**ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA**

**SCUOLA DI INGEGNERIA E ARCHITETTURA**

*Dipartimento di Ingegneria dell'Energia Elettrica e dell'informazione "Guglielmo Marconi"*

*Laurea Magistrale in Ingegneria Elettronica*

**TESI DI LAUREA**

in

Elaborazione statistica dei Segnali

# Elaborazione Audio dei segnali con Reti Neurali Profonde per la Rilevazione di Situazioni di Pericolo

CANDIDATO
Perugini Enrico

RELATORE
Chiar.mo Prof. Rovatti Riccardo

CORRELATORI
Prof. Mangia Mauro
Prof. Guidotti Alessandro
Dott. Ing. Marchioni Alex

Anno Accademico 2019/2020 – Sessione III

*Alle uniche persone che hanno sempre creduto in me*
*e senza le quali non sarei qui*

# Abstract

Nei sistemi di sorveglianza moderni, soluzioni composte dall'unione di telecamere a circuito chiuso e tecniche di intelligenza artificiale, rappresentano lo strumento principale per fronteggiare minacce e pericoli in diversi ambienti: ambienti pubblici, abitazioni private, uffici, strutture critiche come ospedali o scuole.

Questi sistemi vengono equipaggiati da robuste tecniche di computer vision, le quali permettono di riconoscere e rilevare oggetti e persone, attraverso sequenze di immagini in maniera automatica. L'obiettivo è predire l'azione degli elementi osservati in un determinato scenario per aumentare l'efficienza globale di un sistema di sorveglianza. Tuttavia, l'analisi delle immagini può subire importanti cali di prestazioni in diverse circostanze, dovuti alla natura dei sensori video e dalle limitazioni che essi introducono.

Nel progetto di tesi presentato, si discute lo sviluppo di un sistema di riconoscimento di situazioni di pericolo i cui dati elaborati sono acquisiti da sensori audio. Negli ultimi anni, la sorveglianza audio ha riscosso un grande interesse grazie alla flessibilità di utilizzo, sia per la diversità delle situazioni in cui può essere impiegata, sia per la possibilità di essere combinata con la controparte video in sistemi ibridi.

Il sistema proposto è costituito da una rete neurale convoluzionale, la cui architettura si ispira fortemente alla VGG19. Al suo ingresso vengono fornite immagini costruite a partire da porzioni di stream audio e trasformate in rappresentazioni tempo-frequenza quali: spettrogramma, spettrogramma in scala Mel e gammatonogramma. L'obiettivo è stato quello di costruire un modello di classificazione di eventi audio di pericolo, per i quali si sono considerati suoni come: vetri che si infrangono, colpi di pistola e urla. Successivamente si è condotto un confronto sia tra le performance indotte dall'utilizzo delle tre rappresentazioni, sia tra la rete neurale e una tecnica di classificazione standard quale l'SVM.

# SOMMARIO


1 INTRODUZIONE ..... 1
2 STRUMENTI AUTOMATICI PER LA SORVEGLIANZA ..... 3
2.1 Definizione formale del problema ..... 3
2.2 Una panoramica sulla Classificazione Audio ..... 7
2.3 Evento Audio ..... 9
2.4 Tecniche allo stato dell'arte ..... 10
2.5 Flusso di elaborazione in un sistema di sorveglianza ..... 13
2.6 Struttura di un sistema di sorveglianza ..... 15
3 TEORIA DEGLI STRUMENTI ..... 19
3.1 Neural Network ..... 19
3.2 Support Vector Machine ..... 30
3.3 Spettrogramma ..... 37
3.4 Spettrogramma in scala Mel ..... 43
3.5 Gammatogramma ..... 44
3.6 Coefficienti del Cepstrum in scala Mel ..... 50
3.7 Matrice di Confusione ..... 51
4 PAPER DI RIFERIMENTO ..... 58
4.1 SoReNet ..... 58
4.2 MC SVM ..... 61
5 IMPLEMENTAZIONE DEI MODELLI ..... 63
5.1 Descrizione del progetto ..... 63
5.2 Dataset ..... 63
5.3 Costruzione delle feature ..... 65
5.3.1 Neural Network ..... 66
5.3.2 SVM ..... 71
5.4 Training della Rete Neurale ..... 72
5.4.1 Setting dei parametri e training preliminari ..... 72
5.4.2 Confronto tra le strategie di allenamento ..... 81
5.4.3 Confronto tra le rappresentazioni tempo-frequenza ..... 93

5.5 Training dell'SVM ..... 116
6 DETECTOR ..... 122
7 CONCLUSIONI ..... 133
8 BIBLIOGRAFIA ..... 136

# 1 INTRODUZIONE

La sicurezza, sia interna che pubblica, è una preoccupazione sostanziale per i governi di tutto il mondo, impegnati nella protezione dei cittadini e delle infrastrutture critiche. La tecnologia dell'informazione gioca un ruolo significativo in queste iniziative, può aiutare a ridurre il pericolo e fornisce risposte efficaci ai disastri di origine naturale o umana [1].

Il mercato globale della sorveglianza dovrebbe crescere da 45,5 miliardi di dollari del 2020 a 74,6 miliardi di dollari entro il 2025, con un tasso composto di crescita annuale (CAGR) del 10,4%. L'aumento dei finanziamenti del governo per lo sviluppo di città intelligenti e l'impiego di soluzioni smart di sorveglianza per le città, i progressi tecnologici nel settore Big Data, IoT, servizi cloud e le tendenze prevalenti di intelligenza artificiale forniranno grandi opportunità d'investimento nel settore della sicurezza [2].

Alcuni motivi di carattere generale per investire in questo settore sono:

- Arginare i sistemi di sicurezza che agiscono localmente e non cooperano in modo efficiente.
- Proteggere beni di valore estremamente elevato con sistemi tecnologicamente all'avanguardia.
- Rilevare e valutare le minacce necessita una intensa concentrazione umana.

Il valore della sicurezza non si misura esclusivamente mediante previsioni d'investimento o in termini di crescita tecnologica, ma anche tramite la necessità di affrontare un nemico di calibro globale. Il 2020 è stato un anno complicato per l'intera popolazione mondiale, e con l'arrivo della pandemia causata dalla SARS-CoV-2, molti governi hanno dovuto prendere aspre decisioni al fine di salvaguardare la salute dei propri cittadini. Il lockdown totale, in diverse circostanze, ha consentito di limitare considerevolmente i

contagi da Covid-19. Tuttavia, la quarantena forzata non è stato l'unico strumento con cui affrontare il virus, il monitoraggio della diffusione dello stesso ha giocato un ruolo fondamentale nel consentire anche la parziale riapertura delle attività professionali e sociali. In aiuto a queste esigenze molte nazioni hanno impiegato tecnologie di sorveglianza al fine di collezionare dati per il tracciamento di persone sintomatiche e di coloro che sono entrati in contatto con esse.
Durante l'*era Covid*, la sorveglianza digitale sembra essere diventata uno strumento fondamentale per il contenimento della pandemia [3].

I sistemi di sorveglianza hanno quindi come obiettivo primario i terroristi e i violatori delle restrizioni e sono custodite con riservatezza per evitare che far conoscere tali tecnologie possa renderle più vulnerabili. Le aziende di tutto il mondo investono nella ricerca e nello sviluppo di sistemi intelligenti di sorveglianza, software di data mining, sistemi biometrici e tecnologia di geolocalizzazione su Internet. Il crescente utilizzo di queste soluzioni ha suscitato negli anni un imponente interesse pubblico [4].

# 2 STRUMENTI AUTOMATICI PER LA SORVEGLIANZA

## 2.1 Definizione formale del problema

Oggigiorno, la garanzia di un determinato livello di sicurezza è fondamentale per tutelare l'incolumità civile. La sorveglianza è una pratica comune in diverse circostanze, sia in contesti pubblici che privati, come ad esempio centri commerciali, uffici, banche, strade, piuttosto che condomini, smart home o aree residenziali. Il riconoscimento di situazioni di pericolo può includere diverse minacce: rapine, furti, danneggiamento di strutture, salute di persone o salvaguardia di intere aree metropolitane [5].
Parallelamente allo sviluppo di aree urbane, strutture industriali e ambienti critici, la necessità di un sistema automatico di sorveglianza è cresciuta analogamente.

Un generico sistema di sorveglianza è costituito da un insieme di sensori i quali collezionano gli avvenimenti dell'ambiente circostante, con lo scopo di monitorare contemporaneamente i luoghi di interesse.
La prima generazione di sistemi di sorveglianza è costituita interamente da una struttura analogica. Telecamere a circuito chiuso catturano le immagini dell'ambiente da monitorare. Questa metodologia richiede la presenza di uno o più operatori addetti al ruolo di sicurezza, che controllino costantemente le immagini in remoto. Le prestazioni di questi sistemi dipendono fortemente dalla capacità e dall'attenzione umana, la quale può facilmente calare e provocare periodi più o meno lunghi di distrazione e interruzione dell'attività di controllo. Anche il numero di telecamere può incidere sul livello di attenzione e lucidità degli operatori. Tali circostanze possono incidere fortemente sul tasso di riconoscimento di un pericolo, il quale può passare inosservato.
Inoltre, si devono considerare tutte le problematiche legate al mondo analogico video, quali grandi volumi di salvataggio dei dati e necessità di elevata larghezza di banda dei dispositivi.

Ciò nonostante, l'evoluzione tecnologica, l'avvento del digitale e l'esplosione dell'intelligenza artificiale hanno provocato una forte spinta nella ricerca di sistemi intelligenti e automatizzati in grado di assistere gli operatori nel riconoscere potenziali pericoli in tempo reale.
Questa nuova generazione è nata grazie all'unione dei sistemi a circuito chiuso e alla tecnologia del computer vision, la quale permette di tracciare, riconoscere e rilevare oggetti attraverso sequenze di immagini in maniera del tutto automatica. L'obiettivo primario consiste nel capire e predire le azioni degli elementi osservati in un determinato scenario, per rimpiazzare l'attività passiva di osservazione delle immagini ed aumentare l'efficienza globale di un sistema di sorveglianza.
Conseguentemente, l'impiego di una rete di telecamere interconnesse per formare un sistema distribuito di monitoraggio è stato un passaggio quasi prevedibile. L'obiettivo principale consiste nel fornire una buona comprensione delle scene, informazioni in tempo reale e l'impiego di componenti a basso costo.
Sfruttando dunque l'impiego di dispositivi embedded, vantaggi quali scalabilità, possibilità di poter lavorare con un volume maggiore di dati, l'abbattimento globale dei costi, una maggior robustezza del sistema, copertura di un'area maggiore e comunicazioni wireless, sono stati raggiunti.
L'insieme di questi aspetti comporta considerare una nuova ed ulteriore generazione di sistema di sorveglianza, le cui principali applicazioni si presentano nel settore della pubblica sicurezza, e sono rese necessarie dalla rapida crescita di aree metropolitane e dalla necessità di offrire una migliore salvaguardia alle persone.
D'altro canto, complessità di una integrazione e comunicazione efficiente di informazioni sono problematiche ancora presenti. [4]

Tuttavia, un sistema di sorveglianza basato interamente su sensori video può incorrere in cali di performance importanti, che dipendono da diversi fattori, imposti quasi totalmente dall'ambiente circostante, ad esempio: cambi di luce

improvvisi, riflessioni, ombre, condizioni di meteo avverse e per contro anche la sensibilità delle videocamere.
Per di più, condizioni di buio, necessitano differenti approcci, in quanto la scarsa illuminazione o il bagliore delle luci non consente l'utilizzo di videocamere standard. Per ovviare a questo vengono impiegate diverse tipologie di sensori, come sensori ad infrarossi o termici. Sebbene si ottengano buoni risultati, l'elevata sensibilità a fonti di temperatura e la difficoltà di separazione dal background rende spesso inefficaci il loro utilizzo.

Una strada alternativa da percorrere per far fronte alle debolezze dei sistemi video è di sfruttare le informazioni contenute nella loro controparte audio. Utilizzare sensori audio concede l'opportunità di costruire un sistema con caratteristiche interessanti:

- Gli stream audio sono generalmente meno onerosi sia in termini di volume di dati, sia in risorse di calcolo.
- Presentano un costo e un ingombro minore.
- I sensori audio possono essere omnidirezionali senza dover modificare la struttura dei dati e senza un'occupazione maggiore di questi.
- I microfoni consentono di acquisire eventi audio anche se sono presenti ostacoli sul cammino diretto sorgente – sensore.
- Illuminazione e temperatura non inficiano le performance.
- Alcuni eventi audio spesso non hanno la controparte video.
- Da un lato psicologico, il monitoraggio audio risulta decisamente meno invasivo rispetto a quello video.

Queste qualità mettono in risalto le potenzialità di un monitoraggio audio, tuttavia, ogni tipo di approccio ad una sfida presenta pregi e difetti.

Le problematiche di maggior impatto di una soluzione audio sono le seguenti:

- Il background presenta un alto grado di variabilità dovuto ad una caratteristica intrinseca di tempo varianza della maggior parte dei suoni, differentemente da quello video che presenta caratteristiche più statiche e lentamente variabili. Ciò si traduce in una più ardua pulizia delle tracce audio.

- Il segnale audio è decisamente più complesso da analizzare rispetto a quello video, in quanto un singolo microfono cattura a prescindere multiple sorgenti audio a cui si sommano artefatti acustici causati dall'ambiente circostante: echi, riverberi, rifrazioni e riflessioni.

- Il Rapporto Segnale Rumore (SNR) è tipicamente minore del caso video e dipende fortemente dalla distanza a cui un evento audio avviene.

[6]

Senza alcun dubbio, analizzare tracce audio comporta sfide piuttosto impegnative, in quanto: informazioni discriminative spesso risiedono in porzioni dello spettro acustico a bassa frequenza, scene audio non presentano specifiche strutture da poter riconoscere come ad esempio avviene per i fonemi nel *sound recognition*, al contrario di segnali audio musicali gli eventi audio non mostrano alcun tipo di sequenza o struttura melodica da poterne modellare le proprietà.
Alcune delle principali difficoltà legati al mondo dell'analisi audio possono essere le seguenti:

- Riconoscere eventi da un singolo ambiente, un ufficio, un'area residenziale o una strada affollata implica dover sapere riconoscere una moltitudine di suoni diversi

- Le scene audio possono essere decisamente caotiche se si pensa a luoghi pubblici o ambienti trafficati

- Non è possibile costruire un dizionario di suoni base
- Mancanza di dataset consistenti

Detto ciò, si intuisce che probabilmente la strada migliore è quella di raccogliere il meglio dei due metodi e costruire una soluzione ibrida che lavori sinergicamente, in modo da ottenere prestazioni e versatilità superiori. [7]

## 2.2 Una panoramica sulla Classificazione Audio

L'obiettivo della classificazione è presto definito: costruire una metodologia in grado di assegnare autonomamente elementi a classi di appartenenza predefinite.
È una delle applicazioni più note del mondo del machine learning e può essere affrontato con diverse tecniche, le quali presentano un approccio simile. Convenzionalmente, si procede con una fase di preelaborazione in cui vengono scelte delle caratteristiche, in gergo *feature*, le quali sono estratte direttamente dagli elementi. Queste hanno il compito di enfatizzare le peculiarità degli elementi da classificare rappresentandoli in un determinato spazio, al fine di massimizzare le prestazioni di classificazione. In seguito a ciò si deve scegliere un metodo che possa nel migliore dei modi "capire" come gli elementi tendono a disporsi nel suddetto spazio con lo scopo di saper prevedere la classe di appartenenza di nuovi elementi.

La più grande specializzazione della classificazione audio riguarda l'ambito del riconoscimento vocale, in inglese *speech recognition*. Esso è presente in una moltitudine di contesti, alcuni dei quali sono partecipi della nostra vita quotidiana, come l'uso degli assistenti vocali di cui i nostri smartphone o computer sono dotati. Altre comuni applicazioni possono essere trovate in campi quali l'apprendimento delle lingue, IoT, automotive, militare e sanitario [8].

Tuttavia, recentemente grande attenzione è stata dedicata all'analisi di suoni non vocali, poiché diversi settori hanno iniziato a considerare questa come una risorsa interessante da sfruttare. Inoltre, diverse sfide internazionali sono state indette con lo scopo di attirare attenzione in questa direzione.
I contesti in cui è possibile imbattercisi possono essere i seguenti:

- **Ambiente assistito:** i sistemi di monitoraggio acustico sono inseriti in maniera non intrusiva all'interno di abitazioni con lo scopo di anticipare i bisogni delle persone e mantenere la loro sicurezza e comfort. Più frequentemente impiegati per assistere persone anziane e riconoscere situazioni di pericolo.

- **Analisi multimediale:** il rilevamento di eventi audio e la loro classificazione svolge un ruolo importante come complemento di un sistema per il riconoscimento di eventi video. L'integrazione dei due mondi ha permesso di raggiungere un'accuratezza superiore rispetto alla sola fonte video.

- **Monitoraggio ambientale:** vi è un incremento sostanziale nell'adoperare registrazioni audio per monitorare la densità di popolazione di specie animali, i loro flussi migratori, la salute generale di un ecosistema oppure, aiutare le persone a riconoscere gli animali attraverso i loro richiami

- **Classificazione musicale:** le tecniche di classificazione sono largamente utilizzate per classificare brani musicali e successivamente creare categorie secondo il genere, generare playlist sulle piattaforme musicali, raccomandare musica agli ascoltatori.

[9]

Si fornisce ora una semplice tassonomia, la quale suddivide gli algoritmi di classificazione in tre casistiche:

1. Generativa

2. Discriminativa

3. Ibrida

Nel primo caso troviamo l'insieme dei metodi per i quali ogni classe possiede il proprio classificatore. Solitamente questa tipologia viene definita in un contesto Bayesiano, dove il punteggio assegnato dal classificatore è una probabilità a posteriori. Nel caso di un classificatore multiclasse, il classificatore con più alta probabilità assegnerà la classe di appartenenza all'elemento da classificare. In questa sezione troviamo *Hidden Markov Model* (HMM) e *Gaussian Mixture Model* (GMM) quali tecniche principali.
Per quanto riguarda il caso discriminativo, esso propone modelli i quali cercano di costruire le migliori superfici di separazione nello spazio delle features, suddividendolo in sottospazi. *Support Vector Machine* (SVM) e le *Reti Neurali Artificiali* (ANN) sono i modelli più comuni.
Come spesso accade, versioni che accomunano il meglio di più versioni sono costruite, come ad esempio coppie GMM-SVM, in cui le combinazioni dei singoli score sono create e successivamente usate per scegliere la classe [6].

## 2.3 Evento Audio

Un *suono* può essere definito come il propagarsi di un moto ondoso in un mezzo di trasmissione quale l'aria o altri mezzi elastici. Le vibrazioni vengono prodotte da un evento scatenante il quale è la causa dell'espandersi del suono in una o più direzioni. L'origine del suono produce delle variazioni locali della pressione dovute all'oscillazione di particelle del mezzo, il cui movimento viene propagato alle particelle adiacenti e così via. Un suono può anche essere interpretato come un'eccitazione del meccanismo uditivo il quale porta alla percezione del suono stesso. Il lato fisico delle definizioni appena fornite, è un punto di vista più appropriato per chi cerca di interpretare questi elementi come unità distintive di applicazioni musicali o audio [10] [11].
Ciò nonostante, non esiste una definizione univoca di *evento audio*, ma al contempo possono essere provviste alcune interpretazioni.

Un evento audio può essere visto come una composizione iterativa di eventi minori o *unità atomiche* al fine di costruire strutture più complesse, e da queste un alfabeto. Da questo lato della questione, affrontare il problema di riconoscere eventi audio si riduce al riconoscimento di sequenze di unità atomiche [12]. Un evento audio può essere pensato inoltre come unità di suoni concreti e temporalmente limitati, come lo squillare di un telefono, tossire o un colpo di pistola [9]. Ulteriormente, un evento audio può essere considerato come tutto ciò che non è parlato, ovvero l'insieme di quei suoni che non costituiscono il mezzo espressivo di un dialogo.

## 2.4 Tecniche allo stato dell'arte

In letteratura, è possibile trovare diverse metodologie per affrontare il problema del riconoscimento di eventi audio. L'approccio classico è quello di ridurre la dimensione dell'ingresso estraendo delle feature, per poi allenare un modello e attuare la classificazione.

Nel lavoro svolto in [13], gli autori hanno sfruttato un classificatore basato su GMM. Il suddetto modello permette di descrivere la presenza di sottopopolazioni all'interno di un insieme generale, e viene adoperato quando non è conosciuto a quale sottopopolazione un dato di allenamento appartiene, si parla quindi di allenamento non supervisionato. La distribuzione di ogni sottopopolazione viene identificata da una funzione Gaussiana con le seguenti caratteristiche: valore medio, matrice di covarianza e un parametro il quale distingue il peso di ogni campana, conosciuto come *mixture weight*. La somma dei mixture weight deve ritornare 1.
Sono state considerate poi una serie di features quali: Mel Frequency Cepstral Coefficient (MFCC), Linear Predictive Coding (LPC); caratteristiche spettrali quali: centroid, spread, skewness e flatness; caratteristiche di timbro come: Zero Crossing Rate (ZCR), spectral roll-off, brightness e roughness.
Finestre di Hamming di $25ms$ sono considerate con overlap del $50\%$.
L'obiettivo principale è stato quello di classificare direttamente gli eventi piuttosto che riconoscere la presenza di un evento e poi classificarlo. Una

classe aggiuntiva di background è stata considerata per valutare se una finestra contiene un evento da classificare o background. Lavori simili sono stati sviluppati in [14] e [15].

Un diverso tipo di approccio è conosciuto come B*ag of Words* (BoW). È una tecnica comunemente utilizzata per classificare documenti di testo, in cui la frequenza dell'occorrenza di ogni parola è usata come feature per allenare il modello. Questa tecnica si adatta particolarmente a simili scopi perché tecniche classiche di machine learning preferiscono ingressi e uscite di dimensioni fissate.
Proprio per la peculiarità appena descritta, in [16] il Bag of Words è sfruttato per interfacciarsi con eventi audio di durata diversa e con presenza significativa di rumore. Il metodo proposto consiste nel suddividere lo stream audio in piccole finestre, dell'ordine di qualche millisecondo, dalle quali estrarre feature a basso livello, in modo analogo alle parole di un documento. Queste poi, vengono utilizzate per costruire features ad alto livello, i cui elementi sono le occorrenze delle feature a basso livello. Per ogni finestra, l'istogramma delle occorrenze è costruito e utilizzato come ingresso di un classificatore SVM. Un lavoro analogo è proposto in [17].

Diversamente in [18] e [19], tecniche di elaborazione delle immagini sono state applicate per rappresentare dei suoni nel dominio spettrale per il rilevamento di eventi acustici. Vengono utilizzate tecniche di pseudo-colorizzazione per partizionare lo spettrogramma in rappresentazioni separate prima di estrarre le caratteristiche. Successivamente metodi quali kNN e SVM sono stati scelti come classificatori.

Sebbene siano vantaggiose per ridurre la dimensionalità del problema, le feature causano una inevitabile perdita di informazione.
Recentemente, le *Reti Neurali Profonde*, o *Deep Neural Network* (DNN), hanno dimostrato la grande abilità di integrare l'estrazione delle feature durante la fase di allenamento del modello, surclassando le tecniche dello stato dell'arte

in svariati problemi di classificazione. Ciò si traduce in una preelaborazione minima dei dati grezzi. Oltretutto, nonostante un gran costo computazionale sia richiesto in fase di allenamento, l'operazione di classificazione risulta computazionalmente veloce.

Nel lavoro proposto in [20], è stato realizzato un confronto sia sul piano del classificatore di eventi audio, in particolare tra SVM e DNN, sia sul piano delle feature sfruttate. La scelta di quest'ultime è stata rivolta verso caratteristiche uditive biologicamente ispirate e immagini basate su una rappresentazione tempo-frequenza quale lo spettrogramma. I risultati mostrano ragionevoli vantaggi sfruttando le reti neurali in task classici di classificazione.

In [21], è stata utilizzata una rete conosciuta come *Deep Beliefe Network* (DBN), la cui differenza distintiva dagli altri modelli è la non presenza di una relazione tra ogni neurone, ma una relazione globale tra strato e strato. Feature basate su immagini tempo-frequenza sono confrontate con descrittori più tradizionali come MFCC e LPC. Mentre in [22], una combinazione di MFCC e DBN è stata sfruttata specificamente per rilevare suoni di urla per l'audio sorveglianza.

Nel ramo dell'elaborazione audio reti neurali di tipo ricorrente, *Recurrent Neural Network* (RNN), hanno trovato largo impiego in task di riconoscimento vocale e affini. Il nome di questa rete deriva dal fatto che i neuroni ricevono in input le uscite dei neuroni dello strato successivo. Purtroppo, questa architettura non è in grado di apprendere dipendenze a lungo termine, poiché soffre di instabilità numerica. Sono state proposte alcune varianti per risolvere le debolezze della rete, come la *Long Short-Term Memory* RNN (LSTM RNN) o la *Bi-directional* LSTM RNN (BLSTM RNN), nelle quali sono inseriti parametri addizionali in funzione dei quali la rete è in grado di decidere le informazioni da tenere o dimenticare. Un esempio di BLSTM RNN è riportato in [23], dove gli autori hanno svolto un task di riconoscimento di eventi audio in dieci contesti differenti di vita reale.

Gli autori di [24], hanno proposto un classificatore audio basato su di una rete neurale di tipo convoluzionale, soluzione proposta spesso in letteratura negli ultimi anni ed ereditata dal computer vision, che ha dimostrato di essere una

delle architetture migliori in task quali classificazione di immagini. Un banco di filtri su scala Mel è stato costruito per estrarre feature per la rete, la cui uscita identifica se un determinato ingresso appartiene ad una di trenta classi di suoni differenti.
Più recentemente, nell'attività discussa in [25], la questione della sorveglianza audio è stata studiata più in dettaglio, in particolare nell'ambito della sicurezza stradale. L'obiettivo è stato quello di riconoscere due suoni in particolare: slittamento degli pneumatici e incidente d'auto. Nel dettaglio, una finestra mobile di Hamming di lunghezza pari a 3 secondi e con overlap del 66% è stata utilizzata per frammentare lo stream audio. Dopodiché, lo spettrogramma di ogni istanza è stato impiegato per definire la feature da dare in pasto alla rete. I risultati dimostrano che il metodo possiede i requisiti necessari per poter essere implementato su sistemi embedded distribuiti di sorveglianza stradale, grazie al basso costo computazionale che il modello richiede.

## 2.5 Flusso di elaborazione in un sistema di sorveglianza

Per descrivere un sistema di sorveglianza video, si sceglie solitamente una rappresentazione *bottom-up*, evidenziando quattro passi principali di elaborazione delle informazioni:

1. **Background subtraction:** è il primo passaggio nella catena di manipolazione dei dati e consiste nel separare gli oggetti in movimento dalla scena in cui si trovano (background).

2. **Object detection:** è lo step nel quale una volta evidenziati gli oggetti, essi devono essere classificati in una delle classi definite a priori: veicoli, animali, passanti e così via.

3. **Object tracking:** in questa fase si cerca di costruire le traiettorie spaziali degli oggetti identificati per poter monitorare e analizzare le attività. Chiamata anche localizzazione.

4. **Activity tracking:** si vuole infine dare una caratterizzazione allo scenario complessivo, composto dai singoli oggetti identificati nella scena corrente, per avere un monitoraggio globale.

In base a quanto definito poc'anzi si può fornire un'organizzazione strutturale analoga, al fine di descrivere il contesto complessivo di un sistema automatico di sorveglianza audio. Identifichiamo il medesimo approccio bottom-up definendo i seguenti step:

1. **Background subtraction**
2. **Event classification**
3. **Object tracking**
4. **Situation analysis**

Per quanto concerne i primi due passaggi, essi rispecchiano in toto le caratteristiche dei corrispondenti nella struttura video.
Ciò non sussiste invece per l'object tracking, in quanto esso è possibile se si considera l'uso di più sensori disposti nell'aera da voler monitorare, in modo da poter estrarre informazioni quali direzione di arrivo dell'onda sonora e localizzazione spaziale. Questo compito è, sia per l'approccio audio che video, il più complicato, in quanto i livelli spesso bassi di SNR da un lato e la scarsa risoluzione spaziale dall'altro, causano seri problemi di performance.
Nella controparte audio, l'ultimo passaggio viene implementato per ottenere una descrizione completa di una scena audio. Questo tipo di analisi è conosciuta anche come *Computational Auditory Scene Recognition* (CASR) e rappresenta l'obiettivo finale di un sistema automatico di sorveglianza.
Eppure, data la sua complessità nell'essere rappresentata è tutt'oggi un task raramente descritto [6].

## 2.6 Struttura di un sistema di sorveglianza

La realizzazione di un sistema di sorveglianza intelligente distribuito su vasta scala deve comporsi di un insieme di discipline distinte. Intelligenza artificiale, telecomunicazioni e ingegneria dei sistemi sono chiaramente necessarie.
Un approccio distribuito può garantire alcuni vantaggi. In primo luogo, la comunicazione tra i nodi può consentire l'uso di sensori meno costosi e un maggior numero di sensori può essere distribuito su un'area più vasta. In secondo luogo, la robustezza è maggiore, perché anche se alcuni sensori falliscono l'integrità del sistema non è compromessa. Infine, le prestazioni sono più flessibili, vi è una distribuzione dei compiti in diversi luoghi. Per esempio, la probabilità di classificare correttamente un oggetto o un evento aumenta se più sensori sono concentrati su di esso da posizioni diverse [26].

Un'ulteriore caratteristica di una moderna rete di sorveglianza è l'interconnessione wireless dei nodi sensore. Una struttura di questo tipo rappresenta un nuovo livello di rete che integra acquisizione, elaborazione e comunicazione wireless in un sistema distribuito.
Una rete wireless implica solitamente un costo ragionevole dei dispositivi tipicamente alimentati a batterie, con moderate capacità di calcolo e comunicazione, poiché le circostanze spesso richiedono che i dispositivi siano contenuti in peso e dimensioni, con un basso consumo di energia e non intrusivi. Oltretutto, non utilizzare una comunicazione via cavo, causa un risparmio di costi che giustifica l'impiego di una rete wireless. Ogni nodo può essere equipaggiato con diverse modalità di acquisizione: acustiche, video, infrarosse e sismiche. Tutto ciò, offre controllo e informazioni all'utente in ambienti distinti.
Una rete con nodi wireless possiede le carte in regola per migliorare le capacità di un'applicazione incentrata sull'utente al fine di monitorare e prevenire eventi di pericolo.

Un tassello importante da considerare quando si costruisce una rete di sensori è il ruolo del *middleware*, ovvero il software che funge da intermediario tra i vari livelli del sistema e consente l'interfacciamento tra le varie parti.
Esso deve presentare alcune caratteristiche architetturali:

- Scalabilità: fornire strumenti adatti e scalabili per eventuali nuove implementazioni.
- Disponibilità: deve supportare una tolleranza ai guasti sufficiente a sostenere livelli accettabili di disponibilità
- Evolvibilità e integrazione: adattarsi a cambiamenti di hardware e software
- Sicurezza: offrire strutture di sicurezza per affrontare situazioni di attacco

[27]

L'evoluzione tecnologica consente quindi di poter scegliere flessibilmente l'architettura di un moderno sistema di sorveglianza autonomo, in modo da acquisire le informazioni desiderate e costruire un'architettura ad hoc. Diversi stadi di intelligenza distribuita, a livello del sensore piuttosto che in uno dei diversi blocchi dell'architettura, potrebbero aiutare un monitoraggio preventivo.
Una gestione efficiente di reti di sorveglianza moderne è necessaria al fine di raggiungere prestazioni che consentono di adattare il comportamento della rete in accordo con la vastità di situazioni in cui essa potrebbe essere coinvolta [4].

Nelle applicazioni moderne di sorveglianza, vi è una forte necessita di gestire l'enorme volume di dati generato, soprattutto quando si decide di lavorare con una rete distribuita. Principalmente, per quelle applicazioni in cui si processano dati provenienti sia dal mondo video che da quello audio.
Risulta essenziale creare sistemi i quali siano in grado di lavorare sui dati in tempo reale al fine di creare un'elaborazione grezza che raccolga gli eventi

interessanti all'interno di stream dati. Tipicamente, vengono scelte implementazioni che sfruttano soluzioni di calcolo cloud.
Tuttavia, la soluzione real-time comporta ancora alcune sfide importanti da superare, come ad esempio la trasmissione dei dati su sistemi con prestazioni di latenza e banda scarse; questo fa sì che le comunicazioni non siano veloci abbastanza da poter comunicare efficacemente dal luogo in cui vengono acquisiti i dati a dove l'elaborazione prende atto [28].
La soluzione proposta in [28], prevede l'impiego di cloud distribuiti, dispositivi edge e *fog computing*. La dislocazione della computazione avviene su più livelli e include una elaborazione su piccoli dispositivi a livello di nodo sensore, un secondo livello di computazione per raccogliere i dati in nodi fog e infine una struttura cloud con elevata capacità computazionale che permette di concludere la fase di processing.
Per fog computing si intende un'architettura che sfrutta una rete di dispositivi periferici (conosciuti anche come dispositivi edge) i quali generano grandi quantità di dati grezzi o leggermente processati. È tipicamente localizzata tra i nodi sensori e il cloud e ha il compito di performare quanta più computazione possibile in unità fisicamente locali ai sensori.

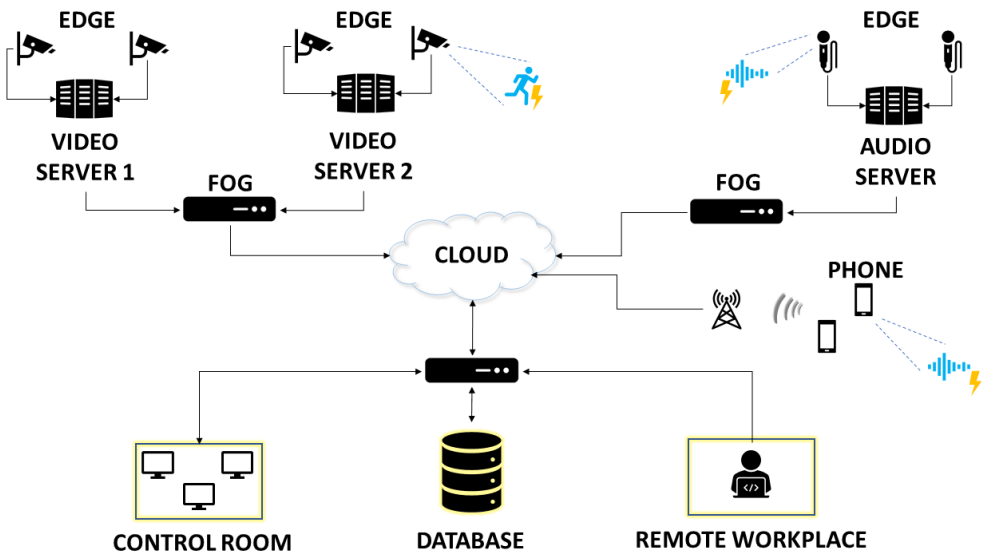


*Figura 1. Schema, puramente generico, di un sistema automatico distribuito di video/audio sorveglianza.*

In figura 1 è rappresentata una schematizzazione generale di un sistema automatico per la rilevazione di situazioni di pericolo. Tutti i livelli di distribuzione delle tecnologie sono presenti: dai dispositivi edge, al fog computing e all'interfacciamento con il cloud. Anche gli smartphone possono essere presi in considerazione tra i dispositivi edge, possono essere sfruttati come sensori mobili all'interno del sistema da monitorare e i loro dati, oltre ad aggiungersi a quelli dei nodi sensori, potrebbero essere utilizzati ipoteticamente per convalidare o smentire possibili minacce monitorando l'attività dell'utente dello smartphone stesso. Sono stati considerati inoltre una control room, dove il personale addetto alla sicurezza può monitorare congiuntamente tutte le informazioni, un database dove vengono salvati i dati provenienti dai sensori e un remote workplace, grazie al quale si potrebbero effettuare attività sia di monitoraggio che di manutenzione dei sistemi.

Al fine di fornire una contestualizzazione del progetto descritto nel presente elaborato, esso potrebbe posizionarsi sia a livello di audio server, sia a livello degli smartphone che direttamente nel cloud. La posizione dipende da quale distribuzione dell'elaborazione dei dati s'intende implementare.

# 3 TEORIA DEGLI STRUMENTI

## 3.1 Neural Network

Una rete neurale o *Neural Network* (NN), è sostanzialmente un modello matematico, la cui struttura si basa sull'interconnessione di singole unità artificiali chiamate neuroni. Si intuisce subito che il nome deriva dall'analogia con la struttura biologica cerebrale, in particolare negli studi iniziali della prima metà del secolo scorso si tenta di costruire una rappresentazione logico matematica della complessa attività del cervello. Il primo fu il *Percettrone*, avo delle odierne reti.

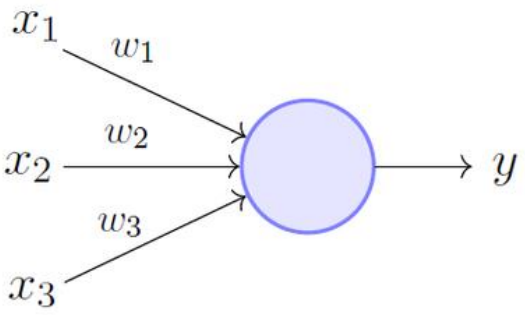


*Figura 2. Schema di base per identificare la struttura di un percettrone. $x_1, x_2, x_3$ sono i segnali di inpunt, $w_1, w_2, w_3$ sono i pesi associati ad ogni segnale, $y$ è il segnale di uscita generato dal percettrone.*

Ovviamente, queste reti, non nascono con lo scopo di rappresentare l'immensa vastità di una rete neurale biologica, ma al contrario si ispirano ad essa per sfruttare a proprio vantaggio i benefici di una così complessa struttura.
Le reti neurali hanno svariati ambiti di applicazione, tra i più importanti: Regressione, Classificazione, Computer Vision, Speech Recognition, Data compression (Encoder/Decoder) [29].

In via generale, si può descrivere una rete neurale come una funzione, la quale dato un vettore di ingresso produce un vettore di uscita. La dimensione di input e output non è la medesima e dipende dalla particolare applicazione che si sta affrontando. Ad ogni componente dei vettori è associato un neurone, ciò significa che un neurone non è altro che un'unità contenente un numero.

L’insieme dei neuroni di ingresso, e allo stesso modo quello di uscita, viene chiamato *layer*.
La rete più semplice (figura 3) vede ogni neurone di ingresso connesso ad ogni neurone di uscita, mentre modelli più complessi sono costruiti con diversi layer tra ingresso e uscita e prendono il nome di strati nascosti, *hidden layers*.
Il valore salvato in ogni neurone si chiama *attivazione* e spesso è un valore compreso tra 0 e 1. Ad esclusione del layer d’ingresso, l’attivazione viene prodotta come combinazione lineare delle attivazioni dello strato precedente e dei pesi associati ad ogni connessione.
Indicando con $a^i$ il vettore rappresentante i valori dei neuroni del layer i-esimo, $w_{kj}$ il valore del peso associato alla connessione tra il k-esimo neurone del layer successivo e il j-esimo neurone di $a^i$, è possibile scrivere

$$a_k^{i+1} = \sigma\left(\sum_{j=0}^{n-1} w_{kj} a_j^i + b_k\right)$$

dove $a_k^{i+1}$ rappresenta il valore del k-esimo neurone del layer successivo al corrente, $b_k$ è la componente di *bias* per ogni vettore del layer a valle, ed è in generale il parametro che determina quanto deve valere la somma pesata per far in modo che il neurone diventi “attivo”, $\sigma$ è la funzione di attivazione la quale mappa il valore di ingresso tra 0 e 1, ad esempio la sigmoide

$$\sigma(x) = \frac{1}{1 - e^{-x}}$$

Nelle applicazioni moderne la ReLU (*Rectified Linear Unit*) è la funzionde di attivazione più utilizzata, soprattutto per motivi legati al carico computazionale in fase di allenamento

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

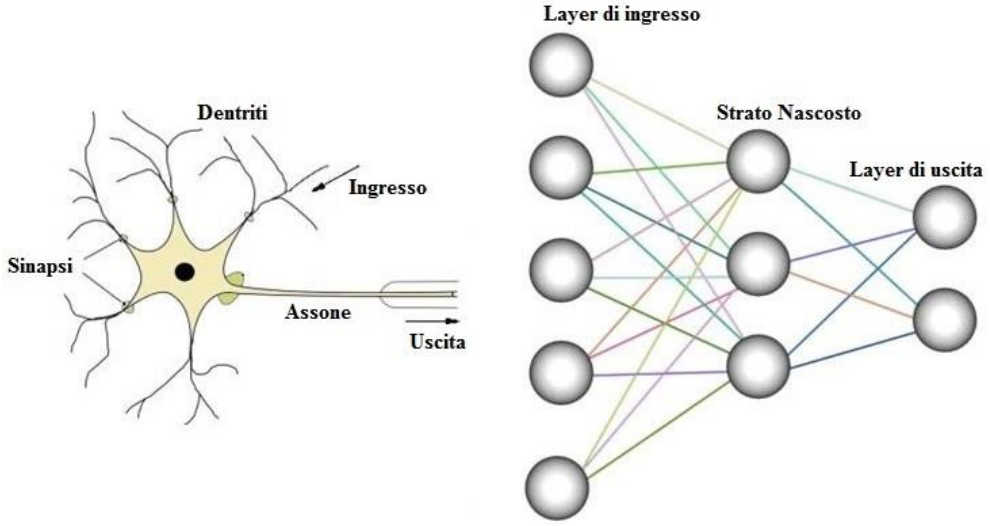


*Figura 3. A sinistra la struttura di un neurone biologico, a destra la struttura di una rete neurale ispirata dalla prima.*

Si può quindi fornire la relazione tra due layer

$$a^{i+1} = \sigma\big(Wa^i + b\big), \quad i = 0, \dots, L$$

con $L$ numero di layer.

Il vettore di uscita deve fornire dei valori coerenti con il task che si sta sviluppando. Ad esempio, ad ogni neurone può essere associata una caratteristica che la rete deve imparare a riconoscere quando viene fornito un determinato ingresso. Generalmente, in queste applicazioni, i neuroni di uscita contengono la probabilità che una particolare caratteristica sia comparsa in ingresso. Questo calcolo viene compiuto tramite la funzione conosciuta come softmax

$$h(x_j) = \frac{e^{x_j}}{\sum_k e^{x_k}}$$

A questo punto risulta semplice notare che la funzione descrivente il modello della rete neurale non è altro che una composizione di altre funzioni.

