenere un valore di TPR, grazie al quale è stato possibile calcolare l'accuratezza del predittore considerato, utilizzando nel calcolo la TNR calcolata sul test set.

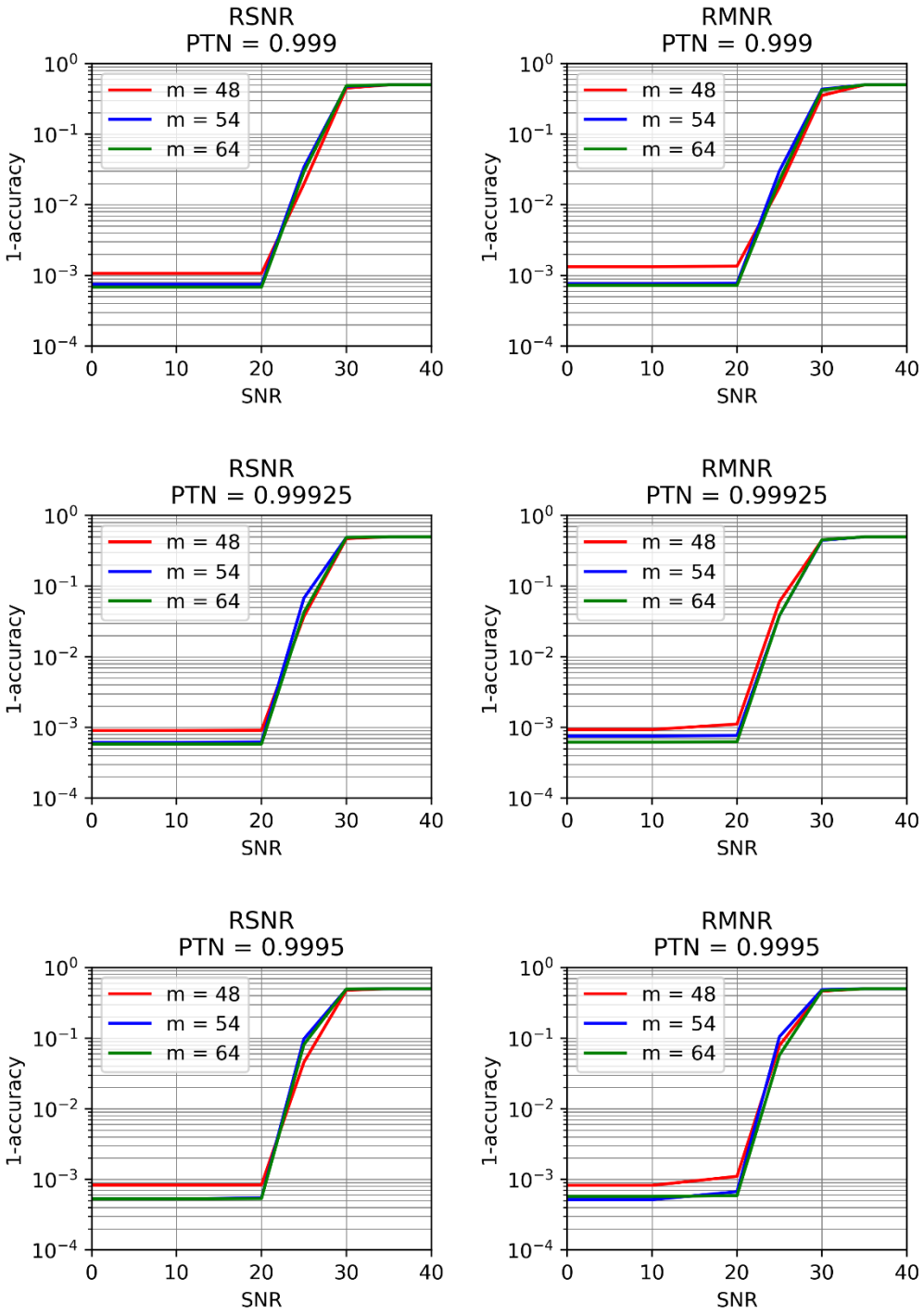
Le aspettative sono che l'accuratezza aumenti al diminuire del SNR, dal momento che, come messo in luce dalle rappresentazioni del capitolo precedente, i livelli di RMNR dei casi affetti da rumore bianco risultano più distanziati da quelli fault-free. Diversamente ci sia aspetta che la differenza nell'impiego delle diverse metriche (RSNR e RMNR) sia perlopiù trascurabile, data la forte correlazione che le lega. Più difficile è la valutazione della probabilità di veri positivi imposta e del rapporto di compressione. Infatti imponendo un valore troppo alto o troppo basso di probabilità si rischia che il predittore tenda a sovrastimare o sottostimare i falsi positivi. Proprio per questo ci si aspetta che esista un valore di ottimo intermedio. Considerando i valori di $m$ risulta più intuitivo che all'aumentare di tale parametro, e quindi alla riduzione della compressione, la rete sia in grado di reperire più informazioni dai dati in ingresso e quindi riuscire a identificare meglio se si è in presenza di un'anomalia o meno, anche se questo trend andrà verificato con le simulazioni.

