UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea triennale in Ingegneria Biomedica

STIMA DELL'ATTIVAZIONE MUSCOLARE DURANTE IL CAMMINO TRAMITE APPROCCIO MACHINE LEARNING

DETECTION OF MUSCOLAR ACTIVATION DURING WALKING USING A MACHINE LEARNING APPROACH

Relatore:

Prof. Francesco Di Nardo

Tesi di Laurea di:

Antonio Nocera

Correlatore:

Prof. Christian Morbidoni

A.A. 2019 / 2020

Abstract

Il consenso in letteratura è minimo sulle metodiche da utilizzare per l'individuazione dell'attivazione nei segnali EMG. Questa tesi propone un approccio basato su tecniche machine learning per il riconoscimento degli istanti di transizione da rumore a segnale, corrispondente alle attivazioni muscolari. Il metodo proposto si compone di una fase di pre-processamento del segnale, in cui tre diverse tipologie di processamento sono state valutate (scalogramma, inviluppo lineare e root mean square value) e di una fase di apprendimento supervisionato, basata su reti neurali, in grado di individuare le transizioni (on/off). Gli esperimenti sono stati effettuati su un dataset di 1040 segnali generati sinteticamente che sono stati utilizzati per formare un trainset di 500 segnali e un testset di 540. Il metodo proposto è stato confrontato con l'algoritmo double threshold dimostrando di avere performance generalmente superiori, suggerendo quindi un'effettiva applicabilità del metodo.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Il segnale elettromiografico di superficie (sEMG) è un bio-segnale emesso durante la contrazione muscolare, che si origina dalla somma degli impulsi di numerose unità motorie (MUAPs) localizzate nelle prossimità di una coppia di elettrodi di superficie posta sul muscolo considerato. L'analisi del segnale EMG può diventare uno strumento per lo studio sia di informazioni sul sistema nervoso periferico, con l'analisi di un solo muscolo, sia di informazioni sul sistema nervoso centrale. In quest'ultimo caso, la valutazione di più muscoli che collaborano ad una specifica task motoria, come nel caso della gait analysis, permette di comprendere quale sia la strategia motoria del cervello anche in casi patologici. Infatti, citando [11], 'malattie neuromuscolari possono indurre cambiamenti tipici nel controllo centrale o nelle proprietà fisiologiche delle fibre muscolari, che porta a modifiche caratteristiche nell'aspetto del segnale EMG'. Per questo motivo il segnale EMG è largamente utilizzato in medicina e chinesiologia.

Inoltre, nel segnale EMG, sono presenti dei parametri in funzione dello sviluppo temporale che sono tipicamente utilizzati nello studio della posizione del corpo e del suo movimento. Uno dei più comuni di questi parametri è l'istante di inizio(onset) o cessione(offset) dell'attività muscolare [8]. Infatti, l'individuazione accurata del timing di attivazione muscolare è fondamentale nello studio del controllo motorio e nelle analisi cliniche e biomeccaniche. Lo stesso articolo [8], dove viene riportato che le differenze temporali tra lo stimolo e l'inizio della contrazione muscolare potrebbero essere tanto basse quanto 20ms, dimostra che per avere condizioni sperimentali ottimali l'accuratezza della predizione di onset e offset sia decisiva.

I primi metodi che venivano inizialmente applicati al riconoscimento dell'onset e dell'offset consistevano in un'ispezione visiva del segnale da parte di un esaminatore

esperto. È chiaro che questa metodica presenti molte problematiche per la dipendenza dei risultati da una valutazione soggettiva, la quale diventa anche più complessa nel caso di segnali particolarmente rumorosi. In aggiunta, l'applicazione di questa metodica diventa particolarmente dispendiosa se si deve lavorare con grandi datasets. Tutto ciò ha portato alla ricerca di metodiche automatizzate per l'individuazione del timing di attivazione nel segnale sEMG e ha prodotto molte proposte metodologiche, ma nessuna che sia definitiva o considerata un 'gold standard'.

Citando e traducendo un'analisi degli algoritmi utilizzati fatta nel 2017 [7]: ' La vasta gamma di algoritmi di EMG Onset detection suggerisce che nessuno degli algoritmi attualmente utilizzati siano sufficienti per l'ampia applicazione dei segnali EMG in biomeccanica. Le nuove metodologie, TKEO e Sample Entropy, derivano dalla ricerca nell'area dell'acustica ed elettrocardiografia rispettivamente. Quindi, è ragionevole pensare di estendere la ricerca degli algoritmi a quelli tipicamente utilizzati in altre aree'. Perciò, l'obiettivo di questa tesi è quello di proporre un approccio innovativo per la detection di onset e offset tramite approccio deep learning che possa essere utilizzato in particolare nella gait analysis.

Gli esperimenti di questa tesi sono stati effettuati su un dataset di segnali sintetici e l'architettura di rete utilizzata è un multiperceptrone; i risultati ottenuti, valutati tramite precision, recall e f1score e tramite l'utilizzo di latency sugli istanti predetti sono stati paragonati con l'algoritmo di Bonato, approccio standard in letteratura, e sembrano essere promettenti per una possibile applicazione della rete neurale.

Capitolo 2

Gait Analisys e Stato dell'Arte

2.1 Gait Analysis

Questa digressione sulla gait analysis non vuole essere un particolare approfondimento della tematica, ma una breve introduzione delle fasi riconosciute durante l'andatura, evidenziando in che modo l'attivazione muscolare, e quindi l'individuazione del suo inizio (onset) e della sua fine (offset) siano in relazione con alcune funzioni di base della camminata. Per i lettori interessati all'argomento viene consigliato il libro [9], il quale ispira tutto questo capitolo.

2.1.1 *'Gait Cycle'**

Un ciclo di andatura, o 'gait cycle' (GC), è la serie di eventi tramite i quali una gamba viene portata in avanti ed è composto essenzialmente da due macro-fasi:

1. Stance, cioè la fase durante la quale la gamba fornisce il supporto e il piede tocca il terreno; l'inizio di questa fase è dettato solitamente dall'evento Heel Strike o Initial Contact e rappresenta circa il 60% del totale passo.

2. Swing, cioè la fase in cui la gamba è la parte mobile che avanza con il piede che non tocca il terreno fino al successivo sito di supporto; inizia con l'evento del Toe Off e rappresenta circa il 40% del GC.

È possibile effettuare un ulteriore suddivisione della fase di stance basata sul contatto del piede [Fig. 2.1]:

1. Initial Two Limb Support Stance, che corrisponde al 10% del GC.

2. Single Limb Support Stance, che rappresenta il 40% del GC; questa fase coincide con lo swing dell'altra gamba

3. Terminal Two Limb Support Stance, un altro 10% del GC.

**il termine gait è stato tradotto sia con andatura sia con il termine passo; con quest'ultimo si intende il concetto di 'stride' quindi il movimento del piede che va da un punto di appoggio al successivo.*

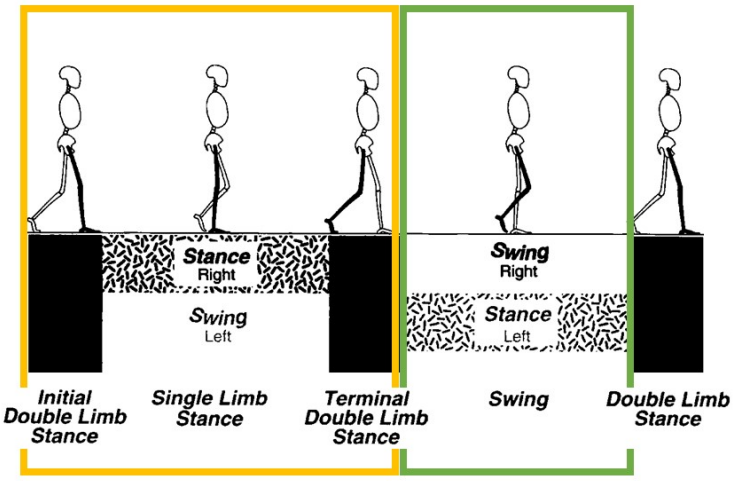


Figura 2.1: *Questa immagine ripresa del libro [9] spiega brevemente le suddivisioni della fase di stance(in giallo) e swing(in verde) e la loro relazione con i pattern di contatto dei piedi con il terreno.*

2.1.2 Le sottofasi del ciclo di andatura

Nella pratica i ricercatori hanno riconosciuto 8 sottofasi funzionali, ognuna delle quali è caratterizzata da uno schema di movimenti in sinergia per raggiungere uno specifico obiettivo funzionale. La combinazione di queste sottofasi in sequenza permette alla gamba di compiere 3 compiti essenziali:

1. Accettazione del peso; questo è sicuramente il compito più faticoso per il corpo, che necessita di tre pattern funzionali: assorbimento dello shock, stabilità della gamba e preservazione della progressione. Si compone delle sottofasi di:

    - Initial Contact (0-2% GC), durante la quale la gamba è posizionata per l'impatto con il terreno e avviene l'evento dell'Heel Strike. Il posizionamento della gamba comincia già da prima con la fine della fase di swing
    - Loading Response(0-10% GC), che corrisponde con la fase di Initial Two Limb Support Stance. Come comprendiamo dallo stesso nome, durante questa sottofase si innescano dei meccanismi in risposta al repentino aumento di carico

sulla gamba di supporto che hanno lo scopo di assorbire parte dell'urto con il terreno. Questi meccanismi sono brevemente illustrati nella seguente sezione.

2. Supporto su una sola gamba; questo compito inizia nel momento in cui una delle due gambe viene sollevata e finisce con l'arrivo della stessa ad un successivo punto di appoggio. È composto da due sottofasi che sono differenziate essenzialmente dai loro meccanismi di progressione del movimento e sono:

   - Mid Stance(10-30% GC) , caratterizzato dal movimento di progressione chiamato 'ankle rocker'
   - Terminal Stance(30-50% GC) , che conclude la progressione con il movimento detto 'forefoot rocker'

3. Avanzamento della gamba; questo è un compito, che inizia con la fine della stance e l'evento di Toe-off e continua durante la fase di swing; consiste nel sollevamento dell'arto, nella sua progressione in avanti e nel suo finale posizionamento per l'inizio della successiva fase di stance; si riconoscono in particolare le seguenti sottofasi funzionali:

   - Pre-Swing (50-60%); la fase finale di stance, chiamata anche 'weight release' o 'weight transfer', perché, a seguito di una fase di doppio supporto, comincia il sollevamento della gamba e il suo adeguato posizionamento per l'inizio dello swing. Le successive tre fasi dividono equamente un 40% del ciclo di andatura (GC).
   - Initial Swing; questa fase ha l'obiettivo di sollevare il piede ad una distanza tale da non urtare il terreno (Foot clearance of the floor)
   - Mid Swing; questa fase, mantenendo il piede a distanza dal terreno, ha il compito di far avanzare la gamba grazie al momento acquisito.
   - Terminal Swing; la fase inizia con la tibia in posizione verticale e finisce con un'estensione attiva del ginocchio, controllata dai quadricipiti, per il giusto posizionamento della gamba, la quale dovrà assorbire con l'evento di Heel Strike buona parte del peso corporeo in un piccolo intervallo temporale.

2.1.3 Le funzioni di base e l'attivazione muscolare

Le funzioni dell'unità locomotrice del corpo, cioè l'insieme di bacino e gambe, possono essere divise in quattro categorie:

1. Stabilità
2. Propulsione / Progressione
3. Assorbimento dello shock

4. Conservazione dell'energia

Nelle seguenti sottosezioni saranno descritte queste funzioni senza andare nel particolare, dando però la dovuta attenzione al ruolo dell'attivazione muscolare.

Stabilità

La stabilità dipende per la maggior parte dall'allineamento del peso del corpo (body vector). Tramite la relativa posizione di un'articolazione e del vettore peso, è possibile comprendere quali siano i momenti torcenti su quella particolare articolazione e quale sia la necessaria risposta di muscoli e legamenti. Infatti, in 'quiet standing', cioè la situazione con il corpo eretto e il peso distribuito equamente tra i piedi, l'intervento muscolare è minimo e teoricamente non sarebbe necessaria alcuna azione muscolare per mantenere il bilanciamento. D'altra parte, durante la camminata, la più vicina situazione di allineamento del corpo è presente nel solo istante a metà della stance e si fa quindi necessario un continuo intervento muscolare e legamentoso per il mantenimento della stabilità dinamica. Il compito cruciale dei muscoli è quello di opporsi all'influenza della gravità e del momento, i quali minacciano questo equilibrio.

Progressione

Lo step iniziale del cammino è causato da un controllo della caviglia nella gamba di supporto da parte del tibiale anteriore, il quale agisce per far avanzare la tibia, permettendo la caduta in avanti del peso corporeo, che è il principale mezzo di propulsione; il soleo interviene leggermente per rallentare il movimento. Una volta che questa caduta è messa in atto sono presenti dei meccanismi per direzionare la forza, che è diretta principalmente verso il terreno, e per preservare il momento: Heel Rocker, Ankle Rocker, Forefoot Rocker, Pre-Swing knee Flexion, Swing phase hip flexion e Swing phase knee extension sono i meccanismi messi in atto. Non si andrà nel particolare di questi schemi motori, ma verranno evidenziati i modi con cui la contrazione del muscolo partecipa a questa progressione:

1. Decelerazione e trasferimento dell'effetto di progressione
   Durante l'heel rocker, l'intervento dei pretibiali è determinante per rallentare la caduta del piede e far avanzare la tibia; quest' effetto di progressione in avanti è poi trasferito alla coscia tramite l'attivazione dei quadricipiti, che non solo riducono la velocità di flessione del ginocchio, conseguente all'heel rocker, ma legano il movimento della tibia al femore. La collaborazione di questi due gruppi muscolari permette il reindirizzamento della forza, la quale viene indirizzata in avanti.

2. Effetto stabilizzatore
   Durante l'ankle rocker, un aspetto cruciale è quello della contrazione del soleo, il quale con l'aiuto del gastrocnemio rende la tibia una stabile base per l'estensione del ginocchio e permette l'avanzamento controllato della tibia stessa.

3. Propulsione
   Durante la fase di Pre-Swing, l'attivazione residuale del gastrocnemio e il rapido trasferimento del peso sull'altra gamba permette la rotazione del piede sull'articolazione metatarsofalangea e una conseguente flessione plantare e del ginocchio. Altro esempio di propulsione dove l'intervento muscolare è determinante, può essere trovato in fase di Terminal Swing, dove l'attivazione del quadricipite permette un'estensione attiva del ginocchio che contribuisce al movimento della gamba in swing e permette un adeguato posizionamento dell'arto, il quale dovrà sostenere la 'caduta' del peso corporeo.

Assorbimento dello shock

Tre sono i meccanismi di assorbimento dello shock che si verificano durante la Loading Response. Questi sono necessari a causa del repentino trasferimento del peso su una singola gamba dopo un periodo di caduta libera in cui il piede, posizionato per la successiva stance, 'cade' per circa 1cm di altezza. Di seguito sono elencati i meccanismi, mettendo in risalto il contributo della contrazione muscolare:

1. Flessione plantare della caviglia
   L'azione muscolare è stata parzialmente già descritta nella precedente sezione; comunque, l'azione dei pretibiali decelera considerevolmente il movimento di flessione plantare (heel rocker) e ritarda in questo modo il contatto del piede con il terreno, rendendo quindi lo stesso meno repentino.

2. Flessione del ginocchio
   Questo movimento è una reazione all'heel rocker ed una conseguenza dell'attivazione dei pretibiali; infatti il femore perde parte del suo supporto dato dalla tibia, la quale segue l'avanzamento della gamba. Pertanto, l'articolazione del ginocchio flette ed è presente un'attivazione dei quadricipiti che limitano questo movimento e trasferiscono parte della forza sulla massa muscolare della coscia.

3. Tilt controlaterale del bacino
   Questo movimento del bacino è diretta conseguenza del repentino carico del peso sulla gamba di supporto e il successivo scarico del peso sull'altra gamba, la quale viene sollevata; la rotazione del bacino sul piano coronale viene limitata dai muscoli abduttori. Comprendiamo come nuovamente l'intervento muscolare sia cruciale per l'assorbimento dello shock.

Conservazione dell'energia

La camminata è uno schema motorio che è reso estremamente efficiente dal sistema nervoso centrale grazie a due principali meccanismi:

1. Controllo della posizione del centro di massa (o centro di gravità C/G); questo riduce il lavoro necessario durante la camminata, in quanto riduce lo spostamento del centro di massa. Infatti, attraverso dei cambiamenti nell'allineamento del bacino, la variazione della posizione del C/G è pari a circa due centimetri sull'asse longitudinale e quattro su quello trasversale.

2. Controllo selettivo dell'attivazione muscolare, che consiste nel modulare sia il timing sia l'intensità dell'attivazione muscolare in modo tale che sia la minima richiesta. La contrazione muscolare viene essenzialmente limitata al solo ruolo di effetto antagonista ai momenti torcenti, i quali creano una condizione di instabilità. Se possibile, l'azione muscolare è sempre sostituita dal passivo posizionamento degli arti e dal momento disponibile. Infatti, il movimento della camminata non è altro che un insieme di meccanismi volti a sfruttare l'iniziale forza data dalla 'caduta libera' del corpo, la quale culmina con l'evento di Heel Strike. In questo schema i muscoli hanno un ruolo determinante nel decelerare il processo e nell'assorbire il peso e rispondono essenzialmente ad una torsione fisiologica causata da un disallineamento del 'body vector' con una specifica articolazione.

Detto questo, analizziamo, per esempio, la figura 2.2 che illustra la torsione fisiologica sull'articolazione della caviglia durante la stance di un passo e la risposta di gastrocnemio, un muscolo flessore, e del tibiale anteriore, muscolo dorsiflessore.

Notiamo che nella fase di 'Loading Response' sia presente una leggera flessione plantare, dovuta all'assorbimento dello shock tramite 'heel rocker', e a questa torsione corrisponde l'attivazione del tibiale anteriore, il quale rallenta il movimento. Al contrario, durante tutta la fase di stance abbiamo una condizione di dorsiflessione a cui corrisponde l'azione muscolare del gastrocnemio fino alla fine della terminal stance; in fase di Pre-Swing la condizione di dorsiflessione è favorevole alla progressione in avanti del corpo e permette l'evento del Toe-off, quindi non c'è un controllo da parte del gastrocnemio.

Durante la fase di swing non è presente una torsione fisiologica sulla gamba sollevata, però l'azione muscolare è volta a mantenere il peso della gamba e del piede in swing. Infatti, notiamo che l'azione dei pretibiali è presente dalla fine del Pre-Swing alla fine della fase di Swing stessa, poiché partecipano alla funzione di 'foot clearance', cioè la funzione per la quale il piede deve essere alzato per evitare l'urto con il terreno e contribuiscono al posizionamento adeguato del piede per il successivo evento di Heel Strike.

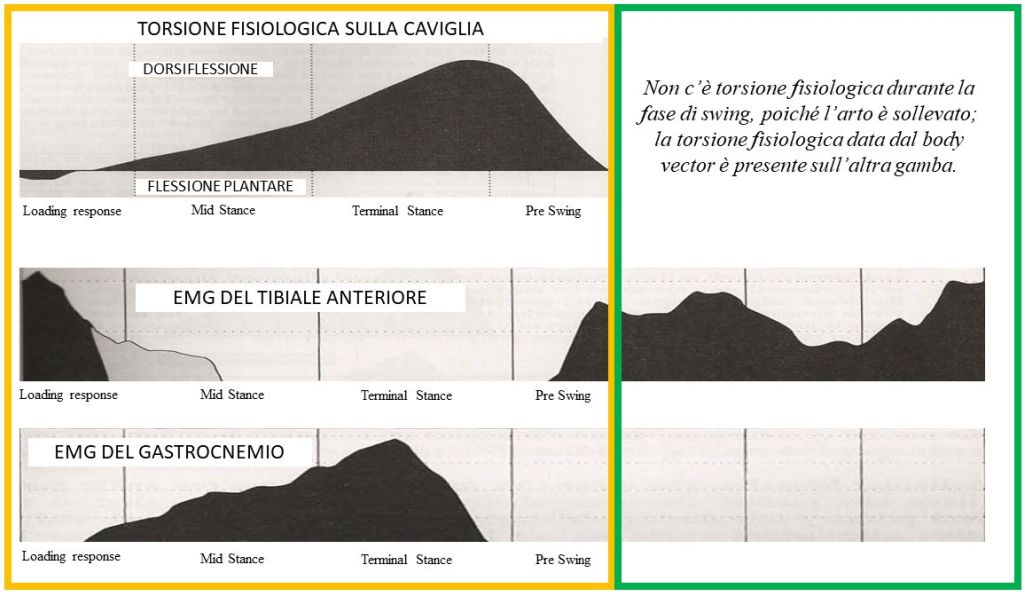


Figura 2.2: *Il primo grafico mostra la torsione fisiologica calcolata in unità fisiologiche (BWXLL) in funzione delle fasi di stance; I due grafici successivi riportano due grafici EMG che evidenziano con le zone nere l'intensità dell'attivazione muscolare come percentuale rispetto al massimo valore del manual muscle test (% MMT)*

2.2 Lo stato dell'arte

La ricerca bibliografica è stata condotta dal 3/07/2020 al 9/07/2020 attraverso tre strumenti di ricerca: Pubmed, Scopus e ResearchGate. L'obiettivo è stato quello di trovare nella letteratura le soluzioni proposte al problema di onset/offset detection tramite approccio machine learning, deep learning e tramite approcci tradizionali.
Qualsiasi sia l'approccio, possiamo in generale individuare tre fasi, come fa notare l'articolo [3] [Fig.2.3]:

1. pre-processamento del segnale
2. detector unit
3. post-processamento

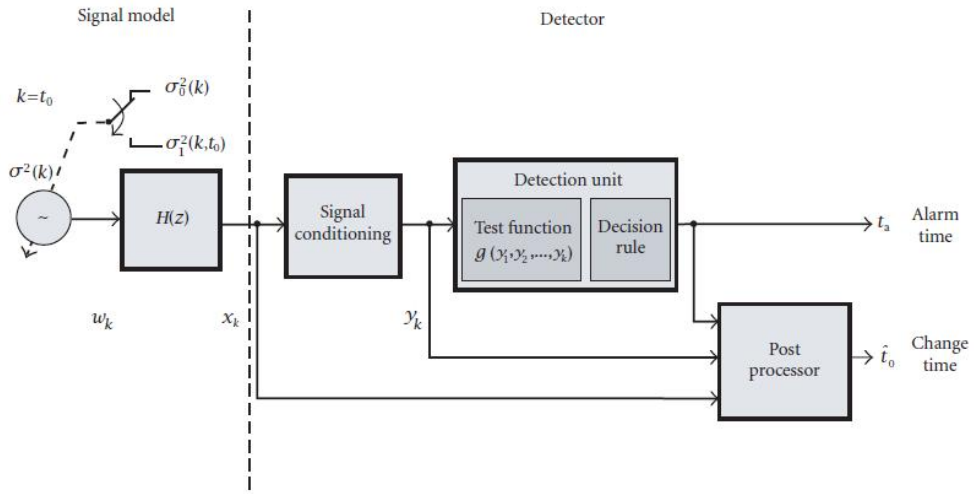


Figura 2.3: *Schema preso dall'articolo [3]*

I primi algoritmi utilizzati negli anni 90' erano basati sull'implementazione di soglie in ampiezza applicate ad un segnale filtrato[8] [1] [6]. Ad esempio, nel 96' P W. Hodges e Bang H. Bui[8] propongono un algoritmo soglia con un pre-processamento tramite passa-basso a 50 Hz e con una soglia posta a 3 standard deviation (std) dalla baseline; questo valore veniva utilizzato per analizzare le transizioni all'interno di una finestra da 25 ms, che rappresentava un'attivazione se tutti gli istanti lo superavano.

