DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Autoencoder/Classificatori Federated per Reti di Sensori Visivi

Relatore: Prof. Simone Milani

Laureando: Luca Zanetti

Anno Accademico 2021/2022

Data di Laurea: 21/02/2022

Indice

List of Tables

List of Figures

1 Introduzione

Il mondo odierno sta affrontando in questi anni un passaggio fondamentale verso la più grande rivoluzione industriale digitale, l'Internet of Things, in cui ogni oggetto dotato di sistemi di comunicazione potrà interagire con il resto della rete.
Questo cambio di paradigma avrà un impatto economico, sociale e strutturale poiché rivoluzionerà ogni settore della comunità coinvolgendo tutti i settori produttivi e socio economici.
Si assisterà ad un'esplosione di contenuti digitali condivisi che dovranno essere quindi gestiti, conservati e condivisi avendo particolare attenzione alla sensibilità delle informazioni che essi contengono.
É qui che trovano sbocco particolari tecniche di prevenzione della privacy quale ad esempio il Federated Learning.
Lo studio di tesi proposto, il quale si basa sul paradigma di apprendimento condiviso, cerca di ottimizzare una particolare struttura di apprendimento condiviso da integrare in un sistema di sensori visivi.

2 Il Federated Learning

2.1 L'apprendimento nel passato

Il primo sistema di condivisione del sapere volto a migliorare la comprensione possiamo affermare esser stato l'apprendimento cooperativo.
I precursori di questo metodo furono individuati in John Dewey, Kurt Lewin, Jean Piaget e Lev Vygotsky.
Dewey promosse nel suo metodo di istruzione, la cooperazione in gruppi come parte del suo famoso metodo di istruzione.
Si concentrò sullo sviluppo sistematico della concettualizzazione di feed-back nei processi di azione di gruppo cioè la relazione tra l'obiettivo e ogni step delle azioni nel processo e nel progresso di gruppo.
Vygotsky sosteneva che imparare è un atto sociale e non deve essere portato avanti da soli.
Ciò rappresenta in un certo senso il fulcro del *collaborative learning*.
Tra gli elementi basilari dell'apprendimento cooperativo si trovano:

- Interdipendenza Positiva: è la percezione di essere legati ad altri in modo tale che se tu non hai successo nemmeno loro lo ottengono.
- Responsabilità individuale: viene valutata la performance di ogni studente e i risultati sono condivisi con il gruppo e l'individuo in modo tale da capire chi avrà più bisogno di assistenza.
- Interazione promozionale faccia a faccia: per dare l'opportunità maggiore allo studente di promuovere il successo dell'altra persona aiutandola, supportandola e incoraggiandola.
- Social Skills: per contribuire al successo di uno sforzo cooperativo sono necessarie abilità interpersonali e di gruppo.
- Elaborazione del Gruppo: i membri del gruppo discutono su quanto bene stanno raggiungendo i propri obiettivi e se stanno mantenendo relazioni di lavoro effettive.

Una meta-analisi del Cooperative Learning Center all'Università del Minnesota[20] concluse che avere studenti che collaborano insieme è più fruttuoso rispetto ad avere studenti che

lavorano individualmente.
In particolare si è visto che, se collaborano in aggregazione, gli studenti riescono a imparare più materiale e si è più certi che capiscano tutti gli argomenti trattati rispetto ad un lavoro personale.
Trovano inoltre più motivazioni dal lavoro in gruppo nell'apprendere nozioni e hanno maggiori attitudini positive.
Uno studio approfondito effettuato da Totten, Sills, Digby, e Russ nel 1991[39], spiega che gli individui all'interno di un contesto di studio collaborativo riescono a capire un argomento ad un livello più profondo avendo un tasso più alto di risultati e mantenimento dei concetti appresi.
Tutto ciò ci spinge a pensare ad una similitudine diretta tra questa metodologia e l'apprendimento centralizzato nel mondo digitale interconnesso, dove, gli attori sono diversi, ma si ha come fondamento un paradigma piuttosto simile.
Mentre nell'ambiente del cooperative learning infatti si cerca di costruire un modello di sviluppo dello studio basato su piccoli gruppi, nell'apprendimento federato, seppur rimanendo il concetto di cluster, viene considerato un bacino di attori notevolmente maggiore.
Si può quindi considerare il sistema federato una evoluzione digitale dell'apprendimento cooperativo, dove non sono più gli individui centrali al processo di apprendimento ma, gli oggetti digitali.

2.2 Big data e IoT

Secondo le stime del GSMA, la copertura Internet mondiale è destinata a raggiungere la percentuale del 90% entro il 2030.
Inizialmente, i sistemi ICT sono stati concepiti come strutture a sè stanti.
Un enorme cambiamento è stato dato dalla tecnologia dei dispositivi mobili, che ha permesso lo scambio e l'elaborazione ubiqua di grandi quantità di dati.
In questo cambiamento tecnologico trova spazio l'Internet of Things, inizialmente chiamato "Internet of Everything" o Internet delle Cose, un sistema di rete esteso appoggiato ad Internet con la finalità di fornire una interazione real-time tra gli oggetti, le macchine e gli umani attraverso varie interfacce e protocolli creando una rete di calcolo onnipresente.
Data la struttura del nuovo paradigma, è cresciuto l'interesse verso l'elaborazione di grandi

quantità di informazioni che ogni dispositivo connesso alla rete può trasmettere, aumentando la consapevolezza riguardo i Big Data.

I servizi che fanno uso di questa tecnologia ormai sono molti, dal merchandising, la vendita online, i servizi finanziari, i servizi postali, il risparmio intelligente, l'efficientamento energetico e tanti altri ancora, inglobando completamente la vita dell'individuo in un mondo digitalizzato.

Si stima che il totale di abbonamenti unici di dispositivi mobile possa raggiungere la cifra di 5.8 miliardi verso il 2025 raggiungendo quindi una copertura di circa il 70% dell'intera popolazione mondiale con un incremento di 610 milioni di nuove sottoscrizioni mobile rispetto al 2019.

Per quanto riguarda l'Internet of Things si pensa raddoppierà il numero di oggetti interconnessi in circa 6 anni con un volume che oscilla tra i 25 e i 40 miliardi, una crescita dei dati generati in rete che varia dai 18.3ZB ai possibili 75ZB del 2025 e connessioni IoT aziendali che nel 2024 probabilmente sorpasseranno quelle dei consumatori toccando i 13,3 miliardi entro circa 4 anni.

Il dato estremamente interessante riguarda però la quantità media dell'uso di dati mobili a livello globale che quadruplicherà il suo ammontare a causa dell'incremento dell'utilizzo di smartphones e della disponibilità di accesso facilitato a servizi di networking ad alta velocità.

La società McKinsey si aspetta che il mercato dell'Internet delle Cose generi dai 3.9 agli 11.1 trilioni di dollari all'anno entro il 2025 con la possibilità quindi di raggiungere l'11% dell'economia globale dove il 40% dello sviluppo interesserà le economie crescenti.

Le proiezioni statistiche della GSMA[4] dicono che nel 2025 si possano già raggiungere 1.1 trilioni di dollari di guadagno per i servizi applicativi di questo mercato con 1.1 miliardi di città smart, 12.5 miliardi di industrie, 1.2 miliardi di veicoli, 3.4 miliardi di dispositivi elettronici di consumo, e 5.41 miliardi di case connesse.

Seppure i dati trasmessi aumentino di anno in anno, la richiesta globale di soluzioni per lo storage sta crescendo ad un tasso minore rispetto alla generazioni dei dati digitali poiché la maggior parte della tecnologia si sta orientando a soluzioni di dati transitori (ad esempio con piattaforme di streaming) che non necessitano di essere immagazzinati.

La diversità dei formati di dato, i metadati, la semantica, i diritti di accesso, l'hardware di calcolo associato, l'insieme di strumenti software per la gestione, la visualizzazione e l'analisi

dei dati, nell'insieme, sono un fattore chiave per la scalabilità e la complessità delle sfide lanciate dai BigData.

Nel contesto dei Big Data, i dati prodotti dall'uomo (email, documenti, testo, dati sui social

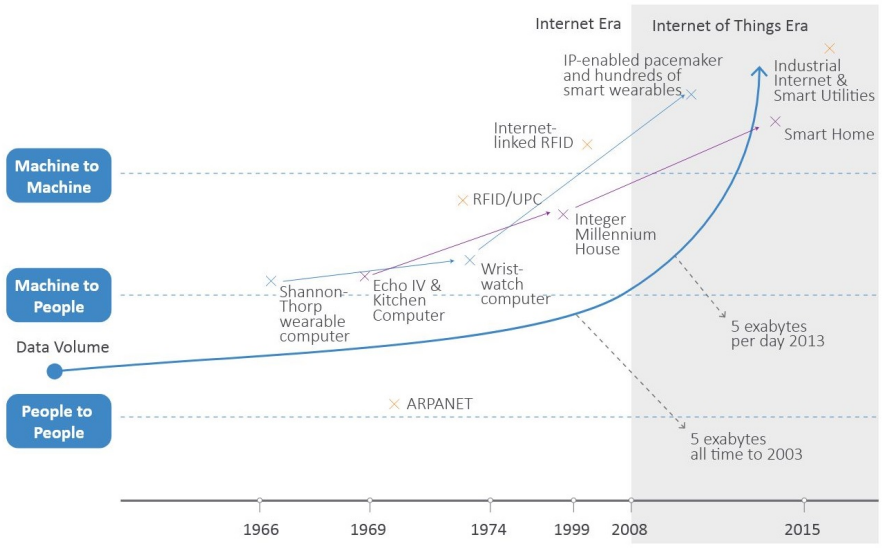


Figure 2: Sviluppo temporale dell'IoT

media, immagini, video ecc) rappresentano una percentuale sul totale sempre più ridotta, e ciò combacia con il fatto che la maggior parte dei dati verrà generata dall'interazione tra oggetti connessi.

Si pensi infatti alle miriadi di sensori remoti di vulcani e foreste, atmosferici, sismici, d'immagine e video di sorveglianza, traffico, i quali condividono i nodi di rete senza necessità di intervento umano.

Per il campo automobilistico l'obiettivo è quello di arrivare ad ottenere dei sistemi di trasporto totalmente indipendenti grazie alle trasmissioni ad elevata frequenza.

Per realizzare il tutto sono necessari sistemi Vehicle-to-Vehicle(V2V) per il rilevamento di veicoli in avvicinamento, Vehicle-to-Pedestrian(V2P) per rilevare la presenza di passanti, Vehicle-to-Network(V2N) per controllare il traffico presente e Vehicle-to-Infrastructure (V2I) per monitorare i segnali stradali lungo il tragitto.

Il progresso della tecnologia dell' automotive è fortemente correlato dall'Intelligenza Artificiale la quale consente di prendere decisioni effettive in base al riconoscimento real-time dell'ambiente circostante, il rafforzamento della sicurezza di guida e non solo.

Nel campo dell'energia, l'uso dell'IoT può portare un beneficio dal punto di vista del risparmio

e della gestione intelligente degli ambienti urbani ed extraurbani.

I flussi di traffico verrebbero gestiti in maniera più efficiente evitando la congestione e abbassando l'inquinamento dei veicoli a motore e nelle strade potranno essere implementati sistemi smart di illuminazione per ridurre la domanda di potenza elettrica limitandola solo alla richiesta necessaria.

Negli edifici si ridurrebbero i costi energetici attivando i sistemi di riscaldamento/raffreddamento in base alla presenza o alla temperatura mentre in agricoltura il monitoraggio dei raccolti con immagini termiche e sensori potrebbe consentire ai coltivatori una coltivazione più accurata ed eco-friendly.

Inoltre, i sistemi di irrigazione verrebbero migliorati apportando un uso migliore delle risorse idriche a disposizione.

Il progresso tecnologico raggiunto finora da questo paradigma digitale è stato possibile principalmente con la crescita esponenziale della tecnologia interna ai Mobile Devices, la quale ha permesso l'invio di Immagini, Video, Documenti e contenuti audio aventi una qualità elevata. Ciò ha consentito l'elaborazione accurata di modelli di apprendimento automatici basati su grandi set di dati.

La varietà dei fenomeni collegati all'IoT può inoltre essere collocata su uno sfondo storico, in

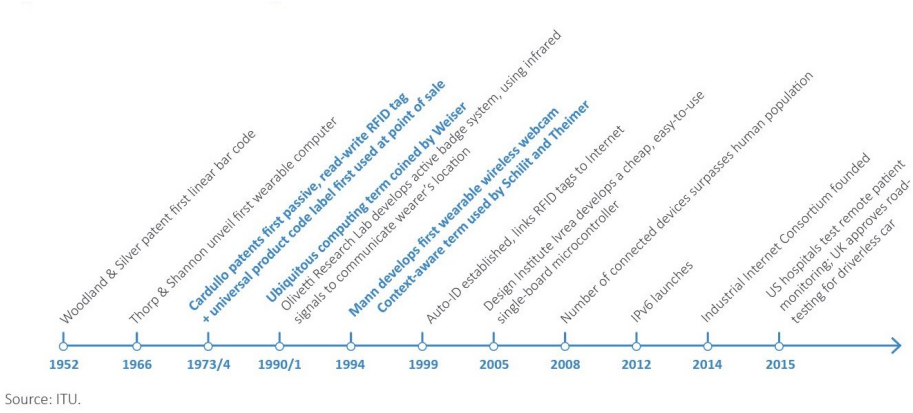


Figure 3: Timeline dello sviluppo che portò all'IoT

particolare rispetto all'evoluzione della comunicazione, mutata da un'interazione da persona a persona verso lo scambio da macchina a macchina.

Le prime piattaforme basate sulla rete Internet come il world wide web all'inizio si focalizzarono principalmente sulle comunicazioni tra individui e gruppi di persone, che può essere tradotta in comunicazioni tra persona e persona.

Diversamente, l'IoT consente ai dispositivi di effettuare comunicazioni tra persona e macchina cosi come tra macchina e macchina (M2M) senza l'intervento umano.
I device che utilizzano il sistema di comunicazione *Machine to Machine(M2M)* tipicamente utilizzano una comunicazione uno a uno con un modulo hardware embedded(per esempio un modulo identificativo o una sim card).
I fattori fondamentali che portarono al fenomeno di cambio di paradigma furono la crescita esplosiva dell'architettura interna (microprocessori DSP) ai dispositivi mobili con potenza sempre minore e performance maggiori, la scalabilità, e ampia disponibilità delle soluzioni di connessioni senza fili (Wireless), e infine la gestione di milioni di dispositivi basata sul cloud computing.
Un altro elemento da considerare fu la decrescita del costo dei sensori che precedentemente condizionavano il costo dell'intero dispositivo.
Si pensi che un sensore di immagini VGA da 0.3 megapixel alla fine degli anni 90 costava intorno ai $100, mentre nel 2020 il prezzo è calato fino a raggiungere 0.5$ ad unità.
Nel 2021, ogni smartphone integra almeno 9 sensori tra i quali troviamo il sensore di prossimità, il rilevatore di luce ambientale, il sensore di suoni ambientali, l'accelerometro, il sensore di temperatura e umidità, il giroscopio, il magnetometro e altri ancora.
Nell'ecosistema degli oggetti connessi, una delle frontiere più promettenti è rappresentata dalla condivisione di informazioni utili all'addestramento di modelli complessi che possono portare a risultati di calcolo migliori.
Possiamo individuare tre segmenti temporali successivi ai primi anni 2000 riguardo allo sviluppo di questa tecnologia.
Il primo periodo può essere individuato dal 2002 (anno della prima pubblicazione) fino alla fine del primo decennio del ventunesimo secolo durante il quale vennero pubblicate solo 9 ricerche, un secondo periodo viene identificato tra il 2009 e il 2015 nel quale si iniziarono a vedere i primi articoli e un terzo periodo dal 2015 ad oggi dove la quantità di articoli pubblicati è cresciuta esponenzialmente.
Per ovviare al problema dell'enorme quantità di informazioni digitali, della larghezza di banda, dello storage remoto e della capacità di immagazzinamento dei dati si è orientati a servizi relativi all'uso dei Big Data come il "fog computing" o "edge computing" e il "Distribuited Ledger".

Si tratta di sistemi di data center distribuiti nel territorio in modo tale da alleggerire il carico sui data center centrali attraverso una pre-elaborazione e inoltro dei dati più rilevanti su una rete WAN collegata all'unità di elaborazione centrale.

In quest'ottica trova spazio l'apprendimento federato, in inglese *Federated Learning*, una tecnica che utilizza il calcolo centralizzato e decentralizzato per l'elaborazione di grandi quantità di dati attraverso la preservazione della privacy.

Già nel primo decennio degli anni 2000, nel pieno dello sviluppo tecnologico, si iniziava a trattare l'accentramento del calcolo computazionale e i suoi benefici nei sistemi di database federati per ottenere delle prestazioni migliori nell'esecuzione delle interrogazioni ai sistemi centrali di elaborazione.

Ne è un esempio il lavoro di Salehm et. all [41] dove è stato utilizzato un algoritmo di ottimizzazione delle richieste per un sistema di database federati.

Nel meccanismo di interrogazione (query) c'è appunto la necessità di accedere nel minor tempo possibile ad una data risorsa.

E' quindi di notevole utilità l'uso di un intermediario centrale che raccolga le informazioni da vari Repository e inoltri al richiedente le informazioni cercate riducendo il tempo di attesa.

Un altro ambito in cui si accennò all'apprendimento federato fu quello dell'e-learning ovvero l'apprendimento con l'ausilio di strumenti digitali nel quale i materiali di varie università vennero usati in un sistema di materiale didattico condiviso con i oggetti di apprendimento (LOs) usati per creare nuovi corsi o per aggiornare i preesistenti.

In un altro lavoro[31] relativo sempre ai processi di apprendimento viene presentato un sistema di rappresentazione visuale mash up nel contesto del Life Long Learning ovvero l'apprendimento a livello personale durante tutto il proprio arco di vita.

Il sistema prevede l'utilizzo di piattaforme diverse usate per fornire un processo di studio configurabile e adattivo per l'utente sulla base della combinazione tra preferenze, conoscenza e contesto.

Possiamo quindi affermare che questi studi sono stati i precursori dell'apprendimento federato poiché, il reperimento e l'accentramento di dati da diverse fonti, rappresenta la struttura portante del paradigma che darà, in futuro, l'impulso ad uno sfruttamento più raffinato delle risorse digitali a disposizione e la tutela dei contenuti sensibili.

2.2.1 Rischi Tecnologici dell'IoT

Nonostante l'Internet of Things ci presenta una miriade di benefici e peculiarità che possono portare miglioramenti ad ogni settore produttivo e ad ogni ambito della nostra vita, non vanno trascurati e soprattutto sottovalutati i rischi che questa nuova tecnologia introduce.
La diffusione di queste nuove applicazioni tecnologiche, e la loro efficacia, spesso dipende dalla disponibilità, qualità e sicurezza della rete sottostante.
L'architettura di un sistema di oggetti intelligenti è composta essenzialmente da quattro livelli.
Al livello più basso abbiamo i sensori, attuatori, possibilmente all'interno di oggetti che formano oggetti smart oppure i sensori e gli attuatori puri che sono essi stessi oggetti intelligenti.
Il numero enorme di oggetti smart porta l'IoT ad essere il più grande sistema creato dall'uomo in termini di entità coinvolte, necessariamente eterogenee, per quanto riguarda la dimensione, funzionalità, protocol stack, sistema radio, sistemi operativi, risorse energetiche, identità ecc.
Il livello successivo sono i *gateway* che comunicano nelle vicinanze degli oggetti che scambiano dati con la rete di accesso.
Le *Access Networks* formano il terzo livello che consiste di infrastrutture per reti fisse o mobili o sistemi basati sulla trasmissione satellitare.
Si tratta di reti locali, regionali o nazionali e possono essere Wi-Fi, LAN, reti telefoniche 2G 3G 4G , GPS, Galileo e in alcuni casi possono consentire la comunicazione diretta, senza passare per il gateway.
In seguito troviamo il livello chiamato *Next Generation Network (NGN)* rappresentato dalla rete Internet, cioè la rete pubblica globale IPv4 e molto presto la rete basata sul protocollo IPv6.
Un livello sopra devono essere presenti i componenti, rappresentanti il *Service Layer*, in cui vengono elaborati e raccolti i dati e attraverso i quali vengono controllate le decisioni prese al livello più basso.
E per finire troviamo l'*Application Layer* che offre varie interfacce per gli utenti umani e possibili capacità di calcolo.
Una questione generale in questo contesto è quando e dove sia necessario criptare dati trasmessi o salvati e quali tipologie di protocolli di sicurezza siano necessari ai vari livelli dell'architettura.

La rete di oggetti globalmente connessi è divisa in milioni gestori di domini, come case, città smart, reti energetiche, access point e reti di trasmissione in uno sviluppo sia dall'alto verso il basso che viceversa.

L'attenzione quindi viene posta intorno a quali conseguenze la costruzione bottom-up e top-down dell'IoT possa comportare in termini di privacy, sicurezza, e affidabilità (SPT).

Un'altra problematica riguarda il consumo dei sistemi IoT e la loro integrazione con l'infrastruttura esistente e la vulnerabilità dovuta a reti aperte e semplici.

Sebbene alcune delle applicazioni IoT possano sfruttare le reti a bassa larghezza di banda, molti sistemi di monitoraggio richiedono una quantità notevole di banda.

Inoltre, in presenza di altre applicazioni all'interno delle aree densamente popolate, esse richiedono una infrastruttura che possa soddisfare la necessità dovuta alla convivenza di più sistemi intelligenti operanti simultaneamente.

A parte l'infrastruttura sottostante, il successo dell'IoT dipende dalla resilienza e sicurezza della rete, dei servizi e delle applicazioni.

Nonostante sia richiesto un alto livello di tutela dal rischio nei servizi, essi usualmente si trovano nel cloud ovvero all'interno dei data center.

Tuttavia, ciò non sempre è possibile e i dati vengono raccolti, controllati fuori dai confini territoriali di competenza.

Questo potrebbe essere un problema per la gestione sicura e protetta degli stessi e ciò rientra nella sfera di controllo dei *Management Domain*. Bisogna impiegare risorse tecnologiche per

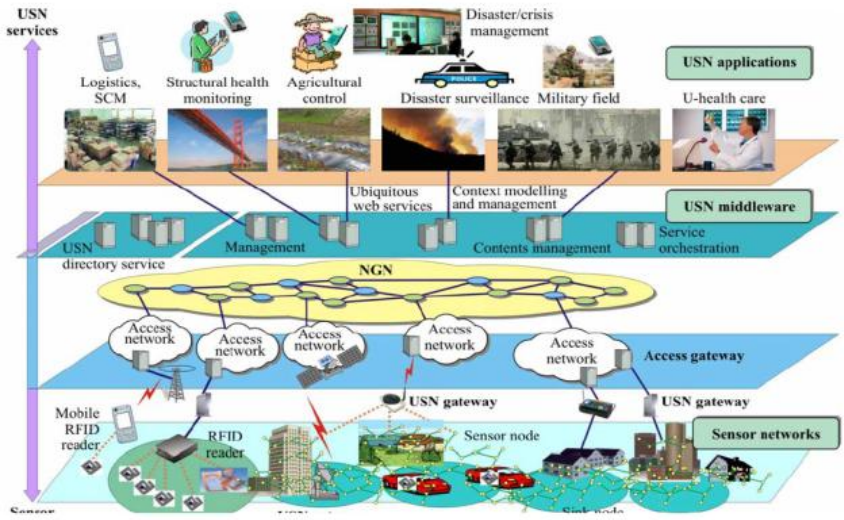


Figure 4: Architettura IoT [48]

mitigare le vulnerabilità informatiche che interessano le reti di sensori, le quali comporterebbero la seria compromissione dello sviluppo e affidabilità delle applicazioni IoT.
Quando si affronta l'argomento sicurezza nell'ambiente IoT è quindi buona norma considerare ogni nodo della rete.
Tecnologie come la Wireless Sensor Network (WSN) e il Identificazione con radiofrequenza (RFID), cosi come le letterature scientifiche collegate, stanno evolvendo con sviluppi in aumento nell'ambito di applicazione delle tecnologie legate ad Internet e devono essere protette da ogni tipo di minaccia informatica.
Il sistema IoT si basa sulla gestione di un'enorme mole di informazioni che possono essere usate per vari servizi, rendendo il paradigma IoT appetibile per possibili truffatori, come hacker occasionali, attivisti hacker, cybercriminali, i quali potrebbero essere interessati a intercettare informazioni sensibili, come dati di posizione, numeri di carte di credito, password di account bancari, manomettendo il dispositivo IoT.
Inoltre, essi possono cercare di compromettere i componenti dei nodi finali della rete al fine di lanciare un attacco ad una entità terza.
La comunicazione tra device smart che scambiano dati tra loro nella rete può comportare un rischio per l'intera infrastruttura qualora anche uno solo di essi venga violato.
C'è quindi la necessità di stabilire dei requisiti di sicurezza per ovviare a considerevoli problemi che una minaccia reale potrebbe causare ai settori che si appoggiano all'infrastruttura tecnologica.
L'avanzamento di altre tecnologie complesse come gli smart dust (piccoli componenti usati in campo militare per percepire l'ambiente, elaborare dati, e comunicarli ad altri componenti) e i tag RFID orientati ad una architettura multilivello ma sviluppati nell'ottica dell'Internet tradizionale, necessitano di essere adattati secondo le nuove dimensioni di efficienza, scalabilità, sicurezza e privacy per l'IoT.
Particolarmente per quanto riguarda l'Internet delle cose si possono verificare i seguenti attacchi o minacce:

1. *The mobile salesman*: Compromissione della sicurezza di connessioni wireless disponibili per il guidatore lungo il tragitto.

2. *Passive Collaboration in Opportunistic Networks*: Sicurezza dell'informazione condivisa peer-to-peer tra persone in una rete collaborativa.

3. *Patient Monitoring*: Sicurezza e privacy di dati personali dei pazienti in un ospedale.

4. *RFID-Based Warehouse Management*: Sicurezza dello stoccaggio e tracciamento merci di un impianto e al di fuori dello stesso.

5. Vulnerabilità in *Ambient Intelligence* nella vita di famiglia in diversi ambienti(casa, lavoro, al parco, durante un pasto).

6. Falle nel sistema di gestione del traffico con causa di incidenti, sollevando molti problemi relativi sia al viaggio che a sistemi intelligenti ambientali per la tutela della salute.

7. Violazione di informazioni personali, in una azienda di raccolta dati, ottenute da reti AmI e che sono parte del loro business principale.

8. Sfruttamento di vulnerabilità in stufe elettriche per la casa, prendendo il controllo di un intero insieme di stufe compromesse, e successivamente, in modo furtivo e simultaneo, portandole alla massima potenza per più tempo mentre le persone sono a lavoro.

9. Compromissione di un sistema sanitario con appuntamenti fittizi causando conseguenti disagi.

10. Attacchi DoS mirati organizzati da organismi, società, gruppi di intelligence.

Per il futuro gli sforzi per affrontare le minacce che si prospettano sono focalizzati nel risolvere rischi collegati ai telefoni cellulari e ai social network, cosi come ai pericoli dovuti al parallelismo, attacchi DoS, dati di sensori e RFID falsati, attacchi su reti wireless e reti di nuova generazione.
Altri attacchi con priorità più basse riguardano hardware e IPv6 maligni e host direttamente raggiungibili, giochi online, frodi elettorali, caduta dei mercati azionari, spionaggio industriale, smart grid, fuoriuscite di petrolio, falsificazione di fabbriche, attacchi contro centrali elettriche ed idroelettriche.
Conseguentemente a questo panorama sulla cybersecurity, l'attenzione maggiore per l'IoT va data ad attacchi man-in-the-middle che compromettono la riservatezza e l'integrità dei dati, eavesdropping, o alla presa di controllo di alcuni componenti dell'intero sistema.
Come prospettiva futura per proteggere l'Internet delle cose si dovrebbero sviluppare spazi di lavoro maggiormente sicuri, che possano affrontare i problemi relativi alla privacy attraverso

tutti gli ambiti.

Inoltre, la ricerca dovrebbe essere focalizzata allo sviluppo di sistemi che siano resilienti ad ogni tipo di minaccia.

La tabella 1 riassume i requisiti per la garanzia e sicurezza delle informazioni. La differenza

Confidenzialità	Garantire che solo gli utenti autorizzati abbiano accesso alle informazioni
Integrità	Garantire completezza, accuratezza, e assenza di manipolazione dati non autorizzata
Disponibilità	Garantire che tutti i sistemi di servizio siano disponibili quando richiesti da un utente
Responsabilità	La capacità di un sistema di ritenere gli utenti responsabili delle loro azioni
Verificabilità	Un'abilità di un sistema di condurre un monitoraggio persistente delle azioni
Affidabilità	La capacità di un sistema di verificare l'identità e stabilire fiducia in una terza parte
Confirmability	L'abilità di un sistema di confermare alcune azioni
Non-repudiation	Il verificarsi/non verificarsi di un'azione
Privacy	Garantire che il sistema rispetti politiche di privacy e consenta agli individui di controllare le informazioni personali

Table 1: Requisiti di garanzia e sicurezza dei dati.

sostanziale tra un oggetto sicuro e un attacco sicuro è che, un oggetto sicuro è un oggetto che soddisfa tutti i requisiti di sicurezza IAS-octave, mentre un attacco sicuro è un attacco che tende a minacciare almeno uno dei requisiti di sicurezza prima citati.

Nella progettazione di nuovi sistemi e servizi si deve quindi prestare particolare attenzione alla sensibilità delle informazioni coinvolte e devono essere presi in considerazione i requisiti richiesti dagli utenti.

2.2.2 La riservatezza dei dati

Il problema, anche se spesso espresso in forma di privacy, è un problema di controllo.
Se il sistema computazionale è tanto nascosto quanto esteso, diventa difficile sapere cosa sta controllando cosa, cosa è connesso con cosa, dove sta fluendo l'informazione, come è stata usata e quali sono le conseguenze di ogni azione.
Per l'IoT, trattandosi di nuovo paradigma, i metodi tradizionali di controllo non è detto che possano essere totalmente applicabili.
Infatti, ci sono casi comuni dove non sono più le persone ad essere utenti, ma piuttosto gli oggetti autonomi interagenti con il servizio, per esempio sensori di traffico, luminosità, temperatura ecc. e gli utenti non sempre sono consapevoli di quando un dispositivo stia raccogliendo dati personali o quali sensori siano presenti nelle sue vicinanze.
La privacy di base di Internet, risalente al passato al Fair Information Pratice Principles, sosteneva che la raccolta di dati personali dovrebbe avvenire soltanto con una notifica e scelta appropriate.
Oggi la legislazione all'interno dell'Unione Europea è gestita dalla direttiva 2002/58/EC che stabilisce le norme per la protezione e privacy dei dati nel settore delle comunicazioni elettroniche e dal 2016 il regolamento GDPR(General Data Protection Regulation), operativo dal 2018, il quale dipende in sostanza dalla direttiva essendo *lex generalis* e quindi non comportando obblighi supplementari.
La direttiva di cui sopra è stata tuttavia ritenuta fin da subito non sufficiente a realizzare la strategia di un mercato unico globale risultando troppo obsoleta visti gli sviluppi tecnologici ed economici come l'utilizzo diffuso di voice-over-IP, servizi di posta elettronica web, servizi di messaggistica nonché l'emergenza di nuove tecniche di web Profiling, comportando un rischio non da poco per la tutela di tutti i consumatori.
Un modello concettuale della privacy descrive come sistemi IoT sviluppati al giorno d'oggi interagiscano con gli utenti e comprende principalmente tre aree:

- Interfaccia utente - usata per impostare e comprendere le preferenze di privacy, per le notifiche e la visualizzazione
- Problema dell'apprendimento - come traslare i dati grezzi degli oggetti intelligenti in deduzioni comprensibili agli umani

- Schema della Privacy - come rappresentare la grande quantità di servizi IoT e le loro politiche di privacy

Viene rappresentata in figura 8 la struttura tipica della privacy nell'interazione con l'utente in un sistema IoT.

