POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettronica

Tesi di Laurea Magistrale

Analisi di segnali a microonde: tecniche di Machine Learning per la rilevazione dei contaminanti alimentari

Relatori
prof. Mario Roberto CASU
dott. Luca URBINATI

Candidato
Lorenzo MANGANI

Anno Accademico 2020-2021

Alla mia famiglia

Sommario

La presenza di corpi estranei all'interno dei prodotti alimentari gioca un ruolo fondamentale nel mercato moderno e può determinare il successo o il fallimento di qualsiasi azienda o campagna commerciale.
Tra tutte le possibili strategie di prevenzione sviluppate nel corso degli anni, una valida soluzione al problema della contaminazione è quella del *sensing a microonde*, tecnica particolarmente utile nel settore alimentare per la sua caratteristica di essere non distruttiva e facile da applicare in real-time.

La tesi in esame affronta e sviluppa il progetto di un dispositivo a microonde in grado di rilevare gli agenti contaminanti nei cibi in barattolo.
Il lavoro si inserisce all'interno del progetto *MIT-Food* e ne eredita i risultati con l'obiettivo di cercare soluzioni alternative per **semplificare il circuito** ed **abbattere i costi** di fabbricazione.
L'ipotesi alla base di questa tesi prevede di ridurre la quantità di informazioni necessarie al dispositivo per rilevare i contaminanti: mentre il prototipo originale utilizza parte reale e parte immaginaria dei segnali a microonde acquisiti dai barattoli in movimento sul nastro trasportatore, si vuole raggiungere le stesse prestazioni lavorando con il solo modulo dei parametri di diffusione.

Il progetto, suddiviso in tre parti principali, ha previsto lo svolgimento di attività diverse, quali:

1. lo **studio teorico** delle tecniche e degli strumenti utilizzati;
2. i processi di allenamento e verifica delle **reti neurali**;
3. il **progetto del circuito** di acquisizione del modulo dei segnali a microonde e la sua simulazione.

Le attività che hanno caratterizzato il corpo centrale della tesi sono quelle relative allo sviluppo e l'allenamento delle reti neurali, grazie alle quali è stato possibile analizzare il modulo dei segnali ricevuti ed arrivare alla validazione dell'ipotesi iniziale.

Principio di funzionamento del dispositivo

Alla base di qualsiasi operazione di imaging a microonde c'è l'utilizzo di un dispositivo in grado di trasmettere e ricevere dei segnali a Radio Frequenza (RF). Sfruttando la proprietà dei materiali di assorbire i raggi in modo diverso a seconda della propria densità, è possibile ricevere il segnale riflesso dal barattolo per verificare le sue condizioni: infatti, la presenza di eventuali contaminanti genera delle discontinuità nelle proprietà dielettriche del campione che possono essere rilevate.

I componenti necessari alla costruzione di un possibile nuovo dispositivo sono:

- **antenne** (6 nel caso del progetto in esame);
- **circuito di ricezione** per l'acquisizione dei segnali a microonde;
- **controllore** per la gestione delle antenne, il salvataggio dei dati e l'inferenza della rete neurale scelta.

Mentre nel prototipo iniziale, un *Vector Network Analyzer* (VNA) è utilizzato per connettere le antenne e ricevere i segnali dopo la trasmissione, con questa tesi sono state cercate soluzioni alternative per costruire un circuito più economico e specifico per l'operazione richiesta.

Risultati ottenuti con l'utilizzo di reti neurali

Le reti neurali sviluppate ed utilizzate per l'analisi dei segnali a microonde hanno l'obiettivo di determinare se il campione alimentare è sicuro o contaminato: la **classificazione binaria** di tutti i barattoli è automatizzata con l'utilizzo delle tecniche di Machine Learning ed i valori di accuratezza e precisione sono massimizzati al termine di ciascun processo di allenamento.

Per poter considerare valida l'ipotesi iniziale di poter discriminare i prodotti contaminati tramite il solo modulo dei parametri di diffusione, è necessario generalizzare i risultati con dati acquisiti in condizioni diverse.
In particolare, per costruire ciascun dataset, sono stati modificati parametri come:

- **frequenza** (tra 9 GHz e 11 GHz);
- **velocità del nastro** di confezionamento (30 cm/s o 50 cm/s);
- tipo di **contaminante** e posizione all'interno del barattolo.

Per rendere lo studio ancora più specifico e massimizzare le prestazioni, a fronte dei risultati ottenuti nel corso del lavoro, sono stati effettuati allenamenti:

1. in **singola frequenza** (ogni campione usato per l'allenamento aveva features alla sola frequenza di 10 GHz);

2. in **multi-frequenza** (ogni campione usato per l'allenamento aveva features appartenenti al range di frequenze 9-11 GHz).

I risultati hanno mostrato che, mentre con la singola frequenza le prestazioni dipendono in modo eccessivo dalle condizioni utilizzate per ciascuna misurazione, lavorando su un range di frequenza più ampio si ottengono ottimi risultati indipendentemente dal dataset considerato.

Progetto del circuito di acquisizione

Con la verifica dell'ipotesi iniziale, è stato possibile procedere con l'implementazione hardware di un nuovo prototipo a componenti discreti.
Il circuito è in grado di generare un segnale RF da trasmettere con una delle sei antenne e ricevere la potenza riflessa con tutte le altre.
I parametri di diffusione possono quindi essere calcolati come rapporto di potenze.

Dopo aver condotto una ricerca bibliografica e commerciale per definire i blocchi necessari alla costruzione del sistema, sono stati selezionati i componenti migliori per ciascuna operazione ed è stato simulato il comportamento finale del dispositivo. Confermando l'efficacia della combinazione microonde e Machine Learning, questa tesi ha permesso lo sviluppo di nuove strategie per il rilevamento dei contaminanti alimentari e ha messo le basi per la costruzione di nuovi prototipi industriali capaci di rendere i cibi che mangiamo sempre più sicuri.

Indice

I Introduzione **8**

1 Il problema della contaminazione dei cibi 9
1.1 Tecniche per la rilevazione dei contaminanti 10
1.1.1 Filtri alimentari 10
1.1.2 Raggi X 11
1.1.3 Metal Detectors 12
1.1.4 Raggi infrarossi 12
1.2 Microwave Imaging 12

II Introduzione al machine learning **14**

2 Machine Learning 15
2.1 Algoritmo generale e riduzione dell'errore 17
2.1.1 Regressione lineare 18
2.1.2 Riduzione dell'errore: Gradient descent 20
2.1.3 Generalizzazione 24
2.1.4 Regolarizzazione 27
2.2 Teoria dei classificatori 30
2.2.1 Regressione logistica 31
2.2.2 Alberi decisionali 32
2.2.3 Support Vector Machines (SVM) 34
2.3 Reti neurali 36
2.3.1 Modello matematico del neurone 36
2.3.2 Architettura 38
2.3.3 Algoritmo di allenamento 40

III Applicazione al caso di studio **43**

3 Presentazione del lavoro 44

3.1 Dispositivi usati per l'acquisizione dei dati 45
3.2 Tipi di contaminanti 49
3.3 Risultati precedenti ed obiettivi 51

4 ML in Python **53**
4.1 Principali librerie e pacchetti 53
4.1.1 Numpy 54
4.1.2 Pandas 55
4.1.3 Scikit-Learn 56
4.1.4 TensorFlow & Keras 57
4.2 Ambiente di sviluppo 58

5 Primo dataset **62**
5.1 Analisi del dataset 62
5.2 Primo modello 67
5.2.1 Ricerca bayesiana 67
5.2.2 Allenamento e validazione 68
5.2.3 Verifica 70
5.3 Secondo modello 73
5.3.1 Ricerca bayesiana 73
5.3.2 Allenamento e validazione 74
5.3.3 Verifica 76

6 Secondo dataset con scudo **79**
6.1 Analisi del dataset 79
6.2 Modello 82
6.2.1 Ricerca bayesiana 82
6.2.2 Allenamento e validazione 84
6.2.3 Verifica 85

7 Terzo dataset: FT System **90**
7.1 Velocità di 30 cm/s 91
7.1.1 Dataset 91
7.1.2 Ricerca bayesiana 92
7.1.3 Allenamento e validazione 93
7.1.4 Verifica 95
7.2 Velocità di 50 cm/s 98
7.2.1 Dataset 98
7.2.2 Ricerca bayesiana 99
7.2.3 Allenamento e validazione 101
7.2.4 Verifica 102

8 Analisi in multi-frequenza 106
8.1 Primo dataset 108
8.2 Secondo dataset con scudo 109
8.2.1 Ricerca bayesiana 110
8.2.2 Allenamento e validazione 111
8.2.3 Verifica 112
8.3 Terzo dataset: FT System 114
8.3.1 Velocità di 30 cm/s 114
8.3.2 Velocità di 50 cm/s 115

IV Implementazione hardware **120**

9 Progetto del circuito 121
9.1 Descrizione generale 121
9.2 Scelta dei componenti 126
9.2.1 Generatore di segnale 126
9.2.2 7-Way power divider & SP7T 130
9.2.3 SPDT, interruttori a due vie 135
9.2.4 Low Noise Amplifier (LNA) 139
9.2.5 Power detectors 142
9.3 Simulazione 145
9.3.1 Circuito ideale 145
9.3.2 Circuito reale 148
9.3.3 Verifica con le reti neurali 153
9.4 Non idealità del circuito AWR 154
9.4.1 Auto-diffusione delle antenne 155
9.4.2 Mismatch delle impedenze e temperatura 156
9.4.3 Controllo dinamico degli componenti 156

V Conclusione **158**

10 Conclusione 159

Elenco delle figure 161

Elenco delle tabelle 164

Bibliografia 166

Parte I

Introduzione

Capitolo 1

Il problema della contaminazione dei cibi

La presenza di corpi estranei nei prodotti alimentari gioca un ruolo fondamentale nel mercato moderno e può determinare il successo o il fallimento di una qualsiasi azienda del settore.

Tra i motivi principali per cui i controlli qualità caratterizzano oggigiorno una delle fasi principali dell'intero processo produttivo, sono sicuramente presenti ragioni economiche e di sicurezza:

- da una parte, la salute del consumatore non può essere messa a rischio e deve essere opportunamente salvaguardata;
- dall'altra, la possibilità di subire azioni legali e perdere credibilità agli occhi delle persone può comportare un grave danno economico e di immagine.

Tecniche differenti sono state sviluppate nel tempo per cercare di rilevare gli agenti contaminanti ed aziende specializzate nell'ispezione alimentare hanno contribuito al miglioramento della qualità ed all'evoluzione delle varie macchine, come ad esempio *FT System*[1].

Con il termine *contaminante* è indicato qualsiasi oggetto o materiale che non dovrebbe essere presente nel prodotto alimentare: è possibile distinguere tra contaminanti *estrinseci*, se derivano da sorgenti esterne, o contaminanti *intrinseci*, se invece fanno parte del prodotto, ma dovrebbero essere rimossi [1][2].

[1] `https://www.ftsystem.com/`

Con l'aumento della globalizzazione e la crescita dell'industria alimentare, anche i tipi di contaminanti sono cambiati nel tempo, e possono essere categorizzati a seconda della loro natura [3]:

- **Contaminanti biologici**: relativi alla presenza di organismi organici, sono la maggiore causa di intossicazioni alimentari e del deterioramento dei prodotti;
- **Contaminanti fisici**: entrano in questa categoria tutti gli oggetti di materiale diverso da quello alimentare, come plastica, vetro, metalli, capelli etc.
- **Contaminanti chimici**: dovuti a sostanze artificiali aggiunte in fase di produzione o confezionamento.

Tra tutte le tecniche di rilevazione, alcune possono essere applicate solo in circostanze specifiche, altre sono in grado di verificare la presenza di solo alcuni dei possibili materiali contaminanti, altri ancora agiscono sul prodotto alimentare e ne rendono impossibile la vendita.

Questo lavoro di tesi si focalizza sulla seconda categoria, quella dei contaminanti fisici, e cerca delle strategie per poter rilevare qualsiasi corpo estraneo intervenendo direttamente sulla catena di produzione, in modo non invasivo e sicuro per gli operatori che ci lavorano.

1.1 Tecniche per la rilevazione dei contaminanti

Sono di seguito elencate alcune delle possibili tecnologie applicabili in campo alimentare per rilevare i contaminanti fisici, dalle più semplici alle più precise, fino ad arrivare a descrivere i principi dell'**ispezione a microonde** che sarà alla base di questa tesi [2].

1.1.1 Filtri alimentari

Un primo esempio di dispositivo in grado di limitare la contaminazione di un alimento è rappresentato dai filtri alimentari: tramite il passaggio del prodotto attraverso una serie di griglie più o meno ampie, è possibile bloccare i corpi estranei più grandi ed evitare che questi finiscano nella confezione finale.

Come è possibile intuire, una soluzione meccanica di questo tipo è efficace solo nei casi in cui il prodotto alimentare è liquido o molto più piccolo degli eventuali contaminanti, e non può essere utilizzato se non in particolari contesti e situazioni.

I filtri trovano comunque un'importante applicazione nella prevenzione e nella sicurezza dell'ambiente di lavoro. Per esempio, utilizzando dei filtri nei condotti di aerazione è possibile ridurre drasticamente la quantità di microparticelle nell'aria ed abbassare la probabilità che polvere o batteri vadano ad alterare le normali proprietà del prodotto finale.

1.1.2 Raggi X

La tecnologia a raggi X, già applicata in campo medico, trova un ampio utilizzo anche in ambito alimentare. Infatti, le onde elettromagnetiche ad alta energia trasmesse da appositi strumenti ionizzanti sono in grado di penetrare il prodotto e rilevare eventuali anomalie interne allo stesso.

Il principio alla base di tale ispezione è la capacità dei materiali di assorbire i raggi in modo diverso in funzione della propria densità: analizzando la risposta del prodotto ai raggi X è possibile determinare la presenza di contaminanti all'interno della confezione, soprattutto nel caso in cui questi hanno una densità molto diversa da quella del prodotto stesso [4].

Utilizzati inizialmente per rilevare la presenza di frammenti di vetro nelle conserve, i sistemi di ispezione a raggi X si sono evoluti nel tempo per rilevare materiali diversi come metallo, ossa, sassolini e plastica (anche se con maggiore difficoltà).

Nonostante i numerosi vantaggi dell'utilizzare tale tecnologia, sono da riportare alcuni aspetti limitanti che hanno caratterizzato la necessità di trovare soluzioni alternative e più sicure:

1. il sistema di generazione e trasmissione deve essere propriamente calibrato e controllato: ogni alimento può richiedere delle impostazioni differenti affinché sia garantita la corretta penetrazione ed analisi delle sue proprietà;

2. poiché la rilevazione è possibile grazie ad una differenza di densità tra prodotto e contaminante, non si è in grado di verificare la presenza di corpi piccoli o di materiali con proprietà elettromagnetiche simili a quelle dell'alimento.

3. sicurezza: nel caso di esposizione ad un'energia troppo alta, esiste un rischio concreto per gli operatori e per il prodotto alimentare stesso.

1.1.3 Metal Detectors

Nel caso specifico di contaminanti metallici, può essere utile l'applicazione di un metal detector. La precisione di questi strumenti può raggiungere risultati soddisfacenti e le stesse macchine possono essere equipaggiate con sistemi per la rimozione automatica dei contaminanti.

Il limite principale per questa tecnologia è però la capacità di rilevare solo oggetti metallici, escludendo dalle analisi tutti gli altri possibili materiali.

1.1.4 Raggi infrarossi

La *spettroscopia* ad infrarossi (*Near Infraread*, NIR) è effettuata posizionando il prodotto alimentare sul percorso di un raggio ad infrarossi. Misurando l'assorbimento delle diverse frequenze da parte del campione è possibile determinare la composizione del prodotto e rilevare così la presenza di eventuali contaminanti [5].

Questa strategia permette di effettuare misurazioni rapide e sicure, ma le analisi e gli studi effettuati dimostrano che le prestazioni dipendono fortemente dalla sostanza sotto esame. I raggi infrarossi hanno una capacità di penetrazione inferiore rispetto ai raggi X precedentemente descritti e non sono indicati per lavorare con soluzioni liquide ed acquose.

1.2 Microwave Imaging

L'ultima strategia qui presentata per rilevare i contaminanti nei cibi in barattolo è l'*Imaging a Microonde* (MWI): si tratta di una tecnica che, utilizzando segnali a bassa energia nel range di frequenza delle microonde, permette di costruire un'immagine spaziale delle proprietà elettromagnetiche del prodotto [2][6].

Trasmettendo i segnali RF tramite una serie di antenne, è possibile misurare le matrici di diffusione dopo l'interazione con l'alimento; l'analisi di queste informazioni facilita l'esecuzione di algoritmi per il riconoscimento dei composti e permette di distinguere tra loro tutti i materiali presenti all'interno barattolo.

In funzione della serie di comportamenti assunti dai vari materiali dopo l'applicazione delle microonde dall'esterno, è quindi possibile verificare la presenza di contaminanti attraverso lo studio delle discontinuità rilevate nelle proprietà dielettriche del prodotto.

Nel caso in cui non fosse necessario costruire la mappa spaziale del campione analizzato, ma fosse richiesta solo la verifica dell'assenza di corpi estranei, si può sostituire la parte di *imaging* con una di *sensing*, più semplice e meno dispendiosa dal punto di vista computazionale.

Il *Microwave Sensing* è una delle più moderne e promettenti tecniche per il rilevamento dei contaminanti, grazie anche alle seguenti caratteristiche:

- Non distruttivo;
- Senza contatto;
- Non ionizzante;
- Real-time;
- Semplice da applicare.

Nei capitoli successivi saranno presentati nel dettaglio i problemi che questa tesi tenta di risolvere, integrando insieme l'utilizzo di strumenti a microonde e tecniche di Machine Learning.

Definendo un piano strategico con il quale affrontare il problema della contaminazione sarà possibile, al termine del progetto, sviluppare un prototipo funzionante in real-time da applicare su nastri trasportatori industriali per rendere la produzione dei prodotti confezionati più sicura che mai.

Parte II

Introduzione al machine learning

Capitolo 2

Machine Learning

Con il termine *Machine Learning* viene indicato un sottoinsieme dell'intelligenza artificiale che si occupa della costruzione di modelli analitici per l'apprendimento automatico della macchina: senza la necessità di una programmazione specifica, un sistema intelligente è in grado di imparare a compiere delle scelte ottimizzandone il risultato con l'esperienza.

L'idea alla base di tutta l'*Artificial Intellingence* è quella di insegnare alle macchine a comportarsi come l'essere umano, a pensare ed operare come se questa avesse un'intelligenza umana; grazie all'osservazione dello scenario in cui viene utilizzata ed all'analisi dei dati a disposizione, la macchina è in grado di rilevare degli schemi e svolgere l'operazione richiesta.

Come sarà visto in questo capitolo, tra i vantaggi principali dell'utilizzare macchine con algoritmi di machine learning piuttosto che programmate, c'è la capacità delle prime di scoprire pattern e relazioni tra input ed output che l'uomo non è in grado di vedere, ottenendo spesso delle performance altrimenti irraggiungibili: una prima dimostrazione delle capacità delle nuove reti neurali è riportata nell'articolo [7], in cui possiamo vedere come il lavoro di alcuni ricercatori abbia portato la macchina ad avere una tasso di riconoscimento delle immagini più alto di quello umano.

Prima di entrare nel merito delle tecniche che saranno utilizzate, è utile definire una terminologia con la quale descrivere le grandezze utili all'apprendimento del modello [8]; tra le varie definizioni si trovano:

- **Label** (etichetta): rappresenta la variabile dipendente che si vuole predire (y); a seconda dell'obiettivo della macchina può essere una grandezza continua o discreta, numerica o categorica;

- **Features** (caratteristiche): sono le variabili di input utilizzate per effettuare la previsione; descrivono il dato x tramite n valori $\{x_1, x_2, ..., x_n\}$;

- **Example** (esempio): rappresenta una singola unità (x) appartenente al set di dati; a seconda dell'utilizzo, gli examples possono essere divisi in:
  - *Labeled* se includono sia le features che le relative labels (utilizzati nella fase di allenamento);
  - *Unlabeled* se contengono solo le features (usati nella fase di test per effettuare le previsioni su dati mai visti);
- **Model** (modello): è il meccanismo che definisce la relazione tra le caratteristiche e l'etichetta, contiene i parametri interni ed è il componente da allenare con i dati noti per poter essere utilizzato nelle future previsioni;
- **Hyperparameters** (iperparametri): sono i parametri regolabili utilizzati per controllare ed ottimizzare le prestazioni del modello nella fase di allenamento.

Nelle sezioni seguenti saranno analizzati i principi matematici alla base del machine learning, partendo da una descrizione generale delle operazioni caratteristiche, fino ad arrivare alla presentazione dei principali modelli tra cui scegliere per svolgere l'operazione desiderata.
In ordine saranno presi in esame i seguenti aspetti:

1. Algoritmo generale e riduzione dell'errore;
2. Classificatori;
3. Reti neurali;
4. Metriche di valutazione e ricerca Bayesiana.

2.1 Algoritmo generale e riduzione dell'errore

Tra i diversi approcci che è possibile seguire per affrontare un problema di machine learning, molte scelte sono guidate dal tipo e dalla quantità di informazioni a disposizione; a seconda del dataset utilizzato per la fase di training, si possono dividere tutti gli algoritmi tra le due principali categorie di *supervised learning* e *unsupervised learning* [9]:

Apprendimento supervisionato

In questo tipo di allenamento, i dati in ingresso contengono per ogni esempio sia le features che la relativa label; lo scopo della macchina è quello di studiare la relazione che esiste tra le due variabili e derivarne di conseguenza una funzione matematica per effettuare le future predizioni.
Fornendo questo tipo di informazioni si vuole far convergere le previsioni del modello verso i reali valori dell'uscita e minimizzare con l'esperienza la differenza tra le due quantità.

Apprendimento non supervisionato

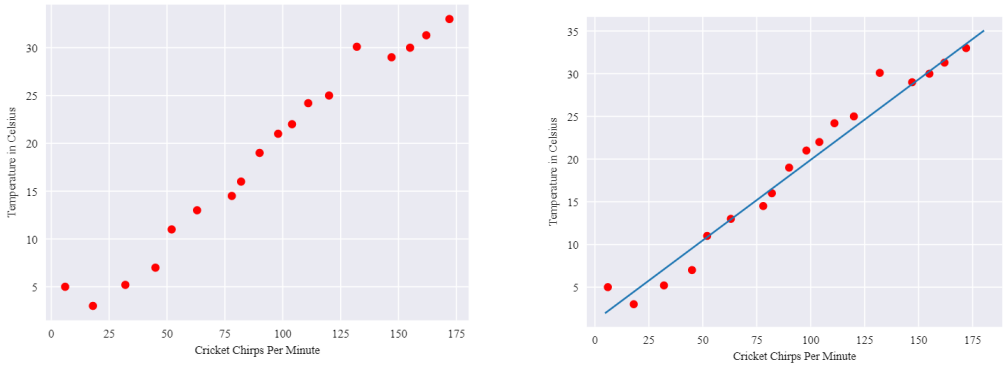
A differenza del primo caso, in un allenamento non supervisionato non è disponibile l'informazione sull'etichetta, ma è da ricercare uno schema tra dati il cui significato è sconosciuto o non rilevante; il modello sarà istruito per analizzare gli esempi, rilevarne comportamenti comuni o anomali e predire un risultato coerente con tutti i dati.

In questa attività di tesi saranno studiati i principali meccanismi per un apprendimento supervisionato ed si entrerà nel dettaglio dei cosiddetti *classificatori*, modelli ed algoritmi specifici per la predizione di valori discreti e categorie di appartenenza.

Saranno inizialmente seguiti i corsi offerti da Google [8] ed IBM [10] per avere un'introduzione generale al mondo dell'intelligenza artificiale e si arriverà nelle sezioni successive a sviluppare una rete neurale specifica per la rilevazione di agenti contaminanti nei cibi in package.

2.1.1 Regressione lineare

Come discusso nel paragrafo precedente, la soluzione ad un problema di machine learning ha inizio con l'ottenimento e l'analisi dei dati caratterizzanti il fenomeno in esame; andando a visualizzare il dataset in un piano n-dimensionale, viene chiesto alla macchina di modellare la relazione tra l'insieme delle features in ingresso ed il relativo valore di uscita.
Immaginando di avere il seguente set di punti, è cercata la funzione che meglio approssima la variabile dipendente y:



(a) Dataset lineare

(b) Funzione di fitting lineare

Figura 2.1: Esempio di dataset lineare e relativa funziona di fitting

In figura 2.1b è possibile vedere come una retta riesca in questo caso ad approssimare tutti gli esempi in modo adeguato ed a capire come questa possa essere utilizzata per prevedere il comportamento di dati ancora non visti.
Il modello sarà impiegato per stimare i coefficienti di una funzione lineare, dove i parametri incogniti sono il *fattore di peso* ω ed il *termine noto* b.

$$y' = \omega x_1 + b_1 \tag{2.1}$$

La ricerca dei valori ottimali per queste variabili caratterizza la fase di allenamento della rete e si basa su un processo iterativo di aggiornamento e valutazione: per capire la bontà del modello sarà necessario introdurre una quantità per misurare l'accuratezza delle previsioni, utile sia per verificare le prestazioni, che per capire se la rete stia migliorando oppure no.

Il problema della regressione porta così a considerare la cosiddetta *funzione di costo* ed a calcolare, per ogni coppia valore reale - previsione, la differenza tra le due grandezze.

Definendo con y l'etichetta reale e con y' il valore predetto per l'esempio x, sono definite diverse misure dell'errore ed è possibile scegliere quella più adatta al problema in funzione della caratteristica da analizzare.
Tra le formule più comuni si trovano:

- **Errore Assoluto Medio** (MAE): $\frac{1}{N}\sum_{i=0}^{N} \mid y_i - y'_i \mid$;
- **Errore Quadratico medio** (MSE): $\frac{1}{N}\sum_{i=0}^{N}(y_i - y'_i)^2$;
- **Radice dell'Errore Quadratico Medio** (RMSE): $\sqrt{\frac{1}{N}\sum_{i=0}^{N}(y_i - y'_i)^2}$.

In tutti i casi, lo scopo ultimo dell'algoritmo di machine learning sarà quello di minimizzare questo valore, arrivando così ad ottenere la funzione che meglio descrive la relazione ingresso-uscita.

Lo schema generale del processo di training può essere quindi riassunto con il seguente diagramma:

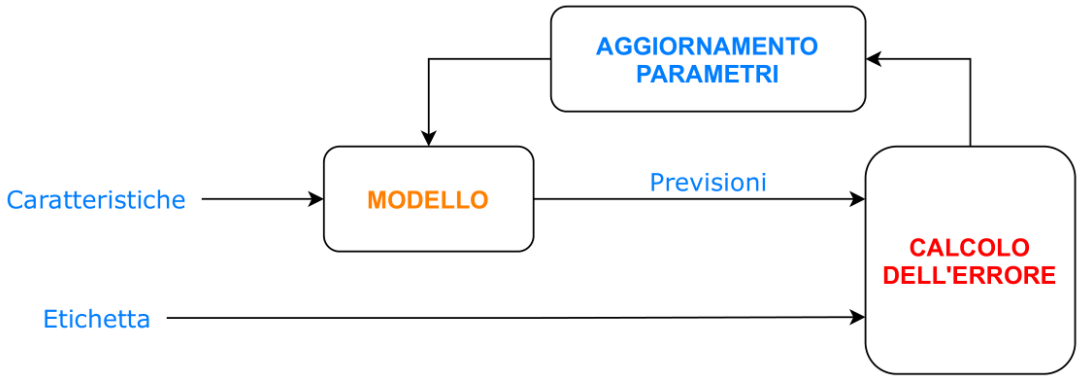


Figura 2.2: Flow chart per l'algoritmo di training

Le caratteristiche e le etichette vengono usate come input del sistema, mentre il modulo per il calcolo dell'errore fornisce lo strumento per modificare e ottimizzare i parametri alla fine di ogni epoca; a processo concluso, l'output sarà rappresentato dal modello stesso, che a questo punto sarà descritto dagli iperparametri allenati e potrà essere esportato per essere poi successivamente testato ed utilizzato.
I valori iniziali dei vari coefficienti non sono importanti ai fini dell'allenamento, e saranno scelti in modo casuale dell'algoritmo.

2.1.2 Riduzione dell'errore: Gradient descent

I meccanismi di ottimizzazione possono essere più o meno complessi e permettono al modello di trovare i punti di minimo nella funzione di costo.
Tra tutti i possibili approcci si possono adoperare:

1. Algoritmi non iterativi che risolvono il problema per un solo punto;
2. Algoritmi iterativi che convergono verso soluzioni accettabili indipendentemente dai valori iniziali;
3. Algoritmi iterativi applicati ad un set di problemi con funzione di costo non convessa (come per le reti neurali); in questi casi l'inizializzazione dei parametri è fondamentale per l'ottenimento di buoni risultati.

L'algoritmo di **Gradient Descent** (discesa del gradiente) è uno dei più utilizzati in machine learning e basa il suo funzionamento sulla modifica graduale dei pesi, in cui il sistema viene ottimizzato con il passare del tempo e degli esempi.

Immaginando di poter rappresentare l'andamento dell'errore J in funzione del parametro ω da ricavare, si può immaginare il seguente comportamento:

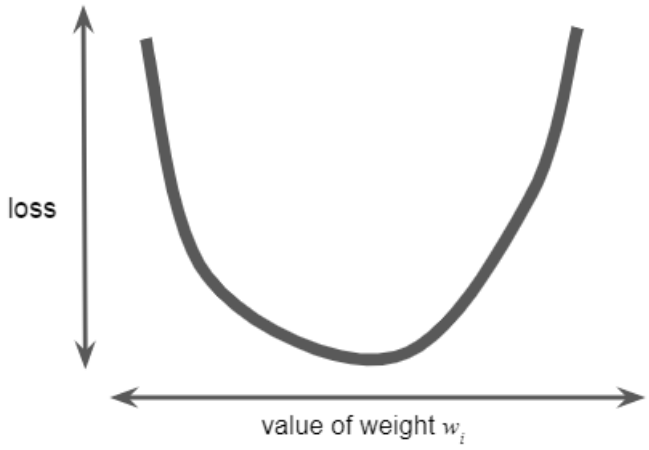


Figura 2.3: Andamento dell'errore in funzione del parametro ω

Una volta ricavata la posizione iniziale sulla curva, il meccanismo di aggiornamento permette di modificare i parametri di una quantità proporzionale al gradiente dell'errore, facendo in modo che il modello converga verso il punto di minimo della funzione stessa.

Ad ogni ciclo, l'algoritmo prevede di calcolare le derivate parziali della funzione di costo e di utilizzare questi valori per ricavare i parametri da utilizzare nell'iterazione successiva [11]; lavorando sui pesi w_i ed i bias b_i della funzione 2.1:

$$\nabla_w = \frac{\partial}{\partial \omega_i} J(\omega) \tag{2.2}$$

$$\nabla_b = \frac{\partial}{\partial b_i} J(\omega) \tag{2.3}$$

I gradienti ∇_w e ∇_b vengono così ottenuti e sottratti da tutti i coefficienti dopo essere stati moltiplicati per il cosiddetto *learning rate* (α).

$$\omega_i = \omega_i - \alpha \nabla_w \tag{2.4}$$

$$b_i = b_i - \alpha \nabla_b \tag{2.5}$$

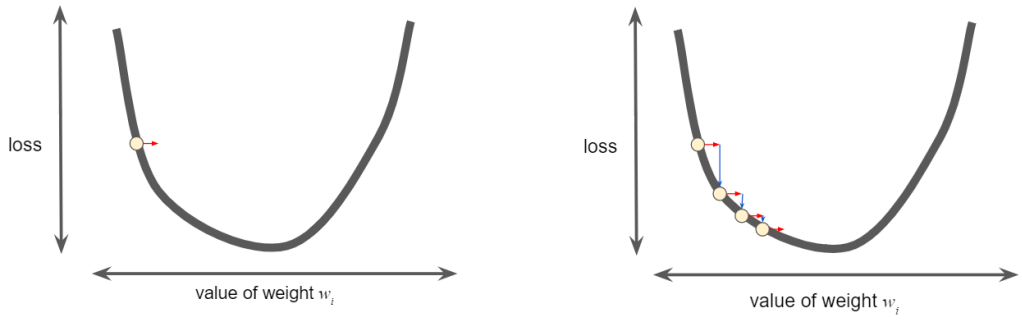


Figura 2.4: Andamento dell'errore in funzione del parametro ω

Per poter applicare correttamente l'algoritmo di gradient descent, occorre impostare alcuni parametri prima dell'avvio della procedura, al fine di massimizzarne l'efficacia e ridurre il costo computazionale dell'operazione.

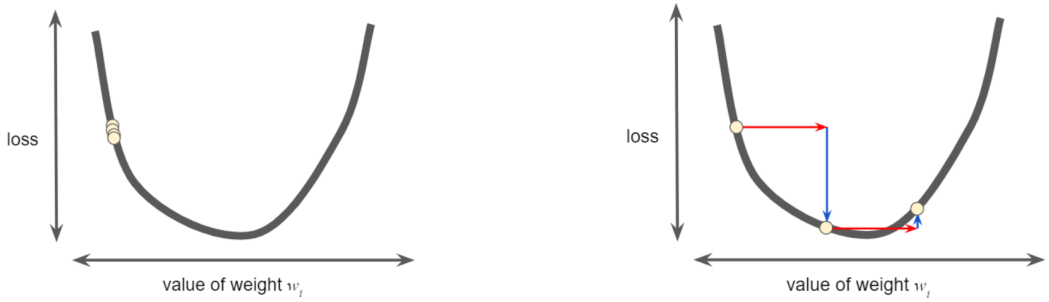
Per capire il significato di queste variabili e l'influenza che queste hanno all'interno della formula, è possibile descrivere per ognuna le principali caratteristiche.

Learning rate (tasso di apprendimento)

Come è visibile nelle equazioni 2.4 e 2.5, il gradiente viene moltiplicato per uno scalare prima di essere sottratto dai valori attuali dei coefficienti, e questo determinerà la grandezza del passo che sarà da effettuare.

Il parametro α prende il nome di *learning rate* e caratterizza la velocità di apprendimento del modello; a seconda del valore scelto ed impostato in fase di training, si può ottimizzare l'algoritmo rispetto alla velocità di esecuzione o all'analisi dettagliata di ogni step intermedio.
Entrambe le soluzioni possono avere degli effetti positivi o negativi a seconda dei casi, ma in generale:

- Un passo troppo piccolo permette di effettuare molti spostamenti controllati, ma richiede molto tempo e può risultare oneroso;
- Un passo troppo grande permette di coprire un range maggiore ad ogni iterazione, ma rischia di superare il punto di minimo e di non raggiungere mai la convergenza della curva.



(a) Learning rate troppo piccolo (b) Learning rate troppo grande

Figura 2.5: Possibili errori sulla scelta del learning rate

Se fosse possibile rappresentare la funzione di costo in relazione a diversi valori di learning rate, i comportamenti ottenuti sarebbero simili ai seguenti, e solo la curva rossa garantirebbe buoni risultati dopo un tempo non eccessivamente lungo:

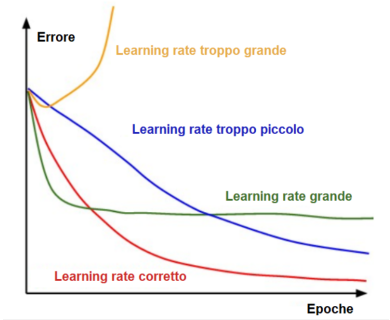


Figura 2.6: Gradient descent con diversi valori di learning rate

Batch size (dimensione del batch)

Nell'algoritmo di gradient descent, è indicato con il termine *batch* il numero di esempi che vengono usati, su una singola iterazione, per calcolare il gradiente.

Come nel caso del learning rate, anche la dimensione del batch può essere modificata per ridurre il tempo necessario all'allenamento, a discapito però della precisione e dell'analisi dei singoli valori; in funzione della quantità di dati utilizzati per ogni passo, si può distinguere [11]:

1. Discesa del gradiente Batch (BCD);
2. Discesa del gradiente stocastica (SGD);
3. Discesa del gradiente mini-batch (mini-batch SGD).

Nel primo caso, l'errore viene calcolato per ogni esempio nel set, ma l'aggiornamento viene effettuato solo dopo che tutte le osservazioni sono state effettuate: è possibile utilizzare un learning rate costante, ma aumenta la probabilità di subire l'effetto della ridondanza di dati uguali.

Nella seconda modalità, i parametri vengono modificati dopo ogni iterazione (batch = 1) ed è per questo possibile eseguire un ciclo con un solo esempio; questa scelta è in genere la più veloce, ma è soggetta a rumore e ad una maggiore varianza del tasso di errore.

L'ultima possibilità rappresenta una combinazione ed un compromesso tra le due soluzioni precedenti: l'aggiornamento dei parametri non viene effettuato dopo aver analizzato tutto il set, ma è sufficiente aspettare l'osservazione di un numero di esempi specificato dal batch size.

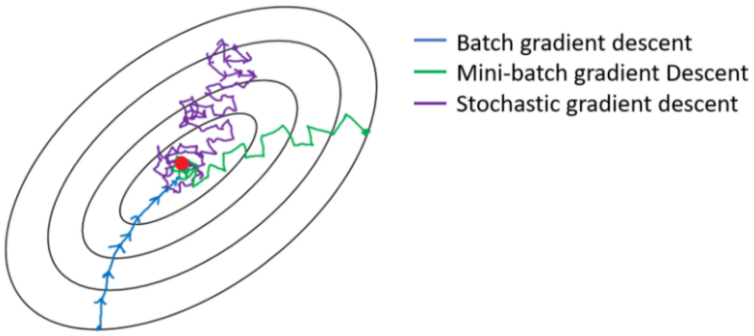


Figura 2.7: Varianti di Gradient Descent in funzione del batch size

2.1.3 Generalizzazione

Come già detto nelle pagine precedenti, una macchina intelligente viene allenata con lo scopo di modellare una funzione che leghi le variabili di ingresso alla relativa uscita; una volta scoperto il legame che unisce i dati tra di loro, si è in grado di applicare la stessa trasformazione a qualunque input e predire così l'etichetta associata a nuovi esempi.

Nell'utilizzo delle tecniche di machine learning, il modello finale deve essere in grado di *generalizzare* qualunque operazione esso svolga e deve per questo essere istruito per fornire risposte adeguate anche difronte ad esempi non presenti nel dataset originale.

La **generalizzazione** è l'abilità della macchina di adattare il proprio comportamento a compiti e situazioni che non ha ancora affrontato e si riferisce a quanto bene i concetti appresi dal modello si applicano ad esempi specifici non visti durante l'apprendimento.

Possibili problemi legati ad una mancanza di generalizzazione sono rappresentati nell'immagine seguente:

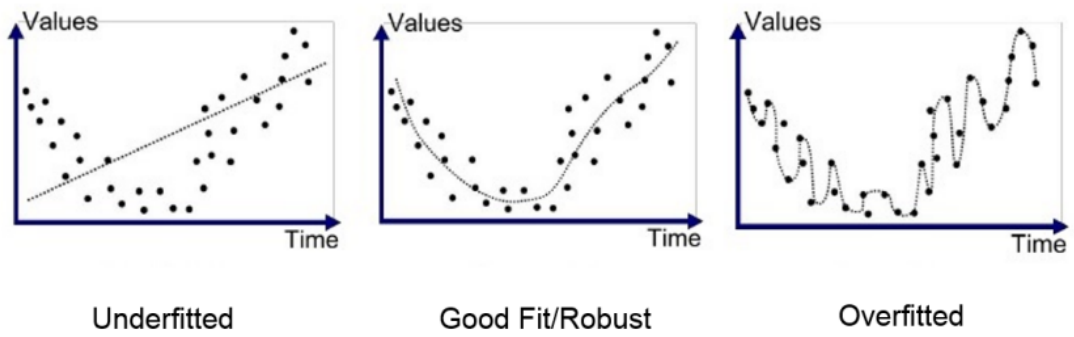


Figura 2.8: Esempi di sotto-adattamento e sovra-adattamento

Nei grafici in Figura 2.8 si vede tratteggiata la funzione imparata dal modello, insieme alla nuvola di punti utilizzata per ricavarla; come indicato nella stessa immagine, solo la curva centrale è in grado di generalizzare bene l'andamento dei valori, mentre le altre due sono soggette rispettivamente ai fenomeni di *sotto-adattamento* e *sovra-adattamento*.

Questi effetti dipendono fortemente dalla complessità del modello e possono essere così descritti:

Underfitting (sotto-adattamento)

La condizione di *underfitting* si verifica nel caso in cui il modello non è in grado di rilevare il pattern dei dati in ingresso; la funzione caratteristica è troppo semplice per poter descrivere la relazione e non sarà capace di effettuare previsioni corrette.

Nelle varie applicazioni è spesso facile verificare una situazione di sotto-adattamento, e può essere risolta semplicemente complicando la rete e/o aggiungendo esempi al dataset di allenamento.

Overfitting (sovra-adattamento)

In modo contrario al precedente, l'overfitting si verifica quando il modello viene istruito eccessivamente su un preciso set di esempi, andando ad imparare una funzione troppo complessa per essere utilizzata in situazioni meno stringenti.

La rete finale è caratterizzata da ottimi risultati se applicata sugli stessi dati di training, ma vede una precisione molto limitata se lavora su dati nuovi e mai visti.

La correzione di un problema di overfitting è sicuramente più difficile di quella di sotto-adattamento e deve coinvolgere sia il modello che il dataset iniziale: se non è possibile collezionare nuovi esempi con cui testare la macchina dopo l'allenamento, conviene dividere, prima di qualunque operazione, i dati a disposizione in più sottogruppi, in modo da riservarne una parte per la verifica finale.