Un altro esempio è l'articolo di Bonato[1], il quale nel 98' propone un approccio double threshold basato su considerazioni statistiche sul segnale EMG, spiegato nel particolare nella sezione 'discussione'.
Nel 2001, tramite un confronto sistematico nell'articolo [3] vengono confrontati questi algoritmi di soglia con algoritmi più performanti basati sulla decisione statisticamente ottimale per l'individuazione di istanti di transizione come AGLRstep e AGLRramp (Approximated generalized likelihood-ratio). Questi algoritmi producono la 'decision rule',

cioè la regola per l'individuazione degli istanti di transizioni, da un confronto tra la stima della varianza di rumore e una stima della varianza su una test window; a seguito di un filtro pre-whitening, l'AGLRstep detector può essere riassunto nei seguenti punti presi dall'articolo [3]:

1. stima della varianza del rumore dai primi M (posto a 200) campioni del segnale come media dell'energia in quella che viene chiamata finestra di reference.

$$\sigma_0^2(k, \theta_0) = \theta_0 = \frac{1}{M} \sum_{i=1}^{M} y_i^2$$

   Questa stima produce un valore costante θ_0.

2. stima della varianza su una test window; il detector cambia da AGLRstep ad AGLR-ramp a seconda della formula utilizzata per questa stima; qui viene riportata la stima per l'algoritmo step. Perciò, l'algoritmo assume che la varianza di una test window di 25 campioni, che scorre su tutto il segnale, sia una costante θ_1 definita come segue:

$$\theta_1 = \frac{1}{k - j + 1} \sum_{i=j}^{k} y_i^2$$

3. la 'decision rule' permette di ricavare istanti di allarme t_a come il minimo valore del campione k per cui nella test window, da k-25 a k, g_k, cioè la test function (log-likelihood ratio), superi un arbitrario threshold posto a 10. A questo punto l'istante di onset/offset t_0 predetto è determinato dal massimo valore che assume la funzione di log-likelihood ratio, nell'intervallo dei successivi 100 campioni a partire da un determinato istante di allarme t_a, in funzione di t_a.

$$t_a = min(k >= W : g_k >= h)$$

$$g_k = S_{k-W+1}^k$$

$$S_j^k = \frac{k - j + 1}{2}(\rho(j, k) - \ln(\rho(j, k)) - 1)$$

$$t_0 = argmax(S_j^{t_a + \delta})$$

$$dove \quad \rho(j, k) = \frac{\theta_1(j, k)}{\theta_0} = \frac{(1/(k - j + 1)) \sum_{i=j}^{k} y_i^2}{(1/M) \sum_{i=1}^{M} y_i^2}$$

In un analisi posteriore nel 2017 [7] vengono utilizzati altri approcci oltre al threshold su un inviluppo lineare, come quello basato sull'applicazione di un pre-processamento chiamato TKEO, Sample Entropy e altri approcci statistici, come ad esempio, Generalized-Likelihood Ratio e Bayesian Change Point. In particolare, l'utilizzo del TKEO viene descritto in [13] [12] ed è una parte del pre-processamento di un segnale riassunto dalla

seguente formula:

$$\psi(x(j)) = x(i)^2 - x(i-1)x(i+1)$$

Questo pre-processamento è inserito in una sequenza di filtraggi: prima un passa-alto a 20 Hz, poi viene applicato il TKEO e infine un passa-alto a 50 Hz. L'applicazione di questa lavorazione rende più performante gli algoritmi soglia.
Altre applicazioni avanzate fanno ad esempio utilizzo di wavelet tranform[4] e deep learning[5].

Capitolo 3

Metodologie

In questa sezione vengono spiegate e dettagliate le varie fasi seguite nel progettare e implementare gli esperimenti volti a valutare l'applicabilità delle reti neurali nella predizione degli istanti di on/off. Per prima cosa è stato generato un dataset sintetico di 1040 segnali tramite la procedura dell'articolo [14] spiegata nel dettaglio nella prima sezione; poi, il dataset è stato suddiviso in un trainset di 500 segnali e un testset di 540 segnali. Sono stati valutati tre pre-processamenti differenti (Inviluppo Lineare, Root Mean Square Value e Scalogramma) come input della rete neurale utilizzata, cioè un multiperceptrone con singolo hidden layer. La rete ha il compito di classificare una finestra di 10 campioni, proveniente da uno specifico pre-processamento o dalla concatenazione degli output di più di uno, in due classi: rumore o attivazione.
In seguito per comprendere quale schema di pre-processamenti utilizzare e quali iperparametri fossero gli ottimali per migliorare la performance della rete è stata utilizzata una 10-fold cross-validation sul trainset dei 500 segnali. Infine, viene spiegato il processo di ricostruzione del segnale e il post-processamento. L'ultima sezione è dedicata ad una spiegazione dei metodi implementati per la valutazione delle performance del modello finale in fase di testing. Di seguito sono elencati i vari punti che verrano approfonditi:

1. Generazione dei segnali sintetici
2. Pre-processamenti
3. Creazione del dataset
4. Rete neurale, preparazione dei dati e cross-validation
5. Ricostruzione del segnale
6. Post-processamento
7. Valutazione dei risultati

3.1 Generazione dei segnali sintetici

L'utilizzo di un approccio tramite rete neurale necessita che la stessa sia addestrata su un dataset di segnali d'esempio di cui si conoscono gli istanti di on e off, cioè le labels. Questo non è possibile con segnali reali ed è quindi necessario svilupparne di sintetici. La generazione di segnali sintetici [1] [2] [3] [4] [5] o semi-sintetici [7] [12] [13] è una pratica standard in letteratura.

3.1.1 Schema generale

La generazione dei segnali è stata effettuata secondo lo schema prima illustrato da Stulen e De Luca [14] e poi successivamente adottato da Bonato [1].

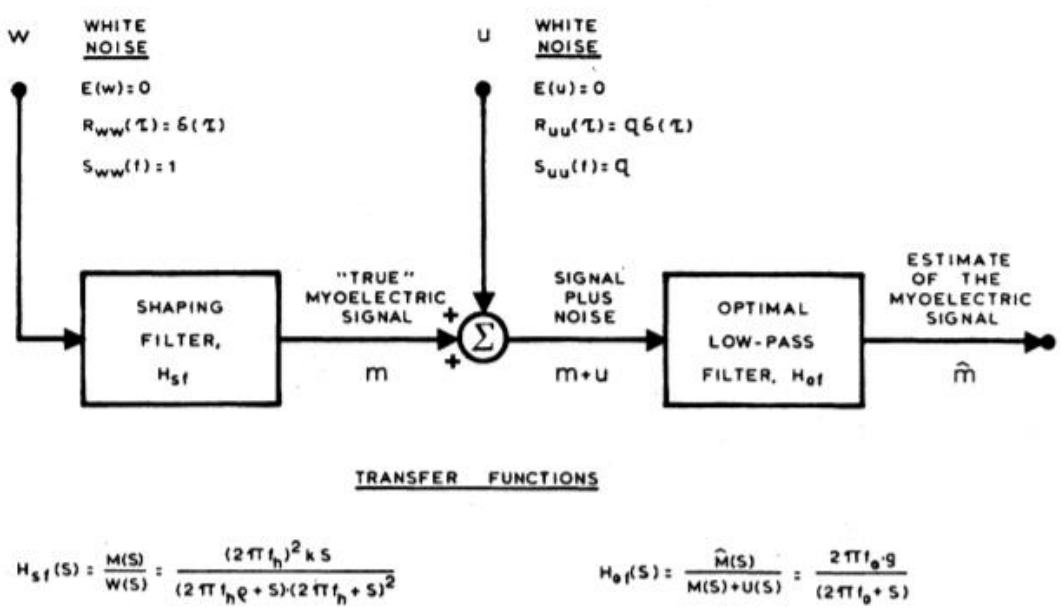


Figura 3.1: *Schema di generazione del segnale sintetico* [14]

Lo schema in figura 3.1 preso da [14] illustra l'intero procedimento che divideremo in:

1. Creazione dell'inviluppo gaussiano
   Una curva gaussiana della lunghezza del segnale (1s nel nostro caso) viene creata a partire da tre principali parametri: σ , cioè la sua larghezza, α, che definisce la finestra di attivazione e μ che è il punto medio. I valori di σ adottati sono gli stessi utilizzati da Bonato, cioè 50 ms, 100 ms e 150 ms e lo stesso vale per α con i valori 1, 1.5, 2, 2.4 ; μ invece è scelto in modo randomico tra 0.1 s e 0.9 s.

2. Modulazione del rumore bianco
   L'inviluppo così creato modula un segnale di rumore bianco della stessa dimensione del segnale che si vuole creare; quindi, istante per istante il rumore bianco viene moltiplicato per la campana creata donando al segnale rumoroso una caratteristica

forma d'onda che può essere paragonata a quella di un segnale sEMG. Dal punto di vista dell'ampiezza in questo modo simuliamo con buona accuratezza il segnale neuromuscolare, ma è necessario che anche il contenuto in frequenza sia simile.

3. Filtraggio passa banda del segnale 'vero'
   Questo filtraggio, progettato da Stulen e De Luca e descritto nel loro articolo[14], è un passa banda tra 80 Hz e 120 Hz con un roll-off più ripido per frequenze più alte e meno per le frequenze sotto gli 80 Hz. Per ottenere questo tipo di filtraggio viene utilizzata la seguente funzione di trasferimento:

$$H = \frac{(2\pi f_h)^2 ks}{(2\pi f_l + s) \times (2\pi f_h + s)^2}$$

   I parametri sono: k che è un fattore di scala, la frequenza di cut-off bassa f_l posta a 80 e f_h,la frequenza di cut off alta posta a 120. Le frequenze di cut-off sono prese dall'articolo di Bonato.
   L'implementazione di questo filtraggio permette di ottenere una simulazione alquanto accurata anche in frequenza.

4. Aggiunta di rumore additivo
   Una seconda sequenza di rumore bianco incorrelato, diverso dal primo, viene aggiunta al segnale con l'adatta attenuazione o amplificazione per ottenere il segnale con Signal To Noise (SNR) voluto.

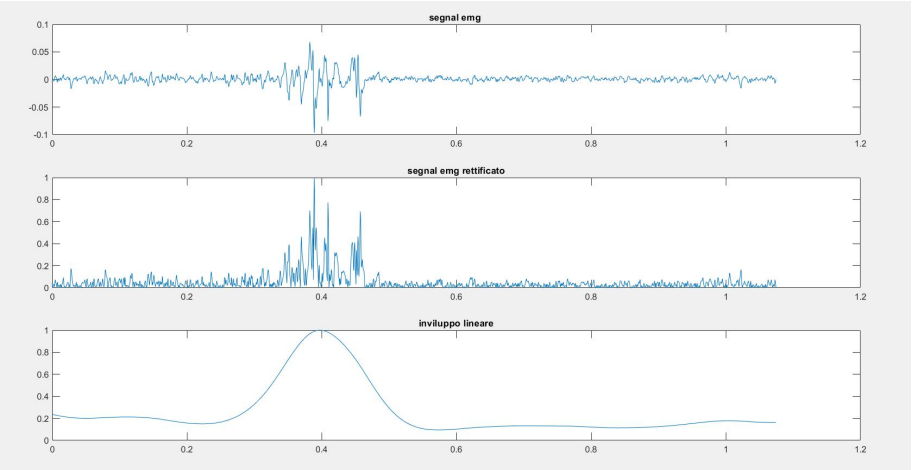
5. Filtraggio passa basso
   Il rumore aggiunto nel punto 4) ha una banda che è realmente molto più larga di quella che ci si aspetterebbe e vanno quindi eliminate le componenti con frequenza elevata con un filtraggio passa-basso. In particolare, nell'articolo[14] non viene specificata una frequenza di cut-off, che è stata posta a 500 Hz per la generazione dei segnali utilizzati in questa tesi. Il filtro è stato implementato come un butterworth di secondo ordine.

Per quanto riguarda la definizione di SNR utilizzata, è la stessa riportata da Bonato:

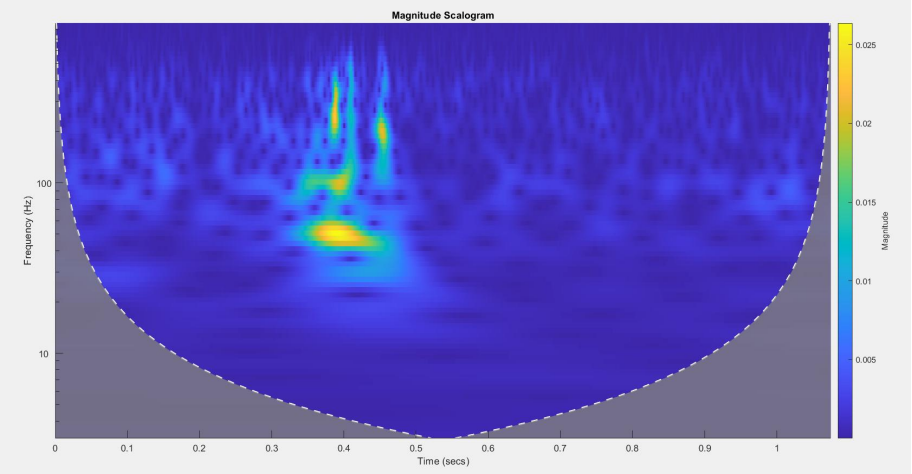
$$SNR = 10 \times \log \frac{\text{varianza segnale}}{\text{varianza rumore}}$$

I segnali creati hanno tutti frequenza di campionamento di 2000 Hz e sono tutti lunghi 1s (2000 campioni) per andare a simulare l'attivazione muscolare durante un passo. Nelle figure 3.2 e 3.3 sono riportati rispettivamente un esempio di segnale reale di un gastrocnemio e del suo scalogramma e un segnale sintetico di un'attivazione muscolare centrata in 0.4 s che va a simulare il precedente segnale reale. Notiamo che il picco di banda è presente tra i 100 e i 200 Hz per entrambi i segnali e che la maggior parte della potenza del segnale è tra i 10Hz e i 500Hz.

Sono comunque visibili le limitazioni nella simulazione, come fanno notare gli stessi autori dell'articolo [14]. In particolare, per alte frequenze questa simulazione non presenta degli avvallamenti ('dips'), mentre per basse frequenze non presenta dei picchi dovuti al 'firing' delle unità motorie e la cui presenza era stata notata dallo stesso De Luca in un altro articolo. Questa simulazione è comunque sufficiente per il problema di attivazione muscolare ed è utilizzata in vari articoli [1][3] [4].Analizzando invece l'ampiezza, la forma d'onda data dall'inviluppo lineare nel terzo subplot è confrontabile.

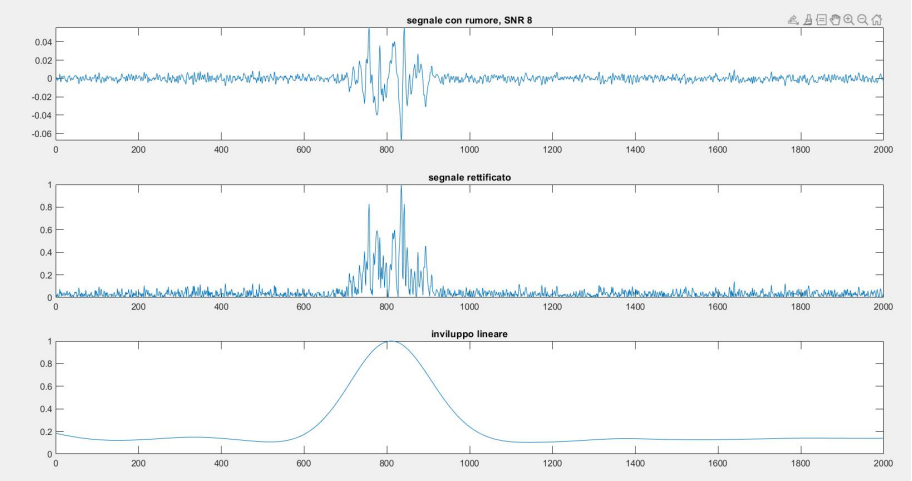


(a) *Segnale reale*

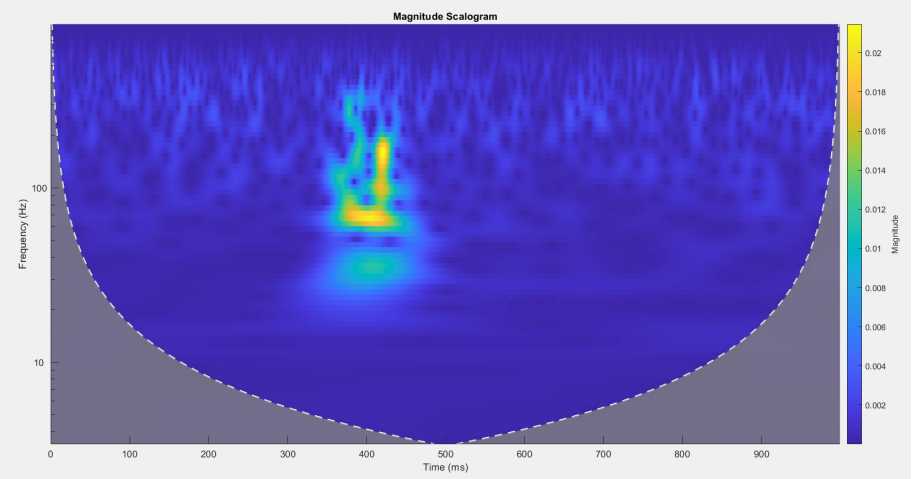


(b) *Scalogramma segnale reale*

Figura 3.2: *Esempio segnale reale del gastrocnemio di un soggetto maschile, preso da un dataset condiviso dal prof. Di Nardo*



(a) *Segnale simulato*



(b) *Scalogramma del segnale simulato*

Figura 3.3: *Esempio segnale simulato*

3.2 Preprocessamento del segnale

Sono stati utilizzati tre pre-processamenti per i segnali:

1. Inviluppo Lineare (LIN)
   L'inviluppo lineare è un processamento tipico del segnale sEMG e si compone di rettificazione del segnale e filtraggio tramite un butterworth passa basso con cut off a 5 Hz e ordine 2 [Fig.3.5].

2. Root Mean Square Value con finestra di 30ms (RMSV30)
   Al segnale viene aggiunto un padding di 15 zeri all'inizio e alla fine e per ogni valore viene calcolato il quadrato; a questo punto il segnale viene diviso in finestre scorrevoli da 30ms e viene calcolata la media dei valori al suo interno,la quale diventa poi argomento di una radice. Questa è un'implementazione della seguente formula:

   $$RMSV = \sqrt{\frac{1}{T}\int_0^T x(t)^2 dt} [Fig.3.6]$$

3. Scalogramma (SCAL)
   Questo processamento permette di avere una rappresentazione del segnale in tempo-frequenza con un'immagine di dimensioni 81 x 2000. Un ulteriore processamento sullo scalogramma è una max-pool che consiste nel prendere i valori massimi per ogni colonna con un passo di tre riducendo la dimensione a 27 x 2000. Questo ha permesso di ridurre il carico sia in memoria sia computazionale ottenendo comunque buoni risultati. Un risultato simile potrebbe essere ottenuto abbassando il numero di 'VoicesPerOctave' nella funzione CWT di Matlab [Fig. 3.7].

I segnali così creati sono normalizzati con una min-max feature scaling; lo scalogramma è normalizzato rispetto a massimo e minimo dell'intera matrice.

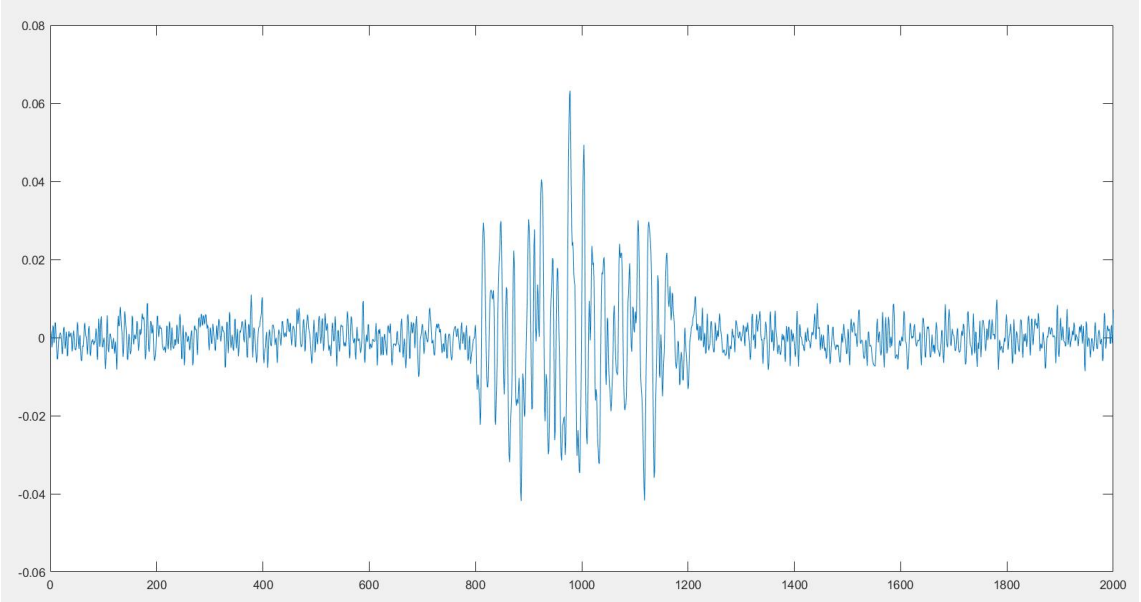


Figura 3.4: *Segnale simulato, SNR 8dB, σ 0.1, $\alpha = 1$, $\mu = 0.5$; sull'asse delle ascisse ci sono i campioni mentre sulle ordinate l'ampiezza*