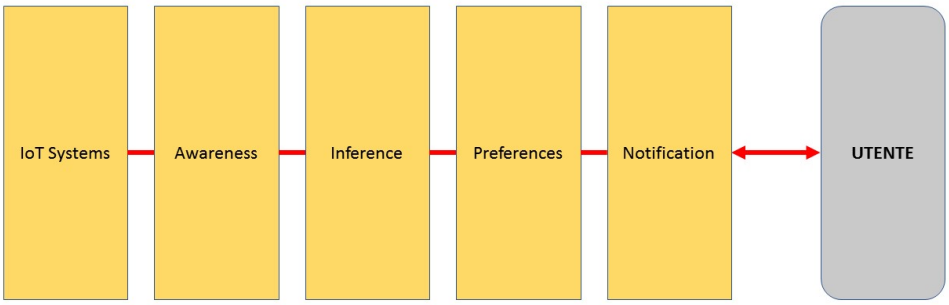


Figure 5: Struttura della privacy nell'Internet of Things

La maggior parte del mondo sviluppato ha cercato di proteggere i consumatori dall'uso illegale di informazioni confidenziali ma in molti casi le leggi non normano totalmente l'ecosistema e non consentono di evitare che i dati sensibili siano recuperati e utilizzati spesso all'insaputa degli utilizzatori.
Uno dei problemi nel web riguarda in particolare l'uso di cookies, che vengono utilizzati dalle aziende per tenere traccia del comportamento dell'utente.
Anche se la legislazione all'interno dell'Unione Europea regola l'uso dei cookies e quali dati degli utenti possono essere raccolti, con la crescita della tecnologia mobile, che non necessita dei cookies per tracciare il comportamento dell'utente, molte delle leggi sono diventate obsolete e vengono sistematicamente aggirate.
Le reti wireless invece necessitano di tutelare la privacy dell'utente attraverso l'implementazione di richieste anonime o pseudo-anonime che non consentano al proprietario della rete stessa oppure agli operatori, controllori, elaboratori o potenziali avversari, di apprendere informazioni dalla comunicazione.
Questi processi di progettazione basati sulla privacy potrebbero ridurre le preoccupazioni

non solo al livello di rete ma anche ad un livello architetturale più alto, scenario tipico delle smart city o delle parti bottom-up dell'IoT, quando il controllo dell'infrastruttura è diviso tra utente e proprietario.
La privacy relativa alla posizione della sorgente è un'altra area associata al livello della rete di sensori nell'architettura IoT dove si necessita di non rivelare quale dei nodi della rete abbia inviato i dati alla stazione base(*sink*).
Tuttavia, anche se i dati venissero criptati è possibile che un avversario estragga informazione solo dall'osservazione e dall'analisi del modello di traffico, riuscendo quindi a rivelare la posizione del nodo che, collegato alla persona, ne esporrebbe la posizione.
Si trova quindi una linea di demarcazione che l'IoT non oltrepassabile per evitare di violare l'etica e la libertà dell'individuo umano.
Ciò lascia spazio a dibattiti e riflessioni su quanto possa spingersi questo nuovo ramo tecnologico.
Per quanto riguarda le minacce relative alla privacy si possono identificare nelle seguenti tipologie:

- *Identification*: L'identificazione riguarda la connessione di un identificatore persistente (l'indirizzo e il nome o lo pseudonimo), di un individuo e un'informazione su esso. L'attacco consiste nell'associare una identità ad una specifica privacy, violando il contesto e può dar vita o facilitare altre minacce. Si verifica principalmente nella fase di elaborazione al livello backend, dove grandi quantità di dati viene raccolta in un punto centrale al di fuori del controllo del soggetto. La sfida principale affrontata nell'identificazione è la progettazione di un sistema intelligente che preferisca un'elaborazione locale a quella centralizzata, interazioni orizzontali piuttosto che verticali, in modo tale che al di fuori della sfera dell'utente i dati identificabili siano minori possibili. Questo potrebbe essere uno degli scenari in può essere d'aiuto la strategia adottata dal *Federated Learning*, ovvero prediligere una elaborazione locale per salvaguardare la privacy ai nodi della rete.

- *Localizzazione e Tracciamento*: La localizzazione e il tracciamento riguardano il pericolo di poter sapere e documentare la posizione di un individuo nel tempo e nello spazio attraverso il GPS, il traffico internet e le celle telefoniche. La maggior parte delle violazioni di privacy avvengono con questa modalità e possono riguardare lo staking GPS,

divulgazione di informazioni personali, la sensazione di stalking. La localizzazione e il tracking non causano di solito violazioni di privacy nelle immediate vicinanze ma emergono come rischio principalmente nella fase di elaborazione dell'informazione quando vengono create tracce di posizione al lato back end, fuori dal controllo dell'utilizzatore.

- *Profiling*: Denota il rischio di raccogliere o arrangiare dossier di informazioni sull'individuo e incrociarle con altri profili e dati per dedurne, attraverso la correlazione, gli interessi. Sono metodi usati principalmente per la personalizzazione nel campo dell'e-commerce (sistemi di raccomandazione, newsletter, banner pubblicitari ecc.) ma anche per l'ottimizzazione interna basata sulla demografia e gli interessi dell'utente. Esempi di quando il profiling danneggia la privacy possono essere discriminazione di prezzo, annunci pubblicitari indesiderati, ingegneria sociale, o decisioni automatiche errate ecc.

- *Privacy violating interaction and presentation*: Avviene quando dati sensibili personali vengono inviati, per mezzo di un mediatore pubblico, e vengono rivelati ad individui indesiderati. I dati dell'utente, spesso usati da più applicazioni IoT quali ad esempio aziende, infrastrutture, sistemi di assistenza sanitaria e medica, necessitano di molte connessioni con l'utente. É infatti concepibile che in questi sistemi alcuni dettagli della persona vengano forniti attraverso l'uso di cose intelligenti presenti nei loro dintorni o viceversa gli utenti possono imporsi nei sistemi attraverso l'utilizzo di oggetti smart nell'ambiente. Tuttavia, le intercomunicazioni a volte sono intrinsecamente pubbliche e rischiano di creare problemi di privacy qualora debbano essere scambiate informazioni segrete tra l'utente e il sistema.

- *Lifecycle transitions*: Si può verificare quando, durante il loro ciclo di vita, vengono rivelati dettagli segreti durante il cambio di proprietà. Questo problema viene notificato quando vengono viste immagini o video compromettenti nelle videocamere o in altri dispositivi. Dato che le contravvenzioni di privacy riguardanti il ciclo di vita sono principalmente dovute alla raccolta di informazioni, ciò dipende dal livello di informazione del modello di riferimento IoT.

- *Inventory Attack*: E' definito come raccolta dati non certificata riguardanti la realtà e le caratteristiche dei device personali. Attraverso i protocolli internet, organizzazioni autorizzate, governi e intelligence militari possono infatti indagare i dispositivi IoT in

rete ed effettuare query sui dispositivi di utenti o proprietari di sistemi certificati da ogni parte del mondo. Queste interrogazioni tuttavia non possono essere eseguite da organizzazioni non autorizzate le quali non possono entrare dentro i sistemi informatici riuscendo magari a redigere un registro dettagliato delle cose di un'area particolare (come ad esempio un edificio, spazi pubblici istituzionali, aree industriali ecc.).

- *Linkage*: In questo tipo di attacco, emerso anche dopo la diffusione dei Data Provider, si cerca di connettere insieme device formalmente separati. Gli utenti sono ignari della perdita dati e dell'ispezione superiore quando tutti i dati e autorizzazioni diverse vengono messi insieme. Le minacce Linkage si aggraveranno durante l'evoluzione dell'IoT principalmente perché sistemi di varie organizzazioni verranno sovrapposti per generare compagini diversificate capaci di fornire nuovi servizi che nessuno dei sistemi è in grado di fornire da solo.

Un mondo che evolverà verso la condivisione di sempre più informazioni e dati sensibili deve tutelare il cittadino, immerso in questo mare tecnologico.
Nel corso degli anni sono state proposte alcune soluzioni per affrontare le preoccupazioni riguardo la privacy e le considerazioni fatte dai service provider:

- *Tecniche crittografiche e manipolazione dell'informazione*: La crittografia è ancora il metodo predominante tra tutte le soluzioni per la tutela della privacy, anche se per la maggior parte dei problemi alcuni sensori non riescono a farne uso a causa di limitazioni di memoria o potenza di calcolo.

- *Consapevolezza della privacy o del contesto*: Le soluzioni per la consapevolezza della privacy si sono orientate principalmente in applicazioni utente supportate da una sensibilizzazione di base per il loro uso negli smart device principalmente per la persona o la casa.

- *Controllo di accesso*: Il controllo di accesso può essere usato in simbiosi con la crittografia e la consapevolezza alla privacy e ciò dà all'utente la possibilità di gestire i propri dati.

- *Minimizzazione dei Dati*: I fornitori di servizi IoT, possono limitare o ridurre la concentrazione di dati sensibili o di quelli di rilevante importanza. Queste organizzazioni

potrebbero inoltre conservare i dati per la durata di tempo per i quali siano necessari a svolgere il servizio tecnologico a favore dell'utente.

Seppure le tecniche a favore della protezione della privacy siano abbastanza efficaci, risulta tuttavia difficile annullare o quantomeno ridurre notevolmente il rischio che i dati trasmessi in rete vengano intercettati, manipolati e usati a scopi malevoli da terze parti.
E' tuttavia emersa una nuova tecnica, il *Federated Learning*, che permette di preservare i contenuti coperti da privacy ed elaborare i dati dell'utente o, quantomeno del dispositivo, facendo transitare in rete informazioni utili al sistema di elaborazione centrale ma prive di dati sensibili.
Quello della privacy resta comunque un problema aperto poiché la cybersecurity non dura illimitatamente e vista l'accelerazione del progresso IoT non è detto che i sistemi riescano ad adeguarsi al cambiamento.

2.2.3 La Cybersecurity nell'IoT

Le brecce cyber sono una delle maggiori minacce nel business al giorno d'oggi.
I danni economici a livello mondiale per quanto riguarda i crimini informatici vengono stimati in più di 300 miliardi di euro l'anno e sono destinati a salire.
Un report di IBM [7] afferma che, nel 2020, in media il costo di una violazione informatica ammonta a 3.86 milioni.
Dal punto di vista della sicurezza IoT la preoccupazione sorge poiché si tratta di violare sistemi apparentemente sicuri con più livelli di protezione attivi.
La complessità dei sistemi di sicurezza in questo ambito è un'area di sviluppo propensa a grandi profitti visto che l'ecosistema IoT avrà in futuro miliardi di oggetti connessi alla rete e tra loro.
Infatti, ogni device resta un punto di accesso potenziale per un hacker che vuole violare un sistema di protezione informatico.
Più certi sistemi, particolarmente quelli industriali, vengono automatizzati e connessi fra loro, più rischiano di essere oggetto di crimini informatici.
Si pensi alla rete intelligente di fornitura di energia elettrica in una città smart che viene utilizzata per trovare i guasti da risolvere e ridurre i costi.
Parallelamente l'infrastruttura intelligente deve essere protetta da eventuali attacchi che

potrebbero minare la stabilità dell'intera rete di approvvigionamento energetico e potenzialmente spegnere addirittura una città intera.
A causa infatti della natura dell'IoT dove ogni oggetto può usare dati di altri oggetti con cui comunica, gli effetti di un data-breach possono causare una reazione a catena causando problemi catastrofici nei sistemi.
Il malfunzionamento di un oggetto infatti causa la trasmissione di dati errata ad un altro oggetto.
Le reti di sensori wireless (Wireless Sensor Network) sono facilmente suscettibili ad attacchi di sicurezza IoT dovuto al fatto che vengono utilizzate trasmissioni a onde medie per il broadcasting.
Alcuni dei maggiori attacchi sono:

- *Physical Attacks*: Un sensore per dispositivi deve essere implementato in ogni oggetto per ottenere le sue totali capacità. Risulta tuttavia difficile attuare una protezione da accessi fisici non autorizzati e proteggere fisicamente il device. Un hacker può manipolare i dati di un nodo/sensore compromettendo il corretto funzionamento dell'intera rete di sensori a rischio.

- *Node Replication*: In questo attacco l'hacker crea una copia dell'identificativo di un sensore esistente e lo immette nella rete come fosse un nuovo sensore causando quindi una duplicazione che porta ad un errato inoltro dei pacchetti, registrando dati falsati dalla lettura del sensore, oppure una disconnessione, disturbando quindi le performance della rete di sensori.

- *Selective Forwarding*: Nelle WSN si assume che ogni nodo riceva i messaggi alla destinazione. Un nodo malintenzionato, nel suo attacco, può inoltrare pacchetti selettivi. Può semplicemente evitare di inviare alcuni pacchetti ed essere difficile da identificare poiché i pacchetti possono transitare attraverso vari nodi della rete.

- *Wormole Attack*: Si tratta di un attacco critico nel quale l'attaccante riesce a recuperare dei pacchetti in un punto all'interno della rete e poi fa un tunneling (protocollo per spostare pacchetti da una rete ad un'altra) verso un'altra destinazione. Questo processo può essere inoltre effettuato selezionando i pacchetti desiderati. Un attacco di questo tipo sul routing di messaggi di controllo va a danneggiarne la routine.

- *Sybil Attack*: Questo attacco venne introdotto nelle reti peer-to-peer e in particolare si verifica quando un computer viene sequestrato e l'hacker dichiara identità multiple facendo pensare di essere in più di un posto in una volta. Un singolo nodo che presenta più identità nella rete può causare una riduzione significativa nella tolleranza ai guasti come la memorizzazione distribuita, la disparità, e il multipath.

- *Sinkhole Attack*: Viene a verificarsi quando un intruso cerca di attrarre il traffico di nodi vicini attraverso il controllo su un nodo della rete. Questo processo può essere effettuato attirando altri nodi e usando algoritmi di routing attraverso attacchi di inoltro, modifica ed eliminazione selettiva di messaggi.

- *Service Attack denial or Denial of service Attack*: L'intento dell'avversario è quello di rendere indisponibile il servizio agli utenti abilitati attraverso l'inondazione di richieste legittime che portano al malfunzionamento dei collegamenti della vittima e ad una negazione del servizio verso gli utenti legittimati.

- *Eavesdropping*: L'intruso intercetta le informazioni durante la trasmissione di dati tra due nodi nella rete senza modificarne il contenuto. Viene quindi messa a repentaglio la privacy dell'utente e l'intruso può sfruttare l'informazione recepita contro il singolo o l'intero sistema.

- *Sleep deprivation torture attack*: Una parte dei dispositivi intelligenti hanno fonti energetiche limitate, funzionando con batterie a potenza limitata. L'attacco riguarda l'estorsione di energia nelle reti di sensori dove gli oggetti smart vengono forzati a comunicare con il gateway più frequentemente con una conseguente perdita maggiore di energia.

- *Jamming*: Un attacco rivolto sempre a dispositivi con energia limitata. Viene usato un segnale distorto in un'area vicina all'oggetto smart in modo tale che esso non possa comunicare con i nodi vicini (sensori, gateway, stazioni base, satelliti di navigazione) pur utilizzando la sua massima potenza di trasmissione.

Gli attacchi contro la tecnologia RFID sono i seguenti:

- *Physical data modification*: L'attaccante riesce ad ottenere e alterare i tag. Viene utilizzata l'induzione di guasto dati per modificare i dati fisici ovvero un processo di modifica

durate la scrittura o l'elaborazione. Ciò causa una non corrispondenza tra i dati salvati nel tag e gli oggetti ai quali il tag è attaccato compromettendo la tracciabilità del prodotto.

- *Obfuscation*: Riguarda l'uso di tecnologie di protezione ingannevoli.
- *Tag omission*: Caratteristiche di sicurezza non presenti negli articoli dei produttori contraffattori pur essendo presente negli articoli autentici.
- *Removal-reapplication*: l'attacco riguarda l'applicazione di caratteristiche di sicurezza autentiche da un prodotto originale ad un articolo contraffatto.
- *Tag cloning*: Si riferisce alla duplicazione di caratteristiche di sicurezza in modo tale che venga riconosciuto come autentico durante l'identificazione. Il tag originale viene sostituito con un nuovo tag e l'RFID originale viene copiato in quest'ultimo. C'è quindi bisogno di una protezione dall'accesso fisico.
- *Tag swapping*: Si tratta di un attacco comune nel quale i tag di due prodotti vengono sostituiti.
- *Denial of service attack*: Quando un lettore RFID richede un'informazione da un tag e riceve il suo codice identificativo, lo confronta con quello salvato nel suo database. Tuttavia, sia il lettore che il database del server sono vulnerabili ad attacchi DoS e quindi quando questo avviene il servizio subisce un'interruzione e il tag fallisce nell'inviare il suo id al lettore.

Responsabilità

Una questione importante riguarda milioni di device che, scambiandosi informazioni, rendono offuscata la linea di demarcazione per quanto riguarda la proprietà dei dati.
Sostanzialmente non è possibile associare un dispositivo con un dato, visto che la potenzialità dell'IoT sta proprio della trasmissione tra oggetti.
Un altro problema di responsabilità sorge quando si incorre in un malfunzionamento di un oggetto che, essendo collocato all'interno di infrastrutture critiche come ponti, dighe e autostrade, è addetto al monitoraggio dell'integrità strutturale e delle condizioni ambientali che possono mettere in pericolo la struttura.

2.3 Deep Learning nell'Evoluzione dell'IoT

Dato il numero crescente di dati, digitali e non, e oggetti digitali che ormai circondano ogni attimo della nostra vita prese piede la ricerca sull'analisi di come ricavare beneficio dal collegamento e della sinergia di questi oggetti in rete in favore dell'umanità.
Basandosi infatti nella natura della loro applicazione, questi device comporteranno grandi o veloci flussi di dati in tempo reale.
Applicare analisi su questo flusso dati per scoprire nuove informazioni, predire nuove intuizioni, e effettuare decisioni di controllo è un compito davvero difficile che rende l'IoT un paradigma adatto per il business e una tecnologia per il miglioramento della qualità di vita.
L'Internet delle Cose integrato con il concetto di Deep Learning gioca un ruolo fondamentale nell'affrontare e gestire la considerevole quantità di informazione prodotta e questa nuova simbiosi produrrà negli anni a venire un aumento di risultati di apprendimento automatico e piattaforme nelle quali può essere eseguito.
Ci si aspetta che il lavoro della apprendimento e dell'industria dell'automazione potrebbe avere un impatto economico rientrante in un range fra i 5.2 e i 6.7 trilioni di dollari entro il 2025.
La più ampia sezione dell'Industria IoT, pari a circa il 41% del totale, sarà occupata dall'assistenza sanitaria, seguita da energia e risorse con il 33% e dai trasporti e, da altri campi, per la parte restante.
Queste aspettative implicano, il crescente aumento ripido dei servizi IoT, i dati generati dagli stessi e conseguentemente i relativi ambiti di mercato negli anni a venire.

Fino ad ora sono state sviluppate molte applicazioni IoT utili alla comunità ma un punto fondamentale è che esse abbiano incluso l'apprendimento smart da input di dati e abbiano applicato diverse operazioni per prevedere i risultati in maniera più intelligente.
Le tecniche generalmente includono analisi di dati, elaborazione, apprendimento automatico, riconoscimento di modelli e data mining ma uno degli strumenti principali usato nelle applicazioni IoT è rappresentato dal Deep Learning, tra i maggiori abilitatori alla conoscenza del lavoro automatico.
La ragione per la quale questo strumento di analisi sia cosi popolare tra gli sviluppatori è che le tecniche di Machine Learning tradizionali non sono più così efficaci per supportare le applicazioni dell'Internet delle cose.

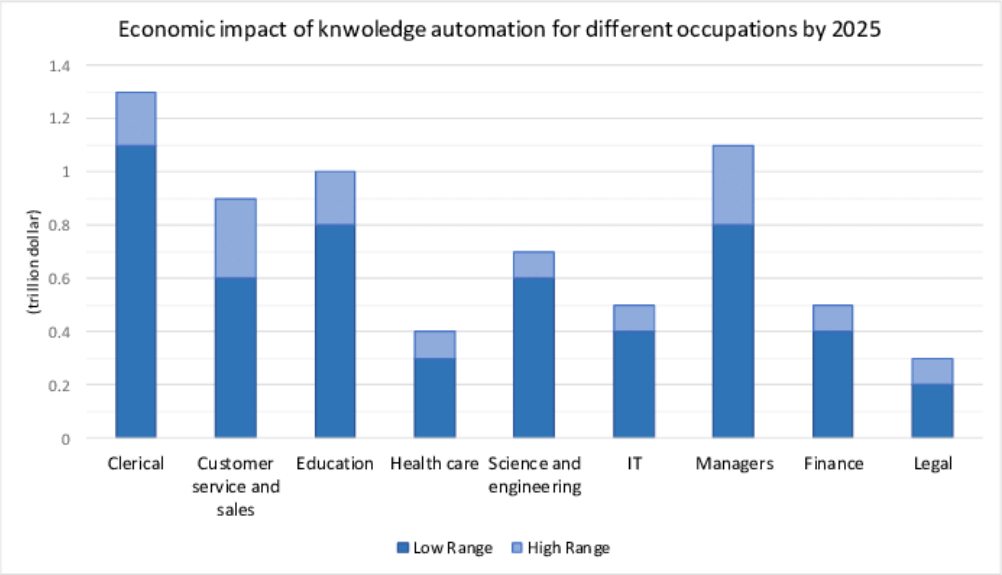


Figure 6: Impatto economico del settore dell'automazione per diverse occupazioni entro l'anno 2025.

I modelli di deep learning riescono infatti a introdurre due miglioramenti importanti nelle due fasi di apprendimento e previsione.
In primis, consentono di ridurre la necessità di set di features artigianali o ingegnerizzate per il training riuscendo ad estrarre facilmente alcune caratteristiche che potrebbero non essere evidenti all'occhio umano e in secondo luogo migliorano l'accuratezza del modello.
Attraverso l'identificazione e l'estrazione di modelli significativi da quantità enormi di dati grezzi in input si può rendere il processo di decisione e, previsione di tendenza, più accurato.
L'estrazione di queste intuizioni e conoscenze dai big data è di enorme importanza per molti business poiché ciò consente loro di ottenere vantaggi nella competizione economica settoriale.
I sistemi IoT necessitano di approcci moderni di analisi dei dati e metodi di intelligenza artificiale secondo la gerarchia della generazione e gestione dei dati IoT come illustrato in figura 7.

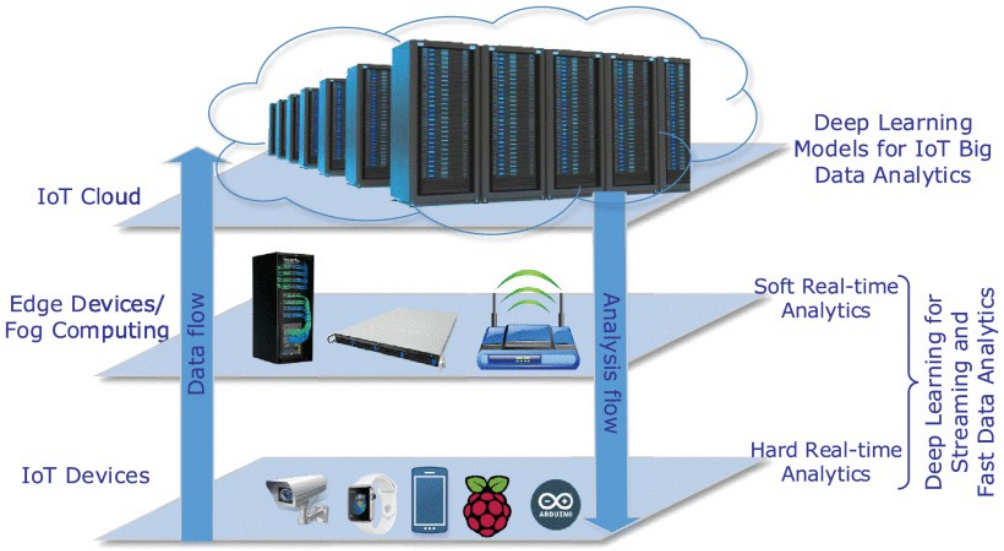


Figure 7: Generazione dei dati IoT ai diversi livelli e modelli di deep learning per affrontare la loro astrazione di conoscenza.

Elemento principale della maggior parte di queste applicazioni è costituito da un meccanismo di apprendimento intelligente (come la regressione, la classificazione e il clustering) per la raccolta dei dati e il riconoscimento di modelli o analisi dei dati in generale.

2.3.1 Caratteristiche dei dati

L'IoT e i Big Data infatti sono in relazione biunivoca, dove l'IoT rappresenta una fonte principale di produzione di big data e dall'altro lato, un obiettivo importante per l'analisi dei big data è quello di migliorare i processi e i servizi dell'IoT.
Tuttavia, esiste una distinzione notevole tra i big data generali e quelli generati dall'IoT, poiché, questi ultimi, possono essere trasmessi continuativamente o accumulati come fonte di grandi quantità di dati.
I Big Data, ovvero grandi datasets che le piattaforme hardware e software comuni non sono totalmente in grado di immagazzinare, gestire, processare e analizzare, per essere analizzati possono richiedere giorni di generazione dati da esser inviati.
Nel nuovo paradigma di oggetti interconnessi, i requisiti per una risposta analitica non sono gli stessi dato che si parla di generazione dati in flusso continuo e in questo caso i risultati delle analisi devono richiedere pochi secondi per essere pronti.
Si parla perciò di una nuova classe di valutazione, ovvero analisi dati veloci e in streaming per

supportare applicazioni con flusso di dati ad alta velocità e richiedenti azioni time-sensitive (real-time o quasi).
Per "Streaming Data" si intende l'estrazione di dati in piccoli intervalli di tempo con un focus ad un output veloce attraverso l'esecuzione di calcoli immediati.
Nel caso dei Big Data gli output delle analisi possono richiedere giorni di generazione dati da essere poi inviati ma lo scenario cambia completamente quando si parla di generazione dati in streaming dove i risultati analitici potrebbero richiedere da millisecondi a pochi secondi per essere pronti.
Applicazioni quali la guida autonoma, previsione di incendi, riconoscimento della postura di un autista anziano (e quindi coscienza e condizioni di salute) piuttosto che il rilevamento di un pedone, richiedono una elaborazione veloce dei dati in arrivo e azioni rapide per il raggiungimento dell' obiettivo.
Molte delle decisioni dell'ambito automotive ad esempio dovrebbero essere supportate da analisi veloci di dati di streaming multimodali provenienti da diverse sorgenti, tra cui i sensori multipli dei veicoli (videocamere, radar, tachimetro, segnali di sinistra/destra, ecc.), comunicazioni tra gli altri veicoli ed entità stradali (semafori, segnali stradali, ecc.).
Queste analisi di dati dovrebbero perciò essere eseguite in prossimità o comunque direttamente alla sorgente dei dati per rimuovere ritardi di comunicazione proibitivi e non necessari.
L'aspetto più importante di questo ambito tecnologico riguarda la necessità di uno strumento che sia in grado di elaborare con efficienza e rapidità l'enorme mole di informazioni aventi le seguenti peculiarità:

- Correlazione spaziale e temporale: nella maggior parte delle applicazioni IoT i sensori sono annessi a una zona specifica e forniscono la posizione e il time-stamp per ogni campione di dato acquisito.

- Trasmissione in larga scala: Nelle applicazioni IoT vengono distribuiti e utilizzati una miriade di device i quali catturano dati generati continuamente. Si viene a costituire quindi un enorme flusso continuo di informazione digitale.

- Eterogeneità: Essendo presenti vari tipi di dispositivi che raccolgono diverse informazioni si viene a creare una struttura eterogenea dei dati disponibili.

- Dati altamente rumorosi: Poiché i dati sono costituiti da piccoli pezzi di informazione

possono essere spesso distorti a causa di errori e rumore nella fase di acquisizione e trasmissione.

É quindi fondamentale l'introduzione di un sistema che faccia fronte a questa varietà di informazione attraverso nuovi algoritmi.
L'incredibile lavoro nel campo del machine learning avanzato e del calcolo veloce che è stato fatto negli anni passati ha giocato un ruolo chiave nell'apprendimento e analisi di questo nuovo grande quantitativo di informazione digitale per l'applicazione nell'Internet of Things.
Questa enorme mole di dati costituisce l'input con il quale si possono analizzare continuativamente i dati dell'IoT estratti in brevi intervalli di tempo in modo più rapido.
Molte ricerche sostengono che l'analisi di flussi di informazioni debbano essere distribuite principalmente in sistemi ad alte prestazioni, piattaforme e cloud server (Google Colab) ma è comunque preferibile eseguirla in piattaforme su scala ridotta (ovvero ai nodi del sistema) o anche negli stessi dispositivi.
I concetti che stanno alla base dell'analisi in questi frameworks sono l'elaborazione progressiva e il parallelismo dei dati.
Per elaborazione progressiva si intende l'elaborazione rapida di una piccola porzione di dati in una serie di attività di calcolo mentre il parallelismo si riferisce a un metodo per mezzo del quale vengono creati tanti piccoli dataset partizionando un insieme di dati più grande nei quali vengono eseguite analisi in parallelo.
Sebbene tuttavia queste tecniche riducano la latenza temporale per restituire una risposta dal framework di analisi dati, esse non sono la migliore soluzione per applicazioni IoT con requisiti di tempo stringenti.
Spostando l'analisi del flusso di dati nelle vicinanze della sorgente dati (cioè nei dispositivi IoT o dispositivi perimetrali quali router, access point o altri) la necessità di parallelismo dei dati e il calcolo incrementale è meno sensata poiché la dimensione dei dati alla fonte permette una più rapida elaborazione.
Un ruolo critico per lo sviluppo di ambienti ubiqui basati sui dati IoT è costituito dalla fusione e condivisione dei dati stessi.
Ove vi è la presenza di applicazioni time-sensitive questa regola è maggiormente critica poiché è di fondamentale importanza una fusione tempestiva dei dati a disposizione per le analisi e conseguenti accurate e affidabili intuizioni utilizzabili.

Normalmente quando si vuole caratterizzare i dati prodotti dall'IoT si usano le caratteristiche delle 6 V:

- **Veracity**: Riguarda consistenza, continuità e correttezza dell'informazione ed è essenziale per l'accuratezza nel riconoscimento del modello e nella stima dei risultati soprattutto quando sono richieste previsioni di quantità.

- **Velocity**: Un elevato tasso di velocità dell'elaborazione e della generazione di dati IoT rappresenta un fattore fondamentale per essere compatibile in tempo reale con i Big Data.

- **Volume**: La differenza tra la massa di dati grezzi tradizionale e i Big Data può essere compresa grazie a questo fattore. I device IoT sono una delle maggiori fonti di produzione di tale enorme quantità di dati.

- **Variety**: I Big Data consistono di varie strutture e forme di dati. I dati prodotti nel contesto IoT possono essere strutturati, semi-strutturati o non strutturati.