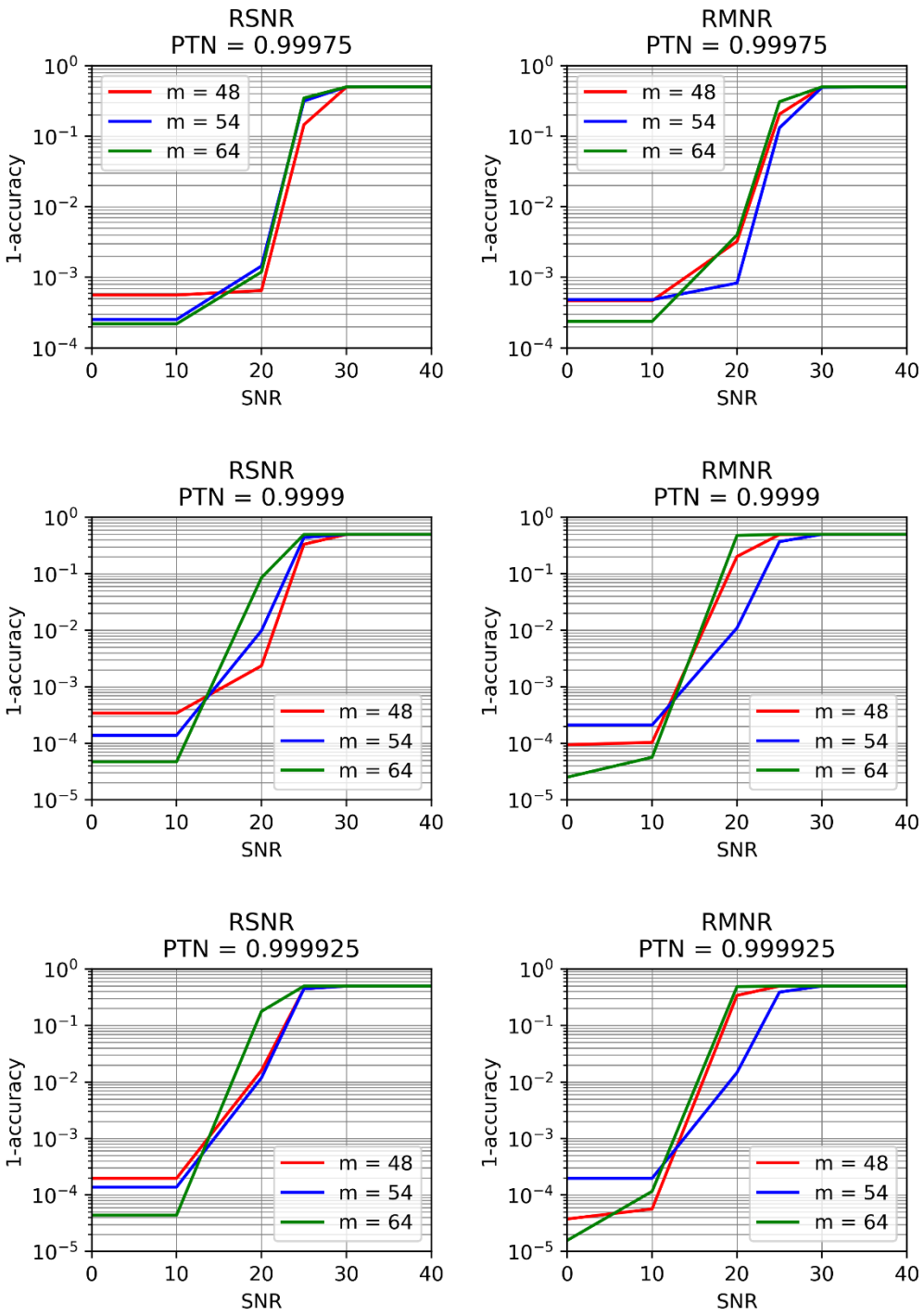
Di seguito sono riportati i grafici che rappresentano il complemento all'accuratezza in funzione del SNR in scala logaritmica in diversi grafici al variare della metrica scelta e della probabilità imposta inizialmente.

Come ci si aspetta, i trend del complemento all'accuratezza risultano in tutte le situazioni crescenti all'aumentare del SNR, a dimostrazione del fatto che più il rumore è alto meglio si riescono a individuare le anomalie. Nonostante questo, si può notare come per valori di SNR pari o superiori a 30 dB l'accuratezza sia piuttosto bassa e non migliori né al variare del fattore di compressione né cambiando la probabilità imposta inizialmente. Questo risulta un ostacolo abbastanza grosso, che limita il campo di applicazione di questa tecnica. Considerando il fattore di compressione è difficile identificare un trend preciso diversamente da quanto ci si aspettava. Se da un lato infatti è vero che riducendo la compressione è più facile individuare anomalie per certe configurazioni, si può notare che per altre avviene l'esatto opposto. Va detto però che in generale, soprattutto per livelli di rumore non troppo elevati, il variare di $m$ non modifica particolarmente le caratteristiche, risultando abbastanza indipendente dall'accuratezza ottenuta. Oltre a questo, se si considera la probabilità imposta, si può notare che al crescere di questa i miglioramenti si vedano soprattutto su valori di rumore alti, portando però un conseguente peggioramento dell'accuratezza per valori di rumore più contenuti. Questo mette in risalto la possibilità che il valore ottimo di probabilità imposta dipenda dal livello di rumore considerato.