Come ogni modello, una volta definita la struttura, deve poi seguire una fase di ottimizzazione (*training*) per ottenere le migliori prestazioni possibili. Per una rete neurale si devono trovare i giusti valori dei pesi e dei bias, $w_{kj}$ e $b_k$

rispettivamente, il cui numero cresce proporzionalmente al numero di layer e alla dimensione di essi. Si può facilmente notare che il numero di parametri può diventare decisamente elevato, anche per reti non troppo grandi.
Per un task di classificazione di numeri scritti a mano, le cui immagini per ogni numero sono di dimensioni $28x28$ pixel, una semplice rete può essere costituita da: un layer di ingresso di $28x28 = 784$ neuroni, due hidden layer di 16 neuroni ciascuno ed un layer di uscita di 10 neuroni (pari al numero di digit), la quale comporta un numero totale di parametri pari a $13.002$.

Affinché sia possibile realizzare una rete neurale, si necessita un dataset, ovvero una raccolta di dati specifici del problema che sia quanto più grande possibile. Ad esempio, per il task descritto poc'anzi, esiste il dataset MNIST, il quale fornisce un totale di $70000$ immagini, dove per ogni immagine è associata una voce indicante la classe di appartenenza della stessa (*label*). Tale collezione è fondamentale per compiere l'allenamento e la verifica delle prestazioni della rete.

Al fine di eseguire l'allenamento di una rete è necessario definire una funzione costo $F(w, b)$, spesso chiamata anche funzione di perdita o *loss*, la quale ha lo scopo di identificare in che misura la rete sbaglia a compiere il suo lavoro. Solitamente essa assume un valore quanto più basso, quanto più il risultato è vicino a quello atteso, alto viceversa. Quindi l'obiettivo è quello di minimizzare questa funzione.
Essa è indispensabile in quanto viene definita in funzione dei parametri da ottimizzare, ciò si traduce nella differenziabilità della funzione costo rispetto i parametri e quindi nel calcolo di un gradiente $\nabla F$, il quale indica la "direzione" che il valore di $F$ deve prendere al fine di raggiungere un minimo locale il più velocemente possibile (*gradient descent*).

$$F(w, b, u, l) = \frac{1}{2}\sum_{d}\sum_{k}\left(u_k^d - l_k^d\right)^2$$

rappresenta una possibile funzione costo, espressa come scarto quadratico, dove $u$ è il vettore di uscita in cui vi è la dipendenza da $w$ e $b$, $l$ è il vettore delle label i cui elementi sono tutti nulli tranne quello in posizione corrispondente alla vera classe di appartenenza del dato in ingresso alla rete il quale vale 1 (questa configurazione delle label si chiama *one-hot-econded*), mentre l'indice $d$ scorre i dati del dataset e $k$ i neuroni dell'ultimo layer.
Ciò non sarebbe possibile con una metrica di misura delle prestazioni come l'accuratezza, in quanto essa dipende esclusivamente dal numero totale di dati e dal numero di risultati corretti forniti dalla rete.
Esistono svariate funzioni costo, ognuna con le sue caratteristiche, la cui scelta dipende dal tipo di applicazione. Per un problema di classificazione un'opzione classica può essere la *Categorical Cross Entropy* o la gemella *Sparse Categorical Cross Entropy*, le quali si differenziano esclusivamente per il tipo di formato in cui si esprimono le label. Nel caso in cui la classe di appartenenza di un elemento sia espressa nel formato one-hot-encoded si sceglie la prima, altrimenti se espressa con un intero $y \in 0, \dots, N° \, classi$, allora si sceglie la seconda. La formulazione della Categorical Cross Entropy è la seguente

$$F(w, b, u, l) = -\sum_{k} l_k \log(u_k)$$

ove per semplicità è stata espressa la formulazione solo per un dato, $k$ scorre i neuroni di uscita. La formulazione della Sparse Categorical Cross Entropy risulta

$$F(w, b, u, l) = -\log(u_l)$$

$l$ in questo caso è un intero e si sfrutta per accedere all'elemento del vettore di uscita per il quale si dovrebbe trovare un alto valore di probabilità. Non si necessità quindi di calcolare la somma di prodotti poiché gli altri elementi di $u$ sarebbero moltiplicati per 0.
L'approccio con cui si effettua l'allenamento di una rete è chiamato *Backpropagation error*, dove l'errore è inteso come il valore della funzione

costo, mentre backpropagation deriva dal procedimento con cui si calcola il gradiente, in quanto il metodo consiste nell'aggiornare i pesi e i bias partendo dal layer di uscita e procedendo a ritroso fino a quello iniziale. Per aumentare il valore di una specifica attivazione ad esempio, si può agire sul valore dei pesi e dei bias del layer precedente. L'algoritmo con cui si implementa il backpropagation è conosciuto come *stochastic gradient descent* (SGD) e consiste nel calcolare la discesa del gradiente in modo iterativo, esso è particolarmente utile in termini di complessità computazionale nel caso in cui si debba affrontare un problema di ottimizzazione la cui computazione della funzione costo diventi onerosa, soprattutto in presenza di dataset vasti. Questa implementazione fornisce un'approssimazione del gradiente ad ogni iterazione, valutandolo su sottoinsiemi del dataset.
La discesa stocastica del gradiente può essere formulata come segue

$$\hat{\gamma} = \gamma - \eta \nabla_{\gamma} F(\gamma, u^d, l^d)$$

ove $\gamma$ indica l'insieme dei parametri, $\eta$ il parametro di apprendimento meglio conosciuto come *learning rate*, il quale porziona l'ammontare del gradiente indicando quanto è grande il passo che $F$ deve fare per avvicinarsi al minimo. Il principale problema di SGD consiste nell'aggiornare i parametri con valori affetti da alta varianza, causando un difficile raggiungimento del minimo locale, questo è dovuto in quanto l'effettivo valore di aggiornamento del gradiente dovrebbe essere calcolato su l'intero dataset.
Inverosimilmente, le fluttuazioni introdotte in alcuni casi invece potrebbero anche portare ad un nuovo e potenzialmente migliore minimo locale.

In totale contrapposizione al SGD si trova il *Batch gradient descent* (BGD), il quale compie l'aggiornamento dei parametri con il gradiente calcolato sull'intero dataset

$$\hat{\gamma} = \gamma - \eta \nabla_{\gamma} F(\gamma)$$

e comporta una computazione più lenta e onerosa di memoria, garantendo una convergenza di $F$ verso il punto di minimo.

Nasce in modo logico, un approccio che si trova a metà strada tra i due precedenti, assumendo una sorta di media delle loro caratteristiche. Conosciuto come *Mini-batch gradient descent*, l'algoritmo aggiorna i parametri ogni $m$ valori del dataset, grandezza comunemente conosciuta come *batch-size*

$$\hat{\gamma} = \gamma - \eta \nabla_{\gamma} F(\gamma, u^{(d:d+m)}, l^{(d:d+m)})$$

Si ottiene così una convergenza più stabile rispetto al SGD ed una velocità di apprendimento migliore del BGD.

Tuttavia, nel corso degli anni, versioni più sofisticate sono state sviluppate, sicché una miglior convergenza potesse essere raggiunta. L'obiettivo era quello di ovviare ai punti di debolezza del SGD, famoso per la sua difficoltà nel superare punti di sella in problemi in cui la funzione errore non è convessa, come accade esattamente per le reti neurali le cui funzioni costo presentano superfici ardue da navigare.
Lo sforzo maggiore è stato intrapreso in direzione di un algoritmo che potesse cambiare dinamicamente il valore del learning rate durante l'allenamento e mitigare la lenta e imprecisa convergenza di un valore fissato, in quanto valori troppo grandi possono far scavalcare il punto di minimo rischiando di divergere.

Il metodo più utilizzato oggi giorno è *Adam* (*Adaptive Moment Estimation*). Esso consente di calcolare valori di $\eta$ adattivi, ovvero per ogni parametro, cercando di compensare una possibile sparsità della frequenza dei dati, associando un learning rate più alto a dati che appaiono meno spesso.
Adam è implementato attraverso stime della media $\widetilde{m_t}$ (primo momento) e della varianza non centrata $\widetilde{v_t}$ (secondo momento) del gradiente con i quali si aggiornano i parametri

$$\hat{\gamma} = \gamma - \frac{\eta}{\sqrt{\widetilde{v_t} + \epsilon}} \widetilde{m_t}$$

[30].

Concentrandosi su quanto concerne il presente lavoro, è di fondamentale importanza illustrare la struttura delle reti neurali convoluzionali (CNN) ora che le basi sono state affrontate.
Fino a questo punto le reti neurali esposte sono quelle comunemente conosciute come *fully connected* oppure reti neurali artificiali (ANN), dove ogni neurone è connesso a tutti i neuroni dello strato successivo.
Come visto, questa tipologia di reti comporta un numero elevato di parametri da allenare, con tempi lunghi di allenamento, e frequenti problemi di overfitting, ovvero l'elevata specializzazione di un modello ad un determinato set di dati. Ciò causa un'elevata probabilità di sbagliare la predizione di nuovi dati. Esistono a tal scopo diverse tecniche di regolarizzazione per ovviare a questo problema, una delle quali consiste nel cambiare l'architettura con cui la rete neurale è costruita. Le CNN forniscono una soluzione in questo senso, contenendo il numero di parametri della rete. L'idea di base si fonda sui processi biologici, per cui diversi studi hanno evidenziato come i neuroni della corteccia visiva rispondano agli stimoli. Ogni singolo neurone tende a rispondere solamente a stimoli in specifiche regioni del campo visivo, conosciute come *campo recettivo*. Ognuno di questi si sovrappone leggermente a quelli adiacenti, e la totalità di essi copre completamente il campo visivo. Intuibilmente, architetture con questa struttura sono vastamente impiegate in tutti quegli ambiti che praticano image processing, ad esempio il Computer Vision [31].
Un surplus non di poco conto, consiste nella praticità di utilizzo di queste reti, in quanto il loro impiego ridimensiona la quantità di preprocessing necessario nel preparare i dati rispetto ad altre tecniche. Questo rende le CNN estremamente più indipendenti dall'estrazione delle feature, poiché è la rete stessa ad estrarle autonomamente.

Le CNN si compongono delle seguenti tipologie di layer:

- **Convolutional**: questo particolare layer è costituito da un insieme di macrounità chiamate *feature map*, il cui numero può essere scelto in fase di definizione dello stesso. Ogni feature map consiste di unità che sono il risultato di una convoluzione tra un kernel (maschera) e una porzione di immagine di grandezza pari a quella del kernel, più un termine di bias ed una funzione di attivazione. La dimensione del kernel e il passo di avanzamento (*stride*) sono parametri anch'essi definibili nel momento in cui si costruisce il modello. Tutte le unità di una feature map condividono gli stessi pesi, quindi ogni unità è in grado di riconoscere uno specifico pattern in una qualsiasi regione dell'immagine, una volta allenato. A sua volta, ogni feature map, si specializza per riconoscere specifiche caratteristiche e andamenti frequenti in un'immagine, in modo da costruire un modello consistente. Questa operazione è l'analoga della risposta dei neuroni nella corteccia visiva ad uno stimolo specifico. Ogni neurone della rete elabora dati solo per il proprio campo recettivo, il quale definisce l'area di ingresso di un neurone.

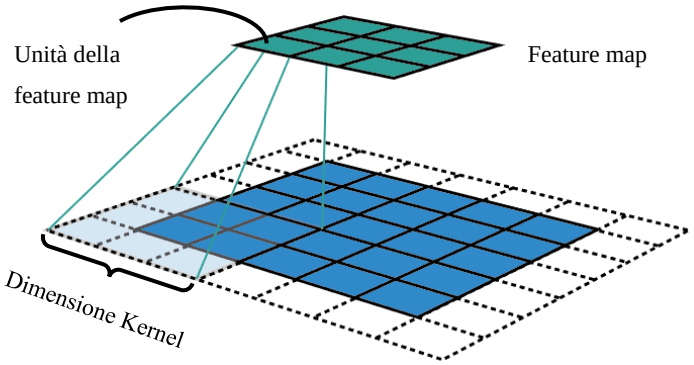


*Figura 4. Visualizzazione dell'operazione di convoluzione. In blu l'immagine su cui scorre un kernel di 3x3 con stride 2. Risultato della convoluzione è la feature map in verde. Si può inoltre specificare se aggiungere una cornice di valori attorno all'immagine, conosciuta come pad, in questo caso di dimensione 1x1.*

- **Pooling** o **subsampling**: questo layer effettua un sottocampionamento di ogni feature map. Ad esempio, ogni unità prende un ingresso di

dimensione $2x2$, il quale viene trasformato in un'uscita computando la media di questi valori oppure selezionandone solamente il massimo. Si consideri sempre la presenza di un valore di bias e di una funzione di attivazione. I campi recettivi di questo layer sono scelti per non sovrapporsi, ma per essere contigui.

- **Fully connected:** la presenza di questo layer è solitamente giustificata in task di classificazione.

[32]

In una tipica architettura convoluzionale è corrente trovare diverse combinazioni a cascata di questi layer, fino ad ottenere alle volte un gran numero di strati, sconfinando in questo modo nella categoria di reti identificate come "profonde" o *Deep Neural Network* (DNN).

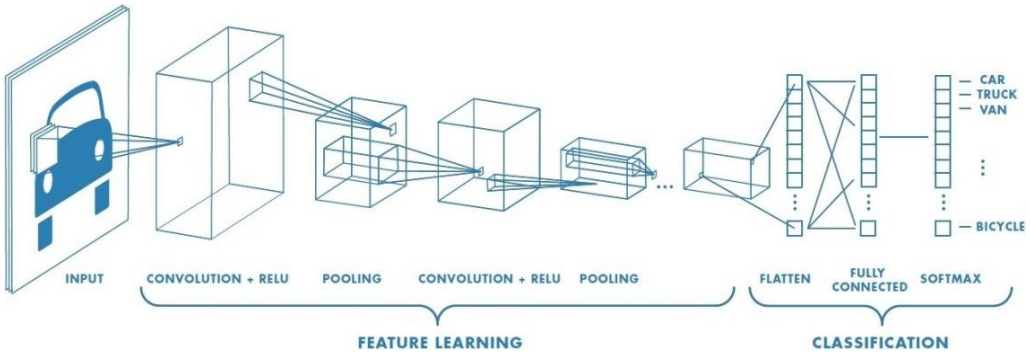


*Figura 5. Rappresentazione completa della struttura di una CNN, divisa nelle due parti principali di cui si compone: estrazione delle feature map e blocco di classificazione.*

L'allenamento di una rete convoluzionale è leggermente diverso da quello classico, l'approccio del backpropagation deve essere modificato per tener conto del vincolo imposto dalla condivisione dei pesi.

Diverse reti conosciute si basano su architettura convoluzionale, ad esempio: *LeNet*, *GoogLeNet*, *ResNet*, *AlexNet* e *VGGNet*. Molte di queste, come AlexNet e VGGNet, sono nate grazie alla vittoria della sfida *ILSVRC* (ImageNet Large Scale Visual Recognition Challenge) nel 2012 e nel 2014 rispettivamente, meritando di essere i migliori modelli per la classificazione di immagini e object-detection.

Per quanto concerne la VGG, il nome è un acronimo e sta per Visual Geometry Group, ovvero il nome del team che l'ha sviluppata. Essa è indicata anche come VGG16 per indicare il numero dei layer che hanno dei pesi allenabili.
I vantaggi di questa rete dipendono dalle seguenti scelte: la profondità, quindi un numero maggiore di layer, dimensione del kernel pari a $3x3$, lo stride pari a $1$ e dimensione del maxpool pari a $2x2$. Da questo modello, ne è stato sviluppato un secondo più profondo: *VGG19*. La sua architettura è la seguente:

| LAYER | FEATURE MAP | KERNEL SIZE |
|---|---|---|
| **Conv(3x3)**<br>**Conv(3x3)**<br>**Maxpool** | 64 | $3x3$ |
| **Conv(3x3)**<br>**Conv(3x3)**<br>**Maxpool(2x2)** | 128 | $3x3$ |
| **Conv(3x3)**<br>**Conv(3x3)**<br>**Conv(3x3)**<br>**Conv(3x3)**<br>**Maxpool(2x2)** | 256 | $3x3$ |
| **Conv(3x3)**<br>**Conv(3x3)**<br>**Conv(3x3)**<br>**Conv(3x3)**<br>**Maxpool(2x2)** | 512 | $3x3$ |
| **Conv(3x3)**<br>**Conv(3x3)**<br>**Conv(3x3)**<br>**Conv(3x3)**<br>**Maxpool(2x2)** | 512 | $3x3$ |
| **FC-4096**<br>**FC-4096**<br>**FC-1000** | / | / |

*Tabella 1. Architettura della VGG19. I layer di Maxpool non vengono conteggiati nel numero che definisce la rete. Le feature map definiscono implicitamente il numero di filtri di ogni layer convoluzionale.*

## 3.2 Support Vector Machine

Il *Support Vector Machine* (SVM) o macchina a vettori di supporto, costituisce un modello in grado di determinare l'iperpiano ottimo, al fine di ottimizzare la distanza tra due classi fornite, la cui ottimizzazione dei parametri corrisponde ad un problema di ottimizzazione convesso, da cui si deduce che ogni soluzione locale è anche un ottimo globale [32].
Questa tecnica è stata sfruttata parecchio come classificatore di eventi audio sia in ambito di sorveglianza [33], [34], che in altre tipologie di task [35], in modo particolare nel periodo compreso tra il 2009 e il 2014. È una tecnica sviluppata all'inizio degli anni novanta e rappresenta uno dei metodi più robusti di predizione, sfruttato maggiormente per la classificazione e la regressione lineare [36].
Assumendo inizialmente un dataset separabile linearmente, si consideri un insieme di $n$ campioni appartenenti a due classi distinte, una positiva ed una negativa:

$$\{(x_1, t_1), \dots, (x_n, t_n)\}$$

dove il generico $x_i \in R^n$, è un vettore a $n$ dimensioni rappresentante una determinata caratteristica estratta dall'i-esimo campione facente parte del dataset, mentre $t_i \in \{-1 + 1\}$ rappresenta la classe a cui appartiene $x_i$.
Nel caso si voglia risolvere un problema di classificazione lineare sfruttando il SVM, il modello assumerebbe la seguente forma:

$$y(x_i) = w^t \phi(x_i) + b$$

dove $w \in R^n$ è il vettore normale al piano, $b$ è la costante di bias, $\phi(x)$ rappresenta una determinata trasformazione dello spazio iniziale. I nuovi $x$ saranno classificati valutando il segno di $y(x)$, in quanto per tutti i punti correttamente classificati si avrà $t_i y(x_i) > 0$.
A parole povere, l'intento è quello di trovare un criterio di separazione tra le distribuzioni delle due classi, nello spazio a $n$ dimensioni in cui esse sono definite. Il criterio è la definizione di un iperpiano $y(x)$ a $n - 1$ dimensioni.

La soluzione dipende dai valori dei parametri $w$ e $b$, e se sono possibili multiple soluzioni allora bisogna cercare quella che fornisce l'errore di generalizzazione più piccolo.
Il Support Vector Machine affronta questa sfida basandosi sul concetto di margine, definito come la più piccola distanza perpendicolare tra il confine di decisione ed uno degli $x_i$. Il confine di decisione è scelto per essere quello per il quale il margine è massimizzato.

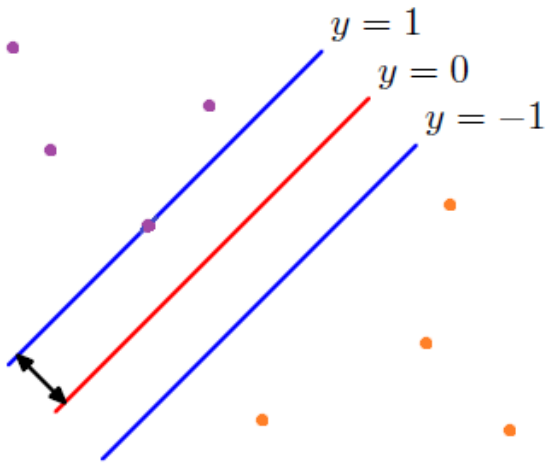


*Figura 6. Il margine (indicato con la doppia freccia in nero) è definito come la distanza perpendicolare tra il confine di decisione e il più vicino dei punti dati. La massimizzazione del margine porta ad una scelta particolare del confine di decisione.*

La distanza perpendicolare di un generico punto da un iperpiano definito da $y(x) = 0$, dove $y(x)$ prende la forma sopra descritta, è data da $|y(x)|/\|w\|$, per cui si ottiene che la distanza dalla superficie di decisione è data da:

$$\frac{t_i y(x_i)}{\|w\|} = \frac{t_i(w^t \phi(x_i) + b)}{\|w\|}$$

A questo punto si può notare la possibilità di poter scalare $w$ e $b$ di un fattore $k$ ( $w_{new} = kw, b_{new} = kb$ ) senza che la distanza tra ogni punto e l'iperpiano di decisione sia invariata. Questo consente di avere un grado di libertà da sfruttare per poter definire una relazione per il punto più vicino all'iperpiano:

$$t_i(w^t \phi(x_i) + b) = 1$$

Ciò significa che per una corretta classificazione i punti devono soddisfare il vincolo:

$$t_i(w^t\phi(x_i) + b) \geq 1 \quad i = 1, \dots, n.$$

Siccome, per definizione, esisterà sempre almeno un punto più vicino degli altri alla superficie di decisione, una volta massimizzato il margine, ci saranno sicuramente almeno due punti per cui il vincolo è soddisfatto.
Questo consente di dire che per massimizzare la distanza dal confine di decisione, semplicemente basta ottimizzare il problema

$$\arg\max_{w,b} \|w\|^{-1}$$

il che equivale ad ottimizzare un problema quadratico come il seguente

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2$$

$b$ non è presente nell'ottimizzazione, ma si può semplicemente recuperare dal vincolo

$$b = t_i - w^t\phi(x_i)$$

Al fine di risolvere questo problema si sfrutta il metodo dei moltiplicatori di Lagrange, il quale introduce dei moltiplicatori $a_i \geq 0$, per ogni vincolo selezionato da $i = 1, .., n$. Si imposta quindi la funzione Lagrangiana

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i\{t_i(w^t\phi(x_i) + b - 1\}$$

con $a = (a, \dots, a_n)^t$. Si valutano le derivate di $L(w, b, a)$ rispetto $w$ e $b$ e si pongono uguali a zero, ottenendo le due condizioni:

$$w = \sum_{i=1}^{n} a_i t_i \phi(x_i)$$

$$0 = \sum_{(i=1)}^{n} a_i t_i$$

Eliminando $w$ e $b$ da $L(w, b, a)$ si ottiene la rappresentazione duale del problema di massimizzazione del margine, in cui si ottimizza massimizzando

$$\tilde{L}(a) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j t_i t_j k(x_i, x_j)$$

rispetto ai vincoli

$$a_i \geq 0 \quad i = 1, \dots, n$$

$$\sum_{i=1}^{n} a_i t_i = 0$$

dove la funzione di kernel è definita come $k(x, x') = \phi(x)^t \phi(x')$.
Una volta allenato il modello, la classificazione di nuovi punti si effettuerà valutando il segno di

$$y(x) = \sum_{i=1}^{n} a_i t_i k(x, x_i) + b$$

Finora si è assunto di lavorare con un dataset linearmente separabile, ciò ha permesso di definire un modello il quale garantisce una separazione lineare nello spazio $\phi(x)$.
Tuttavia, quello che il più delle volte accade nelle applicazioni reali è che la distribuzione dei dati tende a sovrapporsi, motivo per il quale non vi è la possibilità di ottenere un'esatta separazione.
Dunque, bisogna modificare il SVM in modo da consentire ad alcuni punti di essere *misclassificati*, permettendo a questi di poter stare all'interno del margine, ma pesati con un coefficiente di penalizzazione, proporzionale linearmente alla distanza dal margine imposto da $|y(x_i)| = 1$.

A tal scopo, si introduce una variabile $\xi_i \geq 0$ per ogni campione del dataset. Queste sono definite $\xi_i = 0$ per ogni punto che si trova dal lato giusto del confine, mentre $\xi_i = |t_i - y(x_i)|$ per i restanti.
Ciò significa che se un punto si trova sul confine di decisione $y(x_i) = 0$, allora avrà un $\xi_i = 1$, punti misclassificati avranno $\xi_i > 1$, mentre punti con $0 < \xi_i \leq 1$ si trovano nel margine ma dal lato corretto del confine di decisione. Spesso questa condizione viene indicata come *soft margin*.

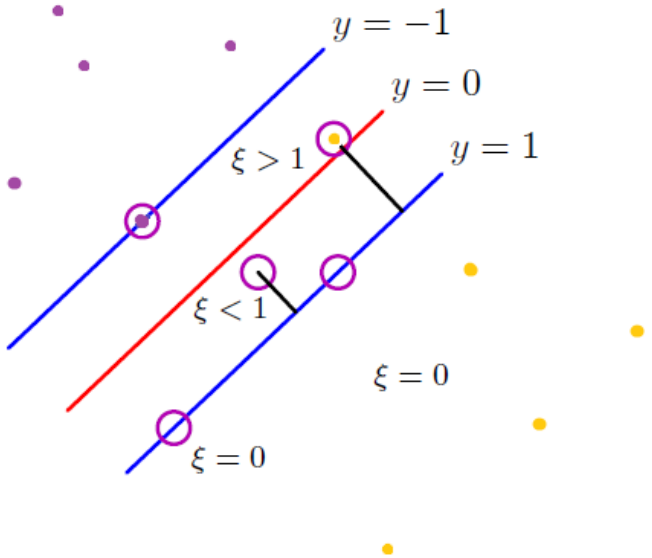


*Figura 7. Illustrazione delle variabili di allentamento $\xi_n \geq 0$. I punti di dati con i cerchi intorno a loro sono vettori di supporto.*

Il problema di ottimizzazione si trasforma nella seguente maniera

$$\arg\min_{w,b,\xi} C\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|w\|^2$$

soggetto ai vincoli

$$\begin{matrix} t_i y(x_i) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{matrix} \quad i = 1, \ldots, n$$

dove la constante $C > 0$ controlla il trade-off tra la penalizzazione e il margine, oppure può essere vista come l'inverso di un coefficiente di regolarizzazione, il

quale controlla il compromesso tra la minimizzazione del problema e la complessità di esso.
Al fine di risolvere il problema di minimizzazione si imposta lo stesso procedimento visto per il caso lineare definendo la Lagrangiana, derivando rispetto a $w$, $b$ e $\xi_i$, ponendoli uguali a zero ed eliminandoli poi dalla Lagrangiana. Si ottiene un risultato del tutto identico al caso lineare

$$\tilde{L}(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j t_i t_j k(x_i, x_j)$$

tranne che per i vincoli i quali risultano

$$0 \leq a_i \leq C \quad i = 1, \dots, n$$
$$\sum_{i=1}^{n} a_i t_i = 0$$

La classificazione di nuovi punti viene compiuta valutando il segno di

$$y(x) = \sum_{i=1}^{n} a_i t_i k(x, x_i) + b$$

relazione ottenuta ripentendo i passaggi del primo caso e con medesimo risultato.

Ciò che resta da indicare è chi sono i vettori di supporto. Per ogni punto del dataset per cui risulta $a_i = 0$, esso non contribuirà alla predizione di nuovi punti. Tutti i rimanenti punti, quelli per cui $a_i > 0$ e che soddisfano $t_i y(x_i) = 1$, sono tutti quei punti che si trovano sull'iperpiano a margine massimo e prendono il nome di *support vectors*. Nell'ipotesi di utilizzo di un approccio soft margin, la definizione di questi vettori non cambia, giocano lo stesso ruolo, e soddisfano una condizione del tipo $t_i y(x_i) = 1 - \xi_i$.

Tutto ciò implica inoltre che a valle dell'operazione di training, gran parte del dataset può semplicemente non essere considerato per effettuare predizione su nuovi punti, risultano sufficienti solamente i vettori di supporto.

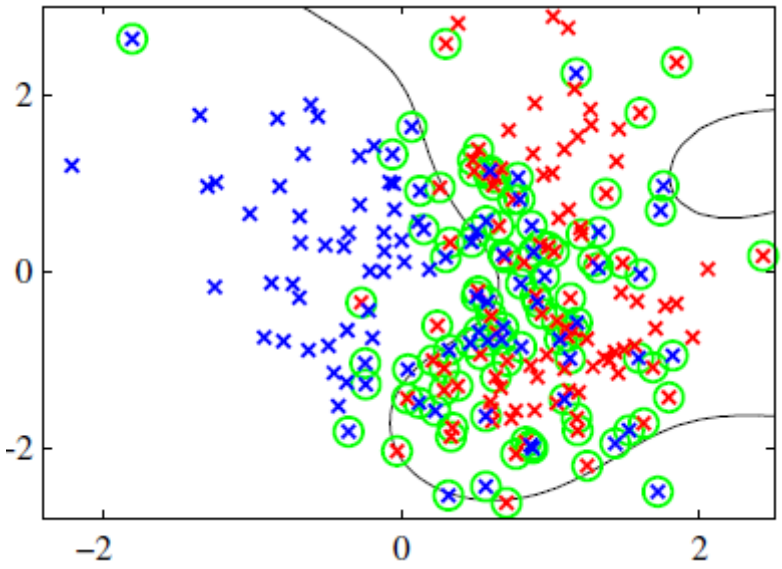


*Figura 8. Illustrazione dell'SVM applicata a un insieme di dati non separabili in due dimensioni. I vettori di supporto sono indicati da cerchi verdi.*

Al fine di giustificare la presenza della funzione di kernel $k(x, x') = \phi(x)^t \phi(x')$, essa viene spesso conosciuta come *kernel trick*, ed è definita come prodotto interno nello spazio descritto dalla funzione $\phi(x)$. Questo fornisce la possibilità di sostituire tutto ciò espresso come prodotto scalare con una scelta di kernel differente, dando vita a quella che viene chiamata come *formulazione duale*.

Nel caso dell'SVM, il kernel trick consente di effettuare una trasformazione che mappa i vettori d'ingresso in uno spazio vettoriale a dimensioni maggiori di quelle iniziali, di grande aiuto nel caso non ci sia la possibilità di definire un modello lineare nello spazio di origine. Una scelta classica è il *Gaussian radial basis function kernel*

$$k(x, x') = \exp(-\gamma \|x - x'\|^2) \quad \gamma > 0$$

[32].

L’SVM nasce come classificatore binario, ma diverse varianti sono state proposte in grado di poter effettuare una classificazione multiclasse. Le tecniche più comunemente implementate sono quelle che riducono il problema in un classificatore binario multiplo. Alcuni degli approcci basati su questo approccio sono: *One-Against-A*ll (OAA) o *One-Against-O*ne (OAO). OAA distingue una classe rispetto tutte le altre, costruendo un numero di modelli pari al numero delle classi. Un nuovo punto da classificare produrrà in uno dei modelli un’uscita più "alta" rispetto agli altri, decidendo in questo modo la classe di appartenenza; OAO invece distingue tra ogni coppia di classi, costruendo un numero di modelli pari a: $n°classi(n°classi-1)/2$ e l'assegnazione viene votata tramite un criterio *max-wins* [37].

## 3.3 Spettrogramma

La trasformata di Fourier è lo strumento tramite cui si visualizza lo spettro di un segnale, ovvero come si distribuisce l’energia di un segnale nel dominio delle frequenze. Questa trasformazione avviene tramite l’applicazione di un operatore lineare $\mathcal{F}\{\cdot\}$, con cui effettuare il cambio di dominio di una funzione tramite una combinazione lineare di esponenziali complessi. L’operatore è definito come segue:

$$X(\omega) = \mathcal{F}\{x(t)\}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} x(t)e^{-j\omega t}\,dt = \langle x(t), e^{-j\omega t}\rangle$$

Dove $e^{-j\omega t}$ rappresenta l’esponenziale complesso, facente parte della famiglia di funzioni di base su cui si sceglie di proiettare il segnale.
Le informazioni estratte generalmente da questa trasformazione sono il modulo e la fase di $X(\omega)$, ovvero $|X(\omega)|$ e $arctg\left\{\frac{Im[X(\omega)]}{Re[X(\omega)]}\right\}$ rispettivamente [38].
Questo strumento perde di efficacia di rappresentazione quando si studiano segnali non stazionari, in quanto gli esponenziali complessi peccano di località temporale e non sono in grado di trasportare nel nuovo dominio informazioni relative al “quando” una determinata componente armonica si è verificata. Più

precisamente, queste informazioni non sono perse, ma nascoste nella fase che non gioca nessun ruolo nel contributo dello spettro di energia di un segnale. In molte applicazioni, questa mancanza si traduce nell'inabilità del vettore delle feature estratte da Fourier di rappresentare il problema [39].
Per poter superare questa barriera alcuni strumenti possono essere adoperati, sfruttando una rappresentazione tempo-frequenza. Uno di questi è la trasformata di Fourier localizzata, *Short-time Fourier Transform* (STFT), in cui un parametro che tiene conto dell'informazione temporale è stato introdotto. Per fare ciò, si deve scorrere il segnale da analizzare con una finestra mobile $w(t)$, funzione reale e simmetrica, da cui si definisce una famiglia di funzioni la quale contiene le traslazioni e le modulazioni di $w(t)$:

$$g_{\omega\tau}(t) = w(t-\tau)e^{-j\omega t}$$

dove i parametri $(\omega, \tau) \in \mathbb{R}^2$.
La trasformata è quindi definita come

$$STFT_x(\omega,\tau) = \int_{-\infty}^{+\infty} x(t)w^*(t-\tau)e^{-j\omega t}\,dt = \langle g_{\omega\tau}(t), x(t)\rangle$$

questo strumento misura la somiglianza tra il segnale $x(t)$ e versioni traslate e modulate della funzione finestra $w(t)$. L'STFT è una trasformazione ridondante, in quanto il dominio del tempo viene mappato in quello del tempo-frequenza. Si definisce spettrogramma, la distribuzione di energia associata alla STFT

$$S(\omega,\tau) = |STFT_x(\omega,\tau)|^2$$

Per avere senso pratico, bisogna discretizzare $\omega$ e $\tau$:

$$(k\Omega_0, nT_0) \quad k,n \in \mathbb{Z}$$

da cui

$$STFT_x(k\Omega_0, nT_0) = \int_{-\infty}^{+\infty} x(t)w^*(t - nT_0)e^{-jk\Omega_0 t}\, dt = STFT_x(k, n)$$

La rappresentazione ottenuta è un'immagine la quale illustra come l'energia si dispone sul piano tempo-frequenza [40].

La scelta della finestra $w(t)$ gioca un ruolo fondamentale nella risoluzione del piano tempo-frequenza. Il principio di indeterminazione asserisce: è impossibile che un segnale $s(t)$ e la sua rappresentazione di Fourier $S(\omega)$ siano ben localizzate simultaneamente nel loro domino; è infattibile ottenere uno spettrogramma con una buona risoluzione temporale e frequenziale contemporaneamente. Infatti, considerando $s(t) \in L(\mathbb{R})^2$ e $\|s(t)\|_2 = 1$ e che valga la condizione $ts^2(t) \to 0$ con $t \to +\infty$ , allora:

$$\sigma_t \sigma_\Omega \geq \frac{1}{2}$$

dove $\sigma_t$ e $\sigma_\Omega$ sono deviazioni standard o la radice della durata temporale $\sigma_t^2$ e la larghezza di banda $\sigma_\Omega^2$ rispettivamente.
Pertanto, si deve scegliere a priori in quale dimensione avere miglior risoluzione. Ciò dipende da come vengono definiti i parametri della finestra $w(t)$, ovvero il numero di campioni (durata temporale) e la sovrapposizione tra finestre successive (traslazione temporale).
La famiglia di funzioni

$$g_{\omega\tau}(t) = w(t - \tau)e^{-j\omega t}$$

e le loro trasformate

$$G_{\omega\tau}(\xi) = e^{-j(\xi-\omega)\tau}W(\xi - \omega)$$

sono localizzate nell'intorno dei valori $\tau = \langle t \rangle$ e $\omega = \langle \xi \rangle$ e con durata temporale e la larghezza di banda $\sigma_t^2$ e $\sigma_\Omega^2$ rispettivamente. Ciò significa che la finestra seleziona una porzione rettangolare di area del piano tempo-frequenza pari a $\sigma_t \sigma_\Omega$, e centrata in $(t, \xi)$. L'area del rettangolo è limitata dal teorema di

indeterminazione ed è conosciuta come mattonella di Heinsenberg, la quale fissa la risoluzione del piano in ogni suo punto.

La funzione finestra è tipicamente impiegata nel progetto di filtri digitali o nell'analisi spettrale. È definita come una funzione ad energia finita, nulla al di fuori di un certo intervallo il cui obiettivo è di diminuire gli artefatti introdotti dal troncamento di una serie infinita e ridurre il fenomeno del leakage, il quale si verifica nel momento in cui si campiona uno spettro continuo. Il troncamento di una serie infinita causa nel corrispondente spettro delle ondulazioni indesiderate dello stesso, conosciute anche come fenomeno di Gibbs. Anche il fenomeno del leakage introduce delle componenti indesiderate in quanto solamente in via ideale è possibile considerare una porzione di funzione estraendone un numero intero di campioni. Quest'ultimo effetto, fa sì che lo spettro risulti decentrato dalla sua frequenza fondamentale e il suo campionamento vada ad acquisire valori non nulli dello stesso [38].
La finestra più elementare è quella rettangolare e se indichiamo con $N$ il numero di campioni della finestra, essa è definita come

$$\begin{cases} w[n] = 1 & |n| \leq \dfrac{N-1}{2} \\ 0 & otherwise \end{cases}$$

il cui corrispettivo spettro ha un andamento simmetrico secondo un seno cardinale

$$W(k) = Nsinc\left(\frac{k\omega_0}{2\pi}\right)$$

Di cui si valutano le seguenti caratteristiche: ampiezza del lobo centrale e sua larghezza di banda, ampiezza del primo lobo secondario e decadimento dei lobi. (INSERISCI IMMAGINE)
Esistono dunque, varie implementazioni della funzione finestra, ognuna delle quali cerca di migliorare una particolare caratteristica. Ad esempio, la finestra di Hamming è costruita per fare in modo di eliminare il più possibile il primo

lobo secondario essendo quello che contribuisce maggiormente nel fenomeno del leakage. Essa è definita come segue

$$\begin{cases} w[n] = 0.54 + (1 - 0.54)\cos\left(\frac{2\pi n}{N-1}\right) & |n| \leq \frac{N-1}{2} \\ 0 & otherwise \end{cases}$$

[41].

La STFT può essere interpretata anche come output di un banco di filtri. In particolare, la si può pensare come il risultato di un prodotto di convoluzione e quindi come uscita di un filtraggio lineare.

$$STFT_x(\omega,\tau) = \int_{-\infty}^{+\infty} x(t)w^*(t-\tau)e^{-j\omega(t-\tau+\tau)}\,dt =$$
$$= e^{-j\omega\tau}\int_{-\infty}^{+\infty} x(t)w^*(t-\tau)e^{-j\omega(t-\tau)}dt =$$
$$= e^{-j\omega\tau}\int_{-\infty}^{+\infty} x(t)c(t-\tau)dt =$$
$$= e^{-j\omega\tau}(x * c)(\tau)$$

dove la risposta all'impulso del filtro $c(t)$

$$c(t) = w^*(-t)e^{j\omega t}$$

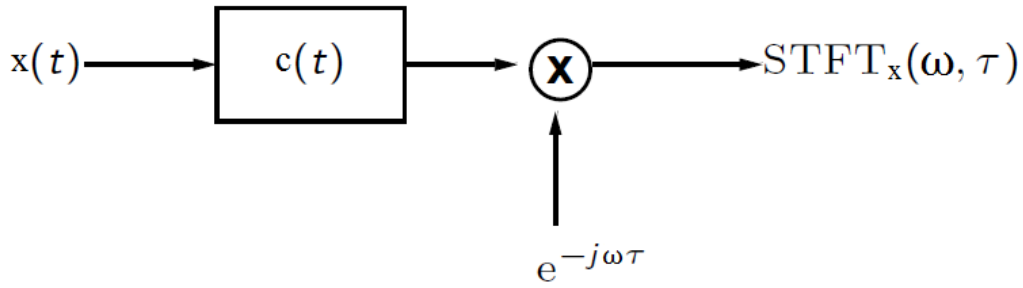


*Figura 9.Formulazione grafica del filtraggio lineare operato da $c(t)$ sul segnale $x(t)$ e successivamente modulata per ottenere i coefficienti $STFT_x(\omega,\tau)$.*

Per cui se $w(t)$ possiede un comportamento passa basso allora c$(t)$ risulterà un filtro passa banda con frequenza centrale $\omega$.
Considerando ora un insieme di filtri numerabile, si può indicare con $\omega = \omega_k = k\omega_0$ ($\Omega_0$ pulsazione di campionamento, $k \in \mathbb{Z}$ numero delle vie) e si possono indicare le espressioni dei filtri

$$c_k(t) = w(-t)e^{-j\omega_k t}$$

$$\mathcal{F}\{c_k(t)\} = C_k(\omega) = W(\omega - \omega_k)$$

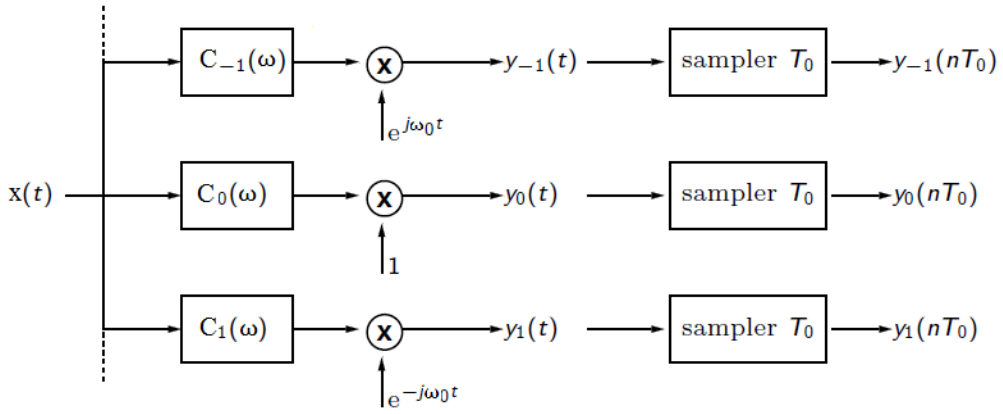


*Figura 10. Rappresentazione grafica della STFT come banco di filtri. Ad ogni $\omega_k$ corrisponde una via del banco. Il segnale $y_k(t)$ rappresenta l'evoluzione temporale dello spettro di $X(\omega)$ nell'intorno di $k\omega_0$.*

Per definire i filtri del banco è quindi sufficiente definirne uno, i restanti saranno derivati dalla traslazione e modulazione del primo $c_0$ in quanto le larghezze di banda e le ampiezze sono uguali per ogni filtro. Da quanto enunciato nel principio di indeterminazione, se la funzione finestra è stretta nel tempo allora $C_k(\omega)$ è a banda larga e viceversa. Il contributo di uscita del filtro k-esimo risulta l'evoluzione temporale dello spettro del segnale $x(t)$ nell'intorno della frequenza a cui è centrato quel filtro $k\omega_0$. Lo spettro, dunque, è analizzato da un insieme di filtri passa banda le cui frequenze centrali suddividono l'intervallo di frequenze con una distribuzione lineare.