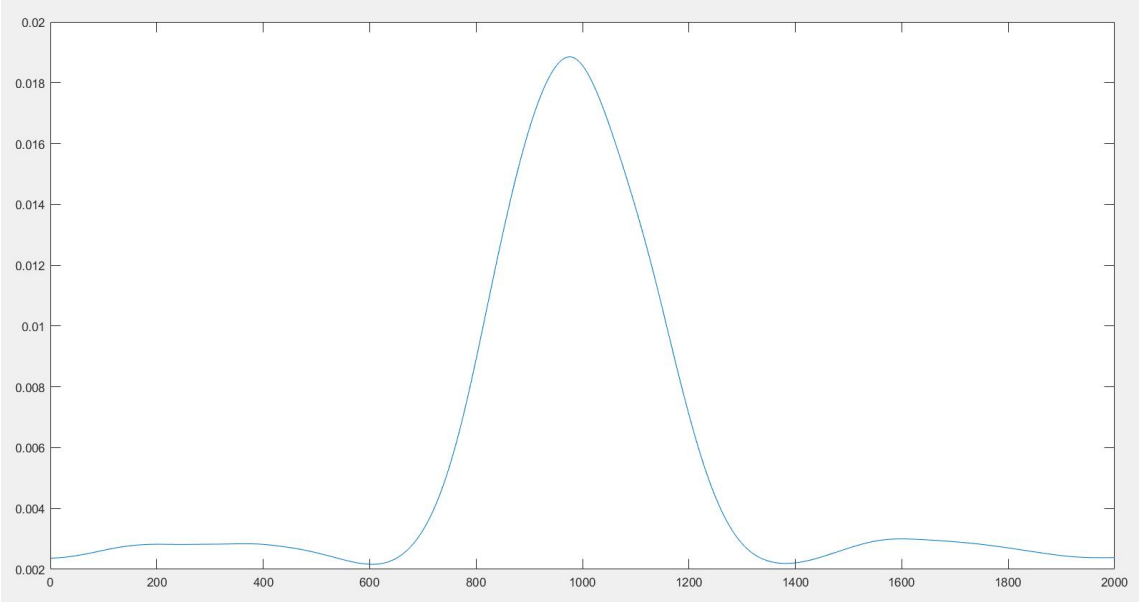


Figura 3.5: *Inviluppo lineare*

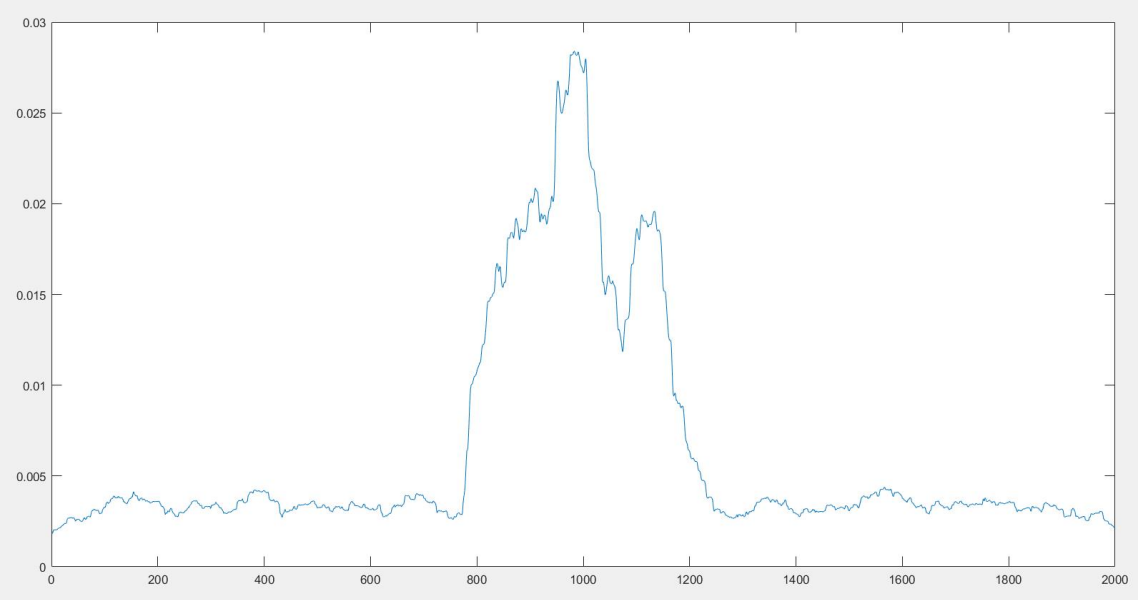


Figura 3.6: *RMSV30*

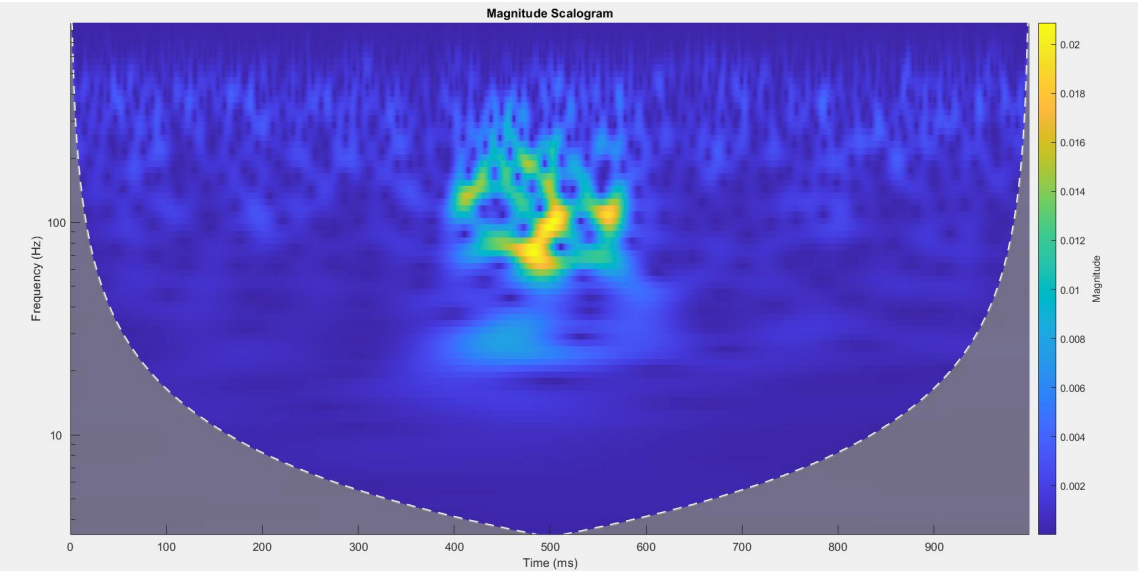


Figura 3.7: *Scalogramma*

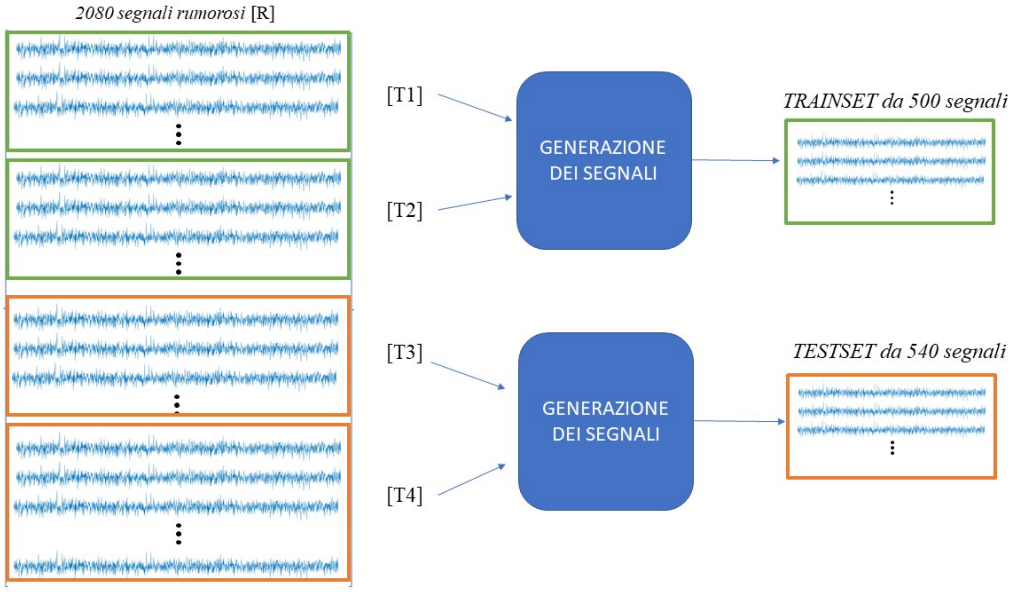


Figura 3.8: *Illustrazione schematica della creazione di TRAIN E TEST SET*

3.3 Creazione del dataset

Lo schema in figura 3.8 riassume brevemente il processo di creazione dei datasets che viene successivamente spiegato nel dettaglio. È stato creato un dataset di 1040 segnali da cui sono stati ricavati train e test set. Inizialmente, una matrice di rumore [R] di dimensioni 2080 x 2000 è stata inizializzata tramite la funzione wgn del toolbox Telecomunication di Matlab, che crea rumore bianco. Questa procedura permette quindi di avere 2080 segnali rumorosi tra loro incorrelati. Dalla matrice [R] vengono formate 4 matrici: [T1] dalle prime 500 righe, [T2] dalle successive 500, [T3] con i segnali dal 1001 al 1540 e [T4] con i rimanenti.

Nella procedura di formazione del segnale sEMG simulato necessitiamo di due segnali rumorosi, uno che viene lavorato tramite modulazione con inviluppo e filtraggio e l'altro che viene semplicemente aggiunto per ottenere il giusto SNR. Perciò, dalla matrice [T1] vengono ottenuti 500 segnali sEMG senza rumore secondo i punti 1), 2) e 3) della precedente sezione. I parametri per σ dell'inviluppo sono stati scelti casualmente tra i valori 0.05,0.1 e 0.15, mentre per α tra 1, 1.5, 2 e 2.4. Questi valori sono forniti in [1]. A questo punto la matrice [T2] è utilizzata per il punto 4), cioè l'aggiunta del rumore additivo. Ad esempio, al segnale generato tramite la prima riga della matrice [T1] verrà aggiunto il segnale rumoroso della prima riga della matrice [T2], amplificato o attenuato in modo adeguato per ottenere l'SNR previsto. Il segnale così formato subisce il filtraggio del punto 5). Lo stesso schema è ripetuto per tutte le 500 righe per ottenere il train set di 500 segnali. L'SNR di questi segnali varia da 1 a 30dB con passo 1.

La stessa procedura è stata seguita per i segnali nelle matrici [T3] e [T4]; l'unica differenza è che sono stati creati 5 segnali sEMG simulati per ogni combinazione tra i tre valori di σ e i quattro valori di α sopra riportati e 9 valori di SNR presi tra 3 e 30 (3,6,10,13,16,20,23,26,30).

3.4 Rete Neurale

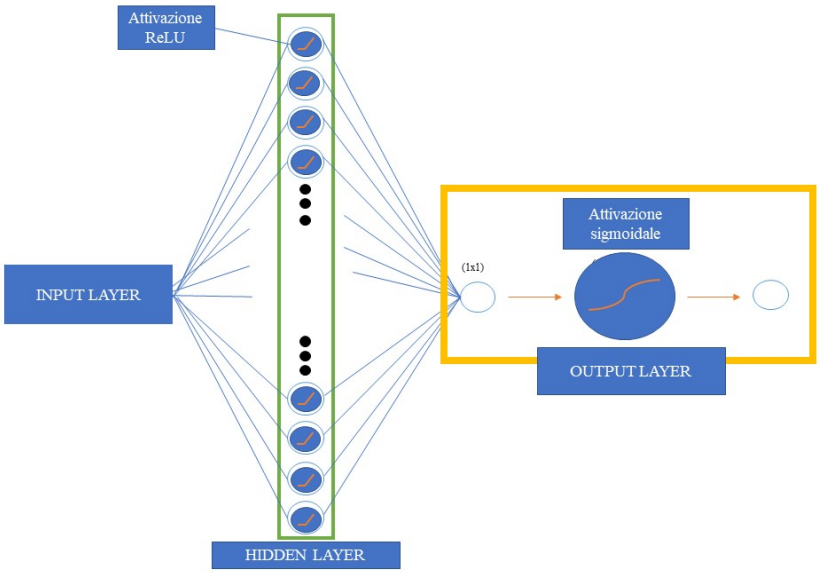


Figura 3.9: *Struttura MLP1;l'input layer è un vettore di valori; nell'immagine all'interno dell'hidden layer è stata inclusa l'attivazione ReLU, con un piccolo simbolo per ogni valore dell'hidden layer; l'ouptut layer per il singolo input è un valore che viene poi passato come argomento ad una sigmoide*;

La rete neurale che è stata utilizzata è un multiperceptrone (MLP).L'architettura è composta da un hidden layer con attivazione ReLU, che fornisce non linearità alla rete, e un'attivazione sigmoidale sull'output.

L'input è un vettore che può cambiare di dimensioni a seconda del preprocessamento utilizzato. L'hidden layer è composto da 32 neuroni e l'output è un solo valore, che diventando argomento della sigmoide sarà sempre tra 0 e 1. Questo numero è quindi arrotondato a 0 se minore di 0.5 rappresentando la sola presenza di rumore, mentre ad 1 se maggiore o uguale a 0.5 rappresentando la presenza di segnale muscolare.

Il problema è quindi affrontato come una classificazione binaria degli input (finestre di 10 campioni) nelle label 0, rumore, ed 1 , attivazione muscolare. La funzione costo utilizzata per ottimizzare i pesi della rete è la BinaryCrossEntropy.

3.4.1 Preparazione dei dati

La rete necessita in entrata di un vettore uni-dimensionale. Quindi, è fondamentale una preparazione dei dati in ingresso alla rete che consiste in:

1. aggiunta di un padding iniziale di 5 zero ed uno finale di 4 zero al segnale; nel caso dello scalogramma si aggiungono matrici di zeri di dimensione di 10x27 all'inizio e alla fine

2. segmentazione in finestre da 10 campioni con scorrimento di un solo campione; nel caso dello scalogramma viene segmentata la matrice in vettori bi-dimensionali di dimensioni 10x27 a cui poi è applicata una MaxPool2d, funzione del modulo Pythorch, che va a ridurre le dimensioni a 5x27. Questi segmenti vengono poi 'appiattiti' per formare un vettore di lunghezza 1x135.

3. Poiché il segnale sEMG è sintetico, il segnale di attivazione è conosciuto e per ogni campione è possibile associare una label 0 o 1, rispettivamente rumore o attivazione. Tuttavia, all'input formato da 10 campioni è associata una label che è la media tra le label di quei 10 campioni, arrotondata al valore più vicino. Se è presente una transizione all'interno della finestra considerata, l'input sarà considerato di attivazione solo se sono presenti almeno 5 label uguali ad 1 (media maggiore o uguale ad 0.5 e quindi approssimata ad 1).

Di conseguenza, per ogni segnale creato si avrà un numero di input pari ai campioni, cioè 2000.

Figura 3.10: *Illustrazione di una cross-validation 10 fold*

3.4.2 Cross Validation e scelta dell'input

Una 10-fold-cross-validation è stata utilizzata sul trainset per valutare gli iperparametri ottimali e le migliori configurazioni di pre-processamento dell'input (che può essere considerato anch'esso un iperparametro, in quanto non è 'appreso' dalla rete, ma scelto a seconda dei risultati). La figura 3.10 illustra il meccanismo di cross-validation, la quale si articola in tre fasi:

1. Il dataset viene inizialmente mescolato. Il mescolamento avviene prima della segmentazione dei segnali in modo tale che a seguito della divisione ci siano informazioni provenienti da un segnale solo sia nel training sia nel validation set. Questo perché se il mescolamento fosse effettuato dopo la segmentazione si avrebbero informazioni riguardanti la transizione di un segnale sia nel training set sia nel validation set, rendendo quindi gli esempi in validation non totalmente indipendenti da quelli in train.

2. Il dataset viene diviso in 10 gruppi e viene eseguita un'iterazione in cui ogni gruppo viene utilizzato una volta come validation set e il rimanente dei gruppi come training set. In questo caso gli input in train e validation sono ora nuovamente mescolati.

3. Si effettuano quindi 10 addestramenti e le metriche finali saranno le medie su questi 10 fold.

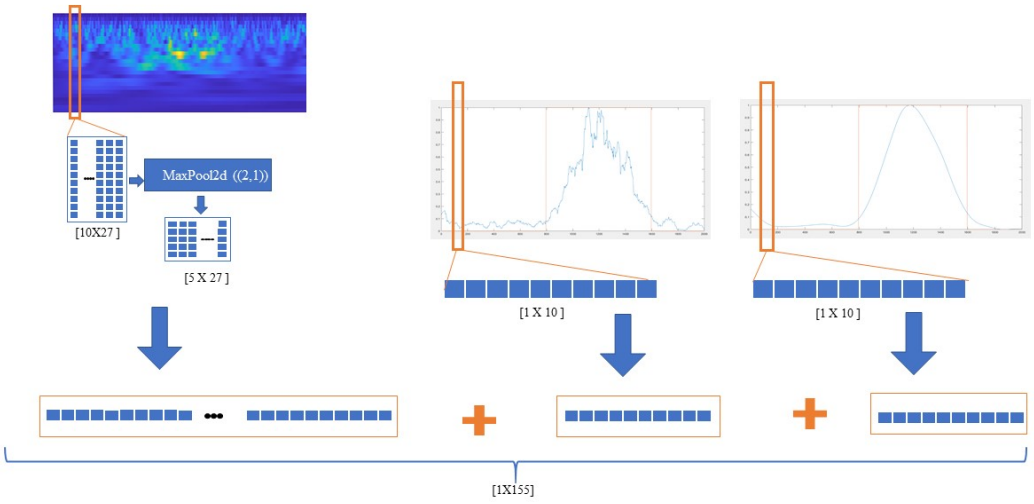


Figura 3.11: *Descrizione della concatenazione degli input*

Per quanto riguarda gli inputs, a seguito di padding e segmentazioni otteniamo 2000 finestre per ogni segnale; quindi, sono 1000000 in tutto, di cui il 90% è utilizzato durante l'addestramento. I tre pre-processamenti descritti precedentemente nella sezione 3.3 sono

utilizzati sia singolarmente sia per creare inputs dati dalla concatenazione degli output dei singoli pre-processamenti, come può essere visto nella figura 3.11, la quale descrive un processo di concatenazione degli output di tutti e tre i pre-processamenti (analoga è la condizione di concatenazione di meno output). Il confronto tra i risultati della cross-validation è utilizzato per valutare la migliore configurazione di pre-processamenti da implementare per la costruzione del modello finale.

Gli iperparametri per l'input con l'inviluppo lineare sono learning rate 0.001, numero di epoche 20 e dimensione della batch 50. Per gli altri input invece, gli iperparametri utilizzati sono learning rate = 0.005, numero di epoche = 15 e dimensione della batch = 50; l'optimizer utilizzato è stato sempre lo stochastic gradient descent con un weigth decay di 0.01.

Per la fase di model building e in seguito di testing è stato scelto l'input dato dalla concatenazione degli output dei tre pre-processamenti [Fig 3.11, il quale presenta in generale le migliori performance. Quindi, un nuovo modello viene addestrato sulla totalità dei 500 segnali di trainset con la configurazione di pre-processamenti scelta e con gli iperparametri definiti attraverso la cross-validation. Questo modello finale verrà poi utilizzato nella fase di testing.

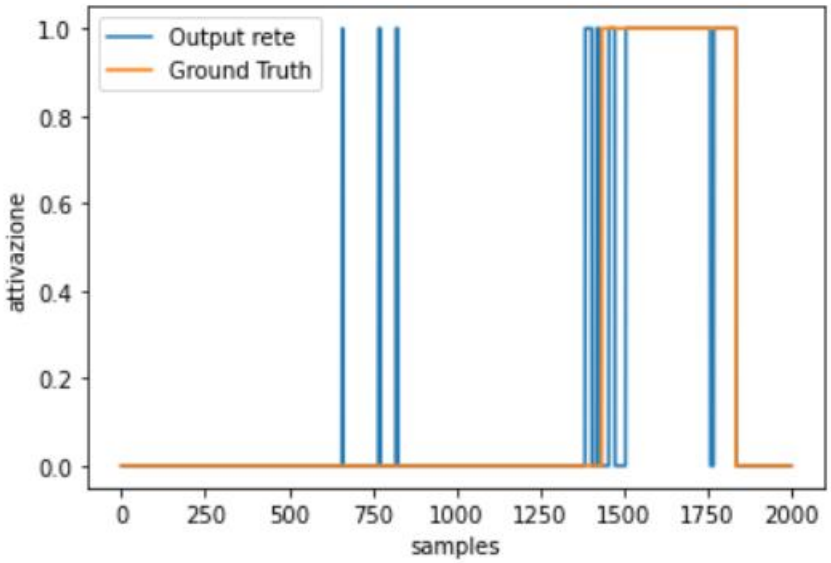


Figura 3.12: *Esempio dell'output della rete per un segnale con SNR di 3dB prima del post processamento*

3.5 Ricostruzione del segnale

Una volta ottenuto il modello, la rete neurale può classificare un dato input di 10 campioni, pre-processati come in figura 3.11 nei valori 0 o 1. Tuttavia, per ricostruire il segnale on-off, dobbiamo implementare un processo di padding e segmentazione del segnale identico a quello utilizzato in fase di training. Il segnale da testare viene quindi diviso in 2000 finestre da 10 campioni che sono ordinate temporalmente. Ognuna di queste viene classificata tramite la rete e viene mantenuta la sequenza temporale degli input. L'output finale è una ricostruzione del segnale on-off come nella figura 3.12.

3.6 Post Processamento

L'output così creato subisce successivamente un post-processamento che ha lo scopo di eliminare le false transizioni, dove con falsa transizione intendiamo una serie successiva di campioni con lo stesso stato 1 o 0 che duri meno di 30 ms. Per far ciò vengono individuate le transizioni positive di durata inferiore 30 ms e vengono eliminate. Successivamente le transizioni negative di 60 campioni o meno vengono portate ad 1. La soglia di 30 ms è giustificata, poiché è accettato che queste transizioni non influenzino in modo rilevante né la cinematica né la cinetica di un ciclo di passo [1].

3.7 Valutazione dei risultati

La rete neurale è stata testata su 540 segnali nella fase di testing tramite latency sull'istante di onset e offset e tramite il numero di transizioni false positive. Per prima cosa l'output

della rete a seguito del post-processamento viene analizzato cronologicamente al fine di trovare gli istanti di onset/offset predetti e classificare i falsi positivi. Un falso positivo è individuato come quella transizione t_p che non rispetta la seguente condizione:

$$|t_p - t_r| <= E$$

dove t_r è l'istante di transizione reale ed E è una tolleranza posta a 100 ms, 10% del gait e la stessa dell'articolo [4]. Al contrario un onset o un offset è true positive se rientra nella regione delimitata da questa tolleranza, mentre è presente una transizione falsa negativa se non c'è alcuna transizione all'interno di questa regione. Le figure 3.13, 3.14 e 3.15 nella pagina successiva riportano degli esempi di individuazione di veri positivi e falsi positivi. A questo punto, viene determinata la latency, definita come l'errore assoluto sull'istante di transizione, calcolato come la differenza in valore assoluto tra t_p, l'istante predetto, e t_r, l'istante reale, il tutto diviso per due, in quanto il risultato dal numero dei campioni viene portato in ms.

$$LATENCY = |t_p - t_r|/2$$

Per l'individuazione dell'istante predetto t_p si è utilizzata la seguente procedura:

- un istante predetto è designato come tale se all'interno della regione di tolleranza; se è presente un solo true positive il calcolo è immediato, mentre se sono presenti più transizioni eleggibili come true positive (come nel caso dell'output senza post-processamento) viene scelto come t_p la media delle transizioni trovate

- se è presente un false negative (FN) e quindi non è possibile individuare alcuna transizione true positive (TP), viene utilizzata la transizione false positive (FP) più vicina all'istante reale. Non si è mai presentata la situazione in cui l'output della rete fosse tutto a zero, quindi queste due regole permettono di regolare tutte le possibili situazioni.

Infine, sono state calcolate anche le metriche di predizione sugli istanti predetti, cioè precision, recall e f1score.

La valutazione dei risultati è stata effettuata sull'output della rete con e senza il post-processamento.