RSNR
PTN = 0.95
RMNR
PTN = 0.95
RSNR
PTN = 0.975
RMNR
PTN = 0.975
RSNR
PTN = 0.99
RMNR
PTN = 0.99
m = 48
m = 54
m = 64
1-accuracy
SNR


RSNR
PTN = 0.9925
RMNR
PTN = 0.9925
RSNR
PTN = 0.995
RMNR
PTN = 0.995
RSNR
PTN = 0.9975
RMNR
PTN = 0.9975
m = 48
m = 54
m = 64
1-accuracy
SNR


RSNR
PTN = 0.999
RMNR
PTN = 0.999
RSNR
PTN = 0.99925
RMNR
PTN = 0.99925
RSNR
PTN = 0.9995
RMNR
PTN = 0.9995
m = 48
m = 54
m = 64
1-accuracy
SNR


RSNR
PTN = 0.99975
RMNR
PTN = 0.99975
RSNR
PTN = 0.9999
RMNR
PTN = 0.9999
RSNR
PTN = 0.999925
RMNR
PTN = 0.999925
1-accuracy
SNR
m = 48
m = 54
m = 64

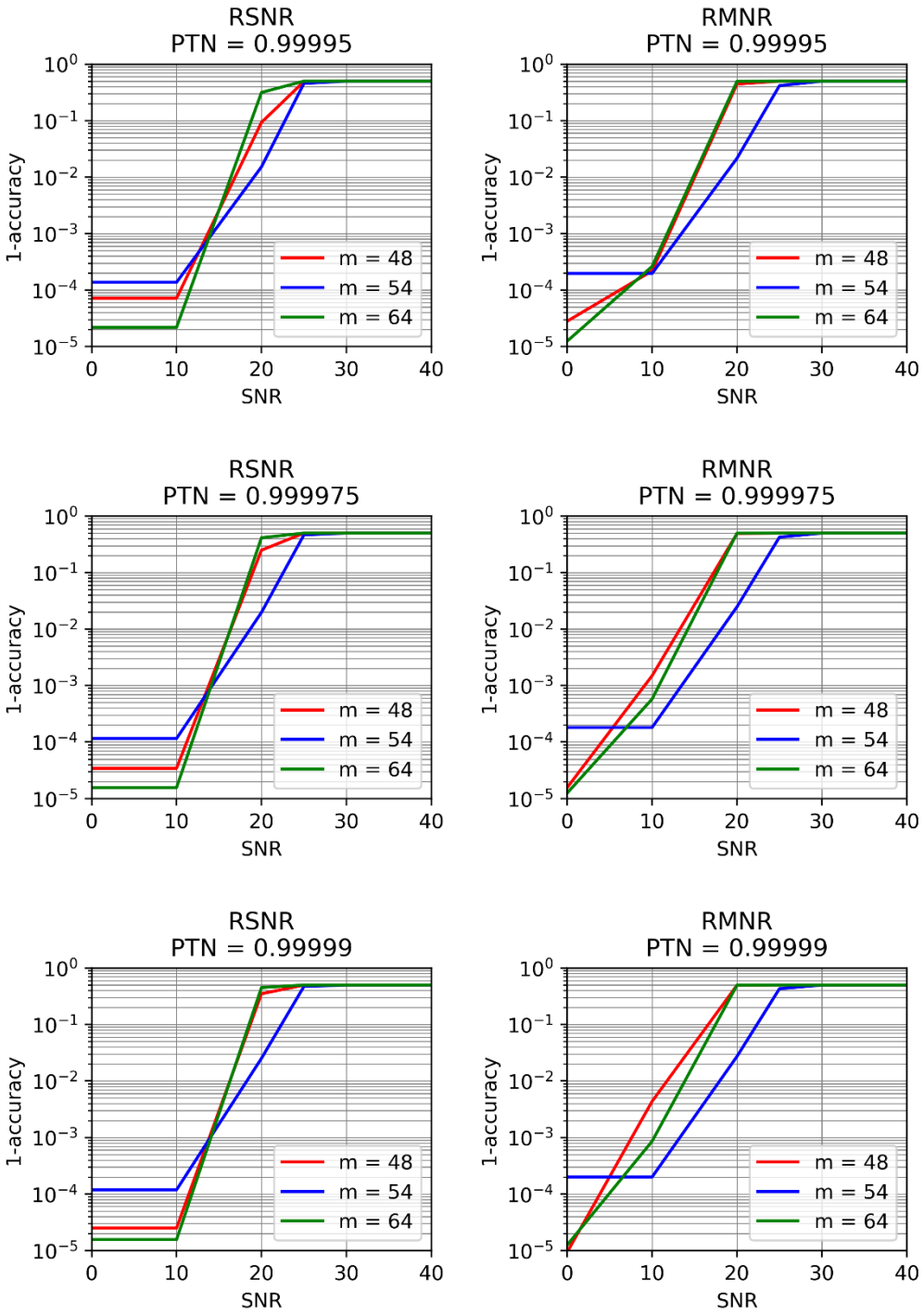


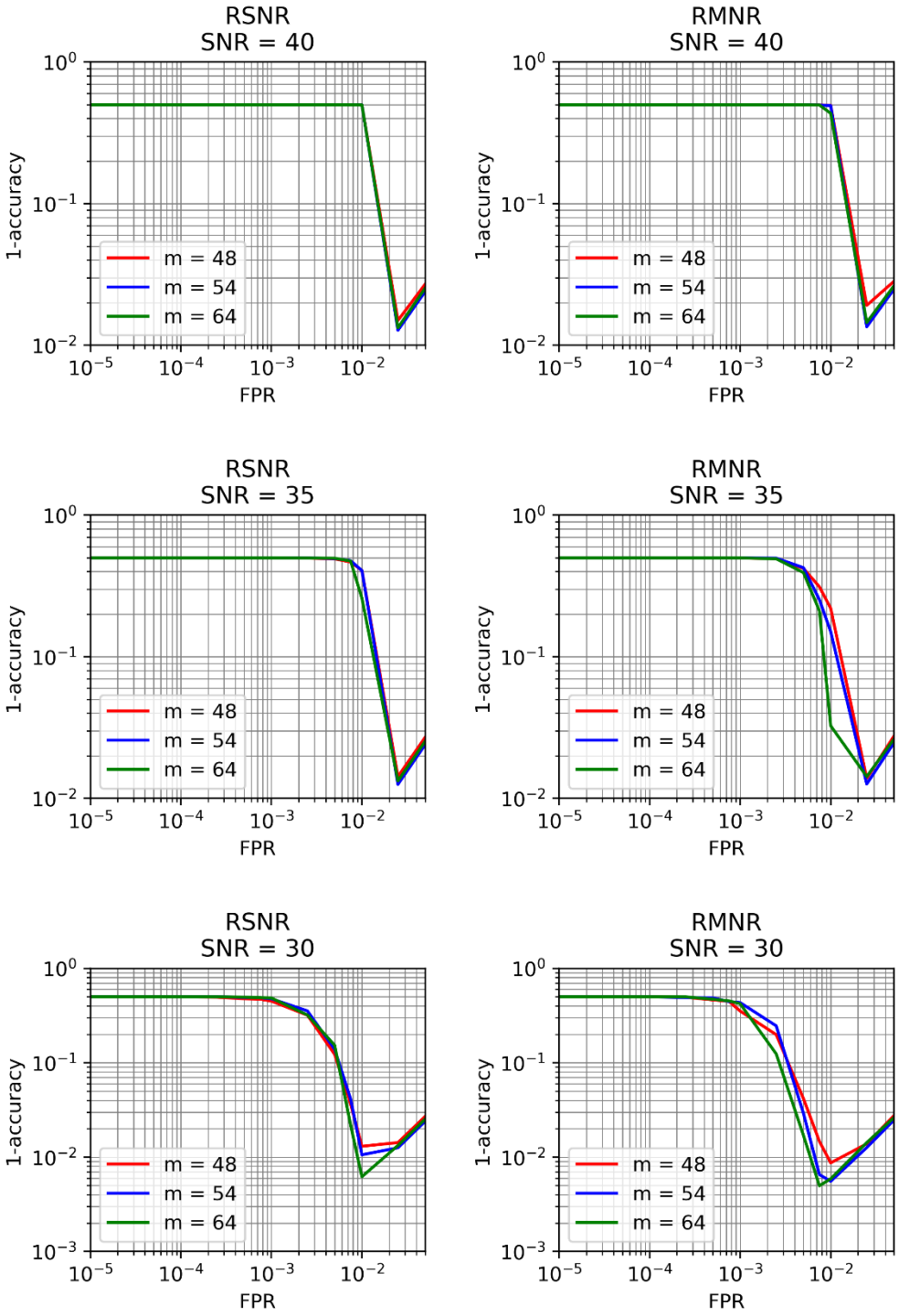

*Figura 4-2: Grafici logaritmici che mettono a confronto SNR e il complemento a l'accuratezza*