- **Variability**: Questo fattore rappresenta la differenza tra il ritmo di flusso dei vari insiemi di dati. Infatti, dispositivi diversi, in un ambiente IoT, possono produrre dati con frequenza diversa a seconda della loro natura e tempo di elaborazione. Inoltre, è possibile che le sorgenti di dati abbiano diversi ritmi di caricamento dati in base a momenti specifici.

- **Value**: Si riferisce alla trasformazione dei Big Data in informazioni utili che permettano agli enti di competere nella concorrenza. Il valore dei dati dipende sia dai processi e servizi sottostanti che nel modo in cui vengono trattati gli stessi.

Oltre alle proprietà di cui sopra, le altre caratteristiche individuate da alcune ricerche sono il fingerprint o il sottoprodotto di una attività digitale o una interazione IoT fonte di Big Data e la scalabilità orizzontale che dovrebbe caratterizzare i sistemi di Big Data, dove le sorgenti di dati dovrebbero venire estese a molteplici dataset.
Nel nuovo mondo di oggetti che apprendono informazioni giorno dopo giorno il Deep Learning fornisce un aiuto fondamentale.
Può essere applicato per aiutare le autorità di trasporto a tenere traccia dei registri di

trasporto e anche per aiutare i veicoli a interagire tra loro diventando più smart.
Risulta quindi fondamentale la gestione intelligente dell'infrastruttura monitorando la quantità di traffico in tutte le aree e creando dei modelli che possano far previsioni ulteriori sullo sviluppo della viabilità.
Possono essere implementati quindi sistemi che utilizzano tecniche di Deep Learning come le Recurrent Neural Network sviluppate in un ambiente di calcolo parallelo.
I sistemi intelligenti di trasporto pubblico e locale incentivano inoltre il rilevamento di segnali stradali per una automazione più precisa, sicura e veloce.
Al giorno d'oggi molti giganti tech sono impegnati nella realizzazione di veicoli a guida autonoma privi di intervento umano i quali saranno equipaggiati con videocamere e vari sensori che raccoglieranno dati digitali.
I metodi di Deep Learning hanno ottenuto incredibili performance nel campo del riconoscimento di immagini, elaborazione di segnali, riconoscimento vocale e rilevamento di guasti.
Ogni rete neurale può funzionare meglio a seconda del dominio di applicazione e alcuni metodi specifici fanno ottenere migliori performance in classi di diversi domini.
Per esempio, gli autoencoder svolgono in modo egregio la loro funzione quando vengono impiegati per il rilevamento di malfunzionamenti mentre le reti neurali convoluzionali danno il loro meglio nelle applicazioni basate sulla grafica.
Alcune applicazioni di questi algoritmi nel campo dell'IoT sono i seguenti:

a. *Smart Homes*: La necessità è quella di usare le applicazioni IoT perché venga raddoppiata la produzione senza compromettere il sistema elettrico. Le previsioni di carico alla rete elettrica sono un'applicazione generale che usa vari metodi di Deep Learning per risolvere il problema in ogni applicazione IoT della casa. La previsione di utilizzo futuro sembra essere più accurata con l'utilizzo dell'architettura LSTM S2S (LSTM Sequence-to-Sequence).

b. *Smart City*: La città smart include vari ambiti che generano grandi quantità di dati i quali possono essere trattati con una prospettiva di machine learning per ottenere risultati di miglior qualità. Il deep learning sta aiutando le autorità dei trasporti per tenere traccia dei registri di trasporto, e i veicoli a diventare più smart nell'interazione fra loro. I modelli di apprendimento possono essere utili per la gestione delle risorse, predire lo spostamento della popolazione o la quantità di traffico presente in alcune aree e mon-

itorare la qualità dell'aria ecc. Uno degli ambiti più importanti dove questi algoritmi di ottimizzazione vengono impiegati è il sistema di gestione dei rifiuti con la classificazione della spazzatura e il monitoraggio della quantità di immondizia prodotta dalla città in un particolar lasso di tempo. Un semplice metodo diretto per risolvere questo problema è l'uso di reti neurali convoluzionali complesse. L'uso di Deep CNN e camere smart è stato impiegato per identificare posti macchina disponibili per parcheggiare i veicoli[14].

Per quanto riguarda gli utilizzi del Deep Learning nell'ambito dei servizi IoT troviamo:

- **Riconoscimento d'immagine**: Molte applicazioni IoT catturano video e immagini in input per poi allenare ed elaborare i dati con metodi di Deep Learning. Questo utilizzo è stato possibile anche grazie alle nuove possibilità offerte dalle moderne videocamere ad alta risoluzione integrate nei dispositivi le quali permettono di catturare immagini e video in qualsiasi posto e momento. Tra i vari dataset utilizzati per ottenere migliori prestazioni troviamo quelli di cifre scritte manualmente(MNIST), dataset di volti(VGG), insiemi di immagini piccole(CIFAR-10 e CIFAR-100) ecc.

- **Riconoscimento Vocale**: Con il progredire della tecnologia mobile il riconoscimento vocale ha assunto un ruolo importante nell'interazione con gli utilizzatori. Ogni smartphone possiede infatti un sistema di riconoscimento vocale che può servire all'utente per un accesso facilitato alle operazioni. In un modello di rete neurale di riconoscimento vocale gli input grezzi sono i dati dei discorsi dell'utente che poi vengono immessi in rete. L'informazione grezza viene elaborata tra gli hidden layer del sistema e il layer di output viene identificato da un suono specifico che assomiglia al discorso dei dati di input.

- **Localizzazione interna**: Le applicazioni IoT che aiutano nella localizzazione di ogni cosa particolare includono sensori per il rilevamento, camere intelligenti per rilevamento dei movimenti, infrarossi o comunicazioni attraverso sensori led luminosi, ecc. Il Deep Learning è una delle tecniche più promettenti per la navigazione interna e il marketing consapevole della localizzazione nei rivenditori. Un sistema conosciuto come DeepFi ad esempio ha consentito di identificare la posizione della persona attraverso le informazioni dello stato della connessione WiFi combinato con il fingerprinting e sfruttando

un metodo di Deep Learning.

- **Privacy e Sicurezza**: Sono tra i maggiori interessi del mondo IoT per ogni tipologia di applicazione. Una categoria generale di intromissione nei sistemi che sono soggetti a gestire flussi di dati è la *False Data Injection (FDI)*. Il rilevamento in tempo reale di attacchi di inserimento di dati corrotti è stato possibile attraverso un meccanismo intelligente di Deep Learning con l'implementazione di una Conditional Deep Belief Network(DBN) in cui sono stati estratti dei dati storici di una smart grid con caratteristiche di attacco FDI. La capacità e la sicurezza del Deep Learning nell'isolare questi comportamenti anomali lo rende essenziale per decidere le probabilità di consenso di tali metodi purtroppo presenti in vari settori dell'IoT.

Sebbene ci siano molti metodi che si integrano con l'ecosistema tecnologico degli oggetti interconnessi, le caratteristiche che lo contraddistinguono quali enormi quantità di dati, varietà e imprevedibilità, sembrano convincere il mondo della ricerca sulla convinzione che il Deep Learning possa dare un contributo determinante per ciò che concerne la sicurezza, la privacy e l'elaborazione in tempo reale di soluzioni che si adattano ad ogni scenario possibile di questa tecnologia emergente.

2.3.2 Servizi principali integrati all'Internet of Things

I modelli di Deep Learning hanno mostrato una utilità promettente nello stato dell'arte di molte aree, come l'elaborazione di segnale, l'elaborazione di linguaggi naturali e elaborazione d'immagine soprattutto nelle applicazioni IoT verticali.
Per esempio le reti convoluzionali garantiscono performance migliori nelle applicazioni visive, mentre gli autoencoder funzionano molto bene con il rilevamento di anomalie, rimozione di rumore, e riduzione di dimensionalità per la visualizzazione di dati.
Nonostante ci siano molti servizi IoT utilizzati nelle applicazioni di smart home o assistenza alle macchine intelligenti, molte delle applicazioni collegate all'IoT utilizzano la classificazione visiva e di immagini (come riconoscimento di segnali stradali o rilevamento di malattie delle piante) come servizio di base intelligente.
La proprietà comune di questi servizi è che dovrebbero essere trattati in maniera analitica veloce piuttosto che accumulare i dati per analisi successive.
La figura 8 mostra i servizi fondamentali e le applicazioni IoT su essi.

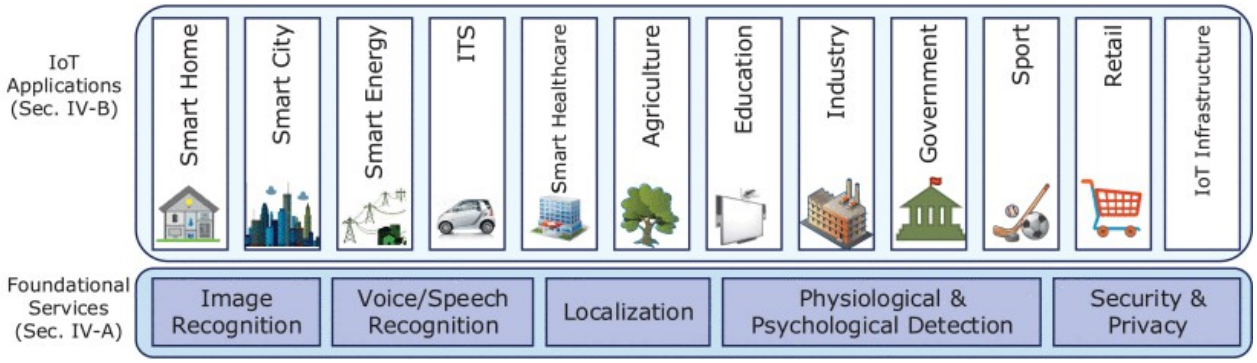


Figure 8: Applicazioni IoT e servizi fondamentali.[34]

Di seguito verranno illustrati i servizi fondamentali nell'IoT che usano il deep learning come strumento di intelligenza artificiale.

1. *Image recognition*

   Una consistente parte delle applicazioni IoT affronta gli scenari in cui i dati di input per la DL sono in forma di video o immagini.

   Riconoscimento o classificazione d'immagini e rilevamento di oggetti sono tra gli usi fondamentali dei mobile device equipaggiati con fotocamere ad alta risoluzione.

   Un problema con i sistemi IoT che trattano il riconoscimento d'immagine è l'uso di dataset sorgente specifici per la valutazione della loro performance poiché, i dataset standard usati (come il MNIST) non rappresentano le caratteristiche specifiche di un ambiente IoT.

   Infatti, potrebbero essere presenti scenari in cui le immagini non siano sempre chiare e queste situazioni non sono gestite attraverso i dataset usuali.

   I modelli allenati basandosi su questi dati non sono abbastanza comprensivi.

2. *Riconoscimento vocale*

   La preoccupazione principale per fornire funzionalità di riconoscimento vocale su device con risorse limitate è nella sua intensità energetica, specialmente quando i dati vengono elaborati su reti neurali.

   In un tipico modello di rete neurale per il riconoscimento vocale, i dati vocali sono i dati grezzi in input alla rete.

   I dati vengono elaborati attraverso gli hidden layer, e la probabilità che i dati vocali si riferiscano ad un particolare discorso viene fatta uscire nel layer di output.

Tra i vari progetti relativi a questo ambito troviamo un chip di Deep Learning a basso consumo per il riconoscimento vocale in cui, per risparmiare energia sono stati progettati tre livelli di riconoscimento di attività vocale con tre reti neuronali separate, ognuna con un livello di complessità diverso.

La rete con la minor complessità, la quale consuma la minor energia, rileva l'attività vocale monitorando il rumore nell'ambiente.

Se questa rete identifica una voce, il chip esegue la rete successiva con livello di complessità aumentato in modo da effettuare una modellazione acustica per identificare se la voce può essere associata ad un discorso.

Qualora ciò si verifichi, la terza rete, con consumo energetico maggiore, viene attivata per identificare le singole parole dell'audio.

3. *Localizzazione interna*

   La fornitura di servizi di localizzazione, come navigazione indoor e advertisement basato sulla posizione di un consumatore, stanno diventando prevalenti in ambienti interni.

   I metodi di localizzazione interna possono essere applicati in altri settori dell'IoT, come nelle case smart, nei campus smart, o negli ospedali.

   I dati di input generati da queste applicazioni di solito provengono da diverse tecnologie (visive, comunicazioni VLC (Visible Light Communications), infrarossi, ultrasuoni, WiFi, RFID, banda ultralarga, e Bluetooth).

   Il deep learning è stato usato con successo per localizzare accuratamente un determinato utente o oggetto.

   Negli esperimenti fatti in questo campo si è notato che il numero di layer e unità interne nei modelli di DL ha un effetto diretto nell'accuratezza della localizzazione.

   Una CNN è stata usata per la localizzazione interna degli utenti fondendo sia i dati magnetici che visivi oppure analizzando un'immagine della loro scena circostante[29].

   Per la localizzazione di un calciatore robot è stato possibile [30], con una rete LSTM, analizzare i dati raccolti da parecchi sensori (Inertia Navigation System e percettori visivi) per predire la sua posizione.

4. *Rilevamento dello stato fisiologico e psicologico*

   L'IoT combinato con le tecniche di deep learning è stato utilizzato per rilevare vari stati fisiologici e psicologici umani, come la posa, l'attività e le emozioni.

Molte applicazioni IoT, per fornire i loro servizi come case smart, automobili smart, intrattenimento, educazione, riabilitazione e assistenza sanitaria, sport e produzioni industriali, incorporano un modulo per la stima della posizione umana o per il riconoscimento di attività.

Questa tipologia di servizi può anche essere usata nell'istruzione per monitorare l'attenzione degli studenti, e nei negozi di vendita al dettaglio per prevedere il comportamento di acquisto dei clienti.

É stata proposta ad esempio una struttura di deep learning che unisce la forza di reti neurali CNN e LSTM per l'identificazione di attività umana da dati di un sensore indossabile (accelerometro, giroscopio ecc.).

L'architettura LSTM può essere usata ad esempio per il riconoscimento dell'attività umana basata sui dati dei sensori di un dispositivo mobile.

L'utilizzo di una deep CNN ha interessato le attività mediche in una sala traumi (per esempio misura della pressione sanguigna, esame della bocca, posizionamento dell'elettrocatetere cardiaco, ecc.) analizzando i dati grezzi da dei tag RFID[25].

Il riconoscimento di gesti dai frame di un video può essere realizzato combinando una CNN e una RNN [37] mentre una rete DNN chiamata Encoder-Recurrent-Decoder è stata usata per il riconoscimento della posa del corpo umano e per la previsione di movimento in video e in dataset di cattura del movimento [13].

I modelli di movimento umani possono inoltre essere usati come una mezzo per l'identificazione e l'autenticazione di un utente.

5. *Sicurezza e Privacy*

   Sicurezza e privacy sono l'interesse maggiore in tutti gli ambiti e le applicazioni IoT.

   La validità delle funzionalità dei sistemi dipende dalla protezione degli strumenti di machine learning e i processi dagli attacchi.

   Nelle reti IoT, la sicurezza informatica relativa al rilevamento di intrusioni e l'identificazione di malware si basa spesso su approcci automatici derivanti dal deep learning.

   Ad esempio, nell'articolo [18] le caratteristiche di un attacco FDI sono estratte dai dati storici della smart grid e queste caratteristiche sono utilizzate per una rilevazione di attacco in tempo reale attraverso una Conditional DBN.

   Gli smartphone inoltre, essendo grandi contribuenti per dati e applicazioni IoT sono in

serio pericolo da attacchi hacker.

Perciò, proteggere questi device da una varietà di problemi di sicurezza è necessario per una prospettiva IoT che vada oltre la preoccupazione dell'utilizzatore.

Il mantenimento della sicurezza e della privacy degli approcci di deep learning sono tra i fattori più importanti che permettono la fiducia nell'uso di questi metodi nel settore IoT.

Al fine di affrontare i problemi di mantenimento della privacy in modelli di DL, quando soggetti a apprendimento distribuito, è stato proposto un metodo da Shokri *et all.* [43] per preservare sia la privacy dei dati di training dei partecipanti, che l'accuratezza del modello.

Nonostante il deep learning sia efficace in semplici problemi di classificazione, può risultare facile bypassare un filtro di analisi IoT basato su architetture di deep learning attraverso task dannosi di tipo adversarial.

Il deep learning adversarial infatti è cruciale nei sistemi IoT dove può essere compromessa la vita di un individuo, come ad esempio il riconoscimento del bordo stradale usato nei veicoli intelligenti.

Piccole manipolazioni alla segnaletica stradale possono infatti portare a risultati opposti distinti nei metodi di deep learning.

Gli avversari possono lanciare attacchi di trasferibilità ricostruendo un modello di deep learning indipendente al fine di simulare altri modelli usando solo dati di input e label di output, senza conoscere i parametri del modello di deep learning originale e la tipologia del modello stesso[27].

2.3.3 Adattamento degli algoritmi deep learning alle caratteristiche dei dispositivi

Fino d oggi, gli algoritmi di deep learning sono stati scarsamente implementati su dispositivi IoT a causa dei limiti sulle risorse di calcolo: i modelli richiedono una certa capacità di calcolo, energia e memoria.

In alcuni casi questi oggetti potrebbero non disporre di capacità necessarie a eseguire un algoritmo di deep learning pre allenato per attività di previsione o classificazione.

I modelli deep learning infatti possono essere caratterizzati da milioni o anche miliardi di

parametri, oltre a richiedere calcoli sofisticati e grandi risorse di memoria.
Una soluzione possibile potrebbe essere rappresentata dalla rimozione di parametri e/o layer per ridurre considerevolmente la complessità delle reti mantenendo comunque un output accettabile e rendendole adatte all'IoT.
Di seguito vengono illustrate le principali tecniche per cercare di adattare le strutture di DL al paradigma IoT in un contesto di dispositivi a risorse ridotte.

- *Compressione di rete*: Un modo per utilizzare le DNN in dispositivi con poche risorse è attraverso l'uso della compressione di rete, nella quale una rete densa viene convertita in una rete sparsa. Questo approccio aiuta nella riduzione dei requisiti di elaborazione e di memoria delle DNN. Il limite principale di questa strategia è che non è abbastanza generale da essere applicata ad ogni tipo di rete. Si è notato che, in termini di tempo di esecuzione per reti CNN, l'ultimo layer convoluzionale tende a impiegare meno tempo di elaborazione al decrescere delle loro dimensioni. Una via per migliorare i modelli CNN sui dispositivi a risorse ridotte può essere quello di sostituire i layer convoluzionali con feed-forward layers ove possibile, che danno risultati più veloci. In aggiunta, scegliendo le funzioni di attivazione usate nelle DNN si può ottenere un grande effetto nell'efficienza temporale. Per quanto concerne la memoria le CNN ne usano una quantità inferiore rispetto alle DNN in quanto il numero di parametri è inferiore. Un'altra soluzione nelle DNN è scartare i parametri meno importanti e ridondanti. Per ottenere ciò si possono eseguire i seguenti passaggi:
  - Allenare la rete per trovare le connessioni con weight elevati.
  - Sfoltire le connessioni irrilevanti che hanno un weight minore di una certa soglia.
  - Dopo lo sfoltimento, potrebbero rimanere alcuni neuroni senza connessioni di input o di output. Il processo di sfoltimento quindi identifica questi neuroni e li rimuove così come tutte le connessioni rimanenti.
  - Riallenare la rete per mettere a punto i pesi del modello aggiornato. I pesi dovrebbero essere trasferiti tra i precedenti step di allenamento invece che essere inizializzati, altrimenti le performance degraderanno in qualche misura.
- *Calcolo Approssimativo*: Il calcolo approssimato è un altro approccio sia per implementare gli strumenti di machine learning sui dispositivi IoT, che per risparmiare ener-

gia nei dispositivi in cui sono usati. La validità di questa soluzione deriva dal fatto che, in molte applicazioni IoT, gli output del machine learning (cioè le previsioni) non devono essere esatte, ma piuttosto ricadere in un range accettabile che fornisca la qualità desiderata. Infatti, questi approcci devono definire dei limiti di qualità che l'output non può oltrepassare. Integrando i modelli di deep learning con il calcolo approssimativo si può ottenere un modello di apprendimento maggiormente efficiente per i dispositivi a risorse limitate. Alcune soluzioni proposte cercano di aggiustare la precisione degli input e dei pesi dei neuroni per arrivare a un compromesso ottimale tra precisione e consumo energetico. Sono stati fatti tentativi che hanno applicato il calcolo approssimativo con il ridimensionamento di precisione su reti CNN e DBN. Tuttavia la pratica corrente richiede che il processo di allenamento del modello e la sua conversione ad un'approssimazione di una struttura deep learning avvenga in una piattaforma con molte risorse e che poi il modello convertito venga sfruttato in un dispositivo a risorse limitate.

- *Acceleratori*: La progettazione di hardware e circuiti specifici è un'altra direzione di ricerca attiva che mira a ottimizzare l'efficienza energetica e l'impatto sulla memoria dei modelli di deep learning nei dispositivi IoT. Tra le soluzioni adottate dobbiamo ricordare gli acceleratori hardware per reti DNN e tecnologie Post-CMOS come la spintronica. Una soluzione complementare per la progettazione di acceleratori hardware proposta è costituita da un acceleratore software per la fase di deduzione di modelli di deep learning su dispositivi mobili in cui vengono utilizzati in runtime algoritmi di controllo delle risorse, uno che comprime i layer, e l'altro che decompone i modelli di una architettura DL tra i vari processori disponibili.

- *Tinymotes*: In aggiunta alle soluzioni precedenti, sta crescendo lo sviluppo di processori con piccole dimensioni (micromotes). Progettati nel range di un millimetro cubo, questi processori possono essere avviati con batterie, consumando circa 300 microwatt mentre eseguono le analisi interne e le previsioni usando acceletatori di reti di deep learning. Attraverso questa tecnologia, molte applicazioni IoT che richiedono tempistiche stringenti possono eseguire un processo decisionale nel dispositivo al posto di inviare i dati in elaboratori ad alte prestazioni e espettando la loro risposta. Per applicazioni in cui la maggior preoccupazione riguarda la sicurezza e la privacy, l'integrazione di hardware e

DL riducono queste preoccupazioni in un certo senso, poiché non devono essere inviati dati, o, parti limitate degli stessi, nel cloud per le analisi.

Le caratteristiche particolari dei dispositivi IoT e le tecniche di DL rendono il supporto di dispositivi con risorse ridotte più difficile dato che questi ultimi raramente possono ospitare modelli di DL anche solo per fare previsioni.
I metodi presentati precedentemente servono quindi ad orientare il processo di training ai dispositivi IoT per avere un utilizzo scalabile e distribuito dei dispositivi.
La compressione di rete, pur essendo un approccio promettente per ottenere analisi quasi in tempo reale sui dispositivi IoT, devono essere fatte maggiori riflessioni per trattare alcune sfide:

- Non è chiaro se gli approcci di compressione della rete siano adatti per lo streaming di dati, specialmente quando il modello di DL è dinamico e può evolvere nel tempo.
- I metodi di compressione per le architetture di serie temporali come le RNN e le LSTM, non sono stati approfonditi e c'è una lacuna sul fatto che le strategie di compressione esistenti possano essere applicate a queste architetture di deep learning.
- C'è la necessità di specificare il compromesso tra il tasso di compressione e l'accuratezza di una DNN, poiché una maggior compressione porta ad un degrado dell'accuratezza.

Altri metodi recentemente utilizzati provano a sfruttare i neuroni poco significativi e piuttosto di manipolare la struttura della rete, cambiano la rappresentazione dei calcoli attraverso una riduzione della quantità di bit.
Sembrerebbero essere applicabili ad una varietà vasta di architetture di DL e possono anche gestire l'evoluzione dinamica dei modelli della rete nel tempo.

2.3.4 La scalabilità al Fog, Edge e Cloud Computing

Le due architetture principali in cui si suddividono gli approcci per l'IoT sono: sistemi di fog-computing in loco e Cloud-centrici.

1. *Sistemi IoT Cloud-centrici*: In un sistema di questo tipo i dati grezzi acquisiti dai sensori (per esempio videocamere) in nodi IoT, vengono mandati ad una rete neurale presente nel Cloud. Dopo che l'attività di previsione (per esempio classificazione e

riconoscimento) è stata eseguita, il risultato viene restituito al nodo IoT. Tuttavia, ci sono parecchi svantaggi per questo sistema basato sul Cloud ovvero:

- Una grande quantità di dati inviata al cloud attraverso la rete, che causa troppa latenza e dispendio energetico.
- La necessità di avere una connessione stabile al cloud sempre, il che è complicato in alcune situazioni.
- Inviare i dati al Cloud può far sorgere questioni di sicurezza e privacy.

2. *Fog computing in loco*: Per affrontare le sfide dei sistemi IoT centrati sul Cloud, nei sistemi IoT basati sul fog computing, il modello di deep learning è allenato nel Cloud usando un dataset statico e potenzialmente molto grande. Poi il modello viene impiegato nell' IoT per eseguire l'attività di previsione.

Il calcolo in cloud è considerata una soluzione per l'analisi di Big Data provenienti dall'IoT ma potrebbe non essere ideale per i dati IoT con restrizioni di sicurezza e/o legali, dovute a politiche di gestione (per esempio i dati non dovrebbero essere inviati a centri cloud al di fuori della nazione), o vincoli temporali.
D'altro canto, l'alto livello di astrazione dei dati per qualche scopo di analisi dovrebbe essere ottenuto aggregando parecchie sorgenti di dati IoT; quindi, in questi casi, è insufficiente sviluppare soluzioni analitiche su nodi IoT individuali.
La grande problematica quando si trattano ambienti IoT è quella di estrarre informazione e modelli importanti basandosi sui dati grezzi catturati da ambienti complessi e rumorosi.
Il cloud però, possedendo un enorme potere di calcolo e scalabilità, consente di essere l'opzione migliore per allenare e valutare le reti neurali.
Piuttosto che muovere costantemente un grande numero di dati grezzi verso il cloud, potrebbe essere utile sfruttare i prestanti dispositivi IoT emergenti per eseguire attività di pre-classificazione o allenamento.
Ciò nonostante, allenare staticamente i modelli potrebbe portare ad una ridotta accuratezza.
Per i sistemi IoT con al centro il Cloud, i dati generati da un dispositivo intelligente necessitano di essere inviati al cloud per l'elaborazione, e poi i risultati sono rispediti indietro al dispositivo.
Tuttavia, la grande trasmissione dati attraverso la rete presenta una grande sfida e può soll-

evare anche a problemi di privacy.
Per far fronte a queste problematiche è stata introdotta l'intelligenza artificiale in loco, ovvero il primo framework di calcolo incrementale autonomo e un'architettura per applicazioni IoT basate sul deep learning.
Nel nuovo approccio proposto per affrontare queste situazioni, ovvero il fog-computing, il calcolo non viene più quindi centralizzato nel cloud ma le elaborazioni e le analisi vengono eseguite nei pressi del dispositivo dell'utente finale.
Si riescono a sfruttare quindi sia i vantaggi del calcolo in cloud riducendo/evitando allo stesso modo lo svantaggio relativo alla latenza e ai rischi di sicurezza.
Sebbene le tecniche di fog-computing e di calcolo in loco eliminino l'effetto di fluttuazione della qualità della rete e proteggano la privacy utente, introducono un nuovo problema: bassa accuratezza nei risultati di previsione.
Inoltre, non bisogna trascurare il fatto che i dati della maggior parte dei sistemi IoT sono dinamici e unlabeled perciò i modelli allenati staticamente potrebbero non essere efficienti nel gestire dati dinamici in loco.
Per far fronte a questo inconveniente sono stati sviluppati sistemi locali che riescano a soddisfare ambienti in continuo cambiamento ottenendo una accuratezza ideale.
Tra le maggiori sfide rivolte a questo tipo di strutture di analisi che integrano il deep learning con il fog-computing si denotano:

- DL service discovery: i nodi fog sono distribuiti densamente in regioni geografiche, ed ogni nodo può avere specifiche capacità di analisi.
  I dispositivi dovrebbero identificare le sorgenti di appropriati fornitori di analisi attraverso delle sorti di protocolli estesi per trovare i servizi per le analisi DL.

- Distribuzione di mansioni e del modello di DL: partizionare l'esecuzione di un modello di DL e i task tra i nodi fog e distribuire in maniera ottimale lo stream di dati tra i nodi disponibili sono delle criticità per alcune applicazioni time-sensitive.
  Aggregare i risultati finali dai nodi di calcolo con bassa latenza rappresenta l'altro lato della medaglia.

- Fattori di progettazione: è buona cosa valutare quali siano i fattori di costruzione dell'ambiente fog e lo sviluppo di modelli di DL può impattare molto sulla qualità dei

servizi di analisi.

- Mobile edge: attraverso l'ubiquità degli ambienti di calcolo mobile periferico e il loro contributo alle analisi IoT, è importante considerare la dinamicità di questi ambienti per sviluppare analisi DL assistite ai margini, poiché i dispositivi mobili possono entrare o uscire dal sistema.

2.3.5 Sfide e sviluppi futuri

La mancanza di disponibilità di grandi dataset reali per applicazioni IoT è un ostacolo principale per incorporare modelli di DL poiché sorge la necessità di maggiori quantità di dati per il deep learning per ottenere una accuratezza migliore.
Una seconda sfida è calcolare una rappresentazione adatta al dispositivo ma efficace per lo specifico task.
La garanzia di sicurezza e privacy nelle applicazioni costituisce un'altra importante tematica. I modelli di training di DL sono soggetti a attacchi malevoli come inserimento di dati falsi o campioni corrotti, con i quali alcuni requisiti funzionali o non funzionali dei sistemi IoT possono essere in pericolo.
Le performance temporali e la complessità di un modello di DL possono essere influenzate dal voluminoso numero di dati in input, il numero di attributi, dal loro alto grado di classificazione e dal livello di rumore presente.
Uno degli obiettivi principali per gli ambienti IoT sarà quello di raggiungere analisi ed elaborazione di dati veloci vista la velocità di generazione presente in questo scenario tecnologico.
Oltre a tutte queste problematiche si deve far fronte alle limitazioni di calcolo e di funzionalità dei dispositivi in quanto i modelli di DL si focalizzano principalmente sulla classificazione, mancando funzionalità di regressione.
Per quanto riguarda le direzioni future della ricerca che mirano a risolvere tutte queste problematiche troviamo:

- Trovare vie efficienti per utilizzare i big data mobile in simbiosi con gli approcci di DL al fine di migliorare i servizi per gli ambiti IoT, specialmente in scenari di città smart.

- Fusione con altre sorgenti di dati, ovvero informazioni del contesto che completino la comprensione dell'ambiente.

- Data la natura del flusso di dati IoT, non è più possibile utilizzare un volume definito di dati di allenamento per i modelli di deep learning, soprattutto nel cloud o fog computing ove sono presenti algoritmi di apprendimento con elaborazione veloce dei dati. E' quindi necessaria una nuova classe di algoritmi che lavori basandosi sul flusso di dati ottenuto in tempo reale.

- La maggior parte degli algoritmi sono supervisionati e necessitano quindi di una grande quantità di dati di training etichettati i quali non sempre possono essere disponibili o devono essere prodotti con fatica. Può essere impiegata quindi una combinazione di algoritmi di machine learning avanzati progettati per scenari semi-supervisionati, dove un piccolo dataset viene usato per il training mentre l'agente di apprendimento migliora la sua accuratezza usando una grande mole di dati unlabeled.