## 3.4 Spettrogramma in scala Mel

La scala Mel, il cui nome deriva da *melody*, nasce dallo studio psico-acustico della percezione del suono da parte dell'orecchio umano. Lo studio fu condotto inizialmente da Stevens, Newman e Volkman. La sua costruzione si basa su esperimenti percettivi ai cui ascoltatori è stato presentato un tono sinusoidale di frequenza fissata, successivamente essi dovevano settare un tono variabile ad una intonazione pari alla metà del tono fisso. Al primo tono è stato assegnato un numero arbitrario, mentre un numero pari la metà del primo è stato assegnato alla frequenza media delle intonazioni giudicate la metà. Inoltre, un secondo esperimento è stato condotto al fine di risolvere il problema di una intonazione "nulla" e di come poterla definire. La scala è stata costruita quindi assegnando uguali intervalli di intonazione soggettiva ad uguali intervalli sulle ordinate del grafico che lega l'andamento della scala Mel a quella lineare. Si è fissato un valore per cui $1000\ Mels$ corrispondano a $1000\ Hz$, mentre $0\ Mels$ corrispondono a $20\ Hz$. Il risultato ottenuto garantisce di ottenere una scala di uguali distanze soggettive, in termini di intonazione, misurata in egual numero di $Mels$ [42].

Esistono diverse formulazioni per la scala Mel, la più popolare è la seguente

$$f_{mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Questa distribuzione delle frequenze suggerisce di costruire un banco di filtri le cui frequenze centrali sono mappate secondo la scala Mel. Quindi si suddivide egualmente l'intervallo di frequenze lineari in un numero pari al numero di vie del banco, per poi trasformarle secondo la nuova scala. Considerando lo spettro dell'udibile, la differenza di uno spettrogramma in scala Mel piuttosto che lo spettrogramma classico, risiede in un'analisi più approfondita della regione delle frequenze basse (al di sotto dei $1000\ Hz$ in particolare) piuttosto che quelle alte. L'immagine che si ottiene è il risultato di un iter completamente analogo a quello dello spettrogramma.

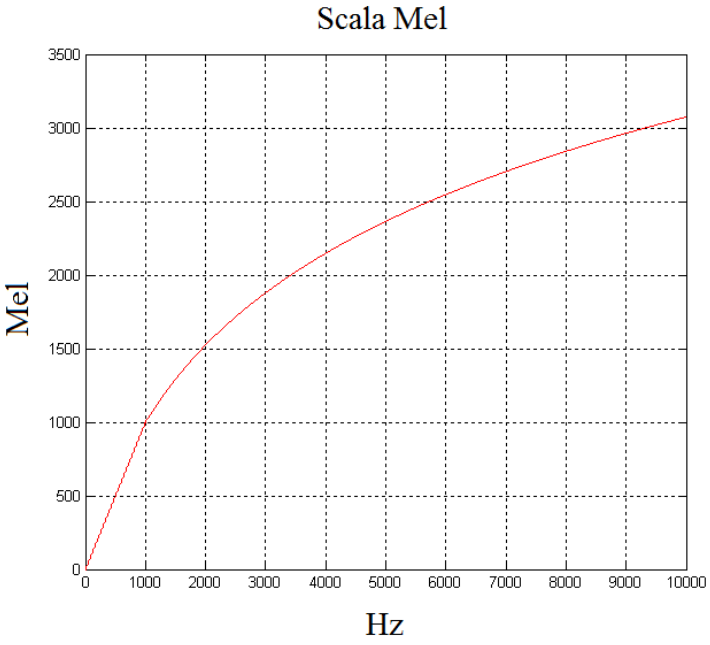


*Figura 11. Andamento della scala Mel rispetto alla scala lineare delle frequenze. Questa è secondo la formulazione di Stevens, per la quale fino a* $1000Hz$ *l'andamento è lineare, oltre è logaritmico.*

## 3.5 Gammatogramma

Si vuole ottenere un'immagine che rappresenti gli elementi da descrivere nel piano tempo-frequenza secondo un particolare filtraggio, il quale si basa sulle caratteristiche acustiche dell'apparato uditivo umano.

Il filtro gammatono è stato introdotto da *Johannsma* nel 1972 per descrivere la forma della risposta all'impulso del sistema uditivo, stimata come l'inverso della funzione di correlazione dei "tempi di sparo neurali". Successivamente sviluppata da de Boer, de Jongh e Kruidenier [43].
È stato dimostrato inoltre che questo filtro è un'approssimazione lineare della risposta impulsiva della membrana cocleare [44].
Il filtro gammatono è definito nel tempo come segue:

$$gt(t) = at^{n-1}e^{-2\pi bt}\cos(2\pi f_c t + \phi) \qquad (t \geq 0)$$

È quindi un filtro causale con un tempo di risposta infinito. Lo si può interpretare semplicemente come un tono sinusoidale portante alla frequenza $f_c$

modulato da un inviluppo: $at^{n-1}e^{-2\pi bt}$, molto simile alla famosa funzione di distribuzione statistica: funzione Gamma, da qui il nome del filtro.
I parametri principali da tenere in considerazione sono: $n$ ordine del filtro, $b$ durata dell'impulso, $f_c$ frequenza della portante, $\phi$ fase della portante, $a$ ampiezza.

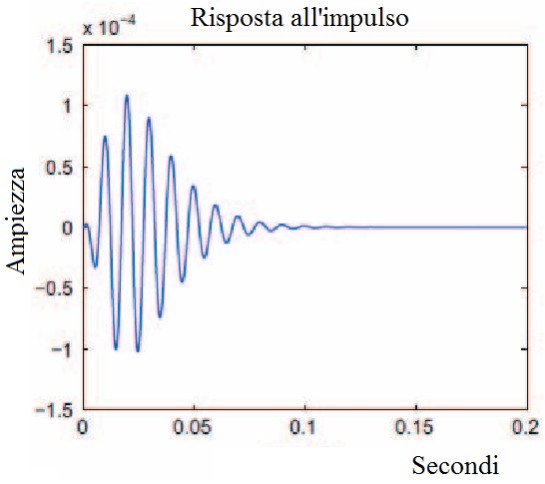


*Figura 12. Risposta all'impulso del filtro gammatono* $gt(t)$

Per ottenere la funzione di trasferimento si può pensare di agire in due modi: applicando secondo teoria la trasformata di Fourier oppure sfruttare una proprietà di questo operatore e pensare che il risultato sarà fornito dalla convoluzione in frequenza del tono sinusoidale a $\pm f_c$ con la trasformata di Fourier della funzione Gamma, data da $(1 + jf/b)^{-n}$ eccetto costante moltiplicativa.
Ponendo la fase $\phi = 0$, la quale non introduce nessun effetto importante sulle caratteristiche frequenziali del filtro, si ottiene la rappresentazione in frequenza:

$$GT(f) = \frac{a(n-1)!}{(2\pi b)^n}\left[1 + \frac{j(f-f_c)}{b}\right]^{-n} + \left[1 + \frac{j(f+f_c)}{b}\right]^{-n}$$

ove

$$-\infty < f < +\infty$$

Appaiono evidenti le seguenti considerazioni:

- $f_c$ rappresenta la frequenza di centro banda del filtro.

- $b$ rappresenta il fattore di scaling per la larghezza di banda. Per valori di $n$ fissati, all'aumentare del fattore di scaling anche la larghezza di banda aumenta.

- $n$ controlla l'intera forma del filtro. Per valori di $b$ fissato, aumentando $n$ diminuisce la larghezza di banda.

- Se il rapporto $f_c/b$ è sufficientemente grande, il secondo termine di $GT(f)$ può essere trascurato. Condizione che viene sempre soddisfatta nel caso si voglia approssimare il filtro uditivo umano, con valori di questo rapporto $4 < f_c/b < 8$.

Inoltre, quando $f_c/b$ è grande è possibile esprimere l'*Equivalent Rectangular Bandwidth* (ERB) del filtro gammatono in maniera proporzionale a $b$:

$$ERB = a_n b.$$

È possibile valutare l'$ERB$ del filtro applicando il teorema di Parseval:

$$\int_{-\infty}^{+\infty} gt(t)^2 dt = \int_{-\infty}^{+\infty} |GT(f)|^2 df = 2|GT(f_c)|^2 ERB[GT]$$

assumendo che $|GT(f)|^2$ sia simmetrico e che il suo massimo valore si trovi in corrispondenza di $f = f_c$. Mentre $ERB[GT]$ sia l'Equivalent Rectangular Bandwidth del filtro gammatono.
L'$ERB$ altro non è che l'approssimazione della larghezza di banda di un determinato filtro, assumendo di modellarlo come se fosse un filtro passa-banda rettangolare, garantendo che l'energia non cambi facendo questa assunzione [43].

Una prima relazione per l'$ERB$ in Hz è stata fornita da *Moore* e *Glasberg* nel 1983:

$$ERB(f_c) = 6.23 * 10^{-6} f_c^2 + 93.39 * 10^{-3} f_c + 28.52$$
$$100 < f_c < 10000$$

i quali nel 1990 hanno fornito una seconda approssimazione lineare:

$$ERB(f_c) = 24.7(0.00437 f_c + 1) \qquad 100 < f_c < 10000$$

[45]

Tramite le due ultime relazioni è possibile trovare la corrispondente larghezza di banda di un filtro rettangolare ideale fornendo la frequenza centrale del filtro reale $f_c$, la quale nel caso del filtro gammatono corrisponde alla frequenza del tono sinusoidale e al corrispondente valore massimo della funzione di trasferimento.

Sfruttando la relazione ottenuta con Parseval è possibile inoltre ottenere un'approssimazione di $a_n$, la quale dipende dall'ordine del filtro $n$. Essa risulta utile nel caso si voglia trovare il valore di $b$ per descrivere il filtro, conoscendo a priori il valore di $ERB$ fissata la frequenza centrale $f_c$ [43].

Queste relazioni risultano particolarmente utili nel caso si voglia costruire un *banco di filtri uditivo*, composto da un insieme di filtri passa banda non uniformi, designati per imitare al meglio la risoluzione dell'udito umano. Questi filtri spesso sono realizzati tramite funzioni gaussiane, esponenziali approssimati o molto spesso filtri gammatono, scelti spesso per la loro precisione nell'approssimazione della risposta frequenziale uditiva.

Al fine di realizzare un banco di filtri non uniforme, è necessario indicare le frequenze centrali di ogni filtro con lo scopo di descrivere al meglio la percezione dei suoni dell'orecchio umano a diverse frequenze. A tal scopo si necessita una relazione in grado di mappare le frequenze lineari secondo una scala che descriva la distribuzione di queste bande [46].

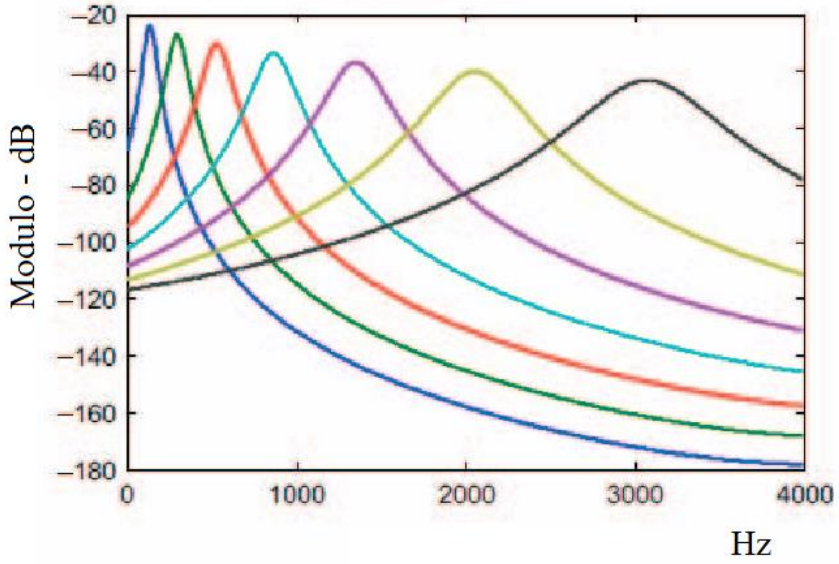


*Figura 13. Rappresentazione di $GT(f)$ di filtri gammatono a diversa scala. Le frequenze centrali di ogni filtro si trovano in corrispondenza del valore massimo dei $GT(f)$ e sono distribuite su una scala ERB.*

Queste scale cercano di replicare ciò che viene identificato come *filtro uditivo*. La conformazione anatomica della membrana basilare dell'orecchio interno fa sì che determinate frequenze risuonino maggiormente in punti diversi lungo la membrana. Questo comporta un'organizzazione della sensitività degli intervalli frequenziali di tipo tonotopico (a regioni), ciò conferisce significato ad una rappresentazione del filtro uditivo globale come ad una collezione di filtri passa banda, ognuno dei quali presenta caratteristiche uniche, frequenza di centro banda e larghezza di banda. Quest'ultima prende il nome di banda critica. L'insieme dei filtri uditivi determina il livello di discriminazione di diversi suoni di un determinato soggetto.

Per ottenere una stima di larghezza di banda e frequenza centrale di ogni filtro uditivo, diversi esperimenti psicoacustici sono stati sostenuti. Essi forniscono dei grafici dell'andamento della soglia dell'udibile di un tono in funzione di parametri di mascheramento del tono.

Uno dei più famosi è il *notched-noise method*. Consiste nel presentare ad un soggetto un tono sinusoidale puro, miscelato con un rumore filtrato nell'intorno

della frequenza del tono. Il rumore serve per valutare la soglia dell'udibile del soggetto a diverse frequenze implementando quindi diversi livelli di SNR. Queste misure sono molto lunghe e devono essere ripetute per numerose configurazioni di SNR, larghezza di notch del rumore, frequenza del tono e disallineamento di quest'ultima rispetto alla frequenza di centramento del notch, in quanto i filtri uditivi sono asimmetrici rispetto la frequenza centrale [47].
Due delle scale più usate sono la scala Bark e la scala ERB. La scala Bark fu introdotta da *Zwicker*, il quale definì 24 bande critiche dell'udito, fornendo i valori di frequenza necessari al fine di poter implementare le bande critiche. Mentre la scala ERB cerca di fornire una migliore spiegazione del perché il volume rimane costane man mano che la larghezza di banda di un suono aumenta fino alla larghezza di banda critica.
La scala ERB viene definita come il numero di ERB al di sotto di una determinata frequenza:

$$ERBS(f) = 21.4 \log_{10}(0.00437f + 1)$$

Al fine di costruire un banco di filtri, è necessario determinare la distribuzione delle frequenze delle varie bande critiche, ciò è possibile fissando valori uniformemente spaziati tra due ERBS d'interesse (corrispondenti alla massima e minima frequenza di lavoro) calcolando successivamente le corrispondenti frequenze, invertendo la relazione di ERBS.

Per costruire un'immagine tipo spettrogramma secondo un filtraggio gammatono, ed ottenere ciò che si potrebbe identificare come gammatonogramma, Dan Ellis suggerisce un'implementazione alternativa alla costruzione di un banco di filtri, molto più conveniente computazionalmente ma leggermente meno precisa in quanto ignora la fase di ogni canale.
La tecnica consiste nel costruire una matrice di pesi in base alle caratteristiche del modulo di ogni filtro del banco e combinarla con uno spettrogramma classico in modo da ottenere la conversione [48].

## 3.6 Coefficienti del Cepstrum in scala Mel

Considerando un vettore monodimensionale, il vettore dei coefficienti del Cepstrum in scala Mel, o *Mel Frequency Cepstrum Coefficient* (MFCC), si costruisce secondo i seguenti passi:

- Si partiziona il segnale audio in finestre successive, *frame*, tipicamente applicando una funzione di Hamming
- Si ottiene la trasformata di Fourier di ogni frame, considerando solamente il contributo del modulo
- Si costruisce un banco di filtri triangolari secondo una scala Mel, il cui numero di vie è fissato generalmente intorno a 40, al fine di accentuare le frequenze importanti
- Si calcola il logaritmo dei coefficienti uscenti dal banco
- Infine, vista la correlazione presente tra ogni frame (come nel caso di tracce vocali), si opera una decorrelazione. È possibile optare per diverse soluzioni, solitamente si adopera la Trasformata Coseno Discreta (DCT):

$$c_n = \sum_{i=0}^{N-1} s_i \cos\left[n\left(i - \frac{1}{2}\right)\frac{\pi}{N-1}\right]$$

  ove $N$ sono le vie del banco, $n = 0, \dots, N-1$, $s_i$ è il logaritmo dei coefficienti uscenti dal banco

Gli MFCC sono nati originariamente per affrontare problemi di classificazione legati al mondo dello speech recognition e ambiti audio correlati, in quanto è dimostrato che il contenuto consistente di energia del parlato risiede al disotto dei 5 $KHz$. Spesso usati in combinazione con feature temporali come *Zero Crossing Rate* o *Spectral Centroid* [49] [6].

## 3.7 Matrice di Confusione

Nel settore del machine learning ed in particolare nella classificazione statistica, si definisce matrice di confusione la rappresentazione delle previsioni di un modello. L'organizzazione di una matrice di confusione prevede generalmente di posizionare sulle righe l'informazione che descrive la vera classe di appartenenza delle istanze e sulle colonne le previsioni effettuate dal modello. Queste informazioni possono trovarsi anche trasposte senza perdita di significato. In parole brevi questo strumento permette di valutare rapidamente l'efficacia di un modello e in che modo esso tende a sbagliare.

Una matrice di confusione è uno strumento semplice e potente, ed è possibile introdurre il suo funzionamento con un esempio di un classificatore binario, il quale è impiegato nel caso in cui il dataset è composto da due classi, identificabili tramite due etichette, *label*.

A valle di tutto il processo di manipolazione e suddivisione del dataset, costruzione del classificatore e suo allenamento, si devono valutare le sue performance.

Un classificatore binario a fronte di una istanza conosciuta in ingresso produce un'uscita con due valori possibili, ad esempio 1 o 0, o anche indicati come positivo e negativo rispettivamente.

Ipotizzando di avere un dataset di $n = 100$ istanze, di cui 60 appartengono alla classe 1 e le restanti alla classe 0, le 100 uscite prodotte possono essere organizzate secondo una matrice di confusione come segue:

| $n = 100$ | **Positivi Previsti (1) Tot = 48** | **Negativi Previsti (0) Tot = 52** |
|---|---|---|
| **Veri Positivi (1) Tot = 60** | 46 | 14 |
| **Veri Negativi (0) Tot = 40** | 2 | 38 |

*Tabella 2. Esempio di matrice di confusione per un classificatore binario.*

Con lo scopo di mettere in risalto il confronto tra *vero* e *previsto* che la matrice di confusione compie, si definiscono solitamente quattro metriche fondamentali:

- True Positive (TP): numero delle istanze identificate come positive e che il classificatore prevede come tali.

- True Negative (TN): numero delle istanze identificate come negative e che il classificatore prevede come tali.

- False Positive (FP): numero delle istanze identificate come negative ma che il classificatore identifica come positive.

- False Negative (FN): numero delle istanze identificate come positive ma che il classificatore identifica come negative.

La matrice può essere generalizzata come segue:

| ***n*** | **Veri Previsti (1)** | **Falsi Previsti (0)** |
|---|---|---|
| **Veri Positivi (1)** | TP | FN |
| **Veri Negativi (0)** | FP | TN |

*Tabella 3. Generalizzazione matrice di confusione per un classificatore binario.*

A questo punto risulta evidente la semplicità della matrice di confusione di presentare la previsione prodotta da un classificatore e di mostrare anche dove tende a sbagliare. Nell'esempio si capisce che il classificatore ha una difficoltà maggiore nel classificare istanze positive, mentre lavora quasi perfettamente con quelle negative. Ciò potrebbe suggerire di fare un passo indietro nel flusso

di costruzione del modello per provare a capire qual è il punto debole del classificatore.
Al fine di fornire una prospettiva non basata esclusivamente su numeri assoluti si sfruttano ulteriori metriche

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

  Totale delle istanze correttamente classificate come positive in proporzione al totale delle istanze

- $ERR = \frac{FP+FN}{TP+TN+FP+FN} = 1 - Accuracy$

  Totale delle istanze correttamente classificate come negative in proporzione al totale delle istanze

- $Precision = \frac{TP}{TP+FP}$

  Esprime il tasso di corretta classificazione delle sole istanze positive

- $Recall = \frac{TP}{TP+FN}$

  Esprime il tasso di corretta classificazione delle istanze positive rispetto a tutte le istanze che erano effettivamente positive, riferendosi a monte del processo di classificazione.

- $SP = \frac{TN}{TN+FP}$

  Conosciuta come Specificità (analogo del Recall)

- $FPR = \frac{FP}{TN+FP} = 1 - SP$
- $F - score = \frac{2*Recall*Precision}{Recall+Precision} = \frac{2}{Recall^{-1}+Precison^{-1}}$

  Esprime una media armonica di Precision e Recall, utile nel caso si vogliano confrontare modelli di classificatore differenti. La media armonica consente di attribuire un peso maggiore ai valori più piccoli.

Nel caso si debba trattare un classificatore in cui le classi siano in numero maggiore di 2, si deve considerare di costruire un modello di classificatore multiclasse. Questo comporta che la matrice di confusione deve essere generalizzata a supportare anche questa casistica.
In realtà, la sua struttura e costruzione non cambia affatto, si troveranno tante righe e colonne quante sono il numero di classi da trattare. Anche le metriche sopra citate non si modificano, semplicemente la loro computazione deve essere ripetuta per la totalità delle classi, considerando alla fine di fornire un unico valore per metrica fornendo una media dei valori ottenuti [50].

Al fine di fornire una visione più completa della matrice di confusione, è possibile fornire un'interpretazione statistica sfruttando il teorema di Bayes, il quale viene adoperato come principio di collegamento.

Tutto ciò consente di evidenziare le caratteristiche probabilistiche delle performance estratte dalla matrice di confusione.
Il teorema di Bayes viene espresso come segue:

$$p(\lambda \mid X) = \frac{p(X \mid \lambda)\, \pi(\lambda)}{\int p(X \mid \lambda')\, \pi(\lambda')}$$

dove $X$ rappresenta il dato osservato, mentre $\lambda$ il parametro di interesse. A sinistra dell'uguale si trova la densità di probabilità a posteriori di $\lambda$ dato $X$, mentre a destra si trova $\pi(\lambda)$, il quale esprime la densità di probabilità a priori di $\lambda$, $p(X \mid \lambda)$ probabilità condizionata di $X$ dato $\lambda$ o *likelihood*. Il denominatore è chiamato likelihood marginale o prova del modello, ed esprime una costante di normalizzazione pari a $p(X)$.
Ciò che può essere evinto, considerando il teorema è lo sfruttare $X$ per aggiornare la probabilità a priori $\pi(\lambda)$ di $\lambda$, fornendo la probabilità a posteriori $p(\lambda \mid X)$.
In un contesto di un classificatore binario: $\lambda$ è la vera casse di appartenenza di una istanza $X$, la quale assume due soli valori. $X$ può essere espresso in due

modi, sia da un'etichetta $l$ assegnata dal classificatore, o da uno score $q$, il cui valore compreso tra 0 e 1 esprime la probabilità che $X$ appartenga ad una delle due classi. Si può quindi esprimere la probabilità a posteriori sia $p(\lambda \mid l = 0)$ oppure $p(\lambda \mid q \geq q_T)$, dove $q_T$ esprime una soglia.
Per precisare meglio, la probabilità a priori, nel caso di un classificatore binario, esprime la probabilità di pescare una delle due classi, e solitamente è chiamata *prevalenza*. Essa non è una proprietà del classificatore ma dell'insieme delle istanze. Per caratterizzare completamente la probabilità a priori basta conoscere $\pi(\lambda = 1) = \pi(\lambda_1) = \pi_1$, in quanto $\pi(\lambda_0) = \pi_0 = 1 - \pi(\lambda_1)$.
La likelihood $p(X \mid \lambda)$ invece, esprime la probabilità condizionata che il classificatore assegni una label ad una determinata istanza. Considerando di lavorare con le label $l$, si possono esprimere le quattro metriche fondamentali nel seguente modo:

- True Positive rate: $S_e \equiv p(l_1 \mid \lambda_1)$.
- True Negative rate: $S_p \equiv p(l_0 \mid \lambda_0)$
- False Positive rate: $\alpha \equiv p(l_1 \mid \lambda_0) = 1 - p(l_0 \mid \lambda_0)$
- False Negativ rate:e $\beta \equiv p(l_0 \mid \lambda_1) = 1 - p(l_1 \mid \lambda_1)$

Da notare che queste metriche sono indipendenti dalla prevalenza, quindi rappresentano proprietà intrinseche del classificatore. Non resta che analizzare il fattore di normalizzazione, la prova del modello.

Nel caso di classificatore binario esso ha due sole componenti:

- Positive labeling rate

$$p(l_1) = p_1 = p(l_1 \mid \lambda_0)\,\pi_0 + p(l_1 \mid \lambda_1)\,\pi_1$$

- Negative labeling rate

$$p(l_0) = p_0 = p(l_0 \mid \lambda_0)\,\pi_0 + \; p(l_0 \mid \lambda_1)\,\pi_1$$

Ora è possibile esprimere il teorema di Bayes, in particolare la probabilità a posteriori, nei termini appena citati, costruendo quattro combinazioni:

- Positive predictive value

$$ppv \equiv p(\lambda_1 \mid l_1) = \frac{p(l_1 \mid \lambda_1)\pi_1}{p_1} = \frac{S_e\pi_1}{p_1}$$

- Negative predictive value

$$npv \equiv p(\lambda_0 \mid l_0) = \frac{p(l_0 \mid \lambda_0)\,\pi_0}{p_0} = \frac{S_p\pi_0}{p_0}$$

- False discovery rate

$$fdr \equiv p(\lambda_0 \mid l_1\,) = \frac{\alpha\pi_0}{p_1} = \; 1 - ppv$$

- False omission rate

$$for \equiv p(\lambda_1 \mid l_0) = \frac{\beta\pi_1}{p_0} = 1 - npv$$

Secondo un'organizzazione tabellare:

| | **Positivi Previsti (1)** | **Negativi Previsti (0)** |
|---|---|---|
| **Veri Positivi (1)** | $ppv = \frac{S_e\pi_1}{p_1}$ | $for = \frac{\beta\pi_1}{p_0}$ |
| **Veri Negativi (0)** | $fdr = \frac{\alpha\pi_0}{p_1}$ | $npv = \frac{S_p\pi_0}{p_0}$ |

*Tabella 4. In questa rappresentazione si può distinatmente notare la forte corrispoendeza con la generalizzazione della matrice di confusione mostrata in tabella 3.*

Ad esempio, il Positive predictive value, esprime la probabilità a posteriori che un evento classificato come positivo sia effettivamente positivo.

Si può considerare di introdurre anche la probabilità congiunta:

$$p(l, \lambda) = p(l \mid \lambda)\pi(\lambda)$$

Proiettando ora sull'asse $l = \lambda$, si ottiene l'accuracy:

$$A \equiv p(l = \lambda) = p(l_0 \mid \lambda_0)\,\pi_0 + p(l_1 \mid \lambda_1)\,\pi_1 = S_p(1 - \pi_1) + S_e\pi_1$$
$$= 1 - \alpha + (\alpha - \beta)\pi_1$$

L'ultimo passaggio permette di affermare che l'accuracy dipende dalla prevalenza e non è quindi proprietà intrinseca del classificatore.

A questo punto è possibile fornire un elenco di approssimazioni delle proprietà studiate, espresse in termini degli elementi di una matrice di confusione:

- Prevalenza: $\pi_1 \approx \frac{tp+fn}{tp+tn+fp+fn}$
- Fattore di normalizzazione: $p_1 \approx \frac{tp+fp}{tp+tn+fp+fn}$
- True positive rate (recall): $S_e \approx \frac{tp}{tp+fn}$
- True negative rate (specificity): $S_p \approx \frac{tn}{tn+fp}$
- False positive rate: $\alpha \approx \frac{fp}{tn+fp}$
- False negative rate: $\beta \approx \frac{fn}{tp+fn}$
- Positive predictive value (precision): $ppv \approx \frac{tp}{tp+fp}$
- False predictive value: $npv \approx \frac{tn}{tn+fn}$
- False discovery rate: $fdr \approx \frac{fp}{tp+fp}$
- False omission rate: $for \approx \frac{fn}{tn+fn}$
- Accuracy: $A \approx \frac{tp+tn}{tp+tn+fp+fn}$

[51].

# 4 PAPER DI RIFERIMENTO

## 4.1 SoReNet

Il primo lavoro scelto come linea guida per il progetto esposto nel presente elaborato è lo studio di *Greco et al.* redatto in [52].

Incoraggiati dal crescente interesse della comunità scientifica e dai pregressi lavori verso l'ambito della sorveglianza audio, hanno proposto un sistema automatico per la rilevazione di situazioni di pericolo il quale incentra la sua struttura sulle ormai affermatissime reti neurali.

L'obiettivo è stato quello di sfruttare la capacità delle reti neurali, dimostrate in problemi analoghi affrontati in videosorveglianza, per costruire un modello di rilevazione automatica di eventi audio. In particolare, sfruttare la comprovata abilità delle reti convoluzionali, CNN. Nella maggior parte dei casi, esse sono state applicate direttamente ai dati grezzi delle immagini o video per effettuare analisi facciali, riconoscimento di oggetti o di attività. Raramente si è tentato di alimentare le reti con feature di alto livello.

Diversamente accade per la controparte audio, nelle cui applicazioni si tende spesso a sfruttare feature complesse, come MFCC, spettrogrammi o rappresentazioni tempo-frequenza date in ingresso a reti convoluzionali. Le reti neurali hanno dimostrato una grande capacità nell'estrazione automatica di feature ad alto livello in contesti di immagini e video. In aggiunta, i lavori presenti in letteratura confermano le migliori prestazioni ottenute grazie a rappresentazioni tempo-frequenza. L'insieme di queste considerazioni, hanno condotto alla combinazione: CNN allenata con immagini tempo-frequenza. La rete ha preso il nome di *SoReNet*, derivato da *Sound Recognition Network*, la cui architettura si ispira alla VGG19. È composta da dodici layer convoluzionali con kernel di dimensione $3x3$, alternati da quattro layer Max pooling. Mentre per i layer completamente connessi è stata scelta una configurazione piramidale, con un numero di neuroni che partono da 2048 fino

ad un layer finale pari al numero di classi da riconoscere. La struttura a piramide rende la rete più stabile a variazioni delle dimensioni del training set, quindi utile nel caso non si posseggano dataset grandi. Quest'ultima peculiarità è di estrema importanza in quanto per la corrente applicazione di sorveglianza audio, i dataset disponibili pubblici sono piuttosto piccoli. La funzione ReLU è stata scelta come funzione d'attivazione, mentre per il layer di uscita una funzione softmax.

| LAYER | FEATURE MAP | KERNEL SIZE |
|---|---|---|
| **Conv(3,64)**<br>**Conv(64,64)**<br>**Maxpool 1x2** | 64 | 3*x*3 |
| **Conv(64,128)**<br>**Conv(128,128)**<br>**Maxpool 2x2** | 128 | 3*x*3 |
| **Conv(128,256)**<br>**Conv(256,256)**<br>**Conv(256,256)**<br>**Conv(256,256)**<br>**Maxpool 2x1** | 256 | 3*x*3 |
| **Conv(256,512)**<br>**Conv(512,512)**<br>**Conv(512,512)**<br>**Conv(512,512)**<br>**Maxpool 2x1** | 512 | 3*x*3 |
| **FC-2048**<br>**FC-512**<br>**FC-128**<br>**FC-32**<br>**FC-8**<br>**FC-4** | / | / |
| ***softmax*** | | |

*Tabella 5. Architettura di dela rete neurale convoluzionale SoReNet.*

Il metodo proposto processa le tracce audio attraverso diversi passaggi. Siccome, gli eventi d'interesse possono avere durate molto diverse se si considerano ad esempio un colpo di pistola e un urlo, è stato scelto di frammentare gli stream audio tramite una finestra mobile di Hamming di durata pari a 3 secondi, con overlap del 75% tra due finestre successive. Per ogni istanza è stato ottenuto uno spettrogramma. Con questa scelta si giustifica l'utilizzo di una rete convoluzionale in quanto si è trasformato il problema audio in una classificazione di immagini, le cui dimensioni sono $(50x200x3)$.

Il dataset utilizzato è il *MIVIA Audio Events Dataset*, proposto dagli stessi autori dell'articolo, da cui sono state estratte 50.000 istanze per la fase di training e 33.000 per la fase di testing. Le classi d'interesse sono: *glass*, *gunshots*, *scream* e *background*, quest'ultima è identificata come classe per poter discriminare situazioni di pericolo da quelle normali. Per l'allenamento della rete è stato scelto un algoritmo mini-batch gradient descent, con una ottimizzazione adattiva del learning rate basata sull'approccio RMSprop.

Due diversi allenamenti sono stati condotti: un "allenamento da zero" con inizializzazione randomica dei parametri dell'intera rete e un "allenamento fine", ovvero preallenando la rete con il dataset di ImageNet solamente per i layer convoluzionali, successivamente un allenamento con il training set MIVIA per tutta la rete. La performance migliore è stata ottenuta nel secondo caso, ottenendo un'accuratezza pari a 88.9%.

Al fine di provare l'efficacia della rete, il modello è stato confrontato sia con metodi basati su rete neurale come: VGG19, utilizzata largamente in applicazioni di computer vision; AENET, modello proposto in [53] per rilevazione di eventi audio; NASNet, la quale ha raggiunto la più alta accuratezza nel task del riconoscimento di immagini; sia con approcci classici come il SVM combinato con feature temporali, spettrali, di energia o una rappresentazione mediante BoW. SoReNet allenata mediante l'allenamento fine ha ottenuto la miglior accuratezza nel confronto con queste tecniche.

## 4.2 MC SVM

Come secondo punto di riferimento, una delle tecniche proposte in [54] è stata presa in considerazione. *Shivam et al.* hanno sviluppato tre tecniche per il riconoscimento automatico di eventi audio appartenenti a sette categorie con caratteristiche eterogenee, quali: *aircraft, construction, music, nature, speech, vehicle* e *train*. Le tecniche scelte sono il SVM, ANN e CNN monodimensionale, ognuna considera feature MFCC. Il contributo a cui il presente lavoro ha prestato attenzione è la costruzione del modello di classificazione mediante SVM, a cui è stato dato il nome di *Multi-class* SVM (MC SVM). Data la differenza, in termini di durata, dei vari eventi audio, la lunghezza della finestra con cui estrarre istanze risulta un fattore chiave. Lo studio quindi è stato svolto con tre lunghezze differenti: $100ms$, $250ms$ e $500ms$.

Il datatset è stato creato con suoni sia provenienti da piattaforme di streaming video e audio, sia auto registrate con dispositivi quali smartphone e laptop PC, in diversi contesti. La frequenza di campionamento è pari a $16KHz$ e la risoluzione $16bit$. Durata totale del dataset pari a 81 ore.

Il primo step eseguito è una fase di preelaborazione, la quale elimina le componenti frequenziali al di sotto dei $10Hz$ per pulire le tracce audio da eventuali artefatti che i sensori possono introdurre essendo esposti a sorgenti sonore problematiche, come ad esempio il vento. Successivamente le tracce audio sono suddivise in frame successivi, la media del segnale sottratta e l'ampiezza delle tracce normalizzata.

A questo punto, le feature MFCC sono state estratte, in particolare: viene usato un filtro di pre-enfasi per amplificare le alte frequenze in modo da ottenere uno spettro più bilanciato, ogni frame è moltiplicato per una funzione finestra di Hamming, successivamente tramite FFT si ottiene lo spettro, il cui modulo è

dato in ingresso ad un banco di 26 filtri triangolari disposti lungo una scala Mel. Infine, al logaritmo dei coefficienti è stata applicata una DCT con lo scopo di decorrelarli. Dei 26 coefficienti ottenuti, solo i primi 13 sono stati considerati.

Per quanto riguarda il SVM, si è scelto un classificatore *one-against-all* di tipo *soft-margin* (paramentro di penalty $C = 1$), con una funzione di kernel di tipo *Gaussian radial basis* ($\gamma = 0.0769$).

# 5 IMPLEMENTAZIONE DEI MODELLI

## 5.1 Descrizione del progetto

Le scelte di progetto sono state prese considerando le tecniche proposte nei lavori di riferimento. In particolare, l'obiettivo è stato quello di implementare un modello di riconoscimento automatico di eventi audio basato sulla rete neurale SoReNet, allenata con tre diverse rappresentazioni tempo-frequenza: spettrogramma, spettrogramma in scala Mel e gammatonogramma, verificare la fattibilità della scelta e confrontarne poi le performance con il classificatore implementato tramite SVM, seguendo i passaggi descritti nel secondo lavoro di riferimento.

A valle di ciò, è stato simulato un *Detector* di situazioni di pericolo. Al fine di valutare l'efficacia dei rilevatori di eventi audio, si è cercato di simulare un'applicazione reale dei modelli, come se un sistema di audio sorveglianza stesse analizzando in tempo reale il segnale audio proveniente da un sensore acustico. Quindi, sono state valutate e confrontate le performance della rete neurale con quelle del SVM, selezionando istanze dai file audio con una finestra mobile dalle tracce audio del testing set.

La lunghezza della funzione finestra è stata fissata a monte e mantenuta invariata per tutto lo svolgimento del progetto: 3 secondi per le istanze della rete neurale, $500ms$ per le istanze dell'SVM.

Nei paragrafi successivi verrà presentato il dataset, discussa la costruzione delle feature, l'allenamento dei modelli, il confronto tra le rappresentazioni tempo-frequenza, l'implementazione del detector.

## 5.2 Dataset

Le applicazioni di computer vision, classificazione d'immagini e di videosorveglianza, trovano una facilità di implementazione maggiore in quanto è possibile trovare dataset grandi e robusti. Si può pensare ad esempio alla

sfida di classificazione d'immagini *ILSVRC*, la quale fornisce un dataset di oltre 14 milioni d'immagini per un totale di 1000 classi [55]. Similmente, per applicazioni audio come lo speech recognition, svariati dataset pubblici sono presenti sul web.

Ciò non esiste per la particolare applicazione di sorveglianza basata su eventi audio. Nonostante i tentavi di sensibilizzazione al problema, come ad esempio le sfide annuali organizzate dalla comunità *DCASE*, dataset consistenti sono in numero limitato.

La scelta è ricaduta quindi sul dataset *Mivia Audio Events Dataset*. Il dataset si compone di tre particolari eventi audio: l'infrangersi di vetri, colpi di pistola e urla, *glass* (G), *gunshots* (GS) e *screams* (S), per ognuno dei quali sono presenti $6000$ eventi suddivisi in $4200$ eventi per il training set e i restanti $1800$ per il testing set.

Per rispecchiare nel modo più verosimile possibile quelle che sono situazioni reali in applicazioni di sorveglianza, gli eventi d'interesse sono stati acquisiti a differenti distanze dal microfono, ciò corrisponde a differenti livelli di *Signal Noise Ratio* (SNR). Detto ciò, il dataset fornisce ogni clip audio a sei diversi livelli di SNR, vale a dire a $5dB, 10dB, 15dB, 20dB, 25dB$ e $30dB$. I livelli di SNR sono ottenuti considerando tracce di background con una potenza costante a cui sono stati sovraimposti gli eventi di interesse con potenza crescente in base al livello di SNR desiderato. Secondariamente, se si considerano i luoghi in cui potrebbe essere necessaria una sorveglianza audio, un fattore comune a tutti questi è sicuramente la componente di background con cui gli eventi d'interesse sono miscelati. I suoni tipici di background possono essere svariati, per questo motivo gli eventi audio sono stati uniti assieme a diversi suoni contestuali, quali: pioggia, fischi, applausi, campane, telefono che squilla, chiacchiericcio, veicoli che passano, ingorgo stradale e rumore bianco.

Le clip audio sono fornite in formato *wave*, lunghe circa 3 minuti, sono state registrate con un modulo Axis P8221Audio in combinazione con un microfono omnidirezionale Axis T83, adatto per applicazioni di sorveglianza audio, campionate con un sample rate di $32000\ sample/sec$ e quantizzate a $16bit$ in *Pulse Code Modulation* (PCM). Per ogni clip audio è associato un file *.xml*, che contiene i metadati degli eventi all'interno del file audio, in particolare i secondi di inizio e fine di ogni evento e le label associate per identificare la classe di appartenenza.

Il dataset si compone quindi di un totale di quasi 30 ore di registrazioni, di 396 clip audio per il training set e di 184 clip audio per il testing set. Le tracce risultano decisamente difficili da affrontare, vista la somiglianza di alcuni suoni di background con gli eventi d'interesse [16].

Composizione del dataset Mivia per numero di eventi e durata in secondi:

| **Classe** | **Training set** | | **Testing set** | |
|---|---|---|---|---|
| | Eventi | Durata (s) | Eventi | Durata (s) |
| **Background (BG)** | / | 58 372 | / | 25 037 |
| **Glass (G)** | 4200 | 6 025 | 1800 | 2 562 |
| **Gunshot (GS)** | 4200 | 1 884 | 1800 | 744 |
| **Screams (S)** | 4200 | 5 489 | 1800 | 2 445 |

*Tabella 6. Riassunto dell numero di eventi d'interesse e della loro durata del dataset Mivia, suddivisi in trainig set e testing set.*

## 5.3 Costruzione delle feature

Come passo iniziale, alle tracce audio è stato sottratto il valore medio e successivamente normalizzate nell'intervallo 0 e 1.

### 5.3.1 Neural Network

Tramite il file .xml sono state estratte le informazioni riguardanti i secondi di inizio, di fine e i nomi degli eventi presenti in ogni file audio. Con questi sono stati selezionati ed estratti gli eventi audio d'interesse da ogni file, e con essi, sono state estratte anche delle sezioni di background di lunghezza pari a 3 secondi al fine di costruire la classe di background. Ciò è stato possibile in quanto tra evento ed evento vi è uno spazio temporale maggiore di 3 secondi.

Nonostante la robustezza del dataset, e l'impiego di una rete convoluzionale con una testa piramidale completamente connessa, la quale aiuta a generalizzare nel caso di dataset ristretti, si è deciso comunque di effettuare *data augmentation* con lo scopo di ampliare il dataset.