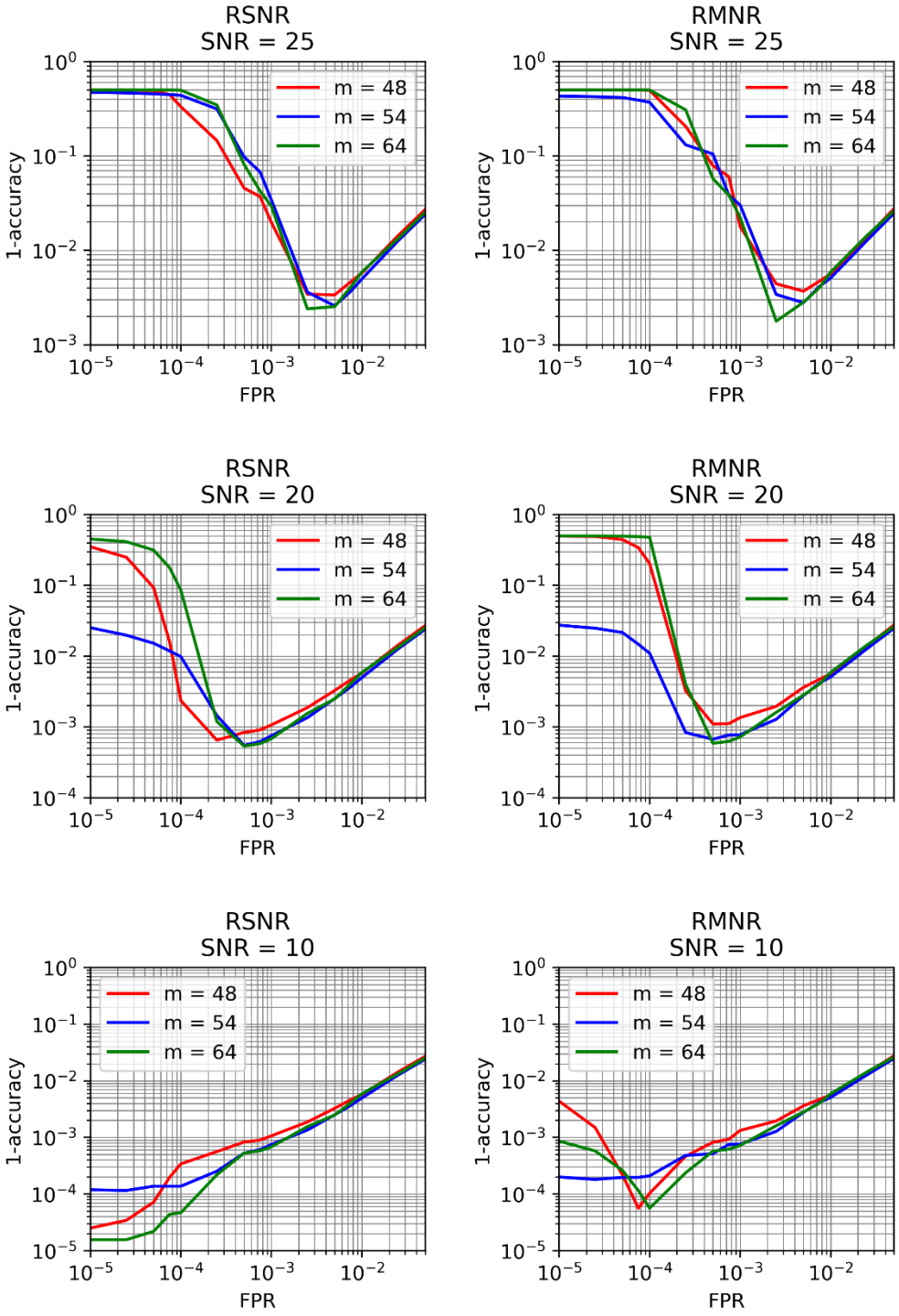
*Tutti i grafici dipendono dalla probabilità imposta inizialmente, dalla metrica utilizzata e dal fattore di compressione.*

Dopo aver verificato come il fattore di compressione e la metrica utilizzata siano abbastanza indipendenti dalla capacità di identificare anomalie, è risultato interessante studiare meglio come il variare della probabilità di veri positivi imposta inizialmente, rappresentata dalla complementare False Positive Rate (FPR), influenzi l'accuratezza del predittore binario. Di seguito sono riportati i grafici che rappresentano il complemento all'accuratezza in scala logaritmica al variare della probabilità imposta. Questi grafici sono parametrici al variare di $m$, della metrica e della quantità di rumore aggiunta.

In questo caso si può chiaramente notare che nella maggior parte dei grafici esista un valore di probabilità ottimo intermedio che ottimizzi l'accuratezza per ogni variabile considerata. In particolare è notevolmente interessate osservare come questo valore si sposti al variare del SNR verso probabilità più alte. D'altronde è abbastanza ragionevole come comportamento se si ricorda come i cluster si dividono all'aumentare del rumore, come spiegato nel paragrafo precedente. Proprio grazie a questo è possibile richiedere soglie che garantiscano inizialmente un minor numero di falsi positivi quando il rumore è elevato, permettendo all'accuratezza di beneficiare di questa scelta. Qualora invece il livello di rumore è tale da non dividere sufficientemente bene le due distribuzioni, richiedere una probabilità di falsi positivi inizialmente troppo alta risulta controproducente, abbassando eccessivamente la soglia del RMNR e originando un eccesso di falsi negativi.