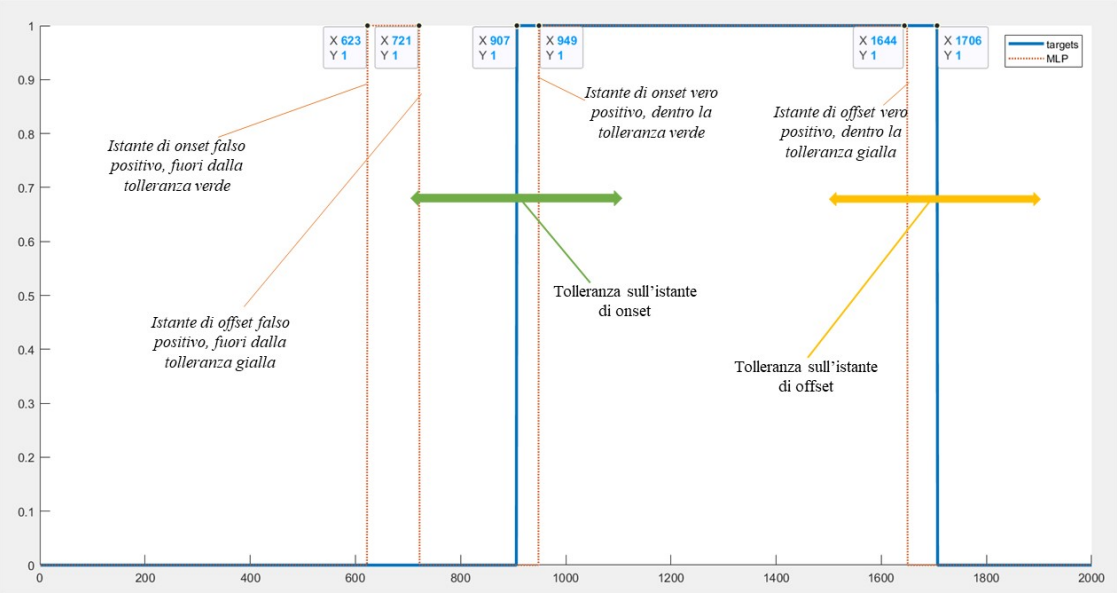


Figura 3.13: *Esempio di una predizione della rete a seguito del post-processamento; le transizioni true positive sono quelle all'interno di una tolleranza, rappresentata con la freccia verde per l'onset e quella gialla per l'offset, e le false positive sono presenti nel caso di transizioni fuori dalla tolleranza; la latency è calcolata tramite la distanza in ms tra quelle transizioni rilevate come vere positive e le relative transizioni reali*

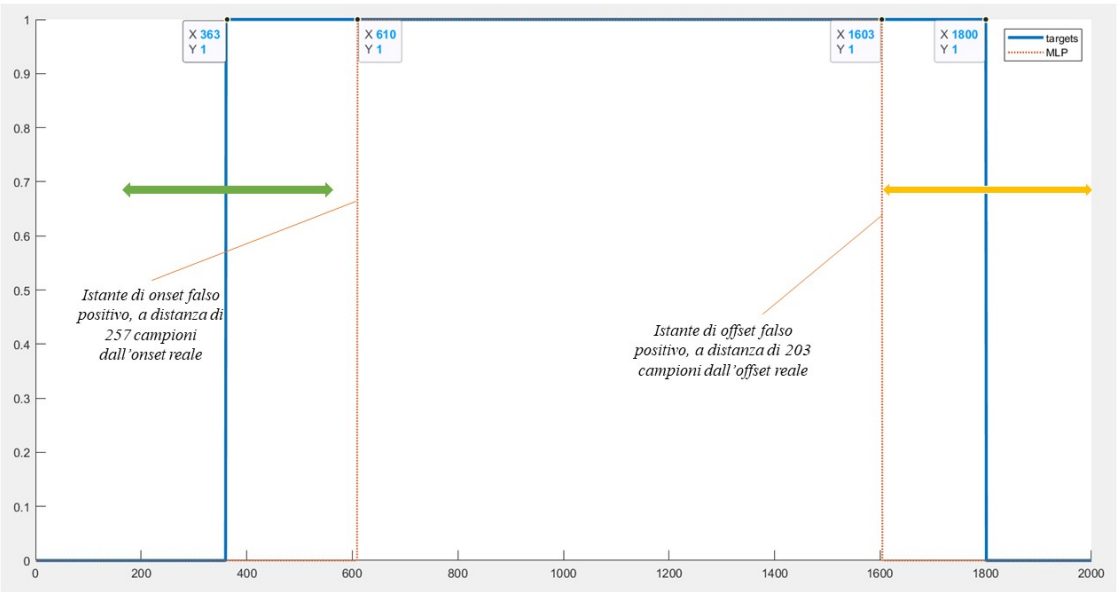


Figura 3.14: *Esempio di una predizione della rete a seguito del post-processamento; non sono presenti transizioni vere positive, ma solo false positive sia per onset sia per offset; in questo caso, non essendo presente alcun vero positivo la latency è calcolata tramite la distanza in ms tra la transizione false positiva più vicina all'istante reale e il relativo istante reale*

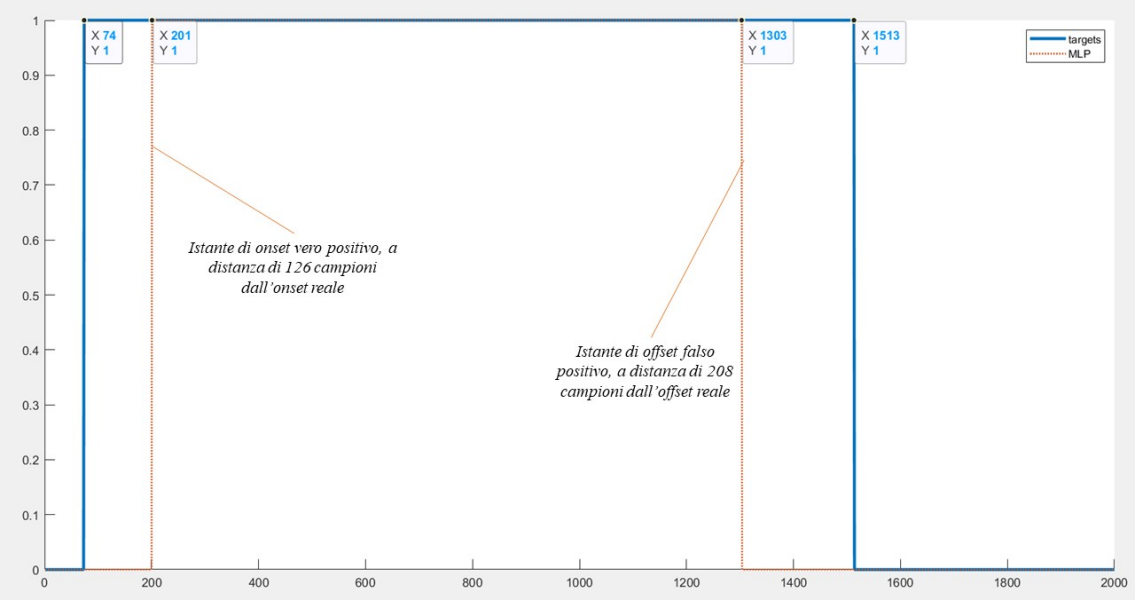


Figura 3.15: *Esempio di una predizione della rete; è presente una transizione vera positiva per l'onset ed una falsa positiva per l'offset, quindi, in questo caso, la latency per l'offset è calcolata tramite la distanza in ms tra la transizione false positiva più vicina (unica in questo esempio) all'istante reale e il relativo istante reale; la latency sull'onset è calcolata come distanza assoluta tra l'onset vero positivo e l'onset reale*

3.8 Linguaggi e piattaforme utilizzate

L'ambiente MATLAB è stato utilizzato per ciò che concerne la lavorazione del segnale, dalla sua generazione e pre-processamento alla creazione del dataset; sempre su questa piattaforma è stato utilizzato l'algoritmo di Bonato sul testset per effettuare un confronto. Invece, Python con il modulo Pythorch, è stato utilizzato per l'implementazione della rete neurale con cross-validation sul trainset e valutazione finale sul testset.
Inoltre, Microsoft Excel è stato utilizzato per la raccolta dei data e i grafici.

Capitolo 4

Risultati

Questa sezione ha lo scopo di mostrare i risultati ottenuti che verranno di seguito commentati nella successiva sezione dedicata alla discussione.

4.1 Risultati delle cross-validation

Di seguito sono mostrate le metriche di diverse cross-validation basate sui diversi input valutati, dove in grassetto sono evidenziate le migliori; le metriche valutate sono l'accuratezza media e la loss, calcolate sia sul train sia sulla validation dei vari fold. Poi, per le label 0, associata alla presenza di solo rumore, ed 1, associata alla presenza di attivazione muscolare, sono riportate precision, recall e f1score con le relative std. Queste sono le metriche della task di classificazione sul singolo input, cioè la finestra di 10 campioni.

	loss train	acc train	loss validation	acc validation
LIN	0.1718 +- 0.0018	93.10 +- 0.08	0.1726 +- 0.0123	92.97 +- 0.61
RMSV30	0.1475 +- 0.0013	94.55+0.09	0.1404 +- 0.0122	94.92 +- 0.94
SCAL	0.0975 +- 0.0030	96.79 +- 0.11	0.0988 +- 0.0142	96.82 +- 0.52
SCAL+LIN	0.0927 +- 0.0016	96.96 +- 0.06	0.0957 +- 0.0090	96.91 +- 0.34
SCAL+RMSV30	0.899 +- 0.0015	97.10 +- 0.0042	0.0926 +- 0.0107	97.11 +- 0.06
SCAL+LIN+RMSV30	**0.0874 +- 0.0015**	**97.18 +- 0.007**	**0.0888 +- 0.0097**	**97.17 +-0.34**

Tabella 4.1: *La tabella mostra le varie metriche ottenute con una cross-validation su un dataset di 500 segnali generati sinteticamente. I vari acronimi della prima colonna stanno ad indicare le varie configurazioni di pre-processamento considerate: LIN sta per Linear Envelope, RMSV30 sta per Root Mean Square Value usando una finestra di 30ms, SCAL sta per scalogramma; il simbolo + sta ad indicare una concatenazione degli output ottenuti dai singoli pre-processamenti sullo stesso segnale. Notiamo come una concatenazione degli output dei tre pre-processamenti LIN, RMSV30 e SCAL produca le performance migliori.*

	precision	recall	f1score
LIN	93.95 +- 0.71	95.99 +-1.14	94.95 +- 0.49
RMSV30	95.02 +- 1.07	97.75 +- 1.28	96.36+-0.74
SCAL	96.84 +- 0.85	98.60 +- 0.44	97.71 +- 0.43
SCAL+LIN	96.80 +- 0.6	98.79 +- 0.39	97.78 +- 0.22
SCAL+RMSV30	97.08 +- 0.75	**98.88 +- 0.25**	97.94 +- 0.36
SCAL+LIN+RMSV30	**97.20 +- 0.34**	98.73 +- 0.35	**97.96+-0.29**

Tabella 4.2: *Metriche per la LABEL 0*

	precision	recall	f1score
LIN	90.65 +- 2.49	86.22 +- 1.48	88.34 +- 1.00
RMSV30	94.73 +- 2.72	88.66 +- 1.94	91.55 +- 1.29
SCAL	96.72+-1.12	92.90 +- 1.37	94.76 +- 0.70
SCAL+LIN	97.13 +- 1.04	92.76+-1.11	94.88 +- 0.46
SCAL+RMSV30	**97.25 +- 0.63**	93.46 +- 1.29	95.31 +-0.54
SCAL+LIN+RMSV30	97,11 +- 0.58	**93.64 +-0.77**	**95.34+-0.45**

Tabella 4.3: *Metriche per la LABEL 1*

4.2 Risultati del modello finale

Il modello finale è stato testato sul testset di 540 segnali. Di seguito sono riportate le metriche di predizione sulla singola finestra di 10 campioni (precision, recall e f1score per entrambe le label), calcolate nelle condizioni di pre e post-processamento[Tab.4.4,4.5].

	prec (0)	recall (0)	f1score(0)
PRE	96.49 +- 6.71	98.92 +-2.08	97.55+-4.03
POST	96.30 +- 7.15	99.15 +- 1.16	97.58 +- 4.11

Tabella 4.4: *Metriche pre e post-processamento per la label 0 calcolate su una singola finestra; il modello utilizzato è quello che ha avuto le migliori performance in cross-validation.*

	prec (1)	recall (1)	f1score(1)
PRE	97.12 +- 4.26	95.13+-6.90	95.89+-3.97
POST	97.73 +- 3.21	94.62 +- 7.85	95.89 +- 4.12

Tabella 4.5: *Metriche pre e post-processamento per la label 1 calcolate su una singola finestra; il modello utilizzato è quello che ha avuto le migliori performance in cross-validation.*

Invece in [Fig. 4.1] sono riportati i valori di F1-score e relativa deviazione standard sia per la predizione degli istanti di OFFSET che di ONSET. Le figure 4.2 e 4.3 riportano le stesse metriche sui due istanti in funzione di alpha e SNR rispettivamente.

	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
pre	0,779	0,194	1,000	0,000	0,859	0,139
post	0,984	0,038	0,989	0,024	0,986	0,032

	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
pre	0,783	0,223	0,999	0,002	0,856	0,167
post	0,979	0,049	0,983	0,035	0,981	0,043

Figura 4.1: *Risultati delle metriche sull'output senza(pre) e con(post) il post-processamento sui due istanti di transizione calcolate attraverso la tolleranza definita; le performance riportate sono le medie su tutti i 540 segnali di test e sono evidenziate in arancione i cambiamenti negativi e in verde quelle positivi a seguito del post-processamento.*

pre	sigma	alpha	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
1	0,05	1,0	0,926	0,169	1,000	0,000	0,953	0,111
2	0,05	1,5	0,805	0,178	1,000	0,000	0,881	0,128
3	0,05	2,0	0,835	0,100	1,000	0,000	0,907	0,061
4	0,05	2,4	0,712	0,225	1,000	0,000	0,813	0,166
5	0,10	1,0	0,835	0,221	1,000	0,000	0,894	0,149
6	0,10	1,5	0,836	0,131	1,000	0,000	0,906	0,083
7	0,10	2,0	0,765	0,244	1,000	0,000	0,846	0,166
8	0,10	2,4	0,772	0,163	1,000	0,000	0,863	0,105
9	0,15	1,0	0,729	0,223	1,000	0,000	0,824	0,170
10	0,15	1,5	0,714	0,194	1,000	0,000	0,819	0,141
11	0,15	2,0	0,806	0,270	1,000	0,000	0,865	0,206
12	0,15	2,4	0,617	0,213	1,000	0,000	0,743	0,175

pre	sigma	alpha	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
1	0,05	1,0	0,931	0,208	1,000	0,000	0,949	0,152
2	0,05	1,5	0,849	0,139	1,000	0,000	0,913	0,085
3	0,05	2,0	0,760	0,230	1,000	0,000	0,843	0,180
4	0,05	2,4	0,830	0,210	1,000	0,000	0,892	0,154
5	0,10	1,0	0,850	0,245	1,000	0,000	0,897	0,191
6	0,10	1,5	0,793	0,238	1,000	0,000	0,865	0,164
7	0,10	2,0	0,788	0,115	1,000	0,000	0,877	0,071
8	0,10	2,4	0,723	0,267	1,000	0,000	0,813	0,193
9	0,15	1,0	0,668	0,294	1,000	0,000	0,763	0,242
10	0,15	1,5	0,797	0,229	1,000	0,000	0,868	0,176
11	0,15	2,0	0,758	0,257	1,000	0,000	0,838	0,193
12	0,15	2,4	0,646	0,250	0,993	0,020	0,757	0,202

(a) *Metriche in funzione di alpha e sigma senza post-processamento*

post	sigma	alpha	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
1	0,05	1,0	1,000	0,000	1,000	0,000	1,000	0,000
2	0,05	1,5	1,000	0,000	1,000	0,000	1,000	0,000
3	0,05	2,0	0,981	0,056	1,000	0,000	0,990	0,030
4	0,05	2,4	1,000	0,000	1,000	0,000	1,000	0,000
5	0,1	1,0	1,000	0,000	1,000	0,000	1,000	0,000
6	0,1	1,5	1,000	0,000	1,000	0,000	1,000	0,000
7	0,1	2,0	0,981	0,056	1,000	0,000	0,990	0,030
8	0,1	2,4	1,000	0,000	1,000	0,000	1,000	0,000
9	0,15	1,0	0,981	0,056	1,000	0,000	0,990	0,030
10	0,15	1,5	1,000	0,000	1,000	0,000	1,000	0,000
11	0,15	2,0	0,978	0,067	0,978	0,067	0,978	0,067
12	0,15	2,4	0,889	0,226	0,889	0,226	0,889	0,226

post	sigma	alpha	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
1	0,05	1,0	1,000	0,000	1,000	0,000	1,000	0,000
2	0,05	1,5	1,000	0,000	1,000	0,000	1,000	0,000
3	0,05	2,0	0,981	0,056	1,000	0,000	0,990	0,030
4	0,05	2,4	1,000	0,000	1,000	0,000	1,000	0,000
5	0,10	1,0	1,000	0,000	1,000	0,000	1,000	0,000
6	0,10	1,5	1,000	0,000	1,000	0,000	1,000	0,000
7	0,10	2,0	0,981	0,056	1,000	0,000	0,990	0,030
8	0,10	2,4	1,000	0,000	1,000	0,000	1,000	0,000
9	0,15	1,0	0,981	0,056	1,000	0,000	0,990	0,030
10	0,15	1,5	1,000	0,000	1,000	0,000	1,000	0,000
11	0,15	2,0	0,933	0,200	0,933	0,200	0,933	0,200
12	0,15	2,4	0,867	0,224	0,867	0,224	0,867	0,224

(b) *Metriche in funzione di alpha e sigma con post-processamento*

Figura 4.2: *Risultati delle metriche sull'output prima(pre) e dopo(post) il post-processamento utilizzando la tolleranza definita; le performance riportate sono le medie su tutti i 540 segnali in funzione degli alpha e sigma; in arancione è stato evidenziato il risultato peggiore; 1 corrisponde alla percentuale 100%.*

pre	SNR	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
0	3	0,427	0,119	1,000	0,000	0,590	0,115
1	6	0,676	0,188	1,000	0,000	0,792	0,143
2	10	0,755	0,181	1,000	0,000	0,849	0,125
3	13	0,797	0,182	1,000	0,000	0,876	0,123
4	16	0,885	0,116	1,000	0,000	0,935	0,067
5	20	0,826	0,147	1,000	0,000	0,898	0,089
6	23	0,860	0,181	1,000	0,000	0,914	0,119
7	26	0,874	0,117	1,000	0,000	0,929	0,070
8	30	0,913	0,099	1,000	0,000	0,952	0,056

pre	SNR	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
0	3	0,377	0,133	0,995	0,017	0,535	0,132
1	6	0,563	0,233	1,000	0,000	0,694	0,196
2	10	0,783	0,201	1,000	0,000	0,864	0,137
3	13	0,853	0,130	1,000	0,000	0,916	0,077
4	16	0,821	0,137	1,000	0,000	0,896	0,085
5	20	0,901	0,127	1,000	0,000	0,943	0,076
6	23	0,935	0,081	1,000	0,000	0,964	0,044
7	26	0,866	0,124	1,000	0,000	0,924	0,075
8	30	0,946	0,107	1,000	0,000	0,969	0,062

(a) *Metriche in funzione dell'SNR senza post-processamento*

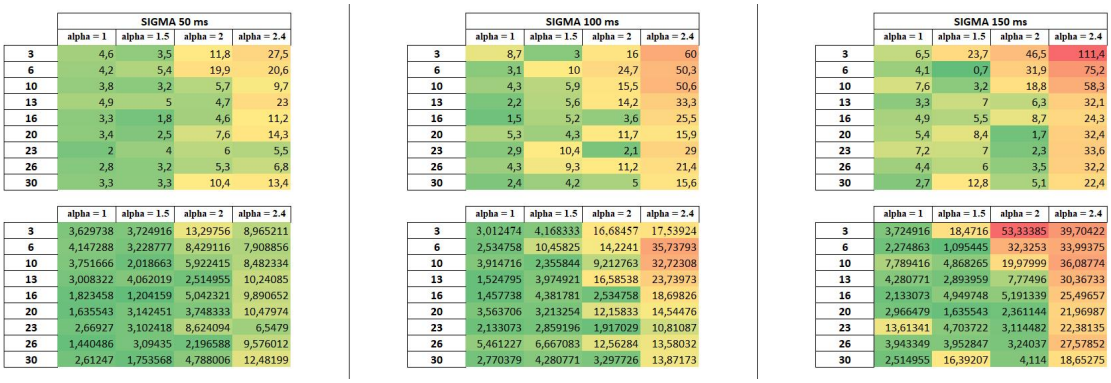
post	SNR	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
0	3	0,892	0,176	0,933	0,178	0,911	0,173
1	6	0,967	0,115	0,967	0,115	0,967	0,115
2	10	1,000	0,000	1,000	0,000	1,000	0,000
3	13	1,000	0,000	1,000	0,000	1,000	0,000
4	16	1,000	0,000	1,000	0,000	1,000	0,000
5	20	1,000	0,000	1,000	0,000	1,000	0,000
6	23	1,000	0,000	1,000	0,000	1,000	0,000
7	26	1,000	0,000	1,000	0,000	1,000	0,000
8	30	1,000	0,000	1,000	0,000	1,000	0,000

post	SNR	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
0	3	0,875	0,194	0,917	0,199	0,894	0,193
1	6	0,950	0,173	0,950	0,173	0,950	0,173
2	10	0,983	0,058	0,983	0,058	0,983	0,058
3	13	1,000	0,000	1,000	0,000	1,000	0,000
4	16	1,000	0,000	1,000	0,000	1,000	0,000
5	20	1,000	0,000	1,000	0,000	1,000	0,000
6	23	1,000	0,000	1,000	0,000	1,000	0,000
7	26	1,000	0,000	1,000	0,000	1,000	0,000
8	30	1,000	0,000	1,000	0,000	1,000	0,000

(b) *Metriche in funzione dell'SNR con post-processamento*

Figura 4.3: *Risultati delle metriche sull'output prima e dopo il post-processamento sui due istanti di transizione calcolate attraverso la tolleranza definita; le performance riportate sono le medie su tutti i 540 segnali in funzione dei valori di SNR considerati.*

In aggiunta, sono riportate le latency di onset e offset con le relative standard deviation per l'output della rete a seguito del post-processamento per ogni combinazione tra gli SNR, alpha e sigma considerati per onset e offset[Fig. 4.4, 4.5].



