ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI INGEGNERIA

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

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRONICA

TESI DI LAUREA MAGISTRALE
IN
STATISTICS AND ARCHITECTURES FOR BIG DATA PROCESSING

"Acquisizione compressa mediante Compressed Sensing: studio del supporto ottimo per segnali non perfettamente sparsi e dell'impatto sulla individuazione di istanze anomale"

CANDIDATO	RELATORE	CORRELATORI
GABRIELE RAVAGLIA	*Prof. Ing.* MAURO MANGIA	*Prof. Ing.* ALEX MARCHIONI *Dr. Ing.* FILIPPO MARTININI *Prof. Ing.* RICCARDO ROVATTI

Anno Accademico 2023/2024

Sessione II

SOMMARIO

Introduzione

Il **campionamento dei segnali** elettrici è un processo di estrema importanza nell'ambito della elaborazione dei segnali. Un segnale, in generale, è una grandezza fisica che trasporta informazione, contenuta nella variazione della grandezza stessa in un determinato dominio (come il tempo o lo spazio). Nel mondo dell'elettronica, un segnale è una grandezza di tipo elettrica, digitale o analogica, espressa nel dominio del tempo o della frequenza, eventualmente ottenuta per conversione di un'altra grandezza fisica da parte di un sensore dedicato.

Il processo di campionamento, dunque, consiste nel convertire un segnale elettrico continuo nel tempo in una sequenza di valori discreti. Questo è fondamentale poiché l'elaborazione è eseguita da sistemi digitali che operano in modo discreto. Il **teorema di Nyquist-Shannon** formalizza il concetto di campionamento, affermando che un segnale a banda limitata può essere ricostruito perfettamente dai suoi campioni se la frequenza di campionamento è almeno il doppio della massima frequenza del segnale. Tale teorema è alla base di molte tecnologie moderne che spaziano su diversissime applicazioni, come dalla risonanza magnetica nel campo medico alla registrazione audio e video in formato digitale.

Un altro aspetto molto importante riguarda la **compressione dei dati**. Essa, è diventata essenziale a causa della grande quantità di dati generati e della necessità di trasmetterli e memorizzarli in modo efficiente. Ad esempio, una immagine ad alta risoluzione, o un video ad alta definizione, può occupare una enorme quantità di spazio di memorizzazione. La compressione, dunque, entra in gioco riducendo la quantità di dati necessari per rappresentare un segnale senza perdere (o comunque con la minor perdita possibile) la qualità percepita. Ovviamente, i benefici acquisiti, quali il risparmio di spazio di memorizzazione, efficienza di trasmissione in termini di tempo e larghezza di banda e quindi minori costi, comportano inevitabilmente anche degli svantaggi, come la perdita di qualità per compressori Lossy (presto definiti), maggiore tempo di elaborazione richiesto per le fasi di compressione e/o decompressione dei dati e scarsa compatibilità tra diversi formati.

Il **Compressed Sensing (CS)** [1], introdotto nel 2004 da Emmanuel Candès, Justin Romberg, Terence Tao e David Donoho, si distingue dagli altri metodi di compressione per la sua capacità di ricostruire segnali sparsi da un numero ridotto di campioni. Come accennato precedentemente, si possono classificare i metodi di compressione **Lossless** da quelli **Lossy**. I Lossless permettono di avere perfetta ricostruzione del segnale, senza alcuna perdita di

informazione, come i formati ZIP e PNG. Invece, i Lossy, durante la fase di compressione rimuovono alcune informazioni, non garantendo più perfetta ricostruzione, come i formati JPEG e MP3. Il CS non è intrinsecamente una tecnica Lossless, ma può essere utilizzato in modo tale da ottenere una ricostruzione del segnale praticamente indistinguibile dall'originale.

Dunque, l'interesse per il CS è principalmente dovuto alla sua peculiarità di consentire una compressione del segnale a basso costo e contemporaneamente di trasferirne la complessità dalla fase di codifica a quella di decodifica. Tale aspetto, meglio definito nel Capitolo 1, si distingue dagli schemi di compressione tipici, dove la fase più complicata e dispendiosa in termini di energia è quella di codifica. Per questo motivo, il CS è specialmente preferito in determinate applicazioni, tra cui la dispersione elettromagnetica inversa, il monitoraggio della salute strutturale e l'elaborazione delle immagini. Un'altra area in cui il CS è particolarmente utile è la progettazione dei nodi delle Body Area Network (BAN), che mirano ad acquisire biosignali in modo efficiente. Questo è stato già dimostrato per i segnali elettrocardiografici (ECG) [2], [3], elettromiografici (EMG) [4] ed elettroencefalografici (EEG) [5].

Come accennato, ad una fase di compressione poco dispendiosa corrisponde una fase di decompressione piuttosto pesante. Uno dei primi metodi proposti in letteratura per affrontare la decodifica è il **Basis Pursuit**, che esegue la ricostruzione del segnale risolvendo un problema di programmazione lineare. Il **Basis Pursuit con DeNoising (BPDN)**, invece, tiene conto anche dell'incertezza dovuta al rumore [6]. Successivamente sono stati sviluppati decodificatori più sofisticati [7], [8] che sfruttano assunzioni sulla classe del segnale e sulle perturbazioni per garantire prestazioni adeguate quando la ricostruzione del segnale deve essere ottenuta utilizzando hardware con capacità computazionali limitate. In questo contesto, algoritmi iterativi come l'**Orthogonal Matching Pursuit (OMP)** [9] ed il **Compressive Sampling Matching Pursuit (CoSaMP)** [10] sono spesso utilizzati per la loro bassa complessità computazionale [11]. Recentemente, è stato preso in considerazione anche l'uso delle **Deep Neural Network (DNN)** per la decodifica CS, principalmente per applicazioni di immagini o video [12]–[15], anche se alcuni lavori mirati ai segnali biomedici possono essere trovati in [16], [17].

Tutti i framework CS menzionati condividono una struttura di codifica a bassa complessità e propongono diversi approcci di decodifica che mirano a ridurre il più possibile il numero di parole digitali che rappresentano una singola istanza di input. In altre parole, l'obiettivo è quello di aumentare il rapporto di compressione, in modo da ridurre ulteriormente la complessità

computazionale del codificatore, così da avere il numero minimo possibile di parole digitali da trasmettere. In generale, ne segue un trade-off: maggiore sarà il livello di compressione, maggiore sarà l'informazione utile scartata.

Per tutti questi decodificatori CS, l'ipotesi fondamentale per consentire una corretta ricostruzione è che la classe dei segnali di input sia **sparsa**, ovvero che ogni istanza del segnale, in una base di sparsità appropriata, possa essere rappresentata da un numero di coefficienti non nulli molto inferiore alla dimensione del segnale stesso. Questa assunzione di sparsità può essere espressa in un altro modo introducendo il concetto di **supporto** del segnale, cioè il sottoinsieme delle posizioni corrispondenti ai coefficienti non nulli nella rappresentazione sparsa. Il lavoro svolto dagli autori (d'ora in avanti indicati sempre con "*gli Autori*", salvo diversa specificazione) dell'articolo "Deep neural oracles for short-window optimized compressed sensing of biosignals" [17] per l'identificazione del supporto, apre la strada alla definizione di una procedura di decodifica alternativa in cui la ricostruzione del segnale si può considerare composta da due fasi principali: la prima fase utilizza una DNN addestrata, chiamata Support Oracle (SO), per l'identificazione del supporto, mentre la seconda fase utilizza il supporto individuato nella fase precedente per recuperare il segnale di ingresso con una semplice operazione di algebra lineare, ovvero la pseudo-inversa. Sempre gli stessi Autori hanno poi sviluppato un altro articolo [18] in cui mostrano i vantaggi introdotti dalla decodifica a due step e la problematica legata all'utilizzo del CS su segnali biomedici (ECG) realistici e quindi non perfettamente sparsi. Per valutare ciò, però, la DNN richiede una nuova fase di addestramento, che necessita di dati ed etichette, i quali devono essere definiti. In particolare, dato un segnale reale in input, questo viene suddiviso in finestre, denominate istanze, e per ciascuna di esse si valuta il supporto utilizzando un algoritmo (Greedy), che verrà descritto in dettaglio successivamente. Tuttavia, tale algoritmo risulta particolarmente oneroso in termini di calcolo.

Dunque, questa tesi si propone di rivisitare il lavoro svolto dagli Autori citati introducendo una nuova metodologia per la valutazione del supporto, basata su un determinato livello di energia rispetto a quella totale di ogni istanza. L'elaborato si compone sostanzialmente di due capitoli. Nel Capitolo1 si affronterà la teoria legata al CS, mostrando le differenze tra l'algoritmo usato dagli Autori e quello proposto in questo lavoro. Si concluderà con l'allenamento della stessa rete neurale con il semplice scopo di dedurre il supporto nel medesimo modo, arrivando così al TSOC (Trained Support Oracle for Compressed signal). Il Capitolo 2, invece, introdurrà il concetto di anomalia e di come essa possa influenzare i segnali biomedici, richiedendo una

tempestiva rilevazione, soprattutto nell'ambito del monitoraggio. In particolare, si utilizzerà il modello allenato per definire un processo di Anomaly Detection e valutare la sua capacità di distinguere tra segnali normali e anomali. La figura 1 introduce lo schema a cui si farà riferimento, mettendo in evidenza la fase di acquisizione e compressione, la fase di trasmissione e ricezione ed infine, la fase di ricostruzione del segnale con l'ausilio del TSOC.

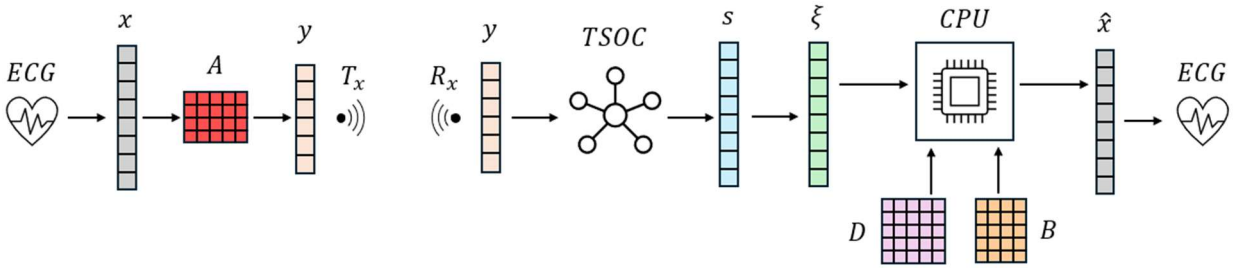


Figura 1: schema di encoding-decoding con fase di trasmissione ed implementazione del TSOC. A è la matrice di Sensing, D è il Dizionario e B deriva dalle prime due. Nel Capitolo 1 tutti questi elementi saranno discussi ampiamente.

Capitolo 1

COMPRESSED SENSING

1.1 Obiettivo

Supposto di avere un segnale biomedico ECG campionato, suddiviso in finestre x non sovrapposte e tutte costituite dallo stesso numero di campioni $n = 128$, quindi tutte della stessa lunghezza ($x \in \mathbb{R}^n$), l'obiettivo che ci si pone è quello di ottenere una rete neurale che data una misura compressa $y \in \mathbb{R}^m$ in ingresso, con $m < n$, indovini il supporto ottimo della rappresentazione sparsa del segnale per la ricostruzione della finestra $\hat{x}$. Tutti questi elementi saranno poi ripresi nel seguito e spiegati con dettaglio. Tuttavia, la rete neurale necessita di una fase di allenamento che prevede già la conoscenza dei supporti associati alle varie istanze, i quali devono quindi essere trovati. Gli Autori sfruttano il lavoro svolto nell'articolo [17] per determinare i supporti mediante il *Support Oracle* (SO), che funziona molto bene nel caso di segnali ECG ideali, ovvero per i quali si possa assumere perfetta sparsità. I segnali reali, invece, non sono perfettamente sparsi e quindi il SO non è in grado di stimare in modo ottimale il supporto delle istanze. Per affrontare il problema, gli Autori propongono un algoritmo, chiamato Greedy, per determinare i supporti, ma tale algoritmo, meglio descritto nel seguito, risulta essere computazionalmente oneroso [18]. Pertanto, sarà proposto un modo alternativo per determinare tali supporti.
Dunque, prendendo come base di partenza l'articolo "Deep Neural Oracle With Support Identification in the Compressed Domain", nel capitolo che segue, verrà affrontata la teoria alla base del Compressed Sensing, analizzando gli aspetti essenziali che caratterizzano questo lavoro. Come già accennato, la fase più complessa non è la compressione, ma la decompressione, per cui verrà mostrato come ottenere una fase di decodifica più semplice ed efficiente rispetto agli altri Decoder CS, ad esempio, come il BP.

1.2 Introduzione al Compressed Sensing

Si è detto che il CS è basato sull'*ipotesi di sparsità* del segnale. In realtà, il principio di sparsità non è l'unica condizione, infatti, la ricostruzione del segnale è possibile grazie anche al *principio di incoerenza*. Un segnale è detto ***sparso*** se può essere rappresentato da un numero relativamente piccolo di coefficienti non nulli in una base appropriata, detta base di sparsità.

Formalmente, un segnale campionato con n campioni $x \in \mathbb{R}^n$ è k-sparso se ha al massimo k componenti non nulli:

$$\|x\|_0 \leq k \tag{1.1}$$

dove $\|\cdot\|_0$ denota la norma L_0 che conta il numero di componenti non nulli di x. La compressione del segnale di ingresso x avviene mediante una semplice operazione lineare, ovvero moltiplicando x per una matrice $A \in \mathbb{R}^{m \times n}$, con $m < n$, detta ***matrice di Sensing*** o ***di compressione***. Per cui, l'istanza compressa $y \in \mathbb{R}^m$ è data da:

$$y = A(x + \tau) \tag{1.2}$$

dove τ rappresenta il rumore e le non idealità del sistema di acquisizione. Per cui A costituisce formalmente l'***Encoder***, mentre il rapporto di compressione (CR) è definito come n/m.
Secondo il teorema di Nyquist-Shannon, per ricostruire perfettamente un segnale a banda limitata, la frequenza di campionamento (f_c) deve essere almeno il doppio della massima frequenza del segnale (f_s), per cui $f_c \geq 2f_s$. Il CS permette di ricostruire un segnale sparso con un numero di campioni molto inferiore a quello richiesto dal teorema di Nyquist-Shannon e questo è possibile proprio grazie alla combinazione di sparsità ed incoerenza. Il ***principio di incoerenza*** è una condizione che deve essere soddisfatta dalla matrice di sensing A se si vuole ottenere una ricostruzione accurata. Una delle condizioni più comuni è la ***Restricted Isometry Property (RIP)*** la quale richiede che tutte le sottomatrici di A di dimensione k siano ortogonali:

$$(1 - \delta_k)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_k)\|x\|_2^2 \tag{1.3}$$

per ogni x k-sparso, dove δ_k è una costante che dipende da k.

La ricostruzione del segnale $\hat{x}$ è eseguita dal ***Decoder***, il quale cerca di recuperare la ***rappresentazione sparsa*** del segnale x, o per lo meno la sua migliore approssimazione. Supposto che x possa essere rappresentato nel dominio sparso come:

$$x = D\xi \tag{1.4}$$

dove $\xi \in \mathbb{R}^n$ è la rappresentazione sparsa ed è un vettore sparso, mentre $D \in \mathbb{R}^{n \times n}$ è una matrice ortogonale, tipicamente chiamata *Dizionario* e che porta ξ dal dominio sparso a quello del segnale x.