Data augmentation si effettua soprattutto nel caso di allenamento di reti neurali, con lo scopo principale di evitare *overfitting*, ovvero evitare che il modello si specializzi troppo a riconoscere una determinata classe in fase di allenamento per poi non essere abbastanza capace di classificare correttamente eventi simili in fase di testing. In altre parole, si rischia di perdere di generalizzazione. Per effettuare data augmentation si è scelto di generare in maniera casuale i parametri necessari, considerando una distribuzione uniforme in un intervallo di valori fissato. Sono state considerate le seguenti modifiche:

- *Time Shift*: gli eventi audio sono stati ritardati temporalmente con valori di shift compresi tra 0 e 0.125 secondi, considerando che l'evento più corto è un colpo di pistola di lunghezza pari a 0.203 secondi.
- *Time Stretch*: gli eventi audio sono stati allungati o accorciati con valori di stretch compresi tra 0.75 e 1.25. Per i valori maggiori di 1 gli eventi audio saranno velocizzati per un massimo del 25%, viceversa rallentati per un massimo del 25%.

- *Pitch Shift*: se si identifica come nota o pitch la frequenza dell'armonica fondamentale di un certo evento audio, si indicherà conseguentemente il suo shift come la variazione di una certa quantità chiamata step. Con una variazione di $+12$ step si ottiene la stessa nota ma di una ottava superiore, oppure uno step equivale ad un semitono. Per il parametro step si è fissato un range di valori da $-2$ a $+2$.

In questo modo è stato possibile ampliare l'intero dataset di quattro volte, ottenendo un training set di 61836 istanze e un testing set di 26520 istanze.

Successivamente, si è reso necessario fissare la lunghezza delle istanze temporali pari a 3 secondi, con lo scopo futuro di ottenere le dimensioni delle rappresentazioni tempo-frequenza uguali. Questa condizione è imposta dalla rete neurale, alla quale occorrono input di dimensione fissata. Mediante la classe background è stato possibile effettuare un pad degli eventi audio nel caso in cui quest'ultimi avessero una durata inferiore a 3 secondi, ovvero minore della durata della finestra mobile. In caso contrario, l'evento è stato troncato a 3 secondi, considerando di mantenere il transiente iniziale dell'evento e scartare l'eccedenza. Si è scelto di costruire le istanze temporali in due configurazioni: la prima è stata costruita garantendo che l'evento sia centrato nella finestra temporale di 3 secondi, implementando una situazione ideale, mentre la seconda è stata costruita in modo che l'evento audio si trovasse in una posizione casuale all'interno della finestra (indicata con *ranpad*), al fine di simulare una situazione più realistica dove le istanze sono estratte da una finestra mobile e non vi è modo di controllare come gli eventi possano posizionarsi all'interno di una finestra. La prima configurazione ha avuto lo scopo di effettuare una prova di fattibilità del presente rilevatore di eventi audio e della possibilità di considerare la seconda configurazione. Inoltre, la costruzione di istanze con questa modalità ha consentito una maggiore velocità della fase di estrazione delle feature, di poter controllare meglio il bilanciamento del dataset in termini di occorrenze delle quattro classi e una minore occupazione di memoria, rispetto all'estrazione di istanze di una finestra mobile sui singoli stream audio.

A questo punto, le istanze temporali sono state trasformate in rappresentazioni tempo-frequenza ad iniziare dallo spettrogramma, per il quale è stata scelta una funzione finestra di Hamming, lunghezza pari a $1024$ campioni ($32ms$) e overlap del $50\%$, lunghezza della trasformata di Fourier pari a 2048 campioni.

Gli spettrogrammi ottenuti, di dimensione pari a $1025x186x1$ pixel, sono stati poi ridimensionati di un fattore 5 per la prima dimensione e di un fattore 3 per la seconda, ottenendo immagini di dimensione $205x62x1$, dove l'ultima dimensione indica il numero di canali colore. Questo passaggio è stato effettuato tramite una funzione di *Maxpooling*, la quale considerando blocchi di grandezza pari a $5x3$ seleziona solamente il pixel con il valore più grande. Ciò ha consentito di limitare il numero dei parametri del primo layer della rete e di riportarsi ad una dimensione simile a quella del lavoro di riferimento.

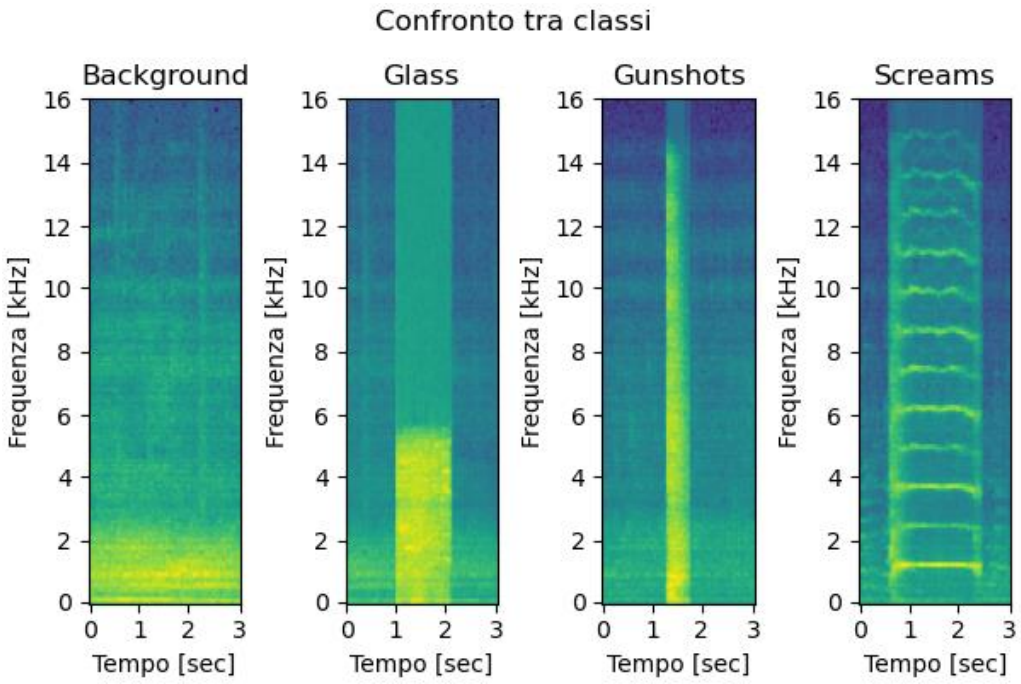


*Figura 14. Spettrogrammi delle quattro classi a confronto. L'asse temporale identifica la durata della finestra. Le immagini sono in scala di grigi, per la visualizzazione è stata applicata una colormap.*

In figura si possono vedere quattro spettrogrammi, uno per ogni classe. L'aspetto interessante da sottolineare che si nota osservando gli spettrogrammi è la distinguibilità degli eventi. Se ne deduce che se le immagini sono

distinguibili ad occhio allora risulta fattibile allenare un modello intelligente che impari a discernere dalle immagini le quattro tipologie di evento.

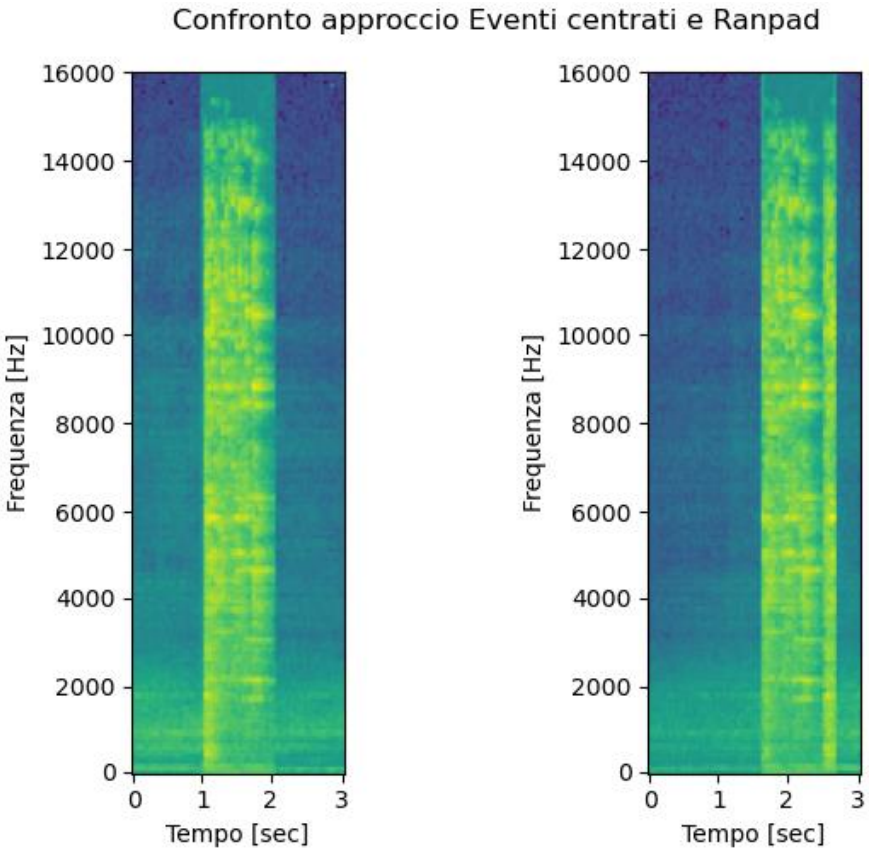


*Figura 15. Confronto tra l'immagine dello stesso evento glass, costruita con l'approccio eventi centrati (a sinistra) e con l'approccio ranpad (a destra).*

Analogamente a quanto appena descritto, sono state costruite successivamente anche le rappresentazioni tramite spettrogramma in scala Mel e gammatonogramma, per le quali, laddove necessario, sono stati impostati gli stessi parametri per funzione finestra, overlap e lunghezza trasformata di Fourier. Anche per questi due casi sono state ottenute immagini di dimensioni pari a $205x62x1$ (figura 16).

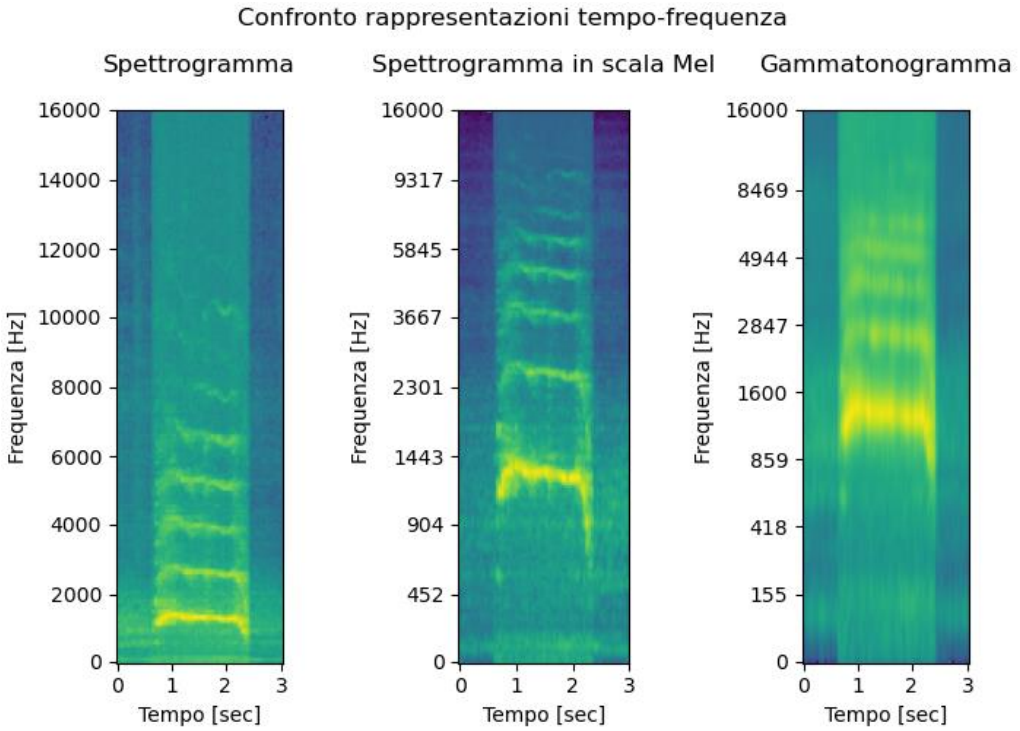


*Figura 16. Confronto delle tre rappresentazioni tempo-frequenza. Da sinistra: spettrogramma, spettrogramma in scala Mel e gammatonogramma.*

L'ultimo passaggio è stato quello di normalizzare il range di valori delle immagini tra 0 e 1 e trasporle ottenendo dimensioni pari a $62x205x1$.

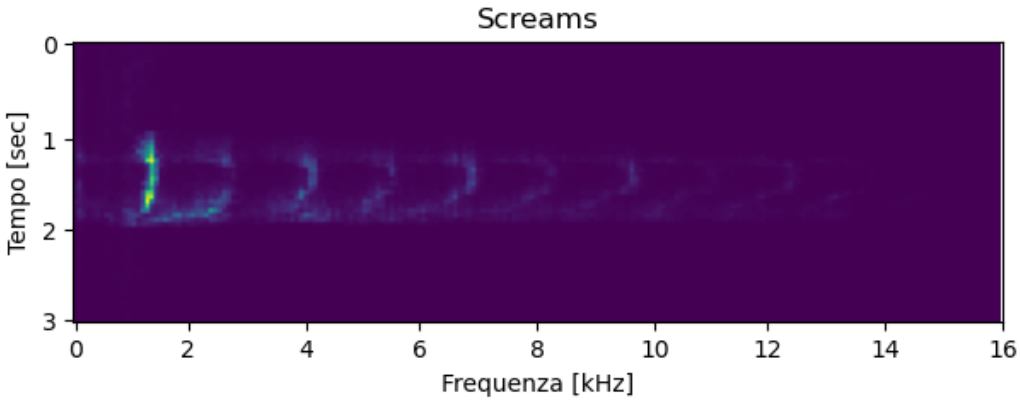


*Figura 17. Immagine trasposta di un evento screams con valori normalizzati tra 0 e 1. Titolo d'esempio di un'immagine fornita alla rete neurale per l'allenamento.*

### 5.3.2 SVM

La costruzione delle feature per il SVM è avvenuta secondo due criteri.
Nel primo sono state estratte le feature MFCC dai singoli eventi audio. Si è considerato quindi lo stesso insieme degli eventi audio estratti per la costruzione delle feature tempo-frequenza. Per estrarre i coefficienti si è considerata una finestra mobile di 2048 campioni, con overlap del 75%, a cui vengono applicati i passaggi necessari per l'estrazione delle feature MFCC. Si ottiene quindi una rappresentazione bidimensionale, per cui una dimensione è fissata dal numero di coefficienti che s'intende considerare (numero di vie del banco di filtri), in questo caso 26, l'altra dipende dal numero di istanze estraibili dall'evento considerato, ed è stimabile come:

$$\frac{N}{(1 - overlap/100) * w_{length}}$$

ove $N$ è il numero di campioni dell'evento considerato, $overlap$ è la sovrapposizione tra finestre espressa in percentuale e $w_{length}$ è la lunghezza della finestra in numero di campioni. Tuttavia, siccome si necessita ottenere un vettore monodimensionale di coefficienti si è deciso di mediare i coefficienti lungo la dimensione temporale.

Per il secondo criterio di estrazione delle feature è stata considerata una funzione finestra mobile di $500ms$ con overlap nullo, con la quale estrarre istanze direttamente dallo stream audio. In questo caso, è stato indispensabile definire una regola per l'assegnazione delle label, in quanto un evento d'interesse può occupare la finestra completamente o in parte. Per le feature utili al fine dell'allenamento del modello quindi, si è scelto di assegnare il nome di un determinato evento ad una istanza se e solo se occupa completamente la finestra, altrimenti viene considerata background. Per identificare il riempimento della finestra è stato definito un parametro *fill*. Esso indica la percentuale di riempimento di una finestra che un evento d'interesse deve raggiungere per poter assegnare la label a quell'istanza. Per l'appunto, in

questo caso è stato fissato a 1. L'insieme di queste feature è stato costruito considerando solamente un terzo dei file audio per ogni livello di SNR, questo perché il SVM non necessità di un training set cospicuo tanto quanto per le reti neurali. L'ultima considerazione per questo criterio è la seguente, tramite questa modalità di estrazione delle feature si generano tantissime istanze di background sbilanciando il dataset, quindi è stato obbligatorio controllare il numero di istanze di questa classe in tempo reale per avere un dataset il più equilibrato possibile al termine dell'estrazione delle feature. Inoltre, questi ultimi due accorgimenti hanno concesso di ridurre il tempo utile per la costruzione dell'insieme di feature, poiché gli step necessari per gli MFCC sono alquanto dispendiosi computazionalmente.

## 5.4 Training della Rete Neurale

### 5.4.1 Setting dei parametri e training preliminari

L'allenamento di una rete neurale è una fase della costruzione di un modello molto importante, e richiede numerosi tentativi, al fine di capire quale combinazione di parametri consente di ottenere le migliori performance e una buona dose di potenza di calcolo, soprattutto quando si cerca di allenare reti molto profonde. Nella rete implementata nel presente progetto si contano un numero totale di parametri allenabili pari a 99.785.612.
I dati da fornire alla rete per l'allenamento sono stati organizzati in due vettori, il primo indicato con *imageTrain* e contiene 61836 immagini per l'allenamento, il secondo, *NameTrain*, contiene i nomi delle immagini e di lunghezza pari a imageTrain. Mentre i dati per la fase di testing sono stati organizzati in *imageTest* e *NameTest*. Questi due vettori sono composti a loro volta da sei vettori i quali contengono le immagini e le label divise per valori di SNR. Ogni vettore contiene 4420 elementi, nello specifico 1200 per le classi d'interesse e 820 per quella di background. Mediante questa soluzione sarà possibile valutare le performance della rete in base al livello dell'SNR.
I vettori delle immagini di training e testing hanno la seguente dimensione: $(N° \; immagini, 62, 205, 1)$.

I due vettori di stringhe NameTrain e NameTest sono stati trasformati in vettori di interi chiamati *LabelTrain* e *LabelTest* rispettivamente, assegnando ai nomi delle classi un indice. Per la classe background l'indice assegnato è 0, per glass 1, per gunshot 2, infine per screams 3.
Il secondo passaggio è stato suddividere imageTrain e LabelTrain in due porzioni, la prima *trainingImage* e *trainingLabel*, la seconda *validationSet*.
Il validationSet è una tupla composta dal 10% di imageTrain e LabelTrain.

La base temporale dell'allenamento di una rete neurale è scandita dalle epoche.
In un'epoca la rete vede come ingresso tutte le immagini contenute in trainingImage e trainingLabel e alla fine valuta le prestazioni attraverso il validationSet, in questo caso il valore della loss e dell'accuratezza.
Ad ogni epoca, trainingImage e trainingLabel vengono mescolati in modo casuale senza perdere il riferimento tra gli elementi dei due vettori, in modo da aumentare la generalizzazione dell'apprendimento.
Per l'allenamento di una rete neurale si necessita definire una serie di parametri, tra cui i sopra citati learning rate, batch size, funzione di *loss* e l'algoritmo tramite il quale calcolare il valore di aggiornamento dei parametri, per garantire il raggiungimento del migliore minimo locale.
Inizialmente per SoReNet sono stati scelti i seguenti parametri di default:

| **Learning rate** | **Batch size** | **Funzione costo** | **Algoritmo di ottimizzazione** |
|---|---|---|---|
| $\boldsymbol{lr = 0.001}$ | $BS = 64$ | Sparse Categorical Cross Entropy | Adam |

*Tabella 7. Insieme dei paramentri per la compilazione della rete neurale.*

Un secondo set di parametri utilizzato sono le *callbacks*. Nello specifico esse sono delle funzioni le quali interagiscono in maniera autonoma con la rete durante l'allenamento, e necessitano di essere inizializzate. Ne esistono diverse, quelle utilizzate sono riassunte di seguito

| | |
|---|---|
| **TerminateOnNan** | Consente di terminare l'allenamento nel caso un valore Nan della funzione costo venisse calcolato |
| **ModelCheckpoint** | Consente di salvare il modello della rete o solamente il valore dei pesi con una certa frequenza in base al miglior valore di una determinata metrica |
| Parametri | metric: una metrica tra *validation loss, loss, accuracy* o *validation accuracy*<br>frequenza salvataggio: ogni quante epoche salvare |
| **ReduceLROnPlateau** | Riduce automaticamente il valore del learning rate nel caso una determinata metrica rimane costante o inverte direzione |
| Parametri | metric<br>factor: fattore diminuzione percentuale del learning rate<br>patience: numero di epoche da attendere prima della modifica<br>Min lr: minimo valore permesso |
| **EarlyStopping** | Termina l'allenamento nel caso una determinata metrica smette di migliorare |
| Parametri | metric<br>patience<br>min delta: minima variazione metrica quantificabile come miglioramento |
| **Tensorboard** | Consente di aprire una dashboard nella quale è possibile visualizzare l'andamento delle metriche scelte per il monitoraggio dell'allenamento. In questo caso: *accuracy*, *loss*, *validation accuracy* e *validation loss* e *learning rate* |

*Tabella 8. Insieme dei paramentri per l'inizlizzazione delle callback*

Durante il training, ogni volta che la rete prende in ingresso un numero di immagini pari al batch size, il valore della loss e dell'accuracy viene calcolato e mostrato a video. In aggiunta, alla fine di ogni epoca il validationSet viene utilizzato per calcolare il valore della *validation loss* e della *validation accuracy*. Queste sono due metriche disgiunte dalla loss e dall'accuracy calcolate in fase di training. Gli elementi contenuti nel validationSet non

vengono mai utilizzati per allenare la rete, ma esclusivamente per valutare le prestazioni durante l'allenamento. È una soluzione molto intelligente per controllare se la rete tende ad andare in overfitting.
Mediante la callbacks Tensorboard è possibile monitorare queste metriche, ma anche la riduzione del learning rate dovuta alla callabacks associata.

Prima di iniziare la fase di allenamento è stata effettuata una valutazione dell'accuracy della rete non allenata sia con trainingImage, sia con validaitonSet, al fine di verificare che non ci fosse una discrepanza iniziale, aspettandosi circa il $25\%$ di accuracy, in quanto il numero di classi è $4$. Il riferimento temporale di questa valutazione è stato identificato come epoca $0$.

Il valore del learning rate è stato il protagonista nel setting corretto dei parametri. Il valore indicato poc'anzi in tabella è generalmente il valore di default da cui partire. Tuttavia, le metriche di validation loss e validation accuracy non hanno accennato a nessun miglioramento, rimanendo costanti al valore di partenza per tutte le epoche successive. Non notando nessun miglioramento delle performance, istintivamente si è scelto un learning rate più alto immaginando che la rete avesse bisogno di aggiornare i parametri con quantità maggiori. Sfortunatamente, sono state necessarie numerose prove per capire di dover diminuire il learning rate, addirittura fino ad un valore pari a $10^{-5}$.
Successivi allenamenti di setting del batch size sono stati condotti, precisamente con valori di $32$ e $128$. Tuttavia, il valore iniziale si è rivelato il valore giusto per un compromesso tra velocità e stabilità di convergenza.
Le prove appena descritte sono state effettuate con trainingImage ridotto, contenente solamente spettrogrammi riferiti ad un SNR di $30dB$ e con gli eventi centrati nelle istanze.

Globalmente, lungo tutto lo studio della migliore configurazione di allenamento sono state considerate tre strategie. La prima consiste nell'istanziare il modello e inizializzare i parametri in maniera casuale e

allenarlo completamente da zero, indicata come *training from scratch*. La seconda è quella conosciuta come *transfer learning*, la quale consiste nell'inizializzare la rete tramite dei pesi provenienti dall'allenamento di una rete simile su un task analogo. Nel caso corrente, sono stati utilizzati i pesi della VGG19 allenata sul dataset di ImageNet. In questo caso, sono stati trasferiti i pesi di tutti i layer tranne per quello convoluzionale d'ingresso (poiché esso dipende dalla dimensione delle immagini e ImageNet fornisce immagini di $224x224x3$) e i layer fully connected in quanto essi sono per numero di layer e numero di neuroni diversi. La terza strategia risulta in un training spezzato, ovvero si allena la rete in più fasi cambiando lo stato di alcuni layer consentendo o meno di poter aggiornare i loro parametri, quindi rendendoli allenabili o non allenabili. Con maggior precisione si caricano i pesi di ImageNet, si effettua un primo allenamento con lo strato fully connected non allenabile, un allenamento successivo con i blocchi convoluzionali non allenabili, e un terzo ed ultimo allenamento della rete complessiva conosciuto anche come *fine tuning*. Si indicherà questa seconda strategia con *step learning*.

Il dataset ridotto è stato poi utilizzato per condurre altri allenamenti relativi alle tre strategie appena introdotte, con lo scopo di conferire validità alla scelta. In particolare, il primo allenamento si riferisce al training from scratch, il secondo al transfer learning, il terzo allo step learning (figure 18, 19 e 20 rispettivamente). In realtà, di quest'ultimo è stata condotta solamente la seconda fase, per la quale si allena solo la parte fully connected della rete, allo scopo di capire se questa strada fosse percorribile.
Qui di seguito sono riportati gli andamenti dell'accuracy e della loss sia per la fase di training, sia per quella di validation.

La curva verde di train rappresenta l'ultimo valore raggiunto dalla metrica di riferimento nel momento in cui la rete ha concluso di scorrere il trainingSet, mentre la curva blu di validation rappresenta il valore della metrica di riferimento calcolato con il validationSet.

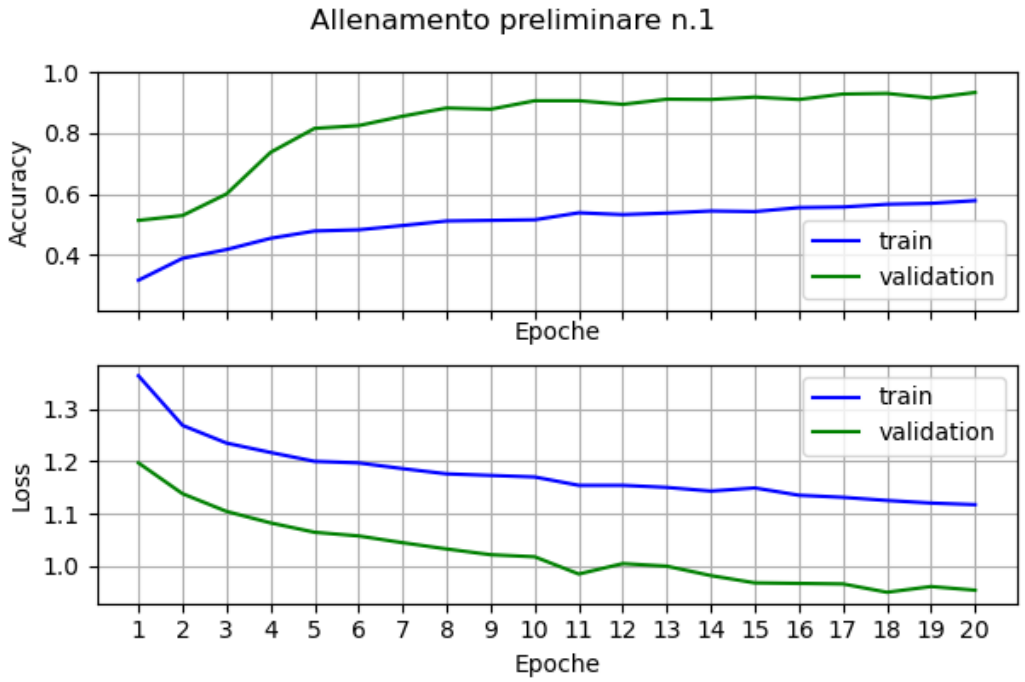


*Figura 18. Strategia training from scratch.*

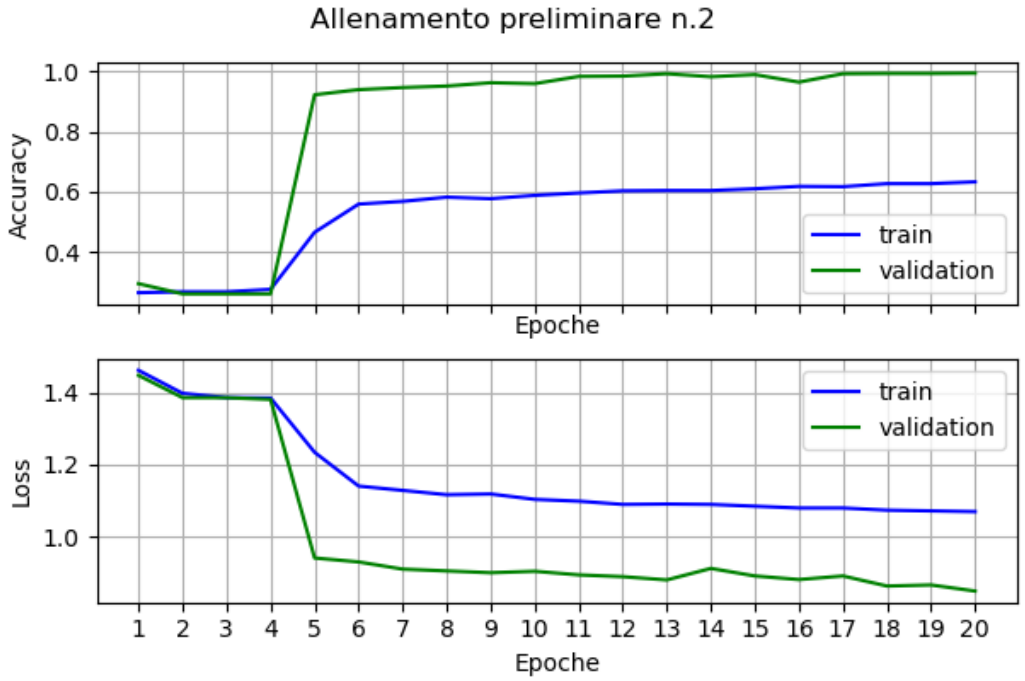


*Figura 19. Strategia transfer learning*

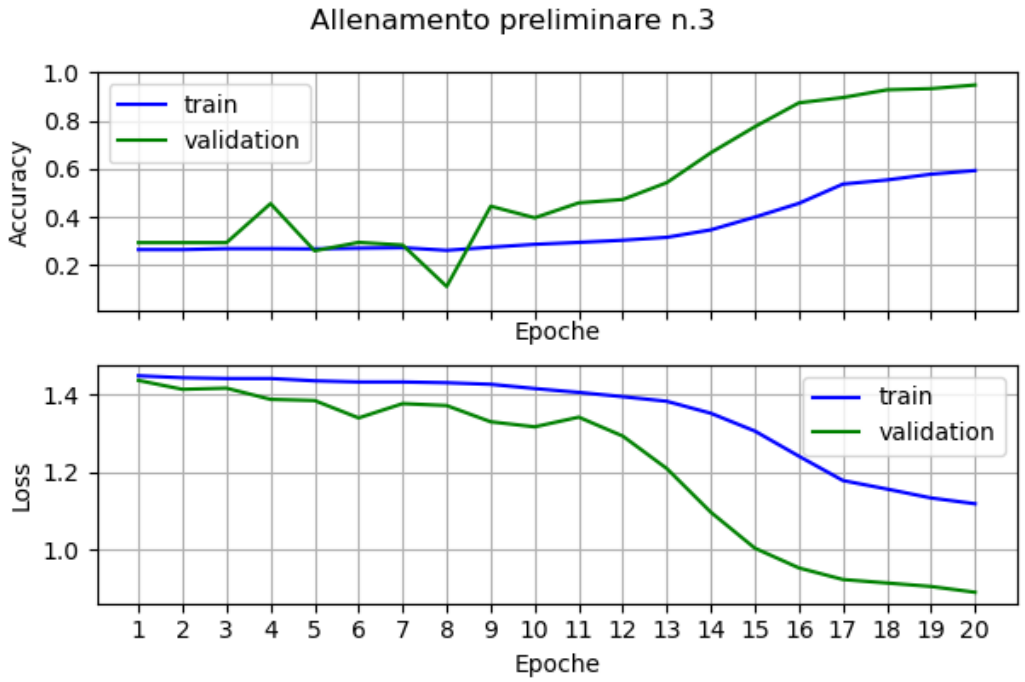


*Figura 20. Strategia step learning, in particolare solo la seconda fase è stata condotta in questo allenamento.*

Come si può notare dagli andamenti delle metriche in quesitone, esse presentano l'andamento che ci si aspetta, ovvero che l'accuracy tenda al valore massimo 1 e che la loss presenti un andamento complessivamente decrescente. Ciò nonostante, gli allenamenti considerati in questo frangente non possono considerarsi completi in quanto il dataset non è sfruttato nella sua interezza e dall'andamento della loss si intuisce che possa migliorare ulteriormente poiché in corrispondenza delle ultimissime epoche si può notare un andamento complessivamente decrescente. Infatti, allenamenti di reti di questo calibro necessitano un numero di epoche di apprendimento tendenzialmente attorno alle centinaia.

L'unico comportamento inaspettato è il gap presente tra le curve di train e validation. Inaspettato perché solitamente ci si aspetta che le curve presentino circa gli stessi valori con la curva del validation leggermente inferiore a quella di train. Questo perché i dati del validationSet non sono mai utilizzati per aggiornare i parametri, si dice che per la rete sono "dati mai visti", per cui è giustificabile una lieve difficoltà in più nel classificarli.

Il motivo legato alla presenza del gap è dovuto ai layer di *dropout* posizionati di seguito ad ogni layer fully connected, per un totale di cinque.
Essi sono dei layer fittizi, in quanto non presentano alcun parametro da allenare, ma svolgono l'importante funzione di prevenire che la rete possa andare in overfitting. Il loro funzionamento è molto semplice, ad ogni aggiornamento dei parametri i layer di dropout spengono casualmente una percentuale (*rate*) di neuroni del layer fully connected posizionato a monte. Questo artifizio fa sì che solamente la porzione di neuroni che non viene spenta può aggiornare i parametri a loro associati, evitando maggiormente che la rete si specializzi a riconoscere esclusivamente i dati contenuti nel trainingSet. I layer di dropout però non entrano in funzione durante la fase di validation, consentendo a tuti i neuroni di contribuire al calcolo della probabilità di uscita. Questo spiega la presenza del gap tra la curva di trainig e validation. Gli allenamenti fin qui mostrati, sono stati condotti con un rate di dropout pari a 0.5. Il valore è stato inizializzato come il valore utilizzato per la VGG19 nel task di classificazione di ILSVRC, anche se è in genere un valore molto alto, ma giustificabile in quanto la sfida propone di classificare 1000 classi distinte, un numero che causa un maggior rischio di overfitting.
Per dimostrare questo comportamento sono stati condotti alcuni training con la modalità training from scratch al con rate di dropout prima uguale a 0, poi a 0.2

Dai grafici illustrati nelle figure 21 e 22, è possibile notare come per un rate di dropout nullo il gap tra gli andamenti è praticamente estinto, mentre si presenta per valori più alti. Nonostante il ridotto numero di epoche di questi allenamenti, nessun accenno di overfitting si è mostrato, ciò si può dedurre dall'andamento della validation loss che continua a scendere senza accenni di aumento. Inoltre, un'ulteriore prova che il gap dipende esclusivamente dall'utilizzo dei layer di dropout è legata al valore di accuracy calcolata con il trainingSet e il validationSet prima di lanciare l'allenamento, in corrispondenza dell'epoca 0, la quale vale circa 24.7% per entrambe.
Per questo motivo per tutti gli allenamenti a venire si è imposto un rate di dropout pari a 0.1.

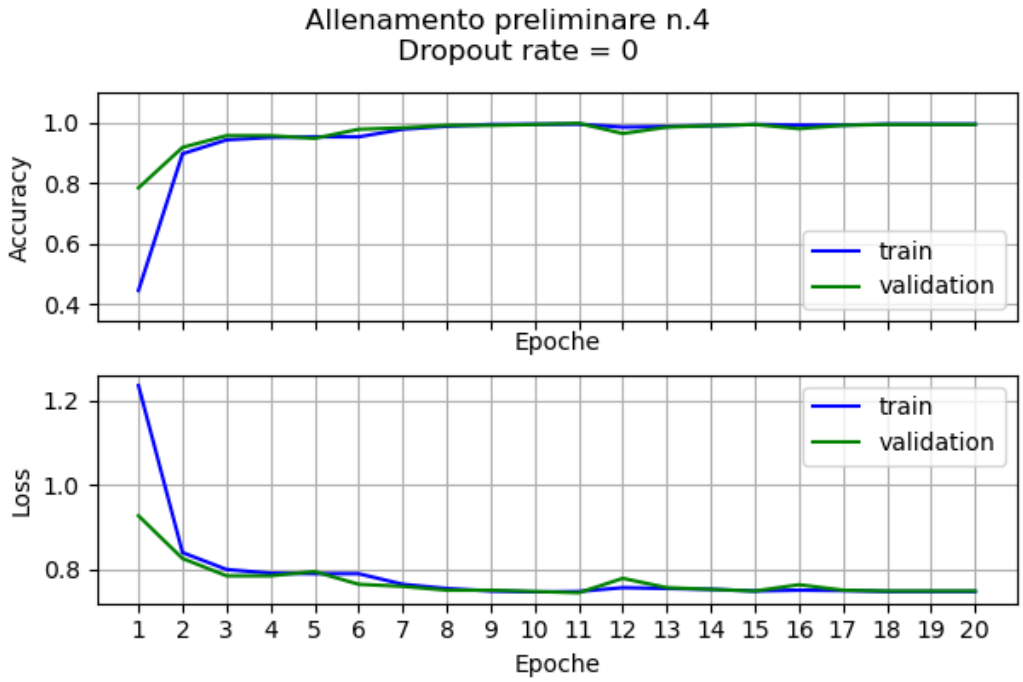


*Figura 21. Test di allenamento con dropout nullo. Il gap tra le curve di train e validation è praticamente nullo.*

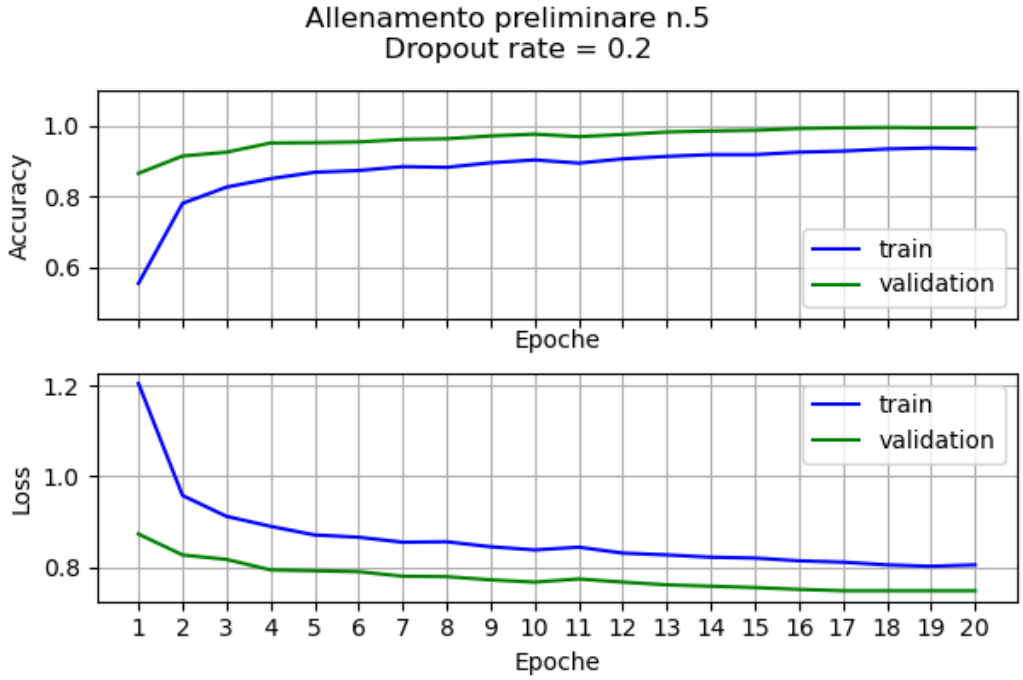


*Figura 22. Allenamento con dropout 0.2. Si evidenzia immediatamente il gap tra le curve di train e validation.*

### 5.4.2 Confronto tra le strategie di allenamento

La fase successiva è stata quella di studiare gli effetti delle tre strategie in allenamento, con lo scopo di scegliere quella che garantisce le migliori prestazioni.
D'ora in poi si considera il trainingSet e il testingSet composti dalla totalità degli elementi, quindi con tutti i livelli di SNR. Gli allenamenti sono condotti ancora una volta con spettrogrammi centrati.
I parametri stabiliti per gli allenamenti sono riassunti nelle tabelle riportate di seguito:

| **Learning rate** | **Batch size** | **Funzione costo** | **Algoritmo di ottimizzazione** |
|---|---|---|---|
| $\boldsymbol{lr = 10^{-5}}$ | $BS = 64$ | Sparse Categorical Cross Entropy | Adam |

*Tabella 9. Insieme dei parametri per la compilazione della rete neurale.*

| | |
|---|---|
| **ModelCheckpoint** | metric: *validation loss*<br>frequenza salvataggio: ogni epoca |
| **ReduceLROnPlateau** | metric**:** *validation loss*<br>factor: 0.5<br>patience: 10<br>Min lr: $10^{-12}$ |
| **EarlyStopping** | Metric: *validation loss*<br>patience: 20<br>min delta: $10^{-7}$ |

*Tabella 10. Insieme dei parametri per l'inizializzazione delle callback..*

A seguire sono riportati gli andamenti delle metriche durante i vari allenamenti, compreso quello del learning rate in risposta all'azione della callback ReduceLROnPlateau. In questo frangente sono stati operati allenamenti maggiormente robusti, in quanto il setting delle callback ha consentito un allenamento più prolungato. In figura 23 training from scratch, in figura 24 transfer learning e nelle tre figure successive, 25, 26 e 27, le tre fasi di allenamento per lo step learning.