SIGMA 50 ms				
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	4,6	3,5	11,8	27,5
6	4,2	5,4	19,9	20,6
10	3,8	3,2	5,7	9,7
13	4,9	5	4,7	23
16	3,3	1,8	4,6	11,2
20	3,4	2,5	7,6	14,3
23	2	4	6	5,5
26	2,8	3,2	5,3	6,8
30	3,3	3,3	10,4	13,4

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	3,629738	3,724916	13,29756	8,965211
6	4,147288	3,228777	8,429116	7,908856
10	3,751666	2,018663	5,922415	8,482334
13	3,008322	4,062019	2,514955	10,24085
16	1,823458	1,204159	5,042321	9,890652
20	1,635543	3,142451	3,748333	10,47974
23	2,66927	3,102418	8,624094	6,5479
26	1,440486	3,09435	2,196588	9,576012
30	2,61247	1,753568	4,788006	12,48199

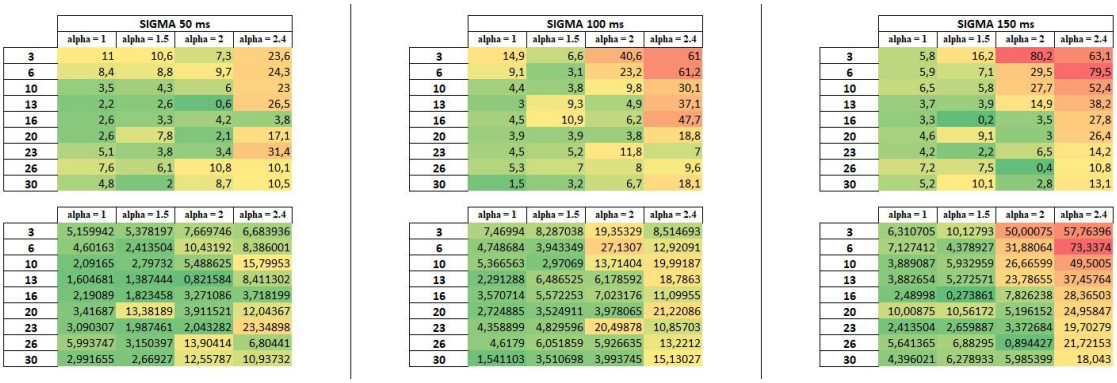
SIGMA 100 ms				
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	8,7	3	16	60
6	3,1	10	24,7	50,3
10	4,3	5,9	15,5	50,6
13	2,2	5,6	14,2	33,3
16	1,5	5,2	3,6	25,5
20	5,3	4,3	11,7	15,9
23	2,9	10,4	2,1	29
26	4,3	9,3	11,2	21,4
30	2,4	4,2	5	15,6

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	3,012474	4,168333	16,68457	17,53924
6	2,534758	10,45825	14,2241	35,73793
10	3,914716	2,355844	9,212763	32,72308
13	1,524795	3,974921	16,58538	23,73973
16	1,457738	4,381781	2,534758	18,69826
20	3,563706	3,213254	12,15833	14,54476
23	2,133073	2,859196	1,917029	10,81087
26	5,461227	6,667083	12,56284	13,58032
30	2,770379	4,280771	3,297726	13,87173

SIGMA 150 ms				
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	6,5	23,7	46,5	111,4
6	4,1	0,7	31,9	75,2
10	7,6	3,2	18,8	58,3
13	3,3	7	6,3	32,1
16	4,9	5,5	8,7	24,3
20	5,4	8,4	1,7	32,4
23	7,2	7	2,3	33,6
26	4,4	6	3,5	32,2
30	2,7	12,8	5,1	22,4

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	3,724916	18,4716	53,33385	39,70422
6	2,274863	1,095445	32,3253	33,99375
10	7,789416	4,868265	19,97999	36,08774
13	4,280771	2,893959	7,77496	30,36733
16	2,133073	4,949748	5,191339	25,49657
20	2,966479	1,635543	2,361144	21,96987
23	13,61341	4,703722	3,114482	22,38135
26	3,943349	3,952847	3,24037	27,57852
30	2,514955	16,39207	4,114	18,65275

Figura 4.4: *Risultati latencies medie (prima riga) e std delle latencies(seconda riga) sull'istante di onset dopo il post-processamento;ogni colonna di tabelle rappresenta i risultati per uno specifico sigma; con la gradazione di verde vengono evidenziati i risultati applicabili nella ricerca di base secondo il criterio fornito in [1]*



SIGMA 50 ms				
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	11	10,6	7,3	23,6
6	8,4	8,8	9,7	24,3
10	3,5	4,3	6	23
13	2,2	2,6	0,6	26,5
16	2,6	3,3	4,2	3,8
20	2,6	7,8	2,1	17,1
23	5,1	3,8	3,4	31,4
26	7,6	6,1	10,8	10,1
30	4,8	2	8,7	10,5

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	5,159942	5,378197	7,669746	6,683936
6	4,60163	2,413504	10,43192	8,386001
10	2,09165	2,79732	5,488625	15,79953
13	1,604681	1,387444	0,821584	8,411302
16	2,19089	1,823458	3,271086	3,718199
20	3,41687	13,38189	3,911521	12,04367
23	3,090307	1,987461	2,043282	23,34898
26	5,993747	3,150397	13,90414	6,80441
30	2,991655	2,66927	12,55787	10,93732

SIGMA 100 ms				
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	14,9	6,6	40,6	61
6	9,1	3,1	23,2	61,2
10	4,4	3,8	9,8	30,1
13	3	9,3	4,9	37,1
16	4,5	10,9	6,2	47,7
20	3,9	3,9	3,8	18,8
23	4,5	5,2	11,8	7
26	5,3	7	8	9,6
30	1,5	3,2	6,7	18,1

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	7,46994	8,287038	19,35329	8,514693
6	4,748684	3,943349	27,1307	12,92091
10	5,366563	2,97069	13,71404	19,99187
13	2,291288	6,486525	6,178592	18,7863
16	3,570714	5,572253	7,023176	11,09955
20	2,724885	3,524911	3,978065	21,22086
23	4,358899	4,829596	20,49878	10,85703
26	4,6179	6,051859	5,926635	13,2212
30	1,541103	3,510698	3,993745	15,13027

SIGMA 150 ms				
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	5,8	16,2	80,2	63,1
6	5,9	7,1	29,5	79,5
10	6,5	5,8	27,7	52,4
13	3,7	3,9	14,9	38,2
16	3,3	0,2	3,5	27,8
20	4,6	9,1	3	26,4
23	4,2	2,2	6,5	14,2
26	7,2	7,5	0,4	10,8
30	5,2	10,1	2,8	13,1

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	6,310705	10,12793	50,00075	57,76396
6	7,127412	4,378927	31,88064	73,3374
10	3,889087	5,932959	26,66599	49,5005
13	3,882654	5,272571	23,78655	37,45764
16	2,48998	0,273861	7,826238	28,36503
20	10,00875	10,56172	5,196152	24,95847
23	2,413504	2,659887	3,372684	19,70279
26	5,641365	6,88295	0,894427	21,72153
30	4,396021	6,278933	5,985399	18,043

Figura 4.5: *Risultati latencies medie(prima riga) e std delle latencies(seconda riga) sull'istante di offset dopo il post-processamento; ogni colonna di tabelle rappresenta i risultati per uno specifico sigma*

Il calcolo dell'errore assoluto sull'istante di predizione è stato determinato anche per l'output senza post-processamento e nelle figure 4.6 e 4.7 (visibili nelle seguenti pagine) sono riportati dei plot di confronto tra le latency della situazione prima e dopo il post-processamento per entrambi gli istanti predetti.

Per i grafici in figura 4.6 che riportano le latency in funzione degli alpha, sull'asse delle ascisse le varie combinazioni di alpha e sigma sono:

- Per $1 <= x <= 4$, $\sigma = 50ms$, ci sono i valori medi delle latencies per $\alpha = 1$ ($x = 1$) , $\alpha = 1.5$ ($x = 2$), $\alpha = 2$ ($x = 3$) , $\alpha = 2.4$ ($x = 4$);
- Per $5 <= x <= 8$, $\sigma = 100ms$, ci sono i valori medi delle latencies per $\alpha = 1$ ($x = 5$) , $\alpha = 1.5$ ($x = 6$), $\alpha = 2$ ($x = 7$) , $\alpha = 2.4$ ($x = 8$)

- Per $9 <= x <= 12$, $\sigma = 150ms$, ci sono i valori medi delle latencies per $\alpha = 1$ ($x = 9$) , $\alpha = 1.5$ ($x = 10$), $\alpha = 2$ ($x = 11$) , $\alpha = 2.4$ ($x = 12$)

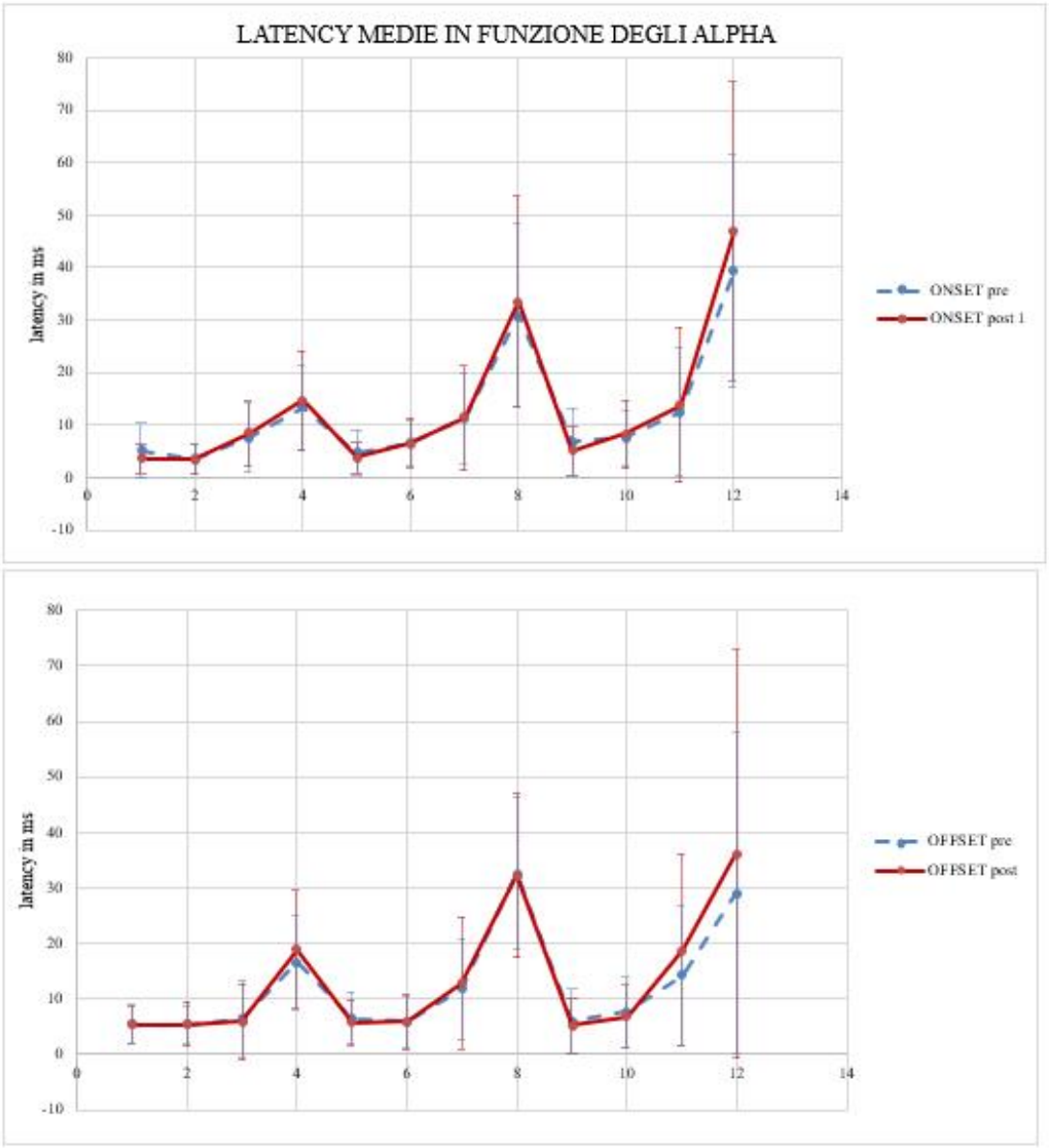


Figura 4.6: *Risultati latency sugli istanti di ONSET e OFFSET in funzione degli alpha prima e dopo il post-processamento sull'output della rete; le barre verticali rappresentano la std media per il valore di latency in cui sono centrate*

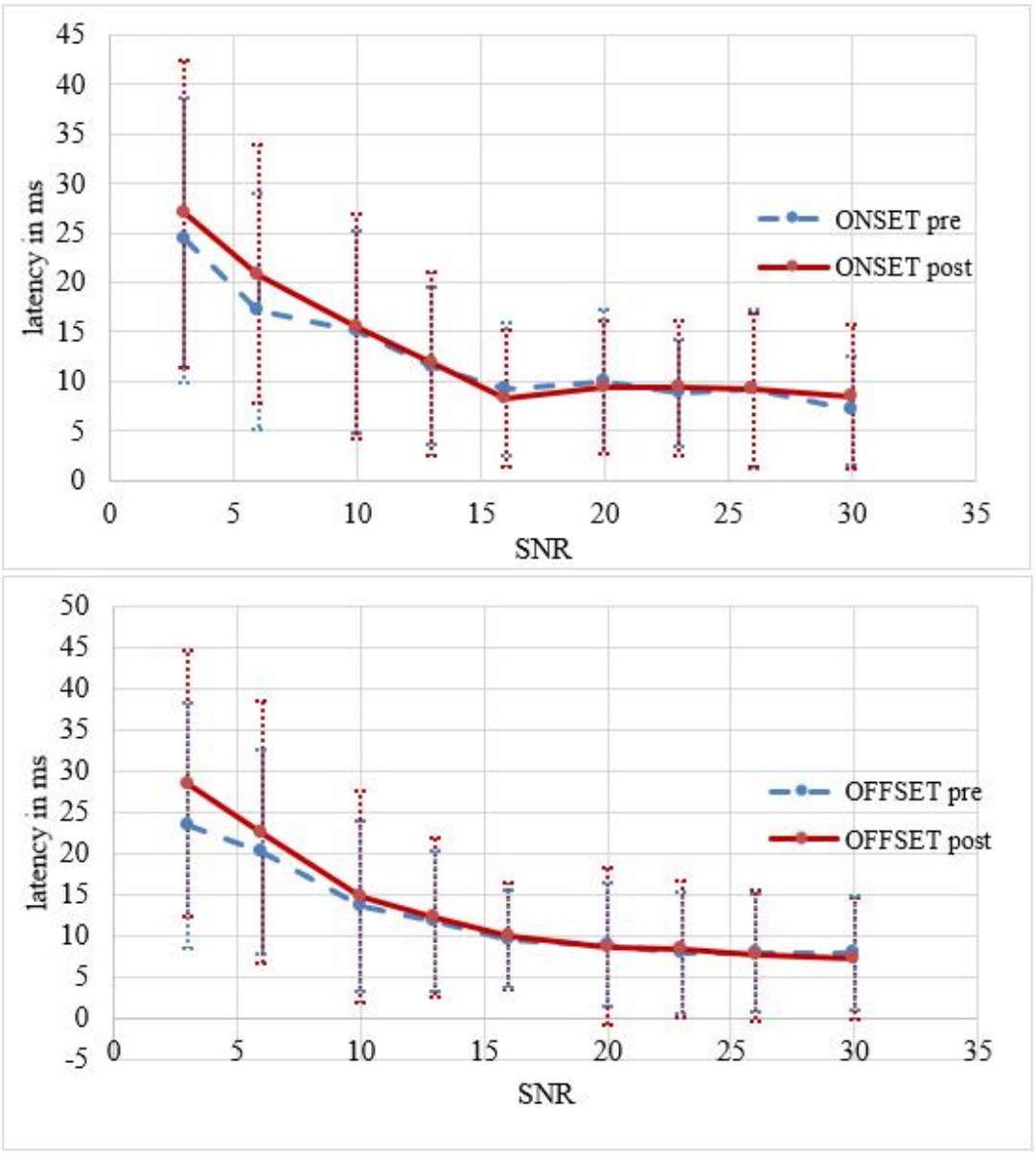


Figura 4.7: *Grafici che pongono a confronto le latency con o senza post-processamento in funzione di un SNR crescente. ONSET/OFFSET pre sta ad indicare le latency ottenute senza il post-processamento; le barre verticali rappresentano la std media per il valore di latency in cui sono centrate*

Capitolo 5

Discussione

5.1 Commento dei risultati

5.1.1 L'effetto del post-processamento

Per comprendere l'effetto del post-processamento verranno presi in esame i risultati delle metriche e delle latency sui due istanti di transizione.

L'effetto su falsi positivi e falsi negativi

Per quanto riguarda le metriche riassunte nella figura 4.1, notiamo che l'effetto del post-processamento è generalmente quello voluto, cioè un aumento della precisione per entrambi gli istanti di circa un 20%, che indica una diminuzione del numero delle transizioni false positive; comunque, è presente un leggero peggioramento di 1 punto percentuale sui recall di entrambi gli istanti, che denota l'introduzione di alcune transizioni false negative, cioè l'assenza di transizioni true positive all'interno della tolleranza. Tuttavia, questo calo è di poca rilevanza se paragonato all'incremento sulla precision e infatti i risultati per f1score crescono di circa un 13% per onset e offset arrivando al 98% circa per entrambi gli istanti.

Per dettagliare il funzionamento del post-processamento sempre per quanto riguarda le false transizioni, sia positive sia negative, prendiamo in esame le figure 4.2 e 4.3, le quali descrivono le metriche di predizione sugli istanti predetti in funzione di alpha e SNR, rispettivamente. Notiamo che per SNR $>$ 10dB non ci sono né false transizioni FP (False Positive), né false transizione FN (False Negative); la condizione peggiore avviene chiaramente per SNR $= 3$, ma comunque l'f1score è pari al 91% per l'onset e pari all'89% per l'offset: in entrambi i casi tra recall e precision è quest'ultima che presenta i risultati peggiori, indicando la presenza di un maggior numero di FP , piuttosto che FN con la diminuzione dell'SNR.

Detto questo, l'effetto del post-processamento è in generale positivo e le considerazioni

fatte sulle medie dei risultati in figura 4.1 possono essere sviluppate attraverso questi dati, poiché si può comprendere che l'aumento della precisione è presente per ogni SNR e che il calo della recall è presente solo per SNR minori di 10dB. In particolare, anche se la precision risulta avere generalmente percentuali minori della recall per bassi SNR è anche la metrica che subisce la variazione maggiore e l'aumento maggiore grazie al post-processamento, con un salto del 46% circa per l'onset dei segnali a 3dB e del 49% circa per l'offset degli stessi segnali.

Invece, l'analisi della figura 4.2, che rappresenta le stesse metriche sugli istanti in funzione di alpha e sigma variabili, è cruciale per comprendere quale sia la relazione tra FP,FN e forma d'onda del segnale; anche se notiamo sempre lo stesso aumento generale della precision su tutte le combinazioni di alpha e sigma, ciò che è interessante è che il calo della recall non è distribuito su ogni forma d'onda ma solo sulle forme d'onde caratterizzate da $\alpha = 2$ e $\alpha = 2.4$ con sigma = 0.15s (le ultime due righe); ciò significa che le uniche transizioni FN sono presenti solo per bassi SNR e per le due forme d'onda citate e in ogni altro caso, per ogni SNR considerato, la rete produce sempre almeno una transizione TP (true positive). In particolar modo, l'ultima riga ($\alpha = 2.4, \sigma = 0.15s$) è stata evidenziata perché rappresenta il peggior caso, con il calo maggiore nella recall per entrambi gli istanti.

Per queste ultime due parametrizzazioni (corrispondenti ai massimi valori di alfa e gamma) è necessaria una precisazione, che sarà utile anche nelle seguenti sezioni, e che riguarda l'effettiva importanza di questi risultati per una possibile applicazione del metodo proposto. Infatti, ricordiamo come le durate di attivazione muscolare più diffuse siano sempre minori delle durate ottenute con questi parametri, 0.6s e 0.72s rispettivamente. Nel caso di $\alpha = 2, \sigma = 0.15s$, con una durata di 0.6s, abbiamo già una condizione al limite della possibile durata massima per l'attivazione dei muscoli solitamente considerati durante l'analisi del cammino.

Infatti, una delle durate di attivazione più lunga è quella del tibiale anteriore, la quale è comunque pari solamente a circa il 50%-55% di un 'gait'. Pertanto, si può comprendere che, se già con le condizioni di alpha pari a 2 e sigma pari a 0.15s siamo ai limiti della simulazione della realtà, con l'ultima riga stiamo realmente indagando su una condizione di attivazione (pari al 72% del totale 'gait') che ha solamente importanza teorica.

In conclusione di questa sottosezione vengono mostrati nelle figure 5.1 e 5.2, alcuni esempi dell'effetto del post-processamento, e viene riassunto nei seguenti punti ciò che è stato trattato nelle precedenti righe :

- l'effetto del post-processamento produce un aumento positivo della precision distribuito su tutti gli SNR e tutte le combinazioni di alpha e sigma, indicando una

generale diminuzione dei FP

- è presente un leggero calo della recall in entrambi gli istanti, che corrisponde all'introduzione di transizioni FN solo per SNR bassi e transizioni caratterizzate da $\alpha = 2, 2.4$ e $\sigma = 0.15s$, le quali hanno una bassa rilevanza dal punto di vista dell'applicabilità per l'eccessiva durata

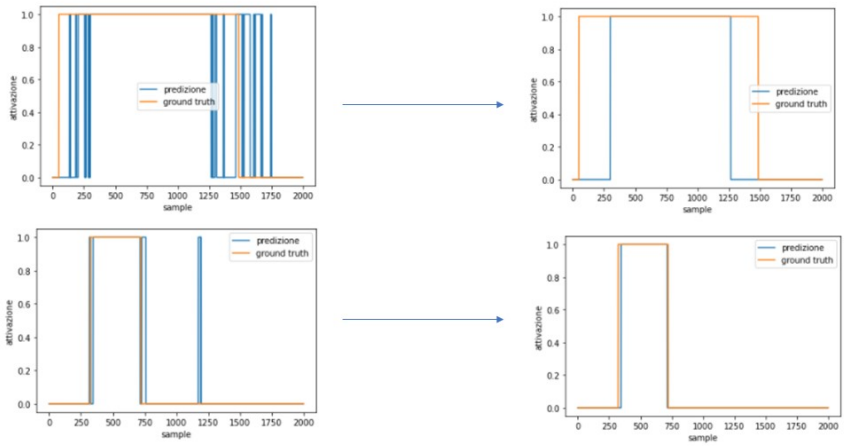


Figura 5.1: *In quest'immagine è illustrato brevemente l'effetto del post processing su due esempi; nel primo risultano evidenti le considerazioni sull'effetto del post processamento, che è particolarmente evidente per attivazioni di durata maggiore. Nel secondo esempio è riportata la tipica situazione per attivazioni di durata medio-bassa dove il post processamento va semplicemente ad eliminare quelle attivazioni che non hanno importanza fisiologica*

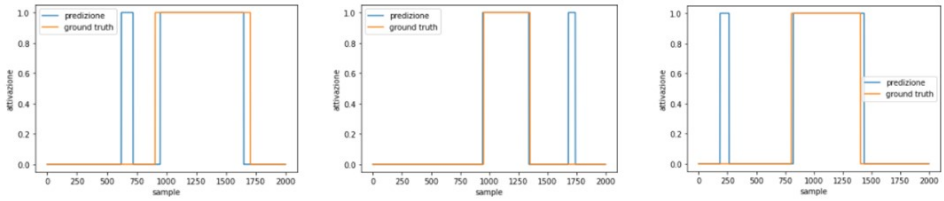


Figura 5.2: *Esempi di errata predizione di un'intera attivazione muscolare; sono tutti e tre presenti per SNR pari a 3dB e queste errate attivazioni potrebbero essere eliminate con un post-processamento più 'invasivo' che vada ad eliminare la transizioni di durata minore ai 100 campioni; questi tre esempi sono quelli che causano un abbassamento della precision nelle righe 3,7 e 9 della figure 4.2*

L'effetto del post-processamento sulle latency

Dall'analisi delle metriche di precision, recall e f1score è complesso comprendere quale sia la situazione per l'errore assoluto sulla predizione dei singoli istanti (latency); per questo motivo sono stati utilizzati i risultati nelle figure 4.6 e 4.7, le quali mostrano le latency dell'output della rete prima e dopo il post-processamento in funzione di alpha e SNR rispettivamente.

Per quanto riguarda riguarda le latency in funzione di alpha, la differenza più importante tra la situazione di pre e post processamento dell'output è presente per x=12 ($\sigma = 0.15s, \alpha = 2.4$), dove si può distinguere una differenza di circa 7ms per entrambi gli istanti. Per quanto riguarda le latency in funzione di SNR crescenti in figura 4.3, la variazione maggiore si ha per SNR $<$ 10dB. Queste considerazioni assieme alla precedente sottosezione permettono di comprendere che l'effetto del post-processamento è generalmente positivo per SNR $>$ 13dB, poiché azzera il numero delle false transizioni sia FP sia FN e non produce un peggioramento nelle latency. L'effetto peggiorativo è invece evidente per bassi SNR e particolari forme d'onda con una lunga durata d'attivazione.