Osservazione: la matrice D è scelta in base al dominio in cui il segnale x è noto per essere sparso. Ad esempio, se il segnale è sparso nel dominio delle frequenze, allora D è una matrice di trasformazione di Fourier. Se, invece, il segnale è sparso nel dominio del coseno, allora D è una matrice di trasformazione del coseno discreto (DCT). La scelta di D, dunque, è strettamente legata alle caratteristiche del segnale stesso.
Tornando al problema di ricostruzione, con la nuova definizione di x, la (1.2) può essere espressa come:

$$y = A(x + \tau) = A(D\xi + \tau) = B\xi + A\tau \tag{1.5}$$

dove $B = AD$.
Il Basis Pursuit con DeNoising (BPDN), come accennato, è stato uno dei primi metodi di decodifica presentati in letteratura e questo metodo consiste nel risolvere il problema di ottimizzazione descritto dalla seguente espressione:

$$\hat{\xi} = \underset{\xi}{\operatorname{argmin}} \|\xi\|_1 \qquad soggetto\ a \qquad \|y - B\xi\|_2 \leq \gamma \tag{1.6}$$

dove $\gamma \geq 0$ è proporzionale a τ. Il BPDN, quindi, cerca di trovare la miglior rappresentazione sparsa tra le infinite soluzioni di un sistema mal posto. Questo problema è combinatorio e, in teoria, richiederebbe di testare tutte le possibili soluzioni, rendendolo un problema NP-hard. Nonostante ciò, la soluzione $\hat{\xi}$ viene trovata lo stesso grazie al principio di incoerenza. Pertanto, la ricostruzione del segnale è ottenuta come:

$$\hat{x} = D\hat{\xi} \tag{1.7}$$

Infine, per la valutazione della qualità di ricostruzione raggiunta, si può utilizzare l'**SNR (Signal to Noise Ratio)** o **RSNR (Reconstructed SNR)**, definito come:

$$RSNR = 20\log_{10}\left(\frac{\|x\|_2}{\|x-\hat{x}\|_2}\right) = \left(\frac{\|x\|_2}{\|x-\hat{x}\|_2}\right)_{dB} \tag{1.8}$$

Ulteriori aspetti che valgono la pena di essere approfonditi riguardano la matrice di Sensing A e la dimensione m, ovvero il numero di campioni del segnale compresso y.
Affinché il CS lavori bene, A deve essere scelta e disegnata in modo opportuno. Essendo questo lavoro una revisione di quanto già svolto nell'articolo citato e siccome si vuole eseguire una comparazione dei risultati, A sarà la stessa utilizzata dagli Autori. Per quanto riguarda m, invece, affinché la ricostruzione sia possibile, il suo valore non deve essere minore di $2k$ (con k elementi non nulli di x), altrimenti, due segnali x k-sparsi possono essere mappati nello stesso segnale compresso y. In generale, la teoria del CS identifica m attraverso la seguente relazione:

$$m = \mathcal{O}\left(k\log\left(\frac{n}{k}\right)\right) \tag{1.9}$$

Con questo paragrafo si è voluto mostrare prevalentemente il principio di funzionamento del CS, mettendo in evidenza come effettivamente la complessità dell'Encoder sia molto minore rispetto a quella del Decoder, che richiede la risoluzione di un problema di ottimizzazione.

1.3 Supporto

Dato un vettore x n-dimensionale e k-sparso, con ξ la sua rappresentazione sparsa, sempre n-dimensionale, si definisce ***supporto di ξ*** il vettore $s \in \{0,1\}^n$ tale per cui l'elemento $s_i = 1$ se $\xi_i \neq 0$, altrimenti $s_i = 0$. Quindi, gli "uni" presenti dentro s indicano le posizioni in cui gli elementi di ξ sono non nulli.
Come dimostrato dagli Autori nel loro lavoro, se si conosce il supporto di ξ, si ottiene un'enorme vantaggio computazionale, rendendo la ricostruzione del segnale, e quindi il Decoder, un'operazione molto più semplice rispetto alla soluzione adottata dal BP o BPDN. Infatti, richiamando la (1.5) ed applicando il concetto di supporto, è possibile definire la seguente espressione:

$$y = B_{|s}\xi_{|s} + A\tau \tag{1.10}$$

dove $\cdot|_s$ definisce la limitazione ad s. Ad esempio, la notazione $B_{|s}$ (da leggere come B limitato ad s) significa che si stanno considerando solo le colonne di B corrispondenti alle posizioni in cui s ha un 1.
A questo punto, la ricostruzione della rappresentazione sparsa $\hat{\xi}$ è espressa dalla relazione:

$$\hat{\xi}_{|s} = B_{|s}^{\dagger}(y - A\tau) = B_{|s}^{\dagger}y - B_{|s}^{\dagger}A\tau = B_{|s}^{\dagger}y - e \tag{1.11}$$

dove $\cdot^{\dagger}$ rappresenta l'operazione di pseudo-inversione di Moore-Penrose, mentre e rappresenta l'errore di ricostruzione della rappresentazione sparsa.
La ricostruzione di $\xi_{|s}$ è dunque una operazione molto semplice, in quanto $B_{|s}$ è una matrice "alta", ovvero con più righe che colonne, e questo è garantito dal fatto che $k < m$. Quindi, supposto di avere $\tau = 0$, per ogni $y = B_{|s}\xi_{|s}$ esiste un'unica controimmagine $\xi_{|s}$, facilmente ottenibile moltiplicando y per $B_{|s}^{\dagger}$. Viceversa, quando $\tau > 0$, la controimmagine non sarà esattamente $\xi_{|s}$, ma sarà qualcosa che si discosterà da essa di una quantità e, come mostrato dalla (1.11), motivo per il quale si deve limitare quanto più possibile l'errore di ricostruzione della rappresentazione sparsa.

1.3.1 Determinazione del supporto mediante Algoritmo Greedy

Come già accennato nel paragrafo 1.1, il SO non può essere utilizzato su segnali reali, ovvero non perfettamente sparsi, chiamati anche *comprimibili*. La causa è legata al fatto che qualunque possibile definizione di supporto della rappresentazione sparsa causerebbe una perdita sostanziale dell'informazione contenuta nel segnale x, ottenendo una ricostruzione $\hat{x}$ con un elevato errore. Pertanto, la (1.4) per segnali comprimibili diventa:

$$x = D\xi + x_d \tag{1.12}$$

dove x_d contiene tutti quei dettagli del segnale che sono di minore interesse. Di conseguenza, la (1.11) diventa:

$$\hat{\xi}_{|s} = B_{|s}^{\dagger}[y - A(x_d + \tau)] = B_{|s}^{\dagger}y - B_{|s}^{\dagger}A(x_d + \tau) = B_{|s}^{\dagger}y - e \tag{1.13}$$

Ovviamente, la scelta del supporto è di estrema importanza al fine di limitare l'errore e.
Per affrontare questo problema, si deve introdurre una politica di selezione che distingui gli elementi di x utili per la ricostruzione, dagli elementi x_d di minore interesse, quindi trascurabili. Come espresso dalla (1.13), il termine di errore dipende sia da x_d che da τ, per cui, avere un supporto s costituito da tanti uni significa tenere in considerazioni tanti elementi di x, che significa ridurre il contributo di x_d in e, ma che significa anche trattenere più rumore e non idealità, quindi incrementare il contributo di τ in e. D'altro canto, considerare pochi elementi di x significa trattenere meno rumore, ma significa anche incrementare lo scarto di informazione x_d. Dunque, esiste un trade-off tra la quantità di rumore che si trattiene ed il numero di componenti utili che si vogliono tenere in considerazione. Gli Autori affrontano questo trade-off utilizzando una politica di selezione che mira ad ottenere il supporto come quello che garantisce il massimo RSNR per quella istanza. In particolare, il supporto ottimo s^{ott} dovrebbe essere trovato risolvendo il seguente problema di ottimizzazione:

$$s^{ott} = \underset{s\in\{0,1\}^n}{\operatorname{argmax}} \frac{\|x\|_2}{\left\|x - D_{|s}\hat{\xi}_{|s}\right\|_2} \tag{1.14}$$

che risulta essere di tipo combinatorio, ovvero, tutte le possibili 2^n soluzioni dovrebbero essere testate. Per questo motivo, gli Autori hanno proposto un algoritmo Greedy in grado di ottenere una buona approssimazione di s^{ott} in n passi al più.
L'algoritmo è così composto:

1. Per ogni finestra, tutti gli elementi della rappresentazione sparsa ξ di x sono ordinati in ordine decrescente in base alla loro ampiezza in valore assoluto, ottenendo ξ^{ord};
2. Il vettore s è inizializzato con tutti zeri e si inizia a considerare un elemento alla volta della rappresentazione sparsa ordinata ξ^{ord}. La posizione in s corrispondente alla posizione in ξ dell'elemento selezionato da ξ^{ord} viene posta ad 1;
3. Dall'elemento selezionato si calcola il segnale ricostruito $\hat{x}$ secondo la (1.7) dove la matrice D viene limitata alla versione aggiornata di s;
4. Applicando la (1.8) si valuta il valore di RSNR e si salva;
5. A questo punto, i passaggi 2, 3 e 4 sono iterati per ogni elemento di ξ^{ord};
6. Concluse le iterazioni, il massimo valore di RSNR è selezionato, assieme al supporto che lo ha definito, ottenendo così s^{ott}.

Si può facilmente intuire come questa procedura abbia come unico difetto la valutazione del valore di RSNR ripetuta al più n volte per ogni singola finestra.

1.3.2 Determinazione del supporto basato sul livello di energia

Al fine di rendere l'operazione di ricerca dei supporti ottimi più leggera in termini computazionali, un approccio alternativo si può basare sulla valutazione dell'energia del segnale che si vuole considerare. Intuitivamente, considerare il 100% dell'energia di una finestra x, significa tenere tutte le sue componenti, quindi $x_d = 0$. Viceversa, volendo tenere un livello di 95% di energia di x, significa scartare alcune delle sue componenti, per cui $x_d \neq 0$. Ovviamente, a causa del trade-off tra x_d e τ, non è possibile tenere un livello di energia pari al 100%. Dunque, occorre determinare in primis il livello di energia che permette di definire un supporto per il quale si ottiene subito il massimo RSNR per quella finestra. In secondo luogo, bisogna definire una politica di selezione delle componenti di x per ottenere il desiderato livello di energia, ovvero quali sono le componenti da tenere e quali quelle da scartare.

Osservazione: essendo la matrice di trasformazione D una matrice ortogonale, parlare di energia di x o di energia di ξ è la stessa cosa.

Siccome questa metodologia si vuole mantenere quanto più possibile coerente con quanto eseguito dall'algoritmo Greedy, la politica di selezione degli elementi sarà la stessa, ovvero, in accordo al primo step dell'algoritmo, gli elementi di ξ vengono ordinati in ordine decrescente in base al valore assoluto dell'ampiezza di ciascun suo campione, ma considerando solo i primi $p < n$ elementi che garantiscono il desiderato livello di energia. In particolare, l'algoritmo è così composto:

1. Per ogni finestra, tutti gli elementi della rappresentazione sparsa ξ di x sono ordinati in ordine decrescente in base alla loro ampiezza in valore assoluto, ottenendo ξ^{ord};
2. Il vettore s è inizializzato con tutti zeri e si inizia a considerare un elemento alla volta della rappresentazione sparsa ordinata ξ^{ord}. La posizione in s corrispondente alla posizione in ξ dell'elemento selezionato da ξ^{ord} viene posta ad 1;

3. Dall'elemento selezionato si calcola il livello di energia ottenuto e lo si confronta con quello che si vorrebbe ottenere, ad esempio il 95%;
4. I punti 2 e 3 sono iterati fintanto che non si raggiunge il livello di energia scelto, ottenendo così il supporto ottimale s^{ott}.

Segue che l'algoritmo proposto possiede un grado di libertà, ovvero la scelta del livello di energia da raggiungere che quindi deve essere determinato.

1.3.3 Confronto tra i due algoritmi

L'algoritmo Greedy, per ogni finestra, cerca il supporto come quel vettore che massimizza il valore di RSNR, operazione ripetuta n volte, ovvero per ogni elemento di ξ^{ord}. Iterare tale processo fino a considerare tutti gli elementi della rappresentazione sparsa ordinata, equivale a prendere il 100% dell'energia della rappresentazione sparsa del segnale, avendo quindi $x_d = 0$. Per questo motivo, superato un certo numero di elementi presi in considerazione, il valore di RSNR tende a diminuire siccome i nuovi elementi aggiunti contribuiscono solo introducendo del rumore. Tale andamento è ben mostrato dalla seguente figura (fig.2) per diversi rapporti di compressione CR, quindi per diversi m.

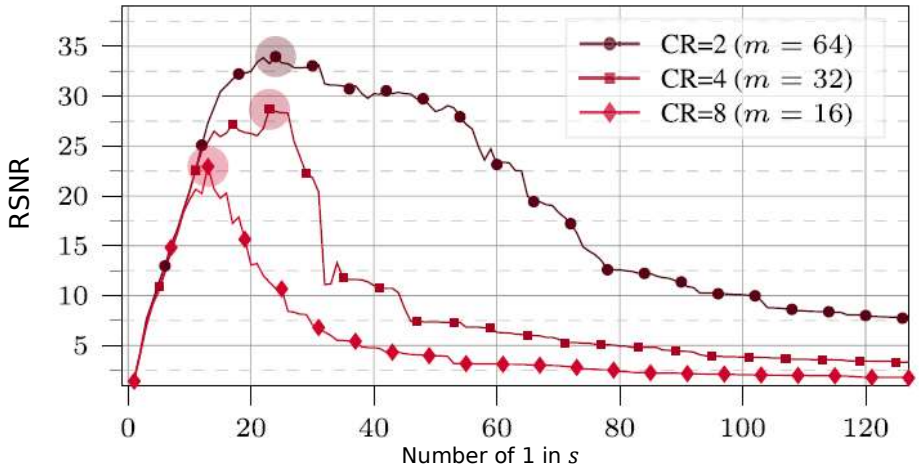


Figura 2: andamento dei valori di RSNR per diversi rapporti di compressione, ottenuto mediante algoritmo Greedy. La figura è stata presa dal paper di riferimento degli Autori [18]. Si osservi il fondo scala essere pari a $n = 128$.

L'algoritmo proposto, invece, non ha bisogno di valutare tutti gli elementi di ξ^{ord} prima di determinare qual è il supporto ottimale per quella data finestra x. Questo perché l'algoritmo si ferma non appena è stato raggiunto il livello di energia desiderato, per cui il numero di iterazioni per ogni istanza sarà $p \ll n$. Tale aspetto introduce un'enorme vantaggio in termini

computazionali. D'altro canto, il grado di libertà sulla scelta del livello di energia da tenere in considerazione introduce un costo implementativo. Infatti, per cercare di raggiungere le stesse performance in termini di RSNR raggiunti dall'algoritmo Greedy, più livelli di energia devono essere considerati. In sostanza, uno stesso grafico simile a quello illustrato nella fig.2 deve essere imitato, dove però, al posto di considerare il numero di uni presenti nel supporto, si considera il livello di energia scelto. Tale risultato è mostrato nella figura sottostante (fig.3).

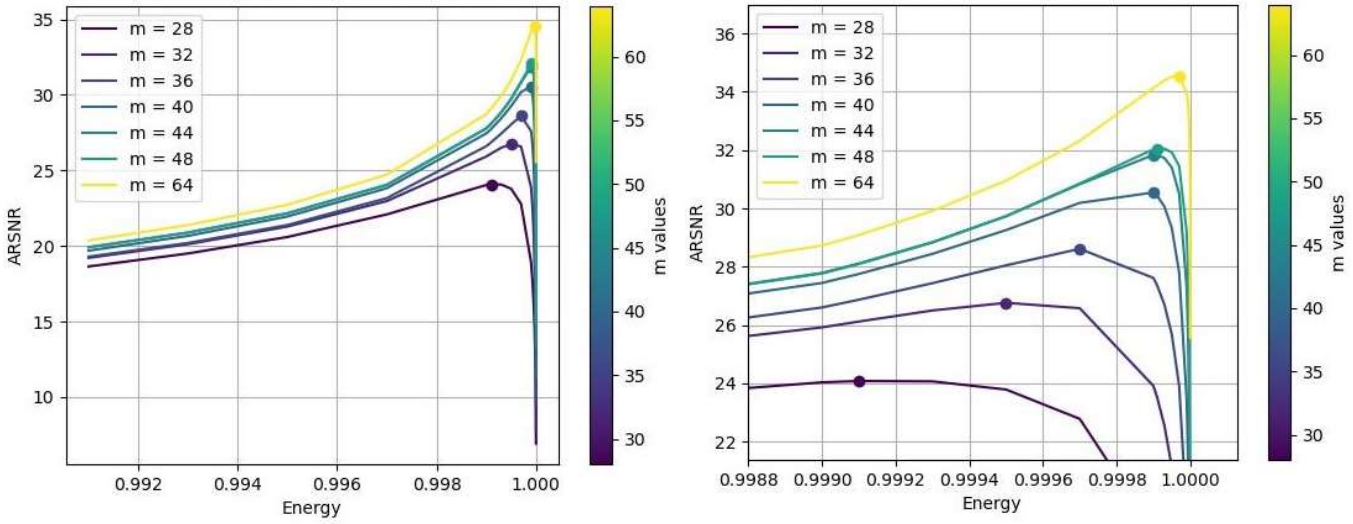


Figura 3: andamento dell'ARSNR al variare dell'energia di riferimento considerata, per diversi livelli di compressione m. Il grafico a sinistra mostra l'intero intervallo considerato, mentre il grafico a destra evidenzia la zona in cui si presentano i massimi di ARSNR, indicati dal pallino.