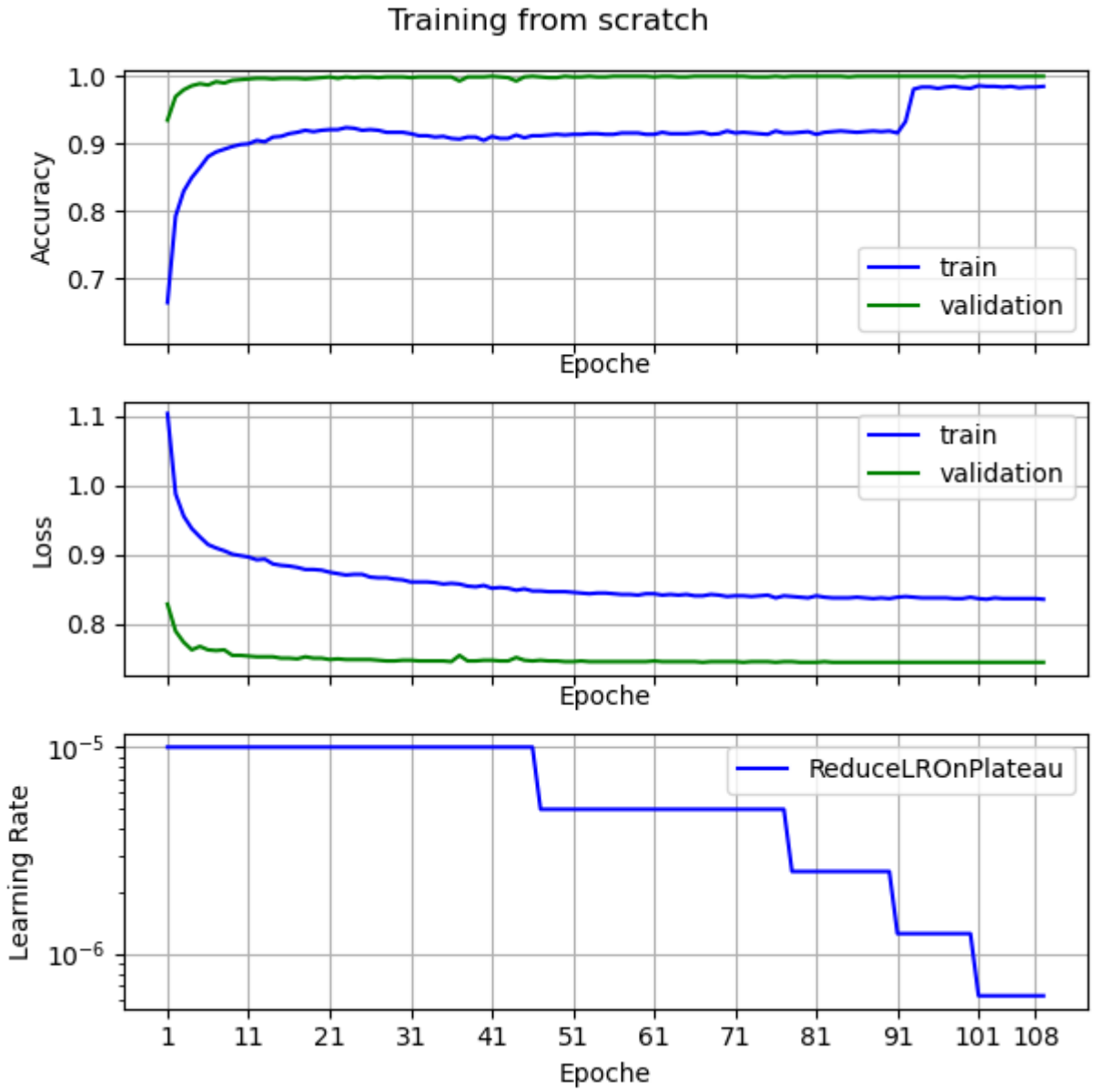


*Figura 23. Allenamento della rete con la strategia training from scratch*

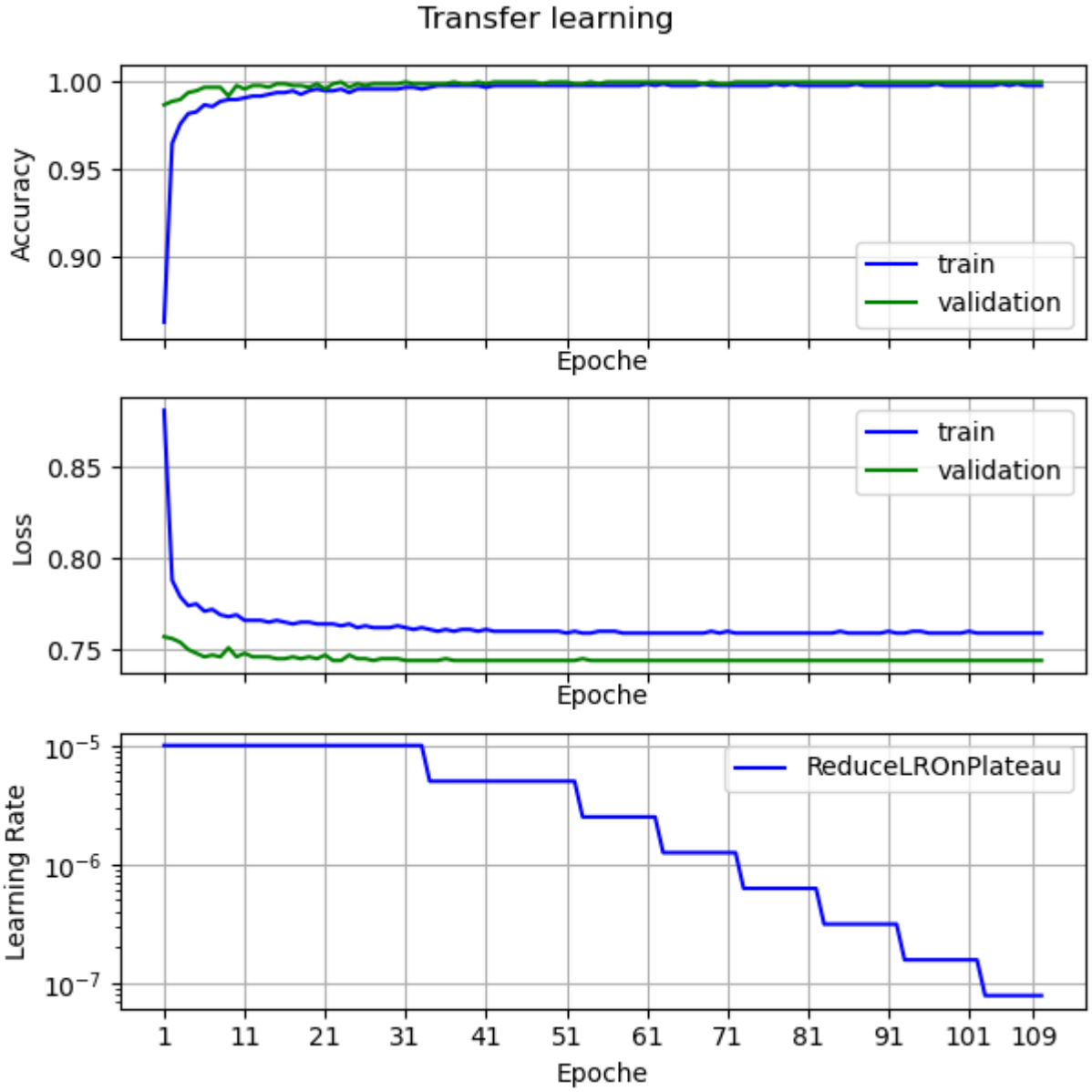


*Figura 24 Allenamento della rete con la strategia transfer learning*

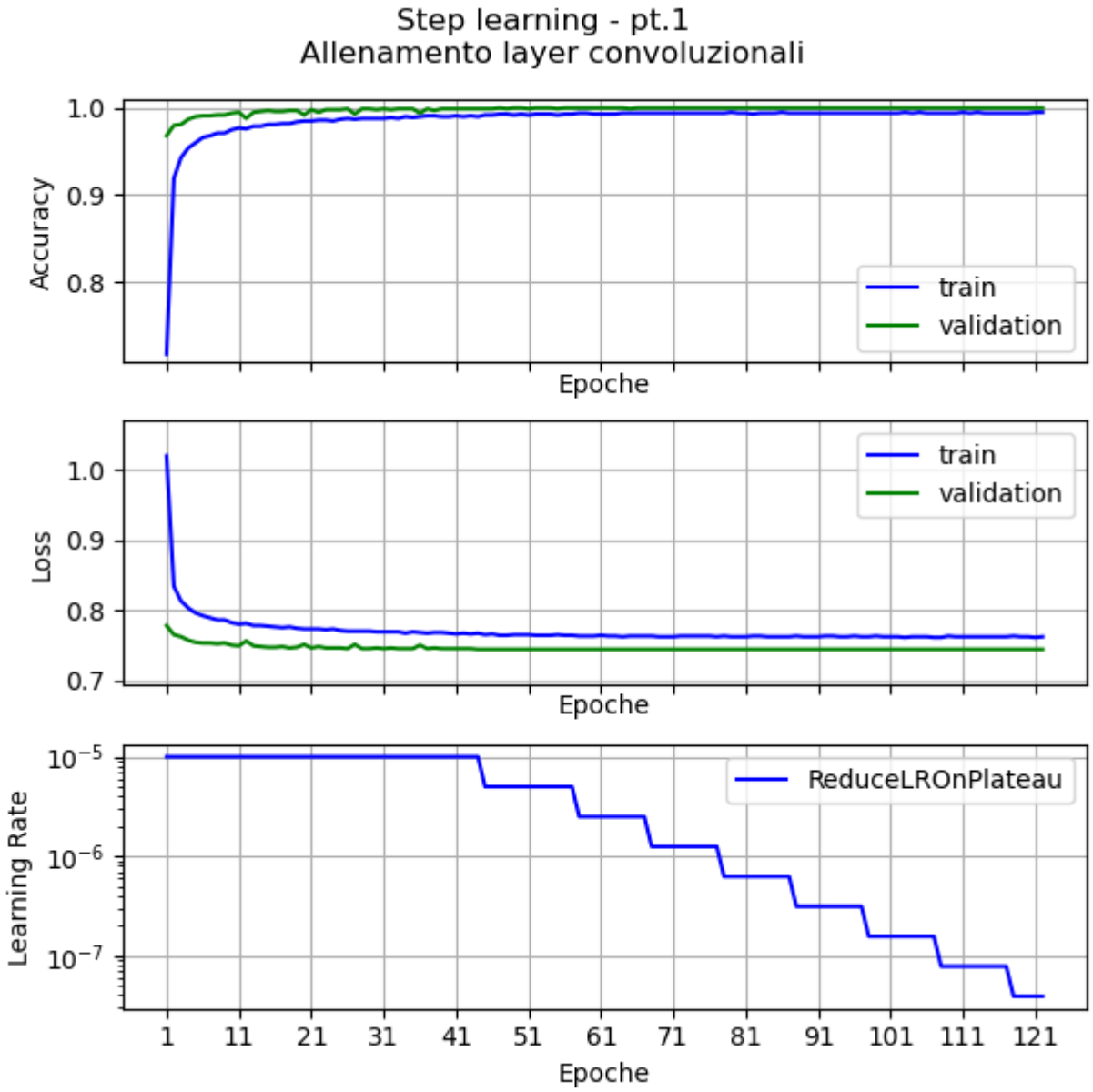


*Figura 25. Allenamento della rete con la strategia step learning fase 1.*

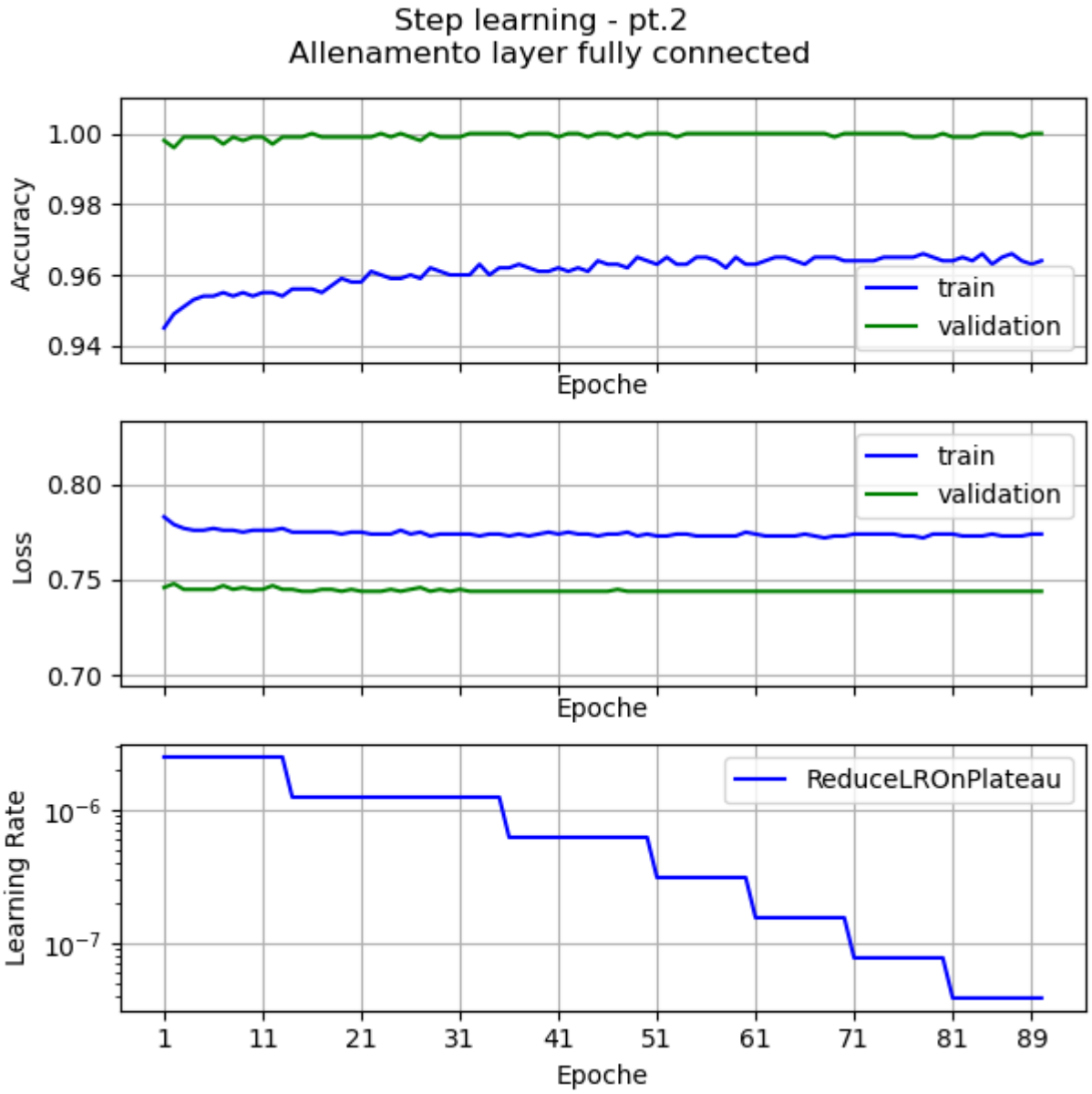


*Figura 26. Allenamento della rete con la strategia step learning fase 2.*

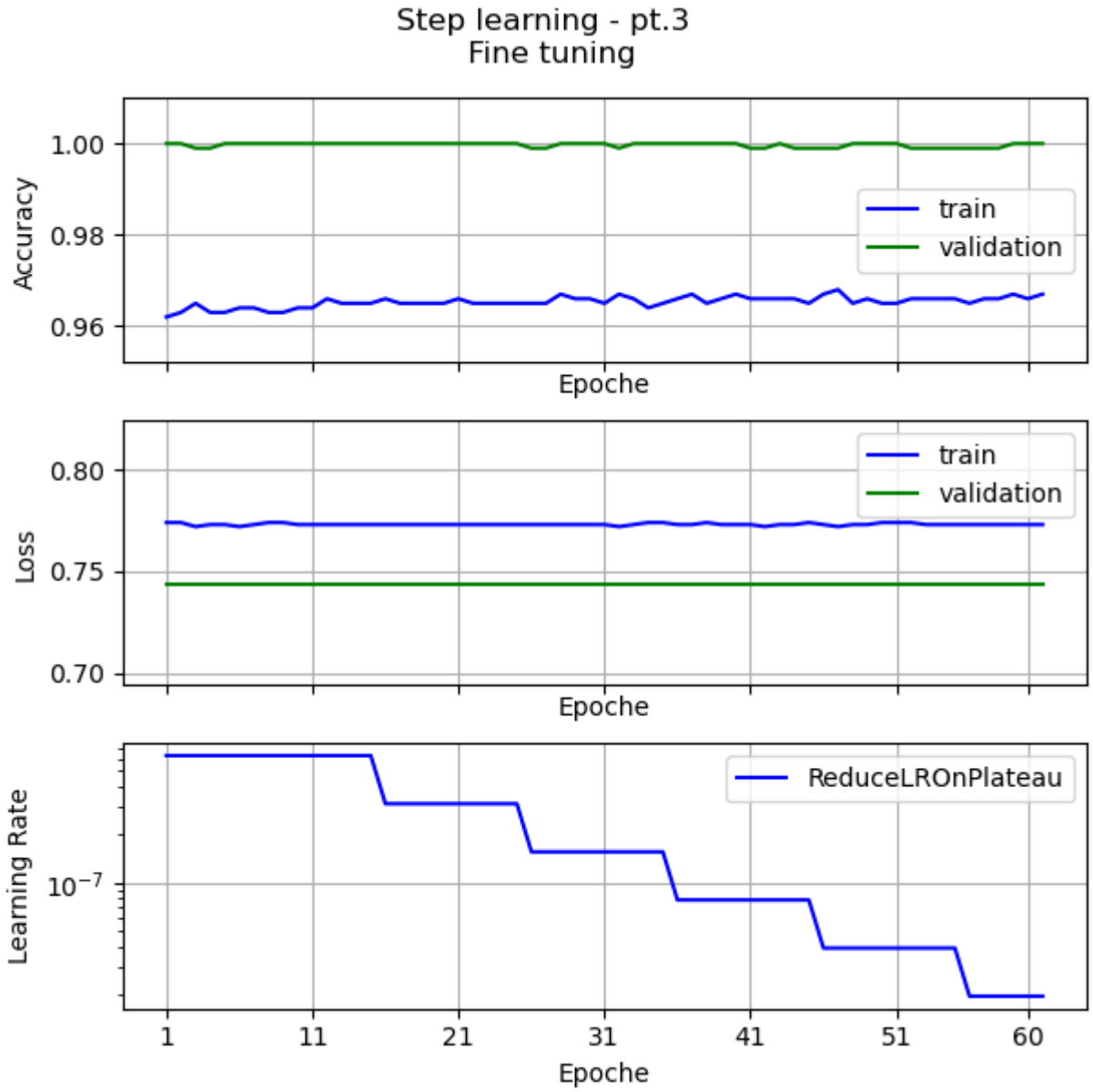


*Figura 27. Allenamento della rete con la strategia step learning fase 3.*

Negli andamenti delle metriche appena riportati, si può dire che la rete è stata allenata correttamente. Questo è possibile dirlo perché per ogni strategia al termine dell'allenamento si è raggiunto un valore di accuracy molto alto, approssimabile ad 1. Allo stesso modo anche la funzione di loss dimostra il raggiungimento di un allenamento ottimale, poiché la curva presenta una pendenza pressoché nulla nell'intorno delle epoche finali. Si può affermare per cui di essersi stabilizzati in un minimo locale della funzione loss, ciò è sicuramente consolidato dall'andamento del learning rate, il quale ha subito in

ogni strategia diversi dimezzamenti. Man mano che la funzione di loss si avvicina al minimo locale è necessario garantire una diminuzione del learning rate graduale, per evitare di compiere passi di aggiornamento della loss troppo grandi, rischiando quindi di saltare il minimo senza raggiungerlo.
L'aggiornamento del learning rate in modalità dinamica è sia garantito dall'algoritmo di ottimizzazione Adam, sia dalla callback ReduceLROnPlateau.
Nello specifico, nella prima modalità training from scratch, è possibile notare un miglioramento sostanziale dell'accuracy in fase di training avvenuto in corrispondenza di un dimezzamento del learning rate. Tuttavia, non risulta completamente chiaro il motivo di questa variazione improvvisa la quale non è accompagnata da una diminuzione della loss in corrispondenza della stessa epoca. Ciò nonostante, l'accuracy di validation è rimasta costante ad un valore pressoché unitario.
Nella seconda modalità, l'andamento delle metriche presenta un andamento approssimativamente ideale. Entrambe le accuracy raggiungono l'unità e la funzione loss si stabilizza ad un minimo locale.
Nel caso dello step tuning si notano delle prestazioni analoghe a quelle ottenute nel transfer learning già nel primo step, nonostante solamente i layer convoluzionali siano stati allenati. Il secondo step aiuta a migliorare i valori dell'accuracy in fase di training mentre nell'ultimo step i valori rimangono pressocché costanti durante l'allenamento. Si può osservare che gran parte della minimizzazione della funzione loss avviene all'incirca nelle trenta epoche iniziali del primo step. Per il secondo step si è scelto un valore iniziale del learning rate pari a $2.5\ 10^{-6}$, mentre per l'ultimo step un valore iniziale pari a $6.25\ 10^{-7}$. Non si notano miglioramenti sostanziali negli step successivi, ma unicamente una diminuzione della varianza dei valori, soprattutto per quanto concerne lo step del fine tuning. Un'ulteriore osservazione riguarda il numero di epoche necessarie alle tre fasi dello step learning. Si può notare che ad ogni step il numero di epoche diminuisce progressivamente, questo è causato dall'azione della callback EarlyStopping, la quale interviene quando non si

ottiene una diminuzione minore della validation loss di $10^{-7}$. Questo è sintomo del raggiungimento del minimo locale.

Tuttavia, queste considerazioni non sono in grado di poter assegnare la medaglia d'oro ad una delle tre strategie. Per poter scegliere la migliore è necessario valutare le performance tramite la valutazione di alcune metriche scelte ad hoc utilizzando i dati di ImageTest.
In primis, si sono valutate l'accuracy e gli errori in termini percentuali per ogni livello di SNR e per ogni classe del dataset. A seguire sono presentate tre tabelle per ogni strategia. Per lo step learning sono riportati i risultati solo per la part del fine tuning. La prima colonna rappresenta l'accuracy al variare del livello di SNR. Le colonne successive mostrano gli errori di classificazione di ogni classe per ogni livello di SNR. Di fianco si può vedere la colorbar con cui è stato mappato il range di valori, da tenere presente che il colore giallo corrisponde a valori ottimi per l'accuracy mentre il colore blu corrisponde a valori ottimi per gli errori.

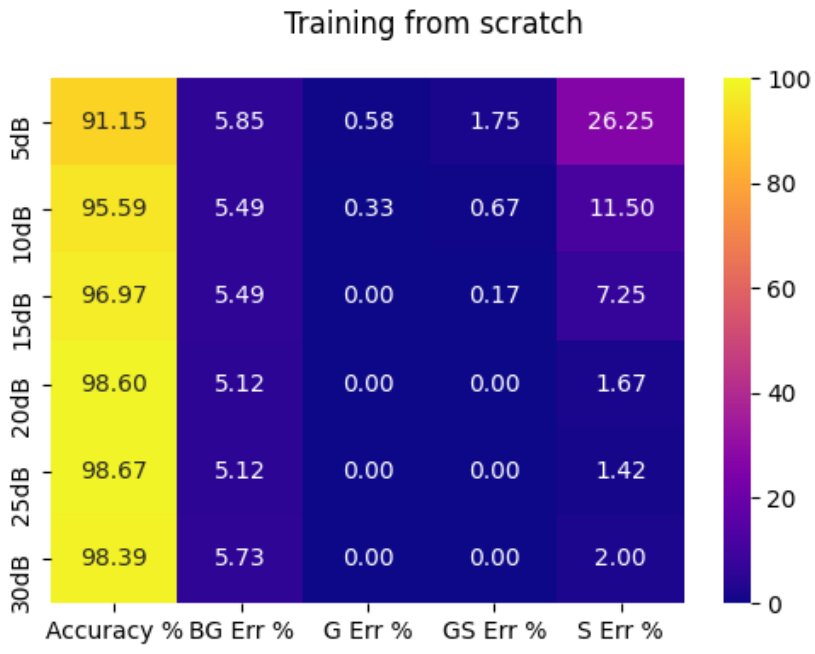


*Figura 28. Tabella dei valori di accuracy e degli errori percentuali per ogni classe e al variare dell'SNR. Strategia training from scratch.*

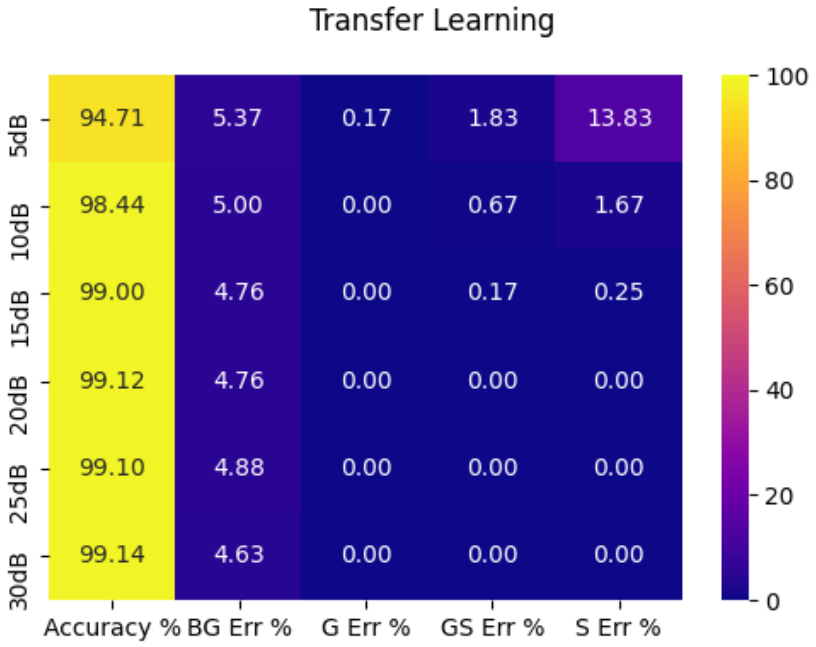


*Figura 29. Tabella dei valori di accuracy e degli errori percentuali per ogni classe e al variare dell'SNR. Strategia transfer learning*

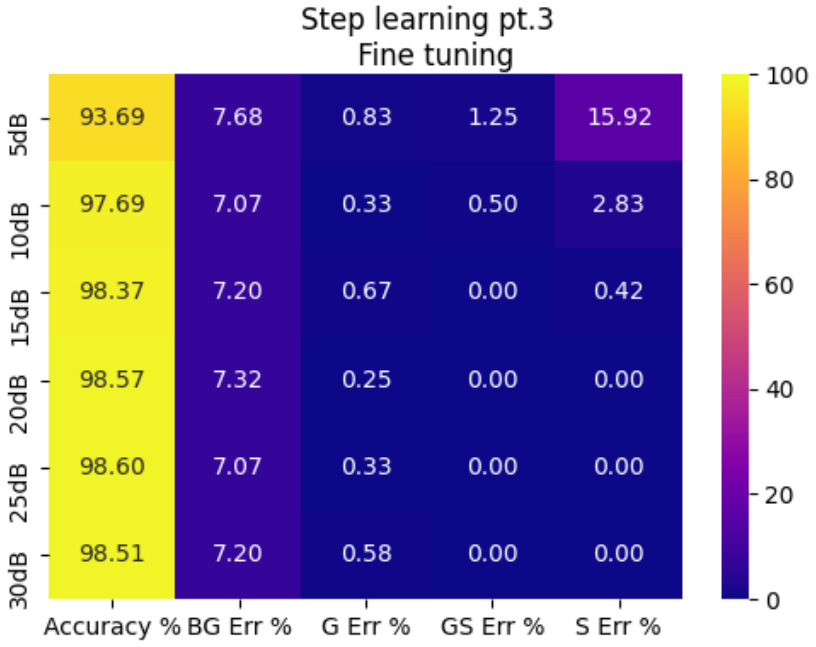


*Figura 30. Tabella dei valori di accuracy e degli errori percentuali per ogni classe e al variare dell'SNR. Strategia step learing fase 3.*

Globalmente si può osservare una notevole robustezza delle tre strategie, ogni approccio presenta valori di accuracy maggiori del 90%, quindi nessuno dei tre è scartabile immediatamente. Tuttavia, attraverso queste immagini è possibile

apprezzare delle differenze tra le strategie. Ovviamente, al diminuire del livello di SNR le prestazioni tendono ad abbassarsi in generale, in quanto alcuni eventi vengono sommersi maggiormente dal background e quindi classificati in maniera errata.
Il transfer learning sembra essere la migliore delle tre, mostrando i più alti valori di accuracy, con il 99% per SNR da $30dB$ a $10dB$. Entrando più nello specifico, si può constatare che questa modalità di allenamento è quella che garantisce maggior robustezza nel riconoscere gli eventi di background, in quanto la percentuale di errore è complessivamente la più bassa delle tre tecniche. Quindi ci si aspetta essere la strategia con il minor tasso di falsi positivi di background. Ovviamente, la percentuale di errore del background al variare dell'SNR rimane circa la stessa in quanto la potenza del rumore è costante nelle tracce audio. Nella classificazione della prima classe d'interesse, ovvero glass, il transfer learning risulta essere ancora una volta il migliore, seguito dallo step learning e a chiudere la fila il training from scratch. Anche se bisogna affermare che tutte tre le modalità si comportano bene nei confronti di questa classe. Continuando nell'analisi si può notare che per la classe gunshots la migliore strategia è lo step learning, e pressocché a parimerito le restanti due, anche se la differenza è quantificabile con qualche punto decimale. Per concludere il giro troviamo la classe screams, e come ci si poteva aspettare considerando le varie tipologie di background presenti, essa risulta la più difficile da classificare. Si può osservare come tutte le strategie facciano tanta fatica a riconoscere correttamente gli eventi screams per valori di SNR bassi come i $5dB$. Il training from scratch è evidentemente la peggiore delle tre, con una percentuale altissima per i $5dB$ e anche per i $30dB$ considerando le percentuali delle strategie rivali.

Grazie a queste tabelle si è in grado di stilare una classifica provvisoria, dove per il momento in prima posizione si trova il transfer learning, seguito dallo step learning e infine il training from scratch. Ciò nonostante, non si è in grado di indagare completamente il modo in cui la rete classifica gli eventi in base alle strategie. Per questo motivo si utilizzano le matrici di confusione.

Esse sono presentate a gruppi, una per ogni livello di SNR, ogni gruppo per una strategia. Sulle righe troviamo le label, ovvero le vere classi di apparteneza delle immagini, mentre sulle colonne il riferimento alle predizioni, *predictions*, quindi le classi che sono state assegnate alle immagini di imageTest dalla rete.

Dalla lettura delle tabelle precedenti (figure 28,29 e 30) si possono riscontrare buoni valori di performance per valori alti di SNR. Mediante le matrici di confusione (figure 31, 32 e 33) è possibile scavare più a fondo nell'analisi della classificazione, in particolare si possono dedurre i due punti deboli della rete, ovvero la classificazione del background e gli screams.

Le cause che la variabilità di suoni e rumori con cui è stato costruito il background comportano si ritrovano in queste matrici. Si può osservare come il background sia l'unica classe che presenta falsi negativi per tutte le altre classi. Ciò significa che ci sono categorie di suoni nel background che sono confuse completamente dalla rete. La classe glass è quella con cui la rete confonde maggiormente gli eventi background, e questo è vero per ogni strategia.
Per gli screams invece, si può notare come la rete tenda a misclassificare molti eventi come background. Questo è un punto importante su cui riflettere poiché nel caso di un'implementazione reale è possibile accettare dei falsi positivi ma è molto meno tollerabile che eventi di interesse siano persi e classificati come background. Ovviamente si vuole cercare di minimizzare il numero di falsi negativi. Nel caso corrente la strategia del transfer learning è quella che si comporta meglio sotto questo aspetto, sia per la classe background che per quella screams.

Per un giudizio finale si è utilizzato il punteggio fornito dalla metrica F-score, riassunto in figura 34, da cui si evince che il transfer learning è tra le tre strategie la più performante.

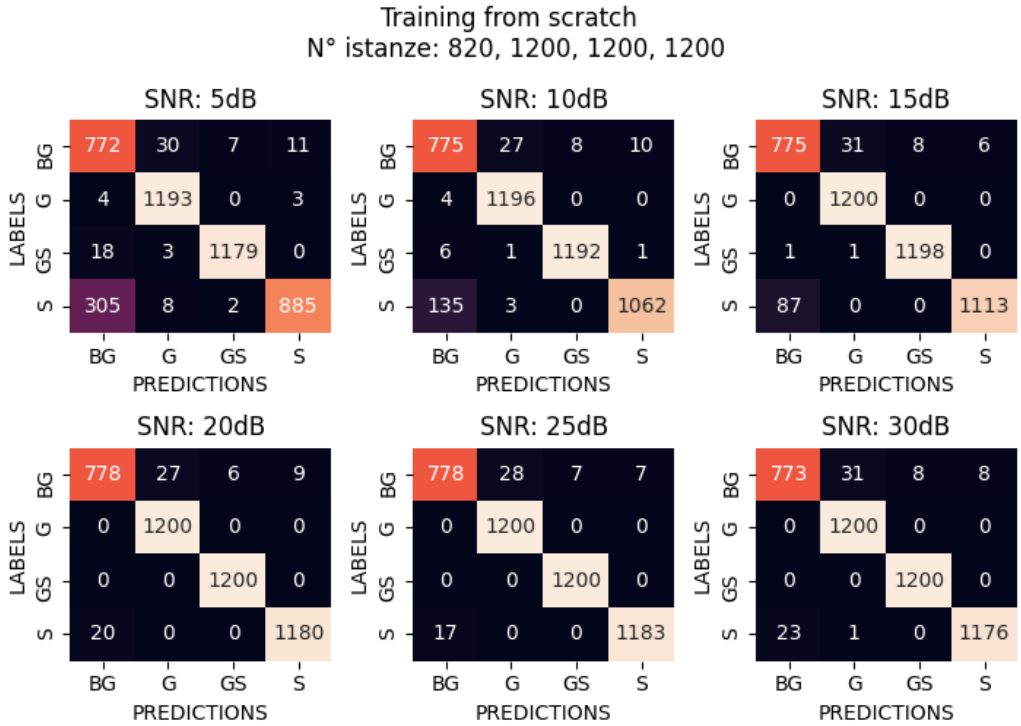


*Figura 31. Matrici di confusione per la strategia training from scratch.*

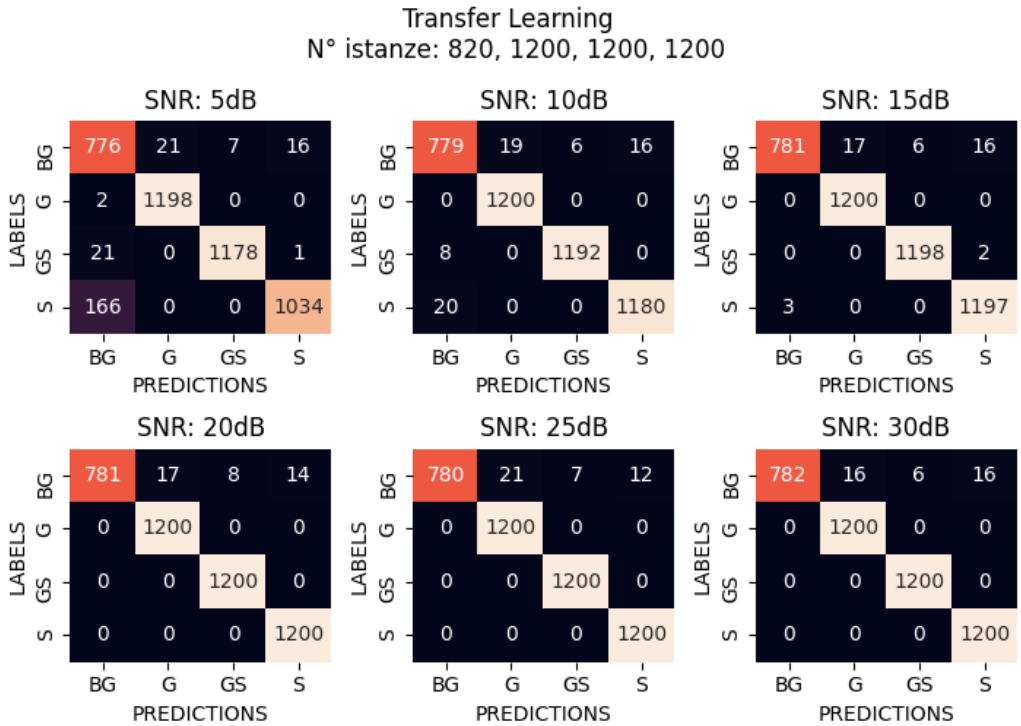


*Figura 32. Matrici di confusione per la strategia transfer learning.*

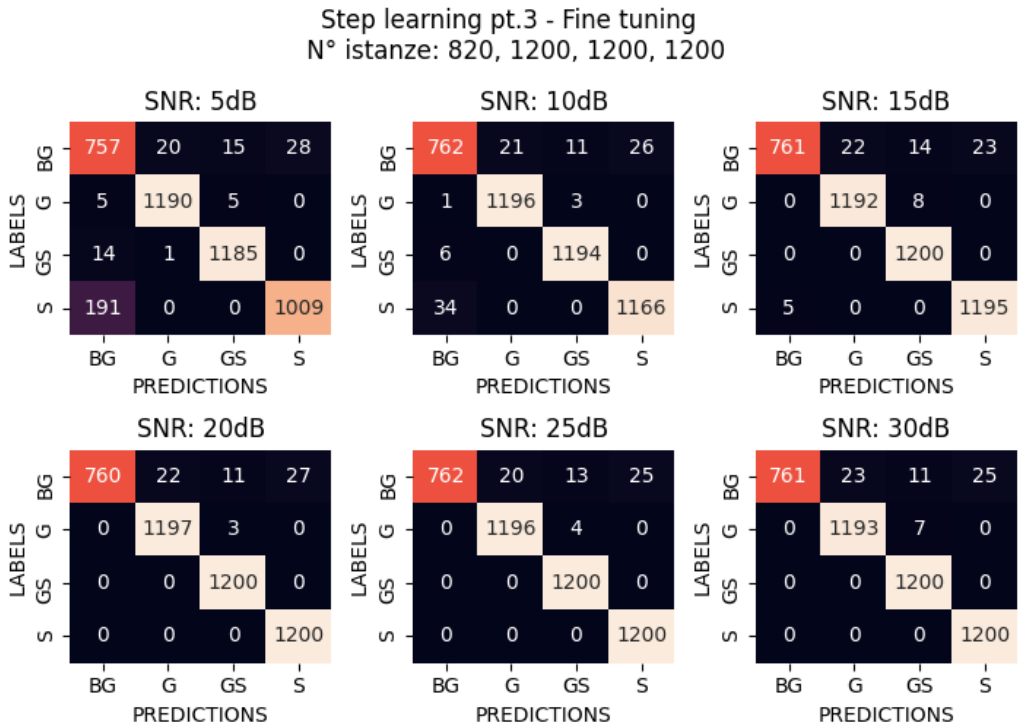


*Figura 33. Matrici di confusione per la strategia step learning fase 3.*

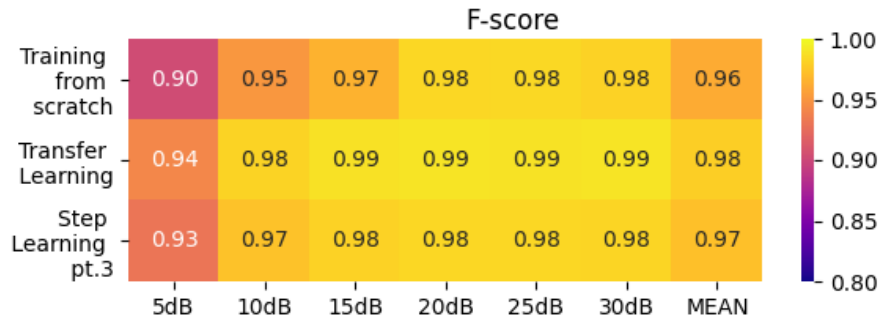


*Figura 34. F-score per il confronto finale tra le tre strategie di allenamento.*

### 5.4.3 Confronto tra le rappresentazioni tempo-frequenza

Vista la difficoltà nel classificare eventi di background, questa classe è stata ampliata cercando di fornire un dataset più omogeneo, imageTrain è stato ampliato fino a 67632 immagini, mentre imageTest fino a 29040 immagini. Per quest'ultimo, per ogni livello di SNR si sono ottenute 4840 immagini, di cui 1200 per ogni classe d'interesse e 1240 per la classe background. imageTrain è stato suddiviso come nella precedente configurazione di allenamento, in trainingImage, trainingLabel e validationSet

Una volta scelta la strategia di allenamento della rete sono stati condotti allenamenti specifici per un confronto tra le tipologie di rappresentazioni tempo-frequenza. In particolare, si è verificato se la scelta di una rappresentazione in scala Mel o un filtraggio gammatono può portare benefici rispetto all'utilizzo di un classico spettrogramma.

La configurazione dei parametri è stata la seguente

| **Learning rate** | **Batch size** | **Funzione costo** | **Algoritmo di ottimizzazione** |
|---|---|---|---|
| $\boldsymbol{lr = 10^{-5}}$ | $BS = 64$ | Sparse Categorical Cross Entropy | Adam |

*Tabella 11. Insieme dei parametri per la compilazione della rete neurale.*

| | |
|---|---|
| **ModelCheckpoint** | metric: *validation loss*<br>frequenza salvataggio: ogni epoca |
| **ReduceLROnPlateau** | metric**:** *validation loss*<br>factor: 0.7<br>patience: 10<br>Min lr: $10^{-12}$ |
| **EarlyStopping** | Metric: *validation loss*<br>patience: 50<br>min delta: $10^{-9}$ |

*Tabella 12. Insieme dei paramentri per l'inizializzazione delle callback*

In questo caso le callback sono state inizializzate in modo da poter effettuare degli allenamenti più lunghi, infatti il parametro patience è stato settato a 50 epoche e il Min lr a $10^{-12}$.Riassumendo gli allenamenti sono stati condotti con la strategia transfer learning, con immagini basate su spettrogrammi, spettrogrammi in scala Mel e gammatonogrammi, sia con eventi centrati sia con eventi ranpad.

Di seguito sono riportati gli andamenti delle curve di allenamento in fase di train e validation, i cui andamenti hanno caratteristiche del tutto analoghe a quelli illustrati nel confronto tra le strategie.