- Visto l'affidamento a sistemi informatici e IoT in larga scala, c'è il bisogno di sistemi di sicurezza contro attacchi malevoli e malfunzionamenti.

- Sebbene l'ampio comparto di nodi di rete e la loro relazione sia una sfida per gli approcci di machine learning tradizionale, apre le opportunità alle architetture di DL per testare la loro competenza in questa area fornendo un range di servizi autonomi quali auto-configurazione, auto-ottimizzazione, auto-riparazione e auto-bilanciamento del carico.

L'integrazione del Deep Learning al nuovo paradigma di oggetti interconnessi e le tecniche innovative verso cui la ricerca si sta spostando sembrano promettere quindi una prospettiva fruttuosa per tutti gli ambiti sociali, economici e industriali toccati dall'IoT.

2.4 Tipologie di Architettura DL

Il Deep Learnign consiste in tecniche, di apprendimento supervisionato e non supervisionato, che si basano su diversi livelli di Reti Neuronali Artificiali (ANNs) le quali sono in grado di apprendere rappresentazioni gerarchiche in architetture profonde.
Le architetture di deep learning consistono di layer multipli di elaborazione dove ogni layer è capace di produrre risposte non lineari basate sui dati del layer di input.
Un'attenzione maggiore verso questo tipo di architettura è dovuta all'ormai superficiale struttura degli approcci di machine learning ora considerati obsoleti.

Come si vede dall'immagine di figura 9, gli algoritmi di Deep Learning sono molto più popolari di altri algoritmi di Machine Learning secondo i trend di Google. Le tecniche di deep

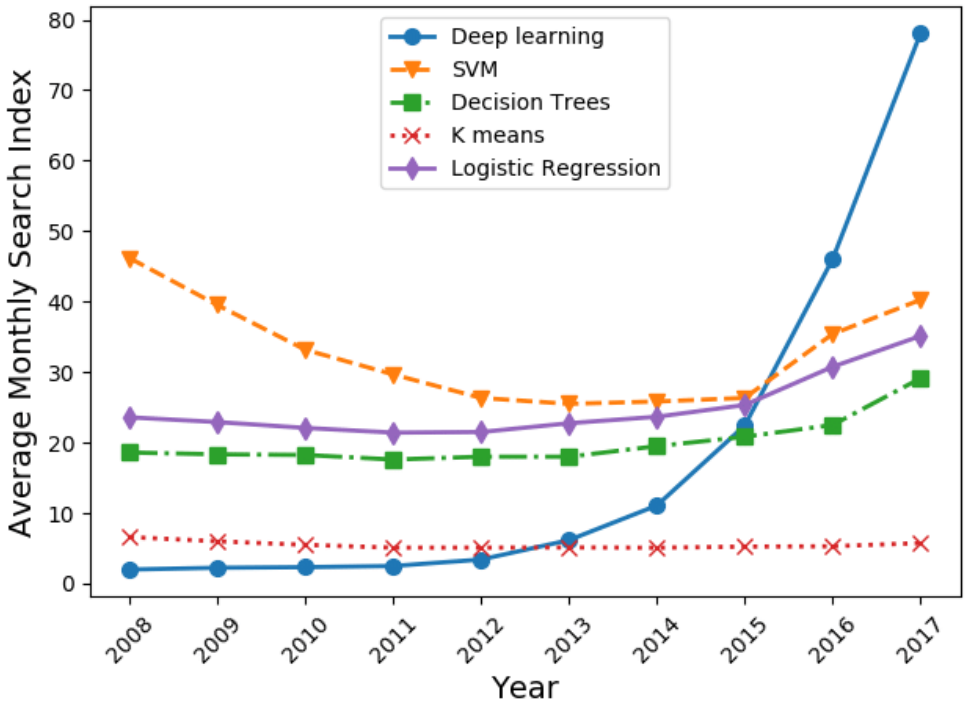


Figure 9: Ricerca Google sulla tendenza dei vari algoritmi di apprendimento.

learning sono state sviluppate sulle tradizionali Reti Neuronali Artificiali.

Reti neuronali Feed-forward(FNN), utilizzate negli anni scorsi per allenare vari sistemi, hanno dimostrato difficoltà nell'allenamento all'aumentare del numero di layer.

Le dimensioni ridotte dei dati di training sono un altro fattore che ha portato i modelli in overfitting.

Oltre a questi fattori, ciò che ha proibito l'implementazione di profonde reti FNN efficienti è stato causato dai limiti di capacità di calcolo che tuttavia sono stati poi risolti con gli avanzamenti hardware recenti e lo sviluppo delle unità si elaborazione grafica (GPU) con le accelerazioni hardware relative.

Gli avanzamenti principali che hanno coinvolto le tecniche di machine learning includono:

- Uso di Rectified Linear Units (RELUs) come funzione di attivazione
- Introduzione di metodi di dropout
- Inizializzazione random dei pesi della rete

- Affrontare la degradazione dell'accuratezza di training con reti di apprendimento residue.
- Risoluzione del problema della scomparsa del gradiente cosi come il problema dell'esplosione del gradiente attraverso l'introduzione e il rafforzamento delle reti Short-Term Memory.

Uno dei vantaggi maggiori di questo tipo di nuovo approccio di apprendimento è quello della gerarchia delle feature ovvero il poter riconoscere caratteristiche più complesse grazie ai layer più interni del modello, poiché le architetture di deep learning aggregano e ricombinano i tratti distintivi dall'output dei layer precedenti.
Tuttavia, i miglioramenti dei modelli di deep learning sono basati su valutazioni empiriche e non c'è fondamento analitico che dimostri un funzionamento più efficiente di tecniche deep learning sulle controparti superficiali.
Non c'è un confine chiaro che possa far distinguere le reti neurali profonde da quelle superficiali basandosi sul numero di layer interni.
Generalmente le reti neurali con due o più hidden layers che incorporano i più recenti algoritmi di training avanzato sono considerati come modelli profondi.
Le principali categorie in cui possono essere suddivise le strutture dei modelli di deep learning sono tre: modelli generativi, discriminativi e ibridi.
I primi usati particolarmente per l'apprendimento non supervisionato, i successivi forniscono approcci per l'apprendimento supervisionato mentre gli ultimi garantiscono sia i vantaggi dei modelli discriminativi che di quelli generativi.
Si elencano una serie di architetture di reti neurali profonde utilizzate nel mondo IoT:

- *Convolutional Neural Networks (CNNs)*: Qualora una rete neurale debba elaborare immagini o altri dati visivi, i modelli di rete neurali più efficaci richiedo delle connessioni dense fra i vari layer che rendono le reti difficili da allenare e non poco scalabili a causa principalmente della proprietà di invarianza di traslazione di questi modelli. Le CNN permettono invece di apprendere le caratteristiche dell'immagine che potrebbero apparire nel dato in input geometricamente trasformate (per esempio la rotazione della mano nel rilevamento della posa). La loro struttura è formata da un input 2-D, da layer convoluzionali interni e da layer totalmente connessi alla fine. I layer convoluzionali costituiscono il cuore della CNN e consistono di un set di parametri allenabili chiamati filtri i quali hanno la stessa forma dell'input ma dimensioni minori. Un altro blocco

dell'architettura CNN è formato dai pooling layer che operano nella mappa delle feature e permettono di ridurre la dimensione spaziale della rappresentazione al fine di ridurre il numero di parametri, i tempi di calcolo e la possibilità di overfitting. Una funzione di attivazione importante in questa rete è la ReLU, che consiste in un neurone in cui è presente una funzione nella forma $f(x) = \max(0, x)$. Questa funzione di attivazione permette un minor tempo di training senza compromettere in modo negativo la generalizzazione della rete. Vari tipi di scenari di applicazione includono la previsione di inondazioni o frane, rilevamento di malattie delle piante usando le immagini negli smartphone, rilevamento di segnali stradali usando le videocamere dei veicoli.

- *Recurrent Neural Network (RNN)*: Vengono utilizzate per trattare attività dove la previsione è dipendente da parecchi campioni precedenti e per classificare un campione singolo sono necessarie le sequenze di input. Le RNN hanno risolto questa situazione in problemi sequenziali (riconoscimento vocale) o di serie temporali (sensori) di varia lunghezza. Alcuni esempi di applicazioni possono essere, il rilevamento del comportamento di veicoli smart, l'identificazione di modelli di movimento individuale e la stima del consumo energetico di una famiglia. L'input di una RNN consiste di sia i campioni correnti e che in campioni osservati precedentemente ovvero l'output di una RNN al tempo $t-1$ influenza l'output al tempo t. Per allenare la rete viene usato l'algoritmo Backpropagation Through Time (BPTT), un'estensione dell'algoritmo backpropagation. La parte principale dell'algoritmo BPTT è una tecnica chiamata srotolamento della RNN, che permette di ottenere una feed-forward network su intervalli di tempo. Gli hidden layer della rete si suppone forniscano una memorizzazione piuttosto che una rappresentazione gerarchica delle caratteristiche.

- *Long Short Term Memory (LSTM)*: La LSTM è una estensione della RNN. In aggiunta al feedback loop che immagazzina l'informazione, ogni neurone nella LSTM (anche chiamato cella di memoria) ha un gate moltiplicativo di dimenticanza, lettura e di scrittura. Questi gate sono stati introdotti per controllare l'accesso alle celle di memoria e per proteggerle dalle perturbazioni dovute a input irrilevanti. Un'importante differenza con le RNN è che le unità di una LSTM utilizzano i gate di dimenticanza, per controllare attivamente lo stato delle celle e garantire che non degradino. Quando i dati sono caratterizzati da una lunga dipendenza nel tempo, i modelli LSTM performano meglio

di quelli RNN. Il lungo ritardo di dipendenza può essere osservato nelle applicazioni IoT, come il riconoscimento di attività umana, la previsione di rendimento scolastico nei programmi online, e previsione di disastri basata sul monitoraggio ambientale.

- *Autoencoders (AEs)*: Gli autoencoder consistono in una rete neurale di più hidden layer con lo stesso numero di unità in input e output. Questa rete cerca di ricostruire l'input trasformando gli input negli output nella maniera più semplice possibile, cercando di non distorcere il campione in ingresso. Questo tipo di reti neurali è stato usato principalmente per risolvere problemi di unsupervised learning così come di transfer learning. Sono impiegate principalmente per diagnosi e attività di rilevamento guasti vista la loro capacità di ricostruire l'input in uscita al layer di output. É grande l'interesse questa architettura per le industrie IoT che vogliono offrire applicazioni come diagnosi di guasto in dispositivi e macchine hardware, e rilevamento di anomalie nelle performance delle linee di assemblaggio.

- *Variational Autoencoders (VAEs)*: Sono un modello generativo le cui assunzioni nella struttura dei dati non è forte, avendo comunque un processo di allenamento veloce attraverso il backpropagation. Questa struttura di rete neurale è stata usata per il semi-supervised learning. Quindi è adatta a soluzioni IoT che abbiano a che vedere con dati diversi e scarsità di dati senza classificazione. In questo ambito troviamo applicazioni di individuazione guasti nei livelli di rilevamento e individuazione di intrusi in sistemi di sicurezza.

- *Generative Adversarial Network (GANs)*: Le GAN sono formate da due reti neurali, ossia le reti generative e discriminative, che lavorano insieme per ottenere dati sintetici e di alta qualità. La rete generativa è responsabile di generare nuovi dati dopo aver appreso la distribuzione di dati dal dataset di allenamento mentre la seconda (ossia la discriminativa) esegue una discriminazione tra i dati reali che provengono dai dati di training dai dati di input falsi che provengono dal generatore. La rete generativa è ottimizzata per produrre dati di input che siano in grado di ingannare il discriminatore così che sia difficile distinguere se i dati siano veri o falsi. Si crea in sostanza una competizione tra la rete generativa e la rete discriminativa. La funzione obiettivo nelle GAN è focalizzata in strategie di minimax in modo tale che una rete cerchi di massimizzare

la funzione valore e l'altra rete punti a minimizzarla. Nelle applicazioni IoT, le GAN possono essere applicate in scenari dove sia richiesta la creazione di qualcosa di nuovo dai dati disponibili. Ciò può includere applicazioni di localizzazione e way-finding, dove la rete generativa nella GAN produce potenziali cammini tra due punti, mentre il discriminatore identifica quale dei cammini è percorribile. Sono utili anche per sviluppare servizi per persone con problemi di vista, come convertitori di immagini in contenuti uditivi usando sia delle GAN per generare testo descrittivo da una data immagine che un altro modello di deep learning per eseguire la conversione del testo in audio.

- *Restricted Boltzmann Machine(RBM)*: Una RBM è una rete neuronale artificiale costituita da due layer: un layer visibile che contiene l'input che conosciamo e un layer nascosto che contiene le variabili latenti. L'RBM dovrebbe costruire un grafo bipartito, tale che ogni neurone invisibile dovrebbe essere connesso a tutti i neuroni dell' hidden layer e viceversa, ma tale che non ci sia connessione tra ogni due unità nello stesso layer. Inoltre, l'unità di bias è connessa a tutti i neuroni visibili e nascosti. L'obiettivo dell'allenamento di una RBM è quello di massimizzare il prodotto tra tutte le probabilità delle unità visibili. Questa struttura è in grado di estrarre le caratteristiche dai dati di input attraverso la modellizzazione della distribuzione di probabilità su un set di input il quale è rappresentato in un insieme di unità nascoste. Alcuni tipi di applicazioni dove possono essere usate le RBM sono la localizzazione interna, previsione di consumo di energia, previsione di congestione del traffico, analisi di postura, e generalmente ogni applicazione che abbia beneficio dall'estrazione delle caratteristiche più importanti tra quelle disponibili.

- *Deep Belief Network (DBNs)*: Le Deep Belief Network sono un tipo di rete ANN generative che consistono di un layer visibile in input e parecchi hidden layer corrispondenti alle variabili latenti. Riescono ad estrarre la rappresentazione gerarchica dei dati di training cosi come ricostruire i dati di input. Possono essere usate per attività di previsione attraverso l'aggiunta di un classification layer come softmax. Il meccanismo che rende la DBN un algoritmo di deep learning veloce riguarda la fase di training che viene effettuata layer by layer in modo tale che ogni layer sia trattato come una RBM allenata sopra il layer precedentemente allenato. Parecchie applicazioni possono trarre beneficio da questa architettura come la classificazione di rilevamenti di guasto

in ambienti industriali, identificazione di minacce in sistemi di allarmi di sicurezza e estrazione di caratteristiche emotive da immagini.

- *Ladder Networks*: Proposte nel 2015 per la prima volta, hanno mostrato di funzionare bene in molte attività, dal riconoscimento della scrittura alla classificazione di immagini. L'architettura è costituita da due codificatori e un decodificatore. I codificatori agiscono sulla parte di apprendimento supervisionato della rete e il decodificatore esegue l'apprendimento non supervisionato. Uno dei codificatori, chiamato *clean encoder*, produce l'elaborazione normale mentre l'altro codificatore, chiamato *corrupted encoder*, aggiunge un rumore Gaussiano a tutti i layer. Il decoder, usando una funzione di denoising, può ricostruire la rappresentazione ad ogni layer avendo a disposizione il corrispondente dato corrotto. L'obiettivo di training è quello di minimizzare la somma del costo della parte supervisionata e di quella non supervisionata della rete. Sebbene non sia stata usata ampiamente nello scenario IoT, le ladder network hanno il potenziale di essere usate in molte analisi IoT basate sulla visione dove la semi-supervisione costituisce un gran incentivo.

I lavori di ricerca su analisi veloci e in tempo reale attraverso l'uso di modelli di Deep Learning per stream di dati sono ancora agli inizi.
Nonostante gli avanzamenti strutturali portati, queste architetture sono state inoltre usate insieme ad altre tecniche di machine learning per raggiungere una maggiore efficienza.
Una delle motivazioni principali che hanno portato a questa scelta sono state la possibilità da parte di altri approcci di machine learning di usare la funzione non lineare di approssimazione dei modelli di deep learning che può supportare centinaia o anche miliardi di parametri.
L'altra spinta per applicare questi modelli con altri approcci è arrivata dalla capacità delle strutture di deep learning di estrarre automaticamente le caratteristiche.
Di seguito viene presentato un riepilogo di questi approcci adatti allo scenario IoT:

1. *Deep Reinforcement Learning*: Si tratta di una combinazione del reinforcement learning (RL) con le DNN che cerca di creare agenti software che possano apprendere da soli per stabilire politiche di successo per ottenere massime ricompense a lungo termine. La necessità di una DNN in un modello di reinforcement learning diventa evidente quando l'ambiente può essere caratterizzato da un numero elevato di stati. In questa

situazione la rete RL non è abbastanza efficiente e il modello DL può essere usato per approssimare i valori di azione al fine di stimare la qualità di un'azione in un certo stato. Nel campo dell'Internet of Things si è utilizzato questo tipo di rete in un ambiente di apprendimento semi-supervised per la localizzazione in ambienti di campus smart per localizzare gli utenti in base ai segnali ricevuti da molteplici dispositivi iBeacon Bluetooth a basso consumo energetico (BLE).

2. *Transfer Learning con Deep Models*: Il transfer learning, che cade nell'area dell'adattamento al dominio e dell'apprendimento multi-task, implica l'adattamento e il miglioramento dell'apprendimento in un nuovo dominio trasferendo la rappresentazione della conoscenza appresa dai dati di un certo dominio. E' una soluzione interessante per molte applicazioni IoT dove il reperimento di dati di training non è un compito così facile. Per esempio, considerando l'allenamento di un sistema di localizzazione attraverso una BLE o il WiFi fingerprinting usando gli smart phones, i valori RSSI nello stesso momento e posizione per diverse piattaforme (iOS e Android) variano. Se abbiamo allenato un modello per una piattaforma, il modello può essere trasferito ad un'altra piattaforma senza raccogliere nuovamente un set di dati di training per quella nuova. I modelli di deep learning sono strutturati bene per il transfer learning grazie alla loro capacità di apprendere dai dati di input sia la rappresentazione a basso livello che quella astratta.

3. *Algoritmi di Online Learning con DL*: Dato che il flusso di dati generati dalle applicazioni IoT va nelle piattaforme di cloud per le analisi, il ruolo degli algoritmi di machine learning online passa più in evidenza, poiché il modello di training deve essere aggiornato dall' incrementale volume di dati. Questo è opposto a quello che supportano le attuali tecnologie, le quali si basano su tecniche di batch learning, dove l'intero set di dati di allenamento dovrebbe essere disponibile per l'allenamento e, quindi, il modello allenato non può evolvere dai nuovi dati.

Alcune architetture recenti come le VAE, le GAN e le Ladder Network ci si aspetta che abbiano un grande impatto nelle applicazioni IoT visto che coprono l'apprendimento semi-supervised.
Queste sono più favorevoli per le applicazioni IoT dove viene generato un grande aumento di dati mentre una piccola frazione di questi possono essere adottati per il machine learning.

Per le applicazioni IoT si è visto che i fattori determinanti per avere un modulo affidabile e efficiente sono i tempi di training, i tempi di esecuzione e l'aggiornamento dinamico dei modelli allenati.

2.5 Gli sviluppi del Federated Learning

Con l'introduzione del Machine Learning e in seguito del Deep Learning nel mondo informatico, ha preso piede l'interesse verso l'elaborazione dell'enorme mole di informazioni ottenute dai dispositivi mobile più performanti e capaci di avere accesso a una varietà di contenuti digitali (audio, immagini, video) con qualità maggiormente raffinata.
In un primo momento il focus fu sull'elaborazione locale con modelli predittivi per ogni ambito sociale.
Tuttavia, nella crescita esponenziale dei sistemi di trasmissione digitali, wireless e cablati, si prese in considerazione lo sfruttamento dell'interconnessione di dispositivi quali fonte di raccolta di enormi quantità di dati.
La vera e propria introduzione del termine *Federated Learning* fu portata da Brendan McMahan di Google in un primo paper del 2016[33].
La domanda a cui voleva rispondere questo nuovo approccio di calcolo era la seguente: *"E' possibile allenare un modello senza spostare e salvare i dati di allenamento ad una posizione centrale?"*
I dati utilizzati nel machine learning, infatti richiedono la centralizzazione su una macchina o in un datacenter.
Tuttavia, qui si ritorna ad un punto cruciale che ha portato questo nuovo approccio ad essere la nuova frontiera dell'elaborazione in campo IoT ovvero la crescente necessità di elaborare dati sparsi in luoghi isolati tutelando allo stesso tempo la privacy degli utenti.
La maggior parte delle informazioni non risiede più in data center centralizzati ma è distribuita uniformemente nell'ambiente.
Il problema della privacy dei dati invece è stato sottoposto a grande attenzione in parecchie giurisdizioni negli ultimi anni.
Si è quindi cercato un modo per elaborare questi ultimi, disponibili essenzialmente ad ogni modello di machine learning, senza abbandonare la fonte da cui sono stati generati.
La strategia è quella di un "edge computing" ovvero di eseguire una prima fase di calcolo

direttamente ai nodi.

Il *Federated Learning* permette ai dispositivi mobili, ma comunque ad un qualunque altro dispositivo intelligente, di apprendere collaborativamente un modello di previsione condiviso tenendo tutti i dati grezzi di training nel device.

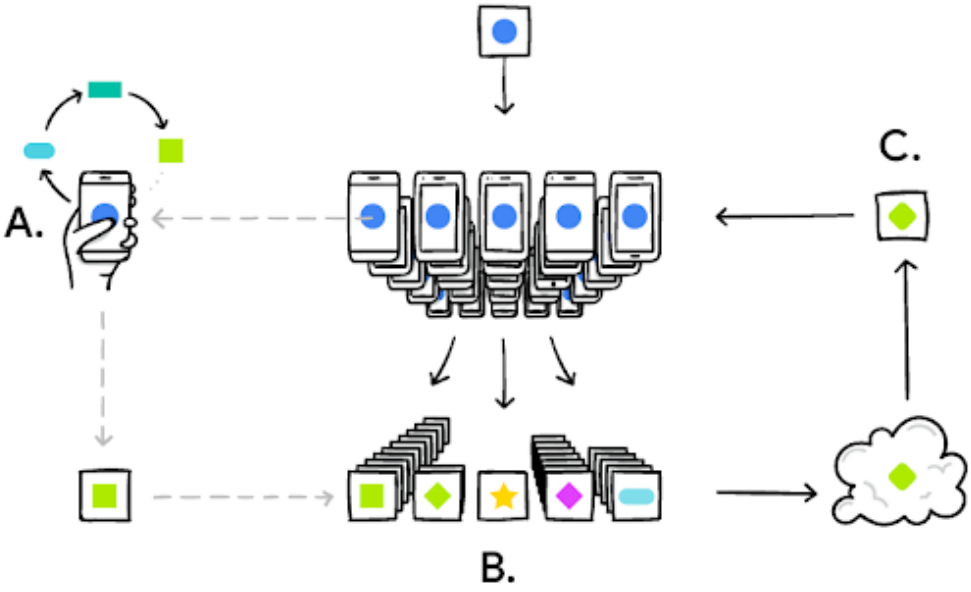


Figure 10: Schema illustrativo di sistema di Federated Learning

Ciò ha portato alla separazione dell'abilità di effettuare machine learning dalla necessità di salvare i dati nel cloud.

I dati digitali contenuti nei moderni smartphone sono sensibili alle norme sulla privacy, il che può precludere la registrazione nei data center (dove dovrebbero essere utilizzati per allenare una rete o un algoritmo centrale).

Il Federated Learning, consente di mantenere i dati di allenamento distribuiti in tutti i nodi della rete e apprendere un modello condiviso aggregando gli aggiornamenti calcolati localmente.

I nodi della rete (*Clients*) vengono coordinati da un'unità centrale (*server*).

Ogni dispositivo ha un set di addestramento che non viene mai inviato al server ed effettua un aggiornamento al modello globale corrente che possiede l'unità centrale e solamente questo aggiornamento viene trasmesso.

Un vantaggio principale di questa struttura è la separazione tra il modello di allenamento e

la necessità di avere accesso diretto ai dati grezzi.
Questo nuovo metodo rivoluzionario, nel caso di applicazioni dove l'obiettivo di allenamento può esser specificato sulla base dei dati disponibili ad ogni client, può ridurre significativamente i rischi di privacy e sicurezza limitando la sfera di attacco ad un solo device piuttosto che al dispositivo e al cloud.
Le principali attenzioni di H.Brendan McMahan e colleghi rivolsero a questa nuova tecnica furono:

- L'identificazione del problema di allenamento su dati decentralizzati da device mobili
- La scelta di un algoritmo pratico che può essere applicato ad questa situazione
- Una valutazione empirica estensiva dell'approccio

Essi introdussero un algoritmo, *Federated Averaging*, che usa il metodo Stochastic Gradient Descent(SDG) localmente su ogni client e lascia al server l'esecuzione della media dei modelli.
I problemi che riguardano il *Federated Learning* hanno in particolare le proprietà seguenti:

1. L'allenamento su dati reali provenienti da dispositivi mobili, dà un vantaggio rispetto a trattare i dati variabili presenti in un data center.
2. I dati sono per la maggior parte sensibili alla privacy ed è quindi preferibile evitare di registrarli nell'unità centrale puramente per poter allenare il modello.
3. Per i task supervisionati, le etichette nei dati possono essere dedotte naturalmente dall'interazione con l'utente.

Molti modelli che hanno un comportamento intelligente sui dispositivi mobili soddisfano i requisiti di cui sopra.
Per esempio troviamo l' *image classification* che può essere usata per prevedere quali foto saranno probabilmente viste o condivise più volte in futuro dall'utente; i *language models* che possono essere utili per migliorare il riconoscimento vocale e l'inserimento del testo nelle tastiere attraverso il miglioramento della decodifica, previsione di parole successive, e anche la previsione di risposte.
Tutti i dati potenzialmente sensibili raccolti dai dispositivi è probabile abbiano distribuzioni che sostanzialmente variano rispetto ai dataset proxy (dataset che sostituiscono quelli reali)

ovvero per esempio i messaggi di testo sono diversi dai testi che troviamo nel web o ancora le immagini scattate da un client sono diverse dalle immagini reperibili in rete.
Allo stato dell'arte per la classificazione di immagini vengono ampiamente applicate le feed-forward deep network e le reti convoluzionali mentre per il trattamento di dati vocali vengono impiegate le reti neurali ricorrenti e in particolare le LSTM.

2.5.1 Caratteristiche del sistema Federated

Il federated learning(FL) è altamente collegato al distributed learning.
Un sistema distribuito tradizionale è realizzato attraverso calcolo e memoria distribuiti.
Sebbene il FL metta una grande enfasi nella protezione della privacy, le ultime ricerche del machine learning distribuito pongono attenzione anche ai sistemi distribuiti con mantenimento della privacy.
L'elaborazione distribuita consiste nel connettere più computer in diverse posizioni attraverso la rete di comunicazione sotto il controllo di un server centrale, così che ogni computer svolga parti diverse della stessa attività per completarla.
Quindi il processo distribuito si concentra principalmente sull'accelerazione della fase di elaborazione, mentre il FL si focalizza sulla costruzione di un modello collaborativo senza perdita di privacy.
Le caratteristiche che distinguono il Federated Learning dal distributed learning sono le seguenti:

1. *Universalità per scenari interorganizational*:
   Il FL proposto da Google è una tecnologia di machine learning criptato che consente ai partecipanti di costruire un modello comune di allenamento mantenendo i dati sottostanti in locale.
   Poi il concetto originale di FL è stato esteso per riferirsi a tutte tecniche di machine learning collaborative decentralizzate con preservazione della privacy.
   Quindi il federated learning è in grado di inglobare non solo i dati divisi orizzontalmente ma anche i dati partizionati verticalmente secondo le caratteristiche che si hanno in una situazione di apprendimento cooperativo.
   L'apprendimento federato potrebbe essere esteso per portare le aziende intersettoriali in contesti federati.

Ciò consentirebbe di realizzare multipiattaforme e un valore di co-creazione regionale con la promessa di proteggere la privacy dei dati.

2. *Distribuzione massiva indipendente non identica*:

   Nel federated learning, i dati sono distribuiti in centinaia di migliaia di nodi terminali o dispositivi mobile e dati disponibili in ogni nodo non possono essere più del numero totale di nodi.

   Mentre nel sistema distribuito, lo scopo principale è di aumentare il grado di parallelismo per alleggerire il calcolo o lo stress di memoria nel server centrale.

   Il numero di nodi di un sistema distribuito non può raggiungere lo stesso ordine di grandezza come nel FL.

   Ogni dispositivo digitale oggi genera parecchi dati e non può essere confrontato con il numero totale di dispositivi.

   In questo caso, il FL è più adatto rispetto all'apprendimento distribuito poiché quest'ultimo lavora principalmente su distribuzione dati bilanciata e i.i.d a differenza dell'apprendimento federato che si concentra su distribuzioni di dati non bilanciati e non-i.i.d. a causa dell'eterogeneità tra le risorse dei dispositivi.

3. *Tecnologia decentralizzata*:

   La decentralizzazione, in senso stretto, non significa avere una decentralizzazione completa, ma che non è presente un centro definitivo.

   La decentralizzazione riguarda solo la distribuzione della conoscenza del nodo centrale.

   Non c'è un centro che determini ogni client, e ogni client influenza il modello centrale.

   L'influenza tra i nodi genera una relazione nella rete formata dai client.

   Il server dei parametri, una tipica tecnologia centralizzata e distribuita, principalmente fa uso di un server centrale che è determinante per l'invio della distribuzione dati e le risorse di elaborazione per ottenere un modello collaborativo efficiente.

   Questo tipo di metodo per l'elaborazione di dati decentralizzati dà luogo ad un doppio overhead di comunicazione dove dati che sono presenti in database diversi vengono prima raccolti per l'apprendimento e poi copiati e inviati al server centrale e successivamente allocati ad ogni client per il calcolo distribuito.

   Per il FL, ogni client è completamente autonomo, i dati non sono allocati dal centro e il processo di training non è gestito dal server.

Quindi si tratta di una tecnologia integrata che accoppia modelli di machine learning e fusione di dati attraverso una collaborazione decentralizzata.

4. *Stato equo per ogni nodo*:

   In questo spazio collaborativo, tutte le parti godono di uno stato e di certi domini uguali, per ottenere una prosperità comune.

   Il possesso di grandi quantità di dati potrebbe causare la preferenza verso organizzazioni con insiemi di dati o immagini con certi tipi di etichettature.

   Per l'allenamento congiunto nella rete di deep learning, queste istituzioni che possiedono big data potrebbero manipolare le previsioni del modello cosi che le medie e piccole aziende non contino nulla nell'apprendimento congiunto.

   Tuttavia, nel FL, la posizione di questi client che hanno dataset ridotti verrebbe promossa data l'uguaglianza per tutte le parti.

2.5.2 Categorie di apprendimento federato

Il federated learning può essere suddiviso principalmente in tre gruppi, rispettivamente, il federated learning orizzontale, verticale e il federated transfer learning.
Visto che i dati sono salvati in nodi o istituzioni diverse principalmente sono nella forma matriciale. I tre gruppi vengono di seguito illustrati.

- *FL orizzontale*:

  Nel caso del FL orizzontale rivolto a dispositivi smart o dispositivi IoT, i dati possono differire significativamente ma in contemporanea si ha una rappresentazione simile nello spazio delle feature ovvero avviene una sovrapposizione tra alcune delle feature dei dati diffusi tra i device.