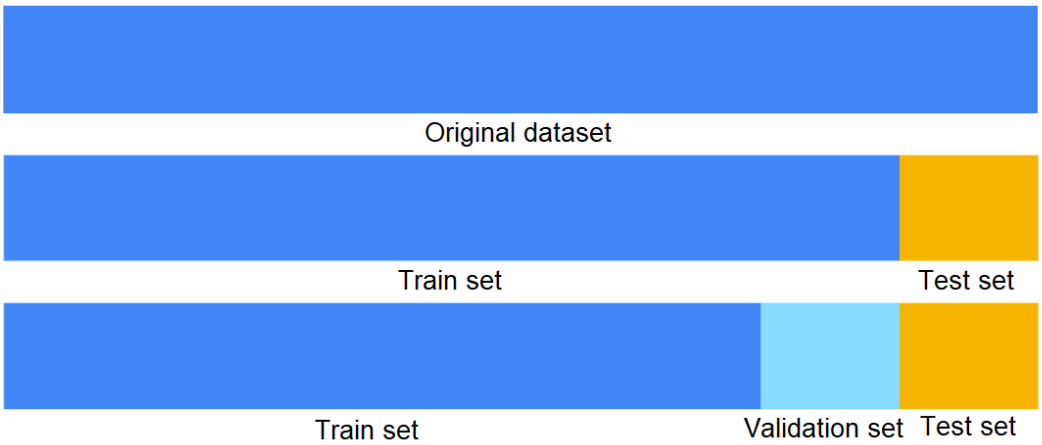


Figura 2.9: Partizione del dataset in train - validation - test set

- Una prima divisione viene effettuata per ottenere il *test set*, l'insieme di dati che saranno utilizzati al termine dell'algoritmo per misurare le prestazioni reali del modello.
- Per ridurre le probabilità di avere un sovra-adattamento, il rimanente *training set* viene ulteriormente partizionato, creando un *validation set* per la valutazione delle prestazioni durante l'allenamento.

L'algoritmo generale viene quindi diviso in tre fasi principali:

1. *Allenamento*: analizzando il training set, la macchina viene istruita cercando di adattare le proprie previsioni alle label degli esempi;
2. *Validazione*: per verificare la qualità dell'allenamento del punto 1, il modello ottenuto viene testato ad ogni ciclo sul validation set e modificato opportunamente in funzione dei risultati;
3. *Verifica*: ad allenamento concluso, i parametri relativi al modello più efficiente vengono salvati ed utilizzati per la verifica finale; questa fase è fondamentale per la valutazione delle reali capacità della rete ed implementa le principali funzioni per il calcolo delle metriche che saranno successivamente valutate.

Riassumendo lo schema descritto in questa prima sezione, il processo di machine learning che permette di allenare una macchina per svolgere la funzione desiderata è così composto:

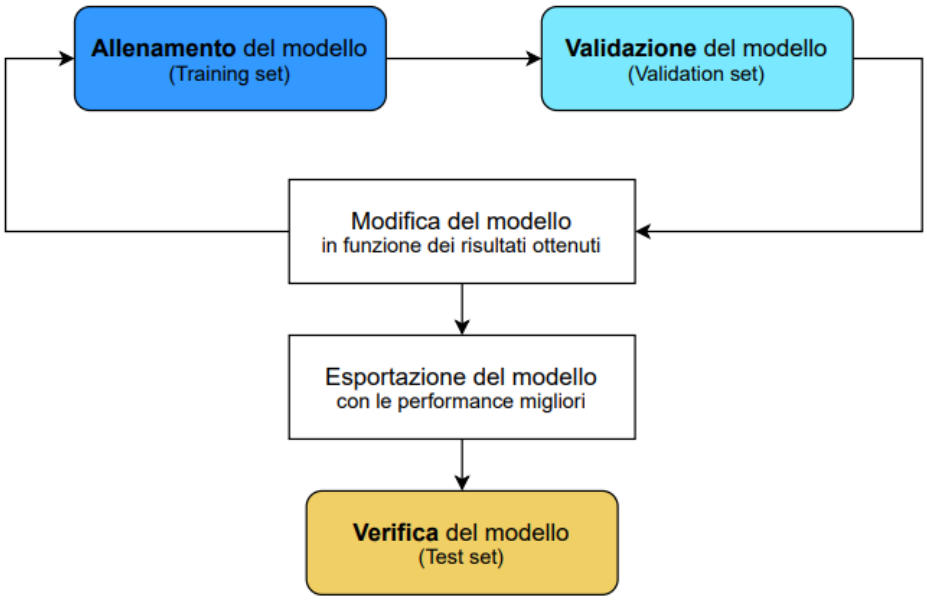


Figura 2.10: Algoritmo di allenamento e verifica del modello

Per evitare di utilizzare lo stesso validation set per tutte le iterazioni, alcune tecniche di **Cross Validation** (convalida incrociata CV) possono essere aggiunte per modificare l'insieme degli esempi da assegnare alle prime due fasi.
Tra gli approcci più comuni è presente il *k-fold CV* che basa il suo funzionamento sulla divisione del training set in k - 1 gruppi.

Intervenendo sulla partizione del dataset iniziale, l'utilizzo di una validazione incrociata aiuta il modello ad evitare, con maggiore probabilità, la condizione di overfitting: permettendo all'algoritmo di allenare la macchina con dati diversi ad ogni ciclo, la possibilità di avere ridondanza tra gli esempi usati si riduce fortemente, con un conseguente miglioramento dei risultati finali.

Come mostrato nell'immagine seguente, il validation set viene modificato durante la fase di training ed utilizzato per allenare da zero il modello in situazioni leggermente diverse ad ogni iterazione; il risultato viene quindi ottenuto calcolando la media delle metriche sul validation set.

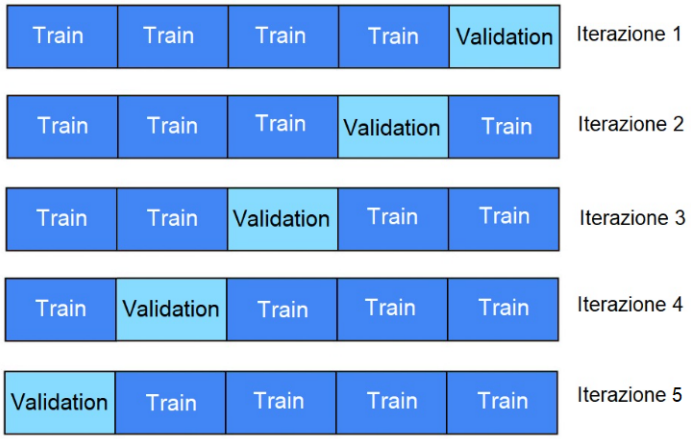


Figura 2.11: Applicazione di un 5-fold cross-validation

2.1.4 Regolarizzazione

Le condizioni di overfitting possono essere rilevate analiticamente e graficamente confrontando tra loro le prestazioni sul training e sul validation set: il comportamento desiderato prevede che le due curve scendano insieme convergendo verso lo zero, ma solitamente si assiste a situazioni in cui la seconda diverge e peggiora con l'aumentare del numero di iterazioni.

Tecniche di *regolarizzazione* possono essere in questo caso utili per ridurre la complessità del modello, evitando il sovra-adattamento mentre si cerca di minimizzare la funzione di costo [8].
Mentre un'applicazione classica prevede la ricerca del valore minimo dell'errore

$$Minimo(Errore(Dato|Modello))$$

applicando delle funzioni di regolarizzazione, si vuole ridurre quanto più possibile la somma tra l'errore e la complessità:

$$Minimo(Errore(Dato|Modello) + \lambda Complessit\grave{a}(Modello))$$

Il termine aggiuntivo interviene nella modifica dei pesi e può essere calcolato in funzione di parametri diversi; tra le tecniche principali e più comuni è possibile trovare:

- **Regolarizzazione L2**: il termine è proporzionale al quadrato dei coefficienti;
- **Regolarizzazione L1**: viene calcolato con il valore assoluto dei coefficienti.

Nel primo caso, il termine di regolarizzazione viene calcolato a partire dai singoli pesi $\omega_1, .., \omega_n$ ed utilizzato per normalizzarne la media vicino allo zero:

$$L_2 = ||\omega||_2^2 = \omega_1^2 + \omega_2^2 + ... + \omega_n^2$$

Nel secondo caso invece, si tenta di annullare completamente alcune features dal modello impostando i relativi pesi esattamente a zero; se la regolarizzazione L2 agisce sui quadrati, L1 interviene nel modificare la somma dei termini semplici:

$$L_1 = |\omega| = |\omega_1| + |\omega_2| + ... + |\omega_n|$$

Per avere un confronto tra le due tecniche e capire quando è meglio sceglierne una rispetto all'altra, è possibile guardare gli effetti che queste hanno sui dati di partenza:

1. Con una regolarizzazione L2, si è in grado di disperdere l'errore tra tutti i pesi del sistema, e di arrivare ad avere un'ottimizzazione più accurata del modello finale; pur mantenendo tutti i coefficienti, è possibile stimare l'importanza di ognuno rispetto al valore di uscita e penalizzare quelli meno significativi per accentuare l'effetto degli altri;

2. La regolarizzazione L1 restituisce in uscita un valore binario tra 0 e 1 per ogni caratteristica del modello e può essere utile per iniziare un'operazione di *feature selection*: in dataset molto grandi può capitare che alcune caratteristiche non siano utili ai fini della classificazione e può risultare conveniente rimuoverne completamente il contributo per semplificare la funzione finale.

Mentre la regolarizzazione L2 cerca di evitare il sovra-adattamento calcolando e sottraendo il valor medio dei dati, la regolarizzazione L1 cerca di trovare il punto di mediana nella distribuzione dei dati stessi, in cui la funzione di costo raggiunge presumibilmente il valore minimo.

Un'altra possibile tecnica di regolarizzazione è data dal cosiddetto **Dropout**: mentre le tecniche L1 e L2 ci permettono di semplificare la rete modificando la funzione di costo, questa particolare strategia basa il suo funzionamento sulla modifica della rete stessa [12].

Durante la fase di training di una rete neurale, prima di iniziare una nuova epoca, alcuni neuroni vengono casualmente rimossi dai livelli intermedi e la rete risultante, più piccola e semplice, viene allenata e valutata.
Il processo viene quindi ripetuto più volte, tenendo e scartando neuroni diversi ad ogni epoca; al termine dell'allenamento, il modello finale viene scelto ed ottimizzato a partire dalla media dei risultati precedenti e viene così costruito sulla base di molte reti leggermente diverse l'una dall'altra.

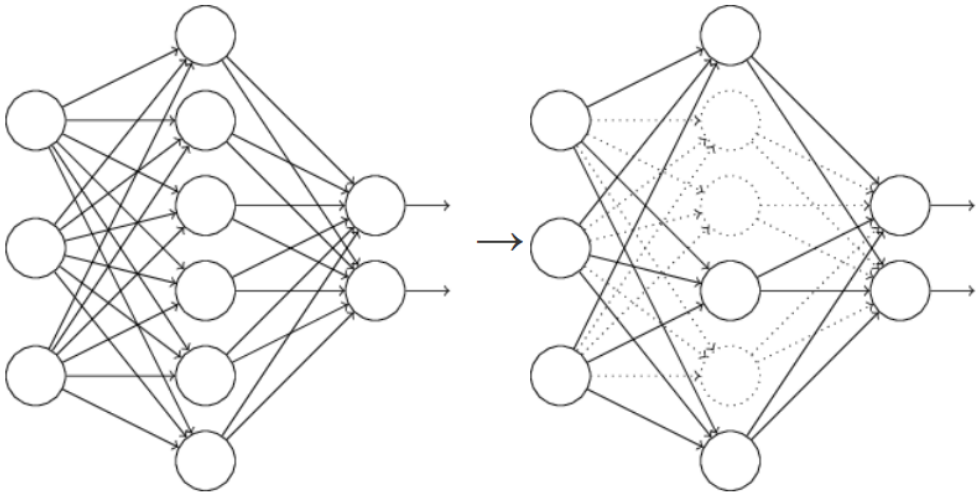


Figura 2.12: Applicazione della tecnica di dropout sulla rete neurale

Questo procedimento viene applicato soltanto durante la fase di training, per facilitare la ricerca degli iperparametri finali, mentre durante il normale utilizzo sarà comunque usata la rete nella sua interezza.

2.2 Teoria dei classificatori

I due principali approcci per risolvere un problema di machine learning basano la loro applicazione sul tipo dell'etichetta da predire; tra i modelli di apprendimento supervisionato è possibile distinguere in particolare [8]:

- Regressori
- Classificatori

I *primi* applicano algoritmi e tecniche per prevedere un valore continuo e per ricercare la funzione (lineare o non-lineare) che leghi la variabile dipendente y ad una o a tutte le caratteristiche dell'esempio x.

Nel *secondo* caso la label richiesta può assumere invece un set limitato di valori. In machine learning, la *classificazione* è vista come il processo necessario per categorizzare gli esempi in specifici gruppi ed insiemi; il valore target è una variabile discreta e rappresenta la classe di appartenenza di ogni dato analizzato.

Il progetto di tesi tenta di rilevare i corpi estranei nei cibi in package: attraverso lo studio dei parametri di diffusione acquisiti sui singoli barattoli, la macchina deve essere in grado di identificare gli alimenti contaminati distinguendoli da quelli sicuri, segnalandoli all'utente con un apposito segnale.

Traducendo il problema in termini di intelligenza artificiale, si vuole costruire un dispositivo in grado di classificare ogni esempio come *contaminato* o *sicuro*, scegliendo tra due categorie e definendo quindi un caso di classificazione binaria.

Tra tutte le possibili tecniche utilizzabili per implementare un classificatore, le più comuni sono brevemente trattate nelle pagine seguenti dove per ognuna sono elencate la principali caratteristiche con lo specifico campo di applicazione [13].

1. Regressione logistica
2. Alberi decisionali
3. Support Vector Machines (SVM)

2.2.1 Regressione logistica

I metodi probabilistici sono alla base della teoria dei classificatori e cercano di identificare la classe migliore di un dato esempio implementando ed utilizzando algoritmi con base statistica.

L'idea fondamentale per il concetto di *regressione logistica* è la stessa già discussa al capitolo 2.1.1 con la regressione lineare: cercando una relazione tra le variabili indipendenti, si vuole prevedere il valore di uscita della funzione, associando a questo un significato particolare a seconda del contesto.

Mentre nel caso dei regressori il valore di uscita rappresenta di per sé l'etichetta ricercata, con i classificatori si tende ad associare a questo numero il significato di probabilità; utilizzando una soglia, è possibile classificare ogni esempio a seconda che la probabilità stessa sia maggiore o minore del valore scelto.

La probabilità che un esempio appartenga ad una specifica classe è indicata con l'espressione seguente, dove la lettera i rappresenta la classe scelta:

$$P(y(x) = i \mid x) \tag{2.6}$$

Dato l'esempio x con tutte le sue caratteristiche, P è la probabilità che la variabile di uscita y rientri nel set di valori della classe i.

Per avere una probabilità compresa tra 0 ed 1, è necessario applicare una funzione il cui output sia sempre limitato tra questi due valori, come la curva sigmoide della Figura 2.13

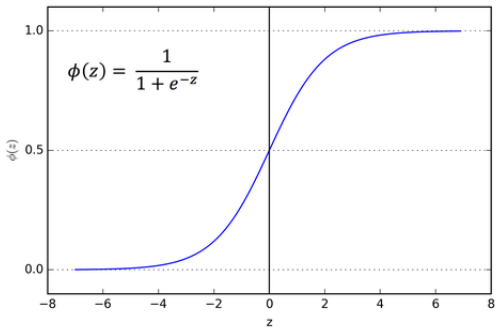


Figura 2.13: Curva sigmoide

In questo modo, a partire dai dati di allenamento, addestriamo il classificatore a svolgere una regressione logistica come:

$$P(y(x) = i \mid x) = \frac{1}{1 + e^{-\theta x}} \tag{2.7}$$

con θ il vettore di parametri da calcolare.

L'utilizzo di questo modello è utile in molte discipline e trova applicazione in campi che vanno da quello medico a quello sociale; in generale, i migliori risultati sono ottenuti quando [8]:

- I dati analizzati prevedono una classificazione binaria (0 o 1)
- L'applicazione ricerca un risultato probabilistico
- I dati possono essere facilmente divisi da una funzione lineare
- Si vuole misurare l'impatto di ogni feature sul valore di uscita.

2.2.2 Alberi decisionali

Questo tipo di classificatore analizza e categorizza gli esempi in ingresso attraverso un partizionamento gerarchico dei dati; il criterio utilizzato per la costruzione della classica struttura ad albero prevede la verifica di uno o più attributi e, in funzione del risultato, un nuovo ramo viene creato e percorso.

L'approccio generale è quello di provare ricorsivamente a dividere i dati ad ogni nodo, in modo da aumentare la differenza esistente tra due specifiche classi: le capacità del modello saranno massimizzate quando gli attributi verificati nei nodi finali saranno mutuamente esclusivi tra le due possibili scelte.

L'algoritmo applicato è costituito da tre macro passaggi ripetuti dall'inizio alla fine in modo ciclico per permettere alla macchina di discriminare correttamente ciascun caso possibile:

1. Scegliere un attributo dal dataset
2. Valutare la capacità dell'attributo scelto di discriminare tra loro le due classi
3. Dividere il dataset in funzione dell'attributo migliore

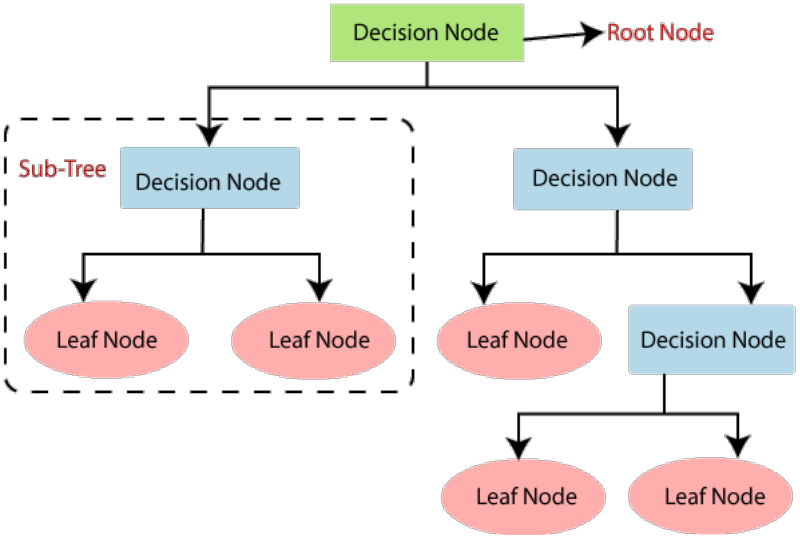


Figura 2.14: Struttura generale di un albero decisionale

Alcuni attributi possono influenzare in modo più importante di altri il risultato di ciascuna scelta, e proprio la selezione dei migliori parametri determinerà la purezza del valore finale.
Per quantificare la differenza tra le possibili scelte ad ogni nodo è possibile calcolare l'*entropia*, la misura del disordine e della casualità contenuta nelle informazioni del dataset [10].

Definita con p_i la probabilità che la classe i sia preferita tra tutte le k possibili scelte al nodo N, l'entropia viene calcolata con la seguente espressione:

$$E(N) = -\sum_{i=1}^{k} p_i log(p_i) \tag{2.8}$$

Tanto più il valore dell'entropia sarà piccolo, tanto meno la distribuzione dei dati sarà uniforme ed il nodo sarà puro; in questi casi, la divisione gerarchica sarà semplificata ed ogni classificazione potrà essere più precisa ed accurata.

Tra i principali vantaggi di utilizzare un albero decisionale rispetto agli altri modelli, troviamo aspetti come [14]:

- Facilità di lettura ed interpretazione
- Facilità di costruzione
- Minore necessità di preprocessare i dati

2.2.3 Support Vector Machines (SVM)

I classificatori di questo tipo utilizzano condizioni lineari per separare le classi tra di loro, ricercando la funzione che meglio distingue le diverse possibilità.

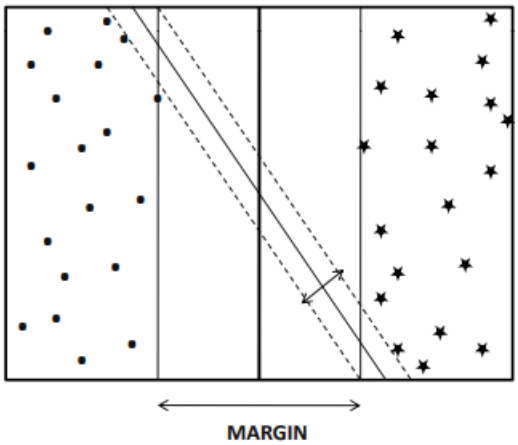


Figura 2.15: Support Vector Machine con ottima separazione degli iperpiani

L'aspetto più importante per il funzionamento delle macchine a supporto vettoriale è la ricerca dell'*iperpiano con il margine massimo.*

Immaginando di avere la distribuzione di punti mostrata in Figura 2.15, è possibile definire due diversi iperpiani con i loro specifici vettori di supporto: da una parte la separazione tra le due classi può essere effettuata attraverso le due linee tratteggiate ed oblique, dall'altra parte lo stesso scopo può essere raggiunto con le due linee continue verticali.
E' evidente in questo caso come un iperpiano presenti un margine maggiore rispetto all'altro e sia per questo preferibile per la sua maggiore generalizzazione nei confronti di esempi mai visti prima.
Per addestrare correttamente una SVM ed ottenere con questa buone prestazioni, è quindi necessario trovare l'iperpiano che presenta tra tutti il margine massimo di separazione tra punti di classi diverse.

In generale, per dati a d dimensioni, si può assumere l'esistenza di un iperpiano della forma

$$Wx + b = 0 \tag{2.9}$$

dove W è un vettore d-dimensionale che rappresenta i coefficienti dell'iperpiano e b è una costante.

Senza perdere di generalità, si può anche assumere che due iperpiani simmetrici siano caratterizzati rispettivamente dalle espressioni:

$$Wx + b = +1 \tag{2.10}$$
$$Wx + b = -1 \tag{2.11}$$

dove il numero a secondo termine rappresenta la codifica di una delle due specifiche classi.

La distanza tra i due iperpiani può quindi essere calcolata semplicemente come:

$$\frac{2}{|| W ||} \tag{2.12}$$

Per poter garantire la giusta classificazione è necessario infine applicare delle condizioni, in modo che ogni punto sia assegnato alla giusta classe di appartenenza (+1 o -1); i vincoli applicati sono:

$$Wx_i + b \geq +1, \forall i : y_i = +1 \tag{2.13}$$
$$Wx_i + b \leq -1, \forall i : y_i = -1 \tag{2.14}$$

Lo stesso ragionamento può essere fatto anche nel caso in cui le due classi non siano linearmente separabili; in questo caso è necessario considerare dei *soft margins* aggiungendo un errore $\xi \geq 0$ che permetta al processo di addestramento di violare leggermente la condizione sul vettore di supporto.

Le macchine a supporto vettoriale trovano spesso un'applicazione nei problemi che richiedono una elevata accuratezza, ma soffrono nel momento in cui è necessaria velocità di classificazione maggiore.

2.3 Reti neurali

Come sottoinsieme dell'intelligenza artificiale, una *Artificial Neural Network (ANN)* è un grafo di unità connesse che rappresentano un modello matematico dei neuroni biologici. In base alla costruzione ed alla funzione insegnata alla rete, una ANN potrà essere utilizzata per implementare un classificatore [13].

Le singole unità, dette nodi o neuroni, sono collegate tramite archi unidirezionali o bidirezionali ed il peso di ciascuna connessione rappresenta la forza del legame esistente tra di loro.

Le reti neurali ricevono i dati di input attraverso un set specifico di nodi di ingresso e forniscono il relativo output da uno o più neuroni di uscita.
Tutte le altre unità sono responsabili della logica ed intervengono nella costruzione del modello matematico da applicare a tutti gli esempi in esame.

Tra gli aspetti principali con cui caratterizzare una neural network, si trovano:

- **Modello del neurone**: come ciascuna unità genera il proprio output;
- **Architettura**: come sono collegati tra loro i diversi neuroni e livelli;
- **Algoritmo di allenamento**: come vengono ottenuti i pesi ottimali.

A seconda della complessità del modello da implementare, la rete neurale potrà essere costituita da uno o più livelli: nel caso più semplice, i neuroni in ingresso saranno direttamente collegati ai nodi di uscita mentre, con dataset più avanzati, sarà necessario introdurre più livelli nascosti (*hidden layers*).

2.3.1 Modello matematico del neurone

Ogni neurone, come illustrato in Figura 2.16 applica una trasformazione ai dati in ingresso $x = (x_1, ..., x_d)$ e produce un'uscita o utilizzando la composizione di due funzioni [13]:

1. **net value function** ξ, che utilizza i parametri ed i pesi $w = (w_1, ..., w_d)$ per comprimere i dati in valori netti $v = \xi(x, w)$. La funzione ξ ha generalmente la forma di una somma di prodotti del tipo:

$$v = \xi(x, w) = w_1x_1 + w_2x_2 + ... + w_dx_d + b$$

2. **funzione di attivazione** Φ: trasforma il valore netto v nel dato in uscita dal singolo nodo $o = \Phi(v)$.

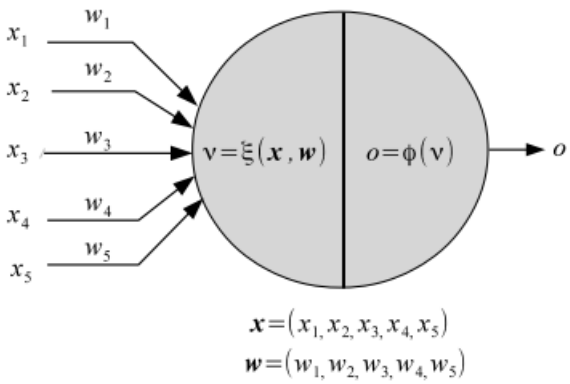


Figura 2.16: Modello matematico di una ANN

La funzione di attivazione decide se il segnale da trasmettere sull'uscita debba essere attivato o inibito, a seconda del tipo di funzione scelto e del valore di ciascun ingresso. Tra le funzioni usate più frequentemente ci sono:

- **Sigmoide** (funzione logistica);
- **Rectified Linear Unit** (ReLU);
- **Tangente iperbolica** (tanh).

Un dettaglio delle tre funzioni è riportato in Figura 2.17.

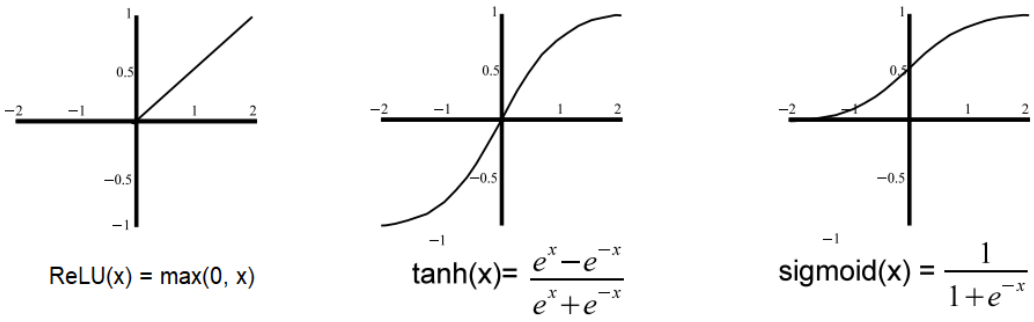


Figura 2.17: Funzioni di attivazione: ReLU, tangente iperbolica e sigmoide

In una rete neurale costituita da più livelli, ciascun layer può utilizzare una funzione di attivazione diversa e specifica per la trasformazione da applicare.

2.3.2 Architettura

La topologia della rete definisce la struttura del grafo con tutti i nodi e le connessioni esistenti tra un livello ed un altro.
Generalmente, ogni neurone è seguito da molti nodi successivi ai quali trasmettere il proprio output e fornire le informazioni su cui lavorare. A seconda del numero e della tipologia di ciascun livello è possibile quindi distinguere diverse categorie di reti neurali, come spiegato qui di seguito.

Single-Layer Neural Network

Gli unici livelli necessari alla definizione di una qualsiasi rete neurale sono quelli di input, fondamentale per ricevere i dati ed inviarli ai nodi interni, e di output, che fornisce l'uscita ed informa sulla decisione finale del modello.
Le reti *single-layer* sono caratterizzate solo da questi due layers e la loro semplicità può essere utilizzata per mostrare le modalità operative di una rete neurale.

Una delle principali implementazioni di questa architettura è rappresentata dai **Single-Layer Perceptron** (SLP): in questo modello, il layer di uscita è costituito da un singolo nodo che, utilizzando una soglia lineare, definisce la classe di ciascun esempio x in ingresso.
L'unità funzionale implementa la funzione *segno* come activation function e determina la classe di ciascun esempio analizzando il prodotto scalare $w^t\ x$.

Ad ogni esempio $x = \{x_1, \ldots, x_n\}$ è associata una etichetta $\{t_1, \ldots, t_n\}$ $(t_n \in \{-1, +1\})$; per ogni input x_n il percettrone genera un output $o_n \in \{-1, +1\}$.
Confrontando tra loro questi due valori, si vuole minimizzare il contributo delle classificazioni errate. Il cosiddetto *perceptron criterion* (E_p) permette di quantificare l'errore e può essere utilizzato per ottimizzare il processo di allenamento:

$$E_p = -\sum_{o_n \neq t_n} t_n w^t x_n \tag{2.15}$$

Il criterio utilizzato prevede di trovare l'iperpiano che meglio separa le due classi nell'origine: il prodotto scalare $-t_n w^t x_n$ è sempre positivo per casi di errata classificazione e, minimizzando il risultato di E_p è possibile ridurre il numero di errori commessi (in modo analogo alla discesa del gradiente della sezione 2.1.2).

Un processo analogo può essere utilizzato nel caso di scelte multiple con l'implementazione di **Multi-Class Perceptron**.

Multi-Layer Network

Le architetture più semplici, come le SLP presentate nella sezione precedente, sono fondamentali per la costruzione di reti più complesse, ma hanno dei nel caratterizzare dataset complessi.

L'aggiunta di livelli nascosti tra il layer di ingresso e quello di uscita permette di complicare il modello matematico rappresentato dall'ANN e può intervenire per:

1. facilitare il processo di modellazione dei dati e favorire la costruzione di una propria rappresentazione interna;
2. utilizzare funzioni di attivazione diverse: lavorando con unità differenti è possibile analizzare un numero maggiore di parametri e rendere l'intera discriminazione più accurata e precisa;
3. applicare un processo di *backpropagation*, che permette un allenamento gerarchico più semplice.

In un **Multi-Layer Perceptron** (MLP), i livelli interni sono organizzati in modo sequenziale e seguono l'andamento dei dati dall'ingresso all'uscita: ciascun nodo riceve il risultato della computazione del neurone precedente e trasmette il proprio output ad una unità successiva.

Questo processo iterativo descrive la cosiddetta *forward propagation*.

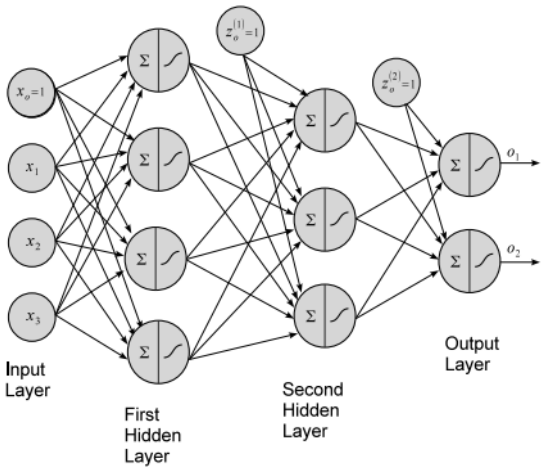


Figura 2.18: Multi-Layer Perceptron per una classificazione binaria

In Figura 2.18 è rappresentato un esempio di MLP applicato ad un problema di classificazione binaria; sono utilizzati in questo caso tre livelli, due interni ed uno di uscita.

Nel caso di un'applicazione multi-classe, con K possibili categorie, saranno utilizzati K nodi di uscita, ognuno dei quali corrisponderà e si attiverà in corrispondenza di una specifica scelta.

2.3.3 Algoritmo di allenamento

Il processo di allenamento ha lo scopo di trovare i valori ottimali per tutti i parametri utilizzati nella costruzione del modello. Per trovare l'iperpiano migliore in grado di discriminare con maggiore facilità e accuratezza gli esempi di ciascuna classe, si vuole minimizzare l'errore ed il numero complessivo di classificazioni errate [8].

Data una rete neurale con i relativi nodi, pesi e le diverse funzioni di attivazione, si vuole utilizzare i dati in ingresso per addestrare la macchina a predire automaticamente i valori dell'uscita, in modo che:

$$y_{output} \sim y_{target} \tag{2.16}$$

Per valutare lo stato di allenamento e misurare quanto si è lontani dal modello finale, è utilizzata la funzione di errore E:

$$E(y_{output}, y_{target}) = \frac{1}{2}(y_{output} - y_{target})^2 \tag{2.17}$$

Forward propagation

Nel caso di una *forward propagation*, l'algoritmo prevede di trasferire i dati dal layer di ingresso a quello di uscita; in modo iterativo, ogni esempio con la rispettiva etichetta (x, y_{target}), è preso ed utilizzato per aggiornare il valore iniziale dei diversi parametri.

Considerando i neuroni di ingresso come qualsiasi altro nodo interno (senza funzione di attivazione), l'uscita sarà uguale al dato di esempio campionato:

$$y_1 = x_{input} \tag{2.18}$$

Utilizzando queste informazioni, i valori in input a ciascun livello saranno i risultati delle computazioni dei livelli precedenti; ogni dato sarà aggiornato in funzione dei pesi e delle relative labels:

$$x_j = \sum_{i,j} w_{ij} y_i + b_j \tag{2.19}$$

La funzione di attivazione interviene su ogni unità per valutare il valore di uscita ad ogni layer:

$$y = \Phi(x) \tag{2.20}$$

Iterando le due equazioni 2.19 e 2.20 è possibile modificare gli iperparametri ed adattare tutte le variabili ai dati di allenamento. In questo caso, l'algoritmo prevede un'evoluzione progressiva dall'ingresso verso l'uscita, seguendo il normale movimento delle informazioni.

Backward propagation

In modo opposto al precedente, un algoritmo di *backward propagation* modifica gli iperparametri solo dopo aver confrontato l'output predetto con il valore atteso. Ciascuna modifica ai pesi ed alla funzione di costo sarà proporzionale alla distanza esistente tra il valore associato alla classe predetta e quello della classe target di ciascun esempio.

Per poter aggiornare i parametri, è necessario calcolare la variazione dell'errore E in funzione di ciascun peso $w = \{w_1, ..., w_n\}$.

$$\frac{dE}{dw_{ij}} \tag{2.21}$$

Una volta ottenuto il risultato della derivata, è possibile modificare ciascun peso tramite l'equazione 2.22:

$$w_{ij} = w_{i_j} - \alpha \frac{dE}{dw_{ij}} \tag{2.22}$$

dove il parametro α rappresenta il tasso di apprendimento (*learning rate*).
A questo punto, con la relazione espressa in 2.23 ha inizio la *backpropagation* delle derivate dall'uscita fino ad arrivare ai nodi di ingresso.

$$\frac{\partial E}{\partial y_{output}} = y_{output} - y_{target} \tag{2.23}$$

É quindi definita la *chain rule* per calcolare la variazione dell'errore in funzione di ciascun ingresso x:

$$\frac{\partial E}{\partial x} = \frac{\partial y}{\partial x}\frac{\partial E}{\partial y} = \frac{d}{dx}f(x)\frac{\partial E}{\partial y} \tag{2.24}$$

dove la derivata della funzione di attivazione f può essere approssimata con:

$$\frac{d}{dx}f(x) = f(x)[1 - f(x)] \tag{2.25}$$

In generale, è possibile ottenere le derivate rispetto ai pesi utilizzando i valori caratteristici di ogni esempio: utilizzando le features e le etichette fornite in ingresso, si possono scrivere le seguenti espressioni:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial x_j}{\partial w_{ij}}\frac{\partial E}{\partial x_j} = y_i\frac{\partial E}{\partial x_j} \tag{2.26}$$

$$\frac{\partial E}{\partial y_i} = \sum \frac{\partial x_{ij}}{\partial y_i}\frac{\partial E}{\partial x_j} = \sum w_{ij}\frac{\partial E}{\partial x_j} \tag{2.27}$$

Nonostante la *backward propagation* permetta un allenamento efficace sugli esempi di training, ci sono casi in cui l'algoritmo può fallire:

- **Vanishing gradients**: i gradienti calcolati nei livelli iniziali diventano troppo piccoli ed il processo di allenamento rallenta o viene interrotto;
- **Exploding gradients**: se i pesi sono troppo grandi, i gradienti rischiano di esplodere senza raggiungere la convergenza;
- **Dead ReLU units**: se l'uscita di una funzione ReLU scende a 0, il nodo può rimanere bloccato a quel valore ed i gradienti che lo attraversano durante la backpropagation si azzerano, non contribuendo all'allenamento dei pesi.

Parte III

Applicazione al caso di studio

Capitolo 3

Presentazione del lavoro

Questo lavoro di tesi si inserisce in un progetto già avviato ed affrontato da alcuni professori del Politecnico di Torino, insieme con dottorandi e tesisti che in passato hanno contribuito allo sviluppo dello stesso ambiente di ricerca usato ed aggiornato per la seguente attività.

Il progetto MIT-Food (*Microwave Imaging Technology*) nasce nel 2017 con l'obiettivo di creare un dispositivo per il monitoraggio della qualità dei prodotti alimentari, con particolare attenzione alla rilevazione dei contaminanti in maniera non distruttiva e totalmente sicura per gli operatori sulla catena produttiva.

La teoria alla base dell'*imaging a microonde*, già descritta al capitolo 1.2, viene applicata concretamente in questo progetto ed utilizzata in maniera pratica per ottenere una mappa spaziale delle proprietà elettromagnetiche del prodotto: insieme con le tecniche di machine learning presentate in precedenza, si è in grado di identificare le differenze intrinseche tra cibo e contaminate e di intervenire istantaneamente per eliminare eventuali impurità.

Prima di entrare nel dettaglio delle analisi effettuate, è opportuno descrivere brevemente alcuni aspetti generali come:

- Insieme dei dispositivi usati per l'ottenimento delle misure;
- Tipi di contaminanti utilizzati;
- Risultati ottenuti prima di questo intervento.

Una volta definito l'obiettivo della tesi e comprese le maggiori differenze rispetto i lavori precedenti, si può iniziare la descrizione delle varie caratteristiche e la presentazione dei nuovi risultati.

3.1 Dispositivi usati per l'acquisizione dei dati

Con l'avanzamento del progetto e l'investimento di nuovi fondi, l'apparecchiatura utilizzata nella fase di acquisizione dati è stata cambiata e progressivamente migliorata nel tempo.
In un'applicazione di machine learning, la scelta dei dispositivi impiegati nell'ottenimento delle misure è essenziale per poter ottenere buoni risultati e l'insieme degli esempi da usare in fase di allenamento deve essere quanto più possibile priva di errori e di rumore.

Come è possibile leggere nella pagina specifica del Politecnico [15], il principale successo del progetto MIT-Food è stato la realizzazione di un dispositivo prototipale funzionante (prima più semplice con due antenne, poi più complesso e preciso con sei antenne) in grado di identificare oggetti estranei in prodotti alimentari impacchettati usando l'imaging a microonde.

Tra tutti i dispositivi inseriti all'interno del sistema in esame, i componenti assolutamente necessari al raggiungimento di tale scopo sono:

1. **Antenne**;
2. **Analizzatore di rete vettoriale (VNA)**;
3. **Controllore**.

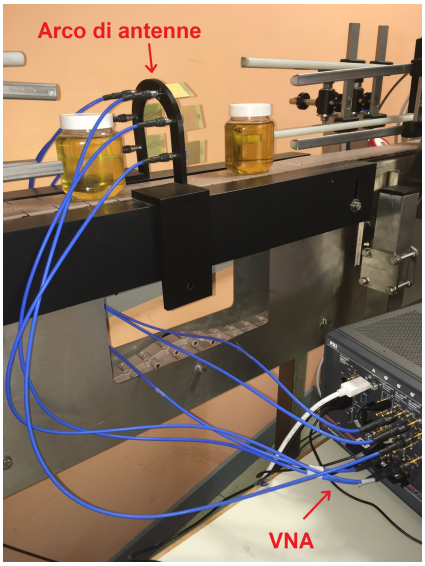


Figura 3.1: Panoramica del prototipo utilizzato per il microwave sensing

Al momento dello svolgimento di questa tesi, il sistema installato nella sede del Politecnico è quello rappresentato in Figura 3.1; nell'immagine è possibile vedere con particolare attenzione l'arco di antenne posizionato sul nastro di confezionamento dei barattoli ed il *vector network analyzer* con le sei porte di I/O per la ricetrasmissione di tutti i segnali.
Cercando di descrivere brevemente il funzionamento delle varie parti, è possibile analizzare i diversi contributi.

Antenne

Le sei antenne utilizzate nel prototipo "finale" sono disposte e connesse su un supporto circolare in resina da posizionare sulla linea di produzione del prodotto; facendo riferimento ad una parte del lavoro del dott. Ricci, documentato anche dal dott. Urbinati [2], possono essere riportare le principali proprietà del singolo componente.