Per diversi valori di energia richiesta, si ottengono differenti valori di ARSNR (Average RSNR). Dal grafico, è importante notare la presenza di un picco di massimo per ogni livello di compressione m. Questi picchi, rappresentati da un puntino, si concentrano tra il 99,9% e il 100% di energia. Volendo ulteriormente rimarcare la presenza del trade-off tra gli elementi di scarto x_d e il rumore/non idealità del sistema di acquisizione τ, man mano che si considerano sempre più elementi, si considera anche più rumore, fintantoché il rumore prevale, abbattendo notevolmente il valore di RSNR. A questo punto, una volta definito il livello di compressione desiderato m, per determinare i supporti ottimali per le istanze si considera il livello di energia corrispondente al picco di ARSNR di quella curva.

Dunque, dopo aver stabilito il livello di energia che consente di ottenere valori di RSNR comparabili a quelli ottenuti con l'algoritmo Greedy, si può procedere con la fase successiva: l'allenamento della rete neurale.

1.4 Allenamento della rete neurale

In questo paragrafo verrà affrontato l'argomento relativo all'allenamento della rete neurale che si andrà poi ad utilizzare per la stima dei supporti, necessari per la fase di decoding. Infatti, fino ad ora, si è affrontato il problema legato alla creazione del dataset, che è una raccolta di dati strutturati. In particolare, un dataset è composto da due elementi principali: i **dati** e le **etichette** (in inglese **labels**). I dati rappresentano le informazioni "grezze" che la rete neurale utilizzerà per apprendere, mentre le labels sono le risposte corrette associate a ciascun dato, le quali guidano il processo di apprendimento, detto, in questo caso, *supervisionato* (supervised). Se non ci fossero le labels, allora il processo sarebbe detto *non supervisionato* (unsupervised). Ad ogni modo, in questo contesto i dati sono le misure, ovvero i segnali compressi y di dimensione m, mentre le labels sono i supporti ottimali associati ad ogni y. Dunque, una volta ottenute le misure con la fase di encoding e affrontato il problema della determinazione dei supporti per segnali comprimibili con l'algoritmo proposto, il dataset è ora completo e pronto all'uso.

1.4.1 Introduzione alla rete neurale utilizzata

La rete neurale che viene utilizzata dagli Autori è quindi quella che deve essere replicata al fine di poter confrontare nel modo più oggettivo possibile le differenze di performance introdotte dai due algoritmi studiati. Così facendo, si è in grado di capire se l'algoritmo proposto, che introduce un vantaggio implementativo, risulti essere equivalente o comunque equiparabile all'algoritmo Greedy usato dagli Autori. La struttura neurale a cui si fa riferimento è riportata nella figura 4. Tale struttura, chiamata **modello**, è composta sostanzialmente da cinque livelli pienamente connessi (tipicamente detti in inglese: fully connected layers). Il primo è detto *input layer* ed è composto da m neuroni; i livelli centrali sono chiamati *hidden layer*s e sono costituiti da $2n$ neuroni, eccetto per il quarto livello che ne ha solo n; l'ultimo livello è chiamato *output layer* ed è composto anch'esso da n neuroni. Per ogni layer è presente una **funzione di attivazione**. In particolare, si utilizza la ReLu (Rectified Linear Unit) per i primi tre e la Sigmoind per l'ultimo layer di output.
Per completezza, si definisce **neurone** [19] una unità di calcolo in grado di:

- *Ricevere gli input*, ovvero segnali provenienti da altri neuroni o direttamente dai dati di input;

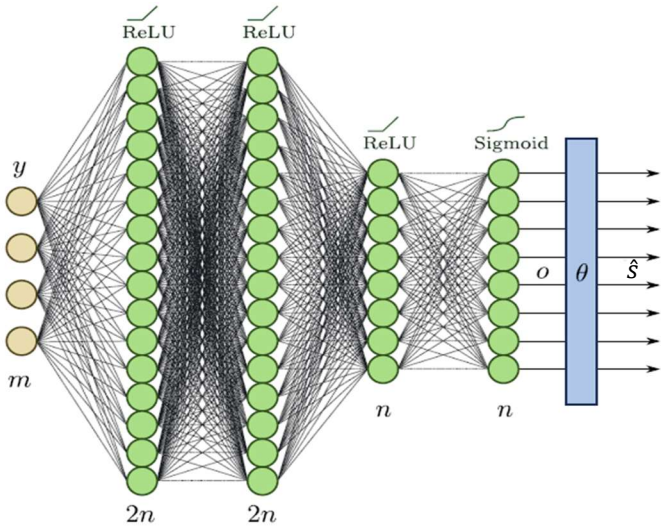


Figura 4: struttura del modello utilizzata dagli Autori e replicata in questo elaborato. L'immagine è stata presa sempre dal paper di riferimento [18].

- *Eseguire il calcolo della somma ponderata*, ovvero tutti gli input ricevuti sono moltiplicati per dei pesi e poi sommati insieme. I pesi sono dei parametri che la rete apprende durante l'addestramento;
- *Aggiungere il Bias*, ovvero alla somma ponderata viene aggiunto un ulteriore termine chiamato Bias;
- *Applicare la funzione di attivazione*, ovvero la somma ponderata viene passata ad una funzione di attivazione che introduce non-linearità nel modello, permettendo così alla rete di apprendere e rappresentare relazioni complesse;
- *Produrre l'output*, il quale corrisponde con il risultato della funzione di attivazione.

È importante sottolineare anche la rilevanza del ruolo della funzione di attivazione, in quanto essa decide se un neurone deve essere attivato o meno, influenzando così il flusso di informazioni attraverso la rete. Come accennato, nella rete utilizzata dagli Autori, le funzioni di attivazione adottate sono [20]:

- **ReLu (Rectified Linear Unit):** definita dalla relazione sottostante, essa pone a zero tutto ciò che è negativo, mentre lascia invariato tutto ciò che è positivo:

$$f(x) = \max(0, x) \tag{1.15}$$

- **Sigmoind:** definita dalla relazione sottostante, essa normalizza l'elemento x affinché il risultato o appartenga all'intervallo $[0,1]$:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.16}$$

A questo punto, sapendo che il layer di output fornisce un vettore n-dimensionale aventi come valore uno qualsiasi compreso tra 0 ed 1, al fine di ricondursi ad un vettore composto solo da zeri e da uni, quindi al fine di trovare il supporto stimato dalla rete $\hat{s}$, al vettore o si applica una sogliatura ϑ. Tale soglia è stata determinata empiricamente dagli Autori nel paper [18] e risulta essere pari a 0.5, ma i risultati ottenuti suggeriscono di usare 0.4 per avere performance leggermente migliori. Dunque, una volta definito il modello così descritto e stabilito la sogliatura in modo da ottenere il supporto stimato $\hat{s}$, non resta che allenare la rete.

1.4.2 Procedura di allenamento

Si è detto che per allenare la rete occorre un dataset, ma questo non basta. Infatti, oltre al dataset e al modello, per allenare una rete occorrono anche una **funzione di Loss** e un **algoritmo di allenamento**. Procedendo con ordine, si parte definendo il dataset, per cui, preso un m:

1. I dati di input y m-dimensionali sono ottenuti applicando la (1.5);
2. Le labels (supporti) sono ottenute applicando l'algoritmo proposto considerando il livello energetico che massimizza l'ARSNR per quello specifico rapporto di compressione;

A questo punto, il dataset, composto da due milioni di finestre, è suddiviso in due porzioni, una usata effettivamente per l'allenamento (95% del dataset), chiamata *training set*, e l'altra usata per la valutazione delle performance (restante 5% del dataset), chiamata *test set*.

Il modello, ovvero l'architettura della rete neurale, è già stato descritto nel sottoparagrafo precedente, ma è importante evidenziare che, siccome la struttura del modello dipende anche dal numero di input m, risulta evidente che per ogni livello di compressione si ha un numero diverso di neuroni nel layer di input, il ché vuole anche dire che la matrice di sensing A risulta essere un *iperparametro* del modello, ovvero un parametro non allenabile.

Per quanto riguarda la funzione di Loss (o funzione di costo), essa misura quanto bene il modello sta performando calcolando la differenza tra le predizioni del modello e i valori reali del dataset. L'obiettivo dell'allenamento è minimizzare questa funzione di Loss. Alcuni esempi di funzioni di costo sono il Mean Squared Error (MSE) [21], solitamente utilizzato per problemi di regressione, e la Cross-Entropy [22], invece utilizzata per problemi di classificazione. Per questo caso specifico, la funzione implementata sarà la stessa definita dagli Autori per le solite ragioni di comparabilità dei risultati finali. In particolare, questa funzione di costo, chiamata Clipped-Cross-Entropy, risulta essere:

$$Loss = -\sum_{j|s_j=1} L_\epsilon(o_j) - \sum_{j|s_j=0} L_\epsilon(1-o_j) \tag{1.17}$$

dove $L_\epsilon(\cdot)$ è la funzione logaritmica tagliata definita come:

$$L_\epsilon(\cdot) = \min\{\log_2(1-\epsilon), \max\{\log_2(\epsilon), \log_2(\cdot)\}\} \tag{1.18}$$

e con ϵ un valore piccolo a piacere. Le ragioni dell'utilizzo di questa funzione di Loss sono legate alla composizione del vettore in uscita dal layer di output. Infatti, essendo esso costituito da numeri compresi tra 0 ed 1, estremi inclusi, esiste una possibilità di avere uno zero come argomento del logaritmo (che tenderebbe a $-\infty$). Quindi, tale funzione di costo è usata per evitare di avere instabilità numerica nel valutare la differenza tra i supporti veri ed i valori stimati.

Infine, l'algoritmo di allenamento, anche chiamato *optimizer*, è il metodo utilizzato per aggiornare i pesi del modello in modo da minimizzare la funzione di Loss. Il più comune è l'algoritmo di discesa del gradiente, anche se, in realtà, sono più utilizzate le sue varianti come il **Stochastic Gradient Descent (SGD)** [23][24], il quale aggiorna i pesi utilizzando un esempio alla volta, il **Mini-Batch Gradient Descent** [24], il quale aggiorna i pesi utilizzando piccoli gruppi di esempi (batch). Nel caso in questione, l'algoritmo di allenamento utilizzato è **Adam** [25], che sarebbe una versione più avanzata in grado di combinare le caratteristiche del SGD con quelle del *momentum*, ovvero una tecnica usata per accelerare l'algoritmo di discesa del gradiente mantenendo una velocità di avanzamento che aiuta a superare i minimi locali e a

convergere più rapidamente verso il minimo globale della funzione di Loss, adatta quando si è in presenza di rumore e non idealità.

Una volta definiti tutti questi punti, il modello è pronto per essere allenato. In particolare, l'allenamento viene svolto imponendo un set di esempi, ovvero il batch, pari a 50, un numero di *epoche* pari a 200 e un *learning rate* di 0.0001. Un'epoca è un ciclo completo attraverso l'intero dataset di addestramento. Durante un'epoca, il modello vede ogni esempio di addestramento una volta soltanto e aggiorna di conseguenza i pesi in base agli errori commessi. Conclusa un'epoca, il dataset viene rimescolato in maniera casuale (processo di *shuffling*), così da avviare un'altra epoca. Più epoche permettono al modello di convergere verso una soluzione ottimale. Il learning rate, o tasso di apprendimento, è un iperparametro che determina la dimensione dei passi che l'algoritmo compie per aggiornare i pesi del modello in risposta all'errore stimato ad ogni iterazione. Di conseguenza, un learning rate troppo alto può causare un addestramento instabile, portando il modello a non convergere o a convergere su un set di pesi subottimale. Al contrario, un learning rate troppo basso può rendere l'addestramento eccessivamente lento, con la possibilità di bloccarsi su un minimo locale.

Concluso l'addestramento della rete, è ora possibile dare in "impasto" un nuovo set di misure y, ottenendo in uscita i rispettivi supporti, che possono essere utilizzati per la ricostruzione dei segnali x.

1.5 Risultati

Per valutare l'efficacia dell'addestramento della rete occorre considerare il test set, in modo da diversificare gli input rispetto a quelli utilizzati per l'allenamento stesso, appartenenti al training set. Il test set è composto da centomila finestre. Pertanto, il modello allenato prende in ingresso i nuovi input e fornisce in uscita i supporti stimati associati a ciascuna misura. La valutazione dell'accuratezza del modello nella stima dei supporti può prendere in considerazione diversi aspetti. Come punto di partenza, si può confrontare il supporto "vero" con il rispettivo supporto stimato dalla rete. Dunque, l'accuratezza del modello può essere determinata dalla differenza nel numero totale di "1" presenti in entrambi i supporti. Tuttavia, questa differenza, di per sé, non fornisce informazioni complete, poiché un "1" in più o in meno può avere un impatto diverso sul valore di RSNR associato alla ricostruzione. Per comprendere

meglio il problema, si faccia riferimento alla figura sottostante (fig.5) che mostra un esempio di supporto "vero" e quattro possibili casi per cui è presente un "1" stimato in più o in meno.

Posizione	1	2	3	4	5	6	7	8	9	10
Vero	1	1	1	1	0	1	0	0	0	0
Caso 1	1	0	1	1	0	1	0	0	0	0
Caso 2	1	1	1	1	0	0	0	0	0	0
Caso 3	1	1	1	1	1	1	0	0	0	0
Caso 4	1	1	1	1	1	1	0	0	1	0

Figura 5: partendo dall'alto, il primo vettore rappresenta un esempio di supporto vero, mentre i vettori sottostanti rappresentano i possibili supporti stimati dalla rete. In essi è messo in evidenza come si differenziano rispetto al caso vero.

Osservazione: quando si parla di RSNR "stimato", si intende il valore di RSNR che si ottiene dalla fase di ricostruzione utilizzando il supporto stimato. Per non appesantire il discorso, si preferisce parlare direttamente di RSNR stimato e di RSNR vero, dove ancora, "vero" si riferisce al valore che si ottiene quando si ricostruisce il segnale considerando il supporto vero ottenuto con l'algoritmo di ricerca proposto.

Tornando all'esempio, il supporto vero è composto da cinque "1", i quali sono associati alle componenti della rappresentazione sparsa ordinata che permettono di raggiungere un determinato livello di energia, di conseguenza aventi un "peso" diverso. Utilizzando tale supporto nella fase di ricostruzione, si ottiene un certo valore di RSNR vero. Supponendo che l'uno in posizione 2 sia "importante", ovvero che il rispettivo elemento della rappresentazione sparsa contribuisca significativamente, nel caso 1 ci sarebbe una perdita sostanziale nella ricostruzione del segnale, ottenendo un valore di RSNR stimato più basso rispetto a quello vero. Viceversa, se si perde l'uno in posizione 6, supposto di essere di scarso valore, come nel caso 2, il valore di RSNR stimato non si discosterà significativamente da quello vero. Eppure, entrambi i casi presentano la stessa differenza in termini di "1". Le stesse considerazioni possono essere fatte anche per i casi 3 e 4, dove si ha un "1" in più in posizioni differenti e per i quali i contributi e gli effetti sul valore di RSNR possono essere diversi, ma la differenza è

sempre di un “1” rispetto al supporto vero. Questo spiega perché la differenza nel numero totale di “1” non può essere usata come unica metrica di valutazione.

L’idea è quindi quella di valutare qual è l’impatto sul valore di RSNR stimato quando:

- lo stimatore ha indovinato tutti gli “1” presenti nel supporto vero;
- lo stimatore ha indovinato più “1” di quelli presenti nel supporto vero;
- lo stimatore ha indovinato meno “1” di quelli presenti nel supporto vero;

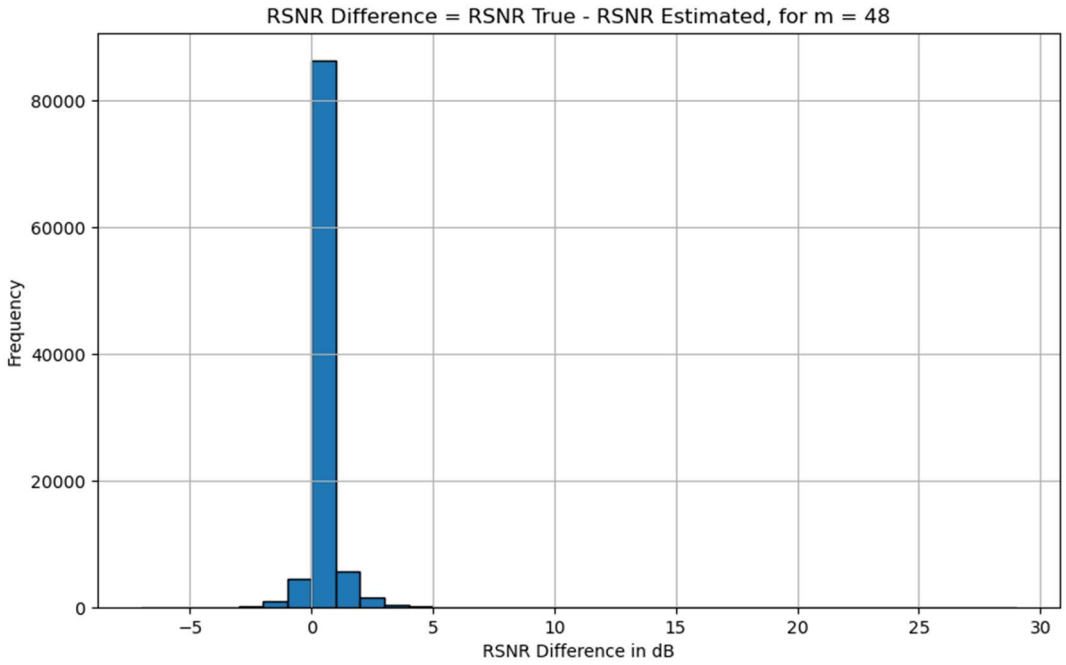


Figura 6: istogramma della distribuzione delle differenze tra i valori veri di RSNR e quelli stimati, per $m = 48$. L’istogramma sembra essere limitato tra -5 e +5 dB. In realtà, sono presenti valori anche sopra i 15 dB, ma la loro frequenza è trascurabile rispetto al bin prevalente, risultando impercettibili.