  Il modello federato utilizzato da McMahan *et all* [33] è un tipo di FL orizzontale poiché i dati hanno le caratteristiche con stesse dimensioni.

  Per sopperire alla mancanza di entità classificate, Gao, Ju, Wei, Liu, Chen, and Yang [16] hanno introdotto una struttura gerarchica di FL orizzontale eterogeneo.

  La mancanza di classificazione sui dati può essere risolta poiché l'adeguamento ai domini eterogenei verrebbe adattato più volte usando ogni volta un partecipante come dominio target.

- *FL verticale*:

  Il FL verticale è adatto per casi in cui i dati sono suddivisi in direzione verticale secondo la dimensione delle caratteristiche.

  Tutte le parti detengono dati omogenei il che significa che c'è sovrapposizione parziale sull'ID del campione mentre differiscono nello spazio delle feature.

  L'aggregazione di tutto il dataset in un server comune per apprendere dal modello globale non funziona nell'apprendimento federato verticale poiché la corrispondenza tra diversi proprietari è ancora una tematica urgente da trattare.

- *Federated transfer learning(FTL)*:

  Diversamente dagli scenari del federated learning orizzontale e verticale, nella maggior parte dei casi, i dati non condividono ne lo spazio dei campioni ne lo spazio delle caratteristiche.

  Quindi, il problema principale in questa configurazione è la mancanza di etichette dati con conseguenza di informazioni di bassa qualità.

  Il transfer learning consente di spostare la conoscenza da un dominio (cioè il dominio sorgente) ad un altro dominio (cioè il dominio di destinazione) per ottenere migliori risultati di apprendimento.

2.5.3 Problematiche principali

Le principali quattro sfide associate alla ottimizzazione dei sistemi FL sono: i costi di comunicazione, eterogeneità dei sistemi, eterogeneità statistica, la protezione della privacy.

1. **Comunicazioni costose**:

   Le reti federate comprendono potenzialmente un numero enorme di dispositivi e le comunicazioni nella rete possono essere più lente di qualche ordine di grandezza rispetto all'elaborazione locale date le risorse limitate come la larghezza di banda, l'energia, e la potenza.

   Per adattare un modello ai dati generati dai dispositivi, nella rete federata, è quindi importante sviluppare metodi con comunicazioni efficienti che inviino iterativamente piccoli messaggi o aggiornamenti del modello come parte del processo di allenamento.

   Per ridurre ulteriormente la comunicazione in queste situazioni, i due aspetti da considerare sono:

(a) ridurre il numero totale dei round di comunicazione

(b) ridurre la dimensione dei messaggi trasmessi ad ogni round

2. **Eterogeneità dei sistemi**:

   Le capacità di memoria, computazionali e di comunicazione di ogni dispositivo nelle reti federate possono differire a causa della variabilità dell'hardware, della connettività di rete e della potenza.

   Inoltre, la dimensione della rete e i vincoli relativi ai sistemi su ogni dispositivo tipicamente si verificano solo per una piccola frazione dei dispositivi contemporaneamente attivi, ovvero centinaia di dispositivi attivi su una rete di milioni di device.

   E' inoltre possibile che i dispositivi attivi abbandonino ad una certa iterazione a causa di limiti di energia o di connettività.

   Queste caratteristiche a livello di sistema aggravano drammaticamente le sfide come la mitigazione dei ritardi e la tolleranza ai guasti.

   I metodi di federated learning sviluppati devono quindi:

   (a) tollerare hardware eterogenei

   (b) anticipare la bassa quantità di partecipazione

   (c) essere abbastanza resistenti a situazioni in cui si hanno device che abbandonano la rete di comunicazione

3. **Eterogeneità statistica**:

   I dispositivi frenquentemente generano e raccolgono dati in modo non-i.i.d. nella rete. Inoltre, il numero di dati tra i dispositivi può variare significativamente, e ci possono essere delle correlazioni tra i dispositivi e le loro distribuzioni associate.

   Questo paradigma di generazione dati viola le assunzioni i.i.d. frequentemente usate nell' ottimizzazione distribuita e ciò può portare complessità in termini di modellazione del problema, analisi teoriche e la valutazione empirica delle soluzioni.

   Infatti, sebbene il problema canonico di federated learning cerchi di apprendere un modello globale singolo, esistono altre alternative come apprendere simultaneamente diversi modelli locali attraverso quadri di apprendimento multitasking oppure metodi di metalearning.

Sia il multitasking che le prospettive di metalearning permettono una modellizzazione specifica del dispositivo o personalizzata.

2.5.4 Strategie di ottimizzazione

In questa sezione si illustrano più nel dettaglio le strategie usate per affrontare i problemi sopra descritti ed ottenere quindi dei migliori risultati.
Vengono inoltre introdotte le ultime tecniche sviluppate in lavori recenti nell'ambito del federated learning.
Per quanto riguarda l'efficienza nella comunicazione, le direzioni generali intraprese dai ricercatori puntano a metodi di aggiornamento locale, schemi di compressione e training decentralizzato.

- **Local Updating**

  Sono stati proposti recentemente parecchi metodi per migliorare l'efficienza della comunicazione in ambienti distribuiti consentendo di applicare una variabile ad ogni macchina in parallelo ad ogni round di comunicazione (piuttosto che calcolarle localmente e poi applicarle centralmente).

  Per funzioni obiettivo convesse, sono emersi metodi di aggiornamento locale distribuito primari duali.

  Questi approcci fanno leva sulle strutture duali per decomporre efficientemente l'obiettivo globale in sottoproblemi che possono essere risolti in parallelo ad ogni round di comunicazione.

  Parecchi metodi primari con aggiornamenti locali distribuiti sono stati proposti, i quali hanno aggiunto il beneficio di essere applicabili a funzioni obiettivo non convesse.

  Tuttavia, i metodi di ottimizzazione più efficaci sono quelli che hanno consentito un aggiornamento locale flessibile e una bassa partecipazione di client come ad esempio il più usato federated averaging.

- **Schemi di compressione**

  Sebbene i metodi di aggiornamento locale possono ridurre il numero totale di round di comunicazione, gli schemi di compressione del modello come la sparsification e la quantization possono ridurre significativamente la dimensione dei messaggi trasmessi ad ogni round.

In letteratura, sono state proposte diverse strategie, come il forzare l'aggiornamento dei modelli ad essere sparsi e di rango basso, eseguire la quantizzazione con rotazioni random strutturate, e usando una compressione con perdita e diluizione per ridurre la comunicazione dal server al dispositivo.

- **Decentralized training**

  In ambienti data center, l'apprendimento decentralizzato è stato dimostrato essere più veloce dell'allenamento centralizzato quando si opera su reti con bassa larghezza di banda o alta latenza.

  Alcuni lavori hanno proposto approcci a scadenza dove tutti i dispositivi calcolano il gradiente locale usando un numero variabile di campioni dentro un ciclo fisso globale, che aiuta a mitigare l'impatto degli stragglers (utenti che non partecipano stabilmente al processo).

  Nel federated learning, gli algoritmi decentralizzati possono, in teoria, ridurre l'alto costo di comunicazione nel server centrale.

Negli ambienti federati, c'è una variabilità nelle caratteristiche dei sistemi nella rete, poiché i dispositivi possono avere diversi hardware, connettività ad internet, ed energia disponibile. Per trattare il problema dell'eterogenicità vengono raggruppate alcune strategie principali ovvero: comunicazione asincrona, campionamento dispositivi attivo e fault tolerance.

- **Comunicazione asincrona**

  Negli ambienti data center tradizionali, schemi sincroni(i partecipanti aspettano tutti per la sincronizzazione) e asincroni(i partecipanti operano indipendentemente senza sincronizzazione) sono entrambi comunemente usati per parallelizzare gli algoritmi di ottimizzazione iterativi, ognuno comunque avente vantaggi e svantaggi.

  Gli schemi sincroni sono semplici e garantiscono un modello computazionale seriale equivalente, inoltre sono molto suscettibili agli stragglers di fronte alla variabilità dei dispositivi.

  Gli schemi asincroni sono un approccio attrattivo usato per ridurre gli stragglers negli ambienti eterogenei, particolarmente nei sistemi di memoria condivisa.

- **Campionamento attivo**

  Tipicamente, nelle reti federate, solo un piccolo sottoinsieme partecipa ad ogni round del training.

  Comunque, la maggior parte dei metodi federati, sono passivi nel senso che non mirano a influenzare quali dispositivi partecipino.

  Un approccio alternativo consiste nella selezione attiva dei partecipanti ad ogni round. Nuove politiche di selezione di dispositivi basate sulle risorse dei sistemi, con l'obiettivo che il server aggreghi quanti più aggiornamenti dei dispositivi possibili in un a predefinita finestra temporale.

  Sebbene questi metodi si focalizzino principalmente sulla variabilità di sistema per eseguire un campionamento attivo, si nota che vale la pena considerare il campionamento attivo di un piccolo insieme di dispositivi sufficientemente rappresentativi basati sulla struttura statistica sottostante.

- **Fault Tolerance**

  Per fault tolerance si intende la robustezza dell'algoritmo FL al verificarsi di malfunzionamenti o mancanza di informazioni; ad esempio, è frequente che alcuni dispositivi partecipanti possano uscire dall'algoritmo di apprendimento prima del completamento di una data iterazione.

  Una strategia pratica è di ignorare semplicemente il fallimento del dispositivo, che può introdurre la polarizzazione nello schema di campionamento dei dispositivi se i dispositivi guasti hanno specifiche caratteristiche dati.

  Per esempio, i dispositivi da aree remote possono essere maggiormente soggetti ad uscire a causa di limiti di connessione di rete e quindi, il modello federato allenato sarà polarizzato verso i dispositivi con condizioni di rete favorevoli.

  Il sistema FedProx[24] affronta l'eterogenicità dei sistemi consentendo per ogni dispositivo selezionato di eseguire un lavoro parziale conforme ai vincoli dei sistemi sottostanti e incorporare in modo sicuro gli aggiornamenti parziali attraverso un periodo prossimale.

  Un'altra opzione è il calcolo codificato, usato per tollerare il fallimento dei dispositivi introducendo una ridondanza algoritmica.

  Nel caso siano presenti degli stragglers, la codifica del gradiente e le sue varianti repli-

cano fedelmente blocchi di dati (cosi come il calcolo del gradiente su questi blocchi di dati) tra i nodi di calcolo per ottenere sia l'esatta che l'inesatta ricostruzione del gradiente autentico.

Questi metodi lanciano sfide fondamentali nelle reti federate, poiché la condivisione di dati ripetuti tra i dispositivi è spesso infattibile a causa dei vincoli di privacy e della dimensione della rete.

Sorgono inoltre problematiche di eterogenicità statistica quando il training di modelli federati viene effettuato su dati che sono altamente non-i.i.d. tra i dispositivi, sia in termini del modellamento di dati eterogenei.
Queste situazioni vengono gestite principalmente per quanto riguarda la modellizzazione di dati eterogenei e la garanzia di convergenza per dati non-i.i.d. .

- **Modellazione di dati eterogenei**

  Esiste una grande parte di letteratura in machine learning che modella l'eterogenicità statistica con metodi come il metalearning e il multitasking learning.

  Una struttura di ottimizzazione progettata per l'ambiente federato consente una personalizzazione apprendendo modelli separati ma correlati per ogni dispositivo, sfruttando una rappresentazione condivisa attraverso l'apprendimento multitasking.

  Questo metodo ha dimostrato garanzia di convergenza teorica per le funzioni obiettivo considerate ma è limitato alla sua abilità di essere scalabile a grandi reti ed è ristretto a funzioni obiettivo convesse.

  Un altro approccio[6] tratta la topologia a stella della rete federated come fosse una rete Bayesiana e l'apprendimento sulla rete viene eseguito usando una variational inference approssimata.

  Sebbene questo metodo può gestire funzioni obiettivo non convesse, è costoso da generalizzare a grandi reti federate.

- **Garanzia di convergenza per dati non i.i.d**

  L'eterogenicità statistica presenta anche nuove sfide per quanto concerne l'analisi della convergenza in situazioni federate.

  Infatti, quando i dati non sono identicamente distribuiti tra i dispositivi nella rete, i metodi come il FedAvg possono divergere in pratica, quando i dispositivi selezionati

effettuano troppi aggiornamenti locali.

Per capire le performance di FedAvg in situazioni eterogenee, è stato proposto un algoritmo(FedProx) che ha alla base l'idea che ci sia un interazione tra eterogenicità statistica e eterogenicità dei sistemi.

L'algoritmo effettua una piccola modifica al metodo FedAvg consentendo ai dispositivi di eseguire un aggiornamento parziale in base ai vincoli dei sistemi sottostanti. Le analisi di convergenza coprono inoltre la situazione in cui ogni dispositivo esegua un certo quantitativo di lavoro in locale.

Ci sono inoltre approcci euristici che cercano di gestire l'eterogenicità statistica, condividendo i dati dei dispositivi locali o alcuni dati proxy del lato server.

2.5.5 La privacy nel Federated Learning: problemi e soluzioni

La privacy è spesso un problema principale nelle applicazioni di federated learning.
Il federated learning fa un passo in avanti per proteggere i dati generati da ogni dispositivo, condividendo gli aggiornamenti del modello, cioè l'informazione sul gradiente, invece che i dati grezzi.
Tuttavia, comunicare gli aggiornamenti del modello attraverso il processo di allenamento possono comunque rivelare informazioni sensibili a terze parti o al server centrale.
Sebbene i metodi recenti cerchino di migliorare la privacy del federated learning usando strumenti come il secure multiparty computation (SMC) o privacy differenziata, questi approcci spesso garantiscono la privacy al costo di prestazioni del modello ridotte o inefficienza del sistema.
Le due categorie principali per la protezione della privacy riguardano rispettivamente la preservazione della privacy al lato client e al lato server.
Queste due branche possono intersecarsi dando origine all'approccio ibrido per il potenziamento della privacy.
I problemi di privacy in situazioni federate spesso motivano la necessità di tenere i dati grezzi localmente in ogni dispositivo.
Tuttavia, la condivisione di altre informazioni come gli aggiornamenti del modello come parte del processo di apprendimento, possono rivelare comunque alcune informazioni del terminale utente.

Oltre a produrre garanzie di privacy rigorose, è necessario sviluppare metodi che siano computazionalmente poco costosi, comunicativamente efficienti e tolleranti ai dispositivi che abbandonano, il tutto senza che la sovrapposizione comprometta l'accuratezza.
Sebbene ci siano varie definizioni di privacy nel federated learning, tipicamente, possono essere classificate in due categorie: la privacy globale e la privacy locale.
La privacy globale richiede che gli aggiornamenti del modello generati ad ogni round siano privati a tutte le terze parti non fidate oltre al server centrale, mentre la privacy locale richiede ulteriormente che gli aggiornamenti siano resi privati anche al server centrale.
Tuttavia, i metodi risultanti incorrono in costi di comunicazione extra.
Altri modelli utilizzano la privacy differenziale e offrono la privacy differenziale globale, ma questi approcci hanno un numero di iper-parametri che condiziona la comunicazione e l'accuratezza e devono essere scelti accuratamente.

2.5.6 Implementazioni future

I trend attuali del FL si concentrano in particolare sull' istituzione di conformità di sicurezza, difesa da attacchi e una diffusione efficiente cosi come elaborazioni eterogenee. In particolare si hanno:

- *Schemi di comunicazione estremi*: Rimane quindi aperto il problema di verificare quanta comunicazione sia necessaria in un sistema di federated learning.
  E' risaputo infatti che i metodi usati per il machine learning possono tollerare una mancanza di accuratezza.
  Sebbene siano stati provati sistemi di comunicazione one-shot o divide-and-conquer in ambienti di data center tradizionali, questi metodi non sono abbastanza adatti nelle reti enormi e statisticamente eterogenee.

- *Riduzione della comunicazione*: E' importante capire come le tecniche di aggiornamento locale e globale si possano comporre con un'altra per analizzare sistematicamente il tradeoff tra l'accuratezza e la comunicazione per ogni approccio.

- *Modelli innovativi di asincronia*: I due schemi di comunicazione più comunemente studiati nell'ottimizzazione distribuita sono gli approcci sincroni di massa e gli approcci asincroni.

Questi schemi sono più realistici nei ambienti data center, dove i client hanno la maggior parte del carico di lavoro ed eseguono un task dopo l'altro.
Invece, nelle reti federate, la maggior parte dei dispositivi non è attiva ad ogni iterazione.
Sarà quindi utile capire gli effetti di uno schema di comunicazione incentrato sui dispositivi, i quali possono "svegliarsi" in qualsiasi momento, richiedere attività al nodo centrale e eseguire alcuni calcoli in locale.

- *Diagnosi di eterogeneità*: I lavori recenti hanno cercato di quantificare l'eterogenicità statistica attraverso metriche come la dissimilarità locale.
  Tuttavia, queste metriche non possono essere calcolate facilmente sulla rete federata prima che avvenga l'allenamento.

- *Vincoli di privacy granulari*: Piuttosto che definire la privacy a livello globale o locale, potrebbe essere necessario definire la privacy ad un livello più granulare, poiché i vincoli di riservatezza possono essere diversi da dispositivo a dispositivo o anche tra i punti dati su un singolo device.

- *Oltre il training supervisionato*: Molti dei dati generati in reti federate realistiche possono essere non classificati o scarsamente classificati.
  Inoltre, il problema in questione potrebbe non essere adattare il modello ai dati, ma, piuttosto eseguire delle analisi dati esplorative, ottenere statistiche aggregate, o eseguire un compito più complesso, come il reinforcement learning.

- *Realizzazione di federated learning*: Ci sono alcune preoccupazioni pratiche che sorgono quando si realizzano sistemi di federated learnign.
  Problemi come il concept drift (quando il modello di generazione dei dati cambia nel tempo), variazioni diurne (quando i dispositivi esibiscono comportamenti diversi lungo la giornata o la settimana), e i problemi di cold-start (quando i dispositivi entrano nella rete) devono essere gestiti con attenzione.

2.5.7 L'Ottimizzazione Federata

Quando si parla di ottimizzazione federata nel *Federated Learning* ci si riferisce al problema di ottimizzazione implicito, creando un collegamento con l'ottimizzazione distribuita.
I punti chiave di questo processo che differisce da una tipica ottimizzazione distribuita sono:

- **Non-i.i.d.** - I dati di training generati sono diversi da dispositivo a dispositivo poiché dipendono fortemente dal comportamento dell'utente e quindi ogni dataset utente non è rappresentativo della distribuzione riguardante l'intero insieme di device.
- **Sbilanciato** - Similmente, l'uso diverso dei servizi e delle app che ogni utente ha, porta a generare quantità variabili di dati di allenamento locali.
- **Distribuzione Massiva** - Il numero di nodi terminali che partecipano nel processo di ottimizzazione tendenzialmente ci si immagina siano più alti del numero di campioni per ogni client.
- **Comunicazione Limitata** - I dispositivi mobili usualmente sono scollegati dalla rete o perlomeno dispongono di connessioni lente e costose.

Un sistema di apprendimento federato complesso deve comunque far fronte a vari altri problemi come: dataset del client che cambia in continuazione, l'accessibilità del client correlata in modo complesso alla distribuzione dei dati locali, o client che non interagiscono o che inviano aggiornamenti corrotti.
Di seguito si illustra a grandi linee la struttura del federated learning.
Si considera un insieme fisso di K client, ognuno con un dataset locale fissato.
All'inizio di ogni round, viene scelta una frazione random C di client, e il server invia lo stato dell'algoritmo globale corrente di ognuno di questi terminali di rete (cioè i parametri del modello corrente).
Ogni nodo finale selezionato esegue poi una elaborazione locale basata sullo stato globale e sul suo dataset e invia un aggiornamento al server.
Il server poi applica questi aggiornamenti al suo stato globale e il processo si ripete.
Nell'ottimizzazione effettuata nei data center i costi di comunicazione sono relativamente piccoli e prevalgono i costi di calcolo anche se ultimamente con l'uso di GPU potenziate si è riusciti a ridurli.

Nell'elaborazione federata si assiste invece ad una aumento dei costi di comunicazione (tipicamente qualche MB/s).
Inoltre i client partecipano volontariamente all'ottimizzazione quando sono pronti, collegati, e con una connessione Wi-Fi non a pagamento rilasciando tuttavia un piccolo numero di aggiornamenti al giorno.
Dato però che il dataset presente su ogni device è piccolo rispetto alla dimensione del dataset globale, e i dispositivi attuali hanno grandi capacità di calcolo, l'elaborazione diventa essenzialmente irrilevante, rispetto ai costi di comunicazione, per molti modelli.
La strategia è quindi quella di sfruttare il più possibile la capacità di elaborazione a discapito di minori round di comunicazione necessari per l'allenamento del modello.
Le due possibilità maggiori per incrementare l'elaborazione possono essere:

1. Incrementare il parallelismo usando più nodi che effettuino calcoli indipendenti in ogni round di comunicazione.

2. Incrementare il calcolo in ogni client dove viene preferito, in ogni round, un calcolo complesso alla semplice esecuzione come il calcolo del gradiente.

L'uso di questa tecnica di apprendimento federato, può essere implementata nelle più svariate modalità, mantenendo la struttura di base, ma cercando allo stesso tempo di adottare algoritmi di elaborazione sempre più raffinati ed adattabili ad ogni scenario di applicazione.

3 Le varie tecniche di Federated Learning

Si illustrano di seguito tre tecniche di Federated Learning.

Mentre molte tecniche di apprendimento distribuito ormai obsolete fanno uso di dataset i.i.d., la soluzione proposta dall'apprendimento federato ha come base fondamentale l'uso di dataset sbilanciati e non-i.i.d. .

3.1 Federated Averaging

Le applicazioni di Deep Learning di successo degli anni scorsi facevano affidamento esclusivamente su varianti dello *stochastic gradient descent(SGD)* che portò passi in avanti nell'apprendimento.

Il modello divenne infatti più suscettibile all'ottimizzazione grazie al suo adattamento ottenuto con tecniche semplici basate sul gradiente.

Invece di focalizzarsi su obiettivi non convessi della rete neurale, l'algoritmo in questione è applicabile ad ogni somma finita nella forma

$$\min_{w \in \mathcal{R}} f(w) \quad \text{where} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Tipicamente infatti per un problema di machine learning si prende $f_i(w) = l(x_i; y_i; w)$ come funzione di loss della previsione fatta con parametri w sul campione (x_i, y_i).

Si assume inoltre che ci siano K client sui quali siano suddivisi i dati, con $\mathcal{P}_k$ il set degli indici dei dati in ogni client k, con $n_k = |\mathcal{P}_k|$.

Si può riscrivere la funzione obiettivo come:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w)$$

Se la suddivisione $\mathcal{P}_k$ fosse formata distribuendo i campioni di training sui nodi terminali in maniera uniforme casuale avremmo avuto $\mathbf{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$ dove l'aspettazione è sul set di campioni assegnati ad un client fissato k.

Ciò si verificherebbe qualora vengano assunte le ipotesi i.i.d. come tipicamente avviene negli algoritmi di ottimizzazione ma non prese in considerazione nell'implementazione del FederatedAveraging.

L'SGD può essere applicato naturalmente al problema di ottimizzazione federato dove il calcolo di un singolo batch con il gradiente su un client random viene eseguito ad ogni round di

comunicazione.

Tuttavia questa tecnica, pur essendo particolarmente efficiente, richiede un grande numero di round di allenamento per produrre un buon modello.

In questo caso federato viene utilizzato come base un SGD sincrono con batch (gruppo di campioni) elevate.

Viene scelta una frazione C di client ad ogni round, e viene calcolato il gradiente del loss su tutti i dati che hanno questi client.

Questa porzione di nodi terminali controlla la dimensione globale della batch.

$C = 1$ corrisponde al gradient descent non stocastico che utilizza l'intera batch.

Fino a qui si ha la realizzazione di quello che viene chiamato FederatedSGD.

Nell'implementazione del FedSGD con $C = 1$ e un tasso di apprendimento fissato η , ogni client k calcola $g_k = \nabla F_k(w_t)$, ovvero la media del gradiente nei suoi dati locali con il modello attuale w_t.

Il server centrale invece raccoglie questi gradienti e applica l'aggiornamento al modello corrente:

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k \quad \text{dove} \quad \sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t).$$

Si ha quindi l'aggiornamento per ogni client dato da:

$$\forall k, w_{t+1}^k \leftarrow w_t - \eta g_k \quad \text{e quindi} \quad w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$$

In pratica, ogni nodo effettua localmente lo step del gradiend descent nel modello corrente usando i suoi dati locali, e il server poi effettua una media pesata dei modelli risultanti.

Il passaggio dal FederatedSGD al FederatedAveraging viene fatto aggiungendo più calcoli ad ogni client iterando l'aggiornamento locale ovvero $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$.

La quantità di elaborazioni è gestita sostanzialmente da tre parametri:

- C ovvero la frazione di client che esegue il calcolo in ogni round.
- E, il numero di volte che ogni client esegue l'allenamento sul suo dataset locale in ogni round.
- B, la dimensione della minibatch locale usata per gli aggiornamenti del client.

Per un client con n_k campioni locali, il numero di aggiornamenti locali è pari a $u_k = E\frac{n_k}{B}$ La tabella 2 riporta lo pseudo-codice dell'algoritmo. Per quanto riguarda le funzioni obiettivo

Algorithm 1 FederatedAveraging. TheK clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

inizialize w_0

for each round $t = 1, 2, ...$ **do**

  $m \leftarrow max(C \cdot K, 1)$

  $S_t \leftarrow$ (random set of m clients)

  for each client $k \in S_t$ **in parallel do**

    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): $//runonclientk$

  $\mathcal{B} \leftarrow$ (split P_k into batches of size B)

  for each local epoch i from 1 to E **do**

    for batch $b \in \mathcal{B}$ **do**

      $w \leftarrow w - \eta \nabla l(w; b)$

  return w to server

Table 2: Pseudocodice dell'Algoritmo FederatedAveraging

non convesse, la media dei modelli fatta nello spazio dei parametri si è visto che potrebbe produrre un modello non troppo efficace.

Si osserva infatti questo comportamento quando si mediano due modelli da un dataset di riconoscimento numeri allenati con inizializzazioni diverse.

Il modello, allenato sul set di training MNIST contenente 600 campioni i.i.d. non intersecati, inizia a fare overfitting dei dataset locali utilizzando un tasso di apprendimento η pari a 0.1, e attuando 240 aggiornamenti su una minibatch B di taglia 50.

Con un' inizializzazione casuale dei pesi invece si ottiene un valore di loss significativamente migliore su tutto il dataset MNIST, rispetto ai risultati ottenuti sui piccoli dataset indipendenti, effettuando proprio la media dei pesi dei due modelli($\frac{1}{2}w + \frac{1}{2}w'$).

L'algoritmo FedAvg allenato su varie architetture di modelli come il multi-layer perceptron,

reti neurali convoluzionali, reti LSTM a due layer, reti LSTM con più parole, permette di ottenere prestazioni migliori usando pochi round di comunicazione.
Questa tecnica di elaborazione sincrona si sposa perfettamente con il federated learning e apre prospetti futuri ad ulteriori implementazioni.

3.2 Tecnica di Federated Learning FedFa

Normalmente i costi di comunicazione e l'eterogenicità statistica del federated learning è dovuta ad una inconsistente distribuzione dei dati su diversi dispositivi e ad una larghezza di banda di comunicazione limitata tra i nodi terminali.
L'algoritmo FedFa cerca di raggiungere risultati migliori in termini di accuratezza, correttezza nel federated learning introducendo uno schema che sfrutta un double momentum gradient il quale riesce ad accelerare il tasso di convergenza del modello.
Gli autori propongono un algoritmo di selezione dei pesi che mette insieme la quantità di informazione dell' accuratezza e frequenza del training.
Questa modalità tende a risolvere la problematica dell' iniquità del federated learning dovuta alle preferenze per alcuni client.
Il metodo Federated Averaging effettua l'ottimizzazione congiunta di dati eterogenei e della comunicazione costosa.
L'eterogenicità è un problema fondamentale per il federated learning poiché se non viene affrontata, l'accuratezza del modello non può essere migliorata, rendendo impossibile l'uso pratico della tecnica di apprendimento.
Nel federated learning si tratta sia l'eterogenicità statistica che sistematica.
Un'altra questione importante riguarda i problemi di ottimizzazione della comunicazione, categorizzati principalmente in comunicazioni basate su larghezza di banda, latenza, e sulla topologia della rete.
Alcuni algoritmi proposti hanno mostrato una convergenza maggiore, con accuratezza migliorata e comunicazioni minori dell'algoritmo principale(FedAvg).
Sono state proposte anche tecniche di compressione quali sparsification, subsampling e quantization che possono ridurre in modo sostanziale la grandezza del messaggio per round di comunicazione.
La peculiarità del metodo FedFa sta invece nel considerare l'accuratezza e la frequenza dei

client coinvolti nel processo di apprendimento per selezionare adeguatamente i pesi ad ogni round di forte aggregazione.
Una delle tecniche più promettenti nel federated learning per ridurre il carico sulla rete è quella di far convergere velocemente e stabilmente la precisione del modello di apprendimento.
Viene quindi proposto un metodo double momentum gradient sia per la parte client che per la parte server.
La struttura dell'algoritmo proposto da Huang ed altri è composto da due parti principali:

- L'approccio del double momentum gradient.
- Le strategie appropriate di ponderazione.

3.2.1 Il Metodo Double Momentum Gradient

Il metodo del gradiente discendente ordinario è utile per risolvere problemi non troppo difficili come la regressione lineare.
Tuttavia, all'aumentare della complessità, questo strumento presenta molte limitazioni, nello specifico riguardo alla formula $w = w - \eta\nabla w$ del gradiente discendente ordinario, dove η è il tasso di apprendimento ovvero lo step di apprendimento che regola l'aggiustamento del gradiente discendente ad ogni step.
Dato che il tasso di apprendimento è fisso, il metodo del gradiente discendente ordinario convergerà più lentamente e potrà anche andare verso minimi locali.
Se consideriamo il gradiente vecchio, porterà i parametri a convergere più velocemente verso il valore ottimo.
L'idea alla base del metodo momentum gradient descent è di calcolare la media ponderata esponenziale del gradiente e usarla per aggiornare i pesi del modello.
Qualora il gradiente scenda sempre verso la stessa direzione dell'ultimo aggiornamento, l'ultimo aggiornamento funge proprio da accelerazione nella ricerca dell'ottimo.
Viceversa, qualora l'ultimo aggiornamento ha una direzione opposta al gradiente si effettua una sorta di decelerazione.
Si crea quindi un sistema in grado di accelerare la convergenza del modello di ogni nodo terminale.

La formula algebrica è la seguente:

$$m = \gamma m + \eta \Delta w_k^{t+1} \tag{1}$$

$$w_k^{t+1} = w_k^t - m \tag{2}$$

dove m è il termine di momento, γ rappresenta l'influenza del vecchio gradiente sull'aggiornamento corrente e t rappresenta i round dell'aggiornamento.

Basandosi poi sull'idea del momentum gradient, si avrà una convergenza più rapida dei parametri raccolti dal server verso il valore ottimale e una comunicazione ridotta.

La formula per l'aggiornamento durante l'allenamento dei parametri lato server è la seguente:

$$w^{t+1} = \frac{1}{k} \sum_{k \in S_t} w_k^{t+1} \tag{3}$$

$$\Delta w = w^{t+1} - w^{t+1} \tag{4}$$

$$m = \gamma m + (1 - \gamma) \Delta w \tag{5}$$

$$w^{t+1} = w^{t+1} - \eta m \tag{6}$$

con Δw il gradiente globale al lato server.