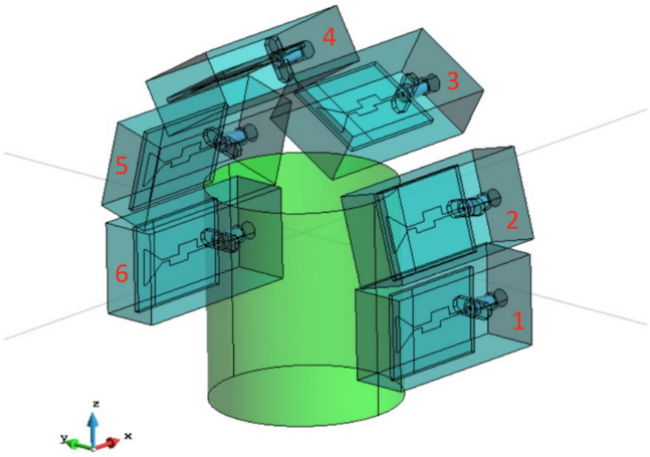


Figura 3.2: Arco di sei antenne intorno al barattolo

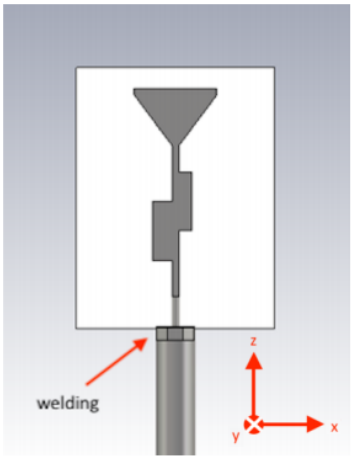


Figura 3.3: Antenna PCB

Come è possibile vedere dall'immagine 3.3, ogni antenna è progettata e costruita utilizzando una PCB ad apertura triangolare, facile ed economica da costruire; le dimensioni di ciascuna board sono di 4cm x 3cm ed i materiali costituenti sono rame (per il circuito) e FR4 (per il substrato).

Dalla simulazione a 10 GHz, la frequenza maggiormente utilizzata per la rilevazione di contaminanti in olio di cartamo (*sofflower oil*) e in crema di nocciole e cacao (*cocoa-hazelnut spread*), si possono leggere le caratteristiche più importanti dell'onda generata:

- Ampiezza del lobo principale: 3.77 dBi;
- Direzione del lobo principale: 233.0 deg;
- Larghezza angolare a 3 dB: 78.9 deg;
- Livello del lobo laterale: -0.5 dB.

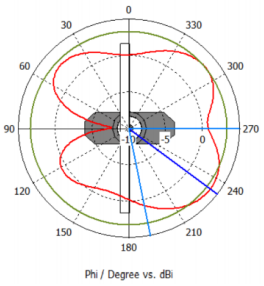


Figura 3.4: Diagramma di radiazione dell'antenna

In base alle necessità ed alla qualità dei dati che si vogliono ricavare, è possibile aggiungere al supporto in resina uno scudo metallico protettivo contro le onde elettromagnetiche provenienti dall'esterno; come sarà detto più avanti con la caratterizzazione dei diversi dataset, l'utilizzo di tale shield può teoricamente facilitare l'acquisizione delle misure e permettere di aumentare di conseguenza la velocità del nastro e l'efficienza della produzione.

Analizzatore di Rete Vettoriale (VNA)

Un *Vector Network Analyzer* è uno strumento utilizzato per analizzare e studiare le proprietà delle reti elettriche, con particolare attenzione alla riflessione e la trasmissione dei segnali [16].

Nel caso particolare del progetto MIT-Food, il VNA è utilizzato per inviare alle antenne il segnale da trasmettere e per ricevere indietro le onde riflesse dal barattolo; misurando la risposta del prodotto alimentare allo stimolo elettromagnetico, il dispositivo è in grado di calcolare i parametri di Scattering (*S-parameters*), gli stessi che saranno poi usati per allenare la rete neurale.

Nella versione iniziale del prototipo era utilizzato il *P9375A Keysight Streamline USB Vector Network Analyzer*, analizzatore caratterizzato da due porte di input/output in grado di lavorare in un range di frequenza tra 300 kHz e 26.5 GHz.

Per poter inserire tale strumento nel progetto e connettere quindi le sei antenne alle sole due porte del dispositivo, era necessario utilizzare una *Switching-matrix*, una matrice di interruttori essenziale per poter misurare i 30 parametri relativi a tutte le combinazioni disponibili (escludendo i valori di auto-diffusione delle singole antenne).

Come possiamo vedere dalla Figura 3.1, nella nuova configurazione del sistema, la stessa matrice non è più presente.
Con il cambiamento del VNA ed il passaggio ad uno strumento dotato di sei porte di I/O, è stato possibile realizzare collegamenti diretti tra l'arco di antenne e l'analizzatore, gestendo così i vari segnali semplicemente con il controllore.

Un parametro importante da monitorare durante tutte le fasi di lavoro, che sarà centrale nella definizione degli obiettivi di questa tesi, è il costo totale dell'apparecchiatura. Con l'aumento dei costi necessari all'acquisto degli strumenti, i benefici legati all'utilizzo del prototipo possono essere facilmente superati dalle spese e, di conseguenza, il progetto rischia di rimanere limitato alla ricerca senza trovare una realizzazione pratica e concreta nell'industria.

Valutando opportunamente i risultati che saranno ottenuti, si potrà verificare la possibilità di semplificare il circuito e, in caso affermativo, si cercherà di sostituire gli strumenti di acquisizione con altri più economici.

Controllore

La gestione ed il salvataggio dei dati è affidato ad un computer sul quale eseguiamo uno script Matlab; tra gli scopi principali del codice troviamo:

- Comandare il VNA per selezionare in ricezione un'antenna alla volta;
- Trasmettere il segnale con tutte le sei antenne;;
- Salvare ogni matrice di Scattering in un file di testo.

L'analizzatore riceve e misura ogni coefficiente di diffusione come un numero complesso, con parte reale e parte immaginaria; il codice originale [17], che lavora separando i due contributi (scrivendoli entrambi sul file di uscita), sarà modificato per calcolare il modulo di ciascun parametro ed usare solo questa informazione per allenare una rete neurale.

Dimezzando il numero di features che caratterizzano ogni esempio (rimuovendo ogni informazione riguardo alla fase del segnale), l'allenamento del classificatore sarà probabilmente più complesso ma, nel caso in cui portasse a buoni risultati, permetterebbe di sostituire l'analizzatore di rete con dispositivi notevolmente più semplici ed economici.

Con la sostituzione del VNA, il controllore è integrato nell'analizzatore di rete e lo script è eseguito all'interno del dispositivo stesso, rendendo di fatto il circuito complessivo più semplice e compatto.

3.2 Tipi di contaminanti

Per poter rilevare contaminati diversi nella forma e nel materiale all'interno dei prodotti alimentari, è necessario testare e verificare il comportamento del dispositivo in varie situazioni: per generalizzare l'apprendimento ed insegnare ad una rete neurale a distinguere le due classi attraverso l'analisi delle proprietà elettromagnetiche, diversi oggetti saranno utilizzati per creare il dataset di allenamento.

Più in dettaglio, le principali caratteristiche utilizzate per diversificare gli esempi tra di loro sono:

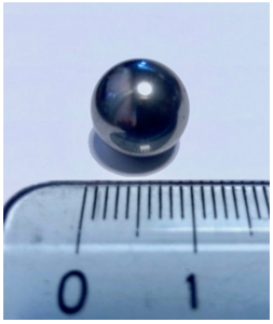
1. Materiale del contaminante;
2. Forma del contaminante;
3. Posizione del contaminante all'interno del barattolo.

Per quanto riguarda il terzo punto, indipendentemente dalle altre due qualità, ogni contaminante può collocarsi in posizioni diverse del contenitore, a partire dall'altezza (in superficie o sul fondo), dalla vicinanza ai bordi, fino ad arrivare all'orientamento che ogni oggetto ha rispetto agli assi del barattolo stesso.

Nella tabella 3.1 seguente e nella successiva immagine, possiamo vedere alcuni esempi di contaminanti utilizzati nella nostra analisi, con particolare attenzione alla dimensione ed al materiale di ciascun oggetto.

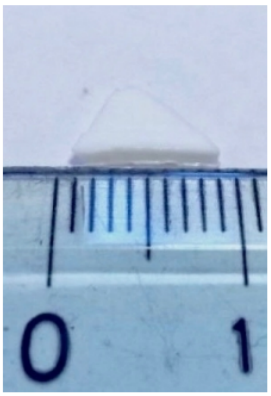
Tipo di contaminante	**Sigla**	**Dimensione (o diametro) minima (mm)**	**Dimensione (o diametro) minima (mm)**
Sfera di metallo	m	10	
Sfera di politetrafluoroetilene	ptfe	2	
Sfera di soda-lime-glass	slg	2	
Scheggia triangolare	t	1	8
Scheggia di vetro	g	2	13
Scheggia di legno	w		
Sfera di nylon	n		

Tabella 3.1: Tipi di contaminanti con dimensione e sigla usata nel dataset



(a) Sfera di metallo

(b) Scheggia di vetro



(c) Scheggia triangolare (frontale)

Figura 3.5: Esempi di contaminanti con dettaglio sulla dimensione

A seconda delle proprietà e del valore della costante dielettrica di ciascun materiale, il contributo di ogni contaminante è diverso rispetto agli altri e, per avere una copertura completa su tutte le possibilità, dovremo raccogliere sufficienti misure per poter allenare una rete neurale a riconoscerli adeguatamente.

3.3 Risultati precedenti ed obiettivi

Per poter proseguire con lo sviluppo del progetto e ripartire dai risultati già raggiunti, questa tesi ha ereditato il lavoro svolto da professori e dottorandi che già in precedenza avevano collaborato per ottenere i primi risultati verso la soluzione del problema.

Cercando di ricostruire i passaggi che hanno caratterizzato l'evoluzione del prototipo, possono essere riassunti i principali risultati documentati nelle varie tesi e negli articoli divulgativi.

Come detto nei capitoli precedenti, tutto il progetto si basa sulla ricerca di una soluzione al problema della contaminazione alimentare; in particolare, la soluzione desiderata deve essere in grado di lavorare in totale trasparenza e sicurezza sia dal punto di vista della produzione che della commercializzazione del prodotto stesso.

1) La prima fase è caratterizzata, come per ogni nuova idea, da uno studio approfondito volto a verificare la possibilità di risolvere il problema attraverso il metodo ipotizzato; in questo caso specifico, l'imaging a microonde si è dimostrato efficace nella rilevazione di corpi estranei all'interno dei barattoli di crema al cioccolato ed è stato per questo possibile proseguire il lavoro verso una nuova fase di sviluppo [18].

Insieme allo studio teorico, in questo primo passaggio è stata condotta una ricerca bibliografica per individuare tutti i progetti che abbiano in precedenza sfruttato la stessa tecnologia: come possiamo leggere nell'articolo *Urbinati [et al.]* [6], l'utilizzo della combinazione di *microwave sensing* e *machine learning* non era nuovo in campo biomedico, ma non era ancora mai stato applicato per la rilevazione dei contaminanti nei cibi impacchettati.

2) Dimostrata l'originalità del lavoro, è cominciata la fase di progettazione del dispositivo partendo da un'analisi del prodotto alimentare: sottoponendo il barattolo ed il suo contenuto a diversi campi elettromagnetici e registrando i dati associati, sono state valutate le condizioni con le quali ottenere le informazioni migliori (la frequenza dell'onda, la velocità del nastro lungo la catena di confezionamento, il numero di antenne necessario, etc.) [2].

In base ai parametri ottenuti, sono state disegnate le antenne PCB, il circuito di collegamento e sono stati effettuati i primi test per verificare il funzionamento del prototipo.

3) A questo punto si è potuto cominciare a ragionare in termini di automazione: le prime prove di machine learning sono state fatte allenando classificatori come Support Vector Machine e Multilayer Perceptron (vedi sezione 2.2) e valutando le prestazioni massime di ogni soluzione; si è quindi approfondito l'utilizzo delle Neural Networks, partendo da macchine con pochi neuroni, fino ad arrivare ad avere reti più complesse e specifiche per il dataset in esame [17].

I risultati ottenuti in questa fase sono stati soddisfacenti ed hanno aperto alla possibilità di esplorare nuove tecniche per ottimizzare il processo in tutti i suoi aspetti: in un momento in cui il prototipo finale del dispositivo è stato opportunamente studiato e testato, è emersa la necessità di semplificarne la costruzione, in modo da espanderne il mercato ed applicarlo più facilmente sulle reali catene di produzione.

Il lavoro di tesi si inserisce in questa ottica con l'obiettivo di verificare la possibilità di effettuare delle modifiche nel dataset per poter sostituire i componenti del circuito con altri più economici; aspetto primario di tutta l'attività è la necessità di non peggiorare le prestazioni della rete già ottenuta, ma piuttosto di raggiungere risultati simili utilizzando strumenti più piccoli e semplici.

Come detto precedentemente nel paragrafo relativo al *controllore*, la prima operazione da svolgere sarà quella di calcolare il modulo di tutti i parametri di scattering ed utilizzare questi valori per rilevare gli eventuali contaminanti nel prodotto alimentare; nel caso in cui i risultati saranno ancora soddisfacenti ed accettabili, si proverà a ridurre il numero di antenne collegate al circuito eliminando i rispettivi contributi da tutti i dati di allenamento.

I processi di training per le varie reti neurali saranno svolti seguendo un procedimento analogo tra tutti i dataset e comprendono i seguenti passaggi:

1. Analisi del dataset;
2. Ricerca bayesiana per ottimizzare gli iperparametri;
3. Allenamento della rete e valutazione delle metriche;

Gli stessi tre step saranno quindi ripetuti più volte fino ad ottenere in uscita una macchina in grado di discriminare con sufficiente precisione i barattoli contaminati da quelli sicuri.

Capitolo 4

ML in Python

Lo sviluppo di una macchina intelligente, così come quello delle più specifiche reti neurali, può essere di base svolto con qualsiasi linguaggio di programmazione, descrivendo ed implementando le funzioni teoriche presentate nei capitoli precedenti.

Tra tutti i linguaggi, *Python* è diventato negli ultimi anni uno dei principali punti di riferimento per l'utilizzo dei modelli di machine learning.

Grazie alla semplicità ed alla velocità con la quale può essere scritto il codice e grazie anche alla grande disponibilità di librerie e funzioni già implementate, il lavoro del programmatore è notevolmente ridotto e le macchine finali sono in grado di raggiungere facilmente ottime prestazioni.

Nelle due sezioni seguenti sono presentati e brevemente descritti i pacchetti più importanti per le applicazioni di ML, così come l'ambiente di sviluppo progettato ed utilizzato nel corso della tesi.

4.1 Principali librerie e pacchetti

Python, come linguaggio di programmazione, trova applicazione in numerosi ambiti e permette, grazie alla sua versatilità, di essere impiegato ampiamente anche in campo scientifico.

In particolare, dopo lo sviluppo di funzioni e librerie specifiche, Python è diventato uno dei migliori candidati per effettuare analisi di dati, manipolazioni algebriche, grafici e svolgere simulazioni di modelli e sistemi.

Tra i principali pacchetti che è possibile utilizzare per lavorare con dataset ed allenare macchine intelligenti, quelli applicati al progetto in esame sono:

- Numpy;
- Pandas;
- Scikit-Learn;
- TensorFlow;
- Keras.

Entrando nel dettaglio, è possibile evidenziare le caratteristiche di ognuno e comprenderne le funzioni all'interno del progetto.

4.1.1 Numpy

La prima libreria qui presentata è *Numpy*, un pacchetto fondamentale per computazioni algebriche ed operazioni aritmetiche ad alto livello [19].

Il contributo principale di Numpy alle classiche funzioni Python è dato dalla possibilità di creare e lavorare in modo efficiente con matrici e vettori multidimensionali (tensori), facilitando l'analisi e l'elaborazione di strutture dati complesse.

La documentazione ufficiale permette di avere una panoramica generale sulle procedure introdotte dalla libreria e descrive alcuni dei vantaggi dell'utilizzare tali funzioni in applicazioni scientifiche.

Le capacità computazionali di Numpy sono associate ad una *vettorizzazione* ottimizzata e precompilata; nello specifico, questa permette nello specifico di:

- eliminare la presenza di loop espliciti ed indici;
- avere un codice più piccolo e leggibile;
- seguire fedelmente le notazioni aritmetiche;
- avere una elaborazione più veloce ed efficiente.

All'interno del progetto di tesi, Numpy ha un ruolo fondamentale in tutte quelle le operazioni che riguardano la manipolazione dei dati ed è stato uno strumento fondamentale per la conversione dei dati acquisiti in questa tesi per renderli utilizzabili dagli algoritmi di ML.

Partendo dai file di testo in cui sono salvati i diversi coefficienti di diffusione, è necessario costruire il dataset di lavoro e dividere tra di loro, in forma vettoriale, tutti gli esempi che saranno utilizzati per l'addestramento e la validazione della rete neurale.

Grazie alla compatibilità di questo pacchetto con le altre librerie, è quindi facilitata l'analisi e la gestione di tutte le matrici di scattering acquisite dal sistema.

4.1.2 Pandas

Così come Numpy, *Pandas* è una libreria software dedicata all'analisi ed alla manipolazione di dati; sviluppata per essere compatibile con codici Python e C, facilita la costruzione di DataFrames e permette di effettuare su di essi tutte le principali operazioni necessarie alla risoluzione del problema [20].

Le due strutture dati principalmente usate con Pandas sono le *serie*, unidimensionali ed equivalenti a vettori Numpy, e i *DataFrames*, matrici bidimensionali in cui è possibile inserire informazioni in modo tabulare ed eterogeneo.

Avendo a disposizione il set di esempi da utilizzare per addestrare la rete neurale è possibile distinguere le features dalle labels ed effettuare su ciascun insieme tutte le operazioni utili all'ottimizzazione del processo di allenamento e di validazione.

All'interno del programma di tesi, le funzioni di questa libreria sono state utilizzate principalmente in due casi:

1. analisi dei dataset e della distribuzione dei punti;

2. costruzione delle matrici di confusione.

Per il primo caso, è stato sviluppato su *Jupyter* un *notebook* specifico dedicato all'analisi ed allo studio dei dataset: prima di allenare qualunque rete neurale e poter procedere quindi con la reale applicazione di machine learning, è sempre utile analizzare i dati per evidenziarne le caratteristiche e le proprietà.

Sfruttando la struttura tabulare offerta da Pandas, è possibile manipolare ogni coefficiente per massimizzarne il peso all'interno del dataset, in modo da facilitare successivamente la classificazione di ogni esempio.

4.1.3 Scikit-Learn

Progettata per lavorare con i pacchetti Numpy e MatPlotLib, *Scikit-Learn* è la libreria principale per operare nel campo dell'intelligenza artificiale [21].

A differenza di altri moduli in cui i modelli di ML devono essere descritti dal programmatore specificando per ognuno gli algoritmi e le funzionalità, con Scikit-Learn è possibile utilizzare delle funzioni built-in per definire facilmente la macchina desiderata.

La libreria fornisce nello specifico un'ampia scelta di algoritmi di apprendimento supervisionato e non-supervisionato e permette l'implementazione di macchine per *classificazioni*, *regressioni* e *clustering*.

Tra tutte le funzionalità di Scikit-Learn, quelle più importanti ed essenziali sono:

1. **creazione del dataset** da utilizzare con il modello di ML;
2. **definizione del modello** e selezione dell'algoritmo di allenamento;
3. applicazione di tecniche di **preprocessamento** dei dati;
4. definizione delle **metriche** per la valutazione della macchina.

All'interno del progetto di tesi, tutto il processo di costruzione ed utilizzo della rete neurale è svolto utilizzando le procedure delle classi di questa libreria.

Dalla divisione degli esempi nei tre gruppi per il training, la validation ed il test, all'applicazione di scalatori e tecniche di feature selection, fino al calcolo delle prestazioni della macchina finale, tutte le funzioni sono importate come API di Scikit-Learn per essere facilmente utilizzate e analizzate.

Attraverso l'utilizzo del pacchetto *Scikit-Optimize*, è possibile inoltre integrare degli algoritmi di ottimizzazione sequenziale: costruendo una *pipeline* con la rete neurale da allenare, si possono inserire nella catena di addestramento dei blocchi e delle funzioni in grado di ottimizzare il risultato finale.

Tra i principali meccanismi aggiunti da questa libreria, si possono sicuramente trovare gli algoritmi di **ricerca bayesiana** e di **validazione incrociata** per la ricerca dei valori ottimali per ciascun iperparametro.

4.1.4 TensorFlow & Keras

Entrambe queste librerie sono specifiche per applicazioni di machine learning e sono particolarmente note per la loro applicazione al *Deep Learning* e per la varietà di funzioni e modelli che sono in grado di fornire [22][23].

Keras è una piattaforma Python dedicata al ML, la cui attenzione è concentrata sugli algoritmi di Deep Learning e Neural Networks.
Lo sviluppo e l'esecuzione stessa delle funzioni Keras sono basati sulle API (*Application programming interface*) di *TensorFlow*, una seconda libreria end-to-end a più basso livello che implementa le procedure principali per la corretta implementazione dei modelli.

All'interno di un progetto Python, le funzioni di questi pacchetti sono fondamentali per la definizione della rete neurale, dalla descrizione dell'architettura fino ad arrivare alle fasi di allenamento e verifica.
In particolare, sono implementati classi e metodi per:

- creare la **struttura sequenziale** costituita da un numero arbitrario di livelli;
- **aggiungere nodi** e neuroni ad ogni livello;
- specificare per ogni layer la **funzione di attivazione** ed il livello di **regolarizzazione** da applicare;
- specificare l'**algoritmo di ottimizzazione** e compilare la rete.

Le stesse funzioni di *fit* (per iniziare il processo di addestramento con le matrici del training set) e di *predict* (per stimare la classe di ciascun esempio) sono fornite da Keras che, utilizzando l'algoritmo scelto in fase di costruzione, permette di ottenere il modello finale.

4.2 Ambiente di sviluppo

L'intero progetto MIT-Food è stato sviluppato in Python utilizzando applicazioni web e software dedicati per avere il massimo controllo di ciascun comando e funzione. Il codice utilizzato in questa tesi per descrivere ed allenare tutte le reti che saranno presentate nei capitoli successivi è stato modificato ed ottimizzato a partire dal lavoro di tesisti e dottorandi che in precedenza hanno contribuito al progetto [15][2][17].

L'ambiente di lavoro è costituito principalmente da due parti, un *envirnoment* nel quale sono sono installate tutte le librerie necessarie all'applicazione ed un *editor* con cui sarà scritto il codice da eseguire. Questi sono rispettivamente:

- **Anaconda**;
- **Jupyter Notebook**.

Anaconda è una delle distribuzione Python più usate al mondo, il cui scopo è quello di organizzare e semplificare la gestione delle librerie per applicazioni di Data Science [24].

A favorire l'utilizzo di tale piattaforma è la presenza di un *Conda package manager*, un gestore di pacchetti in grado di rendere l'installazione e l'applicazione degli stessi più veloce e trasportabile tra diversi calcolatori.

La disponibilità delle principali librerie per il ML (Numpy, Scikit-Learn, TensorFlow, ...) permette di creare in modo sicuro un ambiente di sviluppo per ogni applicazione, permettendo la scelta delle versioni desiderate e semplificando la risoluzione delle dipendenze.

Essendo i pacchetti preconfigurati all'interno della distribuzione stessa di Anaconda, la loro installazione è opportunamente controllata ed anche le *dependencies*, relative a conflitti ed errori, sono ridotte e gestite automaticamente.

La possibilità di creare ambienti indipendenti in cui poter importare librerie differenti, permette di lavorare contemporaneamente ad un numero maggiore di progetti e di creare conflitti tra versioni differenti degli stessi software installati sulla macchina.

Insieme con il software dell'applicazione, oltre alla linea di comando, è disponibile un'interfaccia grafica (*Anaconda Navigator*) che permette all'utente di gestire facilmente i pacchetti e di lanciare i programmi per sviluppare il proprio codice. Tra le applicazioni presenti di default nel sistema, si trovano:

- **JupyterLab**;
- **Jupyter Notebook**;
- QtConsole;
- Spyder.

Tra tutti gli editor di testo di possibile utilizzo, quello scelto nel corso di questa tesi è *JupyterLab* nella sua versione *Notebook* [25].

L'ambiente web offerto da Jupyter permette di sviluppare in modo interattivo documenti che comprendono codice (scegliendo tra molti linguaggi di programmazione), equazioni, immagini e testo narrativo.
La sua architettura è pensata per facilitare la condivisione dei file e degli strumenti all'interno di un gruppo, in modo che un'unica installazione su server sia utilizzabile da più utenti senza il bisogno di ripetere la procedura.

A differenza di un normale script ottenibile con qualsiasi altro IDE, il notebook sviluppato con Jupyter gode di alcune proprietà che lo rendono particolarmente utile nel caso di applicazioni di ML.

1. Prima tra tutti l'**organizzazione e l'esecuzione del codice**: mentre solitamente il programma Python è avviato interpretando insieme tutto il codice scritto nel file, utilizzando l'impostazione del notebook è possibile studiare gli effetti di un numero limitato di righe e porzioni.
   Dividendo il programma in celle la cui funzione è diversa e specifica rispetto alle altre, è possibile eseguire il programma a blocchi ed analizzare i risultati di ognuno senza il bisogno di commentare tutte le altre sezioni.

2. Attraverso l'integrazione di celle di natura diversa, è possibile **documentare l'intero programma** utilizzando sezioni di testo apposite; sostituendo i normali commenti Python ad aree di testo personalizzabili, la lettura e la comprensione del codice sono notevolmente migliori.

3. Pur essendoci la necessità di definire ed aggiornare le variabili globali affinché i risultati di ogni cella siano ritenuti validi, una volta eseguito il codice relativo ad ogni singolo blocco, il notebook è in grado di **salvare i valori di uscita** per mantenerli nella cronologia e poterli analizzare successivamente.

Nello specifico di questo progetto di tesi, sono stati sviluppati tre notebook, uno generale per l'intera applicazione ed altri due specifici per alcune fasi come l'analisi dei dati e la ricerca bayesiana dei migliori iperparametri.

A. Per ogni dataset, è necessario affrontare uno studio preliminare sugli esempi a disposizione e scoprire le caratteristiche utili alla classificazione; in questo senso, è stato sviluppato un notebook apposito per l'analisi del dataset.

   In questo primo script sono inserite ed applicate funzioni per:

   – **inizializzare le variabili** e caricare il dataset sotto forma di spreadsheet;

   – **avere una panoramica** su ciascun coefficiente: per ogni feature è possibile contare il numero di esempi ed avere informazioni riguardanti parametri come *media*, *deviazione standard*, *valore minimo* e *massimo*.
   Questo studio può essere condotto sul dataset completo o distinguendo l'analisi sulle due classi in modo indipendente.

   – eseguire un **algoritmo di PCA** (*Principal Component Analysis*): oltre ad essere utile per ridurre il numero di features ed estrarre dal dataset solo i coefficienti che maggiormente intervengono nella classificazione, diventa uno strumento fondamentale per la visualizzazione dei dati.
   Limitando ogni esempio a sole 2 o 3 componenti, è possibile rappresentare graficamente il dataset distinguendo con colori diversi ciascuna delle due classi.

   – applicare ulteriori tecniche di **feature selection** e **feature extraction**: nel tentativo di ottimizzare le prestazioni delle future reti neurali, sono state previste delle celle con cui poter modificare il dataset. Attraverso un'**analisi della varianza** (*ANOVA*) è possibile selezionare solo alcune caratteristiche originali mentre una **funzione polinomiale** può essere eseguita per aggiungere nuove features agli esempi iniziali.

B. Il notebook generale implementa tutte le funzioni necessarie alla costruzione ed all'analisi delle reti neurali.
   I passaggi principali che definiscono l'evoluzione del programma sono i seguenti e permettono una verifica puntuale per ogni fase operativa:

   1. **inizializzazione delle variabili** e definizione delle funzioni generali;

   2. **lettura dei file di testo** per importare il dataset da utilizzare per l'addestramento e la verifica: tutti gli esempi contenuti nei file txt sono divisi e convertiti in un formato compatibile con le librerie di ML;

3. **divisione del dataset** e **preprocessamento** dei dati: il dataset originale viene diviso nei tre gruppi per l'addestramento, la validazione e la verifica; tutti gli esempi sono quindi scalati prima di poter essere utilizzati;
4. per ogni iperparametro sono ricercati i valori ottimali: un **algoritmo di ricerca bayesiana** è utilizzato per valutare la migliore combinazione di parametri a partire da uno spazio di ricerca scelto dall'utente;
5. **definizione del modello** ed **allenamento**: specificando l'architettura della rete e gli iperparametri, è richiamata la funzione di *fit* per avviare l'addestramento e la validazione sui rispettivi set di misure;
6. **verifica delle prestazioni** su tutti i dataset: per ogni gruppo di misure sono calcolate le metriche e rappresentate le matrici di confusione con il numero di classificazioni errate.

Per completare lo sviluppo della rete neurale è quindi possibile salvare il modello (in particolar modo i pesi che hanno determinato i risultati migliori) per poterlo utilizzare nelle future applicazioni.

C. Per ottimizzare il lavoro ed automatizzare la ricerca del modello migliore, un terzo notebook è sviluppato per iterare il processo di allenamento: utilizzando dei loop per definire la rete e richiamare la funzione di addestramento, è possibile ripetere questa fase un numero arbitrario di volte e salvare, alla fine dell'esecuzione, solo la macchina che ha riscontrato le prestazioni migliori.

Capitolo 5

Primo dataset

5.1 Analisi del dataset

Il primo dataset colleziona tutti i coefficienti di scattering misurati per le prime fasi del progetto ed è caratterizzato da molteplici rilevazioni ottenute a frequenze diverse, da 9 GHz a 11 GHz; per il caso specifico sono stati analizzati ed utilizzati i dati relativi a 10 GHz.

Per poter svolgere un buon lavoro nel campo del machine learning è utile comprendere il significato delle features all'interno di ogni esempio ed essere in grado di riconoscere con precisione le caratteristiche utili da quelle superflue, in modo da facilitare l'allenamento e l'utilizzo della rete.
I parametri principali che definiscono la fase di acquisizione di queste misure sono:

- Frequenza: 10 GHz;
- Velocità del nastro: 15 cm/s;
- Numero di esempi: 1240
- Numero di features per esempio: 30

E' possibile dividere tutti gli esempi in base all'etichetta associata ad ognuno, riconoscendo i casi in cui il barattolo contiene il contaminante.
Per questa analisi possiamo beneficiare delle funzioni Python e possiamo riassumere le due classi nelle tabelle seguenti.

Le misure relative ai casi senza contaminante sono caratterizzate dal valore target *0*, contenute e caricate nella cartella locale *0_free_10_0*.

	S12	S13	S14	S15	S16	S21	...	S64	S65	target
count	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	...	600.000000	600.000000	600.0
mean	0.016355	0.013870	0.014157	0.017648	0.016902	0.018296	...	0.019792	0.010618	0.0
std	0.000984	0.000706	0.001411	0.000862	0.000960	0.001192	...	0.000785	0.000452	0.0
min	0.013916	0.012045	0.009766	0.016148	0.014890	0.014705	...	0.015993	0.009727	0.0
25%	0.015686	0.013367	0.013315	0.016980	0.016217	0.017622	...	0.019442	0.010271	0.0
50%	0.016403	0.013827	0.014005	0.017493	0.016963	0.018249	...	0.019793	0.010544	0.0
75%	0.016915	0.014402	0.014851	0.018133	0.017545	0.018756	...	0.020281	0.010932	0.0
max	0.019740	0.015493	0.017538	0.020328	0.019980	0.023248	...	0.021975	0.011980	0.0

Tabella 5.1: Descrizione degli esempi senza contaminante - primo dataset

Come è possibile notare in tabella 5.1, questa categoria conta un totale di 600 esempi ed ognuno di questi rappresenta una specifica matrice di diffusione ottenuta misurando i contributi di tutte e sei le antenne.
La stessa tabella può essere utilizzata per confrontare tra loro l'andamento dei coefficienti nelle due classi distinte: rappresentando la riga *mean*, che descrive il valor medio di una particolare caratteristica, è possibile confrontare l'andamento dei due dataset per provare ad intuire una soglia con cui riconoscere facilmente il valore target.

Un dettaglio delle prime cinque righe del dataset, con i valori dei rispettivi coefficienti, può essere visto con la tabella seguente; dallo stesso elenco è possibile analizzare la costituzione del set in cui ogni esempio è caratterizzato da tutti i parametri S_{ij} e dal relativo valore target.

	S12	S13	S14	S15	S16	S21	...	S64	S65	target
0	0.017198	0.013115	0.011536	0.019362	0.018557	0.021753	...	0.020143	0.011035	0
1	0.017333	0.013324	0.011039	0.019103	0.018657	0.022413	...	0.020767	0.011001	0
2	0.019082	0.014224	0.017120	0.019413	0.016334	0.015992	...	0.016571	0.010548	0
3	0.017239	0.014024	0.012693	0.018091	0.017299	0.019033	...	0.019208	0.010639	0
4	0.015111	0.014387	0.014078	0.017272	0.017549	0.017772	...	0.020282	0.010758	0

Tabella 5.2: Primi cinque esempi senza contaminante - primo dataset

Per quanto riguarda i casi contaminati, l'analisi si occupa anche di distinguere il tipo specifico di materiale contenuto nel barattolo; la cartella *1_contaminant_10_0* contiene gli esempi interessati, caratterizzati dall'etichetta *1*.

	S12	S13	S14	S15	S16	S21	...	S64	S65	target
count	640.000000	640.000000	640.000000	640.000000	640.000000	640.000000	...	640.000000	640.000000	640.0
mean	0.016563	0.013745	0.013974	0.017609	0.016760	0.019153	...	0.020109	0.010548	1.0
std	0.000927	0.000740	0.001365	0.000779	0.000874	0.000912	...	0.000719	0.000423	0.0
min	0.013878	0.011841	0.010032	0.016152	0.014995	0.016065	...	0.016795	0.009651	1.0
25%	0.015987	0.013255	0.013132	0.017033	0.016053	0.018624	...	0.019607	0.010210	1.0
50%	0.016566	0.013746	0.013697	0.017509	0.016829	0.019169	...	0.020223	0.010545	1.0
75%	0.017126	0.014210	0.014388	0.018077	0.017340	0.019589	...	0.020663	0.010847	1.0
max	0.021654	0.015457	0.018059	0.019863	0.019453	0.022777	...	0.021634	0.011559	1.0

Tabella 5.3: Descrizione degli esempi con contaminante - primo dataset

In questo caso sono disponibili 640 esempi, ancora definiti da 30 features diverse; nel dettaglio, queste misurazioni sono così suddivise:

Contaminante	**Sigla**	**Numero**
Frammento di vetro	g	120
Sfera di soda-lime-glass	l	100
Sfera di metallo	m	100
Sfera di politetrafluoroetilene	p	100
Scheggia triangolare	t	120
Scheggia di legno	w	100

Tabella 5.4: Numero di esempi con ciascun contaminante - primo dataset

Le prime cinque matrici all'interno di questo set sono riportate nella tabella seguente, dove è ben evidente il valore dell'etichetta target.

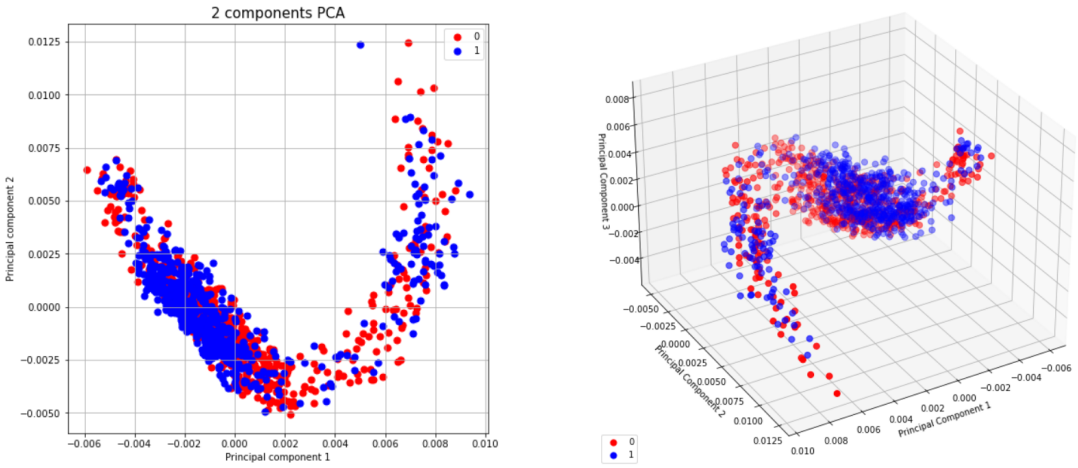
	S12	S13	S14	S15	S16	S21	...	S64	S65	target
600	0.015971	0.014408	0.017147	0.018892	0.015305	0.018351	...	0.019400	0.010014	1
601	0.016693	0.014937	0.017877	0.019029	0.015812	0.017880	...	0.019075	0.010070	1
602	0.018897	0.015196	0.016518	0.019826	0.017722	0.017414	...	0.018371	0.010335	1
603	0.016718	0.014355	0.014011	0.017139	0.017219	0.018740	...	0.020430	0.011058	1
604	0.016313	0.013472	0.013732	0.017182	0.017565	0.018373	...	0.021072	0.011221	1

Tabella 5.5: Primi cinque esempi con contaminante - primo dataset

Grazie alla *Principal Component Analysis* (PCA, discussa al capitolo 4), è possibile comprimere le 30 features di ciascun esempio su un numero ridotto di caratteristiche, in modo da poter visualizzare in modo grafico la disposizione dei punti in un piano n-dimensionale.

Rappresentando in rosso le misure del barattolo senza contaminante ed in blu quelle con l'oggetto estraneo, otteniamo i grafici in Figura 5.1.

La rete neurale che andremo ad allenare dovrà essere in grado di trovare una funzione capace di discriminare tra loro i due insiemi, andando a riconoscere il piano iperdimensionale che meglio permette di dividere le diverse caratteristiche.



(a) 2D PCA (b) 3D PCA

Figura 5.1: Principal Component Analysis su 2 e 3 dimensioni - primo dataset

Prima di iniziare l'allenamento della rete, è conveniente dividere l'intero dataset nei tre sotto gruppi per il training, la validazione ed il test dei risultati; per mantenere lo stesso rapporto tra numero di casi contaminati e casi sicuri, è invocata la funzione di *train_test_split* all'interno di ciascun set e sono ottenuti i seguenti insiemi:

Set	**Numero di esempi**	**Classe 0**	**Classe 1**
Training set	651	48.39 %	51.6 %
Validation set	279	48.39 %	51.6 %
Test set	310	48.39 %	51.6 %

Tabella 5.6: Divisione e distribuzione dei tre set - primo dataset

Solo per questo primo dataset sarà disponibile un altro gruppo di misure, denominato *extra_set*, caratterizzato da 100 rilevazioni di casi con contaminante. Le modalità di acquisizione sono le stesse già presentate, ed i vari esempi sono così suddivisi:

Contaminante	**Sigla**	**Numero**
Frammento di vetro	g	40
Sfera di nylon	n	20
Sfera di politetrafluoroetilene	p	40

Tabella 5.7: Numero di esempi con ciascun contaminante - extra set

Queste misure saranno utilizzate solamente in fase di verifica e permetteranno, insieme con il test set, di valutare le reali prestazioni della macchina attraverso esempi mai visti durante l'allenamento (favorendo così la generalizzazione).

La ricerca bayesiana testerà gli iperparametri sul training set ed i migliori saranno poi utilizzati per allenare il modello per le future previsioni.

Nelle pagine seguenti sono riportati nel dettaglio solo alcuni dei modelli ottenuti durante le diverse analisi, solo quelli che hanno portato ad un miglioramento dei risultati nel corso del lavoro; con gli iperparametri ottenuti dopo ogni ricerca bayesiana sono stati effettuati diversi processi di allenamento, ma ogni modello è stato indipendentemente testato prima di essere effettivamente valutato e salvato.

5.2 Primo modello

5.2.1 Ricerca bayesiana

La teoria e l'implementazione Python della ricerca bayesiana, trattata al capitolo 4, è qui utilizzata per cercare la combinazione migliore tra gli iperparametri inseriti all'interno dello *spazio di ricerca*.

Questa prima analisi è applicata su un *search space* ampio, che cerca come prima cosa di definire un range di valori per ogni iperparametro: tra le principali caratteristiche della rete, la ricerca si interessa ad aspetti come il numero di livelli che il modello deve avere, il numero di neuroni per ogni layer ed i coefficienti di regolarizzazione L2 e dropout.

I possibili valori sono inizialmente definiti come potenze di 2, mentre saranno selezionati in modo più accurato nelle ricerche successive.
Lo spazio di ricerca in questo caso è così costruito, specificando per ogni variabile una lista di valori tra cui scegliere ad ogni iterazione:

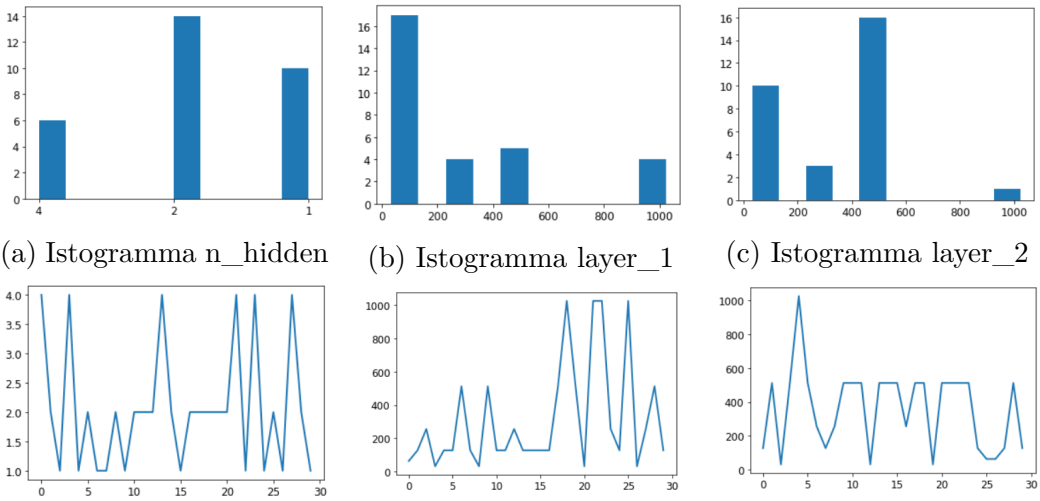
kerasclassifier___n_hidden	:	['1', '2', '4']
kerasclassifier___n_neurons_hidden_1	:	['32', '64', '128', '256', '512', '1024']
kerasclassifier___n_neurons_hidden_2	:	['32', '64', '128', '256', '512', '1024']
kerasclassifier___n_neurons_hidden_3	:	['32', '64', '128', '256', '512', '1024']
kerasclassifier___n_neurons_hidden_4	:	['32', '64', '128', '256', '512', '1024']
kerasclassifier___epochs	:	[700]
kerasclassifier___batch_size	:	[128]
kerasclassifier___learning_rate	:	['0.001']
kerasclassifier___regularization_rate	:	['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate	:	['0.0', '0.1', '0.3']

Con 30 iterazioni e 3 livelli di cross-validation, i risultati ottenuti sono i seguenti:

kerasclassifier___n_hidden	:	2
kerasclassifier___n_neurons_hidden_1	:	128
kerasclassifier___n_neurons_hidden_2	:	256
kerasclassifier___learning_rate	:	0.001
kerasclassifier___regularization_rate	:	0.0
kerasclassifier___dropout_rate	:	0.3

Come è possibile vedere, la ricerca bayesiana consiglia di utilizzare 2 livelli interni con rispettivamente 128 e 256 neuroni (valori simili a quelli ottenuti in [2]).