5.1.2 Confronto tra i risultati per onset e offset

Questa sottosezione è dedicata ad un confronto tra le predizioni di onset/offset nell'output della rete a seguito del post-processamento per comprendere se il comportamento del detector sia infuenzato dalla tipologia di instante da predire. I risultati delle latency e delle std per entrambi gli istanti sono riassunti dai forest plot in figura 5.4. Notiamo come sia in funzione di alpha e sigma crescenti, sia in funzione di SNR crescenti il comportamento del detector nei confronti dei due istanti sia confrontabile. Inoltre la maggior parte dei punti del grafico in funzione delle alpha è sotto i 10 ms, soglia sotto la quale i risultati sono accettabili nella ricerca di base e per applicazioni cliniche [1, vedi p.297]. Questo comportamento è visibile anche nelle tabelle riassuntive [Fig. 4.5 e 4.4] dalle quali si può comprendere che i risultati per $1 < \alpha < 1.5$ siano soddisfacenti per ogni sigma e ogni SNR considerato; i risultati per $\alpha = 2$ sono soddisfacenti per ogni sigma e per SNR $>$ 10dB. La vera problematica è presente per attivazioni formate da $\alpha = 2.4$, l'ultima colonna di ogni tabella, dove i valori sono per la maggior parte sopra la soglia di accettabilità.

Infatti, ci sono dei picchi nel grafico delle latency in funzione delle alpha 5.4 che corrispondono agli $\alpha = 2.4$ per tutti e tre i sigma:

1. Per $\alpha = 2.4, \sigma = 50ms$
   la latency media dell'onset è 14,67 +- 9,40 , offset 18,92 +- 10,68

2. Per $\alpha = 2.4, \sigma = 100ms$
   la latency media dell'onset è 33,51 +- 20,14, offset 32,29 +- 14,64

3. Per $\alpha = 2.4, \sigma = 150ms$
   la latency media dell'onset è 46,88 +- 28,47, offset 36,17 +- 36,76

Inoltre è possibile osservare che, a parità di sigma , le latency per entrambi gli istanti crescono con il crescere degli alpha; invece, a parità di alpha, notiamo comunque un leggero incremento del valore delle latency per sigma crescenti che è più evidente per $\alpha = 2.4$. Da queste considerazioni sembra essere chiaro che la predizione degli istanti di transizione sia quindi influenzata dalla forma d'onda dell'inviluppo gaussiano che genera il segnale e in particolare dall'alpha, che definisce la finestra di attivazione, e da sigma, che definisce la larghezza della campana.

Questo comportamento potrebbe essere spiegato dal fatto che la transizione da rumore a segnale è tanto più repentina e quindi facilmente individuabile quanto sono più bassi i valori di alpha e sigma. Infatti, se l'alpha è piccolo la finestra di attivazione del segnale è modulata da valori della gaussiana in una regione vicino al punto di massimo (punto medio) ed è quindi chiaro che la modulazione maggiore produca degli eventi di transizione che sono più facilmente riconoscibili. Al contrario con alpha più grandi, vengono presi in considerazione degli intervalli dove la modulazione data dalla gaussiana è minore ed è quindi più complesso riconoscere la differenza tra rumore e segnale. Invece, il parametro sigma definisce la larghezza della gaussiana e la pendenza dei sui 'rami': con sigma piccoli avremo gaussiane più strette e quindi dei 'rami' con pendenze maggiori e transizioni più repentine, mentre con sigma più grandi la pendenza dei 'rami' della gaussiana diventa meno ripida. Osservando le figure 5.3, dove sono mostrate le campane gaussiane per $\alpha = 2.4$ e σ variabili, possiamo notare come al variare di sigma la pendenza delle gaussiane cambia e come per sigma = 0.15s abbiamo la crescita più lenta dei valori per la curva modulante.

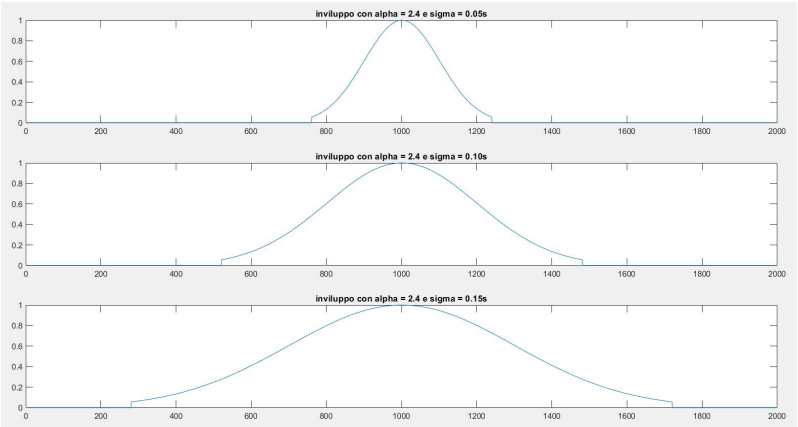


Figura 5.3: *Plot di tre inviluppi gaussiani con alpha=2.4*

Pertanto, possiamo riassumere le precedenti considerazioni in due punti:

- A parità di sigma, il parametro α indica la durata della finestra di attivazione; infatti, l'inizio dell'attivazione del segnale è calcolato come $\mu - \alpha\sigma$ e la fine dell'attivazione come $\mu + \alpha\sigma$. Per alpha più piccoli avremo onset e offset più facili da individuare poiché i valori della gaussiana che modula il segnale di attivazione sono più vicini al valore massimo della gaussiana (punto medio); per alpha più grandi, come 2 o 2.4, la durata dell'attivazione è maggiore e in prossimità delle transizioni è presente una modulazione minore della gaussiana. Questa considerazione potrebbe giustificare i picchi per α=2.4 nella figura 5.4.

- Invece, il parametro sigma definisce la larghezza della gaussiana; a parità di alpha influisce anch'esso sulla dimensione della finestra di attivazione, infatti come vediamo nella figura 5.3 con il crescere di sigma cresce la durata dell'attivazione e cambia la forma d'onda della gaussiana. In particolare, la pendenza dei rami diminuisce con l'aumentare del sigma e questo rende il passaggio da rumore a segnale più 'lento'. Questa considerazione potrebbe giustificare il perché della crescita delle latency con il crescere di sigma e la presenza del massimo valore delle latency per $\alpha = 2.4$ e $\sigma = 0.15s$, cioè il massimo valore di sigma considerato.

Analizzando invece la figura 4.3 possiamo notare che sia latency sia std sui due istanti di transizione diminuiscono con l'aumentare dell'SNR; questo poteva essere anticipato, poiché con l'aumentare dell'SNR la rete deve discriminare tra un segnale sempre più presente in rapporto al rumore di background.

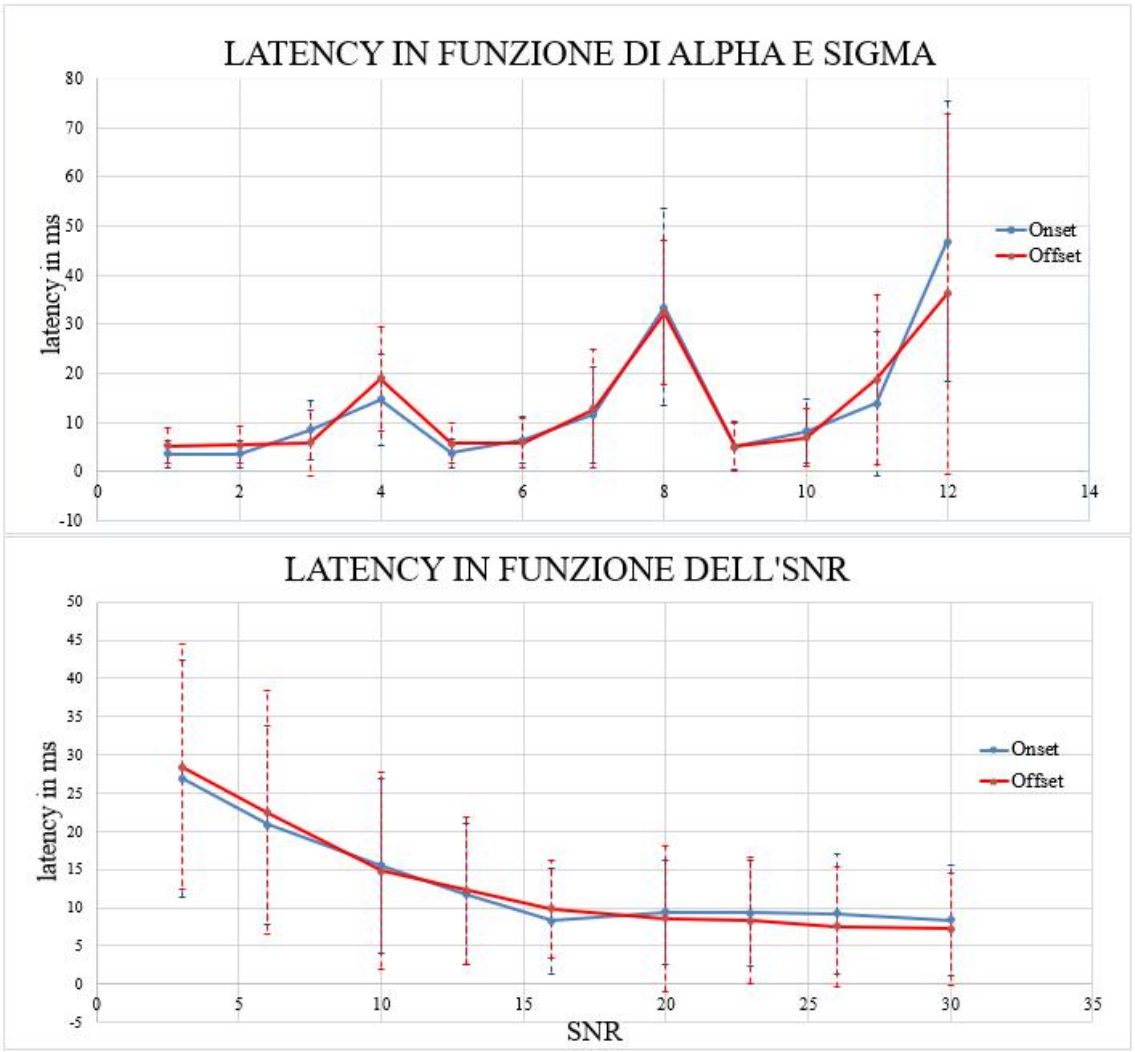


Figura 5.4: *Il primo grafico presenta sull'asse delle ordinate la latencies in ms calcolate come media su tutti gi SNR considerati per entrambi gli istanti di transizione in funzione di una specifica combinazione di alpha e sigma. Il secondo grafico riporta invece le latencies per entrambi gli istanti di transizione in funzione di SNR crescenti come media di tutti i risultati per uno specifico valore di SNR; le barre verticali per ogni valore all'interno dei grafici rappresentano le barre di errore, cioè le std medie corrispondenti a quel valore specifico*

5.2 Algoritmo double threshold*

L'algoritmo proposto da Bonato nel 98' è di fatto uno degli algoritmi più utilizzati per la task dell'individuazione degli intervalli di attivazione. Può essere descritto attraverso queste fasi:

1. Filtraggio pre-whitening
   Questo filtraggio ha lo scopo di rendere incorrelati i valori che formano il segnale sEMG; Nel caso dei segnali sintetici questo filtraggio non è necessario perchè sono già creati come serie di valori incorrelati.

2. Formazione della serie ausiliaria z
   Detto x il segnale incorrelato, allora la serie ausiliaria z è definita come segue:

   $$z_j = x_j^2 + x_{j-1}^2$$

   La serie ha un numero dimezzato di valori rispetto al segnale originale.

3. Utilizzo della prima soglia
   Il primo threshold viene applicato alla serie z e un campione è posto ad 1 se lo supera.

4. Utilizzo della seconda soglia
   Il segnale on-off creato con il criterio del superamento della prima soglia in ampiezza, viene segmentato in finestre di dimensione m, all'interno delle quali viene applicata la seconda soglia r0, che specifica il numero minimo di attivazioni che devono essere presenti all'interno della finestra considerata affinché la finestra stessa sia considerata un'attivazione.

5. Post processamento
   L'output dato da queste due soglie contiene solitamente molte transizioni errate e per eliminarle è presente una lavorazione ulteriore che va ad togliere le transizioni che durano meno di 30ms.

Il primo threshold viene calcolato quando l'utente sceglie quale sia la probabilità di falso allarme, la seconda soglia e la finestra per questa soglia. La scelta di questi parametri fornisce la probabilità di detection che si vuole ricavare. In generale, la scelta della finestra è limitata dalla time resolution che si vuole avere, la scelta di r0 viene solitamente impostata ad 1 per avere la probabilità di detection più alta e la Pfad (probabilità di falso allarme) è scelta in modo da ridurre il numero di false transizioni.

Per i 540 segnali in testing phase sono stati utilizzati come parametri: Pfad = 1%, m=10, r0=1 e di conseguenza è stato calcolato il threshold corrispondente ad ogni segnale in funzione della varianza di rumore, che è conosciuta poiché il segnale è sintetico. Non sono

stati effettuati ulteriori filtraggi prima di applicare il double threshold. I parametri sono stati scelti con le linee guida basate su considerazioni statistiche date dallo stesso articolo; la finestra è stata posta a 10 campioni per avere la stessa time-resolution dell'articolo di Bonato ed r0 è stato posto ad 1 per aumentare la probabilità di prediction; il valore Pfad che influenza le transizioni false positive è stato scelto in modo da ridurle in funzione di alcune prove direttamente sul testset.

Questo pone un vantaggio essenzialmente all'algoritmo di Bonato, che, però, è stato originariamente implementato per essere utilizzato da un utente esperto che possa, quindi, ottimizzare questi parametri grazie ad una conoscenza a priori del dataset e ad una procedura descritta nella stesso articolo in modo da ottenere le probabilità di detection e falso allarme desiderate. In questo caso, utilizzando una probabilità di falso allarme maggiore (Pfad), come quella del 5% utilizzata nell'articolo originale, avrebbe comportato un maggior numero di false transizioni e quindi quel parametro è stato abbassato.

**L'algoritmo è stato implementato completamente per la parte della 'detection unit', che comprende quindi il calcolo della serie ausiliaria e l'applicazione delle soglie, su Matlab seguendo le indicazioni dell'articolo, mentre il post-processamento è stata ripreso da un lavoro precedente di alcuni dottorandi dell'univpm, fornito dal prof. Di Nardo, e che ho leggermente modificato*

5.2.1 Confronto dei risultati con double threshold

In questa sezione verranno mostrati i risultati ottenuti con l'algoritmo double threshold per quanto riguarda le metriche di predizione sugli istanti di onset/offset (precision, recall e f1score) e le latency con le relative std. Per ogni figura mostrata, è presente un confronto con il MLP e delle gradazione di colore che con il verde indicano i risultati migliori tra i due approcci.

I risultati di precision,recall e f1score

	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
Bonato	0,989	0,028	0,989	0,028	0,989	0,028
MLP	0,984	0,038	0,989	0,024	0,986	0,032

	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
Bonato	0,965	0,076	0,965	0,076	0,965	0,076
MLP	0,979	0,049	0,983	0,035	0,981	0,043

Figura 5.5: *Metriche a confronto tra l'approccio con MLP e il metodo di Bonato; in verde i valori migliori*

La figura 5.5 mostra un riassunto del comportamento dei due approcci con le metriche medie su tutto il testset; notiamo che i due approcci hanno mediamente un comportamento simile per l'individuazione degli onset con il double threshold che presenta delle performance leggermente migliori; d'altra parte per l'individuazione dell'offset l'algoritmo di Bonato peggiora mediamente di 2 punti percentuali sull'f1score, mentre l'approccio tramite rete neurale sembra essere più robusto ed avere delle performance stabili intorno al 98% per f1score sia nell'onset (98.6%) sia nell'offset(98.1%).

Per comprendere nel dettaglio le performance analizziamo le figure 5.6 e 5.7, che rappresentano delle tabelle con le metriche in funzione di alpha e SNR rispettivamente; dalla figura 5.6 (pagina successiva) possiamo constatare attraverso le gradazioni di colore lo stesso trend delle medie totali: generalmente la rete MLP ha performance migliori sull'offset, mentre sull'onset le performance sono paragonabili, se non per alcuni casi nelle righe 3,7,9 dove è presente un leggera differenza nella precision e di conseguenza nell'f1score. Generalmente, il metodo MLP presenta una recall più stabile tra onset e offset in funzione della forma d'onda caratterizzata dai parametri α e σ, indicando che il modello proposto fornisce più volte un true positive rispetto al modello di Bonato, il quale presenta quindi più casi di FN.

Bonato	sigma	alpha	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
1	0,05	1	1,000	0,000	1,000	0,000	1,000	0,000
2	0,05	1,5	1,000	0,000	1,000	0,000	1,000	0,000
3	0,05	2	1,000	0,000	1,000	0,000	1,000	0,000
4	0,05	2,4	1,000	0,000	1,000	0,000	1,000	0,000
5	0,1	1	1,000	0,000	1,000	0,000	1,000	0,000
6	0,1	1,5	1,000	0,000	1,000	0,000	1,000	0,000
7	0,1	2	1,000	0,000	1,000	0,000	1,000	0,000
8	0,1	2,4	1,000	0,000	1,000	0,000	1,000	0,000
9	0,15	1	1,000	0,000	1,000	0,000	1,000	0,000
10	0,15	1,5	1,000	0,000	1,000	0,000	1,000	0,000
11	0,15	2	0,978	0,067	0,978	0,067	0,978	0,067
12	0,15	2,4	0,889	0,267	0,889	0,267	0,889	0,267

MLP	sigma	alpha	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
1	0,05	1	1,000	0,000	1,000	0,000	1,000	0,000
2	0,05	1,5	1,000	0,000	1,000	0,000	1,000	0,000
3	0,05	2	0,981	0,056	1,000	0,000	0,990	0,030
4	0,05	2,4	1,000	0,000	1,000	0,000	1,000	0,000
5	0,1	1	1,000	0,000	1,000	0,000	1,000	0,000
6	0,1	1,5	1,000	0,000	1,000	0,000	1,000	0,000
7	0,1	2	0,981	0,056	1,000	0,000	0,990	0,030
8	0,1	2,4	1,000	0,000	1,000	0,000	1,000	0,000
9	0,15	1	0,981	0,056	1,000	0,000	0,990	0,030
10	0,15	1,5	1,000	0,000	1,000	0,000	1,000	0,000
11	0,15	2	0,978	0,067	0,978	0,067	0,978	0,067
12	0,15	2,4	0,889	0,226	0,889	0,226	0,889	0,226

(a) *Confronto tra le metriche in funzione di alpha e sigma per l'istante di OFFSET*

Bonato	sigma	alpha	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
1	0,05	1	1,000	0,000	1,000	0,000	1,000	0,000
2	0,05	1,5	1,000	0,000	1,000	0,000	1,000	0,000
3	0,05	2	1,000	0,000	1,000	0,000	1,000	0,000
4	0,05	2,4	1,000	0,000	1,000	0,000	1,000	0,000
5	0,1	1	1,000	0,000	1,000	0,000	1,000	0,000
6	0,1	1,5	1,000	0,000	1,000	0,000	1,000	0,000
7	0,1	2	0,933	0,200	0,933	0,200	0,933	0,200
8	0,1	2,4	0,933	0,141	0,933	0,141	0,933	0,141
9	0,15	1	1,000	0,000	1,000	0,000	1,000	0,000
10	0,15	1,5	0,978	0,067	0,978	0,067	0,978	0,067
11	0,15	2	0,911	0,267	0,911	0,267	0,911	0,267
12	0,15	2,4	0,822	0,233	0,822	0,233	0,822	0,233

MLP	sigma	alpha	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
1	0,05	1	1,000	0,000	1,000	0,000	1,000	0,000
2	0,05	1,5	1,000	0,000	1,000	0,000	1,000	0,000
3	0,05	2	0,981	0,056	1,000	0,000	0,990	0,030
4	0,05	2,4	1,000	0,000	1,000	0,000	1,000	0,000
5	0,1	1	1,000	0,000	1,000	0,000	1,000	0,000
6	0,1	1,5	1,000	0,000	1,000	0,000	1,000	0,000
7	0,1	2	0,981	0,056	1,000	0,000	0,990	0,030
8	0,1	2,4	1,000	0,000	1,000	0,000	1,000	0,000
9	0,15	1	0,981	0,056	1,000	0,000	0,990	0,030
10	0,15	1,5	1,000	0,000	1,000	0,000	1,000	0,000
11	0,15	2	0,933	0,200	0,933	0,200	0,933	0,200
12	0,15	2,4	0,867	0,224	0,867	0,224	0,867	0,224

(b) *Confronto tra le metriche in funzione di alpha e sigma per l'istante di OFFSET*

Figura 5.6: *Tabelle che mostrano le metriche di predizione degli istanti di onset/offset dei due approcci confrontati in funzione di combinazioni di alpha e sigma; il colore verde è utilizzata per indicare le metriche migliori nel confronto tra i due approcci*

Inoltre, prendendo in esame le figure 5.7(pagina successiva), osserviamo che per SNR >13dB il comportamento tra i due approcci è confrontabile: non è presente alcuna transizione FN o FP; per bassi SNR il metodo di Bonato sembra avere delle performance migliori sull'onset con f1score di 91.7% e 98.3% per snr pari a 3 e 6 rispettivamente, che superano leggermente le statistiche di f1score per il modello MLP con 91.1% e 96.7% nelle stesse condizioni di SNR. D'altro canto, il modello MLP ha risultati migliori nell'individuazione dell'offset per SNR pari a 3dB e 13dB.

Concludendo questa sottosezione, entrambi gli approcci dimostrano di avere delle performance soddisfacenti con il MLP che supera il modello double threshold in particolar modo nell'individuazione degli istanti di offset; il metodo di Bonato presenta comunque dei risultati ottimi, soprattutto per l'istante di onset, superando leggermente la metodica proposta. Comunque, ciò che è rilevante è che entrambi i metodi per SNR $>$ 13dB non presentano false transizioni di alcun tipo e che il modello MLP non sembra essere soggetto a particolari cambiamenti nell'individuazione di un istante passando da onset a offset.