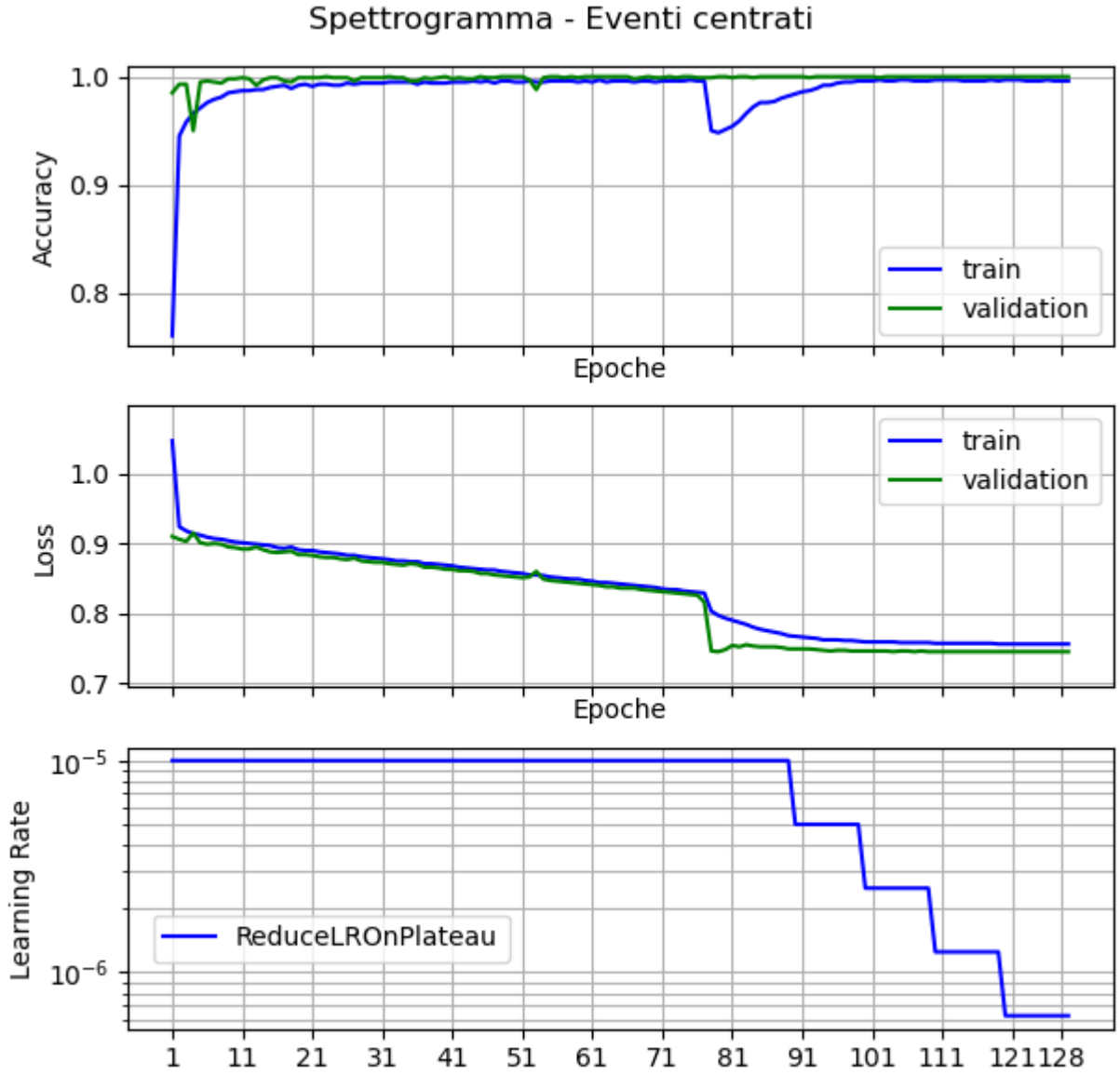


*Figura 35. Allenamento della rete con spettrogrammi e approccio eventi centrati.*

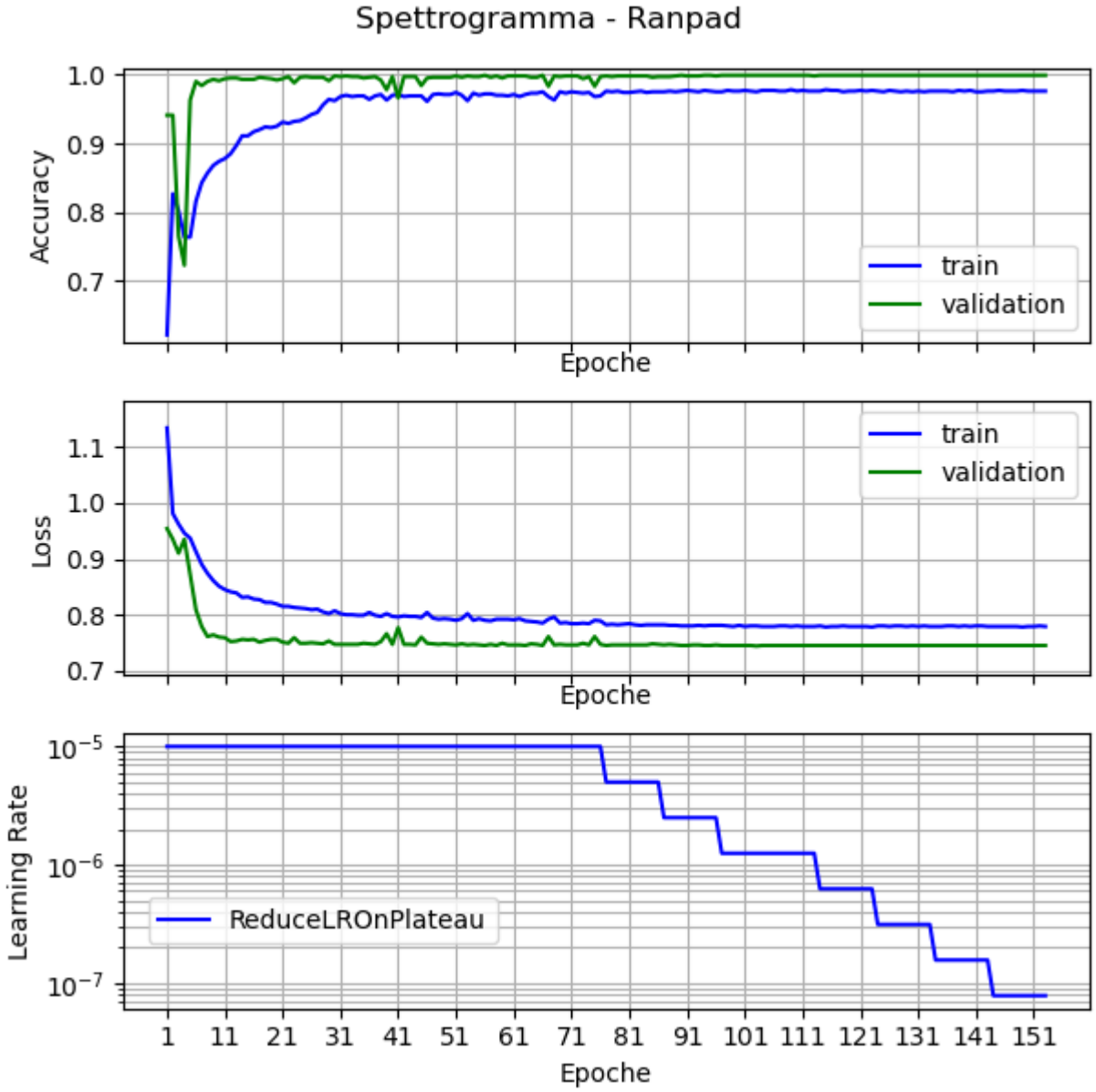


*Figura 36. Allenamento della rete con spettrogrammi e approccio ranpad.*

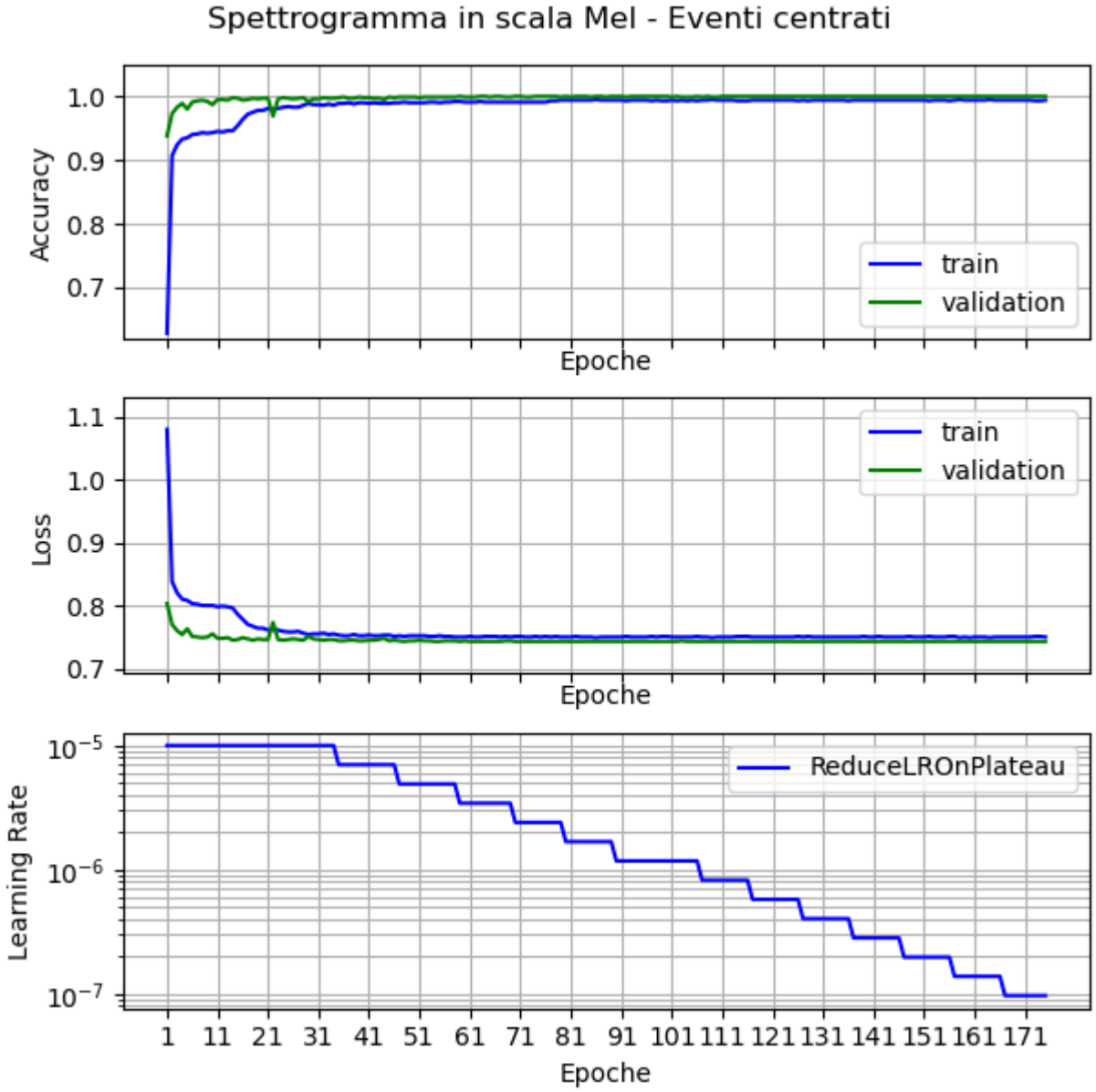


*Figura 37. Allenamento della rete con spettrogrammi in scala Mel e approccio eventi centrati.*

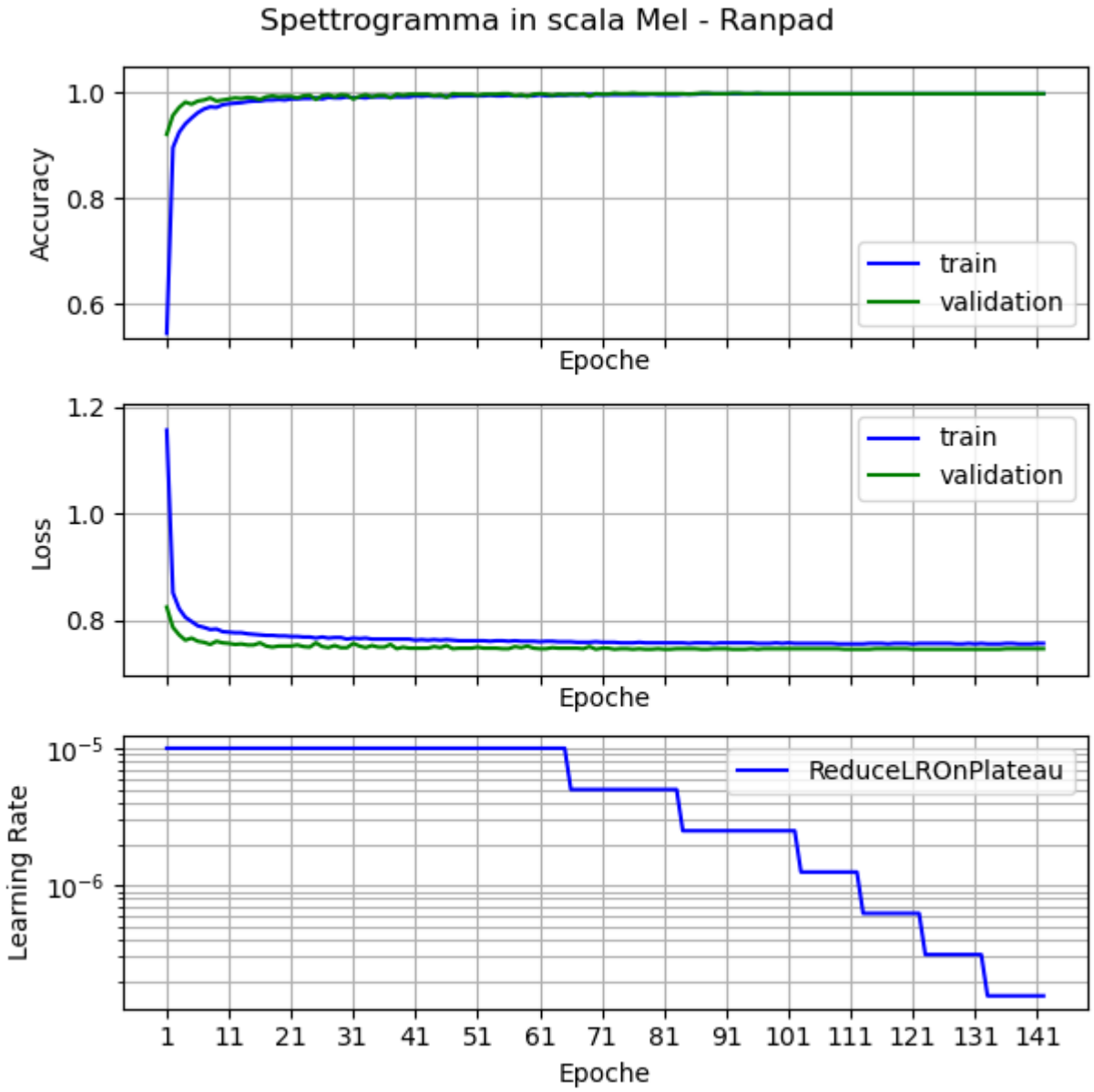


*Figura 38. Allenamento della rete con spettrogrammi in scala Mel e approccio ranpad.*

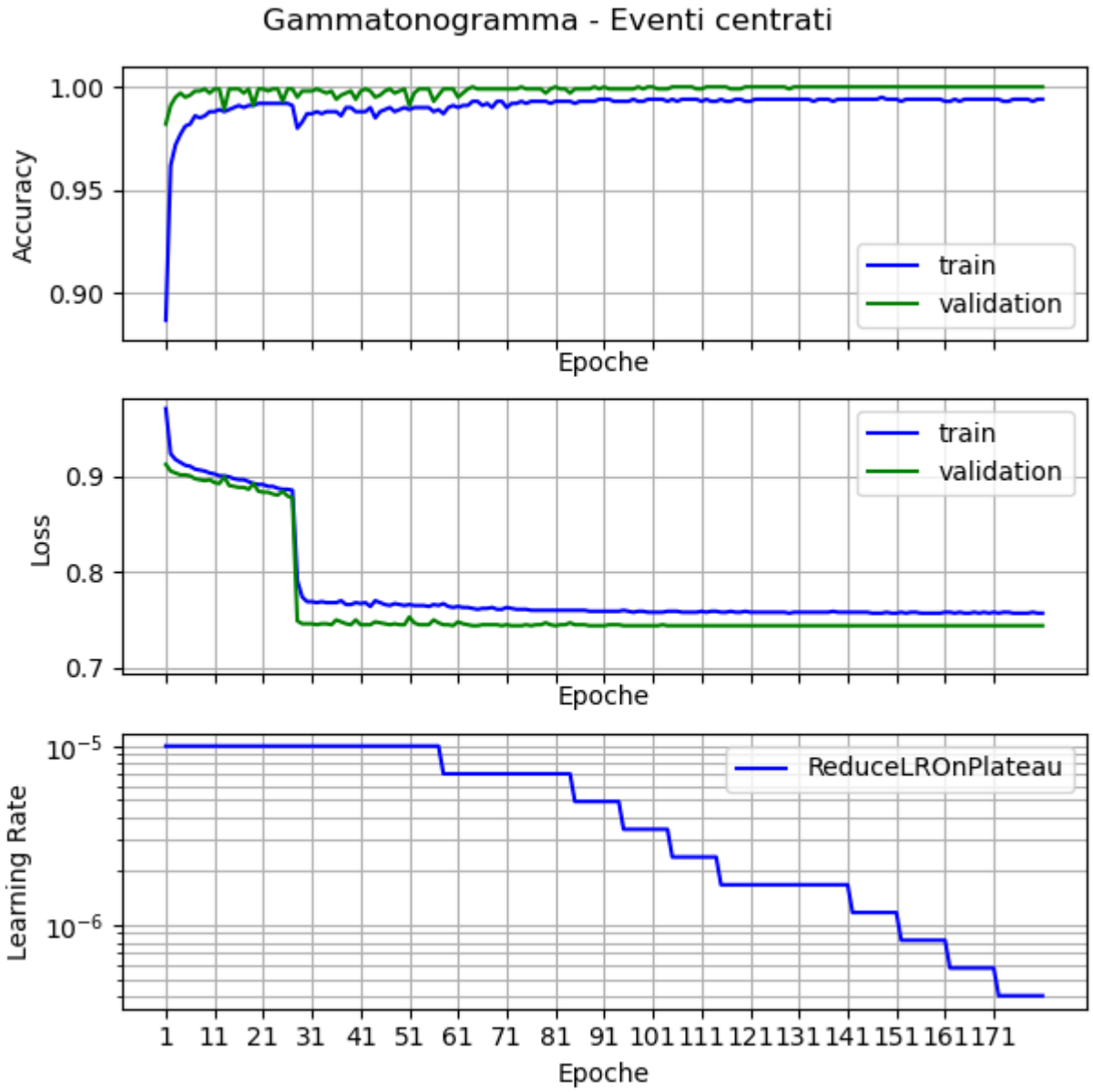


*Figura 39. Allenamento della rete con gammatonogrammi e approccio eventi centrati.*

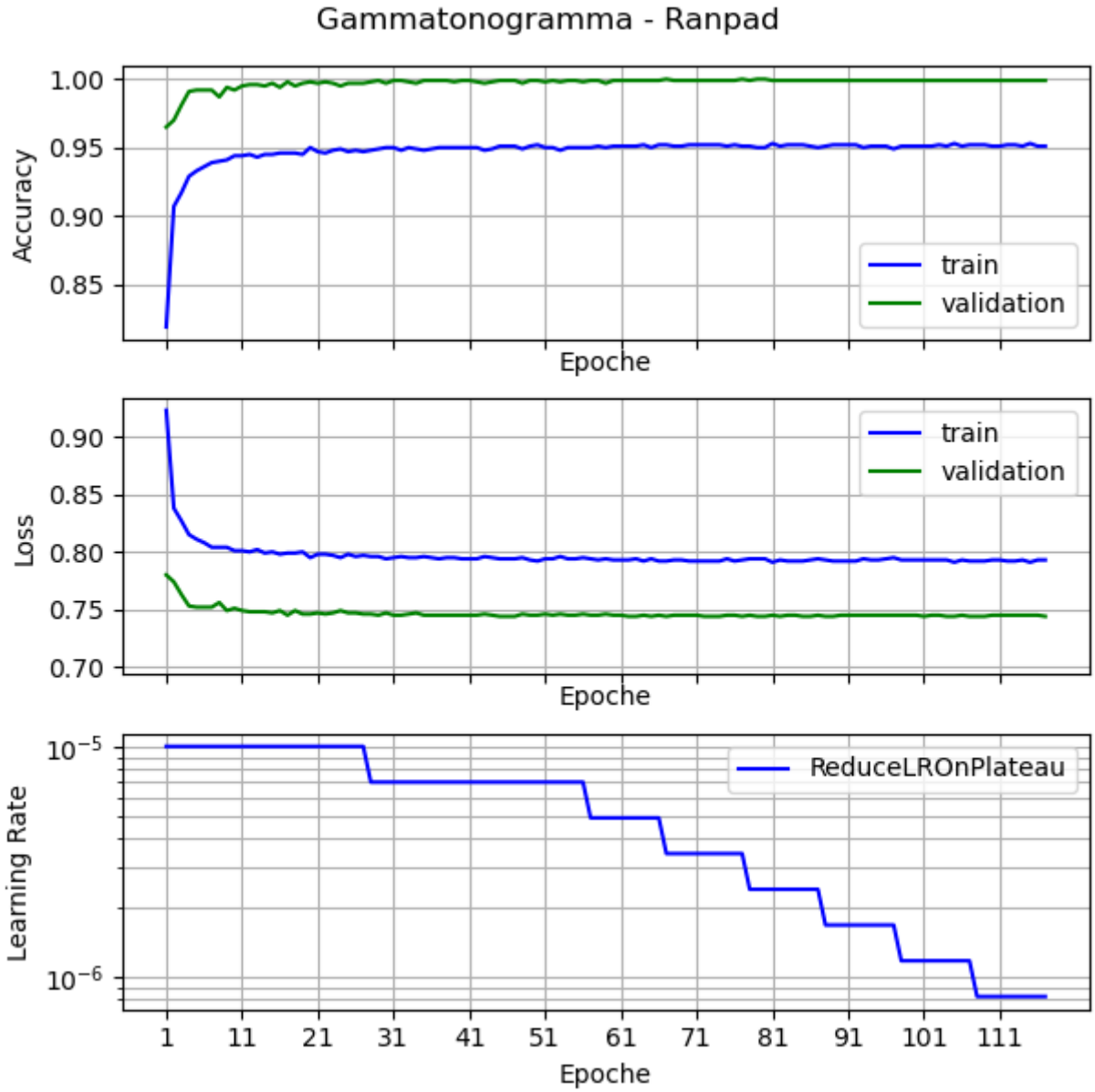


*Figura 40. Allenamento della rete con gammatonogrammi e approccio ranpad.*

Per l'allenamento con spettrogrammi ed eventi centrati (figura 35) l'accuracy raggiunge un valore molto alto approssimabile ad 1. Si può notare un drop dell'accuracy di train il quale avviene in corrispondenza della diminuzione improvvisa della loss di validation. Si potrebbe spiegare questo fenomeno come una fase transitoria di adattamento in risposta alla variazione della loss. Al contempo, l'accuracy di validation non accusa nessuna variazione, questo fa ben sperare in un allenamento robusto. L'allenamento si conclude dopo 128 epoche e i valori dei quattro andamenti si stabilizzano a valori simili a quelli

ottenuti in fase di studio delle strategie. Un comportamento del tutto simile si ottiene durante l'allenamento con spettrogrammi con eventi ranpad (figura 36). L'andamento dell'accuracy inizialmente presenta una varianza maggiore, con la necessità di un tempo maggiore per stabilizzarsi durante il training. Questo è stato un comportamento prevedibile visto che la rete ha dovuto imparare a riconoscere gli eventi indipendentemente dalla loro posizione nell'immagine.

Questo è confermato anche dal fatto che sono state necessarie più epoche per completare l'allenamento. Ciò nonostante, anche in questo caso si può confermare di aver ottenuto un buon allenamento della rete.

Proseguendo con la carrellata di allenamenti, il prossimo è quello con spettrogrammi in scala Mel (figure 37 e 38). In realtà, c'è poco da aggiungere, gli andamenti delle metriche presentano andamenti pressoché ideali e stabili, con valori molto alti di accuracy e discesa della loss costante. Anche in questo caso, il valore della loss raggiunge il solito minimo locale, ciò significa che esso non dipende dal tipo di preelaborazione dei dati grezzi, e quindi da quale rappresentazione tempo-frequenza si scelga. Molto probabilmente esso dipende maggiormente da come la rete viene inizializzata, anche se nello studio delle strategie gli
allenamenti hanno condotto la loss verso lo stesso minimo locale.
Si può dedurre come sia stato più facile per la rete raggiungere il punto di minimo locale, impiegando un numero di epoche minore rispetto al caso con spettrogrammi. La riduzione del learning rate ha sicuramente aiutato a stabilizzare le curve e raggiungere il minimo locale, ma gran parte della diminuzione è avvenuta precedentemente alla prima chiamata della callback ReduceLROnPlateau. Da queste considerazioni si evince che molto sicuramente gli spettrogrammi in scala Mel consentono alla rete di "vedere" meglio le informazioni discriminanti di ogni evento, in quanto, come la teoria suggerisce, molte di queste si trovano a basse frequenze. Quindi certamente questa rappresentazione consente un più facile allenamento della rete.
Stranamente l'allenamento con eventi decentrati esibisce andamenti con una

varianza iniziale minore, a differenza dei casi con eventi centrati e dei soli spettrogrammi, come se questa proprietà ulteriore delle immagini aiutasse indirettamente l'allenamento della rete. Questo lo si può notare anche dalla durata dell'allenamento, in quanto sono state necessarie 30 epoche in meno circa rispetto al caso con eventi centrati. Perfino le curve di train e validation dell'accuracy sono praticamente sovrapposte l'una all'altra, evento mai accaduto negli allenamenti precedenti. Tuttavia, non è da questi grafici che si decide quale delle rappresentazioni è la migliore.

In fine, l'ultimo allenamento è quello con i gammatonogrammi (figure 39 e 40). Questa tipologia di rappresentazione ha causato un allenamento molto simile a quello con gli spettrogrammi. Ciò lo si può dire guardando l'andamento della loss, dalla quale si osserva una diminuzione iniziale costante, quasi lineare, e a seguire la stessa diminuzione repentina, per poi posizionarsi nel minimo locale quasi immediatamente. Anche in questa occasione la chiamata della callback per il learning rate è avvenuta successivamente a questo evento. Per l'allenamento dei gammatonogrammi ranpad si raggiunge il medesimo minimo locale, l'accuracy di validation presenta valori analoghi ai casi precedenti. Solamente l'andamento dell'accuracy di train non riporta gli stessi valori, essa si stabilizza leggermente sopra al 95%. Il numero di epoche necessarie è affine a quello dei due casi con spettrogrammi in scala Mel.

A valle dell'allenamento segue la fase di valutazione delle prestazioni con le immagini di testing contenute in imageTest. Di seguito sono riportate le tabelle con le accuracy ottenute per ogni livello di SNR e gli errori percentuali per ogni classe, raggruppate per tipologia di rappresentazione.

Nel primo caso di studio, ovvero con gli spettrogrammi, l'allenamento con gli eventi centrati ripropone in toto i risultati ottenuti con il transfer learning nel confronto tra le strategie. Si concentra quindi l'attenzione sul confronto tra

eventi centrati e ranpad. Le performance evidenziate rispecchiano quanto anticipato poc'anzi nel commento dell'allenamento, ovvero si può vedere come generalmente il caso ranpad sia meno performante rispetto al primo, mediamente con due punti percentuali in meno per l'accuracy. Il motivo lo si può trovare nella percentuale di errore del background, che nel caso ranpad per l'appunto risulta globalmente maggiore di circa il 9%. Per la classificazione di eventi appartenenti a glass si può affermare di aver ottenuto le stesse prestazioni. Nel caso di gunshots invece per valori alti di SNR si ha la stessa percentuale di errore tra eventi centrati e ranpad, mentre per valori bassi di SNR si nota una difficoltà di classificazione maggiore per il caso ranpad. Contrariamente al trend appena mostrato, l'errore di classificazione per la classe screams spinge a favore del caso ranpad, nel quale per SNR pari a $5dB$ si è ottenuto la metà dell'errore percentuale rispetto agli spettrogrammi centrati.

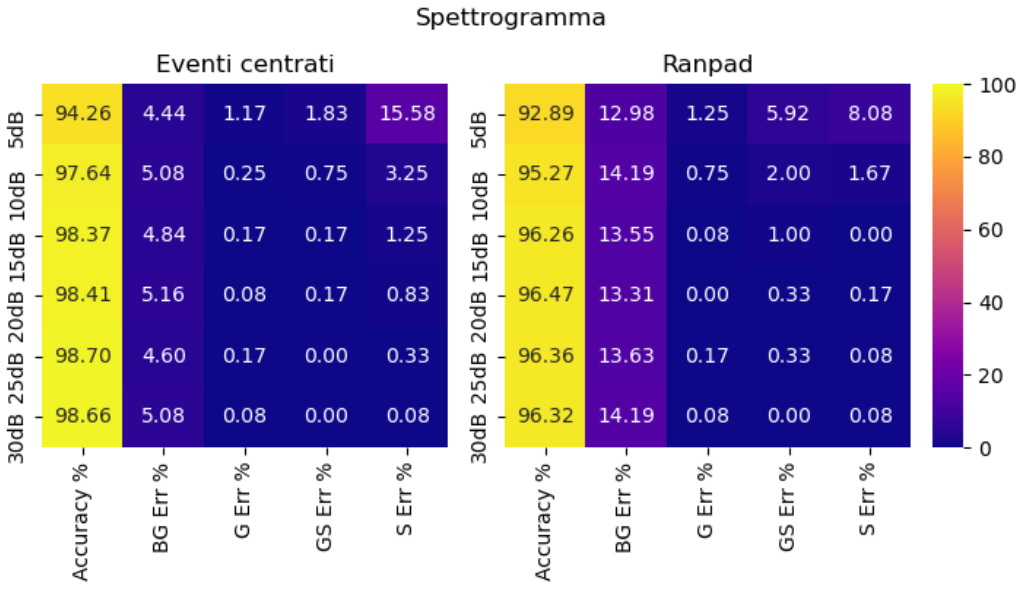


*Figura 41. Tabella dei valori di accuracy e degli errori percentuali per lo spettrogramma. A confronto gli approcci eventi centrati e ranpad.*

A seguito di quest'analisi sono riportate le matrici di confusione per i due casi (figura 42 e 43). Da esse è possibile notare la nota dolente nella classificazione di eventi di background, già evidenziata nelle precedenti tabelle. Si può

osservare come i falsi negativi per la classe background siano presenti per ogni altra classe. E come questi siano in numero maggiori per il caso ranpad. Concentrandosi sulla matrice riferita all'SNR di $5dB$ si può notare con distinzione la differenza nei falsi negativi per la classe screams tra eventi centrati e ranpad. Questo è un dettaglio importante da evidenziare nell'ottica implementativa di un un sistema automatico real time, in quanto ci si aspetta che la probabilità di estrarre un'istanza con un evento centrato sia minore di estrarla con l'evento in una qualsiasi posizione della finestra. Ciò è fondamentale in quanto si vuole garantire di fornire uno strumento che perda il minor numero possibile di eventi d'interesse.

Per le restanti classi si continua ad osservare una notevole robustezza nella classificazione, anche per valori di SNR piuttosto bassi come ad esempio $10dB$. Un buon comportamento di entrambi i metodi lo si nota nel discriminare eventi tra le classi di interesse, ad esempio sono rari i casi in cui un evento glass viene confuso con un evento screams. Ciò non toglie il fatto che tutto si complica con livelli di SNR bassi, infatti questa robustezza viene leggermente meno soprattutto nel caso ranpad per SNR pari a $5dB$.

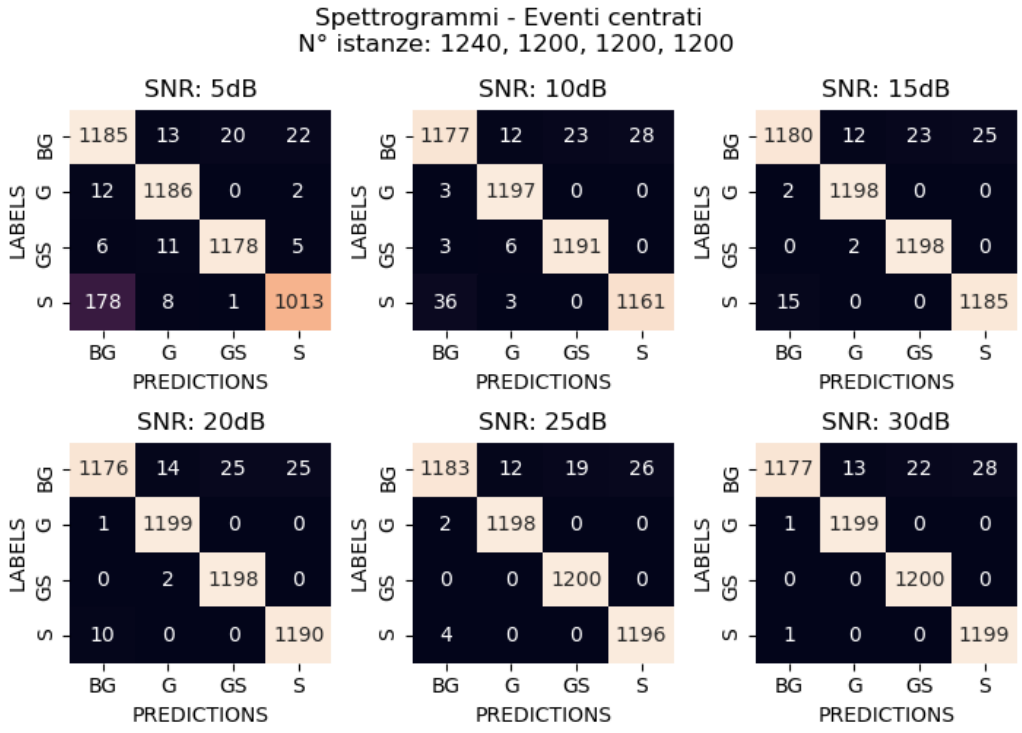


*Figura 42. Matrici di confusione per spettrogrammi e approccio eventi centrati.*

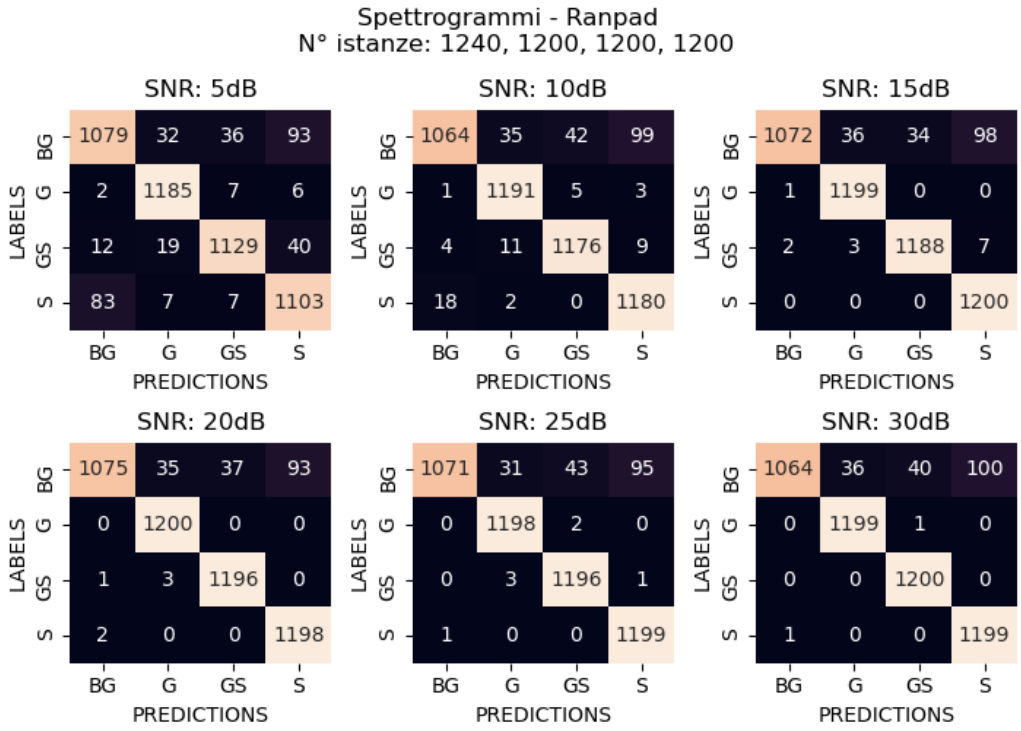


*Figura 43. Matrici di confusione per spettrogrammi e approccio ranpad.*

Con lo scopo di approfondire questo confronto altre due metriche sono state introdotte: *precision* e *recall*. In ogni tabella si trovano i valori delle metriche sia per il caso eventi centrati, sia per il caso ranpad, siglato tramite RP.

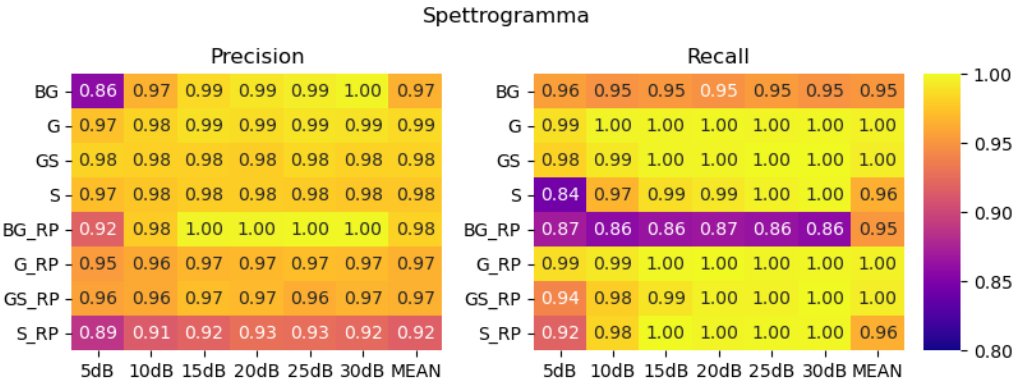


*Figura 44. Metriche di Precision e Recall per lo spettrogramma. A confronto gli approcci eventi centrati e ranpad (RP). Quest'ultimo è posizionato sulle ultime quattro righe di ogni tabella.*

Al fine di analizzare più approfonditamente il comportamento della rete sotto test, la metrica precision aiuta ad evidenziare in quali casi è probabile generare

falsi allarmi, in quanto nella sua definizione si trova il numero di falsi positivi. Nel migliore delle ipotesi, un valore di precision pari ad 1 significa che nessun caso di falsi positivi è stato generato. Ad esempio, per la classe background e un SNR di $30dB$ con eventi centrati, nessuna delle altre classi è stata classificata come background. Si può osservare chiaramente per il caso ranpad classe screams, in cui si evidenziano valori di precision minori rispetto alle altre casistiche, effetto già riscontrato dalle matrici di confusione. Mentre viene evidenziato il migliore comportamento del caso ranpad nel generare un numero minore di falsi positivi per la classe background, soprattutto per SNR pari a $5dB$, il cui valore di precision è $0.92$ contro $0.86$ del caso eventi centrati. È un comportamento già constatato con le matrici di confusione e qui se ne trova la conferma.

Bisogna tenere presente che non si può considerare esclusivamente una metrica di riferimento, soprattutto in un caso multi-classe come il corrente, poiché i falsi positivi per una classe sono anche i falsi negativi di un'altra.

Il comportamento descritto poc'anzi può essere quindi tradotto dal punto di vista della classe screams, e si può dire che nella modalità di eventi ranpad vengono generati un numero minore di falsi negativi. Per questa ragione si tende ad utilizzare in parallelo una seconda metrica, il recall, la quale per l'appunto è definita attraverso i falsi negativi. Essa aiuta a quantificare per quali casi sono stati generati un maggior numero di falsi negativi, il che per le classi d'interesse significa quantificare gli eventi che sono stati persi e classificati soprattutto come background o come un'altra classe. Considerando la classe screams e $5dB$ di SNR si è ottenuto un recall di 0.92 nel caso ranpad contro 0.84 per gli eventi centrati. Da notare che questi due valori sono leggermente più bassi rispetto ai due ottenuti per il precision in quanto la casse background possiede 40 eventi in più delle altre classi, altrimenti sarebbero uguali.

Si analizzano ora le performance riferite alla rappresentazione spettrogramma in scala Mel.

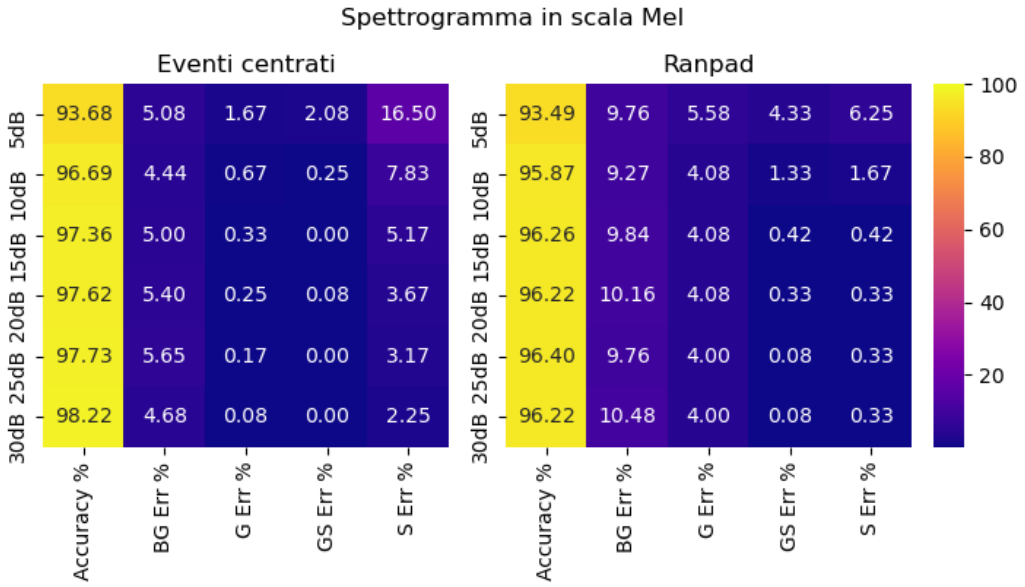


*Figura 45. Tabella dei valori di accuracy e degli errori percentuali per lo spettrogramma in scala Mel. A confronto gli approcci eventi centrati e ranpad.*

In un confronto iniziale, si può affermare che le differenze tra il caso eventi centrati e ranpad sono le stesse risaltate nel caso della rappresentazione con spettrogrammi. In particolar modo, nella percentuale di errore per la classe background.

Risulta più importante quindi sottolineare le differenze tra le due rappresentazioni tempo-frequenza. Considerando il caso eventi centrati, la rappresentazione in scala Mel mostra un punto percentuale in meno rispetto a quella secondo spettrogrammi. Per la classe background esse si potrebbero considerare a parimerito in quanto si hanno all'incirca gli stessi valori.