Per analizzare come i parametri siano stati modificati ad ogni iterazione, si può visualizzare i grafici sotto riportati.



(a) Istogramma n_hidden (b) Istogramma layer_1 (c) Istogramma layer_2

(d) n_hidden (e) n_layer_1 (f) n_layer_2

Figura 5.2: Risultati della prima ricerca bayesiana - modello 1, primo dataset

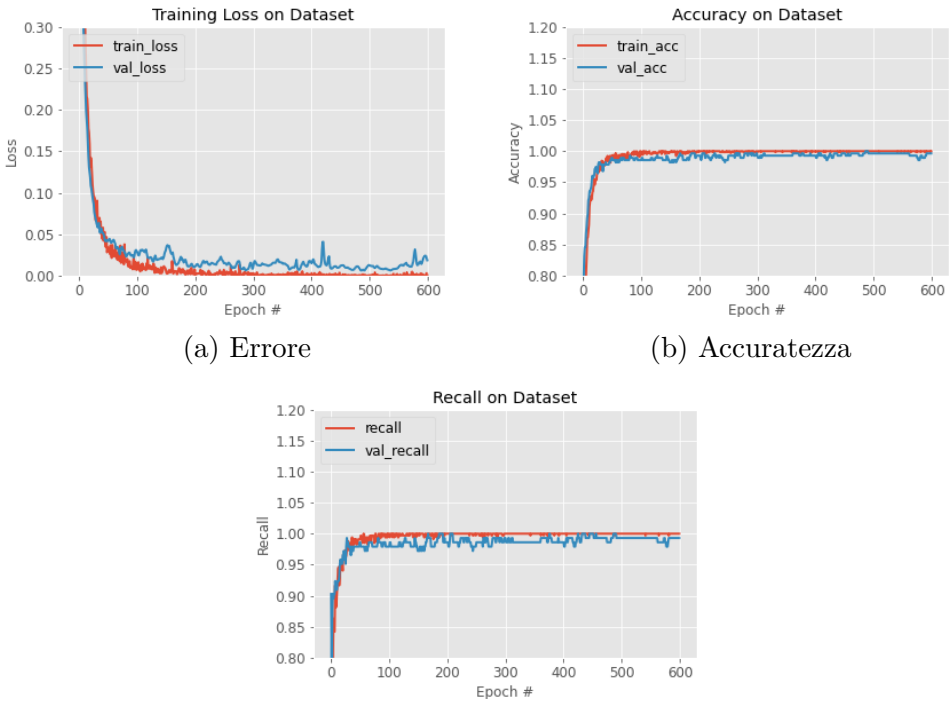
Le tre immagini 5.2a, 5.2b e 5.2c raffigurano gli istogrammi relativi al numero di *hidden layers* e di neuroni nei primi due livelli; grazie a questi possiamo verificare la correttezza dei risultati confrontando l'altezza delle varie barre, ognuna delle quali rappresenta il numero di volte che ciascuna opzione è stata utilizzata per completare un'iterazione della ricerca bayesiana.

Costruendo una rete neurale con i parametri sopra citati, è possibile iniziare la fase di allenamento e validazione sui rispettivi dataset; al termine del processo il modello più accurato sarà utilizzato per fare l'inferenza sugli esempi di test.

5.2.2 Allenamento e validazione

Come già fatto per la ricerca bayesiana, anche in questo caso è riportato il dettaglio del solo modello migliore, definito e costruito sugli iperparametri trovati nel paragrafo precedente.

Lanciando l'allenamento su 600 epoche e confrontando i vari risultati su alcune delle possibili metriche, la rete finale è così caratterizzata:



(a) Errore

(b) Accuratezza

(c) Richiamo

Figura 5.3: Evoluzione delle metriche - modello 1, primo dataset

La Figura 5.3 mostra l'evoluzione dei valori di errore, accuratezza e richiamo durante tutto il processo e ci permette di intuire in modo rapido se l'allenamento ha avuto successo o meno; in particolare, ogni grafico mette a confronto due curve: la linea rossa rappresenta l'andamento della metrica sul training set, mentre la linea blu descrive lo stesso comportamento sugli esempi di validazione.

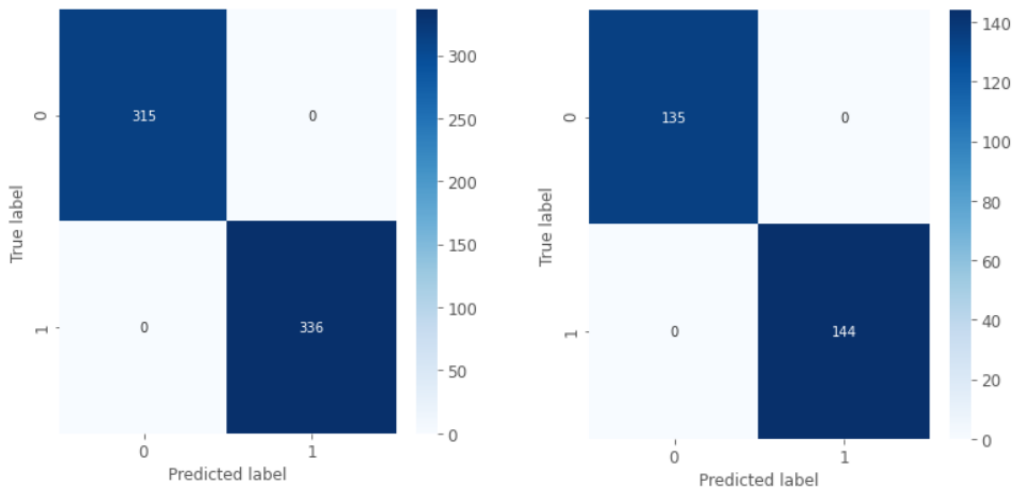
Analizzando nel dettaglio ciascuna funzione:

- **Loss function**: come è possibile vedere, il valore dell'errore raggiunge lo zero con i dati di allenamento, mentre assume un valore molto piccolo sul validation set; la differenza tra i due gruppi è quasi nulla, sinonimo di una buona regolarizzazione della funzione di costo.
- **Accuracy**: l'accuratezza, parametro che misura la percentuale di classificazioni corrette, è molto vicina ad uno per entrambi i casi e questo ci fa supporre che la rete sia in grado di discriminare sufficientemente bene le due classi.
- **Recall**: anche questo valore raggiunge quasi il 100%, suggerendo la capacità del modello di predire correttamente i casi contaminati.

5.2.3 Verifica

Una volta completata la fase di training è possibile verificare le reali prestazioni della macchina sul test set.
Prima di effettuare le predizioni sugli esempi di verifica, è utile calcolare le matrici di confusione e le diverse metriche anche per gli altri set.

- Partendo dal training e dal validation set, ci aspettiamo di avere il numero minimo di errori e di raggiungere per entrambi le performance massime: i dati in esame sono già stati visti dalla rete neurale e questo dovrebbe favorire il riconoscimento delle labels corrette; con una soglia impostata a 0.43, sono ottenuti i seguenti risultati:



(a) Training set (b) Validation set

Figura 5.4: Matrici di confusione training e validation set - modello 1, dataset 1

Accuratezza	:	1.00000
Precisione	:	1.00000
Richiamo	:	1.00000
Errore logaritmico	:	0.00000

Come previsto, in entrambi i casi tutte le misure sono state collocate nella giusta categoria e non è stato rilevato alcun falso positivo o negativo.

- I risultati che il modello ottiene sul test set definiscono le reali capacità della macchina e sono quelli da analizzare per valutare l'abilità della stessa di eseguire la classificazione.

  Confrontando per ciascun esempio l'etichetta reale ed il valore predetto dalla rete, la matrice di confusione associata al test set è così caratterizzata:

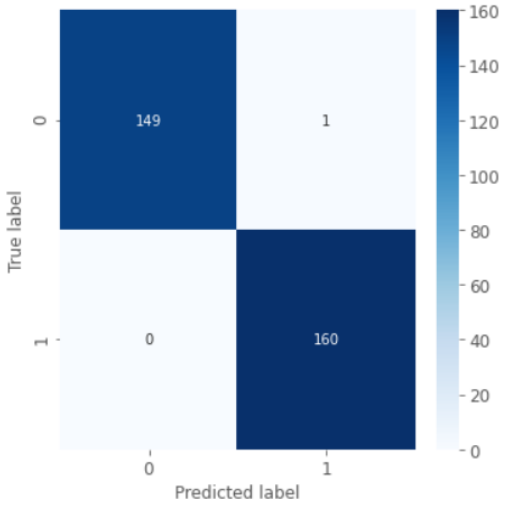


Figura 5.5: Matrice di confusione sul test set - modello 1, primo dataset

Sul totale dei 310 esempi, una sola misura è stata classificata in modo errato: come è possibile vedere in Figura 5.5 tutti i casi con contaminante sono stati rilevati correttamente mentre un barattolo sicuro ha generato l'unico falso positivo della verifica.

Le metriche calcolate sulla stessa ricerca sono quelle riportate di seguito, penalizzate dall'errore ma comunque ottime per il problema affrontato.

Accuratezza	:	0.99677
Precisione	:	0.99379
Richiamo	:	1.00000
Errore logaritmico	:	0.11142

Le prestazioni del modello possono essere ritenute molto soddisfacenti ed è possibile affermare con sicurezza che la funzione ricavata durante la fase di allenamento è in grado di discriminare correttamente le diverse categorie.

- Per concludere la verifica di questo primo modello, possiamo ripetere le stesse operazioni su l'extra set; in questo caso il rapporto tra i valori predetti e quelli reali costituisce la seguente matrice:

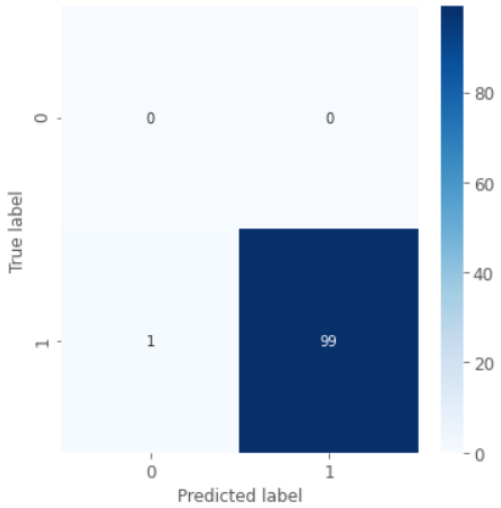


Figura 5.6: Matrice di confusione sull'extra set - modello 1, primo dataset

In un dataset formato da sole misure con contaminante, un esempio è stato ritenuto pulito; a differenza del test set, l'unico errore ottenuto su questo insieme è un falso negativo e questo ha un effetto sul valore di recall.

Accuratezza	:	0.99000
Precisione	:	1.00000
Richiamo	:	0.99000
Errore logaritmico	:	0.34539

Con una ricerca più approfondita possiamo verificare che il tipo di contaminante che ha provocato l'errore è un frammento di vetro ('g').

I risultati sono ottimi anche in questo caso ed il modello potrebbe essere già utilizzato così con gli stessi iperparametri; con uno spazio di ricerca più fine, si potrebbero inoltre ricercare dei valori in grado di annullare i pochi errori rilevati sul test set.
Le ricerche successive verteranno alla semplificazione della rete, focalizzando l'analisi sulla riduzione del numero di neuroni e livelli per ottenere così reti più piccole senza compromettere le performance riscontrate finora.

5.3 Secondo modello

5.3.1 Ricerca bayesiana

Il procedimento seguito per la definizione di un nuovo modello è analogo a quello presentato nella sezione precedente.
A differenza della prima ricerca, con le analisi successive l'attenzione è concentrata sull'ottimizzazione di aspetti specifici come il numero di errori commessi o i valori di accuratezza e precisione.
In questo caso, visti e commentati i risultati ottenuti con il primo modello, è ricercata una rete neurale in grado di raggiungere le stesse performance utilizzando un numero minore di neuroni e livelli nascosti.

La ricerca bayesiana viene avviata ed eseguita su uno spazio di ricerca più preciso e piccolo, in cui i diversi parametri da confrontare sono scelti con una differenza minore rispetto al caso precedente.

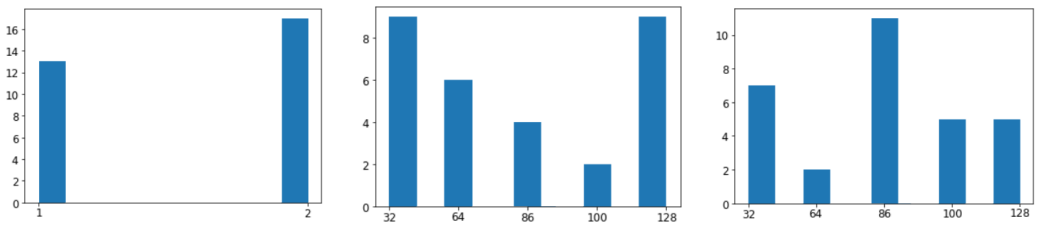
kerasclassifier___n_hidden	:	['1', '2']
kerasclassifier___n_neurons_hidden_1	:	['32', '64', '86', '100', '128'],
kerasclassifier___n_neurons_hidden_2	:	['32', '64', '86', '100', '128'],
kerasclassifier___epochs	:	[700]
kerasclassifier___batch_size	:	[128]
kerasclassifier___learning_rate	:	['0.001']
kerasclassifier___regularization_rate	:	['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate	:	['0.0', '0.1', '0.3']

Confrontando lo spazio sopra riportato con quello usato nella prima ricerca, è possibile notare come il numero massimo di hidden layers ammessi sia ridotto a 2 e lo stesso numero di neuroni nei primi livelli sia concentrato in un range più piccolo di valori.

Il processo è ripetuto più volte utilizzando liste di selezione diverse, e per ogni risultato sono state valutate le prestazioni; l'output relativo alla combinazione migliore, riportato di seguito, è ottenuto dopo 30 iterazioni e con 3 livelli di validazione incrociata.

kerasclassifier___n_hidden	:	2
kerasclassifier___n_neurons_hidden_1	:	64
kerasclassifier___n_neurons_hidden_2	:	86
kerasclassifier___learning_rate	:	0.001
kerasclassifier___regularization_rate	:	0.0
kerasclassifier___dropout_rate	:	0.3

I parametri di regolarizzazione e dropout non sono stati modificati e rimangono fissati rispettivamente a 0 e 0.3; il numero di neuroni negli unici due hidden layers vengono invece ridotti fino ad un totale di 64 ed 86.
Analizzando il risultato della ricerca bayesiana, è nuovamente possibile rappresentare gli istogrammi e studiare il contributo che ciascuna opzione ha avuto sulla scelta dei valori finali.