La figura 6 mostra un istogramma che provvede una prima indicazione di come i valori di RSNR stimati varino rispetto ai valori veri, considerando come dimensione $m = 48$. In particolare, per la maggior parte dei casi (più dell’80% del test set), la differenza tra il valore di RSNR vero e quello stimato non supera il decibel, sintomo di una buona accuratezza. Focalizzandosi sul semiasse positivo, però, si può osservare come tale differenza possa superare anche i $15\ dB$, seppure per pochi casi, tali da non poter essere apprezzati dall’istogramma. Dunque, per alcuni casi, gli uni non stimati causano una perdita sostanziale di informazione utile e di conseguenza la ricostruzione sarà caratterizzata prevalentemente da componenti rumorose, quindi basso valore di RSNR. Per questi limitati casi, la rete non è stata in grado di fornire una stima accurata. D’altro canto, il semiasse negativo è all’incirca popolato quanto

quello positivo, il che vuol dire che i valori di RSNR stimati sono maggiori rispetto a quelli veri. Sorgono spontanee le domande: *come è possibile che lo stimatore, pur sbagliando ad indovinare gli "1", sia in grado di fornire dei supporti con i quali si è in grado di raggiungere livelli di RSNR migliori rispetto a quelli veri? I supporti veri utilizzati per l'allenamento non dovrebbero già essere quelli ottimali e per i quali si hanno i migliori RSNR?*

Per rispondere a questi quesiti bisogna tornare indietro e considerare come vengono generati i supporti veri. Gli Autori determinano il singolo supporto come quel vettore che, tra tutte le n combinazioni, massimizza effettivamente il valore di RSNR. In questo elaborato, invece, il supporto è selezionato come quello che garantisce un certo livello di energia del segnale e quindi, una volta raggiunto tale livello di energia, l'algoritmo si ferma, producendo il supporto. Il livello di energia è però un grado di libertà che va imposto. Come mostrato nella figura 3, diversi livelli di energia sono stati considerati per scegliere quello che massimizza l'RSNR, ma le prove non sono state eseguite su tutti i possibili livelli energetici, in quanto sarebbe stato un altro problema NP-hard. Di conseguenza, l'intervallo discreto dei valori energetici considerati potrebbe aver mancato quello che veramente massimizza l'RSNR. Questo spiega perché lo stimatore, sbagliando, produce un migliore RSNR stimato rispetto a quello vero: quel set di "1" in più o in meno considera o esclude quegli elementi della rappresentazione sparsa che avrebbero contribuito o meno a raggiungere il livello energetico ideale. Volendo chiarificare

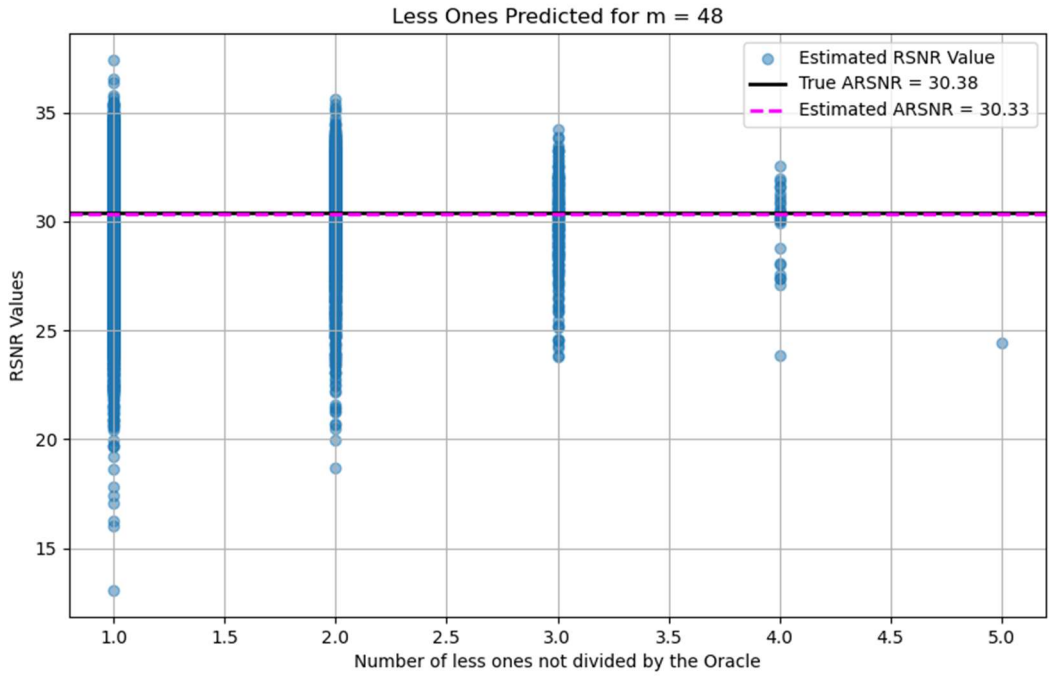


Figura 7.a: visualizzazione della distribuzione dei valori di RSNR stimati quando il TSOC indovina meno uni rispetto a quelli realmente presenti nel supporto vero. La distribuzione è poi suddivisa in base a quanti uni mancano.

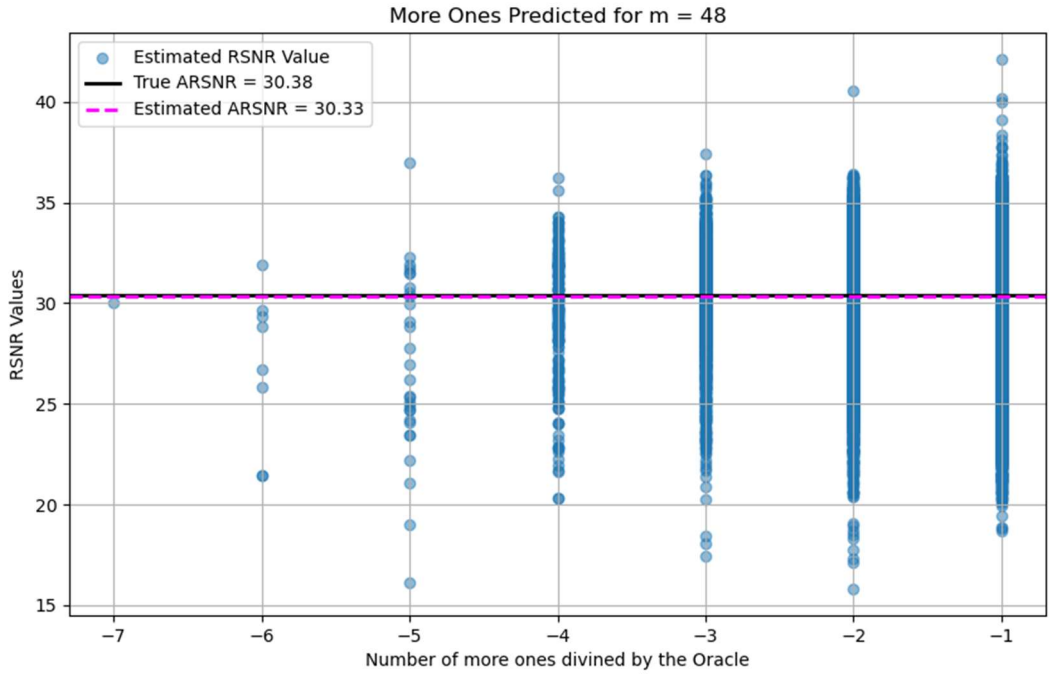


Figura 7.b: visualizzazione della distribuzione dei valori di RSNR stimati quando il TSOC indovina più uni rispetto a quelli realmente presenti nel supporto vero. La distribuzione è poi suddivisa in base a quanti uni in più ci sono.

ulteriormente il concetto, se quando ci sono più "1" e l'RSNR migliora, significa che il livello energetico scelto è basso; viceversa, se ci sono meno "1" e l'RSNR migliora, allora il livello energetico considerato è alto. Segue che nell'implementare l'algoritmo proposto per migliorare l'efficienza nella ricerca dei supporti, è importante considerare anche l'aspetto negativo introdotto dal grado di libertà nella scelta del livello energetico. Quanto appena spiegato non può essere interpretato dall'istogramma, poiché esso mostra solo una distribuzione della differenza tra il valore di RSNR vero e quello stimato per tutti i supporti indovinati, senza distinguere se sono presenti più o meno uni oppure se di pari numero, come accennato precedentemente. Pertanto, la figura 7 permette di visualizzare tale distribuzione per categoria. In particolare, la fig.7a e la fig.7b mostrano tutti gli RSNR stimati, paragonandoli rispetto il valore medio. Esse mostrano come perdere o guadagnare un uno, rispetto al supporto vero, possa incrementare o diminuire il valore di RSNR stimato, a seconda dell'importanza di quell'uno. Tale aspetto è riportato anche per i casi in cui ci siano più o meno uni stimati. Ad ogni modo, la figura 8, mostra le stesse informazioni dell'istogramma, ma in modo più specifico e dettagliato. Infatti, essa permette di visualizzare chiaramente la distribuzione delle differenze di RSNR per ogni casistica, evidenziando come la perdita o il guadagno di un uno possa

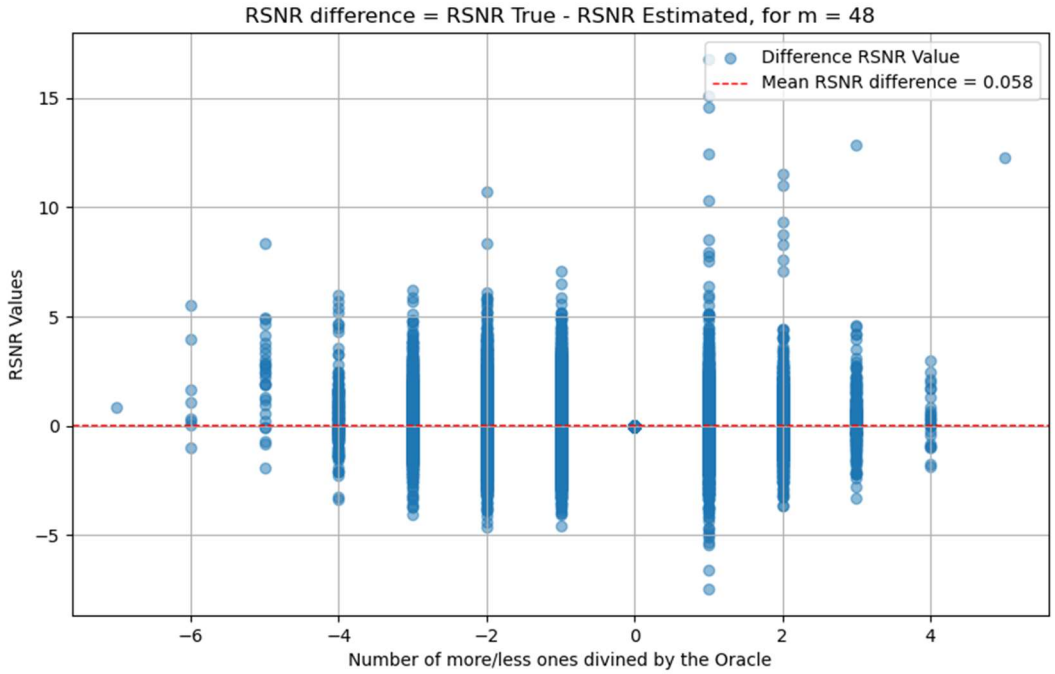


Figura 8: visualizzazione dell'istogramma sottoforma di un "esploso". Infatti, la distribuzione dei valori delle differenze di RSNR rimane prevalentemente concentrata tra -5 e +5 dB, ma in questo modo è possibile anche vedere come i dati si distribuiscono tra le varie casistiche in cui il TSOC ha indovinato più o meno uni. Inoltre, con questo grafico, si possono visualizzare anche i valori "nascosti" dall'istogramma.

influenzare il valore stimato di RSNR, positivamente o negativamente. Sebbene possa risultare ovvio, vale la pena osservare come, per supporti perfettamente stimati, la differenza è nulla. Un'altra considerazione importante è la differenza media. Essa è molto prossima allo zero ed infatti i valori di ARSNR vero e stimato sono pressoché uguali, indice di un allenamento ben svolto. Le figure ed i risultati sono stati commentati per un solo livello di compressione m. Chiaramente, le stesse considerazioni possono essere fatte anche per gli altri livelli di compressione, ma siccome i risultati sono molto simili tra loro, si preferisce riportare solo una tabella con i valori di ARSNR veri e stimati, con rispettivo errore medio, per ciascuna dimensione m.

m	28	32	36	40	44	48	64
ARSNR vero (dB)	23.52	25.91	27.50	29.15	30.17	30.38	32.28
ARSNR stimato (dB)	23.45	25.85	27.44	29.00	30.11	30.33	32.24
Errore medio (dB)	0.071	0.062	0.056	0.147	0.061	0.058	0.038

Questi valori, se comparati con quelli riportati in fig.2, evidenziano come i risultati ottenuti con un approccio differente siano pressoché simili, raggiungendo l'obiettivo prefissato.

Capitolo 2

ANOMALY DETECTION E SELF-ASSESSMENT

2.1 Obiettivo

Nel primo capitolo si è discusso come migliorare l'algoritmo di ricerca dei supporti delle rappresentazioni sparse dei segnali, con l'obiettivo di allenare il modello TSOC in modo da poter direttamente stimare tali supporti e di conseguenza ottenere una fase di decoding del CS più efficace e funzionale. In questo secondo capitolo, invece, si vuole coinvolgere il modello allenato per definire un processo di Anomaly Detection e valutare le sue capacità di distinzione tra segnali normali e segnali anomali. Poiché le anomalie sono comunque eventi rari, occorre definire un test set di misure anomale y_a sfruttando il test set di misure non anomale y_{na}, che rappresenta il comportamento normale. Ricordando che il supporto di un segnale rappresenta l'insieme delle posizioni dei coefficienti non nulli nella sua rappresentazione sparsa, quando una misura anomala viene data in ingresso al TSOC, il relativo supporto stimato dovrebbe differenziarsi chiaramente dal supporto associato alla corrispondente misura non anomala. Questo significa che le posizioni dei coefficienti non nulli nella rappresentazione sparsa del segnale saranno notevolmente diverse rispetto a quelle del segnale normale ricostruito $\hat{x}_{na}$, ottenendo così, una versione ricostruita anomala $\hat{x}_a$. Ciò non è vero in generale, siccome durante la fase di encoding/decoding, l'eventuale anomalia presente può essere "involontariamente" cancellata e quindi non essere rilevata. Ciò nonostante, il processo di compressione e ricostruzione che fa uso del modello TSOC, non dovrebbe soffrire di questa problematica. D'altronde, la non cancellazione dell'anomalia dovrebbe essere garantita dal fatto che il TSOC è stato allenato con un dataset di misure e supporti normali, dunque, nel caso il TSOC dovesse indovinare il supporto di una misura anomala, l'accuratezza di questa stima dovrebbe essere altamente compromessa dalla presenza dell'anomalia, garantendo una ricostruzione del segnale anomalo. In sintesi, l'obiettivo finale è valutare queste considerazioni e verificare se il TSOC, una volta allenato, possa essere effettivamente di aiuto nel processo di Anomaly Detection.