Continuando con la classe glass, lo spettrogramma risulta leggermente migliore in particolare per i valori di SNR pari a $5dB$ e $10dB$. Complessivamente meglio la classe gunshots per il caso in scala Mel rispetto agli spettrogrammi, per la quale si ottiene errore nullo per quasi tutti i livelli di SNR. Non si può dire lo stesso per l'ultima classe, screams, per la quale la rappresentazione in scala Mel lavora decisamente peggio, e contro ogni pronostico ciò accade per i valori più alti di SNR, con una differenza media di errore di circa il 3%.

Nel caso ranpad invece, la situazione non è analoga a quella appena descritta. L'accuracy globale è circa la stessa, ovvero che si riscontrano gli stessi valori tranne per SNR di $10dB$ e $5dB$, per i quali lo spettrogramma in scala Mel si comporta leggermente meglio. Per la classe background si riscontra una decisa differenza, poiché per la prima rappresentazione si ottiene una percentuale di errore maggiore di circa il 4%, ciò sarà sicuramente meglio analizzato con le matrici di confusione. Tuttavia, il tasto dolente lo si tocca con la classe glass, la quale nella rappresentazione mediante scala Mel presenta un errore percentuale di classificazione di quattro punti percentuali in più per tutti i livelli di SNR, tranne per il livello più basso che presenta un errore di 5.58%. Continuando con la classe gunshots si rileva una percentuale di errore lievemente migliore nel caso in scala Mel, mentre per la classe rimanente c'è da sottolineare che in corrispondenza di un livello di SNR di $5dB$ lo spettrogramma si comporta peggio con due punti percentuali in più rispetto alla rappresentazione concorrente. Per completare l'analisi, a seguire le matrici di confusione.

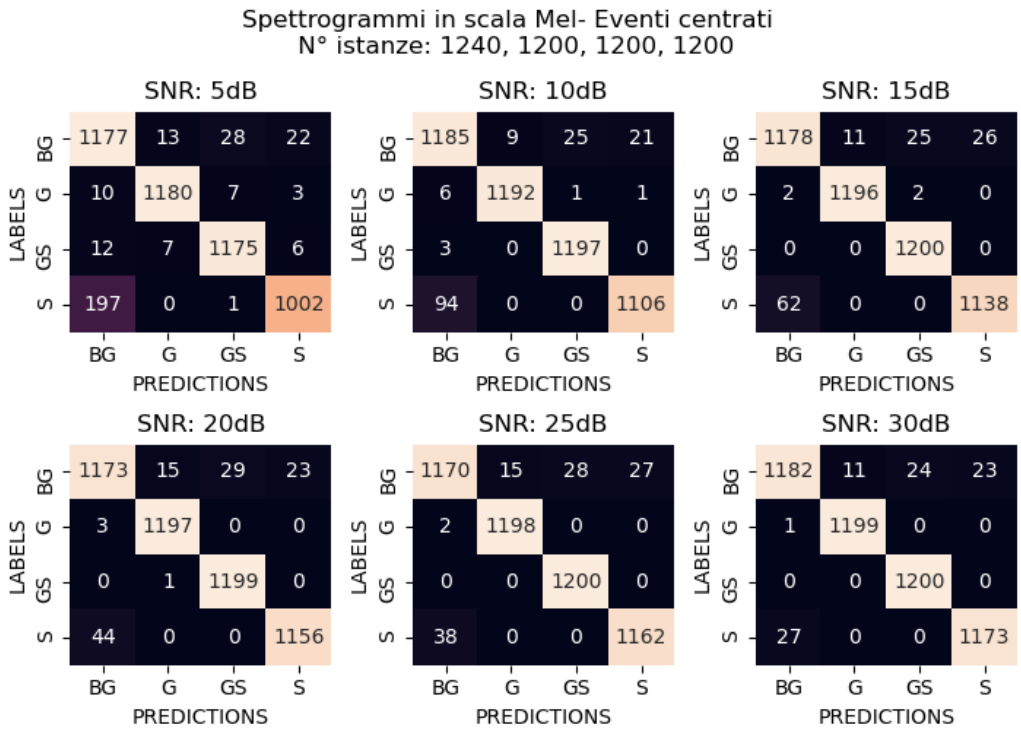


*Figura 46 Matrici di confusione per spettrogrammi in scala Mel e approccio eventi centrati.*

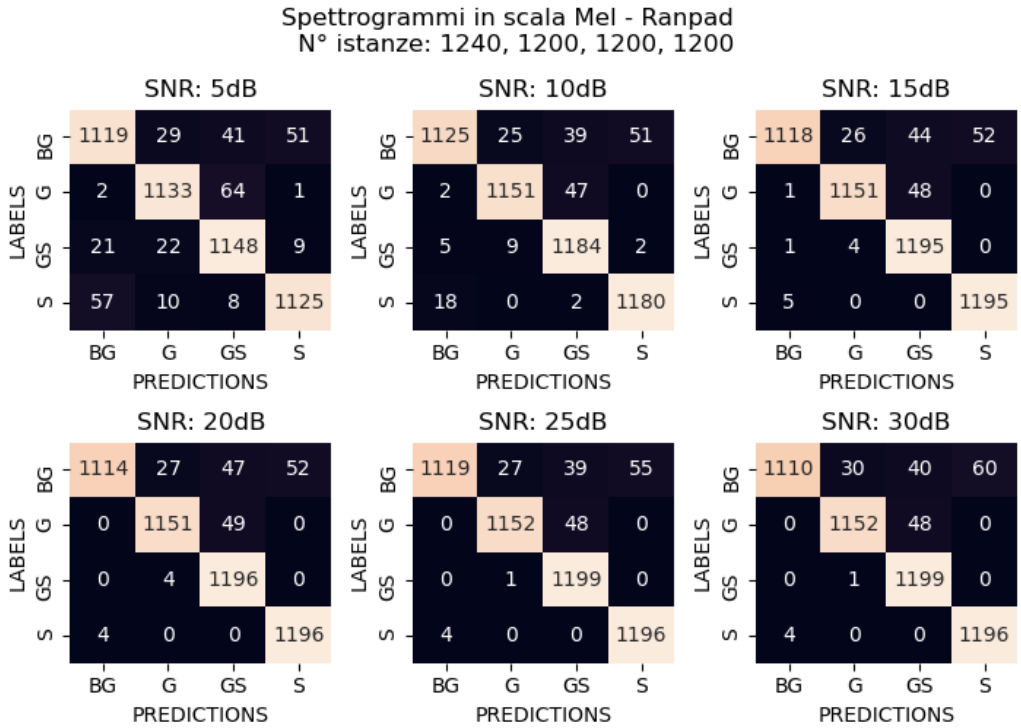


*Figura 47. Matrici di confusione per spettrogrammi in scla Mel e approccio ranpad.*

Ovviamente le matrici di confusione non fanno altro che convalidare il comportamento già descritto nel classificare eventi di background. Si può notare che i falsi negativi di background sono presenti in tutte le altre classi, ciò non fa altro che confermare la difficoltà nel classificare eventi audio immersi in contesti audio composti da una moltitudine di suoni. Tuttavia, si può vedere come la rappresentazione in scala Mel nel caso ranpad abbia aiutato a far diminuire i falsi negativi di screams in riferimento alla classe background, passando nel caso peggiore di $5dB$ da 197 a soli 57 falsi negativi. Ciò fa ben sperare per una implementazione di un detector.

Di seguito le metriche di precision e recall per la rappresentazione spettrogramma in scala Mel.

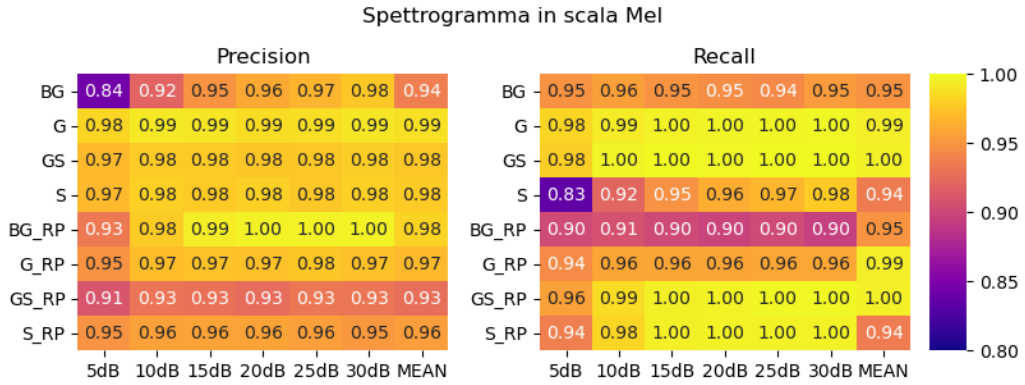


*Figura 48. Metriche di Precision e Recall per lo spettrogramma in scala Mel.*

Concludendo la presentazione delle rappresentazioni, si analizza ora quella basata sul gammatonogramma.

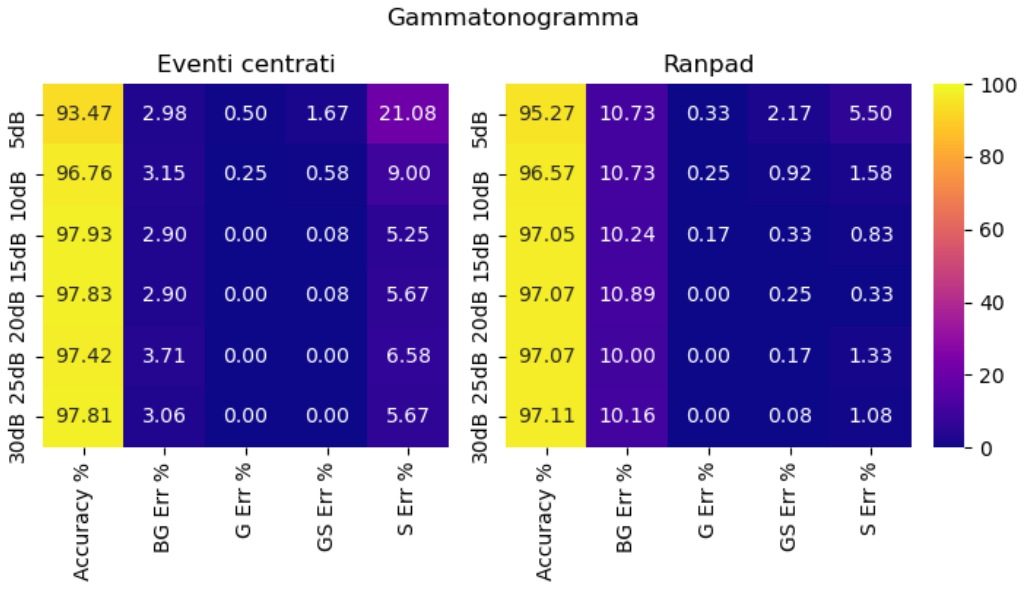


*Figura 49. Tabella dei valori di accuracy e degli errori percentuali per lo gammatonogramma. A confronto gli approcci eventi centrati e ranpad.*

Anche tramite questa rappresentazione si sono ottenuti complessivamente valori di accuracy molto alti, a cavallo di tutti i valori di SNR. Confrontando inizialmente gli eventi centrati con la modalità ranpad, si nota un comportamento migliore di quest'ultima per valori di SNR basso, in particolare con $5dB$, per cui si è ottenuto circa il 2% in più di accuracy. Nonostante questi valori siano simili, ciò non vuol dire che l'errore di classificazione per ogni classe sia simile, infatti, si può notare che per la classe background, la modalità

ranpad dimostra una fatica maggiore nel classificare questi eventi, mostrando in media un errore del 10% contro il 3% circa del caso eventi centrati. Per la classe glass si può invece affermare che sono state ottenute le stesse prestazioni. Proseguendo con la classe gunshots, il caso eventi centrati riporta un errore di classificazione minore rispetto al caso rivale per tutti i livelli di SNR, anche se si può dire che per la classe in questione entrambe le soluzioni lavorano bene. In conclusione, la classe screams come al solito è quella in cui appare la maggior differenza, infatti si può notare come nel caso eventi centrati si sia ottenuto un errore di classificazione del 5.67% in corrispondenza di SNR pari a $30dB$ e addirittura del 21.08% per l'SNR più basso. Tuttavia, ciò non si può dire per il caso ranpad, il quale si comporta decisamente meglio per i valori di SNR alti, nei quali si ottiene un errore percentuale che oscilla tra 0.33% e 1.33%, mentre peggiora per SNR pari a $10dB$ e $5dB$ con un errore percentuale di 1.58% e 5.50% rispettivamente.

Per un'analisi totale è importante studiare come la rappresentazione attraverso gammatonogramma si comporta rispetto lo spettrogramma e lo spettrogramma in scala Mel. Si considera inizialmente il caso eventi centrati. Per l'accuracy i valori sono del tutto paragonabili alle altre due tecniche. Per la prima classe, background, si osserva un errore percentuale minore di circa due punti percentuali, ciò vale per ogni livello di SNR. Ottimi risultati per la classe glass del gammatonogramma, la quale riporta un errore nullo per i valori alti di SNR e solo dello 0.25% e 0.5% in corrispondenza di SNR pari a $10dB$ e $5dB$ rispettivamente, quindi complessivamente minore rispetto le rappresentazioni rivali. Lo stesso si può affermare per la classe gunshots. Al contrario, per la classe screams si sono ottenuti i valori più alti di errore per il gammatonogramma, la peggiore nel confronto tra le tre rappresentazioni.

Considerando ora il caso ranpad, si può notare come la rappresentazione mediante gammatonogramma presenti valori di accuracy più alti in confronto allo spettrogramma e spettrogramma in scala Mel, mediamente di circa un punto percentuale, addirittura di due punti rispetto allo spettrogramma con

SNR di $5dB$. Per la classe di background, si sono ottenuti errori percentuali paragonabili a quelli nel caso mediante scala Mel e quindi globalmente migliori dello spettrogramma. Proseguendo, la classificazione della classe glass con il gammatonogramma ha restituito ottimi risultati, con un errore percentuale per tutti i livelli di SNR inferiore allo $0.5\%$ e in alcuni casi anche nullo. Per la classe gunshots l'errore percentuale è simile per i valori di SNR più alti per tutte tre le rappresentazioni, mentre per i due valori più bassi di SNR si riscontra un errore inferiore per il gammatonogramma. Risultato insolito invece per l'ultima classe, perché si può notare come il gammatonogramma restituisce un errore percentuale minore nel caso di SNR pari a $5dB$ e $10dB$, mentre valori lievemente maggiori con SNR più alti rispetto alle altre due rappresentazioni.

Come consuetudine vengono fornite di seguito le matrici di confusione per la rappresentazione mediante gammatonogramma.

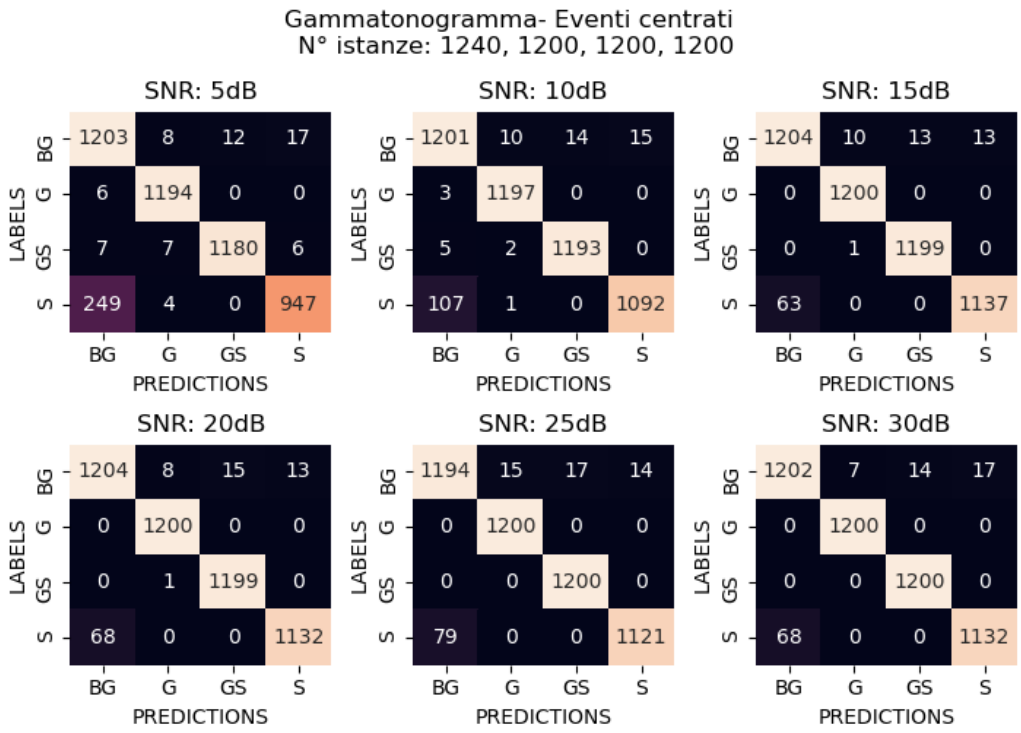


*Figura 50. Matrici di confusione per il gammatonogramma e approccio eventi centrati.*

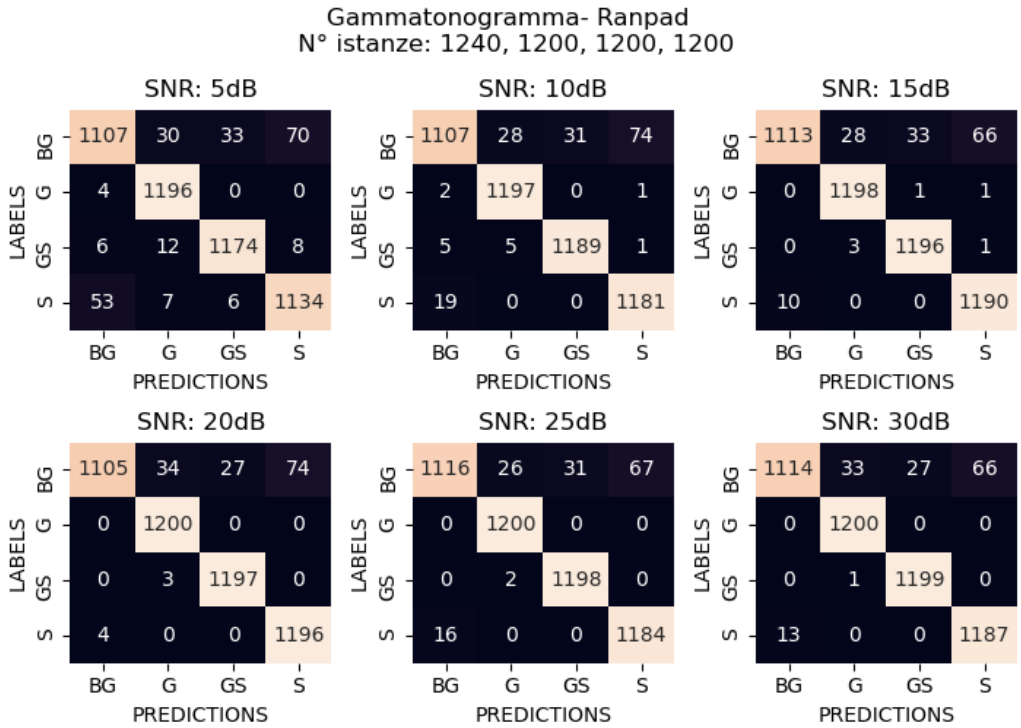


*Figura 51. Matrici di confusione per il gammatonogramma e approccio ranpad.*

Le considerazioni nei confronti delle matrici di confusione ricalcano quelle già fatte precedentemente per le altre rappresentazioni. Si riscontra buona solidità nella classificazione delle classi glass e gunshots. I falsi negativi per la classe background sono sempre presenti per ogni classe, con un peggioramento che si verifica passando dal caso eventi centrati al caso ranpad, mentre si osserva chiaramente la difficoltà nel classificare eventi screams per $5dB$ di SNR nel caso eventi centrati, con un alto tasso di falsi negativi pari a 249 eventi nei confronti della classe background. Al contrario, la situazione si ribalta nel caso ranpad, dove per la stessa casistica si è ottenuto un netto miglioramento. Nuovamente si sottolinea l'arduo compito nel trattare eventi audio vocali, proprio perché il background è ricco di suoni che richiamano voci umane.

Di seguito le metriche di precision e recall per la rappresentazione mediante gammatonogramma.

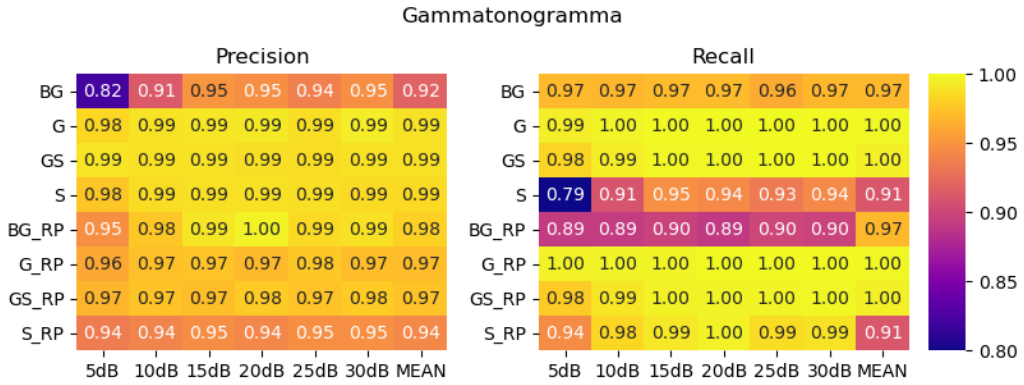


*Figura 52. Metriche di Precision e Recall per lo spettrogramma in scala Mel.*

Il confronto illustrato in questa sezione è stato condotto lungo due dimensioni principali, una è la tipologia di rappresentazione tempo-frequenza, l'altra è come sono inseriti gli eventi all'interno dell'immagine. Si è notato come nessuna implementazione ha prevaricato sull'altra e sono stati evidenziati debolezze e punti di forza di ogni scelta. Al fine di riassumere quanto detto fin ora, di seguito sono riportati i valori dell'F-score. Ogni tabella è organizzata come segue: sulle righe si trova la tipologia di centramento dell'evento, eventi centrati e ranpad, sulle colonne si trovano i livelli di SNR, mentre l'ultima rappresenta il valore medio sui livelli di SNR.

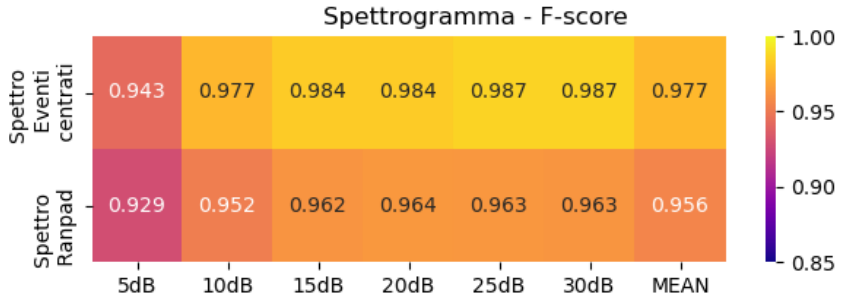


*Figura 53. Punteggio F-score per lo spettrogramma. Confronto tra gli approcci eventi centrati e ranpad.*

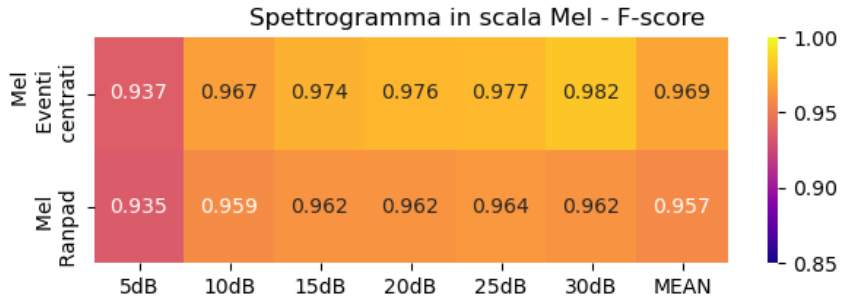


*Figura 54. Punteggio F-score per lo spettrogramma in scala Mel. Confronto tra gli approcci eventi centrati e ranpad.*

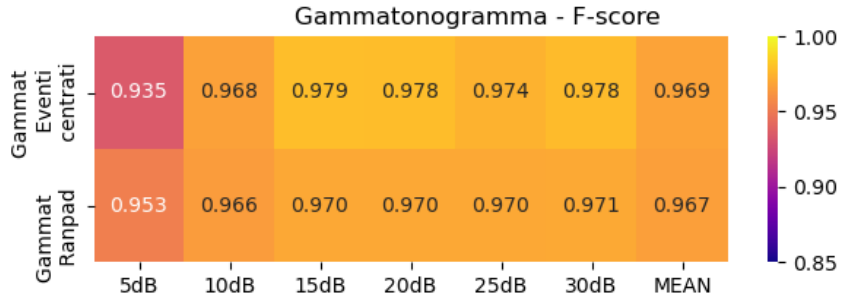


*Figura 55. Punteggio F-score per il gammatonogramma. Confronto tra gli approcci eventi centrati e ranpad.*

Da questi risultati si evince che, indipendentemente dalla rappresentazione considerata, il caso eventi centrati ha conseguito un punteggio medio maggiore del rivale ranpad. Considerando invece il confronto tra le rappresentazioni, la classifica dell'F-score con approccio eventi centrati è la seguente: spettrogramma, spettrogramma in scala Mel e gammatonogramma. Al contrario, considerando l'approccio ranpad, la classifica risulta esattamente opposta. Nel caso più critico, ovvero il livello di SNR $5dB$, si può notare come il gammatonogramma ranpad sia il migliore delle tre rappresentazioni, seguito dallo spettrogramma in scala Mel e infine dallo spettrogramma. Come osservato dal punteggio medio, la classifica si capovolge per il caso eventi centrati.

## 5.5 Training dell'SVM

Per il SVM sono stati condotti due allenamenti, il primo con feature costruite dai singoli eventi e il secondo con feature costruire da finestre successive estratte dai file audio. I due modelli sono stati chiamati *SVM singolo evento* e *SVM real time* rispettivamente. L'approccio singolo evento è stato implementato come proof of concept, ovvero realizzato con lo scopo di dimostrare la validità di costruire un modello SVM. Mentre l'approccio real time simula il comportamento di una possibile implementazione reale, grazie alla modalità con cui sono costruite le feature.

I modelli SVM sono stati implementati secondo la tecnica soft margin con un kernel Gaussian radial basis function.
I dati sono stati organizzati in due coppie di vettori: per la fase di allenamento con nome *mfccTrain* e *LabelTrain*, per la fase di testing *mfccTest* e *LabelTest*. Quest'ultime due sono state organizzate come per i vettori di testing della rete neurale, ovvero organizzate secondo i sei livelli di SNR.
Per il criterio SVM singolo evento mfccTrain è stato costruito con $67632$ vettori di coefficienti, mentre mfccTest da $4840$ vettori per ogni livello di SNR. Per il criterio SVM real time mfccTrain è stato costruito con $11148$ vettori, mentre 836 vettori per ogni livello di SNR di mfccTest.
Per un corretto allenamento dei modelli SVM è stato necessario studiare il valore da assegnare a due parametri: la penalty $C$ e $\gamma$ per il kernel. L'obiettivo è stato quindi identificare la migliore coppia $C, \gamma$ la quale consente di classificare il più accuratamente possibile i nuovi dati, in questo caso quelli appartenenti a mfccTest. Siccome il range possibile di valori per la coppia $C, \gamma$ risulta ampio, la ricerca può diventare lunga. Al fine di ridurre i tempi di allenamento è stato sfruttato l'approccio descritto in [56], dove si suggerisce di utilizzare la funzione *Grid-search* la quale permette di allenare il modello con differenti coppie $C, \gamma$ e che, tramite la tecnica del cross-validation, restituisce il modello con la massima accuracy. È stato scelto un approccio top-down ed è

stato fornito a grid-search un ampio range di valori iniziali per la coppia $C, \gamma$ al fine di effettuare una ricerca grezza dei valori ottimiali; successivamente si è ristretto via via il range di valori per un allenamento fine. Il metodo Grid-search inoltre, consente di parallelizzare gli allenamenti in quanto $C$ e $\gamma$ sono indipendenti, ciò consente di velocizzare la fase di allenamento. A seguire sono riassunti i valori delle coppie $C, \gamma$ per i due criteri di allenamento.

| **SVM singolo evento** | | **SVM real time** | |
|---|---|---|---|
| $\boldsymbol{C = 1.5}$ | $\gamma = 3.1\ 10^{-4}$ | $C = 148.75$ | $\gamma = 10^{-4}$ |

*Tabella 13. Parametri per la classificazione tramite SVM.*

A valle di questo studio è stato lanciato l'allenamento vero e proprio e una volta ottimizzato sono state valutate le prestazioni con mfccTest.
Di seguito sono riportate le percentuali di accuracy e di errore per ogni classe.

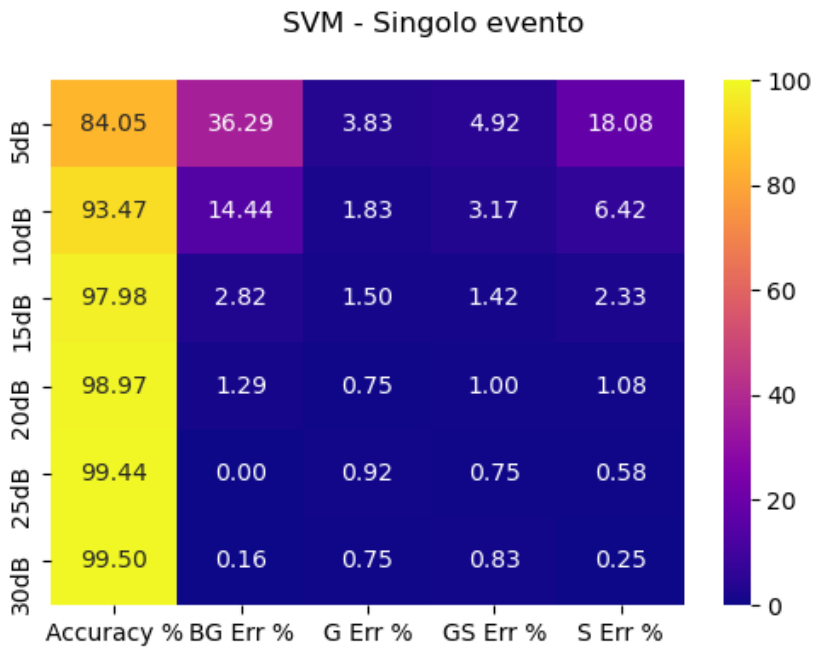


*Figura 56. Tabella dei valori di accuracy e degli errori percentuali per l'SVM singolo evento.*

Si può notare subito come questo modello sia più sensibile al rumore, ciò viene dimostrato dall'accuracy la quale raggiunge l'84% per $5dB$ di SNR; con le reti neurali il caso peggiore è stato ottenuto con lo spettrogramma ranpad: 92.89% di accuracy per $5dB$ di SNR. Si può osservare come a differenza dei risultati

ottenuti con le reti neurali al decrescere del livello di SNR, la classificazione tenda ad essere più difficile per ogni classe. Si può notare come la classificazione di eventi background e screams continui ad essere la più ardua. Questo dimostra che la difficoltà nel classificare queste tipologie di suoni è comune anche a tecniche molto differenti.

Per una visione più completa, sono state costruite le matrici di confusione per i vari livelli di SNR. Si può osservare che per i livelli alti di SNR il comportamento è quasi ideale, addirittura per 25dB di SNR nessun falso negativo di background è presente e questo è l'unico caso in cui è stato ottenuto questo risultato. Si conferma poi, come la classificazione di eventi con basso livello di SNR si complichi drasticamente, già a $10dB$ si osservano falsi negativi per la classe background e per la classe screams in particolare, mentre la classificazione di eventi delle altre classi risulta ancora abbastanza robusta. Ciò non si può più dire per il caso $5dB$, dove si osserva una difficoltà generale per tutte le classi, ma soprattutto come già ribadito più volte, per le classi background e screams.

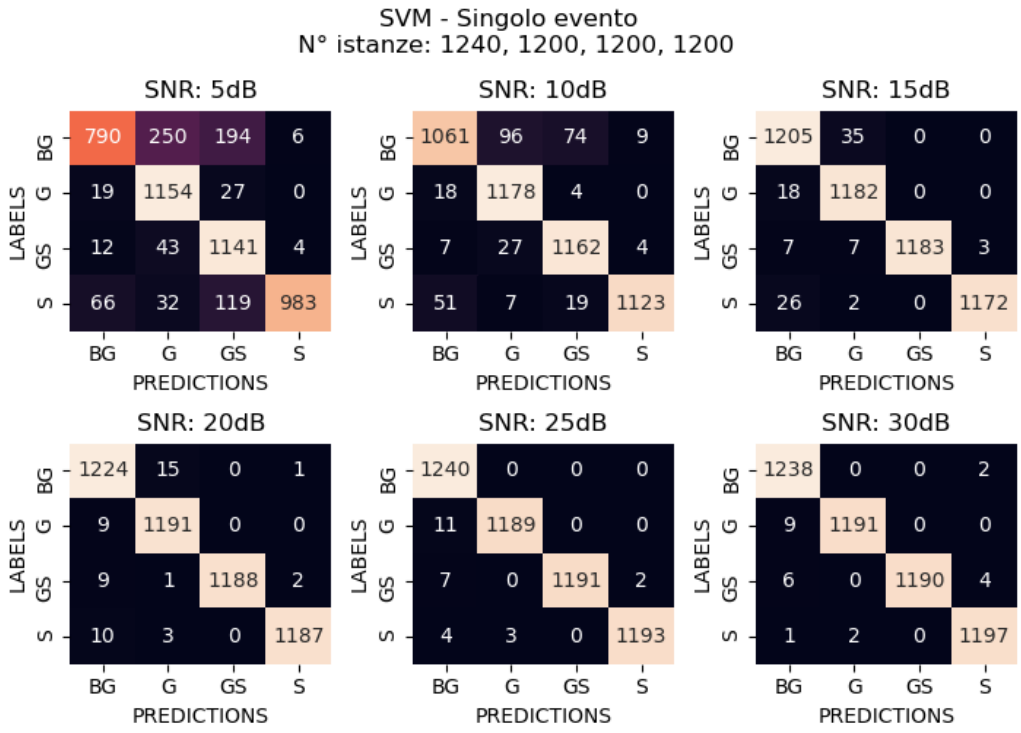


*Figura 57. Matrici di confusione per l'SVM singolo evento.*

Sicuramente, d'interesse maggiore è l'analisi dei risultati per l'approccio real time, in quanto rispecchia maggiormente una possibile implementazione di classificazione su stream audio.

Come ci si poteva aspettare, la classificazione per l'SVM real time è risultata più complicata rispetto al caso precedente. La massima accuracy è del 93% circa per $30dB$ di SNR, la quale si mantiene sopra il 90% fino ad un SNR pari a $15dB$. Diversamente accade per gli SNR più piccoli, la cui accuracy cala vertiginosamente fino ad un valore minimo di 80% circa. L'errore di classificazione per la classe background va dal 9% per il massimo SNR fino al $38.5\%$ per il minimo SNR. La classificazione di eventi glass si comporta meglio, con un errore minimo di 4.83% ad un massimo del 14.1%.
In via del tutto inaspettata, la classe gunshots è quella con il più alto errore percentuale per $30dB$ di SNR, 13.59%, mentre la classe screams presenta valori simili a quelli della classe glass. Si può notare inoltre che l'errore percentuale della classe screams per SNR pari a $5dB$ sia inferiore ai valori ottenuti per le stesse condizioni con le reti neurali nel caso eventi centrati.

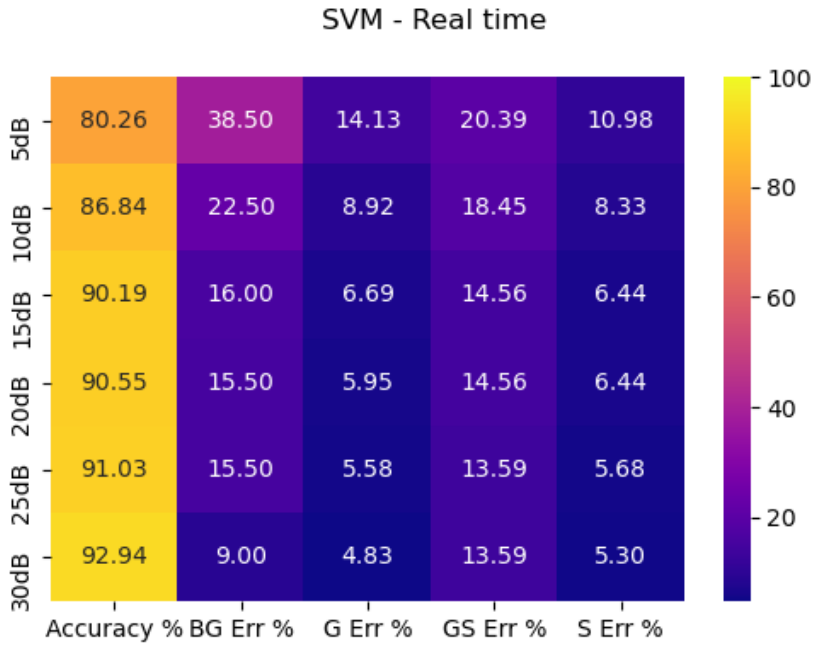


*Figura 58. Tabella dei valori di accuracy e degli errori percentuali per l'SVM real time.*

A seguire le matrici di confusione per il caso SVM real time.

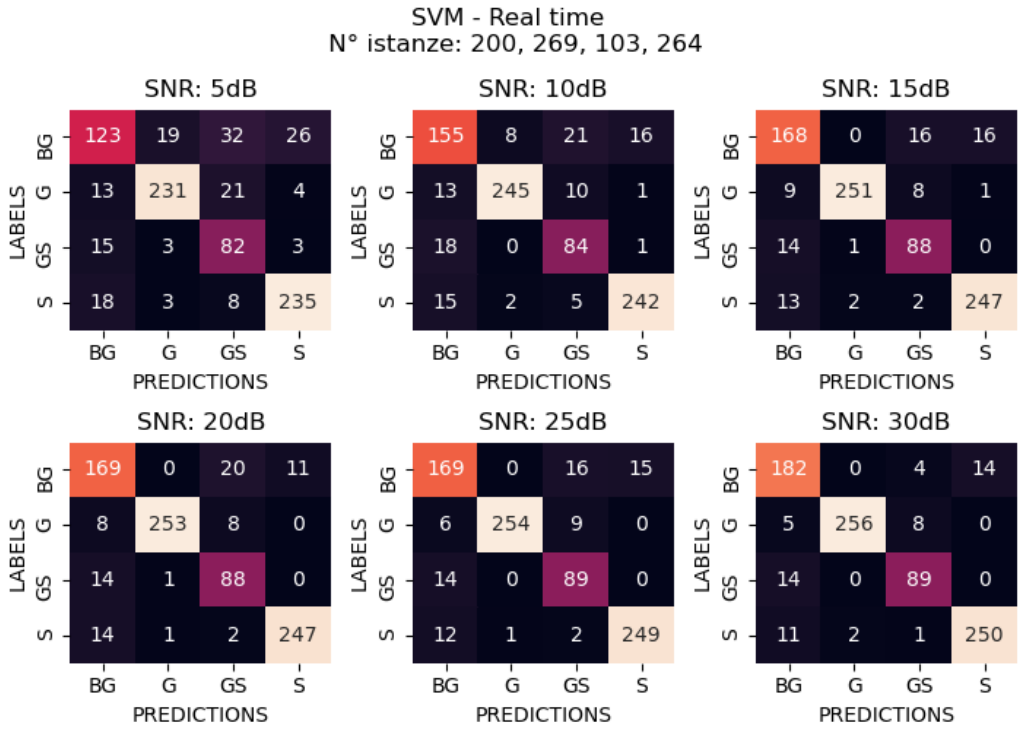


*Figura 59. Matrici di confusione per l'SVM singolo evento*

Le matrici di confusione sottolineano come sia importante implementare un modello robusto nei confronti del background. Questo lo si può dire osservando che la maggior parte dei falsi negativi delle tre classi d'interesse sia in riferimento alla classe background, ciò è vero per tutti i livelli di SNR. Cercare di diminuire i falsi negativi è importante per garantire di perdere il minor numero possibile degli eventi d'interesse. Si può osservare inoltre che per la classe background si riscontri un numero nullo di falsi negativi rispetto alla classe glass, ovvero nessun evento background è stato confuso come evento glass, almeno questo è vero per valori di SNR pari a $15dB$ e maggiori. Ciò non è accaduto per le reti neurali, le quali hanno sempre presentato falsi negativi di background in riferimento a tutte le classi.

Per completare l'analisi del classificatore basato su SVM si riportano si seguito i valori del punteggio calcolato con l'F-score.

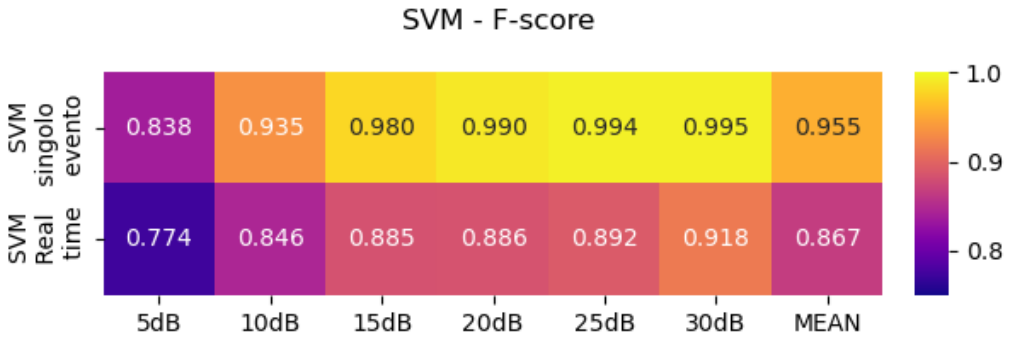


*Figura 60. Punteggio F-score per il confronto tra SVM singolo evento e SVM real time.*

I valori dei punteggi non fanno altro che riassumere quanto dedotto dalle percentuali di errore e dalle matrici di confusione. Ovviamente, l'approccio singolo evento si comporta in maniera più soddisfacente rispetto al caso real time, mentre si può osservare come il punteggio cali drasticamente al diminuire del livello di SNR. Inoltre, dal confronto di questi due approcci si evince che utilizzare finestre piccole causi maggiore difficoltà al riconoscimento di eventi di breve durata come nel caso di gunshots.