(a) Istogramma n_hidden (b) Istogramma layer_1 (c) Istogramma layer_2

Figura 5.7: Risultati della seconda ricerca bayesiana - modello 2, primo dataset

5.3.2 Allenamento e validazione

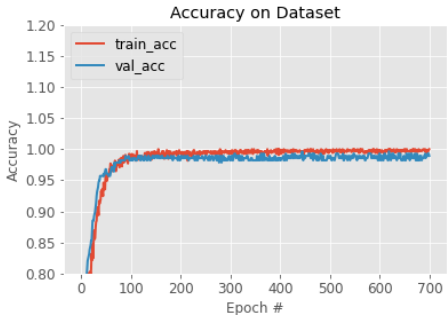
La nuova rete neurale è definita seguendo i risultati della ricerca sopra riportata ed è costituita da un totale di 7661 parametri addestrabili: i 64 neuroni del primo livello vengono implementati utilizzando 1984 parametri, mentre per il secondo layer ne sono utilizzati 5590.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                1984
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 86)                5590
_________________________________________________________________
dropout_1 (Dropout)          (None, 86)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 87
=================================================================
Total params: 7,661
Trainable params: 7,661
Non-trainable params: 0
_________________________________________________________________
```

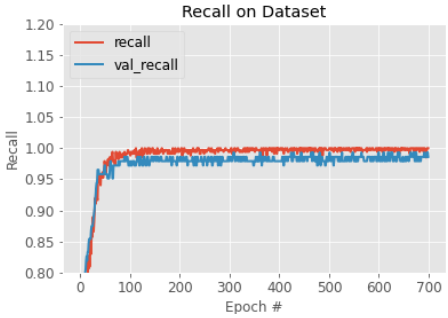
Figura 5.8: Definizione del modello - primo dataset

Avviando l'allenamento, vengono quindi valutate le evoluzioni di errore, accuratezza e richiamo durante le 700 epoche.

(a) Errore



(b) Accuratezza



(c) Richiamo

Figura 5.9: Evoluzione delle metriche - modello 2, primo dataset

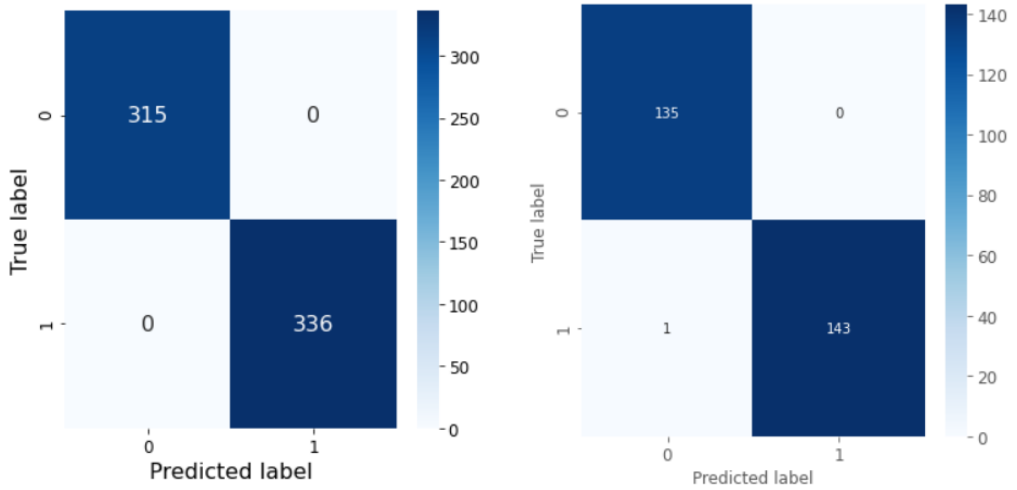
Dalla Figura 5.9a è possibile vedere come l'andamento dell'errore sul training e sul validation set tenda a convergere verso valori sempre più piccoli: l'utilizzo di un numero maggiore di epoche sarebbe stato probabilmente in grado di ridurre ulteriormente il valore finale ma, dalle verifiche che svolgeremo nella sezione seguente, si vede come questi livelli siano già sufficienti per ottenere buone prestazioni.

Come nel caso del primo modello, i parametri di richiamo e accuratezza raggiungono quasi il valore massimo per entrambe le curve, permettendo di evitare completamente l'applicazione di qualsiasi tecnica di regolarizzazione L1 o L2.

5.3.3 Verifica

La valutazione dei risultati finali è eseguita nuovamente andando a rappresentare le matrici di confusione su ciascun dataset: contando il numero ed il tipo di errori commessi, si è in grado di capire se il modello ottenuto può svolgere o meno un buon lavoro.

- La verifica inizia con la classificazione degli esempi del training e del validation set; in questo caso, le prestazioni ottenute sono quelle riportate di seguito in cui vediamo un'ottima capacità del modello di distinguere le due classi e degli altrettanto buoni valori di accuratezza e richiamo.



(a) Training set (b) Validation set

Figura 5.10: Matrici di confusione su training e validation set - modello 2

Accuratezza	:	1.00000
Precisione	:	1.00000
Richiamo	:	1.00000
Errore logaritmico	:	0.00000

(a) Training set

Accuratezza	:	0.99642
Precisione	:	1.00000
Richiamo	:	0.99306
Errore logaritmico	:	0.12379

(b) Validation set

L'unico falso negativo riscontrato nel validation set è associato ad una scheggia di legno ('w'), ma sul totale dei 144 casi con contaminante, il suo effetto sulle metriche è estremamente limitato.

- Proseguendo con il test set, la nuova matrice di confusione viene calcolata e confrontata con quella del modello precedente:

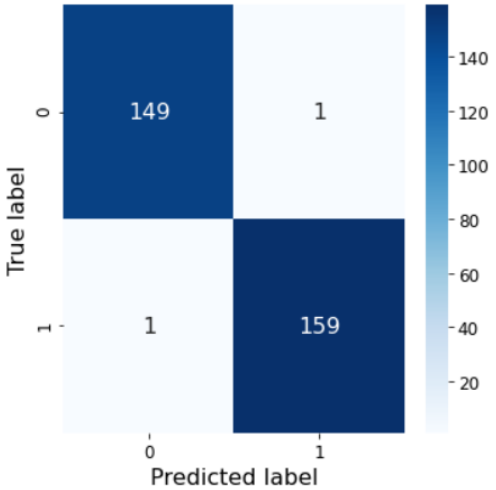


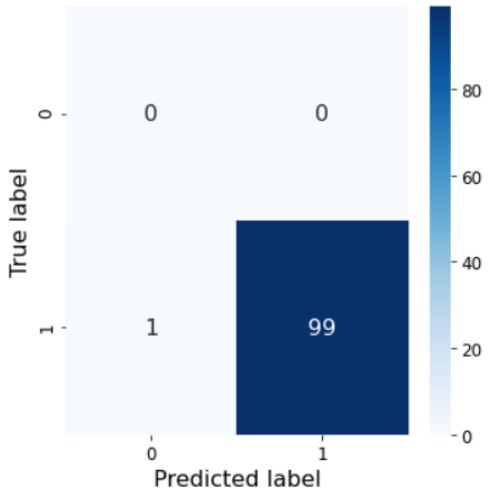
Figura 5.11: Matrice di confusione sul test set - modello 2, primo dataset

In questo caso, con una soglia di 0.5, vengono rilevati insieme un falso positivo ed un falso negativo; nonostante sia stato commesso un errore in più, la rete neurale è in grado di svolgere complessivamente un ottimo lavoro, avvalorato anche dal calcolo dell'errore logaritmico e degli altri parametri.

Accuratezza	:	0.99355
Precisione	:	0.99375
Richiamo	:	0.99375
Errore logaritmico	:	0.22283

Nonostante la rete sia stata costruita su un numero minore di parametri, le prestazioni ottenute sono paragonabili a quelle del primo modello, molto buone sia in termini di precisione che di richiamo.

- E' possibile quindi concludere la verifica analizzando gli stessi risultati su l'extra set: anche su questo gruppo di esempi il numero di errori rimane limitato e conferma la buona riuscita del processo di allenamento.



Accuratezza	:	0.99000
Precisione	:	1.00000
Richiamo	:	0.99000
Errore logaritmico	:	0.34539

Figura 5.12: Matrice di confusione e metriche su l'extra set - modello 2

Così come nel caso del primo modello, contiamo un falso negativo associato ad una scheggia di vetro.

Considerando tutto il lavoro svolto su questo primo dataset, l'ipotesi di poter rilevare i contaminanti utilizzando solo il modulo dei coefficienti di diffusione può essere verificata ed approfondita su nuove misure; i risultati ottenuti favoriscono il proseguimento delle analisi, con l'intenzione primaria di semplificare ulteriormente la rete neurale e ridurre il numero di neuroni necessari.

Prima di proseguire su questa strada è però necessario generalizzare il lavoro cercando di applicare lo stesso concetto ad altri gruppi di esempi.
Nelle sezioni seguenti saranno presentati i risultati relativi a dataset diversi, ottenuti e misurati con tecniche e strumenti differenti da quelli attuali.

Nel caso in cui le prestazioni venissero confermate, potremmo tornare a lavorare su questo modello rimuovendo alcune antenne (e le relative features) e cercando nuove soluzioni per implementare ed utilizzare la rete su una scheda di sviluppo.

Capitolo 6

Secondo dataset con scudo

6.1 Analisi del dataset

Il secondo dataset analizzato in questo lavoro di tesi è caratterizzato da 5455 misure ottenute nelle seguenti condizioni:

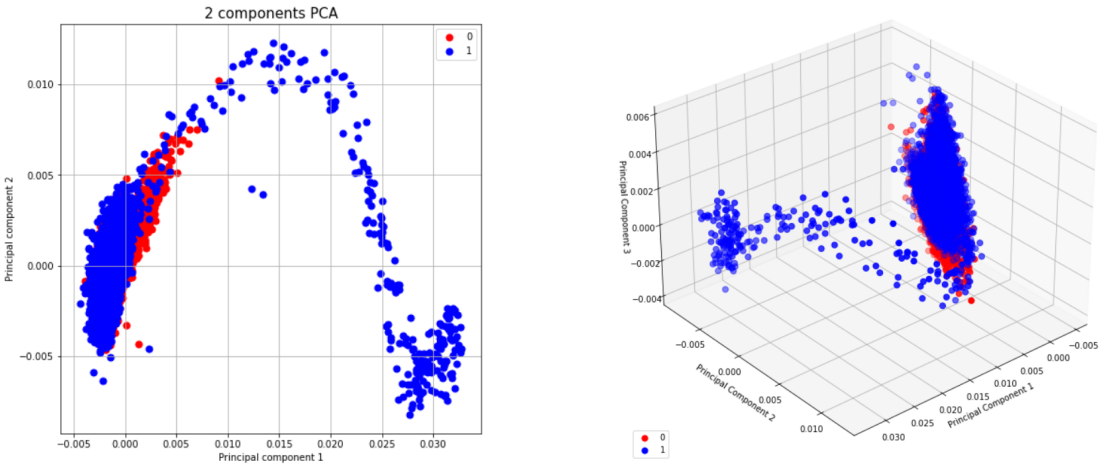
- Frequenza: 10 GHz;
- Velocità del nastro: 30 cm/s;
- Numero di features per esempio: 30;
- Shield metallico montato sul supporto delle antenne.

Come è possibile notare, la velocità del nastro di confezionamento è doppia rispetto al caso precedente, mentre la frequenza ed il numero di caratteristiche per ogni esempio rimangono invariate.

La particolarità di questo dataset risiede però nell'utilizzo dello scudo metallico per proteggere le antenne dalle onde elettromagnetiche esterne: come già introdotto nel capitolo 3.1, lo shield dovrebbe essere in grado di preservare l'ambiente di rilevazione e favorire l'acquisizione di misure precise e quanto più pulite dal rumore.

In questa sezione sarà importante verificare le reali capacità del sistema e, andando a valutare le performance di una rete neurale allenata con gli opportuni dati di input, si cercherà di capire se l'aggiunta dello shield possa portare dei benefici concreti al progetto MIT-Food.

Le matrici di diffusione utilizzate per costruire questo dataset fanno riferimento a barattoli con e senza contaminante; utilizzando la *Principal Component Analysis* è possibile rappresentare tutti i punti in un piano a due e tre dimensioni. Indicando con il rosso gli esempi sicuri ed in blu i casi contaminati, si ottengono i seguenti grafici:



(a) 2D PCA (b) 3D PCA

Figura 6.1: Principal Component Analysis su 2 e 3 dimensioni - secondo dataset

Nel dettaglio, possiamo dividere e analizzare separatamente le due classi: i barattoli sicuri sono salvati nella cartella *0_free_10_0*, in cui è specificato sia il valore target 0 che la frequenza utilizzata per ricavare tale misura.

	S12	S13	S14	S15	S16	S21	...	S64	S65	target
count	1701.000000	1701.000000	1701.000000	1701.000000	1701.000000	1701.000000	...	1701.000000	1701.000000	1701.0
mean	0.012118	0.003723	0.005142	0.012965	0.007269	0.012696	...	0.002208	0.005963	0.0
std	0.000741	0.000520	0.000907	0.000490	0.000866	0.000661	...	0.000785	0.000685	0.0
min	0.008822	0.002583	0.001221	0.004919	0.004919	0.009932	...	0.000153	0.003828	0.0
25%	0.011612	0.003372	0.004489	0.012704	0.006649	0.012255	...	0.001639	0.005516	0.0
50%	0.012236	0.003647	0.005181	0.012958	0.007270	0.012775	...	0.002213	0.005909	0.0
75%	0.012652	0.003961	0.005777	0.013239	0.007805	0.013192	...	0.002743	0.006359	0.0
max	0.014284	0.006548	0.008420	0.014861	0.012436	0.014542	...	0.004782	0.008523	0.0

Tabella 6.1: Descrizione degli esempi senza contaminante - secondo dataset

Rispetto ai 600 esempi del primo dataset, in questo caso si hanno a disposizione 1701 matrici, costituite nuovamente dai 30 parametri di scattering delle sei antenne (in modulo).

Nella tabella 6.1 sono riportati per ogni feature i valori minimi e massimi, insieme con il valor medio e la deviazione standard di ognuno (rispettivamente le righe *min, max, mean* e *std*): questi parametri possono essere utilizzati per confrontare tra loro le distribuzioni degli esempi sicuri e dei barattoli contaminati.

I casi con contaminante sono invece 3754 e comprendono rilevazioni effettuate con oggetti diversi nella forma e nel materiale; impostando la tabella seguente, è possibile distinguere con maggiore precisione ciascun contributo:

Contaminante	**Sigla**	**Numero**
Frammento di vetro	g	501
Frammento di nylon	n	500
Frammento di plastica (scheggia del tappo)	pA	501
Altro frammento di plastica	pB	500
Oggetto di plastica (stampato in 3D)	pC	501
Sfera di soda-lime-glass	slg	501
Sfera di politetrafluoroetilene	ptfe	500
Scheggia di legno	w	250

Tabella 6.2: Numero di esempi con ciascun contaminante - secondo dataset

La disponibilità di un numero relativamente grande di esempi permette di effettuare delle valutazioni che su un dataset limitato non potrebbero essere fatte. Nell'introduzione al machine learning è già stato detto come il processo di training e la funzione imparata della rete migliorino con l'aumentare del numero di misure e, nello stesso modo, è possibile generalizzarne il comportamento finale.

Come è stato fatto anche nel capitolo precedente, il dataset viene quindi diviso nei tre gruppi per il training, la validazione ed il test: riservando il 25% degli esempi totali per la verifica finale, si ottengono i seguenti set:

Set	**Numero di esempi**	**Classe 0**	**Classe 1**
Training set	2863	31.19 %	68.81 %
Validation set	1228	31.19 %	68.81 %
Test set	1364	31.19 %	68.81 %

Tabella 6.3: Divisione e distribuzione dei tre set - secondo dataset

6.2 Modello

6.2.1 Ricerca bayesiana

Per la costruzione della rete neurale sono seguiti gli stessi passaggi svolti anche con il primo dataset: partendo da una ricerca bayesiana si ottengono gli iperparametri consigliati per l'applicazione in esame, e con questi valori è possibile definire la struttura della macchina prima della fase di allenamento.
A fronte anche dei risultati che saranno illustrati nelle sezioni successive, è qui presentato un unico modello, quello che nello specifico ha permesso di ottenere i risultati migliori.

Lo spazio di ricerca è stato modificato più volte per ottimizzare il più possibile gli iperparametri, fino ad essere definito dalle seguenti liste:

kerasclassifier___n_hidden	:	['1', '2', '4']
kerasclassifier___n_neurons_hidden_1	:	['400', '460', '512', '560']
kerasclassifier___n_neurons_hidden_2	:	['64', '128', '200', '256']
kerasclassifier___n_neurons_hidden_3	:	['400', '460', '512', '560']
kerasclassifier___n_neurons_hidden_4	:	['400', '460', '512', '560']
kerasclassifier___epochs	:	[700]
kerasclassifier___batch_size	:	[256]
kerasclassifier___learning_rate	:	['0.001']
kerasclassifier___regularization_rate	:	['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate	:	['0.0', '0.1', '0.3']

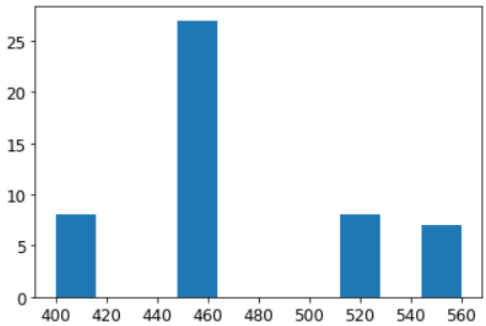
Ad ogni iterazione, un elemento della lista è campionato ed applicato per inizializzare la rete ed al termine dell'operazione i migliori iperparametri sono stampati in output.

kerasclassifier___n_hidden	:	4
kerasclassifier___n_neurons_hidden_1	:	460
kerasclassifier___n_neurons_hidden_2	:	256
kerasclassifier___n_neurons_hidden_3	:	460
kerasclassifier___n_neurons_hidden_4	:	512
kerasclassifier___learning_rate	:	0.001
kerasclassifier___regularization_rate	:	0.001
kerasclassifier___dropout_rate	:	0.3

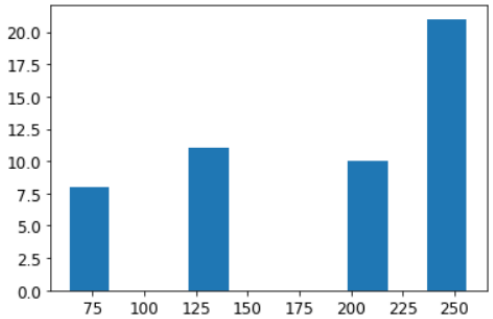
Dopo questa specifica ricerca sono consigliati 4 livelli nascosti, con un numero di neuroni nettamente maggiore rispetto a quelli avuti nel caso del primo dataset.

Durante lo svolgimento di questa analisi sono state allenate anche reti con 8 hidden layers ma, nonostante l'aumento della complessità, non è stato possibile migliorare ulteriormente i risultati avuti con questo modello.

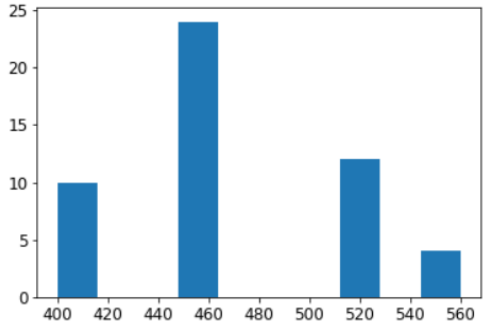
L'operazione che ha portato all'ottenimento degli iperparametri sopra riportati è stata svolta su 50 iterazioni e con tre livelli di validazione incrociata.



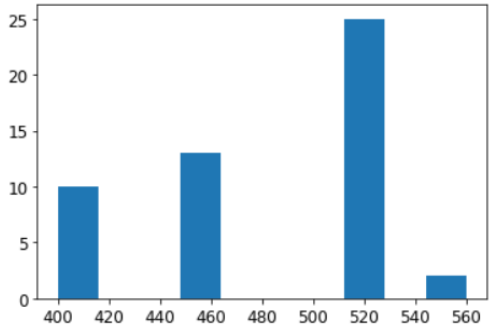
(a) Istogramma layer_1



(b) Istogramma layer_2



(c) Istogramma layer_3



(d) Istogramma layer_4

Figura 6.2: Risultati della ricerca bayesiana - secondo dataset

Dalla Figura 6.2 è possibile notare come, per tutti i livelli nascosti, il numero di neuroni suggerito equivalga a quello maggiormente utilizzato dall'algoritmo e che, per ognuna delle scelte, le performance massime sono ottenute con la combinazione sopra descritta.

Una volta definita la rete neurale ed aver specificato gli iperparametri, la macchina si vede composta da 487041 parametri addestrabili, tutti distribuiti tra i quattro livelli interni.

6.2.2 Allenamento e validazione

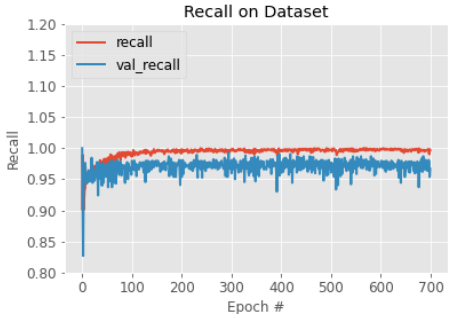
Il processo di allenamento permette alla macchina di imparare una funzione per discriminare tra loro le due classi, analizzando gli esempi del training set e testando ad ogni iterazione le prestazioni sul validation set.

Il parametro da massimizzare durante l'intera operazione è l'accuratezza, metrica che misura la percentuale di classificazioni corrette.
Con un massimo di epoche fissato a 700, è possibile leggere nel file di log l'evoluzione del parametro durante tutto l'allenamento; in particolare è utile sapere che l'ultimo miglioramento si è avuto dopo l'epoca 430 in cui l'accuratezza è passata da 0.95847 a 0.96091.

Analizzando i grafici relativi al processo, si possono valutare anche gli andamenti dell'errore e del richiamo, anche in questo caso rappresentati sia sul training set che sul validation set:

(a) Errore

(b) Accuratezza



(c) Richiamo

Figura 6.3: Evoluzione delle metriche - secondo dataset

- In Figura 6.3a è rappresentato l'errore associato ad entrambi gli insiemi, rispettivamente in rosso quello sul training set ed in blu quello sul validation set; è evidente in questo caso come i due trend si differenzino l'uno dall'altro e suggeriscano in partenza di aumentare il grado di regolarizzazione sul modello finale.

  Nello svolgimento completo della tesi sono stati provati diversi valori di dropout e regolarizzazione L2 ma, su tutte le analisi, l'assegnazione migliore per questi iperparametri è stata quella suggerita dalla ricerca bayesiana, già presentata nella sezione precedente.

  Nonostante i grafici relativi all'allenamento facciano intuire che le prestazioni saranno diverse tra i due gruppi di misure, il modello qui presentato permette di ottenere i risultati migliori tra tutti quelli testati.

- L'accuratezza ed il richiamo hanno nuovamente un comportamento diverso tra i due set: con gli esempi di allenamento, entrambe le misure raggiungono il valore massimo mentre rimangono penalizzate con i dati di validazione.

Per approfondire e valutare le reali capacità della rete, è possibile quindi ripetere le analisi su tutti gli esempi a disposizione e misurare il numero di errori commessi con ciascuna delle tre matrici di confusione.

6.2.3 Verifica

Nello stesso modo in cui è stato fatto per entrambi i modelli descritti al capitolo 5, anche per questo secondo dataset la rete neurale ottenuta dopo l'allenamento è verificata per avere un'opportuna valutazione delle reali possibilità.

Su tutti e tre i set di allenamento, validazione e test sono misurate e commentate le metriche principali, e sono quindi calcolati i numeri di falsi positivi e negativi su ciascuno di essi.

- Partendo dal training set, sono effettuate le predizioni dei 2863 esempi; essendo le stesse misure utilizzate per insegnare alla macchina la funzione da eseguire, sono attese le prestazioni massime ed il numero minimo di errori.

  La matrice di confusione in Figure 6.4 conferma le aspettative, inserendo correttamente tutti i dati nelle rispettive classi.

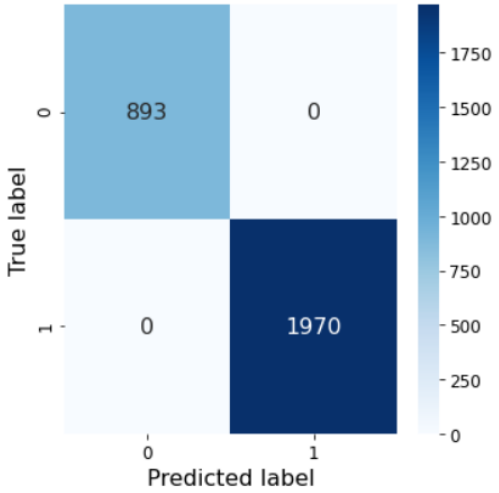


Figura 6.4: Matrice di confusione sul training set - secondo dataset

I valori di accuratezza, precisione e richiamo raggiungono tutti il 100% mentre l'errore logaritmico è completamente annullato.

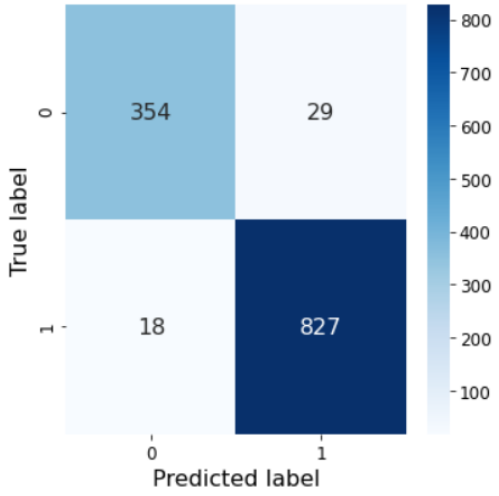
Accuratezza	:	1.00000
Precisione	:	1.00000
Richiamo	:	1.00000
Errore logaritmico	:	0.00000

La soglia scelta è di 0.49, a metà tra la probabilità che la label sia predetta come 0 o 1.
Dai risultati avuti su questo set, sarebbe ipotizzabile la riuscita del processo di allenamento, ma devono essere ancora dimostrate le reali prestazioni con la verifica dell'assenza di un sovra-adattamento.

- Il validation set è composto da 1228 esempi ed è utilizzato per misurare l'accuratezza del modello al termine di ogni epoca.

  Utilizzando in questo caso l'intero set, calcoliamo le prestazioni della rete finale: a differenza dei casi precedenti, il numero di errori commessi è qui più grande e non trascurabile.

  Dall'immagine sotto riportata è possibile vedere nel dettaglio la matrice di confusione e considerare il contributo di ciascuna predizione sui valori delle metriche.



Accuratezza	:	0.96173
Precisione	:	0.96612
Richiamo	:	0.97870
Errore logaritmico	:	1.32194

Figura 6.5: Matrice di confusione sul validation set - secondo dataset

Complessivamente il modello classifica in modo errato 47 esempi: 29 di questi sono falsi positivi e, per quanto dimostrino un comportamento sbagliato della macchina, non sono troppo gravi ai fini dell'applicazione in esame.
Di maggiore interesse sono invece i 18 falsi negativi: questi barattoli sono ritenuti sicuri dalla rete ma presentano al loro interno un agente contaminante e non dovrebbero essere immessi nel mercato.

I valori di precisione e richiamo sono ridotti a causa degli stessi problemi, mentre l'errore logaritmico cresce come conseguenza del numero di predizioni errate.

Rimandando le considerazioni finali fin dopo aver analizzato i risultati del test set, sono già visibili delle differenze rispetto al primo dataset, in cui anche una rete più piccola era in grado di ottenere prestazioni nettamente migliori.

- Concludendo la verifica sul test set, sono predette le labels dei 1364 esempi e sono quindi confrontate con i valori reali; in questo caso si ottengono le seguenti classificazioni:

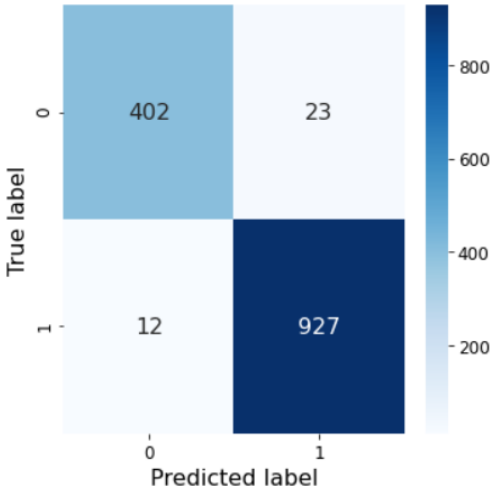


Figura 6.6: Matrice di confusione sul test set - secondo dataset

Il numero di errori è in questo caso pari a 35, minore di quello avuto con il validation set ma ancora elevato; per avere una stima più accurata e verificare se il peggioramento delle prestazione sia associato ad una classe specifica, è possibile condurre una *error analysis*, in modo da ricercare tutti gli esempi che hanno causato una predizione errata.

Considerando quindi l'intero dataset, vediamo che i falsi positivi e negativi sono così distribuiti:

Contaminante	**Sigla**	**Numero di errori**
Frammento di vetro	g	6
Frammento di nylon	n	6
Frammento di plastica (scheggia del tappo)	pA	2
Altro frammento di plastica	pB	1
Oggetto di plastica (stampato in 3D)	pC	1
Sfera di soda-lime-glass	slg	10
Scheggia di legno	w	4
Falsi positivi	e	52

Non avendo rilevato un contaminante prevalente si può assumere che l'intero processo di allenamento non sia stato in grado di generalizzare la rete neurale, causando la differenza di prestazioni vista tra il training set e le altre misure.

I valori delle metriche misurate sul test set sono quelli sotto riportati, buoni nel caso generale di un'applicazione di machine learning, ma non sufficienti per poter usare la macchina risultante nel progetto MIT-Food.

Accuratezza	:	0.97434
Precisione	:	0.97579
Richiamo	:	0.98722
Errore logaritmico	:	0.88627

Dai risultati ottenuti si può intuire come il modello finale sia probabilmente soggetto a sovra-adattamento: gli esempi presenti all'interno del training set vengono classificati correttamente mentre gli altri risentono della mancanza di generalizzazione e generano i numerosi errori rilevati in fase di verifica.

Cercando di riassumere le caratteristiche della rete e considerando l'analisi completa eseguita su di essa, è possibile trarre le seguenti conclusioni:

1. L'applicazione dello scudo metallico per proteggere le antenne dalle onde elettromagnetiche esterne non comporta un miglioramento delle prestazioni ed utilizzando solo il modulo dei coefficienti di diffusione non si è in grado di ottenere una rete neurale capace di discriminare i barattoli contaminati con sufficiente precisione;

2. L'aumento della complessità del modello rispetto alle reti allenate per il primo dataset può essere dovuto a due fattori principali:

   - La dimensione del secondo dataset, come è stato detto, conta un numero di esempi molto maggiore rispetto al primo e permette di condurre un'analisi che solo una tale disponibilità rende possibile; nonostante gli errori siano numerosi, nel rapporto della totalità dei dati non incidono in modo eccessivo sulle metriche che infatti mantengono valori piuttosto alti.
   - La velocità del nastro sulla catena di confezionamento è doppia rispetto al caso precedente: la misurazione di tutti i segnali deve essere svolta in un tempo dimezzato e l'incertezza sulla posizione del barattolo aumenta generando maggiore rumore nei dati acquisiti.

Visti e commentati i risultati ottenuti su questo secondo dataset, non si può ritenere valida l'ipotesi iniziale di poter discriminare la classe di ciascun barattolo semplicemente dal modulo dei coefficienti di scattering.
Per continuare il lavoro di tesi, dopo aver verificato la stessa soluzione su un ulteriore dataset, sarà quindi necessario trovare delle tecniche per compensare la riduzione di informazioni ed ottenere allo stesso tempo delle buone prestazioni.

Capitolo 7

Terzo dataset: FT System

La conferma della validità di tutti i risultati precedenti è ottenuta attraverso lo studio e l'analisi di un ultimo dataset costruito sulle misure di FT System, un'azienda partner leader nel settore dell'imbottigliamento e del controllo qualità dei prodotti alimentari (sito ufficiale: `www.ftsystem.com`).

La collaborazione con una azienda permette al progetto di avanzare sia sul piano della ricerca che sulla verifica del dispositivo realizzato fino ad ora:

- da una parte c'è la possibilità di effettuare nuove misure con strumenti industriali, migliori e sicuramente più specifici per l'analisi da svolgere,
- dall'altra si ha l'occasione di testare il prototipo in situazioni reali, in un contesto in cui tale macchina potrebbe trovare applicazione.

Le misure fornite da FT System caratterizzano un totale di 4 dataset, diversi l'uno dall'altro per il prodotto analizzato e la velocità del nastro di confezionamento; in particolare si hanno a disposizione:

1. Misure su olio di cartamo con nastro a 30 cm/s;
2. Misure su olio di cartamo con nastro a 50 cm/s;
3. Misure su barattoli di cioccolata con nastro a 30 cm/s;
4. Misure su barattoli di cioccolata con nastro a 50 cm/s.

Seguendo gli studi condotti nei capitoli precedenti, in questa tesi sono analizzati i casi 1 e 2 con l'intenzione di massimizzare le prestazioni su ognuno ed evidenziare le differenze dovute al cambiamento di velocità.

7.1 Velocità di 30 cm/s

7.1.1 Dataset

In questa sezione sono utilizzate le misure effettuate con il nastro a 30 cm/s per allenare una rete neurale e testarne le capacità di classificazione; come gli altri dataset, anche questo insieme è composto da esempi con e senza contaminante e le predizioni sono finalizzate alla rilevazione di ciascuno dei possibili corpi estranei.

Entrando nel dettaglio del set specifico, è possibile distinguere le due classi e visualizzarne la distribuzione generale.
Complessivamente si contano 3000 esempi, ognuno dei quali è ottenuto alla frequenza di 10 GHz e rappresenta una matrice di confusione con i 30 coefficienti delle sei antenne.

Le misure relative ai barattoli sicuri sono 1000 e definite dal valore target 0, mentre i barattoli contaminati sono etichettati dal valore 1; su un piano tridimensionale gli esempi sono così distribuiti, mantenendo la solita convenzione per la quale il colore blu è associato ai casi con il corpo estraneo.

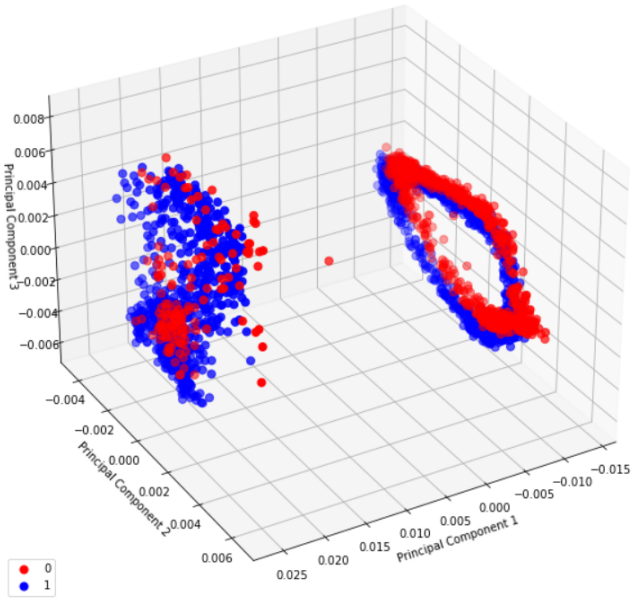


Figura 7.1: PCA su 3 dimensioni - dataset FT System con nastro a 30 cm/s

Analizzando i 2000 esempi con contaminante è possibile quindi distinguere le diverse categorie di materiale.

Contaminante	**Sigla**	**Numero**
Frammento di vetro	g	332
Frammento di plastica (scheggia del tappo)	pA	334
Altro frammento di plastica	pB	332
Sfera di soda-lime-glass	slg	334
Sfera di politetrafluoroetilene	ptfe	334
Scheggia di legno	w	334

Tabella 7.1: Esempi con ciascun contaminante - dataset FT System 30 cm/s

Dividendo il dataset nei tre gruppi per training, validation e test, sono svolte diverse ricerche bayesiane ed i relativi processi di allenamento.

Set	**Numero di esempi**	**Classe 0**	**Classe 1**
Training set	1575	33.33 %	66.67 %
Validation set	675	33.33 %	66.67 %
Test set	750	33.33 %	66.67 %

Tabella 7.2: Divisione e distribuzione dei tre set - dataset FT System 30 cm/s

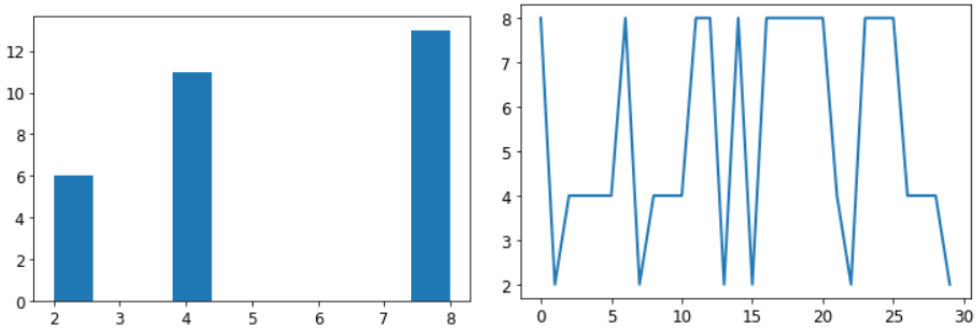
7.1.2 Ricerca bayesiana

L'algoritmo di BayesSearch ricerca per ciascun iperparametro il valore ottimale tra quelli presenti nello spazio di ricerca; dopo alcuni tentativi svolti sul dataset in esame, l'ultima esecuzione è stata effettuata con le seguenti scelte, in cui il numero di livelli intermedi è forzato e limitato ad uno.

kerasclassifier___n_hidden : ['1']
kerasclassifier___n_neurons_hidden_1 : ['2', '4', '8']
kerasclassifier___epochs : [800]
kerasclassifier___batch_size : [256]
kerasclassifier___learning_rate : ['0.001']
kerasclassifier___regularization_rate : ['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate : ['0.0', '0.1', '0.3']

Con 30 iterazioni e tre livelli di validazione incrociata si ottengono i seguenti risultati, in cui 8 neuroni sono preferiti tra tutte le scelte possibili:

kerasclassifier___n_hidden	:	1
kerasclassifier___n_neurons_hidden_1	:	8
kerasclassifier___learning_rate	:	0.001
kerasclassifier___regularization_rate	:	0.00
kerasclassifier___dropout_rate	:	0.1



(a) Istogramma layer_1

(b) Neuroni nel layer_1

Figura 7.2: Risultati della ricerca bayesiana - dataset FT System 30 cm/s

Confrontando i risultati ed i valori presentati in Figura 7.2 con quelli ottenuti per i precedenti dataset, è possibile notare come la rete suggerita in questo caso sia più semplice delle altre e permetta la costruzione di una macchina con pochi neuroni: i dati raccolti con gli strumenti di FT System sembrano facilitare la discriminazione e con il nastro a 30 cm/s sembra possibile il riconoscimento di ciascuna delle due classi di esempio.

7.1.3 Allenamento e validazione

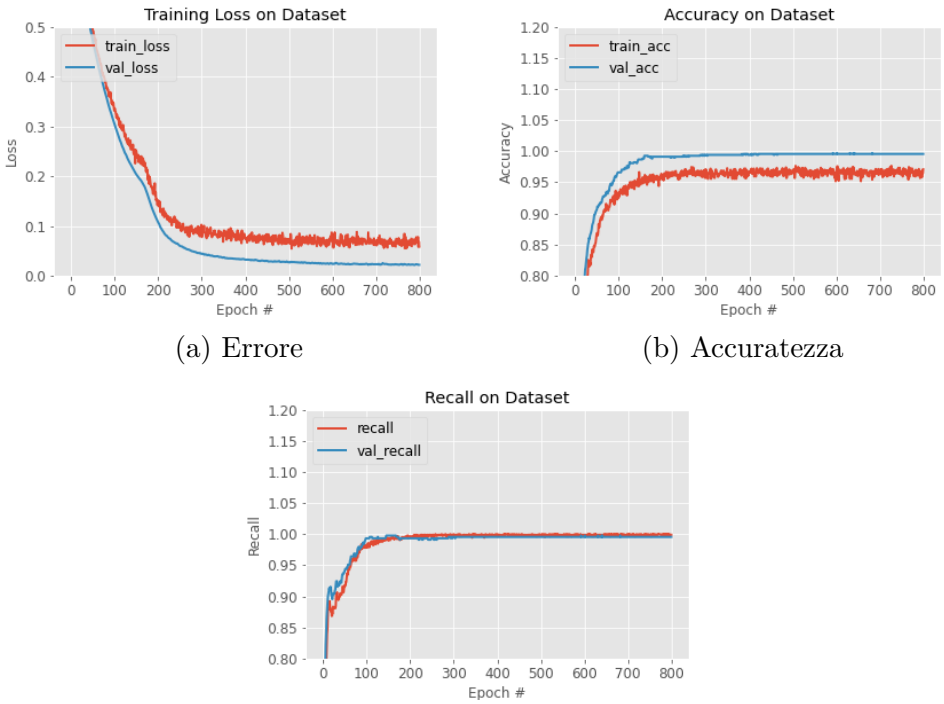
In Figura 7.2 è stato visto come la rete neurale consigliata dalla ricerca bayesiana fosse composta da un hidden layer con 8 neuroni; il processo di allenamento di una rete costruita con questi iperparametri porta ad avere ottimi risultati, tali da permettere un'ulteriore semplificazione della macchina appena testata.

I valori delle metriche ottenute in fase di verifica dimostrano l'ottima capacità del sistema di predire la classe di ciascun barattolo e aprono alla possibilità di ridurre nuovamente la complessità del dispositivo finale.

La rete neurale costruita ed allenata in questo contesto è ottenuta modificando manualmente il numero di neuroni e testando ripetutamente le prestazioni fino a raggiungere le dimensioni minime; questa è caratterizzata da:

- Numero di hidden layers : 1
- Numero di neuroni : 2
- Regolarizzazione L2 : 0.00
- Dropout : 0.1

Impostando il *learning rate* a 0.001 ed avviando il processo su 800 epoche, si ottengono gli andamenti seguenti.



(a) Errore

(b) Accuratezza

(c) Richiamo

Figura 7.3: Evoluzione delle metriche - dataset FT System 30 cm/s

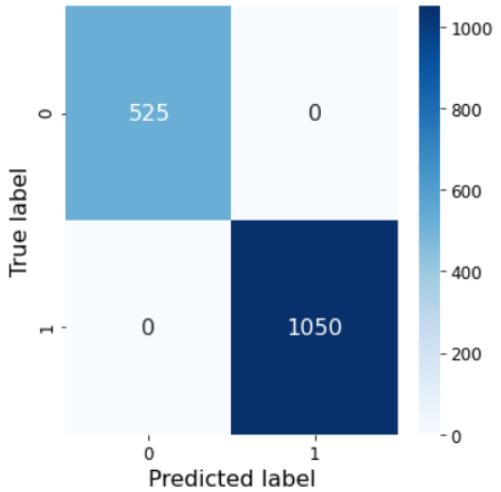
Il parametro scelto e monitorato durante tutto l'allenamento della rete è anche in questo caso l'accuratezza, la cui evoluzione è riportata in Figura 7.3b; lavorando in particolare sul training set, l'ultima epoca in cui è possibile assistere ad un miglioramento di tale metrica è la numero 593, in cui il valore passa da 0.99556 a 0.99704.

Così come per l'accuratezza, anche i valori raggiunti dall'errore (Figura 7.3a) e dal richiamo (Figura 7.3c) suggeriscono buone prestazioni per la macchina appena ottenuta, sia sugli esempi di allenamento che su quelli di validazione.

7.1.4 Verifica

Per concludere l'analisi della rete ottenuta nella sezione precedente, utilizziamo la stessa per predire la classe di tutti gli esempi del dataset, partendo dal training set ed arrivando al test set.

- Gli esempi inseriti all'interno del set di allenamento sono utilizzati per addestrare la macchina e sono gli stessi che la rete ha analizzato per imparare la funzione da applicare.
  Le prestazioni ottenute su queste misure sono massime e la classificazione avviene correttamente per tutti i 1575 barattoli.



Accuratezza	:	1.00000
Precisione	:	1.00000
Richiamo	:	1.00000
Errore logaritmico	:	0.00000

Figura 7.4: Matrice di confusione sul training set - dataset FT System 30cm/s

- Il validation set contiene 675 esempi, 450 dei quali contengono un contaminante; queste misure sono utilizzate per valutare le prestazione della macchina al termine di ogni nuova epoca e permettono di ridurre fin dalla fase di allenamento la probabilità di un sovra-adattamento.

  Con la rete neurale sopra descritta è avviata la predizione su questo set ed è ottenuta la seguente matrice di confusione.

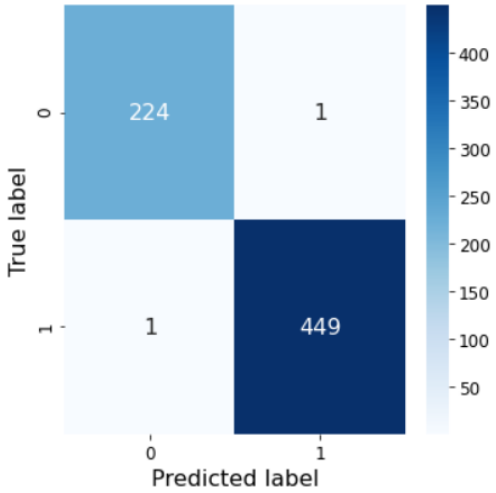


Figura 7.5: Matrice di confusione sul validation set - dataset FT System 30 cm/s

I due errori commessi (un falso positivo ed un falso negativo) riducono leggermente i valori delle metriche, ma non compromettono le prestazioni generali raggiunte sull'intero insieme.

Accuratezza	:	0.99704
Precisione	:	0.99778
Richiamo	:	0.99778
Errore logaritmico	:	0.10234

- La verifica finale viene eseguita sugli esempi di test: come già detto nei capitoli precedenti, i risultati ottenuti in questa fase sono i più significativi per la valutazione delle reali capacità della rete e permettono di simulare il comportamento che questa avrebbe lavorando sulle misure di barattoli nuovi e mai visti prima.

  Ripetendo il processo di predizione e comparando i risultati ottenuti con le etichette corrette è possibile contare il numero di errori commessi e valutarne il peso confrontandolo con il numero di casi totali.

  Lavorando con i 750 esempi contenuti in questo insieme si ha un solo errore come falso positivo.

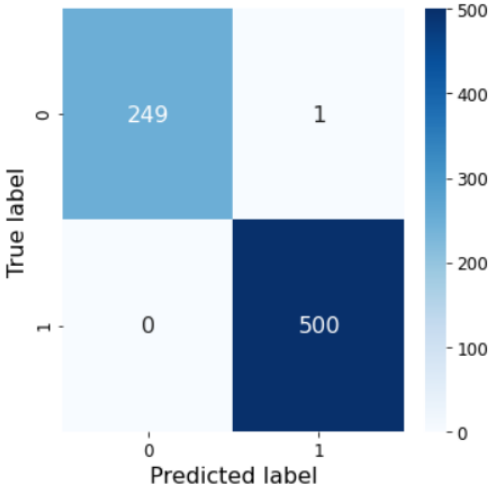


Figura 7.6: Matrice di confusione sul test set - dataset FT System 30 cm/s

Accuratezza	:	0.99867
Precisione	:	0.99800
Richiamo	:	1.00000
Errore logaritmico	:	0.04605

Le prestazioni verificate con questa rete neurale e raggiunte su questo specifico dataset sono le migliori ottenute durante l'intero progetto di tesi (lavorando alla frequenza di 10 GHz con il solo modulo dei parametri di diffusione).
La strumentazione e le condizioni con le quali sono state effettuate le misure hanno garantito il quasi perfetto addestramento della macchina che, senza tener conto dei risultati precedenti, avallerebbe l'ipotesi iniziale sulla riduzione delle features necessarie alla discriminazione.

La possibilità di utilizzare un prototipo funzionante di questo tipo deve però tenere in considerazione tutte le analisi effettuate ed essere quanto più possibile indipendente dalle variabili presenti: la velocità del nastro, la precisione ed il costo degli strumenti impiegati per misurare i segnali modificano significativamente le prestazioni della macchina e non garantiscono sempre la validità dell'intero processo.

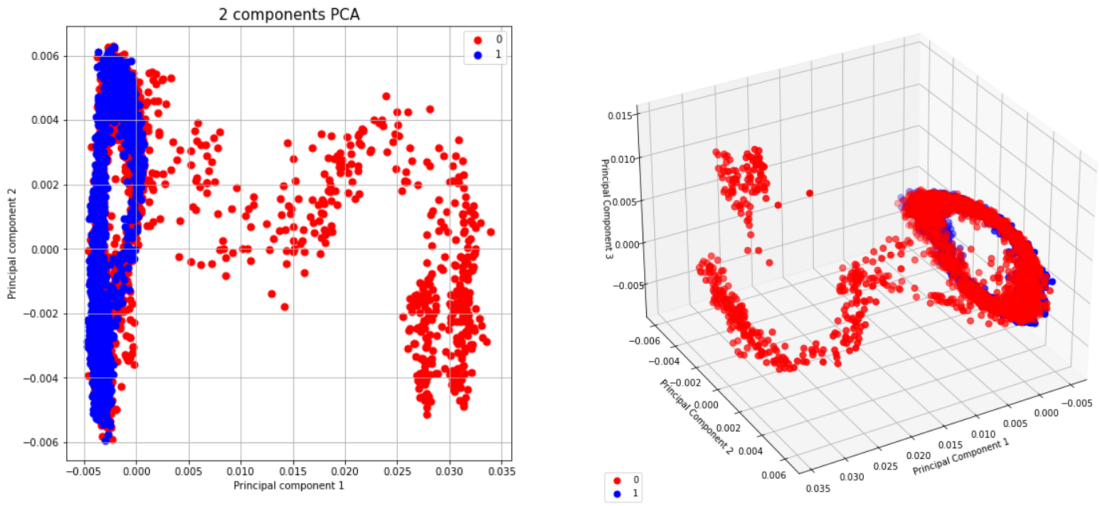
Per concludere questa fase e verificare definitivamente la possibilità di lavorare con il solo modulo dei coefficienti di scattering (con singola frequenza) si ripete l'analisi sul dataset di FT System ottenuto con il nastro di confezionamento a 50 cm/s.

7.2 Velocità di 50 cm/s

7.2.1 Dataset

Il secondo dataset fornito da FT System è caratterizzato da 4502 misure ottenute con il movimento del nastro a 50 cm/s; l'aumento di velocità permette di acquisire un maggior numero di esempi nello stesso periodo di tempo e di incrementare la produzione generale del prodotto alimentare, ma introduce una serie di fattori che potrebbero ridurre fortemente le capacità del classificatore utilizzato.

Le misurazioni sono suddivise tra campioni con e senza contaminante, prendendo nuovamente in esame diversi tipi di materiali e forme; i barattoli puliti sono 3000, contro i 1502 file contenenti i segnali relativi ad un corpo estraneo. La rappresentazione PCA permette di visualizzare la distribuzione dei punti in piani a due e tre dimensioni:



(a) 2D PCA (b) 3D PCA

Figura 7.7: PCA su 2 e 3 dimensioni - dataset FT System 50 cm/s

A differenza di tutti gli altri dataset, in questo caso si ha una prevalenza di esempi senza contaminante: un processo di addestramento, in questo contesto, potrebbe portare all'ottenimento di una rete neurale efficiente nel riconoscere i casi con valore target 0, ma potrebbe risentire di una insufficiente disponibilità di misure della classe con etichetta 1.

I barattoli compromessi sono quindi suddivisi a seconda del tipo di materiale inserito al loro interno, nel seguente rapporto:

Contaminante	**Sigla**	**Numero**
Frammento di plastica (scheggia del tappo)	pA	300
Altro frammento di plastica	pB	300
Sfera di soda-lime-glass	slg	301
Sfera di politetrafluoroetilene	ptfe	301
Scheggia di legno	w	300

Tabella 7.3: Esempi con ciascun contaminante - dataset FT System 50 cm/s

Dividendo le misure nei tre set per il training, la validazione ed il test è possibile procedere alla fase successiva e commentare i risultati ottenuti.

Set	**Numero di esempi**	**Classe 0**	**Classe 1**
Training set	2363	66.65 %	33.35 %
Validation set	1013	66.65 %	33.35 %
Test set	1126	66.61 %	33.39 %

Tabella 7.4: Divisione e distribuzione dei tre set - dataset FT System 50 cm/s

7.2.2 Ricerca bayesiana

Lo spazio di ricerca utilizzato per tutti gli iperparametri è inizialmente grande ed ogni lista al suo interno contiene un range di valori ampio e grossolano.

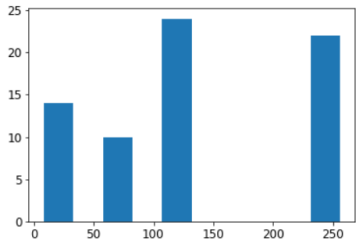
Come è stato visto nei capitoli precedenti, la scoperta dei migliori iperparametri è affidata alla ripetizione di diversi passaggi: alternando un algoritmo di ricerca bayesiana ad una fase completa di addestramento e validazione è possibile semplificare ed ottimizzare al massimo una rete neurale per poter svolgere il lavoro di classificazione al meglio.

Se da una parte i precedenti dataset hanno permesso, con questo processo, di costruire reti gradualmente più semplici ed efficienti, con questo set di misure non è stato possibile.
Nonostante le diverse applicazioni dell'algoritmo di BayesSearch, la variazione suggerita per ogni iperparametro non è risultata sufficiente a migliorare le prestazioni, così come la complessità della macchina non ha subito riduzioni.

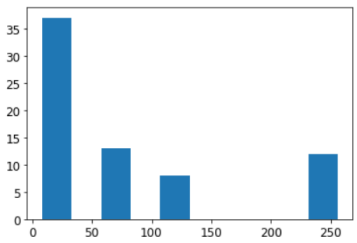
Lo spazio di ricerca che ha fornito i valori per la costruzione della rete migliore è il seguente, con il relativo risultato per ogni iperparametro richiesto.

kerasclassifier___n_hidden	:	['1', '2', '4']
kerasclassifier___n_neurons_hidden_1	:	['8', '16', '32', '64', '128', '256']