2.2 Introduzione alle anomalie

Le anomalie rappresentano eventi, osservazioni o comportamenti che si discostano significativamente dal pattern atteso o normale. Nel contesto dei segnali biomedici, queste anomalie possono indicare problemi di salute, errori di misurazione o interferenze esterne. In generale, le anomalie, in base al loro effetto sulla potenza del segnale, possono essere classificate in tre categorie principali: ***aumento***, ***invariabilità*** e ***diminuzione*** [26]. Le anomalie che causano un **aumento** della potenza del segnale includono *disturbi sovrapposti*, generati da interferenze esterne o da eventi che aggiungono energia al segnale originale. Per esempio, nel segnale ECG, un aumento improvviso della potenza potrebbe essere dovuto a un'interferenza elettromagnetica o a un movimento brusco del paziente che introduce rumore nel segnale. Invece, le anomalie a potenza **invariata** sono quelle che non alterano significativamente la potenza complessiva del segnale, ma la ridistribuiscono, causando *deviazioni* che modificano la sua struttura o il suo contenuto informativo, rendendole più difficili da rilevare. Ad esempio, un'aritmia non cambia la potenza complessiva del segnale, ma altera il pattern normale dell'ECG. Infine, le anomalie che causano una **diminuzione** della potenza del segnale includono *distorsioni non lineari* che riducono l'energia complessiva del segnale, le quali possono essere causate da guasti nei sensori, perdite di segnale o attenuazioni. D'altro canto, esiste un'ulteriore categorizzazione delle anomalie basata sul loro verificarsi nelle diverse fasi del monitoraggio dei segnali: ***prima***, ***durante*** e ***dopo*** l'acquisizione. **Prima** dell'acquisizione, le anomalie possono derivare da problemi con i sensori o i trasduttori utilizzati per rilevare i segnali. Infatti, un sensore mal calibrato può generare dati inaccurati, così come condizioni ambientali avverse possono influenzare negativamente la qualità del segnale. **Durante** l'acquisizione, le anomalie possono manifestarsi come rumore o disturbi nel segnale, dovuto a movimenti del paziente, ma anche problemi hardware, come connessioni difettose che possono introdurre artefatti nel segnale. **Dopo** l'acquisizione, le anomalie possono emergere durante la fase di elaborazione e analisi dei dati. Errori di quantizzazione, perdita di dati durante la trasmissione o problemi di sincronizzazione possono compromettere l'integrità del segnale. Tuttavia, la ridondanza del segnale, ovvero la presenza di informazioni duplicate o sovrabbondanti, può essere utilizzata per migliorare la robustezza del sistema di monitoraggio. Infatti, l'uso di sensori multipli per monitorare lo stesso parametro può aiutare a identificare e correggere le anomalie, garantendo una maggiore affidabilità dei dati raccolti, ma questo rimane un aspetto secondario per questa trattazione.

2.2.1 Tipologie di anomalie per segnali ECG

La figura 9 illustra vari esempi di anomalie che possono verificarsi in un segnale elettrocardiogramma, tutte con lo stesso livello di deviazione, cioè di quanto le curve anomale si discostano da quelle normali. Gli esempi sono suddivisi in categorie, in particolare, la prima

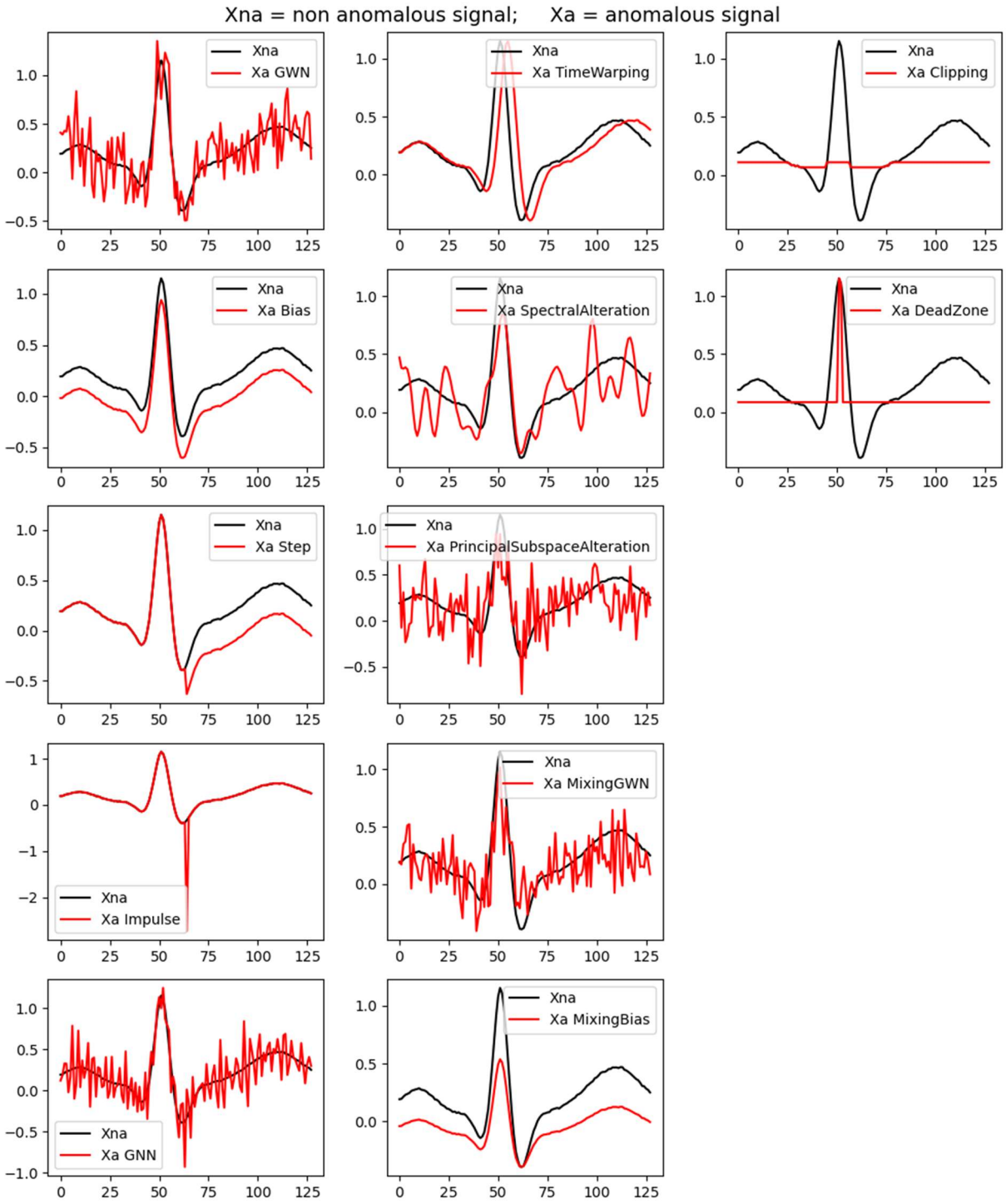


Figura 9: tipologie di anomalie che si possono verificare su un tracciato ECG.

colonna è composta da cinque anomalie appartenenti alla famiglia che causa un aumento della potenza del segnale; la colonna centrale è anch'essa composta da cinque anomalie, ma appartenenti alla famiglia che non varia la potenza del segnale; infine, l'ultima colonna è composta da due anomalie soltanto, appartenenti alla famiglia che riduce la potenza del segnale. Per ciascuna anomalia riportata in figura, se ne descrivono le caratteristiche principali:

1. ***Anomalie che aumentano la potenza del segnale:***
   a. **GWN**: il rumore bianco gaussiano (Gaussian White Noise) è un tipo di disturbo caratterizzato da una distribuzione gaussiana dei valori di ampiezza e da una densità spettrale di potenza costante su tutto lo spettro di frequenza del segnale (definizione di rumore bianco). Esso si sovrappone al segnale ECG rendendo difficile la distinzione tra i picchi del complesso PQRST.
   b. **Bias**: questo tipo di anomalia si riferisce ad una deviazione sistematica che introduce un errore costante nel segnale, spostando l'intero tracciato ECG verso l'alto o verso il basso. Esso può rendere difficile l'interpretazione accurata delle onde P, QRS e T, poiché i valori di base del segnale sono alterati.
   c. **Step**: l'anomalia step si riferisce ad un cambiamento improvviso e sostenuto nel livello del segnale verso l'alto o verso il basso. Le conseguenze sono simili a quelle causate dall'anomalia bias.
   d. **Impulse**: l'anomalia si riferisce ad un impulso di breve durata e con ampiezza non trascurabile che si sovrappone al segnale, tale da poter distorcere significativamente il tracciato ECG.
   e. **GNN**: il rumore a banda stretta gaussiano (Gaussian Narrowband Noise) è un disturbo caratterizzato da una distribuzione gaussiana dei valori di ampiezza, come il GWN, ma con una densità spettrale di potenza concentrata in una banda di frequenza specifica, piuttosto che distribuita uniformemente su tutto lo spettro delle frequenze. Se tale banda coincide con quella tipica del segnale ECG, gli effetti sono analoghi a quelli generati dal GWN.

2. ***Anomalie che non variano la potenza del segnale:***
   a. **Time Warping**: questa anomalia si riferisce ad una distorsione temporale del segnale, introducendo variazioni non lineari nella velocità del segnale, che possono allungare o comprimere segmenti del tracciato ECG. Essa rende difficile l'identificazione accurata

delle onde P, QRS e T, siccome la loro posizione temporale può essere significativamente alterata.

b. **Spectral Alteration**: essa si riferisce a cambiamenti nelle caratteristiche spettrali del segnale, alterando le componenti in frequenza del segnale ECG e dunque ridistribuendo l'energia spettrale. Gli effetti sono simili a quelli causati dal GWN.
c. **Principal Subspace Alteration**: l'alterazione del sottospazio principale è un'anomalia che si verifica quando le componenti principali del segnale ECG vengono modificate in modo significativo. Le componenti principali rappresentano le direzioni di massima varianza, dunque, una rotazione di tali direzioni implica un'alterazione delle componenti che modificano la struttura morfologica del complesso PQRST, rendendola di difficile identificazione.
d. **Mixing GWN**: tale anomalia introduce un rumore bianco gaussiano in modo da mescolare o alterare le caratteristiche originali del segnale, ma senza variarne la potenza complessiva. Tuttavia, gli effetti sono del tutto analoghi a quelli causati dal GWN.
e. **Mixing Bias**: vale lo stesso discorso del punto precedente, ma in riferimento al bias.

3. Anomalie che riducono la potenza del segnale:

a. **Clipping**: tale anomalia si verifica quando l'ampiezza del segnale ECG supera la capacità di registrazione del sistema di monitoraggio, causando un "taglio" del segnale che compromette l'interpretazione del tracciato ECG.
b. **Dead-Zone**: questa anomalia si verifica quando una parte del segnale ECG non viene registrata, risultando in una perdita di informazioni cruciali, come per il Clipping.

2.2.2 Riproduzione ed iniezione nel test set di anomalie a diverse intensità

Gli esempi precedentemente illustrati sono stati riprodotti implementando i modelli delle anomalie proposti nel paper [26]. Come già specificato, si è mantenuto lo stesso livello di deviazione per tutte le anomalie. L'intensità della anomalia, infatti, può essere governata dal parametro δ che può variare all'interno dell'intervallo $[0,1]$. Chiaramente, $\delta = 0$ implica assenza di anomalia applicata al segnale, mentre $\delta = 1$ significa massima predominanza dell'anomalia, rendendo il segnale originale difficilmente distinguibile.

Il primo passo verso l'obiettivo consiste nel comporre dei test set anomali. Dunque, preso il test set composto da misure normali, si sceglie una anomalia e la si applica a tutte le misure con diversi livelli di intensità dell'anomalia. Senza perdita di generalizzazione, la procedura viene eseguita solo per tre anomalie, in particolare: Impulse, Spectral Alteration (SA) e Principal Subspace Alteration (PSA).

La figura 10.a illustra l'applicazione dell'anomalia Impulse, con il parametro δ variabile tra 0 e 0,05. Si è scelto di non superare questo intervallo per evitare che il segnale diventi eccessivamente anomalo. Infatti, già con $\delta = 0{,}05$, l'anomalia inizia a predominare, riducendo il valore di SNR di circa $9\,dB$ rispetto al caso con $\delta = 0{,}01$, come evidenziato dalle illustrazioni. Inoltre, mantenere un intervallo limitato per il parametro δ consente di valutare meglio la sensibilità nella rilevazione delle anomalie, poiché, ovviamente, maggiore è l'intensità dell'anomalia, maggiore è la probabilità che essa venga rilevata, o almeno idealmente. Considerazioni analoghe possono essere ripetute anche per le altre due casistiche riportate nelle figure 10.b e 10.c, dove l'incremento dell'intensità della anomalia al variare di δ si evince maggiormente.

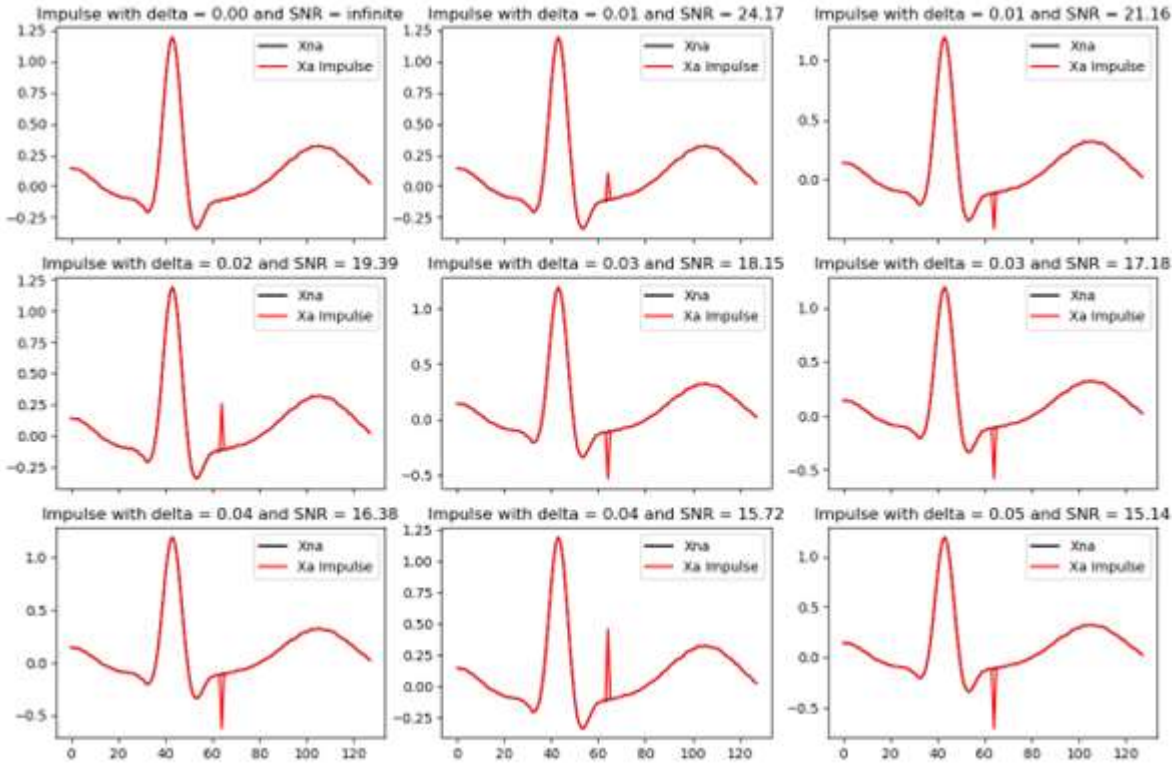


Figura 10.a: applicazione dell'anomalia Impulse al tracciato ECG per diverse intensità.