Proprio per il trade-off appena indicato risulta interessante visualizzare, mediante dei grafici ad assi logaritmici, come la FPR e la FNR si comportino al variare degli altri parametri. Quello che si vorrebbe è che la somma di queste due fosse il più piccola possibile ma, da come si può notare, al decrescere di una l'altra cresce, rendendo necessario l'individuazione di un punto di ottimo. Questi grafici mettono in risalto come il punto di ottimo sia relativo alla particolare applicazione e alla figura di merito che si decide di ottimizzare. Infatti è possibile che i falsi negativi e i falsi positivi abbiano un peso diverso nella particolare applicazione, rendendo necessaria una diversa funzione di ottimizzazione e un posizionamento diverso sul piano FPR-FNR.


RSNR
SNR = 40
RMNR
SNR = 40
RSNR
SNR = 35
RMNR
SNR = 35
RSNR
SNR = 30
RMNR
SNR = 30
1-accuracy
FPR
m = 48
m = 54
m = 64


RSNR
SNR = 25
RMNR
SNR = 25
RSNR
SNR = 20
RMNR
SNR = 20
RSNR
SNR = 10
RMNR
SNR = 10
m = 48
m = 54
m = 64
1-accuracy
FPR

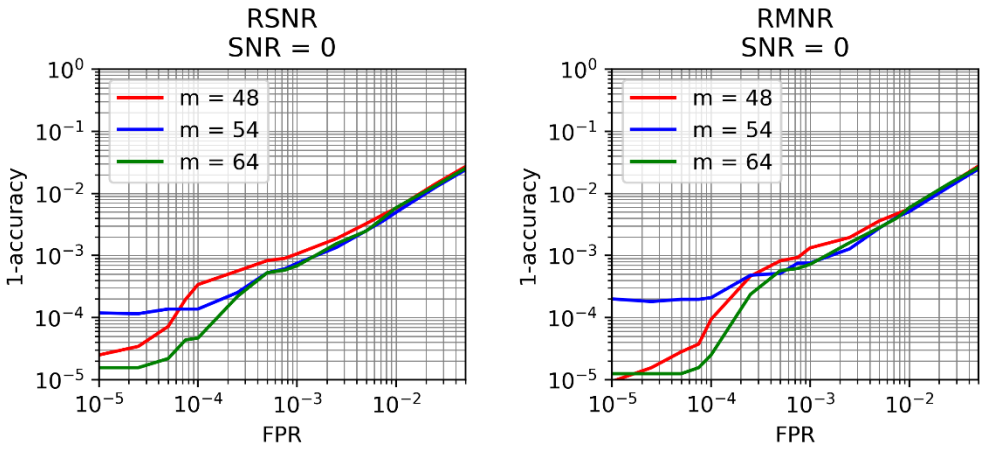
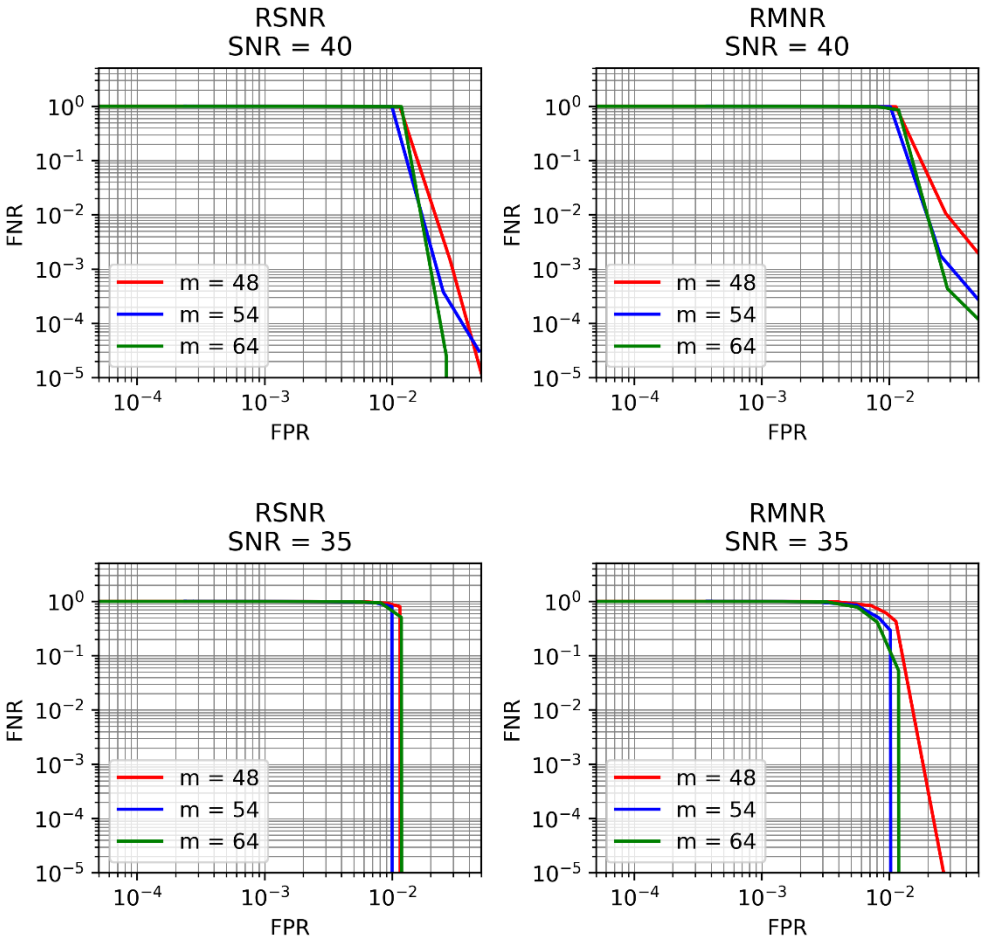