kerasclassifier___n_neurons_hidden_2	:	['8', '16', '32', '64', '128', '256']
kerasclassifier___n_neurons_hidden_3	:	['8', '16', '32', '64', '128', '256']
kerasclassifier___n_neurons_hidden_4	:	['8', '16', '32', '64', '128', '256']
kerasclassifier___epochs	:	[700]
kerasclassifier___batch_size	:	[256]
kerasclassifier___learning_rate	:	['0.001']
kerasclassifier___regularization_rate	:	['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate	:	['0.0', '0.1', '0.3']

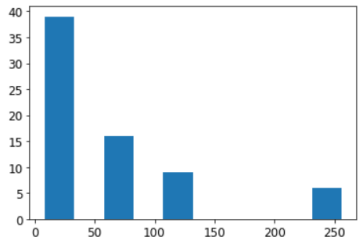
kerasclassifier___n_hidden	:	4
kerasclassifier___n_neurons_hidden_1	:	256
kerasclassifier___n_neurons_hidden_2	:	8
kerasclassifier___n_neurons_hidden_3	:	64
kerasclassifier___n_neurons_hidden_4	:	32
kerasclassifier___learning_rate	:	0.001
kerasclassifier___regularization_rate	:	0.003
kerasclassifier___dropout_rate	:	0.3



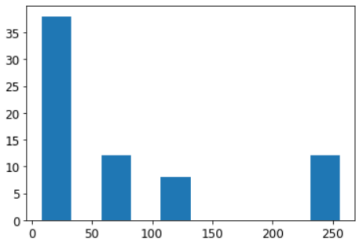
(a) Istogramma layer 1



(b) Istogramma layer 2



(c) Istogramma layer 3



(d) Istogramma layer 4

Figura 7.8: Risultati della ricerca bayesiana - dataset FT System 50 cm/s

Dopo 70 iterazioni è suggerito l'utilizzo di 4 livelli nascosti, ognuno dei quali conta un numero diverso di neuroni; il primo layer, più complesso, applica una prima trasformazione sui dati in ingresso, che poi saranno opportunamente elaborati e regolarizzati nei passaggi più interni.

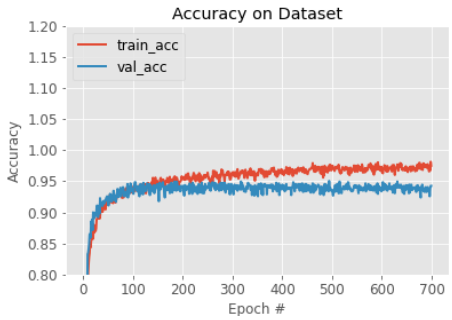
Rispetto ai risultati attesi, in rapporto anche alla rete ottenuta con la velocità del nastro a 30 cm/s, la complessità è elevata: senza avere ancora effettuato alcuna verifica delle prestazioni, il fatto di dover utilizzare un numero maggiore di livelli e neuroni suggerisce una maggiore difficoltà nella classificazione, che potrebbe permettere o meno la giusta categorizzazione di ogni barattolo.

7.2.3 Allenamento e validazione

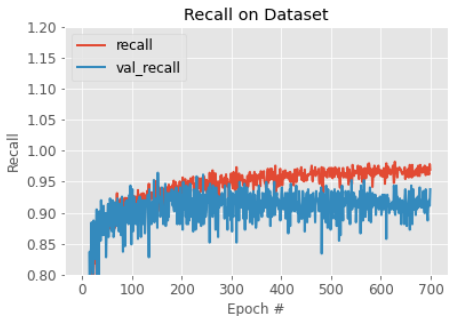
Costruita la rete con i valori sopra riportati, è avviato il processo di training per ottimizzare la funzione del classificatore sul valore dell'accuratezza.

I grafici relativi ai principali parametri da monitorare sono riportati in Figura 7.9

(a) Errore



(b) Accuratezza



(c) Richiamo

Figura 7.9: Evoluzione delle metriche - secondo dataset

Come è possibile intuire dagli andamenti sopra riportati, l'allenamento non è stato in grado di far convergere la funzione di costo della macchina a zero; per ognuno dei parametri descritti è possibile riportare alcune considerazioni, evidenziando i problemi e suggerendo delle soluzioni:

- Il primo grafico (Figura 7.9a) rappresenta l'errore sul training e sul validation set; tralasciando la differenza tra le due curve (fattore di regolarizzazione), è importante tenere in considerazione il livello raggiunto da entrambi: un valore superiore a 0.2 può influire significativamente sulle prestazione della rete, compromettendone le capacità e impossibilitandone l'utilizzo.

  Un aumento di complessità può teoricamente migliorare l'intera struttura, ma dopo alcuni tentativi è stato visto come nel caso in questione non si riesca a ridurre in modo sufficiente il numero di errori commessi in fase di predizione.

- Il valore di accuratezza mostrato in Figura 7.9b può essere ritenuto buono, considerato anche il numero totale di casi analizzati; nell'ultima epoca utile (la 495esima), il valore raggiunto sul training set è passato da 0.94965 a 0.95064, confrontabile con quello avuto anche con gli esempi di validazione.
  Proprio il modello salvato in questa epoca sarà utilizzato per le prossime fasi).

- La terza figura descrive infine il richiamo, metrica utile per monitorare il numero di falsi negativi; il comportamento oscillatorio delle due funzioni può essere attenuato modificando la dimensione del batch ma, per quanto riguarda il valore di entrambe, non è stato possibile raggiungere l'uno.

Con i risultati descritti, le prestazioni della rete ottenuta non sembrano confrontabili con le precedenti; la verifica del capitolo seguente ha lo scopo di dimostrare tali assunzioni, cercando anche di identificare le cause che hanno caratterizzato una tale differenza tra misure acquisite in modo simile l'une alle altre.

7.2.4 Verifica

La rete neurale da utilizzare in questo caso conta un totale di 12681 parametri addestrabili, distribuiti in 4 livelli tra neuroni e l'applicazione delle funzioni di regolarizzazione.

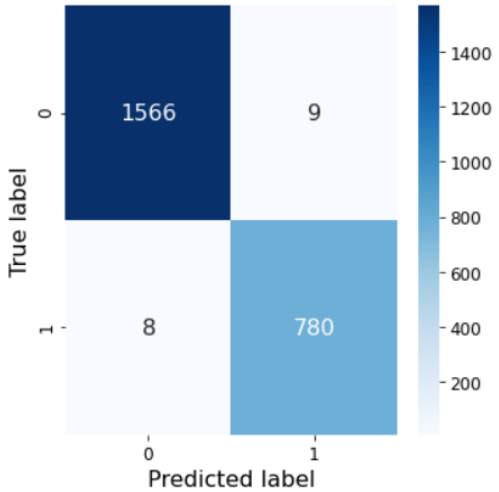
```
=================================================================
Total params: 12,681
Trainable params: 12,681
Non-trainable params: 0
```

Figura 7.10: Dettaglio sui parametri addestrabili della rete

La verifica è organizzata dividendo l'analisi tra i tre set di training, validazione e test, in modo da valutare per ognuno le prestazioni e studiare nel dettaglio il comportamento della macchina in situazioni diverse.

- Il primo gruppo di misure utilizzato è il training set: come è stato detto, questo insieme contiene 2363 esempi, 788 dei quali contengono un contaminante.

  Confrontando il valore predetto con la label corretta, si calcolano i valori delle metriche ed il numero di errori commessi, per un totale di 17.



Accuratezza	:	0.99281
Precisione	:	0.98859
Richiamo	:	0.98985
Errore logaritmico	:	0.24848

Figura 7.11: Matrice di confusione sul training set - dataset FT System 50cm/s

  Entrambi i valori di precisione e richiamo sono penalizzati dai diversi errori, presenti nella matrice nonostante gli stessi esempi fossero stati in precedenza studiati della rete.

  A differenza dei dataset già trattati, ed in particolare allo stesso set di FT System con il nastro ad una velocità di 30 cm/s, il comportamento ottenuto su queste misure è decisamente peggiore: la presenza di 8 falsi negativi presuppone un allenamento insufficiente che non riesce a trovare una funzione adeguata al riconoscimento di entrambe le classi.

  Questo stesso risultato non può che peggiorare con gli altri due dataset, evidenziando nuovamente una mancanza di generalizzazione e caratterizzazione dei dati in ingresso.

- Il validation set è utilizzato durante l'addestramento per validare l'intero processo e dovrebbe confermare il progressivo miglioramento della macchina.

  La matrice di confusione associata al dataset in esame è riportata in Figura 7.12, con il dettaglio su tutte le predizioni corrette ed errate.

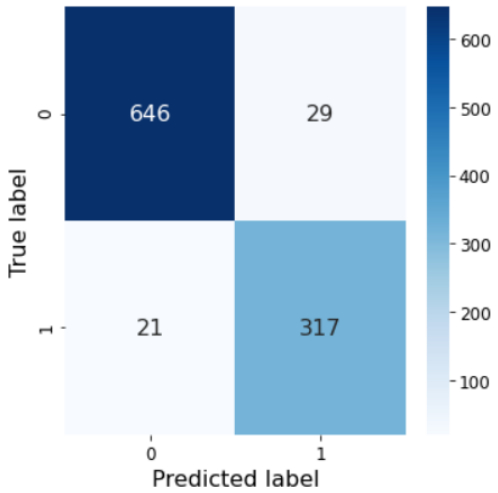


Figura 7.12: Matrice di confusione sul validation set - dataset FT System 50 cm/s

Il numero di errori è piuttosto elevato e questo influisce negativamente sulle metriche, limitando l'utilizzo generale della rete.

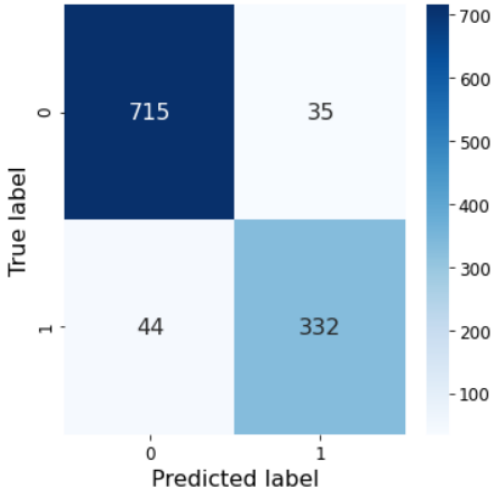
Accuratezza	:	0.95064
Precisione	:	0.91618
Richiamo	:	0.93787
Errore logaritmico	:	1.70480

Tra tutti i parametri, quelli maggiormente penalizzati sono la precisione ed il richiamo; entrambe le grandezze descrivono il rapporto tra il numero di etichette predette come 1 ed il numero totale di esempi in una delle due classi, tenendo quindi in considerazione entrambi i FP e FN.

L'aspetto principale da sottolineare è però l'incremento dell'errore logaritmico: con l'aumento di questo valore l'accuratezza della macchina diminuisce giustificando di conseguenza tutti gli errori commessi in queste fasi.

- La dimostrazione e la verifica finale delle prestazioni vengono effettuate sul test set; questo insieme di misure serve a controllare la corretta generalizzazione della funzione di costo, così come a simulare il comportamento della macchina in circostanze reali.

  Cercando di valutare i risultati, è possibile concludere lo studio di questa rete confrontando le predizioni con ciascuna label corretta.



Accuratezza	:	0.92984
Precisione	:	0.90463
Richiamo	:	0.88298
Errore logaritmico	:	2.42326

Figura 7.13: Matrice di confusione sul test set - dataset FT System 50 cm/s

Con un totale di 79 errori, la rete ottenuta non può essere utilizzata in alcuna operazione ed evidenzia una serie di problemi sull'applicazione del dispositivo: l'aumento di velocità del nastro di confezionamento ha compromesso totalmente le prestazioni, confutando l'ipotesi di poter rilevare i contaminati dall'analisi del solo modulo dei parametri di diffusione (con singola frequenza).

L'incertezza sulla posizione del barattolo, il tempo necessario all'acquisizione di tutti i segnali ed il numero di features scelto per ogni esempio sono solo alcuni dei fattori che influenzano le performance del dispositivo ed è necessario agire su questi per ottimizzarne il lavoro finale.

Per poter continuare il lavoro di tesi e progettare un dispositivo funzionante è necessario trovare una nuova strategia di analisi che garantisca buoni risultati indipendentemente dal sistema utilizzato, che faccia uso di esempi e caratteristiche diverse per riuscire a classificare correttamente ogni barattolo esaminato.

Capitolo 8

Analisi in multi-frequenza

Con la Parte 2 della tesi sono stati introdotti i principi generali alla base di un qualunque modello di machine learning e sono state definite le caratteristiche che maggiormente determinano il buon funzionamento della macchina.
Durante il processo di allenamento, i dati in ingresso al sistema vengono utilizzati per scoprire la relazione input-output esistente tra ciascun esempio e la relativa label e la quantità di informazioni che l'algoritmo di ML è in grado di ricavare dal dataset in esame dipende fortemente dagli esempi che lo compongono.

Per aumentare le probabilità che la rete finale sia in grado di discriminare correttamente le diverse classi e di generalizzare la propria funzione su dati mai visti prima, può essere utile aumentare le informazioni contenute all'interno del dataset di allenamento: tra le varie soluzioni, nel caso di studio affrontato in questa tesi è possibile scegliere di:

1. Aumentare il numero di esempi all'interno del dataset;
2. Aggiungere nuove antenne;
3. Acquisire le misure lavorando in multi-frequenza.

La prima strategia prevede di acquisire le matrici di diffusione su un numero sempre maggiore di barattoli così da aumentare la dimensione del dataset e favorire un processo di allenamento più lungo e preciso.

Con le sezioni precedenti è stato però visto come questa soluzione non garantisca il miglioramento delle prestazioni: il modello ottenuto sul secondo dataset (quello con lo shield metallico applicato sull'arco di antenne [6]) ha dimostrato infatti di essere tra quelli con le performance più basse, nonostante il numero di esempi a sua disposizione fosse tra i più alti di tutti.

Le strategie 2 e 3 modificano invece il dataset intervenendo sui singoli esempi: in entrambi i casi è possibile assistere ad un incremento nel numero di features che potrebbe facilitare l'addestramento del modello e permettere all'algoritmo di scegliere i parametri significativi tra un set più ampio di valori.

La differenza tra le due strategie risiede nella tecnica adottata per acquisire le nuove misure: la soluzione 2 basa il suo funzionamento sull'utilizzo di numero maggiore di antenne che, posizionate sopra il nastro di confezionamento, permettono la costruzione di matrici più grandi e l'ottenimento di ulteriori parametri di scattering; tra i principali problemi di questa applicazione, si deve però considerare che:

- Il numero delle nuove features dipende strettamente da quante antenne sono utilizzate in quel dato momento, e rimane limitato a meno che non si aggiungano molti nuovi segnali;
- L'aggiunta di ogni nuova antenna necessita la creazione di un nuovo collegamento con il vector network analyzer: lo strumento utilizzato deve quindi disporre di nuove porte di I/O o essere equipaggiato con una switching matrix sempre più complessa e difficile da progettare.

La terza soluzione invece non prevede l'aggiunta di alcun nuovo componente e non modifica né il costo né la complessità del dispositivo finale: l'idea generale, in questo caso, è quella di acquisire le matrici di diffusione (con le 6 antenne già installate) non solo a 10 GHz, ma su un range di frequenza più o meno ampio.

Facendo variare la frequenza del segnale generato dal VNA tra 9 GHz e 11 GHz, con un passo di 200 MHz, è possibile ottenere per ciascun barattolo 11 diverse matrici di scattering, caratterizzate ognuna dai 30 parametri S_{ij}; affiancando tutti questi valori, si può costruire un unico esempio che conterà 330 features anziché le sole 30 di partenza.
Le misure in multi-frequenza sono state acquisite nello stesso momento in cui sono state rilevate le precedenti informazioni a 10 GHz ed in particolare, i dataset studiati nei capitoli precedenti sono stati estratti dal più ampio set già raccolto in multi-frequenza.

Nelle sezioni seguenti saranno analizzate e valutate le prestazioni di nuove reti neurali, allenate sugli stessi dataset presentati tra i capitoli 5 e 7, acquisiti però in multi-frequenza.
Saranno riportati in particolare tutti i risultati ottenuti durante le diverse analisi, in modo generale per i modelli che permettevano buone performance anche con una singola frequenza, ed in modo dettagliato per le reti che invece non riuscivano a discriminare i barattoli contaminati lavorando solo a 10 GHz.

8.1 Primo dataset

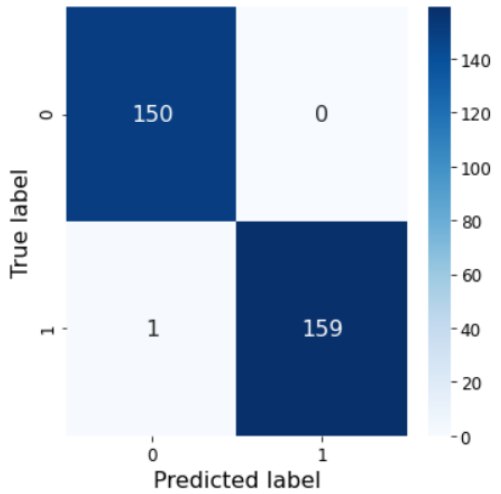
Le prestazioni raggiungibili sul primo dataset, descritto al capitolo 5, sono massimizzate con l'utilizzo del modello presentato nella sezione 5.3 dove, con due livelli nascosti, è possibile ottenere un'accuratezza di 0.99355 sul test set.

I diversi esempi, divisi tra 640 misure con contaminante e 600 casi sicuri, sono correttamente analizzati e discriminati dalla macchina che, nel complesso, commette un totale di 3 errori.

La modifica del dataset e la relativa introduzione di nuove features dovrebbe permettere in casi come questi una riduzione della complessità del sistema, che potrebbe raggiungere o migliorare i risultati utilizzando un numero inferiore di layers e neuroni.
I passaggi eseguiti per costruire ed allenare la nuova rete sono gli stessi visti in precedenza e si basano sulla ricerca bayesiana dei migliori iperparametri e sulla loro applicazione nel modello.

Al termine della fase di addestramento, è possibile quindi verificare le capacità della macchina risultante, valutando i valori delle principali metriche e le matrici di confusione associate a ciascun set di esempi.

In particolare, con una rete neurale costituita da 1 hidden layer e soli 2 neuroni, si ottengono i seguenti risultati:



Accuratezza	:	0.99677
Precisione	:	1.00000
Richiamo	:	0.99375
Errore logaritmico	:	0.11142

Figura 8.1: Matrice di confusione sul test set - primo dataset multi-frequenza

La matrice in Figura 8.1 si riferisce al test set, in cui un falso negativo penalizza leggermente il valore del richiamo.
Le classificazioni su training e validation set sono invece corrette per tutte la misure ed è possibile per questo raggiungere il massimo valore per tutte le metriche.

L'ipotesi iniziale è quindi confermata: utilizzando esempi con 330 features è più facile per il modello riconoscere l'iperpiano utile alla discriminazione delle due classi e la rete necessaria per compiere tale scopo sarà più piccola e semplice.

8.2 Secondo dataset con scudo

Il capitolo 6 descrive i risultati ottenuti sul secondo dataset, acquisito dopo l'applicazione dello scudo metallico sull'arco di antenne.

Nella stessa sezione è possibile verificare come un aumento della velocità del nastro di confezionamento riduca fortemente le capacità della rete neurale che, nonostante utilizzi dati e misure protette dalle onde elettromagnetiche esterne, non riesce a classificare gli esempi con l'accuratezza desiderata.

L'ultimo modello presentato è composto da 4 hidden layers costituiti rispettivamente da 460, 256, 460 e 512 neuroni; nonostante il processo di allenamento riesca a riconoscere tutti i barattoli presenti all'interno del training set, il numero di errori commessi sul validation e sul test set rendono la macchina inutilizzabile (vedi sezione 6.2.3).

In questo caso, una volta modificato il dataset ed aggiunte le informazioni ottenute a frequenze diverse, è utile documentare l'intero lavoro svolto per modellare e costruire la nuova rete neurale.
Ripartendo dalla ricerca bayesiana, saranno quindi proposti i risultati relativi a:

- Ricerca degli iperparametri;
- Addestramento della rete neurale;
- Verifica e calcolo delle metriche.

8.2.1 Ricerca bayesiana

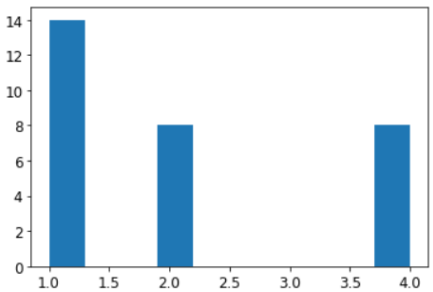
Lo spazio di ricerca è inizialmente ampio e simile a quello utilizzato nelle precedenti ricerche: in questo modo è possibile evidenziare fin da subito le differenze esistenti tra il dataset iniziale ed il nuovo set di esempi, valutando opportunamente il valore degli iperparametri ed il numero di livelli che l'algoritmo ritiene necessario usare.

Successivamente, a fronte dei risultati ottenuti, lo stesso spazio viene ridotto e viene avviata una ricerca più fine sulle seguenti liste:

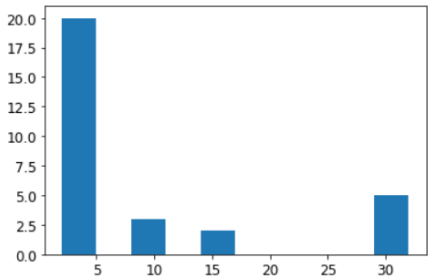
kerasclassifier___n_hidden	:	['1', '2', '4']
kerasclassifier___n_neurons_hidden_1	:	['2', '4', '8', '16', '32'],
kerasclassifier___n_neurons_hidden_2	:	['2', '4', '8', '16', '32'],
kerasclassifier___n_neurons_hidden_3	:	['2', '4', '8', '16', '32'],
kerasclassifier___n_neurons_hidden_4	:	['2', '4', '8', '16', '32'],
kerasclassifier___epochs	:	[700]
kerasclassifier___batch_size	:	[128]
kerasclassifier___learning_rate	:	['0.001']
kerasclassifier___regularization_rate	:	['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate	:	['0.0', '0.01', '0.03']

Al termine del processo, i seguenti valori sono forniti in output:

kerasclassifier___n_hidden	:	1
kerasclassifier___n_neurons_hidden_1	:	16
kerasclassifier___learning_rate	:	0.001
kerasclassifier___regularization_rate	:	0.0
kerasclassifier___dropout_rate	:	0.01



(a) Numero di layers



(b) Primo layer

Figura 8.2: Risultati ricerca bayesiana - dataset con scudo in multi-frequenza

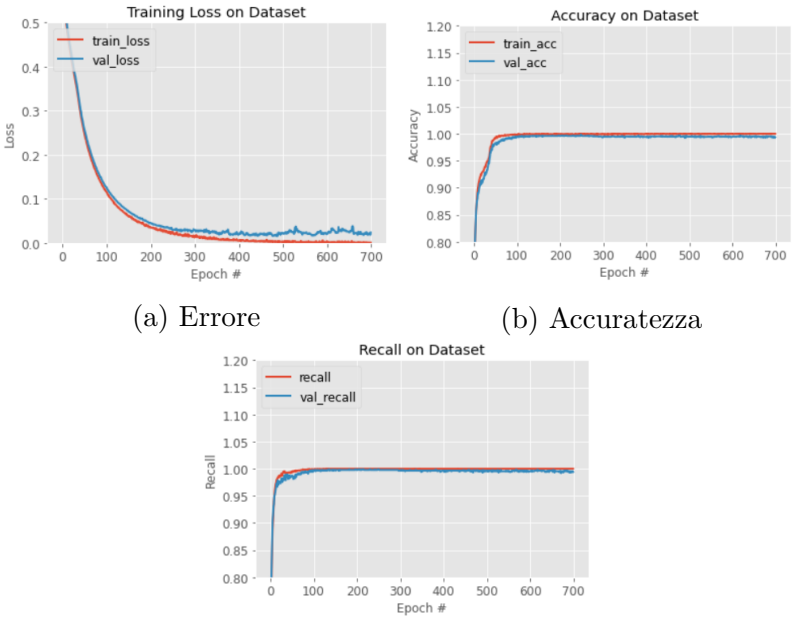
L'aspetto più importante da considerare su questi risultati è il numero di livelli nascosti: da un valore iniziale di quattro, è ora sufficiente un solo layer che, con pochi neuroni, riesce presumibilmente a riconoscere un pattern utile alla discriminazione delle due classi.

8.2.2 Allenamento e validazione

Il processo di addestramento viene avviato su una rete costruita con gli iperparametri sopra riportati e, dopo le 700 epoche indicate nella funzione allenamento, sono valutati i grafici e le prestazioni risultanti.

A fronte dei buoni risultati ottenuti, la rete finale è stata ulteriormente semplificata, fino ad arrivare ad una macchina costituita da 1 hidden layer e 2 neuroni. In questo caso sono state sufficienti poche iterazioni per raggiungere un'accuratezza di 0.99837 sul set di allenamento, valore migliorato per l'ultima volta durante la 249esima epoca su 700.
Proprio il modello salvato in tale epoca è utilizzato per costruire la rete da testare e valutare nella prossima fase.

I grafici in Figura 8.3 mostrano l'andamento delle metriche durante lo stesso processo, suggerendo in tutti i casi la buona riuscita dell'addestramento.



(a) Errore (b) Accuratezza

(c) Richiamo

Figura 8.3: Evoluzione delle metriche - dataset con scudo in multi-frequenza

In particolare è possibile vedere che:

1. Il valore dell'errore sul training set (Figura 8.3a) viene ridotto a zero velocemente, mentre la stessa metrica sul validation set raggiunge un livello discretamente basso: il grado di regolarizzazione L2 scelto sembra essere sufficiente a garantire una generalizzazione per entrambi gli insiemi di esempio;

2. I valori di accuratezza e richiamo convergono invece ad 1: entrambi gli andamenti raggiungono il valore massimo rapidamente e si stabilizzano intorno al 100% con poche oscillazioni.

Confrontando i grafici sopra riportati con quelli mostrati in precedenza in Figura 6.3 (stesso dataset, ma con le informazioni relative solo ai 10 GHz), è possibile verificare i vantaggi del lavorare in multi-frequenza: mentre in quel caso i quattro livelli nascosti della rete non erano ancora sufficienti per raggiungere prestazioni accettabili, la disponibilità di 330 features diverse permette all'algoritmo di allenamento di addestrare facilmente una rete più piccola e semplice.

8.2.3 Verifica

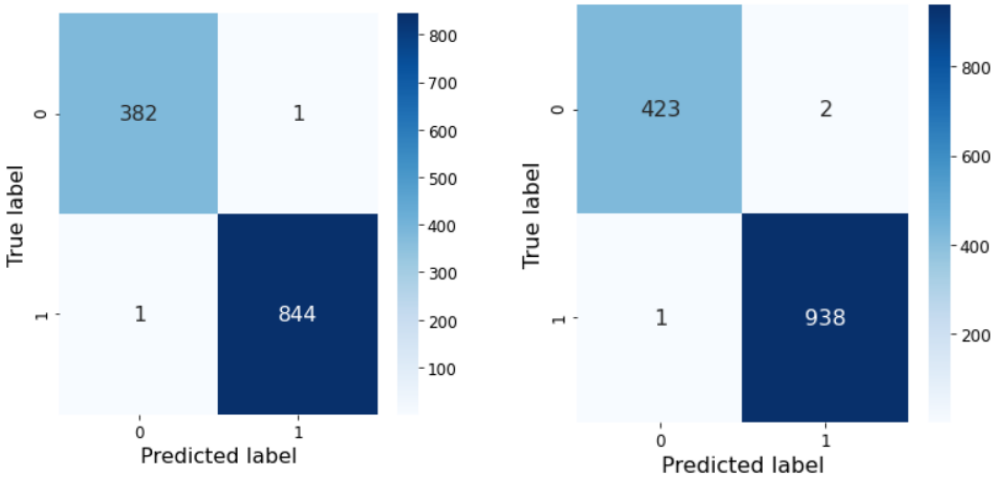
Le matrici di confusione diventano in questo caso lo strumento più significativo per dimostrare l'efficacia del lavorare in multi-frequenza; in modo più evidente rispetto al primo dataset (in cui le prestazioni erano buone già con la singola frequenza), l'ottenimento di buoni risultati su questo insieme avallerebbe l'ipotesi iniziale e permetterebbe di considerare nuovamente la possibilità di acquisire solamente il modulo dei parametri di diffusione.

La prima verifica, effettuata sul training set, è in linea le precedenti esecuzioni e determina una classificazione corretta per tutti i 2863 esempi di allenamento.

Tutti i valori delle metriche raggiungono il 100% (accuratezza, precisione e richiamo), mentre l'errore logaritmico è annullato completamente.

Accuratezza	:	1.00000
Precisione	:	1.00000
Richiamo	:	1.00000
Errore logaritmico	:	0.00000

Proseguendo con il validation set e concludendo con i dati del test set, è possibile valutare le reali capacità della macchina e contare il numero totale di errori commessi.



(a) Validation set

(b) Test set

Figura 8.4: Matrici di confusione - dataset con scudo in multi-frequenza

Accuratezza	:	0.99837
Precisione	:	0.99882
Richiamo	:	0.99882
Errore logaritmico	:	0.05625

(a) Validation set

Accuratezza	:	0.99780
Precisione	:	0.99787
Richiamo	:	0.99894
Errore logaritmico	:	0.07597

(b) Test set

In Figura 8.4 sono riportate le matrici di confusione relative ai due dataset, in cui è evidente il buon funzionamento della macchina: le capacità della rete neurale sono dimostrate dagli ottimi valori delle metriche e dal numero limitato di falsi positivi e falsi negativi.

Confrontando le prestazioni della rete qui descritta con quelle ottenute al capitolo 6.2 è possibile quindi comprendere come l'aggiunta di nuove informazioni sia utile ad un'applicazione di machine learning.
Mentre le 30 features originali non erano sufficienti per caratterizzare le due classi, la ridondanza delle misure a frequenze diverse permette al modello di rilevare il piano n-dimensionale migliore per dividere tra loro gli esempi in ingresso.

In questo caso, risolto il problema dell'aumento di complessità del circuito di acquisizione, sarà possibile implementare la rete neurale in grado di definire la sicurezza di ogni baratolo a partire dal solo modulo dei parametri di diffusione.

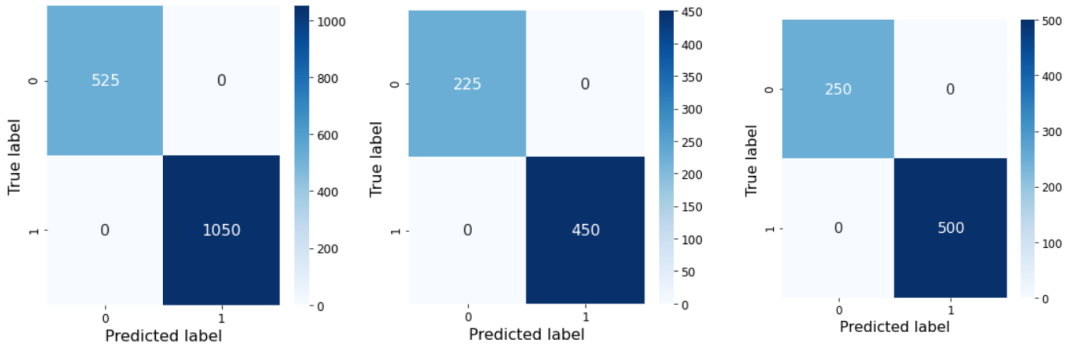
8.3 Terzo dataset: FT System

8.3.1 Velocità di 30 cm/s

Il dataset di FT System acquisito con il nastro di confezionamento a 30 cm/s ha permesso nel capitolo 7.1 di ottenere il modello che più di tutti riesce a dividere gli esempi contaminati da quelli sicuri.

L'applicazione di un numero maggiore di frequenze sarebbe in questo caso superfluo, ma permette in ogni caso di generalizzare l'utilizzo della macchina in situazioni diverse.
É necessario sempre verificare che i nuovi esempi non penalizzino le prestazioni, ma che piuttosto permettano di ottenere risultati analoghi a quelli precedenti senza complicare ulteriormente la macchina.

La ricerca bayesiana condotta su un spazio di ricerca fine ha suggerito la costruzione di una rete composta da 1 livello nascosto e 2 neuroni, con l'applicazione di dropout (0.1) e regolarizzazione L2 (0.001).
Al termine del processo di allenamento è quindi avviata la fase di verifica che si è conclusa con i seguenti risultati:



(a) Training set (b) Validation set (c) Test set

Figura 8.5: Matrici di confusione - FT System 30 cm/s in multi-frequenza

Le tre matrici di confusione in Figura 8.5 si riferiscono agli esempi di addestramento, validazione e test, ottenuti dividendo il dataset iniziale in gruppi rispettivamente di 1575, 675 e 750 misure.

Tutte le predizioni della rete sono confermate dalla verifica finale e la classificazione avviene correttamente per tutti i dati utilizzati; i valori delle metriche

risultanti sono massimi su ciascun insieme ed è per questo possibile proseguire l'analisi considerando l'ipotesi di lavorare con i moduli sempre più valida.

8.3.2 Velocità di 50 cm/s

Il capitolo 7.2 ha documentato il lavoro svolto sul dataset di FT System acquisito con il nastro di confezionamento alla velocità di 50 cm/s.
Al termine dei processi di allenamento e di verifica è stato visto come l'utilizzo del modulo dei parametri di diffusione in singola frequenza non fosse sufficiente ad ottenere dei risultati accettabili e l'ipotesi iniziale della tesi è stata di conseguenza smentita e modificata.

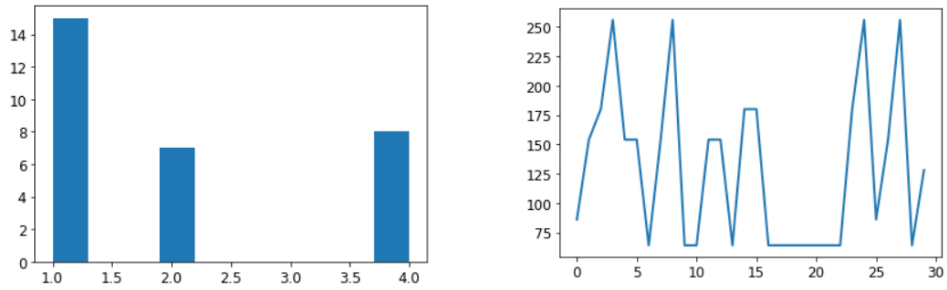
Tutta la sezione 8 prende in esame le analisi effettuate in multi-frequenza e cerca di dimostrare come l'aggiunta di nuove features agli esempi di addestramento possa influire positivamente sulle prestazioni raggiungibili dalla macchina. In quest'ultimo capitolo è quindi descritta la rete ottenuta sullo stesso dataset, lavorando su 330 caratteristiche anziché le 30 originali.

La ricerca bayesiana lavora sulle liste sotto riportate e suggerisce i migliori iperparametri scelti confrontando l'accuratezza ad ogni iterazione.

kerasclassifier___n_hidden	:	['1', '2', '4']
kerasclassifier___n_neurons_hidden_1	:	['32', '64', '156', '180', '256'],
kerasclassifier___regularization_rate	:	['0.0', '0.001', '0.003']
kerasclassifier___dropout_rate	:	['0.0', '0.01', '0.03']

Dopo alcuni tentativi, la rete più semplice in grado di garantire un buon livello di accuratezza è caratterizzata da:

kerasclassifier___n_hidden	:	1
kerasclassifier___n_neurons_hidden_1	:	64
kerasclassifier___regularization_rate	:	0.00
kerasclassifier___dropout_rate	:	0.3



(a) Numero di layers (b) Primo layer

Figura 8.6: Risultati ricerca bayesiana - FT System 50 cm/s in multi-frequenza

A differenza dei modelli ottenuti con i dataset precedenti, gli iperparametri suggeriti in questo caso definiscono una rete leggermente più complessa, in cui 64 neuroni sono necessari per costruire il primo ed unico hidden layer; nel confronto con il classificatore del capitolo 7.2, è comunque evidente il miglioramento complessivo che permette di ridurre fin da subito il numero di livelli e presumibilmente di errori commessi durante l'elaborazione.

L'andamento delle metriche durante l'addestramento fa supporre delle ottime performance sul training set, con un errore leggermente maggiore sugli esempi di validazione; le 800 epoche impiegate per far convergere la funzione di costo sembrano garantire una buona stabilità, generalizzando l'operazione appresa tra tutti i dati analizzati.

(a) Errore (b) Accuratezza (c) Richiamo

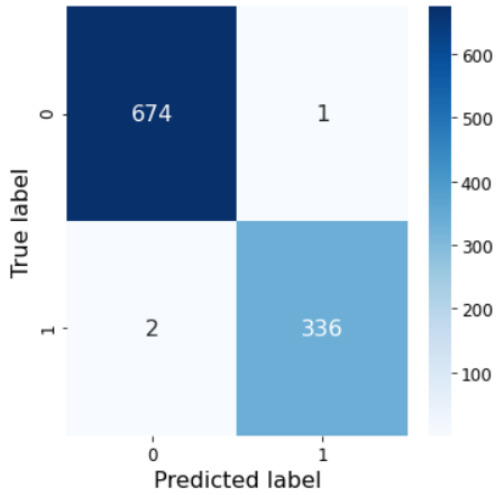
Figura 8.7: Evoluzione delle metriche - FT System 50 cm/s in multi-frequenza

La fase di verifica di quest'ultima rete neurale porta a conclusione il capitolo relativo al machine learning, validando definitivamente l'ipotesi di poter rilevare i contaminanti alimentari a partire dal modulo dei coefficienti di scattering.

- Il training set, composto da 2363 esempi, garantisce come negli altri casi le

prestazioni massime, permettendo al modello di collocare ogni barattolo nella giusta categoria di appartenenza.

- Il validation set interviene nella fase di addestramento per calcolare e confermare il miglioramento delle performance al termine di ogni iterazione e la relativa matrice di confusione conferma la buona riuscita dell'allenamento: i 1013 esempi vengono classificati in modo corretto ad eccezione di un falso positivo e due falsi negativi.

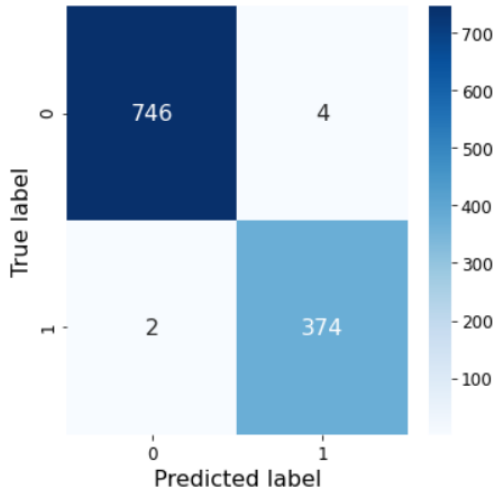


Accuratezza	:	0.99704
Precisione	:	0.99703
Richiamo	:	0.99408
Errore logaritmico	:	0.10229

Figura 8.8: Confusion matrix validation set - FT System 50 cm/s multi-frequenza

Le metriche mantengono un valore elevato che rimane conforme alle necessità del caso di studio; sono evidenti le differenze rispetto alla verifica del capitolo 7.2.4 nella quale venivano commessi diversi errori già nel set di addestramento.

- Il test set fornisce infine l'ultima verifica delle capacità della macchina e permette di confermare definitivamente i vantaggi del lavorare in multi-frequenza.



Accuratezza	:	0.99467
Precisione	:	0.98942
Richiamo	:	0.99468
Errore logaritmico	:	0.18405

Figura 8.9: Confusion matrix test set - FT System 50 cm/s multi-frequenza

I sei errori commessi suggeriscono una maggiore difficoltà nel classificare gli esempi appartenenti a questo dataset, ma le prestazioni raggiunte dopo aver acquisito le informazioni a frequenze diverse sono accettabili e decisamente migliori rispetto al caso precedente.

Riuscendo ad implementare la rete neurale appena studiata in un dispositivo in grado di misurare e analizzare le matrici di diffusione dei barattoli sotto esame, è possibile sostituire il vector network analyzer attualmente utilizzato e ridurre i costi generali del prototipo.

Dopo aver dimostrato che la singola frequenza non è in grado fornire tutte le informazioni necessarie alla discriminazione delle due classi (specialmente con il nastro ad una velocità di 30 o 50 cm/s), l'espansione degli esempi e l'aggiunta di nuove features ha aperto alla possibilità di limitare l'acquisizione delle misure ai soli moduli dei parametri di diffusione.

Rimane comunque valida l'ipotesi di poter lavorare a 10 GHz, nel caso in cui la rete neurale utilizzata è progettata per analizzare contemporaneamente la parte reale e la parte immaginaria dei coefficienti di scattering, come dimostrato dagli ottimi valori di accuratezza raggiunti in un precedente lavoro del gruppo di ricerca [26].

Tutti i modelli ottenuti e presentati nel capitolo 8 offrono buone prestazioni se applicati ai dataset di riferimento e permettono di proseguire l'attività di tesi con

il progetto di una possibile nuova implementazione hardware.

La successiva Parte IV descrive e segue lo sviluppo del circuito per la ricetrasmissione dei segnali, insieme con il confronto dei diversi datasheet letti ed analizzati per scegliere i migliori componenti che saranno utilizzati per realizzare ed infine simulare il prototipo finale.

Parte IV

Implementazione hardware

Capitolo 9

Progetto del circuito

9.1 Descrizione generale

Con la confermata possibilità di rilevare gli agenti contaminanti tramite l'analisi del modulo dei parametri di diffusione, è possibile proseguire il lavoro di tesi con il progetto del circuito necessario all'acquisizione ed elaborazione dei segnali.

L'esclusione delle informazioni di fase dai dati da misurare permette la semplificazione del dispositivo e la sostituzione del vector network analyzer con strumenti più specifici ed economici.
La scelta dei componenti è effettuata in funzione delle caratteristiche da rispettare ed è facilitata dalla possibilità di scomporre il problema in parti che si occupano di operazioni diverse; in particolare, è necessario avere, all'interno del sistema generale, un circuito specifico per:

- Generazione del segnale;
- Selezione dell'antenna per la trasmissione del segnale;
- Selezione dell'antenna per la ricezione del segnale;
- Conversione della potenza RF in una tensione DC;
- Calcolo del parametro di diffusione e salvataggio della matrice.

Tutte queste operazioni devono inoltre essere gestite e sincronizzate con un controllore esterno, presumibilmente lo stesso in cui verrà implementata la rete neurale per l'analisi dei dati ottenuti.
Prima di entrare nel dettaglio dei singoli componenti, è necessario condurre una ricerca per definire la strategia ottimale da adottare per poter acquisire il modulo dei parametri di scattering.

Mentre il VNA è in grado di misurare automaticamente sia la parte reale che la parte immaginaria dei coefficienti, altri strumenti dovranno essere impiegati per calcolare il modulo a partire da grandezze e quantità diverse.
In particolare, è stata verificata la possibilità di utilizzare:

Scalar Network Analyzer

A differenza del VNA, lo *Scalar Network Analyzer* (SNA) non fornisce informazioni sulla fase e permette di ottenere il modulo richiesto con le stesse modalità con le quali sono effettuate le misure nell'attuale prototipo [27].
La generazione del segnale su una prima porta e la ricezione della risposta su di una seconda permette allo strumento di calcolare la quantità ricercata in modo completamente trasparente per l'utente esterno.

Con lo sviluppo dei più moderni VNA, gli scalar vector analyzer stanno però divenendo obsoleti e, nonostante il guadagno economico tra i due dispositivi sia significativo, rimangono limitanti per l'applicazione studiata in questa tesi.

Analizzatore di spettro & Tracking generator

Il funzionamento è analogo a quello dell'SNA; il tracking generator permette di generare un segnale alla stessa frequenza a cui l'analizzatore di spettro sta ricevendo e, se il DUT è posizionato tra i due componenti, lo spectrum analyzer permette di rilevare le variazioni di ampiezza e di acquisire il modulo dei segnali in ingresso [28].

Anche in questo caso però, gli strumenti utilizzati non sono specifici per l'applicazione in esame e potrebbero essere impiegati per svolgere molti altre operazioni. I costi totali rimangono inoltre superiori a quelli desiderati.

Rapporto tra potenze

Non essendo necessario acquisire i parametri di auto-diffusione (quelli in cui una singola antenna è utilizzata sia per trasmettere che per ricevere), è possibile trascurare i segnali riflessi alle due porte di I/O e calcolare quindi il rapporto tra la potenza inviata e quella ricevuta dopo aver attraversato il barattolo sotto esame.

Il rapporto sarà pari al quadrato del modulo del parametro di diffusione, a meno di un fattore aggiuntivo proporzionale al rapporto tra le impedenze caratteristiche.

$$\mid S_{ij} \mid^2 \propto \frac{\text{Potenza ricevuta da un carico } Z_0}{\text{Potenza trasmessa dalla sorgente con carico } Z_0}, \text{ con i} \neq j$$

Riuscendo quindi a misurare due valori di potenza ed elaborando i dati raccolti, è possibile ottenere il modulo di ciascun coefficiente, impiegando questa volta dei dispositivi economici ed ottimizzati per compiere una sola azione specifica [29].

La rilevazione di potenza può essere effettuata con dei rilevatori a microonde, dei diodi raddrizzanti in grado di convertire il segnale RF in ingresso in una tensione in continua, che potrà essere facilmente misurata e quindi discretizzata da un convertitore analogico-digitale.

Per la gestione delle varie combinazioni è necessario progettare un circuito per reindirizzare i segnali sulle antenne desiderate e, in funzione del percorso scelto, una specifica uscita di un multiplexer dovrà essere attivata per trasmettere il segnale al controllore.

La Figura 9.1 rappresenta in modo generale una possibile implementazione del dispositivo, in cui i vari colori permettono di distinguere le parti dedicate a funzioni diverse.
In particolare:

- rosa: Circuito per la generazione del segnale;
- giallo: Componenti utili per indirizzare il segnale verso le antenne;
- arancione: Antenne per la ricetrasmissione di tutti i segnali;
- verde: Circuito di ricezione e analisi della potenza;
- azzurro: Selezione e conversione del segnale.

É assente in questo schema il controllore finale che si dovrà occupare della generazione dei segnali di controllo per tutti i componenti, della ricezione ed il calcolo dei parametri di diffusione.

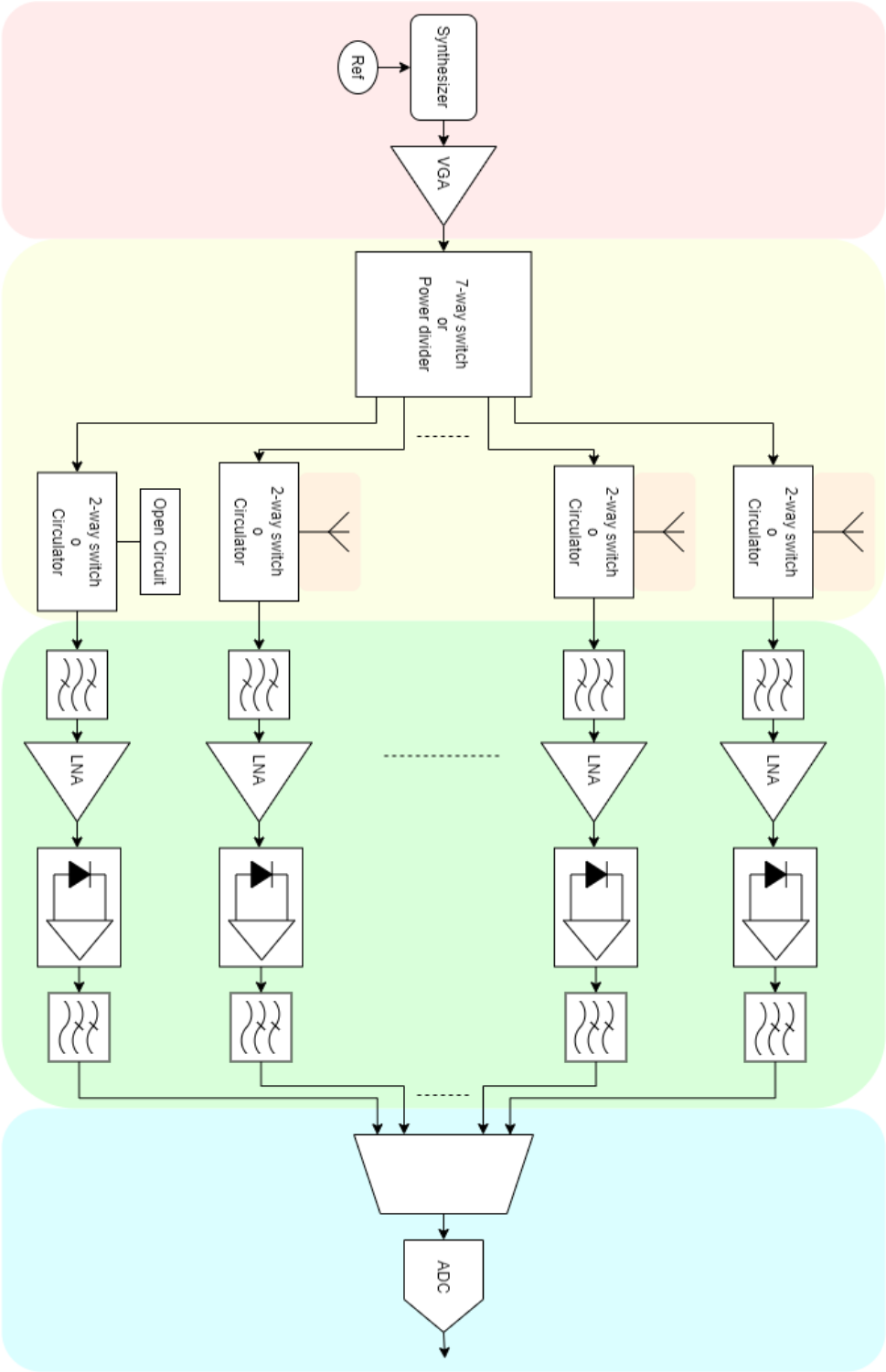


Figura 9.1: Circuito per la generazione e la ricetrasmissione dei segnali