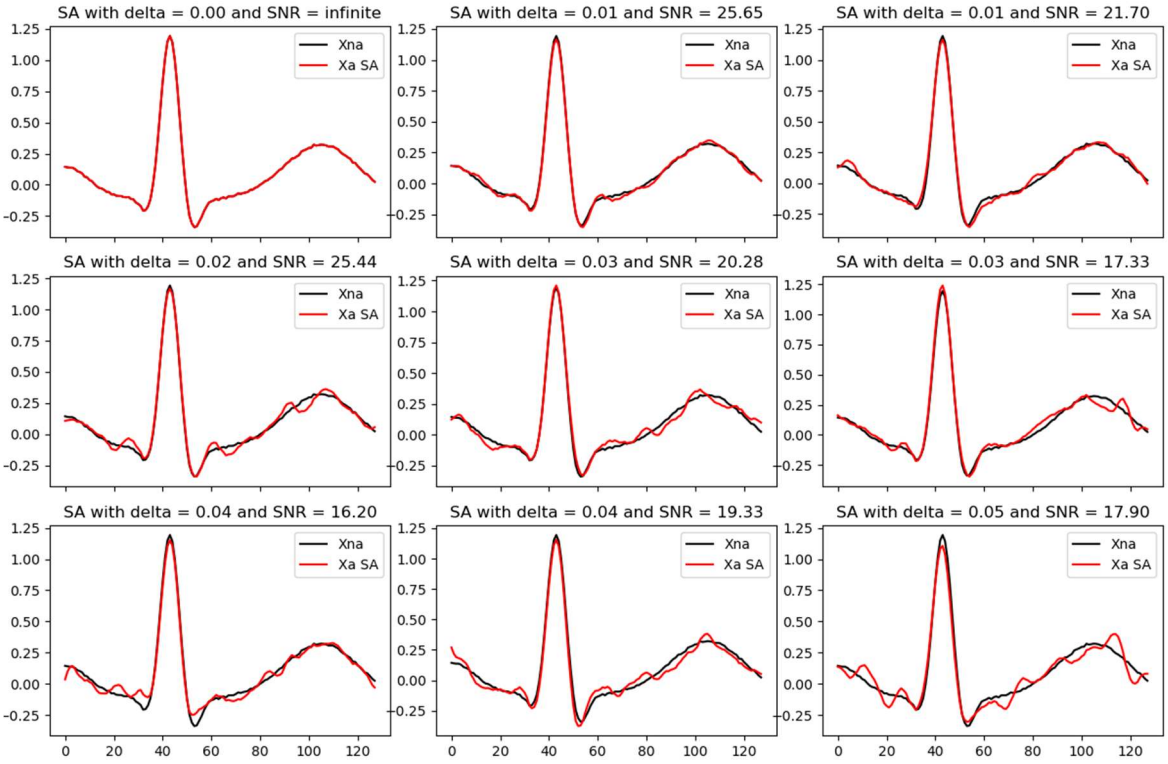


Figura 10.b: applicazione dell'anomalia Spectral Alteration al tracciato ECG per diverse intensità.

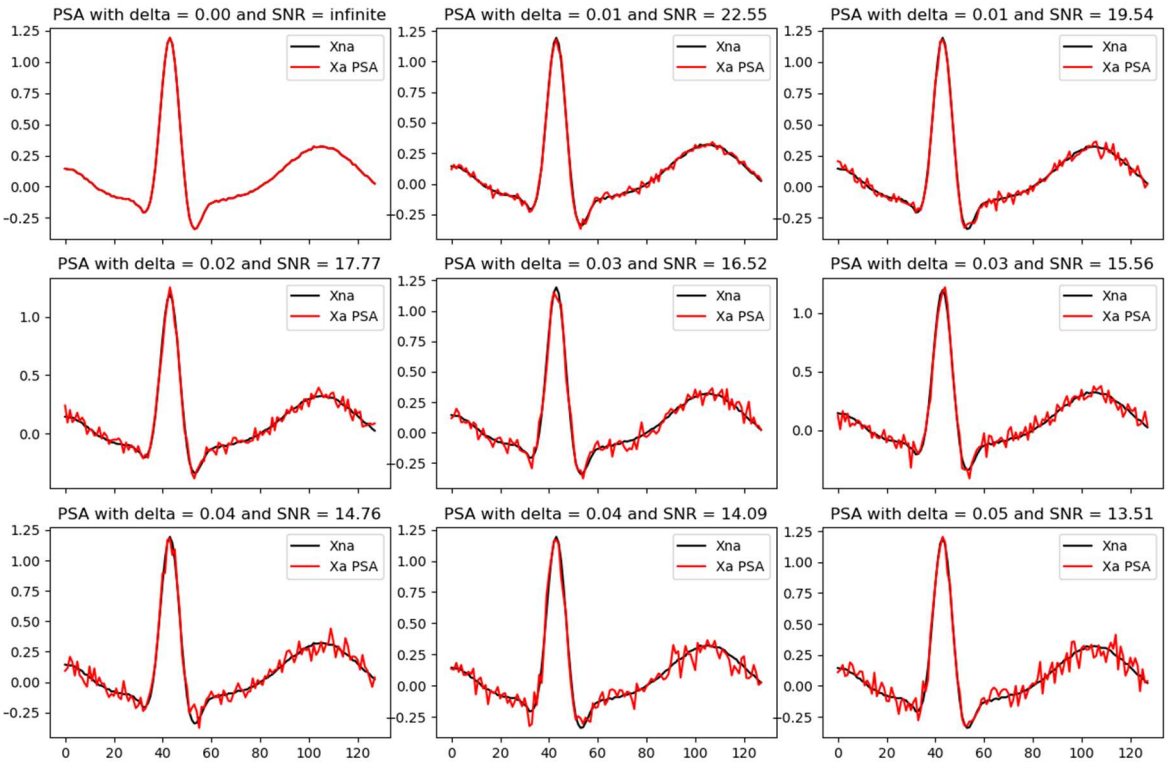


Figura 10.c: applicazione dell'anomalia Principal Subspace Alteration al tracciato ECG per diverse intensità.

A questo punto, dopo aver ottenuto i test set anomali per diverse tipologie di anomalia e per diverse intensità a partire dal test set normale, si può procedere con la fase successiva: l'identificazione di una metrica di valutazione per il processo di Anomaly Detection.

2.3 Introduzione alla Anomaly Detection

Si è detto che una anomalia è un evento, un'osservazione o un comportamento che si discosta significativamente dal pattern atteso o normale. Siccome le anomalie possono indicare errori, guasti o altre situazioni critiche, soprattutto in ambito medico, la loro rilevazione tempestiva è di fondamentale importanza. L'Anomaly Detection è dunque il processo che si occupa di identificare tali anomalie presenti all'interno di un dataset.
Esistono diverse tecniche utilizzate per l'Anomaly Detection [27], di cui le più conosciute sono:

- **DBSCAN**: il Density-Based Spatial Clustering of Applications with Noise è un algoritmo di clustering che identifica regioni dense di punti e considera i campioni isolati come anomalie;
- **LOF**: il Local Outlier Factor misura la densità locale di un punto rispetto ai suoi vicini per identificare outliers;
- **PCA**: il Principal Component Analysis è una tecnica lineare e statistica che riduce la dimensionalità dei dati identificando le direzioni di massima varianza. Le anomalie possono essere rilevate analizzando i residui di ricostruzione, ovvero la differenza tra i dati originali e quelli proiettati sulle componenti principali;
- **Autoencoder**: è una rete neurale che può apprendere una rappresentazione compressa non lineare dei dati e rileva anomalie basandosi sulla ricostruzione degli errori;
- **SVM**: il Support Vector Machine è un algoritmo di classificazione che può essere adattato per rilevare anomalie separando i dati normali da quelli anomali.

Queste tecniche sono state tra le prime ad essere introdotte e sono tuttora utilizzate. Tuttavia, data l'importanza dell'Anomaly Detection nell'ambito della prevenzione, nuove tecniche sempre più complesse e sofisticate sono state proposte in letteratura, differenziandosi in base alle esigenze specifiche dell'applicazione di interesse. Ad esempio, il Compressed Sensing può essere una valida alternativa all'Autoencoder, essendo simili ma con prestazioni e caratteristiche intrinseche differenti. Infatti, l'Autoencoder [28] è una rete neurale che apprende i patterns dei dati, mentre il CS è una tecnica di elaborazione che sfrutta i principi di sparsità e

di incoerenza del segnale, ma entrambi sono caratterizzati da una fase di compressione (encoding) e da una di ricostruzione (decoding). Pertanto, dopo aver discusso il primo capitolo incentrato sul Compressed Sensing, la tecnica di rilevamento delle anomalie che si intende implementare sarà basata su CS, richiamando anche il modello TSOC addestrato.

2.3.1 Compressed Sensing e Anomaly Detection

Nel contesto dell'Anomaly Detection, il CS è particolarmente vantaggioso perché le anomalie, essendo spesso sparse, possono essere rilevate con maggiore precisione ed efficienza. Inoltre, come discusso nel capitolo precedente, il CS permette di ricostruire segnali sparsi da un numero ridotto di campioni, sfruttando, appunto, la loro rappresentazione sparsa in un dominio appropriato. Pertanto, il volume di dati da elaborare viene ridotto, diminuendo così i consumi energetici e migliorando la velocità di elaborazione. Questo aspetto è di cruciale importanza in applicazioni come il monitoraggio biomedico, dove è essenziale rilevare rapidamente le anomalie per intervenire tempestivamente. Come accennato, il prossimo passo consiste nel determinare una metrica di valutazione che permetta di distinguere cosa è anomalo da cosa non lo è. Per fare ciò, rifacendosi al lavoro svolto precedentemente, si procede come segue: il test set di misure normali y_{na}, ottenuto dalla prima fase di encoding, viene dato come input al TSOC, il quale produce i relativi supporti (normali) s_{na} delle rappresentazioni sparse, utilizzate per la fase di ricostruzione dei segnali $\hat{x}_{na}$. Conclusa la procedura di decoding, si ripete lo step di encoding, cioè, si ricomprimono i segnali ricostruiti $\hat{x}_{na}$ mediante la matrice di sensing A, ottenendo le misure "ricostruite" $\hat{y}_{na}$. Lo schema illustrato in figura 11 mette in evidenza i diversi passaggi per il caso normale, ma nulla cambia se si considerano i test set anomali definiti precedentemente. Così facendo, si hanno a disposizione le misure y_{na} e $\hat{y}_{na}$ che possono essere utilizzate per valutare l'errore di ricostruzione. D'altronde, se y_{na} è una misura priva di anomalie e se il TSOC è stato allenato bene, quindi è in grado di fornire una

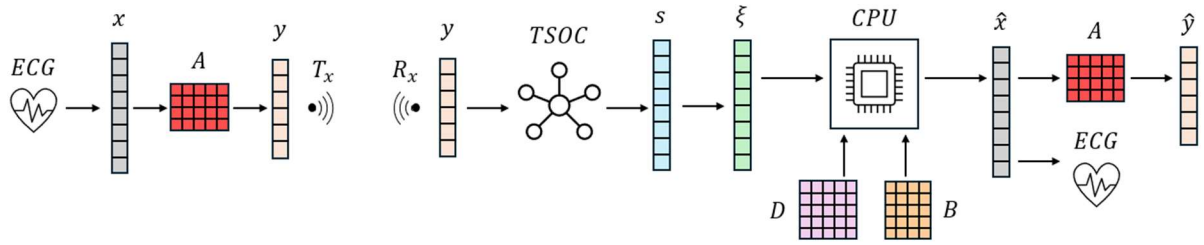


Figura 11: schema di encoding-decoding di riferimento al quale viene aggiunto un ulteriore step di encoding per determinare la misura ricostruita.

stima accurata del supporto s_{na}, allora il segnale ricostruito $\hat{x}_{na}$ è anch'esso privo di anomalie e ci si aspetterebbe che ricomprimendolo si ottenga una misura $\hat{y}_{na}$ che sia pressoché identica a y_{na}. Viceversa, qualora il TSOC dovesse ricevere in ingresso una misura anomala y_a, derivante dalla fase di encoding del segnale x_a che si suppone essere già affetto da una anomalia, allora il passaggio di decoding-encoding dovrebbe mettere in evidenza l'effetto dell'anomalia stessa, siccome il TSOC, non essendo stato allenato per questi tipi di dato, fornisce un supporto s_a che si differenzia da quello normale. Dunque, la ricostruzione del segnale $\hat{x}_a$ con s_a risente di questo difetto, risaltando l'anomalia. In altre parole, per i motivi elencati, ci si aspetterebbe che $\hat{x}_a$ sia diverso da x_a e di conseguenza che lo sia anche la misura ricostruita $\hat{y}_a$ da y_a. Pertanto, si può sfruttare questa differenza tra la misura di partenza e la misura ricostruita per definire una metrica di valutazione per l'identificazione delle anomalie.

2.3.2 Metriche e strumenti di Self-Assessment

In letteratura si possono trovare diverse metriche per l'Anomaly Detection nei problemi regressivi come nel caso in questione, ma alcune delle più utilizzate sono il **Mean Squared Error (MSE)**, il **Mean Absolute Error (MAE)**, il **Root Mean Squared Error (RMSE)** ed infine il **Mean Squared Logarithmic Error (MSLE)**. Ognuna di queste metriche presenta caratteristiche uniche che le rendono adatte a diversi tipi di analisi, ma comunque tutte applicabili per valutare, sotto diversi aspetti, le differenze tra y_{na} e $\hat{y}_{na}$ e tra y_a e $\hat{y}_a$.
Il MSE è, tra tutte, la metrica più comune utilizzata nei modelli di regressione. Essa è espressa dalla seguente formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.1}$$

dove N rappresenta il numero totale di misure presenti nel test set, y_i è la misura ottenuta dalla prima fase di encoding, mentre $\hat{y}_i$ è la misura ricostruita, ottenuta dalla seconda fase di encoding. Dall'espressione è facile intuire come il MSE misura la media dei quadrati degli errori, ovvero la differenza tra la misura e la misura ricostruita. Questa metrica è particolarmente utile perché penalizza maggiormente gli errori più grandi a causa della quadratura, rendendola sensibile agli outlier. Tuttavia, proprio questa sensibilità, che evidenzia l'eventuale presenza di una anomalia, può essere un limite. Infatti, il TSOC, seppur sta

lavorando in assenza di anomalie, può produrre una stima che devia leggermente dal supporto vero, causando una lieve differenza che può essere penalizzata dal MSE, generando un falso positivo. Dunque, per non affidarsi esclusivamente al MSE, vengono implementate anche le altre metriche sopra citate. Procedendo con ordine, l'espressione del MAE è:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.2}$$

Il MAE, a differenze del MSE, misura la media degli errori assoluti tra le due misure. Questa metrica è più robusta rispetto alla precedente in presenza di outlier, poiché tratta tutti gli errori allo stesso modo, senza penalizzare eccessivamente gli errori grandi. Dunque, con questa metrica si ottiene una valutazione più uniforme degli errori.
Il RMSE è semplicemente la radice quadrata del MSE, per cui la formula è:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.3}$$

La radice quadrata riporta l'errore alla stessa unità di misura dei dati originali, rendendo più facile l'interpretazione dell'accuratezza del modello. Tuttavia, derivando dal MSE, anche il RMSE penalizza maggiormente gli errori più grandi.
Infine, la metrica MSLE misura la media dei quadrati delle differenze tra i logaritmi delle misure, come espresso dalla seguente formula:

$$MSLE = \frac{1}{N}\sum_{i=1}^{N}(\log(1 + y_i) - \log(1 + \hat{y}_i))^2 \tag{2.4}$$

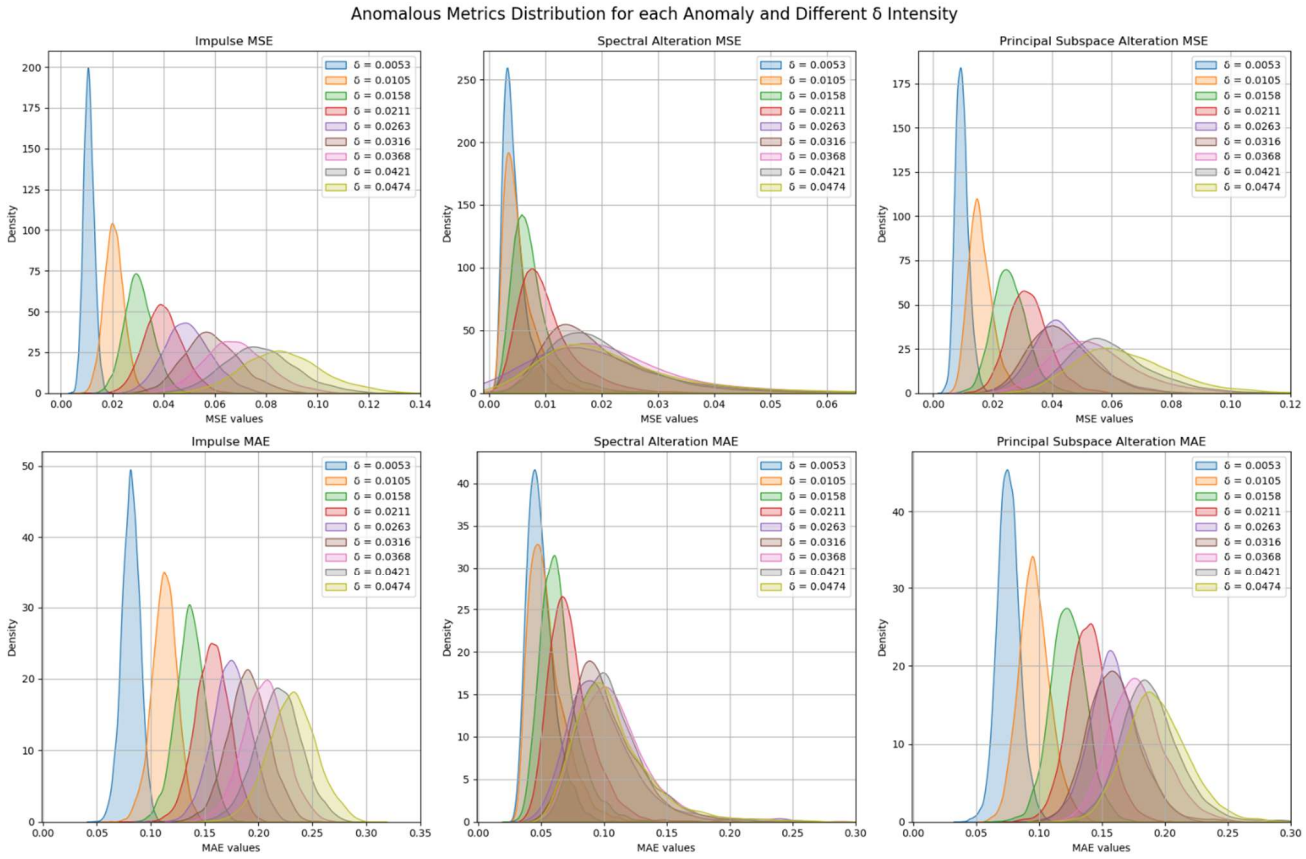
Il MSLE è utile quando si vuole penalizzare maggiormente le sottostime rispetto alle sovrastime. Inoltre, essa è adatta quando si vuole ridurre l'impatto degli outlier, grazie all'azione dei logaritmi che hanno un effetto di compressione sui valori estremi.