Bonato	SNR	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
0	3	0,917	0,233	0,917	0,233	0,917	0,233
1	6	0,983	0,058	0,983	0,058	0,983	0,058
2	10	1,000	0,000	1,000	0,000	1,000	0,000
3	13	1,000	0,000	1,000	0,000	1,000	0,000
4	16	1,000	0,000	1,000	0,000	1,000	0,000
5	20	1,000	0,000	1,000	0,000	1,000	0,000
6	23	1,000	0,000	1,000	0,000	1,000	0,000
7	26	1,000	0,000	1,000	0,000	1,000	0,000
8	30	1,000	0,000	1,000	0,000	1,000	0,000

MLP	SNR	precision ONSET	std	recall ONSET	std	f1 score ONSET	std
0	3	0,892	0,176	0,933	0,178	0,911	0,173
1	6	0,967	0,115	0,967	0,115	0,967	0,115
2	10	1,000	0,000	1,000	0,000	1,000	0,000
3	13	1,000	0,000	1,000	0,000	1,000	0,000
4	16	1,000	0,000	1,000	0,000	1,000	0,000
5	20	1,000	0,000	1,000	0,000	1,000	0,000
6	23	1,000	0,000	1,000	0,000	1,000	0,000
7	26	1,000	0,000	1,000	0,000	1,000	0,000
8	30	1,000	0,000	1,000	0,000	1,000	0,000

(a) *Metriche in funzione di SNR crescenti per l'istant di ONSET*

Bonato	SNR	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
0	3	0,783	0,301	0,783	0,301	0,783	0,301
1	6	0,950	0,124	0,950	0,124	0,950	0,124
2	10	0,983	0,058	0,983	0,058	0,983	0,058
3	13	0,967	0,115	0,967	0,115	0,967	0,115
4	16	1,000	0,000	1,000	0,000	1,000	0,000
5	20	1,000	0,000	1,000	0,000	1,000	0,000
6	23	1,000	0,000	1,000	0,000	1,000	0,000
7	26	1,000	0,000	1,000	0,000	1,000	0,000
8	30	1,000	0,000	1,000	0,000	1,000	0,000

MLP	SNR	precision OFFSET	std	recall OFFSET	std	f1 score OFFSET	std
0	3	0,875	0,194	0,917	0,199	0,894	0,193
1	6	0,950	0,173	0,950	0,173	0,950	0,173
2	10	0,983	0,058	0,983	0,058	0,983	0,058
3	13	1,000	0,000	1,000	0,000	1,000	0,000
4	16	1,000	0,000	1,000	0,000	1,000	0,000
5	20	1,000	0,000	1,000	0,000	1,000	0,000
6	23	1,000	0,000	1,000	0,000	1,000	0,000
7	26	1,000	0,000	1,000	0,000	1,000	0,000
8	30	1,000	0,000	1,000	0,000	1,000	0,000

(b) *Metriche in funzione di SNR crescenti per l'istant di ONSET*

Figura 5.7: *Tabella che mostra le metriche di predizione degli istanti di onset/offset dei due approcci confrontati in funzione degli SNR considerati; il colore verde è utilizzata per indicare le metriche migliori nel confronto tra i due approcci*

I risultati di latency e std a confronto

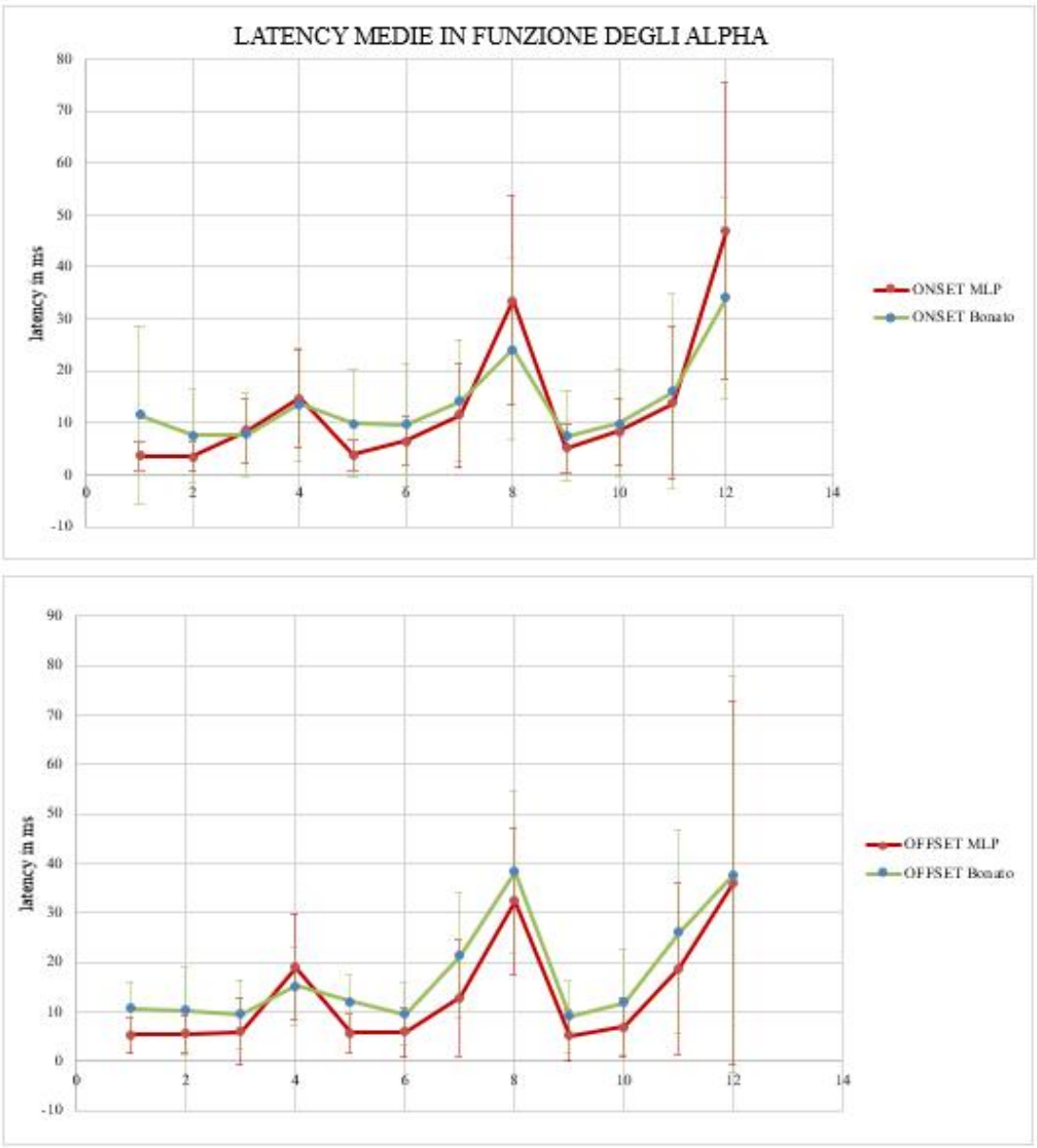


Figura 5.8: *Confronto tra le latency medie di offset e onset in funzione di alpha e sigma; le barre rappresentano l'errore, cioè la std.*

Per questa sottosezione verranno utilizzati i grafici 5.8 e 5.9, che rappresentano un confronto tra le latency dei due approcci in funzione di alpha e SNR rispettivamente. Per quanto riguarda i grafici in funzione delle alpha, possiamo notare che per l'istante di onset, il metodo MLP ha generalmente performance migliori per ogni alpha ad esclusione dei picchi per α=2.4, che rappresentano l'unica problematica. Invece, nell'individuazione dell'offset il MLP conferma il trend precedente, con latency minori del metodo di Bonato.

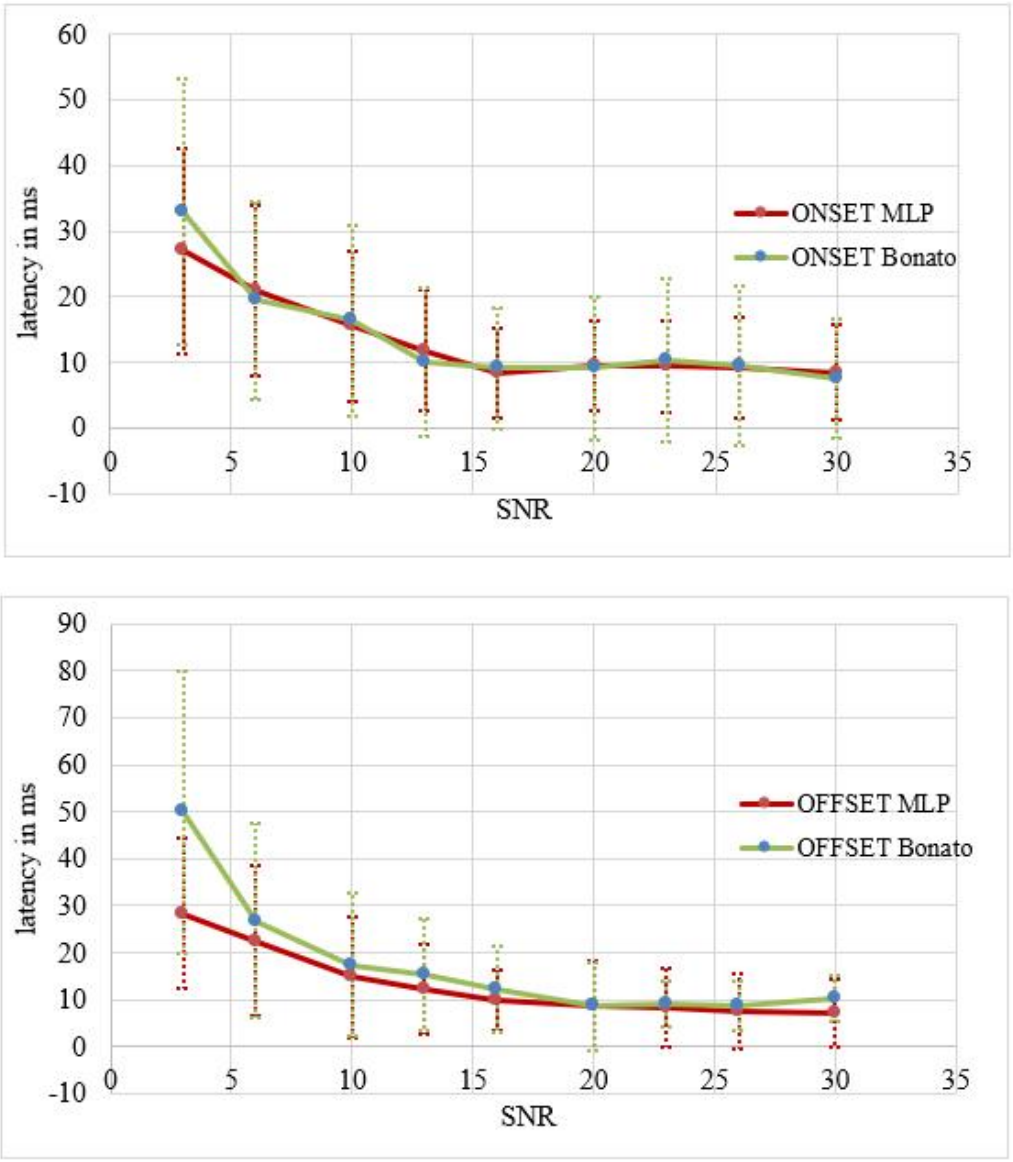


Figura 5.9: *Confronto tra le latency medie di offset e onset in funzione dell'SNR;le barre rappresentano l'errore, cioè la std.*

Per quanto riguarda i grafici in funzione del SNR [5.9], notiamo che per l'individuazione dell'onset i due metodi siano comparabili e che le latency medie quasi si sovrappongano per SNR >5dB con il MLP che presenta un leggero scarto positivo per SNR =3dB. Tuttavia, nella predizione dell'istante di offset la rete neurale ha prestazioni migliori per SNR<20dB e prestazione simili per SNR più alti.

In aggiunta, sono riportati degli esempi di predizione su segnali simulati nelle figure 5.10, 5.11, 5.12 e 5.13.

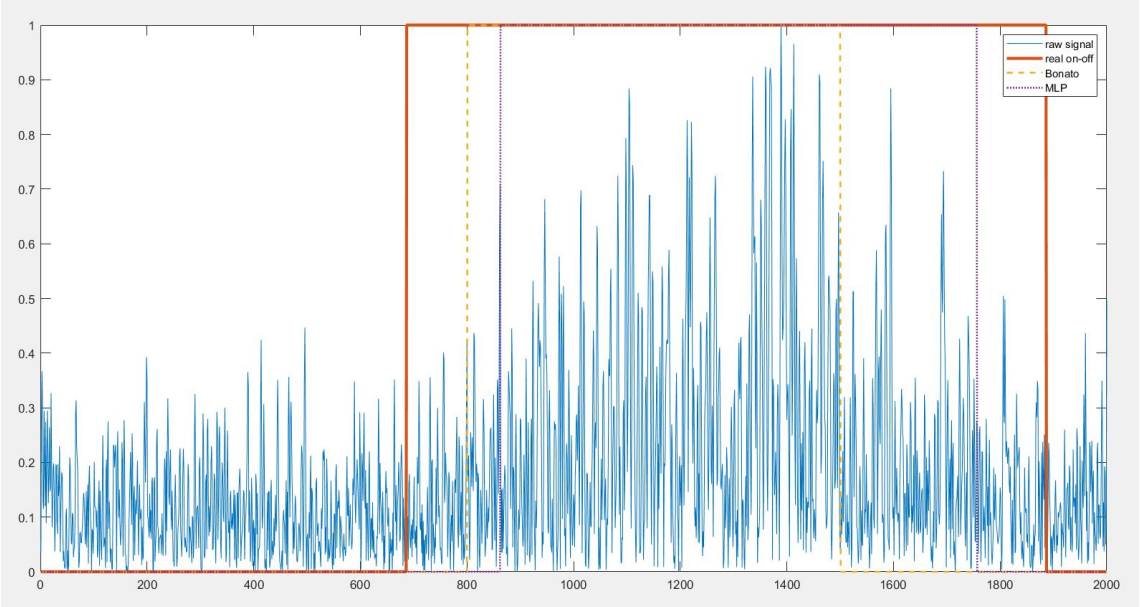


Figura 5.10: *esempio di un segnale con SNR pari a 3dB, sigma = 0.15s, alpha=2.4*

Nella prima figura viene mostrato un esempio di segnale con SNR pari a 3dB, $\alpha = 2.4$ e σ=0.15s; per SNR così bassi il metodo MLP ha delle performance migliori tra onset e offset come osserviamo dalla figure 5.9, anche se possiamo notare che il metodo double threshold riesce a predire un onset più vicino a quello reale, sbagliando con un margine maggiore l'offset.

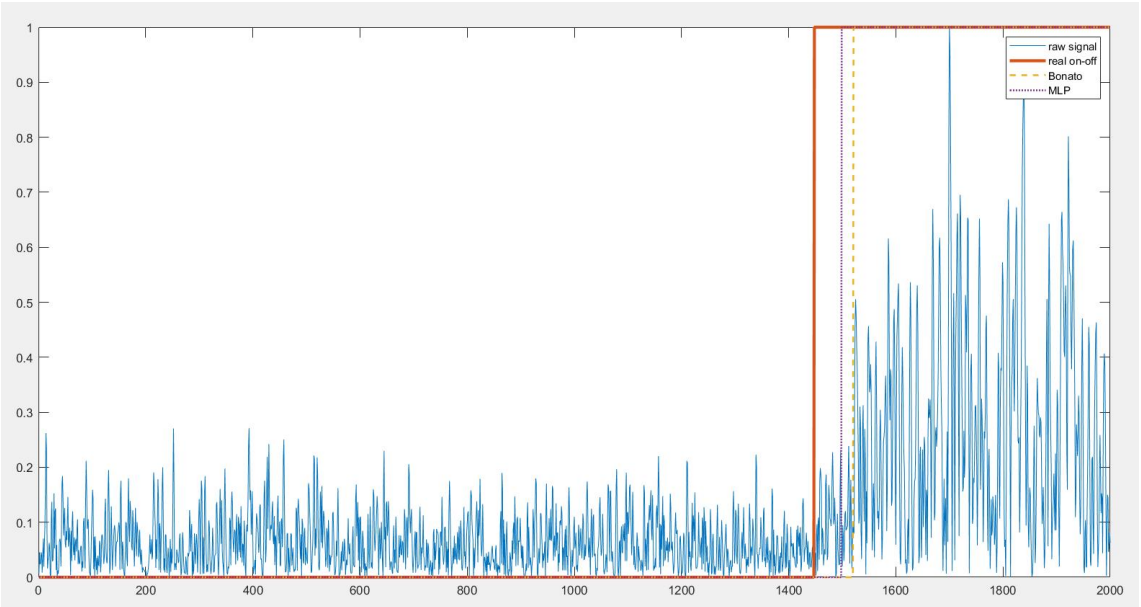


Figura 5.11: *esempio di un segnale con SNR pari a 6dB, sigma = 0.1s, alpha = 1.5*

Nella seconda figura, che rappresenta invece un segnale con SNR a 6dB, $\alpha = 1.5$ e σ=0.1s,notiamo come le predizioni di MLP e dell'algoritmo double-threshold siano molto vicine tra loro, anche se la rete neurale ha performance generalmente migliori, confermando l'andamento generalmente positivo del MLP per SNR bassi.

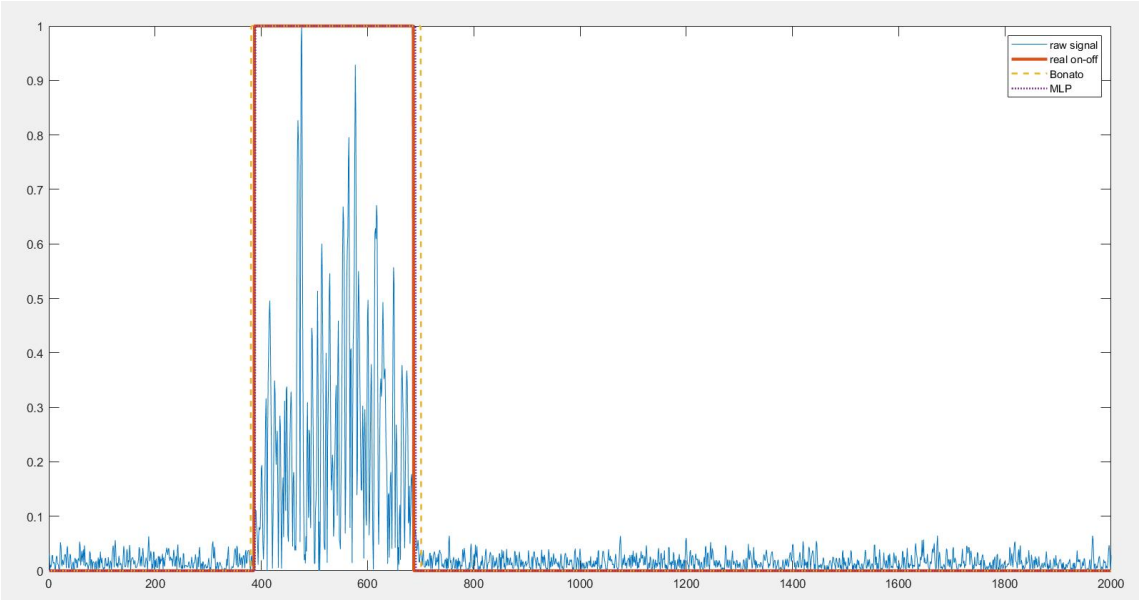


Figura 5.12: *esempio di un segnale con SNR pari a 16dB, sigma = 0.05s, alpha = 1.5*

Nella figura 5.12 è presente un segnale con SNR a 16dB e transizione di durata minore, $\alpha = 1.5$ e $\sigma = 0.05s$: entrambi i metodi hanno delle buone prestazioni, poiché l'SNR è più alto e la transizione è più evidente dato un σ più piccolo.

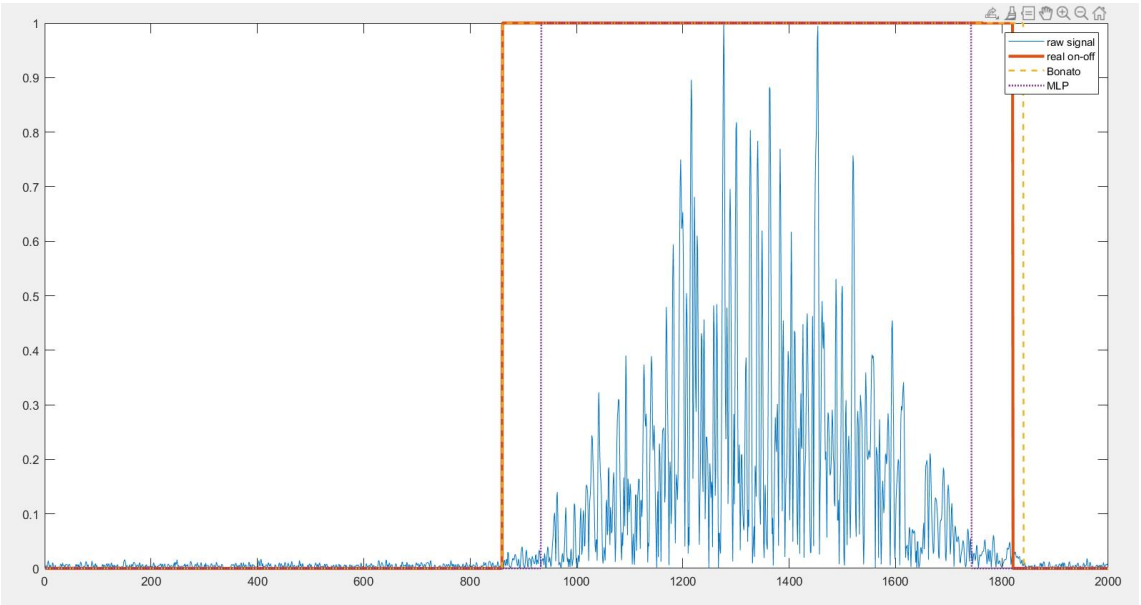


Figura 5.13: *esempio di un segnale con SNR pari a 30dB, sigma = 0.15s, alpha = 1*

L'ultima figura mostra un caso a 30dB dove l'algoritmo di Bonato ha performance migliori, che possono essere spiegate dall'implementazione di una soglia adattiva in funzione della potenza rumorosa.

Infatti, se analizziamo le tabelle in figura 5.14 e 5.15 (pagina successiva), è possibile distinguere dei 'gradini' delle gradazioni di giallo (risultati vicini all'applicabilità) e verde (risultati applicabili) sia per le latency sia per le std, che scendono lungo la diagonale delle tabelle; questo è un comportamento tipico che può essere osservato anche nell'articolo originale e può essere spiegato dall'influenza della forma d'onda nell'individuazione dell'instante di transizione e da questa soglia adattiva. Pertanto, vediamo che Bonato ha per SNR $>$ 20dB performance migliori per le varie combinazioni di sigma e alpha, mentre il MLP non riesce a scendere sotto la soglia dei 10ms per $\alpha = 2.4$.