Il gradiente dal lato server viene approssimato calcolando la differenza tra l'ultimo modello globale convergente (round $t + 1$) e il modello globale al round t.

Considerando che i dati nei client sono non-i.i.d. e la selezione dei client in ogni round è random, l'idea al lato server è quella di aspettare b round prima di usare un metodo di approssimazione per l'aggiornamento del momentum gradient.

La finalità è quella di considerare maggiormente l'impatto del vecchio gradiente al lato server.

Si ha quindi:

$$w^{t+1} = w^{t+1} - \eta m \ (t + 1 = bn, n \in N^*) \tag{7}$$

ovvero il server usa la formula di cui sopra per aggiornare i pesi ogni b round.

Con un valore b adeguato il modello globale considererà il modello vecchio in modo più ampio, riducendo le oscillazioni e garantendo efficienza e corretta convergenza.

3.2.2 Strategie di weighting

Mentre il federated learning cerca di adattare il modello usando qualche tipo di funzione obiettivo che minimizza il rischio empirico sui dati generati dalla rete di nodi, minimizzare la perdita media in una rete così grande può favorire eccessivamente le performance del modello di qualcuno dei terminali.

Inoltre sebbene l'accuratezza media sia alta, quella su ogni dispositivo può essere inferiore.

Le cose possono andare peggio quando ci si trova di fronte a dati reali, che, in questo caso, nei device, sono eterogenei in termine di taglia e distribuzione.

La soluzione proposta dagli autori è quella di adottare una strategia di weighting che consenta un'equa distribuzione delle performance del modello tra i vari nodi della rete federated.

Quando si ha a che fare con campioni non-i.i.d., la qualità dei dati di ogni client è diversa e il modello di training corrispondente può essere vario.

Viene quindi spontaneo pensare che il gradiente di ogni modello locale venga processato in modo diverso nello step di centralizzazione dei dati.

Ogni client inoltre non viene selezionato ad ogni round di comunicazione e quindi il numero di volte che prende parte al meccanismo di training è diverso.

Si adopera una strategia di ponderazione in cui l'accuratezza Acc_i e la frequenza di partecipazione di ogni client f_i vengono utilizzate come fattori di weighting.

Ad ogni round di training il nodo allena il modello localmente e manda al server l'accuratezza di training, insieme al gradiente e al numero di volte che ha preso parte all'allenamento.

Il server poi effettua una aggregazione ponderata basandosi su queste informazioni che gli vengono fornite.

Vengono implementati dei fattori di modifica come misura della tecnica di ponderazione per far fronte alla variazione che avviene in ogni round per quanto riguarda l'adattabilità del modello e il numero di partecipazioni del client nell'allenamento.

Nel dettaglio, dopo che ogni client ha inviato l'informazione al server in ogni round, il server normalizza Acc_i e f_i come segue:

$$Acc_i = \frac{Acc_i}{\sum_{i=1}^{K} Acc_i} \tag{8}$$

$$f_i = \frac{f_i}{\sum_{i=1}^{K} f_i} \tag{9}$$

Le differenze di precisione nel training possono essere significative da client a client a causa della complessità del dataset locale.

Generalmente, più l'accuratezza è bassa, più il client deve apprendere informazioni.

Perciò viene usata la quantità di informazione per misurare i pesi e dare precedenza ad alcuni client nel processo di apprendimento cercando di ottenere un risultato più equo.

$$Acc_i\_inf = \begin{cases} -log_2 Acc_i, & Acc_i \neq 0, \\ -log_2 Acc_i + c, & Acc_i = 0 \end{cases} . \tag{10}$$

dove, c è una costante vicina allo 0 e serve ad evitare che l'antilogaritmo diventi negativo.

Ogni client possiede più o meno informazione a seconda di quante volte partecipa al processo di allenamento.

$$f_i\_inf = \begin{cases} -log_2(1 - f_i), & 1 - f_i \neq 0, \\ -log_2(1 - f_i + c), & 1 - f_i = 0 \end{cases} \tag{11}$$

La quantità di informazione per $Acc_i\_entropy$ e $f_i\_entropy$ viene normalizzata in questo modo:

$$Acc_i\_inf = \frac{Acc_i\_inf}{\sum_{i=1}^{K} Acc_i\_inf} \tag{12}$$

$$f_i\_inf = \frac{f_i\_inf}{\sum_{i=1}^{K} f_i\_inf} \tag{13}$$

I pesi applicati al modello aggregato si basano come detto sopra sull'accuratezza e sulla frequenza dell'apprendimento:

$$weight_i = \alpha Acc_i\_inf + \beta f_i\_inf \tag{14}$$

con $\alpha + \beta = 1$. I parametri α e β rappresentano gli effetti proporzionali di ogni modello sul modello globale.

Si fornisce nella tabella 3 lo pseudocodice.

Dalle analisi statistiche applicate a questo algoritmo i realizzatori dello studio hanno notato un miglioramento significativo sulla convergenza e sulla correttezza dell'apprendimento federato in reti eterogenee rispetto ai risultati esistenti.

FedFaAlgorithm Fairness and Accuracy Federated Learning.

Inputs: parametri di input $K, T, \eta, E, w^0, N, p_k, k = 1, \ldots, N$

for $t = 0, ..., T-1$ **do**

Il server sceglie S_t di K devices in maniera casuale (ogni device k viene scelto con probabilità p_k)

Il server manda w^t a tutti i device scelti

CLIENT:

Ogni device $k \in S_t$ aggiorna w^t con E epochs di gradient descent con momentum su F_k con uno step di dimensione η per ottenere w_k^{t+1};

Ogni device $k \in S_t$ manda $w_k^{t+1}, Acc_k^{t+1}, f_k^{t+1}$ di ritorno al server;

SERVER:

calcola la quantità di informazione per Acc_k_inf e f_k_inf;

aggiorna i pesi: $weight_k = \alpha Acc_k\_inf + \beta f_k\_inf$;

raggruppa le $w's$ come $w^{t+1} = \sum_{k \in S_t} weight_k \times w_k^{t+1}$

calcola la differenza di w tra due round con il gradiente: $\Delta w = w^{t+1} - w^t$;

calcola il termine di momentum: $m = \lambda m + (1 - \lambda)\Delta w$;

aggiorna w: $w^{t+1} - \lambda m$

end for

Table 3: FederatedAveraging

3.3 Sparse Ternary Compression

Uno dei problemi principali del sistema Federato per la preservazione della privacy è l'alto overhead di comunicazione durante il training.

In ogni iterazione della procedura di apprendimento, ogni client deve comunicare l' aggiornamento del suo modello completo il quale ha la stessa dimensione del modello allenato, che nelle apparecchiature moderne assume una grandezza nell'ordine di gigabyte con milioni di parametri.

Qualora le iterazioni diventino centinaia di migliaia su grandi dataset i costi di comunicazione di ogni nodo possono crescere rapidamente a più di un *petabyte*.

La limitazione della bandalarga o il costo di comunicazione causano una improduttività o impraticabilità del Federated Learning.

Per ovviare a questa problematica alcuni metodi proposti hanno cercato di ridurre la quantità di dati trasmessi tra i nodi e l'unità centrale.

Il limite che hanno alcuni dei metodi proposti è quello di ridurre la quantità di dati trasmessi al server senza però comprimere in flusso in download oppure di funzionare bene solo nel caso si abbiano dati indipendenti e identicamente distribuiti, il che non avviene praticamente mai nel Federated Learning.

Sattler e gli altri autori propongono invece un approccio[12] chiamato Sparse Ternary Compression (STC) ovvero un metodo di compressione specifico per l' ambiente dell' apprendimento federato.

Il numero totale di bit che devono essere scaricati e inviati ad ogni aggiornamento dell' apprendimento è dato dalla seguente formula:

$$b^{up/down} \in \mathcal{O}(\underbrace{N_{iter} \times f}_{\#\text{ updates}} \times \underbrace{|\mathcal{W}| \times (H(\Delta \mathcal{W}^{up/down}) + \eta)}_{\text{update size}}) \tag{15}$$

dove N_{iter} è il numero totale di iterazioni eseguite da ogni dispositivo, f è la frequenza di comunicazione, $|\mathcal{W}|$ la taglia del modello, $H(\Delta \mathcal{W}^{up/down})$ l'entropia degli aggiornamenti dei pesi scambiati durante l'upload e il download e η è la differenza tra la vera dimensione aggiornata e la minima dimensione aggiornata cioè l'inefficienza nella codifica.

Per ridurre questa quantità ci sono tre vie possibili:

a. diminuire la frequenza di comunicazione f

b. ridurre l'entropia dell'aggiornamento dei pesi $H(\Delta \mathcal{W}^{up/down})$ con tecniche di compressione con perdita

c. usare codifiche più efficienti per ridurre η

Basandosi sulle proprietà del Federated Learning quali, dati non-i.i.d. e sbilanciati, grande numero di nodi della rete, grande quantità di parametri del server, partecipazione parziale all'apprendimento, energia e memoria dei device limitate, gli algoritmi di ottimizzazione dovrebbero soddisfare le seguenti richieste:

- Compressione dei dati inviati e ricevuti.
- Robustezza a dati non-i.i.d. sbilanciati e piccole dimensioni dei gruppi di dati.
- Efficienza nel gestire un grande numero di client e la loro partecipazione parziale.

Gran parte dei metodi utilizzati finora non riescono a soddisfare tutte queste richieste.
Le strategie che cercano di ridurre sia la compressione in download che in upload sono molto sensibili alle distribuzioni di dati non-i.i.d. .
Viene proposto un nuovo protocollo di comunicazione in grado di sopperire a questa mancanza.
Nell'ampio dominio del deep learning per l'efficientamento delle comunicazioni distribuite i rami di ricerca si sono focalizzati principalmente in tre opzioni ovvero: metodi relativi al ritardo della comunicazione dove viene ridotta la frequenza di comunicazione f, metodi di sparsificazione ovvero basati sulla riduzione dell'entropia degli aggiornamenti attraverso una limitazione dei cambiamenti su piccola parte dei parametri e infine metodi di fitta quantizzazione ove viene ridotta l'entropia dell'aggiornamento dei pesi restringendo gli aggiornamenti ad un set di valori ridotto.

3.3.1 Limitazioni dei metodi di compressione

Le tecniche di deep learning efficientemente distribuito considerano quasi esclusivamente distribuzioni di dati i.i.d. tra i client, in modo tale che i gradienti locali siano imparziali rispetto al gradiente su tutti i dati secondo

$$\mathrm{E}_{x\sim p_i}[\nabla_{\mathcal{W}} l(x, \mathcal{W})] = \nabla_{\mathcal{W}} R(\mathcal{W}) \quad \forall i = 1, \ldots, n \tag{16}$$

dove p_i rappresenta la distribuzione dei dati nel client i-esimo e $R(\mathcal{W})$ è la funzione di rischio empirico sui dati di training abbinati.
Nel caso del Federated Learning questa assunzione non è valida poiché possiamo sperare di imparzialità soltanto nella media

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{x^i\sim p_i}[\nabla_{\mathcal{W}} l(x^i, \mathcal{W})] = \nabla_{\mathcal{W}} R(\mathcal{W}) \tag{17}$$

mentre il gradiente del singolo nodo sarà influenzato dal dataset locale nel modo seguente

$$\mathrm{E}_{x\sim p_i}[\nabla_{\mathcal{W}} l(x, \mathcal{W})] = \nabla_{\mathcal{W}} R_i(\mathcal{W}) \neq \nabla_{\mathcal{W}} R(\mathcal{W}) \quad \forall i = 1, \ldots, n \tag{18}$$

Ma questo viola il secondo requisito per il Federated Learning.
Come base per la nuova strategia viene considerata la sparsificazione *Top-k*.
Si tratta di un metodo di compressione che non soffre di divergenza dei pesi come il Federated Averaging e funziona egregiamente con dati iid.
Tuttavia, in questo metodo avviene solo la compressione della comunicazione di upload dati.
La tecnica proposta cerca di risolvere la questione focalizzandosi su:

- Aggiunta di una compressione del flusso di download per avere una comunicazione efficiente dal server ai clients.
- Implementazione di un meccanismo di caching per tenere i client sincronizzati nell'eventualità di una partecipazione parziale degli stessi.

Vengono inoltre implementate la quantizzazione e una codifica lossless ottimale per l'update dei pesi del modello.

3.3.2 Estensione della Compressione del flusso di Download

Sia $top_{p\%} : \mathrm{R}^n \to \mathrm{R}^n, \Delta\mathcal{W} \to \widetilde{\Delta\mathcal{W}}$ l'operatore di compressione che mappa un aggiornamento di weight (flatten) $\Delta\mathcal{W}$ su un aggiornamento sparsificato $\widetilde{\Delta\mathcal{W}}$ impostando tutti gli elementi a zero tranne la frazione $p\%$ con la grandezza più alta.

La regola di aggiornamento della tecnica Top-k è la seguente

$$\Delta\mathcal{W}^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{top_{p\%}(\Delta\mathcal{W}_i^{(t+1)} + A_i^{(t)})}_{\widetilde{\Delta\mathcal{W}_i^{(t+1)}}} \tag{19}$$

$$A_i^{(t+1)} = A_i^{(t)} + \Delta\mathcal{W}_i^{(t+1)} - \widetilde{\Delta\mathcal{W}_i^{(t+1)}} \tag{20}$$

con l'inizializzazione $A_i^{(0)} = 0 \in \mathbb{R}^n$ in tutti i client.

Si ha che, nel caso peggiore, il numero di elementi non nulli nell'aggiornamento $\Delta\mathcal{W}^{(t+1)}$ che viene trasmesso dal server ai client cresce linearmente con il numero di nodi terminali che partecipano mentre gli aggiornamenti $\Delta\mathcal{W}_i^{(t+1)}$ in upload sono sempre sparsi.

Se il tasso di partecipazione supera la sparsità inversa $\frac{1}{p}$, l'aggiornamento $\Delta\mathcal{W}^{(t+1)}$ diventa essenzialmente denso.

Come soluzione gli autori propongono di applicare lo stesso meccanismo di compressione nel server al fine di comprimere il flusso di download.

La formula diventa:

$$\widetilde{\Delta\mathcal{W}^{(t+1)}} = top_{p\%}\Big(\frac{1}{n}\sum_{i=1}^{n} \underbrace{(top_{p\%}(\Delta\mathcal{W}_i^{(t+1)} + A_i^{(t)})}_{\widetilde{\Delta\mathcal{W}_i^{(t+1)}}} + A^{(t)}\Big) \tag{21}$$

con l'update residuo lato client e lato server

$$A_i^{(t+1)} = A_i^{(t)} + \Delta\mathcal{W}_i^{(t+1)} - \widetilde{\Delta\mathcal{W}_i^{(t+1)}} \tag{22}$$

$$A^{(t+1)} = A^{(t)} + \Delta\mathcal{W}^{(t+1)} - \widetilde{\Delta\mathcal{W}^{(t+1)}} \tag{23}$$

Questa regola di aggiornamento può essere effettuata sia per l'upload che per il download come combinazione di della compressione in upload pura e di una maschera di filtraggio generalizzata.

Ogni client può usare la maschera filtro di sparsificazione M_i durante l'upload e stessa cosa può fare il server usando una maschera M durante la fase di download.

Si può quindi arrivare allo stesso aggiornamento $\widetilde{\Delta\mathcal{W}^{(t+1)}}$ se tutti i client usano le maschere

filtranti $\widetilde{M}_i = M_i \odot M$.

Questo nuovo tipo di aggiornamento ci permette quindi di avere un tasso di sparsità maggiore rispetto alla sparsificazione *Top-k*.

3.3.3 Aggiornamento dei Pesi Caching

Nello scenario tipo del federated learning non tutti i nodi partecipano al processo di apprendimento in un round di comunicazione particolare.

Poiché i client non scaricano il modello completo $\mathcal{W}^{(t)}$ ma solo gli aggiornamenti complessi del modello $\Delta\widetilde{\mathcal{W}}^{(t)}$, ciò pone una problematica nella sincronizzazione dei client.

Per risolvere questo problema e limitare il carico utile dei client viene usato un meccanismo di caching sul server.

Assumendo che gli ultimi τ round di comunicazione abbiano prodotto gli aggiornamenti $\{\Delta\widetilde{\mathcal{W}}^{(t)} \ | t = T-1, \ldots, T-\tau\}$ il server può depositare tutte le somme parziali di questi aggiornamenti fino ad un certo momento $\{P^{(s)} = \sum_{t=1}^{s} \Delta\widetilde{\mathcal{W}}^{(T-t)} \ | s = 1, \ldots, \tau\}$ insieme al modello globale $\mathcal{W}^{(T)} = \mathcal{W}^{(T-\tau-1)} + \sum_{t=1}^{\tau} \Delta\widetilde{\mathcal{W}}^{(T-t)}$.

In questo modo ogni volta che un nodo della rete vuole sincronizzarsi con il server può scaricare $P^{(s)}$ o $\mathcal{W}^{(T)}$ a seconda dei round a cui non ha partecipato.

Per aggiornamenti sparsi generali il limite per l'entropia è

$$H(P^{(\tau)}) \leq \tau H(P^{(1)}) = \tau H(\Delta\widetilde{\mathcal{W}}^{(T-1)}) \tag{24}$$

Si ha quindi che con l'aumento dei round di comunicazione la dimensione del download aumenterà linearmente, problema di tutti i metodi di compressione che comunicano solo i parametri aggiornati al posto dell'aggiornamento del modello intero.

Il numero medio di round in cui un client non partecipa all'aggiornamento, rappresentato da η, è tollerabile poiché il processo di download è più veloce e ha molta più banda disponibile rispetto al collegamento in upload.

La partecipazione parziale nel processo di allenamento causa anche un rallentamento nella convergenza del training federato.

3.3.4 Eliminazione della ridondanza

Per migliorare ulteriormente la tecnica di sparsificazione si può cercare di eliminare le fonti di ridondanza nella comunicazione.

Viene effettuata quindi una combinazione tra la sparsità e la binarizzazione.

La comunicazione con piena precisione di frazioni di grandi componenti o, la non comunicazione che causa uno sbilanciamento nella precisione, fa perdere importanti informazioni.

Si è visto poi che vengono ottenuti risultati di compressione migliori quando la sparsificazione viene usata con la quantizzazione degli elementi non nulli.

Viene effettuata quindi una compressione binaria e gli elementi rimanenti top-k degli aggiornamenti sparsificati alla grandezza media nel numero di nodi, ottenendo un tensore ternario di valori $\{-\mu, 0, \mu\}$.

Il metodo di quantizzazione è presentato dall'algoritmo seguente:

STC Algorithm Sparse Ternary Compression.

Inputs: tensore flatten $T \in \mathrm{R}^n$, sparsità p

output: tensore ternario sparso $T^* \in \{-\mu, 0, \mu\}^n$

$k \leftarrow max(np, 1)$

$v \leftarrow top_k(|T|)$

$mask \leftarrow (|T| \geq v) \in \{0, 1\}^n$

$T^{masked} \leftarrow mask \odot T$

$\mu \leftarrow \frac{1}{k} \sum_{i=1}^{n} |T_i^{masked}|$

return $T^* \leftarrow \mu \times sign(T^{masked})$

Table 4: FederatedAveraging

Questo step di ternarizzazione riesce a ridurre l'entropia dell'aggiornamento da

$$H_{sparse} = -plog_2(p) - (1-p)log_2(p) + 32p \tag{25}$$

a

$$H_{STC} = -plog_2(p) - (1-p)log_2(p) + p \tag{26}$$

se confrontata con la classica sparsificazione.

Per comunicare poi un set di tensori ternari sparsi prodotti dall'algoritmo di compressione,

viene effettuata una codifica senza perdite.

Vengono trasferite solo le posizioni degli elementi non nulli presenti nei tensori resi flatten con un bit per ogni aggiornamento non nullo, per identificare il segno di μ.

Invece di comunicare la posizione degli elementi non nulli, è più conveniente comunicare la distanza tra loro.

Assumendo un pattern di sparsità random si ha che per grandi valori di $|W|$ e $k = p|W|$, le distanze sono circa geometricamente distribuite con probabilità di successo uguale al tasso di sparsità p.

Viene quindi utilizzato il codice Golomb per codificare le distanze in modo ottimale.

Questa codifica riduce il numero medio di bit di posizione di

$$\bar{b}_{pos} = \mathbf{b}^* + \frac{1}{1-(1-p)^{2^{b^*}}} \tag{27}$$

con $\mathbf{b}^* = 1 + \lfloor log_2(\frac{log(\phi-1)}{log(1-p)}) \rfloor$ e il tasso ottimale $\phi = \frac{\sqrt{5}+1}{2}$.

La struttura completa della procedura di compressione che definisce la compressione tramite sparsificazione di sia il flusso in upload che in download, la ternarizzazione e la codifica ottima degli aggiornamenti è rappresentata dall'algoritmo in Tabella 5.

La strategia fin qui illustrata può esser considerata come un paradigma alternativo per una ottimizzazione federata comunicativamente efficiente che preferisce un collegamento ad alta frequenza e basso volume rispetto ad uno con bassa frequenza e alto volume.

La sua applicazione può avvenire in casi di ambienti con canali di comunicazione tra server e client a bassa latenza e larghezza di banda limitata.

Algorithm2 Apprendimento Federato con Parametri di Server utilizzando una Sparse Ternary Compression.

Inputs: parametri iniziali $\mathcal{W}$

Output: parametri migliorati $\mathcal{W}$

Inizializzazione:tutti i client $C_i, i = 1, \ldots, N$ inizializzati con stessi parametri ($\mathcal{W}_\rangle \leftarrow \mathcal{W}$). Ogni client ha un dataset diverso D_i, con $|y : (x, y) \in D_i|$ =[Classi per client] di dimensione $|D_i| = \varphi_i| \cup_j D_j|$. I residui vengono inizializzati a zero ($\Delta\mathcal{W}, \mathcal{R}_i, \mathcal{R} \rightarrow 0$).

for $t = 1, \ldots, T$ **do**

for $i \in I_t \subseteq \{1, \ldots, N\}$ **in parallel**

do

Client C_i esegue:

msg$\leftarrow$ download$_{S \rightarrow C_i}$(msg)

$\Delta\mathcal{W} \leftarrow$decode(msg)

$\mathcal{W}_i \leftarrow \mathcal{W}_i + \Delta\mathcal{W}$

$\Delta\mathcal{W}_i \leftarrow \mathcal{R}_i +$ SGD$(\mathcal{W}_i, D_i, b) - \mathcal{W}_i$

$\Delta\widetilde{\mathcal{W}}_i \leftarrow STC_{p_{up}}(\Delta\mathcal{W}_i)$

$\mathcal{R}_i \leftarrow \Delta\mathcal{W}_i - \Delta\widetilde{\mathcal{W}}_i$

msg$_i \leftarrow$ codifica$(\Delta\widetilde{\mathcal{W}}_i)$

upload$_{C_i \rightarrow S}$(msg$_i$)

end

Il server S esegue:

raggruppa$_{C_i \rightarrow S}(\Delta\widetilde{\mathcal{W}}_i), i \in I_t$

$\Delta\mathcal{W} \leftarrow \mathcal{R} + \frac{1}{|I_t|} \sum_{i \in I_t} \Delta\widetilde{\mathcal{W}}_i$

$\Delta\widetilde{\mathcal{W}} \leftarrow$STC$_{p_{down}} \Delta\widetilde{\mathcal{W}}$

$\mathcal{R} \leftarrow \Delta\mathcal{W} - \Delta\widetilde{\mathcal{W}}$

$\mathcal{W} \leftarrow \mathcal{W} + \Delta\widetilde{\mathcal{W}}$

msg$\leftarrow$ codifica$(\Delta\widetilde{\mathcal{W}})$

trasmette$_{S \rightarrow C_i}$(msg)$, i = 1, \ldots, M$

end

return $\mathcal{W}$

Table 5: Efficient Sparse Ternary Compression

4 Autoencoder-Classificatore per il Federated Learning

Il progetto di tesi viene svolto all'interno del paradigma dell'Artificial Intelligent of Things (AIoT) ovvero l'integrazione tra l'intelligenza artificiale e l'ecosistema IoT.
Si è cercato di progettare una architettura di Deep Learning nella forma di un Autoencoder-Classificatore per la previsione di immagini in una rete di sensori visivi.
Il sistema è pensato per l'utilità in ambienti smart dove i dati video vengono convertiti in informazioni utili per la fornitura di servizi IoT o per applicazioni intelligenti.
Le limitazioni che si incontrano in un ambiente intelligente principalmente, come già menzionato, riguardano in particolare la tutela della privacy e i collegamenti in rete che non sempre permettono grandi quantità di banda larga per quanto concerne la trasmissione di considerevoli quantità di dati in formato grezzo.
Tuttavia, la recente tecnologia mobile ha portato notevoli prestazioni di calcolo e ciò permette di sfruttare l'elaborazione locale ai fini di una maggiore precisione e affidabilità sul modello globale ottenuto nell'apprendimento centralizzato.
Vista la quantità di dati da trasmettere non è possibile infatti pensare ad un sistema centralizzato dove raccogliere i dati poiché ciò comporterebbe notevoli costi e ritardi di trasmissione.
Tecniche di Federated Learning supportate da architetture di Deep Learning permettono quindi ai nodi della rete di allenare un modello di rete neurale efficientemente raccogliendo i dati in locale.
Si vuole cercare di capire se vi è un miglioramento delle prestazioni di deduzione di un modello di classificazione immagini utilizzando la ricostruzione dei dati originali di input, senza la compromissione della privacy.
L'ambiente di Federated Learning che si vuole analizzare comprende un insieme di client equipaggiati con un'architettura di deep learning costituita da un Autoencoder e un Classificatore i quali partecipano al processo di training e da un'unità centrale, avente la stessa architettura, che si occupa della fase di aggregazione dei modelli e della classificazione finale.
Durante il processo di apprendimento federato, si presume che ogni client alleni il modello di Autoencoder e il modello di Classificazione.
Una volta allenato il modello spedisce sia il file contenente i parametri dell'architettura

Autoencoder-Classificatore sottoforma di file .h5 compresso che il bitstream dell'immagine codificata dall'encoder presente nella struttura Autoencoder.
Il nodo centrale procede con la raccolta dei parametri provenienti da ogni modello ed effettua una media dei parametri al fine di ottenere un modello globale ottimizzato.

4.1 Apprendimento distribuito per la classificazione di immagini

In seguito all'avanzamento delle tecnologie mobili e la popolarità di dispositivi smart gli utenti oggi possono produrre una grande quantità di materiale audiovisivo che ha i potenziali come vettore per la creazione di tecnologie IoT emergenti o applicazioni su città intelligenti. Con lo sviluppo delle tecniche di machine learning è ora possibile trasformare questi contenuti digitali in dataset utilizzabili in tempo reale.
Basandosi sul nuovo paradigma, uno scenario prospettato potrebbe essere quello di una piattaforma AIoT che raccoglie video da un pool di sensori, effettui delle analisi per mezzo dell'intelligenza artificiale al fine di ricavare informazioni utili, e infine fornisca servizi di analisi ai proprietari dei dispositivi ai nodi o ad altri consumatori. Il funzionamento di base

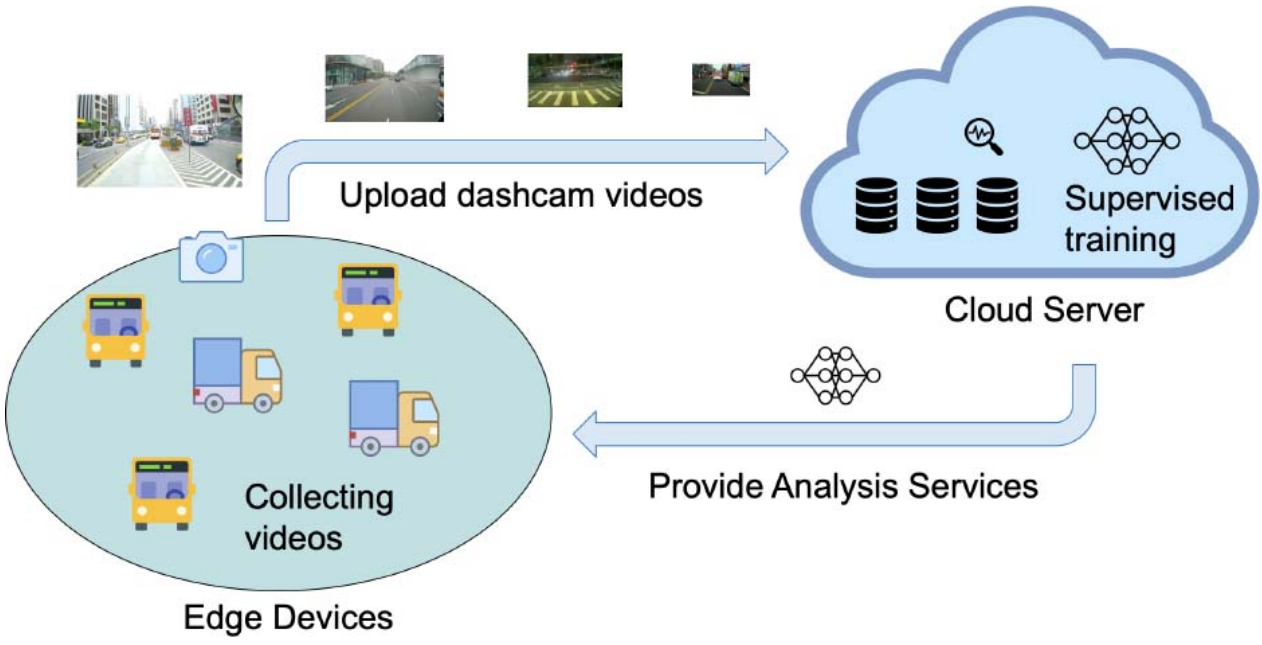


Figure 11: Scenario di una piattaforma di analisi video

di un sistema di apprendimento ai nodi nel campo del Federated Learning è il seguente:

1. Ogni device mette a punto le immagini dal video ed effettua l'apprendimento per allenare il modello.

2. I pesi del modello allenato di ogni nodo della rete vengono caricati periodicamente al server cloud.

3. Il cloud server esegue l'apprendimento federato mediando i pesi per la creazione di un nuovo modello.

4. Il modello mediato viene poi scaricato da ogni nodo della rete e rimpiazzato con il modello originario per poter effettuare di nuovo un miglioramento del modello locale partendo dal modello globale.

Alcune problematiche dell'apprendimento da sequenze video, ma comunque presente anche in altre situazioni, riguarda la differenza tra le distribuzioni delle categorie degli oggetti di interesse da dispositivo a dispositivo.
Questa diversità di dati video risulta avere una divergenza di peso, cioè la differenza dei pesi (*Weight Divergence*) tra due processi di training diversi inizializzati con gli stessi pesi, si verifica quando il server cloud cerca di unire i pesi dei modelli dei nodi terminali.

4.2 Struttura dell'autoencoder convoluzionale

L'autoencoding è un algoritmo di compressione dati dove le funzioni di compressione e decompressione sono:

- *Specifiche per i dati*: ovvero sono in grado di comprimere dati simili a quelli su cui è stato effettuato l'allenamento. La stessa rete applicata a immagini contenenti oggetti diversi restituisce risultati completamente diversi.

- *Con perdita*: ovvero gli output decompressi sono degradati rispetto ai dati di input originali

- *Apprese automaticamente da campioni*: cioè è facile allenare istanze specifiche dell'algoritmo che funzioneranno bene in un tipo specifico di input e non necessitano di ingegnerizzazione umana.