Il circuito è progettato per permettere al segnale generato dal sintetizzatore di essere trasmesso ciclicamente da ognuna delle sei antenne ed essere ricevuto, in modo altrettanto regolare, da tutte le altre; attivando ed alternando la ricezione tra tutti i possibili rami, sarà possibile calcolare la totalità delle combinazioni che compongono le matrici di diffusione.

Dopo aver sintetizzato il segnale alla frequenza desiderata ed averne amplificato la potenza (sezione rosa in Figura 9.1), occorre indirizzare la trasmissione verso una specifica antenna; oltre ai sei collegamenti con le antenne fisiche, sarà necessario aggiungere un'ulteriore via per poter misurare la potenza trasmessa ad ogni istante ed utilizzarla nel rapporto finale (sezione gialla).

Per poter selezionare e controllare l'output corretto ad ogni occorrenza, è stato analizzato e confrontato l'utilizzo di due componenti; nello specifico:

- 7-way switch (SP7T);
- power divider.

Nel primo caso, l'interruttore a 7 posizioni permette un maggiore isolamento tra i diversi percorsi ma necessita di un segnale di controllo che gestisca e sincronizzi tutte le operazioni; il divisore di potenza, invece, lavora in modo completamente combinatorio ed istantaneo, ma rende necessaria la generazione di una potenza maggiore per garantire la corretta ricezione del segnale.

Ogni antenna è utilizzata sia per la trasmissione che per la ricezione dei segnali: per selezionare l'operazione specifica ad ogni ciclo, un componente con tre porte deve essere aggiunto e controllato; nel caso di una trasmissione, infatti, l'antenna dovrà essere collegata al sintetizzatore per ricevere il segnale da inviare mentre, nel caso di una ricezione, la potenza in arrivo dovrà essere inviata direttamente al circuito di misurazione. Per questo scopo sono stati presi in esame:

- 2-way switch (SPDT);
- circolatori.

I circolatori permetterebbero, come nel caso dei power divider, di semplificare il circuito di controllo, ma non saranno trattati in questa tesi perché non garantirebbero la copertura completa della banda richiesta dall'applicazione.

La sezione successiva elenca i componenti analizzati nel corso della tesi, confrontando le diverse caratteristiche e cercando di selezionare le più adatte ad essere applicate al progetto in esame.

9.2 Scelta dei componenti

Una volta definito lo schema generale da dover seguire ed implementare per trasmettere i segnali e misurare la potenza riflessa dal barattolo, è necessario selezionare i componenti più adatti prima di poter procedere con la reale costruzione del nuovo prototipo.

Per garantire il funzionamento del dispositivo sono state analizzate le principali specifiche da rispettare in fase di progetto e, per tutte le scelte, sono stati confrontati i datasheet di diversi componenti; tra tutti gli aspetti da considerare, quelli che maggiormente hanno guidato questa fase sono:

1. specifiche elettriche (tensione, corrente, potenza del segnale);
2. isolamento tra i diversi canali;
3. velocità di switch (per gli eventuali interruttori e multiplexer);
4. precisione del ADC e frequenza di campionamento.

Partendo dal circuito dedicato alla generazione del segnale, sono riportati i parametri di maggiore interesse ed è presentato, per ogni sezione, il principio generale che ha portato alla scelta di un particolare componente.

9.2.1 Generatore di segnale

La parte rosa dello schema in Figura 9.1 caratterizza il circuito di generazione del segnale. Nel caso di un'applicazione generica, tutti i componenti al suo interno potrebbero essere sostituiti ed implementati con un singolo *signal generator*, capace di fornire per ogni uscita una particolare onda diversa per forma e frequenza.

Nel tentativo di limitare fin da subito i costi ed applicare dei componenti dedicati alla singola operazione da svolgere, nel caso specifico di questa tesi, è possibile scomporre il circuito e considerare l'utilizzo di un sintetizzatore a microonde.
Per la valutazione dei possibili strumenti, sono state considerate:

- banda di frequenza;
- tensione di alimentazione;
- potenza del segnale generato;
- possibilità di essere controllato elettricamente.

Il progetto prevede di utilizzare delle frequenza comprese tra 2 GHz e 11 GHz. La scelta di allargare la banda verso valori più bassi rispetto a quelli descritti in questa tesi deriva dall'intenzione di applicare, in futuro, lo stesso prototipo su sostanze alimentari diverse.

É stato visto nei capitoli introduttivi come la frequenza del segnale dipenda fortemente dalla sostanza analizzata e, in particolare, dal valore della sua costante dielettrica: per barattoli contenenti cioccolata (o sostanze oleose) è necessaria una frequenza intorno ai 10 GHz, relativa ad un valore di ϵ_r di circa 2.86; le sostanze con base acquosa sono invece caratterizzate da una costante dielettrica maggiore (78.4 a 25°C per l'acqua pura), alla quale deve corrispondere una frequenza più bassa per mantenere le stesse capacità di penetrazione.

Avendo bisogno di modificare i parametri durante l'acquisizione delle misure, per rilevare i dati in multi-frequenza, è necessario disporre di alcuni pin di controllo, accessibili tramite pc o con un collegamento diretto al microcontrollore: per quanto riguarda questo aspetto, sarà valutata la semplicità con il quale potrà essere effettuata la modifica, la velocità di switch ed il protocollo necessario per la comunicazione.

Analog Devices ADF5356

Il primo dispositivo analizzato per questo scopo è il sintetizzatore *ADF5356* di Analog Devices, il cui datasheet è riportato in [30].

Si tratta di un sintetizzatore microonde a banda larga, con un VCO (*Voltage Controlled Oscillator*) integrato; la relativa scheda di valutazione *EVAL-ADF5356* è raffigurata in Figura 9.2.

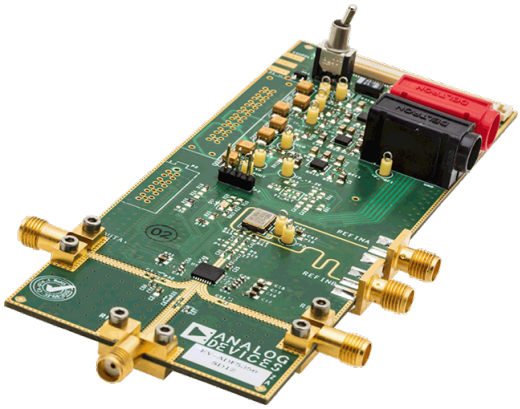


Figura 9.2: Scheda di valutazione EVAL-ADF5356

Il circuito, integrato nel chip, permette l'implementazione di un PLL (*Phase-Locked Loop*) N-frazionale ed uno N-intero che, insieme con il VCO, permettono di coprire una banda di frequenza tra 53 MHz e 13 GHz.
Le specifiche tecniche sono riportate nel datasheet e sono qui in parte estratte per evidenziare gli aspetti di maggiore interesse:

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Segnale di riferimento	10		250	MHz	Single-ended
	10		650	MHz	Differential
Frequenza generata	53.125		6800	MHz	RF_{outA}
	6800		13600	MHz	RF_{outB}
RF output power		8		dBm	RF_{outA}, 1 GHz
		-1		dBm	RF_{outA}, 6.8 GHz
		0		dBm	RF_{outB}, 6.8 GHz
		2		dBm	RF_{outB}, 13.6 GHz
Power supply	3.15	3.3	3.45	V	
VCO power supply	4.75	5.0	5.25	V	
Frequenza del clock SPI			50	MHz	

Tabella 9.1: Specifiche ADF5356

Come è possibile vedere, il sintetizzatore in esame è caratterizzato da due porte di uscita (con connettore SMA), ognuna delle quali fornisce il risultato di uno specifico percorso: RF_{outA} connette l'uscita del VCO ad una serie di divisori e permette di ottenere una frazione della frequenza fondamentale, mentre il segnale RF_{outB} attraversa un moltiplicatore x2 e raggiunge frequenze più alte.

Il segnale di riferimento REF_{in}, così come il loop filter, deve essere collegato dall'esterno per garantire il funzionamento del PLL; la board di valutazione integra già tutti i componenti e contiene in particolare un oscillatore al cristallo a 122.88 MHz (differenziale).

Il controllo del circuito è possibile attraverso una comunicazione SPI (*Serial Peripheral Interface*) che permette di scrivere nei registri interni attraverso il software dedicato; la mappatura di ogni bit è descritta nel datasheet, ma è comunque gestita automaticamente dal programma per pc.

Analog Devices ADF5610

Sempre da Analog Devices è offerto il secondo esempio di un sintetizzatore compatibile con le caratteristiche richieste.

Il sintetizzatore *ADF5610* implementa un PLL N-frazionale o N-intero quando utilizzato con loop filter ed un segnale di riferimento esterno ed integra un VCO per lavorare ad una frequenza compresa tra 57 MHz e 14.6 GHz.

Le caratteristiche del circuito sono specificate nuovamente nel datasheet [31] e sono analizzate per verificare la reale compatibilità con il caso di studio.

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Segnale di riferimento	DC	50	350	MHz	
Frequenza generata	7300		14600	MHz	RF_{out}
	57		14600	MHz	PDIV, NDIV
RF output power	0	5	10	dBm	Tutte le frequenze
Armoniche (RF_{out})		-20		dBc	Seconda
		-30		dBc	Terza
Power supply	3.1	3.3	3.5	V	
VCO power supply	4.75	5.0	5.25	V	

Tabella 9.2: Specifiche ADF5610

La scheda di valutazione associata al circuito è la *EVAL-ADF5610* in Figura 9.3 ed implementa, insieme con il sintetizzatore, un oscillatore a 50 MHz, il loop filter (4 kHz) e l'interfaccia USB necessaria alla comunicazione.

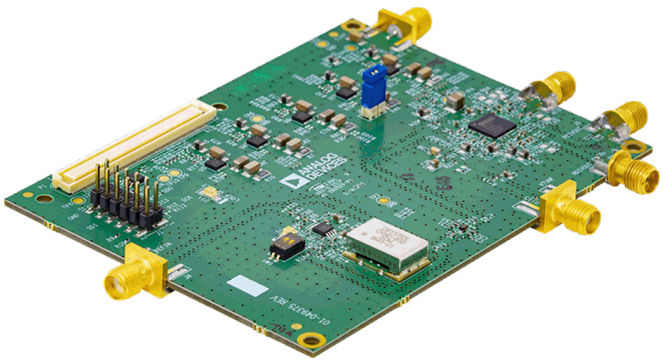


Figura 9.3: Scheda di valutazione EVAL-ADF5610

L'uscita RF_{out} rappresenta l'output del VCO e può essere utilizzata per frequenze comprese tra 7 e 14 GHz; nel caso in cui fosse necessario scendere ulteriormente di banda sono disponibili due ulteriori uscite complementari (PDIV e NDIV) che collegano lo stesso segnale RF_{out} ad una serie di divisori variabili.

Anche in questo caso è disponibile un software dedicato per la gestione di tutte le impostazioni ed il controllo generale è affidato ad un computer.

I due componenti presentati hanno prestazioni simili l'uno all'altro, con la differenza architetturale che il secondo è in grado di fornire tutte le frequenze sulla stessa uscita PDIV.
La selezione dei diversi parametri può essere integrata in entrambi i casi impostando correttamente le stringhe da scrivere nei registri, scambiando le informazioni tramite il microcontrollore ed attendendo la conclusione dell'operazione richiesta.

Paragonando i due componenti, la potenza dei segnali da loro generati varia su due range confrontabili, tra gli 0 dBm ed i 10 dBm: il livello che sarà necessario trasmettere dipende sia dalla frequenza operativa della macchina che dal valore atteso dagli strumenti a valle e sarà fondamentale studiare la propagazione e l'attenuazione interna al circuito per permettere al rilevatore di essere sensibile a variazioni sufficientemente piccole tra le diverse misure.

9.2.2 7-Way power divider & SP7T

La sezione gialla nello schema di Figura 9.1 permette al segnale generato ed amplificato nella parte precedente di raggiungere le antenne per essere trasmesso.

Le specifiche principali da valutare in questo capitolo sono:

- Banda di frequenza;
- Isolamento elettrico tra i diversi canali;
- Insertion loss;
- Velocità di commutazione (per gli interruttori);
- Power supply e tensioni di controllo.

Oltre a fornire il collegamento fisico tra il sintetizzatore e le antenne, i componenti utilizzati in questa parte del circuito devono essere in grado di costruire una rete di controllo e fare in modo che, ad ogni istante, uno solo dei canali di uscita venga utilizzato per trasmettere il segnale.

Tra tutti i dispositivi utilizzabili per indirizzare la potenza su una specifica via, sono analizzati in questa tesi solo i *divisori di potenza* e gli *interruttori*, entrambi con almeno 7 vie per garantire anche la misurazione della potenza trasmessa.

- I power divider permettono di dividere la potenza in ingresso in modo uniforme tra tutti i canali e non necessitano di alcun controllo; il valore della potenza iniziale deve essere scelto in modo accurato per essere sicuri di riuscire a rilevare il segnale dopo ciascuna ricezione;

- Gli interruttori sono in grado di connettere il segnale generato ad un singolo canale, ma hanno bisogno di una alimentazione e di un controllo appropriato dall'esterno.

In funzione dei parametri sopra citati sono quindi valutati alcuni componenti e confrontati per scegliere il migliore da utilizzare nel prototipo finale.

SigaTek SP11855, 8-way SMA Power Divider

Tra i divisori di potenza è stato selezionato il modello SP11855 di SigaTek, il cui datasheet è riportato in [32].

Il modello, nello specifico, è in grado di connettere l'unico ingresso ad 8 possibili uscite, tramite connettori SMA in un range di frequenza tra 2 GHz e 18 GHz.

Figura 9.4: SigaTek SP11855 power divider

Le specifiche sono descritte in Tabella 9.3 e rappresentano il comportamento elettrico del dispositivo.

Parametro	**Min**	**Max**	**Unità**
Range di frequenza	2	18	GHz
Insertion loss		2.2	dB
Isolamento	15		dB

Tabella 9.3: Specifiche SP11855

Tutta la banda di frequenze richiesta dalle specifiche del progetto è coperta ed utilizzabile con lo stesso dispositivo e, su ogni canale, avviene una perdita di potenza di massimo 2.2 dB.

Ipotizzando di aver ricevuto su ciascuna antenna una segnale con potenza sufficiente ad essere rilevato e misurato, un possibile problema per i power divider potrebbe essere rappresentato dal valore dell'isolamento.
Questa quantità è definita come il rapporto tra la potenza di un segnale in arrivo ad una particolare uscita e quella misurata su una qualsiasi delle altre, ed è utilizzato per valutare quanto bene un segnale rimane confinato sul proprio canale.

Un isolamento di 15 dB potrebbe essere sufficiente se il power divider fosse seguito da degli interruttori SPDT (che permettono isolamenti maggiori), mentre potrebbe non essere in grado di limitare i segnali provenienti da circolatori o altri componenti dalle prestazioni più basse.

Nel caso in cui la potenza in ingresso fosse superiore al valore di isolamento fissato dal componente, ci sarebbe il rischio di avere trasmissioni spurie da parte di più antenne e tutte le misure acquisite in tale circostanza non sarebbero più valide e significative.

Tenendo in considerazione di dover generare un segnale di controllo composto da almeno 7 bit, è utile analizzare anche le prestazioni degli interruttori che permetterebbero, nel caso fosse possibile applicarli, un isolamento sicuramente migliore.

JFW 50S-1785, SP8T switch

Il primo interruttore proposto è il SP8T *50S-1785* prodotto da JFW Industries.

É un interruttore allo stato solido con 8 possibili uscite, controllato elettricamente da tensioni collegate dall'esterno [33].

In questo caso sarà necessario valutare anche la velocità di switch, nonché i livelli di tensione da utilizzare per comandare ciascun canale.

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Frequenza del segnale	500		18000	MHz	
Insertion loss		3.3		dB	0.5 - 8 GHz
		4.5		dB	8 - 18 GHz
			5.0	dB	0.5 - 18 GHz
Isolamento		90		dB	0.5 - 8 GHz
		85		dB	8 - 18 GHz
	65			dB	0.5 - 18 GHz
Velocità di switch		75	100	ns	50% -> 10% o 90%
Potenza RF in ingresso			+27	dBm	
Logica di controllo	0		0.8	V	Basso
	2.0		5.0	V	Alto

Tabella 9.4: Specifiche JFW 50S-1785

Anche in questo caso la banda di frequenze richiesta dall'applicazione è totalmente coperta, ma si hanno specifiche leggermente diverse in funzione delle operazioni effettuate.

Nonostante la perdita associata all'insertion loss sia leggermente maggiore rispetto al precedente power divider, in questo caso è possibile trasmettere una potenza maggiore senza rischiare accoppiamenti con gli altri canali; il valore di isolamento raggiunge i 90 dB per frequenze comprese tra 0.5 GHz e 8 GHz, garantendo un disaccoppiamento quasi totale.

I segnali di controllo, uno per canale, sono interpretati secondo una logica TTL (*Transistor-Transistor Logic*): con un'alimentazione standard di 5V, un segnale sarà definito *basso* se la sua tensione è limitata tra 0V e 0.8V, *alto* se invece sarà compreso tra 2V e 5V.

La richiesta di modifica della condizione di una qualsiasi uscita (aperta o chiusa) è accompagnata da un intervallo di tempo in cui l'interruttore cambia stato: il *tempo di switch* deve essere valutato tenendo in considerazione il tempo a disposizione per effettuare tutte le acquisizioni desiderate.

Il SP8T considerato dovrà collegare il segnale generato dal sintetizzatore ad una singola antenna per volta, variando la posizione dell'uscita un totale di 6 volte nel corso dell'elaborazione.
In questo caso non sarà necessaria una velocità eccessiva ed anche 100 ns del 50S-1785 sono sufficienti a completare l'operazione nei nei tempi richiesti.

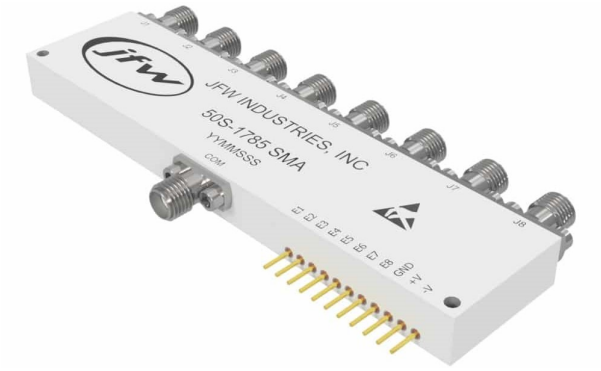


Figura 9.5: JFW 50S-1785 SP8T

PMI P7T-0R8G18G-60-T-SFF-SMC, SP7T

Il secondo esempio di switch è prodotto da PMI (*Planar Monolithics Industries*) ed è identificato dal part number P7T-0R8G18G-60-T-SFF-SMC [34].

In questo caso si tratta di un interruttore allo stato solido a 7 vie, specifico per collegare tutti i canali del progetto.
Come il precedente, è un dispositivo attivo e richiede un'alimentazione esterna duale per garantire il corretto funzionamento ed il controllo adeguato.

Dal datasheet e dal test report è possibile estrarre le specifiche in Tabella 9.5.
Tutti i parametri sono confrontabili con quelli in Tabella 9.4 e descrivono in modo generale le caratteristiche di tutti gli switch allo stato solido.

Rispetto all'interruttore di JFW, questo modello ha un valore di isolamento inferiore, compensato però da una maggiore velocità di commutazione.
Entrambi i componenti sono adatti all'implementazione del prototipo, ma necessitano di uno studio più approfondito per poter gestire correttamente la posizione dello switch ed i valori di potenza ottenibili ad ogni uscita.

Parametro	Min	Typ	Max	Unità	Condizione
Frequenza del segnale	800		18000	MHz	
Insertion loss			4.3	dB	
Isolamento	60			dB	
Velocità di switch			15	ns	Rise e fall
			75	ns	On e Off
Potenza RF in ingresso			+20	dBm	CW
Logica di controllo	0		0.8	V	Basso -> On
	2.0		5.0	V	Alto -> Off

Tabella 9.5: Specifiche P7T-0R8G18G-60-T-SFF-SMC

9.2.3 SPDT, interruttori a due vie

Come descritto nell'introduzione del circuito, ciascuna delle antenne deve essere utilizzata sia in fase di trasmissione che di ricezione; ogni volta che un'antenna riceve ed invia il segnale RF generato dal sintetizzatore, tutte le altre devono aprire il collegamento con il circuito di rilevazione.

Tra tutti i possibili componenti, è stata verificata la possibilità di utilizzare interruttori SPDT (*Single-Pole Double-Throw*) o circolatori: la seconda scelta è però da escludere in quanto non in grado di coprire interamente la banda con un unico dispositivo.

I parametri considerati per ogni scelta sono analoghi a quelli elencati per gli interruttori a 7 vie, con maggiore attenzione alla velocità di commutazione.
Dovendo acquisire in breve tempo tutte le informazioni a frequenze diverse, è necessario effettuare un totale di 330 commutazioni prima che il barattolo abbia superato l'arco di antenne; il tempo di switch deve essere ridotto il più possibile.

Per le caratteristiche fisiche e costruttive, gli switch elettromeccanici non possono essere utilizzati: il tempo necessario a muovere meccanicamente il braccio dell'interruttore e creare il collegamento è superiore a quello ammesso e non ci sarebbe la possibilità di acquisire tutte le misure.
Sono quindi tenuti in considerazione switch allo stato solido (diodi PIN e MESFET) che garantiscono velocità maggiori se propriamente controllati dalla tensione di controllo.

RF Lambda, RFSP2TR0218G

Il primo componente qui analizzato è il SPDT *RFSP2TR0218G* di RF-LAMBDA.

Figura 9.6: RF-LAMBDA RFSP2TR0218G

É un interruttore coassiale che permette lo switch tra due possibili uscite e copre un range di frequenze tra 2 GHz e 18 GHz; tra le specifiche riportate nel datasheet [35] è possibile evidenziare quelle di maggior rilievo con la tabella seguente.

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Frequenza del segnale	2		18	GHz	
Insertion loss		1.2	1.5	dB	2-8 GHz
		1.6	2.0	dB	8-12 GHz
		2.0	2.5	dB	12-18 GHz
Isolamento	80	90		dB	2-8 GHz
	80	85		dB	8-12 GHz
	80	83		dB	12-18 GHz
Velocità di switch			100	ns	

Tabella 9.6: Specifiche RFSP2TR0218G

Considerando un tempo di commutazione di 100 ns, sufficienti per effettuare e stabilizzare il collegamento, questo switch permette di completare tutte le operazioni in 33 μs. Trascurando il tempo richiesto per acquisire ogni singola misura, con il nastro di confezionamento ad una velocità di 50 cm/s, il barattolo esaminato compirebbe uno spostamento di circa 16.5 μm, compatibile con la variazione massima accettabile prima che i dati non siano più validi.

L'alimentazione richiesta per il funzionamento del componente è di ±5V, la stessa tensione necessaria anche per lo switch a 7 posizioni di PMI.

Anche in questo caso la logica di controllo è TTL compatibile e permette di interpretare il livello di ogni segnale in funzione dei valori di soglia standard visti in precedenza.

Analog Devices HMC1118

Un altro componente con specifiche adatte ad un utilizzo nel progetto è identificato dal part number *HMC1118* di Analog Devices [36].
La board che integra tale circuito è la *EV1HMC1118LP3D* che mette a disposizione i connettori SMA ed i pin dal quale controllare la posizione dell'interruttore.

Figura 9.7: Analog Devices EV1HMC1118LP3D

Le caratteristiche sono simili a quelle del modello precedente ma, in questo caso, un valore minore di insertion loss è associato ad un isolamento più basso; tra le caratteristiche più significative è possibile leggere:

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Frequenza del segnale	0.009		13000	MHz	
Insertion loss		0.5	1.0	dB	9 kHz - 3 GHz
		1.3	2.0	dB	9 kHz - 13 GHz
Isolamento	40	50		dB	9 kHz - 3 GHz
	12	25		dB	9 kHz - 13 GHz
Velocità di switch		0.85		μs	10%/90% RF
		2.7		μs	50% V_{CTRL} to 10%/90% RF

Tabella 9.7: Specifiche RFSP2TR0218G

Il modello richiede un'alimentazione duale di +3.3V e -2.5V ed ammette una tensione di controllo positiva tra 0 e 3.3V; in particolare, i livelli logici vengono interpretati secondo i valori:

- Basso (V_{IL}): tra -0.3V e 0.8V
- Alto (V_{IH}): tra 2.0V e 3.3V

Per quanto riguarda la velocità di commutazione sarà necessario attendere almeno 2.7 μs prima che ogni segnale di uscita possa essere ritenuto valido.
Complessivamente, considerato il processo generale con tutte le 330 acquisizioni, saranno richiesti circa 900 μs ed a questo tempo sarà associato uno spostamento del barattolo di circa 0.5 mm (con velocità del nastro di 50 cm/s).

La scelta del componente da utilizzare in questa parte del circuito dipende quindi da due fattori:

1. Isolation;
2. Insertion loss.

Confrontando tra loro i modelli presentati in questa sezione è possibile distinguere due possibili casi: in una prima applicazione in cui le potenze in gioco raggiungono valori elevati, è importante valutare lo stato di isolamento per ciascuna connessione ed essere sicuri che tutti i segnali ricevuti rimangano propriamente confinati sulla linea a loro dedicata.
Nel momento in cui, invece, tutte le grandezze risultano limitate in range di valori non critici, sarà opportuno prevenire quanto più possibile l'introduzione dall'esterno di rumore e perdite; a seconda del caso considerato, converrà quindi dare priorità ad una caratteristica piuttosto che all'altra.

Nel progetto in esame, entrambi gli aspetti dovranno essere considerati: da una parte, avere un isolamento maggiore permette di lavorare con segnali più facilmente misurabili, ma limitare i contributi esterni è sempre importante per ottenere dei valori più accurati.

La possibilità di utilizzare in serie interruttori a 7 vie e switch a due posizioni riduce notevolmente la probabilità di accoppiamento tra i diversi canali e prevenire il ritorno di potenza verso il circuito di generazione del segnale; rimane comunque importante valutare lo stato e la potenza ai ricevitori e simulare correttamente ciascuna operazione per verificare le reali capacità dei componenti.

9.2.4 Low Noise Amplifier (LNA)

Sul circuito di ricezione, una volta filtrate le informazioni alla frequenza di lavoro, è opportuno amplificare e pulire il segnale per permettere una misurazione quanto più efficace ed accurata possibile.

Gli amplificatori a basso rumore diventano strumenti fondamentali per i processi in cui i segnali non raggiungono livelli di potenza elevati, per i quali è necessario isolare il contributo delle informazioni contenute al loro interno senza degradare il rapporto segnale-rumore.
In casi come questo, i LNA sono in grado di limitare la quantità di rumore aggiunto e permettono di amplificare la sola parte utile del segnale.

Il segnale generato dal sintetizzatore è indirizzato e trasmetto da un'antenna alla volta e ricevuto, durante lo stesso ciclo, da tutte le altre. Dopo aver attraversato il barattolo ed essere stato condizionato dalle proprietà elettromagnetiche della sostanza alimentare, la potenza dell'onda è notevolmente ridotta e non può essere misurata con sufficiente precisione.

La scelta di ciascun amplificatore è spesso legata alle caratteristiche del rilevatore a valle del circuito e deve essere effettuata in modo tale da permettere al segnale risultante di raggiungere valori simili al fondo scala dello strumento di misurazione; solo in questi casi sarà possibile discriminare tra loro due segnali la cui differenza è pari alla variazione minima rilevabile.

Tra tutti i possibili modelli, sono stati valutati e confrontati parametri come:

- Range di frequenza;
- Fattore di amplificazione (gain);
- Cifra di rumore (NF).

Sono qui descritte le proprietà di due possibili LNA che potrebbero essere utilizzati nella costruzione del prototipo finale.

Mini-Circuits ZX60-123LN-S+

Il primo modello è progettato da Mini-Circuits ed è commercializzato con il nome *ZX60-123LN-S+*; riepilogando le specifiche del datasheet [37], è possibile costruire la seguente tabella:

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Frequenza del segnale	0.5		12	GHz	
Noise figure		2.6		dB	0.5 GHz
		2.4		dB	8 GHz
		3.1		dB	12 GHz
Gain		18.4		dB	0.5 GHz
	15.1	16.2	18.4	dB	8 GHz
		14.4		dB	12 GHz

Tabella 9.8: Specifiche ZX60-123LN-S+

Con un fattore di amplificazione di 15 dB permette di aumentare il valore di potenza di circa 30 volte e mantiene il rapporto tra rumore in ingresso e rumore in uscita con una cifra di rumore intorno ai 2.5 dB.

Il componente ha bisogno di un'alimentazione esterna di 12V e può ricevere una potenza massima di 8 dBm (continua) o 23 dBm (per massimo 5 minuti) senza compromettere irrevocabilmente le prestazioni.

Figura 9.8: Mini-Circuits ZX60-123LN-S+

Per quanto riguarda le principali caratteristiche, l'amplificatore in esame sembra adatto ad un utilizzo pratico nel circuito, a meno di garantire la corretta alimentazione durante tutto il processo: non avendo alcun altro componente che richieda 12V di tensione tra quelli presentati in precedenza, potrebbe essere svantaggioso il dover dedicare una linea solo per questo, e conviene valutare l'utilizzo di un dispositivo diverso che accetti un'alimentazione più bassa.

Analog Devices ADL9005

Questo secondo amplificatore di Analog Devices copre un range di frequenze da 0.001 GHz a 26.5 GHz, con caratteristiche leggermente diverse tra le fasce 0.01-14, 14-20 e 20-26 GHz.

Nello specifico, leggendo il datasheet [38], i valori relativi alla prima banda sono:

Parametro	**Min**	**Typ**	**Max**	**Unità**	**Condizione**
Frequenza del segnale	0.01		14	GHz	
Noise figure		2.5		dB	
Gain		15.5	17.5	dB	
Return loss		15		dB	Input
		14		dB	Output

Tabella 9.9: Specifiche ADL9005 - da 0.01 GHz a 14 GHz

Figura 9.9: Analog Devices ADL9005

Anche in questo caso, i valori di guadagno e di rumore sono compatibili con i segnali utilizzati nel progetto ed il componente può essere applicato nel circuito finale; i livelli di return loss, misura che caratterizza l'onda riflessa a ciascuna delle due porte, sono coerenti con gli altri strumenti RF e descrivono una buona caratterizzazione delle linee di trasmissione.

L'alimentazione massima ammessa dal componente è di 7V, con un valore tipico di 5V; il segnale RF deve avere una potenza massima di 22 dBm.
A differenza dell'amplificatore di Mini-Circuits, la tensione richiesta da questo LNA è facilmente ottenibile da una qualsiasi microcontrollore e non ha bisogno di una linea esterna dedicata per garantire il funzionamento.

9.2.5 Power detectors

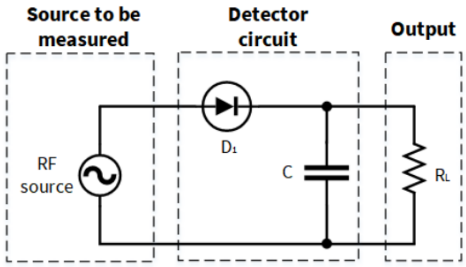
Il componente principale di tutto il circuito di ricezione è rappresentato dal *rilevatore di potenza*; il segnale RF in arrivo su ciascuna antenna deve essere convertito in una grandezza facilmente misurabile ed, in particolare, un determinato valore di tensione in continua deve essere associato ad ogni livello di potenza.

Il principio di funzionamento di un *diode power detector* è legato alla capacità del diodo di rettificare il segnale in ingresso (componente AC) dal quale potrà poi essere estratta la componente DC da misurare.

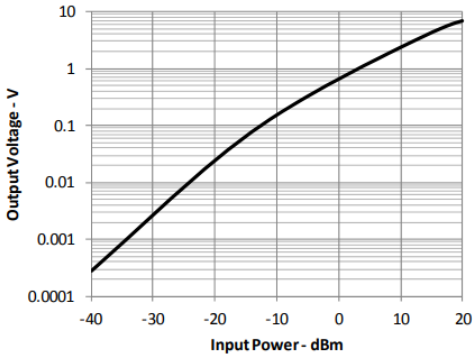
I principali parametri su cui concentrarsi per scegliere il modello più adatto al progetto sono legati agli intervalli di potenza (in input) e tensione (in output) raggiungibili da ciascuno strumento.

Tanto più questi intervalli saranno ampi, tanto più sarà accurata e facilitata la misurazione di ciascun segnale; in particolare, essendo la funzione di conversione lineare (almeno teoricamente, Figura 9.10b), la possibilità di distribuire i segnali su tutto il range massimizza il set di valori distinguibili per ogni uscita.

- Range di frequenza;
- Valori di potenza RF in ingresso;
- Valori di tensione DC in uscita.



(a) Circuito di rilevazione



(b) Caratteristica potenza-tensione

Figura 9.10: Circuito e caratteristica potenza-tensione di un power detector

Confrontando le caratteristiche, sono qui selezionati due possibili componenti: il primo è un dispositivo attivo multicanale, adatto in situazioni in cui il budget economico è limitato, il secondo, completamente passivo, richiede una spesa maggiore, ma garantisce prestazioni ottimali.

Linear Technology LTC5533

Progettato da *Linear Technology*, il power detector *LTC5533* permette di rilevare contemporaneamente due segnali RF nel range di frequenza tra 300MHz e 11GHz. In Figura 9.11 è rappresentata la board di valutazione *DC821A*.

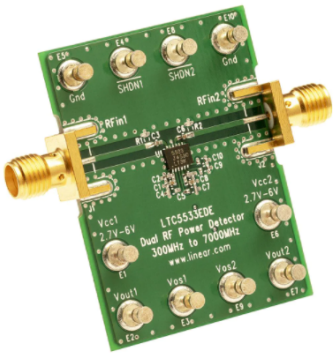


Figura 9.11: Linear Technology DC821A

La presenza di due canali distinti permette di effettuare, nello stesso ciclo, due misurazioni diverse e di ridurre il numero complessivo di componenti necessari a collegare tutte le antenne del circuito.
Pur essendo due percorsi indipendenti, l'implementazione di più canali sulla stessa board fa aumentare la probabilità di avere un accoppiamento, e rende di conseguenza necessario valutare il fattore di isolamento riportato sul datasheet.
Sullo stesso report, è possibile leggere le caratteristiche elettriche e funzionali [39]:

Parametro	**Min**	**Typ**	**Max**	**Unità**
Frequenza del segnale	0.3		11	GHz
Potenza in ingresso	-32		12	dBm
Alimentazione (V_{CC})	2.7		6	V
Tensione in uscita	0		V_{CC}	V
Tensione iniziale	85	110 - 150	170	mV
Isolamento		45		dB

Tabella 9.10: Specifiche LTC5533

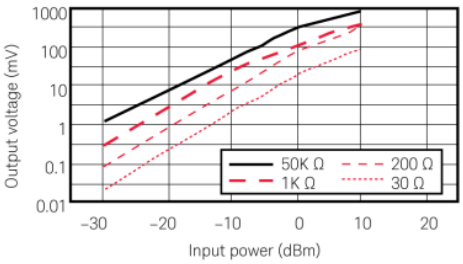
Il valore di isolamento è misurato ad una frequenza di 2 GHz e sembra garantire, con i 45 dB specificati, una buona copertura su tutto il range di potenza in ingresso ai singoli componenti.

L'alimentazione è da applicare in modo indipendente ai due canali di rilevamento e determina il massimo valore raggiungibile dalla tensione in uscita dopo la trasformazione; con uno span di circa 6V, è possibile lavorare in modo sufficientemente preciso su tutti i segnali e permette di discriminare, dopo una eventuale amplificazione, differenze relativamente piccole tra un livello ed un altro.

Keysight 8471E

Il secondo rilevatore di potenza è progettato e distribuito da *Keysight*.
Il modello *8471E*, insieme ad altri della stessa famiglia, è un rilevatore a microonde completamente passivo che implementa dei diodi Schottky per rilevare e convertire una potenza RF.
Il componente specifico mette a disposizione delle porte SMA e SMC per connettere rispettivamente l'ingresso e l'uscita.

(a) Keysight 8471E



(b) Caratteristica potenza-tensione

Figura 9.12: Circuito e caratteristica potenza-tensione di un power detector

Il datasheet riportato in [40] descrive le specifiche tecniche e le condizioni ambientali da adottare durante l'utilizzo e rappresenta graficamente la funzione potenza-tensione tipica del dispositivo (Figura 9.12b).

Il power detector lavora in un range di frequenze compreso tra 0.01 GHz e 12 GHz ed accetta una potenza massima in ingresso di 200 mW (+23 dBm).
Essendo ciascun canale, in questo caso, fisicamente separato da tutti gli altri, non è importante valutare alcun livello di isolamento: nessun segnale (diverso da quello connesso alla porta di ingresso) può influenzare il comportamento del singolo componente.

9.3 Simulazione

Per verificare il corretto funzionamento del prototipo e analizzare il comportamento di ciascun componente prima del suo acquisto, è opportuno effettuare delle simulazioni con software e tools appositi. In questa tesi è stato utilizzato, nello specifico, il programma *AWR Design Environment* di Cadence [41].
Il software permette lo sviluppo di applicazioni RF o a microonde ed è in grado di gestire l'evoluzione di un progetto dagli aspetti fisici della tecnologia (IC, package, PCB), fino alla simulazione e la verifica del circuito completo.
In particolare, il simulatore *Visual System Simulator* (VSS) lavora a livello di sistema e facilita l'acquisizione delle misure per arrivare alla validazione finale.

9.3.1 Circuito ideale

Prima di inserire i componenti con le specifiche scelte nella sezione precedente, sono state effettuate delle prove con dispositivi ideali, per prendere familiarità con il programma ed assicurarsi la corretta rappresentazione del circuito.
É stato quindi costruito il seguente schema elettrico:

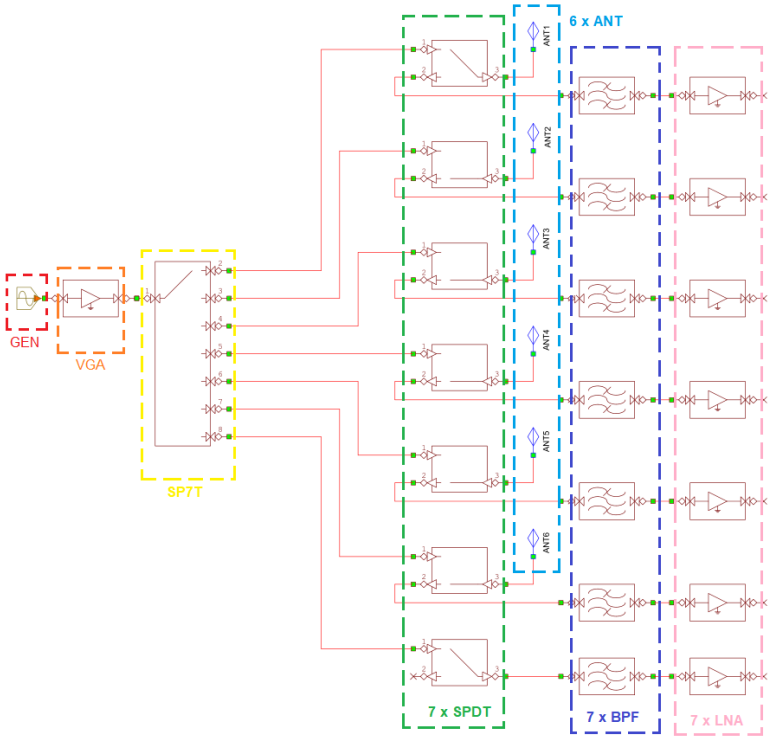


Figura 9.13: Circuito AWR - Componenti ideali

Da sinistra verso destra sono inseriti rispettivamente:

- generatore di segnale (frequenza di 10 GHz e potenza di 8 dBm);
- GEN: amplificatore a gain variabile (16 dBm);
- SP7T: Single-Pole 7-Throw;
- 7 SPDT: Single-Pole Double-Throw;
- 7 BFF: Band Pass Filters;
- 7 LNA: amplificatori a basso rumore (17.5 dBm).

Oltre a questi elementi sono inoltre implementate delle equazioni per simulare i power detectors e le diverse interazioni tra le varie antenne.
Utilizzando i grafici e le tabelle riportate nel datasheet del rilevatore di potenza [39], è stata ottenuta la funzione caratteristica del componente ed utilizzata in AWR per convertire facilmente i valori in dBm in una tensione DC.

Per simulare la presenza di un barattolo di cioccolata e condizionare adeguatamente il segnale da trasmettere, sono stati utilizzati i parametri di scattering di uno degli esempi del dataset di FT System acquisito con il nastro a 30 cm/s: moltiplicando il segnale in uscita dal SPDT attivo sul canale di trasmissione per i coefficienti di diffusione è infatti possibile modificare il valore della potenza ricevuta come se questa fosse realmente attenuata dal prodotto alimentare.
In Figura 9.14 sono rappresentati i moduli di *scaling* applicati alla prima antenna per generare i segnali in arrivo ai canali 2 - 6.

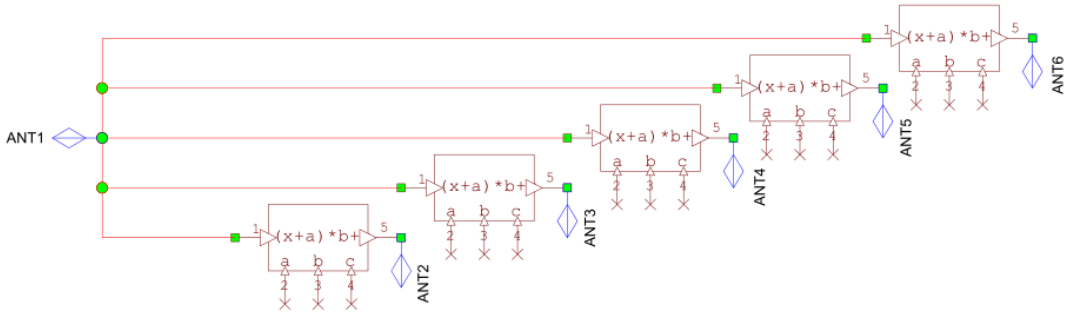


Figura 9.14: Circuito AWR - Matrice di switching per antenna 1

L'idealità di questo primo circuito è data dai valori assegnati ai parametri di *loss* e *isolation*: impostando una perdita di 0 dBm ed un isolamento massimo, è assicurato il fatto che la potenza trasmessa e ricevuta su ciascun canale non sia ridotta a causa dei componenti elettronici, ma sia solo perturbata dalla presenza dell'eventuale barattolo.

Avviando la simulazione e misurando i valori di potenza in uscita dagli amplificatori si prevede, con queste impostazioni, di ottenere parametri di diffusione uguali a quelli acquisiti con il VNA, gli stessi utilizzati per scalare il segnale trasmesso.

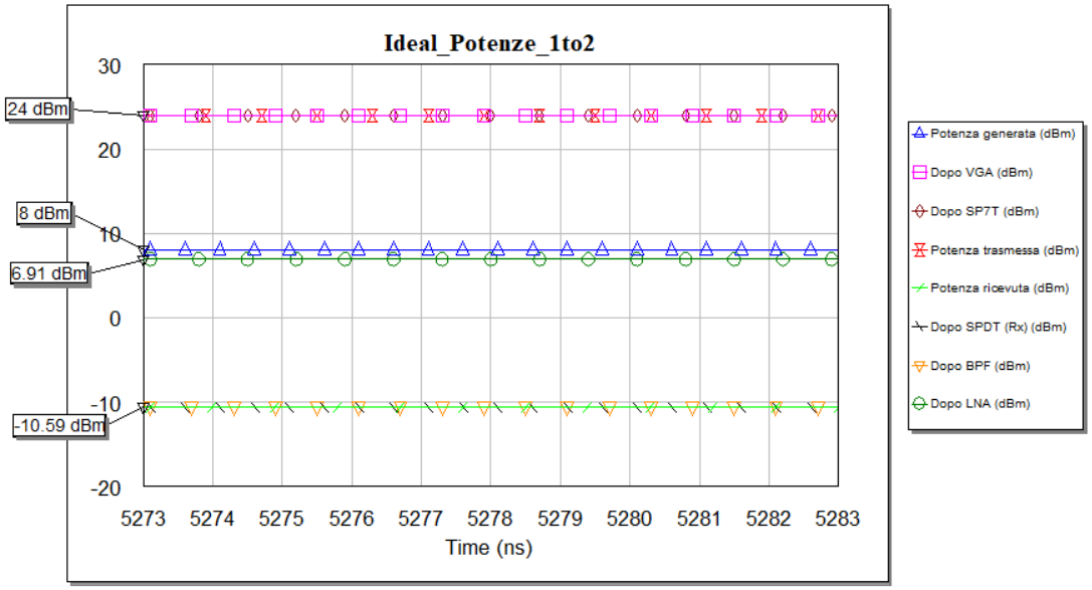


Figura 9.15: Circuito AWR - Potenze misurate sul circuito ideale