*Figura 4-3: Complemento all'accuratezza al variare della FPR in un diagramma log-log*

*I parametri che caratterizzano ciascun grafico sono la figura di merito utilizzata, il livello di rumore additivo considerato e la compressione impiegata.*



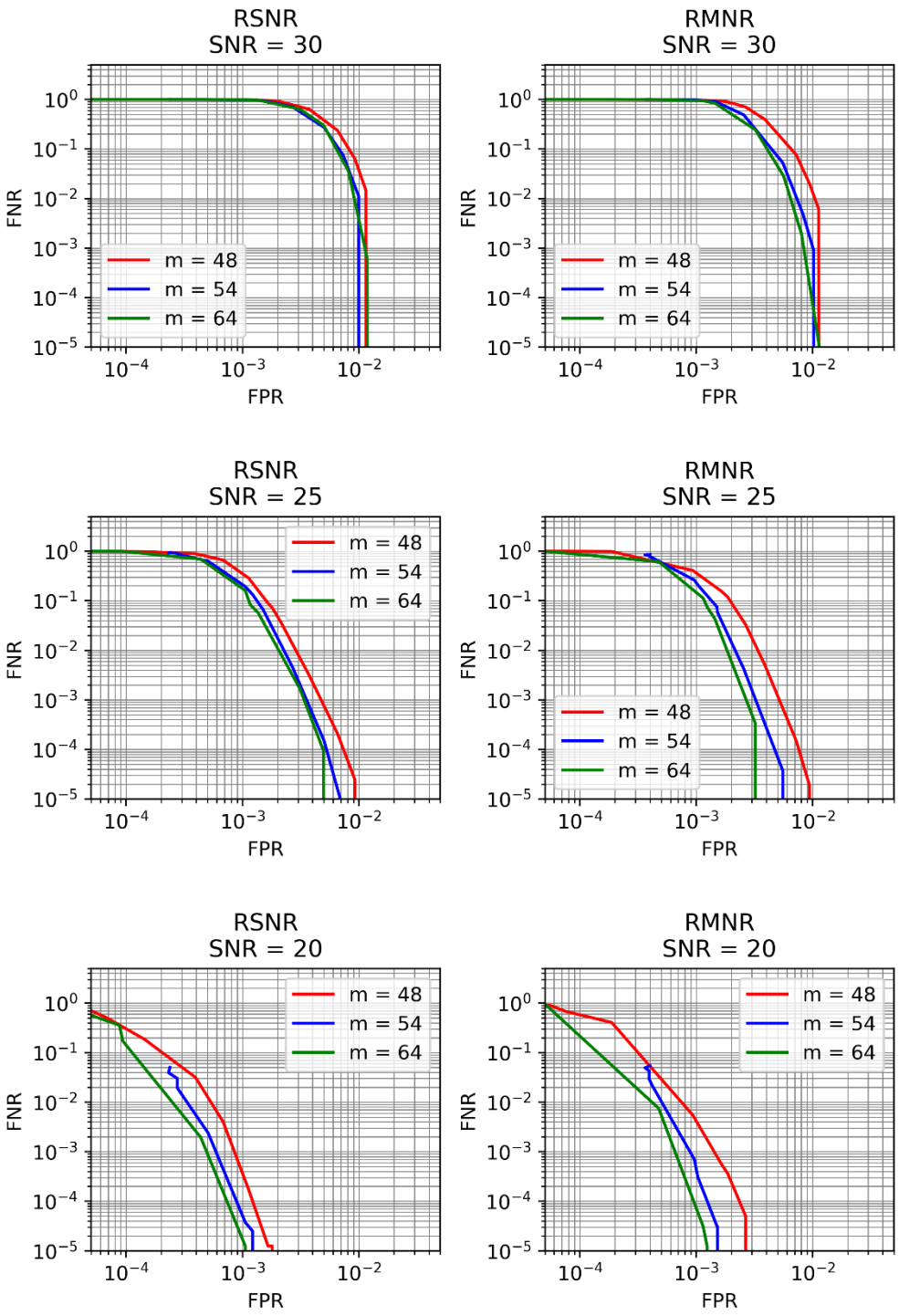



*Figura 4-4: Rappresentazione delle FPR e FNR al variare degli altri parametri in grafici log-log.*

*I grafici per valori di SNR elevati non sono stati riportati in quanto presentano un gran numero di punti mancanti dal momento che i valori sono talmente piccoli da essere interpretati come degli zeri di macchina*