Si tratta di una tecnica di apprendimento definita *self-supervised*, ossia un'istanza specifica di apprendimento supervisionato dove gli obiettivi sono generati dai dati di input.
Usualmente le funzioni di compressione e decompressione dell'autoencoder sono implementate con reti neurali.
Per costruire un autoencoder sono essenziali tre elementi: una funzione di codifica, una funzione di decodifica e una funzione di perdita che misuri la quantità di informazione persa tra la rappresentazione compressa dei dati e quella decompressa.
L'encoder e il decoder vengono implementati con funzioni parametriche ovvero tipicamente con reti neurali con parametri che potranno essere ottimizzati per minimizzare la perdita nella ricostruzione usando lo Stochastic Gradient Descent (SGD) o altri algoritmi di ottimizzazione.
Gli autoencoder sono tipicamente usati nel *denoising* e nel *dimensionality reduction*.
Con vincoli di dimensionalità e sparsità possono apprendere proiezioni di dati maggior accuratamente del Principal Component Analysis (PCA) e altre tecniche basilari.
In figura 12 se ne illustra la composizione di base.

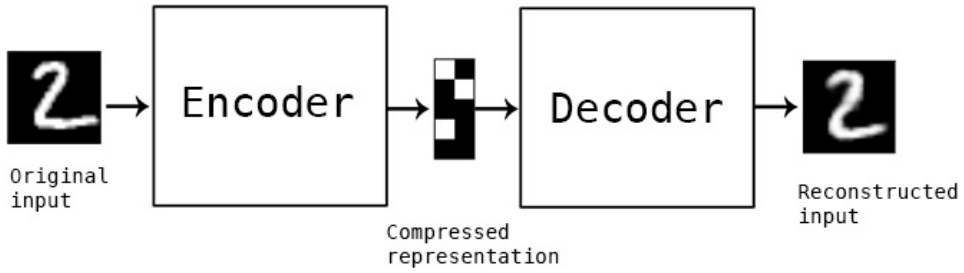


Figure 12: Schema generale di un Autoencoder

La struttura dell'autoencoder è sostanzialmente costituita da due reti convoluzionali con layer di convoluzione e layer di pooling intervallati tra loro.
Il convolutional layer Conv2D è la parte principale dell'architettura ed effettua una operazione di convoluzione al fine di estrarre le caratteristiche ad alto livello come ad esempio i bordi dall'immagine di input.
L'elemento principale che si trova nell'operazione di convoluzione è il filtro (Kernel) costituito da una matrice.
Nel nostro caso si è usato un convolutional layer con un Kernel 3 x 3.

Il secondo elemento della struttura è il pooling layer, simile al layer convoluzionale, serve per ridurre la dimensione spaziale delle features convolute.
Permette inoltre di diminuire la potenza computazionale richiesta per l'elaborazione dei dati e allo stesso tempo è utile per estrarre le feature predominanti invarianti rispetto alla posizione e alla rotazione.
In questo caso il MaxPooling restituisce il valore massimo della porzione di immagine coperta dal Kernel.

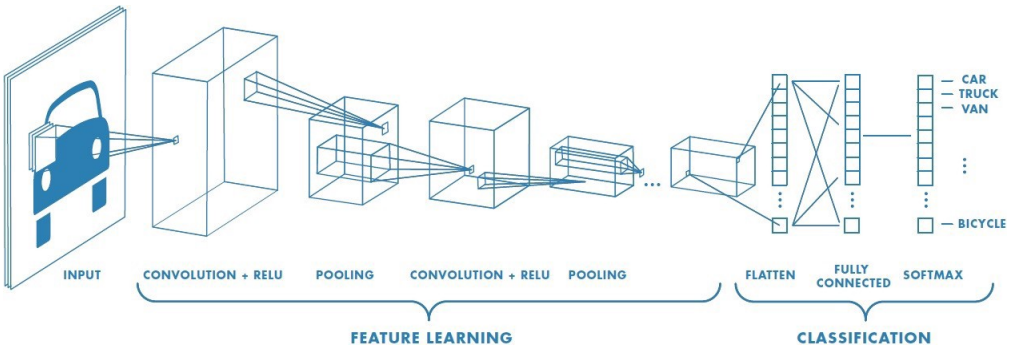


Figure 13: Schema generale di una rete convoluzionale

La peculiarità di questa struttura di rete di apprendimento sta nel creare un modello che abbia "imparato" qualcosa dall'allenamento dei dati per ricostruire fedelmente in output all'immagine ricevuta in input. Pur non essendo una vera e propria tecnica di unsupervised learning, ma una tecnica di apprendimento self-supervised, la quale cerca di apprendere caratteristiche peculiari da obiettivi sintetici interessanti e funzioni di perdita.

4.2.1 Ambiente di Sviluppo e Dataset

L'intero lavoro si è svolto utilizzando la piattaforma di cloud computing Google Colab, uno spazio di lavoro dove è possibile sfruttare la capacità di calcolo di server condivisi che mettono a disposizione un framework di calcolo online con capacità di elaborazione grafica avanzate e memoria condivisa.
Si è utilizzato il linguaggio di programmazione Python poiché si integra con gli strumenti di sviluppo per progettare strutture di machine learning e deep learning.
Nel caso in questione si è fatto uso della strumentazione messa a disposizione dall'API Keras, una libreria open source per la progettazione di reti neurali e l'apprendimento automatico in linguaggio Python che si interfaccia con l'ecosistema TensorFlow.

Come set di dati è stato utilizzato il MNIST ovvero un database avente 60000 campioni per addestramento e 10000 campioni per test di cifre scritte a mano con dimensione di 28 x 28 pixel.
Questo database è utile per apprendere tecniche e metodi di riconoscimento modelli su dati del mondo reale con sforzi di formattazione e pre-elaborazione minimi.

4.3 Calcoli preliminari e pre-progettazione

Per la realizzazione del lavoro di tesi sono state effettuate delle pre-elaborazioni al fine di capire come progettare la struttura complessa della rete neurale.
Si è partiti dal realizzare un autoencoder convoluzionale ovvero una rete neurale composta da layer convoluzionali intervallati da layer di pooling.
L' autoencoder iniziale è costituito da una rete neurale convoluzionale con encoder realizzato da tre layer di convoluzione spaziale intersecati con 3 layer di max pooling per dati spaziali in 2D.
Il codificatore, presente al lato client, rappresenta la parte della rete che comprime l'input in uno spazio di variabili latenti e che può essere rappresentato da una funzione $h = f(x)$ ove x rappresenta l'input.
Il codice viene rappresentato di seguito

```
#ENCODER
x1 = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(
    input_img)
x2 = layers.MaxPooling2D((2, 2), padding='same')(x1)
x3 = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x2)
x4 = layers.MaxPooling2D((2, 2), padding='same')(x3)
x5 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(x4)
encoded = layers.MaxPooling2D((2, 2), padding='same', name = '
    autoencoder_output')(x5)
```

Il decoder, è stato progettato specularmente, dovendo seguire la struttura di un autoencoder che prevede la ricostruzione dell'immagine di input.
Esso costituisce la parte della rete neurale che si occupa di ricostruire l'input sulla base delle informazioni precedentemente raccolte ed è rappresentato dalla funzione $r = g(h)$ con h il flusso di bit in input.

Nel progettare l'intera struttura si ipotizza che il decodificatore sia collocato lato server nel sistema federato.

Se ne riporta il codice di seguito.

```
#DECODER
x6 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(encoded)
x7 = layers.UpSampling2D((2, 2))(x6)
x8 = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x7)
x9 = layers.UpSampling2D((2, 2))(x8)
x10 = layers.Conv2D(16, (3, 3), activation='relu')(x9)
x11 = layers.UpSampling2D((2, 2))(x10)
decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same',
    name = 'autoenc_out')(x11)
```

L'autoencoder può esssere quindi descritto nel suo complesso dalla funzione $d(f(x)) = r$ dove r rappresenta la ricostruzione dell'input originale x.

Da questa rete neurale si riescono ad estrarre informazioni dallo spazio delle variabili latenti h in modo tale che possa assumere delle caratteristiche a noi utili.

Ciò avviene costringendo lo spazio h a dimensioni minori di quelle di x e viene chiamato *Undercomplete*.

Qualora invece non si diano vincoli alla rete o la dimensione del sottospazio latente abbia la stessa grandezza o grandezza maggiore dello spazio di partenza allora la rete si limita al compito di copiare l'input in output senza apprendere nulla.

Come operazione preliminare è stata creata una funzione di shuffling randomico del dataset MNIST per evitare di avere il dataset originario e poter operare su una istanza casuale.

```
def shuffle_data(training_set_x,training_set_y,test_set_x,test_set_y):
  demo_x_train = training_set_x.reshape((len(training_set_x),28*28))
  demo_x_test = test_set_x.reshape((len(test_set_x),28*28))

  demo_xy_train = np.c_[demo_x_train,training_set_y]
  demo_xy_test = np.c_[demo_x_test,test_set_y]

  rng = np.random.default_rng()
  shuffled_demo_xy_train = rng.permutation(demo_xy_train,axis=0)
  shuffled_demo_xy_test = rng.permutation(demo_xy_test,axis=0)
  shuffled_x_train = shuffled_demo_xy_train[:,0:28*28]
```

```
12  shuffled_y_train = shuffled_demo_xy_train[:,28*28:28*28+1]
13  shuffled_x_test = shuffled_demo_xy_test[:,0:28*28]
14  shuffled_y_test = shuffled_demo_xy_test[:,28*28:28*28+1]
15  training_set_x = shuffled_x_train
16  training_set_y = shuffled_y_train
17  test_set_x = shuffled_x_test
18  test_set_y = shuffled_y_test
19
20  training_set_x = np.reshape(training_set_x, (len(training_set_x), 28,
     28))
21  test_set_x = np.reshape(test_set_x, (len(test_set_x), 28, 28))
22  training_set_y = np.reshape(training_set_y, len(training_set_y))
23  test_set_y = np.reshape(test_set_y, len(test_set_y))
24
25  return training_set_x, training_set_y, test_set_x, test_set_y
```

Si è quindi proceduto cercando di far andare in overfitting la rete creata, ovvero buone performance sui dati di training e basse prestazioni su quelli di test.

Questa operazione è stata eseguita per capire la pesantezza della rete in termini di bit rate e hidden features.

Nella parte centrale dell'autoencoder si trova infatti la rappresentazione compressa che può essere trasmessa dal nodo terminale verso il server e quindi decodificata.

Nella valutazione sono stati compresi l'MSE (*Mean Squared Error*) e la dimensione del file .h5, il quale contiene in sostanza i pesi dell'intero modello e ci da un'idea sommaria del bit rate parziale che il client deve trasmettere.

Si sono adottate 9 configurazioni possibili per capire l'andamento del valore di loss e della dimensione del file .h5 che rappresenta la pensantezza della rete correlata al bit rate dell'autoencoder ad ogni client.

Per il calcolo della dimensione della rete, dopo il salvataggio del file .h5, si è effettuata una compressione del modello in formato .zip e si è calcolata la pesantezza del file compresso.

```
1 autoencoder.save('/content/Conv_autoencoder_'+'.h5')
2 zip = ZipFile('/content/Conv_autoencoder_'+'.zip','w')
3 zip.write('/content/Conv_autoencoder_'+'.h5')
4 zip.close()
```

Nella figura 14 si possono vedere i risultati di tre tipologie di rete neurale, la prima (light) costituita da layer con 8-16-32 nodi, la seconda (medium) da 16-32-64 e la terza (heavy) da 32-64-128 nodi ognuna delle quali è stata testata su un set di campioni via via incrementale con 50 epoch e batch da 128 campioni.

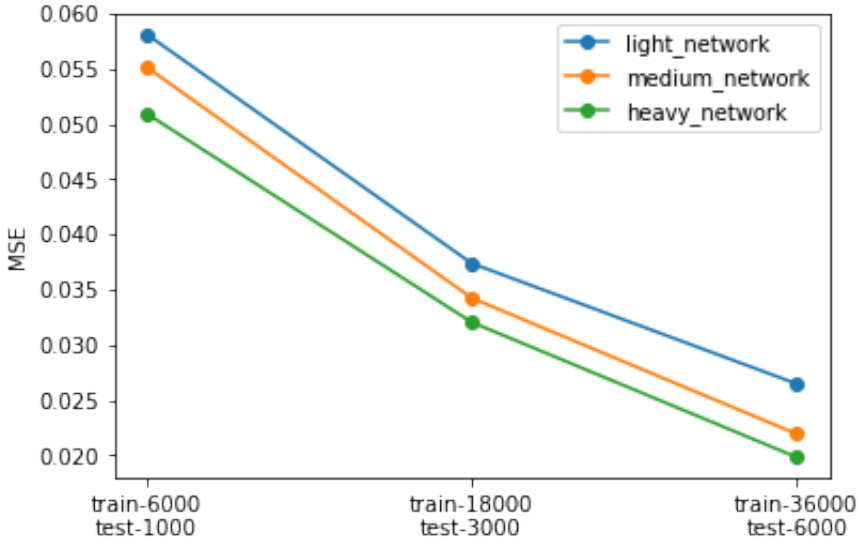


Figure 14: Nine configuration Convolutional Autoencoder

Si può notare come il valore di perdita diminuisca notevolmente all'aumentare della dimensione del dataset mentre non si identificano cambiamenti sostanziali cambiando la dimensione della rete convoluzionale.
Per quanto riguarda la pesantezza della rete in termini di bit memoria si è verificato un aumento al crescere della dimensione delle hidden features ovvero dei nodi interni della rete. Il grafico di figura 15 illustra le 9 configurazioni risultanti.

Per la costruzione dell'autoencoder definitivo è stata scelta quindi la rete neurale composta da layer convoluzionali con 16, 32 e 64 nodi, la quale garantisce un giusto compromesso tra perdita e bit rate da trasmettere.
Al fine di costruire la struttura vera e propria del progetto di tesi si è proseguito con la divisione del dataset in tre parti uguali da fornire in input a tre autoencoder convoluzionali equivalenti.
I tre autoencoder, i quali, in ipotesi di applicazione reale, sono la rappresentazione di un insieme di client che partecipano ad un processo di federated learning, dopo aver allenato il

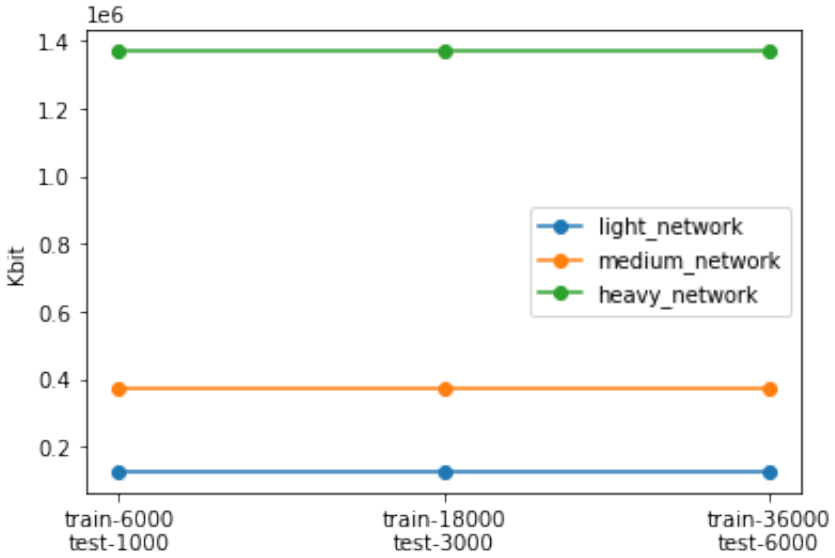


Figure 15: Bit rate dell'autoencoder in 9 configurazioni

modello in locale, inviano i pesi della rete ad un quarto autoencoder che effettua la media dei pesi dei tre modelli ottenendo un modello globale.

I modelli effettuano il processo di apprendimento con MEAN SQUARED ERROR come funzione di loss e funzione di ottimizzazione Stochastic Gradient Descent(SGD).

Il calcolo matematico teorico alla base di questa operazione è il seguente

$$w_k^{global} = \frac{1}{N} \sum_{i=1}^{N} w_k^{local_i} \quad \forall k = 1, \ldots, K \tag{28}$$

ove K rappresenta il numero di parametri della rete.

Dopo aver effettuato questa operazione l'autoencoder globale effettua, attraverso il metodo EVALUATE il calcolo dell'errore sui dati di test.

Abbiamo quindi visionato l'andamento dell'errore quadratico medio (MSE) e la dimensione della rete globale confrontata con le dimensioni delle reti locali. Si noti come l'errore dell'autoencoder globale sia più consistente degli MSE globali pur tuttavia avendo un andamento similare.

Ciò è coerente con il fatto che il modello centrale non effettua l'operazione di training poiché non dispone dei dati di allenamento ma esegue solo l'operazione di valutazione sui dati di test.

Ho verificato che la dimensione della rete relativa ai tre autoencoder, coincide come da aspettativa, mentre quella dell'autoencoder globale è leggermente minore probabilmente a causa

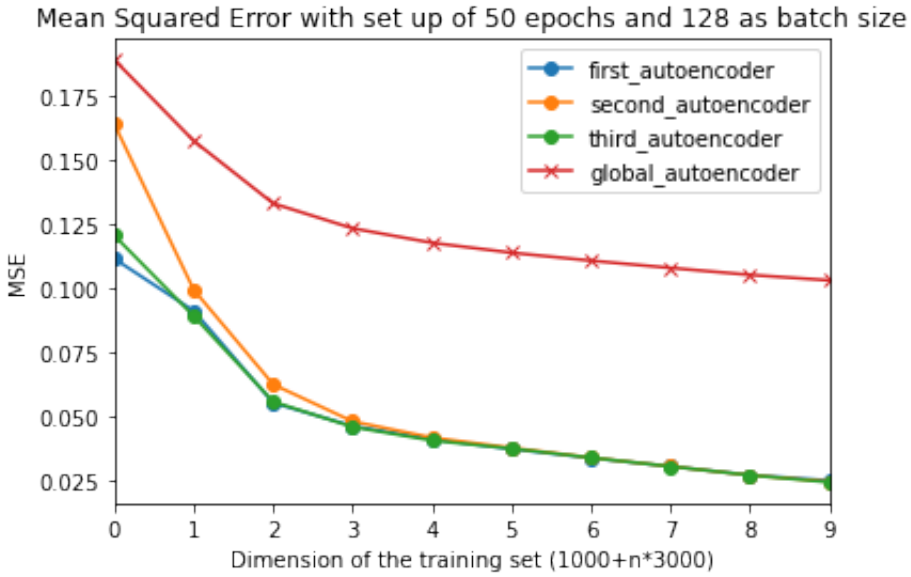


Figure 16: Mean Squared Error degli Autoencoder locali e del Globale

del fatto che esso non contiene le informazioni relative al processo di allenamento.

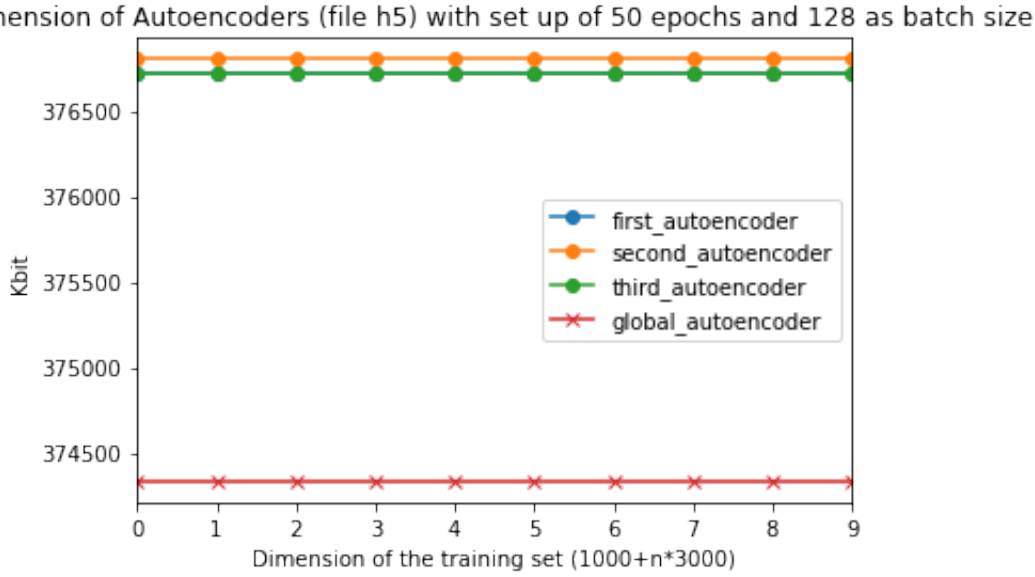


Figure 17: Dimensione della rete degli autoencoder e della rete dell'autoencoder globale

4.4 Classificazione di immagini MNIST

Per procedere con la seconda parte del progetto di tesi viene introdotto il metodo di classificazioni delle immagini MNIST.

Il dataset in questione mette a disposizione quattro array, il TRAIN_IMG e TRAIN_LABLES per il training mentre TEST_IMG e TEST_LABLES per il test.

Le immagini sono array di pixel 28x28 con ogni pixel che può assumere un valore da 0 a 255 per rappresentare una tonalità di grigio.

Mentre le etichette sono un array formato da numeri interi da 0 a 9 riferiti ad ogni singola immagine.

Prima di poter inviare i dati alla rete neurale è necessario normalizzarli ovvero ridimensionare il valore dei pixel in un intervallo da 0 a 1.

Si effettua la stessa operazione per i dati di training e i dati di test:

```
(train\_img, train\_lables),(test\_img, test\_lables) = mnist.load_data()
train\_img = train\_img.astype('float32') / 255
test\_img = test\_img.astype('float32') / 255
```

Si procede quindi alla costruzione del modello della rete neurale che sarà costituito da alcuni layer che estraggono la rappresentazione dei dati inseriti in essi.

Il Dense layer, come la stragrande maggioranza dei layer, possiede parametri che vengono appresi durante la fase di training.

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

Il Flatten layer riformatta le immagini portandole da una rappresentazione matriciale bidimensionale (28 x 28 pixel) ad una rappresentazione unidimensionale di 28 x 28 = 784 pixel a differenza del progetto di tesi dove abbiamo una matrice bidimensionale 4 x 4 poiché l'input utilizzato è l'output del codificatore.

Dopo aver appiattito l'input, si trovano due layer Dense in sequenza, cioè strati neurali completamente connessi.

Il primo con 128 nodi mentre il secondo, di 10 neuroni, restituisce un array logit di lunghezza 10 dove ogni nodo contiene il punteggio che indica a quale classe l'immagine di input appar-

tiene.

In seguito per procedere alla classificazione si effettua la compilazione del modello scegliendo la funzione di perdita e l'ottimizzatore, poi la fase di training e infine la valutazione dell'accuratezza.

4.5 L'autoencoder-classificatore

Per la creazione vera e propria della struttura del progetto di tesi sono partito dal considerare i tre autoencoder, dove, in corrispondenza del flusso di bit centrale ovvero dei pesi che i singoli modelli locali devono inviare, è stata aggiunta la struttura di un classificatore di immagini.
Ogni client ha al suo interno una rete neurale di un encoder e un decoder, i quali vanno a costituire l'autoencoder, e un classificatore che ha input in corrispondenza dell'output dell'encoder.
L'autoencoder-classificatore quindi in sostanza svolge sia la funzione di autoencoder ovvero può ricostruire immagini di input nell'output e sia la funzione di classificatore di immagini.
Permette infatti di ricavare il modello globale di un autoencoder da trasmettere ai client dell'ambiente federato e una volta che questi ultimi inviano le hidden features delle immagini è in grado di ricostruirle attraverso il decodificatore che è presente nella struttura.
Il classificatore è costituito da un Flatten layer che ha come input l'uscita del codificatore e due layer totalmente connessi (Dense) rispettivamente formati da 128 e 10 nodi.
La dimensione del primo Dense layer ha permesso risultati di accuratezza di classificazione accettabili e quindi non si è appesantita la rete aumentando il numero di nodi del layer.
Il secondo dense layer è necessario abbia 10 nodi poiché il dataset MNIST contiene dieci classi di immagini mappabili in output.

```
...
#CLASSIFICATOR LAYERS
  k = layers.Flatten(input_shape = (4,4))(encoded)
  k = layers.Dense(128, activation= 'relu', name = 'AuClass_DenseLayer')(k
   )
  output_dense = layers.Dense(10,activation = 'softmax',name ='out_dense')
   (k)
  return tf.keras.Model(inputs= input_img, outputs = output_dense)
```

La struttura semplificata dell'autoencoder-classificatore integrati in un ambiente di apprendimento federato viene rappresentata nella figura seguente.

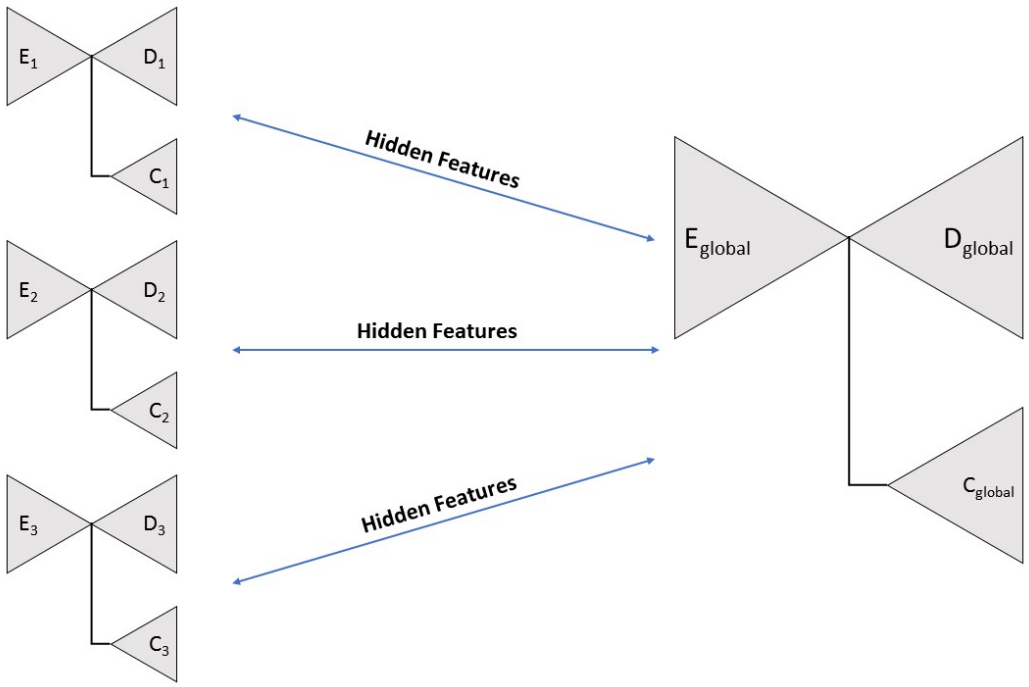


Figure 18: Schema con Autoencoder -Classificatori in un ambiente Federato

L'autoencoder-classificatore globale presenta quindi un decodificatore e un classificatore con la struttura identica a quella della rete dei client.

Gli autoencoder-classificatori possono trasmettere al modello globale non solo i pesi della rete dell'autoencoder locale utili alla ricostruzione delle immagini ma anche le hidden features relative alla struttura di classificazione.

Questo modello di rete globale utilizza una strategia di ponderazione dei pesi basata sull'inverso della perdita ottenuta da ogni modello locale e privilegiando i client con perdita minore ovvero

$$w_k^{global} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{loss_i} (w_k^{local_i}) \quad \forall k = 1, \ldots, K \tag{29}$$

con N numero di client e K numero di pesi della rete.

L'autoencoder classificatore è quindi in grado di ottenere un modello globale ponderato avendo ottenuto in ingresso i pesi degli autoencoder-classificatori locali che, dopo aver effettuato il processo di training, mettono a disposizione l'informazione appresa.

In un contesto reale di apprendimento federato però, il server non possiede le immagini originali per poter allenare il modello e quindi si vuole cercare una via, sfruttando le hidden feature dell'autoencoder, per trasferire la conoscenza appresa ad un modello globale, cosi che esso possa decodificare l'informazione, inviata sottoforma di flusso di bit, e ottenere una

rappresentazione piuttosto precisa dei dati grezzi originali.

Questa operazione prende il nome di transfer learning ovvero le informazioni di una rete pre-addestrata su un set di dati di grandi dimensioni, vengono usate in un'altra rete senza dover riallenare il modello.

L'obiettivo della tesi consiste quindi nel comprendere se, effettuando un fine tuning al classificatore globale, quest'ultimo sia in grado di ottenere una classificazione più accurata di immagini specifiche.

Al lato server si trovano un codificatore, un decodificatore e un classificatore poiché le immagini vengono inviate sotto forma di flusso di bit ovvero l'informazione che arriva al decodificatore sono le hidden feature dei modelli locali allenati, che dopo la decodificazione, permettono di avere le immagini ricostruite da poter inserire in input al classificatore globale.

Si vuole appunto capire se ricostruendo le immagini a lato server, si ottiene una classificazione più accurata rispetto alla soluzione in cui vengono raccolti i pesi dei classificatori locali per la creazione del classificatore globale.

5 Risultati e valutazioni

Per analizzare l'accuratezza e l'efficienza del metodo proposto si è utilizzato il dataset MNIST.
E' stata effettuata una operazione di shuffling sui campioni iniziali per migliorare le performance di previsione e la qualità del modello ed avere quindi una situazione di dati indipendenti e identicamente distrbuiti (i.i.d.).
Si è quindi proceduto alle analisi preliminari descritte in precedenza al fine di determinare la struttura dell'autoencoder.
Sono quindi stati creati tre autoencoder identici e sono stati allenati suddividendo i dati di input in tre sottoinsiemi diversi e poi dandoli in ingresso ai tre modelli.
Per l'allenamento sugli autoencoder è stato usato l'ottimizzatore a Stochastic Gradient Descent (SGD) con errore quadratico medio (MSE) come loss function.

```
1 autoencoder.compile(optimizer='SGD', loss='mean_squared_error')
```

E' stato quindi creato l'autoencoder globale semplicemente effettuando la media dei pesi dei tre autoencoder ovvero

$$w_k^{global} = \frac{1}{3}\sum_{i=1}^{3} w_k^{local} \qquad \forall k = 1, \ldots, K$$

con K il numero di nodi della rete. Questo autoencoder sarà la base su cui costruire gli autoencoder-classificatori che sono stati creati trasferendo i pesi del modello di autoencoder globale in un nuovo modello in cui è stata aggiunta la parte del classificatore.
Si è dovuta creare una struttura più semplice rispetto a quella richiesta in un contesto reale a causa di alcune limitazioni degli strumenti di sviluppo.
Dopo le operazioni preliminare per la costruzione della struttura autoencoder, il dataset di training è stato diviso in tre sottoinsiemi della stessa dimensione da fornire in ingresso ai tre autoencoder-classificatori.
Per quanto riguarda l'analisi dell'autoencoder-classificatore locale e globale sono stati valutati il loss, l'accuratezza e la dimensione del modello.
L'autoencoder classificatore locale è stato creato attraverso una tecnica di transfer learning in cui i pesi del modello globale sono stati copiati in questa nuova struttura a cui poi sono stati aggiunti i layer di classificazione.
Nell'allenamento dell'autoencoder-classificatore sono stati bloccati (non allenabili) i pesi della rete autoencoder e impostati a "trainable" solo i pesi dei layer costituenti la struttura del

classificatore ovvero il Flatten layer e i due Dense layer.