Dunque, tutte queste metriche sono utilizzate per valutare la differenza tra la misura iniziale e quella ricostruita, sia per il test set normale, sia per quello a cui sono state iniettate le anomalie.

In questo modo, è possibile ottenere una distribuzione per i valori di MSE_{na} per le misure non anomale e una distribuzione di MSE_a per le misure anomale. Idem per le altre metriche. A questo punto, avendo definito una classe di valori normali e una classe di valori anomali, non rimane che applicare degli strumenti di Self-Assessment. La caratteristica dei dati che si hanno a disposizione predilige degli strumenti adatti a problemi di classificazione, come l'Area Under the Curve (AUC) [29]. Infatti, essa è una metrica utilizzata prevalentemente per valutare le performance dei modelli di classificazione binaria. L'AUC appartiene alla categoria delle metriche di valutazione basate sulle curve ROC (Receiver Operating Characteristic) e rappresenta la probabilità che il modello classifichi correttamente tra le due classi in questione, che in questo caso sono MSE_{na} e MSE_a per l'errore quadratico medio.

2.4 Risultati

La figura 12 mostra le distribuzioni delle metriche anomale per diverse intensità di anomalie. La prima riga presenta tre grafici distinti che rappresentano le distribuzioni di MSE_a per ciascuna anomalia: Impulse, Spectral Alteration e Principal Subspace Alteration. Questi grafici offrono una chiara visualizzazione dell'impatto delle anomalie sui valori di MSE_a, evidenziando come la distribuzione degli errori quadratici medi varia con l'aumento



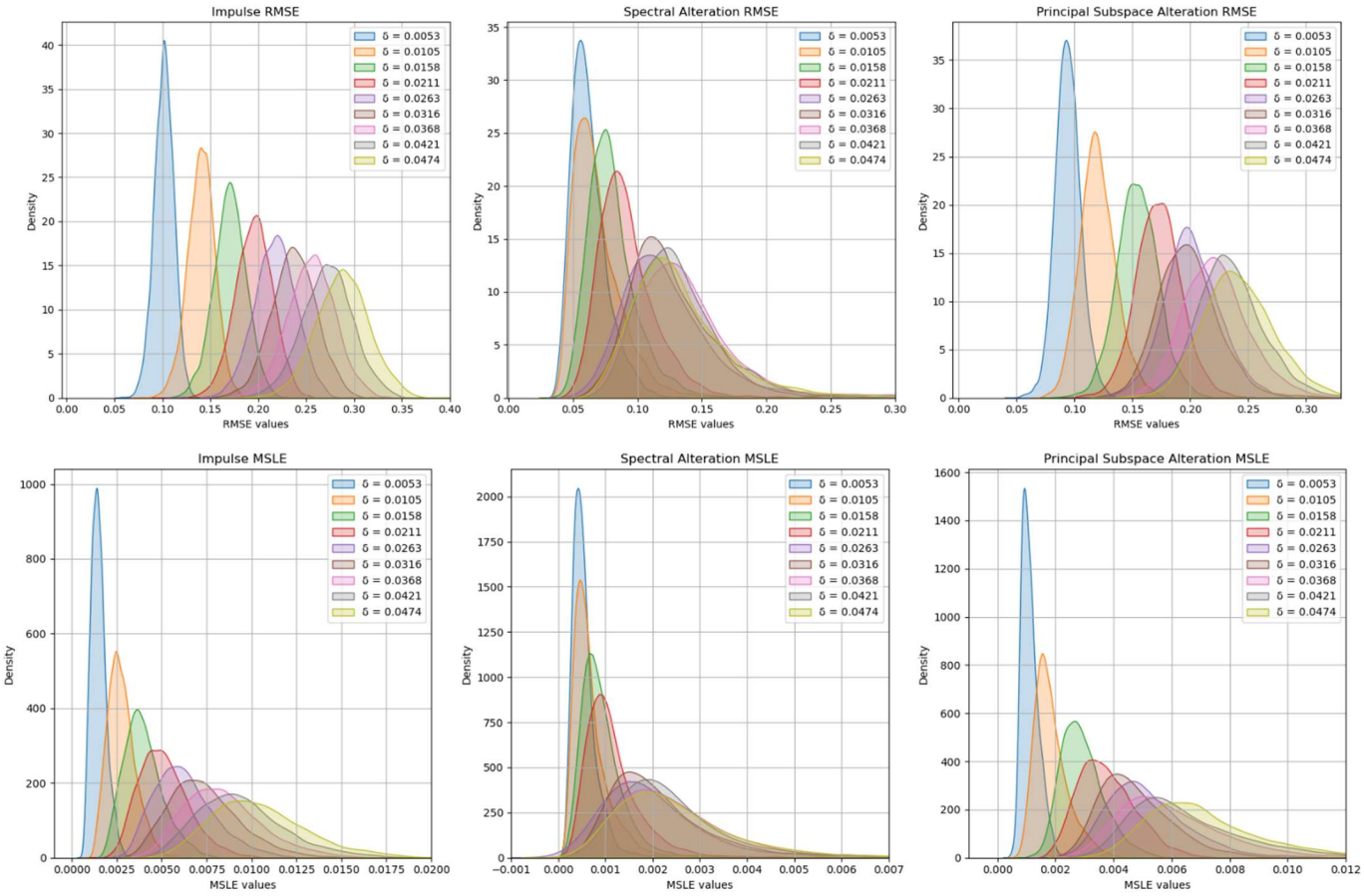


Figura 12: confronto tra le distribuzioni delle varie metriche all'aumentare del parametro δ. Partendo dall'alto, la prima riga evidenzia le distribuzioni dei valori di MSE anomali per le tre anomalie studiate. Susseguono MAE, RMSE e MSLE.

dell'intensità delle anomalie. Infatti, soprattutto per Impulse e PSA, si vede bene come all'aumentare di δ si verifica una traslazione del picco della curva verso destra. Questo indica che i valori di MSE_a tendono ad aumentare con l'intensificarsi delle anomalie. Inoltre, si osserva una diminuzione dell'ampiezza della densità, il che indica che i valori di MSE_a sono meno concentrati attorno al picco centrale. Infine, c'è un aumento della deviazione standard, indicando una maggiore dispersione dei valori di MSE_a. Questo comportamento presenta sia vantaggi che svantaggi. Da un lato, l'aumento dei valori di MSE_a con l'incremento di δ facilita il compito del detector nel distinguere questi valori da quelli di MSE_{na}. Dall'altro lato, una maggiore dispersione di tali valori comporta una sovrapposizione, seppur limitata, con i valori di MSE_{na}. Questo rende più difficile per il detector distinguere correttamente tra le due classi, aumentando il rischio di falsi positivi e/o veri negativi, ovvero confondendo i valori di una classe con quelli dell'altra. Considerazioni del tutto analoghe possono essere fatte anche per le altre metriche. Infatti, le distribuzioni di MAE_a sono paragonabili a quelle di MSE_a per tutte e

tre le anomalie, così come quelle di $RMSE_a$. D'altronde il RMSE riporta i dati di MSE alla stessa unità di misura dei segnali compressi, come già spiegato. Le uniche distribuzioni che si distinguono sono quelle di $MSLE_a$. Infatti, l'effetto di compressione applicato dalla funzione logaritmica la si può osservare sull'asse delle ascisse, che mostra un intervallo più limitato rispetto alle altre distribuzioni. Segue che le distribuzioni sono molto "vicine" tra loro. Pertanto, ci si aspetta che con questa metrica l'AUC restituisca dei valori più bassi rispetto a quelli che può fornire per le altre metriche. Ad ogni modo, le illustrazioni riportate mettono in evidenza come le distribuzioni anomale varino tra di loro all'aumentare di δ. Invece, la figura 13 mette

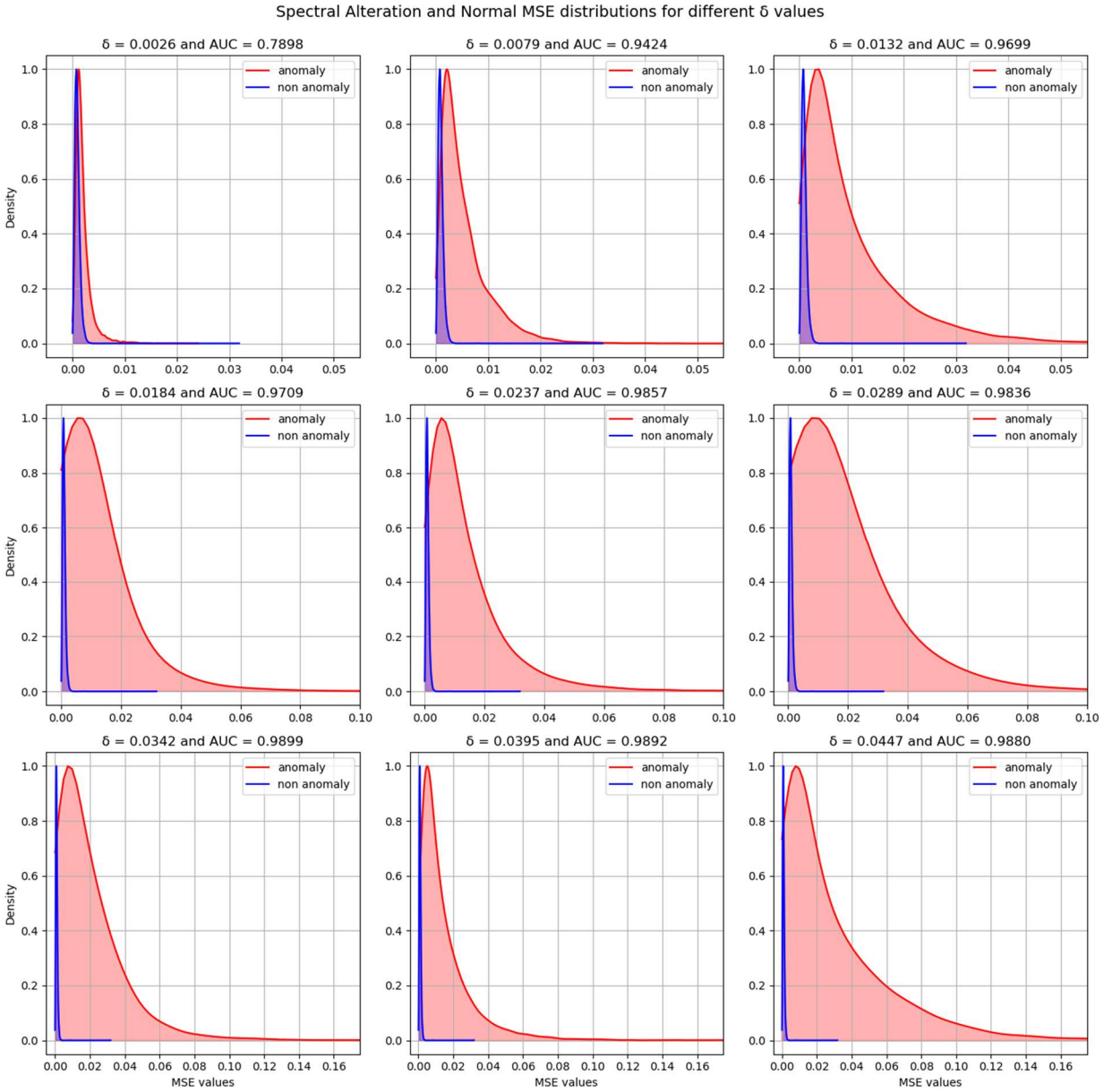


Figura 13: comparazione tra la distribuzione dei valori di MSE normali (curva blu) con la distribuzione dei valori di MSE anomali (curva rossa), per la sola anomalia SA e per diversi valori di δ.

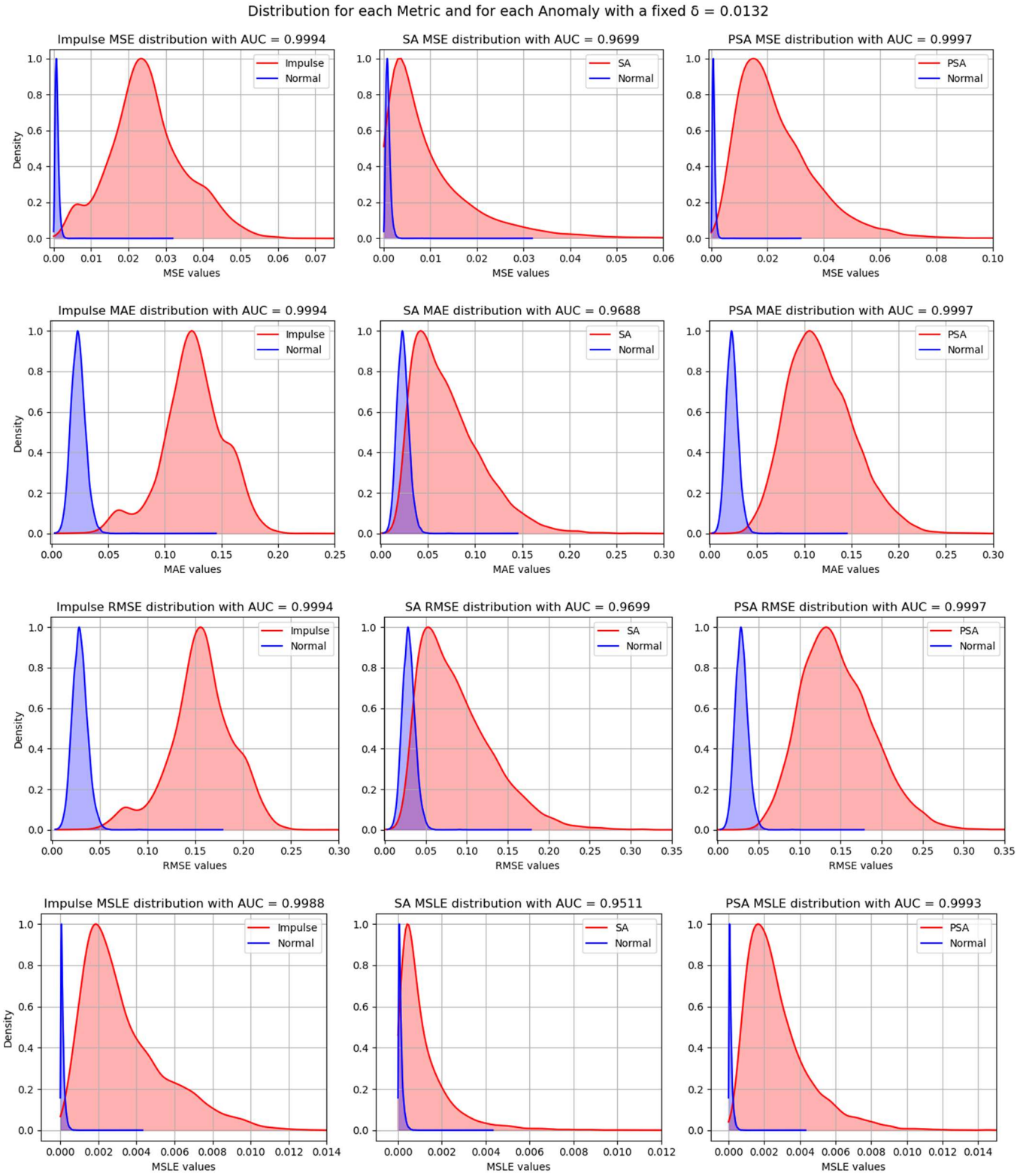


Figura 14: fissato un valore di δ, si confrontano le distribuzioni normali e anomale per ciascuna anomalia e per ciascuna metrica.

in relazione le singole curve di MSE_a con la curva di MSE_{na}, per l'anomalia SA e solo per la metrica MSE. Come si può osservare, la curva blu, rappresentante la distribuzione di MSE_{na} ha una bassa deviazione standard, concentrandosi nell'intorno del picco, in prossimità dello zero. Viceversa, la curva rossa, rappresentante la distribuzione di MSE_a, all'aumentare di δ tende a spostarsi verso destra e ad incrementare la deviazione standard. Gli assi delle ascisse sono stati riproporzionati per ogni riga in modo da visualizzare meglio entrambe le distribuzioni. Per le stesse ragioni, anche i picchi sono stati normalizzati, con un valore massimo di densità pari a 1. Oltre al valore di δ associato a ciascuna distribuzione, è stato riportato anche il valore di AUC, che si ricorda essere la probabilità che il modello classifichi correttamente i valori per le due classi. Come previsto dall'analisi dei grafici in figura 12, con l'incremento di δ il detector riesce a distinguere più facilmente le due classi. Tuttavia, a causa della sovrapposizione tra le due curve, la distinzione non potrà essere perfetta, come indicato anche dal valore di AUC, che non raggiunge il massimo, ovvero 1. La figura in questione, però, si limita solo ad una metrica e per una sola anomalia. Volendo visualizzare le diverse distribuzioni anomale paragonate con quelle normali, per diverse metriche e per diverse anomalie, si faccia riferimento alla figura 14. Dall'illustrazione è possibile avere una panoramica di come si differenziano le curve per diverse anomalie e metriche. Ovviamente, affinché le curve possano essere paragonabili tra di loro, si è scelto e fissato il valore di δ, che quindi non cambia da grafico a grafico. Ad ogni modo, si può osservare come per certe anomalie e per certe metriche si riesca ad ottenere un valore di AUC maggiore, traducendosi in una maggiore probabilità di rilevare l'anomalia.