	SIGMA 0.05 s			
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	10,1	10,9	5,9	30,6
6	6,1	3,3	25,9	18,8
10	10,1	1,4	4,6	17,1
13	16,6	3,6	2,7	12,5
16	1,6	8,8	8,5	6,5
20	6,8	9,7	3,1	10
23	18,2	14,1	3,9	10,4
26	10,1	12,2	10,7	3,8
30	24,6	4,1	5,2	12

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	15,0723	12,8666	5,77062	10,3284
6	2,74773	1,5248	28,252	11,6651
10	14,0285	0,65192	4,00625	16,2612
13	30,4495	2,10357	1,44049	9,31397
16	1,08397	13,6684	7,49166	8,71063
20	9,26418	15,3891	1,59687	13,7523
23	31,1861	16,3378	3,1504	12,9489
26	9,05815	15,3891	17,7926	1,30384
30	41,2817	3,1504	3,17411	14,9708

	SIGMA 0.1 s			
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	4,7	17,4	29,8	72,3
6	4,6	10,3	32,7	36,2
10	11,6	6,1	14,9	29,7
13	4,5	11,1	12,1	24,3
16	8,3	5,4	6,5	12,3
20	12,5	8,9	3,5	14,8
23	22,4	13,8	9,6	11,9
26	8,3	11,7	13,5	14,2
30	11,9	1,9	5	1,9

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	3,59861	19,1944	23,6553	22,6567
6	2,5836	15,234	15,3647	28,2812
10	13,9884	7,56142	9,95866	30,244
13	3,58818	13,3763	10,3646	17,0206
16	7,86289	3,18983	4,37321	9,23038
20	20,5639	10,609	1,90394	12,6521
23	15,7655	13,595	13,5573	17,8515
26	8,47496	20,9929	23,2836	15,7504
30	16,456	2,27486	1,90394	2,24722

	SIGMA 0.15 s			
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	5,1	20,6	55,6	133
6	4	2,9	32,4	55,8
10	12	12,6	20,9	55,1
13	2,1	9,5	5,7	17,1
16	8,9	15,1	9,6	17,7
20	13,9	16,2	2,3	7,9
23	4,6	7,3	1,6	6,5
26	9,2	2,7	13,4	2,3
30	7,3	2,6	2,4	11,2

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	5,61694	13,0643	74,4198	37,3648
6	4,9371	4,42154	27,1901	37,8378
10	16,632	11,7919	22,7552	25,9119
13	1,43178	14,5988	8,96939	23,2847
16	9,11318	19,5109	11,5401	15,0524
20	18,6393	15,6708	1,82346	8,75643
23	4,20416	8,11326	1,71026	11,7898
26	10,8317	3,40221	17,3328	2,16795
30	5,05717	2,96648	3,20936	11,6651

Figura 5.14: *Tabelle che mostrano latency (prima riga) e std (seconda riga) medie sull'istante di onset per l'algoritmo di Bonato; con la gradazione di verde vengono evidenziati i risultati accettabili per l'applicazione*

	SIGMA 0.05 s			
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	6,9	8,9	22,5	39,4
6	7,7	6,5	21,5	29,4
10	7,9	6	2,9	22,3
13	7,6	6,6	5	7,9
16	10,7	11,2	4,6	8,3
20	9,8	11,4	4,3	6,6
23	13,8	15,9	8,7	7,6
26	13,9	15,8	9,3	7
30	17,8	10	7	8,2

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	5,71621	8,80625	17,9722	8,01093
6	6,8884	4,54148	11,7845	14,9139
10	4,3359	4,51387	3,38009	16,0725
13	5,16478	4,8657	4,75657	6,249
16	7,32803	4,39602	3,99061	6,50577
20	6,80625	18,9057	6,45755	4,70903
23	5,10637	13,207	3,05369	4,26321
26	1,74642	14,1713	4,64489	4,84768
30	3,81772	6,77311	6,50961	4,98247

	SIGMA 0.1 s			
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	15,5	5,9	98,8	103,1
6	7,8	9,3	34,4	92,7
10	10,4	4,8	17	45,7
13	9,7	6,9	2,9	49,4
16	11,7	9	9,1	26,7
20	9,7	13,1	2,4	12
23	13,6	9,3	6,8	3,2
26	13,9	12,3	7,7	3,4
30	16,1	15,5	13	7,7

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	21,0416	6,09713	20,6779	28,1988
6	5,66348	12,3875	38,3787	23,6923
10	4,8657	4,16233	25,2958	30,6668
13	5,05717	1,85068	3,48927	21,9186
16	3,05369	3,22102	7,04805	16,9359
20	2,36114	9,46309	3,11047	16,3592
23	2,13307	5,44977	7,46324	3,36526
26	2,98747	5,93296	5,43829	5,27257
30	2,96648	9,37417	2,93684	1,98746

	SIGMA 0.15 s			
	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	15,5	41,7	134,5	106,8
6	1,5	12,7	40,6	57,6
10	4,8	8,9	16,9	61,3
13	6,5	6,5	19,3	56,9
16	9,9	2,3	4,2	38,6
20	7,2	11,8	2,9	11,5
23	12,4	5,4	10,5	1,7
26	10,4	8,9	1,1	1,8
30	12,7	9,1	4	3,2

	alpha = 1	alpha = 1.5	alpha = 2	alpha = 2.4
3	25,8409	38,9946	82,2633	97,6279
6	1,87083	13,1082	37,347	79,0746
10	3,91472	10,5795	16,5733	58,3434
13	3,84057	6,58597	22,0613	57,0015
16	7,10985	3,38378	9,39149	38,9541
20	10,3598	8,18993	5,94138	19,8085
23	4,54698	5,2607	3,40955	3,27109
26	6,73981	5,65022	2,45967	2,68328
30	2,84165	5,27257	5,65685	4,50833

Figura 5.15: *Tabelle che mostrano latency (prima riga) e std (seconda riga) medie sull'instante di OFFSET per l'algoritmo di Bonato;con la gradazione di verde vengono evidenziati i risultati accettabili per l'applicazione*

In conclusione, l'analisi delle latency può essere riassunta dai seguenti punti:

- è stato rilevato un comportamento più stabile del MLP tra onset e offset, dove per quest'ultimo le performance risultano migliori sia nel confronto tra latency in funzione dei vari alpha e sigma sia in funzione di SNR crescenti
- il MLP presenta un comportamento migliore rispetto all'algoritmo di Bonato per bassi SNR
- le performance del double threshold sull'onset sono leggermente superiori al MLP
- le performance del double threshold riescono ad scendere sotto la soglia di applicabilità per tutte le combinazioni di alpha e sigma considerata per SNR alti, mentre il MLP non ha prestazioni ottimali per alpha = 2.4

5.3 Confronto con un simile approccio

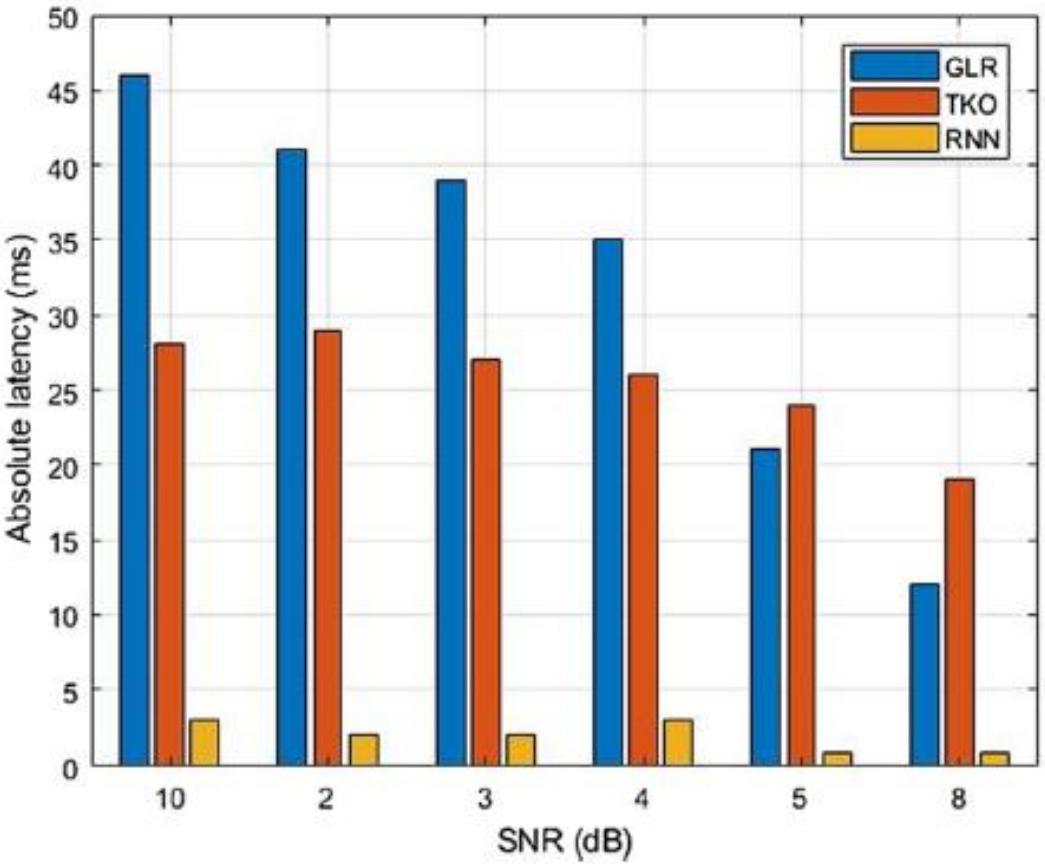


Figura 5.16: *Confronto tra più metodi preso dall'articolo [5], che utilizza una rete ricorsiva RNN per il problema di individuazione di istanti di onset e offset*

Il problema di individuazione dell'attivazione muscolare è stato affrontato tramite approccio deep learning solo in un altro articolo trovato durante la ricerca bibliografica[5]. Il metodo utilizzato è simile a questo proposto perché si tratta di un'estrazione di features da segnali sintetici tramite un pre-processamento e un successivo utilizzo di una rete neurale per il problema di classificazione, ma la tecnica di generazione del segnale, il pre-processamento e la rete neurale utilizzata nell'articolo sono diversi.

Per quanto riguarda la generazione del segnale,gli autori di [5] utilizzano una tecnica con cui vengono simulati i segnali di più MUAP(Motor Unit Action Potential) per poi essere sommati nel segnale finale, simulando quindi il modo con cui è formato il segnale elettromiografico di superficie. Non è riportato il numero di segnali creati per formare il dataset, ma solo che la rete neurale è addestrata con batch di 100 segnali alla volta lunghi 2s con SNR diverso; non è riportata la frequenza di campionamento. Le attivazioni muscolari iniziano tra 0 e 0.3 s e hanno una durata di 0.2s +- 0.1s; questa è una differenza fondamentale rispetto ai segnali che vengono creati in questa tesi; infatti, i segnali con cui

la rete è stata addestrata, oltre a poter avere un'attivazione che inizia e finisce in un punto qualsiasi all'interno del segnale, hanno un range di attivazione molto più grande: tra un minimo di 0.1 s (nel caso di $\sigma = 0.05$ s e $\alpha = 1$) e un massimo di 0.72 s (nel caso di $\sigma = 0.15$ s e $\alpha = 2.4$).

Per quanto riguarda il pre-processamento, viene utilizzato lo spettrogramma ed è utilizzato interamente come input di una rete neurale ricorsiva (RNN); l'intuizione dietro quest'articolo sta nel trasportare nel campo del processamento di segnali neuromuscolari tecniche di processamento tipiche del campo dello speech processing, che funzionano bene poiché le RNN possono fare delle predizioni basate su un'osservazione più lunga nel tempo; infatti le RNN per processare l'output al'istante t mantengono memoria degli input fino all'istante t-1. Non viene riportata la specifica tipologia di rete ricorsiva utilizzata, ma viene indicato che le attivazioni utilizzate siano delle ReLU per gli strati intermedi ed una sigmoide per l'output. Non vengono riportati accuracy, precision, recall e f1score,

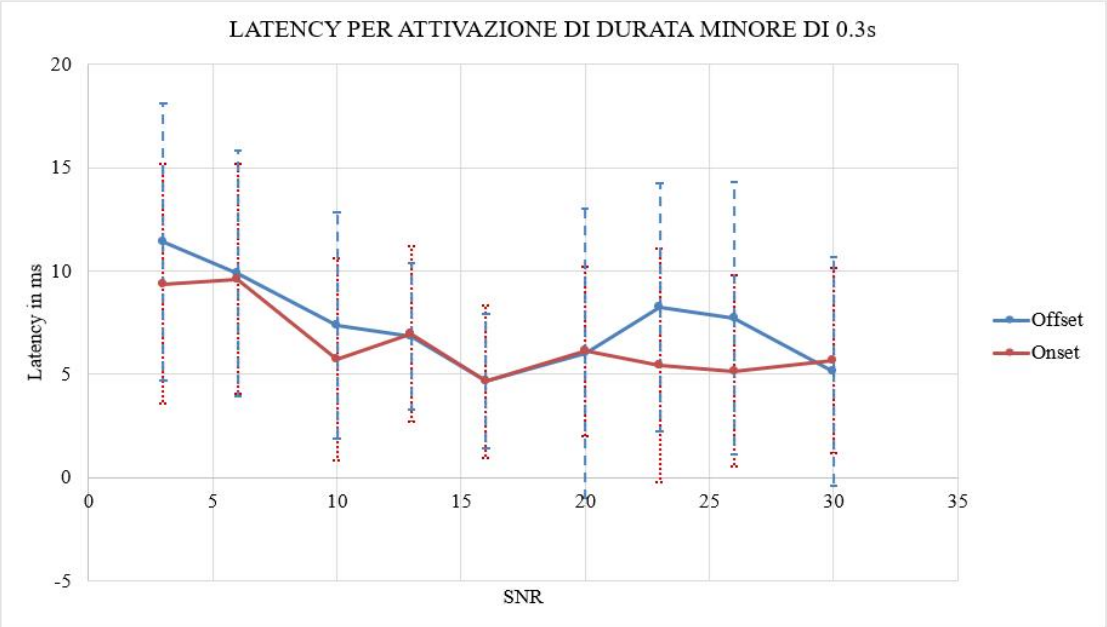


Figura 5.17: *grafico che rappresenta le latency medie su attivazioni da 0.1 a 0.3 s con il relativo errore per il modello MLP di questa tesi*

ma viene riportato un grafico che confronta le latency ottenute con Generalize Likelihood Ratio e Threshold method con TKEO [Fig 5.16 a inizio sezione]. I risultati mostrati sono tutte latency minori di 5 ms per SNR minori di 10 dB; se volessimo confrontare i risultati proposti in questa tesi con questo articolo potremmo realmente fare solo un confronto qualitativo non avendo chiaramente il modello che hanno utilizzato; possiamo però dire che osservando la figura 5.17 i risultati tramite MLP sono generalmente peggiori del modello RNN per transizioni che durano al massimo 0.3s. Tuttavia, bisogna fare delle cosiderazioni:

1. i testset sono diversi; la generazione dei segnali tramite MUAP sintetiche potrebbe generare segnali più semplici rispetto ai segnali tramite inviluppo, alcuni dei quali, come già scritto precedentemente, hanno delle forme d'onda che rendono difficile la task di classificazione binaria

2. la rete neurale proposta nell'articolo è addestrata solo su attivazioni che durano tra gli 0.1 e 0.3 s. Si potrebbe quindi dire che è specializzata nelle attivazioni più basse. Infatti, l'articolo si concentra su quel tipo di attivazioni per poi utilizzare il modello su un sEMG reale di un vasto laterale, che ha un pattern di attivazione in quella durata

Comunque, le latency che risultano dal modello MLP per attivazioni basse sono mediamente minori di 10ms e hanno mediamente std minore di 15ms quindi, anche se non minori di 5ms, rientrano nelle soglie di utilizzo clinico e di ricerca.

Capitolo 6

Conclusione e Sviluppi futuri

In conclusione, in questa tesi è stato proposto un approccio innovativo al problema di identificazione di onset e offset nel segnale sEMG testato su un dataset simulato e confrontato con una il metodo double threshold di Bonato. I risultati ottenuti sono soddisfacenti in quanto per $\alpha <= 1.5$ rispettano i limiti per l'applicazione in ricerca per ogni sigma ed SNR, mentre per $\alpha = 2$ lo stesso può essere detto per SNR $>$ 10dB. Le problematiche principali si presentano per quelle forme d'onda con α =2.4, che presentano mediamente latency di 16.79 +- 10.04 ($\sigma = 0.05s$), 32.51 +- 17.39($\sigma = 0.10s$) e 41.52 +- 32.62($\sigma = 0.15s$) e per cui le latency non scendono mai sotto la soglia dei 10ms.
Comunque, le performance peggiori si hanno per attivazioni che durano 0.72 s ($\alpha = 2.4 e \sigma = 0.15s$), mentre nell'applicazione reale le attivazioni sono presenti al massimo per il 50-55% del gait. Inoltre, il confronto con il metodo double threshold suggerisce come questa metodica sia più stabile al variare dell'SNR e generalmente migliore, permettendone un possibile utilizzo in ricerca.

Allo scopo di tracciare possibili direzioni di ricerca future, di seguito si riportano alcuni aspetti che potrebbero essere investigati per migliorare l'approccio e i relativi risultati.

1. Miglior tuning degli iperparametri
   Le performance potrebbero essere migliorate utilizzando la rete stessa, ma migliorando gli iperparametri utilizzati, come ad esempio il numero di epoche, il learning rate, la batch size e l'optimizer.

2. Pre-processamento
   Le possibili strade da indagare per quanto riduarda il pre-processamento sono molteplici e la configurazione di pre-processamenti qui proposta potrebbe non essere la migliore per la task di classificazione; ad esempio, la concatenazione di RMSV30 e Inviluppo Lineare potrebbe essere ridondate. Altre idee potrebbero essere:

   - l'introduzione di filtraggi, come un passa banda tra 10 Hz e 500 Hz, tipico della lavorazione del segnale neuromuscolare, o filtraggi più invasivi volti a rimuovere

quanto più rumore possibile. Infatti, la rete ha lavorato con input in entrata non filtrati e si potrebbe dire che ha dovuto 'filtrare' da sola il rumore con l'addestramento.

- l'utilizzo del TKEO [13] [12], pre-processamento a cui si è accennato durante l'introduzione
- lavorare solo con certe bande di frequenza dello scalogramma: sappiamo che infatti la maggior parte della potenza nei segnali sEMG è concentrata nell'intervallo tra i 100 e i 200 Hz e un'idea potrebbe essere quella di individuare quindi il contenuto in frequenza che è più rilevante per l'individuazione degli onsets e offsets.
- cambiare la mother wavelet dello scalogramma; in questa tesi è stata utilizzata la wavelet di default, cioè la 'Morse'

3. Preparazione dei dati
   La finestra da 10 campioni potrebbe essere allargata in modo da avere più informazioni.

4. Architettura di rete
   L'archittettura di rete utilizzata è la più semplice che esiste, cioè un multiperceptrone con 32 neuroni; sarebbe possibile quindi cambiare il numero dei neuroni del singolo hidden layer o aumentare il numero di layer per testare le accuracies di modelli più complessi. L'altra opzione sarebbe l'utilizzo di una rete ricorsiva RNN come nell'articolo [5], che potrebbe portare ad un aumento delle performance.Infatti, queste tipologie di reti sono molto utilizzate per la lavorazione di segnali perchè particolarmente adatte a trovare pattern in segnali che si sviluppano nel tempo.

5. Migliorare e allargare il dataset
   Un dataset di train più grande dovrebbe portare ad un miglioramento delle performance. Inoltre, si potrebbe sperimentare con diverse tipologie di dataset con distribuzioni di SNR diverso. Infatti, l'articolo [5] ha trovato che addestrando la rete con SNR minori di 3dB le loro performance siano migliorate. Questo viene da loro spiegato tramite l'intuizione per cui la rete impara a distinguere meglio tra rumore e segnale 'vero' in segnali dove entrambe le componenti sono presenti in buona quantità.

   Questo potrebbe essere anche spiegato con il concetto di Hard Negative e Hard Positive Mining; nei problemi di object detection binaria, infatti, è solitamente necessario introdurre all'interno del dataset i casi più difficili da classificare per migliorare la performance della rete. Ad esempio nell'articolo [10], dove c'è una prima applicazione di face detection tramite MLP, viene per la prima volta impiegata la

tecnica di Hard Negative Mining, che consiste brevemente in tre punti che vengono iterati fino a convergenza delle performance della rete:

- Addestramento della rete su un trainset piccolo e incompleto
- Test su samples mai visti dalla rete; selezione di quelli che vengono classificati come falsi positivi
- Aggiornamento del trainset e iterazione della procedura fino a convergenza

Questo procedimento permette di includere nel trainset gli esempi che la rete classificherebbe normalmente come falsi positivi e lo stesso può essere fatto per i falsi negativi.
Nel caso dei segnali non è necessario un reale mining dei samples, perchè conosciamo già quali siano gli esempi più difficili da classificare, cioè gli esempi a basso SNR. Detto questo, è chiaro che quindi esista un insieme di SNR per cui le performance di una rete possano essere raffinate e che per SNR troppo alti, la rete impara poco, mentre per SNR troppo bassi il rumore sovrasta il segnale impedendo l'apprendimento.

6. Allargare il testset
   Per rendere più generali le considerazioni fatte nella fase di testing il numero di segnali utilizzati dovrebbe essere molto più alto; ad esempio in [1] vengono utilizzati 10800 segnali, 100 per ogni combinazioni di SNR, alpha e sigma. Inoltre, è necessario valutare se la rete performa bene anche per molteplici attivazioni all'interno di un gait.

7. Post-processamento migliore
   Questa tesi non ha rivolto una particolare attenzione verso il post-processamento, ma più verso la 'detection unit'; adottare post-processamenti più sofisticati potrebbe portare ad un perfezionamento dei risultati, in particolare per snr bassi e α o σ alti dove il post-processing produce l'effetto peggiorativo maggiore.

Bibliografia

[1] Knaflitz M. Bonato P., D'Alessio T. A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait. *IEEE Trans. Biomed. Eng. 45(3), pp. 287-99*, 1998.

[2] Radka Bačáková Bronislav Kračmar Lenka Satrapová Petr Novotný Daniel Spulák, Roman Cmejla. Muscle activity detection in electromyograms recorded during periodic movements. *Computers in Biology and Medicine 47, pp 93-103*, 2014.

[3] Martin D. Werner W. Gerhard S., Claus F. Onset detection in surface electromyography signals: A systematic comparison of methods. *EURASIP Journal on Applied Signal Processing 2, pp 67-81*, 2001.

[4] Tommaso D'Alessio Giuseppe Vannozzi, Silvia Conforto. Automatic detection of surface emg activation timing using a wavelet transform based method. *Journal of Electromyography and Kinesiology 20 767–772*, 2010.

[5] Ahmed Abotabl Iman Akef Khowailed. Neural muscle activation detection: A deep learning approach using surface electromyography. *Journal of Biomechanics, vol 95*, 2019.

[6] A. Van Der Bilt J.H. Abbink and H.W. Van Der Glas. Detection of onset and termination of muscle activity in surface electromyograms. *Journal of Oral Rehabilitation vol, 25; pp 365–369*, 1998.

[7] Courtney A. Haynes Matthew S. Tenan, Andrew J. Tweedell. Analysis of statistical and standard algorithms for detecting muscle onset with surface electromiography. *PLOS ONE 12(5)*, 2017.

[8] Bang H. Bui Paul W. Hodges. A comparison of computer-based methods for the determination of onset of msucle contraction using electromiography. *Electroencephalography and clinical Neurophysiology 101, pp 511-519*, 1996.

[9] Jaquelin Perry. *GAIT ANALYSIS, Normal and Pathological Function.* 1992.

[10] Tomaso A. Poggio. Example based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[11] J. Harlaar G.Zilvold R.F.M Kleissen, J.H. Buurke. Electromyography in the biomechanical analysis of human movement and its clinical application. *Gait and Posture 8, pp. 143-158*, 1998.

[12] Ken Steinweg Paul DeVita Tibor Hortoba´gyi Stanislaw Solnik, Patrick Rider. Teager–kaiser energy operator signal conditioning improves emg onset detection. *Eur J Appl Physiol 110:489–498*, 2010.

[13] Patrick Rider Benjamin Long Stanisław Solnik, Paul DeVita and Tibor Hortobágyi. Teaset-kaiser operatore inproves the accuracy of emg onset detection indipendent of signal-to-noise ratio. *Acta Bioeng Biomech.; 10(2): 65–68.*, 2008.

[14] F. Stulen and C. J. De Luca. Frequency parameters of the myoelectricsignal as a measure of muscle conduction velocity. *IEEE Trans.Biomed. Eng., vol. 28, pp. 512–523*, 1981.

Ringraziamenti

I miei ringraziamenti vanno al Prof. Di Nardo e al Prof. Morbidoni, rispettivamente relatore e correlatore di tesi, i quali mi hanno seguito e guidato durate questo percorso, pur con le limitazioni che questo periodo di pandemia ci ha posto. In particolare, ringrazio il prof. Morbidoni per il suo fondamentale aiuto nella stesura di questa tesi.

Alla mia famiglia e ai miei amici, compagni di studio e di serata, che sono stati parte integrante ed essenziale di questi tre anni. Grazie.

A Libera, che mi ha sopportato nei momenti in cui, per l'eccessivo interesse rivolto verso questo progetto, sono stato distratto, intrattabile o asociale, continuamente chino sul computer con lo sguardo fisso sul monitor, aspettando dei risultati che potessero soddisfarmi. Grazie per essere stata mia complice.

Al mio computer, che poche volte ho spento negli ultimi mesi lasciandolo lavorare anche di notte per avere dei risultati al mattino successivo. Grazie.