Il grafico in Figura 9.15 mostra i valori di potenza misurati all'uscita di ogni componente in funzione del tempo. Come è ben evidente dalla simulazione, gli unici fattori che modificano il livello del segnale sono i guadagni degli amplificatori e la trasmissione dello stesso: generata la potenza RF ed amplificata fino a 24 dBm, questa rimane imperturbata fino all'antenna ricevente, dove arriva ridotta fino circa -11 dBm; l'LNA finale porta il valore da convertire a 6.91 dBm.

Il calcolo dei parametri di diffusione è effettuato in questo caso facendo il rapporto dei valori di potenza in Watt, senza utilizzare per il momento i power detectors: il denominatore, ottenuto indirizzando il segnale da trasmettere sull'ultimo ramo del circuito, è pari a 14.11 W.

Una volta effettuati tutti i rapporti, è possibile costruire la Tabella 9.11.
Ogni riga contiene i moduli dei parametri di scattering S_{ij}, dove l'antenna i trasmette il segnale e l'antenna j lo riceve. Confrontando i valori qui riportati con quelli acquisiti dal VNA per il barattolo selezionato, è possibile verificare che questi coincidono, validando così l'ipotesi iniziale per cui un modello ideale non inserisce perturbazioni aggiuntive.

	A1	**A2**	**A3**	**A4**	**A5**	**A6**
A1	1e-5	0.01865	0.00875	0.01461	0.01294	0.01342
A2	0.01964	1e-5	0.03062	0.00995	0.00690	0.01608
A3	0.00884	0.02974	1e-5	0.02163	0.01293	0.01842
A4	0.01874	0.00969	0.02247	1e-5	0.03598	0.01239
A5	0.01305	0.00874	0.01167	0.03560	1e-5	0.00556
A6	0.01335	0.01131	0.01843	0.01284	0.00530	1e-5

Tabella 9.11: Circuito AWR - Parametri di diffusione con componenti ideali

Siccome i rami di ricezione (incluso il settimo per l'acquisizione del valore di potenza trasmessa) sono formati tutti dagli stessi componenti, utilizzando modelli senza perdite è chiaro il motivo per cui il rapporto rimane invariato: la stessa trasformazione viene applicata sia a numeratore che a denominatore ed il risultato finale non può che essere uguale al valore sperimentale utilizzato per simulare l'interazione tra le antenne.

9.3.2 Circuito reale

Per avere una rappresentazione più realistica del circuito che verrebbe implementato con i componenti fisici scelti nel capitolo precedente, è stata effettuata una nuova simulazione per tenere conto delle non-idealità.

Il sistema costruito su AWR rimane invariato rispetto a quello mostrato in Figura 9.13, ma gli interruttori e gli amplificatori utilizzati in questo caso contengono i valori specificati sui datasheet corrispondenti.
In particolare sono inseriti i contributi di:

- *insertion loss*;
- *isolation*;
- rumore e *noise figure*.

Lo scopo di questa nuova simulazione è quello di verificare che l'inserimento di questi parametri non invalidi le misure acquisite a valle dei power detectors. I coefficienti di scattering utilizzati in questo caso sono gli stessi considerati anche nella precedente simulazione.
La figura seguente rappresenta nuovamente i livelli di potenza registrati sulla catena che va dall'antenna 1 per la trasmissione, all'antenna 2 per la ricezione.

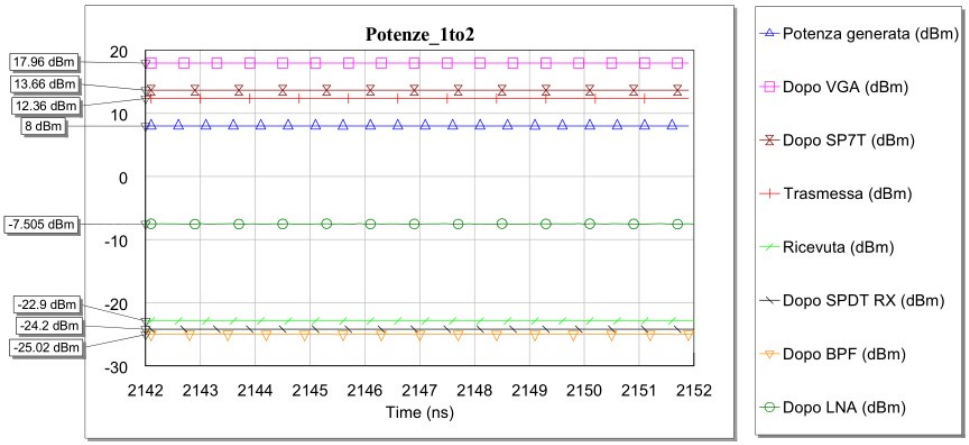


Figura 9.16: Circuito AWR - Potenze misurate sul circuito reale

Come è possibile vedere, i valori misurati dopo ciascun componente variano in funzione della perdita e dell'attenuazione specifica di ogni strumento.
La potenza generata di 8 dBm viene amplificata dal VGA, ma il livello raggiunto all'uscita non è più pari a 24 dBm come nel caso ideale, ma raggiunge i 17.96 dBm; a causa delle perdite introdotte dagli switch, la potenza trasmessa dalla prima antenna è di circa 12 dBm. La riflessione con il barattolo riduce il valore a -23 dBm e il *Low Noise Amplifier* ne amplifica il contributo fino a circa -7 dBm.

Ripetendo le simulazioni per tutte le trasmissioni dei singoli canali, sono acquisiti i valori di potenza ricevuti da ogni antenna. Per poter effettuare il rapporto e calcolare ciascun coefficiente di scattering è stata misurata, sul settimo ramo, la potenza trasmessa pari a 39.11 mW.
La tabella seguente contiene i parametri di diffusione al termine del processo, dopo aver ottenuto il rapporto ed aver effettuato la radice quadrata.

	A1	**A2**	**A3**	**A4**	**A5**	**A6**
A1	0.00032	0.07274	0.03313	0.05701	0.05049	0.05236
A2	0.07661	0.00034	0.11930	0.03882	0.26910	0.06274
A3	0.03451	0.11590	0.00036	0.08434	0.05049	0.07187
A4	0.07312	0.03784	0.08763	0.00034	0.14010	0.04835
A5	0.05093	0.03411	0.04552	0.13870	0.00032	0.02171
A6	0.05209	0.04413	0.07190	0.05011	0.02071	0.00029

Tabella 9.12: Circuito AWR - Parametri di diffusione con componenti reali

Confrontando i parametri ottenuti con quelli sperimentali inseriti in AWR per simulare l'interazione tra antenne e barattolo, è possibile verificare come questi siano uguali a meno di un coefficiente moltiplicativo di 3.9.
Anche in questo caso gli stessi componenti sono utilizzati per acquisire tutte le misure, e le stesse trasformazioni sono applicate sia a numeratore che a denominatore. La presenza di un fattore di scala è collegato direttamente alle non-idealità che influenzano il singolo segnale ma non modificano il comportamento finale.

Un aspetto fondamentale da analizzare per poter ritenere valido il processo di acquisizione è l'isolamento tra i diversi canali: gli interruttori SPDT e SP7T devono essere in grado di confinare la potenza nei percorsi prestabiliti, limitando quanto più possibile la generazione di segnali spuri sui canali idealmente spenti. A questa misura è da aggiungere anche il contributo del segnale riflesso, la cui potenza è proporzionale al valore del *return loss* di ciascun componente.

Per valutare quindi il comportamento del sistema è avviata una nuova simulazione in cui sono monitorati i livelli di potenza sul circuito di ricezione della prima antenna (utilizzata per trasmettere) e sui canali *off* dell'interruttore a 7 posizioni.

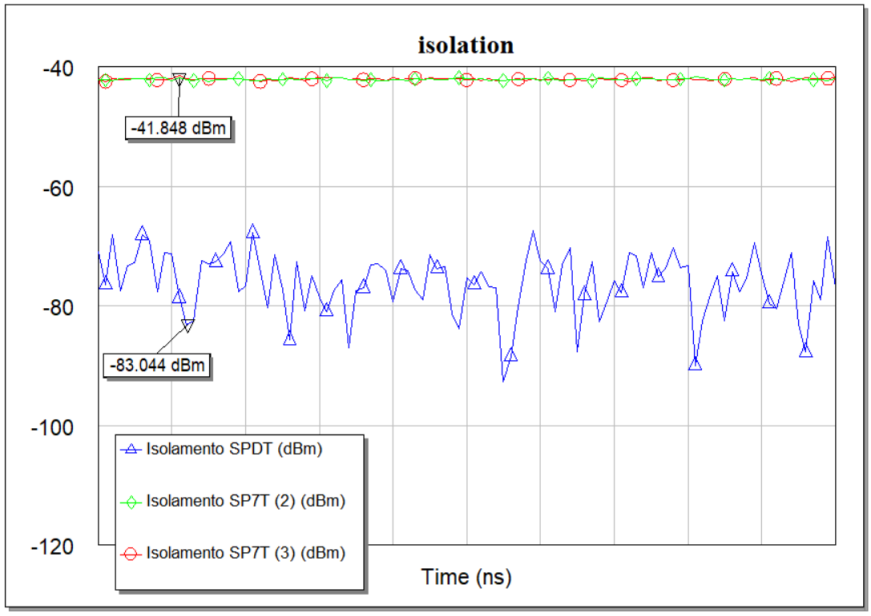


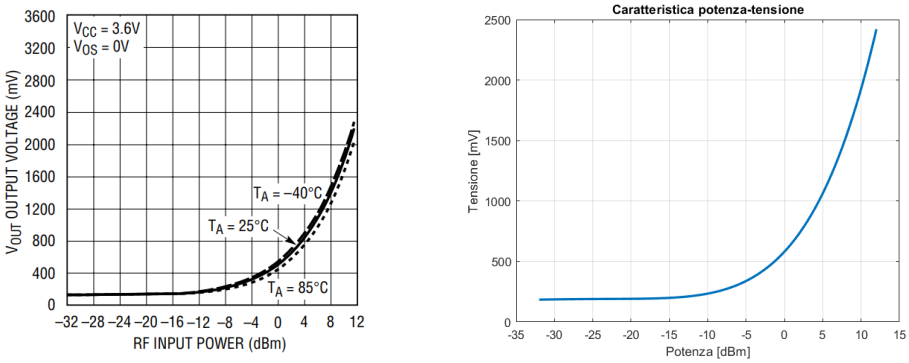
Figura 9.17: Circuito AWR - Isolamento di SPDT e SP7T

Considerando insieme tutti gli effetti, sui canali 2 e 3 del SP7T sono misurati circa -42 dBm di potenza, mentre il segnale in uscita dalla posizione spenta del primo SPDT varia in un range tra -92 e -64 dBm.

Il problema legato alla possibile presenza di questi segnali è l'attivazione di canali che, per la misura sotto esame, dovrebbero essere spenti; immaginando di avere una risoluzione sufficiente sul power detector e sull'ADC utilizzati per convertire la potenza prima in tensione e poi in un numero digitale, il contributo di segnali spuri può modificare il reale valore della grandezza misurata e compromettere irrevocabilmente l'analisi effettuata sul campione.
Con un livello medio registrato di -40 dBm è però assicurato l'isolamento tra i diversi canali e la perturbazione associata a questi segnali è trascurabile rispetto alla potenza di interesse stimata intorno ai -6 dBm.

Per verificare il comportamento di tutti i componenti scelti e poter costruire il nuovo prototipo, occorre simulare anche il funzionamento dei rilevatori di potenza. Ricevendo come input un segnale in radiofrequenza, ciascun power detector è in grado di generare in uscita una tensione equivalente alla potenza in ingresso. Per implementare in AWR la funzione caratteristica del power detector scelto, è specificata nel programma l'equazione da risolvere per calcolare ciascuna conversione; in particolare, misurando la potenza in uscita dagli LNA, è possibile ottenere i valori delle tensioni DC che si avrebbero con il dispositivo reale.

Nella figura sotto riportata, sono messe a confronto rispettivamente la curva potenza-tensione presente sul datasheet di Analog Devices [39] e quella ottenuta interpolando in Matlab gli stessi valori tabulati.



(a) Datasheet

(b) Matlab

Figura 9.18: Curva caratteristica del power detector

Convertendo in tensioni valori di potenza misurati ad ogni uscita del circuito (a valle degli LNA), è possibile ottenere i parametri di diffusione calcolando la radice quadrata del rapporto delle tensioni.

Il valore equivalente alla potenza p_7 acquisita sull'ultimo canale (utilizzata come denominatore del rapporto) è di 3.707 V.

	A1	**A2**	**A3**	**A4**	**A5**	**A6**
A1		0.27921	0.23590	0.25917	0.25162	0.25370
A2	0.28443		0.34283	0.23976	0.23123	0.26631
A3	0.23619	0.33816		0.29491	0.25162	0.27810
A4	0.27974	0.23892	0.24626		0.37059	0.24925
A5	0.25210	0.23590	0.24625	0.36865		0.22894
A6	0.25344	0.24483	0.27815	0.25119	0.22859	

Tabella 9.13: Circuito AWR - Parametri di diffusione con rapporto tra tensioni

Dalla Tabella 9.13 si possono evidenziare due aspetti principali:

1. Nel confronto con i parametri sperimentali e quelli ottenuti senza l'applicazione dei power detector, questi coefficienti assumono valori nettamente più grandi e lo stesso rapporto tra le due grandezze non è costante. I rilevatori di potenza introducono delle *non-linearità* che rendono gli esempi così acquisiti incompatibili con le reti neurali precedentemente addestrate.

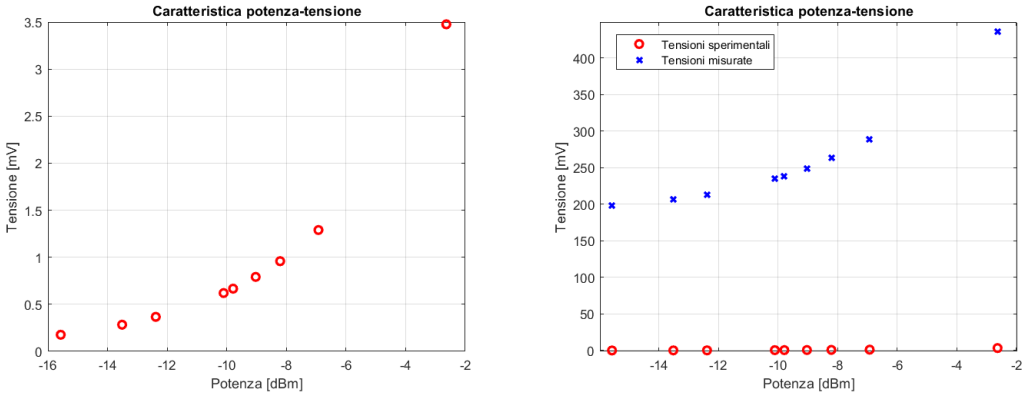
2. La distribuzione e l'andamento generale dei parametri rimane uguale a quello sperimentale: i coefficienti raccolti con il VNA che presentavano un valore alto riportano anche con queste simulazioni un valore alto, mentre i parametri più piccoli continuano ad essere quelli con il valore più basso.

Avendo a disposizione i parametri di diffusione sperimentali relativi ad un modello ideale, è possibile ricavare l'equazione che permetterebbe al power detector di effettuare una conversione puntuale ed esatta; interpolando i punti in Matlab, si ottiene l'espressione algebrica desiderata, che può quindi essere confrontata con quella reale a disposizione.

Mentre le due curve presentano singolarmente lo stesso andamento, la loro sovrapposizione mette in evidenza la non-linearità del componente: in Figura 9.19b si può vedere come i valori misurati (in blu) siano ben diversi da quelli ideali (in rosso), non solo nel valore assoluto, ma anche nel rapporto tra coppie di tensioni di pari potenza, il quale non è costante (vedi Tabella 9.14).

Rapporto	224.1	728.5	314.7	378.7	357.9	125.4	580.7	1123.0	274.3

Tabella 9.14: Rapporto tra tensione misurata e tensione sperimentale



(a) Tensione ideale (b) Confronto tensioni

Figura 9.19: Tensione delle potenze sperimentali e confronto con quelle misurate

Tenendo conto di questi risultati, i valori all'uscita dei power detectors non possono essere confrontati direttamente con i dati sperimentali a disposizione, ed il loro utilizzo richiederebbe l'acquisizione di nuovi dati per l'allenamento di nuove reti neurali.
Ciononostante, la simulazione dei componenti a monte dei rilevatori di potenza conferma il buon funzionamento del circuito ed i dati ottenuti con il rapporto finale sono confrontabili con i parametri di scattering originali.

9.3.3 Verifica con le reti neurali

Per concludere questa parte e verificare che i parametri acquisiti come rapporto di potenze possano garantire delle classificazioni corrette, gli stessi coefficienti sono utilizzati come nuovi esempi per le reti neurali già allenate.

Prima di effettuare le predizioni, sono misurati su AWR i coefficienti relativi ad alcuni casi con contaminante, in modo da avere una maggiore copertura sugli esempi a disposizione. Il modello ottenuto sul dataset di FT System con nastro a 30 cm/s (Sezione 7.1) è quindi utilizzato per predire la classe di tre barattoli.

La matrice di confusione in Figura 9.20 mostra il risultato del processo di predizione e conferma la buona riuscita della classificazione. Le metriche finali assumono tutte i loro valori massimi, anche se la simulazione è effettuata con un numero di esempi non sufficiente a generalizzare completamente il lavoro.

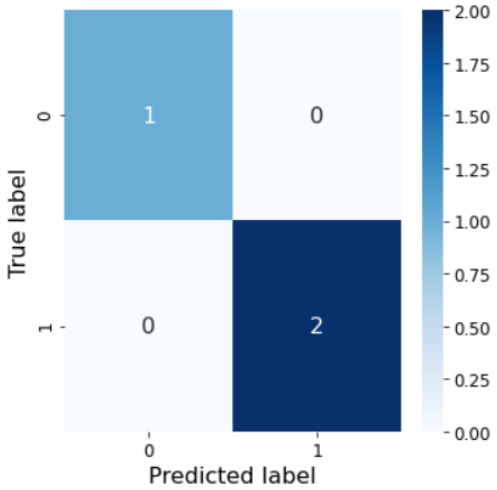


Figura 9.20: Matrice di confusione sugli esempi acquisiti con AWR

Come ultimo tentativo sono stati utilizzati, con la stessa rete neurale, anche i coefficienti acquisiti come rapporto di tensioni, dopo la conversione della potenza da parte dei power detectors.
Nonostante la non-linearità del componente, anche in questo caso la classificazione è corretta per tutti e tre gli esempi analizzati: ciascuna matrice di diffusione, modificata dall'applicazione dello stesso *scaler* del dataset originale, risulta compatibile con il modello allenato, anche se rimane preferibile addestrare una nuova rete neurale per caratterizzare specificatamente le nuove misure.

9.4 Non idealità del circuito AWR

Il circuito descritto ed utilizzato in AWR per simulare il comportamento dei componenti scelti per la costruzione del prototipo semplifica e rimuove una serie di non-idealità che potrebbero compromettere le reali prestazioni della macchina.

Sia per le possibilità offerte dal software, sia per la mancanza di alcune informazioni, non tutti gli aspetti caratteristici dei vari dispositivi sono tenuti in considerazione. Per cercare di coprire quanti più effetti non-ideali possibili, sono state effettuate nuove simulazioni dopo aver modificato propriamente il sistema.

Sono qui elencate alcune delle non-idealità considerate, con le modifiche apportate al circuito per poterne misurare il contributo.

9.4.1 Auto-diffusione delle antenne

La prima difficoltà riscontrata nella descrizione del modello è stata la gestione e la simulazione del barattolo: non esistendo nel CAD un elemento che ne potesse simulare la presenza, è stato scelto di utilizzare una soluzione diversa per implementarne almeno il contributo.

Come specificato nel paragrafo **??**, l'effetto legato all'interazione tra le antenne ed il barattolo è stato considerato tramite l'utilizzo dei parametri sperimentali acquisiti in laboratorio. Questi coefficienti, utilizzati per scalare la potenza del segnale trasmesso, tengono però in considerazione non solo l'attenuazione del prodotto alimentare, ma anche tutte le perturbazioni associate ai componenti elettronici della catena di trasmissione.

Per non rischiare di introdurre più volte gli stessi effetti nella simulazione, nel circuito AWR non è stato necessario inserire il modello delle antenne, ma semplicemente sono stati connessi tra loro i nodi di trasmissione e ricezione dopo l'applicazione dei fattori di scala.

Nonostante i risultati finali confermino il buon funzionamento del circuito, un effetto importante è così trascurato: per ogni antenna, insieme alla trasmissione del segnale dovrebbe essere considerata anche la generazione di un segnale riflesso. La presenza di questo nuovo segnale potrebbe perturbare il sistema e modificare il valore di potenza in arrivo ai rilevatori ed al convertitore analogico-digitale.
Un tentativo di implementarne tale effetto è stato fatto inserendo un carico finale dopo le antenne, in modo da rappresentare il parametro di auto-diffusione S_{ii}.

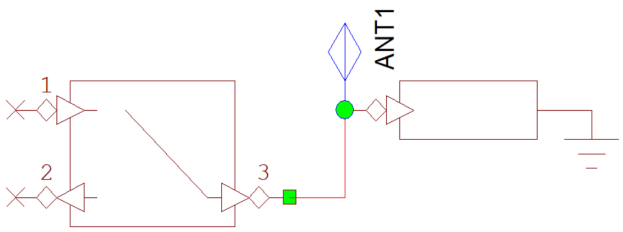


Figura 9.21: Circuito AWR - Elemento per simulare il carico dell'antenna

Il contributo di questo nuovo elemento non è però evidente nelle simulazioni, i cui risultati rimangono comunque uguali a quelli ottenuti in precedenza.
Rimane però valido il ragionamento fatto sui valori di isolamento dei vari interruttori: essendo i segnali riflessi deboli e con una potenza molto limitata, l'isolamento combinato dei SPDT e del SP7T dovrebbe essere sufficiente ad impedire una propagazione di questa non-idealità.

9.4.2 Mismatch delle impedenze e temperatura

Alcuni parametri che possono essere modificati per simulare il dispositivo in situazioni diverse sono le impedenze e la temperatura operativa dei componenti.

Il circuito AWR iniziale è progettato per lavorare con impedenze in ingresso ed uscita pari a 50 Ω: l'adeguamento delle impedenze è perfetto e questo garantisce l'annullamento del segnale riflesso e la massimizzazione della potenza trasmessa. É sufficiente modificare alcune di queste grandezze per vedere un cambiamento sui livelli di potenza raggiunti negli elementi delle varie catene di ricezione.

Dopo aver impostato alcune impedenze su 100 Ω, è stata ripetuta la simulazione del sistema e sono stati misurati i nuovi parametri di diffusione.
Il confronto tra la matrice di scattering ottenuta al termine dell'elaborazione ed i parametri sperimentali mostra, anche in questo caso, come il rapporto sia mantenuto costante e tutti i valori siano proporzionali con un fattore di scala di 5.2.

Lo stesso processo è ripetuto modificando la temperatura T del sistema: per alcuni componenti era infatti specificata nel datasheet una variazione dei parametri in funzione di T, e questa dipendenza permette di testare il circuito in una situazione più vicina alla realtà.
I parametri di diffusione calcolati dopo questa simulazione sono nuovamente proporzionali ai valori sperimentali, a conferma del fatto che le stesse perturbazioni sono applicate sia a numeratore che a denominatore.

9.4.3 Controllo dinamico degli componenti

Per semplicità, tutti i componenti del modello AWR (dagli interruttori agli amplificatori) sono stati scelti nella loro versione statica: il guadagno di VGA e LNA è stato impostato nella fase iniziale e mantenuto costante per tutta la simulazione; la posizione degli interruttori è modificata manualmente, così come la conversione delle potenze ed il calcolo dei parametri di diffusione sono effettuate al termine dell'elaborazione.

Nel prototipo reale è invece necessario gestire un controllo dinamico di tutti i componenti: il controllore scelto ed utilizzato per implementare la rete neurale si deve occupare anche di generare i segnali per organizzare e definire le diverse fasi di lavoro, dalla scelta dei gain allo switch degli interruttori.

Tra i problemi associati a delle operazioni dinamiche, sono da considerare:

- **mancata sincronizzazione** tra gli interruttori: se il canale di trasmissione non viene propriamente collegato prima che la potenza raggiunga l'antenna, il segnale ricevuto da tutti gli altri elementi sarà ridotto e non più corretto;
- **gain variabili** e non costanti: se il segnale di controllo generato dalla board non è costante, il guadagno offerto dagli amplificatori potrebbe variare e modificare il comportamento del circuito durante la simulazione stessa.
- la necessità di eseguire velocemente molti switch potrebbe incrementare il livello di **rumore nel sistema** e, se il tempo necessario alla stabilizzazione dei segnali non è rispettato, l'intera acquisizione non sarebbe più valida.

Tutti questi aspetti devono essere tenuti in considerazione nel caso di operazioni reali e potrebbero comportare la generazione di problemi tali da compromettere l'intera classificazione.

Parte V

Conclusione

Capitolo 10

Conclusione

Questa tesi è nata come un tentativo di riduzione dei costi di progettazione e sviluppo di un dispositivo a microonde per il rilevamento di contaminanti alimentari all'interno di barattoli di cioccolata impacchettati industrialmente.

Il lavoro si è inserito all'interno del progetto MIT-Food [15] e ne ha ereditato i risultati per semplificare il circuito e verificare la possibilità di ridurre la quantità di informazioni necessarie alla classificazione di ciascun barattolo.
Partendo dai risultati già raggiunti dal gruppo di ricerca [2][17], è stata verificata l'ipotesi secondo la quale *la rilevazione dei prodotti contaminati può essere effettuata analizzando solo il modulo dei coefficienti di diffusione del sistema formato dall'arco di antenne, aria e barattolo di cioccolata*: semplificando il processo di acquisizione è possibile evitare di dover misurare sia la parte reale che la parte immaginaria di ciascun segnale (come avveniva nel prototipo precedente) ed utilizzare quindi strumenti più semplici e specifici per raggiungere le stesse prestazioni.

I risultati raggiunti dopo l'addestramento delle nuove reti neurali con il modulo dei coefficienti di diffusione hanno dimostrato promettenti capacità di discernimento dei barattoli sicuri da quelli contaminati e, per poter generalizzare i risultati, sono stati analizzati dataset diversi acquisiti in condizioni differenti l'uno dall'altro. In particolare, sono state utilizzate misure acquisite con:

1. nastro di confezionamento alla velocità di 15 cm/s (sezione 5);
2. nastro a 30 cm/s e scudo metallico contro le interferenze esterne (sezione 6);
3. strumenti di FT System e velocità del nastro di 30 cm/s (sezione 7.1);
4. strumenti di FT System e velocità del nastro di 50 cm/s (sezione 7.2);

Tutte le analisi sono state ripetute lavorando prima alla singola frequenza di 10 GHz, poi con le caratteristiche appartenenti al range di frequenze 9-11 GHz.

Il nuovo prototipo progettato in questa tesi per effettuare le misurazioni del modulo dei parametri di scattering prevede l'utilizzo dei *power detectors*, grazie ai quali è possibile misurare i valori di potenza ed ottenere tutti i coefficienti richiesti. La tesi si è conclusa quindi con la simulazione del circuito: tutti gli esperimenti effettuati hanno garantito la corretta classificazione dei casi analizzati, ma non è stato possibile verificare completamente il dispositivo tenendo in considerazione tutte le non-idealità.

Tra i possibili sviluppi di questo progetto, i principali riguardano la realizzazione del nuovo prototipo. In particolare, partendo dall'assemblaggio del circuito proposto in questo lavoro con i componenti discreti suggeriti, si potrebbe procedere con:

1. la verifica del funzionamento del nuovo dispositivo e la programmazione del controllore per l'acquisizione automatica delle misure;
2. l'addestramento di nuove reti neurali per analizzare i coefficienti di diffusione ottenuti come rapporto di tensioni dopo la conversione dei power detectors;
3. l'allenamento di un nuovo modello di rete neurale per discriminare i contaminanti in prodotti alimentari diversi dall'olio di cartamo e dalla cioccolata utilizzati in questa tesi: i componenti scelti in fase di progetto coprono complessivamente una banda tra 2 GHz e 12 GHz e questo permette di penetrare ed analizzare cibi molto diversi tra loro, sia a base acquosa (meno densi) che oleosa (più densi);
4. integrazione del prototipo in un hardware progettato ad-hoc per questo scopo.

La conferma avuta con i risultati di questa tesi apre a molte possibilità sia in ambito di ricerca che di sviluppo: i risultati e le simulazioni di questa tesi fanno ben sperare sull'utilizzo del modulo dei coefficienti di diffusione come mezzo per rilevare la presenza di contaminanti all'interno di prodotti impacchettati industrialmente. Tuttavia occorrerà realizzare il nuovo prototipo e verificare le sue prestazioni sul campo per confermare ancora una volta l'efficacia della combinazione microonde e Machine Learning, con l'obiettivo di arrivare ad un prodotto industriale a basso costo che possa rendere i cibi che mangiamo sempre più sicuri.

Elenco delle figure

2.1 Esempio di dataset lineare e relativa funziona di fitting 18
2.2 Flow chart per l'algoritmo di training 19
2.3 Andamento dell'errore in funzione del parametro ω 20
2.4 Andamento dell'errore in funzione del parametro ω 21
2.5 Possibili errori sulla scelta del learning rate 22
2.6 Gradient descent con diversi valori di learning rate 22
2.7 Varianti di Gradient Descent in funzione del batch size 23
2.8 Esempi di sotto-adattamento e sovra-adattamento 24
2.9 Partizione del dataset in train - validation - test set 25
2.10 Algoritmo di allenamento e verifica del modello 26
2.11 Applicazione di un 5-fold cross-validation 27
2.12 Applicazione della tecnica di dropout sulla rete neurale 29
2.13 Curva sigmoide 31
2.14 Struttura generale di un albero decisionale 33
2.15 Support Vector Machine con ottima separazione degli iperpiani . . 34
2.16 Modello matematico di una ANN 37
2.17 Funzioni di attivazione: ReLU, tangente iperbolica e sigmoide . . 37
2.18 Multi-Layer Perceptron per una classificazione binaria 39

3.1 Panoramica del prototipo utilizzato per il microwave sensing . . . 45
3.2 Arco di sei antenne intorno al barattolo 46
3.3 Antenna PCB 46
3.4 Diagramma di radiazione dell'antenna 47
3.5 Esempi di contaminanti con dettaglio sulla dimensione 50

5.1 Principal Component Analysis su 2 e 3 dimensioni - primo dataset 65
5.2 Risultati della prima ricerca bayesiana - modello 1, primo dataset 68
5.3 Evoluzione delle metriche - modello 1, primo dataset 69
5.4 Matrici di confusione training e validation set - modello 1, dataset 1 70
5.5 Matrice di confusione sul test set - modello 1, primo dataset . . . 71
5.6 Matrice di confusione sull'extra set - modello 1, primo dataset . . 72

5.7 Risultati della seconda ricerca bayesiana - modello 2, primo dataset 74
5.8 Definizione del modello - primo dataset 74
5.9 Evoluzione delle metriche - modello 2, primo dataset 75
5.10 Matrici di confusione su training e validation set - modello 2 . . . 76
5.11 Matrice di confusione sul test set - modello 2, primo dataset . . . 77
5.12 Matrice di confusione e metriche su l'extra set - modello 2 78

6.1 Principal Component Analysis su 2 e 3 dimensioni - secondo dataset 80
6.2 Risultati della ricerca bayesiana - secondo dataset 83
6.3 Evoluzione delle metriche - secondo dataset 84
6.4 Matrice di confusione sul training set - secondo dataset 86
6.5 Matrice di confusione sul validation set - secondo dataset 87
6.6 Matrice di confusione sul test set - secondo dataset 88

7.1 PCA su 3 dimensioni - dataset FT System con nastro a 30 cm/s . 91
7.2 Risultati della ricerca bayesiana - dataset FT System 30 cm/s . . 93
7.3 Evoluzione delle metriche - dataset FT System 30 cm/s 94
7.4 Matrice di confusione sul training set - dataset FT System 30cm/s 95
7.5 Matrice di confusione sul validation set - dataset FT System 30 cm/s 96
7.6 Matrice di confusione sul test set - dataset FT System 30 cm/s . . 97
7.7 PCA su 2 e 3 dimensioni - dataset FT System 50 cm/s 98
7.8 Risultati della ricerca bayesiana - dataset FT System 50 cm/s . . 100
7.9 Evoluzione delle metriche - secondo dataset 101
7.10 Dettaglio sui parametri addestrabili della rete 102
7.11 Matrice di confusione sul training set - dataset FT System 50cm/s 103
7.12 Matrice di confusione sul validation set - dataset FT System 50 cm/s104
7.13 Matrice di confusione sul test set - dataset FT System 50 cm/s . . 105

8.1 Matrice di confusione sul test set - primo dataset multi-frequenza 108
8.2 Risultati ricerca bayesiana - dataset con scudo in multi-frequenza 110
8.3 Evoluzione delle metriche - dataset con scudo in multi-frequenza . 111
8.4 Matrici di confusione - dataset con scudo in multi-frequenza . . . 113
8.5 Matrici di confusione - FT System 30 cm/s in multi-frequenza . . 114
8.6 Risultati ricerca bayesiana - FT System 50 cm/s in multi-frequenza 116
8.7 Evoluzione delle metriche - FT System 50 cm/s in multi-frequenza 116
8.8 Confusion matrix validation set - FT System 50 cm/s multi-frequenza117
8.9 Confusion matrix test set - FT System 50 cm/s multi-frequenza . 118

9.1 Circuito per la generazione e la ricetrasmissione dei segnali 124
9.2 Scheda di valutazione EVAL-ADF5356 127
9.3 Scheda di valutazione EVAL-ADF5610 129
9.4 SigaTek SP11855 power divider 131

9.5 JFW 50S-1785 SP8T 134
9.6 RF-LAMBDA RFSP2TR0218G 136
9.7 Analog Devices EV1HMC1118LP3D 137
9.8 Mini-Circuits ZX60-123LN-S+ 140
9.9 Analog Devices ADL9005 141
9.10 Circuito e caratteristica potenza-tensione di un power detector . . 142
9.11 Linear Technology DC821A 143
9.12 Circuito e caratteristica potenza-tensione di un power detector . . 144
9.13 Circuito AWR - Componenti ideali 145
9.14 Circuito AWR - Matrice di switching per antenna 1 146
9.15 Circuito AWR - Potenze misurate sul circuito ideale 147
9.16 Circuito AWR - Potenze misurate sul circuito reale 149
9.17 Circuito AWR - Isolamento di SPDT e SP7T 150
9.18 Curva caratteristica del power detector 151
9.19 Tensione delle potenze sperimentali e confronto con quelle misurate 153
9.20 Matrice di confusione sugli esempi acquisiti con AWR 154
9.21 Circuito AWR - Elemento per simulare il carico dell'antenna . . . 155

Elenco delle tabelle

3.1 Tipi di contaminanti con dimensione e sigla usata nel dataset . . . 49

5.1 Descrizione degli esempi senza contaminante - primo dataset . . . 63
5.2 Primi cinque esempi senza contaminante - primo dataset 63
5.3 Descrizione degli esempi con contaminante - primo dataset 64
5.4 Numero di esempi con ciascun contaminante - primo dataset . . . 64
5.5 Primi cinque esempi con contaminante - primo dataset 64
5.6 Divisione e distribuzione dei tre set - primo dataset 65
5.7 Numero di esempi con ciascun contaminante - extra set 66

6.1 Descrizione degli esempi senza contaminante - secondo dataset . . 80
6.2 Numero di esempi con ciascun contaminante - secondo dataset . . 81
6.3 Divisione e distribuzione dei tre set - secondo dataset 81

7.1 Esempi con ciascun contaminante - dataset FT System 30 cm/s . 92
7.2 Divisione e distribuzione dei tre set - dataset FT System 30 cm/s 92
7.3 Esempi con ciascun contaminante - dataset FT System 50 cm/s . 99
7.4 Divisione e distribuzione dei tre set - dataset FT System 50 cm/s 99

9.1 Specifiche ADF5356 128
9.2 Specifiche ADF5610 129
9.3 Specifiche SP11855 132
9.4 Specifiche JFW 50S-1785 133
9.5 Specifiche P7T-0R8G18G-60-T-SFF-SMC 135
9.6 Specifiche RFSP2TR0218G 136
9.7 Specifiche RFSP2TR0218G 137
9.8 Specifiche ZX60-123LN-S+ 140
9.9 Specifiche ADL9005 - da 0.01 GHz a 14 GHz 141
9.10 Specifiche LTC5533 143
9.11 Circuito AWR - Parametri di diffusione con componenti ideali . . 148
9.12 Circuito AWR - Parametri di diffusione con componenti reali . . . 149
9.13 Circuito AWR - Parametri di diffusione con rapporto tra tensioni 152

9.14 Rapporto tra tensione misurata e tensione sperimentale 152

Bibliografia

[1] Friedrichs R Wright D. *Foreign Bodies - Techniques for Investigation and Identification*. 2017. URL: https://www.rssl.com/~/media/rssl/en/files/documents/white-paper/rssl-foreign-bodies.pdf?la=en.

[2] Luca Urbinati. «Detection of food contaminants with Microwave Sensing and Machine Learning». 2018-2019. URL: https://webthesis.biblio.polito.it/13241/.

[3] Atkare Prajwal et al. «Food Quality Detection and Monitoring System». In: *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*. 2020, pp. 1–4. DOI: 10.1109/SCEECS48394.2020.175.

[4] *Raggi X per la rilevazione di corpi estranei nei prodotti alimentari*. URL: http://www.inevo.it/articoli/2192/.

[5] D. Cozzolino. «Near infrared spectroscopy as a tool to monitor contaminants in soil, sediments and water—State of the art, advantages and pitfalls». In: *Trends in Environmental Analytical Chemistry* 9 (2016), pp. 1–7. ISSN: 2214-1588. DOI: https://doi.org/10.1016/j.teac.2015.10.001. URL: https://www.sciencedirect.com/science/article/pii/S221415881530009X.

[6] Urbinati Luca & Ricci Marco & Turvani Giovanna & Vasquez Jorge A. Tobon & Vipiana Francesca & Casu Mario R. «A Machine-Learning Based Microwave Sensing Approach to Food Contaminant Detection». In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020. DOI: 10.1109/ISCAS45731.2020.9181293. URL: https://ieeexplore.ieee.org/abstract/document/9181293.

[7] Kaiming He & Xiangyu Zhang & Shaoqing Ren & Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. URL: https://arxiv.org/pdf/1502.01852.pdf.

[8] Google. *Machine Learning Crash Course*. URL: https://developers.google.com/machine-learning/crash-course.

[9] Adam Geitgey. *Machine Learning is Fun!* URL: https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471.

[10] IBM - EdX. *Machine Learning with Python.* URL: https://www.edx.org/course/machine-learning-with-python-a-practical-introduct.

[11] Imad Dabbura. *Gradient Descent Algorithm and Its Variants.* URL: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3.

[12] Alessio Morselli. *Deep Learning - Regolarizzazione.* URL: https://alessiomorselli.github.io/DeepLearning/web/pages/teoria/regularization.html.

[13] Charu C. Aggarwal. *Data Classification, Algorithms and Applications.* Chapman & Hall/CRC, 2015.

[14] CorporateFinanceInstitute. *Decision Tree.* URL: https://corporatefinanceinstitute.com/resources/knowledge/other/decision-tree/.

[15] Politecnico di Torino - Vipiana Francesca. *MIT-Food, Microwave Imaging Technology for Food Contamination Monitoring.* URL: https://www.researchers.polito.it/success_stories/progetti_metti_in_rete_la_tua_idea_di_ricerca/microonde_per_monitorare_gli_alimenti.

[16] *Analizzatore di reti.* URL: https://it.wikipedia.org/wiki/Analizzatore_di_reti.

[17] Bernardita Stitic. «A multiclass neural network model for contaminant detection in hazelnut-cocoa spread jars». 2019-2020. URL: https://webthesis.biblio.polito.it/14393/.

[18] Vasquez J. Tobon & Rivero J. & Scapaticci R. & Farina L. & Crocco L. & Vipiana F. «Monitoring of Food Contamination via Microwave Imaging». In: *2019 International Applied Computational Electromagnetics Society Symposium (ACES).* 2019, pp. 1–2.

[19] *Numpy.* URL: https://numpy.org/.

[20] *Pandas.* URL: https://pandas.pydata.org/.

[21] *Scikit-Learn.* URL: https://scikit-learn.org/stable/.

[22] *TensorFlow.* URL: https://www.tensorflow.org/.

[23] *Keras.* URL: https://keras.io/.

[24] *Anaconda.* URL: https://www.anaconda.com/.

[25] *Jupyter.* URL: https://jupyter.org/.

[26] Marco Ricci et al. «Machine-Learning-Based Microwave Sensing: A Case Study for the Food Industry». In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 11.3 (2021), pp. 503–514. DOI: 10.1109/JETCAS.2021.3097699.

[27] David Herres. 2020. URL: https://www.rfwireless-world.com/Terminology/SNA-Scalar-Network-Analyzer-vs-VNA-Vector-Network-Analyzer.html.

[28] *Spectrum Analyzer Tracking Generator*. URL: https://www.electronics-notes.com/articles/test-methods/spectrum-analyzer/spectrum-analyser-tracking-generator.php.

[29] *Designing a microwave amplifier*. URL: https://www.robkalmeijer.nl/techniek/electronica/radiotechniek/hambladen/hr/1987/09/page30/index.html.

[30] *ADF5356*. datasheet. Analog Devices, 2017. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ADF5356.pdf.

[31] *ADF5610*. datasheet. Analog Devices, 2019. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ADF5610.pdf.

[32] *SP11855*. datasheet. SigaTek. URL: https://www.sigatek.com/rf-power-divider-combiner-splitter-stripline-wilkinson-sma-8way/SP11855.pdf.

[33] *50S-1785*. datasheet. JFW. URL: https://www.jfwindustries.com/pdf/50S-1785.pdf.

[34] *P7T-0R8G18G-60-T-SFF-SMC*. datasheet. PMI. URL: https://d2f6h2rm95zg9t.cloudfront.net/81644761/Typical_Characteristics_on_P7T_0R8G18G_60_T_SFF_SMC_82275669.pdf.

[35] *RFSP2TR0218G*. datasheet. RF-LAMBDA, 2021. URL: https://www.rflambda.com/pdf/switchers/RFSP2TR0218G.pdf.

[36] *HMC1118*. datasheet. Analog Devices. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/HMC1118.pdf.

[37] *ZX60-123LN-S+*. datasheet. Mini-Circuits. URL: https://www.minicircuits.com/pdfs/ZX60-123LN-S+.pdf.

[38] *ADL9005*. datasheet. Analog Devices. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/adl9005.pdf.

[39] *LTC5533*. datasheet. Linear Technology. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/5533f.pdf.

[40] *8471E*. datasheet. Keysight. URL: https://www.keysight.com/it/en/assets/7018-06719/data-sheets/5952-0802.pdf.

[41] *AWR Design Environment*. URL: https://www.awr.com/awr-software/products/awr-design-environment.

[42] Mario R. Casu et al. «A COTS-Based Microwave Imaging System for Breast-Cancer Detection». In: *IEEE Transactions on Biomedical Circuits and Systems* 11.4 (2017), pp. 804–814. DOI: 10.1109/TBCAS.2017.2703588.

[43] Vinod K. Jacob et al. «RF System Design for Passive Detection of Humans». In: *2018 IEEE MTT-S International Microwave and RF Conference (IMaRC)*. 2018, pp. 1–4. DOI: 10.1109/IMaRC.2018.8877273.

[44] Noriaki Tawa e Tomoya Kaneko. «A 950MHz RF 20MHz bandwidth direct RF sampling bit streamer receiver based on an FPGA». In: *2017 IEEE MTT-S International Microwave Symposium (IMS)*. 2017, pp. 594–597. DOI: 10.1109/MWSYM.2017.8058637.