# 5 CONCLUSIONI

Lo studio ha permesso di valutare in maniera approfondita la possibilità di impiego di piattaforme edge nell'esecuzione di reti neurali, considerando poi il caso applicativo specifico del compressed sensing. Dopo aver valutato diverse proposte commerciali che si adattano pienamente al paradigma dell'edge computing, è stato possibile analizzare come queste rappresentino in buona parte una soluzione concreta alla realizzazione di un sistema di calcolo distribuito che faccia uso di compressed sensing, verificando come questa tecnica sia effettivamente una soluzione valida e facilmente attuabile e aprendo la strada a molteplici applicazioni specialmente in ambito internet of things. Oltre a questo è stato possibile approfondire la possibilità di utilizzare il compressed sensing non solo come algoritmo di compressione del segnale ma anche come strumento per la rilevazione di anomalie. Anche in questo caso i risultati sono estremamente promettenti e permettono di aggiungere funzionalità a questa tecnica senza aumentarne lo sforzo computazionale.

Entrando più nello specifico si è visto che, se si considera il rumore additivo bianco come fonte di anomalia, è possibile ottenere un'accuratezza del predittore particolarmente interessante quando il SNR scende sotto di 30 dB, risultando probabilmente meno efficiente di altre tecniche di rilevamento di anomalie, ma comunque sufficiente in buona parte delle situazioni. Questo è particolarmente interessante se si considera che il compressed sensing nasce come tecnica di compressione del segnale, la cui possibilità di individuare anomalie risulta essere solamente una funzione aggiunta.

Per riassumere le potenzialità degli argomenti trattati è stato realizzato un esempio applicativo. Per il nodo edge si è impiegata una Raspberry Pi 4, scelta prevalentemente per la sua completezza a livello di connettività, con framework TensorFlow Core e modelli non quantizzati. Il nodo sensore è stato emulato mediante computer, e connesso tramite collegamento Bluetooth alla Raspberry Pi 4. Su entrambi i dispositivi sono stati eseguiti script Python per l'implementazione di un sistema completo di comunicazione che facesse uso di compressed sensing per segnali ECG ed è stato possibile verificare come la piattaforma edge fosse in grado di ricostruire correttamente il segnale e allo stesso tempo permettesse di individuare anomalie in caso il SNR calasse oltre certi livelli.

Le enormi potenzialità dimostrate dal compressed sensing unite alla disponibilità di soluzioni hardware che permettono di implementarlo a costi relativamente contenuti lo rendono un sistema di elaborazione del segnale maturo e pronto all'impiego in numerose applicazioni reali. Specialmente per le tendenze individuate nel primo capitolo, secondo le quali il paradigma dell'edge computing prenderà sempre più piede soprattutto nel crescente campo dell'internet of things, il compressed sensing e le piattaforme edge considerate per la sua implementazione rappresentano un'ottima risposta alle necessità che i nuovi paradigmi di calcolo presentano, soprattutto considerando la bassa latenza e i ridotti consumi energetici.

# INDICE DELLE FIGURE

# INDICE DELLE TABELLE

# BIBLIOGRAFIA

[1] K. Seeburn, «Edge Computing: How Is The Revolution Or Evolution Knocking On Our Doors?,» African Academic Network on Internet Policy, 2019. [Online].

[2] M. Bursać, D. Milošević e K. Mitrović, «Proposed Model for Automatic Learning Style Detecting Based ok Artificial Intelligence,» in *Science and Higher Education in Function of Sustainable Development*, 2019.

[3] M. Nielsen, «How the backpropagation algorithm works,» 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap2.html.

[4] S. R. Lee, «Training Neural Network,» 2020. [Online]. Available: https://www.endtoend.ai/mooc/dlnd/5/.

[5] M. H. N. Taha, «Improving QoS of Data Transmission over Wireless Sensor Networks,» 2013.

[6] Redazione Elettronica Open Source, «Wireless sensor network,» [Online]. Available: https://it.emcelettronica.com/wireless-sensor-network.

[7] RightScale, «State of the Art Cloud Report,» 2018.

[8] N. Bavis, «Alibaba Cloud Market Share 2019,» 2019. [Online]. Available: https://medium.com/@jaychapel/alibaba-cloud-market-share-2019-25d30bc096f7.

[9] M. Mangia, L. Prono, A. Marchioni, F. Pareschi, R. Rovatti e G. Setti, «Deep Neural Oracles for Short-window Optimized Compressed Sensing of Biosignals,» *IEEE Transaction on biomedical circuit and systems*, 2019.

[10] C. N. Höfer e G. Karagiannis, «Cloud computing services: taxonomy and comparison,» *Future Net Service Models & Design*, 2011.

[11] I. Godfellow, Y. Bengio e A. Courville, Deep Learning, 2017.

[12] E. Hamilton, «What is Edge Computing: The Network Edge Explained,» Cloudwards, 2018. [Online]. Available: https://www.cloudwards.net/what-is-edge-computing/.

[13] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal e H. Flinck, «Mobile Edge Computing Potential in Making Cities Smarter,» *IEEE Communications Magazine*, 2017.