É infatti necessario che venga allenata la parte della rete relativa alla classificazione poiché i pesi della parte autoencoder, qualora riallenati, andrebbero a compromettere il lavoro fatto sulla costruzione di un autoencoder efficiente che deve essere disponibile al lato server necessario a ricostruire adeguatamente le immagini per l'operazione di Fine Tuning.

É stato scelto l'ottimizzatore Adam con un tasso di apprendimento pari a 0.001 e funzione di loss SparseCategoricalCrossEntropy che calcola la crossentropia tra i valori delle etichette e i valori stimati.

Nel machine learning, la funzione di loss crossentropy viene usata di solito quando sono presenti due o più classi di etichette.

```
model.compile(optimizer= tf.keras.optimizers.Adam(learning_rate=0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])
```

I tre autoencoder-classificatori sono stati prima allenati sui tre dataset diversi e poi valutati sullo stesso insieme di dati di test.

Il modello dell'autoencoder-classificatore globale è stato creato con la stessa struttura dei modelli locali ed effettuando la media ponderata dei pesi secondo l'equazione 29.

Dopo l'inserimento dei pesi nel modello globale viene fatta la procedura di validazione per mezzo del metodo evaluate e si considerano il loss e l'accuratezza come metriche di valutazione del modello.

```
loss_global_NO_tuning, accuracy_global_NO_tuning =
    auto_glob_class_NO_finetuning.evaluate(new_x_test,new_y_test,
    batch_size=batch_size, verbose=0)
```

Nelle figure 19, 20 e 21 vengono riportati i grafici dove nell'asse delle ascisse si trova la formula di incremento dei campioni usati nell'addestramento mentre nell'asse delle ascisse troviamo rispettivamente per i tre grafici, l'accuratezza, il loss e la dimensione del modello.

Sono stati messi a confronto i tre autoencoder-classificatori con l'autoencoder-classificatore globale rappresentato dalla linea rossa.

Si può notare come l'accuratezza e la perdita del modello globale ottengano lo stesso andamento dei modelli locali ma con prestazioni minori.

L'incremento significativo sia per quanto riguarda l'accuratezza che per la perdita si ha con circa 10000 istanze usate nell'apprendimento.

Tuttavia, si nota una tendenza ad overfitting del modello globale sopra i 15000 campioni.
Come si può vedere la dimensione della rete neurale degli autoencoder classificatori rimane costante al cambiare del numero delle istanze usate nell'apprendimento mentre si nota una differenza sostanziale tra la dimensione delle reti locali e la rete globale.
Ciò potrebbe esser dovuto al fatto che il modello globale non è interessato dalla fase di training.

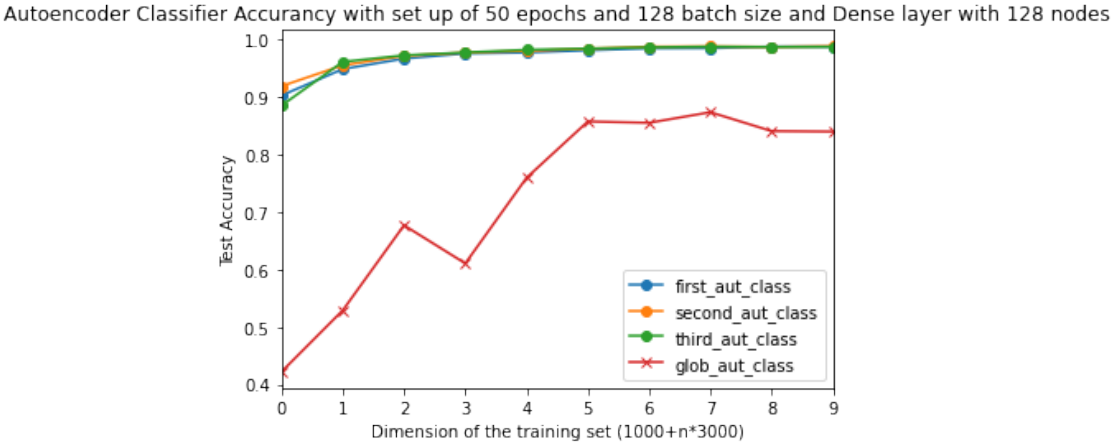


Figure 19: Accuratezza degli Autoencoder-Classificatori locali e del modello globale

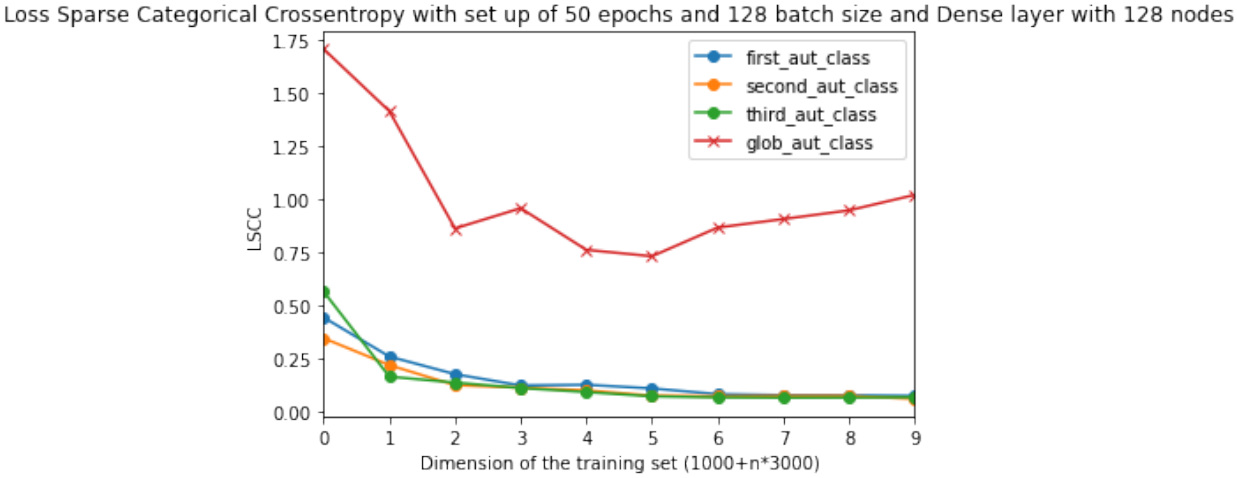


Figure 20: Perdita degli Autoencoder-Classificatori locali e del modello globale

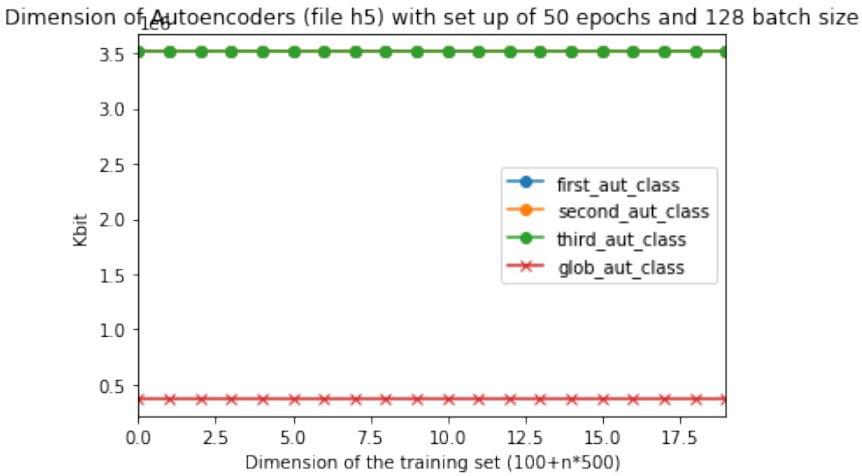


Figure 21: Dimensione della rete degli Autoencoder-Classificatori locali e del modello globale

5.1 Fine tuning

L'ultima parte del lavoro di tesi ha riguardato un processo di transfer learning e di messa a punto.

Il transfer learning consiste nel trasferimento di conoscenza appresa da un problema precedentemente trattato ad un altro problema.

La rappresentazione più usuale del transfer learning consiste nel:

- Prendere i layer da un modello precedentemente addestrato
- Bloccarli per evitare di perdere l'informazione appresa nei successivi round di addestramento
- Aggiungere alcuni layer allenabili oltre a quelli congelati
- Addestrare i nuovi layer su un set di dati

Nel nostro caso, è servito a trasferire il modello di autoencoder globale all'autoencoder-classificatore così che egli, senza aver accesso ai dati grezzi, possa ricostruire l'input originale.

La fase successiva consiste nel fine tuning ovvero utilizzare i dati grezzi ricostruiti per mettere a punto il modello di classificazioni globale.

E' stato quindi costruito un nuovo dataset a partire dal dataset di partenza per poter essere utilizzato nell'allenamento del classificatore globale.

Per semplicità di calcolo, le immagini di fine tuning sono state ricostruite dai singoli autoencoder locali.

Si è quindi provveduto a ricostruire le immagini utilizzando come input il dataset X_TEST e la funzione di previsione del modello allenato degli autoencoder.

```
decoded_imgs_0 = autoencoder_0.predict(x_test)
decoded_imgs_1 = autoencoder_1.predict(x_test)
decoded_imgs_2 = autoencoder_2.predict(x_test)
```

I tre sottoinsiemi sono poi stati uniti a formare un unico dataset su cui è stata effettuata una operazione di shuffling dei dati.

Questo nuovo dataset è appunto il dataset usato per la fase di fine tuning ovvero va a costituire un nuovo insieme di immagini che verranno date in input al classificatore globale per osservare se è possibile un miglioramento delle metriche di valutazione del modello.

Si è usata questa strategia per semplicità anche se nell'ambiente reale i client trasmettono solo la rappresentazione compressa delle immagini che dovrà poi essere decodificata dall'encoder presente al lato server.

Mentre prima quindi si effettuava solo l'operazione di validazione dei dati avendo a disposizione solo i pesi della rete, ora viene aggiunta nella computazione del classificatore globale l'operazione di training sul nuovo dataset.

Ci si aspetta infatti che le prestazioni di accuratezza e perdita migliorino rispetto alla soluzione precedentemente adottata.

Vengono riportati i grafici in figura 22 e 23.

Dall'analisi si può notare un miglioramento sostanziale dell'accuratezza e della perdita del modello globale con fine tuning rispetto alla soluzione priva di fine tuning soprattutto con pochi campioni usati.

Si nota inoltre un accenno di generalizzazione del modello a partire da 22000 campioni utilizzati.

Figure 22: Accuratezza degli Autoencoder Classificatori locali e del globale con fine tuning

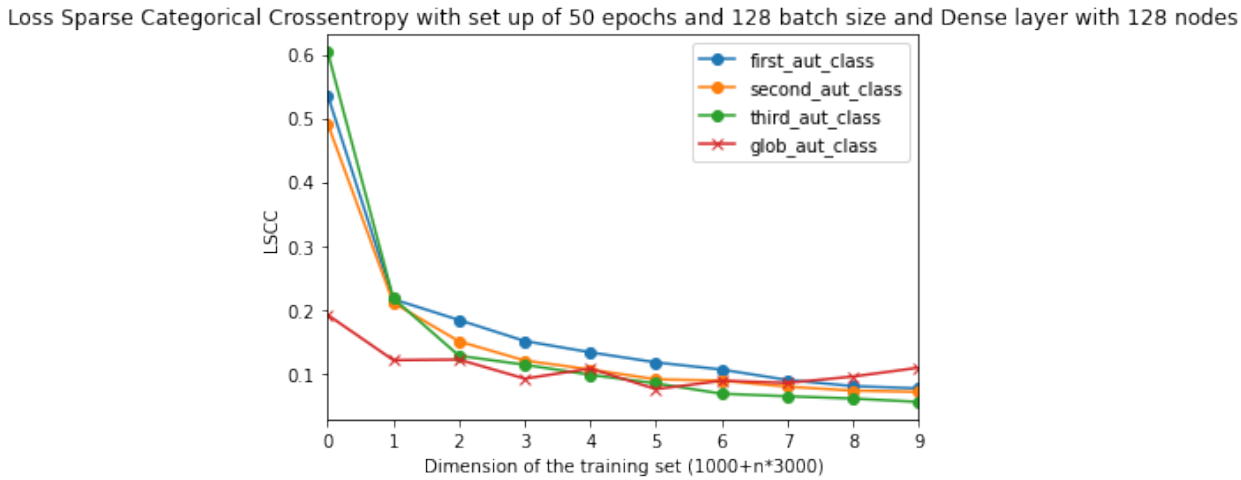


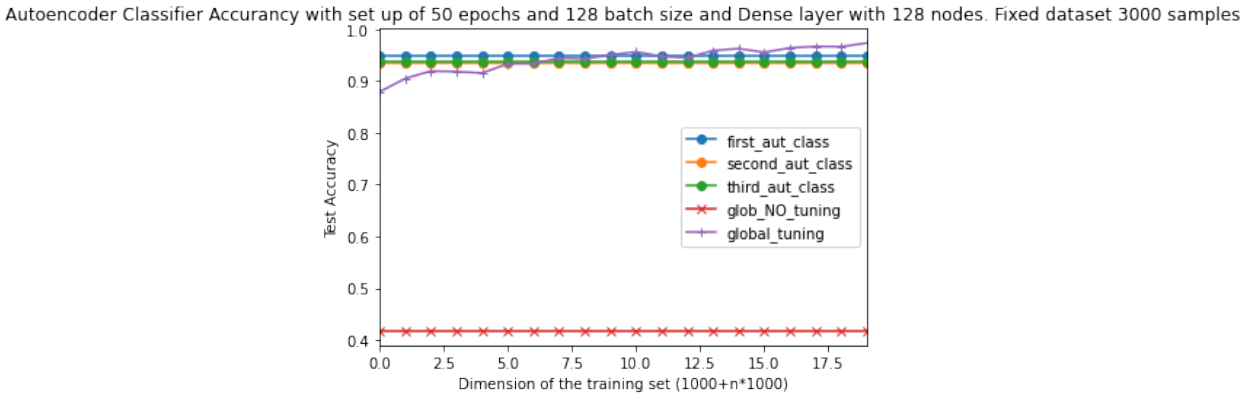
Figure 23: Perdita degli autoencoder-classificatori locali e del globale con fine tuning

Per quanto concerne la quantità di informazione (bit rate) che deve essere trasmessa in rete dai client verso il server si calcola con la seguente espressione:

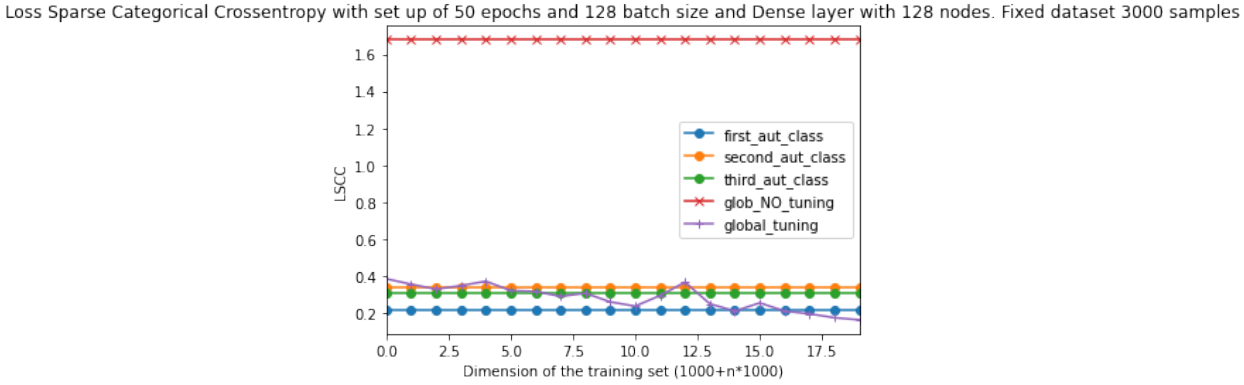
$$N \times K_{hiddenlayer} \times 4bytes + zip(autoencoder.h5) \tag{30}$$

Ovvero il numero di immagini trasmesse per il numero di nodi del layer centrale della rete neurale moltiplicato per la dimensione di ogni nodo della rete (4 byte) sommato al modello in file .h5 dell'autoencoder inviato con compressione zip.
Il server utilizza questo flusso di dati parallelamente per ricostruire le immagini mantenendo la riservatezza dei dati e allo stesso tempo calcolare il modello globale dell'autoencoder.

Infine, ho posto a confronto il loss e l'accuratezza dei modelli locali, del globale senza fine tuning e del globale con fine tuning.
Questo per poter vedere l'effetto migliorativo apportato dal fine tuning al variare del numero di immagini ricostruite usate ai fini dell'apprendimento del server centrale.



(a) *Confronto Accuratezza degli Autoencoder Classificatori locali e del globale con dati di tuning variabili.*



(b) *Confronto Perdita degli autoencoder-classificatori locali e del globale con fine tuning variabile.*

Figure 24: Statistiche Autoencoder Classificatori con Fine Tuning

Per questa analisi sono stati fissati 3000 campioni di allenamento per le reti neurali locali e per il modello globale senza tuning.
Si nota ampiamente il notevole miglioramento che porta la fase di training nel modello globale il quale riesce ad ottenere prestazioni di accuratezza e perdita migliori comparate con i modelli locali a partire da circa 10000 di campioni di training.
Per quanto riguarda questa configurazione il bit rate per trasferire l'informazione appresa all'unità centrale sarà di $3000 \times 16 \times 4 + 367KB \simeq 560KB$ dove 3000 sono le immagini che

ogni client deve inviare al server, 16 sono i nodi dell'hidden layer, 4 sono i byte che occupa ogni peso di una rete neurale e $367KB$ è la pesantezza del modello locale il quale serve per poter costruire il modello globale.

5.2 Problematiche riscontrate

I problemi riscontrati durante il lavoro di tesi riguardano principalmente l'ambiente di lavoro e l'uso della libreria keras.

La piattaforma di *cloud computing* Google Colab non mi ha permesso sempre un utilizzo completo delle prestazioni migliori offerte dagli elaboratori grafici messi a disposizione poiché si sono verificati spesso fenomeni di congestione della rete rallentando notevolmente il calcolo di reti neurali consistenti.

É stata inoltre complessa la gestione dei pesi delle reti neurali dovuto alla complessità delle strutture dati usate da tensorflow quali array e matrici.

Piuttosto che creare strutture troppo complesse infatti si è cercato un approccio più semplice trattando modelli di reti neurali molto simili al fine di una gestione migliore dei parametri presenti ad ogni layer.

Un'altra problematica ha riguardato la fusione dei pesi della rete poiché si è visto che la strategia iniziale di effettuare una fusione attraverso la media aritmetica non portava grandi risultati ed è stata adottata una modalità differente.

Inoltre, a causa dei limiti di alcune librerie non si è potuto creare una rappresentazione fedele alla realtà dell'autoencoder classificatore poiché non è stato possibile trattare una rete ramificata con due output differenti, uno per l'autoencoder e uno per il classificatore.

Si è quindi adottata anche qui una strategia semplificata riducendo la rete per permetterne l'allenamento.

Tutte le analisi sono state effettuate in un contesto di dati indipendenti e identicamente distribuiti anche se è noto che un ambiente IoT è costituito usualmente da dati non-i.i.d e quindi il lavoro svolto potrebbe non rappresentare al meglio un ambiente di implementazione realistico.

6 Conclusioni

Il mondo degli oggetti interconnessi cambierà radicalmente l'approccio alla vita quotidiana e condizionerà le abitudini degli individui in modo radicale.

In una simbiosi totale tra uomo e macchina dove miliardi di oggetti potranno trasmettere dati nella rete è necessario uno sforzo sostanziale per limitare questo nuovo paradigma tecnologico alla diffusione di contenuti sensibili che possano compromettere la privacy degli utilizzatori.

Sebbene infatti il progresso tecnologico proceda a ritmi serrati, il resto degli ecosistemi con cui si integrerà, le norme giuridiche che lo regoleranno e gli aspetti di sicurezza e protezione, non sempre saranno al passo con il cambiamento.

In un futuro dove i dati digitali condivideranno enormi quantità di informazioni saranno necessari accorgimenti utili alla gestione e protezione degli stessi.

Ecco che le tecniche di *Federated Learning* sono fondamentali per apportare un aiuto importante alla tutela dei consumatori mantenendo intatta la privacy e la sicurezza e fornendo allo stesso tempo capacità di elaborazione dati notevoli.

Come in passato con la condivisione del sapere che portò a grandi evoluzioni cosi anche oggi l'apprendimento digitale federato, controllato dall'uomo, e usato principalmente con senso etico, potrebbe rivoluzionare radicalmente la qualità di vita delle generazioni future.

I risultati ottenuti in questo studio di tesi hanno dimostrato la possibilità di incrementare l'ottimizzazione del sistema federato per quanto concerne sistemi che fanno uso di immagini e/o video sequenze.

L'aver consentito al sistema di elaborazione centrale di effettuare un raffinamento attraverso l'uso di una rappresentazione fedele ai dati grezzi ha portato i vantaggi sperati in termini di accuratezza e loss function.

Rimangono da considerare le analisi per valutare il tradeoff ottimale tra i costi di comunicazione e l'efficacia del modello proposto.

La dimensione della rete autoencoder per la ricostruzione dei dati di input deve essere calibrata in maniera adeguata al fine di non compromettere la pesantezza della struttura federated.

Infatti, minore è la complessità del modello neurale minori saranno i costi di trasmissione e potranno essere soddisfatti con maggior probabilità i limiti eterogenei dei dispositivi partecipanti al processo.

Rimane inoltre aperta l'analisi di una situazione realistica con un sistema eterogeneo e dati non-i.i.d. poiché la soluzione proposta non si è spinta a questa valutazione.
Ciò lascia spazio a studi futuri mirati ad ottenere un fine tuning più accurato e contemporaneamente un adattamento efficiente a sistemi eterogenei.

7 Ringraziamenti

Desidero innanzitutto ringraziare il mio Relatore Professor Simone Milani non solo per il progetto di tesi ma per aver anteposto il rapporto umano di cordialità e disponibilità allo studio realizzato nonostante la lontananza fisica. Voglio altresì ringraziare tutti coloro che in questo lungo e travagliato percorso di studi hanno manifestato verso me la loro fiducia e amicizia vera. Un grazie va a tutti i colleghi incontrati nei vari anni al Dipartimento di Ingegneria, in particolar modo a Elisa, Jacopo, Pietro, Roberto, Michele e Alessandro, ai veri Professori che mi hanno trasmesso passione per gli argomenti trattati, sempre disponibili e capaci ad ascoltare. Voglio poi rendere grazie a tutte le persone che in questi anni della mia vita di studio hanno saputo dimostrarmi affetto e vicinanza spronandomi a non mollare, i miei amici, i colleghi educatori di Este e il gruppo della Pastorale Giovanile di Padova, Paolo, Giorgio, Maurizio S. e gli amici del gruppo Alter-Ego. Posso ora finalmente esclamare di essere arrivato fino a qui con le mie forze, certo, affrontando momenti di delusione e sconforto, ma senza mai darmi per vinto. Per ultima ma non per questo meno importante ringrazio dal profondo del cuore la mia Famiglia, zii e cugini che mi hanno sempre dimostrato vicinanza e mi hanno sostenuto con amore nell'affrontare in serenità questo percorso. Pensando al primo anno di smarrimento totale durante il quale non passai nemmeno un esame non mi sembra possibile di essere arrivato a questo obiettivo. Tutto questo è stato possibile solo grazie a tutti loro, e soprattutto per volontà di Dio che mi ha tenuto la mano nel mio punto zero, dove tutto sembrava perduto mi ha sollevato dalla polvere e mi ha accompagnato fino a questo traguardo.

References

[1] *Measuring the Information Society Report*. International Telecommunication Union, 2015.

[2] Regolamento (ue) 2016/679 del parlamento europeo e del consiglio, 2016.

[3] Proposal for a regulation of the european parliament and of the council concerning the respect for private life and the protection of personal data in electronic communications and repealing directive 2002/58/ec (regulation on privacy and electronic communications), 2021.

[4] GSM Association. The gsma guide to the internet of things, 2020.

[5] Richard Chow. The last mile for iot privacy. *IEEE Security & Privacy*, 15(6):73–76, 2017.

[6] Luca Corinzia, Ami Beuret, and Joachim M. Buhmann. Variational federated multi-task learning, 2021.

[7] IBM Corporation. Cost of a data breach report 2020. 2020.

[8] Roger T. Johnson David W. Johnson. Cooperative learning, 2017.

[9] Yasir Al Denis Kozlov, Jari Veijalainen. Security and privacy threats in iot architectures. In *Proceedings of the 7th International Conference on Body Area Networks*, pages 256–262, 2012.

[10] Ayisha Manzoo Eman Shaikh, Iman Mohiuddin. Internet of things (iot): Security andprivacythreats. In *International Conference on Computing, Communication, and Intelligent Systems(ICCCIS)*, 2019.

[11] Dr. Margarita Esponda. Trends in hardware architecture for mobile devices. Technical report, Freie Universität Berlin, 2004.

[12] Klaus-Robert Müller Felix Sattler, Simon Wiedemann and Wojciech Samek. Robust and communication efficient federated learning from non iid data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, 2020.

[13] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics, 2015.

[14] Amato G., Carrara F., Falchi F., Gennaro C., Meghini C., and Vairo C. Deep learning for decentralized parking lot occupancy detection. *Expert systems with applications*, 72:327–334, 2017.

[15] He Gang, Wu Chang, Li Lei, Zhou Jinjia, Wang Xianglin, Zheng Yunfei, Yu Bing, and Xie Weiying. A video compression framework using an overfitted restoration neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[16] Dashan Gao, Ce Ju, Xiguang Wei, Yang Liu, Tianjian Chen, and Qiang Yang. Hhhfl: Hierarchical heterogeneous horizontal federated learning for electroencephalography, 2020.

[17] Richard Harper. People versus information: the evolution of mobile technology. In *Human-Computer Interaction with Mobile Devices and Services*, pages 1–14, 2003.

[18] Youbiao He, Gihan J. Mendis, and Jin Wei. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. *IEEE Transactions on Smart Grid*, 8(5):2505–2516, 2017.

[19] Wei Huang, Tianrui Li, Dexian Wang, Shengdong Du, and Junbo Zhang. Fairness and accuracy in federated learning. 2020.

[20] D. W. Johnson, Geoffrey M. Maruyama, R. Johnson, D. Nelson, and L. Skon. The effects of cooperative, competitive, and individualistic goal structures on achievement: a meta-analysis. *Psychological Bulletin*, 89:47–62, 1981.

[21] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, 2018.

[22] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[23] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *CoRR*, abs/1908.07873, 2019.

[24] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[25] Xinyu Li, Yanyi Zhang, Ivan Marsic, Aleksandra Sarcevic, and Randall S. Burd. Deep learning for rfid-based activity recognition. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, page 164–175, New York, NY, USA, 2016. Association for Computing Machinery.

[26] Tao Lin. Deep learning for iot. *CoRR*, abs/2104.05569, 2021.

[27] Tao Lin and J Gao. Graphic user interface testing based on petri net. *Application Research of Computers*, 33(3):768–772, 2016.

[28] Ronald Lippitt. Kurt lewin, 1890-1947. adventures in the exploration of interdependence. *Sociometry*, 10(1):87–97, 1947.

[29] Zhenguang Liu, Luming Zhang, Qi Liu, Yifang Yin, Li Cheng, and Roger Zimmermann. Fusion of magnetic and visual sensors for indoor localization: Infrastructure-free and more effective. *IEEE Transactions on Multimedia*, 19(4):874–888, 2017.

[30] Wenhuan Lu, Ju Zhang, Xinli Zhao, Jianrong Wang, and Jianwu Dang. Multimodal sensory fusion for soccer robot self-localization based on long short-term memory recurrent neural network. *Journal of Ambient Intelligence and Humanized Computing*, 8:885–893, 11 2017.

[31] Riccardo Mazza Luca Mazzola. Gvis: an integrating infrastructure for adaptively mashing up user data from different sources. In *14th International Conference Information Visualisation*, 2010.

[32] Denise Lund, Vernon Turner, Carrie MacGillivray, and Mario Morales. Worldwide and regional internet of things(iot) 2014–2020 forecast: A virtuous circle of proven value and demand. Technical report, IDC, 2014.

[33] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *CoRR*, abs/1602.05629, 2016.

[34] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018.

[35] Gordon E. Moore. Cramming more components onto integrated circuits. *IEEE Solid-State Circuits Society Newsletter*, 11, 2006.

[36] Solmaz Niknam, Harpreet S. Dhillon, and Jeffery H. Reed. Federated learning for wireless communications: Motivation, opportunities and challenges, 2020.

[37] Lionel Pigou, Aäron van den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video, 2016.

[38] Monika Roopak, Gui Yun Tian, and Jonathon Chambers. Deep learning models for cyber security in iot networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0452–0457, 2019.

[39] Totten S., Sills T., Digby A., and Russ P. *Cooperative Learning: A guide to research*. Garland bibliographies in contemporary education, 12.; Garland reference library of social science, 674. Clarendon Press, 1991.

[40] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018.

[41] Mofreh M. Salem. A (fm/drdpe)based approach to improve federated learning optimizer. In *2007 International Conference on Computer Engineering & Systems*, 2007.

[42] Carlo Ratti Shawn DuBravac. The internet of things: Evolution or revolution, 2015. aig.co.uk.

[43] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery.

[44] Mingcong Song, Kan Zhong, Jiaqi Zhang, Yang Hu, Duo Liu, Weigong Zhang, Jing Wang, and Tao Li. In-situ ai: Towards autonomous and incremental deep learning for

iot systems. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 92–103, 2018.

[45] Sharil Tumi Sylvia Encheva. Cooperative learning objects. In *Proceedings of the IADIS International Conference on e-Society*, pages pp 76–82, 2008.

[46] tensorflow.org. Transfer learning and fine-tuning.

[47] Ratik Tiwari, Nikhil Sharma, Ila Kaushik, Archit Tiwari, and Bharat Bhushan. Evolution of iot amp; data analytics using deep learning. In *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 418–423, 2019.

[48] International Telecommunication Union. *Measuring the Information Society Report*, chapter 5. United Nations Publications, 2015.

[49] Jianxin Wang, Ming K. Lim, Chao Wang, and Ming-Lang Tseng. The evolution of the internet ofthings (iot) over the past 20 years. *Computers & Industrial Engineering*, 155, 2021.

[50] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. 2018.

[51] Yousaf Bin Zikria, Muhammad Khalil Afzal, Sung Won Kim, Andrea Marin, and Mohsen Guizani. Deep learning for intelligent iot: Opportunities, challenges and solutions. *Computer Communications*, 164:50–53, 2020.

"Sancte Michael Archangele, defende nos in proelio; contra nequitiam et insidias diaboli esto praesidium. Imperet illi Deus, supplices deprecamur: tuque, Princeps militiae caelestis, Satanam aliosque spiritus malignos, qui ad perditionem animarum pervagantur in mundo, divina virtute, in infernum detrude."

"Poi vidi in cielo un altro prodigio, grande e meraviglioso: sette Angeli colle sette piaghe ultime perchè con esse ha da compirsi l' ira di Dio. E vidi come un mare di vetro misto di fuoco, e sul mare di vetro stavan quelli che avevan vinta la bestia e la sua immagine e il numero del suo nome."