Infine, la figura 15 mostra chiaramente come il valore di AUC vari con l'intensità dell'anomalia δ, confermando che un'intensità maggiore aumenta la probabilità che il modello classifichi correttamente. Inoltre, si evidenzia che l'uso della metrica MSLE comporta valori di AUC più bassi, per le ragioni discusse in precedenza. Questo suggerisce che, sebbene l'aumento dell'intensità dell'anomalia migliori la capacità del modello di distinguere tra le classi, l'adozione della metrica MSLE può ridurre l'efficacia complessiva della classificazione, probabilmente a causa della maggiore sensibilità della MSLE alle differenze nei valori di errore.

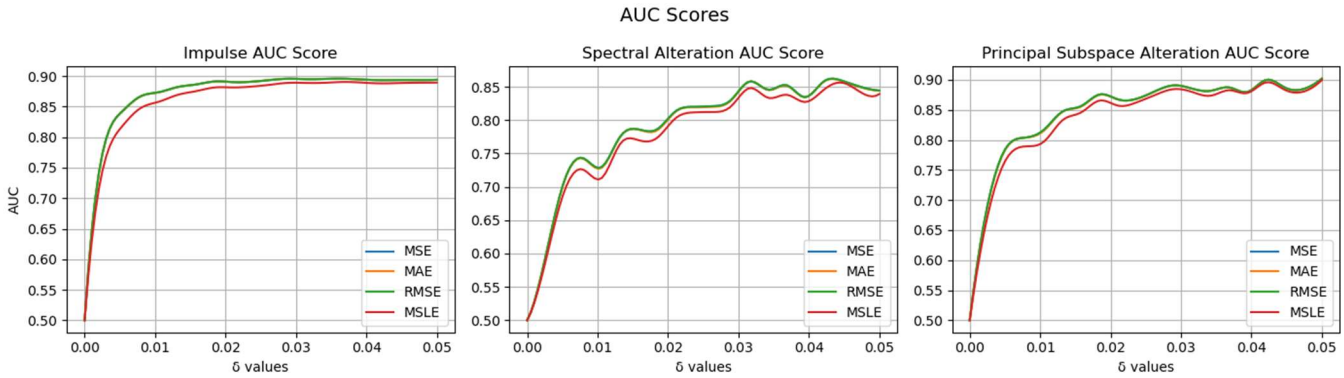


Figura 15: andamento dei valori di AUC all'aumentare di δ, per ciascuna anomalia e per ciascuna metrica.

Conclusioni

Nel primo capitolo si sono studiati i principi base che stanno dietro al Compressed Sensing, metodo di compressione ampiamente utilizzato per i suoi vantaggi che offre. D'altro canto, il problema principale del CS è legato alla fase di ricostruzione (decoding), la quale richiede la risoluzione di un problema di ottimizzazione per trovare la giusta rappresentazione sparsa, tale da poter ricostruire il segnale proiettando i dati compressi dal dominio $\mathbb{R}^m$ al dominio $\mathbb{R}^n$ con il minor errore possibile. Questo è vero se il supporto della rappresentazione sparsa non è noto. Viceversa, dato il supporto, si tratta solo di effettuare una operazione di pseudo-inversa, la cui soluzione è unica. Dunque, nasce l'idea di sviluppare una rete neurale in grado di indovinare il supporto della rappresentazione sparsa data la misura. Ma l'allenamento della rete richiede la definizione di un dataset, composto dai dati, ovvero le misure, e dalle labels, ovvero i supporti. Di conseguenza, occorre cercare e determinare tali supporti. Differenziandosi dal classico decoder BP per la ricerca della rappresentazione sparsa, gli Autori hanno proposto un algoritmo, chiamato Greedy, che permette di cercare i supporti della rappresentazione sparsa come quei vettori che massimizzano il valore di RSNR. Per ogni istanza, possono essere compiuti al più n passi. Questa procedura, da effettuare su tutte le due milioni di finestre che compongono il dataset delle acquisizioni, può risultare particolarmente onerosa in termini computazionali. Per questo motivo, il primo obiettivo di questo elaborato è stato quello di cercare e definire un algoritmo alternativo, più leggero ma comunque equivalente in termini di prestazioni e risultati. L'algoritmo proposto identifica i supporti come quegli elementi della rappresentazione sparsa che permettono di raggiungere un certo livello di energia rispetto a quella totale della singola istanza. Con i supporti così determinati, si è in grado di ottenere un certo valore di RSNR in fase di ricostruzione, che però non è necessariamente il massimo raggiungibile. Di conseguenza, il valore di RSNR è legato al livello di energia di riferimento, che diventa un grado di libertà da impostare, introducendo un problema di ricerca esaustivo NP-hard. Pertanto, definito un intervallo discreto di valori energetici, il livello di energia di riferimento da scegliere sarà quello che permetterà di massimizzare l'RSNR. Questo per ogni dimensione m. A questo punto il dataset può essere formato e l'allenamento della rete avviato. Sfruttando il test set, composto dal 5% dei dati del dataset, è possibile valutare l'accuratezza della rete nello stimare i supporti associati alle misure. Nonostante l'inconveniente sulla scelta del grado di libertà, l'algoritmo proposto, oltre a ridurre i costi computazionali, permette anche di generare un dataset dal quale la rete riesce ad allenarsi molto bene, fornendo dei risultati pressoché accurati, come discusso nell'ultimo sottoparagrafo. In conclusione, l'algoritmo proposto soddisfa

pienamente i requisiti imposti, ovvero leggerezza computazionale ed equivalenza con i risultati ottenuti dagli Autori.

Nel Capitolo 2, si è approfondito il concetto di anomalie, esplorando le loro cause e le diverse tipologie che possono influenzare i segnali biomedici, in particolare gli ECG. Le anomalie nei segnali ECG possono derivare da vari fattori, tra cui errori di misurazione, interferenze ambientali o condizioni patologiche del paziente. Queste anomalie possono manifestarsi in diverse forme, come rumore impulsivo, alterazioni spettrali (SA) o variazioni nel sottospazio principale (PSA), ognuna delle quali presenta caratteristiche differenti.
Partendo dalle basi gettate nel capitolo uno, si è definito un processo di Anomaly Detection.
Questo processo inizia con la fase di encoding del CS, durante la quale vengono ottenute le misure iniziali dei segnali. Successivamente, viene utilizzato il TSOC per stimare i supporti delle rappresentazioni sparse, potendo procedere con la fase di decoding per ricostruire i segnali originali. Dopo la ricostruzione, si riapplica la matrice di sensing per ottenere le misure "ricostruite", completando così un ciclo di encoding-decoding-encoding.
A questo punto, possono essere utilizzate diverse metriche per valutare la qualità delle ricostruzioni e rilevare eventuali anomalie. Le metriche utilizzate includono l'errore quadratico medio (MSE), l'errore assoluto medio (MAE), la radice dell'errore quadratico medio (RMSE) e l'errore quadratico medio logaritmico (MSLE). Queste metriche permettono di ottenere due distribuzioni distinte: una per la coppia di misure normali y_{na} e $\hat{y}_{na}$, e una per la coppia di misure anomale y_a e $\hat{y}_a$. Per valutare l'efficacia del modello nel distinguere tra le due classi, viene implementato l'AUC (Area Under the Curve), che è uno strumento di Self-Assessment che misura la probabilità che il modello classifichi correttamente le due classi. Infine, i risultati ottenuti mostrano come all'aumentare dell'intensità delle anomalie, il modello riesce a distinguere con maggiore precisione tra le misure normali e quelle anomale. Questo è particolarmente evidente per le anomalie di intensità crescente, dove l'AUC raggiunge valori elevati, indicando un'alta probabilità di classificazione corretta. In conclusione, il capitolo due ha dimostrato come l'integrazione del Compressed Sensing con tecniche di Anomaly Detection possa migliorare significativamente la rilevazione delle anomalie nei segnali biomedici. L'approccio proposto non solo riduce il volume di dati da elaborare, diminuendo i consumi energetici e migliorando la velocità di elaborazione, ma offre anche una maggiore precisione nella rilevazione delle anomalie. Questo è cruciale in applicazioni come il monitoraggio biomedico, dove è essenziale rilevare rapidamente le anomalie per intervenire tempestivamente. L'uso combinato di metriche diverse e dell'AUC come strumento di

valutazione ha permesso di ottenere una visione completa delle prestazioni del modello, evidenziando i punti di forza e le aree di miglioramento. I risultati ottenuti sono promettenti e suggeriscono che ulteriori ricerche e ottimizzazioni potrebbero portare a miglioramenti ancora maggiori, rendendo questo approccio una soluzione valida ed efficiente per l'Anomaly Detection nei segnali biomedici in generale.

Bibliografia

[1] D. L. Donoho, "Compressed Sensing", *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289-1306, Apr. 2006, doi: 10.1109/TIT.2006.871582.

[2] D. Gangopadhyay, E. G. Allstot, A. M. R. Dixon, K. Natarajan, S. Gupta, and D. J. Allstot, "Compressed sensing analog front-end for biosensor applications," IEEE J. Solid State Circuits, vol. 49, no. 2, pp. 426–438, Feb. 2014, doi: 10.1109/JSSC.2013.2284673.

[3] F. Pareschi, P. Albertini, G. Frattini, M. Mangia, R. Rovatti, and G. Setti, "Hardware-algorithms co-design and implementation of an analog-to-information converter for biosignals based on compressed sensing," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 149–162, Feb. 2016, doi: 10.1109/TBCAS.2015.2444276.

[4] A. M. R. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ECG and EMG wireless biosensors," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 2, pp. 156–166, Apr. 2012, doi: 10.1109/TBCAS.2012.2193668.

[5] M. Shoaran, M. H. Kamal, C. Pollo, P. Vandergheynst, and A. Schmid, "Compact low-power cortical recording architecture for compressive multichannel data acquisition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 6, pp. 857–870, Dec. 2014, doi: 10.1109/TBCAS. 2014.2304582.

[6] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005, doi: 10. 1109/TIT.2005.858979.

[7] H. Zhu, G. Leus, and G. B. Giannakis, "Sparsity-cognizant total least-squares for perturbed compressive sampling," *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 2002–2016, May 2011, doi: 10.1109/TSP. 2011.2109956.

[8] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2011, pp. 2168–2172, doi: 10.1109/ISIT.2011.6033942.

[9] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007, doi: 10.1109/TIT.2007. 909108.

[10] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, May 2009, doi: 10.1016/j.acha.2008.07.002.

[11] F. Pareschi *et al.*, "Energy analysis of decoders for rakeness-based compressed sensing of ECG signals," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 6, pp. 1278–1289, Dec. 2017, doi: 10.1109/TBCAS. 2017.2740059.

[12] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009, doi: 10.1137/080716542.

[13] A. Mousavi, A. B. Patel, and R. G. Baraniuk, “A deep learning approach to structured signal recovery,” in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1336–1343, doi: 10. 1109/ALLERTON.2015.7447163.
[14] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, “Deep fully connected networks for video compressive sensing,” *Digit. Signal Process.*, vol. 72, pp. 9–18, Jan. 2018, doi: 10.1016/j.dsp.2017.09.010.
[15] J. Zhang and B. Ghanem, “ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1828–1837, doi: 10. 1109/CVPR.2018.00196.
[16] B. Sun, H. Feng, K. Chen, and X. Zhu, “A deep learning framework of quantized compressed sensing for wireless neural recording,” *IEEE Access*, vol. 4, pp. 5169–5178, 2016, doi: 10.1109/ACCESS.2016. 2604397.
[17] M. Mangia, L. Prono, A. Marchioni, F. Pareschi, R. Rovatti, and G. Setti, “Deep neural oracles for short window optimized compressed sensing of biosignals,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 3, pp. 545–557, Jun. 2020, doi: 10.1109/TBCAS.2020.2982824.
[18] L. Prono, M. Mangia, A. Marchioni, F. Pareschi, R. Rovatti and G. Setti, "Deep Neural Oracle With Support Identification in the Compressed Domain," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 458-468, Dec. 2020, doi: 10.1109/JETCAS.2020.3039731.
[19] https://it.wikipedia.org/wiki/Rete_neurale_artificiale
[20] M. Kaloev and G. Krastev, "Comparative Analysis of Activation Functions Used in the Hidden Layers of Deep Neural Networks," *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey, 2021, pp. 1-5, doi: 10.1109/HORA52670.2021.9461312.
[21] https://it.wikipedia.org/wiki/Errore_quadratico_medio
[22] J. Cao, Z. Su, L. Yu, D. Chang, X. Li and Z. Ma, "Softmax Cross Entropy Loss with Unbiased Decision Boundary for Image Classification," *2018 Chinese Automation Congress (CAC)*, Xi'an, China, 2018, pp. 2028-2032, doi: 10.1109/CAC.2018.8623242.
[23] L. Guo, M. Li, S. Xu and F. Yang, "Application of Stochastic Gradient Descent Technique for Method of Moments," *2020 IEEE International Conference on Computational Electromagnetics (ICCEM)*, Singapore, 2020, pp. 97-98, doi: 10.1109/ICCEM47450.2020.9219400.
[24] L. S. Riza, M. A. Ashari and R. Megasari, "The Implementation of Gradient Descent Based Methods Using Parallel Computing in R for Regression Tasks," *2018 International Symposium on Advanced Intelligent Informatics (SAIN)*, Yogyakarta, Indonesia, 2018, pp. 37-42, doi: 10.1109/SAIN.2018.8673361.
[25] M. Lamine and S. -C. Kim, "Introduction of Optimization Algorithm for Adaptive Gradient," *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Bali, Indonesia, 2023, pp. 837-839, doi: 10.1109/ICAIIC57133.2023.10066993.

[26] A. Enttsel, S. Onofri, A. Marchioni, M. Mangia, G. Setti and R. Rovatti, "A General Framework for the Assessment of Detectors of Anomalies in Time Series," in *IEEE Transactions on Industrial Informatics*, doi: 10.1109/TII.2024.3413359.
[27] Machine Learning part2, slides of Statistics and Architectures for Big Data Processing M, prof. Mauro Mangia and Francesco Conti.
[28] Richard J. Preen, Stewart W. Wilson and Larry Bull, "Autoencoding with Classifier System", in *IEEE Transactions on Evolutionary Computation*, doi: 10.1109/TEVC.2021.3079320.
[29] S. Sadafule, S. Sarkar and S. Wu, "G-AUC: An improved metric for classification model selection," *2022 26th International Computer Science and Engineering Conference (ICSEC)*, Sakon Nakhon, Thailand, 2022, pp. 100-104, doi: 10.1109/ICSEC56337.2022.10049319.