# 6 DETECTOR

Nel confronto tra i modelli, stando alla valutazione delle metriche, si può dire che le reti neurali siano in generale più robuste e meno sensibili alla classe background rispetto all'SVM. Si è ribadito più volte come questa sia una peculiarità desiderata in uno strumento automatico di classificazione considerando i vari contesti in cui esso può essere inserito. Si è osservato poi, come le reti forniscano accuracy più elevate dell'SVM; e, considerando le implementazioni ranpad per le reti neurali e SVM real time, si è visto come le prime forniscano una classificazione con una percentuale di errore inferiore e con un punteggio di F-score più alto. Tuttavia, ciò non significa per forza che le reti neurali riescano a classificare con maggior precisione rispetto ad un modello SVM, in quanto gli allenamenti condotti non rispecchiano a pieno una situazione reale, dove le istanze collezionate saranno il risultato di una estrazione di dati audio raccolti dai sensori in tempo reale.

Per questo motivo sono stati valutati i comportamenti dei vari modelli implementando un detector. Il detector è definito come uno strumento di classificazione automatico, implementato in questo caso con le soluzioni studiate, con lo scopo ultimo di analizzare i comportamenti dei modelli e di definirne punti di forza e debolezze. Il detector lavora direttamente con gli audio file del testing set, quindi estrae istanze con una finestra mobile la quale scorre i file audio di tre minuti. Una volta creato l'insieme di istanze, da esse sono state estratte le feature per i vari modelli, quindi le rappresentazioni tempo-frequenza per le reti e i coefficienti mfcc per l'SVM.

Le istanze per le rappresentazioni tempo-frequenza sono state estratte con una funzione finestra rettangolare, di lunghezza pari a 3 secondi e con overlap del 75%, mentre per la costruzione dei coefficienti mfcc la funzione finestra è rettangolare, di lunghezza pari a $500ms$ e overlap del 50%. Sono state estratte mediamente 244 istanze per il primo caso e 733 per il secondo, il numero può variare in base alla lunghezza della traccia e da come sono state estratte le istanze in corrispondenza della fine della traccia audio.

Il problema principale nel costruire il detector è stata l'assegnazione delle label ad ogni istanza. Al fine di valutare metriche e prestazioni è necessario che a monte del processo di predizione effettuato dai modelli si debba assegnare un nome ad ogni istanza per definirne la classe di appartenenza. Dal momento che il detector deve scansionare le tracce audio in tempo reale per segnalare tempestivamente potenziali eventi d'interesse, quest'ultimi non vengono mai estratti centrati nella funzione finestra, al contrario si trovano decentrati o nel caso di eventi audio lunghi a cavallo di più finestre consecutive. Ciò comporta che una determinata istanza può contenere solamente parte di un evento d'interesse e che non si possa definire con certezza se questa appartiene alla classe background o a quella dell'evento d'interesse.
Il problema è stato affrontato definendo un parametro in termini percentuali chiamato *fill*, fissato al 50% per le istanze tempo-frequenza e al 60% per i coefficienti mfcc. Se una determinata istanza contiene almeno il 50% della durata dell'evento o la finestra è occupata almeno per il 50% della sua lunghezza, allora quella istanza contiene un evento d'interesse. Mediante questa unica soluzione è stato possibile gestire sia l'estrazione di istanze con finestre di lunghezze diverse, sia interfacciarsi con eventi audio di durata differente. Al fine di poter calcolare il riempimento della finestra e l'occupazione temporale di un evento è stata costruita una maschera la quale vale 1 dove è presente un evento d'interesse e 0 altrimenti.

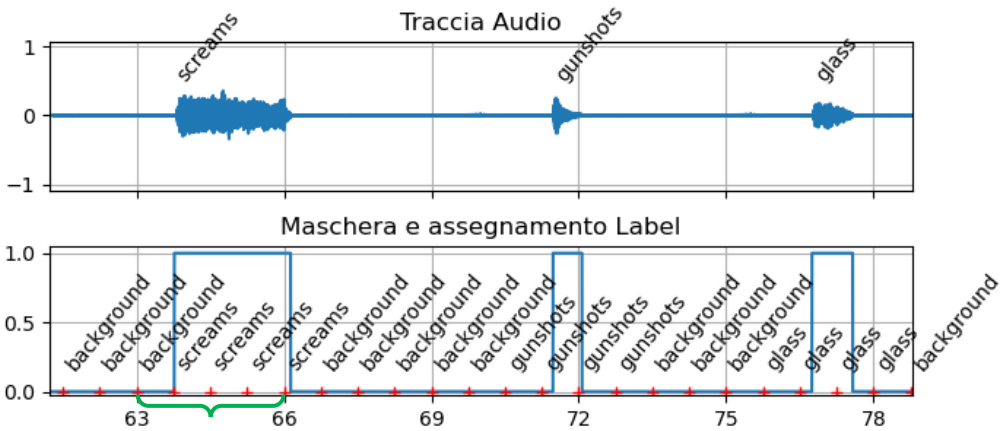


*Figura 61. In quest'immagine si riporta l'esempio di assegnazione delle label per il caso del detector con rete neurale, finestra di 3 secondi. In alto una porzione di un clip audio compresa tra i seocondi 62 e 79. Per il caso del detector con SVM l'assegnazione delle label è analoga, con finesta di 500ms.*

Nella figura è illustrata una parte di traccia audio di circa 20 secondi, dove sono presenti tre eventi audio d'interesse e sopra di essi il nome della classe di appartenenza. Al di sotto della traccia audio sono illustrati: l'andamento della maschera, l'inizio di ogni istanza (di lunghezza 3 secondi) mediante delle croci rosse e le label assegnate ad ognuna di esse. In particolare, considerando l'istanza indicata mediante la graffa verde, tra il secondo 63 e 66, si può osservare che tutto l'evento screams cade in essa e quindi a quell'istanza è stata assegnata la label screams. Facile da giudicare notando che la maschera è più lunga della metà della graffa. Il nome assegnato ad ogni istanza è posizionato a metà di essa.

Un'ulteriore indagine rivela che una volta conclusa l'assegnazione delle label di istanze estratte da un singolo file audio, in media il 50% di esse appartiene alla classe background, mentre le classi d'interesse si spartiscono diversamente la metà rimanente traccia audio per traccia audio. Questa distribuzione sbilanciata del numero di istanze per classi è normale vista la modalità di estrazione, però implica che la valutazione dell'accuracy risulti poco veritiera in quanto essa sarà influenzata maggiormente dalla classificazione della classe background essendo quella in numero maggiore. Per questo motivo si sono considerate metriche alternative quali, *False Positive Rate* (FPR) e *Recognition Rate*. L'FPR in questo caso è definito come il numero totale di falsi positivi in presenza di background diviso il numero totale d'istanze delle tre classi. L'RR invece è calcolato come il numero d'istanze appartenenti alle tre classi d'interesse classificate correttamente diviso il numero totale d'istanze delle tre classi.

Il primo confronto è stato condotto tra una rete allenata secondo l'approccio eventi centrati e una allenata con l'approccio ranpad. Di seguito sono riportati i valori del Recognition Rate e False Positive Rate per le quattro tipologie di modelli. Da osservare che i valori riferiti per l'SVM sono gli stessi nei due casi in quanto per esso non esiste una casistica eventi centrati e ranpad.

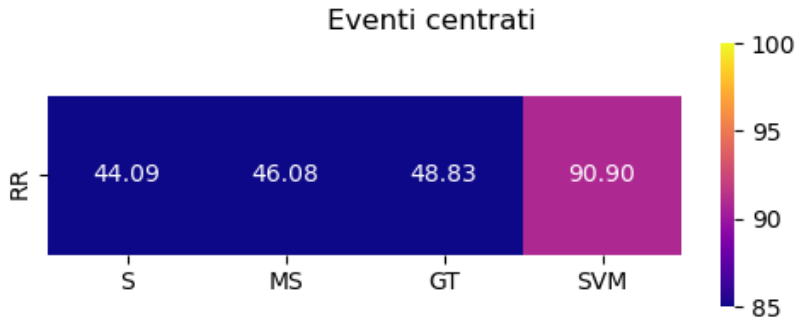


*Figura 62. Recognition Rate per i quattro modelli con approccio eventi centrati per le reti neurali.*

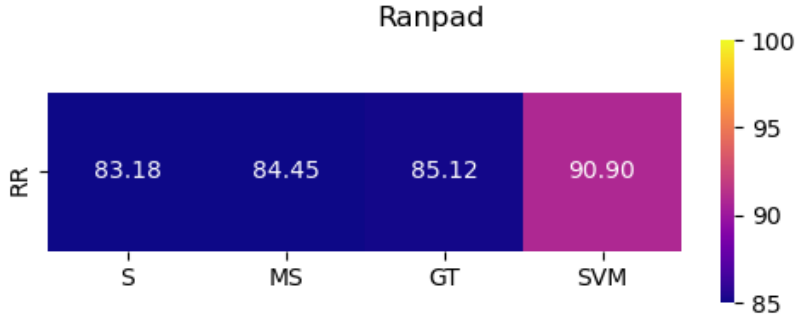


*Figura 63. Recognition Rate per i quattro modelli con approccio ranpad per le reti neurali.*

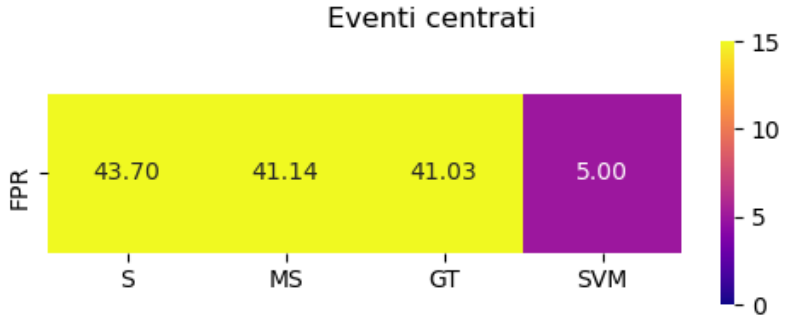


*Figura 64. False Positive Rate per i quattro modelli con approccio eventi centrati per le reti neurali.*

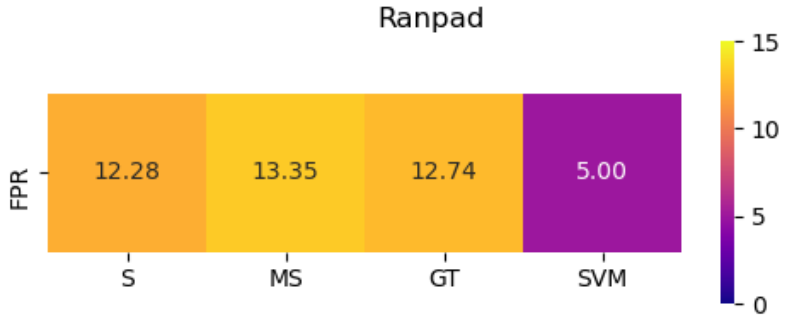


*Figura 65. False Positive Rate per i quattro modelli con approccio ranpad per le reti neurali.*

La prima osservazione riguarda il confronto dei due approcci di allenamento della rete neurale, si può affermare che il caso ranpad è nettamente più performante del caso eventi centrati come dimostrato dai valori di RR ottenuti. Come sottolineato più volte era prevedibile che l'allenamento con questo approccio avrebbe sicuramente rispecchiato maggiormente la realtà. Si può osservare inoltre come anche l'FPR sia più basso nel caso ranpad rispetto a quello eventi centrati. In un caso pratico questa caratteristica conferisce la sicurezza di avere un numero minore di falsi allarmi generati dalla rete.
La seconda osservazione invece riguarda il confronto tra le rappresentazioni tempo-frequenza, considerando il caso ranpad si può affermare che la rappresentazione mediante gammatonogramma presenta l'RR più alto, a seguire lo spettrogramma in scala Mel ed infine lo spettrogramma. Mentre i valori dell'FPR esprimono un tasso per lo spettrogramma pari al 12.28%, seguito dal gammatonogramma con 12.74% ed infine dallo spettrogramma in scala Mel con 13.35%. Complessivamente si può affermare che per le tre rappresentazioni questa metrica risulta piuttosto alta
La terza osservazione importante da fare è quella nei confronti del Support Vector Machine, che in questo confronto ha riportato un tasso di Recognition Rate pari al 90.9%, più alto dei valori di RR per i modelli basati su rete neurale, ed un FPR del 5%.

Bisogna sottolineare un dettaglio per fornire una visione più chiara delle valutazioni appena effettuate. L'estrazione delle feature condotta per il detector ha generato 7200 istanze per le tre classi d'interesse nel caso della finestra di 3 secondi, mentre ha generato 5310 istanze per la classe glass, 1722 istanze per la classe gunshots e 5046 istanze per la classe screams, nel caso della finestra lunga $500ms$. Si nota quindi come l'uso di una finestra più corta abbia generato un numero di istanze minori per la classe gunshot. A tal proposito si ricorda che nell'analisi dei risultati dell'allenamento dell'SVM real time, la classe gunshot ha riportato un errore percentuale molto alto per tutti i livelli di SNR, con una media pari a 15.86%. Da questa considerazione si può dedurre che si necessiterebbe di valutare l'RR e l'FPR per l'SVM con un numero di

istanze per classe il più confrontabile possibile. In modo da poter confrontare al meglio i modelli.

Ciò che è stato possibile condurre per fornire un risultato il più confrontabile possibile è stato calcolare il punteggio tramite F-score escludendo sempre le istanze di background e valutando solamente le classi d'interesse.

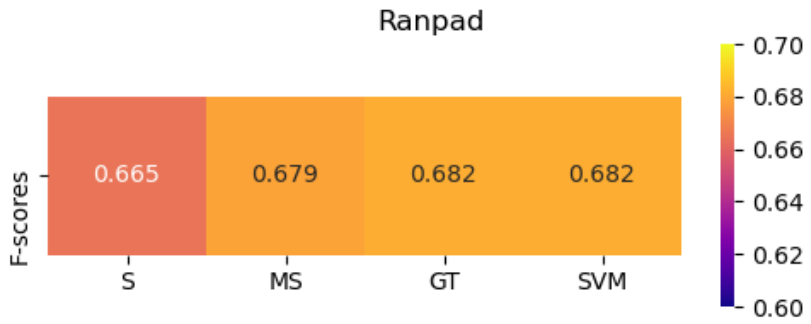


*Figura 66. Punteggio F-score per il confronto tra i modelli.*

Si ricorda che il punteggio tramite F-score è una media armonica dei valori di precision e recall. La media armonica consente di considerare maggiormente i valori più piccoli assegnandogli un peso maggiore. Quindi nel caso specifico dell'F-score, il suo valore sarà tanto più alto quanto più alti saranno precision e recall. Come si può osservare, la rappresentazione gammatonogramma fornisce il valore più alto di F-score rispetto le altre due concorrenti, e come questo sia confrontabile con il punteggio ottenuto per l'SVM.

Per analizzare più nel profondo il comportamento del detector si è scelto di studiare le probabilità a posteriori del classificatore. A tal proposito si è creata una dashboard per visualizzare l'andamento delle probabilità di uscita dei quattro classificatori. Esse vengono mostrate in confronto con una traccia audio selezionata dal testing set, in cui sono state indicate le label delle classi di appartenenza di ogni evento presente, per avere un riscontro diretto sul modo in cui i modelli reagiscono in base alle istanze fornitegli in ingresso.
Di seguito sono mostrate due immagini contenenti le tracce della dashboard, la prima (figura 67) è per un SNR di $30dB$, la seconda (figura 68) per un SNR di $5dB$.

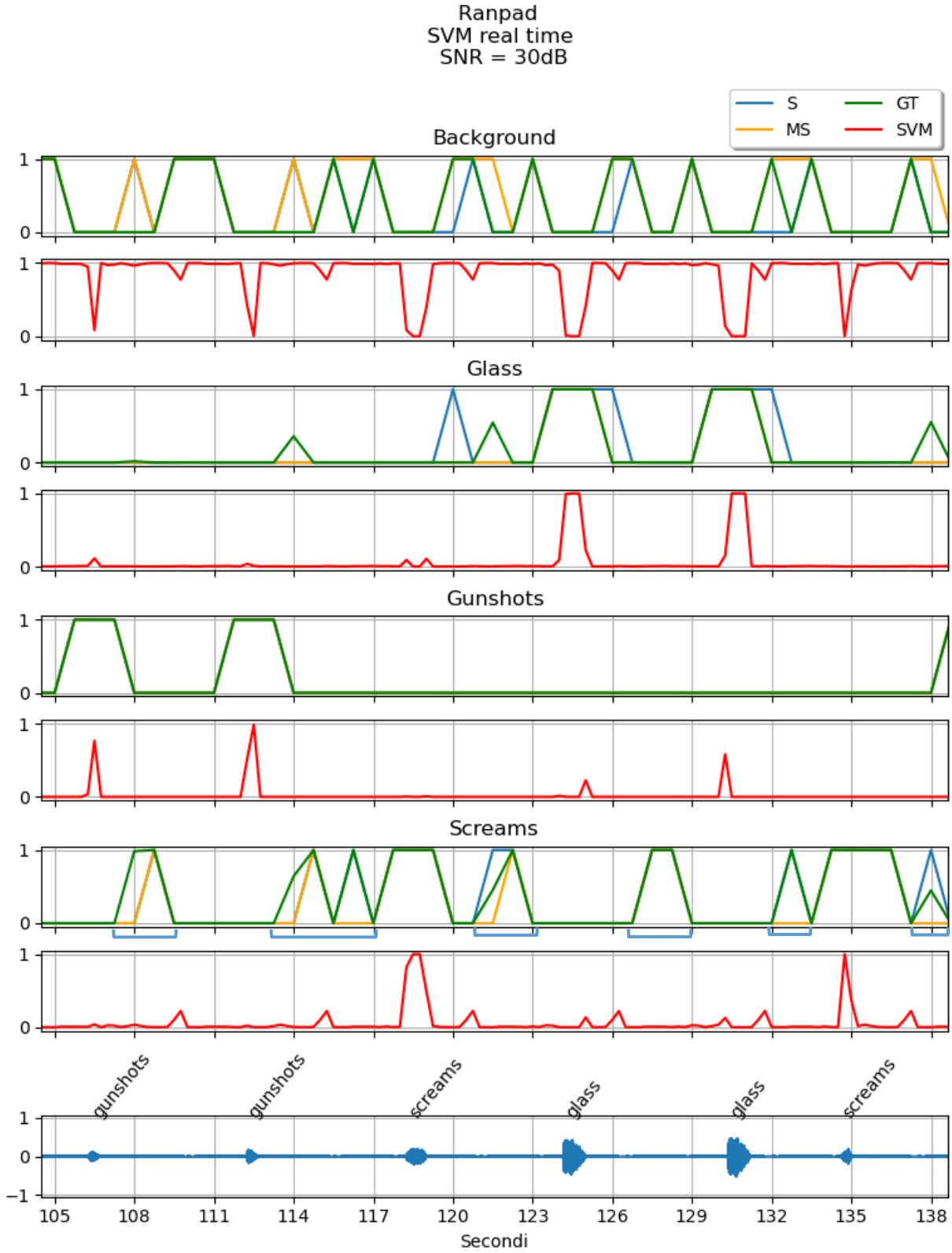


*Figura 67. Andamento delle probabilità a posteriori (ordinate, valori compresi tra 0 e 1) delle reti neurali e dell'SVM. In fondo alla dashboard la traccia audio normalizzata rispetto al valore massimo. Confronto con SNR di 30dB.*

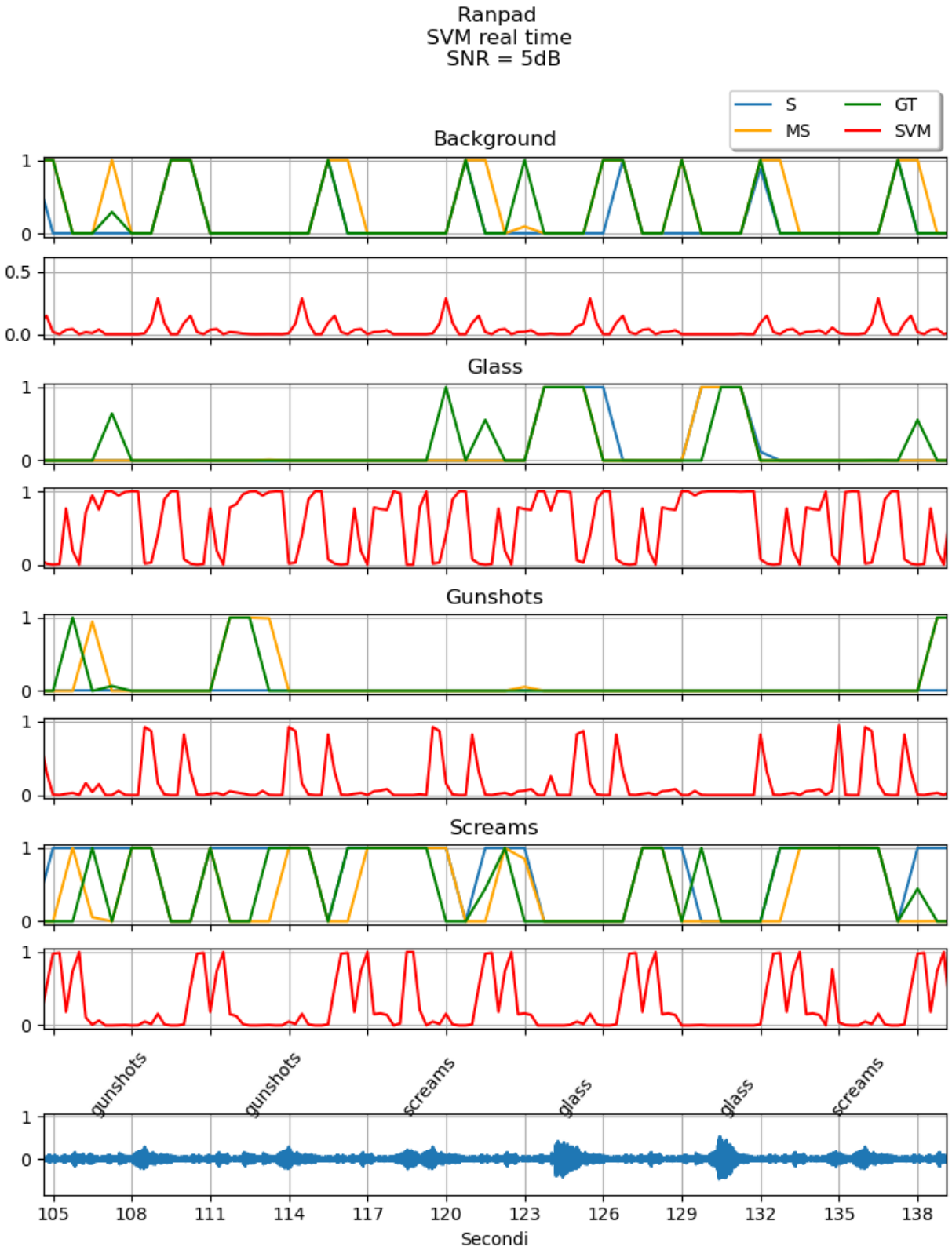


*Figura 68. Andamento delle probabilità a posteriori (ordinate, valori compresi tra 0 e 1) delle reti neurali e dell'SVM. In fondo alla dashboard la traccia audio normalizzata rispetto al valore massimo. Confronto con SNR di 5dB.*

Si è preso in considerazione l'approccio ranpad per le reti neurali e il caso real time per l'SVM. Il file audio con cui sono state estratte le istanze è l'ultima traccia della dashboard, ed è stato scelto per la tipologia di background con cui è costruito, in particolare esso contiene suoni di voci come fossero un chiacchierio di una folla o di un luogo pubblico, e si è sottolineato più volte come i modelli, soprattutto le reti neurali, siano sensibili a questa classe di suoni. Il motivo di questa decisione è quello di studiare il comportamento dei modelli nelle situazioni di stress maggiore.
Gli andamenti delle probabilità sono raggruppati per classe, ad esempio per la classe background si trova una prima traccia la quale raggruppa gli andamenti per le tre rappresentazioni tempo-frequenza, mentre la traccia sottostante mostra l'andamento legato all'SVM. La legenda in alto a destra definisce i colori delle tracce per ogni rappresentazione e modello. Per un commento di qualità migliore si è considerata solamente una sezione della dashboard, compresa tra il secondo 105 e 142 circa.

Dalla figura 67 si può osservare come l'SVM riconosca molto bene tutti gli eventi della sezione considerata, inoltre nella traccia della classe screams si possono distinguere chiaramente alcuni valori non nulli di probabilità in corrispondenza del solo background. Ciò conferma la scelta della traccia. Continuando l'analisi delle probabilità, si può riassumere dicendo che per l'SVM, in questa particolare sezione, riconosce tutti gli eventi audio d'interesse compresa un'ottima distinzione del background. Passando invece all'analisi delle probabilità per le reti neurali, si evidenzia in primis come esse siano molto più sensibili a suoni vocali, ovvero quelli provenienti dal background. Ciò è dimostrato dalla presenza di falsi positivi per la classe screams dovuti ad eventi background. Essi sono indicati con una parentesi blu al disotto della traccia in questione. Si può notare inoltre un falso positivo nella classe glass per lo spettrogramma. Per le restanti classi anche le reti neurali si comportano abbastanza bene. Ciò nonostante nessun evento d'interesse è perso, e questo è fondamentale.

Continuando l'analisi, si considera ora la dashboard per la configurazione SNR $5dB$ (figura 68). In questo caso la traccia audio presenta un compito di classificazione decisamente più arduo del caso precedente, lo si può evincere dagli eventi gunshots che non sono distinguibili ad occhio e sono totalmente sovrastati dal background, come accade anche per gli eventi screams. Al contrario si riescono ancora a vedere gli eventi glass grazie al loro transiente più pronunciato. Il classificatore SVM mostra in pieno i suoi punti deboli nei confronti di un basso valore di SNR. Si analizza quindi una classe alla volta. Per la classe background si può osservare come il valore della probabilità sia sempre inferiore a 0.5, questo significa che per la sezione presa in considerazione il classificatore non è in grado di discriminare a pieno la presenza di un evento d'interesse dal background. In riferimento alla classe glass si può osservare una smisurata presenza di falsi positivi, ovvero molte istanze non appartenenti a questa classe ma classificati come tale. Questo comportamento è ingiustificato, in quanto la maggior parte degli eventi, soprattutto quelli di background, vengono riconosciuti come eventi glass. Si dimostra ancora una volta la sensibilità dell'SVM a bassi rapporti segnale rumore. I due eventi glass presenti sono in realtà rilevati dall'SVM, ma si può notare come quest'ultimo riconosca eventi glass sia prima che immediatamente dopo le istanze corrispondenti al vero evento. In un'ottica reale, un tale comportamento causerebbe una possibile perdita degli eventi d'interesse, in quanto immersi in una marea di falsi riconoscimenti della medesima classe. Proseguendo con l'analisi, si osserva una disfatta anche per la classe gunshots, per la quale nessuna delle istanze riferite a questa classe è stata riconosciuta correttamente, ma classificate come glass, mentre i casi di riconoscimenti di eventi gunshots sono chiaramente dei falsi positivi. Un giudizio analogo può essere fornito per la classe screams, anche se alcune delle istanze appartenenti a questa classe sono state classificate in maniera adeguata. La presenza di falsi positivi è alta e si potrebbe dire giustificata per la tipologia di background.
Al contrario, la classificazione mediante le rappresentazioni tempo-frequenza in combinazione con le reti neurali è nettamente più robusta nei confronti di SNR bassi. Iniziando l'analisi dalla classe glass si può discernere facilmente

dove questi eventi sono effettivamente riconosciuti. Si nota la presenza di qualche falso positivo, ma in numero sicuramente inferiore a quanto visto per l'SVM. Gli eventi gunshots sono riconosciuti quasi tutti tranne il primissimo per il quale tutte le rappresentazioni tempo frequenza sono incerte, soprattutto lo spettrogramma secondo cui nessun evento gunshots è presente, mentre le restanti due lo riconoscono solamente per una istanza. Si può notare l'assenza di falsi positivi per questa classe in questa sezione. Chiaramente nel caso della classe screams si ripropone la sensibilità di queste tecniche rispetto questa classe e soprattutto a background di questa tipologia. Ciò è possibile dichiararlo vista la presenza di numerosi falsi positivi. Tuttavia, nessun evento screams è perduto.

# 7 CONCLUSIONI

Nel presente elaborato di tesi sono state presentate soluzioni per l'implementazione di un sistema automatico di riconoscimento di situazioni di pericolo. Si è considerato di implementare il suddetto sistema mediante una rete neurale profonda basata sull'architettura della VGG19, la quale è stata allenata a riconoscere immagini tempo-frequenza realizzate mediante tre rappresentazioni: spettrogramma, spettrogramma in scala Mel e gammatonogramma. Si è studiato in maniera approfondita la fase di allenamento della rete, dimostrando che la strategia basata sul transfer learning sia stata quella che ha garantito il miglior allenamento, rispetto alle strategie training from scratch e step learning. È stato inoltre confrontato l'approccio eventi centrati e ranpad, da cui si è evinto in base ai risultati ottenuti dal detector che l'approccio ranpad è nettamente più efficiente del rivale, in quanto numeri alla mano esso ha ottenuto un recognition rate di 85.12% contro 48.83% nel caso gammatonogramma. Quindi, nonostante in fase di allenamento l'approccio eventi centrati si sia dimostrato globalmente migliore, è necessario indiscutibilmente considerare l'approccio ranpad nell'ottica di classificazione di eventi estratti da uno stream audio in tempo reale. Sono state confrontate poi le performance ottenute in fase di allenamento dalle tre rappresentazioni tempo-frequenza, per le quali, considerando l'approccio ranpad, la rappresentazione gammatonogramma ha sicuramente riportato valori di accuracy ed errori percentuali migliori rispetto allo spettrogramma in scala Mel e allo spettrogramma classico; lo spettrogramma in scala Mel si comporta globalmente meglio rispetto allo spettrogramma. Dalle considerazioni di cui sopra si evince: utilizzare rappresentazioni che si ispirano alle caratteristiche acustiche dell'apparato uditivo, quali la scala Mel e il filtraggio gammatono, ha causato un miglioramento sensibile a fini della classificazione. Ciò è ulteriormente confermato considerando i risultati ottenuti studiando il comportamento del detector. In termini di Recognition Rate si è ottenuto 85.12% per il gammatonogramma, 84.45% per lo spettrogramma in scala Mel e 83.18% per lo spettrogramma.

In parallelo è stato costruito uno strumento per la classificazione automatica basato sul modello del Support Vector Machine, il cui scopo è stato quello di confronto rispetto i modelli costruiti con la rete neurale. Contro ogni prospettiva si è visto chiaramente che nell'implementazione del detector l'SVM ha dimostrato grandi abilità di classificazione, con un Recognition Rate pari al 90.9%. D'altro canto, si è appurata la grande sensibilità di questa tecnica al variare del valore di SNR. Quindi, nonostante la capacità di riconoscimento degli eventi d'interesse dimostrata dall'SVM, costruire un modello il più robusto possibile nei confronti dei suoni di background e del rumore in generale è una caratteristica di cui si deve assolutamente tenere conto, e da quanto visto è decisamente un punto debole di questa tecnica.

I risultati ottenuti dall'SVM suggeriscono un approccio di allenamento per le reti neurali, che non è stato considerato e andrebbe ad aggiungersi alla lista composta da eventi centrati e ranpad. Nello specifico si può pensare di costruire le immagini tempo-frequenza da fornire alla rete in fase di allenamento, come istanze estratte direttamente dagli stream audio e con l'assegnamento delle label come effettuato per il detector. Si replicherebbe così in fase di allenamento ciò che accade nella vera implementazione di un sistema automatico per sorveglianza. Questa considerazione è ulteriormente sostenuta dallo studio condotto nel detector, nel quale l'approccio ranpad si è rivelato di gran lunga migliore dell'approccio eventi centrati.

Un'ulteriore considerazione riguarda la funzione finestra con cui si è scelto di lavorare nel caso della rete neurale. La lunghezza della funzione finestra è stata fissata a 3 secondi per tutta la durata del progetto. Quindi nessuna analisi è stata compiuta considerando di variare la lunghezza, l'overlap e la tipologia di finestra.

In conclusione, si può affermare che le reti neurali, nonostante un Recognition Rate inferiore rispetto all'SVM, hanno dimostrato un comportamento costante di classificazione al variare dell'$SNR$, constatando quindi una propensione

maggiore per un’eventuale implementazione reale. Inoltre, tenendo a mente le considerazioni fatte poc’anzi, su un diverso approccio per la fase di allenamento, ci si possono aspettare buoni margini di miglioramento delle performance del detector costruito con reti neurali profonde.

# 8 BIBLIOGRAFIA


[1] M. Reiter e P. Rohatgi, «Homeland Security,» *IEEE Internet Computing,* vol. 8, n. 6, pp. 16-17, 2004.

[2] «Video Surveillance Market,» MarketsAndMarkets, 04 2020. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/video-surveillance-market-645.html?gclid=CjwKCAiAm-2BBhANEiwAe7eyFGRgWdaLmPzu6VU3I4a-YVhOZSY_cyiBSZT9u1TTPds2owxIPTu2BRoC0pYQAvD_BwE.

[3] R. Najah, «Surveillance technologies in the Covid-19 era,» *Policy center for the new south,* 06 2020.

[4] T. D. Raty, «Survey on Contemporary Remote Surveillance System for Public Safety,» *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews,* vol. 40, n. 5, pp. 493-515, 2010.

[5] N. Stavros, «Audio surveillance,» *WIT Transactions on State of the Art in Science and Engineering,* vol. 54, pp. 191-205, 2012.

[6] M. Crocco, M. Cristiani, A. Trucco e V. Murino, «Audio Surveillance: a Systematic Review,» Verona, 2014.

[7] S. Chandrakala e S. L. Jayalakshmi, «Environmental Audio Scene and Sound Event Recognition for Autonomous Surveillance,» *ACM Computing Surveys,* vol. 52, n. 3, pp. 63: 1-34, 2019.

[8] Wikipedia, «Speech recognition,» 05 01 2021. [Online]. Available: https://en.wikipedia.org/wiki/Speech_recognition#Applications.

[9] Q. H. Phan, «Audio Event Detection, Classification, and Beyoind,» , Lübeck, 2017.

[10] F. A. Everest e K. C. Pohlmann, Master Handbook of Acustic, McGraw Hill Companies, 2009.

[11] Wikipedia, «Suono,» 16 Nov 2020. [Online]. Available: https://it.wikipedia.org/wiki/Suono.

[12] A. kumar, P. Dighe, R. Singh, S. Chaudhuri e B. Raj, «Audio event detection from acoustic unit occurrence patterns,» *International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 489-492, 2012.

[13] C. Woohyun, R. Jinsang, H. David K. e K. Hanseok, «Selective Background Adaptation Based Abnormal Acoustic Event Recognition for Audio Surveillance,» *IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance,* pp. 118-123, 2012.

[14] L. Seunghyung, P. Jinuk, Sangjun Park e H. Minsoo, «Sound Event Classificarion with Feature Vector Combination for Automatic Audio-based Serveillance,» *IEEE International Conference on Consumer Electronics (ICEE),* pp. 147-148, 2016.

[15] R. Regunathan , D. Ajay e S. Paris , «AUDIO ANALYSIS FOR SURVEILLANCE APPLICATIONS,» *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics,* pp. 158-161, 2005.

[16] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio e M. Vento, «Reliable Detection of Audio Events in Highly Noisy Environments,» *Pattern Recognition Letters,* vol. 65, pp. 22-28, 2015.

[17] V. Carletti, ,. P. Foggia, ,. G. Percannella, ,. A. Saggese, ,. N. Strisciuglio e M. Vento, «Audio Surveillance Using a Bag of AuralWords Classifier,» *10th IEEE International Conference on Advanced Video and Signal Based Surveillance,* pp. 81-86, 2013.

[18] J. Dennis, ,. H. D. Tran e H. Li, «Image Feature Representation of the Subband Power Distribution for Robust Sound Event Classification,» *IEEE Transactions on Audio Speech and Language Processing,* vol. 21, n. 2, pp. 367-377, 2011.

[19] J. Dennis, H. D. Tran e H. Li, «Spectrogram Image Feature for Sound Event Classification in Mismatched Conditions,» *IEEE Signal Processing Letters,* vol. 18, n. 2, pp. 130-133, 2011.

[20] I. McLoughlin, H. Zhang, Z. Xie, Y. Song e W. Xiao, «Robust Sound Event Classification Using Deep Neural Networks,» *Ieee/Acm*

*Transactions On Audio, Speech, And Language Processing,* vol. 23, n. 3, pp. 540-552, 2015.

[21] F. Guo, D. Yang e X. Chen, «Using Deep Belief Network to capture temporal information for audio event classification,» *International Conference on Intelligent Information Hiding and Multimedia Signal Processing,* pp. 421-424, 2015.

[22] M. Z. Zaheer, Y. Kim, H.-G. Kim e S. Y. Na, «A Preliminary Study on Deep-Learning Based Screaming Sound Detection,» *5th International Conference on IT Convergence and Security (ICITCS),* pp. 1-4, 2015.

[23] G. Parascandolo, H. Huttunen e T. Virtanen, «Recurrent Neural Networks For Polyphonic Sound Event Detection In Real Life Recordings,» *International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 6440-6444, 2016.

[24] M. LIm, D. Lee, H. Park, Y. Kang, J. Oh, J.-S. Park, G.-J. Jang e J.-H. Kim, «Convolutional Neural Network based Audio Event Classification,» *KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS,* vol. 12, n. 6, pp. 2748-2760, 2018.

[25] P. Foggia, . A. Saggese, N. Strisciuglio, M. Vento e V. Vigilante, «Detecting Sounds of Interest in Roads with Deep Networks,» *Lecture Notes in Computer Science,* vol. 11752, pp. 583-592, 2019.

[26] M. Valera e S. Velastin, «Intelligent distributed surveillance systems: a review,» *IEE Proc.-Vis. Image Signal Process,* vol. 152, n. 2, pp. 192-204, 2005.

[27] H. Detmold, A. Dick, K. Falkner, D. S. Munro, A. van den Hengel e R. Morrison, «Middleware for Video Surveillance Networks,» *Proceedings of the international workshop on Middleware for sensor networks,* pp. 31-36, 2006.

[28] I. Ledakis, T. Bouras, G. Kioumourtzis e M. Skitsas, «Adaptive Edge and Fog Computing Paradigm for Wide Area Video and Audio Surveillance,» *9th International Conference on Information, Intelligence, Systems and Applications (IISA),* 2018.

[29] Wikipedia, «Artificial neural network,» 20 01 2021. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network.

[30] S. Ruder, «An overview of gradient descent optimization algorithms,» 2016.

[31] Wikipedia, «Convolutional neural network,» 03 02 2021. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[32] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[33] D. Zhao, H. Ma e L. Liu, «Event Classification For Living Environment Surveillance Using Audio Sensor Network,» *IEEE International Conference on Multimedia and Expo*, pp. 528-533, 2010.

[34] M.-W. Mak e S.-Y. Kung, «Low-Power Svm Classifiers For Sound Event Classification On Mobile Devices,» *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1985-1988, 2012.

[35] J. Portelo, M. Bugalho, I. Trancoso, J. Neto, A. Abad e A. Serralheiro, «Non-Speech Audio Event Detection,» *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1973-1976, 2009.

[36] Wikipedia, «Support - Vector Machine,» 05 02 2021. [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine.

[37] R. V. Sharan e T. J. Moir, «Comparison of multiclass SVM classification techniques in an audio surveillance application under mismatched conditions,» *Proceedings of the 19th International Conference on Digital Signal Processing*, pp. 83-88, 2014.

[38] A. Antoniou, Digital Signal Processing, McGraw-Hill, 2006.

[39] E. Sejdic, I. Djurovic e J. Jinag, «Time–frequency feature representation using energy concentration: An overview of recent advances,» *Digital Signal Processing*, vol. 13, n. 1, pp. 153-183, 2009.

[40] M. Vetterli e J. Kovacevic, Wavelets and Subband Coding, Englewood Cliffs: Prentice Hall, 1995.

[41] Wikipedia, «Window Function,» 29 01 2021. [Online]. Available: https://en.wikipedia.org/wiki/Window_function.

[42] P. Pedersen, «The Mel Scale,» *Journal of Music Theory,* vol. 9, n. 2, pp. 295-308, 1965.

[43] J. Holdsworth, I. Nimmo-Smith, R. Patterson e P. Rice, «Implementing a GammaTone Filter Bank,» 1988.

[44] A. Greco , N. Petkov, A. Saggese e M. Vento , «AReN: A Deep Learning Approach for Sound Event Recognition Using a Brain Inspired Representation,» *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 15, pp. 3610-3624, 2020.

[45] Wikipedia, «Equivalent rectangular bandwidth,» 28 01 2021. [Online]. Available: https://en.wikipedia.org/wiki/Equivalent_rectangular_bandwidth.

[46] J. O. Smith e J. S. Abel, «Bark and ERB Bilinear Transforms,» *IEEE Transactions on Speech and Audio Processing,* vol. 7, n. 6, pp. 697-708, 1999.

[47] Wikipedia, «Critical band,» 06 02 2021. [Online]. Available: https://en.wikipedia.org/wiki/Critical_band.

[48] D. Ellis, «Gammatone-like spectrograms,» 2009. [Online]. Available: https://www.ee.columbia.edu/~dpwe/resources/matlab/gammatonegram/.

[49] B. Logan, «Mel Frequency Cepstral Coefficient for Music Modeling,» *In International Symposium on Music Information Retrieval,* 2000.

[50] L. Govoni, «Matrice di confusione: cos'è e come funziona?,» [Online]. Available: https://lorenzogovoni.com/matrice-di-confusione/.

[51] L. Demortier, «Taking the Confusion out of the Confusion Matrix,» 26 07 2016. [Online]. Available: https://lucdemortier.github.io/articles/16/PerformanceMetrics.

[52] A. Greco, A. Saggese, M. Vento e V. Vigilante, «SoReNet: a novel deep network for audio surveillance applications,» *IEEE International Conference on Systems, Man and Cybernetics (SMC),* pp. 546-551, 2019.

[53] N. Takahashi, M. Gygli e L. V. Gool, «Aenet: Learning deep audio features for video analysis,» *IEEE Transactions on Multimedia*, vol. 20, n. 3, pp. 513-524, 2018.

[54] S. Soni, S. Dey e M. S. Manikandan, «Automatic Audio Event Recognition Schemes for Context-Aware Audio Computing Devices,» *Seventh International Conference on Digital Information Processing and Communications (ICDIPC)*, pp. 23-28, 2019.

[55] S. U. P. U. Stanford Vision Lab, «IMAGENET,» [Online]. Available: http://image-net.org/about-stats.

[56] C.-W. Hsu, C.-C. Chang, e C.-J. Lin, «A Practical Guide to Support Vector Classification,» 2003.